Installation Manual

Contents

Installation Manual

1、Product Overview

1.1 Introduction to OpenPLC

OpenPLC is an open-source programmable logic controller. It is easy-to-use software and the first fully functionally standardized open-source PLC, compliant with standards in both software and hardware. The OpenPLC project follows the IEC 61131-3 standard, which defines the basic software architecture and programming languages of PLCs. OpenPLC is mainly used in fields such as industrial and home automation, IoT, and SCADA research.

OpenPLC Editor supports the five languages defined in the IEC 61131-3 standard: Ladder Diagram (LD), Function Block Diagram (FBD), Instruction List (IL), Structured Text (ST), and Sequential Function Chart (SFC). These programming languages provide users with multiple choices to adapt to different programming needs and preferences. The versatility and ease of use of OpenPLC Editor make it an ideal tool for PLC programming.

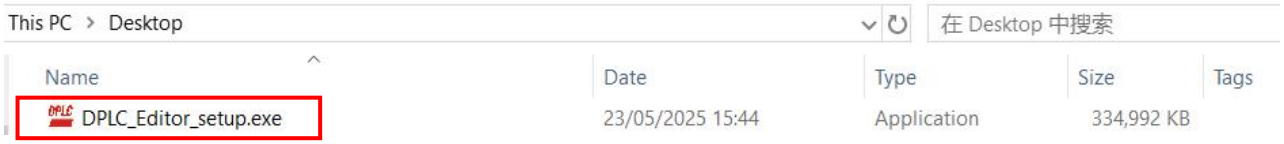
1.2 System Requirements

Processor	Intel Core i3 or above	Display Scaling Settings	100%（Recommended）
Memory	Not less than 4G	Display Resolution	1920*1080（(Recommended)))
Hard Disk	300GB or above	Operating System	Windows 7 and above (Recommended)

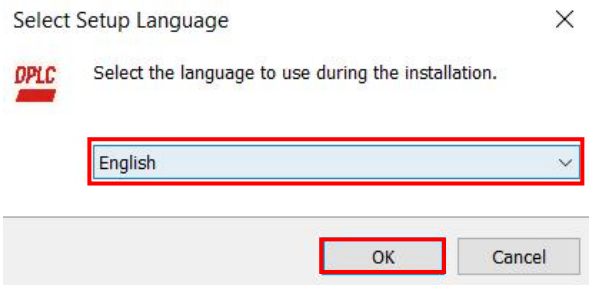
1.3 Installation Steps

Download path for programming software: https://www.meanwell.cc/productSoftware.aspx

Click the installation package to start installation.

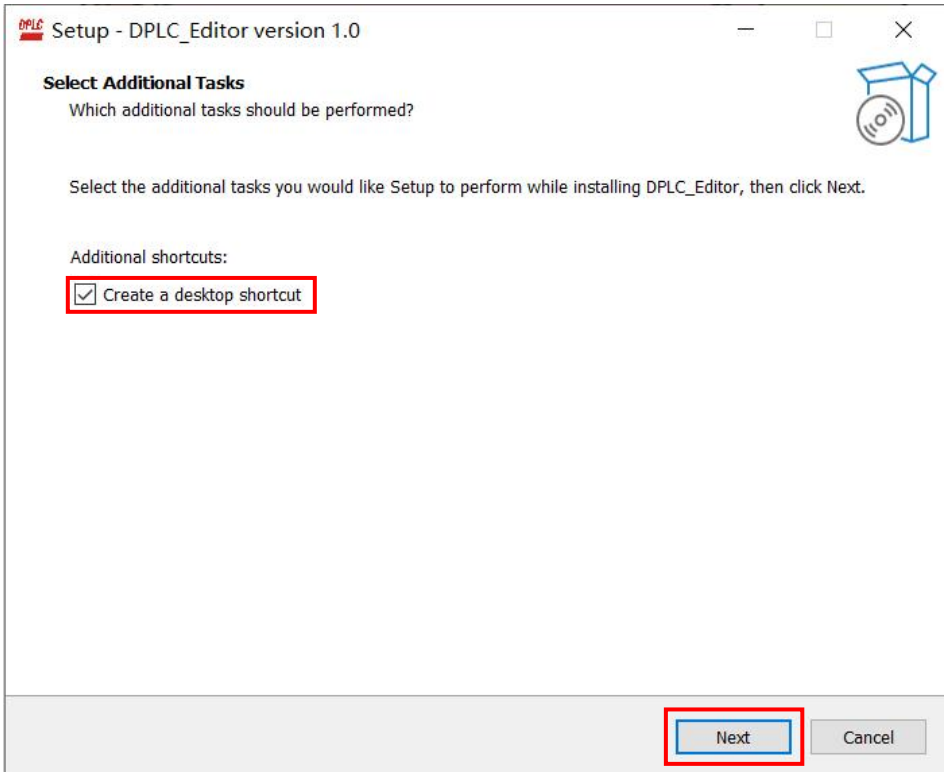


The following window will pop up. Select the language used during installation:

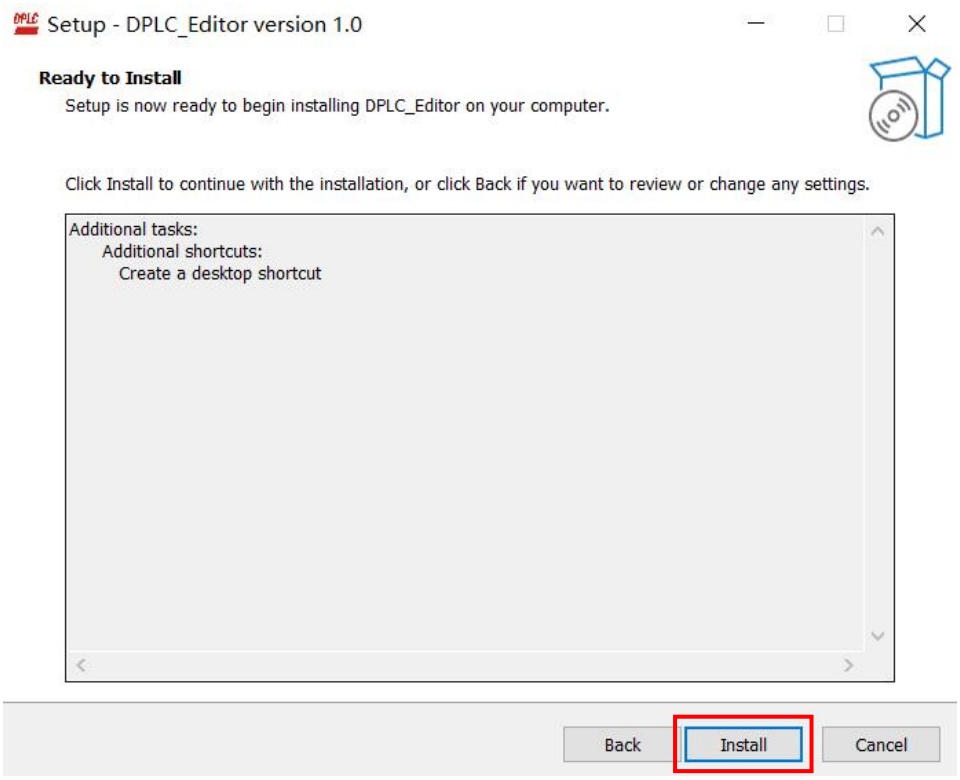


Select Create a shortcut and click Next:

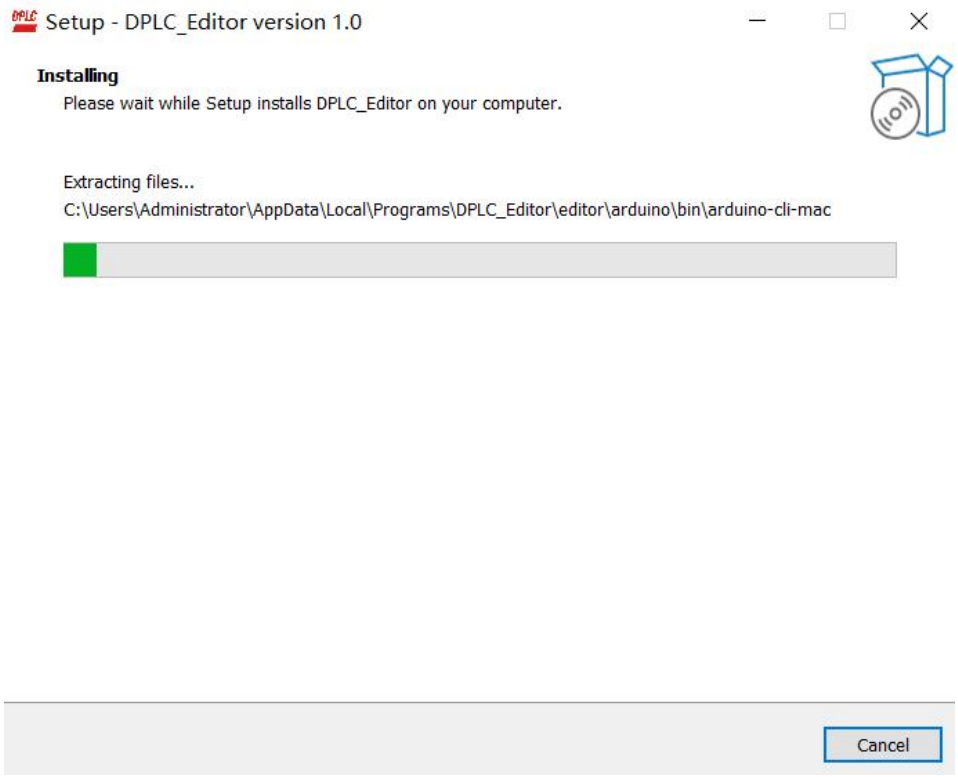
Installation Manual



Click to install:



Installation Manual



After the download is complete, click "Close", and the "DPLC_Editor" icon will appear on the desktop.

Installation Manual

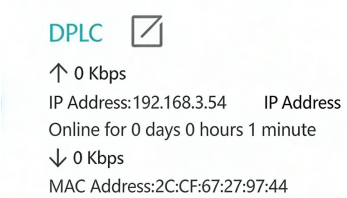
2、Programming Environment

2.1 System Configuration

2.1.1 Ethernet Connection

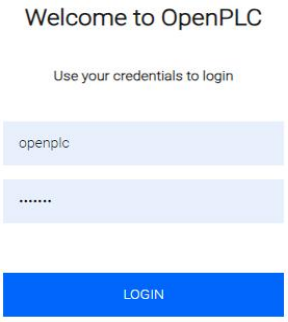
（1）Connect the DPLC to the router with a network cable, observe whether the RJ45 yellow light is constantly on and whether the green light is flashing. Connect the computer's network to the same router, and the Ethernet connection configuration is complete. The red box in the figure below is the DPLC network port connection point.

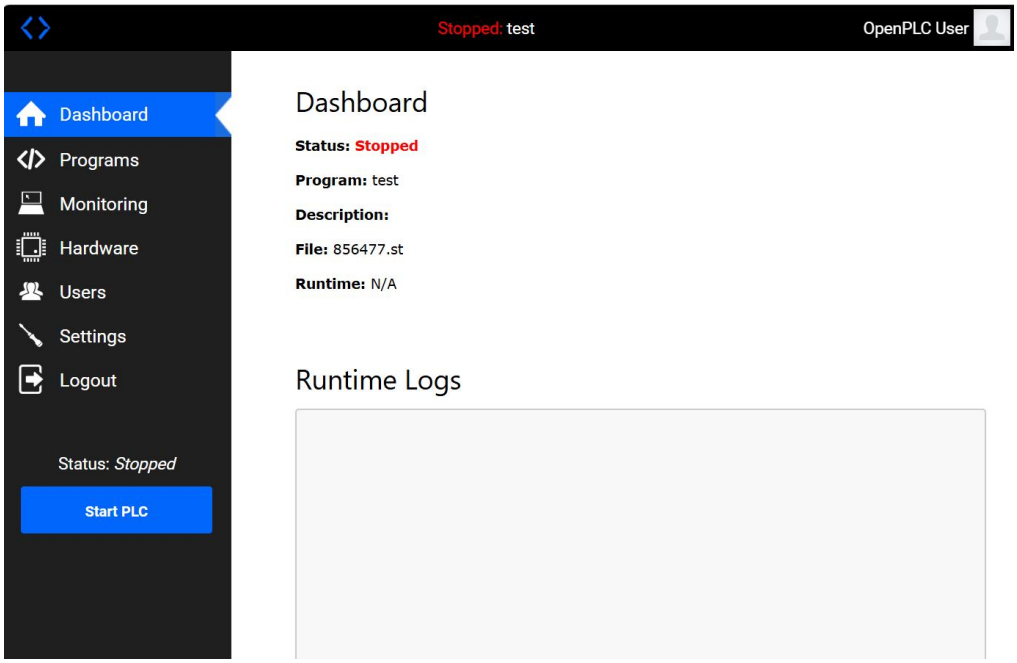
（2）Enter the router to view the IP address of the DPLC.



（3）Enter IP:80 in the browser.

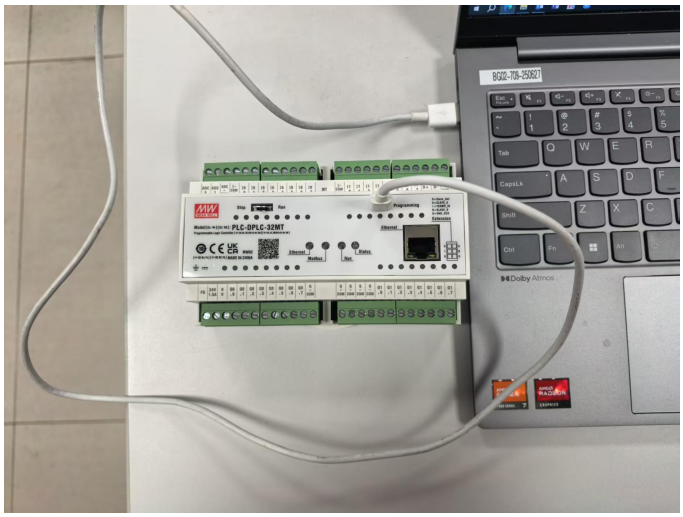
（4）Press Enter to enter the login interface. The default username and password are both openplc. Click LOGIN to enter the burning interface.



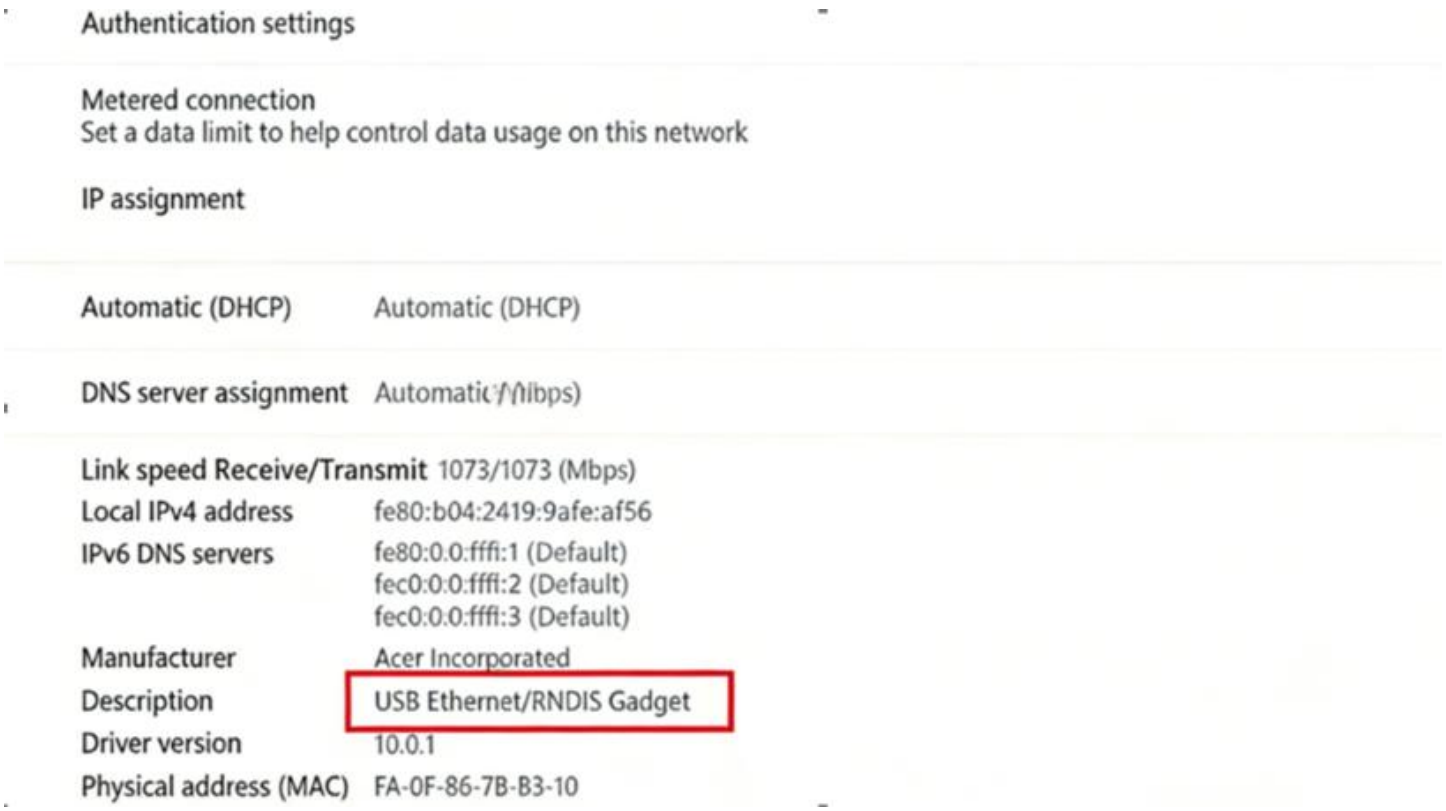


2.1.2 USB Connection

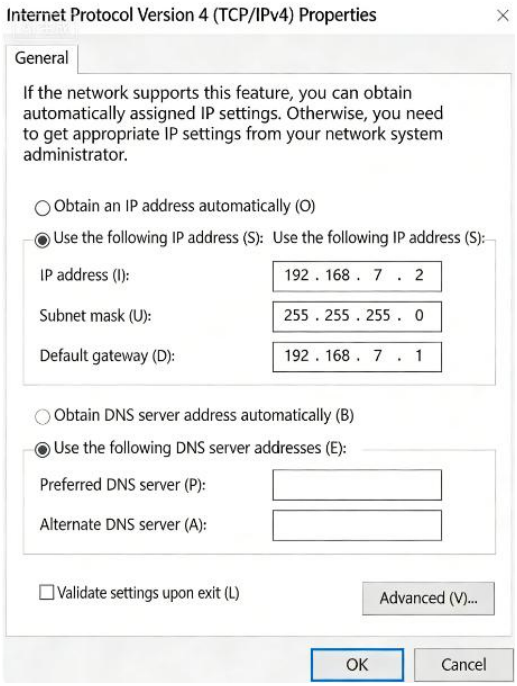
（1）Connect the DPLC Type-C port and the computer.



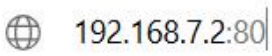
（2）Find the Raspberry Pi CM4 WLAN settings.

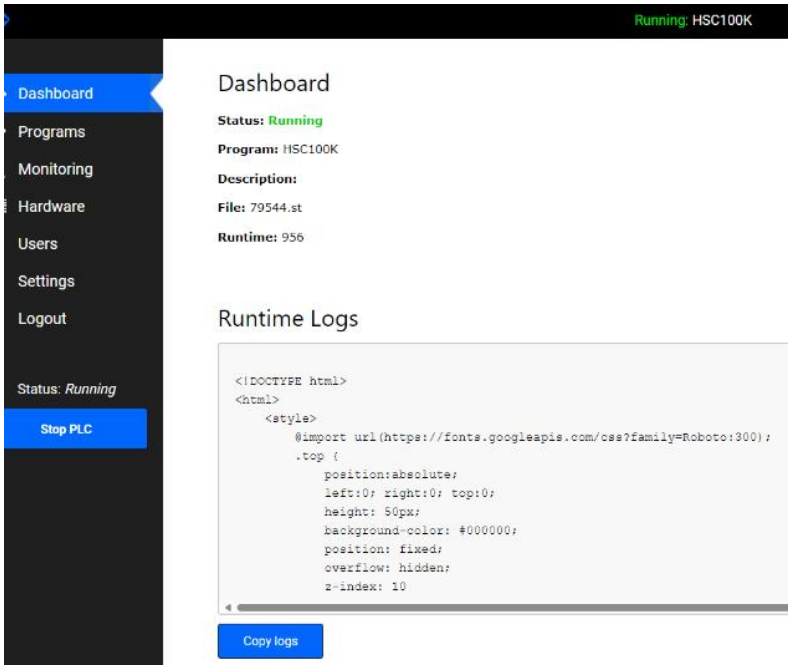


(3) Enter WLAN settings to set the computer IP address and save.



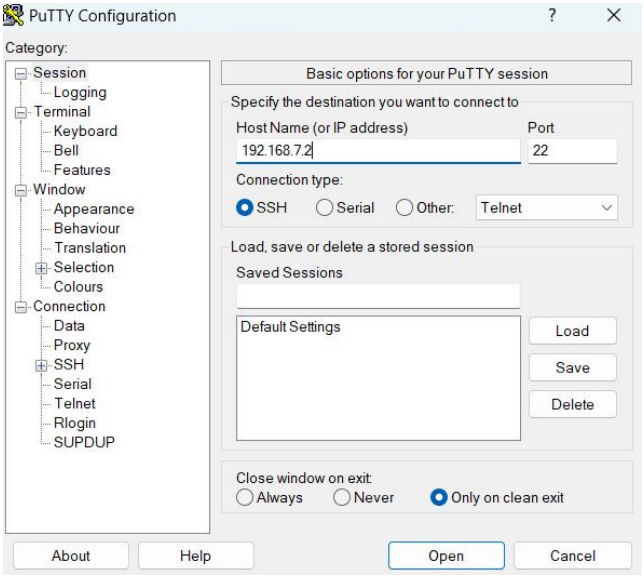
(4) Enter 192.168.7.2:80 in the browser and enter.



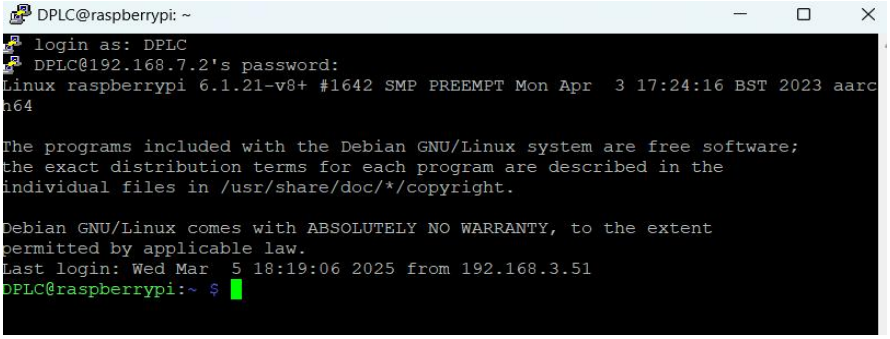


(5) If not connected to Ethernet, use USB connection to update time, follow the steps below:

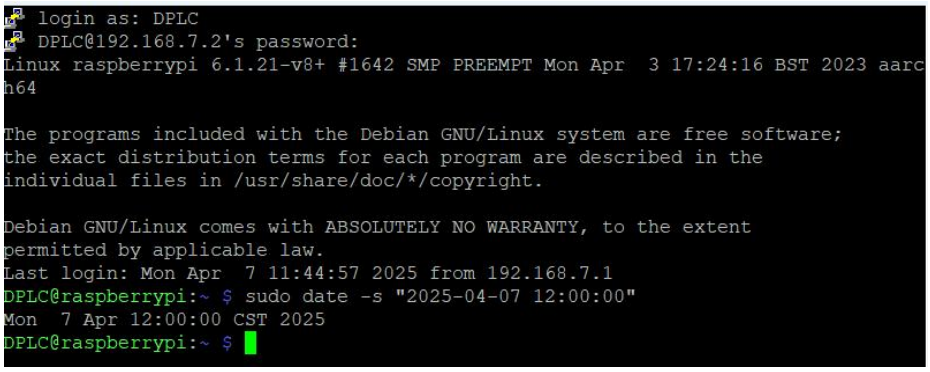
1. Open remote control software and enter IP address 192.168.7.2.



2. Enter name and password, both are DPLC, to enter the control panel.



3. Enter the command sudo date -s "2025-04-07 12:00:00". 2025-04-07 is the date, and 12:00:00 is the time.

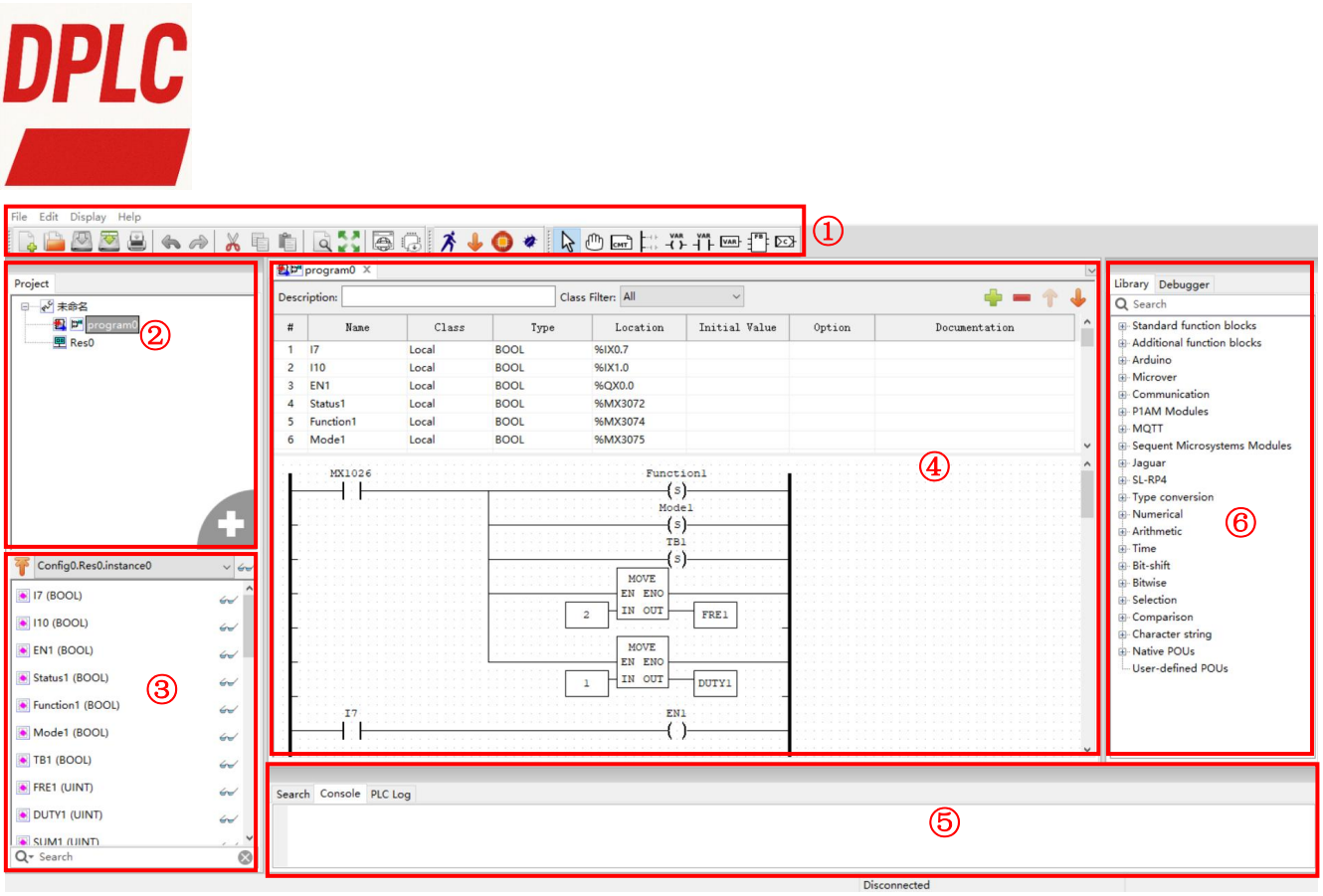


4. If offline for a long time and you want to connect to Ethernet to update time, first power on and boot up, then plug in the network cable.

