运动控制器
MOTION CONTROLLER

EURA DRIVES

使用手册
EAC100系列 运动控制器

EURA 欧瑞传动
DRIVES EURA DRIVES

目 录

第一章 概述 - 1 -
1.1 EAC100 系列运动控制器简述 - 1 -
1.2 EAC100 系列运动控制器命名规范及订货信息 - 2 -
1.3 EAC100 系列运动控制器技术规格及工作环境参数 - 7 -
1.4 安全注意事项 - 7 -
1.4.1 安全信息定义 - 8 -
1.4.2 警告标示 - 8 -
1.4.3 安全指导 - 8 -
第二章 运动控制器硬件介绍 - 9 -
2.1 控制器外形结构 - 9 -
2.1.1 EAC111 系列 - 9 -
2.1.2 EAC112 系列 - 10 -
2.2 外形尺寸及安装方式 - 11 -
2.2.1 外形尺寸 - 11 -
2.2.2 控制器安装方式 - 12 -
2.3 运动控制器本体部分 - 12 -
2.3.1 控制器本体运行状态指示及系统工作状态 - 12 -
2.3.2 DIP 拨码开关 - 13 -
2.3.3 控制器本体对外接口 - 14 -
2.3.4 控制器输入（DI） - 16 -
2.3.5 控制器输出（DO） - 17 -
2.3.6 数字量 DI/DO 接线图 - 19 -
2.4 运动控制器数字量拓展 IO 部分 - 19 -
2.4.1 控制器数字量拓展 IO 运行状态指示及系统工作状态 - 19 -
2.4.2 DI 16*DC24 - 20 -
2.4.3 DO 16*DC24 - 21 -
2.4.4 DI 8*DC24+DO 8*DC24 - 24 -
第三章 控制器软件功能 - 27 -

3.1 概述 - 27 -
3.2 控制器软件功能及设置 - 27 -
3.2.1 网页设置功能 - 27 -
3.2.2 U 盘更新功能 - 30 -
3.3 上位机软件 CODESYS - 31 -
3.3.1 概述 - 31 -
3.3.2 CODESYS 的安装 - 31 -
3.3.3 基本界面 - 36 -
3.3.4 示例程序安装 - 36 -
3.3.5 设备描述文件的安装与更新 - 38 -
3.3.6 库文件的安装与更新 - 40 -
3.3.7 创建一个示例程序 - 41 -
第四章 故障分析与解决 - 69 -
附录 1：CmpEuraFunctions 组件库说明： - 70 -
附录 1.1 串口自由通讯相关枚举及功能函数 - 71 -
枚举数据定义： - 71 -
结构体数据定义： - 72 -
函数定义： - 72 -
附录 1.2 系统温度监控功能块 - 76 -
附录 2：Modbus RTU 使用说明： - 77 -
概述： - 77 -
Modbus RTU 主站及其实现： - 77 -
Modbus RTU 从站及其实现： - 81 -
附录 3 Modbus TCP 使用说明 - 86 -
概述 - 86 -
Modus TCP 主站及其实现 - 86 -
Modus TCP 从站及其实现 - 90 -
敬告用户： - 93 -

第一章　　概述

本章简要介绍了 EAC100 系列运动控制器主要特点。主要内容为：产品特点、产品型号命名规范、产品技术参数和使用产品的注意事项等，有助于用户初步了解产品的构成和使用规范。

1.1 EAC100 系列运动控制器简述

EAC100 系列运动控制器是由欧瑞传动电气股份有限公司研发的经济型软 PLC 型运动控制器系列，支持 LD、ST、IL、FBD、SFC 和 CFC 等 6 种编程语言，符合 IEC 61131-3 标准。EAC100 系列提供了控制和可视化的一体化软件开发运行环境，并集成强大的运动控制功能。内置的高性能 EtherCAT 主站，同步时间误差<1μs。

优秀的控制性能和开放式的软件架构，使 EAC100 系列为工业自动化领域提供了一个良好的控制系统开发平台。

该系列产品具有如下特点

1. 标准化的软 PLC 开发平台

支持 LD,ST,IL,SFC,FBD 和 CFC 等 6 种编程语言，符合 IEC 61131-3 标准，支持 CODESYS

2. 丰富强大的控制功能

集成逻辑控制、运动控制和可视化功能，包含直线插补、圆弧插补、电子齿轮、电子凸轮和 CNC、机器人控制等功能模块

3. 控制与可视化的集成式开发环境

在同一平台下完成控制和可视化程序开发，可极大简化传统的“HMI+PLC”的开发模式

4. 内置高性能 EtherCAT 主站

非 DC 模式的最小通信周期 1ms；DC 模式的最小通信周期 2ms，节点间的同步时间误差<1μs

5. 开放式软硬件架构

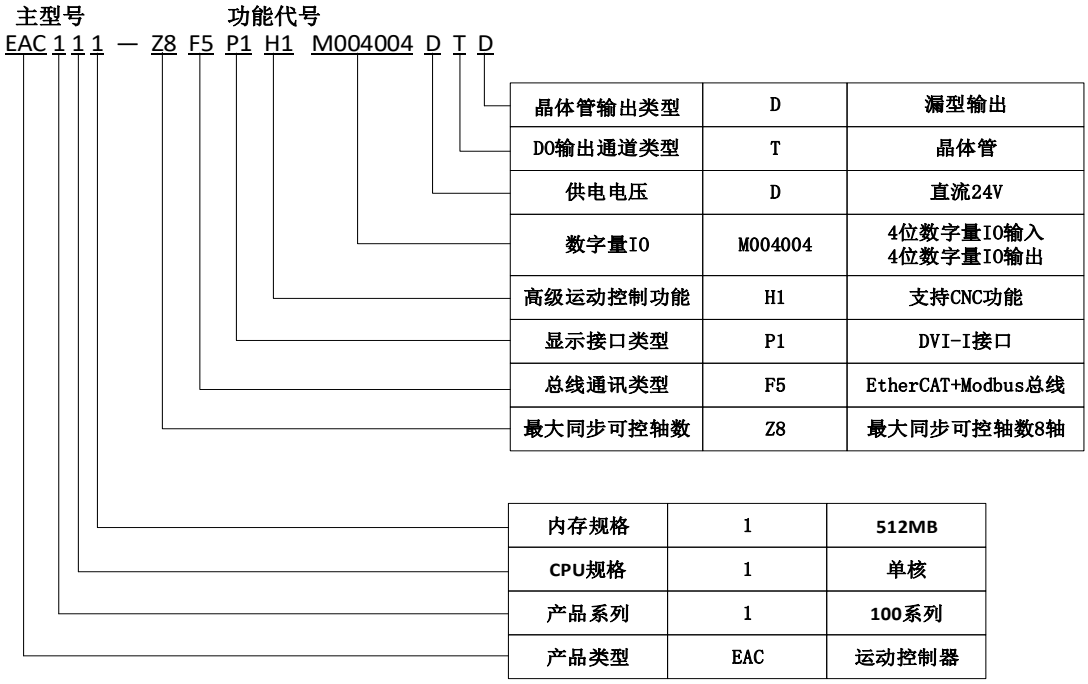
面向 OEM 用户提供计算机高级编程语言开发接口和软硬件定制服务，支持 Visual Studio 开发环境

6. 丰富的接口和协议栈

EAC100 提供 RJ45、RS232、RS485、DVI 和数字 IO 等工业通用接口，并内置 EtherCAT 主站、Modbus RTU 主站/从站，能够轻松便捷的与其他设备进行数据交互。

1.2 EAC100 系列运动控制器命名规范及订货信息

EAC100 系列运动控制器产品命名分为两部分：主型号和功能代号。其中主型号包括产品系列名称、CPU 规格和内存规格；功能代号包括最大同步控制轴数、高级软件支持等一系列功能的描述，详细可以在功能代号表中查询。标准的产品命名如下图所示。



EAC100 系列产品命名示例

主型号中各位置含义请参照下表：

功能名称	功能代码	含义
CPU 类型	1	单核 CPU
	2	双核 CPU
	4	4 核 CPU
内存规格	1	512MB
	2	1GB
	3	2GB

功能代码中各位置含义请参照下表：

功能名称	功能代码	含义
最大同步控制轴数	Z8	最大同步控制轴数为 8 轴
	Z16	最大同步控制轴数为 16 轴
	Z32	最大同步控制轴数为 32 轴
总线通讯类型	F5	EtherCAT+Modbus
	F13	EtherCAT
显示接口类型	P1	DVI-I 接口
	P3	HDMI 接口
高端运动控制功能	H1	支持 CNC 和 Robot 功能
数字量 IO	M004004	支持 4 位数字 IO 输入，4 位数字 IO 输出
	M020020	支持 20 位数字 IO 输入，20 位数字 IO 输出
	M036036	支持 36 位数字 IO 输入，36 位数字 IO 输出
	M068068	支持 68 位数字 IO 输入，68 位数字 IO 输出
供电电压	D	直流 24V
DO 输出通道类型	T	晶体管
晶体管输出类型	D	漏形输出

现有产品的订货信息如下表所示：

订货号	规格	备注
EAC111-Z8F5P1H1M004004DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus / DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 4 位数字 IO 输入 / 4 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC111-Z16F5P1H1M004004DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 4 位数字 IO 输入 / 4 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC112-Z8F5P3H1M004004DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus / HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 4 位数字 IO 输入 / 4 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z16F5P3H1M004004DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 4 位数字 IO 输入 / 4 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z32F5P3H1M004004DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 32 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 4 位数字 IO 输入 / 4 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC111-Z8F5P1H1M020020DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus/ DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 20 位数字 IO 输入 / 20 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC111-Z16F5P1H1M020020DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 20 位数字 IO 输入 / 20 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC112-Z8F5P3H1M020020DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 20 位数字 IO 输入 / 20 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z16F5P3H1M020020DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 20 位数字 IO 输入 / 20 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z32F5P3H1M020020DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 32 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 20 位数字 IO 输入 / 20 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC111-Z8F5P1H1M036036DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus/ DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 36 位数字 IO 输入 / 36 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC111-Z16F5P1H1M036036DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 36 位数字 IO 输入 / 36 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC112-Z8F5P3H1M036036DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 36 位数字 IO 输入 / 36 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z16F5P3H1M036036DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 36 位数字 IO 输入 / 36 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z32F5P3H1M036036DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 32 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 36 位数字 IO 输入 / 36 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC111-Z8F5P1H1M068068DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 68 位数字 IO 输入 / 68 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC111-Z16F5P1H1M068068DTD	单核 CPU / 512MB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ DVI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 68 位数字 IO 输入 / 68 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

订货号	规格	备注
EAC112-Z8F5P3H1M068068DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 8 轴 / EtherCAT+Modbus HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 68 位数字 IO 输入 / 68 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z16F5P3H1M068068DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 16 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 68 位数字 IO 输入 / 68 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	
EAC112-Z32F5P3H1M068068DTD	单核 CPU / 1GB 内存/ 最大同步控制轴数 32 轴 / EtherCAT+Modbus/ HDMI 显示输出接口 / 支持 CNC 和 Robot 功能/ 供电电压 DC24V / 68 位数字 IO 输入 / 68 位数字 IO 输出 / IO 输出类型为晶体管漏型输出	

1.3 EAC100 系列运动控制器技术规格及工作环境参数

EAC100 系列运动控制器技术规格及工作环境参数如下表所示：

技术参数	规格
工作电源（直流）	DC24V±15%，>3.2A
工作温度	-10℃～50℃
存储温度	-25℃～70℃
相对湿度	<95%无冷凝
防护等级	IP20
工作环境	无水滴、蒸汽、腐蚀、易燃、灰尘及金属微粒的场所

1.4 安全注意事项

本节对与本系列产品相关的安全注意事项进行说明。如果不遵守这些注意事项，可能会导致死亡或重伤、并损坏本产品、相关机器及系统。因未遵守本使用说明书的内容而造成的伤害和设备损坏，本公司将不负任何责任。

1.4.1 安全信息定义

危险：如不遵守相关要求，就会造成严重的人身伤害，甚至死亡。

警告：如不遵守相关要求，可能会造成人身伤害或者设备损坏。

注意：为了确保正确的运行而采取的步骤。

培训并合格的专业人员：是指操作本设备的工作人员必须经过专业的电气培训和安全知识培训并且考试合格，已经熟悉本设备的安装，调试，投入运行以及维护保养的步骤和要求，并能避免产生各种紧急情况。

1.4.2 警告标示

警告用于对可能造成严重的人身伤亡或设备损坏的情况进行警示，给出建议以避免发生危险。本手册中使用下列警告标识：

标识	名称	说明	简写
危险	危险	如不遵守相关要求，就会造成严重的人身伤害，甚至死亡。	
警告	警告	如不遵守相关要求，可能会造成人身伤害或者设备损坏。	
注意	注意	为了确保正确的运行而采取的步骤。	注

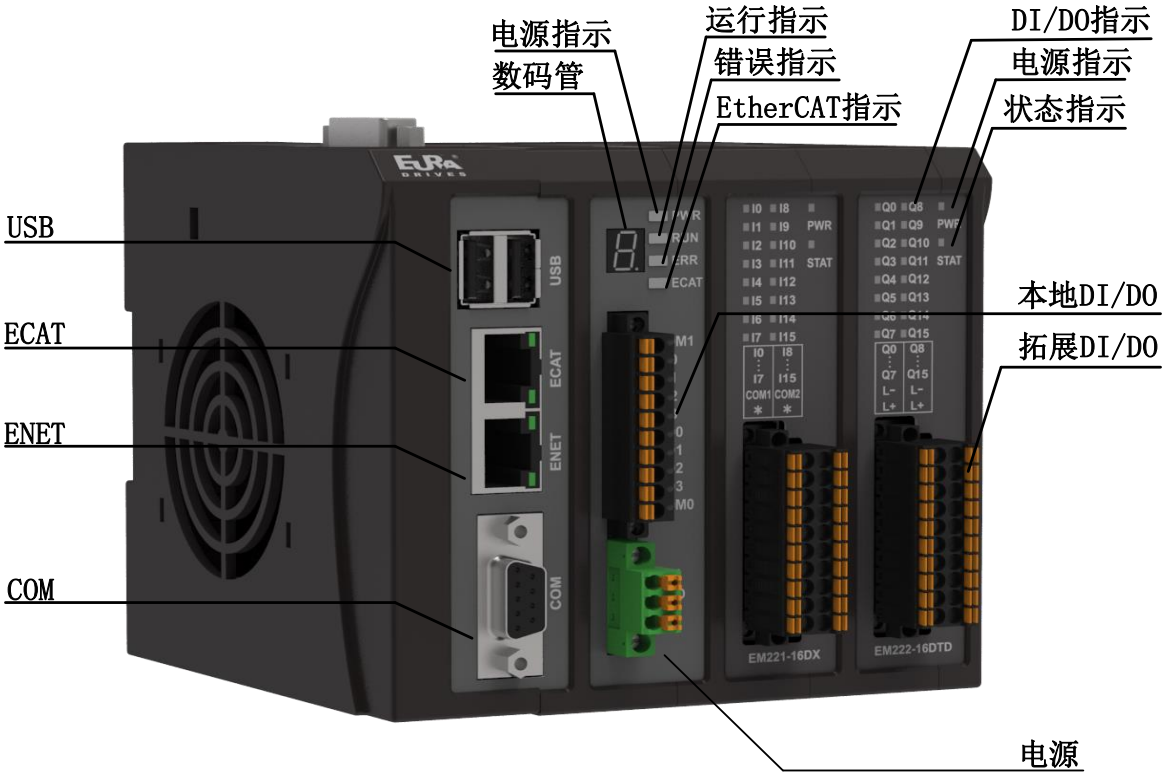
1.4.3 安全指导

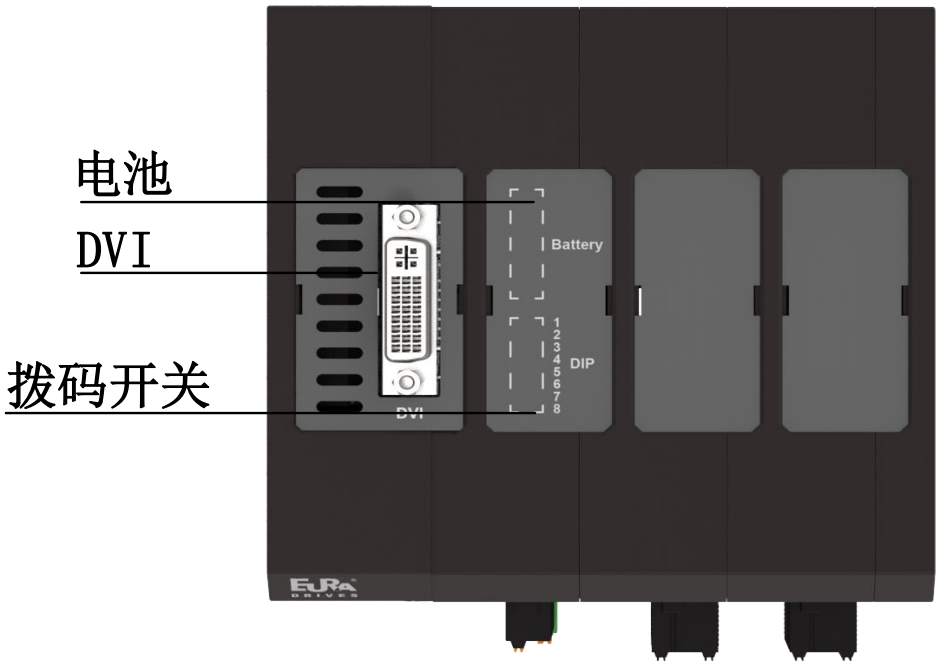
	◇只有经过培训并合格的人员才允许进行相关操作。 ◇禁止在电源接通的情况下进行接线，检查和更换器件等作业。进行接线及检查之前，必须确认所有输入电源已经断开
	◇未经授权严禁对运动控制器进行的改装和拆卸，否则可能引起火灾，触电或其他伤害。 ◇禁止将运动控制器安装在易燃物上，并避免运动控制器紧密接触或粘附易燃物。 ◇客户收到产品后，请检查控制器有无外壳损坏，包装箱内有无水渍，如有请联系当地经销商或者当地办事处。

第二章　　运动控制器硬件介绍

2.1 控制器外形结构

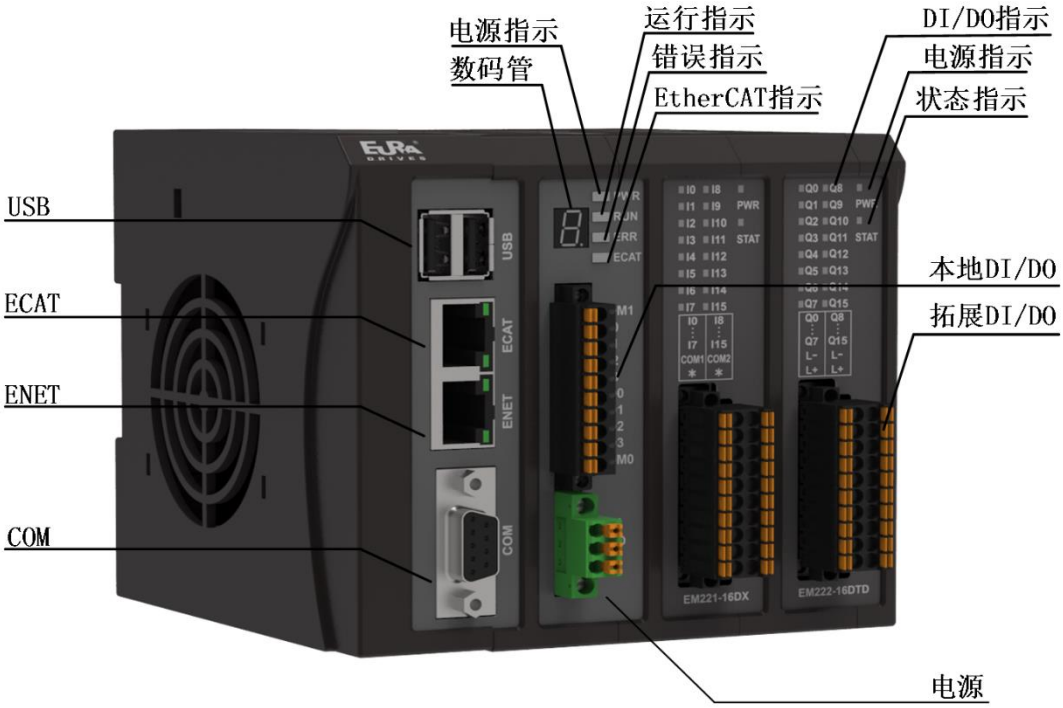
2.1.1 EAC111 系列

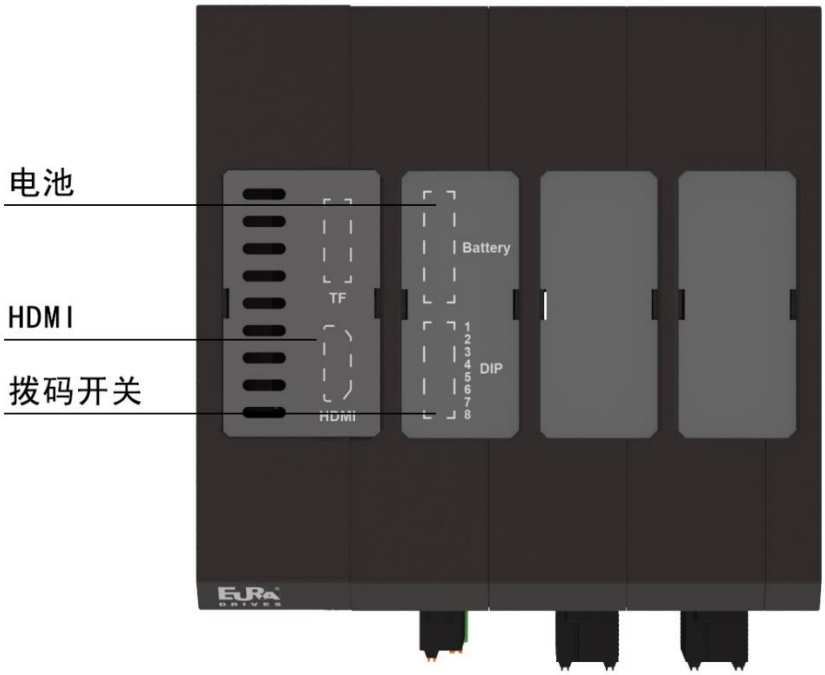




EAC111 系列运动控制器整体视图及结构名称

2.1.2 EAC112 系列

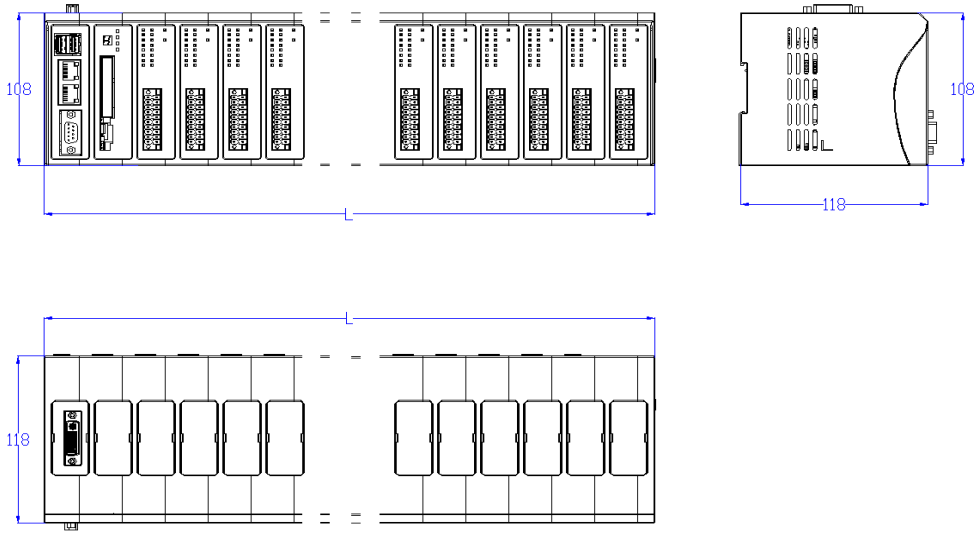




EAC112 系列运动控制器整体视图及结构名称

2.2 外形尺寸及安装方式

2.2.1 外形尺寸



注：L=68+24*N（N=模块数）　　单位：mm

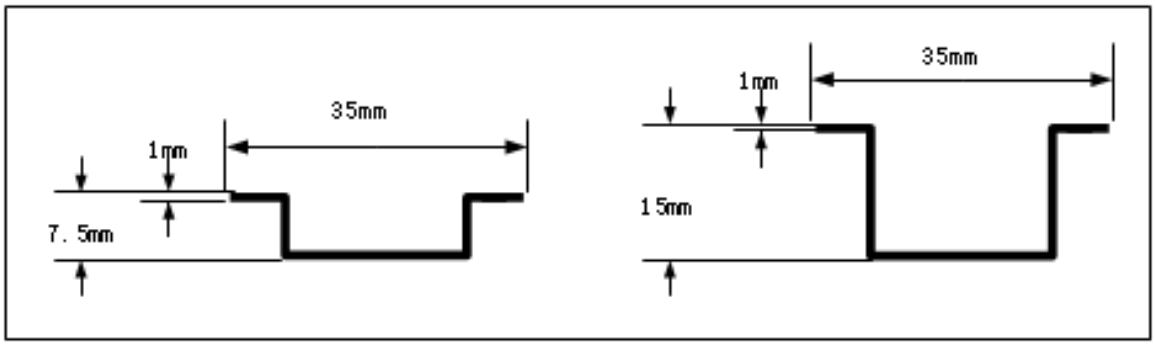
型号	L	W	H	建议安装卡扣数量
EAC111-Z8F5P1H1M004004DTD	68	118	108	2
EAC111-Z16F5P1H1M004004DTD				
EAC112-Z8F5P3H1M004004DTD				
EAC112-Z16F5P3H1M004004DTD				
EAC112-Z32F5P3H1M004004DTD				
EAC111-Z8F5P1H1M020020DTD	116	118	108	2
EAC111-Z16F5P1H1M020020DTD				
EAC112-Z8F5P3H1M020020DTD				
EAC112-Z16F5P3H1M020020DTD				
EAC112-Z32F5P3H1M020020DTD				
EAC111-Z8F5P1H1M036036DTD	164	118	108	3
EAC111-Z16F5P1H1M036036DTD				
EAC112-Z8F5P3H1M036036DTD				

EAC112-Z16F5P3H1M036036DTD				
EAC112-Z32F5P3H1M036036DTD				
EAC111-Z8F5P1H1M068068DTD	260	118	108	4
EAC111-Z16F5P1H1M068068DTD				
EAC112-Z8F5P3H1M068068DTD				
EAC112-Z16F5P3H1M068068DTD				
EAC112-Z32F5P3H1M068068DTD				

2.2.2 控制器安装方式

使用 DIN 导轨安装步骤

①准备好标准的 35mm 宽 DIN 导轨，有两种规格，如下图所示。



DIN 导轨示意图

②将导轨安装至需要的位置，将控制器卡接于导轨上。

方法：将控制器底部的 35mm 导轨卡接滑块拉下，从导轨的上部装入控制器，向前推控制器下部直到控制器紧贴导轨，然后再将卡接滑块推到原位即可。

注：为保证良好的散热和通风要求，控制器安装时四周应各留出至少 60mm 空间。

2.3 运动控制器本体部分

2.3.1 控制器本体运行状态指示及系统工作状态

控制器运行状态指示包括数码管和电源（PWR）、运行（RUN）、故障（ERR）、通讯（ECAT）四个状态指示灯，如下图所示。

设备的各个工作状态都可以通过设备上的数码管和指示灯进行显示，系统的各个工作状态及显示如下表所示。

序号	描述	数码管显示	指示灯显示				备注
			PWR	RUN	ERR	ECAT	
1	系统启动	8.	蓝色转绿色	不亮	不亮	不亮	
2	配置状态	2	绿色常亮	蓝色常亮	不亮	不亮	
3	系统程序更新中	3	绿色常亮	蓝色闪烁	不亮	不亮	
4	Runtime 启动中	4	绿色常亮	红色闪烁	不亮	不亮	
5	Runtime 启动完成，但无 PLC 程序	5	绿色常亮	红色常亮	不亮	不亮	
6	PLC 程序正常运行	6	绿色常亮	绿色常亮	不亮	N/A	ECAT 灯状态取决于程序中 EtherCAT 总线运行状态，正常运行为绿色常亮
7	PLC 程序停止	7	绿色常亮	红色常亮	不亮	N/A	ECAT 灯状态取决于程序中 EtherCAT 总线运行状态，正常运行为绿色常亮

2.3.2 DIP 拨码开关

DIP 拨码开关用于在系统配置阶段对系统功能进行配置。使用时应在系统上电前将对应拨码切换到需求位置，随后正常上电即可。各拨码位定义如下表所示：

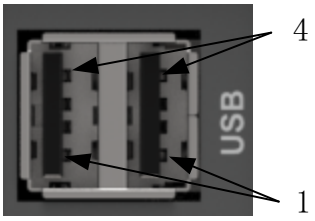
DIP 编号	拨码功能
1	ON：系统启动后自动运行 CODESYS 程序； OFF：启动后保持配置阶段
2	ON：当 DIP5 为 ON 时清除 PLC 程序
3	ON：当 DIP5 为 ON 时清除掉电保持数据
4	ON：屏蔽 RTC 时钟复位错误
5	ON：DIP2、DIP3、DIP8 功能有效
6	保留待用
7	保留待用
8	ON：复位与上位机通讯网卡的 IP 地址和 MAC 地址

2.3.3 控制器本体对外接口

2.3.3.1 USB 接口

EAC100 系列运动控制器共有 2 个 USB2.0 接口，支持标鼠标、键盘、U 盘等标准 USB 设备，并可以通过 HUB 进行扩展，USB 通讯口定义如下表所示：

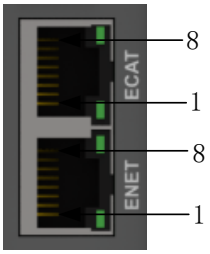
引脚号	描述	信号
1	+5V	VCC
2	Data-	D-
3	Data+	D+
4	电源地	GND



2.3.3.2 网络接口（RJ45）

EAC100 系列运动控制器带有 2 个 RJ45 接口，ECAT 接口用于连接 EtherCAT 从站设备，ENET 连接上位机，用于程序下载、在线监控、网页配置等功能，接口定义如下表所示：

引脚号	描述	信号
1	数据发送正端	TX+
2	数据发送负端	TX-
3	数据接收正端	RX+
6	数据接收负端	RX-

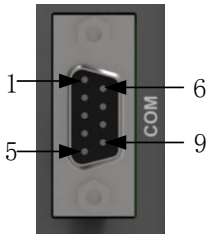


2.3.3.3 串行接口（COM）

EAC100 系列运动控制器配备有一个 RS232 和一个 RS485 串行通讯接口，两个接口共用一个 DB9 母型接口。该接口可以用作与第三方设备的通讯口。当采用屏蔽电缆时，RS232 通讯的距离建议不超过 15 米，RS485 通讯的最大距离可达 1000 米。串行接口的定义
如下表所示：

RS-232		
孔号	描述	信号
2	接收数据	RXD
3	发送数据	TXD
5	信号地	GND

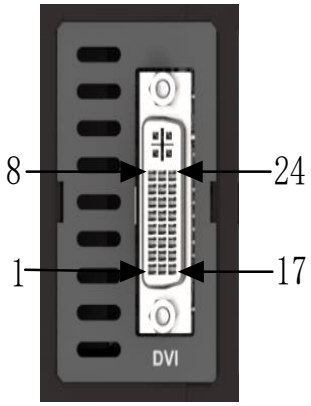
RS-485		
孔号	描述	信号
7	RS485+	A+
8	RS485-	B-



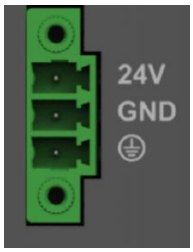
2.3.3.4 显示接口（DVI-I）

EAC111 系列运动控制器配备有一个标准的 DVI-I 接口，可以连接带有 DVI 接口的显示设备进行显示和人机交互。DVI 接口定义如下表：

引脚号	信号	引脚号	信号
1	TMDS_DATA2_N	17	TMDS_DATA0_N
2	TMDS_DATA2_P	18	TMDS_DATA0_P
3	GND	19	GND
4~8	空	20，21	空
9	TMDS_DATA1_N	22	GND
10	TMDS_DATA1_P	23	TMDS_CLK_P
11	GND	24	TMDS_CLK_N
12~16	空		



2.3.3.5 电源接口

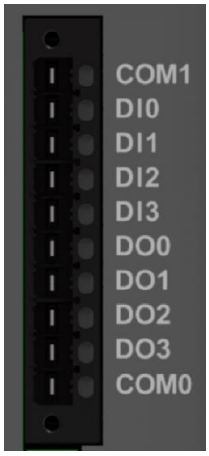


名称	描述
24V	24V 电源正
GND	24V 电源负
⏚	大地

2.3.3.6 数字 IO 接口（DI/DO）

EAC100 系列运动控制器本体配备有 4 路数字量输入和 4 路数字量输出。其接口定义 如下所示：

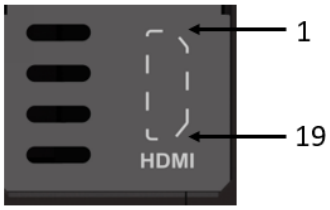
信号	描述
COM1	输入公共端
DI0	输入端口 0
DI1	输入端口 1
DI2	输入端口 2
DI3	输入端口 3
DO0	输出端口 0
DO1	输出端口 1
DO2	输出端口 2
DO3	输出端口 3
COM0	输出公共端



