HAITIAN
DRIVE SYSTEMS

E300 系列可编程控制器

用户手册【CODESYS 软件篇】

宁波海天驱动技术有限公司

发布版本：V1.10

发布日期：2024-08-12

权利声明

版权声明

商标声明

免责声明

本手册依据现有信息制作，内容如有更改，恕不另行通知。宁波海天驱动有限公司在编写该手册时，已尽最大努力保证其内容准确可靠，但不对本手册中的遗漏、不准确或印刷错误导致的损失和损害承担责任。

前 言

感谢您购买宁波海天驱动有限公司的 E300 系列 PLC 产品！

在使用我公司 E300 系列 PLC 产品前，敬请您仔细阅读本手册，以便更清楚地掌握产品的特性，更安全地应用，充分利用本产品丰富的功能。

安全注意事项

负责产品安装、操作的人员必须经过严格培训，遵守相关行业的安全规范，严格遵守该手册提供的相关设备注意事项和特殊安全指示，按正确的操作方法进行设备的各项操作。

为防止人身伤害或设备损坏，对务必遵守的事项特做一下说明。错误使用本产品而可能带来的危害和损害程度，见相关符号说明。

	**警告** 表示由于没有按要求操作造成的危险，可能导致人身伤亡。
	**注意** 表示由于没有按要求操作造成的危险，可能会导致人身轻度或中度伤害和设备损坏。
	**提示** 表示对操作的描述进行必要的补充或说明。

版本履历

将手册版本升级所更新的内容罗列出来并插入超链接，可从此处直接跳转到特定修改处，便于查看修改内容。

发行日期	版本	修订内容
2023 年 6 月	V1.00	• 首版发布
2024 年 6 月	V1.10	•

目 录

CODESYS 软件概述

1

1.1 CODESYS 软件介绍

CODESYS 软件适用于 E300 系列 PLC 组态编程，采用集成化的开发环境，用户界面友好，功能丰富，可对 E300 系列 PLC 工程进行组态、编译、工程下载等操作。用到的 CODESYS 平台主要有 SP15 和 SP19 版本，因此全文主要对这个两个版本进行操作说明。CODESYS SP19 是 SP15 的升级，两个版本的操作大多数是相同的。

CODESYS 支持多种语言编程：指令表（IL）、结构化文本（ST）、梯形图（LD）、功能块图（FBD）、连续功能图（CFC）。

OCDESYS 的特性如下：

1、自带多种功能块库并支持用户自定义库。

2、支持离线仿真，不需要连接 PLC，伺服等实物连接即可完成程序仿真调试。

3、强大的通讯功能：支持 EtherCAT、CANopen、EtherNet/IP、Modbus、Profinet 等通信。

4、强大的运控功能：单轴控制、电子齿轮、电子凸轮、多轴 CNC 等。

1.2 软件安装包的获取

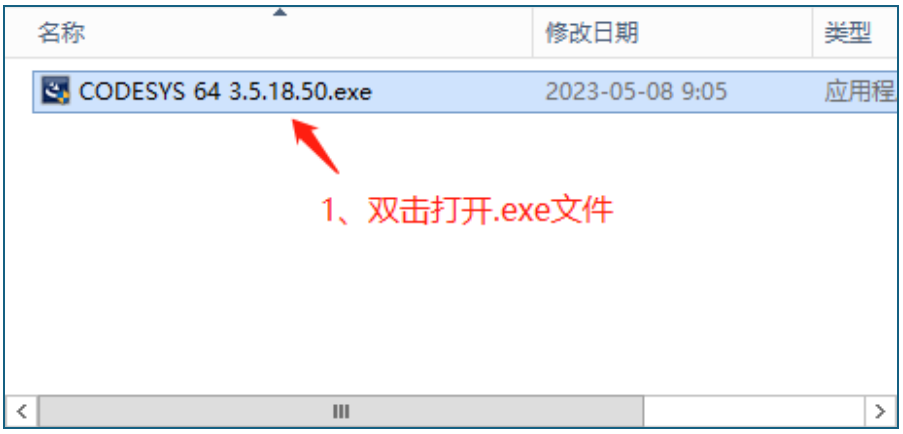
◆ **CODESYS SP19 下载链接：**

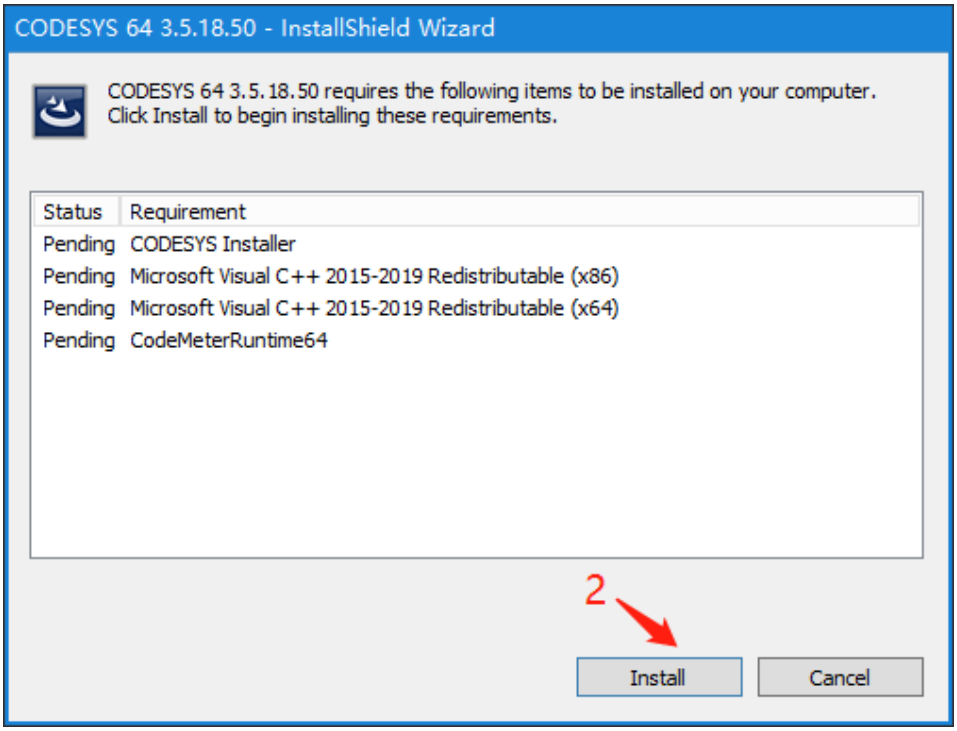
CODESYS Development System V3 | CODESYS Store International

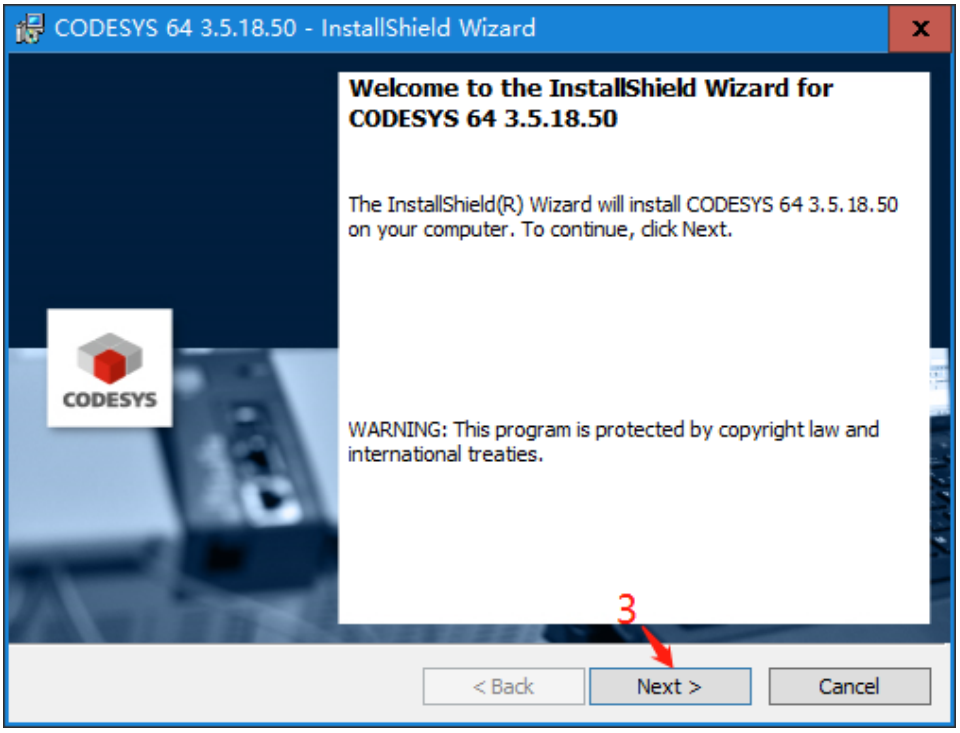
1.3 软件安装与卸载

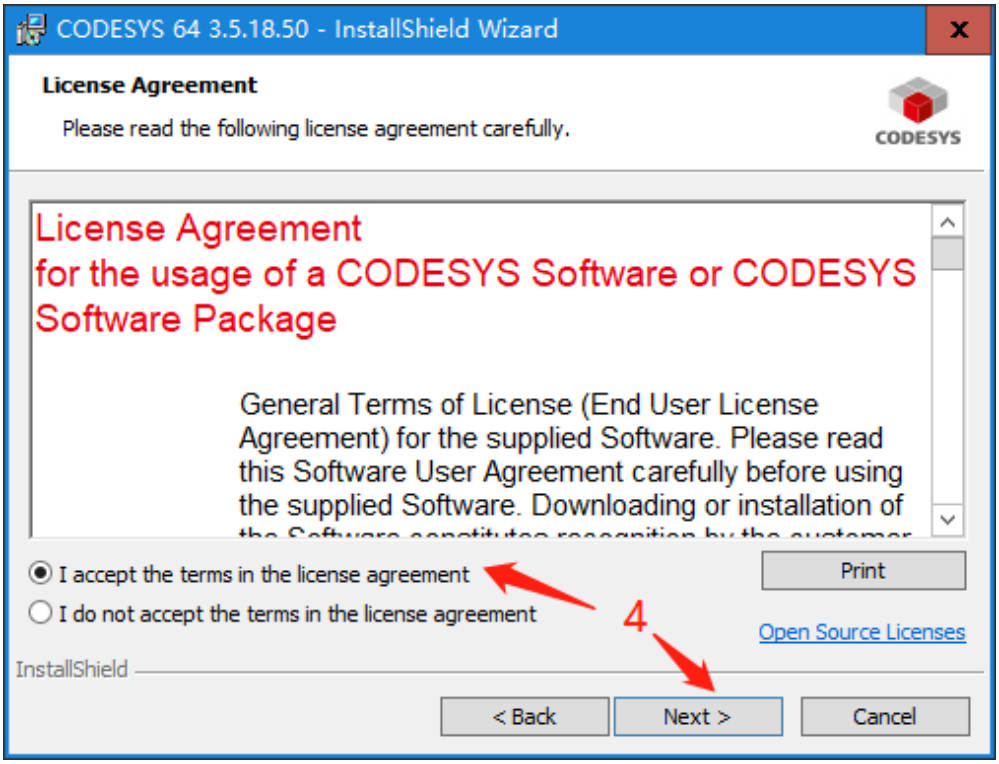
安装注意事项：安装前请关闭电脑的杀毒软件。

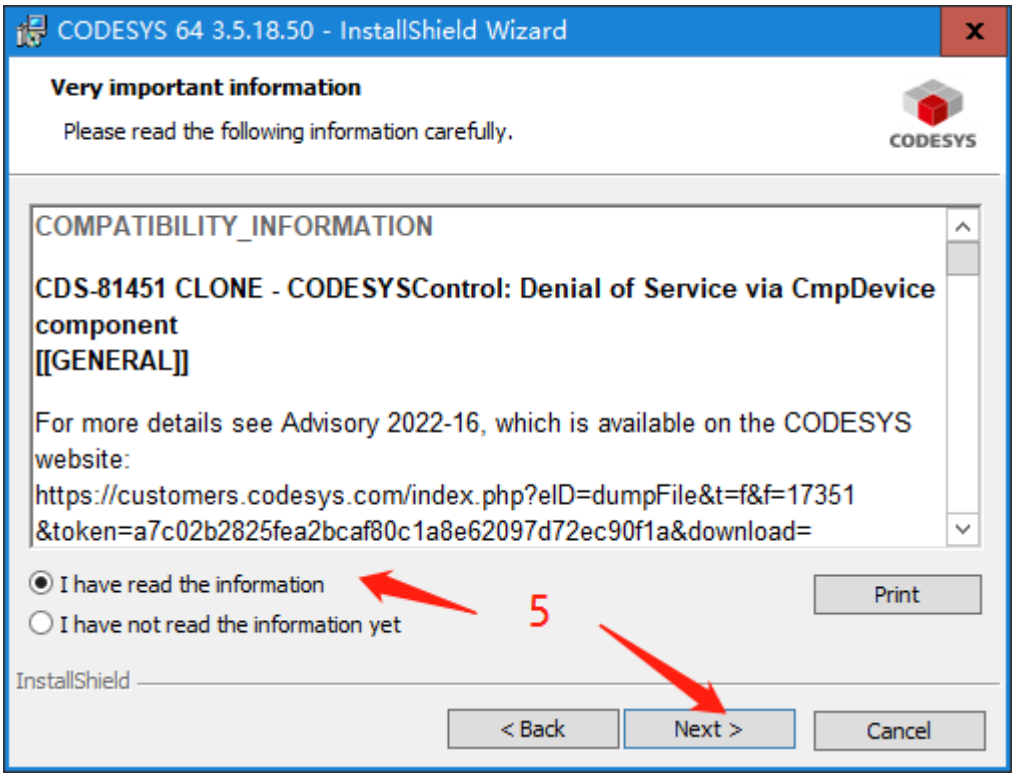
安装步骤（以 SP18 为例，SP19 与之类似）：

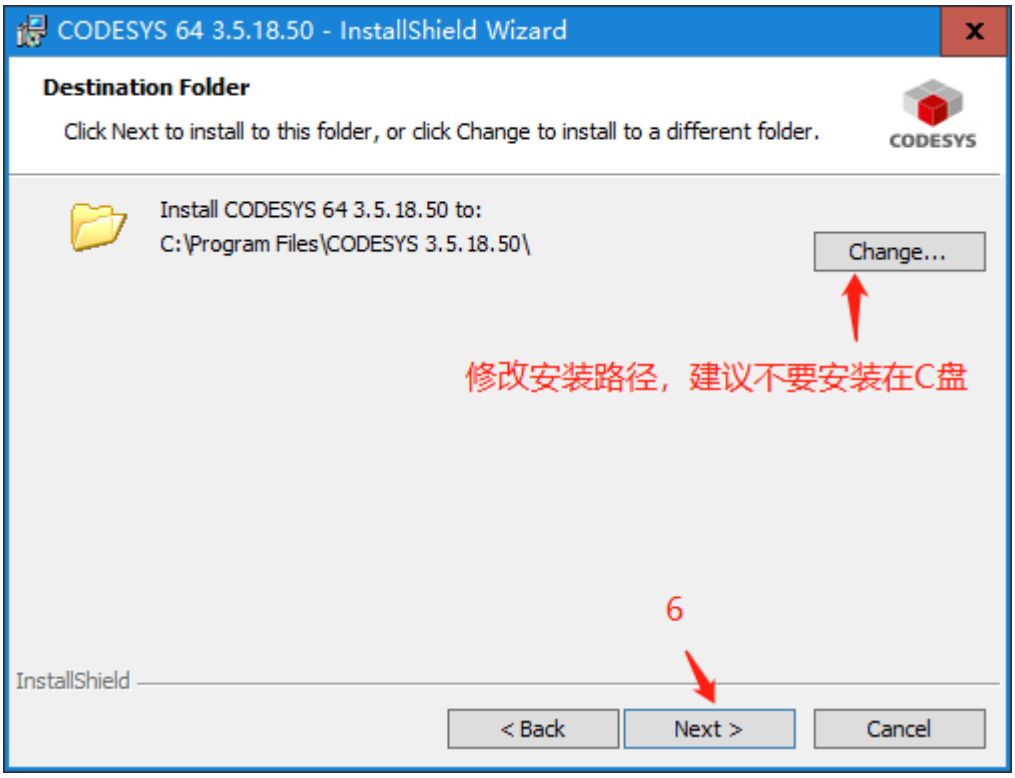


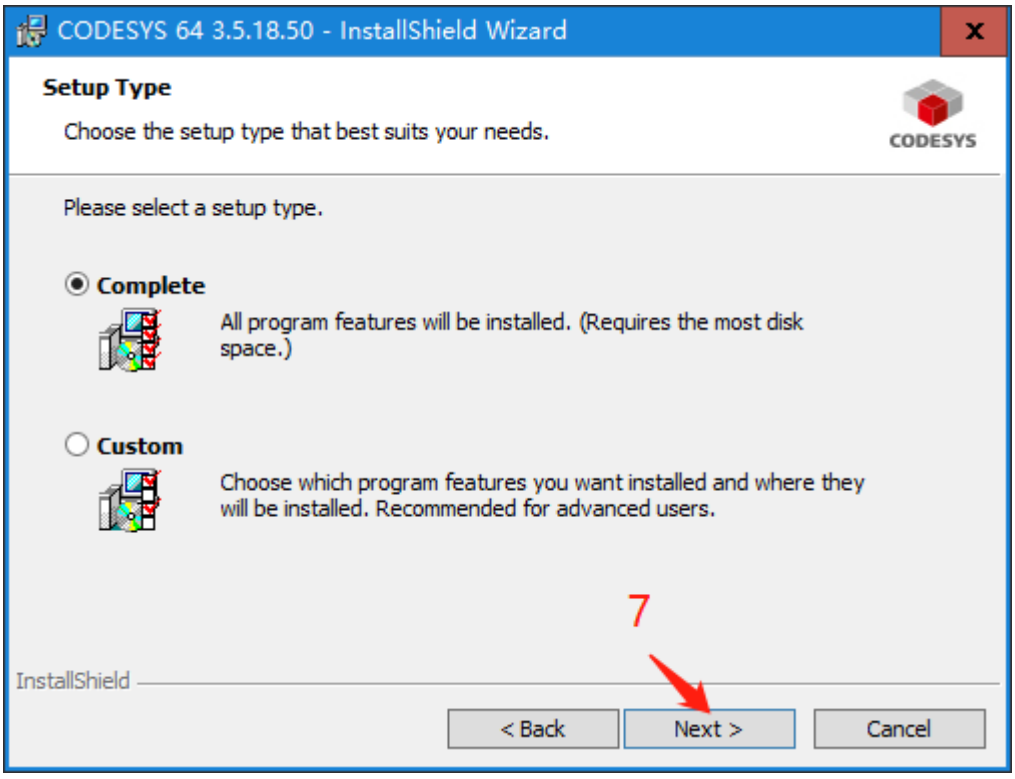

CODESYS 64 3.5.18.50 - InstallShield Wizard
CODESYS 64 3.5.18.50 requires the following items to be installed on your computer.
Click Install to begin installing these requirements.
Status
Requirement
Pending
CODESYS Installer
Pending
Microsoft Visual C++ 2015-2019 Redistributable (x86)
Pending
Microsoft Visual C++ 2015-2019 Redistributable (x64)
Pending
CodeMeterRuntime64
2
Install
Cancel

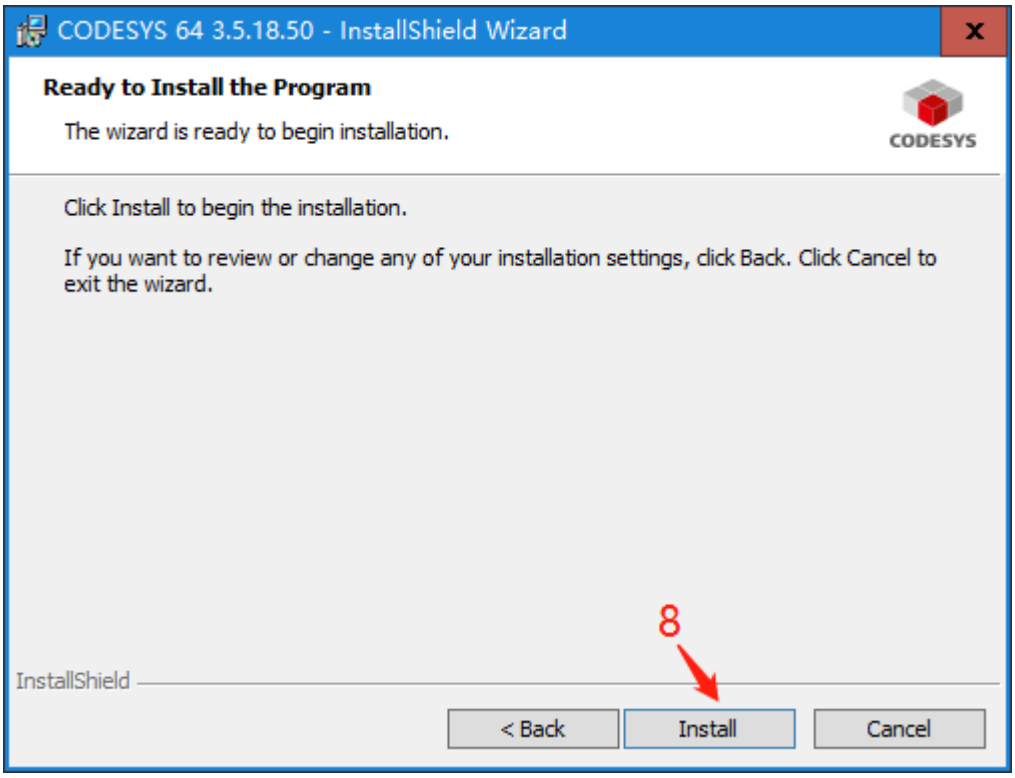

CODESYS 64 3.5.18.50 - InstallShield Wizard
Welcome to the InstallShield Wizard for CODESYS 64 3.5.18.50
The InstallShield(R) Wizard will install CODESYS 64 3.5.18.50 on your computer. To continue, click Next.
WARNING: This program is protected by copyright law and international treaties.
CODESYS
3
< Back
Next >
Cancel

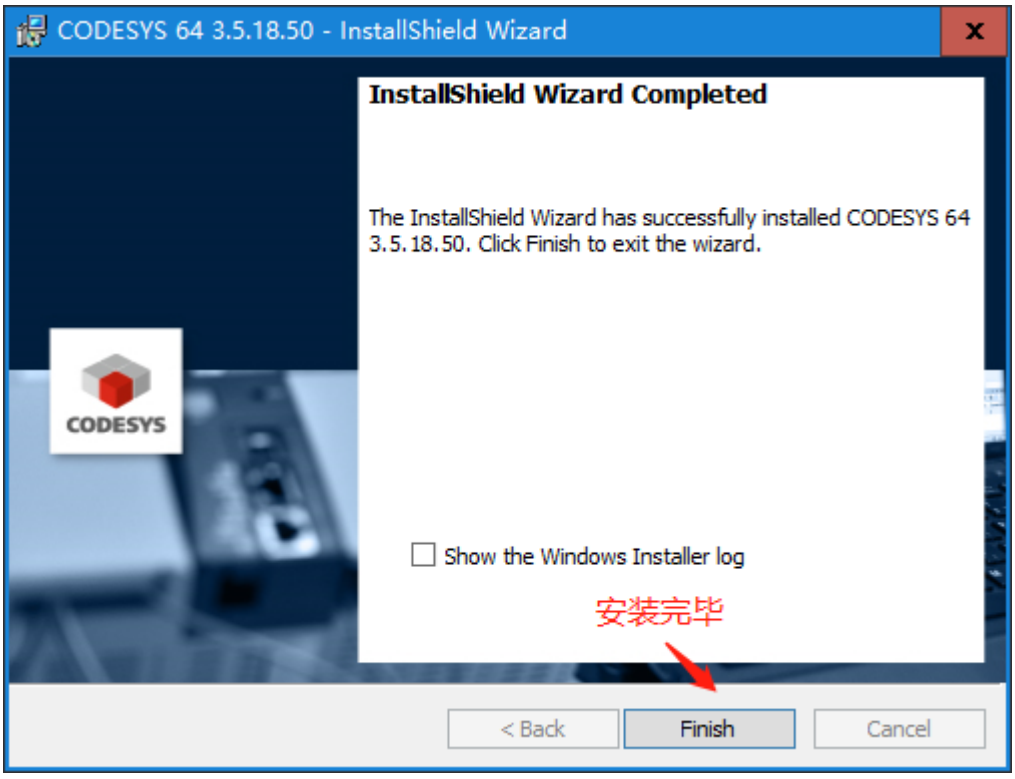

CODESYS 64 3.5.18.50 - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
CODESYS
License Agreement
for the usage of a CODESYS Software or CODESYS Software Package
General Terms of License (End User License Agreement) for the supplied Software. Please read this Software User Agreement carefully before using the supplied Software. Downloading or installation of
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Print
Open Source Licenses
4
InstallShield
< Back
Next >
Cancel


CODESYS 64 3.5.18.50 - InstallShield Wizard
Very important information
Please read the following information carefully.
CODESYS
COMPATIBILITY_INFORMATION
CDS-81451 CLONE - CODESYSControl: Denial of Service via CmpDevice component
[[GENERAL]]
For more details see Advisory 2022-16, which is available on the CODESYS website:
https://customers.codesys.com/index.php?eID=dumpFile&t=f&f=17351&token=a7c02b2825fea2bcaf80c1a8e62097d72ec90f1a&download=
I have read the information
I have not read the information yet
Print
5
InstallShield
< Back
Next >
Cancel


CODESYS 64 3.5.18.50 - InstallShield Wizard
Destination Folder
Click Next to install to this folder, or click Change to install to a different folder.
CODESYS
Install CODESYS 64 3.5.18.50 to:
C:\Program Files\CODESYS 3.5.18.50\
Change...
修改安装路径，建议不要安装在C盘
6
InstallShield
< Back
Next >
Cancel


CODESYS 64 3.5.18.50 - InstallShield Wizard
Setup Type
Choose the setup type that best suits your needs.
CODESYS
Please select a setup type.
Complete
All program features will be installed. (Requires the most disk space.)
Custom
Choose which program features you want installed and where they will be installed. Recommended for advanced users.
7
InstallShield
< Back
Next >
Cancel

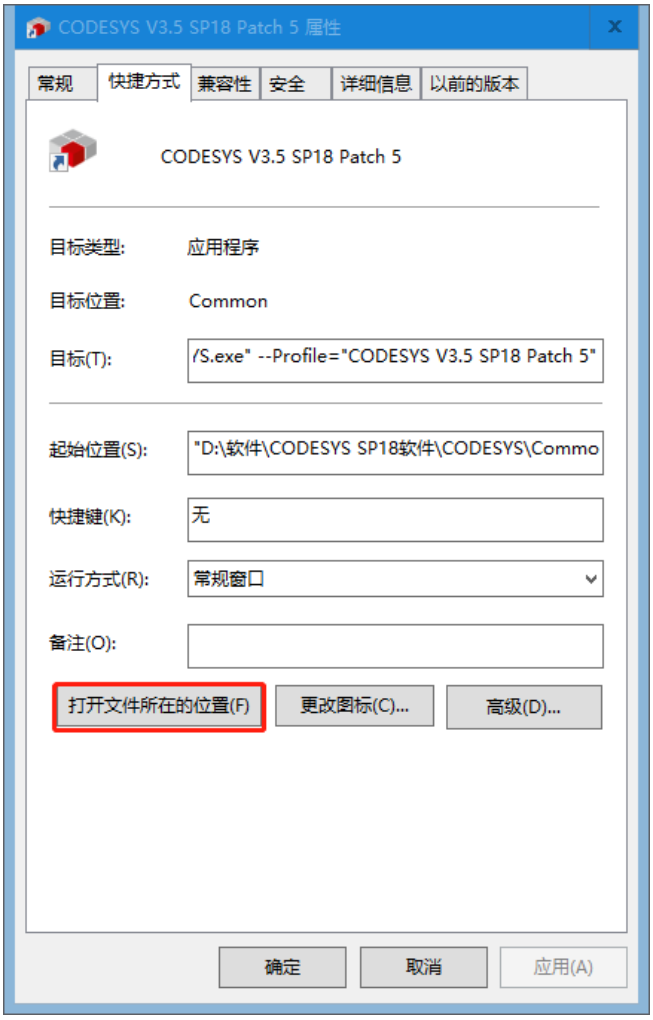


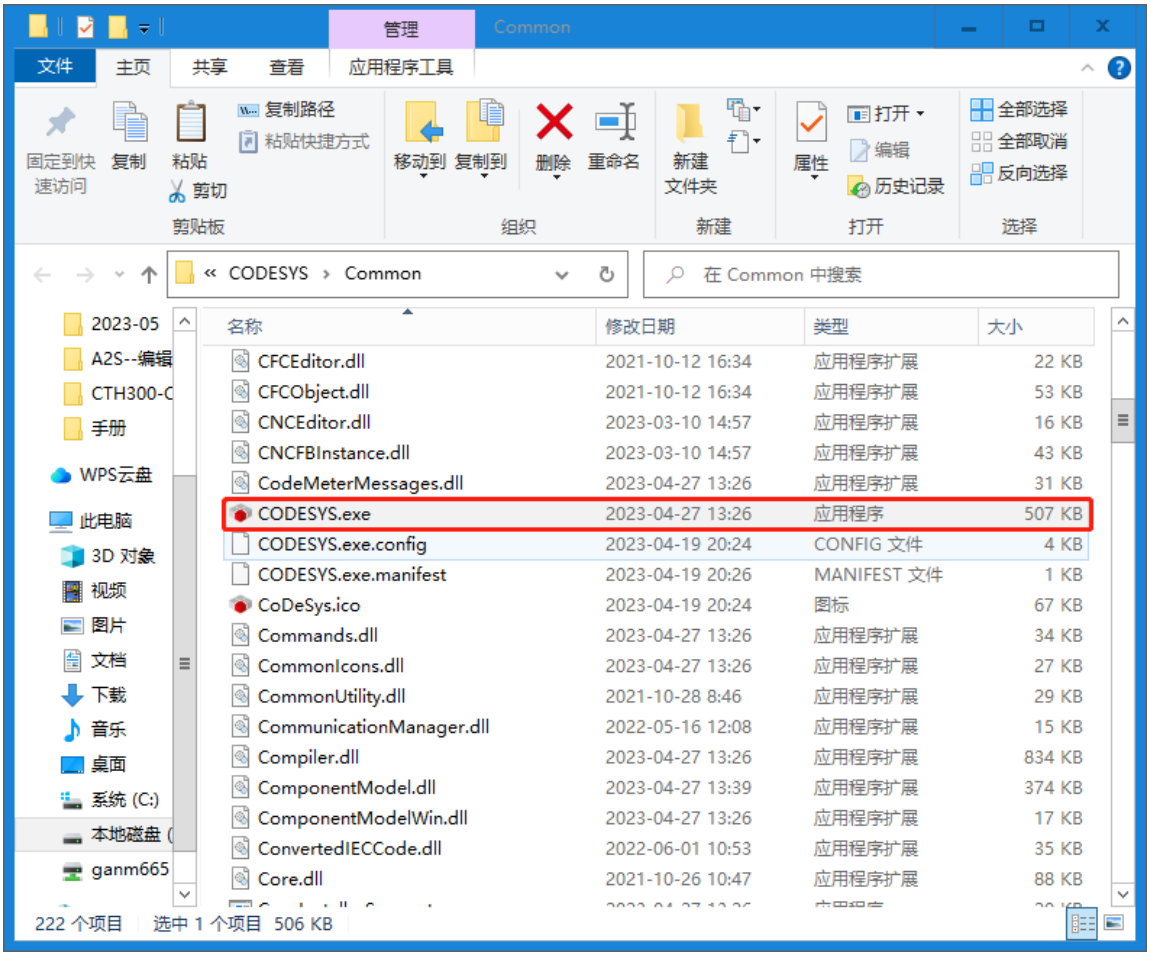


卸载软件：

方法 1、通过找到软件所在路径卸载

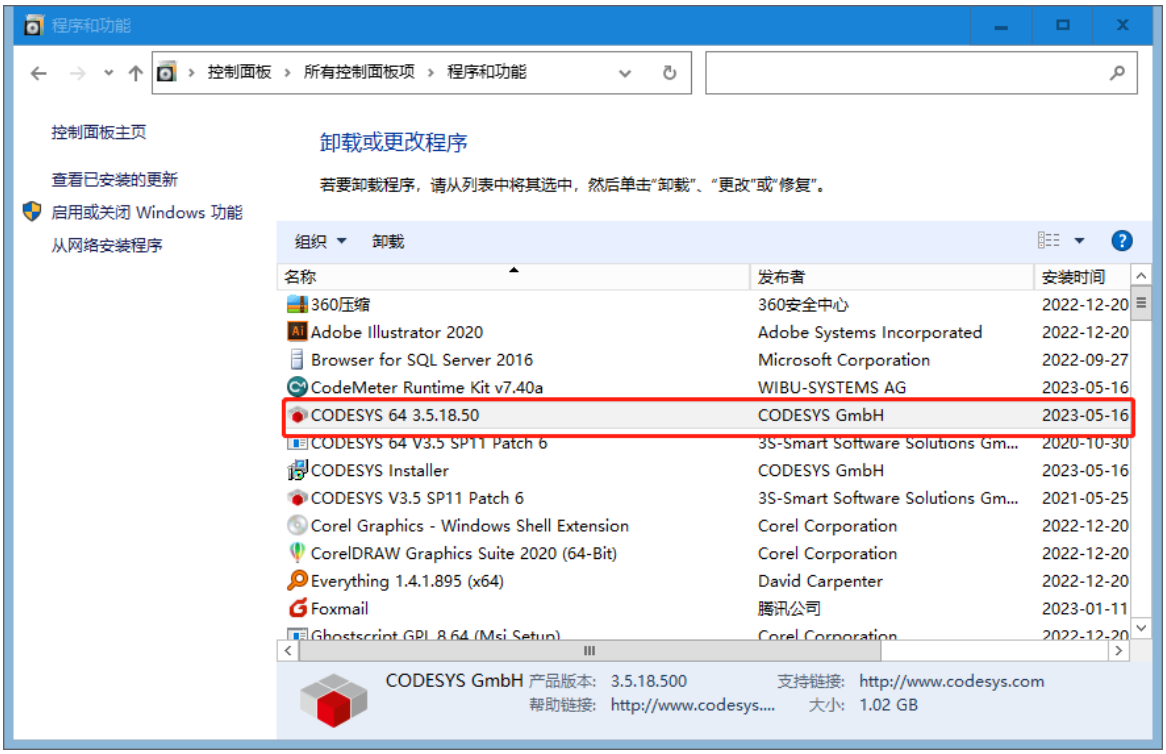
右键点击软件图标并选择属性，打开文件所在的位置后找到 CODESYS.exe 文件，双击该文件确认删除即可。


CODESYS V3.5 SP18 Patch 5 属性
常规
快捷方式
兼容性
安全
详细信息
以前的版本
CODESYS V3.5 SP18 Patch 5
目标类型:
应用程序
目标位置:
Common
目标(T):
/S.exe" --Profile="CODESYS V3.5 SP18 Patch 5"
起始位置(S):
"D:\软件\CODESYS SP18软件\CODESYS\Commo
快捷键(K):
无
运行方式(R):
常规窗口
备注(O):
打开文件所在的位置(F)
更改图标(C)...
高级(D)...
确定
取消
应用(A)


管理
Common
文件
主页
共享
查看
应用程序工具
固定到快速访问
复制
粘贴
剪切
复制路径
粘贴快捷方式
剪贴板
移动到
复制到
删除
重命名
组织
新建文件夹
新建
属性
打开
编辑
历史记录
打开
全部选择
全部取消
反向选择
选择
« CODESYS › Common
在 Common 中搜索
2023-05
A2S--编辑
CTH300-C
手册
WPS云盘
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
系统 (C:)
本地磁盘 (
ganm665
名称
修改日期
类型
大小
CFCEditor.dll 2021-10-12 16:34 应用程序扩展 22 KB
CFCObject.dll 2021-10-12 16:34 应用程序扩展 53 KB
CNCEditor.dll 2023-03-10 14:57 应用程序扩展 16 KB
CNCFBInstance.dll 2023-03-10 14:57 应用程序扩展 43 KB
CodeMeterMessages.dll 2023-04-27 13:26 应用程序扩展 31 KB
CODESYS.exe 2023-04-27 13:26 应用程序 507 KB
CODESYS.exe.config 2023-04-19 20:24 CONFIG 文件 4 KB
CODESYS.exe.manifest 2023-04-19 20:26 MANIFEST 文件 1 KB
CoDeSys.ico 2023-04-19 20:24 图标 67 KB
Commands.dll 2023-04-27 13:26 应用程序扩展 34 KB
CommonIcons.dll 2023-04-27 13:26 应用程序扩展 27 KB
CommonUtility.dll 2021-10-28 8:46 应用程序扩展 29 KB
CommunicationManager.dll 2022-05-16 12:08 应用程序扩展 15 KB
Compiler.dll 2023-04-27 13:26 应用程序扩展 834 KB
ComponentModel.dll 2023-04-27 13:39 应用程序扩展 374 KB
ComponentModelWin.dll 2023-04-27 13:26 应用程序扩展 17 KB
ConvertedIECCode.dll 2022-06-01 10:53 应用程序扩展 35 KB
Core.dll 2021-10-26 10:47 应用程序扩展 88 KB
222 个项目
选中 1 个项目 506 KB

方法 2、通过控制面板卸载

打开控制面板→程序和功能，找到 CODESYS SP18 直接卸载即可。

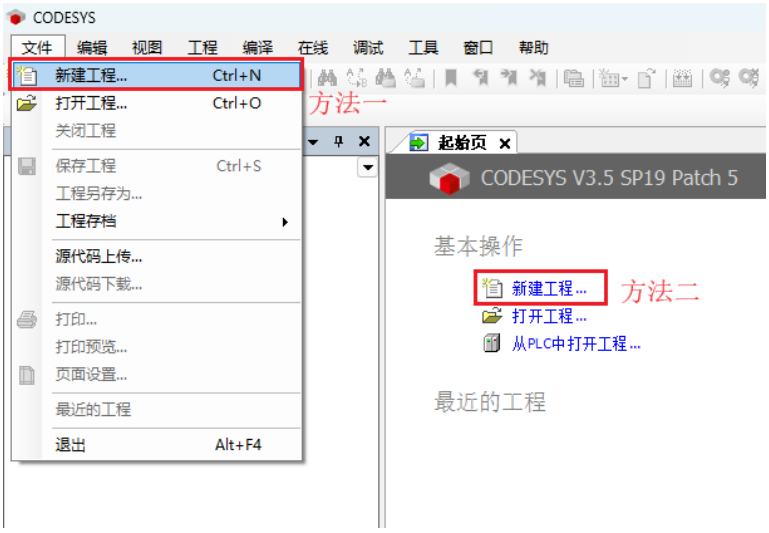


快速入门

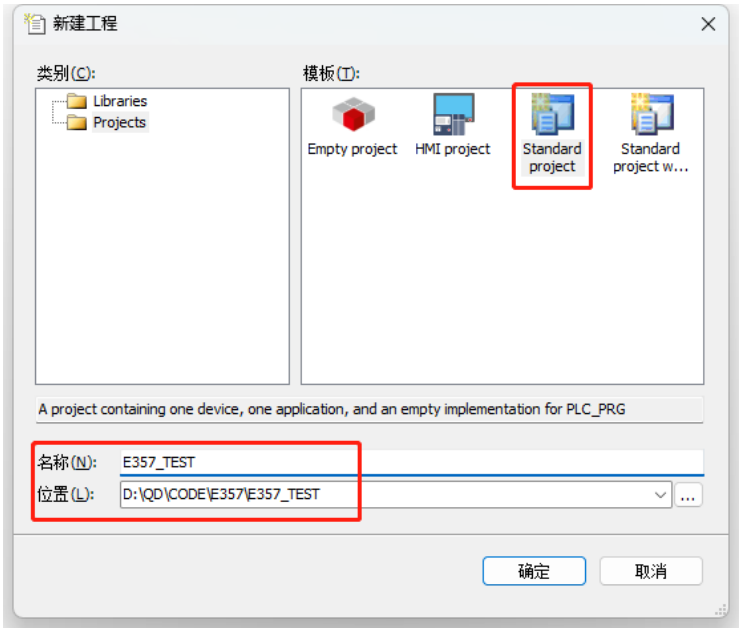
2

2.1 新建工程

1、启动 CODESYS SP19，起始界面如下，点击新建工程。

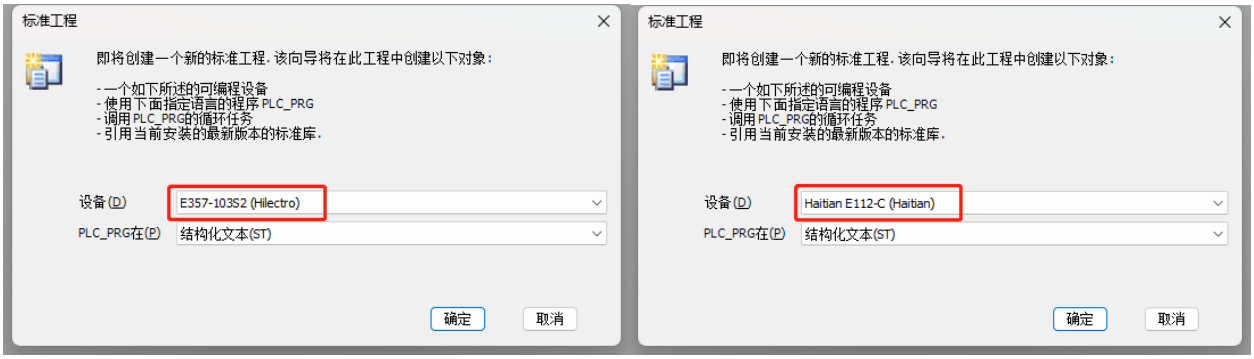


2、新建一个标准工程，设置工程名称和工程保存路径。

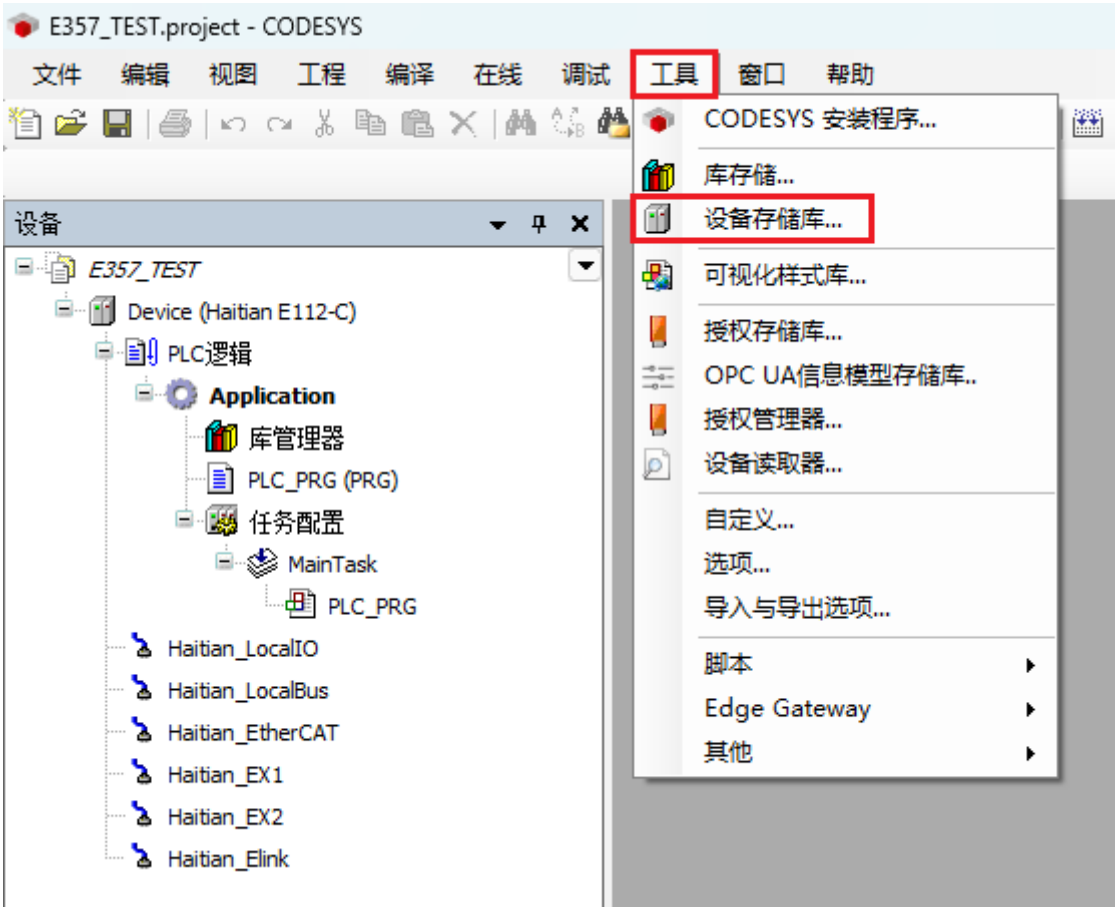


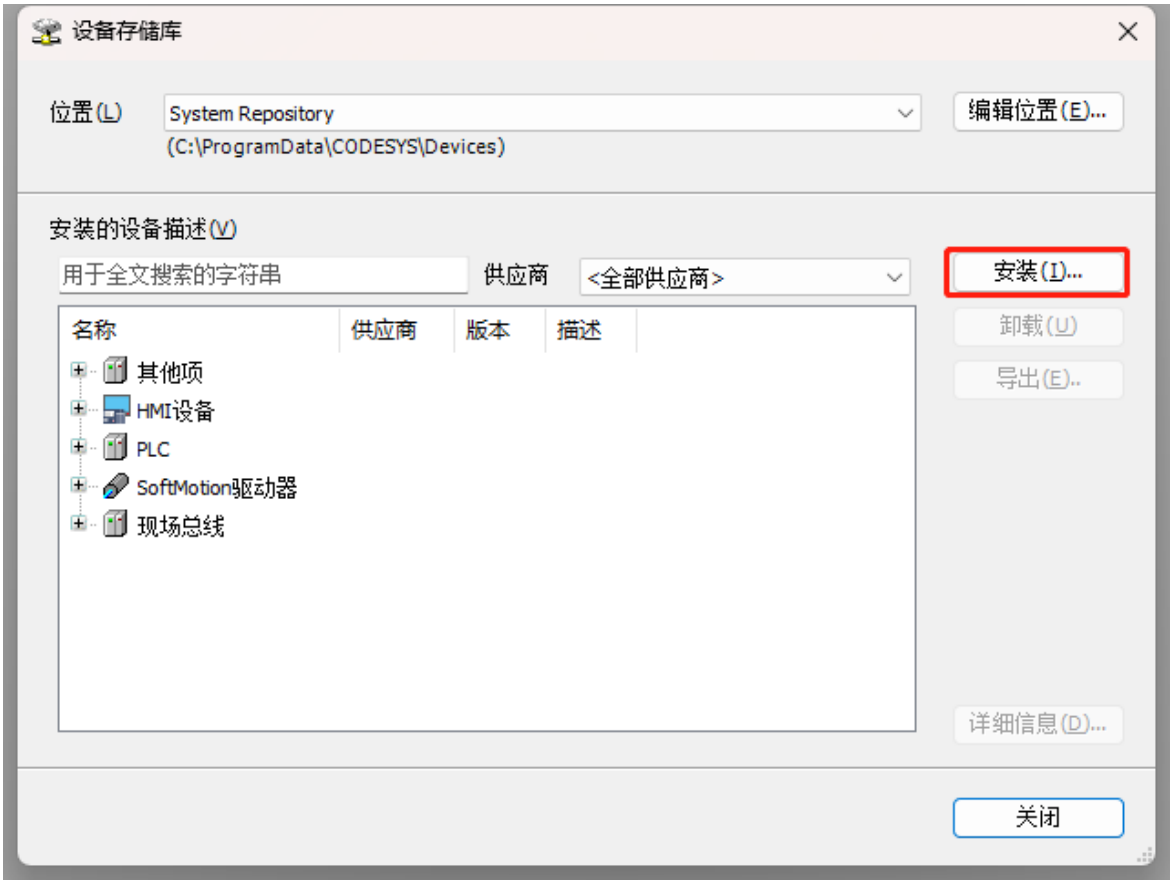
3、选择设备以及编程语言，若没有所需设备，可以选择默认设备创建工程，然后在工程中安装所需设备的描述文件，安装完成后直接更新工程设备即可。具体操作如下：

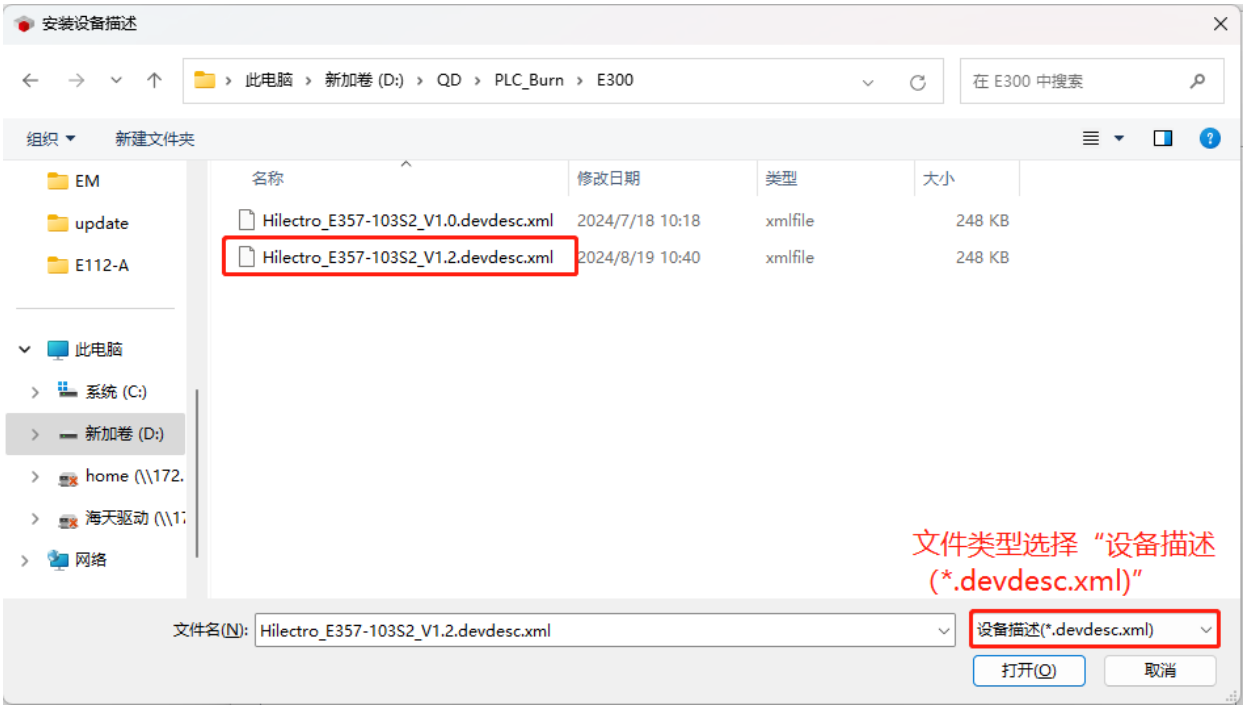
1）本次需要创建一个 E357 的工程。若已安装该设备的描述文件，则直接选择对应设备；若没有安装该设备的描述文件，我们创建一个已有的 E112 工程。



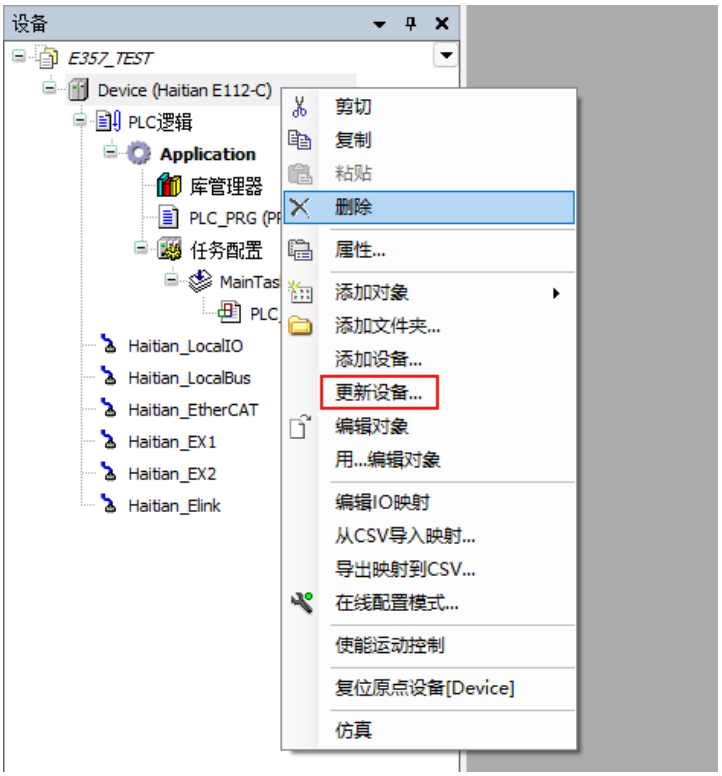
2）进入工程界面，安装 E357 的设备描述文件。

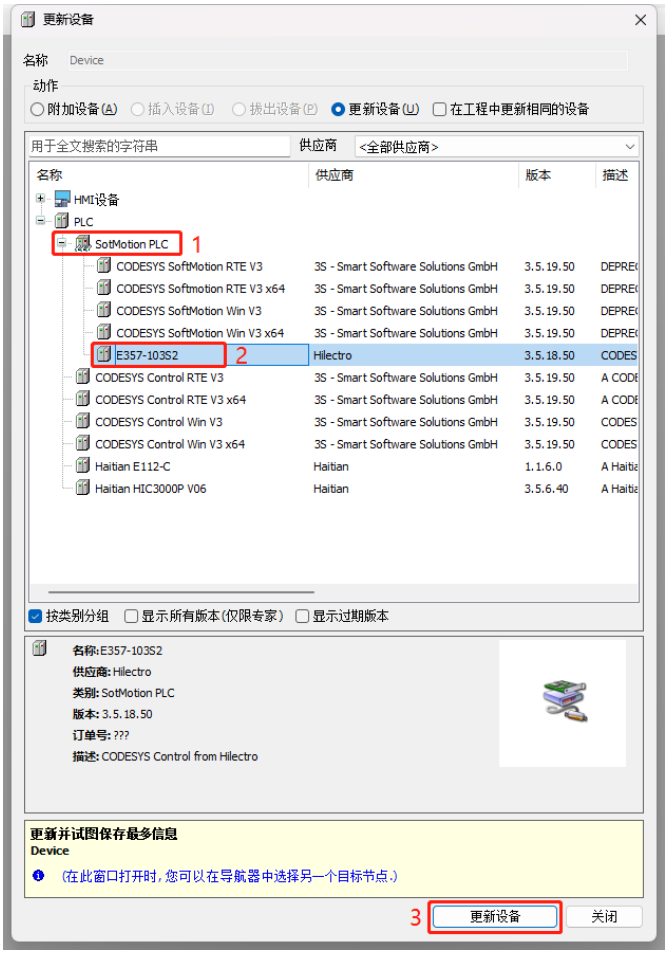




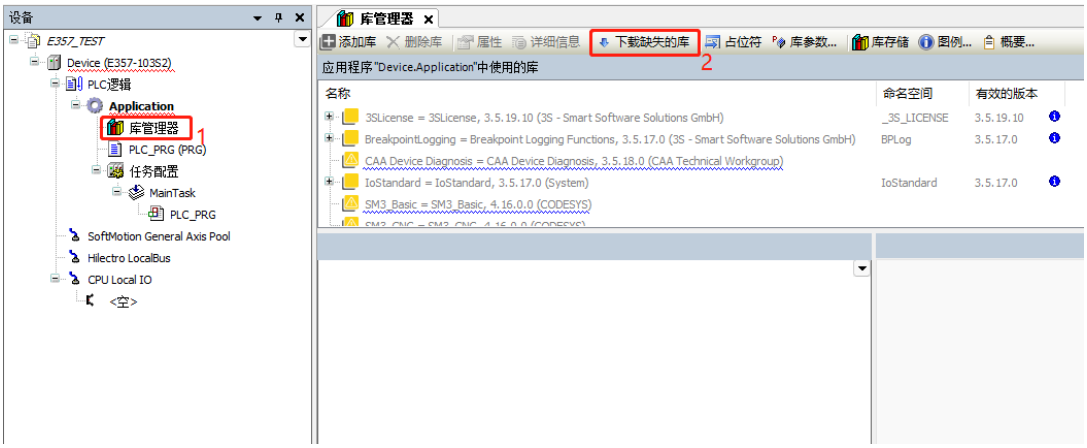


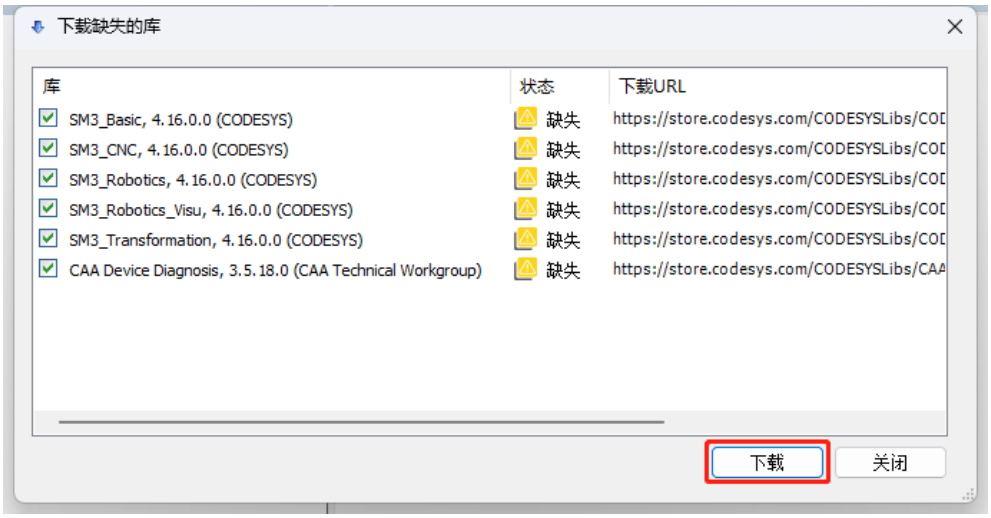
3）右键点击 Device，更新工程设备。



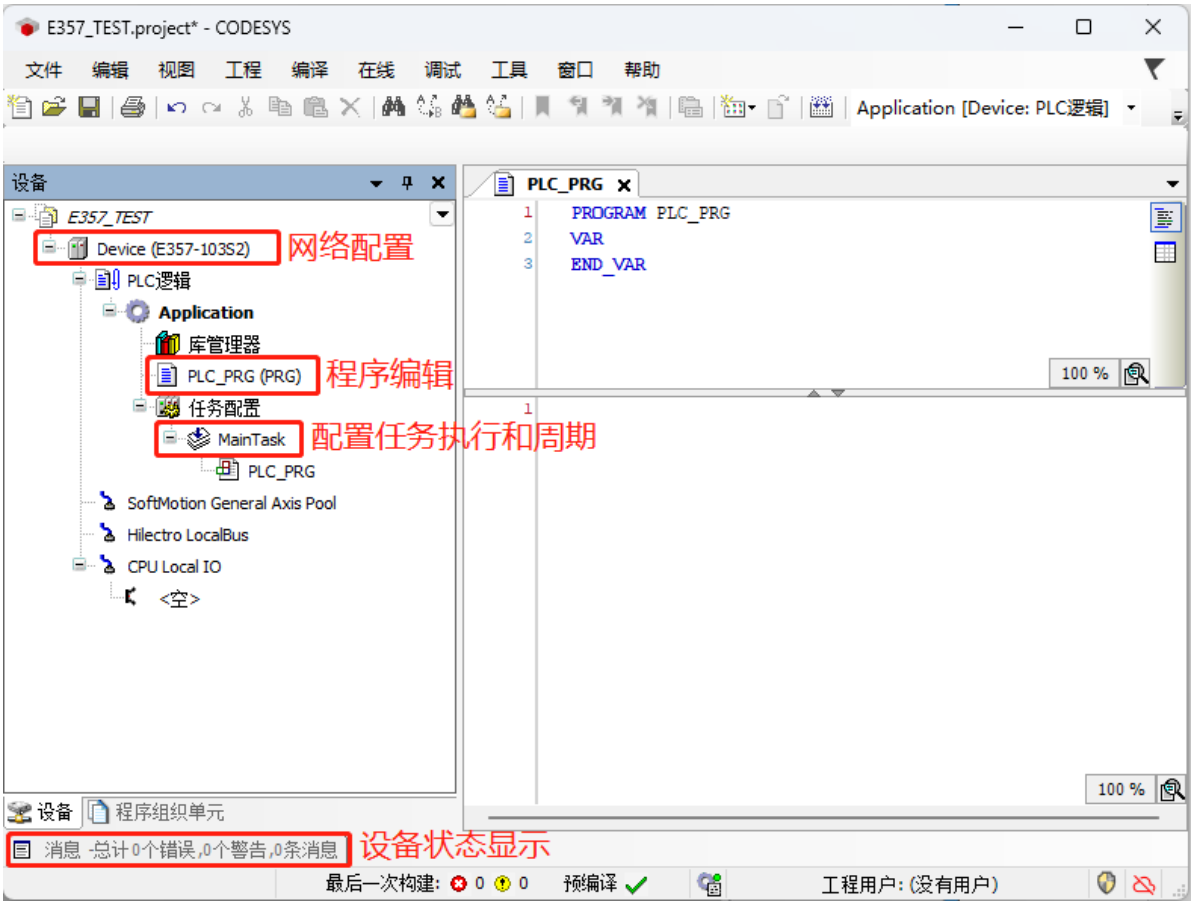


4、下载缺失的库，下载期间需联网。





5、工程基本界面如下：



2.2 设置通信

1、将编程设备的 IP 设为与 PLC 同一个网段

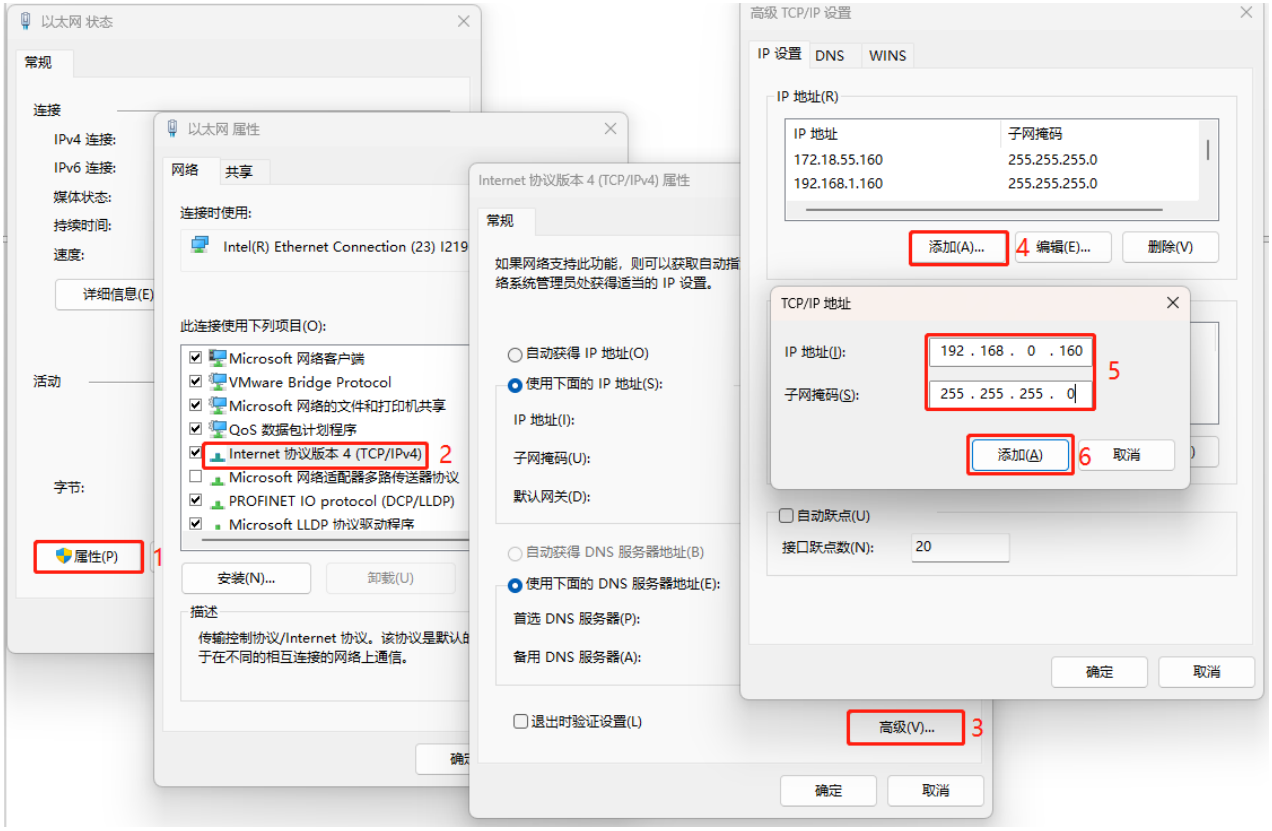
设置通信前，需要将编程 PC 的 IP 设置与 PLC（IP：192.168.0.x）同一个网段.

