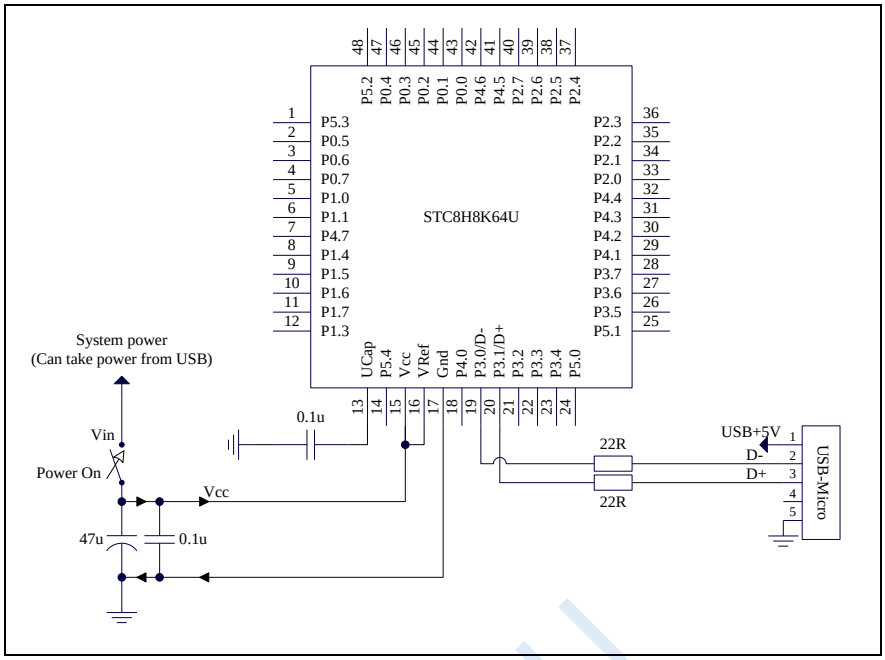
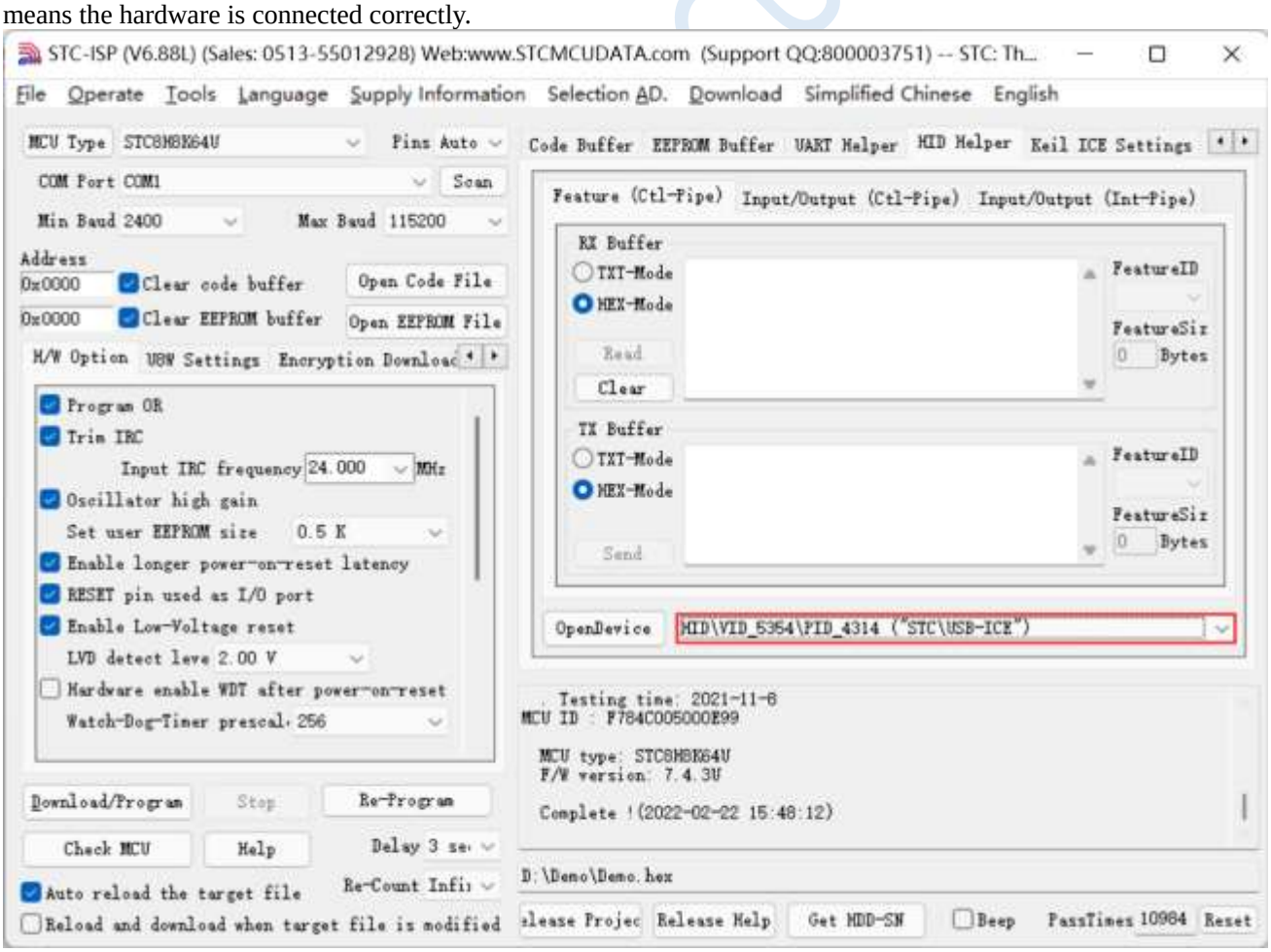
3. Referring to the connection method in the figure below, connect the target MCU with the emulated chip to the USB port of the computer.

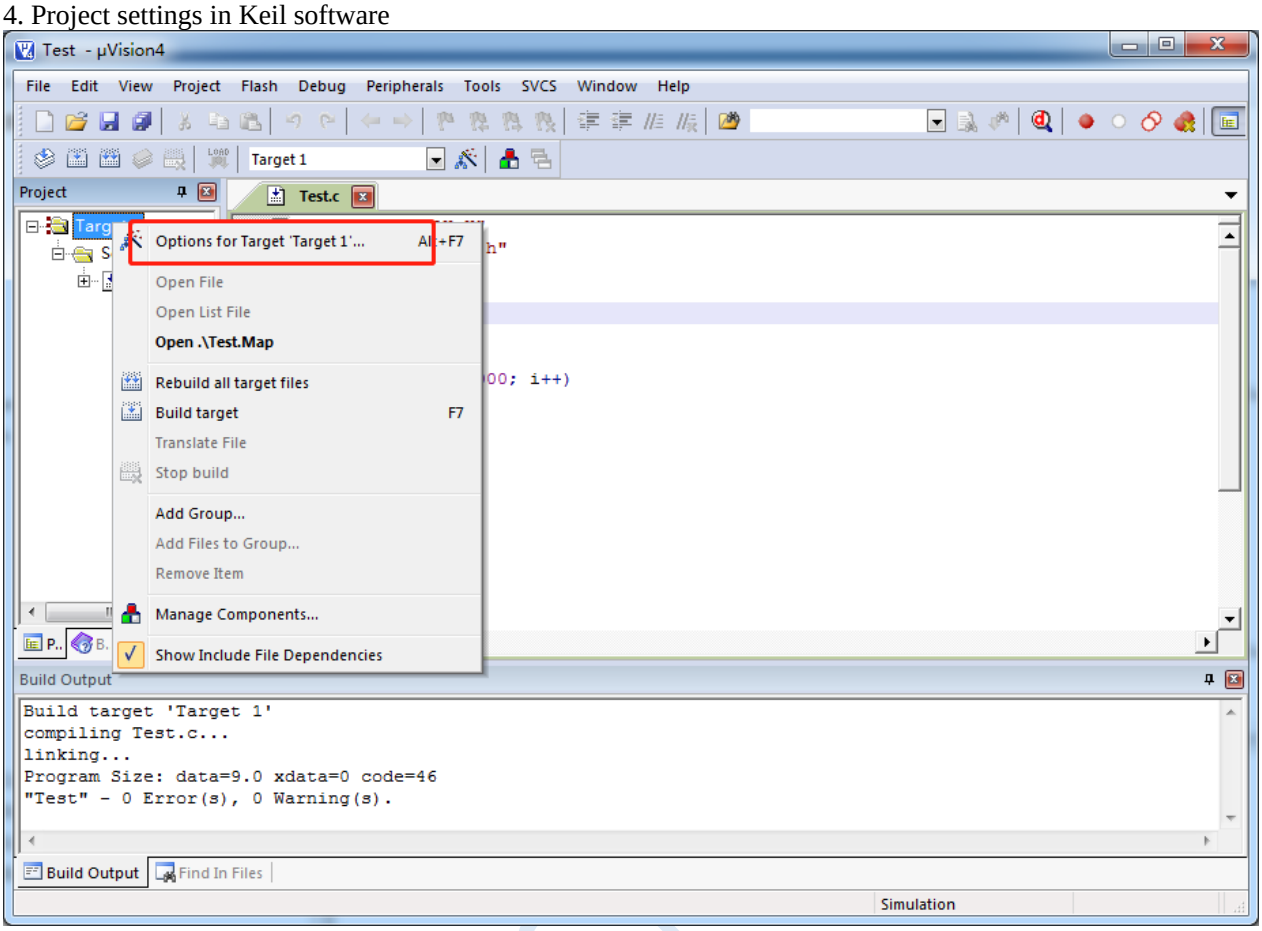



When the "STC\USB-ICE" device can be correctly displayed in the HID assistant in the downloaded software, it means the hardware is connected correctly.

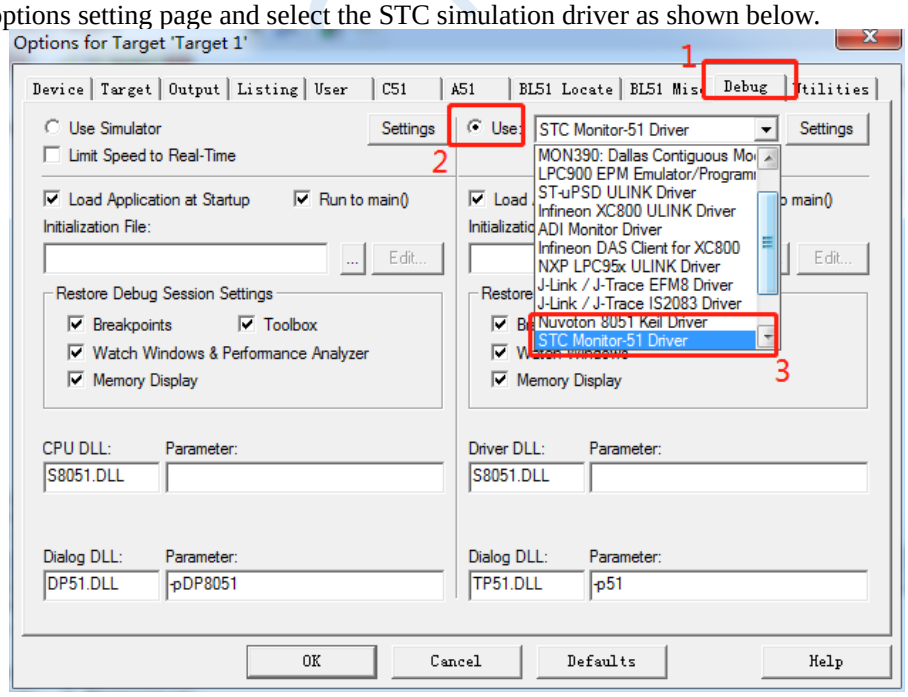



Note: When the displayed device name is "STC\USB-ISP", it means that the target chip is in the USB-ISP download mode, please make sure that the P3.2 port is at a high level and reconnect to the USB.

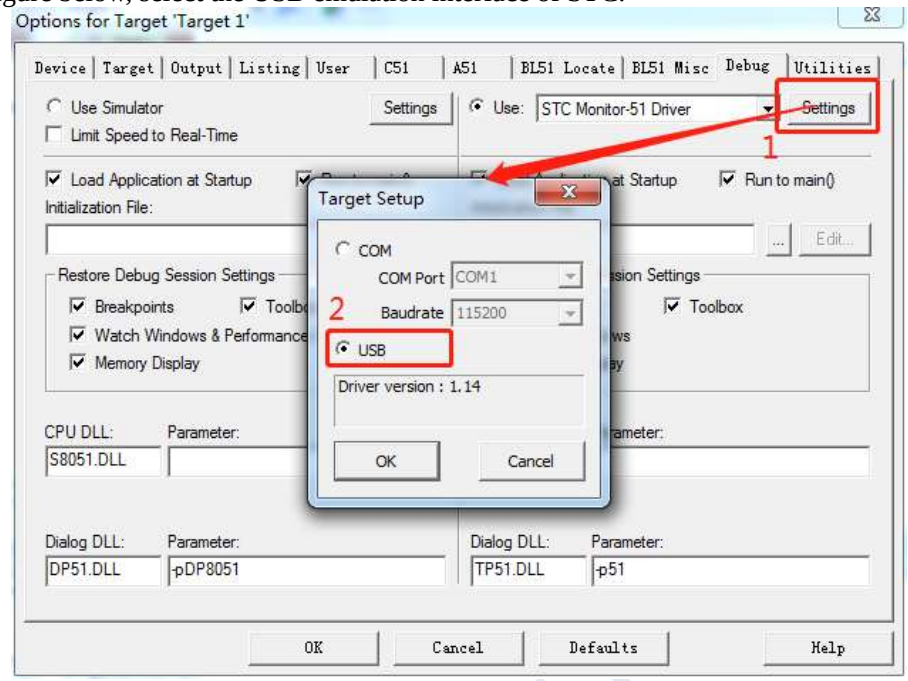
4. Project settings in Keil software




Open the project options setting page and select the STC simulation driver as shown below.

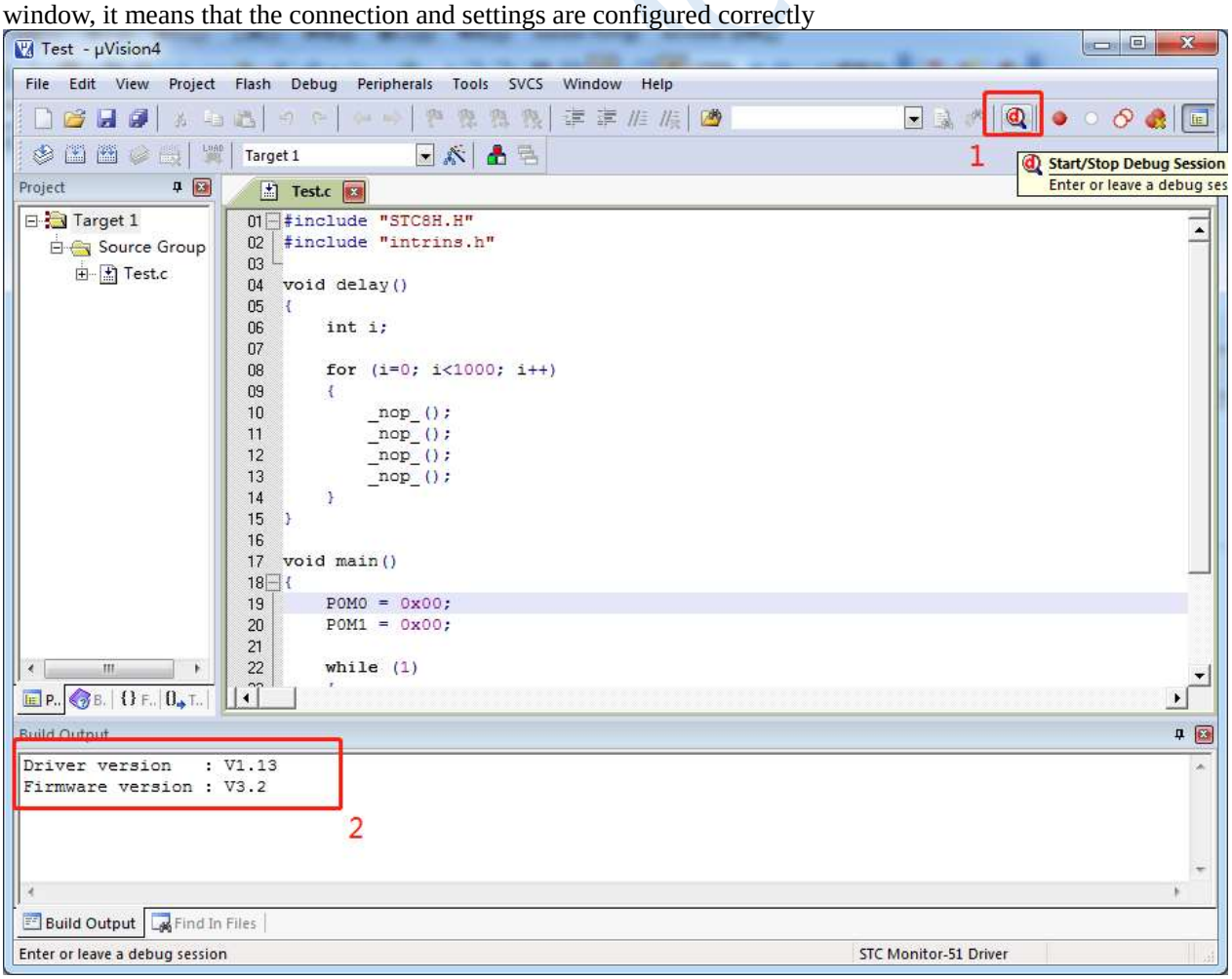


As shown in the figure below, select the USB emulation interface of STC.

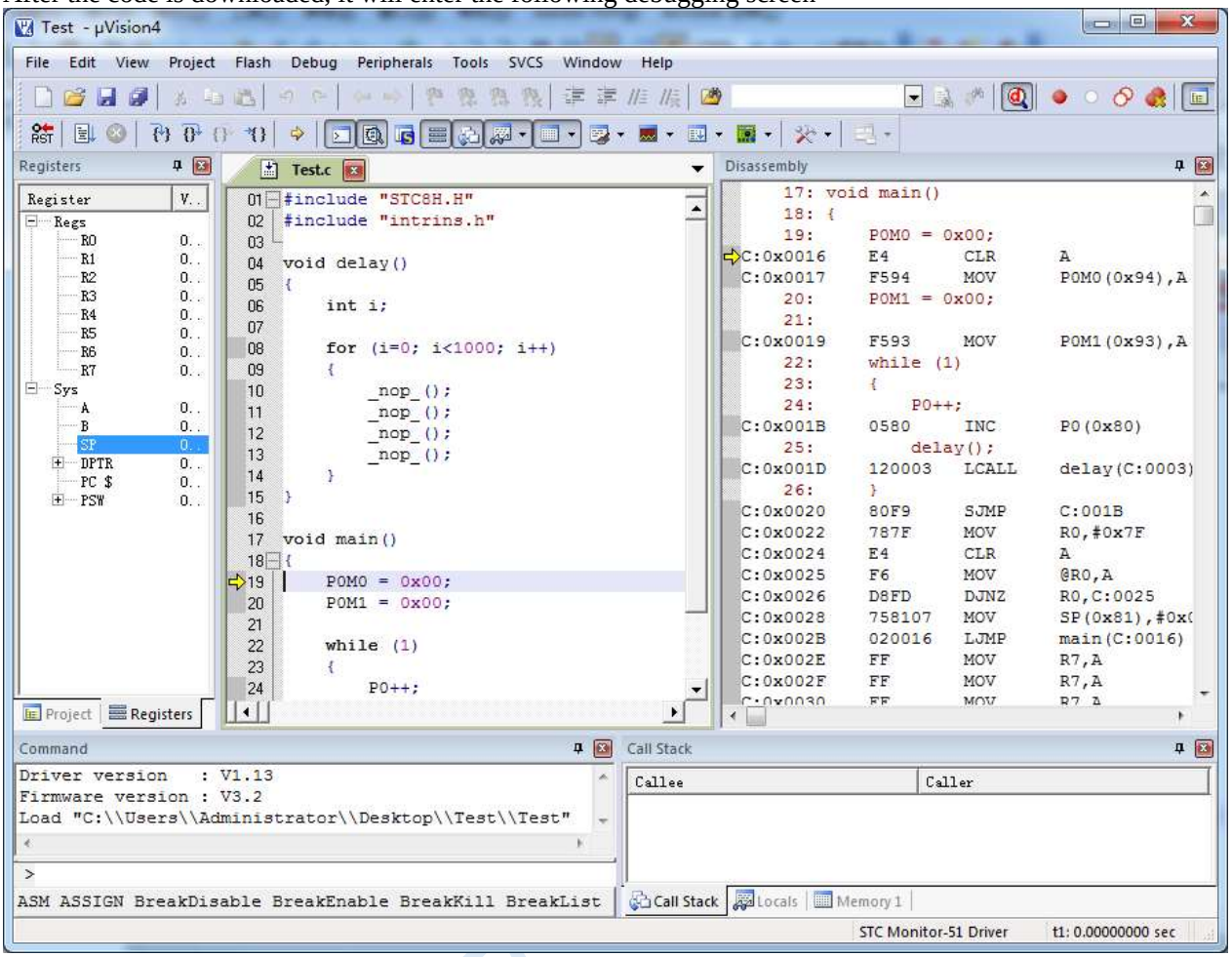



5. After the setting is completed, start the simulation

Click the Start Simulation button in the Keil software, if the version number is displayed correctly in the output window, it means that the connection and settings are configured correctly

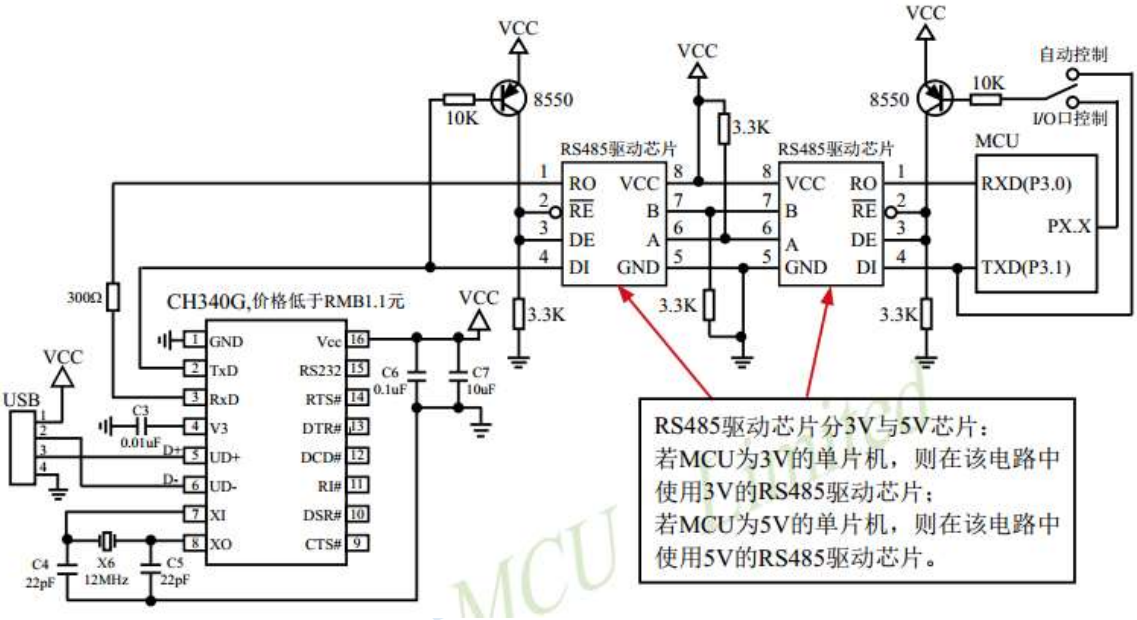


After the code is downloaded, it will enter the following debugging screen

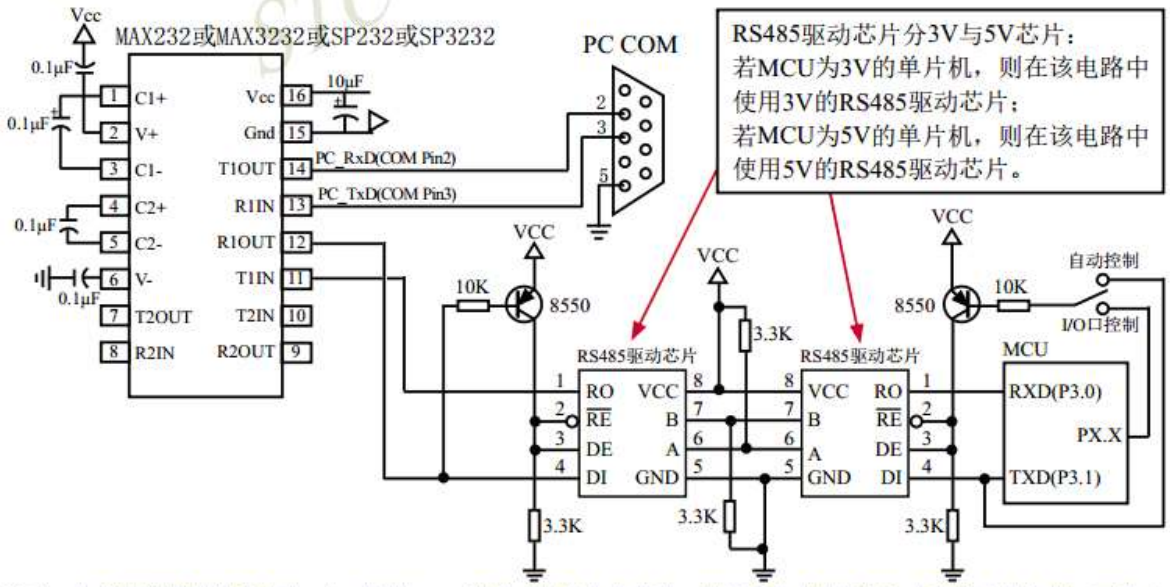


# Appendix G RS485 Automatic control or I/O port control circuit diagram

1. Use the USB to serial port to connect the computer's RS485 to control the download circuit diagram (automatic control or I/O port control)




2. Use RS232 to serial port to connect the computer's RS485 to control the download circuit diagram (automatic control or I/O port control)




注意：如果要设置单片机某个I/O口控制RS485发送或接收命令有效，则必须将单片机焊入电路板之前先用U8下载工具结合电脑ISP软件对该单片机进行“RS485控制”设置并烧录一下（如上节所述），否则将单片机实现不了RS485控制功能。

建议用户将本节所述“RS485控制下载线路图(自动控制或I/O口控制)”设计到您的用户板上

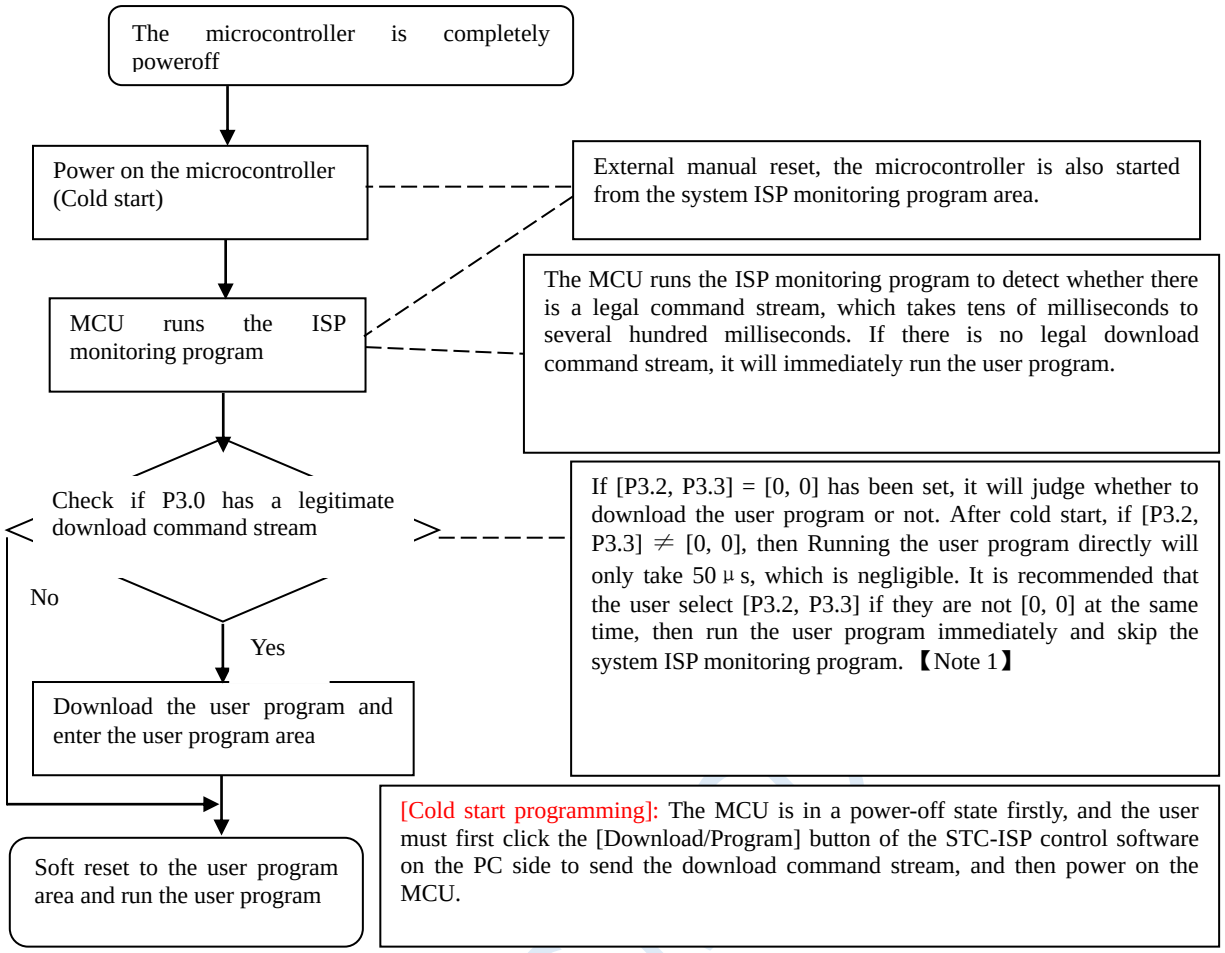
# Appendix H STC tool instruction manual

## H.1 Overview

U8W/U8W-Mini is a series of programming tools that integrates online download and offline download. STC Universal USB to Serial Port Tool is a programming tool that supports online download and online simulation.

| Tool type | Online Download | Offline download | Burner download | Online simulation | Price(CNY) |
|---|---|---|---|---|---|
| U8W | support | support | support | Need to set pass-through mode | 100 yuan |
| U8W-Mini | support | support | not support | Need to set pass-through mode | 50 yuan |
| Universal USB to serial port | support | not support | not support | support | 30 yuan |

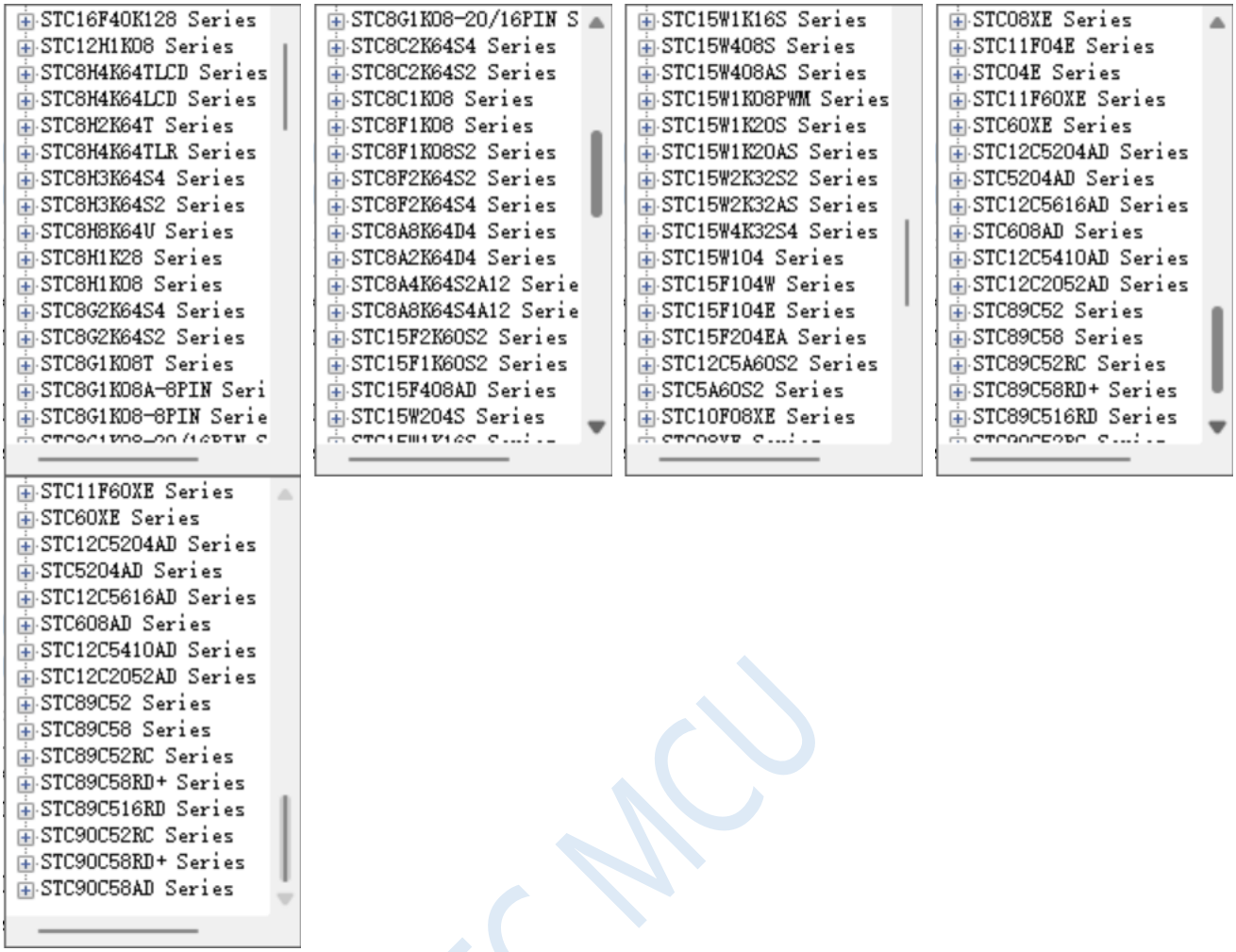
## H.2 System Programmable (ISP) Process Description




Note: Because [P3.0, P3.1] is used for download/simulation (download/simulation interface is only available [P3.0, P3.1]), it is recommended that users put serial port 1 on P3.6/P3.7 Or P1.6/P1.7, if the user does not want to switch and insists on using P3.0/P3.1 to work or communicate as the serial port 1, be sure to check the "Next cold start, when downloading the program, The program can be downloaded only when P3.2/P3.3 is 0/0". 【Note 1】
[Note 1]: The programming protection pins of STC15, STC8 and later new chips are P3.2/P3.3, and the programming protection pins of earlier chips are P1.0/P1.1.

## H.3 USB type online/offline download tool U8W/U8W-Mini

The application range of U8W/U8W-Min can support all current MCU series of STC, and the Flash program space and EEPROM data space are not restricted. Support includes the following and upcoming STC full series chips:



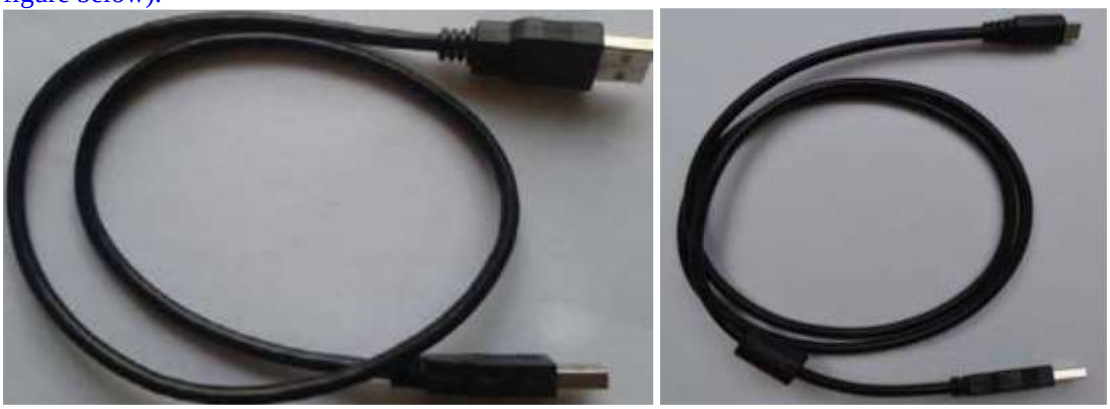

The offline download tool can be used for downloading without the computer, and can be used for mass production and remote upgrades. The offline download board can support multiple functions such as automatic increment, download limit, and encrypted transmission of user programs.

The following picture shows the front and back views of U8W tools and the front and back views of U8W-Mini:


In addition, some wires and tools are used together as follows, such as:

(1) Two-end male USB cable (shown on the left in the figure below) and USB-Micro cable (shown on the right in the

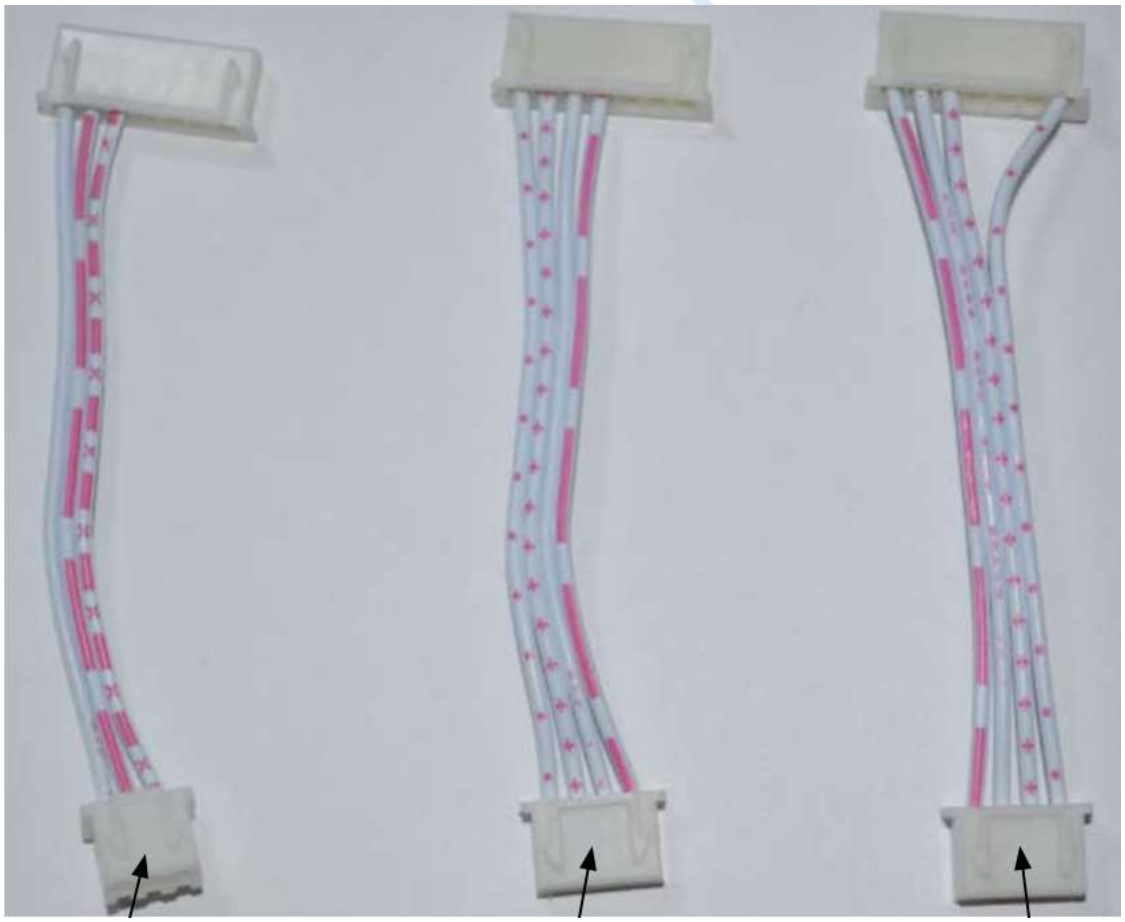
figure below):



Note: This USB cable is a USB enhanced cable specially customized by our company, which can ensure that the download can be successful when directly powered by USB. On the market, some relatively low-quality two-end male USB cables have too much internal resistance and cause a large voltage drop (for example, the voltage of the USB when it is empty is about 5.0V. When using a low-quality USB cable to connect U8W/U8W-Mini/U8 /U8-Mini, the voltage to our download board may drop to 4.2V or lower, causing the chip to be in a reset state and fail to download successfully).

(2) The download cable connecting U8W/U8W-Mini to the user system (ie the connecting cable between U8W/U8W-Mini and the target MCU on the user board), such as

As shown below:



U8W/U8W-Mini and
User systems are independent
Power supply cable

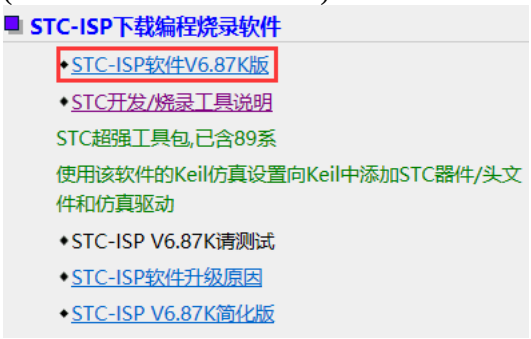
U8W/U8W-Mini to
The user's system power supply
wiring

User system gives
U8W/U8W-Mini
Power supply cable

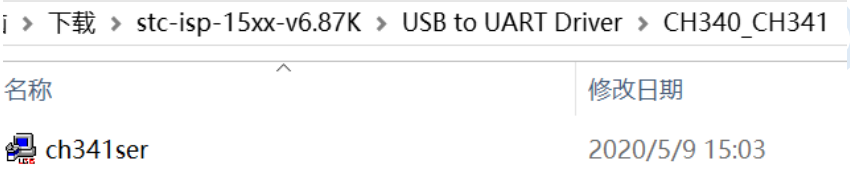
# H3.1 Install U8W/U8W-Mini driver

The U8W/U8W-Mini download board uses a CH340 USB-to-serial universal chip. This saves the trouble that some computers without a serial port must buy a USB to serial port tool to download. But CH340 is the same as other USB-to-serial tools, the driver must be installed before use.

Obtain the driver by downloading the STC-ISP software package
The following is the download location of the STC-ISP software package provided on the STC official website (www.STCMCUDATA.com):




After downloading, decompress, the path of CH340 driver installation package is stc-isp-15xx-v6.87K\USB to UART Driver\CH340_CH341:

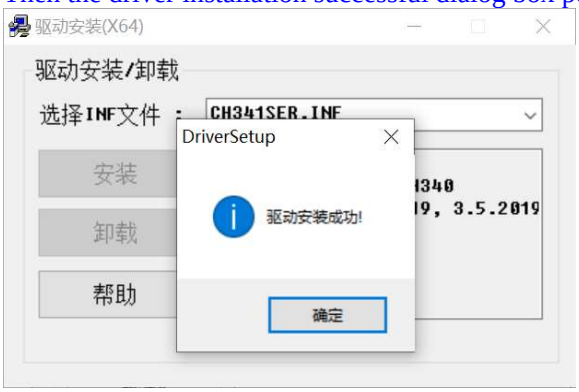



Download the driver manually through STC's official website or in the latest STC-ISP download software
Download the driver manually on the official website of STC or in the latest STC-ISP download software. The download link of the driver is: U8 programmer USB to serial port driver (http://www.stcmcu.com/STCISP/CH341SER.exe ). The driver address on the website and STC-ISP download software is shown in the figure below:

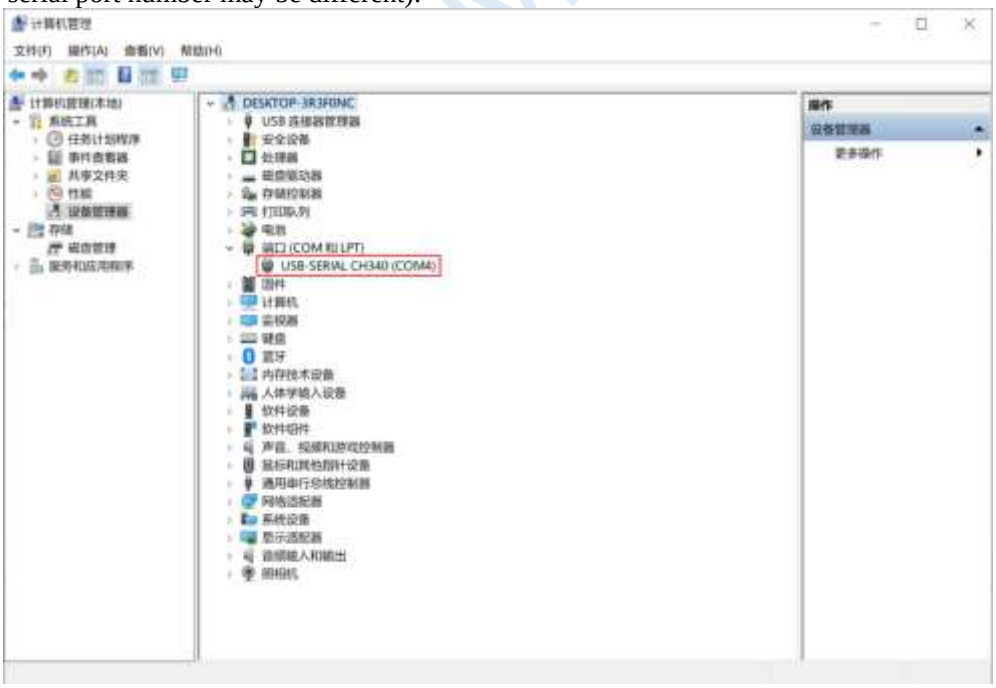

## Install U8W/U8W-Mini driver

After the driver is downloaded to the machine, double-click the executable program and run it. The interface shown in the figure below appears, click the "Install" button to start the automatic driver installation:

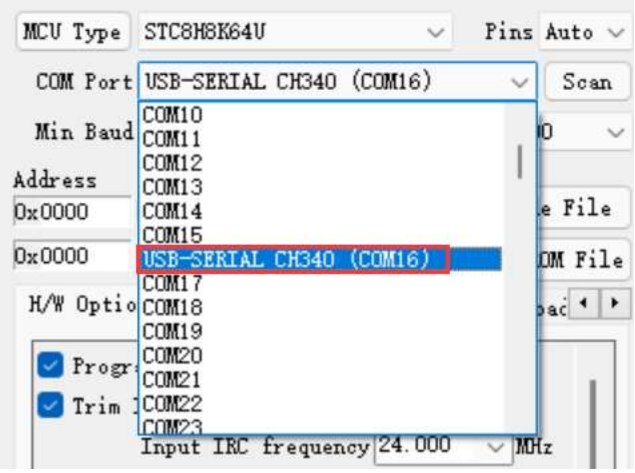

Then the driver installation successful dialog box pops up, click the "OK" button to complete the installation:




Then use the USB cable provided by STC to connect the U8W/U8W-Mini download board to the computer, open the device manager of the computer, and under the port device category, if there is a device similar to "USB-SERIAL CH340 (COMx)", it means U8W/U8W-Mini can be used normally. As shown in the figure below (different computers, the serial port number may be different):

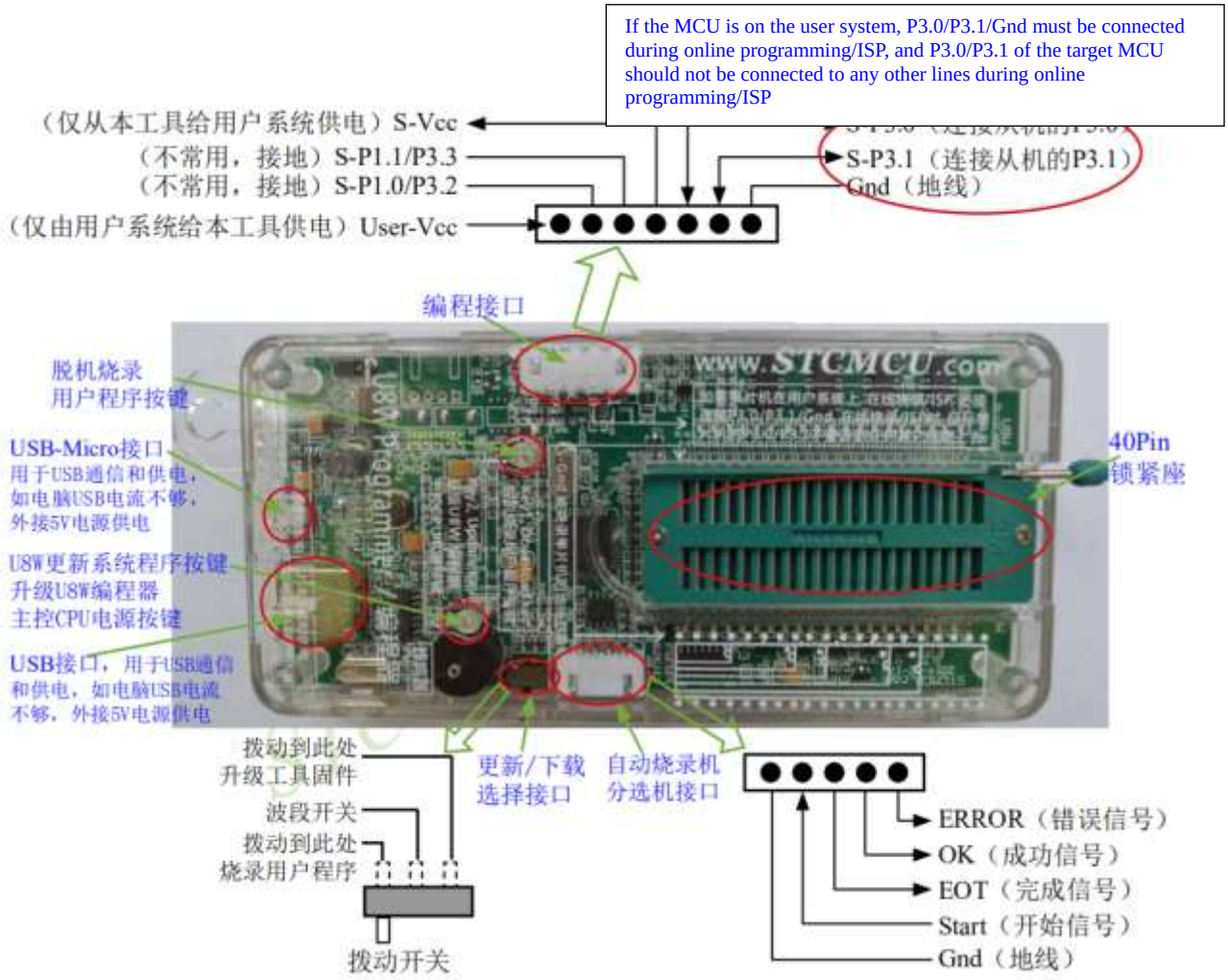



Note: When using STC-ISP to download software later, the selected serial port number must be the corresponding serial port number, as shown in the figure below:




# H3.2 U8W function introduction

The main interfaces and functions of U8W tools are described in detail below:



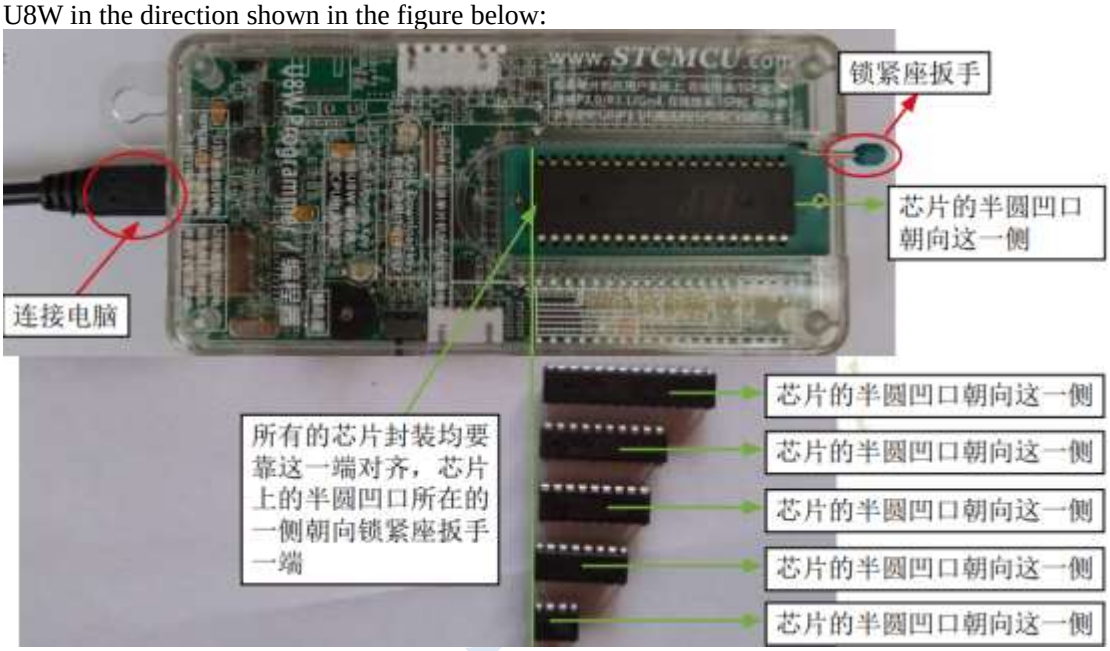

Programming interface: Use different download cables to connect the U8W download board and the user system according to different power supply methods.

U8W update system program button: used to update U8W tools. When there is a new version of U8W firmware, you need to press this button to update the U8W main control chip (note: you must first set the toggle switch on the update/download selection interface Toggle to upgrade tool firmware).

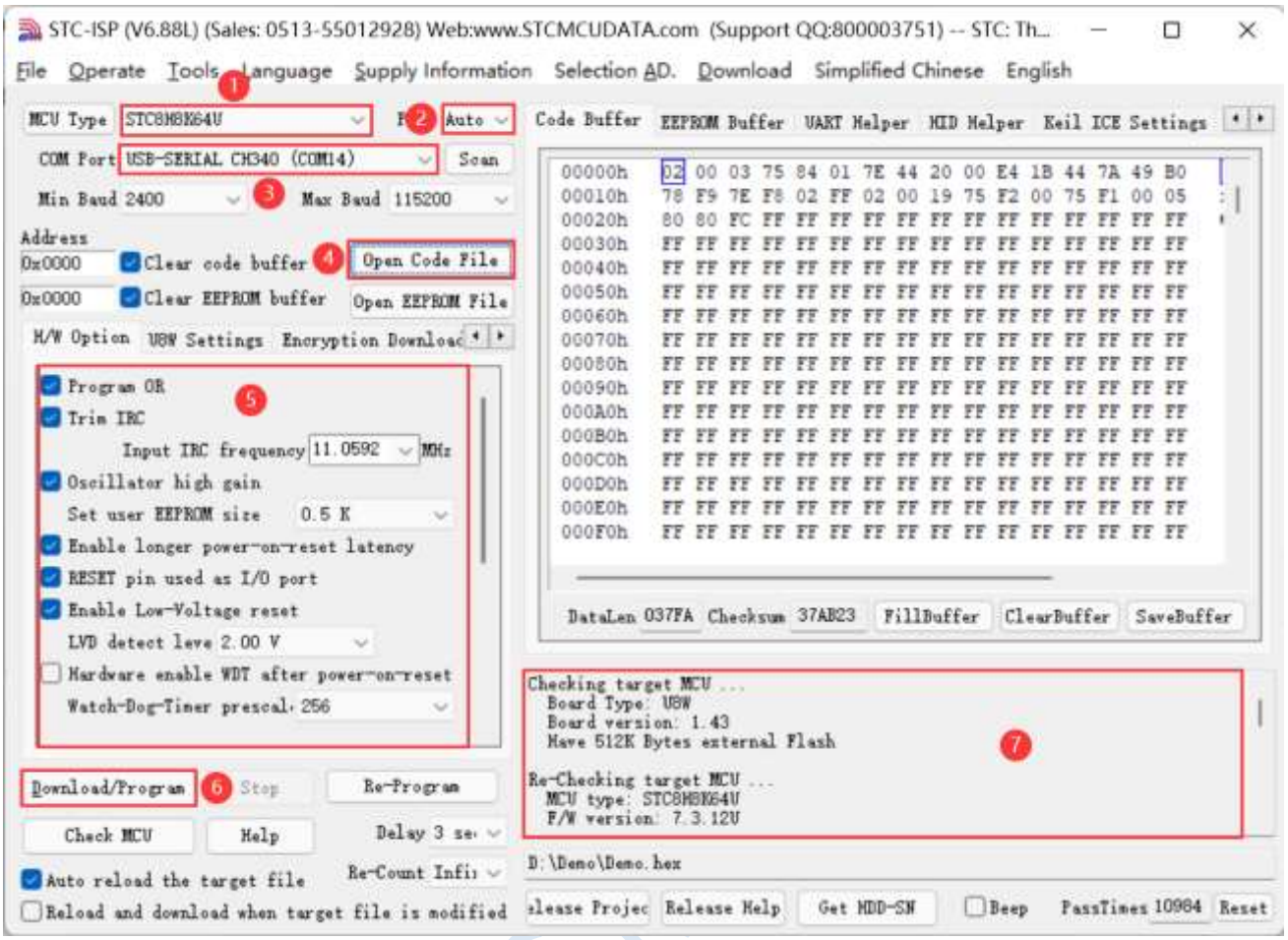
Offline download user program button: Start offline download button. First, the PC downloads the offline code to the U8W board, and then uses the download cable to connect the user system to the U8W, and then press this button to start the offline download (the user code will also start to download every time the power is turned on).
Update/download selection interface: When you need to upgrade the underlying firmware of U8W, you need to switch this toggle switch to the firmware upgrade tool. When you need to program the target chip through U8W, you need to turn the toggle switch to Burn the user program.
(Please refer to the figure above for the connection of the toggle switch)
Automatic burner/sorter interface: It is a control interface used to control the automatic burner/sorter for automatic production.

## H3.3 U8W online download instructions

The target chip is installed on the U8W locking base and connected to the computer by U8W for online download
First use the USB cable provided by STC to connect the U8W to the computer, and then install the target MCU on the U8W in the direction shown in the figure below:




Then use STC-ISP to download the software to download the program, the steps are as follows:




1 Select the MCU model;
2 Select the number of pins. When the chip is directly installed on the U8W to download, be sure to select the correct number of pins, otherwise the download will fail;
3 Select the serial port number corresponding to U8W;
4 Open the target file (HEX format or BIN format);
5 Set the hardware options;
6 Click the "Download/Program" button to start burning;
7The step information of the programming process is displayed, and the message "Completed!" is displayed when the programming is completed.

