HCFa
禾川股份

Q系列可编程逻辑控制器

软件手册

自动化中心（ATC）
编号： ATC/MQ1S2011
2020 年 5 月 11 日

前言

进入二十一世纪以来，信息技术快速发展，传统工业控制模式已经无法满足用户对于控制系统小型化、智能化和网络化的需求，深入研究基于新体系架构、新技术、高性能、低成本、开放式多轴运动控制器，既有市场需求，又有良好的技术背景环境。

早期 PLC 技术主要由国外几家大公司掌握资源，只能对少数指定几种 PLC 进行编程控制，使用范围有限，价格高，用户的后期维护也十分困难，由于 PLC 产品不断更新，产品的生命周期十分有限。因此现在的趋势是使用以计算机为基础的编程设备，在这样的背景下，禾川选择使用 CODESYS 作为新一代中型控制器 Q 系列可编程逻辑控制器的软件平台，CODESYS 作为功能强大的工业自动化软件开发平台为国内广大的自动化公司开发自己的软硬件产品提供了便捷的技术支持和广阔的平台，基于 CODESYS 的程序开发系统功能强大，它可以使用丰富的编程语言（IEC61131-3）对应用于不同行业和领域的项目进行编程，也支持和高级语言进行数据交互，将计算机领域积累的编程思路、程序架构和复杂算法引入工业控制领域，大大丰富了工业控制的灵活性和可能性，同时 CODESYS 也提供了数据监控、采集和分析等许多功能；在硬件接口上，Q 系列可编程逻辑控制器为用户提供了更多的可能性，控制器配置了标准的以太网、EtherCAT、CANOPEN、RS232 以实现联网以及和多种通用工业现场总线协议之间的通讯，并且搭载了高速 IO 以便用户进行高速脉冲信号的采集和输出，USB3.0 则使得控制器和外部之间的数据存储和转移更为简单。

我国是一个制造业大国，但控制技术的水平还不是很高，跟欧美和日本等发达国家之间的差距仍然十分巨大，严重制约了我国制造业水平的提高，用户只能用更高的成本使用国外成熟的产品和技术，随着国内越来越多的自动化公司对于嵌入式控制器研发的的投入和硬件制约的解除，相信会有更多类似 Q 系列可编程逻辑控制器的产品和禾川这样的公司不断出现，推动实现工业控制的自动化和信息化。

读者对象

禾川 Q 系列控制器的用户，可以按照本书进行硬件配置，软件编程和调试，需要用户具备一定的计算机和自动化基础。

主要内容

第一章介绍了 CODESYS 及其编程规范 IEC61131-3

第二章介绍了 CODESYS 软件安装及界面

第三章介绍了 CODESYS 软件架构和 PLC 应用的配置

第四章介绍了 HMI 项目工具的介绍及其应用

第五章介绍了如何创建新的 CODESYS 项目包括外部硬件的添加、通讯及程序的下载、监控和调试

第六章介绍了如何实现多种通讯，包括基于以太网的 TCPIP\ Modbus TCP\ EtherCAT 以及 Canopen

术语和缩略语

图表 1.1-1 术语和缩略语

术语/缩略语	说明/全称
Q1	禾川中型控制器
POU	Program Organization Unit，程序组织单元
PLC	Programmable Logic Controller，可编程控制器
DUT	Data Unit Type，数据单元类型

第1章 系统概述

1.1 文档说明

本文档暂时仅针对 Q1 中型控制器的上位编程软件 CODESYS V3.5 SP13，对于其他 CODESYS 软件版本的兼容性不做保证，请安装正确的 CODESYS 软件版本后按照本书操作，同时根据章节内容提供对应例程以供用户参考，帮助理解课程内容。书将跟随 Q1 控制器内软件版本的更新做不定时更新，版本更新时间和内容将会在本章节 1.4 做详细说明，并可以在禾川官网下载最新的软件书和配套例程，书编写难免出现不全面及错误，对于本书的意见和建议请直接发送邮件至：
wujingwen@hcfa.cn

1.2 软 PLC 解决方案

PLC 从硬件结构上来区分，可以分成硬 PLC 和软 PLC。硬 PLC，也就是传统 PLC，是由硬件或者一块专用的 ASIC 芯片来实现 PLC 指令的执行。而软 PLC，也称为软逻辑（SoftLogic），是使用个人计算机（PC）或者嵌入式控制器作为硬件支撑平台，利用软件实现硬件 PLC 的功能。也可以说是讲 PLC 的控制功能封装在软件内，运行于 PC 或者嵌入式控制器的环境中。

利用工业控制计算机（IPC，简称工控机）或嵌入式控制器的硬件和软件资源，用软件实现全部硬件 PLC 能实现的功能，这就是软 PLC 技术。随着计算机技术的发展，计算机标准化的通讯协议和成熟的局域网技术使得组网十分简便，还可以通过 Internet 与外界进行数据交互和共享，也使得硬 PLC 通用性和兼容性差的弊端愈来愈明显。软 PLC 也使得物联网技术得以更为便捷的应用于工业现场，符合工业 4.0 智慧工厂的发展理念。Q1 中型控制器就是禾川推出的新一代软 PLC 产品，是未来工业 PLC 发展的趋势，也是禾川迈向工业 4.0 的扎实一步。

软 PLC 综合了计算机和 PLC 的开关量控制、模拟量控制、数学运算、数值处理、网络通讯、PID 调节等功能，通过一个多任务控制内核，提供强大的指令集、快速而准确的扫描周期、可靠的操作和可连接各种 IO 系统及网络的开放式架构。相比硬 PLC，具有如下优点：

① 开放的体系架构，兼容各类 IO 和总线端口
② 可维护性强，遵循国际工业标准，开发更为规范
③ 充分利用 PC 资源，不断发展的计算机技术为软 PLC 提供了更优质的软硬件平台
④ 降低成本，统一的国际标准，允许更多的开发商开发自己的产品，形成良性竞争

1.3 CODESYS 简介

CODESYS 全称是 Controller Development System 由总部位于德国巴伐利亚的 3S（Smart Software Solution GmbH）公司开发。

CODESYS 是可编程逻辑控制器 PLC 的完整开发环境，支持 IEC 编程语言，系统的编辑器和调试器的功能是建立在高级编程语言的基础上（如 visual C++）。目前 CODESYS 在全球范围内得到了广泛的应用，包括 ABB、BECKHOFF、Bachmann、EPEC、Rexroth 等供应商都是用 CODESYS 平台开发自己的编程软件。

1.4 IEC61131-3 编程规范

1993 年 3 月由国际电工委员会（International Electro-technical Commission，IEC）正式颁布可编程控制器的国际标准 IEC 1131（在 1131 前面添加 6 后作为国际标准的编号，即 IEC 61131）。IEC61131 标准将信息技术领域的先进思想和技术（如软件工程、结构化编程、模块化编程、面向对象的思想及网络通讯技术等）引入工业控制领域，弥补了传统 PLC、DCS 等控制系统的不足（如开放性差、兼容性差、应用软件可维护性差及可再用性差等）。

IEC61131 是第一个关于 PLC 编程技术的国际标准，其中 IEC61131-3 是建立统一 PLC 编程语言的基础，是实现软 PLC 技术的重要条件。IEC 编程标准支持两大类总共六种编程语言。两大类分别为文本化编程语言和图形化编程语言，其中文本化编程语言包括指令表（Instruction List，IL）和结构化文本（Structured Text，ST），图形化编程语言包括梯形图（Ladder Diagram，LD），功能块图（Function Block Diagram，FBD），顺序功能图（Sequence Function Chart，SFC）和连续功能块图（Continuous Function Chart，CFC）。多种编程语言之间可以共同服务于同一个项目，优化程序设计，统一的编程语言也减少了用户对于单一硬件供应商的依赖，增加了程序的可读性和安全性，减少了后期维护成本。

1.5 文档更新和发布状态

图表 1.5-1 文档更新和发布状态

发布时间	手册编号	更新内容	发布状态
2019 年 5 月 28 日	ATC/MQS01-1.0	第一版	正式发布
2019 年 8 月 27 日	ATC/MQS01-1.1	第二版	正式发布
2020 年 5 月 11 日	ATC/MQ1S20111	第三版	正式发布

第2章 软件安装和界面介绍

2.1 安装环境要求

CODESYS V3.5 是目前 Q1 控制器的上位编程软件，支持编程、调试和硬件配置。由于 CODESYS V3.5 软件比较复杂，需要处理的数据也比较多，因此对于安装 CODESYS 的 PC 平台的硬件及系统环境都有一定要求，要求的最低配置和推荐配置见图表 2.1-1

图表 2.1-1 安装环境要求

描述	最低配置	推荐配置
操作系统	Windows 2000 (Windows Vista/Windows 7/8/10)	Windows 7/8/10(32/64bit)
内存	512M	4GB
硬盘空间	200M	2GB
处理器	Pentium V, Centrino>1.8GHz, Pentium M>1.0GHz	Pentium V, Centrino>1.8GHz, Pentium M>1.5GHz

2.2 安装步骤

安装时请关闭杀毒软件或者添加 CODESYS 软件到白名单以免安装过程中部分功能无法正常实现或者被删除等造成软件无法打开或功能不全。

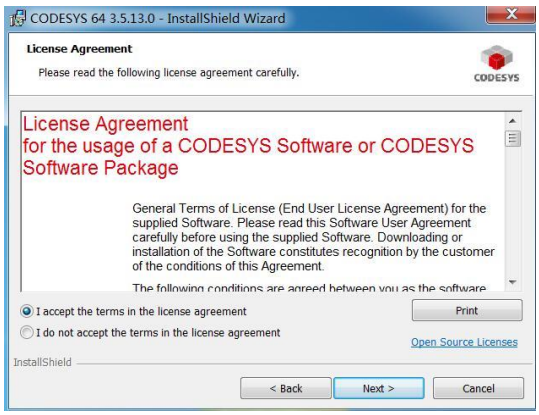
软件安装包可以在 CODESYS 官方网站进行下载，下载链接是：

http://www.CODESYS.cn/list-DOWNLOAD-2.html

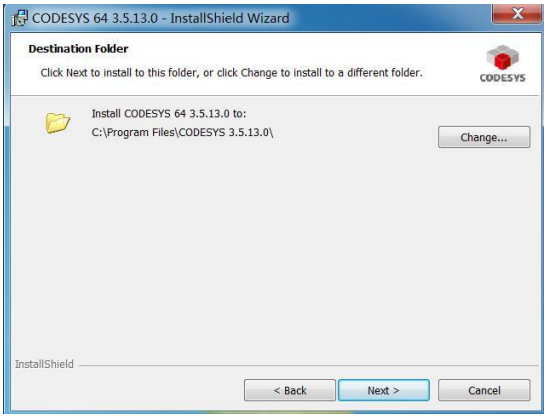
下载完整的 CODESYS V3.5 SP13 的安装包后，直接双击运行软件安装包即可进入安装引导界面。在软件正式安装之前，程序会对安装环境进行检测，需要确保有 Microsoft Visual C++和 Microsoft .NET Framework4.6（或以上版本）的安装环境，否则程序会自动安装这部分软件，之后会弹出界面需要用户同意 3S 公司的软件使用规范，如图 2.2-2，同意并进入下一步。程序默认的安装路径为“C:\Program File\CODESYS 3.5.13.0\”，如果需要修改，可以通过右侧“Change”按钮重新选择安装路径。进入下一个页面，用户可以选择安装全部或者进行自定义安装，初次安装建议选择“Complete”安装全部；用户也可以根据实际需求进行自定义安装，参考 2.2-5，按照引导进入下一个页面，点击“Install”进行安装。

图表 2.2-1

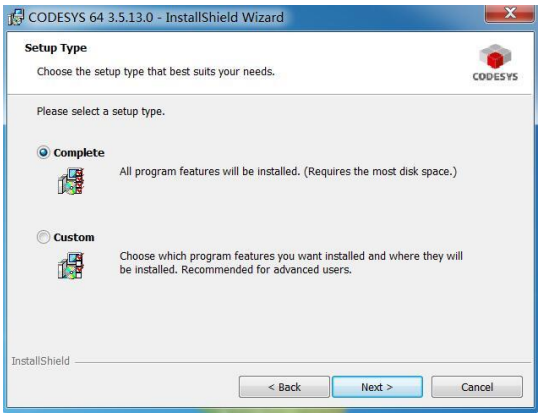
图表 2.2-2



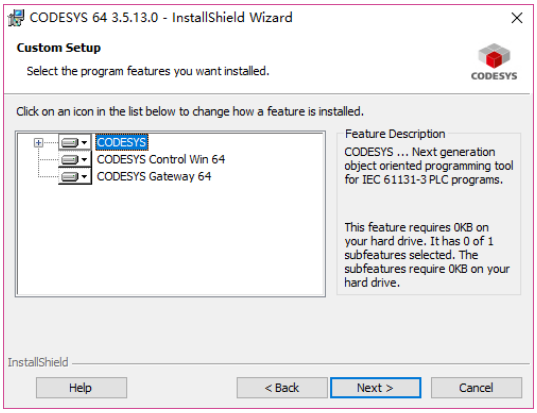
图表 2.2-3



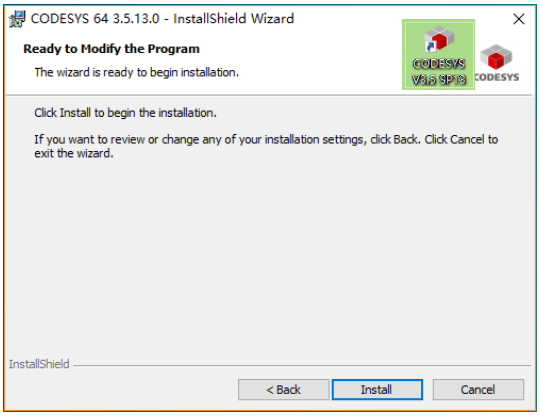
图表 2.2-4



图表 2.2-5



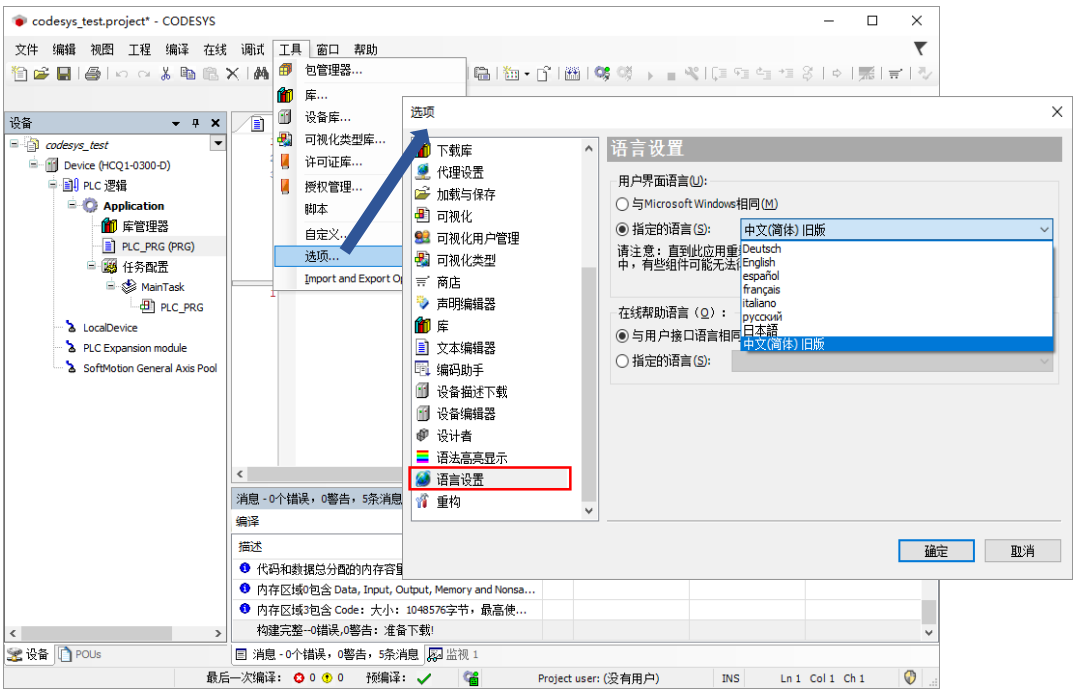
图表 2.2-6



完成以上步骤了，等待软件安装完成，完成后可以在桌面上找到 CODESYS 软件图标 ，双击进入进行编辑。

安装完成后，用户如果需要切换 CODESYS 软件界面的语言显示，可以在菜单栏工具→选项→语言设置→用户界面语言中，通过下拉选项卡进行界面语言切换，点击确认并重启 CODESYS 软件应用设置

图表 2.2-7



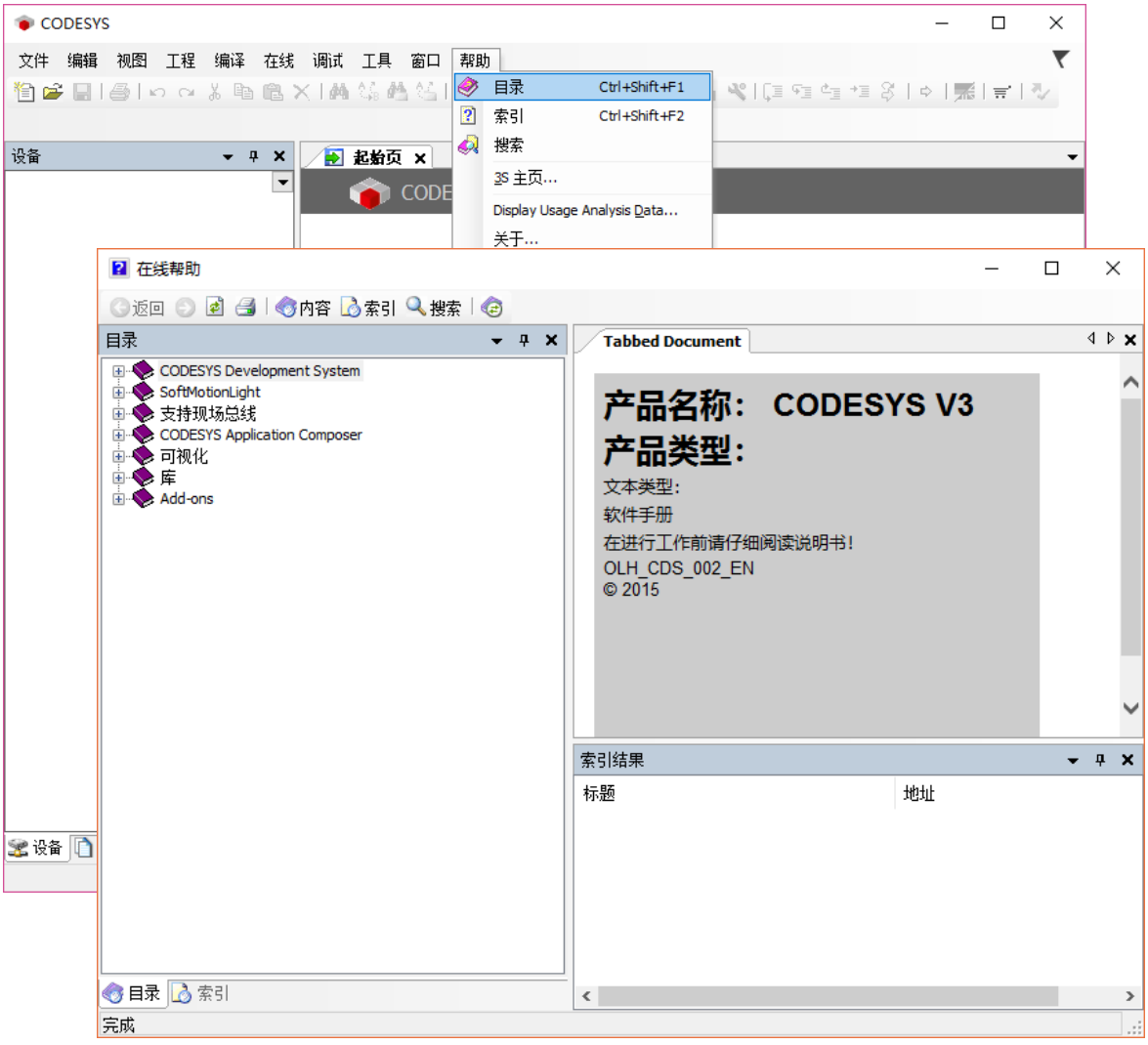
2.3 版本管理

CODESYS 暂时不支持软件之间的多版本切换，高版本会自动向下兼容低版本。软件内的组件支持同时安装多个版本，并且可以组合使用，程序编译器也可以安装与使用多个版本，并且无须更新整个版本就可以新增独立的功能。

2.4 帮助系统

CODESYS 完整版本安装完成后默认会安装对于该软件的帮助说明。用户打开软件后可以在菜单栏找到“帮助”，单击“目录”即可打开在线帮助。用户可以根据索引或者搜索关键字快速查找。

图表 2.4-1

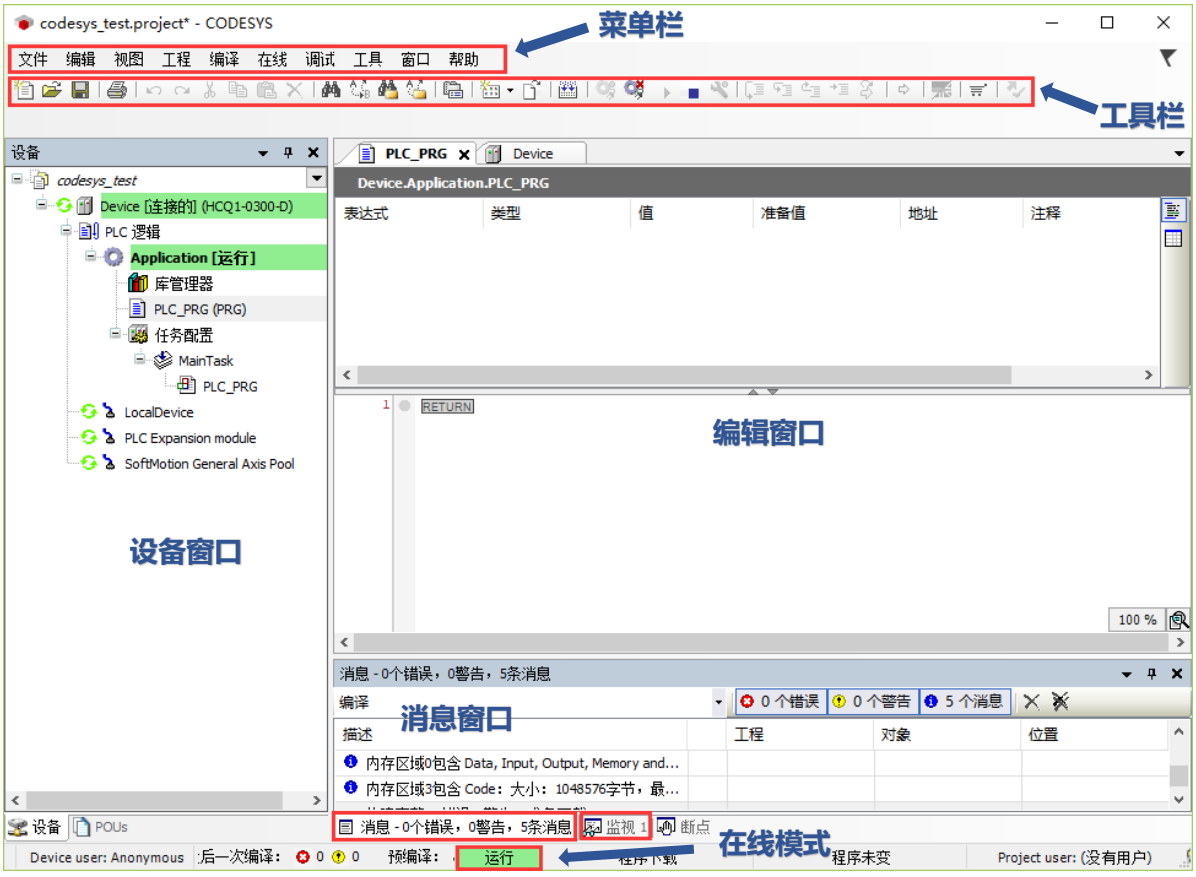


2.5 CODESYS 界面介绍

启动 CODESYS 软件后，可以看到 CODESYS V3.5 SP13 的开发界面，标准组件主要有菜单栏、工具栏、编辑窗口、设备窗口、监视窗口、消息窗口和在线模式等。接下去对用户开发环境进行详细介绍。

在 CODESYS 软件界面中，所有窗口及视图都是不固定的，用户可以根据自己的习惯将窗口和视图通过鼠标拖拽的方式移动到目标位置，对窗口和视图按照喜好重新排列。

图表 2.5-1



消息窗口下的选项卡可以进行切换进入监视窗口，在监视窗口中可以查看程序中的任意表达式

图表 2.5-2

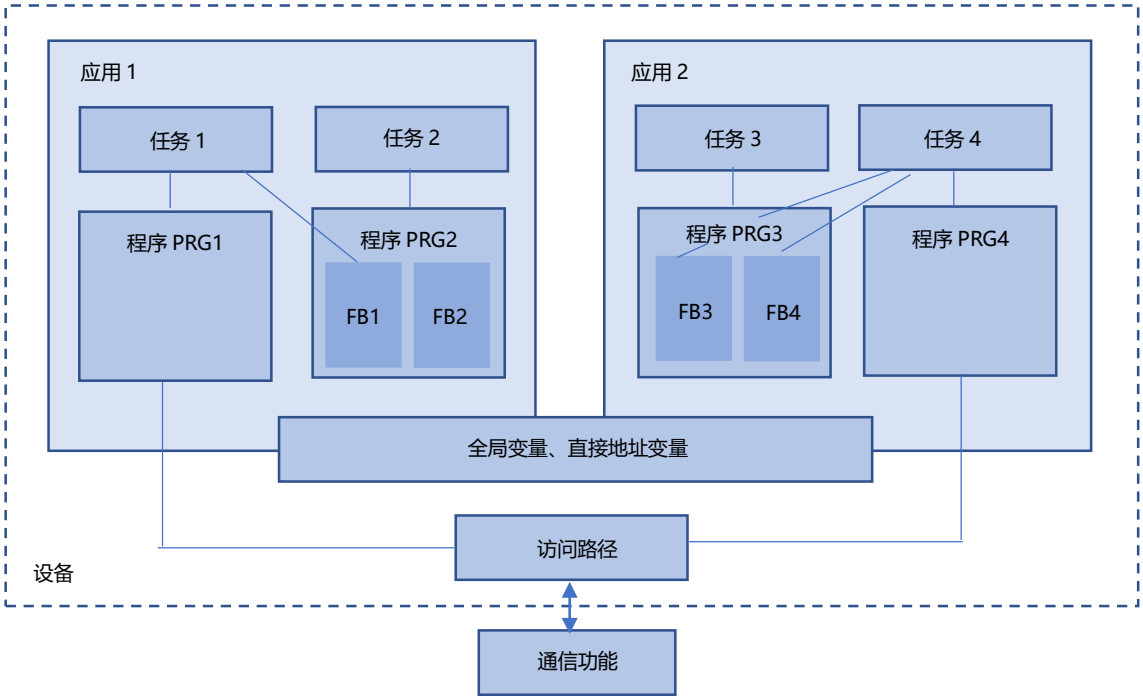
在线模式会显示当前程序运行状态

第3章 CODESYS 软件模型

3.1 CODESYS 软件架构

CODESYS 的软件模型用分层架构表示，描述了基本功能单元及其相互关系，其中包括：设备、应用、任务、全局变量、访问路径和应用对象。参考图表 3.1-1（以两个应用作为示例），软件模型描述了一台可编程口控制器如何实现多个独立程序的同时运行。实现对程序执行的完全控制等。

图表 3.1-1

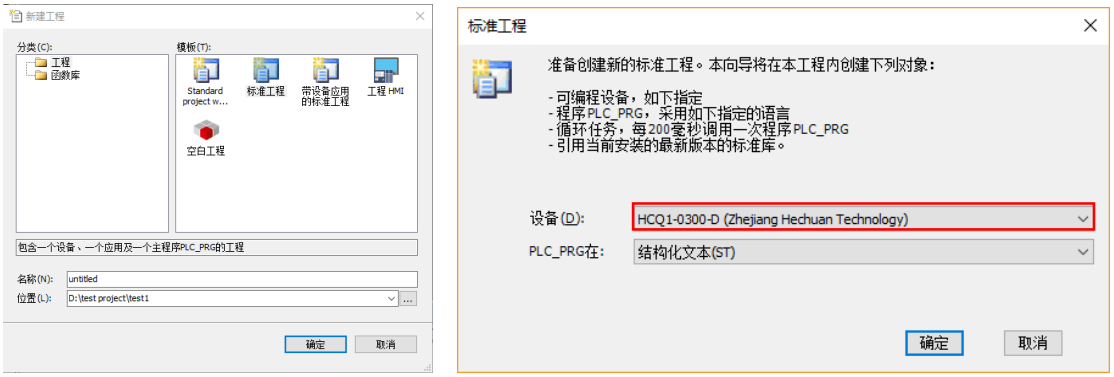


在 PLC 系统中，设备为应用提供软硬件环境，并且将多个“应用”结合成组，为应用提供数据交互的手段。在一个设备中，可以有一个或者多个应用，应用为程序和 PLC 物理 IO 之间提供了一个接口。应用会被分配在一个 PLC 的 CPU 单元内，应用的主要成员包括全局变量、任务和程序组织单元（Program Organization Unit，POU）。访问路径提供不同应用之间交换数据和信息的方法，每一个应用内的变量也可以通过这种方式建立和其他远程设备之间的通讯。

3.2 设备管理器

用户在新建工程时，系统会自动弹出“新建工程”对话框，在“模板”列表框中，用户可以选择新建一个空工程或标准工程，在选择标准工程时，需要选择实际连接在“设备”中的硬件。

图表 3.2-1



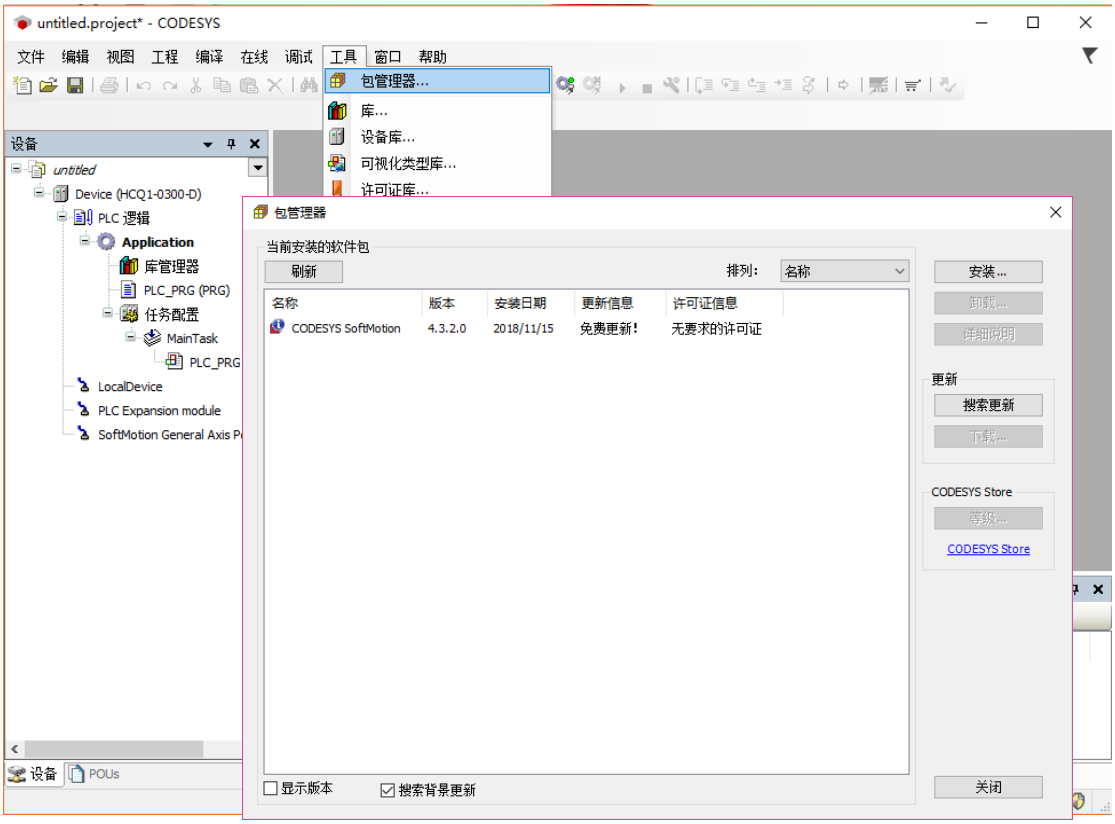
单击“确认”即可完成项目的创建，需要注意的是，在 CODESYS 中添加的设备和软件功能需要用户进行安装和更新。

3.2.1 包管理器

所有的软件包，都需要在工具→包管理器中进行安装，在包管理器中用户也可以对已安装的软件包进行删除和更新。

针对不同的硬件设备，需要不同的硬件设备配置文件，文件一般包含：代码生成器、内存管理、PLC 功能、IO 模块等配置信息，另外还包含了库文件、设备描述文件、网管驱动程序、错误代码的 ini-files 等等

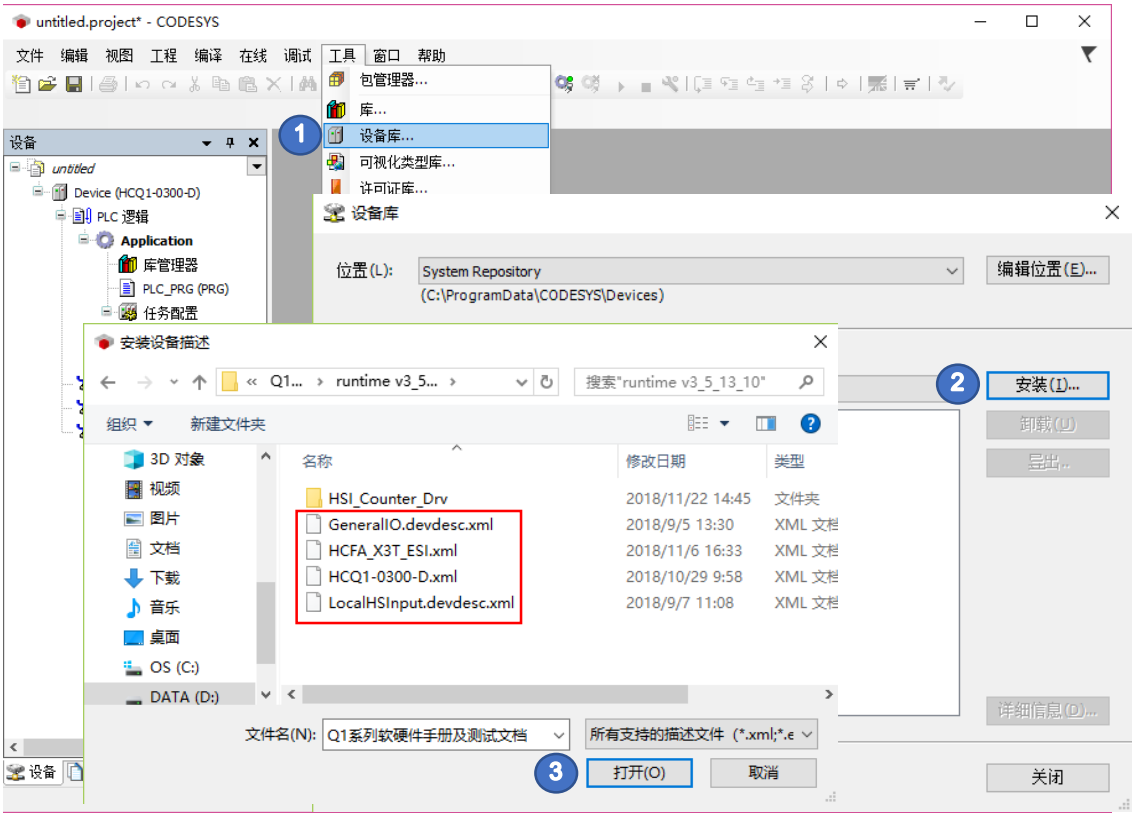
图表 3.2-2



3.2.2 设备库

“设备库”选项负责安装、卸载设备，以及供用户查看设备信息，从属设备只有在设备库添加完成后，用户才能在“添加设备”选项卡中找到对应的从属设备，否则不能使用该从属设备。选择工具→设备库，在设备库的对话框中选择“安装”可以导入相应的设备文件，可添加的设备有供应商的 PLC、SoftMotion 运动控制设备（编码器、驱动器等）、现场总线及专用接口等设备 。

图表 3.2-3



安装过程中引用的设备描述文件和所有的附加文件将会复制到一个内部地址中。如果供应商修改了设备描述文件，将不会影响已安装的设备，对于更新了软件版本的设备，建议用户改变设备描述文件内部版本号（也可以选择卸载现有设备重新安装更新后设备描述文件）。

> 注意：
> 内部设备库是通过一定规范安装后引用到 CODESYS 内部地址中，不支持用户手动修改内部设备库当中的任意文件，也不支持用户从内部设备库复制或者复制文件到内部设备库，必须使用“设备库”对话框来添加或者删除设备。

3.3 PLC 应用

CODESYS 应用（Application）的对象包括库管理器、程序组织单元（POU）、任务配置、全局变量和采样追踪等，单个设备中可以添加多个应用。

3.3.1 库文件管理

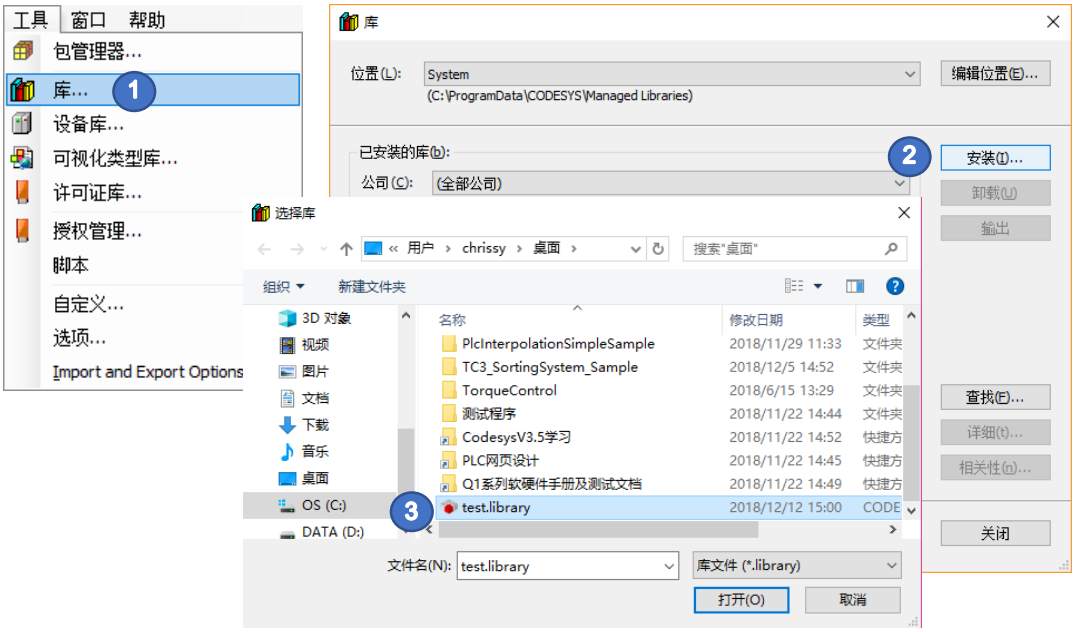
库文件用于存放在程序中多次使用的程序组织单元（POU）。这些 POU 可以是已经创建好供用户使用的库，也可以是用户自定义的库。库文件除了是函数、功能块和程序的集合，其中还包含一些特殊定义的结构体、枚举类型等。支持创建和调用后缀为“.library”和“.complied-library”的库文件，其中“.library”后缀的是标准的库文件，用户可以直接打开库文件对其进行编辑；“.complied-library”是编译后的库文件，用户无法直接打开该后缀的库文件，进而查看源代码和进行修改，但可以正常调用库文件中包含的所有 POU。

库文件的安装

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首先对库文件进行安装和调用。CODESYS V3.5 版本支持“lib”（CODESYS V2 版本的库文件标准，用 V3 打开会自动转换成“.library”文件，做到向下兼容）“library”和“complied-library”。

选择菜单栏“工具”下的“库”，单击打开，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

图表 3.3-1

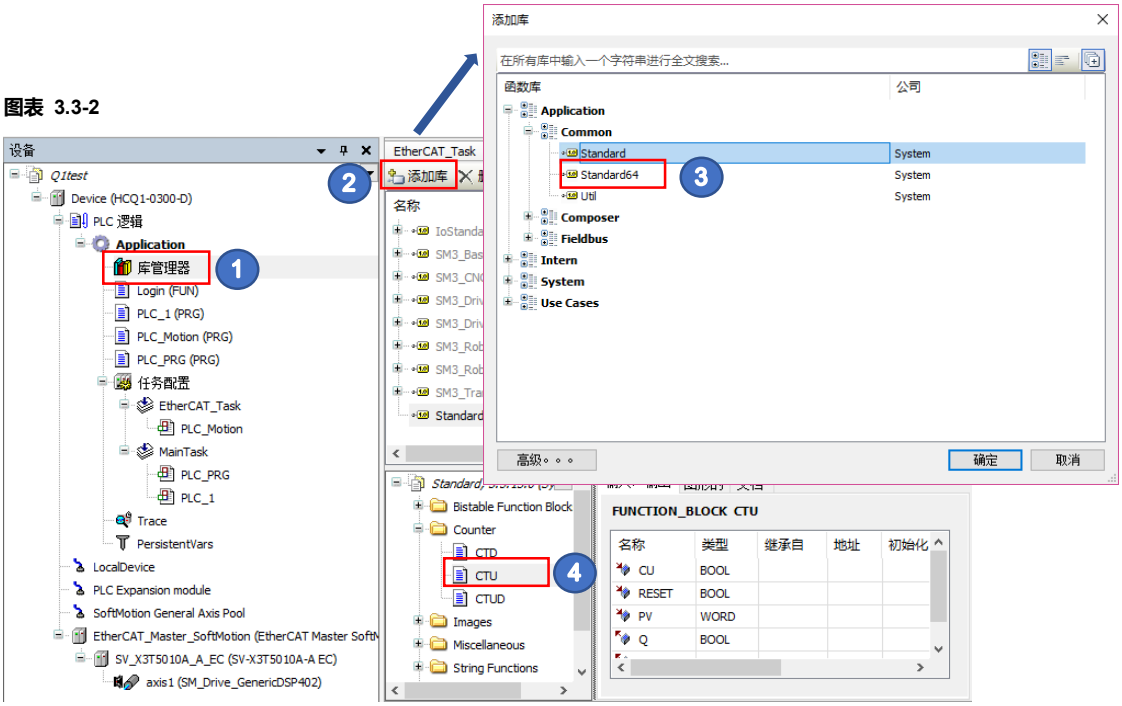


库文件的调用

安装完成后的库会自动添加到 CODESYS 系统资源库中，用户不可以自行将库文件添加到目标路径下，也不可以对目标路径下其他的库文件进行编辑，库文件的安装、调用和卸载都需要通过系统提供的库管理器实现。

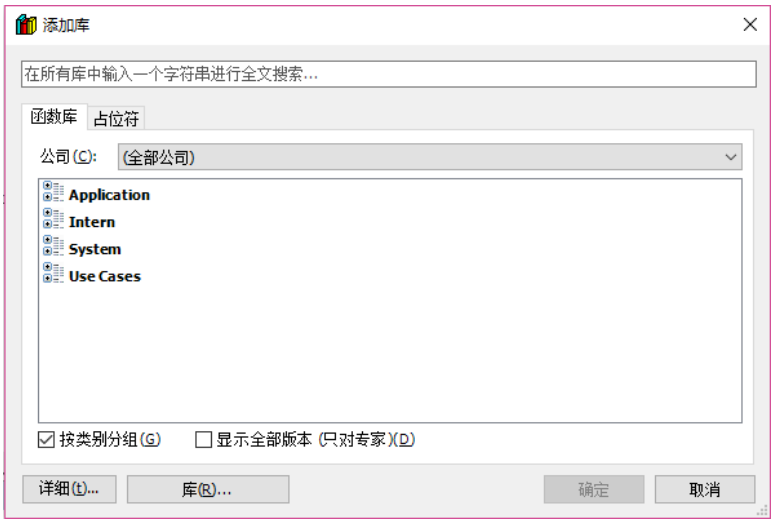
双击“库管理器”，进入“库管理器”的配置页面，在左上角单击“添加库”可以在弹出页面中选择需要添加的库，点击“确定”进行调用，调用成功后，可以在库管理器页面中查看库对应的函数和功能块及其简单说明。

图表 3.3-2



用户也可以单击 “高级”选项后，根据供应商名、库文件功能及版本号进行选择，同样通过占位符和已知名称的功能块（需包含在已安装库中）搜索后快速调用库。

图表 3.3-3



库文件的创建

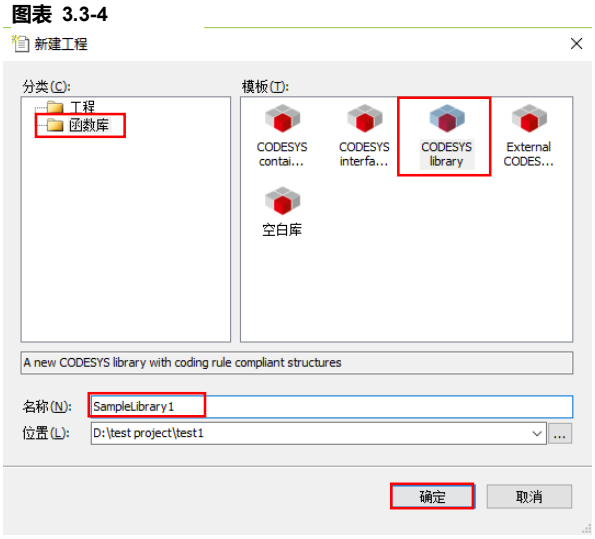
除了供应商提供的库文件外，用户也可以根据自己的需求把常用的函数和功能块封装起来，创建自己的库文件，以便分享和应用于其他项目，创建库文件的步骤如下：

- 创建空白库文件

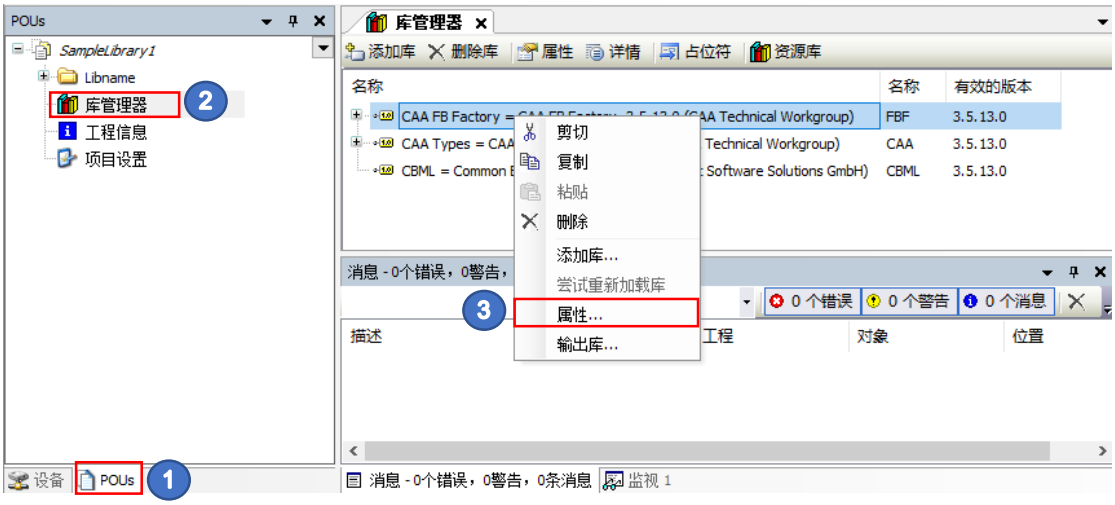
使用菜单“文件”→“新建工程”→“函数库”→“CODESYS library”，在“名称”文本框中输入库的名称“SampleLibrary1”，选择库文件保存的位置后，单击“确定”自动生成一个新的库文件。

创建库工程时，如果引用了其他库，可以在每个被引用库的“属性”栏中定义它被引用后的行为参考图表 3.3-6

图表 3.3-4



图表 3.3-5

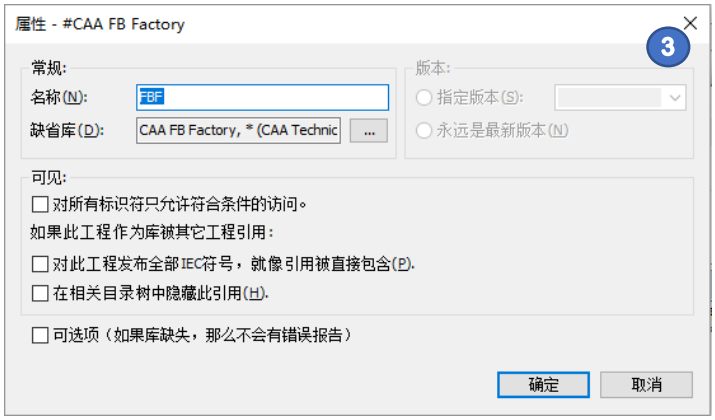


有以下几点需要注意：

① 在库管理器中被调用的库一般列在它的“父”库下面，可以显示或者隐藏

② 如果一个库被其他库引用而被包含进一个工程中时，它的行为如何，这个“行为”包括版本处理、命名空间、可视性和访问属性，需要用户在引用库的属性对话框中设置，以后当该库被工程调用时就会按照这些设置进行处理

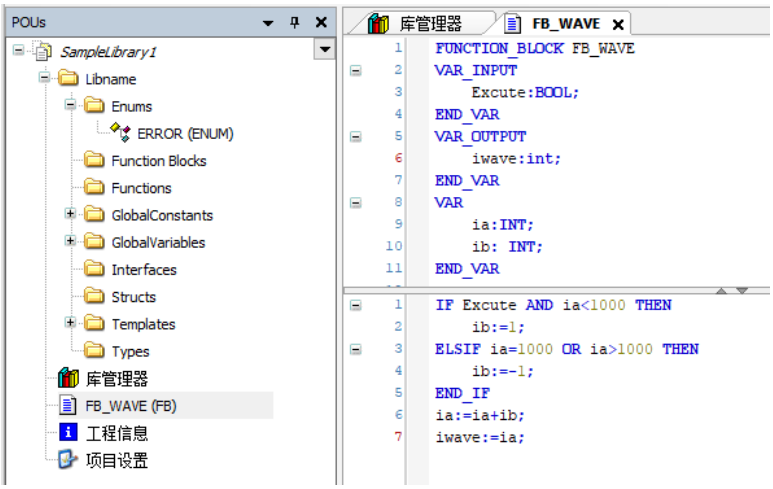
图表 3.3-6



- 创建 POU

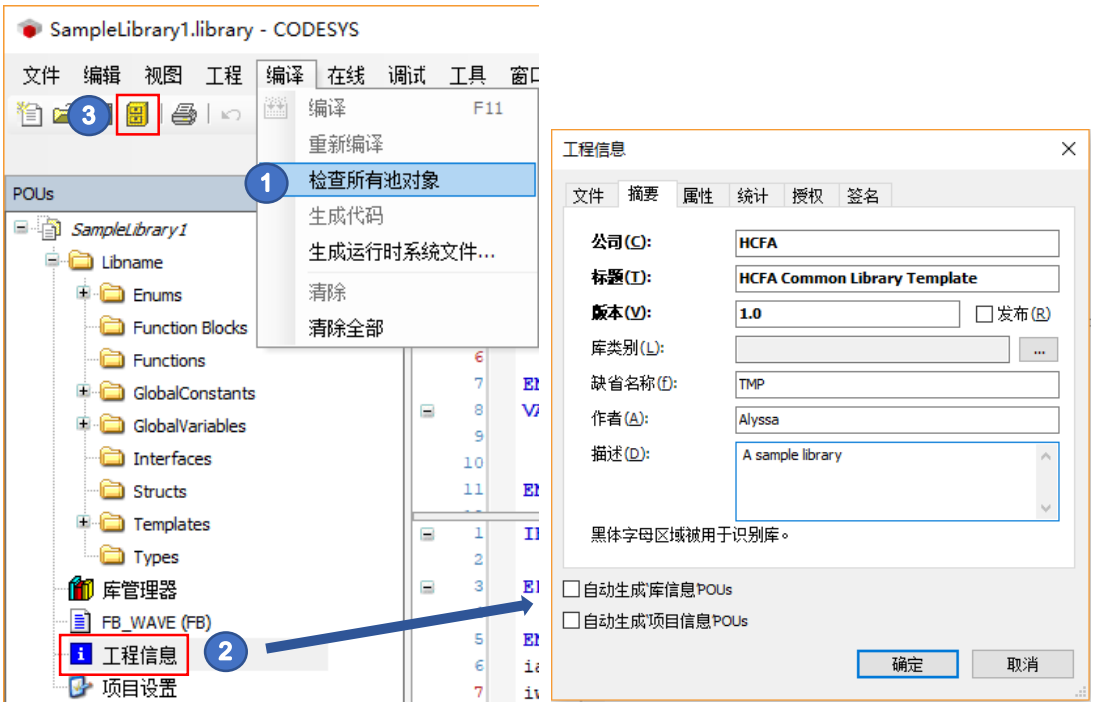
在库文件中可以创建自己的函数和功能块也可以编辑枚举、结构体、接口和函数全局变量等，右击“SampleLibrary”→“添加对象”→“POU”，选择新建“功能块（B）”作为例子，编程语言选择“ST”，编写示例程序如下：

图表 3.3-7



完成示例程序的编写需要对其进行编译检查及保存，依次选择“编译”→“检查所有池对象”；确定没有错误后，选择“工程信息”完善库文件的属性，可以编辑库文件所属公司、库文件标题、版本号、作者名和库的简要说明等，其中黑色加粗部分必须填写；最后，单击 保存工程，并装入库，即可完成对库文件的保存。

图表 3.3-8



在之后新建的项目调用此库文件，只需要通过“工具”→“库”安装库文件即可，对于使用不到的库也可以通过同样的方式选择“工具”→“库”，选中需要卸载的库文件之后选择卸载按钮即可。用户更新自定义库时需要注意编辑库文件的 CODESYS 版本的一致性，更新完成后，卸载旧版本的库，重新安装新版本就可以完成自定义库的更新。

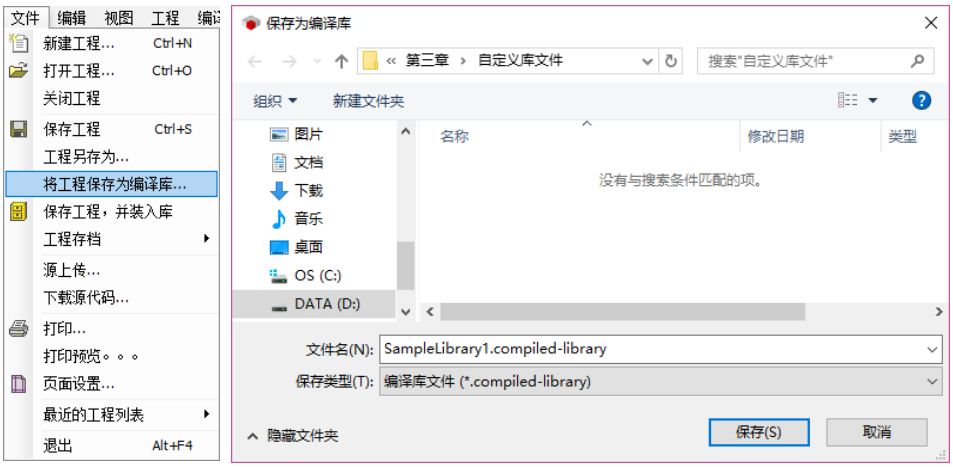
库文件加密

用户自定义库如果需要分享给客户，但不想开放库文件源代码，有以下两种处理方式：

- 保存库文件为编译后的库文件版本“complied-library”