设置方法：

1）将 PC 机的以太网属性打开，

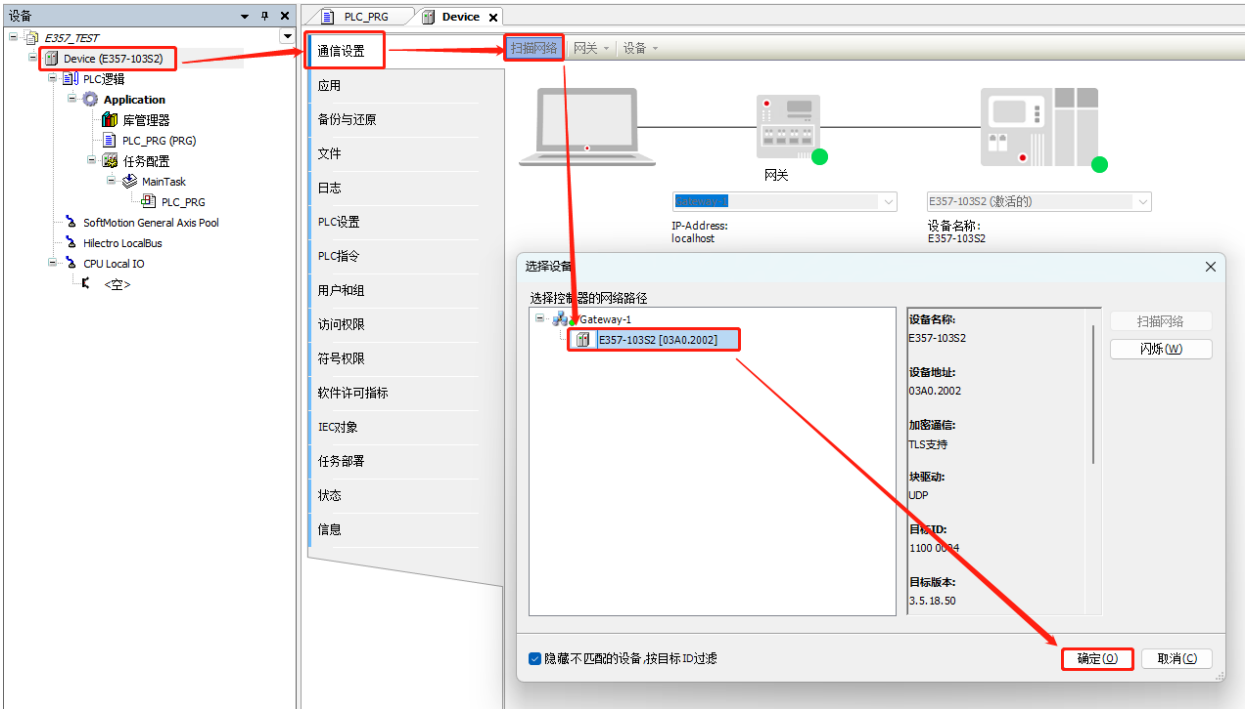
2）双击 Internet TCP/IP 协议，

3）将“自动获得 IP 地址”更改为“使用下面的 IP 地址”，然后在 IP 地址中填写“192.168.0.X”即可。



2、在 CODESYS 设备视图中执行“通讯设置”

在设备视图中双击【Device】，然后在选项卡【通信设置】中点击【扫描网络】，选中扫描出来的设备并确定，右边显示设备信息。



回到通讯设置界面可看到两个绿色指示灯，表示通讯通道激活，也就是所有与通讯相关的操作与该通道关联，设备成功通信。

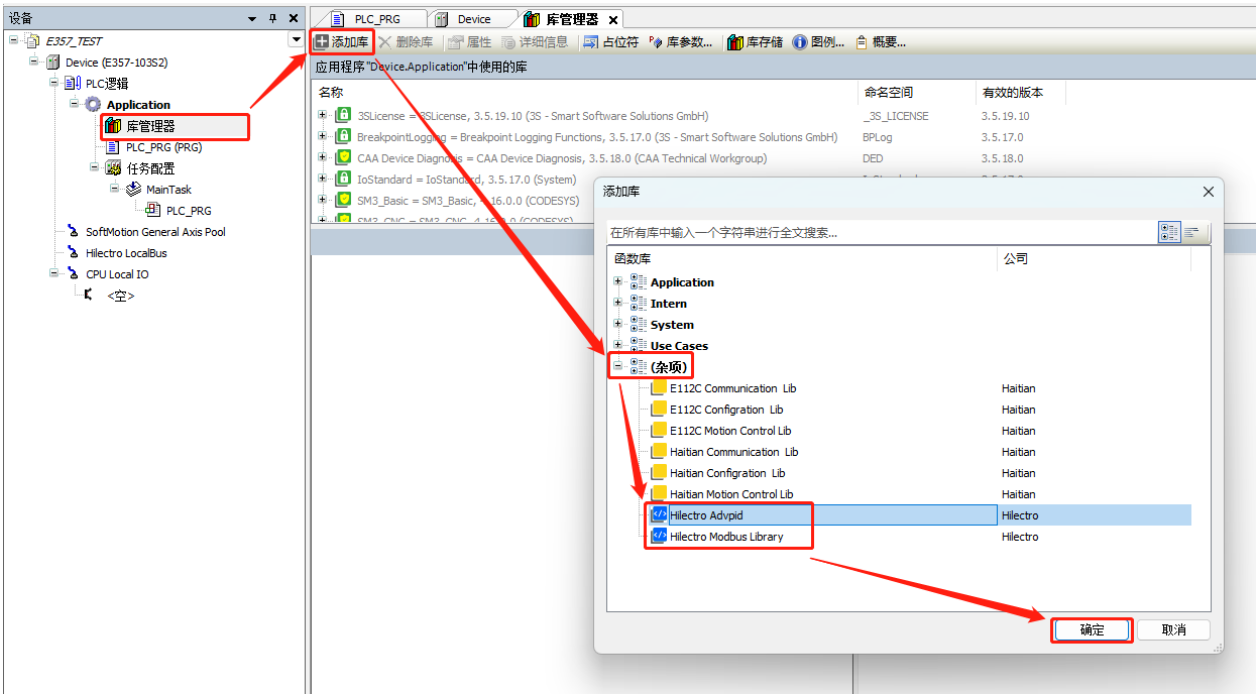
2.3 工程组态

一个标准的 PLC 工程，包含程序，任务和库。

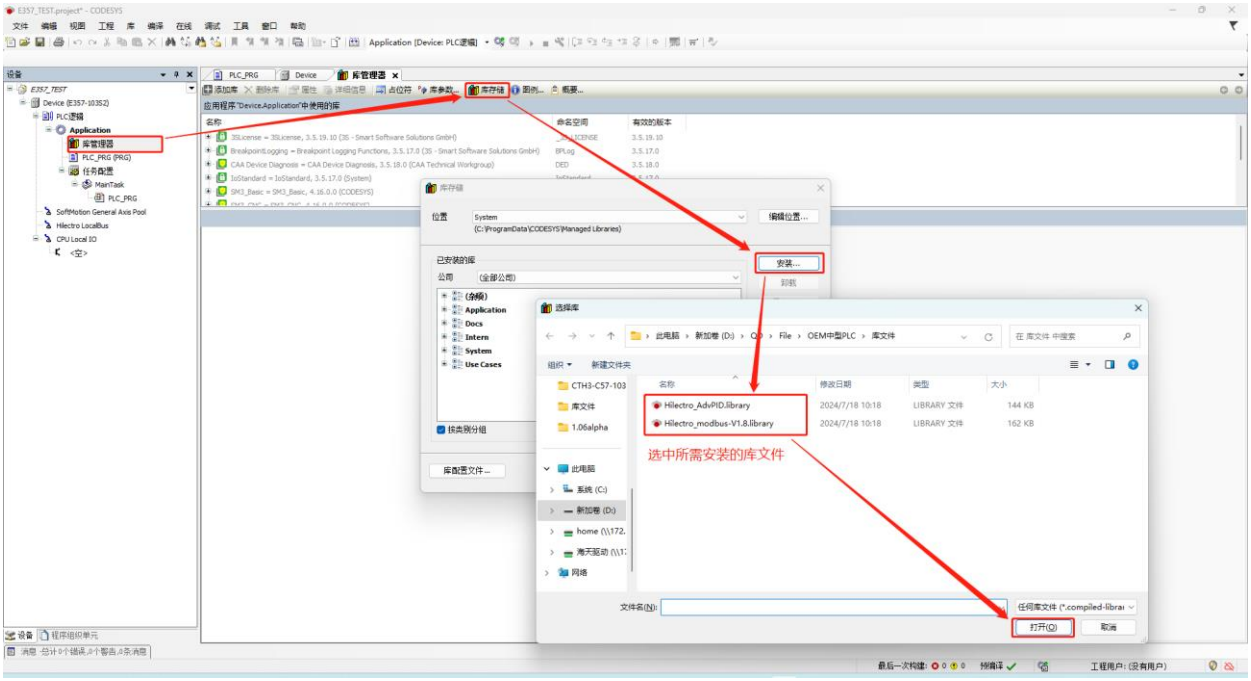
2.3.1 添加库

编写程序时，经常用到一些指令或功能块，比如触发器、功能块、字符串处理指令、计数功能块、PID 控制功能块等。CODESYS 将这些指令和功能块进行分类，然后建立专门的库。

一般工程建立时，默认 Standard 库是在工程中的。当工程中用到其他库的指令时，就需要添加库到工程中。



<注意> 添加库文件到工程中和添加库到 CODESYS 中是不一样的，首先我们需要添加外部库到 CODESYS 中，如此在 CODESYS 中有了外部库，然而建立工程时这些外部库是没有包含在工程中的，当工程需要用到这些外部库的时候就需要将库添加进去。

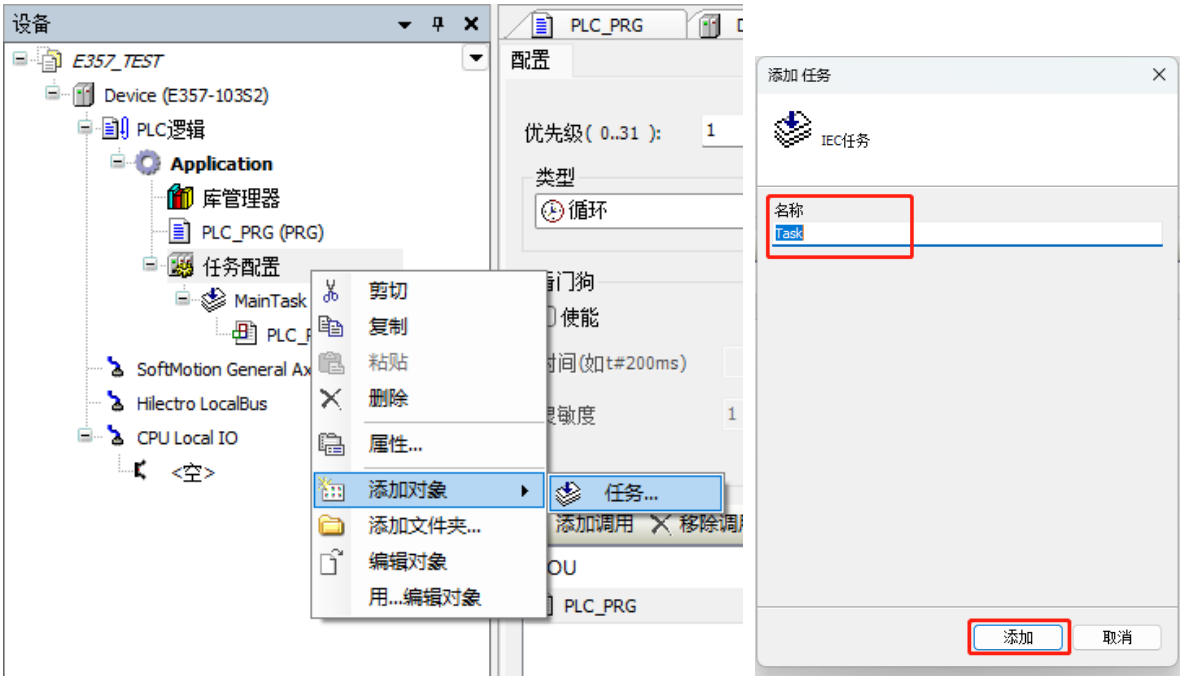


2.3.2 配置任务

在树形菜单任务配置中可以进行任务的管理，新建一个标准的 PLC 项目会自动生成一个循环执行的任务，该任务自动关联 PLC_PRG，任务周期默认是 4ms，优先级为 1，PLC 程序只有被任务调用才会参与编译和实际执行。右击任务配置→添加对象→任务，定义任务名称后完成新任务的创建，不同类型的任务最多可以创建 100 个，按照用户设定的优先级顺序执行，数字越小优先级越高，优先级相同的情况下，按照在任务配置当中的顺序从上往下执行。

1、新建任务

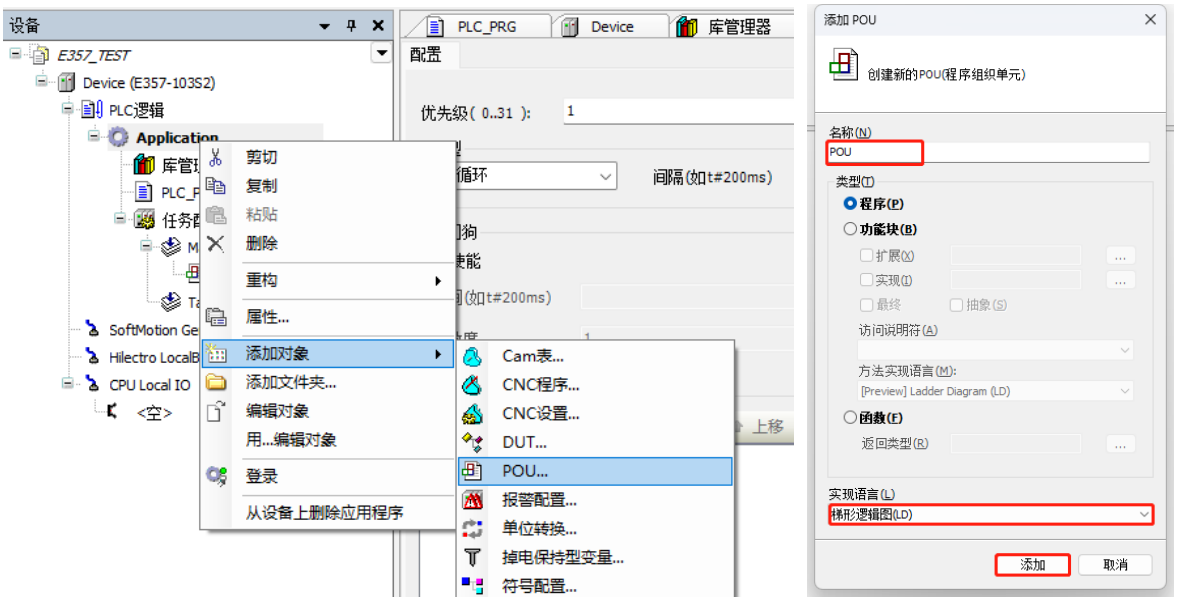
用户可自行新建任务，选中设备树中“任务配置”右键选择“添加对象”，点击“任务”，为该任务命名后点击“打开”。



2、新建程序

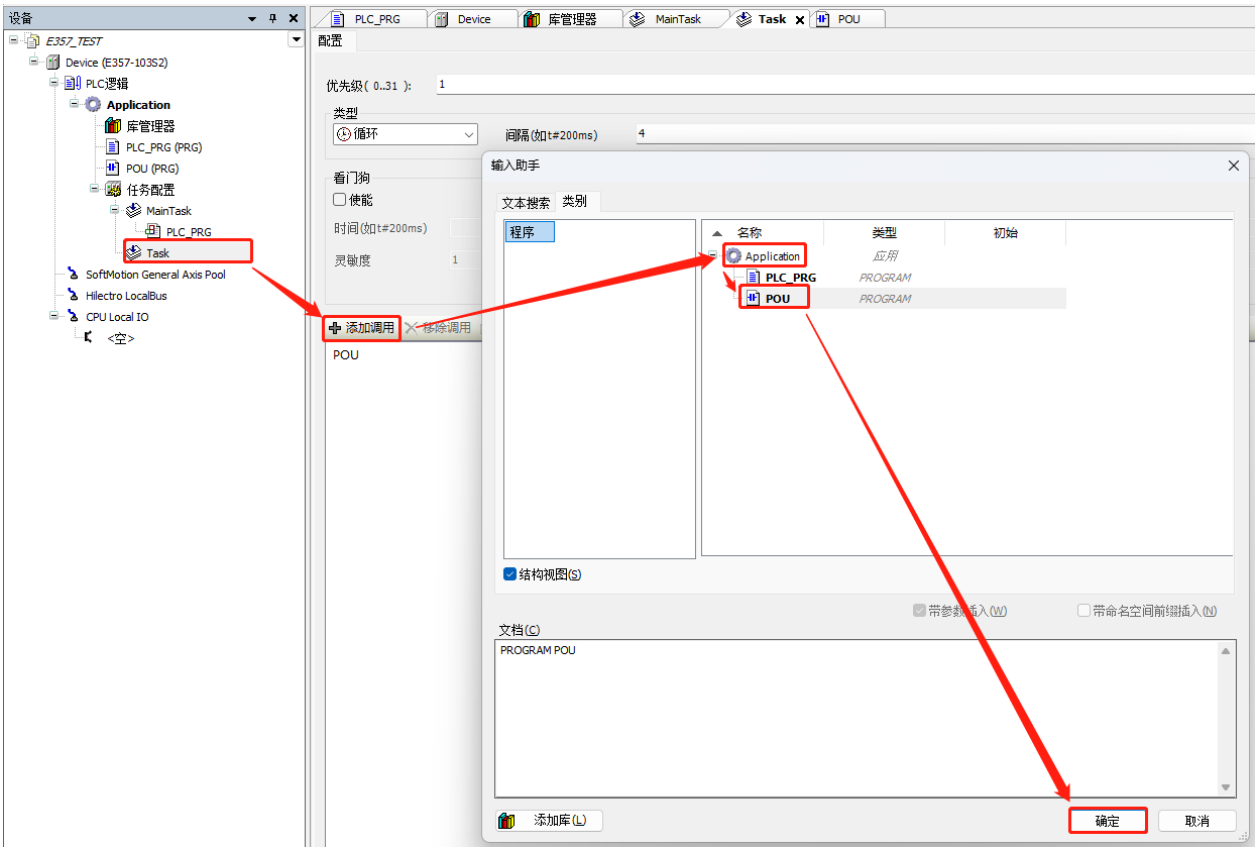
除新建工程时所创建的 PLC_PRG 程序文件外，用户也可根据自己的需求创建新的程序，右键 Application

选择“添加对象”，点击 POU，命名程序，选择语言，添加程序。



3、添加程序至任务

对于用户新建的 PLC 程序,需要手动进行任务配置和调用，否则程序不执行，双击 Task→添加调用→需要调用的 PLC 程序点击确定完成调用。

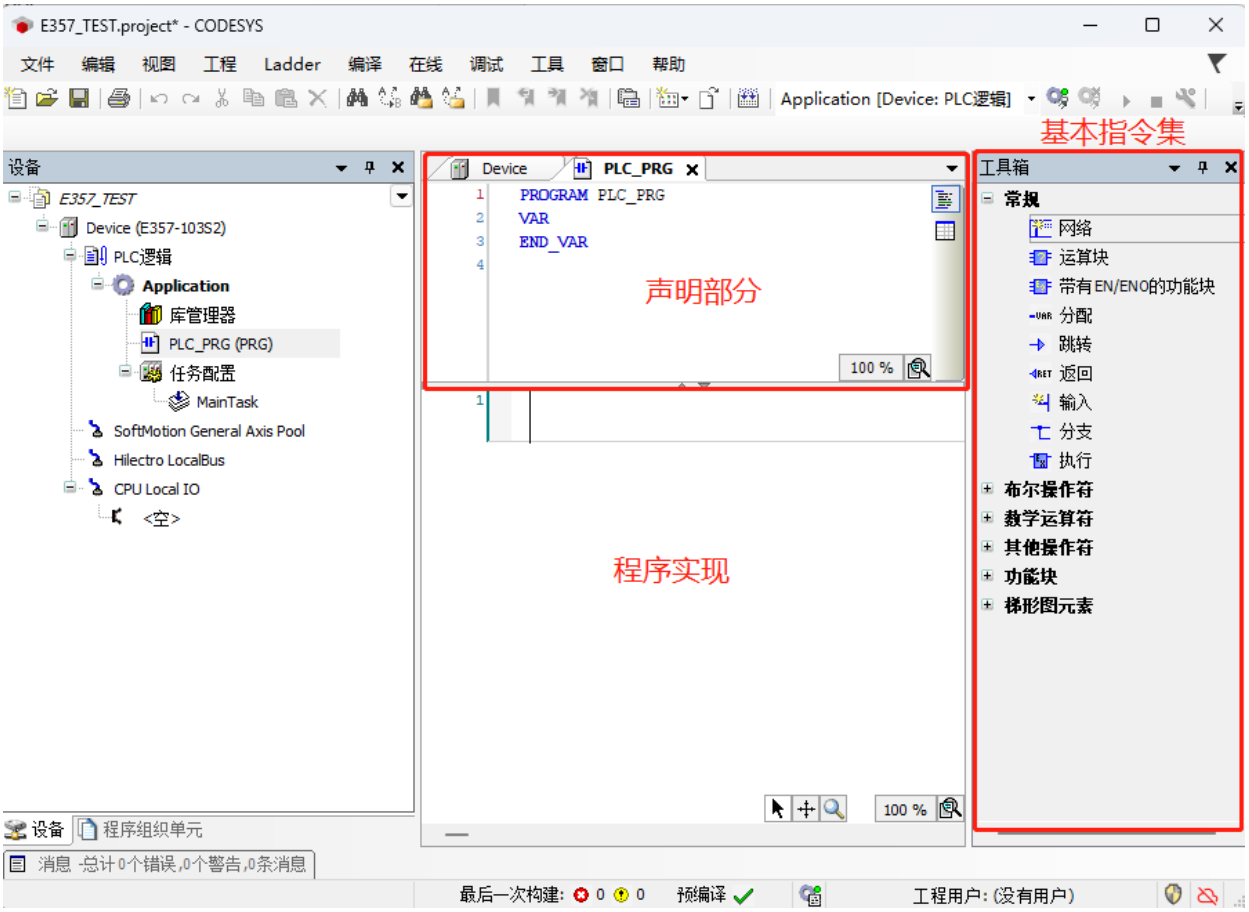


2.4 编写程序

本例程序实现目的：定时器 1 和定时器 2 在 5S 间隔中周而复始的进行置位和复位操作。

在设备视窗中，程序 POU 为“PLC_PRG”，双击设备视图中“PLC_PRG”，自动在 CODESYS 用户界面

中部的 LD 语言编辑器中打开。LD 语言编辑器包含声明部分、实现部分。

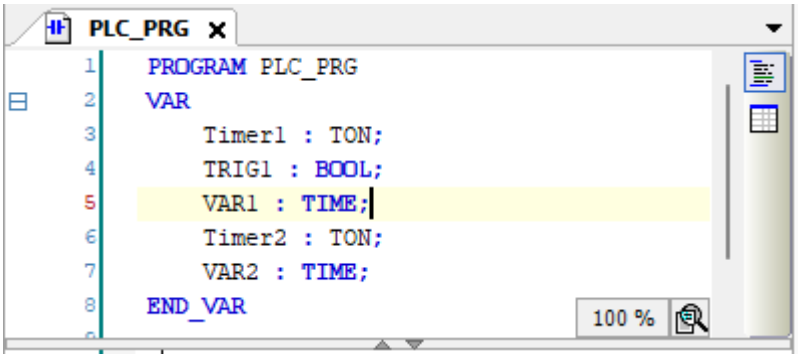


声明部分包括：显示在左侧边框中的行号、POU 类型和名称（如“PROGRAM PLC_PRG”），以及在关键字“VAR”和“END_VAR”之间的变量声明。

实现部分为空，仅显示行号 1。

1、在 PLC_PRG 中声明变量

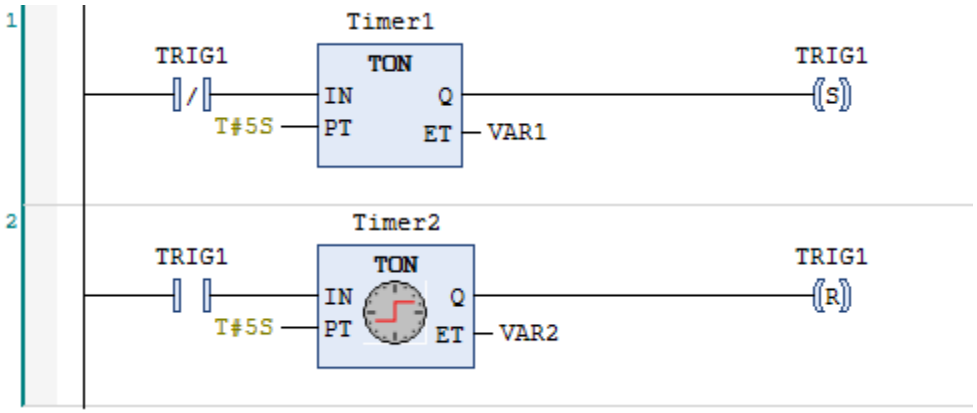
在编辑器的声明部分，将光标移到 VAR 后，点击回车，插入新的空行，声明需要使用的变量。或者在程序的实现部分，使用自动声明功能：在程序的实现部分输入指令，点击回车，如果在新行中有未声明的变量，则系统打开自动声明对话框，在此可以进行声明设置。本例声明部分内容如下图：



2、在 PLC_PRG 的实现部分输入指令

在 LD 语言编辑器右侧的工具箱中展开“梯形图元素”，使用鼠标将“通电延时定时器 TON”拖曳到语言编辑器的实现部分，然后从工具箱的“梯形图元素”中拖曳一个复位线圈到 TON 指令输出点 Q 后方。使用同样的操作，在网络 2 中创建一个 TON 指令。

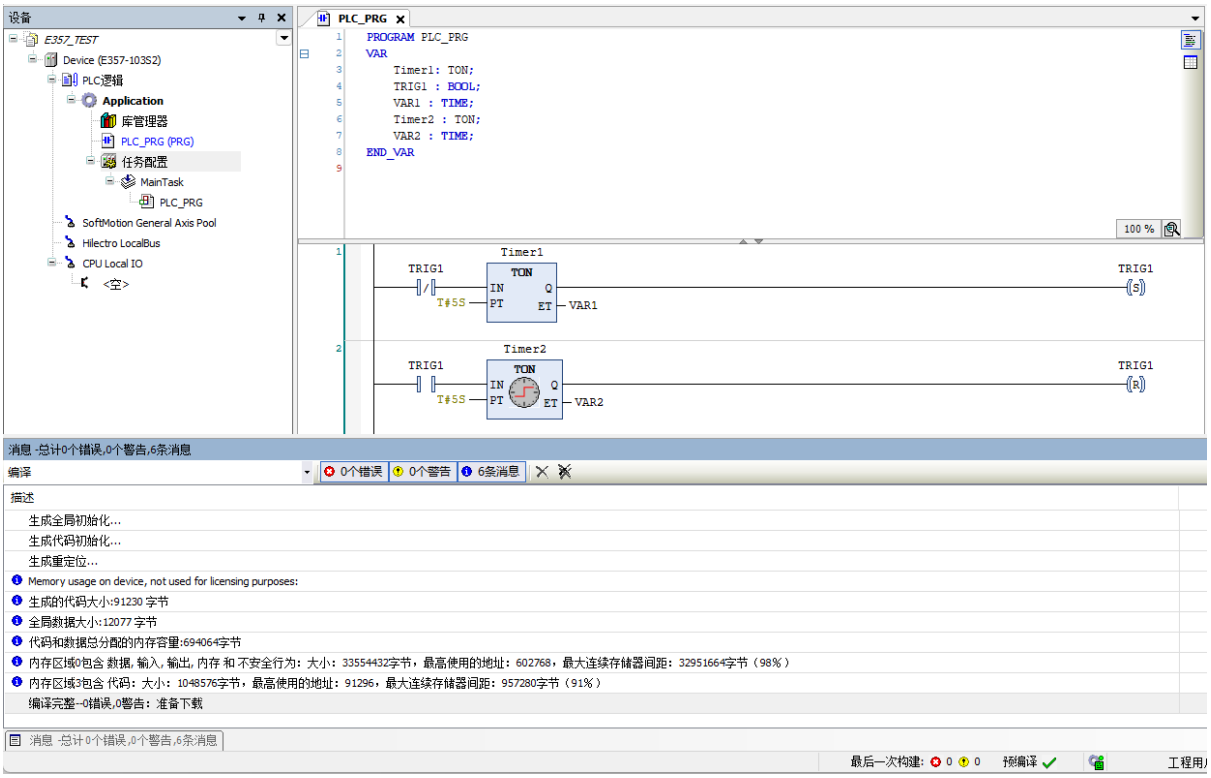
本例最终完成的 LD 程序如下：



2.5 编译和下载

1、保存当前工程并进行编译

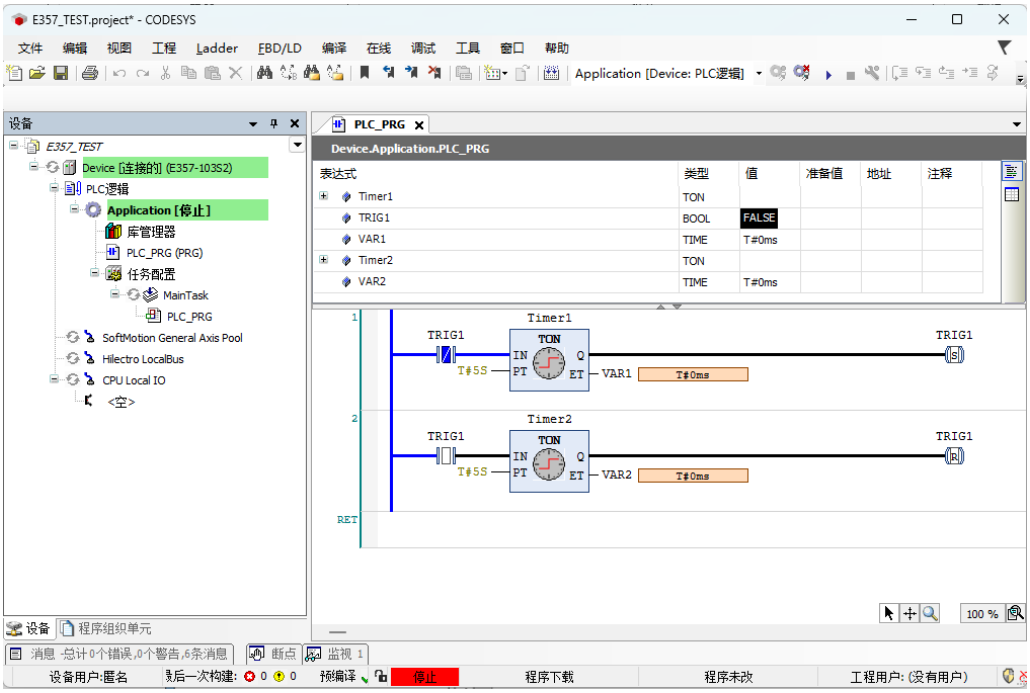
程序编写完成后，保存当前工程，然后选择菜单项“编译”即可对当前对象进行语法检查，语法检查完成时，任何错误消息和警告会在“编译”类的消息窗口中显示出来。若无错误，则表示编译成功。



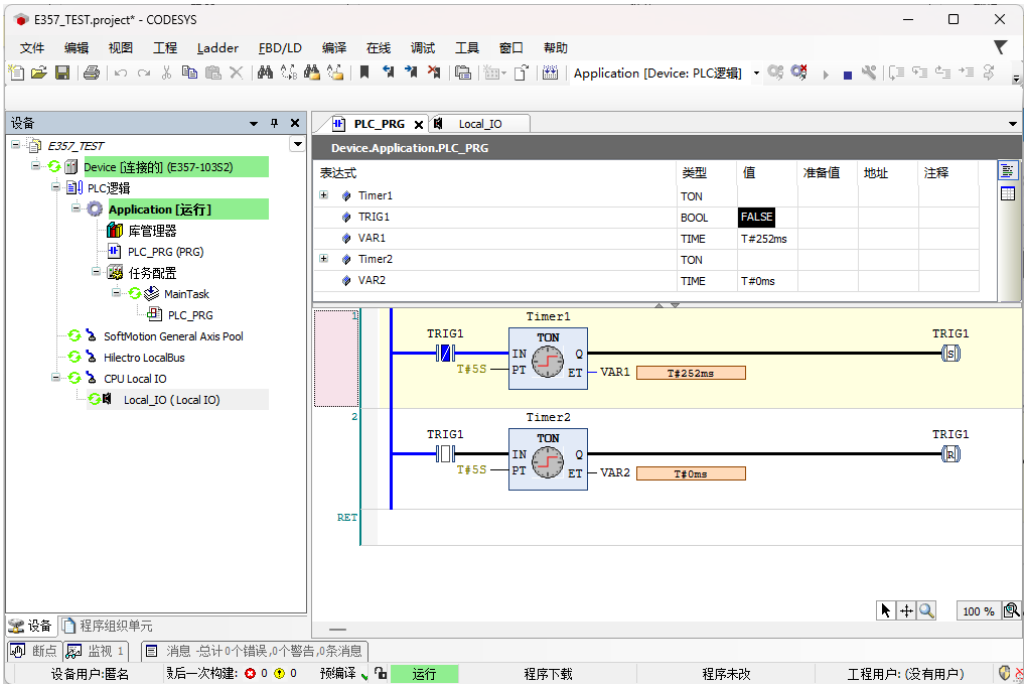
2、登录、下载和启动程序

选择菜单项“在线”→“登录...”或直接点击工具栏 图标，使应用程序与 PLC 建立起连接，并进入在线状态，如果已经进行了通讯设置，会弹出如下对话框提示：

点击“是”启动程序的下载，登录成功后的界面如下：



登录成功后，选择菜单项“调试”→“启动”或直接点击运行图标 ，运行 PLC 程序，此时可对当前工程进行监控和调试。



2.6 监控和调试

使用以下三种方法，可以监控应用程序中的变量和地址：

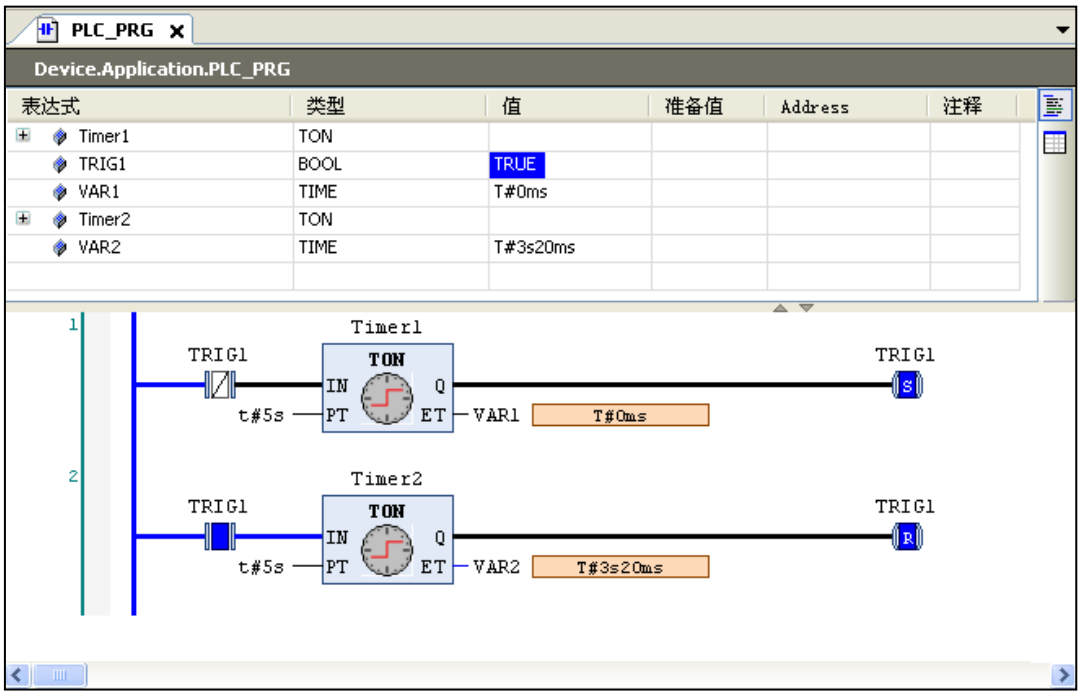
• 包含已定义监视列表的监视窗口

• 写入和强制变量

• 特殊 POU 的在线视图

1、打开程序的示例窗口

双击打开 PLC_PRG，出现如下在线视图：上半部分显示对应于 PLC_PRG 实现部分视图的程序主体，由每一个变量后的内部监视窗口显示实际的值。

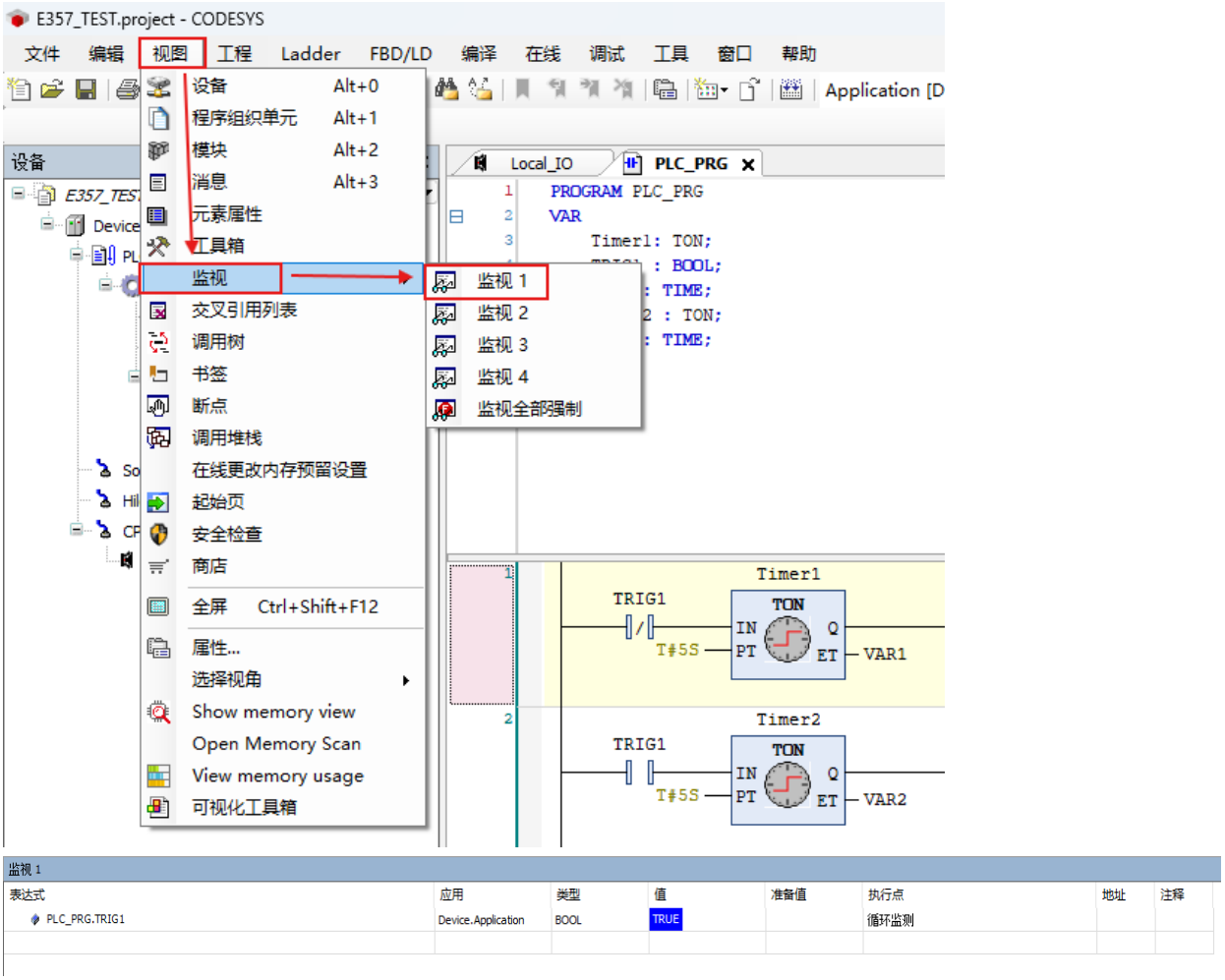


2、写入和强制变量

通过写入或强制的方式将某一“准备值”赋给变量 TRIG1，在下一周期开始，变量将显示为该值。在“准备值”的输入框输入所需的整数值，单击回车或者该区域外部，然后执行命令“写入值”或者“强制值”，即可将该值写入或强制到 PLC。

3、使用监视视窗

选择菜单项“视图”→“监视”→“监视 1”打开监视窗口。然后，鼠标点击表达式列第一行，打开编辑框，输入要监视的变量 TRIG1 的完整路径：“Device.Application.PLC_PRG.TRIG1”，随后即可对变量进行写入和强制值。



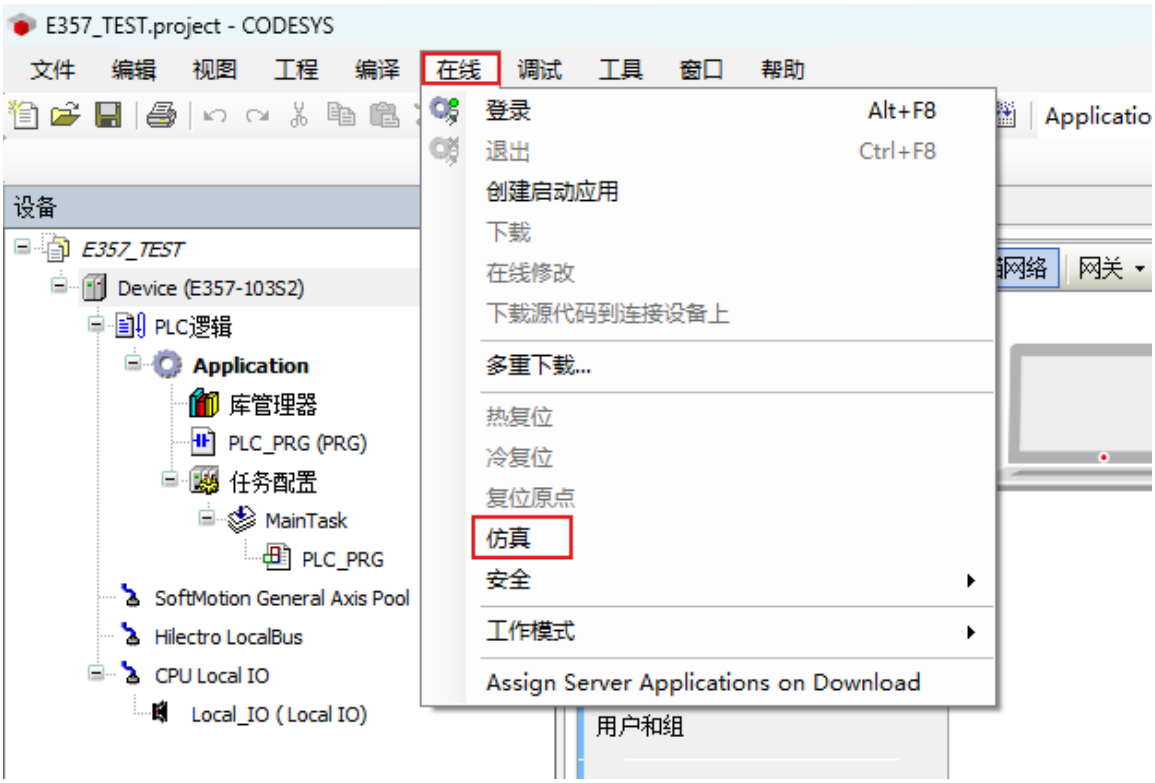
2.7 仿真

CODESYSY SP11 及以上版本支持仿真功能，通过仿真功能，即使在没有连接实际硬件设备的情况下，可检查 PLC 程序的执行，分析程序的逻辑是否正确无误，此外还可以仿真运动功能等等。

具体操作如下：

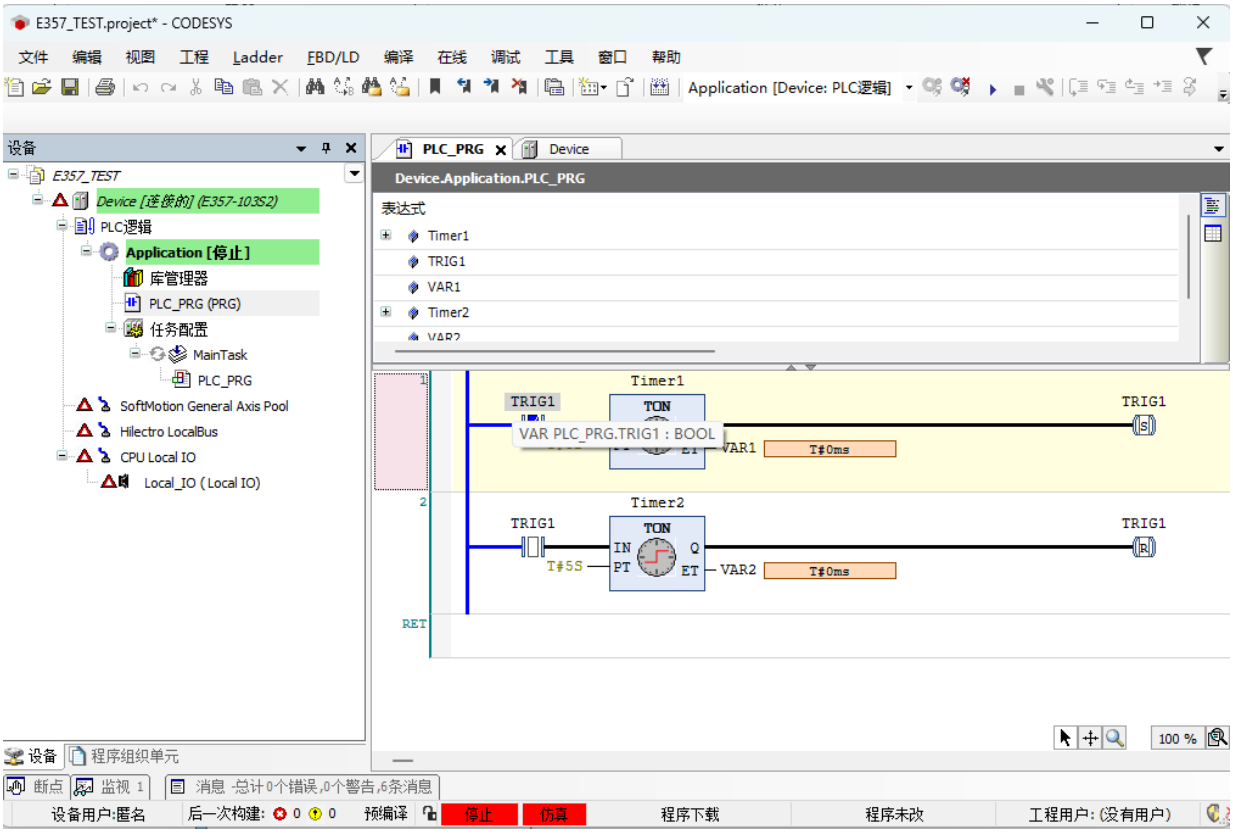
程序编辑完成并进行编译显示无错误后，点击【在线】→【仿真】，下载程序到 PLC 中，在程序界面监控程序可看到程序的执行状况。

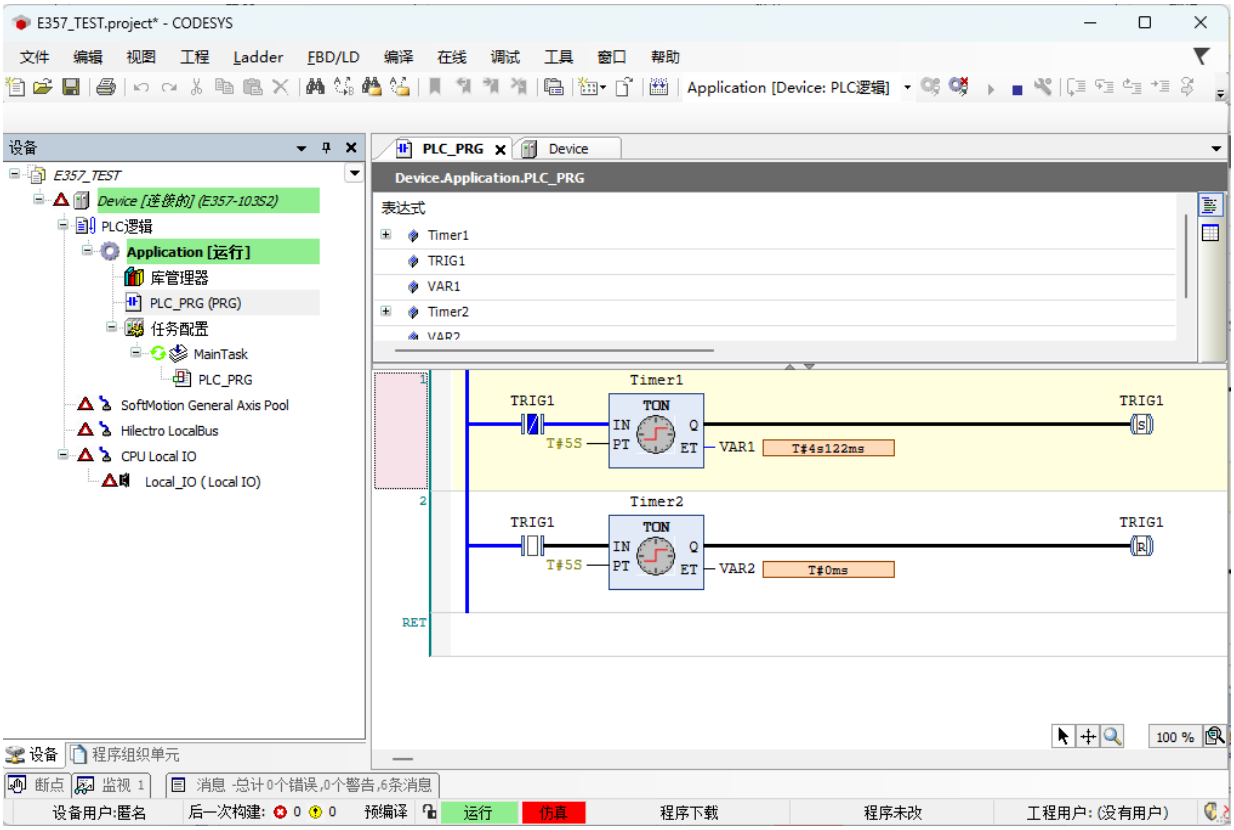
以一个简单程序的执行为例：


E357_TEST.project - CODESYS
文件 编辑 视图 工程 编译 在线 调试 工具 窗口 帮助
登录 Alt+F8
退出 Ctrl+F8
创建启动应用
下载
在线修改
下载源代码到连接设备上
多重下载...
热复位
冷复位
复位原点
仿真
安全
工作模式
Assign Server Applications on Download
设备
E357_TEST
Device (E357-103S2)
PLC逻辑
Application
库管理器
PLC_PRG (PRG)
任务配置
MainTask
PLC_PRG
SoftMotion General Axis Pool
Hilectro LocalBus
CPU Local IO
Local_IO (Local IO)
网络
网关
用户和组


