



# Installation Manual

## 目录

一、产品概述.....	3
1.1 OpenPLC 介绍.....	3
1.2 系统要求.....	3
1.3 安装步骤.....	3
二、编程环境.....	6
2.1 系统配置.....	6
2.1.1 以太网连接.....	6
2.1.2 USB 连接.....	7
2.2 编程界面.....	10
2.2.1 编译界面.....	10
2.2.2 烧录界面.....	11
2.3 使用说明.....	16
2.3.1 新建工程.....	16
2.3.2 编译.....	17
2.3.3 烧录.....	18
2.3.4 调试.....	22
2.3.5 韧体更新 Firmware Update Ethernet.....	24
三、编程操作.....	26
3.1 编程方式.....	26
3.2 梯形图的编辑.....	26
3.2.1 地址映射.....	27
3.2.2 软元件介绍.....	27
3.2.3 函数块介绍.....	35
3.3 变量结构定义.....	40
3.4 特殊寄存器.....	41
3.4.1 脉冲指令.....	41
3.4.2 高速计数指令.....	43
3.5 特殊暂存器功能.....	44
四、通讯功能.....	49
4.1 Modbus 通讯功能.....	49
4.2 主从机使用介绍.....	49
4.2.1 主从机接线.....	49
4.2.2 程序编写.....	51
4.2.3 异常报警.....	51
4.3 通讯标志位与寄存器.....	51
4.3.1 通讯标志位.....	51



# Installation Manual

4.3.2DPLC 变量对应 Modbus 地址 .....	52
五、范例 .....	54
5.1IO 功能使用 .....	54
5.1.1 类比输入使用 .....	54
5.1.2 HSC 输入使用 .....	54
5.1.3PWM 输出使用 .....	55
5.1.4 数位输入输出使用 .....	55
5.2 恢复原点功能 .....	56



# Installation Manual

## 一、产品概述

### 1.1 OpenPLC 介绍

OpenPLC 是一个开源的可编程逻辑控制器，它是一个易于使用的软件，是首个完全功能标准化的开源 PLC，在软硬件方面均符合标准。OpenPLC 项目遵循 IEC 61131-3 标准，该标准定义了 PLC 的基本软件架构和编程语言。OpenPLC 主要用于工业和家庭自动化、物联网和 SCADA 研究等领域。

OpenPLC Editor 支持 IEC 61131-3 标准中定义的五种语言：梯形图 (LD)、功能块图 (FBD)、指令表 (IL)、结构化文本 (ST) 和顺序功能图 (SFC)。这些编程语言为用户提供了多种选择，以适应不同的编程需求和偏好。OpenPLC Editor 的多功能性和易用性使其成为 PLC 编程的理想工具。

### 1.2 系统要求

处理器	Intel Core i3 以上	显示器缩放设置	100% (推荐)
内存	不少于 4G	显示器分辨率	1920*1080 (推荐)
硬盘	300GB 以上	操作系统	windows7 及以上 (推荐)

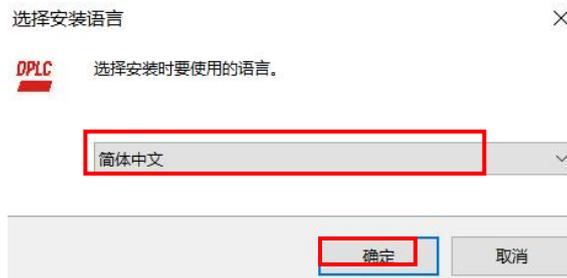
### 1.3 安装步骤

编程软件下载路径：<https://www.meanwell.cc/productSoftware.aspx>

点击安装包，开始安装。



弹出以下窗口，选择安装时的语言。



选择创建快捷方式，点击下一步：



# Installation Manual

DPLC 安装 - DPLC\_Editor 版本 1.0

## 选择附加任务

您想要安装程序执行哪些附加任务？

选择您想要安装程序在安装 DPLC\_Editor 时执行的附加任务，然后点击“下一步”。

附加快捷方式：

创建桌面快捷方式(D)



下一步(N)

取消

点击安装：

DPLC 安装 - DPLC\_Editor 版本 1.0

## 准备安装

安装程序现在准备开始安装 DPLC\_Editor 到您的电脑中。

点击“安装”继续此安装程序。如果您想要回顾或修改设置，请点击“上一步”。

附加任务：

附加快捷方式：

创建桌面快捷方式(D)



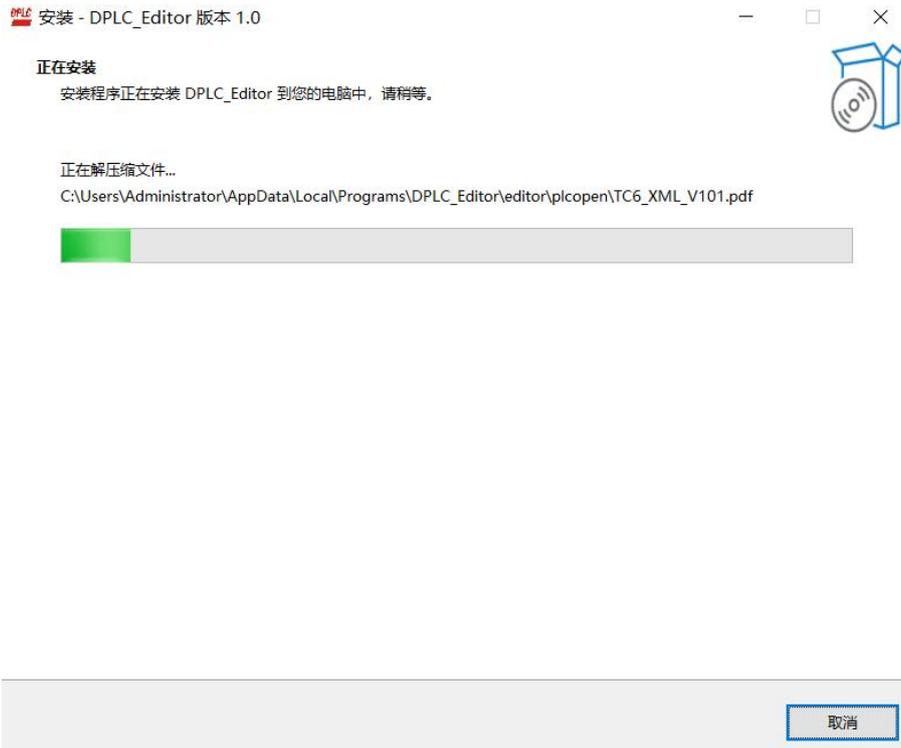
上一步(B)

安装(I)

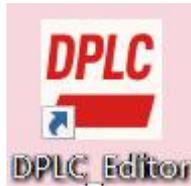
取消



# Installation Manual



下载完成后，点击“Close”，桌面出现“DPLC\_Editor”图标。



## 二、编程环境

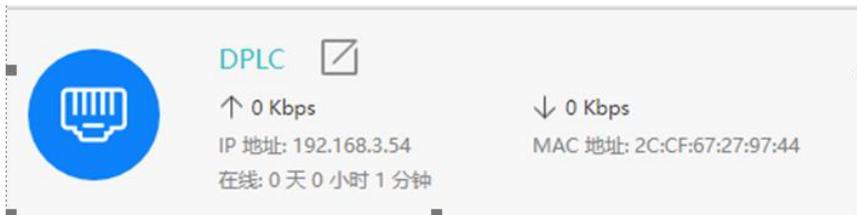
### 2.1 系统配置

#### 2.1.1 以太网连接

(1) 用网线将 DPLC 与路由器相连，观察 RJ45 黄灯是否常亮，绿灯是否有闪动。电脑的网络连上同一路由器，以太网连接配置完成。下图红框处为 DPLC 网口连接处。



(2) 进入路由器查看 DPLC 的 IP 地址。



(3) 浏览器输入 IP:80

🌐 192.168.3.54:80

(4) 回车进入登录界面，用户名和密码默认都是 openplc，点击 LOGIN，进入烧录界面。

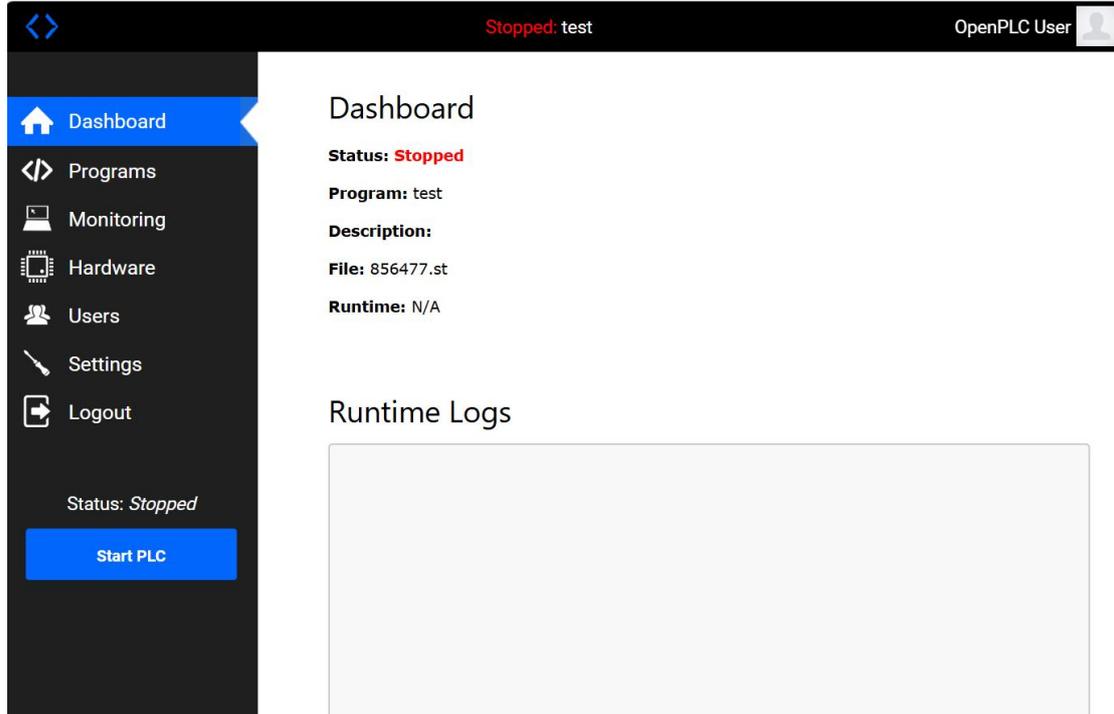
Welcome to OpenPLC

Use your credentials to login

openplc

.....

LOGIN



## 2.1.2 USB 连接

(1) 连接 DPLC Type-C 口和电脑。



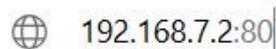
(2) 找到树莓派 CM4 WLAN 设置。

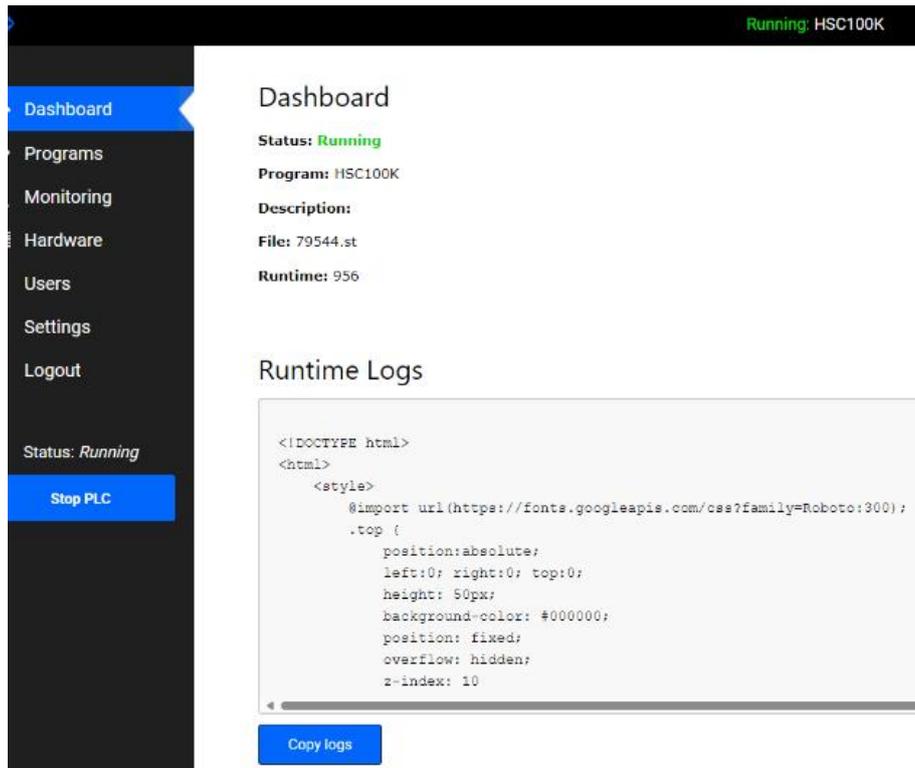


(3) 进入 WLAN 设置电脑 IP 地址并保存。



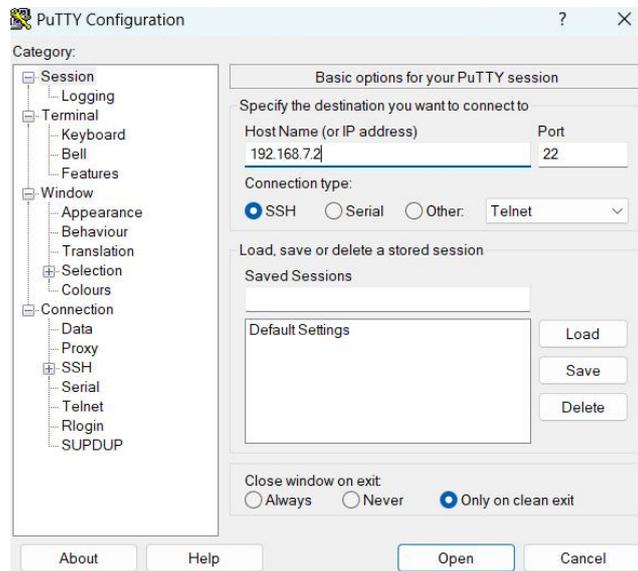
(4) 在浏览器输入 192.168.7.2:80 并进入。



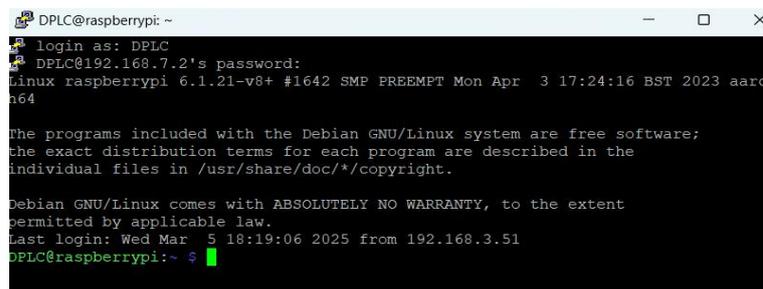


(5) 未接以太网用 USB 连接更新时间，操作以下步骤

1. 打开远程控制软件输入 IP 地址 192.168.7.2



2. 输入名称和密码，均为 DPLC，进入到控制面板



# Installation Manual

3.输入 `sudo date -s "2025-04-07 12:00:00"`指令即可，2025-04-07 为日期，12:00:00 为时间。

```
login as: DPLC
DPLC@192.168.7.2's password:
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

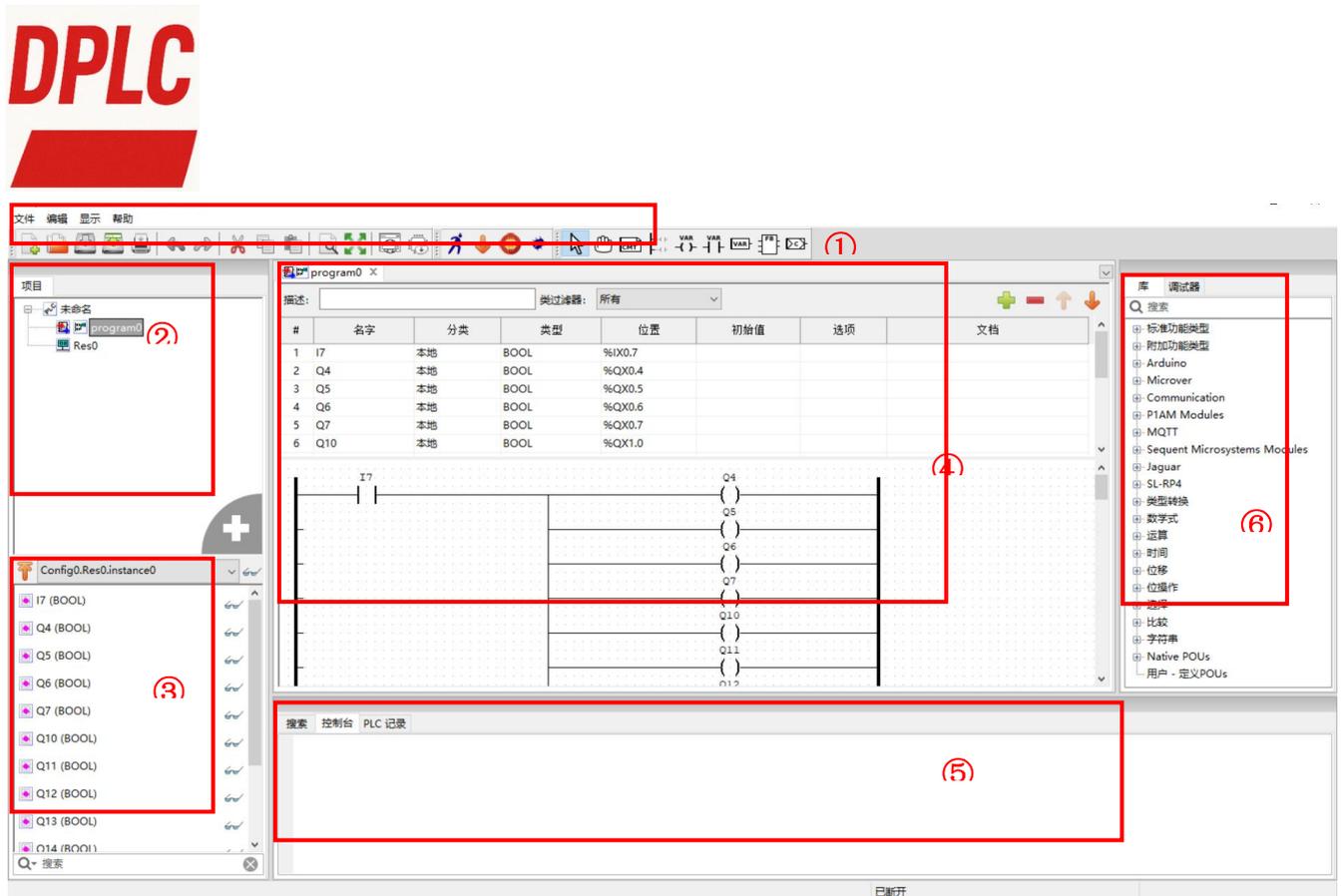
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr  7 11:44:57 2025 from 192.168.7.1
DPLC@raspberrypi:~$ sudo date -s "2025-04-07 12:00:00"
Mon  7 Apr 12:00:00 CST 2025
DPLC@raspberrypi:~$
```

4.长时间没有联网，想要连接以太网更新时间时，先上电开机，再插上网线。

## 2.2 编程界面

### 2.2.1 编译界面

双击“DPLC\_Editor”，打开编程软件。



DPLC\_Editor 编程界面中主要包含：

- ① 菜单工具栏
- ② 项目窗口
- ③ 调试窗口
- ④ 编程窗口

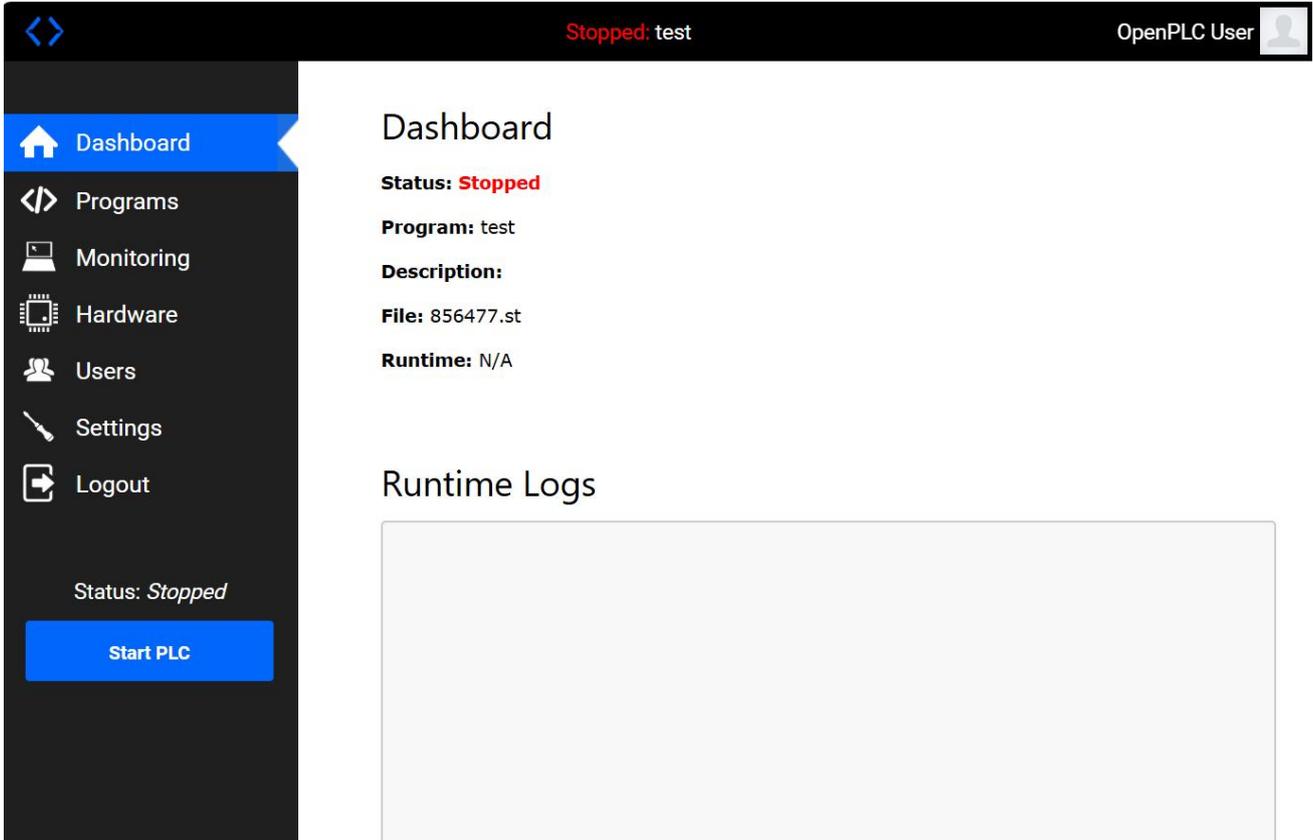


# Installation Manual

- ⑤消息窗
- ⑥工具箱

## 2.2.2 烧录界面

(1) Dashboard: Runtime 状态显示。



# Installation Manual

(2) Programs: 在此界面可以上传工程，查看历史工程。

Stopped: test OpenPLC User

## Programs

Here you can upload a new program to OpenPLC or revert back to a previous uploaded program shown on the table.