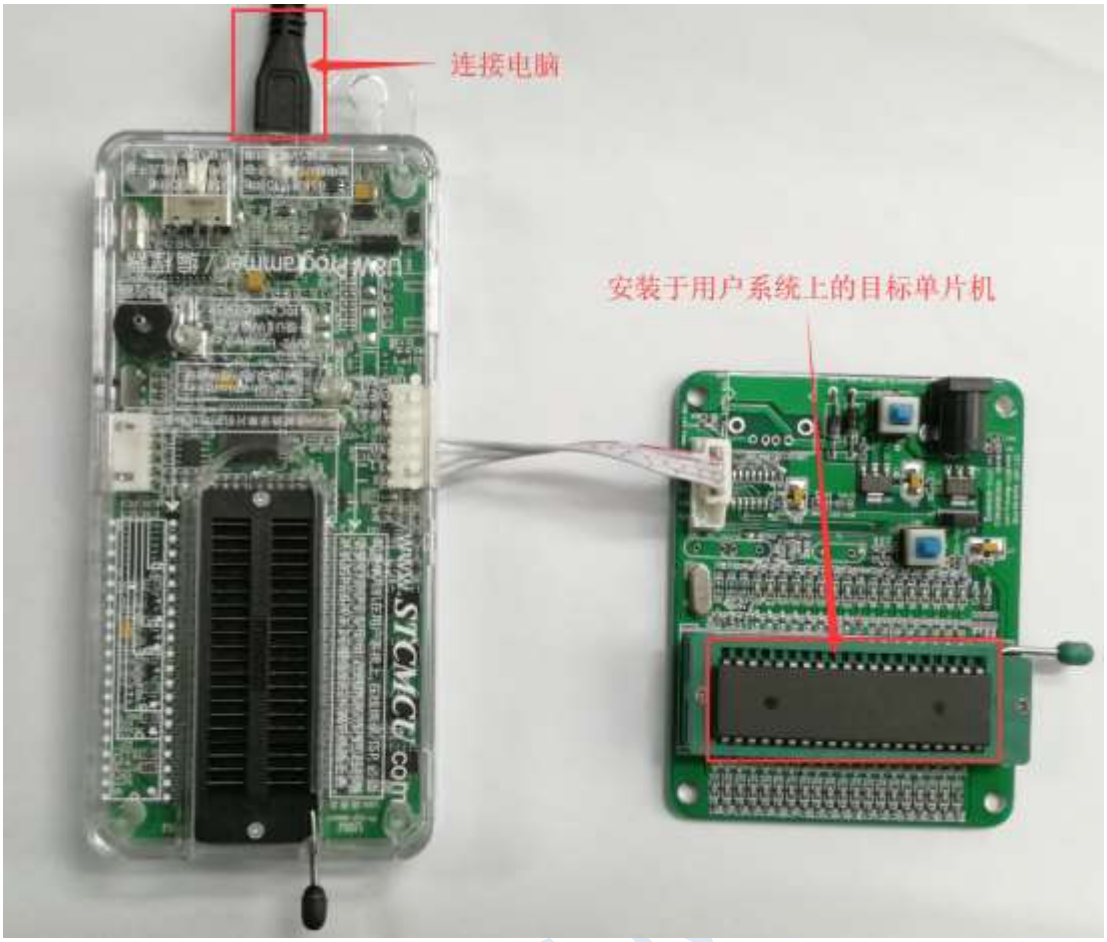
When there is the version number information of the output download board and the corresponding information of the external Flash in the information box, it means that the U8W download tool has been correctly detected.

During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, All 4 LEDs will be on and off at the same time; if the download fails, all 4 LEDs will be off.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

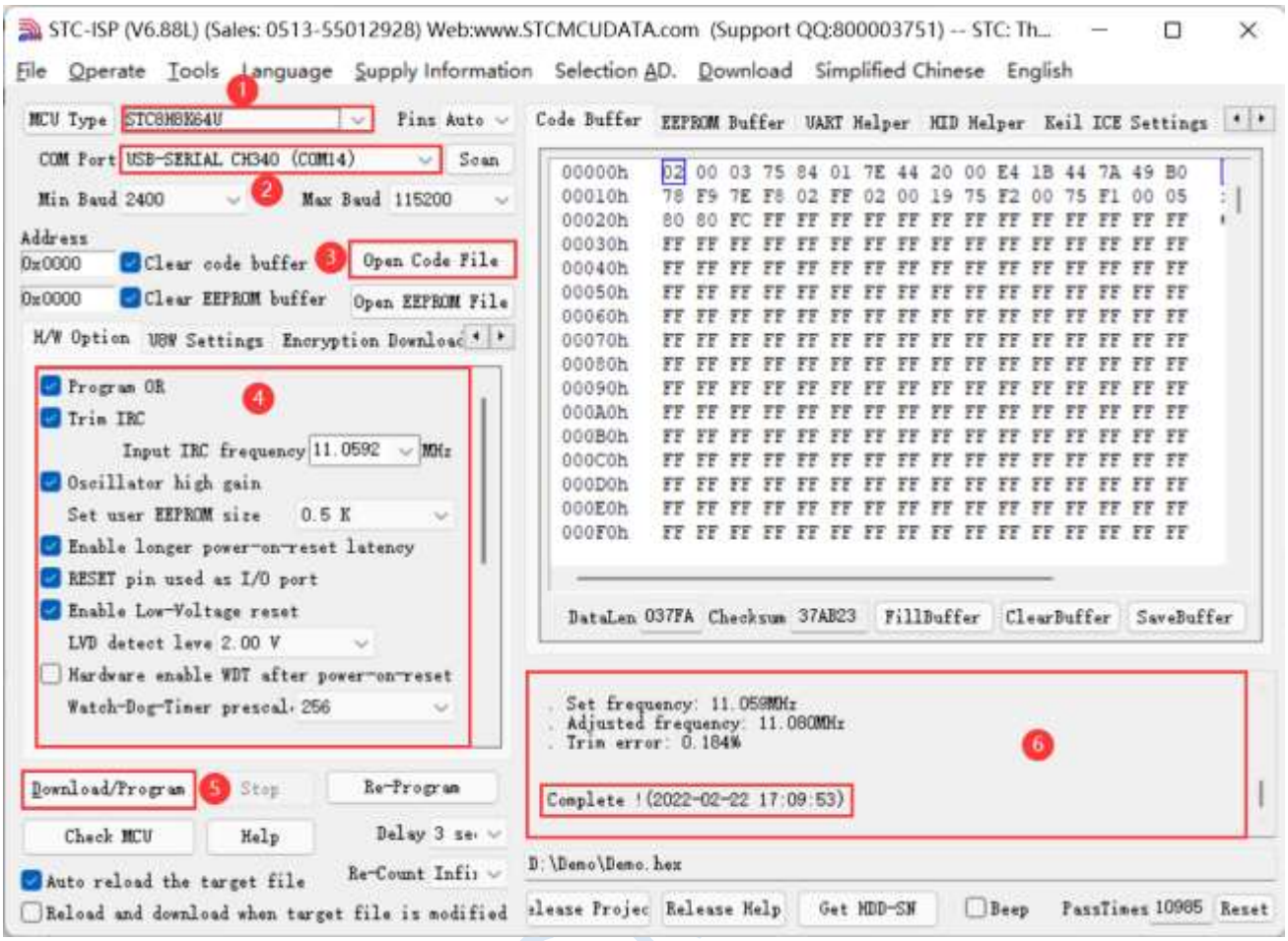
**The target chip is connected to U8W through the user system leads and U8W is connected to the computer for online download**

Firstly, use the USB cable provided by STC to connect U8W to the computer, and then connect U8W to the target MCU of the user system through the download line. The connection method is shown in the following figure:




Then use STC-ISP to download the software to download the program, the steps are as follows:

1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set hardware options;
5. Click the "Download/Program" button to start burning;
6. The step information of the programming process is displayed, and the message "Completed!" is displayed when the programming is completed.




When there is the version number information of the output download board and the corresponding information of the external Flash in the information box, it means that the U8W download tool has been correctly detected.

During the download process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

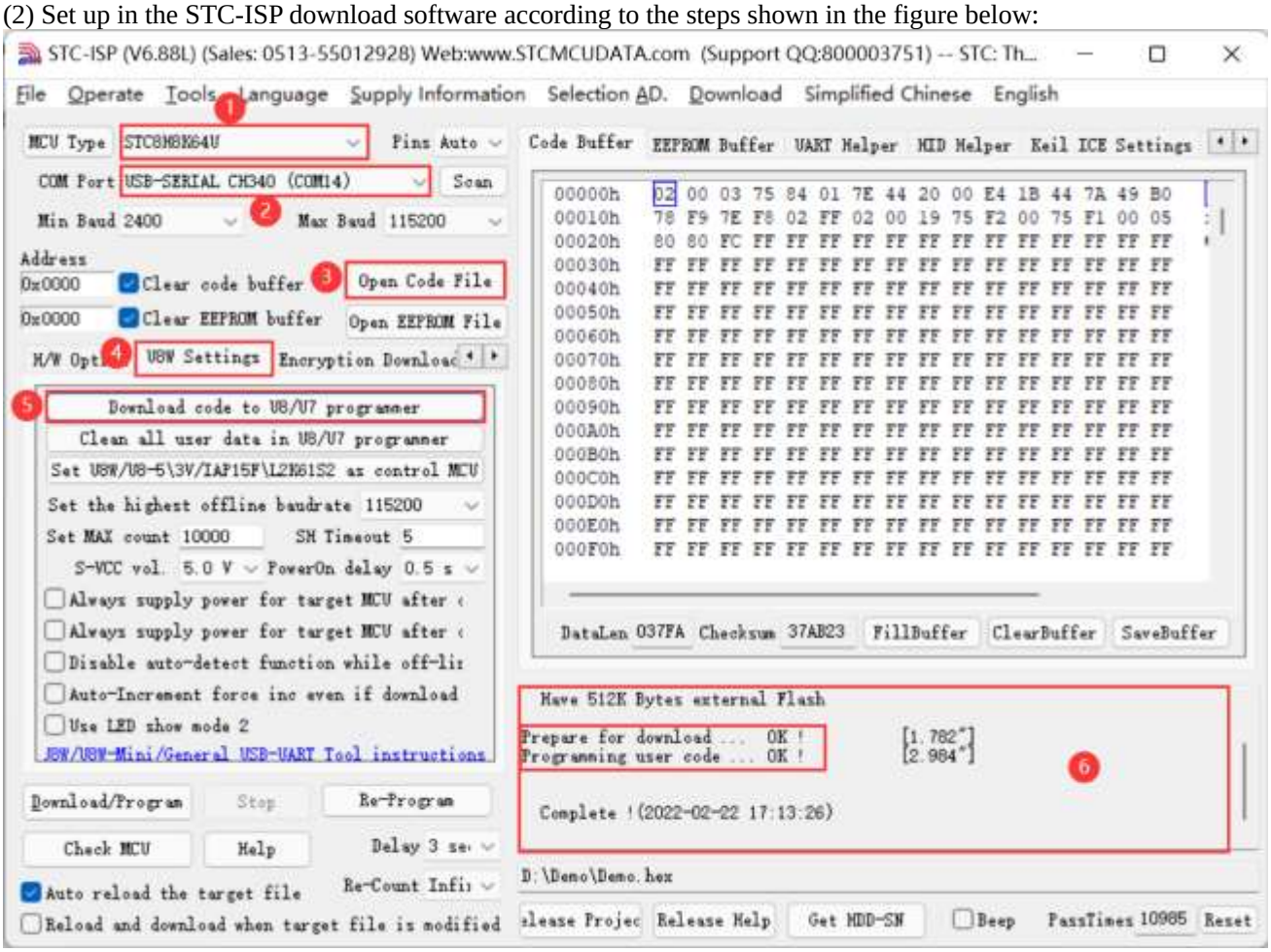
## F.3.4 U8W offline download instructions

The target chip is installed on the U8W socket and locked and connected to the computer via USB to power the U8W for offline download

The steps to use USB to power U8W for offline download are as follows:

(1) Use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:


(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

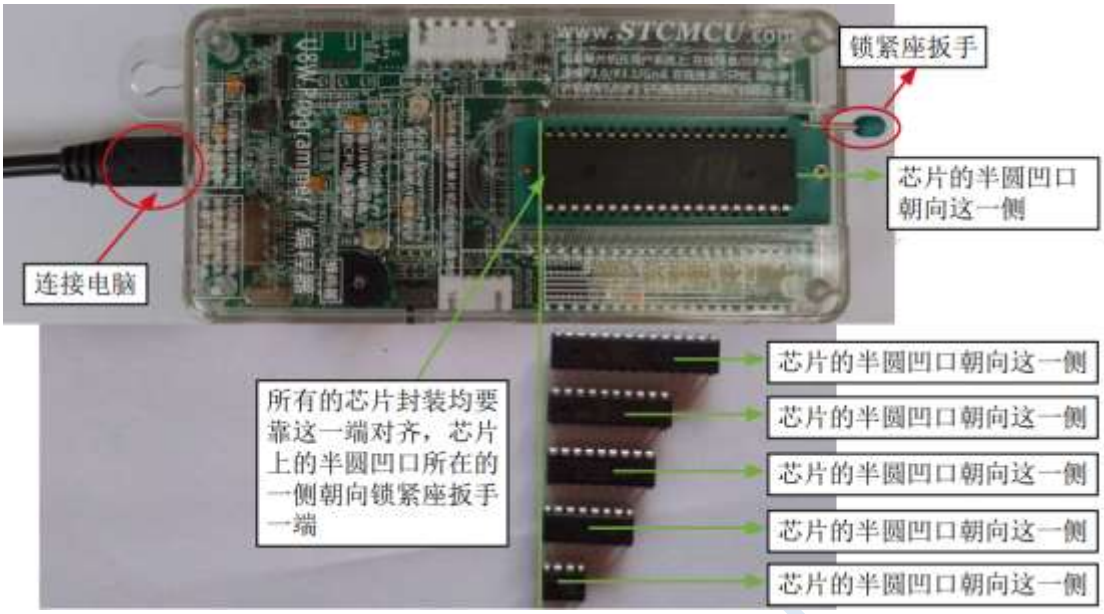



1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button ;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

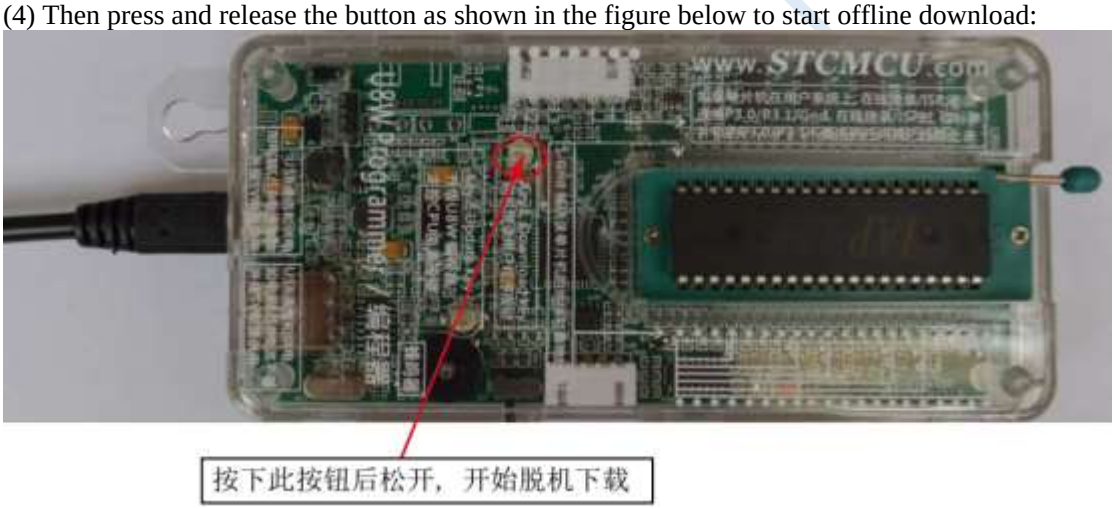
According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is

strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).
(3) Place the target MCU in the U8W download tool in the direction shown in the figure below, as shown in the figure below:




(4) Then press and release the button as shown in the figure below to start offline download:



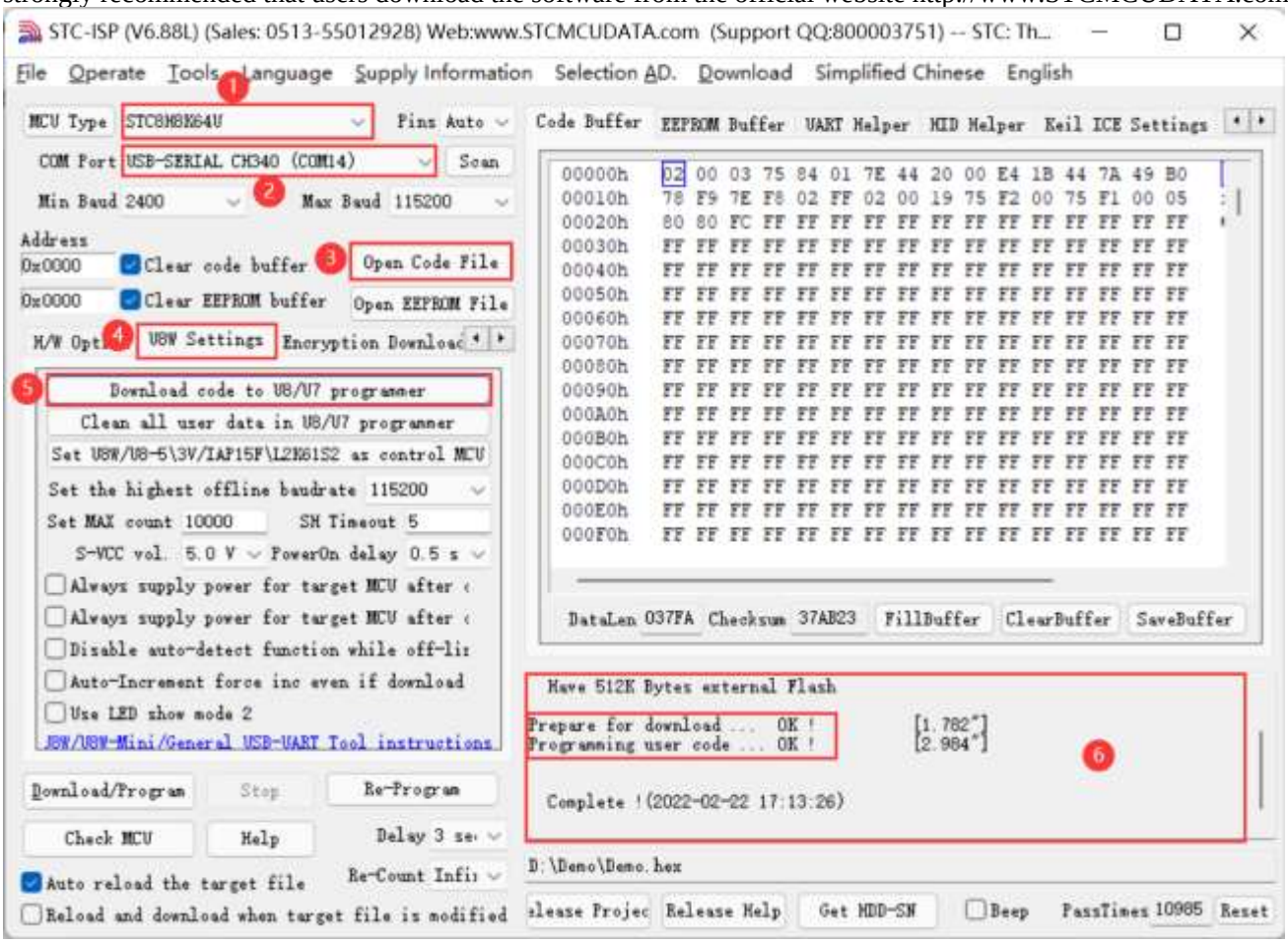

During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

Offline download plug and play burning function introduction:
1. After completing the above steps (1) and (2), U8W is in the plug-and-play programming state by default when it is connected to the computer and powered on;
2. Put the chip into the programming socket according to the instructions in step (3). While tightening the socket wrench, U8W will automatically start programming;
3. Display the burning process and burning result through the indicator light;
4. After programming is completed, loosen the wrench and take out the chip;
5. Repeat steps 2, 3 and 4 for continuous programming, eliminating the need to press the button to trigger the programming action.

**The target chip is connected to U8W by the user system lead and connected to the computer via USB to supply power to U8W for offline download.**

The steps for offline download using USB to supply power to U8W are as follows:
(1) Use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:

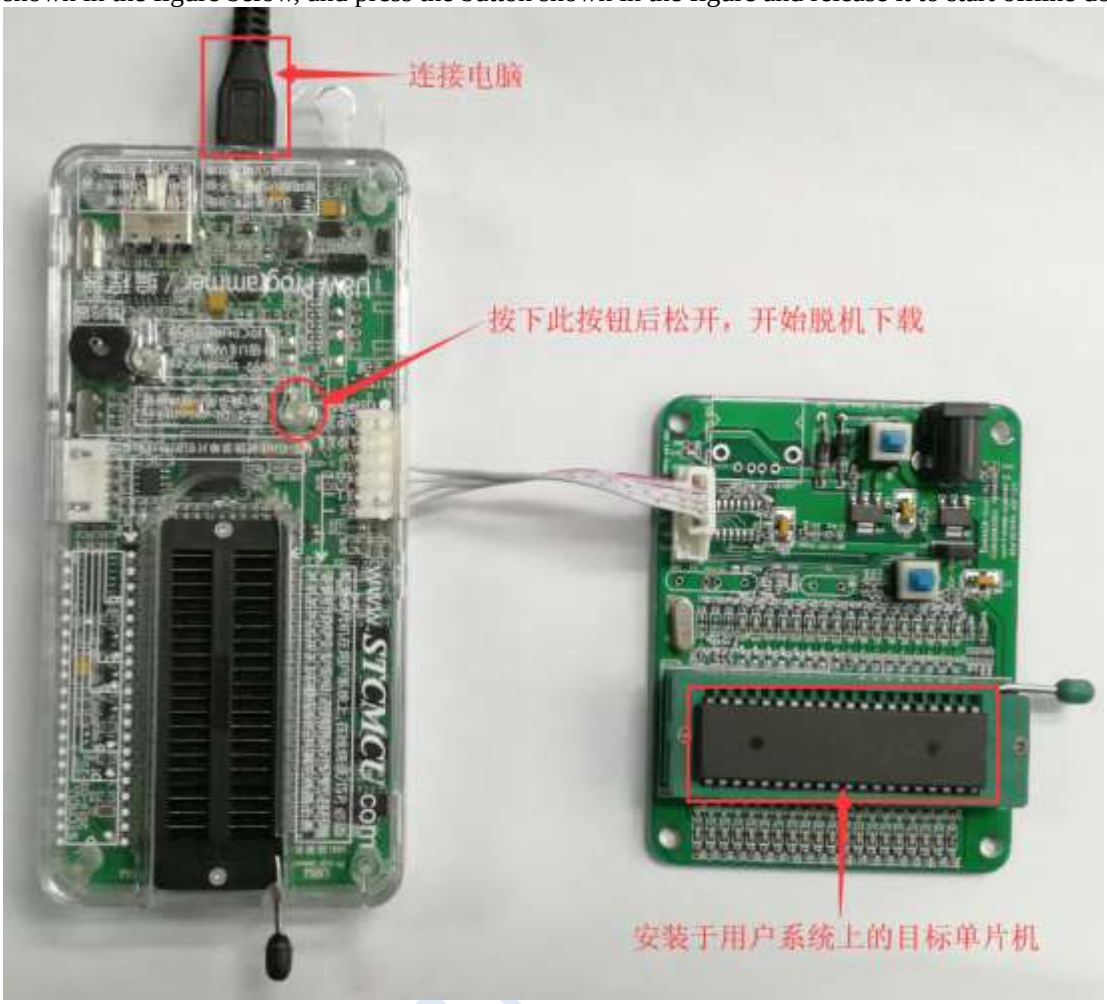

(2) Set up in the STC-ISP download software according to the steps shown in the figure below:
It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).




1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip operating voltage;
Click the "Download user program to U8/U7 programmer for offline download" button;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

(3) Then use the cable to connect the computer, connect the U8W download tool and the user system (target MCU) as shown in the figure below, and press the button shown in the figure and release it to start offline downloading:




During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

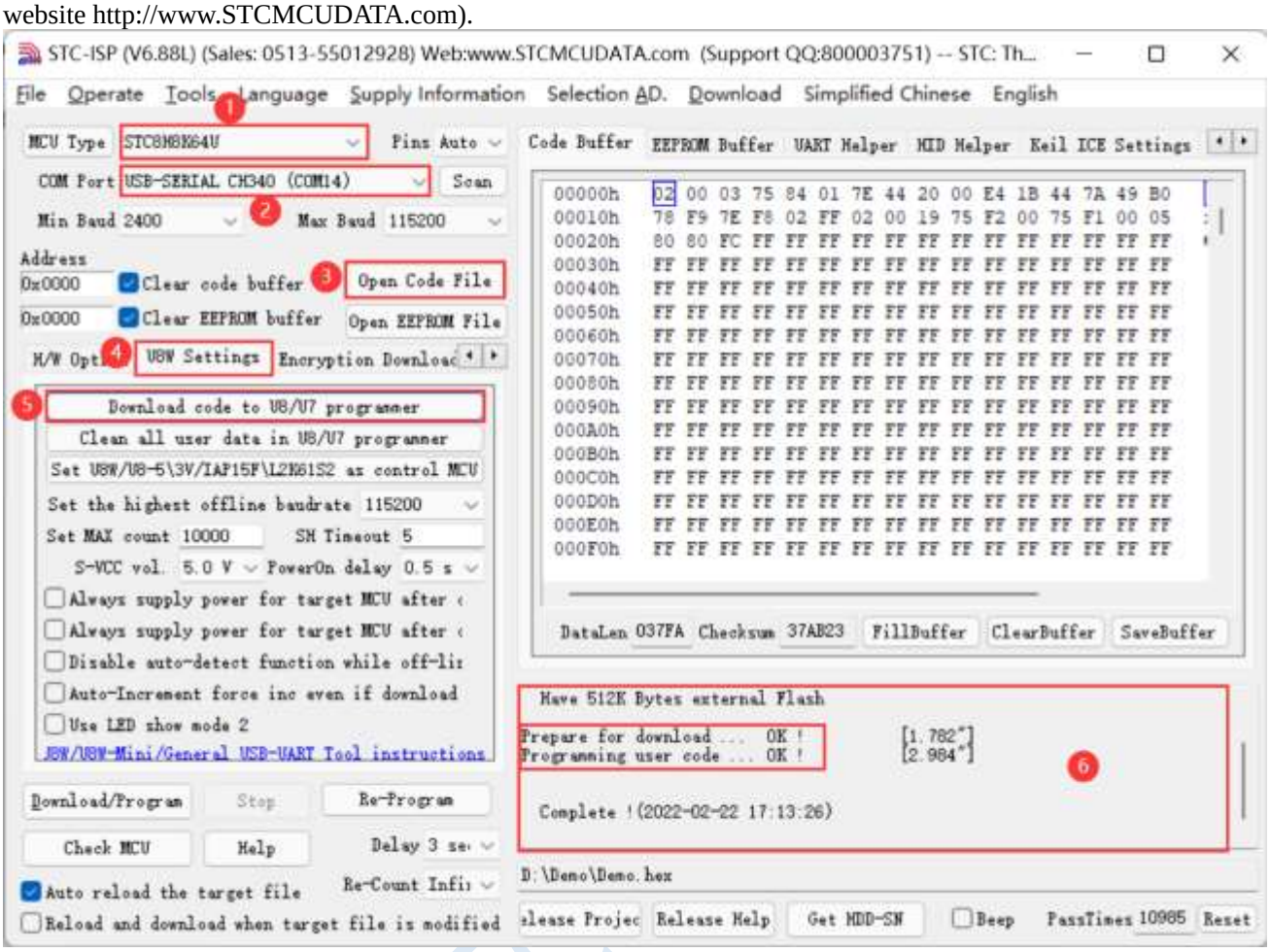
**The target chip is connected to U8W by the user system lead, and the user system supplies power to U8W for offline download**

(1) Firstly, use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:

(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website

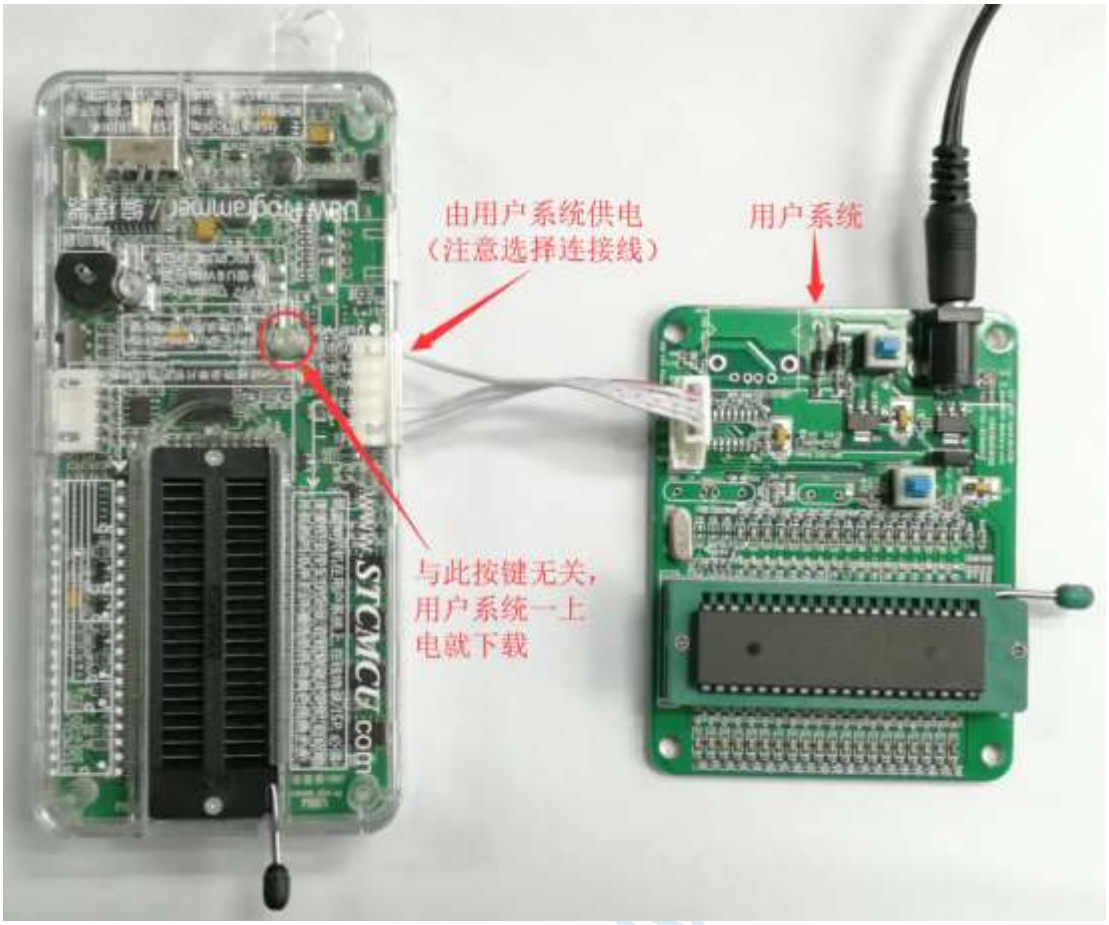
http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).




1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip operating voltage;
Click the "Download user program to U8/U7 programmer for offline download" button;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

(3) Then connect U8W to the user system as shown in the figure below, supply power to the user system, and then start offline downloading:



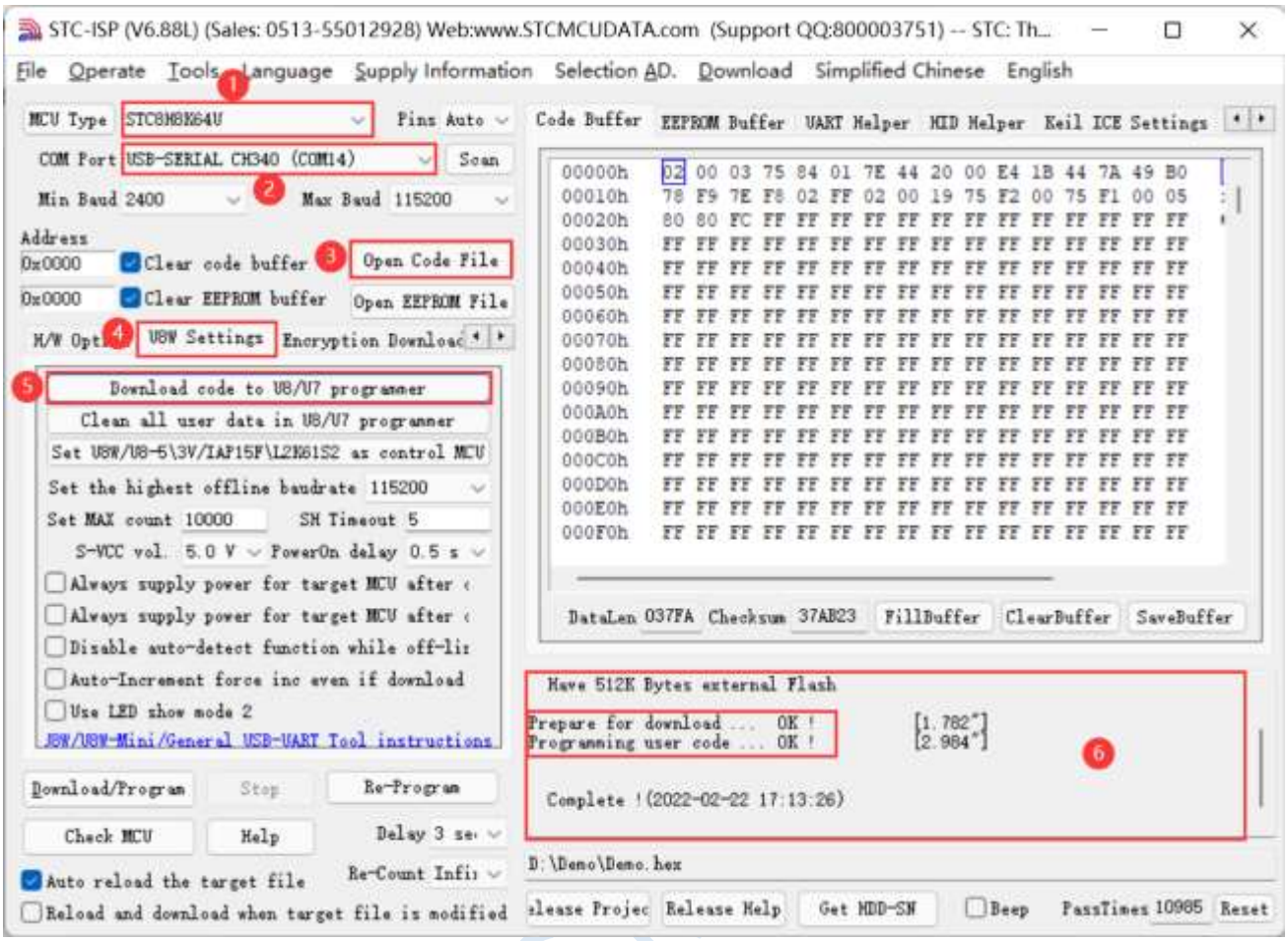

During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

**The target chip is connected to U8W by the user system lead, and U8W and the user system are independently powered for offline download**

(1) Firstly, use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:


(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

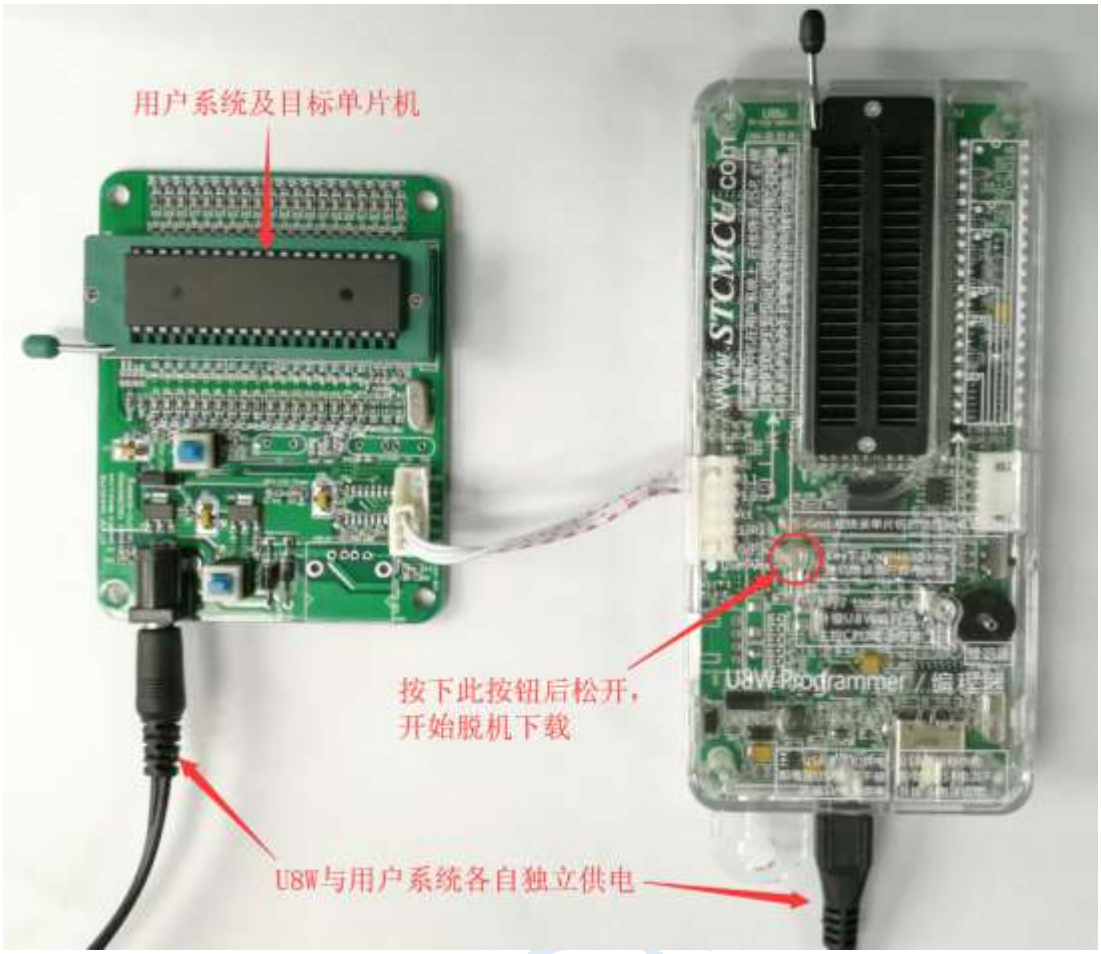
It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).




1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip operating voltage;
Click the "Download user program to U8/U7 programmer for offline download" button;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

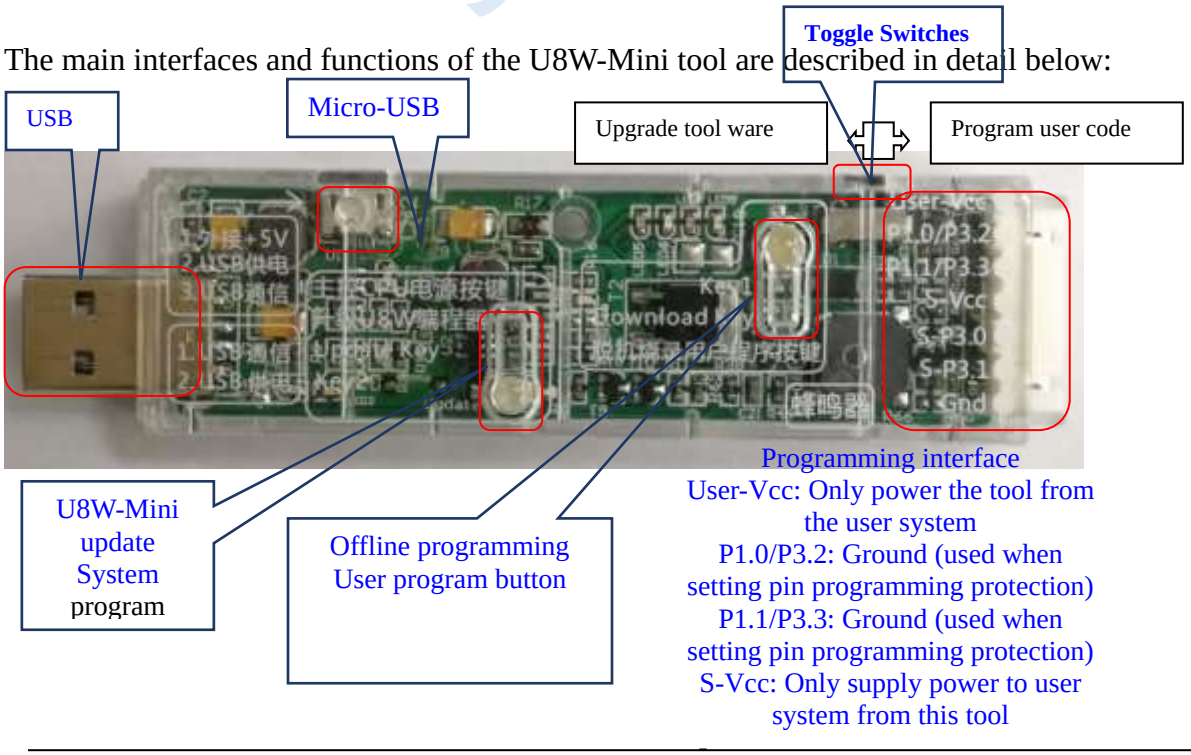
(3) Then connect U8W to the user system as shown in the figure below, and press the button shown in the figure first and then release it, ready to start offline download, and finally power on/on the user system to download the user program Officially begin:




During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

## H.3.5 U8W-Mini's function introduction

The main interfaces and functions of the U8W-Mini tool are described in detail below:

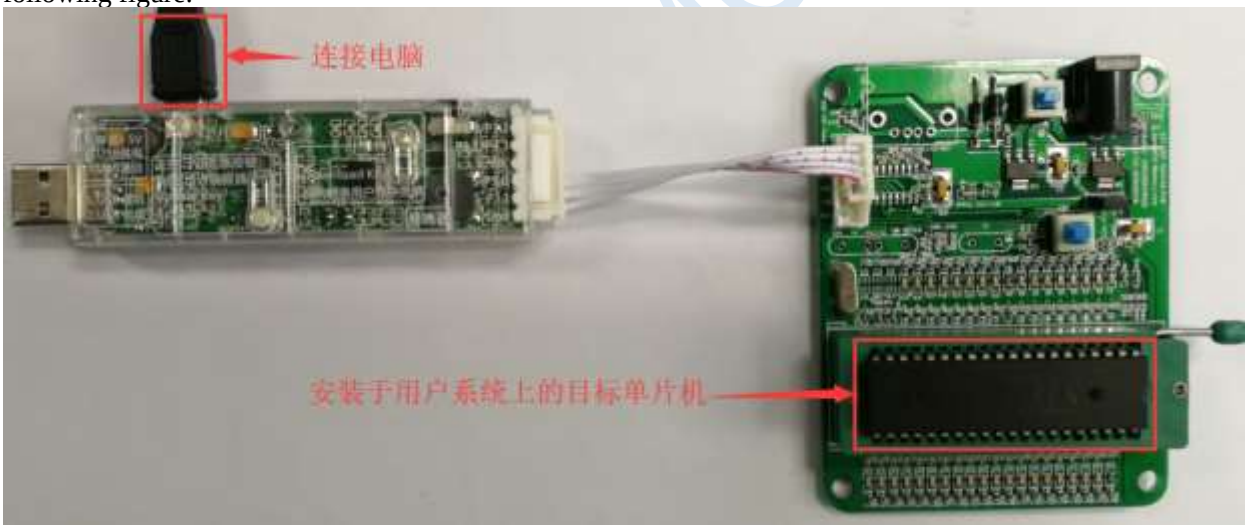


Programming interface: According to different power supply methods, use different download cables to connect the U8W-Mini download board and the user system.
U8W-Mini update system program button: used to update U8W-Mini tools. When there is a new version of U8W firmware, you need to press this button to update the U8W-Mini main control chip (note: you must first select update/download The toggle switch on the interface is moved to the upgrade tool firmware).
Offline download user program button: Start offline download button. First, the PC downloads the offline code to the U8W-Mini, and then uses the download cable to connect the user system to the U8W-Mini, and then press this button to start the offline download (the user will also start downloading immediately every time the power is turned on Code).
Update/download selection interface: When you need to upgrade the underlying firmware of U8W-Mini, you need to move this toggle switch to the firmware of the upgrade tool. When you need to program the target chip through U8W-Mini, you need to Toggle the switch to burn the user program. (Please refer to the figure above for the connection of the toggle switch)
USB interface: The USB interface has the same function as the Micro-USB interface. Users can connect one of them to the computer as needed.

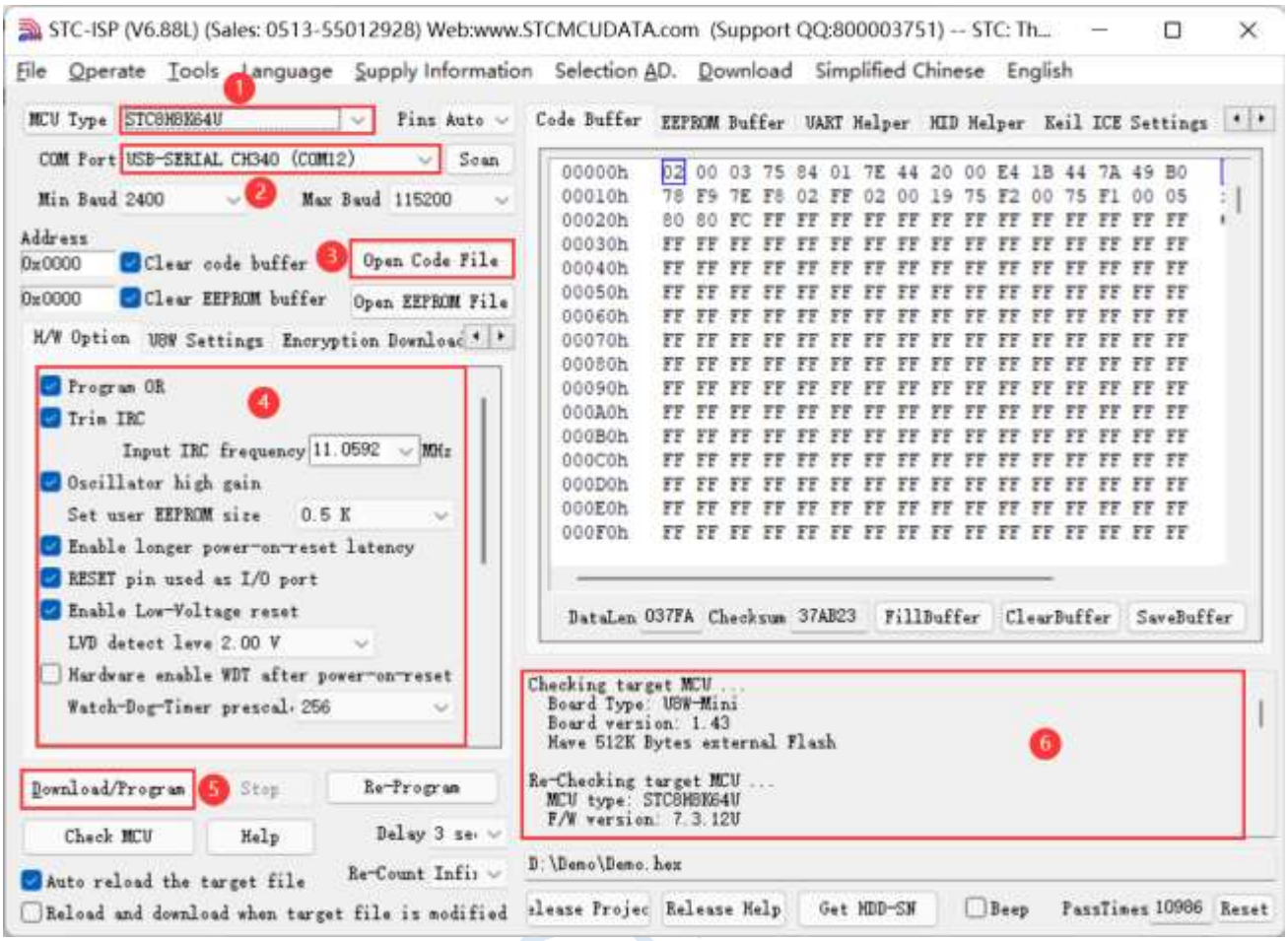
## H.3.6 U8W-Mini online download instructions

**The target chip is connected to the U8W-Mini through the user system lead, and the U8W-Mini is connected to the computer for online download**

Firstly, use the USB cable provided by STC to connect the U8W-Mini to the computer, and then connect the U8W-Mini to the target MCU of the user system through the download cable. The connection method is shown in the following figure:




Then use STC-ISP to download the software to download the program, the steps are as follows:



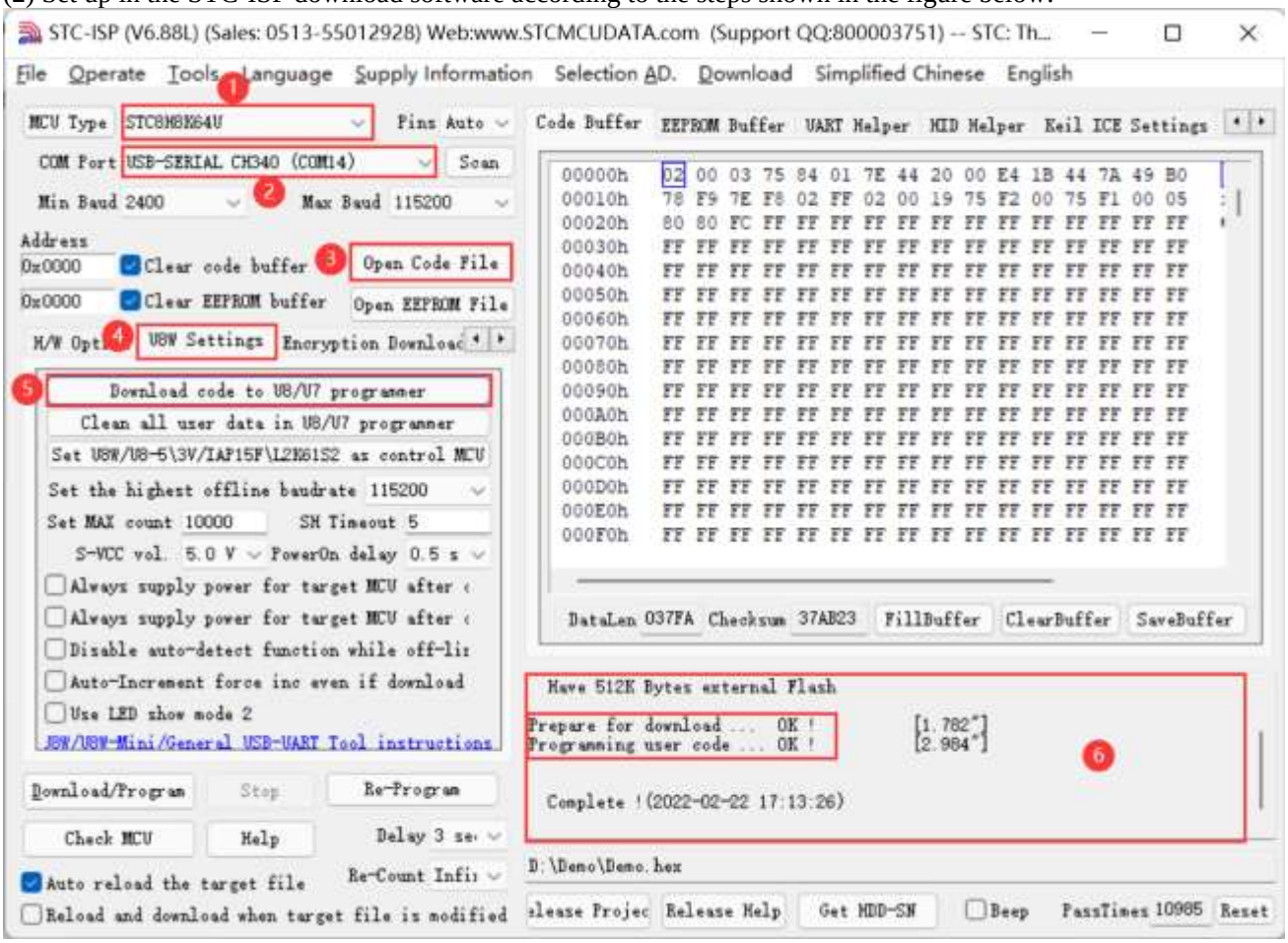

1. Select the MCU model;
2. Select the number of pins. When the chip is directly installed on the U8W-Mini to download, be sure to select the correct number of pins, otherwise the download will fail;
3. Select the serial port number corresponding to U8W-Mini;
4. Open the target file (HEX format or BIN format);
5. Set the hardware options;
6. Click the "Download/Program" button to start burning;
7. The step information of the burning process is displayed, and the message "Completed!" is displayed when the burning is completed.

When there is the version number information of the output download board and the corresponding information of the external Flash in the information box, it means that the U8W-Mini download tool has been correctly detected.
During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.
It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software on the official website http://www.STCMCUDATA.com).

## H.3.7 U8W-Mini offline download instructions

**The target chip is connected to the U8W-Mini by the user system lead and connected to the computer via USB to power the U8W-Mini for offline download.**
The steps for offline download using USB to power the U8W-Mini are as follows:
(1) Use the USB cable provided by STC to connect the U8W-Mini download board to the computer, as shown below:


(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

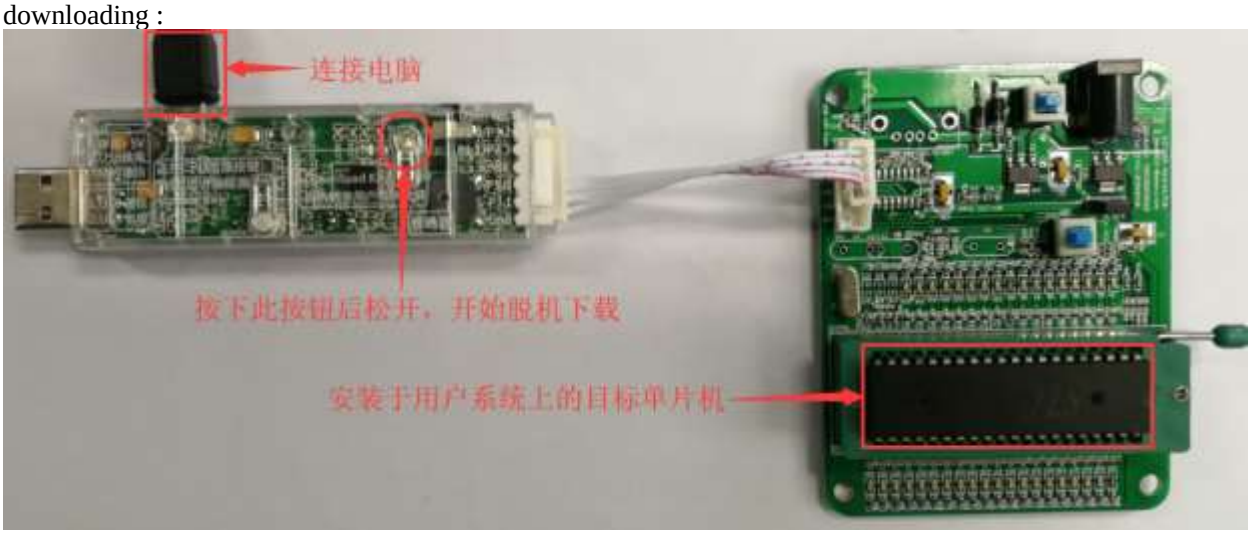



1. Select the MCU model;
2. Select the serial port number corresponding to U8W-Mini;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button ;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

Follow the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W-Mini download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is

strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).
(3) Then use the cable to connect the computer, connect the U8W-Mini download tool and the user system (target MCU) as shown in the figure below, and press the button shown in the figure and release it to start offline downloading :

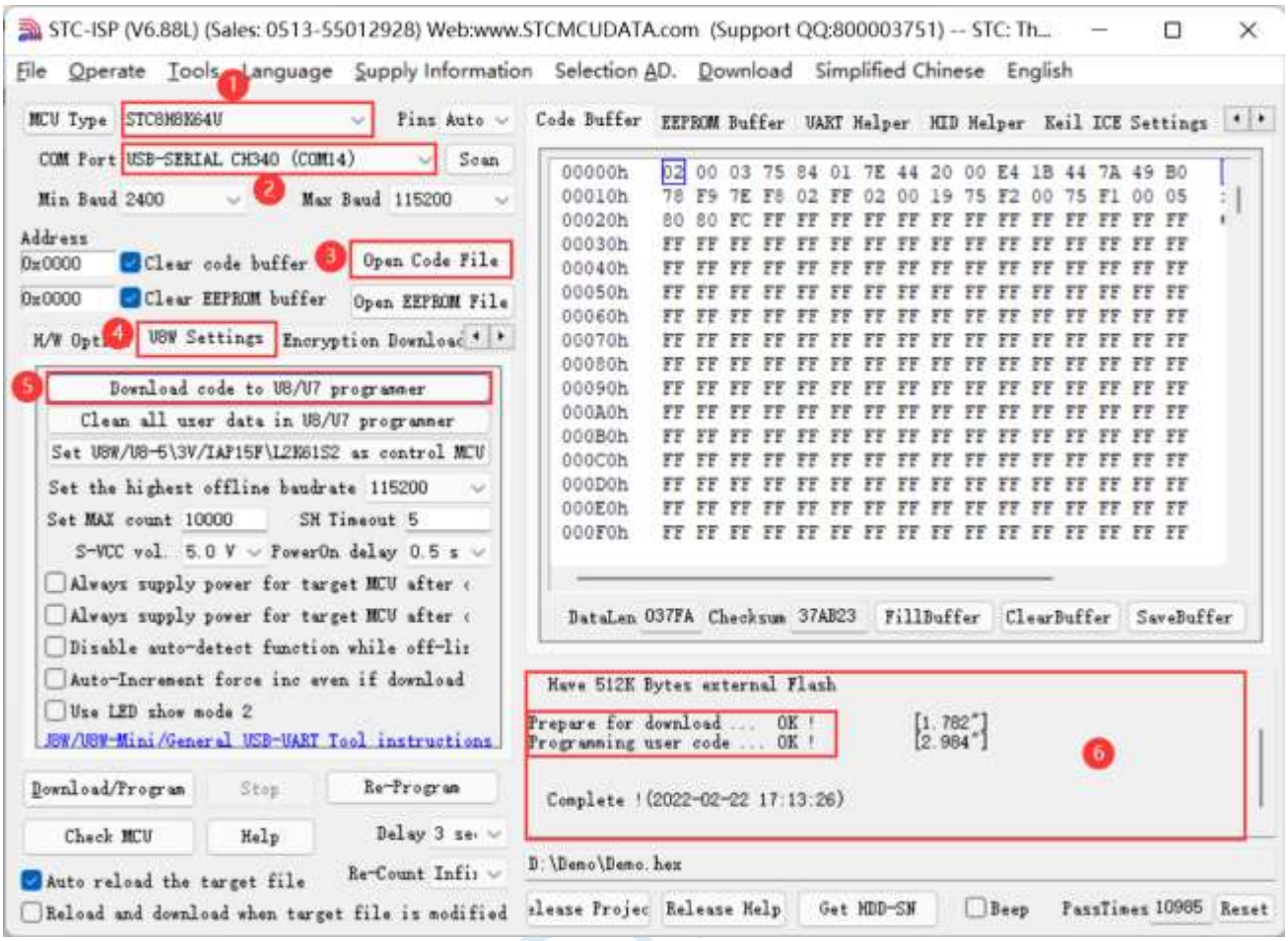



During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

**The target chip is connected to the U8W-Mini by the user system lead, and the U8W-Mini is powered by the user system for offline download.**

(1) Firstly, use the USB cable provided by STC to connect the U8W-Mini download board to the computer, as shown below:


(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

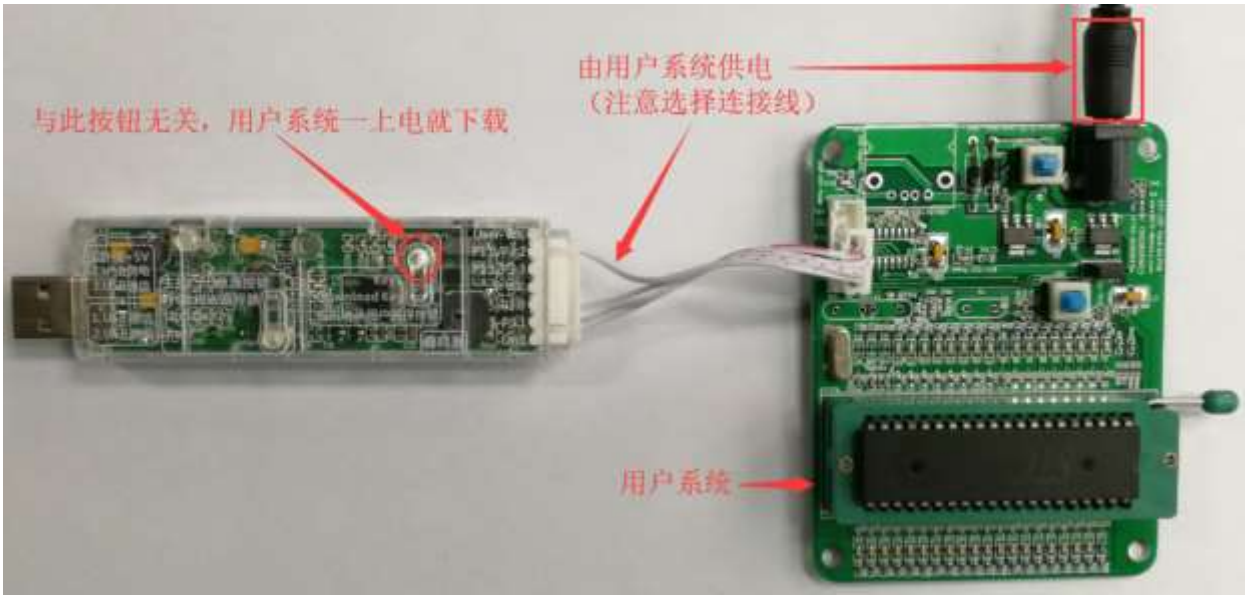



1. Select the MCU model;
2. Select the serial port number corresponding to U8W-Mini;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button ;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

Follow the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W-Mini download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com) Software used).