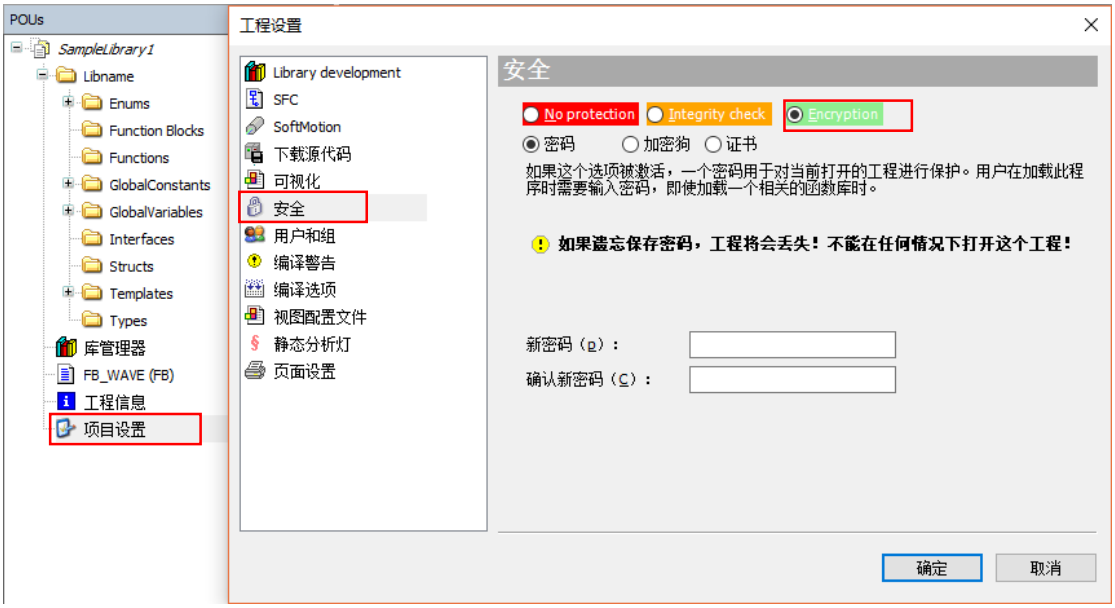
自定义库文件默认保存格式为“.library*”，可以打开并进行库文件的编辑，用户可以选择“文件”→“将工程保存成编译库”选择保存路径后点击“保存”就可以得到编译库文件，之后可以将这个文件分享给其他用户并保证代码不公开。

图表 3.3-9



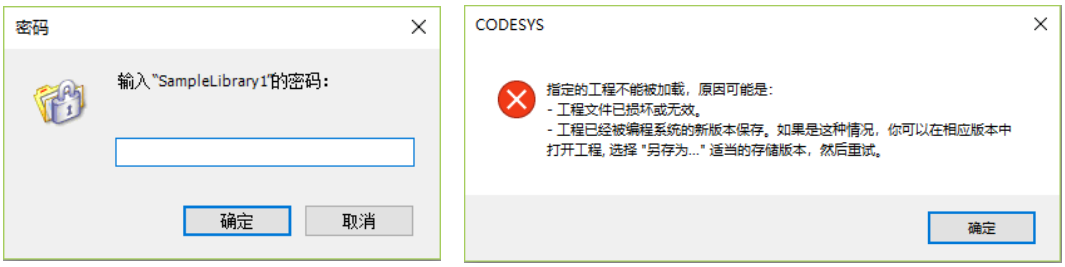
- 通过“项目设置”打开“工程设置”选择“安全”，比较常用的是设置“密码”进行保护

图表 3.3-10



设置完密码后重新保存工程并安装，之后每次打开库文件系统都会提示输入密码，密码错误则不能打开库文件查看源代码，这种加密方式也适用于程序文件，配置方式也一样

图表 3.3-11



3.3.2 任务配置

每一个 PLC 应用下都可以创建多个 POU，在同一个应用下，不同 POU 之间的执行顺序是通过“任务”来配置的。在“任务”中完成 POU 的配置后，POU 会按照任务配置周期性或由一个特定的事件触发开始执行。配置任务有以下几条规则：

- 循环任务的最大数为 100
- 惯性滑行任务的最大数为 100
- 循环任务的最大数为 100
- 主程序“PLC_PRG”可能会在任何情况下作为一个惯性滑行程序执行，而不需要手动插入任务配置中
- 处理和调用程序是根据任务配置内自上而下的顺序执行的

在介绍任务窗口的配置之前，先来了解一下 PLC 执行程序的流程。

图表 3.3-12

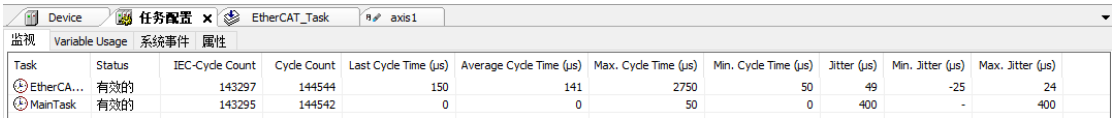
PLC 在一个扫描周期内除了上述 3 个阶段的流程之外，还需要完成内部诊断、通讯，输入输出数据的处理等。因此 PLC 扫描周期的长度取决于输入电路的硬件滤波时间，输出电路的滞后时间，PLC 循环扫描的工作方式和用户程序。

接下去，详细了解一下，影响程序执行的“任务配置”选项卡，在左侧树形菜单中找到“任务配置”选项，双击任务配置，在右侧配置界面中可以看到四个选项卡，分别是：

监视

当程序进入在线模式后，在“任务配置”右侧的监视窗口中，用户可以监控当前状态、循环次数、程序实际执行时间，平均/最大/最小循环时间等任务执行相关参数。

图表 3.3-13


Device | 任务配置 | EtherCAT_Task | axis1

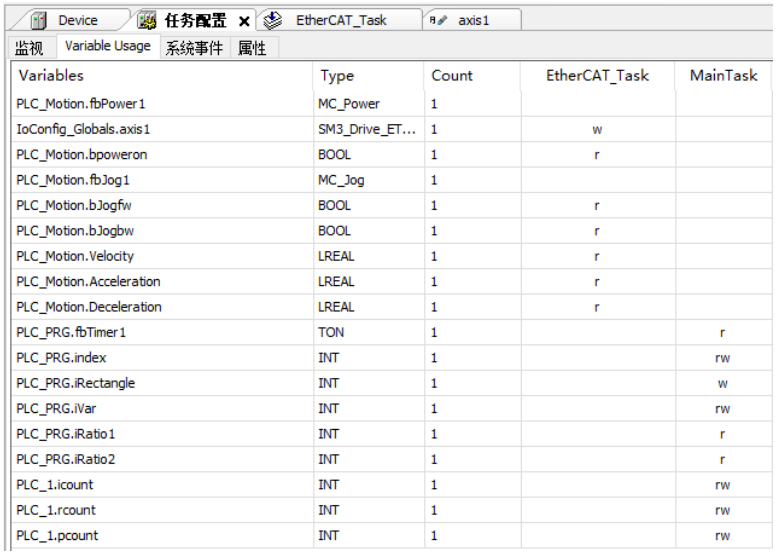
监视 | Variable Usage | 系统事件 | 属性

Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)	Min. Jitter (µs)	Max. Jitter (µs)
EtherCA...	有效的	143297	144544	150	141	2750	50	49	-25	24
MainTask	有效的	143295	144542	0	0	50	0	400	-	400

Variable Usage

在 Variable Usage 页面下用户可以观察在所有程序中变量的使用情况，包括变量的数据类型，在哪个任务中被调用，参与运算次数等

图表 3.3-14


Device | 任务配置 | EtherCAT_Task | axis1

监视 | Variable Usage | 系统事件 | 属性

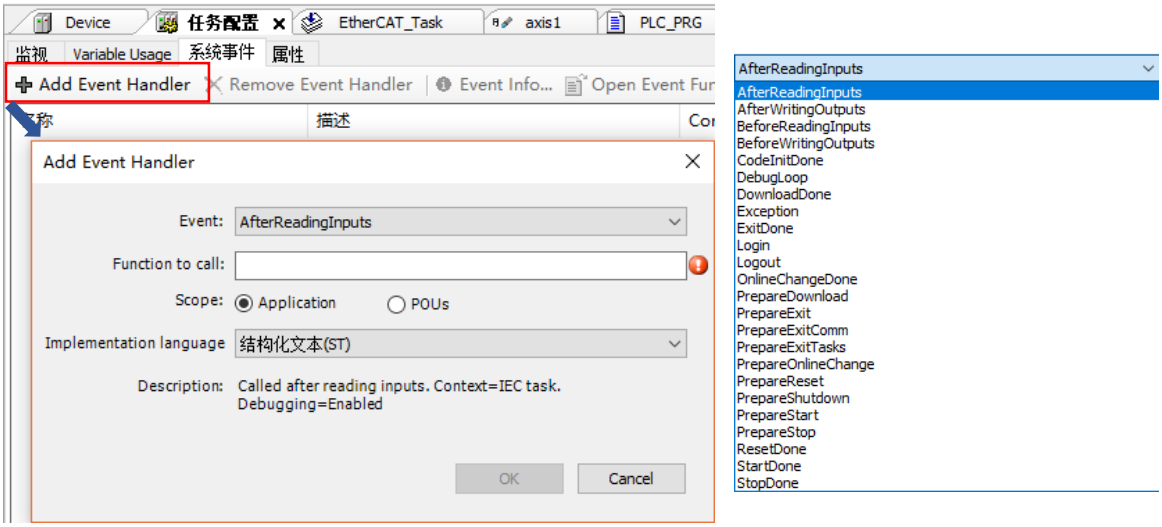
Variables	Type	Count	EtherCAT_Task	MainTask
PLC_Motion.fbPower1	MC_Power	1		
IoConfig_Globals.axis1	SM3_Drive_ET...	1	w	
PLC_Motion.bpoweron	BOOL	1	r	
PLC_Motion.fbJog1	MC_Jog	1		
PLC_Motion.bJogfw	BOOL	1	r	
PLC_Motion.bJogbw	BOOL	1	r	
PLC_Motion.Velocity	LREAL	1	r	
PLC_Motion.Acceleration	LREAL	1	r	
PLC_Motion.Deceleration	LREAL	1	r	
PLC_PRG.fbTimer1	TON	1		r
PLC_PRG.index	INT	1		rw
PLC_PRG.iRectangle	INT	1		w
PLC_PRG.iVar	INT	1		rw
PLC_PRG.iRatio1	INT	1		r
PLC_PRG.iRatio2	INT	1		r
PLC_1.icount	INT	1		rw
PLC_1.rcount	INT	1		rw
PLC_1.pcount	INT	1		rw

系统事件

用户可选择的系统事件是由实际硬件决定的，目标系统对应的库文件会提供相应的系统事件，因此，不同的硬件设备对应的系统事件可能会有所不同。在任务配置中，可以对系统事件进行设置，通用的系统事件有停止、开始、登入、登出和改变等。

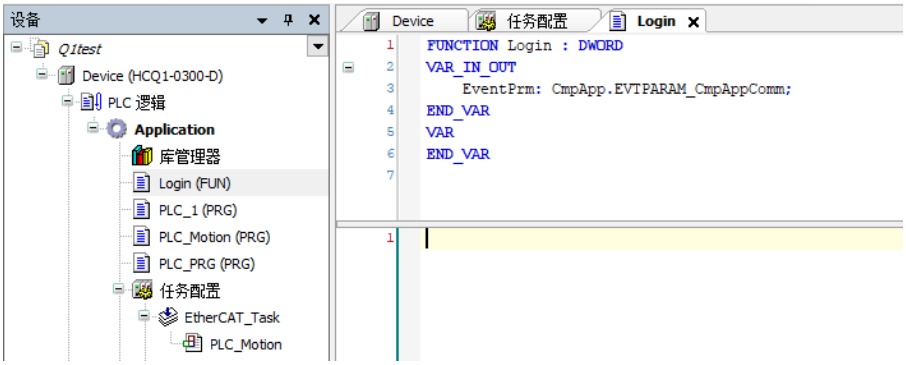
通过“任务配置”→“系统事件”，进入配置页面，点击“Add Event Handler”按钮可以添加系统事件，在弹出对话框的 Event 下拉可以选择需要添加的系统事件。

图表 3.3-15



添加完成之后，需要在“Function to call”中给定函数名称，不可以使用 POU 中已经存在的函数。“Implementation language”是调用函数所使用的编程语言，设置完成后点击 OK。以 Login 为例，选择系统事件为 Login（系统登录），在 Application 下会出现新创建的系统函数块，双击进入即可按照设定的编程语言进行编辑，该“Login”函数会在每次系统登录的时候自动调用触发一次。

图表 3.3-16

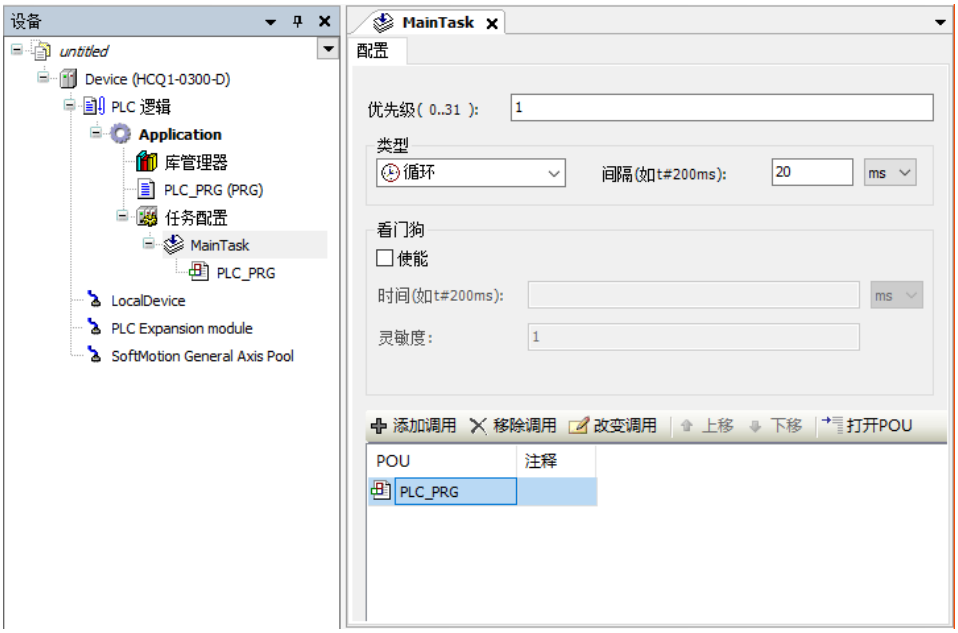


属性

属性选项卡下显示当前任务配置的基础属性。

除了任务配置页面下的设置之外，任务配置中还包含执行优先级、任务启动类型、看门狗设置和调用选项。接下去对它们进行详细的说明：

图表 3.3-17

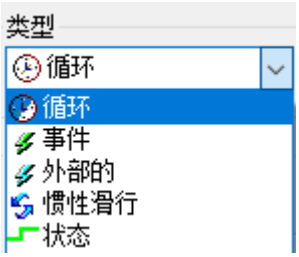


任务启动类型

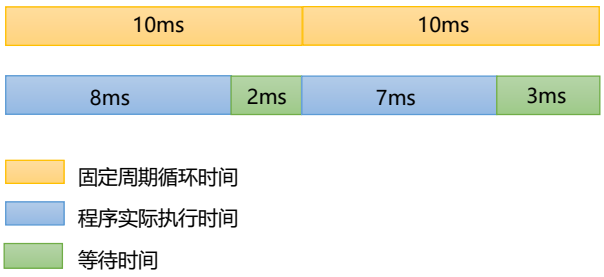
CODESYS 默认提供四种任务启动类型，加载了禾川 Q1 的设备描述文件之后可以增加“外部的”的选项，用户可以对于应用中配置的任务可以选择任意启动类型。

1. 循环（Cyclic）

该模式下的配置的程序会按照固定周期循环执行，根据程序量的大小和指令执行与否，程序实际执行时间会有所变化。



图表 3.3-18

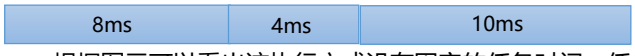


如果程序实际执行时间在规定的固定周期循环设定时间内，则空余时间用作等待，不会立即执行下一次循环。如果应用中还有优先级较低的任务未被执行，则剩下的等待时间用来执行相对低优先级的任务。

2. 惯性滑行（Freewheeling）

该模式下，程序一开始运行，任务就会调用该程序并执行，一个运行周期结束后，会立即进入下一个循环中自动重新启动，惯性滑行的执行方式不受程序扫描周期的影响，只确保程序执行完所有指令后自动进入下一个循环周期。图表 3.3-4 为惯性滑行执行顺序的时序。

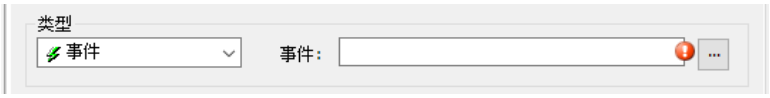
图表 3.3-19



根据图示可以看出该执行方式没有固定的任务时间，任务每次执行的时间可能都不一样，无法确保程序的实时性，在实际应用中使用较少。

3. 事件（Event）

图表 3.3-20

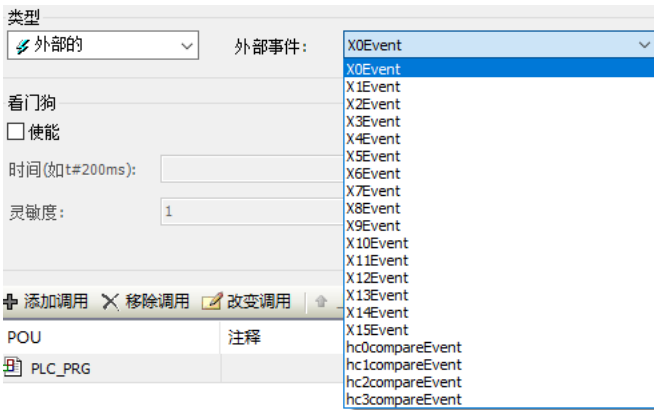


在该模式下，任务会在事件区域变量得到一个上升沿的时候开始执行。

4. 外部的（External）

外部触发模式，仅在用户成功加载 Q1 的设备描述文件情况下可以选择并使用，该模式包含两种类型的触发条件：　　（暂时无法测试，保留）

图表 3.3-21



① 用户可以通过 Q1 默认提供的高速输入对任务执行与否进行外部触发

②

5. 状态（Status）

状态触发模式和事件触发类似，但是事件触发是由事件变量的上升沿触发执行，而状态触发只需要事件区域的变量状态为 TRUE，任务就会执行。

优先级

CODESYS 中任务的优先级总共有 32 个级别（0~31 之间的一个数字，数字越小优先级越高）。当一个程序在执行时，优先级高的任务优先于优先级低的任务，用户不可以分配具有相同优先级的任务，一般运动控制程序所在任务区分于界面和其他程序逻辑所在任务并需要分配更高的优先级。

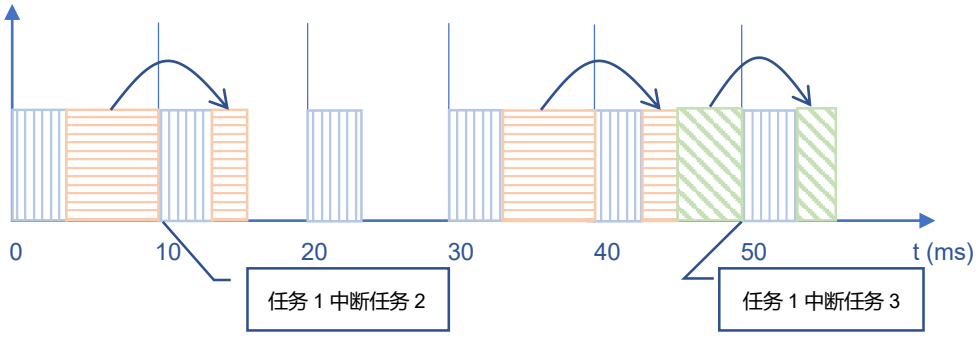
以“循环”任务类型为例，介绍优先级的执行顺序。假设有 3 个不同的任务，对应 3 种不同的优先级，具体配置如下：

图表 3.3-22

任务 1：优先级 0，循环时间 10ms

任务 2：优先级 1，循环时间 30ms

任务 3：优先级 2，循环时间 40ms

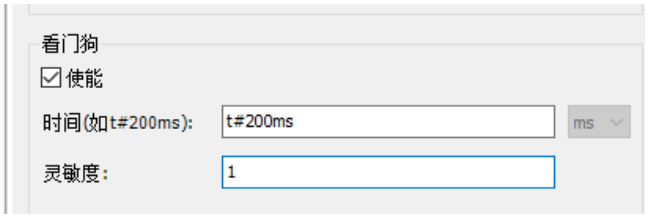


看门狗

“看门狗”（Watchdog）是一种控制器硬件式的计时设备，在“任务配置”中默认关闭，用于监控程序执行时出现的异常或者内部时钟发生的故障。当系统出现死机或者程序进入死循环时，“看门狗”会被触发对系统发出重置信号或者停止 PLC 当前运行的程序。

配置“看门狗”需要用户定义两个参数：时间和灵敏度。针对每个任务可以单独配置“看门狗”，默认的“看门狗”时间单位为 ms，给定“看门狗”时间后还需要设置“灵敏度”；“灵敏度”用于定义在应用或程序触发“看门狗”功能之前允许的例外数，默认值为 1。

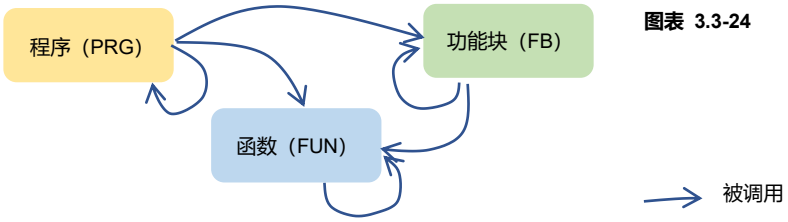
图表 3.3-23



因此实际“看门狗”触发时间=时间×灵敏度，程序实际执行时间超过设定“看门狗”时间，就会触发“看门狗”功能并终止当前任务及对应程序。“看门狗”功能一般应用于对实时性和安全等级要求较高的较高的场合，主要是为了防止 PLC 死机或者程序进入死循环。

3.3.3 PLC 编程

PLC 编程界面由声明区和代码区组成，编程时创建的 POU，按照功能划分可以分成函数（FUN）、功能块（FB）、程序（PRG），程序和功能块中使用的变量为静态变量，在登录状态下可以直接进行监控，并且在完成运算后的下一个周期可以被调用；对于函数来说，它的返回值是唯一的，并且函数内使用的变量为临时变量，需要设置断点，否则无法直接在登录状态下查看变量的值。三种块类型之间的调用关系如下：



图表 3.3-24

用户可以在树形菜单“Application”右击找到“添加对象”后选择“POU”，在弹出对话框中，用户可以选择添加程序、功能块或函数，“实现语言”下拉可以选择对应的编程语言

图表 3.3-25

当用户在声明区给定变量的名称、类型和初始值，所有需要在程序中使用的变量都需要在声明区定义，这些变量包括：输入变量、输出变量、输入/输出变量、本地变量和常量。变量声明格式基于IEC61131-3，简单归纳为以下几点：

- 变量首字符不能是数字，可以是字母或下划线
- 变量名不区分大小写
- 变量名不允许出现连续的下划线
- 变量名不允许出现空格和特殊字符
- 不能使用关键字、函数和功能块名称作为变量名
- 程序中的关键字会自动以蓝色大写出现
- 单行注释可以用 “//”表示，注释需要插入在声明语句（非字符串内部）中间可以选择“(**)”，支持中文注释，

  具体格式如下，:

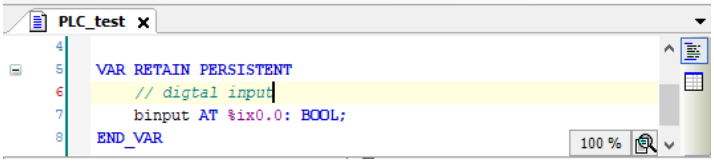
  {(*<注释内容>*)}

  <变量名称> {AT<Address>}：<数据类型>{：=<初始值>}； {//<注释内容>}

  { }内为可选部分

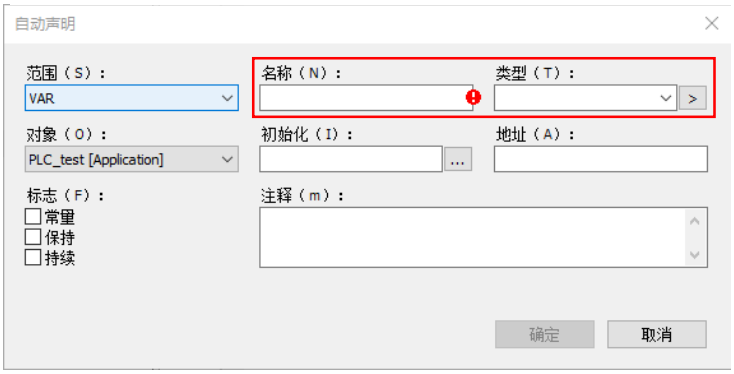
  其中具有固定地址的变量通过输入寄存器（I）、输出寄存器（Q）和内存区（M）来区分变量存储区域，通过位（X）、字节（B）、字（W）、双字（D）来定义变量类型

按照上述格式在声明区定义一个数字量输入：



对于不熟悉变量声明格式的用户，可以在声明区同时选择 Shift+F2 或者右击调用“自动声明”对话框，来避免格式上的错误，其中红框内的部分必须填写，其他部分可根据客户实际需求填写：

图表 3.3-26



范围：变量作用域，本地变量或接口变量

名称：变量名，按照 IEC61131-3 标准定义

类型：变量数据类型

对象：变量所在应用

初始化：变量初始值

地址：外部变量地址

标志：定义变量为常量、保持型或持续型变量

注释：变量注释

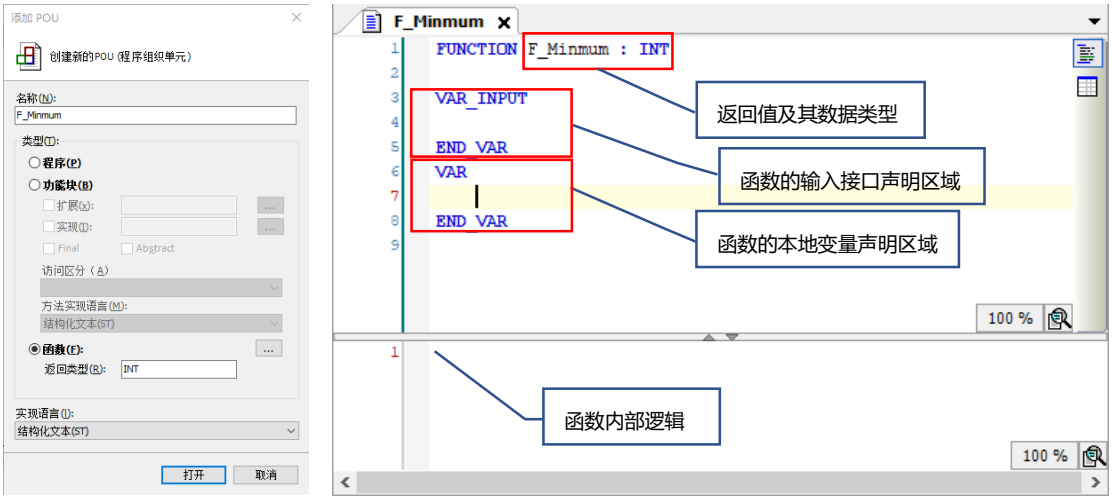
在代码区，用户可以根据自己的需求和擅长语言选择编程语言。接下去对 POU 中提供的三种块类型做更为详细的说明。

函数

函数（FUN）是至少有一个输入变量，无静态变量，仅有一个返回值的基本算法单元，可以被函数、功能块和程序调用。

函数的内部逻辑可以使用 IEC61131-3 规范中的任意一种，函数名即为函数的返回值，也可以理解成函数的输出，自定义函数的界面如下：

图表 3.3-27



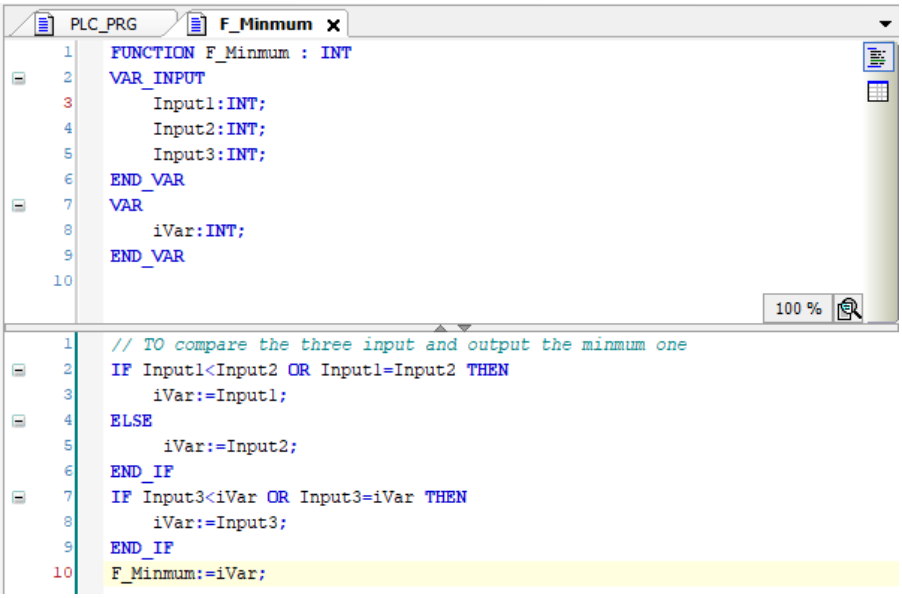
用户按照以上格式自定义函数时，应注意如下事项：

- 函数可以拥有多个输入变量，但是只能拥有唯一一个返回值，对于返回值的数据类型没有限制，可以是任意数据类型，甚至是用户自定义的结构体
- 函数没有指定的内存分配，不需要像功能块一样在调用前进行实例化，用户直接在函数定义好的输入端口赋值，就可以在得到唯一的返回值
- 函数的内部变量无法存储数值，所以当用户需要查看函数在运行过程中产生的数值，需要配合断点进行查看
- 函数的输入端口（Var_INPUT）可以是空的、常数、变量或者是函数调用。

【例 1】：按照以上说明，创建一个函数，输出三个整型变量中的最小值。

函数声明：

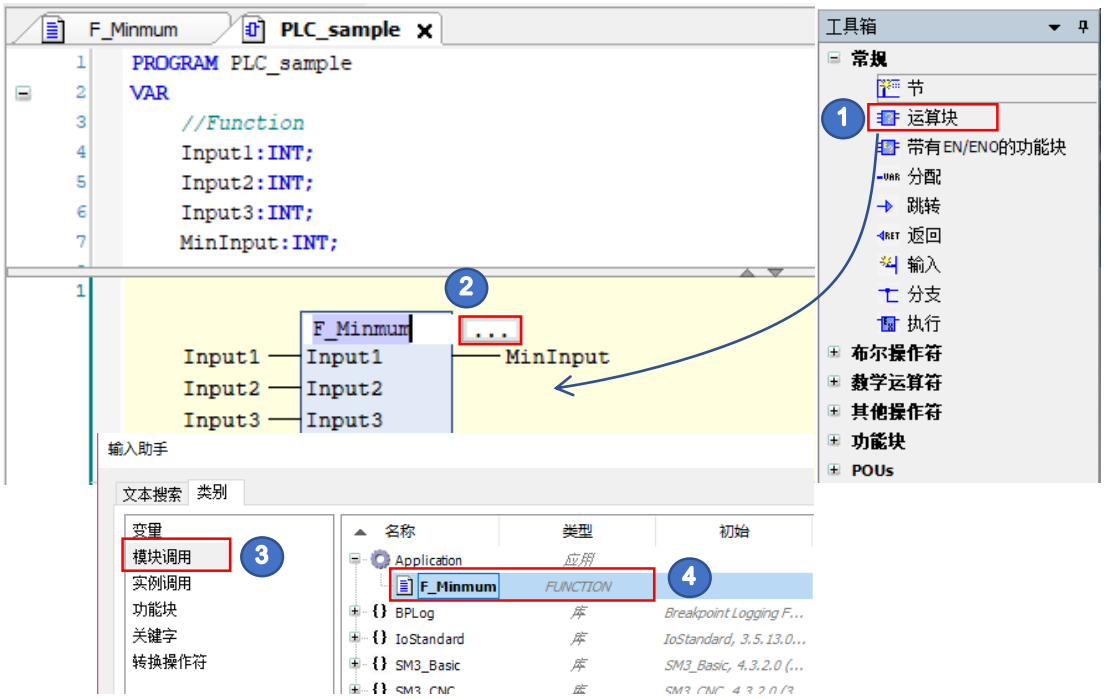
图表 3.3-28



函数在程序中的调用：

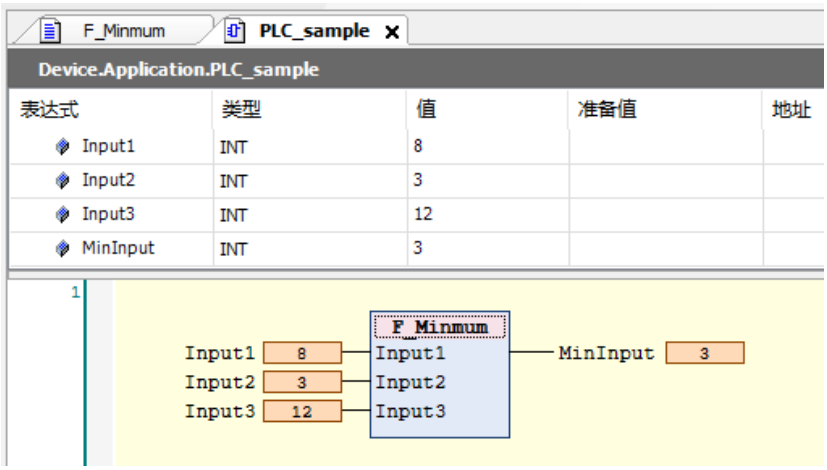
函数的调用无需实例化，为了方便查看函数接口等，选择程序的编程语言为“FBD”在程序编辑区通过“工具箱”→“常规”选择“运算块”，拖拉到程序编辑区，按照输入输出接口填写完整即可—

图表 3.3-29



函数在线运行结果：

图表 3.3-30

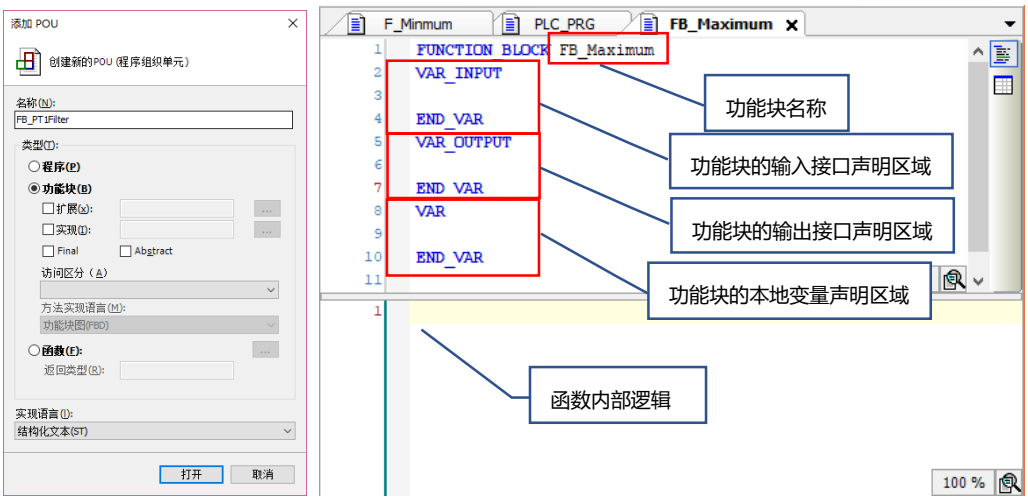


功能块

功能块（FB）是把反复使用的部分程序进行封装，以便用户在编程中调用类似功能的程序组织单元。功能块拥有自己的内部变量，这些内部变量在控制器内执行时需要分配内存，这一点区分于函数。

功能块的内部逻辑编写语言同样可以使用 IEC61131-3 规范中的任意一种，自定义功能块的界面如下：

图表 3.3-31



用户按照以上格式自定义功能块时，应注意如下事项：

- 为确保功能块不依赖于硬件，功能块的变量声明中不允许将具有固定地址的地址变量（如%IX1.1）作为局部变量，但在调用时可以给其赋值
- 功能块在调用前需要实例化，这一点和函数不一样，在同一个程序中多次调用同一个功能块时，建议定义不同的实例
- 功能块支持添加方法（Method），扩展（Extend）、实现（Implement）等面向对象的编程方式

函数和功能块调用后的表达式虽然类似，但是两者具有非常明显的区别，主要区别见下表：

图表 3.3-32

	函数（FUN）	功能块（FB）
内存分配	没有指定的内存分配地址	全部数据分配内存地址
输入/输出变量	只允许一个输出变量	输出变量无限制，可以没有也可以多个
调用关系	可调用函数，不可调用功能块	可以调用函数和功能块

【例 2】：按照以上说明，自定义一个功能块，实现简单 PT1 一阶低通滤波算法，通过调节增益参数 rk 和时间常数 tT 来使输出更为平滑，其中滤波系数α为采样周期/（滤波时间+采样周期）（滤波系数 0<α<1），该滤波算法有如下规律，用户根据自己的需求设置低通滤波增益系数和时间常数：

- 时间常数越大，滤波系数越小，滤波结果越稳定，但是灵敏度低
- 时间常数越小，滤波系数越大，灵敏度越高，但是滤波结果越不稳定

功能块声明：

图表 3.3-33

F_Minmum | PLC_PRG | FB_PT1Filter | FB_Peak | PLC_sample | Trace

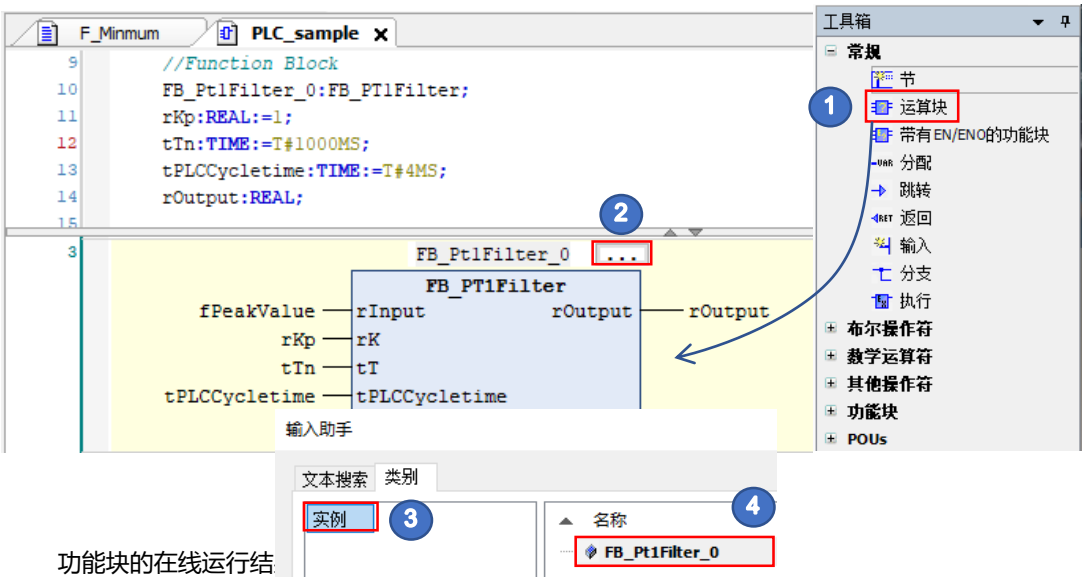
```
FUNCTION_BLOCK FB_PT1Filter
VAR_INPUT
    rInput:REAL:=0.0;           //采样值
    rK:REAL:=1.0;               //增益
    tT:TIME;                    //时间常数
    tPLCCycletime:TIME;         //PLC循环周期
END_VAR
VAR_OUTPUT
    rOutput:REAL:=0.0;          //输出数据
END_VAR
VAR
    _rT:REAL;
    _rPreOutput:REAL;           //上一个周期输出数据
END_VAR
```

```
//简单一阶滞后滤波功能
IF _rT=0.0 THEN
    rOutput:=rInput;                        //输出数据初始化
END_IF
_rT:=(TIME_TO_REAL(tT)+TIME_TO_REAL(tPLCCycletime))/TIME_TO_REAL(tPLCCycletime);
IF ABS(rOutput-rInput)<1E-6 THEN
    rOutput:=rInput;
ELSIF _rT>0.0 THEN
    rOutput:=(rK/_rT)*rInput+(1.0-(1.0/_rT))*_rPreOutput;
ELSE
    rOutput:=rInput;
END_IF
_rPreOutput:=rOutput;
```

功能块在程序中的调用：

功能块需要实例化后才能在程序中调用，为了方便查看功能块接口等，选择程序的编程语言为“FBD”在程序编辑区通过“工具箱”→“常规”选择“运算块”，拖拉到程序编辑区，按照输入输出接口填写完整即可

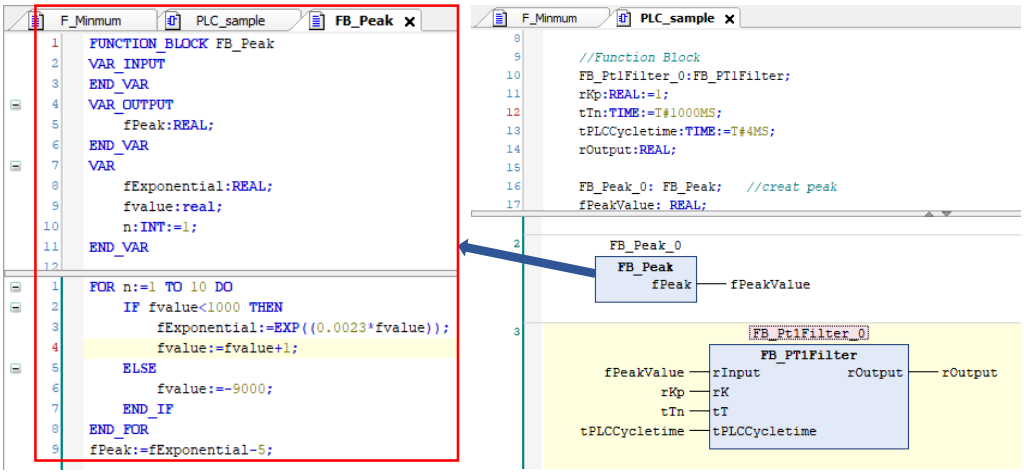
图表 3.3-34



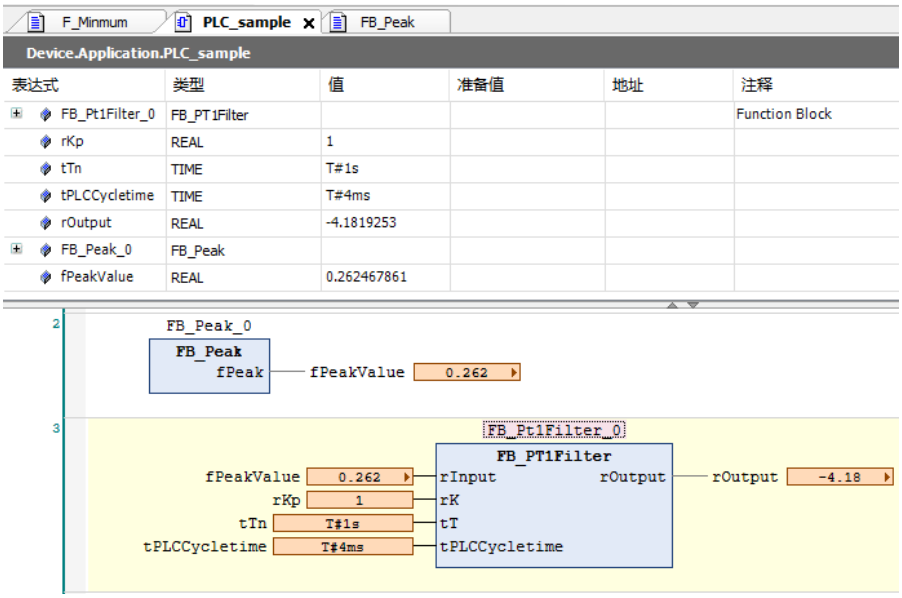
功能块的在线运行结

为了更好显示一阶低通滤波的实际滤波效果，添加单边指数脉冲波形

图表 3.3-35

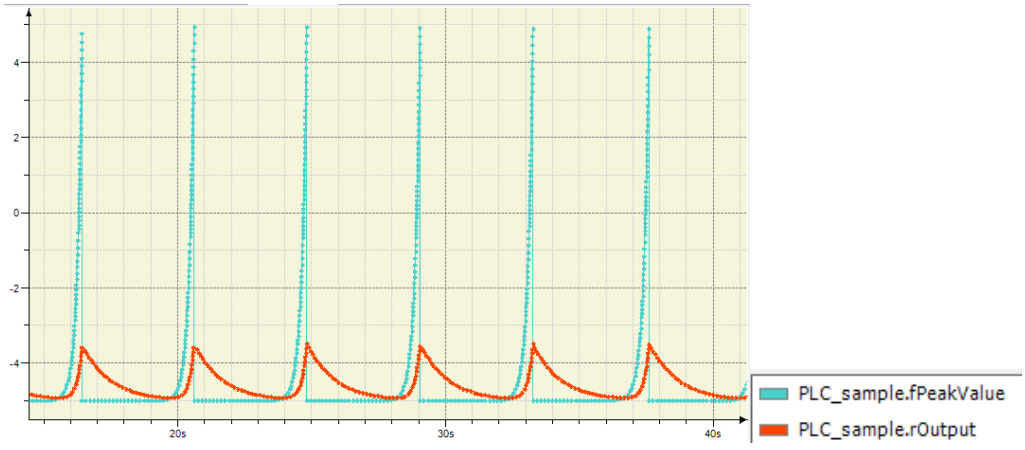


图表 3.3-36



rOutput 输出波形：

图表 3.3-37



程序

程序是规划一个任务的核心，在程序中可以定义局部变量、全局变量、外部变量（映射硬件地址）。一个程序可包含地址的配置，允许声明存放 PLC 物理地址的直接表示变量，直接表示的地址配置仅用于程序中内部变量的声明，直接表示变量允许分级寻址方式描述，可以直接在程序声明中按照如下格式填写：

bVar AT%IX0.0：INT；

在程序编辑区可用如下的语句给直接表示变量赋值：

%Q0.0：=TRUE；

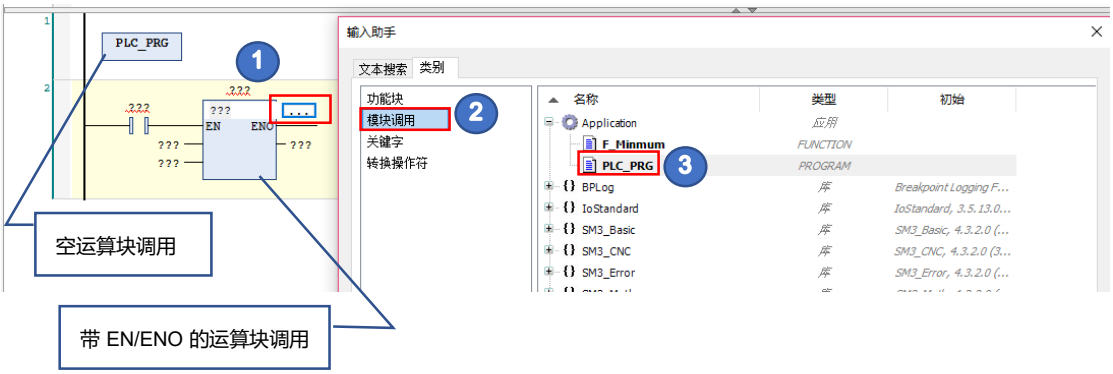
程序组织单元不能直接或者间接调用其本身，但是可以调用其他程序。程序的需要被任务调用才会参与编译和实际执行，否则程序将不被执行。

在选择文本编程语言“ST”时，调用程序的格式如下：

PRGsample（）；

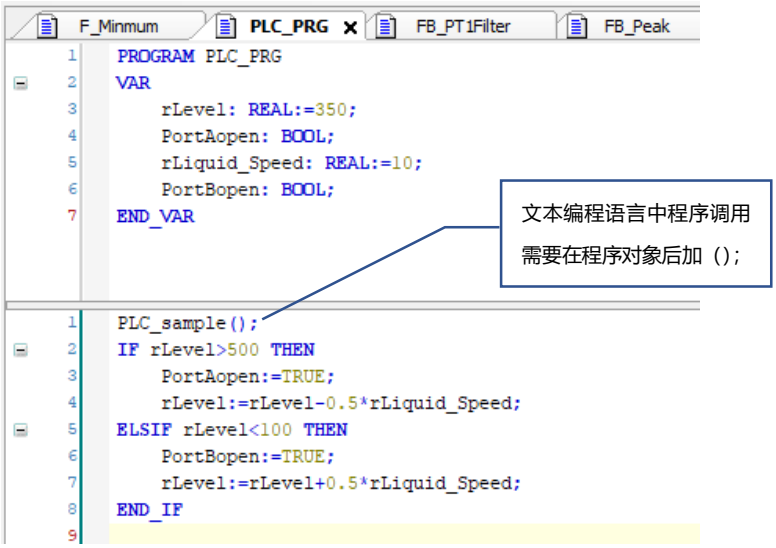
在选择图形化编程语言“LD”时，通过插入空运算块或者带 EN/ENO 的运算块后，在模块调用中找到程序对象，完成调用，调用程序的格式如下：

图表 3.3-38



【例 3】：按照以上说明，编写程序实现液位控制，当 rLevel 大于警戒水位 500 时，进行放水，小于最低水位 100 时进行注水。

图表 3.3-39

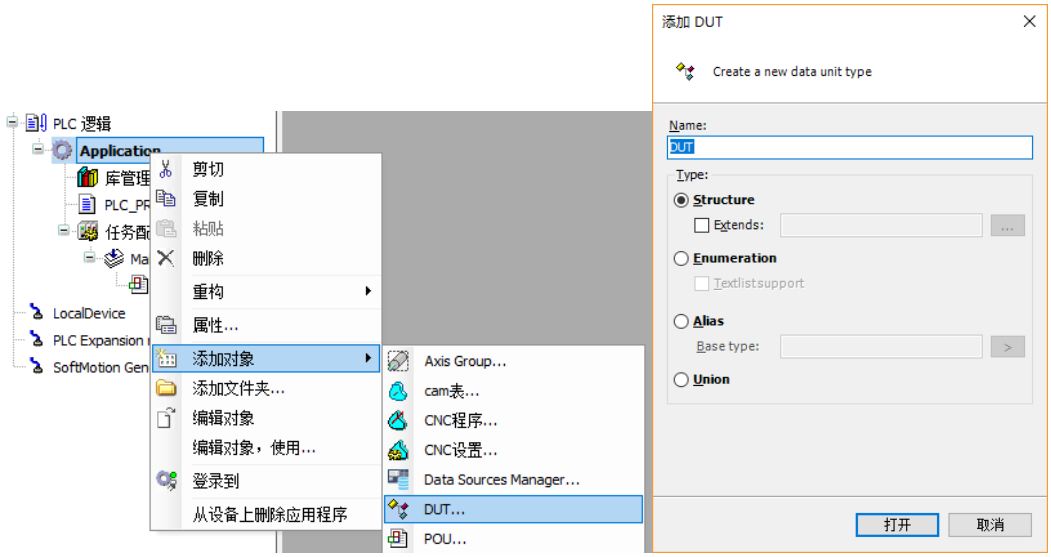


3.3.4 数据单元类型

数据单元类型(Data Unit Type,DUT)也可以称为用户自定义数据类型,其中包括枚举(Enumeration)、结构体（Structure)、别名（Alias）和联合（Union)。

右键单击“Application”选择“添加对象”→“DUT”，在弹出对话框“添加 DUT”中选择需要添加的数据单元类型，给定名称后，点击“打开”进行创建

图表 3.3-40



和功能块类似，数据单元类型中的结构体也支持使用 Extends(扩展)，也就是说用户可以通过一个已经在另一个工程中定义的 Structure 扩展另一个 Structure，是一种对原有结构基础的补充。

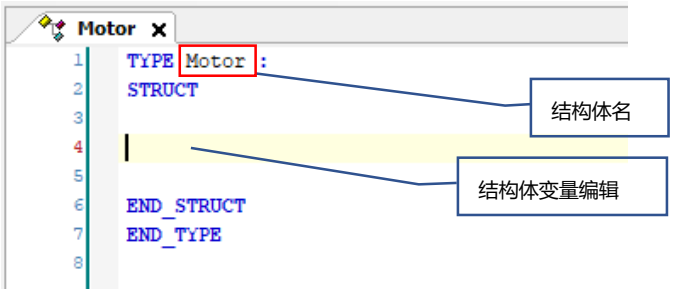
接下去按照顺序对三种用户自定义数据类型做简单介绍:

结构体

结构体（Struct）是由一系列具有相同货不同类型的数据构成的数据集合，是一种用户自定义数据类型。用户有时候需要将不同数据类型的数据组合成一个整体以便引用，例如一台电机通常都有其对应的信息，包括产品型号、生产厂家、额定电压、额定电流、极对数以及是否有抱闸等信息。这些信息都和这个电机相关联，以独立变量进行声明很难反应变量和电机之间的内在关系，此时就可以选择使用结构体。

结构体声明语句如下 ：

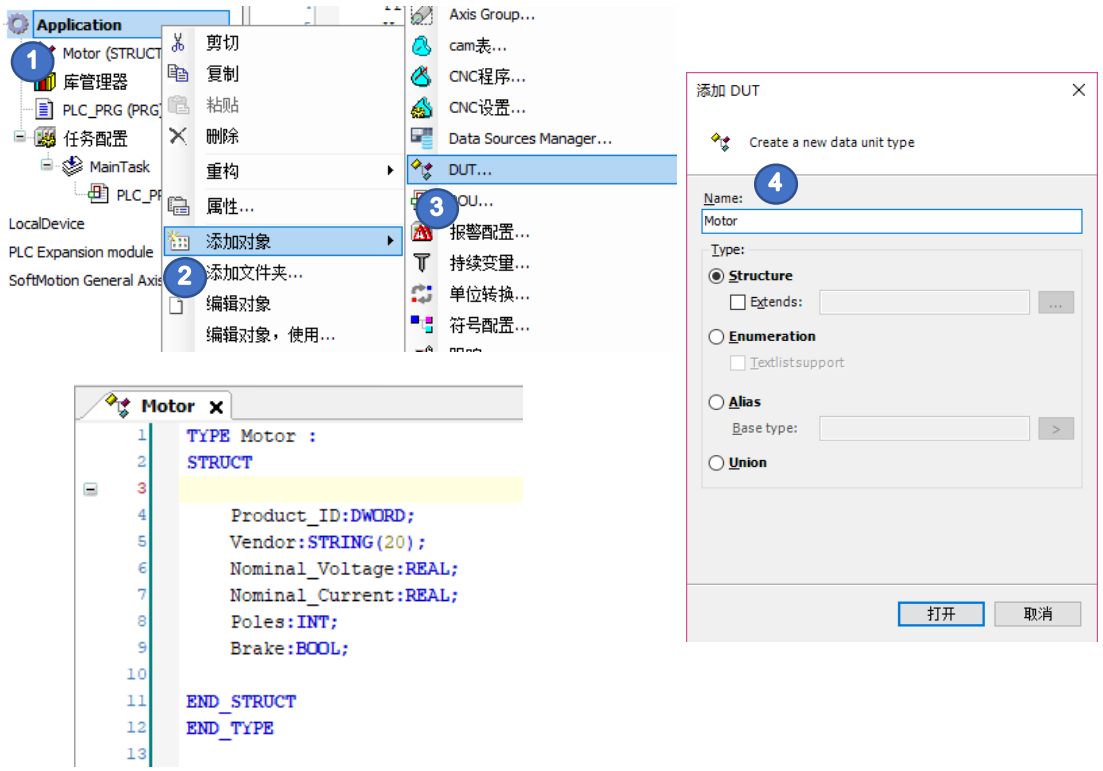
图表 3.3-41



【例1】 在数据单元类型中声明一个 Motor 结构体，包含电机的型号，厂家、额定电压、额定电流、极对数和是否有抱闸等信息，并在结构体 Motor 的基础上扩展结构体 Motor_hcfa，增加元素防护等级和额定扭矩，并在程序中调用该结构体。