Program Name	File	Date Uploaded
test	856477.st	Mar 07, 2025 - 11:41AM
batch	207514.st	Nov 05, 2024 - 10:50AM
DVT	60319.st	Oct 30, 2024 - 07:04PM
test	191934.st	Oct 28, 2024 - 05:37PM
Demo	929879.st	Oct 15, 2024 - 06:25PM

List all programs

## Upload Program

选择文件 未选择文件 Upload Program 上传工程

Status: Stopped Start PLC

上传工程步骤参考 2.3.3 烧录。

(3) Hardware: 固定选择 Meanwell，无其他选择。

Stopped: test OpenPLC User

## Hardware

OpenPLC controls inputs and outputs through a piece of code called hardware layer (also known as driver). Therefore, to properly handle the inputs and outputs of your board, you must select the appropriate hardware layer for it. The Blank hardware layer is the default option on OpenPLC, which provides no support for native inputs and outputs.

**OpenPLC Hardware Layer**

Meanwell

Status: Stopped Start PLC

(4) Users: 添加登录的用户名和密码。初次使用有初始用户名和密码，如有需要可以自行添加。

Stopped: test OpenPLC User

## Users

Full Name	Username	Email
OpenPLC User	openplc	openplc@openplc.com

Add new user

Dashboard Programs Monitoring Hardware Users Settings Logout

Status: Stopped

Start PLC

Stopped: test OpenPLC User

## Add User

Name: John Appleseed

Username: username **新增登录此烧录界面的用户**

Email: your@email.com

Password: password **新增登录此烧录界面的密码**

Picture: 选择文件 未选择文件

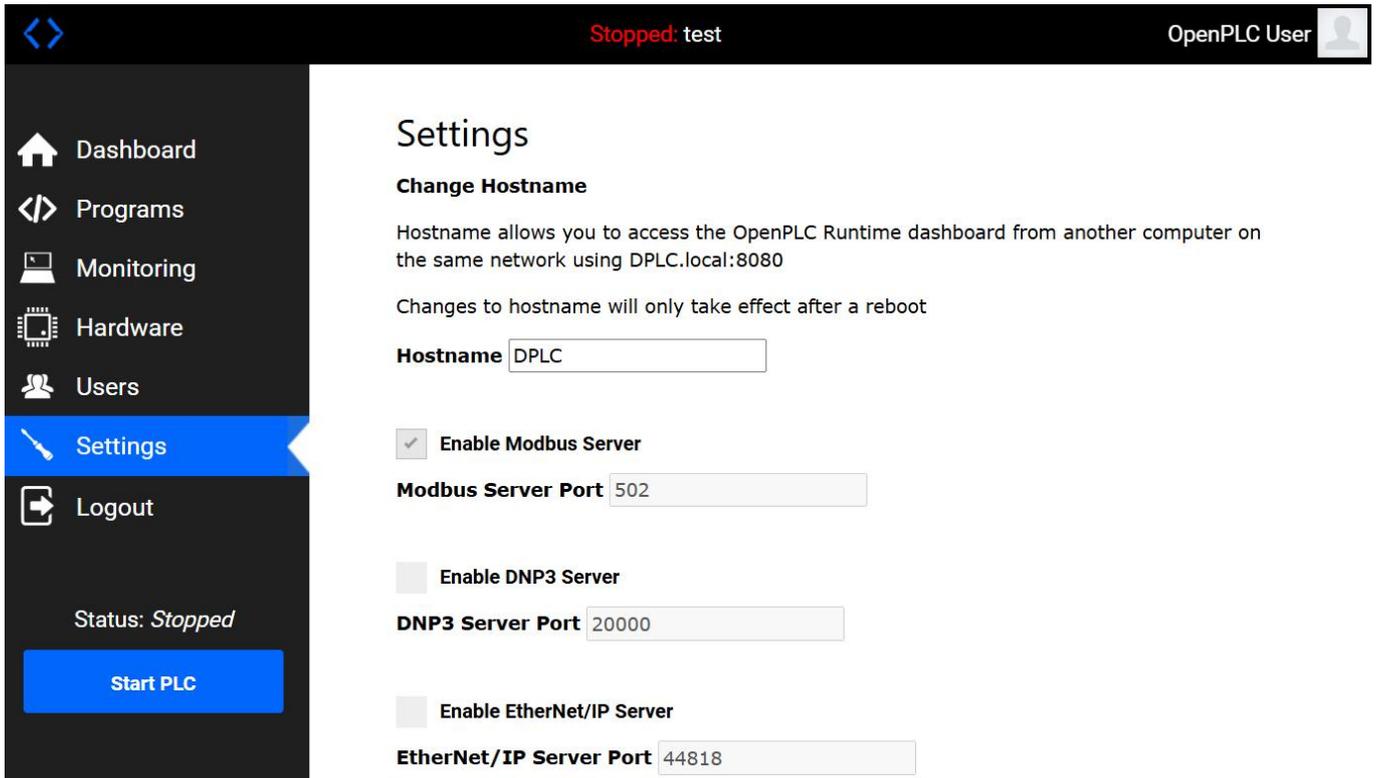
Save user

Dashboard Programs Monitoring Hardware Users Settings Logout

Status: Stopped

Start PLC

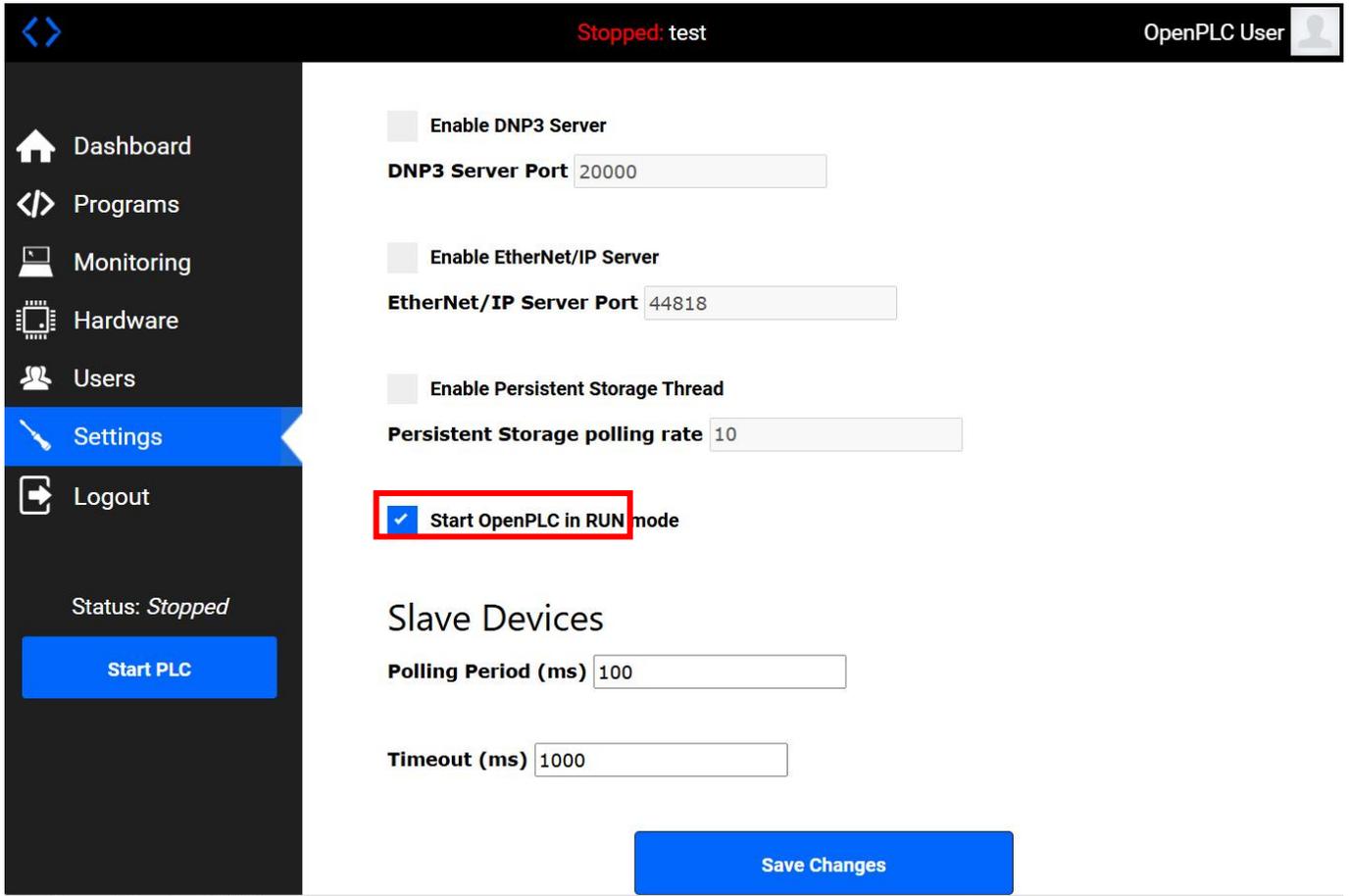
(5) Settings: 关于烧录的设置。



The screenshot shows the OpenPLC Settings interface. The top bar indicates the system is "Stopped: test" and the user is "OpenPLC User". The left sidebar contains navigation options: Dashboard, Programs, Monitoring, Hardware, Users, Settings (highlighted), and Logout. The main content area is titled "Settings" and includes a "Change Hostname" section. Below this, there are three server configuration options: "Enable Modbus Server" (checked), "Enable DNP3 Server" (unchecked), and "Enable EtherNet/IP Server" (unchecked). Each option has a corresponding port number in a text input field.

Server Type	Enabled	Port
Modbus Server	<input checked="" type="checkbox"/>	502
DNP3 Server	<input type="checkbox"/>	20000
EtherNet/IP Server	<input type="checkbox"/>	44818

建议勾选“Start OpenPLC in RUN mode”，退出烧录界面时，可以保持 RUN 状态。



Stopped test OpenPLC User

- Enable DNP3 Server  
DNP3 Server Port: 20000
- Enable EtherNet/IP Server  
EtherNet/IP Server Port: 44818
- Enable Persistent Storage Thread  
Persistent Storage polling rate: 10
- Start OpenPLC in RUN mode

### Slave Devices

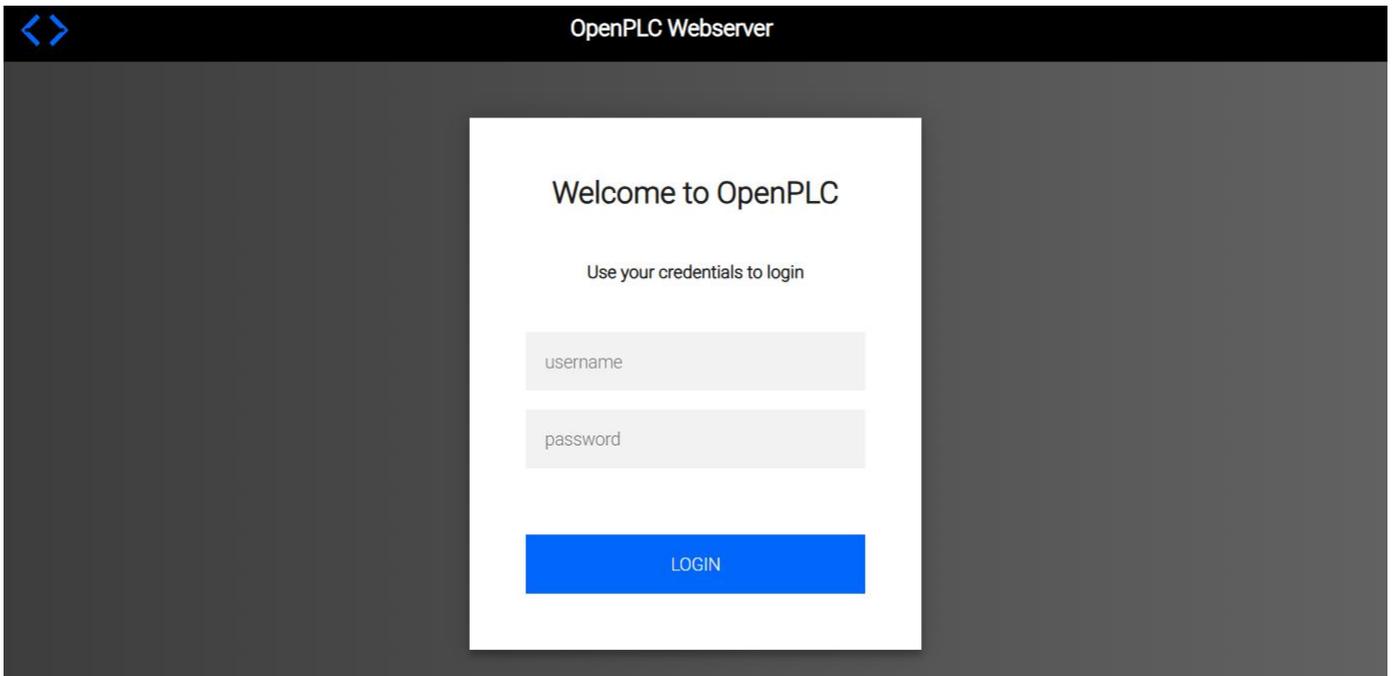
Polling Period (ms): 100

Timeout (ms): 1000

Start PLC

Save Changes

(6) Logout: 退出烧录界面，需要再次输入用户名、密码进入。



OpenPLC Webserver

Welcome to OpenPLC

Use your credentials to login

username

password

LOGIN

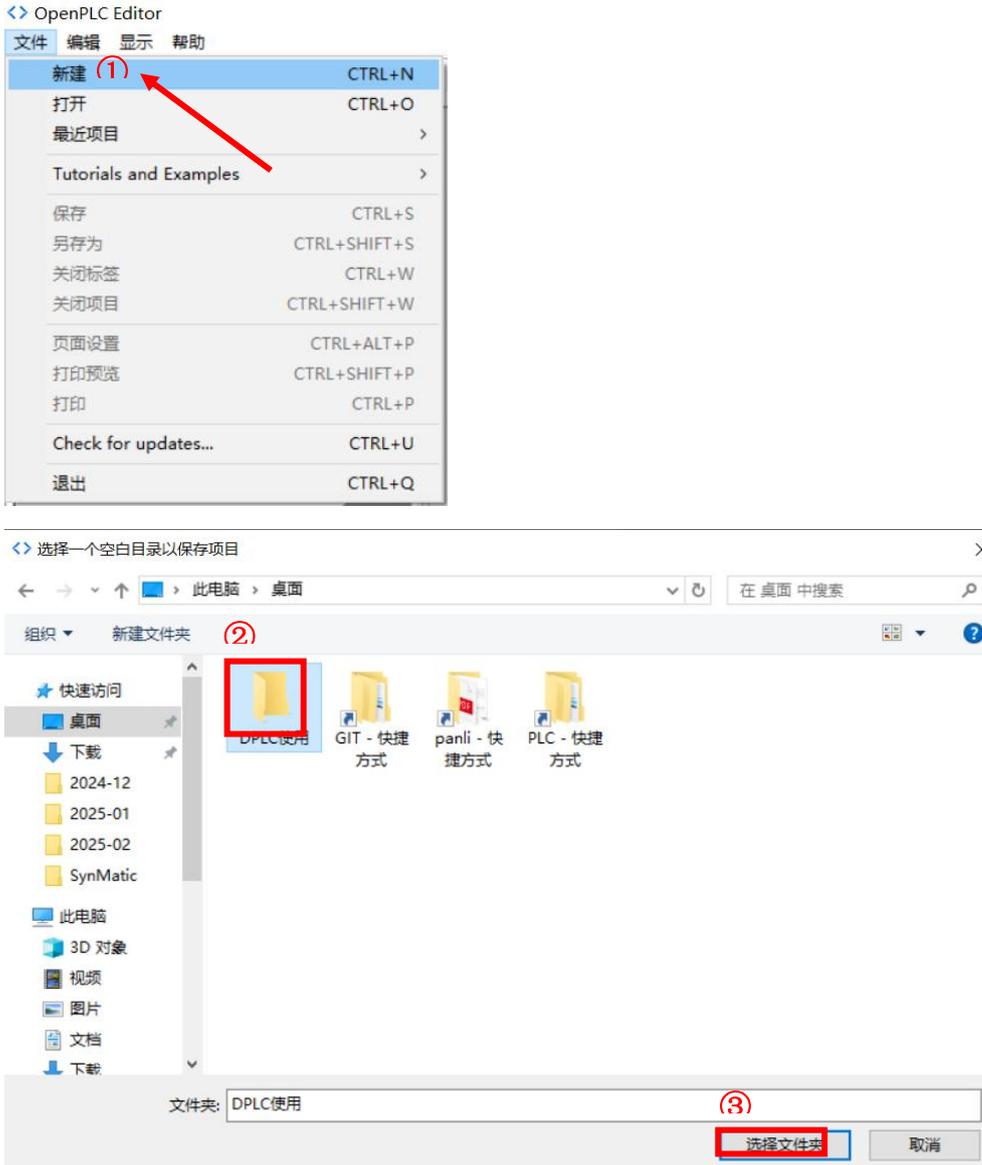
## 2.3 使用说明

### 2.3.1 新建工程

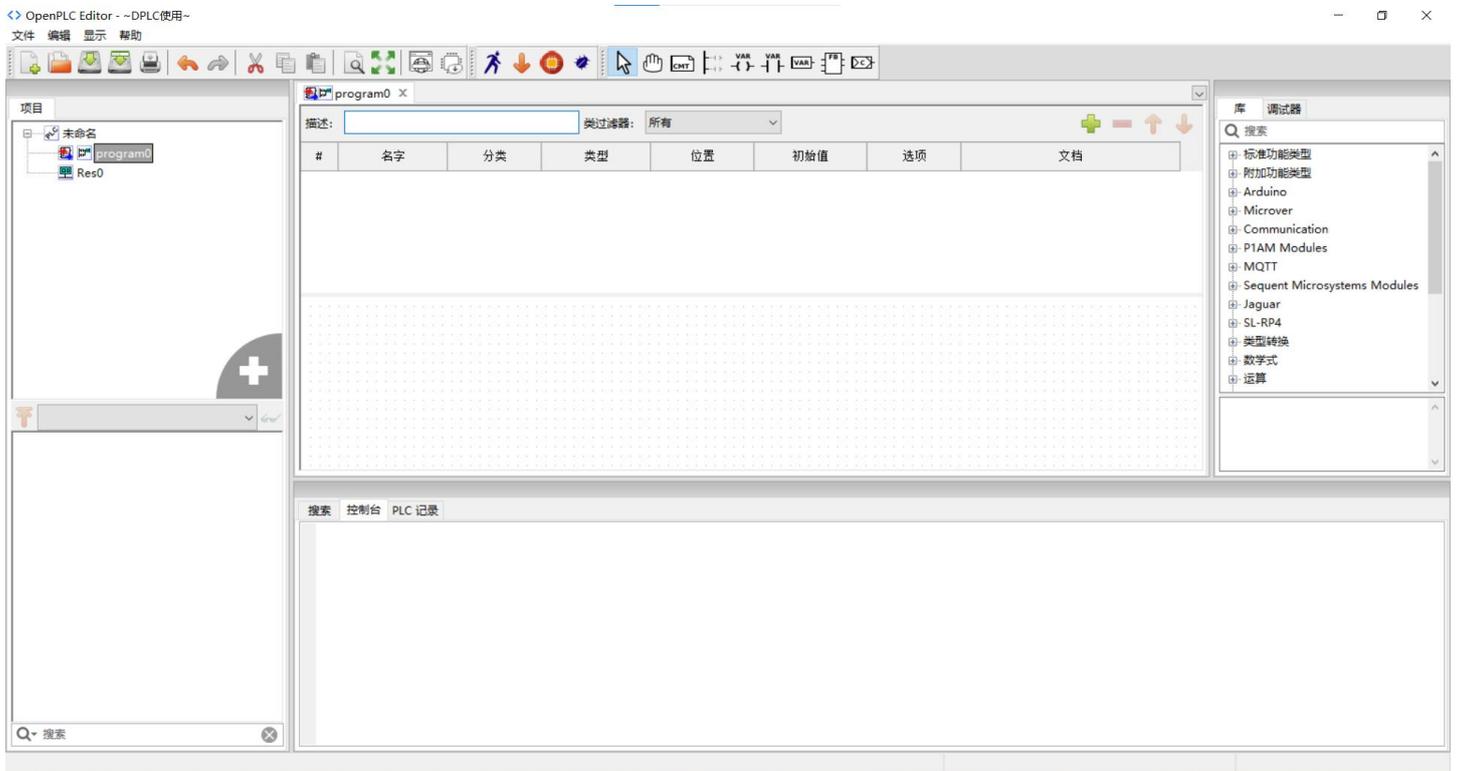
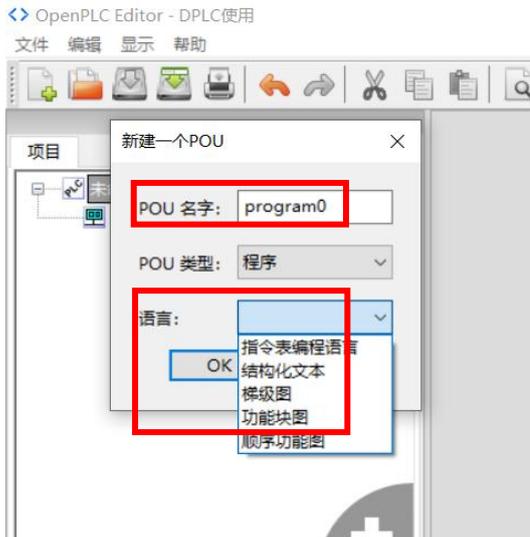
双击“DPLC\_Editor”，打开编程软件。



文件->新建->选择空白文件夹，不是空白文件夹会新建失败!!!