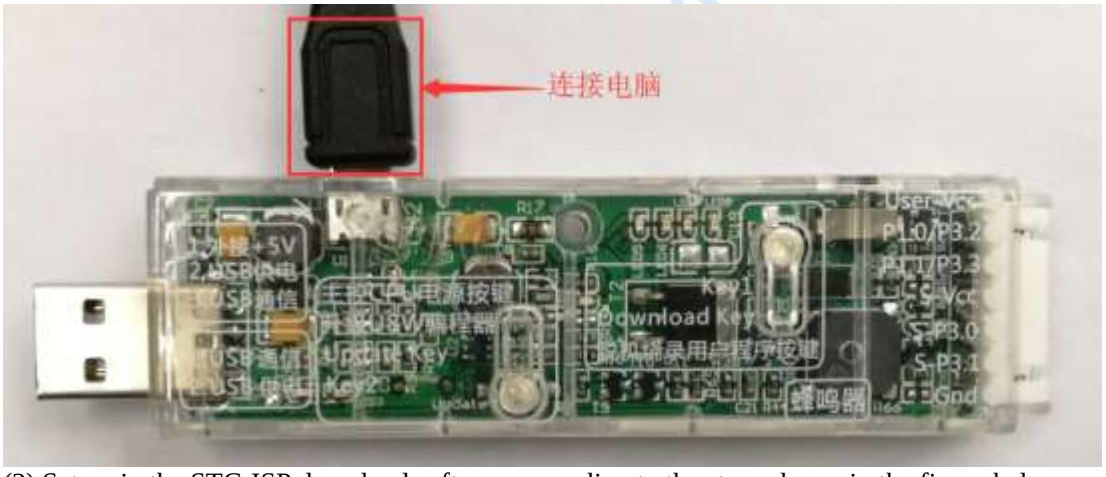
(3) Then connect U8W-Mini to the user system as shown in the figure below. Once the user system is powered on, it will start offline downloading:



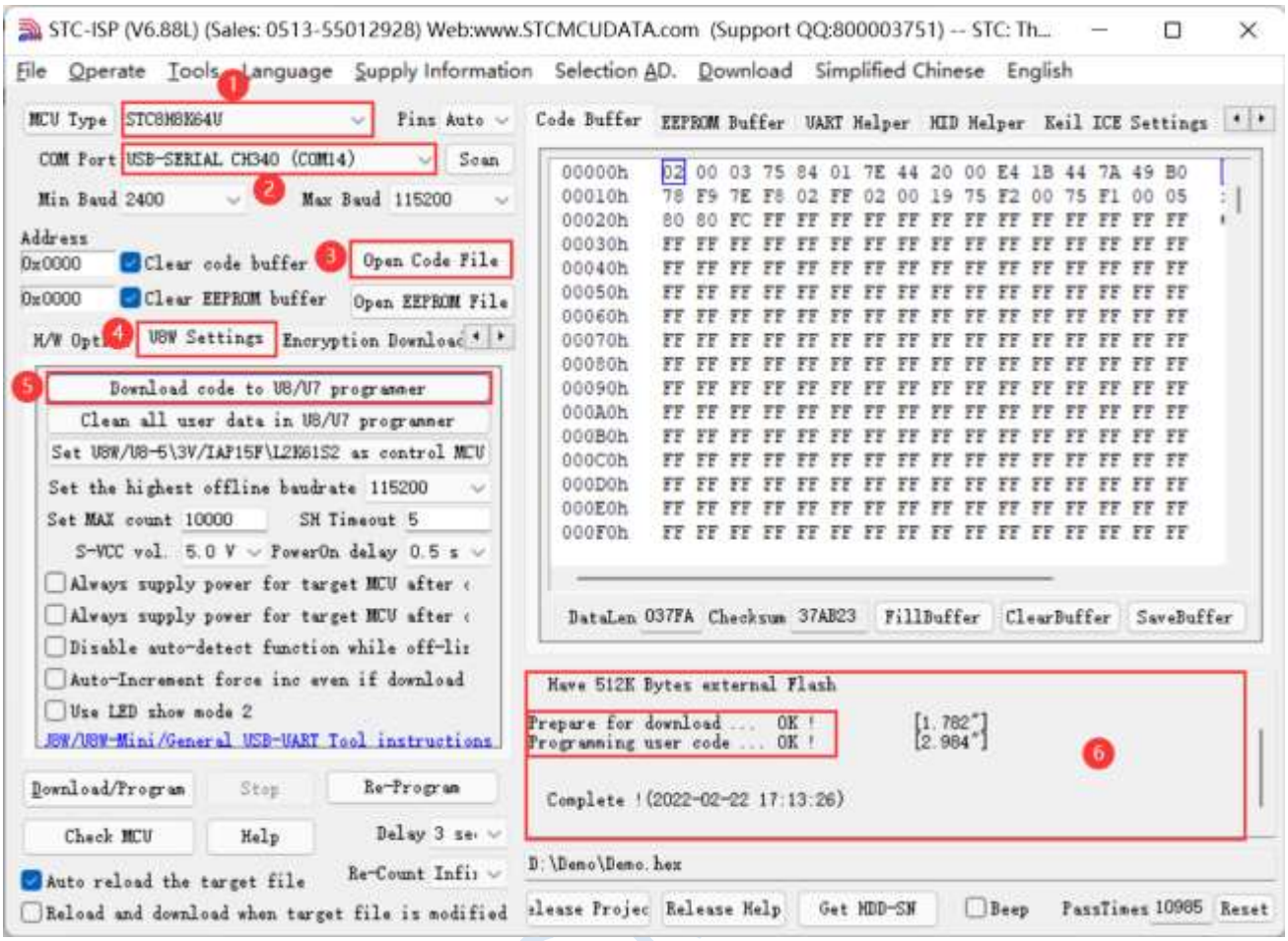

During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

**The target chip is connected to U8W-Mini by the user system lead, and U8W-Mini and the user system are independently powered for offline download.**

(1) Firstly, use the USB cable provided by STC to connect the U8W-Mini download board to the computer, as shown below:




(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

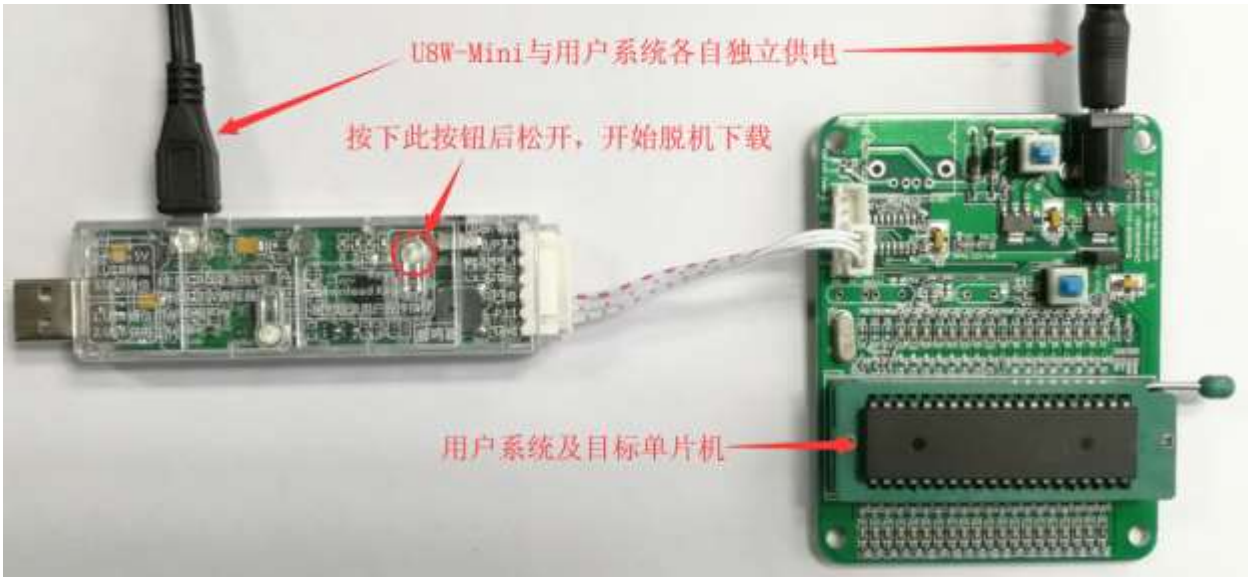



1. Select the MCU model;
2. Select the serial port number corresponding to U8W-Mini;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button ;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

Follow the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W-Mini download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

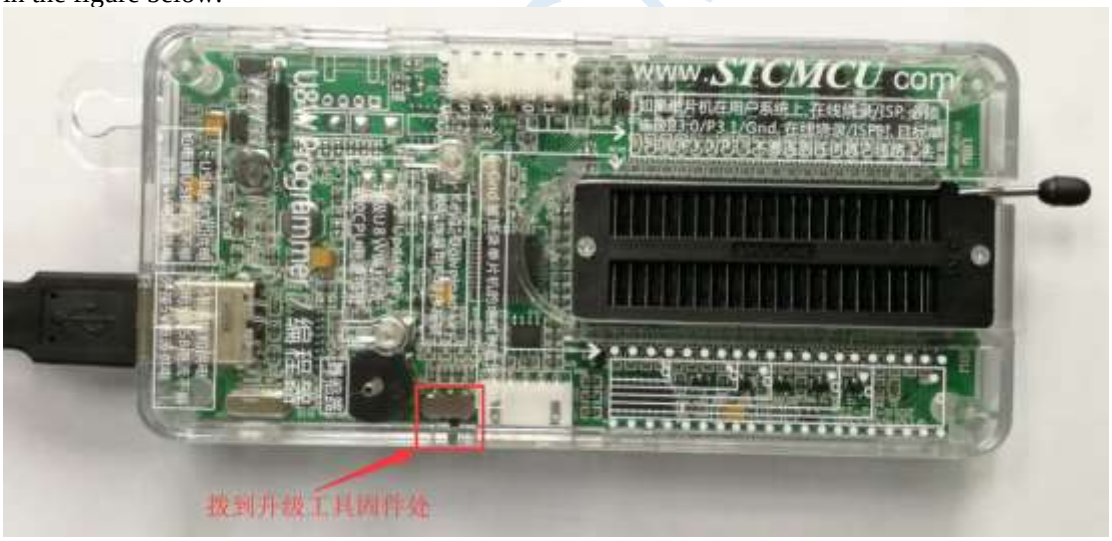
(3) Then connect U8W-Mini to the user system as shown in the figure below, and press the button shown in the figure first and then release it, ready to start offline download, and finally power on/on the user system to download The user program officially starts:



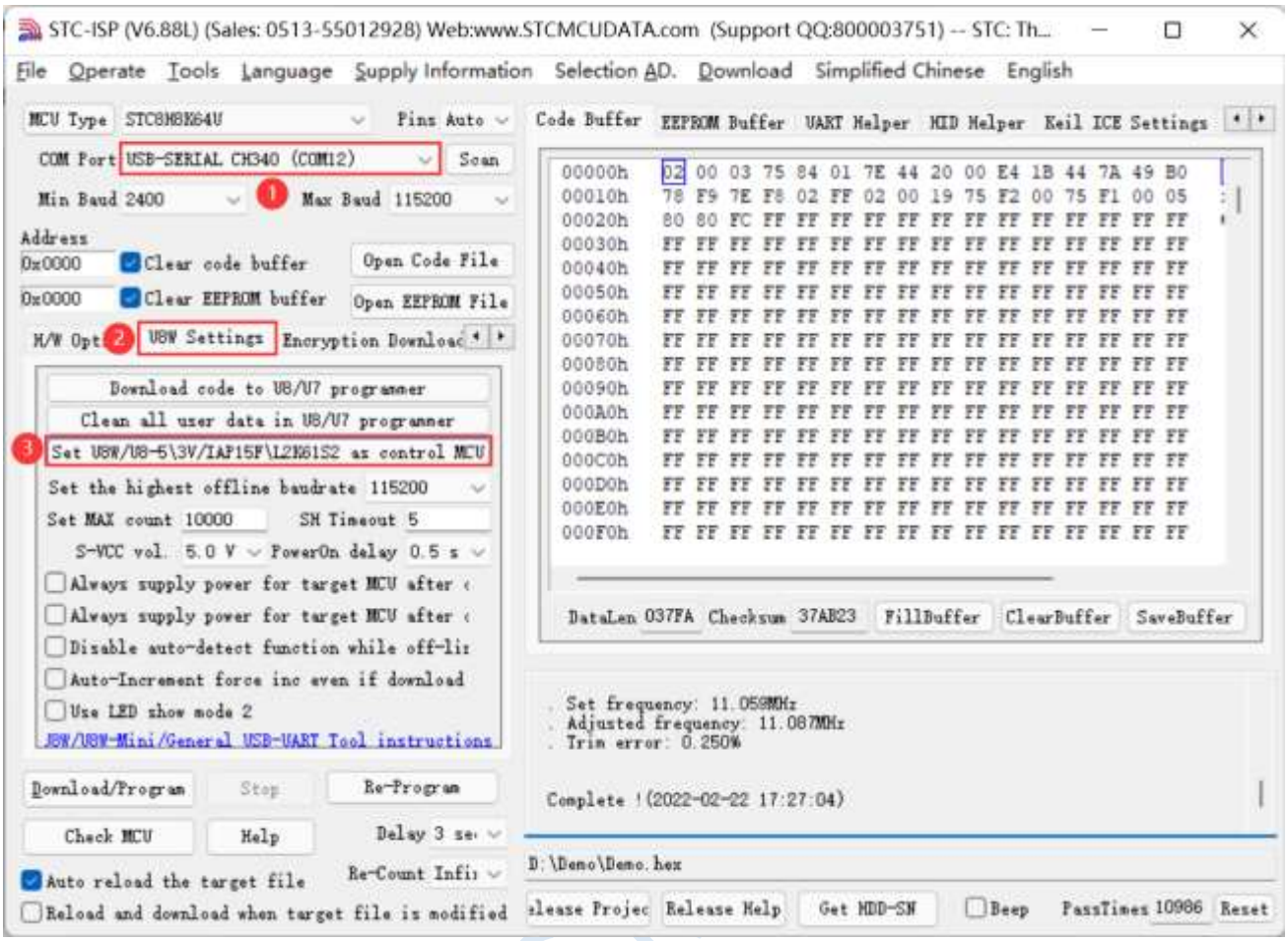

During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

## H.3.8 Make/Update U8W/U8W-Mini

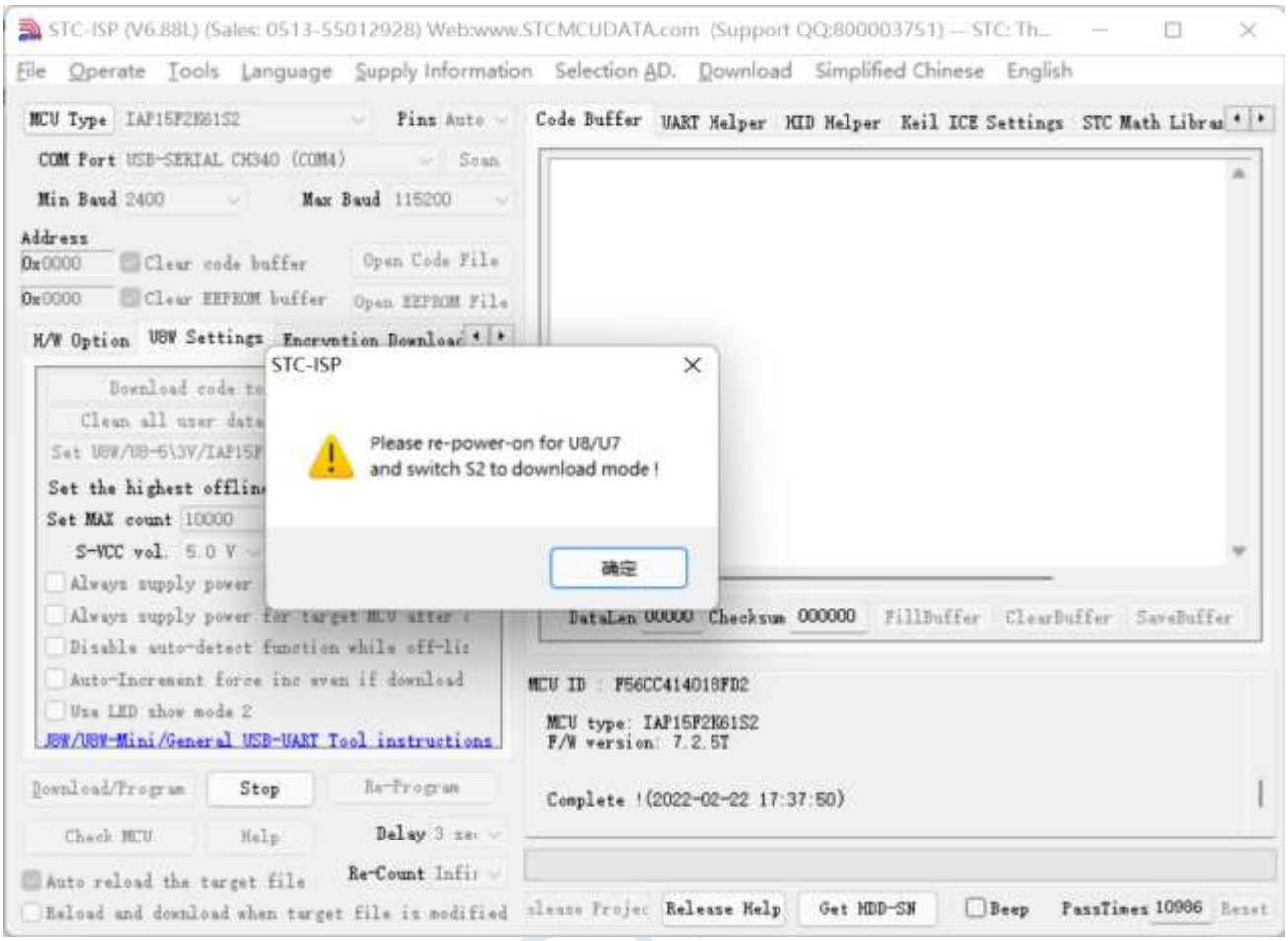
The process of making U8W/U8W-Mini download master is similar. To save space, the following uses U8W as an example to describe how to make U8W download master. Before making the U8W download master, you need to dial the "Update/Download Selection Interface" of the U8W download board to "Upgrade Tool Firmware", as shown in the figure below:



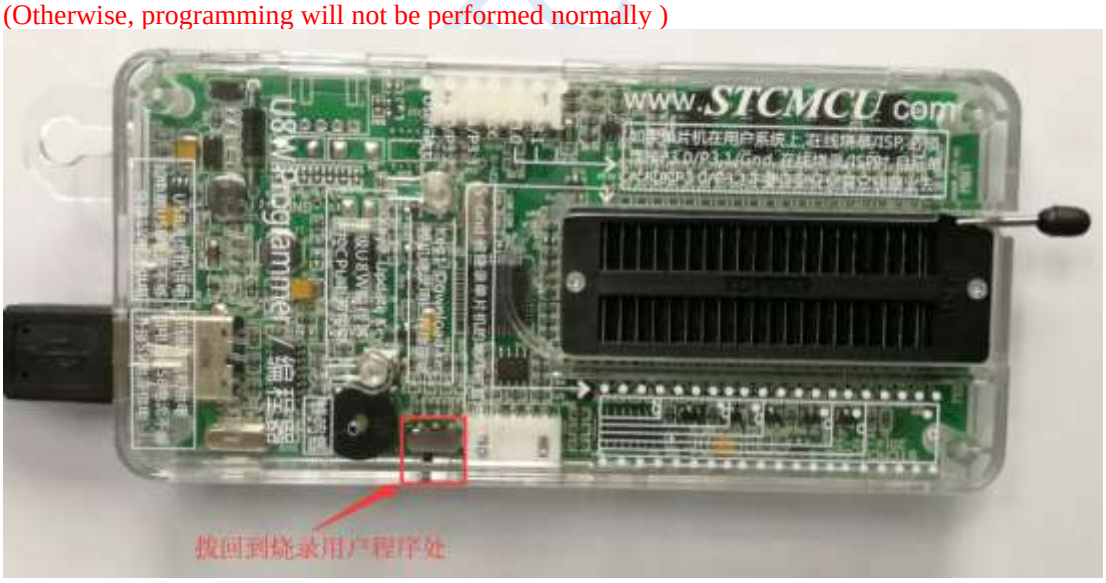

Then click the "Set U8W/U8-5V/U8-3V as the offline download master chip" button on the "U8W Offline/Online" page in the STC-ISP download program, as shown in the figure below: (Note: Be sure to select The serial port corresponding to U8W)




When the following screen appears, it indicates that the U8W control chip is made:




After the production is completed, do not forget to dial the "Update/Download Selection Interface" of U8W back to the "Burn User Program" mode, and power on the U8W download tool again, as shown in the figure below: (Otherwise, programming will not be performed normally )




# H.3.9 U8W/U8W-Mini set through mode (can be used for simulation)

To use U8W/U8-Mini for simulation, you must first set U8W/U8-Mini to pass-through mode. The method of U8W/U8W-Mini to realize USB to serial port pass-through mode is as follows:

1. First, the U8W/U8W-Mini firmware must be upgraded to v1.37 and above;

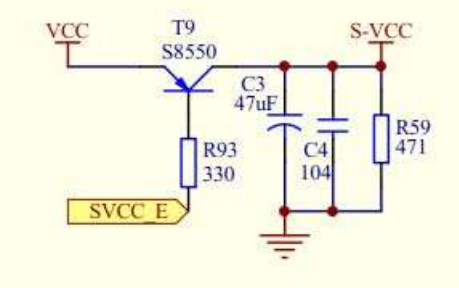
2. After U8W/U8W-Mini is powered on, it is in normal download mode. At this time, press and hold the Key1 (download) button on the tool and do not release it. Press the Key2 (power) button again, and then release the Key2 (power) button. Then release the Key1 (download) button, U8W/U8W-Mini will enter the USB to serial port pass-through mode. (Press Key1→ Press Key2 → Release Key2 → Release Key1);
3. The U8W/U8W-Mini tool that enters the pass-through mode is just a simple USB to serial port and does not have the offline download function. If you need to restore the original functions of U8W/U8W-Mini only need to press the Key2 (power) button separately again.

## H.3.10 Reference circuit of U8W/U8W-Mini

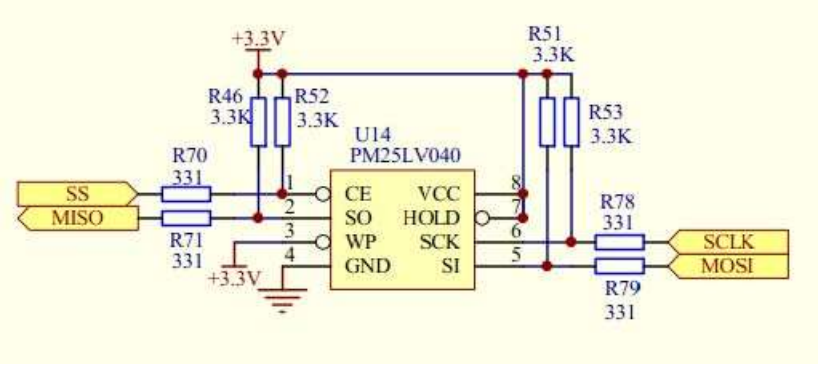
USB online/offline download board U8W/U8W-Mini provides users with the following common control interfaces:

| Pin function | Port | Function description |
|---|---|---|
| Power control pin | P2.6 | Low effective |
| Download communication pin | P1.0 | Serial port RXD, connect to the TXD of the target chip (P3.1) |
| | P1.1 | Serial port TXD, connect to RXD of target chip (P3.0) |
| Programming button | P3.6 | Low effective |
| display | P3.2 | LED1 |
| | P3.3 | LED2 |
| | P3.4 | LED3 |
| | P5.5 | LED4 |
| External serial Flash control pin | P2.4 | Flash CE Pin |
| | P2.2 | Flash SO Pin |
| | P2.3 | Flash SI Pin |
| | P2.1 | Flash SCLK Pin |
| Automatic burning tool Sorter signal | P3.6 | Start signal |
| | P1.5 | Completion signal |
| | P5.4 | OK signal (good signal) |
| | P3.7 | ERROR signal (defective product signal) |
| Buzzer (BEEP) control | P2.5 | High effective (sound at high level) |

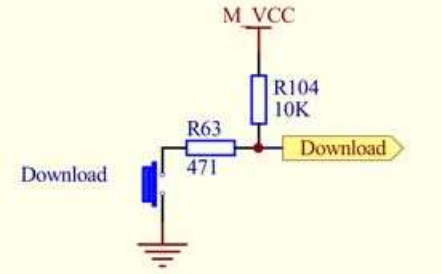
Reference circuit diagram for power control part:



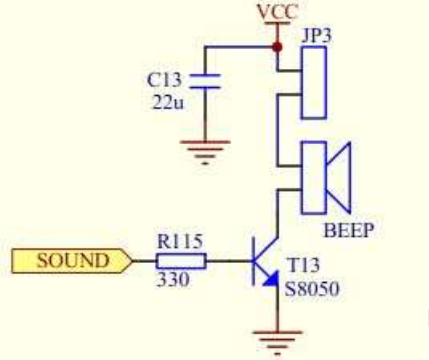

Reference circuit diagram of Flash control part:



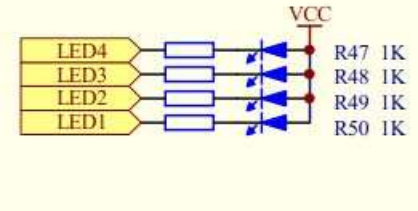

This Flash memory is required when the user program is larger than 41K
The reference circuit diagram of the button part:



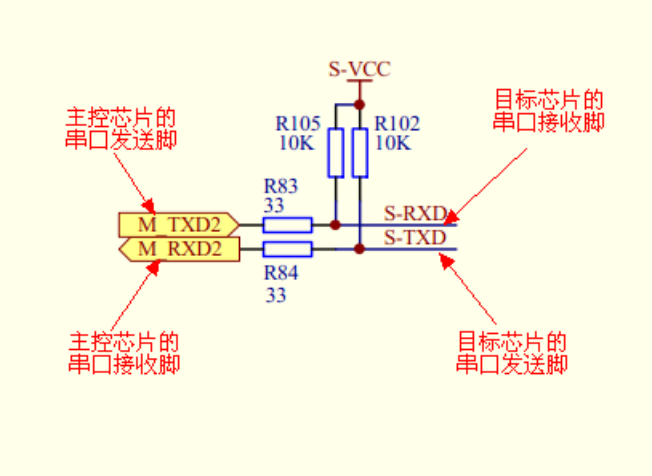

Reference circuit diagram of the buzzer part:




LED display part reference circuit diagram:



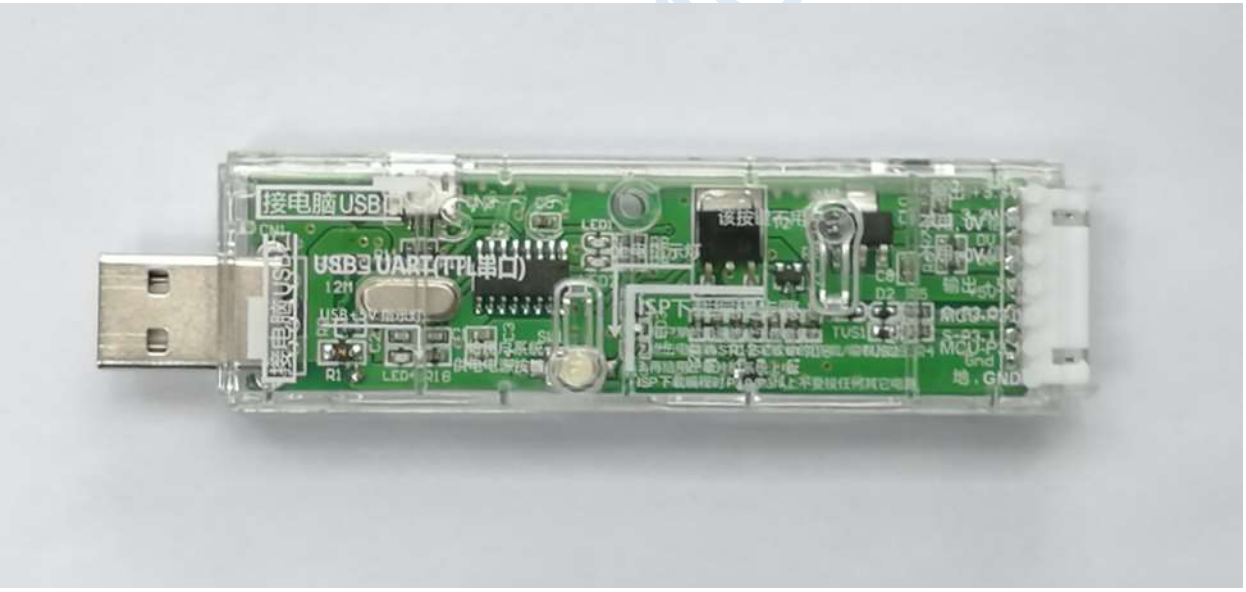

Reference circuit diagram of serial port communication pin connection part:




# H.4 STC Universal USB to Serial Tool

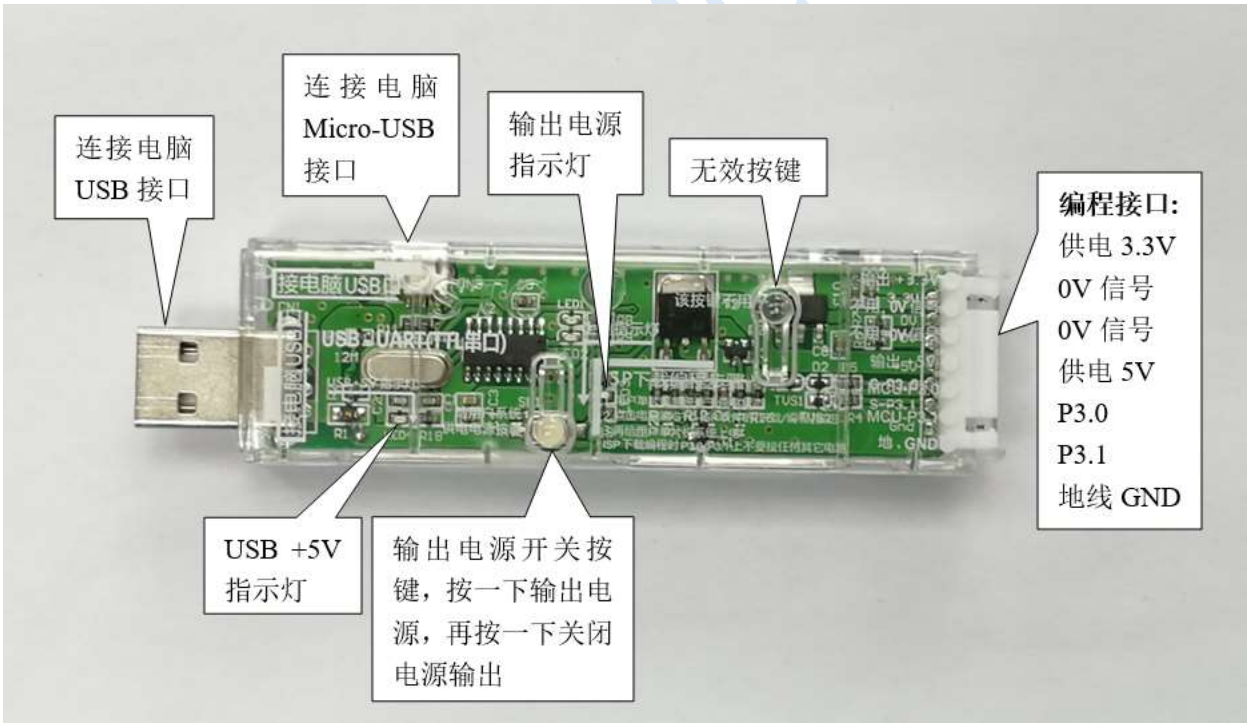
## H.4.1 Appearance of STC Universal USB to Serial Tool

Front:




Back:


## H.4.2 STC general USB to serial tool layout diagram




Here, the "power switch" needs to be explained:

The function of this button is the same as the self-locking switch. When the switch button is pressed for the first time, the switch is turned on and held, that is, self-locking. When the switch button is pressed for the second time, the switch turns off the power. In view of the characteristics of self-locking switches that are easily damaged during use, we have designed a set of circuits that use light touch switches to replace self-locking switches to increase the service life of the tools.

For STC microcontrollers, if you want to perform ISP download, you must receive the serial port command at power-on reset to start executing the ISP program, so the correct steps to download the program to the MCU using the STC universal USB to serial tool are:

1. Use STC universal USB to serial port tool to connect the MCU to be burned with the computer;
2. Open STC's ISP to download the software;
3. Select the MCU model;
4. Select the serial port corresponding to the STC Universal USB to Serial Tool;
5. Open the target file (HEX format or BIN format);
6. Click the "download/program" button in the ISP download software;
7. Press the "power switch" on the STC Universal USB to Serial Tool to power the MCU, and the download can start.

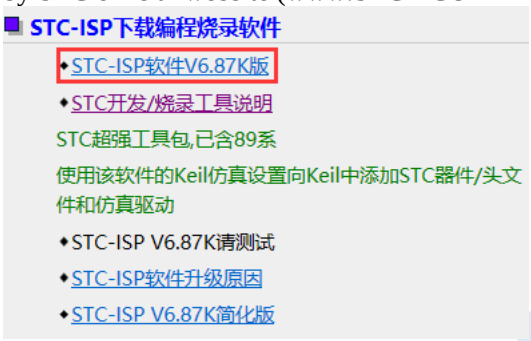
【Cold start burning】

In addition, the USB interface has the same function as the Micro-USB interface, and the user can connect one of the interfaces to the computer as needed.

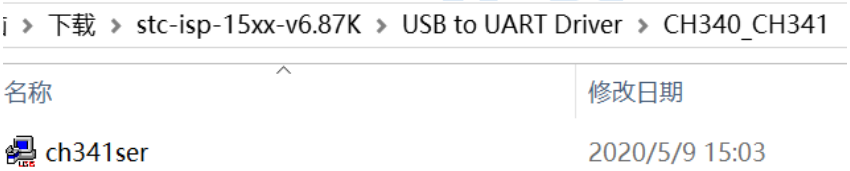
The 0V signal pin of the programming interface has a 470 ohm resistor grounded. It can be downloaded only when P1.0/P1.1=0/0 or P3.2/P3.3=0/0 is set. You can set P1.0, P1 .1 or P3.2, P3.3 are connected to the 0V signal pin.

## H.4.3 STC Universal USB to Serial Tool Driver Installation

STC general USB to serial port tool uses CH340 USB to serial chip (can plug in crystal oscillator, more accurate), just download the general CH340 serial driver and install it. The following is the CH341SER serial driver provided by STC official website (www.STCMCUDATA.com) Download location:

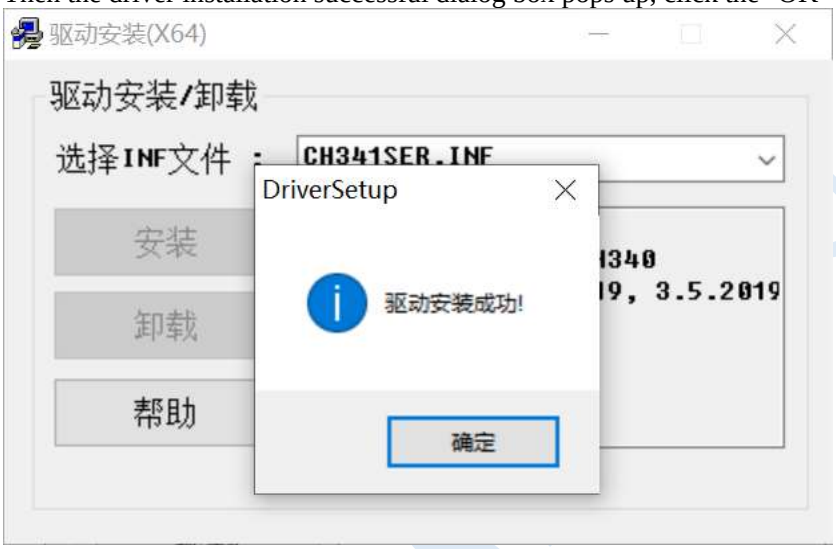



After downloading, decompress, the path of CH340 driver installation package is stc-isp-15xx-v6.87K\USB to UART Driver\CH340_CH341:



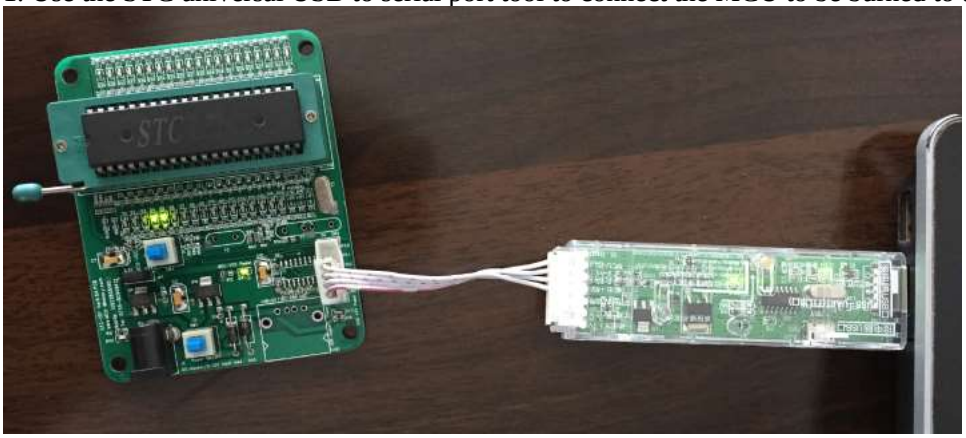

Take the CH341SER serial port driver provided by STC official website as an example, double-click the "CH341SER.exe" installation package, and click the "Install" button on the pop-up main interface to start installing the driver:


Then the driver installation successful dialog box pops up, click the "OK" button to complete the installation:



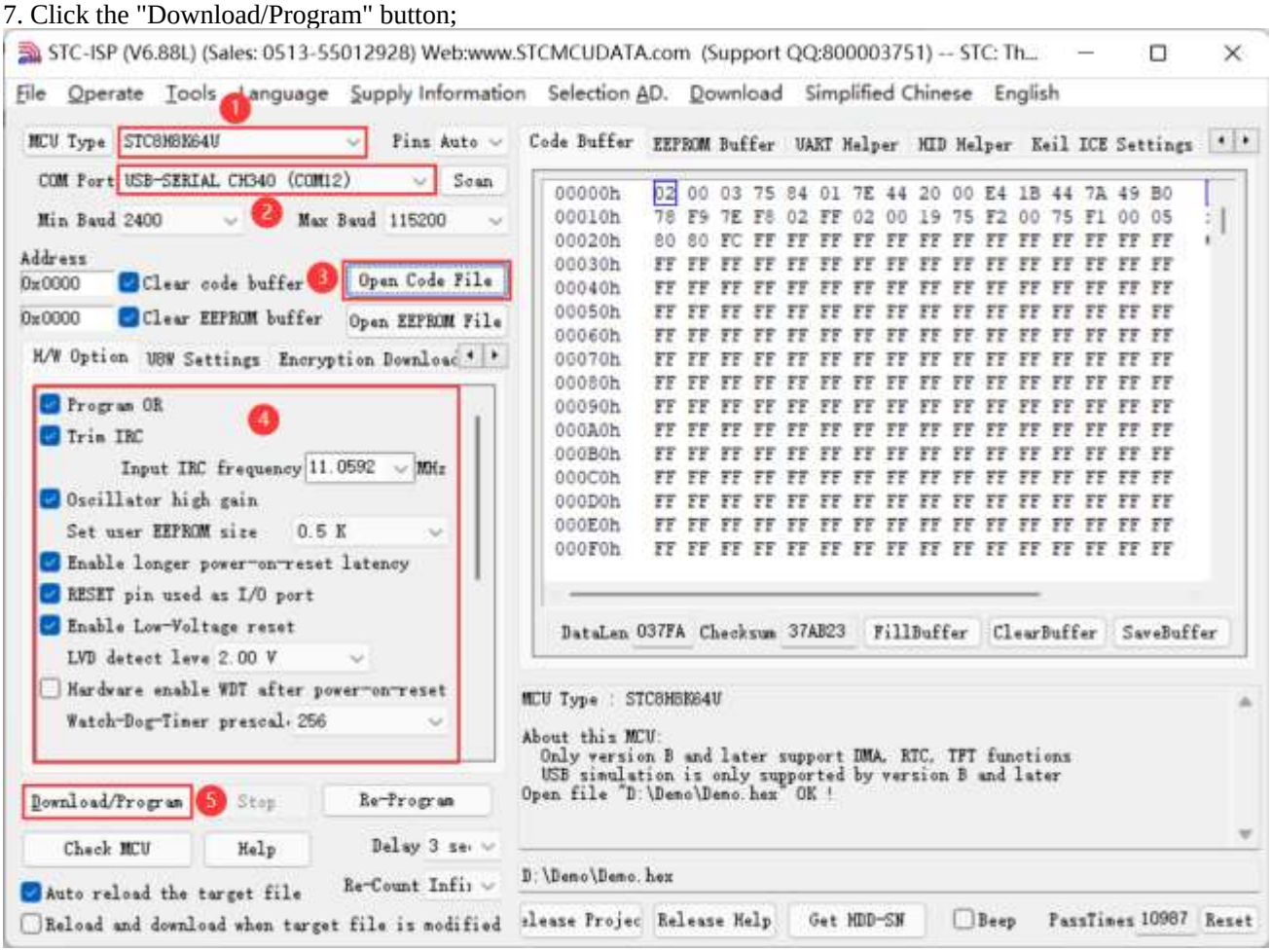

# H.4.4 Use STC universal USB to serial port tool to download program to MCU

1. Use the STC universal USB to serial port tool to connect the MCU to be burned to the computer:

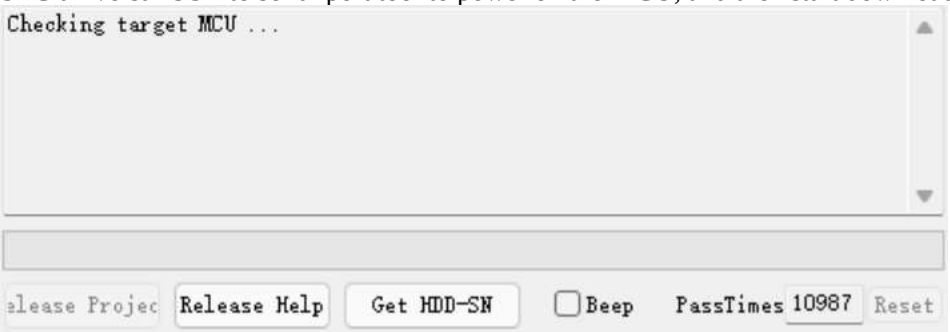


2. Open the STC-ISP software;

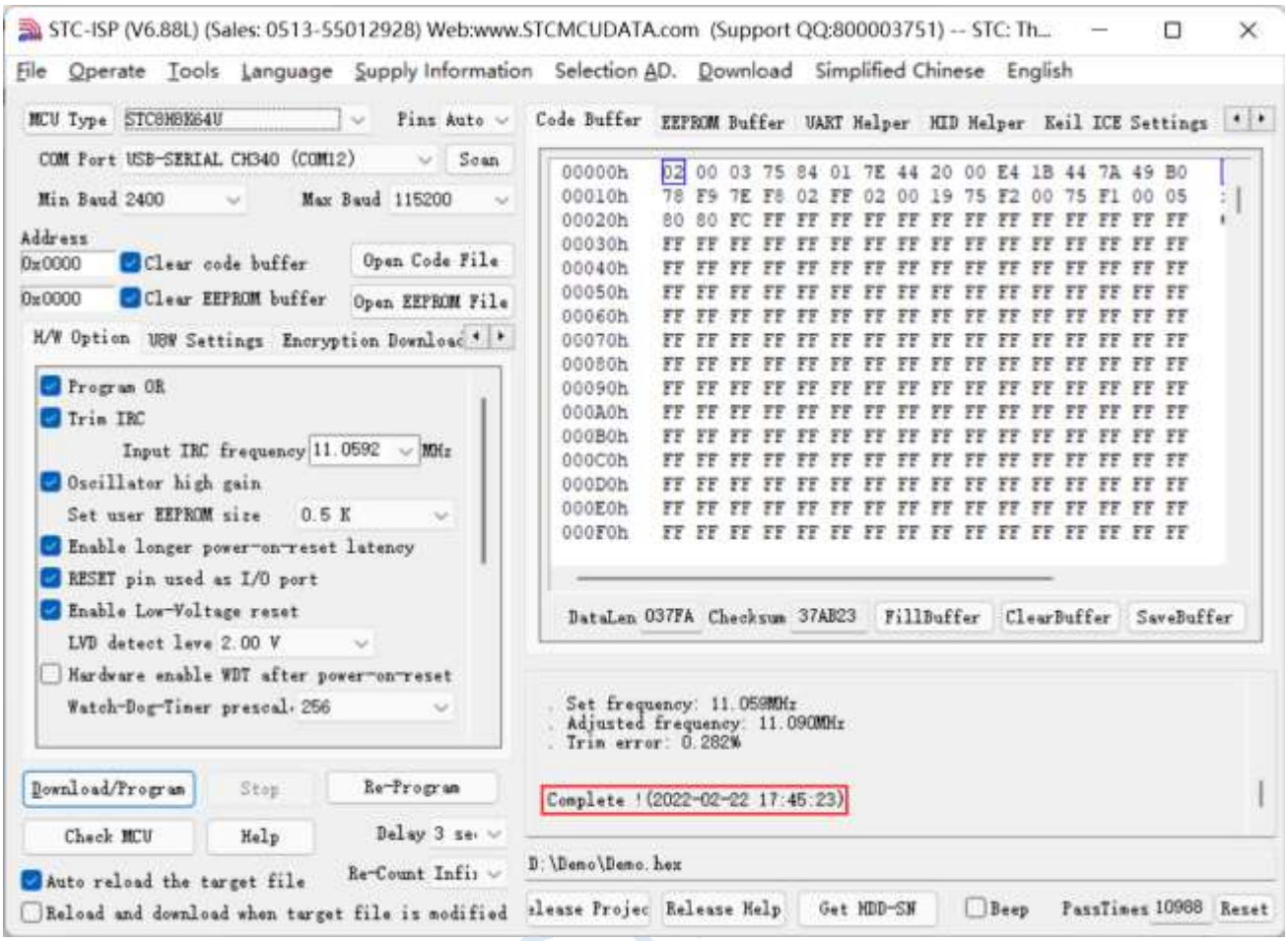
3. Select the model corresponding to the burning chip;
4. Select the serial port number recognized by the STC universal USB to serial tool (when the STC universal USB to serial tool is correctly connected to the computer, the software will automatically scan and identify the serial port named "USB-SERIAL CH340 (COMx)", the specific COM The number will vary from computer to computer). When multiple USB-to-serial cables are connected to the computer, they must be selected manually;
5. Load the burning program;
6. Set burning options;
7. Click the "Download/Program" button;




8. When the prompt box in the lower right corner displays "Checking target MCU...", press the "power switch" on the STC universal USB to serial port tool to power on the MCU, and then start downloading [Cold Start Programming];



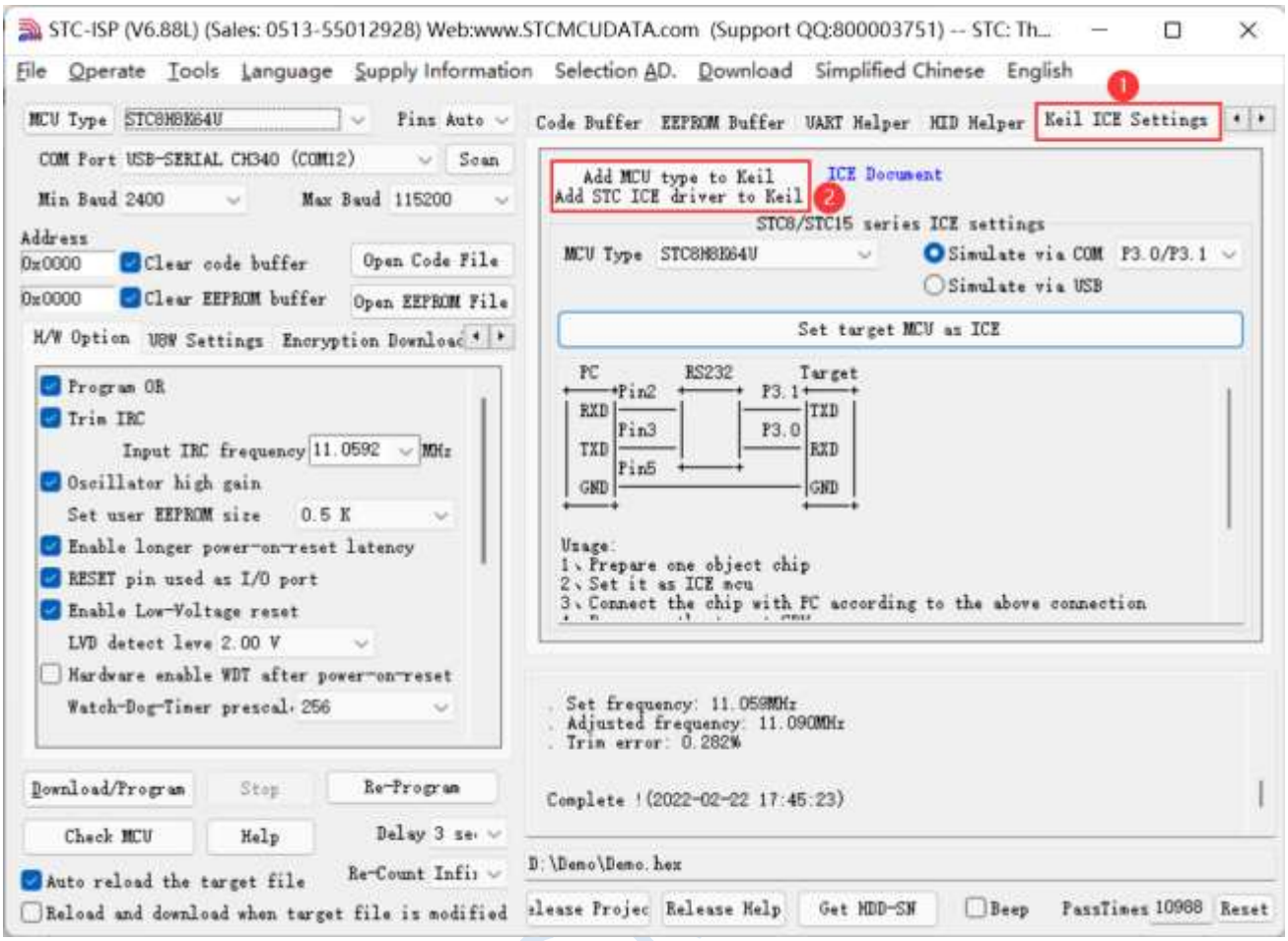

9. Wait for the download to end. If the download is successful, the prompt box in the lower right corner will display "Complete!"



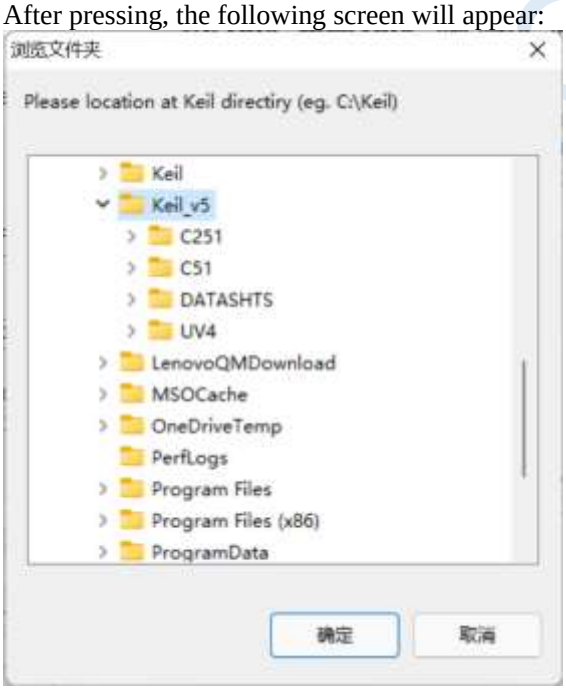

# H.4.5 Use STC universal USB to serial port tool to simulate user code

The current STC simulation is based on the Keil environment, so if you need to use the STC universal USB to serial port tool to simulate user code, you must install the Keil software.

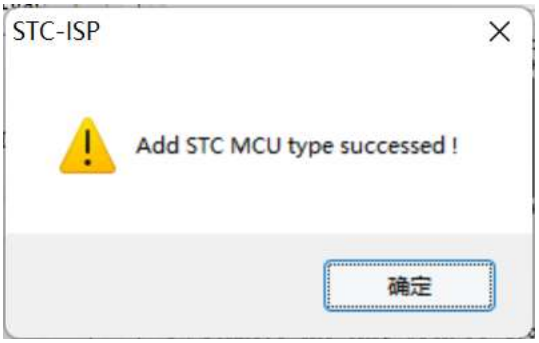
After the Keil software is installed, you also need to install the STC simulation driver. The installation steps of STC's simulation driver are as follows:
Firstly, open STC-ISP to download the software;
Then click the "Add MCU Type to Keil / Add STC ICE driver to Keil" on the "Keil ICE settings" page in the functional area on the right side of the software:



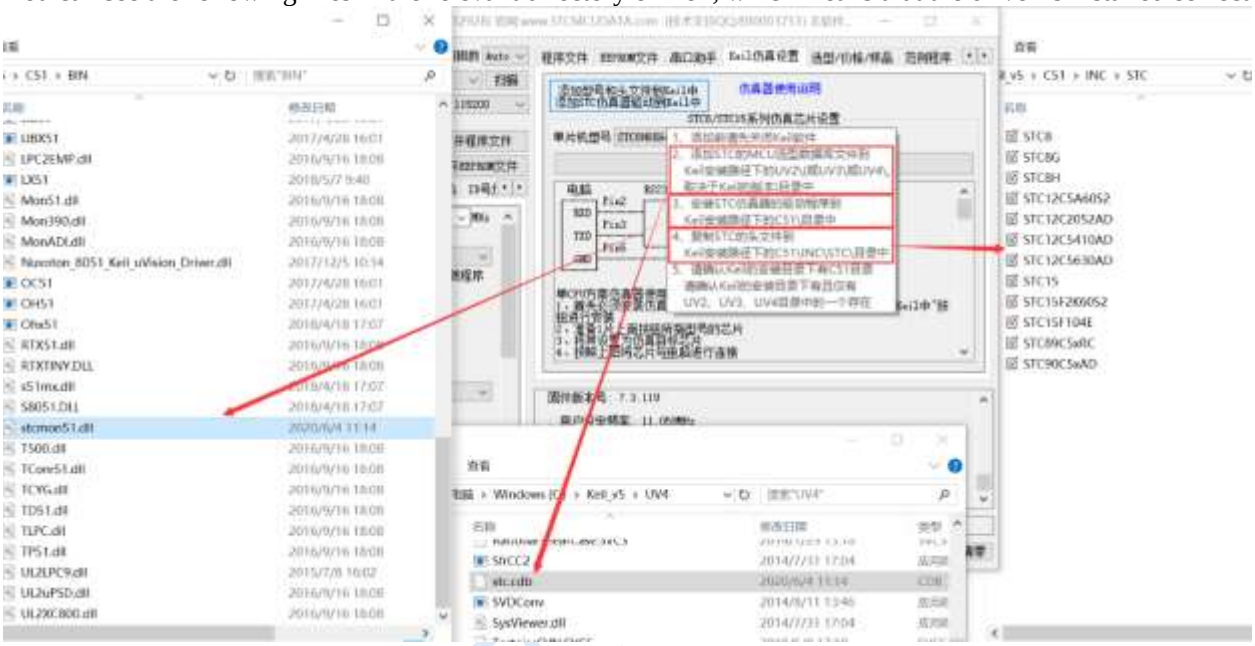

After pressing, the following screen will appear:




Locate the directory to the installation directory of the Keil software, and then confirm.
After the installation is successful, the following prompt box will pop up:

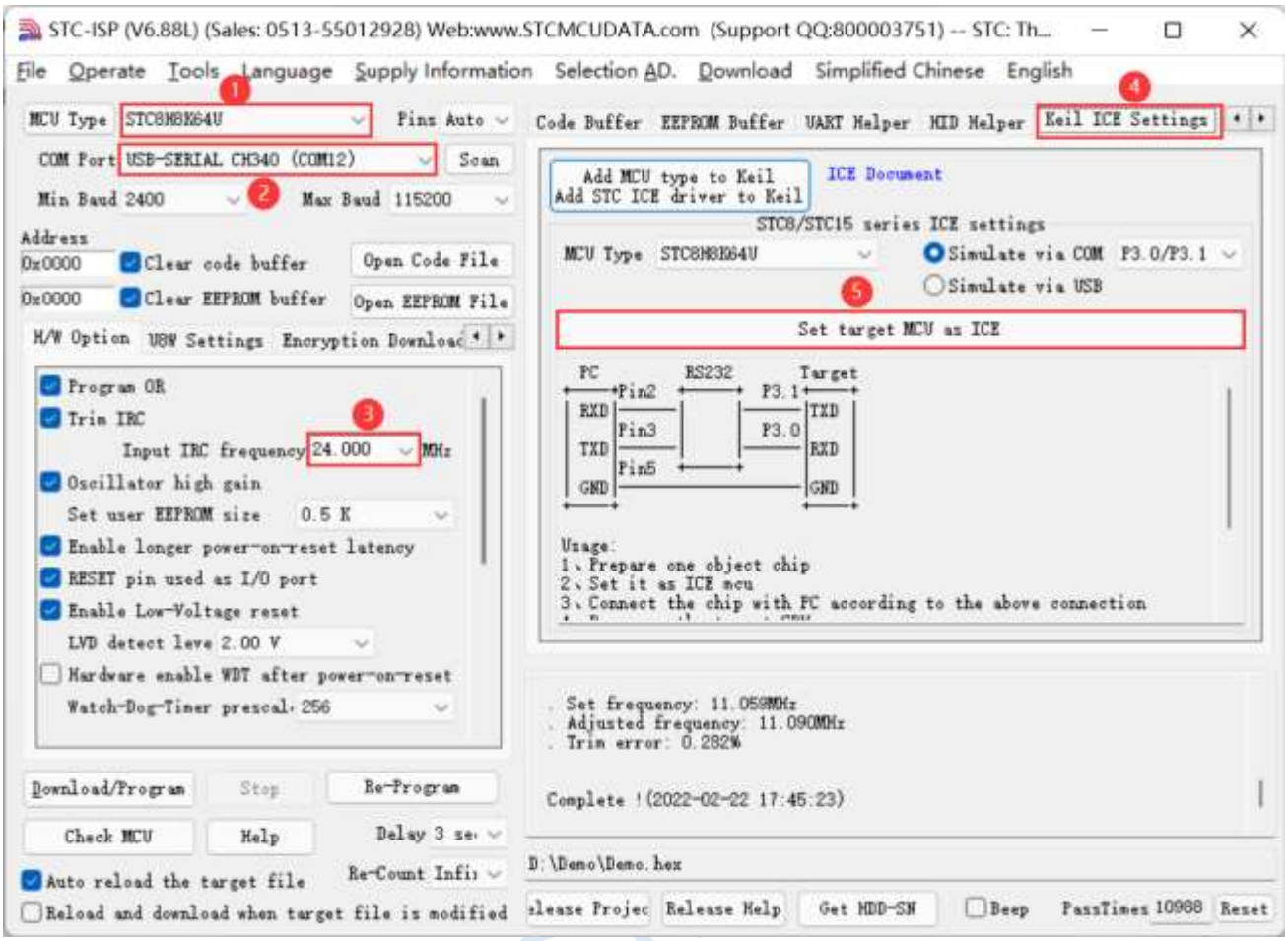



You can see the following files in the relevant directory of Keil, which means that the driver is installed correctly.