CODESYS
设备'Device'上不存在应用'Sim.Device.Application'，是否创建此应用并进行下载？
是
否(N)
详细信息(D

此时 PLC 已经


E357_TEST.project* - CODESYS
文件 编辑 视图 工程 Ladder FBD/LD 编译 在线 调试 工具 窗口 帮助
Application [Device: PLC逻辑]
设备
E357_TEST
Device [连接的] (E357-103S2)
PLC逻辑
Application [停止]
库管理器
PLC_PRG (PRG)
任务配置
MainTask
PLC_PRG
SoftMotion General Axis Pool
Hilectro LocalBus
CPU Local IO
Local_IO (Local IO)
PLC_PRG
Device
Device.Application.PLC_PRG
表达式
Timer1
TRIG1
VAR1
Timer2
VAR2
Timer1
TON
TRIG1
VAR PLC_PRG.TRIG1 : BOOL
VAR1
T#0ms
Timer2
TON
IN
PT
Q
ET
T#5S
VAR2
T#0ms
RET
100 %
设备
程序组织单元
断点
监视 1
消息 -总计0个错误,0个警告,6条消息
设备用户:匿名
后一次构建:
预编译
停止
仿真
程序下载
程序未改
工程用户: (没有用户)



2.8 复位功能

在调试过程中，如果用户希望对程序进行复位操作，CODESYS 提供了三种复位方式：热复位、冷复位和初始值复位。三种方式可以在“在线”菜单中进行选择，单击执行后会弹出提示对话框进行确认，不同指令执行后会对程序中定义的不同类型的变量的影响见下表。

热复位：属于在线命令，在线模式下有效。热复位后，除了保持型变量(retain 和 persistent 变量）外，其它当前活动的变量都被重新初始化。如果设置了初始值的变量，热复位后变量值为设定的初始值，其它变量都设置为标准初始值（例如：设置为 0）。

冷复位：属于在线命令，只在线模式下有效。跟热复位命令不同的是，冷复位命令不但将普通变量的值设置为当前活动应用程序的初始值，而且将保持型变量（retain 和 persistent 变量）的值也设置为初始值。冷复位发生在程序下载到 PLC 之后，运行之前（冷启动）。一般在总线中断后，可以采取该方式重新启动总线。

初始值复位：将所有变量（包括剩余变量）都复位为其初始化值。擦除控制器上的所有用户文件，将控制器置于“空”状态。

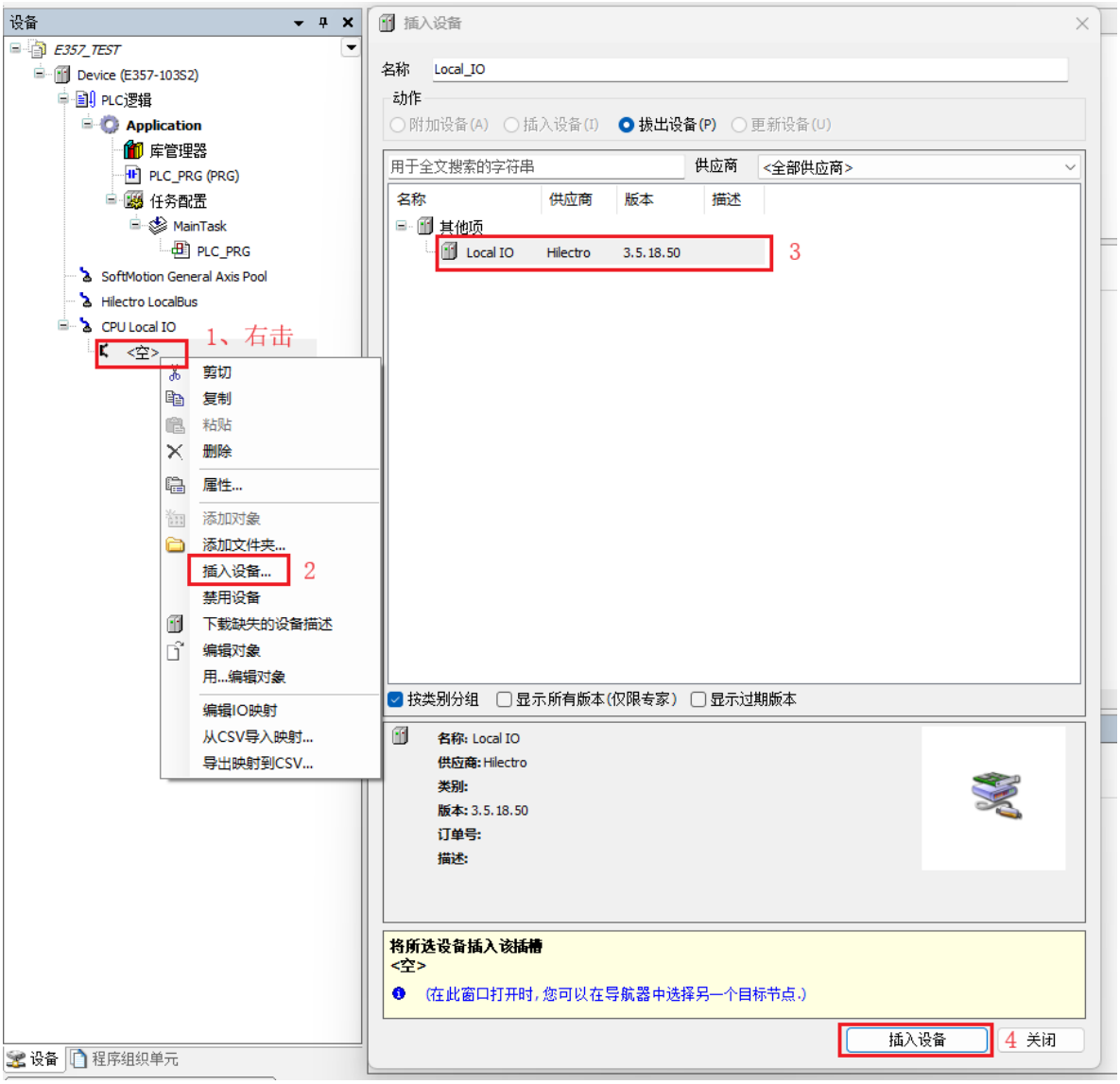
2.9 PLC 本机高速 IO 的使用说明

E357-103S2 本机自带 10 路高速 IO，高速输入输出的端口同样可以作为普通输入输出使用。

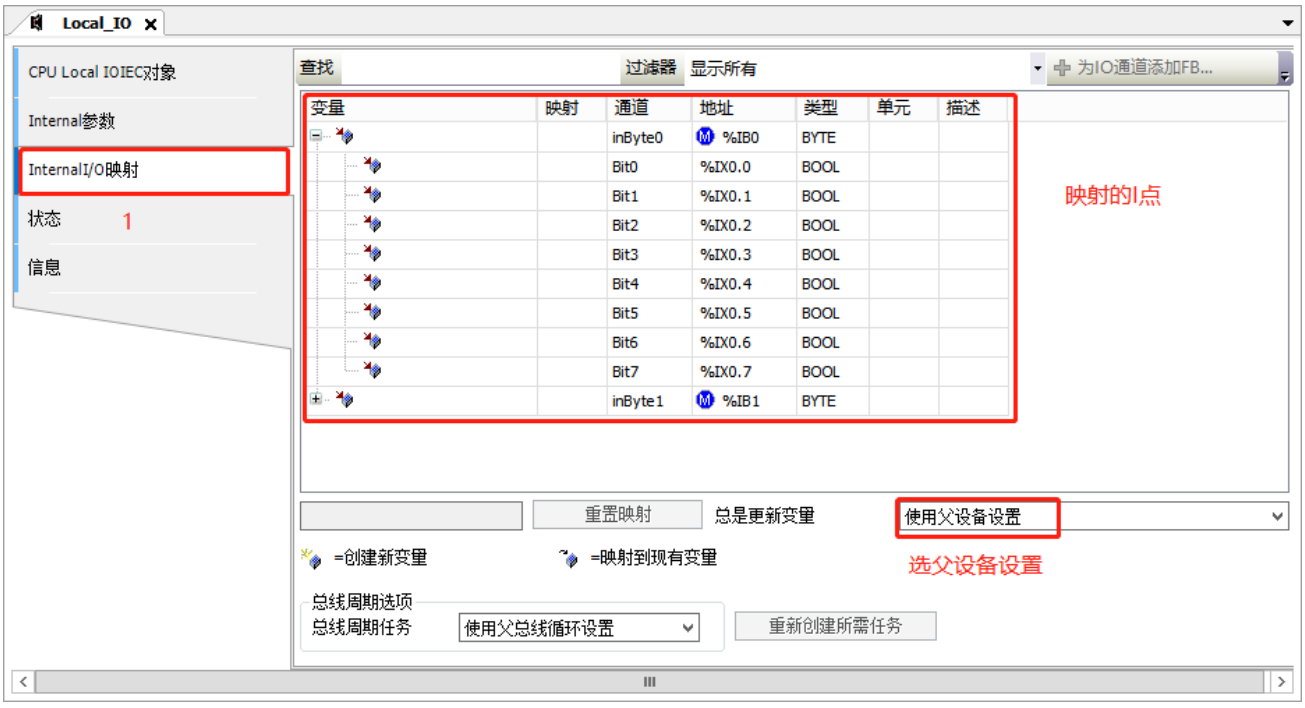
2.9.1 本机 IO 作为普通输入使用

作为普通输入 IO 时的接线与数字量输入模块的接线相同，在软件上的组态步骤如下：

1、首先添加插入设备，设备选择本地 IO。



2、添加完成后在右侧 General I/O 映射下就可以监控当前本地 I 点的状态。



2.9.2 本机 IO 作为高速输入说明

表2-1 高速计数器默认输入点

描述	输入		
HSC0	I0.0	I0.1	I0.2
HSC1	I0.3	I0.4	I0.5
HSC2	I0.6	I0.7	--
HSC3	I1.0	I1.1	--
HSC4	I0.2	--	--
HSC5	I0.5	--	--

表2-2

0	带有内部方向控制的单相计数器	时钟	--	--
1		时钟	--	复位
2		--	--	--
3	带有外部方向控制的单相计数器	时钟	方向	--
4		时钟	方向	复位
5		--	--	--
6	带有增减计数时钟的两相计数器	增时钟	减时钟	--
7		增时钟	减时钟	复位
8		--	--	--
9	A/B 相正交计数器	时钟 A	时钟 B	--
10		时钟 A	时钟 B	复位
11		--	--	--

高速计数器参数

参数	名称	类型	默认值	描述
Module Id	模块 ID	DWORD	16#0a000605	Module Id
channel 0-3 filter	通道 0-3 滤波器	BYTE	6	滤波时间 1：0.2ms；2：0.4ms；3：0.8ms；4：1.6ms；5：3.2ms；6：6.4ms；7：12.8ms；8：0.2us；9：0.4us；10：0.8us；11：1.6us；12：3.2us；13：6.4us；14：12.8us。
channel 4-7 filter	通道 4-7 滤波器	BYTE	6	
channel 8-9 filter	通道 8-9 滤波器	BYTE	6	
HSC0 MODE	HSC0 模式	BYTE	0	bit0~bit3：HSC 模式(0，1，3，4，6，7，9，10) bit4：Z lock disable；0：enable；1：disable

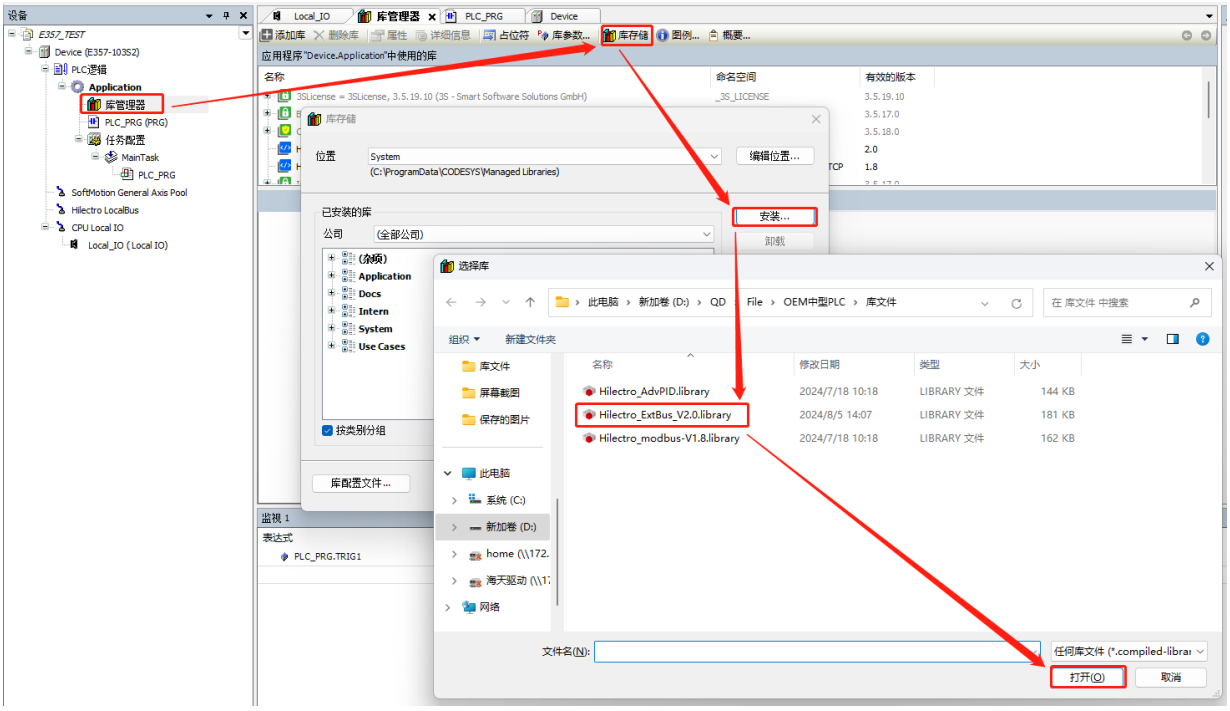
				bit5：Z dear disable；0：enable；1：disable bit6：reserve bit7：clear lock data；0：clear；1：not clear
HSC0 Filter	HSC0 滤波	BYTE	16#2	bit0~bit3：HSC Filter(Hz) 1：750k；2：500k；3：375k；4：250k 5：125k；6：100k；7：75k
HSC0 Current Value	HSC0 当前值	DINT	0	-
HSC0 Preset value	HSC0 预设值	DINT	0	-
HSC0 Speed Test Time(ms)	HSC0 速度测试时间（ms）	BYTE	5	-
HSC0 Ctrl	-	BYTE	16#F9	bit0：Reset Level（0：Low；1：High） bit1~bit2：quad Rate（0：4x；1：2x；2：1x） bit3：Direction（0:Decrease 1：Increase） bit4：Direction Update（0：Not Update；1：Update） bit5：Preset Value Update（0：Not update；1：Update） bit6：Current Value Update（0：Not update 1：Update） bit7：HSC Enable（0：Disable；1：Enable）
HSC1 MODE	HSC1 模式	BYTE	0	bit0~bit3：HSC 模式(0，1，3，4，6，7，9，10) bit4：Z lock disable（0：enable；1：disable） bit5：Z dear disable（0：enable；1：disable） bit6：reserve bit7：clear lock data（0：clear；1：not clear）
HSC1 Filter	HSC1 滤波	BYTE	16#2	bit0~bit3：HSC Filter(Hz) 1：750k；2：500k；3：375k；4：250k 5：125k；6：100k；7：75k
HSC1 Current Value	HSC1 当前值	DINT	0	-
HSC1 Preset value	HSC1 预设值	DINT	0	-
HSC1 Speed Test Time(ms)	HSC1 速度测试时间（ms）	BYTE	5	-
HSC1 Ctrl	-	BYTE	16#F9	bit0：Reset Level（0：Low；1：High） bit1~bit2：Quad Rate（0：4x；1：2x；2：1x） bit3：direction（0：decrease；1：increase） bit4：direction update（0：not update；1：update） bit5：preset value update（0：not update；1：update） bit6：Current Value update（0：not update；1：update） bit7：HSC Enable（0：disable 1：enable）
HSC2 MODE	HSC2 模式	BYTE	0	bit0~bit3：HSC 模式(0，1，3，4，6，7，9，10) bit4：Z lock disable（0：enable；1：disable）

				bit5：Z dear disable（0：enable；1：disable） bit6：reserve bit7：clear lock data（0：clear；1：not clear）
HSC2 Filter	HSC2 滤波	BYTE	16#2	bit0~bit3：HSC Filter(Hz) 1：750k；2：500k；3：375k；4：250k 5：125k；6：100k；7：75k
HSC2 Current Value	HSC2 当前值	DINT	0	-
HSC2 Preset value	HSC2 预设值	DINT	0	-
HSC2 Speed Test Time(ms)	HSC2 速度测试时间（ms）	BYTE	5	-
HSC2 Ctrl	-	BYTE	16#F9	bit0：Reset Level（0：Low；1：High） bit1~bit2：Quad Rate（0：4x；1：2x；2：1x） bit3：direction（0:decrease；1：increase） bit4：direction update（0：not update；1：update） bit5：preset value update（0：not update；1：update） bit6：Current Value update（0：not update；1：update） bit7：HSC Enable（0：disable；1：enable）
HSC3 MODE	HSC3 模式	BYTE	0	bit0~bit3：HSC 模式(0，1，3，4，6，7，9，10) bit4：Z lock disable（0：enable；1：disable） bit5：Z dear disable（0：enable；1：disable） bit6：reserve bit7：clear lock data（0：clear；1：not clear）
HSC3 Filter	HSC3 滤波	BYTE	16#2	bit0~bit3：HSC Filter(Hz) 1：750k；2：500k；3：375k；4：250k 5：125k；6：100k；7：75k
HSC3 Current Value	HSC3 当前值	DINT	0	-
HSC3 Preset value	HSC3 预设值	DINT	0	-
HSC3 Speed Test Time(ms)	HSC3 速度测试时间（ms）	BYTE	5	-
HSC3 Ctrl	-	BYTE	16#F9	bit0：Reset Level（0：Low；1：High） bit1~bit2：Quad Rate（0：4x；1：2x；2：1x） bit3：direction（0：decrease；1：increase） bit4：direction update（0：not update；1：update） bit5：preset value update（0：not update；1：update） bit6：Current Value update（0：not update；1：update） bit7：HSC Enable（0：disable；1：enable）
HSC4 MODE	HSC4 模式	BYTE	0	bit0~bit3：HSC 模式(0，1，3，4，6，7，9，10) bit4：Z lock disable（0：enable；1：disable） bit5：Z dear disable（0：enable；1：disable）

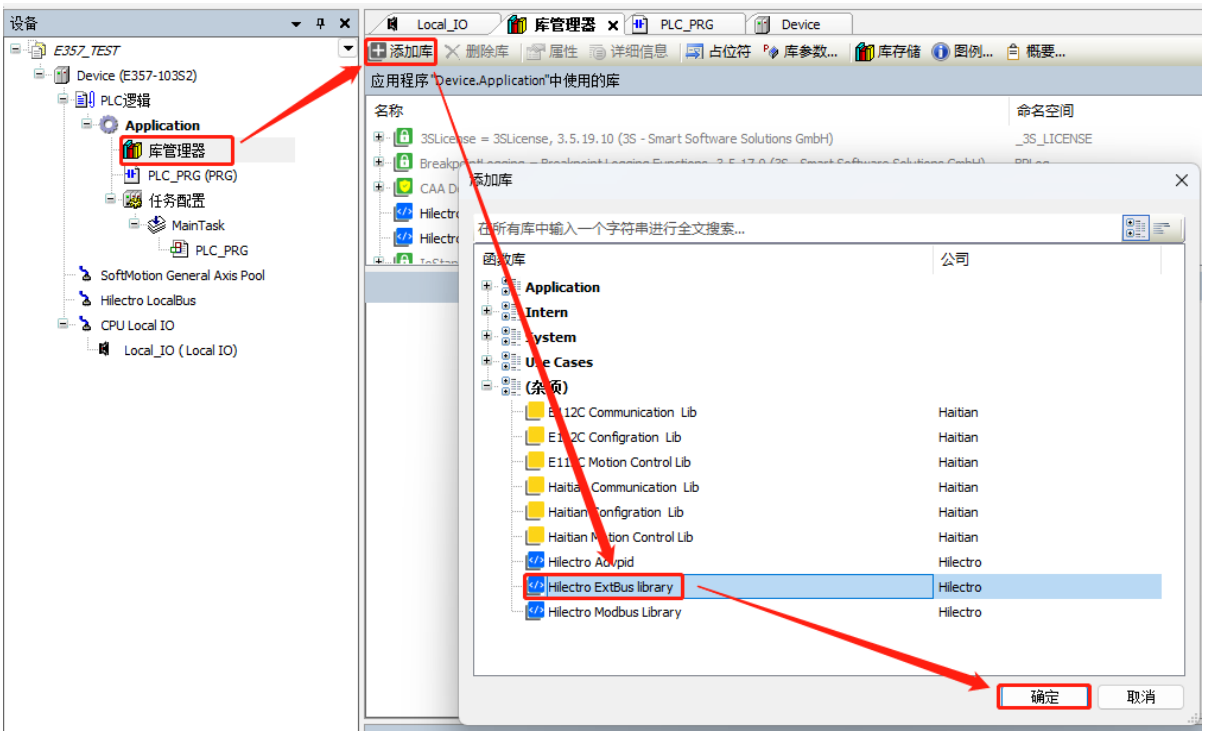
				bit6：reserve bit7：clear lock data（0：clear；1：not clear）
HSC4 Filter	HSC4 滤波	BYTE	16#2	bit0~bit3：HSC Filter(Hz) 1：750k；2：500k；3：375k；4：250k 5：125k；6：100k；7：75k
HSC4 Current Value	HSC4 当前值	DINT	0	-
HSC4 Preset value	HSC4 预设值	DINT	0	-
HSC4 Speed Test Time(ms)	HSC4 速度测试时间（ms）	BYTE	5	-
HSC4 Ctrl	-	BYTE	16#F9	bit0：Reset Level（0：Low；1：High） bit1~bit2：Quad Rate（0：4x；1：2x；2：1x） bit3：direction（0：decrease；1：increase） bit4：direction update（0：not update；1：update） bit5：preset value update（0：not update；1：update） bit6：Current Value update（0：not update；1：update） bit7：HSC Enable（0：disable；1：enable）
HSC5 MODE	HSC5 模式	BYTE	0	bit0~bit3：HSC 模式(0，1，3，4，6，7，9，10) bit4：Z lock disable（0：enable；1：disable bit5：Z dear disable（0：enable；1：disable bit6：reserve bit7：clear lock data（0：clear；1：not clear
HSC5 Filter	HSC5 滤波	BYTE	16#2	bit0~bit3：HSC Filter(Hz) 1：750k；2：500k；3：375k；4：250k 5：125k；6：100k；7：75k
HSC5 Current Value	HSC5 当前值	DINT	0	-
HSC5 Preset value	HSC5 预设值	DINT	0	-
HSC5 Speed Test Time(ms)	HSC5 速度测试时间（ms）	BYTE	5	-
HSC5 Ctrl	-	BYTE	16#F9	bit0：Reset Level（0：Low 1：High） bit1~bit2：Quad Rate（0：4x；1：2x；2：1x） bit3：direction（0:decrease；1：increase） bit4：direction update（0：not update；1：update ） bit5：preset value update（0：not update；1：update） bit6：Current Value update（0：not update；1：update）

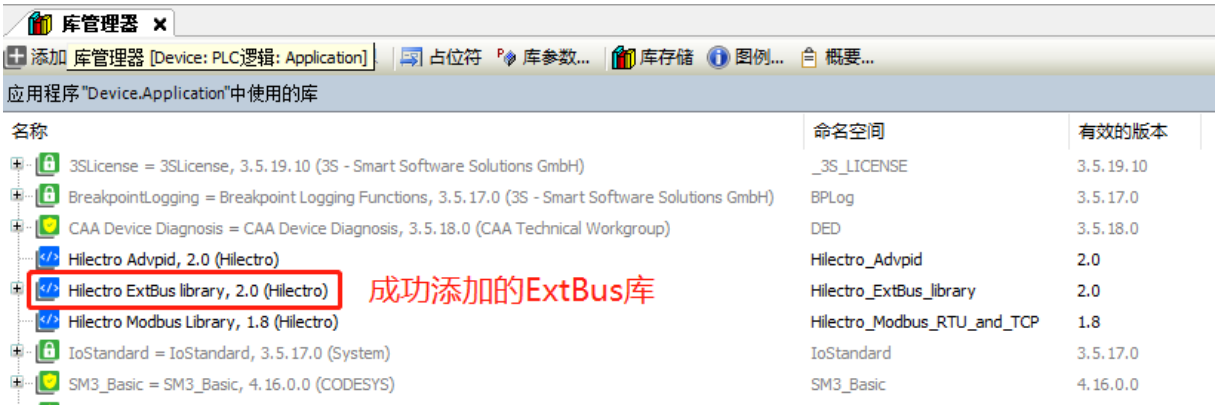
在组态高速输入前，需要先将 Extbus 库添加到 CODESYS 中，然后还需要添加到工程中方便调用该库。

首先将 Extbus 库添加到 CODESYS 中，操作如下：



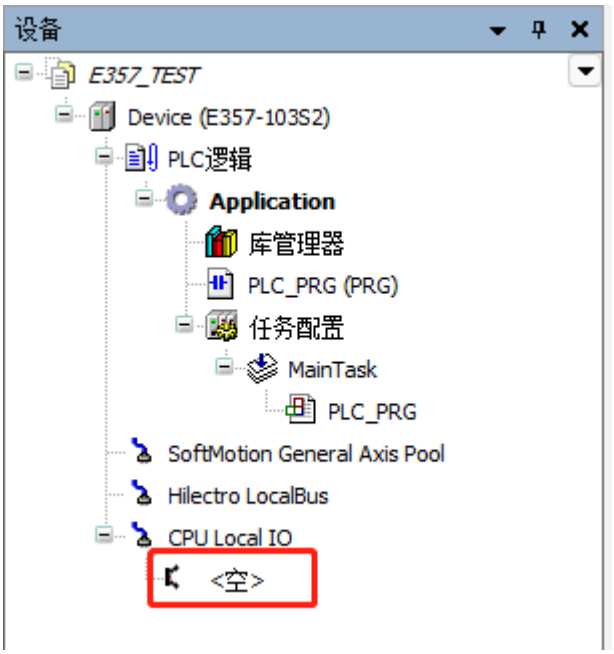
将 Extbus 库添加到工程中，操作如下：

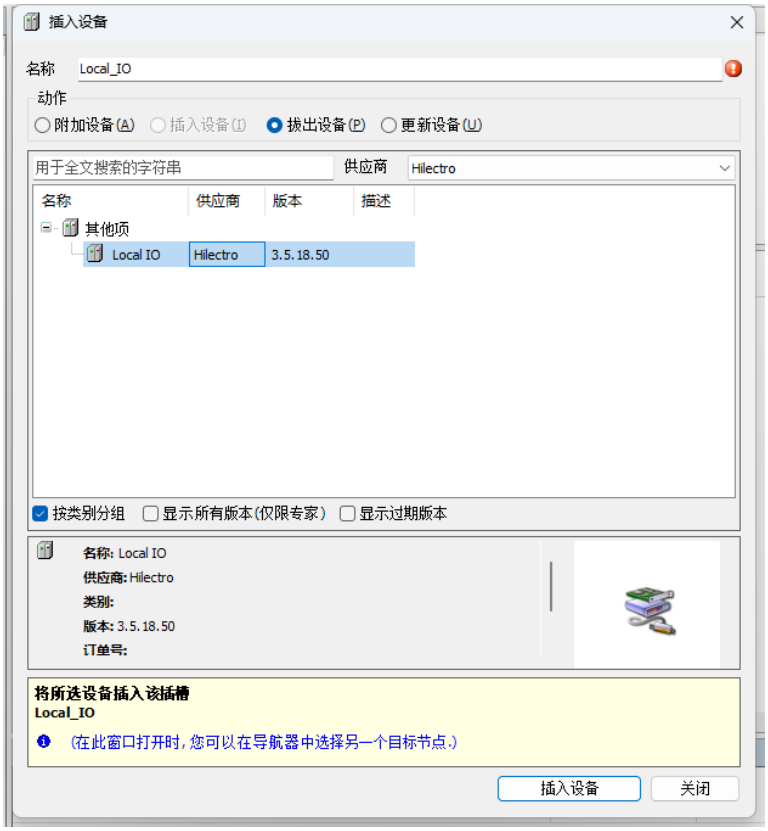




组态工程

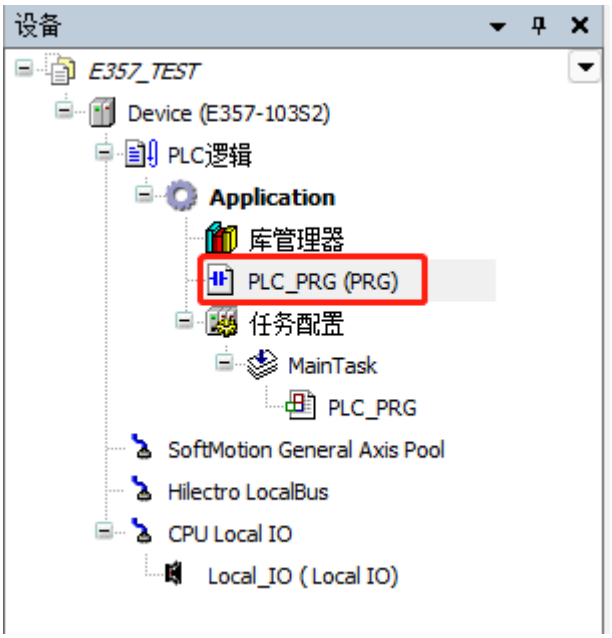
在设备目录中右键点击【CPU Local IO】下方的【<空>】，选择【插入设备】，然后在弹出的对话框中将供应商选择为【Hilectro】，选择【Local IO】，最后点击【插入设备】即可添加该设备，添加成功的设备将显示在 CPU Local IO 下方。



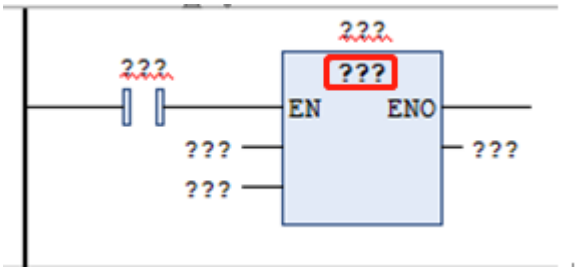


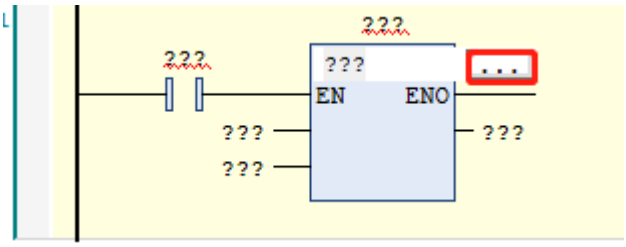
2)使用库文件进行编程

库文件添加成功即可调用高速计数器的指令进行编程，以 HSC_GETCV 指令为例子。首先在 PLC_PRG 中进行编程。

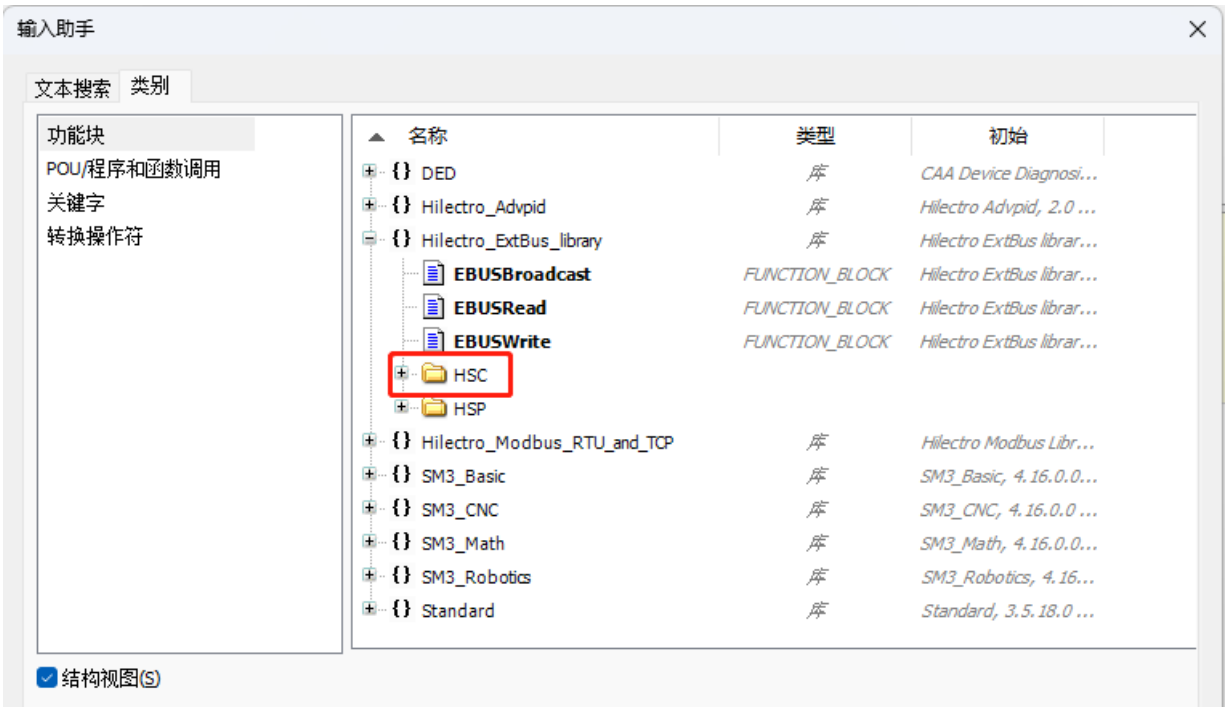


从右边的“工具箱”拖入一个带有 EN/ENO 的功能块，点击功能块方框里的三个问号，出现选项框。

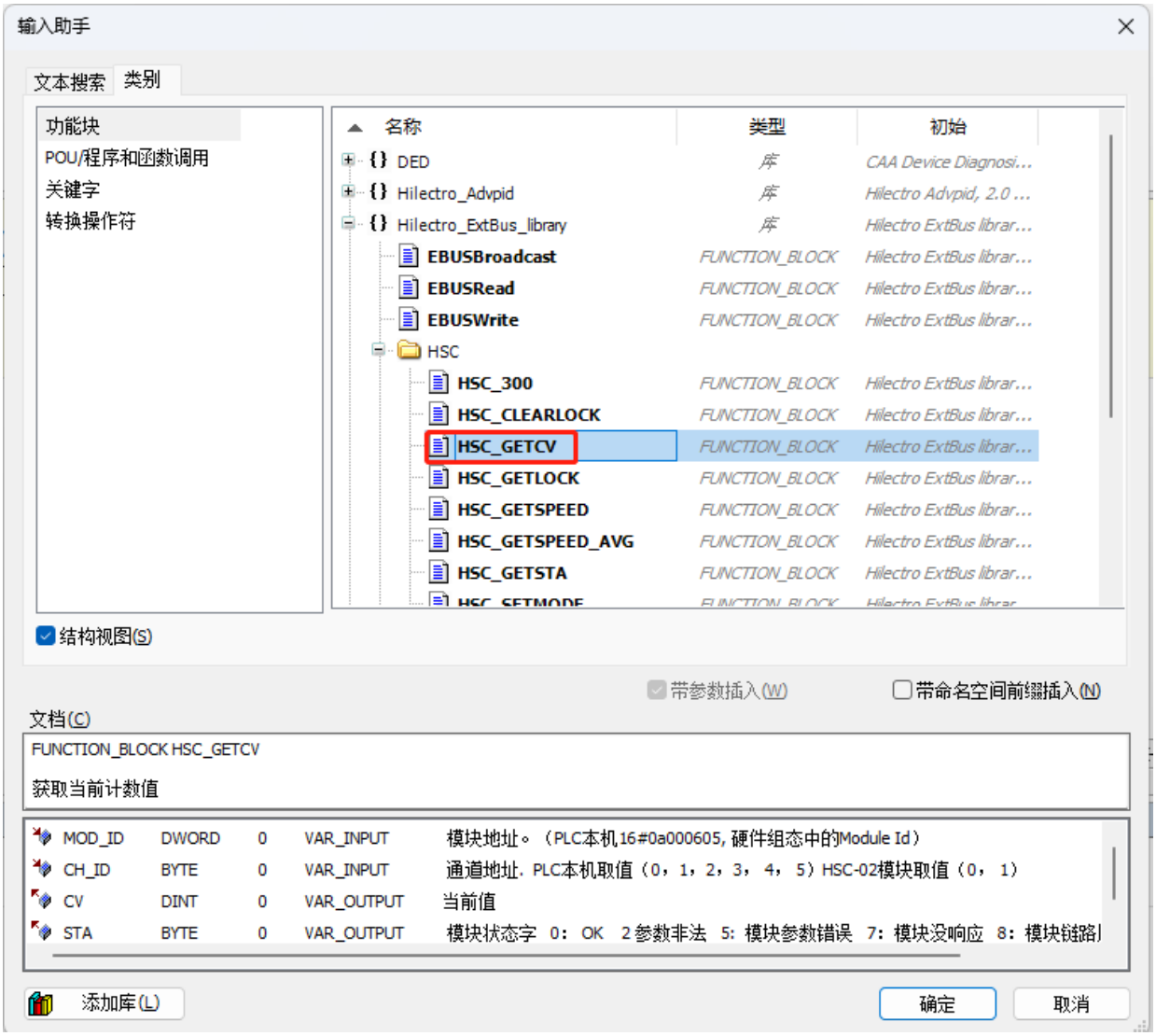




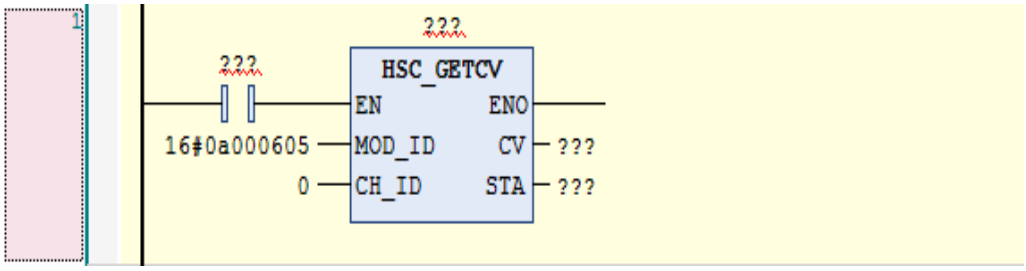
点击选项框，选择【Hilectro _ExtBus_library】→【HSC】即可看到高速计数器相关的指令，点击需要的指令确定即可将指令调入程序中。



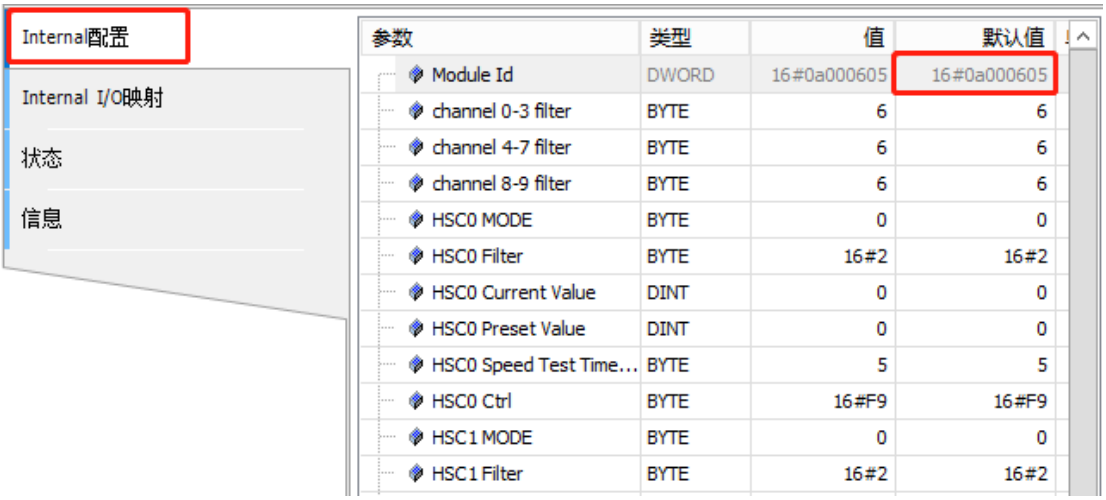
选择 HSC_GETCV 指令。



调用该指令后，指令输入参数 MOD_ID 使用的值为“Internal 配置”中的 MOD_ID 默认值。



点击设备目录的“Local_IO”,选择“Internal 配置”，即可看到 MOD_ID 默认值。



备注：所有的 HSC 相关指令的输入参数 MOD_ID 的值，均使用的是“Local_IO”中“Internal 配置”的 MOD_ID 默认值。

步骤 5：调试与监控程序

1）选择菜单项“在线”→“登录...”使应用程序与 E357 建立起连接，并进入在线状态；然后，选择菜单项“调试”→“启动”使 E357 中的应用程序开始运行。

2）调试程序

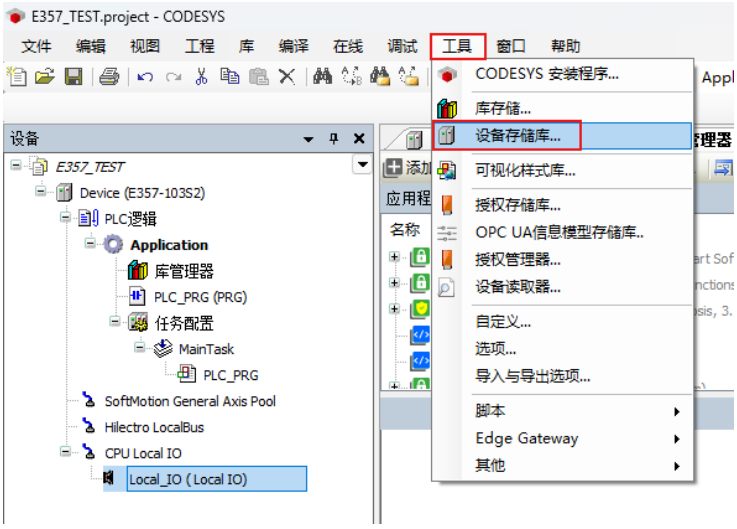
通过程序中的 HSC_GETCV、HSC_GETSPEED 等指令读取伺服电机的当前位置、速度等值。

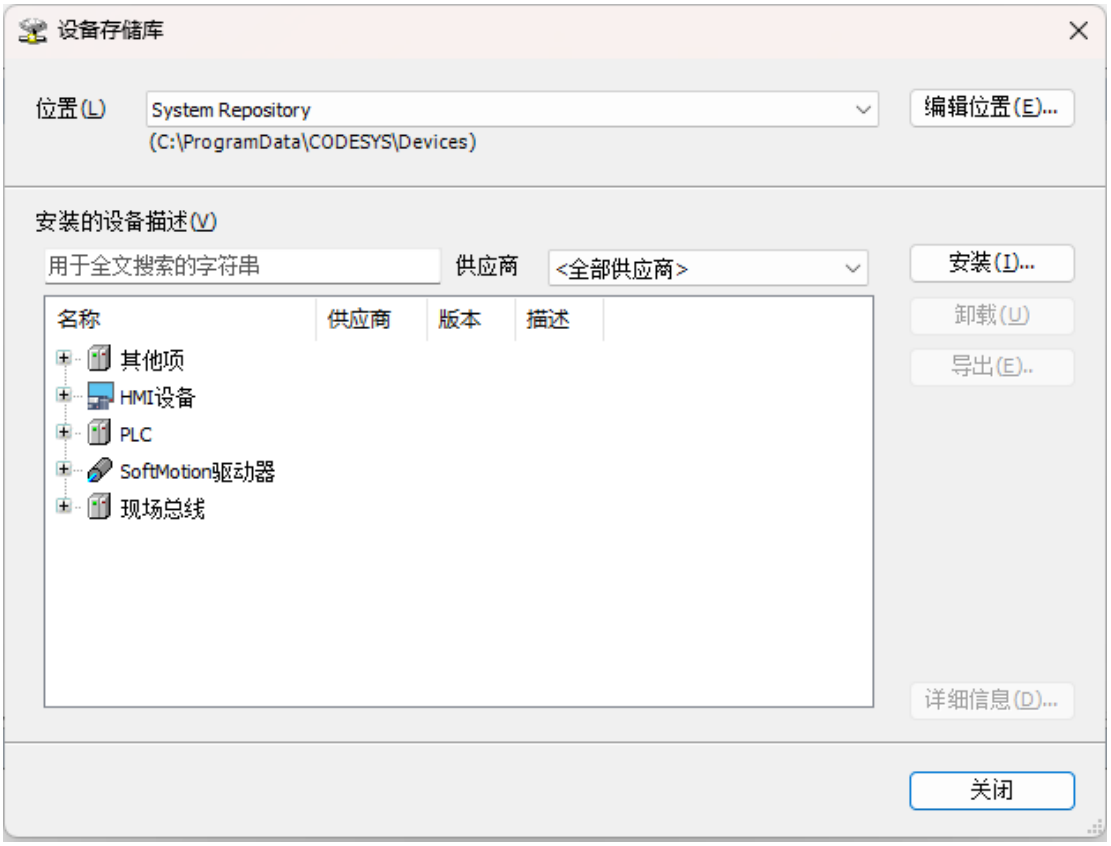
2.10 安装设备描述文件和库

2.10.1 安装设备描述文件

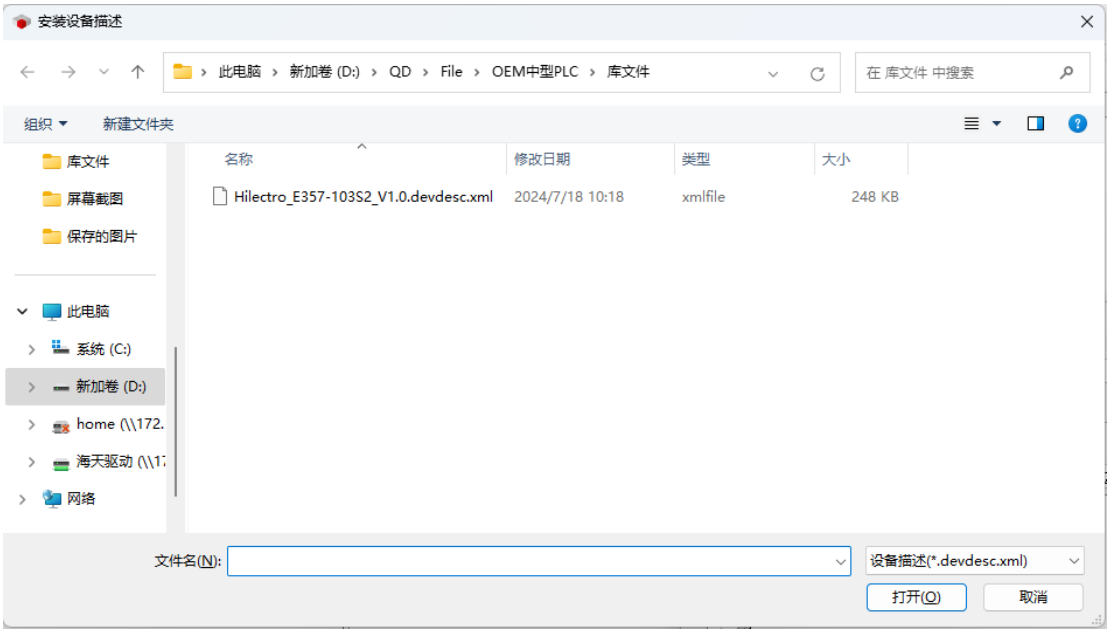
CODESYS 软件本身没有我司的设备，因此想要在 CODESYS 中组态 PLC 设备，必须添加相应的设备描述文件。具体添加步骤如下：

1、选择菜单项【工具】→【设备存储库】，然后点击【安装】。





2、找到描述文件所在的路径后，可根据所需要安装的设备选择文件类型，选择设备后直接安装即可。



设备描述文件可联系我司技术人员获取。

2.10.2 安装库

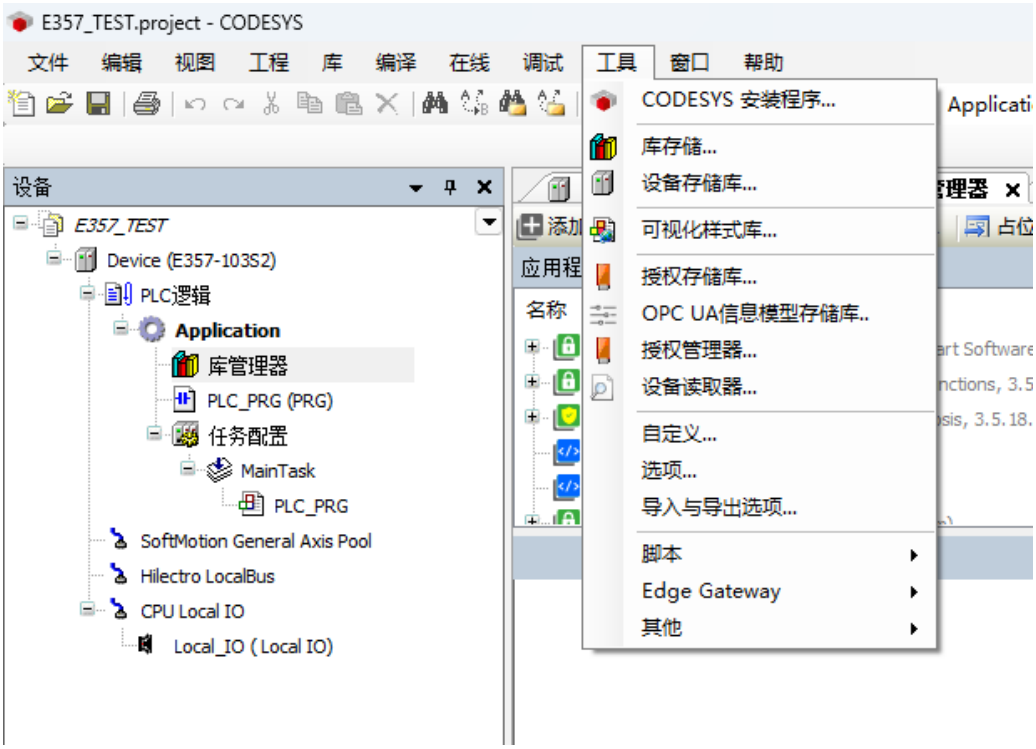
除了系统默认安装并调用的库文件之外，如果用户需要使用外部库或者自定义库文件都需要对库文件进行安装。例如用到我司 E357 自带的 6 路高速计数器时，需要安装 ExtBus 库文件（库文件请联系我司技术人员获取）。

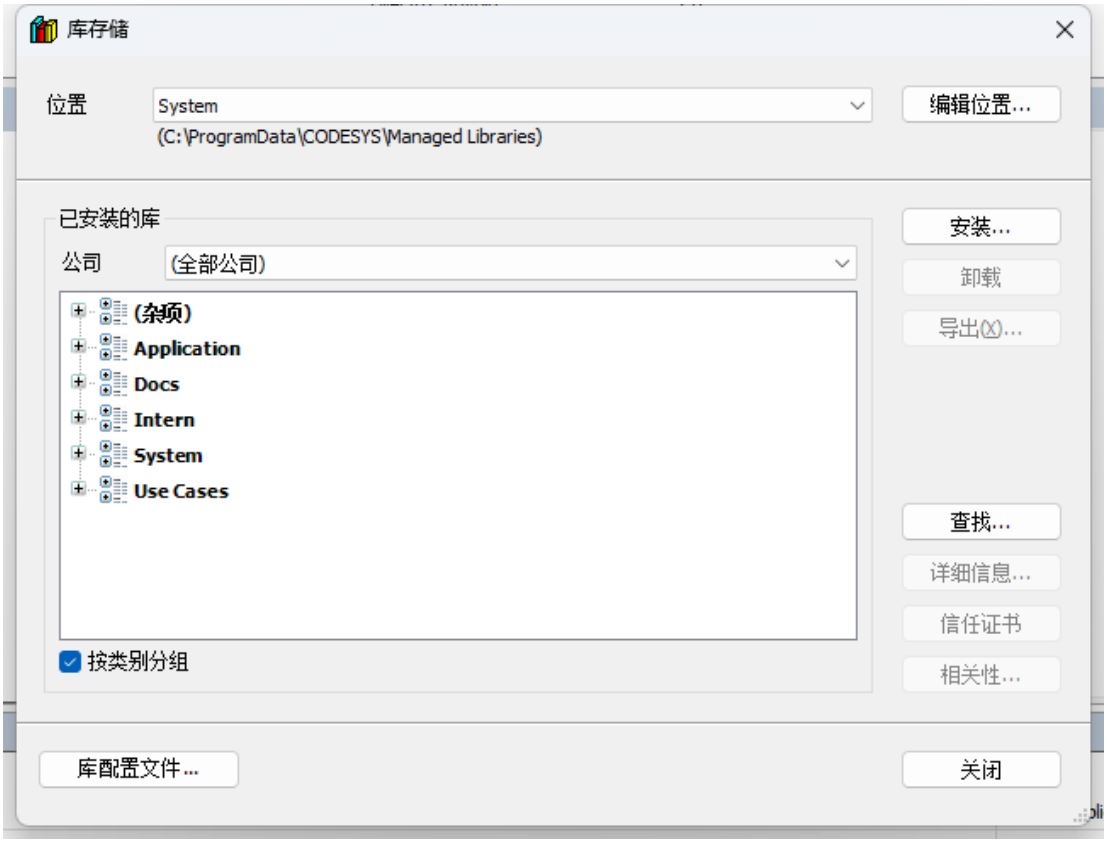
目前可安装我司的库文件有：

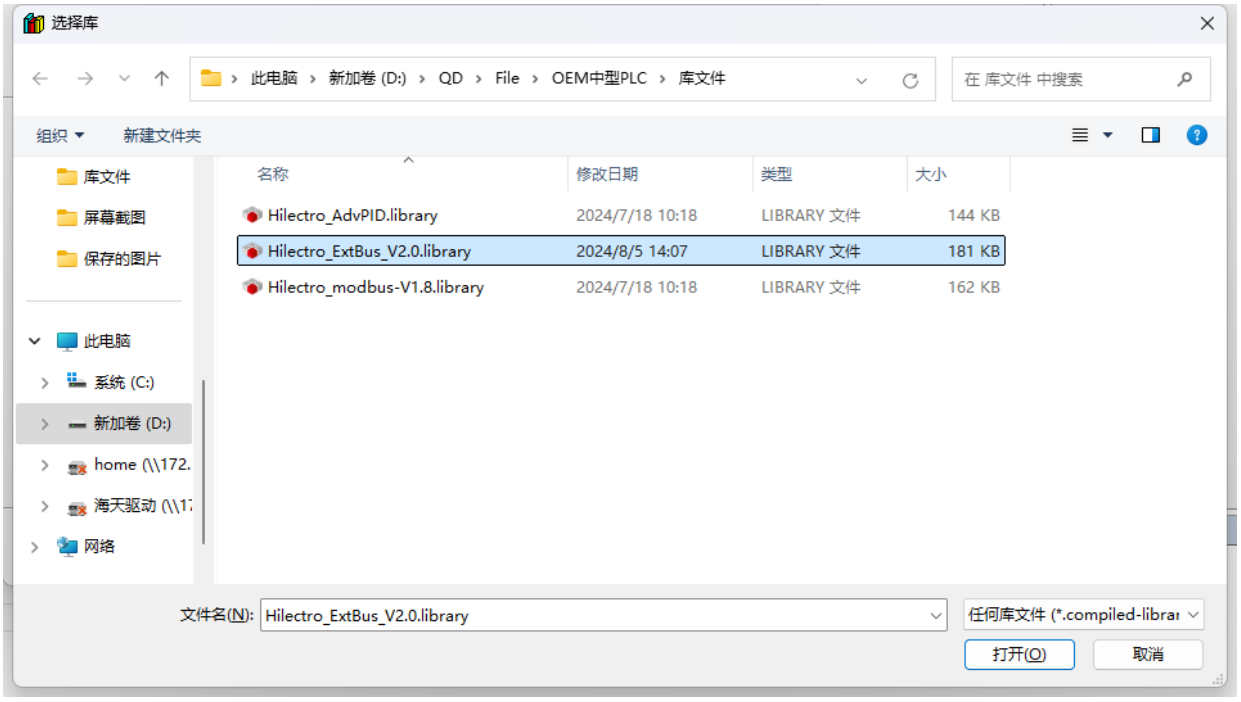
ExtBus 库文件—属于脉冲库，使用 PLC 本机自带的 IO 时用到。

Modbus 库文件—属于通信库，进行 Modbus 通讯使用到该库

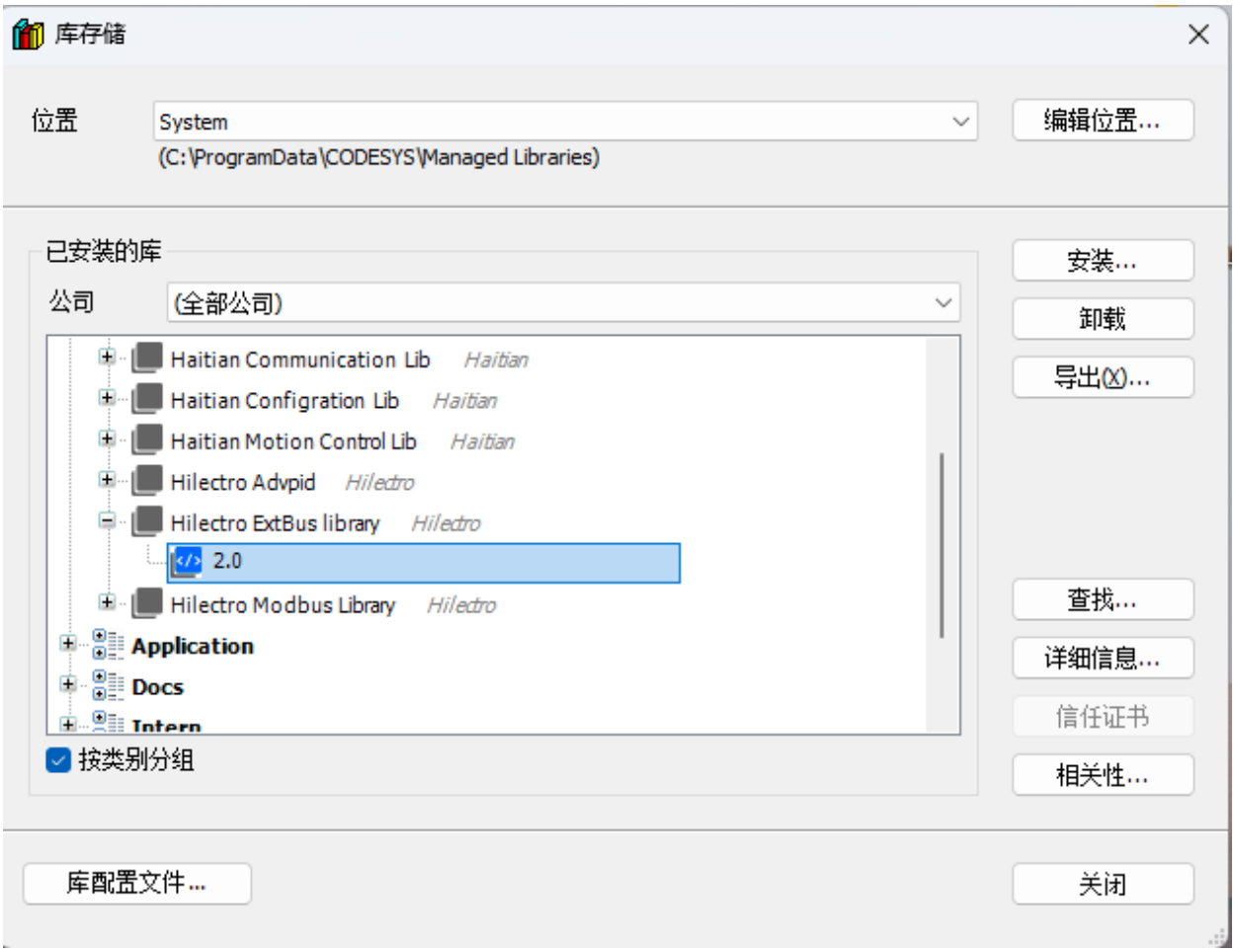
选择菜单栏【工具】下的【库存储】，在弹出对话框右侧选择【安装】，选择需要安装到系统中的库文件后，选择“打开”进行安装。