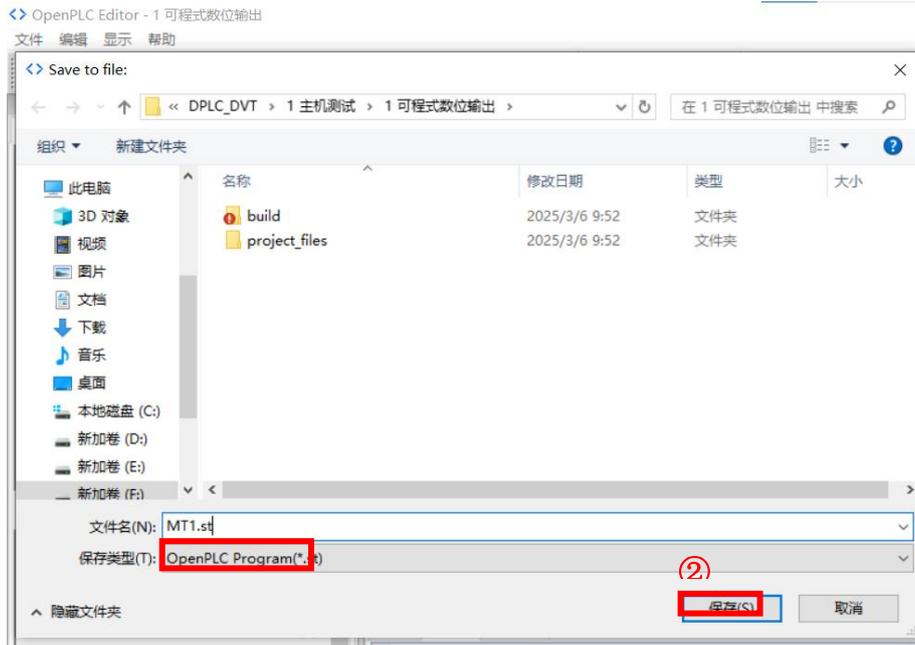
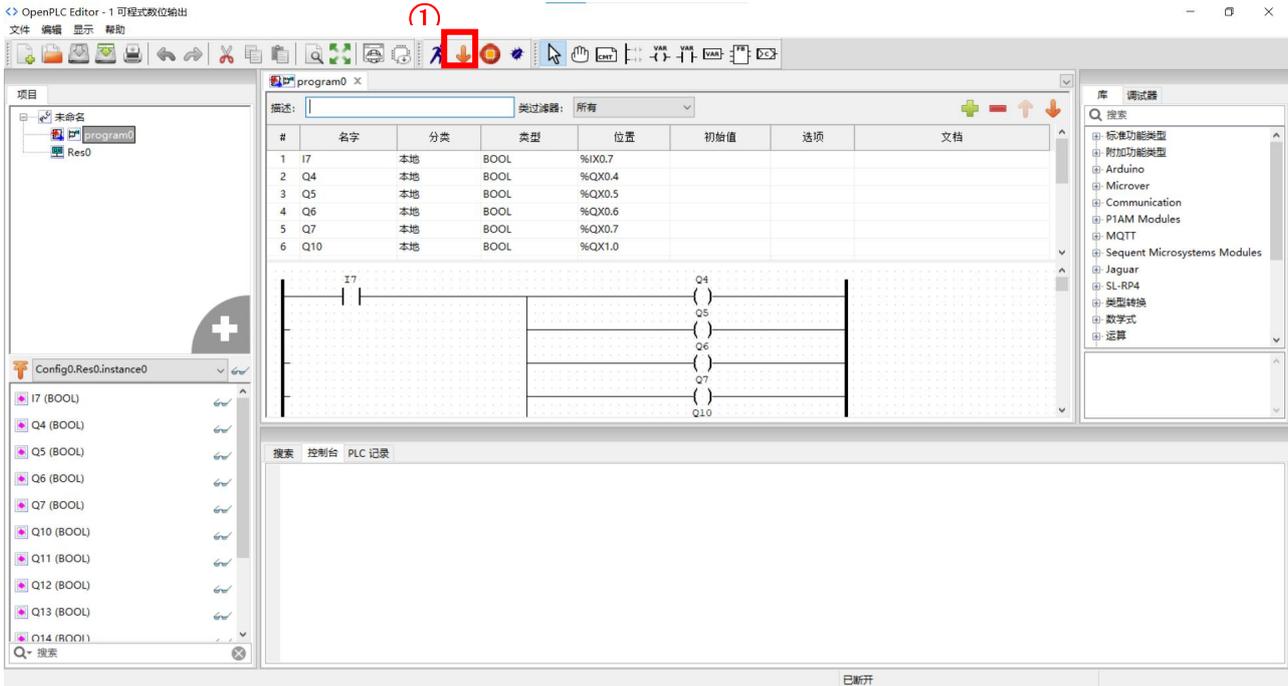
新建一个项目后，可以修改项目名称和编译语言，修改后，单击 OK。



## 2.3.2 编译

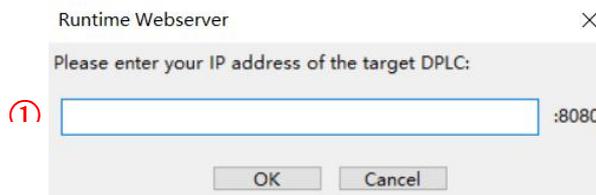
电脑与 DPLC 使用同一路由器连接。点击菜单工具栏“Generate program for OpenPLC Runtime”->生成.st 文件，保存。

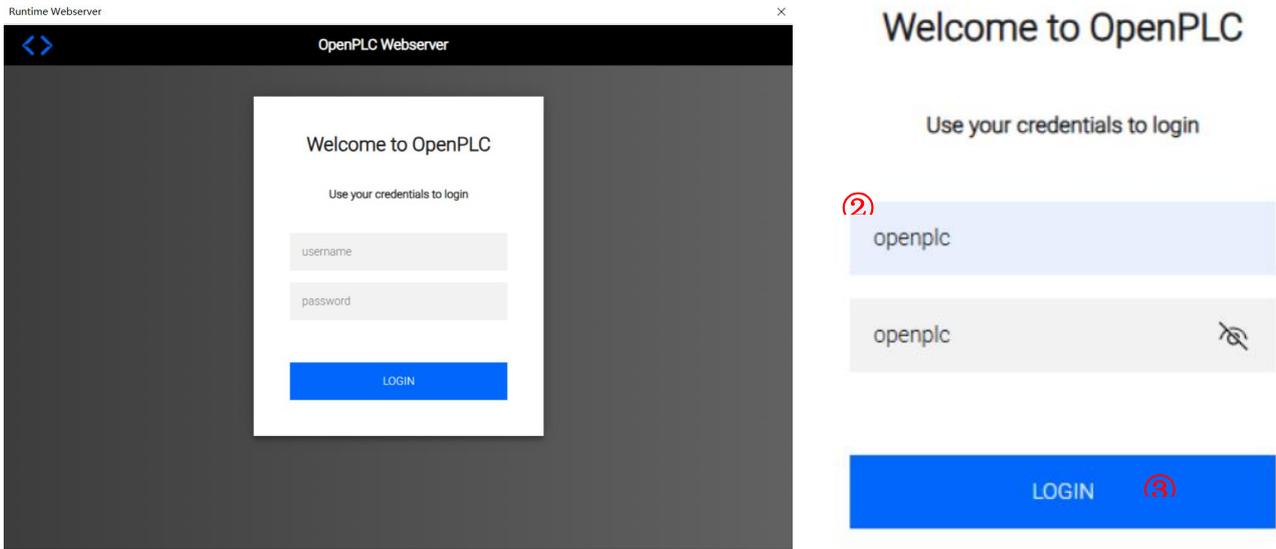
# Installation Manual



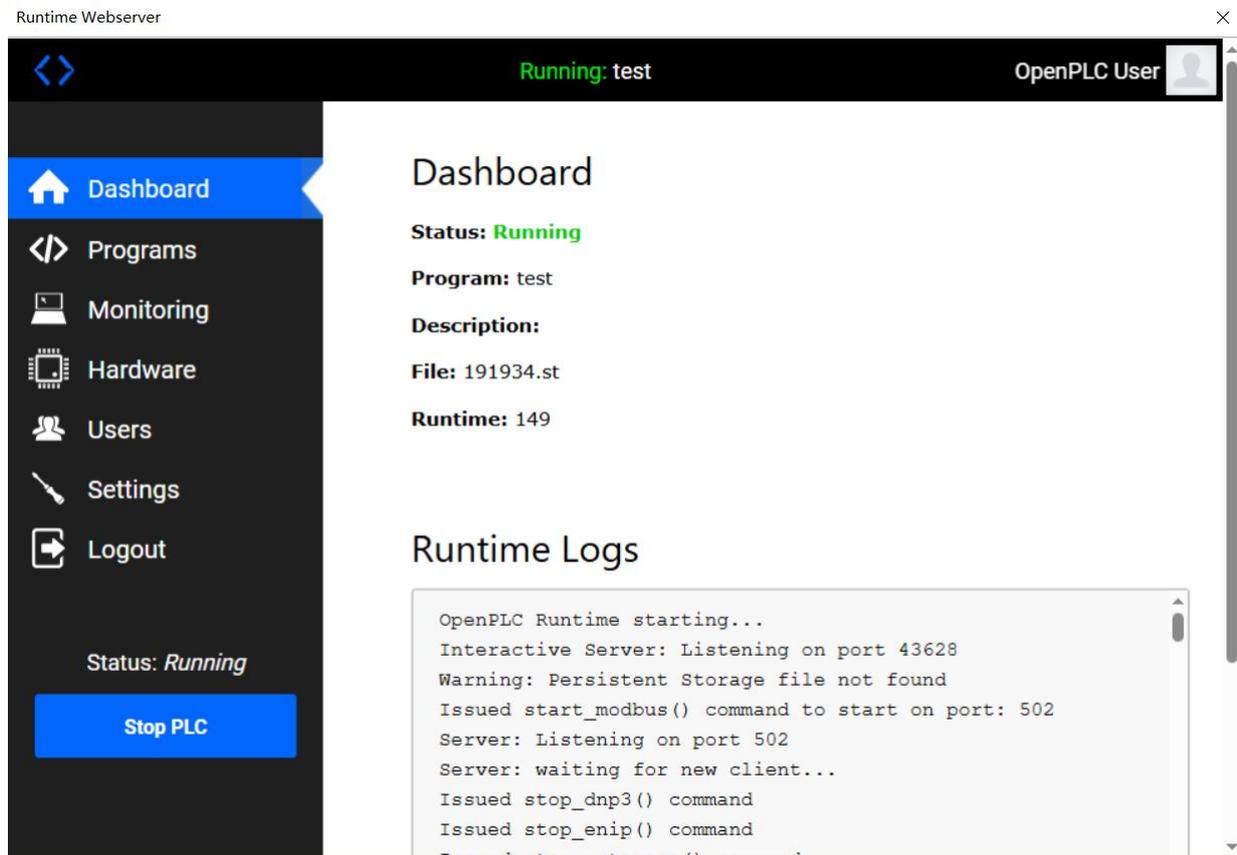
## 2.3.3 烧录

点击菜单工具栏“Runtime server”->输入 DPLC 的 IP->输入用户名和密码，默认用户名和密码都为 openplc->点击 LOGIN。

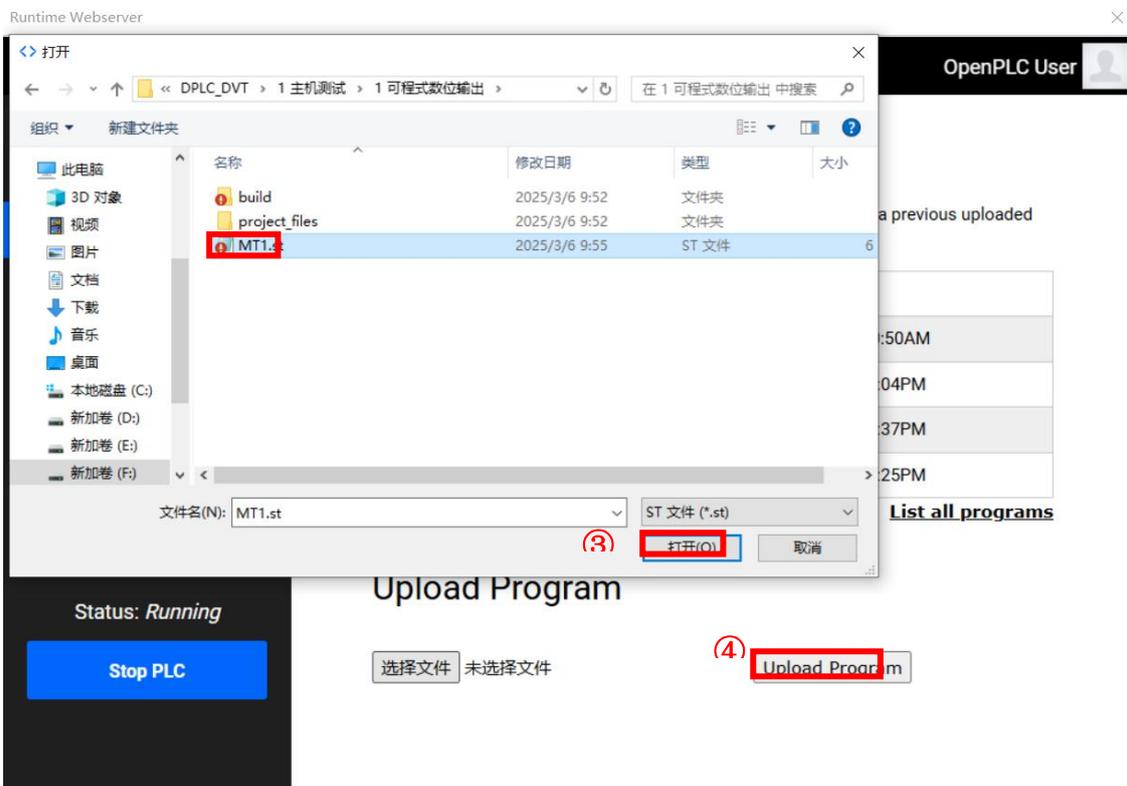
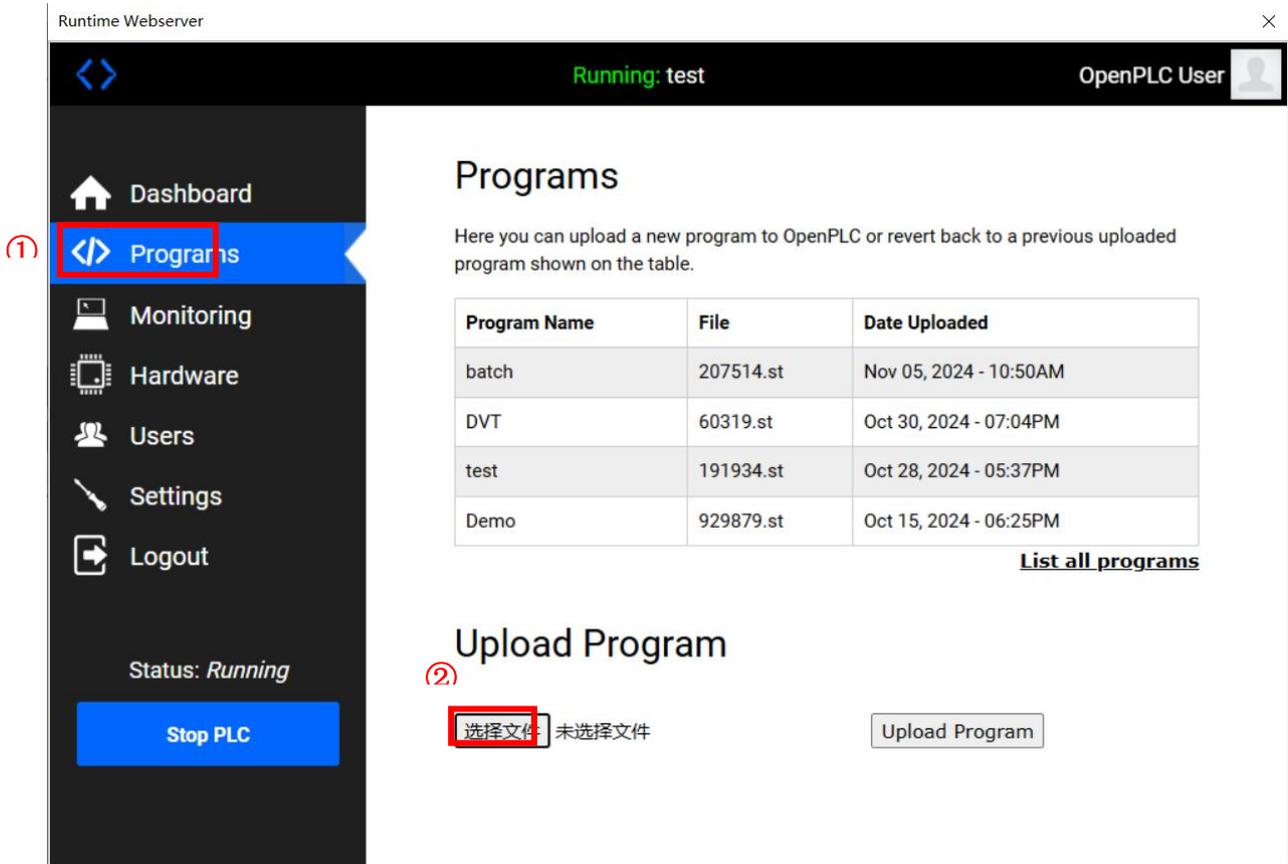




进入界面如下图所示：



单击“Programs”->选择文件->选择上面生成的.st 文件->点击“Upload Program”，完成文件上传。



重新命名项目，也可加上描述->点击“Upload program”等待加载完成后->点击“Go to Dashboard”。

# Installation Manual

Runtime Webservice

Running: test

OpenPLC User

Name: test **命名**

Description: Insert the program description here **描述**

File: 238244.st

Date Uploaded: Mar 07, 2025 - 10:30AM

**Upload program**

Status: Running

Stop PLC

Dashboard

Programs

Monitoring

Hardware

Users

Settings

Logout

Runtime Webservice

Compiling: test

OpenPLC User

### Compiling program

```
Generating C files...
POUS.c
POUS.h
LOCATED_VARIABLES.h
VARIABLES.csv
Config0.c
Config0.h
Res0.c
Moving Files...
Compiling for Raspberry Pi
Generating object files...
Generating glueVars...
varName: __IX0_7      varType: BOOL
varName: __QX0_4      varType: BOOL
varName: __QX0_5      varType: BOOL
varName: __QX0_6      varType: BOOL
varName: __QX0_7      varType: BOOL
varName: __QX1_0      varType: BOOL
varName: __QX1_1      varType: BOOL
varName: __QX1_2      varType: BOOL
```