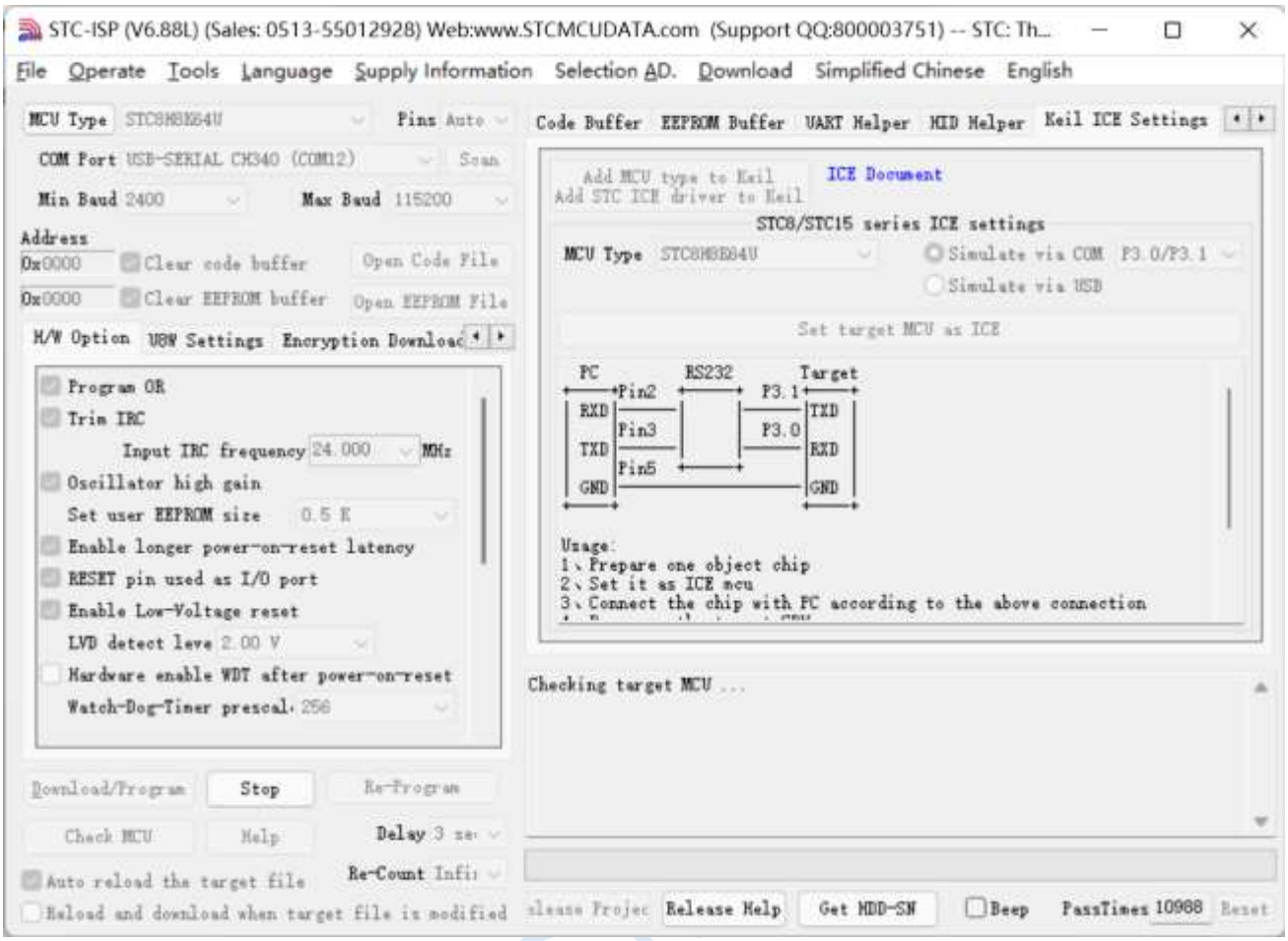


Since in the default state, the main control chip of STC is not an emulation chip and has no emulation function, if simulation is needed, the main control chip of STC needs to be set as an emulation chip.
The steps of making a simulation chip are as follows:

Firstly, use STC universal USB to serial port tool to connect the MCU to the computer;
Then open STC's ISP to download the software, and select the serial port number corresponding to the serial port tool in the serial port number drop-down list;
Select the MCU model;
Select the IRC frequency of the user program to run, and the frequency selected when making the simulation chip is consistent with the frequency set by the simulated user program, in order to achieve the real running effect.

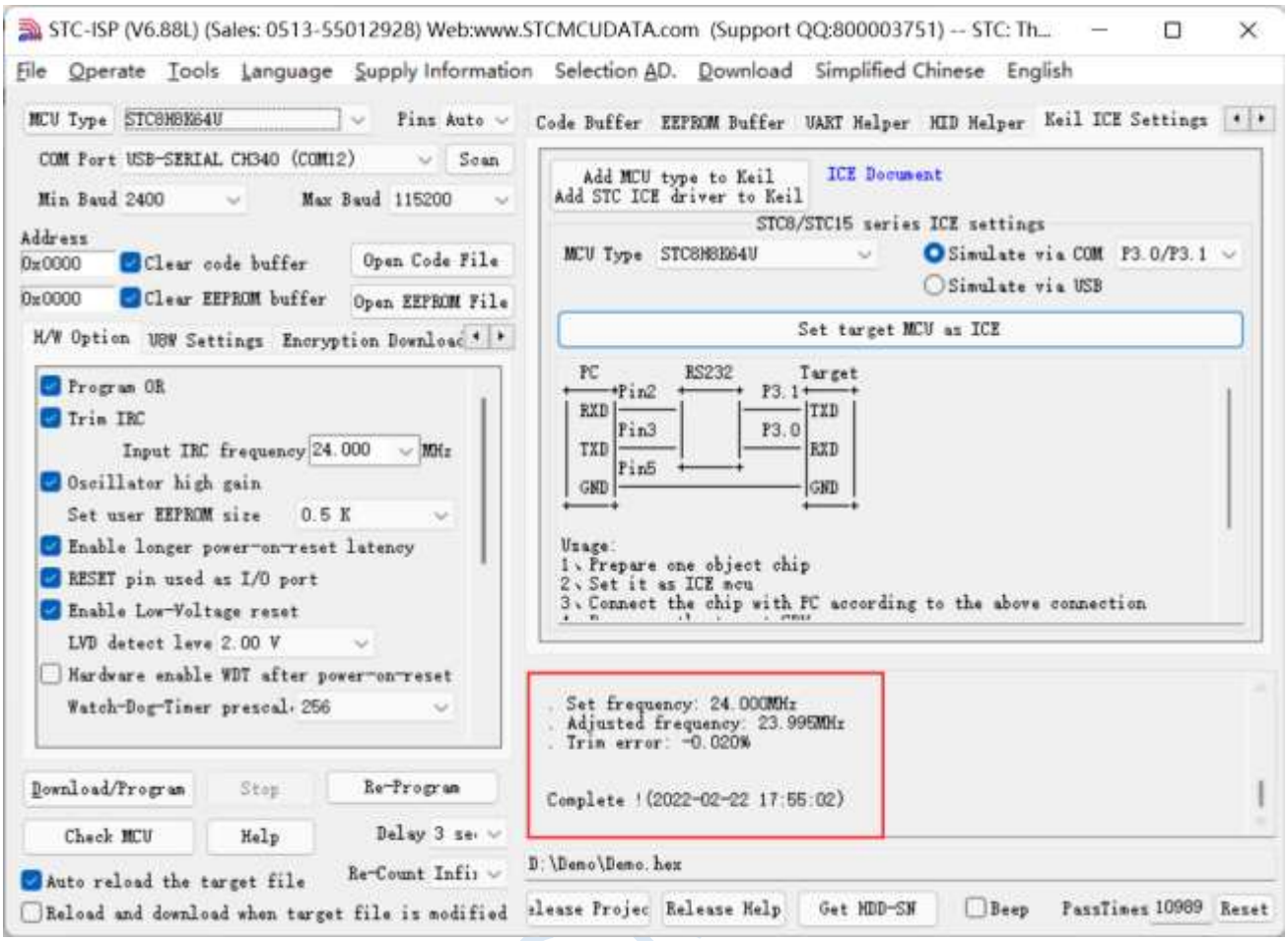



Then click the "Set target MCU as ICE" button on the "Keil ICE Settings" page in the right functional area of the software,After pressing, the following screen will appear:



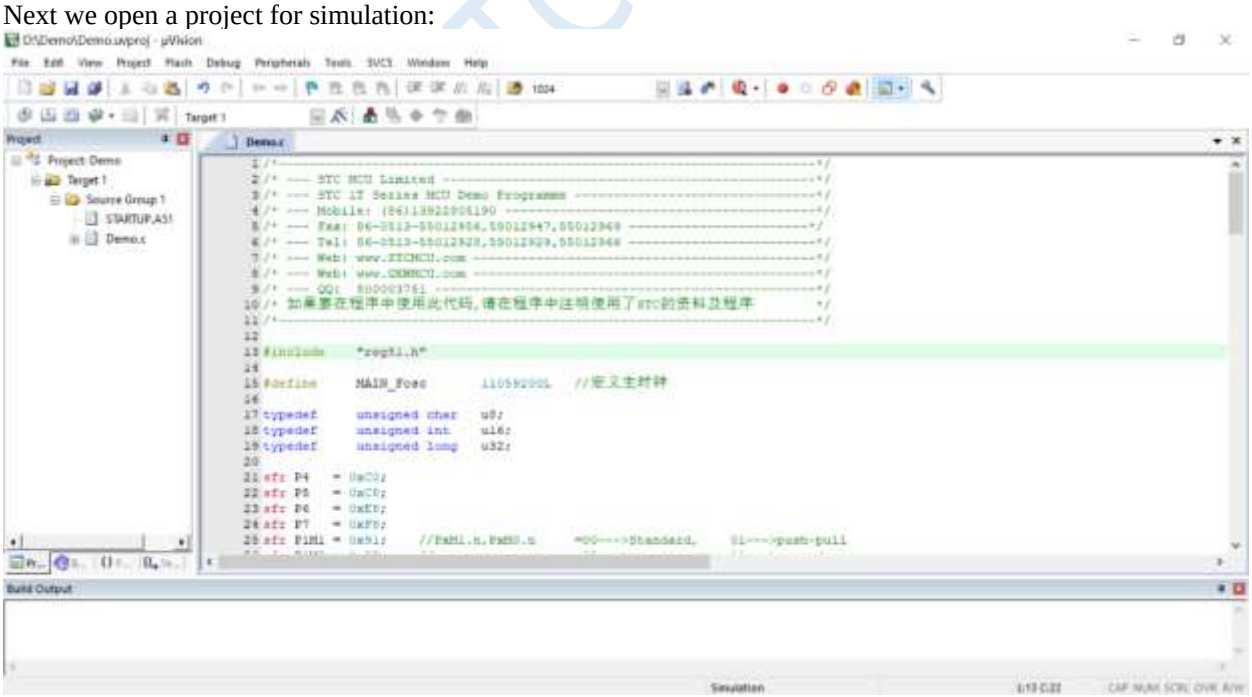

Next, you need to press the "power switch" on the STC Universal USB to Serial Tool to supply power to the MCU [cold start], and you can start to make the simulation chip.

If the setting is successful, the following screen will appear:




At this point, the simulation chip has been made successfully.

Next we open a project for simulation:

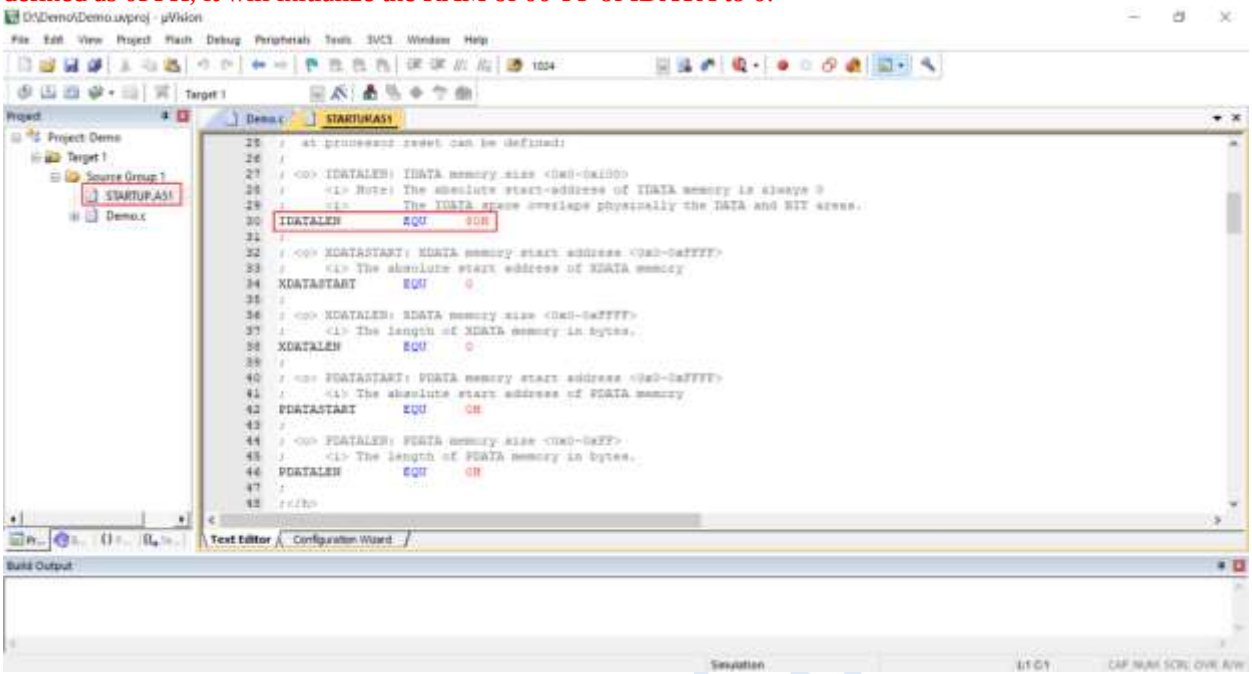


Then make the following project settings:

An additional note:

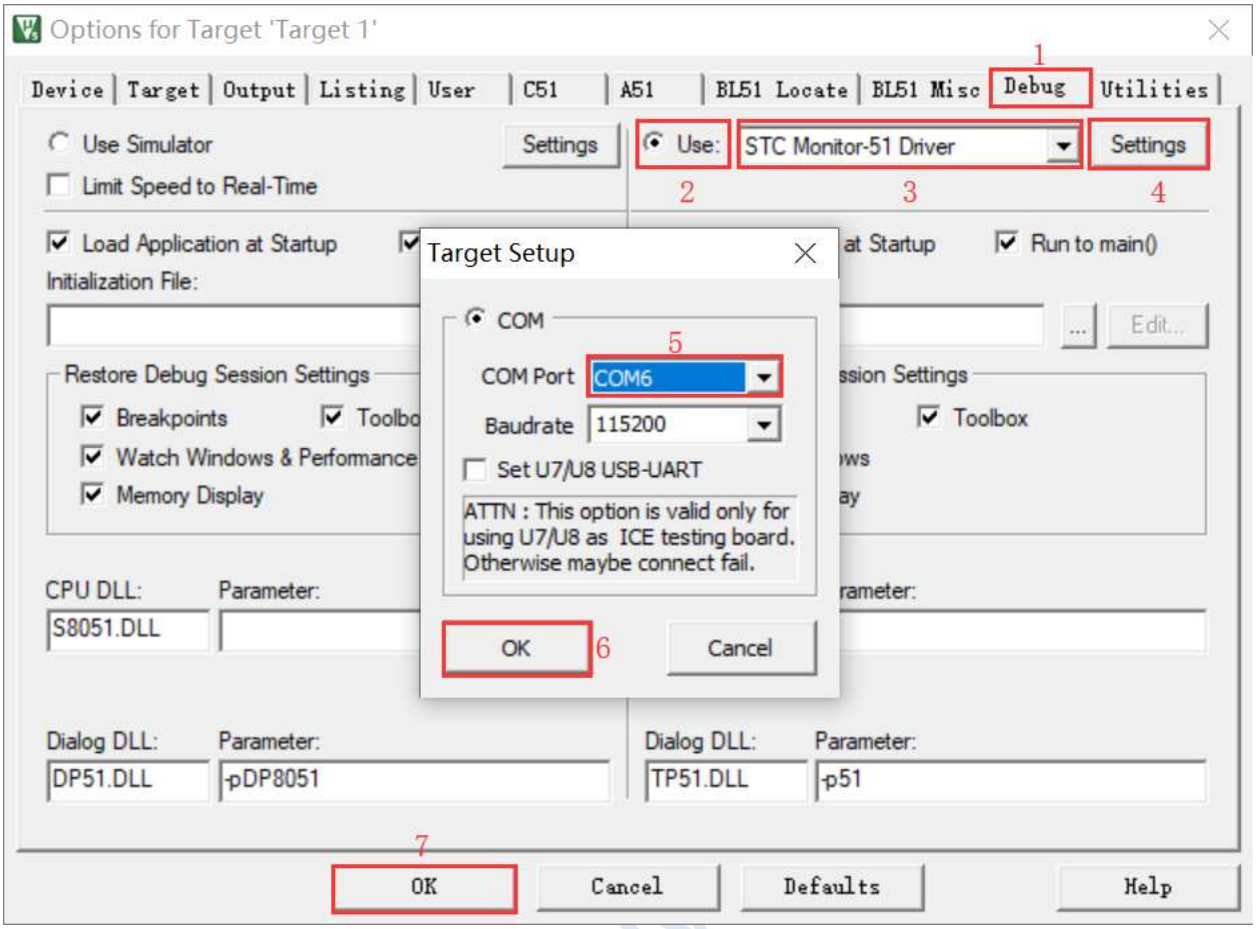
When a C language project is created and the startup file "STARTUP.A51" is added to the project, there is a macro

definition named "IDATALEN", which is used to define the size of IDATA. The default value is 128, which is 80H in hexadecimal, and it is also the size of IDATA that needs to be initialized to 0 in the startup file. So when IDATA is defined as 80H, the code in STARTUP.A51 will initialize the RAM of 00-7F of IDATA to 0; similarly, if IDATA is defined as 0FFH, it will initialize the RAM of 00-FF of IDATA to 0.

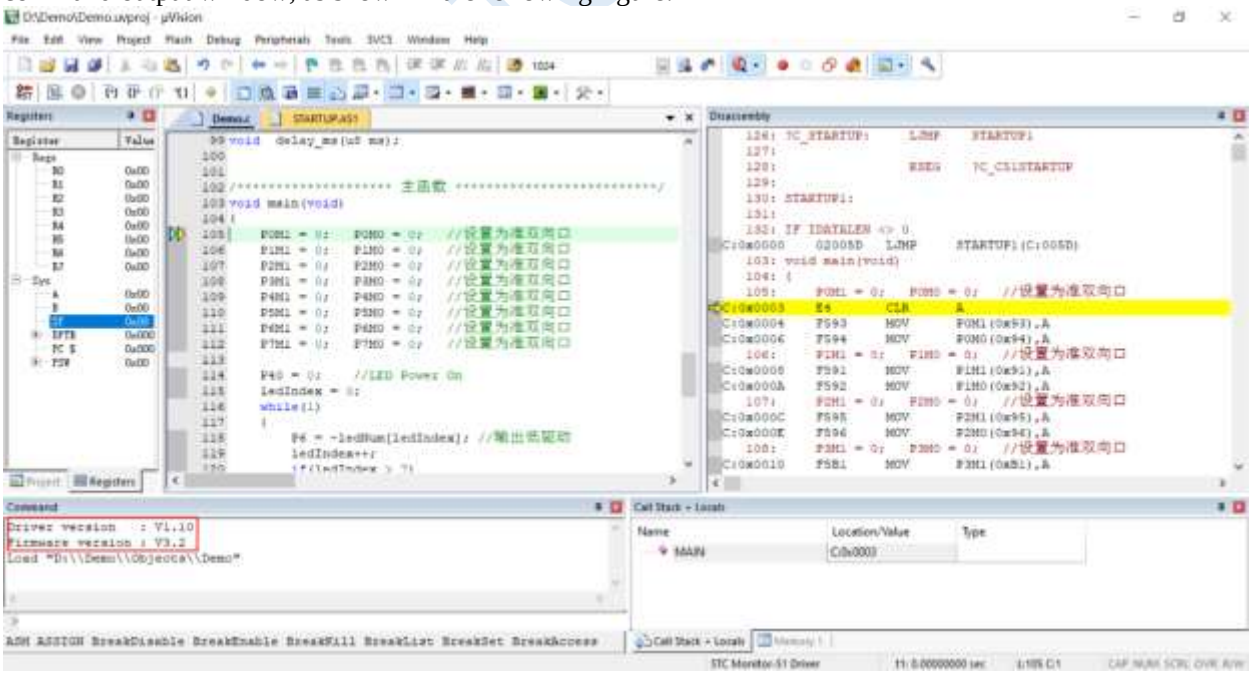



The IDATA size of the STC8H series microcontroller we selected is 256 bytes (00-7F DATA and 80H-FFH IDATA), but because the last 17 bytes of RAM have written ID numbers and related test parameters, If users need to use this part of data in the program, they must not define IDATALEN as 256.

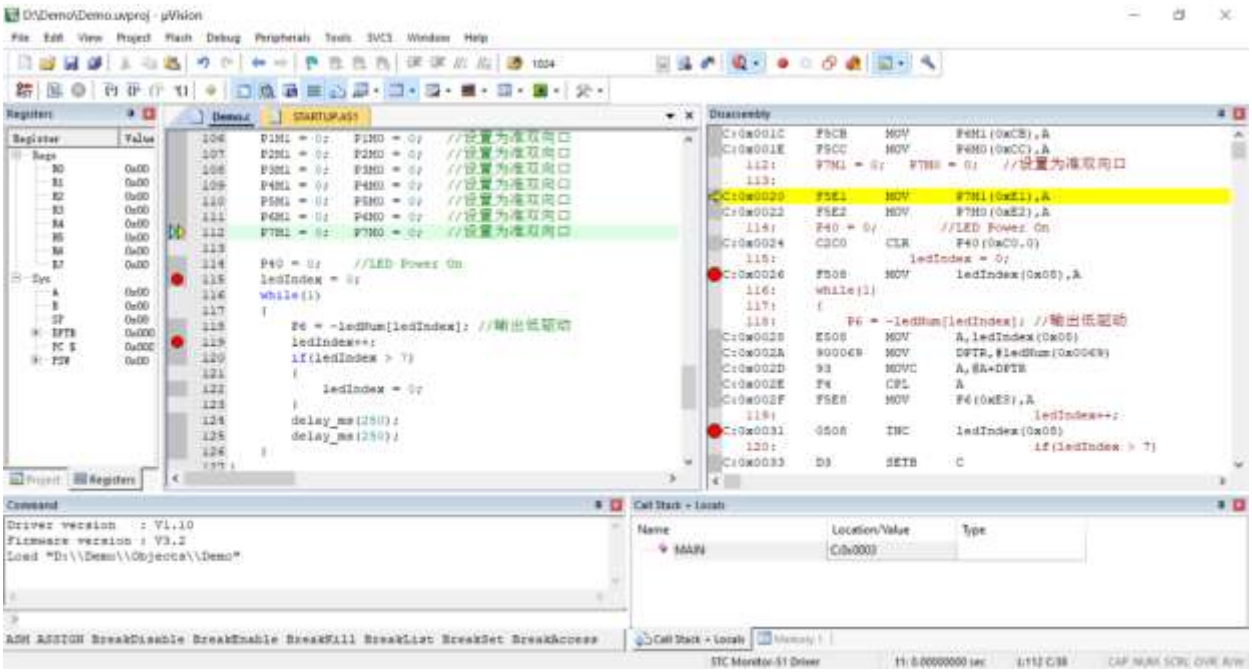
Press the shortcut key "Alt+F7" or select "Option for Target 'Target1'" in the menu "Project" to configure the project in the "Option for Target 'Target1'" dialog box:
Step 1. Enter the project setting page and select the "Debug" setting page;
Step 2. Select the hardware emulation "Use …" on the right;
Step 3. Select "STC Monitor-51 Driver" item in the simulation driver drop-down list;
Step 4. Click the "Settings" button to enter the serial port settings screen;
Step 5: Set the port number and baud rate of the serial port. The serial port number should be the serial port corresponding to the STC universal USB to serial port tool. The baud rate is generally 115200 or 57600.
Make sure to complete the simulation settings.
The detailed steps are shown in the figure below:




After finishing all the above work, you can press "Ctrl+F5" in Keil software to start simulation debugging.
If the hardware connection is correct, you will enter a debugging interface similar to the following, and display the current simulation driver version number and the current simulation monitoring code firmware version number in the command output window, as shown in the following figure:



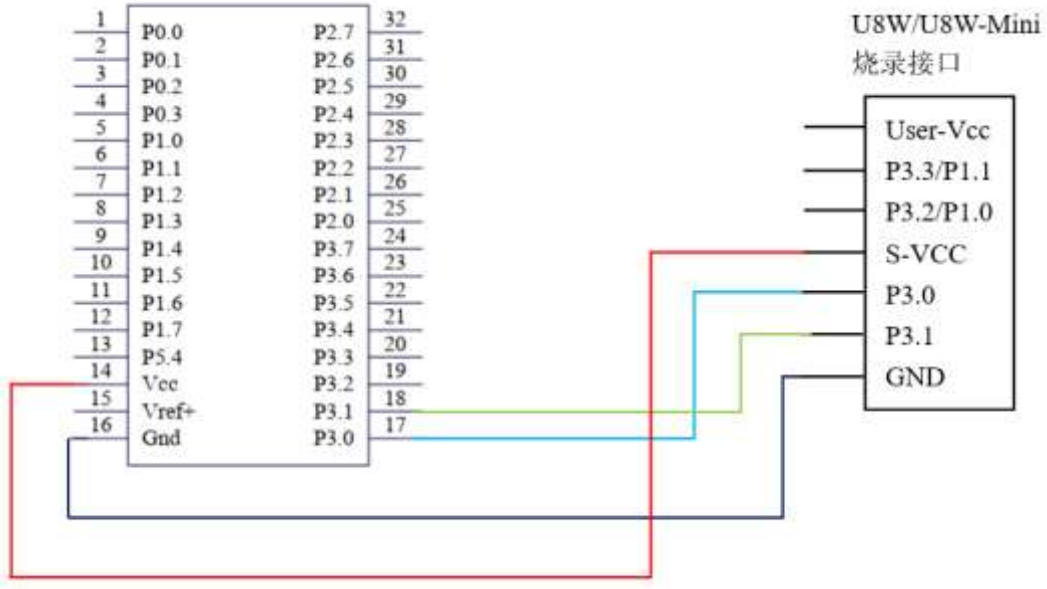

During the simulation debugging process, you can perform multiple operations such as resetting, running at full speed, single stepping, and setting breakpoints.



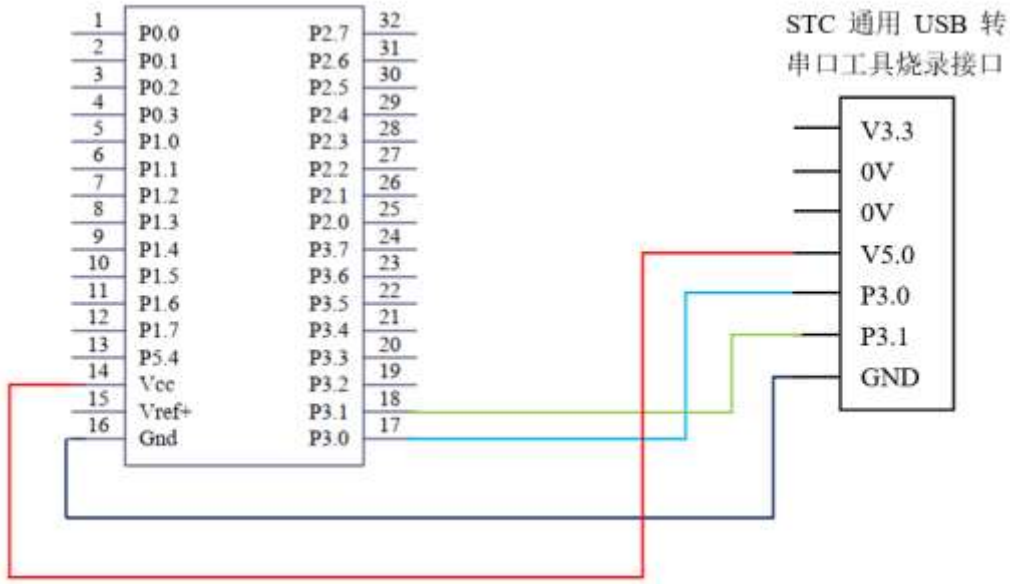
As shown in the figure above, multiple breakpoints can be set in the program, and the maximum number of breakpoint settings currently allowed is 20 (in theory, any number can be set, but setting too many breakpoints will affect the speed of debugging).

## H.5 Application circuit diagram

### H.5.1 U8W tool application reference circuit diagram



# H.5.2 STC Universal USB to Serial Tool Application Reference Circuit Diagram



# Appendix I STC Emulation Instruction Manual

## I.1 Overview

All STC8G/8H series microcontrollers support online emulation, including downloading user code, chip reset, full-speed operation, single-step operation, setting breakpoints (the number of theoretical breakpoints is unlimited, but in order to improve the emulation efficiency, the current limit is up to 20 breakpoints), viewing variables, etc. The emulation operation is convenient for users to debug the code and find logical errors in the code, thereby shortening the project development cycle.

The emulation interface can be USB or serial port, the microcontroller itself is an emulator, and all emulation functions can be realized without an additional emulator. The corresponding USB port or serial port is originally a dedicated port for emulation, but when the emulation function is turned off, the user can freely use the emulation interface as GPIO, USB or serial port.

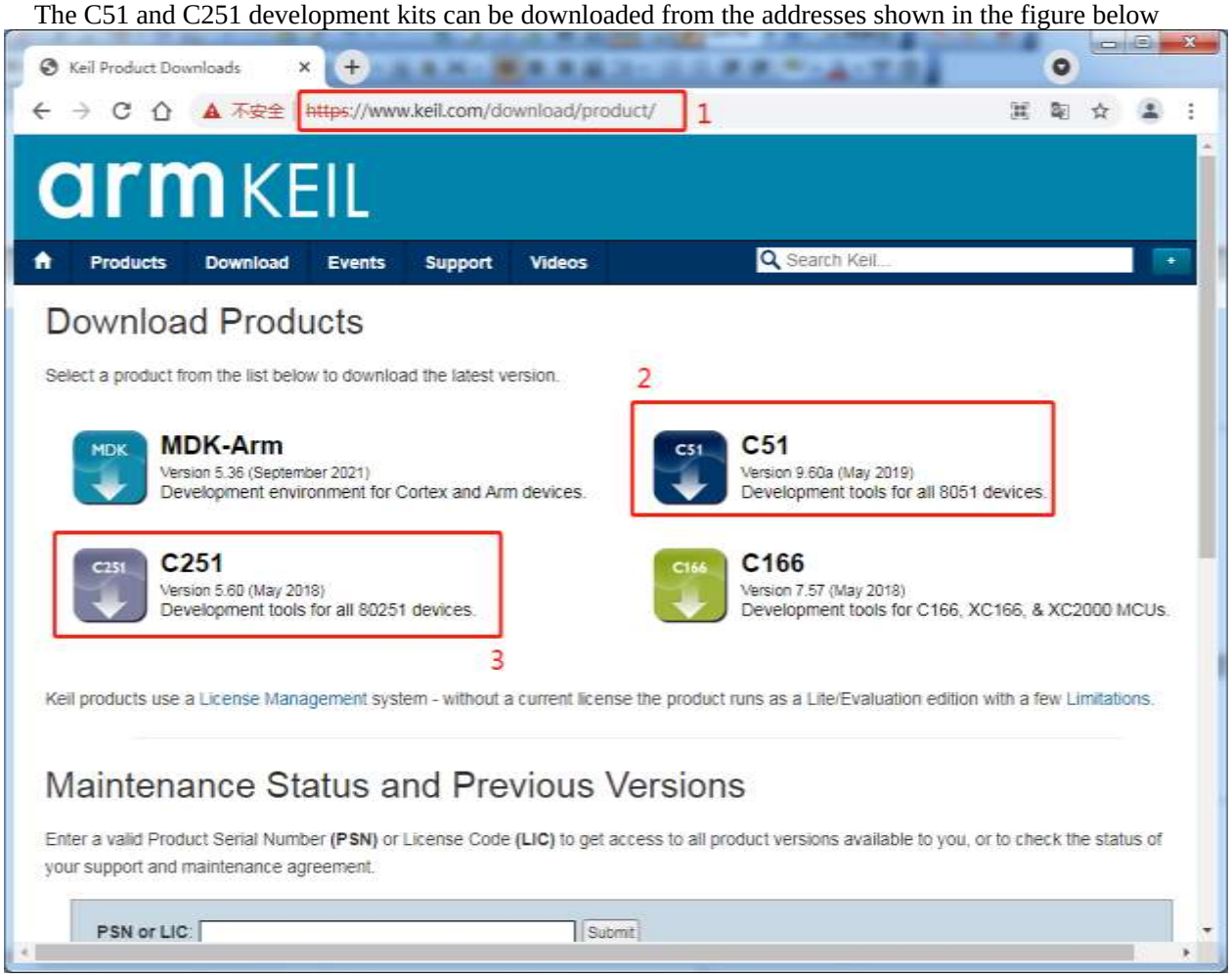
At present, the emulation mode of all MCUs is the software monitoring emulation mode, which will occupy part of the system resources. The resources occupied by each series of MCU simulation are shown in the following table **(NO Flash memory is occupied):**

| MCU Series | Emulation Port | Resource occupied | |
|---|---|---|---|
| | | Port | Data memory (XDATA) |
| STC8H8K64U family-B version | USB | D+，D- | 768 bytes (1D00H-1FFFH) |
| | UART | P3.0，P3.1 | 768 bytes (1D00H-1FFFH) |
| | | P3.6，P3.7 | |
| | | P1.6，P1.7 | |
| | | P4.3，P4.4 | |
| STC8H8K64U family-A version | UART | P3.0，P3.1 | 768 bytes (1D00H-1FFFH) |
| | | P3.6，P3.7 | |
| | | P1.6，P1.7 | |
| | | P4.3，P4.4 | |
| STC8H4K64T family | UART | P3.0，P3.1 | 768 bytes (0D00H-0FFFH) |
| STC8H3K64S4 family | UART | P3.0，P3.1 | 768 bytes (0900H-0BFFH) |
| STC8H1K16 family | UART | P3.0，P3.1 | 768 bytes (0100H-03FFH) |
| STC8H1K08 family | UART | P3.0，P3.1 | 768 bytes (0100H-03FFH) |
| STC8G2K64S4 family | UART | P3.0，P3.1 | 768 bytes (0500H-07FFH) |
| STC8G1K08 family | UART | P3.0，P3.1 | 768 bytes (0100H-03FFH) |
| STC8C2K64S4 family | UART | P3.0，P3.1 | 768 bytes (0500H-07FFH) |
| STC8A8K64D4 family | UART | P3.0，P3.1 | 768 bytes (1D00H-1FFFH) |
| STC8A8K64S4A12 family | UART | P3.0，P3.1 | 768 bytes (1D00H-1FFFH) |
| STC8F2K64S4 family | UART | P3.0，P3.1 | 768 bytes (0500H-07FFH) |
| STC8F1K08S2 family | UART | P3.0，P3.1 | 768 bytes (0100H-03FFH) |
| IAP15W4K58S4 | UART | P3.0，P3.1 | 768 bytes (0D00H-0FFFH) |
| IAP15F2K61S2 | UART | P3.0，P3.1 | 768 bytes (0500H-07FFH) |

## I.2 Install Keil software

The emulation of STC microcontroller is based on the Keil development environment, so before the emulation, the Keil software must be installed.

The C51 and C251 development kits can be downloaded from the addresses shown in the figure below



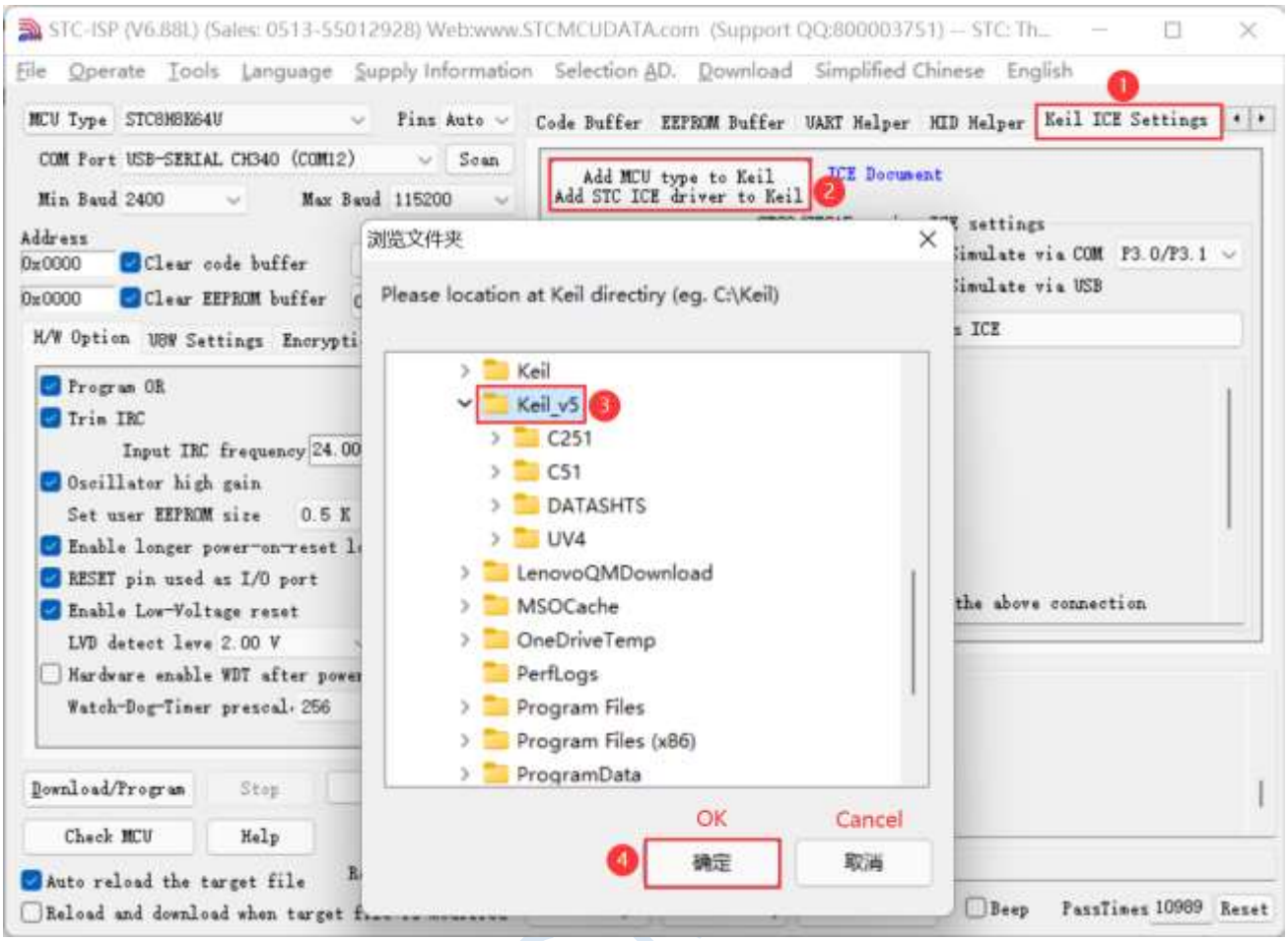

**Note: The latest Keil-UV5 software does not include 8051 and 80251 toolkits by default, and must be downloaded and installed manually.**

# I.3 Install the emulation driver

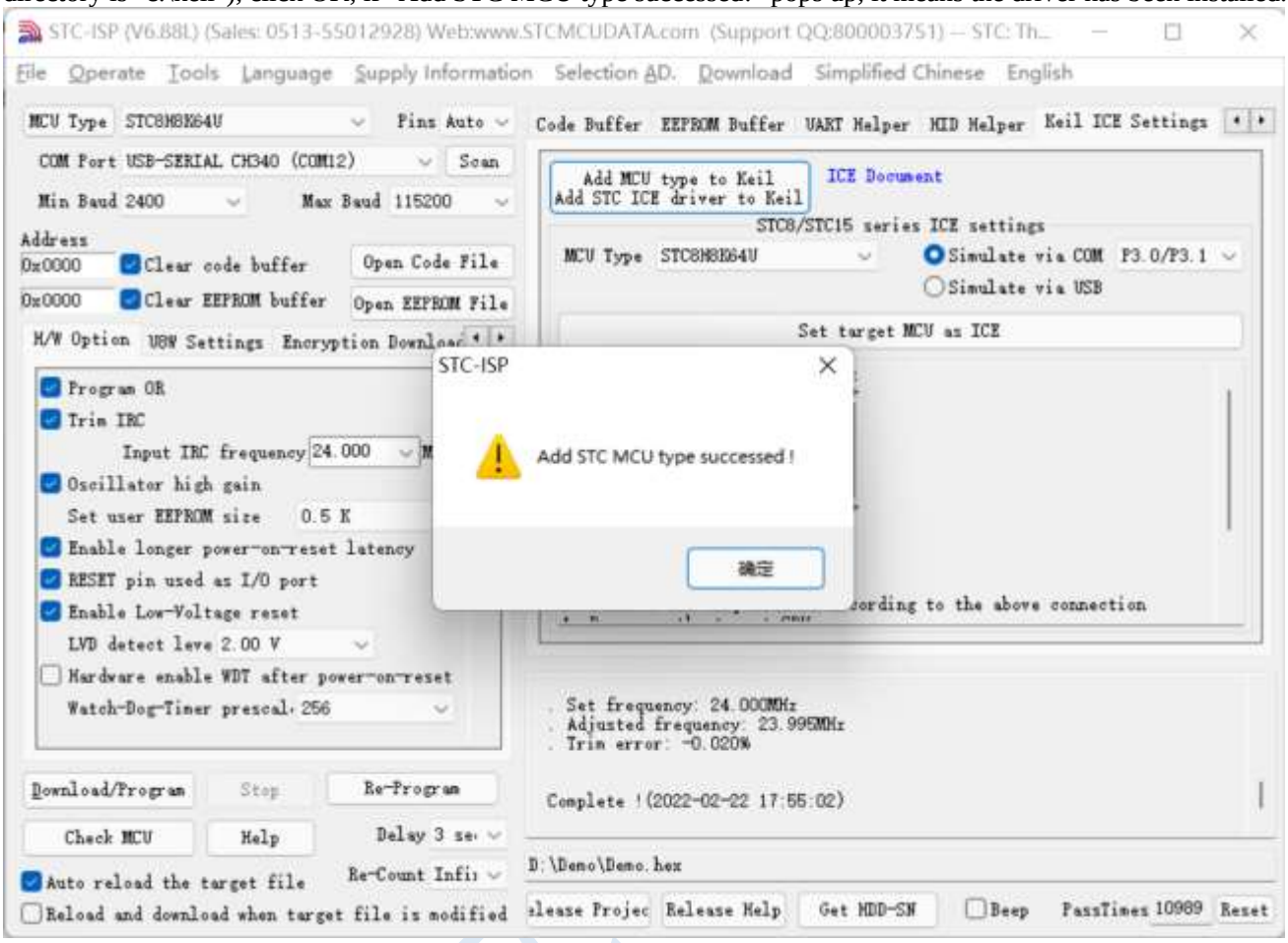
After the Keil development environment is installed, you also need to install the STC-specific emulation driver. Proceed as follows:
Firstly, download the latest STC-ISP download software from the STC official website


After downloading and unzipping, open the "stc-isp-vxx.exe" executable in the package.


Click the "Add MCU Type to Keil..." button in the "Keil ICE Settings" page of the download software (Figure below)


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
MCU Type STC8H8K64U
Pins Auto
Code Buffer EEPROM Buffer UART Helper MID Helper Keil ICE Settings
COM Port USB-SERIAL CH340 (COM12)
Scan
Min Baud 2400
Max Baud 115200
Add MCU type to Keil
Add STC ICE driver to Keil
ICE Document
Address
0x0000
Clear code buffer
0x0000
Clear EEPROM buffer
H/W Option
Program OR
Trim IRC
Input IRC frequency
Oscillator high gain
Set user EEPROM size
0.5 K
Enable longer power-on-reset
RESET pin used as I/O port
Enable Low-Voltage reset
LVD detect leve 2.00 V
Hardware enable WDT after power
Watch-Dog-Timer prescal 256
Download/Program
Stop
Check MCU
Help
Auto reload the target file
Beep
PassTimes 10989
Reset
浏览文件夹
Please location at Keil directiry (eg. C:\Keil)
Keil
Keil_v5
C251
C51
DATASHTS
UV4
LenovoQMDownload
MSOCache
OneDriveTemp
PerfLogs
Program Files
Program Files (x86)
ProgramData
OK
Cancel
确定
取消
1
2
3
4

In the pop-up "Browse Folder" window, select the Keil installation directory (usually Keil's installation directory is "c:\keil"), click OK, if "Add STC MCU type successed!" pops up, it means the driver has been installed. .



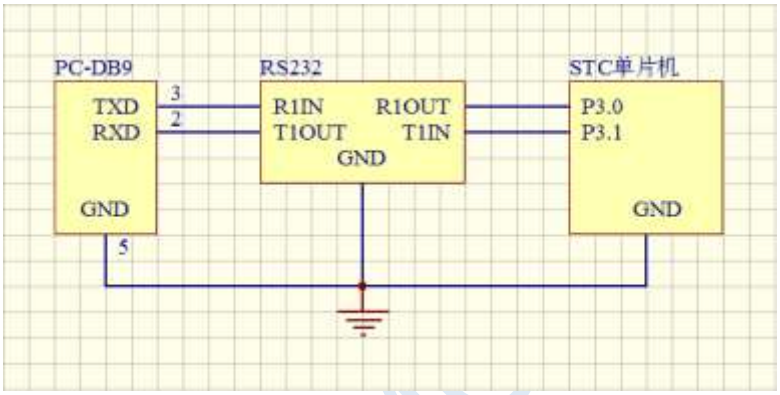
# I.4 Serial port emulation directly

## I.4.1 Make serial port emulation chip

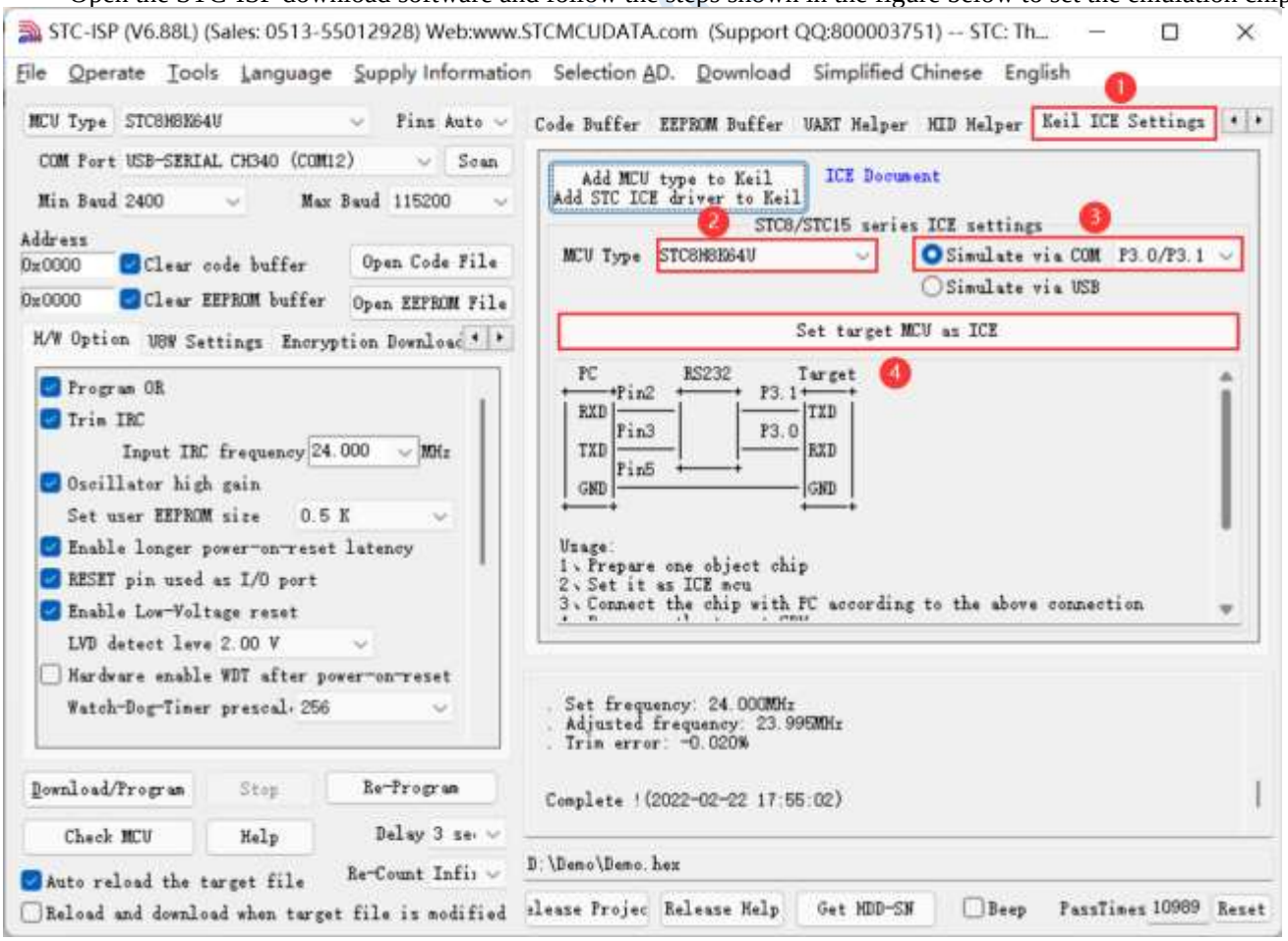
When the STC microcontroller is shipped, the emulation function is disabled by default. If you want to use the emulation function, you need to use the STC-ISP download software to set the target microcontroller as the emulation chip.

The setting steps are as follows:

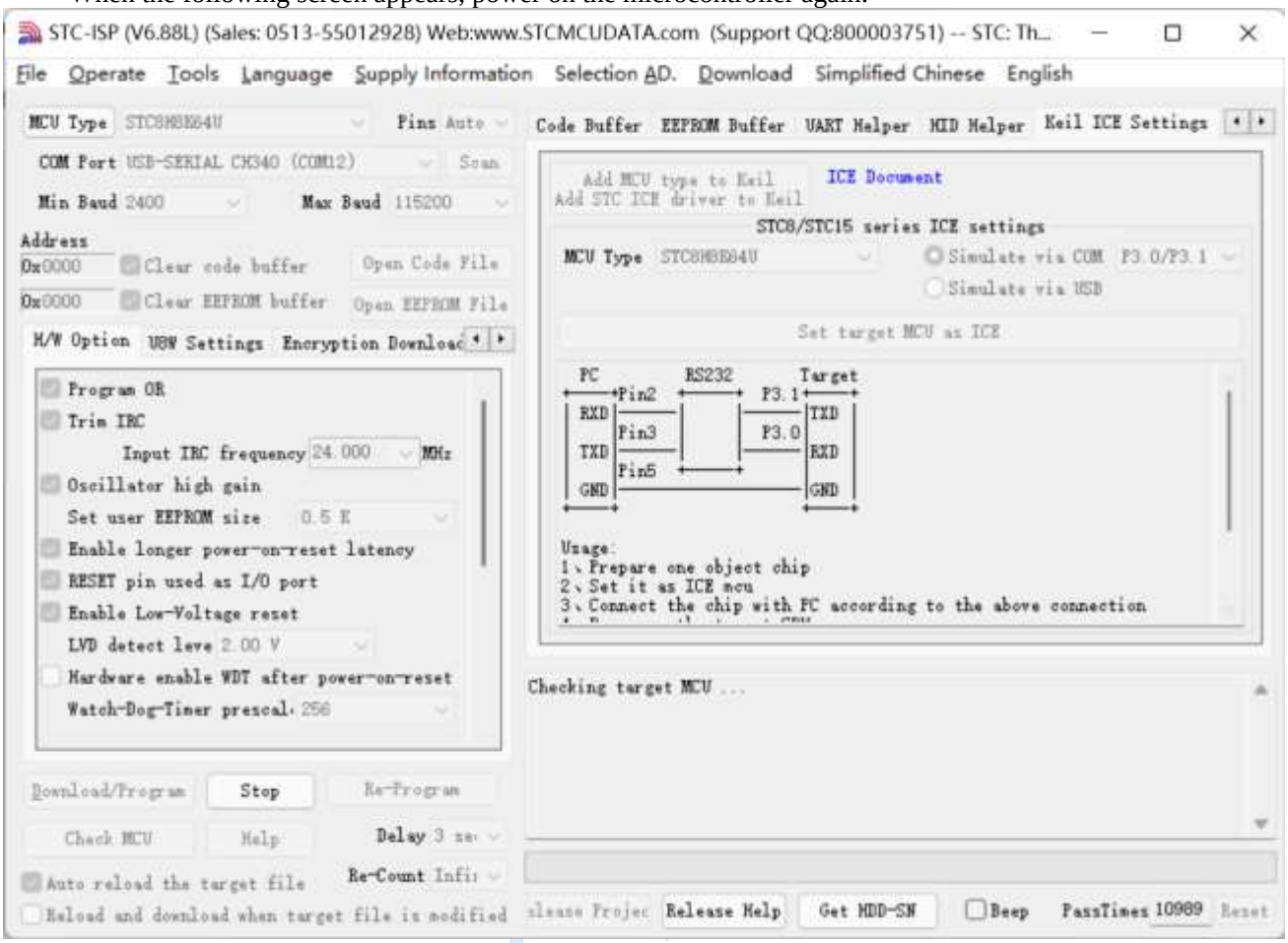
Firstly, connect the target chip to the serial port of the computer as shown in the figure below, and power off the microcontroller



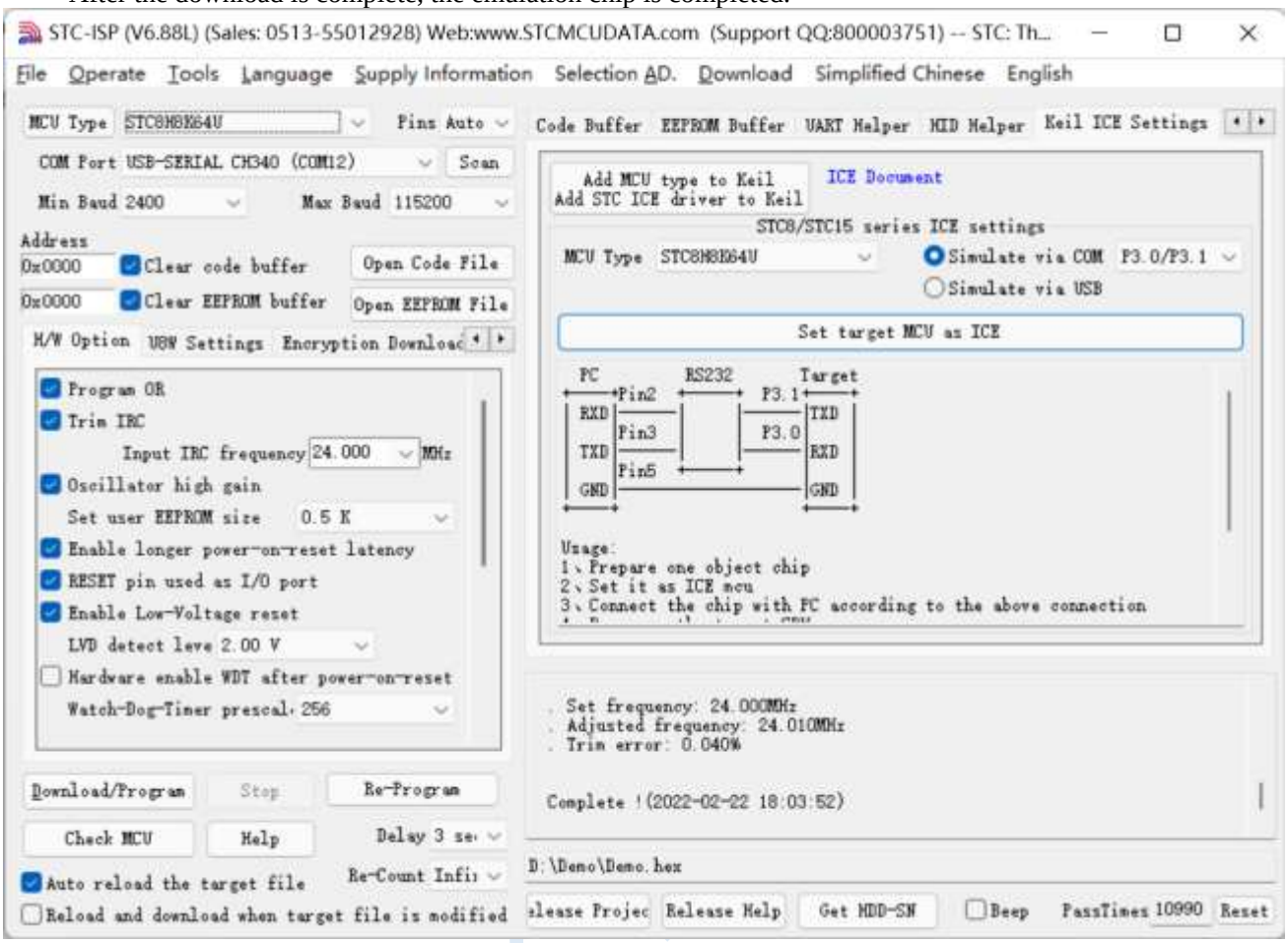

Open the STC-ISP download software and follow the steps shown in the figure below to set the emulation chip.



When the following screen appears, power on the microcontroller again.

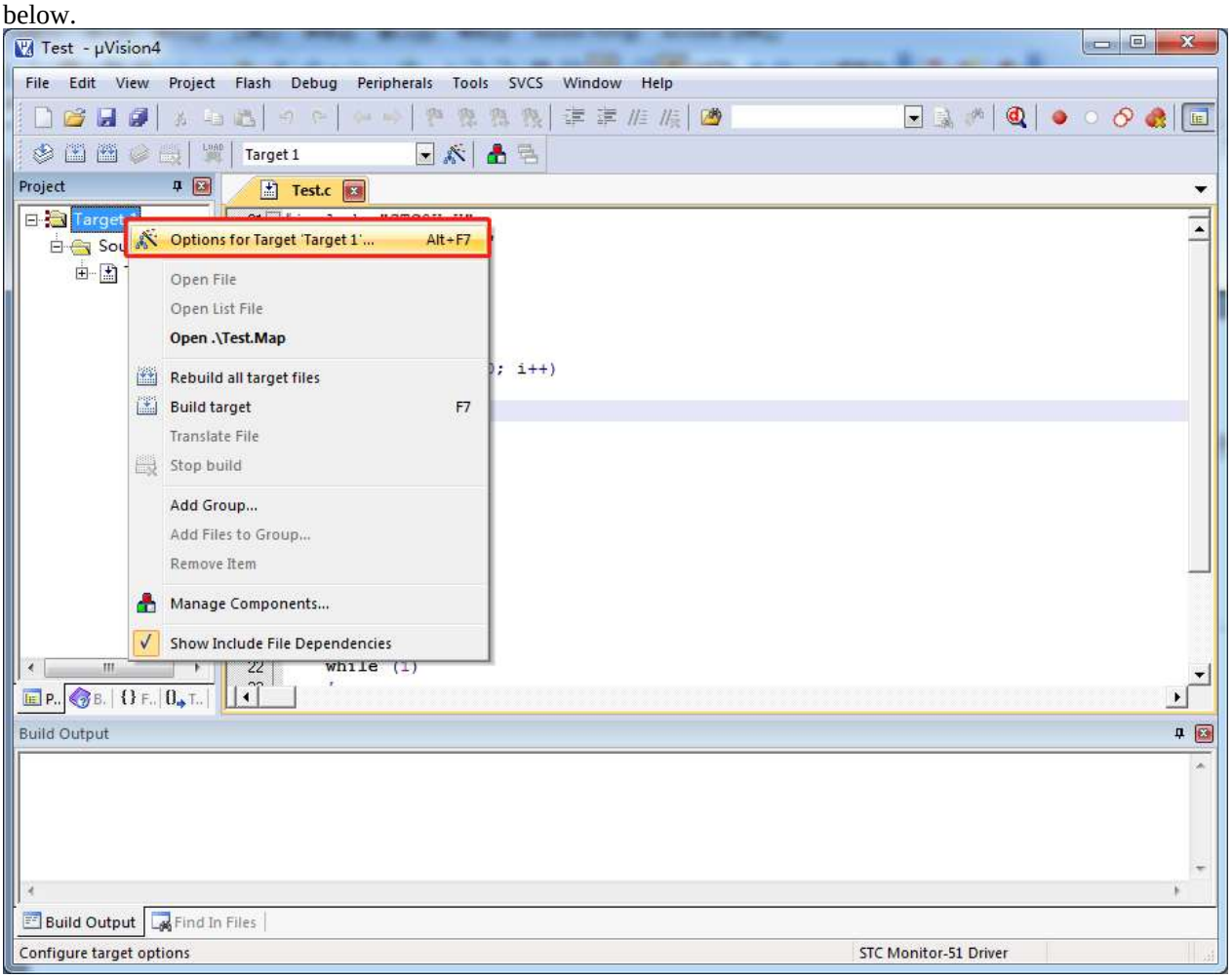


After the download is complete, the emulation chip is completed.