2.3.3.7 HDMI 接口

EAC112 系列运动控制器本体配备有 1 个 HDMI 输出口。其接口定义如下：

引脚号	信号	引脚号	信号
1	HOT_PLUG_DET	11	TMDS_DATA0_N
2	+5V	12	GND
3	GND	13	TMDS_DATA0_P
4	SDA	14	TMDS_DATA1_N
5	SCL	15	GND
6~7	空	16	TMDS_DATA1_P
8	TMDS_CLK_N	17	TMDS_DATA2_N
9	GND	18	GND
10	TMDS_CLK_P	19	TMDS_DATA2_P



2.3.4 控制器输入（DI）

运动控制器本体提供 4 路 DI 通道，用于普通的数字量（开关量）输入。各输入通道与内部 CPU 电路之间均有电气隔离，并有状态指示灯指示各通道的输入状态。

DI 输入通道主要特点：

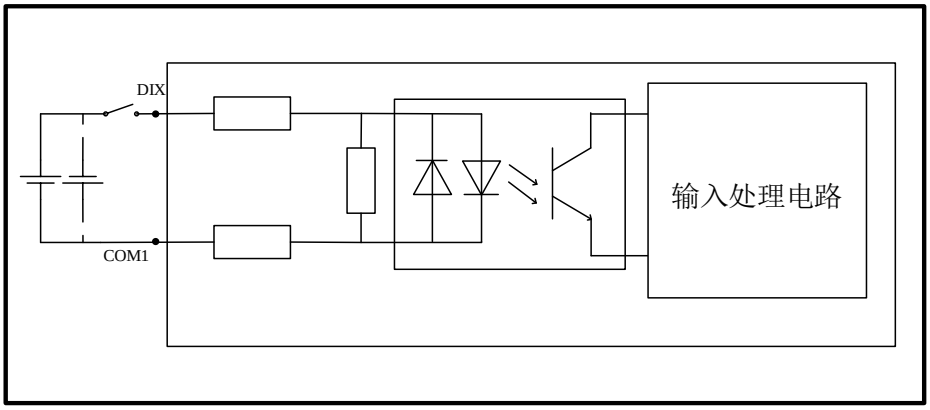
- 4 路晶体管输入通道，共分成 1 组
- 各组既可接源型输入（共阴极），也可以接漏型输入（共阳极）
- 额定输入电压为 DC24V
- 现场信号与内部电路之间有电气隔离器
- 各通道有独立的状态指示灯

DI 通道技术参数：

项目	规格
输入类型	源型/漏型可选
额定输入电压	24VDC
额定输入电流	4.1mA@24VDC
最大输入电压	30VDC
逻辑 1 最小输入电压	15V@2.5mA
逻辑 0 最大输入电压	2.8V@0.7mA
输入与内部逻辑电路的隔离方式	光电耦合器
输入与内部逻辑电路的隔离电压	1500VAC/1 分钟

状态指示	状态指示灯亮绿色/红色

DI 输入通道电气原理图：



2.3.5 控制器输出（DO）

运动控制器本体提供 4 路 DO 通道，用于普通的数字量（开关量）输出。各输出通道与内部电路之间均有电气隔离，并有状态指示灯指示各输出通道的通断状态。

晶体管型 DO 输出通道主要特点:

- 4 路晶体管输出通道，分成 1 组
- 额定供电电压为 DC24V
- 额定输出电压为 DC24V，每通道最大输出电流为 500mA，漏型（NPN）
- 供电电源接入极性保护
- 感性负载输出保护
- 短路保护（每路输出电流大于 500mA 时）
- 同一组内通道允许并联
- 输出与内部电路之间光电隔离

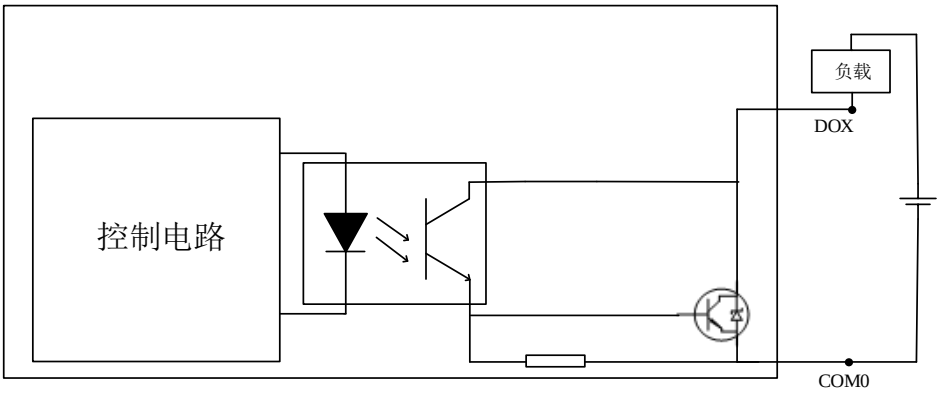
晶体管型 DO 通道技术参数

项目	规格
输出类型	漏型[注]
额定输出电压	24VDC
额定供电电压	24VDC
电源接入极性保护	有

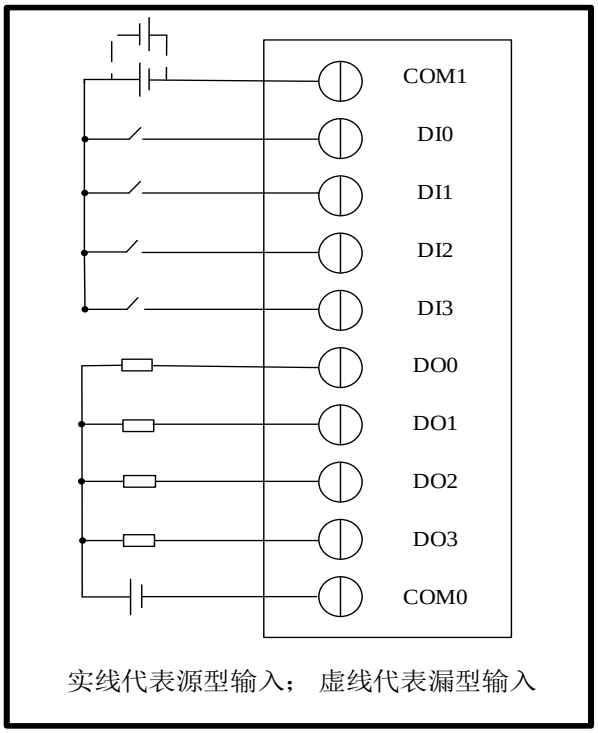
每通道输出电流	最大 500mA @24VDC
输出漏电流	25μA（最大）
输出与内部逻辑电路的隔离方式	光电隔离
输出与内部逻辑电路的隔离电压	1500VAC/1 分钟
通道并联功能	有
短路保护功能	有
状态指示	状态指示灯亮绿色

注：漏型输出时无限流电阻

DO 输出通道电气原理图：



2.3.6 数字量 DI/DO 接线图



2.4 运动控制器数字量拓展 IO 部分

2.4.1 控制器数字量拓展 IO 运行状态指示及系统工作状态

控制器数字量拓展部分运行状态指示包括电源（PWR）、状态（STAT）2 个状态指示灯，如右图所示。

系统上电后各个阶段及其显示状态如下表所示：

序号		状态	描述	备注
1	PWR	绿色常亮	系统电源正常	
2		不亮	系统电源异常	
3	STAT	不亮	停止状态	
4		绿色闪烁	初始化状态，地址分配完成	
5		绿色常亮	正常运行状态	
6		红色闪烁	通信超时或地址分配错误	

2.4.2 DI 16*DC24

该模块的型号为：EM221-16XD

EM221-16XD 提供 16 路 DI 通道，用于普通的数字量（开关量）输入。各输入通道与内部 CPU 电路之间均有电气隔离，并有状态指示灯指示各通道的输入状态。

注：EM221-16XD 不可单独订货；

信号	描述	信号	描述
I0	输入端口 0	I8	输入端口 8
I1	输入端口 1	I9	输入端口 9
I2	输入端口 2	I10	输入端口 10
I3	输入端口 3	I11	输入端口 11
I4	输入端口 4	I12	输入端口 12
I5	输入端口 5	I13	输入端口 13
I6	输入端口 6	I14	输入端口 14
I7	输入端口 7	I15	输入端口 15
COM1	输入公共端 1	COM2	输入公共端 2
*	未定义	*	未定义

DI 输入通道主要特点：

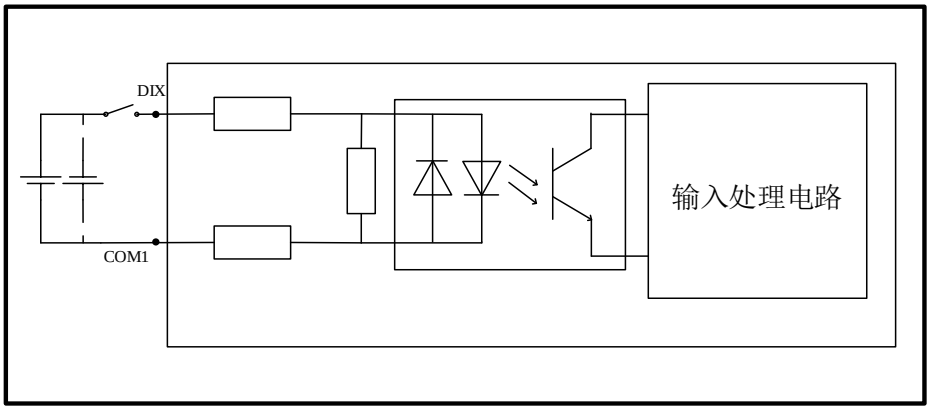
- 16 路晶体管输入通道，共分成 2 组（I0~I7 为一组，共用公共端 COM1, I8~I15 为一组，共用公共端 COM2）
- 各组既可接源型输入（共阴极），也可以接漏型输入（共阳极）
- 额定输入电压为 DC24V
- 现场信号与内部电路之间有电气隔离器
- 各通道有独立的状态指示灯

DI 通道技术参数：

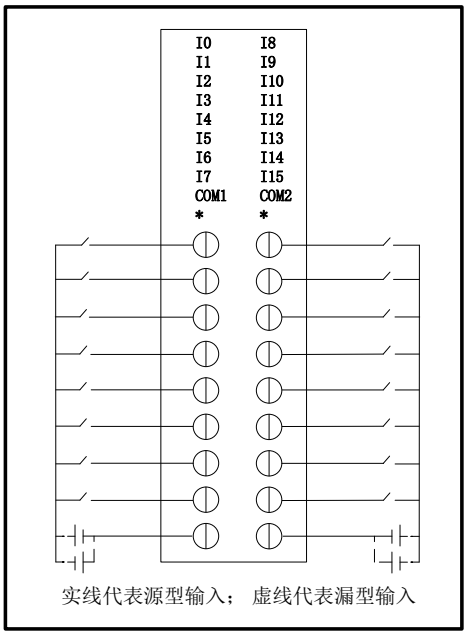
项目	规格
输入类型	源型/漏型可选
额定输入电压	24VDC
额定输入电流	4.1mA@24VDC
最大输入电压	30VDC
逻辑 1 最小输入电压	15V@2.5mA

逻辑 0 最大输入电压	2.8V@0.7mA
输入与内部逻辑电路的隔离方式	光电耦合器
输入与内部逻辑电路的隔离电压	1500VAC/1 分钟
状态指示	状态指示灯亮绿色/红色

DI 输入通道电气原理图：



接线图：



2.4.3 DO 16*DC24

该模块的型号为：EM222-16DTD

EM222-16DTD 提供 16 路 DO 通道，用于普通的数字量（开关量）输出。各输出通道与内部电路之间均有电气隔离，并有状态指示灯指示各输出通道的通断状态。

注：EM222-16DTD 不可单独订货；

信号	描述	信号	描述
Q0	输出端口 0	Q8	输入端口 8
Q1	输出端口 1	Q9	输入端口 9
Q2	输出端口 2	Q10	输入端口 10
Q3	输出端口 3	Q11	输入端口 11
Q4	输出端口 4	Q12	输入端口 12
Q5	输出端口 5	Q13	输入端口 13
Q6	输出端口 6	Q14	输入端口 14
Q7	输出端口 7	Q15	输入端口 15
L-	24V 电源负	L-	24V 电源负
L+	24V 电源正	L+	24V 电源正

注：L+、L-为外接电源，接 1 组即可；

晶体管型 DO 输出通道主要特点:

- 16 路晶体管输出通道
- 额定供电电压为 DC24V
- 额定输出电压为 DC24V，每通道最大输出电流为 500mA，漏型（NPN）
- 供电电源接入极性保护
- 感性负载输出保护
- 短路保护（每路输出电流大于 500mA 时）
- 同一组内通道允许并联
- 输出与内部电路之间光电隔离

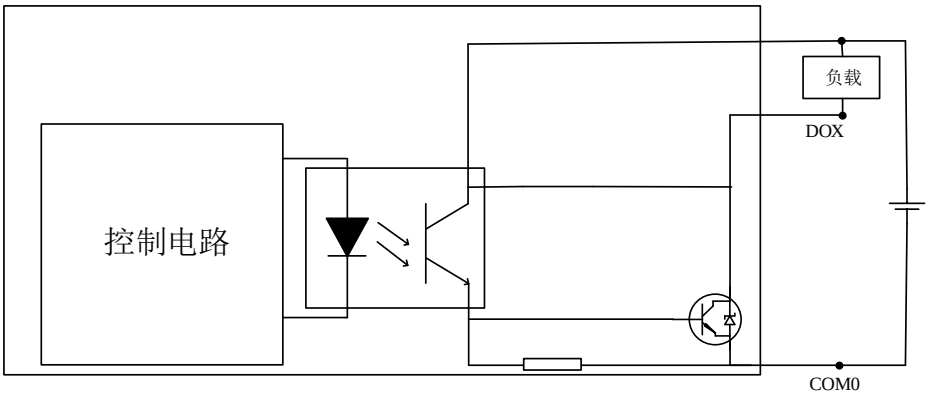
晶体管型 DO 通道技术参数

项目	规格
输出类型	漏型[注]
额定输出电压	24VDC
额定供电电压	24VDC
电源接入极性保护	有
每通道输出电流	最大 500mA @24VDC

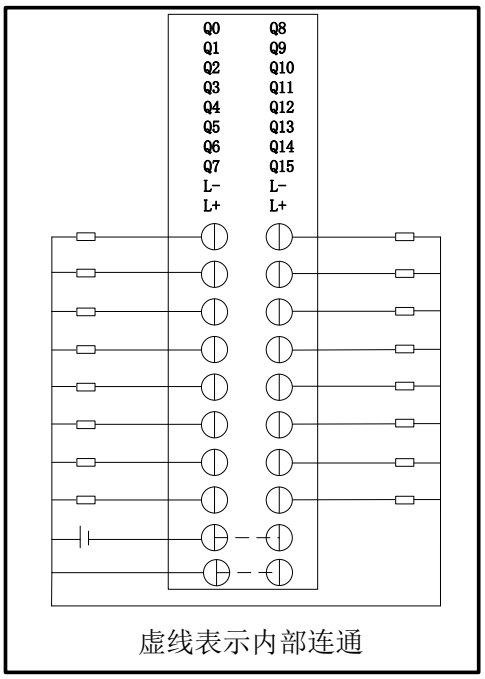
输出漏电流	25μA（最大）
输出与内部逻辑电路的隔离方式	光电隔离
输出与内部逻辑电路的隔离电压	1500VAC/1 分钟
通道并联功能	有
短路保护功能	有
状态指示	状态指示灯亮绿色

注：漏型输出时无限流电阻

DO 输出通道电气原理图：



接线图：



2.4.4 DI 8*DC24+DO 8*DC24

该模块的型号为：EM223-16DTD

EM223-16DTD 供 8 路 DI 通道和 8 路 DO 通道，用于普通的数字量（开关量）输入、输出。各输入通道与内部 CPU 电路之间均有电气隔离，并有状态指示灯指示各通道的输入状态。各输出通道与内部电路之间均有电气隔离，并有状态指示灯指示各输出通道的通断状态。

注：EM223-16DTD 不可单独订货；

信号	描述	信号	描述
I0	输入端口 0	Q0	输出端口 0
I1	输入端口 1	Q1	输出端口 1
I2	输入端口 2	Q2	输出端口 2
I3	输入端口 3	Q3	输出端口 3
I4	输入端口 4	Q4	输出端口 4
I5	输入端口 5	Q5	输出端口 5
I6	输入端口 6	Q6	输出端口 6
I7	输入端口 7	Q7	输出端口 7
COM1	输入公共端 1	L-	24V 电源负
*	未定义	L+	24V 电源正

注：L+、L-为外接电源

DI 输入通道主要特点：

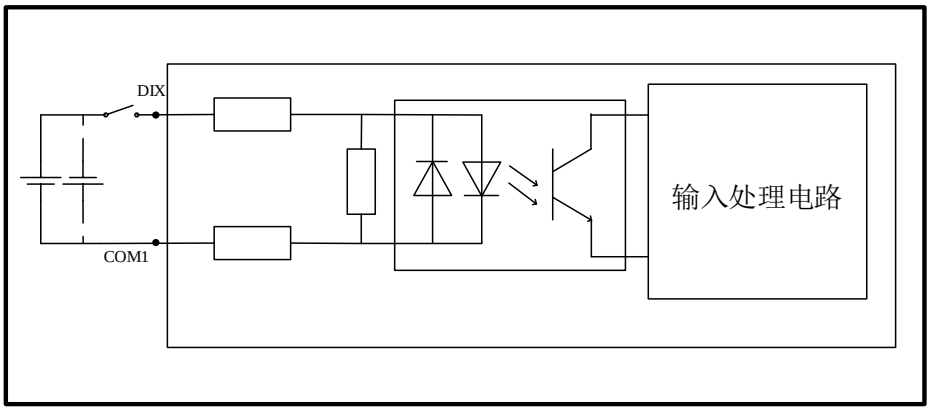
- 8 路晶体管输入通道，共分成 1 组，共用 COM1 端口
- 各组既可接源型输入（共阴极），也可以接漏型输入（共阳极）
- 额定输入电压为 DC24V
- 现场信号与内部电路之间有电气隔离器
- 各通道有独立的状态指示灯

DI 通道技术参数：

项目	规格
输入类型	源型/漏型可选
额定输入电压	24VDC
额定输入电流	4.1mA@24VDC
最大输入电压	30VDC

逻辑 1 最小输入电压	15V@2.5mA
逻辑 0 最大输入电压	2.8V@0.7mA
输入与内部逻辑电路的隔离方式	光电耦合器
输入与内部逻辑电路的隔离电压	1500VAC/1 分钟
状态指示	状态指示灯亮绿色/红色

DI 输入通道电气原理图：



晶体管型 DO 输出通道主要特点:

- 8 路晶体管输出通道，分成 1 组
- 额定供电电压为 DC24V
- 额定输出电压为 DC24V，每通道最大输出电流为 500mA，漏型（NPN）
- 供电电源接入极性保护
- 感性负载输出保护
- 短路保护（每路输出电流大于 500mA 时）
- 同一组内通道允许并联
- 输出与内部电路之间光电隔离

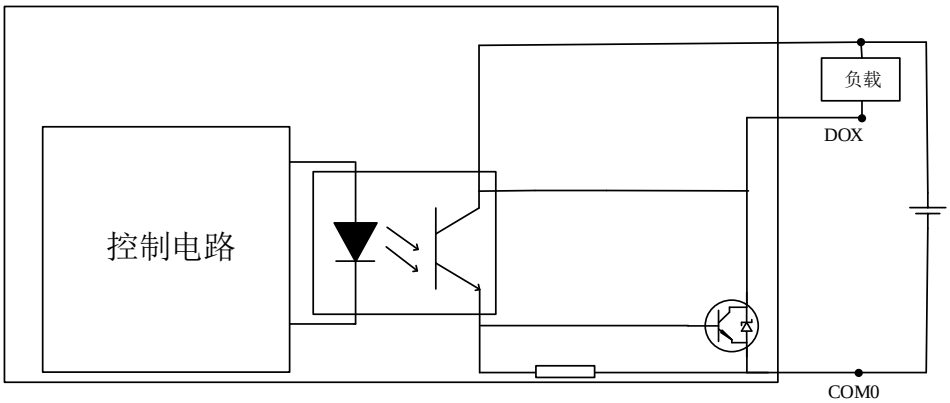
晶体管型 DO 通道技术参数

项目	规格
输出类型	漏型[注]
额定输出电压	24VDC
额定供电电压	24VDC

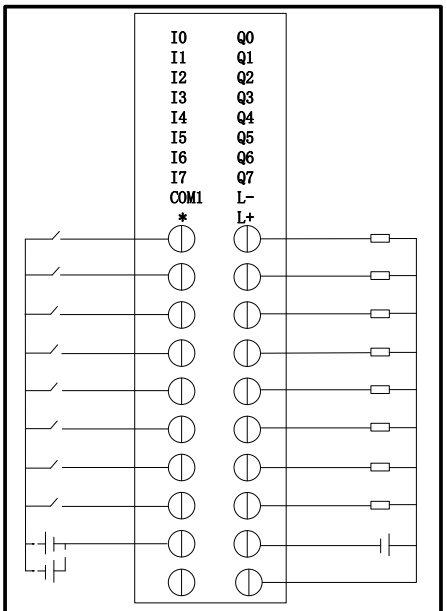
电源接入极性保护	有
每通道输出电流	最大 500mA @24VDC
输出漏电流	25μA（最大）
输出与内部逻辑电路的隔离方式	光电隔离
输出与内部逻辑电路的隔离电压	1500VAC/1 分钟
通道并联功能	有
短路保护功能	有
状态指示	状态指示灯亮绿色

注：漏型输出时无限流电阻

DO 输出通道电气原理图：



接线图



实线代表源型输入； 虚线代表漏型输入

第三章　　控制器软件功能

3.1 概述

本章用于对 EAC 系列运动控制器产品自身功能及其上位机编程软件 CODESYS 进行说明，方便用户快速熟悉并使用设备的软件部分功能。

3.2 控制器软件功能及设置

3.2.1 网页设置功能

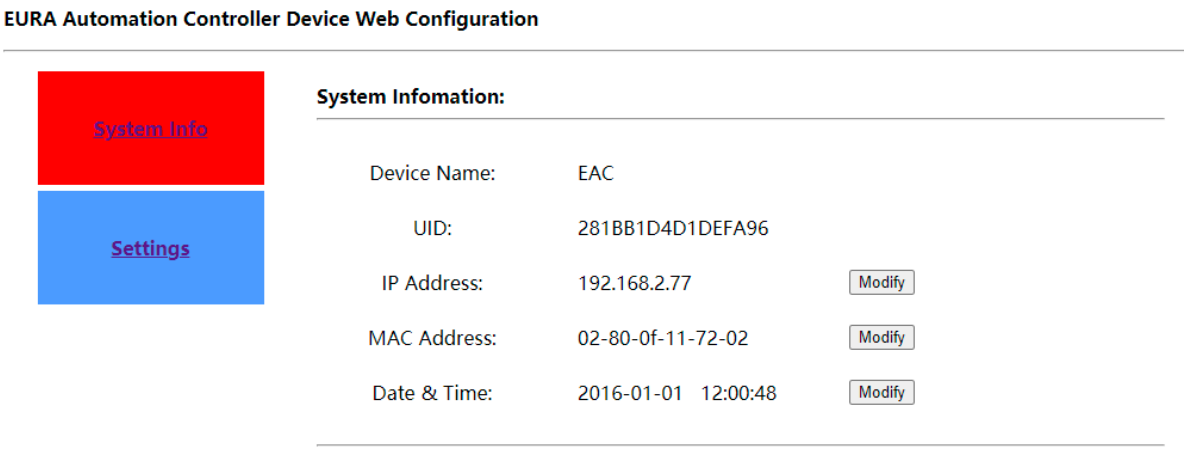
用户可以通过 PC 连接 EAC 设备，并通过浏览器对设备的系统时间、IP 地址等参数进行查看和修改。

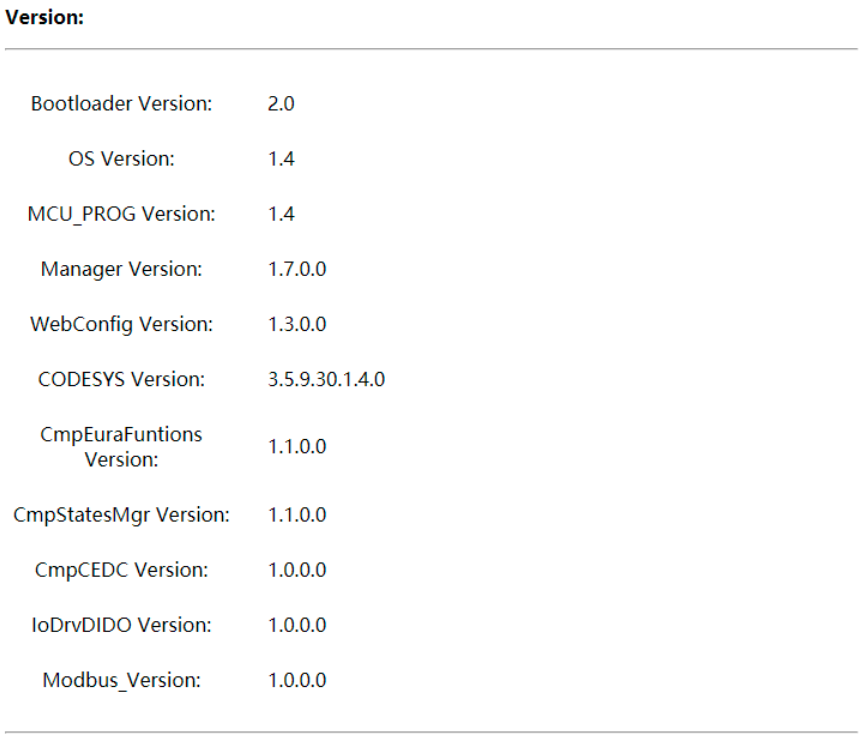
3.2.1.1 连接并进入设置

1. 使用网线连接 PC 机网口和设备的 ENET 网口；
2. 请确认 PC 机网口的 IP 地址与设备的 IP 地址在同一网段上，如不在同一网段请修改 PC 机 IP 地址。

注：设备默认 IP 地址为 192.168.2.77。如果无法确定设备当前 IP 地址，可通过将拨码开关的 DIP8 位置于 ON 状态来使设备的 IP 地址重置为默认 IP 地址。

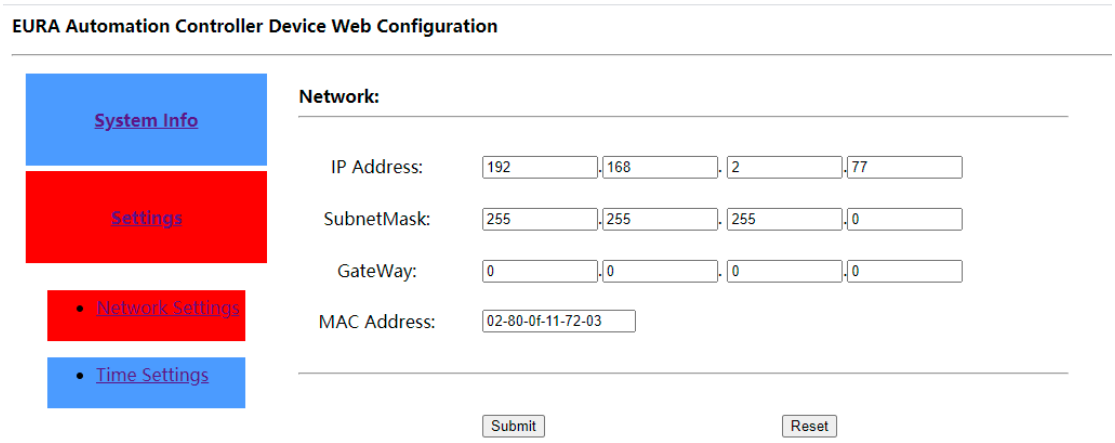
3. 打开浏览器，在浏览器地址栏中输入“设备 IP 地址/config”(默认为 192.168.2.77/config)，点击回车键进入网页设置，
4. 网页设置主界面如下图所示，主界面上包含控制器系统信息和控制器固件、软件的版本号，便于用户了解控制器固件、软件当前的升级情况。





3.2.1.2 修改 IP 地址/MAC 地址

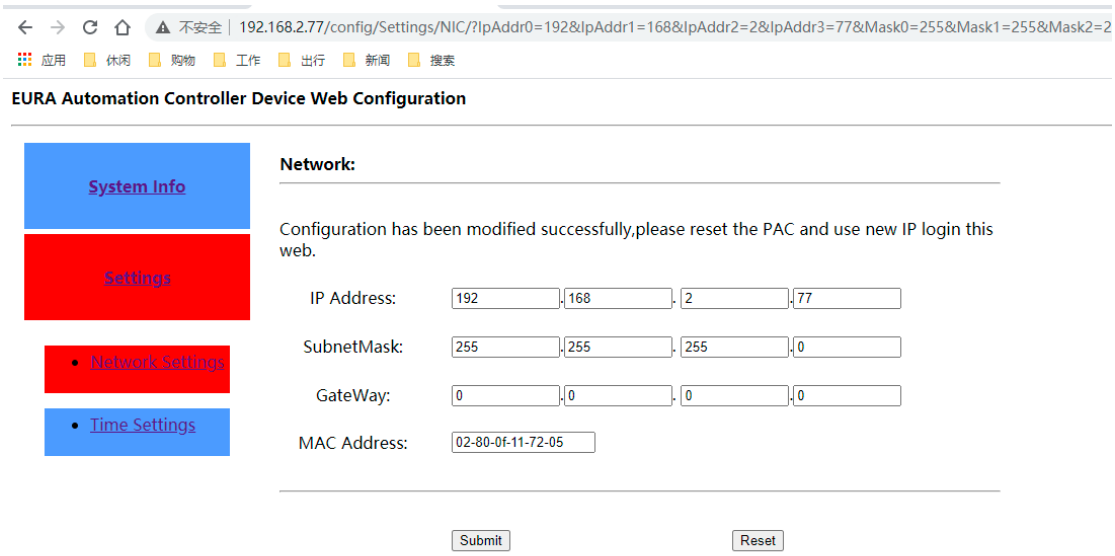
点击左侧的 Settings 按钮或者点击主页 IP 地址或 MAC 地址右侧的 Modify 按钮，可以进入网络设置页面，如下图所示。



修改完成后点击 Submit 按钮，成功修改后系统会提示重启设备使新地址设置生效，如下图所示。

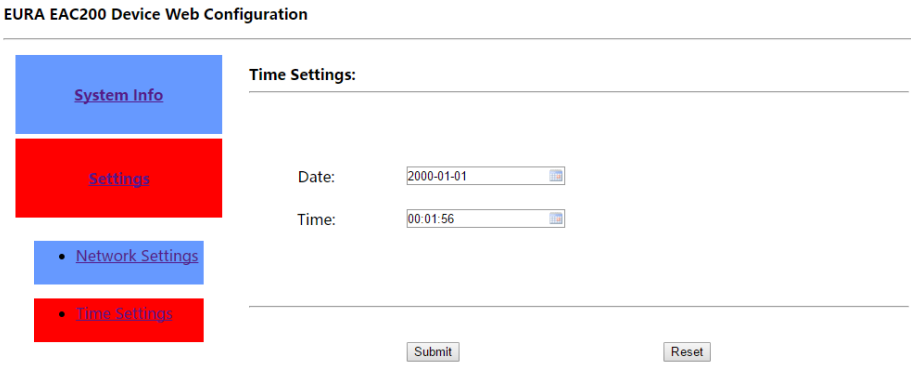
注 1：重启前请检查 IP 地址复位拨码（DIP8）是否置于 OFF 状态，否则新设置的 IP 地址会被还原；

注 2：重启后请使用新设置的 IP 登陆网页设置界面，参见 3.2.1.1 节。

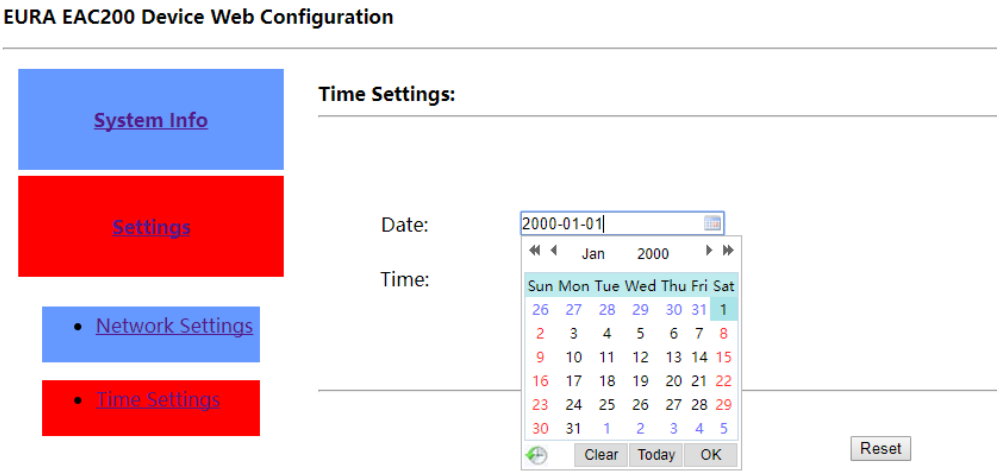


3.2.1.3 修改系统时间

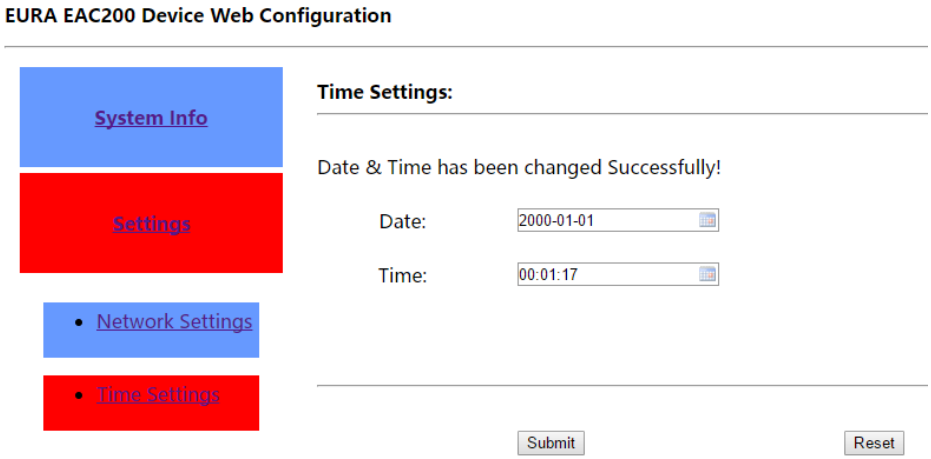
网页设置提供对系统时间的修改，点击时间右侧的 Modify 按钮或者点击左侧 Settings 再点击子菜单中的 Time Settings 即可进入时间设置。



