



MagicWorks PLC 编程软件

用户手册

权利声明

版权声明

深圳市合信自动化技术有限公司版权所有，并保留对本手册及本声明的最终解释权和修改权。未得到深圳市合信自动化技术有限公司的书面许可，任何单位及个人不得以任何方式或形式对本手册内的任何部分进行复制、摘录、备份、修改、传播、汇编、翻译成其它语言、将其全部或部分用于商业用途。

商标声明



TrustPLC®、MagicWorks®、COTRUST®均为深圳市合信自动化技术有限公司或其关联公司的注册商标，受法律保护，侵权必究。未经深圳市合信自动化技术有限公司或其关联公司的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、永久下载、修改、传播、抄录或其他产品捆绑销售，或注册为其域名或无线网址名称，或域名或无线网址的主要部分。

免责声明

本手册依据现有信息制作，其内容如有更改，恕不另行通知。深圳市合信自动化技术有限公司在编写该手册的时候已尽最大努力保证其内容准确可靠，但深圳市合信自动化技术有限公司不对本手册中的遗漏、不准确或印刷错误导致的损失和损害承担责任。

前言

本手册完整概述了如何使用 MagicWorks PLC 进行编程，并在安装和调试该软件时它可以为您提供支持。

适用对象

本手册适用于使用 MagicWorks PLC 执行控制任务的人员。

在线帮助

集成于软件中的在线帮助是对本手册的补充，在线帮助是在使用该软件时提供详细的支持。

服务支持

除本手册外，客户还可通过深圳市合信自动化技术有限公司网站 <http://www.co-trust.com> 或官方微信及时了解最新产品动态，以及下载需要的技术文档。

版本修订记录

发行日期	修订后版本	修订内容
2020 年 9 月	V1.63	MagicWorks PLC 编程软件 V2.20 新增追踪功能，手册修订内容如下： 1. 增加追踪视图功能描述，参见章节 10.3.4 追踪视图。 2. 新增特殊存储区 SMB550~SMB2047 的诊断功能说明，参见章节 D 特殊存储器（SM）标志位（中型 PLC 专用）
2020 年 5 月	V1.62	MagicWorks PLC 编程软件 V2.19 新增 C 语言编程功能支持沿功能、新增 H56 CPU 及 H56 硬件组态功能、新增编码器轴配置功能，手册修订内容如下： 1. 增加编码器轴配置功能描述，参见章节 7.4.4.1 轴配置 2. 完善 C 程序块描述 5.3.5 C 程序块（CTH300 系列 PLC 适用） 3. 可编程 PLC 新增 H56-006 型号，完善章节 1.3 可编程 PLC 型号、7.4 硬件组态 4. 完善 FC/FB 支持的 IN/OUT/IN_OUT 的总数，参见章节 5.3.2 FB 功能块（CTH3 系列 PLC 适用）
2020 年 1 月	V1.61	MagicWorks PLC 编程软件 V2.18 新增 C 语言编程、200 系列 CPU 支持硬件组态等新功能，手册详细修订内容如下： 1. 增加 C 语言编程功能，参见章节 5.3.5 C 函数（CTH300 系列 PLC 适用） 2. 中型 PLC 增加三款新型号：CTH3 H31-000S1、CTH3 H32-000S1 和 CTH3 H36-00AS2 3. 小型 PLC 增加三款新型号：运控 CPU M228IL、CPU M228ML 和 CPU M228SL，SM 区放大至 1K 4. 增加 200 系列硬件组态功能，只支持 CPU 与运控模块 EM253 的配置，参见章节 7.4.6 200 系列 PLC 硬件组态 5. 支持轴与电子凸轮配置功能，参见章节 7.4.4 轴配置向导、7.4.5 电子凸轮向导 6. 增加新功能：编译时检查重复输出，选项对话框可设置是否检查，默认不检查，参见章节 5.7 编译功能
2019 年 4 月	V1.60	MagicWorks PLC 编程软件 V2.17 新增轴配置、电子凸轮配置等新功能，手册详细修订内容如下： 1. 硬件组态 300 站点新增轴配置向导和电子凸轮向导，参见章节 8.6.4 轴配置与电子凸轮配置 2. 新增插入至对比趋势图功能描述，参见章节 10.3 如何使用状态表监控 3. 新增结束状态程序监控，查看状态程序监控，参见章节 10.2 如何使用程序状态监控 4. 程序编辑器页面增加 POU 管理功能，参见章节 4.4 程序编辑器窗口 5. 新增重新编译程序块功能，编译功能改为只编译改动过的程序块，参见章节 5.8 编译功能

		6.交叉引用表中新增调用结构说明，参见章节 5.4 交叉引用表 7.系统块配置中通信端口说明新增 TCP/IP 端口参数配置界面，参见章节 5.6.1 通信端口 8.导入 EDS 文件的步骤和截图由 8.6.4 挪至 8.6.2, 参见章节 8.6.2 组态 CANopen 通信 9.新增中型 PLC 扩展模块诊断说明，参见章节 11.1.3 模块诊断事件 10.PLC 支持类型新增 H226M, H226XM, H226XC, 参见章节 1.3 可编程 PLC 型号 11.新增重新编译程序块功能，在线更新固件新增支持 CTH200 CPU, 参见章节 4.1.1 菜单栏
2019 年 4 月	V1.50	MagicWorks PLC 编程软件 V2.15 新增 PID/PTO/HSC 向导、socket 库等新功能，手册详细修订内容如下： 1.增加 demo 功能，首次安装时，弹出打开 demo 的提示，参见章节 2.2 安装步骤 2.导出硬件组态符号表，参见章节 5.2 符号表 3.新增 PID 指令向导, PTO 指令向导, HSC 向导，参见章节 5.5.7 PID 指令向导、5.5.8 PTO 指令向导、5.5.9 HSC 向导 4.支持通过 CP6453 网关编程及 Mico 远程编程，参见章节 6.3.1 设置通信接口 5.新增 CP6453-RS485 网关模块配置说明，参见章节 6.3.2 设置通信连接 6.增加 socket 库，参见章节 8.4 库的使用 7.硬件组态中新增 EtherCAT 从站 IO 组态功能及支持 H36-001S2, 新增如何添加第三方 EtherCAT 从站 IO 组态，参见章节 8.6.3 组态 EtherCAT 通信

目录

权利声明	II
前言	III
版本修订记录	IV
1 软件介绍	1
本章概要	1
1.1 软件特点	1
1.2 基本操作流程	1
1.3 可编程 PLC 型号	4
2 软件安装	5
本章概要	5
2.1 安装环境	5
2.2 安装步骤	5
2.3 卸载软件	8
3 入门指南	9
本章概要	9
3.1 创建新工程	9
3.2 编辑程序	11
3.3 建立通信和编译程序	17
3.4 下载和运行程序	18
3.5 在线帮助的使用	19
4 软件组成	21
本章概要	21
4.1 项目管理器窗口	22
4.1.1 菜单栏	22
4.1.2 工具栏	25
4.1.3 项目树	26
4.1.4 输出窗口	27
4.1.5 状态条	28
4.2 符号表编辑器窗口	28
4.3 初始化 V 内存数据块编辑器窗口	28
4.4 程序编辑器窗口	29
4.5 交叉引用查看器窗口	29
4.6 状态表编辑器窗口	30
4.7 EasyCAN 组态窗口	30
4.8 硬件组态窗口	31
5 PLC 编程系统	32

本章概要.....	32
5.1 工程结构.....	32
5.2 符号表.....	32
5.3 程序块.....	36
5.3.1 OB 组织块	37
5.3.2 FB 功能块（CTH3 系列 PLC 适用）	38
5.3.3 FC 功能	44
5.3.4 DB 数据块（CTH3 系列 PLC 适用）	45
5.3.5 C 程序块（CTH300 系列 PLC 适用）	51
5.3.6 SFC.....	55
5.3.7 局部变量表	55
5.3.8 程序块的导入导出	57
5.4 交叉引用表	59
5.5 向导的使用	63
5.5.1 NETR/W-PPI 指令向导	63
5.5.2 NETR/W-UDP 指令向导.....	68
5.5.3 MODBUS-TCP 指令向导	72
5.5.4 文本显示向导	80
5.5.5 配方向导.....	100
5.5.6 数据记录向导	103
5.5.7 PID 指令向导.....	107
5.5.8 PTO 指令向导	112
5.5.9 HSC 向导	119
5.6 库的使用	122
5.7 编译功能.....	125
6 通信设置	128
本章概要.....	128
6.1 通过串口与 PLC 建立 PPI 通信	128
6.2 通过 CP6451 网关与串口 PLC 建立通信.....	130
6.3 通过 CP6453 网关与串口 PLC 建立通信.....	133
6.4 通过 CTH200/300 集成以太网口与 PLC 建立通信	140
6.5 通过 iNet 与远程 CTH200/300 建立通信	141
6.6 通过 Mico 与远程 CTH200/300 建立通信	142
7 PLC 配置	146
本章概要.....	146
7.1 系统块配置	146
7.1.1 通信端口.....	146
7.1.2 断电数据保持	147
7.1.3 密码保护.....	148
7.1.4 数字量输出表	150
7.1.5 模拟量输出表	150

7.1.6	数字量输入滤波	150
7.1.7	模拟量输入滤波	151
7.1.8	脉冲捕捉位	152
7.1.9	背景时间	153
7.1.10	EM 配置	153
7.1.11	LED 配置	153
7.1.12	运行模式编程	153
7.2	数据块配置	153
7.3	EasyCAN 组态	157
7.3.1	EasyCAN 组态软件特点	157
7.3.2	组态并通信	158
7.3.3	EasyCAN 组态文件的导入导出	162
7.4	硬件组态	163
7.4.1	一般操作步骤	164
7.4.2	组态 CANopen 通信	170
7.4.3	组态 EtherCAT 通信	179
7.4.4	轴配置向导	186
7.4.5	电子凸轮向导	197
7.4.6	200 系列 PLC 硬件组态	201
7.4.7	其它操作	202
7.5	安全保护	205
8	PLC 存储区	209
	本章概要	209
8.1	内存类型	209
8.2	CPU 内存地址范围	210
8.3	寻址内存	211
8.3.1	直接寻址	211
8.3.2	符号寻址	213
8.3.3	间接寻址	213
8.4	常数数值范围	214
8.5	关键字	216
9	PLC 在线控制	218
	本章概要	218
9.1	PLC 运行/停止模式	218
9.2	PLC 上电复位	219
9.3	上载/下载/清除	219
9.3.1	上载程序	221
9.3.2	下载程序	222
9.3.3	清除 PLC 存储区	223
9.4	PLC 信息	223
9.5	存储卡配置	224

9.6	存储卡擦除	225
9.7	从 RAM 建立数据块	225
9.8	实时时钟	226
9.9	将 PLC 与当前项目中的块进行比较	227
9.10	运行模式编程	227
9.11	PLC 类型选择	229
10	调试与监控程序	231
	本章概要	231
10.1	如何模拟过程条件	231
10.2	如何使用程序状态监控	233
10.3	如何使用状态表监控	239
	10.3.1 状态表监控	239
	10.3.2 使用趋势视图	240
	10.3.3 使用对比趋势视图	242
	10.3.4 追踪视图	245
11	诊断	249
	本章概要	249
11.1	诊断信息	249
	11.1.1 PLC 非致命错误	249
	11.1.2 PLC 致命错误	251
	11.1.3 模块诊断事件 (CTH300 系列 PLC 适用)	252
	11.1.4 运行模式转换	254
	11.1.5 同步错误	255
	11.1.6 SOE 事件诊断	255
11.2	时间戳不匹配	255
12	打印	257
	本章概要	257
12.1	支持打印的项目	257
12.2	基本操作	257
	附录	261
A	术语解释	261
B	指令集	262
	B.1 指令一览表	262
	B.2 指令详解	267
C	特殊存储器 (SM) 标志位 (小型 PLC 专用)	360
	C.1 SMB0 至 SMB29 (只读特殊内存)	360
	C.2 SMB30 至 SMB549 (读 / 写特殊内存)	364
D	特殊存储器 (SM) 标志位 (中型 PLC 专用)	372
E	中断事件优先级列表	376
F	国际助记符	386

1 软件介绍

MagicWorks PLC 是 COTRUST 中小型 PLC 的编程调试平台,用户可使用该软件编写调试 PLC 的控制程序,实现各种自动化控制功能。

本章概要

- 软件特点
- 基本操作流程
- 可编程 PLC 型号

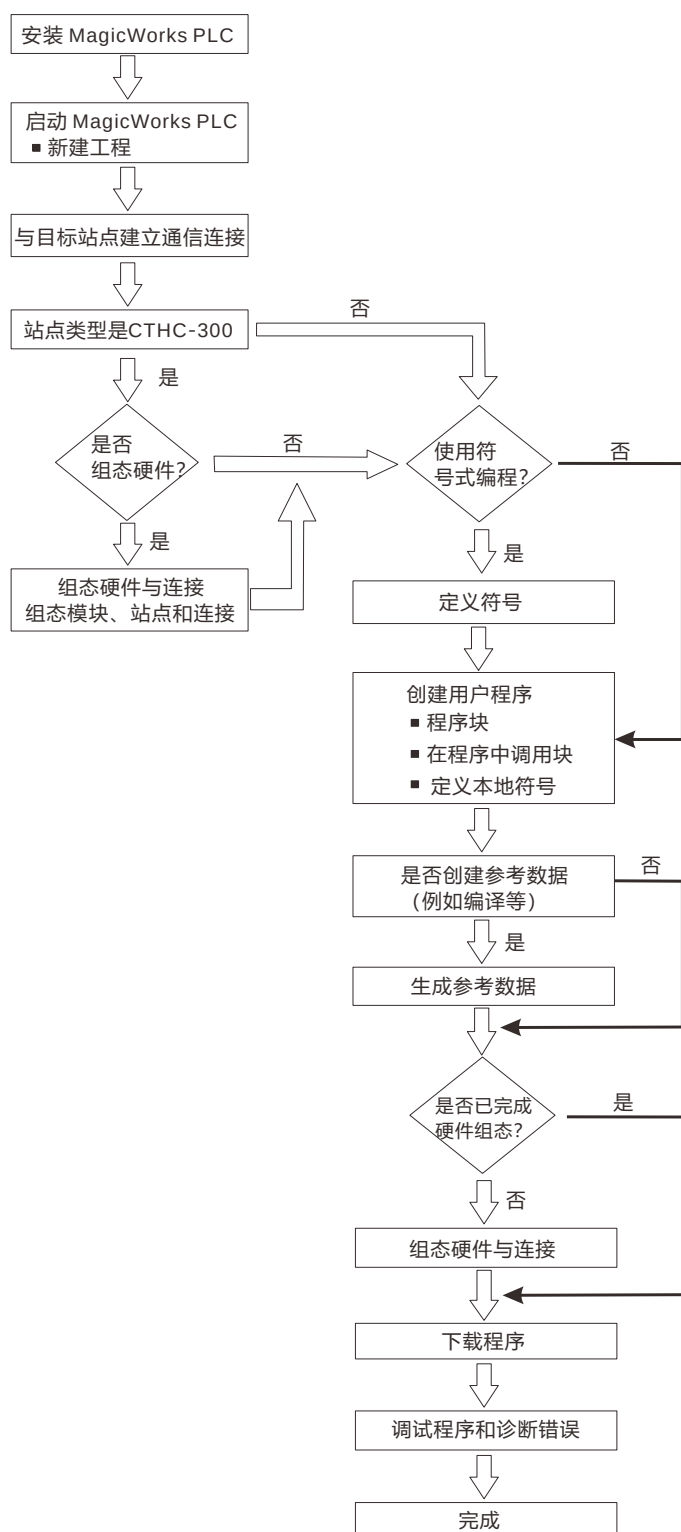
1.1 软件特点

MagicWorks PLC 软件具有如下特点:

- 采用多文档管理风格;
- 支持中小型 PLC 编程,支持中型 PLC 硬件组态;
- 支持 STL (指令表) 和 LAD (梯形图) 两种标准语言;
- 丰富的指令集,内置 IEC61131-3 定义的标准功能以及一些特殊的应用指令;
- 支持中断服务程序;
- 支持用户自定义功能(子程序);
- 允许在程序中选择显示绝对地址或者是其符号变量名称,便于工程的实施、维护;
- 完善的联机功能,包括下载、上载、清除、在线监控、强制、读写实时时钟等;
- 定义了完善的快捷键、右键菜单,方便用户的使用;
- 对用户的错误操作尽可能地予以屏蔽、提示,体贴用户的操作;
- 支持中英文切换;
- 硬件组态中增加新模块或升级模块类型无需升级软件版本,通过“硬件更新”实现支持新的扩展模块;
- 具有自动检测软件更新自动下载升级的功能;
- 允许用户自定义库,以实现程序的复用;

1.2 基本操作流程

当您使用 **MagicWorks PLC** 创建一个自动化解决方案时，您可以参考以下操作流程图，完成一般项目需要执行的步骤。



如上图所示，您可以先组态硬件，然后再对块进行编程；您也可以选择先对块进行编程，再进行硬件组态。

步骤注释

◆ 启动 MagicWorks PLC

通过 Windows 用户接口启动 MagicWorks PLC；

如果所使用的操作系统版本是 Windows 7，请按此方式启动：右键选择 MagicWorks PLC 图标，在弹出的菜单项中选择“以管理员身份运行”。否则有可能部分功能不正常。

◆ 创建项目结构

所有的数据均按照一种体系化的结构存储在项目文件夹中，并随时可供使用。在项目创建完毕后，所有其它的任务均将在该项目中执行；

◆ 组态一个站

组态站时，您可以指定您希望使用的 PLC；

◆ 硬件组态

在对硬件进行组态时，您可在组态表中指定要使用的模块以及用户程序用来对模块进行访问的地址。也可通过使用参数对模块的属性进行设置；

◆ 组态网络和通讯连接

通讯的基础是预先组态的网络。因此，您需要创建自动化网络所需要的子网、设置子网属性、以及设置已联网工作站的网络连接属性和某些通讯连接；

◆ 定义符号

根据作用域的不同，可以将符号分为两类：一类是全局符号，通过符号表定义，在整个站点中生效；另一类是局部符号，通过程序编辑器右上方的局部符号表定义，只在当前 POU 生效。当两种符号重名时，局部符号优先。

◆ 创建程序

使用一种可编程语言（STL 或 LAD）创建一个与模块相链接或与模块无关的程序，并将其存储为块、源文件或图表；

◆ 组态变量

若在 MagicWorks PLC 中创建了变量，就要为其分配所需的属性。使用传送程序，将所创建的变量传送到系统的数据库中；

◆ 将程序下载至可编程控制器

在完成所有的组态、参数分配以及编程任务之后，您可以将整个用户程序或其中的单个块下载至可编程控制器；

◆ 调试程序

为进行调试，您可显示用户程序或 CPU 中的变量值，为变量分配数值，以及为您想要显示或修改的变量创建一个变量表；

◆ 监控操作、诊断硬件

可以通过程序状态在线监控、变量表监控、趋势视图、查看 PLC 诊断信息和硬件组态在线诊断来协助调试。

1.3 可编程 PLC 型号

MagicWorks PLC 支持对 CTSC-100/200 系列和 CTH200/300 系列 PLC 进行编程，用户编程时可配置编程的 PLC 型号如下：

CPU 系列	可编程型号
100 系列	CPU 122、CPU 124、CPU 124XP、CPU126
200 系列	CPU 224I、CPU 224+、CPU 224E、CPU 224XP、CPU 226I、CPU 226M、CPU 226L、CPU 226H、CPU 226M-CAN、CPU 226H-CAN、CPU H224、CPU H224X、CPU H226M、CPU H226L、CPU H226XM、CPU H226XC、CPU H226XL、CPU H228XL、CPU M228IL、CPU M228ML、CPU M228SL
300 系列	H35-006 系列、H36-006 系列、H31-006 系列、H32-006 系列、H56-006

2 软件安装

本章主要介绍 MagicWorks PLC 的安装环境、安装步骤和卸载软件的操作。

本章概要

- ☐ 安装环境
- ☐ 安装步骤
- ☐ 卸载软件

2.1 安装环境

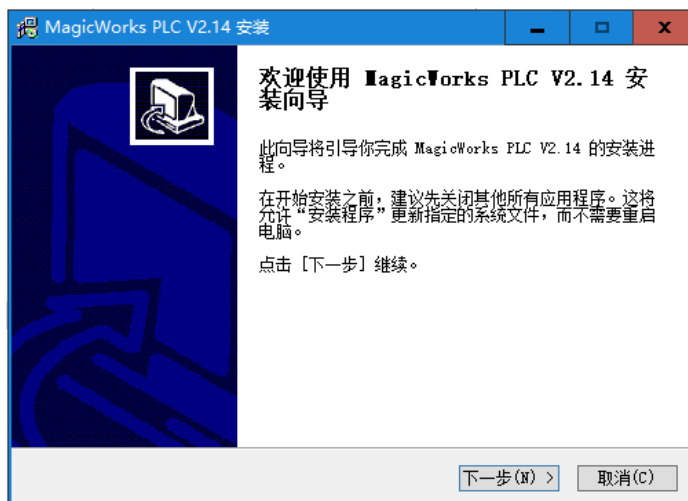
操作系统：Windows 所有操作系统。在 Windows7 及以上系统中安装后，首次使用时需以管理员身份运行。

硬件配置：IBM 486 以上兼容机，内存 256MB 以上，至少 500MB 以上硬盘剩余空间，至少有 1 个 RS232 通讯口或标准网口。

通信电缆：PC/PPI 电缆或标准网线用于计算机与 PLC 的连接。

2.2 安装步骤

双击 MagicWorks PLC 的安装文件，弹出如下对话框：

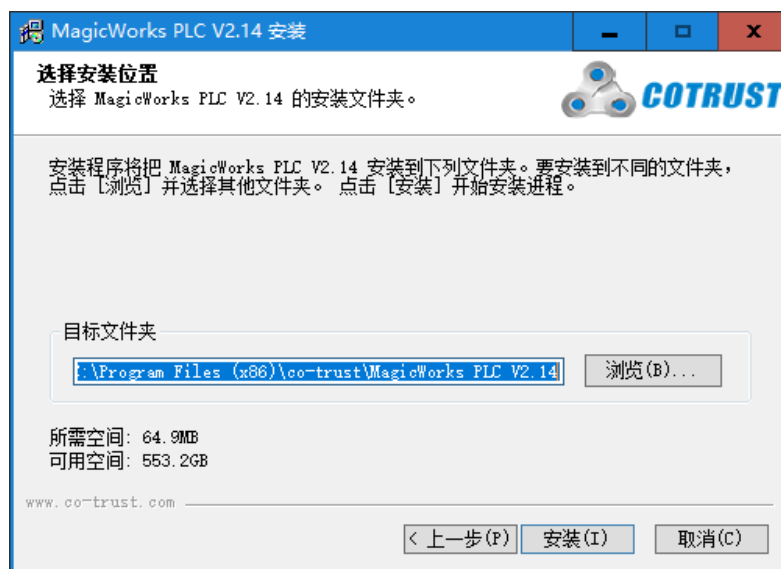
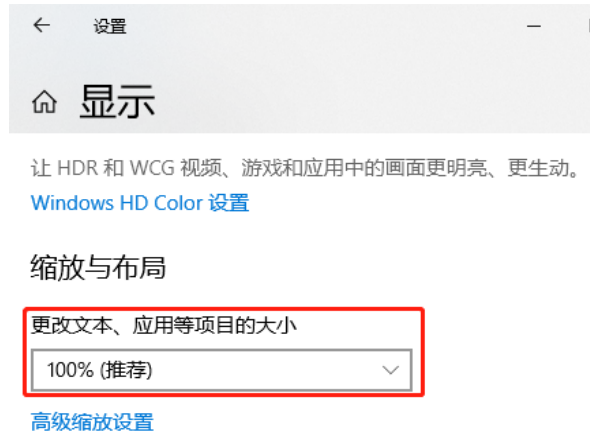