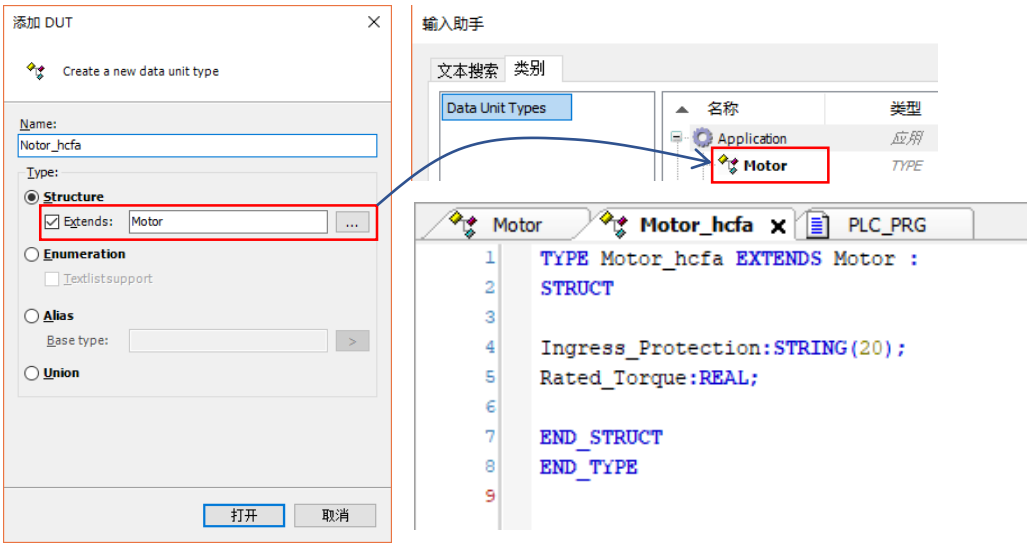
声明结构体 Motor：

图表 3.3-42



扩展结构体 Motor_hcfa：

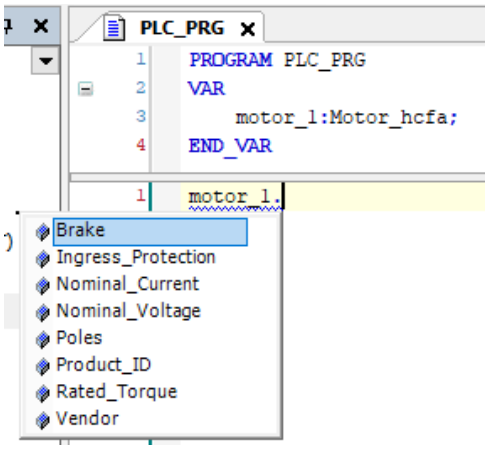
图表 3.3-43



在程序中调用结构体 Motor_hcfa：

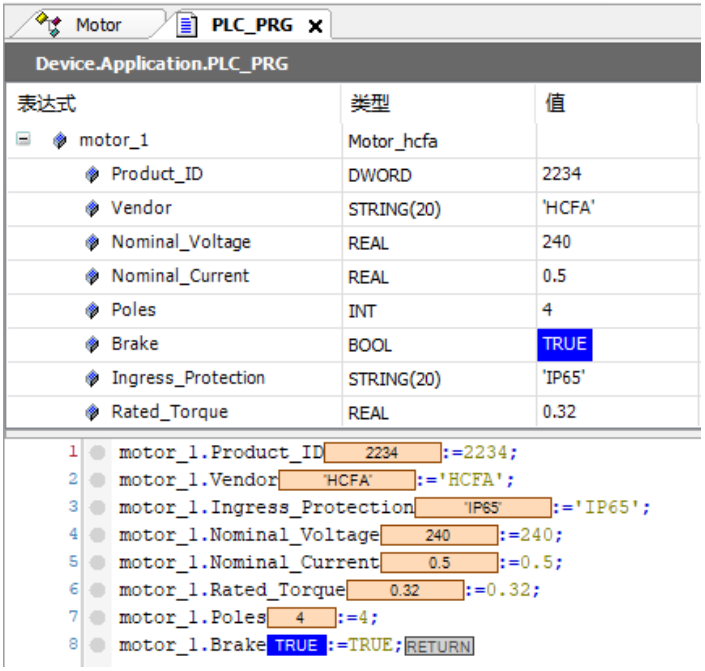
结构体在程序中的调用和功能块类似，需要首先在声明区实例化，之后可以直接在程序中使用声明好的结构体变量，结构体变量中的元素可以通过“.”索引的方式调用：

图表 3.3-44



逐一调用结构体中元素并赋值：

图表 3.3-45

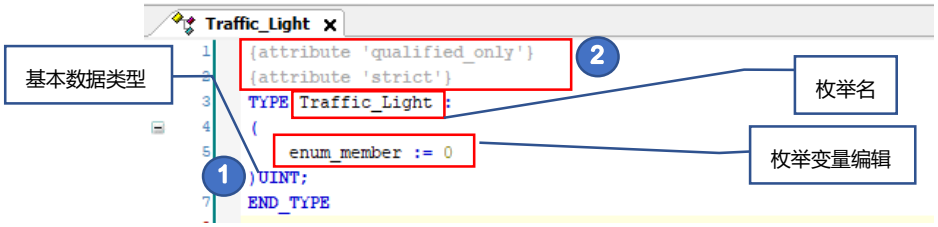


枚举

枚举（Enum）是一个被命名的整型常数的集合，枚举在日常生活中十分常见。如果一个变量有几种可能的值就可以定义成枚举类型，例如定义变量表示交通灯状态，这个变量可以有三个值：红色、黄色和蓝色，那用户就可以定义描述交通灯状态的枚举类型。

枚举类型的声明语句如下：

图表 3.3-46



- 枚举类型的基本数据类型默认是 INT 类型，也可以由用户明确指定，如上图①所示位置
- 枚举类型在没有赋初始值的情况下从 0 开始按照 INT 整型数据递加
- 整数可以直接复制给一个枚举类型

注意：默认的枚举声明中会自动插入属性编译，在上图②位置可以，其中通过{attribute 'qualified_only'}定义的枚举需要通过指定的全局变量名寻址，通过{attribute 'strict'}定义的枚举不可以参与数学运算，也不可以将其他数据类型包括常量的值直接赋值给枚举元素。如果用户不需要属性编译可以直接删除。

否则编译会出现类似下图报错。

C0358: 'test' is not a valid value for strict ENUM type 'Weekday'

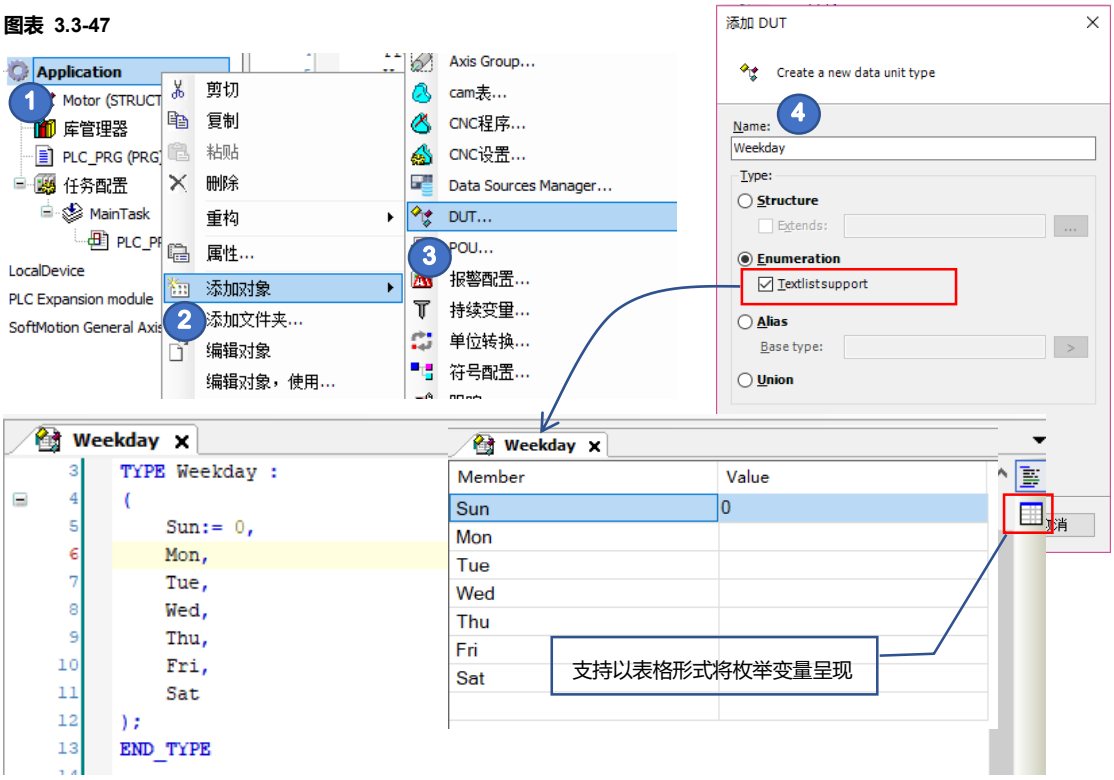
C0359: Arithmetics not allowed on strict ENUM type 'Weekday'

C0358: '(Weekday_1 + UINT#1)' is not a valid value for strict ENUM type 'Weekday'

【例2】 使用枚举类型，将一个星期中的七天按照 Sun、Mon、Tue、Wed、Thu、Fri 和 Sat 的形式显示，并在一个任务周期变化一次。

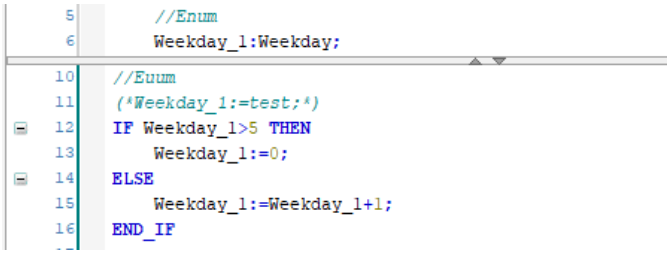
声明枚举：

图表 3.3-47



在程序中调用枚举 Weekday：

图表 3.3-48



```
    //Enum
    Weekday_1:Weekday;

//Euum
(*Weekday_1:=test;*)
IF Weekday_1>5 THEN
    Weekday_1:=0;
ELSE
    Weekday_1:=Weekday_1+1;
END_IF
```

登录并查看运行结果如下：

图表 3.3-49

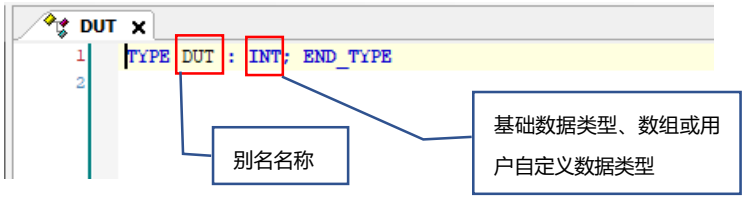
```
//Euum
(*Weekday_1:=test;*)
IF Weekday_1[Sat]>5 THEN
    Weekday_1[Sat]:=0;
ELSE
    Weekday_1[Sat]:=Weekday_1[Sat]+1;
END_IF
```

别名

别名（Alias）简单来说就是给一个基础数据类型、数组或用户自定义数据类型另一个名称，方便用户间接的使用和管理这个数据类型声明的变量。例如定义一个长度固定的字符串数据类型存放 IP 地址，为了方便用户识别和统一修改 IP 地址的默认长度，此时可以选择使用别名。

别名的声明：

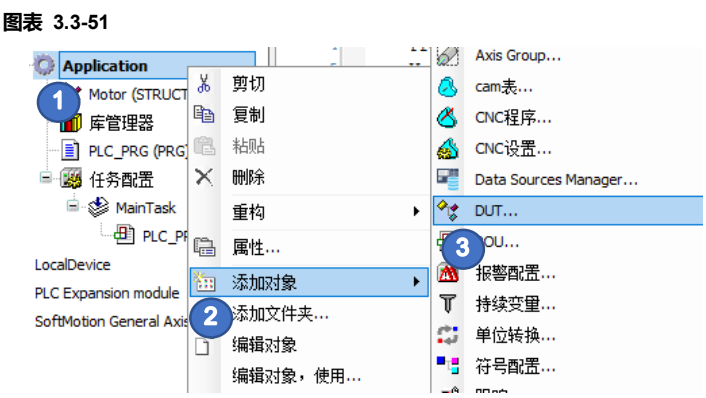
图表 3.3-50

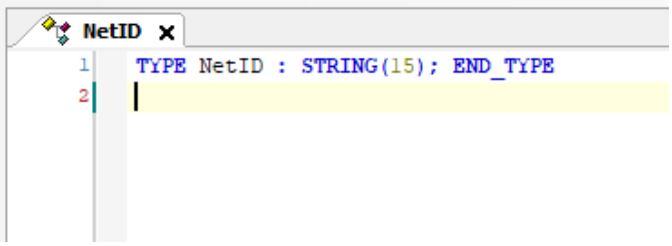


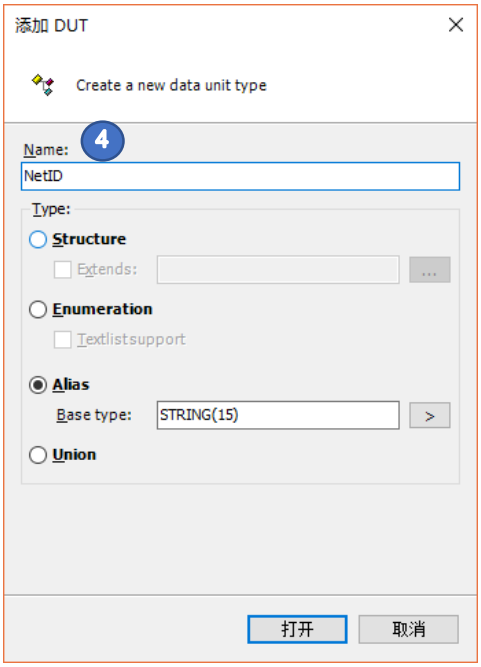
【例 3】使用别名 NetID 定义一个字符串 String（15）以存放 IP 地址。

声明别名：

图表 3.3-51







在程序中调用别名 NetID 并赋值，当 NetID 在程序、功能块、功能等多处调用的情况下，用户因为实际需求需要修改 NetID 的长度时，只需要修改别名的定义即可统一完成修改：

图表 3.3-52

```
    //Alias
    IPAddr:NetID;
END_VAR
//Alias
IPAddr:='192.168.1.1';
```

联合

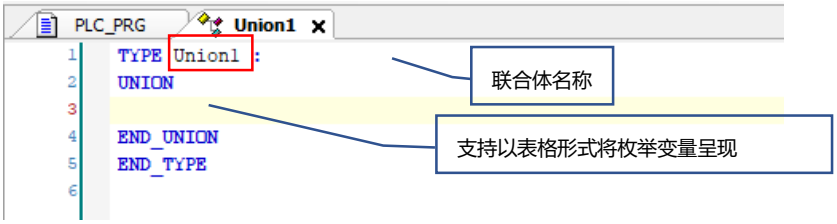
联合体（Union）也叫共用体，可以将几种不同类型的变量存放到同一段内存单元中，如可以将 INT 型变量、BYTE 型变量和一个 DWORD 类型放在同一个地址开始的内存单元中，如下表，不同类型变量都从同一地址 16#000 开始存放：

图表 3.3-53

	16#000	16#001	16#002	16#003
INT				
BYTE				
DWORD				

联合体的声明：

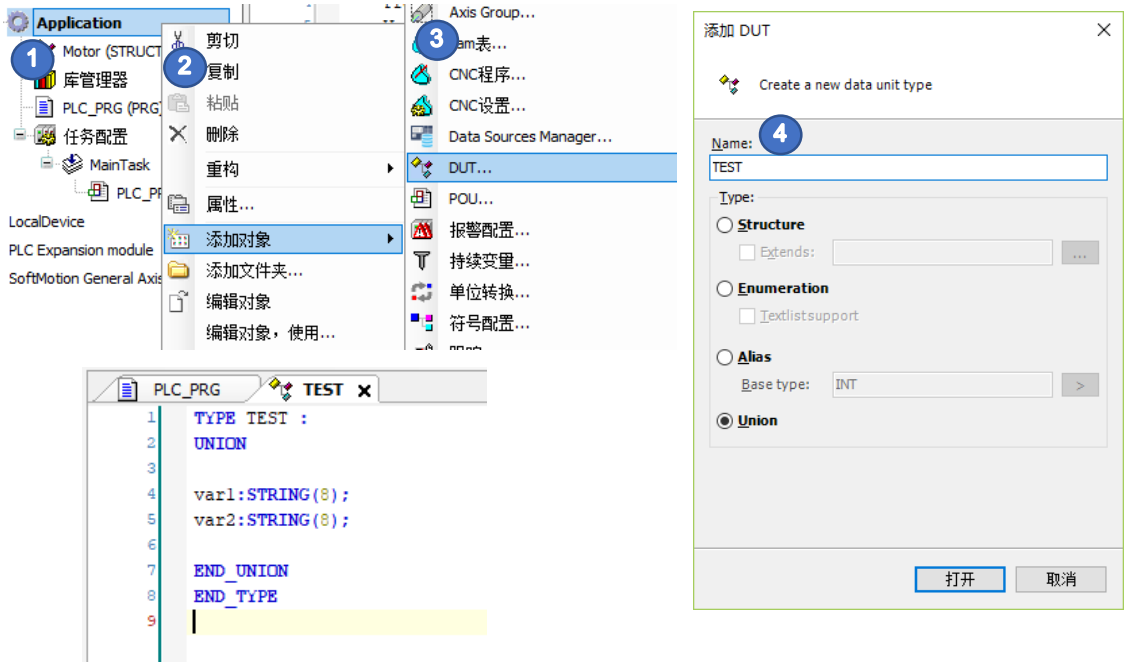
图表 3.3-54



【例3】 定义一个联合体 TEST，拥有多个成员，对其中一个元素赋值，查看其它元素的值。

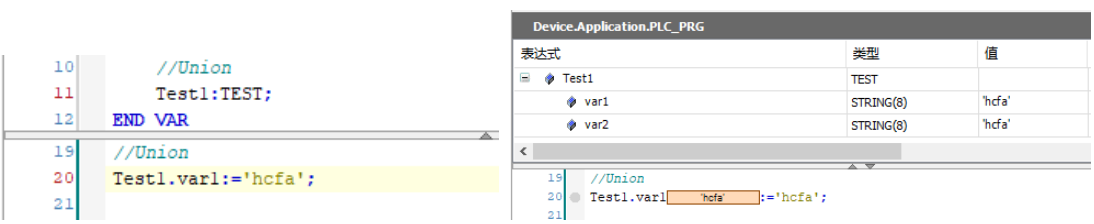
声明联合体：

图表 3.3-55



在程序中调用联合体 TEST，并给其中一个元素 var1 赋值，可以发现 var2 同时也会共享输入的值：

图表 3.3-56

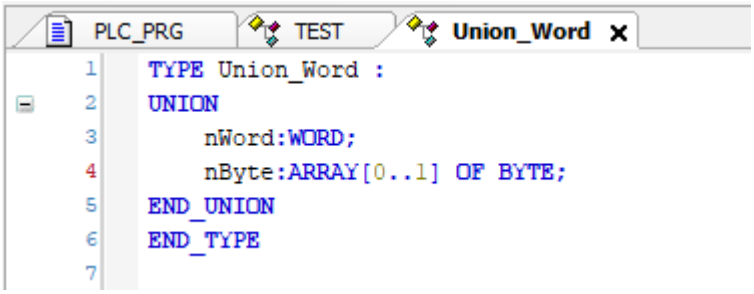


此外，联合体的元素数据类型也可以不一样。

【例4】 使用联合体实现将两个字节变量整合成一个字变量的功能。

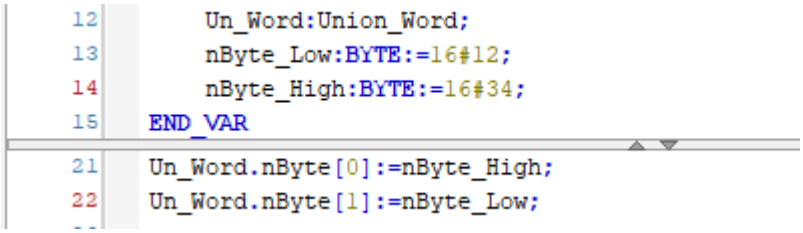
声明联合体：

图表 3.3-57



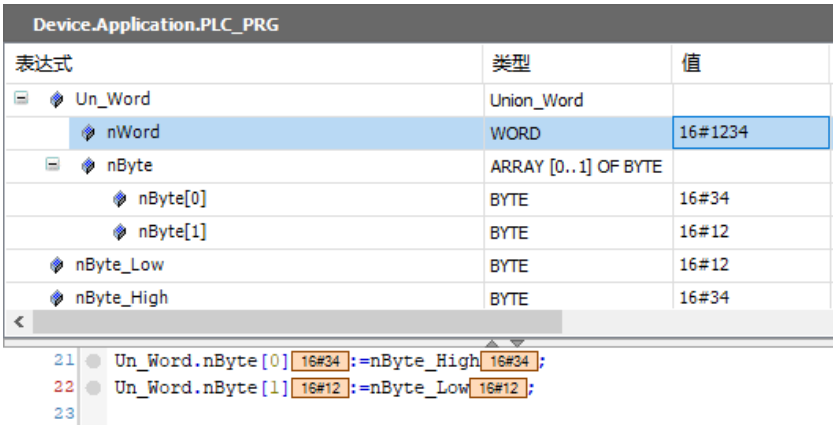
在程序中调用联合体 Union_Word，并编写示例程序如下：

图表 3.3-58



登录并查看运行结果，可以看到在程序中给数组 nByte 中的元素赋值，nWord 由于联合体的关系，数值也一并被写入：

图表 3.3-59



在联合体 Union_Word 中地址映射关系如下：

图表 3.3-60

变量	高 8 位	低 8 位
nWord	15~8	0~7
nByte[0]	15~8	
nByte[1]		0~7

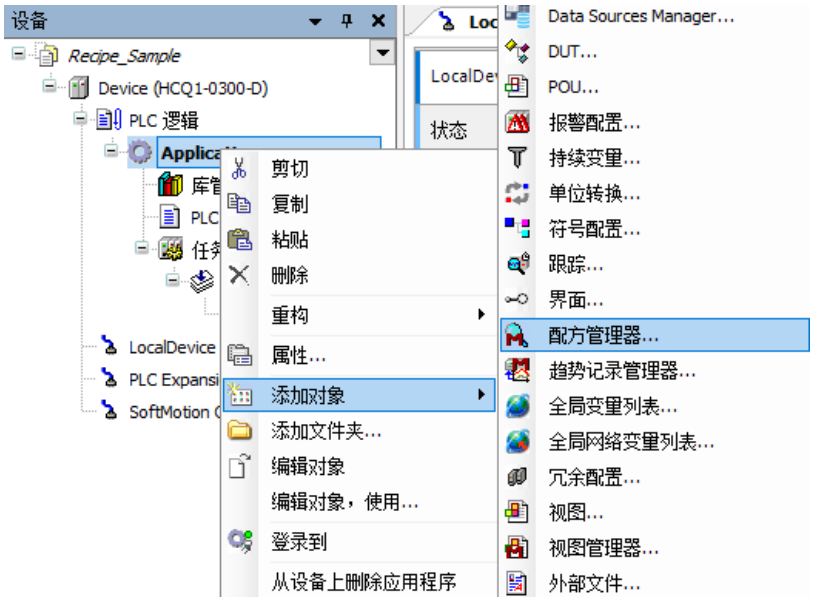
从上表可以看出 nByte[1]对应 nWord 的低 8 位，nByte[0]对应 nWord 的高 8 位，因此在程序中可以通过将两个 Byte 类型的变量直接对应数组当中的元素，即可实现将两个 Byte 类型的数值整合成一个 Word 类型变量的功能。需要注意的是，如果在联合体中声明的变量的数据类型不一致，需要保证数据类型占用的存储空间大小相同，否则可能会出现数据错乱。

3.3.5 配方管理器

配方是一组参数值，用来提供生产产品和控制生产过程中所需的信息。例如，生产面包的原料（包括小麦粉、鸡蛋、黄油、白砂糖等）和烘焙时间等生产过程需要的参数的数据类型和数值等。配方也可以用于设置和监控 PLC 的控制参数，为了达成这个目的，用户可以从 PLC 读取和写入数据，也可以从文件载入或者存成文件。

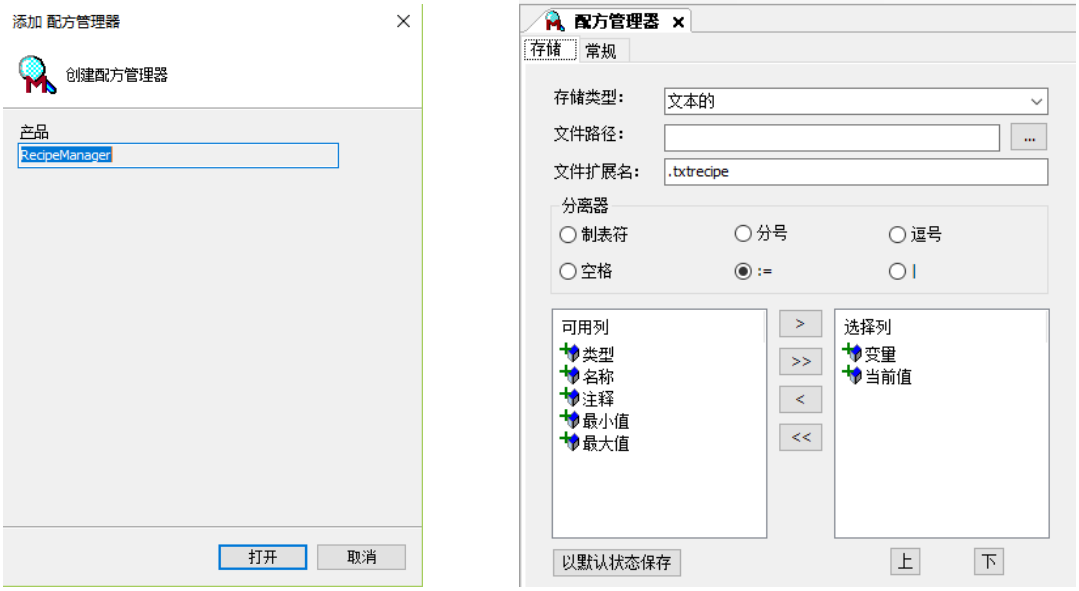
通过“Application”右击选择“添加对象”→“配方管理器”添加配方管理器

图表 3.3-61



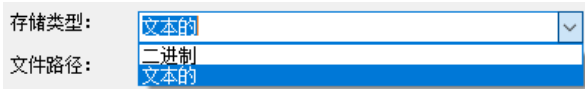
左图为添加配方管理器的界面，右图为添加后的配方管理器配置页面

图表 3.3-62



在配方管理器的配置界面中，用户可以在“存储类型”中选择“文本的”或者“二进制”的存储方式

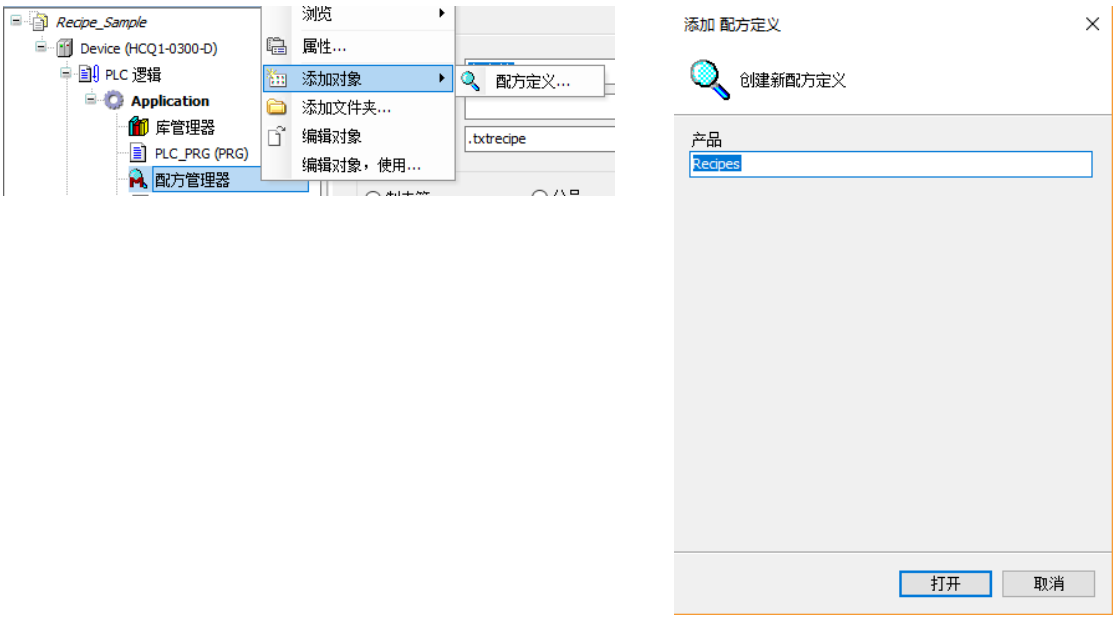
图表 3.3-63



在“存储”选项卡的 “文件路径”中指定配方文件的存储路径，也可以定义存储配方的“文件扩展名”，文本的存储会根据所选的“分离器”（分隔符）将其和配方名称分开，最终格式显示为<方法>.<定义方法>.<扩展文件>，其中“分离器”只有在储存类型为文本的选择时生效。在配方管理器的配置页面中还显示了所有配方定义的列，即“可用列”，右侧为当前配方定义的列，即“选择列”，“选择列”中的内容会被存储，可用通过 < 或 > 按钮，可用在“可用列”和“选择列”之间移动选择的配方定义，也可以使用 >> ； << 所有条目一次性移动，按钮 上 下 用调整选择列中选择的配方定义的顺序，储存文件会按照“选择列”中设定的顺序保存。

在配方管理器的配置页面设置完成后，通过右击“配方管理器”选择“添加对象”→“配方定义...”添加新的配方定义

图表 3.3-64



添加的配方定义页面如下，在配方定义①中，用户可以为变量定义不同的值集作为不同的配方②，通过③和④可以切换配方定义的显示。配方定义中显示的内容即为配方管理器配置页面中提供的列内容，其中，配方定义下的变量可以通过变量右侧的 ... 添加程序中声明好的变量，用户可以在配方定义中自定义“名称”和“注释”，通过设定“最小值”和“最大值”来限制在配方定义中用户可以输入的值。

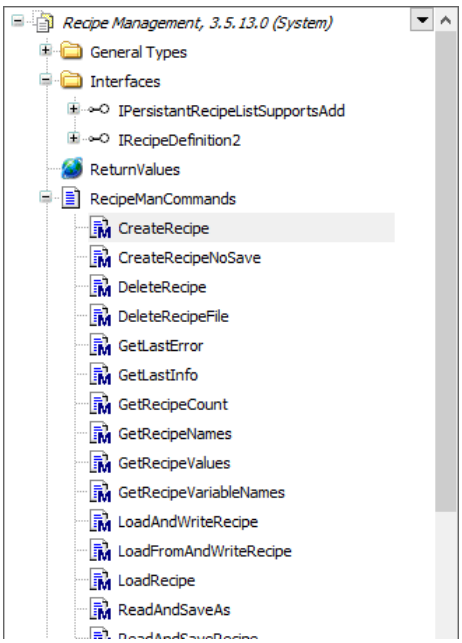
图表 3.3-65

其中不同配方②可以通过菜单栏“配方”下的选项添加，在“配方”菜单下，用户还可以进行删除原有配方，加载或保存配方文件等操作。

图表 3.3-66

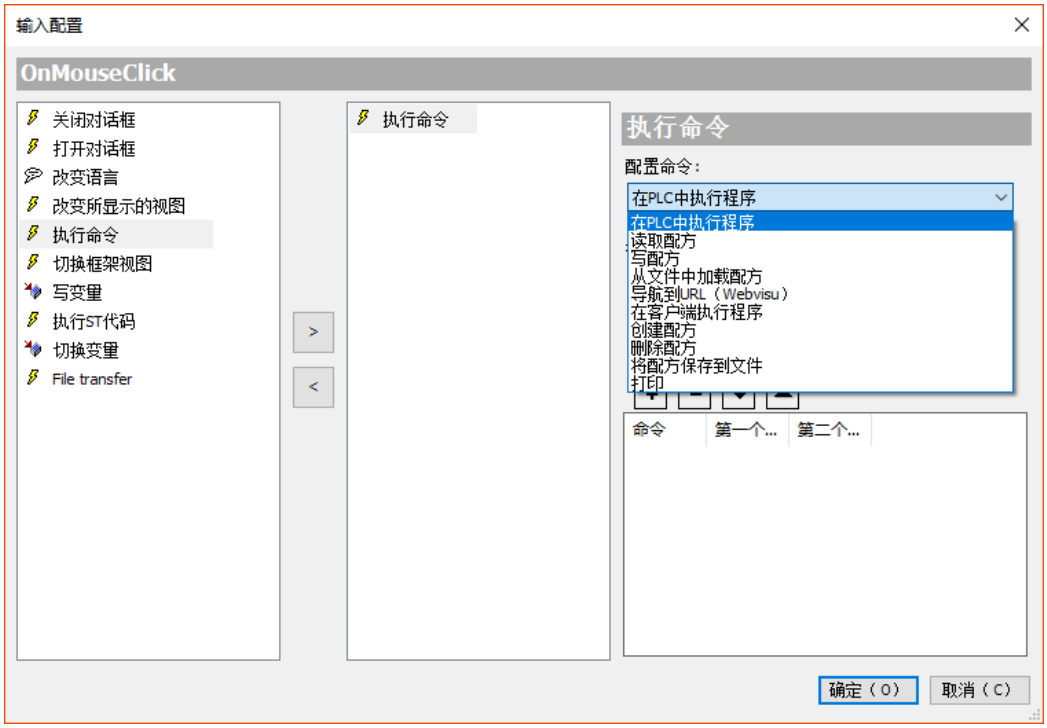
除了在 CODESYS 页面中直接编辑配方文件，用户也可以通过 CODESYS 提供“Recipe Management”库文件，对配方进行创建、删除、加载、保存、检查错误等操作。

图表 3.3-67



同样的 CODESYS 嵌入的 Visualization(可视化界面)同样也提供了配方的接口，在之后的可视化章节中会作详细说明：

图表 3.3-68



【例1】 创建两个配方分别生产草莓味面包和橙味面包

在程序中创建配方需要使用的变量：

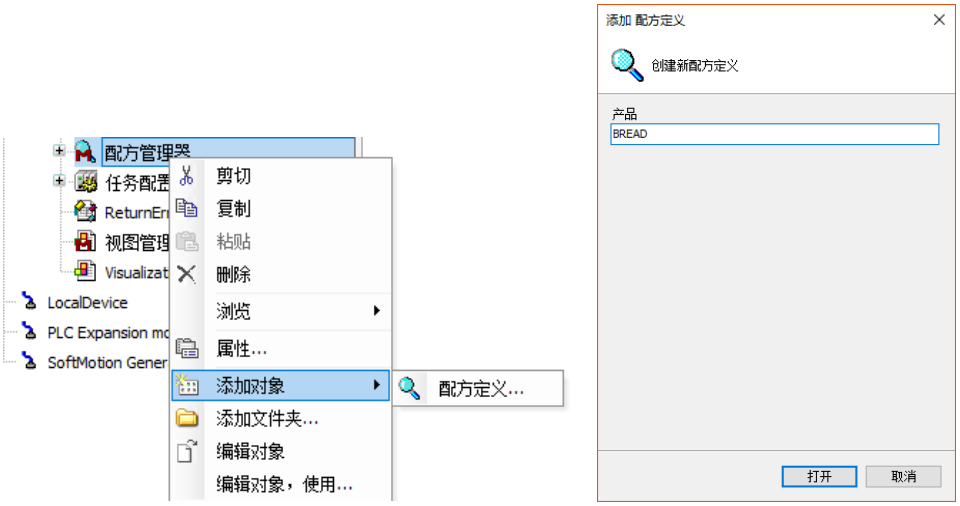
图表 3.3-69

PLC_PRG | BREAD | Product

```
1  PROGRAM PLC_PRG
2  VAR
3      egg:INT;              //鸡蛋
4      mlik:REAL;            //牛奶
5      flour:REAL;           //面粉
6      butter:REAL;          //黄油
7      sugar:REAL;           //砂糖
8      baking_time:TIME;     //烘焙时间
9      strawberry_jam:REAL;  //草莓果酱
10     orange_jam:REAL;      //橙子果酱
11     duration:TIME;        //持续时间
12     product1:Product;     //产品名称
13
14 END_VAR
```

添加新的配方定义：

图表 3.3-70



在配方中插入变量：

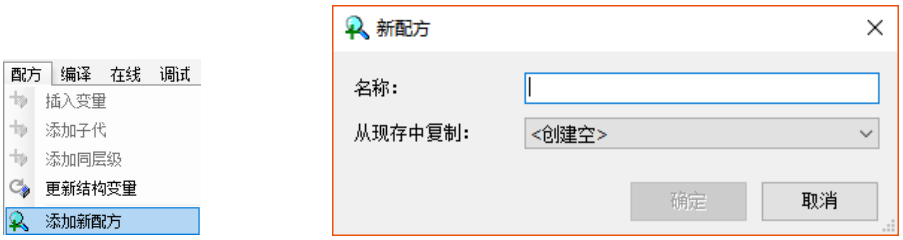
通过配方下的插入变量或者，配方定义中的 ... 完成配方中变量的添加：

图表 3.3-71

添加新配方：

选择菜单栏 “配方” 下的 “添加新配方”，在弹出对话框中输入 “Orange_Bread” 和”Starwberry_Bread” 两个新的配方

图表 3.3-72



在添加好的新配方中按照下图输入所需配方值：

需要注意的是，配方中的“变量”和“类型”是从程序中添加变量的时候自动加载的，“名称”、“注释”、“最小值”和“最大值”由用户在当前配方定义中自行添加，登录程序在线监控后，“当前值”会显示当前正在使用的配方数值

图表 3.3-73

BREAD | Visualization | Recipes

变量	类型	名称	注释	最小值	最大值	当前值	Orang...	Straw...
PLC_PRG.egg	INT	egg	g			0	50	40
PLC_PRG.flour	REAL	flour	g			0	200	250
PLC_PRG.mlik	REAL	牛奶	g			0	38.8	42.5
PLC_PRG.strawb...	REAL	草莓果酱	g			0	0	30.3
PLC_PRG.sugar	REAL	砂糖	g			0	25.5	20.8
PLC_PRG.butter	REAL	黄油	g			0	20	33.3
PLC_PRG.baking_...	TIME	烘焙时长	min			T#0ms	t#15m	t#15m
PLC_PRG.orange...	REAL	橙子果酱	g			0	22.2	0
PLC_PRG.duration	TIME	生产时长	min			T#0ms	t#15m	t#15m
PLC_PRG.product...	STRING	产品名称				''	'Orange...	'Strawb...
PLC_PRG.product...	DWORD	产品批次				0	1	1

在未登录状态下，用户在“配方定义”种选中自定义的配方数值区域，在菜单栏“配方”选项下可以对配方进行加载、删除和保存；在登录状态下，用户除了可以执行以上操作外，还可以读写配方

图表 3.3-74

对 Orange_Bread 执行写配方操作后，“当前值”会显示成写入的值

图表 3.3-75

BREAD | Visualization | Recipes

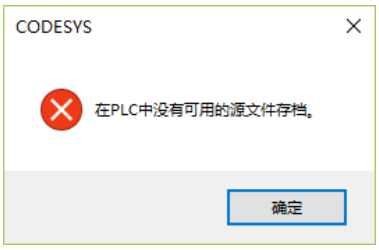
变量	类型	名称	注释	最小值	最大值	当前值	Orang...	Straw...
PLC_PRG.egg	INT	egg	g			50	50	40
PLC_PRG.flour	REAL	flour	g			200	200	250
PLC_PRG.mlik	REAL	牛奶	g			38.8	38.8	42.5
PLC_PRG.strawb...	REAL	草莓果酱	g			0	0	30.3
PLC_PRG.sugar	REAL	砂糖	g			25.5	25.5	20.8
PLC_PRG.butter	REAL	黄油	g			20	20	33.3
PLC_PRG.baking_...	TIME	烘焙时长	min			T#15m	t#15m	t#15m
PLC_PRG.orange...	REAL	橙子果酱	g			22.2	22.2	0
PLC_PRG.duration	TIME	生产时长	min			T#15m	t#15m	t#15m
PLC_PRG.product...	STRING	产品名称				'Orange_...	'Orange...	'Strawb...
PLC_PRG.product...	DWORD	产品批次				1	1	1

注意：配方定义中用户自行编辑输入的名称和注释都可以是中文，但是在配合配方接口程序使用的时候（配方接口程序包含在库文件“RecipeManagement”中），会出现乱码的现象，主要原因是 库文件“RecipeManagement”中提供的配方方法使用的变量数据类型为“String”只支持 ASCII 码，无法显示中文。例如方法：RecipeManCommand.GetRecipeVariableNames

3.3.6 源代码的上传和下载

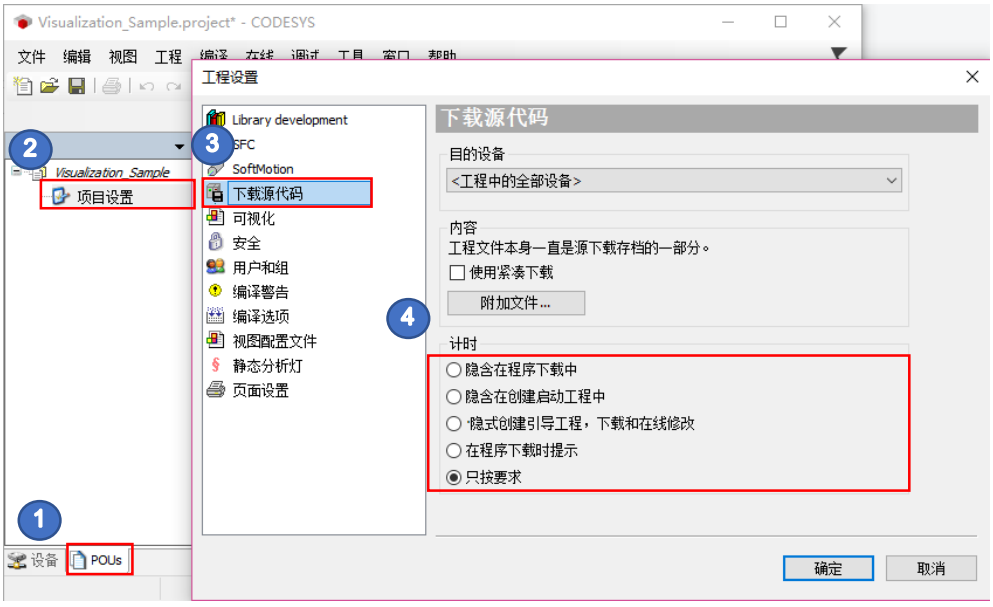
当用户对当前正在编辑的程序执行登录和运行操作时，CODESYS 默认不会将用户的源代码下载到目标设备中，此时用户直接从目标设备中上载程序会出现下图报错：

图表 3.3-76



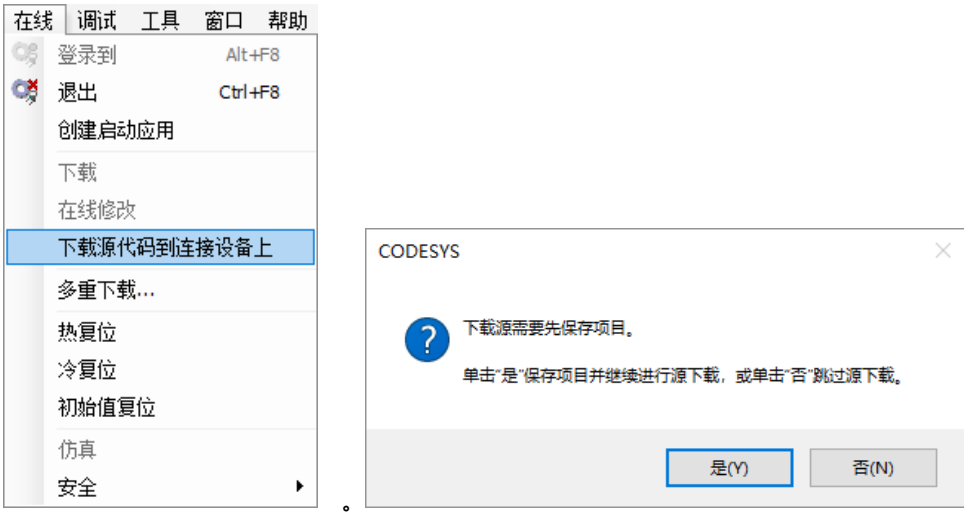
如果用户需要下载源代码，以便后续使用者进行上载和编辑等，需要在“项目设置”进行配置：

图表 3.3-77



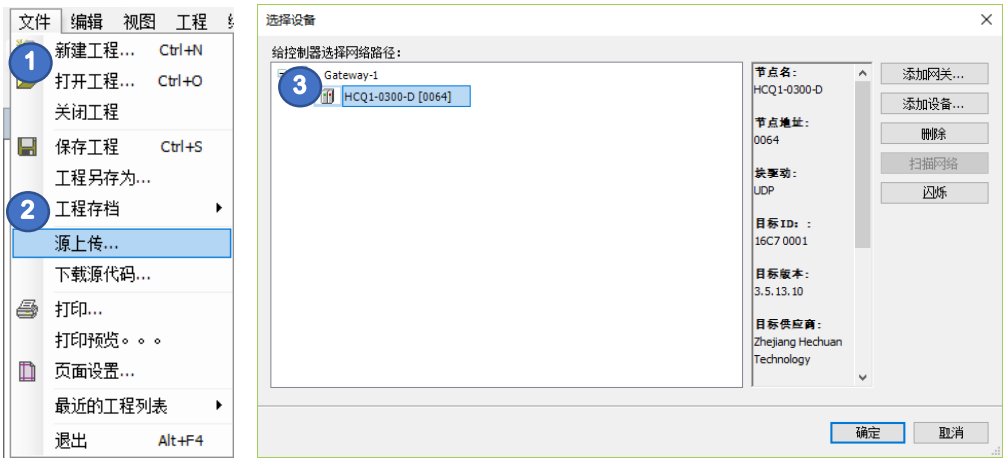
在步骤 4 中设置在哪种状态下进行源代码下载，设置完成后，单击“确定”保存配置。例如，选择“只按要求”后，在登录状态下，找到菜单栏“在线”→“下载源代码到连接设备”

图表 3.3-78



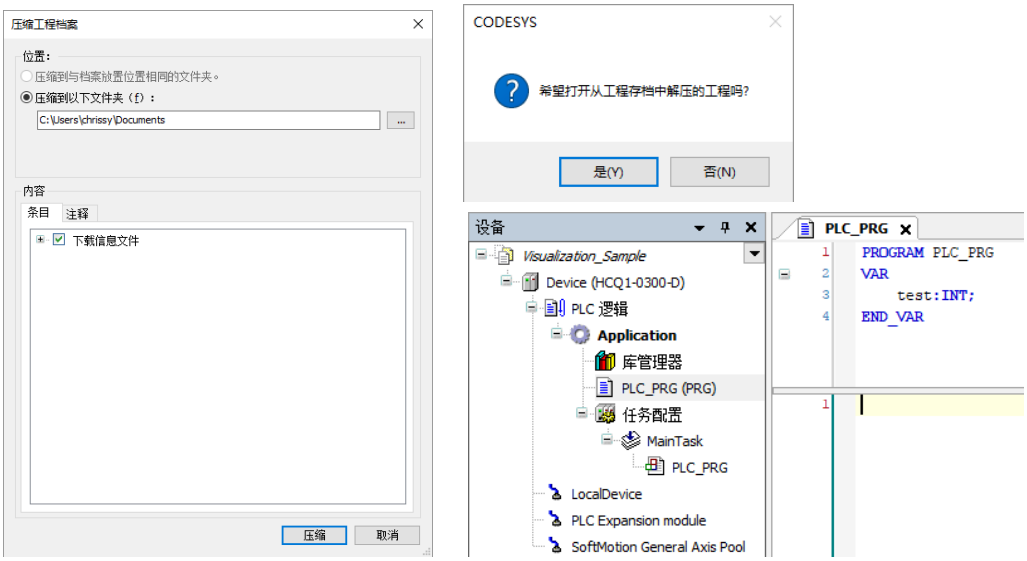
完成源代码下载后，用户后续可以成功的将源代码从控制器中上载：

图表 3.3-79



选择上载源代码后的目标路径，并打开，就可以看到加载上来的源代码：

图表 3.3-80

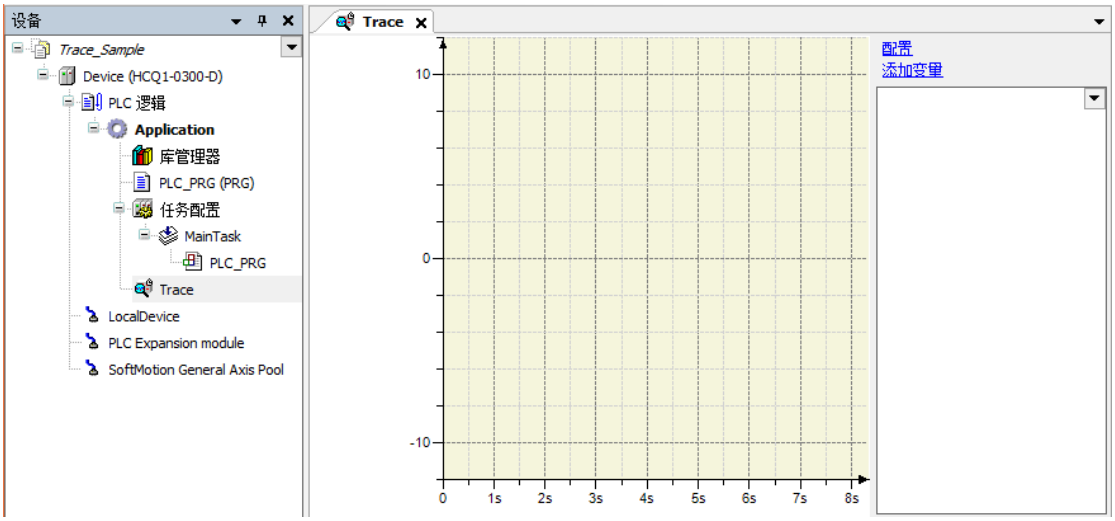


3.4 采样跟踪

采样追踪是 CODESYS 提供的图形化数据监控软件，类似示波器，在程序调试和诊断过程中，是非常实用和有效的工具。在程序运行过程中产生的数据都是转瞬即逝的，无法根据实时产生的数据进行分析，通过采样追踪，可以把程序执行过程中产生的过程数据全部记录下来，例如运动控制过程中的电机当前位置、速度、加速度等。通过对采集到的数据分析，就可以清楚的观察到系统运行的整个过程。

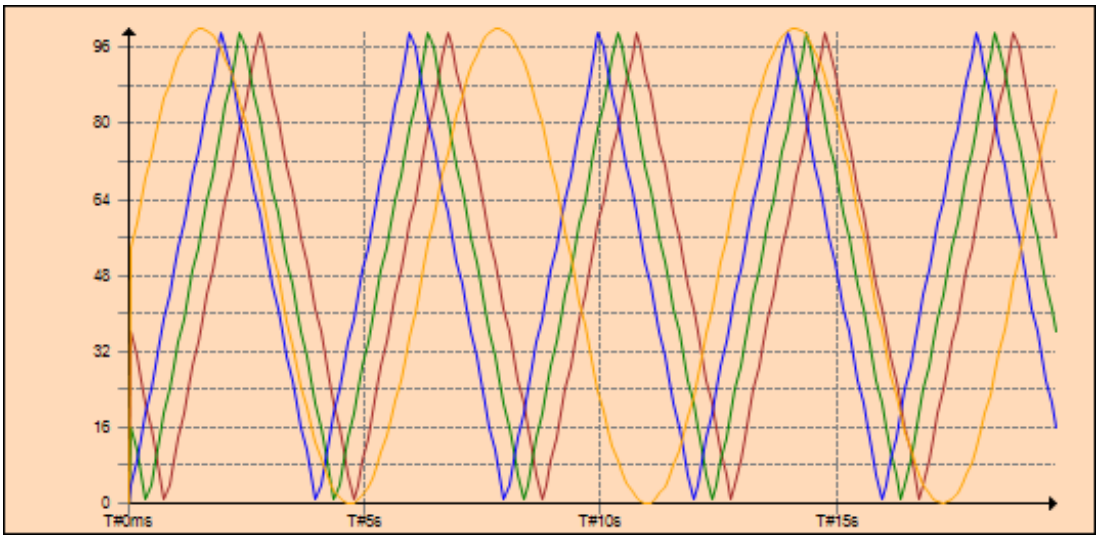
采样追踪提供“跟踪配置”和“跟踪对象”两个配置窗口，可以采集 PLC 运行过程中产生的数据波形，用户可以在同一个 PLC 应用下创建多个跟踪配置文件，如有需要用户也可以设置采样触发条件、采样周期以及对采样数据进行保存。

图表 3.4-1



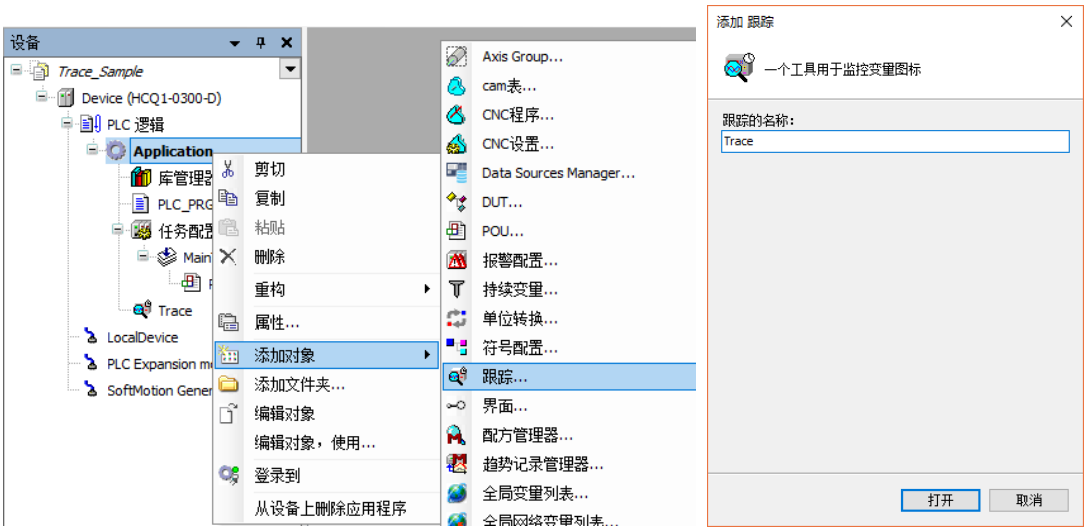
实时采样结果可能如下图：

图表 3.4-2



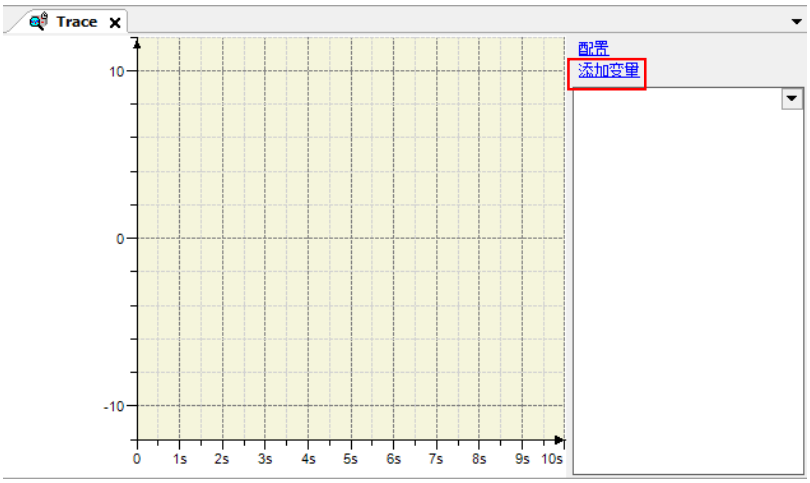
- 新建采样跟踪

  右击“Application”，选择“添加对象”→“跟踪…”输入跟踪名称后，点击“打开”确认



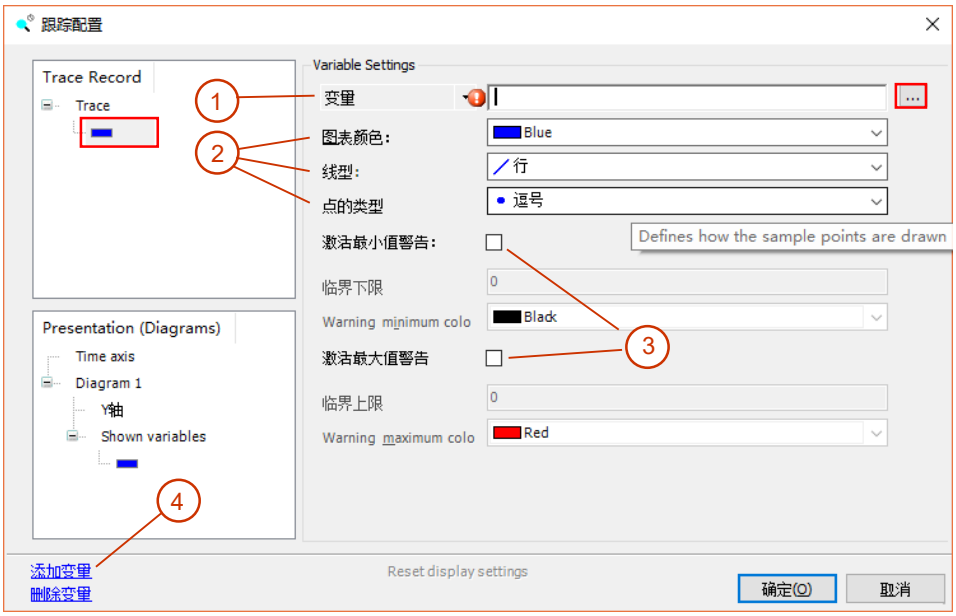
打开新建的跟踪后，选择界面右侧“添加变量”，进行跟踪变量的添加

图表 3.4-3



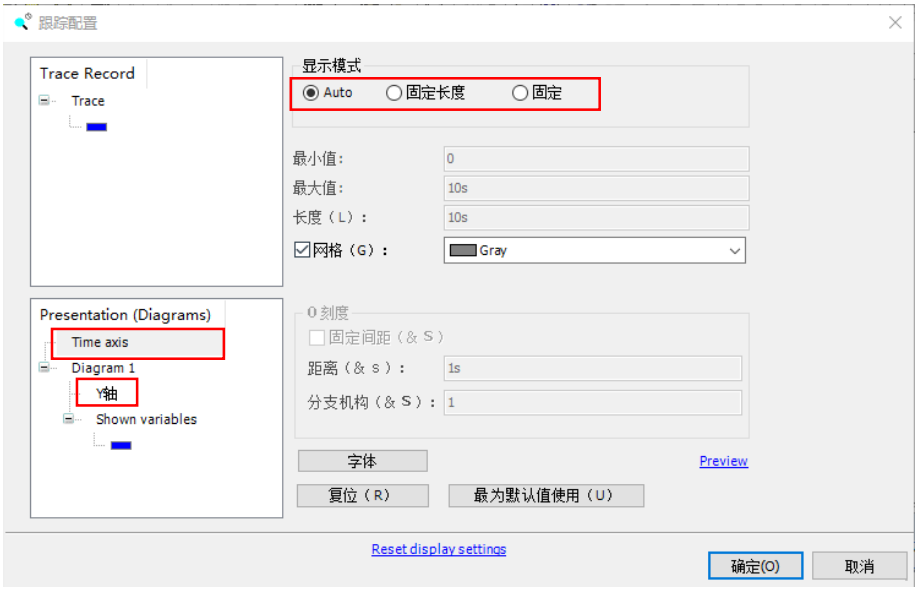
在“跟踪配置”对话框的 Trace Record 中，选择树形菜单 Trace 下的新添加的变量后，通过变量①右侧 … 进入“输入助手”添加程序中需要进行采样追踪的变量，通过②可以配置采样数据的变量颜色、线型和点的类型，通过③可以对采样数据激活最值警告，最后通过左下角④可以添加和删除在跟踪配置中的变量