点击日期或者时间文本框，会弹出日期/时间选择对话框，在对话框中选择需要的日期或者时间。



完成日期/时间设置后点击 Submit 提交，提交成功出现对应文字提示。



3.2.2 U 盘更新功能

U 盘更新功能用于对 EAC 设备中的软件进行更新或者重置。

使用时将需要更新的文件或文件夹放入 U 盘根目录，在设备断电状态下将 U 盘插入设备 USB 接口，上电后设备自动运行更新程序，数码管由原状态转为显示“3”。视更新文件大小及数目影响，更新时间为 1-5 秒，更新完成后重新读取 DIP 拨码开关状态并执行相应操作。

注 1：更新前请仔细检查 U 盘根目录下是否只包含需要更新的文件或文件夹，防止将设备中其他软件还原为旧版本。

U 盘更新文件如下表所示：

序号	程序或文件夹名称	说明
1	DeviceUpdate.exe	U 盘更新程序
2	Update	包含有更新文件和更新配置文件的文件夹

3.3 上位机软件 CODESYS

3.3.1 概述

CODESYS 是德国 3S 公司推出的一款基于 IEC-61131 标准的工业自动化开发编程系统，利用 CODESYS 软件可以快速实现对 EAC 设备的编程、程序下载、程序在线监控等功能。

安装 CODESYS 的计算机推荐配置：

操作系统：Windows 7/8/10 (32 位/64 位)；

内存：4GB；

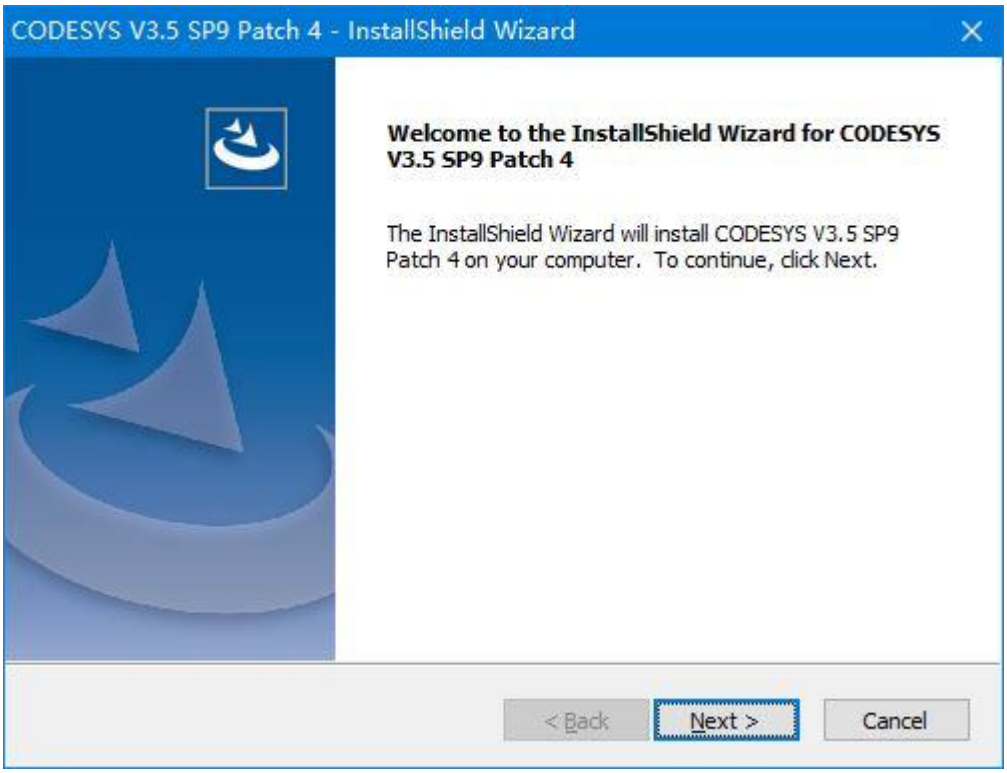
硬盘：2GB；

CPU：双核 CPU

3.3.2 CODESYS 的安装

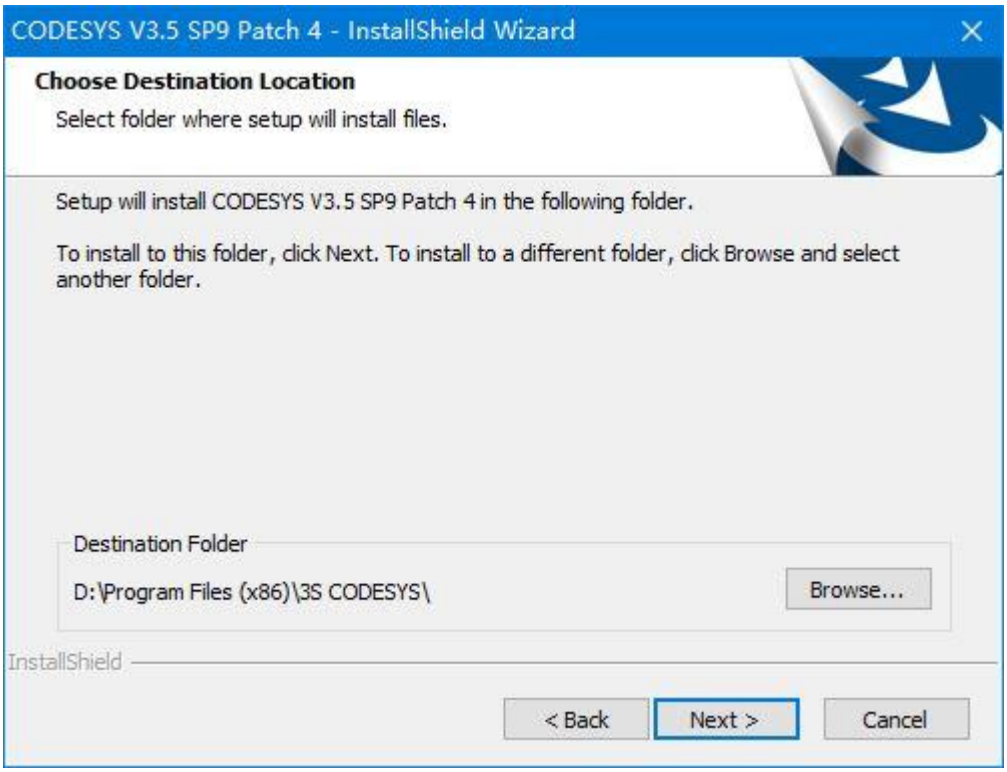
CODESYS 的安装程序可以在 CODESYS 官网进行下载，建议使用版本为 **V3.5 SP11**。

CODESYS 安装程序是一个标准的安装向导，**建议安装前关闭 360 等软件，避免禁用 CODESYS 相关服务。**双击 CODESYS 安装程序图标，打开 CODESYS 安装向导，点击 Next。

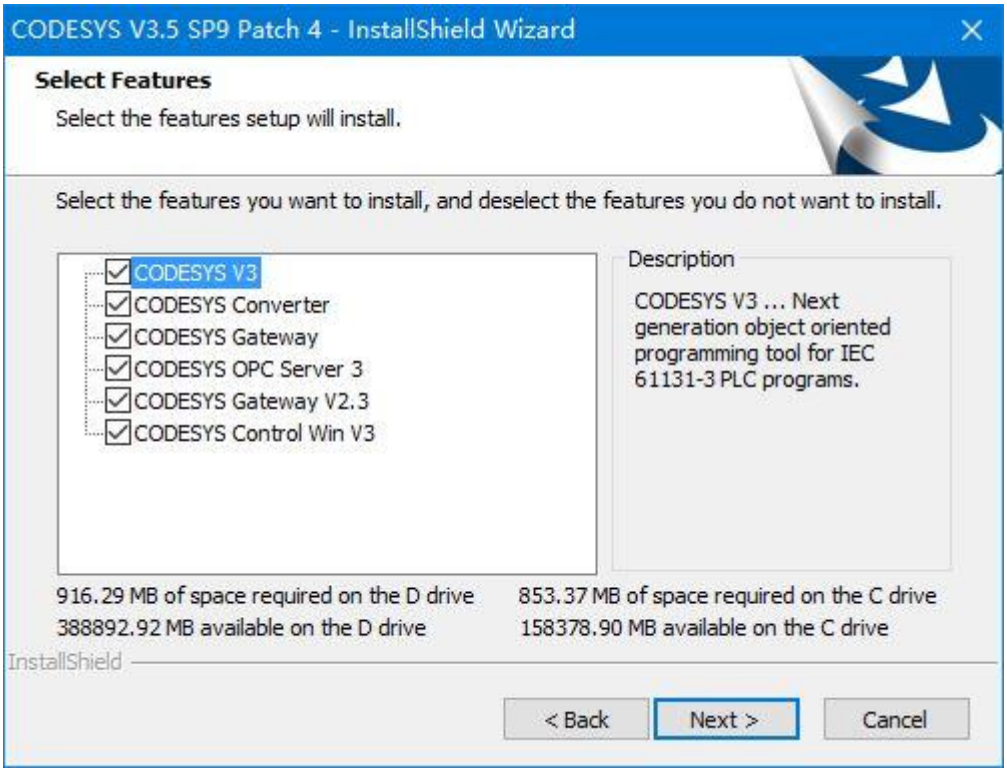


勾选“I accept the terms of the license agreement”，点击 Next。

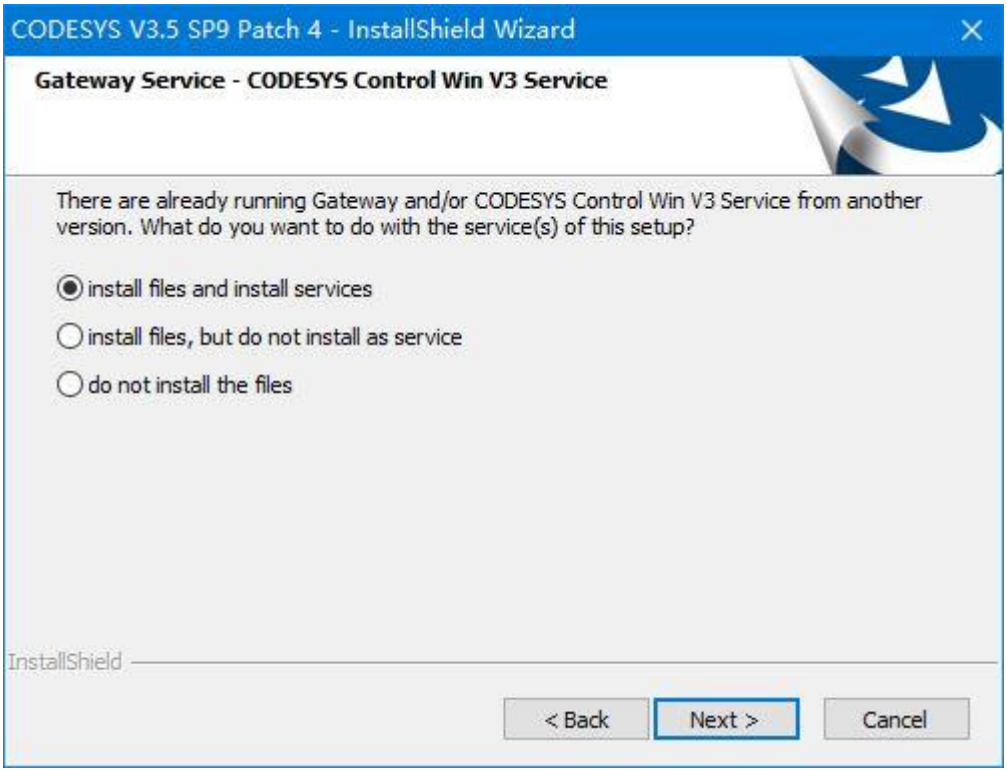
可修改默认安装路径，修改完成后点击 Next。



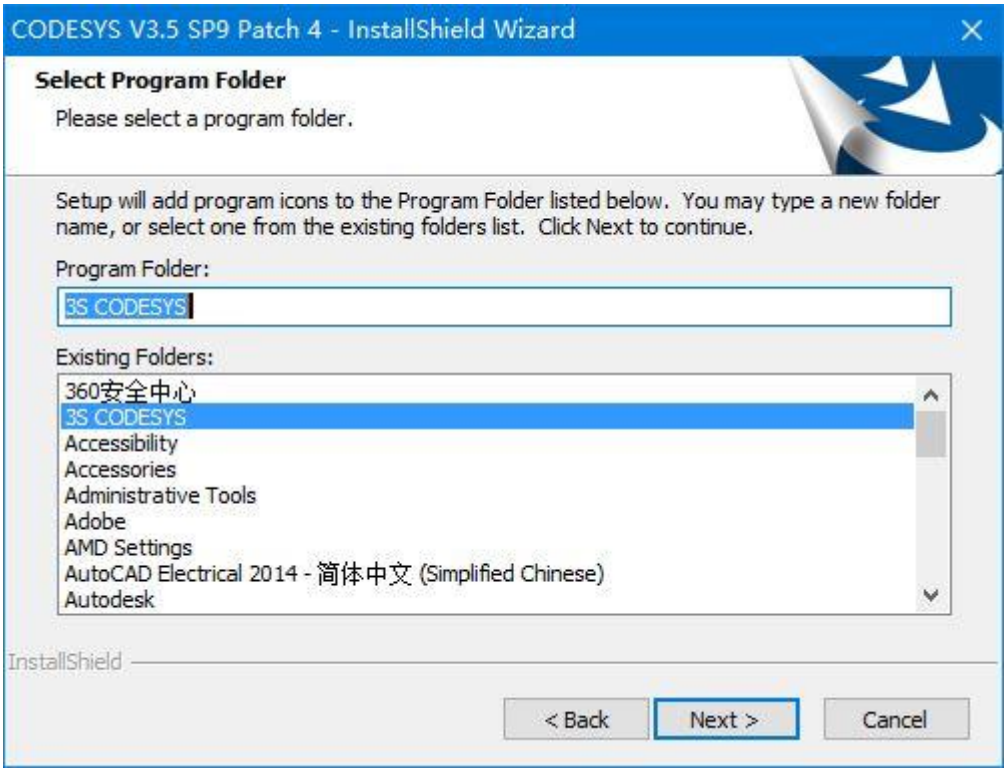
默认安装全部功能，点击 Next。



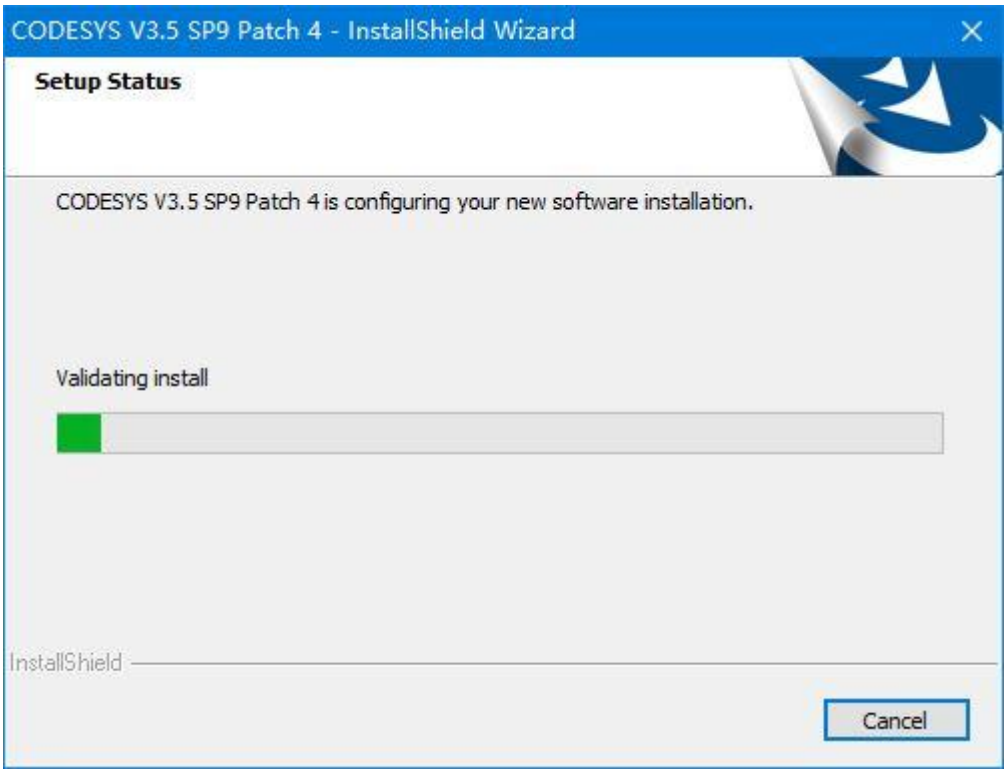
默认安装文件和服务，不修改，点击 Next。



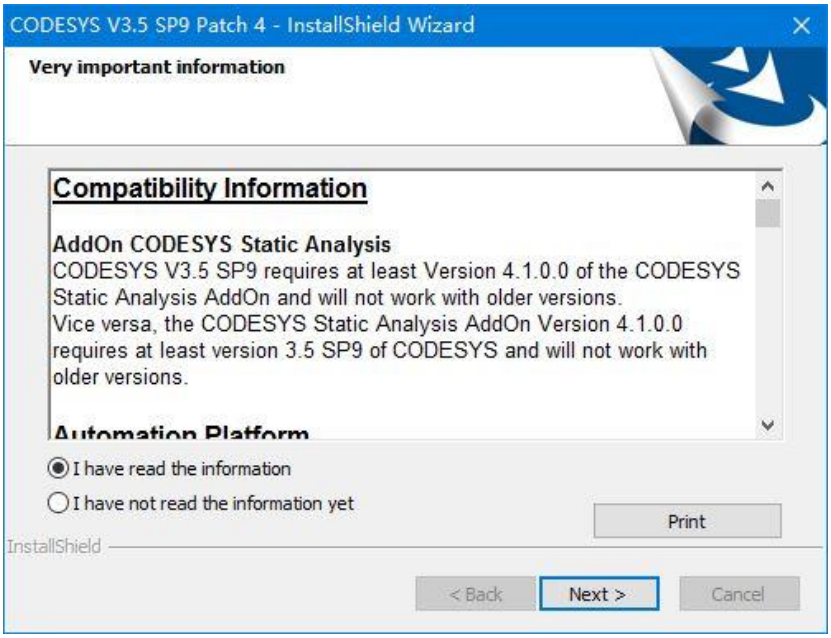
修改程序文件夹名，暂不修改，点击 Next。



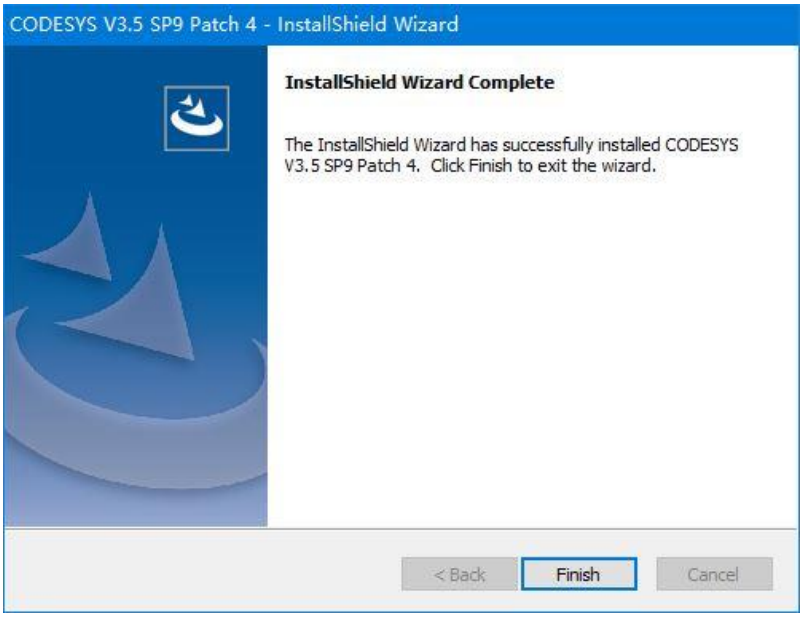
开始安装，过程可能持续十几分钟，请耐心等待。



勾选“I have read the information”，点击 Next。



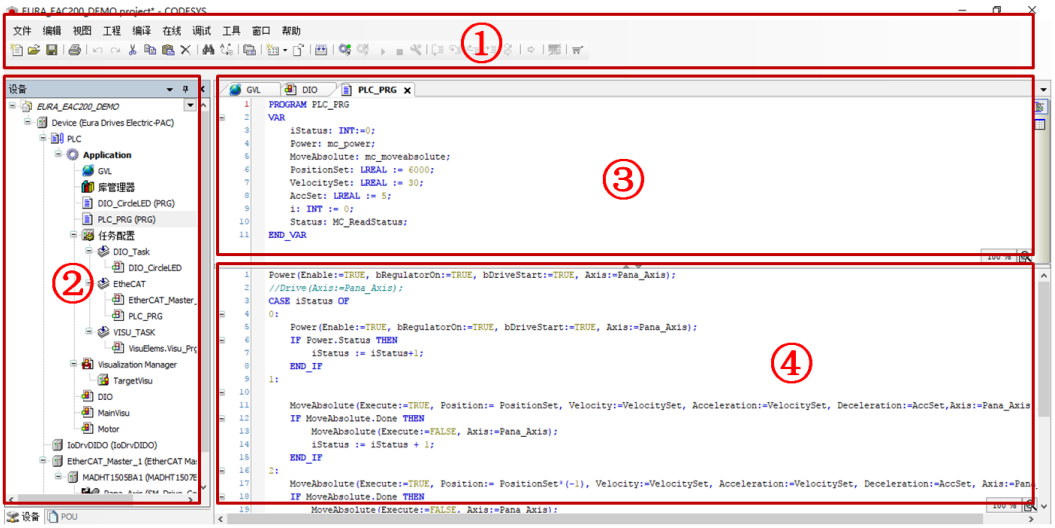
点击 Finish，完成安装。



此时在桌面上会出现 CODESYS 的图标，任务栏右侧会出现两个托盘图标，左侧的是 Gateway 程序，右侧的是 PLC 程序（不需要 PC 模拟时 PLC 程序可以关闭）。

3.3.3 基本界面

运行 CODESYS 后基本界面如下图所示，各部分功能如下表所述。

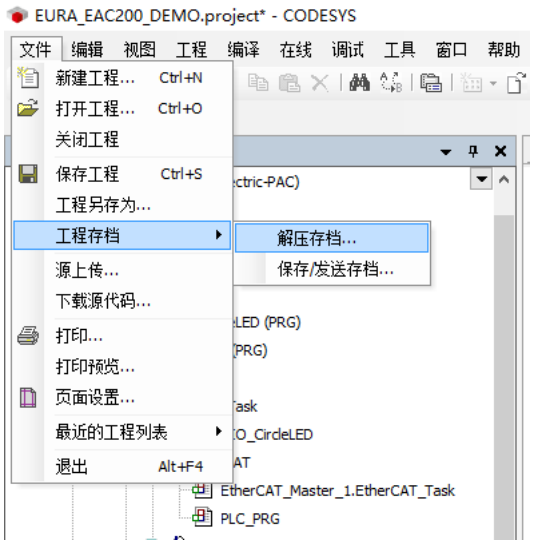


区域标号	说明
1	菜单栏和工具栏区，集成常用命令及操作
2	设备树区，显示整个工程的设备结构和程序组织结构
3	变量定义区，显示适用于该段程序中的变量定义
4	编程区，显示程序实现

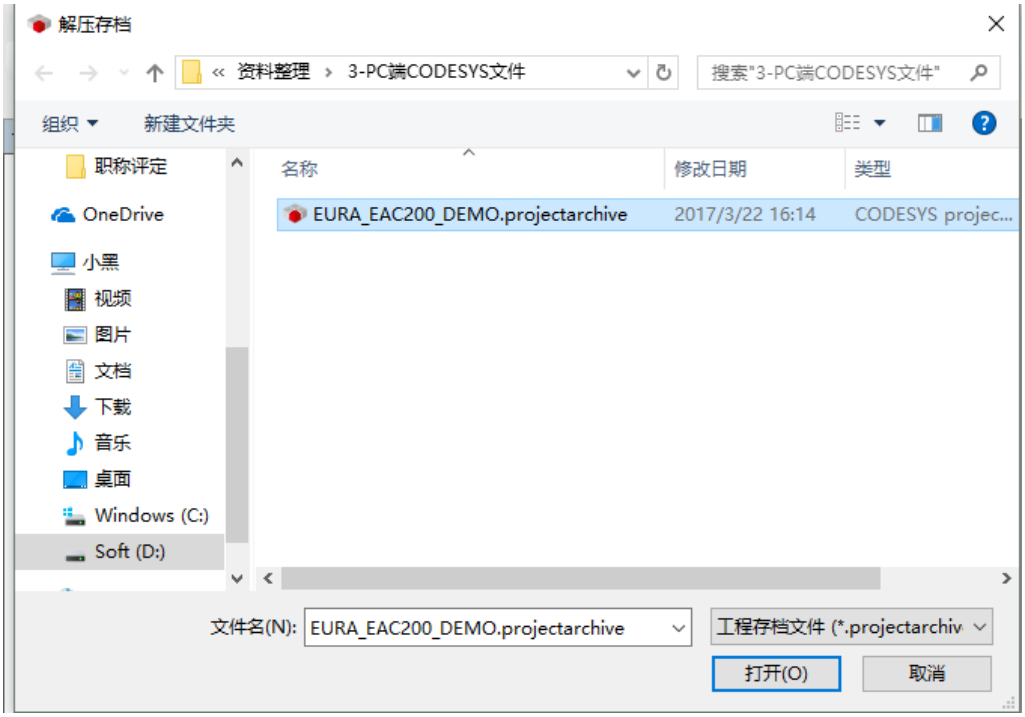
3.3.4 示例程序安装

实例程序可以在官网进行下载，里面包含 EAC 设备相关的设备描述文件和库文件，以及对 IO 和伺服控制的示例程序代码。

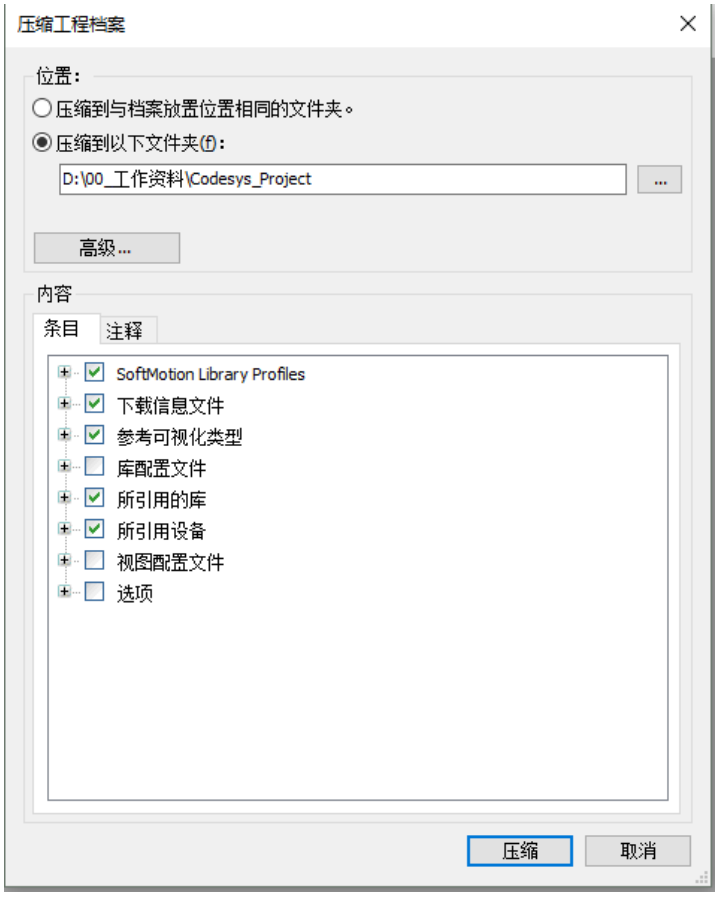
运行 CODESYS 进入主界面之后，选择文件-工程存档-解压存档



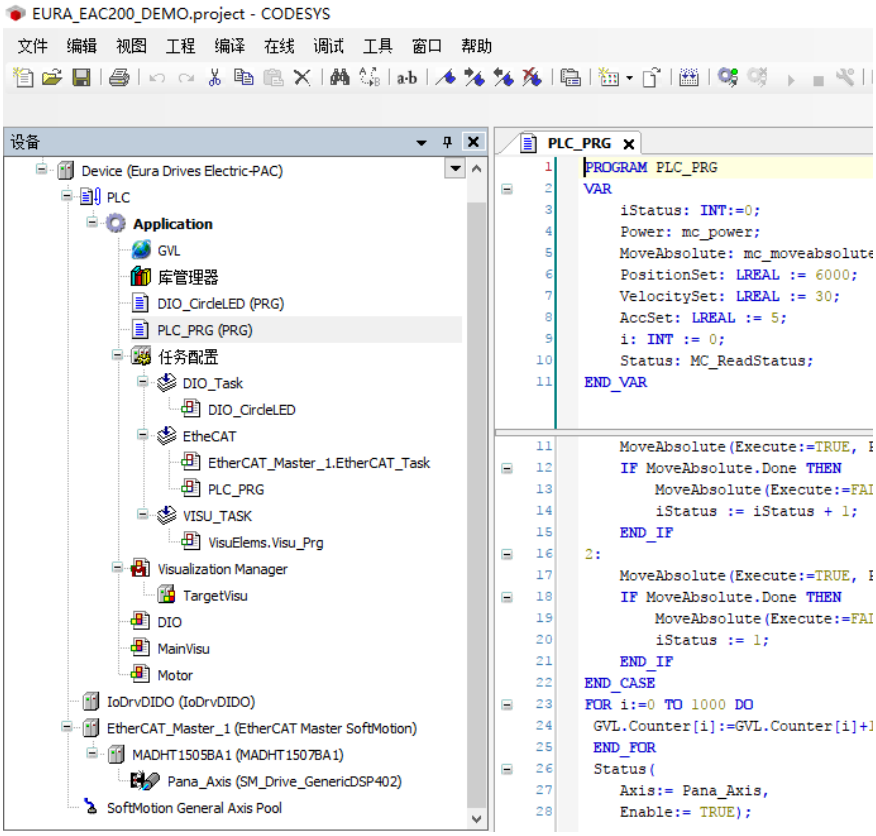
在弹出的对话框中选择示例工程文件路径及文件，点击打开。



在弹出的对话框中选择工程解压缩的路径，同时确认内容区域区域中的“所引用的库”和“所引用的设备”两个选项为勾选状态后，点击“压缩”按钮。



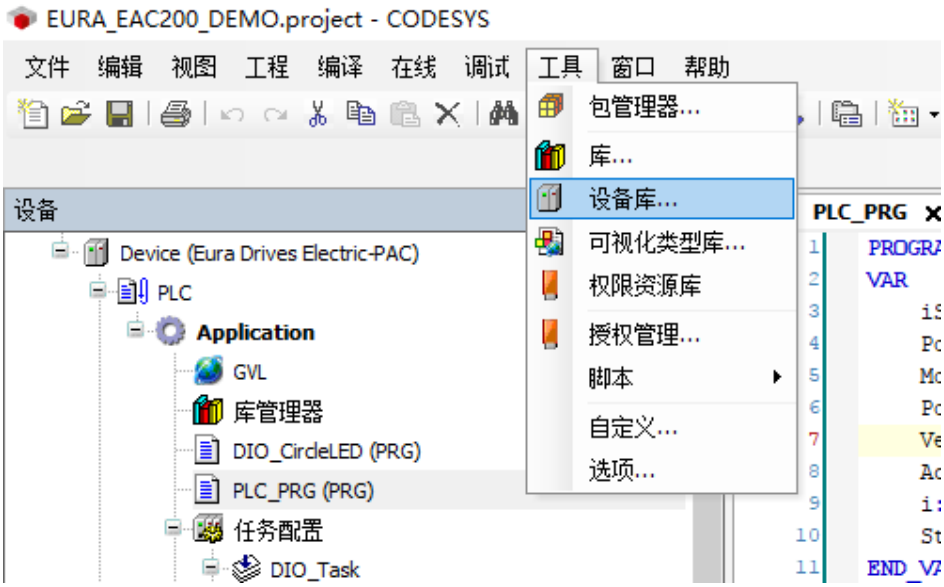
解压缩完成后，示例程序和 EAC200 平台所需的设备描述文件都已经安装到系统中，无需再次安装对应的设备描述文件。