在以上窗口点击“下一步(N)>”按钮即弹出如下对话框：

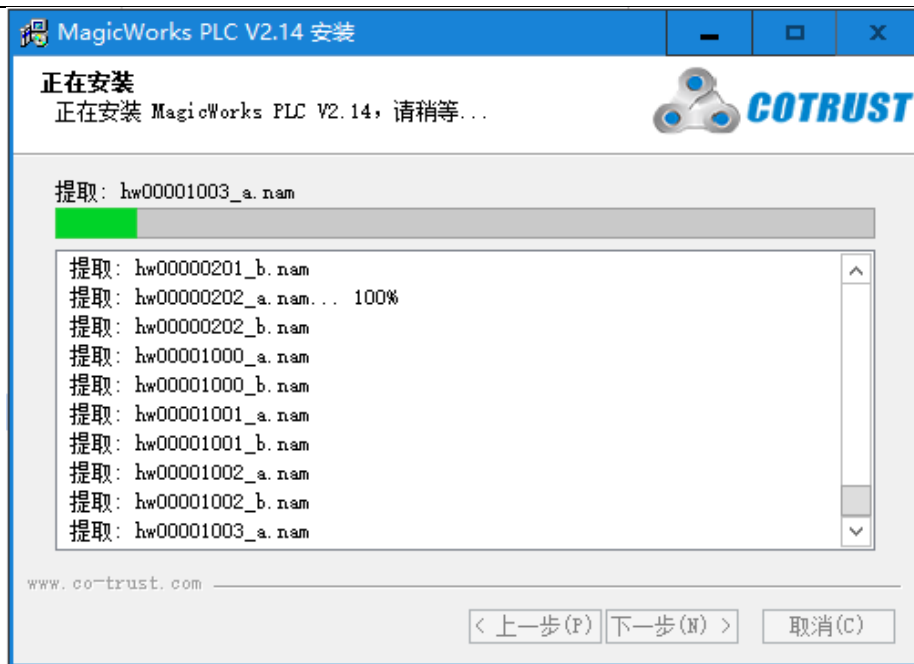


在以上窗口点击“我接受(I)”按钮，出现如下窗口：

提示：若遇到对话框文本显示不全，可在操作系统的显示设置里调小缩放百分比。如下图所示：



以上窗口点击“浏览(B)...”选择文件安装目录，然后点击“安装(I)”进行安装，安装进程如下：



安装完成后，出现如下窗口。

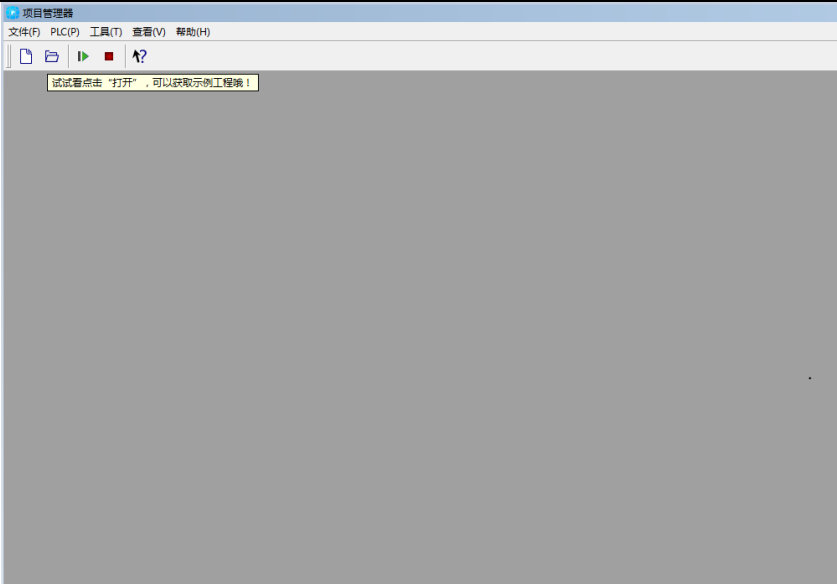
在 Windows 7 或 Windows 10 操作系统下首次启动时，取消勾选“运行 MagicWorks PLC V2.14(R)”并点击“完成(F)”即可完成安装，然后在桌面图标上点击右键选择“以管理员身份运行此程序”，之后就可以通过双击启动程序。

其他操作系统下，请直接勾选“运行 MagicWorks PLC V2.14(R)”并点击“完成(F)”启动程序。



在安装完成界面勾选“运行 MagicWorks PLC V2.14(R)”，并点击“完成(F)”按钮打开软件，进入以下界面，弹出“试试看点击‘打开’，可以获取示例工程哦！”提示框，点击打开按钮，并进入如下路径，即可选择所需示例工程。

示例工程路径：MagicWorks PLC2.14\MagicWorks PLC V2.14\Projects\Demo



打开 Demo 下的文件夹列表，如下图所示

名称	修改日期	类型	大小
em231_7hf_pid	2018-10-18 15:19	文件夹	
em231_pid	2018-10-18 15:19	文件夹	
em231_weight	2018-10-18 15:19	文件夹	
em253_motion_ctrl	2018-10-18 15:19	文件夹	
ext_mem_100K	2018-10-18 15:18	文件夹	
modbus	2018-10-18 15:18	文件夹	
motion_ctrl	2018-10-18 15:19	文件夹	
PID_T	2018-10-18 15:18	文件夹	
save_memory	2018-10-18 15:18	文件夹	
simpleDemo	2018-10-18 15:18	文件夹	
tcp_socket	2018-10-18 15:18	文件夹	

<备注> 关于 MaigcWorks PLC V2.20 版本在 windows7 上安装后运行报错，提示缺少 api-ms-win-crt-runtime-l1-1-0.dll，原因在于系统没有更新，MaigcWorks PLC V2.20 与之不兼容。解决方法为：打开合信官网 www.co-trust.com，依次点击下载中心---MaigcWorks PLC---MaigcWorks PLC V2.20 下载 KB2999226 补丁，解压后安装（根据系统是 64 位还是 32 位安装对应的版本），然后再运行 MaigcWorks PLC 即能成功。

2.3 卸载软件

使用标准 Windows 方法卸载 MagicWorks PLC：

- 1) 在“控制面板”中双击“添加/删除程序”打开 Windows 软件安装对话框；
- 2) 在已安装的显示列表中选择 MagicWorks PLC，并点击“更改/删除”按钮；
- 3) 在弹出的对话框中选择“是”进行卸载，如果不确定该操作，请选择“否”。

3 入门指南

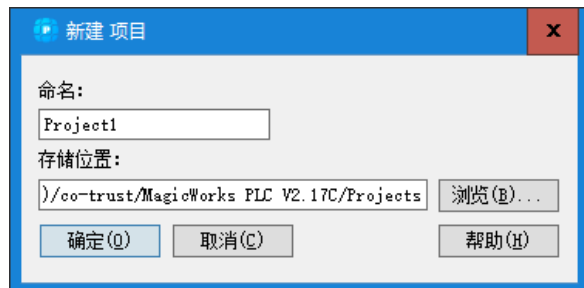
本章概要

- ☐ 创建新工程
- ☐ 系统块属性配置
- ☐ 编辑程序
- ☐ 建立通信和编译程序
- ☐ 下载和运行程序

3.1 创建新工程

创建新项目

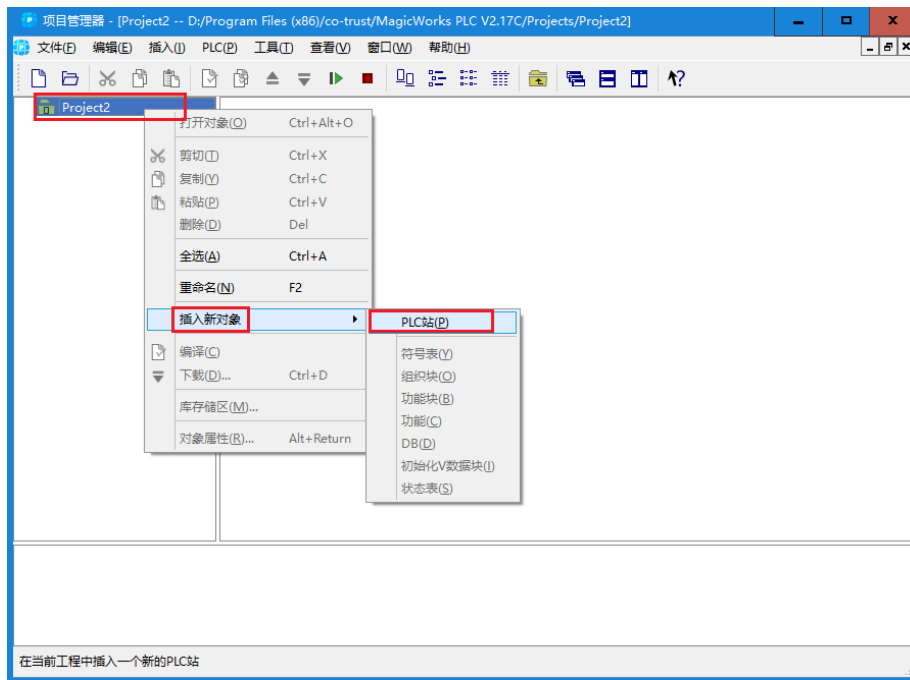
在项目管理器界面选择“文件”→“新建”，或在菜单项直接点击图标“”创建新项目，在弹出的“新建项目”界面可为您的新项目命名（项目名称只能包含字母、数字和下划线）和选择储存位置，设置完成后点击“确定”即完成新项目的创建，如下图所示：



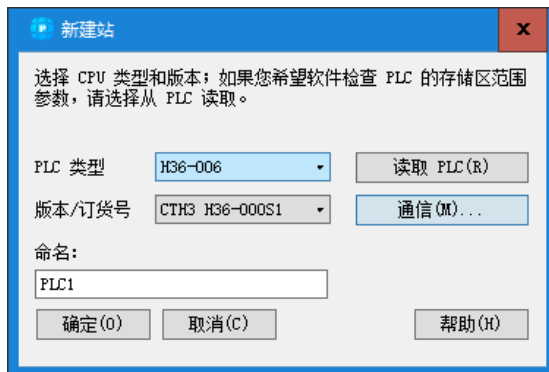
MagicWorks PLC 管理器是一个多文档应用程序，每次只能创建一个运行实例，在同一个运行实例里管理多个项目，用户可以在不同的项目子窗口之间进行切换并复制粘贴项目对象。

插入 PLC 站点

在项目管理器界面右键选择创建好的项目，选择“插入新对象”→“PLC 站”，或是在菜单项选择“插入”→“PLC 站”，可在当前工程插入一个 PLC 站，如下图所示：



在弹出的“新建站”中，您可以按照需要选择“PLC 类型”和“版本”，也可以修改 PLC 名称及为其选择通信方式。详细通信连接方式设置请参考章节 [6 通信设置](#)。

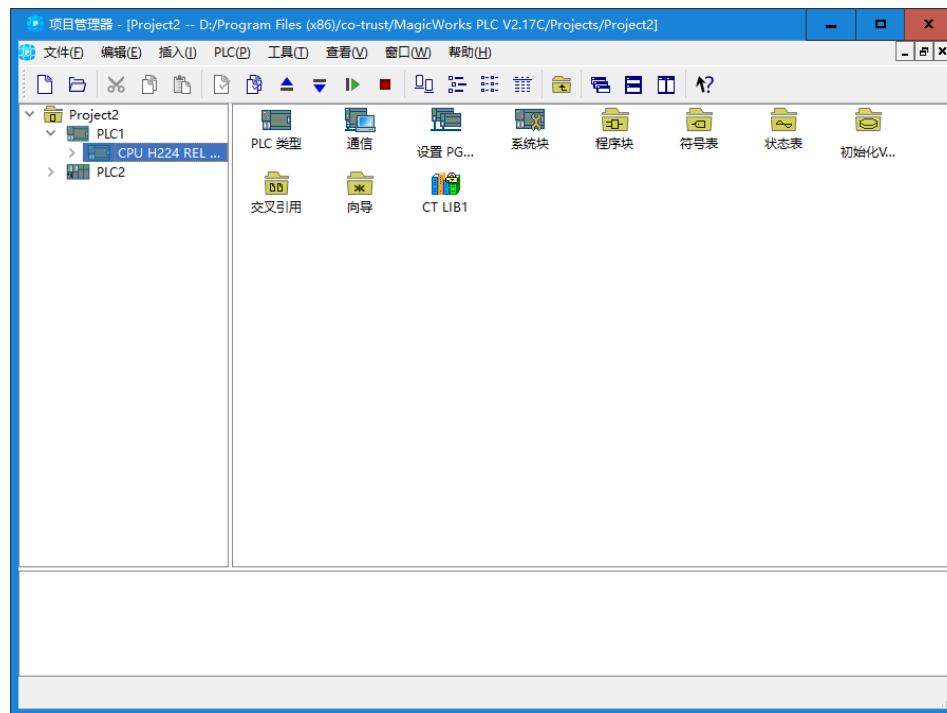


PLC 站点成功插入后如下图所示：

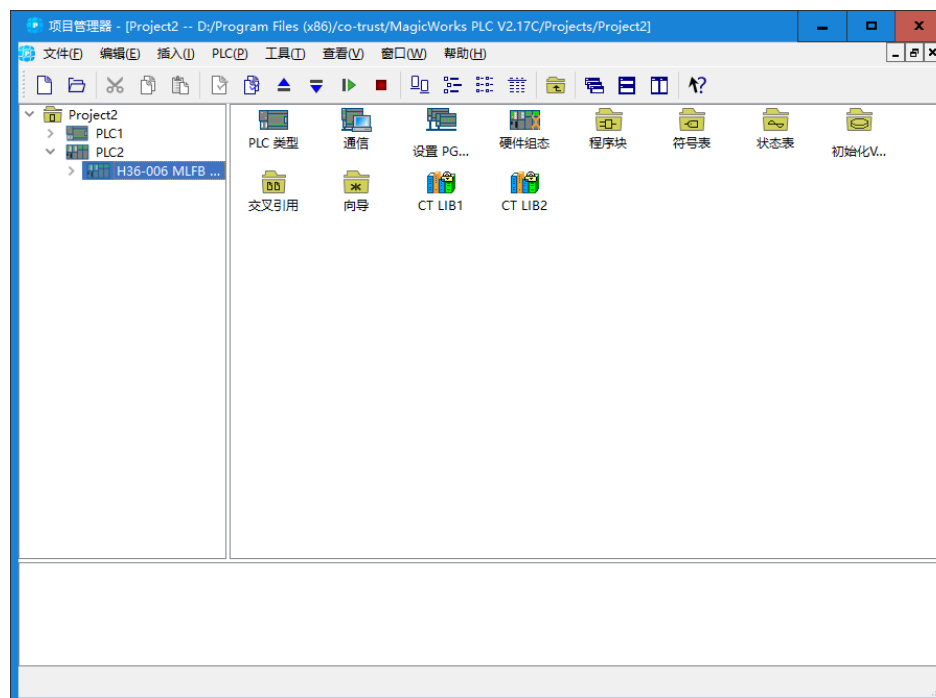
<提示>在 MagicWorksPLC 编程软件中支持 200 系列工程单向转换为 300 系列工程。

用户选择 CTH3 PLC 类型后，转换时自动备份用户原程序，转换后生成一个新工程。

200 系列 PLC 站点项目管理器界面



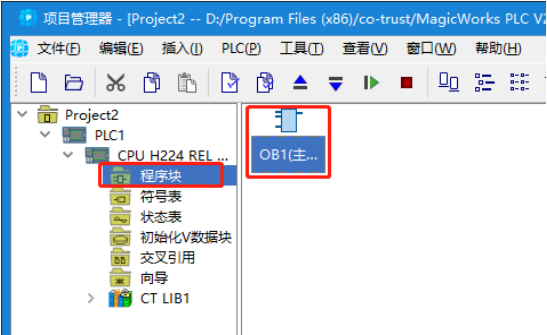
300 系列 PLC 站点项目管理器界面



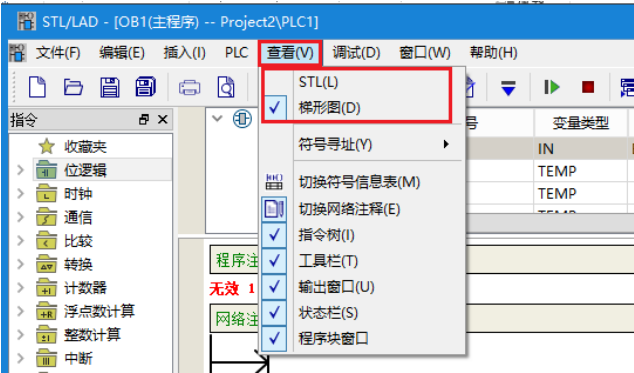
3.2 编辑程序

在插入的 PLC 站点子目录选项下选择“程序块”，在右边的项目窗口双击“OB1（主程序）”，

进入程序编辑器界面。



您使用 STL 或 LAD 编程语言编辑程序，在程序编辑器界面选择“查看”，在弹出的菜单项中可选择 STL 或梯形图两种编程方式的切换，勾选状态即表示选中该方式。



下文为两种编程方式的操作步骤：

1、在 STL 中输入语句

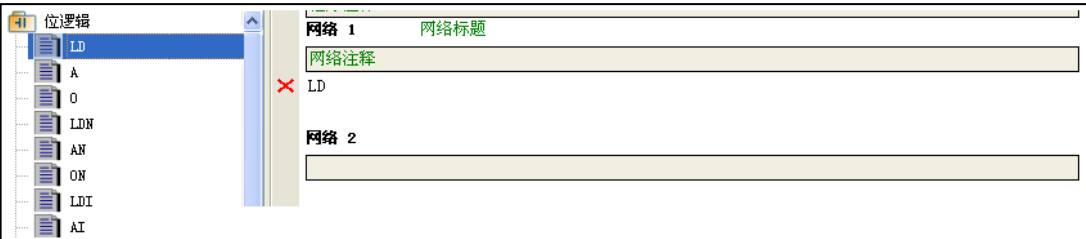
您可以在程序编辑器窗口中键入 STL 指令或用在指令树双击的方式插入 STL 指令。

欲在 STL 编辑器中键入语句：

- 1) 将光标放在一行的开始。
- 2) 输入指令。（例如：LD）
- 3) 从键盘按空格键或 TAB（制表符）键。
- 4) 输入操作数，操作数可以是绝对值（例如：I0.0）、符号（例如：Input1）或常数（例如：100）。带指令和地址的整行范例：LD I0.0。
- 5) 如果无须注解，敲 ENTER 键，光标移至下一行。
- 6) 如需注解，可在一行中通过//号来开始一句注释。

从指令树双击的方式插入指令：

- 1) 展开指令树，选择要插入的指令。
- 2) 鼠标双击，选中的指令就会自动插入到右边的编辑窗口光标所在的位置。





提示

- 每行仅输入一个指令和/或注解。如果一行有多个操作数，您可以使用空格、制表符或逗号将其分开。
- 当程序超过一个屏蔽时，使用滚动条查看程序。
- 请勿使用 MEND、RET、RETI、SBR 或 INT 终止程序的 POU（OB、FC、FB）。这些指令由编译程序自动插入；如果您在 POU 中使用这些指令，编译程序会返回一则错误。

2、在 LAD 中输入程序

LAD 编程的相关规则

• 触点放置

每个网络必须以一个触点开始，网络不能以触点结束。

• 线圈放置

网络不能以线圈开始，线圈用于终止逻辑网络。一个网络可有若干个线圈，只要线圈位于该特定网络的并行分支上。不能在网络上串联一个以上线圈（即不能在一个网络的一条水平线上放置多个线圈）。

• 方框放置

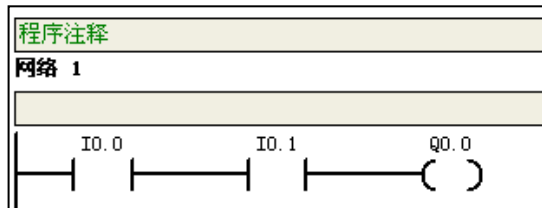
如果方框有 ENO，使能位扩充至方框外；这意味着您可以在方框后放置更多的指令。在网络的同级线路中，可串联若干个带 ENO 的方框。如果方框没有 ENO，则不能在其后放置任何指令。

• 网络尺寸

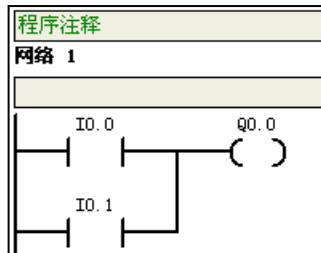
您可将程序编辑器窗口视作划分为单元格的网格（单元格是可放置指令、为参数指定值或绘制线段的区域）。在网格中，一个单独的网络最多能垂直扩充 32 个单元格或水平扩充 32 个单元。点击程序编辑器工具栏最右边的放大/缩小按钮 ，上下拖动滑块可调节梯形图指令的显示宽度。

下面列出一些 LAD 编辑器中可能存在的逻辑结构：

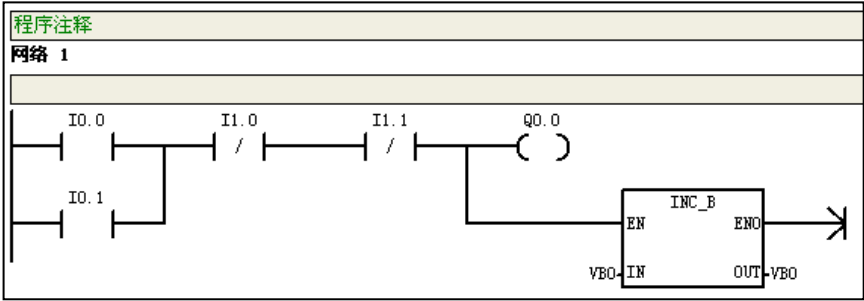
① 串联



② 并联



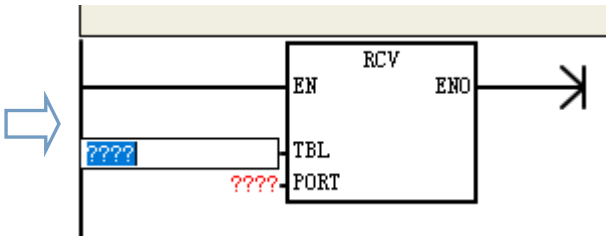
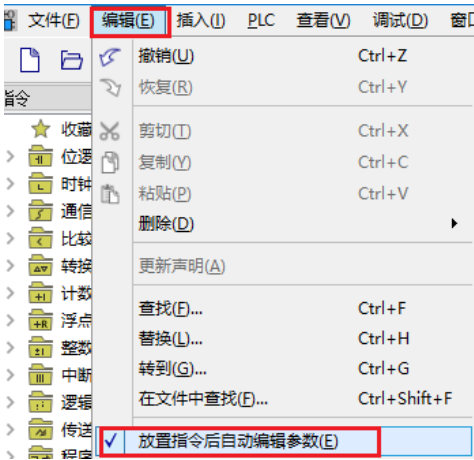
③ 既有串联也有并联的复杂逻辑



输入 LAD 指令的方法

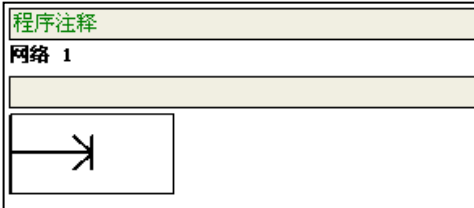
您可以双击指令树或从指令树拖曳或借助工具条按钮来输入 LAD 指令。

在编辑菜单中勾选“放置指令后自动编辑参数”，则指令放置后光标会自动进入第一个参数启动编辑，如果没有勾选该选项，则指令放置后光标自动挪到下一个指令放置位置，您可以通过鼠标单击参数输入框 **???** 使之变为可编辑状态，键入您要设置的参数。该选项默认勾选。双击后，指令在程序编辑器窗口中显示，鼠标单击参数输入框 **???** 使之变为可编辑状态，键入您要设置的参数。

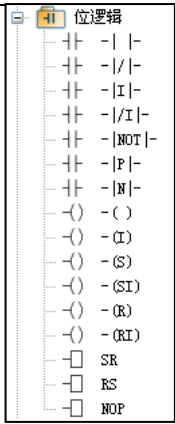


从指令树双击

1) 在程序编辑器窗口中将光标放在所需的位置，您也可以通过键盘上的上下左右键来移动光标，一个选择方框在该位置周围出现。

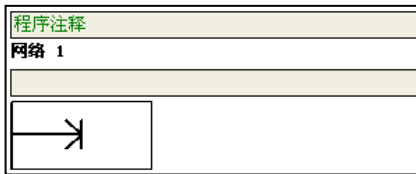


2) 在指令树中，浏览至所需的指令并双击该指令。



使用工具条按钮或功能键

1) 在程序编辑器窗口中将光标放在所需的位置，您也可以通过键盘上的上下左右键来移动光标，一个选择方框在位置周围出现。



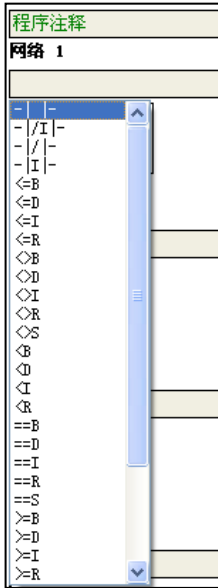
2) 点击适当的工具条按钮   或使用适当的功能键（F4=触点、F6=线圈、F9=方框）插入一个类属指令。



提示

该类属指令不能编址，并会阻止编译程序；您必须按步骤 3 所示选择一个具体指令类型。

3) 出现一个下拉列表。滚动或键入开头的几个字母，浏览至所需的指令。双击所需的指令或使用 ENTER 键插入该指令。



划线

您可以从“程序”工具条使用水平和垂直线  ，或按住键盘上的 CTRL 键并按左、右、上或下箭头键，必要时在网络和左侧电源杆元素之间划线，完成网络绘制。

如何输入程序注释

程序编辑器中的注释共分为五个部分：块注释、网络标题、网络注释、程序中间的注释（只有 STL 编程语言才支持程序中间加注释）。

块注释

在网络 1 上方的灰色方框中点击，输入 POU 注解。

您可以用点击"切换 POU 注解"按钮或选择和取消选择检视（View）> POU 注解（POU Comments）选项的方法，在 POU 注解"打开"（可见）或"关闭"（隐藏）之间切换。

每条 POU 注解最多可使用 4096 个字符。POU 注解是选用项目，显示时始终位于 POU 顶端，并在第一个网络之前显示。

网络标题

将光标放在网络标题行中的任何位置，输入一个可识别该逻辑网络的标题。

网络注释

在网络 1 下方的灰色方框中点击，输入网络注解。您可以输入一条识别该逻辑网络的注解，并增加有关网络内容的说明。

您可以用点击"切换网络注解"按钮或选择和取消选择检视（View）> 网络注解（Network Comments）选项的方法，在网络注解"打开"

（可见）或"关闭"（隐藏）之间切换。

网络注解中最多可使用 4096 个字符。

程序中间的注释

所有 STL 中的注解前均须有双斜线"//"

您可以将一整行作为注解；您可以把注解作为一行放在有效指令之后。双斜线至行末的所有内容均被编译程序视作注解。

最大行长（包括注解）为 255 个字符。

符号信息表

当程序中使用到符号变量时，符号信息表就会出现在程序编辑器窗口，如图：

网络 1		网络标题
		网络注释
LD	Always_On:SM0.0	
=	Q0.0	
LD	Clock_1s:SM0.5	
=	Q0.1	
符号	地址	注释
Always_On	SM0.0	Always ON
Clock_1s	SM0.5	Clock pulse that is ON for 0.5 s, OFF for 0.5 s, for...