2.2 Programming Interface

2.2.1 Compilation Interface

Double-click "DPLC_Editor" to open the programming software.



The DPLC_Editor programming interface mainly contains:

①Menu Toolbar

②Project Window

③Debug Window

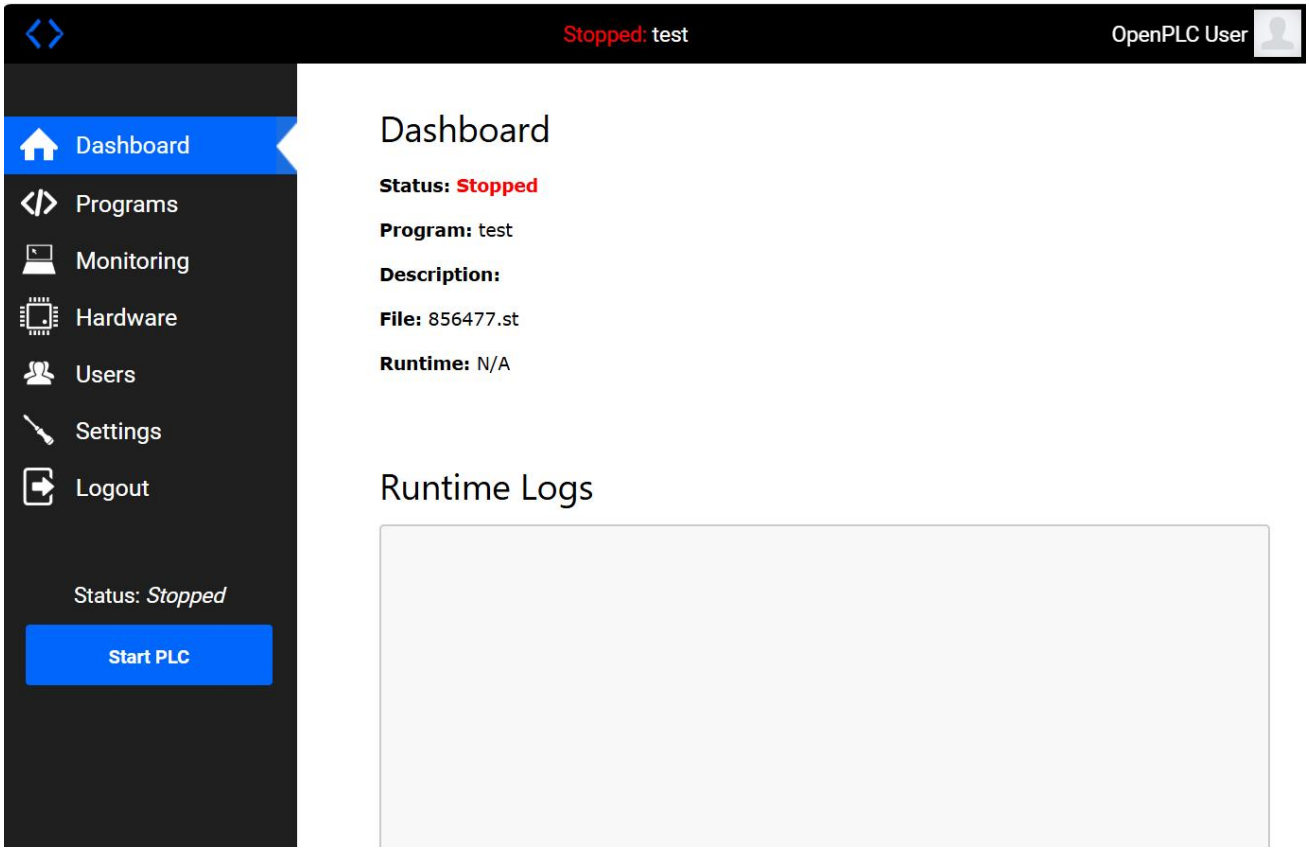
④Programming Window

⑤Message Window

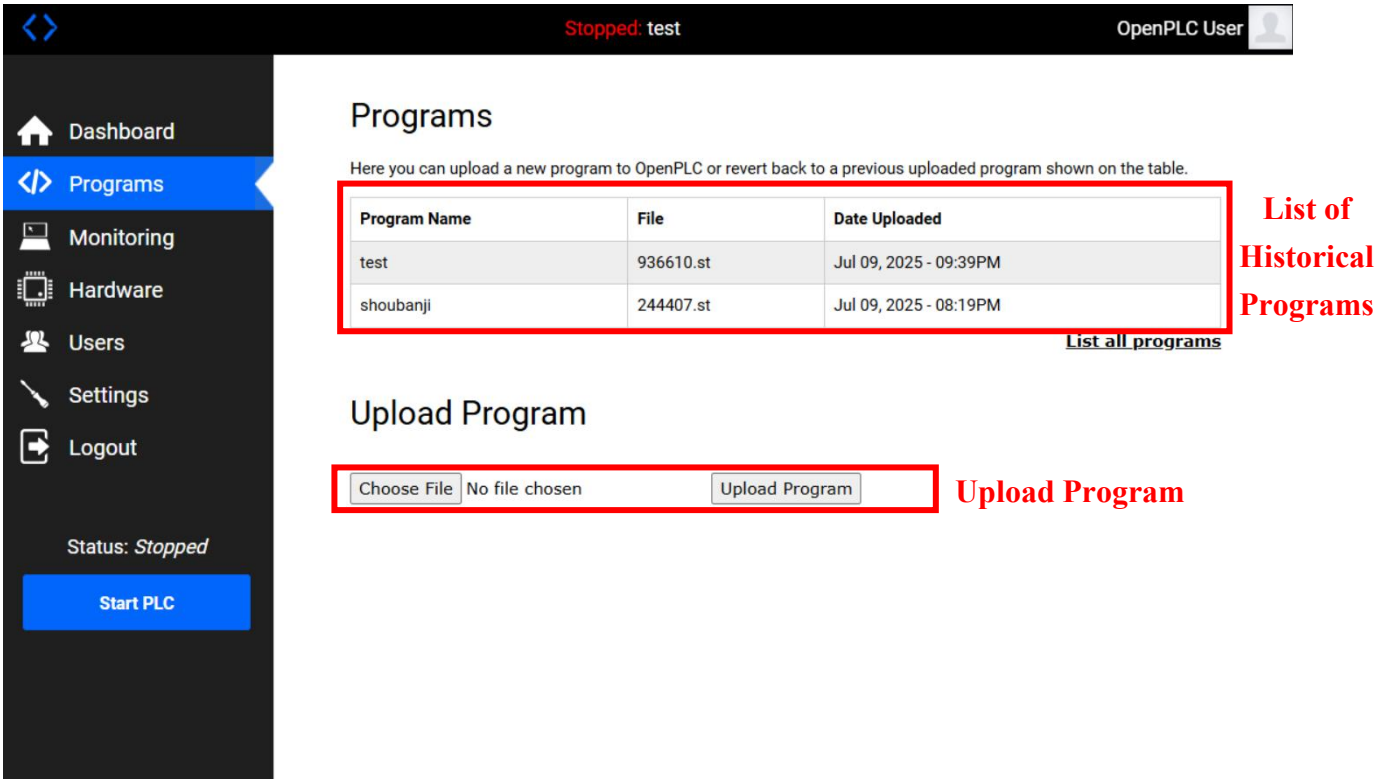
⑥Toolbox

2.2.2 Burning Interface

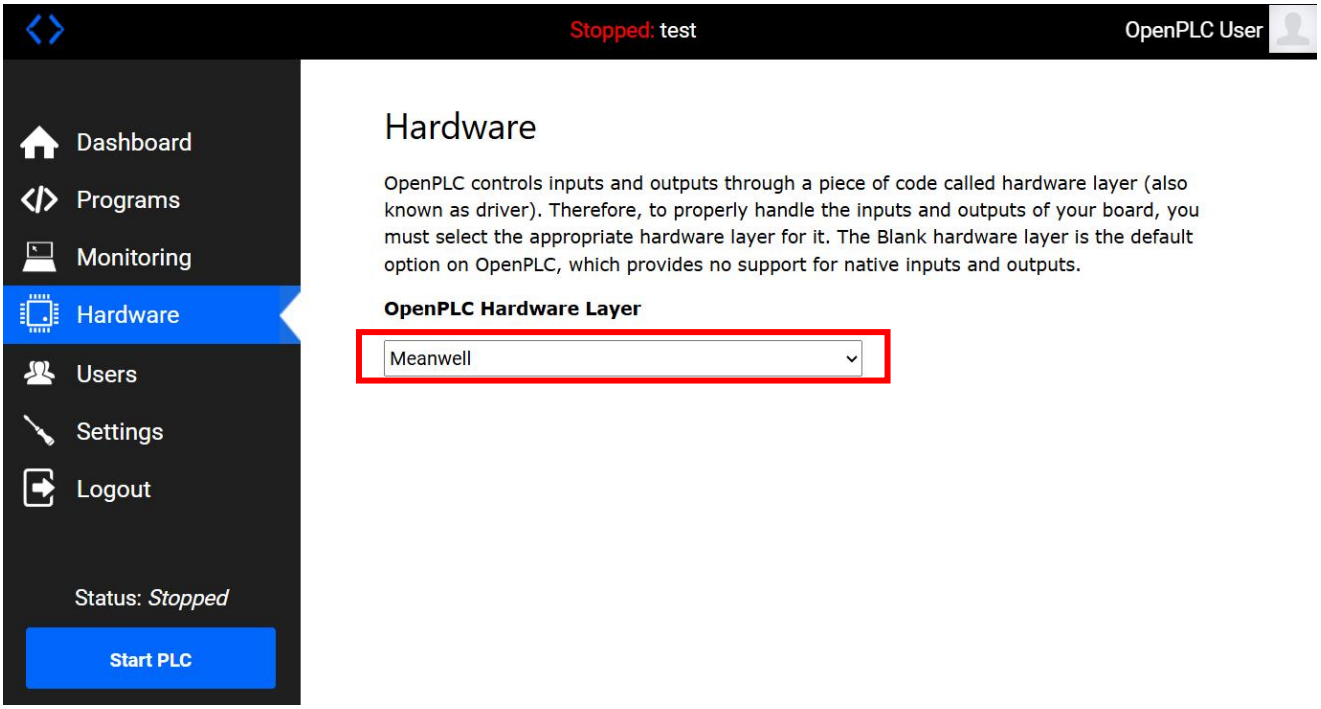
（1）Dashboard：Runtime status display.



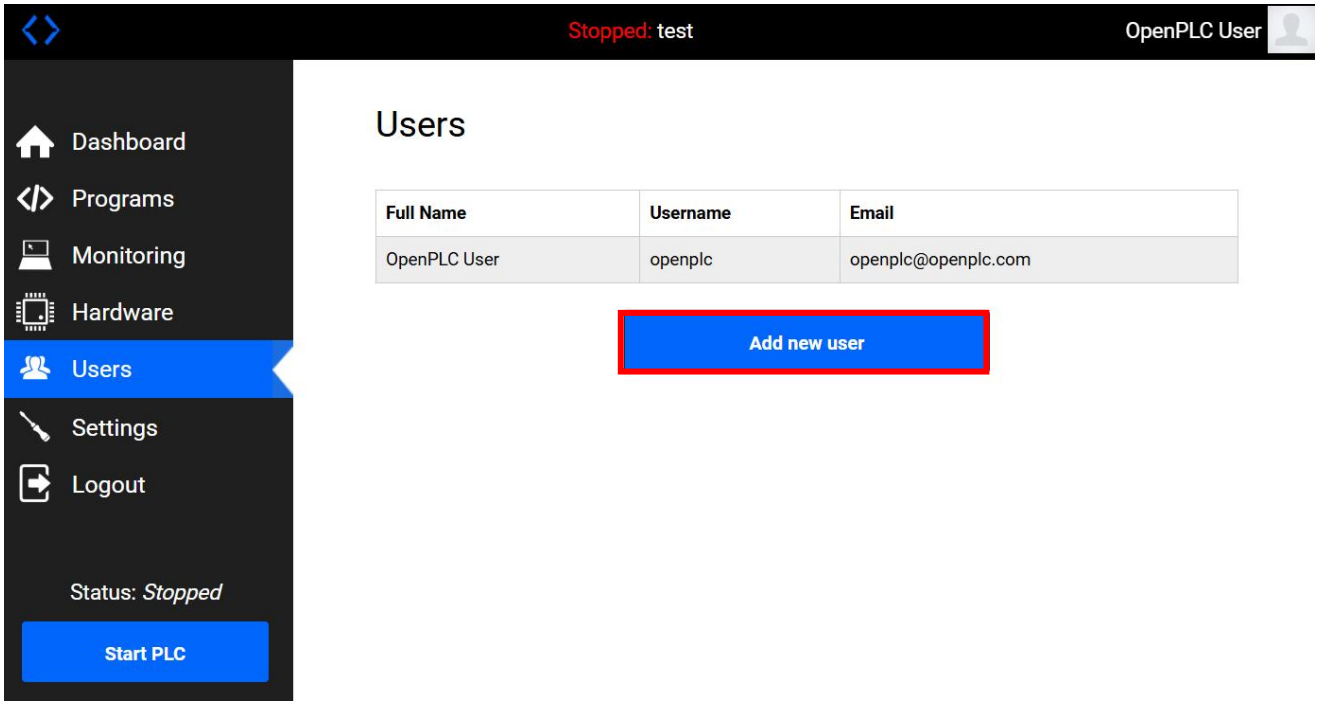
(2) Programs: In this interface, you can upload projects and view historical projects.

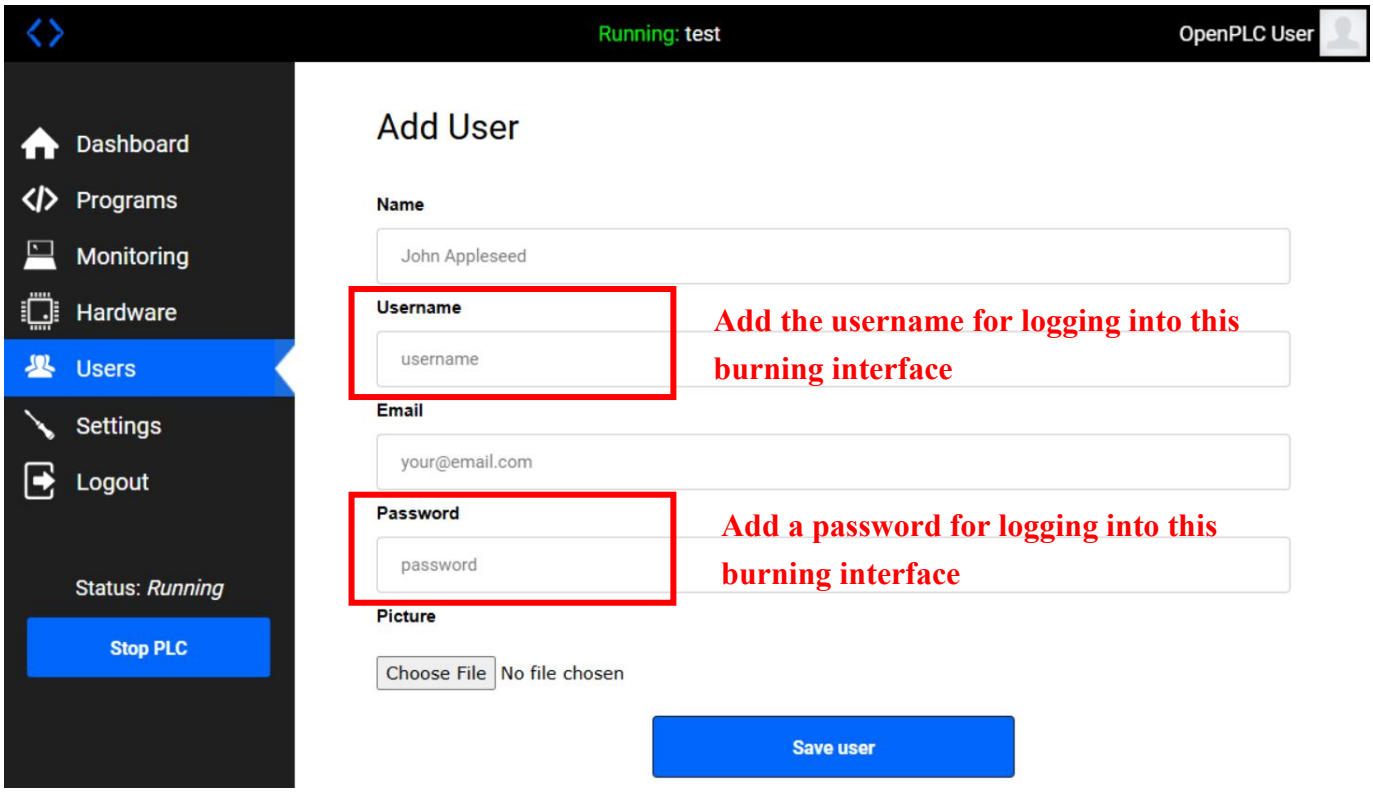


（3）Hardware: Only Meanwell is selected, with no other alternatives.

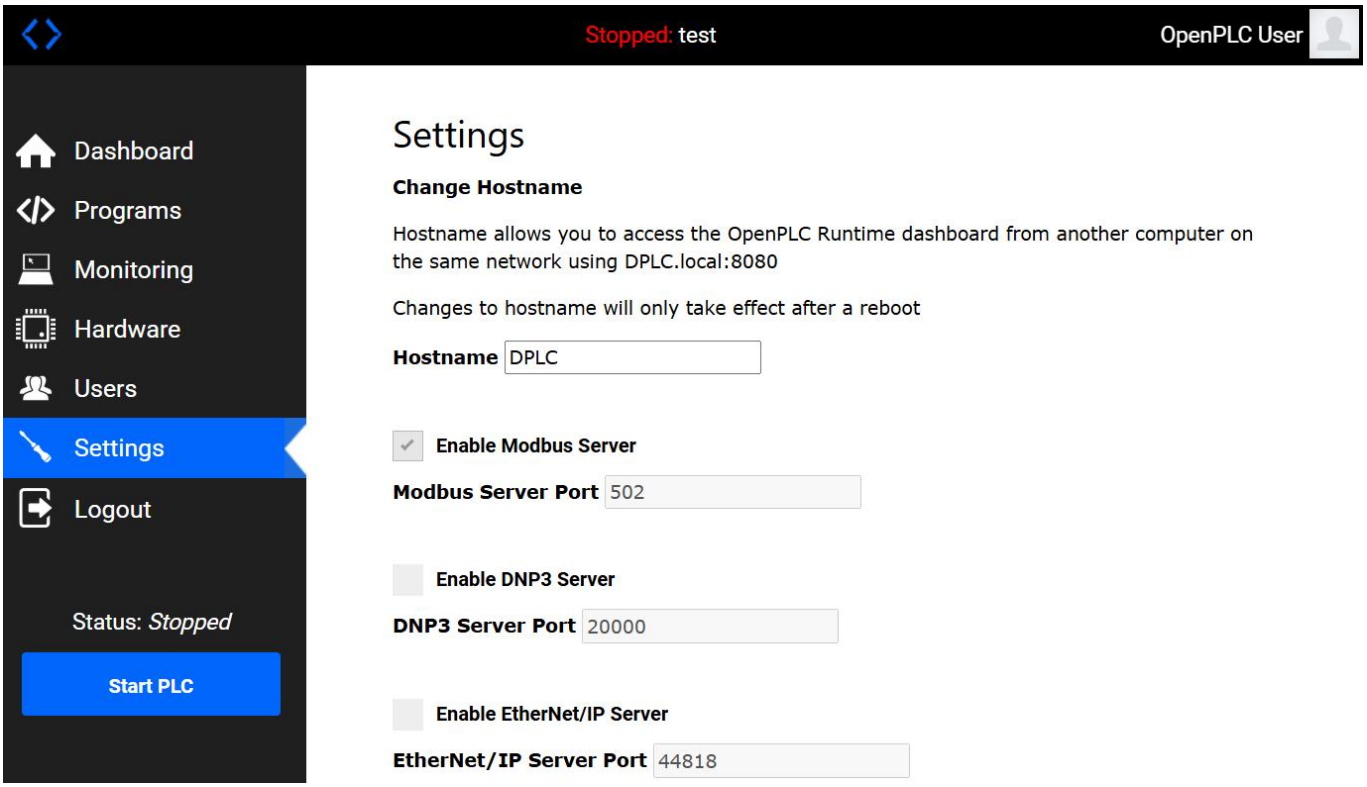


（4）Users: Add the login username and password. A default username and password are provided for first-time use, and you may add new ones as needed.

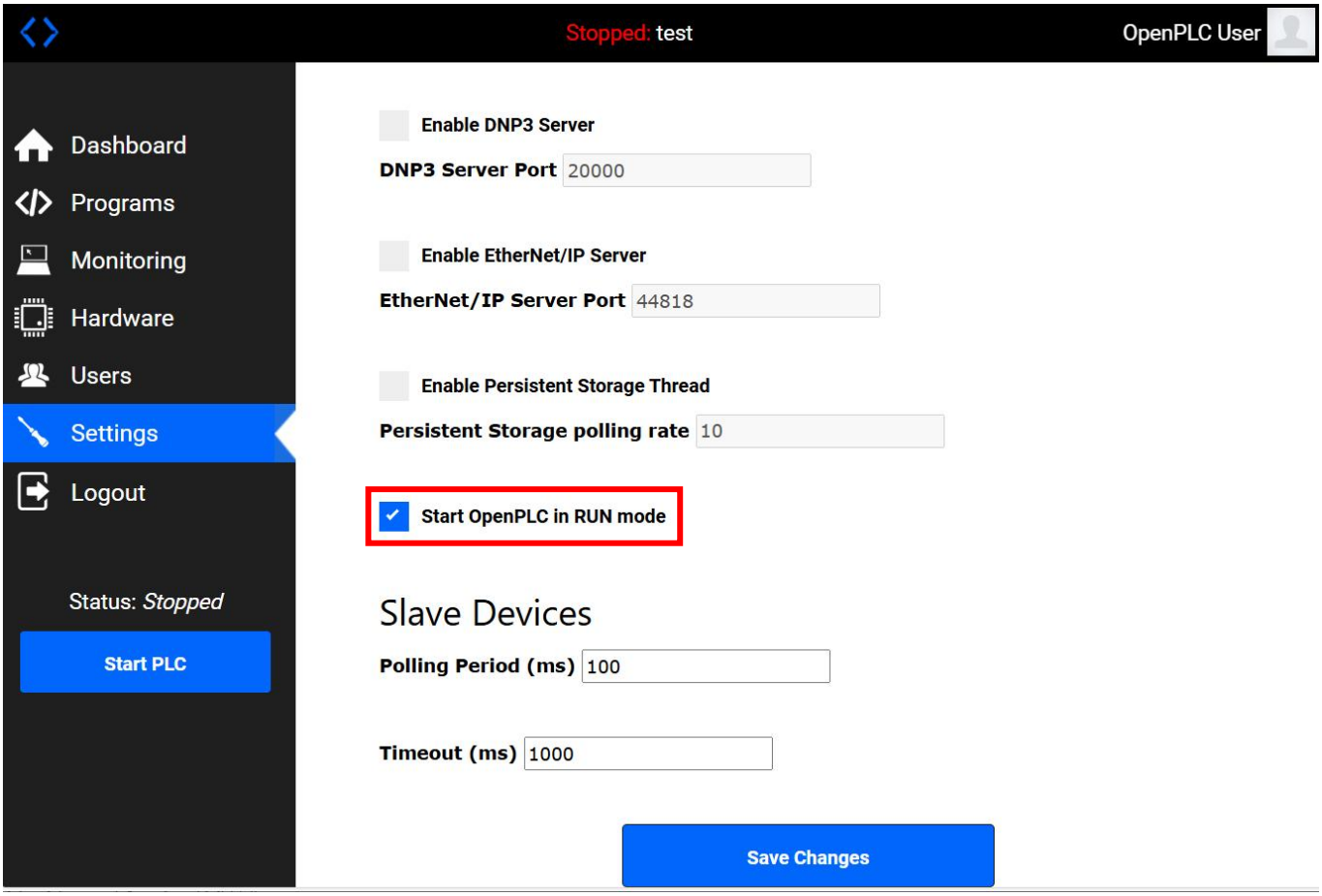




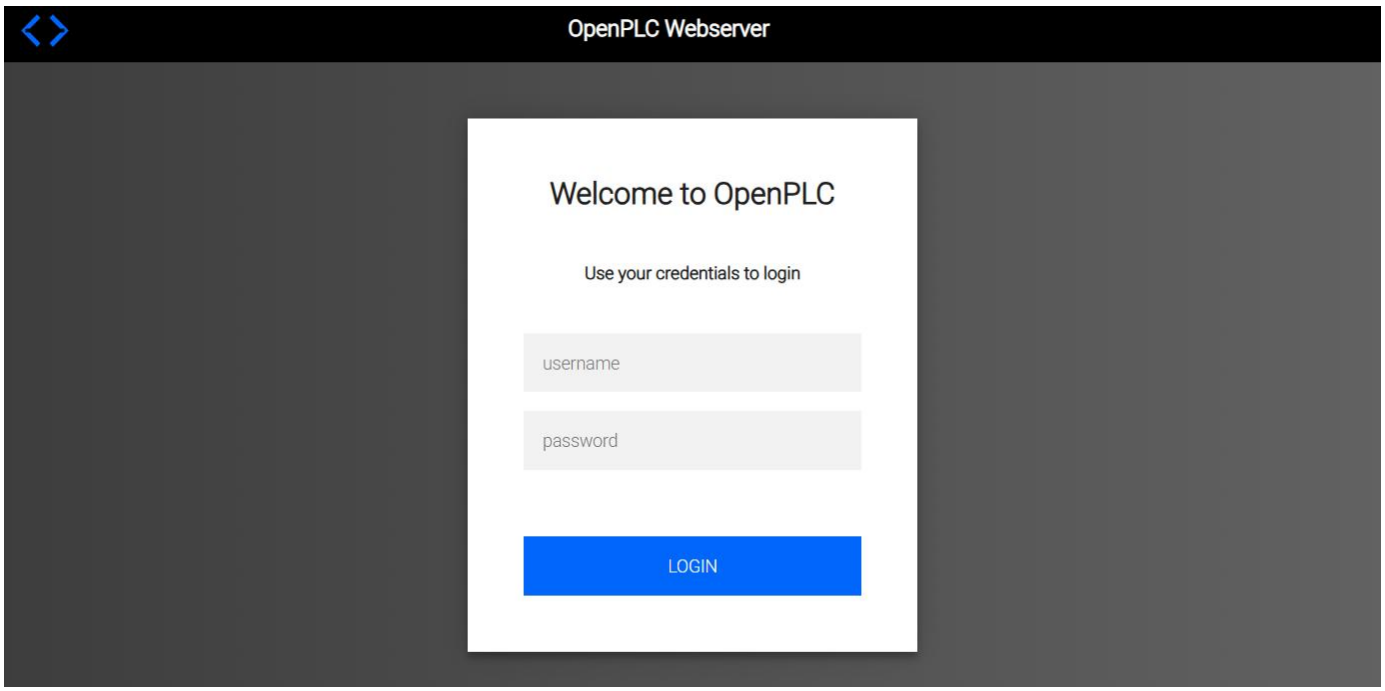
（5）Settings: Settings regarding burning.