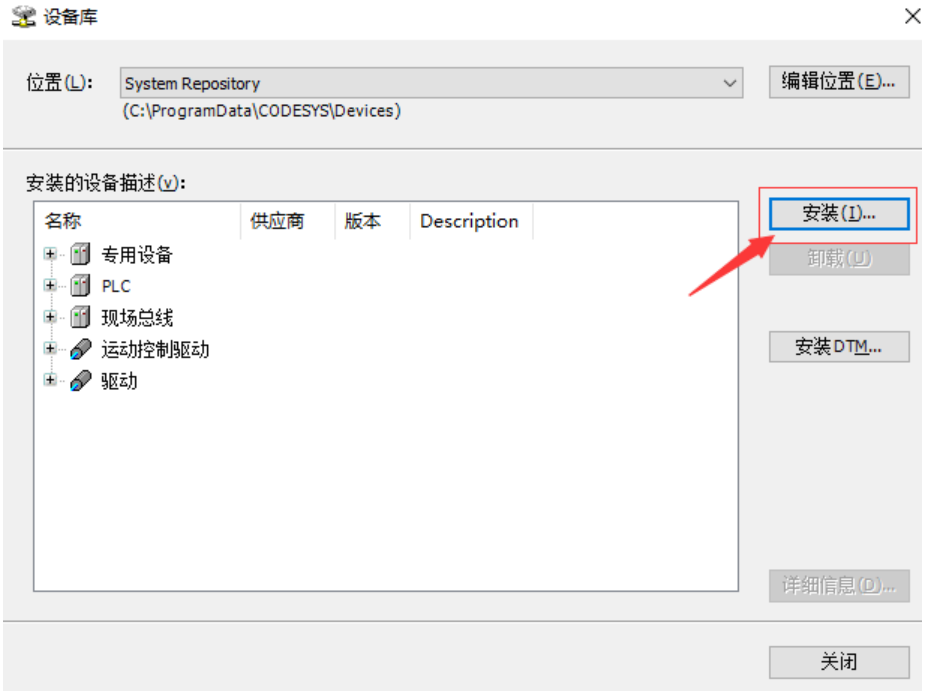
3.3.5 设备描述文件的安装与更新

当有新的设备需要安装或者已安装设备需要升级时，使用本节说明进行安装或更新。

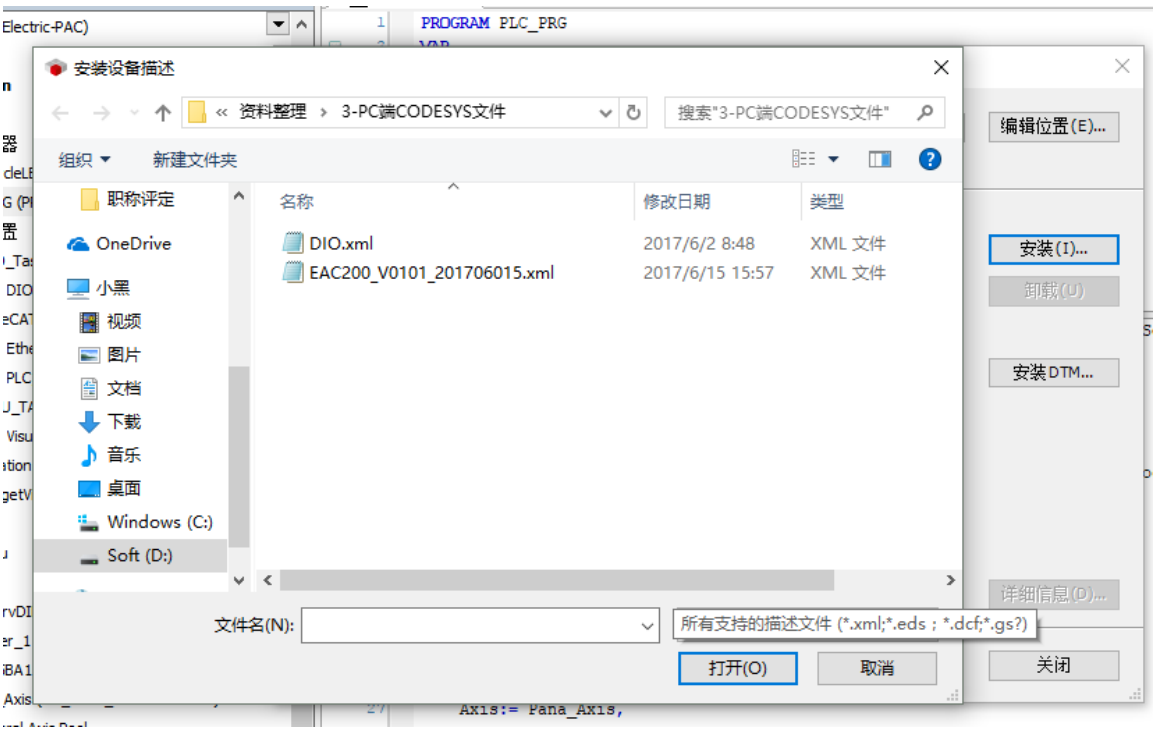
运行 CODESYS，点击菜单栏中“工具”——“设备库”选项。



在弹出的对话框中点击“安装...”按钮。



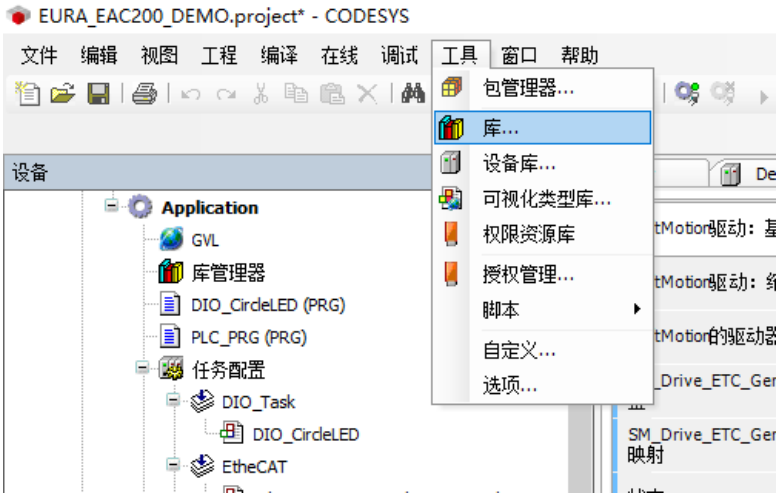
在弹出的对话框筛选文件类型一栏中选择“所有支持的描述文件”，选择需要安装或者更新的文件，点击打开，设备描述文件会自动安装或更新。



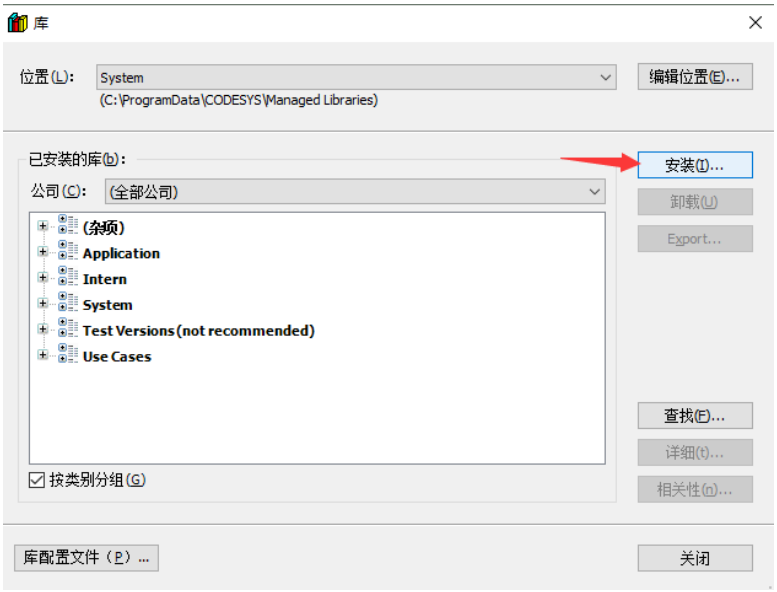
3.3.6 库文件的安装与更新

CODESYS 可以将函数和功能块封装成库文件，方便第三方用户使用，当需要添加库文件和升级已有的库文件时，按照本节内容进行操作。

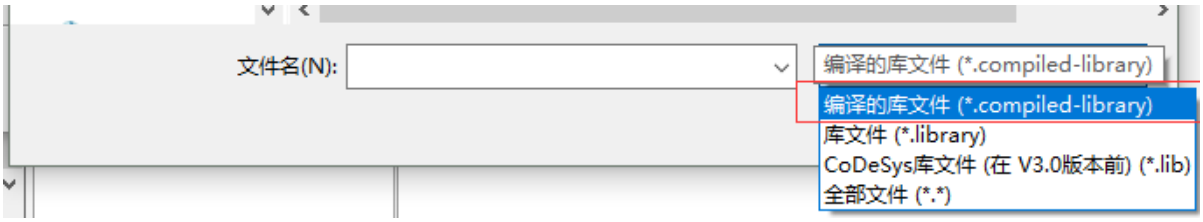
运行 CODESYS，点击菜单栏中“工具”——“库”选项。



在弹出的对话框中点击“安装...”按钮。



在弹出的对话框筛选文件类型一栏中选择“编译的库文件”或者“库文件”，选择需要安装或者更新的文件，点击打开，系统会自动安装或更新。

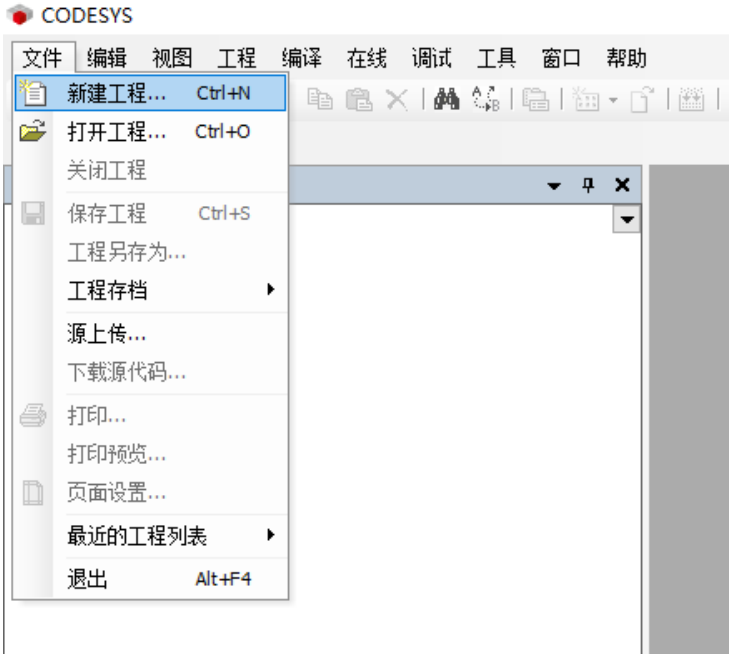


3.3.7 创建一个示例程序

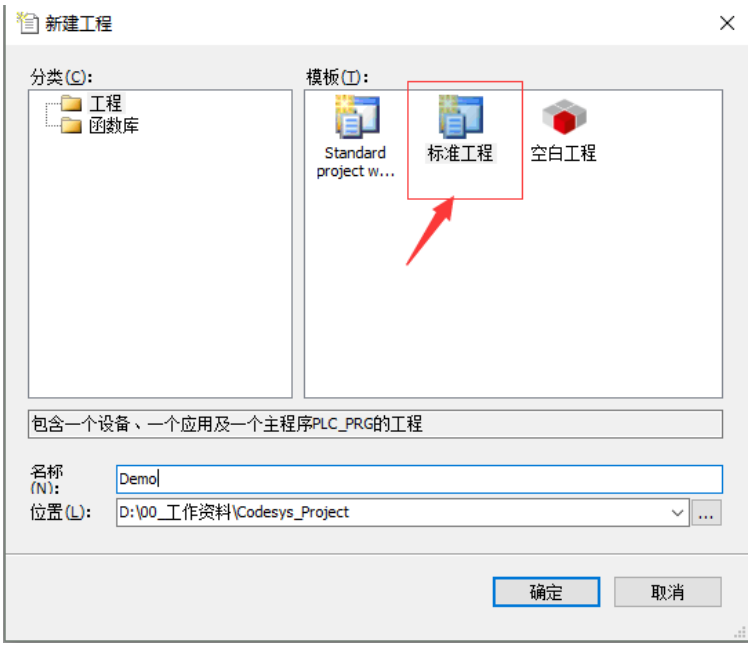
本节以示例程序为例，说明如何创建一个能在 EAC 设备上运行的工程以及简单的下载和在线监控等操作。示例程序包括基于板载 IO 的流水灯程序、基于 EtherCAT 和欧瑞伺服的伺服正反转动程序以及对应的视图界面。完整的实例程序可以在官网进行下载。关于编程环境基本的界面操作和编程语法可以参考软件自带的帮助文档。

3.3.7.1 选择设备及主程序编程语言

启动 CODESYS，在菜单栏中选择“文件”——“新建工程”。



在弹出的对话框中选择“标准工程”，输入工程名称和存储路径，点击确定。

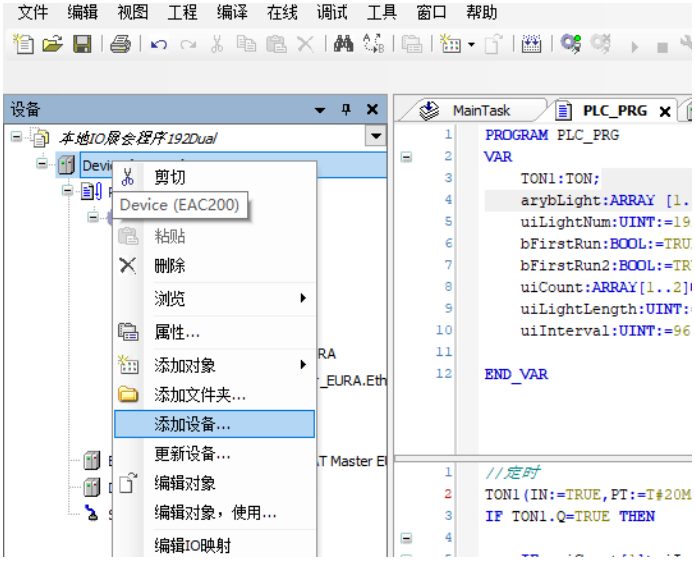


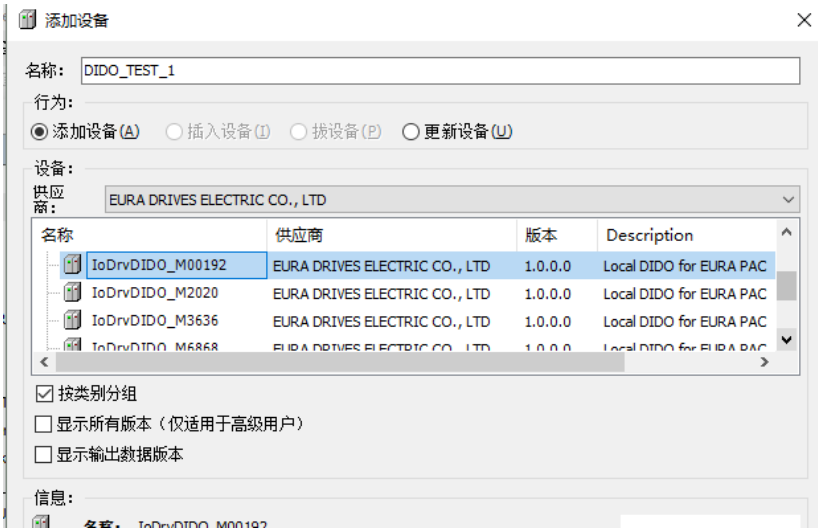
在弹出的对话框中设备一栏选择“Eura Drives Electric-PAC”，主程序编程语言 PLC_PRG 选项选择“结构化文本（ST）”，点击确定按钮。

3.3.7.2 添加 IO 设备

当项目中使用到数字 IO 时，用户可以通过添加对应的 DIO 模块设备进行后续的变量映射。

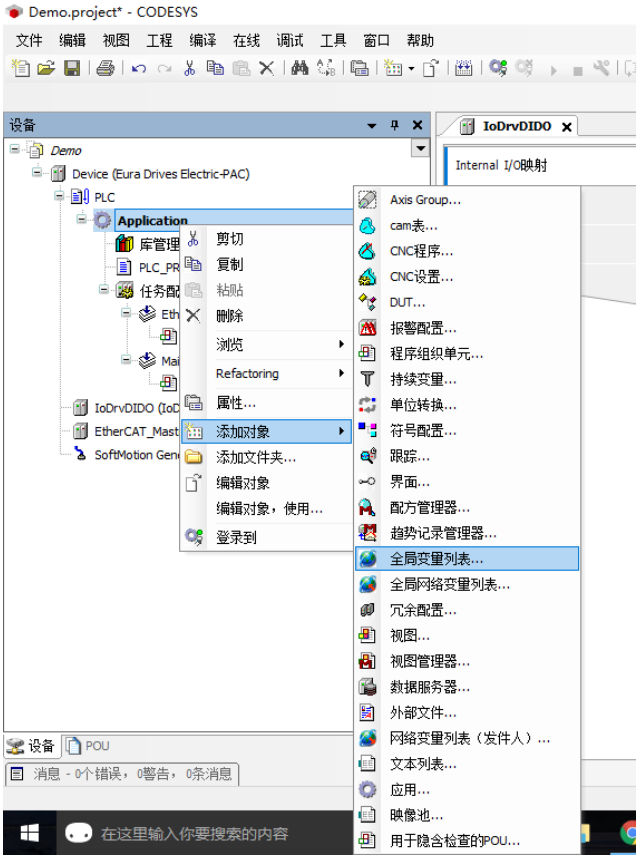
右击设备树“Device”节点，在右键菜单中选择“添加设备”，在弹出的对话框中的供应商选项中选择“EURA DRIVES ELECTRIC CO., LTD”后，根据实际应用的扩展模块选择相应的设备，点击添加设备。



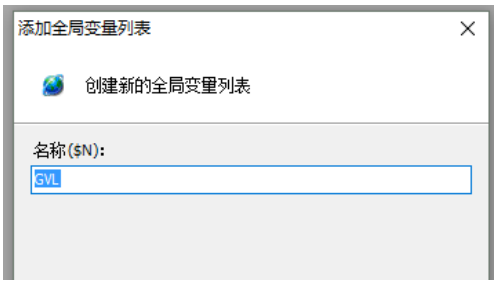


3.3.7.3 数字 IO 部分

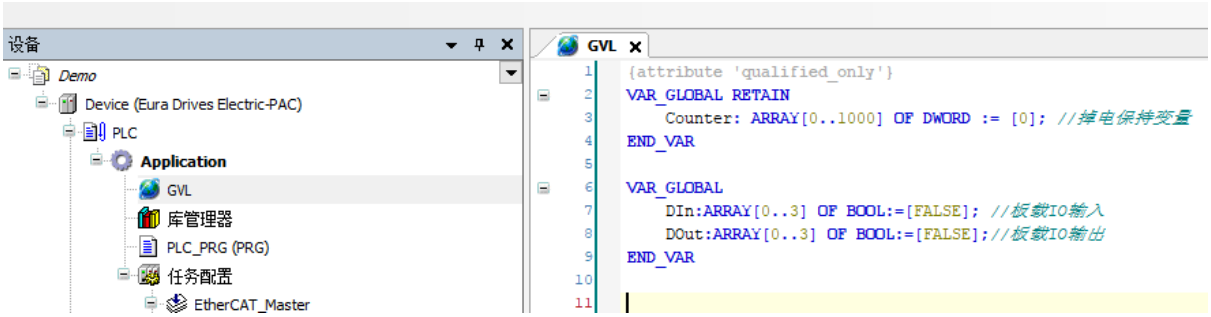
创建全局 IO 变量：在右侧设备树选项卡中的“Application”节点右击，在弹出的菜单中选择“全局变量列表”选项。



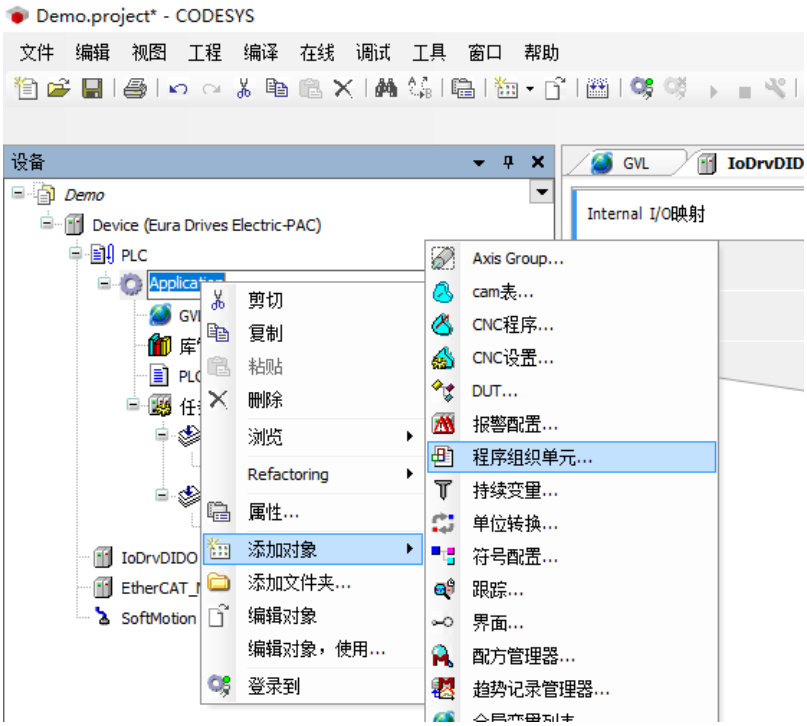
填写全局变量列表名称，点击打开。



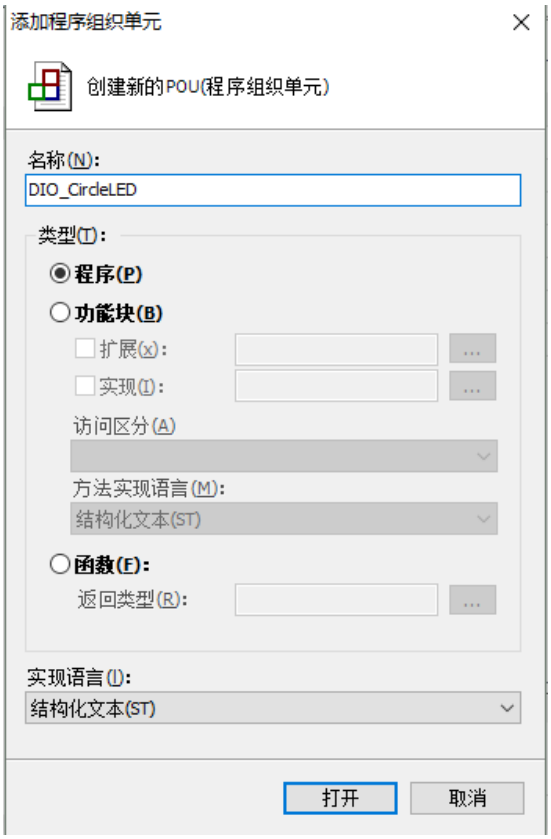
在右侧的设备树中找到建立的全局变量列表，双击打开编辑器，在变量定义区添加保持变量（掉电变量数据不丢失）定义和其他全局变量定义。



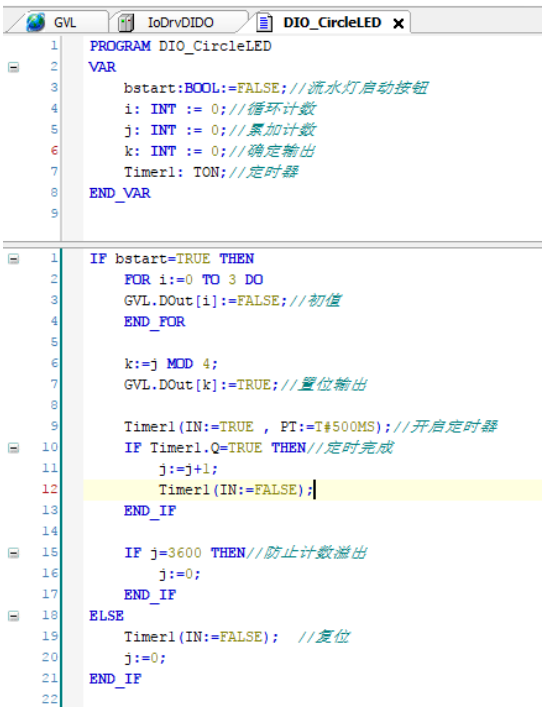
创建 IO 程序：在右侧设备树选项卡 Application 节点右击，选择“添加对象”—“程序组织单元”。



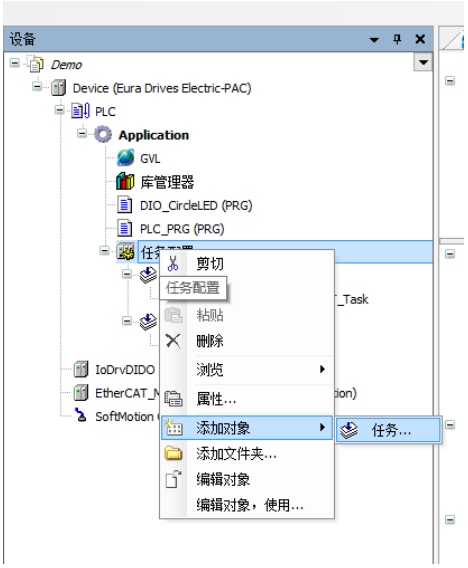
填写程序名称，实现语言选择结构化文本，点击打开。



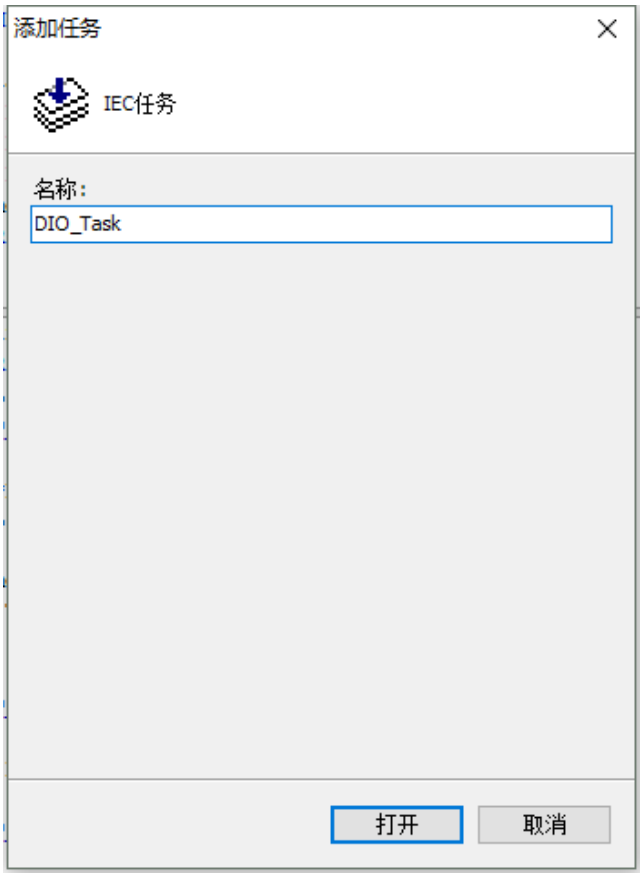
在打开的程序编辑器中完成 IO 部分流水灯程序的变量定义和程序实现。



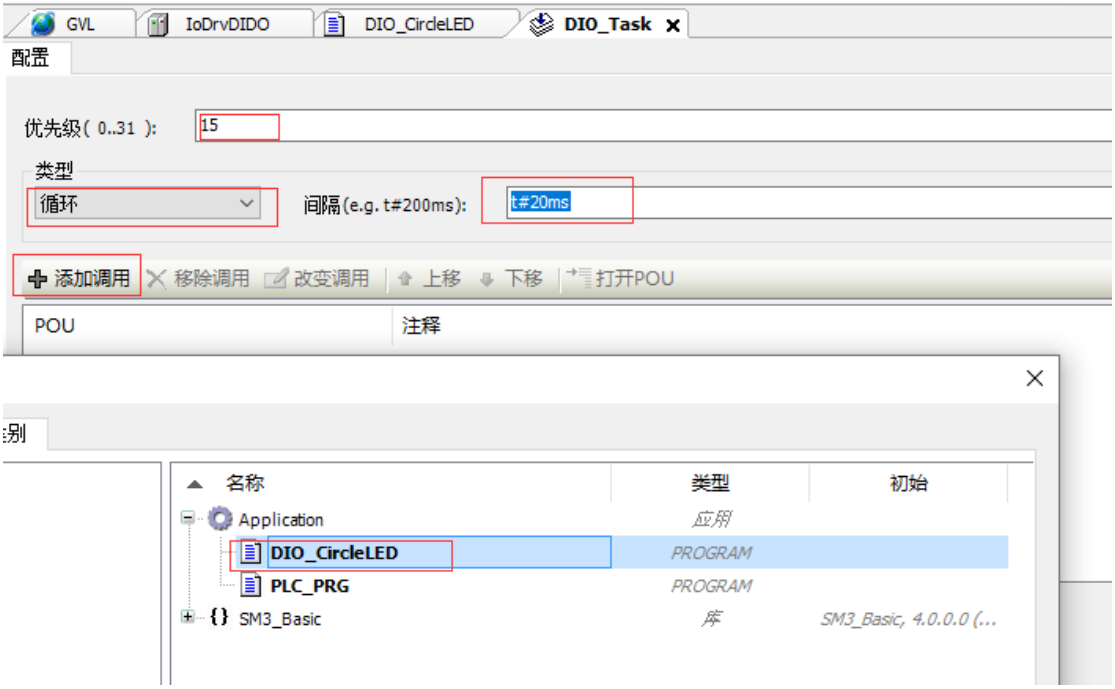
创建 IO 任务：在右侧设备树选项卡“任务配置”节点右击，选择“添加对象”—“任务”。



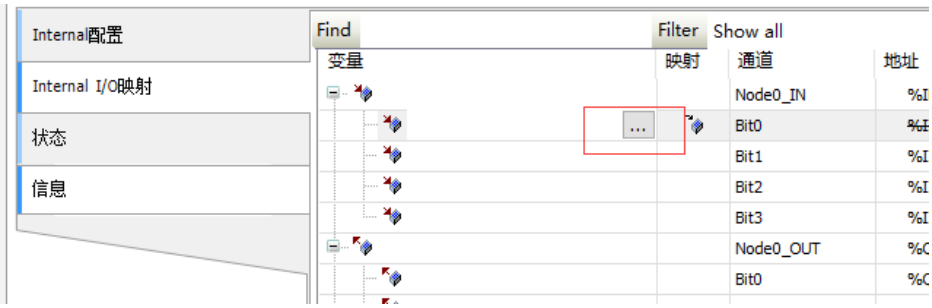
填写任务名称，点击打开



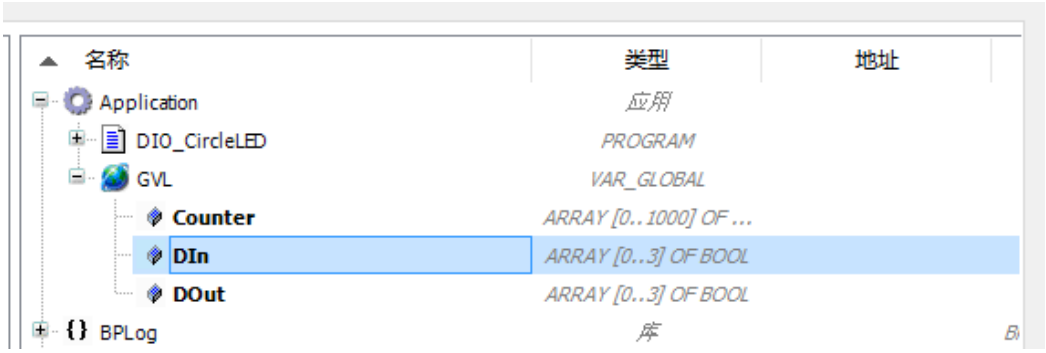
在打开的编辑器里选择对应的任务循环类型，填写优先级和循环间隔，点击添加调用，选择之前编写的 DIO_CircleLED 程序。



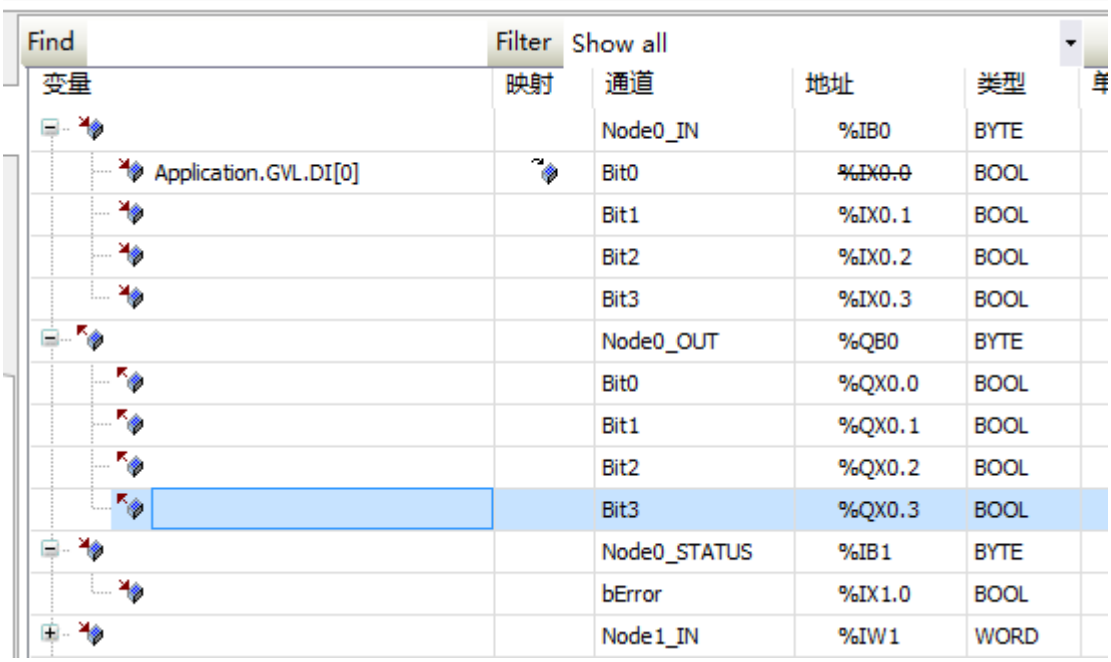
创建 IO 设备映射：双击设备树选项卡中的板载 IO 设备“IoDrvDIDO”，在右侧编辑器“I/O 映射”选项卡表格的变量一栏双击，然后点击右侧出现的“...”按钮。