当您查看符号信息表时，符号名、绝对地址、数据类型和注释会在 LAD/FBD/STL 程序中每个网络下方按字母顺序显示。未定义的符号名也包括在内。

该表列出该网络中所有符号的信息；不包含全局符号的网络不显示符号信息表。所有重复条目均被删除。

如果您位于 IEC 模式，表格显示一个附加数据类型列。

符号信息表无法编辑。

提示：

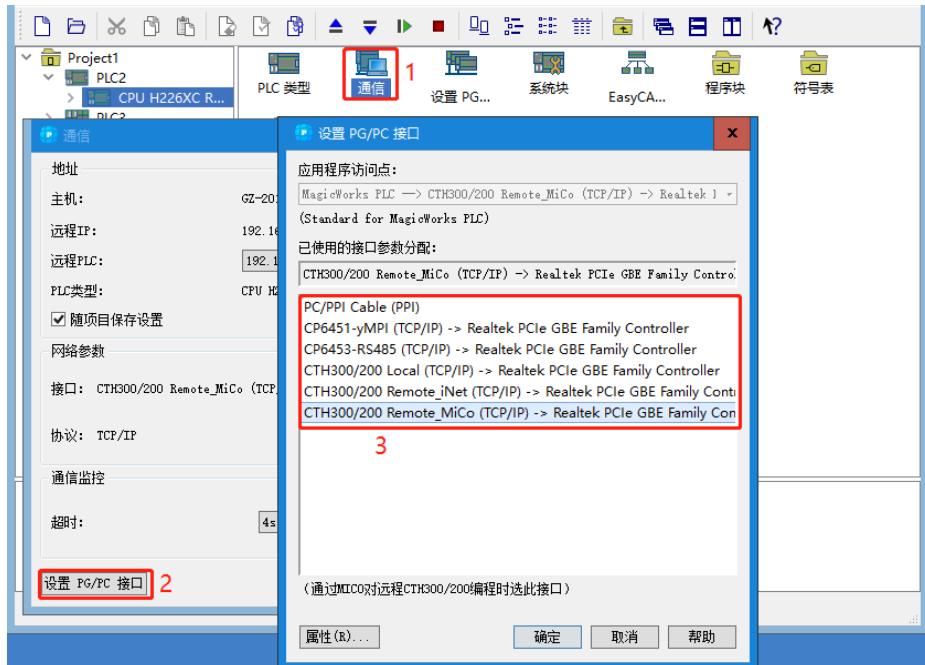
- 1) 用光标键浏览符号信息。
- 2) 对符号信息表的一个单元格双击，会弹出编辑符号对话框，用户可以在此重新编辑符号。
- 3) 您可以从列边缘拖动分隔光标，重新确定表格大小。表格不得超过 32 个单元格。所有无法放入一个单元格中的文本均以省略号结束。

3.3 建立通信和编译程序

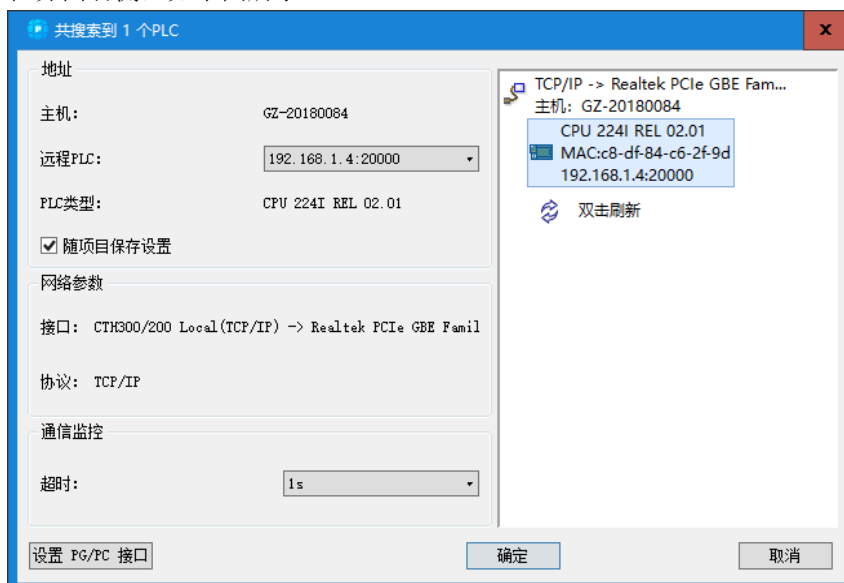
建立通信

建立通信前，请先确认 PLC 与编程设备之间已经连接好电缆且上电成功。

1) 在项目管理器界面双击“通信”图标，在弹出的“通信”界面点击“设置 PC/PG 接口”进行接口参数分配设置。或在项目管理器界面点击“设置 PC/PG 接口”直接进入设置界面。在此设置界面用户可根据实际需要选择相应的接口参数，具体设置请参考章节 6 通信设置。若使用编程电缆连接 PLC，请选择“PC/PPI Cable(PPI)”；若使用标准网线连接 PLC，请选择“CTH300/200 Local(TCP/IP)->Realtek PCIe GBE Family Controller”。



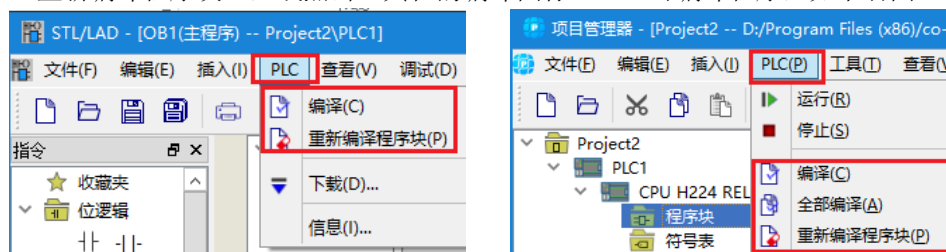
2) 完成上述接口参数分配后回到“通信”界面双击“双击刷新”，成功连接的 PLC 则会出现 在界面右侧，如下图所示。



MagicWorks PLC 在同一时间仅与一个 PLC 通讯。如果你想更改通信 PLC，您可以双击另一个 PLC，更改为与该 PLC 通讯。

编译程序

编辑好的程序在下载前需进行编译，编译无错才能进行程序的下载，MagicWorks PLC 提供 2 种编译程序的方法。在程序编辑器界面选择“PLC”→“编译”/“重新编译程序块”，或点击工具栏的编译图标“”可编译程序，如下左图；在项目管理器界面选择“PLC”→“编译”/“重新编译程序块”，或点击工具栏的编译图标“”可编译程序，如下右图。



详细的编译功能说明参见章节 [5.8 编译功能](#)。

3.4 下载和运行程序

下载程序

如果您已经成功地在运行 MagicWorks PLC 的个人计算机和 PLC 之间建立通讯，您可以将程序下载至该 PLC，请按照下列步骤操作：



提示

从当前工程将程序块、初始化 V 内存数据块、系统块、硬件组态块和 EasyCAN 配置块下载至 PLC 时，下载的块内容会覆盖目前在 PLC 中的块内容（如果 PLC 中存在这些块），而下载的配方与数据记录则会覆盖存储卡的相应内容。在开始下载之前，请核实您希望覆盖的块。

- 1) 下载至 PLC 之前，您必须将 PLC 设为 STOP（停止）模式。
- 2) 在“项目管理器”界面或是“程序编辑器”界面使用以下方法中的一种打开下载对话框：
 - 点击工具条中的“”按钮；
 - 选择菜单命令“PLC”→“下载”；
 - 按 Ctrl+D 快捷键组合；
- 3) “选项”方框显示程序块、数据块、系统块/硬件组态块、配方、数据记录和 EasyCAN 配置块复选框。请核实已选择您希望下载的块复选框，并取消选择您不希望下载的任何块，然后点击“下载”按钮，开始下载程序。
- 4) 若下载成功，PLC 运行程序之前，须从 STOP（停止）模式转换回 RUN（运行）模式。
- 5) 如果下载成功或有错误，信息框会给出相应的提示。

运行程序

使用以下一种方法改变 PLC 操作模式：

手动切换 PLC 的模式开关，可以在两种模式中切换：运行（RUN），停止（STOP）；

选择“PLC”→“运行”菜单命令，将 PLC 切换至运行状态，如果此时模式开关在 STOP 状态，系统会给出错误信息，如下：



选择“PLC”→“停止”菜单命令，将 PLC 切换至停机状态。

- 点击工具条上的“运行”按钮，将 PLC 切换至运行状态，如果此时模式开关在 STOP 状态，系统同样会给出错误信息；
- 点击工具条上的“停止”按钮，将 PLC 切换至停机状态；
- 在程序中插入 STOP 指令，使 PLC 在一定条件下停机。

PLC 模式开关说明

- 将模式开关打在停止模式，停止程序的执行；
- 将模式开关打在运行模式，启动程序的执行；
- 如果模式开关打在 STOP，且电源状态发生变化，则当电源恢复时，PLC 会自动进入 STOP 模式；
- 如果模式开关处于 RUN 模式，且电源状态发生变化，则当电源恢复时，PLC 会进入 RUN 模式。

PLC 操作模式说明

COTRUST PLC 有两种操作模式：停止模式和运行模式。PLC 前面板上的 LED 状态显示了当前的操作模式。

停止模式下，不执行程序，不执行输入更新，用户中断条件被禁止，PLC 处于半空闲状态。

您可以新建、编辑、下载、上载或删除用户程序，也可以使用调试协助，例如首次扫描和多次扫描功能。STOP 模式下发生通信中断时，PLC 收到信息并在适当的情况下执行请求。PLC 模式下也会执行 PLC 自诊断程序，包括定期检查操作系统 EEPROM、I/O 模块状态检查和在每次访问扩展 I/O 时执行的 I/O 扩展总线检查。PLC 操作系统会记录致命错误，并强制从运行模式更改为停止模式。如果 PLC 检测到致命错误，在致命错误条件依然存在时不允许从停止模式更改为运行模式。PLC 操作系统功能也记录非致命错误，用于检查，但不会停机。

在运行模式下，PLC 将执行用户程序。PLC 读取输入、执行程序、写入输出、对通信请求作出应答、更新智能模块、执行内部管理工作以及对中断条件作出应答。在运行模式中，您可以新建、编辑和监控程序操作和数据，使用提供的调试功能可增强追踪程序操作和识别编程问题的能力。

每次扫描周期开始时先读取当前输入位数值，并将这些数值写入输入映象寄存器。与具有实际输入位位于同一个字节中，但无相应的实际物理输入点的输入位，会在每次输入更新循环时在映象寄存器中被清零，除非它被强制。读取输入后，程序从第一条指令开始执行，直至遇到结束指令。遇到结束指令时，PLC 检查系统的智能模块是否需要服务。如果需要，信息将被读取并缓存，以用于循环周期的下一阶段。在扫描周期的信息处理阶段，从通信端口接收的信息被处理。相关的应答完成后被暂存，等待在适当的时间传送给通信请求方。自我诊断检查包括对操作系统 EEPROM 和用户程序存储区以及 I/O 模块状态进行定期检查。

最后，输出映象寄存器数值被写入输出模块，从而完成一次扫描周期。

3.5 在线帮助的使用

在项目管理器主界面选择菜单项“帮助”，即可使用帮助信息。帮助菜单项如下：

目录

选择菜单命令“帮助”→“目录”可显示有关 MagicWorks PLC 的“帮助”。可从目录页面浏览至其它帮助主题。通过“索引”和“搜索”按钮可显示有关特定术语的信息。

MagicWorks PLC 帮助为您提供使用 MagicWorks PLC 软件时所需的基本知识。

为对 MagicWorks PLC 帮助加以补充，还为您提供针对不同的 MagicWorks PLC 应用程序的在线帮助（例如 MagicWorks PLC 项目管理器“帮助”）。应用程序帮助为您提供有关所选对象（如菜单命令或对话框）的上下文相关信息。

上下文相关帮助

选择菜单命令“帮助”→“上下文相关帮助”可显示当前上下文的帮助主题（例如，有关所选的对象）。

可通过以下方式启动上下文关联帮助：

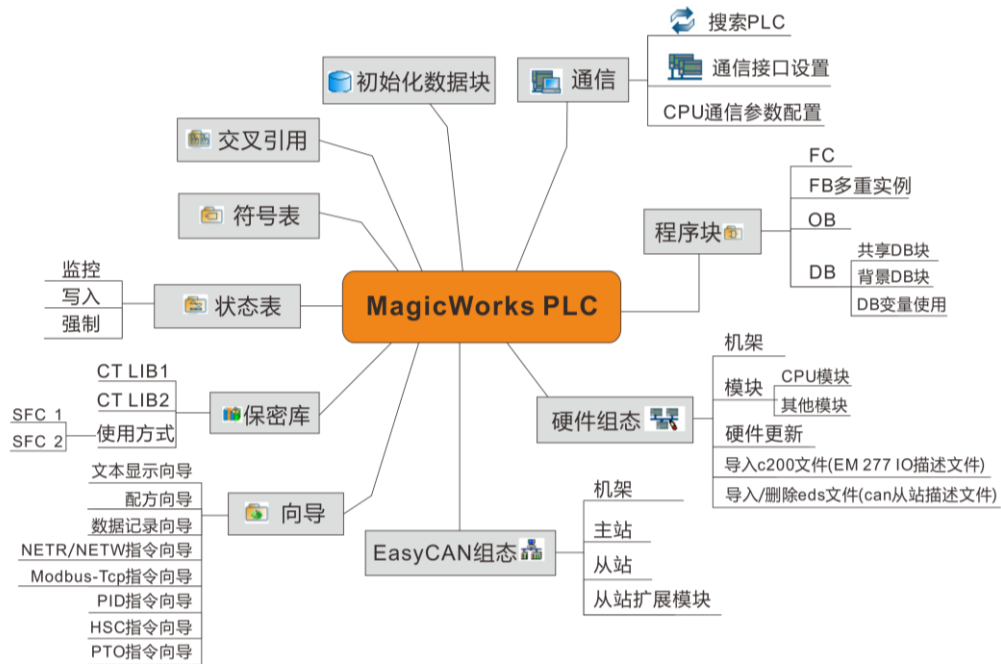
- 按 F1；
- 在对话框中单击“帮助”按钮；
- 单击一次 ，然后在使用的界面中再单击一次需要帮助的对象；

关于

菜单命令“帮助”→“关于”可用来查看当前应用程序组件的版本号和版权信息。

4 软件组成

MagicWorks PLC 编程系统所含的独立应用程序包括：项目管理器、符号表编辑器、初始化数据块编辑器、程序编辑器、交叉引用查看器、状态表编辑器、DB 编辑器、EasyCAN 组态和硬件组态。



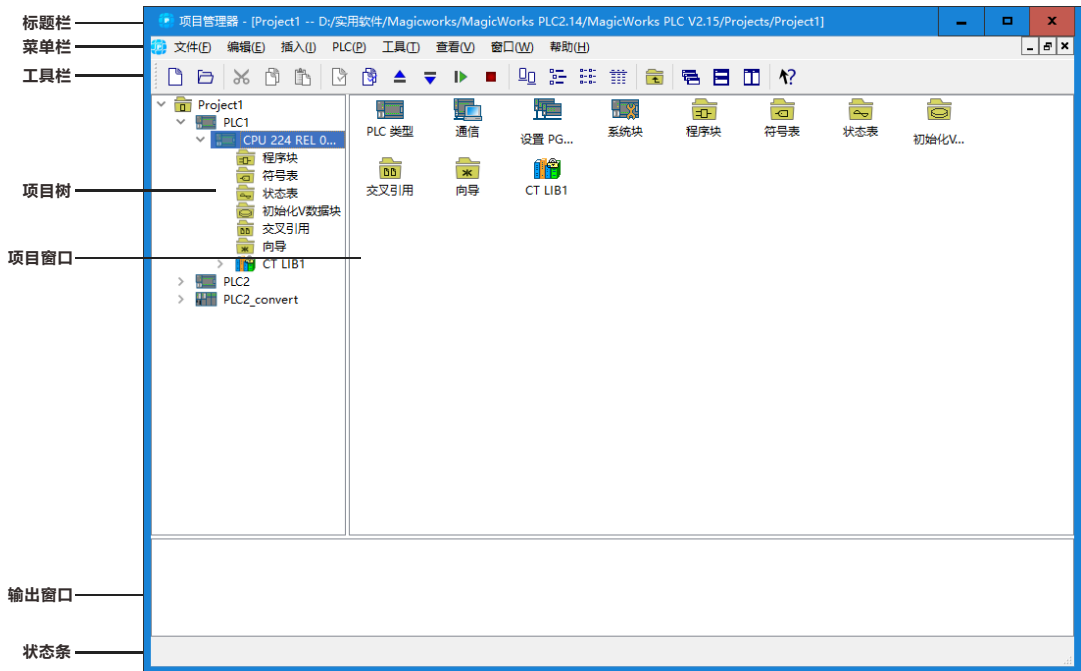
各软件均采用多文档管理的风格布局和窗口元件组织结构。各编辑界面主要包括：菜单栏、工具栏、程序元素窗口、工作区和状态栏。

本章概要

- ❑ 项目管理器窗口
- ❑ 符号表编辑器窗口
- ❑ 初始化 V 内存数据块编辑器窗口
- ❑ 程序编辑器窗口
- ❑ 交叉引用查看器窗口
- ❑ 状态表编辑器窗口
- ❑ EasyCAN 组态窗口
- ❑ 硬件组态窗口

4.1 项目管理器窗口

打开 MagicWorks PLC 后，新建一个工程，然后为该工程新建一个 PLC 对象，PLC 新建成功后的项目管理器主界面如下图所示：



4.1.1 菜单栏

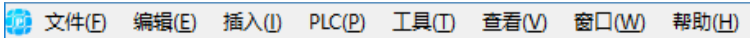


表 4-1 文件(F)菜单

名称	图标	功能描述
新建(N)		创建一个新的项目
打开(O)...		打开硬盘上保存的硬件配置文件
关闭		关闭当前页面
保存(S)...		将当前项目保存到硬盘
另存为(A)...		复制已打开的项目，并用新的名称对其进行保存
删除工程		移除硬盘上已经存在的工程文件
设置密码(W)		允许您限制访问当前项目
新建库(L)		基于您的项目，创建一个指令库
添加/删除库(R)		在项目中增加指令库或从项目拆卸指令库
库存储区(M)		为项目目前使用的指令库分配存储区
更新库		更新指令库至最新版本
1、 2、 3、 4		最近打开的项目，选择所需号码可打开相应项目
退出(X)		关闭该应用程序的所有窗口，并退出该程序

表 4-2 编辑(E)菜单

名称	图标	功能描述
剪切(I)		剪切所选对象并将其放入剪贴板
复制(Y)		复制所选对象并将其放入剪贴板
粘贴(P)		在光标位置粘贴剪贴板的内容
删除(D)		删除所选择的对象
全选	-	选择当前窗口中的所有对象
重命名(N)	-	修改所选对象的名称
对象属性(R)...	-	显示当前块的属性以进行编辑
打开对象	-	打开所选择的对象

表 4-3 插入(I)菜单

名称	图标	功能描述
PLC 站(P)	-	在当前工程中插入一个新的 PLC 站
符号表(Y)	-	新建一个符号表插入到当前项目中
初始化 V 内存数据块(I)	-	插入 V 内存数据块
组织块(O)	-	新建一个组织块插入到当前项目中
功能块(B)	-	新建一个功能块插入到当前项目中
功能(C)	-	新建一个 FC 插入到当前项目中
DB(D)	-	新建一个背景数据块插入到当前项目中
状态表(S)	-	新建一个状态表插入到当前项目中

表 4-4 PLC(P)菜单

名称	图标	功能描述
运行(R)		将 PLC 设置为 RUN（运行）模式
停止(S)		将 PLC 设置为 STOP（停止）模式
编译(C)		编译当前改动过的程序块
全部编译(A)		编译整个项目
重新编译程序块(P)		清理并重新编译所有程序块。在使用编译遇到不明错误或改动比较多时建议使用此功能。
上载(U)		从 PLC 将项目组件上载至 MagicWorks PLC 程序编辑器
下载(D)		从 MagicWorks PLC 向 PLC 下载项目组件
清除(L)	-	清除 PLC 存储区的指定区域
上电复位(W)	-	从 PLC 清除致命错误并返回 RUN（运行）模式
信息(I)	-	读取 PLC 信息
编程卡配置(G)	-	配置编程卡参数：选择需要下载后被覆盖的块、设置编程卡的使用次数
存储卡擦除(E)	-	擦除 PLC 存储卡
从 RAM 建立数据块(B)	-	将 PLC 的 V 存储区保存至 PLC 的 EEPROM。
实时时钟(D)	-	查看或设置存储在 PLC 中的当前时间和日期
比较(M)	-	将 MagicWorks PLC 中当前打开的项目与 PLC

		的内容进行比较
运行模式编程(P)	-	在 RUN（运行）模式下执行程序编辑功能
类型(T)	-	选择 PLC 类型
获取诊断信息(G)	-	在线获取诊断信息
在线查看数据记录(V)	-	在线查看数据记录

表 4-5 工具(T)菜单

名称	图标	功能描述
PID 指令向导(P)		打开 PID 指令向导对话框
NETR/W-PPI 指令向导(N)		打开 NETR/W-PPI 指令向导对话框
NETR/W-UDP 指令向导(E)		打开 NETR/W-UDP 指令向导对话框
MODBUS-TCP 指令向导		打开 MODBUS-TCP 指令向导对话框
HSC 指令向导(H)		打开 HSC 指令向导对话框
PTO 指令向导		打开 PTO 指令向导
文本显示向导(T)		打开文本向导对话框
配方向导(W)		打开配方向导对话框
数据记录向导(L)		打开数据记录向导对话框
设置 PG/PC 接口(I)		对设备的接口和参数分配进行处理，并对其互相分配
更新 CPU 固件(TCP/IP)(F)		更新 CPU200/300 的固件版本
离线查看数据记录(V)		
擦除设备 ID		设备远程连接的服务器变更时请先使用此功能擦除设备 ID，再下载新的系统块/硬件组态块，否则有可能无法成功连接到新的远程服务器。其它情况下无需使用此功能。
选项(O)		自定义编程环境： • 编程模式 • 助记符集 • 默认编辑器 • 语言 • 默认文件位置 • 新项目的 PLC 类型 • 系统符号表

表 4-6 查看(V)菜单

名称	图标	功能描述
大图标(R)		工作区域以大图标形式显示
小图标(M)		工作区域以小图标形式显示
列表(L)		工作区域以列表形式显示
详细资料(D)		工作区域以详细信息的形式显示
工具栏(T)	✓	勾选表示显示工具栏，否则不显示。
状态栏(S)	✓	勾选表示显示状态栏，否则不显示。

表 4-7 窗口(W)菜单

名称	图标	功能（状态栏信息）
层叠窗口（ <u>C</u> ）		排列此应用程序中的全部打开窗口以使其重叠
横向平铺（ <u>H</u> ）		从上到下均匀排列此应用程序的所有打开窗口
纵向平铺（ <u>V</u> ）		从左到右均匀排列此应用程序的所有打开窗口
全部关闭（ <u>L</u> ）		关闭在此应用程序中打开的全部窗口
1, 2, 3, ...		此应用程序中打开的所有数据块

表 4-8 帮助(H)菜单

名称	图标	功能（状态栏信息）
目录（ <u>C</u> ）...		提供对帮助信息进行访问的各种不同方式
上下文相关帮助（ <u>H</u> ）		显示与当前上下文相关的帮助文本
检查更新(<u>C</u>)		对当前软件版本进行检查，若发现新版本即可选择是否更新
自动检查更新		自动检查并将当前软件版本更新至最新版本
关于（ <u>A</u> ）...		提供此应用程序的版本信息

4.1.2 工具栏

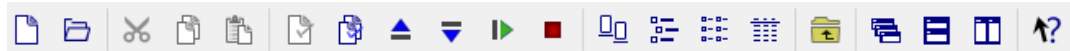
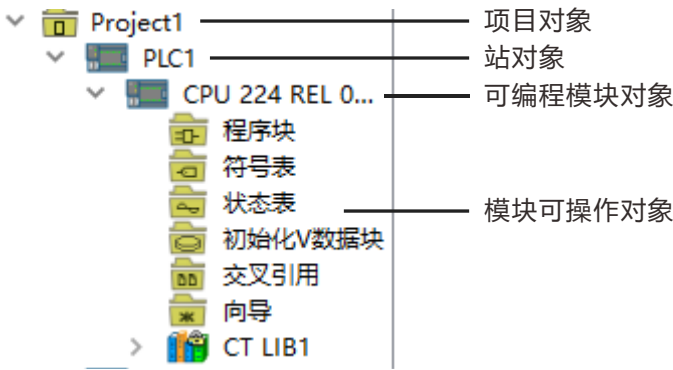