**Go to Dashboard**

Dashboard

Programs

Monitoring

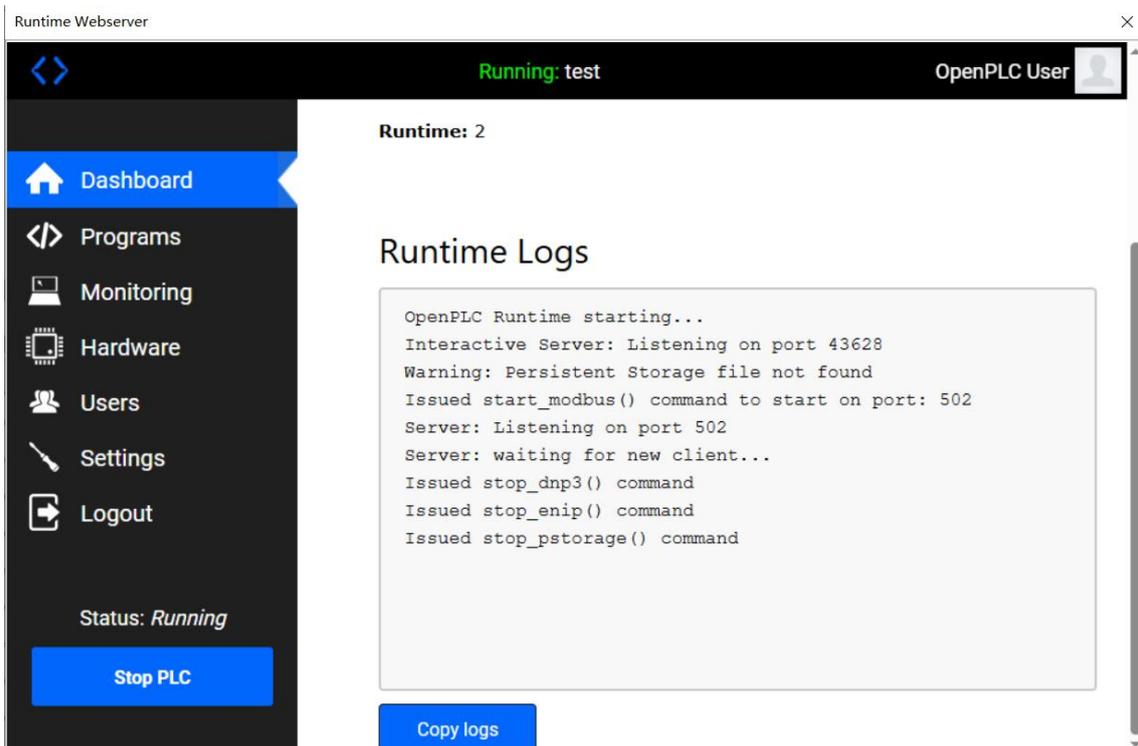
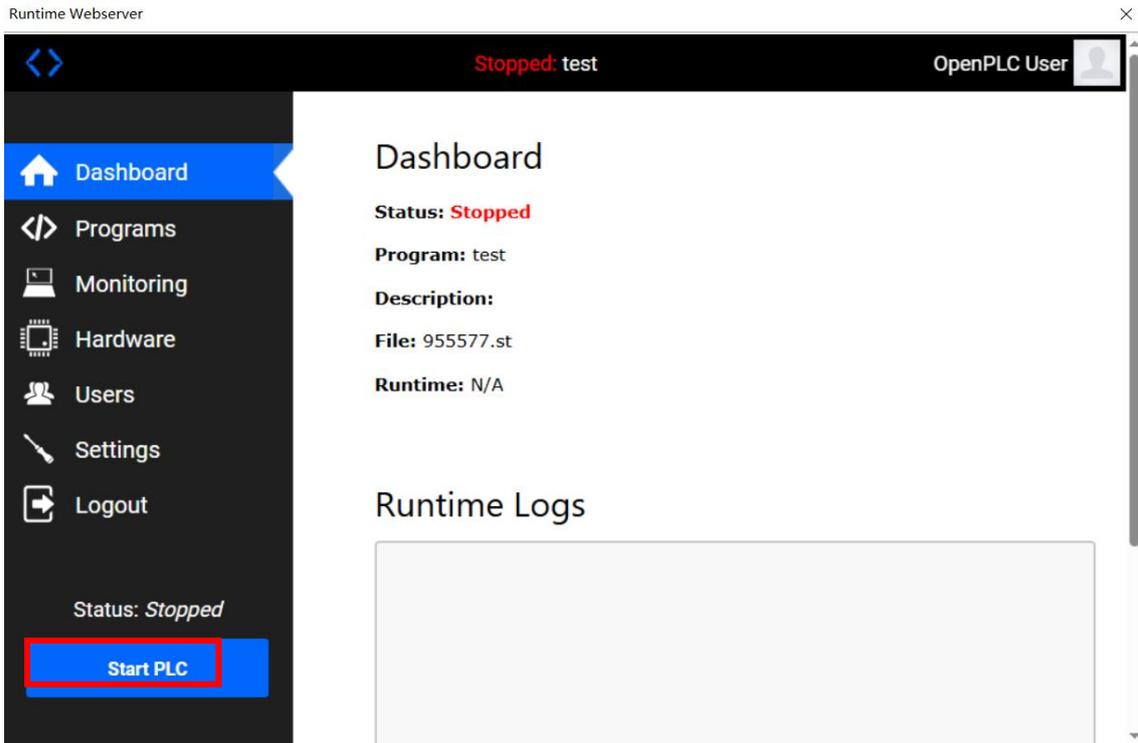
Hardware

Users

Settings

Logout

点击“Start PLC”，程序开始运行。

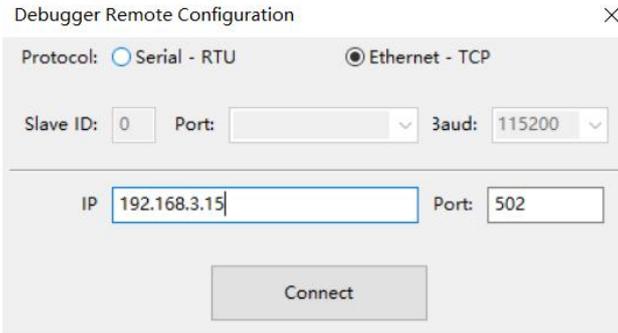


## 2.3.4 调试

点击菜单工具栏中的“Live debug remote PLC”->在弹出的窗口输入 DPLC 的 IP，再点击“Connect”



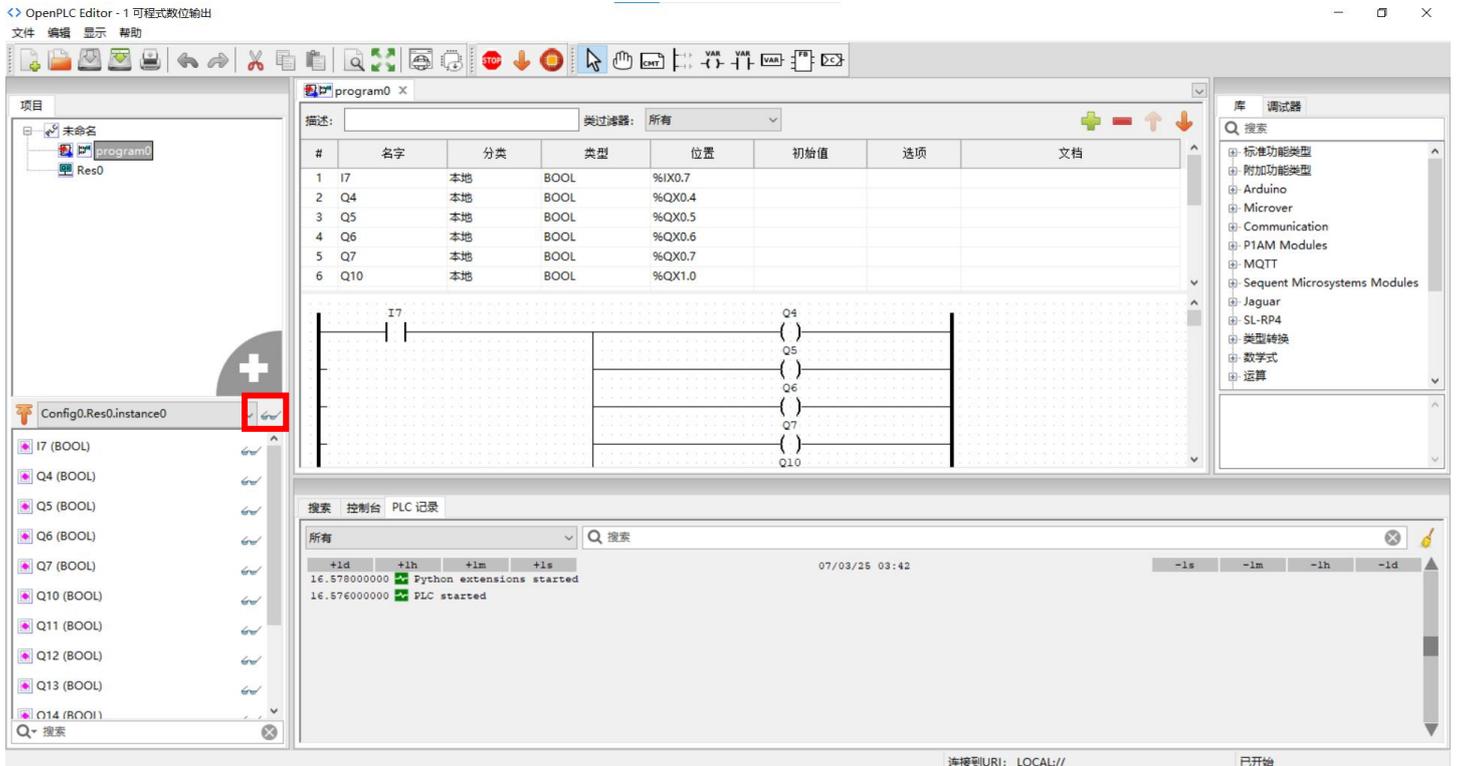
# Installation Manual

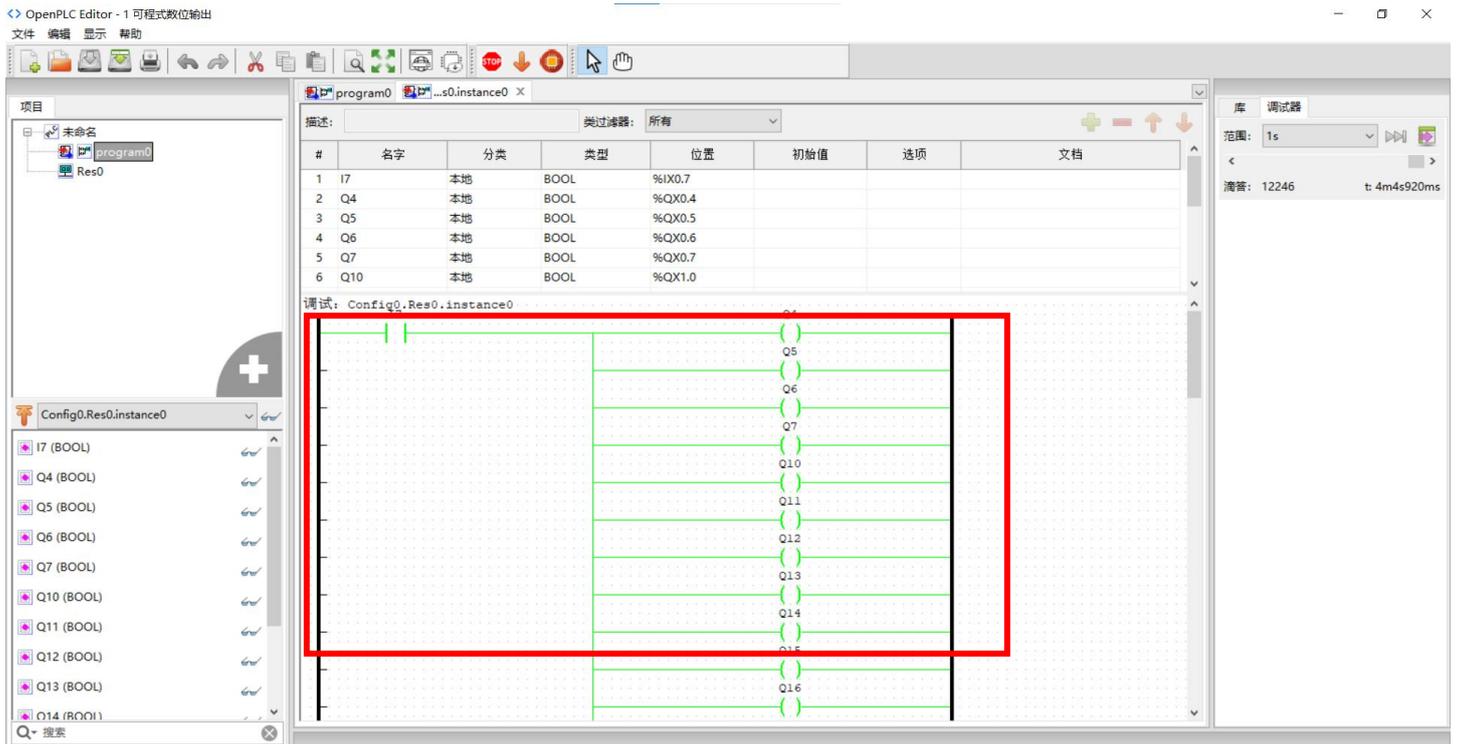


Connect 后，消息窗口显示如下：

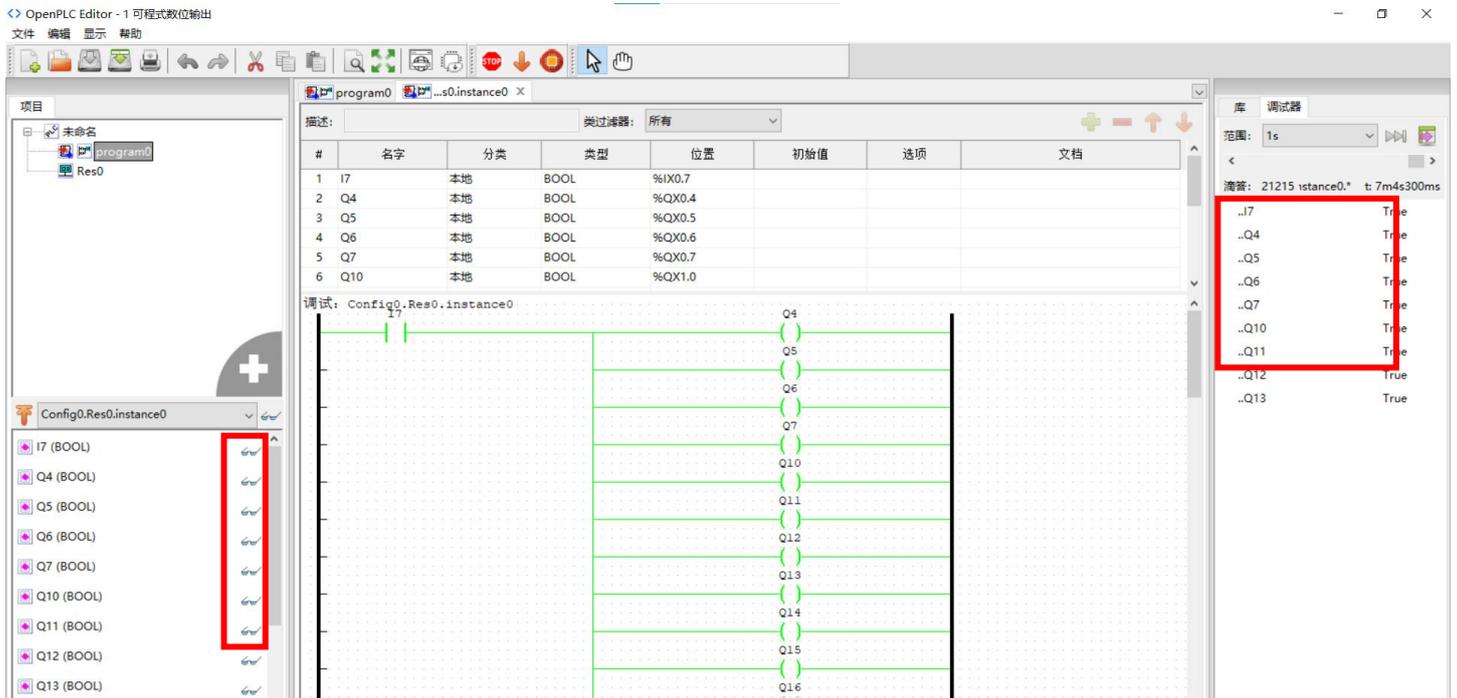


点击调试窗口的“调试实例”，就可以 debug 了。





点击每一个变量的“调试”，右侧调试器中可以监控每个变量的状态。

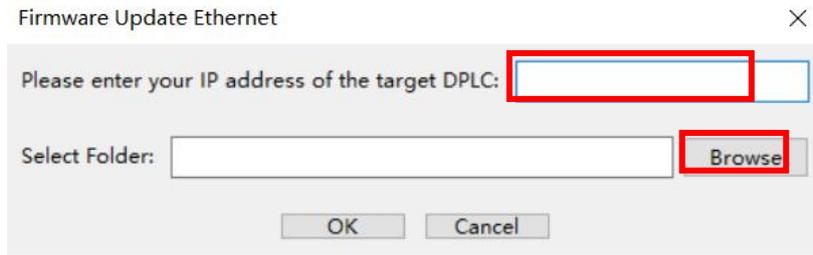


### 2.3.5 韧体更新 Firmware Update Ethernet

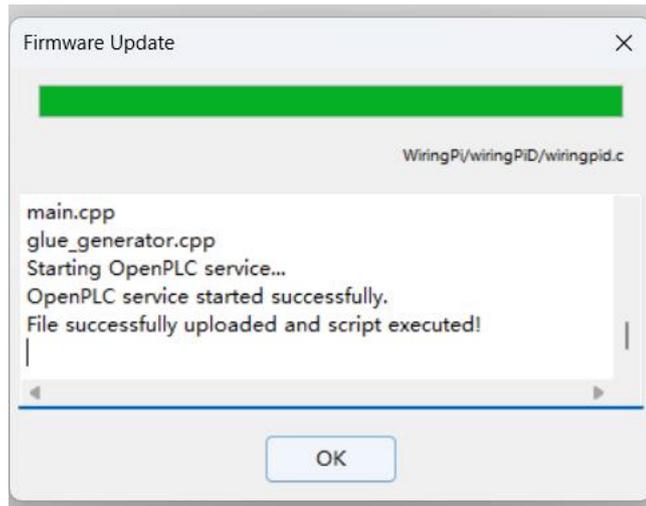
点击菜单工具栏中的“Firmware Update Ethernet”->在弹出的窗口，输入 DPLC 的 IP，上传要更新的韧体文件->点击 OK。



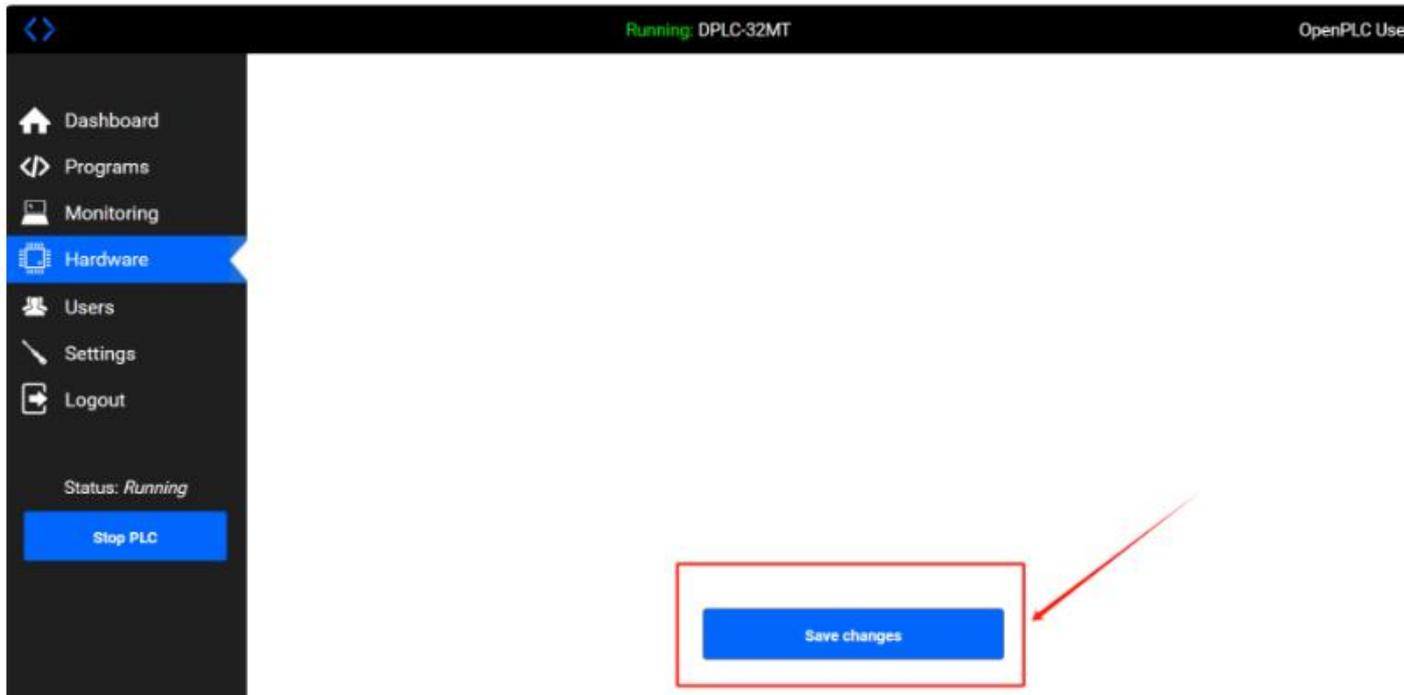
# Installation Manual



上传完的界面如下所示



进入 80 端口网页，点击 Hardware 硬件层选择，然后点击 Save changes 进行编译，即完成固件更新。



## 三、编程操作

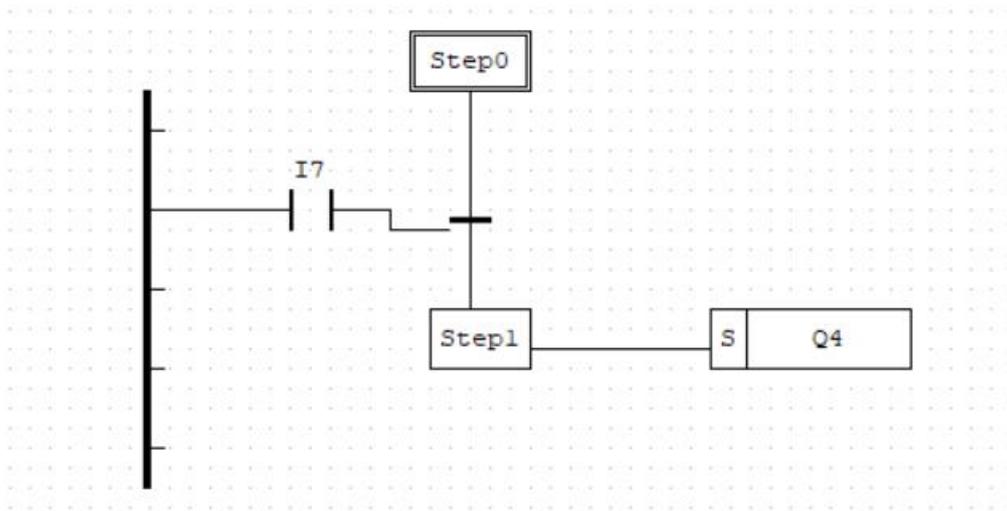
### 3.1 编程方式

OpenPLC Editor 常用的编程方式：梯形图 (LD)、结构化文本 (ST) 和顺序功能图 (SFC)。

- 梯形图：直观方便，是大多数 PLC 编程人员和维护人员选择的方法。
- 结构化文本：类似于高级编程语言的 PLC 编程方式，使用结构化文本的编程方式更容易完成复杂的算法控制。
- 顺序功能图：是一种图形化的功能性说明语言，适用于复杂控制流程的表达。