It is recommended to check "Start OpenPLC in RUN mode", so that the RUN state can be maintained when exiting the burning interface.



(6) Logout: Exit the burning interface; you need to enter the username and password again to enter.

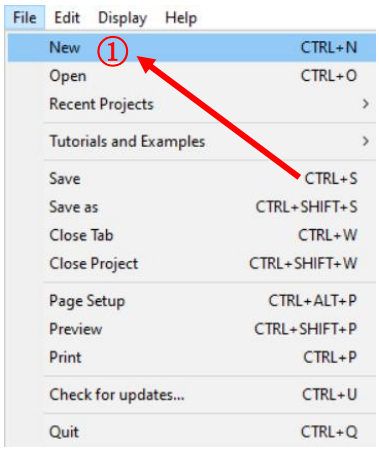


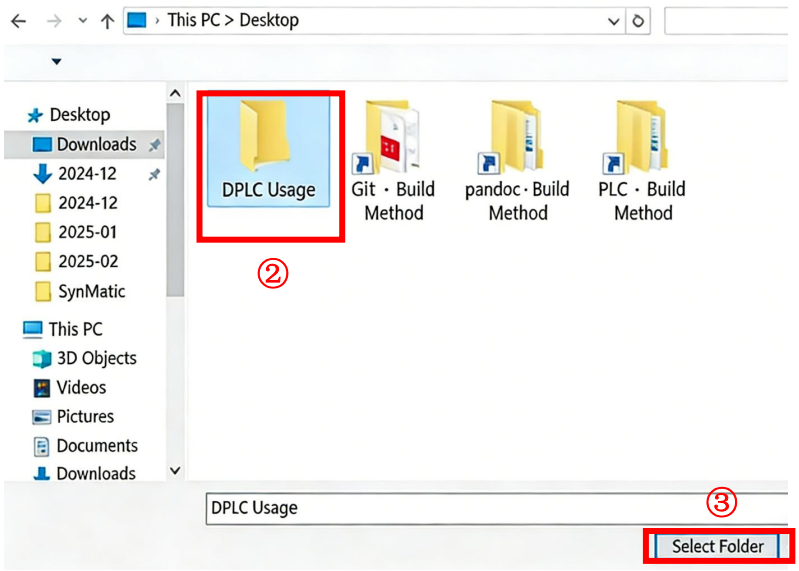
2.3 Usage Instructions

2.3.1 Creating a New Project

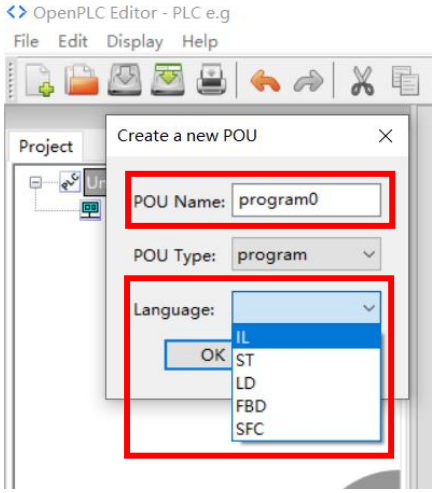
Double-click "DPLC_Editor" to open the programming software.

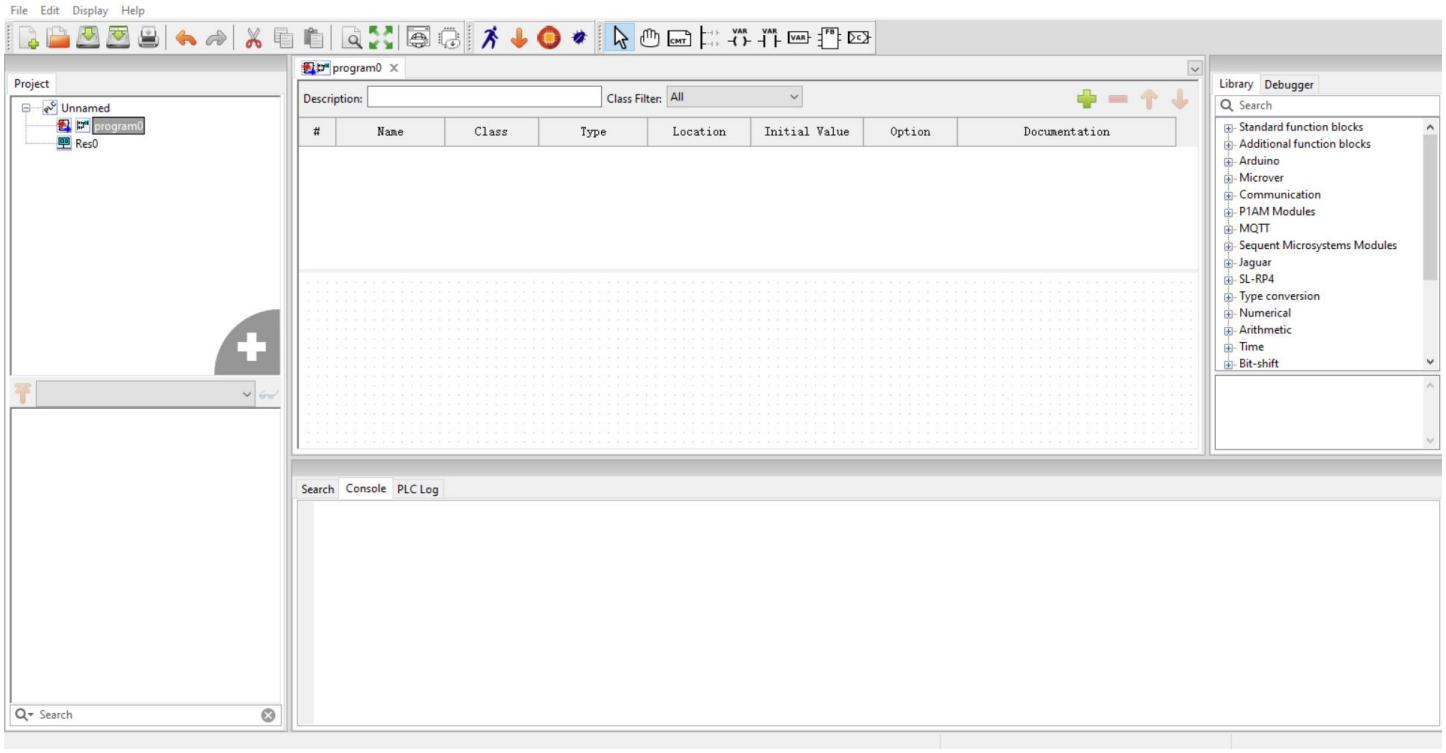
File -> New -> Select a blank folder. If it is not a blank folder, creation will fail!!!





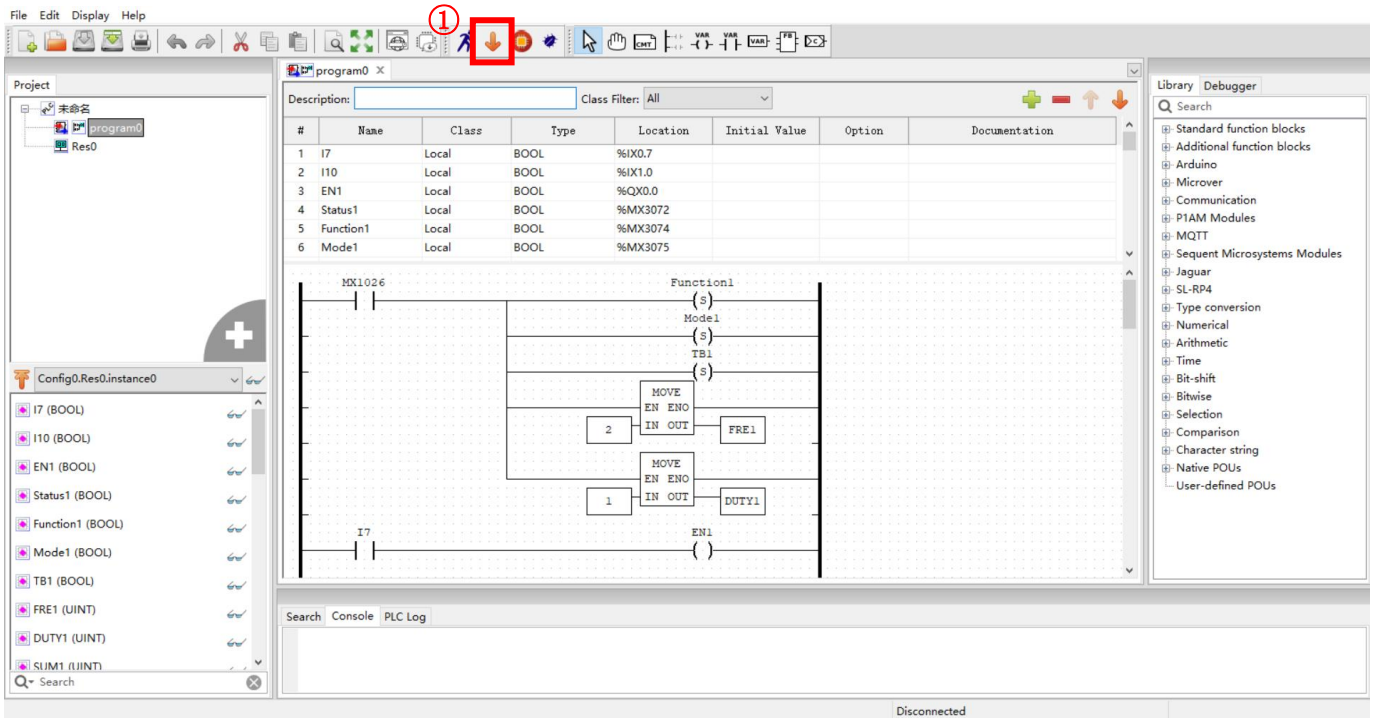
After creating a new project, you can modify the project name and compilation language. After modification, click OK.

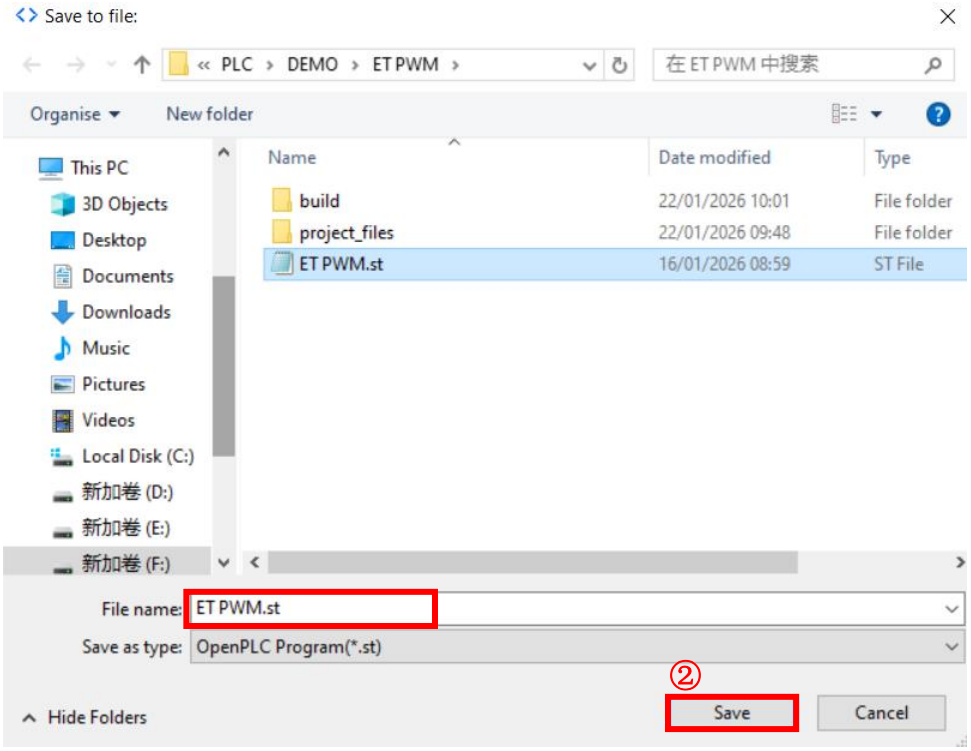




2.3.2 Compiling

The computer and DPLC are connected via the same router. Click "Generate program for OpenPLC Runtime" in the menu toolbar -> generate the .st file and save it.

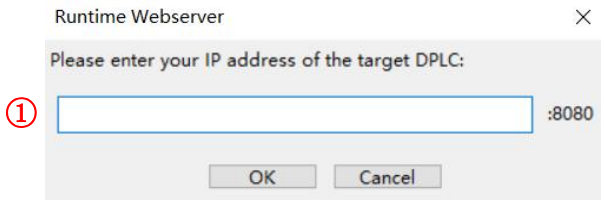


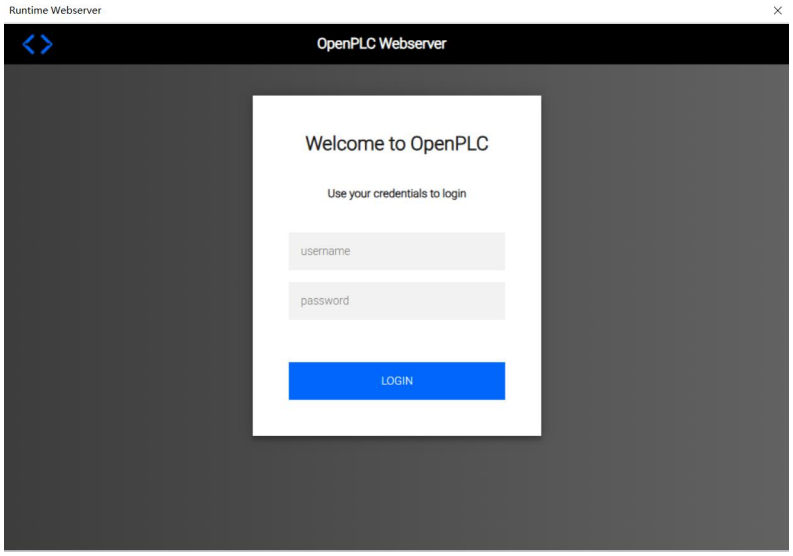


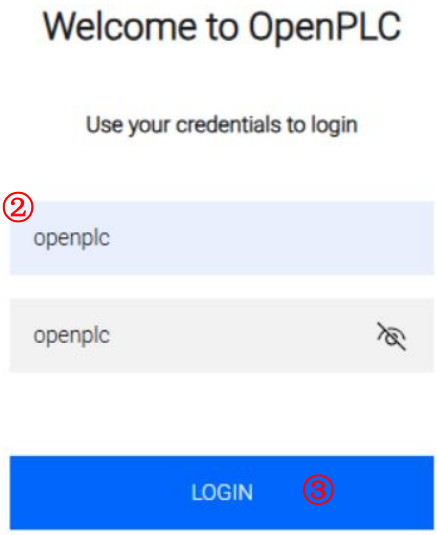
2.3.3 Burning

Click the menu toolbar "Runtime server" -> Enter DPLC's IP -> Enter username and password. Default username and password are both openplc -> Click LOGIN.

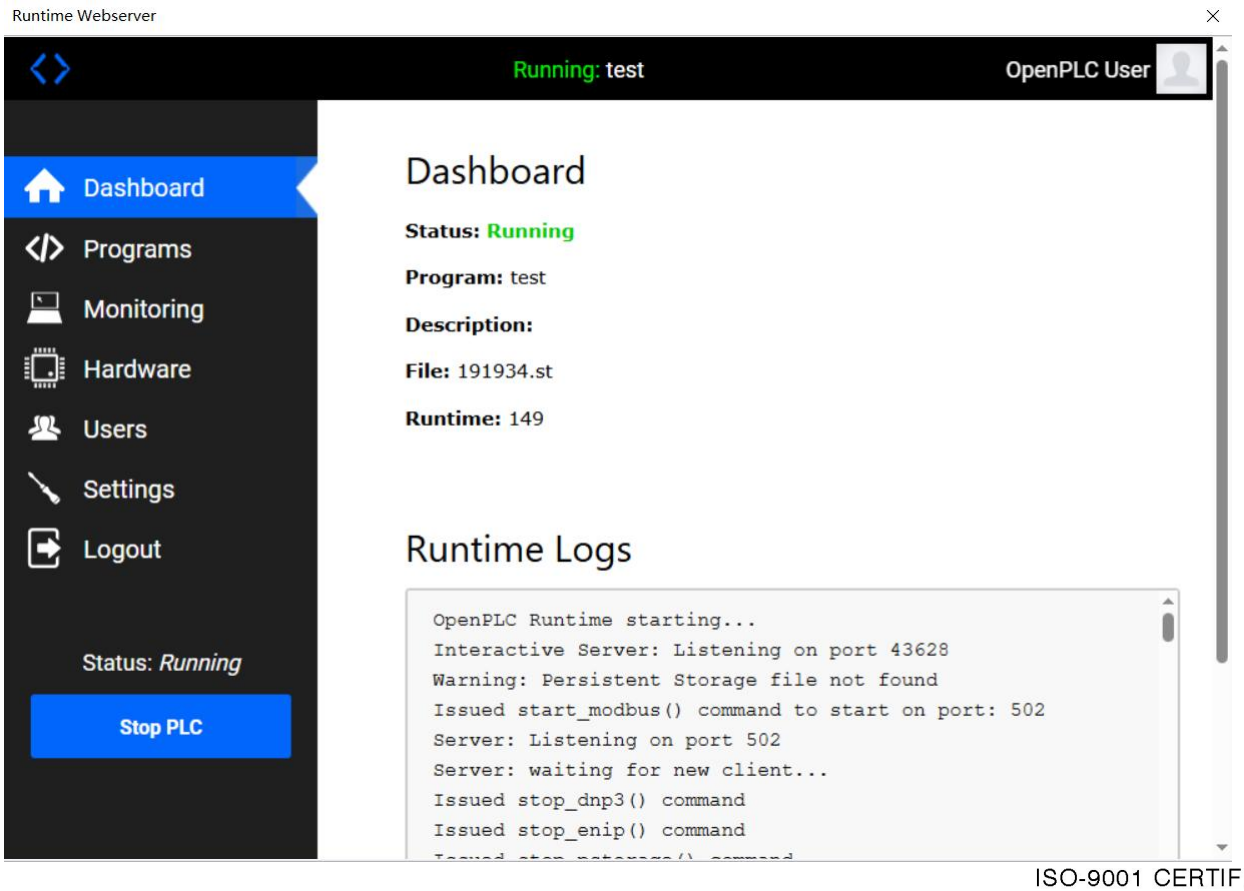
Installation Manual





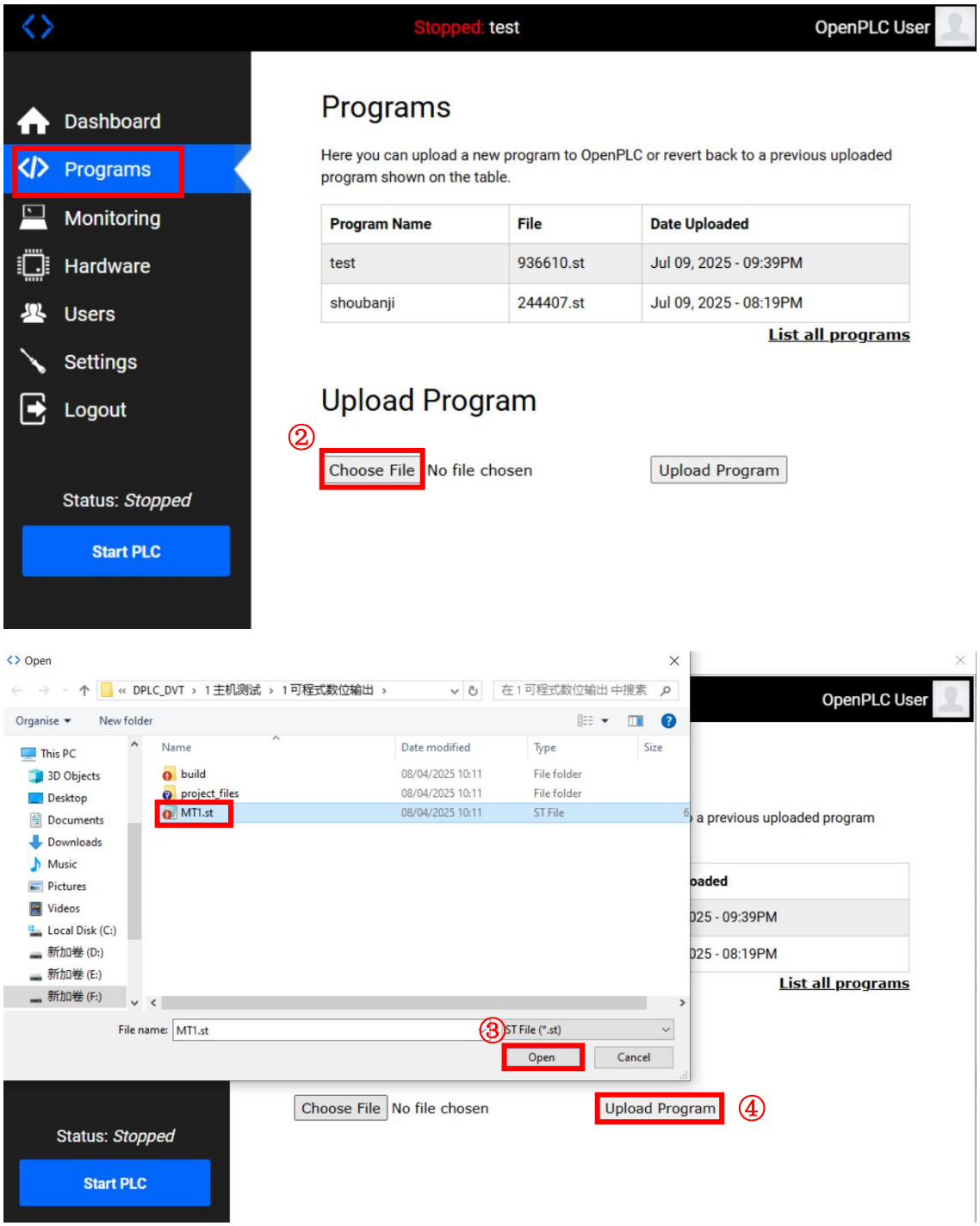


The interface entered is as shown below:

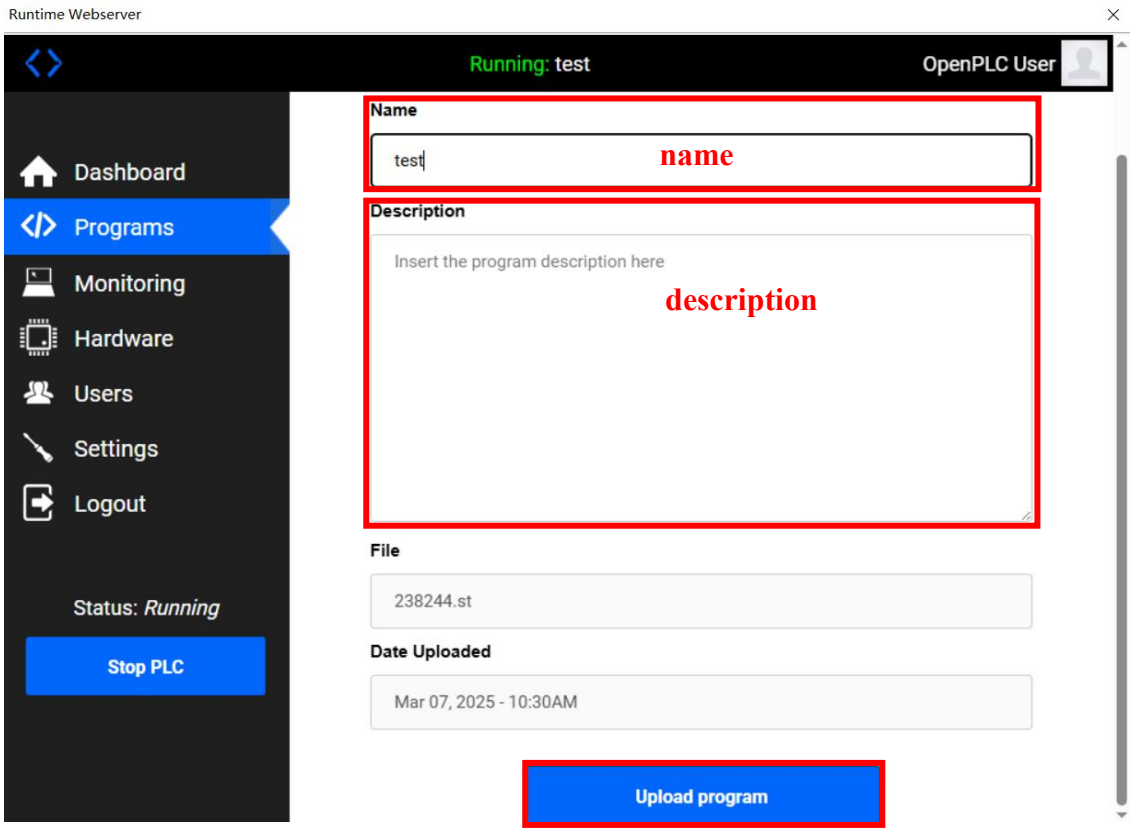


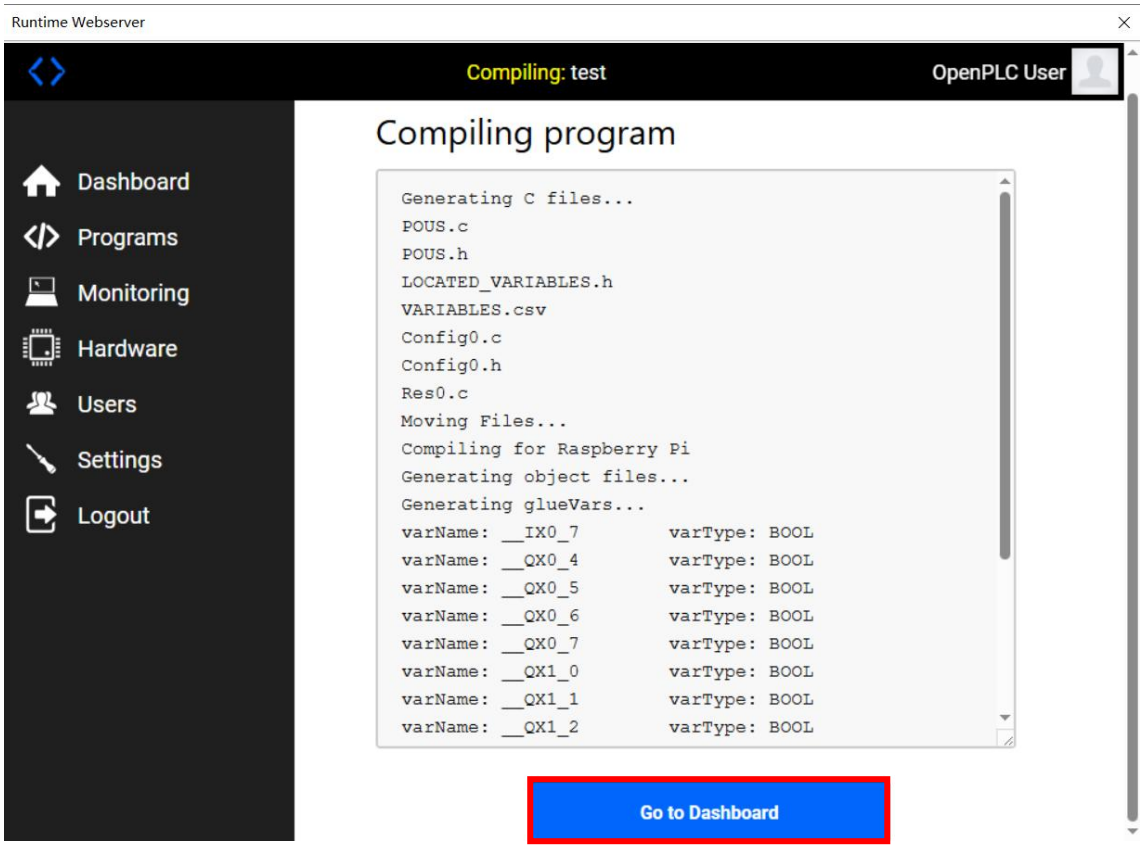
Installation Manual

Click "Programs" -> Select file -> Select the .st file generated above -> Click "Upload Program" to complete file upload.