图表 3.4-4



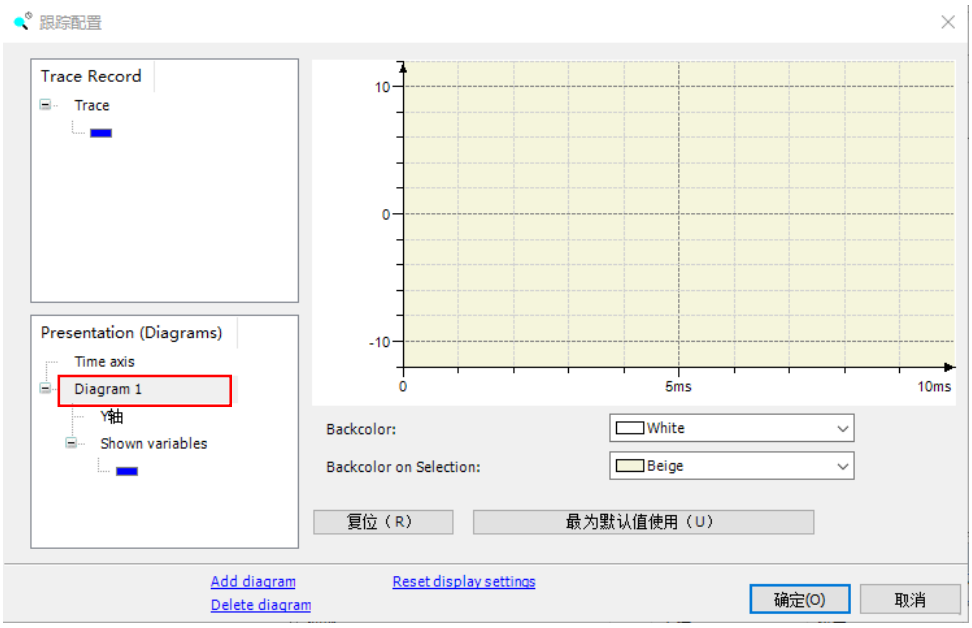
在“跟踪配置”对话框的 Presentation（Diagrams）中可以对追踪的背景和坐标轴进行设置，在“Time Axis”时间轴中可以配置显示模式为 Auto（自动）、固定长度、固定，设置成自动后，采样过程中，时间轴会自动调整；选择固定长度，则由用户自定义显示时长，采样窗口将只显示长度内的采样结果；最后，设置成固定，将会由用户指定显示采样过程中某一段的曲线。在同一页面中还可以对字体、网格等进行配置，选择 Y 轴之后可以观察到类似的页面，用户按照需求进行配置即可。

图表 3.4-5



在树形菜单“Diagram1”中则可以配置追踪背景

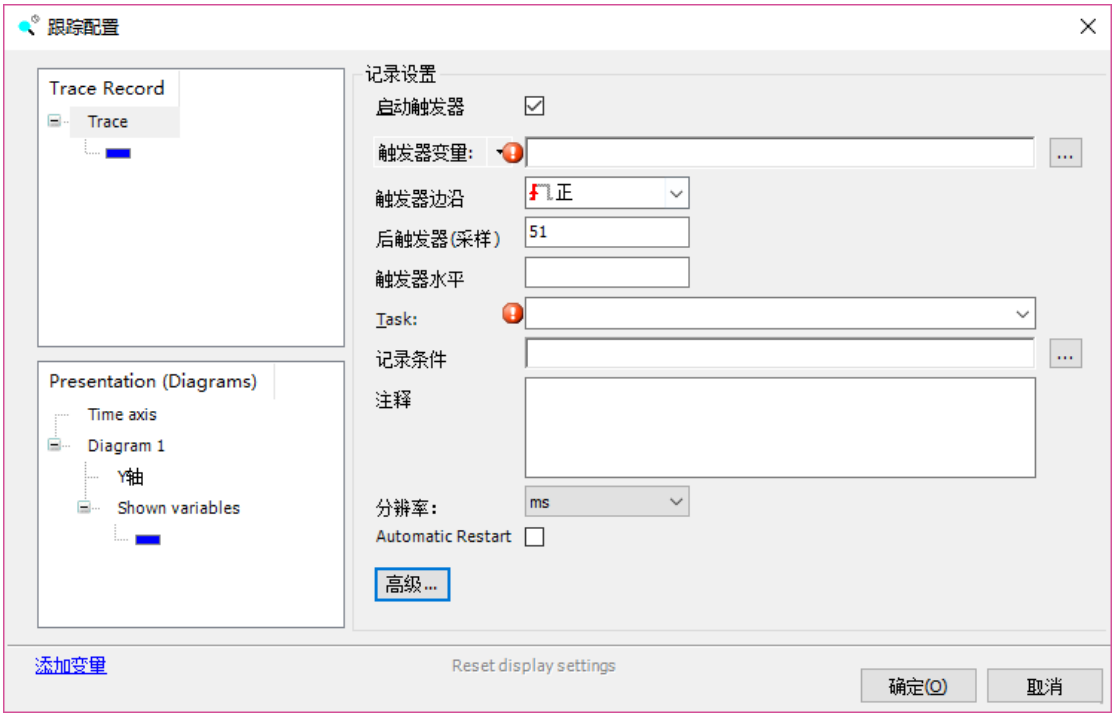
图表 3.4-6

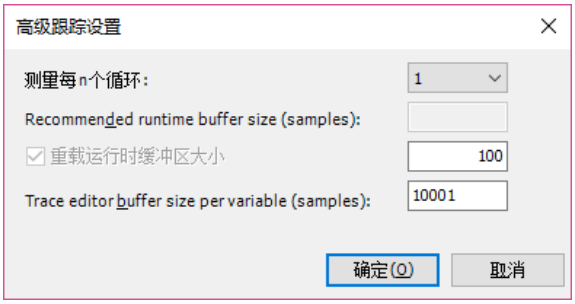


- 触发采样追踪

CODESYS 默认提供的采样追踪功能支持不同方式的触发采样，在树形菜单“Trace”中可以进行配置，触发采样这个功能是可选的，用户在没有配置这个功能的情况下，需要手动触发采样，只有在启动触发器打勾后，触发采样才会按照配置生效：

图表 3.4-7





1. 触发器变量

触发变量可以是一个 BOOL 类型的变量，表达式或者模拟变量、枚举变量等。当该变量满足了定义的值，该值根据“触发器边沿”类型来决定，一旦“触发器变量”为 True 或者满足某一特定值后，跟踪会根据设定的时间进行采样。

2. 触发器边沿

正：上升沿或数字信号大于“触发器水平”值时触发

负：下降沿或数字信号大于“触发器水平”值时触发

都：边界触发，上升沿或者下降沿触发

3. 后触发器（采样）

触发事件发生后，在这里可以设置每个跟踪记录的数量，默认值 50，范围从 0 直到（$2^{32}-1$）

4. 触发器水平

当使用模拟量作为触发变量，在此处定义该变量为多少时触发采样

5. 任务

从可获得的任务中选择一个任务用于定义当前输入信号的触发任务

6. 记录条件

在这里定义一个 BOOL 变量、表达式、一个数值或者是属性，如果条件为真，则启动跟踪采样，如果在记录条件中没有输入，则在下载跟踪配置并且应用开始运行后，立即开始跟踪记录

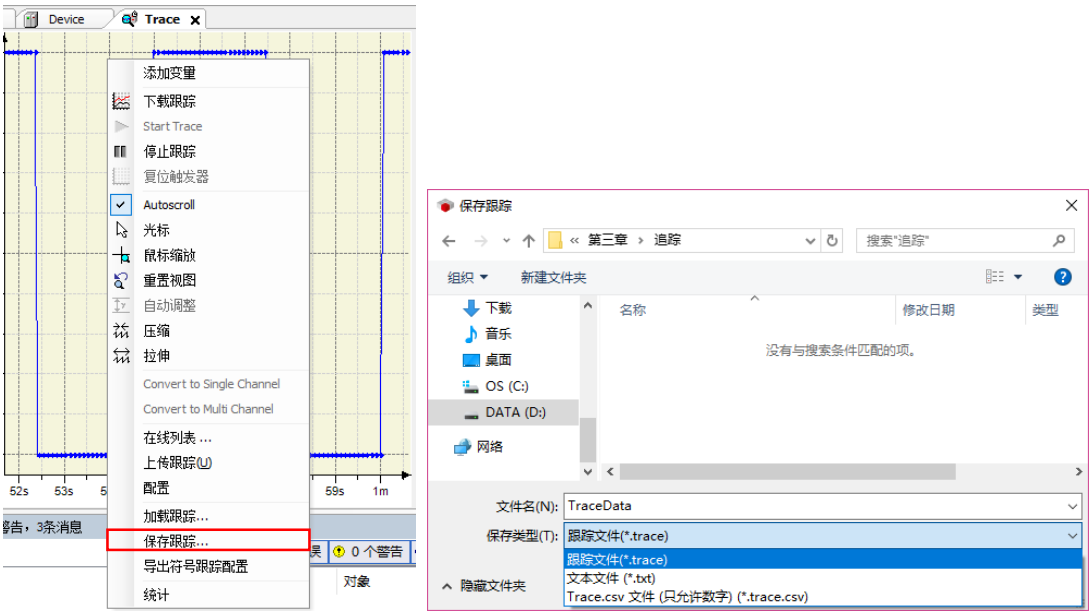
7. 注释

对当前记录的一个注释文本

- 数据的保存

当完成 PLC 数据的采集后，用户需要对数据进行本地保存，以便后续对数据进行分析，可以直接在采样追踪界面右击选择“保存追踪”，在弹出对话框“保存跟踪”中选择保存路径和给定文件名称后，可以选择保存的格式有“.trace”、“.txt”、“.trace.csv”，其中“.trace”文件可以用 CODESYS 打开，而“.txt”和“.trace.csv”保存出来的文件可以直接用 Excel 打开。

图表 3.4-8



8. 高级跟踪设置

在高级跟踪设置对话框中，可以在运行系统中定义跟踪元素的跟踪缓存区大小。

- 跟踪常用功能

图表 3.4-9

图标	说明
	下载跟踪，当用户没有设定触发采样时，下载跟踪会直接开始采集跟踪曲线
	启动或停止跟踪，用于启动或者停止当前跟踪
	复位触发器，在一个触发事件发生后，重置触发器，充值后，跟踪将重新由触发事件启动并记录最新值
	光标，用于确定数值的 X 轴坐标值，一个采样追踪中可以添加两个光标，用来确定每个光标单独的 X 轴绝对位置及两个光标之间的 X 轴相对位置
	鼠标缩放，用于激活鼠标缩放模式，当该模式被激活后，可以在跟踪窗口中画一个矩形重新定义跟踪曲线的显示区域，该区域会扩大 滚动鼠标来缩放坐标系 X-Y 轴，使用数字键盘键<->><+>可以实现相同的功能 摁住<shift>键的同时，滚动鼠标滚轮，只缩放 X 轴 摁住<ctrl>键的同时，滚动鼠标滚轮，只缩放 Y 轴
	重置视图，在用户自定义了跟踪配置的外观后，使用该功能可以恢复默认的外观设置，在配置对话框中可以对记录的默认设置进行定义

	自动调整，用户在自定义了跟踪配置的外观后，使用该功能可以使跟踪自动调整外观到合适视图
	压缩，使用该指令能在一个更大的时间短内观察跟踪变量的进展，可实现命令的多重执行
	拉伸，使用该指令可以拉伸显示的采样跟踪的值

3.5 持续变量

在 PLC 系统实际运行过程当中，因为工业现场的实际情况，要求系统关机或异常断电以后，PLC 系统中的某些变量值能够保存在 PLC 内部，在下一次用户正常上电开机后，可以被调用，并保持断电前的数据继续被程序使用。CODESYS 提供了两种保持型变量类型，分别是 RETAIN 和 PERSISTENT RETAIN 这两种，后者在实际应用中使用更为频繁。

① 新建持续变量

持续变量的

② 持续变量使用

第4章 Visualization 可视化界面编辑

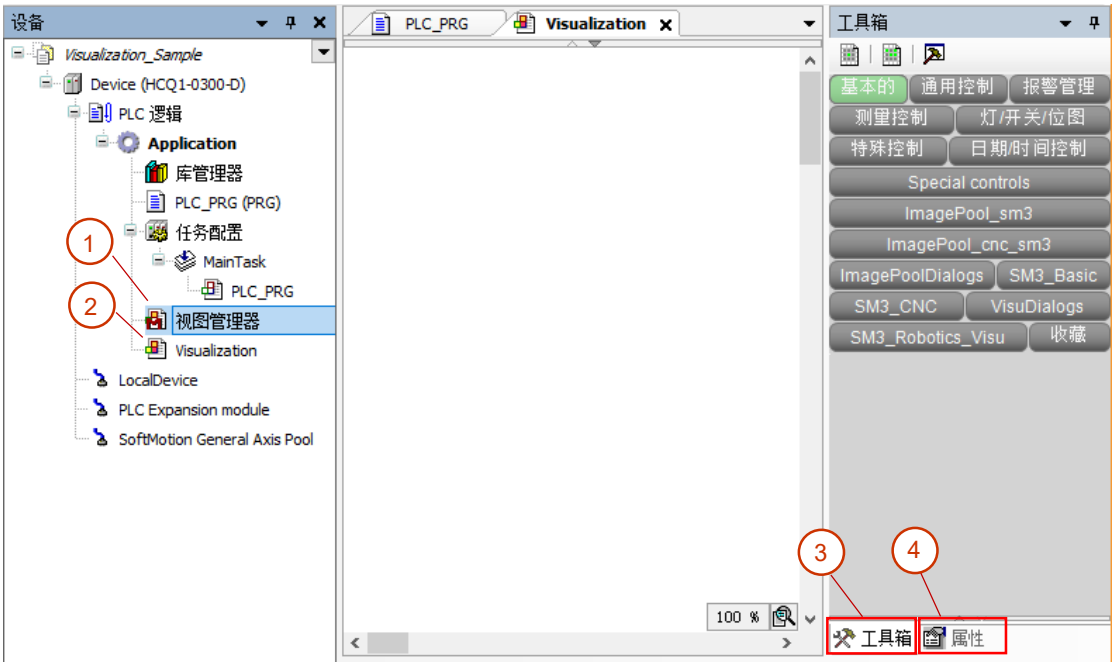
4.1 可视化项目简介

当用户在编辑复杂控制逻辑或者提供调试界面给最终用户的时候，复杂的逻辑程序对于使用者而言并不友好，即使编写程序的工程师提供了完整的注释文字化的程序给人的感受也非常不直观，为了方便用户调试程序以及方便供应商提供给最终用户的调试页面更为友好，方便调试，用户可以选择给 PLC 程序编写可视化界面。

可视化对象可以在对象管理器的“可视化界面”中进行管理，它包含可视化元件的管理并且对不同的对象可以根据个人需要进行管理。一个标准的“Application”下可以创建多个可视化对象，不同的可视化对象之间可以进行通讯。

通过图形编辑器，用户也可以把程序内部变量的数值实时的显示在界面中，通过这些变量在图形中的变化，能直观的观察当前程序的运行状态。

图表 4.1-1



① 视图管理器，可以对可视化相关参数进行配置

② 可视化编辑器，可以在这个页面中编辑需要的可视化界面

③ 工具箱，提供可视化编辑器页面下的各种图形

④ 属性，编辑可视化元素相关属性

Q1 的可视化还提供了不同的客户端其中包括;

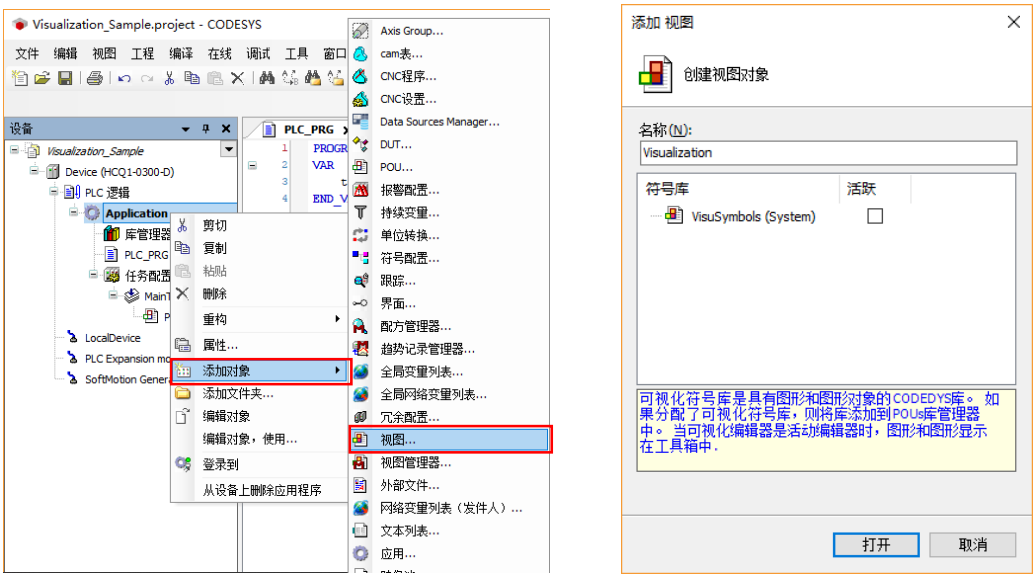
WebVisu：通过网页服务器链接到相关的应用程序

TargetVisu：用来在外部设备中运行可视化应用

4.2 新建视图

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。用户可以通过在树形菜单“Application”右击找到“添加对象”选择“视图”，给定视图对象名称后，单击“打开”添加新的可视化，新建的可视化包括一个可视化编辑器和视图管理器。

图表 4.2-1



当第一个可视化对象加入到工程中，相应的可视化库将根据目前状态被自动添加到库管理器中配置。

图表 4.2-2

可视化库总是被设置为"占位符函数库"的特殊类型，表示要使用该库的确切版本没有得到解决，只有工程中包含了库才可以。只有这样，当前激活的配置文件才能确定使用哪个版本。请注意，这种类型的库是不同于设备特定的“占位符函数库”，因为占位符可以通过设备描述来解决。当在一个标准工程中添加可视化对象时，默认包含的基本库有 VisuElems，VisuElemMeter，VisuElemWinControls，VisuElemTrace，VisuInputs，这些基本库中还另外包含有更多的库。

注意：

可视化库文件不需要用户手动添加，一般默认会在用户第一次在应用中创建视图的时候自动添加，也不会很明显的使用到这些库文件。

4.3 视图基本操作

在可视化编辑器中用户可以执行的基本操作包括添加可视化元素，对齐，删除等

4.3.1 添加视图元素

可以通过“编辑区直接绘制视图元素”及“拖拉视图元素到编辑区”两种方式都可以添加视图元素

- 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选择要放置的位置，视图元素会按照指定的大小和位置添加到画面编辑中。

- 拖拉视图元素到编辑区

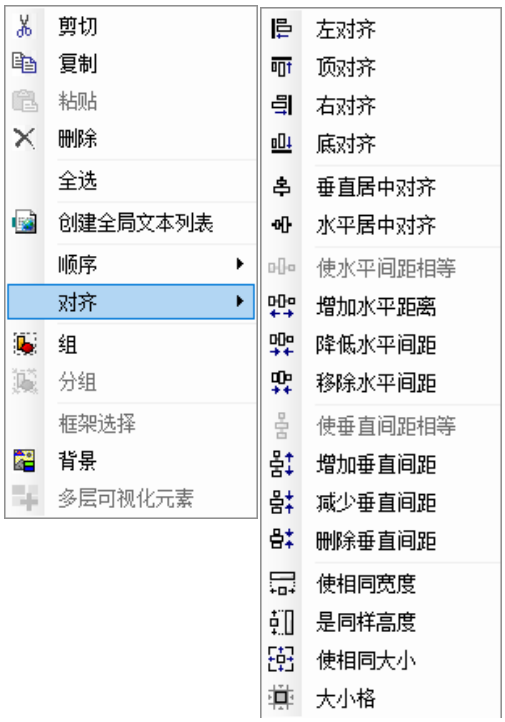
  在工具箱中选择要添加的视图元素并将其拖拽至画面编辑区上，视图元素会按照默认大小添加到画面编辑区指定区域。

4.3.2 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：

图表 4.3-1

图表 4.3-2



- 左对齐：将选定的视图元素沿它们的左边对齐
- 顶对齐：将选定的视图元素沿它们的上边对齐
- 右对齐：将选定的视图元素沿它们的右边对齐
- 底对齐：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
- 水平居中对齐：将选定的视图元素水平居中对齐
- 使水平间距相等：使选定的视图元素水平间距相等
- 增加水平距离：增加选定视图元素的水平间距
- 降低水平间距：减少选定视图元素的水平间距
- 移除水平间距：移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：使选定的视图元素垂直间距相等
- 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：减少选定的视图元素垂直间距
- 删除垂直间距：删除选定的视图元素垂直间距
- 使相同宽度：使选定的视图元素宽度相等
- 使同样高度：使选定的视图元素高度相等
- 使相同大小：使选定的视图元素大小相等

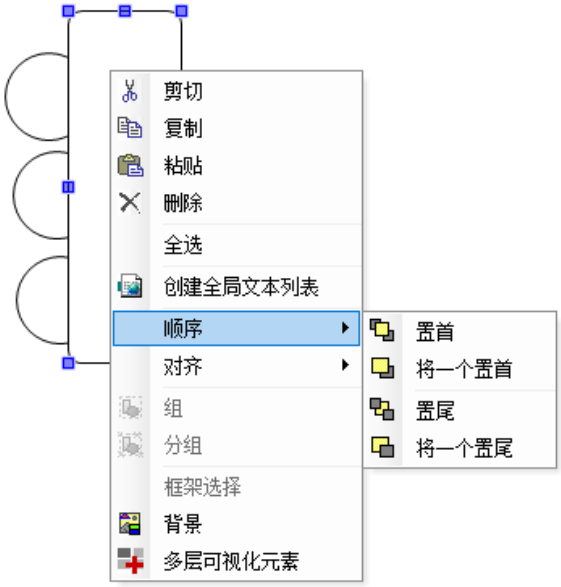
4.3.3 删除视图元素

删除视图元素的方法很简单，用户可以直接选中需要删除的视图元素，单击鼠标右键，在弹出的快捷菜单中选择“删除”，或者直接选中，摁下键盘上的<Delete>即可删除

4.3.4 更改视图顺序

用户按照顺序添加了部分视图元素后，如果需要组合成图形，就可能会将多个图形进行堆叠，堆叠的过程中，后续添加的图形默认会遮挡更早创建的图形，当用户需要更改堆叠顺序时，可以直接在可视化编辑窗口中选中需要更改堆叠顺序的图形，右击，在弹出菜单中找到“顺序”后选择需要执行的动作即可，如下：

图表 4.3-3

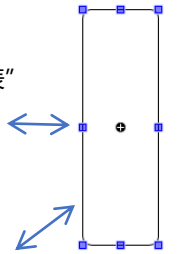


置首：将选中的视图视图元素置于画面最上层
讲一个置首：将选中的视图视图元素在画面中上移一层
置尾：将选中的视图视图元素置于画面最最底层
讲一个置尾：将选中的视图视图元素在画面中下移一层

4.3.5 调整视图元素大小和位置

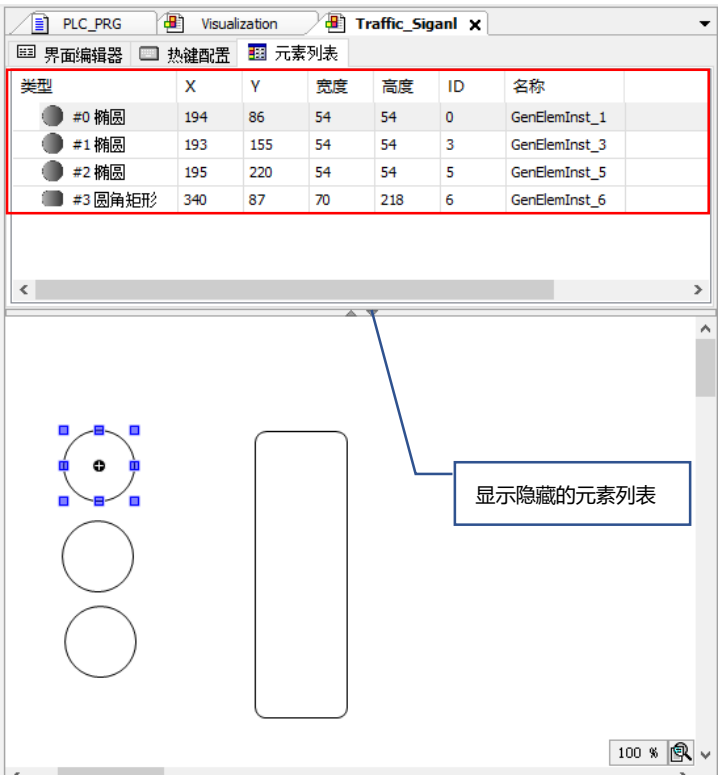
对于已经添加的视图元素，用户如果需要调整视图元素的大小，只需要在选中视图元素后通过元素周围提供的调整框直接调整到合适大小即可。

图表 4.3-4



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。

图表 4.3-5



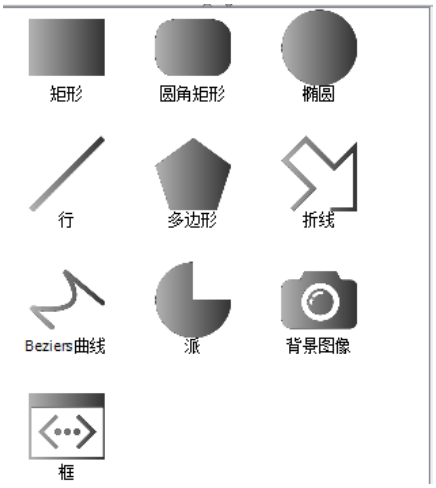
4.4 工具箱

工具箱，同可视化编辑器一起使用，提供可嵌入到可视化窗口的可视化元素。

4.4.1 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。

图表 4.4-1



文本输入框/文本显示框

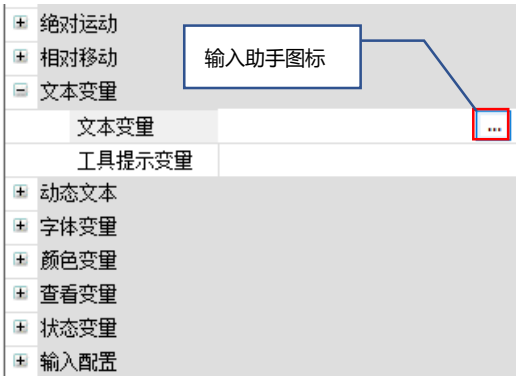
在 CODESYS 的视图中，没有独立的文本输入框/文本显示框，如果需要制作文本或者数值的显示/输入框，那么可以通过添加“矩形/圆角矩形/椭圆”框来实现。

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。

图表 4.4-2



接下去需要配置显示类型，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。

图表 4.4-3

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：

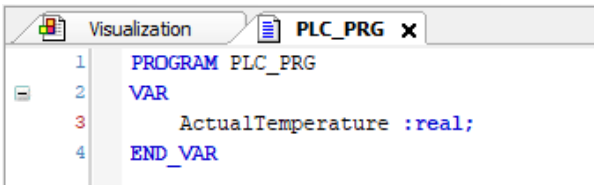
如果想显示一个百分号（%）并与上面提到的格式化输出命令进行组合，则必须输入“%%”，如输入“Rate in %%: %s”，则在线模式下，将显示“Rate in %: 12”（关联文本显示的变量当前值为“12”）。其中，输入的格式化输出命令区分大小写，需为小写，否则无法正常识别。

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。

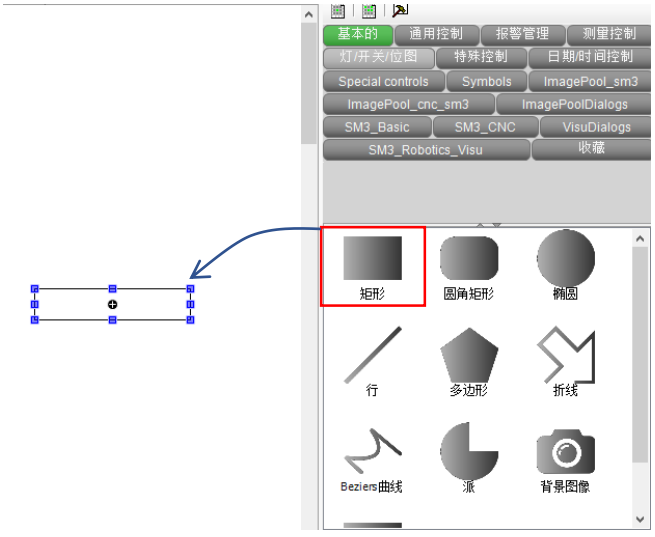
在程序中添加需要显示的温度变量

图表 4.4-4



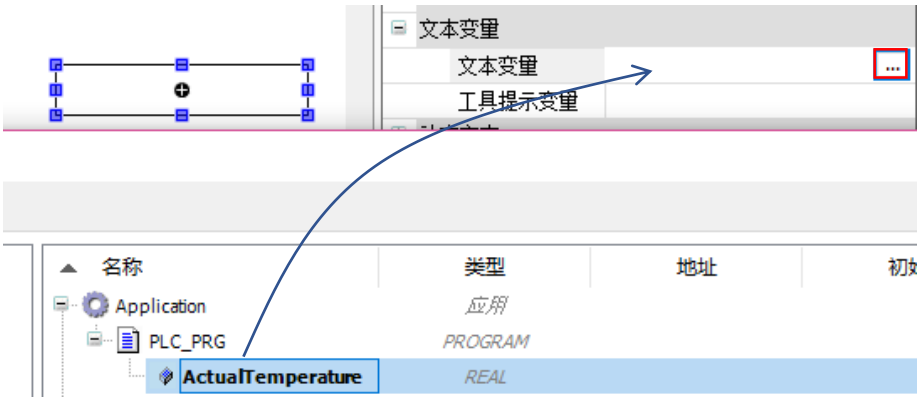
首先从工具箱的“基本的”中选择矩形框拖拽到视图编辑区

图表 4.4-5



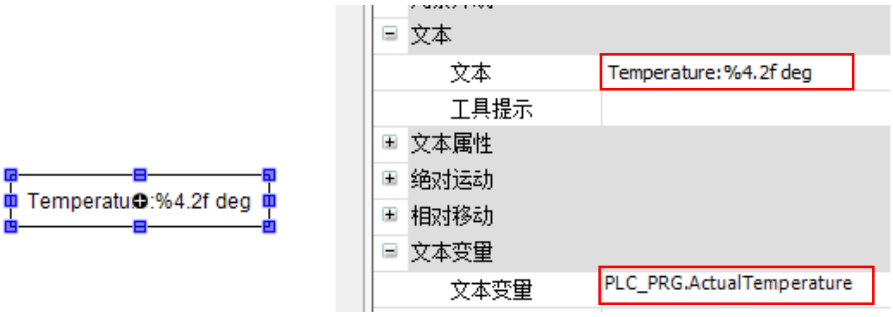
选中矩形框后，在右侧“属性”菜单中找到“文本变量”完成变量映射

图表 4.4-6



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。

图表 4.4-7



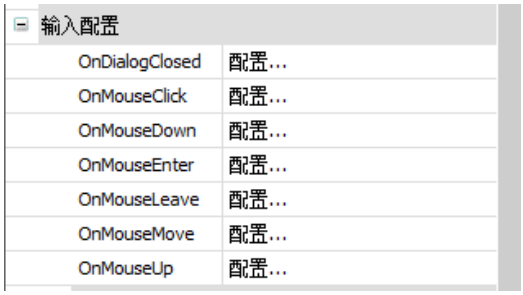
在线运行结果如下：

图表 4.4-8

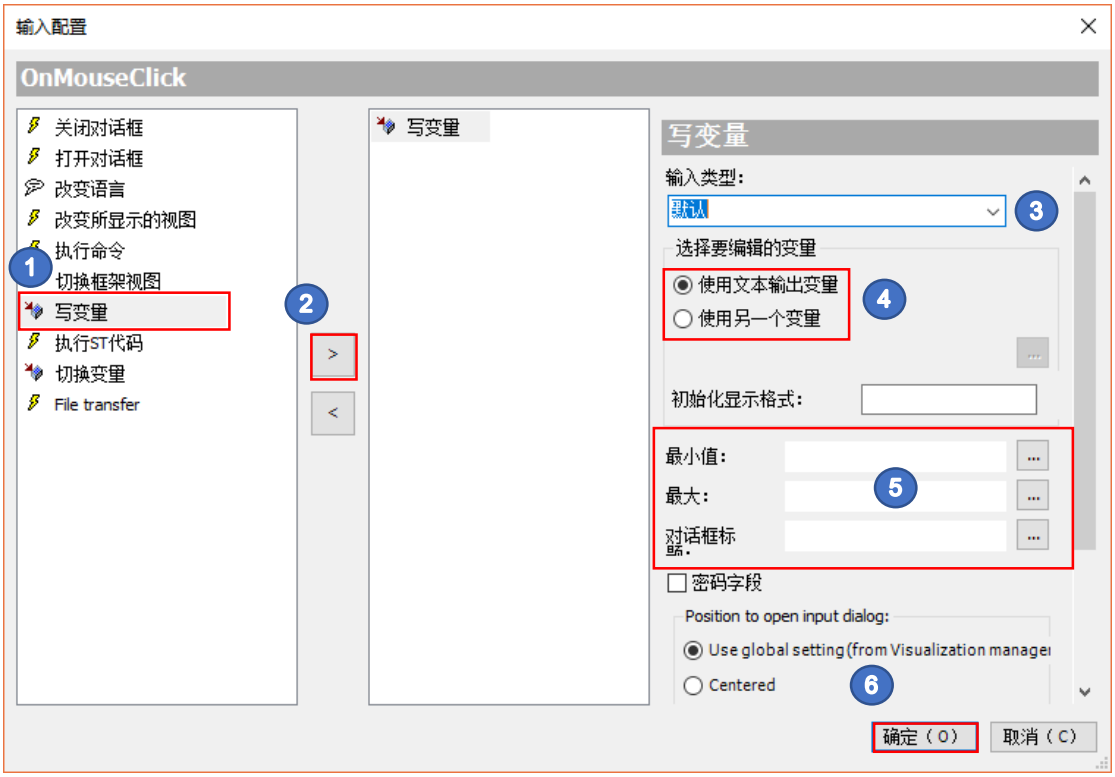
2. 配置“文本输入框”

文本输入需要首先实现文本显示，否则用户无法在可视化界面中，查看输入的变量，所以，同样需要先设置“变量映射”和“显示类型”，此外，还需要额外增加一个步骤，即事件触发设置，在属性的“输入配置”中单击“配置…”进行事件触发设置。

图表 4.4-9



单击“配置...”后，系统自动弹出“输入配置”对话框，按照如下步骤进行配置。



① 选择触发类型，输入变量是需要对变量进行写操作，选择“写变量”

② 通过图中向右箭头添加至右侧

③ 设置输入类型，可以直接使用键盘输入或者可以通过虚拟全键盘及虚拟数字小键盘等

图表 4.4-10

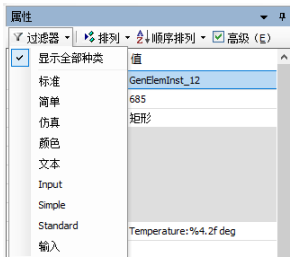
④ 选择相关联的输入变量，可以是当前输出显示的变量，也可以重新关联其他变量，决定文本框中显示内容的

⑤ 处于设备安全因素的考虑，设计人员可以在此设置输入值的上限值/下限值

⑥ 设置完成后，单击“确认”完成配置

注意：

当用户选择矩形框，在右侧属性菜单无法显示全部配置页面，例如用户在“输入配置”下无法找到“OnMouseClick”此时用户需要在树形菜单嘴顶端的“过滤器”中选择“显示全部种类”即可



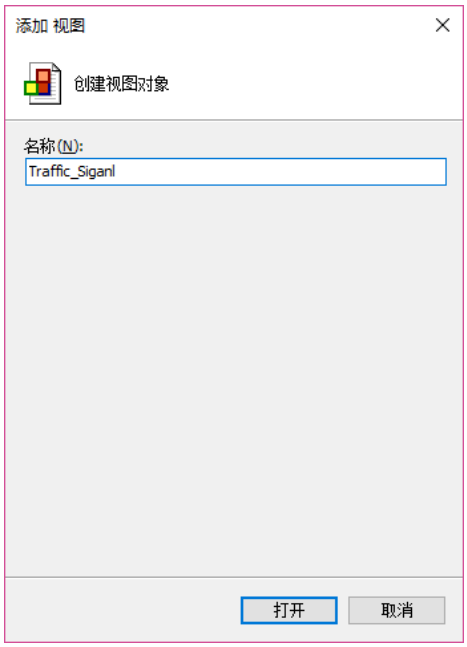
颜色显示框

基本工具不仅能作为文本输入及显示使用，还可以设计成简单的颜色指示框作为指示灯使用。在这种用法下当指示框对应的程序中变量编程 ON 后，用户可以通过修改颜色指示框的报警填充颜色从而实现颜色指示框的变色效果。

【例2】 创建一个颜色显示框，来模拟十字路口交通灯的状态，并且每 2s 更换一次指示灯颜色。

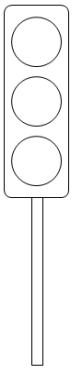
首先创建一个可视化界面，并命名为“Traffic_Signal”

图表 4.4-11



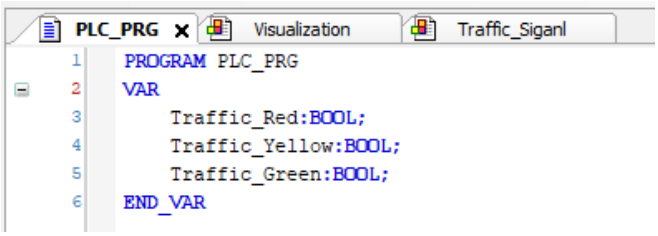
使用基本视图元素创建交通灯图形，其中会使用到“矩形”、“圆角矩形”和“椭圆”等视图元素，按照一定的顺序进行排列如右侧。

图表 4.4-12



在程序中编辑颜色变量

图表 4.4-13



通过基本视图元素绘制交通灯图形后，选中所有控件，并右击，选择“组”将图形作为一个整体方便移动和调整大小。

图表 4.4-14

完成变量映射

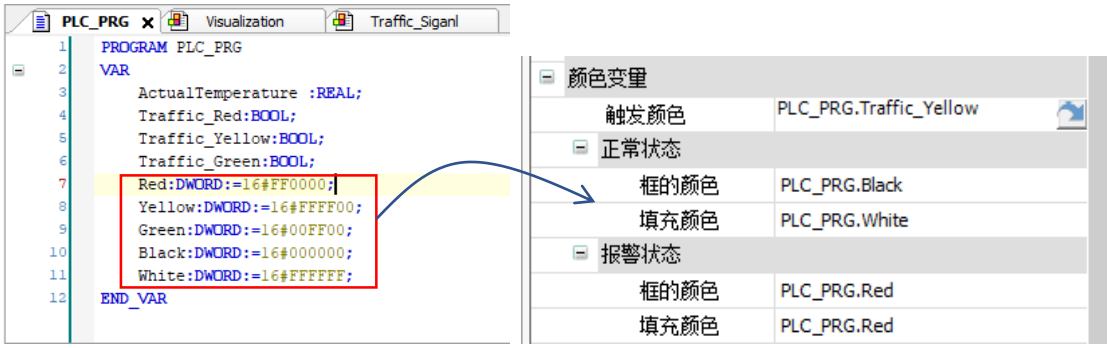
设置属性菜单中的“颜色变量”，单击“输入助手”图标进行相关变量的映射，设置完成后，当颜色变量状态改变时，会触发显示报警状态下的框和填充颜色，从而达到状态指示灯的效果

图表 4.4-15

设置指示灯在 ON 状态下的颜色

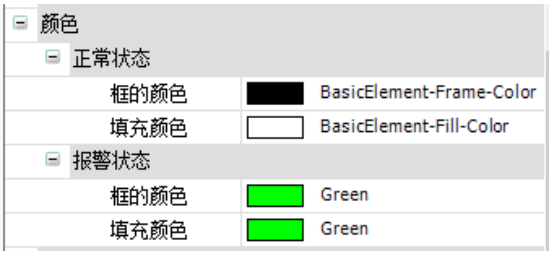
如果用户需要通过在程序中设置的变量来修改指示灯在不同状态下的颜色，则可以在上图中直接完成变量映射，完成后，当颜色变量状态改变时，将根据设置的变量数值（对应颜色 RGB 值，默认为 16 进制）来给定具体的颜色，这样的方式，用户可以更自由的给定具体的颜色数值。

图表 4.4-16



当然用户也可以选择更为简单的方式来设定，通过在属性菜单中找到“颜色”，直接给定颜色也可以完成设置。

图表 4.4-17



当用户既设置了属性菜单下的“颜色”同时也在“颜色变量”中给定了颜色的 RGB 变量值，那最终在线显示效果以“颜色变量”中给定的具体的 RGB 值为准。

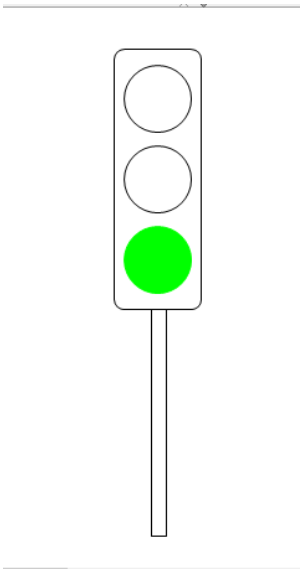
编写示例程序：

图表 4.4-18

```
Ton1(IN:= NOT Ton1.Q , PT:=T#2S ,);
CASE Index OF
1:
Traffic_Red:=TRUE;
Traffic_Yellow:=FALSE;
Traffic_Green:=FALSE;
IF ton1.Q THEN
    Index:=2;
END_IF
2:
Traffic_Yellow:=TRUE;
Traffic_Red:=FALSE;
Traffic_Green:=FALSE;
IF ton1.Q THEN
    Index:=3;
END_IF
3:
Traffic_Red:=FALSE;
Traffic_Yellow:=FALSE;
Traffic_Green:=TRUE;
IF ton1.Q THEN
    Index:=1;
END_IF
END_CASE
```

在线运行结果如下：（其中某一个指示灯亮起的情况）

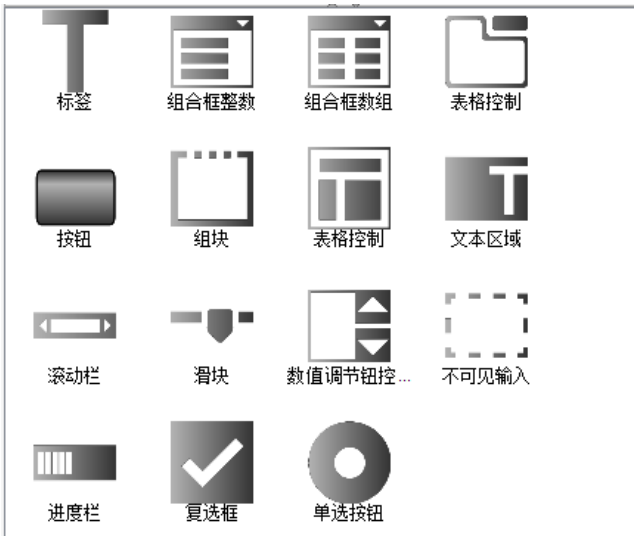
图表 4.4-19



4.4.2 通用控制工具

通用控制工具主要包括一些常用的图形制作元素，可以利用这些工具制作标签、组合框、制作表格和按钮等，下面会对这些视图元素逐一说明。

图表 4.4-20

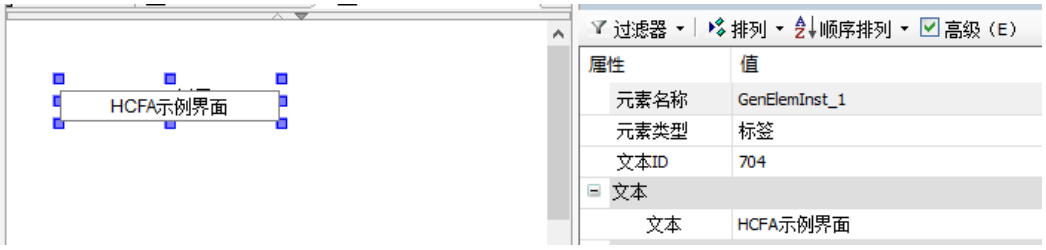


标签（Label）

标签控件主要用于显示用户不能编辑的文本，标识窗体上的对象（如给文本框、列表等添加描述信息）。

用户在“工具箱”找到通用控制后，选择标签  即可进行添加。在可视化界面中添加了标签后，只需要直接在标签的属性面板中设置“文本”属性，用户在“文本”中输入的内容将会直接显示在可视化界面中，并且在登录后标签的内容不可更改。

图表 4.4-21



在工具箱提供的很多控件中，用户可以找到“状态变量”→“不可见”这个选项，通过这个选项用户可以添加状态变量，来显示或者隐藏控件，如果状态变量映射的变量为“TRUE”则当前控件不可见；如果状态变量映射的变量为“FALSE”则当前控件可见，

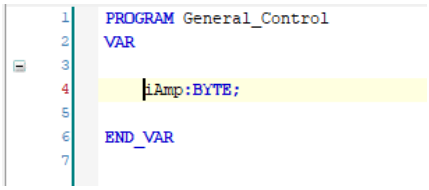
组合框整数（ComboBoxInteger）

组合框分为组合框整数和组合框表格两种，其中组合框整数允许用户根据下拉菜单选择想要写入的整型数据，该数据最终会被写入到变量中。

■ 建立整型数据

新建 POU 程序命名为 General_Control，并编写示例程序如下：

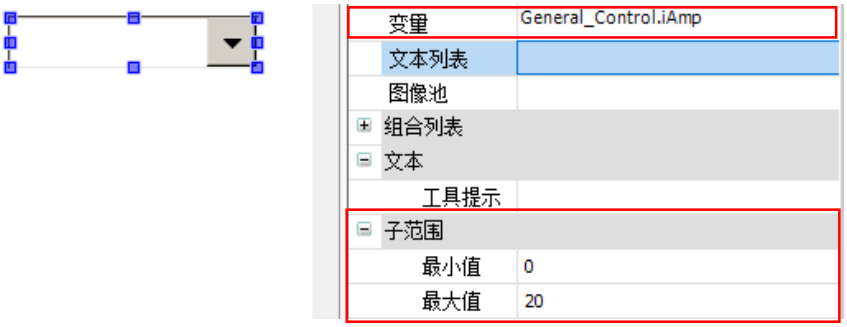
图表 4.4-22



■ 创建组合框整数

在可视化编辑窗口中打开工具箱，找到“通用控制”添加其中的  组合框整数，通过设置属性菜单中的“变量”，完成和程序中新建变量之间的映射后，组合框整数即可使用，当然用户还可以通过“子范围”中提供的“最大值”和“最小值”设置输入数据的范围。

图表 4.4-23



在线运行结果如下：

图表 4.4-24

组合框数组（ComboBoxTable）

组合框数组和组合框整数一样，允许用户进行下拉选择，但是组合框表格允许用户选择数组列表中的数据对象作为可选对象。

【例3】 创建一个二维数组，通过在可视化界面中选择相应的行，将该行数据写入到程序当中。

在程序中建立数组数据，并对这个二维数组变量赋初始值，如下：

图表 4.4-25

```
    iFactor:BYTE;
    arrFactor: ARRAY[0..2, 0..4] OF STRING := ['BMW', 'Audi', 'Mercedes',
 'VM', 'Fiat', 4('150'), '100', 'blau', 'grau', 'silber', 'bronze', 'rot'];
END_VAR
```

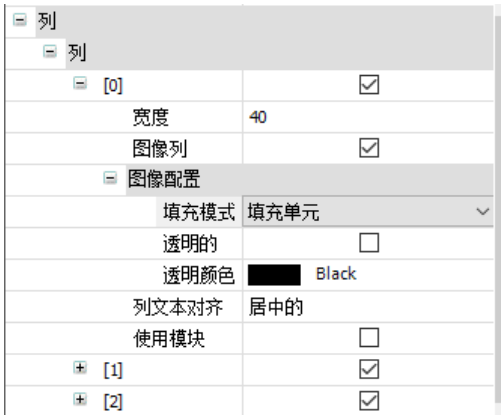
在可视化编辑窗口中打开工具箱，找到“通用控制”添加其中的  组合框数组，通过设置属性菜单中的“变量”和“数据组”和程序中变量及数据间的映射关系。

图表 4.4-26

变量	General_Control.iFactor
数据组	General_Control.arrFactor

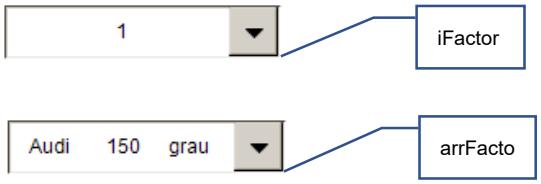
在属性菜单的“列”选项可以自动根据所链接的数组来判断有多少个列，并将每个列的属性逐一显示，可以通过在列的右侧打勾将其激活，展开后还可以设置每个列的宽度等信息。

图表 4.4-27



完成上述步骤后，为了方便在可视化界面中可以直接切换组合框表格内的信息，在可视化界面中添加“组合框整数”，将程序中的“iFactor”映射到“组合框整数”控件上，最终在线显示结果如下：

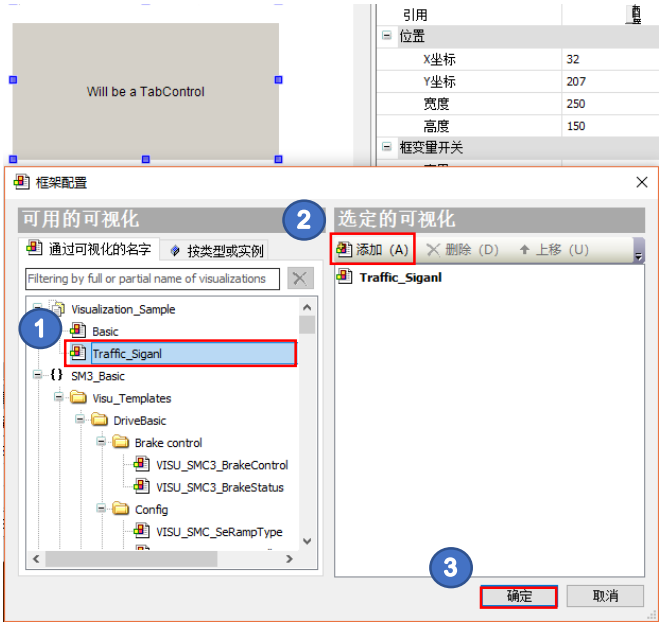
图表 4.4-28



表格控制（TapControl）

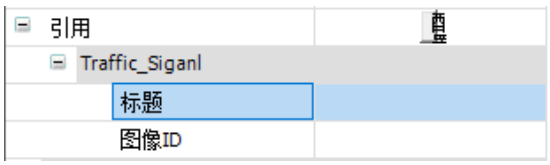
表格控制也就是框架，，定义了当前可视化的一个局部区域，它可以包含其他的一些可视化界面，并可以在在线模式下进行切换。通过对这些可视化输入的适当配置，每个相关区域的框架会显示一个特定可视化界面。

将  “表格控制”添加到可视化编辑窗口中，此时，系统会自动弹出“框架配置”对话框，也可以在“引用”双击调用“框架配置”，在这个页面中会罗列所有可用的可视化，选择一个添加到框架中。



在“表格控制”的属性菜单下的“引用”中可以修改所调用视图的标题，也可以给定图像 ID

图表 4.4-29



通过属性菜单下的“缩放类型”可以设置所调用的视图在“表格控制”控件中的调用方式

图表 4.4-30

通过不同的“缩放类型”可以指定框架应该如何进行大小的变化：

均等的： 图像保留它们的比例，即使独立修改的框架的高度或宽度，图像的高宽比将会保持一致。

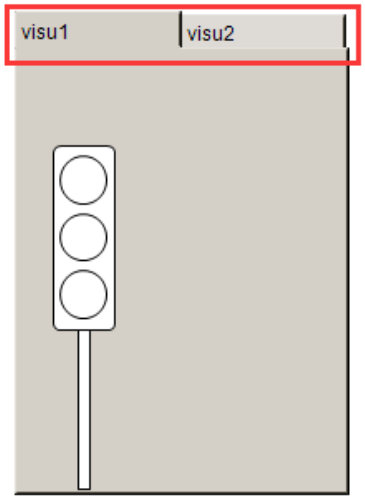
各项异性：框架按照尺寸变化，因此可以独立地修改图像的高度和宽度

固定的：图像的原始大小将保持不变，而不管框架的大小

固定和滚动: 使用这个选项之后相关的图像将会不进行任何缩放的显示。如果图像比框架窗口要大，框架将会自动的添加一个滚动条用于显示可视化相关区域，如果希望通过变量设置滚动条位置，使用属性水平滚动条位置或者竖直滚动条位置

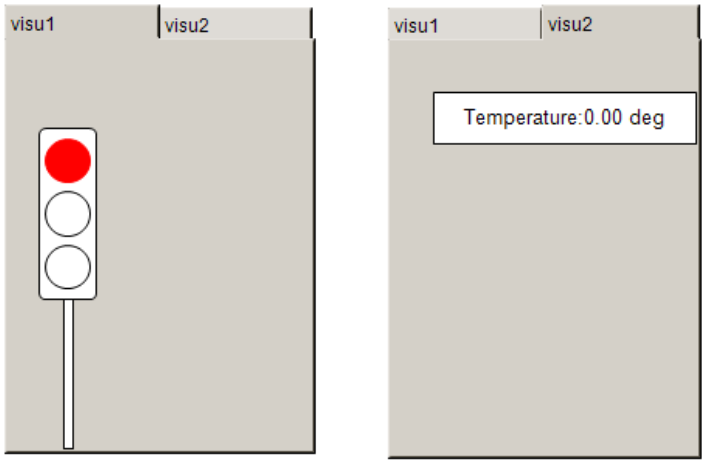
当用户设置了多个视图对象在同一个“表格控制”下，用户可以通过控件顶端的便签进行切换

图表 4.4-31



设置完成后，运行程序查看实际运行效果

图表 4.4-32



按钮（Button）

按钮控件允许用户通过单击来执行操作，在按钮控件中，既可以显示文本，也可以添加图像，在按钮控件被单击时，可以通过控件的属性选择其触发方式。

【例4】 创建一个按钮，单击按钮控件是，执行一段 ST 代码。

在工具箱中，找到“通用控制”找到 

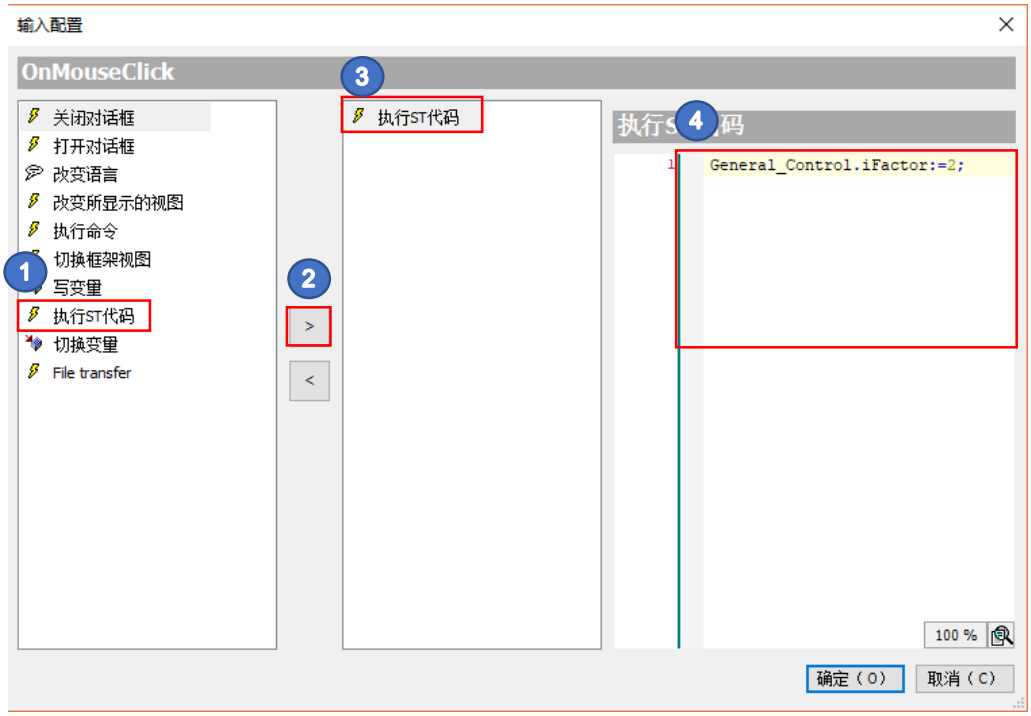
，将其添加到视图编辑窗口中。在按钮的属性菜单中选择“输入配置”，选择其中的鼠标触发动作“OnMouseClick”，即当鼠标单击此控件时，会触发执行相应事件。

图表 4.4-33

单击“配置...”在弹出的“输入配置”左侧选择执行 ST 代码，通过 

添加到右侧并输入需要执行的 ST 代码，需要注意的是，ST 代码中使用的所有变量需为全局变量，调用局部变量的时候需加上命名空间，否则无法通过编译。

图表 4.4-34



【例5】 在可视化界面中，创建两个按钮，作为电机的正反按钮。当按下正转按钮时，电机进行正转，释放按钮时，电机停止；当按下反转按钮时，电机进行反转，释放按钮时，电机停止。

从工具栏中添加两个按钮到可视化编辑区域，在按钮中输入文本“正转”和“反转”作为标识

图表 4.4-35

因为需要按钮按下时有信号，松开后，信号自动消失，需要在属性中找到“输入配置”中选择“触发”的输入方式，在其“变量”中通过输入助手完成变量映射

按钮的触发方式有两种，一种是“触发”（Tap）另外一种“切换”（Toggle）

① “触发”（Tap），按钮按下时触发，变量置为 ON，松开取消，变量置为 FALSE。

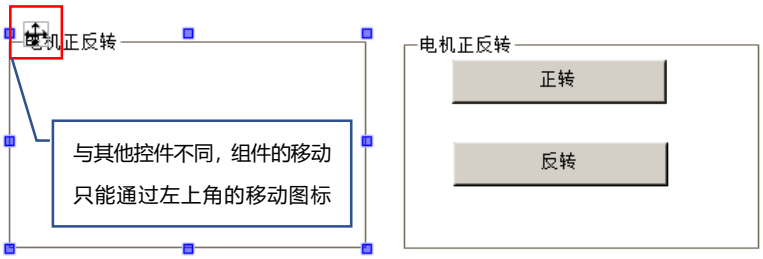
② “切换”（Toggle），按钮按下时，变量一直置为 ON，如果需要让它变成 FALSE，需要手动再触发一下按钮，同样的在“切换”→“变量”下映射变量即可使用，如果需要按钮摁下时为 FALSE，则勾选切换 FALSE。

组块（GroupBox）

组块控件主要是为其他工具提供分组，按照控件的分组来细分可视化界面的功能。在其所包含的工具集周围显示边框，并且可以显示标题，但是组块工具不可以使用滚动条。

在工具箱中，找到“通用控制”后找到  ，将其添加到视图编辑窗口中。选中该控件后，在属性菜单“文本”中可以更改标题，之后将需要分组的工具拖拉至该组块中，即可实现视图元素的分组。 左图为组块空间的视图，右图为在线状态下的显示效果

图表 4.4-36



文本区域（Textfield）

通过“文本区域”空间可以进行文本显示，文本可以是直接在该元素的属性菜单中输入的或者通过“文本变量”的帮助来实现，其功能与矩形框类似，可以直接参考矩形框的配置说明，“文本区域”的图标为 

滚动栏（Scrollbar）

滚动栏允许用户通过拖拉滚动条的位置修改所映射变量的数值。在工具箱中，找到“通用控制”选择  将其添加到视图编辑窗口中。默认的滚动条将水平显示，如果添加到可视化编辑界面中，通过鼠标调整控件大小的时，一旦它的高度超过了滚动条的宽度，将会竖直显示。

在程序中添加如下变量：

图表 4.4-37

```
PROGRAM General_Control
VAR
    ScrollValue:LREAL;
END_VAR
```

选中“滚动条”控件后在属性窗口的“值”选项中完成变量映射，并设置 “最小值”和“最大值”

图表 4.4-38

属性	值
元素名称	GenElemInst_15
元素类型	滚动栏
值	General_Control.ScrollValue
最小值	0
最大值	100
页面大小	
完整滚动	☐

在线状态下显示效果如下，更改滚动条位置，所映射变量数值也会随之改变：

图表 4.4-39

ScrollValue	LREAL	36.76470565795...