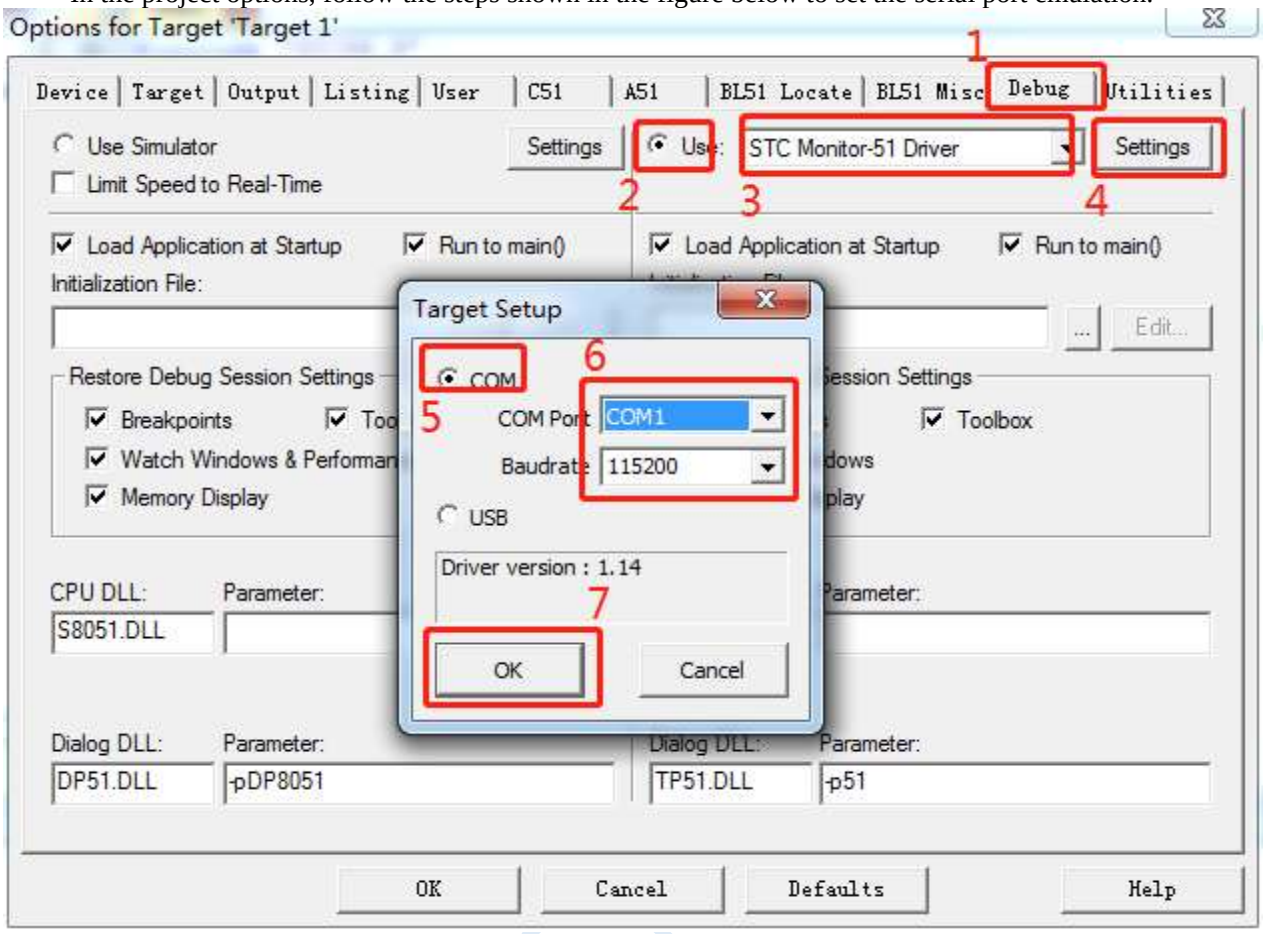


# I.4.2 Serial port emulation settings in Keil software

Open the project file in Keil software, and click "Options for ..." in the right-click menu as shown in the figure below.



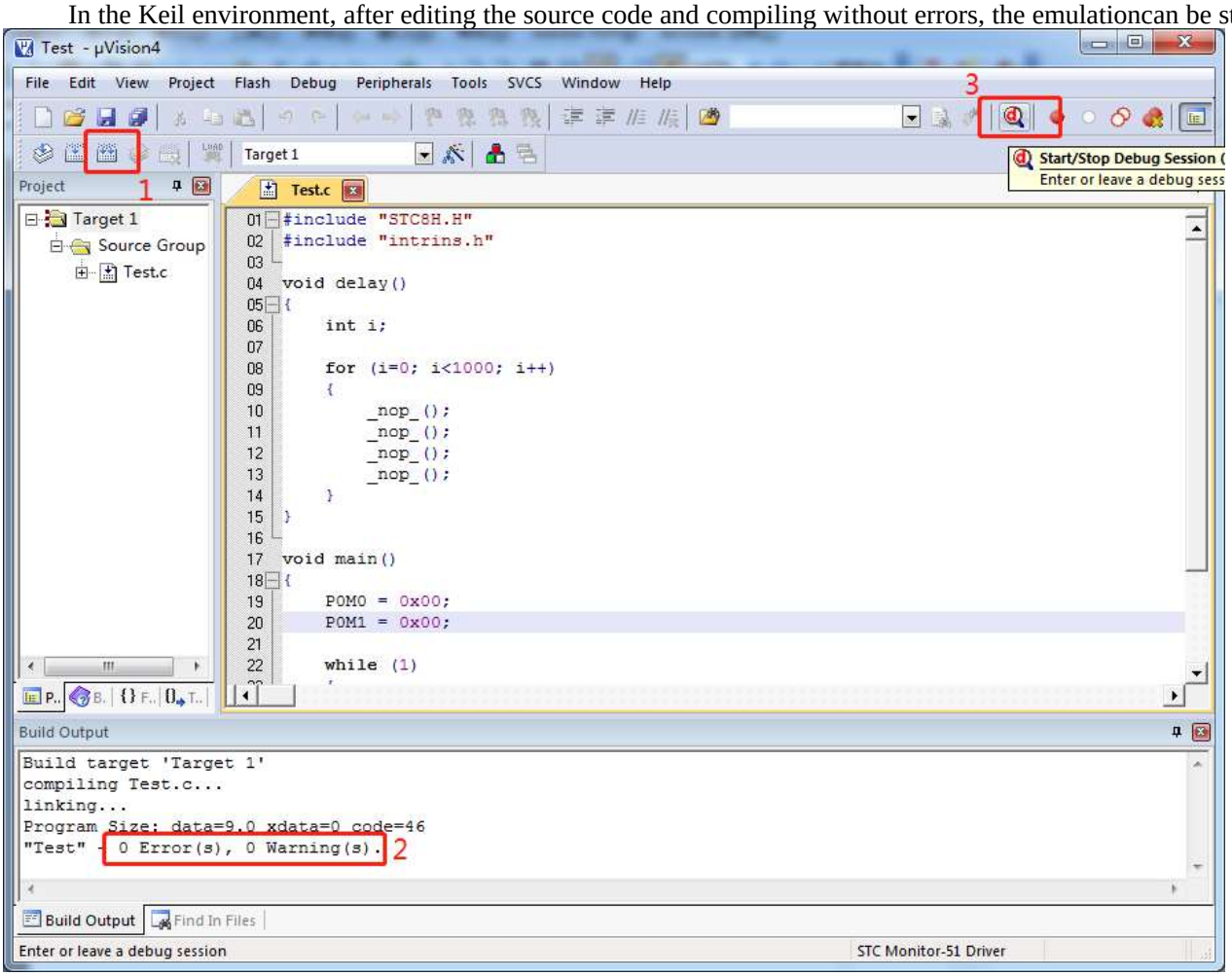
In the project options, follow the steps shown in the figure below to set the serial port emulation.

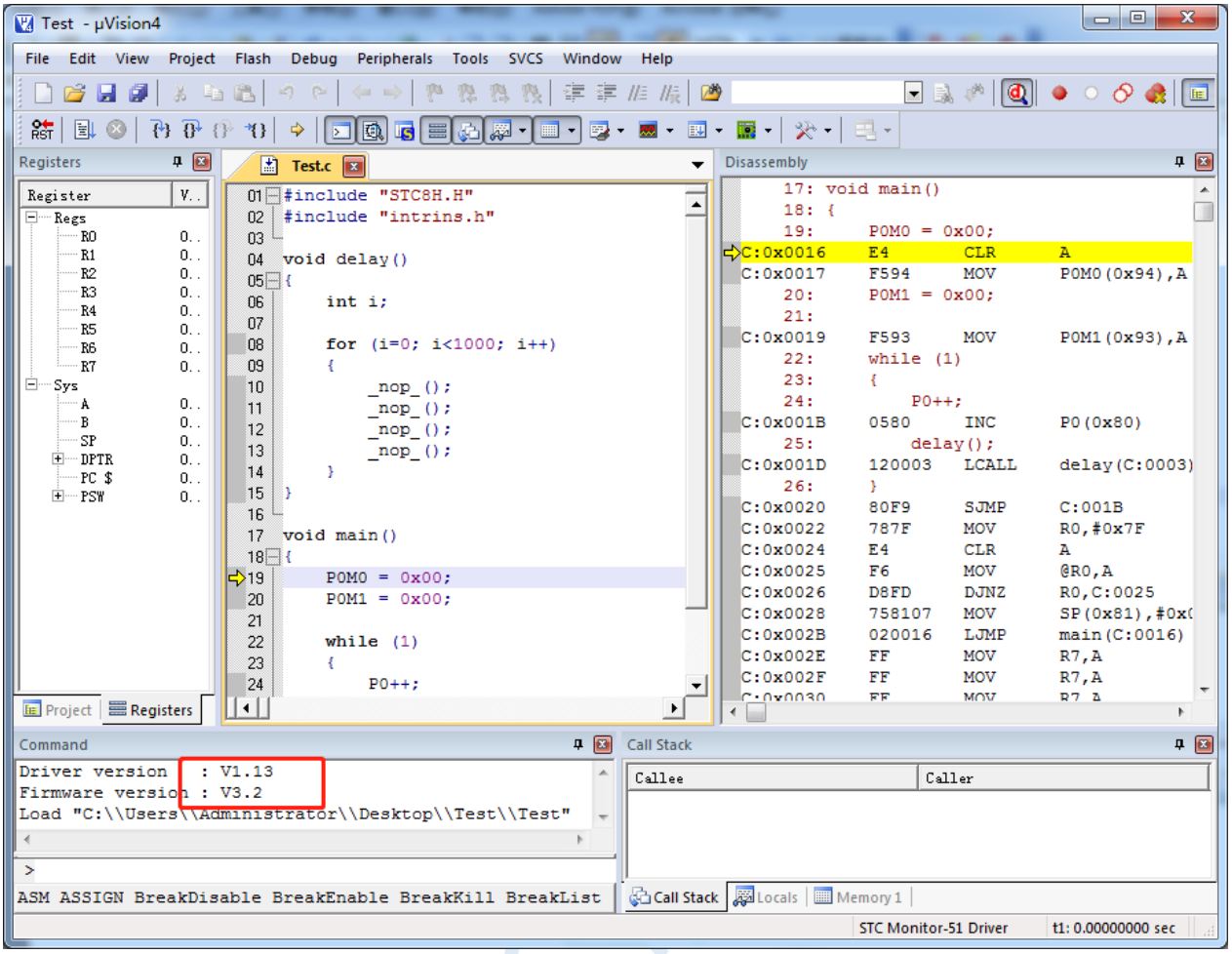



Note: Please select the serial port according to the actual connection, and the baud rate is generally 115200.

# I.4.3 Emulation using serial port in Keil software

In the Keil environment, after editing the source code and compiling without errors, the emulationcan be started.






If the chip making and connection are correct, the emulation driver version will be displayed as shown in the figure above, and the user code can be downloaded to the microcontroller correctly, and then debugging functions such as running, single step, and breakpoint can be performed.

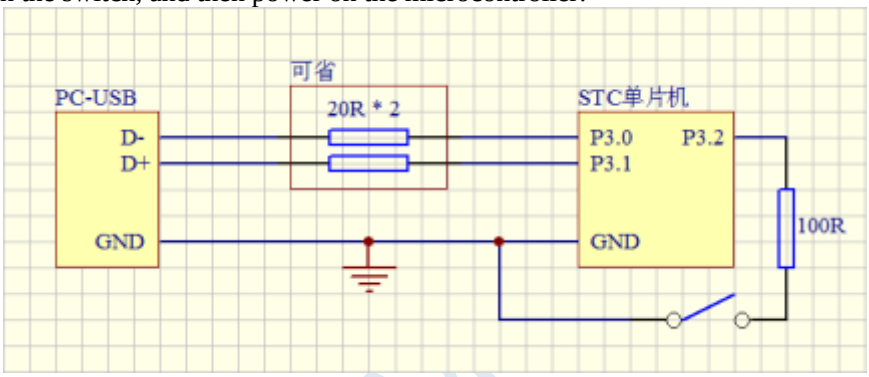
# I.5 USB direct emulation (currently only supported by STC8H8K64U-B version chip)

## I.5.1 Making a USB emulation chip

To make a USB emulation chip, you can use the serial ISP to make it according to the steps in Section 4.1, or use the USB-ISP method. This section will introduce how to use the USB-ISP to make it.

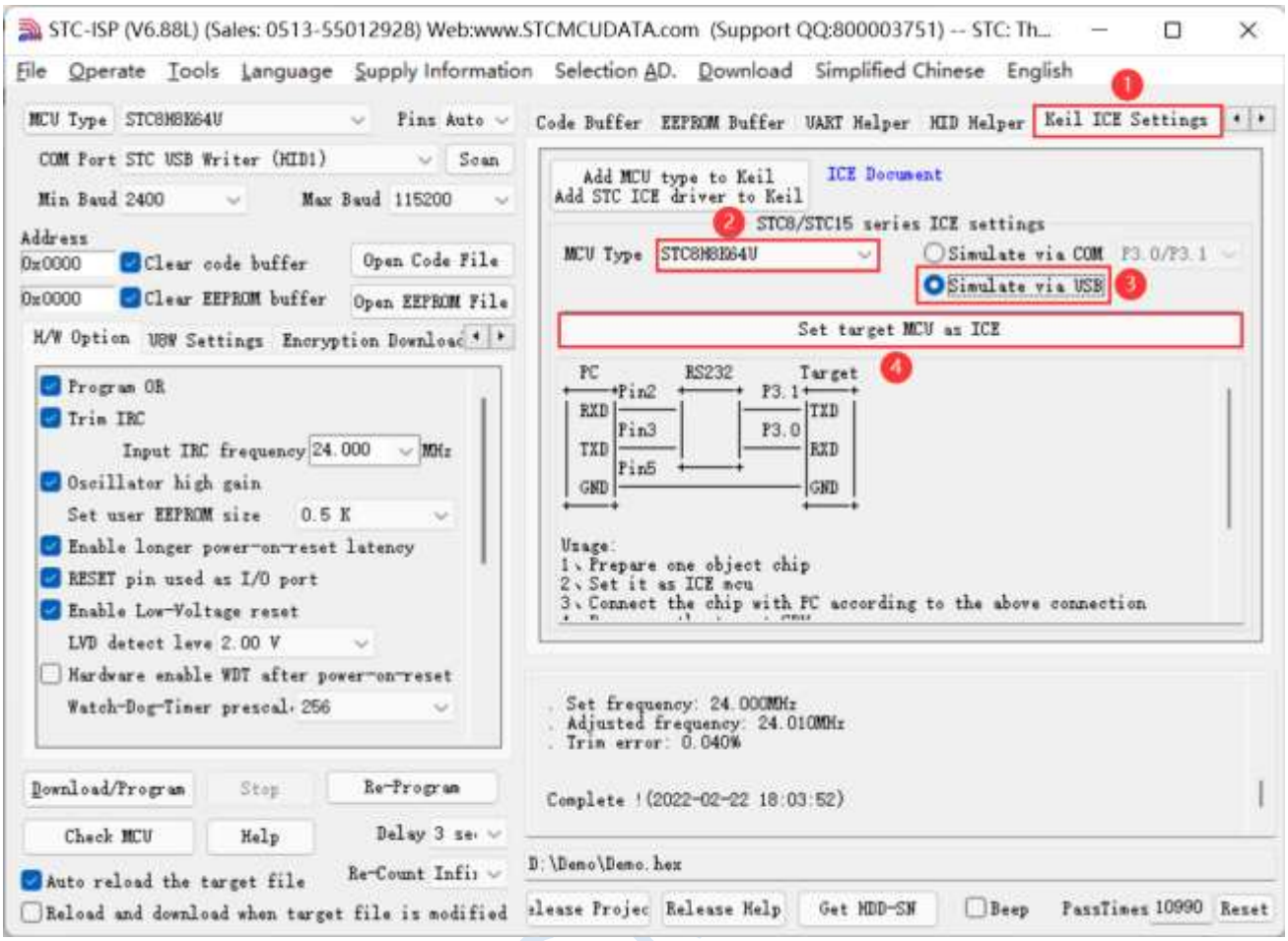
The setting steps are as follows:

Firstly, connect the target chip to the serial port of the computer as shown in the figure below, and short-circuit P3.2 to GND through the switch, and then power on the microcontroller.

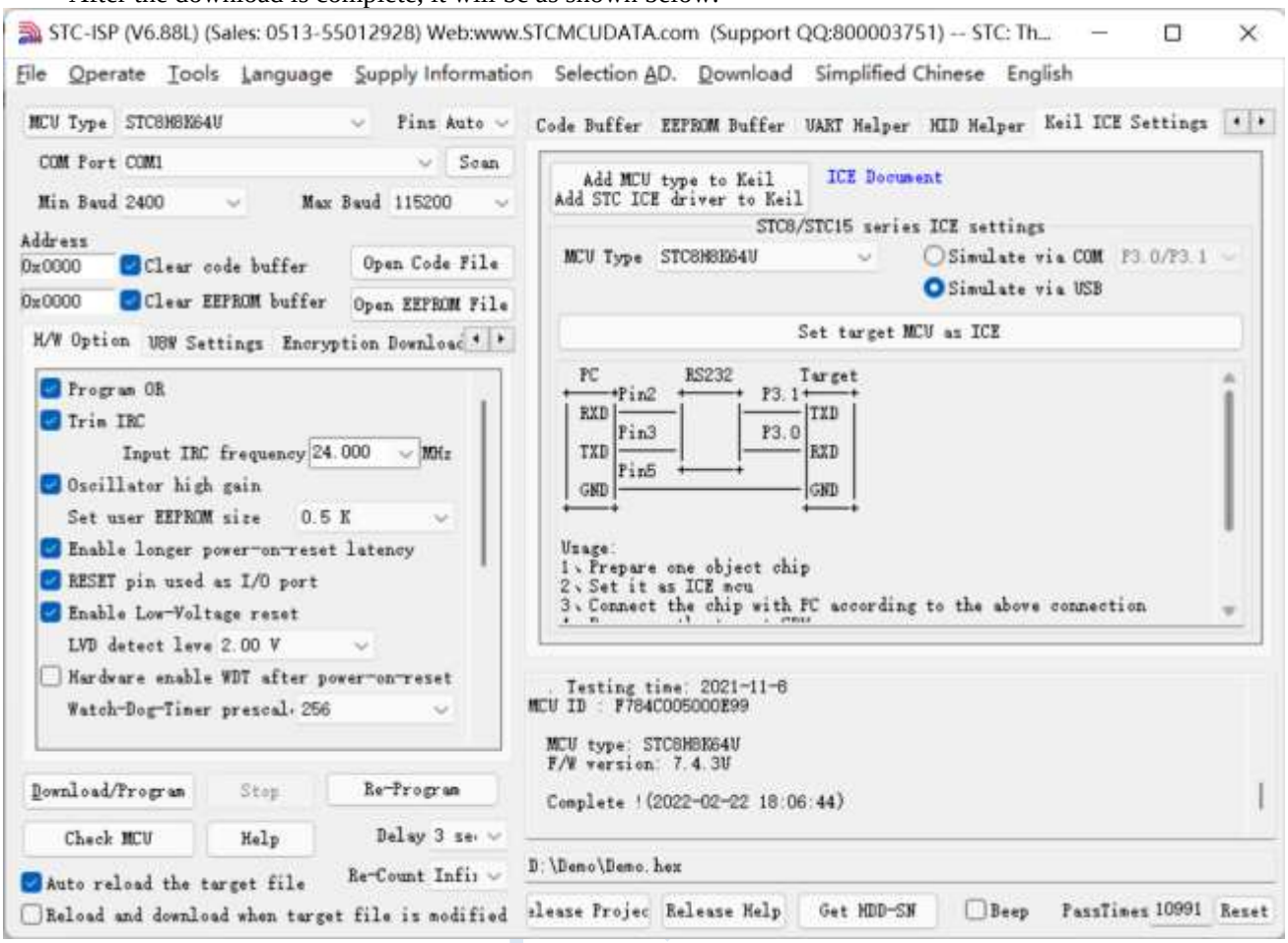



If "STC USB Writer (HID1)" can be automatically scanned in the ISP software, it means the connection is correct.

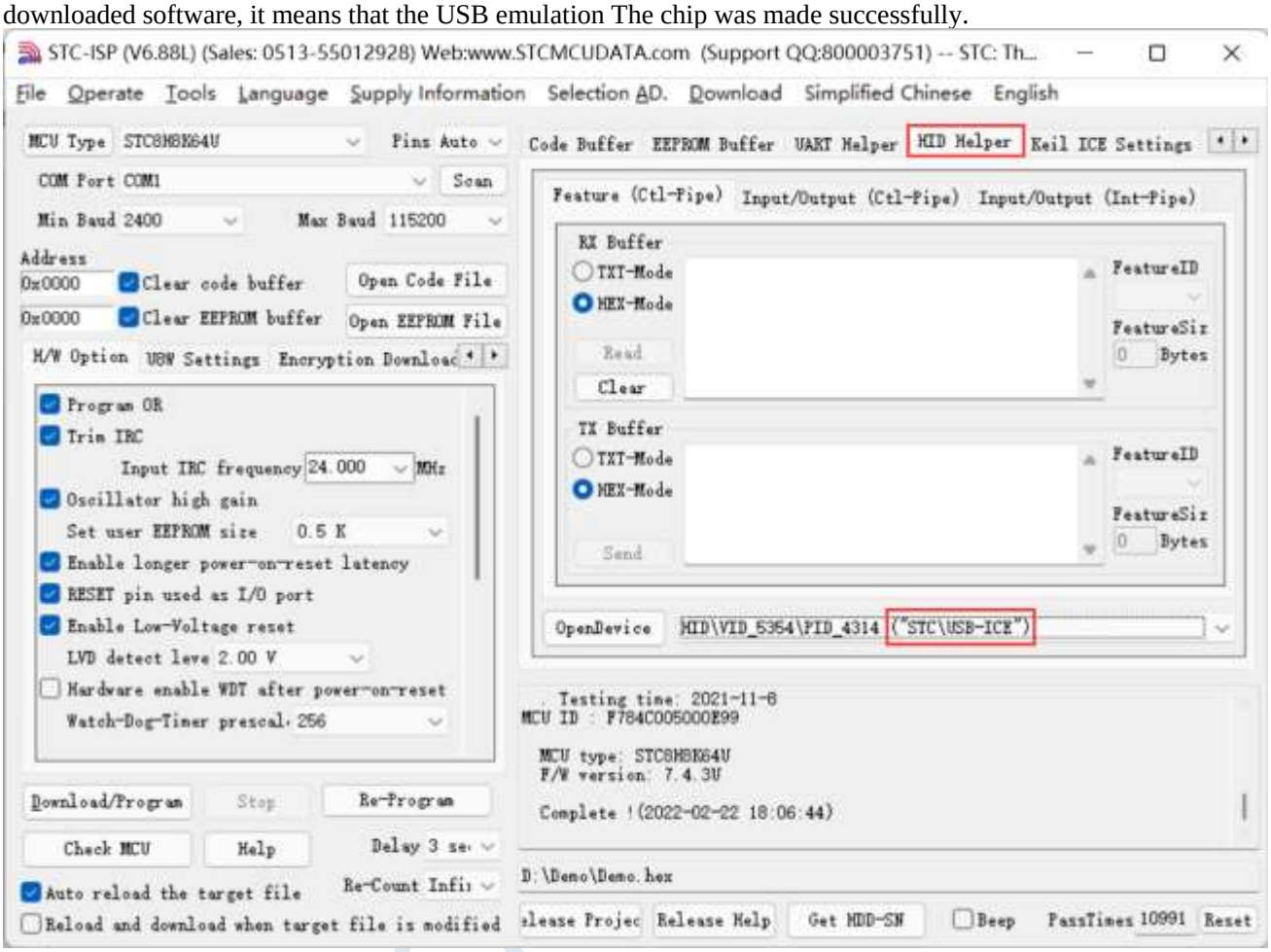

Next, in the STC-ISP download software, follow the steps shown in the figure below to set up the emulation chip.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
MCU Type STC8H8K64U
Pins Auto
COM Port STC USB Writer (HID1)
Scan
Min Baud 2400
Max Baud 115200
Address
0x0000
Clear code buffer
Open Code File
0x0000
Clear EEPROM buffer
Open EEPROM File
H/W Option
U8W Settings
Encryption Download
Program OR
Trim IRC
Input IRC frequency 24.000 MHz
Oscillator high gain
Set user EEPROM size 0.5 K
Enable longer power-on-reset latency
RESET pin used as I/O port
Enable Low-Voltage reset
LVD detect leve 2.00 V
Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256
Download/Program
Stop
Re-Program
Check MCU
Help
Delay 3 se
Auto reload the target file
Re-Count Infi
Reload and download when target file is modified
Code Buffer
EEPROM Buffer
UART Helper
MID Helper
Keil ICE Settings
Add MCU type to Keil
Add STC ICE driver to Keil
ICE Document
STC8/STC15 series ICE settings
MCU Type STC8H8K64U
Simulate via COM
P3.0/P3.1
Simulate via USB
Set target MCU as ICE
PC
RS232
Target
Pin2
P3.1
RXD
TXD
Pin3
P3.0
TXD
RXD
Pin5
GND
GND
Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection
. Set frequency: 24.000MHz
. Adjusted frequency: 24.010MHz
. Trim error: 0.040%
Complete !(2022-02-22 18:03:52)
D:\Demo\Demo.hex
Release Projec
Release Help
Get MDD-SN
Beep
PassTimes 10990
Reset

After the download is complete, it will be as shown below.

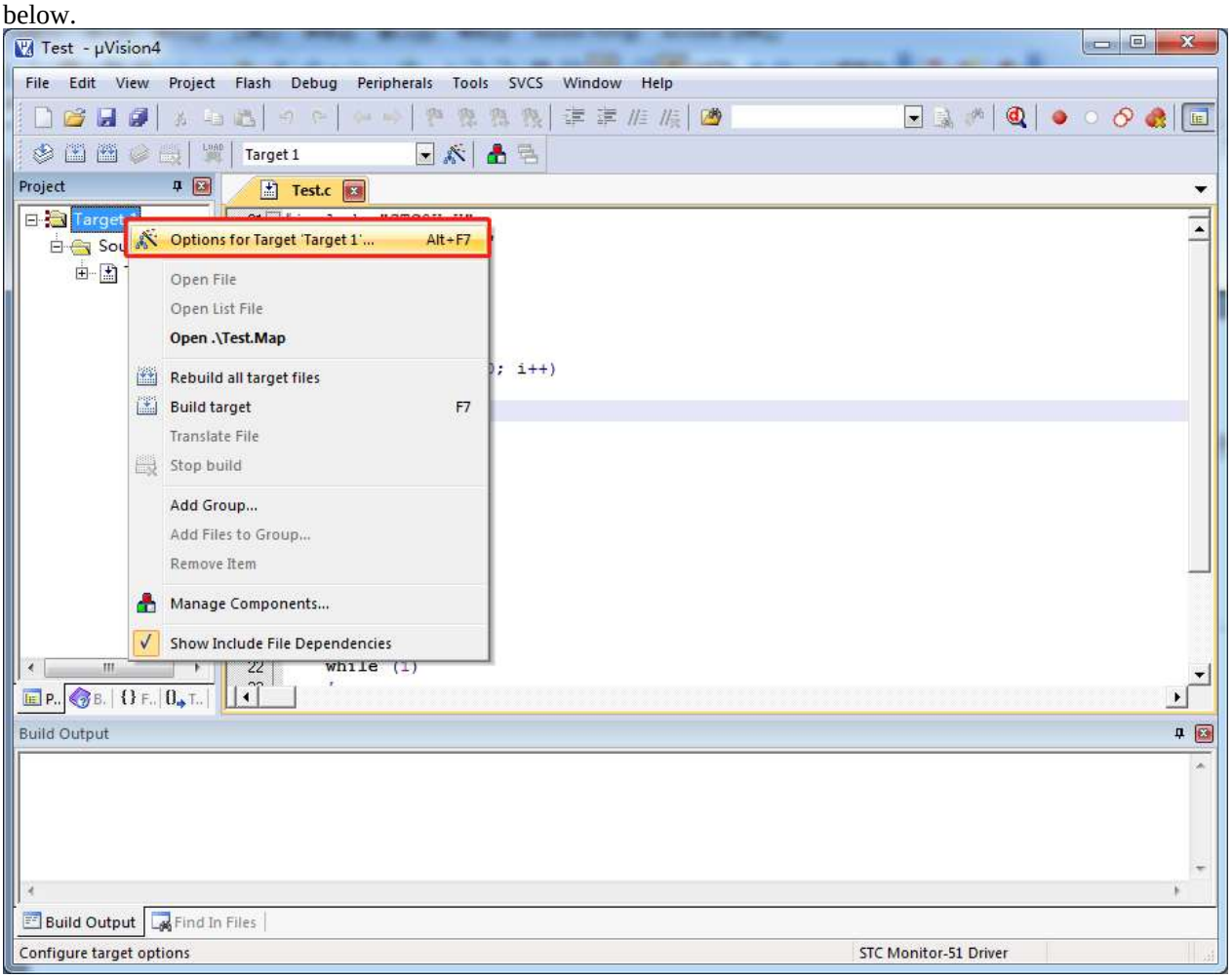


After the production is completed, the grounding switch of the P3.2 port needs to be disconnected, and the MCU needs to be powered on again. If the "STC\USB-ICE" device can be detected in the "HID Helper" in the downloaded software, it means that the USB emulation The chip was made successfully.

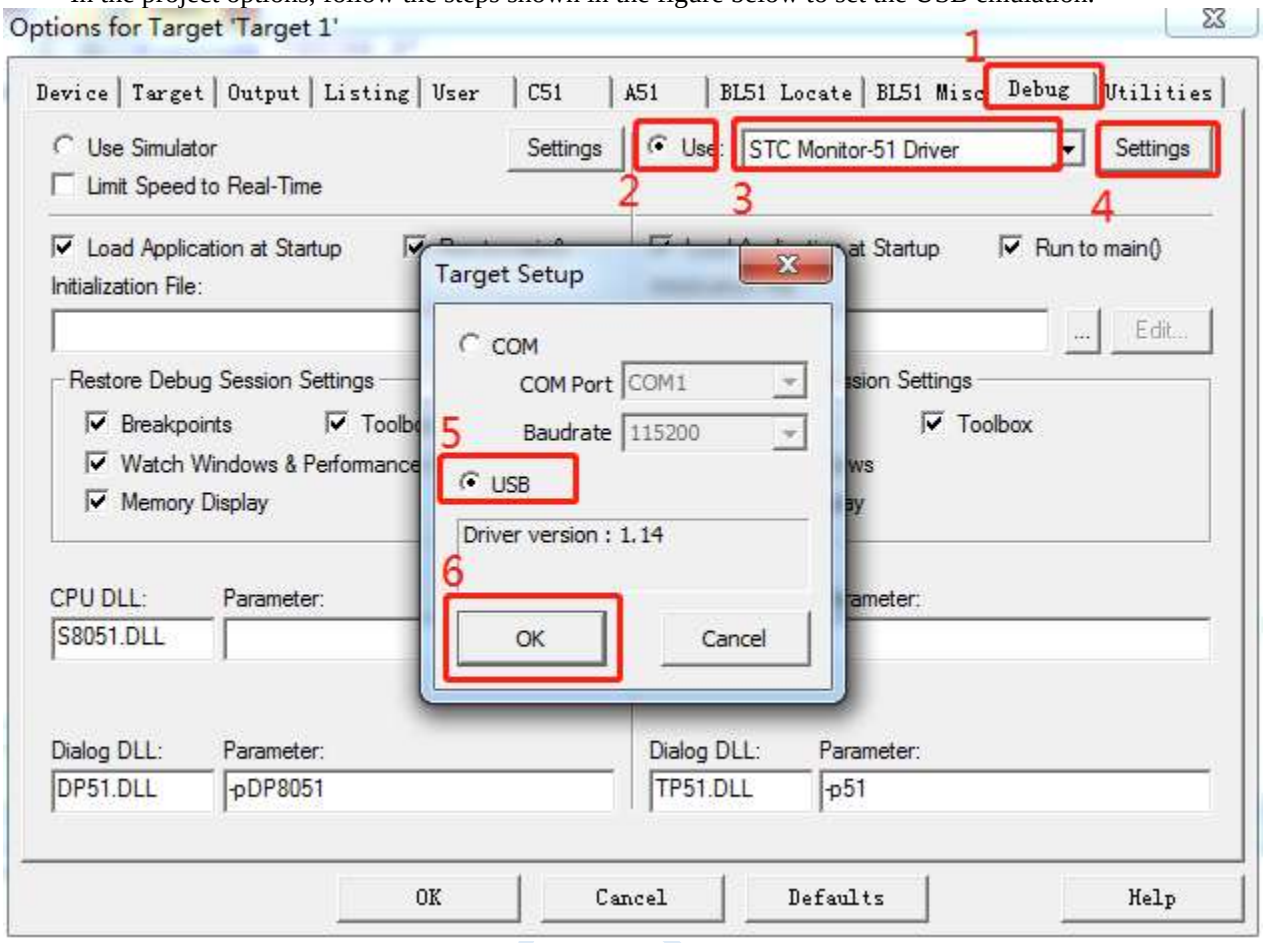


# I.5.2 USB emulation settings in Keil software

Open the project file in Keil software, and click "Options for ..." in the right-click menu as shown in the figure below.

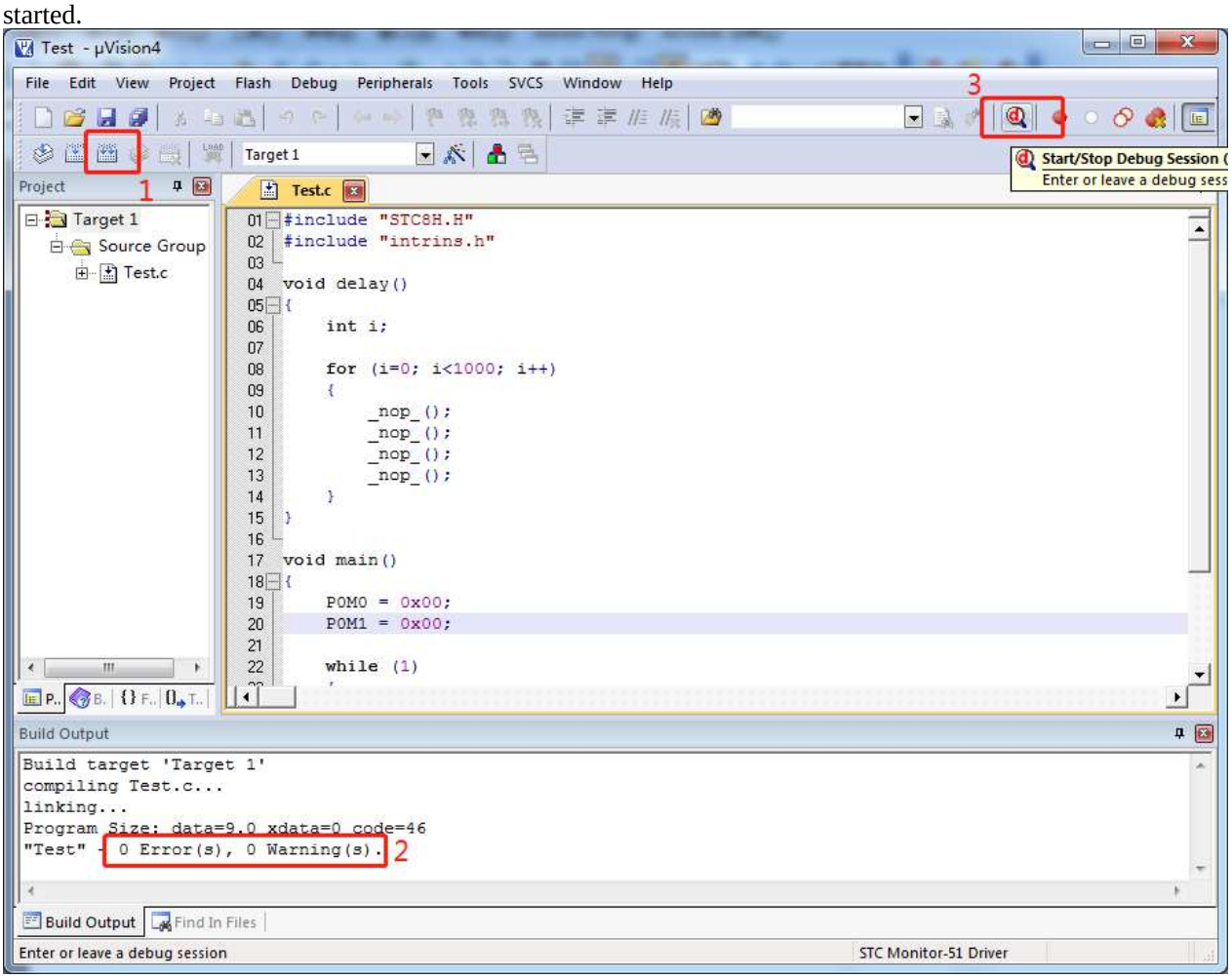


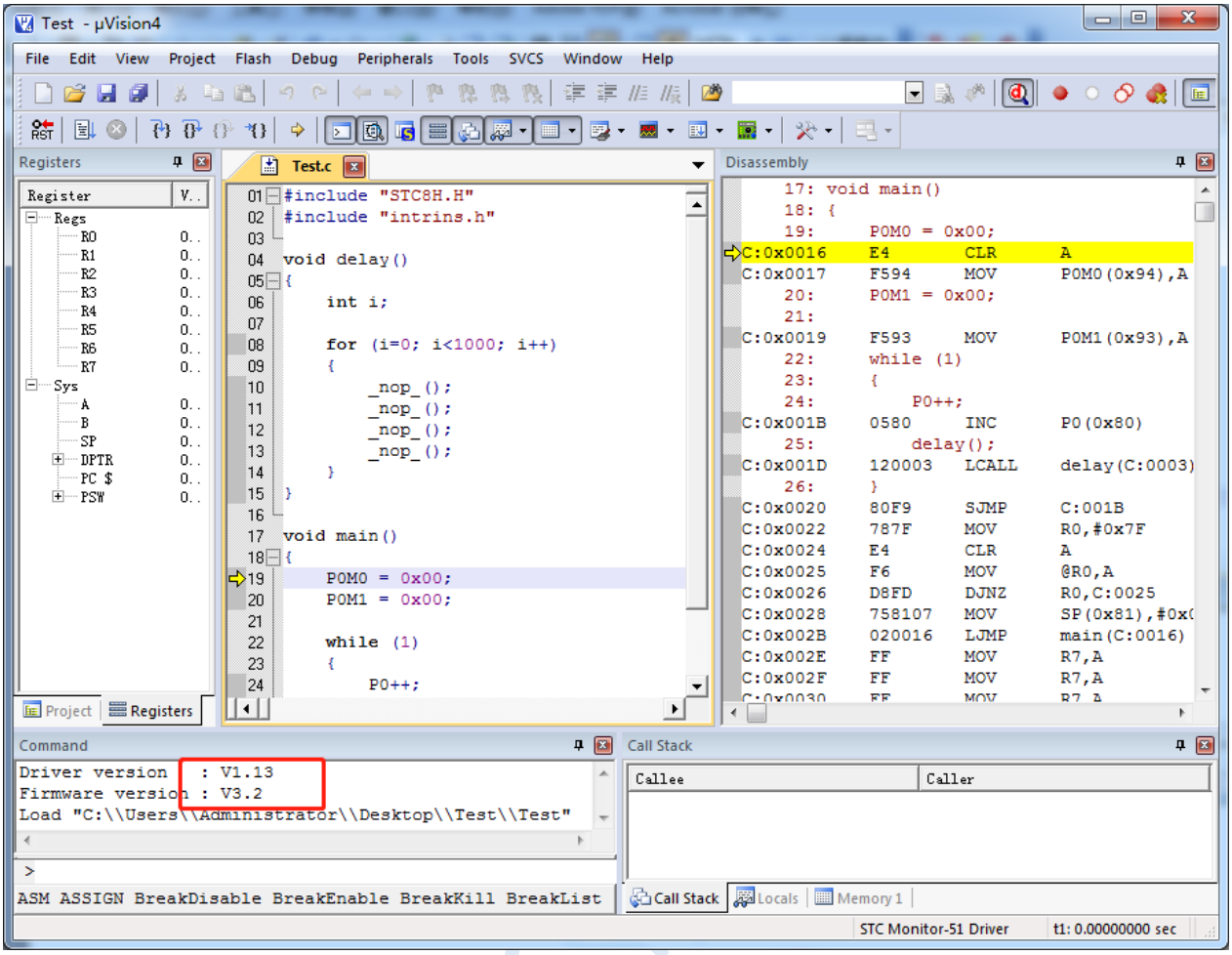
In the project options, follow the steps shown in the figure below to set the USB emulation.



# I.5.3 Emulation using USB in Keil software

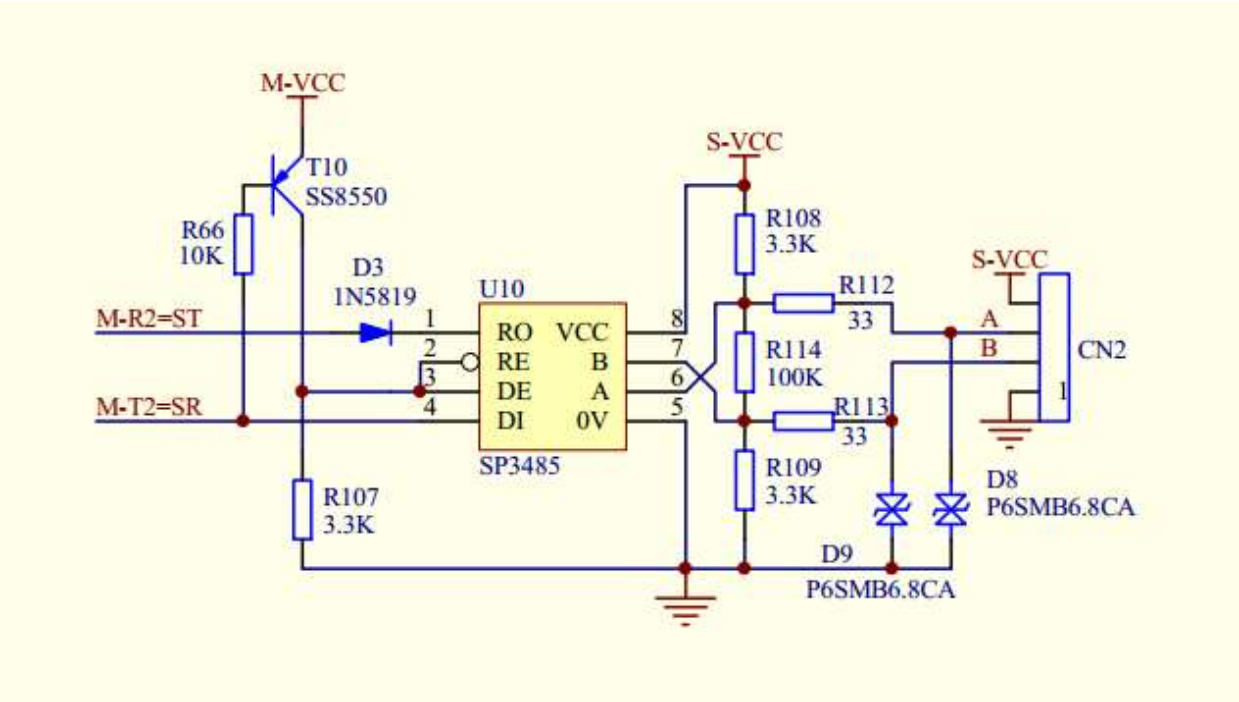
In the Keil environment, after editing the source code and compiling without errors, the emulation can be started.






If the chip making and connection are correct, the emulation driver version will be displayed as shown in the figure above, and the user code can be downloaded to the microcontroller correctly, and then debugging functions such as running, single step, and breakpoint can be performed.

# Appendix J Partial Circuit of RS485 in U8W Download Tool




BOM list:

| Label | Model | Package | Note |
|---|---|---|---|
| U10 | SP3485EN | SOP8 | RS485 chip |
| R66 | 10K | 0603 | Resistor |
| R107 | 3.3K | 0603 | Resistor |
| R108 | 3.3K | 0603 | Resistor |
| R109 | 3.3K | 0603 | Resistor |
| R112 | 33R | 0603 | Resistor |
| R113 | 33R | 0603 | Resistor |
| R114 | 100K | 0603 | Resistor |
| T10 | SS8550 | SOT-23 | PNP Triode |
| D3 | 1N5819 | 0603 | Schottky diode |
| D8 | P6SMB6.8CA | DO-214AA | TVS diode |
| D9 | P6SMB6.8CA | DO-214AA | TVS diode |
| CN2 | | SIP4 | Communication Interface |

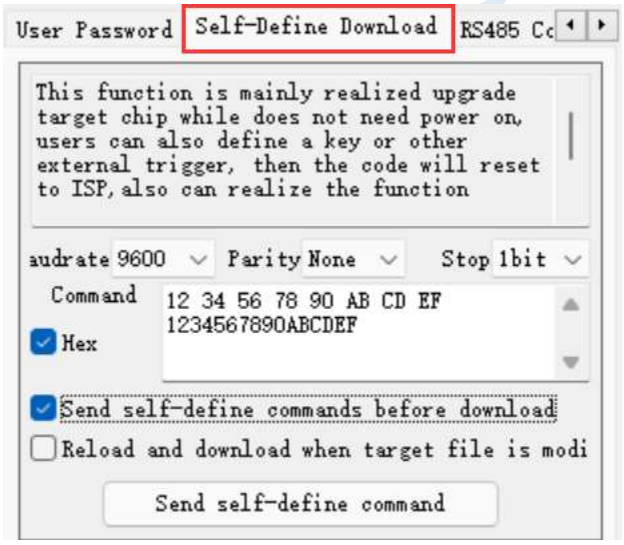
# Appendix K ISP Download Starts Automatically After Receiving User Command While Running User Program (no Power-down)

"User-defined download" and "user-defined encrypted download" are two completely different functions. Compared with the function of user-defined encrypted download, the function of user-defined download is simpler.

The specific functions is: Before the computer or offline download board starts to send the real ISP download programming handshake command, it first sends a user-defined string of commands (for this string of serial commands, user can set the baud rate, parity, and stop bits), and then immediately sends the ISP download programming handshake command.

The function of "user-defined download" is mainly used in the early development stage of the project, which can download user code without power-off (without re-power-on to the target chip). The specific implementation method is: User needs to add a piece of code to detect the custom command in user program. When the command is detected, execute the assembly code of "MOV IAP_CONTR, # 60H" or the C language code of "IAP_CONTR = 0x60;" , MCU will reset to ISP area to execute ISP code automatically.

As shown in the figure below, set the custom command sequence with a baud rate of 115200, no parity bit, and one stop bit: 0x12, 0x34, 0x56, 0xAB, 0xCD, 0xEF, 0x12. When the option "Send custom commands before each download" is checked, the user-defined download function can be implemented.




Click "Send user-defined download command" or click the "Download / Program" button in the lower left corner of the window, the application will send the serial data as shown below.


Function
Data ( Hex )
IRP_MJ_CREATE
Port Opened - stc-isp-15xx.exe
IOCTL_SERIAL_SET_BA...
Baud Rate: 115200
IOCTL_SERIAL_SET_LI...
StopBits: 1, Parity: No, DataBits: 8
IRP_MJ_WRITE
Length: 0001, Data: 12
IRP_MJ_WRITE
Length: 0001, Data: 34
IRP_MJ_WRITE
Length: 0001, Data: 56
IRP_MJ_WRITE
Length: 0001, Data: AB
IRP_MJ_WRITE
Length: 0001, Data: CD
IRP_MJ_WRITE
Length: 0001, Data: EF
IRP_MJ_WRITE
Length: 0001, Data: 12
IOCTL_SERIAL_SET_BA...
Baud Rate: 2400
IOCTL_SERIAL_SET_LI...
StopBits: 1, Parity: Even, DataBits: 8
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
IRP_MJ_WRITE
Length: 0001, Data: 7F
1. 用户的串口设置
2. 用户的自定义命令
3. ISP的串口设置及命令
单片机型号
引脚数
Auto
串口号
通讯端口 (COM1)
扫描
最低波特率
2400
最高波特率
115200
起始地址
0x0000
清除代码缓冲区
打开程序文件
自定义加载下载
RS485控制
自定义下载
波特率
115200
校验位
无校验位
停止位
1位
自定义命令
(HEX模式)
123456abcdef12
12 34 56 AB CD EF 12
每次下载前都先发送自定义命令
当目标文件变化时自动装载并发送自定义下载命令
发送自定义下载命令

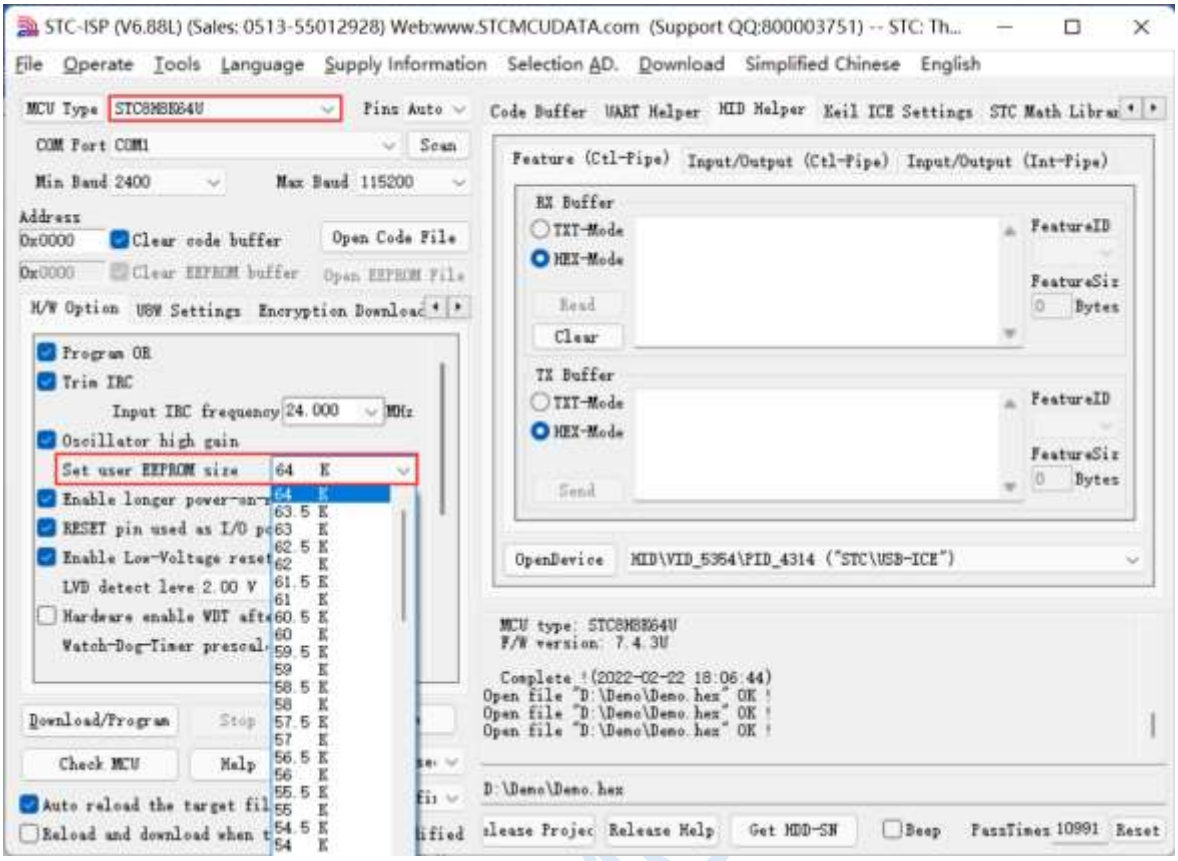
# Appendix L Use STC's IAP series MCU to develop your own ISP program

With the continuous development of IAP (In-Application-Programming) technology in the field of single-chip microcomputers, it has brought great convenience to the application system program code upgrade. STC's serial ISP (In-System-Programming) program uses the IAP function to upgrade the user's program online, but for the sake of user code safety, neither the underlying code nor the upper application is open source. For this reason, STC launched With the IAP series single-chip microcomputer, that is, the Flash space of the entire MCU, users can rewrite in their own programs, so that the idea that users need to develop their own ISP programs can be realized.

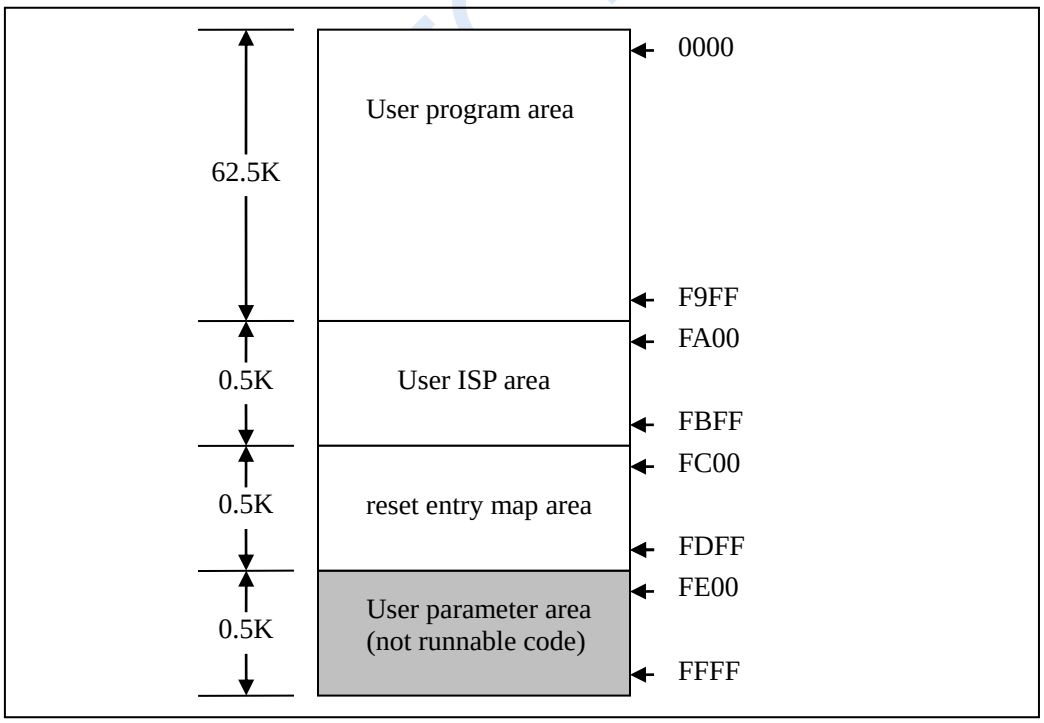
All the models in the STC8H series microcontrollers that can customize the EEPROM size during ISP download are IAP series.Film machine. At present, the STC8H series have the following types of microcontrollers as the IAP series: STC8H1K12, STC8H1K17, STC8H1K28, STC8H1K33, STC8H3K64S2, STC8H3K64S4, STC8H8K64U, STC8H2K64T. This article uses STC8H8K64U,As an example, the method of using STC's IAP MCU to develop user's own ISP program is explained in detail, and the assembly and C source code based on Keil environment are given.

## The first step: internal FLASH planning

Since the EEPROM of the IAP microcontroller of the STC8H series is set by the user during ISP download, if the user needs to implement his own ISP, when downloading the user's own ISP program, the user needs to follow the figure below to set all 64K Set it to EEPROM, so that the user program space and EEPROM space are completely overlapped, so that users can modify and update their own program space.