在弹出的对话框中选择 DIn，点击确定。

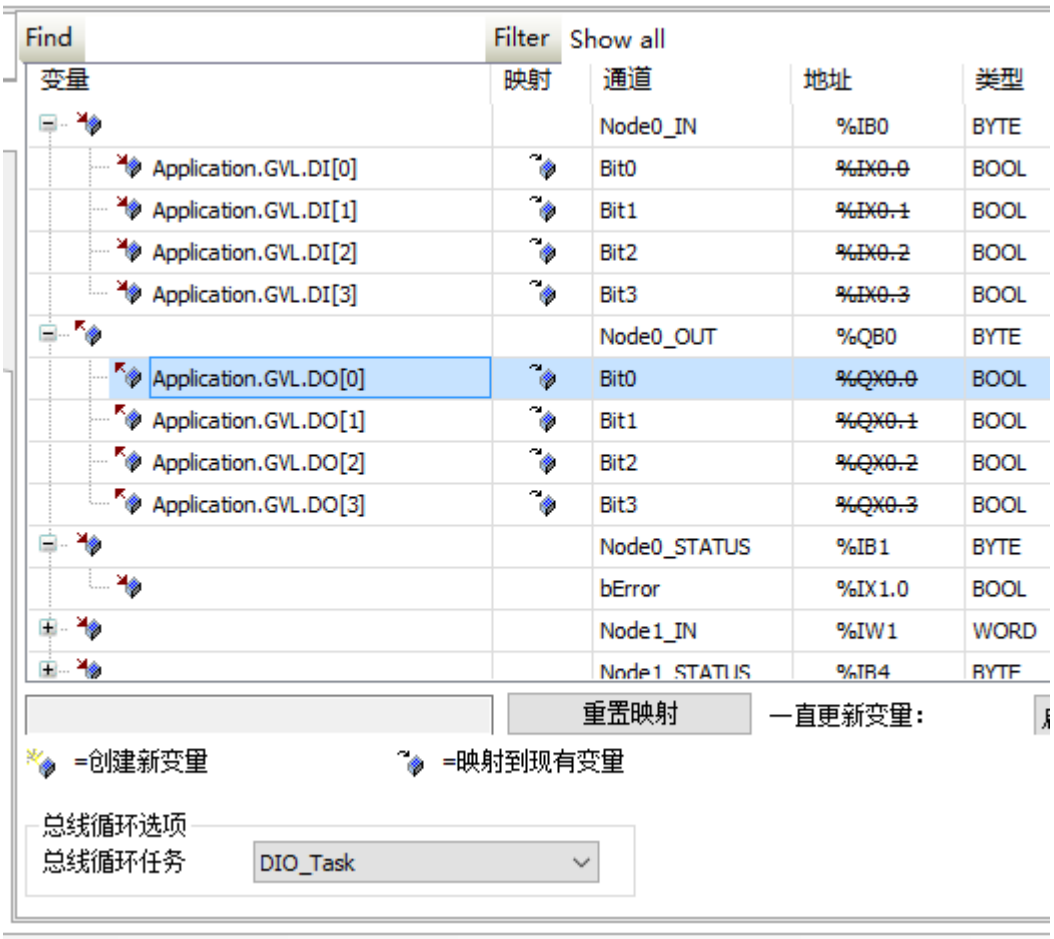


由于 Din 为数组类型，需要手动添加元素序号“[0]”。



变量	映射	通道	地址	类型
		Node0_IN	%IB0	BYTE
Application.GVL.DI[0]		Bit0	~~%IX0.0~~	BOOL
		Bit1	%IX0.1	BOOL
		Bit2	%IX0.2	BOOL
		Bit3	%IX0.3	BOOL
		Node0_OUT	%QB0	BYTE
		Bit0	%QX0.0	BOOL
		Bit1	%QX0.1	BOOL
		Bit2	%QX0.2	BOOL
		Bit3	%QX0.3	BOOL
		Node0_STATUS	%IB1	BYTE
		bError	%IX1.0	BOOL
		Node1_IN	%IW1	WORD

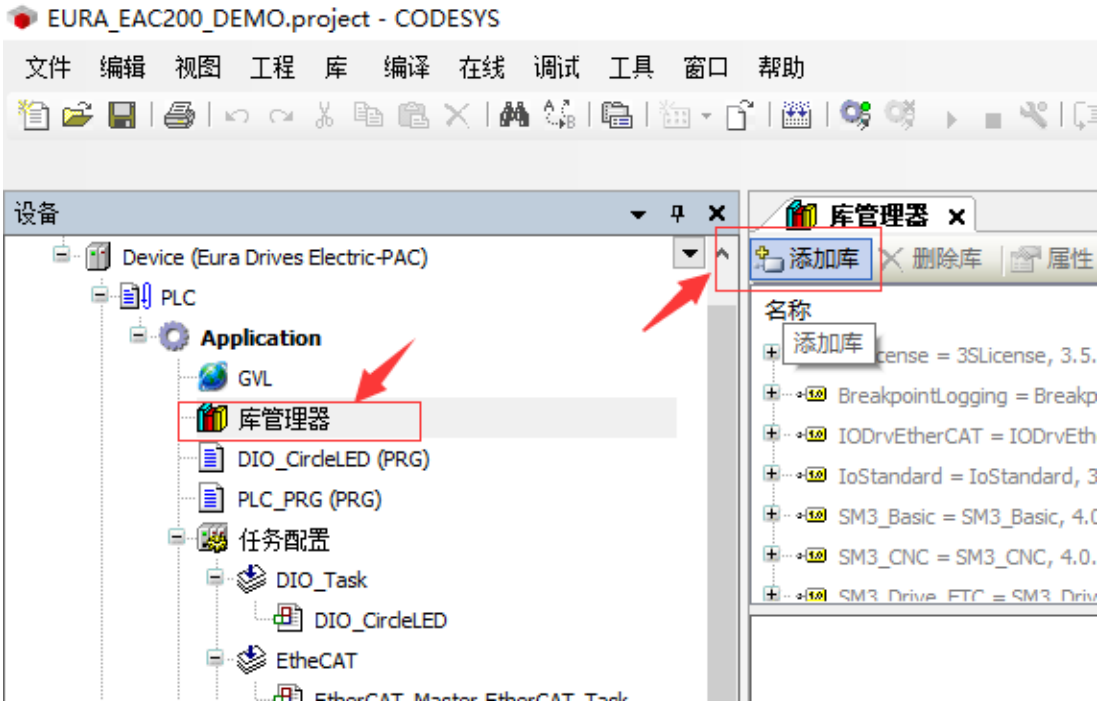
重复上述操作，将板载 IO 设备上的输入和输出映射到之前定义好的全局变量数组中，同时将“总线循环任务”中的选项更改为 DIO_Task 任务。



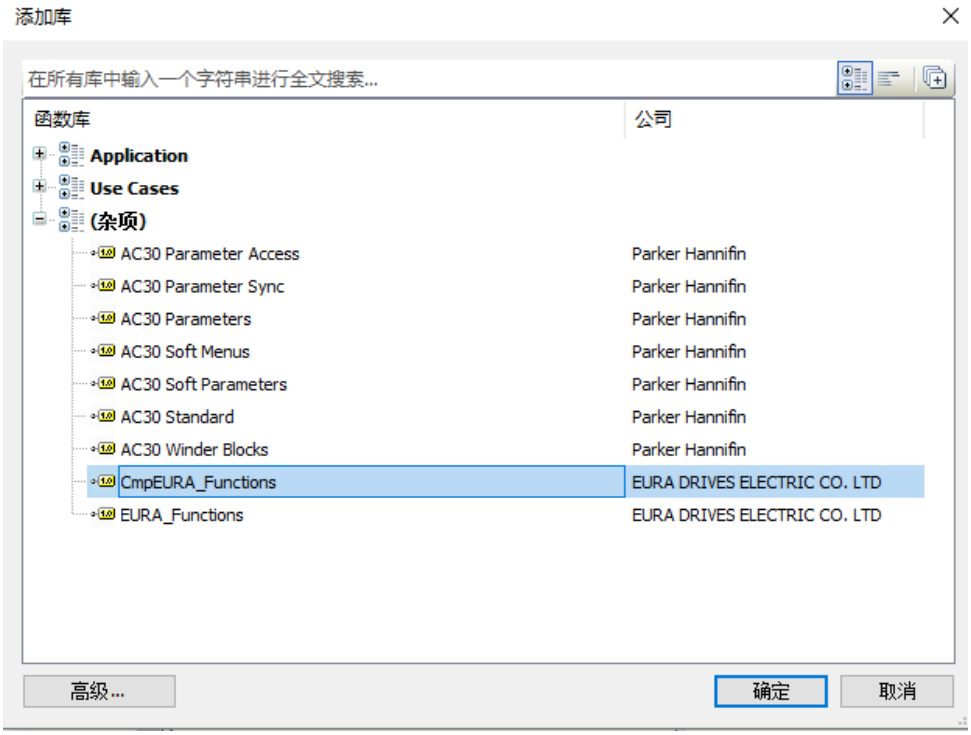
变量	映射	通道	地址	类型
		Node0_IN	%IB0	BYTE
Application.GVL.DI[0]		Bit0	~~%IX0.0~~	BOOL
Application.GVL.DI[1]		Bit1	~~%IX0.1~~	BOOL
Application.GVL.DI[2]		Bit2	~~%IX0.2~~	BOOL
Application.GVL.DI[3]		Bit3	~~%IX0.3~~	BOOL
		Node0_OUT	%QB0	BYTE
Application.GVL.DO[0]		Bit0	~~%QX0.0~~	BOOL
Application.GVL.DO[1]		Bit1	~~%QX0.1~~	BOOL
Application.GVL.DO[2]		Bit2	~~%QX0.2~~	BOOL
Application.GVL.DO[3]		Bit3	~~%QX0.3~~	BOOL
		Node0_STATUS	%IB1	BYTE
		bError	%IX1.0	BOOL
		Node1_IN	%IW1	WORD
		Node1_STATUS	%IB4	BYTE

3.3.7.4 串口部分

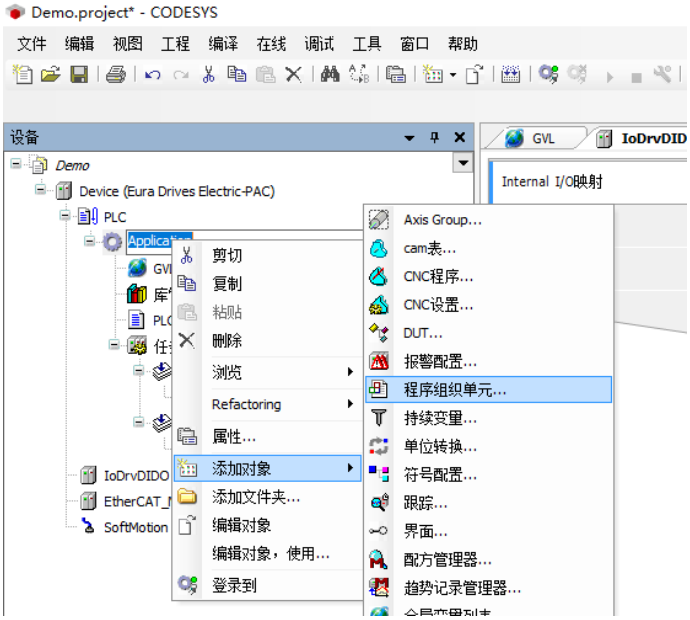
添加串口功能块库：双击在右侧设备树选项卡中的“库管理器”节点，在左侧的编辑区中单击“添加库”按钮。



在弹出的对话框中杂项标签下选择“CmpEuraFuncions”库，单击确定。



创建串口程序：在右侧设备树选项卡 Application 节点右击，选择“添加对象”—“程序组织单元”。



填写程序名称，实现语言选择结构化文本，点击打开。

在打开的程序编辑器中完成 RS232 串口通讯程序的变量定义和程序实现。

```
VAR

    iState: DINT := 0;
    bOpen: BOOL := FALSE;
    ComSet:EAC200.COM_Setting;
    bSet: BOOL := FALSE;
    bWrite: BOOL := FALSE;
    str : STRING := 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890$R$N';
    abyWriteData: array[0..4095] of BYTE;
```

```
CASE iState OF
    0:
    bOpen:=EAC200.OpenCom_RS232(Result=> Result);
    IF bOpen=FALSE THEN
        iState:=1000;
    ELSE
        FOR i:=0 TO 4095 DO
        abyWriteData[i] := str[i MOD 64];
        END_FOR
        iState:=1;
    END_IF

    1:
    //设置通讯参数
    ComSet.Baudrate:=EAC200.COM_Setting_Baudrate.CBR_115200;
    ComSet.Databits:=8;
    ComSet.Parity:=EAC200.COM_Setting_PARITY.NOPARITY;
    ComSet.StopBit:=EAC200.COM_Setting_STOPBIT.ONESTOPBIT;
    bSet:=EAC200.SetCom_RS232(ComSetting:=ComSet , Result=>Result);
    IF bSet=FALSE THEN
        iState:=1000;
    ELSE
```

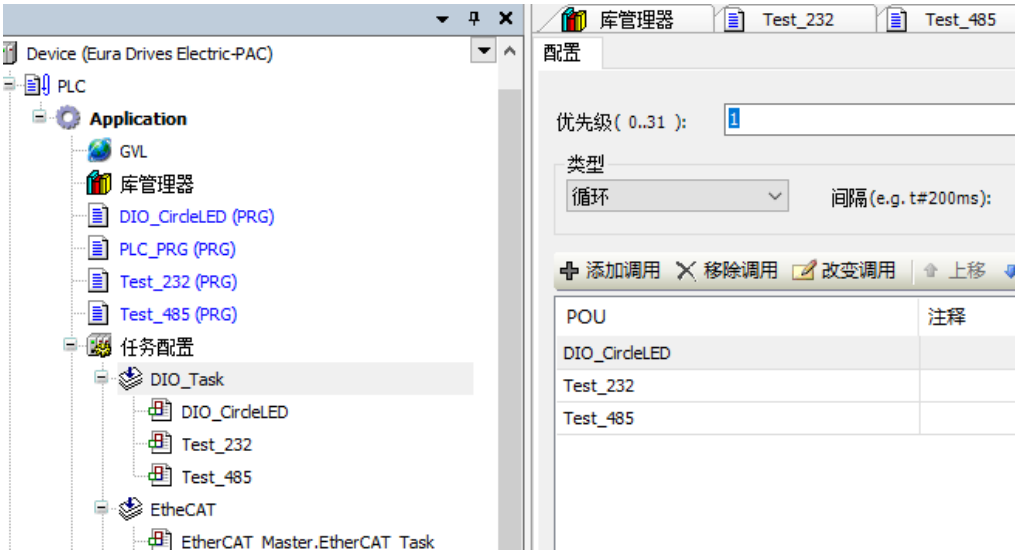
同样创建 RS485 的程序 POU 和实现

```
    iState: DINT := 0;
    bOpen: BOOL := FALSE;
    ComSet:EAC200.COM_Setting;
    bSet: BOOL := FALSE;
    bSend: BOOL := FALSE;
    bWrite: BOOL := FALSE;
    str : STRING := 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890$R$N';
    abySendData: ARRAY[0..4095] OF BYTE;
    abyReadData: ARRAY[0..6] OF BYTE;
```

```
    2:
    bSend:=EAC200.SendCom_RS485(pbyWriteBuffer:=ADR(abySendData) , ulNumOfBytesToSend:=64 , ulActualBytesSend=> dwWritten, Resu
    IF (bSend=FALSE OR dwWritten=0) THEN
        iState:=1000;
    ELSE
        iState:=50;
    END_IF

    3:
    bRead:=EAC200.ReceiveCom_RS485(pbyReadBuffer:=ADR(strRead) , ulNumOfBytesToRecv:=7 , ulActualBytesRecv=>dwRead , Result=>Re
    IF bRead=FALSE THEN
        iState:=1000;
    END_IF
    IF  dwRead<>0 THEN
        bWrite:=EAC200.SendCom_RS485(pbyWriteBuffer:=ADR(strRead) , ulNumOfBytesToSend:=dwRead , ulActualBytesSend=> dwWritten,
        IF (bWrite=FALSE OR dwWritten=0) THEN
            iState:=1000;
        ELSE
            iState:=50;
        END_IF
    END_IF
END_CASE
```

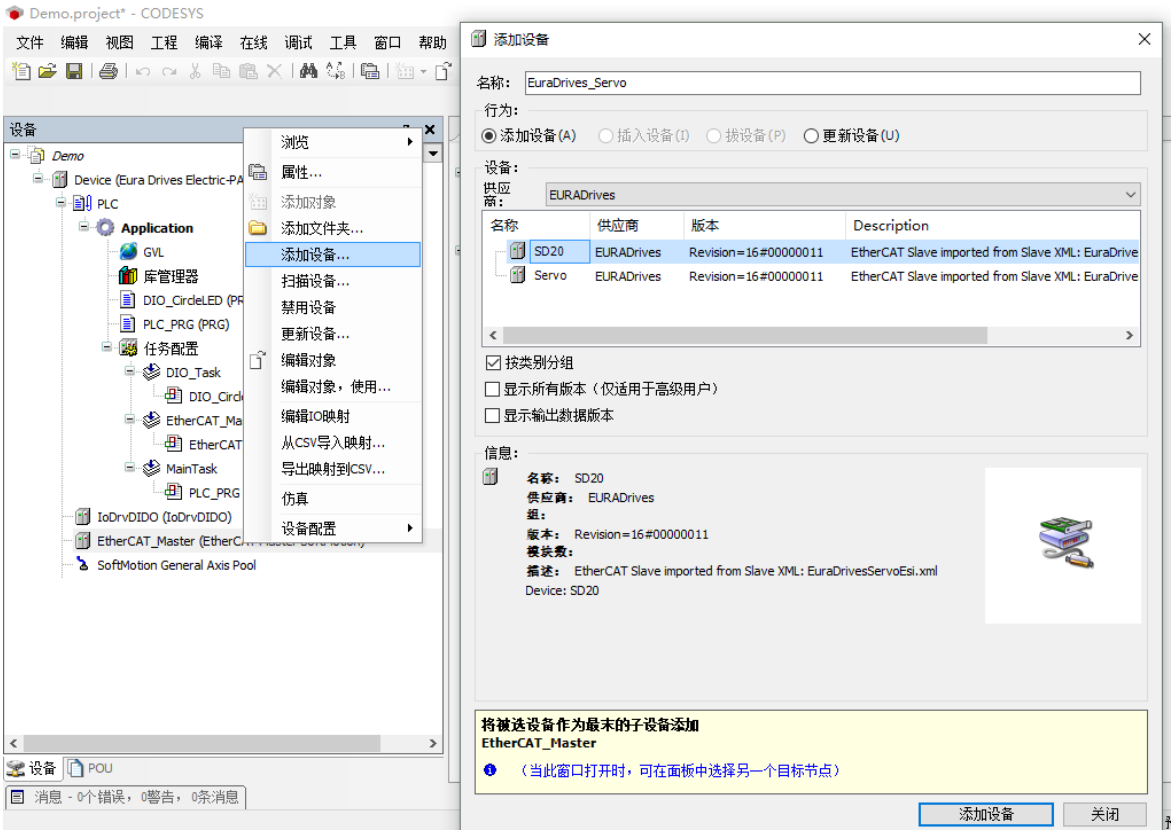
设置串口程序任务：双击右侧设备树“任务配置”节点下的“DIO_Task”子节点，在左侧编辑区点击“添加调用”按钮，将 Test_232 和 Test_485 两个程序添加到该任务下。



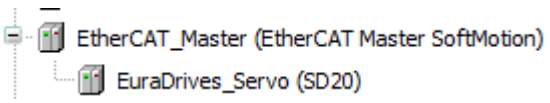
3.3.7.5 伺服部分

添加 EtherCAT 伺服设备：EtherCAT 总线伺服设备的添加有手动添加和在线扫描两种方式，本节依次介绍这两种方式。

手动添加：右击设备树 EtherCAT_Master 节点，在右键菜单中选择“添加设备”，在弹出的对话框中的供应商选项中选择“EURADrives”后，选中 SD20 设备，点击添加设备。



完成添加的设备会出现在设备树 EtherCAT_Master 节点下的子节点中。

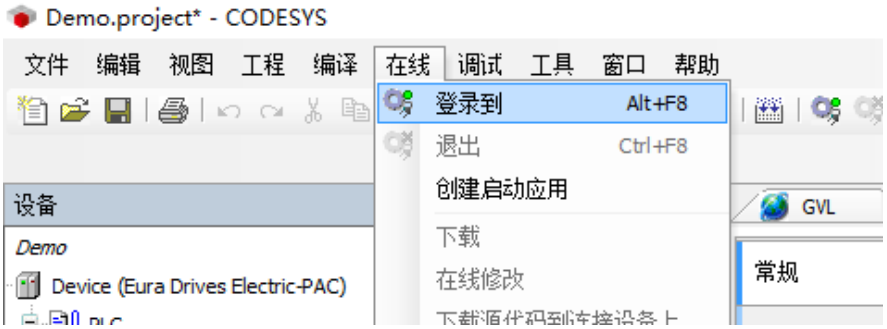


在线添加：在线添加适用于总线伺服设备和运动控制器已经全部物理连接完成的情况下，可以直接在线扫描所有连接的总线设备，一次性完成添加。

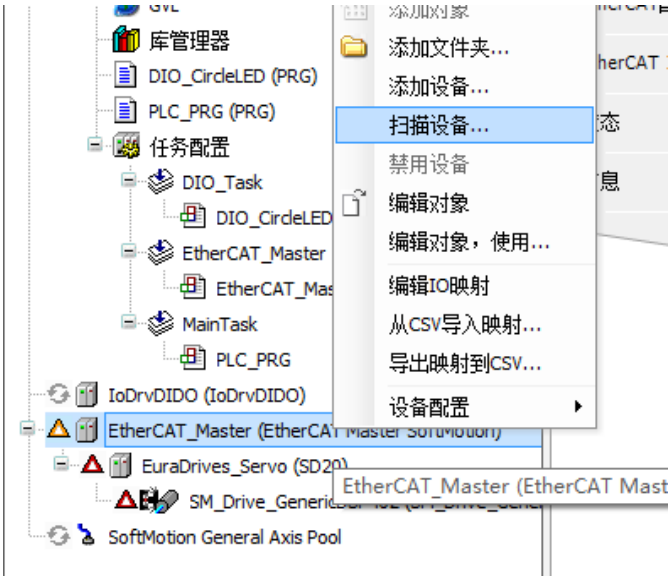
使用网线连接 PC 网口和 EAC 设备的下载调试网口，将 PC 的 IP 地址设置为和运动控制器同一网段（查看或者复位运动控制器 IP 地址请参考 2.4.1 节内容）。

双击设备树中的“Device”节点，在右侧的编辑框中选择 Gateway-1 节点，点击扫描网络；选中扫描出的设备，点击设置活动路径完成 EAC 设备的识别。

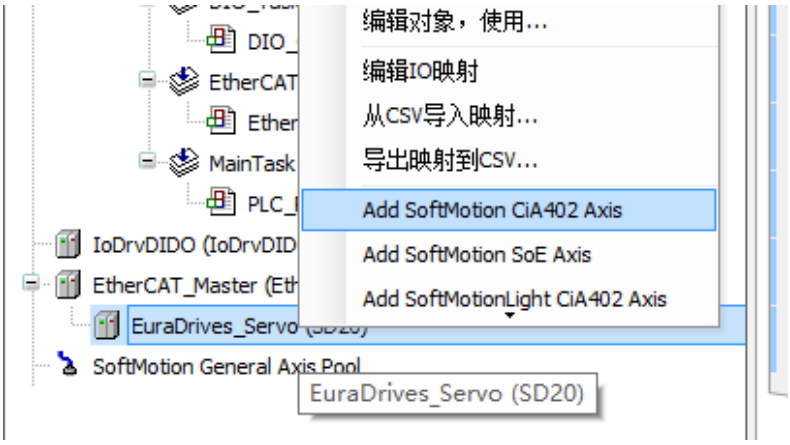
点击菜单栏中的“在线”——“登录到”，在弹出的对话框选择确认继续下载。



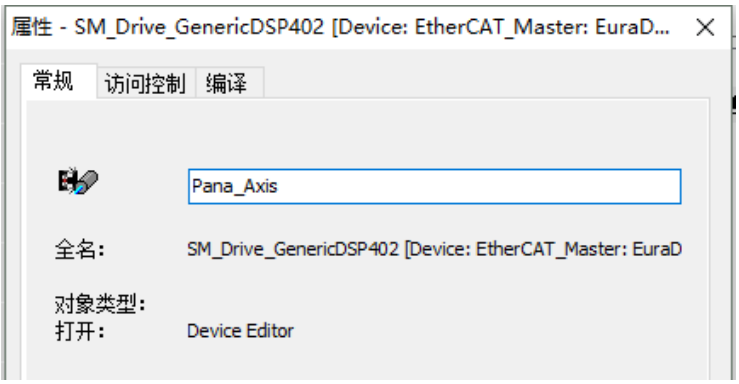
右击设备树 EtherCAT_Master 节点，在右键菜单中选择“扫描设备”，系统会自动扫描已连接的所有设备，点击“复制所有设备到工程中”完成设备的添加。



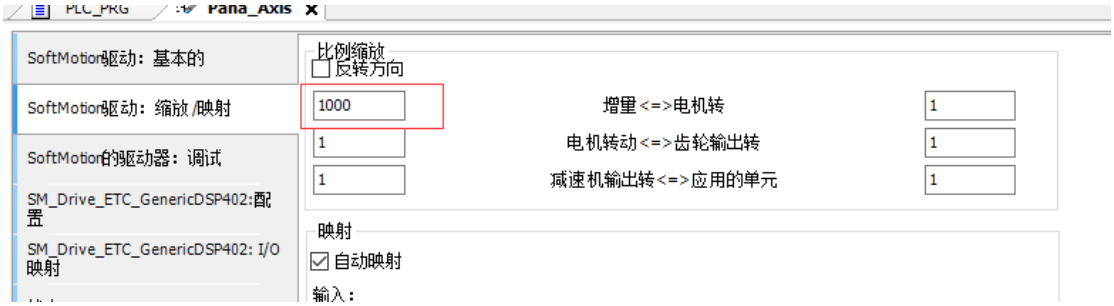
添加虚轴：右击设备树上的“SD20”设备节点，在弹出的菜单中选择“Add Softmotion CiA402 Axis”。（若出现信息提示请点击 OK 按钮）。



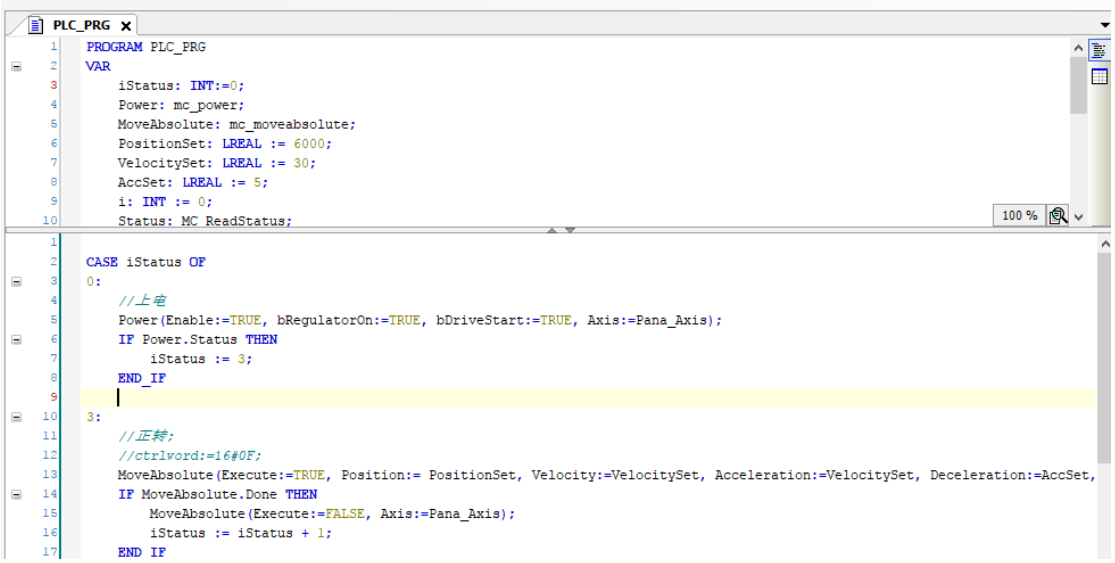
右击 402 轴，在菜单中选择属性一项，在弹出的对话框中将 402 轴名称更改为“Pana_Axis”。点击确定。



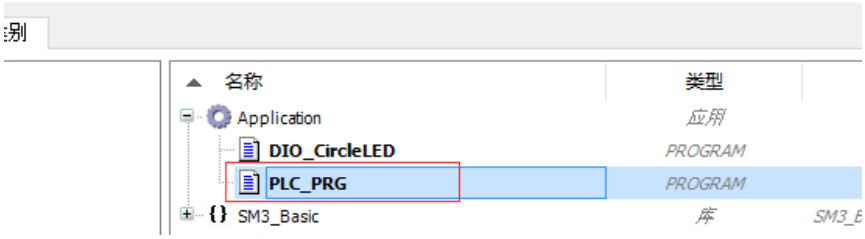
双击 402 轴节点“Pana_Axis”,在“缩放/映射”选项卡中修改增量值为 1000；



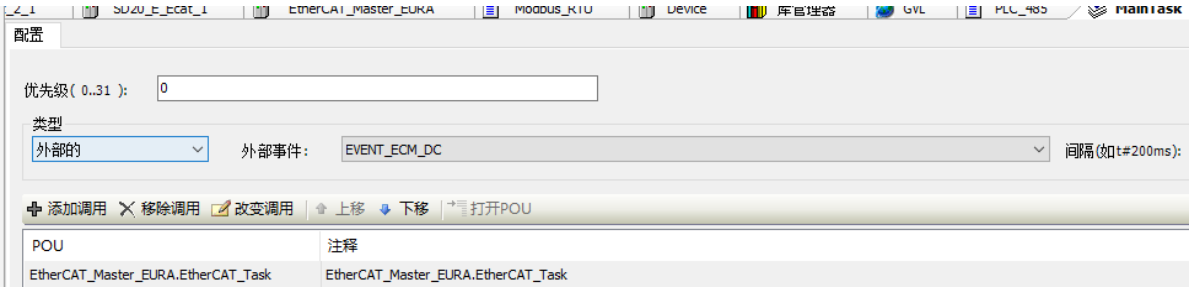
添加轴控主程序：双击设备树中的 PLC_RPG 程序，在打开的程序编辑器中完成轴控部分程序的变量定义和程序实现。



添加轴控任务：右击设备树任务节点下“Main_Task”子节点,在菜单中选择删除。双击任务下的“EtherCAT_Master”子节点，在编辑器中点击添加调用，在弹出的对话框中选择“PLC_RPG”程序。