滑块（Slider）

滑块类似滚动条，允许用户通过拖拉滑块的位置修改所映射变量的数值。在工具箱中，找到“通用控制”选择  ，将其添加到视图编辑窗口中。

在程序中添加如下变量：

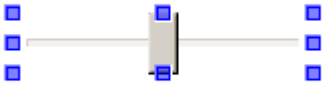
图表 4.4-40

General_Control | General_Contro

```
PROGRAM General_Control
VAR
    SliderValue:LREAL;
END_VAR
```

选中“滑块”控件后在属性窗口的“变量”选项中完成变量映射，通过展开“刻度”可以设置滑块的“刻度始端”和“刻度末端”，也可以设置主副刻度等，具体配置如下：

图表 4.4-41



刻度	
显示刻度	☐
刻度始端	0
刻度末端	100
主刻度值的长度	10
子刻度	5
刻度格式(C-语法)	%i
缩放比例	0

在线状态下显示效果如下，更改滚动条位置，所映射变量数值也会随之改变：

图表 4.4-42

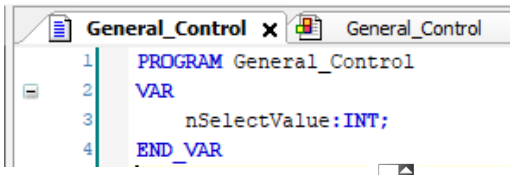
SliderValue	LREAL	27.20588302612...

数值调节钮控件（SpinControl）

数值调节钮控件是一个显示和输入数值的控件。该控件提供上下箭头按钮，用户可以通过单击上下箭头调节控件中映射变量的数值，当然，也可以直接输入数值进行调节。该控件可以设置最值，如果输入的数值大于设置的值，则输入值会自动调整成设置的最大值。反之，如果输入的数值小于这个属性设定的最小值，则输入值会自动调整成设定的最小值。

在程序中添加如下变量：

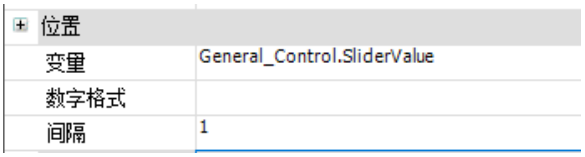
图表 4.4-43



```
General_Control ×  General_Control
1  PROGRAM General_Control
2  VAR
3      nSelectValue:INT;
4  END_VAR
```

在可视化编辑界面的工具箱中，找到“通用控制”大类，添加 数值调节钮控件，选中该控件后在属性菜单中找到“变量”完成变量映射，通过间隔可以设置上下箭头调节数值时的间隔。

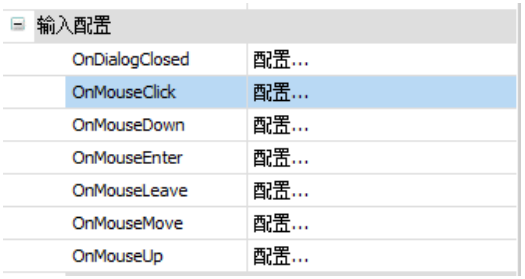
图表 4.4-44

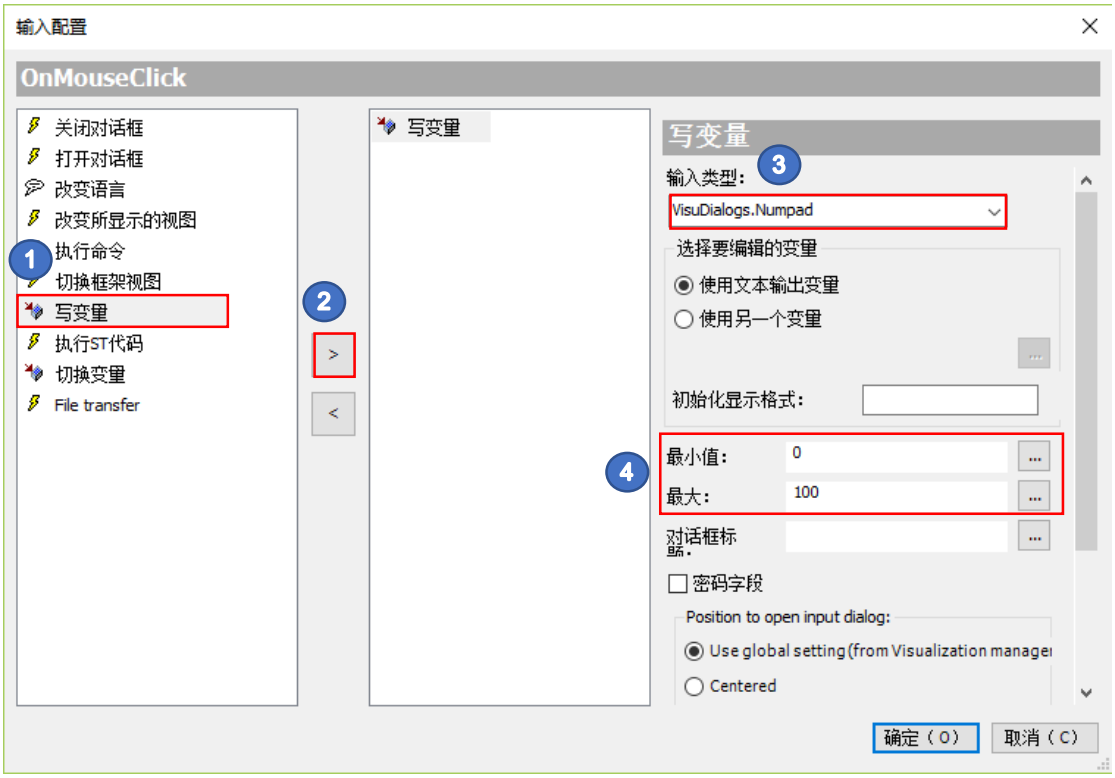


位置	
变量	General_Control.SliderValue
数字格式	
间隔	1

在属性菜单的“输入配置”中选择“OnMouseClick”单击“配置…”允许进行数值输入，并且可以在输入配置的窗口中设置输入数值的限制。

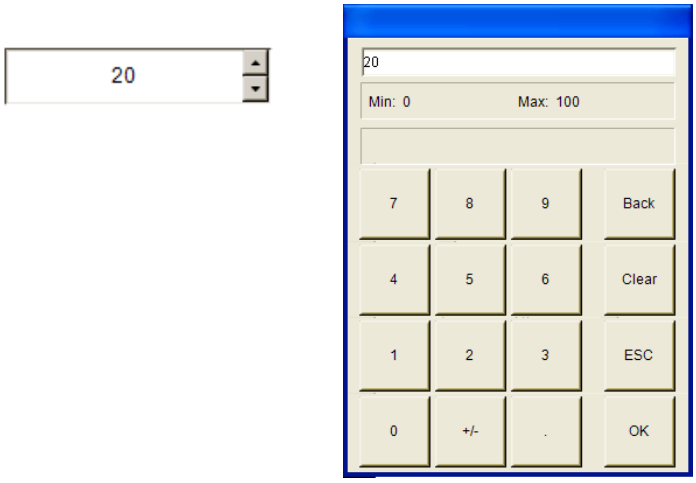
图表 4.4-45





在线运行效果如下：

图表 4.4-46



不可见输入（InvisibleInput）

该元素在视图编辑窗口中添加后，显示为虚线矩形框，但是在线模式下会被隐藏，元素的动作可以通过“输入配置”进行定义。

首先在属性菜单中的“状态变量”中添加

状态变量	
停止输入	General_Control.bInvisible
输入配置	
OnDialogClosed	配置...
OnMouseClick	配置...
OnMouseDown	配置...
OnMouseEnter	配置...
OnMouseLeave	配置...
OnMouseMove	配置...
OnMouseUp	配置...

进度栏（Progressbar）

进度栏允许用户将显示当前进度变量映射到控件中，该元素会根据用户设定的“最小值”和“最大值”，并根据当前映射变量的值，显示当前进度。

通过在程序中创建自累加程序来模拟进度条的增加过程，程序编辑如下：

图表 4.4-47

```
    nProcessValue:INT;
    ton1:TON;
END_VAR
```

```
ton1(IN:=NOT ton1.Q , PT:=T#1S , );
IF nProcessValue<100 AND ton1.Q THEN
    nProcessValue:=nProcessValue+1;
ELSIF nProcessValue=100 AND ton1.Q THEN
    nProcessValue:=0;
END_IF
```

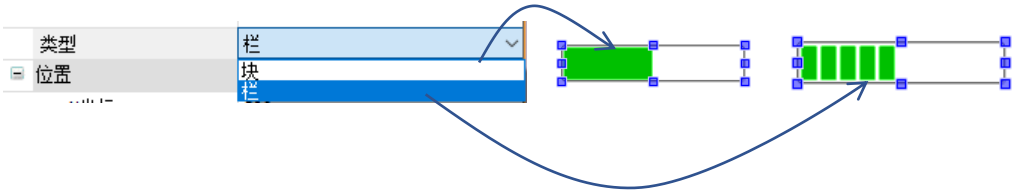
在可视化编辑界面的工具箱中，找到“通用控制”大类，添加  进度栏控件到可视化编辑窗口中，在可视化编辑窗口中选中进度栏控件后在属性窗口找到“变量”中进行变量映射。同时，根据示例程序设置“最小值”和“最大值”。

图表 4.4-48

变量	General_Control.nProcessValue
最小值	0
最大值	100
类型	块

其中进度条的显示形态有两种，同样在属性菜单中找到“类型”下来即可进行选择，

图表 4.4-49



在线运行后进度条会随着变量值的增加缓慢增加：

图表 4.4-50

nProcessValue	INT	95

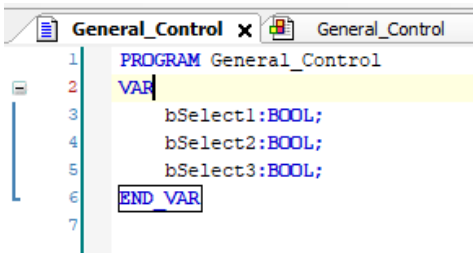
复选框（Checkbox）

复选框用来表示是否选取了某个选项条件，常用于为客户提供具有是/否或真/假值的选项。

下面介绍复选框的基本使用方法：

在可视化编辑界面工具箱中，找到“通用控制”大类，添加其中的  复选框。之后在程序中创建需要使用的变量，提供复选框选项。

图表 4.4-51

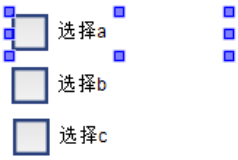


General_Control | General_Control

```
1  PROGRAM General_Control
2  VAR
3      bSelect1:BOOL;
4      bSelect2:BOOL;
5      bSelect3:BOOL;
6  END_VAR
7
```

在可视化界面的编辑区选中复选框，通过设置其属性中的“变量”建立与程序之间的映射关系，映射完成后，复选框中的选择会直接写入程序的 BOOL 变量中，同时在文本中输入对于该选项的描述。分别对三个复选框进行变量映射。

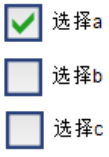
图表 4.4-52



元素类型	复选框
文本ID	966
位置	
变量	General_Control.bSelect1
框架大小	样式
文本	
文本	选择a
工具提示	

在线运行结果显示如下：

图表 4.4-53



bSelect1	BOOL	TRUE
bSelect2	BOOL	FALSE
bSelect3	BOOL	FALSE

单选按钮（RadioButton）

单选按钮为用户提供由两个或多个互斥选项组成的选项集，用户选中某个单选按钮时，同一组中的其他单选按钮将不能被同时选择，这一点和复选框完全不一样。

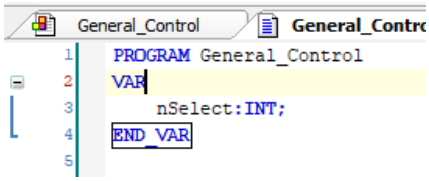
下面介绍单选按钮的基本使用方法：

首先创建单选按钮，在可视化编辑界面工具箱中，找到“通用控制”大类，添加其中的  单选按钮。

图表 4.4-54

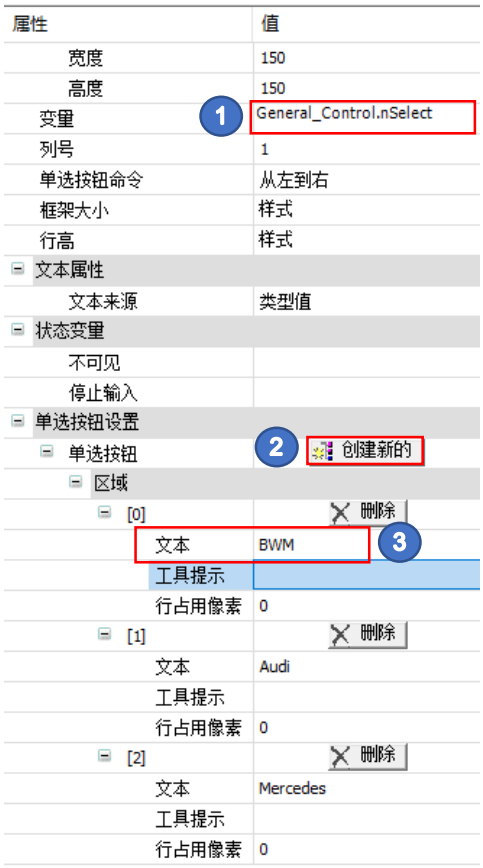
在程序窗口添加示例变量：

图表 4.4-55



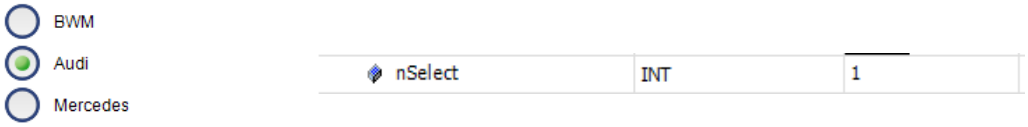
在视图编辑窗口选中单选按键后，在其属性菜单中找到“变量”完成变量映射，接下去需要确认需要多少个单选按钮，通过属性菜单下的“单选按钮设置”展开可以单击“创建新的”添加新的单选按钮到用户所需数量，并在添加出来的单选按钮中填写“文本”，作为单选按钮选项的标识：

图表 4.4-56



运行程序并在线查看运行效果，单选按钮总共提供了三个选项，在同一时刻，仅能选择其中某一项。

图表 4.4-57



4.4.3 报警管理

可视化界面中的报警管理主要包括了报警表格和报警标志。

图表 4.4-58

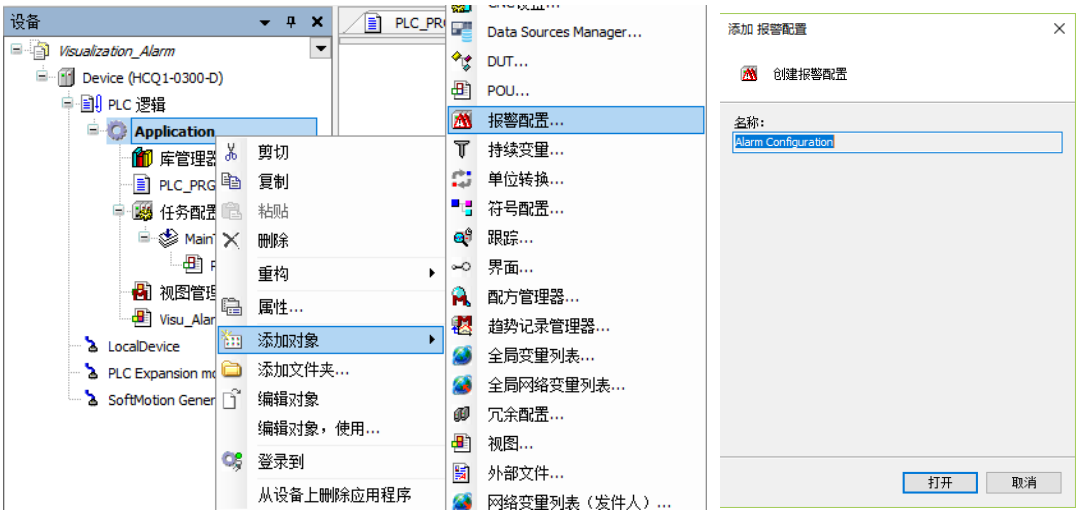
报警表格

在 CODESYS 中用户可以自定义可视化报警，但是必须在报警配置中预先进行配置。首先在工具箱的“报警管理”中找到 报警表格，拖拽到可视化编辑窗口中。可视化报警配置主要分成两个部分：第一部分，需要在“Application”中设置报警配置；第二，需要在可视化编辑器中配置报警表格或者报警标志。

- 在“Application”中添加报警配置

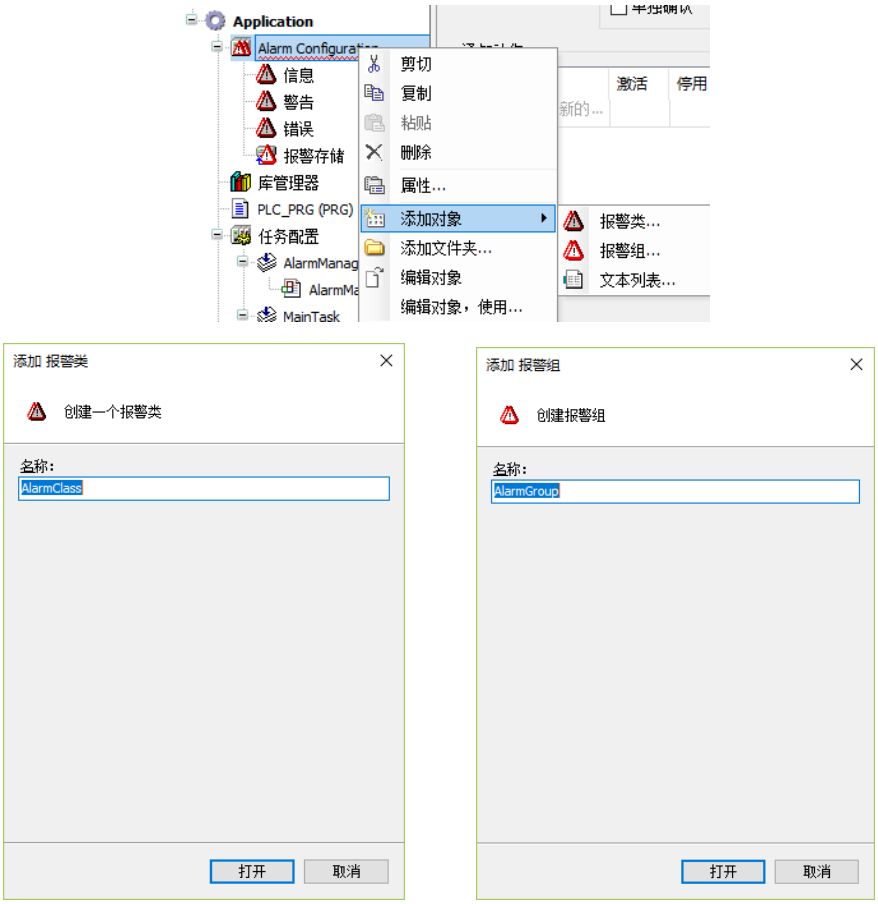
单击“Application”右击选择“添加对象”，找到其中的“报警配置”后单击打开，在弹出“添加报警配置”的对话窗口中输入报警配置的名称后，单击“打开”进行创建。

图表 4.4-59



在“Alarm Configuration”（报警配置）中用户需要设置报警内容及触发机制。首先，选中“Alarm Configuration”右键单击找到“添加对象”分别添加“报警类…”和“报警组…”，给定名称后单击打开新建。

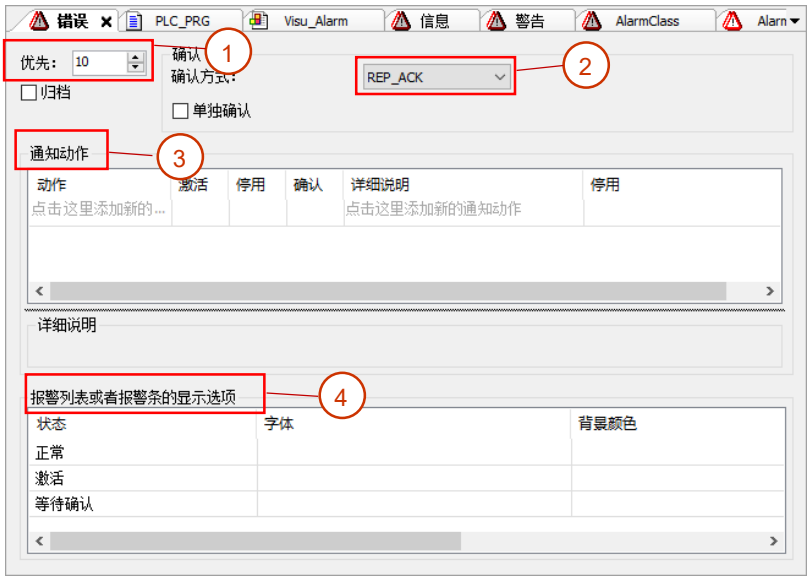
图表 4.4-60



- 报警类型配置（AlarmClass）

添加报警配置后，报警类型是默认分成 3 类，分别为“Error”、“Info”和“Warning”，不同的报警类型之间主要的区别在于报警的优先级及确认方式。报警配置界面如下：

图表 4.4-61

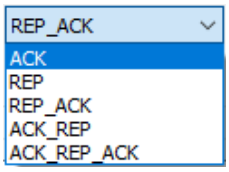


① 优先（Priority）：定义想要显示的所有报警的优先级，允许设置的范围是 0~255，数字越小优先级越高，所以其中优先级最高的是“0”，最低的是“255”，较高或者中等优先级的报警通常要求用户立即确认，而优先级非常低的报警则可能不做要求。对于已经产生的报警，尽管生成报警的条件可能已经消失（例如，电机运行过程中温度过高产生的报警，之后电机温度已经正常）但是在用户确认之前，该报警本身不会自动取消，也就是说不会被认为已经得到解决。

② 确认（Acknowledgement）：报警发生时，用户（系统）必须确认报警。确认只是表示用户（系统）注意到了该报错，并不表示报错已经得到解决。它也不代表报警条件会返回到正常，有时，即使没有任何外界干预，它也有可能自行恢复正常，主要提供一下几种报警确认方式：

REP_ACK：经过（单个）修复和确认后报警不激活
ACK：确认
REP：移除导致的问题后，报警不激活
ACK_REP：经过确认和修复后报警不激活
ACK_REP_ACK：经过接收、修复及确认后报警不激活

图表 4.4-62



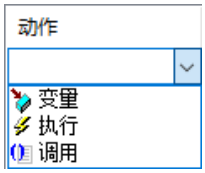
③ 通知动作（Notification actions）：通知动作下包含了“动作”、“激活”、“确认”、“详细说明”和“停用”其中动作提供变量、执行和调用三种可供选择；通过是否勾选“激活”可以选择调用或不调用当前动作；通过是否勾选“确认”可以选择调用该动作时是否需要用户确认；通过“详细说明”可以配置当前通知动作映射的变量；通过“停用”可以通过连接变量控制当前通知动作是否启用，下面三种动作类型进行详细说明：

变量：选择“变量”后系统会自动弹出命令提示，针对该报警信息，用户可以设置对应的变量或表达式

执行：输入当前报警出现时的“执行文件”名。在“详细说明”中可以直接输入任意参数调用

调用：输入要调用的“功能块实例”的名字

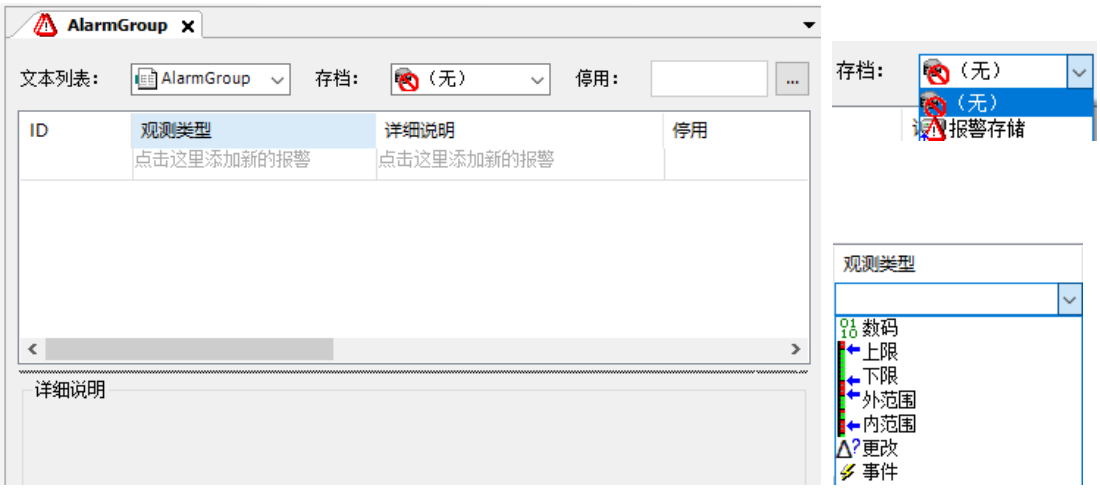
图表 4.4-63



- 报警组配置（AlarmGroup）

在“AlarmGroup”中需要用户配置“观测类型”也就是报警触发类型，也允许用户配置报警的存档：

图表 4.4-64



“AlarmGroup”中的“报警存储”如果启用的话用户需要在“AlarmConfiguration”下找到“报警存储”配置报警存储的路径和限制。

图表 4.4-65

对几种报警触发类型进行详细说明如下表格：

图表 4.4-66

触发类型	说明
数码	左侧输入要监控的表达式，右侧输入要检查的表达式，中间选择想用的操作符（=或<>）
上限	同“数码”类似，但是对于比较操作符>或>=，有选择的使用“滞后%”的定义
下限	同“数码”类似，但是对于比较操作符<或<=，有选择的使用“滞后%”的定义
内范围	输入要监视的表达式。“区域”：当监视的表达式到达定义的内范围值时，报警出现。在左侧输入表达式表示下极限，在右侧输入上极限，被监视的表达式显示在不可编辑区域。合理设置操作符，有选择的使用“滞后%”的定义
外范围	输入要监视的表达式。“区域”：当监视的表达式到达定义的外范围值时，报警出现。在左侧输入表达式表示下极限，在右侧输入上极限，被监视的表达式显示在不可编辑区域。合理设置操作符，有选择的使用“滞后%”的定义
更改	“表达式”：输入要监视的表达式，当它的值发生变化时，报警出现
事件	这种情况下报警通过应用触发，使用 AlarmManager.library 中的函数

注意：

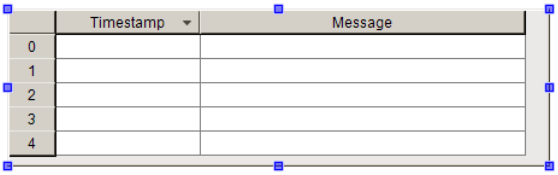
“滞后%”：如果使用了滞后，那么报警情况会保持 TRUE，直到达到一个具体的滞后值来改变。滞后的大小以极限值的百分比计。例如，将上极限设置为“i_temp>=30”，将滞后设置为 10%。当变量 i_temp 达到过超过 30 时，报警出现，直到它的值降到 27，报警情况才会消失。

配置的格式如：PLC_PRG.i_temp>(0.1*30)，该表达式应用了 10%的滞后功能

- 在可视化界面中配置报警

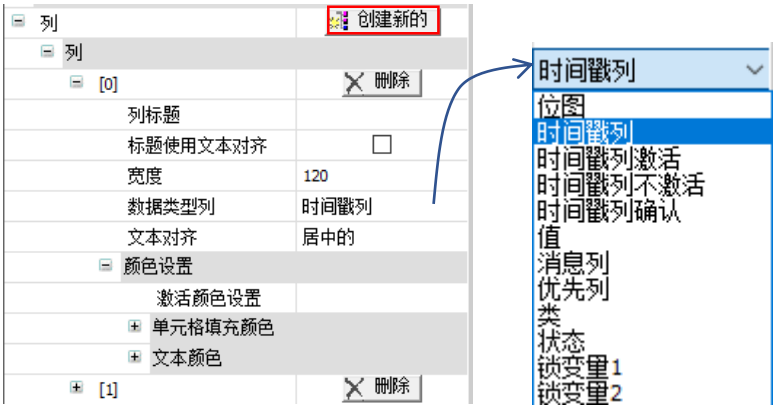
首先在可视化工具箱的“报警管理”中找到报警表格并拖拽到可视化编辑窗口

图表 4.4-67



默认添加出来的报警表格只有两列信息，用户可以根据实际需求添加更多的列。通过在可视化编辑窗口选中“报警表格”后在右侧属性菜单中找到“列”进行列的删除和添加，对于添加的列，可以设置标题、宽度、文本显示格式等，列中需要显示的内容是可选择的，在“数据类型列”下拉列表中提供了多种类型的列供客户选择。

图表 4.4-68



时间戳列：最后一个状态变化的报警的日期和时间

时间戳列激活：最后一次警报活跃的日期和时间

时间戳列不激活：最后得到无效的警报的时间和日期

时间戳列确认：最后确认的时间和日期

值：当前表达式的值

消息列：监控值

优先列：警报优先级

类：警报类

状态：警报状态

报警表中显示的报警需要操作人员确认，与确认动作相关的变量可以在“控制变量”中进行设置

图表 4.4-69

确认所选变量：如果该变量为 TRUE，那么报警表中所选中的报警将会被确认

确认所有可见变量：如果该变量为 TRUE，那么报警表中所有的报警都会被确认

历史： 如果该变量为 TRUE，那么报警表将会转化为历史模式。这意味着将会按照日期对所有报警降序排列。任何新事件将会被添加到当前表中。

冻结滚动位置：如果该变量为 TRUE，那么在历史模式下即使有新的报警被激活，滚动条的当前位置也将会被锁定。否则，在这种情况下滚动条会跳转到报警表的第一行。

计数报警：当前显示的报警数目将被存储如该变量

可见行数：通过该变量可以设置报警表的可见行数

当前的滚动指数： 通过该变量可以跳转到报警表第一个可建行的索引

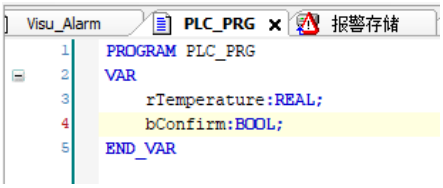
当前排序列：通过该变量排序警告

排序方向变量：通过该变量定义表格上的排序方向，TRUE 为升序，FALSE 为降序

【例5】 设置一个报警温度。当温度大于 50°时，出现温度过高报警，当温度低于 10°时，出现温度过低报警。

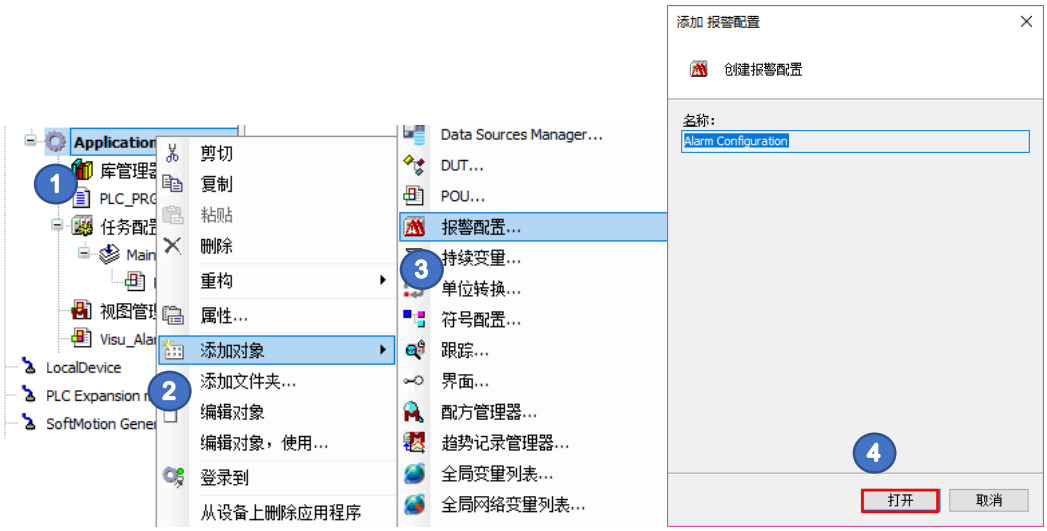
首先在程序中添加温度变量 rTemperature，数据类型为实数，添加 bConfrim 作为确认信号

图表 4.4-70



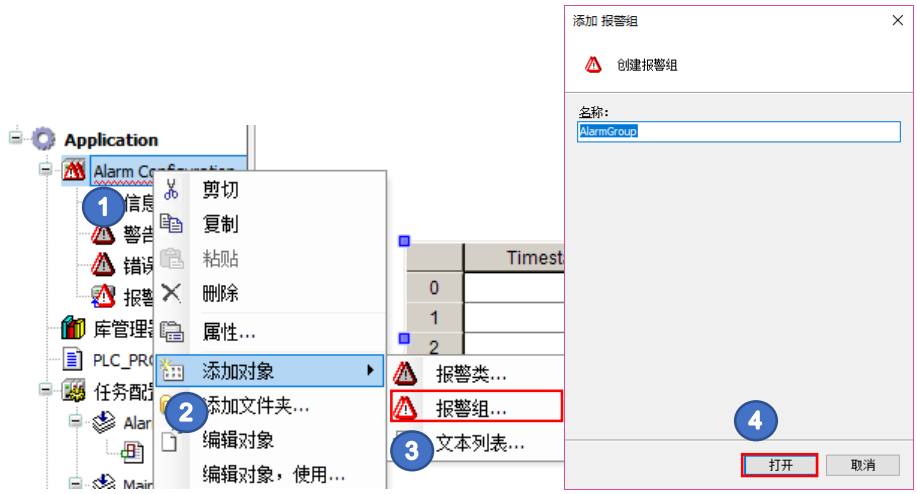
在“Application”右击添加报警配置，给定报警配置名称后点击“打开”新建

图表 4.4-71



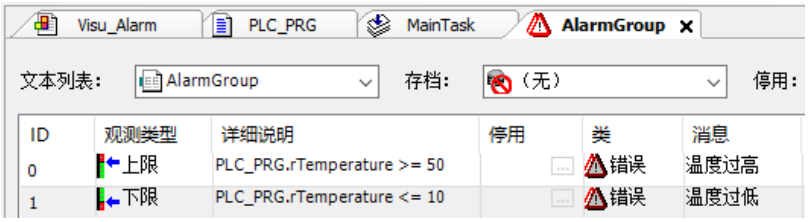
在新建的报警配置下添加新的报警组，右击“Alarm Configuration”

图表 4.4-72



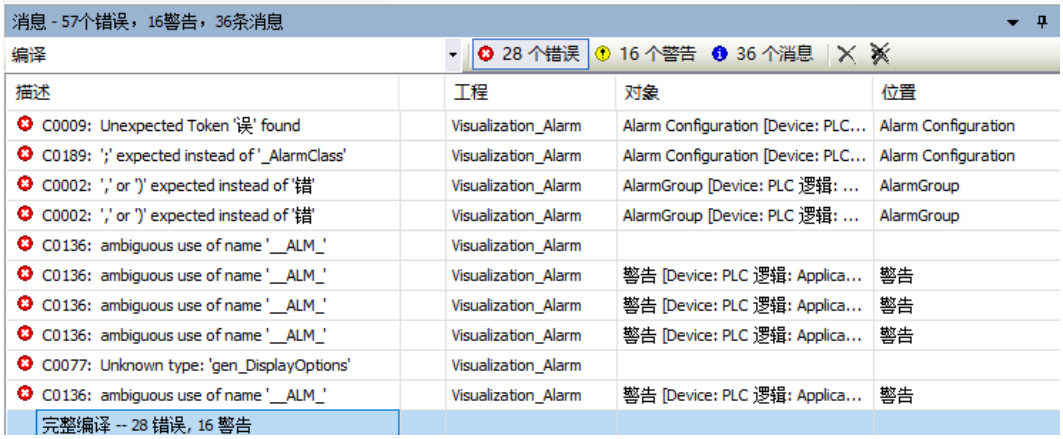
报警组的内容设置为上下限的触发方式，并在“Message”中添加相应的文字报警信息

图表 4.4-73

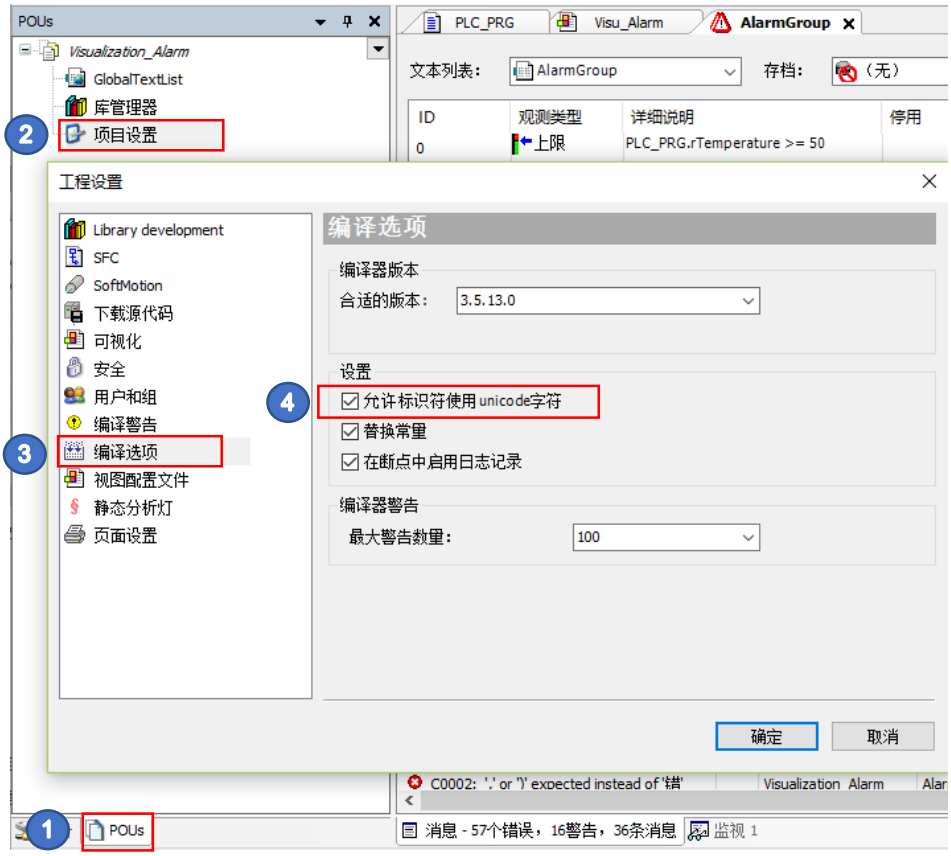


此时，如果是汉化的 CODESYS 创建的报警配置，在消息窗口会出现编辑报错：

图表 4.4-74

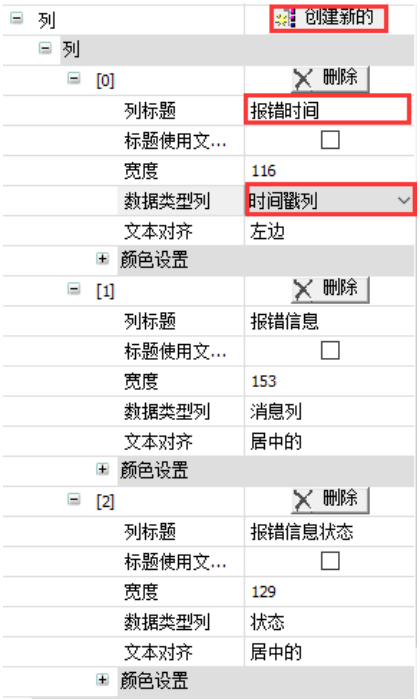


这是由于新建的“报警配置”下，默认添加的报警类型“错误”、“警告”和“消息”为中文，需要支持 Unicode 编码，否则会报错，在项目设置中勾选 Unicode 编码即可



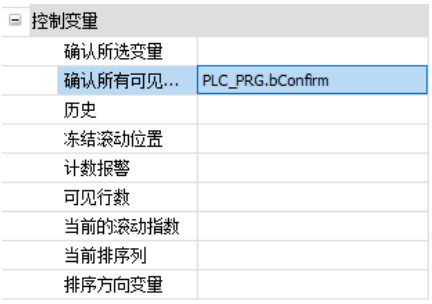
接下去需要在可视化界面中添加报警表，并定义列属性。报警表默认只有两列，用户可以自行增加更多的列以显示报警信息。通过“创建新的”按钮可以增加新的列，通过自定义列标题方便用户查看该列信息，通过在数据类型的下拉菜单中选择列显示内容来源。

图表 4.4-75



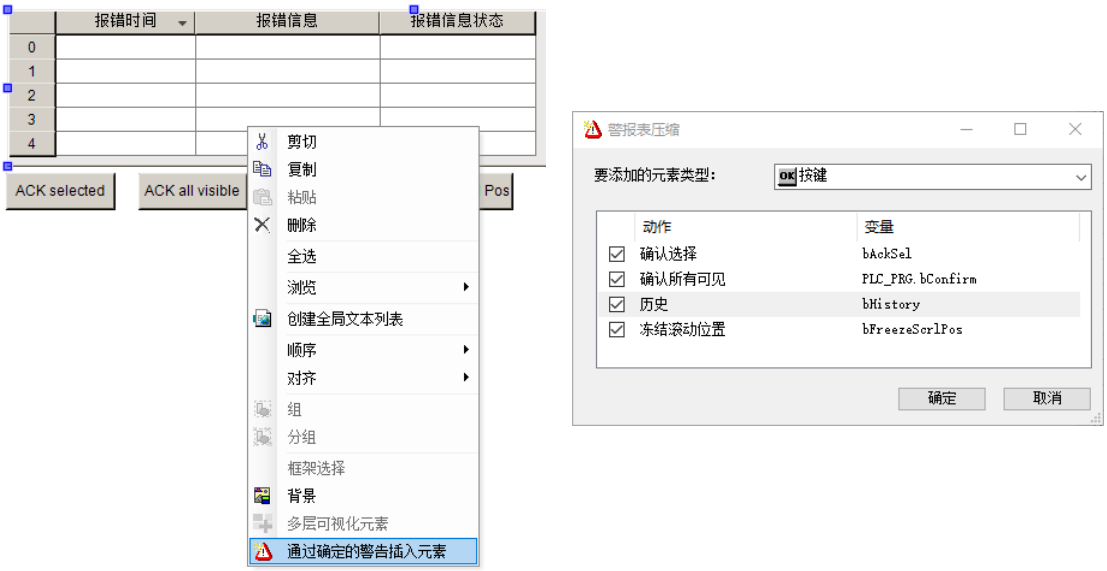
找到“控制变量”，在其中的“确认所有可见变量”中映射程序中“bConfirm”作为确认信号

图表 4.4-76



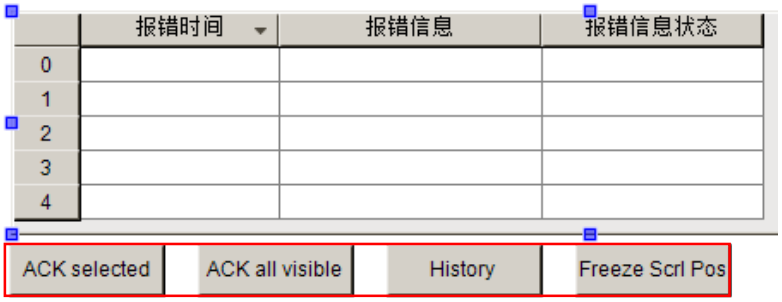
接下去，添加控制变量“按钮”，通过默认提供的四个附加按钮可以实现控制变量中提供的部分功能。在可视化编辑区的报警表格右击找到“通过确定的警告插入元素”后，在弹出的对话框中勾选需要添加的附加按钮。勾选的附加按钮会自动映射在“控制变量”中用户添加的变量，如果用户在控制变量中没有添加定义在应用中的变量（如：PLC_PRG.bConfirm），系统将自动创建并输入一个本地可视化变量，例如 bAckAel 用于确认选择。

图表 4.4-77



点击确定后，勾选的动作对应的附加按键会默认添加到表格下方。

图表 4.4-78



登陆后查看在线运行效果：

图表 4.4-79

报警条

报警条类似报警表格，是报警表的简单版本，可以用于报警组和类的单一报警可视化配置，属于特殊报警类别的“报警配置”。

“报警条”的具体报警配置过程可以参考报警表的配置。

4.4.4 测量控制工具

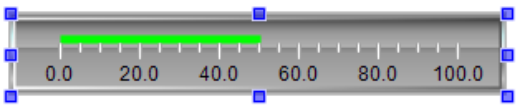
测量控制控制工具主要包括一些常用的图形指示工具，例如“显示图像栏”，“仪表盘显示”和“直方图”等等。

图表 4.4-80

显示图像栏

显示图像栏，也可以称为条状图，单击工具箱后找到“测量控制”中的  添加到可视化界面中，该控件一般可以用来显示一个固定区间内活动的数值，例如液位的显示、气压的显示和温度的显示等等。

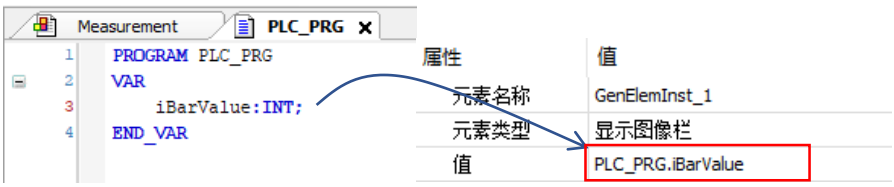
图表 4.4-81



显示图像栏用于显示映射的固定变量的数值，并通过其中的柱状图来表示其值的变化。下面介绍显示图像栏的基本使用方法。

在工具箱的“测量控制”中找到需要添加的“显示图像栏”，将其拖拉到可视化编辑区中。要使用该控件，首先需要进行变量映射，在可视化编辑区中选中该控件后，在属性菜单的“值”一栏中映射程序中变量。

图表 4.4-82



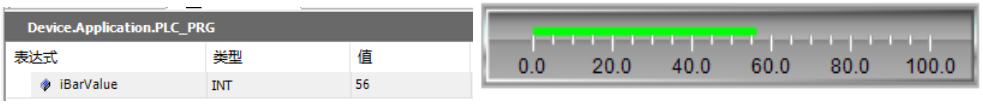
完成变量映射后，设置刻度中的相关参数，包括“刻度始端”，“刻度末端”，“主刻度”和“子刻度”等。

图表 4.4-83

刻度	
刻度始端	0
刻度末端	100
主刻度...	20
子刻度	5
刻度线宽	1

完成上述设置后，显示图像栏的基本功能就可以实现了，实际运行效果如下：

图表 4.4-84



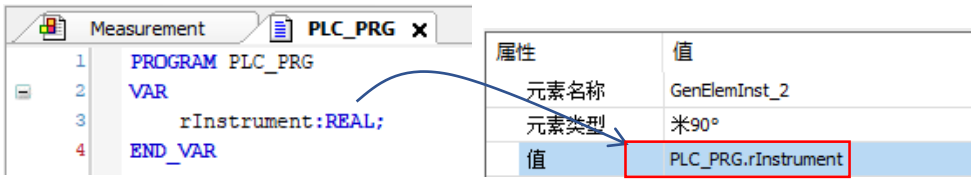
仪表盘（90°，180°，360°）

仪表盘可以用于显示固定上下限的变量的数值变化，在生活当中也是十分常见，例如汽车的油量显示和当前速度，单击工具箱后找到“测量控制”中的“米 90°”、“米 180°”、“米 360°”三个控件，选择其中任意一个添加到可视化界面中。

图表 4.4-85

接下去介绍如何在可视化编辑区使用仪表盘控件，在工具箱的“测量控制”中找到需要添加的仪表，将其拖拉到可视化编辑区中。要使用该控件，首先需要进行变量映射，在可视化编辑区中选中该控件后，在属性菜单的“值”一栏中映射程序中变量。

图表 4.4-86



完成变量映射后，需要设置刻度格式，找到属性菜单中“标签”下的刻度格式，刻度格式（C-语法）是指根据 C 艳艳语法定义刻度标号的格式，默认值是“%.1f”，其中小数点之前的 1 表示在输出设备上输出，小数点后的 1 表示四舍五入后保留一位小数。“f”是 float 的缩写，表示浮点型数据。

图表 4.4-87

标签	
标签	内部
单元	
字体	Font-Standard
刻度格式(C-语法)	%.1f
最大标签文本宽度	34
标签文本高度	14
字体颜色	Element-Meter-LabelFontColor

接下去设置刻度中的相关参数，包括“刻度始端”，“刻度末端”，“主刻度”和“子刻度”等。

图表 4.4-88

刻度	
子刻度位置	外部
刻度类型	行
刻度始端	0
刻度末端	100
主刻度值的长度	20
子刻度	5
刻度线宽	1
刻度颜色	Element-Meter-ScaleColor
3D刻度	☑