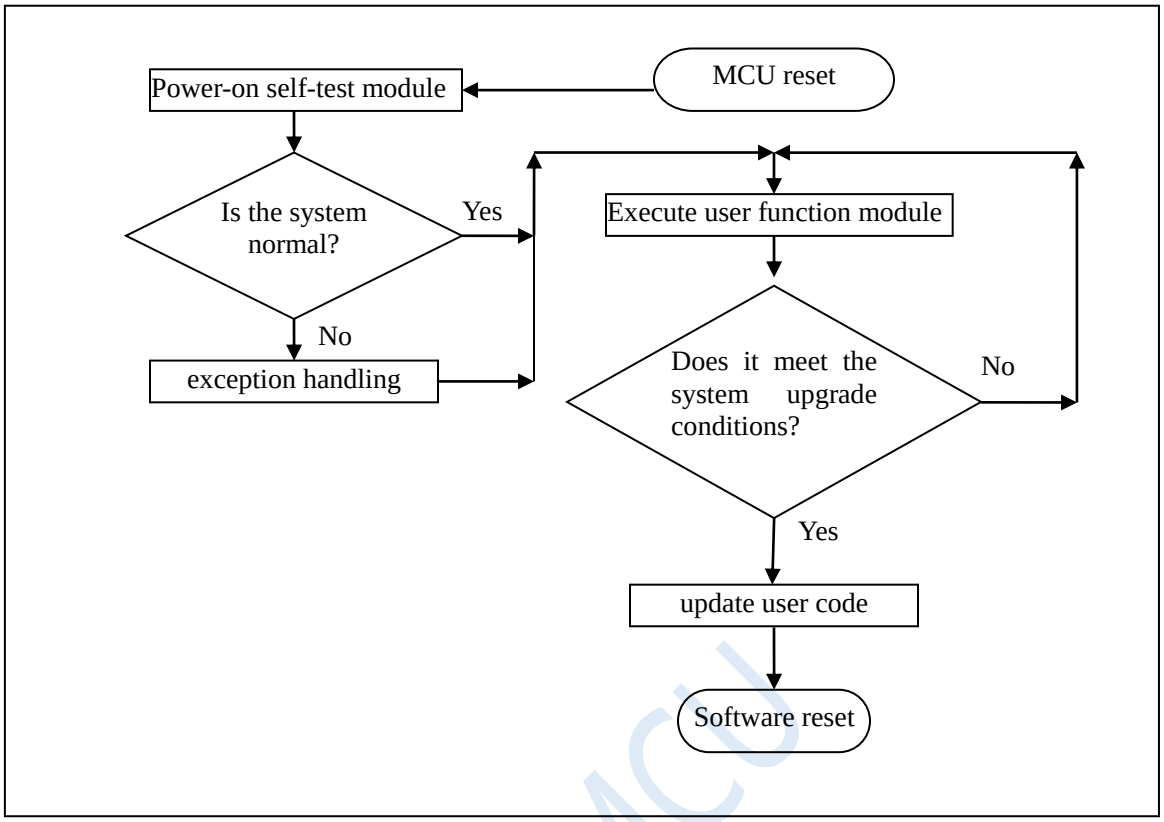



The following assumes that the user has set the entire 64K program space as EEPROM. Now the entire 64K program space is divided as follows:




In the FLASH space, the continuous 62.5K bytes of space starting from address 0000H is the user program area. When the specific download conditions are met, the user is required to jump the PC to the user ISP program area. At this time, the user program area can be erased and rewritten to achieve the purpose of updating the user program

## The second step, the basic framework of the program




## The third step, the firmware program description of the lower computer

The firmware program of the lower computer includes two parts: ISP (ISP code) and AP (user code)

### ISP Code (assembly code)

```
; Operating frequency for test is 11.0592MHz


UARTBAUD    EQU     0FFE8H      ;Define the serial port baud rate   (65536-11059200/4/115200)

AUXR        DATA    08EH        ;Additional Function Control Register
WDT_CONTR   DATA    0C1H        ;Watchdog Control Register
IAP_DATA    DATA    0C2H        ;IAP data register
IAP_ADDRH   DATA    0C3H        ;IAP High Address Register
IAP_ADDRL   DATA    0C4H        ;IAP Low Address Register
IAP_CMD     DATA    0C5H        ;IAP Command Register
IAP_TRIG    DATA    0C6H        ;IAP Command Trigger Register
IAP_CONTR   DATA    0C7H        ;IAP Control Register
IAP_TPS     DATA    0F5H        ;IAP latency control register

ISPCODE     EQU     0FA00H      ;ISP module entry address (1 page), also external interface address
APENTRY     EQU     0FC00H      ; Application entry address data (1 page)

            ORG     0000H

            LJMP    ISP_ENTRY   ;System reset entry

RESET:
```

```
                MOV     SCON,#50H               ;Set serial port mode (8 data bits, no parity bit)
                MOV     AUXR,#40H               ;Timer 1 is in 1T mode
                MOV     TMOD,#00H               ;Timer 1 works in mode 0 (16-bit reload)
                MOV     TH1,#HIGH UARTBAUD      ;set reload value
                MOV     TL1,#LOW UARTBAUD
                SETB    TR1                     ;start timer 1
NEXT1:
                MOV     R0,#16
NEXT2:
                JNB     RI,$                    ;Waiting for serial data
                CLR     RI
                MOV     A,SBUF
                CJNE    A,#7FH,NEXT1            ;Determine whether it is 7F
                DJNZ    R0,NEXT2
                LJMP    ISP_DOWNLOAD            ;Jump to download interface

                ORG     ISPCODE

ISP_DOWNLOAD:
                CLR     A
                MOV     PSW,A                   ;ISP module uses group 0 registers
                MOV     IE,A                    ;Disable all interrupts
                CLR     RI                      ;Clear serial port receive flag
                SETB    TI                      ; Set serial port send flag
                CLR     TR0
                MOV     SP,#5FH                 ;set stack pointer

                MOV     A,#5AH                  ;Return 5A 55 to PC, indicating ISP erase module is ready
                LCALL   ISP_SENDUART
                MOV     A,#055H
                LCALL   ISP_SENDUART
                LCALL   ISP_RECVACK             ;Receive response data

                MOV     IAP_ADDRL,#0            ;First write the “LJMP ISP_ENTRY” instruction at the starting
address of page 2
                MOV     IAP_ADDRH,#02H
                LCALL   ISP_ERASEIAP
                MOV     A,#02H
                LCALL   ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV     A,#HIGH                 ISP_ENTRY
                LCALL   ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV     A,#LOW ISP_ENTRY
                LCALL   ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code

                MOV     IAP_ADDRL,#0            ;User code address starts from 0
                MOV     IAP_ADDRH,#0
                LCALL   ISP_ERASEIAP
                MOV     A,#02H
                LCALL   ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV     A,#HIGH                 ISP_ENTRY
                LCALL   ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV     A,#LOW ISP_ENTRY
                LCALL   ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code

                MOV     IAP_ADDRL,#0            ;new code buffer address
                MOV     IAP_ADDRH,#02H
                MOV     R7,#124                 ;Erase 62.5K bytes
ISP_ERASEAP:
```

```
                LCALL   ISP_ERASEIAP
                INC     IAP_ADDRH               ;target address+512
                INC     IAP_ADDRH
                DJNZ    R7,ISP_ERASEAP          ;Determine if erasing is complete

                MOV     IAP_ADDRL,#LOW APENTRY
                MOV     IAP_ADDRH,#HIGH APENTRY
                LCALL   ISP_ERASEIAP

                MOV     A,#5AH                  ;Return 5A A5 to PC, indicating that the ISP programming
module is ready
                LCALL   ISP_SENDUART
                MOV     A,#0A5H
                LCALL   ISP_SENDUART
                LCALL   ISP_RECVACK             ;Receive response data

                LCALL   ISP_RECVUART            ;Receive length high byte
                MOV     R0,A
                LCALL   ISP_RECVUART            ;Receive length low byte
                MOV     R1,A
                CLR     C                       ; total length -3
                MOV     A,#03H
                SUBB    A,R1
                MOV     DPL,A
                CLR     A
                SUBB    A,R0
                MOV     DPH,A                   ;Total length complement stored in DPTR

                LCALL   ISP_RECVUART            ;Map user code reset entry code to map area
                LCALL   ISP_PROGRAMIAP          ;0000
                LCALL   ISP_RECVUART
                LCALL   ISP_PROGRAMIAP          ;0001
                LCALL   ISP_RECVUART
                LCALL   ISP_PROGRAMIAP          ;0002

                MOV     IAP_ADDRL,#03H          ;User code start address
                MOV     IAP_ADDRH,#00H
ISP_PROGRAMNEXT:
                LCALL   ISP_RECVUART            ;receive code data
                LCALL   ISP_PROGRAMIAP          ;Program the data into the user code area
                INC     DPTR
                MOV     A,DPL
                ORL     A,DPH
                JNZ     ISP_PROGRAMNEXT         ;length detection

ISP_SOFTRESET:
                MOV     IAP_CONTR,#20H          ;Software reset system
                SJMP    $


ISP_ENTRY:
                MOV     WDT_CONTR,#17H          ;clear watchdog
                MOV     IAP_CONTR,#80H          ;Enable IAP function
                MOV     IAP_TPS,#11             ;Set the IAP wait time parameter
                MOV     IAP_ADDRL,#LOW ISP_DOWNLOAD
                MOV     IAP_ADDRH,#HIGH ISP_DOWNLOAD
                MOV     IAP_DATA,#00H           ;Test data 1
                MOV     IAP_CMD,#1              ;Read command
```

```
                MOV     IAP_TRIG,#5AH           ;Triger ISP command
                MOV     IAP_TRIG,#0A5H
                MOV     A,IAP_DATA
                CJNE    A,#0E4H,ISP_ENTRY       ; If the data cannot be read, wait for the voltage to stabilize
                INC     IAP_ADDRL               ;Test address FC01H
                MOV     IAP_DATA,#45H           ;Test data 2
                MOV     IAP_CMD,#1              ;Read command
                MOV     IAP_TRIG,#5AH           ;Triger ISP command
                MOV     IAP_TRIG,#0A5H
                MOV     A,IAP_DATA
                CJNE    A,#0F5H,ISP_ENTRY       ; If the data cannot be read, wait for the voltage to stabilize

                MOV     SCON,#50H               ;Set serial port mode (8 data bits, no parity bit)
                MOV     AUXR,#40H               ;Timer 1 is in 1T mode
                MOV     TMOD,#00H               ;Timer 1 works in mode 0 (16-bit reload)
                MOV     TH1,#HIGH UARTBAUD      ;set reload value
                MOV     TL1,#LOW UARTBAUD
                SETB    TR1                     ;start timer 1
                SETB    TR0

                LCALL   ISP_RECVUART            ; Check if there is serial data
                JC      GOTOAP
                MOV     R0,#16
ISP_CHECKNEXT:
                LCALL   ISP_RECVUART            ; receive sync data
                JC      GOTOAP
                CJNE    A,#7FH,GOTOAP           ;Determine whether it is 7F
                DJNZ    R0,ISP_CHECKNEXT
                MOV     A,#5AH                  ; Return 5A 69 to PC, indicating ISP module is ready
                LCALL   ISP_SENDUART
                MOV     A,#69H
                LCALL   ISP_SENDUART
                LCALL   ISP_RECVACK             ;Receive response data
                LJMP    ISP_DOWNLOAD            ;Jump to download interface
GOTOAP:
                CLR     A                       ; Reset SFR to reset value
                MOV     TCON,A
                MOV     TMOD,A
                MOV     TL0,A
                MOV     TH0,A
                MOV     TL1,A
                MOV     TH1,A
                MOV     SCON,A
                MOV     AUXR,A
                LJMP    APENTRY                 ; Run the user program normally

ISP_RECVACK:
                LCALL   ISP_RECVUART
                JC      GOTOAP
                XRL     A,#7FH
                JZ      ISP_RECVACK             ; skip sync data
                CJNE    A,#25H,GOTOAP           ; Response data 1 detection
                LCALL   ISP_RECVUART
                JC      GOTOAP
                CJNE    A,#69H,GOTOAP           ; Response data 2 detection
                RET

ISP_RECVUART:
```

```
        CLR     A
        MOV     TL0,A           ; Initialize timeout timer
        MOV     TH0,A
        CLR     TF0
        MOV     WDT_CONTR,#17H  ;clear watchdog
ISP_RECVWAIT:
        JBC     TF0,ISP_RECVTIMEOUT ; Timeout detection
        JNB     RI,ISP_RECVWAIT ; wait for receive to complete
        MOV     A,SBUF          ; Read serial data
        CLR     RI              ;Clear flag
        CLR     C               ; Receive serial data correctly
        RET
ISP_RECVTIMEOUT:
        SETB    C               ; timeout
        RET

ISP_SENDUART:
        MOV     WDT_CONTR,#17H  ;clear watchdog
        JNB     TI,ISP_SENDUART ; Wait for the previous data transmission to complete
        CLR     TI              ; Clear flag
        MOV     SBUF,A          ; send current data
        RET

ISP_ERASEIAP:
        MOV     WDT_CONTR,#17H  ;clear watchdog
        MOV     IAP_CMD,#3      ; Erase command
        MOV     IAP_TRIG,#5AH   ; Trigger ISP command
        MOV     IAP_TRIG,#0A5H
        NOP
        NOP
        NOP
        NOP
        RET

ISP_PROGRAMIAP:
        MOV     WDT_CONTR,#17H  ;clear watchdog
        MOV     IAP_CMD,#2      ; programming command
        MOV     IAP_DATA,A      ; send the current data to IAP data register
        MOV     IAP_TRIG,#5AH   ; Trigger ISP command
        MOV     IAP_TRIG,#0A5H
        NOP
        NOP
        NOP
        NOP
        MOV     A,IAP_ADDRL     ;IAP address +1
        ADD     A,#01H
        MOV     IAP_ADDRL,A
        MOV     A,IAP_ADDRH
        ADDC    A,#00H
        MOV     IAP_ADDRH,A
        RET


        ORG     APENTRY
        LJMP    RESET
```

```
END
```

The ISP code includes the following external interface modules:
ISP_DOWNLOAD: program download entry address, absolute address FA00H
ISP_ENTRY: Power-on system self-check program (automatically called by the system)
For the user program, the user only needs to jump the PC value to ISPPROGRAM (ie FA00H, Absolute address), you can update the code.

## User codes (C language code)

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

#define   FOSC              11059200L           // system clock frequency
#define   BAUD              (65536 - FOSC/4/115200)  //Define the serial port baud rate
#define   ISPPROGRAM        0xfa00              //ISP download program entry address

sfr       AUXR        =     0x8e;               // Baud Rate Generator Control Register
sfr       P1M0        =     0x92;
sfr       P1M1        =     0x91;

void (*IspProgram)() = ISPPROGRAM;              // define pointer function
char cnt7f;                                     //Isp_Check Internally used variables

void uart() interrupt 4                         // UART Interrupt Service Routine
{
    if (TI) TI = 0;                             // send complete interrupt
    if (RI)                                     // Receive complete interrupt
    {
        if (SBUF == 0x7f)
        {
            cnt7f++;
            if (cnt7f >= 16)
            {
                IspProgram();                   // Invoke the download module (****important statement ****)
            }
        }
        else
        {
            cnt7f = 0;
        }
        RI = 0;                                 // Clear the reception completion flag
    }
}

void main()
{
    SCON = 0x50;                                // Define the serial port mode as 8-bit, variable baud rate, no parity bit
    AUXR = 0x40;
    TH1  = BAUD >> 8;
    TL1  = BAUD;
    TR1  = 1;
    ES = 1;                                     //Enable UART interrupt
    EA = 1;                                     //Enable CPU interrupt

    P1M0 = 0;
```

```
    P1M1 = 0;

    while (1)
    {
        P1++;
    }
}
```

## User codes (assembly code)

```
; Operating frequency for test is 11.0592MHz

UARTBAUD    EQU     0FFE8H      ;Define the serial port baud rate   (65536-11059200/4/115200)
ISPPROGRAM  EQU     0FA00H      ; ISP download program entry address

AUXR        DATA    08EH        ; Additional Function Control Register

CNT7F       DATA    60H         ; Receive 7F counter

            ORG     0000H
            LJMP    START       ;System reset entry

            ORG     0023H
            LJMP    UART_ISR    ; UART interrupt entry

UART_ISR:
            PUSH    ACC
            PUSH    PSW
            JNB     TI,CHECKRI  ; Detect transmission interruption
            CLR     TI          ; clear flag
CHECKRI:
            JNB     RI,UARTISR_EXIT ; Detect receive interruption
            CLR     RI          ; clear flag
            MOV     A,SBUF
            CJNE    A,#7FH,ISNOT7F
            INC     CNT7F
            MOV     A,CNT7F
            CJNE    A,#16,UARTISR_EXIT
            LJMP    ISPPROGRAM  ; Invoke the download module (****important statement ****)
ISNOT7F:
            MOV     CNT7F,#0
UARTISR_EXIT:
            POP     PSW
            POP     ACC
            RETI

START:
            MOV     R0,#7FH     ;Clear RAM
            CLR     A
            MOV     @R0,A
            DJNZ    R0,$-1
            MOV     SP,#7FH     ; Initialize SP

            MOV     SCON,#50H   ; Set UART mode (8-bit data, variable baud rate, no parity bit)
            MOV     AUXR,#15H   ; BRT works in 1T mode, start BRT
            MOV     TMOD,#00H   ;Timer 1 works in mode 0 (16-bit reload)
            MOV     TH1,#HIGH UARTBAUD  ;set reload value
            MOV     TL1,#LOW UARTBAUD
```

```
        SETB    TR1             ;start timer 1
        SETB    ES              ; Enable UART interrupt
        SETB    EA              ; Enable CPU interrupt

MAIN:
        INC     P0
        SJMP    MAIN

        END
```

User code can be written in C or assembly language, but one thing to note about assembly code: the instruction at the reset entry address of 0000H must be a long jump statement (similar to LJMP START). In the user code, the serial port needs to be set up, and when the download conditions are met, the PC value is jumped to ISPPROGRAM (that is, the absolute address of FA00H) to achieve code update. For assembly code, we can use the "LJMP 0FA00H" instruction to call, as shown below

```
UARTBAUD    EQU     0FFE8H          ;定义串口波特率 (65536-11059200/4/115200)
ISPPROGRAM  EQU     0FA00H          ;ISP下载程序入口地址

AUXR        DATA    08EH            ;附件功能控制寄存器
```

```
18          CLR     TI              ;清除标志
19  CHECKRI:
20          JNB     RI,UARTISR_EXIT ;检测接收中断
21          CLR     RI              ;清除标志
22          MOV     A,SBUF
23          CJNE    A,#7FH,ISNOT7F
24          INC     CNT7F
25          MOV     A,CNT7F
26          CJNE    A,#16,UARTISR_EXIT
27          LJMP    ISPPROGRAM      ;调用下载模块(****重要语句****)
28  ISNOT7F:
29          MOV     CNT7F,#0
30  UARTISR_EXIT:
31          POP     PSW
32          POP     ACC
33          RETI
34
35  START:
```

In the C code, you must define a function pointer variable, and assign this variable to 0xFA00, and then call, as shown below:

```
#include "reg51.h"

#define FOSC            11059200L                   //系统时钟频率
#define BAUD            (65536 - FOSC/4/115200)     //定义串口波特率
#define ISPPROGRAM      0xfa00                      //ISP下载程序入口地址

sfr AUXR    =   0x8e;                               //波特率发生器控制寄存器
sfr P1M0    =   0x92;
sfr P1M1    =   0x91;

void (*IspProgram)() = ISPPROGRAM;                  //定义指针函数
char cnt7f;                                         //Isp_Check内部使用的变量

void uart() interrupt 4                             //串口中断服务程序
{
    if (TI) TI = 0;                                 //发送完成中断
    if (RI)                                         //接收完成中断
    {
        if (SBUF == 0x7f)
        {
            cnt7f++;
            if (cnt7f >= 16)
            {
                IspProgram();                       //调用下载模块(****重要语句****)
            }
        }
        else
        {
            cnt7f = 0;
        }
        RI = 0;                             //清接收完成标志
    }
```

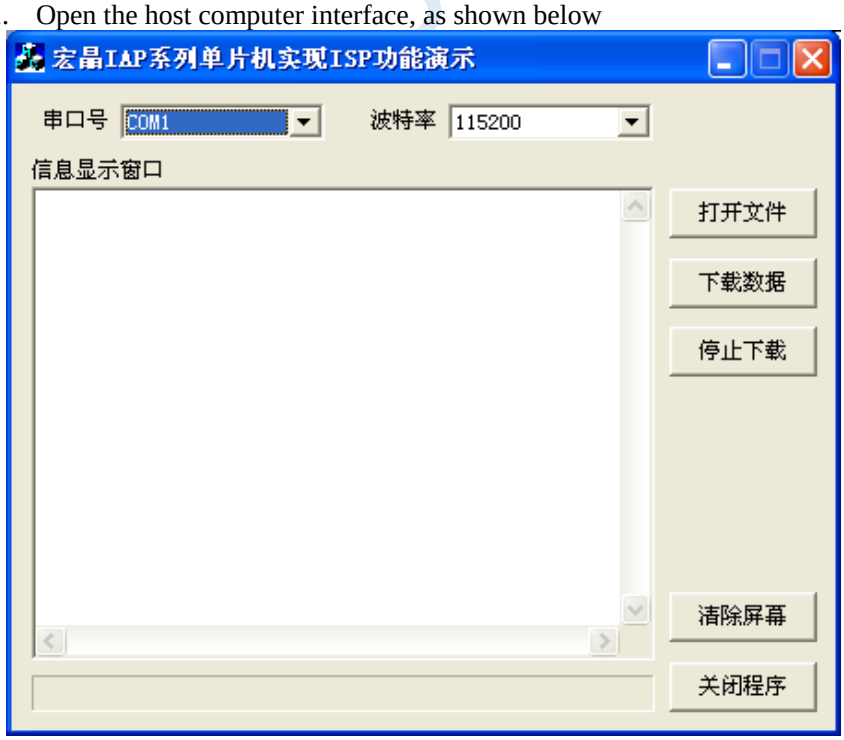
## Step 4, the host computer application program description

The program of the upper computer is a dialog box project based on MFC. The access to the serial port is to directly call the API function of Windows without using the serial port control, which saves the registration of the control and many problems of system version incompatibility. The interface is relatively simple, but it provides a framework for the realization of this function. Other functions and requirements can be added in the past.

The core module of the host computer program is a friend function "UINT Download(LPVOID pParam);" based on CISPDlg,This function is responsible for communicating with the lower computer and sending various communication commands to complete the update of the user program. Users can add commands according to their different needs.

## Step 5: How to use the host computer application

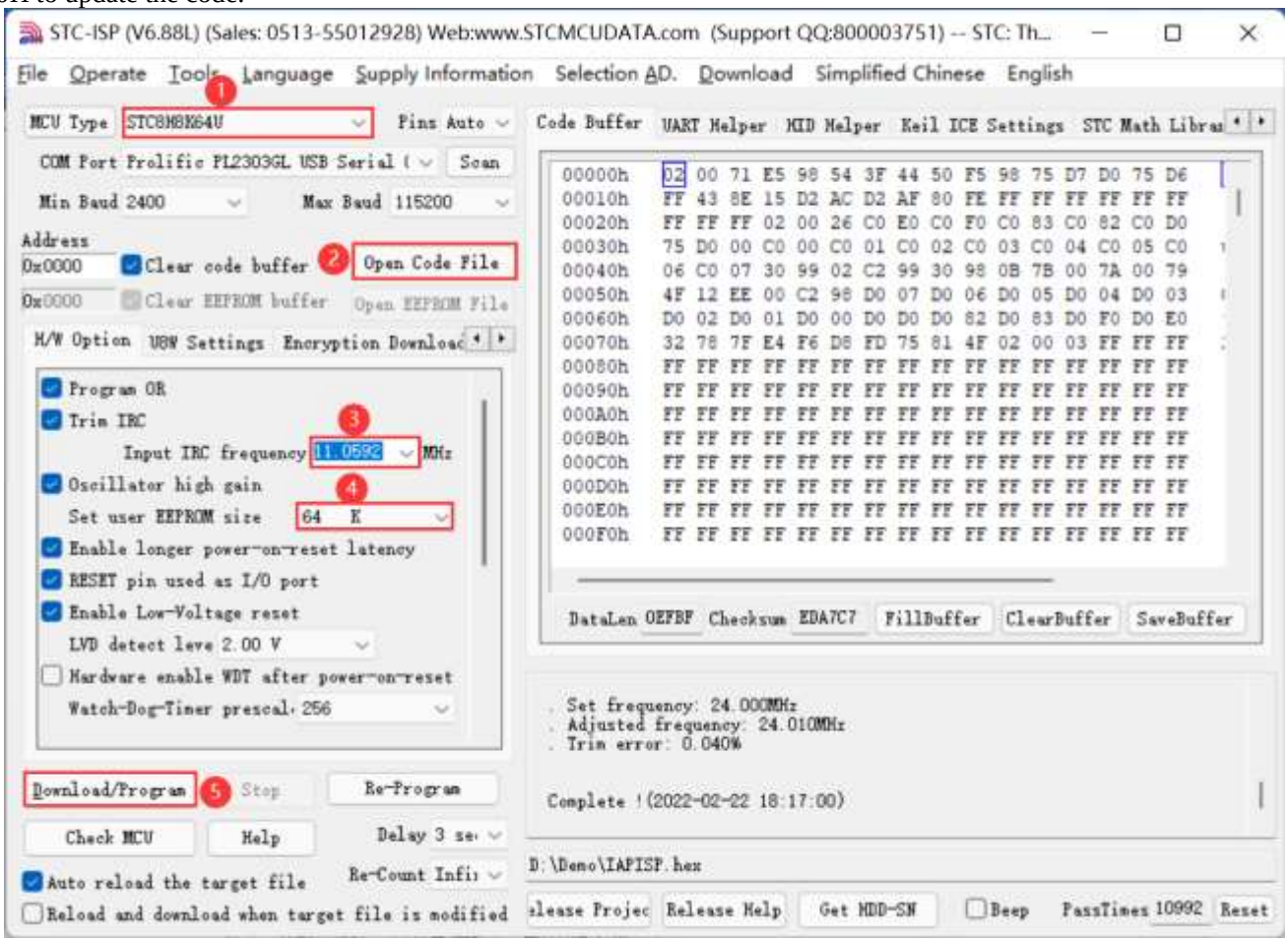
1. Open the host computer interface, as shown below



2. Select the serial port number, set the same serial port baud rate as the lower computer
3. Open the source data file to be downloaded, either in Bin or Intel hex format
4. Click the "download data" button to start downloading data

## The sixth step, how to use the firmware of the lower computer

The target file of the lower computer has two "IAPISP.hex" and "AP.hex". For a new single-chip computer, for the first time, you must use the ISP download tool of Acer Technology to write "IAPISP.hex" into the chip. As shown below. No need to write after updating "IAPISP.hex" is the file. The "AP.hex" in the attachment is just a template for the user program. When the download conditions are met, the user only needs to jump the PC value to the address of FA00H to update the code.



# Appendix J The method of resetting the user program to the system area for ISP download (without power off)

When the project is in the development stage, it is necessary to repeatedly download the user code to the target chip for code verification, and the STC microcontroller

For normal ISP downloads, the target chip needs to be re-powered, which will make the project development phase more cumbersome. For this reason, STC MCU has added a special function register IAP_CONTR. When the user writes 0x60 to this register, the software can be reset to the system area, and then ISP download can be performed without power failure.

But how do users judge whether ISP download is in progress? When to write 0x60 to register IAP_CONTR to trigger a soft reset? Regarding these two issues, four methods of judgment are introduced below:

## Use P3.0 port to detect serial port start signal

The serial port ISP of the STC microcontroller uses P3.0 and P3.1. When the ISP download software starts to download, it will send a handshake command to the P3.0 port of the microcontroller. If the user's P3.0 and P3.1 are only used for ISP download, you can use the P3.0 port to detect the start signal of the serial port to judge the ISP download.

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     IAP_CONTR   =   0xc7;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;
sbit    P30         =   P3^0;

void main()
{
    P3M0 = 0x00;
    P3M1 = 0x00;
    P30 = 1;

    while (1)
    {
        if (!P30) IAP_CONTR = 0x60;          // The low level of P3.0 is the serial port start signal
                                             // Software reset to system area

        ...                                  //User codes
    }
```

```
}
```

## Use the falling edge interrupt of P3.0/INT4 to detect the serial port start signal

Method B is similar to method A, except that method A uses query mode, and method B uses interrupt mode. Because the P3.0 port of the STC single-chip computer is the interrupt port of INT4.

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     IAP_CONTR   =   0xc7;
sfr     INTCLKO     =   0x8f;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;

void Int4Isr() interrupt 16                 //INT4 interrupt service routine
{
    IAP_CONTR = 0x60;                       // UART start signal triggers INT4 interrupt
                                            // Software reset to system area
}

void main()
{
    P3M0 = 0x00;
    P3M1 = 0x00;

    INTCLKO |= 0x40;                        // Enable INT4 interrupt
    EA = 1;

    while (1)
    {
        ...                                 //User codes
    }
}
```

## Use P3.0/RxD to receive and detect the 7F sent by the ISP download software

Method A and Method B are very simple, but easy to be interfered. If there is any interference signal on P3.0 port, it will trigger the software Reset, so method C is to verify the serial port data.

When STC's ISP download software performs ISP download, it will first use the lowest baud rate (usually 2400) + even parity 9+1 stop bit to continuously send the handshake command 7F, so the user can set the serial port to 9 in the program Bit data bit + 2400 baud rate, and then continue to detect 7F. For example, if 8 7Fs are continuously detected, it means that ISP download is required, and then a software reset is triggered.

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define FOSC            11059200UL
#define BR2400          (65536 - FOSC / 4 / 2400)

sfr     IAP_CONTR   =   0xc7;
```

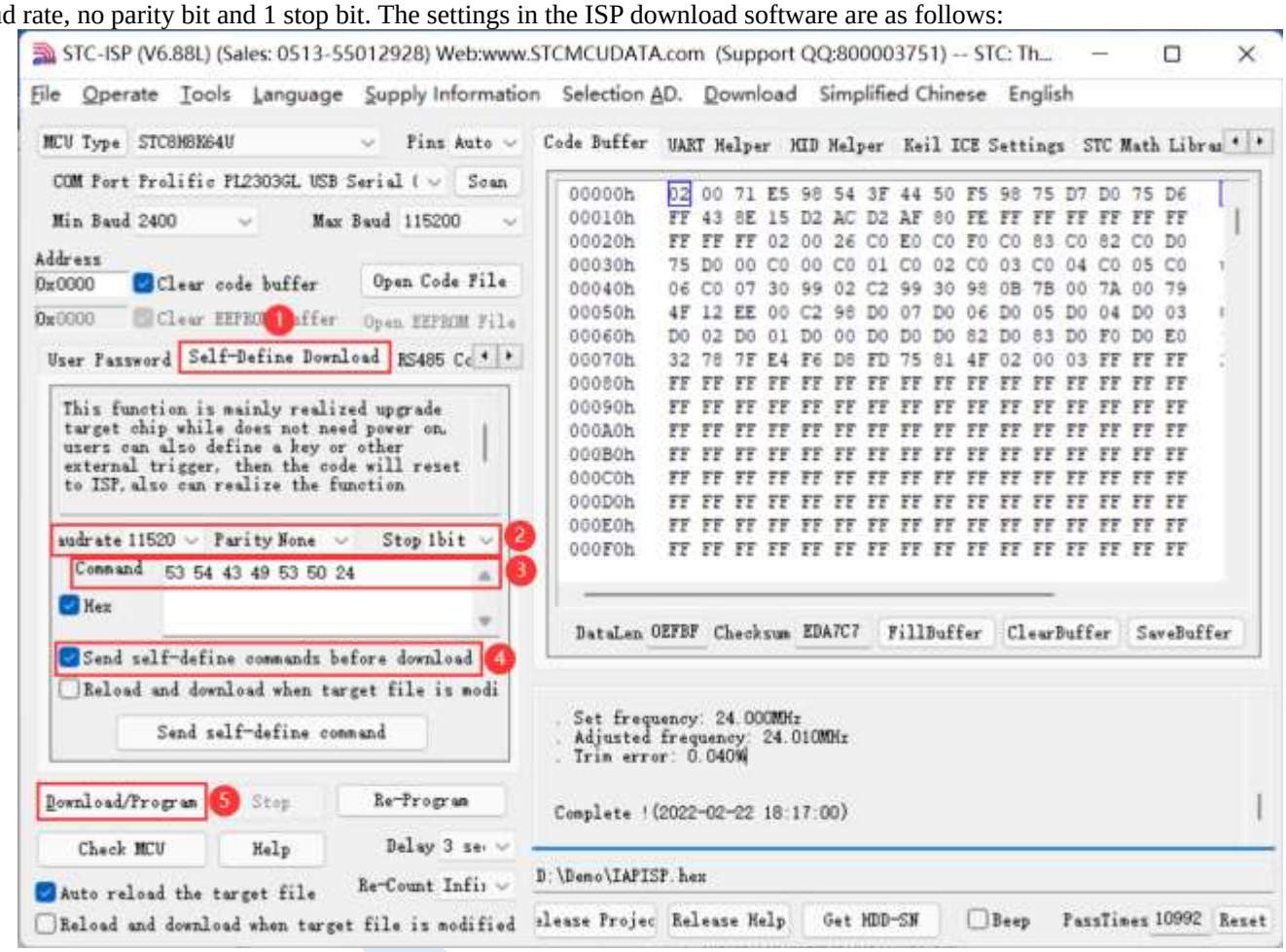
```
sfr         AUXR        =   0x8e;
sfr         P3M0        =   0xb2;
sfr         P3M1        =   0xb1;

char cnt7f;

void UartIsr() interrupt 4                  // UART Interrupt Service Routine
{
    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;
        if ((SBUF == 0x7f) && (RB8 == 1))   // Handshake command 7F sent by ISP download software
                                            //The even parity bit of 7F is 1
        {
            if (++cnt7f == 8)               // When 8 consecutive 7Fs are detected
                IAP_CONTR = 0x60;           // reset to system area
        }
        else
        {
            cnt7f = 0;
        }
    }
}

void main()
{
    P3M0 = 0x00;
    P3M1 = 0x00;

    SCON = 0xd0;                            // Set the UART to 9 data bits
    TMOD = 0x00;
    AUXR = 0x40;
    TH1 = BR2400 >> 8;                      // Set the UART baud rate to 2400
    TL1 = BR2400;
    TR1 = 1;
    ES = 1;
    EA = 1;

    cnt7f = 0;

    while (1)
    {
        ...                                 //User codes
    }
}
```

## Use P3.0/RxD to receive and detect user download commands sent by ISP download software

If the user code needs to use the serial port for communication, the above 3 methods may not be applicable. At this time, you can use the interface provided by the STC ISP download software to customize a set of dedicated user download commands (you can specify the baud rate, Check bit and stop bit). If this function is enabled, the ISP download software will use the user-specified baud rate, check bit and stop bit to send the user download command

before ISP download, and then send the handshake command. The user only needs to monitor the serial port command sequence in his own code. When the correct user download command is detected, the software is reset to the system area to realize the ISP function without power failure.

The following assumes that the user download command is the string "STCISP$", the serial port is set to 115200 baud rate, no parity bit and 1 stop bit. The settings in the ISP download software are as follows:




User sample code is as follows:

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define   FOSC          11059200UL
#define   BR115200      (65536 - FOSC / 4 / 115200)

sfr       IAP_CONTR  =  0xc7;
sfr       AUXR       =  0x8e;
sfr       P3M0       =  0xb2;
sfr       P3M1       =  0xb1;

char stage;

void UartIsr() interrupt 4                    // UART Interrupt Service Routine
{
    char dat;
```

```
        if (TI)
        {
            TI = 0;
        }

        if (RI)
        {
            RI = 0;

            dat = SBUF;
            switch (stage)
            {
            case 0:
            default:
L_Check1st:
                if (dat == 'S') stage = 1;
                else stage = 0;
                break;
            case 1:
                if (dat == 'T') stage = 2;
                else goto L_Check1st;
                break;
            case 2:
                if (dat == 'C') stage = 3;
                else goto L_Check1st;
                break;
            case 3:
                if (dat == 'I') stage = 4;
                else goto L_Check1st;
                break;
            case 4:
                if (dat == 'S') stage = 5;
                else goto L_Check1st;
                break;
            case 5:
                if (dat == 'P') stage = 6;
                else goto L_Check1st;
                break;
            case 6:
                if (dat == '$')                 // When the correct user download command is detected
                    IAP_CONTR = 0x60;           // reset to system area
                else goto L_Check1st;
            break;
            }
        }
}

void main()
{
        P3M0 = 0x00;
        P3M1 = 0x00;

        SCON = 0x50;                            // Set the UART to 8 data bits
        TMOD = 0x00;
        AUXR = 0x40;
        TH1 = BR2400 >> 8;                      // Set the UART baud rate to 115200
        TL1 = BR2400;
        TR1 = 1;
```

```
    ES = 1;
    EA = 1;

    stage = 0;

    while (1)
    {
        ...                                     // User codes
    }
}
```

# Appendix N Example Routine of ISP download for STC8H series MCUs using third-party MCU

## C language code

```
/*Note: When using this code to download the STC8H series of microcontrollers, you must execute the Download code before powering on the target chip, otherwise the target chip will not download correctly.*/

#include "reg51.h"

typedef    bit              BOOL;
typedef    unsigned char    BYTE;
typedef    unsigned short   WORD;

//Macro and constant definition
#define    FALSE            0
#define    TRUE             1
#define    LOBYTE(w)        ((BYTE)(WORD)(w))
#define    HIBYTE(w)        ((BYTE)((WORD)(w) >> 8))

#define    MINBAUD          2400L
#define    MAXBAUD          115200L

#define    FOSC             11059200L          //Main chip working frequency
#define    BR(n)            (65536 - FOSC/4/(n))   // Calculation formula of serial port baud rate of main chip
#define    T1MS             (65536 - FOSC/1000)    // 1ms timing initial value of main chip

#define    FUSER            24000000L          //STC8H Series target chip operating frequency
#define    RL(n)            (65536 - FUSER/4/(n))  //STC8H Serial target chip baud rate calculation formula

sfr        AUXR = 0x8e;
sfr        P3M1 = 0xB1;
sfr        P3M0 = 0xB2;

// Variable definitions
BOOL f1ms;                          //1ms flag
BOOL UartBusy;                      // Serial transmit busy flag
BOOL UartReceived;                  // Serial data receiving completion flag
BYTE UartRecvStep;                  // Serial data receiving control
BYTE TimeOut;                       // Serial communication timeout counter
BYTE xdata TxBuffer[256];           // Serial data transmission buffer
BYTE xdata RxBuffer[256];           // Serial data receiving buffer
char code DEMO[256];                // Demo code data
```

```
// Functions declarations
void Initial(void);
void DelayXms(WORD x);
BYTE UartSend(BYTE dat);
void CommInit(void);
void CommSend(BYTE size);
BOOL Download(BYTE *pdat, long size);

// Main function entry
void main(void)
{
	P3M0 = 0x00;
	P3M1 = 0x00;

	Initial();
	if (Download(DEMO, 256))
	{
		// download successfully
		P3 = 0xff;
		DelayXms(500);
		P3 = 0x00;
		DelayXms(500);
		P3 = 0xff;
		DelayXms(500);
		P3 = 0x00;
		DelayXms(500);
		P3 = 0xff;
		DelayXms(500);
		P3 = 0x00;
		DelayXms(500);
		P3 = 0xff;
	}
	else
	{
		// download failed
		P3 = 0xff;
		DelayXms(500);
		P3 = 0xf3;
		DelayXms(500);
		P3 = 0xff;
		DelayXms(500);
		P3 = 0xf3;
		DelayXms(500);
		P3 = 0xff;
		DelayXms(500);
		P3 = 0xf3;
		DelayXms(500);
		P3 = 0xff;
	}

	while (1);
}

//1ms Timer interrupt service routine
void tm0(void) interrupt 1
{
	static BYTE Counter100;
```

```
        f1ms = TRUE;
        if (Counter100-- == 0)
        {
            Counter100 = 100;
            if (TimeOut) TimeOut--;
        }
}

// Serial port interrupt service routine
void uart(void) interrupt 4
{
        static WORD RecvSum;
        static BYTE RecvIndex;
        static BYTE RecvCount;
        BYTE dat;

        if (TI)
        {
            TI = 0;
            UartBusy = FALSE;
        }

        if (RI)
        {
            RI = 0;
            dat  = SBUF;
            switch (UartRecvStep)
            {
            case 1:
                if (dat != 0xb9) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 2:
                if (dat != 0x68) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 3:
                if (dat != 0x00) goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 4:
                RecvSum = 0x68 + dat;
                RecvCount = dat - 6;
                RecvIndex = 0;
                UartRecvStep++;
                break;
            case 5:
                RecvSum += dat;
                RxBuffer[RecvIndex++] = dat;
                if (RecvIndex == RecvCount)     UartRecvStep++;
                break;
            case 6:
                if (dat != HIBYTE(RecvSum))     goto L_CheckFirst;
                UartRecvStep++;
                break;
            case 7:
                if (dat != LOBYTE(RecvSum))     goto L_CheckFirst;
                UartRecvStep++;
```

```
                break;
            case 8:
                if (dat != 0x16) goto L_CheckFirst;
                UartReceived = TRUE;
                UartRecvStep++;
                break;
L_CheckFirst:
            case 0:
            default:
                CommInit();
                UartRecvStep = (dat == 0x46       ? 1 : 0);
                break;
            }
        }
}

// system initialization
void Initial(void)
{
        UartBusy = FALSE;

        SCON = 0xd0;                        // Serial data format must be 8-bit data + 1-bit even check
        AUXR = 0xc0;
        TMOD = 0x00;
        TH0 = HIBYTE(T1MS);
        TL0 = LOBYTE(T1MS);
        TR0 = 1;
        TH1 = HIBYTE(BR(MINBAUD));
        TL1 = LOBYTE(BR(MINBAUD));
        TR1 = 1;
        ET0 = 1;
        ES = 1;
        EA = 1;
}

//Xms Delay program
void DelayXms(WORD x)
{
        do
        {
            f1ms = FALSE;
            while (!f1ms);
        } while (x--);
}

// Serial data sending program
BYTE UartSend(BYTE dat)
{
        while (UartBusy);

        UartBusy = TRUE;
        ACC = dat;
        TB8 = P;
        SBUF = ACC;

        return dat;
}
```

```
// Serial communication initialization
void CommInit(void)
{
	UartRecvStep = 0;
	TimeOut = 20;
	UartReceived = FALSE;
}

// Send serial communication packets
void CommSend(BYTE size)
{
	WORD sum;
	BYTE i;

	UartSend(0x46);
	UartSend(0xb9);
	UartSend(0x6a);
	UartSend(0x00);
	sum = size + 6 + 0x6a;
	UartSend(size + 6);
	for (i=0; i<size; i++)
	{
		sum  += UartSend(TxBuffer[i]);
	}
	UartSend(HIBYTE(sum));
	UartSend(LOBYTE(sum));
	UartSend(0x16);
	while (UartBusy);

	CommInit();
}

//对 STC15H Series of chips for ISP download
BOOL Download(BYTE *pdat, long size)
{
	BYTE arg;
	BYTE offset;
	BYTE cnt;
	WORD addr;

	// Shake hands
	CommInit();
	while (1)
	{
		if (UartRecvStep == 0)
		{
			UartSend(0x7f);
			DelayXms(10);
		}
		if (UartReceived)
		{
			arg = RxBuffer[4];
			if (RxBuffer[0] == 0x50) break;
			return FALSE;
		}
	}

	// Set parameters (set the parameters such as the highest baud rate used by the slave chip and erase wait time)
```

```
TxBuffer[0] = 0x01;
TxBuffer[1] = arg;
TxBuffer[2] = 0x40;
TxBuffer[3] = HIBYTE(RL(MAXBAUD));
TxBuffer[4] = LOBYTE(RL(MAXBAUD));
TxBuffer[5] = 0x00;
TxBuffer[6] = 0x00;
TxBuffer[7] = 0x97;
CommSend(8);
while (1)
{
    if (TimeOut == 0) return FALSE;
    if (UartReceived)
    {
        if (RxBuffer[0] == 0x01) break;
        return FALSE;
    }
}

//prepare
TH1 = HIBYTE(BR(MAXBAUD));
TL1 = LOBYTE(BR(MAXBAUD));
DelayXms(10);
TxBuffer[0] = 0x05;
TxBuffer[1] = 0x00;
TxBuffer[2] = 0x00;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
CommSend(5);
while (1)
{
    if (TimeOut == 0) return FALSE;
    if (UartReceived)
    {
        if (RxBuffer[0] == 0x05) break;
        return FALSE;
    }
}

// Erase
DelayXms(10);
TxBuffer[0] = 0x03;
TxBuffer[1] = 0x00;
TxBuffer[2] = 0x00;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
CommSend(5);
TimeOut = 100;
while (1)
{
    if (TimeOut == 0) return FALSE;
    if (UartReceived)
    {
        if (RxBuffer[0] == 0x03) break;
        return FALSE;
    }
}
```

```
// Write user code
DelayXms(10);
addr = 0;
TxBuffer[0] = 0x22;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
offset = 5;
while (addr < size)
{
    TxBuffer[1] = HIBYTE(addr);
    TxBuffer[2] = LOBYTE(addr);
    cnt   = 0;
    while (addr < size)
    {
        TxBuffer[cnt+offset] = pdat[addr];
        addr++;
        cnt++;
        if (cnt >= 128) break;
    }
    CommSend(cnt + offset);
    while (1)
    {
        if (TimeOut == 0) return FALSE;
        if (UartReceived)
        {
            if ((RxBuffer[0] == 0x02) && (RxBuffer[1] == 'T')) break;
            return FALSE;
        }
    }
    TxBuffer[0] = 0x02;
}

//// Write hardware options
//// If you do not need to modify the hardware options, this step can be skipped directly.At this time, all the hardware options
//// remain unchanged, the frequency of the MCU is the last adjusted frequency
//// If you write the hardware option, the MCU's internal IRC frequency will be fixed to 24MHz,
//// and other options will be restored to the factory settings.
////Suggestion: Set the hardware options of the slave chip when you use STC-ISP download software the first time.

//// Do not write hardware options when downloading programs from the master chip to the slave chip.
//DelayXms(10);
//for (cnt=0; cnt<128; cnt++)
//{
//      TxBuffer[cnt] = 0xff;
//}
//TxBuffer[0] = 0x04;
//TxBuffer[1] = 0x00;
//TxBuffer[2] = 0x00;
//TxBuffer[3] = 0x5a;
//TxBuffer[4] = 0xa5;
//TxBuffer[33] = arg;
//TxBuffer[34] = 0x00;
//TxBuffer[35] = 0x01;
//TxBuffer[41] = 0xbf;
//TxBuffer[42] = 0xbd;                  //P5.4 is I/O port
////TxBuffer[42] = 0xad;                //P5.4 is reset pin
//TxBuffer[43] = 0xf7;
//TxBuffer[44] = 0xff;
```

```
    //CommSend(45);
    //while (1)
    //{
    //    if (TimeOut == 0) return FALSE;
    //    if (UartReceived)
    //    {
    //        if ((RxBuffer[0] == 0x04) && (RxBuffer[1] == 'T')) break;
    //        return FALSE;
    //    }
    //}

    // Download completed
    return TRUE;
}

char code DEMO[256] =
{
    0x80,0x00,0x75,0xB2,0xFF,0x75,0xB1,0x00,0x05,0xB0,0x11,0x0E,0x80,0xFA,0xD8,0xFE,
    0xD9,0xFC,0x22,
};
```

Note: If user needs to set different working frequencies, please refer to the example codes in chapters 7.3.7 and 7.3.8.

# Appendix O Use a third-party application program to call the STC release project program to download the ISP of the MCU

The release project program generated by STC's ISP download software is an executable EXE format file. The user can directly double-click the released project program to run it for ISP download, or call the release project program in a third-party application for ISP download. Two methods of calling are introduced below.

## Simple call

In the third-party application, it is only a simple process of creating and publishing the project program. All other download operations are carried out in the publishing project program. The third-party application only needs to wait for the completion of the publishing project program and clean the scene.

**VC code**

```
BOOL IspProcess()
{
	// define related variables
	STARTUPINFO si;
	PROCESS_INFORMATION pi;
	CString path;

	// Full path of publishing project program
	path = _T("D:\\Work\\Upgrade.exe");

	// variable initialization
	memset(&si, 0, sizeof(STARTUPINFO));
	memset(&pi, 0, sizeof(PROCESS_INFORMATION));

	// Set startup variables
	si.cb = sizeof(STARTUPINFO);
	GetStartupInfo(&si);
	si.wShowWindow = SW_SHOWNORMAL;
	si.dwFlags = STARTF_USESHOWWINDOW;

	// Create Publish Project Program Process
	if (CreateProcess(NULL, (LPTSTR)(LPCTSTR)path, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		// Wait for the publish project program operation to complete
		// Since the main process will be blocked here, it is recommended to create a new working process and wait in the worker process
		WaitForSingleObject(pi.hProcess,INFINITE);
```

```
        // clean up
        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);

        return TRUE;
    }
    else
    {
        AfxMessageBox(_T("创建进程失败!"));

        return FALSE;
    }
}
```

## Advanced call

The process of creating and publishing project programs in third-party applications, including selecting serial ports and starting ISP in third-party applications

All ISP download operations such as programming, ISP programming with vibration stopped, and closing the release project program, do not require interface interaction in the release project program.

### VC 代码

```
// A data structure that defines the parameters of the callback function
struct CALLBACK_PARAM
{
    DWORD dwProcessId;                  // main process id
    HWND hMainWnd;                      // main window handle
};

// Callback function for enumerating windows to get the main window handle
BOOL CALLBACK EnumWindowCallBack(HWND hWnd, LPARAM lParam)
{
    CALLBACK_PARAM *pcp = (CALLBACK_PARAM *)lParam;
    DWORD id;

    GetWindowThreadProcessId(hWnd, &id);
    if ((pcp->dwProcessId == id) && (GetParent(hWnd) == NULL))
    {
        pcp->hMainWnd = hWnd;

        return FALSE;
    }

    return TRUE;
}

BOOL IspProcess()
{
    // define related variables
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    CALLBACK_PARAM cp;
    CString path;

    // the IDs of some controls in the publishing project program
    const UINT ID_PROGRAM       = 1046;
    const UINT ID_STOP          = 1044;
```

```
    const UINT ID_COMPORT       = 1009;
    const UINT ID_PROGRESS      = 1044;

    // Full path of publishing project program
    path = _T("D:\\Work\\Upgrade.exe");

    // variable initialization
    memset(&si, 0, sizeof(STARTUPINFO));
    memset(&pi, 0, sizeof(PROCESS_INFORMATION));
    memset(&cp, 0, sizeof(CALLBACK_PARAM));

    // Set startup variables
    si.cb = sizeof(STARTUPINFO);
    GetStartupInfo(&si);
    si.wShowWindow = SW_SHOWNORMAL;                 // If it is set to SW_HIDE here, the operation interface for publishing the project program will not be displayed, and all ISP operations can be performed in the background

    si.dwFlags = STARTF_USESHOWWINDOW;

    // Create a process for publishing a project program
    if (CreateProcess(NULL, (LPTSTR)(LPCTSTR)path, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
    {
        // Wait for the release project program process initialization to complete
        WaitForInputIdle(pi.hProcess, 5000);

        // Get the main window handle of the publishing project program
        cp.dwProcessId = pi.dwProcessId;
        cp.hMainWnd = NULL;
        EnumWindows(EnumWindowCallBack, (LPARAM)&cp);

        if (cp.hMainWnd != NULL)
        {
            HWND hProgram;
            HWND hStop;
            HWND hPort;

            // Get the handle of some controls in the main window of the publishing project program
            hProgram = ::GetDlgItem(cp.hMainWnd, ID_PROGRAM);
            hStop = ::GetDlgItem(cp.hMainWnd, ID_STOP);
            hPort = ::GetDlgItem(cp.hMainWnd, ID_COMPORT);

            // Set the serial port number in the release project program, the third parameter is 0:COM1, 1:COM2, 2:COM3, ...
            ::SendMessage(hPort, CB_SETCURSEL, 0, 0);

            // Trigger the programming button to start ISP programming
            ::SendMessage(hProgram, BM_CLICK, 0, 0);

            // wait for programming to complete,
            // Since the main process will be blocked here, it is recommended to create a new working process and wait in the worker process
            while (!::IsWindowEnabled(hProgram));

            // Close the release project program after programming is complete
            ::SendMessage(cp.hMainWnd, WM_CLOSE, 0, 0);
        }

        // wait for the process to end
        WaitForSingleObject(pi.hProcess,INFINITE);
```

```
        // clean up
        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);

        return TRUE;
    }
    else
    {
        AfxMessageBox(_T("创建进程失败!"));

        return FALSE;
    }
}
```

# Appendix P STC8H series orthogonal decoding example (courtesy of Chengdu Zhufei Technology)

Entrusted by STC, this article intends to share the use of the enhanced PWM module of the STC8H series of single-chip microcomputers to realize the orthogonal decoding function, so as to further realize the two-way speed measurement of the encoder output by the orthogonal code signal.

**Hardware platform:** Zhufei STC8H8K-64 pin core board + Zhufei 1024 line mini quadrature encoder

**Compilation environment:** keil V9.60

**Host computer:** serial port assistant

From the previous open source library of Zhufei Technology, we can understand that there is no orthogonal decoding routine in the open source library. The main reason is that STC8H only has two PWM modules. If we recommend using an orthogonal encoder, it means one encoding. However, this year's energy-saving group requires a balanced car to be made, which means that there are two motors, so two encoders are needed. Then the two PWM modules of the microcontroller will be occupied. However, the motor of the car Control also requires PWM function, so it is not recommended that you use the PWM module to achieve quadrature decoding. Instead, it is recommended that you use an encoder with directional output, so that the PWM module can be used by the motor.

Of course, there can also be another idea. It is feasible to use a PWM module to realize the speed measurement of an encoder with quadrature encoding, another motor uses an encoder with a direction signal, and an ordinary timer to capture pulses. Yes, but there is no need to be so troublesome.

One more thing to note is that the method of using the PWM module to count and using the timer module to capture pulse count is different. The PWM module captures the encoder data by counting the edges, which means that this module counts when a rising or falling edge occurs, and the timer captures pulses to obtain the number of high and low level flips. Here we will find through experiments that using the same quadrature encoder to rotate 360 °, the

encoder data collected by the PWM module is twice the pulse data captured by the timer, but this data does not increase the accuracy, but the count of the single-chip microcomputer. The way led to double the result.

The following is an example program of using STC8H8K64U to collect quadrature coded signal output encoder:

C Language Code

```
#include "headfile.h"

int16 encoder_data;

//------------------------------------------------------------------
//@brief        PWMA Module Orthogonal Decoding Initialization
//@param        void
//@return       void
//@since        v1.0
//Sample usage: PWMA_encoder_init();                    // Initialize Orthogonal Decoding
//@note
//------------------------------------------------------------------
void PWMA_encoder_init(void)
{
    P_SW2 |= 1<<7;                                      // Enable access to XFR
    PWMA_ARR = 0xFFFF;                                  // Set the auto-reload value. When the auto-reload value is 0, the
counter does not work.

    PWMA_CCMR1 |= 1<<0;                                 // IC1 is mapped on TI1FP1, i.e. use the P10 pin to get the
direction
    PWMA_CCMR2 |= 1<<0;                                 // IC2 is mapped on TI2FP2, that is, using the P22 pin to capture
edge transitions
    PWMA_SMCR |= 1<<0;                                  // Encoder mode 1, according to the level of TI1FP1, the counter
counts up/down on the edge of TI2FP2
    PWMA_CR1 |= 1<<0;                                   // Enable PWMA counter
    PWMA_PS |= 1<<2;                                    //Channel 1 of PWMA uses P10, and channel 2 of PWMB uses
P22.
}

//------------------------------------------------------------------
// @brief       PWMA module obtains the quadrature decoding value
// @param       void
// @return      void
// @since       v1.0
// Sample usage: encoder_data = PWMA_get_encoder();     // Get the quadrature decoding value
// @note
//------------------------------------------------------------------
int16 PWMA_get_encoder(void)
{
    int16 res;

    res = PWMA_CNTR;                                    // Save the current counter value
    PWMA_CNTR = 0;                                      // clear counter
    return res;
}