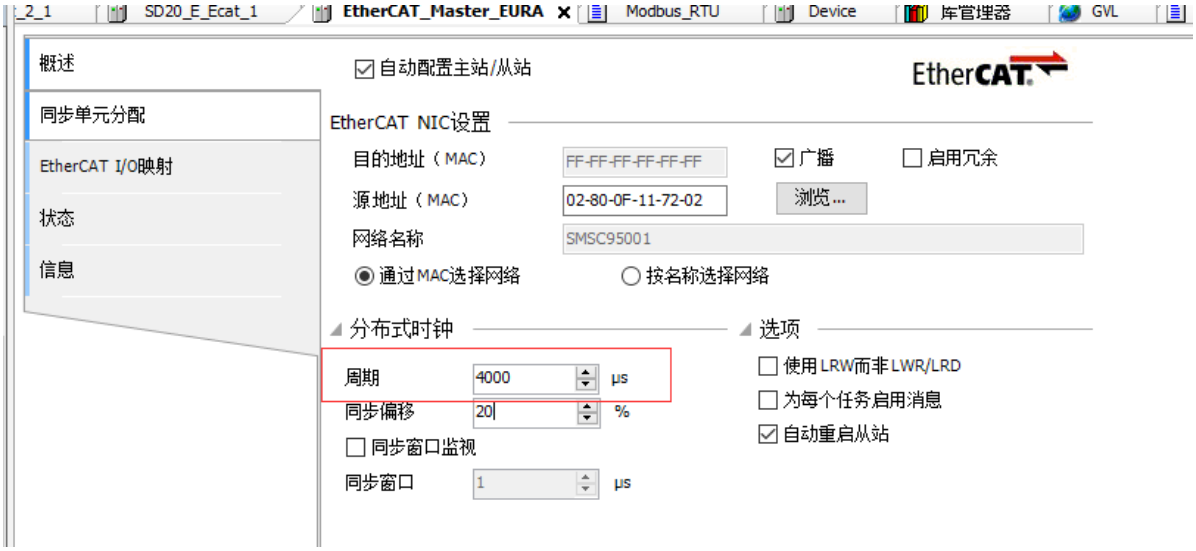


选择任务循环类型为外部的，外部事件为 EVENT_ECM_DC。

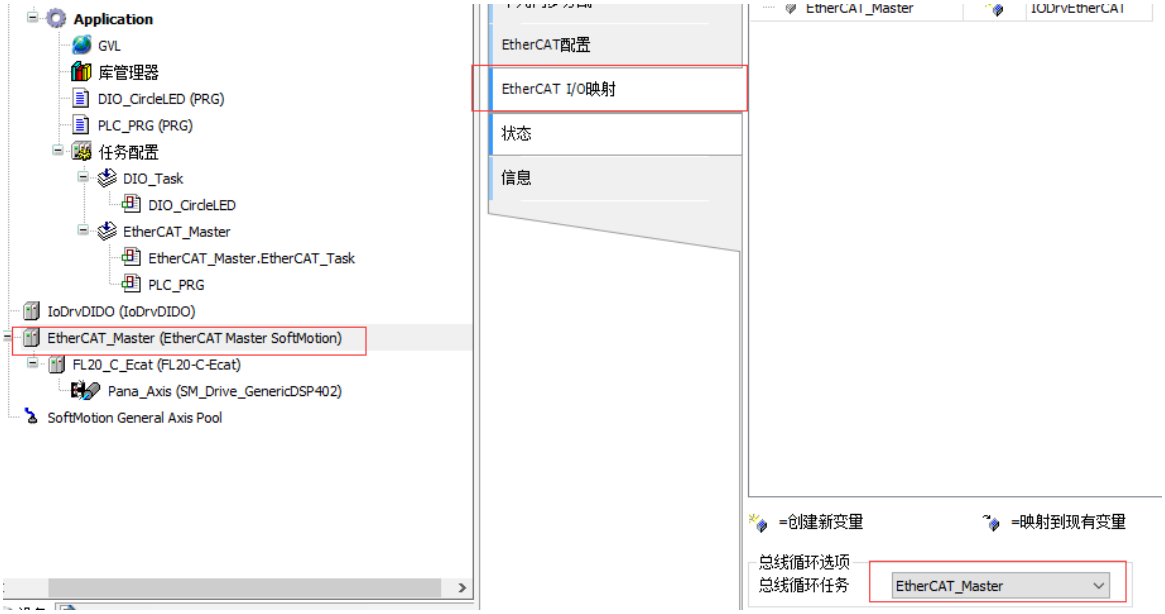


注：包含有 EtherCAT 任务和运动控制功能块程序的任务必须使用外部事件方式

双击设备树下的“EtherCAT_Master”设备，在概述页中修改通讯周期，并选择“按名称选择网络”。

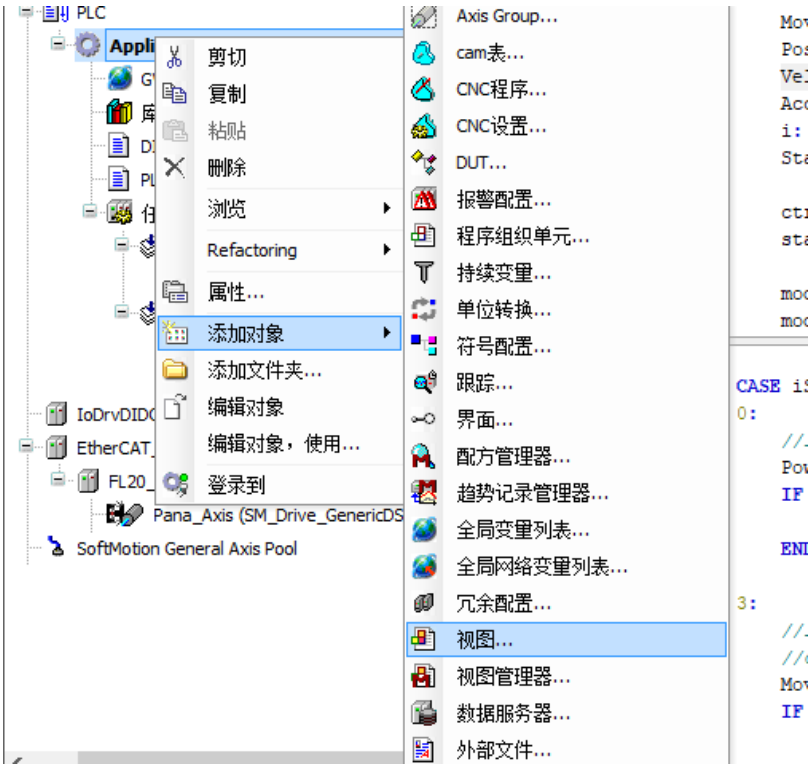


在“EtherCAT I/O 映射”选项卡中修改总线循环任务为 EtherCAT_Master；

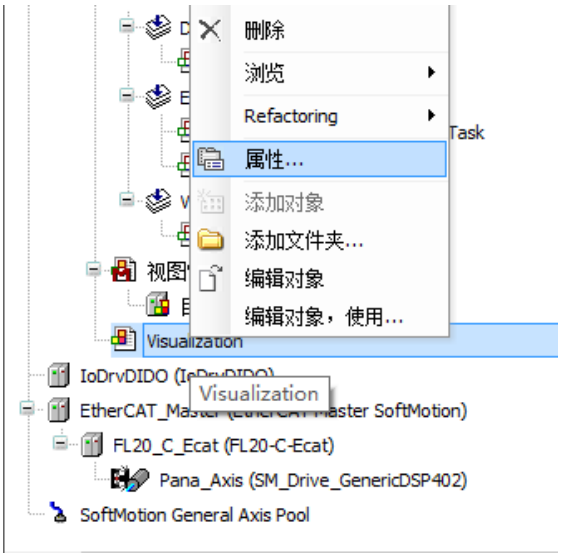


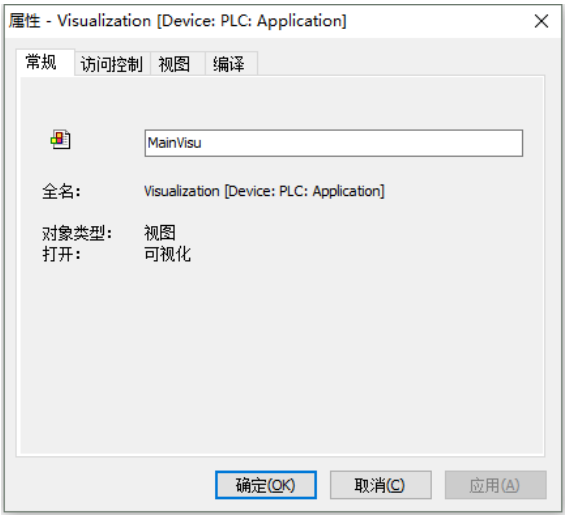
3.3.7.6 界面部分

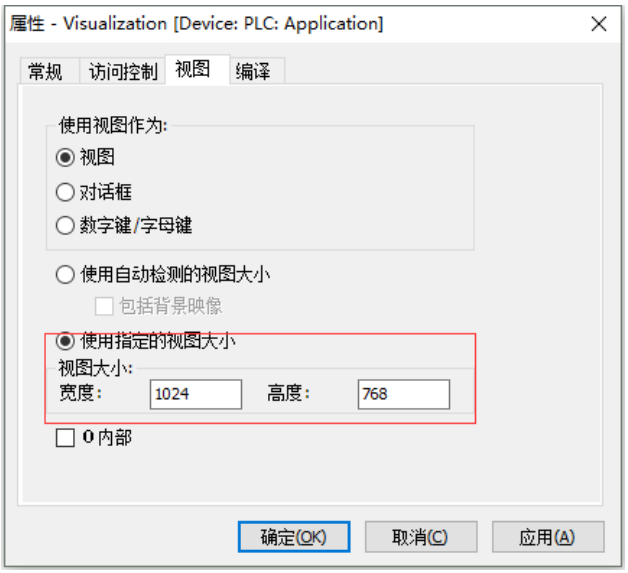
添加视图管理器：在右侧设备树选项卡 Application 节点右击，选择“添加对象”—“视图”。



在设备树下默认视图“Visualization”右击，在弹出菜单中选择属性，在弹出的对话框常规选项卡中修改名称为“MainVisu”，在视图选项卡中选择“使用指定视图大小”。点击确定完成设置。

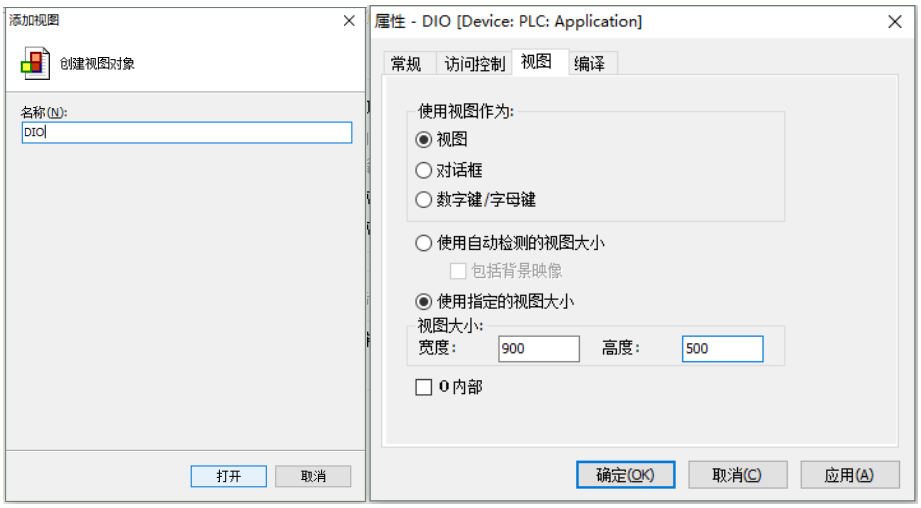




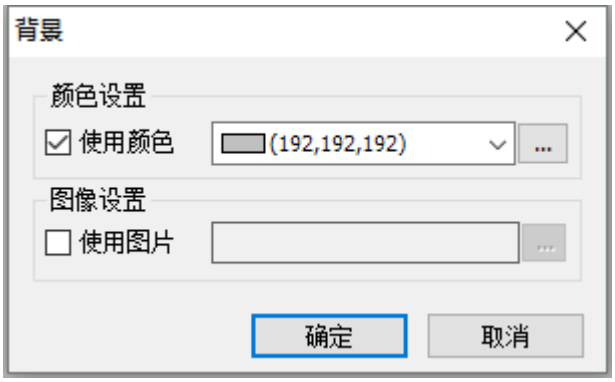


添加新视图：选项卡 Application 节点右击，选择“添加对象”—“视图”，在弹出的对话框中输入新视图的名称“DIO”，参照“MainVisu”的设置方法设置视图大小为 900×500。

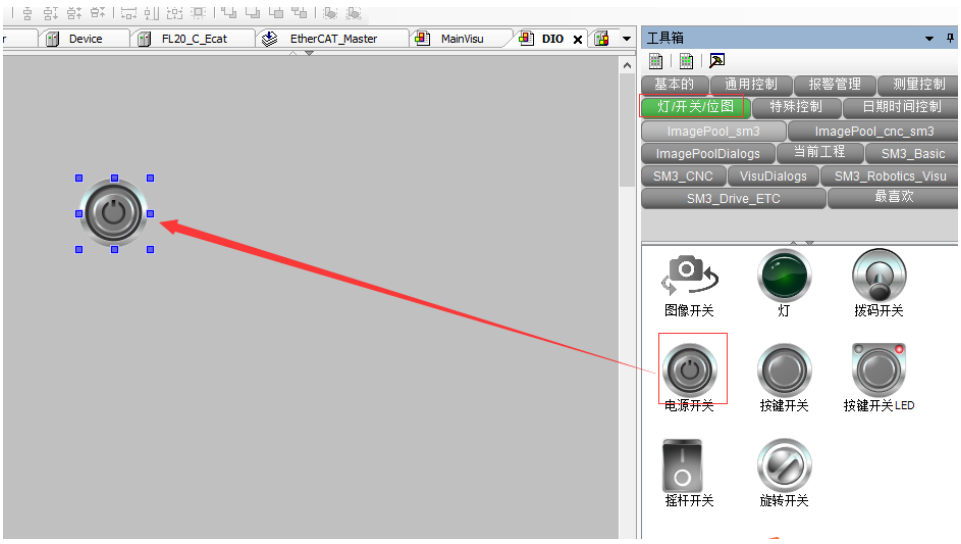
同理再添加三个视图“Motor”、“RS232”、“RS485”，视图大小同样为 900×500。



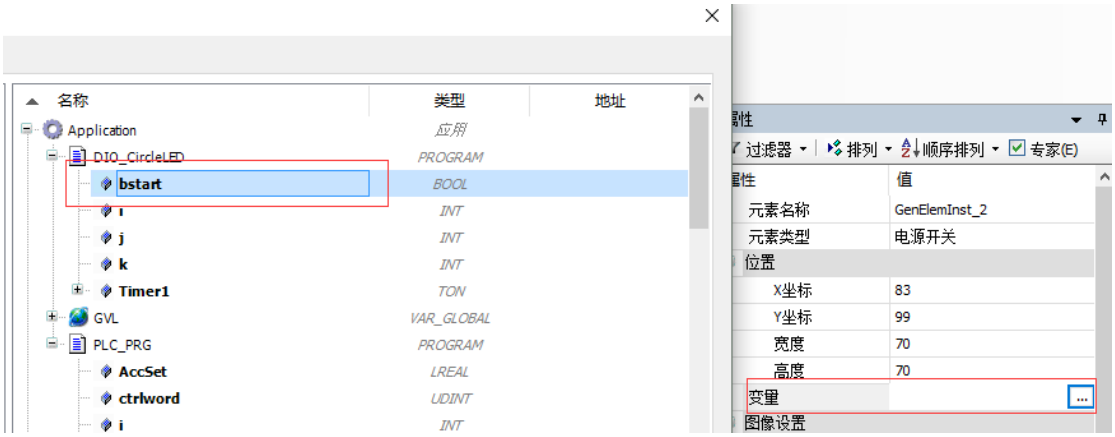
双击设备树中的 DIO 视图打开编辑器，右击编辑区，在菜单中选择“背景”，在弹出的对话框中选择“使用颜色”并将背景颜色更改为灰色。



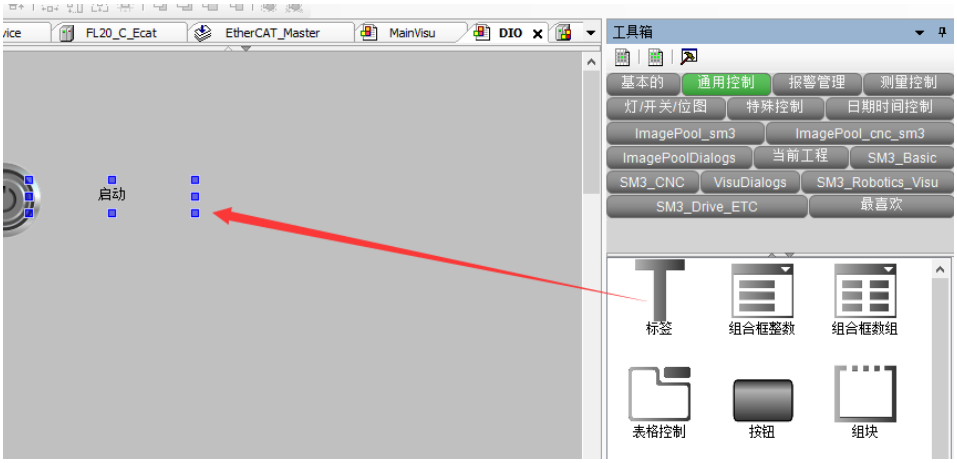
在编辑区右侧工具栏中选择“灯/开关/位图”选项卡，在下方选择合适的开关图形拖至编辑区的合适位置。



在右侧属性栏中变量一栏中点击“...”按钮在弹出的对话框中选择“bStart”变量，点击确定完成变量映射。

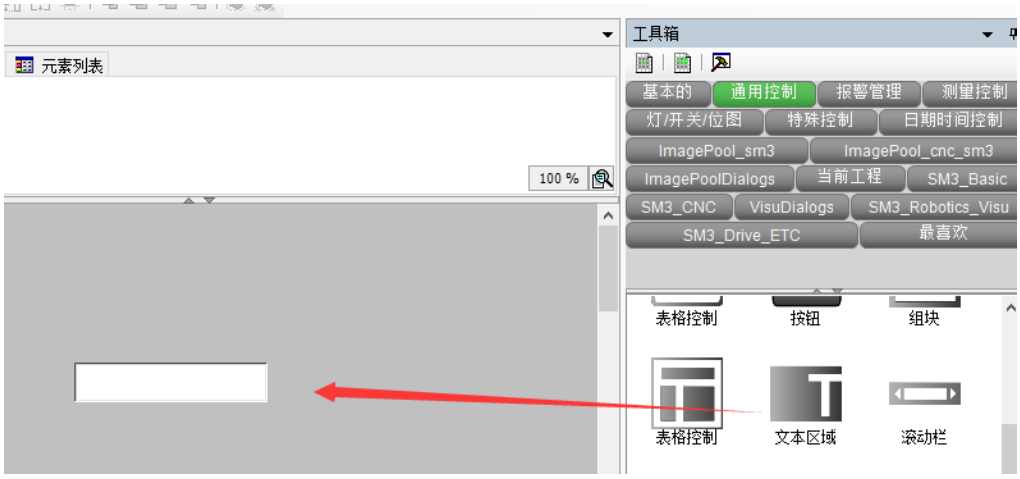


在工具栏通用控制选项卡中选择标签并拖到合适位置，点击编辑区的标签输入需要的内容，同时可以在标签的属性栏中设置字体和字号。

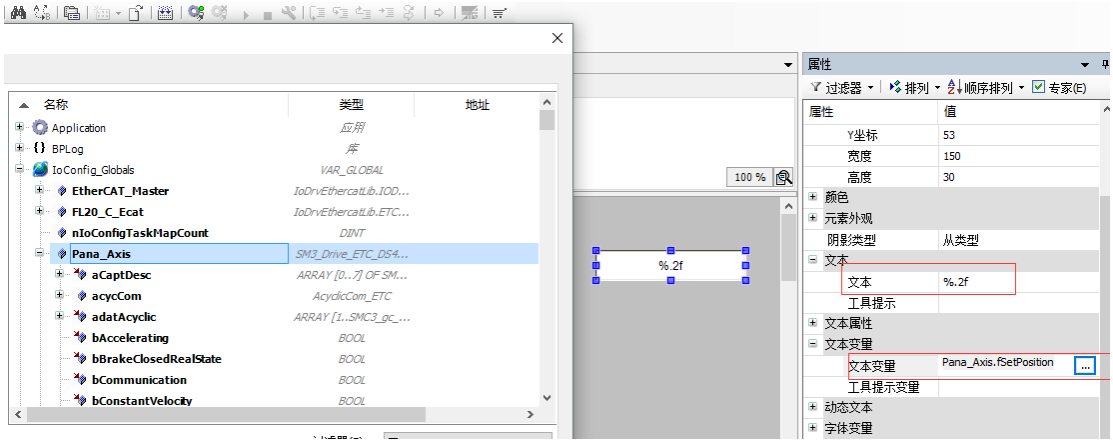


仿照上述方法，添加“灯/开关/位图”选项卡中的灯的图像，同样在属性中映射到全局变量的 DIn 和 DOut 数组，并添加文字标签作为说明。

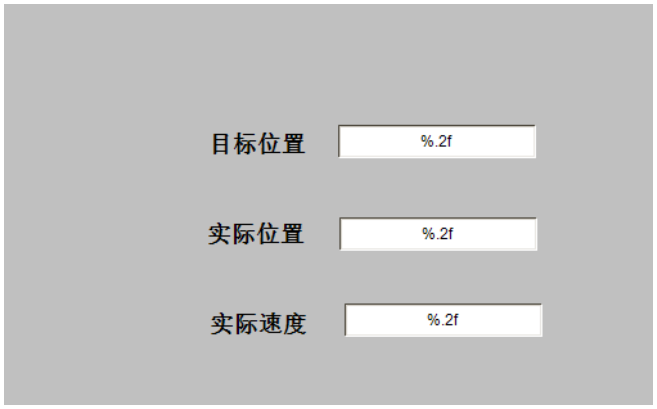
双击 Motor 视图，按照同样的方法设置背景颜色，在工具栏通用控制选项卡中选择文本区域图形并拖到合适位置。



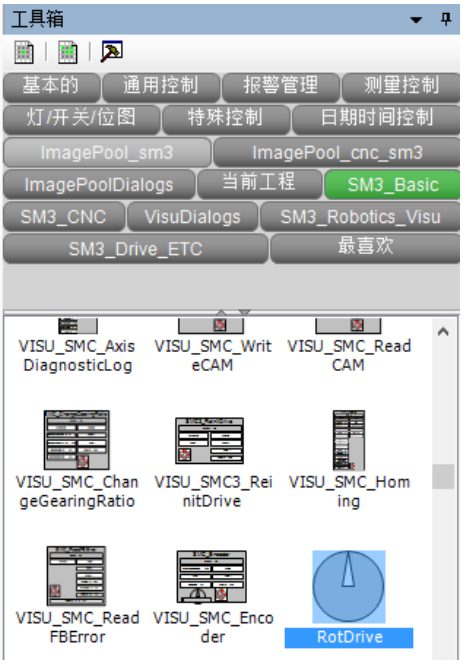
点击编辑区的“文本区域”控件，在属性栏文本一栏中输入格式符“%.2f”（格式符的具体说明参见 CODESYS 自带的帮助文档）。在文本变量一栏中关联变量“Pana_Axis.fSetPosition”。



同理，添加显示实际位置和实际速度的文本区域控件，分别关联“Pana_Axis.fActPosition”和“Pana_Axis.fActVelocity”变量。

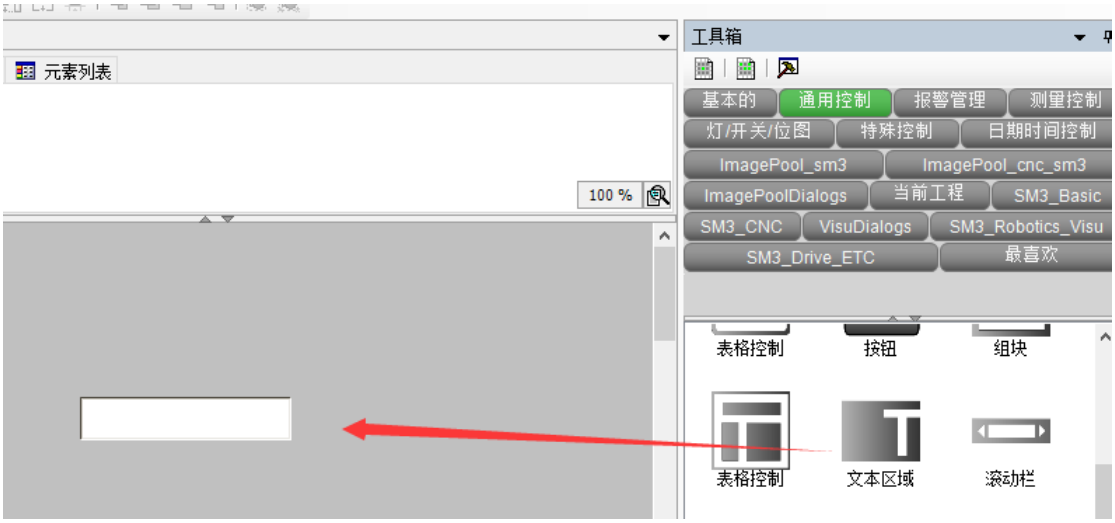


在工具箱“SM3_BASIC”选项卡中选择“RotDrive”控件，拖至编辑区。

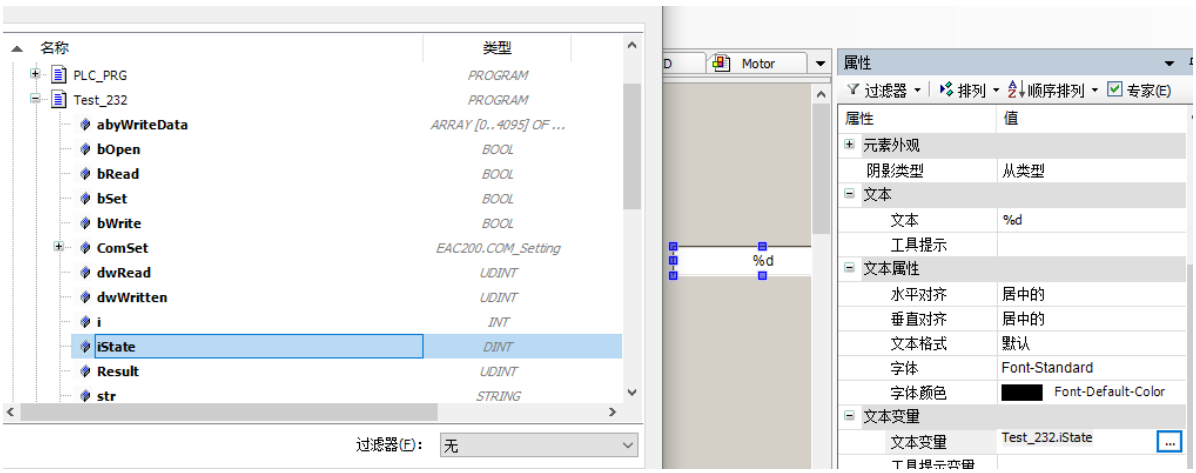


在弹出的对话框中值中输入“Pana_Axis”进行关联。

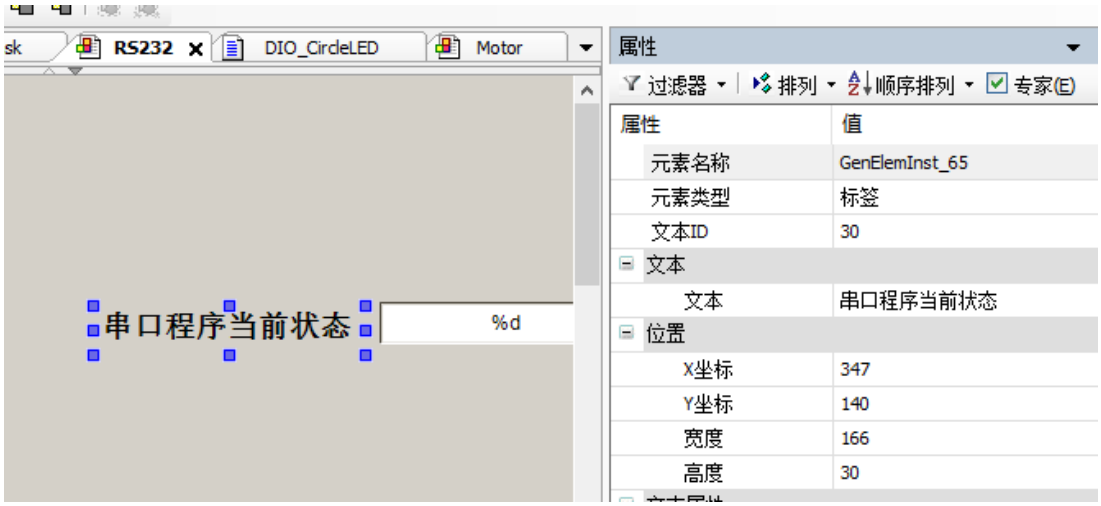
双击 RS232 视图，设置好背景颜色之后，在工具栏通用控制选项卡中选择文本区域图形并拖到合适位置。



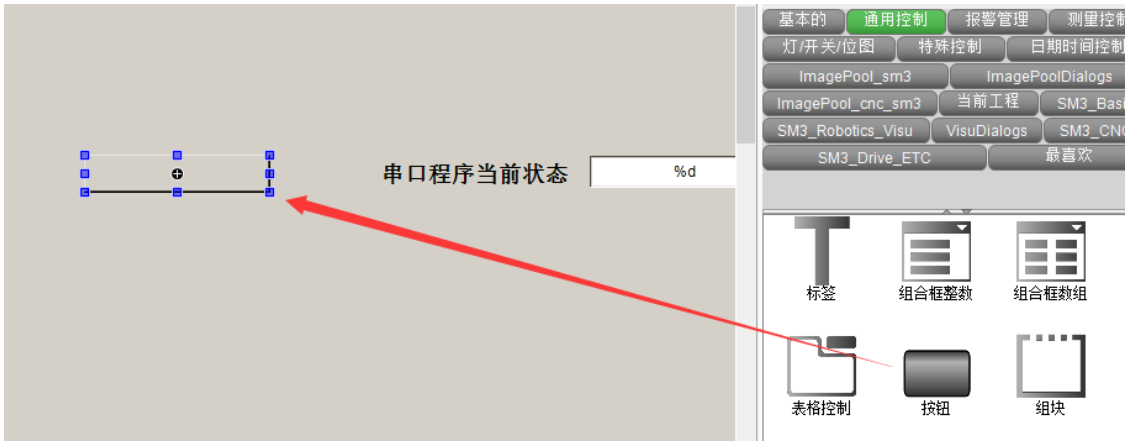
点击编辑区的“文本区域”控件，在属性栏文本一栏中输入格式符“%d”（格式符的具体说明参见 CODESYS 自带的帮助文档）。在文本变量一栏中关联变量“Test_232.iState”。



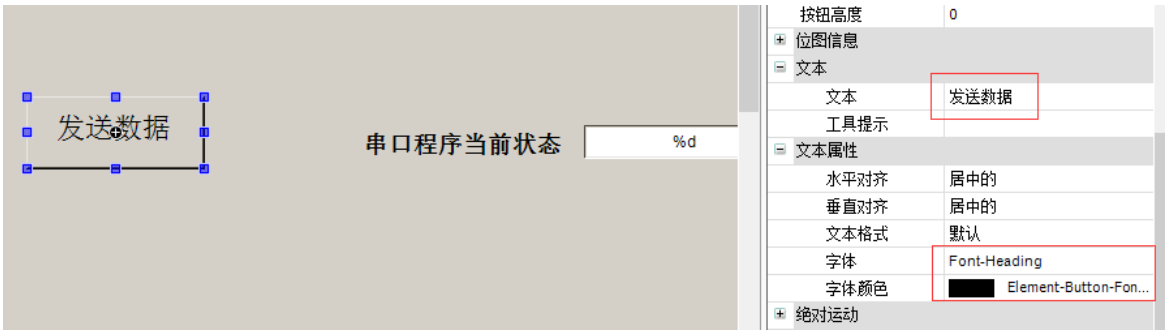
添加一个标签控件用于对文本区域进行说明：



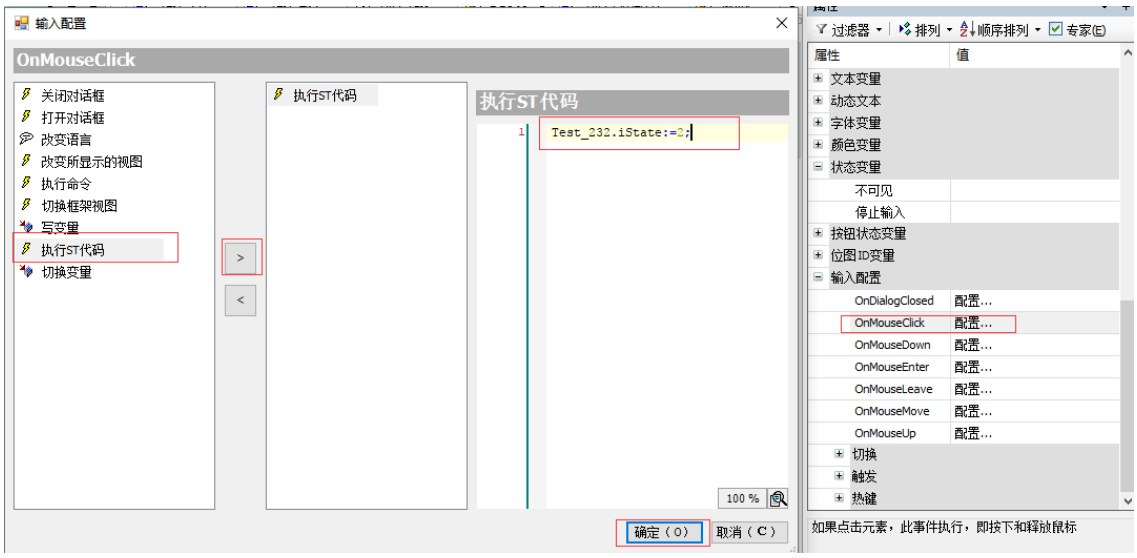
在工具箱中选择按钮控件，拖动到指定位置，并缩放至合适大小。



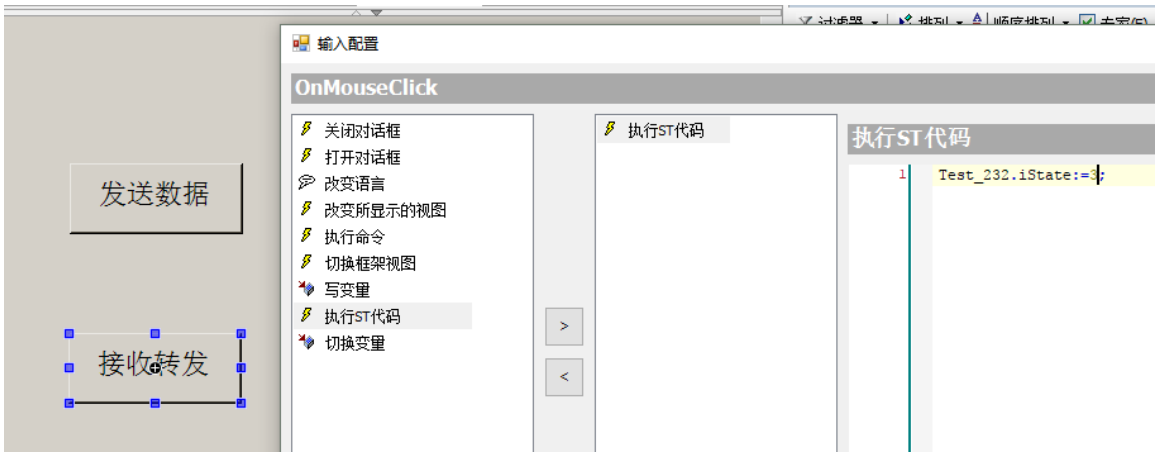
点击按钮控件，在属性栏中“文本”一栏中输入“发送数据”，并设置合适的字体大小和颜色。