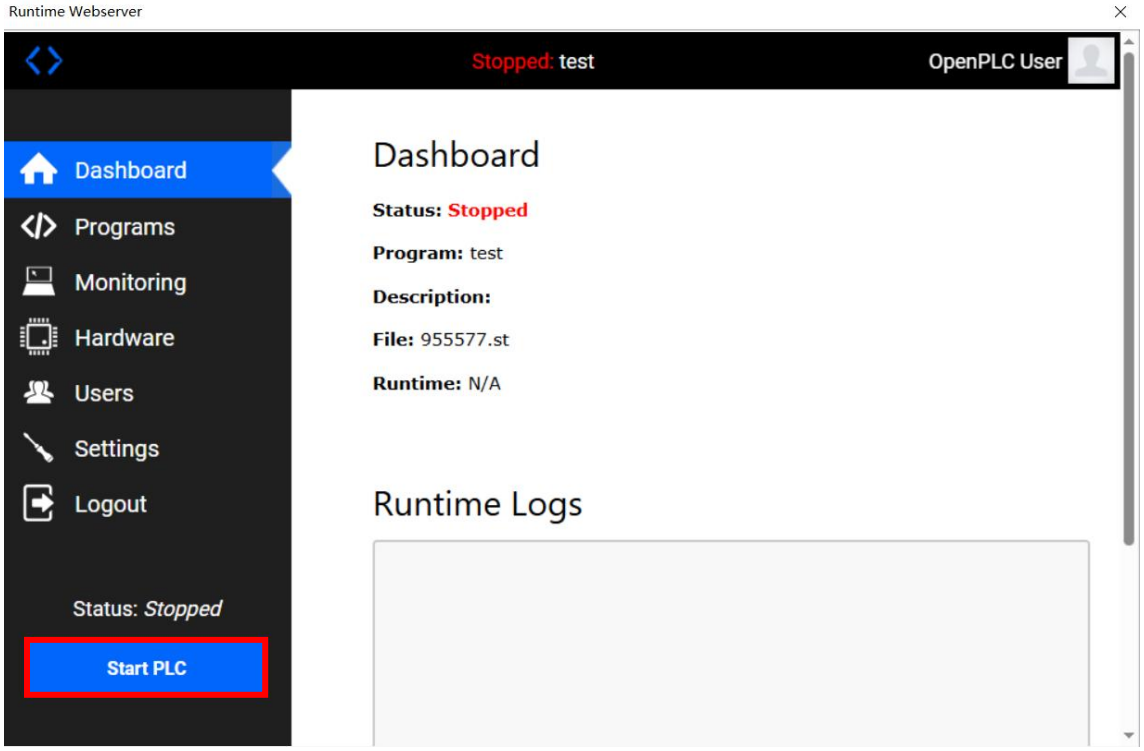


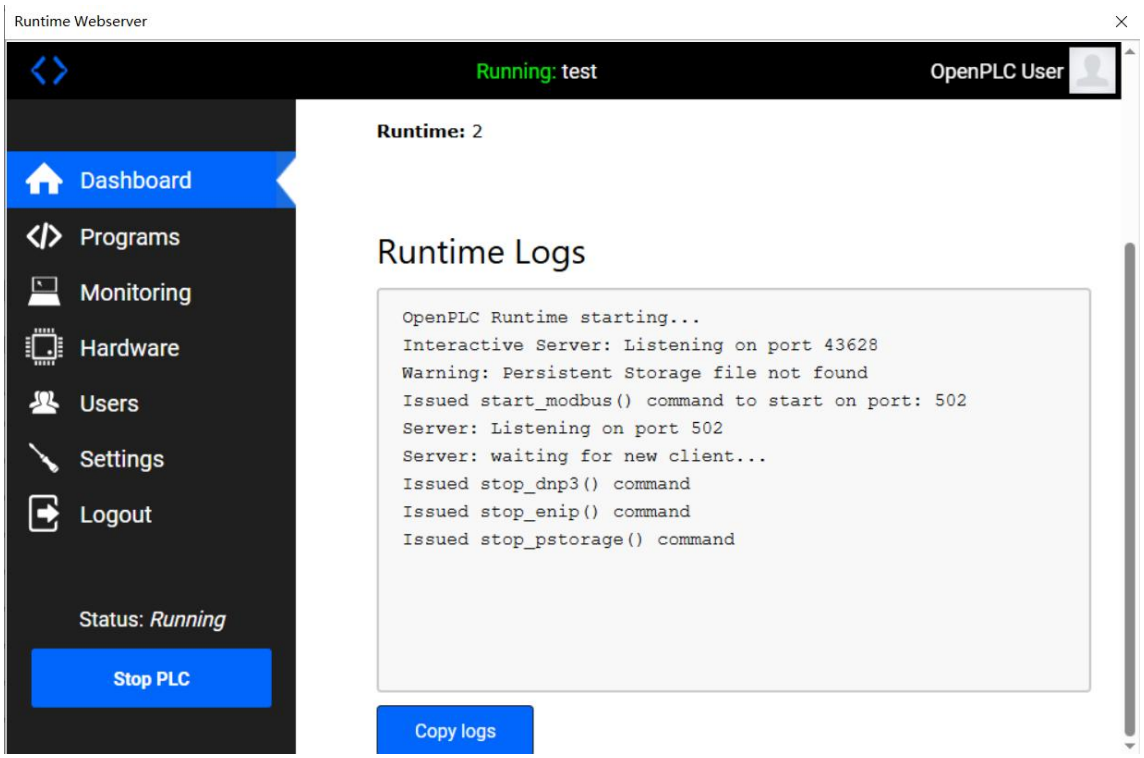
Rename the project, you can also add a description -> Click "Upload program" and wait for loading to complete -> Click "Go to Dashboard".





Click "Start PLC", the program starts running.

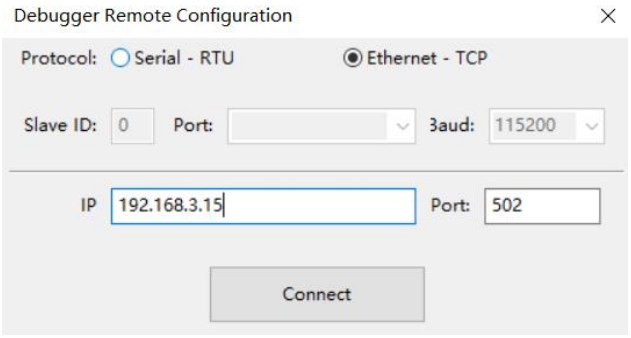




2.3.4 Debugging

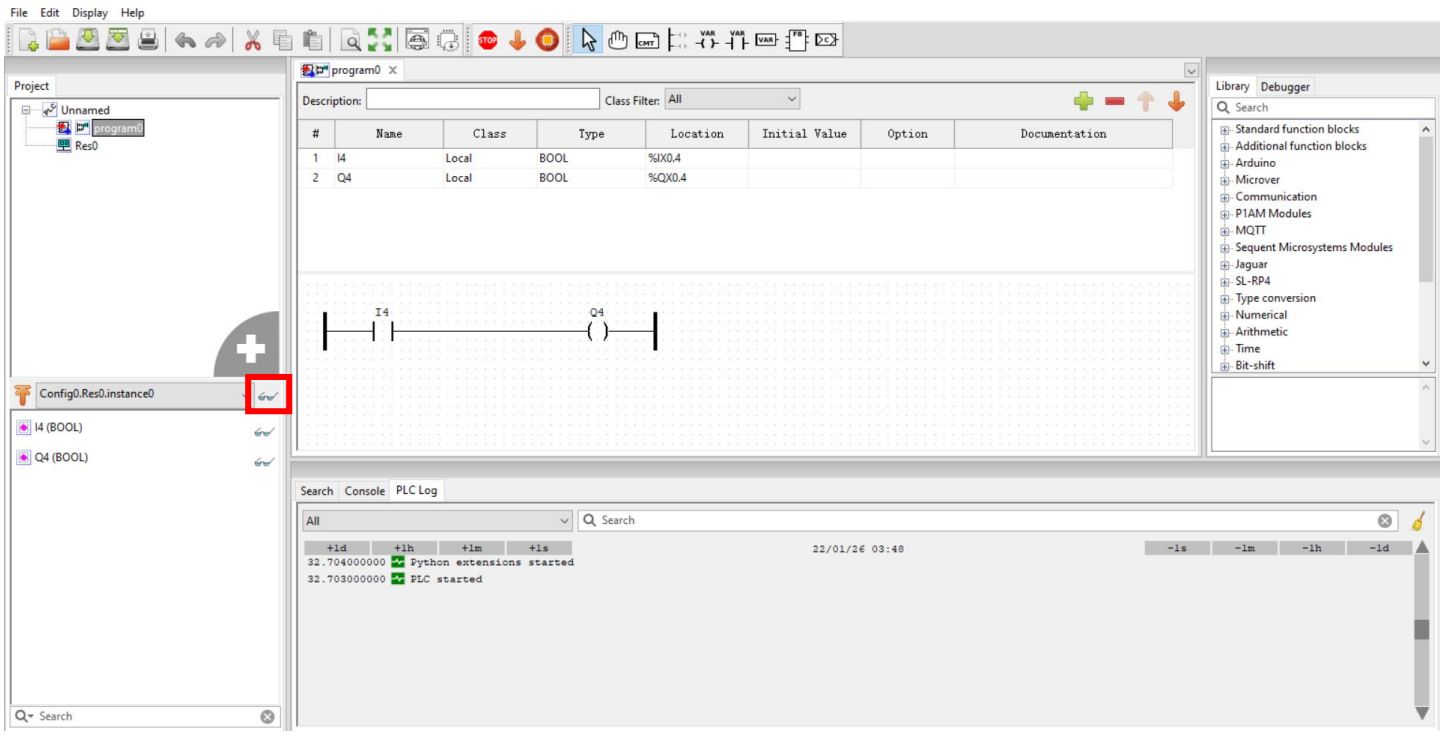
Click "Live debug remote PLC" in the menu toolbar -> Enter DPLC's IP in the popped-up window, then click "Connect".

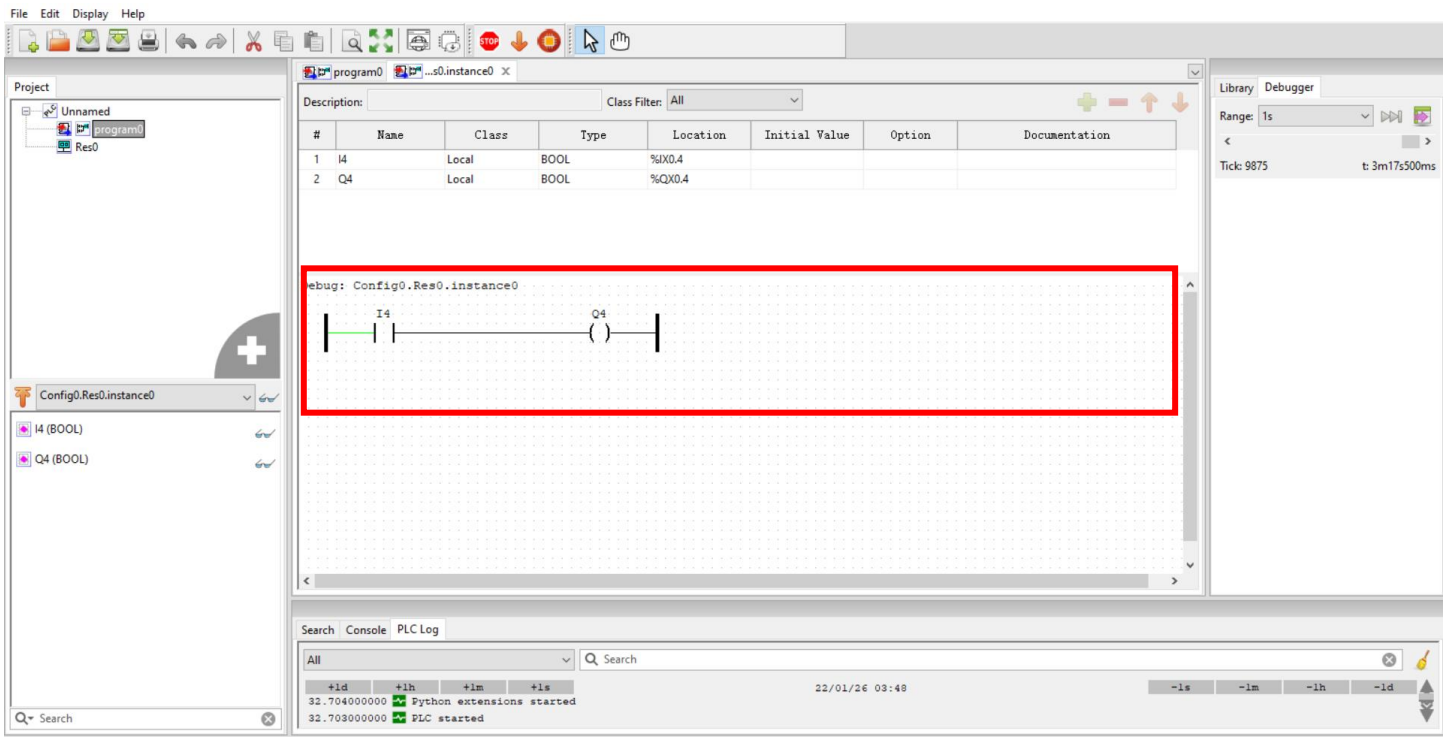
Installation Manual



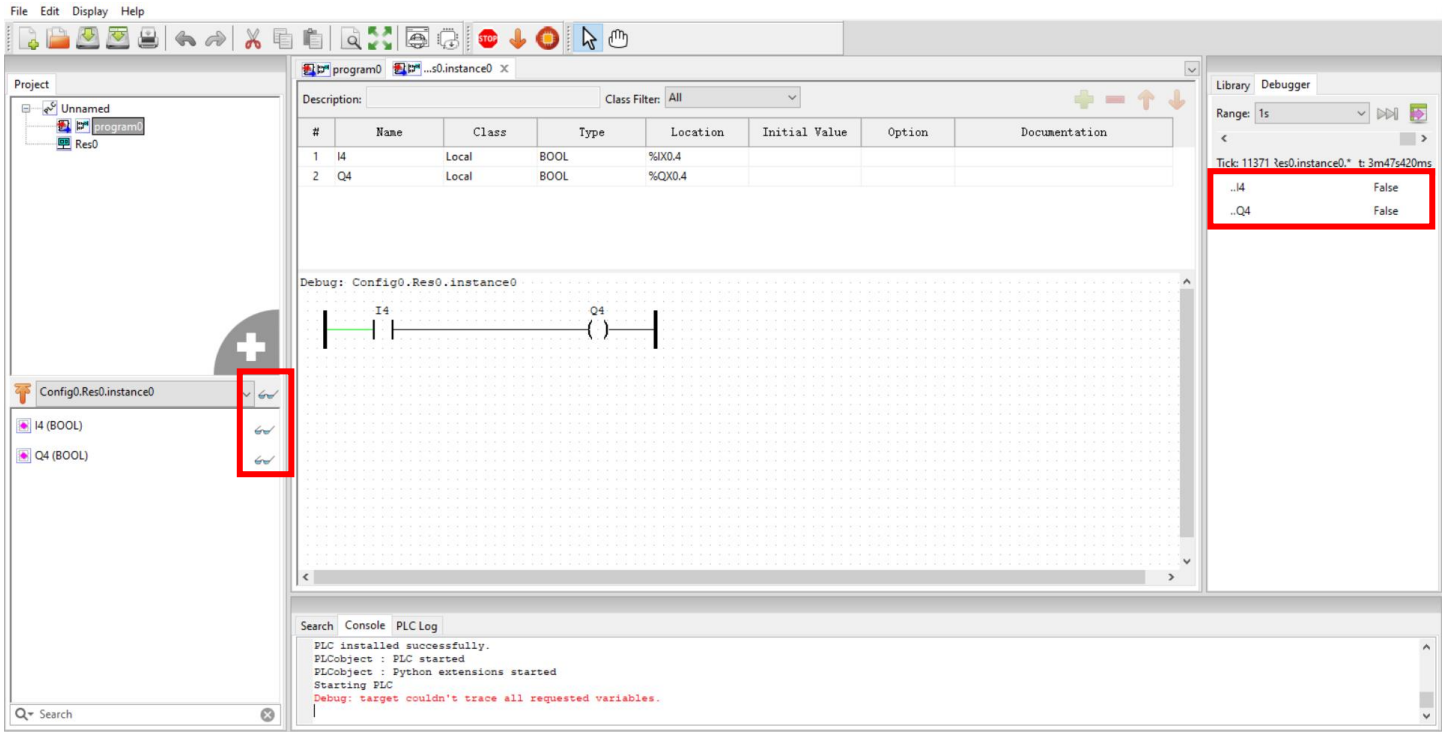
After Connect, the message window displays as follows:

Click "Debug Instance" in the debug window to start debugging.



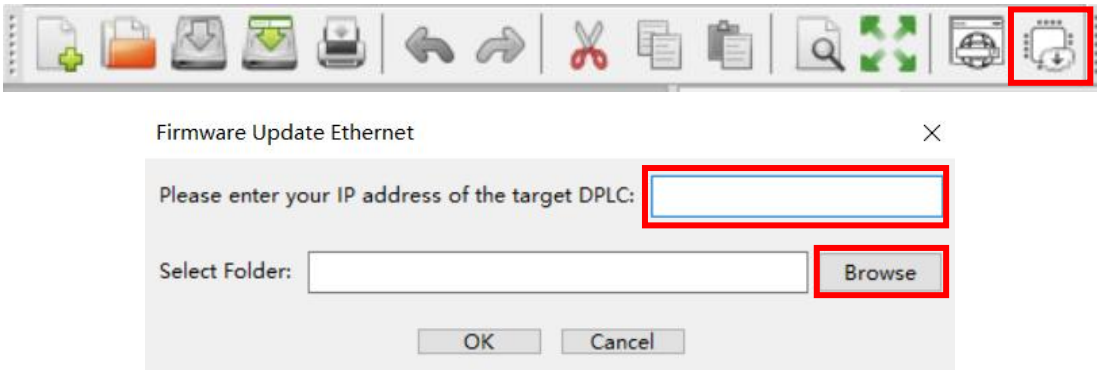


Click "Debug" for each variable, and the status of each variable can be monitored in the debugger on the right.

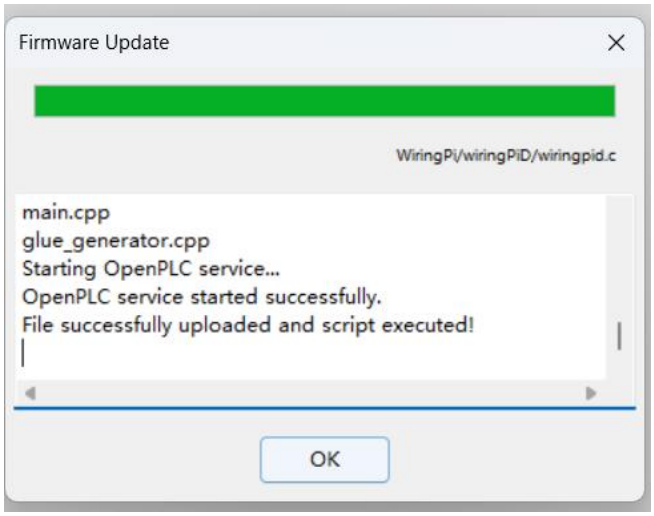


2.3.5 Firmware Update Ethernet

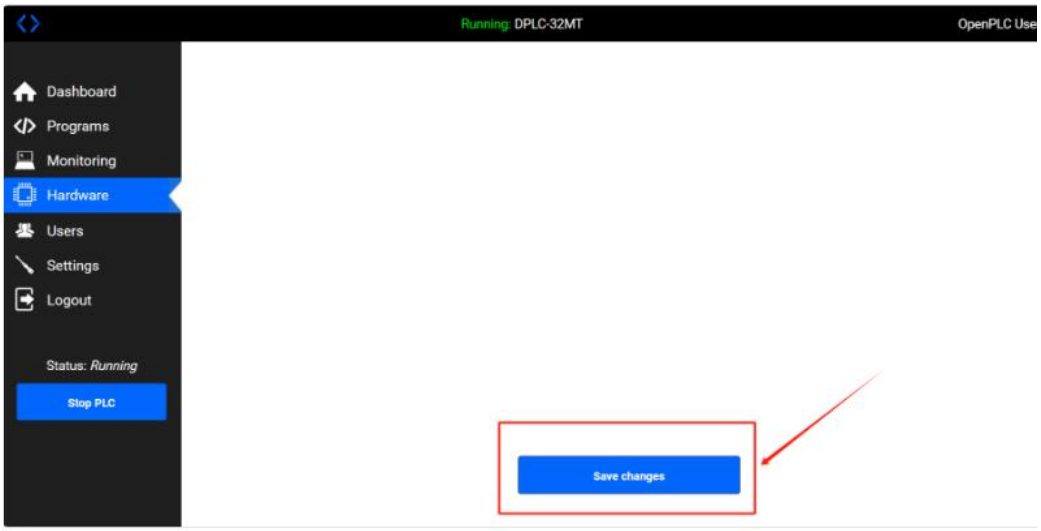
Click "Firmware Update Ethernet" in the menu toolbar -> In the popped-up window, enter DPLC's IP, upload the firmware file to be updated -> Click OK.



The interface after uploading is shown below "File successfully uploaded and script executed!"



Enter the port 80 webpage, click Hardware hardware layer selection, then click Save changes to compile, completing the firmware update.



3、Programming Operations

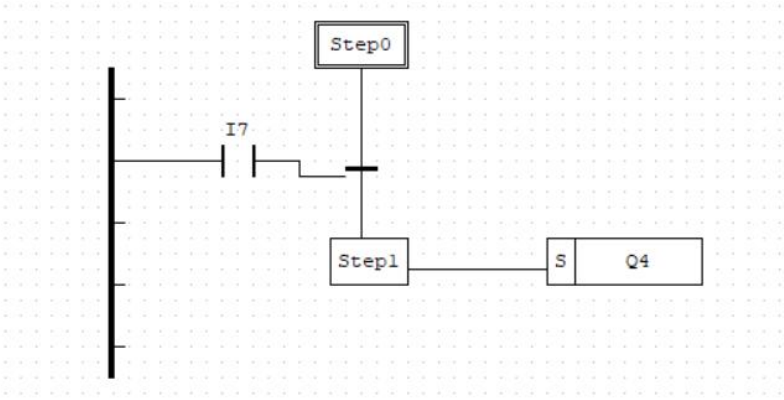
3.1 Programming Methods

Common programming methods for OpenPLC Editor: Ladder Diagram (LD), Structured Text (ST), and Sequential Function Chart (SFC).

- Intuitive and convenient, it is the method chosen by most PLC programmers and maintenance personnel.
- A PLC programming method similar to high-level programming languages. Using structured text programming makes it easier to complete complex algorithm control.
- It is a graphical functional description language suitable for expressing complex control processes.