表 4-9 工具栏菜单

名称	图标	功能描述
新建(<u>N</u>)		创建一个新的工程
打开(<u>O</u>)...		打开已有的工程文件
剪切		将选定内容剪切并放置于 Windows 剪贴板中
复制		复制选定内容到粘贴缓冲区
粘贴		将粘贴缓冲区的对象粘贴到指定位置
编译		编译当前选中的块
全部编译		编译所有块，包括程序块、数据块和系统块
上载		从 PLC 中将项目组件上载至编程软件中
下载		从编程软件下载项目组件或程序到 PLC 中
运行		该命令使 PLC 处于运行状态
停止		该命令使 PLC 处于停止状态
大图标		将工作区域的图标设置为大号显示
小图标		将工作区域的图标设置为小号显示
列表		将工作区域的图标以列表的形式排列
详细资料		显示工作区域的图标的详细资料
向上一层		退回到上一层
层叠窗口		排列此应用程序中的全部打开窗口以使其重叠
横向平铺命令		从上到下均匀排列此应用程序的所有打开窗口

纵向平铺命令		从左到右均匀排列此应用程序的所有打开窗口
帮助		该命令可以打开帮助文档

4.1.3 项目树



项目文件夹

项目文件夹用于存储用户创建的项目数据。

<备注> MagicWorks PLC V1.52 以前的版本只支持一个项目包含一个 PLC，V1.52 后的版本支持一个项目可创建多个 PLC 站。

PLC 站节点

PLC 站节点用于存储当前的 CPU 对象，您可以选中此节点查看当前的 CPU 名称和类型。PLC 站节点支持重命名。

CPU 文件夹

CPU 文件夹用于存储当前 CPU 所包含的程序和数据，属于 CPU 的数据包括下列内容：

- 保密库 

保密库 CT LIB1 和 CT LIB2 可配置的子项目包括符号表、程序包、交叉引用及向导：



- 符号表文件夹 

符号表文件夹用来存储用户为当前项目创建的符号表，包括：

POU 符号表：包含您为项目的 POU 或其他组件（例如数据块、状态表或符号表）指定的符号信息，只读。

SC-200 符号表：包含系统中常用的 SM 位/字节/字/双字的符号信息，只读。

用户自定义符号表：用户自己创建的符号变量信息，可编辑，用户可在此新建、修改或删除一个符号变量。

- 程序块文件夹 

程序块文件夹是用来存储用户为当前项目创建的程序块，包含以下类型：

组织块 OB：OB1 作为用户程序中的主程序，其它 OB 作为中断程序。

功能块 FB（仅中型 PLC 支持）：用户编写的包含经常使用的功能的子程序，有存储区。

功能 FC：用户编写的包含经常使用的功能的子程序，无存储区。

数据块 DB（仅中型 PLC 支持）：包含背景数据块（DI）和共享数据块（DB）。其中背景数据块（DI）是调用 FB 时用于传递参数的数据块；共享数据块（DB）是存储用户数据的数据区域，供所有的块共享，即为 V 存储空间。

<备注> 有关程序块的使用请参考章节 [5.3 程序块](#)。

- 初始化 V 内存数据块文件夹 

初始化 V 内存数据块文件夹用来存储用户为当前项目所创建的初始化数据页，用户可以在右边的窗口中双击其中一个初始化数据页图标以打开文件进行查看和编辑。

- 交叉引用文件夹 

交叉引用文件夹用来存储交叉引用图标，用户可以在右边窗口双击图标选定以某种方式查看当前程序中是否已经使用和在哪里使用某一符号名或存储区赋值的情况。用户可以通过三种方式来进行查看：位使用表、字节使用表、交叉引用表、沿引用表。

<注意> 在查看交叉引用表之前必须先编译程序。

- 状态表文件夹 

状态表文件夹用来存储用户为当前项目创建的状态表页文件，用户可以在右边的窗口中双击其中一个状态表页图标以打开文件。

- 向导文件夹 

向导文件夹用来存储当前版本支持的向导图标，用户可以通过点击项目树的向导结点打开向导文件夹来查看。当前支持的向导为：文本显示向导、配方向导、数据记录向导、NETR/NETW 指令向导、MODBUS-TCP 向导、PID 向导、PTO 向导、HSC 向导。

<备注> 有关各向导的使用请参考章节 [5.5 向导的使用](#)。

4.1.4 输出窗口

输出窗口位于项目树和工作窗口的下方，可用来查看工程编译的结果。

输出窗口保留最近编译的块信息和在编译过程中产生的错误列表，如下图：



修正程序后，重新执行编译，输出窗口将自动清除旧的编译信息并输出新的编译结果。

将鼠标放在输出窗口中，用鼠标右键点击，可拷贝或删除输出窗口的内容。

将鼠标移到输出窗口的上边沿，待光标变为可拖动形状后，按下鼠标左键并上下拖动可以改变输出窗口的高度。

双击程序块/数据块编译信息的错误行，光标会自动跳转到错误网络/行。

4.1.5 状态条

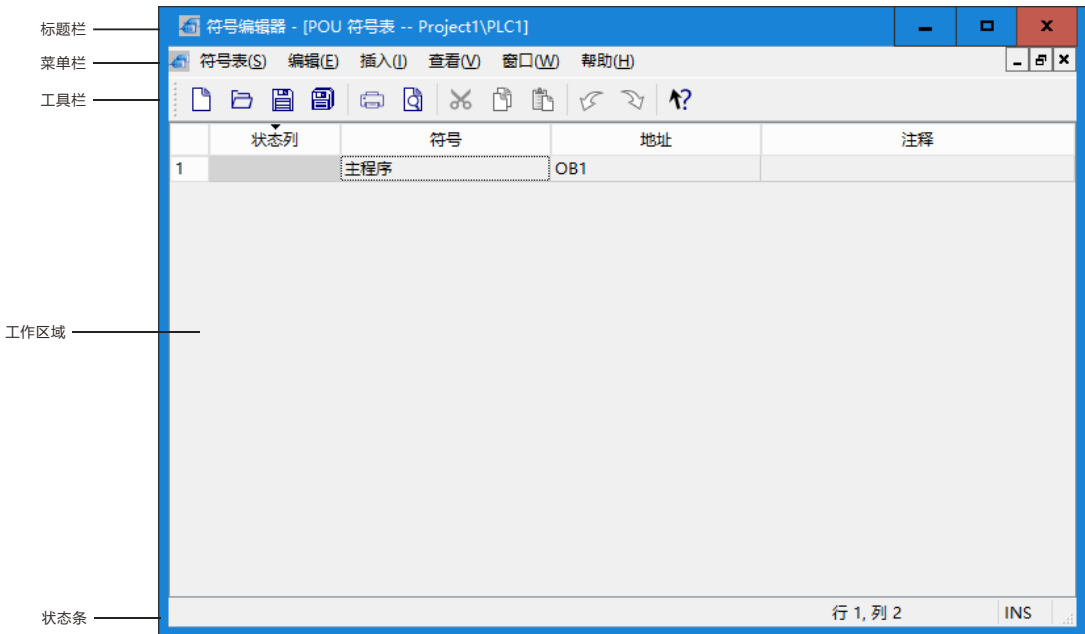
状态栏位于主程序底部，提供有关在 MagicWorks PLC 中操作的信息。

当您在编辑模式中工作时，显示编辑器信息。状态栏根据具体情形显示下列信息：

- 简要状态说明
- 当前网络号码
- 光标位置（用于 STL 或 LAD 编辑器的行和列）
- 表示背景任务状态的图标，如保存或打印等

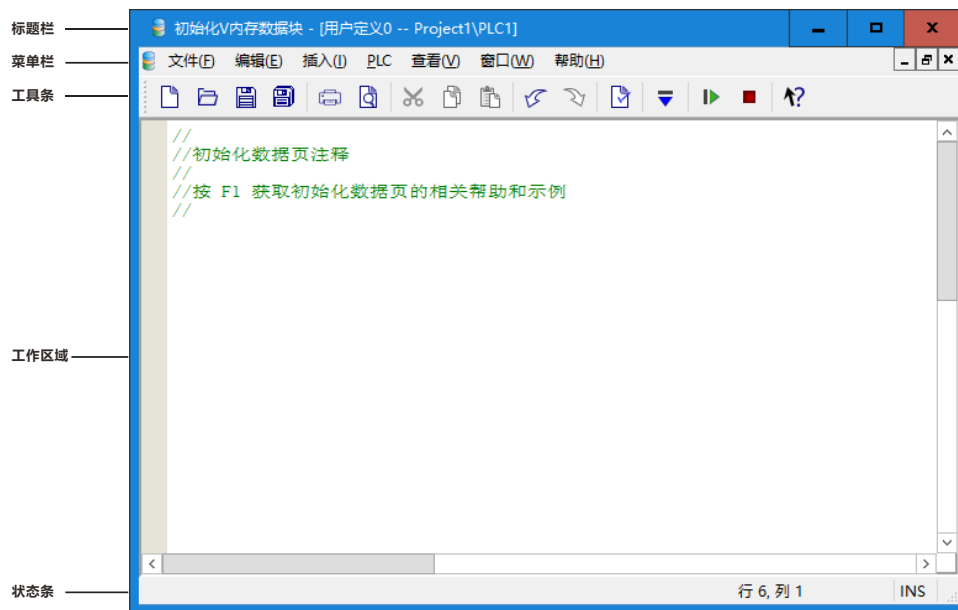
4.2 符号表编辑器窗口

在此界面您可以为当前项目创建符号表，包含 POU 符号表及用户自定义符号表。



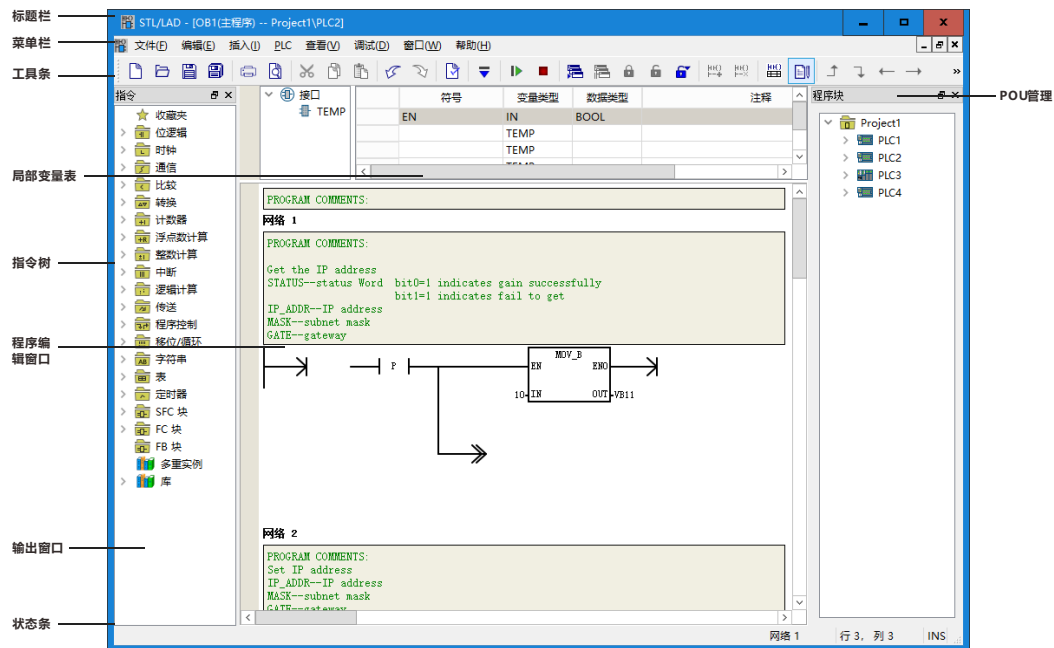
4.3 初始化 V 内存数据块编辑器窗口

在此界面您可以为当前项目创建初始化数据页，对 PLC 的 V 存储区进行初始化。



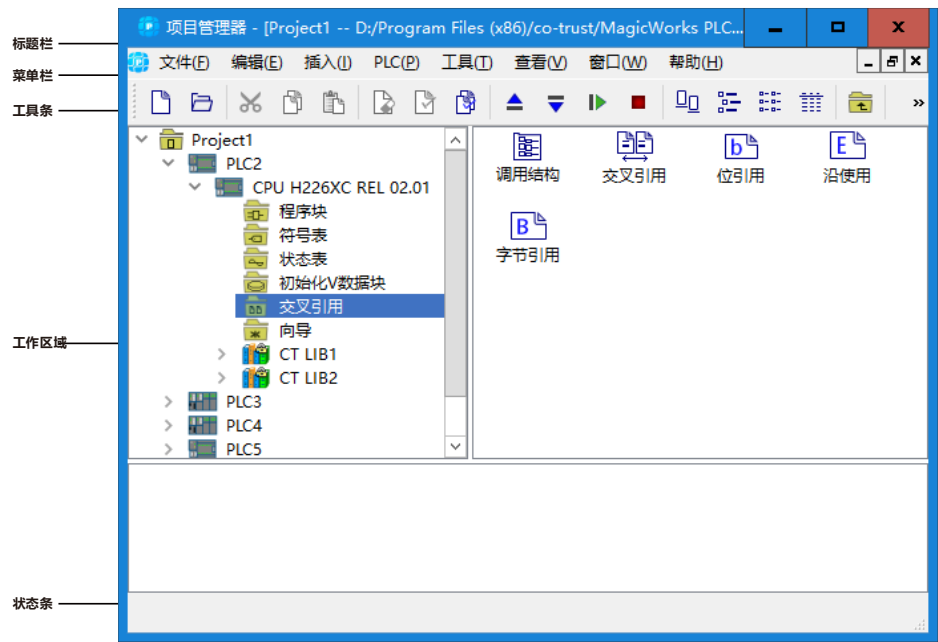
4.4 程序编辑器窗口

在此页面您可以进行程序的编辑、编译及工程的上下载等。



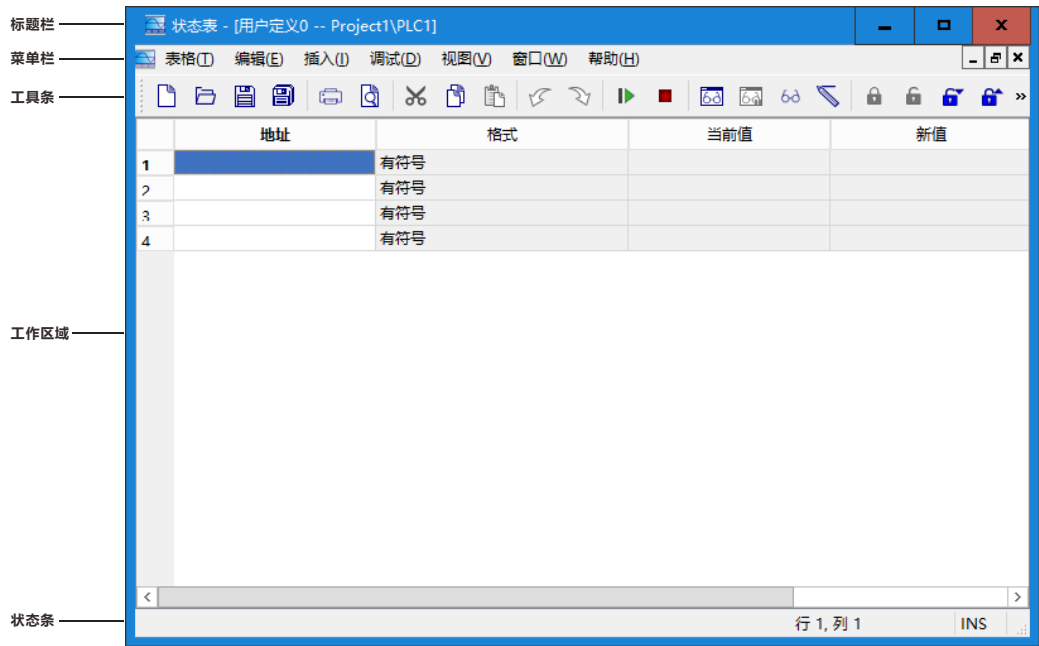
4.5 交叉引用查看器窗口

在此界面您可以工作区域右边双击选定某种方式查看当前程序是否已经使用和在哪里使用某一符号名或存储区赋值的情况。



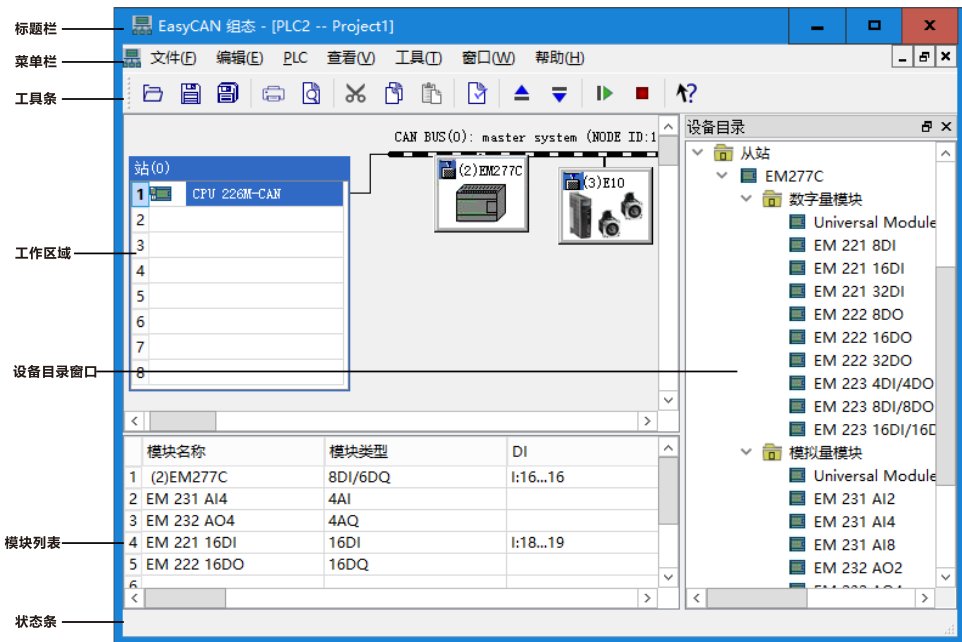
4.6 状态表编辑器窗口

此界面您可以为当前项目创建状态表及监控查看程序状态。



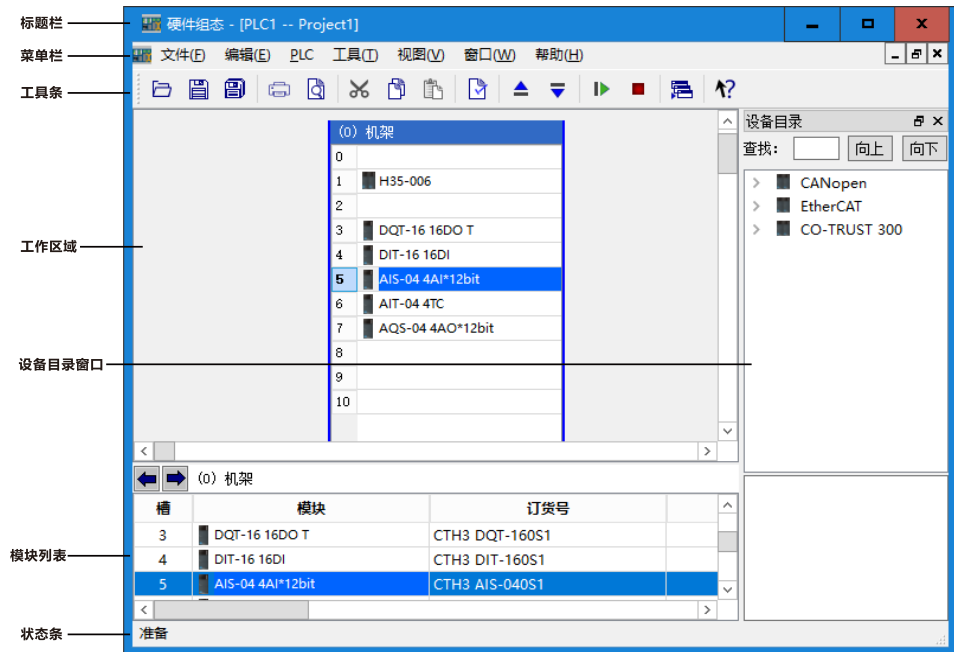
4.7 EasyCAN 组态窗口

EasyCAN 组态可用于组态 200 系列 PLC 的 EasyCAN 通讯，其主界面如下图所示：



4.8 硬件组态窗口

硬件组态可用于组态 CTH3 系列 PLC，其主界面如下图所示：

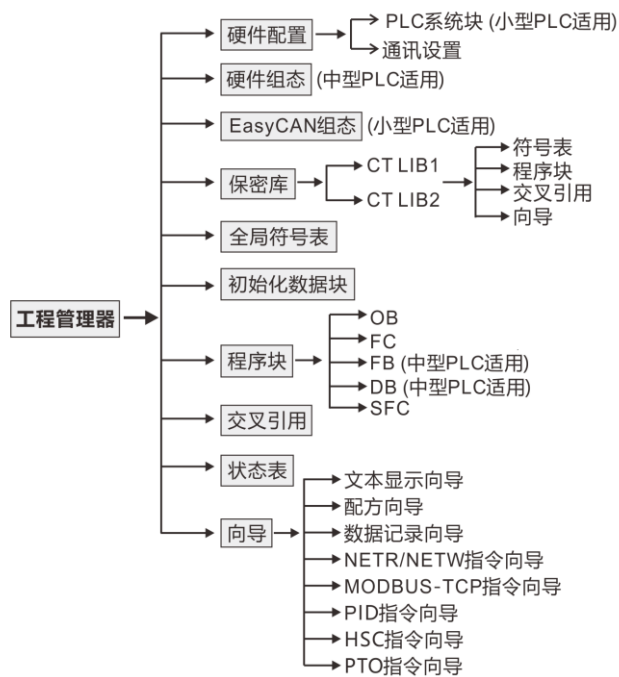


5 PLC 编程系统

本章概要

- 工程结构
- 符号表
- 程序块
- 交叉引用表
- 向导的使用
- 库的使用
- 编译功能

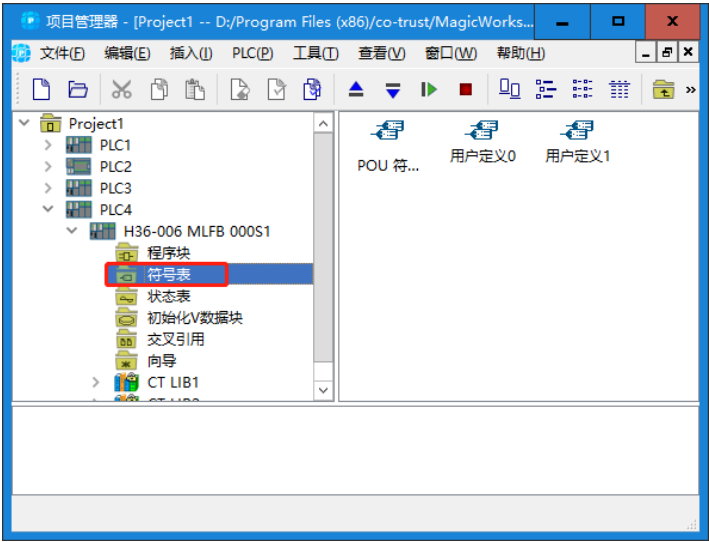
5.1 工程结构



5.2 符号表

符号分类	包含属性
全局符号	包含变量名称、变量地址、变量注释三个属性。
局部符号	包含变量地址、变量名称、变量类型、数据类型、变量注释五个属性，变量类型有五种类型（IN、IN_OUT、OUT、STATIC、TEMP）。

打开符号表：在资源管理器的符号表目录下双击任一符号表页图标或是在图标上单击鼠标右键选择菜单项“打开对象”即可启动符号表进入编辑器界面。



如果在符号表编辑器中有一个以上符号表，使用符号表编辑器窗口菜单中的排列窗口菜单项或是选择其中的已打开窗口列表菜单项切换符号表。

如果您已经打开一个空符号表，编辑行，在符号表中指定赋值。

符号表类型说明

POU 符号表，列出 POU 的 ID 与符号名，由软件自动生成，不允许用户编辑；库符号表，用户调用库指令时软件自动生成，不允许用户编辑，需要用户分配符号对应的地址，即分配库存储区，并不是所以库都会生成库符号表；向导生成的符号表；用户自定义符号表；

符号表中指定符号赋值

全局变量是为 PLC 的某个地址定义的有意义的符号名称，该符号名称在整个工程范围内都可被访问到，相当于使用该变量对应的元件，全局变量在全局变量表中定义。

全局变量声明包含变量名称、变量地址、变量注释三个属性。

全局变量的定义规则为：由 A~Z、a~z、0~9、下划线、汉字混合组合而成，变量名称不能以数字开始，也不能是单独的数字。名称区分大小写，长度不能超过 23 个字符，不能使用元件类型字母+数字作为程序和变量名称。名称中不能包含空格，不能使用和关键字相同的名称，保留的关键字包括：基本数据类型名称、指令名称、指令表语言中的运算符。

可按下列步骤为地址或常数值指定符号：

- 1) 打开符号表。
- 2) 在“符号名”列键入符号名（例如，Input1）。允许使用的最大符号长度为 23 个字符。使用 TAB、ENTER 或 ARROW 键确认，并移至下一个单元格。
- 3) 在“地址”列中键入地址或数值（例如，V 或 123）。
- 4) 键入注释（最多允许 79 个字符，也可为空）。