以下是三种语言对于同一编程逻辑的表达：

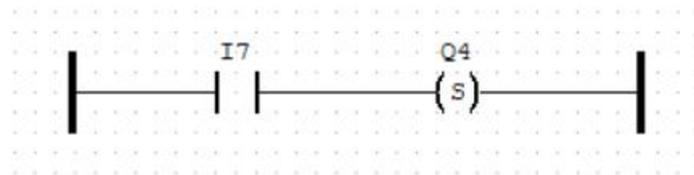
#### (1) 顺序功能图 (SFC)



#### (2) 结构化文本 (ST)

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

#### (3) 梯形图 (LD)



注意：

编程数据格式为十进制。

### 3.2 梯形图的编辑

## 3.2.1 地址映射

### (1) 输入映射

#	名字	分类	类型	位置	初始值
1	sensor0	本地	BOOL	%IX0.0	
2	sensor1	本地	BOOL	%IX0.1	
3	IW0	本地	UINT	%IW0	

名字：命名的输入设备（名称需要用全英文）。

分类：默认选择 Local，无需更改。

类型：数据类型，高低电平输入数据类型为 BOOL，类比输入数据类型为 UINT。

位置：映射地址，高低电平的地址%IX0.0 对应 I0.0，%IX0.1 对应 I0.1，以此类推；类比输入的地址%IW0 对应 IW0，%IW1 对应 IW1，以此类推。

初始值：初始值，有需要可以自行设置。

### (2) 输出映射

#	名字	分类	类型	位置	初始值
4	LED0	本地	BOOL	%QX0.0	
5	LED1	本地	BOOL	%QX0.1	

名字：命名的输出设备（名称需要用全英文）。

分类：默认选择 Local，无需更改。

类型：数据类型，输出数据类型为 BOOL。

地址：映射地址，地址%QX0.0 对应 Q0.0，%QX0.1 对应 Q0.1，以此类推。

初始值：初始值，有需要可以自行设置。

### (3) 继电器映射

#	名字	分类	类型	位置	初始值
6	MB0	本地	BYTE	%MB0	
7	MX0	本地	BOOL	%MX0	
8	MW0	本地	INT	%MW0	
9	MD0	本地	DINT	%MD0	

名字：命名的继电器（名称需要用全英文）。

分类：默认选择 Local，无需更改。

类型：数据类型，MB 数据类型为 Byte，MX 数据类型为 BOOL，MW 数据类型为 UINT/INT，MD 的数据类型为 UDINT/DINT。

地址：映射地址，地址%MB0 对应 MB0，%MB1 对应 MB1，以此类推；%MX0 对应 MX0，%MX1 对应 MX1，以此类推；%MW0 对应 MW0，%MW1 对应 MW1，以此类推；%MD0 对应 MD0，%MD1 对应 MD1，以此类推。

初始值：初始值，有需要可以自行设置。

## 3.2.2 软元件介绍

### (1) 输入触点

# Installation Manual

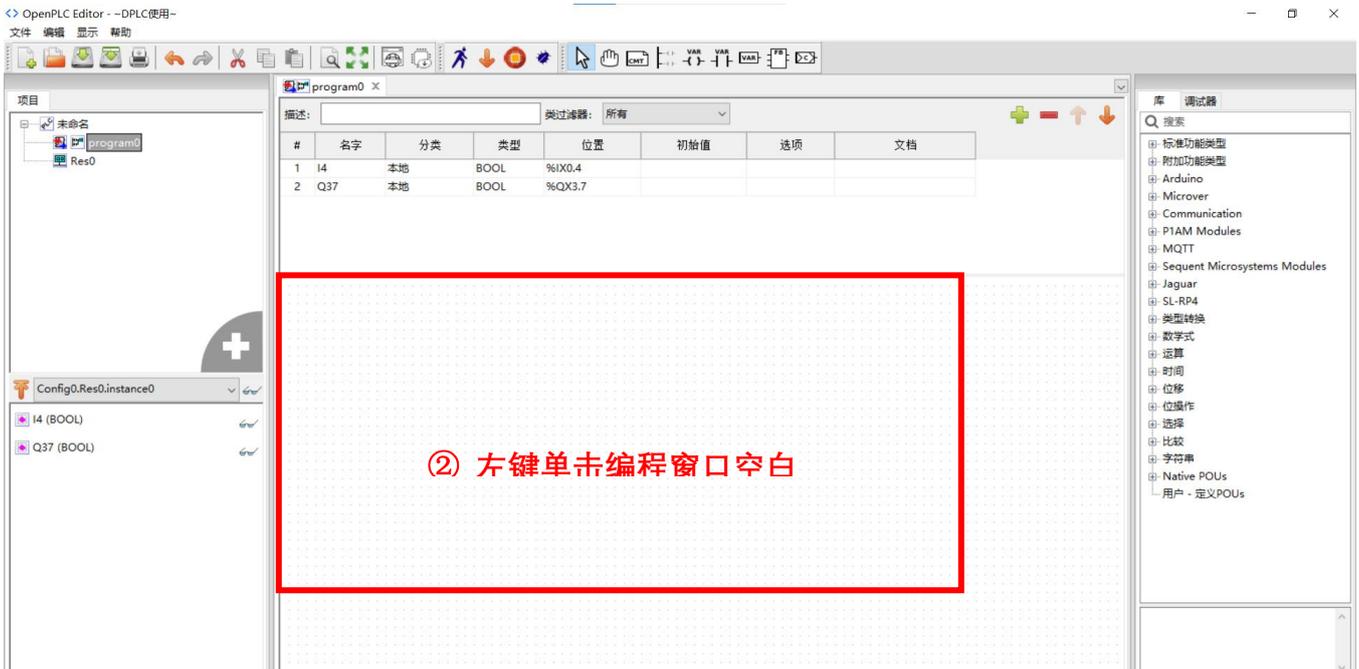
图标	功能
	常开触点
	常闭触点
	上升沿触点
	下降沿触点

插入触点示例：

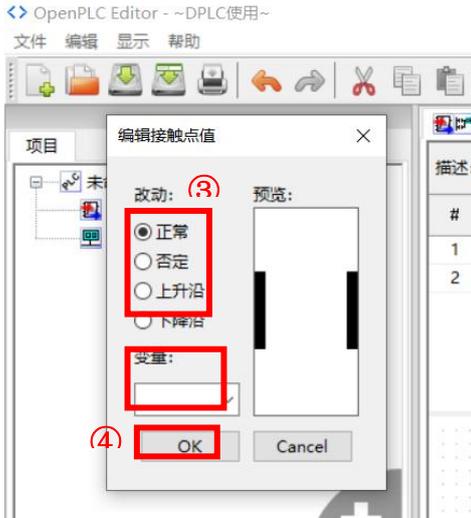
点击菜单工具栏的“新建一个接触点”。



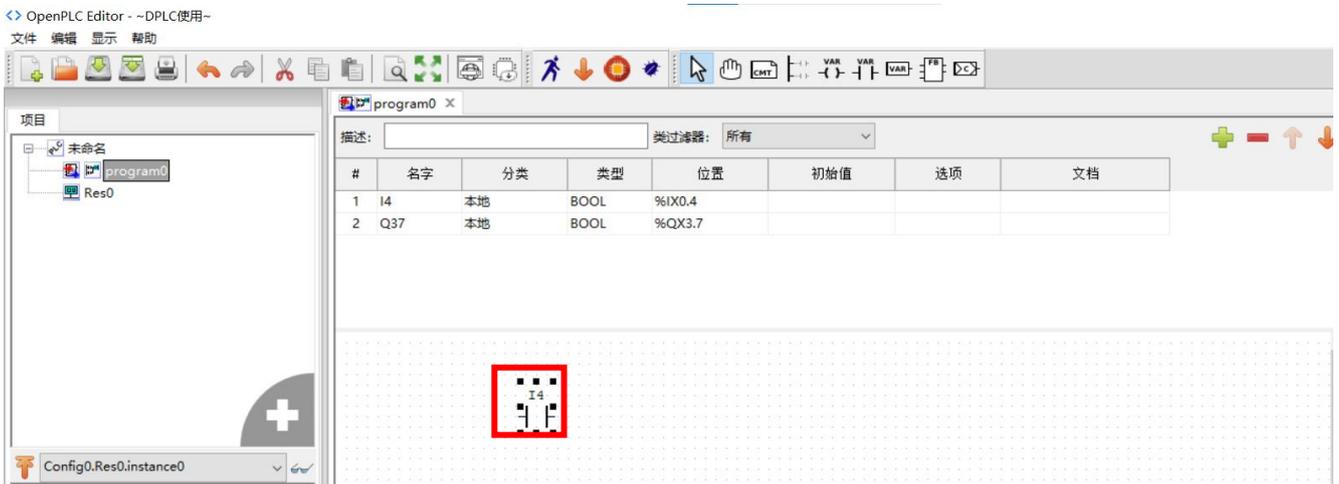
再点击编程窗口的任意空白处。



此时，输入触点选择窗口弹出，选择输入触点，并且映射到对应的变量。选择完毕后，点击 OK。



编程窗口出现新建的触点：



## (2) 输出线圈

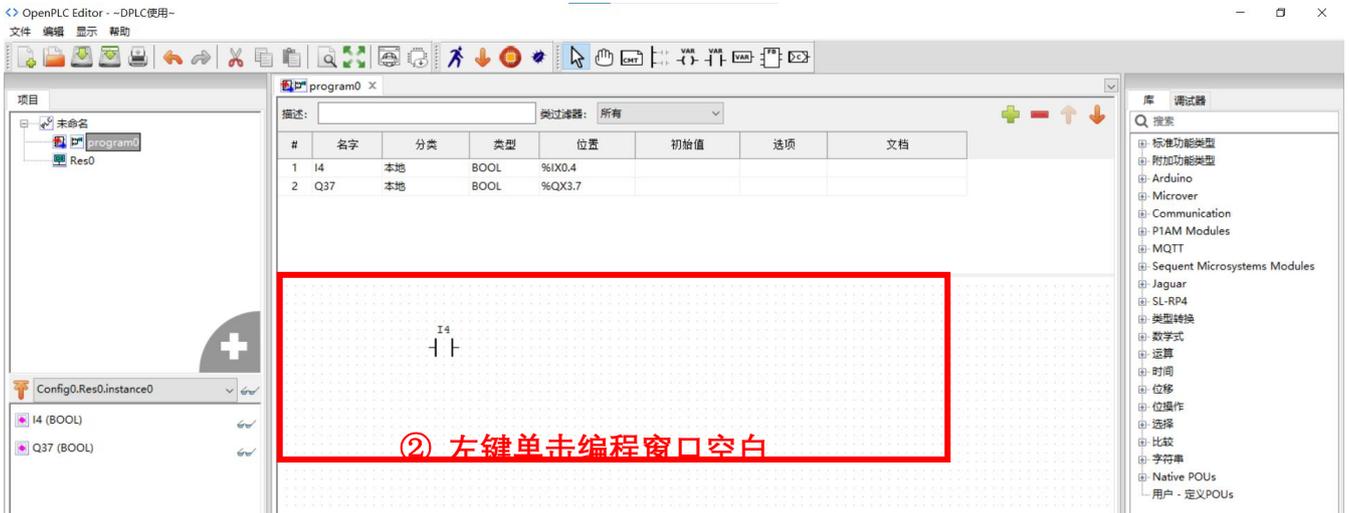
图标	功能
	赋值线圈
	赋值取反线圈
	置位线圈
	复位线圈
	上升沿线圈
	下降沿线圈