  The following are expressions of the same programming logic in three languages:

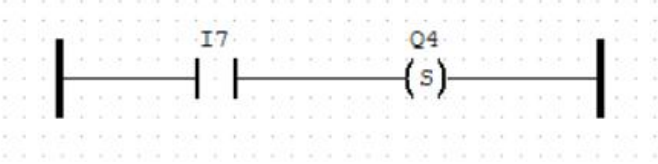
（1）SFC:



（2）ST:

```
1  IF I7 := TRUE THEN
2
3    Q4 := TRUE; (*set*)
4
5  END_IF;
```

（3）LD



Note: The programming data format is decimal.

3.2 Editing Ladder Diagrams

3.2.1 Address mapping

（1）Input mapping

#	Name	Class	Type	Location	Initial Value
1	sensor0	Local	BOOL	%IX0.0	
2	sensor1	Local	BOOL	%IX0.1	
3	IW0	Local	UINT	%IW0	

Name: Input device name (the name must be in full English).

Class: Default selection is Local, no need to change.

Type: Data type, the data type for high and low level input is BOOL, and the data type for analog input is UINT.

Location: Mapping address, the address %IX0.0 for high and low level input corresponds to I0.0, %IX0.1 corresponds to I0.1, and so on; the address %IW0 for analog input corresponds to IW0, %IW1 corresponds to IW1, and so on.

Initial Value: Initial value, you can set it yourself if necessary.

（2）Output mapping

#	Name	Class	Type	Location	Initial Value
4	LED0	Local	BOOL	%QX0.0	
5	LED1	Local	BOOL	%QX0.1	

Name: Output device with a name (the name should be in English).

Class: The default choice is Local, no need to change.

Type: Data type, the output data type is BOOL.

Location: Mapping address, %QX0.0 corresponds to Q0.0, %QX0.1 corresponds to Q0.1, and so on.

Initial Value: Initial value, it can be set by yourself if necessary.

（3）Relay mapping

#	Name	Class	Type	Location	Initial Value
6	MB0	Local	BYTE	%MB0	
7	MX0	Local	BOOL	%MX0	
8	MW0	Local	INT	%MW0	
9	MD0	Local	DINT	%MD0	

Name: Named Relay (The name should be in full English).

Class: Default selection is Local, no need to change.

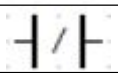
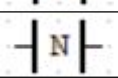
Type: Data type, MB data type is Byte, MX data type is BOOL, MW data type is UINT/INT, MD data type is UDINT/DINT.

Location: Mapping address, %MB0 corresponds to MB0, %MB1 corresponds to MB1, and so on; %MX0 corresponds to MX0, %MX1 corresponds to MX1, and so on; %MW0 corresponds to MW0, %MW1 corresponds to MW1, and so on; %MD0 corresponds to MD0, %MD1 corresponds to MD1, and so on.

Initial Value: Initial value, can be set by yourself if necessary.

3.2.2 Introduction to Soft Components

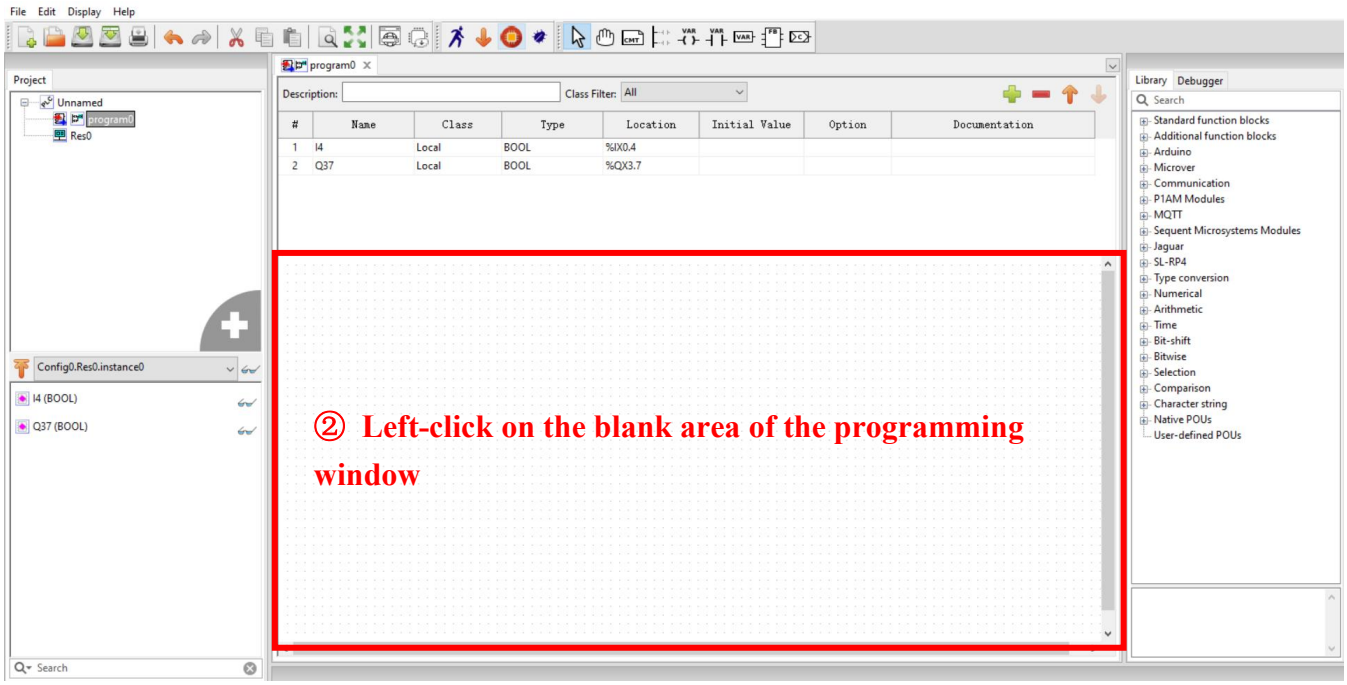
（1）Input Contacts

Icon	Function
	Normally Open Contact
	Normally Closed Contact
	Rising Edge Contact
	Falling Edge Contact

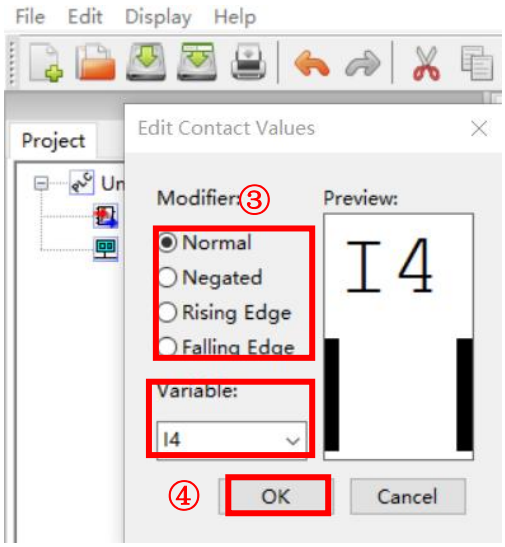
Example of Inserting Contact:

Click "New Contact" on the menu toolbar.

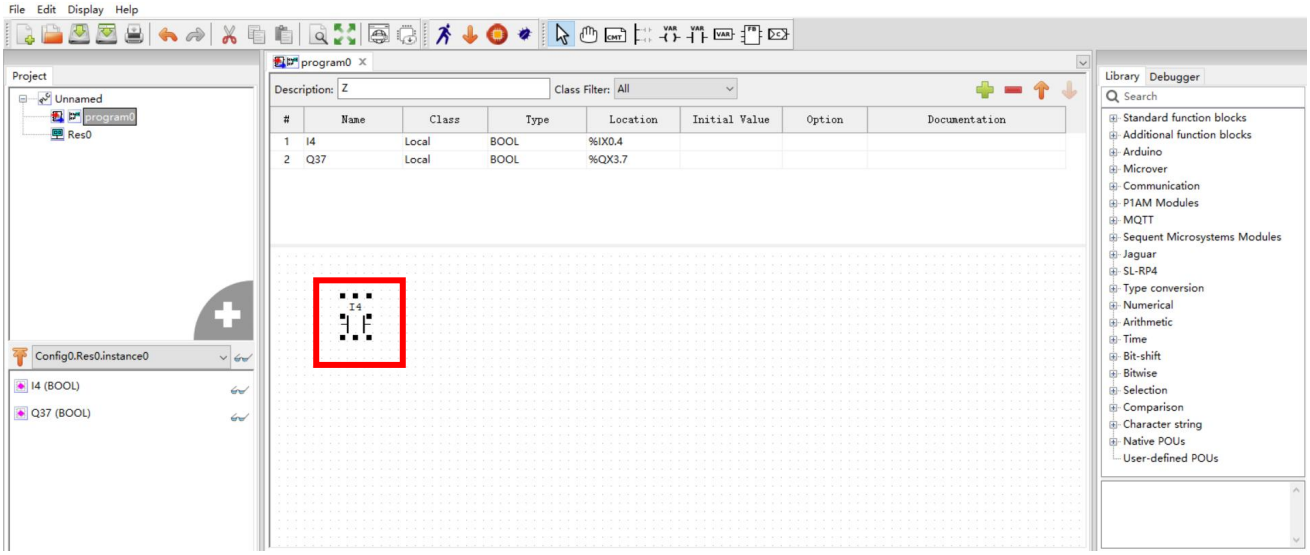
Then click anywhere in the blank space of the programming window.



At this time, the input contact selection window pops up. Select the input contact and map it to the corresponding variable. After selection, click OK.



The newly created contact appears in the programming window:



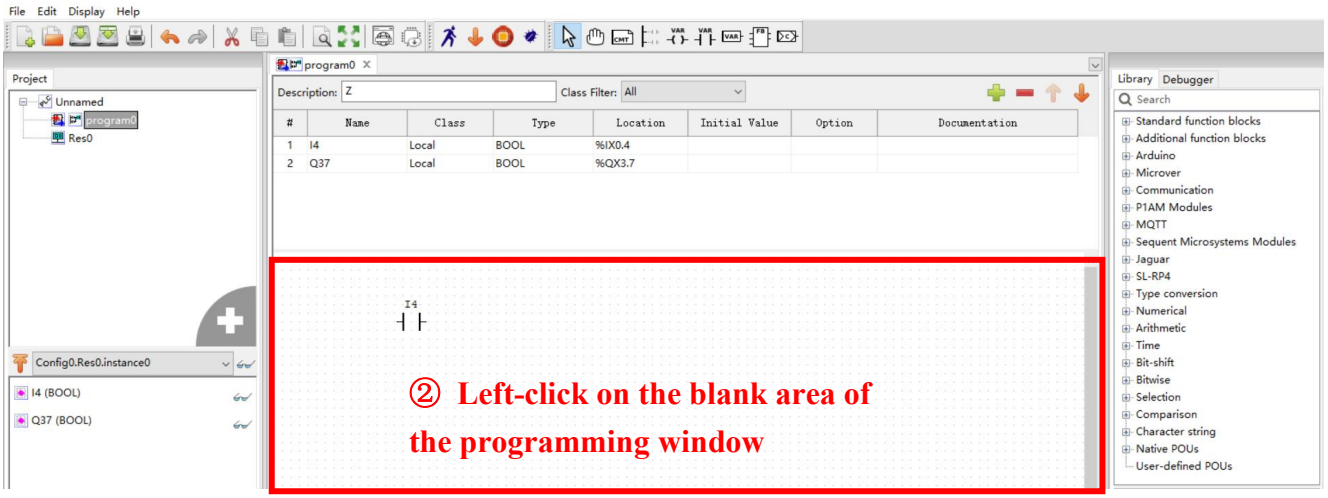
（2）Output Coils:

Icon	Function
-( )-	Assignment Coil
-(/)-	Negated Assignment Coil
-(S)-	Set Coil
-(R)-	Reset Coil
-(P)-	Rising Edge Coil
-(N)-	Falling Edge Coil

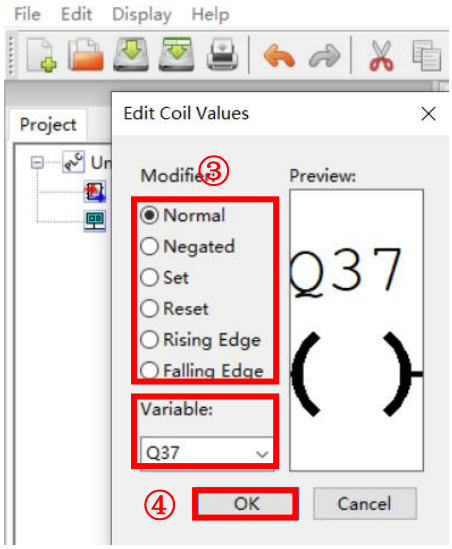
Example of Inserting Coil

Click "New Coil" on the menu toolbar.

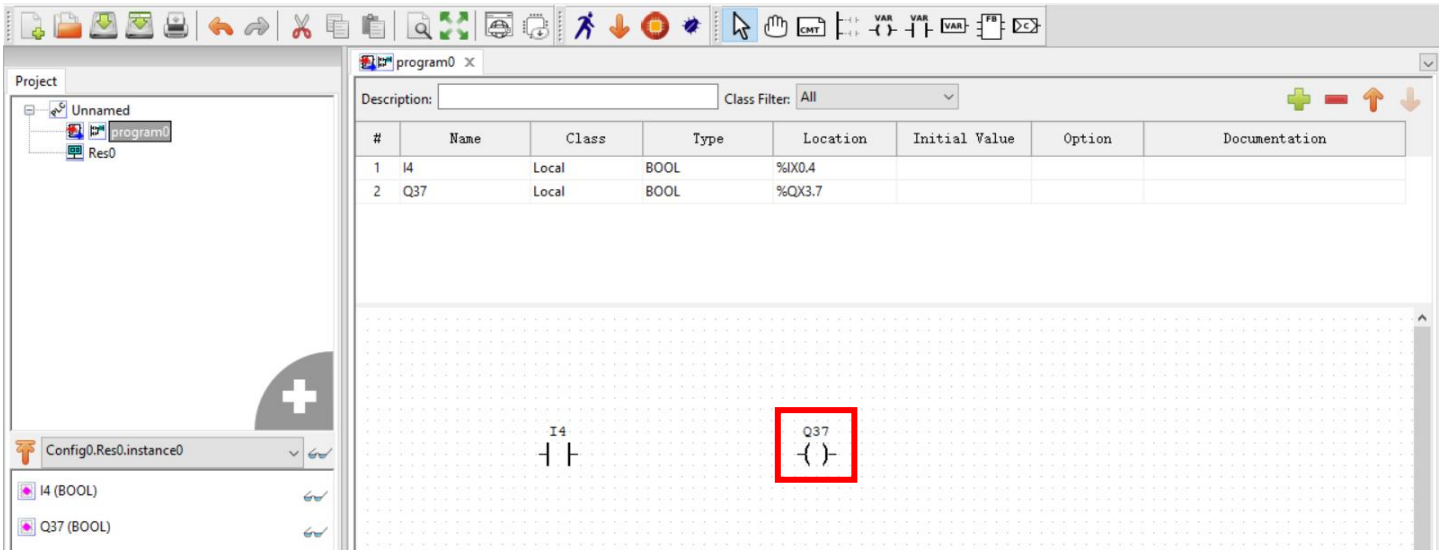
Then click anywhere in the blank space of the programming window.



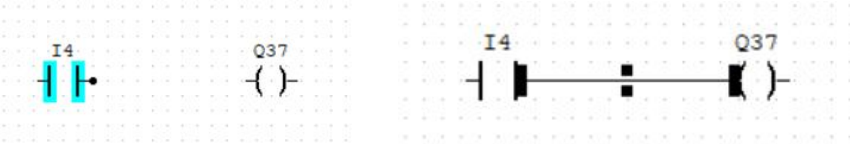
At this time, the output coil selection window pops up. Select the output coil and map it to the corresponding variable. After selection, click OK.



The newly created coil appears in the programming window:



Drag the rightmost side of the input contact to connect the two.



（3）Power Rails

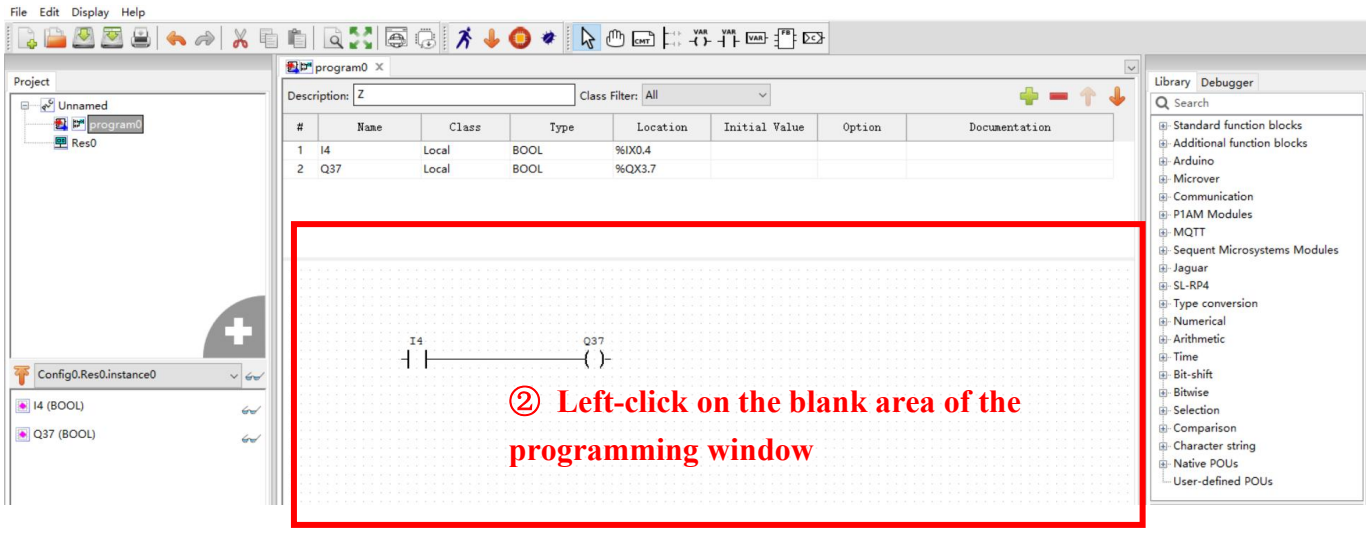
Icon	Function
	Left Power Rail
	Right Power Rail

Example of Inserting Rail:

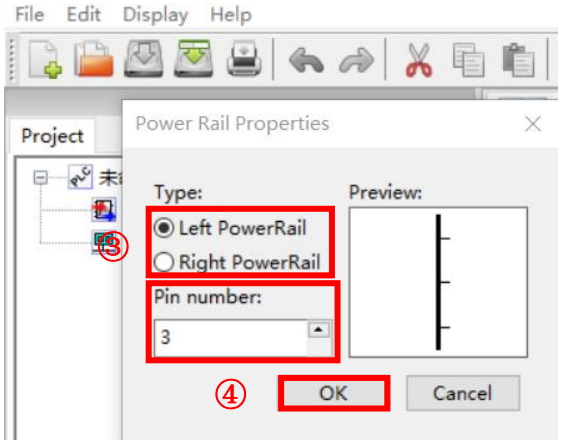
Click "New Power Rail" on the menu toolbar.

Then click anywhere in the blank space of the programming window.

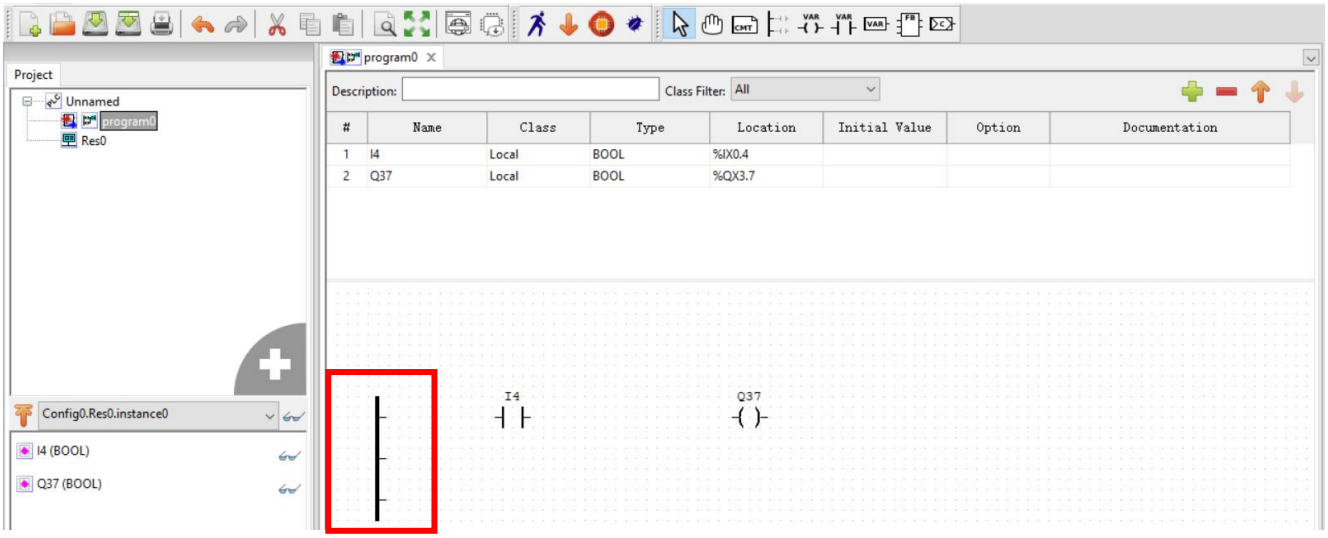
Installation Manual



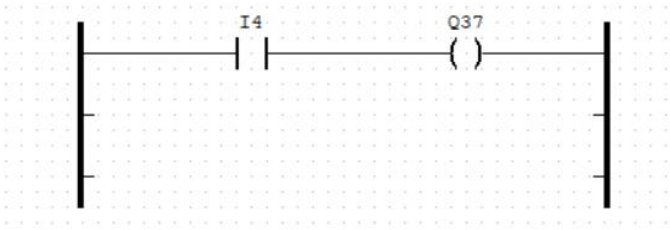
At this time, the power rail selection window pops up. Select the left power rail and select the number of pins. After selection, click OK.



The newly created power rail appears in the programming window.



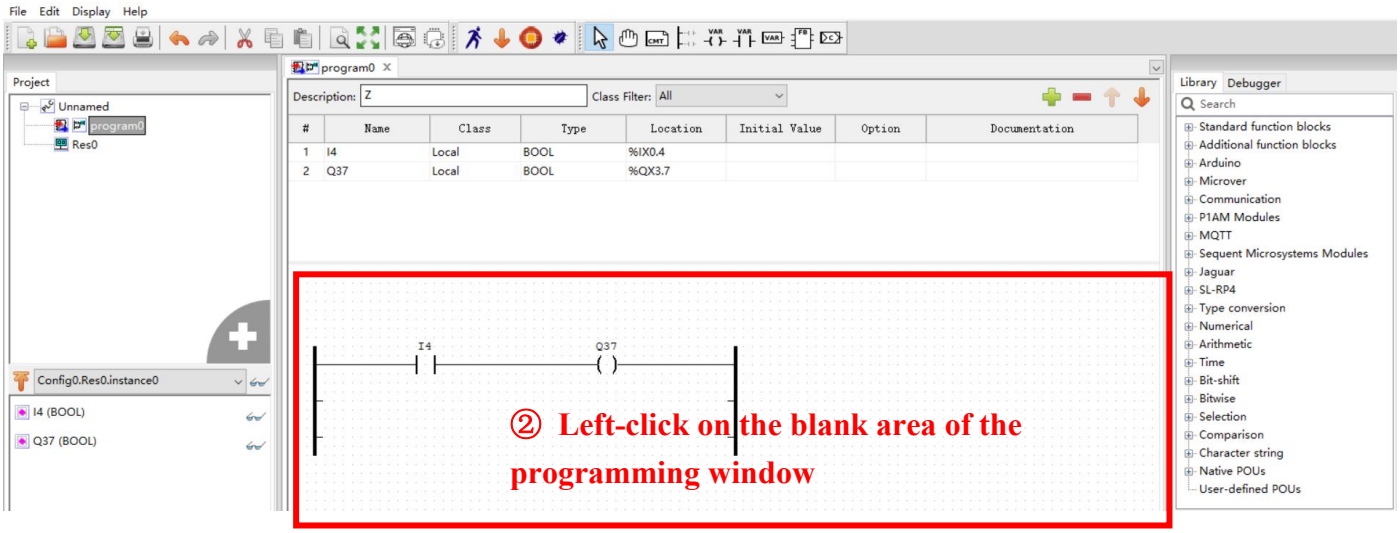
Similarly, add the right power rail. Then connect the left power rail, input contact, output coil, and right power rail end-to-end.



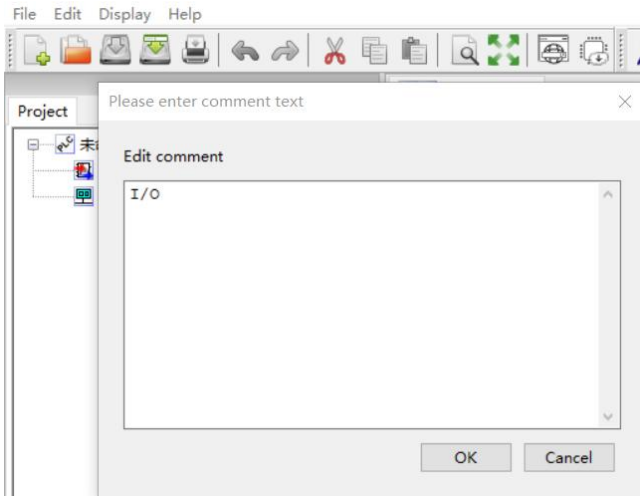
（4） Comments

Click "New Comment" on the menu toolbar.