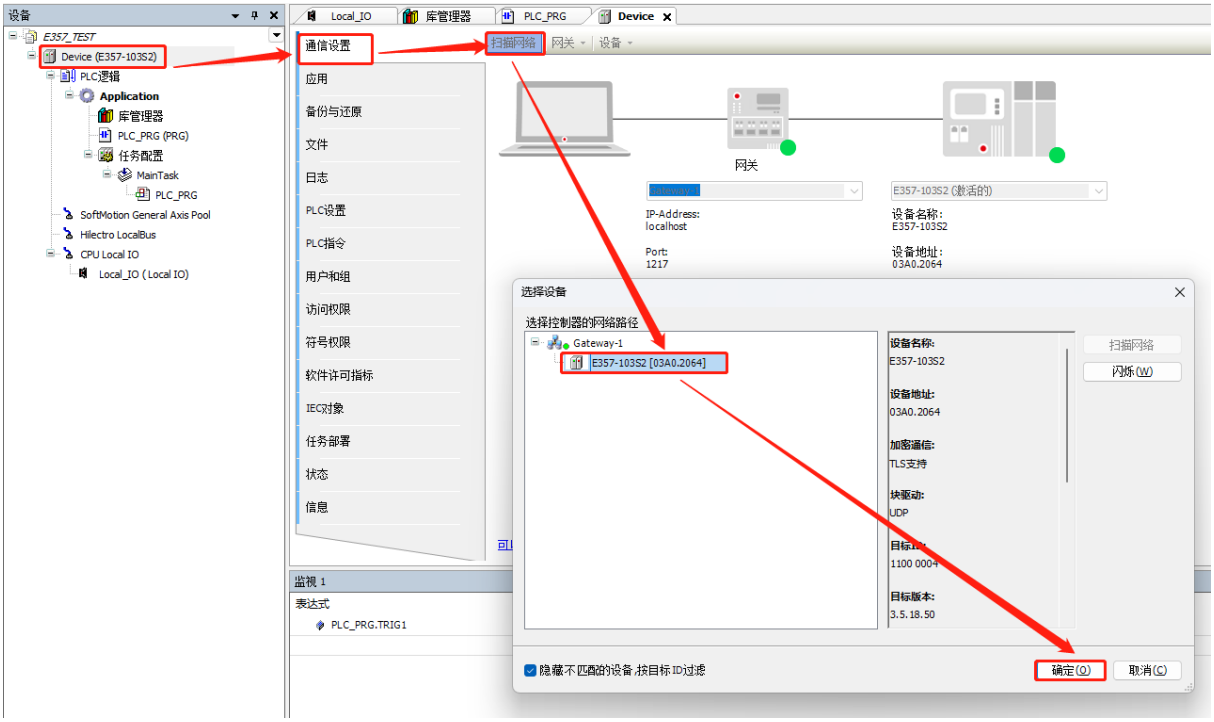


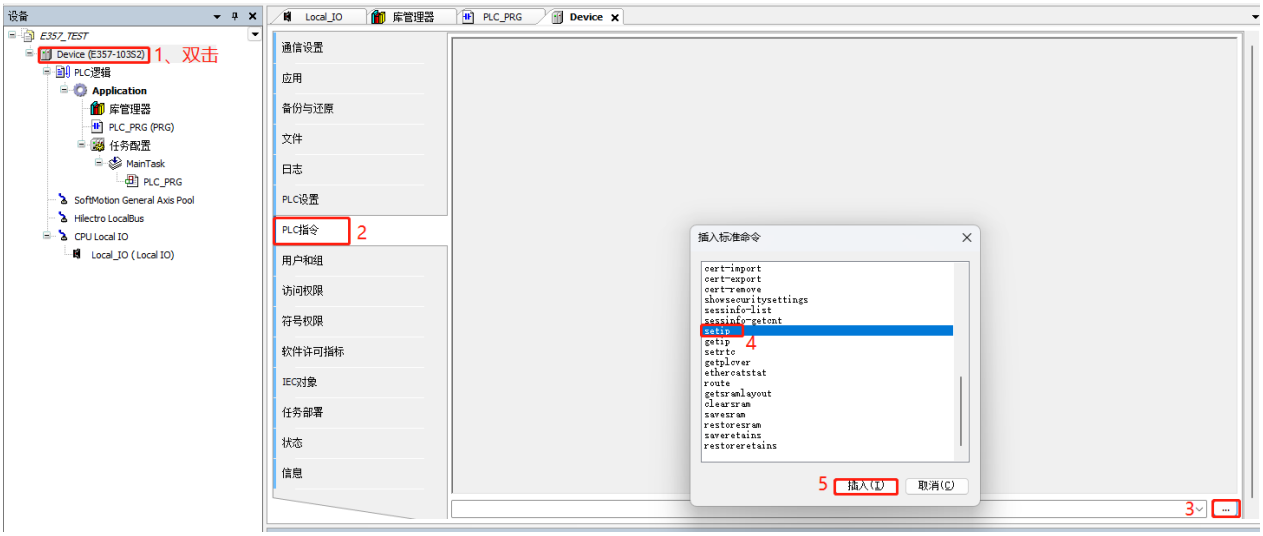
安装完毕可看到已安装库目录下的 ExtBus 库：



2.11 修改 PLC 的 IP 地址

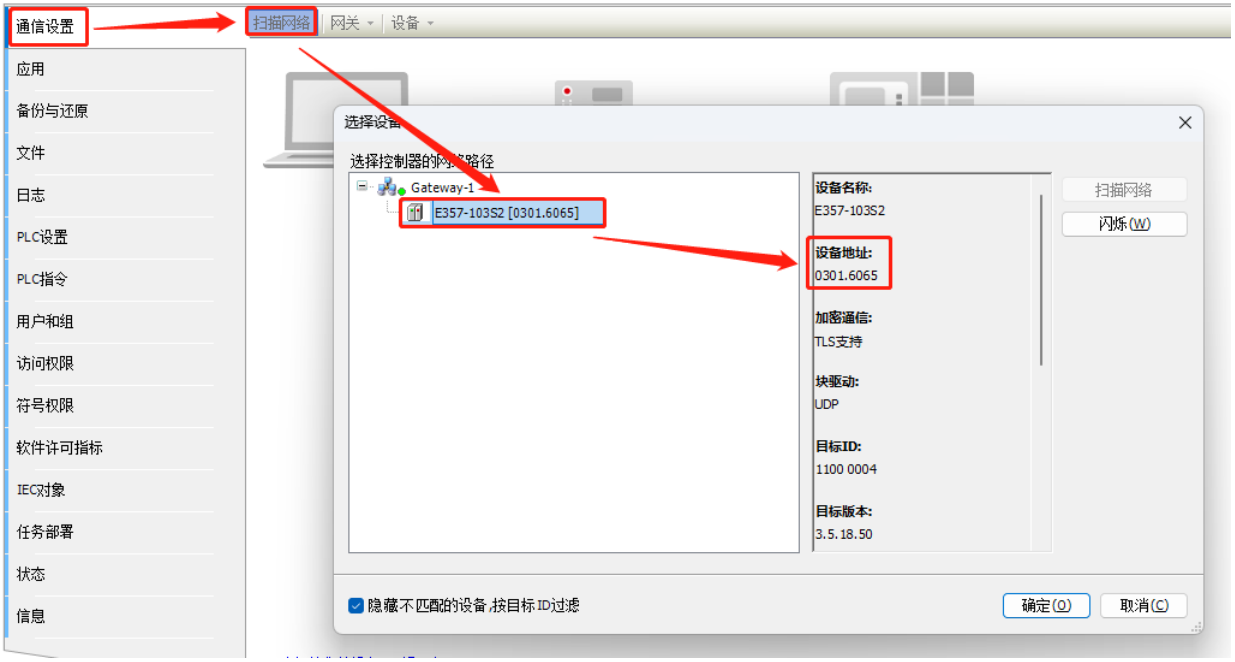
IP 地址需通信成功后，在 PLC 指令中修改，具体操作如下：





也可以忽略前面的 3，4，5 步，直接在输入框里输入 setip eth0 ip:192.168.0.X

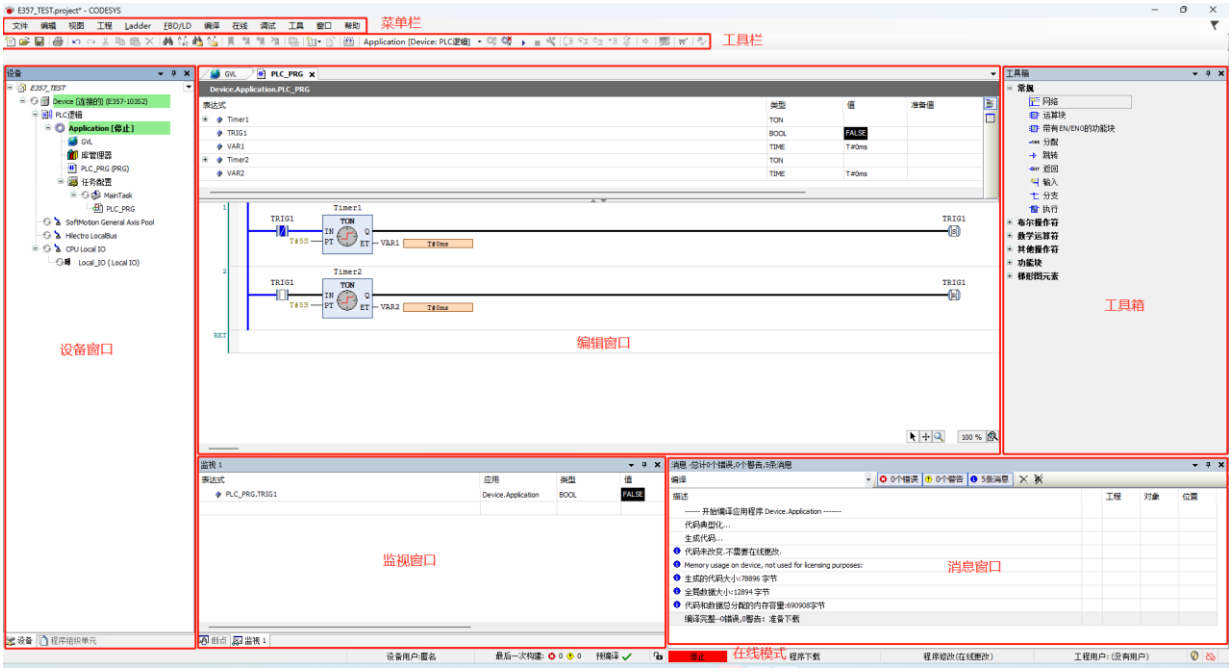
修改 IP 后，可以在通信设置中看到修改成功的 IP 地址。



CODESYS 的结构

3

CODESYS 是一种与设备无关的 PLC 编程系统。CODESYS 不仅支持所有符合 IEC 61131-3 标准的编程语言，还支持 C 语言。与 CODESYS 实时系统结合，可以在一个工程中对多个控制器设备进行配置。CODESYS 编程用户界面由【组件】排列组成，参考下图：



<备注> 本手册仅对 CODESYS 进行简单的应用介绍，有关 CODESYS 的详细使用，请参考 CODESYS 软件的在线帮助。

3.1 工程的构成

一个工程包含了 PLC 程序中的所有对象，以工程文件命名存储。工程中包含下列对象：POU，数据类型，可视化界面，资源及库。

POU

POU 是指【程序组织单元】，可以是程序、功能块或函数类型，它们可以随时增补。

每一个 POU 都包含一个局部和主题声明定义，主体部分可以用 IEC 的语言来编写，这些语言包括指令列表（IL），结构化文本（ST），顺序功能图（SFC），功能模块图（FBD），梯形图（LD）或连续功能图表（CFC）。

CODESYS 支持所有 IEC 标准的 POUs，如果在工程文件中使用这些 POU，必须在工程文件中包含标准库文件 standard.lib。POUs 可以调用其它的 POUs，但递归调用是不允许的。

函数

一个函数是一个 POU，它正确地产生一个数据元素（可以包含若干元素，比如，字段或结构体），在处理过程中，可以作为文本化语言表达式中的一个操作数来调用它。在声明一个函数的时候，一定要给它一个类型，这就是说，在函数名后面加上一个冒号然后跟一个数据类型。

一个正确的函数声明可以参考下面的例子：

FUNCTION Fct: INT

另外，必须分配给函数一个结果，即把函数名作为一个输出变量。

函数的声明从关键字 FUNCTION 开始。

在 IL 中函数的调用被安排在单步操作或单个转换之内。在 ST 中一个函数的调用可以作为表达式中的操作数。

下面是一个在 IL 中一个函数带有三个输入变量并返回前两个变量的乘积与第三个变量相除的结果。

功能块

一个功能块是一个 POU，在程序中提供一个或多个值。与函数相反，一个功能块没有返回值。功能块的声明用关键字 FUNCTION_BLOCK 开始，可以创建功能块的备份或实例。

程序

一个程序是一个 POU，它在操作过程中返回几个值，程序在工程文件中是全局的。程序的所有最终值将保留直到下一次程序运行。

PLC_PRG

PLC_PRG 是一个特殊的预定义的 POU，每一个工程文件中必须包含一个这样的特殊的程序。实际上这个 POU 在每个控制循环中只调用一次。

在一个新工程文件创建之后，将首次使用【Project（工程）】→【Object Add（添加对象）】命令，在 POU 的对话框的缺省项目是一个名为 PLC_PRG 的程序类型的 POU。不能更改这些默认的设置。

如果定义了任务，那么工程中可以不包含 PLC_PRG，因为在这种情况下，程序的时序依赖于任务的分配。

<注意> 不要删除或者重命名 POU PLC_PRG(假如没有使用任务配置)，PLC_PRG 是一个单任务程序中的主程序。

动作

动作能够被定义并分配给功能块和程序，动作表现为一个更进一步的执行，它可以用其它的语言进行创建，每一个动作都有一个名称。

每一个动作都是和功能块或者程序中的数据一起工作的，动作使用相同的输入/输出变量和局部变量来执行。

资源

你需要用资源来配置和组织的工程文件和追踪变量的值。

- 全局变量用于全部工程文件或网络中。
- 库管理器用于添加库文件到工程中。
- 日志文件记录在线期间的动作。
- PLC Configuration 配置可编程控制器的硬件。
- 任务配置通过任务划分来引导程序的工作。
- 监视和接收管理器来显示变量值和设置默认变量值。
- 目标系统设置用来选择和在必要时进行目标系统的最终配置。
- 工作区作为工程选项的映像。

在 CODESYS 中构建工程需要的目标系统和目标系统设置，也可能用到下列资源：

- 用于变量值图形显示的采样追踪。
- 用于在同一个网络中与其它控制器交换数据的参数管理器。
- 作为控制器监视的 PLC 浏览器。
- 工具，与目标系统相关，在 CODESYS 内外调用外部工具程序。

库文件

可以在工程文件中包含一系列的库文件，可以象使用自定义的变量一样使用库文件的 POU，数据类型，和全局变量。库文件中的 standard.lib 和 util.lib 是可以自由调用的程序单元。

数据类型

参照标准的数据类型，用户可以定义自己的数据类型，可以建立结构体枚举类型和引用类型。

可视化界面

CODESYS 提供了可视化界面，因此可以显示工程的变量，通过可视化的帮助可以在离线的情况下绘制几何图形，在联机模式下能够按照特定变量的值而改变他们的形状，颜色和文本输出。

可视化的界面可以用作带 CODESYS 的 HMI 的 PLC 实用操作接口，或者作为一个网页或目标系统显示，通过因特网或 PLC 直接可视化。

3.2 编程语言

CODESYS 提供了对应的编辑器，支持所有在 IEC_61131 标准中规定的编程语言：

1）文本形式的语言：指令表（IL）、结构化文本（ST）

2）图形化的语言：顺序功能图表（SFC）、功能模块图（FBD）、梯形图（LD）、基于功能模块图的连续功能图表（CFC）

3.2.1 指令表（IL）

指令表中包含一系列的指令，依赖于操作的类型，每一条指令在一个新行开始并且包含运算符号和一个或多个用逗号隔开的操作数。在一个指令前面，还可以有一个标号，后缀一个冒号。注释部分在一行的最后，指令与指令之间可以插入空行。

在指令列表中将用到下面的操作符和限定符：

限定符：

C 与操作符 JMP, CAL, RET 连用：当前面的表达式处理的结果为 TRUE 时，才执行此指令。

N 与操作符 JMPC, CALC,RETC 连用：当前面的表达式处理的结果为 FALSE 时，才执行此指令。

N 用于其它情况：取操作数的反（不包括累加器）。

下面是操作符和它们可能的限定符以及相关的意义：

操作符及限定符意义	
LD N	使当前的值等于操作数
ST N	在操作数的位置保存当前值
S	当前的值为 TRUE 时，把布尔型操作数置为 TRUE
R	当前的值为 TRUE 时，把布尔型操作数置为 FALSE
AND N，（	位逻辑运算符号【与】
OR N，（	位逻辑运算符号【或】
XOR N，（	位逻辑运算符号【异或】
ADD（	加法
SUB（	减法
MUL（	乘法
DIV（	除法
GT（	>
GE（	>=
EQ（	=

NE（	<>
LE（	<=
LT（	<
JMP CN	跳转到标号
CAL CN	调用程序功能块
RET CN	离开 POU 并返回到调用的地方
）执行延时操作	

单击这里可以得到所有 IEC 操作符的列表。

例子：使用限定符编写的程序

LD TRUE (* 把 TRUE 加载到累加器中*)

ANDN BOOL1 (* 执行 AND 和 BOOL1 变量的取反后【与】 *)

JMPC mark (* 当上面的结果为 TRUE 时，跳转到标号【mark】处*)

LDN BOOL2 (* 保存 BOOL2 的反 *)

ST ERG (* 把 BOOL2 保存在 ERG*)

Lable:

LD BOOL2 (* 保存 BOOL2 的值 *)

ST ERG (*把 BOOL2 保存在 ERG*)

在 IL 中也可以在操作之后放一个圆括号。圆括号内的值被认为是一个操作数。例如：

LD 2

MUL 2

ADD 3

Erg

这里 Erg 的值为 7，但是如果加一个圆括号：

LD 2

MUL (2

ADD 3

)

ST Erg

Erg 的结果是 10，当到达")"时操作 MUL 才开始计算；此时操作数 5 计算 MUL。

3.2.2 结构化文本（ST）

结构化文本中包含一系列的指令，指令由表达式和关键字组成。ST 语言提供了其他脚本语言一样的功能支持，如 IF...ELSE、CASE、WHILE、FOR 等条件判断与值环执行语句。

表达式

表达式由运算符和操作数组成，操作数可以是常量、变量、函数调用或其它表达式。例如：

常量：11，2023

变量：ivar

函数调用：fct(a,b)

其他表达式：(x*y)/z

表达式的计算

当表达式中含有多个运算符时，需要按照一定的规则计算出表达式的值，优先级最高的运算符首先参加运算，然后是优先级稍高的运算符，直到所有的运算符都被处理为止。相同优先级的运算符号的处理是从左到右的顺序。

下面是结构文本中运算符号优先级的级别排列：

操作	符号	优先级
圆括号	(表达式)	优先级最高
函数调用	函数名（参数列表）	
求幂	EXPT	
取反	-	
互补	NOT	
乘法	*	
除法	/	
取余	MOD	
加法	+	
减法	-	
比较	<，>，<=，>=	
等于	=	
不等于	<>	
布尔运算【与】	AND	
布尔运算【异或】	XOR	
布尔运算【或】	OR	优先级最低

<备注> 以上表格的优先级按表格从上到下的顺序，优先级逐级递减，合并的单元格表示这几个操作符优先级一致。

下面这些是结构化文本中的其它指令，和例子一起安排在一个表中。

指令类型	说明	例子
:=	赋值	A:=B; CV:= CV + 1；C:=SIN(X);
功能块调用	调用一个功能块并使用功能块输出	CMD_TMR(IN:= %IX5, PT:=300); A:=CMD_TMR.Q
RETURN	返回	RETURN;
IF	选择	D:=B*B; IF D<0.0 THEN C:=A; ELSIF D=0.0 THEN

		C:=B; ELSE C:=D; END_IF;
CASE	多重选择	CASE INT1 OF 1: BOOL1:= TRUE; 2: BOOL2:= TRUE; ELSE BOOL1:= FALSE; BOOL2:= FALSE; END_CASE；

FOR	FOR 循环	J:=101;
		FOR I:=1 TO 100 BY 2 DO
		IF ARR[I] = 70 THEN
		J:=I;
		EXIT;
		END_IF;
		END_FOR;

WHILE	WHILE 循环	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT 循环	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
CONTINUE	继续下次循环	CONTINUE;
JMP	跳转	label: i:=i+1; JMP label;
EXIT	退出循环	EXIT;
;	空指令	;

赋值操作符

赋值表达式：将一个数值赋给一个变量或将一个变量的值赋给另一个变量。

赋值形式：变量名=常量或表达式;

赋值符号左边是一个操作数（变量，地址），右边是赋予它的表达式的值，例如：

Var1:=Var2*10

在运算结束后，变量 Var1 就得到了 Var2 的 10 倍值。

调用功能块

通过写功能块的实例名，以及随后在括号中给参数分配值来调用一个功能块。在下面的例子中，通过给两个参数 IN 和 PT 赋值来调用一个定时器，然后结果变量 Q 的值赋予变量 A。

CMD_TMR(IN:= %IX5, PT:=300);

A:=CMD_TMR.Q

RETURN 指令

返回指令可以用来按照条件离开一个 POU（POU）。

IF 指令

IF 指令可以检验一个条件，根据这个条件，执行指令。

语法：

IF <Boolean_expression1> THEN

<IF_instructions>

{ELSIF <Boolean_expression2> THEN

<ELSIF_instructions1>

.

.

ELSIF <Boolean_expression n> THEN

<ELSIF_instructions n-1>

ELSE <ELSE_instructions>}

END_IF;

在{}中的部分是可选的。

如果布尔运算表达式<Boolean expression>返回 TRUE，只有 if 指令部分执行，其它部分不执行。否则，布尔运算表达式从<Boolean expression 2>开始，一个接一个的计算，直到某个布尔表达式返回为 TRUE。然后，在这个布尔运算表达式 2 之后，ELSE 或 ELSE IF 之前的部分被计算。

如果没有任何一个布尔运算表达式返回 TRUE，那么只计算 ELSE 下的指令。例如：

IF temp<17

THEN heating_on:= TRUE;

ELSE heating_on:= FALSE;

END_IF;

这里当温度降到 17 度以下时加热开始，否则保持关闭状态。

CASE 指令

使用 CASE 指令，可以在一个结构中，用同一个条件变量组合多个有条件判断的指令。句式：

CASE <Var1> OF

<Value1>: <Instruction 1>

Languages...

2-14 CoDeSys V2.3

<Value2>: <Instruction 2>

<Value3, Value4, Value5>: <Instruction 3>

<Value6 .. Value10>: <Instruction 4>

...

<Value n>: <Instruction n>

ELSE <ELSE instruction>

END_CASE;

CASE 指令根据下面的模式来处理：

如果变量 Var1 有值 Value1，那么执行指令 Instruction1。

如果变量 Var1 不是所指定的值，那么执行 ELSE Instruction。

如果有多个变量值要执行同一个指令，那么这些条件执行一个公共指令。

如果对于一个变量在一个值的范围内执行同一个指令，那么在初始值和最后值之间用两个句点隔开，所以可以规定公共条件。

例如：

CASE INT1 OF

1, 5: BOOL1:= TRUE;

BOOL3:= FALSE;

2: BOOL2:= FALSE;

BOOL3:= TRUE;

10..20: BOOL1:= TRUE;

BOOL3:= TRUE;

ELSE

BOOL1:= NOT BOOL1;

BOOL2:= BOOL1 OR BOOL2;

END_CASE;

FOR 循环

通过 FOR 循环程序可以编写重复执行的处理过程。

句式：

INT_Var: INT;

FOR <INT_Var>:= <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO

<Instructions>

END_FOR;

{}内的部分是可选的。

只要计数器 INT_Var 不大于 END_VALUE，指令 Instructions 就一直执行，在执行 Instructions 之前首先检查计数器的值，如果 INIT_VALUE 比 END_VALUE 大的话 Instructions 将不在执行。

当 Instructions 执行后，INT_Var 通常要增加一个 Step size，Step size 可以是任何整型值，如果没有 Step size，它将设置为 1，当 INT_Var 大到一定值时，循环结束。

例如：

FOR Counter :=1 TO 5 BY 1 DO

Var1:=Var1*2;

END_FOR;

Erg:=Var1;

我们假设 Var1 的默认值是 1，那么在循环结束后它将得到值 32。

<注意> END_VALUE 一定不要大于等于与计数器 INT_VAR 的极限值，例如：如果变量计数器是一个 SINT 类型，并且 END_VALUE 为 127，那么这将是一个死循环。

WHILE 循环

WHILE 循环可以像 FOR 循环那样使用，不同之处在于 WHILE 循环的退出条件可以是任何布尔型表达式，当条件满足时，就会执行循环。句式：

WHILE <Boolean expression>

<Instructions>

END_WHILE;

只要 Boolean_expression 返回 TRUE，那么就重复执行 Instructions，如果 Boolean_expression 在首次计算出 FALSE，那么指令将不再执行，如果 Boolean_expression 从不出现 FALSE，Instructions 将没完没了的重复执行。

<注意> 程序员必须保证不出现死循环，这可以通过改变循环中指令部分的条件来实现，例如：可以通过计数器增加或减少。

例如：

WHILE counter<>0 DO

Var1:= Var1*2;

Counter:= Counter-1;

END_WHILE

对于 WHILE 和 REPEAT 循环，在循环之前不必知道循环的次数，从这个意义上来说，这两种循环要比 FOR 要强大一些。

因此在此类情况下，可以用这两种循环。如果循环数比较明确，那么因为 FOR 没有死循环反而更好应用。

REPEAT 循环

REPEAT 循环和 WHILE 循环的不同之处在于它的中断条件是在循环执行之后才被检查，这就是说，循环至少要执行一次，不管中断是什么条件。

句式：

REPEAT

<Instructions>

UNTIL <Boolean expression>

END_REPEAT;

Instructions 一直执行到 Boolean expression 返回 TRUE，如果 Boolean expression 第一次就赋予真值，Instructions 只执行一次，否则 Instructions 将重复执行将会导致时间延迟。

<注意> 程序员可以通过改变循环中指令部分的条件来保证没有死循环出现，例如：可以通过计数器增加或减少。

例如：

```
REPEAT
Var1:= Var1*2
Counter:= Counter-1;
UNTIL
Counter=0
END_REPEAT;
```

EXIT 指令

如果在 FOR，WHILE 或 REPEAT 循环中有 EXIT 指令，那么内循环就结束，不管中断是什么条件。

3.2.3 顺序功能图表（SFC）

顺序功能图表是基于图形化的语言，用它可以描述一个程序中不同动作的先后顺序。因为这些动作分配给单步元素，以及采用过度转换变量来控制处理的顺序。

步（Step）

用顺序功能图编写的 POU 包含了一系列的步，这些步之间是通过定向连接（转换条件）实现的。

有两种类型的步：

- 简单类型：每步包括一个动作和一个标记，这个标记用来表示此步是否激活。如果单步动作正在执行，那么在步的右上角方向会出现一个小三角形。
- IEC 类型：每步包含一个标记和一个或多个赋值的动作或布尔变量。相关的动作出现在步的右边。

动作（Action）

一个动作可以包含一系列的指令表或结构化文本指令，功能模块图或梯形图许多的网络，或者又包含另外顺序功能图。

在简单步中，动作经常是和步连接在一起的，为了能编辑一个动作，在步上双击鼠标或选择此步，再选择菜单命令【Extras】【Zoom Action / Transition】。另外，每一个步中只允许一个输入或输出动作。

IEC 步的动作是附加在顺序功能图- POU 内的对象管理器中，通过双击或者在它的编辑器中按 Enter 键可以加载它。也可以通过【Project】【Add Action】来创建一个新的动作。可以为一个 IEC 步分配最多九个动作。

进入和退出动作

可以额外的为一个步添加一个进入和退出的动作，在一个步激活后，一个进入动作只能执行一次。退出动作只在步失效之前执行一次。

带进入动作的步左下角一个【E】来表示，退出动作用右下角的【X】表示。

转换/转换条件

在步和步之间有所谓的转换。

转换条件的值必须是 TRUE 或 FALSE，因而它可以是一个布尔变量、布尔地址或布尔常量。在结构化文本句式（例如（I<=100）AND b）或者在任何一种期望的语言（参照‘Extras’‘Zoom Action/Transition’）中，它也能包括一系列有布尔结果的指令。转换中不能包括程序、功能块或赋值。

<注意> 除了转换外，也能用渐进模式跳到下一步，查看 SFCtip 和 SFCtipmode。

激活步

在调用顺序功能图的 POU 后，将首先执行初始化步的动作（被一个双边线包围）。正在执行的步动作，状态是激活的，在联机模式下，激活的步显示为蓝色。在一个控制循环中激活步的所有动作都将执行。所以，当激活步之后的转换条件是 TRUE 时，它之后的步被激活。当前激活的步将在下个循环中再执行。

<注意> 如果激活的步包含一个输出动作，譬如它下面转换条件是 TRUE，那么它只能在下个循环过程中执行。

IEC 步

在顺序功能图中可以使用标准的 IEC 步。

为了能使用 IEC 步，必须在工程文件中联接 Iesfc.lib 库文件。

一个 IEC 步中不能分配超过九个动作，IEC 的动作不象简单步那样固定地作为输入或输出到某个步的动作，而是和步分开存储并且能够在一个 POU 中重复使用多次。因此，它们必须用命令【Extras Associate action】和单个步联系在一起。

除了动作，布尔变量也能分配给步。

能够使用所谓的限定词来控制激活和未激活的动作和布尔变量。可能有时间延迟，如果一个动作依然激活中，而下一个步已经开始处理了，通过限定词 S（设置），可以取得并发的过程。

随着每一个顺序功能模块的调用，相关联的布尔变量被设置或复位，也就是说，随着每一次调用，这个值将在 TRUE 到 FALSE 之间来回变化。

IEC 步的关联动作在步右边的两长方形中表示，左边的区域包含了限定词，可能带有时间常量，右边的区域包含了动作名和各自的布尔变量名。

限定符

为了关联动作和 IEC 步，用到下面的限定词

N	非存储	动作和步一起激活
R	复位	动作是未激活的
S	设置	动作被激活，再复位前保持激活状态
L	时间限制	动作激活一段时间，最大值和步激活时间一致
D	时间延迟	如果步仍然激活，动作在一定时间后激活，然后只要步是激活的，它就保持激活
P	脉冲	如果步激活，动作只执行一次。
SD	存储和时间延迟	在一定时间之后动作激活并保持激活状态到下一个复位开始。
DS	延迟和保持	只要步仍然激活并且保持到下一个复位开始，那么在一定时间后动作被激活
SL	保持和时间限制	动作激活并保持一段时间

限定符 L、D、SD、DS 和 SL 需要一个 TIME 常量格式的时间值。

<注意> 当一个动作失去激活时，它会再执行一次。这就是说每个动作至少执行两次。

SFC 中的隐含变量

在 SFC 中使用一些隐含声明的变量。

每一个步都有一个标记，标记中存储着步的状态。对于 IEC 步来说，步的标记（激活或未激活）被称为<StepName>.x 或者对一个简单的步来说称为<StepName>。当关联的步激活的时候这个布尔变量值为 TRUE，反之则值为 FALSE。它能够用在 SFC 模块中的每一个动作和转换中。

可以通过查询<ActionName>.x 来查询一个 IEC 步是否激活。

隐含变量<StepName>.t 能够用来查询步激活的时间。

隐含变量也能够被其它程序访问，例如，boolvar1：=sfc1.step1.x; 这里 step1.x 是隐含布尔变量，它代表了 POUsfc1 中的 IEC 步 step1 的状态。

SFC 标志符

标志符用来控制 SFC POU 操作，它在工程运行期间隐含创建。为了能读这些标志符，必须定义合适的全局变量或局部变量。例如，如果在一个 SFC POU 中一个步激活的时间超过了它定义的属性，那么就会设置一个标志符，通过用一个【SFCError】变量可以访问到这个标志符（此时 SFCError 得到真值）。

可以定义下列标志符变量：

SFCEnableLimit:这个变量的类型是布尔型，当它的值为 TRUE 时，这一步的超时将会注册进 SFCError，其它的超时将被忽略。

SFCInit:当这个布尔变量值为 TRUE 时，顺序功能图复位到初始状态，其它的 SFC 标志符也会被复位。初始步保持激活，直到变量值为 TRUE 时，才开始执行。只有当 SFCInit 被重新设置为 FALSE 时，模块才能正常工作。

SFCReset:这个布尔变量，与 SFCInit 很相似，不同之处在于，进一步的处理发生在初始化步的初始化之后，因而，例如 SFCReset 标志符可以在初始化步中被复位到 FALSE。

SFCQuitError:当这个布尔变量得到 TRUE 时，SFC 的执行将会停止，因此，在变量 SFCError 中一个可能超时将复位，当这个变量呈现 FALSE 时，激活步中的所有时间都会复位，先决条件是在 SFC 中预先定义 SFCError，同时注册任何超时设定的标志符。

SFCPause:当这个布尔变量值为 TRUE 时，SFC 图表就停止执行。

SFCError:当在 SFC 图表中有超时，这个变量得到 TRUE。如果在这个之后还有其它的超时发生，除非是这个变量已经复位，否则，这些状态将不会记录。如果想使用其它的时间控制标志符(SFCErrorStep,SFCErrorPOU, SFCQuitError,SFCErrorAnalyzation)的前提条件是定义 SFCError。

SFCTrans:当一个转换被激活时，这个布尔变量得到真值。

SFCErrorStep:这个变量是一个字符串变量，如果 SFCError 中注册了一个超时，这个变量将存储这个超时步的名字。前提条件是在 SFC 中已定义了注册任何超时的标志符 SFCError。

SFCErrorPOU:这个字符串变量包含了发生超时的模块名字。前提条件是在 SFC 中已定义了注册任何超时的标志符 SFCError。

SFCCurrentStep:这个字符串变量存储了那个被激活步的名字，它与时间监控无关。在仿真的情况下，此步存储在外部权限的分支中。如果一个超时发生，超时将不再注册，而且 SFCError 也不会复位。

SFCErrorAnalyzationTable:这是数组型变量 ARRAY [0..n]，它提供一个转换表达式的分析结果。对有影响的每个元素转换注册为 FALSE，以及上一步超时记录，所以要把下列信息写进 ExpressionResult 结构中，写进：名称，地址，注释，当前值。

最大可以容纳 16 个元素，因此，数组的范围从（0~15）。

ExpressionResult 结构和隐含使用的分析模块都是由 AnalyzationNew.lib 库文件提供的，分析模块也能够被其它的不用 SFC 编写的 POU 显式使用。

上一步的超时注册是分析一个转换表达式的先决条件，因此在这里必须有一个时间监视的应用，而且 SFCError 必须在声明窗口被定义。

SFCTip, SFCTipMode:这个布尔变量允许 SFC 的渐进模式。当用在 SFCTipMode=TRUE 来切换它时，如果 SFCTip 设置值为 TRUE 它只可能跳到下一个步，只要 SFCTipMode 是设置为 FALSE 时，它可能跳过转换。

可选分支

在 SFC 中可以定义两个或两个以上的可选择分支。每一个分支的开始和结束必须带有一个转换。可选分支可

以包含并行的分支和其它的选择分支，一个可选分支开始于一个水平线并终止于一个水平线（选择结束），或是一个跳跃。

如果在可选分支开始行前面的步是激活的，每一个可选分支的首次变换将从左到右被计算，最先的转换将从左边转换条件为 TRUE 的开始，然后下面的步被激活。

并行分支

在 SFC 中可以定义两个或两个以上的分支为并行分支。每一个并行分支在开始和结束处必须有一个步。并行分支可以包含可选择的分支或其它并行分支，一个并行分支开始于一个双划线，结束于一个双划线或者一个跳跃，它能提供一个跳跃标识。

如果并行分支的先前步是激活的，并且这个步之后的变换条件值是 TRUE 时，那么并行分支的第一步激活。这些分支彼此并行处理。当所有并行步激活并且这些步之后转换条件为 TRUE 时，那么并行分之末端线后的那一步被激活跳转。

跳转是对在跳转符号下面指明的步名的一个连接。当在不允许创建向上或互相交叉联络的时候，必须使用跳转。

3.2.4 功能模块图（FBD）

功能模块图是一种基于图形的编程语言，它用一串网络来工作，每一个网络包含一个提供算术或逻辑表达式、功能块的调用、跳转或返回指令的结构。

3.2.5 连续功能图表（CFC）

连续功能图表编辑器不像功能模块图那样操作，但是可以自由放置元素，允许使用反馈。

3.2.6 梯形图（LD）

梯形图也是一种基于图形化的编程语言，它接近于电子电路的结构，一方面，梯形图很适合构建逻辑开关，另一方面，它也能创建像 FBD 中的网络图，所以梯形图在控制调用其它 POU 的时候是很有用的。

梯形图包含了一系列的网络，左右两边各有一个垂直的电流线，网络仅限于左右两母线之间的范围内，在中间是由线圈触点和连接线组成的电路图。

每一个网络包含左边的一系列触点，这些触点根据布尔变量值的 TRUE 和 FALSE 来传递从左到右的开和关的状态。每一个触点是一个布尔变量，如变量值为 TRUE，电路从左到右通过连接线就连通。否则右边接收到【关】的值。

触点

在梯形图中的每一个网络图的左边都有触点（触点是用两个并行线| |来表示），它用来表示电路的【开】【关】状态。

这些状态与布尔变量 TRUE 和 FALSE 相一致。布尔变量属于每一个触点。如果变量值为 TRUE，那么状态可以通过连接线从左传到右边。否则，右边接收到的是【断开】。

触点可以并联使用，其中的一个并联分支必须传递【开】状态时，并联分支才能传递【开】。或者触点串联连接，此时，触点必须传递【开】状态时，最后的触点才传递【开】，这些与串并联连接的电路一致。

线圈

在梯形网络图的右边有一些所谓的线圈，它们用（）表示，并且只能通过水平线来连接。线圈传递从左到右的连接状态，并且复制状态到布尔变量中，可以描述入口线的状态为【开】（对应布尔变量的 TRUE）或者【关闭】状态（对应布尔变量的 FALSE）。

触点和线圈也可以取反值（在上例中的触头 SWITCH1 和线圈%QX3.0 是取反值）。如果触点取反值（在触

头符号中用【/】来表示），然后它复制反后的值到相应的布尔型变量中。如果一个触点取反值，仅当相应的布尔变量取到 FALSE 时，电路才能连通。

梯形图中的功能块

可以在网络图中添加功能块和程序，但它们必须具有布尔型值的输入和输出，并且可以像触点那样用在梯形图的左边。

触点的设置/复位

触点可以被定义为设置/复位触点。设置触点在触点符号中用【S】表示，它从不覆盖相应的布尔变量中的 TRUE 值，也就是说，如果变量一旦设置为 TRUE，它将保持状态不变。

复位触点用【R】来表示，它从不覆盖相应布尔变量中 FALSE 值，如果变量已经设置了 FALSE，它将保持这种状态不变。

当使用梯形图时候，可以用触点开关的结果来控制其它的 POU。一方面可以用线圈把结果输出给全局变量，这个全局变量可以在其它的地方使用。也可以通过引入一个带 EN 输入的 POU 来直接在梯形网络图中插入可能的调用。

这些 POU 是完整的正常的操作数、功能、程序或功能块。它们都有一个附加的输入标志符 EN。EN 输入是一个布尔变量，只有当 EN 值为 TRUE 时，带有 EN 输入的 POU 才会被计算。

3.3 结构体

3.4 指针

3.5 联机调试功能

采样追踪

采样追踪允许追踪变量的连续变化的值，它依赖于所谓的触发事件，触发事件是先前定义的布尔变量（触发变量）的上升沿或下降沿。CODESYS 允许对 20 个变量进行追踪，每一个变量可以追踪 500 个值。

调试

CODESYS 的调试功能使得错误很容易被找到。为了调试，运行命令【Project】【Options】，在自动弹出

的创建选项的对话框中选择已经激活的选项。

断点

断点是程序处理过程中停止的位置，因而它可以在程序中的特定位置观察变量值的变化。

断点可以在编辑器中设置，在文本编辑器中断点在行的编号处设置，在连续功能图中和梯形图中是在网络编号处设置，在 CFC 中是在 POU 处设置，在 SFC 中是在步处设置，在功能模块图的实例中不能设置断点。

单步

单步是：

在指令表中：执行程序直到运行 CAL LD 和 JMP 命令。

在结构化文本中：执行下一条指令。

在功能模块图梯形图中：执行下一步网络。

在顺序功能图中：继续目前的动作直到下一个步开始。

通过一步一步的运行可以检查程序中的逻辑错误。

单循环

如果选择了单循环，每一个循环结束，执行也就结束。

联机模式下改变值

在操作过程中，变量可以设置为一个特定的值（write value），或者在每一个循环之后重新定义为特定的值（force value）。在联机模式下可以通过双击变量的值来改变它的值，布尔变量从 TRUE 变为 FALSE， 或从 FALSE 变为 TRUE。对于每一种类型的变量都可以打开 Write Variable xy 对话框，在这里可以编辑变量的真实值。

监视

在联机模式下，所有的变量可以从控制器中读出并及时的显示。可以在定义和程序编辑器中找到这些显示。也可以在观察和接收器中读出变量的当前值，并且可以看到它们。如果要监视功能块的实例中的变量，相应的实例块必须已经打开。

仿真

在仿真模拟过程中，创建的 PLC 程序不在实际的 PLC 中运行，而是在 CODESYS 系统中的计算器中运行。所有的联机功能都是可用的。它允许在无需 PLC 硬件的情况下检测逻辑的正确性。

<注意> 外部库文件的 POU 是不能运行在仿真模式的。

日志

日志记录着用户的操作、内部进程、状态变换和联机模式处理过程中发生的意外的情形。它用来监视和跟踪错误。

3.6 标准化

IEC61131-3 是一种国际标准化的 PLC 编程语言。在 CODESYS 中提供的可编程语言符合标准化的要求。根据这个标准，一个程序包含以下元素：

- 结构
- POU

- 全局变量

通用的语言的元素在标识符、地址、类型、注释和常量部分中已经讲述。CODESYS 程序的处理是从一个特殊的 POU（POU）PLC_PRG 开始的，POU PLC_PRG 能调用其它的 POU。

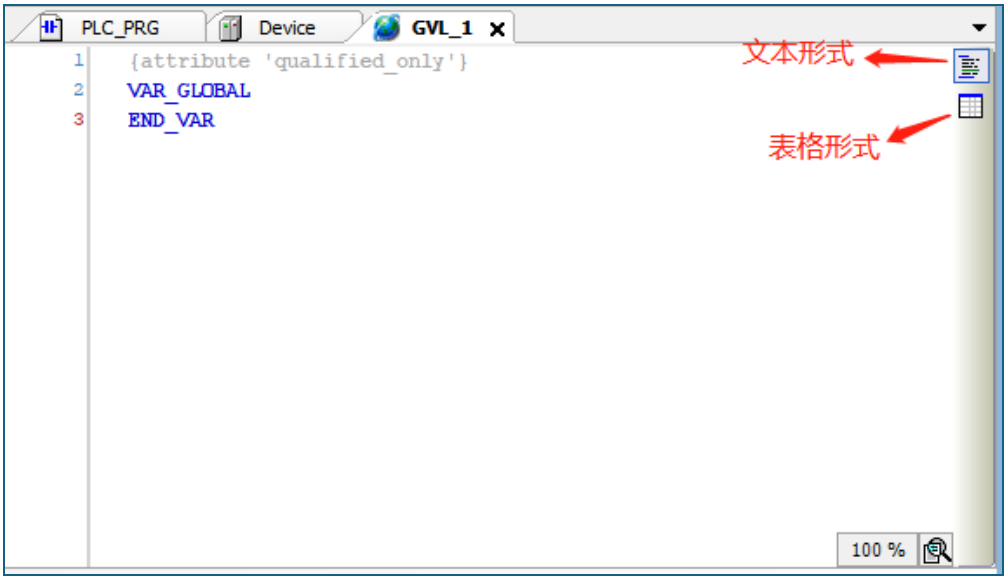
3.7 变量

CODESYS 中有多种变量供用户使用

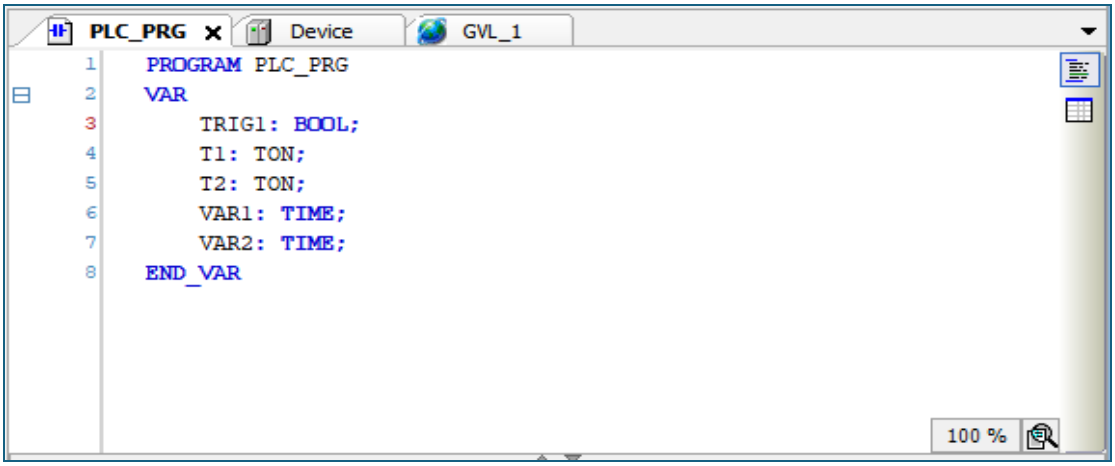
- 本地变量：VAR
- 输入变量：VAR_INPUT
- 输出变量：VAR_OUTPUT
- 输入/输出变量：(VAR_IN_OUT)
- 全局变量：VAR_GLOBAL
- 临时变量：VAR_TEMP
- 静态变量：VAR_STAT
- 外部变量：VAR_EXTERNAL
- 实例变量：VAR_INST
- 配置变量：VAR_CONFIG
- 常数变量：VAR CONSTANT
- 保持变量：PERSISTENT
- Retain 变量：RETAIN

变量声明

使用变量前，必须要对变量进行声明，变量可以在变量声明编辑器中定义。声明编辑器有两种显示形式：文本形式和表格形式，两种形式可以互相转换。



文本形式声明：



表格形式声明：

声明的内容一致，只是表现的形式不同。

3.7.1 局部变量

局部变量在编程对象的声明部分的关键字 VAR 和 END_VAR 之间声明。

您可以使用实例路径对局部变量进行只读访问。

您可以使用 attribute 关键字扩展局部变量。

例如：

```
VAR
    iVar1 : INT;
END_VAR
```

3.7.2 全局变量

全局变量是在整个工程中可以识别的普通变量，常量，外部或剩余变量。

您可以在全局变量列表中或在编程对象的声明部分中的关键字 VAR_GLOBAL 和 END_VAR 之间声明全局变量

在变量名前加点号（例如.iGlobVar1）时，系统会识别全局变量。

例如

```
VAR_GLOBAL
```

```
    iVarGlob1 : INT;
END_VAR
```

符号：

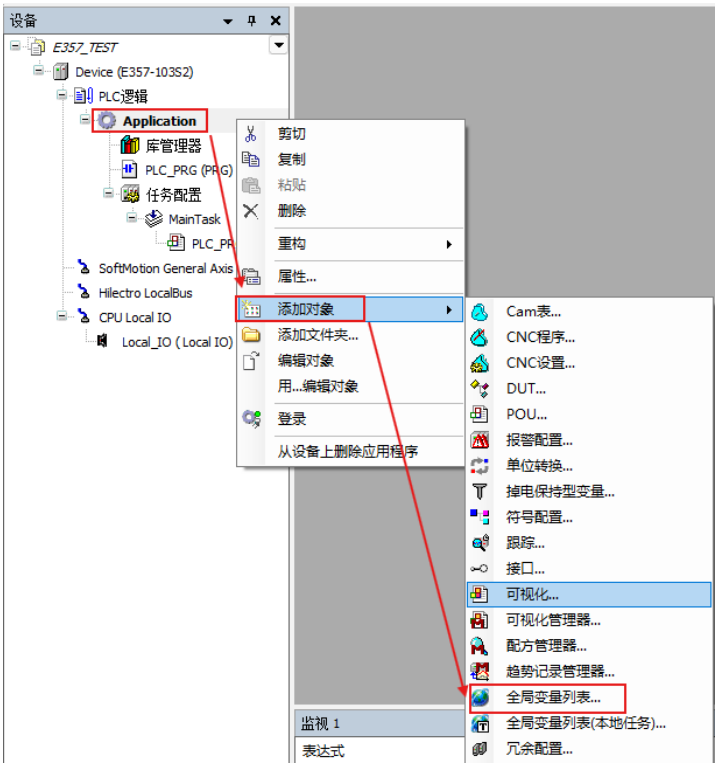
全局变量列表用于声明，编辑和显示全局变量。

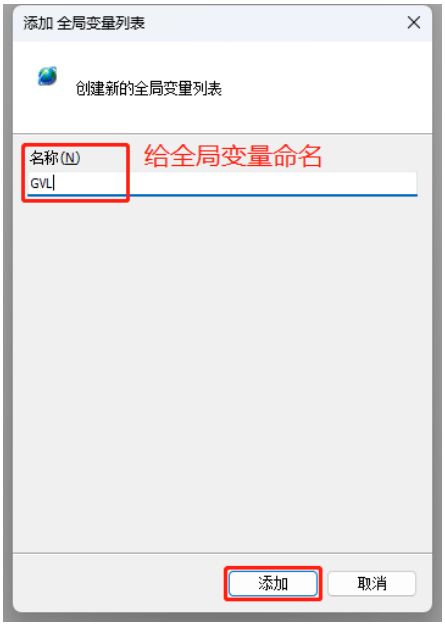
使用命令【工程】→【添加对象】→【全局变量列表】将 GVL 添加到应用程序或项目中。

如果在设备树中的某个应用程序下插入 GVL，则变量在该应用程序内有效。如果在 POUs 视图中添加 GVL，则变量对整个项目均有效。

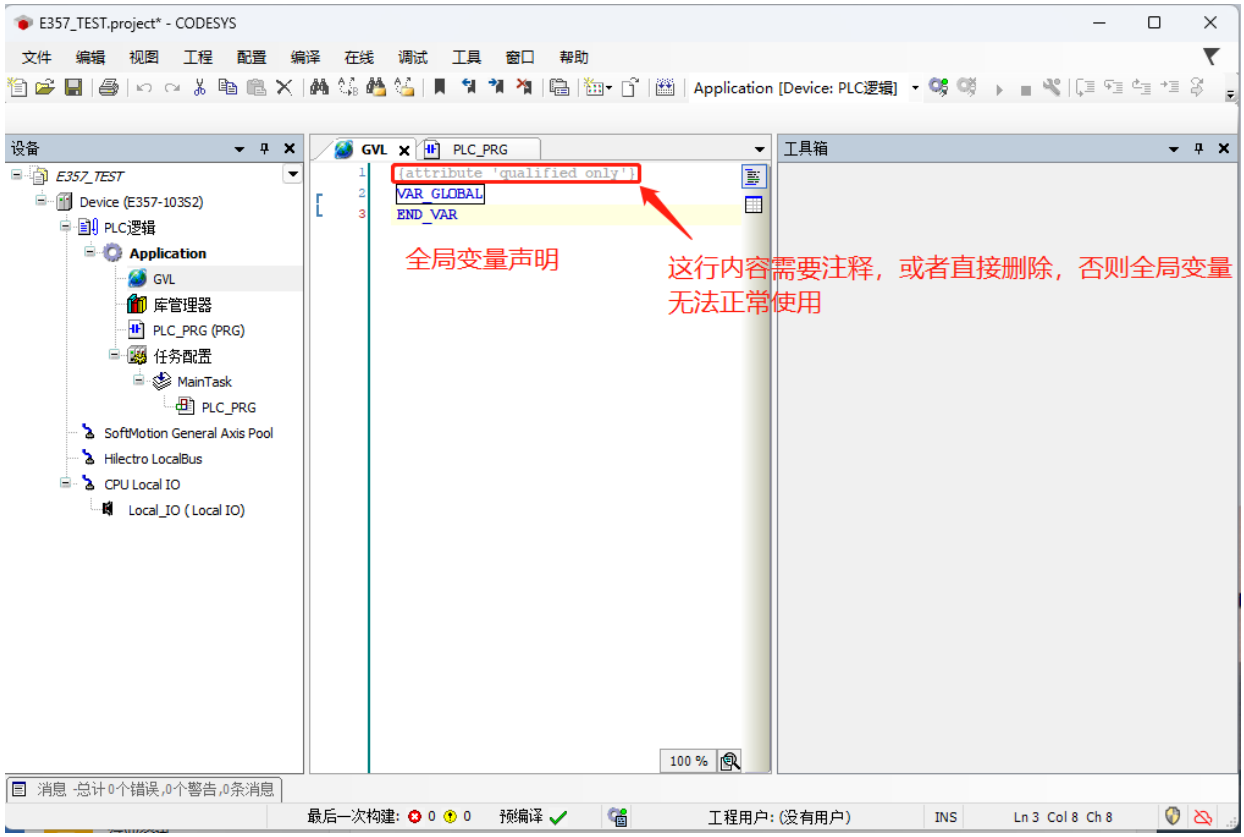
您可以在【工具】→【选项】的声明编辑器和文本编辑器类别中为对象的编辑器应用设置。

如果目标系统支持网络功能，则可以将 GVL 的变量转换为网络变量，然后将其用于与网络中的其他设备进行数据交换。为此，必须在属性对话框的网络变量选项卡中为 GVL 定义相应的属性。





建立好的全局变量表如下：



通信

4

4.1 EtherNET/IP 通信

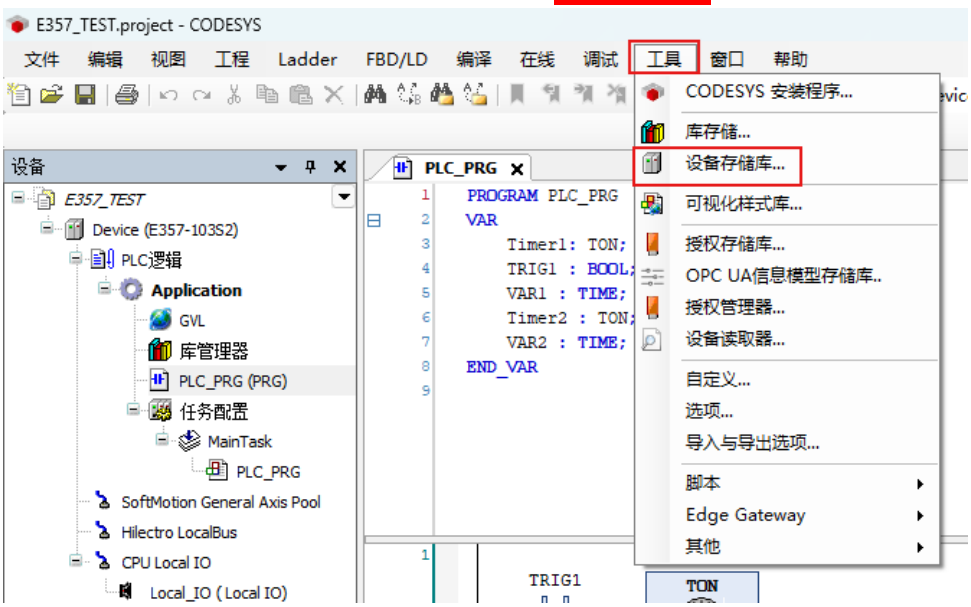
EtherNet/IP（Ethernet Industrial Protocol）指的是以太网工业协议，是罗克韦尔自动化公司开发的工业以太网通讯协定，由 ODVA 管理。EtherNet/IP 是基于 TCP/IP 系列协议，它定义了一个开放的工业标准，将传统的以太网与工业协议相结合。