提示

- 符号定义过程中，符号的标记状态有几个：错误，重复（包括符号名重复和地址重复），包含。错误在状态列以  表示，重复（包括符号名重复和地址重复）在状态列以  表示，地址包含的两个符号还是有效符号，可以在程序中使用，标记只是起提示作用。
- 双引号用于在符号表中给 ASCII 常数字符串赋值时指定字符串的开始和结束，这就建立了一个符号 ASCII 字符串常数。与此相似，单引号用于起始和结束 ASCII 字符常数（只限字节、字、双字）以创建一个符号 ASCII 字符常数。

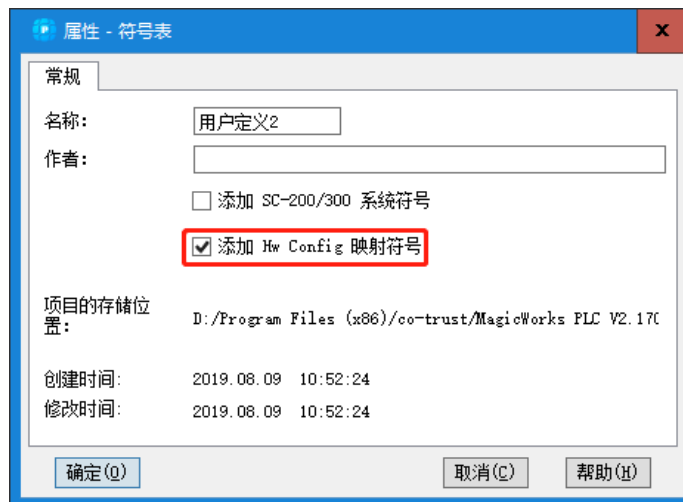
插入新对象

使用下列方法之一在符号表中插入新行：

- 在符号表中选中一行，然后在菜单项选择“插入”→“插入行”，即可在选中行上方插入新行。
- 用鼠标右键点击符号表中的一个单元格。在弹出的菜单项选择“插入行”。将在光标的当前位置上方插入新行。
- 欲在符号表底部插入新行，将光标放在最后一行的任意一个单元格中，键盘按向下箭头即可插入新行。

使用下列方法在符号表中插入硬件组态映射地址符号表：

在项目管理器界面，鼠标右键点击符号表，弹出的命令菜单中选择“插入新对象”，再选择“符号表”，打开以下界面，可修改名称等属性：



勾选“添加 Hw Config 映射符号”选项，即可插入硬件组态映射地址符号表。

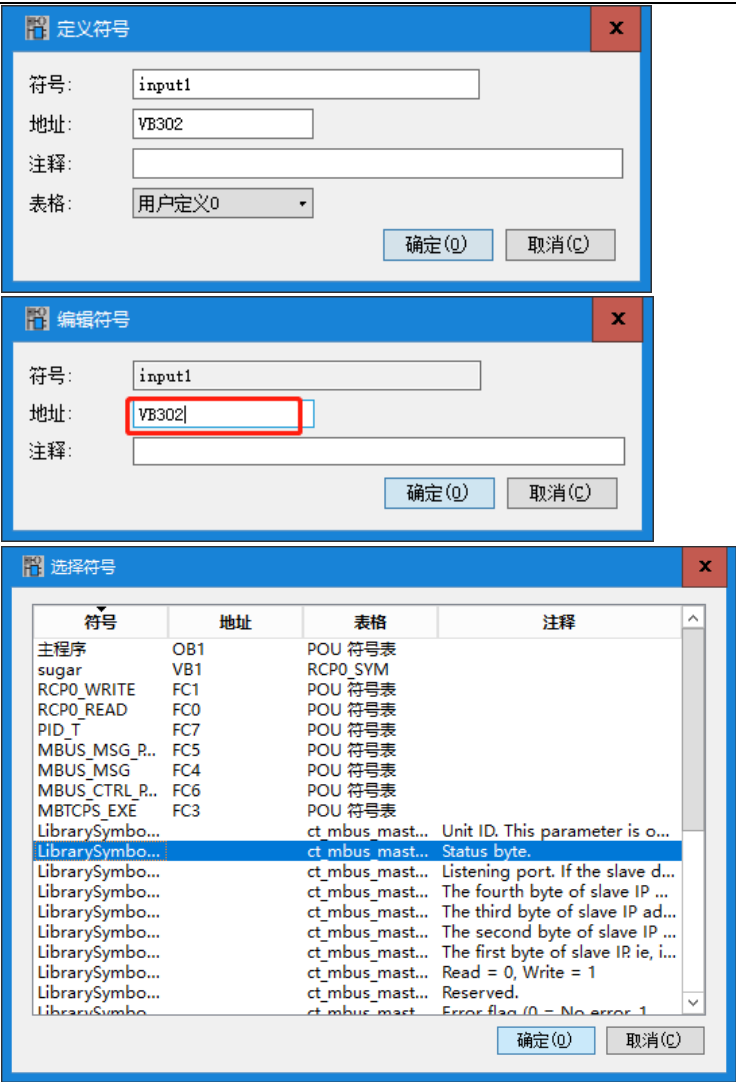
<注意> 只有 CTH300 系列 CPU 才有“添加 Hw Config 映射符号”选项。

使用定义、编辑或选择符号命令

定义、编辑或选择符号等命令，允许您在使用程序编辑器或符号表时，定义新符号、从列表上选择现有符号、或编辑符号属性。新的或修改后的赋值将被自动加入到符号表内。只有在“符号寻址”视图中工作时才能使用符号命令。

欲调用“符号”命令

1) 在程序编辑器界面用鼠标右键点击对某指令参数或对“符号表”地址单元来说不完全（或不正确）的符号名，然后从弹出菜单中选择“定义符号”，完成定义后即可选择“编辑符号”对符号地址进行修改，或“选择符号”编辑符号。



- 2) 定义新符号，编辑现有符号的属性，或由全部用户指定符号的概括中选择一个现有符号。在定义符号地址时，请在“地址”域内放一个地址。在定义符号常数时，请在“地址”域内放一个数值。
- 3) 点击“确定”确认作业，并关闭对话框，或点击“取消”取消选择。

按照列排序

用鼠标点击列表头，您可按照“名称”或“地址”列排序表格，排序时可按正向或逆向（字母）顺序排列，对于中文，按照音序排序。

新建符号表

新建好一个工程后，初始包含用户定义 0、POU 符号表两张表，在资源管理器中使用菜单功能可以插入符号表对象。使用符号表编辑器中窗口菜单进行表间的切换。

POU 符号表

如果您已经为项目的任何 POU 或其他组件（例如数据块、状态表或符号表）指定符号名，将在“POU”符号表页中列出符号名赋值。该页为只读，不能在此页编辑赋值。

使用间接寻址符号

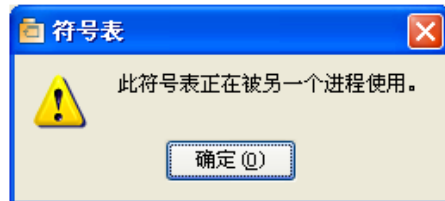
与绝对地址相同，您可在符号中使用间接寻址符号（&和*）。

存储区类型

您可为下列存储区类型建立符号名：I、Q、M、SM、AI、AQ、V、S、C、T、HC

互斥情况

一个工程有多个组件（如程序块、状态表、数据块、符号表）可进行符号编辑操作，当一处正在进行符号编辑操作尚未完成或保存，其它组件若在此刻编辑操作符号，将弹出符号表正在被使用的对话框，点击确定后将弹出符号操作编辑对话框，为只读形式。



5.3 程序块

控制程序的基本构成元素

CPU 的控制程序由以下程序组织单位（POU）类型组成：

- 主程序

程序的主体（OB1），是放置控制应用程序指令的位置。主程序中的指令按顺序执行，每次 CPU 扫描周期时执行一次。

- 子程序

子程序是指令的一个选用集，存放在单独的程序块中，仅主程序或另一个子程序调用时被执行。

- 中断程序

中断程序是指令的一个选用集，存放在单独的程序块中，仅在中断事件发生时被执行。

MagicWorks PLC 通过为每个 POU 在程序编辑器窗口中提供单独的标记组织程序。主程序 OB1 总是第一个标记，其后才是您建立的子程序。

如何终止 POU

由于程序被划分（每个 POU 占有一个单独的标记），OB1 或各种不同的子程序和中断程序的终止位置不成问题。编译程序根据实际情况用一个无条件 END、MEND、RET 或 RETI 终止每个 POU。如果您在程序中放入无条件 END、MEND、RET 或 RETI，编译程序会返回一则错误信息。

子程序

子程序在您希望反复执行一个功能时特别有用；您只需在子程序中写入一次逻辑，然后在主程序过程中每次需要时调用子程序即可，而无需在主程序中您希望执行功能的每个位置重写逻辑。使用子程序的优点如下：

- 代码总数减少。
- 主程序扫描时间减少。有条件地调用子程序，且在扫描过程中不被调用时不对子程序进行评估。
- 子程序容易移动。您可单独挑出一个功能，并将其复制至其他程序中，由此可避免重复操作。

<注释> V 内存的使用限制子程序的可移植性，因为一个程序的 V 内存地址赋值可能与另一个程

序中的赋值发生冲突。相反，将局部变量表用于所有地址赋值的子程序却很容易移动，因为不必担心会出现编址冲突。

中断程序

您可以写入中断程序，处理某些预定义的中断事件：中断程序不由主程序调用；在中断事件发生时由 PLC 操作系统调用。因为不可能预测系统何时会调用中断，所以最好不要允许中断程序写入可能在程序其他位置使用的内存。您可以使用局部变量表确保中断程序仅使用临时内存，而覆盖程序其他位置的数据。

程序组织单元

块	简述	规格
组织块（OB）	OB1 作为用户程序中的主程序，其它 OB 作为中断程序	小型 PLC 最多支持 255 个 OB
		中型 PLC 的 OB、FB、FC 的总数之和最多 1024 个，空间之和最大 128KBytes，单块最大空间 64KBytes，OB+FB+FC 大小≤CPU 程序空间大小
功能块（FB）	用户编写的包含经常使用的功能的子程序，有存储区	小型 PLC 不支持 FB
		中型 PLC 的 OB、FB、FC 的总数之和最多 1024 个，空间之和最大 128KBytes，单块最大空间 64KBytes，OB+FB+FC 大小≤CPU 程序空间大小
功能（FC）	用户编写的包含经常使用的功能的子程序，无存储区	小型 PLC 最多支持 255 个 FC
		中型 PLC 的 OB、FB、FC 的总数之和最多 1024 个，空间之和最大 128KBytes，单块最大空间 64KBytes，OB+FB+FC 大小≤CPU 程序空间大小
背景数据块（DI）	调用 FB 时用于传递参数的数据块	小型 PLC 不支持 DI
		中型 PLC 中 DI 和 DB 块总共最多 254 个，DI 和 DB 在 CPU 中的总大小≤118KBytes，单块最大空间 64KBytes；支持掉电保持功能（用户可选）；
共享数据块（DB）	存储用户数据的数据区域，供所有的块共享，即为 V 存储空间	小型 PLC 不支持 DB
		中型 PLC 中 DI 和 DB 块总共最多 254 个，DI 和 DB 在 CPU 中的总大小≤118KBytes，单块最大空间 64KBytes；支持掉电保持功能（用户可选）；
SFC	保密库 CT LIB1 和 CT LIB2 的接口。	2 个，若想使用 CT LIB1 或 CT LIB2，需在程序中调用此 POU。

<备注> 各个块的描述详见后续章节。

5.3.1 OB 组织块

组织块（OB）表示操作系统和用户程序之间的接口。组织块由操作系统调用，控制循环中断驱动的程序执行、PLC 启动特性和错误处理。可以对组织块进行编程来确定 CPU 特性。其中，程序的主体 OB1 是放置控制应用程序指令的位置，主程序中的指令按顺序执行，每次 CPU 扫描周期时执行一次。

组织块的优先级

组织块确定单个程序段执行的顺序（启动事件）。一个 OB 调用可以中断另一个 OB 的执行。

哪个 OB 允许中断另一个 OB 取决于其优先级。高优先级的 OB 可以中断优先级低的 OB。

中断事件

启动事件触发 OB 调用称为中断。您可以通过中断指令赋给 OB 中断事件号，然后根据组内优先级执行中断。中断优先级列表请参考附录 [E 中断事件优先级列表](#)。

例如，将中断事件号 10 赋给 OB3，STL 程序如下：

```
LD SM 0.0
ATCH OB3, 10
ENI
```

5.3.2 FB 功能块（CTH3 系列 PLC 适用）

FB 功能块是拥有自己的存储区（背景数据块）的块，每次调用功能块时需要提供各种类型的数据给功能块，功能块也要返回变量给调用它的块。这些数据以静态变量（STAT）的形式存放在指定的背景数据块（DI），临时变量 TEMP 存储在局部数据堆栈中。

调用 FB 时，必须指定 DI 的编号。在编译 FB 时，自动生成背景数据块中的数据。一个功能块可以有多个背景数据块，用于不同的被控对象。

可以在 FB 的变量声明表中给形参赋初值。如果调用块时没有提供实参，将使用上一次存储在 DI 中的参数。

局部符号表

数据类型

1、支持的参数类型

可以在变量表中编辑变量和块参数，不同的类型参数可使用不同的区域。显示区域的编号由正在编辑的对象类型决定。FB 有 5 个区域类型：

区域	声明	图标
IN	输入参数	
IN_OUT	输入输出参数	
OUT	输出参数	
STAT	静态数据（仅针对功能块（FB））	
TEMP	临时数据（L 存储区）	

2、参数支持的数据类型

输入变量（IN）数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL、STRINGCHAR；

输出变量（OUT）数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL；

输入输出变量（IN_OUT）数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL；

静态变量（STAT）数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL、FB<nr>、STRINGCHAR；

临时变量（TEMP）数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL；



提示

1) STRING 类型保持与 CTSC-200 PLC 的一致性，STRING 类型占用四个字节，存储的实际内容是一个地址，该地址指向字符串的首地址；

- 2) **STRINGCHAR** 类型默认长度为 256 个字节，第一个字节为字符串的长度，从第二个字节起，存储字符串中的每个字符；
- 3) 数据类型 **FB** 的变量：如果将用户程序结构化，以便 **FB** 能包含现存功能块的调用，那么可以在调用 **FB** 的变量声明表中包含作为数据类型 **FB** 的静态变量调用的 **FB**。允许在一个背景数据块（多重实例）中嵌套变量和集中实例数据，最大允许嵌套多重实例的层数为 7 层，嵌套只支持单向性（比如 **FB5** 组态 **FB4**，**FB4** 组态 **FB3**，**FB3** 组态 **FB2**，**FB2** 组态 **FB1**，此时 **FB3** 允许组态 **FB1** 和 **FB2**，而不能够组态 **FB3/FB4/FB5**）。

3、支持参数、临时变量、静态变量的数目

FC 支持 **IN/OUT/IN_OUT** 的总数最多 62 个，**FB** 支持 **IN/OUT/IN_OUT** 的总数最多 60 个，临时变量占用的空间和参数占用空间的总大小不超过 128 个字节（临时）。

FB 支持静态变量的数目，受背景数据块最大空间的限制，即 **FB** 组态的总大小，不能超过背景数据块的最大空间。

- 临时变量与静态变量

临时变量（**TEMP**）：相关块执行完毕后被清除，临时存储在 **L stack** 中。
静态变量（**STAT**）：相关块执行完毕后仍然保持，存储在 **DB** 中；只能在 **FB** 中使用。

- 列属性

从左到右依次表示内容为：名称、变量类型、数据类型、地址、初始化值、注释；

列	说明
名称	字母、汉字、数字字符和下划线组成；字母开头；名称不能以下划线结尾；禁止出现两个连续的下划线；最大长度限制为 23 个字符；禁止使用关键字，变量名不能重复。
数据类型	BOOL 、 BYTE 、 INT 、 WORD 、 DINT 、 DWORD 、 REAL 、 STRINGCHAR 和 FB 。 可以在数据类型列的下拉列表中选择， String 变量占用 256 个字节（首字节表示长度，其他 255 字节存储字符）。
地址	是相对于分配的 DB 区首地址的偏移量，根据数据类型自动分配。
初始化值	对内存初始化，语法自动检查，错误的值以红色字体显示，变量声明后使用默认的初始值。 BOOL 型默认初始值是 OFF ；整型默认初始值是 0 ；浮点型默认初始值是 0.0 ； StringChar 型默认初始值是""（空）。
注释	最多支持 79 个字符

将初始值分配给形式参数

可以在 **FB** 的声明部分将初始值分配给形式参数。这些值写入到与 **FB** 有关的实例 **DB** 中。

如果在调用声明中没有将实际参数分配给形式参数，则将使用保存在实例 **DB** 中的值。这些值也可以是在 **FB** 变量声明表中输入的初始值。除了临时变量之外，**IN/OUT/IN_OUT/STAT** 都允许分配初始值。

删除/插入块的参数

当一个块已在程序中被调用之后，再增加或删除块的参数，必须更新其他块中的该块的调用。

否则，由于在调用时该块新增的参数没有被分配实际参数，则 CPU 会 STOP 或者块的功能不能实现。比如，新插入一个参数，它必须在所有块调用中提供实际参数。

当块的声明表由于插入或删除形式参数被修改之后保存时，将弹出警告信息，提示可能出现的问题。

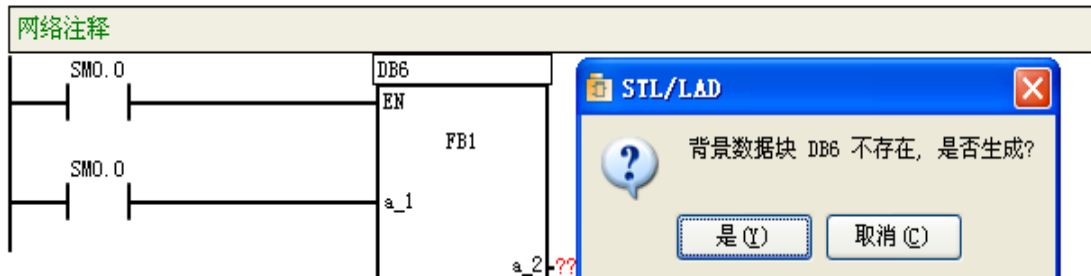
程序编辑区

生成背景数据块

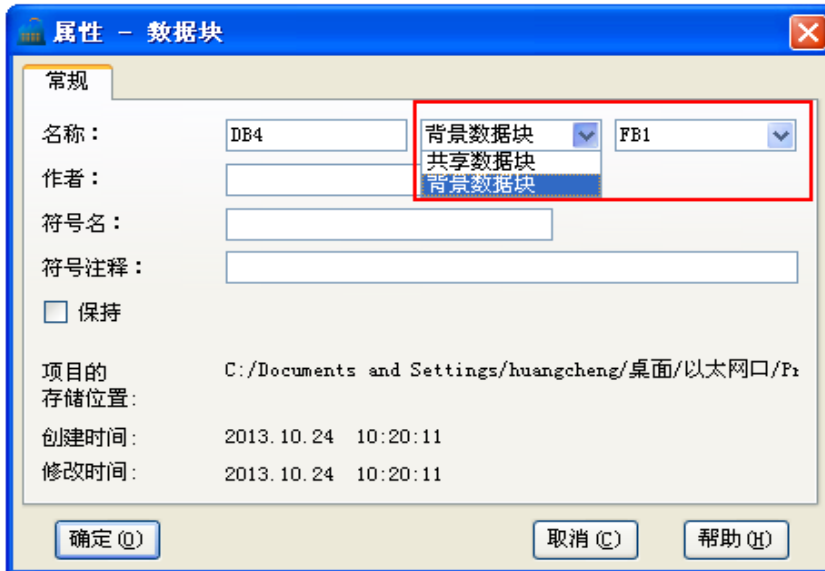
生成一个新的背景数据块有以下两种方法：

- 1、在调用 FB 时，为 FB 指定一个背景 DB 后，如果该数据块不存在则弹出以下提示信息，单击“是”按钮可自动生成一个新的背景数据块。

网络 1 网络标题



- 2、创建一个新的 DB 时，选择其类型为“背景数据块”，并指定为某个 FB 的数据块。



输出窗口

支持单独块的编译功能，在用户打开的当前程序编辑区中可以看到编译的最新消息，在程序编辑区最下方的输出窗口显示编译的错误等信息，该区可以根据用户的配置显示或隐藏。

程序元素表

程序元素标签中新增两个元素：功能块（FB blocks）、多重实例（Multiple instance）

预组态的功能块（FB blocks）

展开已有的功能块生成的指令，供程序的 CALL 指令进行调用；

已声明的多重实例（Multiple instance）

通过声明一个具有功能块数据类型的静态变量，创建一个多重实例。只在程序元素目录中包括已经声明的多重实例。多重实例的符号改变取决于是否带参数以及带多少个参数。

在变量中声明多重实例

下列规则均适用于多重实例的声明：

- 1) 必须将背景数据块分配给在其中对多重实例进行声明的功能块；
- 2) 只能将多重实例声明为静态变量（说明类型为 STAT）；

功能块的使用

功能块是一种带数据块（DB 块）的块，功能块中的参数值会保存到与它关联的背景数据块中，临时变量保存到本地数据堆栈中。

FB 执行完成后，它的参数值会一直保存到实例 DB 中，不会丢失（临时变量会丢失）。

例如：下图为 FB1 的参数：

符号	变量类型	数据类型	地址	Initial Value
a_1	IN	BOOL	0.0	TRUE
	IN			
	IN_OUT			
a_2	OUT	BOOL	0.1	TRUE
	OUT			
a_3	STAT	BOOL	0.2	TRUE
	STAT			
	TEMP			

下图为 FB1 关联的实例 DB（背景 DB）：

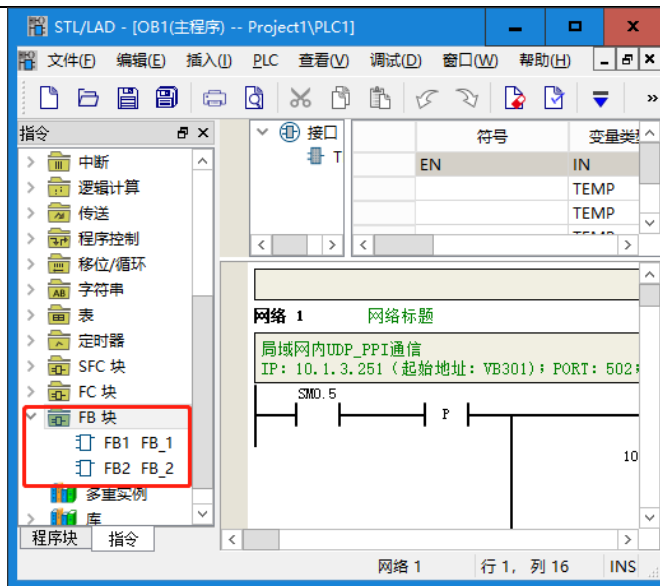
	地址	声明	名称	类型	初始值
1	+0.0	IN	a_1	BOOL	TRUE
2	+0.1	OUT	a_2	BOOL	TRUE
3	+0.2	STAT	a_3	BOOL	TRUE

可以看到实例 DB 与 FB1 的参数是一一对应的，FB1 执行完后，FB1 参数的值会保存到实例 DB 中对应的变量中。

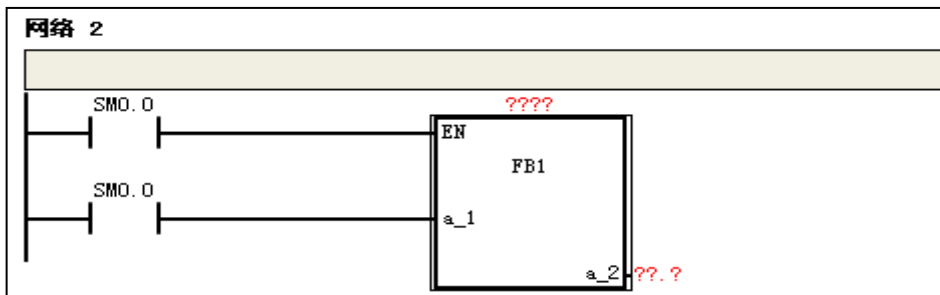
功能块(FB)可以当成一种特殊的功能(FC)：与 FC 相比，FB 多了一个 DB 类参数。