完成上述设置后，显示图像栏的基本功能就可以实现了，实际运行效果如下：

图表 4.4-89

Device.Application.PLC_PRG

表达式	类型	值
iBarValue	INT	0
rInstrument	REAL	33.3

直方图

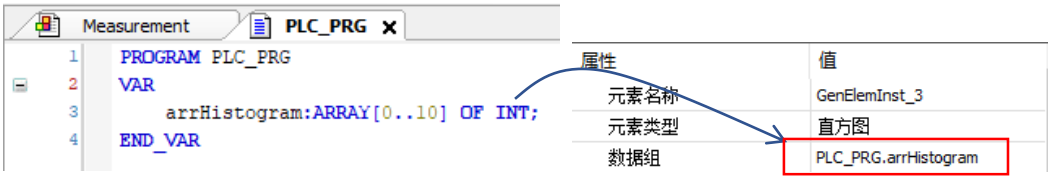
直方图又称为质量分布图，是一种几何形状图标，它是可视化界面中表示数据变化情况的一种主要工具。用直方图可调比较直观的看出产品质量特性的分布状态，对于分布状况一目了然，便于用户判断其总体质量分布的情况。

图表 4.4-90

下面介绍直方图工具的基本使用方法。

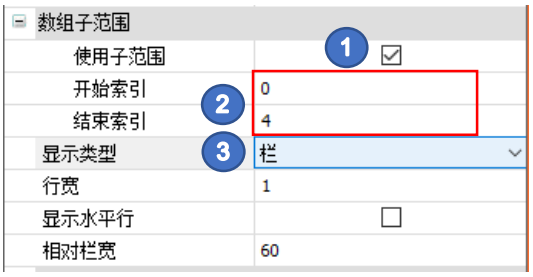
在工具箱的“测量控制”中找到需要添加的“直方图”  ，将其拖拉到可视化编辑区中。要使用该控件，首先需要进行变量映射，在可视化编辑区中选中该控件后，在属性菜单的“数据组”一栏中映射程序中变量。

图表 4.4-91



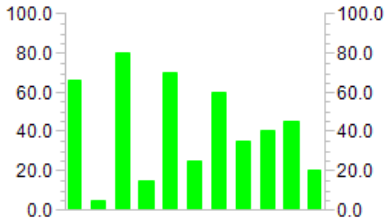
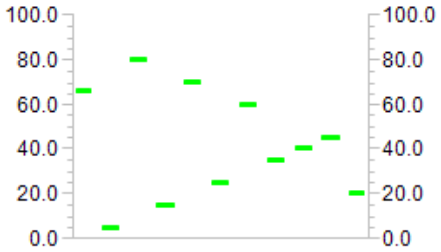
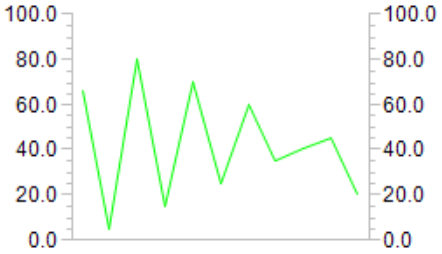
然后，使能“使用子范围”，选中“使能子范围”后的复选框即可。接下去通过修改“数据子范围”中的“开始索引”和“结束索引”，可以设置在该直方图中显示的变量的数量。

图表 4.4-92



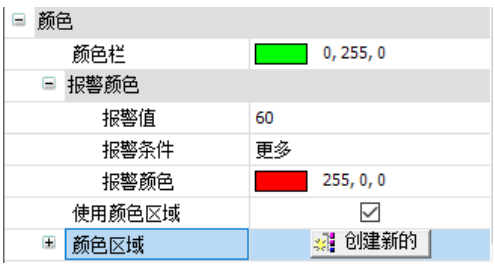
最后还需要用户选择图形的显示类型，可以将其设置为“栏”、“行”或者“曲线”具体区别见下表：

图表 4.4-93

显示类型	图例
栏	
行	
曲线	

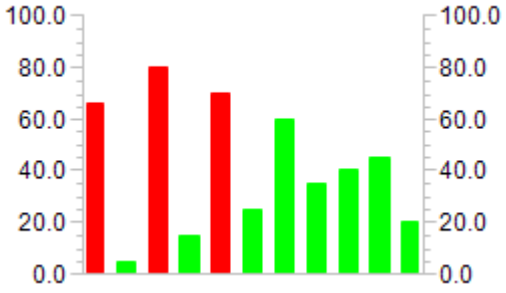
当数值超过/小于某个设定值时，可以使用报警颜色，例如，当数值超过或等于 60 的时候，启用报警颜色对直方图进行显示，在属性菜单的“颜色”下也可以设置直方图正常状态下的显示颜色。

图表 4.4-94



运行程序显示效果如下：

图表 4.4-95

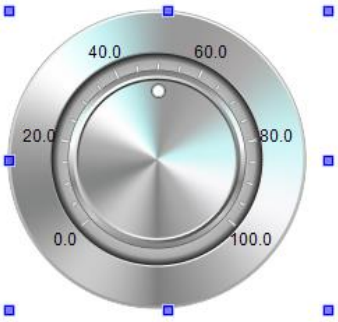


arrHistogram	ARRAY [0..10] OF INT	
arrHistogram[0]	INT	66
arrHistogram[1]	INT	5
arrHistogram[2]	INT	80
arrHistogram[3]	INT	15
arrHistogram[4]	INT	70
arrHistogram[5]	INT	25
arrHistogram[6]	INT	60
arrHistogram[7]	INT	35
arrHistogram[8]	INT	40
arrHistogram[9]	INT	45
arrHistogram[10]	INT	20

电位器

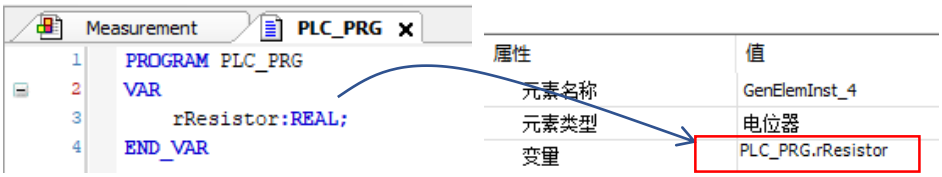
实际生活中的电位器是当电刷沿电阻体移动时，在输出端即获得与位移成一定关系的电阻值或者电压值，在可视化界面中，则将根据当前滑动位置转换成对应数值的显示。

表 4.4-96



接下去介绍如何在可视化编辑区使用电位器控件，以电阻调节为示例，首先在工具箱的“测量控制”中找到需要添加的“电位器”  ，将其拖拉到可视化编辑区中。要使用该控件，需要进行变量映射，在可视化编辑区中选中该控件后，在属性菜单的“变量”一栏中映射程序中变量。

图表 4.4-97



完成变量映射后，设置刻度格式，找到属性菜单中“标签”下的刻度格式，刻度格式（C-语法）是指根据 C 艳艳语法定义刻度标号的格式，默认值是“%.1f”，其中小数点之前的 1 表示在输出设备上输出，小数点后的 1 表示四舍五入后保留一位小数。“f”是 float 的缩写，表示浮点型数据。

图表 4.4-98

标签	
标签	内部
单元	
字体	Font-Standard
刻度格式(C-语法)	%.1f
最大标签文本宽度	34
标签文本高度	14
字体颜色	Element-Meter-LabelFontColor

接下去设置刻度中的相关参数，包括“刻度始端”，“刻度末端”，“主刻度”和“子刻度”等。

图表 4.4-99

刻度	
子刻度位置	外部
刻度类型	行
刻度始端	0
刻度末端	100
主刻度值的长度	20
子刻度	5
刻度线宽	1
刻度颜色	Element-Meter-ScaleColor
3D刻度	☑

用户还需要在属性菜单“箭头”下通过“起始箭头”和“结束箭头”定义电位器上的刻度的角度值，通过直接给定角度值或者给定变量完成定义，这里有效的变量需为整型数据，定义角度值，位于刻度的零点位置按照顺时针到刻度的起始位置，刻度的零点位置是“3 o’clock”，角度范围从 0°到 360°。

图表 4.4-100

箭头	
箭头类型	圆
颜色	255, 255, 255
起始箭头	220
变量	PLC_PRG.iStart
结束箭头	-40
变量	PLC_PRG.iEnd

```
    iStart:INT:=220;
    iEnd:INT:=-40;
END VAR
```

完成上述设置后，显示图像栏的基本功能就可以实现了，实际运行效果如下：

图表 4.4-101

rResistor	REAL	33
iStart	INT	220
iEnd	INT	-40

4.4.5 灯/开关/位图工具

工具箱中提供的灯、开关、位图工具主要用于提供不同类型的指示灯和开关，允许用户根据自身需求选择合适的指示灯和开关进行界面的编辑。

图表 4.4-102

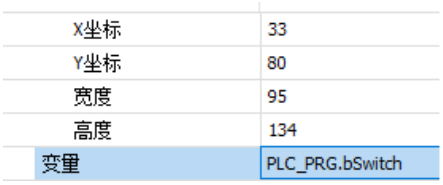
图像开关

图像开关可以通过用户输入确定图像打开或关闭，和 BOOL 型变量比较类似。鼠标在图像元素上单击，图像开关映射的程序中变量将会变成 True。

下面介绍图像开关的使用步骤，在工具箱中找到“灯/开关/位图”并选择想要添加的“图像开关”，将其拖拽至可视化编辑区。

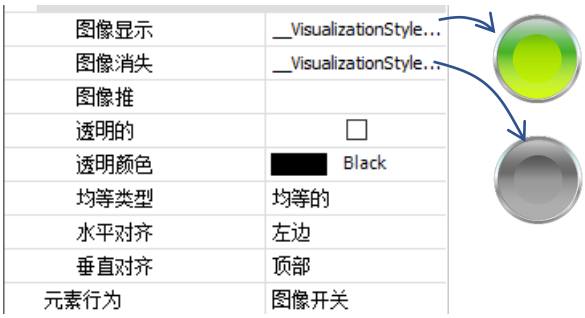
在主程序中添加“bSwitch”变量后，在属性菜单中找到“变量”完成变量映射

图表 4.4-103



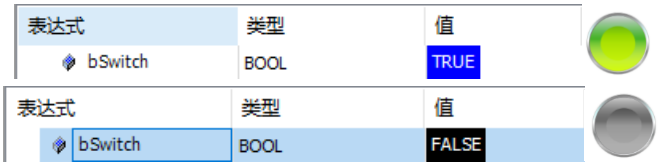
在属性菜单“图像设置”中添加“图像显示”和“图像消失”对应的图形，分别映射变量的“True”和“False”

图表 4.4-104



登录查看在线运行效果，单击图像开关可以直接切换开关变量的状态：

图表 4.4-105



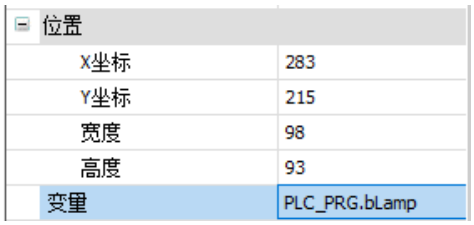
灯

当关联到的变量被置位时，灯将会被点亮，通过属性菜单，背景下的“背景图片”可以选择灯的颜色。

下面介绍灯控件的使用步骤，在工具箱中找到“灯/开关/位图”并选择想要添加的“灯”，将其拖拽至可视化编辑区。

在主程序中添加“bLamp”变量后，在属性菜单中找到“变量”完成变量映射

图表 4.4-106

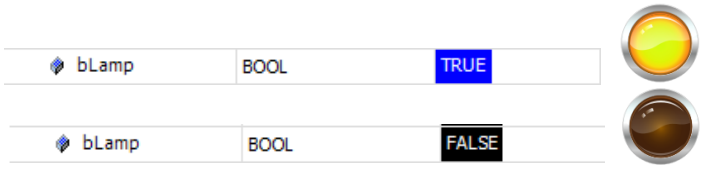


在属性菜单“背景”找到“背景图像”下拉可以选择灯的颜色

图表 4.4-107

登录查看在线运行效果，通过修改程序中“bLamp”的状态：

图表 4.4-108



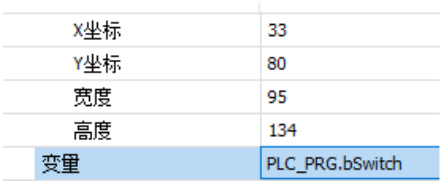
拨码开关/电源开关/按键开关/按键开关 LED/摇杆开关/旋转开关

位接触开关种类比较多但是配置都比较类似，因此选择其中一种开关进行介绍，以按键开关 LED 为例介绍开关的使用。

在工具箱中找到“灯/开关/位图”并选择想要添加的“按键开关 LED”，将其拖拽至可视化编辑区。

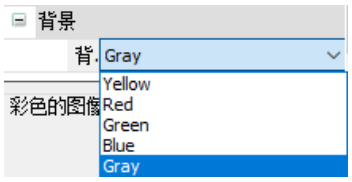
在主程序中添加“bSwitch”变量后，在属性菜单中找到“变量”完成变量映射

图表 4.4-109



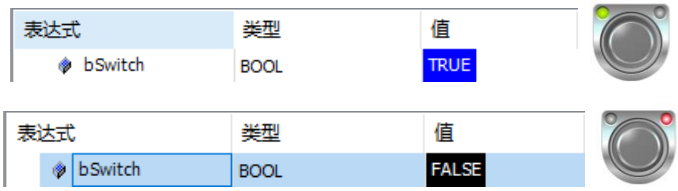
完成变量映射后，用户可以根据实际的需求在“背景”选项下的“背景图像”中设置需要的颜色

图表 4.4-110



登录查看在线运行效果，单击按键开关 LED 可以直接切换开关变量的状态：

图表 4.4-111



4.4.6 特殊控制工具

特殊控制工具主要包括一些常用的图形指示工具，如趋势图，ActiveX 元素等，特殊控制工具在可视化界面中主要包括如下图几种，接下去会对其中几种常用的工具进行详解。

图表 4.4-112

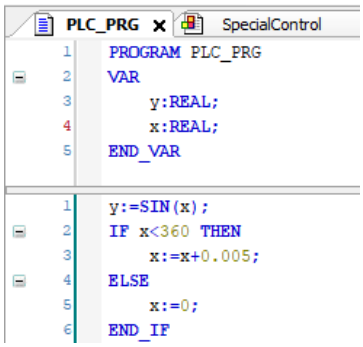
Trace

通过这个元素用户可以在可视化界面中插入一个跟踪表格。需要跟踪的对象由用户在跟踪控件的属性中指定。

【例6】 在可视化界面中新建 Trace 用于监控 y:=sin(x)的波形

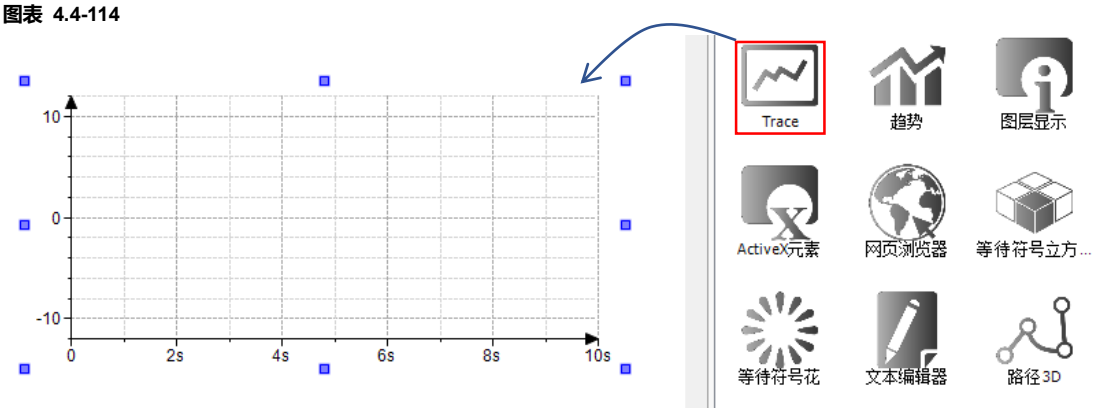
首先在主程序中编写示例程序

图表 4.4-113



接下去在工具箱中找到“特殊控制”后添加“Trace”  到可视化编辑区域

图表 4.4-114

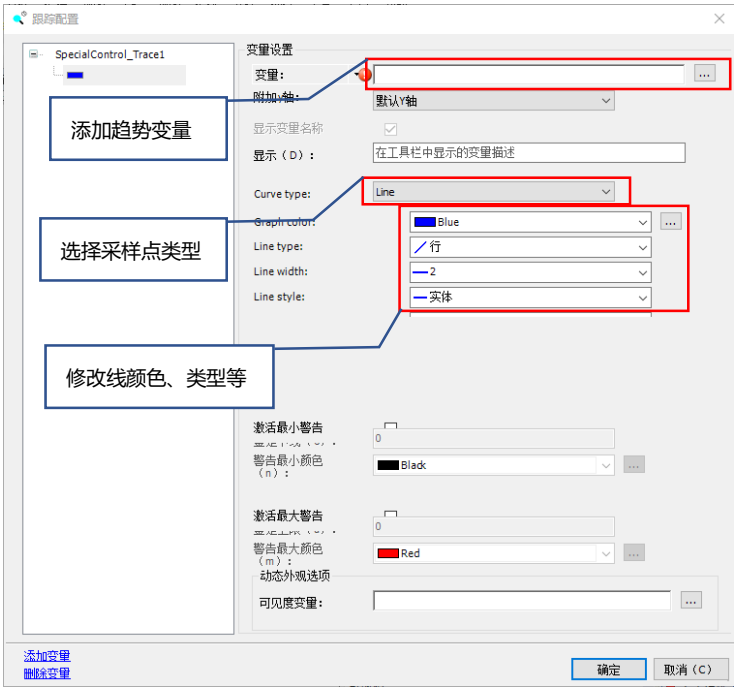


在可视化编辑区中选中“Trace”控件，右击选择“配置跟踪。。。”单击打开

图表 4.4-115

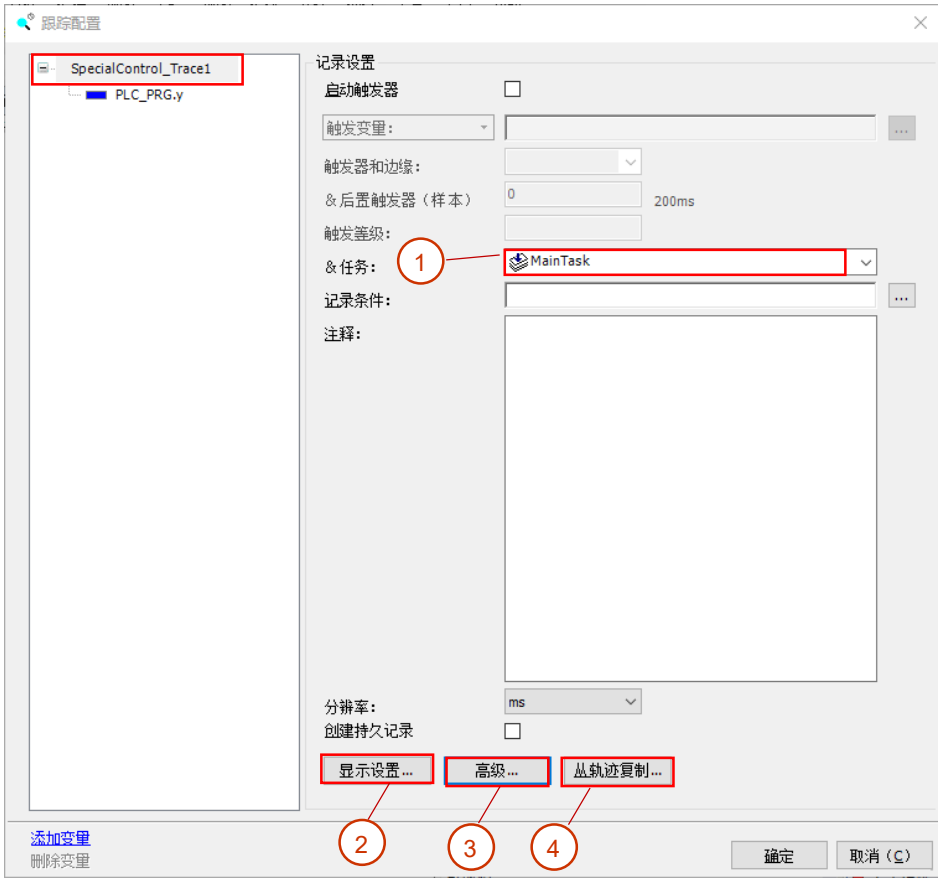
在弹出对话框“跟踪配置”中进行变量添加和跟踪任务配置，在添加变量对话框中同样可以设置采样点类型和采样变量的线类型颜色等

图表 4.4-116



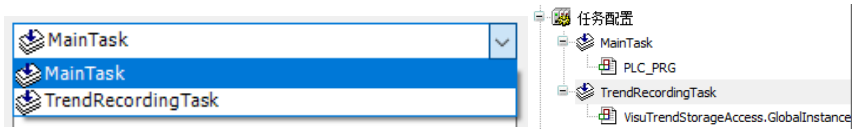
单击左侧树形菜单第一列配置采样任务、采样触发等

图表 4.4-117



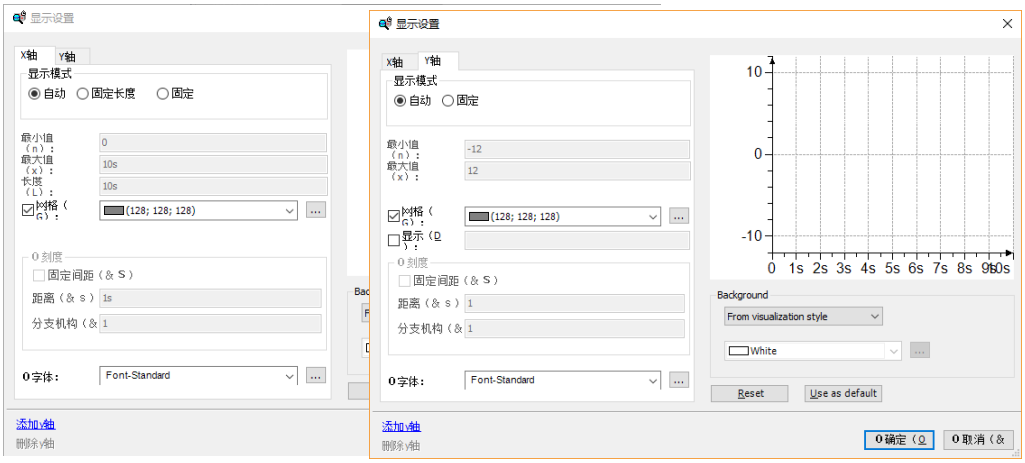
① 配置采样周期，提供了两种采样周期供用户选择，分别是“MainTask”和“TrendRecordingTask”，其中“MainTask”是主程序任务周期，“TrendRecordingTask”在用户添加了 Trace 之后会在任务配置下自动生成

图表 4.4-118



② 显示设置，在显示设置中可以配置跟踪画面中 X 轴和 Y 轴，可以采用默认的自动方式或者选择固定长度的方式。

图表 4.4-119



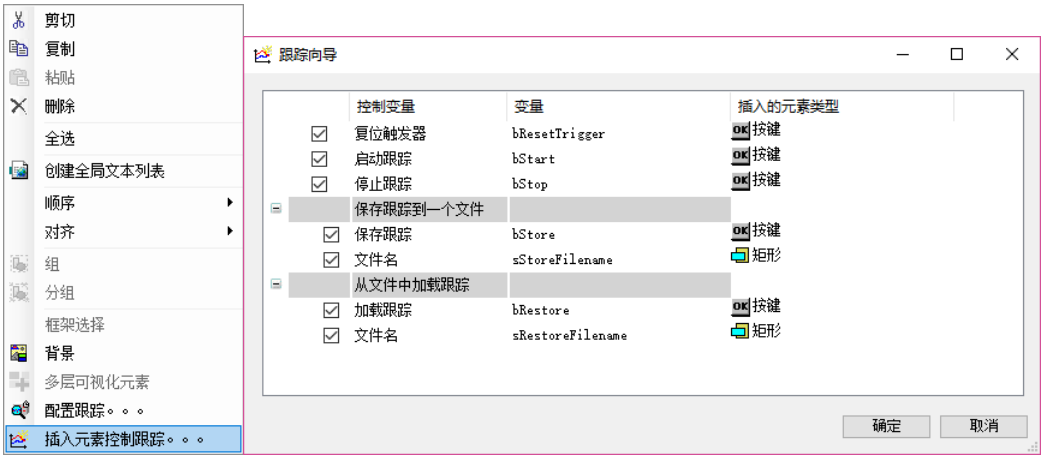
③ 在高级跟踪设置中，用户可以选择 X 轴经过多少个周期进行一次采样

图表 4.4-120

④ 通过主配置界面中的“从跟踪中复制”可以复制其他跟踪实例中的变量配置

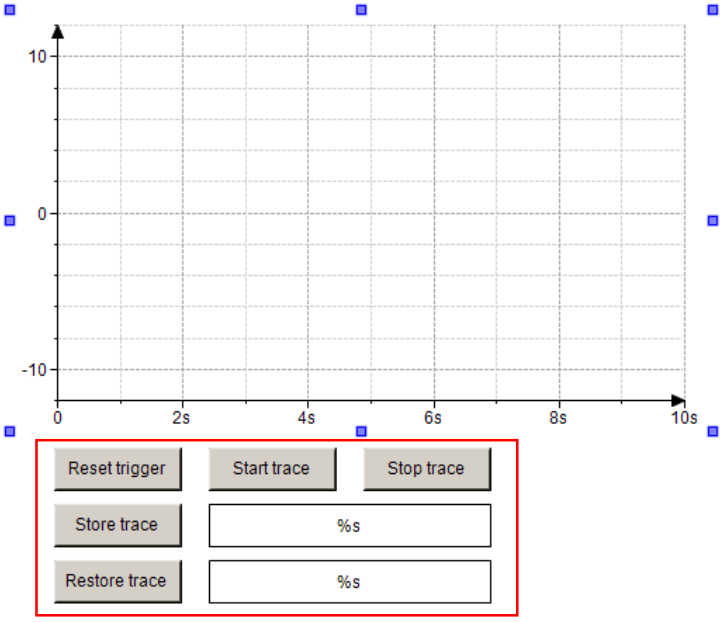
选中跟踪控件后，单击鼠标右键，在弹出的快捷菜单中选择“插入元素控制跟踪”，在弹出“跟踪向导”的界面中，可以选择需要添加的按钮及显示框，默认将所有控制插件都勾选上，系统会自动创建控制变量并在界面中生成按钮及显示框

图表 4.4-121



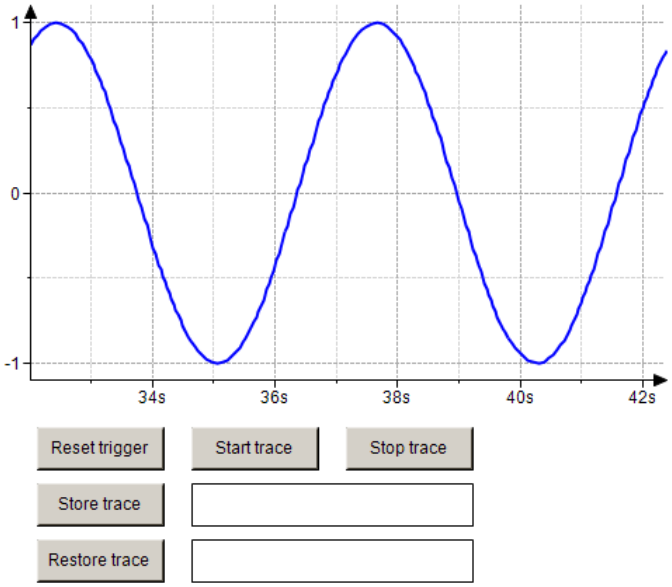
勾选所有的控制插件后生成的界面如下：

图表 4.4-122



运行程序后查看在线运行效果：

图表 4.4-123



ActiveX 元素

ActiveX 在可视化界面中可以实现 ActiveX 控制。该元素可用于 Windows32 位系统的界面中。在工具箱的“特殊控制”中找到“ActiveX 元素”控件，并将其拖拽至可视化编辑区域。

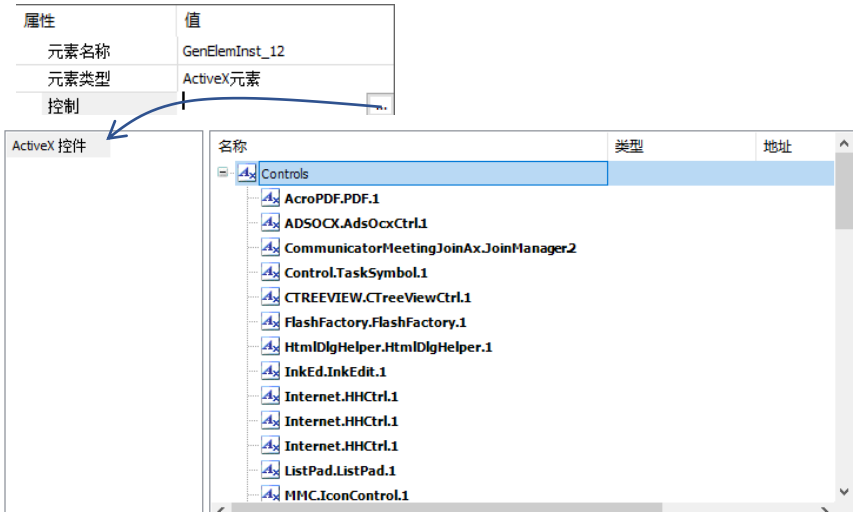
在可视化编辑界面中选中添加的“ActiveX 元素”，在右侧属性菜单中可以提供三种不同的调用方式：

- 初始调用：初始化时要进行调用的方法可以在这里进行定义，只在第一个任务周期内进行处理。
- 循环调用：可视化中周期调用的方法可以在该属性下进行定义，可以在每个可视化执行周期事件中进行更新。
- 条件调用：附加的一个“条件调用”可以被关联。条件调用可以在可视化事件更新的时候进行调用。不同于初始化调用或者循环调用，条件调用可以关联到属性方法，并且只在条件调用的上升沿进行处理。

下面介绍 ActiveX 元素的基本使用方法。

在工具箱“特殊控制”中选择“ActiveX 元素”  并将其拖拽到可视化编辑区，选中添加的控件后，在属性菜单中找到“控制”选项，可以通过其后的“输入助手”图标选择不同的插件类型

图表 4.4-124



选择需要调用的插件后，用户需要配置“ActiveX 元素”的触发方式，可以选择“初始调用”、“循环调用”和“条件调用”三种，通过属性菜单不同调用方法下的“创建新的”按钮创建触发方式，在新创建的方法下完成变量映射。

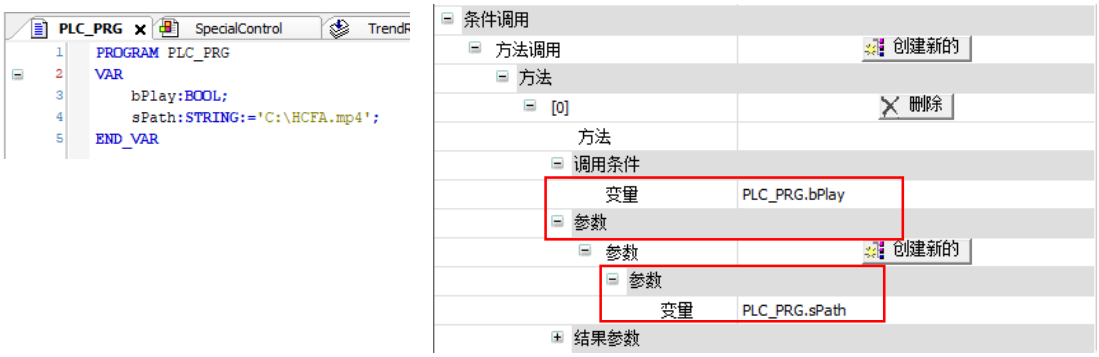
图表 4.4-125

【例7】 通过 ActiveX 元素，实现相应变量收到上升沿信号后，播放视频

在工具箱“特殊控制”中选择“ActiveX 元素”，在其属性菜单中找到“控制”通过“输入助手”添加“WMplayer.OCX.7”

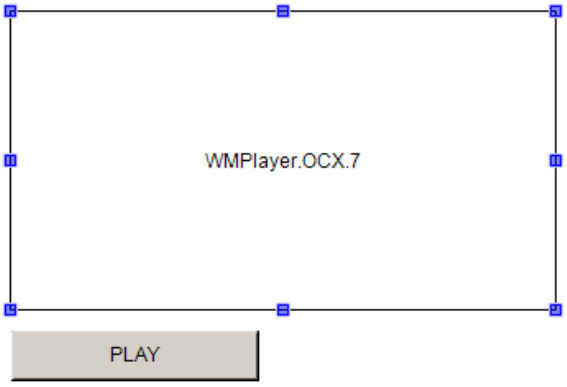
根据要求选择“条件调用”方式，增加新的方法调用后，在“调用条件”的“变量”中添加触发变量“bPlay”并在“参数”下新建参数，添加变量“sPath”作为系统参数，将视频文件的路径通过该变量索引。

图表 4.4-126



添加按钮并映射触发变量“bPlay”方便在界面中触发播放视频，最终界面编辑如下：

图表 4.4-127



登录并查看在线运行效果如下：

第5章 简单 PLC 项目的创建

5.1 启动 CODESYS

当用户对控制系统有了整体规划后，就需要开始为控制器创建项目以完成控制目的，首先需要用户连接上 Q1，在 Q1 平台内编辑相应的 IO 组态配置后按照需求编辑逻辑程序，最后进行在线调试后将项目下载到控制器内，本章将会介绍如何创建一个自己的项目。

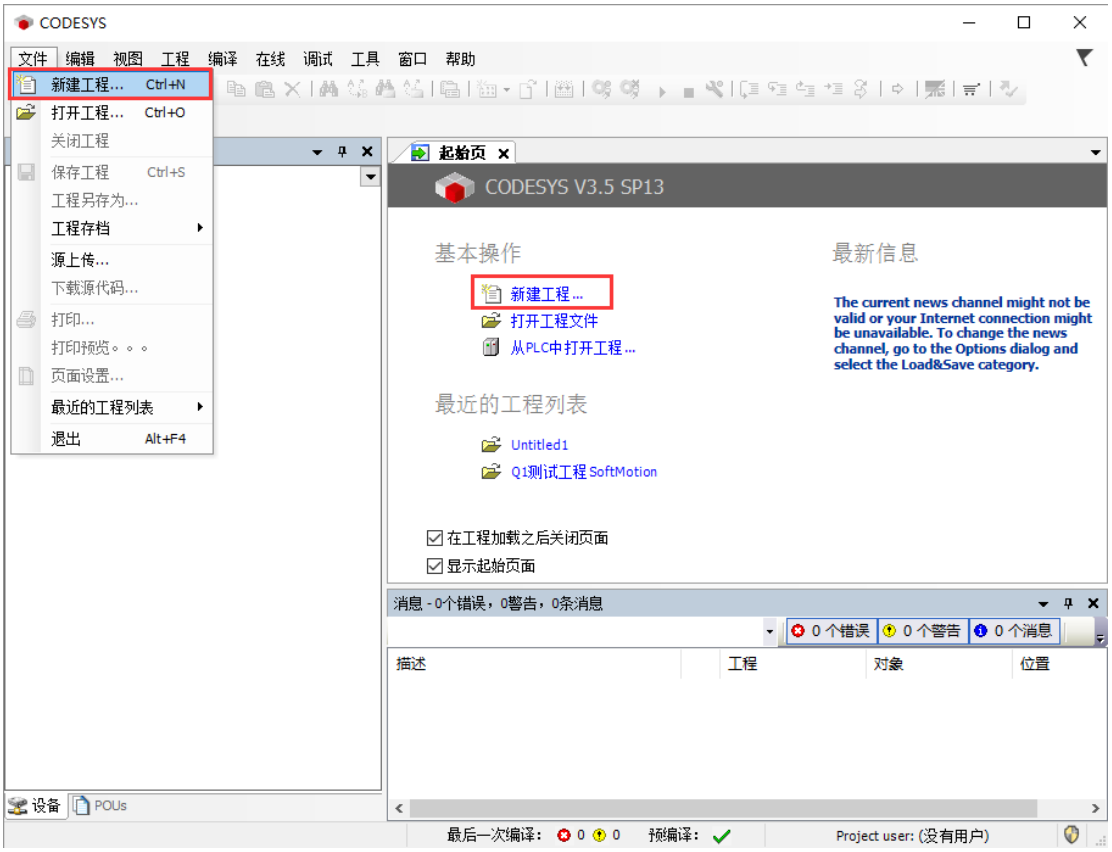
在 PC 平台安装好 CODESYS 软件之后，在桌面可以找到 

快捷方式，双击打开。

5.2 新建 CODESYS 项目

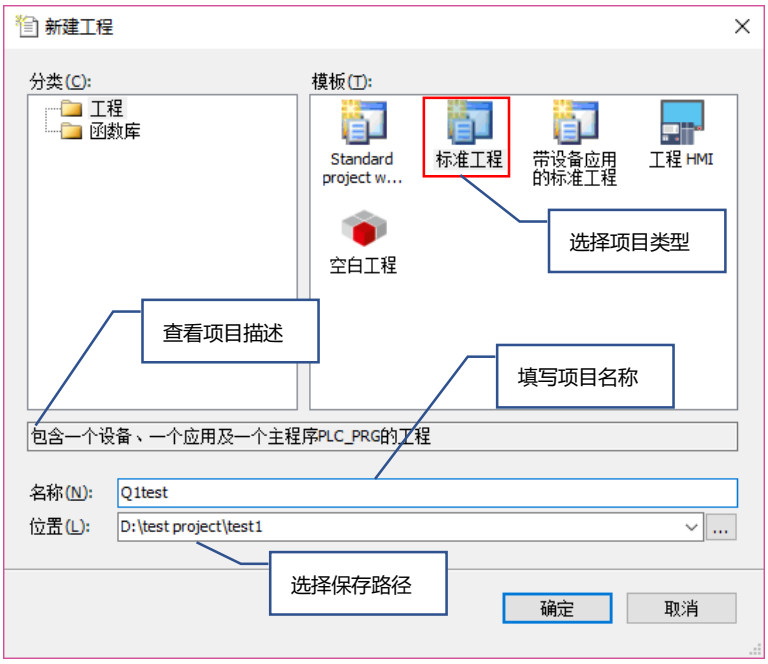
打开 CODESYS 软件后，在软件的起始页找到“新建工程”或者在菜单栏 “文件”中选择“新建工程”对于之前进行编辑的项目用户也可以直接从“最近的工程列表”中直接选择打开

图表 5.2-1



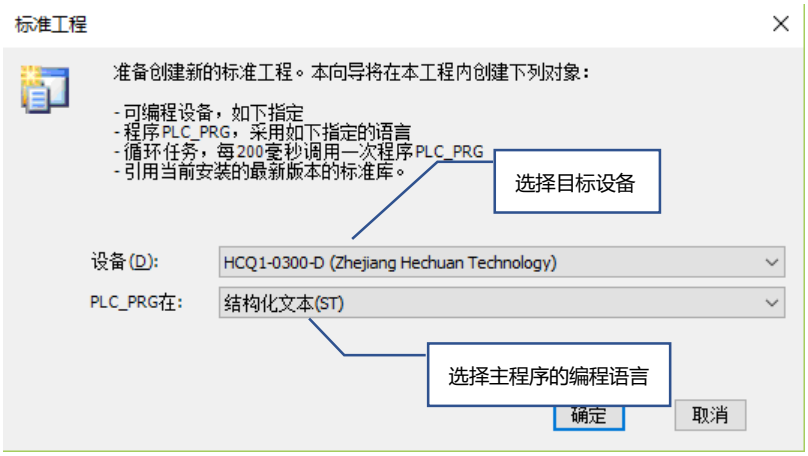
在弹出的“新建工程”对话框中选择对应的“模板”，对于每一个模板的说明可以在选择该模板后在下方看到对应说明，输入名称和存储路径后，点击“确定”

图表 5.2-2



按照 CODESYS 默认的引导，选择目标设备及主程序 PLC_PRG 的编程语言，Q1 设备默认未安装，所以首先需要进行 Q1 设备的安装，否则无法选择正确的目标设备，新设备描述文件安装步骤参考 3.2

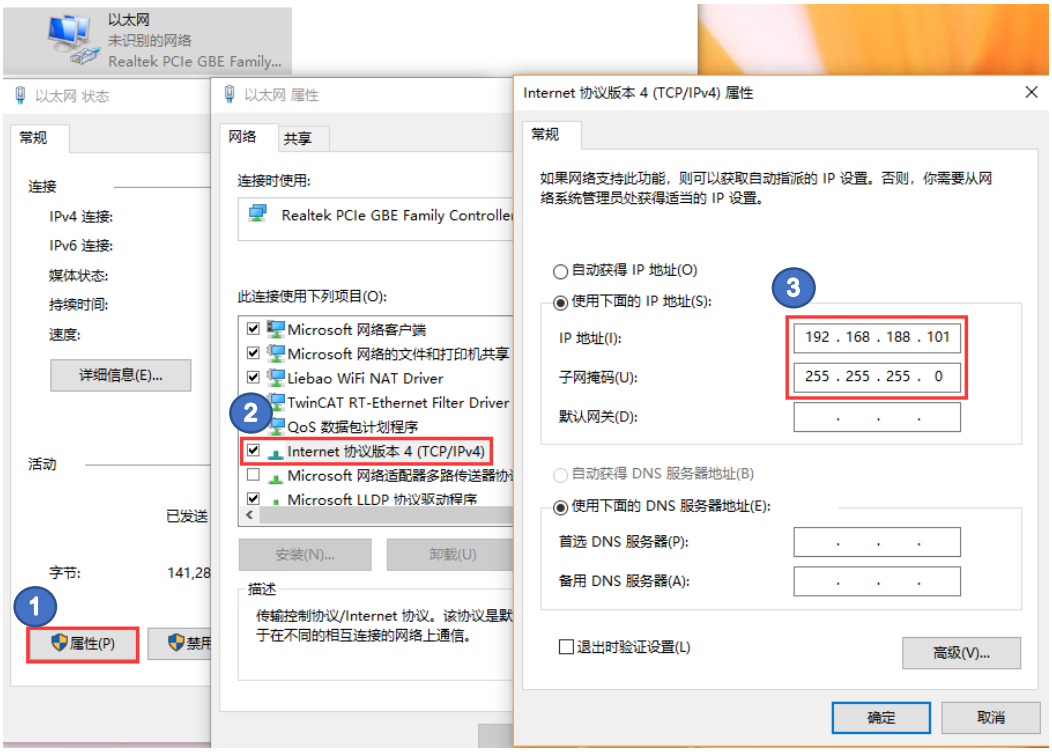
图表 5.2-3



5.3 和 Q1 建立通讯

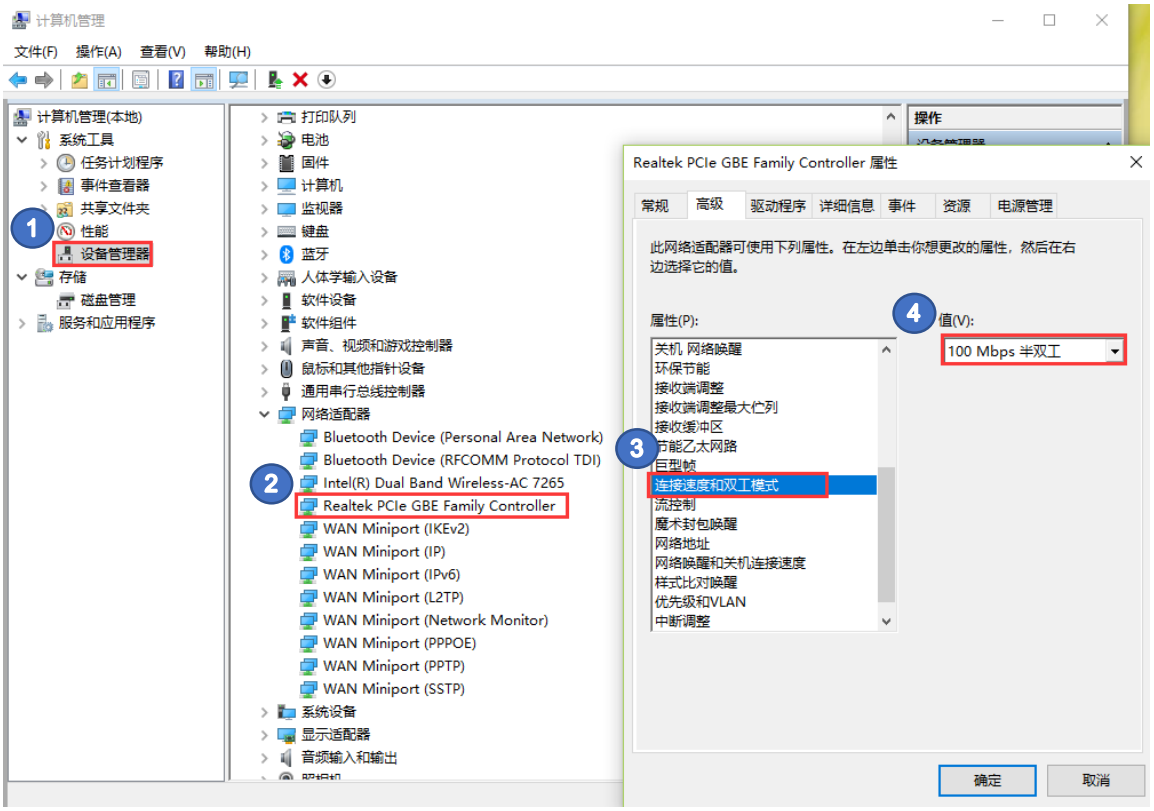
Q1 控制器 Port1 默认的 IP 地址是：192.168.188.100 子网掩码：255.255.255.0，在 PC 端网络适配器中修改 IP 地址到同一个网段（Port2 默认的 IP 地址为 192.168.88.xxx 子网掩码：255.255.255.0）。

图表 5.3-1



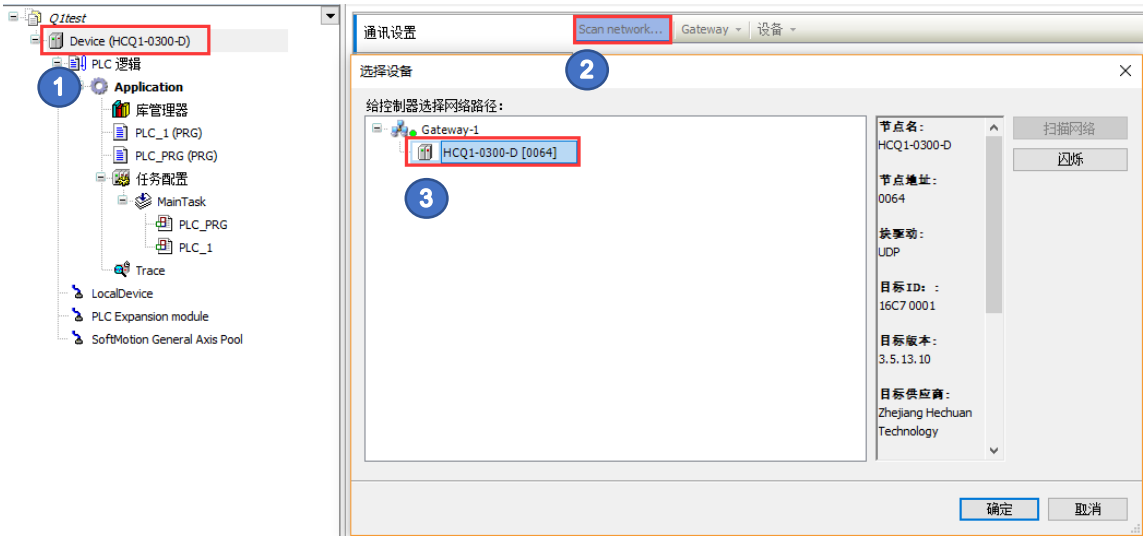
在 PC 端的设备管理器中修改网卡的连接速度和双工模式为 100Mbps 半双工

图表 5.3-2



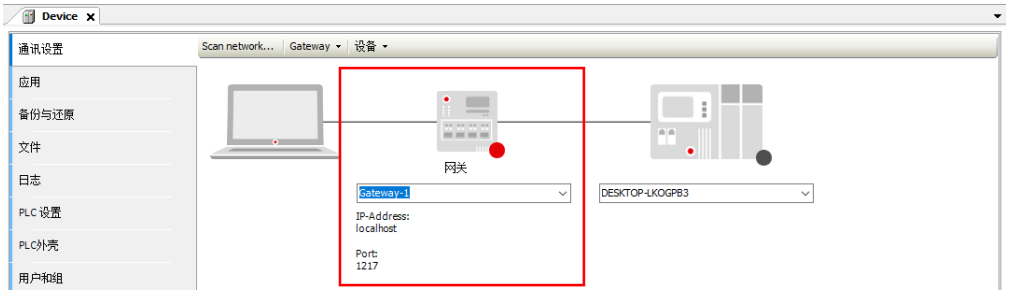
完成 PC 端网卡的相关设置后，双击 CODESYS 软件新建项目下左侧树形菜单“Device”进入通讯设置，确保网关正确开启后，点击”Scan network”，扫描到 Q1 之后选中设备，点击确定进行添加

图表 5.3-3



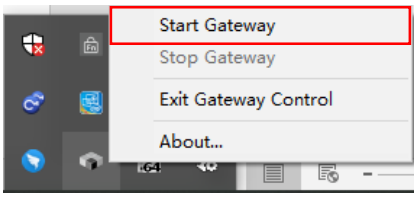
如果 CODESYS 网关未开启，在“通讯设置”页面中会显示成红色，此时需要用户自行开启

图表 5.3-4



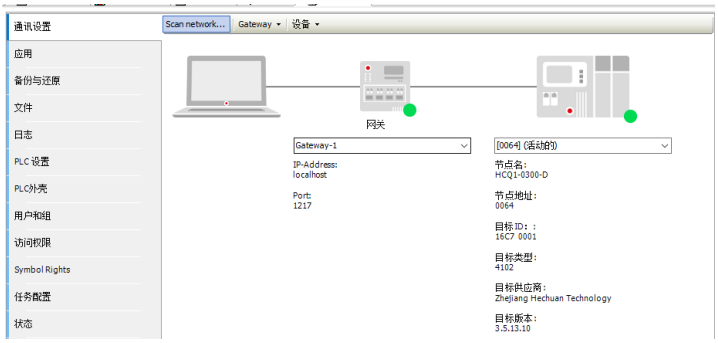
在 PC 端右下角找到 CODESYS 图标，右击，选择“StartGateway”，启动网关后，执行扫描和添加

图表 5.3-5



正确添加的设备显示如下

图表 5.3-6



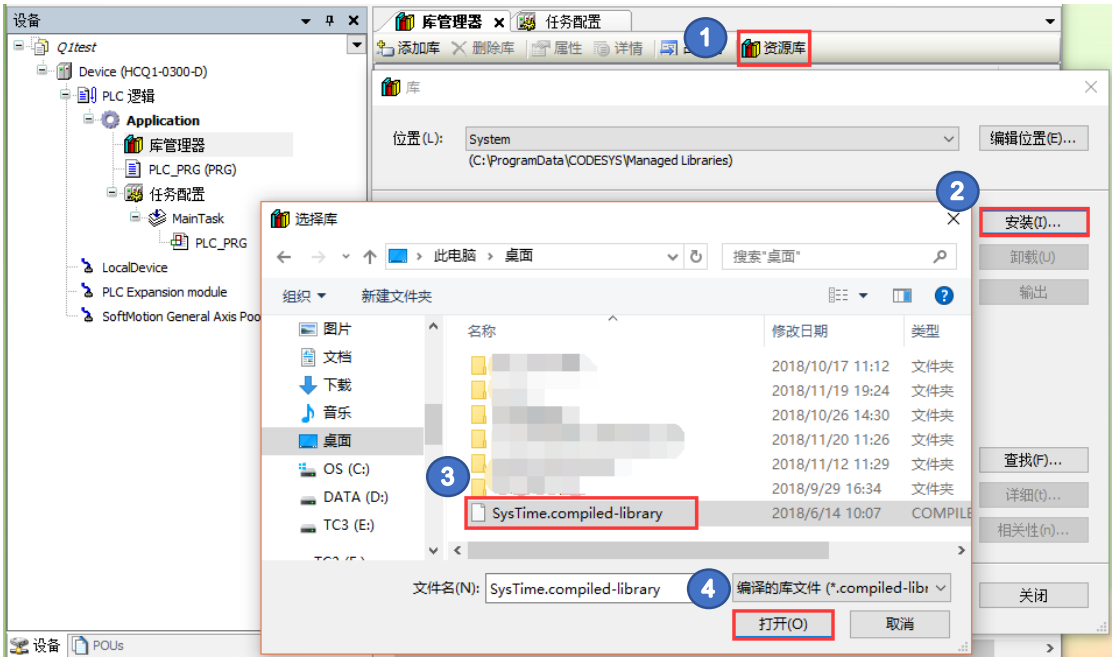
5.4 PLC 项目的创建

一个标准的 PLC 项目，包含库、任务和程序，可以添加跟踪对程序中变量进行实时监控，为了方便调试，用户也可以选择建立可视化界面。PLC 程序文件的创建，是运行结构运行顺序的建立，也是编程模式的建立。一个新建的 PLC 项目，系统会默认分配一个连续性任务，任务中包含有一个默认的程序“PLC_PRG”。

5.4.1 库的添加

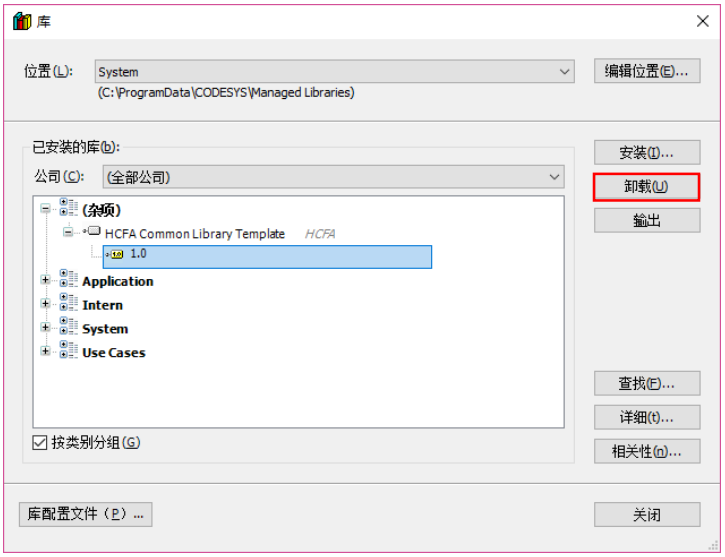
如果一个项目需要使用外部库，需要用户自行安装库文件，双击左侧树形菜单”库管理器”，进入”库文件管理器”后，找到”资源库”→”安装”→”需要安装的库文件”→”打开”，即可完成库文件的安装.