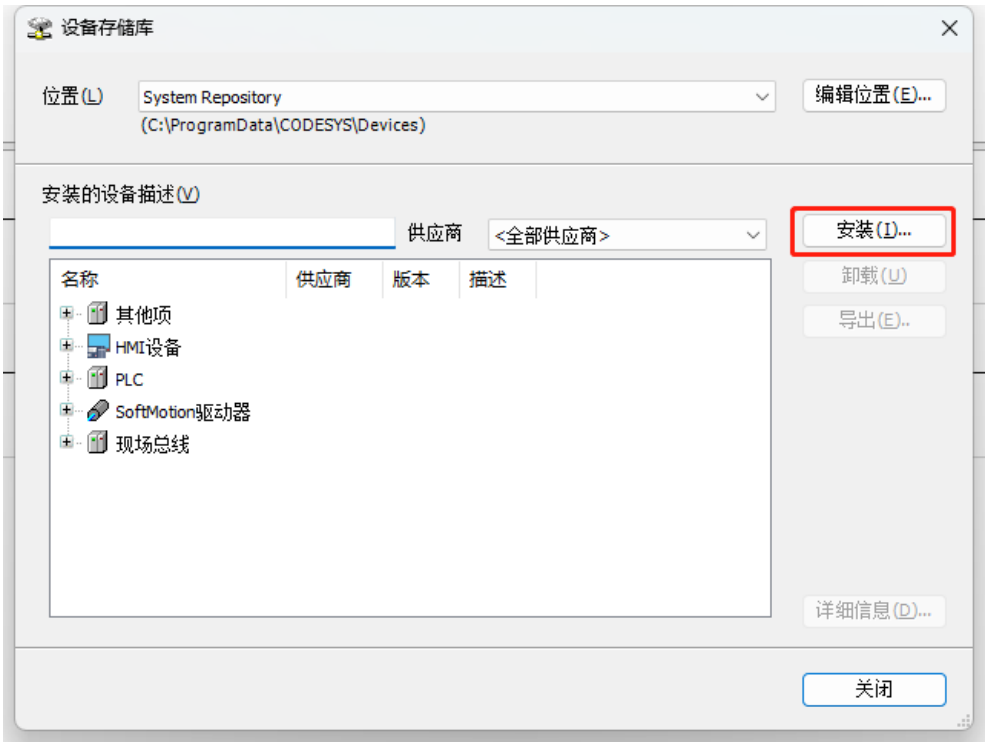
EtherNet/IP 通信中，需要了解两个概念：扫描器和适配器。进行 EtherNet/IP 通信时，一台设备相对于另一台设备，打开连接的一侧称为“扫描器”，类似主站；被打开的一侧称为“适配器”，类似从站。

在配置工程之前，可先将描述文件添加到 COSESYS 软件中。

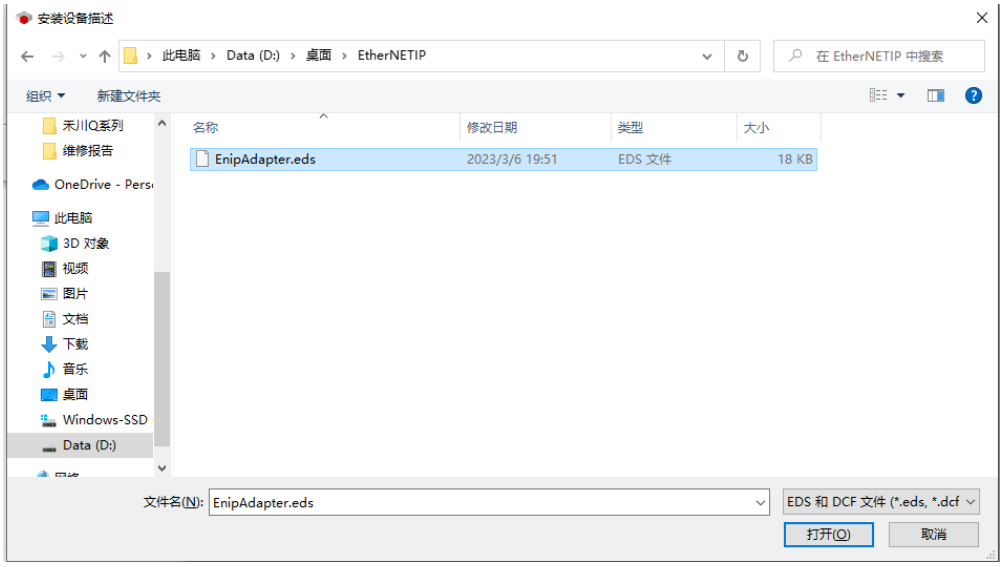
具体操作如下：

通过设备库添加 EtherNet/IP 从站设备描述文件（eds 文件），如下：





找到描述文件所在的路径后，直接安装描述文件即可。



设备描述文件可联系我司技术人员获取。

EtherNet/IP 主站配置步骤

（1） 新建工程，选择现场总线->EtherNetIP->以太网适配器下的 EtherNet。

（2） 导入第三方 eds 文件，将 EtherNet/IP 从站加入组网工程。

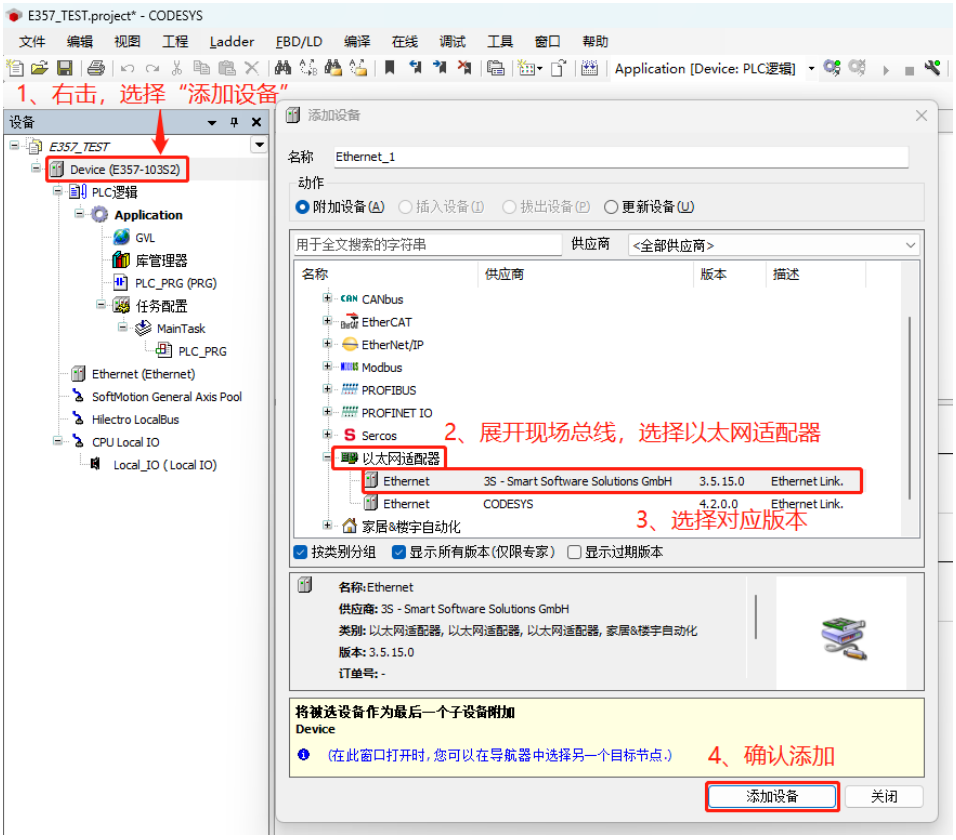
（3） 设置 EtherNet 通用设置的 IP 地址，以及从站通用界面的 IP 地址，确保均在一个局域网内。

（4） 添加从站默认连接，根据需要修改 RPI 通讯周期和任务周期时间。

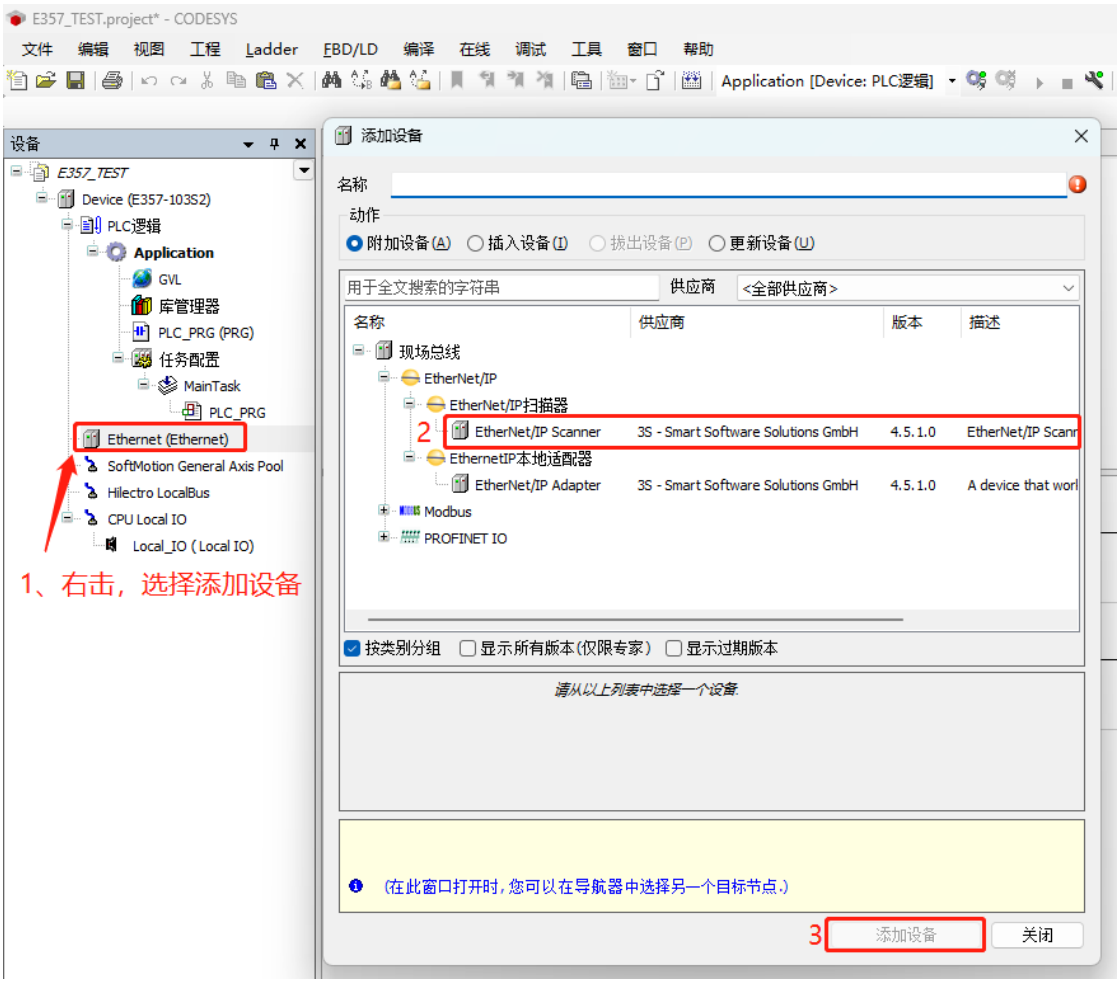
（5） 在从站 EtherNet/IP I/O 映射中进行参数映射。

（6） 根据需求使用编写用户 POU 程序。

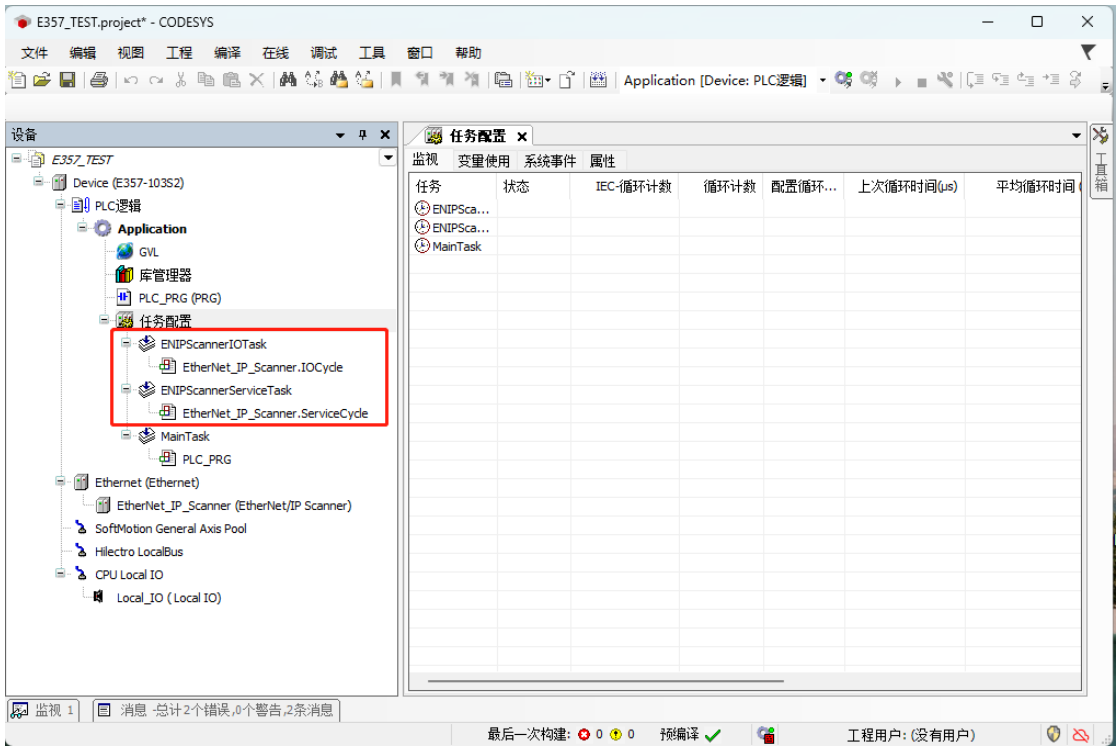
1、添加以太网适配器

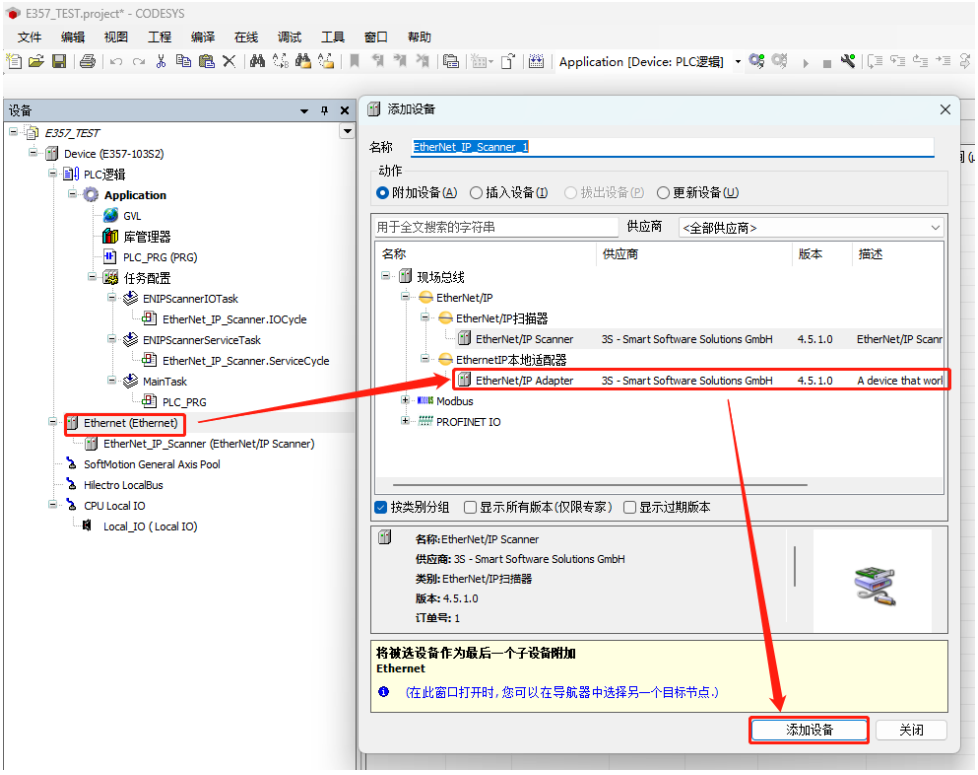


2、添加以太网扫描仪



3、在加入 EtherNet/IP 设备后的任务配置下，系统自动生成 ENIPScannerIOTask 和 ENIPScannerServiceTask 两个任务，用于更新 EtherNet/IP 的循环通讯数据和服务数据。ENIPScannerIOTask 任务优先级默认为 0，用户可根据实际情况进行调整，比如工程中另外有 EtherCAT 任务，其优先级必须指定为 0（最高），那么 ENIPScannerIOTask 的优先级可以修改为 1。





EtherNet/IP 从站工程一般配置步骤

（1） 新建工程，选择现场总线->EtherNetIP->以太网适配器下的 EtherNet。

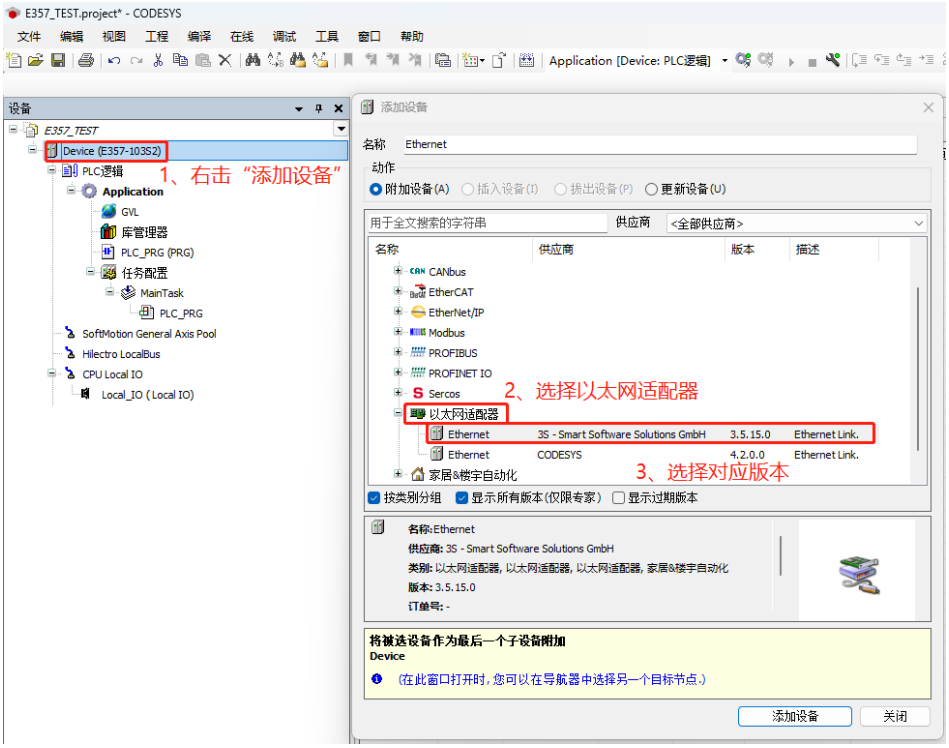
（2） 设置 EtherNet 通用设置的 IP 地址，和主站的从站通用界面的 IP 地址一模一样。

（3） 添加输入输出模块。

（4） 配置从站的 ENIPAdapterIOTask 任务周期与主站的 RPI 时间一致。

（5） 导出 EDS 文件。

（6） 根据需求使用编写用户 POU 程序。



4.2 Modbus TCP 通信

Modbus TCP 使用 TCP/IP 在站点间传送 Modbus 报文，Modbus TCP 结合了 TCP/IP 协议以及以 Modbus 协议作为应用协议标准的数据表示方法。Modbus TCP 通信报文被封装于以太网 TCP/IP 数据包中。与传统的串口方式相比，Modbus TCP 插入一个标准的 Modbus 报文到 TCP 报文中，不再带有数据校验和地址。

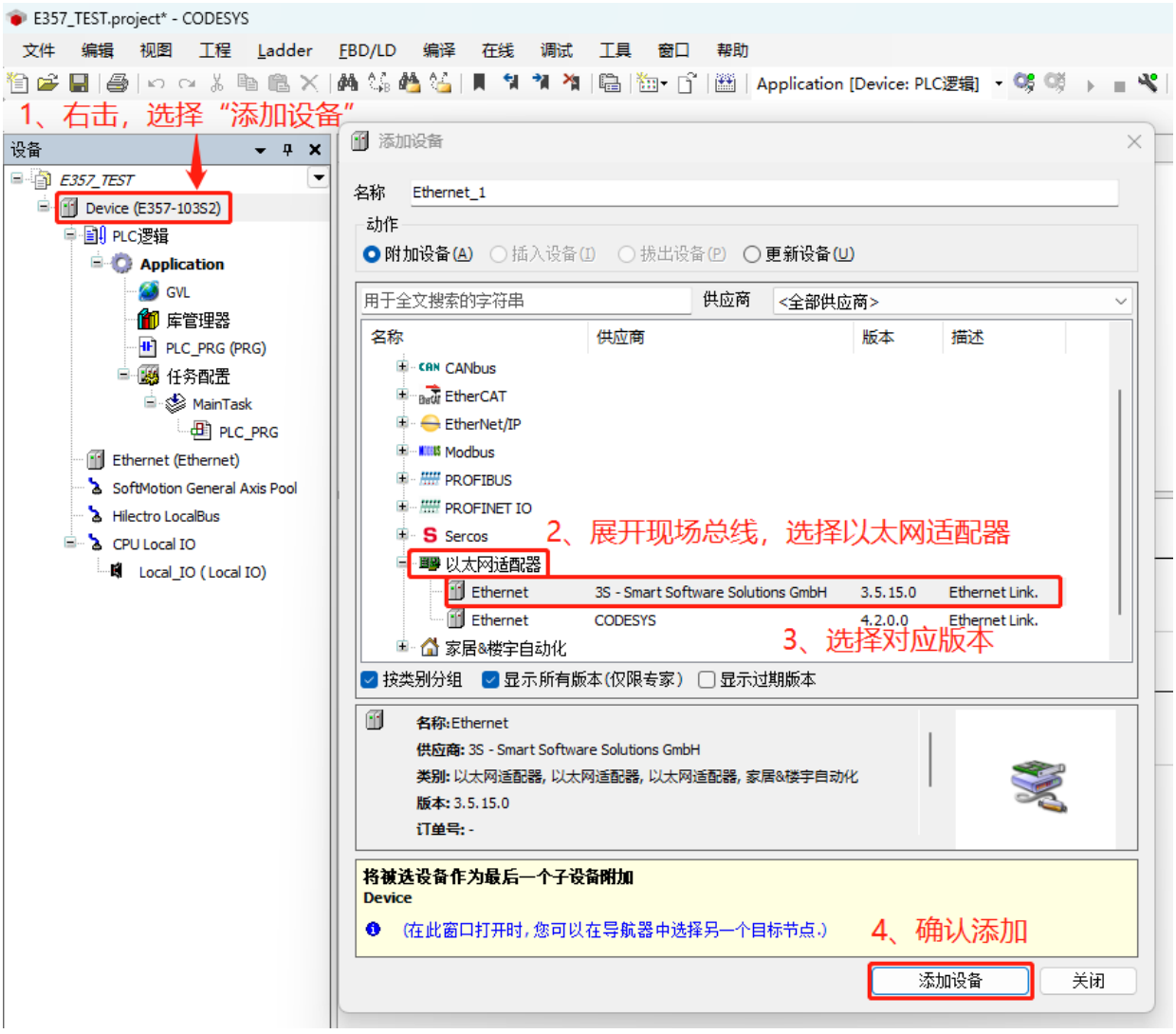
E300 系列可编程控制器本体 EtherNET 口支持 Modbus TCP 协议通讯主、从机形式。

主站形式：可编程控制器作为主站设备时，可与其它使用 Modbus TCP 协议的从机设备通讯。一个主站最多可以连接 32 个从站。

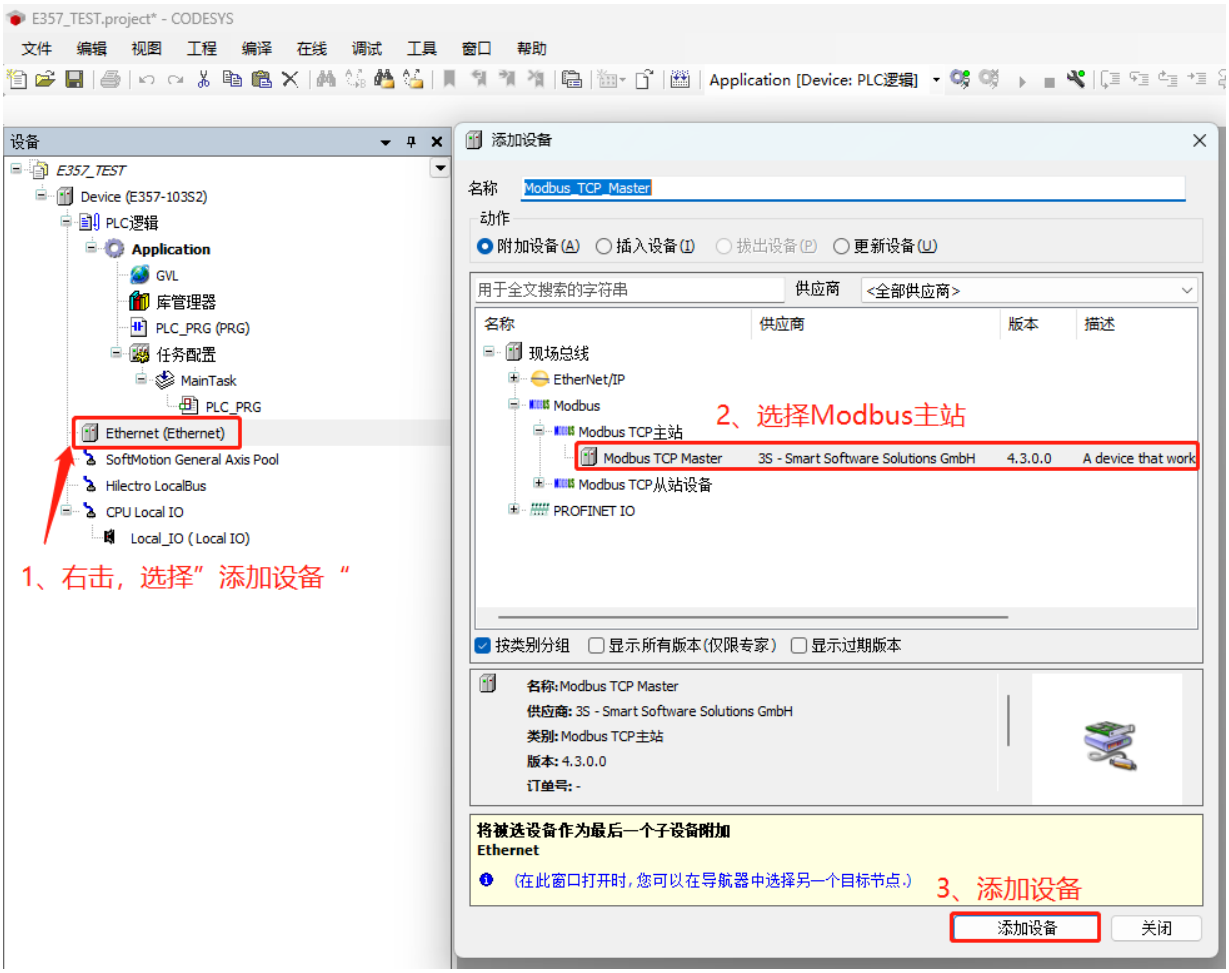
从站形式：可编程控制器作为从站设备时，只能对其它主站的要求做出响应。

通信配置步骤如下：

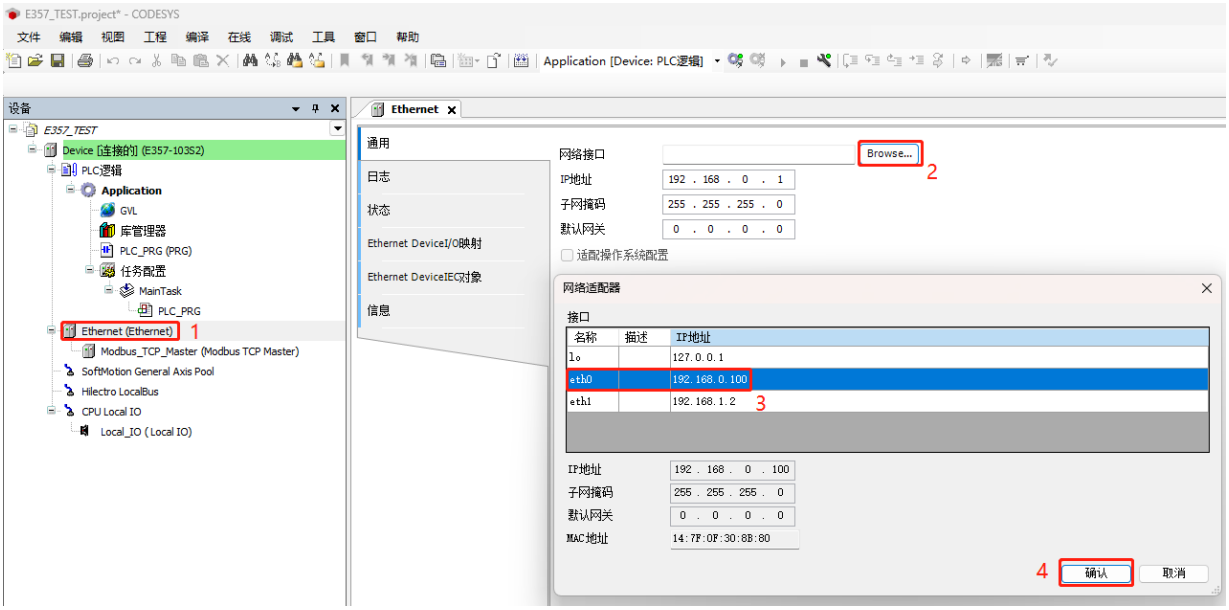
1、添加 EtherNET 网络，以太网适配器版本选择与软件相同的版本。



2、添加 Modbus TCP 主站



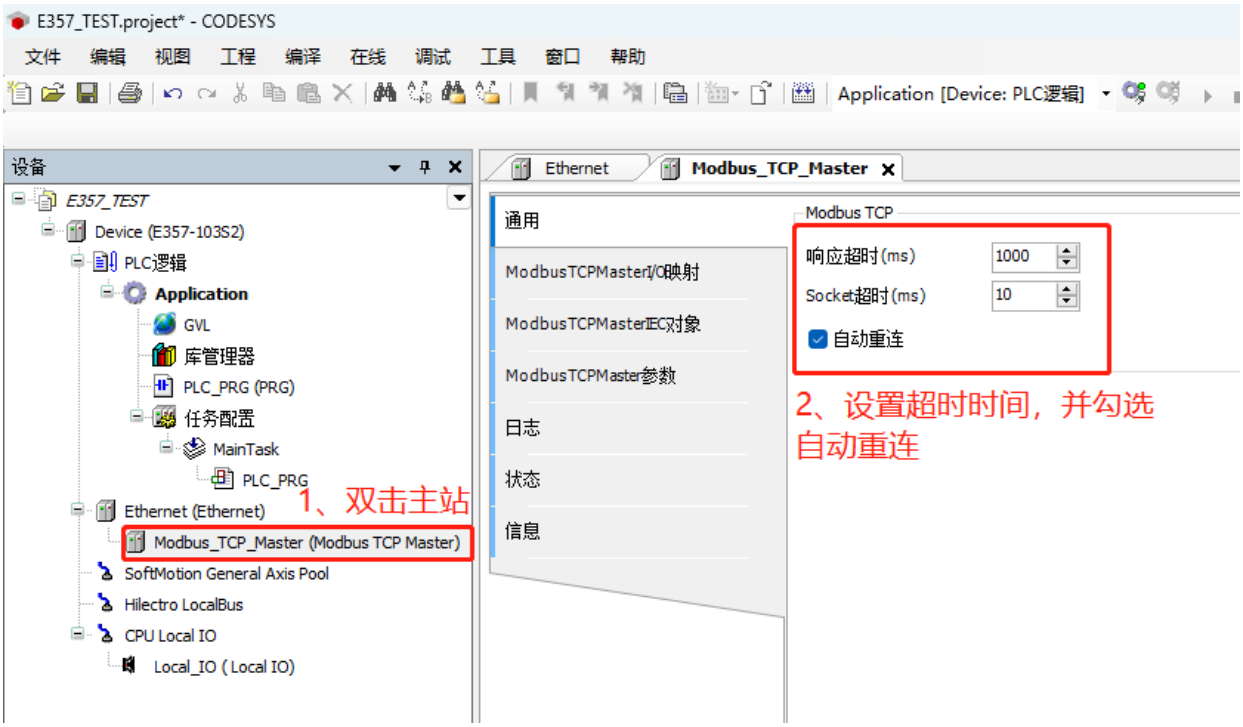
EtherNET 配置——需要连上 PLC



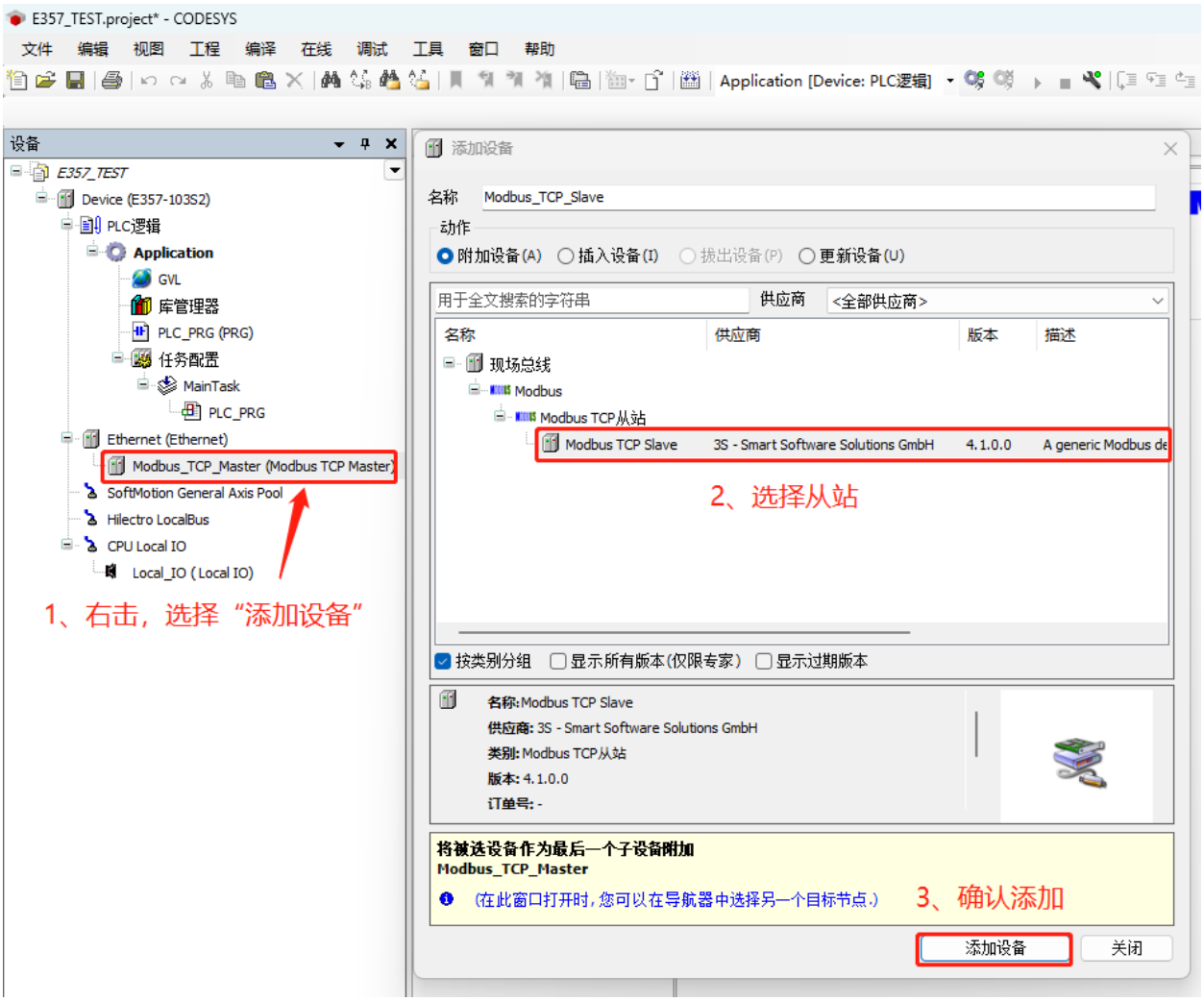
Modbus TCP 主站配置

响应超时：主站发出指令后，超过该时间从站未响应。

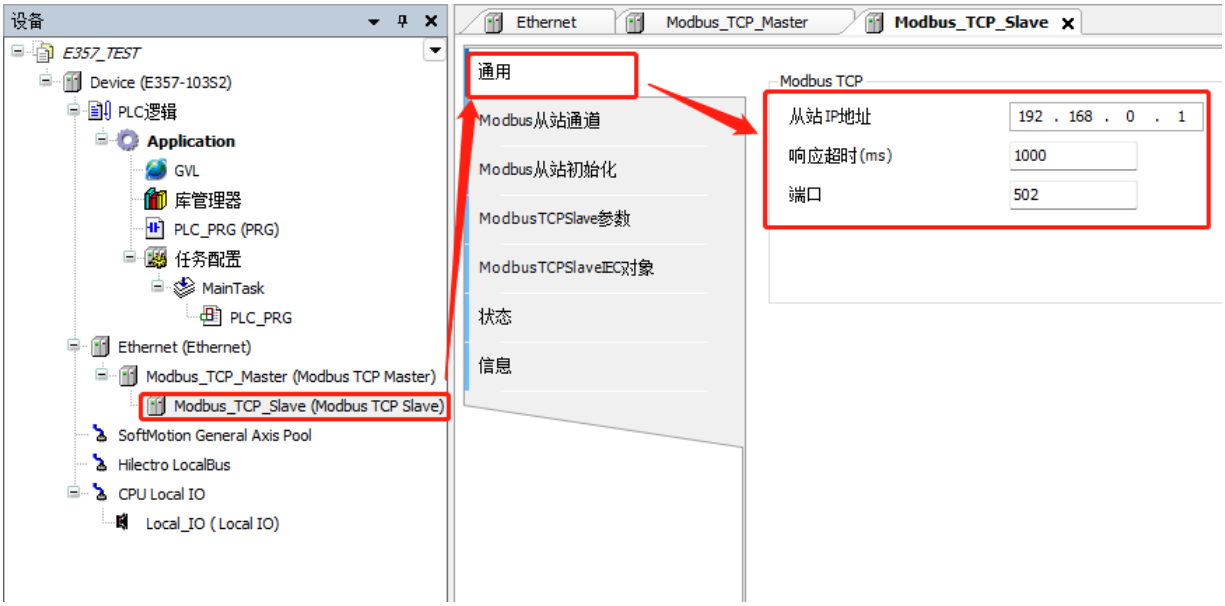
Socket 超时：



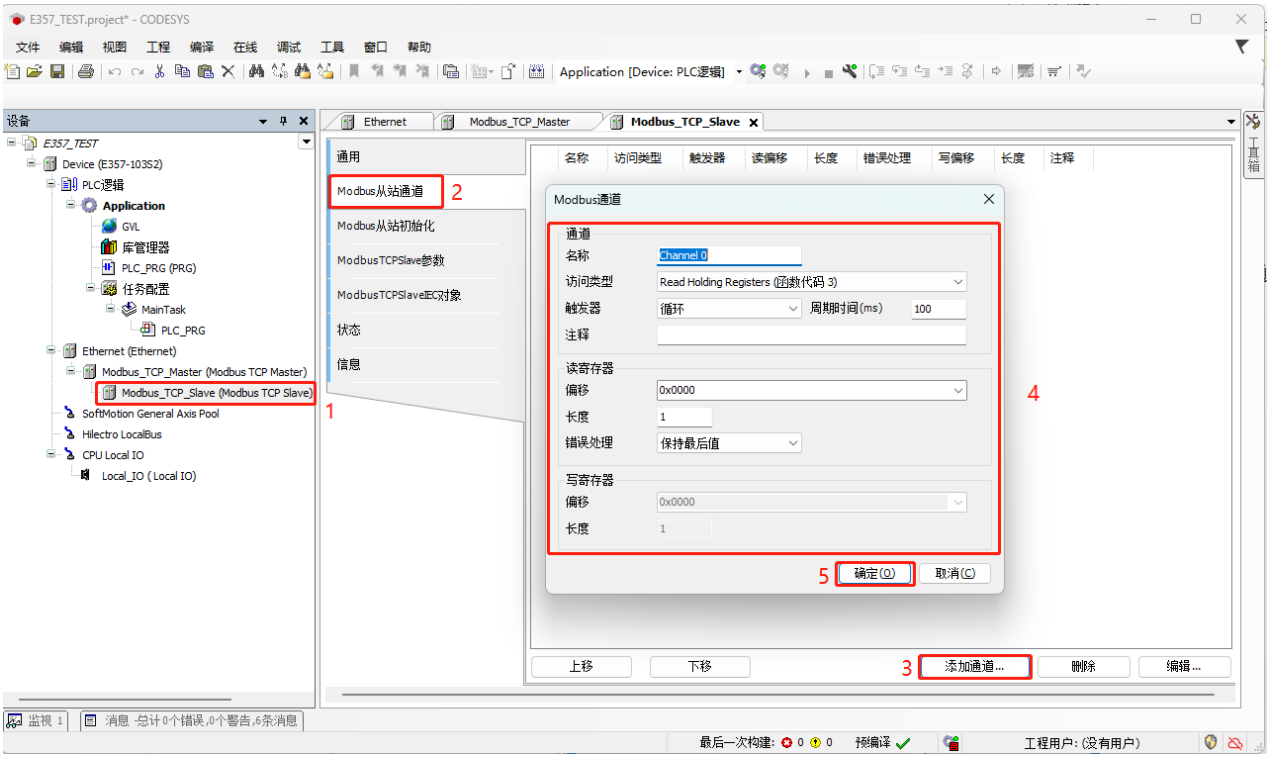
添加 Modbus TCP 从站



Modbus TCP 从站 IP 和端口号配置



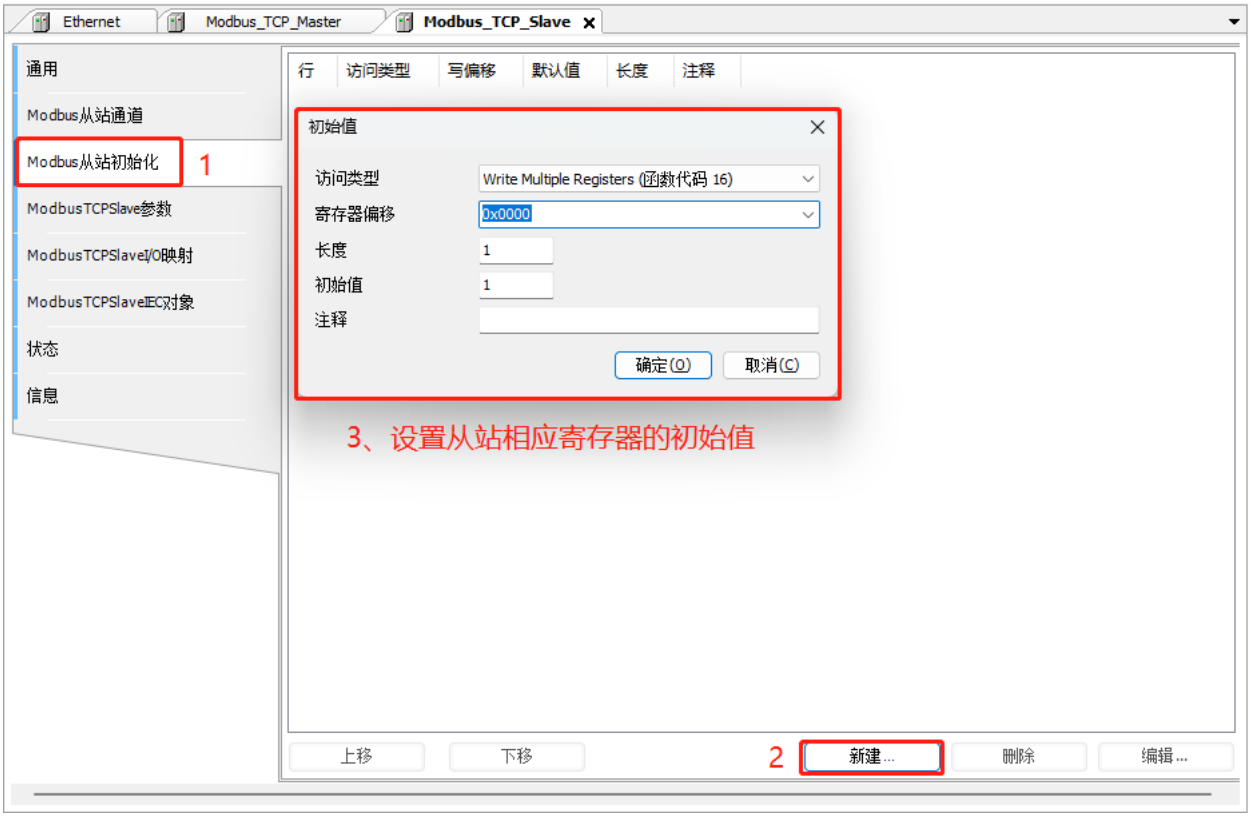
Modbus TCP 从站通道配置，根据下图添加 Modbus TCP 从站通道，然后再根据表格说明配置从站通道。



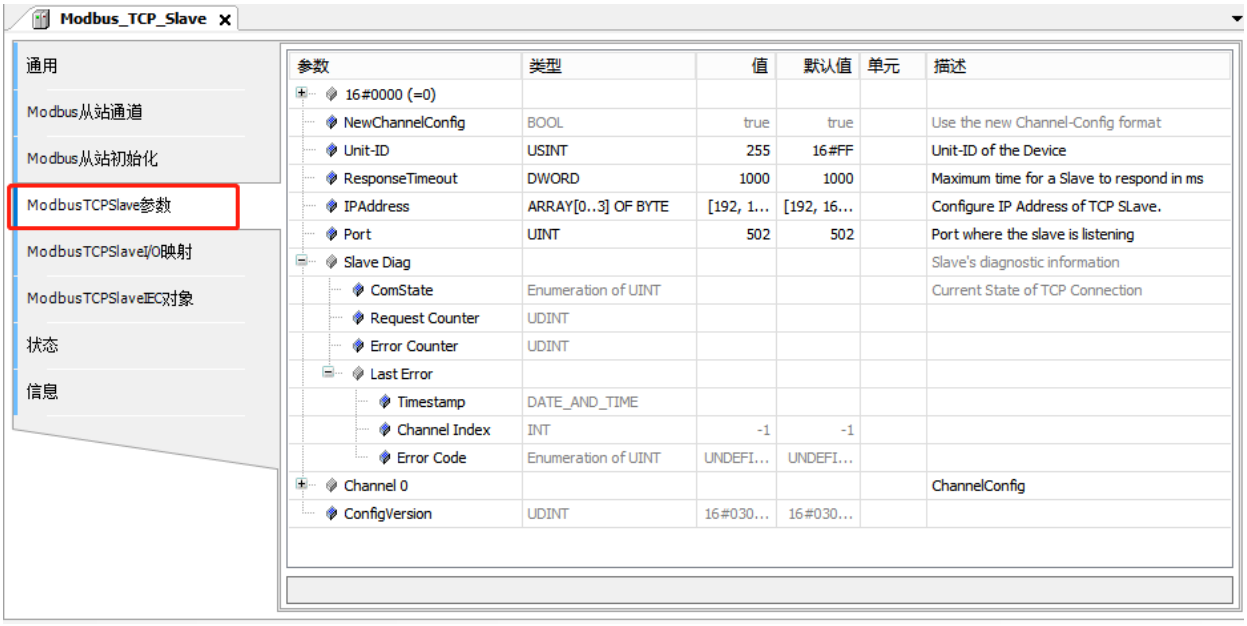
项目	说明
通道	
名称	通道名称，每一个通道代表一个 Modbus TCP 请求。

<table>
<tr><td>访问类型</td><td>
<table>
<tr><th>类型</th><th>说明</th></tr>
<tr><td>Read Coils(函数代码 1)</td><td>读线圈</td></tr>
<tr><td>Read Discrete Inputs(函数代码 2)</td><td>读离散输入</td></tr>
<tr><td>Read Holding Registers(函数代码 3)</td><td>读保持寄存器</td></tr>
<tr><td>Read Input Registers(函数代码 4)</td><td>读输入寄存器</td></tr>
<tr><td>Write Single Coils(函数代码 5)</td><td>写单个线圈</td></tr>
<tr><td>Write Single Registers(函数代码 6)</td><td>写单个保存寄存器</td></tr>
<tr><td>Write Multiple Coils(函数代码 15)</td><td>写多个线圈</td></tr>
<tr><td>Write Multiple Registers(函数代码 16)</td><td>写多个保持寄存器</td></tr>
<tr><td>Read / Write Multiple Registers(函数代码 23)</td><td>读/写多个保持寄存器</td></tr>
</table>
</td></tr>
<tr><td>触发器</td><td>触发类型：循环触发、上升沿触发、应用程序触发。</td></tr>
<tr><td>周期时间</td><td>触发器周期</td></tr>
<tr><td colspan="2">读寄存器</td></tr>
<tr><td>偏移</td><td>需要访问的从站寄存器偏移地址，偏移地址可设置。</td></tr>
<tr><td>长度</td><td>表示所读从站的数据长度</td></tr>
<tr><td>错误处理</td><td>发生错误时寄存器的值</td></tr>
<tr><td colspan="2">写寄存器</td></tr>
<tr><td>偏移</td><td>需要访问的从站寄存器偏移地址，偏移地址可设置。</td></tr>
<tr><td>长度</td><td>表示所写入从站的数据长度</td></tr>
</table>

Modbus TCP 从站初始化配置

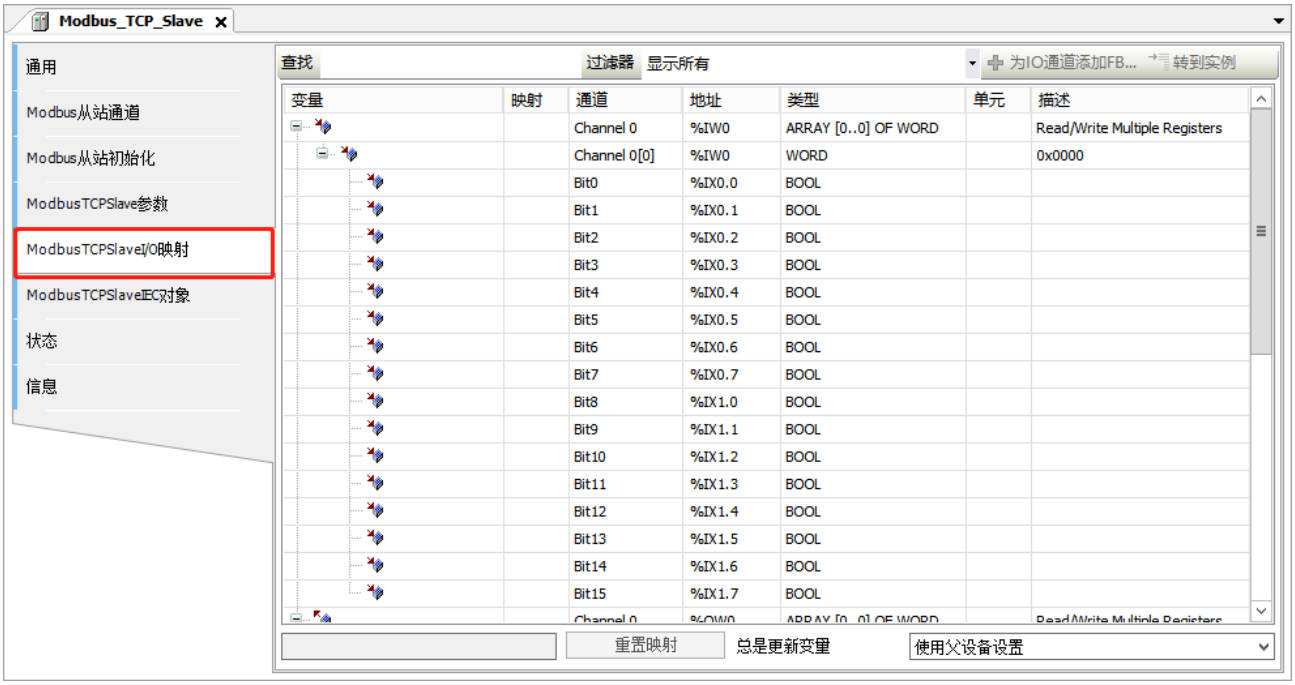


Modbus TCP 从站配置参数



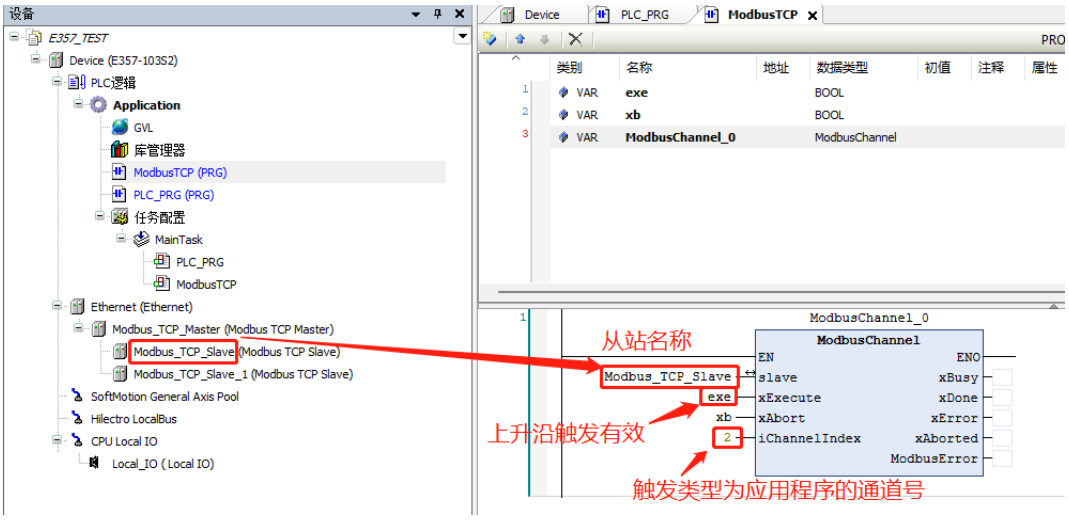
Modbus TCP 从站 IO 映射地址

在 Modbus TCP 从站通信设置中添加主从站通信配置后，Internal I/O 映射中会自动分配每条配置的映射地址，如上图第一行的%IW0 表示将读取的一个寄存器数值映射到%IW0 这个地址。另外，还可以通过输入助手或者直接输入示例变量路径，将程序中的自定义变量映射到 I/O 地址。



触发器类型为上升沿时，需要给对应的触发变量上升沿后才会进行相应通道数据传输。

触发器类型为应用程序时，需要触发 Modbuschannel 功能块来进行对应通道数据传输。



通过功能块引脚查看 Modbus 通讯状态等

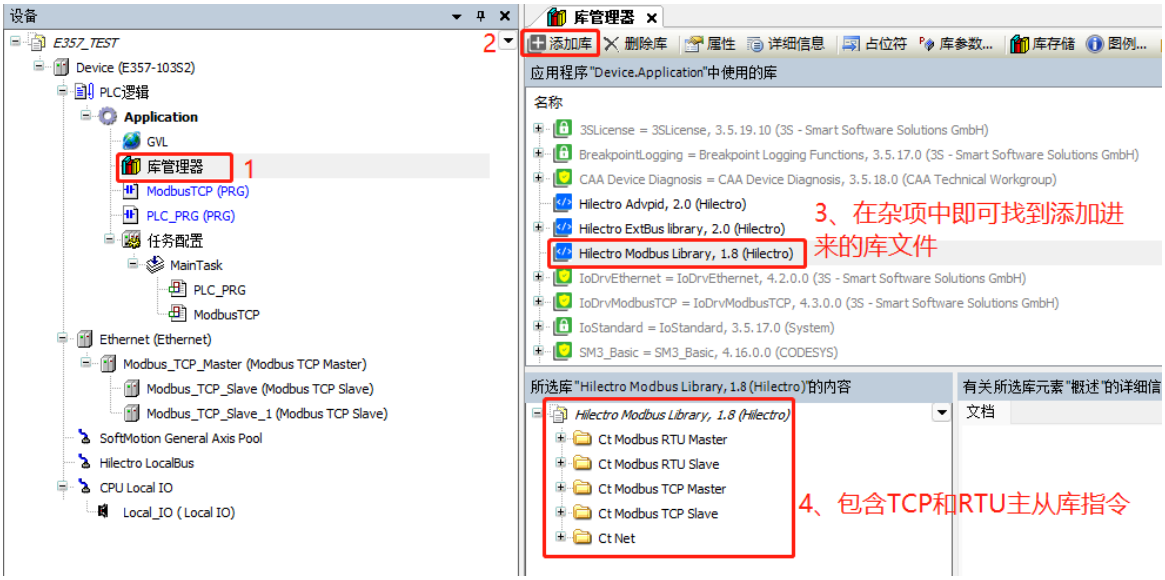
名称	输入 输出	数据类型	初始值	描述
xstop		bool	FALSE	如果为 TRUE，则停止对所有从站的每个新请求。如果为 FALSE，该过程将继续
xSlaveError		bool	FALSE	如果为 TRUE，表示有从站出错