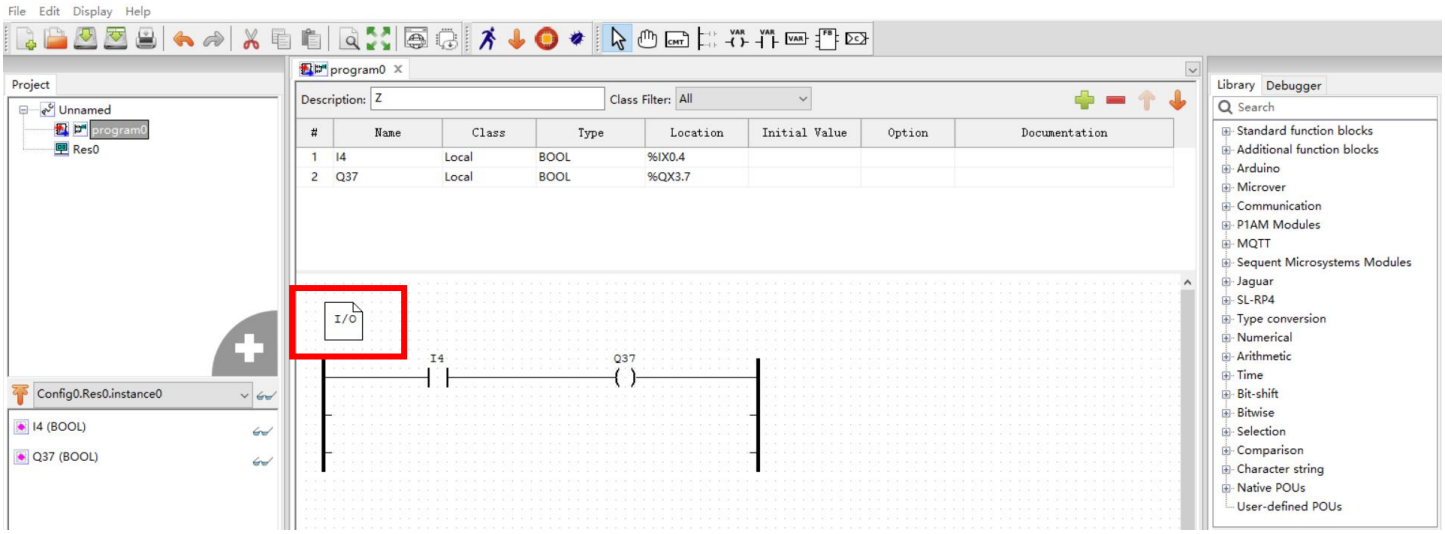
Then click anywhere in the blank space of the programming window.



At this time, the comment text window pops up. After editing the comment, click OK.

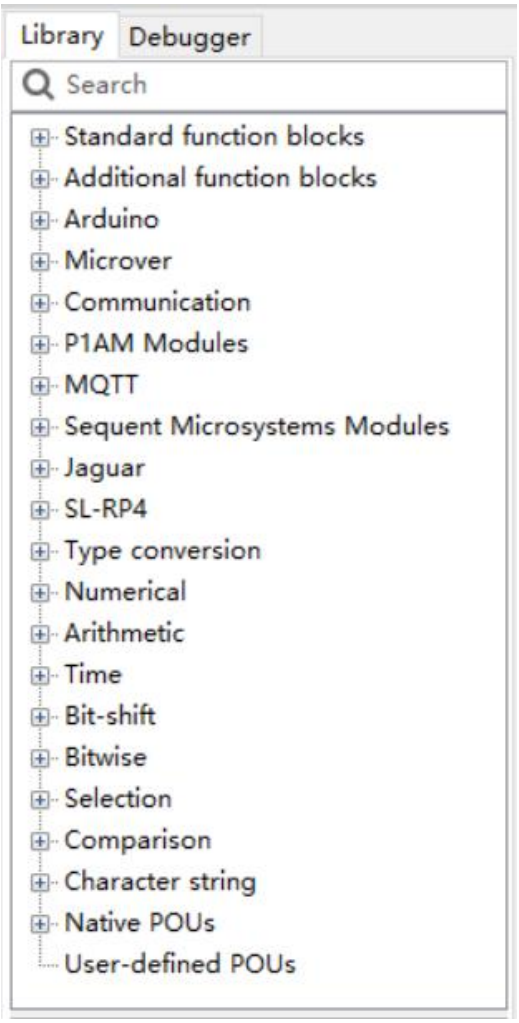


The added comment appears in the programming window; drag the comment to a suitable position.



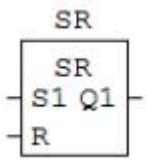
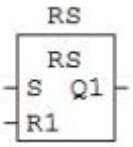
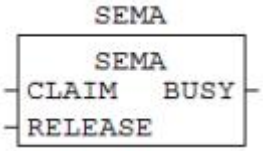
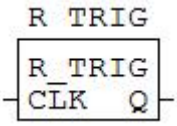
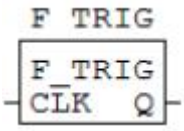
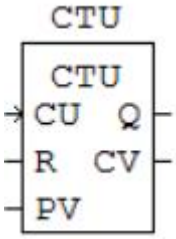
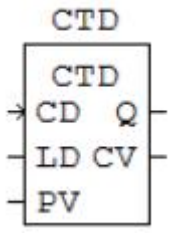
3.2.3 Introduction to Function Blocks

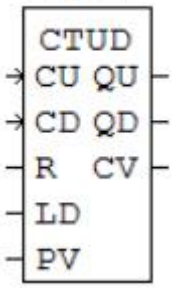
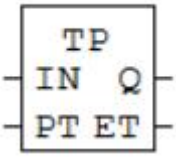
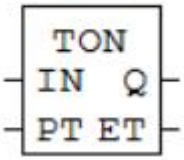
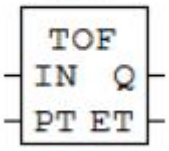
The library contains a variety of standard and user-defined function blocks, which support use in different programming languages.



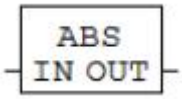
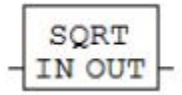
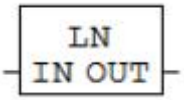
Next, standard function blocks and commonly used function blocks will be explained in detail.

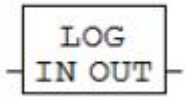
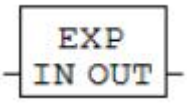
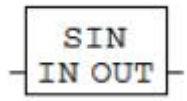
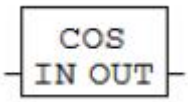
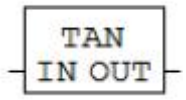
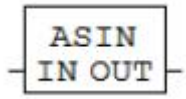
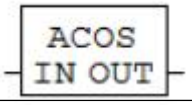
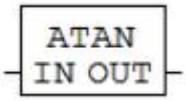
（1） Standard Function Blocks

Name	Icon	Function
SR		SR bistable is a set-dominant latch. This function is a standard set-dominant flip-flop. When input S1 is TRUE and R is FALSE, Q1 output becomes TRUE. Similarly, when input S1 is FALSE and R is TRUE, Q1 output becomes FALSE. After one of the transitions, when both S1 and R signals return to FALSE, Q1 output maintains the previous state until a new condition appears. If both signals are TRUE, Q1 output is forced to TRUE (Set dominant).
RS		RS bistable is a reset-dominant latch. This function is a standard reset-dominant flip-flop. When input S is TRUE and R1 input is FALSE, Q1 output is TRUE. Similarly, when input S is FALSE and R1 input is TRUE, Q1 output is FALSE. After one of the transitions, when both S and R1 signals return to FALSE, Q1 output maintains the previous state until a new condition appears. If both signals are TRUE, Q1 output is forced to FALSE (Reset dominant).
SEMA		Semaphores provide a mechanism that allows software elements to access a resource mutually exclusively. This function block implements a semaphore function. Normally, this function is used to synchronize events. The BUSY output is activated by a TRUE condition on the CLAIM input and deactivated by a TRUE condition on the RELEASE input.
R_TRIG		When a rising edge is detected, the output generates a single pulse. This function is a rising edge detector. When a 0 to 1 (or FALSE to TRUE or OFF to ON) condition is detected on the CLK input, the Q output becomes TRUE and remains in this state throughout the scan cycle.
F_TRIG		When a falling edge is detected, the output generates a single pulse. This function is a falling edge detector. When a 1 to 0 (or TRUE to FALSE or ON to OFF) condition is detected on the CLK input, the Q output becomes TRUE and remains in this state throughout the scan cycle.
CTU CTU_DINT, CTU_LINT, CTU_UDINT, CTU_ULINT		The up-counter can be used to generate a signal when the count value reaches the maximum value. CTU: (BOOL:CU, BOOL:R, INT:PV)=>(BOOL:Q, INT:CV) CTU_DINT: (BOOL:CU, BOOL:R, DINT:PV) => (BOOL:Q, DINT:CV) CTU_LINT: (BOOL:CU, BOOL:R, LINT:PV) => (BOOL:Q, LINT:CV) CTU_UDINT: (BOOL:CU, BOOL:R, UDINT:PV) => (BOOL:Q, UDINT:CV) CTU_ULINT: (BOOL:CU, BOOL:R, ULINT:PV) => (BOOL:Q, ULINT:CV) This function represents an up-counter. A rising edge on the CU input increments the counter by 1. When the preset value of PV is reached, the Q output becomes TRUE. Applying a TRUE signal on the R input resets the counter to zero (asynchronous reset). CV outputs the current count value.
CTD CTD_DINT, CTD_LINT, CTD_UDINT, CTD_ULINT		The down-counter can be used to generate a signal when the count value decreases from a preset value to 0. CTD: (BOOL:CD, BOOL:LD, INT:PV)=>(BOOL:Q, INT:CV) CTD_DINT: (BOOL:CD, BOOL:LD, DINT:PV) => (BOOL:Q, DINT:CV) CTD_LINT: (BOOL:CD, BOOL:LD, LINT:PV) => (BOOL:Q, LINT:CV) CTD_UDINT: (BOOL:CD, BOOL:LD, UDINT:PV) => (BOOL:Q, UDINT:CV) CTD_ULINT: (BOOL:CD, BOOL:LD, ULINT:PV) => (BOOL:Q, ULINT:CV) This function represents a down-counter. A rising edge on the CD input decrements the counter by 1. When the count value is equal to or less than zero, the Q output becomes TRUE. Applying a TRUE signal on the LD input loads the value of the PV input into the counter. CV outputs the current count value.

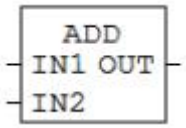
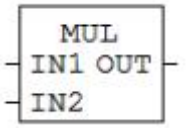
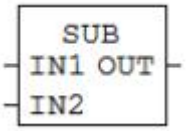
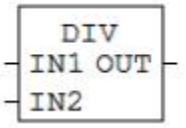
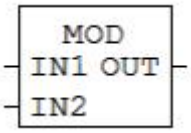
CTUD CTUD_DINT, CTUD_LINT, CTUD_UDINT, CTUD_ULINT	CTUD 	The up-down counter has 2 inputs, CU and CD, which can be used to count up on one input and count down on the other simultaneously. CTUD: (BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, INT:PV) => (BOOL:QU, BOOL:QD, INT:CV) CTUD_DINT: (BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, DINT:PV) => (BOOL:QU, BOOL:QD, DINT:CV) CTUD_LINT: (BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, LINT:PV) => (BOOL:QU, BOOL:QD, LINT:CV) CTUD_UDINT: (BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, UDINT:PV) => (BOOL:QU, BOOL:QD, UDINT:CV) CTUD_ULINT: (BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, ULINT:PV) => (BOOL:QU, BOOL:QD, ULINT:CV) This function represents a counter capable of counting up and down. A rising edge on the CU (Count Up) input increments the counter by 1, while a rising edge on the CD (Count Down) input decrements the counter by 1. Applying a TRUE signal on the R input resets the counter to zero. A TRUE condition on the LD signal loads the value applied to input PV (programmed value) into the counter. When the count value is greater than or equal to PV, QU outputs TRUE. When the count value is less than or equal to zero, QD outputs TRUE. CV outputs the current count value.
TP	TP 	The pulse timer can be used to generate output pulses of a given duration. (BOOL:IN, TIME:PT)=>(BOOL:Q, TIME:ET) This timer has the same behavior as a one-shot timer or monostable timer. When a rising edge is detected at the input, the Q output is TRUE. This condition continues until the programmed time PT applied to the relevant pin ends. After PT ends, if input IN is still valid, output Q remains in ON state, otherwise output Q returns to OFF state. This timer is not re-triggerable. This means that once the timer starts, it cannot be stopped until the entire session ends. ET outputs the current elapsed time.
TON	TON 	The On-Delay Timer can be used to delay setting an output to TRUE after a fixed time when the input is TRUE. (BOOL:IN, TIME:PT)=>(BOOL:Q, TIME:ET) Asserting the input signal IN of this function starts the timer. When the programmed time of the preset input PT elapses and input IN is still valid, the Q output becomes TRUE. This condition continues until input IN is released. If the IN input is released before the time elapses, the timer is cleared. ET outputs the current elapsed time.
TOF	TOF 	The Off-Delay Timer can be used to delay setting an output to FALSE after a fixed time has passed since the input went to FALSE (BOOL:IN, TIME:PT)=>(BOOL:Q, TIME:ET) The input signal IN of this function immediately activates the Q output. At this time, releasing the input starts the time elapse. When the programmed time of the preset input PT elapses while input IN is still released, the Q output becomes FALSE. This state remains until the input is released. If the IN input is released before the time elapses, the timer is cleared and the Q output remains TRUE. ET outputs the current elapsed time.

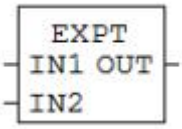
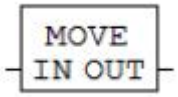
(2) Mathematical Function Blocks

Name	Icon	Function
ABS		Absolute Value (ANY_NUM:IN) => (ANY_NUM:OUT) ST syntax example: OUT := ABS(IN1);
SQRT		Square Root (Base 2) (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := SQRT(IN1);
LN		Natural Logarithm (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := LN(IN1);

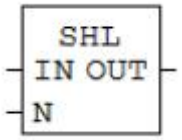
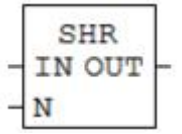
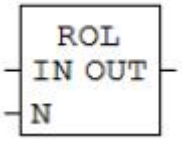
LOG		Logarithm Base 10 (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := LOG(IN1);
EXP		Exponent (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := EXP(IN1);
SIN		Sine (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := SIN(IN1);
COS		Cosine (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := COS(IN1);
TAN		Tangent (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := TAN(IN1);
ASIN		Arc Sine (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := ASIN(IN1);
ACOS		Arc Cosine (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := ACOS(IN1);
ATAN		Arc Tangent (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := ATAN(IN1);

（3） Arithmetic Function Blocks

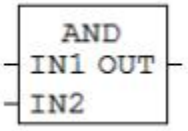
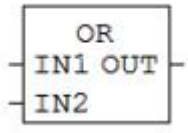
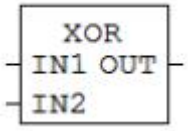
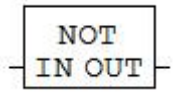
Name	Icon	Function
ADD		Addition (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 + IN2. The input quantity can be expanded. ST syntax example: OUT := IN1 + IN2;
MUL		Multiplication (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 * IN2. The input quantity can be expanded. ST syntax example: OUT := IN1 * IN2;
SUB		Subtraction (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 – IN2. ST syntax example: OUT := IN1 - IN2;
DIV		Division (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 / IN2. EX.: 1234 / 10 = 3. ST syntax example: OUT := IN1 / IN2;
MOD		Remainder/Modulo (ANY_INT:IN1, ANY_INT:IN2) => (ANY_INT:OUT) OUT = IN1 modulo IN2. EX.: 1234 modulo 10 = 4. ST syntax example: OUT := IN1 MOD IN2;

EXPT		Exponentiation (ANY_REAL:IN1, ANY_NUM:IN2) => (ANY_REAL:OUT) OUT = IN1 ^{IN2} EX.: $2^3 = 8$. ST syntax example: OUT := IN1 ** IN2;
MOVE		Assignment (ANY:IN) => (ANY:OUT) OUT = IN. ST syntax example: OUT := IN1;

(4) Shift Function Blocks

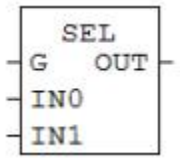
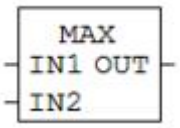
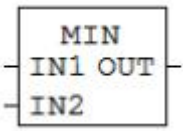
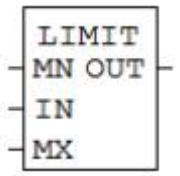
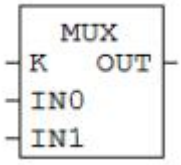
Name	Icon	Function
SHL		Shift Left (ANY_BIT:IN, ANY_INT:N) => (ANY_BIT:OUT) "OUT" represents the IN variable that is offset N bits to the left. Fill the right side of the OUT variable with zeros. ST syntax example: OUT := SHL(IN := IN1, N := IN2);
SHR		Shift Right (ANY_BIT:IN, ANY_INT:N) => (ANY_BIT:OUT) "OUT" represents the IN variable that is offset N bits to the right. Fill the left side of the OUT variable with zeros. ST syntax example: OUT := SHR(IN := IN1, N := IN2);
ROR		Rotate Right (ANY_NBIT:IN, ANY_INT:N) => (ANY_NBIT:OUT) "OUT" represents the IN variable that moves N bits to the right. The rightmost bit of each movement is filled to the leftmost bit of the OUT variable. ST syntax example: OUT := ROR(IN := IN1, N := IN2);
ROL		Rotate Left (ANY_NBIT:IN, ANY_INT:N) => (ANY_NBIT:OUT) "OUT" represents the IN variable that moves N bits to the left. Each time the leftmost bit is moved, it is filled to the rightmost bit of the OUT variable. ST syntax example: OUT := ROL(IN := IN1, N := IN2);

(5) Bitwise Operation Function Blocks

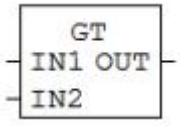
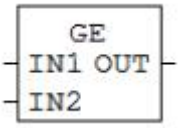
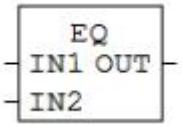
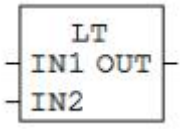
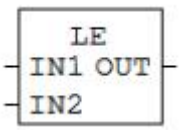
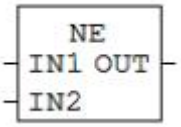
Name	Icon	Function
AND		Bitwise AND (ANY_BIT:IN1, ANY_BIT:IN2) => (ANY_BIT:OUT) OUT = IN1 AND IN2. The input quantity can be expanded. ST syntax example: OUT := IN1 AND IN2;
OR		Bitwise OR (ANY_BIT:IN1, ANY_BIT:IN2) => (ANY_BIT:OUT) OUT = IN1 OR IN2. The input quantity can be expanded. ST syntax example: OUT := IN1 OR IN2;
XOR		Bitwise Exclusive OR (ANY_BIT:IN1, ANY_BIT:IN2) => (ANY_BIT:OUT) OUT = IN1 EXCLUSIVE OR IN2. The input quantity can be expanded. ST syntax example: OUT := IN1 XOR IN2;
NOT		Bitwise NOT (ANY_BIT:IN) => (ANY_BIT:OUT) OUT = NOT IN. ST syntax example: OUT := IN1 NOT IN2;

(6) Selection Function Blocks

Name	Icon	Function

SEL		Binary Selection (1 of 2) (BOOL:G, ANY:IN0, ANY:IN1) => (ANY:OUT) If G is FALSE, then OUT will be consistent with the IN0 value. If G is TRUE, OUT will be consistent with the value of IN1.
MAX		Maximum Value (ANY:IN1, ANY:IN2) => (ANY:OUT) This function block compares the magnitudes of the values when IN1 and IN2 are input and outputs their maximum values. The input quantity can be expanded.
MIN		Minimum Value (ANY:IN1, ANY:IN2) => (ANY:OUT) This function block compares the magnitudes of the values when IN1 and IN2 are input and outputs their minimum values. The input quantity can be expanded.
LIMIT		Limit (ANY:MN, ANY:IN, ANY:MX) => (ANY:OUT) OUT is the value between the minimum value MN and the maximum value MX. If IN is less than the minimum value MN, OUT displays the value of MN. If IN is greater than the maximum MX value, OUT displays the MX value.
MUX		Multiplexer (1 of many) (ANY_INT:K, ANY:IN0, ANY:IN1) => (ANY:OUT) Depending on the K value, IN1, IN2... In Inn, the selected value is indicated, and OUT represents the selected input value. The input quantity can be expanded.

(7) Comparison Function Blocks

Name	Icon	Function
GT		Greater Than (ANY:IN1, ANY:IN2) => (BOOL:OUT) If IN1 is greater than IN2, OUT is TRUE; otherwise, OUT is FALSE. The input quantity can be expanded. ST syntax example: OUT := IN1 > IN2;
GE		Greater Than or Equal To (ANY:IN1, ANY:IN2) => (BOOL:OUT) If IN1 is greater than or equal to IN2, OUT is TRUE; otherwise, OUT is FALSE. The input quantity can be extended. ST syntax example: OUT := IN1 >= IN2;
EQ		Equal To (ANY:IN1, ANY:IN2) => (BOOL:OUT) If IN1 equals IN2, OUT is TRUE; otherwise, OUT is FALSE. The input quantity can be extended. ST syntax example: OUT := IN1 = IN2;
LT		Less Than (ANY:IN1, ANY:IN2) => (BOOL:OUT) If IN1 is less than IN2, OUT is TRUE, otherwise OUT is False. The input quantity can be expanded. ST syntax example: OUT := IN1 < IN2;
LE		less than or equal to (ANY:IN1, ANY:IN2) => (BOOL:OUT) If IN1 is less than or equal to IN2, OUT is TRUE, otherwise OUT is False. The input quantity can be expanded. ST syntax example: OUT := IN1 <= IN2;
NE		Not Equal To (ANY:IN1, ANY:IN2) => (BOOL:OUT) If IN1 is not equal to IN2, OUT is True; otherwise, OUT is False. The input quantity can be expanded.

		ST syntax example: OUT := IN1 < > IN2;

3.3 Variable Structure Definition

Project				Scope	
Relays	I	External Input Relay		I0.0~I15.7 (Octal encoding, 128 points)	Total 256 (expansion machine+host points)
	Q	External Output Relay		Q0.0~Q15.7 (Octal encoding, 128 points)	
	MX	Auxiliary Relay	General Use	MX0~MX1023, 1024 points	total 6144 points
			Power-off Retention	MX4096~MX6143, 2048 points	
			Special Use	MX1024~MX4095, 3072 points, Partially for power outage maintenance	

Project					Scope	
	IW	External Input ADC			ADC 16 bit - IW0~IW1, IW2~29 (Decimal encoding), 4 points, totaling 30 points	
	MB/ MW/ MD	Data Registers	General Use	16 bit	MW0~MW1023,1024 points	total 12288 points
				32 bit	MD0~MD1023,1024 points	
			Power-off Retention	16 bit	MW4096~MW6143, 2048 points	
				32 bit	MD4096~MD6143, 2048 points	
			Special Use	8 bit	MB3072~MB4095, 1024 points,	
				16 bit	MW1024~MW4095, 3072points, Partially for power outage maintenance	
				32 bit	MD1024~MD4095, 3072points, Partially for power outage maintenance	
constant	K	decimal			K-32,768 ~ K32,767 (16 bit operation) K-2,147,483,648 ~ K2,147,483,647 (32 bit operation)	

3.4 Special Registers

3.4.1 Pulse Instructions

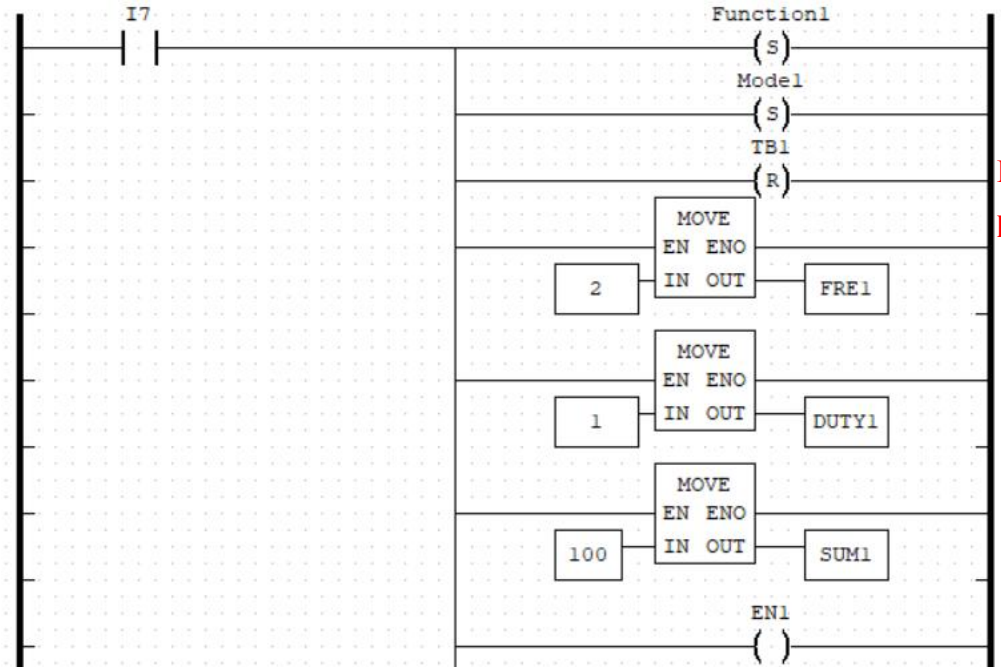
(1) Special Registers

Register Address	Functional Description	Numerical Definition
MX3072/MX3088/MX3104/MX3120	Read, Q0.0-Q0.3 Status Flag	1: Pulse output complete, 0: Incomplete
MX3074/MX3090/MX3106/MX3122	Write, Q0.0-Q0.3 Function Flag	1: PWM/PTO, 0: GPIO
MX3075/MX3091/MX3107/MX3123	Write, Q0.0-Q0.3 Mode Flag	1: PWM, 0: PTO
MX3076/MX3092/MX3108/MX3124	Write, Q0.0-Q0.3 Time Base Selection Flag	1: 1ms/div, 0: 1us/div

MW3072/MW3088/MW3104/MW3120	Write，Q0.0-Q0.3 Output Pulse Frequency Setting	2~65535（0.015Hz~500kHz）
MW3073/MW3089/MW3105/MW3121	Write，Q0.0-Q0.3 Output Pulse Width Setting)	1~32767
MW3074/MW3090/MW3106/MW3122	Write，Q0.0-Q0.3 Output Pulse Count Setting	1~65535

(2) Instruction Writing

Example: Q0.0 outputs 500kHz, Duty: 50% pulse.