调用功能块

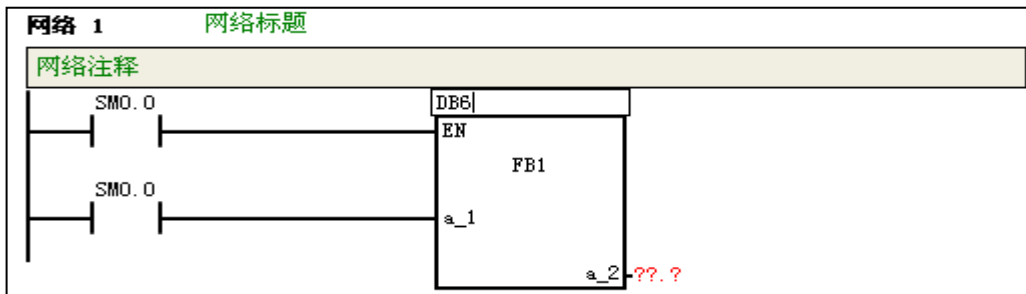
- 1) 打开“程序块”界面，在左侧的指令树中找到 FB 块节点，如图：



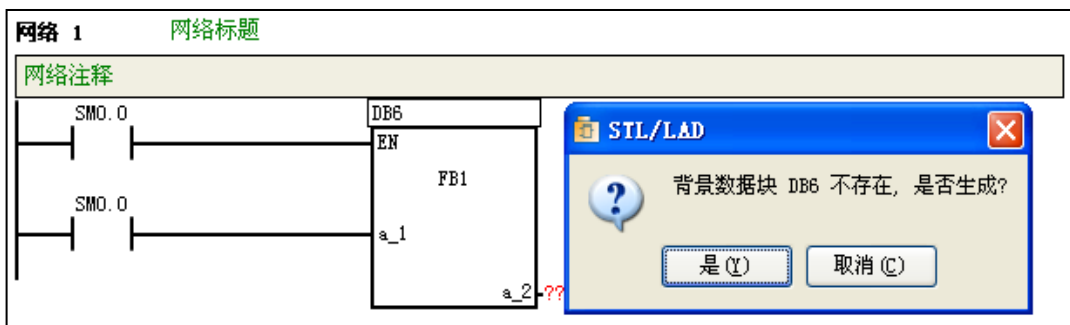
2) 双击或拖拽相应的 FB 至程序编辑器中:



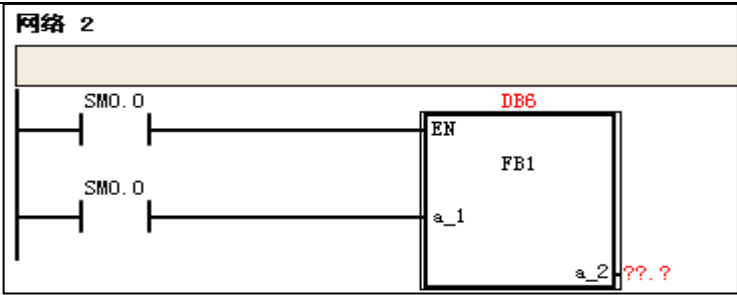
3) 然后输入参数, FB 需要一个 DB 类型的参数, 在指令的顶部输入, 如下图所示:



若输入的 DB 块参数不存在则会提示是否生成:

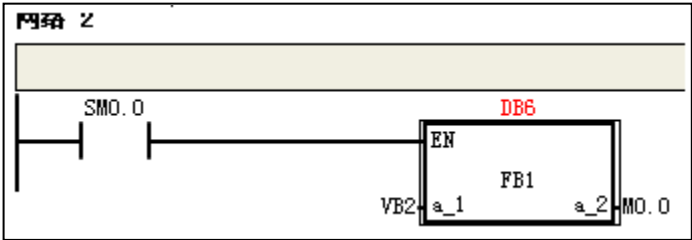


4) 在调用 FB 后, 如果修改了被调用 FB 的输入输出参数列表, 则所有调用此 FB 的地方, 此 FB 的参数都会变成红色, 表示 FB 的接口改变了:

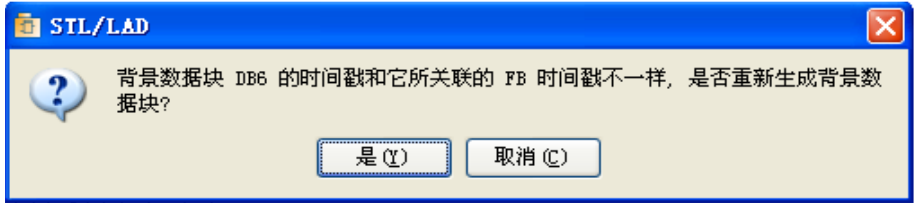


此时需要把此调用删除，并重新调用。

在调用 FB 后，如果修改了被调用 FB 的 STAT 参数列表，则所有调用此 FB 的地方，此 FB 的 DB 参数会变红，表示 FB 的接口改变了：



此时只需用鼠标点击变红的 DB 参数，再点击其它的地方，则会弹出提示：



选择“是”，即可重新生成背景数据块。

调用多重实例

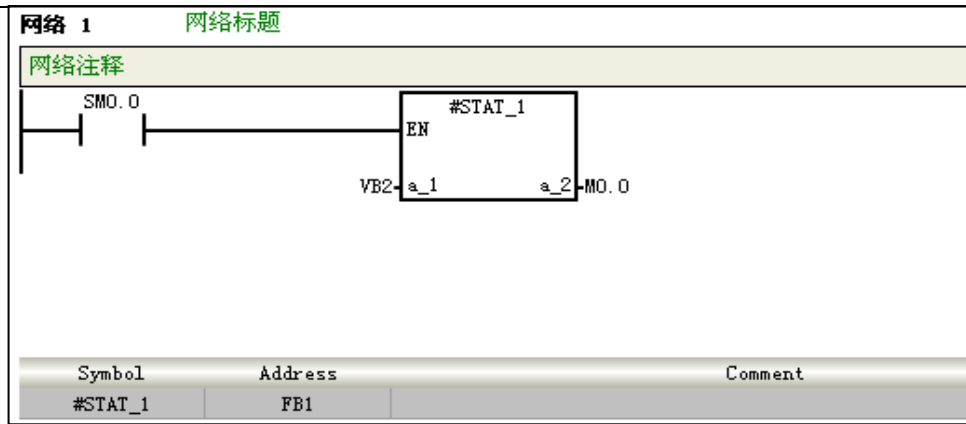
多重实例即 FB 以另一个 FB 作为它的 STAT 类型的参数，以 FB1 作为 FB 的 STAT 参数，如图：

a_1	IN	INT	0.0	0
	IN			
	IN_OUT			
	OUT			
STAT_1	STAT	FB 1	2.0	
	STAT			
	TEMP			

此时在左侧的指令树中多重实例节点就可以看到 STAT_1：



双击或拖动 STAT_1 到程序编辑器即可：



<备注> 多重实例没有 DB 类型的参数（背景数据块），它使用宿主的背景数据块（DB），宿主指调用这个多重实例的块。

5.3.3 FC 功能

功能（FC）属于个人编程的块。功能是一种不带存储器的逻辑块。属于 FC 的临时变量保存在本地数据堆栈中。执行 FC 时，该数据将丢失。为永久保存该数据，功能也可使用共享数据块。由于 FC 本身没有存储器，因此必须始终给它指定实际参数。无法给 FC 的本地数据分配初始值。

应用

FC 包含由另一个逻辑块调用该 FC 时，始终执行的程序段。可使用下列功能：

- 将功能值返回调用块（例如，算术功能）
- 执行技术功能（例如，具有位逻辑操作的单个控制功能）

将实际参数分配给形式参数

形式参数是实际参数的哑元。调用该功能时，实际参数将替换形式参数。必须始终将实际参数分配给 FC 的形式参数。FC 所使用的输入、输出以及输入/输出参数作为指针保存到调用 FC 的逻辑块的实际参数中。

FC 和 FB 输出参数之间的重要区别

在功能块（FB）中，访问参数时，使用背景数据块中的实际参数副本。如果调用 FB 时，没有传送输入参数或没有写访问输出参数，那么将使用原先保存在背景数据块（背景数据块=FB 存储器）中的值。

但功能（FC）没有存储器，因此，与 FB 相反，必须将形式参数分配给这些 FC。通过地址（指针跨过区域边界指向目标）访问 FC 参数。当数据区（数据块）地址或调用块的局部变量用作实际参数时，实际参数的副本将临时保存到用于传送参数的调用块的本地数据区中。

<警告> 此时，如果没有数据写入到 FC 中的 OUTPUT 参数，那么该块可能输出随机值。如果没有将保留给副本的调用块的本地数据区分配给 OUTPUT 参数，因此没有数据写入到该区域。由此，该区保持不变，例如，由于缺省情况下本地数据没有自动设置为“0”，所以将输出在该位置保存的随机值。

因此，请遵守下列几点：

- 如有可能，请初始化 OUTPUT 参数。
- 始终确保将数据写入到 OUTPUT 参数中，而与块中的程序路径无关。

<注意> 虽然 FB 的 OUTPUT 参数或 FC 和 FB 的 OUTPUT 参数不会输出随机值（即使没有数据写入到该参数中，也保持原输出值或作为输出值的输入值），仍然应该遵守上述各点注意事项，避免意外处理“原”值。

5.3.4 DB 数据块（CTH3 系列 PLC 适用）

数据块是中型 PLC 支持的一种存放数据的块。与逻辑块相反，数据块不包含 PLC 指令。它们用来存储用户数据，即数据块包含用户程序使用的变量数据。其中共享数据块可用来存储可由所有其它块访问的用户数据。

数据块与 CTSC-200 PLC 的 V 内存不同，需要用户创建并在数据块中声明需要的数据类型后，才能够使用。从这个意义上讲，V 内存可以理解作为一种特殊的数据块，在 PLC 中已经默认分配好空间的数据块。

数据块的编辑方式不同于 CTSC-200 PLC 中的 V 内存，使用独立的编辑器 datablockeditor.exe 来打开、编辑、修改、保存数据块中的数据。创建数据块的方式与创建逻辑块 OB/FC/FB 一致，在工作区域的空白处点击右键→“插入新对象”→“数据块”来弹出数据块的属性，再经过用户配置后，可以创建对应的数据块。

数据块包括共享数据块和背景数据块，它们分别有以下功能：

- 共享数据块的新建、编辑、下载、监控功能；
- 背景数据块的新建、下载、监控功能；

数据块的基本信息

可以在数据块（DB）中存储数值以便为机器或设备所访问。与逻辑块 OB/FB/FC 相比，数据块只包含变量声明部分。这就表示此处与代码段和编程段无关。

数据块支持声明视图和数据视图，使用菜单命令“视图”→“声明视图”或“数据视图”，可在这两种视图之间切换。

声明视图

- 1) 新建、编辑、修改共享数据块的数据结构；
- 2) 查看相关功能块（FB）的数据块的数据结构。

在声明视图中，不能修改与功能块相关联的数据块的结构。若要修改它们，必须首先修改相关的 FB，然后创建一个新的数据块。只有共享的数据块，才能改变声明视图。

声明视图中列属性的详细解释：

列	解释
地址	当输入完声明时，显示自动分配给变量的地址。
声明	仅为实例数据块显示该列。它显示功能块变量声明中的变量是如何声明的： 输入参数(IN) 输出参数(OUT) 输入/输出参数(IN_OUT) 静态数据(STAT)
名称	在此输入必须为各个变量分配的符号名称。
类型	输入希望分配给变量的数据类型(BOOL、INT、WORD、等等)。变量可以有

	基本数据类型、复杂数据类型或用户自定义数据类型。
初始值	如果不希望使用所输入数据类型的默认值，那么可以在此输入初始值。所有的值都必须与数据类型兼容。当首次保存块时，如果没有明确定义变量的实际值，那么，初始值将作为当前值使用。 <注意> 初始值不能下载给 CPU。
注释	在该域中输入一条注释，有助于对变量编写文档。

数据视图

数据视图显示数据块中所有变量的当前值。只能在数据视图中改变这些值。对于所有共享数据块，该视图中的表格表示均相同。对于背景数据块，将显示一个附加的“声明”列。

数据视图列属性的详述：

列	解释
地址	自动为变量分配的地址。
声明	仅为实例数据块显示该列。它显示功能块变量声明中的变量是如何声明的： 输入参数(IN) 输出参数(OUT) 输入/输出参数(IN_OUT) 静态数据(STAT)
名称	在变量声明中为变量分配的符号名称。在数据视图中不能编辑该域。
类型	显示为变量定义的数据类型。对于共享数据块，这里将只列出基本数据类型，因为对于具有复杂或用户自定义数据类型的变量，其元素将在数据视图中逐个列出。对于实例数据块，还会显示参数类型，对于具有复杂或用户自定义数据类型的输入/输出参数(IN_OUT)，指针将指向“实际值”列中的数据类型。
变量初始值	如果不希望使用指定数据类型的默认值时，可以为变量输入初始值。当首次保存数据块时，如果没有明确定义变量的实际值，将使用初始值作为当前值。 <注意> 与实际值不同，初始值不能下载给 CPU。
实际值	打开数据块时的变量值，或上一次修改、保存过的变量值。如果在监控状态下则为 CPU 中变量的值。
注释	所输入的用来为变量编写文档的注释。在数据视图中不能编辑该域。

共享数据块和背景数据块之间的差别

共享数据块：共享数据块是全局共享的，可以随意修改数据块中的变量。

背景数据块：背景数据块和特定的 FB 关联，不能随意修改里面的变量，只能通过 FB 生成，新建一个数据块时可以指定它与一个 FB 块关联。背景数据块中的变量是与它关联的 FB 的输入输出及静态参数。

共享数据块支持的数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL、STRINGCHAR；

背景数据块支持的数据类型：BOOL、BYTE、INT、WORD、DINT、DWORD、REAL、STRINGCHAR、FB<nr>；

数据块编号的取值范围和大小

数据块编号的取值范围：1~254；

每个数据块的最大空间为：64K（65536 个字节）；

数据块中允许创建的最大行数：组态数据块的大小，只要未超过单个数据块的最大空间 64K，

则允许用户一直创建。

数据块的监控

声明视图下的监控

声明视图不支持监控模式，在声明视图下启动监控模式时，会弹出提示“当前模式不支持监控功能，是否跳转到数据视图进行监控”，选择“是”，则跳转到数据视图进行监控；在用户取消监控后，会跳转回声明视图模式。

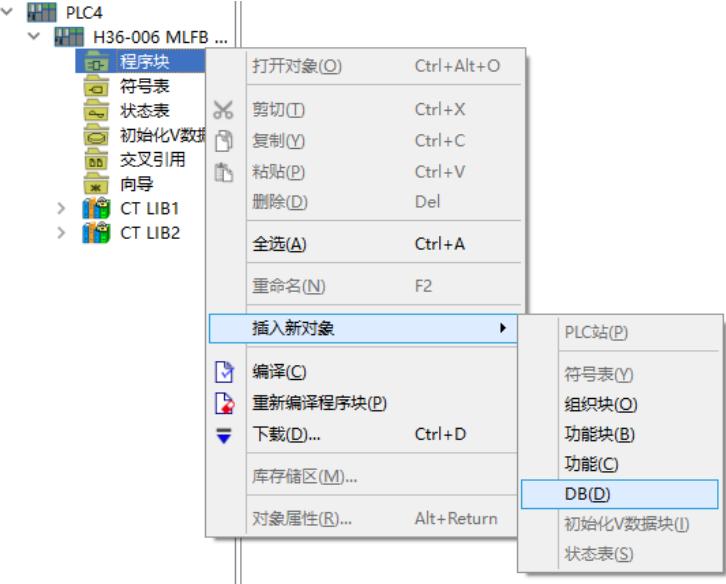
数据视图下的监控

数据视图启动监控模式，会在“实际值”列显示 CPU 中对应的数据值，而且只有实际值列可以修改；支持 STRINGCHAR 数据类型的监控。

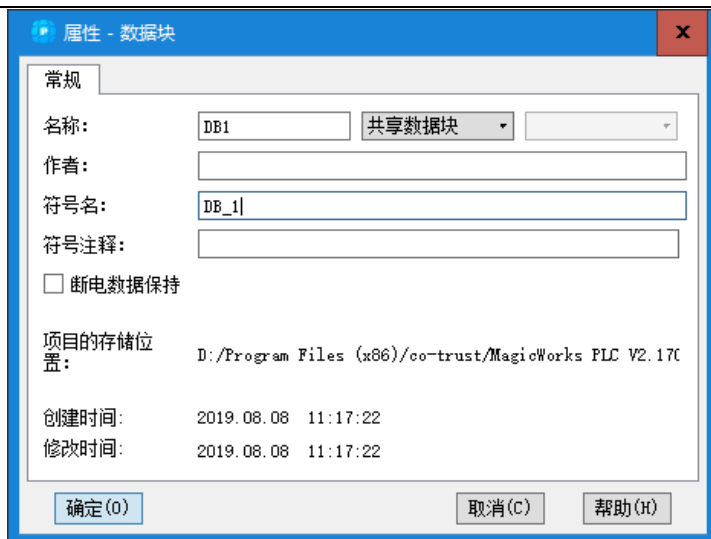
DB 块的使用

新建数据块

- 在 CTH300 系列 CPU 站点下选择程序块，右键选择“插入新对象”→“DB”插入新的 DB 块，如下图所示：

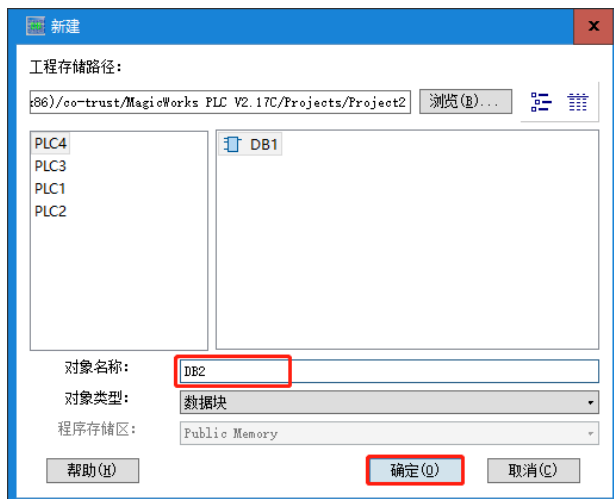


选择“DB”后会弹出如下对话框：

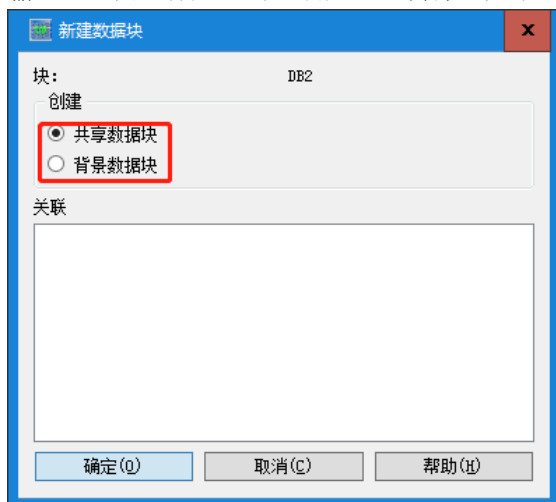


在此可以设置新数据块的名称、符号名等属性，配置完后点击“确定”即可完成数据块的创建。

- 在程序块界面双击建好的 DB 数据块，进入 DB 参数设置界面，在该界面菜单项选择“文件”→“新建”，会弹出如下对话框：

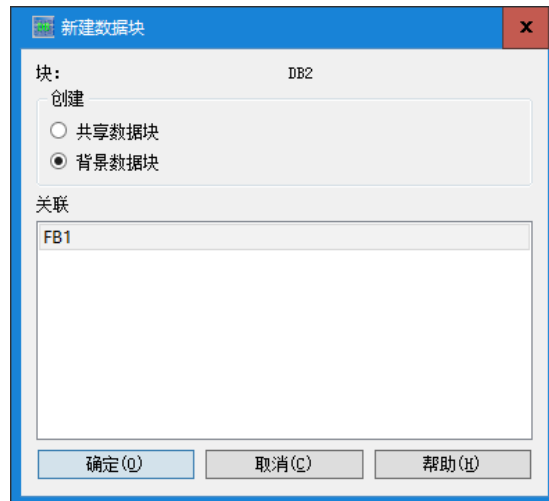


输入 DB 块名称后点击“确定”，会弹出如下对话框：



此处可以选择新建共享数据块还是背景数据块。如果选择了背景数据块下面的列表会列出可以

关联的 FB，如下图：

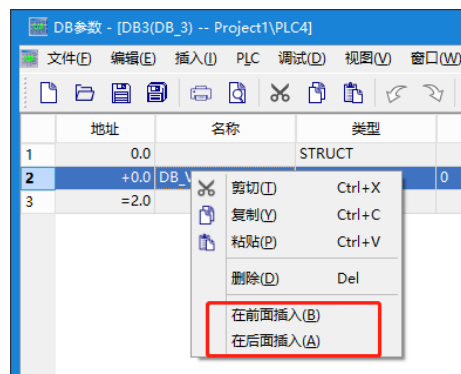


选择关联的 FB 后，点击“确定”完成数据块的创建。

插入行

只有共享数据块才可以插入行，具体操作如下：

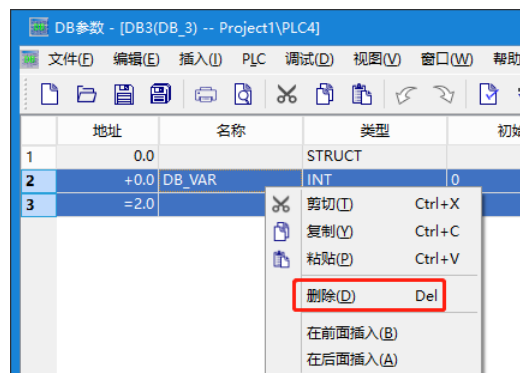
- 选择菜单项“视图”→“声明视图”即切换到声明视图；
- 选中一个单元格：单击右键选择“在前面插入”/“在后面插入”即可在当前单元格前面或后面插入一行数据。



删除行

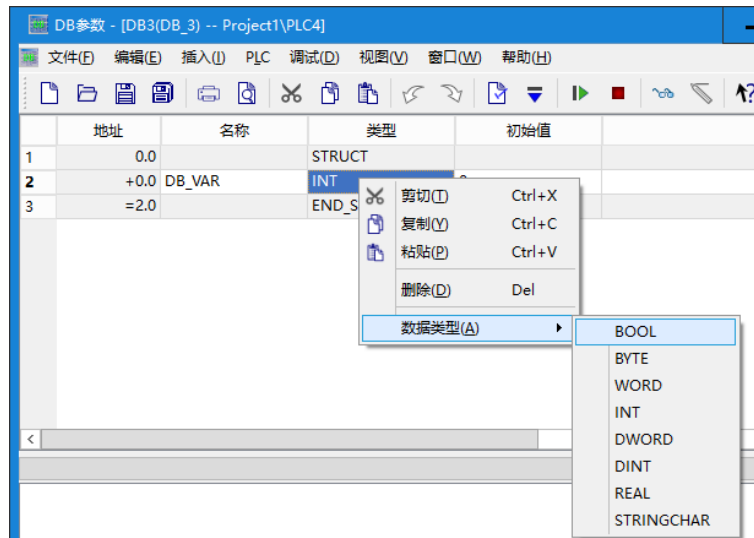
只有共享数据块才可以删除行，具体操作如下：

- 选择菜单项“视图”→“声明视图”即可切换到声明视图；
- 选中单行或多行数据，单击右键选择“删除”。



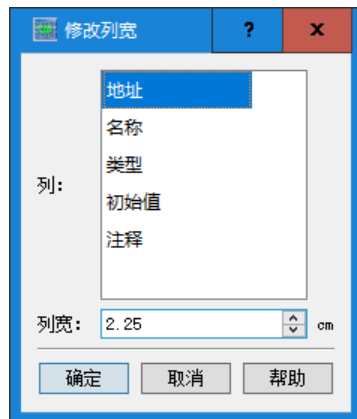
修改变量名称：双击“名称”列，即可对变量名称进行编辑。

修改变量类型：右键点击“类型”列→“数据类型”，如下图：



修改列宽

在 DB 参数页面选择菜单项“视图”→“列宽”弹出以下窗口，可进行表格各项内容列宽设置：



编译：点击工具栏的编译按钮或选择菜单项“PLC”→“编译”则只编译改动过的程序块；若选择“PLC”→“重新编译程序块”则将编译所有程序块，包括未改动程序块。

下载：点击工具栏的下载按钮或选择菜单项“PLC”→“下载”

写入

- 只有在监控状态下才可以写入；
- 只有在修改了实际值时才可以写入，否则写入菜单为灰色；
- 修改了实际值后，被修改的行监控会暂停，只有写入后才会继续监控；

写入过程：

- 1) 进入监控模式
- 2) 修改实际值
- 3) 点击工具栏写入按钮或选择菜单项“PLC”→“写入”

监控

- 1) 选择菜单项“视图”→“数据视图”即切换到数据视图
- 2) 工具栏点击监控按钮或在菜单项选择“调试”→“监控”即可切换到监控画面

使用 DB 块变量

- 1、通过直接地址使用

DB [number].DB [X,B,W,D][byteoffset].[bitoffset]

[number]为数据块编号，[X,B,W,D]，表示数据类型，[byteoffset]表示字节偏移，[bitoffset]表示位偏移。例如：

位变量：DB1.DBX0.0

字节变量：DB1.DBB0

字变量：DB1.DBW0

双字变量：DB1.DBD0

2、通过符号使用

[DBsymbol].[varsymbol]