插入线圈示例：

点击菜单工具栏的“新建一个线圈”。



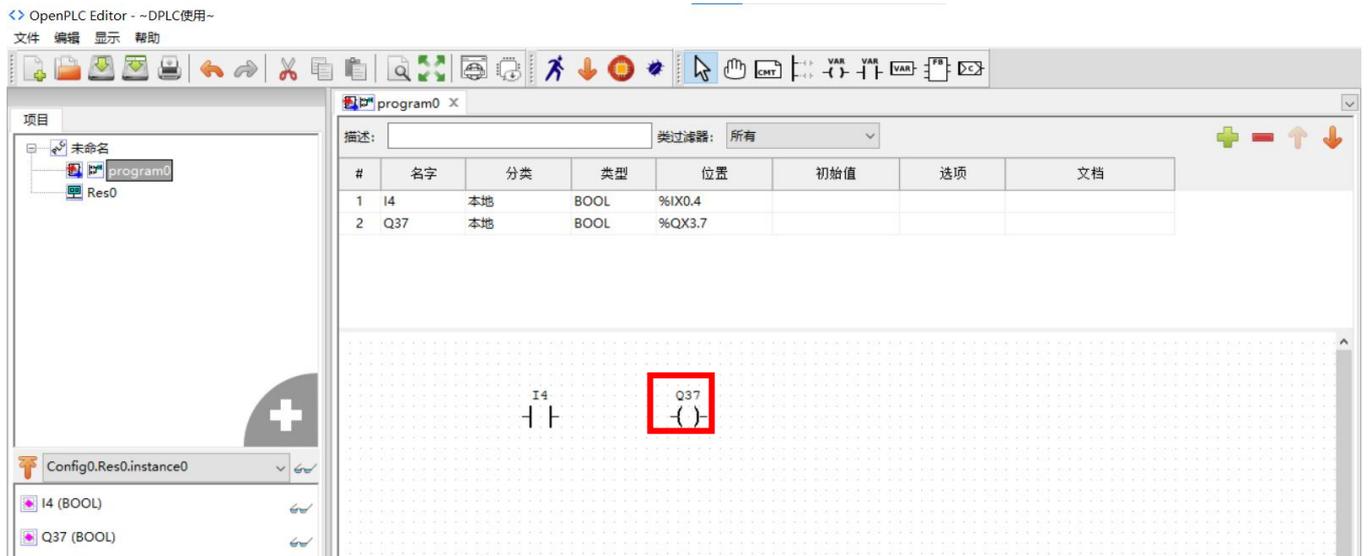
再点击编程窗口的任意空白处。



此时，输出线圈选择窗口弹出，选择输出线圈，并且映射到对应的变量。选择完毕后，点击 OK。



编程窗口出现新建的线圈：



拖动输入触点的最右侧，将两者相连。



### (3) 电源导轨

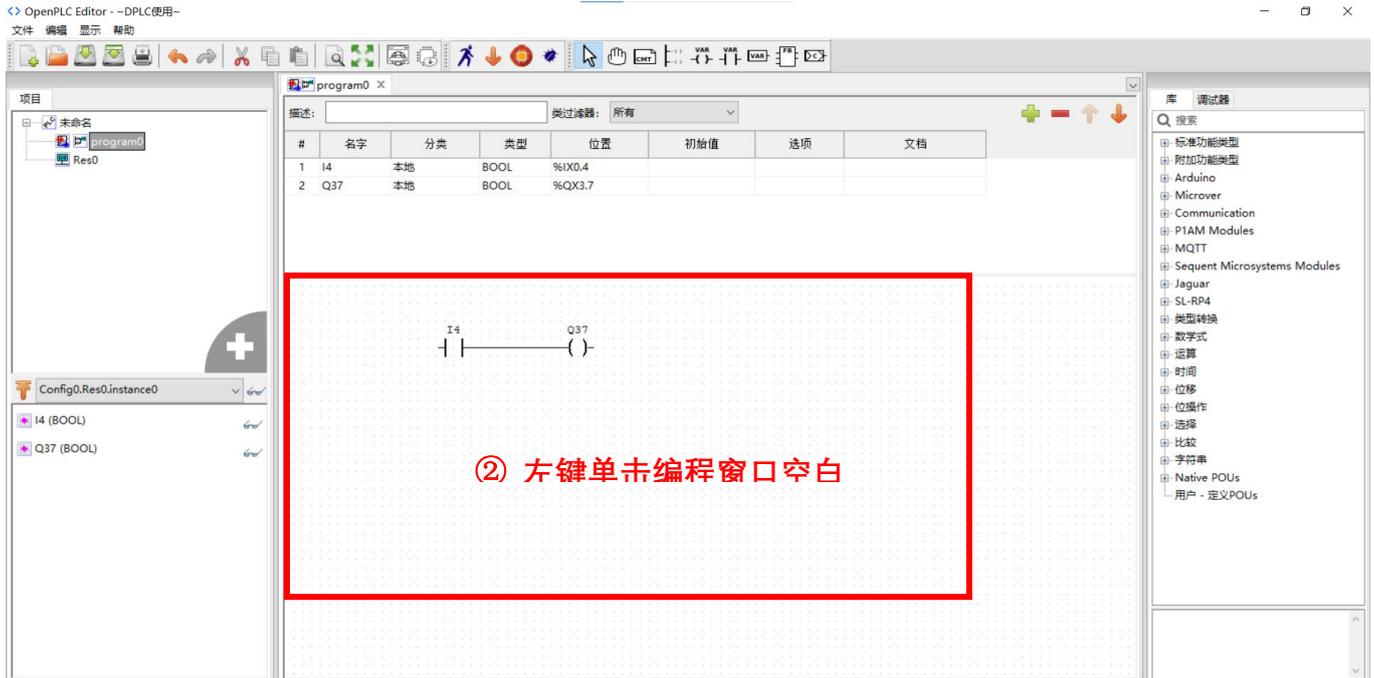
图标	功能
	左电源导轨
	右电源导轨

插入导轨示例：

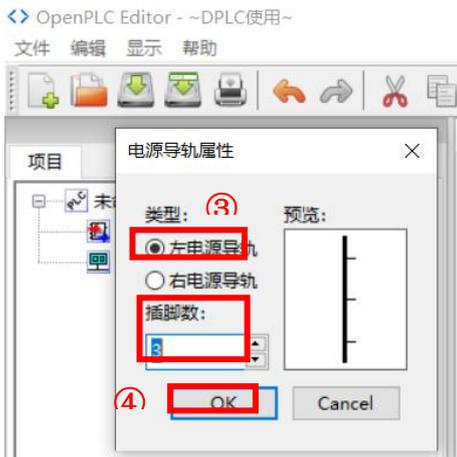
点击菜单工具栏的“新建一个电源导轨”。



再点击编程窗口的任意空白处。

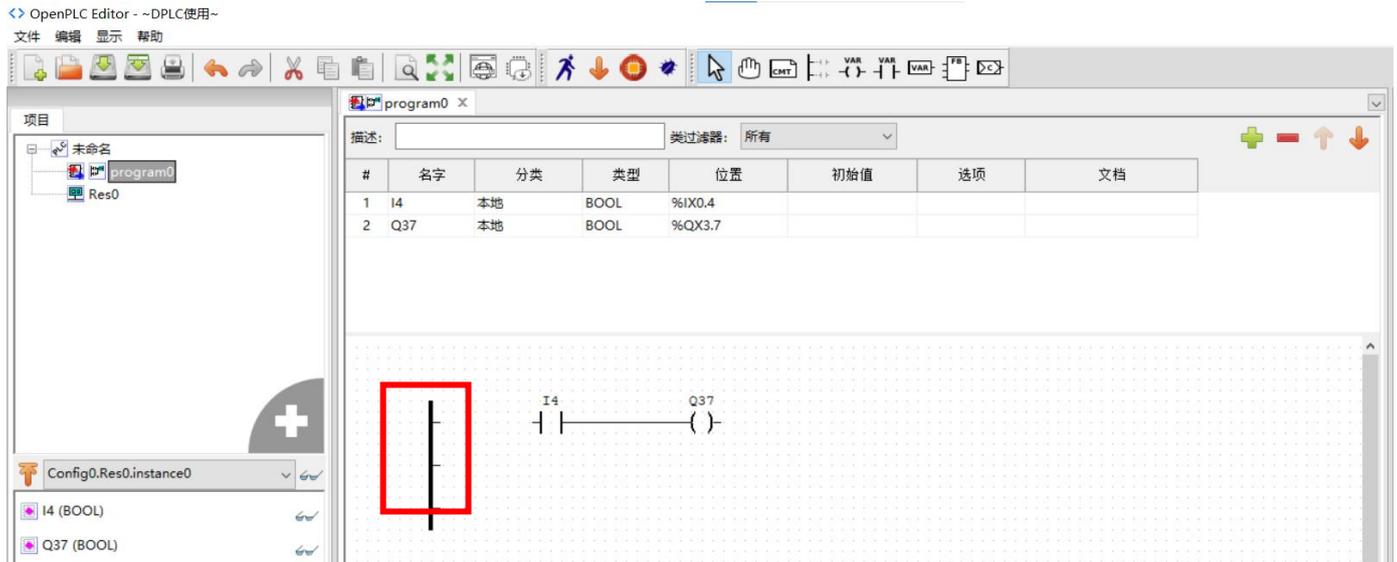


此时，电源导轨选择窗口弹出，选择左电源导轨，并且选择插脚数。选择完毕后，点击 OK。

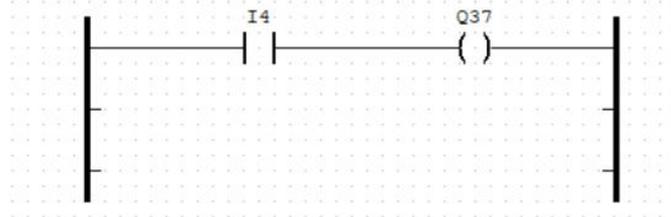


编程窗口出现新建的电源导轨。

# Installation Manual



同理，添加右电源导轨。再将左电源导轨、输入触点、输出线圈、右电源导轨首尾相连。



#### (4) 备注

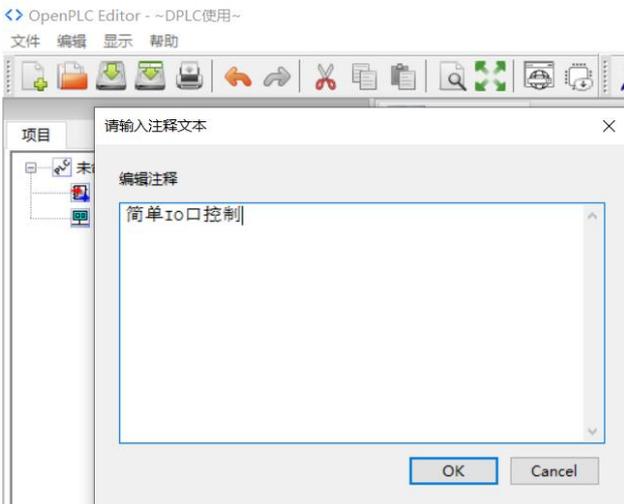
点击菜单工具栏的“新建一个备注”。



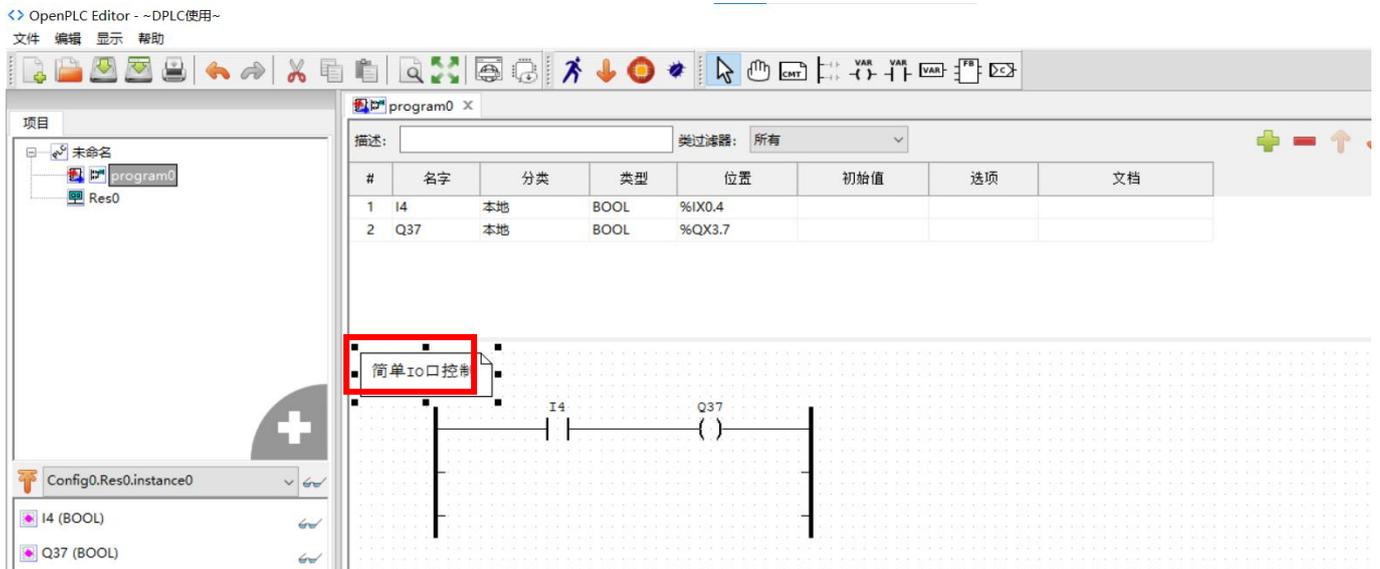
再点击编程窗口的任意空白处。



此时，注释文本窗口弹出，编辑完注释后，点击 OK。



编程窗口出现添加的备注，拖动备注到合适的位置。



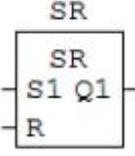
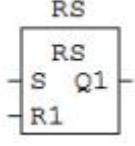
## 3.2.3 函数块介绍

库包含多种标准和用户自定义函数块，这些函数块支持在不同编程语言中使用。

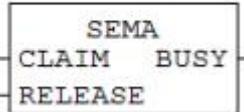
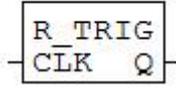
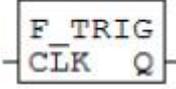
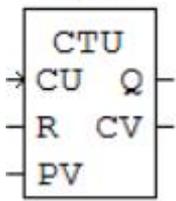
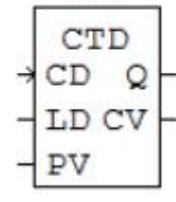
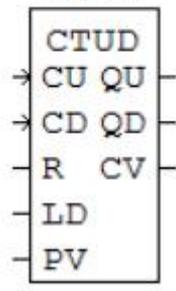


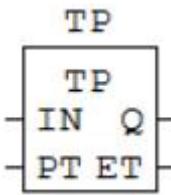
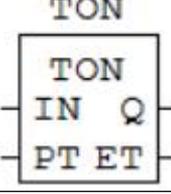
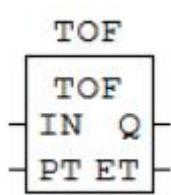
接下来将对标准功能函数块和常用的函数块详细说明。

### (1) 标准功能函数块

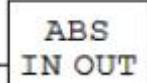
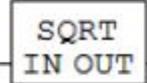
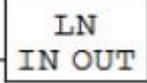
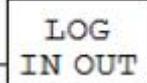
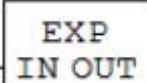
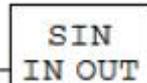
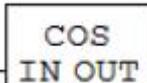
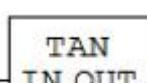
名称	图标	功能
SR		<p>SR 双稳态是一个置位有限锁存器。  <math>(BOOL:S1, BOOL:R) \Rightarrow (BOOL:Q1)</math>            此函数是标准的置位主导触发器。当输入 S1 为 TRUE, R 为 FALSE 时, Q1 输出变为 TRUE。同样, 当输入 S1 为 FALSE, R 为 TRUE 时, Q1 输出为 FALSE。在其中一个转换之后, 当 S1 和 R 信号都返回 FALSE 时, Q1 输出保持以前的状态, 直到出现新的条件。如果两个信号都为 TRUE 时, 则 Q1 输出强制为 TRUE (置位主导)。</p>
RS		<p>RS 双稳态是一个复位优先锁存器。  <math>(BOOL:S, BOOL:R1) \Rightarrow (BOOL:Q1)</math>            此函数是标准的复位主导触发器。当输入 S 为 TRUE, R1 输入为 FALSE 时, Q1 输出为 TRUE。同样, 当输入 S 为 FALSE, R1 输入为 TRUE 时, Q1 输出为 FALSE。在其中一个转换之后, 当 S 和 R1 信号都返回 FALSE 时, Q1 输出保持以前的状态, 直到出现新的条件。如果两个信号都为 TRUE 时, Q1 输出强制为 FALSE (复位主导)。</p>

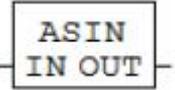
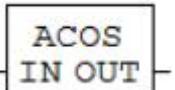
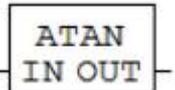
# Installation Manual

<p>SEMA</p>		<p>信号量提供一种机制，允许软件元素互斥排他的访问某一资源。 (BOOL:CLAIM, BOOL:RELEASE)=&gt;(BOOL:BUSY) 这个函数块实现了一个信号量函数。正常情况下，此功能用于同步事件。BUSY 输出由 CLAIM 输入上的 TRUE 条件激活，并由 RELEASE 输入上的 TRUE 条件停用。</p>
<p>R_TRIG</p>		<p>当检测到一个上升沿时，输出产生一个单独的脉冲。 (BOOL:CLK)=&gt;(BOOL:Q) 这个函数是一个上升边缘检测器。当在 CLK 输入上检测到 0 到 1 (或 FALSE 到 TRUE 或 OFF 到 ON) 条件时，Q 输出变为 TRUE，并在整个扫描周期中保持此状态。</p>
<p>F_TRIG</p>		<p>当检测到下降沿时，输出产生一个单独的脉冲。 (BOOL:CLK)=&gt;(BOOL:Q) 这个函数是一个下降沿检测器。当在 CLK 输入上检测到 1 到 0 (或 TRUE 到 FALSE 或 ON 到 OFF) 条件时，Q 输出变为 TRUE，并在整个扫描周期中保持此状态。</p>
<p>CTU CTU_DINT, CTU_LINT, CTU_UDINT, CTU_ULINT</p>		<p>当计数值达到最大值时，加计数器能被用来产生信号。 CTU: (BOOL:CU, BOOL:R, INT:PV)=&gt;(BOOL:Q, INT:CV) CTU_DINT: (BOOL:CU, BOOL:R, DINT:PV) =&gt; (BOOL:Q, DINT:CV) CTU_LINT: (BOOL:CU, BOOL:R, LINT:PV) =&gt; (BOOL:Q, LINT:CV) CTU_UDINT: (BOOL:CU, BOOL:R, UDINT:PV) =&gt; (BOOL:Q, UDINT:CV) CTU_ULINT: (BOOL:CU, BOOL:R, ULINT:PV) =&gt; (BOOL:Q, ULINT:CV) 该函数代表一个向上计数器。CU 输入的上升沿将使计数器增加 1。当 PV 的预设值达到时，Q 输出变为 TRUE。在 R 输入上应用 TRUE 信号将计数器复位为零 (异步复位)。CV 输出当前计数值。</p>
<p>CTD CTD_DINT, CTD_LINT, CTD_UDINT, CTD_ULINT</p>		<p>当计数值从一个预设值减到 0，减计数器能被用来产生信号。 CTD: (BOOL:CD, BOOL:LD, INT:PV)=&gt;(BOOL:Q, INT:CV) CTD_DINT: (BOOL:CD, BOOL:LD, DINT:PV) =&gt; (BOOL:Q, DINT:CV) CTD_LINT: (BOOL:CD, BOOL:LD, LINT:PV) =&gt; (BOOL:Q, LINT:CV) CTD_UDINT: (BOOL:CD, BOOL:LD, UDINT:PV) =&gt; (BOOL:Q, UDINT:CV) CTD_ULINT: (BOOL:CD, BOOL:LD, ULINT:PV) =&gt; (BOOL:Q, ULINT:CV) 该函数表示一个向下计数器。CD 输入的上升沿将使计数器减 1。当计数值等于或小于零时，Q 输出变为 TRUE。在 LD 输入上应用 TRUE 信号将用 PV 输入的值加在计数器。CV 输出当前计数值。</p>
<p>CTUD CTUD_DINT, CTUD_LINT, CTUD_UDINT, CTUD_ULINT</p>		<p>加减计数器有 2 个输入 CU 和 CD，能被用来同时在一个输入上加计数在另一个上减计数。 CTUD: ( BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, INT:PV ) =&gt; ( BOOL:QU, BOOL:QD, INT:CV ) CTUD_DINT: ( BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, DINT:PV ) =&gt; ( BOOL:QU, BOOL:QD, DINT:CV ) CTUD_LINT: ( BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, LINT:PV ) =&gt; ( BOOL:QU, BOOL:QD, LINT:CV ) CTUD_UDINT: ( BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, UDINT:PV ) =&gt; ( BOOL:QU, BOOL:QD, UDINT:CV ) CTUD_ULINT: ( BOOL:CU, BOOL:CD, BOOL:R, BOOL:LD, ULINT:PV ) =&gt; ( BOOL:QU, BOOL:QD, ULINT:CV ) 该函数表示一个可向上、向下的计数器。CU (向上计数) 输入上的上升沿使计数器加 1，而 CD (向下计数) 输入上的上升沿使计数器减 1。在 R 输入上应用 TRUE 信号会将计数器重置为零。LD 信号上的 TRUE 条件将向计数器加载应用于输入 PV 的值 (编程值)。当计数值大于或等于 PV 时，QU 输出 TRUE。当计数值小于或等于零时，QD 输出 TRUE。CV 输出当前计数的值。</p>

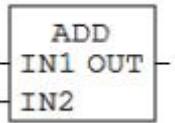
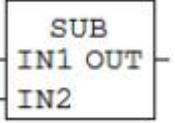
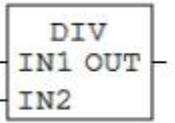
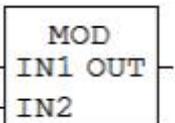
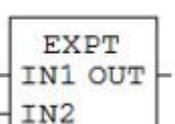
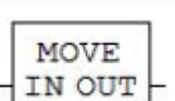
TP		<p>脉冲定时器能够被用来产生给定持续时间的输出脉冲 (BOOL:IN, TIME:PT)=&gt;(BOOL:Q, TIME:ET) 此定时器与单触发定时器或单稳定定时器具有相同的行为。 当在输入端检测到上升沿时, Q 输出为 TRUE。这种情况会持续到应用于相关引脚的编程时间 PT 结束为止。在 PT 结束后, 如果输入 IN 仍然有效, 则 Q 输出保持 ON 状态, 否则 Q 输出返回 OFF 状态。此计时器不可重新触发。这意味着定时器启动后, 在整个会话结束之前不能停止。ET 输出当前经过的时间。</p>
TON		<p>接通延迟定时器能被用来延迟设置一个输出为 TRUE, 当输入为 TRUE 时, 经过固定的时间, (BOOL:IN, TIME:PT)=&gt;(BOOL:Q, TIME:ET) 断言此函数的输入信号 IN 会启动计时器。当预设的输入 PT 的编程时间过去时, 输入 IN 仍然有效, Q 输出变为 TRUE。这种情况将持续到输入 IN 被释放为止。如果在时间过去之前释放 IN 输入, 则计时器将被清除。ET 输出当前经过的时间。</p>
TOF		<p>断开延迟定时器能被用来延迟设置一个输出为 FALSE, 当输入到 FALSE 后经过一个固定的时间。 (BOOL:IN, TIME:PT)=&gt;(BOOL:Q, TIME:ET) 这个函数的输入信号 IN 会立即激活 Q 输出。此时, 释放输入将开始时间流逝。当预设的输入 PT 的编程时间过去, 而输入 IN 仍然被释放时, Q 输出变为 FALSE。这个状态会一直保持, 直到输入被释放。如果在时间过去之前释放 IN 输入, 则定时器将被清除, Q 输出仍然为 TRUE。ET 输出当前经过的时间。</p>

## (2) 数学式函数块

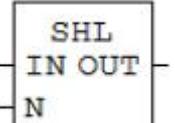
名称	图标	功能
ABS		<p>绝对值 (ANY_NUM:IN) =&gt; (ANY_NUM:OUT) ST syntax example: OUT := ABS(IN1);</p>
SQRT		<p>平方根 (底数 2) (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := SQRT(IN1);</p>
LN		<p>自然对数 (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := LN(IN1);</p>
LOG		<p>底数 10 的对数 (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := LOG(IN1);</p>
EXP		<p>幂 (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := EXP(IN1);</p>
SIN		<p>正弦 (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := SIN(IN1);</p>
COS		<p>余弦 (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := COS(IN1);</p>
TAN		<p>正切 (ANY_REAL:IN) =&gt; (ANY_REAL:OUT) ST syntax example: OUT := TAN(IN1);</p>

ASIN		反正弦 (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := ASIN(IN1);
ACOS		反余弦 (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := ACOS(IN1);
ATAN		反正切 (ANY_REAL:IN) => (ANY_REAL:OUT) ST syntax example: OUT := ATAN(IN1);

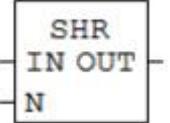
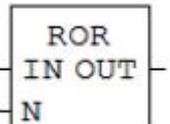
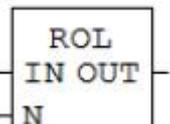
### (3) 运算函数块

名称	图标	功能
ADD		加法 (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 + IN2. 输入数量可以扩展。 ST syntax example: OUT := IN1 + IN2;
MUL		乘法 (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 * IN2. 输入数量可以扩展。 ST syntax example: OUT := IN1 * IN2;
SUB		减法 (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 - IN2. ST syntax example: OUT := IN1 - IN2;
DIV		除法 (ANY_NUM:IN1, ANY_NUM:IN2) => (ANY_NUM:OUT) OUT = IN1 / IN2. 示例: 1234 / 10 = 123. ST syntax example: OUT := IN1 / IN2;
MOD		余数 (模) (ANY_INT:IN1, ANY_INT:IN2) => (ANY_INT:OUT) OUT = IN1 modulo IN2. 示例: 1234 modulo 10 = 4. ST syntax example: OUT := IN1 MOD IN2;
EXPT		指数 (ANY_REAL:IN1, ANY_NUM:IN2) => (ANY_REAL:OUT) OUT = IN1 <sup>IN2</sup> 示例: 2 <sup>3</sup> = 8. ST syntax example: OUT := IN1 ** IN2;
MOVE		分配 (ANY:IN) => (ANY:OUT) OUT = IN. ST syntax example: OUT := IN1;

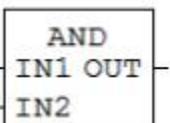
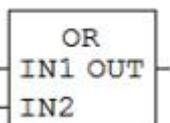
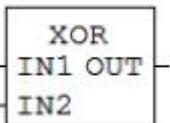
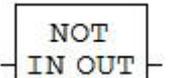
### (4) 位移函数块

名称	图标	功能
SHL		左移 (ANY_BIT:IN, ANY_INT:N) => (ANY_BIT:OUT) OUT 表示向左偏移 N 位的 IN 变量。在 OUT 变量的右侧填充零。 ST syntax example: OUT := SHL(IN := IN1, N := IN2);

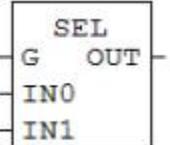
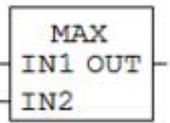
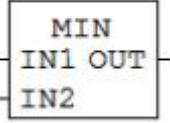
# Installation Manual

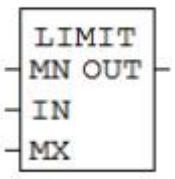
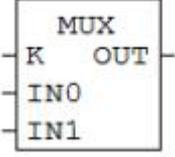
SHR		<p>右移  <math>(ANY\_BIT:IN, ANY\_INT:N) \Rightarrow (ANY\_BIT:OUT)</math>            OUT 表示向右偏移 N 位的 IN 变量。OUT 变量的左侧填充零。            ST syntax example: <code>OUT := SHR(IN := IN1, N := IN2);</code></p>
ROR		<p>循环右移  <math>(ANY\_NBIT:IN, ANY\_INT:N) \Rightarrow (ANY\_NBIT:OUT)</math>            OUT 表示向右移动 N 位的 IN 变量。每个移动的最右侧位都填充到 OUT 变量的最左侧位。            ST syntax example: <code>OUT := ROR(IN := IN1, N := IN2);</code></p>
ROL		<p>循环左移  <math>(ANY\_NBIT:IN, ANY\_INT:N) \Rightarrow (ANY\_NBIT:OUT)</math>            OUT 表示向左移动 N 位的 IN 变量。每次移动最左侧的位都填充到 OUT 变量的最右侧位。            ST syntax example: <code>OUT := ROL(IN := IN1, N := IN2);</code></p>

### (5) 位操作函数块

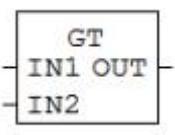
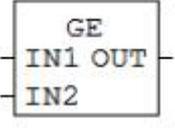
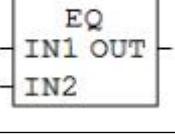
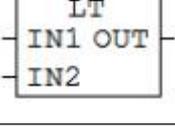
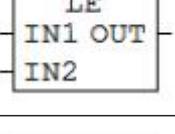
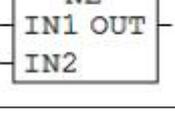
名称	图标	功能
AND		<p>按位“与”  <math>(ANY\_BIT:IN1, ANY\_BIT:IN2) \Rightarrow (ANY\_BIT:OUT)</math>  <math>OUT = IN1 \text{ AND } IN2.</math>            输入数量可以扩展。            ST syntax example: <code>OUT := IN1 AND IN2;</code></p>
OR		<p>按位“或”  <math>(ANY\_BIT:IN1, ANY\_BIT:IN2) \Rightarrow (ANY\_BIT:OUT)</math>  <math>OUT = IN1 \text{ OR } IN2.</math>            输入数量可以扩展。            ST syntax example: <code>OUT := IN1 OR IN2;</code></p>
XOR		<p>按位“异或”  <math>(ANY\_BIT:IN1, ANY\_BIT:IN2) \Rightarrow (ANY\_BIT:OUT)</math>  <math>OUT = IN1 \text{ EXCLUSIVE OR } IN2.</math>            输入数量可以扩展。            ST syntax example: <code>OUT := IN1 XOR IN2;</code></p>
NOT		<p>按位“反向”  <math>(ANY\_BIT:IN) \Rightarrow (ANY\_BIT:OUT)</math>  <math>OUT = NOT \text{ IN}.</math>            ST syntax example: <code>OUT := IN1 NOT IN2;</code></p>

### (6) 选择函数块

名称	图标	功能
SEL		<p>二进制选取 (二选一)  <math>(BOOL:G, ANY:IN0, ANY:IN1) \Rightarrow (ANY:OUT)</math>            如果 G 为 FALSE, 则 OUT 将与 IN0 值一致。如果 G 为 TRUE, OUT 将与 IN1 值一致。</p>
MAX		<p>最大值  <math>(ANY:IN1, ANY:IN2) \Rightarrow (ANY:OUT)</math>            该函数块比较输入 IN1 和 IN2 时的值的大小, 并输出其最大值。            输入数量可以扩展。</p>
MIN		<p>最小值  <math>(ANY:IN1, ANY:IN2) \Rightarrow (ANY:OUT)</math>            该函数块比较输入 IN1 和 IN2 时的值的大小, 并输出其最小值。            输入数量可以扩展。</p>