MX3074 SET，PWM/PTO function

MX3075 SET，PWM mode

MX3076 RESET，time base is 1us per grid

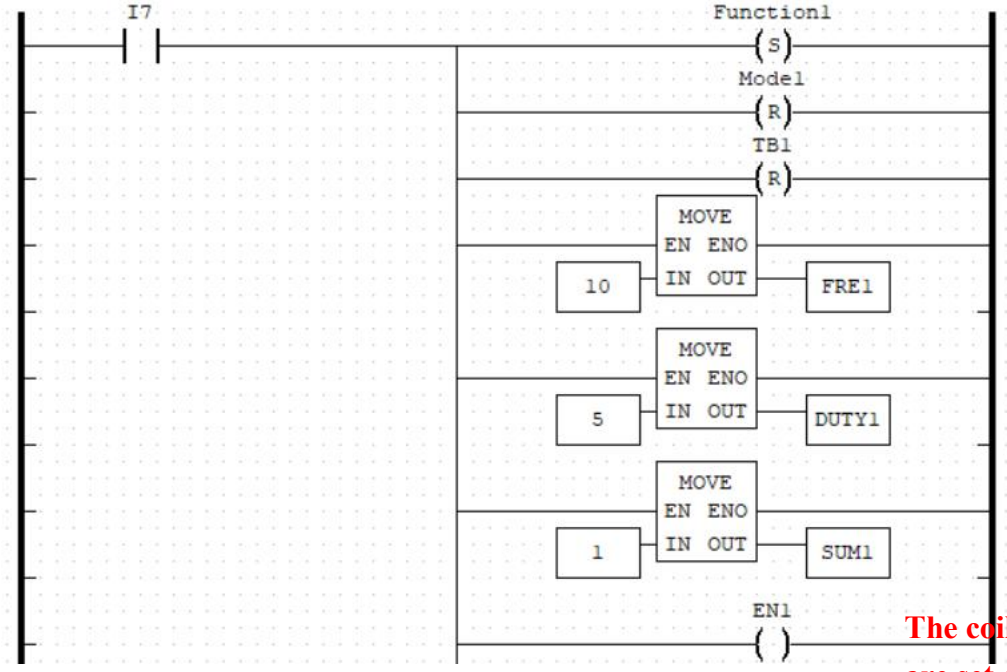
Set MW3072 to 2，a frequency of 500kHz means 2us

Set MW3073 to 1，a duty cycle of 50% means 1us

In PWM mode,MW3074 can be set arbitrarily.

The coil is Q0.0.After the above registers are set,enable Q0.0

- Example: Q0.0 outputs 100kHz, Duty: 50%, 1 pulse.



MX3074 SET，PWM/PTO

MX3075 RESET，PTO mode

MX3076 RESET，
time base is 1us per grid

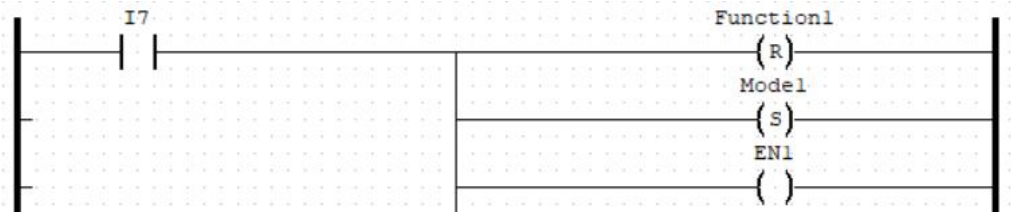
Set MW3072 to 10，a frequency of 100kHz means 10us

Set MW3073 to 5，a duty cycle of 50% means 5us

Set MW3074 to 1, and the pulse count to 1.

The coil is Q0.0.After the above registers are set,enable Q0.0

Example: Q0.0 outputs high/low level.



MX3074 RESET，GPIO function

MX3075 SET by default

The coil is Q0.0.

3.4.2 High-Speed Counter Instructions

(1) Special Registers

Register Address	Functional Description	Numerical Definition
MX2048/MX2064/MX2080/MX2096	write，I0.0-I0.3 Enable Detection Flag	1: Enable, 0: Disable
MX2049/MX2065/MX2081/MX2097	write，I0.0-I0.3 Counting Method Flag	1: Rising edge count, 0: Falling edge count
MX2050/MX2066/MX2082/MX2098	write，I0.0-I0.3 Counting Mode Flag	1: Count Up, 0: Count Down
MX2051/MX2067/MX2083/MX2099	read，I0.0-I0.3 Status Flag	1: Input in use, 0: Input unused
MX2052/MX2068/MX2084/MX2100	write，I0.0-I0.3 Input Pulse Frequency Detection Flag	1: Start, 0: Stop
MD2048/MD2064/MD2080/MD2096	read，I0.0-I0.3 Input Pulse Frequency	/
MD2049/MD2065/MD2081/MD2097	write，I0.0-I0.3 Input Pulse Preset Value，above this value, no longer count	0~4294967295
MD2050/MD2066/MD2082/MD2098	read，I0.0-I0.3 Input Pulse Count	/

(2) Instruction Writing

- HSC High-Speed Counting Mode：Example: I0.0 in rising edge, count up mode.

  Add a register address to the variable table and assign an initial value to use this function.

program0

Description: Class Filter: All

#	Name	Class	Type	Location	Initial Value
10	CEN1	Local	BOOL	%MX2048	1
11	CEDGE1	Local	BOOL	%MX2049	1
12	CMODE1	Local	BOOL	%MX2050	1
13	CSTATUS1	Local	BOOL	%MX2051	
14	CDC1	Local	BOOL	%MX2052	1
15	CFRE1	Local	UDINT	%MD2048	
16	CPRO1	Local	UDINT	%MD2049	
17	CSUM1	Local	UDINT	%MD2050	

MX2048 SET, and enable

MX2049 SET, and count on the rising

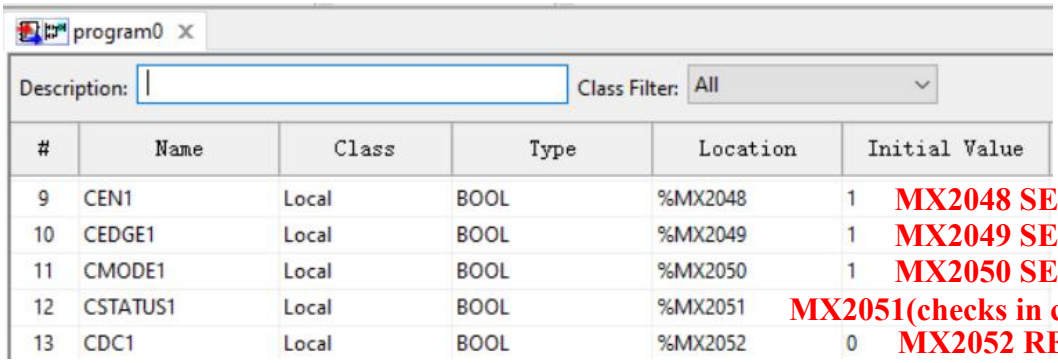
MX2050 SET, and count up

MX2052 SET, start the high-counting mode

MD2049 for preset values

MX2051 (checks if in counting mode), MD2048 (reads pulse frequency), MD2050 (reads pulse count), Through Debug monitoring

- Normal High/Low Level Input Mode: Example: I0.0 in high/low level output mode.



program0

Description: Class Filter: All

#	Name	Class	Type	Location	Initial Value
9	CEN1	Local	BOOL	%MX2048	1
10	CEDGE1	Local	BOOL	%MX2049	1
11	CMODE1	Local	BOOL	%MX2050	1
12	CSTATUS1	Local	BOOL	%MX2051	
13	CDC1	Local	BOOL	%MX2052	0

MX2048 SET, and enable
MX2049 SET, and count on the rising edge
MX2050 SET, and count up
MX2051(checks in counting mode), Through Debug
MX2052 RESET, GPIO function

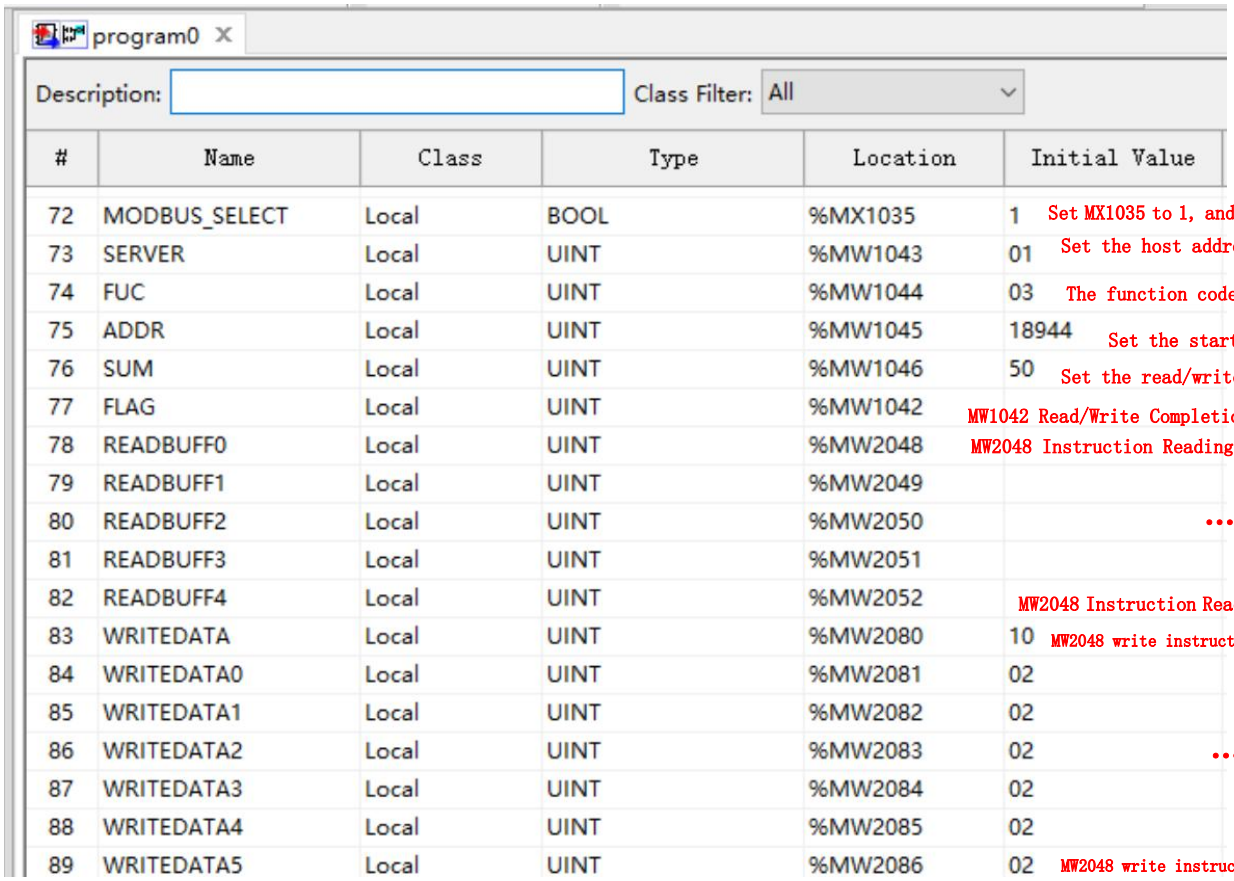
3.4.3 External communication instructions for the host machine

(1) Special Registers

Register Address	Functional Description	Numerical Definition
MX1035	write, Master-slave selection	1: As the host, 0: As the slave
MW1042	write, The flag indicating the completion of read/write operations when Modbus acts as the host	1: Read/Write Completed, 0: Read/Write Not Completed
MW1043	write, Modbus slave address	1~247
MW1044	write, Modbus as the master device, the function code	03 (reading)/ 06 (writing a single one)/ 10 (writing multiple ones)
MW1045	write, The starting address when Modbus acts as the host	0~65535
MW1046	write, The length of read/write addresses when Modbus acts as the host	/
MW2048 ↓ MW2079	read, PLC COM (RS-485) communication read instruction. When this instruction is executed, after the trusted end receives it, it will return a message. This message will be stored in MW2048 to MW2079. Users can use the content of these registers to view the returned data.	0~65535
MW2080 ↓ MW2143	write, PLC COM (RS-485) communication write instruction. When this instruction is executed, the content of the instruction sent out is stored in MW2080 to MW2143. Users can use the content of these registers to check the sent data.	0~65535

(2) Instruction Writing

Add the register address to the variable table and assign an initial value to enable this function (note that the host communication baud rate is 115.2K, and the slave's baud rate must be set to 115.2K).



program0

Description: Class Filter: All

#	Name	Class	Type	Location	Initial Value
72	MODBUS_SELECT	Local	BOOL	%MX1035	1
73	SERVER	Local	UINT	%MW1043	01
74	FUC	Local	UINT	%MW1044	03
75	ADDR	Local	UINT	%MW1045	18944
76	SUM	Local	UINT	%MW1046	50
77	FLAG	Local	UINT	%MW1042	
78	READBUFF0	Local	UINT	%MW2048	
79	READBUFF1	Local	UINT	%MW2049	
80	READBUFF2	Local	UINT	%MW2050	
81	READBUFF3	Local	UINT	%MW2051	
82	READBUFF4	Local	UINT	%MW2052	
83	WRITEDATA	Local	UINT	%MW2080	10
84	WRITEDATA0	Local	UINT	%MW2081	02
85	WRITEDATA1	Local	UINT	%MW2082	02
86	WRITEDATA2	Local	UINT	%MW2083	02
87	WRITEDATA3	Local	UINT	%MW2084	02
88	WRITEDATA4	Local	UINT	%MW2085	02
89	WRITEDATA5	Local	UINT	%MW2086	02

Set MX1035 to 1, and it will be in host mode.
Set the host address of MW1043 to 01
The function code for MW1044 is set to 03.
Set the starting address of MW1045
Set the read/write length of MW1046
MW1042 Read/Write Completion Flag
MW2048 Instruction Reading Data Buffer Register
...
MW2048 Instruction Reading Data Buffer Register
MW2048 write instruction to data buffer register
...
MW2048 write instruction to data buffer register

3.5 Special Register Functions

Special Register	Functional Description	Off ⇩ On	STOP ⇩ RUN	RUN ⇩ STOP	Attribute	Power outage maintenance	Default
MB3072	Axis Forward Limit Bit	-	-	-	R/W	NO	-
MB3073	Axis Origin Limit Bit	-	-	-	R/W	NO	-
MB3074	Axis Reverse Limit Bit	-	-	-	R/W	NO	-
MB3075	Axis Output Control Bit	-	-	-	R/W	NO	-
MB3076 ↓ MB3087	Reserved for Q0.0 axis	-	-	-	R/W	-	-
MB3088 ↓ MB3103	Q0.1 axis limit position, such as Q0.0	-	-	-	R/W	NO	-
MB3104 ↓ MB3119	Q0.2 axis limit position, such as Q0.0	-	-	-	R/W	NO	-
MB3120 ↓ MB3135	Q0.3 axis limit position, such as Q0.0	-	-	-	R/W	NO	-
MB3136 ↓ MB3583	Expand the shaft limit position, as above.	-	-	-	R/W	NO	-

MB3584 ↓ MB4095	Reserved	-	-	-	-	-	-
MX1024	Operation monitoring normally open contact (contact A)	Off	On	Off	R	NO	Off
MX1025	Operation monitoring normally closed contact (B contact)	On	Off	On	R	NO	On
MX1026	Start positive (RUN moment 'On') pulse wave	Off	On	Off	R	NO	Off
MX1032	Incorrect communication service requirements	Off	-	-	R	NO	Off
MX1035	Master-slave machine selection (1: as the host, 0: as the slave)	Off	-	-	R/W	NO	Off
MX2048	1.0 count enable flag (ON: enable, OFF: disable)	Off	Off	-	R/W	NO	Off
MX2049	I1.0 counting mode flag (ON: rising edge interrupt, OFF: falling edge interrupt)	Off	Off	-	R/W	NO	Off
MX2050	I1.0 Count Mode Flag (ON: Up, OFF: Down)	Off	Off	-	R/W	NO	Off
MX2051	I1.0 Clear Status Flag (ON: Start, OFF: Close)	Off	Off	-	R/W	NO	Off
MX2052	I1.0 Input Pulse Frequency Detection Flag (ON: Start, OFF: Close)	Off	Off	-	R/W	NO	Off
MX2053 ↓ MX2063	Reserved for I1.0						
MX2064 ↓ MX2079	I1.1 flag, as above I0.0	Off	Off	-	R/W	NO	Off
MX2080 ↓ MX2095	I1.2 flag, as above I0.0	Off	Off	-	R/W	NO	Off
MX2096 ↓ MX2111	I1.3 flag, as above I0.0	Off	Off	-	R/W	NO	Off
MX2112 ↓ MX2559	Expansion machine high-speed counting flag	Off	Off	-	R/W	NO	Off
MX2560 ↓ MX3071	Reserved for I						
MX3072	Q0.0 Status Flag (ON: Start, OFF: Close)	Off	Off	-	R/W	NO	Off
MX3073	Q0.0 Axis movement direction control position (ON: Positive axis counting, OFF: Negative axis counting)	Off	Off	-	R/W	NO	Off
MX3074	Q0.0 Function flag (ON: PWM/PTO function, OFF: GPIO)	Off	Off	-	R/W	NO	Off
MX3075	Q0.0 mode flag (ON: PWM mode, OFF: PTO mode)	Off	Off	-	R/W	NO	Off
MX3076	Q0.0 Benchmark selection flag (ON: 1ms/ grid, OFF: 1us/ grid)	Off	Off	-	R/W	NO	Off
MX3077	Q0.0 Restores the origin enable bit (ON: The axis begins to return to the origin. OFF: The shaft stops and returns to the origin	Off	Off	-	R/W	NO	Off

MX3078	Selection of the method to restore the origin (ON: Restore the origin to PTO mode. OFF: Restore to origin in limit sensor mode	Off	Off	-	R/W	NO	Off
MX3079	The Q0.0 axis direction is reversed (ON: QX.X when counting positively, the corresponding direction output is set to 0; conversely, when counting negatively, the output is set to 1. OFF: When QX.X counts positively, the output in the corresponding direction is set to 1 Conversely, the negative count output is set to 0.	Off	Off	-	R/W	NO	Off
MX3080	The limit direction of the Q0.0 axis is reversed When the input of the ON: QX.X limit sensor is 1, the corresponding pin flag is 0 OFF: When the input of the QX.X limit sensor is 1, the corresponding pin flag is 1.	Off	Off	-	R/W	NO	Off
MX3081	The Q0.0 axis returns to the origin direction position (ON: Restore the origin direction towards the positive counting direction of the axis. OFF: Return to the origin direction towards the negative counting direction of the axis	Off	Off	-	R/W	NO	Off
MX3082	The Q0.0 axis returns to the front and back positions of the origin (ON: Restore the origin to positive count and leave the origin position. OFF: Restore to the origin to reverse count and leave the origin position.	Off	Off	-	R/W	NO	Off
MX3083	The time base for restoring the Q0.0 axis to the origin (ON: 1ms, OFF: 1us)	Off	Off	-	R/W	NO	Off
MX3084	Q0.0 axis crawling speed time base (ON: 1ms, OFF: 1us)	Off	Off	-	R/W	NO	Off
MX3085 ↓ MX3087	Reserved	-	-	-	-	-	-
MX3088 ↓ MX3103	The Q0.1 flag, like Q0.0	Off	Off	-	R/W	NO	Off
MX3104 ↓ MX3119	The Q0.2 flag, like Q0.0	Off	Off	-	R/W	NO	Off
MX3120 ↓ MX3135	Q0.3 flag, like Q0.0	Off	Off	-	R/W	NO	Off
MX3136 ↓ MX3583	The flag of expansion machine Q2.0-Q14.3, like Q0.0	Off	Off	-	R/W	NO	Off
MX3584 ↓ MX4095*	Reserve for Q	Off	Off	-	R/W	NO	Off

Special Register	Functional Description	Off ⇩ On	STOP ⇩ RUN	RUN ⇩ STOP	Attribute	Power outage maintenance	Default
MW1025	The system program version of the DPLC model (users can read the firmware version of the PLC from this register.) For example, MW1025=0, that is, firmware version 0.0)	-	-	-	R	NO	#
MW1028	Modbus host address, 1-247	-	-	-	R	Yes	#
MW1029	Current scanning cycle (unit: 1ms)	-	-	-	R	NO	0
MW1030	Minimum scanning period (unit: 1ms)	-	-	-	R/W	NO	0
MW1031	Maximum scanning period (unit: 1ms)	-	-	-	R/W	NO	0
MW1033	Perpetual calendar (RTC) seconds 00 to 59	-	-	-	R/W	Yes	0
MW1034	The perpetual calendar (RTC) ranges from 00 to 59	-	-	-	R/W	Yes	0
MW1035	Perpetual calendar (RTC) hours: 00-23	-	-	-	R/W	Yes	0
MW1036	Perpetual calendar (RTC) Days 01 to 31	-	-	-	R/W	Yes	0
MW1037	The period from January 1st to December in the perpetual calendar (RTC) month	-	-	-	R/W	Yes	0
MW1038	The perpetual calendar (RTC) is from Monday to 7th of the week	-	-	-	R/W	Yes	0
MW1039	Perpetual calendar (RTC) years 00 to 99	-	-	-	R/W	Yes	0
MW1042	The flag indicating the completion of read/write operations when Modbus acts as the host	-	-	-	R	No	0
MW1043	Modbus slave address, 1 - 247		-	-	R/W	No	0
MW1044	Modbus as the master device, the function code	-	-	-	R/W	No	0

MW1045	The starting address when Modbus acts as the host	-	-	-	R/W	No	0
MW1046	The length of read/write addresses when Modbus acts as the host	-	-	-	R	No	0
MW2048 ↓ MW2079	PLC COM2 (RS-485) provides convenient read instructions. When this instruction is executed, the command characters sent out are stored in MW2048 to MW2079. Users can check whether the commands are correct based on the contents of this cache.	0	-	-	R	否	0
MW2080 ↓ MW2143	Modbus communication instruction data processing, built-in RS-485 communication convenience instruction of PLC. The instruction sent out during execution, when the trusted end receives it, it will return a message, and this message will be stored in MW2080 to MW2143. Users can use the content of this register to check the returned data.	0	-	-	R	否	0
MD2048	I0.0 input frequency	0	-	-	R/W	NO	0
MD2049	I0.0 Input the preset value of the pulse	0	-	-	R/W	NO	0
MD2050	I0.0 Input the value of the pulse meter	0	-	-	R/W	NO	0
MD2051 ↓ MD2063	Reserve for I0.0						
MD2058 ↓ MD2060	I0.1 period, pulse width, pulse count, such as I0.0	0	-	-	R/W	NO	0
MD2068 ↓ MD2070	I0.2 period, pulse width, pulse count, such as I0.0	0	-	-	R/W	NO	0
MD2078 ↓ MD2080	I0.3 period, pulse width, pulse count, such as I0.0	0	-	-	R/W	NO	0
MD2081 ↓ MD2559	The value of the high-speed counter of the expansion machine	0	-	-	R/W	NO	0
MD2560 ↓ MD3071	Reserve for I						
MD3072	Q0.0 pulse register, used to record the actual position of the motion axis	0	-	-	R/W	NO	0
MD3073	The speed change time for the Q0.0 axis to return to the origin	0	-	-	R/W	NO	0

MD3074 ↓ MD3087	Reserve for Q0.0	0	-	-	R/W	NO	0
MD3088 ↓ MD3103	Q0.1 pulse register, used for recording the actual position of the motion axis	0	-	-	R/W	NO	0
MD3104 ↓ MD3119	Q0.2 pulse register, used for recording the actual position of the motion axis	0	-	-	R/W	NO	0
MD3120 ↓ MD3115	Q0.3 pulse register, used for recording the actual position of the motion axis	0	-	-	R/W	NO	0
MD3116 ↓ MD3583	The flag of expansion machine Q2.0-Q14.3, such as Q0.0	Off	Off	-	R/W	NO	Off
MD3584 ↓ MD4095*	Reserve for Q						
MW3072	Q0.0 Output cycle setting	0	-	-	R/W	NO	0
MW3073	Q0.0 Output pulse width setting	0	-	-	R/W	NO	0
MW3074	Q0.0 Setting of the number of output pulses	0	-	-	R/W	NO	0
MW3075	The speed of the Q0.0 axis returning to the origin	0	-	-	R/W	NO	0
MW3076	Q0.0 axis crawling speed	0	-	-	R/W	NO	0
MW3077	Q0.0 axis return to origin speed duty cycle	0	-	-	R/W	NO	0
MW3078	Duty cycle of Q0.0 axis crawling speed	0	-	-	R/W	NO	0
MW3079 ↓ MW3087	Reserve for Q0.0						
MW3088 ↓ MW3103	Q0.1 outputs the period, width and count, such as Q0.0	0	-	-	R/W	NO	0
MW3104 ↓ MW3119	Q0.2 outputs the period, width and count, such as Q0.0	0	-	-	R/W	NO	0
MW3120 ↓ MW3135	Q0.3 outputs the period, width and count, such as Q0.0	0	-	-	R/W	NO	0

MW3136 ↓ MW3583	The output cycle, width and count of the expansion machine Q2.0-Q14.3, such as Q0.0	0	-	-	R/W	NO	0
MW3584 ↓ MW4095	Reserve for Q	0	-	-	R/W	NO	0

Installation Manual

4、Communication Functions

4.1 Modbus Communication Function

DPLC uses Modbus RTU Master-Slave transmission mode. Baud rate is 115.2Kbps, 1 Start bit, 8 Data bits, No parity,1 Stop bit.

Due to the Modbus protocol definition limiting the maximum transmission amount of packets, the maximum limit in RTU mode is 128 bytes, so the interval between transmitted packets must not be less than 20mSec. When transmitting data, except for Error Check (CRC16) data, all word data must follow the principle of sending the High byte first.

Master Mode：

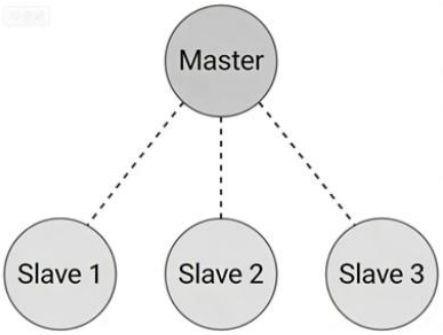
When PLC acts as a master device, it can communicate with other slave devices using Modbus-RTU or Modbus-ASCII protocols through Modbus instructions; exchanging data with other devices.

Slave Mode：

When PLC acts as a slave device, it can only respond to requests from other masters.

Concept of Master-Slave：

In an RS485 network, at any moment, there can be one master and multiple slaves (as shown in the figure below). The master can perform read/write operations on any of the slaves. Slaves cannot exchange data directly with each other. The master needs to write a communication program to read/write a specific slave. Slaves do not need to write communication programs; they only need to respond to the master's read/write operations. (Wiring method: All 485+ connected together, all 485- connected together).