[DBsymbol]表示数据块的符号（新建数据块时可以输入），[varsymbol]表示数据块中变量名（数据块中的名称列）。例如：

ABC.var_1

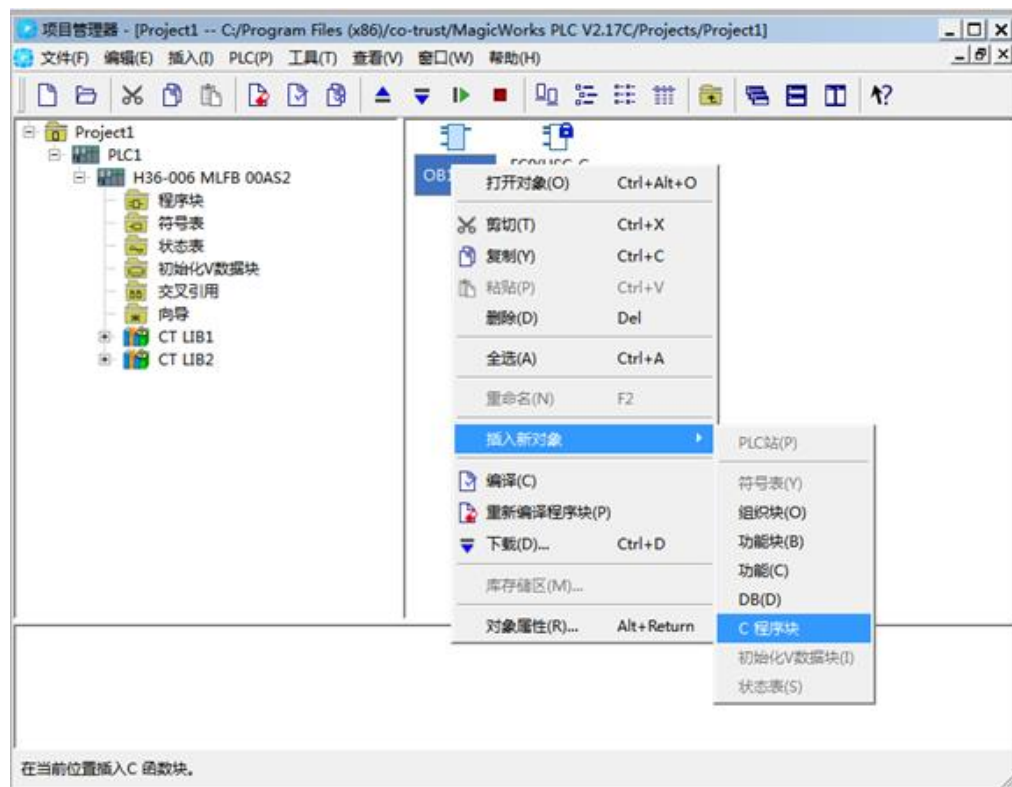
5.3.5 C 程序块（CTH300 系列 PLC 适用）

MagicWorks PLC 编程软件 V2.18 及以上版本支持 C 语言编程功能（CTH300 系列 PLC 适用）。利用 C 语言编写功能块，增强了程序的保密性，同时，C 语言丰富的运算函数可实现各种调用功能。节省了内部空间，也提高了编程效率。

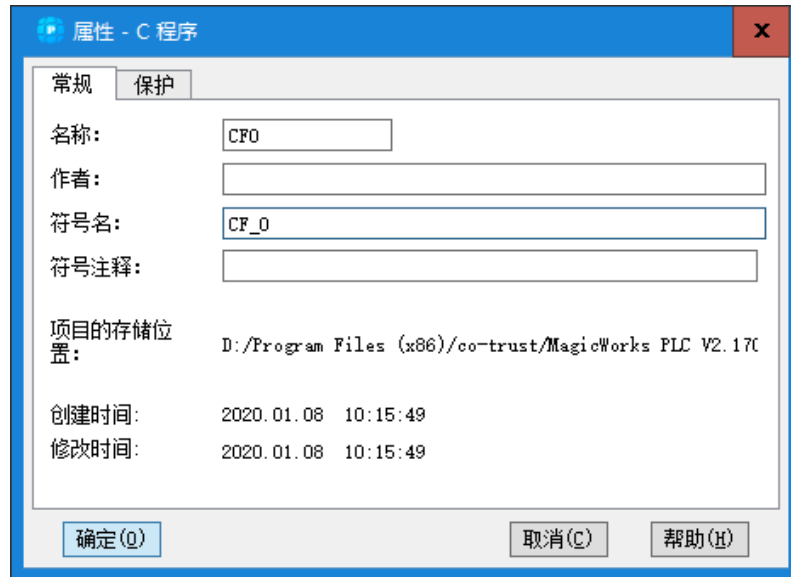
注意：使用前，需将 ccompiler（C 编译器软件）解压到编程软件所在文件根目录下。相关文件可于合信官网下载，下载地址：<http://www.co-trust.com/Download/index.html>

插入 C 程序块

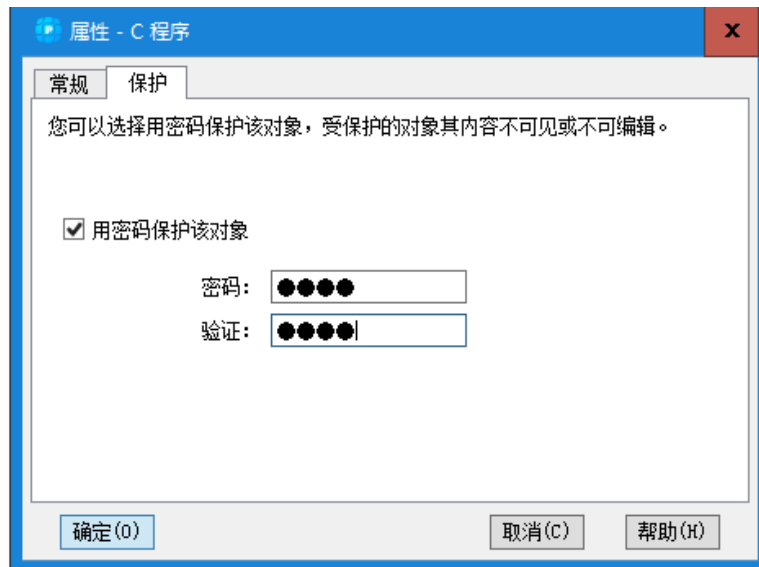
在 CTH300 系列 CPU 站点下，展开 PLC 站点，选中“程序块”并点击鼠标右键，在弹出的选项里依次选择“插入新对象”→“C 程序块”。



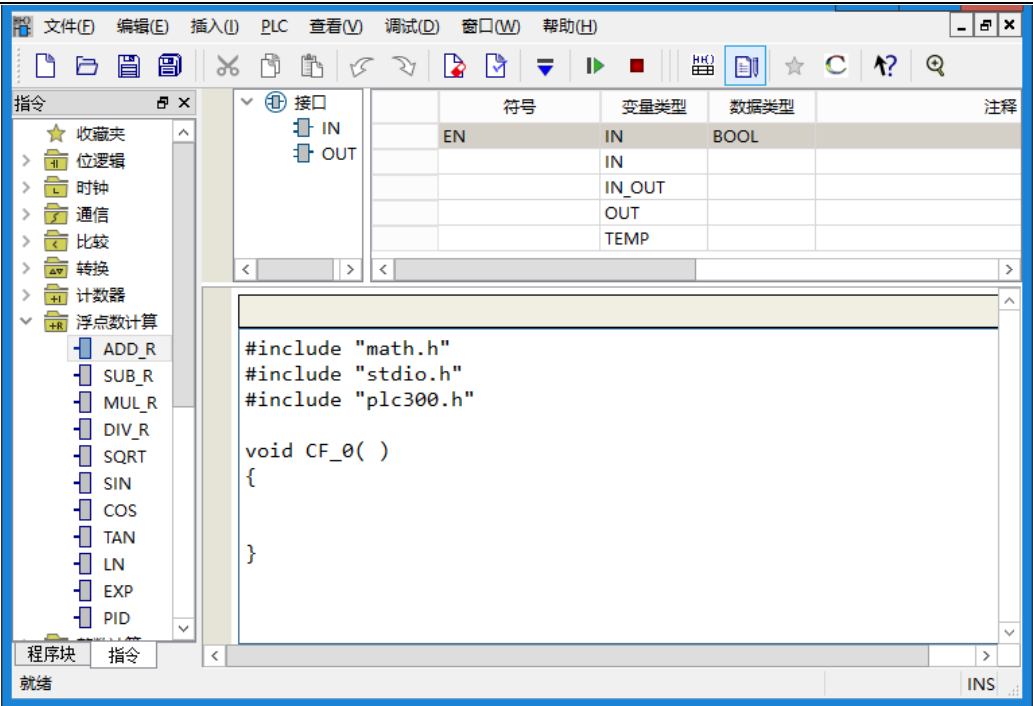
在弹出的窗口中可以选择修改“名称”和“符号名”，修改后点击“确定”完成新建。



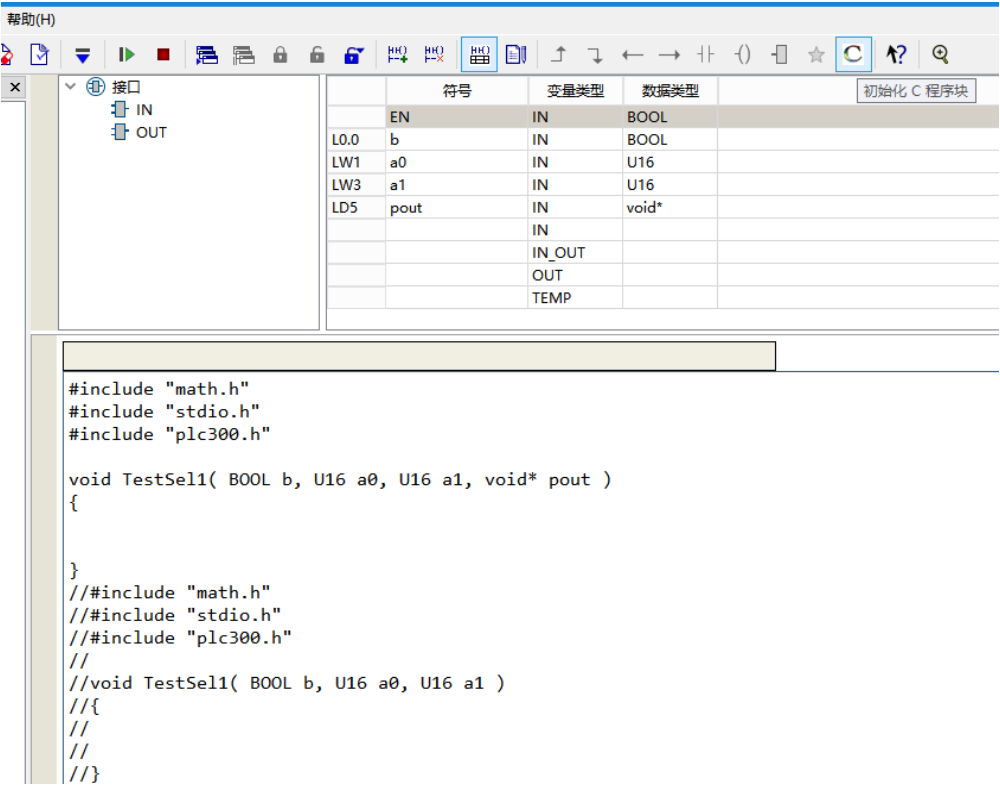
也可在保护选项下设置密码以保护代码，如下图所示。



完成以上操作后，可在程序块界面出现 C 程序块，双击打开，如下图所示。



从参数符号表定义参数，参数实际只变量类型只能取 IN，支持个数为 16 个，C 函数无返回值，返回值只能通过指针传回。函数名和参数形式需保持一致，勿随便改动，否则可能会导致出错。参数定义好后，点击初始化 C 程序块，重新生成函数模板。



参数通过指针在 C 系列和 STL 进行传递时，由于 C 语言系统使用的是小端模式，而 STL 系统使用的是大端模式，所以需要调用大小端转换接口。转换接口程序如下：

```

#include "math.h"
#include "stdio.h"
#include "plc300.h"
|
void TestSel0( BOOL b, U16 a0, U16 a1, void* pout )
{
    if(b)
    {
        setU16(pout, a1);
    }
    else
    {
        setU16(pout, a0);
    }
}

```

用户可以在 C 语言中选择使用自定义的全局变量，或自定义函数。全局变量会在程序下载过程中和 PLC 上电时被初始化。

```

#include "math.h"
#include "stdio.h"
#include "plc300.h"

int g = 33;

void fnInc(int *p)
{
    *p = (*p + 1);
}

void G_INC( void* pout )
{|
    fnInc(&g);
    setU32(pout, g);
}

```

用户可以选择使用编译器自带的标准库函数，比如数学函数 `sqrt`。

```

#include "math.h"
#include "stdio.h"
#include "plc300.h"

void TMAHT(FP32x, FP32y, viod* pout)
{
    FP32 r;
    r=sqrt(x*x=y*y);
    setFP32(pout, r)
}

```

合信 PLC 提供一个由合信自己开发的库 `libplc300.a`，以使用户能够调用底层函数和 STL PLC 系列进行交互。

 AM335x.lds	2019-11-04 15:31	LDS 文件	3 KB
 libplc300.a	2019-11-26 14:56	360压缩	65 KB

PLC 能够使用的函数，存放于头文件编程软件安装目录\lccompile\include\plc300.h。下表列出了一些默认的基本函数，用户可以在 C 函数中直接调用。

函数名	输入	说明
Set S16	Ps16: 从 STL 系统传过来的指针; S16: C 语言所使用的 S16 数据	将一个 S16 数据从 C 语言写回 STL 系统
Set U32	从 STL 系统传过来的指针	将一个 U32 数据从 C 语言写回 STL 系统
Set S32	从 STL 系统传过来的指针	将一个 S32 数据从 C 语言写回 STL 系统
Set FP32	从 STL 系统传过来的指针	将一个 FP32 数据从 C 语言写回 STL 系统

5.3.6 SFC

SFC 为系统功能，是由 COTRUST 提供的具有特定功能的 FC，不允许用户编辑和查看。不同类型的 CPU 可能支持不同的功能，目前 COTRUST 发布的 SFC 有：

SFC1: ct_lib1 的调用接口；

SFC2: ct_lib2 的调用接口；

支持 SFC 的 COTRUST CPU 类型见下表：

CPU 类型	支持的 SFC
CPU124/ CPU126/ CPU224/ CPU224I/ CPUH224X/ CPUH224/ CPU H226L/ CPU H226M	SFC1
CPU226/ CPU226I/ CPU224XP CNI/ CPU H226XL/ CPU H226XM/ CPU H226XC/ CPU H228XL/ CPU H228ML/ CPU H228IL/ CPU H228SL/ H35-006/ H36-006	SFC1、SFC2

5.3.7 局部变量表

以下两种情况下可以使用局部变量：

- 您希望建立不引用绝对地址或全局符号的可移动子程序。
- 您希望使用临时变量（说明为 TEMP 的局部变量）进行计算，以便释放 PLC 内存。

如果以上说明对您的情形不适用，您无须使用局部变量；您可以在符号表中定义符号数值，将所有的符号数值定义为全局变量。

您可以使用程序编辑器的局部变量表指定对个别子程序或中断程序唯一的变量。

局部变量可用作传递至子程序并可用于增加子程序的移动性或重新使用子程序的参数。

程序中的每个 POU 都有自己的局部变量表，配备 64 个字节的 L 内存。这些局部变量表允许您定义具有范围限制的变量：局部变量只在建立该变量的 POU 中才有效。相反，在每个 POU 中均有效的全局符号只能在符号表中定义。当您为全局符号和局部变量使用相同的符号名时（例如 INPUT1），定义局部变量的 POU 中的局部定义优先，全局定义用于其他 POU。

在局部变量表中赋值时，您指定说明类型（TEMP、IN、IN_OUT 或 OUT）和数据类型，但不

指定内存地址；程序编辑器自动在 L 内存区中为所有的局部变量指定内存位置。

局部变量表符号地址分配会将一符号名与存储有关数据值的 L 内存地址关联。局部变量表不支持向符号名直接赋值的符号常数（这在符号表中是许可的）。

<提示> PLC 不会将局部变量数据值初始为零。您必须在程序逻辑中初始化您使用的局部变量。

局部变量的变量类型

对局部变量赋值指定的类型取决于在其中赋值的 POU。主程序（OB1）、中断程序和子程序可使用临时（TEMP）变量。只有在执行块时，临时变量才可用，块执行完成后，临时变量可被覆盖。子程序也可使用调用参数（IN、IN_OUT、OUT）。

区域	说明
IN	调用 POU 提供的输入参数
IN_OUT	数值由调用 POU 提供的参数，由子程序修改，然后返回调用 POU
OUT	返回调用 POU 的输出参数
STAT	静态数据（仅针对功能块（FB））
TEMP	临时保存在局部数据堆栈中的临时变量。一旦 POU 完全执行，临时变量数值则无法再用。在两次 POU 执行之间，临时变量不保持其数值

局部变量的数据类型

将局部变量作为子程序参数传递时，在该子程序局部变量表中指定的数据类型必须与调用 POU 中数值的数据类型相匹配。

举例：

您从 OB1 调用 SBR0，将称为 INPUT1 的全局符号用作子程序的输入参数。

在 SBR0 的局部变量表中，您已经将一个称为 FIRST 的局部变量定义为输入参数。

当 OB1 调用 SBR0 时，INPUT1 数值被传递至 FIRST。

INPUT1 和 FIRST 的数据类型必须匹配。

如果 INPUT1 是实数，FIRST 也是实数，则数据类型匹配。如果 INPUT1 是实数，但 FIRST 是整数，则数据类型不匹配，只有纠正了这一错误，程序才能编译。

检视/隐藏局部变量表

如果您将水平分裂条拉至程序编辑器窗口的顶部，则局部变量表不再显示，但仍旧存在。将分裂条下拉即可再次显示局部变量表。

如何在局部变量表中赋值



提示

- 在程序中使用局部变量之前，在局部变量表中赋值最为有效。在程序中使用符号名时，程序编辑器首先检查适当 POU 的局部变量表，然后检查符号表。如果符号名在这两处均未定义，程序编辑器则将其视为未定义的全局符号；此类符号用绿色波浪状下划线标明。程序编辑器不会自动重新读取局部变量表并对您的程序逻辑做出修正。如果您后来进行了定义该符号名的数据类型分配（在局部变量表中），则须以手动方式在该符号名前插入一个#符号，例如：#UndefinedLocalVar（在程序逻辑中）。
- 每个子程序调用的输入/出参数的最大限制是 16，如果您尝试下载的程序超过此限制，则超出的临时变量将不会被下载至 PLC 中。

如何输入第一个局部变量赋值

欲在局部变量表中赋值，请遵循以下步骤：

- 1) 确保正确的 POU 在程序编辑器窗口中显示，如有必要，点击所需的 POU 标记。（因为每个 POU 都有自己的局部变量表，您需要确保对正确的 POU 赋值）
- 2) 如果局部变量表处于隐藏状态，下拉水平分裂条，显示局部变量表。
- 3) 为需要定义的变量选择具有正确说明类型的行，在“名称”域中为该变量键入一个名称。（如果您在 OB1 或中断程序中赋值，局部变量表只包含 TEMP 变量。如果您在子程序中赋值，局部变量表包含 IN、IN_OUT、OUT 和 TEMP 变量。）您不需要在局部变量表中的变量名前加#号。#号只用在程序代码中的局部变量前。



提示

- 局部变量表可包含数字，下划线和中英文字符，不可以数字开头，不能包含扩展字符。
- 局部变量表变量名被下载并存储于 CPU 内存中，使用较长的变量名可能减少用于存储程序的内存空间。

- 4) 在“数据类型”域中点击鼠标指针，并使用列表框为局部变量选择适当的数据类型。

<注释>

- 当您为局部变量指定为子程序的参数时，您必须保证为局部变量指定的数据类型不与子程序调用中使用的操作数发生冲突（请参阅上文中的数据类型检查）。
- 为“名称”和“数据类型”域提供数值后，程序编辑器会自动为局部变量指定 L 内存地址。

如何在局部变量表中输入附加赋值

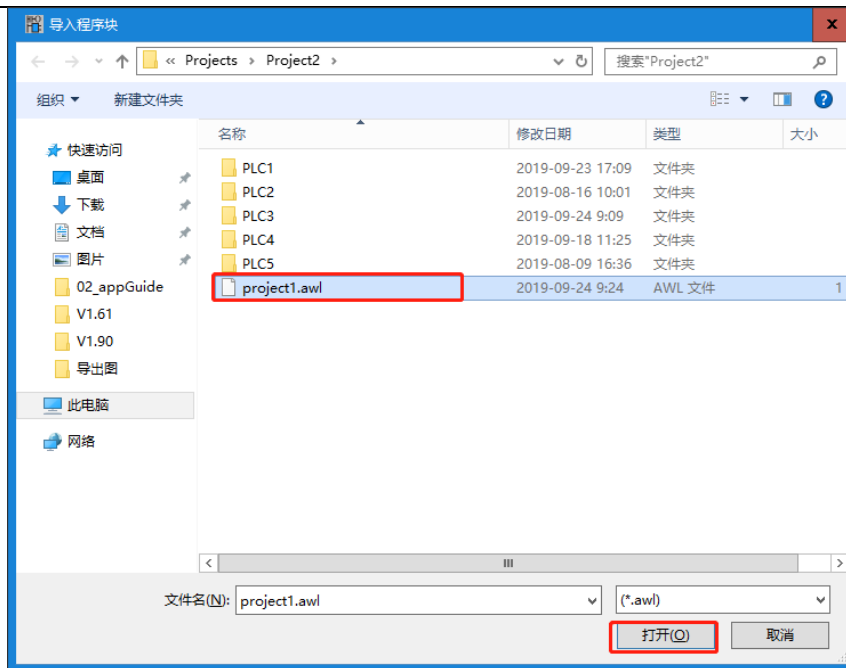
对于 OB1 和中断程序，局部变量表显示一组已被预先定义为 TEMP 变量的行。在 OB1 或中断程序中，只能使用这种说明类型。欲在表中增加更多行，只需点击最后一行中的一个单元格，然后使用 ENTER 键移过该行并向下移动，即自动生成一个新行。

对于子程序，局部变量表显示按照以下顺序预先定义说明类型的一组行：IN、IN_OUT、OUT 和 TEMP。您不能改变该顺序。局部变量在该表中的顺序必须符合当您为子程序调用指令进行操作数赋值时对应的操作数顺序。如果您希望增加附加局部变量，选中一个现有行，单击鼠标右键选择“插入行”即可在表中插入新行。

5.3.8 程序块的导入导出

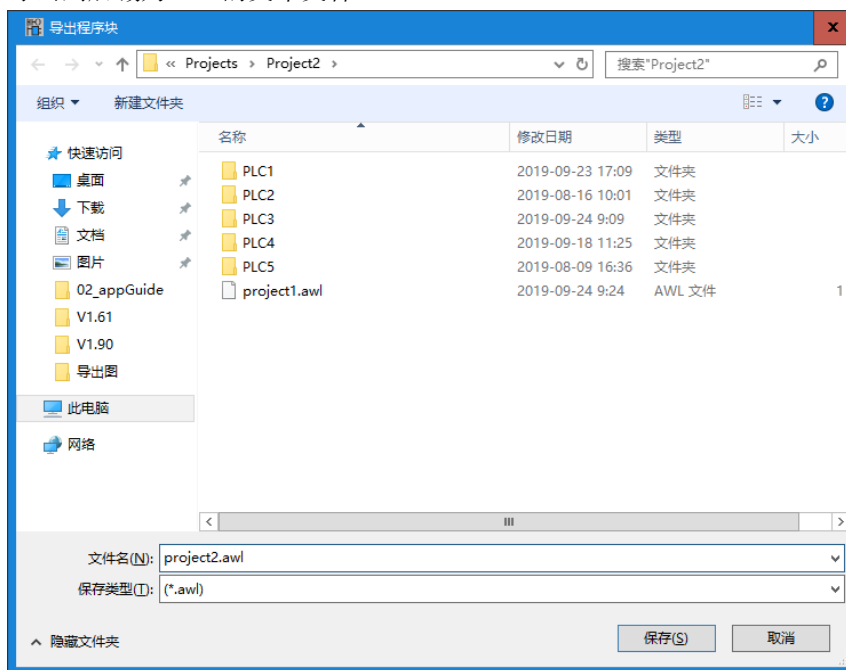
导入程序块文件

选择菜单项“文件”→“导入”打开如下对话框，即可将 ASCII 文件格式的程序块文件（后缀为.awl）内容导入到当前激活的程序块文件。



导出程序块文件

选择菜单项“文件”→“导出”打开如下对话框，将当前程序块文件的内容以 ASCII 文件格式导出到后缀为.awl 的文本文件。

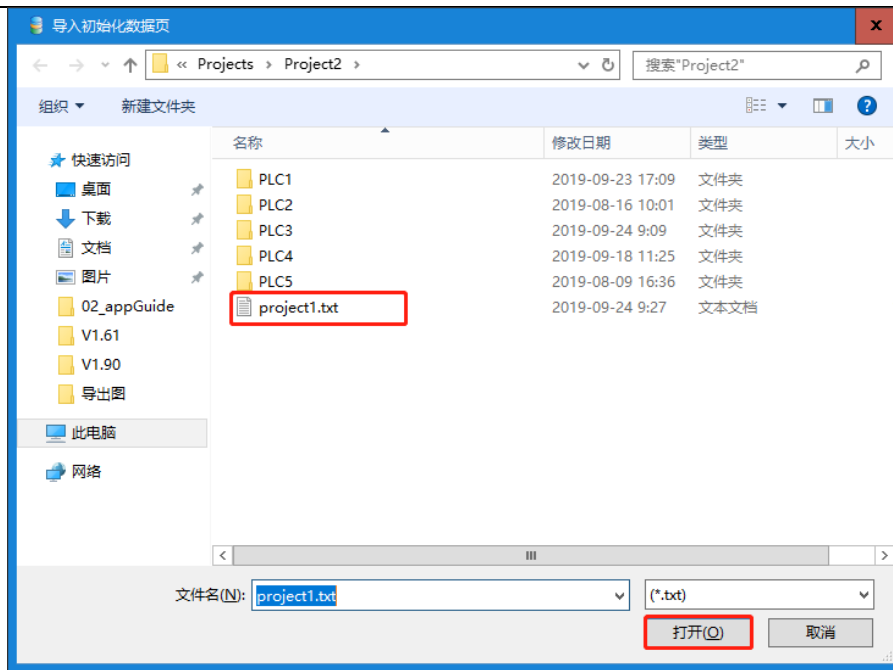


查看 ASCII 文本文件格式的一种简单方法是导出一个现有程序，然后使用 Windows Notepad（记事本）编辑器检查文件内容。

如果导入 ASCII 文本文件（例如，test.awl）因语法错误失败，程序编辑器会以对话框的形式弹出提示。如果文件通过语法测试，则被添加至项目。因为 LAD 和 FBD 编辑器不能显示行注释，因此导入文件中的注释会丢失。