//------------------------------------------------------------------
// @brief       Interrupt service function of timer 0 5ms
// @param       void
// @return      void
// @since       v1.0
// Sample usage:
```

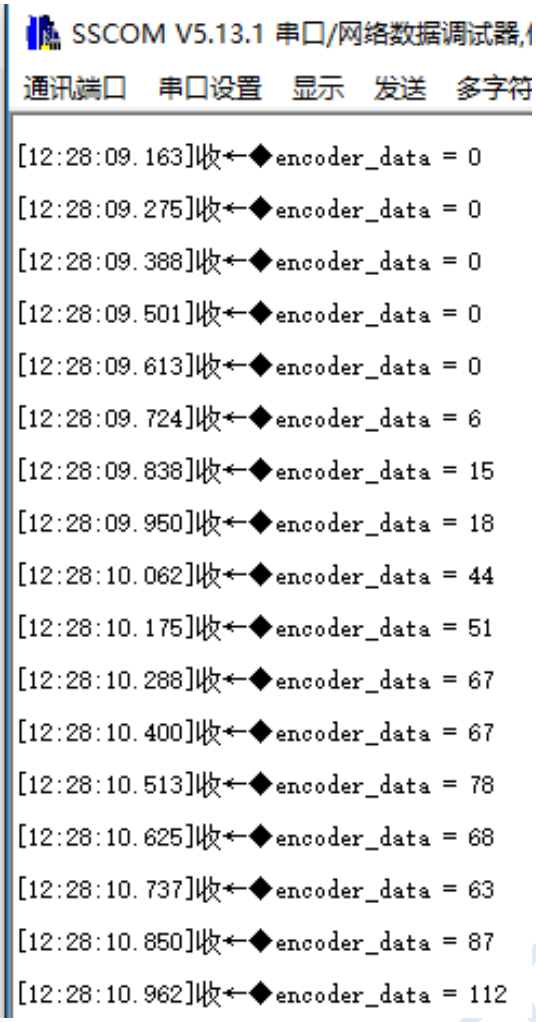
```
// @note
//-------------------------------------------------------------
void TM0_Isr() interrupt 1
{
    encoder_data = PWMA_get_encoder();          // Get the quadrature decoding encoder value
}