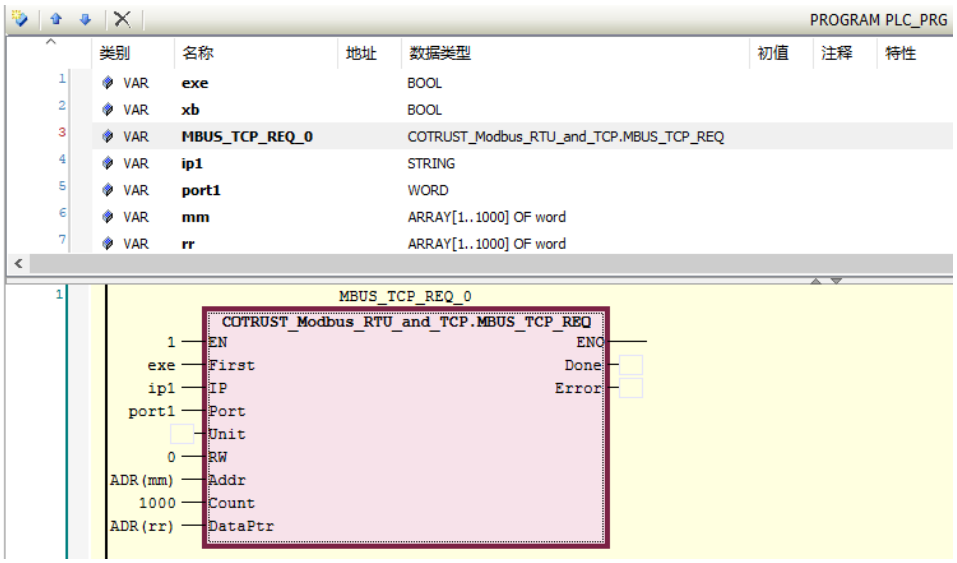
uiConnectedSlaves		UINT	0	通过 TCP/IP 连接的从属设备数量
xConfirmError		bool	FALSE	这个过程以上升的边缘继续进行
xInitDone		bool	FALSE	判断从站是都通讯上，为 on 通讯成功，OFF 通讯失败
xBusy		bool	FALSE	TRUE 表示请求正在处理
xDone		bool	FALSE	如果请求已成功结束为 TRUE
byModbusError		MB_ErrorCodes		枚举中定义的当前错误 MB_ErrorCodes
ComSettings		ModbusTCPCommSettings		当前配置的通信设置（从站的 IP 及端口号）

Modbus 通讯错误代码

名称	错误码	内容
RESPONSE_SUCCESS	16#0	一切正常
ILLEGAL_DATA_ADDRESS	16#2	从站不支持此寄存器偏移
SLAVE_DEVICE_FAILURE	16#4	执行请求时出现不可恢复的错误
MEMORY_PARITY_ERROR	16#8	功能代码 20、21 的特殊错误
GATEWAY_DEVICE_FAILED_TO_RESPOND	16#B	使用网关后面的设备时出现特殊错误（设备不响应）
RESPONSE_TIMEOUT	16#A1	没有及时回复
RESPONSE_CRC_FAIL	16#A2	响应的校验和不正确
RESPONSE_WRONG_SLAVE	16#A3	响应不是来自预期的从属服务器
RESPONSE_WRONG_FUNCTIONCODE	16#A4	响应不是预期的函数代码
REQUEST_FAILED_TO_SEND	16#A5	本地 COM 端口错误，请求未发送
RESPONSE_INVALID_DATA	16#A6	响应包含无效数据
RESPONSE_INVALID_PROTOCOL	16#A7	响应不是 Modbus 协议
RESPONSE_INVALID_HEADER	16#A8	Modbus MBAP（协议，长度）标头的任何字段都是无效
INVALID_PARAMETER	16#A9	Modbus MBAP（协议，长度）标头的任何字段都是无效
UNDEFINED	16#FF	Modbus MBAP（协议，长度）标头的任何字段都是无效

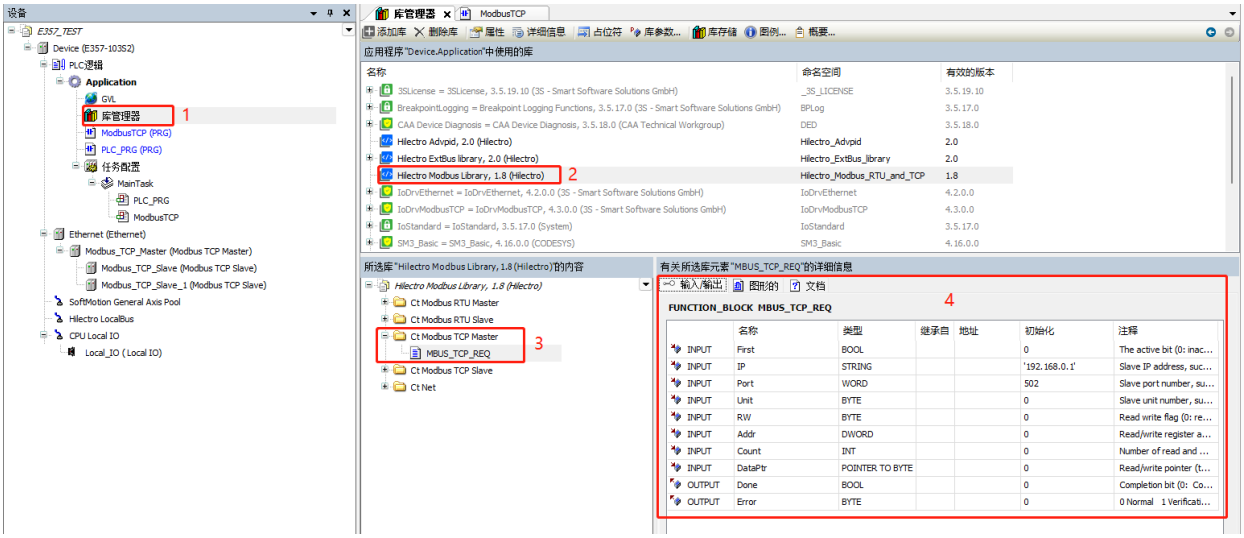
Modbus TCP 库指令应用示例

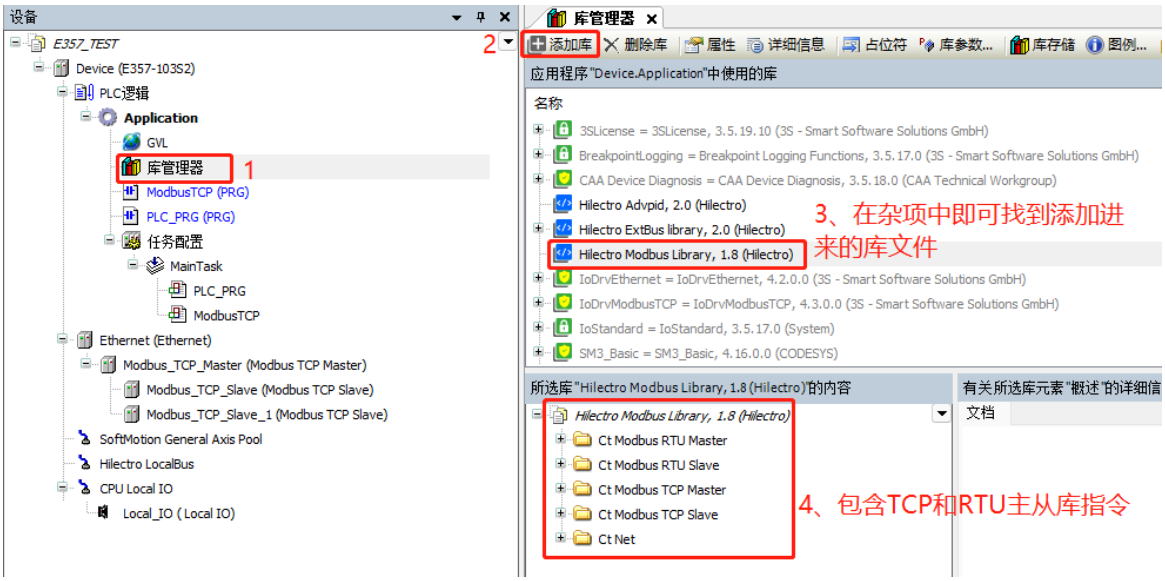


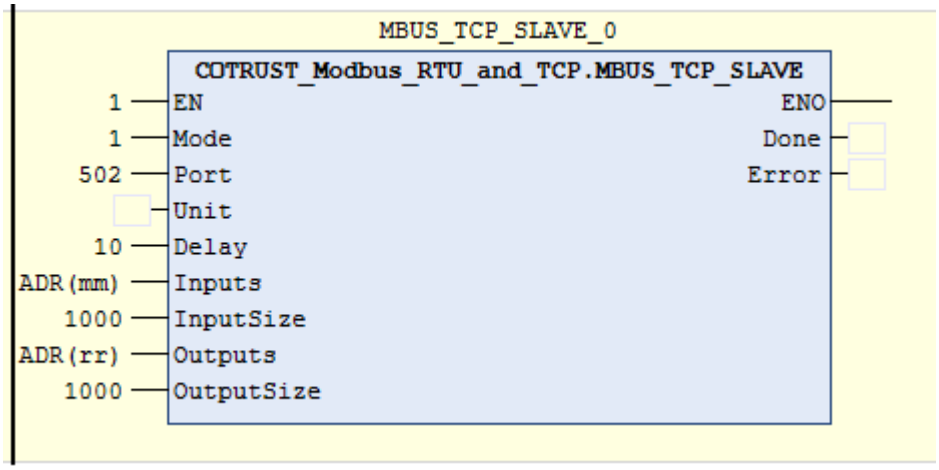


FUNCTION_BLOCK MBUS_TCP_REQ

名称	类型	继承自	地址	初始化	注释
First	BOOL			0	激活位(0:非激活 1:激活)，每激活一次，指令执行一次。
IP	STRING			'192.168.0.1'	从站ip地址,例如'192.168.0.1'
Port	WORD			502	从站端口号,例如502
Unit	BYTE			0	从站单元号,例如0
RW	BYTE			0	读写标志(0:读 1:写)
Addr	DWORD			0	读写寄存器地址(比如 40001,30001)
Count	INT			0	读写字节数量(取值范围0-120个字)
DataPtr	POINTER TO BYTE			0	读写指针(存放读取到的数据的位置或存放要写到寄存器的数据)
Done	BOOL			0	完成位(0:指令正在执行 1:指令执行完成)
Error	BYTE			0	0 正常 1 校验错误 2 波特率错误 3 超时错误 4 请求错误 5 总线未使能 6 通讯忙 7 应答错误 8 CRC错误 9 硬件错误



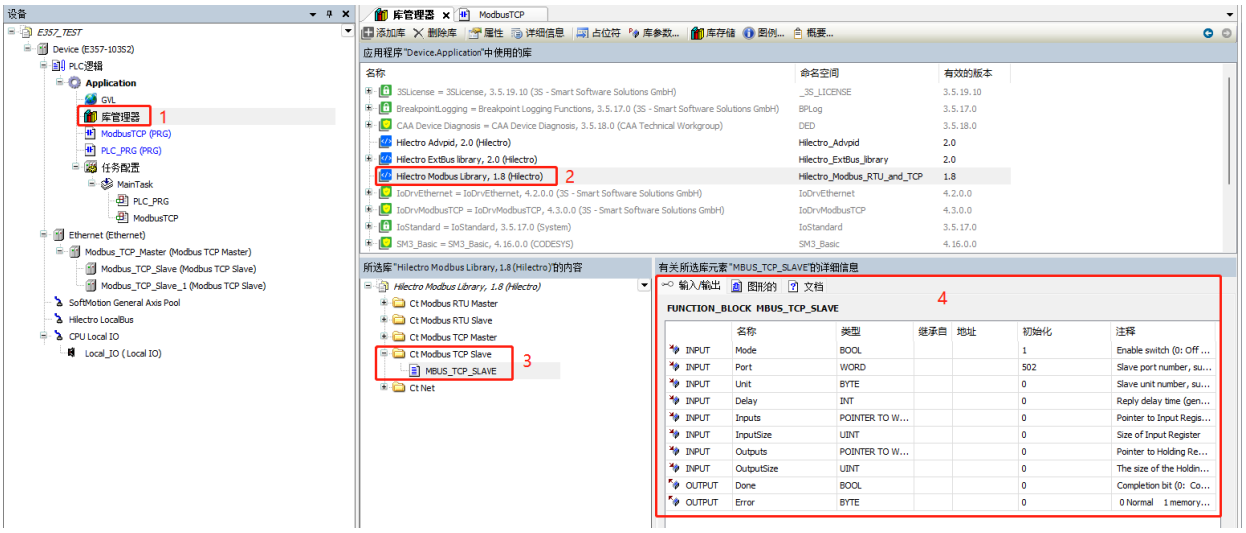




输入/输出 图形的 文档

FUNCTION_BLOCK MBUS_TCP_SLAVE

名称	类型	继承自	地址	初始化	注释
Mode	BOOL			1	使能开关(0:关闭 1:打开)
Port	WORD			502	从站端口号,例如502
Unit	BYTE			0	从站单元号,例如0
Delay	INT			0	回复延时时间(一般设置为0即可，如其他通讯参数均正确但仍无法通讯，可适当增加该值)
Inputs	POINTER TO WORD			0	指向 Input Register 的指针
InputSize	UINT			0	Input Register 的大小
Outputs	POINTER TO WORD			0	指向 Holding Register 的指针
OutputSize	UINT			0	Holding Register 的大小
Done	BOOL			0	完成位(0:指令正在执行 1:指令执行完成)
Error	BYTE			0	0 正常 1 内存范围错误 2 非法的波特率或校验 3 非法的从站地址 4 无效的协议参数 5 保持寄存器与modbus缓冲区重叠 6 接收校验错误 7 接收功能码请求 8 非法的功能码请求 9 非法的内存地址请求 10 modbus 总线未初始化



Modbus 对应地址关系

MB0	MW0	40001	MD0
MB1			
MB2	MW1	40002	
MB3			
MB4	MW2	40003	MD1
MB5			
MB6	MW3	40004	
MB7			
MB8	MW4	40005	MD2
MB9			
MB10	MW5	40006	
MB11			
MB12	MW6	40007	MD3
MB13			
MB14	MW7	40008	
MB15			
MB16	MW8	40009	MD4
MB17			
MB18	MW9	40010	

Modbus(位)		MW	MB	MB
40007		6=40007-40001	12=6*2	13=6*2+1
6				

MXn.n	MW		MB 0	MB 1	modbus	
16	8=int(16/2)		16=8*2	17=8*2+1	40009=40001+8	40009
2(0~7)			14=8+6	6=8-2	14	2

modbus	modbus	MW		MD
40188	40188	187=40188-40001		93=int(187/2)

MD	modbus	MW	MW	MD
18	40037=40001+36	36=18*2	37=36+1	18

MB	modbus	MW		MD
500	40251=40001+250	250=int(500/2)		125=int(250/2)

MW	modbus	MB	MB	MD
45	40046=40001+45	90=45*2	91=45*2+1	22=int(45/2)

M0.0	40001.16
M0.1	40001.15
M0.2	40001.14
M0.3	40001.13
M0.4	40001.12
M0.5	40001.11
M0.6	40001.10
M0.7	40001.9
M1.0	40001.8
M1.1	40001.7
M1.2	40001.6
M1.3	40001.5
M1.4	40001.4
M1.5	40001.3
M1.6	40001.2
M1.7	40001.1

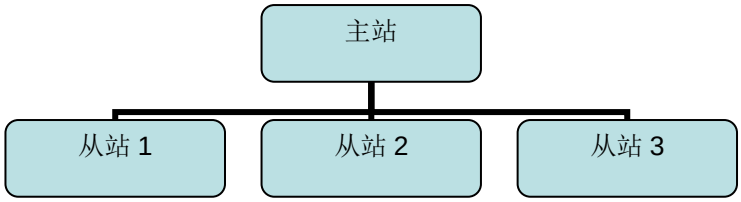
4.3 RS485 串口通讯通信

E300 系列可编程控制器本体自带一到两个 RS485 串口，支持 Modbus 协议通讯主、从机形式。

主站形式：可编程控制器作为主站设备时，可与其它使用 Modbus 协议的从机设备通讯；与其他设备进行数据交换。例：海天 E300 系列 PLC，可以通过通讯来控制变频器。

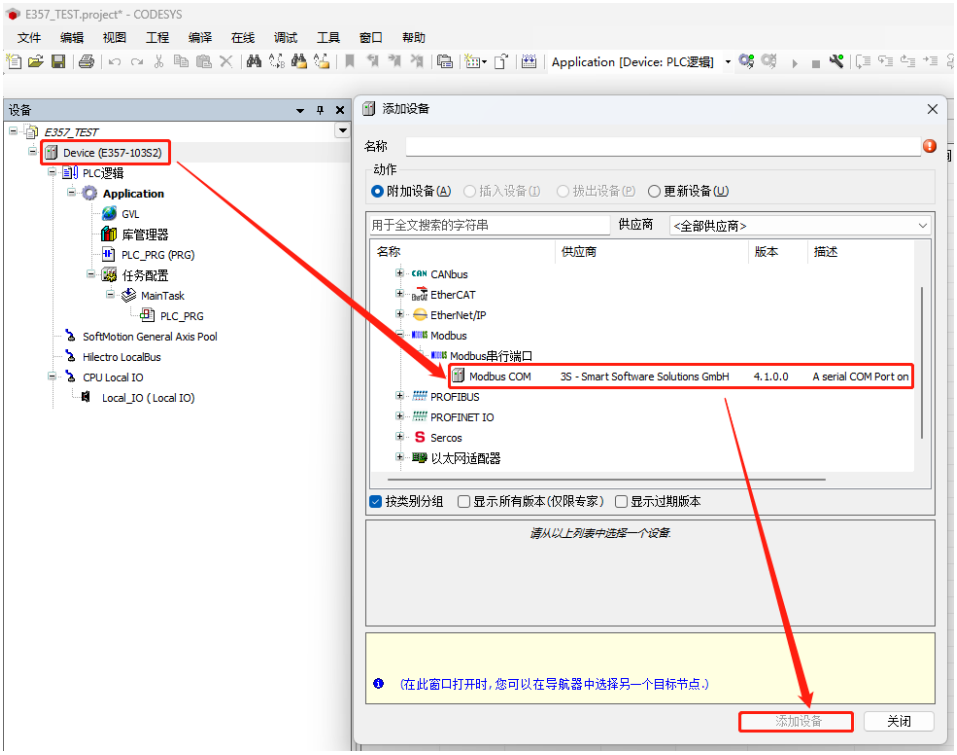
从站形式：可编程控制器作为从站设备时，只能对其它主站的要求做出响应。

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 RS485+连在一起，所有的 RS485-连在一起）

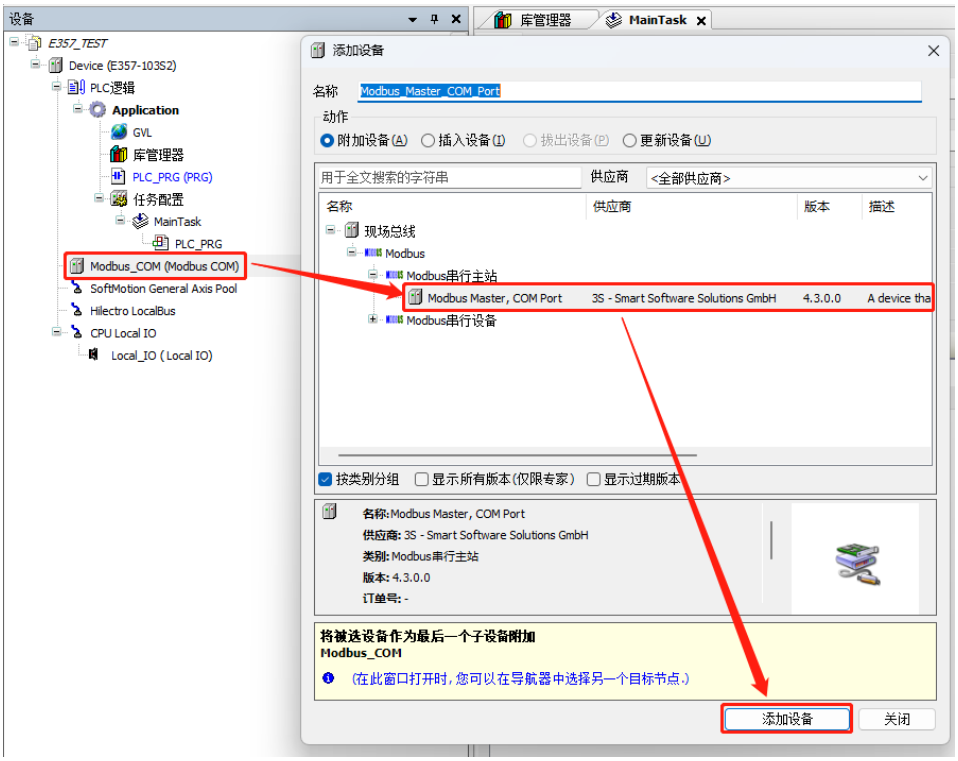


CODESYS Modbus master RTU 主站通信示例

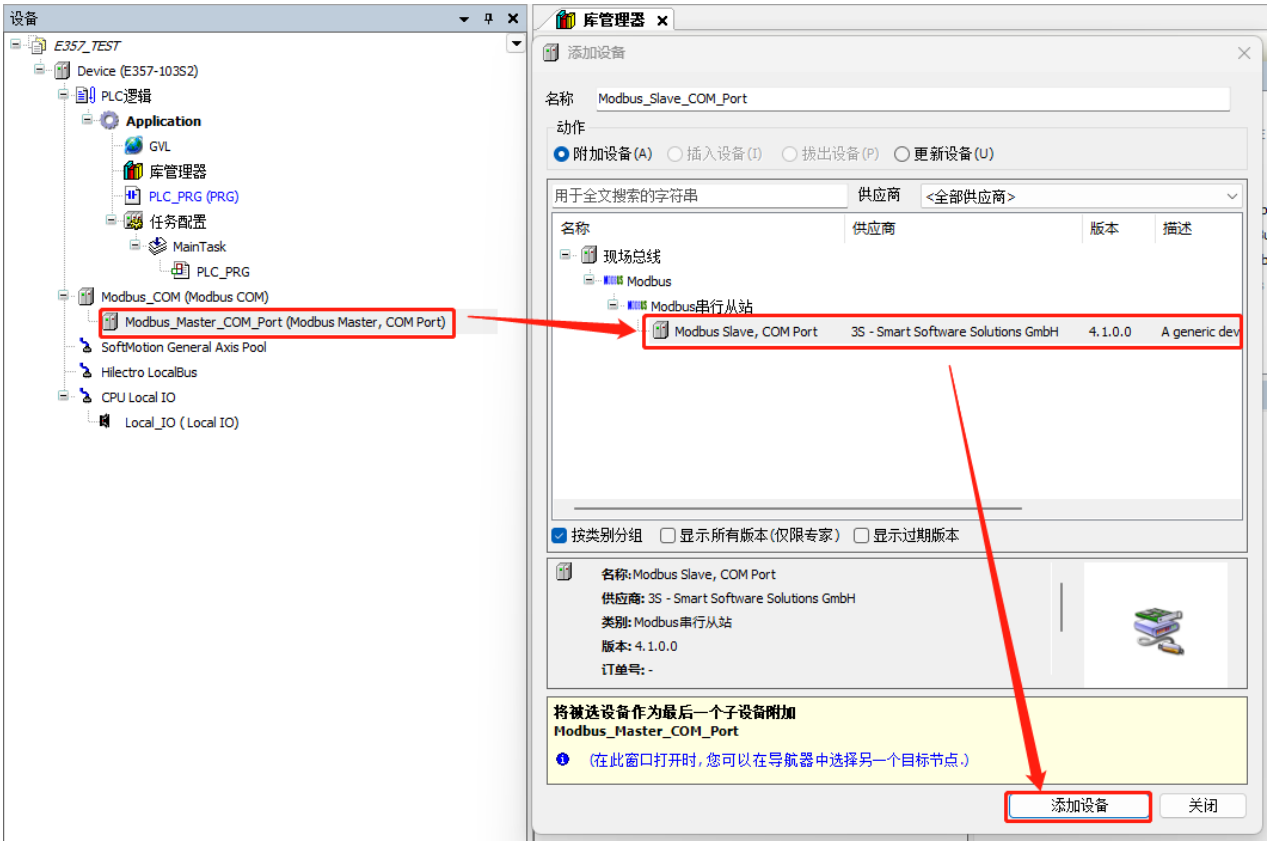
1、添加 Modbus 串行端口



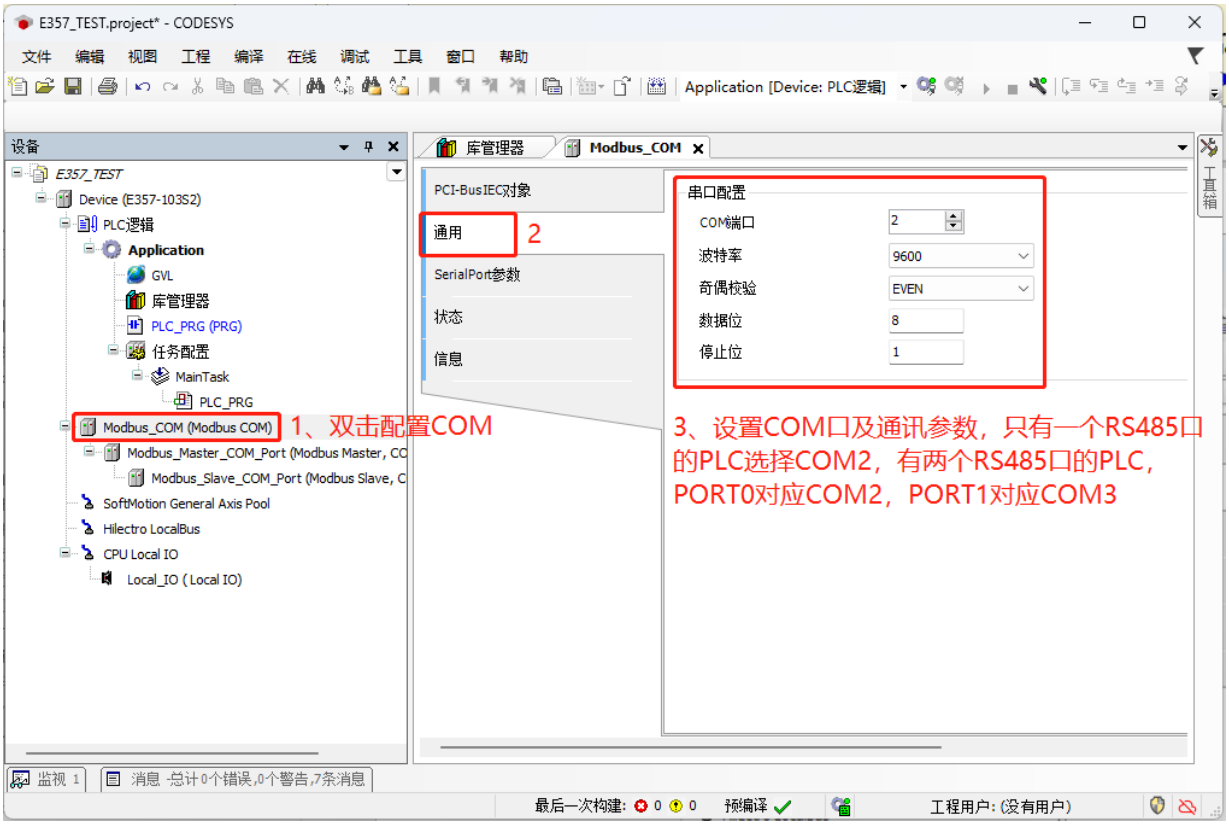
2、在 Modbus 串行端口下添加 Modbus 串行主站



3、在 Modbus 主站下添加 Modbus 串行从站



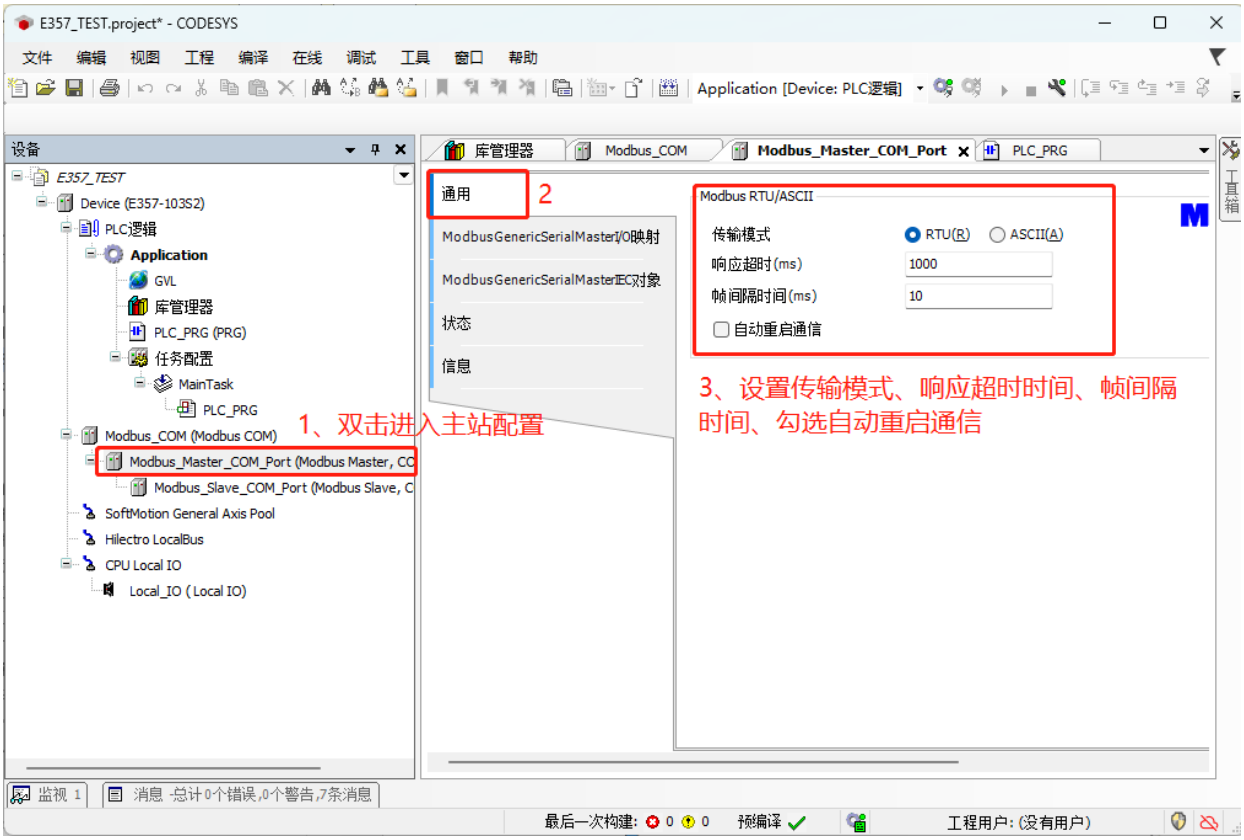
4、设置 COM 口及通讯参数



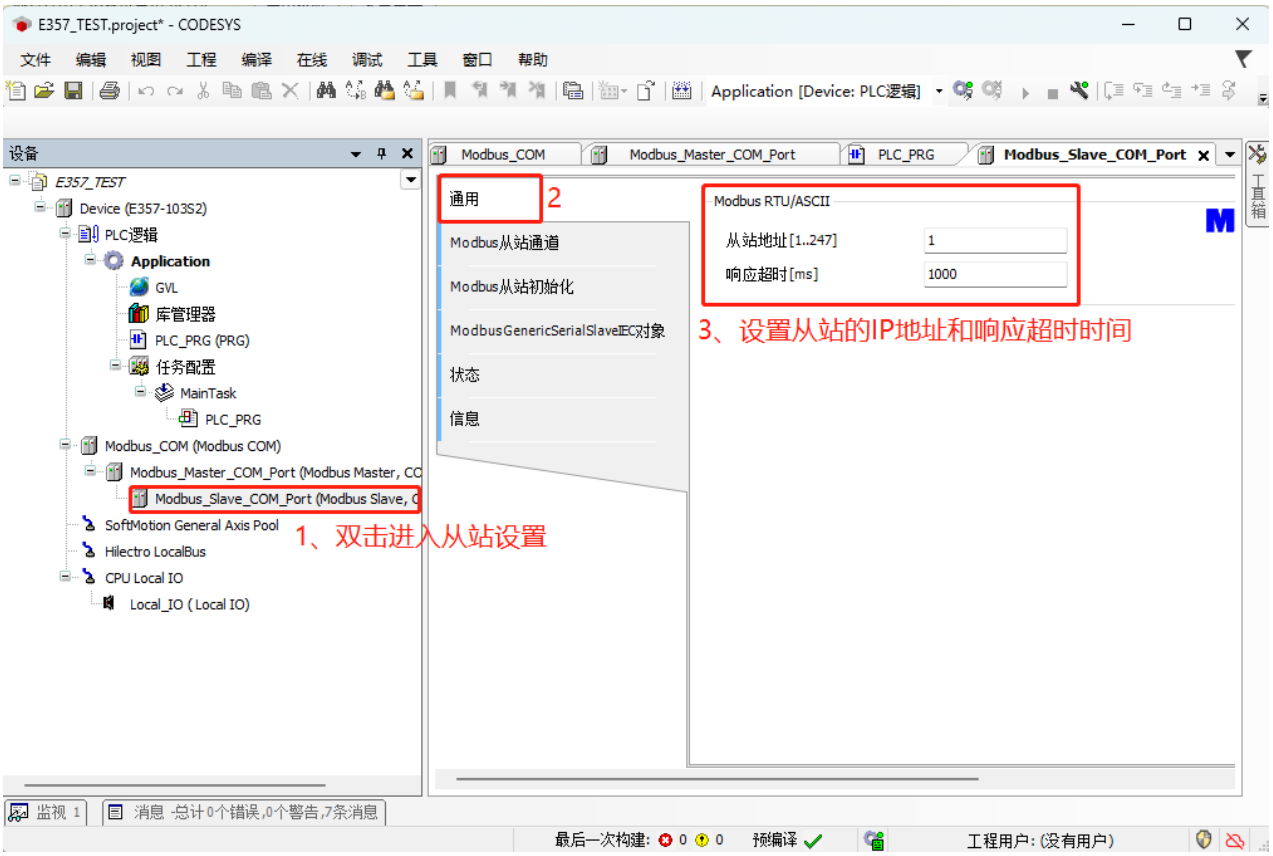
5、配置主站

响应超时：主站发出指令后，超过该时间从站未响应。

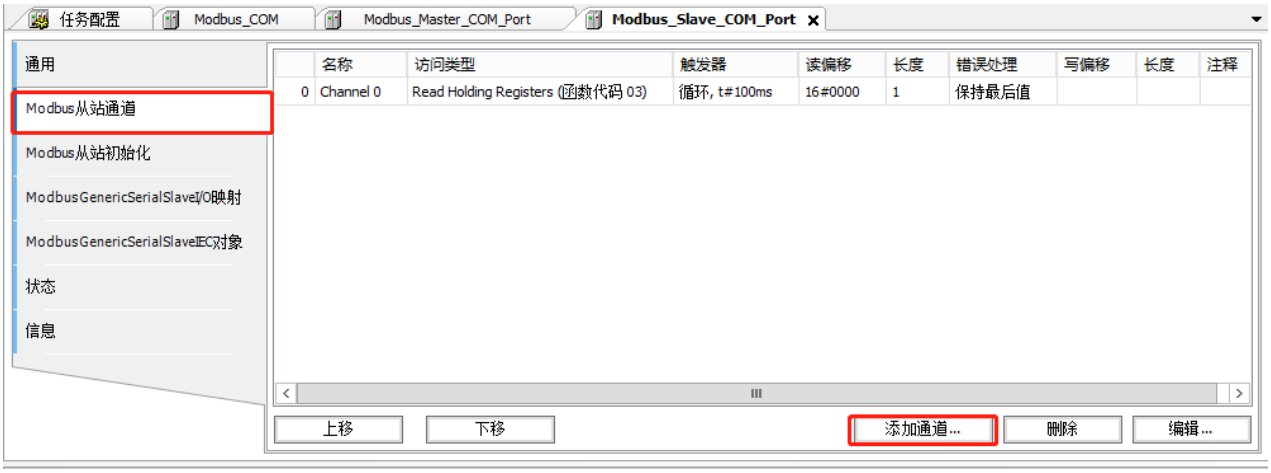
帧间隔时间：主站接收上一个响应数据帧到下一个请求数据帧等待的时间间隔。

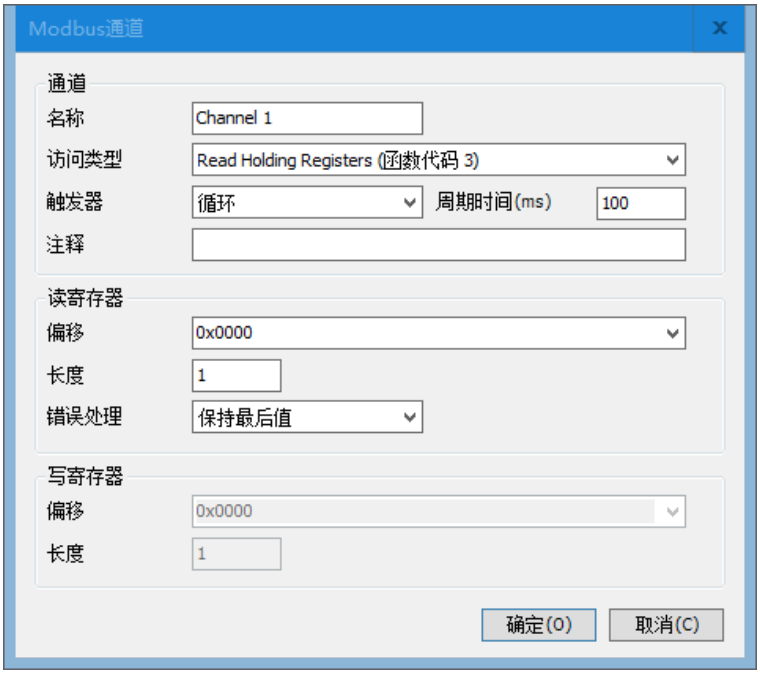


6、配置从站



7、添加并配置从站通道

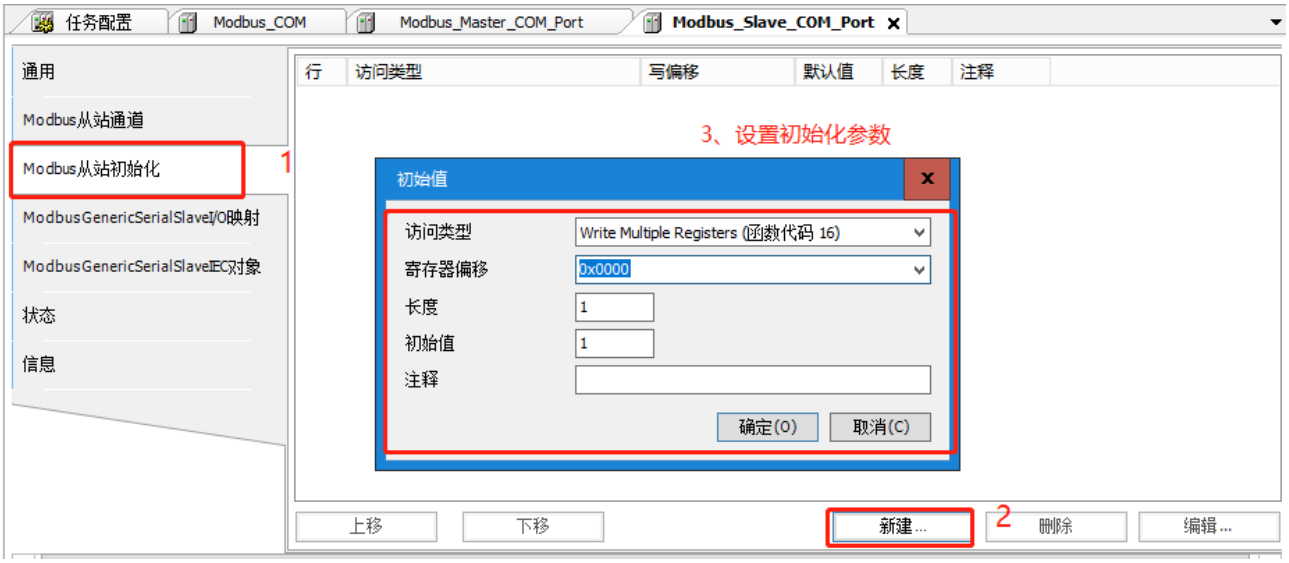




项目	说明
通道	
名称	通道名称，每一个通道代表一个 Modbus TCP 请求。

<table>
<tr><td>访问类型</td><td>
<table>
<tr><th>类型</th><th>说明</th></tr>
<tr><td>Read Coils(函数代码 1)</td><td>读线圈</td></tr>
<tr><td>Read Discrete Inputs(函数代码 2)</td><td>读离散输入</td></tr>
<tr><td>Read Holding Registers(函数代码 3)</td><td>读保持寄存器</td></tr>
<tr><td>Read Input Registers(函数代码 4)</td><td>读输入寄存器</td></tr>
<tr><td>Write Single Coils(函数代码 5)</td><td>写单个线圈</td></tr>
<tr><td>Write Single Registers(函数代码 6)</td><td>写单个保存寄存器</td></tr>
<tr><td>Write Multiple Coils(函数代码 15)</td><td>写多个线圈</td></tr>
<tr><td>Write Multiple Registers(函数代码 16)</td><td>写多个保持寄存器</td></tr>
<tr><td>Read / Write Multiple Registers(函数代码 23)</td><td>读/写多个保持寄存器</td></tr>
</table>
</td></tr>
<tr><td>触发器</td><td>触发类型：循环触发、上升沿触发、应用程序触发。</td></tr>
<tr><td>周期时间</td><td>触发器周期</td></tr>
<tr><td colspan="2">读寄存器</td></tr>
<tr><td>偏移</td><td>需要访问的从站寄存器偏移地址，偏移地址可设置。</td></tr>
<tr><td>长度</td><td>表示所读从站的数据长度</td></tr>
<tr><td>错误处理</td><td>发生错误时寄存器的值</td></tr>
<tr><td colspan="2">写寄存器</td></tr>
<tr><td>偏移</td><td>需要访问的从站寄存器偏移地址，偏移地址可设置。</td></tr>
<tr><td>长度</td><td>表示所写入从站的数据长度</td></tr>
</table>

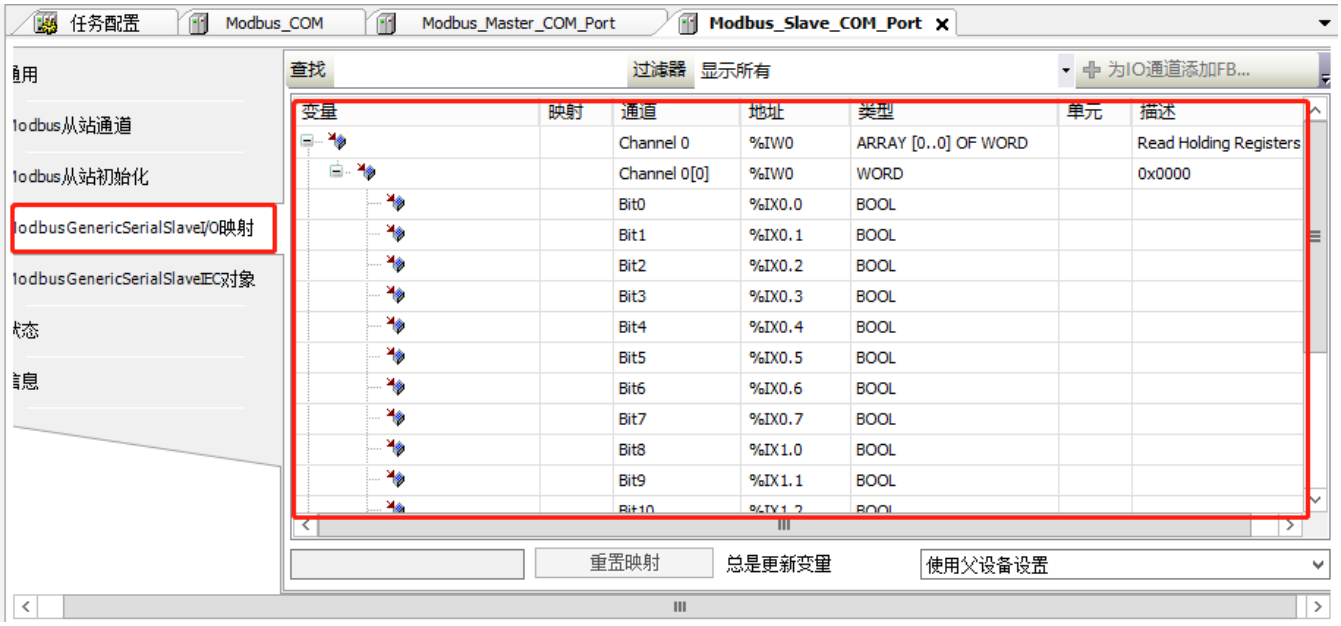
8、从站初始化



Modbus TCP 从站 IO 映射地址

在 Modbus TCP 从站通信设置中添加主从站通信配置后，Internal I/O 映射中会自动分配每条配置的映射地

址，如上图第一行的%IW0 表示将读取的一个寄存器数值映射到%IW0 这个地址。另外，还可以通过输入助手或者直接输入示例变量路径，将程序中的自定义变量映射到 I/O 地址。



4.4 Socket 通信

Socket 用于两个基于 TCP/IP 协议的应用程序之间相互通信。最早出现在 UNIX 系统中，是 UNIX 系统主要的信息传递方式。在 Windows 系统中，Socket 称为 Winsock。

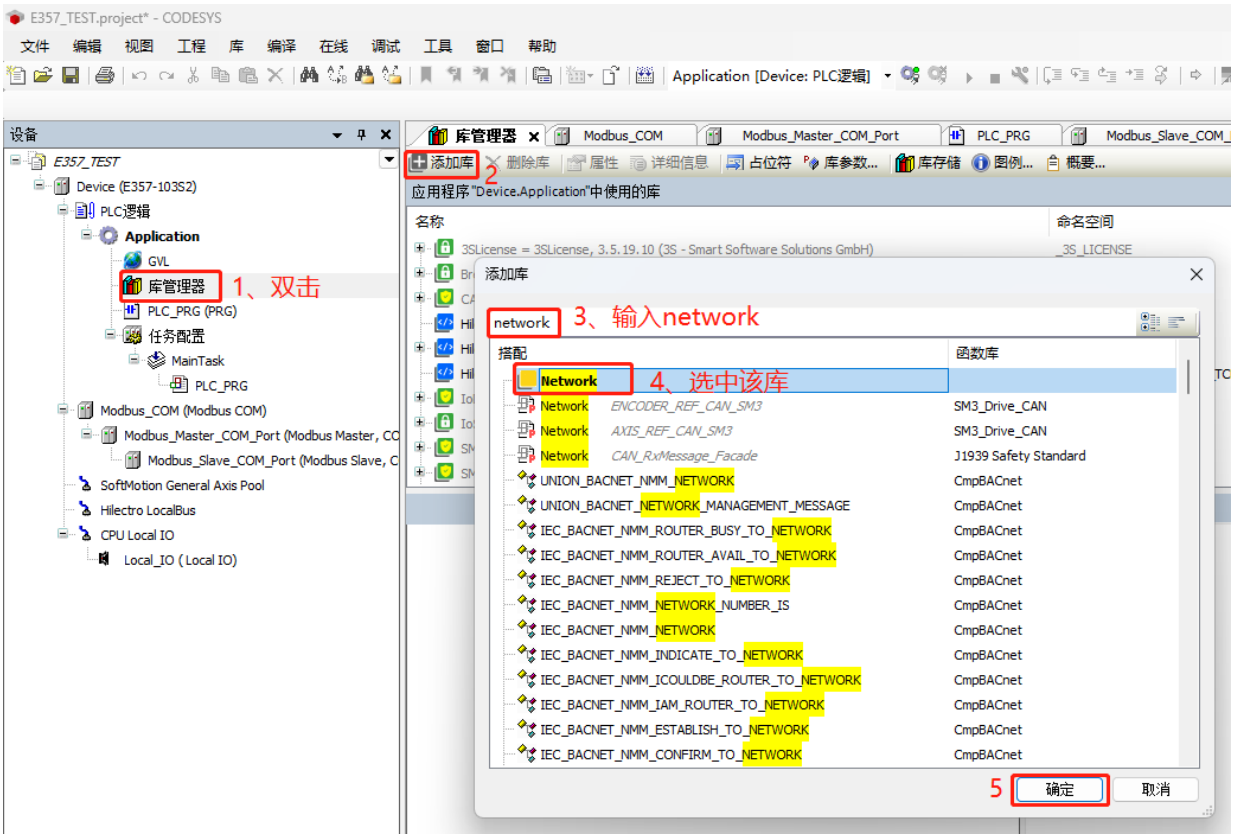
两个基本概念：

客户方和服务方。当两个应用之间需要采用 Socket 通信时，首先需要在两个应用之间（可能位于同一台机器，也可能位于不同的机器）建立 Socket 连接，发起呼叫连接请求的一方为客户方，接受呼叫连接请求的一方成为服务方。客户方和服务方是相对的，同一个应用可以是客户方，也可以是服务方。
在客户方呼叫连接请求之前，它必须知道服务方在哪里。所以需要知道服务方所在机器的 IP 地址或机器名称，如果客户方和服务方事前有一个约定就好了，这个约定就是 PORT（端口号）。也就是说，客户方可以通过服务方所在机器的 IP 地址或机器名称和端口号唯一的确定方式来呼叫服务方。在客户方呼叫之前，服务方必须处于侦听状态，侦听是否有客户要求建立连接。一旦接到连接请求，服务方可以根据情况建立或拒绝连接。连接方式有两种，同步方式（Blocking）和(noBlocking).
客户方发送的消息可以是文本，也可以是二进制信息流。当客户方的消息到达服务方端口时，会自动触发一个事件（event），服务方只要接管该事件，就可以接受来自客户方的消息了。

本次 Socket 功能演示包含作为 TCP 服务器，TCP 客户端以及 UDP 通讯。

TCP 客户端

在库管理器中添加库文件，Network。



调用 NBS.TCP_Client、NBS.TCP_Read、NBS.TCP_Write 指令，本程序实现每 500ms 自动触发一次写指令块

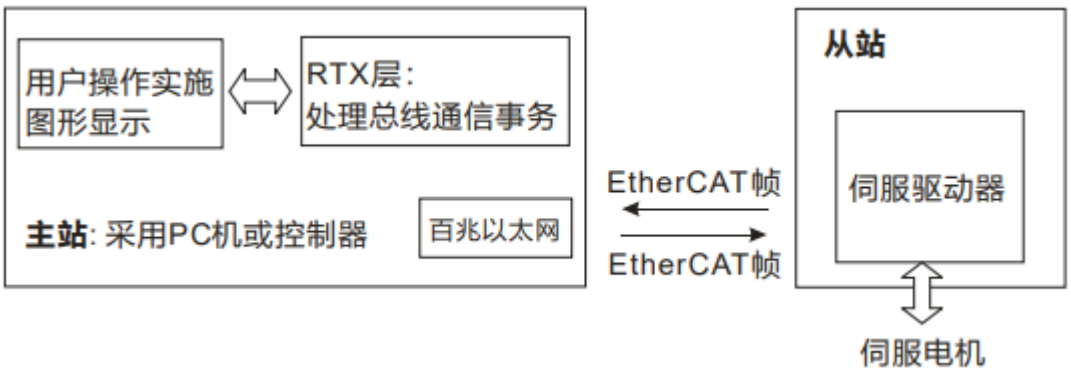
4.5 EtherCAT 通信

EtherCAT 概述

EtherCAT 是 Ethernet for Control Automation Technology 的简称。是 Beckhoff Automation GmbH 开发的实时以太网用的主站和从站间的开放网络通信，由 ETG（EtherCAT Technology Group）进行管理。

系统构成

基于 EtherCAT 的伺服控制器系统采用主从式的结构，其总体结构如下图所示：



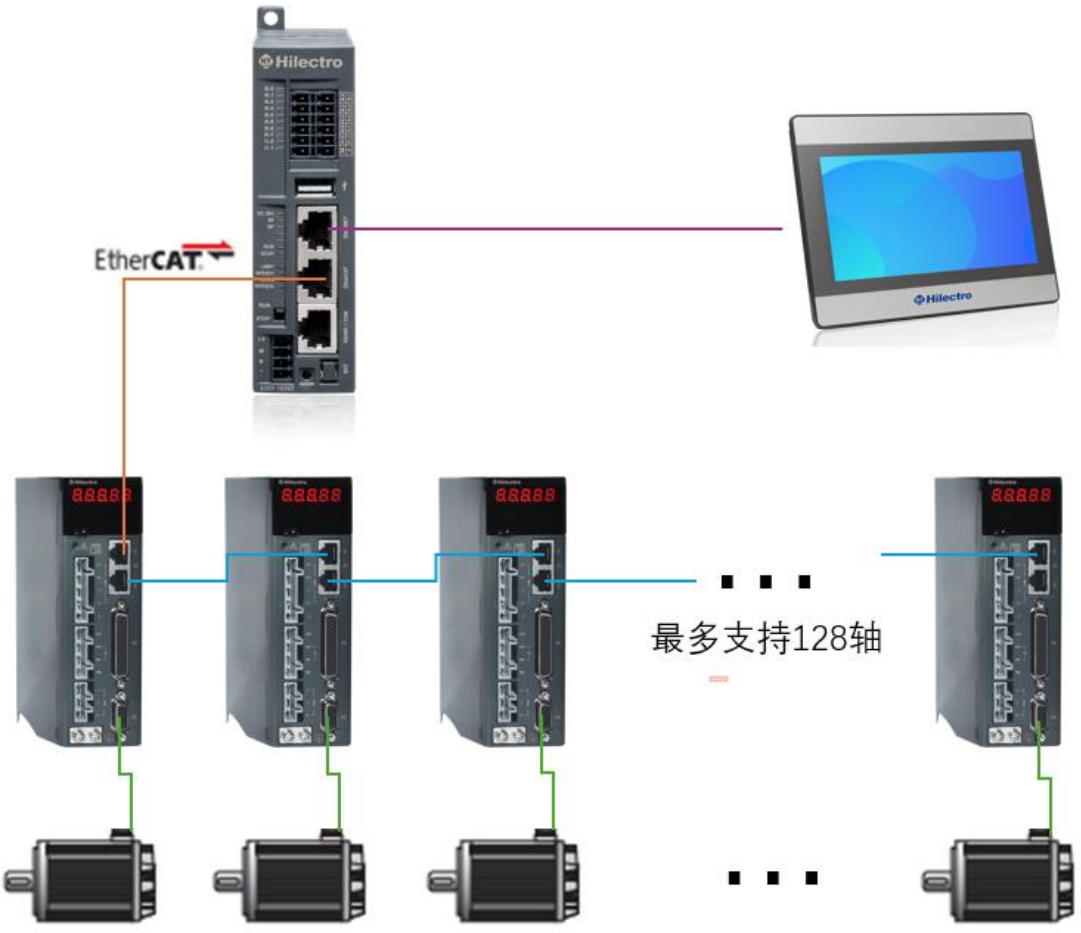
从站发起通信，实现对从站设备（伺服驱动器）以及被驱动装置（伺服电机）的监控；从站（伺服驱动器）接收主站所发控制命令，同时向主站发送被驱动装置（伺服电机）状态。

项目	规格
通讯接口	1 个 EtherCAT 通讯接口

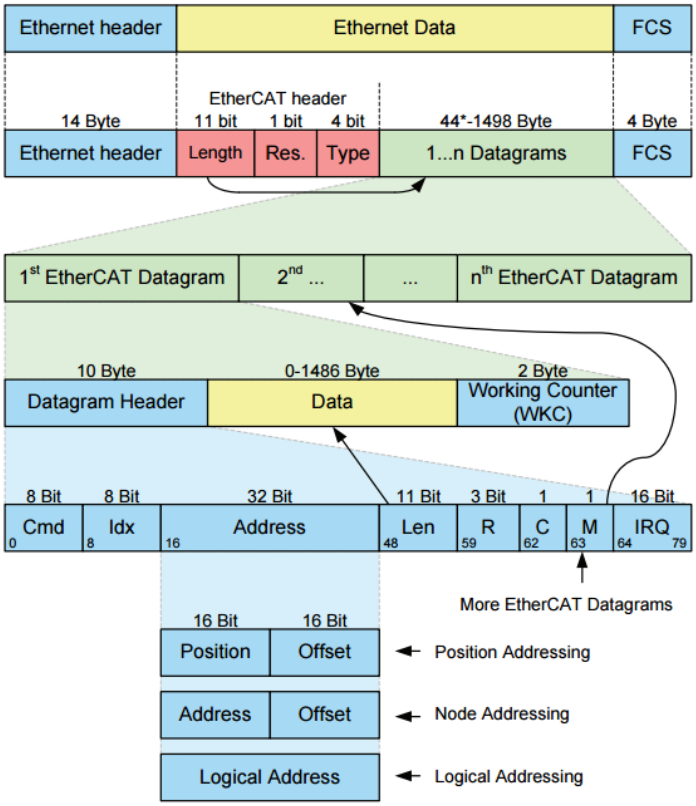
波特率	支持 100Mbps
协议类型	EtherCAT 接口协议
每段最大电缆长度	100m
最大站点数	每个主站最多支持 128 个 EtherCAT 从站
支持功能	支持分布时钟设置、冗余设置
	支持启动参数 SDO 配置
	支持配置 PDO 参数和映射
	支持配置总线循环周期、配置启动检查供应商 ID 和产品 ID
隔离	是
支持的 CiA402 控制模式	轮廓位置模式(PP) 轮廓速度模式(PV) 轮廓转矩模式(PT) 原点回归模式(HM) 同步周期位置模式(CSP) 同步周期速度模式(CSV) 同步周期转矩模式(CST)

控制器、驱动器之间可以通过廉价通用的网线进行连接，不需要繁琐的接线。

整个总线网络为线型结构，其中 E300 系列控制器为主站，海天总线控制型伺服为从站。300C 系列 PLC 带有三个网口，上面的网口为 Ethernet，用于连接 Codesys 上位机或其他以太网设备；中间的网口为 EtherCAT 连接口，用于连接 EtherCAT 从站实现 EtherCAT 通讯。海天 EtherCAT 伺服驱动器的两个通讯网口则需遵循“上进下出”的原则。