图表 5.4-1



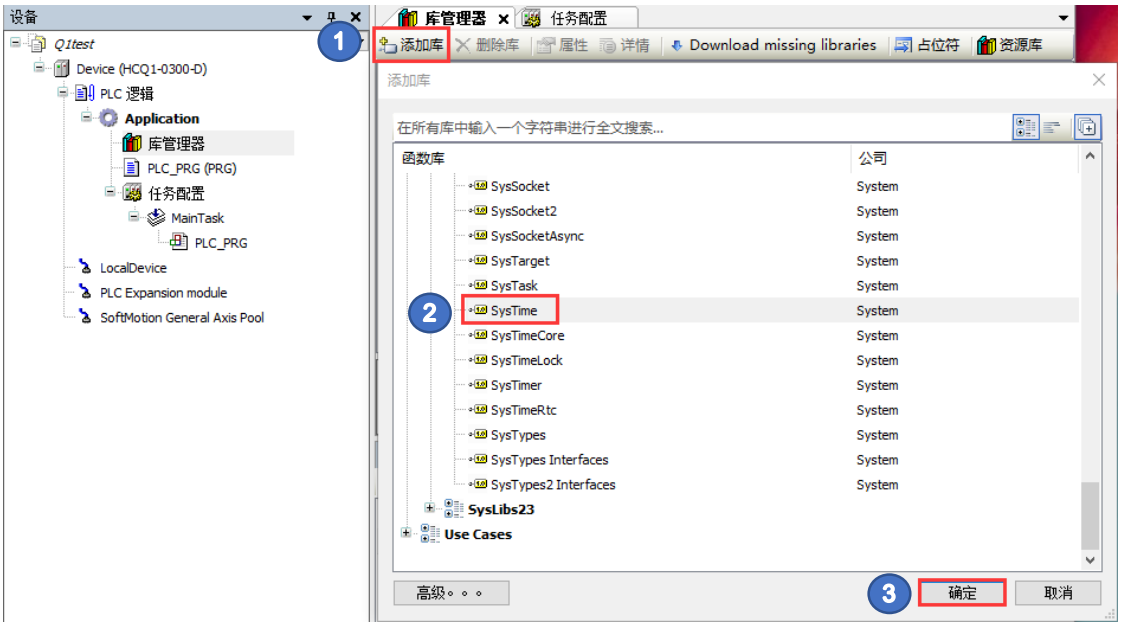
CODESYS 可以自动识别库版本并支持多个版本的库文件之间的切换，如需卸载，只要选中需要卸载的库文件在同一个界面下，点击卸载按钮即可

图表 5.4-2



对已经安装好的库，可以在添加库按钮中对其进行调用，成功调用的库所包含的所有功能块和函数都可以在程序中使用，需要注意的是，所有调用的库文件都会占用 PLC 内存资源，用户可以根据自己的需求添加和使用库文件

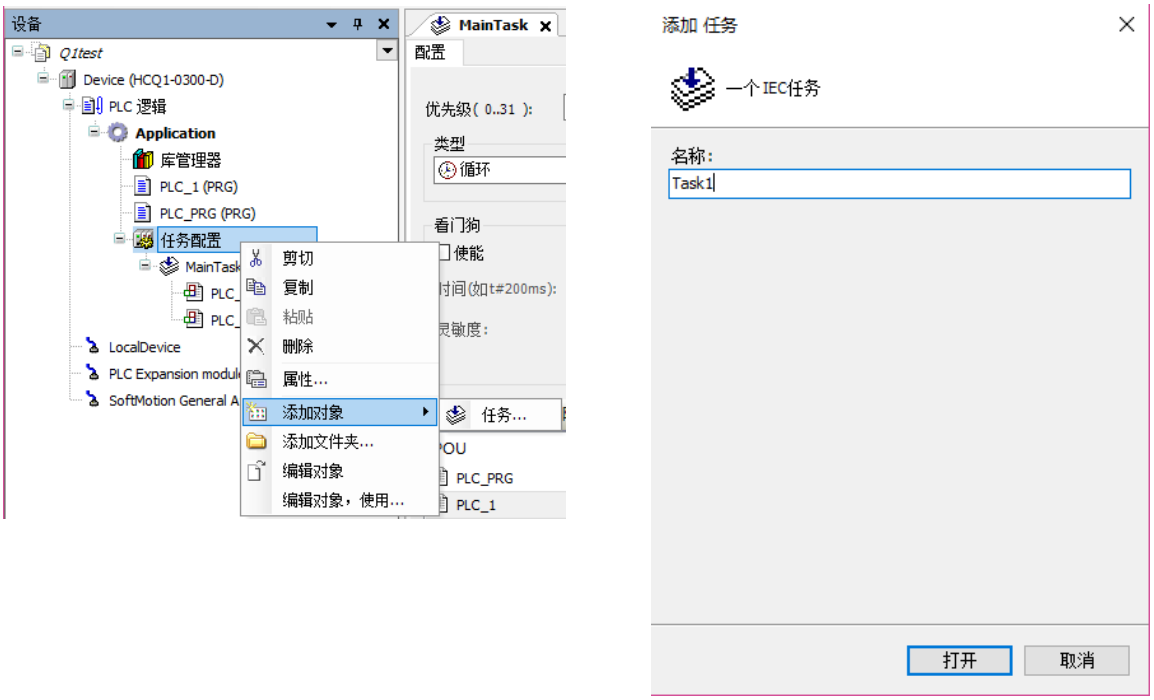
图表 5.4-3



5.4.2 任务的设置

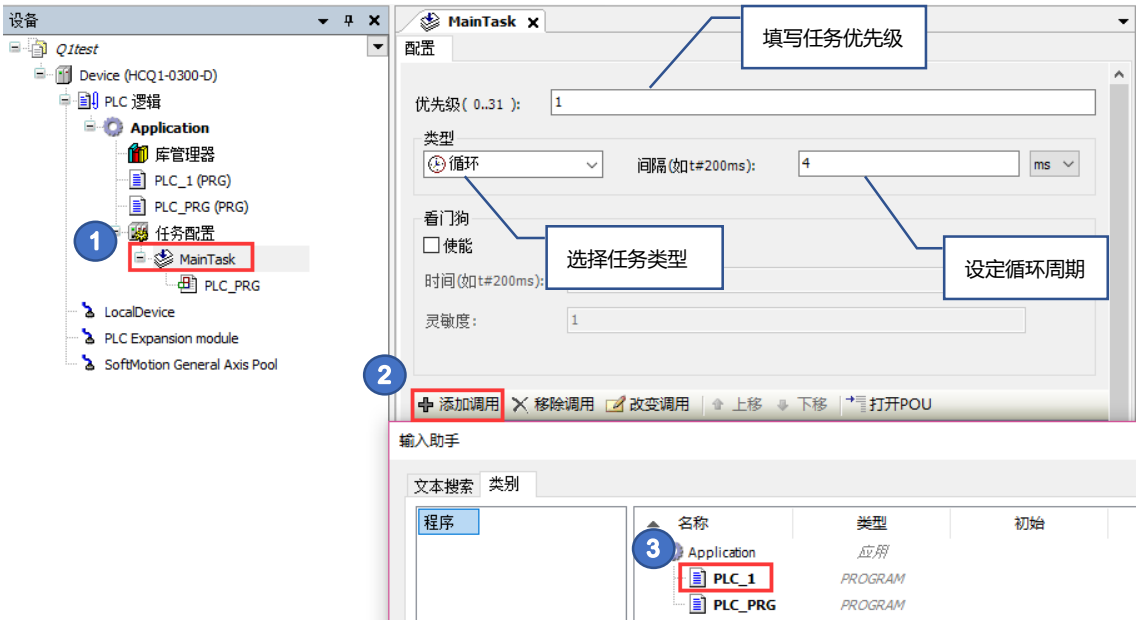
在树形菜单任务配置中可以进行任务的管理，新建一个标准的 PLC 项目会自动生成一个循环执行的任务，该任务自动关联 PLC_PRG，任务周期默认是 4ms，优先级为 1，PLC 程序只有被任务调用才会参与编译和实际执行。右击任务配置→添加对象→任务，定义任务名称后完成新任务的创建，不同类型的任务最多可以创建 100 个，按照用户设定的优先级顺序执行，数字越小优先级越高，优先级相同的情况下，按照在任务配置当中的顺序从上往下执行。

图表 5.4-4



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 MainTask→添加调用→需要调用的 PLC 程序点击确定完成调用

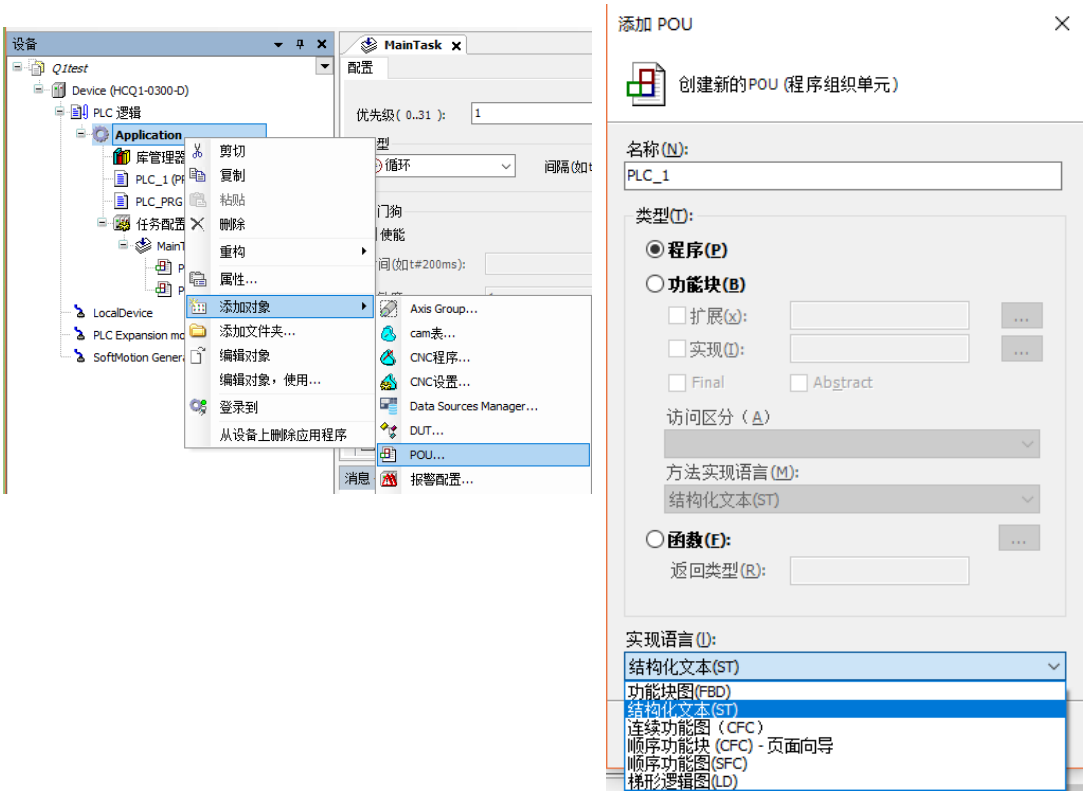
图表 5.4-5



5.4.3 PLC 程序的编写

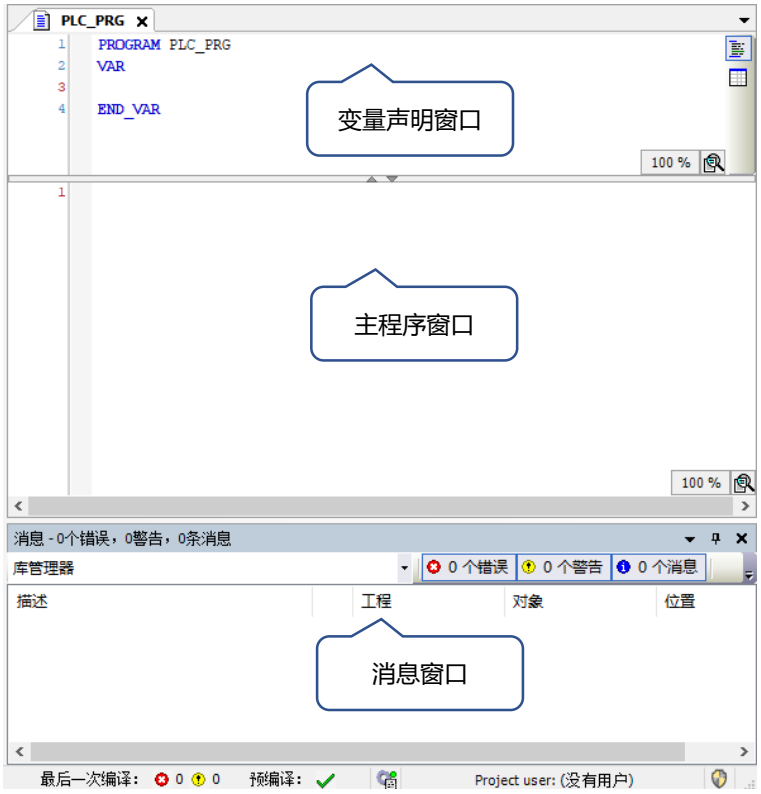
新建 POU（Program Organization Unit ），首先右击 “Application” → “添加对象” → “POU”，在弹出的 “创建新的 POU” 对话框中填写 POU 的名称后选择程序块类型，CODESYS 支持六种编程语言，本次示例选择结构化文本（ST）编写程序

图表 5.4-6



双击 PLC_PRG 进入编程界面，从上往下分别是变量声明窗口，主程序窗口，消息窗口，程序编译产生的错误、警告和编译信息都会显示在消息窗口中

图表 5.4-7



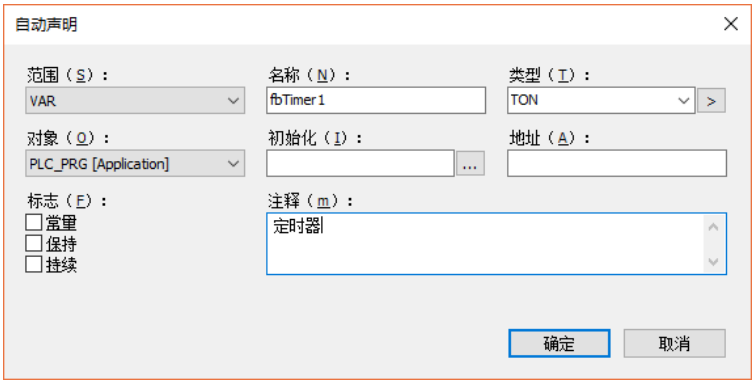
编写示例程序为可调占空比的方波如下：

首先对程序中需要使用的变量按照 IEC61131-3 编程标准进行声明，

变量声明表达式为：变量名：变量的数据类型：=初始值；初始值可不给定，变量存在默认初始值，通过“//”进行单行注释，支持中文注释，但是目前中文注释存在光标位置显示不正确的问题，暂时建议使用英文注释

在变量声明窗口中 Shift+F2，可以弹出自动声明对话框，其中名称和类型必须填写

图表 5.4-8



范围：变量作用域

名称：变量名称

类型：变量数据类型

对象：变量所在应用

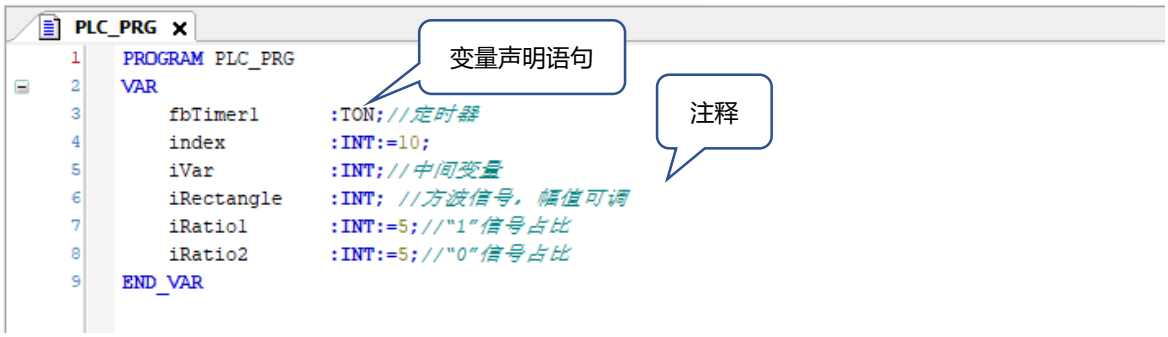
初始化：变量初始值

地址：变量和外部硬件点之间的映射关系

标志：可以设置变量类型为常量、保持和持续型

注释：注释，格式为（*注释内容*）

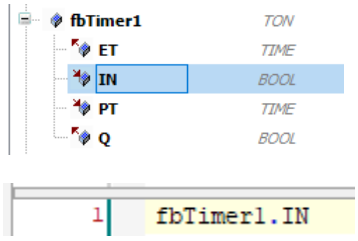
图表 5.4-9



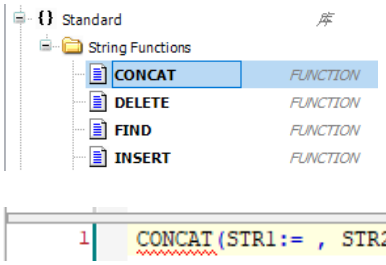
在主程序窗口编写示例程序，正确声明的变量可以在主程序窗口中通过 F2 调用，调用时需要注意调用类别

变量：单个变量，需声明

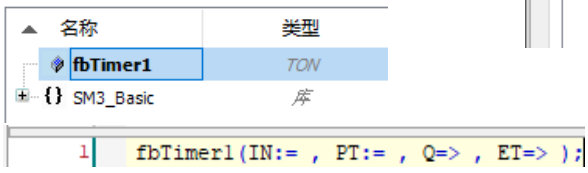
图表 5.4-10

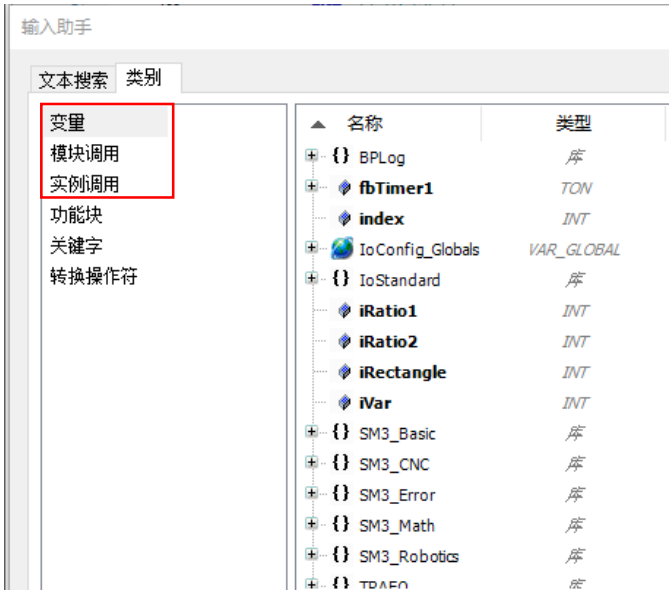


模块调用：函数调用，无需声明，直接调用



实例调用：功能块语句，需声明





按照以上规则编写示例程序如下：

图表 5.4-11

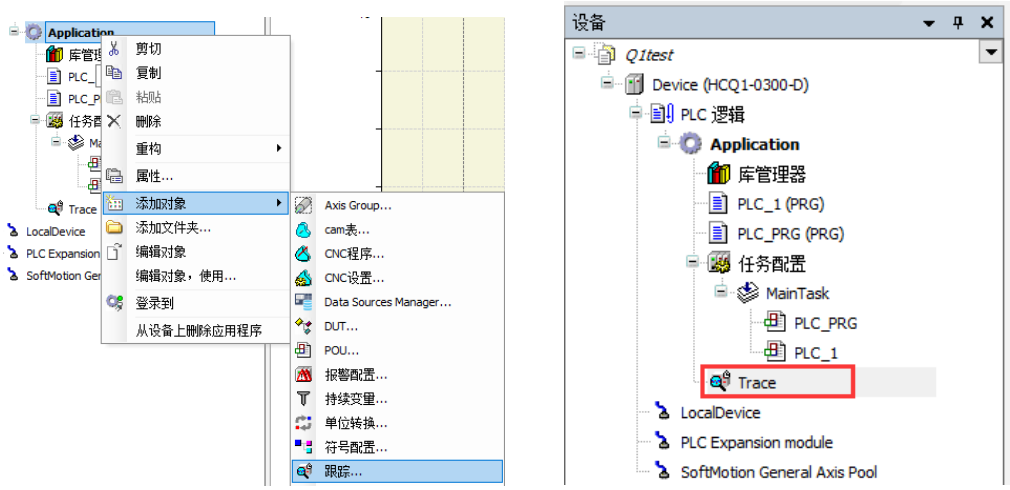
```
(*示例程序是一个可调占空比的方波，通过给定常数Ratio1和Ratio2可以设置方波iRectangle的"1""0"的周期长度和所占比例*)
    fbTimer1(IN:=NOT fbTimer1.Q, PT:=T#500MS  );
CASE index OF
    10:
    irectangle:=1;
    IF iVar<iRatio1 AND fbTimer1.Q THEN
        iVar:=iVar+1;
    ELSIF iVar>iRatio1 OR iVar=iRatio1 THEN
        index:=20;
        iVar:=0;
    END_IF

    20:
    irectangle:=0;
    IF iVar<iRatio2 AND fbTimer1.Q THEN
        iVar:=iVar+1;
    ELSIF iVar>iRatio2 OR iVar=iRatio2 THEN
        index:=10;
        iVar:=0;
    END_IF
END_CASE
```

5.5 添加跟踪以监控程序变量

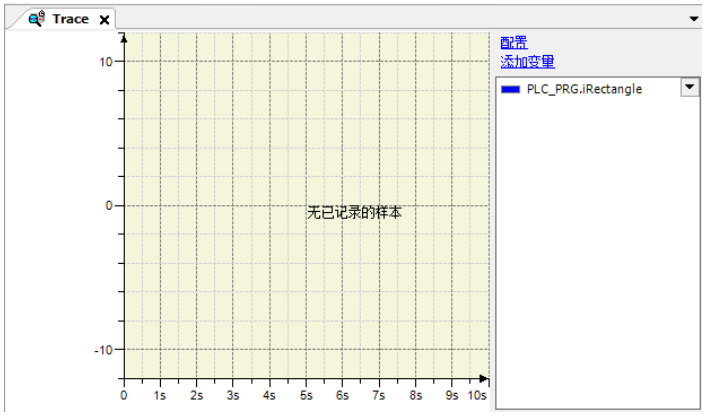
右击 Application→添加对象→跟踪，成功添加后在左侧树形菜单会出现 Trace

图表 5.5-1



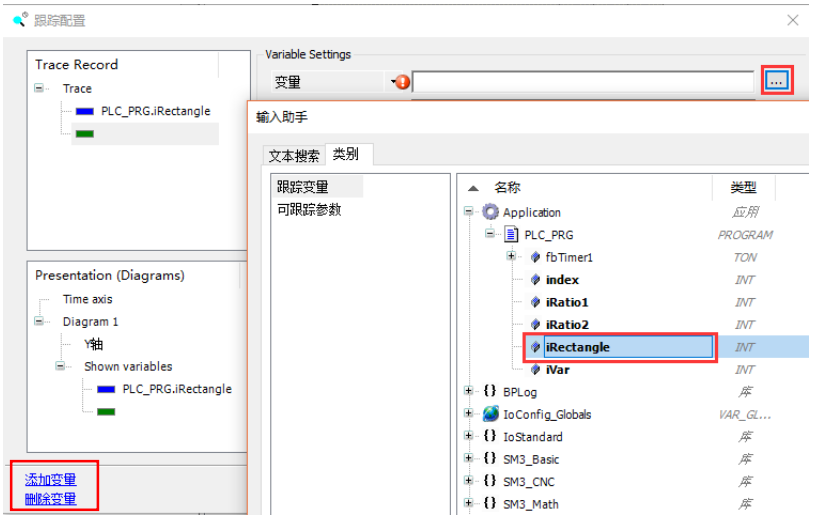
双击进入 Trace 页面进行配置

图表 5.5-2



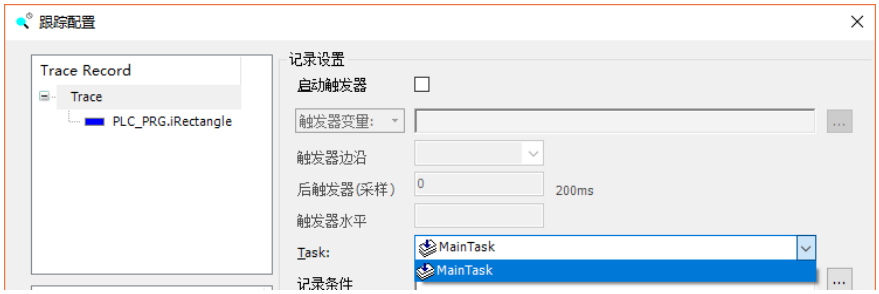
单击 Trace 界面右上角配置按钮，进行监控变量的添加删除和采样周期的设置，打开跟踪配置页面后，通过左下角添加/删除变量按钮进行变量的添加和删除

图表 5.5-3



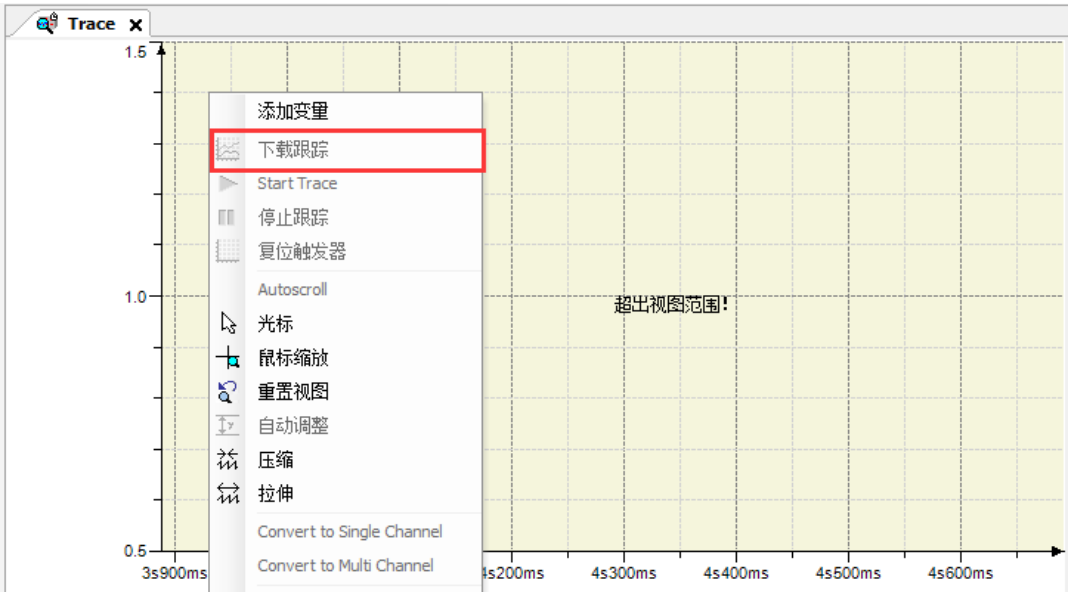
通过单击对话框左侧树形菜单 Trace 配置采样周期所跟随的 Task，下拉进行选择，示例选择 MainTask

图表 5.5-4



完成 Trace 的基本配置后，需要将程序登录并正常运行后才可以在 Trace 界面任意位置下载并启动跟踪（Start Trace）并观察变量运行轨迹否则该选项默认为灰色，无法选择

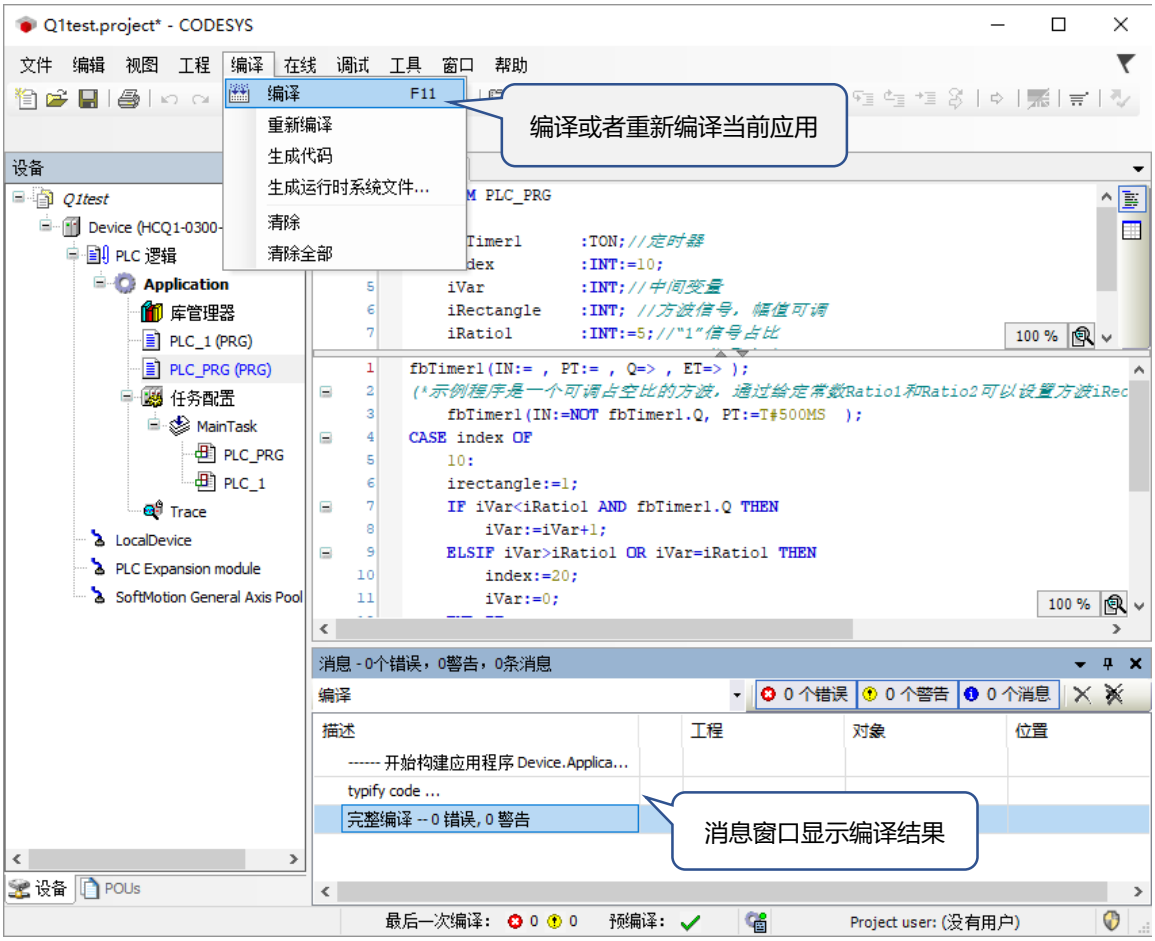
图表 5.5-5



5.6 程序下载和在线监控

5.6.1 PLC 程序编译和下载

PLC 程序编写完成后，在下载之前需要对程序进行编译。编译命令会对编写的程序进行语法检查，要求创建的 POU 已经添加到 Task，否则不参与编译。选择菜单栏编译→编译，执行编译命令后消息窗口会显示编译结果，通过选择错误、警告和消息可以单独显示需要查看的信息。



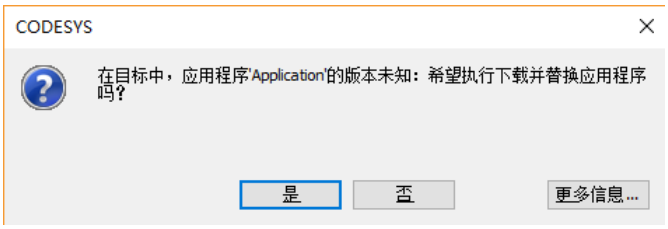
5.6.2 登录和运行

编译显示程序无错误后，执行连接登录指令，在工具栏找到登录图标  单击进入程序在线监控状态

图表 5.6-1

弹出对话框提示：下载的应用会覆盖 PLC 中原有应用，是否执行下载，点击“是”

图表 5.6-2

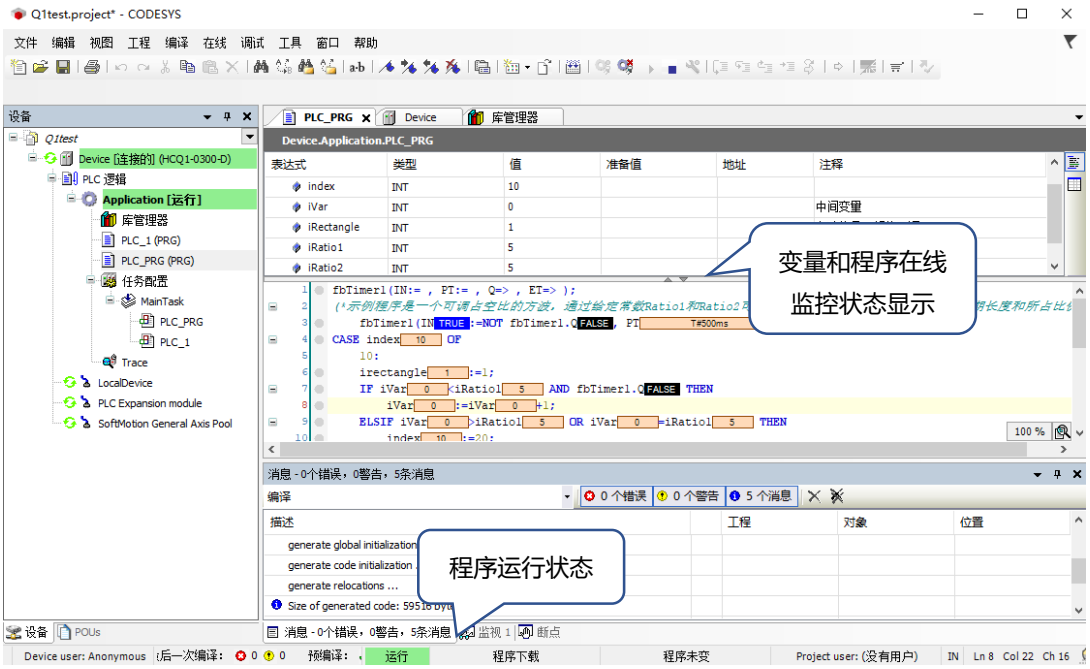


成功登录后，点击运行图标，运行程序

图表 5.6-3

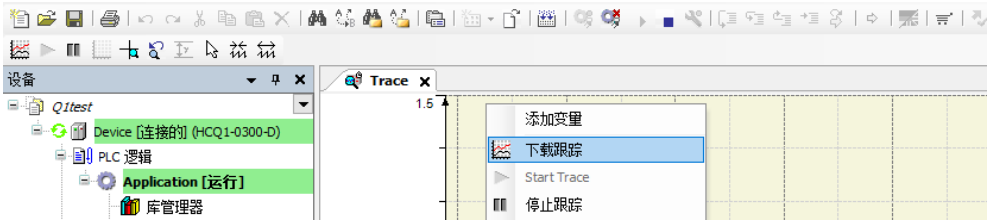
程序成功登录运行界面如下

图表 5.6-4



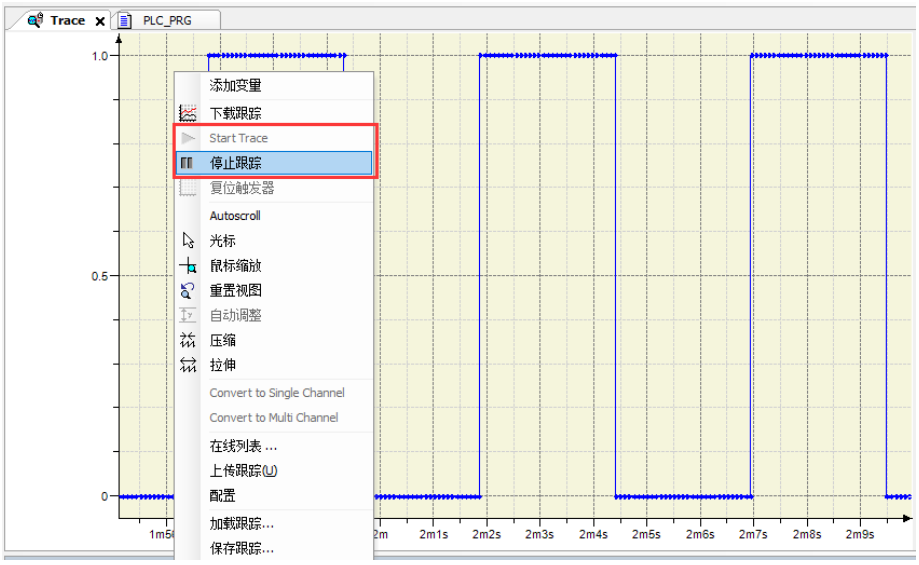
启动 Trace，采集变量的运行轨迹

图表 5.6-5



在 Trace 页面任意位置右击同样可以暂停或者启动跟踪，通过鼠标可以移动观察历史跟踪，滚轮可以进行跟踪波形的缩放

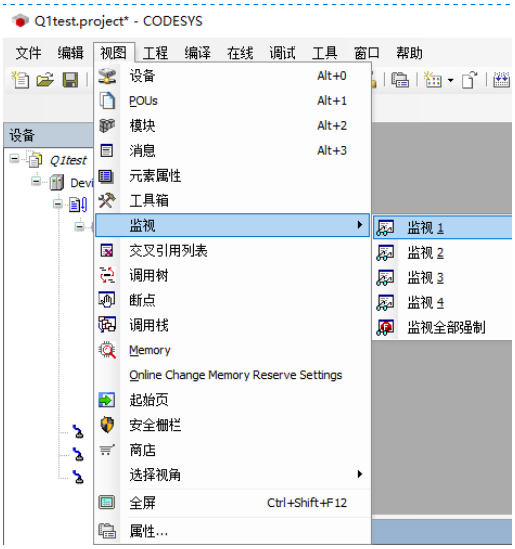
图表 5.6-6



5.6.3 在线监控

程序成功登录运行后，主程序窗口和声明窗口都会显示变量的实际值。如果用户想要同时监视多个 POU，可以把需要观察的变量集合起来，在菜单栏找到视图→监视→监视 1，则程序会自动建立监视 1 列表，可以创建 4 个监视列表。

图表 5.6-7

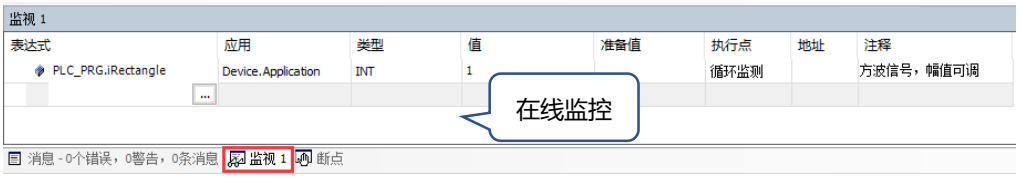


打开监视 1，单击表达式下的空栏，可以通过点索引的方式直接调用不同 POU 中的变量或全局变量，也可以通过点击右侧 ... 展开输入助手添加需要在线监视的变量。

图表 5.6-8

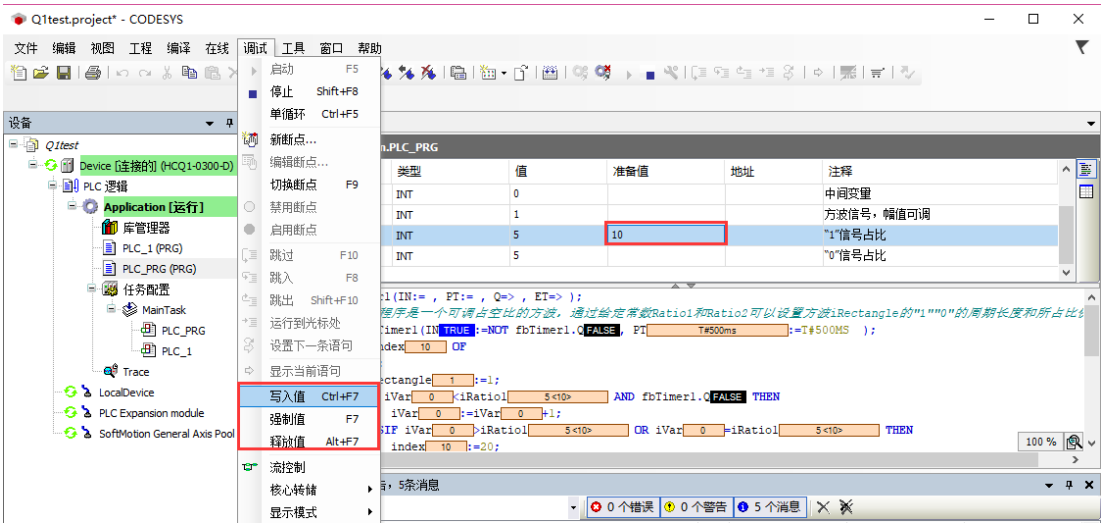
配置完成后在程序正常运行过程中可以通过消息窗口下的“监视 1”选项卡查看变量的实时值，以及对当前变量进行赋值等操作，即使是程序正常运行过程中也可以手动添加或者删除监控变量，非常方便

图表 5.6-9



在登录状态下，可以在准备值中输入想要在下一个周期写入的数值，该操作在变量声明窗口，主程序窗口和监控窗口都可以完成，输入完成后，需要通过菜单栏调试→写入值，进行正常的赋值操作。在 CODESYS 中用户给变量赋值有两种方式，括号内为快捷键：写入值（Ctrl+F7）和强制值（F7），通过强制值写入的数值左侧会有  标识，可以通过释放值将其释放还原。

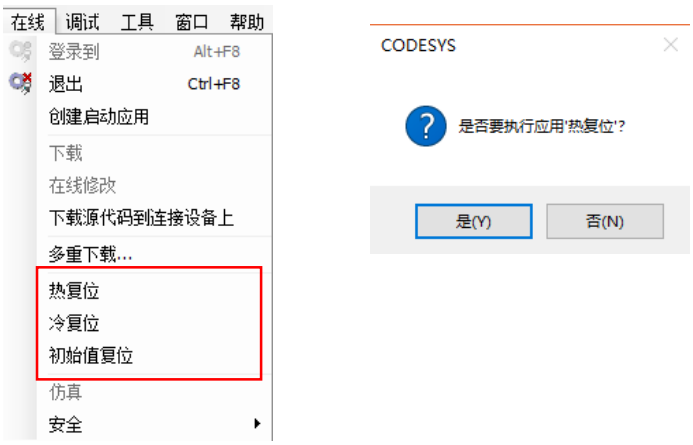
图表 5.6-10



5.6.4 复位功能

在调试中，如果用户希望对程序进行复位操作，CODESYS 提供了以下三种复位方式：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。

图表 5.6-11



注： X：保留值 O：初始值

图表 5.6-12

在线命令	VAR	VAR RETAIN	VAR RETAIN PERSISTENT
热复位	O	X	X
冷复位	O	O	X
初始值复位	O	O	O
下载	O	O	X
在线改变	X	X	X

5.7 仿真

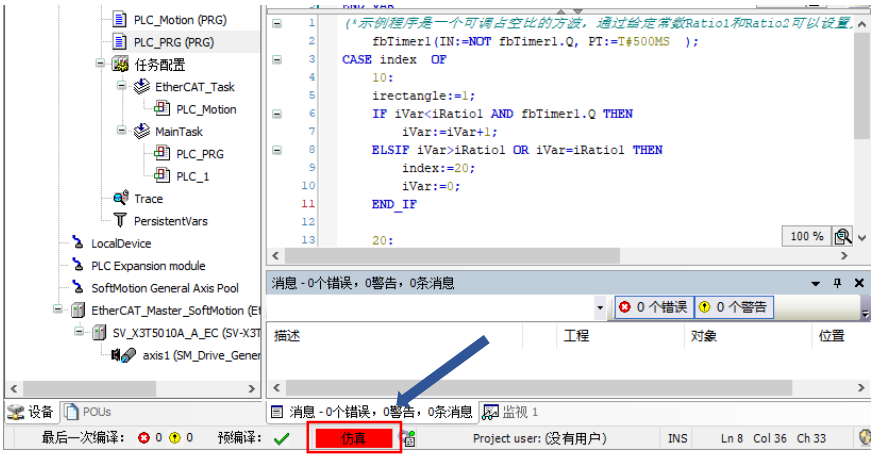
CODESYS 在用户没有实际硬件设备的情况下允许用户使用仿真功能，用 PC 仿真实际的 PLC 程序，以保证用户在现场调试前可以对程序进行完整的测试，更早的发现程序的逻辑错误，缩短开发周期。在程序的仿真过程中，不需要使用 PLC 硬件，用户可以直接在自己的 PC 上对编辑好的程序进行编译、登录和运行等操作。

下面介绍 PLC 仿真的使用步骤，在菜单栏找到“在线”选择“仿真”，就可以进入仿真模式，勾选后在仿真左侧会出现

图表 5.7-1

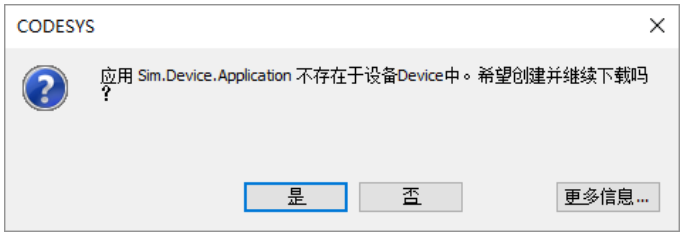
进入仿真模式后，在界面下方会出现红色仿真图标显示

图表 5.7-2



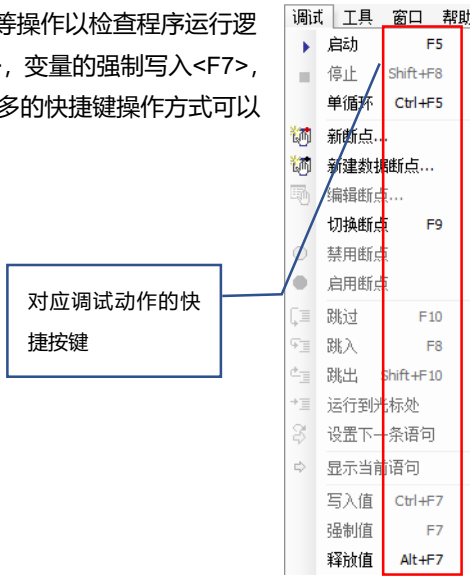
此时用户可以直接执行编译，消息窗口显示没有错误后，下载程序，开始下载时，系统弹出对话框提示仿真器中没有程序，单击“是”按钮继续

图表 5.7-3



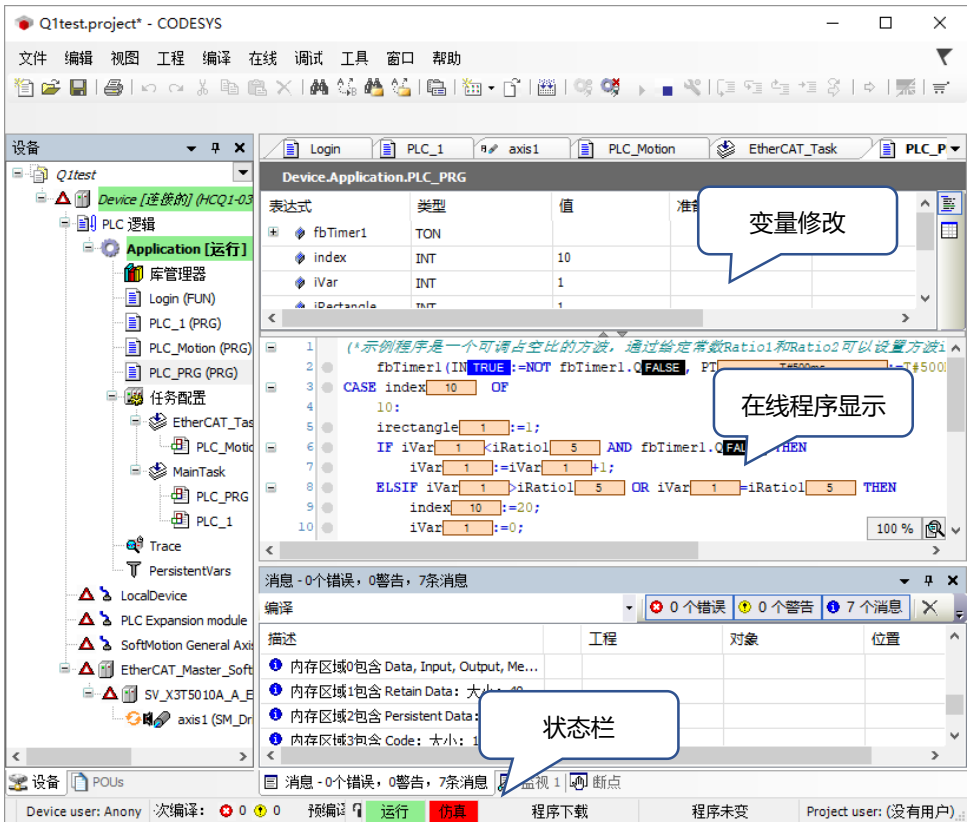
程序在仿真器中运行时，用户可以执行的操作和在实际控制器中运行可以执行的操作没有差别，用户可以对程序中变量进行修改、强制等操作以检查程序运行逻辑是否出错。其中，常用的功能包括变量的写入<Ctrl+F7>，变量的强制写入<F7>，取消强制写入<Alt+F7>等都可以直接使用快捷键实现，更多的快捷键操作方式可以参考菜单栏“调试”下的选项。

图表 5.7-4



仿真模式程序运行效果如下：

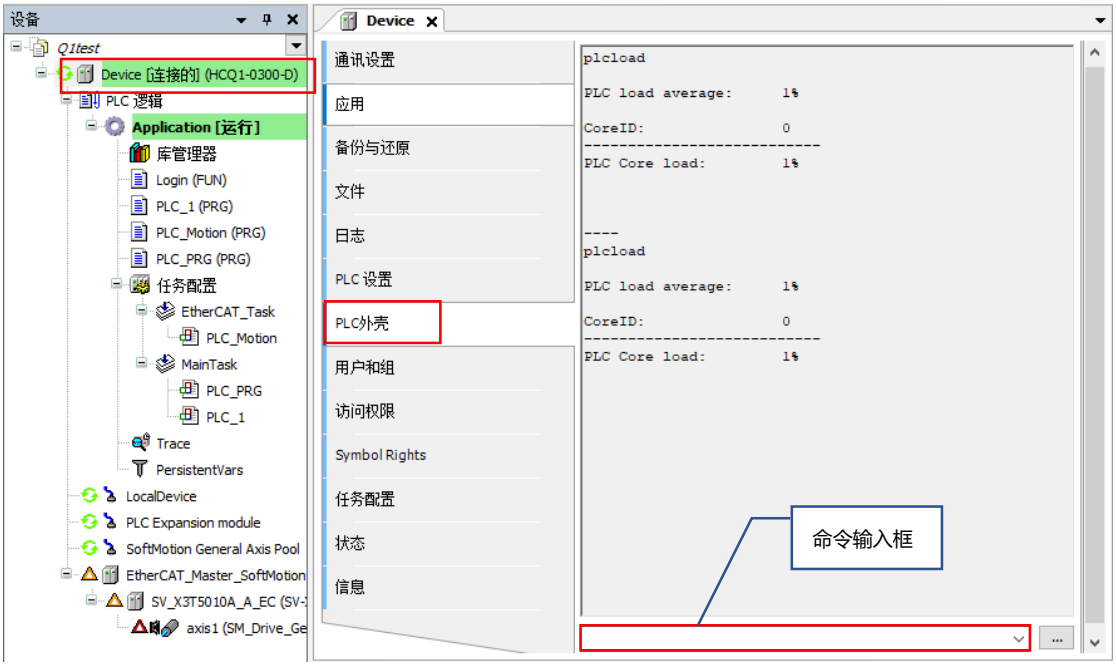
图表 5.7-5



5.8 PLC 脚本功能

PLC 脚本是一个基于文本的控制监视（终端）。从控制器中得到的具有特定信息的命令以一个输入行进行输入并且作为一个字符串发送到控制器，返回相应的字符串在脚本浏览窗口中显示，这个功能用于诊断和调试目的，界面显示如下：

图表 5.8-1



PLC 脚本具体的命令见下表，鼠标双击“Device”后在，右侧的设备界面中找到“PLC 外壳”，在 PLC 外壳的编辑界面正下方提供了命令输入框，用户只需要在命令输入框中输入相应的指令，Q1 就给返回指令对应的反馈信息。输入“？”，则会返回显示所有 Q1 支持的命令。

图表 5.8-2

```
----
?

?
       Prints list of available commands.
getcmdlist
       Used internally to display all available commands.
mem <address> [<size>]
       Print Hexdump of spezified memory region.
reflect
       Just Reply the command which was entered (for testing the connection).
applist
       Print List of currently loaded applications.
pid [<app name>|<app index>]
       Dump Guids of one specific or all loaded applications.
pinf [<app name>|<app index>]
       Dump Project informations of one specific or all loaded applications.
startprg [<app name>|<app index>]
       Start one specific or all loaded applications.
stopprg [<app name>|<app index>]
       Stop one specific or all loaded applications.
resetprg [<app name>|<app index>]
       Reset one specific or all loaded applications.
resetprgcold [<app name>|<app index>]
       Perform a cold reset one specific or all loaded applications.
reload [<app name>|<app index>]
       Reload one specific or all loaded applications from their bootprojects.
getprgprop
       [not implemented, yet]
getprgstat [<app name>|<app index>]
       Get the status of one specific or all loaded applications.
plcload
       Get the Processor load of the PLC tasks.
rtsinfo
       Print Runtime System Informations, like Processor and Runtime Version.
channelinfo
       Return communication channel information.
rtc-get
       Get UTC via DateTime string.
rtc-set
       Set UTC via DateTime string (see ISO 8601).
       Required format: "rtc-set YYYY-MM-DDThh:mm:ss[,sss]"
saveretains [<applicationname>]
       Save retains to files(s). [Optional only from specified application].
restoreretains [<applicationname>]
       Restore retains from file(s). [Optional only for specified application].
```

将这些命令翻译并以表格形式整理如下：

图表 5.8-3

命令	描述
?	显示所有可使用的脚本信息
getcmdlist	显示所有可获得的内部使用命令
mem <address> [<size>]	显示十六进制特定内存范围
reflect	返回上次输入的指令（主要用于连接测试）
applist	显示当前加载的应用列表
pid [<app name>|<app index>]	一个特定的跳转或者所有加载的应用
pinf [<app name>|<app index>]	一个特定的跳转工程信息或者所有加载的应用
startprg [<app name>|<app index>]	启动一个特定的或者所有的应用程序
stopprg [<app name>|<app index>]	停止一个特定的或者所有的应用程序
resetprg [<app name>|<app index>]	复位一个特定的或者所有的应用程序
resetprgcold [<app name>|<app index>]	针对特定的或者加载的应用执行冷复位
reload [<app name>|<app index>]	重新加载一个特定的或者从启动工程中加载的应用
getprgprop	目前暂未实现
getprgstat [<app name>|<app index>]	获取一个特定的或者所有的应用程序的状态
plcload	获取当前 PLC 任务的负载率
rtsinfo	显示实时核运行系统的信息
channelinfo	返回通讯通道信息
rtc-get	从时间字符串中获取 UTC（协调世界时）
rtc-set	通过时间字符串设定 UTC（需要按照以下格式设定"`rtc-set YYYY-MM-DDThh:mm:ss[,sss]`")
saveretains [<applicationname>]	将 Retain 掉电保持数据保存到文件中（特定应用程序）
restoreretains [<applicationname>]	将 Retain 掉电保持数据重新存储到文件中(特定应用程序)

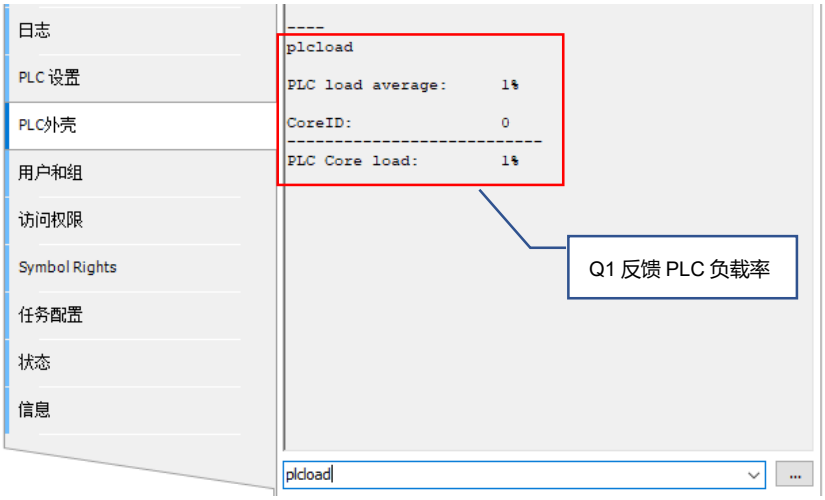
注意：

如果需要使用脚本功能，首先需要用户登录 PLC 才可以使用相应指令

【例2】 使用 PLC 脚本功能查看 PLC 的运行负载率

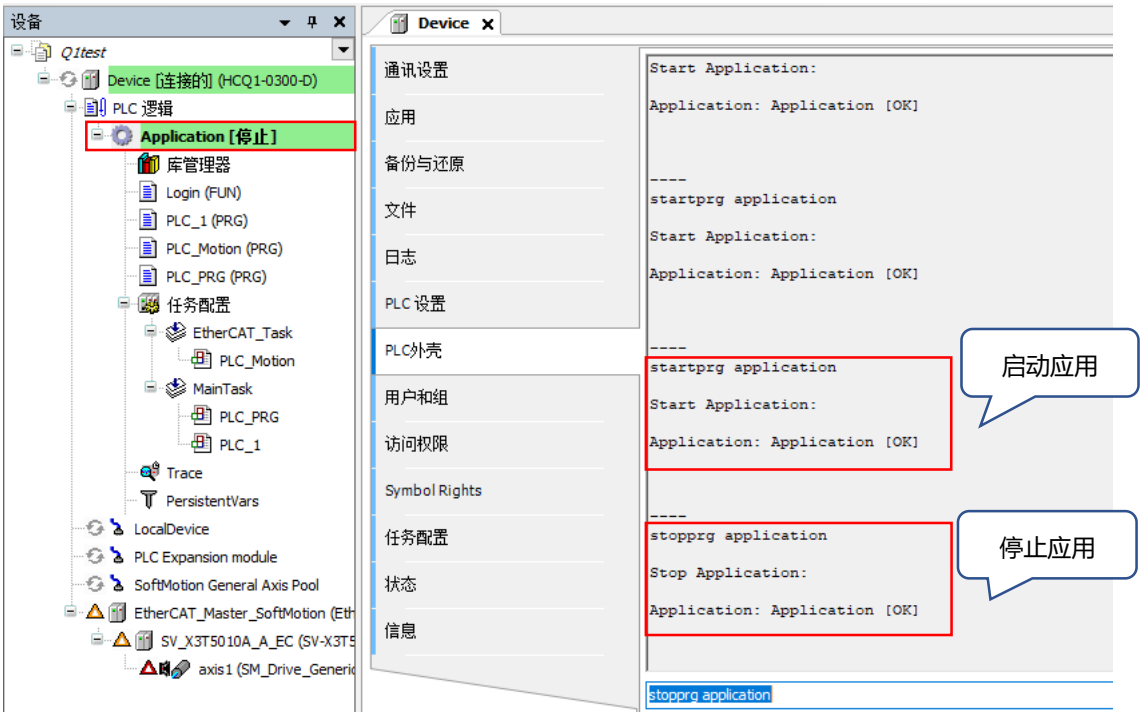
和 Q1 建立通讯后，在左侧树形菜单“Device”下找到“PLC 外壳”，在“PLC 外壳”右侧提供的界面下方命令输入框内，输入“plcload”