void main()
{
    DisableGlobalIRQ();                         // Disable CPU interrupt
    board_init();                               // Initialize internal registers, do not delete this code.
    pit_timer_ms(TIM_0, 5);                     // Initialize the timer, execute once in 5ms
    PWMA_encoder_init();                        // The PWMA module is initialized to the quadrature decoding
function
    EnableGlobalIRQ();                          //Enable CPU interrupt
    while(1)
    {
        delay_ms(100);                          // Output printing information every 100ms
        printf("encoder_data = %d \r\n" ,encoder_data);   // UART 1 prints encoder data
    }
}
```

Demo video link: https://www.bilibili.com/video/BV1zT4y177Ht

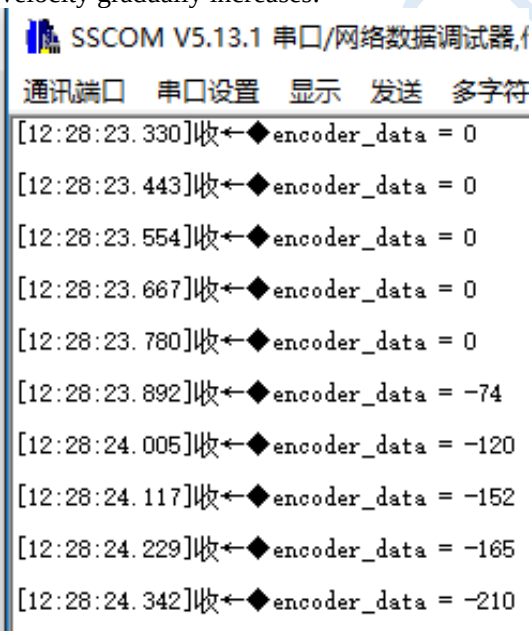
Video description: We compile the written routine, then download it to the microcontroller, open the serial port assistant to receive the print data of the microcontroller, rotate the encoder to observe the data changes, we find that the output data is 0 when the encoder is not rotating, and when the encoder faces Two kinds of values can be output when rotating in different directions. The faster the rotation, the greater the absolute value of the value. Positive and negative are used to indicate the two rotation directions. Which direction is positive and which direction is negative can be defined by yourself. At the same time, we also see from the program example that the print data is 100ms once, and the data collection is 5ms once, so the printed data is equivalent to intermittent. At the same time, because the encoder has a high precision of 1024 lines, it is observed that the data changes relatively large. , But if the motor is driven with no-load fixed PWM duty cycle, you can see that the encoder's data output is very stable.

Screenshot of serial port assistant receiving data:

The figure below is the data of the quadrature-encoded encoder rotating clockwise and the angular velocity gradually increasing

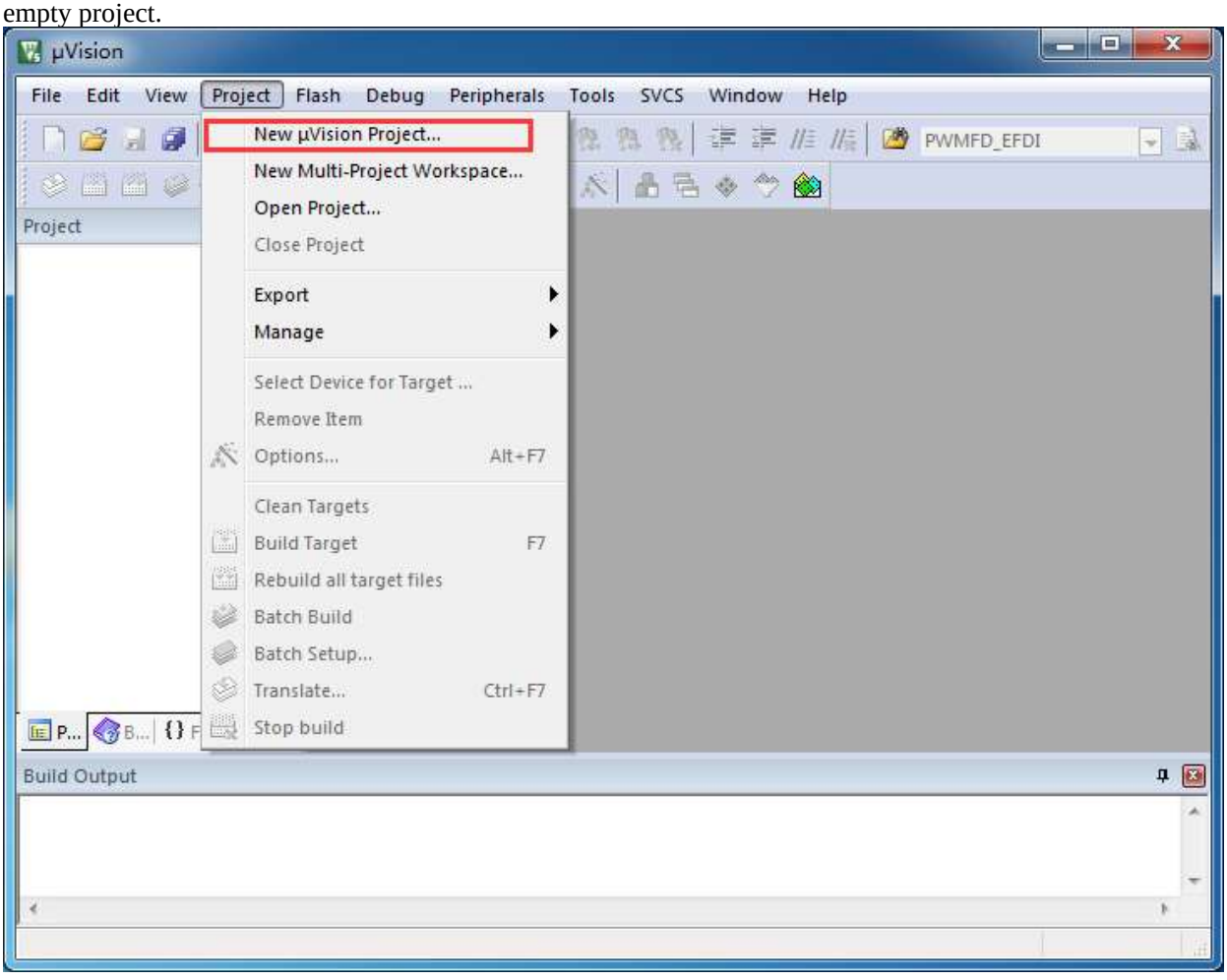



The following figure shows the data when the quadrature-encoded encoder rotates counterclockwise and the angular velocity gradually increases:

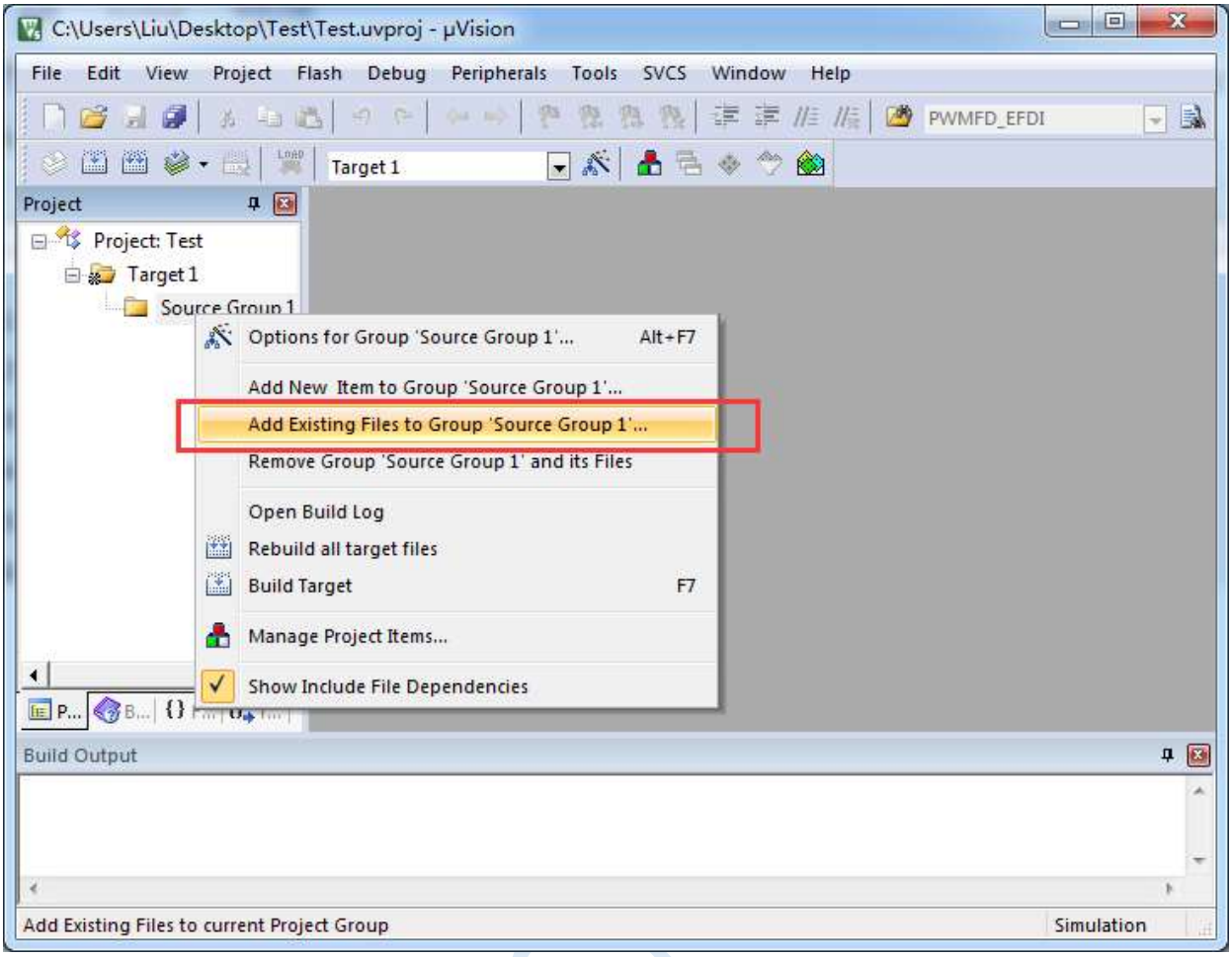


# Appendix Q Method for Creating Multi-file Projects in Keil

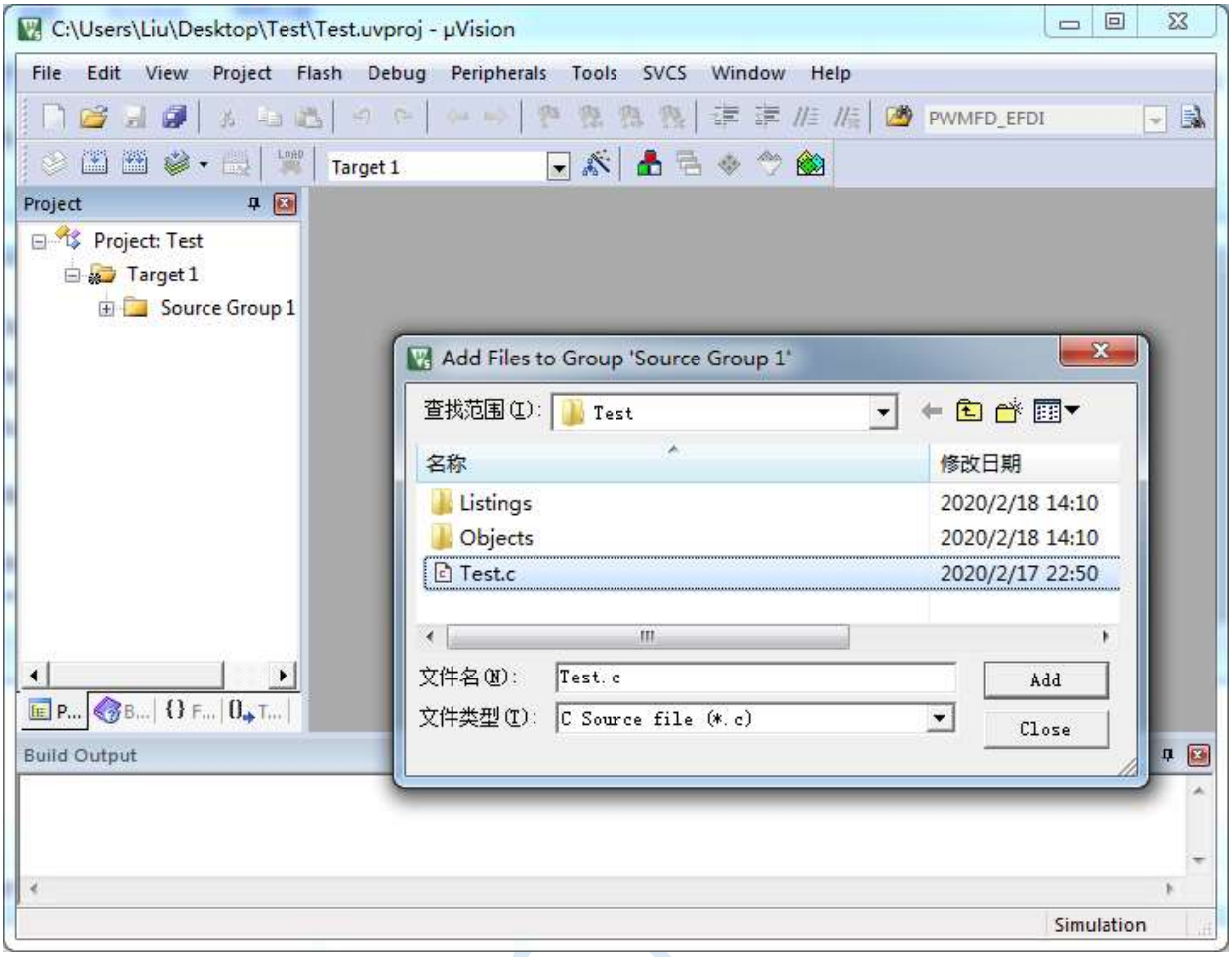
In Keil, relatively small projects generally have only one source file, but for some slightly more complex projects, multiple source files are often required. Here's how to set up a multi-file project:
1. Open Keil firstly and select "New uVision Project ..." from the "Project" menu to complete the creation of an empty project.



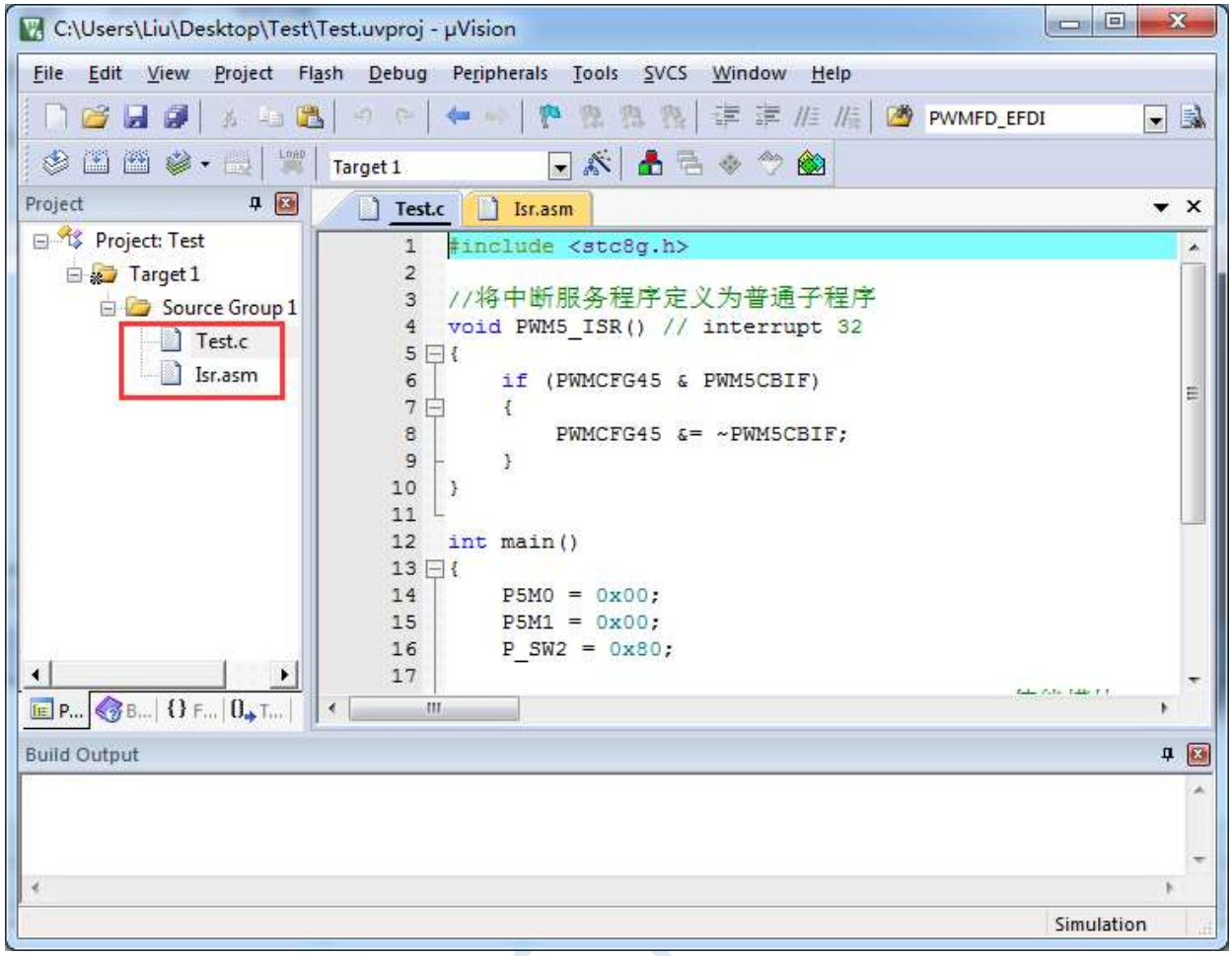

2. In the project tree of the empty project, right-click "Source Group 1" and select "Add Existing Files to Group" Source Group 1 "..." from the right-click menu.




3. In the file dialog that pops up, add the source file multiple times.

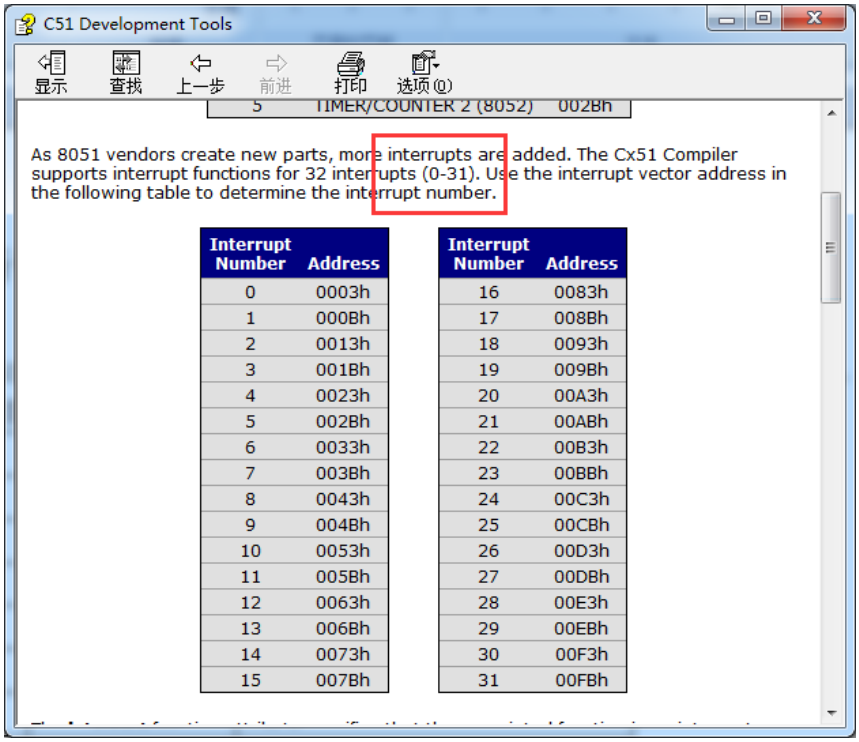



Complete the creation of the multi-file project as shown in the figure below.


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
Isr.asm
Test.c
Isr.asm
#include <stc8g.h>
//将中断服务程序定义为普通子程序
void PWM5_ISR() // interrupt 32
{
if (PWMCFG45 & PWM5CBIF)
{
PWMCFG45 &= ~PWM5CBIF;
}
}
int main()
{
P5M0 = 0x00;
P5M1 = 0x00;
P_SW2 = 0x80;
Build Output
Simulation

# Appendix R Handling of Compilation Error in Keil with Interrupt Numbers Greater Than 31

In Keil's C51 compilation environment, only 0 ~ 31 of the interrupt number are supported, that is, the interrupt vector must be less than 0100H.



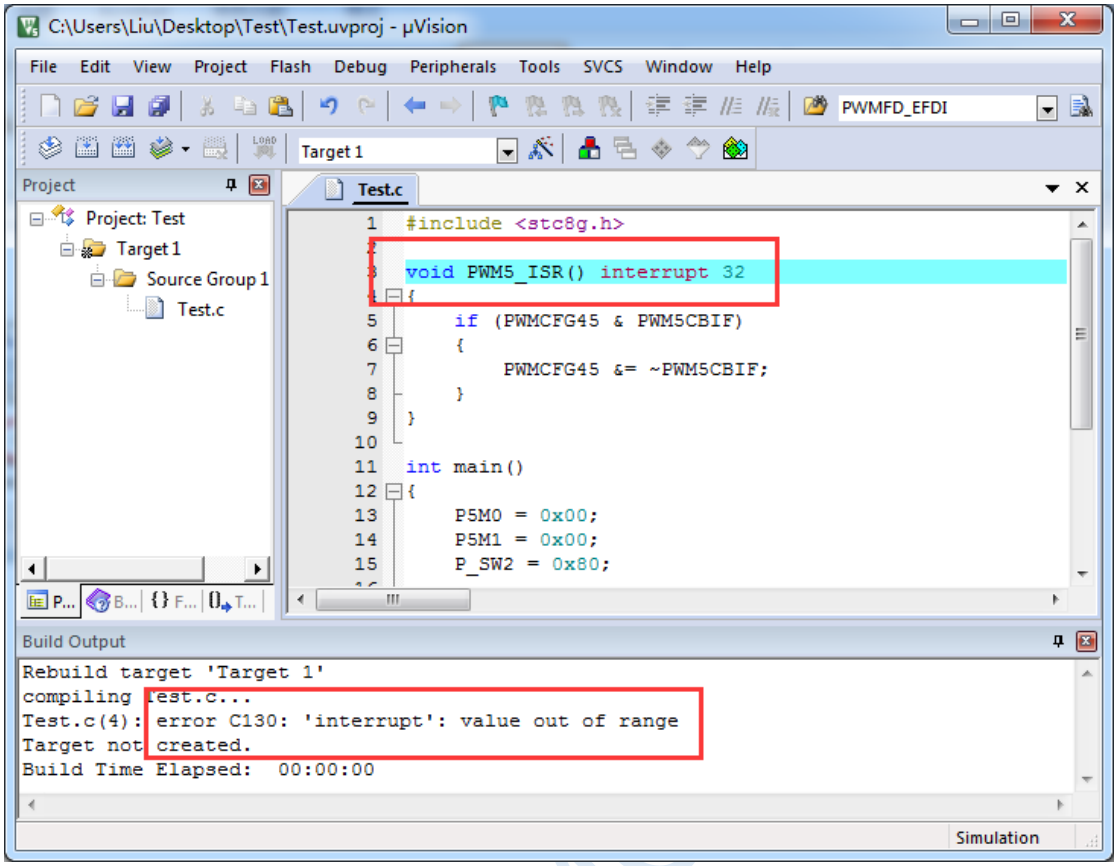

| Interrupt Number | Address | Interrupt Number | Address |
|---|---|---|---|
| 0 | 0003h | 16 | 0083h |
| 1 | 000Bh | 17 | 008Bh |
| 2 | 0013h | 18 | 0093h |
| 3 | 001Bh | 19 | 009Bh |
| 4 | 0023h | 20 | 00A3h |
| 5 | 002Bh | 21 | 00ABh |
| 6 | 0033h | 22 | 00B3h |
| 7 | 003Bh | 23 | 00BBh |
| 8 | 0043h | 24 | 00C3h |
| 9 | 004Bh | 25 | 00CBh |
| 10 | 0053h | 26 | 00D3h |
| 11 | 005Bh | 27 | 00DBh |
| 12 | 0063h | 28 | 00E3h |
| 13 | 006Bh | 29 | 00EBh |
| 14 | 0073h | 30 | 00F3h |
| 15 | 007Bh | 31 | 00FBh |

The following table is a list of interrupts for all current STC series:

| Interrupt number | Interrupt vector | Interrupt type |
|---|---|---|
| 0 | 0003 H | INT0 |
| 1 | 000B H | Timer 0 |
| 2 | 0013 H | INT1 |
| 3 | 001B H | Timer 1 |
| 4 | 0023 H | UART 1 |
| 5 | 002B H | ADC |
| 6 | 0033 H | LVD |
| 7 | 003B H | PCA |
| 8 | 0043 H | UART 2 |
| 9 | 004B H | SPI |
| 10 | 0053 H | INT2 |

| | | |
|---|---|---|
| 11 | 005B H | INT3 |
| 12 | 0063 H | Timer 2 |
| **13** | **006B H** | |
| 14 | 0073 H | System internal interrupt |
| 15 | 007B H | System internal interrupt |
| 16 | 0083 H | INT4 |
| 17 | 008B H | UART 3 |
| 18 | 0093 H | UART 4 |
| 19 | 009B H | Timer 3 |
| 20 | 00A3 H | Timer 4 |
| 21 | 00AB H | Comparator |
| 22 | 00B3 H | Waveform generator 0 |
| 23 | 00BB H | Waveform generator fault 0 |
| 24 | 00C3 H | I2C |
| 25 | 00CB H | USB |
| 26 | 00D3 H | PWMA |
| 27 | 00DB H | PWMB |
| 28 | 00E3 H | Waveform generator 1 |
| 29 | 00EB H | Waveform generator 2 |
| 30 | 00F3 H | Waveform generator 3 |
| 31 | 00FB H | Waveform generator 4 |
| **32** | **0103 H** | **Waveform generator 5** |
| **33** | **010B H** | **Waveform generator fault 2** |
| **34** | **0113 H** | **Waveform generator fault 4** |
| **35** | **011B H** | **Touch Key** |
| **36** | **0123 H** | **RTC** |
| **37** | **012B H** | **P0 interrupt** |
| **38** | **0133 H** | **P1 interrupt** |
| **39** | **013B H** | **P2 interrupt** |
| **40** | **0143 H** | **P3 interrupt** |
| **41** | **014B H** | **P4 interrupt** |
| **42** | **0153 H** | **P5 interrupt** |
| **43** | **015B H** | **P6 interrupt** |
| **44** | **0163 H** | **P7 interrupt** |
| **45** | **016B H** | **P8 interrupt** |
| **46** | **0173 H** | **P9 interrupt** |

It is not difficult to find that starting from the interrupt of the waveform generator 5, there will be errors when all subsequent interrupt service routines be compiled in keil, as shown in the following figure:

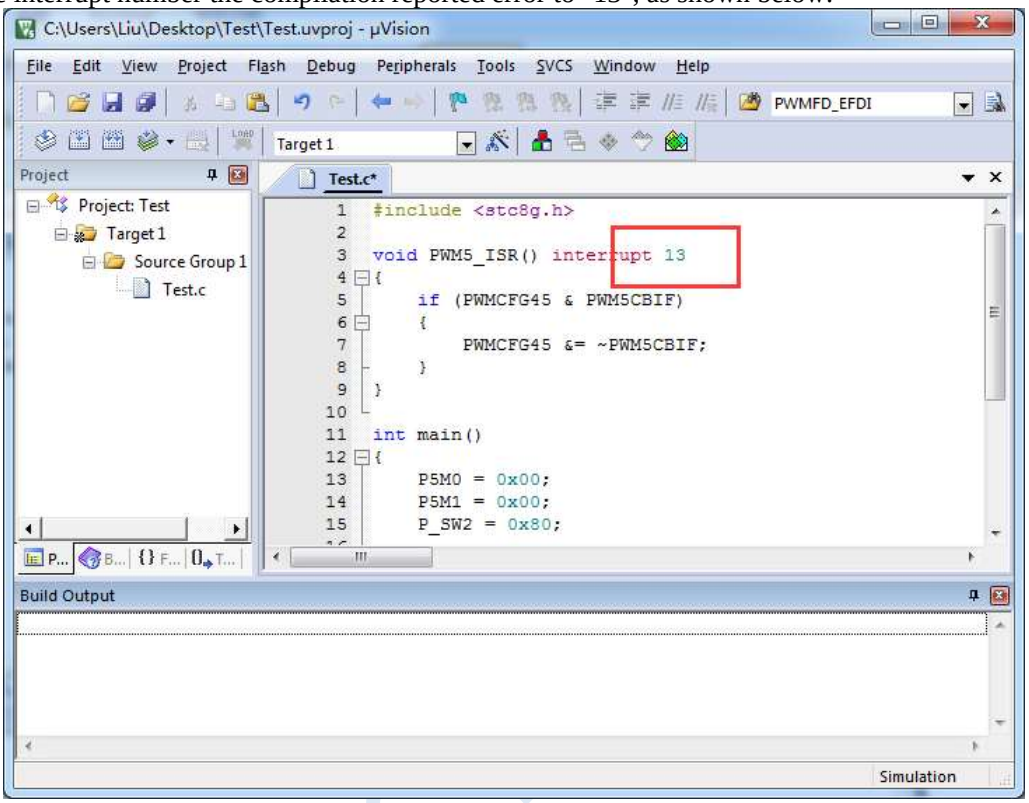



There are three ways to deal with this kind of error: (All of them need the help of assembly code, the first method is recommended)

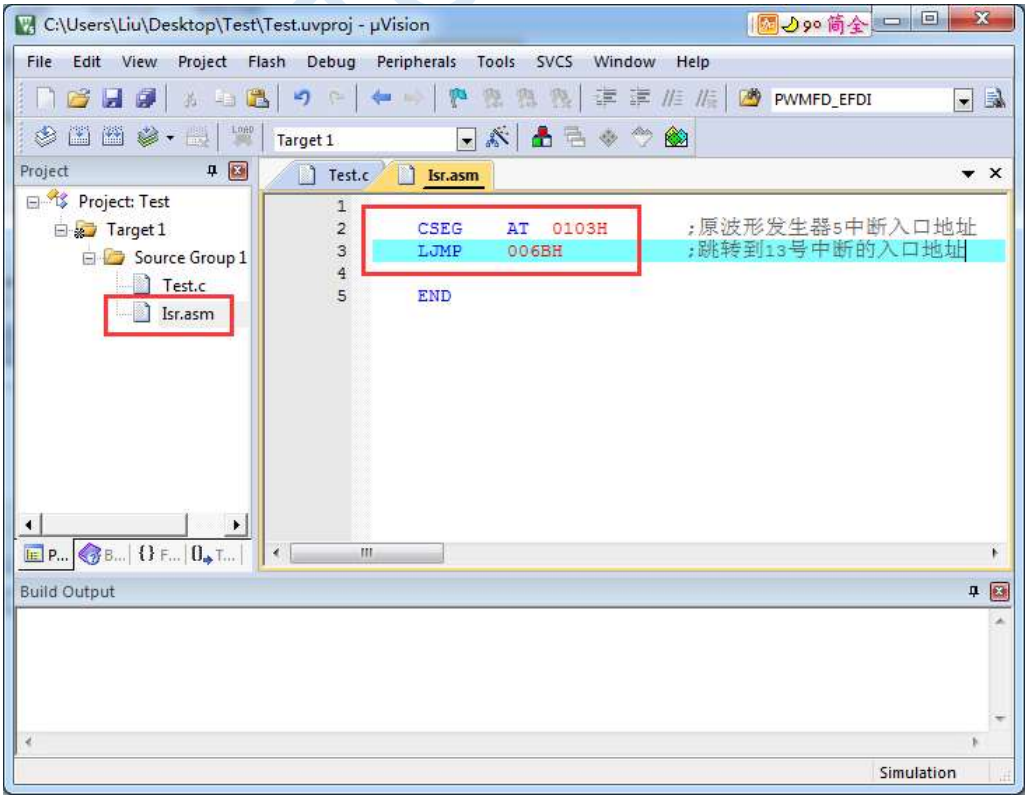
# Method 1: Borrow Interrupt Vector 13

Among interrupts 0 ~ 31, the 13th is a reserved interrupt number, we can borrow this interrupt number.
The steps are as follows:

1. Change the interrupt number the compilation reported error to "13", as shown below:

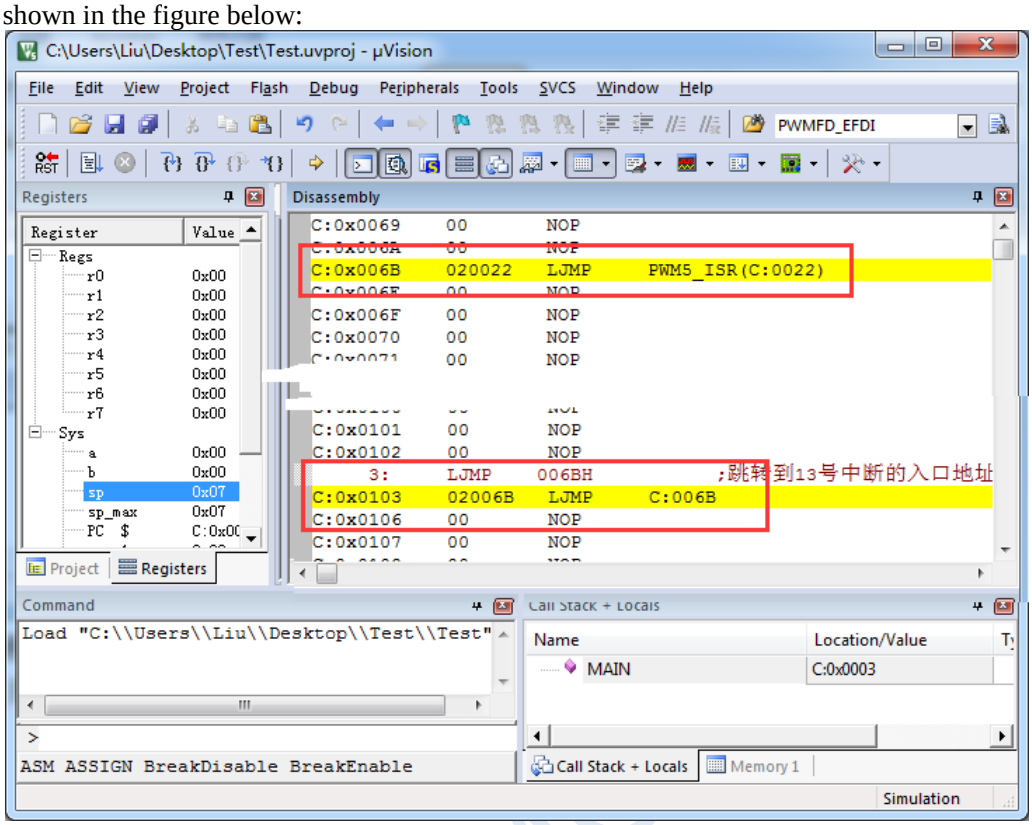



2. Create a new assembly language file, such as "isr.asm", add it to the project, and add "LJMP 006BH" at the address "0103H", as shown below:

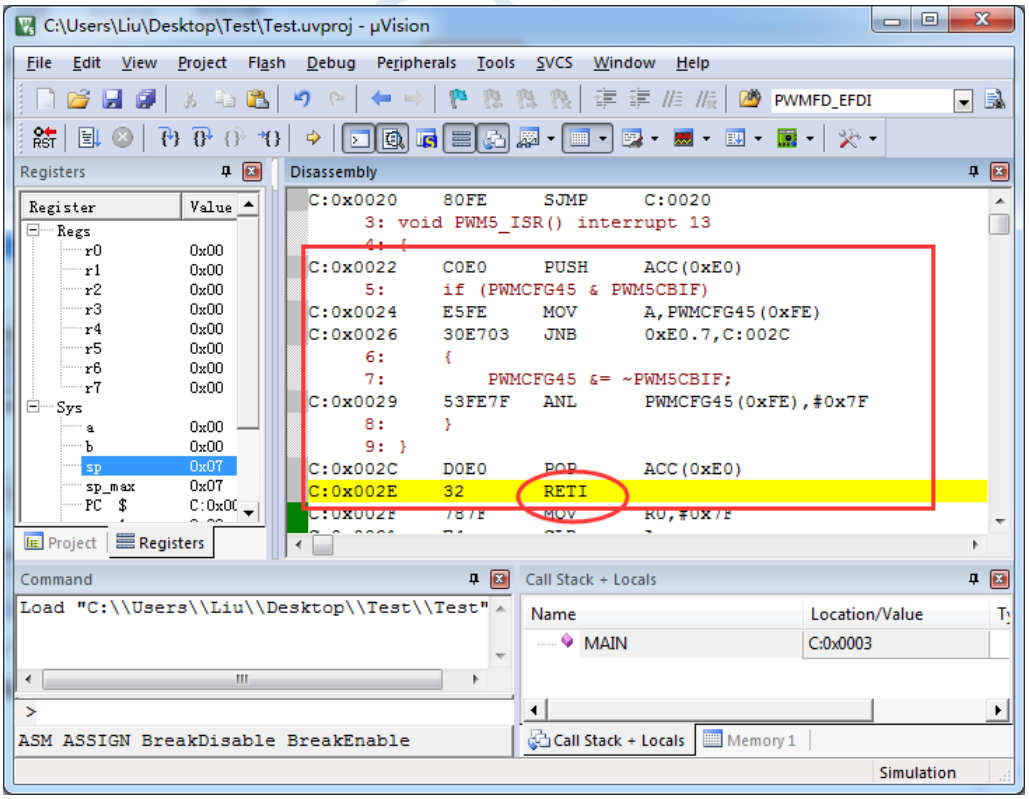



3. Compile successfully.

Now, after being compiled by Keil's C51 compiler, there is an "LJMP PWM5_ISR" at 006BH and an "LJMP 006BH" at 0103H, as shown in the figure below:



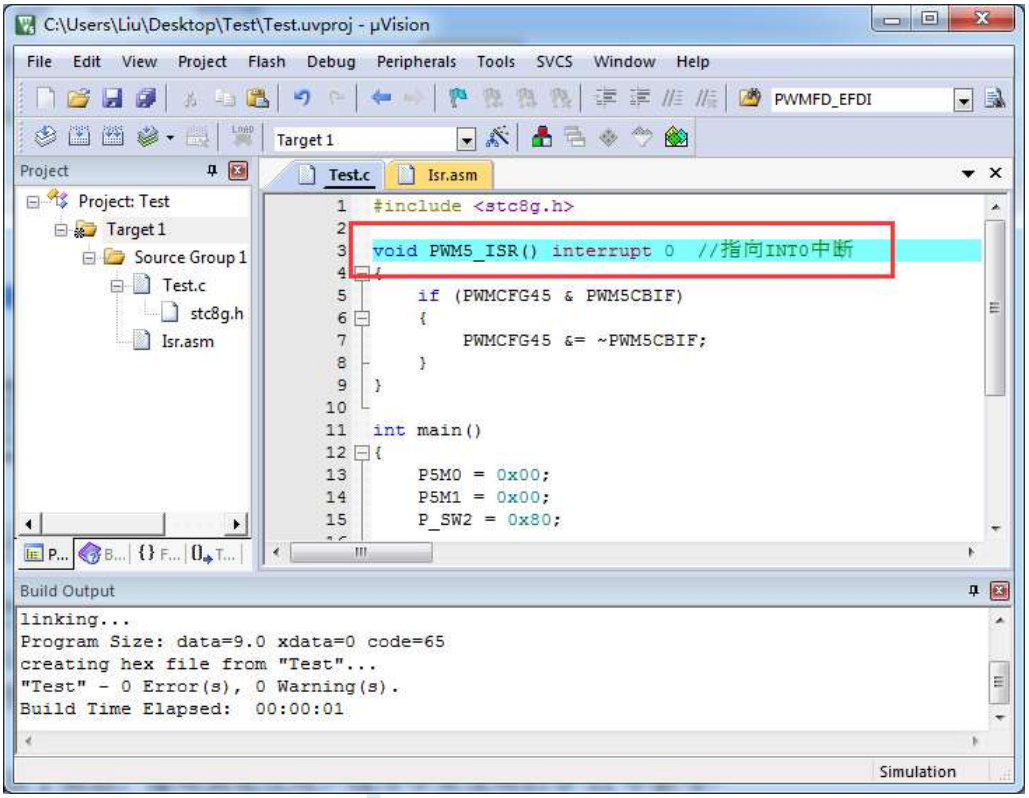

When the PWM5 interrupt occurs, the hardware will jump to the 0103H address automatically to execute "LJMP 006BH", and then execute "LJMP PWM5_ISR" at 006BH to jump to the real interrupt service routine, as shown in the figure below:

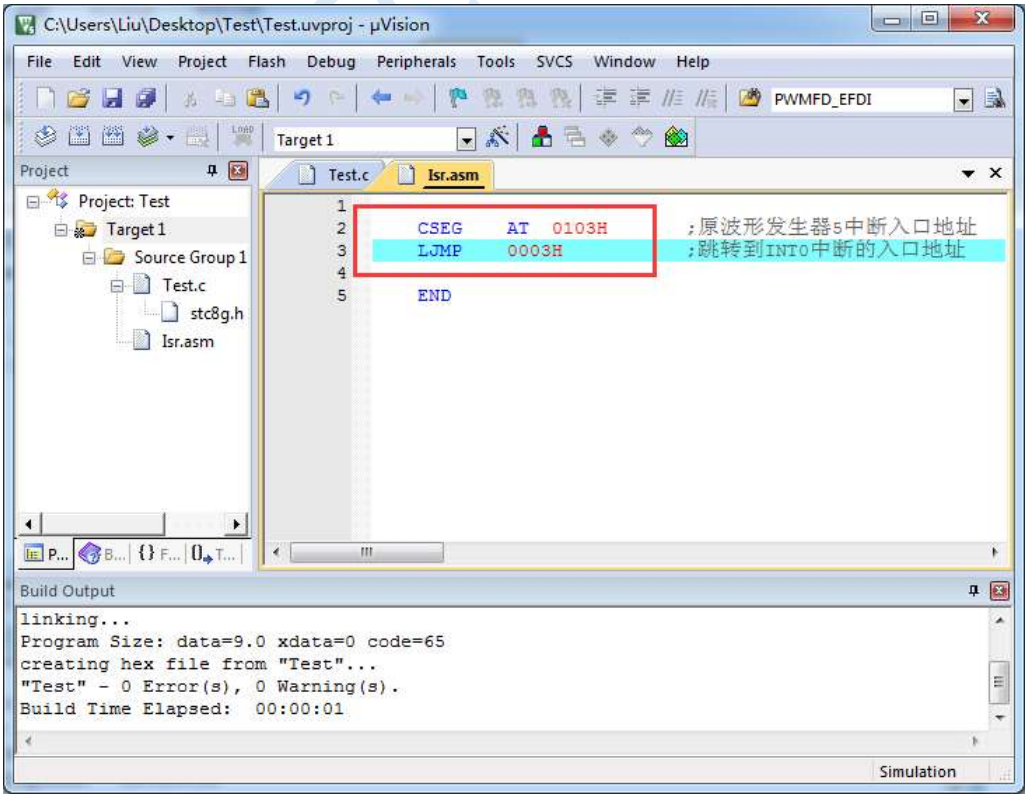


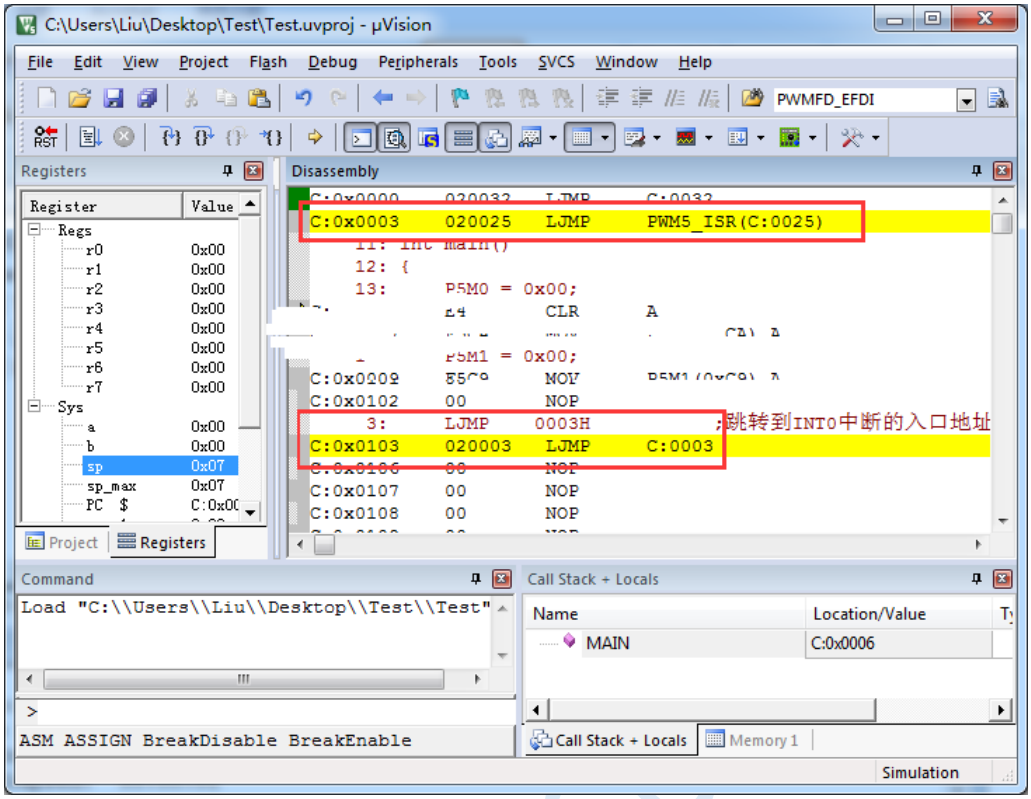

After the execution of the interrupt service routine is completed, it returns through the RETI instruction. The entire interrupt response process just executed an additional LJMP statement.

## Method 2: Similar to method 1, borrow unused interrupt numbers from 0 to 31 in user program.

For example, in the user's code, if the INT0 interrupt is not used, the above code can be modified similarly to method 1:



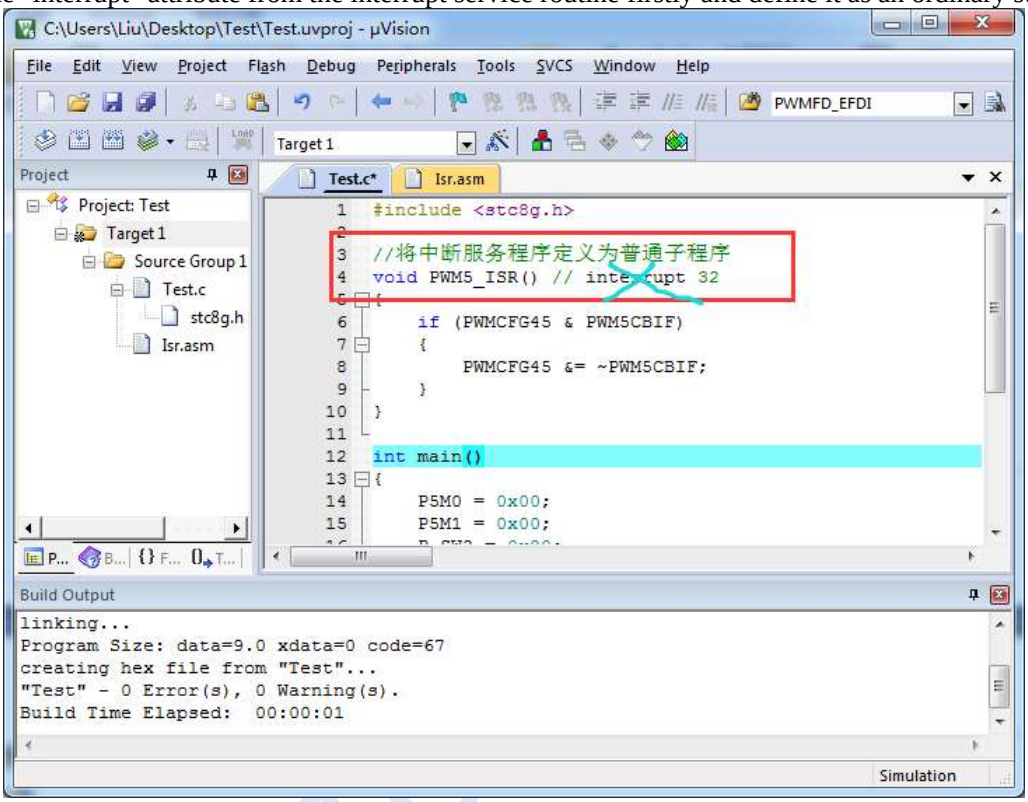



The execution effect is the same as Method 1. This method is applicable to the situation where multiple interrupt numbers greater than 31 need to be remapped.

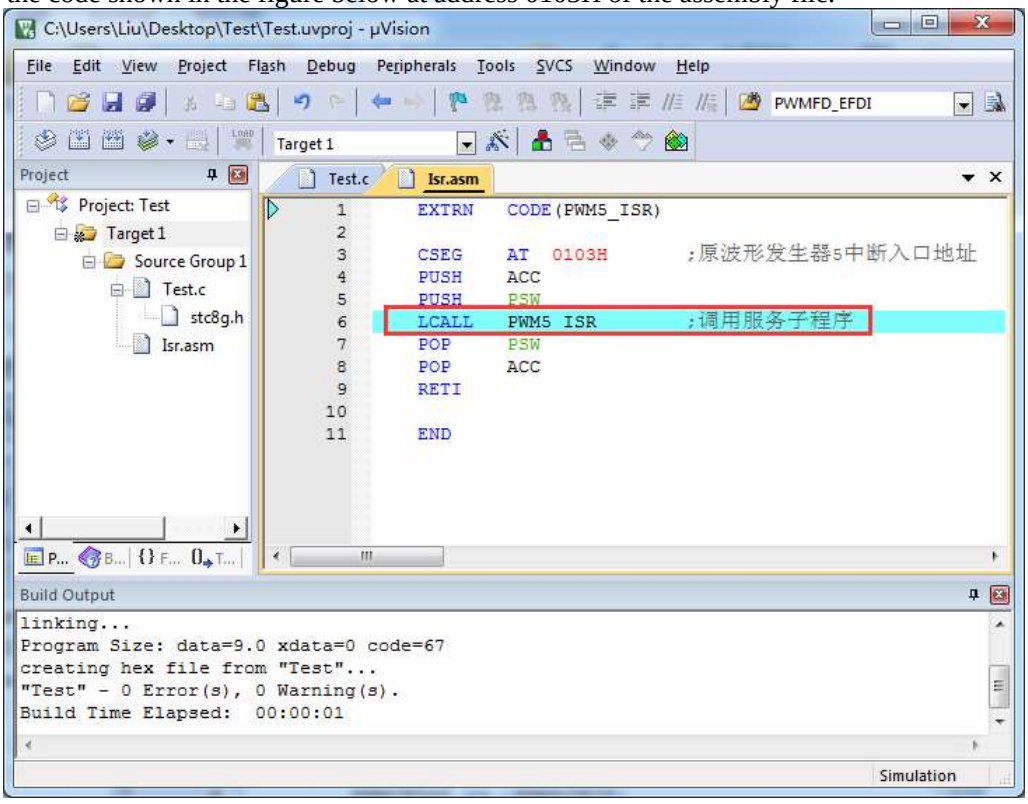
## Method 3: Define the interrupt service routine as a subroutine, and then use the LCALL instruction in the interrupt entry address in the assembly code to execute the service routine.

The steps are as follows:

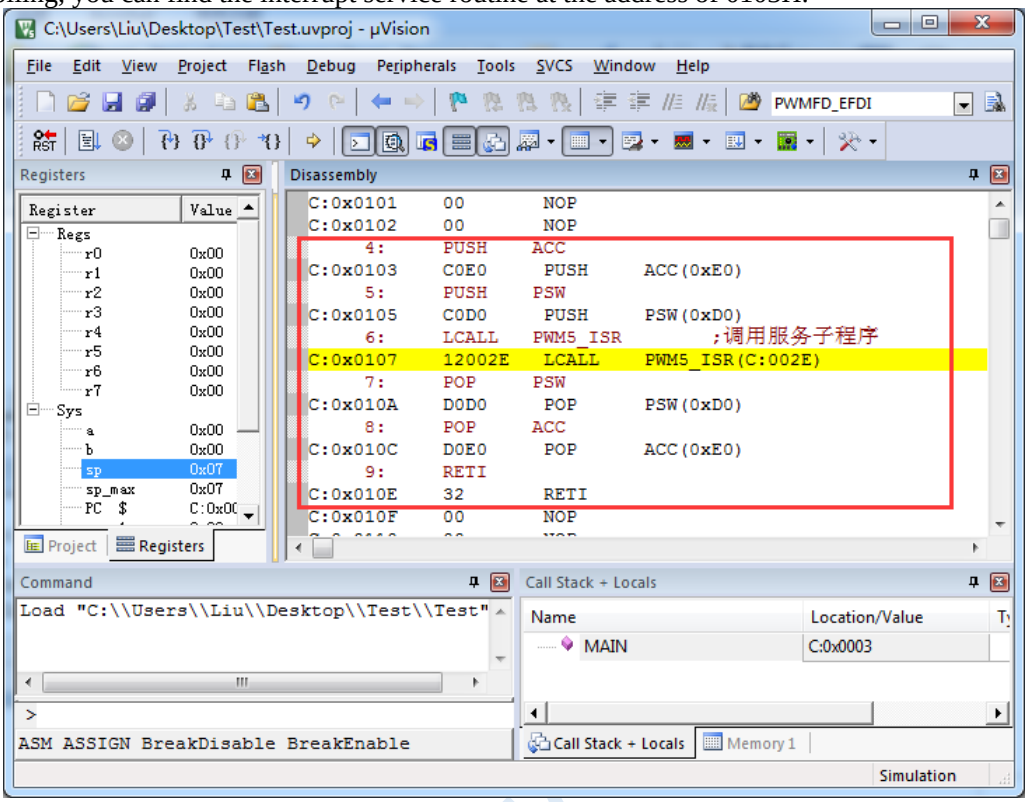
1. Remove the "interrupt" attribute from the interrupt service routine firstly and define it as an ordinary subroutine.




2. Then enter the code shown in the figure below at address 0103H of the assembly file.



3. After compiling, you can find the interrupt service routine at the address of 0103H.




This method does not need to remap interrupt entries. But there is a problem with this method. It requires the user to check the disassembly code of the C program to determine which registers need to be pushed onto the stack in the assembly file. PSW, ACC, B, DPL, DPH and R0 ~ R7 are included generally. In addition to the PSW must be pushed onto the stack, registers which are used in the user subroutine must be pushed onto the stack.

# Appendix S Electrical Characteristics

## S.1 Absolute Maximum Rating

| Parameters | Minimum | Maximum | UNIT | Description |
|---|---|---|---|---|
| Storage temperature | -55 | +125 | ℃ | |
| Operating temperature | -40 | +85 | ℃ | If the operating temperature is higher than 85℃ (such as around 125℃), due to the large temperature drift of the internal IRC clock frequency at high temperature, it is recommended to use an external high temperature clock or crystal oscillator. In addition, when the temperature is high, the frequency does not run fast, and it is recommended to use a working frequency below 24M; if the system must run at a higher temperature, please use an external high-reliability low-frequency active clock.<br>If the working temperature is around -55℃, the working voltage should not be too low. It is strongly recommended that the MCU-VCC voltage should not be lower than 3.0V. In addition, the rising speed of the power supply must also be as fast as possible, preferably at the millisecond level. |
| Operating Voltage | 1.9 | 5.5 | V | |
| VDD to ground voltage | -0.3 | +5.5 | V | |
| I/O port to ground voltage | -0.3 | VDD+0.3 | V | |

## S.2 DC ELECTRICAL CHARACTERISTICS (3.3V)

(VSS=0V, VDD=5.0V, test temperature =25℃)

| Symbol | Parameter | Limits | | | | Test Conditions |
|---|---|---|---|---|---|---|
| | | MIN | TYP | MAX | UNIT | |
| $I_{PD}$ | Power-down mode current | - | 0.4 | - | uA | |
| $I_{WKT}$ | Power-down Wake-up timer | - | 1.5 | - | uA | |
| $I_{LVD}$ | Low-voltage detection module | - | 10 | - | uA | |
| $I_{CMP}$ | Comparator power consumption | - | 90 | - | uA | |
| $I_{IDL}$ | Idle mode current (6MHz) | - | 0.88 | - | mA | |
| | Idle mode current (12MHz) | - | 1.0 | - | mA | |
| | Idle mode current (24MHz) | - | 1.16 | - | mA | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Idle mode current (internal 32KHz) | - | 0.48 | - | mA | |
| $I_{NOR}$ | Normal mode current（internal 32KHz） | - | 0.48 | - | mA | Equivalent to 0.5M of traditional 8051 |
| | Normal mode current（500KHz） | - | 0.88 | - | mA | Equivalent to 7M of traditional 8051 |
| | Normal mode current（600KHz） | - | 0.88 | - | mA | Equivalent to 8M of traditional 8051 |
| | Normal mode current（700KHz） | - | 0.90 | - | mA | Equivalent to 9M of traditional 8051 |
| | Normal mode current（800KHz） | - | 0.91 | - | mA | Equivalent to 11M of traditional 8051 |
| | Normal mode current（900KHz） | - | 0.91 | - | mA | Equivalent to 12M of traditional 8051 |
| | Normal mode current（1MHz） | - | 0.94 | - | mA | Equivalent to 13M of traditional 8051 |
| | Normal mode current（2MHz） | - | 1.05 | - | mA | Equivalent to 26M of traditional 8051 |
| | Normal mode current（3MHz） | - | 1.17 | - | mA | Equivalent to 40M of traditional 8051 |
| | Normal mode current（4MHz） | - | 1.26 | - | mA | Equivalent to 53M of traditional 8051 |
| | Normal mode current（5MHz） | - | 1.40 | - | mA | Equivalent to 66M of traditional 8051 |
| | Normal mode current（6MHz） | - | 1.49 | - | mA | Equivalent to 79M of traditional 8051 |
| | Normal mode current（12MHz） | - | 2.09 | - | mA | Equivalent to 158M of traditional 8051 |
| | Normal mode current（24MHz） | - | 3.16 | - | mA | Equivalent to 317M of traditional 8051 |
| $V_{IL1}$ | Input low voltage | - | - | 0.99 | V | Turn on Schmitt trigger |
| | | - | - | 1.07 | V | Turn off Schmitt trigger |
| $V_{IH1}$ | Input high voltage1(general I/O) | 1.18 | - | - | V | Turn on Schmitt trigger |
| | | 1.09 | - | - | V | Turn off Schmitt trigger |
| $V_{IH2}$ | Input high voltage2(RST pin) | 1.18 | - | 0.99 | V | |
| $I_{OL1}$ | Output low-level sink current | - | 20 | - | mA | Port voltage 0.45V |
| $I_{OH1}$ | Output high level current (bi-direction mode) | 200 | 270 | - | uA | |
| $I_{OH2}$ | Output high level current (Push-pull mode) | - | 20 | - | mA | Port voltage 2.4V |
| $I_{IL}$ | Logic 0 input current | - | - | 50 | uA | Port voltage 0V |
| $I_{TL}$ | Logical 1 to 0 transition current | 100 | 270 | 600 | uA | Port voltage 2.0V |
| $R_{PU}$ | I/O port pull-up resistor | 5.8 | 5.9 | 6.0 | KΩ | |
| I/O speed | I/O high current drive, I/O fast conversion | | 25 | | MHz | PxDR=0, PxSR=0 |
| | I/O high current drive, I/O fast conversion | | 22 | | MHz | PxDR=1, PxSR=0 |
| | I/O low current drive, I/O slow conversion | | 16 | | MHz | PxDR=0, PxSR=1 |
| | I/O low current drive, I/O slow | | 12 | | MHz | PxDR=1, |

| | | | | | | |
|---|---|---|---|---|---|---|
| | conversion | | | | | PxSR=1 |
| Comparators | Fastest speed | | 10 | | MHz | Turn off all analog and digital filtering |
| | Analog filter time | | 0.1 | | us | |
| | Digital filter time | | 0 | | system clock | LCDTY＝0 |
| | | | n+2 | | | LCDTY=n (n=1~63) |
| $I_{PD2}$ | Power-down mode power consumption when the comparator is enabled | - | 400 | - | uA | |
| $I_{PD3}$ | Power-down mode power consumption when LVD is enabled | - | 470 | - | uA | |

# S.3 DC ELECTRICAL CHARACTERISTICS (5V)

(VSS=0V, VDD=5.0V, test temperature =25℃)

| Symbol | Parameter | Limits | | | | Test Conditions |
|---|---|---|---|---|---|---|
| | | MIN | TYP | MAX | UNIT | |
| $I_{PD}$ | Power-down mode current | - | 0.6 | - | uA | |
| $I_{WKT}$ | Power-down Wake-up timer | - | 4.4 | - | uA | |
| $I_{LVD}$ | Low-voltage detection module | - | 30 | - | uA | |
| $I_{CMP}$ | Comparator power consumption | - | 90 | - | uA | |
| $I_{IDL}$ | Idle mode current (6MHz) | - | 0.58 | - | mA | |
| | Idle mode current (12MHz) | - | 0.98 | - | mA | |
| | Idle mode current (24MHz) | - | 1.10 | - | mA | |
| | Idle mode current (internal 32KHz) | - | 1.25 | - | mA | |
| $I_{NOR}$ | Normal mode current（internal 32KHz） | - | 0.58 | - | mA | Equivalent to 0.5M of traditional 8051 |
| | Normal mode current（500KHz） | | 0.97 | | mA | Equivalent to 7M of traditional 8051 |
| | Normal mode current（600KHz） | | 0.97 | | mA | Equivalent to 8M of traditional 8051 |
| | Normal mode current（700KHz） | | 1.00 | | mA | Equivalent to 9M of traditional 8051 |
| | Normal mode current（800KHz） | | 1.01 | | mA | Equivalent to 11M of traditional 8051 |
| | Normal mode current（900KHz） | | 1.01 | | mA | Equivalent to 12M of traditional 8051 |
| | Normal mode current（1MHz） | | 1.03 | | mA | Equivalent to 13M of traditional 8051 |
| | Normal mode current（2MHz） | | 1.15 | | mA | Equivalent to 26M of traditional 8051 |
| | Normal mode current（3MHz） | | 1.27 | | mA | Equivalent to 40M of traditional 8051 |
| | Normal mode current（4MHz） | | 1.35 | | mA | Equivalent to 53M of traditional 8051 |
| | Normal mode current（5MHz） | | 1.49 | | mA | Equivalent to 66M of traditional 8051 |
| | Normal mode current（6MHz） | - | 1.59 | - | mA | Equivalent to 79M of traditional 8051 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Normal mode current（12MHz） | - | 2.19 | - | mA | Equivalent to 158M of traditional 8051 |
| | Normal mode current（24MHz） | - | 3.27 | - | mA | Equivalent to 317M of traditional 8051 |
| $V_{IL1}$ | Input low voltage | - | - | 1.32 | V | Turn on Schmitt trigger |
| | | - | - | 1.48 | V | Turn off Schmitt trigger |
| $V_{IH1}$ | Input high voltage1(general I/O) | 1.60 | - | - | V | Turn on Schmitt trigger |
| | | 1.54 | - | - | V | Turn off Schmitt trigger |
| $V_{IH2}$ | Input high voltage2(RST pin) | 1.60 | - | 1.32 | V | |
| $I_{OL1}$ | Output low-level sink current | - | 20 | - | mA | Port voltage 0.45V |
| $I_{OH1}$ | Output high level current (bi-direction mode) | 200 | 270 | - | uA | |
| $I_{OH2}$ | Output high level current (Push-pull mode) | - | 20 | - | mA | Port voltage 2.4V |
| $I_{IL}$ | Logic 0 input current | - | - | 50 | uA | Port voltage 0V |
| $I_{TL}$ | Logical 1 to 0 transition current | 100 | 270 | 600 | uA | Port voltage 2.0V |
| $R_{PU}$ | I/O port pull-up resistor | 4.1 | 4.2 | 4.4 | KΩ | |
| I/O speed | I/O high current drive, I/O fast conversion | | 36 | | MHz | PxDR=0, PxSR=0 |
| | I/O high current drive, I/O fast conversion | | 32 | | MHz | PxDR=1, PxSR=0 |
| | I/O low current drive, I/O slow conversion | | 26 | | MHz | PxDR=0, PxSR=1 |
| | I/O low current drive, I/O slow conversion | | 22 | | MHz | PxDR=1, PxSR=1 |
| Comparators | Fastest speed | | 10 | | MHz | Turn off all analog and digital filtering |
| | Analog filter time | | 0.1 | | us | |
| | Digital filter time | | 0 | | system clock | LCDTY＝0 |
| | | | n+2 | | | LCDTY=n (n=1~63) |
| $I_{PD2}$ | Power-down mode power consumption when the comparator is enabled | - | 460 | - | uA | |
| $I_{PD3}$ | Power-down mode power consumption when LVD is enabled | - | 520 | - | uA | |

# S.4 Internal IRC temperature drift characteristic (reference temperature 25℃)

| Temperature | Range | | |
|---|---|---|---|
| | MIN | TYP | MAX |

| | | | |
|---|---|---|---|
| -40℃～85℃ | | −1.38%～+1.42% | |
| -20℃～65℃ | | −0.88%～+1.05% | |

## S.5 Low voltage reset threshold voltage (test temperature 25 ℃)

| Level | Voltage | | |
|---|---|---|---|
| | MIN | TYP | MAX |
| POR | | （1.69V~1.82V） | |
| LVR0 | | 2.0V（1.88V~1.99V） | |
| LVR1 | | 2.4V（2.28V~2.45V） | |
| LVR2 | | 2.7V（2.58V~2.76V） | |
| LVR3 | | 3.0V（2.86V~3.06V） | |

# Appendix T Application Considerations

## T.1 Notes on STC8H series IO ports

1. For the IO ports of STC8H series chips, except for the ISP download ports P3.0 and P3.1, the initial modes of the remaining IO ports after power-on are high-impedance input mode, and the user cannot directly output the level, so the user initializes the program It is necessary to use the PxM0 and PxM1 registers to initialize the corresponding IO mode in order to be used normally.
2. All I/O ports of STC8H series chips can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, each I/O can independently enable internal 4K Pull resistance
3. STC8H series chips will not automatically set the IO port mode for special IO, such as ADC port, serial port, I2C port, and SPI port. The user must set the corresponding port to the appropriate mode.
4. If the enable P5.4 pin is a reset pin, the reset level is low
5. Special attention: Since all I/Os of STC8H series (except ISP download port P3.0/P3.1) are in high-impedance input mode after power-on, the external level of I/O is not fixed. The MCU directly enters the power-down mode/stop mode, which will cause extra power consumption of the I/O. Before the MCU enters the power-down mode/stop mode, all I/O ports must be set according to the actual situation. mode, all unused external floating I/Os need to be set as quasi-bidirectional ports, and output a high level fixed. Especially for chips with some pins, since some I/O ports inside the chip are not wired to external pins, these I/Os are also in a floating state, and these I/Os also need to be set as quasi-bidirectional ports. And fixed output high level.

## T.2 Notes on STC8H2K64T series

1. STC8H2K64T series A version of the chip, when setting the P0, P1, P5 port I/O port interrupt enable bit register P0INTE,P1INTE and P5INTE will affect the digital input enable registers P0IE, P1IE, P5IE of P0, P1, and P5.
2. When testing the ADC function of the A version of the STC8H2K64T series chip, do not set the P0IE, P1IE and P5IE registers, because the settings are invalid.
3. The RTC function of version A chips of STC8H2K64T series has yet to be improved, please do not use
4. Special attention: Since all I/Os of STC8H series (except ISP download port P3.0/P3.1) are in high-impedance input mode after power-on, the external level of I/O is not fixed. The MCU directly enters the power-down mode/stop mode, which will cause extra power consumption of the I/O. Before the MCU enters the power-down mode/stop mode, all I/O ports must be set according to the actual situation. mode, all unused external floating I/Os need to be set as quasi-bidirectional ports, and output a high level fixed. Especially for chips with some pins, since some I/O ports inside the chip are not wired to external pins, these I/Os are also in a floating state, and these I/Os also need to be set as quasi-bidirectional ports. And fixed output high level.

## T.3 Notes on STC8H4K64TLR series

1. When the clock source of the RTC selects an external 32.768K crystal oscillator, it is necessary to turn off the digital channel of the crystal oscillator connected to pins P1.6 and P1.7, otherwise there will be additional leakage after entering the STOP mode. (Set both bits 6 and 7 of register P1IE to 0 to close the digital channels of P1.6 and P1.7)

## T.4 Notes on STC8H8K64U series

1. For the B version chip of STC8H8K64U, the RTC cannot use the internal 32K as the clock source.
2. For the B version chip of STC8H8K64U, the comparator cannot select the ADC channel as the positive

input.

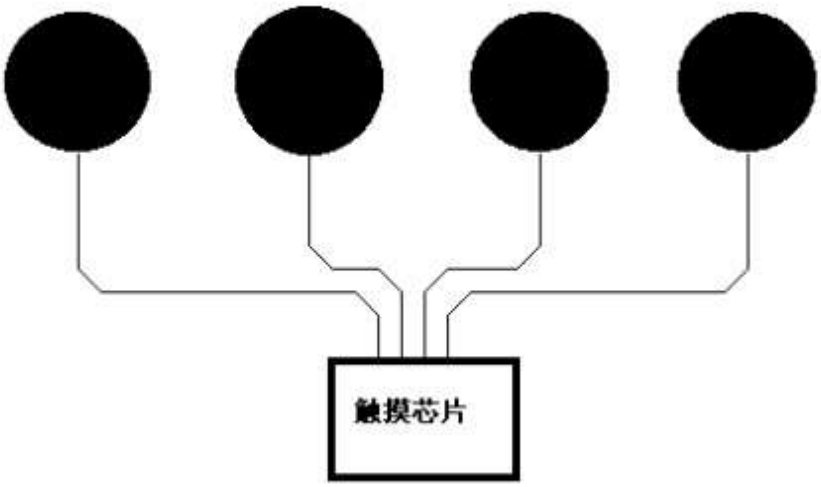
3. The B version chip of STC8H8K64U will consume about 3uA after entering the power saving mode.

For the above 3 problems, the C version of STC8H8K64U will all be modified correctly

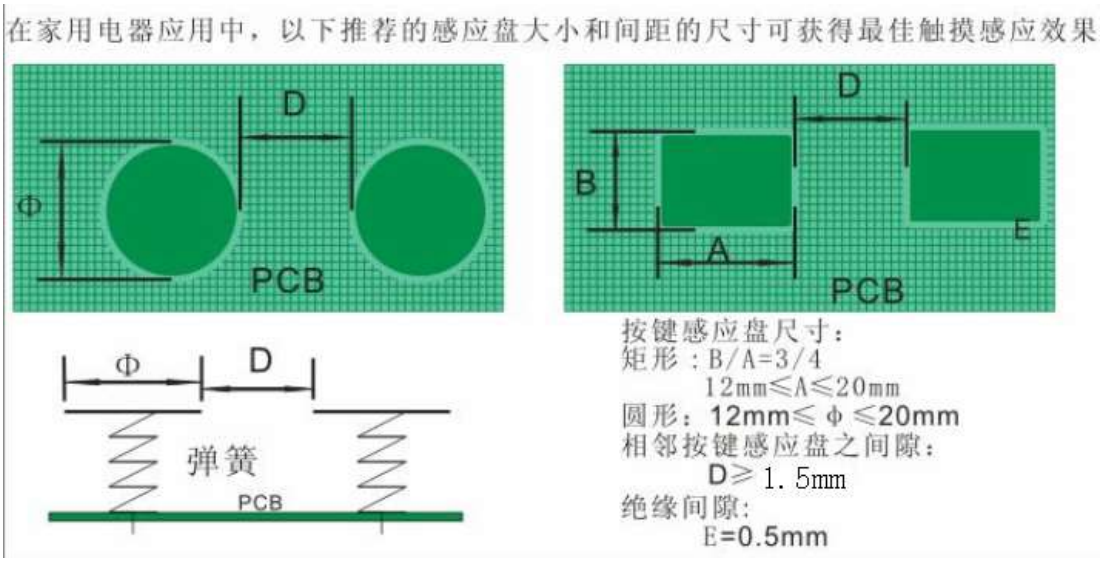
# Appendix U PCB design guidance for touch keys

The touch key has strict requirements on PCB design, otherwise its effect will be greatly reduced or even fail.It is recommended that users follow the following principles when designing PCB:

1. **Follow the basic principles of common digital-analog hybrid circuit design.**
   The capacitive touch button module integrates an analog circuit for precise capacitance measurement, so it should be treated as an independent analog circuit when designing the PCB. Follow the basic principles of common digital-analog hybrid circuit design.
2. **Use star grounding**
   The ground wire of the touch chip should not be shared with other circuits. It should be connected to the ground point of the board's power input separately, which is usually called "star grounding".
3. **The impact of noise generated on the power supply on the touch chip**
   The power ripple and noise should be as small as possible. It is best to use an independent trace to take power from the power supply point of the board and add filtering measures. Do not share the power circuit with other circuits.
4. **The connection between the IC and the sensing board should be as long as possible, so that it has an approximate distributed capacitance, as shown in the figure below.**




5. **The size and gap of the key induction plate (capacitive sensor)**
   In the case of meeting the aesthetic design requirements of the panel, the optimal touch sensing effect must be obtained through a reasonable arrangement of the size of the sensing plate and the interval size. The induction plate is placed on the bottom layer, and the IC is also placed on the bottom layer. There should be no vias between the induction plate and the IC. The distance between the edges of adjacent sensing plates is preferably at least 1.5mm (dimension D in the figure below). If the PCB area allows, try to use a larger distance. The distance between the copper paving and the induction plate is 0.5mm (dimension E in the figure below).




**6. Copper plating**

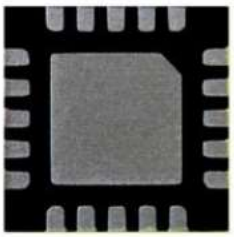
The bottom layer can be covered with grid copper or solid copper. Note that the distance between the copper and the induction plate is 0.5mm. The silk screen information of the top layer is printed on the button. The frame shape of the silk screen is the same as the bottom sensor plate. The top layer corresponding to the bottom sensor plate should not be covered with copper, otherwise the touch action will be shielded. The copper on the top layer is the same as the copper on the bottom layer.

**7. Wiring processing**

It is better to use a smaller line width for the connection between the sensor plate and the IC, such as between 10 and 15 mils. The connection between the induction plate and the touch chip should not cross the lines with strong interference, high frequency, and high current. Do not run other signal lines within 1.5mm of the connection between the sensor pad and the touch chip, the farther away the better. The top layer corresponds to the bottom sensor plate and connecting line, it is best not to put any line.

# Appendix V QFN/DFN packaged components welding method

In the package form of STC products, the more popular QFN and DFN packages have been added. Since the pins of the chip in this package are at the bottom of the chip, manual soldering is difficult. There are small companies on the market that specialize in welding engineering samples, which can undertake engineering sample proofing. If users need to weld by themselves, please refer to the following welding method.



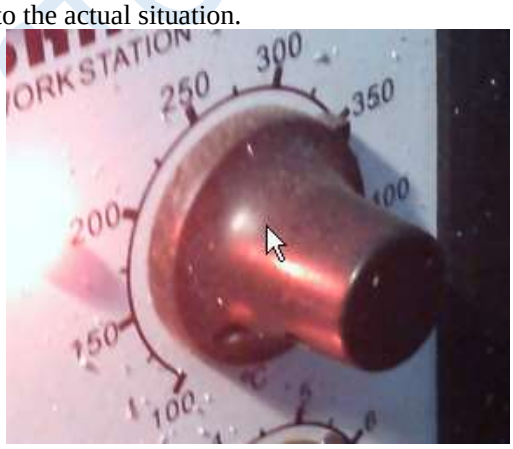
1. Firstly, you need to prepare the following tools: electric soldering iron, hot air gun, tweezers, fixing frame and other tools

2. The PCB boards and chips that need to be soldered are as follows:

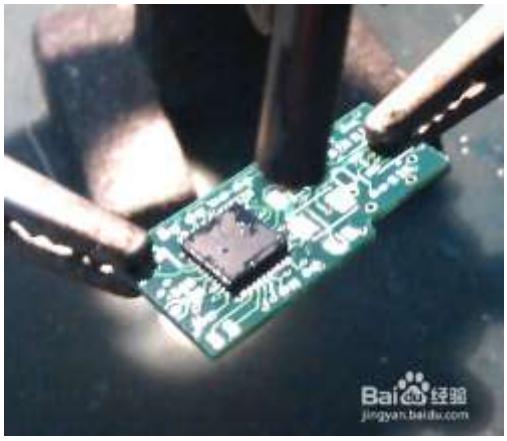
3. Tin the pads of the chip on the board:


4. Then apply tin to the bottom of the chip. After the tin is applied, it should be flattened to reduce the tin as much as possible, but it cannot be eliminated.

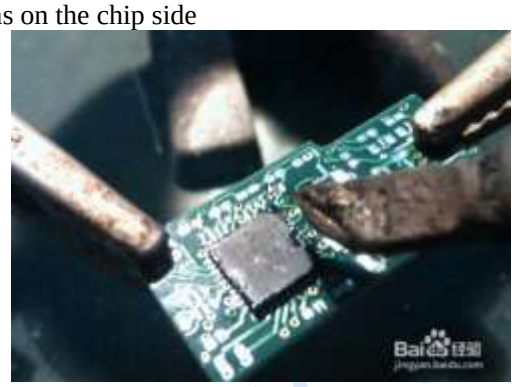
5.Adjust the temperature of the hot air gun, the actual air output is about 240 degrees, because the quality of the air gun is different, adjust it according to the actual situation.



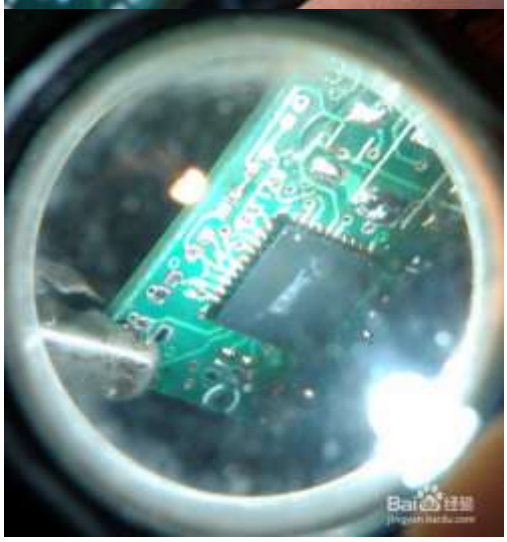

6. Put the chip on the pad, be sure to place it straight, and then blow it with a hot air gun at a uniform speed until the tin melts, usually within 20 seconds.



7. Use a soldering iron to tin the pins on the chip side



8. The effect after welding



# Appendix W About whether to bake before reflow soldering

According to the requirements of the International Moisture Sensitivity Level 3 (MSL3) specification, the SMD components must be reflowed within 168 hours and 7 days after the vacuum packaging is unpacked. If not, they must be baked again at high temperature.

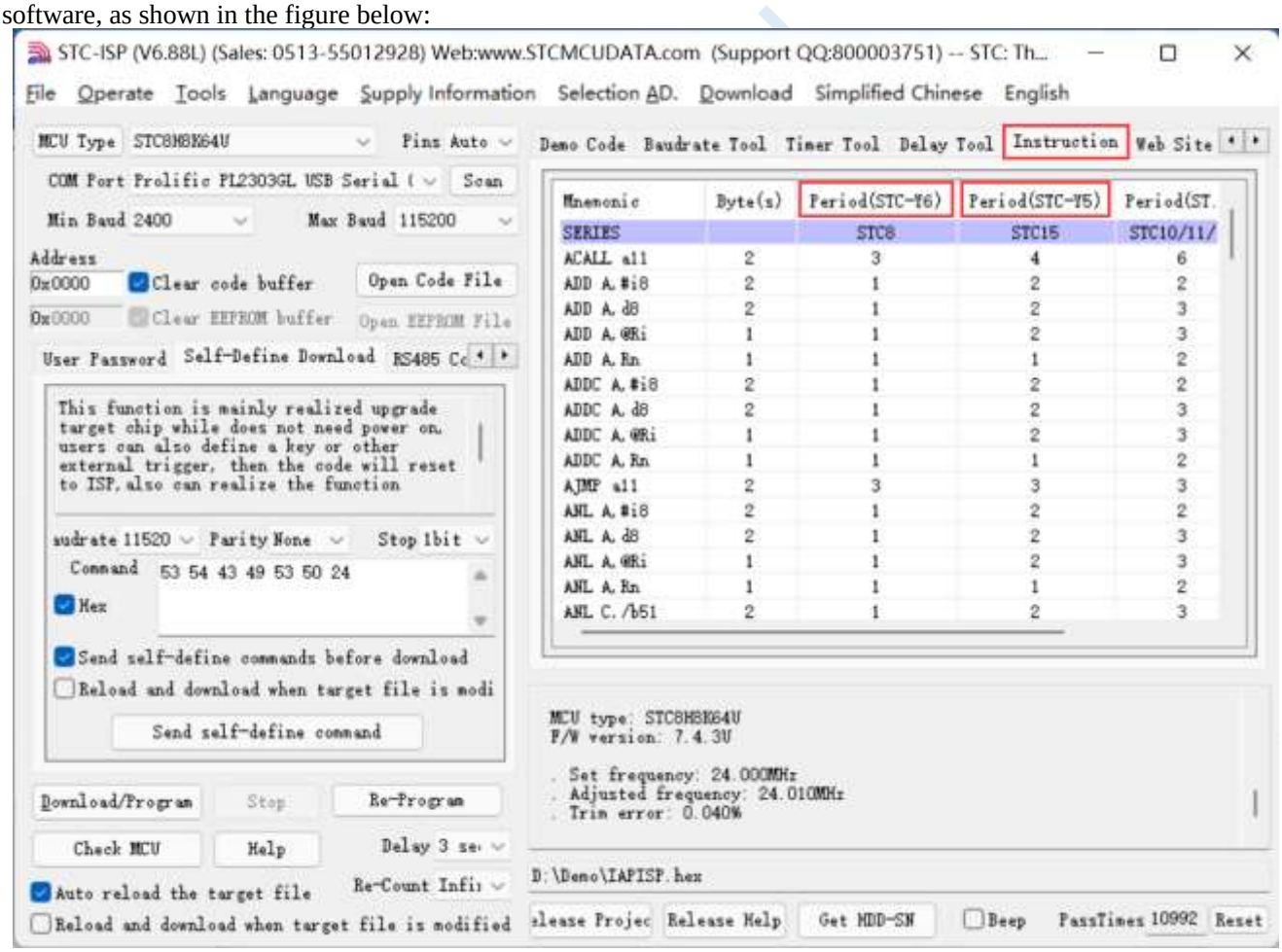
SOP/TSSOP plastic tubes cannot withstand high temperatures above 100 degrees, and must be reflow soldered within 7 days after unpacking. Otherwise, remove the plastic tubes that cannot withstand high temperatures above 100 degrees before reflow soldering, put them in metal trays, and re-bake them. Baking: 110～125℃, 4～8 hours

LQFP/QFN/DFN trays can withstand high temperatures above 100 degrees, and must be reflowed within 7 days after unpacking.

# Appendix X Precautions for STC8H series MCU to replace STC15 series

## ■ MCU instructions

The instruction code of STC8H series is completely consistent with STC15 series, so the code of STC15 series is transplanted to STC8H,The operation is still correct, but the instruction speed of the STC8H series is faster than that of the STC15 series. The instruction system of the STC15 series belongs to the STC-Y5 series of instructions, while the instructions of the STC8H series belong to the STC-Y6 series of instructions. Most of the instructions of the STC-Y6 series are executed. Only one CPU clock is required. If there is a command delay code in the user code, it needs to be adjusted. For the comparison of each instruction, please refer to the instruction table of the STC download software, as shown in the figure below:




## ■ I/O port

After the STC8H series MCU is powered on, the I/O mode is different from that of the STC15 series. All I/O ports of STC15 series single-chip microcomputers are in 8051 quasi-bidirectional port mode after power-on. In the I/O of STC8H series single-chip microcomputers, except for ISP download pins P3.0/P3.1 which are quasi-bidirectional port modes, all the other I/O ports are in high-impedance input mode after power on. The traditional

8051 and STC 15 series single-chip microcomputers are in quasi-bidirectional port mode and output high level after power-on. Often customers use I/O to drive motors or LED lights in their systems, so there will be moments when the single-chip microcomputer is powered on. Move it once or the LED will flash once. The I/O of the STC8H series is in high-impedance input mode after power-on, which can avoid this kind of malfunction of the motor and LED.

Because in the I/O of STC8H series single-chip microcomputer, except ISP download pin P3.0/P3.1 which is quasi-bidirectional port mode, all other I/O ports are in high-impedance input mode after power-on, so when users need Before the I/O ports of STC8H series output signals, the two registers PxM0 and PxM1 must be used to set the I/O working mode.

## ■ Reset foot

The P5.4 port of the STC8H series and STC15 series is generally used as a normal I/O port. When the user sets P5.4 as the reset pin function during ISP download, the P5.4 port is the reset of the microcontroller Pin (RESET pin). For the STC15H series, when the reset pin is high, the microcontroller is in the reset state, and when the reset pin is low, the microcontroller is released from the reset state. The reset levels of STC8 series and STC15H series are reversed, that is, for STC8H series, when the reset pin is low, the microcontroller is in the reset state, and when the reset pin is high, the microcontroller is released from the reset state.

Therefore, when the user enables the reset pin function of port P5.4, it is necessary to pay attention to the reset level.

## ■ ADC

The addresses of the ADC_CONTR, ADC_RES and ADC_RESL 3 registers of STC8H series and STC15 series are the same.But the STC8H series adds two new registers: ADCCFG and ADCTIM.

STC15 series start ADC conversion bit ADC_START is located at BIT3 of register ADC_CONTR, while STC8H series is located at BIT6 of ADC_CONTR

The STC15 series ADC conversion complete flag ADC_FLAG is located at BIT4 of the register ADC_CONTR, while the STC8H series is located at BIT5 of ADC_CONTR

STC15 series ADC speed control is ADC_SPEED located at BIT6-BIT5 of register ADC_CONTR, and STC8H series is located at BIT3-BIT0 of ADCCFG

The alignment control bit ADRJ of the STC15 series ADC conversion result is located at BIT5 of the register CLK_DIV, while the alignment control bit RESFMT of the STC8H series is located at BIT5 of ADCCFG

The STC8H series adds a more precise ADC conversion timing control mechanism, which can be set through the register ADCTIM

## ■ EEPROM

The waiting time for EEPROM erasing and programming of STC15 series is set by Bit2-Bit0 of the register IAP_CONTR. The setting is only an approximate frequency range value. The STC8H series adds a new register IAP_TPS (SFR address: 0F5H), dedicated to setting EEPROM erasing In addition to the waiting time for programming, and the user does not need to calculate, just fill in IAP_TPS directly according to the current CPU working frequency, and the hardware will automatically calculate the waiting time. (For example: the current CPU operating frequency is 24MHz, you only need to fill in 24 to IAP_TPS).

# Appendix Y Precautions for STC8H series single-chip microcomputer to replace STC8A/8F series

## ■ I/O port

After the STC8H series MCU is powered on, the I/O mode is different from that of the STC8A/8F series. All I/O ports of STC8A/8F series single-chip microcomputers are in 8051 quasi-bidirectional port mode after power-on. In the I/O of STC8H series single-chip microcomputers, except for ISP download pins P3.0/P3.1 which are quasi-bidirectional port modes, the rest All of the I/O ports are in high impedance input mode after power on. The traditional 8051 and STC 15/8A/8F series single-chip microcomputers are in quasi-bidirectional port mode and output high level after power-on. Often customers' systems use I/O to drive motors or LED lights, so the single-chip microcomputer will be powered on. The motor will move or the LED will flash. The I/O of the STC8H series is in high-impedance input mode after power-on, which can avoid this kind of malfunction of the motor and LED.

Because in the I/O of STC8H series single-chip microcomputer, except ISP download pin P3.0/P3.1 which is quasi-bidirectional port mode, all other I/O ports are in high-impedance input mode after power-on, so when users need Before the I/O ports of STC8H series output signals, the two registers PxM0 and PxM1 must be used to set the I/O working mode.

## ■ Reset foot

The P5.4 port of the STC8H series and STC8A/8F series is generally used as a normal I/O port. When the user sets P5.4 as the reset pin function during ISP download, the P5.4 port is a microcontroller The reset pin (RESET pin). For the STC8A/8F series, when the reset pin is high, the microcontroller is in the reset state, and when the reset pin is low, the microcontroller is released from the reset state. The reset levels of STC8H series and STC8A/8F series are reversed, that is, for STC8H series, when the reset pin is low, the microcontroller is in the reset state, and when the reset pin is high, the microcontroller is released from the reset state.

Therefore, when the user enables the reset pin function of port P5.4, it is necessary to pay attention to the reset level.

## ■ EEPROM

The waiting time for erasing and programming of EEPROM of STC8A/8F series is set by Bit2-Bit0 of the register IAP_CONTR. The setting is only a rough frequency range value. The STC8H series adds a new register IAP_TPS (SFR address: 0F5H), dedicated to setting The waiting time of EEPROM erasing and programming, and the user does not need to calculate, just fill in IAP_TPS directly according to the current CPU operating frequency, and the hardware will automatically calculate the waiting time. (For example: the current CPU operating frequency is 24MHz, you only need to fill in 24 to IAP_TPS)

# Appendix Z Internally tested models

## Z.1 STC8H2K64T-35I-LQFP48/QFN48

### Z.1.1 Features and Price

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

| MCU | Operating voltage (V) | Flash Code Memory (100 thousand times) (Byte) | idata Internal DATA RAM(Byte) | xdata Internal extended SRAM (Byte) | Enhanced Dual DPTR increasing or decreasing | EEPROM 100 thousand times) (Byte) | Maximum I/O Lines | Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU) | All I/O ports support interrupts and can wake up MCU | UARTs which can wake-up CPU | SPI which can wake-up CPU | I²C which can wake-up CPU | Touch key | LED driver | RTC | MDU16 (Hardware 16-bit Multiplier and Divider) | Timers/Counters (T0-T4 Pin can wake-up CPU) | 16-bit advanced PWM timer with Complementary symmetrical dead-time | Power-down Wake-up timer | 15 channels high speed ADC (8 PWMs can be used as 8 DACs) | Comparator (May be used as ADC to detect external power-down) | Internal LVD interrupt (can wake-up CPU) | Watch-dog Timer | Internal high reliable reset circuit with 4 levels optional reset threshold voltage | Internal high precision Clock (adjustbal under 36MHz) | Clock output and Reset | Program encrypted transmission (Anti-blocking) | Password can be set for next update | Support RS485 download | Support software USB download directly | Online debug itself | Price & Package: LQFP48 <9mm*9mm> | Price & Package: QFN48 <6mm*6mm> | products supply information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STC8H2K32T | 1.9-5.5 | 32 K | 256 | 2K | 2 | 32K | 44 | Y | Y | 4 | Y | Y | Y | Y | Y | Y | 5 | 8 | Y | 12-bit | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Some available |
| STC8H2K60T | 1.9-5.5 | 60 K | 256 | 2K | 2 | 4K | 44 | Y | Y | 4 | Y | Y | Y | Y | Y | Y | 5 | 8 | Y | 12-bit | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | |
| STC8H2K64T | 1.9-5.5 | 64 K | 256 | 2K | 2 | IAP | 44 | Y | Y | 4 | Y | Y | Y | Y | Y | Y | 5 | 8 | Y | 12-bit | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | |

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 29 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported
- **Operating voltage**
  - ✓ 1.9V～5.5V
- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)
- **Flash memory**

- ✓ Up to 64Kbytes of Flash memory to be used for storing user code
- ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
- ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
- ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

➢ **SRAM**
- ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 2048 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

➢ **Clock**
- ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 36MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz～33MHz) and external clock
  Users can freely choose the above 3 clock sources

➢ **Reset**
- ✓ Hardware reset
  - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)
    The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
  - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
  - ✓ Watch dog timer reset
  - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
- ✓ Software reset
  - ✓ Writing the reset trigger register using software

➢ **Interrupts**
- ✓ 29 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, $I^2C$, comparator, PWMA, PWMB, RTC, TKS, EXP0, EXP1, EXP2, EXP3, EXP4, EXP5.
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2 (P1.2), T3 (P0.4), T4 (P0.6), RXD (P3.0/P3.6/P1.6/P4.3), RXD2 (P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3) and comparator interrupt, low voltage detection interrupt, power-down wake-up timer.

➢ **Digital peripherals**
- ✓ 5 16-bit timers: timer0, timer1, timer2, timer 3, timer 4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ $I^2C$: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt.

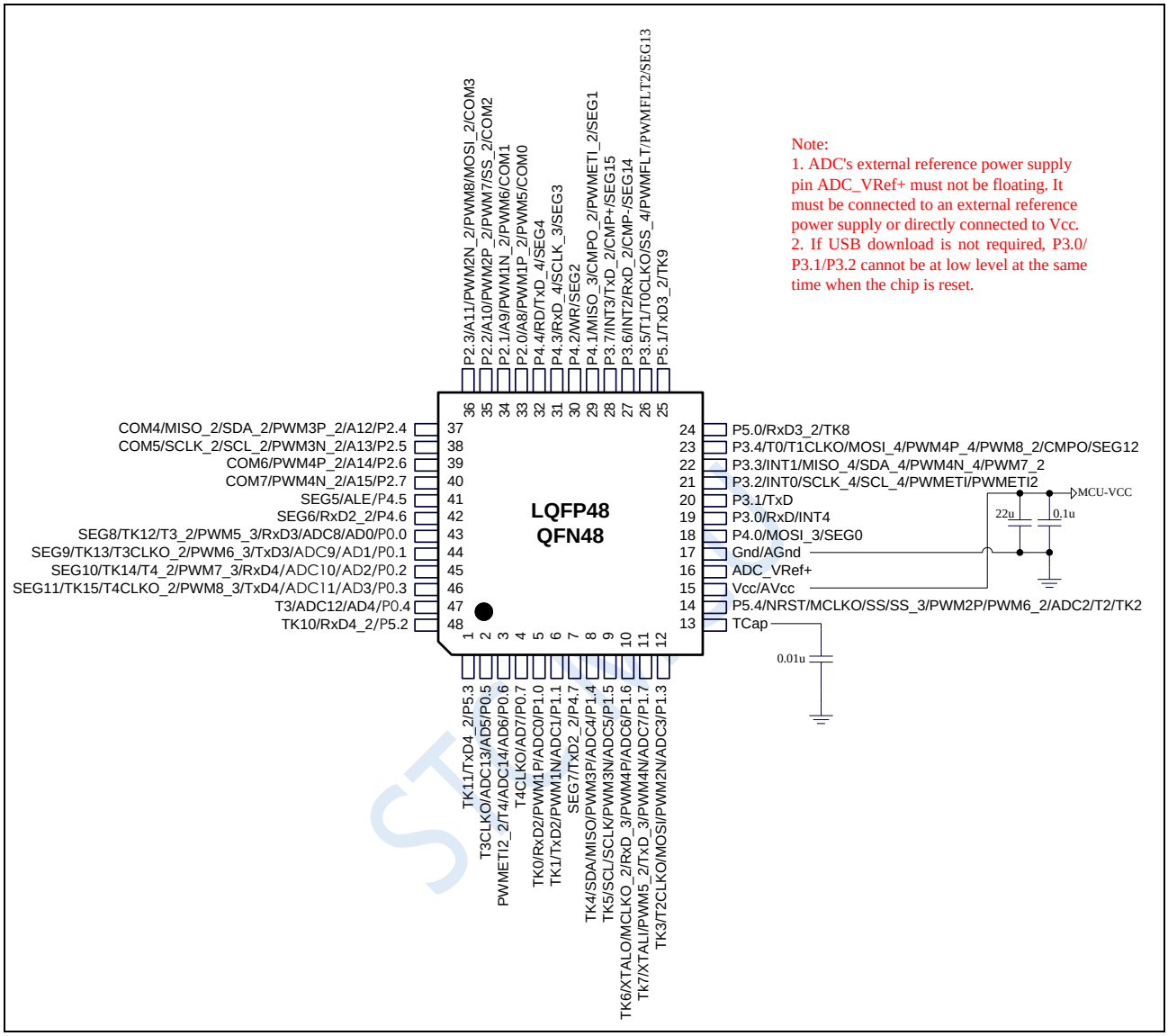
➢ **Analog peripherals**
- ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital

conversion. The maximum speed can be 800K(800K ADC conversions per second)
- ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
- ✓ Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
- ✓ Touch key: The microcontroller supports up to 16 touch keys. Every touch key can be enabled independently. The internal reference voltage is adjustable with 4 levels. Charge and discharge time settings and internal working frequency settings are flexible. The touch key supports wake-up CPU from low-power mode.
- ✓ LED driver: can drive up to 256 (8*16*2) LEDs; support common cathode mode, common anode mode and common cathode/common anode mode; support 8-level grayscale adjustment (brightness adjustment)
- ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

➢ **GPIO**
- ✓ Up to 44 GPIOs: P0.0~P0.7, P1.0~ P1.7 (No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

➢ **Package**
- ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

# Z.1.2 Pinouts


Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.
LQFP48
QFN48
1 P5.3/TK11/TxD4_2
2 P0.5/T3CLKO/ADC13/AD5
3 P0.6/PWMETI2_2/T4/ADC14/AD6
4 P0.7/T4CLKO/AD7
5 P1.0/TK0/RxD2/PWM1P/ADC0
6 P1.1/TK1/TxD2/PWM1N/ADC1
7 P4.7/SEG7/TxD2_2
8 P1.4/TK4/SDA/MISO/PWM3P/ADC4
9 P1.5/TK5/SCL/SCLK/PWM3N/ADC5
10 P1.6/TK6/XTALO/MCLKO_2/RxD_3/PWM4P/ADC6
11 P1.7/Tk7/XTALI/PWM5_2/TxD_3/PWM4N/ADC7
12 P1.3/TK3/T2CLKO/MOSI/PWM2N/ADC3
13 TCap
14 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2
15 Vcc/AVcc
16 ADC_VRef+
17 Gnd/AGnd
18 P4.0/MOSI_3/SEG0
19 P3.0/RxD/INT4
20 P3.1/TxD
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO/SEG12
24 P5.0/RxD3_2/TK8
25 P5.1/TxD3_2/TK9
26 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/SEG13
27 P3.6/INT2/RxD_2/CMP-/SEG14
28 P3.7/INT3/TxD_2/CMP+/SEG15
29 P4.1/MISO_3/CMPO_2/PWMETI_2/SEG1
30 P4.2/WR/SEG2
31 P4.3/RxD_4/SCLK_3/SEG3
32 P4.4/RD/TxD_4/SEG4
33 P2.0/A8/PWM1P_2/PWM5/COM0
34 P2.1/A9/PWM1N_2/PWM6/COM1
35 P2.2/A10/PWM2P_2/PWM7/SS_2/COM2
36 P2.3/A11/PWM2N_2/PWM8/MOSI_2/COM3
37 P2.4/COM4/MISO_2/SDA_2/PWM3P_2/A12
38 P2.5/COM5/SCLK_2/SCL_2/PWM3N_2/A13
39 P2.6/COM6/PWM4P_2/A14
40 P2.7/COM7/PWM4N_2/A15
41 P4.5/SEG5/ALE
42 P4.6/SEG6/RxD2_2
43 P0.0/SEG8/TK12/T3_2/PWM5_3/RxD3/ADC8/AD0
44 P0.1/SEG9/TK13/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1
45 P0.2/SEG10/TK14/T4_2/PWM7_3/RxD4/ADC10/AD2
46 P0.3/SEG11/TK15/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3
47 P0.4/T3/ADC12/AD4
48 P5.2/TK10/RxD4_2
MCU-VCC
22u
0.1u
0.01u

# Z.2 STC8H4K64LCD-45I-LQFP64/QFN64/LQFP48/QFN48

## Z.2.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

| MCU | Operating voltage（V） | Flash Code Memory (100 thousand times) (Byte) | idata Internal DATA RAM(Byte) | xdata Internal extended SRAM (Byte) | Enhanced Dual DPTR increasing or decreasing | EEPROM 100 thousand times) (Byte) | Maximum I/O Lines | Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU) | All I/O ports support interrupts and can wake up MCU | DMA UARTs which can wake-up CPU | DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit) | LCD driver (4COM*40SEG) | Touch key | RTC | DMA SPI which can wake-up CPU | I²C which can wake-up CPU (No DMA) | MDU16 (Hardware 16-bit Multiplier and Divider) | Timers/Counters (T0-T4 Pin can wake-up CPU) | 16-bit advanced PWM timer with Complementary symmetrical dead-time | Power-down Wake-up timer | DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs) | Comparator (May be used as ADC to detect external power-down) | Internal LVD interrupt (can wake-up CPU) | Watch-dog Timer | Internal high reliable reset circuit with 4 levels optional reset threshold voltage | Internal high presision Clock (adjustbal under 45MHz) | Clock output and Reset | Program encrypted transmission (Anti-blocking) | Password can be set for next update | Support RS485 download | Support software USB download directly | Online debug itself | Price & Package | | | | products supply information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | LQFP64 <12mm*12mm> | QFN64 <8mm*8mm> | LQFP48 <9mm*9mm> | QFN48 <6mm*6mm> | |
| STC8H4K32TLCD | 1.9-5.5 | 32K | 256 | 4K | 2 | 32K | 60 | Y | Y | 4 | Y | Y | Y | Y | Y | Y | Y | 5 | 8 | Y | 12bit | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | √ | | | | Samples |
| STC8H4K48TLCD | 1.9-5.5 | 48K | 256 | 4K | 2 | 16K | 60 | Y | Y | 4 | Y | Y | Y | Y | Y | Y | Y | 5 | 8 | Y | 12bit | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | √ | | | | |
| STC8H4K64TLCD | 1.9-5.5 | 64K | 256 | 4K | 2 | IAP | 60 | Y | Y | 4 | Y | Y | Y | Y | Y | Y | Y | 5 | 8 | Y | 12bit | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | √ | | | | |

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 42 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported
- **Operating voltage**
  - ✓ 1.9V～5.5V
- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)
- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used to store user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
  - ✓ 4096 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- **Clock**
  - ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - ✓ Error:±0.3% (at the temperature 25℃)
    - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
    - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - ✓ Internal 32KHz low speed IRC with large error
  - ✓ External crystal (4MHz～33MHz) and external clock

Users can freely choose the above 3 clock sources

- **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)
      The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
    - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - ✓ Watch dog timer reset
    - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
  - ✓ Software reset
    - ✓ Writing the reset trigger register using software
- **Interrupts**
  - ✓ 42 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, $I^2C$, comparator, PWMA, PWMB, RTC, TKS, P1, P2, P3, P4, P5, P6, P7, LCM driver, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCM driver and DMA interrupt of memory-to-memory.
  - ✓ 4 interrupt priority levels
  - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.
- **Digital peripherals**
  - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
  - ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
  - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - ✓ $I^2C$: Master mode or slave mode are supported.

- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM
- ✓ LCM (TFT color screen) dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
  - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and control lines, plus 2 common I/Os to control chip selection and reset (many TFT color screen chip selections and reset manufacturer has carried out automatic processing, does not need software control)
- ✓ LCD dirver: support up to 4COM*40 SEGs and 8 levels grayscale adjustment

➢ **Analog peripherals**

- ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)
- ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
- ✓ Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
- ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

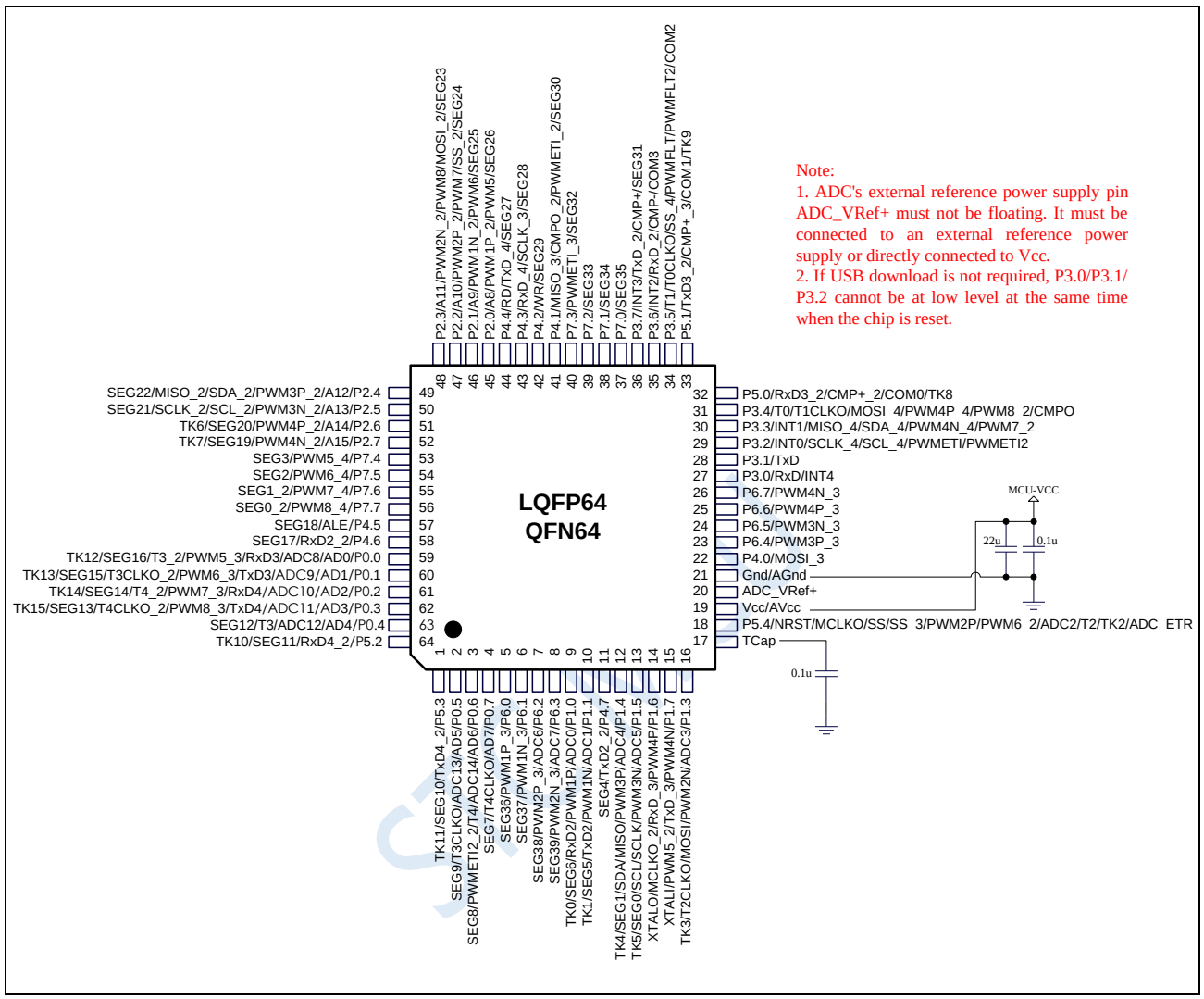
➢ **GPIO**

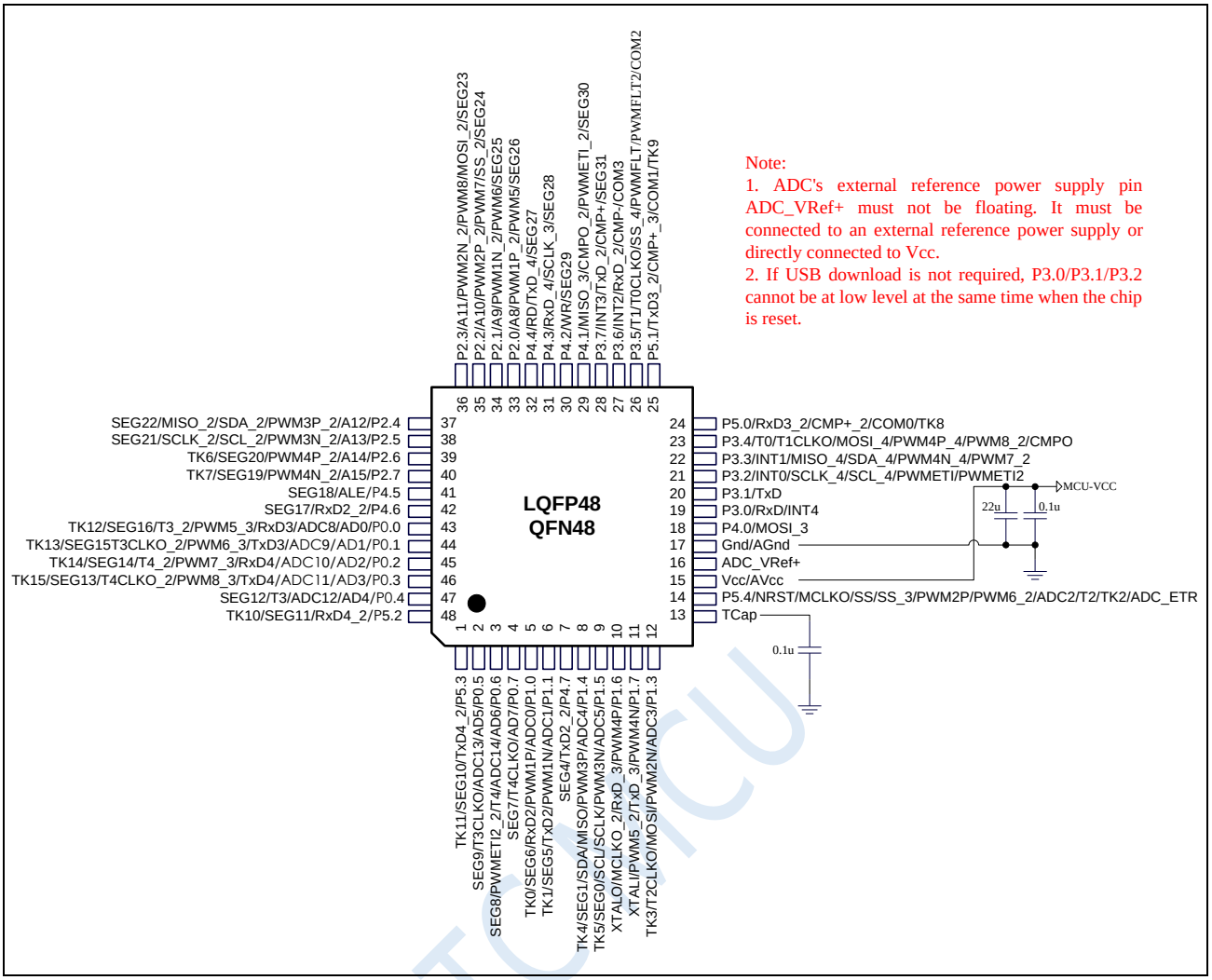
- ✓ Up to 61 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

➢ **Package**

- ✓ LQFP64 <12mm*12mm>, QFN64 <8mm*8mm>, LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

# Z.2.2 Pinouts


LQFP64
QFN64
Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.
MCU-VCC
22u
0.1u
0.1u
SEG22/MISO_2/SDA_2/PWM3P_2/A12/P2.4
SEG21/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
TK6/SEG20/PWM4P_2/A14/P2.6
TK7/SEG19/PWM4N_2/A15/P2.7
SEG3/PWM5_4/P7.4
SEG2/PWM6_4/P7.5
SEG1_2/PWM7_4/P7.6
SEG0_2/PWM8_4/P7.7
SEG18/ALE/P4.5
SEG17/RxD2_2/P4.6
TK12/SEG16/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
TK13/SEG15/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
TK14/SEG14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
TK15/SEG13/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
SEG12/T3/ADC12/AD4/P0.4
TK10/SEG11/RxD4_2/P5.2
P2.3/A11/PWM2N_2/PWM8/MOSI_2/SEG23
P2.2/A10/PWM2P_2/PWM7/SS_2/SEG24
P2.1/A9/PWM1N_2/PWM6/SEG25
P2.0/A8/PWM1P_2/PWM5/SEG26
P4.4/RD/TxD_4/SEG27
P4.3/RxD_4/SCLK_3/SEG28
P4.2/WR/SEG29
P4.1/MISO_3/CMPO_2/PWMETI_2/SEG30
P7.3/PWMETI_3/SEG32
P7.2/SEG33
P7.1/SEG34
P7.0/SEG35
P3.7/INT3/TxD_2/CMP+/SEG31
P3.6/INT2/RxD_2/CMP-/COM3
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/COM2
P5.1/TxD3_2/CMP+_3/COM1/TK9
P5.0/RxD3_2/CMP+_2/COM0/TK8
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
P3.1/TxD
P3.0/RxD/INT4
P6.7/PWM4N_3
P6.6/PWM4P_3
P6.5/PWM3N_3
P6.4/PWM3P_3
P4.0/MOSI_3
Gnd/AGnd
ADC_VRef+
Vcc/AVcc
P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
TCap
TK11/SEG10/TxD4_2/P5.3
SEG9/T3CLKO/ADC13/AD5/P0.5
SEG8/PWMETI2_2/T4/ADC14/AD6/P0.6
SEG7/T4CLKO/AD7/P0.7
SEG36/PWM1P_3/P6.0
SEG37/PWM1N_3/P6.1
SEG38/PWM2P_3/ADC6/P6.2
SEG39/PWM2N_3/ADC7/P6.3
TK0/SEG6/RxD2/PWM1P/ADC0/P1.0
TK1/SEG5/TxD2/PWM1N/ADC1/P1.1
SEG4/TxD2_2/P4.7
TK4/SEG1/SDA/MISO/PWM3P/ADC4/P1.4
TK5/SEG0/SCL/SCLK/PWM3N/ADC5/P1.5
XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
XTALI/PWM5_2/TxD_3/PWM4N/P1.7
TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3


Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.
LQFP48
QFN48
P2.3/A11/PWM2N_2/PWM8/MOSI_2/SEG23
P2.2/A10/PWM2P_2/PWM7/SS_2/SEG24
P2.1/A9/PWM1N_2/PWM6/SEG25
P2.0/A8/PWM1P_2/PWM5/SEG26
P4.4/RD/TxD_4/SEG27
P4.3/RxD_4/SCLK_3/SEG28
P4.2/WR/SEG29
P4.1/MISO_3/CMPO_2/PWMETI_2/SEG30
P3.7/INT3/TxD_2/CMP+/SEG31
P3.6/INT2/RxD_2/CMP-/COM3
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/COM2
P5.1/TxD3_2/CMP+_3/COM1/TK9
SEG22/MISO_2/SDA_2/PWM3P_2/A12/P2.4
SEG21/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
TK6/SEG20/PWM4P_2/A14/P2.6
TK7/SEG19/PWM4N_2/A15/P2.7
SEG18/ALE/P4.5
SEG17/RxD2_2/P4.6
TK12/SEG16/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
TK13/SEG15T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
TK14/SEG14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
TK15/SEG13/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
SEG12/T3/ADC12/AD4/P0.4
TK10/SEG11/RxD4_2/P5.2
P5.0/RxD3_2/CMP+_2/COM0/TK8
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
P3.1/TxD
P3.0/RxD/INT4
P4.0/MOSI_3
Gnd/AGnd
ADC_VRef+
Vcc/AVcc
P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
TCap
MCU-VCC
22u
0.1u
0.1u
TK11/SEG10/TxD4_2/P5.3
SEG9/T3CLKO/ADC13/AD5/P0.5
SEG8/PWMETI2_2/T4/ADC14/AD6/P0.6
SEG7/T4CLKO/AD7/P0.7
TK0/SEG6/RxD2/PWM1P/ADC0/P1.0
TK1/SEG5/TxD2/PWM1N/ADC1/P1.1
SEG4/TxD2_2/P4.7
TK4/SEG1/SDA/MISO/PWM3P/ADC4/P1.4
TK5/SEG0/SCL/SCLK/PWM3N/ADC5/P1.5
XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
XTALI/PWM5_2/TxD_3/PWM4N/P1.7
TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3

# Appendix AA Update Records

- **2022/3/9**
  1. Update data sheet
- **2021/12/21**
  1. STC8H8K64U adds LQFP32 and TSSOP20 pin diagram
  2. Correct typos in the document
  3. Add new product MCU notice information
- **2021/12/17**
  1. Revise the description of automatic switching from SPI master mode to slave mode
  2. Update RTC sample program
  3. Change the reset pin name to NRST in all pin diagrams
  4. Revise the timing calculation formula of timer 2/3/4
  5. Update the selection price list
  6. Update the pin diagram and pin description of STC8H8K64U
- **2021/11/23**
  1. Retype the EEPROM application example program
  2. Add appendix "STC Simulation User Manual" chapter
  3. Add detailed description to read-only special function register (CHIPID)
  4. Increase the use of VID and the allocation of PID in USB product development
  5. Add the pin diagram of LQFP32 and TSSOP20 of STC8H4K64TLR series
- **2021/11/16**
  1. Add the chapter "Unique ID Number and Important Parameters Stored in Read-Only Special Function Registers"
  2. Add a sample program for reading important parameters from read-only special function registers
  3. Add the appendix chapter of "USB Emulation Step Demonstration"
  4. Add EEPROM application example program
- **2021/10/27**
  1. Correct the description of LCD operating voltage setting register
  2. Update the interrupt structure diagram
  3. Correct typos in the document
  4. All BMMs are renamed to DMA
  5. Update the supply information of STC8H3K64S2 series and STC8H3K64S2 series
  6. Add the application precautions of using external crystal oscillator for RTC of STC8H4K64TLR system
- **2021/10/8**
  1. Update the port switching table for 8-bit data and 16-bit data in the LCM chapter
  2. Correct typos found in the document
  3. Add the description of external interrupts to the interrupt source table in the Interrupt System chapter
  4. Take screenshots using the new version of the software in the Emulator chapter in the appendix
  5. Added the chapter "How to Test I/O Ports" in the appendix
- **2021/9/26**
  1. Add a sample program for serial port DMA timeout processing and data verification
- **2021/8/30**
  1. Modify the title of some chapters
  2. Add the chapter "Notes on Baking Before Reflow Soldering" in the appendix

- **2020/8/26**

1. Add the chapter of timer calculation formula
2. Add the chapter of serial port baud rate calculation formula
3. Added 16-bit advanced PWM output frequency calculation formula chapter
4. Add ADC related calculation formula chapter
5. Add 12-bit ADC static parameter reference data
6. Add the parameter of the number of clocks required for MDU16 operation
7. Increase the time parameter required for EEPROM operation
8. The special function registers in all chapters are listed separately as directory subsections for easy searching
9. Precautions for adding STC8H series MCU to replace STC8A/8F series

- **2020/8/21**

1. Modify some errors in the description of the document
2. Added description to the 16-bit advanced PWM timer chapter
3. The first set of 16-bit advanced PWM timer PWM1 is renamed PWMA
4. The second group of 16-bit advanced PWM timer PWM2 is renamed to PWMB
5. Add "Using PWM to realize complementary SPWM" sample program

- **2020/8/10**

1. Add watchdog timer chapter
2. Organize the chapter on wake-up timer
3. Add the appendix chapter about STC download tool usage instructions

- **2020/8/6**

1. Explain the working temperature
2. Add a sample program of 16-bit advanced PWM timer for measuring period, duty cycle, high level and low level width
3. Added the description of the external 32.768KHz crystal oscillator control register X32KCR
4. Add the application circuit diagram downloaded using the universal USB to serial tool
5. Update application notes

- **2020/7/16**

1. Add description of BUS_SPEED register
2. Added soldering instructions for QFN/DFN packaged chips
3. Add reference circuit diagram of BLDC brushless DC motor drive (without HALL)
4. Add quadrature encoder mode sample program
5. Add the orthogonal decoding example provided by Chengdu Zhufei Technology, see Appendix L
6. Add EEPROM programming instructions
7. Add the method of setting U8W/U8-Mini to pass-through mode in the chapter of downloading ` application circuit diagram

- **2020/7/3**

1. Modify the problem of disordered layout of some text in the file

- **2020/7/2**

1. Add STC8H2K64T series
2. Add the appendix chapter, "How to reset the user program to the system area for ISP download without power failure"
3. Add the appendix chapter, "Develop your own ISP program using STC's IAP series MCU"
4. Add the appendix chapter, "Precautions for STC8H series MCU to replace STC15 series"
5. Add appendix chapter, "Official website description"
6. Add LED driver description chapter
7. Add touch button description chapter
8. Added RTC real-time clock description chapter
9. Add ADC conversion timing diagram in ADC chapter

- **2020/6/15**

1. Add ADC_VRef+ pin description
2. Added instructions for using diodes and resistors in the USB to serial reference circuit
3. Modify the description of the I/O port drive current control register PxDR (1: normal drive current; 0: strong drive current)
4. Add description of I2C slave device address

- **2020/6/8**

1. Add the description of the fastest ADC conversion speed
2. Detailed description of I2C bus speed setting
3. Update analog USB and hardware USB mode ISP download reference circuit diagram
4. Add the preset user-selectable internal IRC frequency description during hardware USB download
5. Add DFN8, QFN20, QFN32, QFN48, QFN64 substrate description in the package drawing
6. Add sample program for comparator multiplexing (comparator + ADC input) application

- **2020/5/29**

1. Add adding circuit application to ADC chapter
2. Add the description of register EAXFR
3. Correct the errors in the DFN8 package dimension drawing
4. Add the method of using a third-party application to call the release project program

- **2020/5/25**

1. Add negative voltage detection circuit in ADC chapter
2. Fix garbled characters in some pictures

- **2020/5/20**

1. Update the power consumption parameters of the clock stop mode when the low-voltage detection wake-up function is enabled in the electrical characteristics
2. Update the power consumption parameters of the clock stop mode when the comparator power-down wake-up function is enabled in the electrical characteristics
3. The ADCTIM register is added to the ADC sample program to control the ADC internal timing
4. Correct some clerical errors in the document
5. Add an interrupt that can be used to wake up from clock stop mode in the features of each microcontroller series
6. Add sample program for I/O port interrupt
7. Added ISP download step guide in the ISP download application circuit diagram
8. Add power-down wake-up timer to wake up the power saving mode sample program

- **2020/5/14**

9. Add description of comparator multiplexing
10. Add PWM trigger ADC sample program
11. Added ADC working clock frequency description in ADC chapter
12. Add ADC reference circuit diagram in ADC chapter
13. Update the power consumption parameters of low voltage detection and comparator in electrical characteristics
14. Update the reference price of STC8H1K08 series
15. Add reference circuit diagram of PWM as DAC
16. Update the pin diagram and the description of the PWM external trigger pin PWMETI in the pin description
17. Add power-on reset and button reset reference circuit diagram

- **2020/4/29**

1. Change the serial port download reference circuit diagram, the series resistance on the TxD pin of the MCU is changed from 300 ohms to 100 ohms
2. Fix the error in the power supply part of the reference circuit diagram using PL2303GL for ISP download

- **2020/4/26**

1. Update I/O speed parameters in electrical characteristics
2. Add the reference pin diagram of PDIP40 of STC8H8K64U series
3. Update the speed parameter of the comparator in the electrical characteristics
4. Correct the timing of setting TI and RI in Chapter 14.6 Serial Notes
5. Added the description about analog filtering and digital filtering in the chapter of comparator
6. In Appendix E, the connection error between MAX232 and RS485 is corrected

- **2020/4/8**

1. Add the description of the power-down wake-up timer register
2. Update the content about the overall drive current in the I/O port chapter

- **2020/3/27**

1. Delete STC8H8K64S2U series
2. STC8H8K64S4U series was renamed STC8H8K64U series
3. The IRC24MCR register is renamed HIRCCR

4. Add STC8H8K64S4U model and use PL2303GL to download the reference circuit diagram
5. Add STC8H8K64S4U model direct hardware USB download reference circuit diagram
6. Rename the power-related pin names of all chips in a unified style
7. Update the power consumption of the chip in the DC characteristics at different operating frequencies
8. Added a description at the beginning of the advanced PWM timer chapter
9. Update STC8H8K64U series selection price list
10. Correct the formula for calculating voltage in the chapter "Using ADC channel 15 to measure external voltage or battery voltage"

- **2020/3/13**
  1. Re-calibrate the content in the advanced PWM timer chapter
  2. Add the sample program of advanced PWM timer used as external interrupt
- **2020/3/6**
  1. Correct the incorrect description of the internal reference signal source in the document
  2. Add application circuit diagrams of general precision ADC and high precision ADC
  3. Add static parameters of ADC module
  4. Added STC8H8K64S4U series LQFP48 pin diagram and pin description
  5. Added STC8H8K64S2U series LQFP48 pin diagram and pin description
  6. Delete STC8H3K64S4 series
  7. Delete STC8H3K64S2 series
  8. Added I/O Port Interrupt Chapter
  9. Add the description of I/O interrupt in the interrupt system chapter
  10. Add USB sample program (HID interface)
  11. Reorganized the chapter structure of the pin diagram
  12. Correct the chip characteristics of STC8H1K17 model
- **2020/1/20**
  1. Add sample code related to advanced PWM
  2. Add "a typical triode control" circuit
  3. Add "typical LED control" circuit
  4. Add the reference circuit of "I/O port interconnection of 3V/5V devices in mixed voltage power supply system"
  5. Add the reference circuit of "How to make the I/O port be low when power-on reset"
  6. Add "Using 74HC595 to drive 8 digital tubes (serial expansion, 3 lines)" reference circuit
  7. Add "I/O port directly drive LED digital tube" reference circuit
  8. Added the description of "Automatically start ISP download after receiving user command when running user program"
- **2020/1/17**
  1. Add MDU16 operation clock description
  2. Added STC8H1K08 series QFN20 pin diagram
- **2020/1/15**
  1. Add "ADC as capacitive sensing touch button" chapter
  2. Add "ADC as key scan application circuit diagram" chapter
  3. Add appendix "RS485 automatic control or I/O port control circuit diagram"
  4. Add appendix "RS485 partial circuit diagram in U8W download tool"
- **2019/12/31**
  1. Modified the number of I/O ports of STC8H8K64S2U series and STC8H8K64S4U series, the actual number is 60 I/O at most
- **2020/12/30**
  1. Create STC8H series MCU technical reference manual document
  2. Add STC8H1K28 series
  3. Add STC8H1K08 series
  4. Add STC8H3K64S4 series
  5. Add STC8H3K64S2 series
  6. Add STC8H8K64S4U series
  7. Add STC8H8K64S2U series
  8. Add MDU16 multiplication and division unit description

# Appendix BB STC8 series naming tidbits

**STC8A:** The letter "A" stands for ADC, which is the starting product of STC 12-bit ADC

**STC8F:** No ADC, PWM and PCA functions, the current version of the STC8F chip is fully compatible with the original STC8F pins, but the internal design has been optimized and updated, the user needs to modify the program, so it is named STC8C

**STC8C:** The letter "C" stands for revision, which is a revised chip of STC8F

**STC8G:** The letter "G" was originally a typo when the chip was produced. Later, the G series was defined as the "GOOD" series. The STC8G series is easy to learn.

**STC8H:** The letter "H" is taken from the first letter of the English word "High", and "High" means "16-bit advanced PWM"