EtherCAT 帧结构

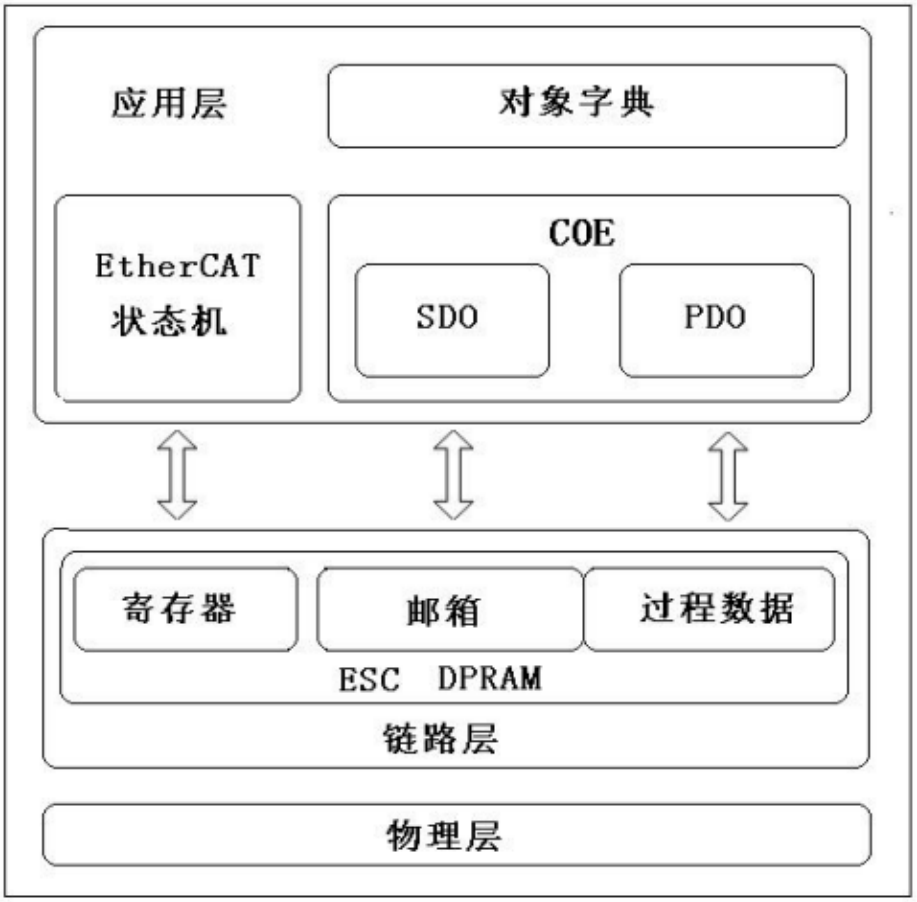


项目	数据类型	描述
Cmd	BYTE	EtherCAT 命令类型
Idx	BYTE	索引是主节点用于识别重复/丢失的数据报，不应由 EtherCAT 从站更改
Address	BYTE[4]	地址
Len	11bit	数据长度
R	3bit	保留，0
C	1bit	循环框架： 0：帧不循环 1：框架已循环一次
M	1bit	更多 EtherCAT 数据报： 0：最后一个 EtherCAT 数据报 1：更多 EtherCAT 数据报将随之而来
IRQ	WORD	所有从站的 EtherCAT 事件请求寄存器与逻辑或
Data	BYTE[n]	读/写数据
WKC	WORD	工作计数器

EtherCAT 通信结构

应用层包含了 EtherCAT 状态机，对象字典。对象字典中又包含有过程数据 PDO 映射以及 SDO 服务。PDO

过程数据是伺服控制的实时数据，主站通过周期性修改获取 PDO 数据，从而达到控制伺服的目的。SDO 是非周期性数据，主要是用于配置 PDO 映射和修改获取对象字典。



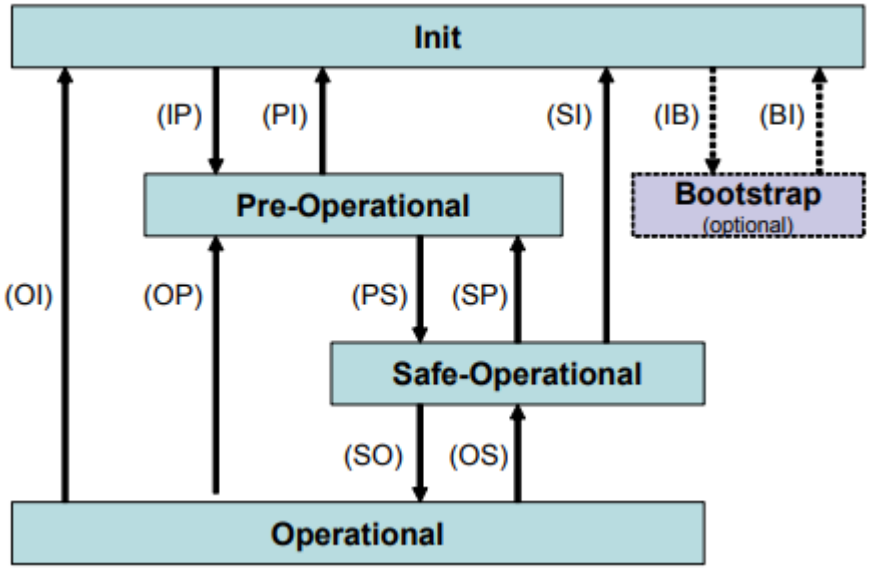
EtherCAT 状态机

以下为 EtherCAT 状态转换框图：

EtherCAT 设备必须支持 4 种状态，负责协调主站和从站应用程序在初始化和运行时的状态关系。

Init：初始化，简写为 I； Pre-Operational：预运行，简写为 P；

Safe- Operational：安全运行，简写为 S； Operational：运行，简写为 O



状态	说明
Init	设备初始化，无法启动SDO和PDO
Pre-Operational	当前状态可以使用SDO
Safe- Operational	可以读取PDO输入数据(TxPDO) 不能接收PDO输出数据(RxPDO) 可以使用SDO服务
Operational	进行周期性通信数据TxPDO/RxPDO

状态切换	说明
IP	开始SDO通信
PI	中断SDO通信
PS	开始更新输入数据TxPDO
SP	终止更新输入数据TxPDO
SO	开始更新输出数据RxPDO
OS	终止更新输出数据RxPDO
OP	终止更新输入TxPDO/输出数据RxPDO
SI	终止更新输入数据TxPDO及SDO通信
OI	终止更新输入TxPDO/输出数据RxPDO及SDO通信

启动参数 SDO/过程数据 PDO

SDO（Service Data Object）的数据交换使用邮箱通信，所以 SDO 的数据刷新时间变得不稳定。服务数据对象 SDO 配置项目一般用于系统运行之初对从站设备的功能码初始化，运行过程中也可以通过 SDO_READ 之类的功能块进行参数的访问，其通信及时性会比较低，而且会占用额外的 EtherCAT 通信开销，对于总线负载率比较高的应用中，还会引起同步超时故障，使用要慎重； PDO(Process Data Object)是实时过程数据传输，分为 TxPDO 和 RxPDO。TxPDO 是驱动器反馈数据到主站，RxPDO 是主站发送控制参数命令道驱动器。

	发信侧	收信侧
RxPDO	主站	从站
TxPDO	从站	主站

PDO 映射对象

驱动器提供了各 4 组 PDO 配置，每组 PDO 均可自由配置，每个 PDO 可以最多配置 20 个对象。

其中 0x1600~0x1603 是配置 RPDO，0x1A00~0x1A03 是配置 TPDO。

使用 CIA402 轴和不使用 CIA402 轴对象字典映射参数不同

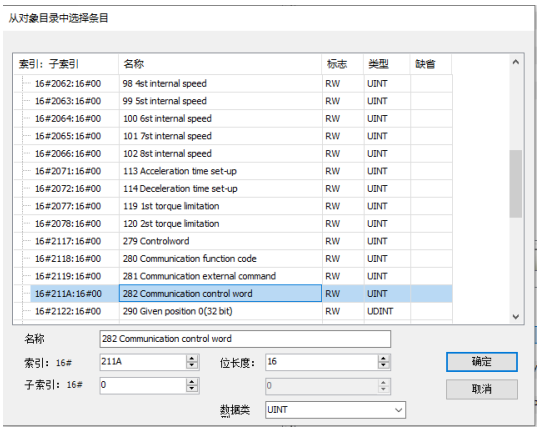
过程数据 PDO

不使用 CIA402 轴 PDO 映射关系（海天伺服）：

参数号	Index(主索引)	sub-index（子索引）
P0~P641	16#2000+16#参数号（参数号要转为 16 进制）	0

例如：P282 转换为 16 进制为 011A，加上 16#2000 为 16#211A，所以 P282 的主索引为 16#211A，具体示例如下：

参数号	Index(主索引)	sub-index（子索引）
P282	16#211A	0
P290	16#2122	0
P97	16#2061	0



使用 CIA402 轴 PDO 映射关系：

RxPDO	对象字典
0x1600	0x6040 控制字
	0x607A 给定位置
	0xYYYY
TxPDO	对象字典
0x1A00	0x6041 控制字
	0x6064 反馈实际位置
	0xYYYY

对象字典

Index	Sub	名称	位长度	数据类型	读写类型	PDO
603Fh	00h	错误信息	16	UINT	ro	TxPDO
6040h	00h	控制字	16	UINT	rw	
6041h	00h	状态字	16	UINT	ro	TxPDO
6060h	00h	控制模式	16	UINT	rw	
6061h	00h	实际控制模式	16	UINT	ro	TxPDO
6063h	00h	内部反馈位置	32	UDINT	ro	TxPDO
6064h	00h	反馈位置	32	UDINT	ro	TxPDO
606Bh	00h	速度指令	16	UINT	ro	TxPDO
606Ch	00h	实际反馈速度	32	UDINT	ro	TxPDO
6071h	00h	目标转矩	16	UINT	ro	TxPDO
6072h	00h	转矩限制	16	UINT	rw	
6074h	00h	转矩指令	16	UINT	ro	TxPDO
6077h	00h	实际转矩	16	UINT	ro	TxPDO
607Ah	00h	目标位置	32	UDINT	rw	
607Fh	00h	最大速度限制	32	UDINT	rw	

Index	Sub	名称	位长度	数据类型	读写类型	PDO
6080h	00h	电机最大速度限制	32	UDINT	rw	
6081h	00h	轮廓速度	32	UDINT	rw	
6083h	00h	加速度	32	UDINT	rw	
6084h	00h	减速度	32	UDINT	rw	
6085h	00h	急停加速度	32	UDINT	rw	
6098h	00h	回原方法	16	UINT	rw	
6099h	01h	回原搜索速度	32	UDINT	rw	

6099h	02h	回原爬行速度	32	UDINT	rw	
60B8h	00h	探针控制字	16	UINT	rw	
60B9h	00h	探针状态字	16	UINT	ro	TxPDO
60BAh	00h	探针 1 上升沿触发位置	32	UDINT	ro	TxPDO
60BBh	00h	探针 1 下降沿触发位置	32	UDINT	ro	TxPDO
60BCh	00h	探针 2 上升沿触发位置	32	UDINT	ro	TxPDO
60BDh	00h	探针 2 下降沿触发位置	32	UDINT	ro	TxPDO
60FDh	00h	数量输入	32	UDINT	ro	TxPDO
60FFh	00h	目标速度	32	UDINT	rw	

控制字 6040H(Control word)

该对象是主控器单元控制伺服的运行状态的命令字，用于控制伺服的使能、启停、告警复位等运行状态，这是最基本的控制命令字，因此，6040H（Control word）是 PDO 配置表的一个必选项。（所有模式）

bit	名称	描述
0	伺服准备好	1-有效，0-无效
1	接通主回路电	1-有效，0-无效
2	快速停机	1-有效，0-无效
3	伺服运行	1-有效，0-无效
4~6		与各伺服运行模式相关
7	故障复位	对于可复位故障和警告，执行故障复位功能 bit7 上升沿有效；bit7 保持为 1，其他控制指令均无效。
8	暂停	各模式下的暂停方式请查询对象字典 605Dh
9~10	NA	预留
11~15	厂家自定义	预留，未定义

控制字的每一个 bit 位单独赋值无意义，必须与其他位共同构成某一控制指令。bit0~bit3 和 bit7 在各伺服模式下意义相同，必须按顺序发送命令，才可将伺服驱动器按照 CiA402 状态机切换流程引导入预计的状态，每一命令对应一确定的状态。bit4~bit6 与各伺服模式相关（ 请查看不同模式下的控制指令 ）

目标位置 607AH（Target Position）

由主控器发送的伺服运行的目标位置命令。伺服多数情况下以位置模式运行，300C 的 MC 应用中，多以周期同步位置模式（CSP），控制器命令伺服在下一 EtherCAT 周期运行的目标位置，其量纲为用户设置的物理量纲。在对应轴数据结构变量 Axis.fSetPosition 可监控得到。因此，目标位置对象 607AH（Target Position）是 PDO 配置表的一个必选项。（PP/CSP 模式）

设置轮廓位置模式与周期同步位置模式下的伺服目标位置：

6040h 的 bit6	描述

0	607A 是当前段的目标绝对位置；当前段定位完成后，位置反馈 6064 = 607A
1	607A 是当前段的目标增量位移；当前段定位完成后，位置反馈增量 = 607A

伺服预运行模式 6060h（Modes of operation）

主控制器可通过对象 6060h 对伺服运行状态进行设置，选择运行（所有模式）

设定值	伺服模式	300C 系列 PLC 支持
0X00	NA	
0X01	PP（轮廓位置模式）	支持
0X02	NA	
0X03	PV（轮廓速度模式）	支持
0X04	PT（轮廓转矩模式）	支持
0X05	NA	
0X06	HM（回原模式）	支持
0X07	IP（插补模式）不支持	-
0X08	CSP（周期同步位置模式）	支持，默认运行模式
0X09	CSV（周期同步速度模式）	支持
0X0A	SCT（周期同步转矩模式）	支持

伺服运行模式切换时的注意事项：

伺服驱动器处于任何状态下，从轮廓位置模式或周期同步位置模式切入其他模式后，未执行的位置指令将被抛弃。

伺服驱动器处于任何状态下，从轮廓速度模式、轮廓转矩模式、周期同步速度模式、周期同步转矩模式切入其他模式后，首先执行斜坡停机，停机完成后，可切入其他模式。

伺服处于回零模式，且正在运行时，不可切入其他模式；回零完成或被中断（故障或使能无效）时，可切入其他模式伺服运行状态，从其他模式切换到周期同步模式下运行时，请间隔至少 1ms 再发送指令，否则将发生指令丢失或错误。

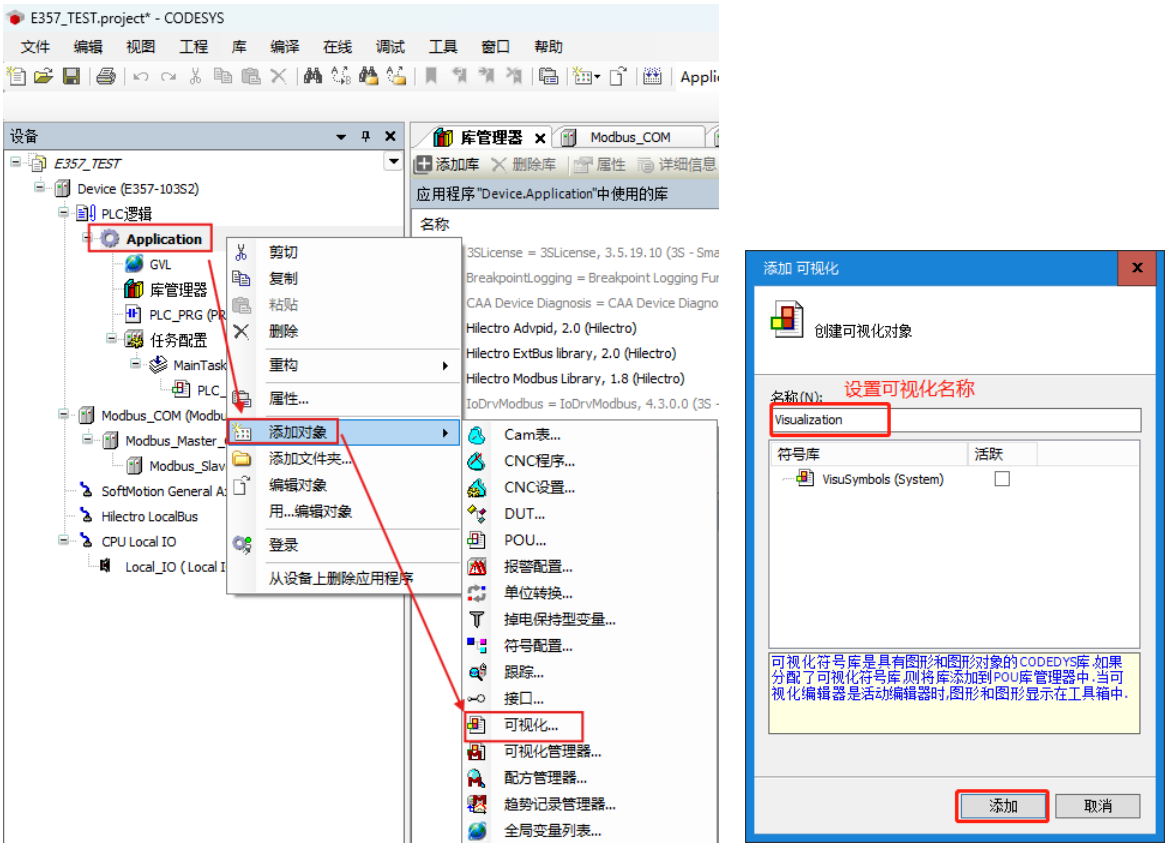
5 可视化

5.1 可视化概述

CODESYS 为用户提供了可视化以便用户实现模拟、操作或监视机器或设备。

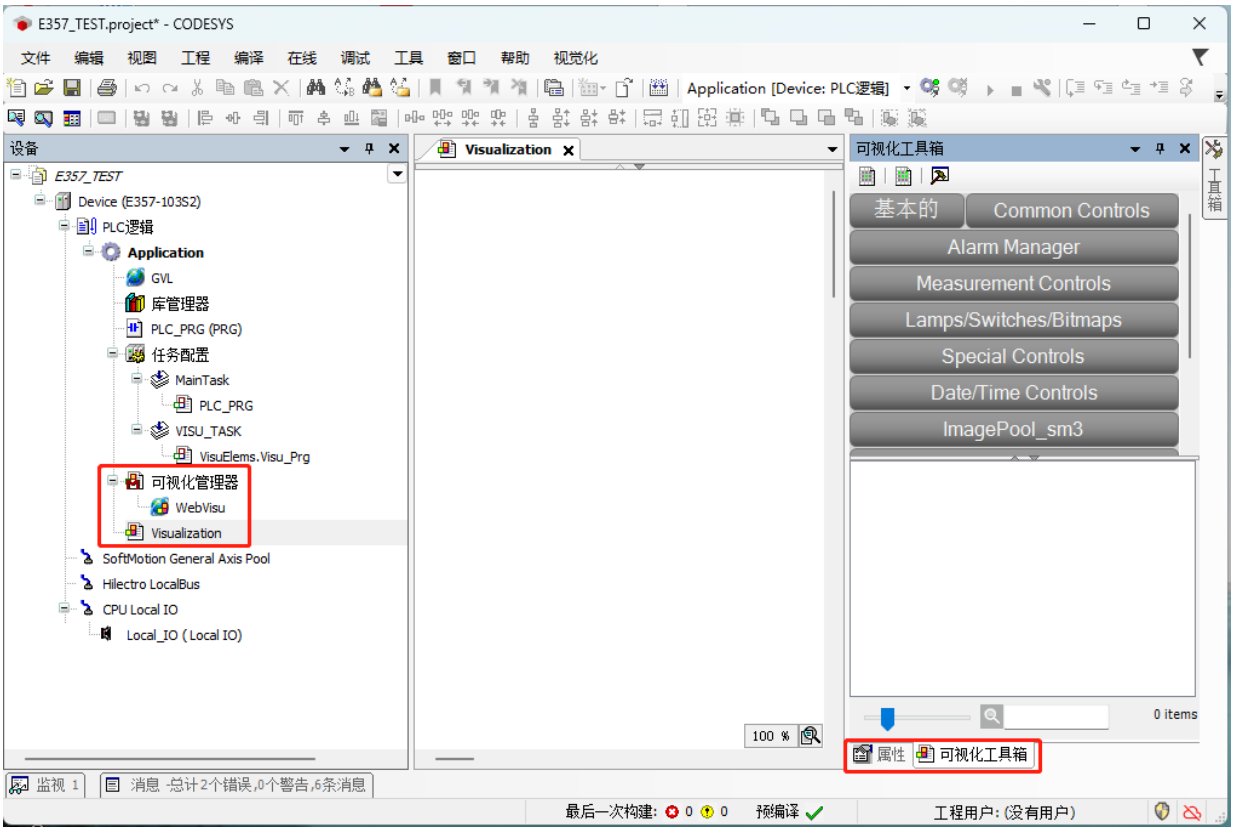
5.2 新建视图

用户可以通过在树形菜单“Application”右击找到“添加对象”选择“可视化”，给定可视化对象名称后，单击“打开”添加新的可视化。操作如下：



新建的可视化包括一个可视化编辑器和视图管理器，可视化界面介绍如下：

项目	说明
可视化管理器	可以对可视化相关参数进行配置
WebVisu	通过网页服务器链接到相关的应用程序
Visualization	可视化编辑器，可以在这个页面中编辑需要的可视化界面
可视化工具	提供可视化编辑器页面下的各种图形
属性	编辑可视化元素相关属性



5.3 基本操作

5.3.1 添加或删除视图元素

添加视图元素

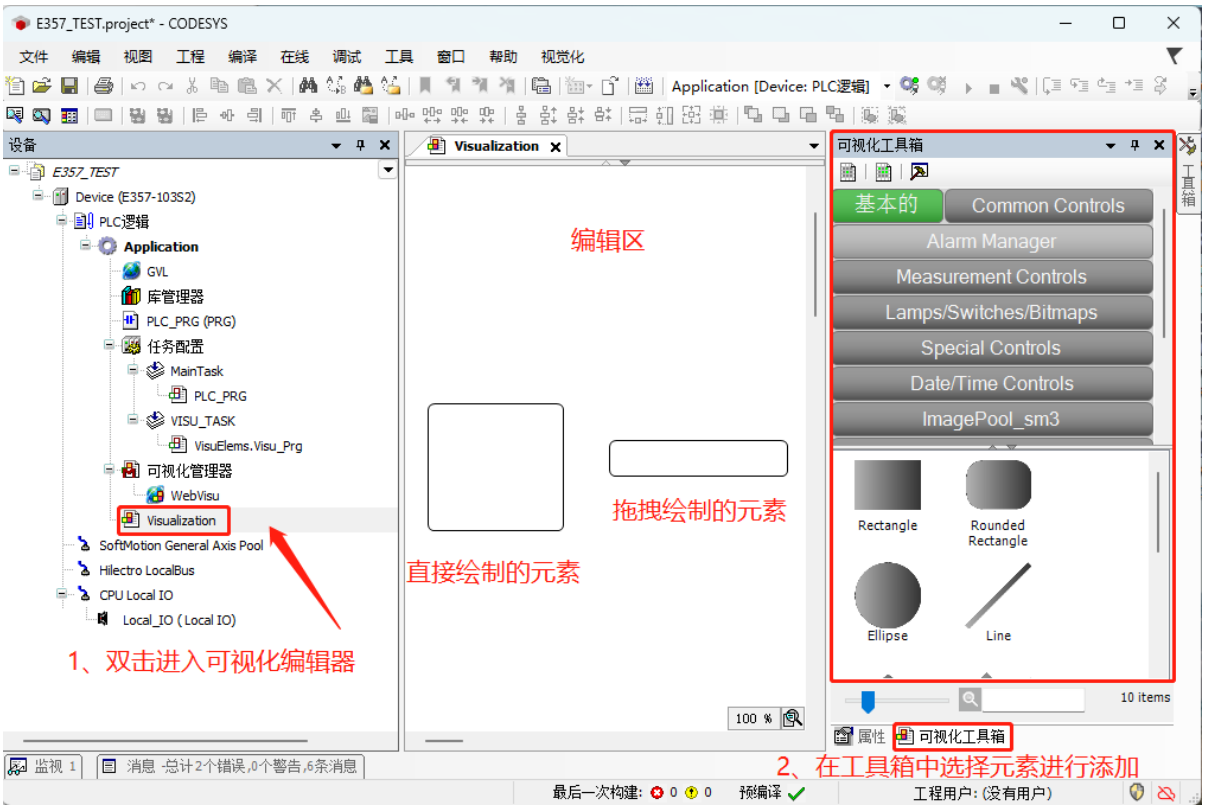
两种方式添加视图元素：

1、在编辑区直接绘制视图元素

在工具箱中选择需要添加的视图元素，然后鼠标移动到画面编辑区，视图元素会按照指定的大小和位置添加到编辑区中。

2、拖拉视图元素到编辑区

在工具箱中选择要添加的视图元素并将其拖拽至画面编辑区上，视图元素会按照默认大小添加到画面编辑区指定区域。



删除视图元素

两种方式添加视图元素：

1、选中需要删除的视图元素，单击鼠标右键→选择“删除”。

2、直接选中视图元素，通过键盘上的<Delete>删除。

5.3.2 对齐视图元素

当用户在画面编辑区添加了一组视图元素后，如果需要将这些元素对齐，首先需要选定主导元素（首先被选定的元素就是主导元素），对齐后的最终位置取决于主导元素的位置，再选择菜单中的对齐方式，可以通过鼠标右击选择“对齐”后选择不同对齐方式，也可以在工具栏直接选择对齐方式，如下：

图标	操作	描述
	左对齐	将选定的视图元素沿它们的左边对齐
	水平居中对齐	将选定的视图元素水平居中对齐
	右对齐	将选定的视图元素沿它们的右边对齐
	上对齐	将选定的视图元素沿它们的上边对齐
	垂直居中对齐	将选定的视图元素垂直居中对齐
	下对齐	将选定的视图元素沿它们的上边对齐
	使水平间距相等	使选定的视图元素水平间距相等
	增加水平间距	增加选定视图元素的水平间距
	降低水平间距	减少视图元素之间的水平距离
	删除水平间距	删除选定视图元素的水平间距，使元素之间没有间隙

	使垂直间距相等	使选定的视图元素水垂直距相等
	增加垂直间距	增加选定的视图元素垂直间距
	减少垂直间距	减少选定视图元素的垂直间距
	删除垂直间距	删除选定的视图元素垂直间距
	保持相同宽度	使选定的视图元素宽度相等
	保持相同高度	使选定的视图元素高度相等
	保持相同大小	使选定的视图元素大小相等

5.3.3 更改视图顺序

添加了视图元素后，如果需要组合成图形，可能会涉及到多个图形叠加。叠加过程中，后面添加的图形默认遮挡前面创建的图形，相当于每个图形不在同一个图层。更改图层顺序时，先选择对象元素，然后在工具栏选择需要执行的图形顺序即可。

图标	操作	描述
	置首	将选中的视图视图元素置于画面最上层
	将一个置首	将选中的视图视图元素在画面中上移一层
	将一个置尾	将选中的视图视图元素置于画面最底层
	置尾	将选中的视图视图元素在画面中下移一层

5.3.4 调整视图元素大小和位置

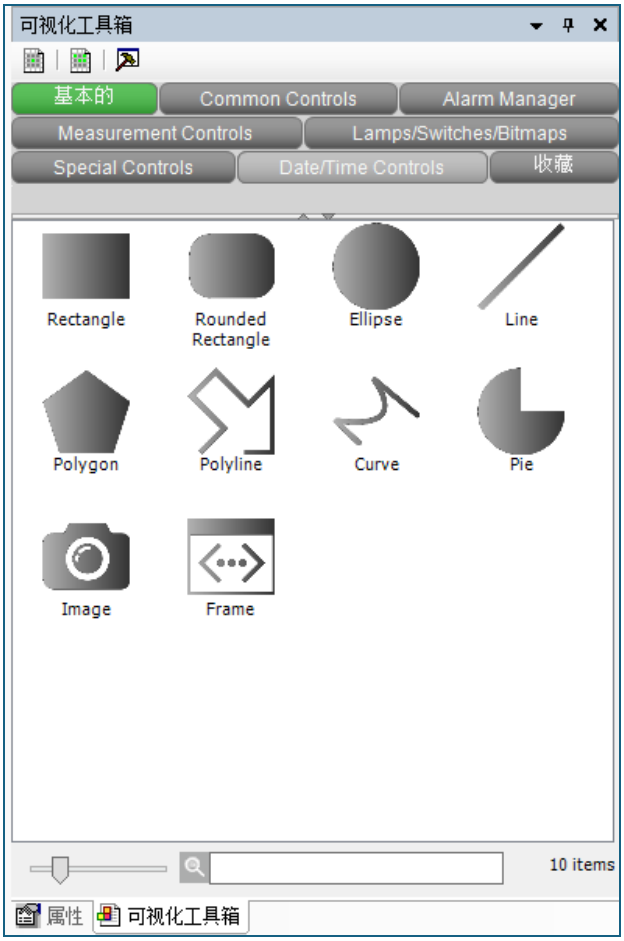
对于已经添加的视图元素，用户如果需要调整视图元素的大小，只需要在选中视图元素后通过元素周围提供的调整框直接调整到合适大小即可。但是上述调整方式无法满足用户对于元素大小的严格要求，例如需要严格按照设定的数值进行规范，此时用户可以选择视图编辑页面上方提供的向下三角包将“元素列表”调用出来，在元素列表中，当前画面中的所有使用到的元素都会列出，用户可以在这个列表中给定当前元素的“X 轴”和“Y 轴”位置进行精确定位，给定当前元素的“宽度”和“高度”来设定元素大小，也可以根据需求修改元素名称。

调整视图元素的大小

调整视图元素大小的方式有两种，一种是直接在视图编辑区选中视图元素，此时会出现可调节大小的编辑点，直接用鼠标拉动编辑点就可以调节试图元素大小；另一种是直接调节元素列表中的宽度高度。

调整视图元素位置

5.4 工具箱



5.4.1 基本控制工具

基本工具主要包括一些常用的图形制作元素，如下，可以用这些元素制作文本输入框/文本显示框/颜色显示框及图形。

文本输入框/文本显示框

在 CODESYS 的视图中，没有独立的文本输入框/文本显示框，如果需要制作文本或者数值的显示/输入框，可以通过添加“矩形/圆角矩形/椭圆”框来实现。

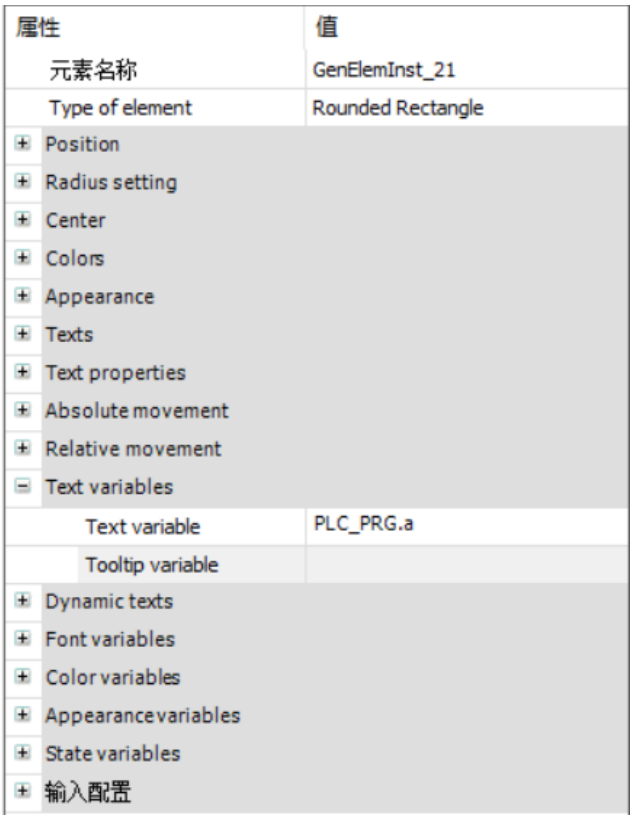
1、创建“文本显示框”

如果需要实现文本的显示，需要用户完成两个步骤：

第一，建立变量映射关系；

第二，对需要显示变量的变量类型进行设置；

首先需要建立变量映射关系，选中添加的视图元素（矩形/圆角矩形/椭圆），在右侧属性菜单中会显示当前视图元素支持的属性配置，通过设置属性菜单中的“文本变量”，单击“输入助手”图标按钮进行相关变量映射，设置完成后点击确认即可。



然后需要配置显示类型，对于常量的显示，只需要在“文本”框中输入对应需要显示的文本或者数值即可，对于非固定常量的显示，需要根据具体的数据类型进行配置，例如需要显示的文本为字符串类型，则需要在“文本”框中输入“%s”，除了“s”之外，还可以使用标准 C 库函数 sprintf 中其他格式化输出命令。

格式化输出命令	描述
d、i	十进制数
o	无符号八进制数
x	无符号十六进制数
u	无符号十进制数
c	单个字符
s	字符串
f	实数

注意：

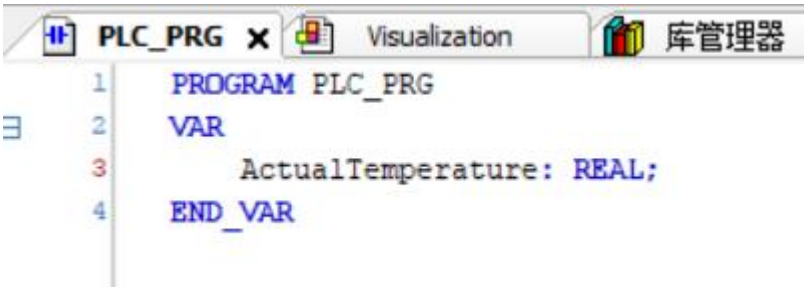
如果想显示一个百分号（%）并与上面提到的格式化输出命令进行组合，则必须输入“%%”，如输入“Rate in %%：%s”，则在线模式下，将显示“Rate in %：12”（关联文本显示的变量当前值为“12”）。

其中，输入的格式化输出命令区分大小写，需为小写，否则无法正常识别。

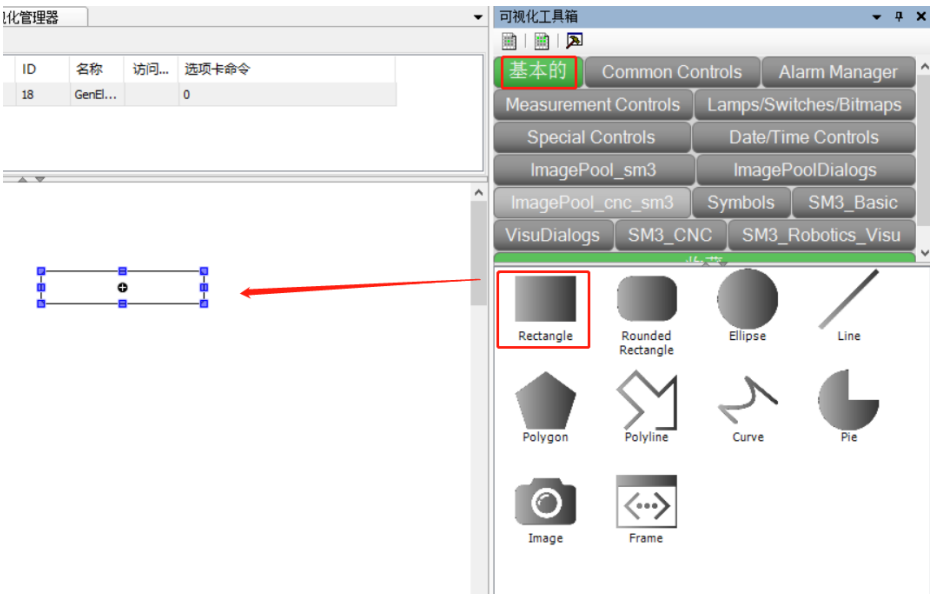
浮点型数据如果要约定小数点之后的个数，需要按照%<对齐格式><最小宽度>.<精度>f 的格式输入，比如小数点后两位则为：%.2f，在线模式下显示：1.02（关联文本显示的变量当前值为 1.02344…）

【例 1】 创建一个文本显示框，用于显示室内的实际温度。

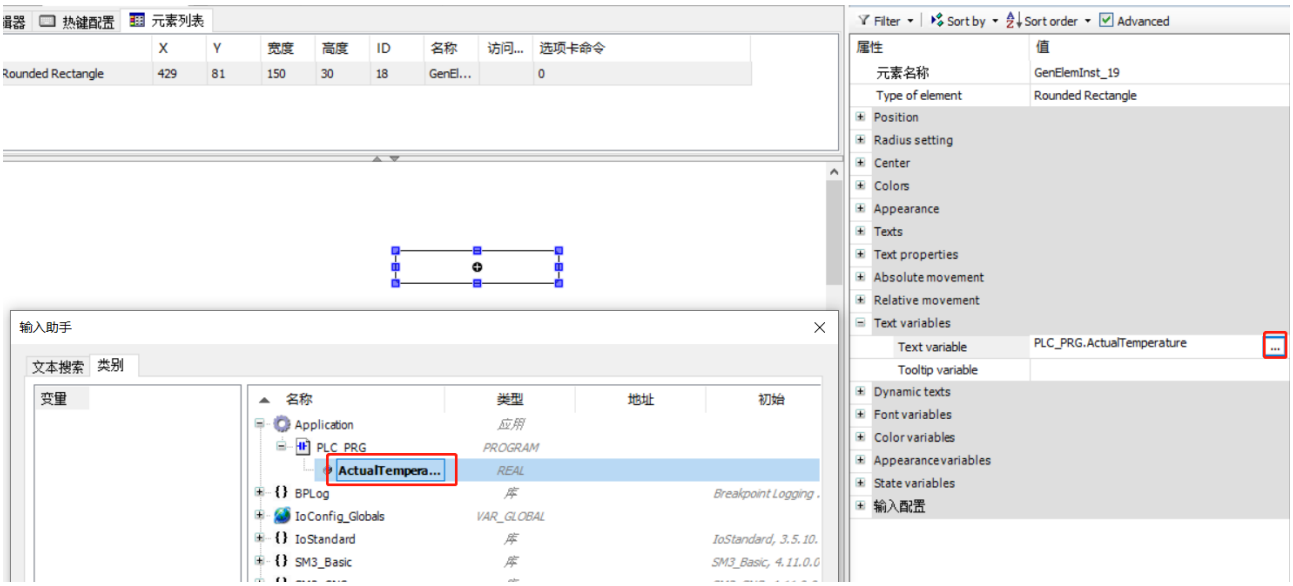
1.在程序中添加需要显示的温度变量



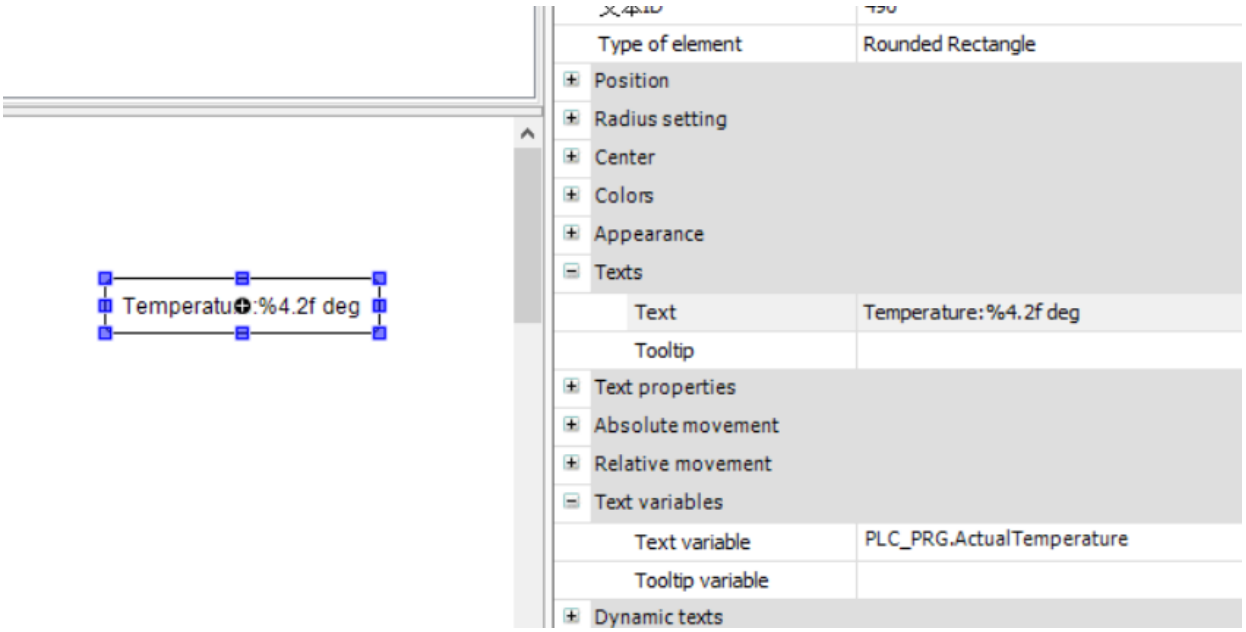
2.从工具箱的“基本的”中选择矩形框拖拽到视图编辑区



3.选中矩形框后，在右侧“属性”菜单中找到“文本变量”完成变量映射



4.配置显示类型，在右侧“属性”菜单中找到“文本”输入“Temperature:%4.2f deg”其中“%4.2f”设置的是浮点型数据的显示，表示，保留宽度为 4 位，小数点后 2 位的实数，其余均为固定文本。



在线运行结果

5.4.2 Common Controls 通用控制工具

5.4.3 Alarm Manager 报警处理

5.4.4 Measurement Controls 测量控制工具

5.4.5 Lamp/Switch/Bitmaps 灯，开关，位图工具

5.4.6 Special Controls 特殊控制工具

5.4.7 Data/Time Conrtols 日期，时钟控制工具

附录

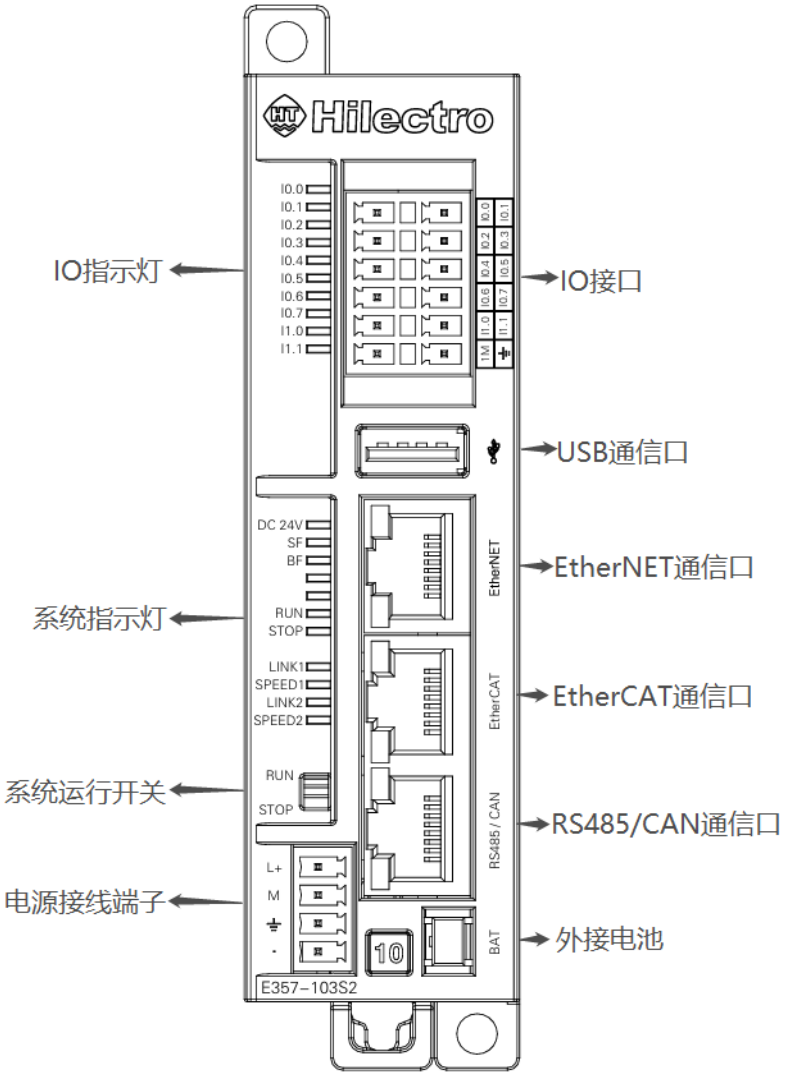
6

6.1 PLC 介绍

E300 系列运动控制器是一款基于嵌入式系统的中央处理单元，它同时具备运动控制和逻辑控制功能。E300 系列运动控制器及其扩展单元，通过各种运算的处理以及用户程序的运行，对外设进行访问和控制以实现用户所需的机器动作。

6.1.1 PLC 外观

E357-103S2 外观示意图如下：



6.1.2 E357 PLC 规格参数

表6-1 常规规格

物理特性	
尺寸（W×H×D）	34×115×101.6 mm
功耗	19.2W

内存		
用户程序空间		32MB
掉电保持空间		E357：总共 64KB。 分为： 内存 1（32KB）：GVL 全局变量里面的 RETAIN（保留型变量） 内存 2（32KB）：PersistentVars（持续型变量） 以上两个内存分区需支持条件：固件 V2.0、描述文件 V2.0。
电源特性		
额定输入电压		24V DC
输入电压范围		20.4V~28.8V DC
输入电流		0.8A
极性反接保护		有
总线电源电压		+5V DC
总线电源电流		1.6A
隔离		外部电源与系统电源之间隔离
LED 指示灯特性		
24V 电源指示灯	绿色	亮起：24VDC 供电正常，熄灭：无 24V DC 供电
SF 指示灯	红色	亮起：系统故障，熄灭：无错
BF 指示灯	红色	亮起：总线故障，熄灭：无错
RUN 指示灯	绿色	亮起：系统运行，熄灭：系统停止
STOP 指示灯	橙色	亮起：系统停止，熄灭：系统运行
RJ45 网口指示灯	绿色	闪烁：连接，熄灭：未连接
	黄色	亮起：100Mbps，熄灭：10Mbps
CAN 口指示灯	绿色	数据接收指示灯
	黄色	数据发送指示灯
EtherCAT 口指示灯		
Link1 指示灯	绿色	亮起：连接；熄灭：未连接
SPEED1 指示灯	黄色	亮起：100Mbps；熄灭：10Mbps
Link2 指示灯	绿色	亮起：连接；熄灭：未连接
SPEED2 指示灯	黄色	亮起：100Mbps；熄灭：10Mbps
IO 指示灯		
I0.0~I1.1	绿色	亮起：有信号输入；熄灭：无信号输入
实时时钟		

精度	每月偏差<60 秒
掉电保持时间	掉电保持时间约 112 小时（典型值） 外接电池后，掉电保持的时间为典型值 2 年以上
读取/设置实时时钟	通过指令 setrtc 读取/设置
保护功能	
电源保护	供电电源端提供反接保护功能及浪涌吸收功能
接口保护	通讯端口防雷保护
扩展 I/O 能力	
1 个 CPU 支持 INT-00 机架数	4
每机架支持最多模块数	主机架：11 个（电源模块，CPU，中继模块，8 个信号模块）， 从机架：10 个（电源模块，中继模块，8 个信号模块）
1 个 CPU 支持 ECT-00 从站数	最多 128 个
每 ECT-00 支持模块数	最多支持 8 个模块
隔离	
电源隔离	外部电源与系统电源之间隔离
通讯隔离	以太网、RS485 隔离、CAN 隔离
编程软件	
编程软件包	CODESYS V3.5（SP19 版本）
编程语言	符合 IEC61131-3 的编程语言：CFC/FBD/LD/IL/ST/SFC
运动控制器功能	
支持的运动控制器功能	SoftMotion 和 CNC

表6-2 通讯口规格

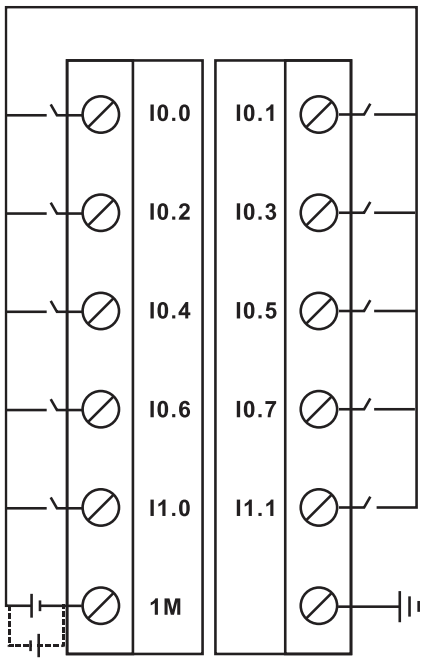
EtherNET 通讯口	
通讯接口	标配1个以太网口
协议类型	CODESYS 定制协议，基于 UDP 和 TCP（支持以太网编程和 CPU 间以太网通信），socket、Modbus_TCP、EtherNET IP 协议。
每段最大电缆长度	100m
一个站点最大连接数	UDP 最多支持16个连接，Modbus TCP 最多支持32个连接； socket 无限制、EtherNET IP 最多支持253个连接。
用户数据	最多支持240个字节用于 Modbus_TCP 传输，对于 socket，EtherNET IP 数据量传输没有限制。
波特率	10/100Mbps 自适应
隔离	通信口隔离
EtherCAT 通讯口	
通讯接口	1个 EtherCAT 接口

最大站点数	每个主站最多支持128个从站						
协议类型	EtherCAT 接口协议						
支持功能	支持分布时钟配置						
	支持启动参数配置						
	支持配置 PDO 参数和映射						
	支持配置总线循环周期						
每段最大电缆长度	100m						
波特率	10/100Mbps 自适应						
隔离	通信口隔离						
RS485通讯口							
通讯接口	1个 RS485通讯口						
最大站点数	每段32个站，每个网络126个站						
协议类型	Modbus 主站/从站协议						
Modbus 波特率	1200、4800、9600、19200、38400、57600、115200bps						
隔离	通信口隔离						
CANopen 通讯口							
通讯接口	1 个 CAN 通讯主站接口						
最大从站点数	1 个主站后面最多可连接 32 个从站						
协议类型	CANopen DS301 标准协议						
支持功能	支持自动启动 CANopen Manager						
	支持可选从站轮询						
	支持启动从站						
	支持 NMT						
	支持同步生产						
	支持同步消耗						
	支持心跳产生						
	支持激活时间创建						
传输速率（kbit/s）	1000	800	500	250	125	50	20
最大长度（m）	25	50	100	250	500	1000	2500
隔离	通信口隔离						

E357 运动控制器数字量输入特性

数字量输入特性		
本机集成 IO 点数		10
输入类型		漏型/源型
额定电压		24V DC
输入电压范围		20.4~28.8V DC
浪涌电压		35V DC，持续0.5s
逻辑1信号 (最小) 逻辑0信号 (最大)		15 VDC，2.5mA 5 VDC，1mA
连接2线接近开关传感器（BERO）		1mA（允许的最大漏电流）
输入滤波		可配置，支持0.2us，0.4us，0.8us，1.6us，3.2us，6.4us，12.8us、0.2ms，0.4ms，0.8ms，1.6ms，3.2ms，6.4ms，12.8ms，默认为6.4ms
隔离(现场与逻辑) 隔离组		500V AC，1分钟 见接线图
同时接通的输入		10
最大电缆长度 屏蔽 非屏蔽		500米(标准输入) 50米(高速计数器输入) 300米(标准输入)
数字量输入特性		
脉冲捕捉输入		10
高数计数器	总计	6
	单相	6×500kHz
	两相	4×250kHz

CPU 自带 IO 接线图：



E357 运动控制器高速计数器输入点

模式	描述	输入		
	HSC0	I0.0	I0.1	I0.2
	HSC1	I0.3	I0.4	I0.5
	HSC2	I0.6	I0.7	--
	HSC3	I1.0	I1.1	--
	HSC4	I0.2	--	--
	HSC5	I0.5	--	--
0	带有内部方向控制的单相计数器	时钟	--	--
1		时钟	--	复位
2		--	--	--
3	带有外部方向控制的单相计数器	时钟	方向	--
4		时钟	方向	复位
5		--	--	--
6	带有增减计数时钟的两相计数器	增时钟	减时钟	--
7		增时钟	减时钟	复位
8		--	--	--
9	A/B 相正交计数器	时钟 A	时钟 B	--
10		时钟 A	时钟 B	复位
11		--	--	--

6.1.3 PLC 接口定义

表6-3 电源接口（X4）信号定义

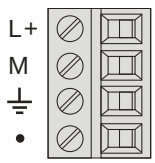
4 位可拆卸端子（X4）	符号	信号定义
	L+	24V 电源正
	M	24V 电源负
	⏚	大地
	--	--

表6-4 EtherNET/EtherCAT 通信接口信号定义

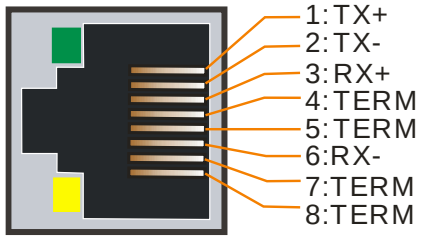
RJ45 网口	位号	信号	信号定义
	1	TX+	数据发送正端
	2	TX-	数据发送负端
	3	RX+	数据接收正端
	4	--	--
	5	--	--
	6	RX-	数据接收负端
	7	--	--
	8	--	--
	连接器外壳	PE	机壳接地

表6-5 CAN 接口定义

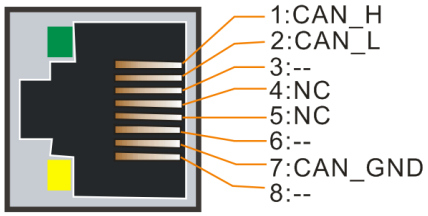
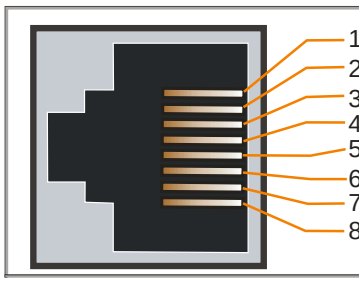
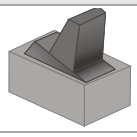
RJ45 网口	位号	信号	信号定义
	1	CAN_H	数据发送正端
	2	CAN_L	数据发送负端
	3	--	--
	4	--	--
	5	--	--
	6	--	--
	7	CAN_GND	CAN 的地
	8	--	--
	连接器外壳	PE	机壳接地

表6-6 RS485/CAN 接口定义（E357-103S2 的 RS485 与 CAN 接口为同一个接口）

RJ45 网口	位号	信号	信号定义
	1	CAN_H	数据发送正端
	2	CAN_L	数据发送负端
	3	--	--
	4	A0	RS485 信号 A
	5	B0	RS485 信号 B
	6	--	--
	7	CAN_GND	CAN/RS485 地
	8	--	--

	连接器外壳	PE	机壳接地

表6-7 系统运行开关拨码定义

系统运行开关	信号	拨码方向	信号定义
	RUN	向上	系统运行
	STOP	向下	系统停止

6.2 PLC 安装

安装环境要求：

❑隔离 PLC 与加热装置、高电压和电子噪声

安装设备器件时，产生高压高电子噪声的设备与 E300 系列 PLC 分隔开。

在控制柜的背板上安排 E300 系列 PLC 时，应考虑把电子器件安排在控制柜中温度较低的区域内，因电子器件长期在高温环境下工作会缩短其无故障时间。

要考虑控制柜的背板布线，尽量避免把交流供电线、高能量、开关频率很高的直流信号线与低压信号线、通讯电缆设计在同一个线槽中。

❑为散热和接线留出适当的空间

E300 系列 PLC 的设计采用自然对流散热方式，在模块的上下方都必须留有至少 30mm 的空间，以便于正常的散热。前面板与背板的板间距离也应保持至少 80mm。

安装注意事项

在安装和拆卸 PLC 及其相关设备之前，要确保 PLC 及与其相连设备的供电均已被切断。

警告

若未切断所有电源而在带电情况下安装或拆卸 PLC 及其相关设备有可能导致电击或是设备误动作，从而造成设备损坏或是严重的人身伤害甚至人员死亡。

在更换或安装 PLC 时，要确定使用了正确或等同的模块。在更换时，除了要使用相同的模块外，还要确保安装的方向和位置是正确的。

注意

如果您安装了不正确的模块，PLC 的程序可能会产生错误的功能。

6.1 电源模块 EM300-PWR

EM300-PWR 电源模块，专门为 E300 系列 CPU 及扩展模块（除数字量模块外）提供 24V DC 电源。每个机架请选配一个电源模块，数字量输入输出电源和传感器电源请选择其他供电电源。

表6-8 电源模块 EM300-PWR 规格

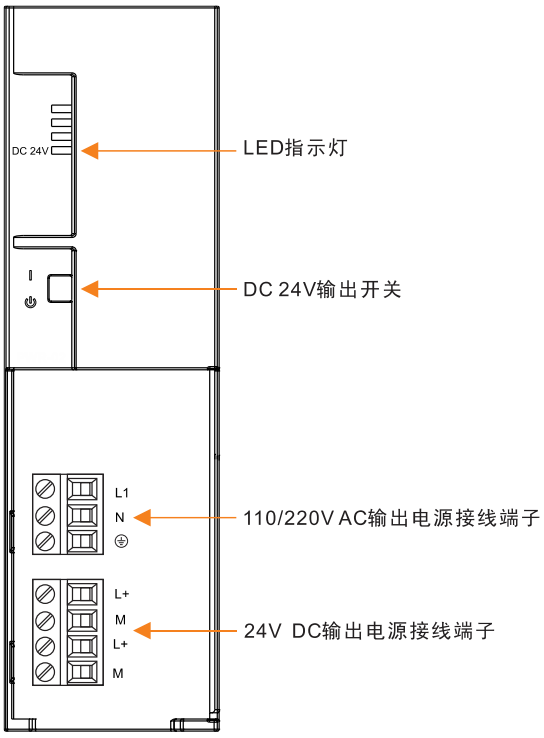
名称	规格描述	订货号
EM300-PWR 电源模块	输入：85～264V AC，输出：24V DC/2A	EM300-PWR