LIMIT		<p>限制 (ANY:MN, ANY:IN, ANY:MX) =&gt; (ANY:OUT) OUT 是最小值 MN 和最大值 MX 之间的值。如果 IN 小于最小值 MN, OUT 显示 MN 值; 如果 IN 大于最大 MX 值, OUT 显示 MX 值。</p>
MUX		<p>多路器 (多选一) (ANY_INT:K, ANY:IN0, ANY:IN1) =&gt; (ANY:OUT) 取决于 K 值, IN1, IN2..Inn 其中的被选中, OUT 表示选中的输入值。 输入数量可以扩展。</p>

(7) 比较函数块

名称	图标	功能
GT		<p>大于 (ANY:IN1, ANY:IN2) =&gt; (BOOL:OUT) 如果 IN1 大于 IN2, OUT 则为 TRUE, 否则为 OUT 为 FALSE。 输入数量可以扩展。 ST syntax example: OUT := IN1 &gt; IN2;</p>
GE		<p>大于或等于 (ANY:IN1, ANY:IN2) =&gt; (BOOL:OUT) 如果 IN1 大于或等于 IN2, OUT 则为 TRUE, 否则为 OUT 为 FALSE。 输入数量可以扩展。 ST syntax example: OUT := IN1 &gt;= IN2;</p>
EQ		<p>等于 (ANY:IN1, ANY:IN2) =&gt; (BOOL:OUT) 如果 IN1 等于 IN2, OUT 则为 TRUE, 否则为 OUT 为 FALSE。 输入数量可以扩展。 ST syntax example: OUT := IN1 = IN2;</p>
LT		<p>小于 (ANY:IN1, ANY:IN2) =&gt; (BOOL:OUT) 如果 IN1 小于 IN2, OUT 则为 TRUE, 否则为 OUT 为 FALSE。 输入数量可以扩展。 ST syntax example: OUT := IN1 &lt; IN2;</p>
LE		<p>小于或等于 (ANY:IN1, ANY:IN2) =&gt; (BOOL:OUT) 如果 IN1 小于或等于 IN2, OUT 则为 TRUE, 否则为 OUT 为 FALSE。 输入数量可以扩展。 ST syntax example: OUT := IN1 &lt;= IN2;</p>
NE		<p>不等于 (ANY:IN1, ANY:IN2) =&gt; (BOOL:OUT) 如果 IN1 不等于 IN2, OUT 则为 TRUE, 否则为 OUT 为 FALSE。 输入数量可以扩展。 ST syntax example: OUT := IN1 &lt;&gt; IN2;</p>

### 3.3 变量结构定义

		项 目	范 围	
继电	I	外部输入继电器	I0.0~I15.7, 八进制编码, 128 点	合计 256(扩充机+主机点数)
	Q	外部输出继电器	Q0.0~Q15.7, 八进制编码, 128 点	

器 位 元 型 态	MX	辅 助 继 电 器	一般用	MX0~MX1023, 1024 点	合 计 6144 点
			停电保持用	MX4096~MX6143, 2048 点	
			特殊用	MX1024~MX4095, 3072 点, 部分为停电保持	

项 目			范 围			
MB/MW /MD	资料 暂 存 器	外部输入 ADC	IW0~IW1, IW2~29, 十进制编码, 4 点, 合计 30 点			
		一般用	16 位	MW0~MW1023,1024 点		合 计 12288 点
			32 位	MD0~MD1023,1024 点		
		停电保 持用	16 位	MW4096~MW6143, 2048 点		
			32 位	MD4096~MD6143, 2048 点		
		特殊用	8 位	MB3072~MB4095, 1024 点, 部分是停电保持		
			16 位	MW1024~MW4095, 3072 点, 部分是停电保持		
			32 位	MD1024~MD4095, 3072 点, 部分是停电保持		
常数	K	十进制	K-32,768 ~ K32,767 (16 位元运算) K-2,147,483,648 ~ K2,147,483,647 (32 位元运算)			

## 3.4 特殊寄存器

### 3.4.1 脉冲指令

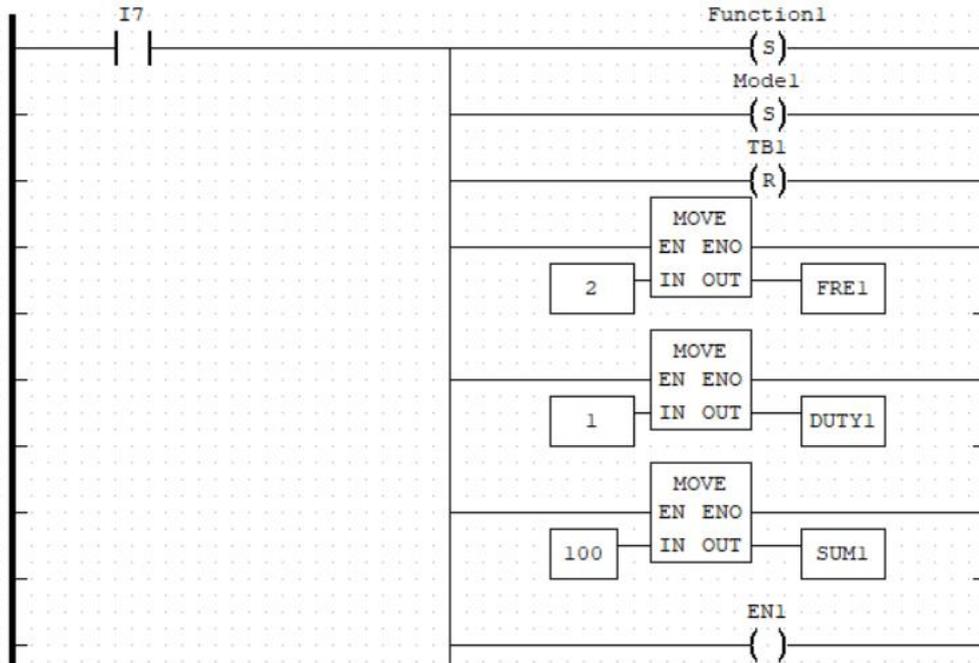
#### (1)特殊寄存器

寄存器地址	功能说明	数值定义
MX3072/MX3088/MX3104/MX3120	读项目, Q0.0-Q0.3 状态旗标	1: 脉冲输出完成, 0: 脉冲输出未完成
MX3074/MX3090/MX3106/MX3122	写项目, Q0.0-Q0.3 功能旗标	1: PWM/PTO 功能, 0: GPIO
MX3075/MX3091/MX3107/MX3123	写项目, Q0.0-Q0.3 模式旗标	1: PWM 模式, 0: PTO 模式
MX3076/MX3092/MX3108/MX3124	写项目, Q0.0-Q0.3 时基选择旗标	1: 1ms/格, 0: 1us/格
MW3072/MW3088/MW3104/MW3120	写项目, Q0.0-Q0.3 输出脉冲频率设定	2~65535 (0.015Hz~500kHz)
MW3073/MW3089/MW3105/MW3121	写项目, Q0.0-Q0.3 输出脉冲宽度设定	1~32767
MW3074/MW3090/MW3106/MW3122	写项目, Q0.0-Q0.3 输出脉冲数目设定	1~65535

#### (2)指令写法

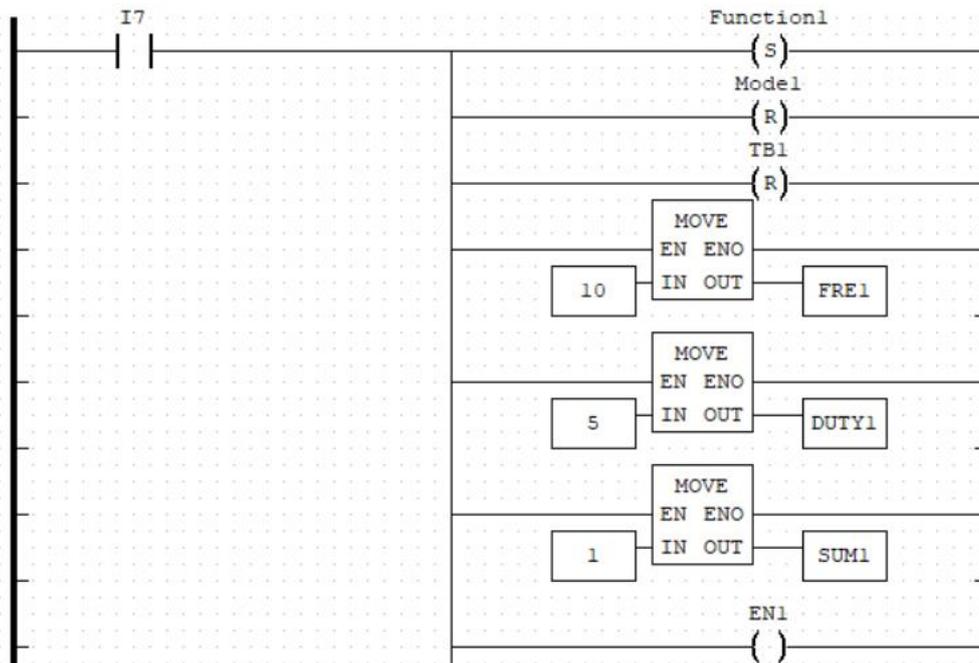
- PWM 模式: 示例为 Q0.0 输出 500kHz, Duty: 50%的脉冲。

# Installation Manual



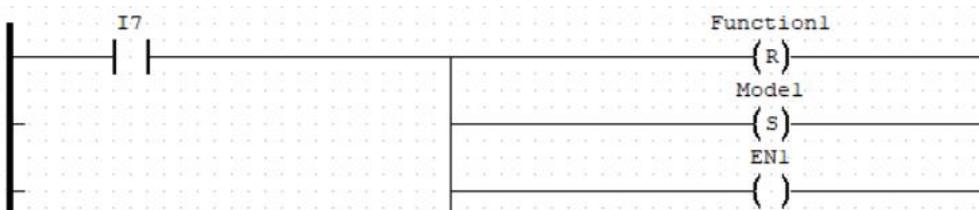
MX3074 设为 1. PWM/PTO  
 MX3075 设为 1. PWM 槽  
 MX3076 设为 0. 时基是 1us/  
 MW3072 设为 2, 频率  
 500k, 即为 2us  
 MW3073 设为 1, 占空比  
 50%, 即为 1us  
 PWM 模式下, 脉冲数目  
 (MW3074) 可以任意设  
 该赋值线圈为 Q0.0, 上  
 述参数设置完毕后

- PTO 模式: 示例为 Q0.0 输出 100kHz, Duty: 50%, 1 个脉冲。



MX3074 设为 1. PWM/PTO  
 MX3075 设为 0. PTO 槽  
 MX3076 设为 0. 时基是 1us/  
 MW3072 设为 10, 频率  
 100k, 即为 10us  
 MW3073 设为 5, 占空比  
 50%, 即为 5us  
 MW3074 设为 1, 脉冲数  
 该赋值线圈为 Q0.0, 上  
 述参数设置完毕后

- 普通高低电平输出模式: 示例为 Q0.0 输出高低电平。



MX3074 设为 0. GPTO  
 MX3075 默认设为 1. PWM  
 赋值线圈 Q0.0 输出高



# Installation Manual

## 3.4.2 高速计数指令

### (1)特殊寄存器

寄存器地址	功能说明	数值定义
MX2048/MX2064/MX2080/MX2096	写项目, I0.0-I0.3 使能侦测旗标	1: 使能, 0: 未使能
MX2049/MX2065/MX2081/MX2097	写项目, I0.0-I0.3 计数方式旗标	1: 上升沿计数, 0:下降沿计数
MX2050/MX2066/MX2082/MX2098	写项目, I0.0-I0.3 计数模式旗标	1: 向上计数, 0:向下计数
MX2051/MX2067/MX2083/MX2099	读项目, I0.0-I0.3 状态旗标	1: 正在使用该输入, 0: 未使用该输入
MX2052/MX2068/MX2084/MX2100	写项目, I0.0-I0.3 输入脉波频率侦测旗标	1: 启动, 0:关闭
MD2048/MD2064/MD2080/MD2096	读项目, I0.0-I0.3 输入脉冲频率	/
MD2049/MD2065/MD2081/MD2097	写项目, I0.0-I0.3 输入脉冲预设值, 高于该值, 不再计数	0~4294967295
MD2050/MD2066/MD2082/MD2098	读项目, I0.0-I0.3 输入脉冲数目	/

### (2)指令写法

- HSC 高数计数模式: 示例为 I0.0 处于上升沿、向上计数模式。

在变量表中添加暂存器地址, 赋予初始值即可使用该功能。

#	名字	分类	类型	位置	初始值
10	CEN1	本地	BOOL	%MX2048	1
11	CEGE1	本地	BOOL	%MX2049	1
12	CMODE1	本地	BOOL	%MX2050	1
13	CSTATUS1	本地	BOOL	%MX2051	
14	CDC1	本地	BOOL	%MX2052	1
15	CFRE1	本地	UDINT	%MD2048	
16	CPRO1	本地	UDINT	%MD2049	
17	CSUM1	本地	UDINT	%MD2050	

MX2051 (是否为计数状态)、MD2048 (读取脉冲频率)、MD2050 (读取脉冲数目) 通过

- 普通高低电平输入模式: 示例为 I0.0 处于高低电平输出模式。

#	名字	分类	类型	位置	初始值
9	CEN1	本地	BOOL	%MX2048	1
10	CEGE1	本地	BOOL	%MX2049	1
11	CMODE1	本地	BOOL	%MX2050	1
12	CSTATUS1	本地	BOOL	%MX2051	
13	CDC1	本地	BOOL	%MX2052	0

## 3.5 特殊暂存器功能

特 MB	功能说明	Off ↓ On	STOP ↓ RUN	RUN ↓ STOP	属性	停电保持	出厂值
MB3072	Q0.0 轴正转限制位	-	-	-	R/W	否	-
MB3073	Q0.0 轴原点限制位	-	-	-	R/W	否	-
MB3074	Q0.0 轴反转限制位	-	-	-	R/W	否	-
MB3075	Q0.0 轴输出控制位	-	-	-	R/W	否	-
MB3076 ↓ MB3087	保留给 Q0.0 轴	-	-	-	R/W	-	-
MB3088 ↓ MB3103	Q0.1 轴限制位, 如 Q0.0	-	-	-	R/W	否	-
MB3104 ↓ MB3119	Q0.2 轴限制位, 如 Q0.0	-	-	-	R/W	否	-
MB3120 ↓ MB3135	Q0.3 轴限制位, 如 Q0.0	-	-	-	R/W	否	-
MB3136 ↓ MB3583	扩充机轴限制位, 同上。	-	-	-	R/W	否	-
MB3584 ↓ MB4095	预留	-	-	-	-	-	-
特 MX	功能说明	Off ↓ On	STOP ↓ RUN	RUN ↓ STOP	属性	停电保持	出厂值
MX1024	运转监视常开接点 (A 接点)	Off	On	Off	R	否	Off
MX1025	运转监视常闭接点 (B 接点)	On	Off	On	R	否	On
MX1026	启始正向 (RUN 的瞬间'On') 脉波	Off	On	Off	R	否	Off
MX1032	有不正确的通讯服务要求	Off	-	-	R	否	Off
MX2048	I1.0 计数使能旗标 (ON: 使能, OFF: 不使能)	Off	Off	-	R/W	否	Off
MX2049	I1.0 计数方式旗标 (ON: 上升沿中断, OFF: 下降沿中断)	Off	Off	-	R/W	否	Off
MX2050	I1.0 计数模式旗标 (ON: 向上, OFF: 向下)	Off	Off	-	R/W	否	Off
MX2051	I1.0 清除状态旗标 (ON: 启动, OFF: 关闭)	Off	Off	-	R/W	否	Off
MX2052	I1.0 输入脉波频率侦测旗标 (ON: 启动, OFF: 关闭)	Off	Off	-	R/W	否	Off
MX2053 ↓ MX2063	保留给 I1.0						
MX2064	I1.1 旗标, 如上 I0.0	Off	Off	-	R/W	否	Off



# Installation Manual

↓ MX2079								
MX2080 ↓ MX2095	I1.2 旗标, 如上 I0.0	Off	Off	-	R/W	否	Off	
MX2096 ↓ MX2111	I1.3 旗标, 如上 I0.0	Off	Off	-	R/W	否	Off	
MX2112 ↓ MX2559	扩充机高速计数旗标	Off	Off	-	R/W	否	Off	
MX2560 ↓ MX3071	保留给 I							
MX3072	Q0.0 状态旗标 (ON: 启动, OFF: 关闭)	Off	Off	-	R/W	否	Off	
MX3073	Q0.0 轴运动方向控制位(ON: 轴正计数, OFF: 轴负计数)	Off	Off	-	R/W	否	Off	
MX3074	Q0.0 功能旗标 (ON: PWM/PTO 功能, OFF: GPIO)	Off	Off	-	R/W	否	Off	
MX3075	Q0.0 模式旗标 (ON: PWM 模式, OFF: PTO 模式)	Off	Off	-	R/W	否	Off	
MX3076	Q0.0 基准选择旗标 (ON: 1ms/格, OFF: 1us/格)	Off	Off	-	R/W	否	Off	
MX3077	Q0.0 恢复原点使能位 (ON: 轴开始恢复原点 OFF:轴停止恢复原点)	Off	Off	-	R/W	否	Off	
MX3078	恢复原点方式选择: (ON:恢复原点为 PTO 模式。 OFF:恢复原点为限位传感器模式)	Off	Off	-	R/W	否	Off	
MX3079	Q0.0 轴方向取反位 (ON: QX.X 正计数时对应的方向输出置 0, 反之负计数输出置 1 OFF: QX.X 正计数时对应的方向输出置 1 反之负计数输出置 0)	Off	Off	-	R/W	否	Off	
MX3080	Q0.0 轴限位方向取反位 (ON: QX.X 限位传感器输入为 1 时, 对应引脚旗标为 0 OFF: QX.X 限位传感器输入为 1 时, 对应引脚旗标为 1)	Off	Off	-	R/W	否	Off	
MX3081	Q0.0 轴恢复原点方向位 (ON: 恢复原点方向往轴正计数方向 OFF: 恢复原点方向往轴负计数方向)	Off	Off	-	R/W	否	Off	
MX3082	Q0.0 轴恢复原点前后位 (ON: 恢复原点为正计数离开原点位置 OFF: 恢复原点为反计数离开原点位置)	Off	Off	-	R/W	否	Off	
MX3083	Q0.0 轴恢复原点时基(ON: 1ms, OFF: 1us)	Off	Off	-	R/W	否	Off	



## Installation Manual

MX3084	Q0.0 轴爬行速度时基(ON: 1ms, OFF: 1us)	Off	Off	-	R/W	否	Off
MX3085 ↓ MX3087	保留	-	-	-	-	-	-
MX3088 ↓ MX3103	Q0.1 旗标 , 如 Q0.0	Off	Off	-	R/W	否	Off
MX3104 ↓ MX3119	Q0.2 旗标 , 如 Q0.0	Off	Off	-	R/W	否	Off
MX3120 ↓ MX3135	Q0.3 旗标 , 如 Q0.0	Off	Off	-	R/W	否	Off
MX3136 ↓ MX3583	扩充机 Q2.0-Q14.3 旗标 , 如 Q0.0	Off	Off	-	R/W	否	Off
MX3584 ↓ MX4095*	保留给 Q	Off	Off	-	R/W	否	Off

特 MW	功能说明	Off ↓ On	STOP ↓ RUN	RUN ↓ STOP	属性	停电保持	出厂值
MW1025	DPLC 机种系统程序版本, (用户可从此暂存器中读出 PLC 的韧体版本。例如, MW1025=0, 即韧体版本 0.0)	-	-	-	R	否	#
MW1028	Modbus 主机地址, 1-247	-	-	-	R	是	#
MW1029	现在扫描周期 (单位: 1ms)	-	-	-	R	否	0
MW1030	最小扫描周期 (单位: 1ms)	-	-	-	R/W	否	0
MW1031	最大扫描周期 (单位: 1ms)	-	-	-	R/W	否	0
MW1033	万年历(RTC) 秒 00~59	-	-	-	R/W	是	0
MW1034	万年历(RTC) 分 00~59	-	-	-	R/W	是	0
MW1035	万年历(RTC) 时 00~23	-	-	-	R/W	是	0
MW1036	万年历(RTC) 日 01~31	-	-	-	R/W	是	0



## Installation Manual

MW1037	万年历(RTC) 月 01~12	-	-	-	R/W	是	0
MW1038	万年历(RTC) 星期 1~7	-	-	-	R/W	是	0
MW1039	万年历(RTC) 年 00~ 99	-	-	-	R/W	是	0
MD2048	I0.0 输入频率	0	-	-	R/W	否	0
MD2049	I0.0 输入脉冲预设值	0	-	-	R/W	否	0
MD2050	I0.0 输入脉冲计数值	0	-	-	R/W	否	0
MD2051 ↓ MD2063	保留给 I0.0						
MD2058 ↓ MD2060	I0.1 周期, 脉冲宽度, 脉冲计数, 如 I0.0	0	-	-	R/W	否	0
MD2068 ↓ MD2070	I0.2 周期, 脉冲宽度, 脉冲计数, 如 I0.0	0	-	-	R/W	否	0
MD2078 ↓ MD2080	I0.3 周期, 脉冲宽度, 脉冲计数, 如 I0.0	0	-	-	R/W	否	0
MD2081 ↓ MD2559	扩充机高速计数器值	0	-	-	R/W	否	0
MD2560 ↓ MD3071	保留给 I						
MD3072	Q0.0 脉冲暂存器,用于记录运动轴实际位置	0	-	-	R/W	否	0
MD3073	Q0.0 轴恢复原点变速时间	0	-	-	R/W	否	0
MD3074 ↓ MD3087	保留给 Q0.0	0	-	-	R/W	否	0
MD3088 ↓ MD3103	Q0.1 脉冲暂存器,用于记录运动轴实际位置	0	-	-	R/W	否	0
MD3104 ↓ MD3119	Q0.2 脉冲暂存器,用于记录运动轴实际位置	0	-	-	R/W	否	0