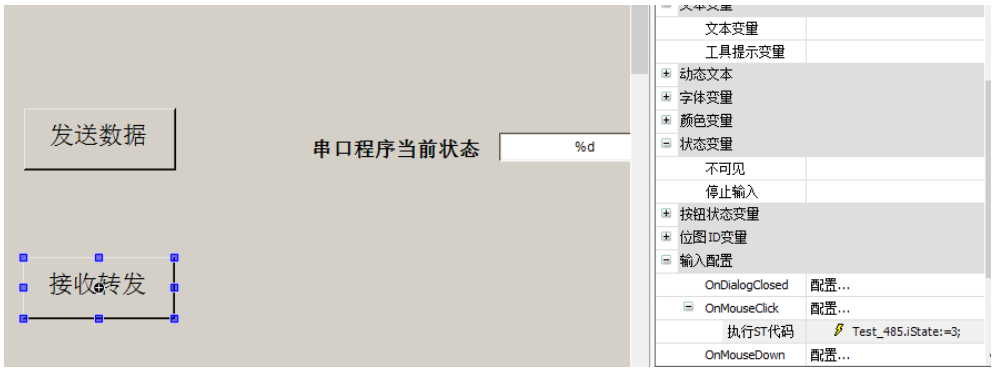
在属性栏中找到“输入配置”属性，点击“OnMouseClick”右侧的“配置...”按钮；在弹出的对话框中选中执行 ST 代码，点击“>”按钮使其生效，在对话框右侧编辑区中编写代码“Test_232.iState:=2;”，单击确定。



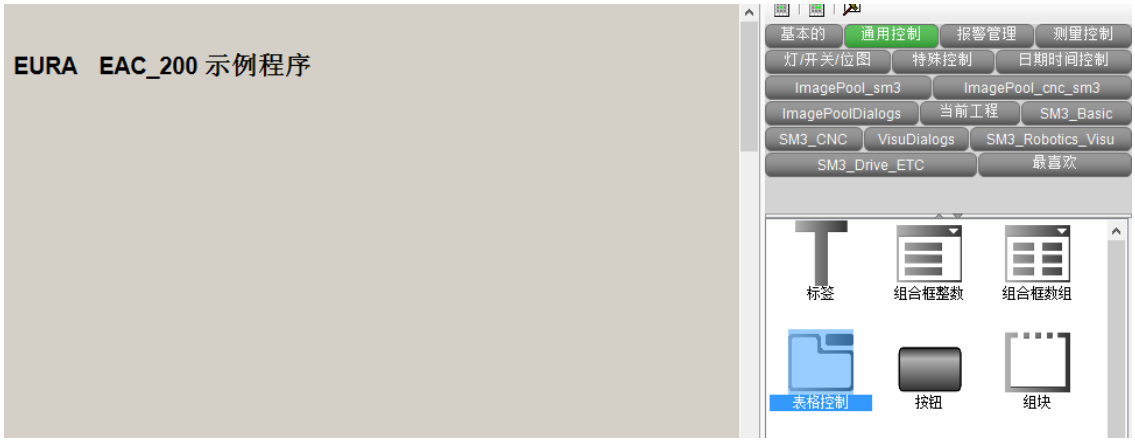
同理添加“接收转发”按钮，执行 ST 代码的编辑区中代码为“Test_232.iState:=3;”。



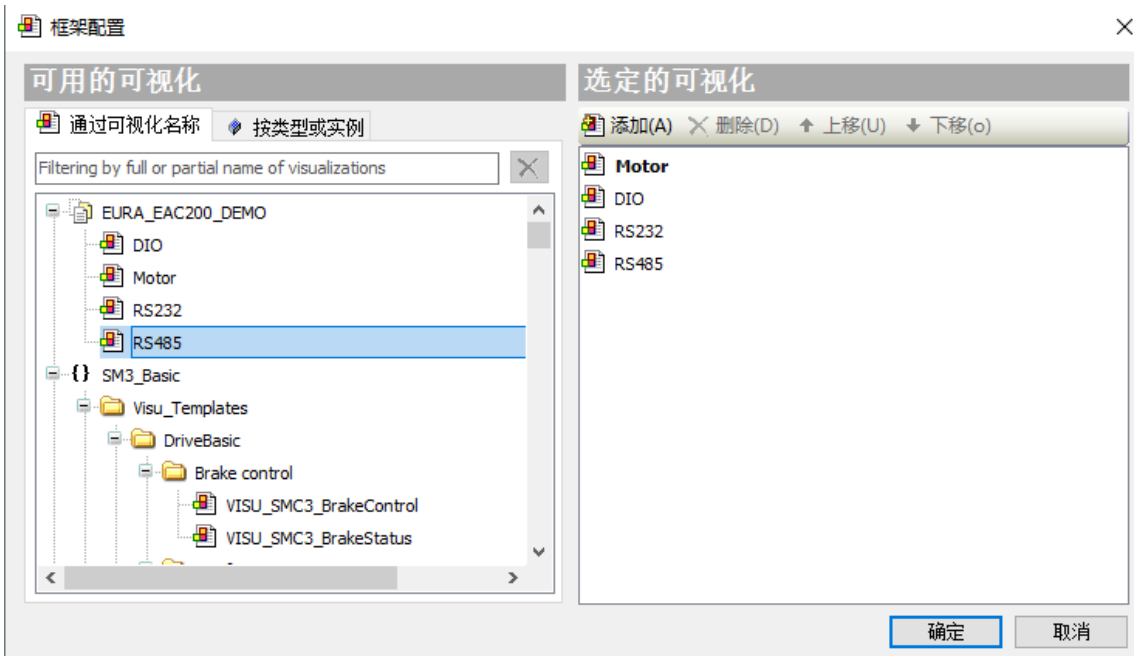
双击“RS485”视图，控件添加同“RS232”视图，按钮执行的 ST 代码更改为“Test_485.iState:=2;”和“Test_485.iState:=3;”，文本区域文本变量更改为“Test_485.iState”。



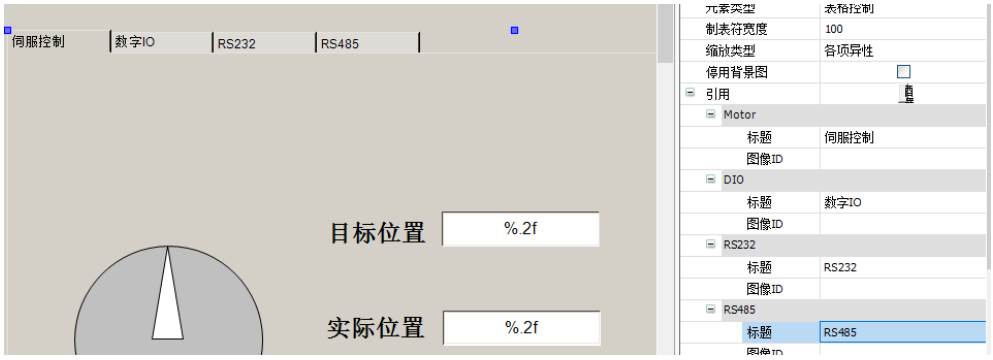
双击“MainVisu”视图，设置好背景和标题，在工具栏通用控制中选择“表格控制”拖至编辑区。



在弹出的对话框中选择“DIO”、“Motor”、“RS232”、“RS485”四个视图，点击添加按钮添加到右侧。点击确定。

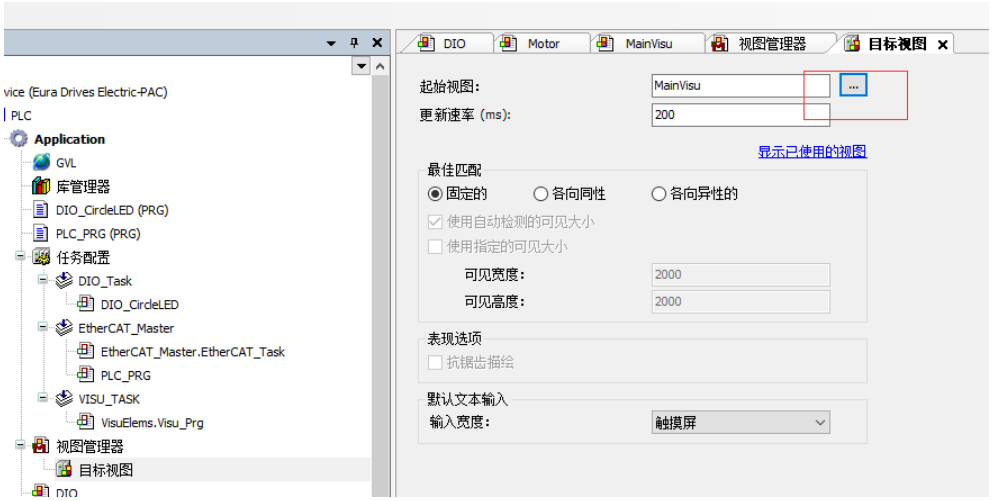


调整控件大小，并在属性中填写对应的标题。



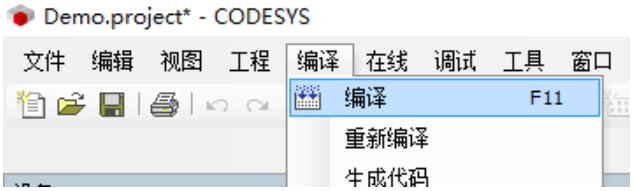
视图设置：双击视图管理器，在编辑区中勾选“使用 Unicode 字符串”和“使用当前视图变量”。

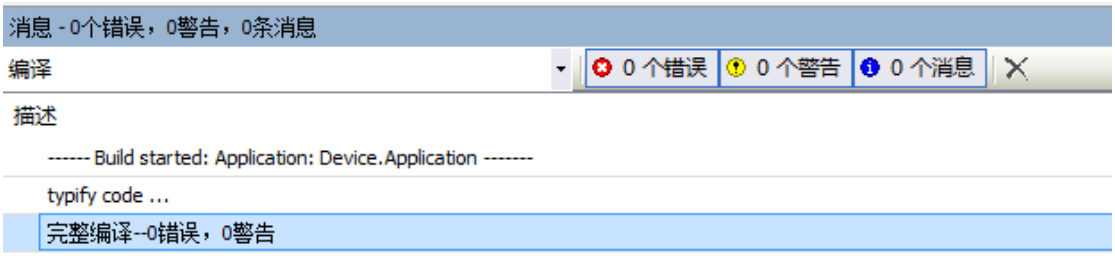
初始视图设置：双击设备树中的目标视图节点，在编辑区点击起始视图右侧的“...”按钮，在弹出的对话框中选择“MainVisu”作为初始视图。



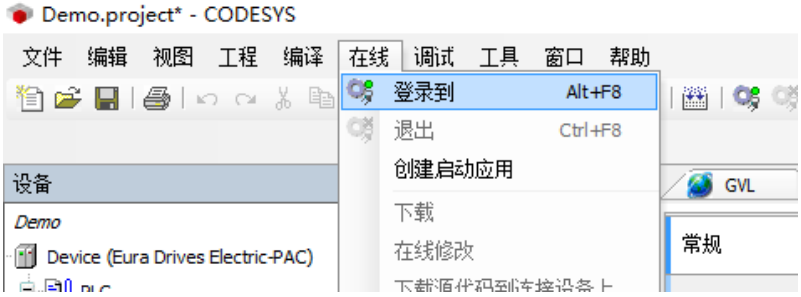
3.3.7.7 程序编译与下载

程序编译：点击菜单栏“编译”菜单下的“编译”选项编译程序，编译信息会出现在消息区域中，如果有错误双击错误可以进行定位。





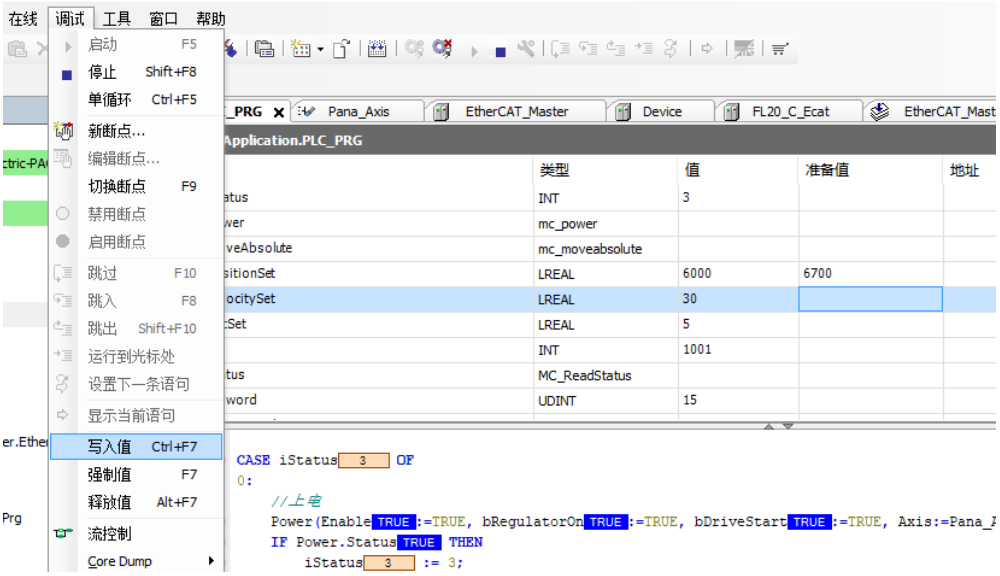
程序下载：连接 EAC 设备并在上位机软件中设置 EAC 设备为活动路径（参见 3.6.3 节的在线扫描部分），点击菜单栏中的“在线”——“登录到”，在弹出的对话框选择确认下载。



程序下载完成后，处于停止状态，点击菜单栏上的启动运行程序。

3.3.7.8 在线监控及在线修改

在线下载程序后原本的变量定义区变更为在线变量监控区，用户可以将需要调试的变量值写入对应变量的准备值一栏中，然后点选菜单栏中的“调试”—“写入值”进行变量的在线修改。



第四章　　故障分析与解决

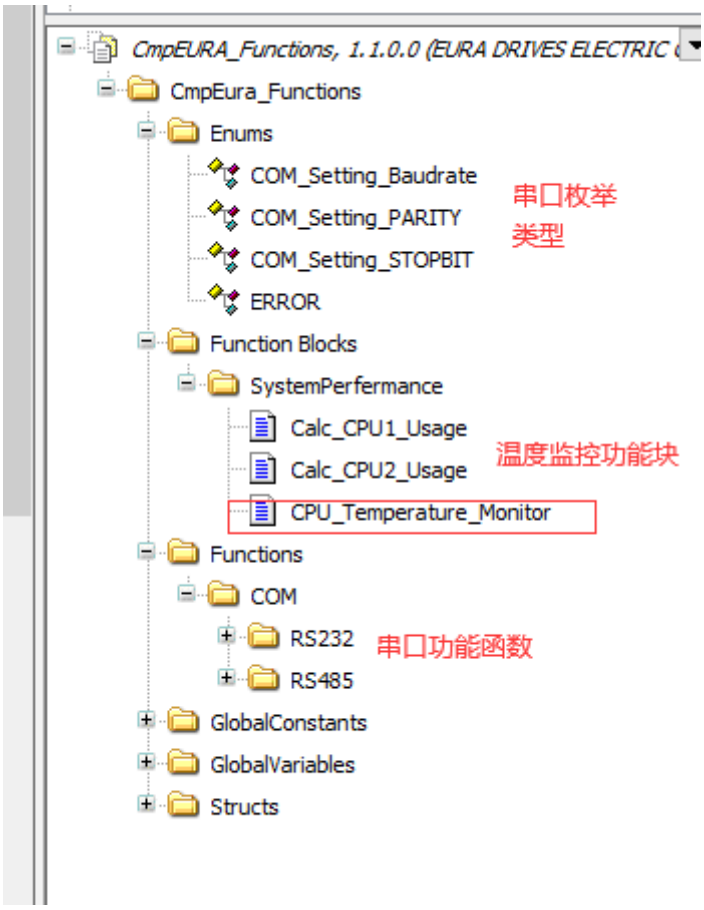
序号	现象	数码管显示	指示灯	原因	解决方案
1	系统上电后不启动	无显示	无显示	供电电源无电压	断电检查外部电源
2		无显示	PWR 红灯常亮	电源接反	检查电源极性
3	系统启动后不进入 CODESYS	2	PWR 绿灯常亮 ERR 灯不亮	DIP 拨码设置为不自动启动 CODESYS	根据需求设置拨码后重启
4		d	ERR 红灯常亮	因 RTC 电池掉电或其他原因导致系统时钟复位（系统年时间<2016）	可通过网页配置重新设置时间，重启后仍提示该错误请更换电池，可以通过拨码设置屏蔽该错误
5		E	ERR 红灯常亮	系统欠压	断电检查外部电源，修复后上电
6	设备程序下载后不运行或运行过程突然停止	A	ERR 红灯常亮	程序中存在死循环等导致系统卡死的代码	断电重启，并使用拨码开关清除设备内现有程序，重新下载修改后的程序
7		b	ERR 红灯常亮	EtherCAT 总线通讯中断或者出错	检查网线是否存在松动，重新连接后重启
8		C	ERR 红灯常亮	程序或通讯中出现异常导致 CODESYS 退出	断电重启，并使用拨码开关清除设备内现有程序，重新下载修改后的程序
9		E	ERR 红灯常亮	系统欠压	断电检查外部电源，修复后重新上电
10		F	ERR 红灯常亮	系统掉电	检查外部电源，修复后重新上电
11	系统正常运行时	h 闪烁	ERR 红灯闪烁	高温预警	
12		H	ERR 红灯常亮	运动控制器芯片温度过高	
13	上电后扩展 IO 模块无效	N/A	扩展 IO 模块 STAT 灯红色闪烁	本体与扩展 IO 通讯异常	重新上电

附录 1：CmpEuraFunctions 组件库说明：

CmpEuraFunctions 组件库用于提供欧瑞自己的函数和功能块实现，目前实现的功能有以下两种：

1.串口自由通讯相关枚举及功能函数。

2.系统温度监控功能块。



附录 1.1 串口自由通讯相关枚举及功能函数

枚举数据定义：

COM_Setting_Baudrate

说明：该枚举类型用于提供串口波特率值的选择

枚举值：

名称	值
CBR_4800	4800
CBR_9600	9600
CBR_19200	19200
CBR_38400	38400
CBR_57600	57600
CBR_115200	115200

COM_Setting_PARITY

说明：该枚举类型用于提供串口校验类型的选择

枚举值：

名称	值	说明
NOPARITY	0	无校验
ODDPARITY	1	奇校验
EVENPARITY	2	偶校验

COM_Setting_STOPBIT

说明：该枚举类型用于提供串口停止位类型的选择

枚举值：

名称	值	说明
ONESTOPBIT	0	1 位停止位
TWOSTOPBITS	2	2 位停止位

结构体数据定义：

COM_Setting

说明：该结构体整合串口设置需要的波特率、数据位、停止位和校验类型，用于提供给功能块进行串口配置。

结构成员：

名称	类型	说明
Baudrate	COM_Setting_Baudrate	用于给定串口的波特率
Parity	COM_Setting_PARITY	用于给定串口的校验类型
Databits	DINT	用于给定串口的数据位
StopBit	COM_Setting_STOPBIT	用于给定串口的停止位

函数定义：

OpenCom_RS232

说明：该函数用于打开 RS232 串口。

返回值：TRUE 表示串口打开成功，FALSE 表示打开失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

CloseCom_RS232

说明：该函数用于关闭 RS232 串口。

返回值：TRUE 表示串口关闭成功，FALSE 表示关闭失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

SetCom_RS232

说明：该函数用于对 RS232 串口参数进行设置。

返回值：TRUE 表示串口设置成功，FALSE 表示关闭失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
ComSetting	输入	COM_Setting/	用于串口通讯参数的给定
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

SendCom_RS232

说明：该函数用于 RS232 串口向外发送数据。

返回值：TRUE 表示数据发送成功，FALSE 表示发送失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
pbyWriteBuffer	输入	POINTER TO BYTE	数据指针，指向存有需要发送数据的字节型数组。
ulNumOfBytesToSend	输入	UDINT	需要发送的字节数
ulActualBytesSend	输出	UDINT	实际发送的字节数
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

注：当串口连接的设备对于接受到数据有返回时，发送函数 SendCom_RS232 和接收函数 ReceiveCom_RS232 应成对使用，或者延时足够长时间后才能再次调用发送函数，否则发送数据和和设备返回数据形成干扰，无法正常通信。

RecevieCom_RS232

说明：该函数用于接收 RS232 串口获得的数据。

返回值：TRUE 表示数据接收成功，FALSE 表示接收失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
pbyReadBuffer	输入	POINTER TO BYTE	数据指针，指向需要存放接收数据的字节型数组。
ulNumOfBytesToRecv	输入	UDINT	需要接收的字节数
ulActualBytesRecv	输出	UDINT	实际接收的字节数
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

OpenCom_RS485

说明：该函数用于打开 RS485 串口。

返回值：TRUE 表示串口打开成功，FALSE 表示打开失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

CloseCom_RS485

说明：该函数用于关闭 RS485 串口。

返回值：TRUE 表示串口关闭成功，FALSE 表示关闭失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

SetCom_RS485

说明：该函数用于对 RS485 串口参数进行设置。

返回值：TRUE 表示串口设置成功，FALSE 表示关闭失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
ComSetting	输入	COM_Setting	用于串口通讯参数的给定
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

SendCom_RS485

说明：该函数用于 RS485 串口向外发送数据。

返回值：TRUE 表示数据发送成功，FALSE 表示发送失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
pbyWriteBuffer	输入	POINTER TO BYTE	数据指针，指向存有需要发送数据的字节型数组。
ulNumOfBytesToSend	输入	UDINT	需要发送的字节数
ulActualBytesSend	输出	UDINT	实际发送的字节数
Result	输出	UDINT	返回串口失败代码，0 表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

注：当串口连接的设备对于接受到数据有返回时，发送函数 SendCom_RS485 和接收函数 ReceiveCom_RS485 应成对使用，或者延时足够长时间后才能再次调用发送函数，否则发送数据和和设备返回数据形成干扰，无法正常通信。

RecevieCom_RS485

说明：该函数用于接收 RS485 串口获得的数据。

返回值：TRUE 表示数据接收成功，FALSE 表示接收失败，失败代码通过参数 Result 给出。

输入输出参数：

名称	输入输出类型	类型	说明
pbyReadBuffer	输入	POINTER TO BYTE	数据指针，指向需要存放接收数据的字节型数组。
ulNumOfBytesToRecv	输入	UDINT	需要接收的字节数
ulActualBytesRecv	输出	UDINT	实际接收的字节数
Result	输出	UDINT	返回串口失败代码，0表示无错误，其余值含义可以查询微软标准的系统错误代码（System Error Codes）

附录 1.2 系统温度监控功能块

CPU_Temperature_Monitor (FB)

说明：该功能块用于监控系统温度，并进行高温报警和预警输出提示。

输入输出参数：

名称	输入输出类型	类型	说明
bStart	输入	BOOL	为 True 时启动功能块功能
bError	输出	BOOL	错误标志位
uiErrorCode	输出	UINT	错误代码： 0：无错误 1：传感器打开失败， 2 传感器读取温度失败
uiCPUTemperature	输出	UINT	CPU 当前温度
bHighTemperature_Waring	输出	BOOL	CPU 高温警告 t>105 ℃
bHighTemperature_PreWaring	输出	BOOL	CPU 高温预报警 t>90℃

附录 2：Modbus RTU 使用说明：

概述：

Modbus 是一种串行异步通讯协议。广泛用于 PLC 、控制器、触摸屏等设备间的数据交互。Modbus RTU 常用的物理接口为串口。EAC20 系列运动控制器具有 1 个 RS232 接口和 1 个 RS485 接口，均可以独立的作为 Modbus 主站/从站使用。其技术参数如下表所示：

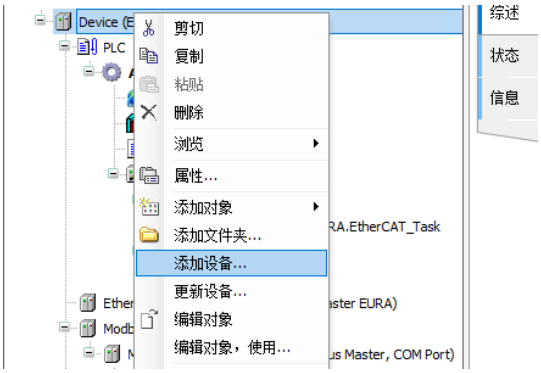
Modbus 主站/从站	
物理接口	RS232/RS485
通讯格式	RTU
波特率	4800/9600/19200/38400/57600/115200
校验位	无校验/奇校验/偶校验
数据位	8 位
停止位	1 位（奇偶校验时）/2 位（无校验位时）
支持的 Modbus 功能码	03/04/05/06/15/16

Modbus RTU 主站及其实现：

本节将详细说明在上位机软件 CODESYS 中如何配置 Modbus 主站，并建立与从站通讯的通道。在 CODESYS 中 Modbus RTU 主站及端口的实现以各种设备组件的形式存在。

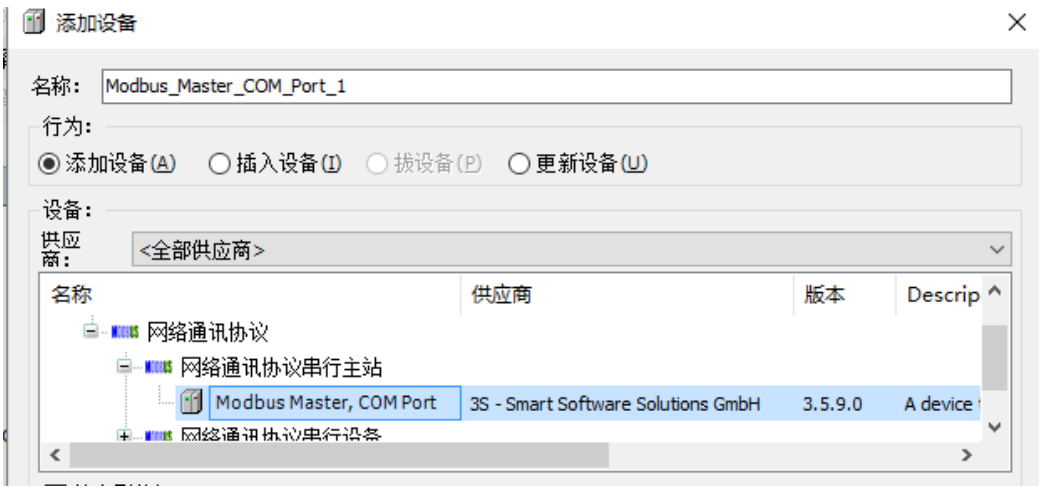
1. 安装串行接口设备文件：从本公司网站下载需要的设备描述文件：EAC200_ModbusPort_RS232.xml 和 EAC200_ModbusPort_RS485.xml，并依照说明书中的 3.3.5 设备描述文件的安装和更新一节安装 2 个设备描述文件到 CODESYS 软件中。

2. 添加端口：在 CODESYS 程序左侧的设备树 Device 节点右击，选择添加设备。在弹出的对话框中根据需要选择现场总线——网络通讯协议——网络通讯协议端口下的 Modbus_Port_RS232 设备或者 Modbus_Port_RS485 设备，点击确定。

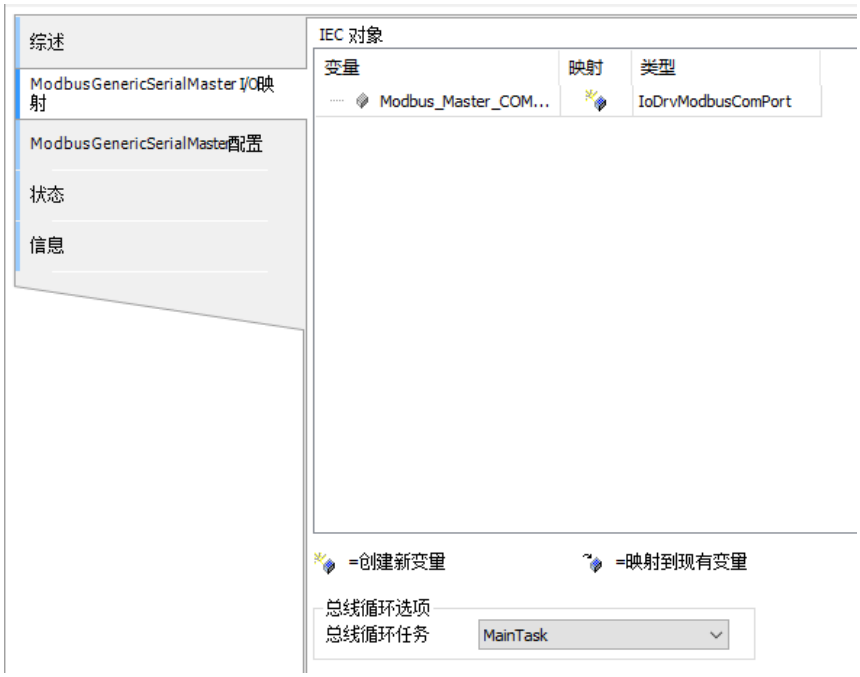


3. 设置串口通讯格式：双击右侧设备树上的端口设备 Modbus_Port_RSXXX，在右侧的综述标签页中配置串口的基本参数，使其与通讯的从站相对应。

4. 添加 ModbusRTU 主站：在左侧的设备树端口设备节点 Modbus_Port_RSXXX 右击，选择添加设备。在弹出的对话框中根据需要选择现场总线——网络通讯协议——网络通讯协议串行主站下的 Modbus_MasterComPort 设备，点击确定。

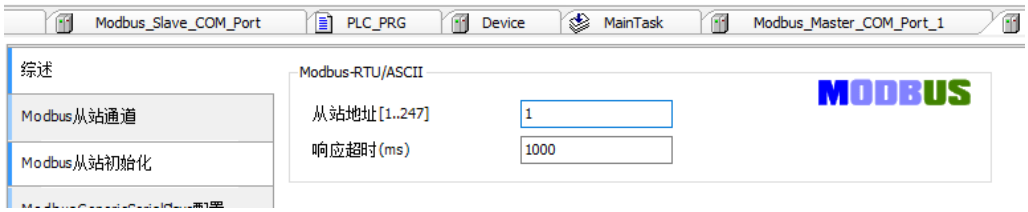


5. 设置主站设备：双击右侧设备树上的 Modbus 主站设备 Modbus_MasterComPort，在右侧的 I/O 映射标签页中设置总线循环任务为主程序任务。

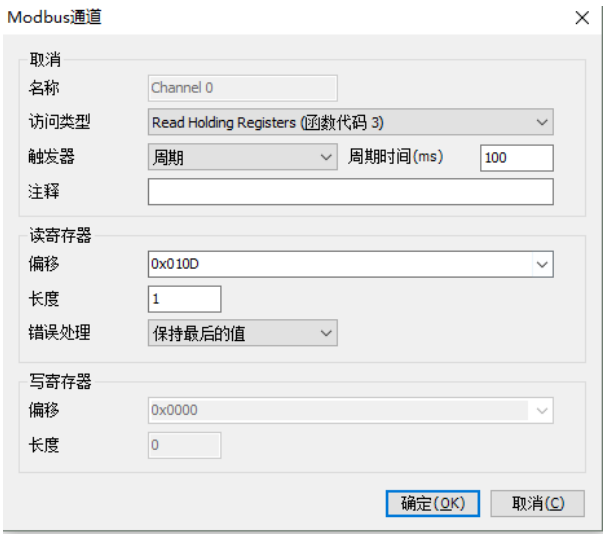


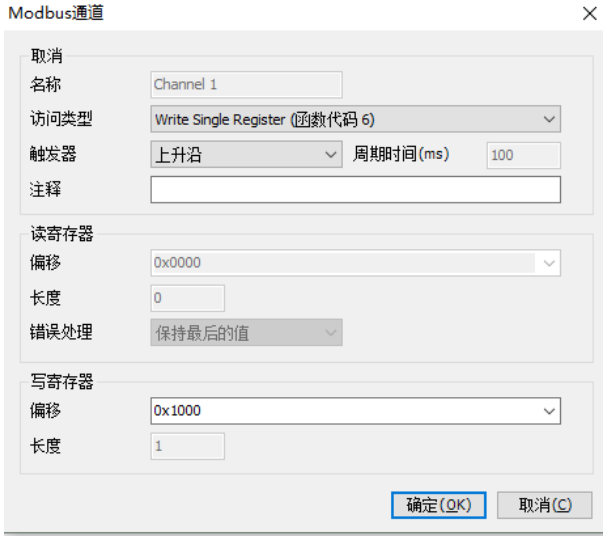
6. 添加 Modbus RTU 从站：在左侧的设备树端口设备节点 Modbus_MasterComPort 右击，选择添加设备。在弹出的对话框中根据需要选择现场总线——网络通讯协议——网络通讯协议串行从站下的 Modbus_Slave 设备，点击确定。