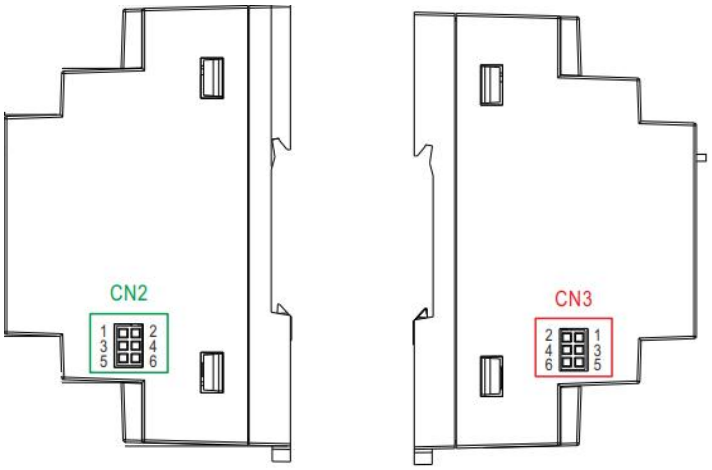
The reason why there are dashed arrows in the figure is that theoretically, in two networks, as long as each PLC does not send data, any PLC in the network can be used as the master station, and other PLCs can be used as slave stations; However, due to the lack of a unified clock reference among multiple PLCs, it is easy for multiple PLCs to send data at the same time, which can lead to communication conflicts and failures. Therefore, it is not recommended to use this method.

4.2 Introduction to Master-Slave Usage

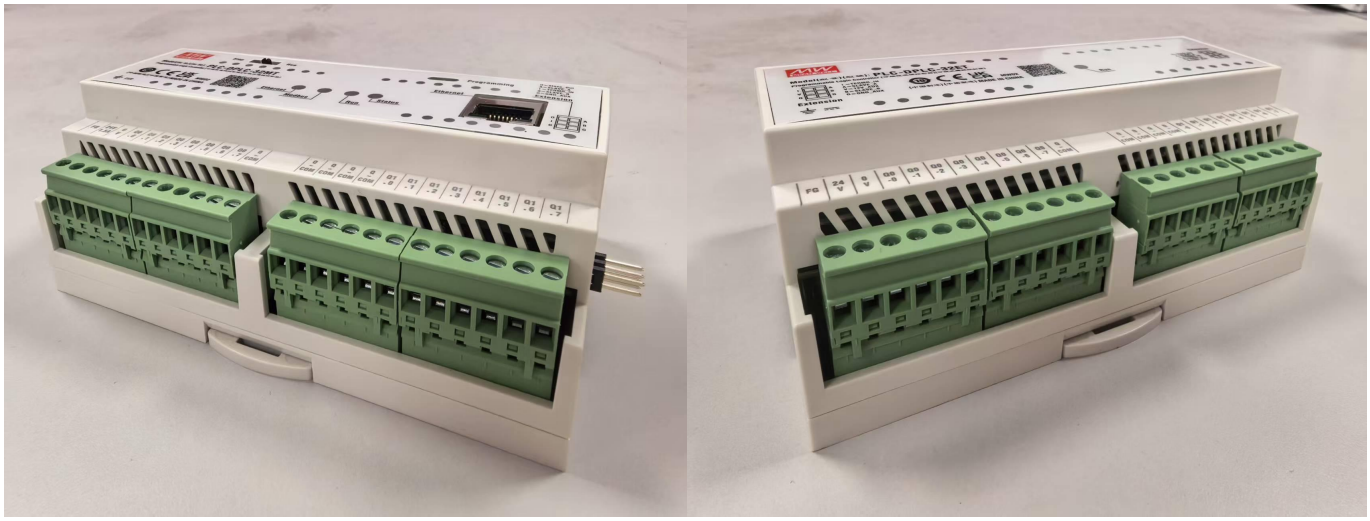
4.2.1 Master-Slave Wiring

Connect the female port CN2 of the master device to the female port CN3 of the slave device using pin headers or flat cables.If multiple slave devices need to be connected, follow the same connection method for the rest.

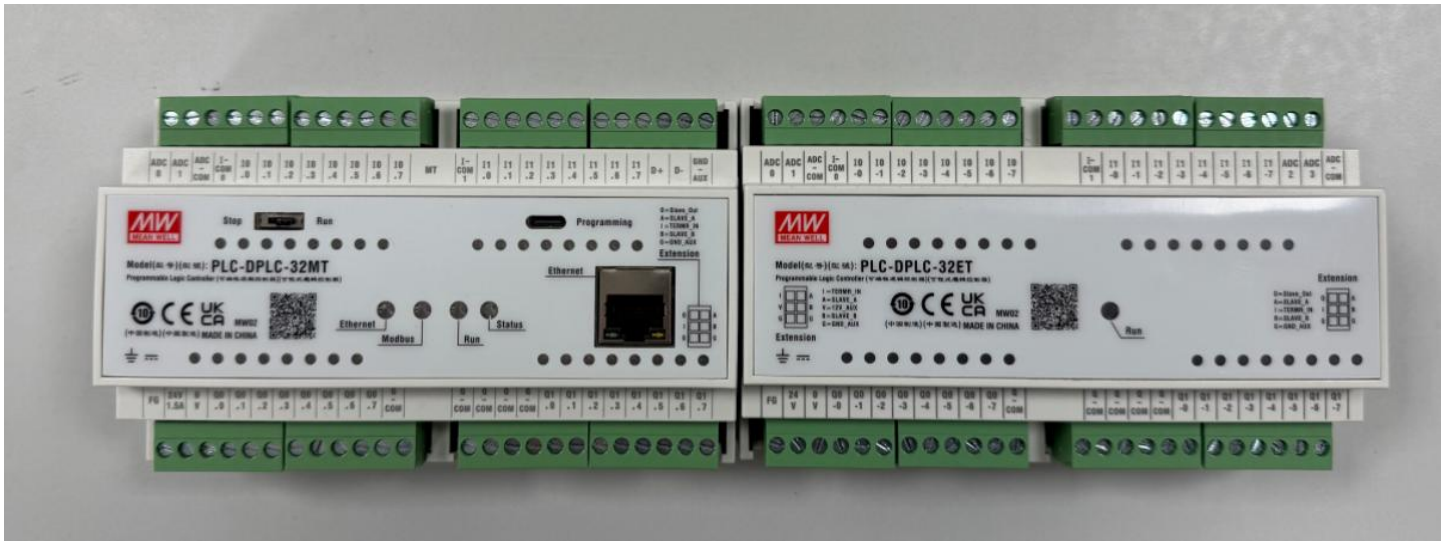
Installation Manual



The left picture below shows the CN2 of the main unit plugged in with the pin header, and then the CN3 of the slave unit in the right picture is plugged in.

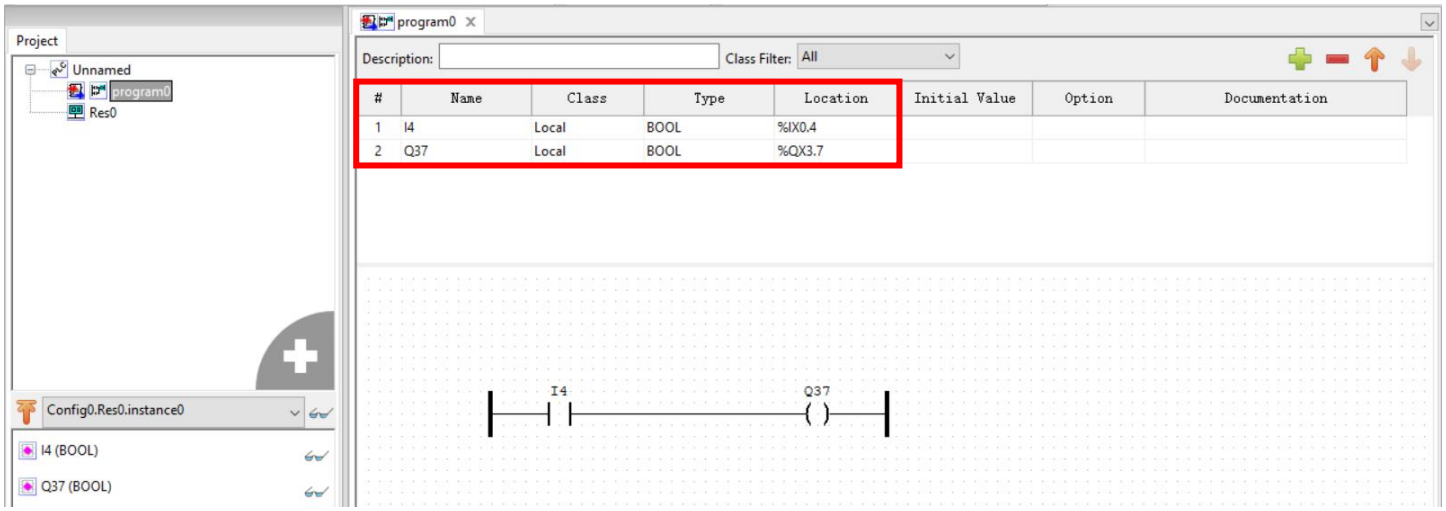


The following is a physical picture of the connection between the master and slave machines using pin headers.



4.2.2 Program Writing

The figure below is an example of the Master I port controlling the Slave Q port:



Add new variables in the data variable table, then map them to the corresponding position of the DPLC.

4.2.3 Abnormal Alarm

Master:

Light	Green	Orange	Red
Status Light	Solid on: System abnormal Flashing: Storage action	/	/
Run Light	Solid on: Normal work Flashing: stop status	Solid on: Extension unit disconnected Flashing: Burning program	Solid on: Program execution abnormal
Modbus Light	Solid on: Communicating	/	Solid on: Communication abnormal
Ethernet Light	Solid on: Communicating	/	Solid on: Communication abnormal

Extension Unit:

Light	Green	Orange	Red
Run Light	Addressing judgment completed	Solid on: Judging addressing	Solid on: Communication abnormal

4.3 Communication Flags and Registers

4.3.1 Communication Flags

Communication Flags	Functional Description	Numerical Definition
MX1032	Read item, Monitor if communication function is correct	0: No incorrect communication; 1: Incorrect communication exists.
MW1028	Write item, Modbus Master Address	1~247

4.3.2DPLC Variables Corresponding to Modbus Addresses

Register position range	The number of bytes	Name of PLC software component	Description	Function Code
0x0000~ 0x007F	16	Q_STATUS	Output (Q0~Q127 Operating status	0x01、0x05、0X0F
0x0080~ 0x03FF	-	Reserved	-	-
0x0400~ 0x07FF	128	MX0~MX1023	Relays are generally used	0x01、0x05、0X0F
0x0800~ 0x0BFF	-	Reserved	-	-
0x0C00~ 0x0FFF	128	MX1024-MX2047	Special relay	0x01、0x05、0X0F
0x1000~ 0x102F	6	MX2048-MX2095	Special relay (I0-I3)	0x01、0x05、0X0F
0x1030~ 0x13FF	-	Reserved	-	-
0x1400~ 0x153F	40	MX3072-MX3391	Special relay (Q0-Q31)	0x01、0x05、0X0F
0x1540~ 0x17FF	-	Reserved	-	-
0x1800~ 0x1FFF	256	MX4096-MX6143	Power-off holding relay	0x01、0x05、0X0F
0x2000~ 0x23FF	-	Reserved	-	-
0x2400~ 0x33FF	-	Reserved	-	-
0x3400~ 0x347F	16	I_STATUS	Input (I0~I127 Operating status	0x02
0x3480~ 0x37FF	-	Reserved	-	-
0x3800~ 0x38FF	512	MB3072-MB3583	8-bit special data register	0x03、0x06、0X10
0x3900~ 0x39FF	-	Reserved	-	-
0x3A00~ 0x3DFF	2048	MW0-MW1023	16-bit general data register	0x03、0x06、0X10
0x3E00~ 0x41FF	-	Reserved	-	-
0x4200~ 0x45FF	2048	MW1024-MW2047	16-bit special data register	0x03、0x06、0X10
0x4600~ 0x47FF	1024	MW2048-MW2559	16-bit special data register	0x03、0x06、0X10
0x4800~ 0x49FF	-	Reserved	-	-
0x4A00~ 0x4BFF	1024	MW3072-MW3583	16-bit general data register (Q0-Q3)	0x03、0x06、0X10
0x4C00~ 0x4DFF	-	Reserved	-	-

0x4E00~ 0x55FF	4096	MW4096-MW6143	16-bit power-off data retention register	0x03、0x06、0X10
0x5600~ 0x59FF	-	Reserved	-	-
0x5A00~ 0x61FF	4096	MD0-MD1023	32-bit general data register	0x03、0x06、0X10
0x6200~ 0x69FF	-	Reserved	-	-
0x6A00~ 0x71FF	4096	MD1024-MD2047	32-bit special data register	0x03、0x06、0X10
0x7200~ 0x75FF	2052	MD2048-MD2560	32-bit special data register (I0-I3)	0x03、0x06、0X10
0x7600~ 0x79FF	-	Reserved	-	-
0x7800~ 0x7DFF	2048	MD3072-MD3583	32-bit pulse register (Q0-Q3)	0x03、0x06、0X10
0x7E00~ 0x81FF	-	Reserved	-	-
0x8200~ 0x91FF	8192	MD4096-MD6143	32-bit power-off data retention register	0x03、0x06、0X10
0x9200~ 0xA1FF	-	Reserved	-	-
0xA200~ 0xB1FF	-	Reserved	-	-
0xB200~ 0xB21D	60	IW0-IW29	ADC analog input value	0x04
0xB21E~ 0xB5FF	-	Reserved	-	-
0xB600~ 0xFFFF	-	Reserved	-	-

Installation Manual

5、Examples

5.1IO Function Usage

5.1.1Analog Input Usage

#	Name	Class	Type	Location	Initial Value	Option
1	ADC0	Local	UINT	%IW0		
2	ADC1	Local	UINT	%IW1		
3	storage0	Local	UINT	%MW0		
4	storage1	Local	UINT	%MW1		

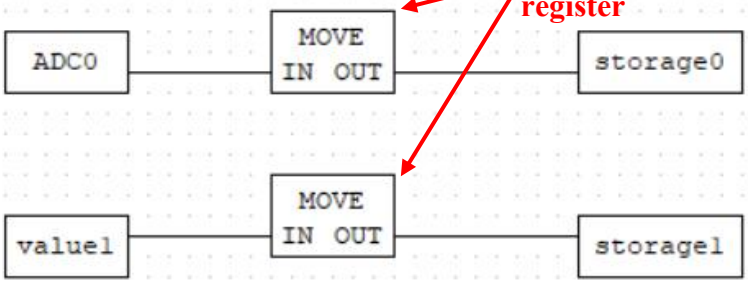
IW0 is the 1st Analog Input Channel for MT

IW1 is the 2nd Analog Input Channel for MT

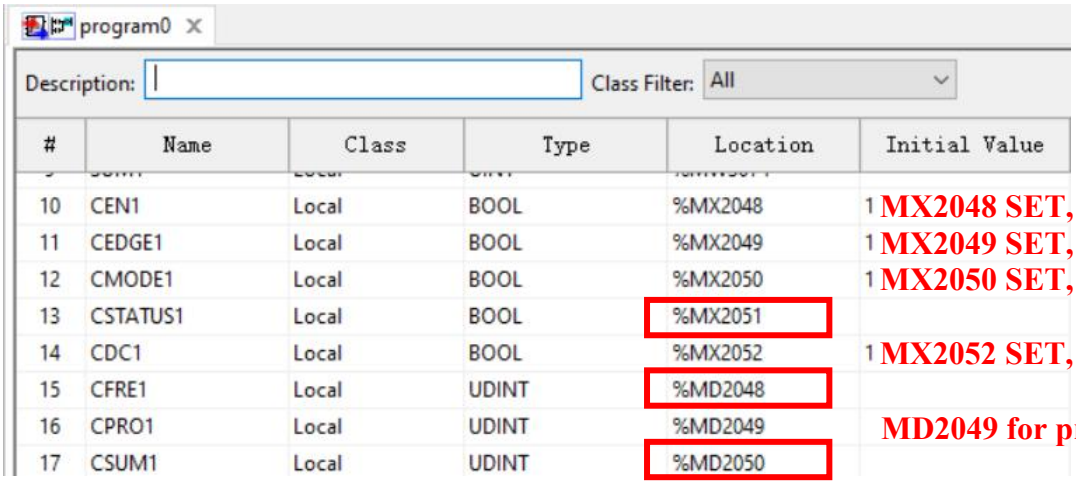
Set up two registers to store the Analog quantities

The Debug function can be used to view the values of ADC0 and ADC1.

Assign the Analog quantity to the register



5.1.2 HSC Input Usage



#	Name	Class	Type	Location	Initial Value
10	CEN1	Local	BOOL	%MX2048	1
11	CEDGE1	Local	BOOL	%MX2049	1
12	CMODE1	Local	BOOL	%MX2050	1
13	CSTATUS1	Local	BOOL	%MX2051	
14	CDC1	Local	BOOL	%MX2052	1
15	CFRE1	Local	UDINT	%MD2048	
16	CPRO1	Local	UDINT	%MD2049	
17	CSUM1	Local	UDINT	%MD2050	

MX2048 SET, and enable

MX2049 SET, and count on the rising edge

MX2050 SET, and count up

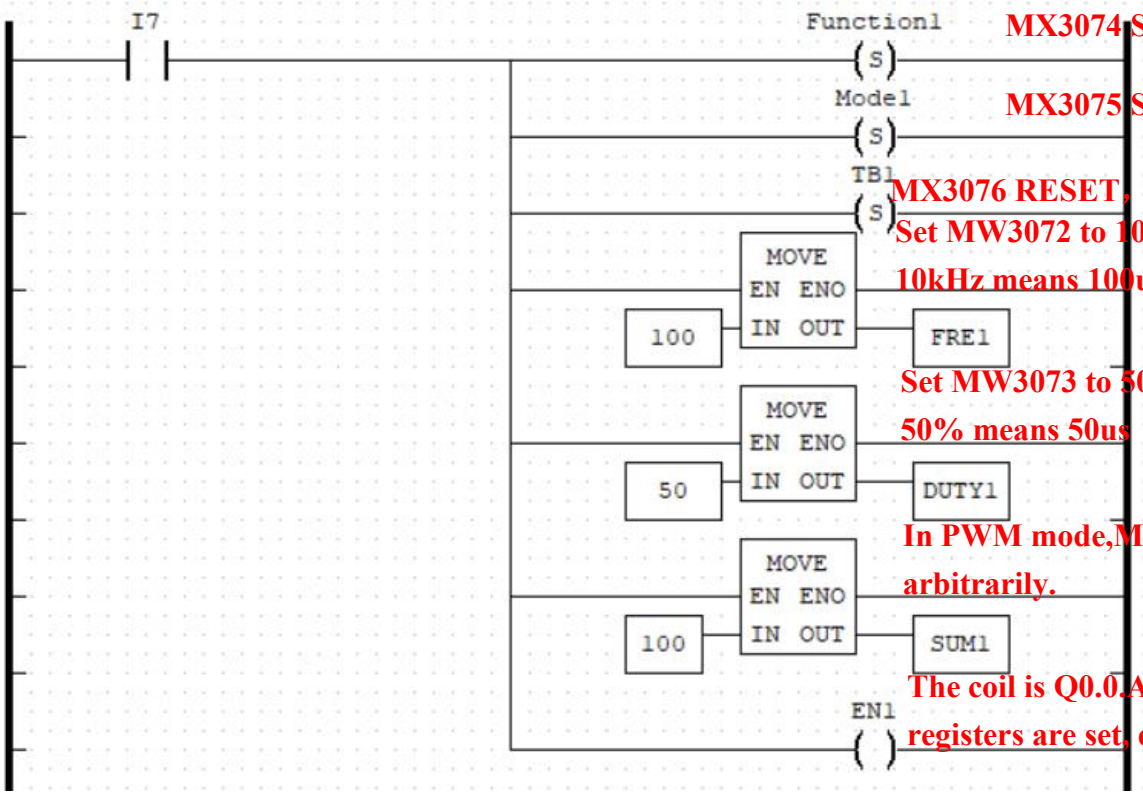
MX2052 SET, start the high-counting mode

MD2049 for preset values

MX2051 (checks if in counting mode), MD2048 (reads pulse frequency), MD2050 (reads pulse count), Through Debug monitoring.

5.1.3 PWM Output Usage

#	Name	Class	Type	Location	Initial Value	Option
1	I7	Local	BOOL	%IX0.7		
2	EN1	Local	BOOL	%QX0.0		
3	Status1	Local	BOOL	%MX3072		
4	Function1	Local	BOOL	%MX3074		
5	Mode1	Local	BOOL	%MX3075		
6	TB1	Local	BOOL	%MX3076		
7	FRE1	Local	UINT	%MW3072		
8	DUTY1	Local	UINT	%MW3073		
9	SUM1	Local	UINT	%MW3074		



5.1.4 Digital Input/Output Usage

#	Name	Class	Type	Location	Initial Value	Option
1	I4	Local	BOOL	%IX0.4	Digital input I0.4	
2	Q4	Local	BOOL	%QX0.4	Digital output Q0.4	

I0.4 SET, Q0.4 SET
I0.4 RESET, Q0.4 RESET

5.2 Return to Homing Function

Example:

#	Name	Class	Type	Location	Initial Value	Option
1	START	Local	BOOL	%IX0.0		
2	Y_SET	Local	BOOL	%QX0.0		
3	Y_P	Local	BYTE	%MB3072		
4	Y_Z	Local	BYTE	%MB3073	8	
5	Y_N	Local	BYTE	%MB3074		
6	Y_OUT	Local	BYTE	%MB3075	4	
7	Y_Status	Local	BOOL	%MX3072		
8	Y_DIR	Local	BOOL	%MX3073		
9	Y_IO	Local	BOOL	%MX3074	1	
10	Y_PWM	Local	BOOL	%MX3075	1	
11	Y_BASE	Local	BOOL	%MX3076	0	
12	Y_REST_EN	Local	BOOL	%MX3077		
13	Y_REST_FUN	Local	BOOL	%MX3078	0	
14	Y_OUT_REV	Local	BOOL	%MX3079	0	
15	Y_IN_REV	Local	BOOL	%MX3080	0	
16	Y_REST_DIR	Local	BOOL	%MX3081	0	
17	Y_REST_ZDIR	Local	BOOL	%MX3082	1	
18	Y_SYS	Local	UINT	%MW3072	100	
19	Y_DUTY	Local	UINT	%MW3073	50	
20	Y_FSYS	Local	UINT	%MW3075	100	
21	Y_FDUTY	Local	UINT	%MW3076	50	
22	Y_CSYS	Local	UINT	%MW3077	10	
23	Y_CDUTY	Local	UINT	%MW3078	5	
24	Y_CHANGE_TIME	Local	DINT	%MD3073	100	

MB3072：Q0.0 Axis forward rotation limit position

MB3073：Q0.0 Axis origin limit position

MB3074：Q0.0 Axis reversal rotation limit position

MB3075：Q0.0 Axis output control bit

MX3072：Q0.0 Status flag (1: Start, 0: Stop)

MX3073：Direction control bit (1: positive, 0: negative)

MX3074：1：PWM/PTO,0：GPIO

MX3075：1：PWM,0：PTO

MX3076：Time base (1:1ms per grid, 0:1us per grid)

MX3077：Q0.0 Enable flag to the original point

MX3078：Q0.0 Axis direction inversion flag

MX3079：Q0.0 Axis restoration of origin direction flag

MX3080：Q0.0 Restore the original position

MX3081：Time base to restore the original point (ON：1ms, OFF：1us)

MX3082：Time base to creep speed (ON：1ms, OFF：1us)

Set MW3072 to 100，a frequency of 10kHz means 100us

Set MW3073 to 50，a duty cycle of 50% means 50us

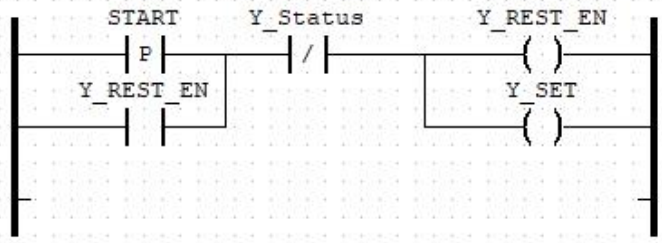
MW3075：Q0.0 Restore to original starting speed

MW3076：Q0.0 Axis crawling speed

MW3077：Q0.0 Ratio of duty cycle for restoring the shaft to its original position speed Axis return to origin acceleration time

MW3078：Q0.0 The duty cycle of the shaft crawling speed

MD3073：Q0.0 Axis return to origin acceleration time



After pressing the start button, axis Q0.0 starts homing enable and operation enable, returns to the origin at the set homing speed. When approaching the origin, it switches to crawling speed until it reaches the origin

5.3 Host external communication function

Example:

program0

Description: Class Filter: All

#	Name	Class	Type	Location	Initial Value	
72	MODBUS_SELECT	Local	BOOL	%MX1035	1	Set MX1035 to 1, and it will be in host mode.
73	SERVER	Local	UINT	%MW1043	01	Set the host address of MW1043 to 01
74	FUC	Local	UINT	%MW1044	03	The function code for MW1044 is set to 03.
75	ADDR	Local	UINT	%MW1045	18944	Set the starting address of MW1045
76	SUM	Local	UINT	%MW1046	50	Set the read/write length of MW1046
77	FLAG	Local	UINT	%MW1042		MW1042 Read/Write Completion Flag
78	READBUFF0	Local	UINT	%MW2048		MW2048 Instruction Reading Data Buffer Register
79	READBUFF1	Local	UINT	%MW2049		
80	READBUFF2	Local	UINT	%MW2050		...
81	READBUFF3	Local	UINT	%MW2051		
82	READBUFF4	Local	UINT	%MW2052		MW2048 Instruction Reading Data Buffer Register
83	WRITEDATA	Local	UINT	%MW2080	10	MW2048 write instruction to data buffer register
84	WRITEDATA0	Local	UINT	%MW2081	02	
85	WRITEDATA1	Local	UINT	%MW2082	02	
86	WRITEDATA2	Local	UINT	%MW2083	02	...
87	WRITEDATA3	Local	UINT	%MW2084	02	
88	WRITEDATA4	Local	UINT	%MW2085	02	
89	WRITEDATA5	Local	UINT	%MW2086	02	MW2048 write instruction to data buffer register

Instruction 03: Set MX1035 to 1, MW1043 to 01, MW1044 to 03, MW1045 to the desired read address, and MW1046 to the length of the desired read address. Then, check the value of MW1042. If the value is 1, it indicates that the read transmission is complete. By checking the values of MW2048 to MW2079, you can obtain the value sent back by the slave device. (Note: The initial values of the registers need to be converted to decimal first.)

06/10 Instruction: Set MX1035 to 1, MW1043 to 01, MW1044 to 06/10 (10 in decimal is 16), MW1045 to the desired write address, MW1046 to the length of the desired write address, MW2080 to MW2143 as the write data. Then check the value of MW1042. If the value is 1, it indicates that the write operation is completed (note: the initial values of the registers need to be converted to decimal first).

Installation Manual

Revision History

Version	Date	Reason
V0.0	2025/03/07	Create document
V1.0	2025/05/19	Add the steps to update the system time
V1.1	2026/01/06	Modify the example
V1.2	2026/01/14	1. Add a physical diagram of the main and slave machine connector connection. 2. Make the alarm light code consistent with the specification sheet. 3. Modify some layout.
V1.3	2026/03/02	Description of Adding External Communication Function for Host Machine