## Installation Manual

MD3120 ↓ MD3115	Q0.3 脉冲暂存器,用于记录运动轴实际位置	0	-	-	R/W	否	0
MD3116 ↓ MD3583	扩充机 Q2.0-Q14.3 旗标 , 如 Q0.0	Off	Off	-	R/W	否	Off
MD3584 ↓ MD4095*	保留给 Q						
MW3072	Q0.0 输出周期设定	0	-	-	R/W	否	0
MW3073	Q0.0 输出脉冲宽度设定	0	-	-	R/W	否	0
MW3074	Q0.0 输出脉冲个数设定	0	-	-	R/W	否	0
MW3075	Q0.0 轴恢复原点速度	0	-	-	R/W	否	0
MW3076	Q0.0 轴爬行速度	0	-	-	R/W	否	0
MW3077	Q0.0 轴恢复原点速度占空比	0	-	-	R/W	否	0
MW3078	Q0.0 轴爬行速度占空比	0	-	-	R/W	否	0
MW3079 ↓ MW3087	保留给 Q0.0						
MW3088 ↓ MW3103	Q0.1 输出周期, 宽度, 计数, 如 Q0.0	0	-	-	R/W	否	0
MW3104 ↓ MW3119	Q0.2 输出周期, 宽度, 计数, 如 Q0.0	0	-	-	R/W	否	0
MW3120 ↓ MW3135	Q0.3 输出周期, 宽度, 计数, 如 Q0.0	0	-	-	R/W	否	0
MW3136 ↓ MW3583	扩充机 Q2.0-Q14.3 输出周期, 宽度, 计数, 如 Q0.0	0	-	-	R/W	否	0
MW3584 ↓ MW4095	保留给 Q	0	-	-	R/W	否	0

## 四、通讯功能

### 4.1 Modbus 通讯功能

DPLC 采用 Modbus RTU 主从传输模式，Baud rate 为 115.2Kbps、1 Start bit、8 Data bits、No parity、1 Stop bit。

因 Modbus 协定定义限制封包最大传输量，在 RTU 模式下时最大限制为 128bytes，因此传送封包之间彼此间距不得小于 20mSec。在传送资料时，除 Error Check (CRC16)资料外，所有的 word 资料必须符合 High byte 先传送之原则。

#### 主站形式：

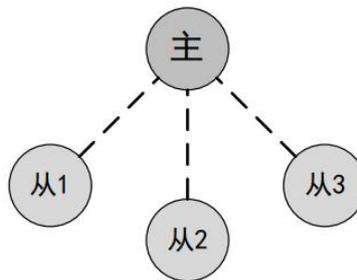
PLC 作为主站设备时，通过 Modbus 指令可与其它使用 Modbus-RTU 或者 Modbus-ASCII 协议的从机设备通讯；与其他设备进行数据交换。

#### 从站形式：

PLC 作为从站设备时，只能对其它主站的要求作出响应。

#### 主从的概念：

在 RS485 网络中，某一时刻，可以有一主多从（如下图），其中主站可以对其中任意从站进行读写操作，从站之间不可直接进行数据交换，主站需编写通讯程序，对其中的某个从站进行读写，从站无需编写通讯程序，只需对主站的读写进行响应即可。（接线方式：所有的 485+连在一起，所有的 485-连在一起）



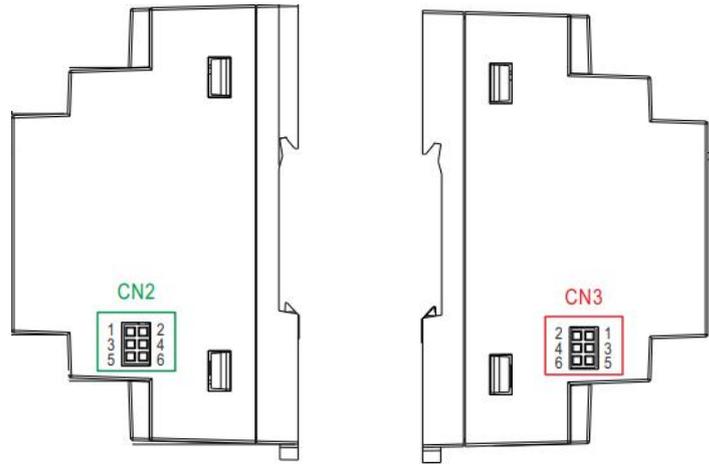
之所以图中有虚线箭头，是因为理论上在两个网络中，只要各个 PLC 不发数据，网络中任意 PLC 都可以用来作为主站，其它 PLC 作为从站；但是由于多个 PLC 之间没有一个统一的时钟基准，容易出现在同一时刻有多个 PLC 发送数据，会导致通讯冲突失败，因此不建议这样使用。

### 4.2 主从机使用介绍

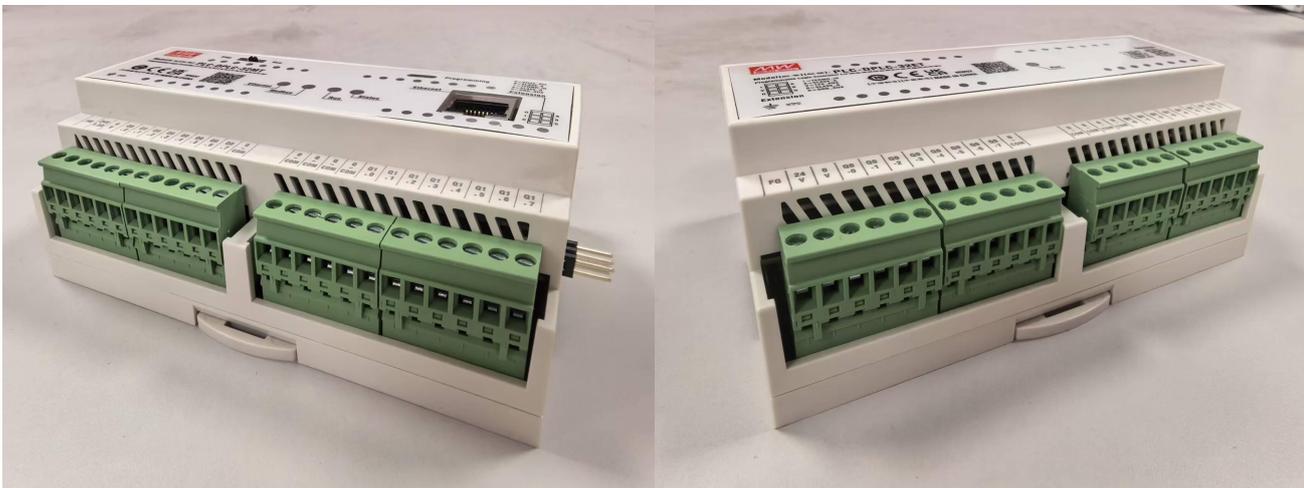
#### 4.2.1 主从机接线

主机的 CN2 母座和从机的 CN3 母座，使用用排针或是排线连接。若要连接多台从机，以此类推。

# Installation Manual



以下左图为主机 CN2 插上排针，再插上右图从机的 CN3。



下图为主从机用排针连接的实物图。



## 4.2.2 程序编写

下图为主机 I 口控制从机 Q 口的示例：



在数据变量表中添加新变量，再映射到 DPLC 对应的位置。

## 4.2.3 异常报警

主机：

灯号	绿色	橙色	红色
<b>Status 灯</b>	常亮 系统异常 闪烁 存储动作	/	/
<b>Run 灯</b>	常亮 正常工作 闪烁 stop 状态	常亮 扩充机断线 闪烁 烧录程式中	常亮 程式执行异常
<b>MODBus 灯</b>	常亮 通讯中	/	常亮 通讯异常
<b>Ethernet 灯</b>	常亮 通讯中	/	常亮 通讯异常

从机：

灯号	绿色	橙色	红色
<b>Run 灯</b>	常亮 定址判断完成	常亮 判断定址中	常亮 通讯异常

## 4.3 通讯标志位与寄存器

### 4.3.1 通讯标志位

通讯标志位	功能说明	数值说明
特殊继电器 MX1032	读项目，监控通讯功能是否正确	0: 没有不正确通讯; 1: 有不正确通讯
特殊暂存器 MW1028	写项目，Modbus 主机地址	1~247



# Installation Manual

## 4.3.2DPLC 变量对应 Modbus 地址

暂存器位置范围	Byte 数目	PLC 软件元件名称	叙述	功能码
0x0000~0x007F	16	Q_STATUS	输出(Q0~Q127) 运行状态	0x01、0x05、0X0F
0x0080~0x03FF	-	Reserved	-	-
0x0400~0x07FF	128	MX0~MX1023	一般用继电器	0x01、0x05、0X0F
0x0800~0x0BFF	-	Reserved	-	-
0x0C00~0x0FFF	128	MX1024-MX2047	特殊继电器	0x01、0x05、0X0F
0x1000~0x102F	6	MX2048-MX2095	特殊继电器(I0-I3)	0x01、0x05、0X0F
0x1030~0x13FF	-	Reserved	-	-
0x1400~0x153F	40	MX3072-MX3391	特殊继电器(Q0-Q31)	0x01、0x05、0X0F
0x1540~0x17FF	-	Reserved	-	-
0x1800~0x1FFF	256	MX4096-MX6143	停电保持继电器	0x01、0x05、0X0F
0x2000~0x23FF	-	Reserved	-	-
0x2400~0x33FF	-	Reserved	-	-
0x3400~0x347F	16	I_STATUS	输入(I0~I127) 运行状态	0x02
0x3480~0x37FF	-	Reserved	-	-
0x3800~0x38FF	512	MB3072-MB3583	8 位特殊资料暂存器	0x03、0x06、0X10
0x3900~0x39FF	-	Reserved	-	-
0x3A00~0x3DFF	2048	MW0-MW1023	16 位一般资料暂存器	0x03、0x06、0X10
0x3E00~0x41FF	-	Reserved	-	-
0x4200~0x45FF	2048	MW1024-MW2047	16 位特殊资料暂存器	0x03、0x06、0X10
0x4600~0x47FF	1024	MW2048-MW2559	16 位特殊资料暂存器	0x03、0x06、0X10
0x4800~0x49FF	-	Reserved	-	-
0x4A00~0x4BFF	1024	MW3072-MW3583	16 位一般资料暂存器 (Q0-Q3)	0x03、0x06、0X10
0x4C00~0x4DFF	-	Reserved	-	-
0x4E00~0x55FF	4096	MW4096-MW6143	16 位停电保持资料暂存器	0x03、0x06、0X10
0x5600~0x59FF	-	Reserved	-	-
0x5A00~0x61FF	4096	MD0-MD1023	32 位一般资料暂存器	0x03、0x06、0X10



## Installation Manual

0x6200~ 0x69FF	-	Reserved	-	-
0x6A00~ 0x71FF	4096	MD1024-MD2047	32 位特殊资料暂存器	0x03、0x06、0X10
0x7200~ 0x75FF	2052	MD2048-MD2560	32 位特殊资料暂存器 (I0-I3)	0x03、0x06、0X10
0x7600~ 0x79FF	-	Reserved	-	-
0x7800~ 0x7DFF	2048	MD3072-MD3583	32 位脉冲暂存器 (Q0-Q3)	0x03、0x06、0X10
0x7E00~ 0x81FF	-	Reserved	-	-
0x8200~ 0x91FF	8192	MD4096-MD6143	32 位停电保持资料暂存器	0x03、0x06、0X10
0x9200~ 0xA1FF	-	Reserved	-	-
0xA200~ 0xB1FF	-	Reserved	-	-
0xB200~ 0xB21D	60	IW0-IW29	ADC 类比输入值	0x04
0xB21E~ 0xB5FF	-	Reserved	-	-
0xB600~ 0xFFFF	-	Reserved	-	-

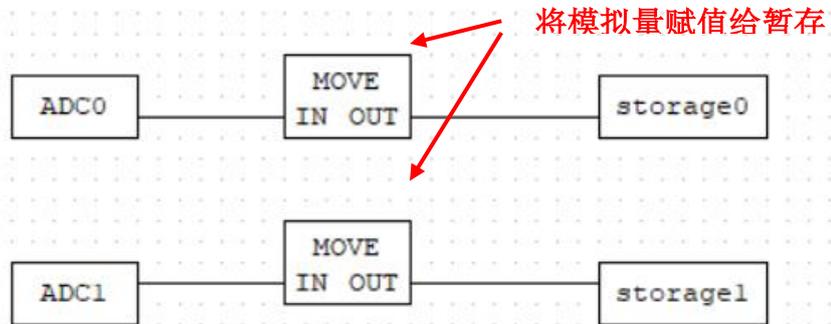
## 五、范例

### 5.1 IO 功能使用

#### 5.1.1 类比输入使用

#	名字	分类	类型	位置	初始值	选项
1	ADC0	本地	UINT	%IW0		IW0 为主机类比输入
2	ADC1	本地	UINT	%IW1		IW1 为主机类比输入
3	storage0	本地	UINT	%MW0		设置两个暂存器存入
4	storage1	本地	UINT	%MW1		

可以使用 Debug 功能查看 ADC0、ADC1 的值



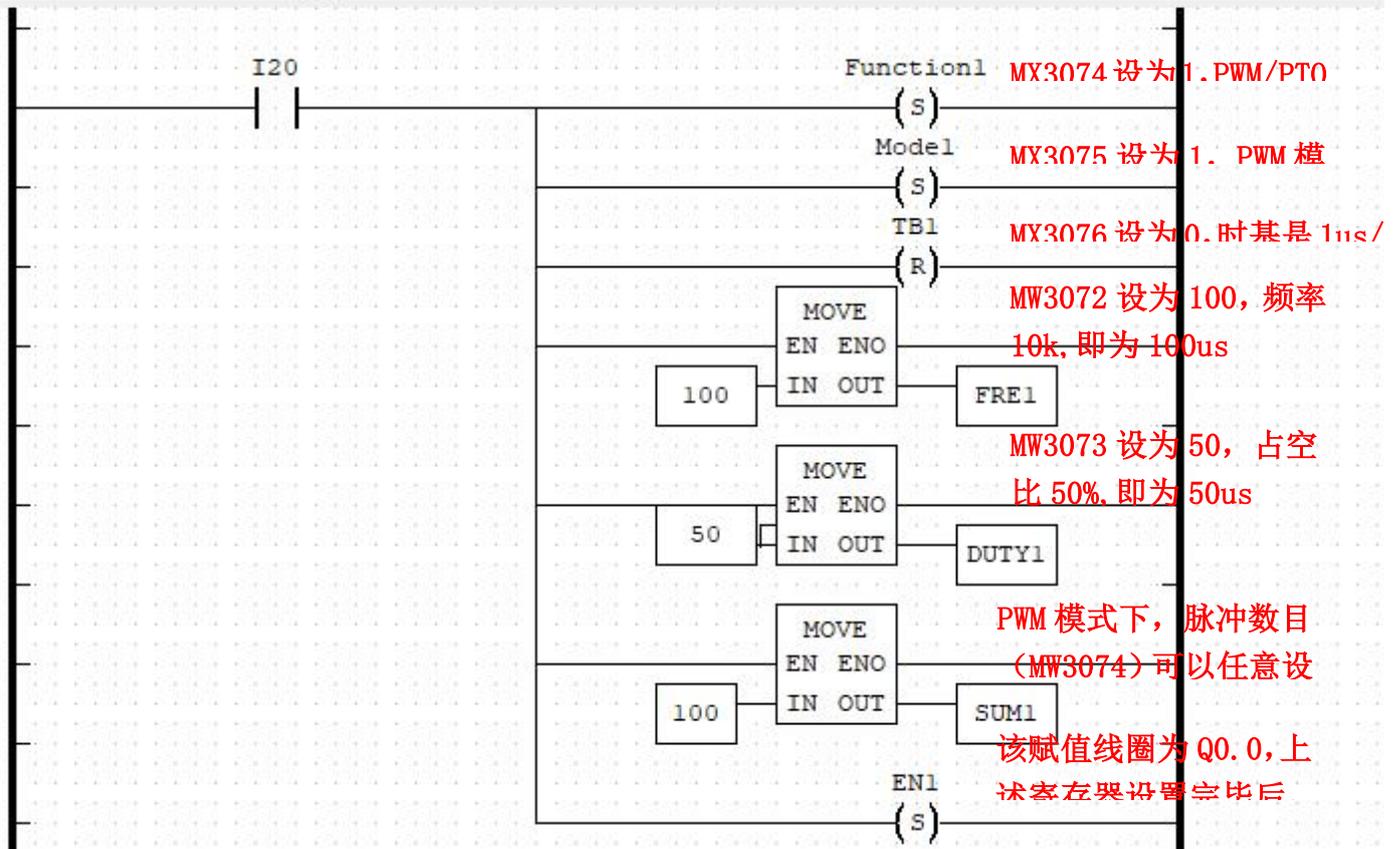
#### 5.1.2 HSC 输入使用

#	名字	分类	类型	位置	初始值
10	CEN1	本地	BOOL	%MX2048	1
11	CEGE1	本地	BOOL	%MX2049	1
12	CMODE1	本地	BOOL	%MX2050	1
13	CSTATUS1	本地	BOOL	%MX2051	
14	CDC1	本地	BOOL	%MX2052	1
15	CFRE1	本地	UDINT	%MD2048	
16	CPRO1	本地	UDINT	%MD2049	
17	CSUM1	本地	UDINT	%MD2050	

MX2051 (是否为计数状态)、MD2048 (读取脉冲频率)、MD2050 (读取脉冲数目) 通过

## 5.1.3 PWM 输出使用

#	名字	分类	类型	位置	初始值	选项
56	EN1	本地	BOOL	%QX0.0		
57	Status1	本地	BOOL	%MX3073		
58	Function1	本地	BOOL	%MX3074		
59	Mode1	本地	BOOL	%MX3075		
60	TB1	本地	BOOL	%MX3076		
61	FRE1	本地	UINT	%MW3072		
62	DUTY1	本地	UINT	%MW3073		
63	SUM1	本地	UINT	%MW3074		



## 5.1.4 数位输入输出使用

#	名字	分类	类型	位置	初始值	选项
5	I4	本地	BOOL	%IX0.4		数位输入 I0.4
6	Q4	本地	BOOL	%QX0.4		数位输出 Q0.4

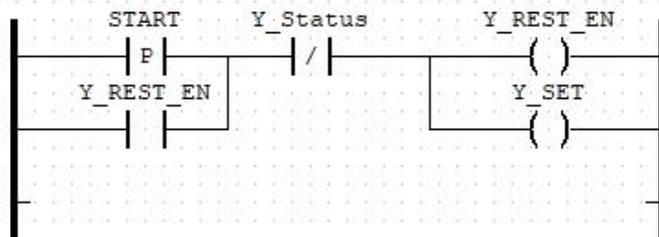


**I0.4 置高则 Q0.4 置高**  
**I0.4 置低则 Q0.4 置低**

## 5.2 恢复原点功能

示例:

#	名字	分类	类型	位置	初始值	选项
1	START	本地	BOOL	%IX0.0		
2	Y_SET	本地	BOOL	%QX0.0		
3	Y_P	本地	BYTE	%MB3072		<b>MR3072: 00.0 轴正转限制位</b>
4	Y_Z	本地	BYTE	%MB3073	8	<b>MR3073: 00.0 轴原占限制位</b>
5	Y_N	本地	BYTE	%MB3074		<b>MR3074: 00.0 轴反转限制位</b>
6	Y_OUT	本地	BYTE	%MB3075	4	<b>MR3075: 00.0 轴输出控制位</b>
7	Y_Status	本地	BOOL	%MX3072		<b>MX3072: 00.0 状态标志(1: 启动)</b>
8	Y_DIR	本地	BOOL	%MX3073		<b>MX3073: 运动方向控制位(1: 正转)</b>
9	Y_IO	本地	BOOL	%MX3074	1	<b>MX3074: 1: PWM/PTO 0: GPIO</b>
10	Y_PWM	本地	BOOL	%MX3075	1	<b>MX3075: 1: PWM 0: PTO</b>
11	Y_BASE	本地	BOOL	%MX3076	0	<b>MX3076: 基准选择标志(1: 1ms/格)</b>
12	Y_REST_EN	本地	BOOL	%MX3077		<b>MX3077: 00.0 恢复原占使能位</b>
13	Y_REST_FUN	本地	BOOL	%MX3078	0	<b>MX3078: 00.0 轴方向取后位</b>
14	Y_OUT_REV	本地	BOOL	%MX3079	0	<b>MX3079: 00.0 轴恢复原占方向位</b>
15	Y_IN_REV	本地	BOOL	%MX3080	0	<b>MX3080: 00.0 恢复原占前后位</b>
16	Y_REST_DIR	本地	BOOL	%MX3081	0	<b>MX3081: 恢复原点时基(ON: 1ms,</b>
17	Y_REST_ZDIR	本地	BOOL	%MX3082	1	<b>MX3082: 爬行速度时基(ON: 1ms,</b>
18	Y_SYS	本地	UINT	%MW3072	100	<b>MW3072 设为 100, 频率 10kHz, 即</b>
19	Y_DUTY	本地	UINT	%MW3073	50	<b>MW3073 设为 50, 占空比 50%.</b>
20	Y_FSYS	本地	UINT	%MW3075	100	<b>MW3075: 00.0 轴恢复原点速度</b>
21	Y_FDUTY	本地	UINT	%MW3076	50	<b>MW3076: 00.0 轴爬行速度</b>
22	Y_CSYS	本地	UINT	%MW3077	10	<b>MW3077: 00.0 轴恢复原点速度</b>
23	Y_CDUTY	本地	UINT	%MW3078	5	<b>MW3078: 00.0 轴爬行速度占空</b>
24	Y_CHANGE_TIME	本地	DINT	%MD3073	100	<b>MD3073: 00.0 轴恢复原点变速</b>



按下开始按钮后,轴 Q0.0 开始回原使能、运行使能,按照设定的回原速度回到原点,靠近原点时,切换为爬行速度,直到



# Installation Manual

修订历史

版本	日期	原因
V0.0	2025/03/15	创建文档
V1.0	2025/09/04	添加更新系统时间的步骤
V1.1	2026/01/06	修改示例
V1.2	2026/01/14	1、添加主从机排针连接实物图。 2、修改报警灯号与规格书一致。 3、修改部分排版。