7. 配置 Modbus RTU 从站：双击右侧设备树上的 Modbus 从站设备，在右侧的综述标签页中设置从站站号。

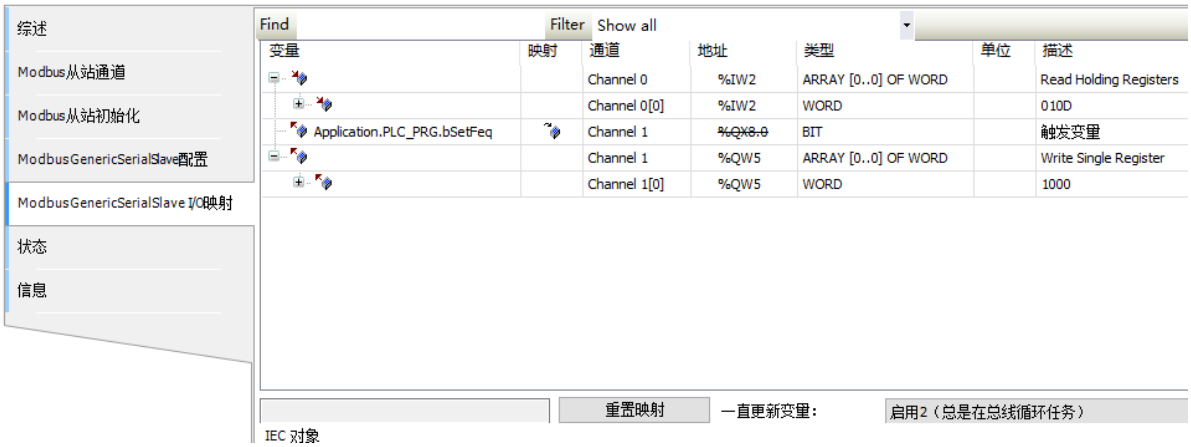


8. 建立从站通道：在 Modbus 从站通道标签页中点击添加通道按钮，在弹出的对话框中选择读取/写入方式、触发方式、从站目的地址和数据长度；变量的触发方式分为周期和上升沿两种。周期为按照设定的周期时间定时进行写入/读取,大多用于对从站常用数值或监控状态的读取；上升沿方式则是当指定变量状态产生上升沿变化时（由 FALSE 变为 TRUE）触发对目的地址的写入和读取，通常用于对从站的指令写入和非常用状态的读取。





9. 映射从站变量：建立完从站通道后选择 IO 映射选项卡，在这个页面中可以将建立的从站通道与程序中定义的变量进行映射关联，使读取到的数据直接赋值给程序中的变量，同时使程序中变量的数据写入到目的地址。最后将一直更新变量选项更改为总是在总线任务，完成主站的创建。



Modbus RTU 从站及其实现：

本节将详细说明在上位机软件 CODESYS 中如何配置 Modbus 从站。在 CODESYS 中 Modbus RTU 从站使用带有掉电保持的全局数组和功能块进行实现。

Modbus RTU 从站中的寄存器分为保持寄存器和输入寄存器两类。保持寄存器中的数值既可以被主站读取，也可以被主站改写，其地址从 4x0000 开始。输入寄存器只能被主站读取，不能被主站改写，其地址从 3x0000 开始。

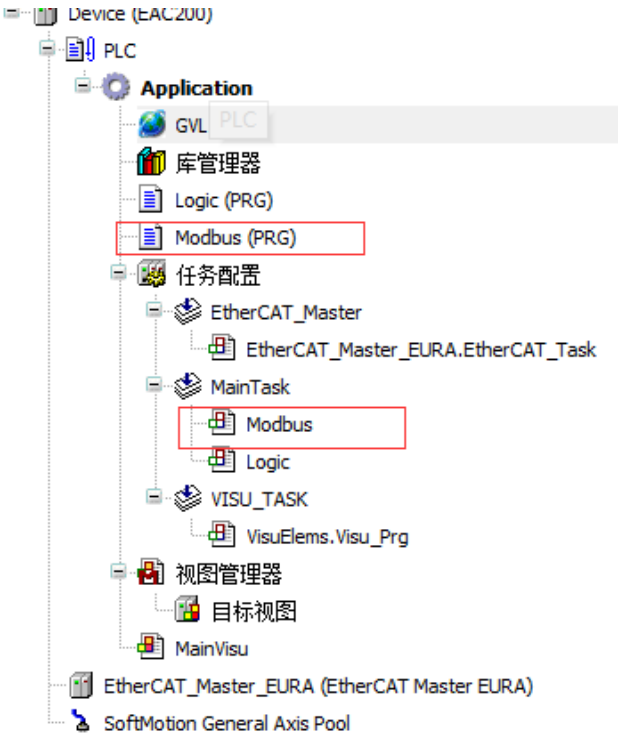
从站的实现步骤如下所示：

1. 添加全局寄存器变量：在 CODESYS 程序左侧的设备树 Application 节点右击，添加全局变量列表，在全局变量列表中定义输入和保持寄存器变量数组、寄存器大小变量、以及用于从站功能块的布尔型标志位变量，一个定义的实例如下所示：（注：用户可以根据实际需要更改寄存器数组大小和表示寄存器数目的变量，但要保证表示寄存器数目的变量的值小于等于所定义寄存器数组大小。）

```
VAR_GLOBAL RETAIN
    wHoldRegister: ARRAY[0..9] OF WORD;//保持寄存器变量数组
    wInputRegister:ARRAY[0..9] OF WORD;//输入寄存器变量数组
END_VAR
VAR_GLOBAL
    uiNumofHoldingReg:UINT:=10;//保持寄存器大小
    uiNumofOutputReg:UINT:=10;//输入寄存器大小
    bStart:BOOL:=FALSE; //modbus 从站功能块使能标志位
    bReset:BOOL:=FALSE;//modbus 从站功能块复位标志位
    bError:BOOL:=FALSE;//modbus 从站功能块错误标志位
END_VAR
```

2. 调用从站功能块，实现 Modbus 从站程序：新建一个子程序“Modbus”，并在任务配置

中调用该程序，推荐循环周期为 100ms；



双击子程序 Modbus，定义从站站号，串口数据结构体变量和 Modbus 从站功能块实例，一个示例的定义如下所示

```
PROGRAM Modbus
VAR
    uiStatus:UINT:=0;//状态机
    bySlaveID:BYTE:=1;//从站站号
    comSettings:ComSettings;//串口结构体
    ModbusServer: ModbusServer;//从站功能块
END_VAR
```

在程序实现中对串口数据结构体变量进行赋值设置，并调用 Modbus 从站功能块实例，如下所示：

```
CASE uiStatus OF
0:    //初始化
    comSettings.sPort:=1;//控制器串口号：  1：RS232   2:485
    comSettings.ulBaudrate:=9600;//波特率
    comSettings.byParity:=SYS_NOPARITY;//校验位
    comSettings.byStopBits:=SYS_ONESTOPBIT;//停止位
```

```
        comSettings.ulBufferSize:=32;//缓冲区大小
        comSettings.ulTimeout:=10;//超时时间
        uiStatus:=1;
    1:
        ModbusServer(
            byUnitID:= bySlaveID, //从站 ID
            comPortSettings:=comSettings , //串口配置
            pInputData:=ADR(gvl.wInputRegister) , //输出寄存器地址
            pOutputData:=ADR(gvl.wHoldRegister) , //保持寄存器地址
            uiInputDataSize:= gvl.uiNumofOutputReg, //保持寄存器数目
            uiOutputDataSize:=gvl.uiNumofHoldingReg , //输出寄存器数目
            xEnable:=gvl.bStart , //使能标志位
            xReset:=gvl.bReset ,
            tTimeout:= 0, xError=>gvl.bError );
    END_CASE
```

其中串口数据结构体变量中各个成员的取值如下表所示：

名称	变量	取值	说明
串口号	comSettings.sPort	1	控制器 RS232 接口
		2	控制器 RS485 接口
波特率	comSettings.ulBaudrate	4800/9600/19200/38400 /57600/115200	
校验位	comSettings.byParity	SYS_NOPARITY	无校验
		SYS_ODDPARITY	奇校验
		SYS_EVENPARITY	偶校验
停止位	comSettings.byStopBits	SYS_ONESTOPBIT	1 位停止位
		SYS_TWOSTOPBITS	2 位停止位

另外请注意 ModbusServer 功能块需要其使能输入 xEnable 上升沿触发输入并一直保持有效才能正常运行和通讯，因此需要在主程序中将全局变量 gvl.bStart 赋值为 True 并一直保持。

3. 寄存器中数据的使用：用户可以在主程序中使用和更改输入/保持寄存器全局变量数值中数值。对于布尔型变量的输出可以按位操作；对于浮点型变量，由于占用两个地址，因此需要先组合起来再进行读取，一些寄存器的读取示例如下所示：

定义部分：

```
PROGRAM Logic
VAR
  ftest1:REAL:=0; //接收并显示浮点数
  uiInupt:UINT:=0;//接收并显示整数
  bRecv0:BOOL:=FALSE;//接收布尔型数据
  bRecv1:BOOL:=FALSE;//接收布尔型数据
  bRecv2:BOOL:=FALSE;//接收布尔型数据
  bRecv3:BOOL:=FALSE;//接收布尔型数据

  uiOutput:UINT:=5678;//整数输出
  ftest2:REAL:=12.34;//浮点数输出

  bSend0:BOOL:=FALSE;//发送布尔型数据
  bSend1:BOOL:=FALSE;//发送布尔型数据
  bSend2:BOOL:=FALSE;//发送布尔型数据
  bSend3:BOOL:=FALSE;//发送布尔型数据

  //对多存器变量转换功能块
  Read_2_Register_to_LREAL1: Modbus_Trans.Read_2_Register_to_LREAL;
  Write_LREAL_to_2_Register1: Modbus_Trans.Write_LREAL_to_2_Register;
END_VAR

END_VAR
```

实现部分：

```
  //主站整数输入
      uiInupt:=gvl.wHoldRegister[0];
```

```
//主站浮点数输入
//浮点数（小数）占用 2 个 Modbus 字地址，但是不同的主站高低字节的顺序不同
//因此需要读取 modbus 重新合成浮点数
    Read_2_Register_to_LREAL1(HighRegister:=gvl.wHoldRegister[3],
    LowRegister:=gvl.wHoldRegister[2] , fOutputValue=>ftest1

//主站布尔型输入,关联到控制器 IO 输出,保持寄存器可以按位操作
    bRecv0:=gvl.wHoldRegister[1].0;//4x0001.0
    bRecv1:=gvl.wHoldRegister[1].1;//4x0001.1
    bRecv2:=gvl.wHoldRegister[1].2;//4x0001.2
    bRecv3:=gvl.wHoldRegister[1].3;//4x0001.3

//整数输出到主站
    gvl.wInputRegister[0]:=uiOutput;

//浮点数输出到 modbus 主站
//浮点数（小数）占用 2 个 Modbus 字地址，但是不同的主站高低字节的顺序不同
//因此需要读取 modbus 重新合成浮点数
    Write_LREAL_to_2_Register1(lInputValue:=ftest2 ,
    HighRegister=>gvl.wInputRegister[3] , LowRegister=>gvl.wInputRegister[2]);

//布尔型变量输出到主站
    gvl.wInputRegister[1].0:=bSend0;
    gvl.wInputRegister[1].1:=bSend1;
    gvl.wInputRegister[1].2:=bSend2;
    gvl.wInputRegister[1].3:=bSend3;
```

附录 3　Modbus TCP 使用说明

概述

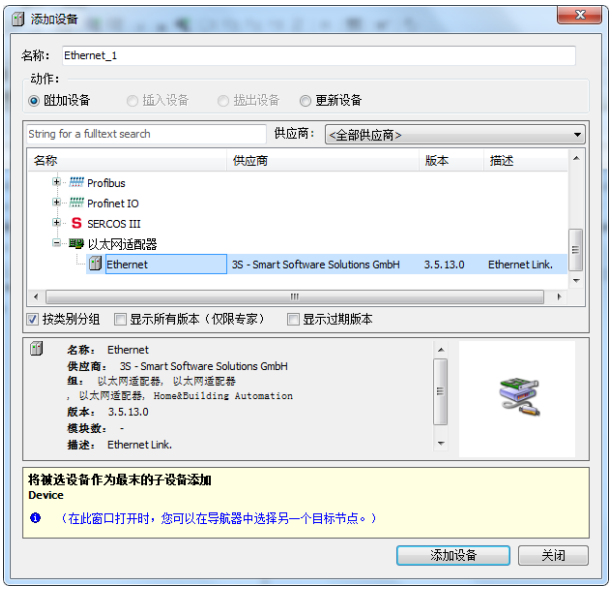
Modbus 是一项应用层报文传输协议，用于在通过不同类型的总线或网络连接的设备之间的客户机/服务器通信。广泛用于 PLC、控制器、触摸屏等设备间的数据交互。EAC 系列运动控制器具有一个 ENET 网口，可以独立的作为 modbus 主站/从站使用。

Modus TCP 主站及其实现

本节将详细说明在上位机软件 CODESYS 中如何配置 Modbus 主站，并建立与从站通讯的通道。在 CODESYS 中 Modbus TCP 主站及端口的实现以各种设备组件的形式存在。

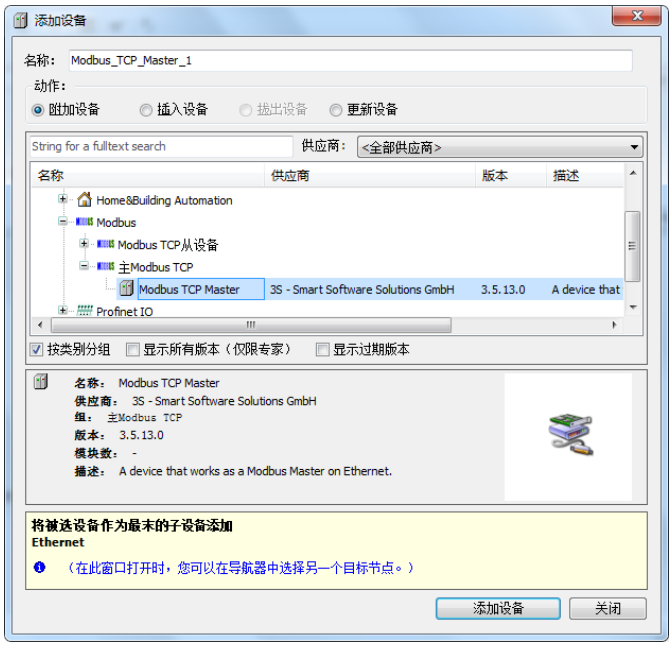
主站的实现步骤如下所示：

1. 添加网络适配器：在 CODESYS 程序左侧的设备树 Device 节点右击，选择添加设备。在弹出的对话框中根据需要选择现场总线--以太网适配器--以太网适配器端口下的 Ethernet 设备，点击添加设备。



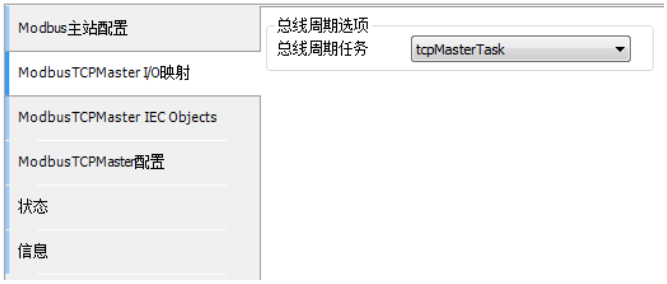
添加网络适配器

2. 添加 Modbus TCP 主站：在左侧的设备树端口设备节点 Ethernet 右击，选择添加设备。在弹出的对话框中根据需要选择现场总线——Modbus——主 Modbus TCP 下的 Modbus TCP Master 设备，点击添加设备。



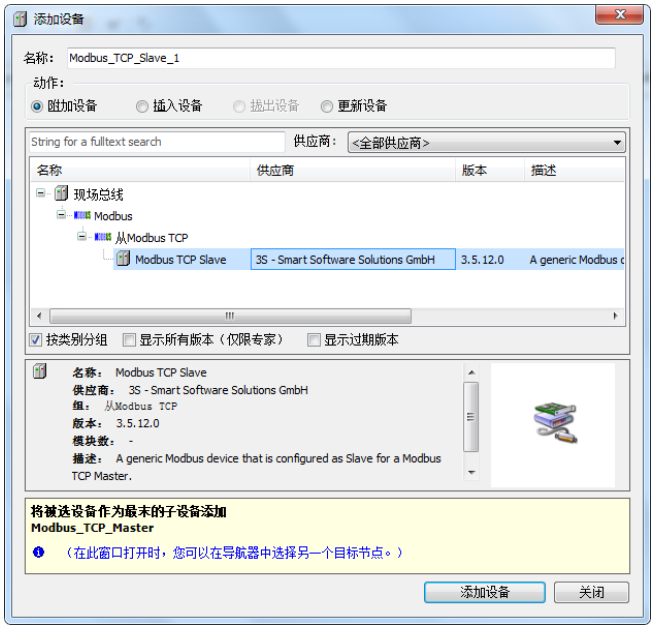
添加 Modbus TCP 主站设备

3. 设置主站设备：双击左侧设备树上的 Modbus 主站设备 modbus_TCP_Master,在右侧的 I/O 映射标签页中设置总线循环任务为主程序任务。



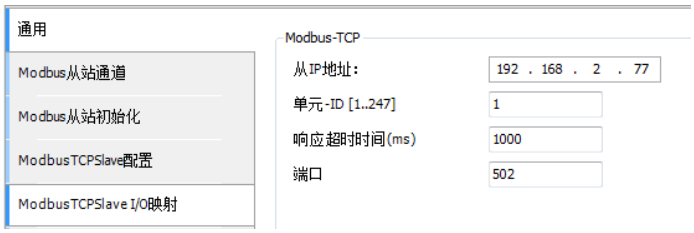
Modbus TCP 主站设备设置页

4. 添加 Modbus TCP 从站：在左侧的设备树端口设备节点 Modbus_TCP_Master 右击，选择添加设备。在弹出的对话框中根据需要选择现场总线——Modbus——从 Modbus TCP 下的 Modbus TCP Slave 设备，点击添加设备。



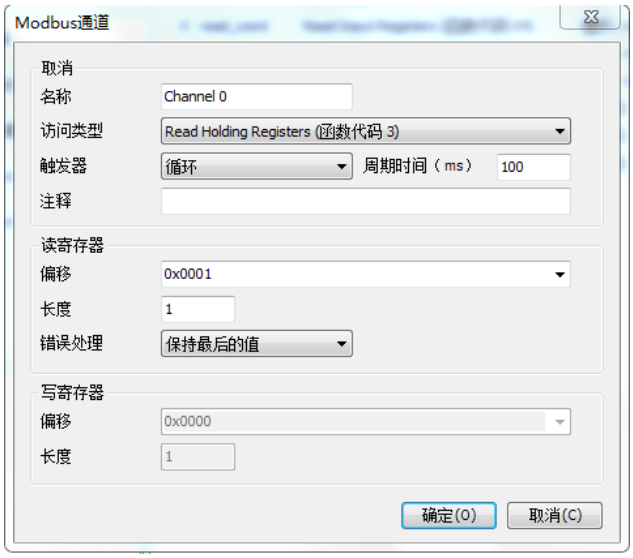
添加 Modbus TCP 从站设备

5. 配置 Modbus TCP 从站：双击左侧设备树上的 Modbus 从站设备，在右侧的通用标签页在设置从站 IP 地址、从站站号、单元-ID 和端口。

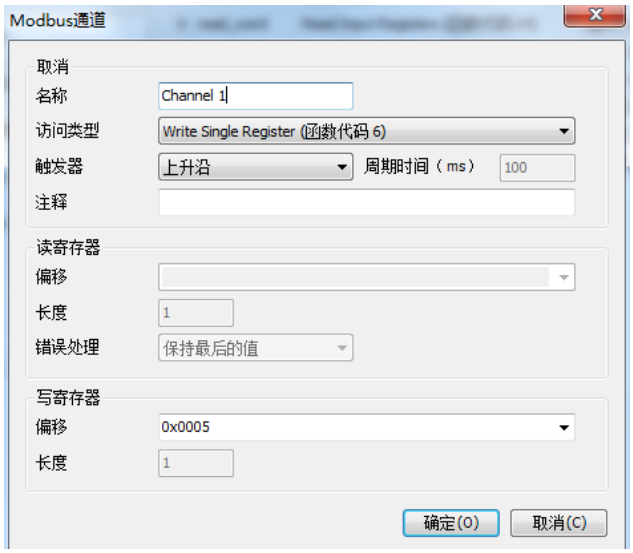


Modbus TCP 从站配置页

6. 建立从站通道：在 Modbus 从站通道标签页中点击“添加通道”按钮，在弹出的对话框中选择读取/写入方式、触发方式、从站目的地址和数据长度；变量触发方式分为周期和上升沿两种。周期为按照设定的周期时间定时进行写入/读取，大多用于对从站常用数值或监控状态的读取；上升沿方式则是当指定变量状态产生上升沿变化时（由 FALSE 变成 TRUE）触发对目的地址的写入和读取，通常用于对从站指令写入和非常用状态的读取。

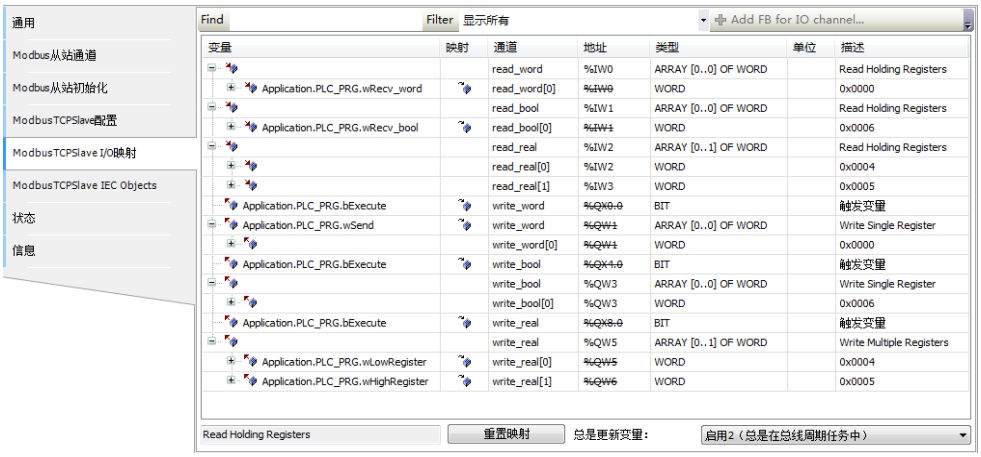


创建读从站通道



创建写从站通道

7. 映射从站变量：建立完从站通道后选择 ModbusTCPSlave I/O 映射选项卡，在这个页面中可以将建立的从站通道与程序中定义的变量进行映射关联，使读取到的数据直接赋值给程序中的变量，同时使程序中变量的数据写入到目的地址。最后将一直更显变量选项更改为总是在总线周期任务中，完成主站的创建。



从站变量映射页

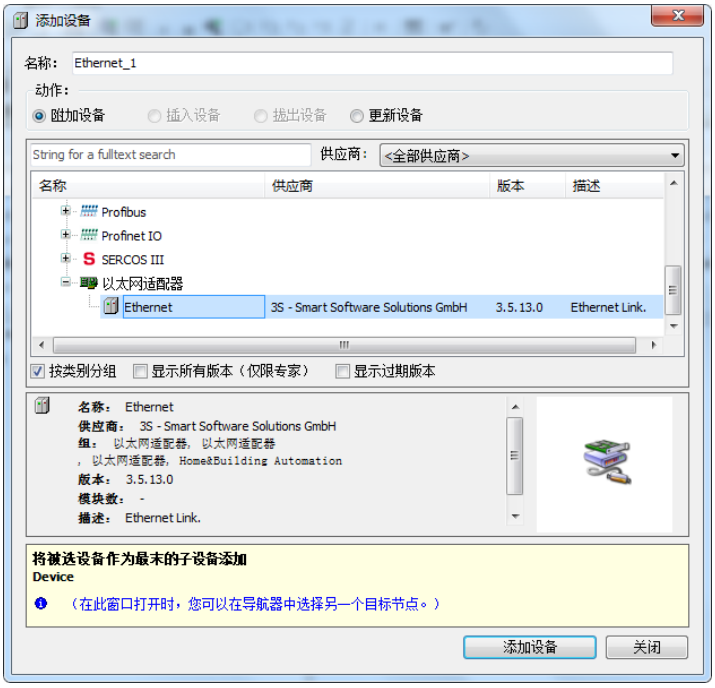
Modus TCP 从站及其实现

本节将详细说明在上位机软件 CODESYS 中如何配置 Modbus 从站。在 CODESYS 中 Modbus TCP 从站及端口的实现以各种设备组件的形式存在。

Modus TCP 从站中的寄存器分为保持寄存器和输入寄存器两类。保持寄存器的数值既可以被主站读取，也可以被主站改写，其地址从 4x0000 开始。输入寄存器只能被主站读取，不能被主站改写，其地址从 3x0000 开始。

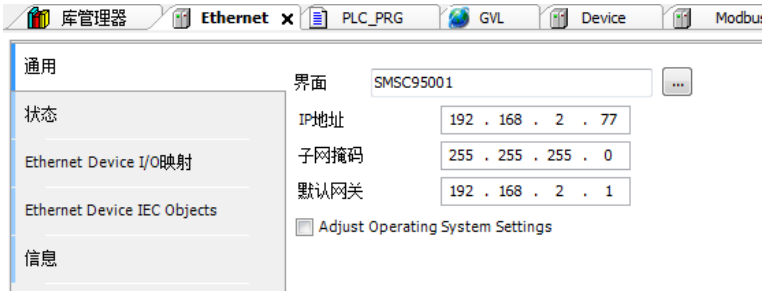
从站的实现步骤如下所示：

1. 添加网络适配器：在 CODESYS 程序左侧的设备树 Device 节点右击，选择添加设备。在弹出的对话框中根据需要选择现场总线--以太网适配器--以太网适配器端口下的 Ethernet 设备，点击添加设备。



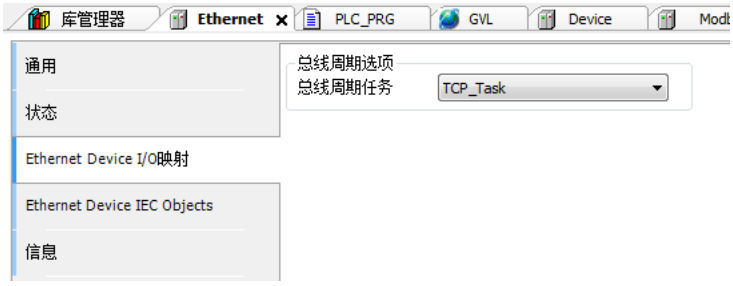
添加网络适配器

2. 设置网络适配器：双击左侧的设备树端口设备节点 Ethernet，在右侧的通用页中点击界面右侧的“...”按钮，选中扫描出的设备。



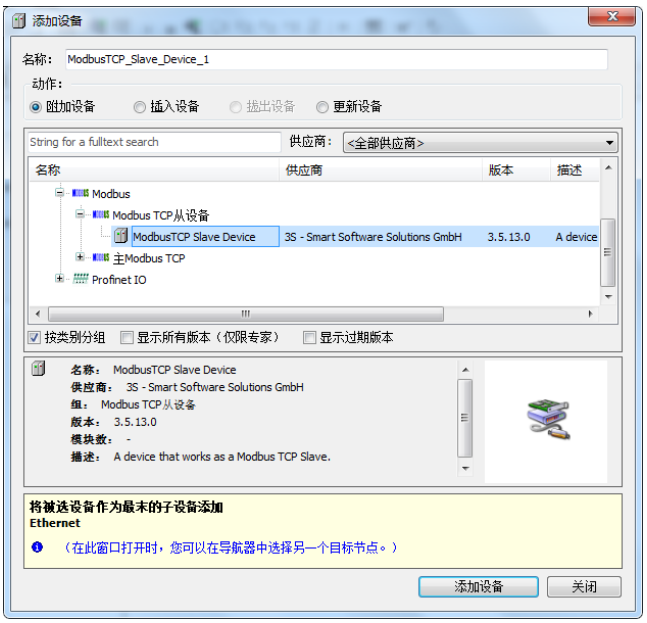
网络适配器设置页

3. Ethernet 设备 I/O 映射：选择 Ethernet Device I/O 映射选项卡，在这个页面中设置总线周期任务。



网络适配器总线周期任务设置

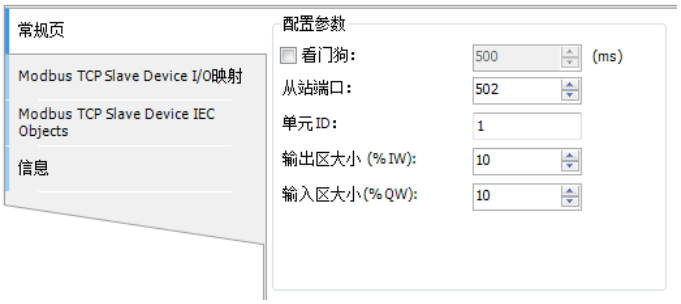
4. 添加 Modbus TCP 从站：在左侧的设备树端口设备节点 Ethernet 右击，选择添加设备。在弹出的对话框中根据需要选择现场总线--Modbus--Modbus TCP 从设备下的 ModbusTCP Slave Device 设备，点击添加设备。



添加 Modbus TCP 从站设备

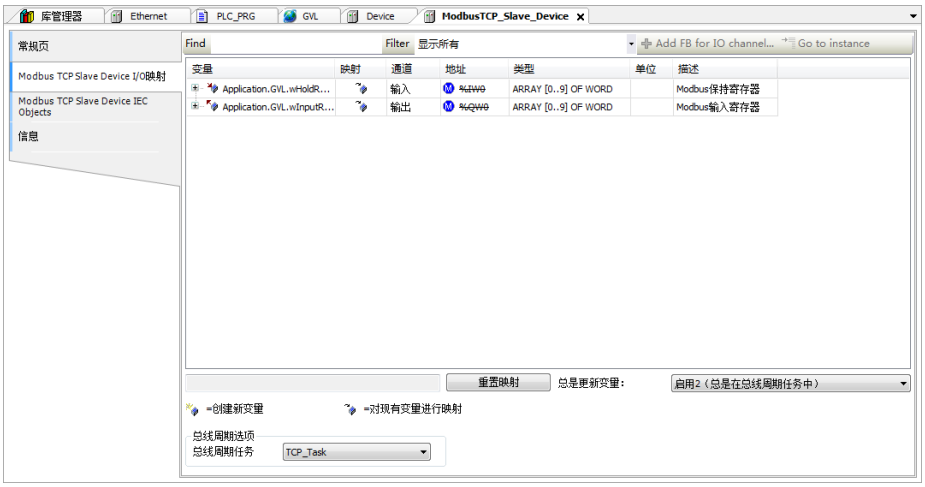
5. 设置 Modbus TCP 从站：双击左侧设备树上的 Modbus 从站设备，在右侧的常规页中设置单元

ID、输出区大小和输入区大小。



Modbus TCP 从站参数设置页

6. 映射从站变量：选择 Modbus TCP Salve Device I/O 映射选项卡，在这个页面中可以将保持寄存器和输入寄存器与程序中定义的变量进行映射关联，使读到的数据直接赋值给程序中的变量，同时使程序中的数据写入到目的地址。最后将总是更新变量选项更改为总是在总线周期任务中，完成从站的创建。



从站变量映射页

敬告用户：

感谢您选用我司产品，为保证您正确使用本产品及得到我司最佳售后服务，请认真阅读下述条款，并做好相关事宜。

只有具备一定的电气知识的操作人员才能够对本产品进行接线、上电操作；手册中示例程序仅供参考，不保证其实用性。

本公司致力于产品的不断改善和升级，手册提供资料如有变更，恕不另行通知，请自行访问本公司网站获取。

产品保修范围：按使用要求正常使用情况下，所产生的故障。

产品保修期限：本公司产品的保修期为自出厂之日起，十二个月以内。保修期实行长期技术服务。

非保修范围：任何违反使用要求的认为意外、自然灾害等原因导致的损坏，以及未经许可而擅自对产品拆卸、改装及修理的行为，视为自动放弃保修服务。

从中间商处购入产品：凡从经销代理商处购买产品的用户，在产品发生故障时，请与经销商、代理商联系。

免责条款：因下列原因造成的产品故障不在厂家 12 个月免费保修服务范围之内：

（1）、厂家不依照《产品手册》中所列程序进行正确的操作；

（2）、用户未经与厂家沟通自行修理产品或擅自改造产品；

（3）、因用户环境不良导致产品器件异常老化或引发故障；

（4）、因用户超过产品的标准范围使用产品；

（5）、由于地震、火灾、风水灾害、雷击、异常电压或其他自然灾害等不可抗力的原因造成的产品损坏；

（6）、因购买后由于人为摔落及运输导致硬件损坏。

责任：无论从合同、保修期、疏忽、民事侵权行为、严格的责任、或其他任何角度讲，EURA 和他的供货商及分销商都不承担以下由于设备所造成的特殊的、间接的、继发的损失责任。其中包括但不仅仅局限于利润和收入的损失，使用供货设备和相关设备的损失，资金的花费，代用设备的花费，工具费和服务费，停机时间的花费，延误，及购买者的客户或任何第三方的损失。另外，除非用户能够提供有力的证据，否则公司及它的供货商将不对某些指控如：因使用不合格原材料、错误设计、或不规范生产所引发的问题责任。

解释权归欧瑞传动电气股份有限公司。

如果您对 EURA 的产品还有疑问，请与 EURA 公司或其办事处联系。技术数据、信息、规范均为出版时的最新资料，EURA 公司保留部事先通知而更改的权利，并对由此造成的损失不承担任何责任。解释权归 EURA 公司。

20230725

欧瑞传动电气股份有限公司
EURA DRIVES ELECTRIC CO.,LTD
24小时服务热线： 4006-866-333
公司网址： www.euradrives.com