表6-9 电源模块 EM300-PWR 技术参数

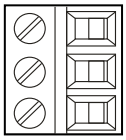
物理特性	
尺寸（W×H×D）	34×115×101.6 mm
LED 指示灯特性	
24V 电源指示灯（绿色）	亮起：有 24V DC 输出，熄灭：无 24V DC 输出
开关特性	
开关	控制 24V DC 电源输出 ON：有 24V DC 输出，OFF：无 24V DC 输出
输入电压特性	
电压范围	85～264V AC，宽电压输入
额定频率 频率范围	50Hz/60Hz 47Hz～63Hz
效率	75%
交流电流	0.9A/110V、0.5A/220V
浪涌电流（25℃最大）	≤20A/110V、≤35A/220V
泄露电流	≤5mA/220V AC
输出电压特性	
直流电压/额定电流	24V DC/2A
额定功率	48W
纹波和噪声（最大）	150mVp-p
电压输出范围	±5%
启动/上升/保持时间	≤2.5s/≤50ms/≥20ms
隔离（电源输入与输出）	110V/220V AC 与 24V DC 之间隔离
保护功能	
过载保护	105%～130%的额定输出功率，切断输出，故障排除自动恢复
过压保护	115%～135%Ue；保护方式：打嗝模式，故障排除自动恢复
浪涌保护	供电电源端提供浪涌吸收功能
过流保护	电源输出端提供过流保护
安全电磁兼容	
耐电压	输入~输出：1.5KVDC，输入-PE：1.5kV DC，输出-PE：500V DC

隔离电阻	输入~输出，输入-PE，输出-PE：100MΩ/500V DC
依据标准	安全参照UL60950和UL1950，电磁兼容参照EN55022

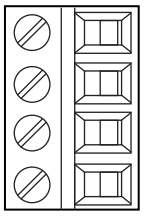
电源模块展示



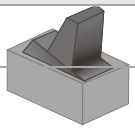
EM300-PWR 的 220V AC 输入电源接口定义

3 位可拆卸端子	信号	信号定义
	L	火线
	N	零线
	⏚	大地

EM300-PWR 的 24V DC 输出电源接口定义

4 位可拆卸端子	信号	信号定义
	L+	24V 电源正
	M	24V 电源负
	L+	24V 电源正
	M	24V 电源负

EM300-PWR 的拨码开关定义

两态开关	位号	拨码方向	信号定义
	ON	向上	有 24V DC 输出

	OFF	向下	无 24V DC 输出

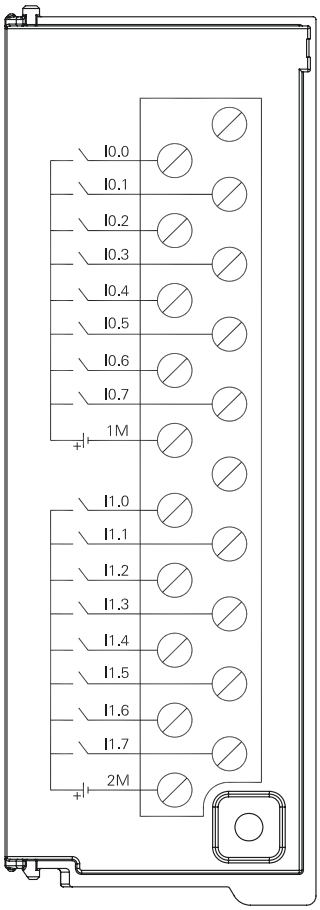
6.2 数字量模块

6.2.1 数字量输入模块

特性		EM300-DI16T
尺寸（W×H×D）		34×115×100 mm
输入点数		16
电流消耗	24V DC	4mA/通道
	+5V 总线	80mA
输入类型		漏型/源型（IEC 1 类漏型）
输入额定电压		24V DC，6mA
输入电压范围		20.4~28.8V DC
浪涌电压		35V DC，持续 0.5s
逻辑 1（最小）		15V DC，2.5mA，翻转电平：10.5V DC±15%
逻辑 0（最大）		5V DC，1mA
连接 2 线接近开关传感器（BERO）允许的漏电流（最大）		1mA
输入滤波		可配置，支持 0.2ms、0.4ms、0.8ms、1.6ms、3.2ms、6.4ms（默认）、12.8ms
输入频率		1.5kHz，占空比 50%
输入阻抗		6.6KΩ
隔离		500V AC，持续 1min
每组隔离点数		8
同时 ON 点数		16
电缆长度	屏蔽	500m
	非屏蔽	300m

接线规格

- ◆ EM300-DI16T 接线图



6.2.2 数字量输出模块

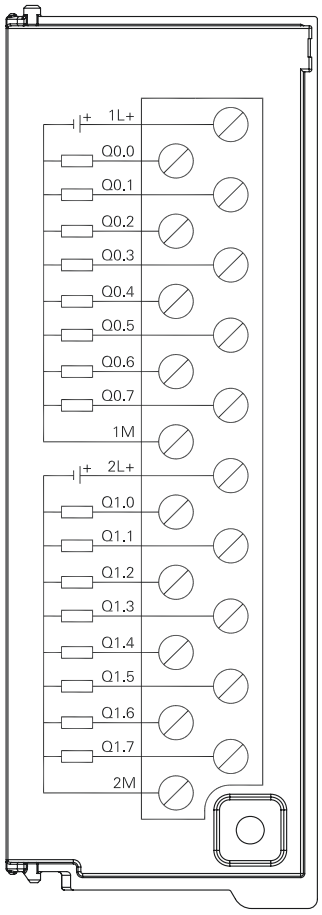
表6-10 技术规格

特性		EM300-DO16TN
尺寸（W×H×D）		34×115×100 mm
输出点数		16
电流消耗	24V DC	95mA
	+5V 总线	120mA
输出类型		固态-MOSFET，源型
输出额定电压		24V DC
输出电压范围		20.4~28.8V DC
逻辑 1（最小）		20V DC
逻辑 0（最大）		0.1V DC，10KΩ 负载
输出电流（最大）		0.5A
每个公共端电流		最大 4A
输出漏电流（最大）		15uA
浪涌电流		8A，100ms
灯负载		5W

接触电阻		0.3Ω，最大 0.6Ω
输出延迟		断开到接通（最大）：50us 接通到断开（最大）：200us
输出频率（最大）		1kHz
机械寿命（无负载）		-
触点寿命（额定负载）		-
隔离 现场到逻辑 线圈到触点		 500V AC，持续 1min -
每组隔离点数		8
同时 ON 点数		16
两个输出并联		支持同组内两个输出并联
电缆长度	屏蔽	500m
	非屏蔽	150m

接线规格

◆ EM300-DO16TN 接线图



6.3 故障诊断

当系统出现故障时，CPU 执行以下操作：

1) 进入 STOP（停止）模式

2) 点亮 SF(红)LED 指示灯和 STOP 指示灯

3) 断开输出

请先检查以下条件是否满足：

1) E300 系列主控模块及扩展模块是否正常供电。

2) E300 系列主控模块及扩展模块 I/O 端子的螺丝和接插件是否松动。

3) 检查通信电缆的连接情况，确保无误。

4) 搜索不到 PLC，请检查通信设置，例如改变波特率、连接串口或 IP 等重新搜索。

除以上方法外，还可通过 PLC 的 LED 指示灯状态检查 PLC 自身和外部有无异常。

6.3.1 通过主控模块进行诊断

表6-11 LED 状态指示灯状态描述

指示灯	颜色	描述
RUN 指示灯	绿色	亮起：系统运行，熄灭：系统停止
STOP 指示灯	橙色	亮起：系统停止，熄灭：系统运行
SF 指示灯	红色	亮起：系统故障，熄灭：无错
LINK 指示灯	绿色	点亮：网口硬件已连接 闪烁：有数据交换 熄灭：网口硬件已断开
RMC 指示灯	绿色	点亮，连接远程服务器成功 熄灭：远程未连接或远程未使能

<注>：在设备开始掉电到完全掉电的电源波动阶段，STOP 指示灯和 SF 指示灯会同时亮起，系统会记录一个事件。

6.4 指令速查

6.4.1 CODESYS 中的操作符总结

下表为操作符总结，这些操作符分别都存在于 CODESYS 的 Standard.lib 和 Util.lib 程序库中。

注意‘IL 符’栏：只有使用此符号的行才被显示。一个必要条件是：第一组必要的操作数已被顺利载入前一行（如，LD in）

‘Mod.IL’栏显示 IL 中有可能出现的修饰符：

C	前行表达式为 TRUE 的结果的修饰符。
N	对于 JMPC，CALC，RETC：如前行的结果为 FALSE，则执行此命令。

N	否则：否定操作数（非累积器）
(	括弧中操作符：只有括号出现后操作才开始执行。

操作符总结表

在 ST 中	在 AWL 中	Mod.AWL	描述
'			字符串分隔符
.. [ ]			Array 值域大小
:			声明中操作数与类型间的分隔符。
;			说明终止。
^			指示符废止。
	LD var1	N	缓冲器中变量 1 值的载入。
:=	**ST var1**	N	将实际结果存入变量 1。
	S boolvar		当实际结果为 TRUE，准确设置操作数布尔变量为 TRUE。
	R boolvar		当实际结果为 FALSE，准确设置操作数布尔变量为 TRUE。
	JMP label	**CN**	直接跳至标签。
<Program name>	**CAL** prog1	**CN**	调用程序 prog1。
<Instance name>	**CAL** inst1	**CN**	调用功能程序 inst1。
<Fctname>(vx, vy,..)	**<Fctname> vx, vy**	**CN**	调用功能 fctname 和 transmit 变量 vx，vy。
RETURN	**RET**	CN	Leave POU and go back to caller
	(		括号后的值作操作数处理，括号前的运算在括号内的表达式之前不执行。
	)		现执行已被设置运算
AND	**AND**	N,(	位 AND
OR	**OR**	N,(	位 OR
XOR	**XOR**	N,(	位专用 OR
NOT	**NOT**		位 NOT
+	**ADD**	(	加法
-	**SUB**	(	减法
*****	**MUL**	(	乘法
/	**DIV**	(	除法
>	**GT**	(	大于
>=	**GE**	(	大于等于

=	**EQ**	(	等于
<>	**NE**	(	不等于
<=	**LE**	(	小于等于
<	**LT**	(	小于
MOD(in)	**MOD**		模相除
INDEXOF(in)	**INDEXOF**		POU in1 的内部指数
SIZEOF(in)	**SIZEOF**		已知 in 数据类型所需字节数
SHL(K,in)	**SHL**		操作符 in 位左移 k
SHR(K,in)	**SHR**		操作符 in 位右移 k
ROL(K,in)	**ROL**		操作符 in 位旋转 k 到左
ROR(K,in)	**ROR**		操作符 in 位旋转 k 到右
SEL(G,in0,in1)	**SEL**		in0（G 为 FALSE）和 in1（G 为 TRUE）两个操作数之间的两进制选择。
MAX(in0,in1)	**MAX**		2 个值中较大的一个的复原。
MIN(in0,in1)	**MIN**		in0 和 in1 两值之中较小一个的复原。
LIMIT(MIN,in,Max)	**LIMIT**		限制值域（in 重置为 MIN，或 MAX，以免超出范围）。
MUX(K,in0,...in_n)	**MUX**		一组值中（in 至 In_n）选择第 k 个值。
ADR(in)	**ADR**		DWORD 中操作数的地址。
ADRINST()	**ADRINST**()		调用操作符即功能块操作符的地址。
BITADR(in)	**BITADR**		DWORD 中操作数的位平衡。
BOOL_TO_<type>(in)	**BOOL_TO_<type>**		布尔操作数的类型转换。
<type>_TO_BOOL(in)	**<type>_TO_BOOL**		类型转换成 BOOL。
INT_TO_<type>(in)	**INT_TO_<type>**		INT 操作数类型转换位另一基本类型。
REAL_TO_<type>(in)	**REAL_TO_<type>**		REAL 操作数类型转换位另一基本类型。
LREAL_TO_<type>(in)	**LREAL_TO_<type>**		LREAL 操作数类型转换位另一基本类型。
TIME_TO_<type>(in)	**TIME_TO_<type>**		TIME 操作数类型转换位另一基本类型。
TOD_TO_<type>(in)	**TOD_TO__<type>**		TOD 操作数类型转换位另一基本类型。
DATE_TO_<type>(in)	**DATE_TO_<type>**		DATE 操作数类型转换位另一基本类型。
DT_TO_<type>(in)	**DT_TO_<type>**		DT 操作数类型转换位另一基本类型。
STRING_TO_<type>(in)	**STRING_TO_<type>**		字符串操作数类型转换位另一基本类型，in 必须含所求类型的有效值。
TRUNC(in)	**TRUNC**		从 REAL 转换为 INT。
ABS(in)	**ABS**		操作数 in 的绝对值。

SQRT(in)	**SQRT**		操作数 in 的平方根。
LN(in)	**LN**		操作数 in 的自然对数。
LOG(in)	**LOG**		操作数 in 的以 10 为底的对数。
EXP(in)	**EXP**		操作数 in 的幂方。
SIN(in)	**SIN**		操作数 in 的正弦。
COS(in)	**COS**		操作数 in 的余弦。
TAN(in)	**TAN**		操作数 in 的正切。
ASIN(in)	**ASIN**		操作数 in 的反正弦。
ACOS(in)	**ACOS**		操作数 in 的反余弦。
ATAN(in)	**ATAN**		操作数 in 的反正切。
EXPT(in,expt)	**EXPT** expt		操作数 in with expt 的幂乘方。

<备注> 请于 CODESYS 程序库的相关附录里获取详细的使用信息。

6.4.2 Standard.lib 库

Standard.lib 库指令

在 ST 中	在 AWL 中	描述
LEN(in)	**LEN**	操作数 in 的字符串长度
LEFT(str,size)	**LEFT**	已知字符串 str 大小的左边第一个字符串。
RIGHT(str,size)	**RIGHT**	已知字符串 str 大小的右边第一个字符串。
MID(str,size,pos)	**MID**	POS 处已知大小的部分 str 字符串。
CONCAT('str1','str2')	**CONCAT** 'str2'	两个后继字符串的合并。
INSERT('str1','str2',pos)	**INSERT** 'str2',p	在 pos 处把字符串 str1 插入字符串 str2
DELETE('str1',len,pos)	**DELETE** len,pos	删除部分字符串（长度 len），启动位置为 str1 的 pos 处。
REPLACE('str1','str2',len,pos)	**REPLACE** 'str2',len,pos	替换长度字符串为 str2，起始位置为 str1 的 pos 处。
FIND('str1','str2')	**FIND** 'str2'	在 str1 中寻找部分字符串 str2。
SR	**SR**	双稳态的 FB 设置为显性。
RS	**RS**	双稳态的 FB 重设。
SEMA	**SEMA**	FB：臂板软件（可中断）
R_TRIG	**R_TRIG**	FB：上升边界被监测。
F_TRIG	**F_TRIG**	FB：下降边界被监测。
CTU	**CTU**	FB：升值计数。
CTD	**CTD**	FB：下降计数。

CTUD	**CTUD**	FB：上下计数。
TP	**TP**	FB：触发
TON	**TON**	FB：On-Delay 时钟。
TOF	**TOF**	FB：Off-Delay 时钟。
RTC	**RTC**	FB：现实时钟。

6.4.3 Util.lib 库

应用指令库 Util.lib 属于外部库，工程建立时不会自动添加到工程中，因此需要用户手动添加到工程中。

在该库中最常用到两个指令：PID、BLINK。

Util.lib 库指令

指令	指令说明
BCD_TO_INT	字节转换：BCD 转为 INT 格式。
INT_TO_BCD	字节转换：INT 转为 BCD 格式。
EXTRACT(in,n)	DWORD in 的第 n 位以 BOOL 类型还原。
PACK	多达 8 位被压入一个字节。
PUTBIT	DWORD in 一个位被设位一定的值。
UNPACK	字节被作为单个的位被复原。
DERIVATIVE	局部派生。
INTEGRAL	整数。
LIN_TRAFO	REAL 值的转换。
STATISTICS_INT	INT 格式的最小，最大和平均值。
STATISTICS_REAL	REAL 格式的最小，最大和平均值。
VARIANCE	变异。
PD	PD 控制器。
PID	PID 控制器。
BLINK	脉冲信号。
FREQ_MEASURE	布尔输入信号的测量频率。
GEN	周期性功能。
CHARCURVE	线性功能
RAMP_INT	限制供应功能（INT）减弱的上升。
RAMP_REAL	限制供应功能（REAL）减弱的上升。
HYSTERESIS	滞后作用。
LIMITALARM	注意输入值是否超出定义范围的限制。

6.4.4 SoftMotion 库

SM3_Basic Library

该库是 CODESYS SoftMotion 应用程序的基本库，因此它必须在 SoftMotion 工程中被包括。这将在一个 SoftMotion 设备插入时自动进行。

该库提供了以下功能块和函数：

● PLCopen 功能块既允许实现单轴运动控制，又允许实现两轴同步运动。除了状态检查、参数化和一般操作的库元素以外，还有以定义的速度、加速度参数来驱动轴的功能块。此外还有同步主、从轴和传动轴的模块。

● DriveInterface 基本驱动程序功能块（AXIS_REF_SM3, AXIS_REF_VIRTUAL_SM3, AXIS_REF_MAPPING_SM3）

● 工具函数，比如文件服务和错误报告

● 特定驱动器驱动程序的功能块或实现，基本驱动功能块的扩展。这些特殊驱动功能块（名为 AXIS_REF_*）通常被包括在一个单独的库中，该库被这个设备的相应设备描述所参考。

功能块整理如下：

● 驱动接口

指令名称	功能
AXIS_REF	这个驱动接口是一个功能块。它由【SM3_Basic.library】库提供，并包括作为驱动器的工作。每一个 SoftMotion 轴都是 AXIS_REF_SM3 实例的一个扩展。

● 数据类型，全局变量

指令名称	功能
MC_DIRECTION	它用作几个功能块的输入并指定运动方向。请注意，不是所有的模式都能够应用于所有的功能块和轴类型（模数/有限）。
MC_TAPPETMODE	它用于功能块 MC_DigitalCAMSwitch 以确定 tappet 值应参考主轴设定点（1），还是实际位置（2）。在自动检测（0）情况下，功能块的判断依赖于：驱动的控制状态是使用了设定值（bRegulatorOn = TRUE）还是实际值（bRegulatorOn = FALSE）。
SMC_BRAKESETSTATE	它用于功能块 SMC3_BrakeControl，决定驱动器如何设置机械制动。
SMC_CAMTAPPETACTION	它确定了当凸轮已被经过时，哪些开关动作被执行。
SMC_CAMTAPPETTYPE	它用于确定凸轮必须经过哪个方向才能被激活。
SMC_CONTROLLER_MODE	它描述了电机的哪个量被控制。
SMC_ERROR	它包括了 SoftMotion 功能块可能返回的所有错误号。
SMC_HOMING_MODE	在 SMC_Homing 使用时它定义了归航序列。
SMC_INT_STATUS	它描述插值器实例的状态。
SMC_LANGUAGE_TYPE	它用于函数 SMC_ErrorString 以确定语言。

● 数据类型，结构

指令名称	功能

MC_CAM_ID	它是一个描述 CAM 工作台的内部数据结构。它由 MC_CAMTableSelect 生成并做为 MC_CamIn 的输入。
MC_CAM_REF	它代表一个普通 CAM 并包含如下元素：
MC_CAMSWITCH_REF	它描述了 CAMSwitches 表并用作 MC_DigitalCAMSwitch 的输入。
MC_CAMSWITCH_TR	它描述了随行件的切换位置。
MC_OUTPUT_REF	它定义了 32 个布尔数组的形式的挺杆输出。 TYPE MC_OUTPUT_REF : ARRAY [0..31] OF BOOL; END_TYPE
MC_TRACK_REF	

● DriveBasic

指令名称	功能
SMC3_BrakeControl	如果被驱动装置和它的 SoftMotion 驱动程序支持，该功能块确定了机械刹车制动器的行为。
SMC3_BrakeStatus	它读取机械制动器的状态。
SMC_SetControllerMode	如果驱动设备支持，该模块可用来切换到其他控制器模式。
SMC_GetMaxSetAccDec	它用来测量一个轴的加速度（或减加速度）绝对值的最大值。
SMC_GetMaxSetVelocity	它用来测量一个轴速度最大值。
SMC_GetTrackingError	它测量用来补偿死区时间的实际和最大迟滞误差，这种误差可能在现场总线通讯时发生并将持续很多个周期(byDeadTimeCycles)。
SMC_InPosition	它检查轴的实际位置是否在一段确定的时间内在某个位置值以内。
SMC_MeasureDistance	它用于一个旋转轴，以测量走过的距离（外壳也被计算在内）。
SMC_CheckLimits	它可用来检查驱动装置的实际设置点是否超过了在控制器中配置的最大值。检查的结果将被 bLimitsExceeded 输出指示。
SMC_FollowPosition	它将不作任何检查地把位置设置点写入轴。
SMC_FollowPositionVelocity	除了可能额外定义的速度以外，它的使用和 SMC_FollowPosition 等效。
SMC_FollowSetValues	它将不作任何检查地把设置值写入轴。
SMC_FollowVelocity	它将不作任何检查地将设置速度写入轴。
SMC_ClearFBEror	它用来删除最老的功能块错误信息。
SMC_ReadFBError	它用来读取功能块错误的最老信息。
SMC_Homing	它执行轴的参考运动。做为参考开关，一个布尔值被使用，典型地该值是由一个硬件输入的。
SMC_ChangeGearingRatio	在该模块的帮助下，传动比和驱动设备类型可以被修改。

● Error

指令名称	功能

SMC_ErrorString	依赖于输入 ErrorID 和语言，函数 SMC_ErrorString 将返回一个字符串表达的错误。

● File

指令名称	功能
SMC_ReadCam	它用来在运行时下载 CAM 并使其对 MC_CamTableSelect 和 MC_CamIn 有效。
SMC_WriteCam	该功能块在运行时向*.CAM-file 文件写入一个在 CAM 编辑器中创建的 CAM。
SMC_AxisDiagnosticLog	它用来将属于一个轴的一系列参数值循环写入到文件中。这个输出文件是故障诊断的依据。

● PLCopen

指令名称	功能
SMC_ReadSetPosition	它用来读取驱动设备的当前设置位置。
SMC_SetTorque	如果驱动设备处于控制器模式【torque】时，它可用来创建一个力矩。
SMC_CAMBounds	它计算从设备的最大位置、速度和加速度值，该设备以绝对模式与一个主设备相连，主设备按规定的最大速度和加速度/减加速度移动。
SMC_CAMBounds_Pos	与 SMC_CAMBounds 相对的，该功能块仅计算从设备的位置最大、最小值，该设备以绝对模式与一个主设备相连，主设备按规定的最大速度和加速度/减加速度移动。
SMC_CAMRegister	它代表一个挺杆控制单元，它和 MC_CamIn 一样，作用于 MC_CAM_REF 结构，撤销原轨迹信息而只读挺杆信息。
SMC_GetCamSlaveSetPosition	如果（从）轴通过 CAM 与其他（主）轴的运动相连接，该模块计算从轴的当前目标位置。因此两个轴都不会被移动或影响。
SMC_GetTappetValue	它评估 MC_CamIn 功能块的输出 Tappet 并包含当前 tappet 状态。
MC_CamIn	它占用 CAM。
MC_CamOut	它将立即从主轴中释放从轴。
MC_CamTableSelect	它通过设置与相关工作台的连接来选择 CAM 工作台。
MC_GearIn	它通过设置与相关工作台的连接来选择 CAM 工作台。
MC_GearInPos	它通过设置与相关工作台的连接来选择 CAM 工作台。
MC_GearOut	它通过设置与相关工作台的连接来选择 CAM 工作台。它使从轴脱离主轴。
MC_Phasing	它将占用 CAM。
MC_AccelerationProfile	它用来控制一个时间-加速度已锁定的运动分布。
MC_Halt	它用来命令一个受控运动停止。功能块将终止所有进行中的功能块的执行。
MC_Home	它的执行引发轴执行【search home】序列。该序列的详情依赖于加工工艺并可以由轴的参数设置。
MC_MoveAbsolute	它用来命令受控运动到一个指定的绝对位置。

MC_MoveAdditive	它用来在离散运动状态下命令一个受控运动，该运动把指定相对距离与新近要求的位置相加。
MC_MoveRelative	它用来命令一个受控运动，在执行时该运动指定了相对于轴的实际位置的距离。
MC_MoveSuperImposed	它用来命令一个受控运动，该运动将指定的相对距离加到已有的运动中。已有的运动不会被中断，而是叠加附加运动。
MC_MoveVelocity	它用来以特定速度命令一个永不终止的受控运动。
MC_PositionProfile	它用来命令一个时间－位置锁定的运动分布。
MC_Power	它用来控制 power stage（【开】或【关】）。
MC_ReadActualPosition	它将返回参考轴的实际位置。
MC_ReadAxisError	它用来描述与功能块无关的普通轴错误。
MC_ReadBoolParameter	它返回厂家指定的 BOOL 类型的参数。
MC_ReadStatus	关于当前进程中的运动，它返回轴的详细状态。
MC_ReadParameter	它返回厂商指定的参数。
MC_Reset	它通过重置所有内部关于轴的错误，把 ErrorStop 转换成 StandStill 状态－这会影响功能块实例的输出。
MC_Stop	它的执行将导致受控运动停止并将轴设置为状态【Stopping】。
MC_VelocityProfile	它用来命令一个时间－速度锁定的运动分布。
MC_WriteBoolParameter	它可用来修改厂家设定的布尔类型的参数。
MC_WriteParameter	它可用来修改厂家设定的参数。
MC_AbortTrigger	它是用来终止与触发事件（例如 MC_TouchProbe）相连接的功能块。
MC_DigitalCamSwitch	和电动机轴上的开关类似：功能块控制一组离散输出位元，来开关一组就像是由机械凸轮控制的、与轴相连的开关。向前和向后运动都是允许的。
MC_ReadActualTorque	当 Enable 保持为 TRUE 时，功能块将返回实际转矩或力的值。
MC_ReadActualVelocity	当 Enable 保持为 TRUE 时，功能块将返回实际速度的值。
MC_SetPosition	它用来移动一个轴的坐标系。
MC_TouchProbe	它用来在触发事件发生时记录轴的位置。
SMC_MoveContinuousAbsolute	它执行绝对运动，但是与 MC_MoveAbsolute 不同，它不是以速度＝0 到达目标位置的，而是以指定速度。
SMC_MoveContinuousRelative	它执行相对运动，但是与 MC_MoveRelative 不同，它不是以速度＝0 到达目标位置的，而是以指定速度。

● SimpleTest

指令名称	功能
SMC_StartupDrive	它包含一组经常使用的功能块，它用来测试和试运转一个轴。

SM3_CNC Library

SM3_CNC.Library 库里的模块用来（依照 DIN 66025）解码在 CNC 编辑器中创建的轨迹。解码是为了转化轨迹到一个结构中，以使其能够被 IEC 程序所用（轨迹预处理），或能够被修改、被插补以及被转换以使其对于驱动设备是可读的（例如，SMC_NCDecoder，ToolCorr，SMC_AvoidLoop，SMC_SmoothPath，SMC_RoundPath，SMC_Interpolator）。

模块的功能故障允许某些组件－比如解码器－被为特殊需要而设计的模块所代替。此外，其他模块－比如某个轨迹预处理或者优化－可以被毫无问题地加入到已有组件中，这样不需要考虑库中组件间的适配性。因此，程序逻辑完全由 PLC 程序处理，只有纯动作信息才由库模块处理。

库 SM3_CNC.Library 的数据结构（例如 SMC_POSINFO，SMC_VECTOR3D）用来描述位置、轨迹段和向量，并管理 GEOINFO 对象（QUTQUEUE 结构）。

- 数据类型

指令名称	功能
SMC_CNC_REF	这个数据结构管理被解析了的 G 代码文件。
SMC_GCODE_TEXT	数据源与 SMC_NCDecoder 模块的同名输出相连。它用来向功能块 SMC_GCodeViewer 传递 CNC 文件最后一行已解码的字符串。
SMC_GCODEVIEWER_DATA	该结构体用一队数组作为SMC_GCodeViewer功能块的缓存；一个该类型的变量存储一条曲线轨迹对应的G代码文件。
SMC_GCODE_WORD	它包括某个 G 代码字如【N20】或【G1】的数据。
SMC_GEOINFO	该结构是被设计用来存储轨迹对象的，也就是已编程轨迹的一段，比如一条直线、一段圆弧或者样条曲线段。
SMC_M_PARAMETERS	它可以为当前活动的 M 函数定义附加参数，这些参数可被 SMC_GetMParameters 计值，以显示各个 M 函数的附加值。
SMC_OUTQUEUE	该结构用来管理一个已定义大小的表中的 GEOINFO 对象。
SMC_POSINFO	对一个特殊位置点该结构描述了它的坐标和附加轴的位置。
SMC_VECTOR3D	该结构体描述了三维的速度。

- 全局变量

指令名称	功能
SMC_AL_STATUS	它的值指明了 FB 实例的实际状态。
SMC_DEC_STATUS	它的值指示了 FB 实例的实际状态。
SMC_DIRECTION	它的值指明了功能块实例的实际插补方向。
SMC_INT_STATUS	它的值指示了 FB 实例的实际状态。可能的状态：
SMC_INT_VELMODE	它的值确定了速度分布图。
SMC_MOVTYP	它是 SMC_GEOINFO 的一个成员，它指明了对象的几何形状。
SMC_TC_STATUS	它的值代表 FB 实例的实际状态。
SMC_TOOLCORRMODE	它的值代表 SMC_Toolcorr 运行时的模式。
SMC_VECTORDIR	它作为SMC_CalcDirectionFromVector的输入变量，用来定义速度的倾角、反向或垂直。

功能块记录

- 在 SMC_CNC_POUs\File 文件夹内

指令名称	功能
SMC_READNCFILE	它从控制器的文件系统中读取一个 NC-ASCII 文档，使得该文档对 SMC_NCDecoder 可用。因此在运行时一个 NC 程序可被读入并被执行。
SMC_READNCQUEUE	它从 PLC 文件系统中读取一个由 CNC 编辑器创建的 OutQueue 文件，并提供一个 OutQueue 结构，典型地该结构是由解码器处理的。

SMC_VARLIST	IEC 61131-3 标准不可能通过字符串的符号名来决定一个变量的值。但这是必要的，假如变量的功能由 SMC_CNC_REF 提供给用户使用，也应该可以从一个文件中读取 CNC 程序。这个可以被结构 SMC_VARLIST 所管理。

● SMC_CNC_POUs\SoftMotion CNC\Coordinate Transformation 文件夹内

指令名称	功能
SMC_DETERMINECUBOIDBEARING	它用给出的 6（3/2/1）个点确定了一个立方体（角标志，边对齐）在空间中的位置。
SMC_COORDINATETRANSFORMATIONS3D	它计算在旧坐标系中给出、并转换到新坐标系下的点的坐标。坐标变换是由原点的变换和新单位向量（按旧坐标系给出）来定义的。
SMC_INVCOORDINATETRANSFORMATION3D	它计算一个在新坐标系下给出的点的坐标并将其换算到旧坐标系中。
SMC_TEACHCOORDINATESYSTEM	辅助用户计算新坐标系的向量。
SMC_UNITVECTORTORPY	从新坐标系的单位向量计算 RPY-角度（参考旧坐标系）
SMC_ROTATEQUEUE2D	在该模块执行时，存储在 poqDataIn 中的轨迹将被绕 Z 轴旋转给出的角度 dPhi[°]。正角度是指数学上的正向旋转（逆时针）。
SMC_SCALEQUEUE3D	在它执行时，在 poqDataIn 所包含的轨迹将被因数 fScaleFactor 拉伸。
SMC_TRANSLATEQUEUE3D	通过它的执行存储在 poqDataIn 中的路径将会按照 vec 转换，vec 是 SMC_VECTOR3D 的数据类型。

● SMC_CNC_POUs\SoftMotion CNC\Direct Axis Control 文件夹内

指令名称	功能
SMC_CONTROLAXIXBYPOS	用于向驱动结构写目标位置，以测试跳跃的结构。它被广泛用于 CNC 和 SMC_Interpolator。对于方向控制轴的设置值，应该使用始于 SMC_Follow*的一个功能块。
SMC_CONTROLAXIXBYPOSVEL	除了在添加时速度可以被定义以外，它的使用等同于 SMC_ControlAxisByPos。它通常用于 CNC 和 SMC_Interpolator。

● SMC_CNC_POUs\SoftMotion CNC\GCode Viewer 文件夹内

指令名称	功能
SMC_GCODEVIEWER	它可以与 SMC_NCDecoder 结合使用。

● SMC_CNC_POUs\SoftMotion CNC\Posinfo Functions 文件夹内

指令名称	功能
SMC_POSINFO	对一个特殊位置点该结构描述了它的坐标和附加轴的位置。

● SMC_CNC_POUs\SoftMotion CNC\OutQueue Functions 文件夹内

指令名称	功能
SMC_APPENDOBJ	该函数块用来处理 SMC_OUTQUEUE 结构体对象。
SMC_DELETEOBJ	该布尔型函数用来处理一个 SMC_OUTQUEUE 结构对象。
SMC_GETCOUNT	类型为 UNIT 的函数将返回存储在 SMC_OUTQUEUE 表（POQ）中的对象个数。
SMC_GETOBJ	假如输入 poq 被正确初始化并含有至少 N+1 个元素，函数将返回一个

	指向表中第 N 个 GEOINFO 对象的指针，计数从表的开头算起；所以对于 N=0 第一个元素被返回。
SMC_GETOBJFROMEND	假如输入 poq 被正确初始化并含有至少 N+1 个元素，函数将返回一个指向表中第 N 个 GEOINFO 对象的指针，计数从表的结尾算起；所以对于 N=0 最后一个元素被返回。
SMC_OUTQUEUEINIT	它设置 SMC_OUTQUEUE 数据结构的变量为它的默认值。
SMC_RESTOREQUEUE	该函数块将恢复一个已被插补的或已被处理了的结构。仅当整个轨迹能存储到由 poq 引用的表中时，这才是可能的。
SMC_SETQUEUECAPACITY	用于设置 SMC_OUTQUEUE 数据结构的变量空间。

● SMC_CNC_POUs\SoftMotion CNC\SoftMotion Function Blocks 文件夹内

指令名称	功能
SMC_AVOIDLOOP	该功能块被用来做轨迹预处理。它拷贝一个轨迹，但并不剪断其内部含有的环圈。
SMC_CHECKFORLIMITS	用于检查轨迹是否离开其特定的矩形范围。
SMC_CHECKVELOCITIES	该模块检查特殊轨迹段的轨道速度。
SMC_EXTENDEDVELOCITYCHECKS	它减少路径的速度，加速度和减速度，使附加轴(Z, P, Q, U,V, W)产生的速度加速度和减速的值不超过 SMC_GEOINFO 变量（adVelAddAx[0..6], adAccAddAx[0..6] and adDecAddAx[0..6]）的设置值。
SMC_INTERPOLATOR	这个功能块用来把由 SMC_GEOINFO 对象描述的一个连续轨迹转化为离散轨迹位置点，这些点考虑到了速度分布图和时间图。
SMC_INTERPOLATOR2DIR	从函数以及输入输出的配置看，该模块相当于功能块 SMC_Interpolator，但是它还能够反向地插补一个轨迹。
SMC_INTERPOLATOR2DIR_SLOWTASK	该模块负责产生反向轨迹。它被从 SMC_Interpolator2Dir 中分割出来，这样它就能在完全拉伸的低性能系统中被封装给一个低优先级任务。
SMC_LIMITDYNAMICS	该模块用于减小轨迹的速度、加减速度，以使附加轴（Z, P, Q, U, V, W）的速度、加减速值不超过 FB 的输入值 dMaxVel, dMaxAcc 和 dMaxDec。
SMC_LIMITCIRCULARVELOCITY	该模块检查 OutQueue 中的特定元素并限制圆形元素沿其半径反方向的轨迹速度。
SMC_NCDECODER	该模块用来将一个在 CNC 编辑器中被创建的 CNC 程序（DIN 6625，G 代码）转化为 SoftMotion-GEOINFO 结构对象的表。在每次循环中程序里的一行将被解码。
SMC_ROUNDPATH	SMC_ROUNDPATH 功能块与 SMC_SmoothPath 模块十分相象。它由两线相交圆整截得的圆弧的角度。
SMC_SMOOTHPATH	SMC_SmoothPath 功能块可用于轨迹预处理。它可使轨迹尖角(path angles)变得圆滑，从而生成一个圆滑的轨迹（slur path）。 它用于对轨迹的精度要求不高，而对速度要求较高，且不允许速度降到 0 的情况。
SMC_TOOLCORR	The SMC_ToolCorr 用于路径预处理。
SMC_XINTERPOLATOR	该SMC_XInterpolator功能块实现了CAM和CNC的混合。假设你想把一个工件切一个特定的形状（由G代码描述），根据工件由另一个进程的移动（例如，沿X轴移动）和其他的轴（y，z等）应当按照工件（X）当前位置控制，并且目标由路径要点给出。
SMC_GETMPARAMETERS	插补器正处理一个 M 函数，该模块用来轮询已为该 M 函数设置的参

	数（K，L，O，见 HERE）。
SMC_PREACKNOWLEDGEMFUNCTIONS	它用来在插补器到达该模块之前确认一个 M 函数。这样在 M 函数上的停顿就能够避免了。

● SM_Trafo_POUs\Additional FBs 文件夹内

指令名称	功能
SMC_CALCDIRECTIONFROMVECTOR	使用它时工具的方向（dAlpha）可被计算。

● SM_Trafo_POUs\Gantry Systems 文件夹内

指令名称	功能
SMC_TRAFO_GANTRY2	对于龙门系统不能进行变换，因此模块只能给 x,y 轴加上偏移量。
SMC_TRAFO_GANTRY2TOOL1	功能块用来反求龙门系统的刀具轴向偏置，也就是相对 Z 轴的轴向偏置，它们近似在一条直线上。
SMC_TRAFO_GANTRY2TOOL2	功能块用来反求龙门系统的刀具轴向偏置，也就是相对 Z 轴的轴向偏置，它们近似成直角。
SMC_TRAFO_GANTRY3	对于龙门系统不能进行变换，因此模块只能给 x,y,z 轴加上偏移量。
SMC_TRAFO_GANTRYCUTTER2	为一个正在被控制并指出沿着路径切线旋转轴的三维龙门系统计算一个逆向变换。
SMC_TRAFO_GANTRYCUTTER3	为一个正在被控制并指出沿着路径切线旋转轴的三维龙门系统计算一个逆向变换。
SMC_TRAFO_GANTRYH2	这个转变模块用来提供解决有着固定的驱动器的H龙门系统的反向转变。
SMC_TRAFOF_GANTRY2	功能模块用来操作一个没有动作的三维龙门操作系统，因此，该模块仅在X,Y轴各自增加了一个偏移。每一个SMC_TRAFOF_GANTRY2模块的实例都能与一个命名为SMC_VISU_GANTRY2的可视化模板相连接。
SMC_TRAFOF_GANTRY2TOOL1	功能块用来反求龙门系统的刀具轴向偏置，也就是相对 Z 轴的轴向偏置，它们近似在一条直线上。
SMC_TRAFOF_GANTRY2TOOL2	功能块用来反求龙门系统的刀具轴向偏置，也就是相对 Z 轴的轴向偏置，它们近似成直角。
SMC_TRAFOF_GANTRY3	功能模块用来操作一个没有动作的三维龙门操作系统，因此，该模块仅在X,Y,Z轴各自增加了一个偏移。每一个SMC_TRAFOF_GANTRY3模块的实例都能与一个命名为SMC_VISU_GANTRY3的可视化模板相连接。
SMC_TRAFOF_GANTRYCUTTER2	该模块将解决有着指向沿着切线方向的旋转轴的三维龙门系统的前向转变。
SMC_TRAFOF_GANTRYCUTTER3	它将解决指向当前轨迹的切线方向的旋转轴的三维龙门系统的前向转变。
SMC_TRAFOF_GANTRYH2	这个转变模块用来提供解决有着固定的驱动器的H龙门系统的前向转变。
SMC_TRAFOV_GANTRY2	反向转换模块被用来提供处理考虑轨迹速度轨迹方向作为轴的控制变量的二维龙门系统。
SMC_TRAFOV_GANTRY3	反向转换模块被用来提供处理考虑轨迹速度轨迹方向作为轴的控制变量的三维龙门系统。
SMC_TRAFOV_GANTRYCUTT	反向转换模块被用来处理考虑轨迹速度轨迹方向作为轴的控制变量

ER2	的二维龙门系统。
SMC_TRAFOV_GANTRYCUTTER3	反向转换模块被用来提供处理考虑轨迹速度轨迹方向作为轴的控制变量的三维龙门系统。
SMC_TRAFOV_GANTRYH2	反向转换模块被用来提供处理考虑轨迹速度轨迹方向作为轴的控制变量的二维 H 龙门系统。

● SM_Trafo_POUs\Parallel Systems 文件夹内

指令名称	功能
SMC_TRAFO_BIPOD_ARM	转换模块用来提供处理双脚架臂的后向转换。
SMC_TRAFO_TRIPOD	转换模块用来提供处理三脚架的后向转换。
SMC_TRAFO_TRIPOD_ARM	转换模块用来提供处理三脚架臂的后向转换。
SMC_TRAFOF_BIPOD_ARM	转换模块用来提供处理双脚架臂的前向转换。
SMC_TRAFOF_TRIPOD	转换模块用来提供处理三脚架的前向转换。
SMC_TRAFOF_TRIPOD_ARM	转换模块用来提供处理三脚架臂的前向转换。

● SM_Trafo_POUs\Scara Systems 文件夹内

指令名称	功能
SMC_TRAFO_SCARA2	转换模块用来提供处理第二关节 Scara 系统的后向转换。
SMC_TRAFO_SCARA3	转换模块用来提供处理第三关节 Scara 系统的后向转换。
SMC_TRAFOF_SCARA2	转换模块用来提供处理第二关节 Scara 系统的前向转换。
SMC_TRAFOF_SCARA3	转换模块用来提供处理第三关节 Scara 系统的前向转换。

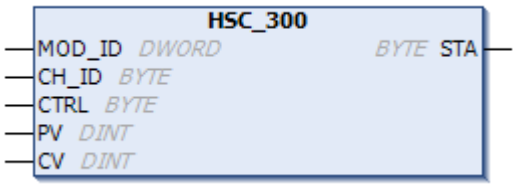
6.4.5 ExtBus 库指令说明

高速计数器支持的指令库（ExtBus），各指令的描述参考如下说明。

指令	名称
HSC_300	设置计数参数指令
HSC_CLEARLOCK	清除锁存值
HSC_GETLOCK	获取当前锁存值
HSC_GETCV	获取当前计数值
HSC_GETSPEED	获取当前计数速度
HSC_GETSPEED-AVG	使用平均值获取当前速度
HSC_GETSTA	获取当前计数状态
HSC_SETMODE	设置计数器模式

1、设置计数参数指令

函数名：HSC_300



功能：设置计数参数。

参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
CTRL	IN	控制字	BYTE	16#F9	见下表
PV	IN	预设值	DINT	0	
CV	IN	当前值	DINT	0	
STA	OUT	返回状态	BYTE	0	模块状态字 0：OK；5：模块参数错误；7：模块没响应； 8：模块链路层校验错误。

控制字（R/W）

7	6	5	4	3	2	1	0
计数使能	当前值更新	预设值更新	计数方向更新	计数方向	正交计数选择		复位电平

复位电平：1-高电平复位，0-低电平复位

正交计数选择：00-4x 倍数，01-2x 倍数，10-1x 倍数

计数方向：0-减计数，1-增计数

计数方向更新：0-不更新，1-更新

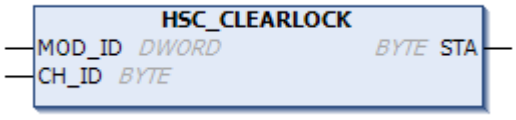
预设值更新：0-不更新，1-更新

当前值更新：0-不更新，1-更新

计数使能：0-不使能，1-使能

2、清除锁存值

函数名：HSC_CLEARLOCK



功能：清除锁存值。

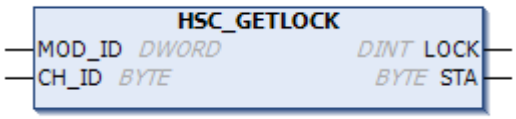
参数

参数名	输入输	参数描述	类型	初始化	备注

	出属性				
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0~5）HSC模块取值（0，1）
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；5：模块参数错误；7：模块没响应；8：模块链路层较验错误。

3、获取当前锁存值

函数名：HSC_GETLOCK



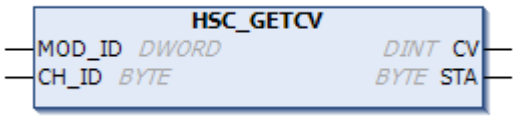
功能：获取当前锁存值

参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
LOCK	OUT	锁存值	DINT	0	锁存值
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；2：参数非法；5：模块参数错误； 7：模块没响应；8：模块链路层较验错误。

4、获取当前计数值

函数名：HSC_GETCV



功能：获取当前计数值

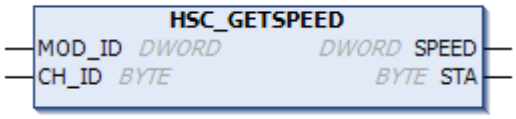
参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
CV	OUT	当前计数值	DINT	0	当前值
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；2：参数非法；5：模块参数错误；

					7：模块没响应；8：模块链路层较验错误。

5、获取当前计数速度

函数名：HSC_GETSPEED



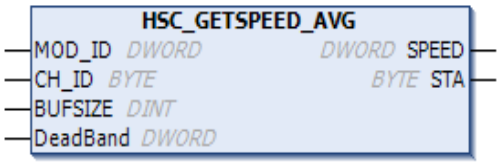
功能：获取当前计数速度

参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
SPEED	OUT	计数速度	DWORD	0	Hz
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；2：参数非法；5：模块参数错误； 7：模块没响应；8：模块链路层较验错误。

6、使用平均值获取当前速度

函数名：HSC_GETSPEED-AVG



功能：使用平均值获取当前速度

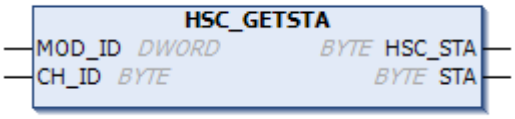
参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
BUFSIZE	IN	平均值缓冲区	DINT	16	平均值缓冲区大小大于0小于64
DeadBand	IN	死区值	DWORD	20000	平均值与现有值的差值小于死区值，以平均值为准，大于死区平均值为现有值
SPEED	OUT	计数速度	DWORD	0	Hz
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；2：参数非法；5：模块参数错误；

					7：模块没响应；8：模块链路层较验错误。

7、获取当前计数状态

函数名：HSC_GETSTA



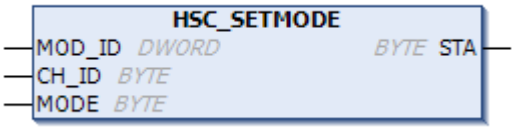
功能：获取当前计数状态

参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
HSC_STA	OUT	计数状态	BYTE	0	Bit0~Bit3：当前模式 Bit4：预留 Bit5：HSC0当前计数方向位：1=增计数 Bit6=1：当前值等于预设值位 Bit7=1：当前值大于预设值位
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；2：参数非法；5：模块参数错误； 7：模块没响应；8：模块链路层较验错误。

8、设置计数器模式

函数名：HSC_SETMODE



功能：设置计数器模式。

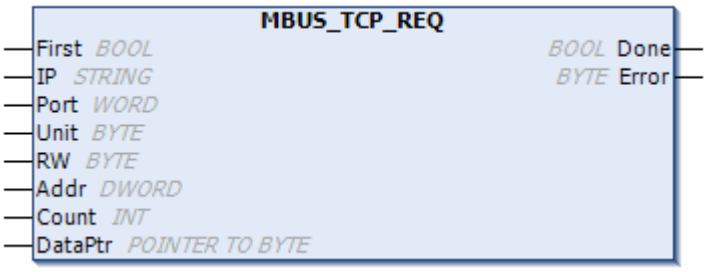
参数

参数名	输入输出属性	参数描述	类型	初始化	备注
MOD_ID	IN	模块地址	DWORD	0	PLC 本机16#0a000605，硬件组态的 Module Id
CH_ID	IN	通道地址	BYTE	0	PLC本机取值（0-5）HSC模块取值（0，1）
MODE	IN	计数模式	BYTE	0	Bit0~Bit3：HSC 计数模式（本机支持 0，1，3，4，6，7，9，10；HSC-02 支持 0-11 模式） 模式 0~2：具有内部方向控制的单相计数器。 模式 3~5：具有外部方向的单相计数器。 模式 6~8：具有 2 个时钟输入的双相计数器。 模式 9~11：A\B 相正交计数器。

					Bit4：Z 信号锁存功能，0：锁存，1：不锁存 Bit5：Z 信号清零功能，0：清零，1：不清零 Bit6：预留 Bit7：锁存值清零 0：无效，1：有效
STA	OUT	模块状态字	BYTE	0	模块状态字 0：OK；2：参数非法；5：模块参数错误； 7：模块没响应；8：模块链路层较验错误。

6.4.6 Modbus_TCP 库文件

1）Modbus_Tcp 主站 MBUS_TCP_REQ 使用说明



MBUS_TCO_REQ 指令的参数说明

参数	输入输出属性	数据类型	初始化	说明
First		BOOL	0	激活位(0:非激活 1:激活)，激活一次，指令执行一次。
IP		STRING	192.168.0.1	从站 ip 地址,例如'192.168.0.1'
Port		WORD	502	从站端口号,例如 502
Unit		BYTE	0	从站单元号,例如 0
RW		BYTE	0	读写标志 0：读（Read） 1：写（Write）
Addr		DWORD	0	读写寄存器地址(比如 40001,30001)
Count		INT	0	读写字节数量(取值范围 0-120 个字)
DataPtr		POINTER TO BYTE	0	读写指针(存放读取到的数据的位置或存放要写到寄存器的数据)）
Done		BOOL	0	完成位 0:指令正在执行 1:指令执行完成
Error		BYTE	0	错误码

错误代码	意义
0	无错误
1	保留

2	保留
3	接收超时（从站无响应）
4	请求参数出错（IP 地址、Modbus 地址、count、RW，Unit）
5	Modbus 未使能
6	Modbus 正在忙于其它请求
7	响应错误（响应不是请求的操作）
8	响应 CRC 校验和错误
9	硬件错误
101	从站不支持请求的功能
102	从站不支持数据地址（起始地址的长度范围）
103	从站不支持此种数据类型
104	从站设备故障
105	从站接收了信息，但是响应被延迟
106	从站忙，拒绝了该信息
107	从站拒绝了信息
108	从站存储器奇偶错误

2）Modbus_Tcp 从站 MBUS_TCP_SLAVE 使用说明

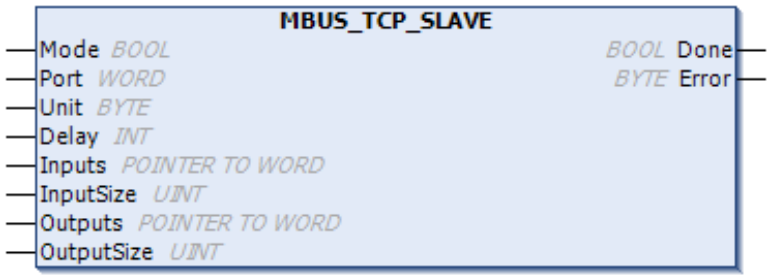


表 E-10　MBUS_TCP_SLAVE 指令的参数说明

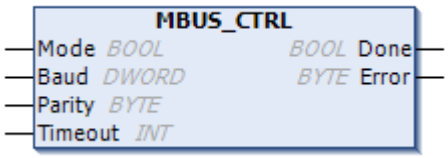
参数	输入输出属性	数据类型	初始化	说明
Mode		BOOL	1	使能开关 0:关闭 1:打开
Port		WORD	502	从站端口，例如：502
Unit		BYTE	0	从站单元号，例如：0
Delay		INT	0	回复延时时间(一般设置为 0 即可，如其他通讯参数均正确但仍无法通讯，可适当增加该值)
Inputs		POINTER TO WORD	0	指向 Input Register 的指针

InputSize		UINT	0	Input Register 的大小
Outputs		POINTER TO WORD	0	指向 Holding Register 的指针
OutputSize		UINT	0	Holding Register 的大小
Done		BOOL	0	完成位(0:指令正在执行 1:指令执行完成)
Error		BYTE	0	错误码，见下表

错误代码	意义
1	存储区范围错误
2	非法波特率/校验
3	非法的单元 ID
4	非法协议 (Modbus 参数)
5	保持寄存器与 Modbus 从站符号地址重叠
6	保留
7	保留
8	非法功能请求
9	请求中有非法存储区地址
10	Modbus 未初始化

6.4.7 Modbus_RTU 库文件

Modbus_RTU 主站 MBUS_CTRL 使用说明



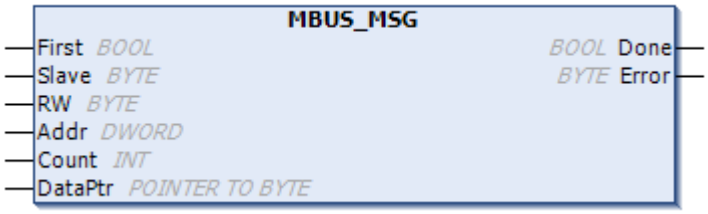
MBUS_CTRL 指令的参数说明

参数	输入输出属性	数据类型	范围	说明
Mode		BOOL		使能开关(0:关闭 1:打开)
Baud		DWORD	1200，2400，4800，9600 19200，38400，57600，115200	波特率
Parity		BYTE		设置校验 0：无校验

				1：奇校验 2：偶校验
Timeout		INT		超时时间(从站没有在设置时间内回复数据即认为读写超时)
Done		BOOL		完成位 0：指令正在执行 1：指令执行完成
Error		BYTE		错误码（Done 为 1 时生效），见下表

错误代码	意义
0	无错误
1	校验选择非法
2	波特率选择非法
3	超时时间非法
4	模式选择非法
9	硬件错误

主站 MBUS_MSG 使用说明



MBUS_MSG 指令的参数说明

参数	输入输出属性	数据类型	初始化	说明
First		BOOL	0	激活位(0:非激活 1:激活)，没激活一次，指令执行一次。
Slave		BYTE	0	从站站号
RW		BYTE	0	R/W 读写 0：读 1：写
Addr		DWORD	0	读写寄存器地址（比如 40001,30001）
Count		INT	0	读写字节数量(取值范围 0-120 个字)
DataPtr		POINTER TO BYTE	0	读写指针(存放读取到的数据的位置或存放要写到寄存器的数据)
Done		BOOL	0	完成位 0：指令正在执行中 1：指令执行完成

Error		BYTE	0	错误码（Done = 1 时生效），见下表

错误代码	意义
0	无错误
1	响应校验错误
2	未用
3	接收超时（从站无响应）
4	请求参数错误（从站地址，Modbus 地址，count，RW）
5	Modbus/自由口未使能
6	Modbus 正在忙于其它请求
7	响应错误（响应非请求的操作）
8	响应 CRC 校验和错误
9	硬件错误
101	从站不支持请求的功能
102	从站不支持数据地址（起始地址的长度范围）
103	从站不支持数据类型
104	从站设备故障
105	从站接收了信息，但是响应被延迟
106	从站忙，拒绝了该信息
107	从站拒绝了信息
108	从站存储器奇偶校验错误

<备注> 同一时间，只能有一个 MBUS_MSG 可以被激活。如果多个 MBUS_MSG 同时被激活，MBUS_MSG 将返回错误代码 6。

从站 MBUS_INIT 使用说明

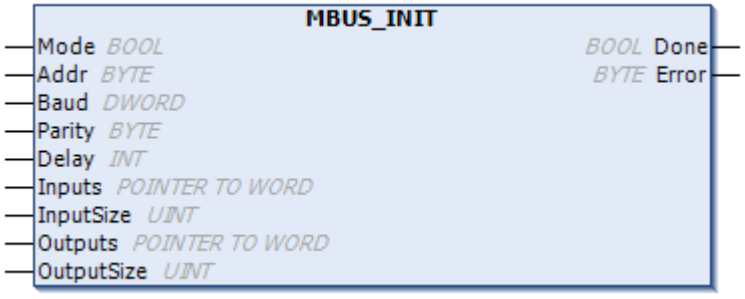
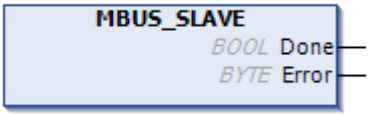


表 E-3　MBUS_INIT 指令的参数说明

参数	输入输出属性	数据类型	初始化	说明
Mode		BOOL	1	使能开关(0:关闭 1:打开)
Addr		BYTE	2	本站地址
Baud		DWORD	9600	波特率 (1200,2400,4800,9600,19200,38400,57600,115200)
Parity		BYTE	0	奇偶校验 0：无校验 1：奇校验 2：偶校验
Delay		INT	0	回复延时时间(一般设置为 0 即可，如其他通讯参数均正确但仍无法通讯，可适当增加该值)
Inputs		POINTER TO WORD	0	指向 Input Register 的指针
InputSize		UINT	0	Input Register 的大小
Outputs		POINTER TO WORD	0	指向 Holding Register 的指针
OutputSize		UINT	0	Holding Register 的大小
Done		BOOL	0	完成位(0:指令正在执行 1:指令执行完成)
Error		BYTE	0	错误码，见下表

错误代码	意义
1	存储区范围错误
2	非法波特率或校验
3	非法从站地址
4	非法协议（Modbus 参数）
5	保持寄存器与 Modbus 从站符号地址重叠
6	接收校验错误
7	接收 CRC 错误
8	非法功能请求
9	请求中有非法存储区地址
10	Modbus 未初始化

从站 MBUS_SLAVE 使用说明



MBUS_SLAVE 指令的参数说明

参数	输入输出属性	数据类型	说明
Done		BOOL	完成位(0:指令正在执行 1:指令执行完成)
Error		BYTE	错误码，见下表

错误代码	意义
1	存储区范围错误
2	非法波特率或校验
3	非法从站地址
4	非法协议 (Modbus 参数)
5	保持寄存器与 Modbus 从站符号地址重叠
6	接收校验错误
7	接收 CRC 错误
8	非法功能请求
9	请求中有非法存储区地址
10	Modbus 未初始化