图表 5.8-4



【例 3】通过 PLC 脚本启动/停止应用中的程序。

输入“？”后根据提示得知实现以上功能需要使用 startprg/stopprg 命令，按照指令格式输入“命令+应用程序的名字”： startprg application（大小写不做区分）



5.9 程序隐含检查功能

用户在编写程序的过程中，可能会发生如下几种情况：

- 除法运算过程中除数为零
- 指针在赋值的过程中指向空地址
- 调用数组的过程中，出现数组的上下标越界

针对以上情况，CODESYS 默认提供了用于隐含检查的 FUN，用户可以添加这些 FUN 检查已编辑程序，“用于隐含检查的 POU”属性中提供了以下几种函数功能：

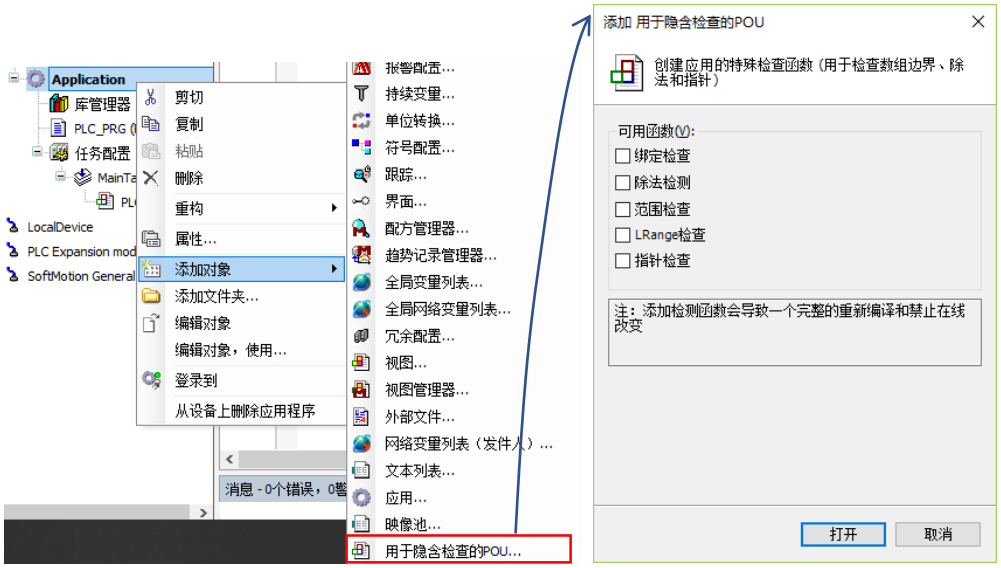
- 绑定检查（CheckBounds）
- 除法检测（CheckDivDInt/ CheckDivLInt/ CheckDivReal/ CheckDivLreal）
- 范围检查（CheckRangeSigned/ CheckRangeUnsigned）
- LRange 检查（CheckLRangeSigned/ CheckLRangeUnsigned）
- 指针检查（CheckPointer）

注意：

用于隐含检查的函数添加后，会在程序在线运行时，自动在后台运行并根据所提供的功能对程序进行检查，检查过程会占用 CPU 内存资源，所以用户可以在程序编辑调试的时候添加这些检查函数，在程序实际应用于工业现场时删除这些检查函数，以节省 CPU 内存资源。当然如果程序实际运行过程中仍有可能出现数组越界，除零等，那建议仍然添加这些检查函数，以保证程序正常运行。

在 POU 中添加一个检查之后，会按照选定的编程语言打开，系统默认的编程环境是 ST 结构化文本。用户可以通过右键单击“Application”在弹出的快捷菜单中选择“添加对象”，找到“用于隐含检查的 POU...”随后系统会弹出 “添加用于检查的 POU” 对话框，下面对这这几种常用函数进行介绍。

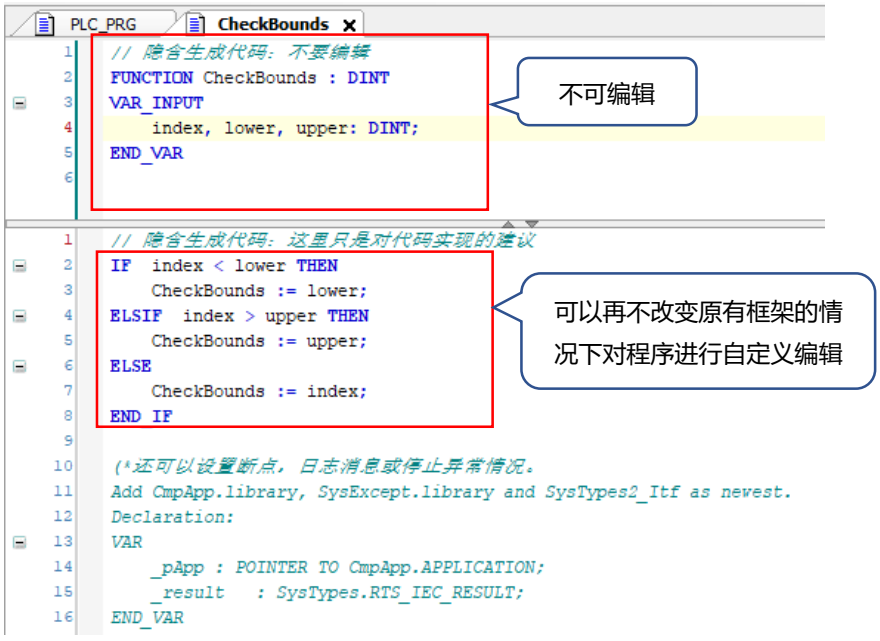
图表 5.9-1



CheckBounds（数组边界检查函数）

通过此函数可以检查数组的边界是否超出，通常应用在存在可变数组类型的应用中，使用该检查函数可以有效的保证数组不出现边界溢出的情况。添加数组边界检查函数后，可以在程序编辑区看到该函数的源代码如下：

图表 5.9-2



注意：

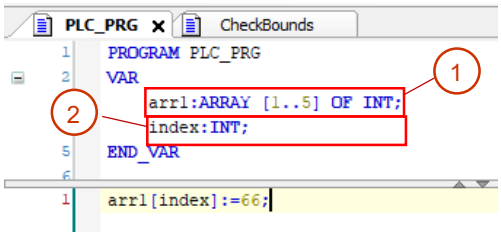
所有检查函数的变量声明都不可编辑，如果用户进行变量的删除和添加等操作，程序编辑将出现错误，但是检查函数的程序区是可编辑的，用户可以在不改变原有程序架构的情况下对程序进行自定义编辑。

下面通过一个例子介绍数组边界检查函数的运行效果。

【例3】 添加 CheckBounds 功能，对可变数组类型进行边界溢出检测。

首先在程序中添加可变数组，声明如下：

图表 5.9-3

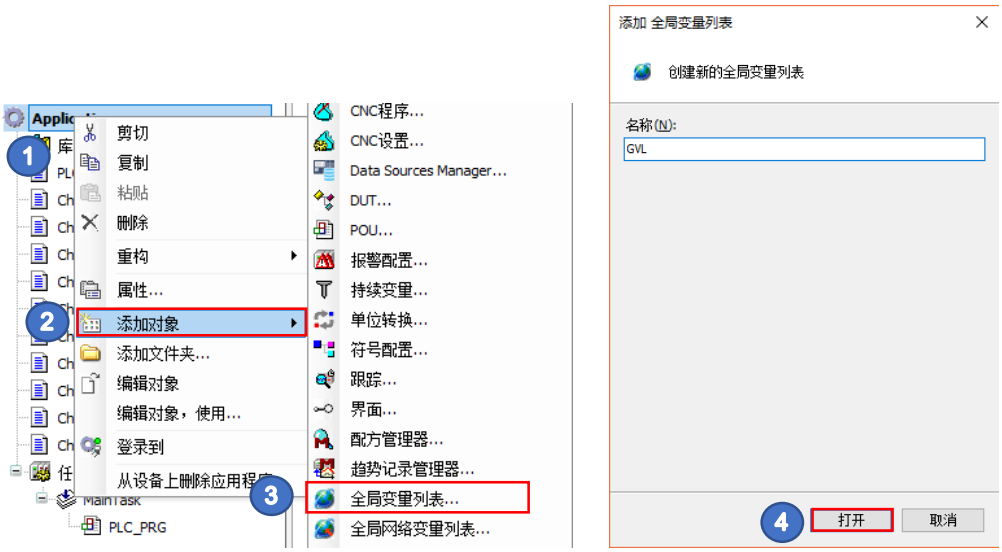


在程序中，数组 arr1 只有 1~5 这 5 个元素，index 为 INT 整型变量，在没有给定初始值的情况下，默认数值为 0，那程序运行后，在没有给定 index 其他数值的情况下，执行结果就是 arr1[0]:=66;但是原始数组中并没有定义 arr1[0]这个元素，“0” 已经超出了数组的定义范围，运行程序可能会出现程序崩溃等状态。

针对上述示例中可能会产生的情况，可以使用 CheckBounds 函数，添加数组边界检查函数后，程序会自动将索引限制在 “0” - “5” 之间，对于超出上下限的会选择上下限进行写入，例如本例中 arr1[index] 最终会被限制为 arr1[1]：=66；，而非 arr1[0]：=66；

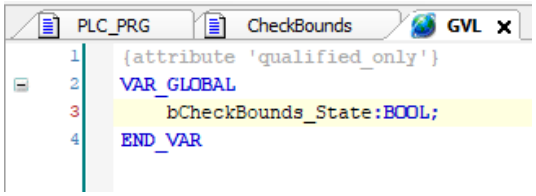
为了方便在线后查看数组是否出现越界的情况，在全局变量中添加”bCheckBounds_State”变量用于显示当前数组越界状态，首先在 “Application” 右击选择 “添加对象”，找到其下 “全局变量列表” 在弹出对话框 “添加全局变量列表” 中给定名称后，单击打开进行创建：

图表 5.9-4



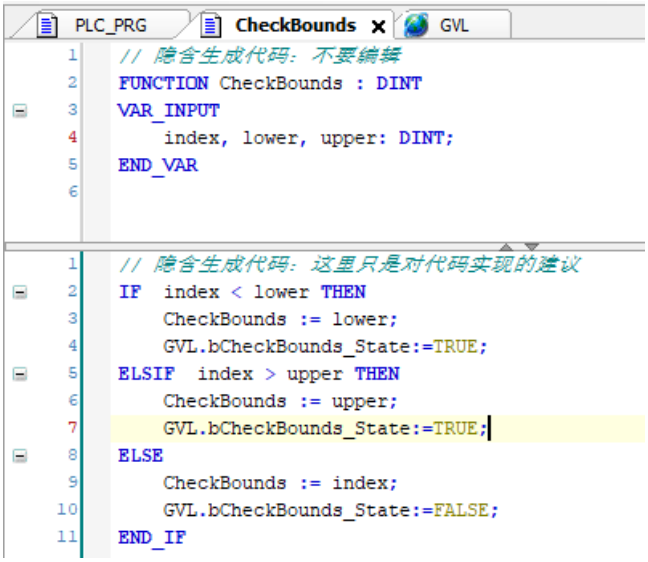
新建全局变量列表后，在变量声明区中添加”bCheckBounds_State”变量用于显示当前数组越界状态：

图表 5.9-5



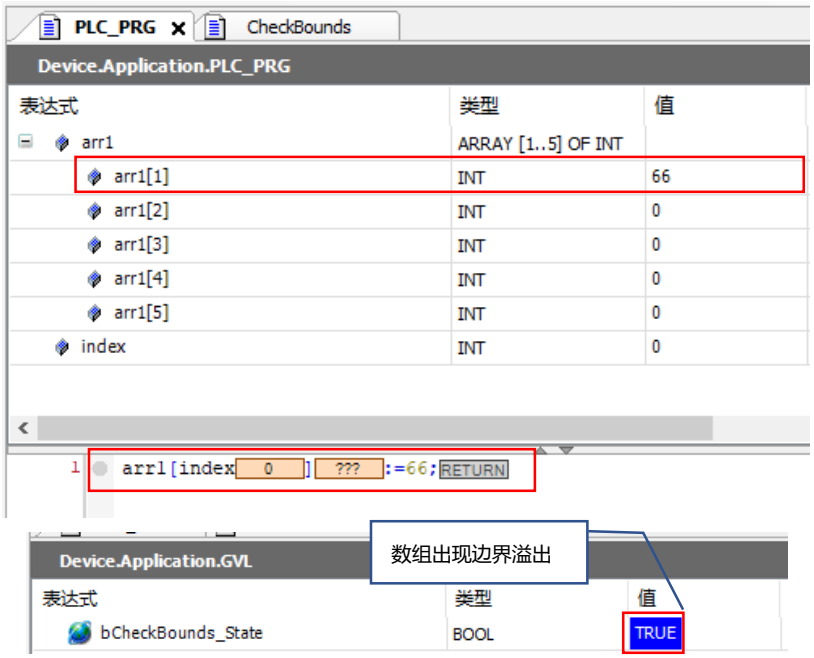
在数组边界溢出检查函数中添加该状态变量到程序中如下：

图表 5.9-6



运行程序并查看运行结果：

图表 5.9-7

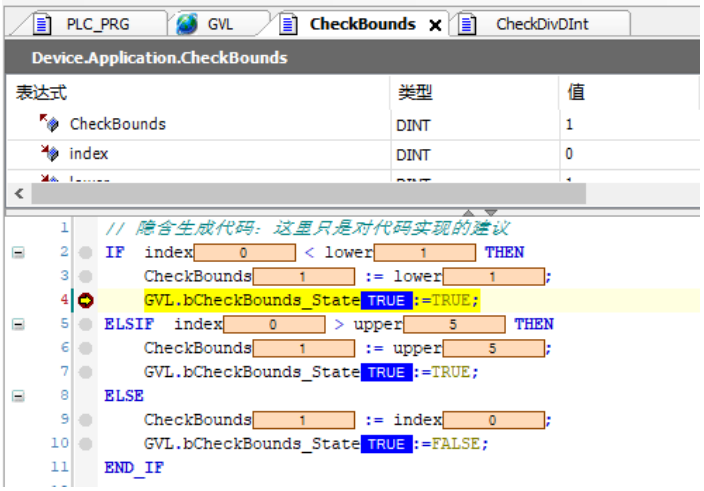


由于检查函数属于 FUN，无法在在线状态下直接查看变量的数值，用户也可以添加断点以查看变量的在线值，选中需要添加断电的行，右击在弹出的快捷菜单中选择“新断点”或者直接通过键盘上的“F9”添加并激活断点：

图表 5.9-8

成功添加之后即可查看函数内的变量在线数值：

图表 5.9-9



CheckDiv[Data Type]（除零检查函数）

为了避免

CheckRangeSigned/ CheckRangeUnsigned（边界检查函数）

CheckPointer（指针检查函数）

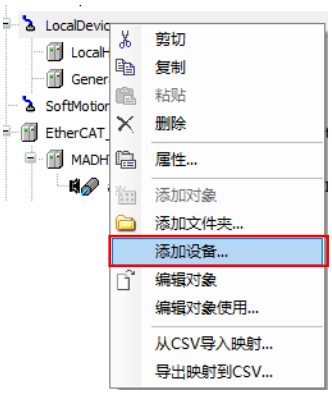
5.10Q1 自带高速 IO 调试说明

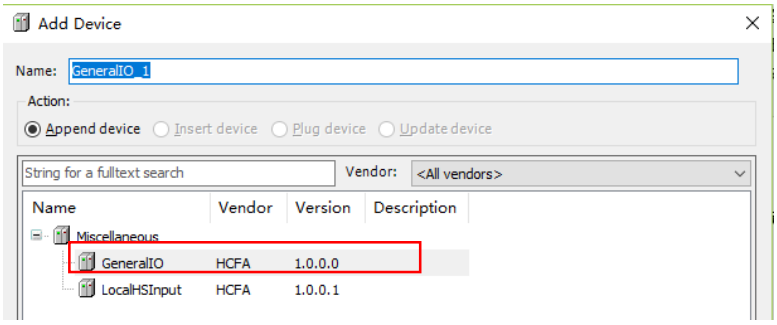
Q1 本机自带高速输入输出，高速输入输出的端口同样可以作为普通输入输出使用。

5.10.1 Q1 本地 IO 作为普通输入输出使用说明

Q1 本地 IO 作为普通输入输出的接线可以直接参考数字量输入输出模块，同样支持 PNP 和 NPN。首先需要在设备管理器中安装对应的描述文件，详细过程可以参考 3.2，安装完成后，确保 Q1 与 CODESYS 正常通讯，在 LocalDevice→添加设备

图表 5.10-1

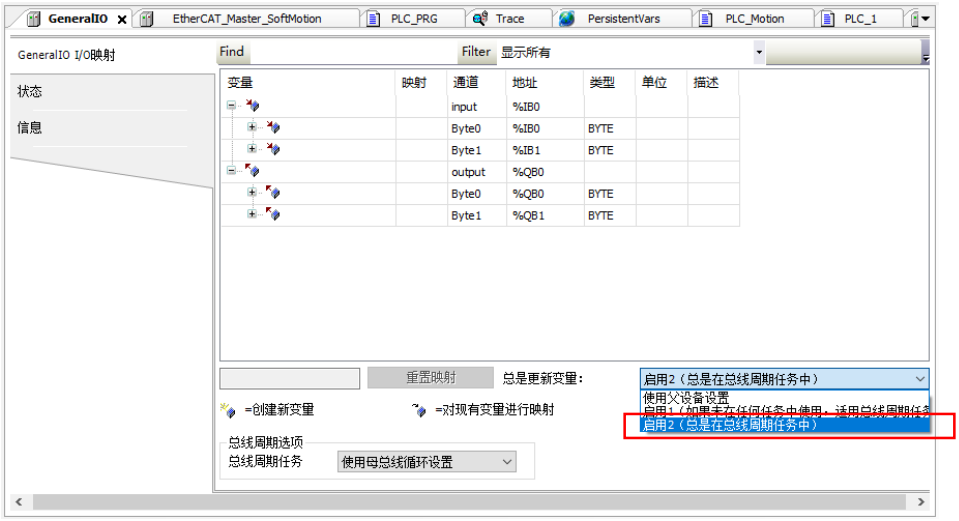




该模式下的本地 IO 的接线方式可以直接参考数字量输入模块的接线方式，添加完成后在右侧 General I/O 映射下就可以监控当前本地 IO 的状态或进行测试性输出

需要注意的时，在无程序测试的情况下需要将 IO 刷新周期选择为：总是在总线周期任务中，否则未映射变量的 IO 不会刷新，映射了程序变量则当前 IO 会随程序变量刷新时间刷新

图表 5.10-2

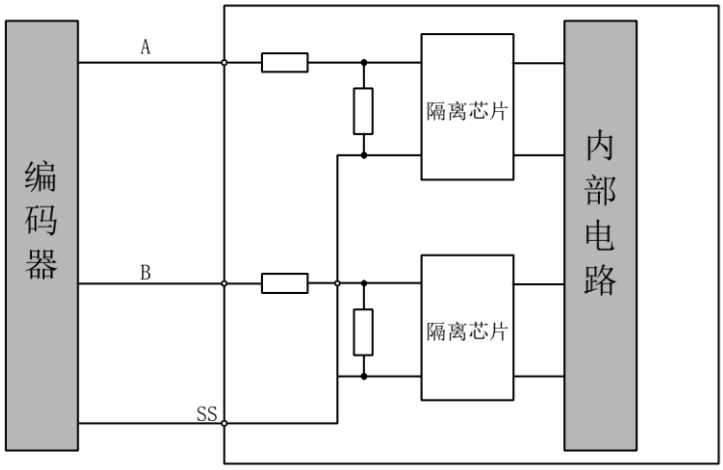


5.10.2 Q1 本地 IO 作为高速输入输出使用说明

Q1 本地 IO 作为高速输入端口时，接线方式如下：

1）高速输入接线（以通道 1 作为示例）

图表 5.10-3



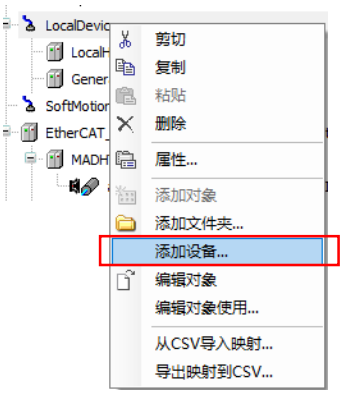
通道	端口	端口	通道
hsi_cnt	X0	X8	hsi_cnt4
	X1	X9	
hsi_cnt1	X2	X10	hsi_cnt5
	X3	X11	
hsi_cnt2	X4	X12	hsi_cnt6
	X5	X13	
hsi_cnt3	X6	X14	hsi_cnt7
	X7	X15	

高速输入支持 PNP 及 NPN，接线方式，取决于编码器的输入类型，如果是 PNP 类型的输入信号则公共端接 24V，如果是 NPN 类型的则接 0V，公共端内部导通。

2）高速输出接线（暂无高速输出功能，后续完善）

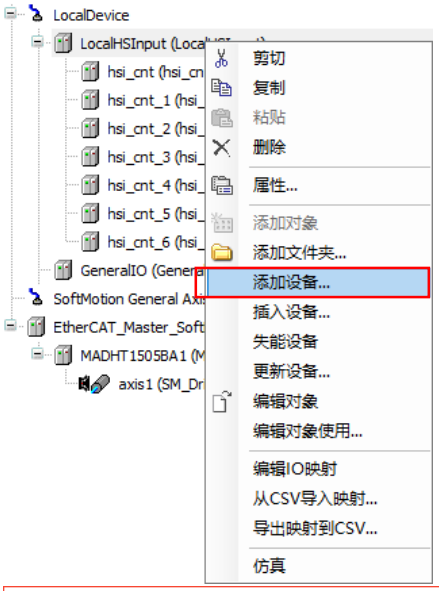
同时可以添加 General IO 使用其中包含的普通输出，但是对于输入（输出）来说只能选择一种输入（输出）方式。使用高速输入接口时，需要安装文件夹中包含的设备描述文件：HIS_Counter_Drv 中的文件和 LocalHSInput.xml，安装步骤参考 3.2，安装完成后，在 LocalDevice→添加设备→LocalHSInput 添加高速 IO

图表 5.10-4

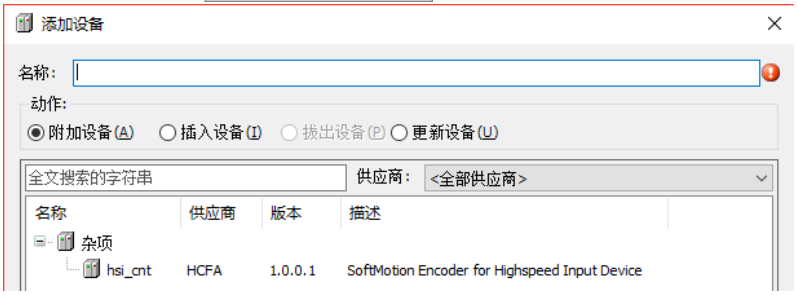


添加完成后在 LocalHSInput→插入设备添加高速输入通道，最多可以添加 8 个通道，对应关系参考下表：

图表 5.10-5



通道	端口	端口	通道
hsi_cnt	X0	X8	hsi_cnt4
	X1	X9	
hsi_cnt1	X2	X10	hsi_cnt5
	X3	X11	
hsi_cnt2	X4	X12	hsi_cnt6
	X5	X13	
hsi_cnt3	X6	X14	hsi_cnt7
	X7	X15	



禾川为高速 IO 添加了对应的库文件，在使用前首先需要安装 hcfalib.compiled-library，安装完成后在该库文件中可以找到名称为 hsi_ref 的功能块，在主程序中添加该功能块并填写对应的端口即可使用高速输入。功能块简要说明如下：

图表 5.10-6

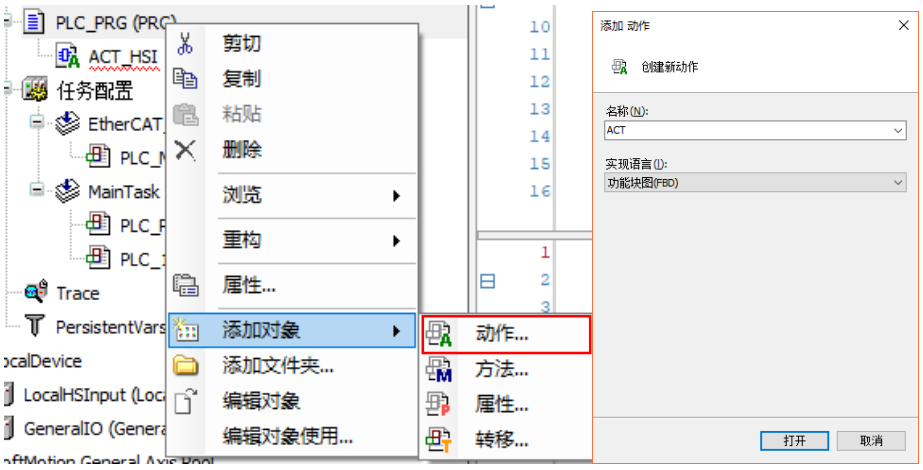
FUNCTION_BLOCK hsi_ref

InOut:

Scope	Name	Type	Comment
Input	wDriveID	WORD	
	bCounterEnable	BOOL	
	bEventEnable	BOOL	
	bDspdEnable	BOOL	
	bLatchEnable	BOOL	
	wCountMode	WORD	0: 2-phase-x1; 1: 2-phase-x2; 2: 2-phase-x4; 3: 1-phase-2-input; 4: 1-phase-1-input hardware mode; 5: 1-phase-1-input software mode
	bSoftDirection	BOOL	
	bSoftPreTrg	BOOL	Current Count Value update triger
	diEventCmpValue	DINT	Event Task compare value
	wHardTrgMethod	WORD	port_Z and port Latch EdgeCheck method;
	wCmpoutCtrlword	WORD	enable Compare output
	wMeasureUnitTime	WORD	
	wStatus_clr	WORD	
	diSoftPreValue	DINT	Current Count Value update value
	diCntMinValue	DINT	
	diCntMaxValue	DINT	
	bTabCmpEnable	BOOL	
	wStartNum	WORD	
	wEndNum	WORD	
	BZport_sel	BYTE	
	Blatch_sel	BYTE	
Output	dwDspdFreqValue	DWORD	
	diCurCountValue	DINT	
	diLatchData	DINT	
	wCounterStatu	WORD	direction,overflow,underflow and so on

在主程序右击选择添加对象下的动作（Action），在弹出的对话框中选择功能块图（FBD）作为编程语言

图表 5.10-7



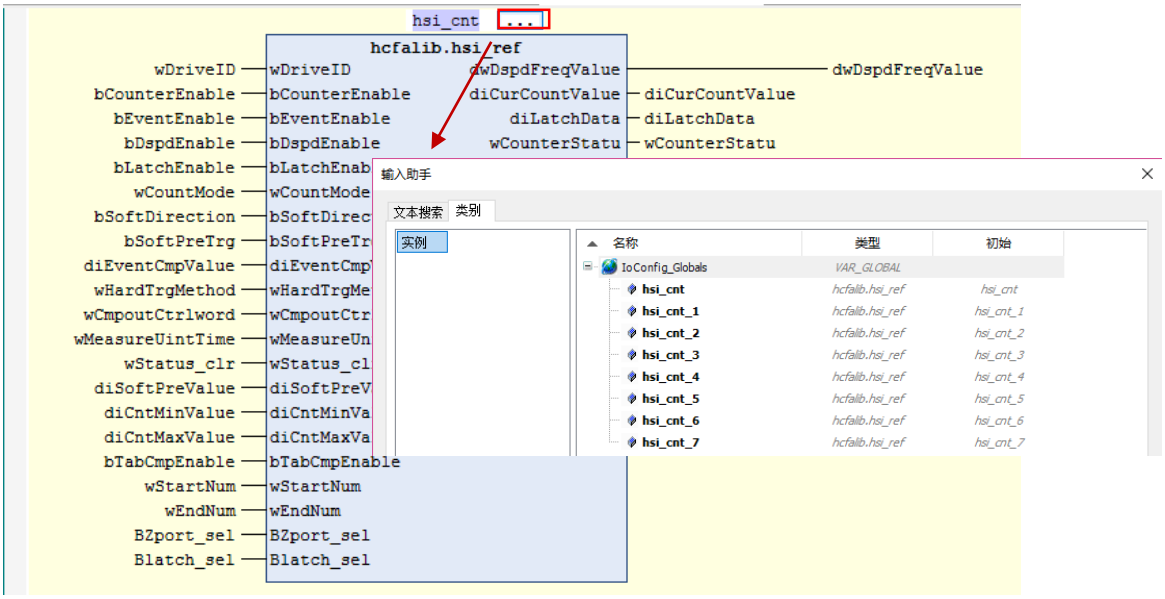
在主程序的变量声明窗口中定义功能块中需要使用的变量如下：

图表 5.10-8

```
//高速输入功能块端口定义
wDriveID        :WORD;
bCounterEnable  :BOOL;
bEventEnable    :BOOL;
bDspdEnable     :BOOL;
bLatchEnable    :BOOL;
wCountMode      :WORD:=0;
bSoftDirection  :BOOL;
bSoftPreTrg     :BOOL;
diEventCmpValue:DINT;
wHardTrgMethod  :WORD;
wCmpoutCtrlword:WORD;
wMeasureUintTime:WORD;
wStatus_clr     :WORD;
diSoftPreValue  :DINT;
diCntMinValue   :DINT:=0;
diCntMaxValue   :DINT:=10000;
bTabCmpEnable   :BOOL;
wStartNum       :WORD;
wEndNum         :WORD;
BZport_sel      :BYTE;
Blatch_sel      :BYTE;
dwDspdFreqValue :DWORD;
diCurCountValue :DINT;
diLatchData     :DINT;
wCounterStatu   :WORD;
```

在新添加的动作（Action）中找到全局变量中的 his_cnt,对需要使用的通道进行配置

图表 5.10-9



配置完成后给定对应的参数即可使计数器工作在不同的模式下，对于不同模式的描述及测试结果请参考附件 HCQ1-输入测试.xlsx

高速输入暂为内测功能，测试结果如有不同或者测试过程中有什么问题可以直接联系坐着作者，正式版本请等待后续正式发布，高速输出暂无功能块和测试报告。

5.11拓展的添加

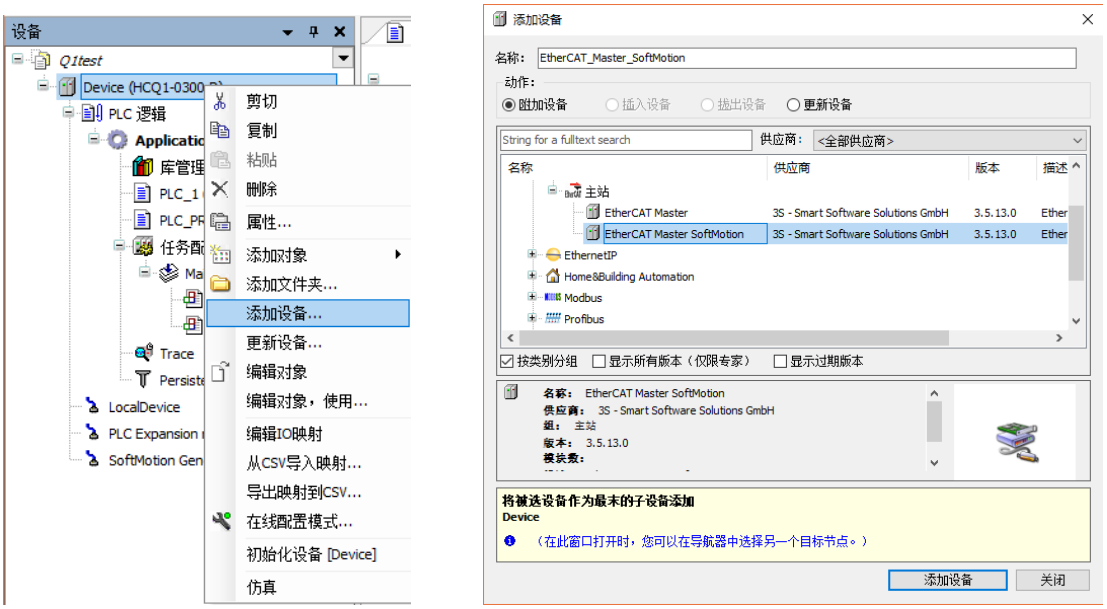
Q1 V301 版本支持在右侧扩展本地 IO，最大功率为 16W，下表为 Q1 支持的扩展模块的功率（数据都是按照上浮 10%-15%计算，选型时候不要把功率卡的太死，金属片传输总线电流信号会随着时间产生一定电阻，从而消耗一部分功率）：

图表 5.11-1

序号	模块型号	参考功率
1	LocalEtherCATDevice	16W
2	HCQX-EC-D	1.344W
3	HCQX-ID16-D	0.78W
4	HCQX-0D16-D	1.32W
5	HCQX-MD-D	1.032W
6	HCQX-AD04-D	1.044W
7	HCQX-DA04-D	1.056W

右击树形菜单 Device→添加设备，在弹出对话框添加设备中选择 EtherCAT Master SoftMotion，点击添加设备

图表 5.11-2

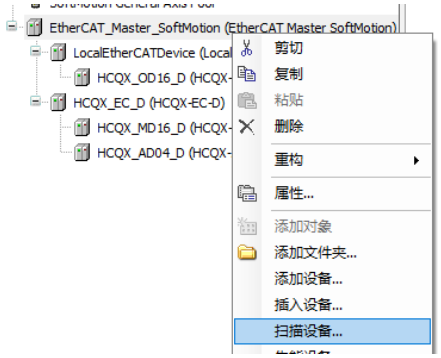


添加完成后，选择正确的网卡并将其下载到 Q1 中（登录 Q1），建议用户选择按名称选择网络，以便程序在更多设备上配套（不同设备网口 Mac 地址不同）

图表 5.11-3

在登录状态下，右击 EtherCAT_Master_SoftMotion 选择扫描设备，建议通过扫描设备来进行设备的添加，如果条件不允许，可以选择添加设备，将所需要的设备添加到设备树中

图表 5.11-4



在扫描设备对话框中如果出现未知的设备，则需要用户检查所使用的描述文件是否匹配，正确扫描到的模块显示如下：

图表 5.11-5

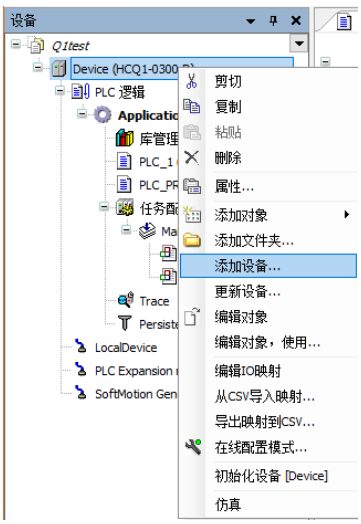
选择，复制所有设备到工程中，就完成了模块的添加，需要特别注意的是，3.0 版本的 Q1 支持本地扩展，手动添加时需要添加 LocalEtherCATDevice 才能添加后续挂载的模块

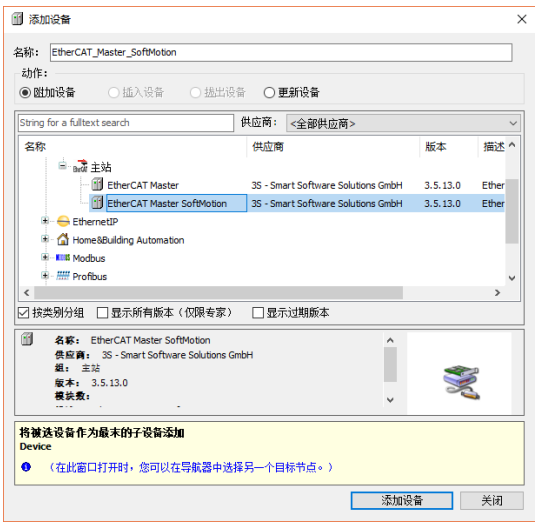
5.12Motion 项目的创建

5.12.1 添加 SoftMotion 项目

右击树形菜单 Device→添加设备，因为驱动器为 X3T，和控制器之间通过 EtherCAT 通讯，所以在弹出对话框添加设备中选择 EtherCAT　Master SoftMotion，点击添加设备

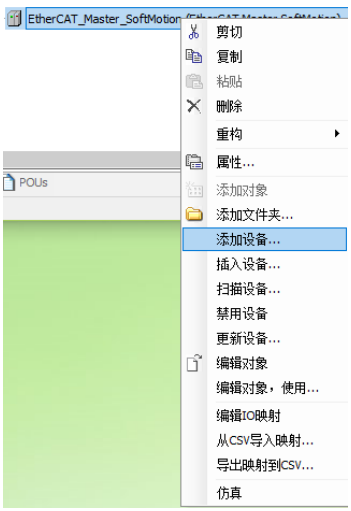
图表 5.12-1

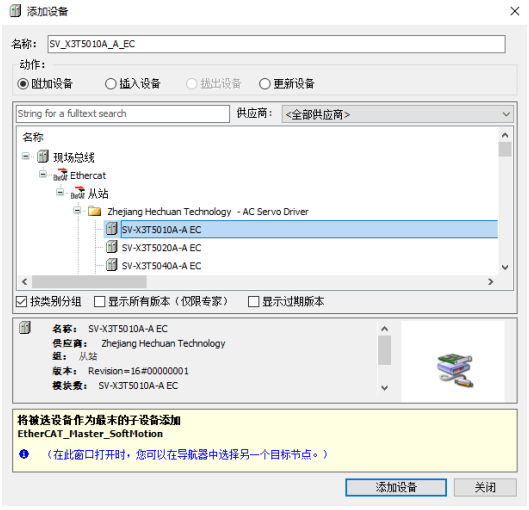




在 EtherCAT Master SoftMotion 右击选择添加设备，此时需要添加测试所需驱动器，找到从站下的禾川驱动器，测试使用 SV-X3T5010A-A-EC，选中后点击添加设备

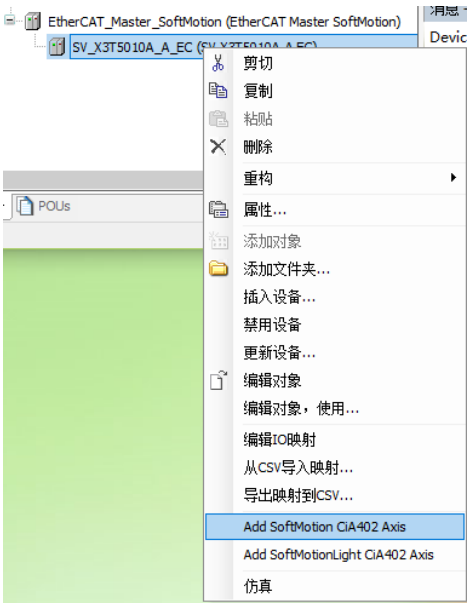
图表 5.12-1

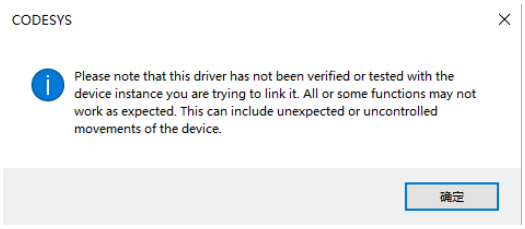




在添加好的从站 SV-X3T5010A-A-EC 右击，选择 Add SoftMotion CiA402 Axis，系统弹出提示框，点击确定

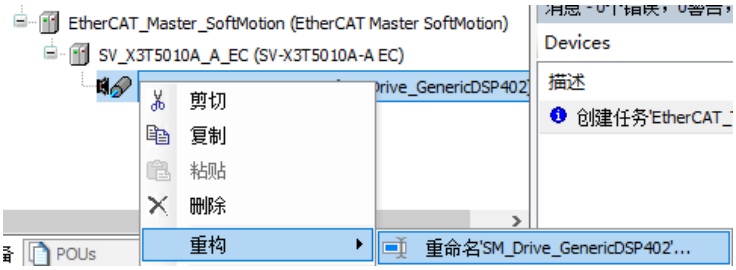
图表 5.12-2





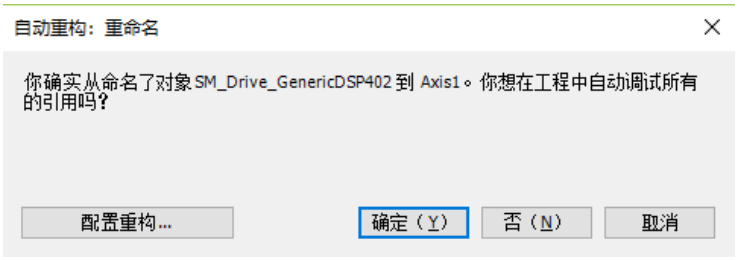
右击添加好的 SM_Drive_GenericDSP402，选择重构→重命名 ‘SM_Drive_GenericDSP402’，修改成 Axis1 以便后续在 PLC 程序中调用相关轴变量

图表 5.12-3



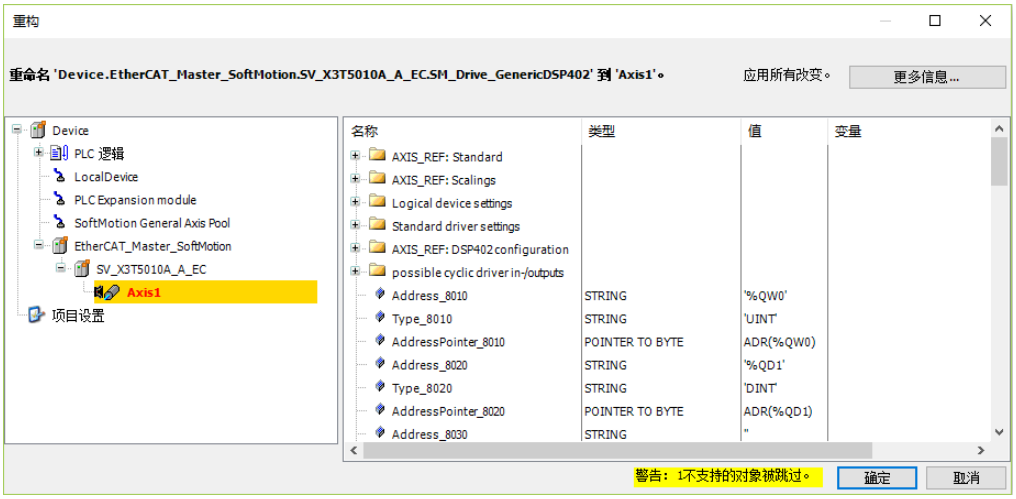
重命名后弹出对话框，选择“确定”

图表 5.12-4



系统会自动完成轴变量重命名后的映射，点击“确定”完成轴的重命名

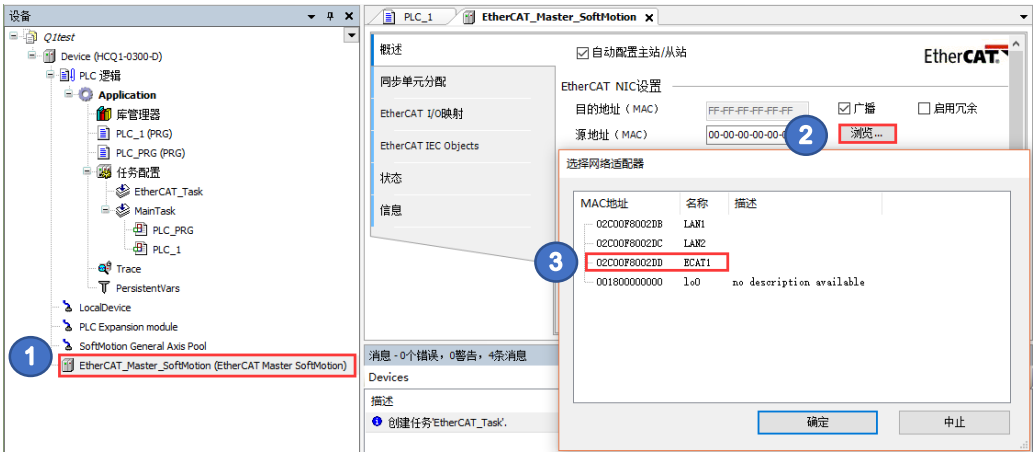
图表 5.12-5



5.13修改 EtherCAT 主站信息和通讯参数

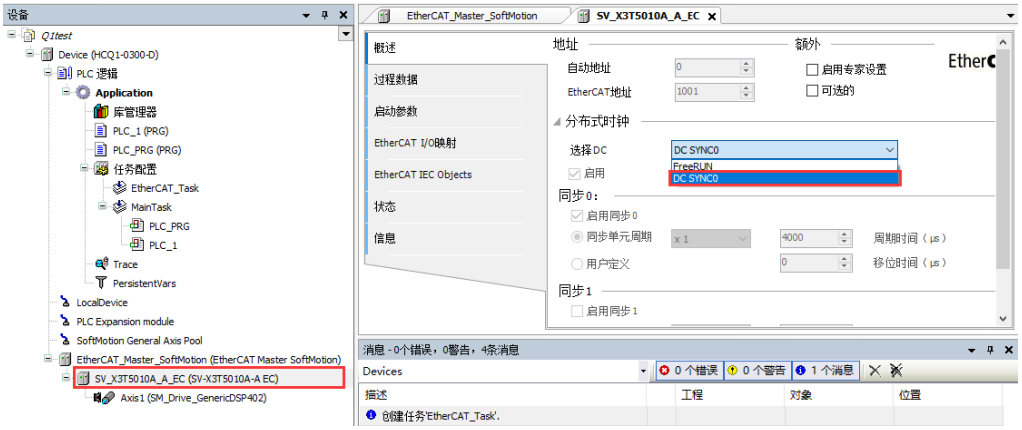
双击添加出来的 EtherCAT 主站，首先需要选择进行通讯的网络适配器，找到概述页面中的源地址右侧浏览，选择名称一栏显示为 ECAT1 的网卡，点击确定

图表 5.13-1



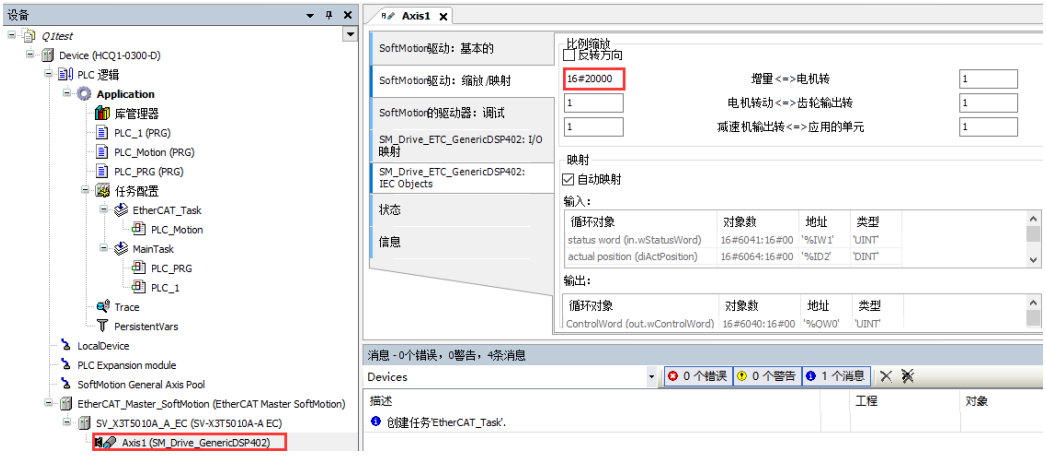
双击 EtherCTA 从站 SV_X3T5010A_A_EC，在右侧概述中展开分布式时钟→选择 DC→DC SYNC0，选择 DC 同步模式

图表 5.13-2



修改轴 Axis1 中关于编码器参数的设置，X3T 为 17 位编码器，修改增量为 16#20000

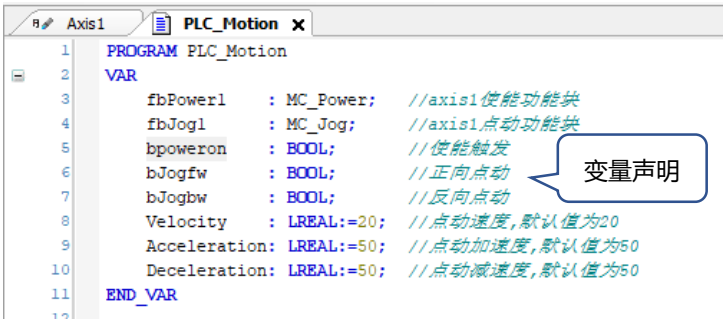
图表 5.13-3



5.14单轴运动控制指令的实现

新建 POU 并将其命名为 PLC_Motion 用以编写单轴运动控制指令，具体步骤参考 4.3，运动控制指令编写如下：

图表 5.14-1

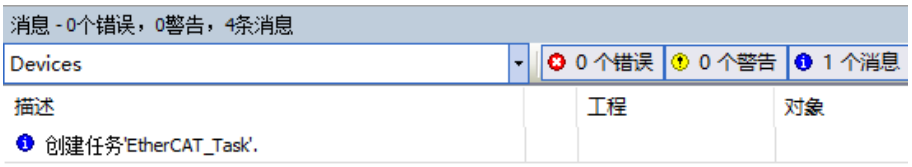


```
fbPower1(
    Axis:=axis1 ,
    Enable:=TRUE ,
    bRegulatorOn:=bpoweron ,
    bDriveStart:=TRUE , );
fbJog1(
    Axis:=axis1 ,
    JogForward:=bJogfw ,
    JogBackward:=bJogbw ,
    Velocity:=Velocity ,
    Acceleration:=Acceleration ,
    Deceleration:=Deceleration ,
    Jerk:= , );
```

主程序

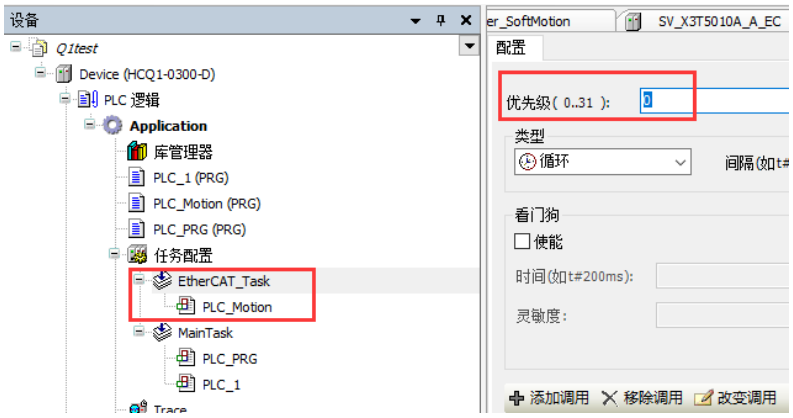
在 CODESYS 中添加 Motion 项目会自动在在任务配置下生成一个 EtherCAT_Task

图表 5.14-2



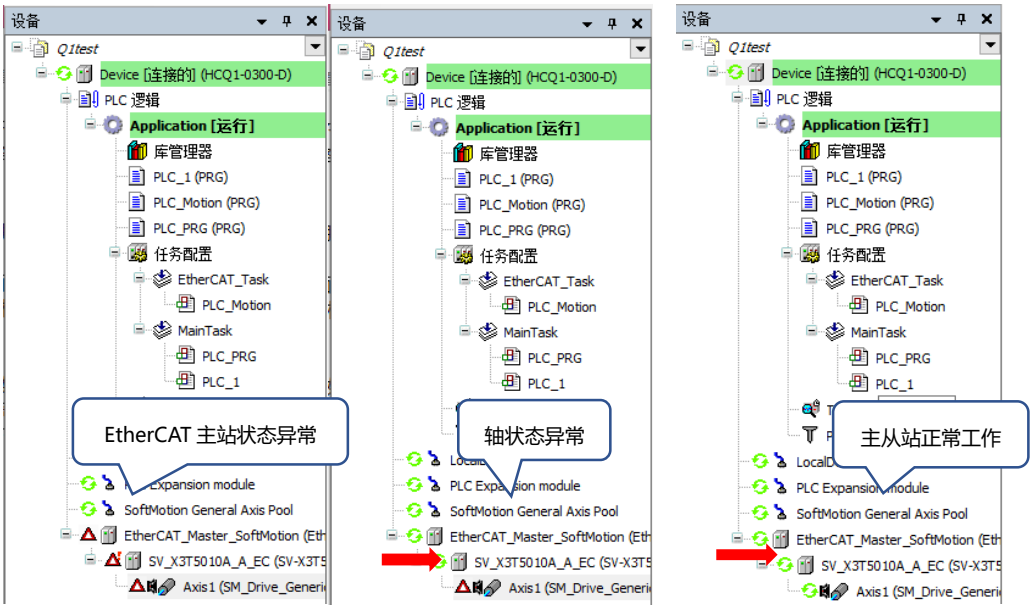
用户需要手动配置 EtherCAT_Task 中的相关参数，否则在后续轴运行过程中会出现同步周期丢失的报错，双击 Application→任务配置→EtherCAT_Task，在该任务中调用 PLC_Motion,具体步骤参考 4.2，并设置优先级为 “0”

图表 5.14-3



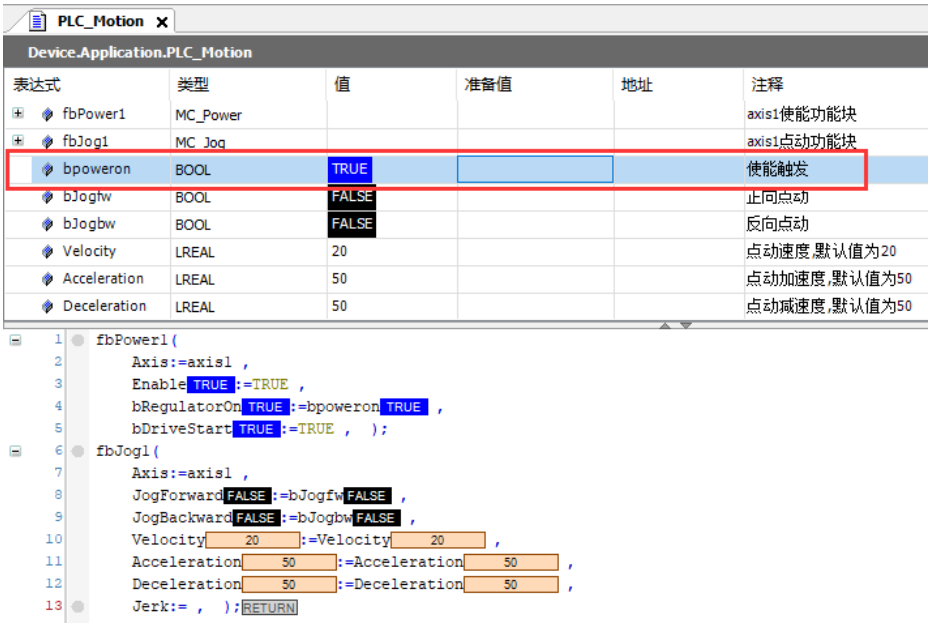
完成硬件配置的修改之后，登录并运行程序，等待驱动器和轴状态正常后，双击 PLC_Motion

图表 5.14-4



首先对轴使能，将 bPoweron 置为 TRUE

图表 5.14-5



完成对 Axis1 的使能之后，通过给 bJogfw 置 TRUE 对轴正向点动，通过给 bJogbw 置 TRUE 对轴反向点动，其他轴功能块的编辑和使用可以直接参考 CODESYS 在线帮助：https://help.CODESYS.com/

用我们的工作　　创造美好的生活

浙江禾川科技股份有限公司
电话：0570-7117888
地址：浙江省龙游县工业园阜财路9号

杭州研发中心
技术支持热线:400126969
地址：杭州市余杭区五常街道文一西路1001号D幢4楼