导入初始化数据页文件

将 ASCII 文件格式的初始化数据页文件（后缀.txt）内容导入到当前激活的初始化数据页面窗口。



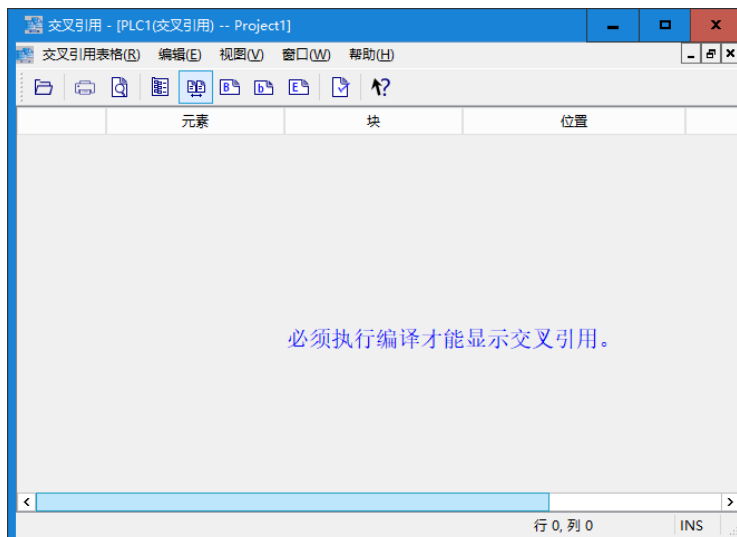
5.4 交叉引用表

使用以下方式打开交叉引用表：

在项目管理器的项目树中选中交叉引用结点，即可看到右边的工作窗口中出现调用结构、交叉引用、字节引用、位使用、沿使用五个图标，双击其中一个图标或者在图标上单击鼠标右键选择“打开”菜单项即可启动交叉引用查看器并显示相应表页。

交叉引用查看器是多文档窗口，使用交叉引用查看器窗口菜单中的排列窗口菜单项或是选择其中的已打开窗口列表菜单项切换表页。

<注意> 必须执行程序编译后才能查看“交叉引用”表。



交叉引用表

当您想了解程序中使用了哪些符号名或存储区赋值时，可使用“交叉引用”表。“交叉引用”表列出在程序中使用的全部操作数，并指出它所在的位置（POU、网络或行位置）以及每次使用时的指令上下文。

名称	功能描述
元素	程序中已经使用的某一符号名或绝对地址
块	符号或地址所在的程序块
位置	符号或地址所在某程序块的网络与行描述的位置
关联	显示该符号或地址相关联的指令，相当于显示某具体位置行的程序语句中以该符号或地址为操作数的指令。

LAD 交叉引用示例：

交叉引用 - [PLC1(交叉引用) -- Project1]				
交叉引用表格(R) 编辑(E) 视图(V) 窗口(W) 帮助(H)				
	元素	块	位置	关联
1	SM0.0	MAIN (OB1)	网络2	- -
2	SM0.0	MAIN (OB1)	网络9	- -
3	SM0.1	MAIN (OB1)	网络1	- -
4	T32	MAIN (OB1)	网络1	-(R)
5	T32	MAIN (OB1)	网络2	TON
6	T32	MAIN (OB1)	网络3	- -
7	T32	MAIN (OB1)	网络3	-(R)
8	VB4	MAIN (OB1)	网络3	INC_B
9	VB4	MAIN (OB1)	网络3	INC_B
10	VB5	MAIN (OB1)	网络3	INC_B
11	VB5	MAIN (OB1)	网络3	INC_B
12	VW0	MAIN (OB1)	网络3	FILL_N
13	VW4	MAIN (OB1)	网络3	FILL_N
行 0, 列 0 INS				

STL 交叉引用示例：

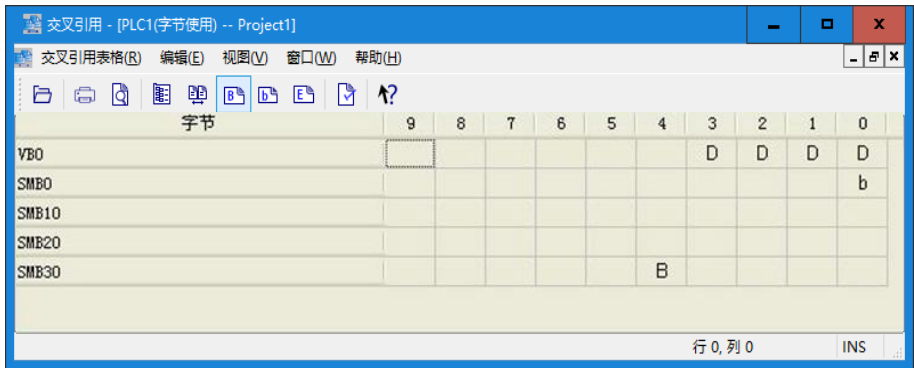
交叉引用 - [PLC1(交叉引用) -- Project1]				
交叉引用表格(R) 编辑(E) 视图(V) 窗口(W) 帮助(H)				
	元素	块	位置	关联
1	Always_On:SM0.0	FC0	网络1, 行1	LD
2	Always_On:SM0.0	FC1	网络1, 行1	LD
3	Always_On:SM0.0	OB2	网络1, 行1	LD
4	FC0	OB1	网络2, 行2	CALL
5	FC1	OB1	网络3, 行2	CALL
6	First_Scan_On:SM0.1	OB1	网络1, 行1	LD
7	OB2	OB1	网络1, 行4	ATCH
8	Q1.0	FC0	网络1, 行2	S
9	Q1.0	FC1	网络1, 行2	R
10	Time_0_Intrvl:SMB34	OB1	网络1, 行3	MOVB
11	VD0	OB1	网络1, 行2	MOVB
12	VD0	OB1	网络2, 行1	LDD=
13	VD0	OB1	网络3, 行1	LDD=
14	VD0	OB2	网络1, 行2	INCD
行 0, 列 0 INS				

字节使用表

“字节使用”表允许您查看程序中使用了哪些字节以及在哪里使用的，您可借此检查程序中重复赋值错误。

- b: 表示已经使用存储区的一个位。
- B: 表示已经使用存储区的一个字节。
- W: 表示已经使用存储区的一个字。
- D: 表示已经指定存储区的一个双字。
- X: 用于定时器和计数器类型。

字节使用示例：
如下图所示，程序中使用了地址 VD0, SM0.X 和 SMB34:



可通过检查字节使用表，识别重复的赋值。在适当的程序赋值中，字节/字/双字的标记应该是连续的，中间不会有不同的标记。如 DDwb 表示有重复赋值，而 DDDD 则没问题。

如下面的程序中存在重复赋值：

网络 1 网络标题

网络注释

LD

MOVB

MOVW

MOVD

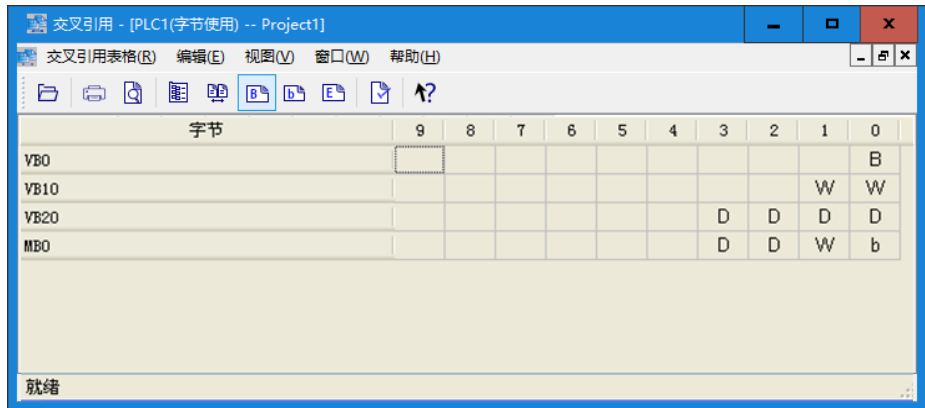
MO. 0

MB0, VB0

MW0, VW10

MD0, VD20

编译后就会出现下面的字节使用表：



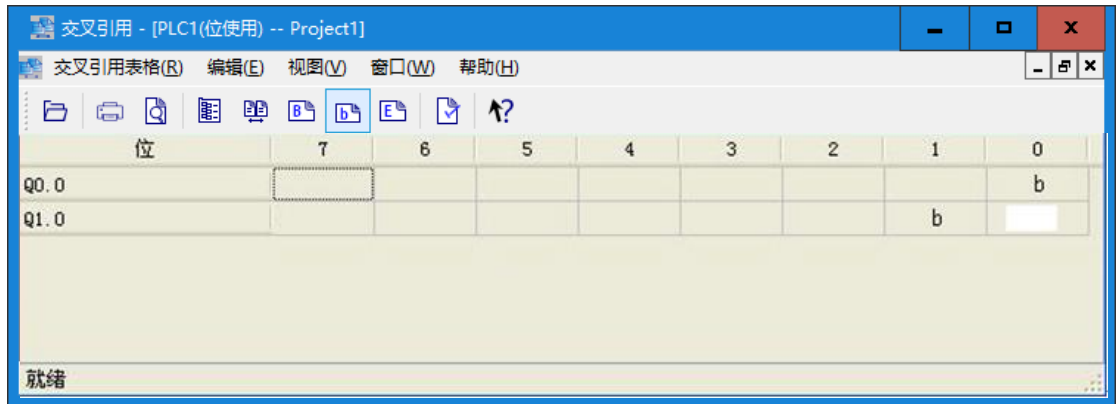
位使用表

位使用表允许您查看程序中已经使用了哪些位及在哪里使用的，您可借此检查程序中重复赋值错误。

- b: 表示已经使用存储区的一个位。
- B: 表示已经使用存储区的一个字节。

- W: 表示已经使用存储区的一个字。
- D: 表示已经指定存储区的一个双字。
- X 用于定时器和计数器类型。

位使用示例:
如下图所示, 程序中使用了地址 Q0.0 和 Q1.1



可通过检查位使用表, 识别重复的赋值。在适当的程序赋值中, 字节/字/双字的标记应该是连续的, 中间不会有位值标记。如 **BBBBBBb** 表示有重复赋值, 而 **BBBBBBBb** 没问题。

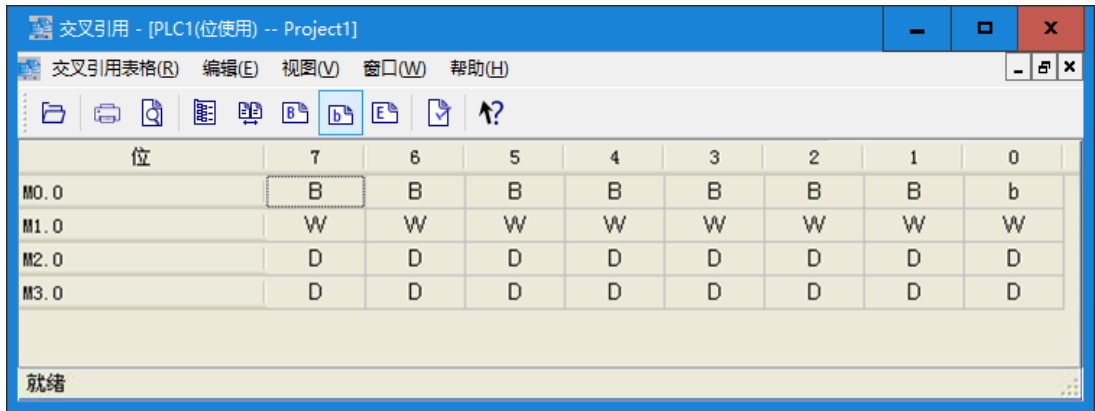
例如: 下面的程序中存在重复赋值:

网络 1 网络标题

网络注释

LD M0.0
MOV MB0, VB0
MOV MW0, VW10
MOV MD0, VD20

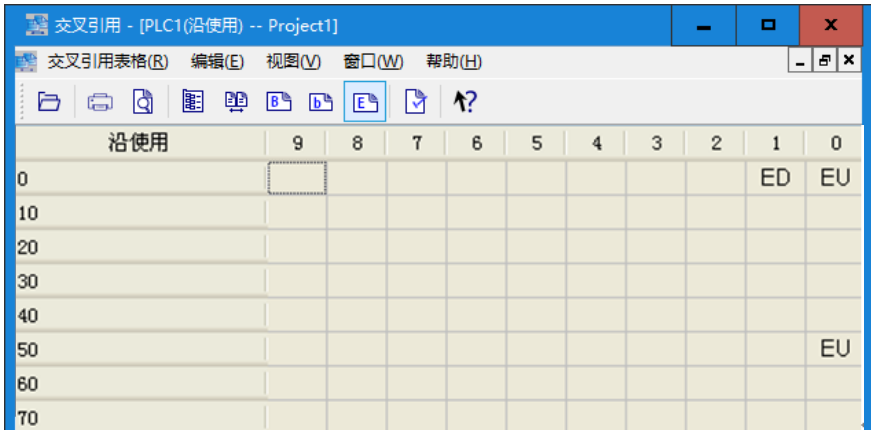
编译后就会出现下面的位使用表:



沿使用表

沿使用表允许您查看程序中已经使用的沿指令的 ID, 有助于您在线编辑时查找沿指令 ID 重复使用错误。

沿使用示例: 由下图可知, 程序中使用了 ID 为 0 和 50 的上升沿指令和 ID 为 1 的下降沿指令。



调用结构
调用结构允许您查看程序中各个块之间的调用和被调用关系。

5.5 向导的使用

向导文件夹  用来存储当前版本支持的向导图标，用户可以通过点击项目树的向导结点打开向导文件夹来查看。当前支持的向导见下表：

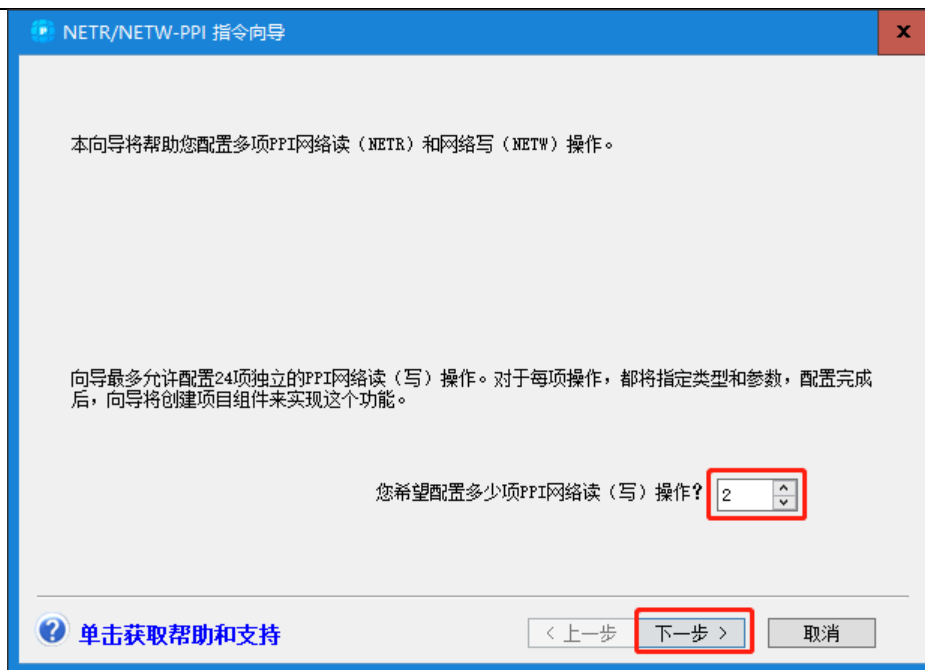
指令向导	PID	支持
	NETR/W-PPI	支持，每条读写 200 字节。
	NETR/W UDP	支持
	MODBUS-TCP	支持
	HSC	支持
	PTO	支持
文本显示向导		支持 TD200 3.0 版，支持 TD400 1.0 和 2.0 版。
配方向导		支持
数据记录向导		支持

5.5.1 NETR/W-PPI 指令向导

- “NETR/W-PPI 指令向导”可以创建项目代码，用于实现网络中的两个 PLC 之间的数据交互。
- 欲使用 NETR/W-PPI 指令向导，请遵循以下基本步骤：
- 1) 选择菜单命令“工具”→“NETR/W-PPI 指令向导”或点击项目树中的向导结点，在右边的工作区域双击 NETR/W-PPI 向导图标 ，启动 NETR/W-PPI 指令向导。并随后打开此向导或某现有配置。
 - 2) 点击“下一步”按钮前往 NETR/W-PPI 向导的每个新页。
 - 3) 完成向导中的步骤后，点击“完成”按钮。

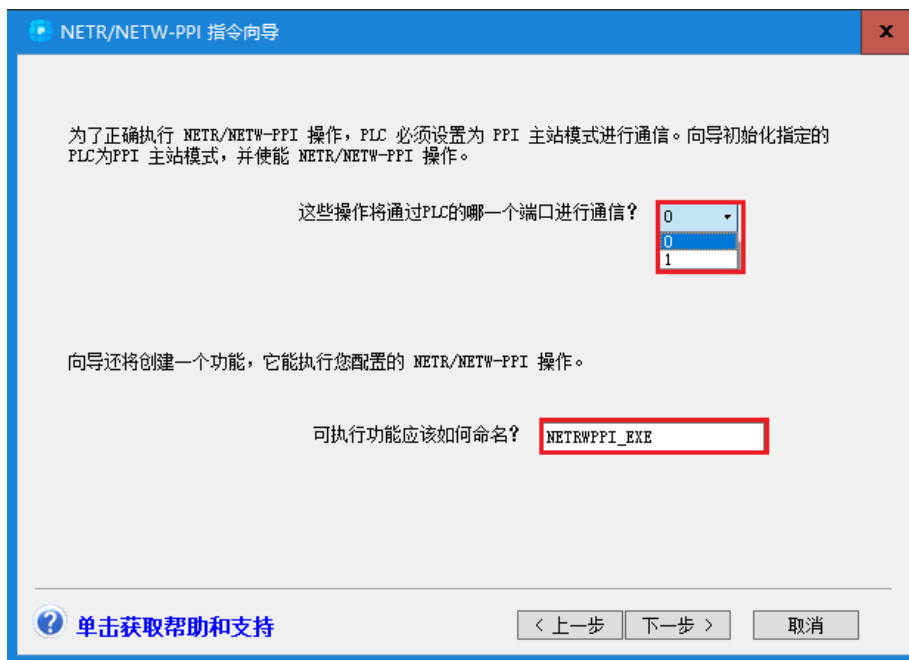
具体操作步骤如下：

<第 1 步> 选择 PPI 通信端口进行网络读写操作，指定要配置的网络操作数：



在对话框“您希望配置多少项 PPI 网络读（写）操作？”中设定要配置的网络操作数，目前允许的操作数范围是 1~24。

<第 2 步> 现在需要指定使用的通信端口和生成的功能名称：



在此页面中你可以更改以下信息：

- 1) 作为 PPI 主站的 PLC 使用哪一个通信端口（PPI 通信口）与 PPI 从站进行通信。
- 2) 编辑向导生成的功能符号名。配置完成后，向导会建立一个用于执行具体网络操作的以此为名的参数化功能。

<第 3 步> 配置网络操作

NETR/NETW-PPI 指令向导

网络读/写操作第 1 项/共 2 项

请选择操作类型
NETW

请指定网络读（写）的字节数
2

本地PLC 远程PLC地址: 2

数据位于本地 PLC 的哪个位置? 数据写入到远程 PLC 的哪个位置?

VB0 至 VB1 VB0 至 VB1

删除操作 < 上一项操作 下一项操作 >

单击获取帮助和支持 < 上一步 下一步 > 取消

NETR/NETW-PPI 指令向导

网络读/写操作第 2 项/共 2 项

请选择操作类型
NETR

请指定网络读（写）的字节数
2

本地PLC 远程PLC地址: 2

数据存储在本地 PLC 的哪个位置? 从远程 PLC 的哪个位置读取数据?

VB2 至 VB3 VB0 至 VB1

删除操作 < 上一项操作 下一项操作 >

单击获取帮助和支持 < 上一步 下一步 > 取消

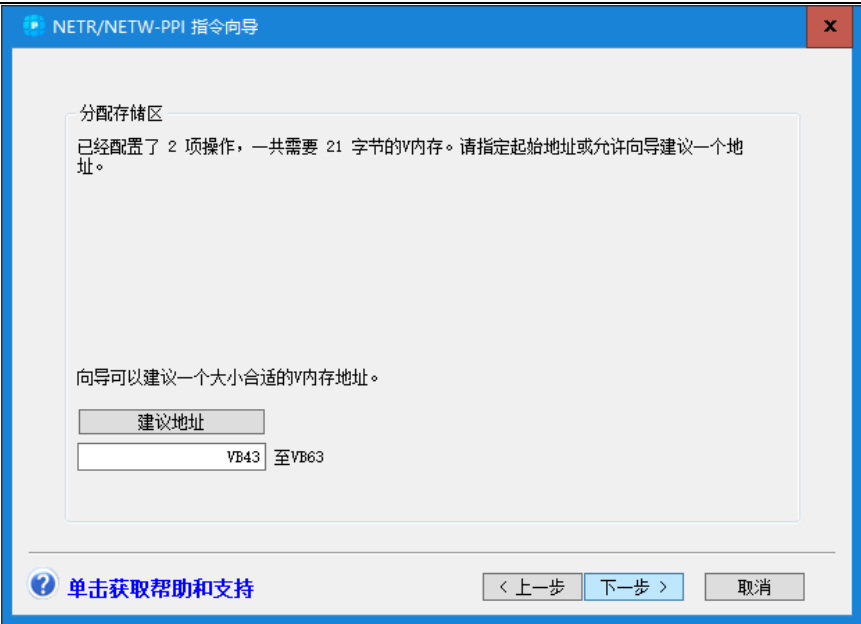
网络操作要配置的信息包括：

- 1) 此操作是网络读还是写
- 2) 从从站读入或写入从站多少个字节
- 3) 要交互的 PLC 从站地址
- 4) 本地 PLC 地址映射（只能是 V 内存），即读回的数据或待写入的数据存储在本地 PLC V 内存中的位置

此界面上三个按钮：

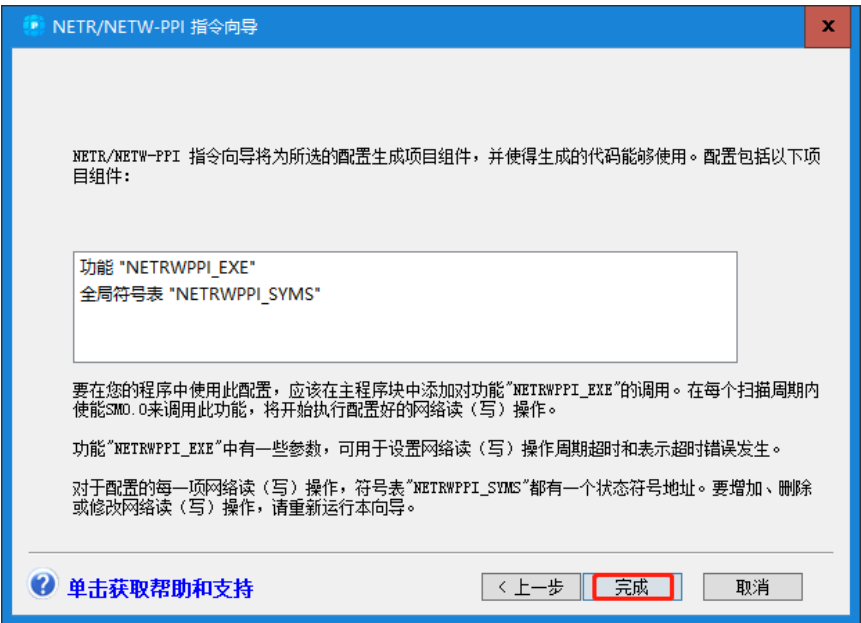
- 删除操作：删除当前的操作，此操作导致已配置的操作数目减一
- 上一个操作：将界面切换到上一个操作
- 下一个操作：将界面切换到下一个操作

<第 4 步> 分配 V 存储区



对于您配置的每一项网络操作，要求有 12 个字节的 V 存储区。在此页中您可以选择想要存放该配置块的 V 内存地址。如果您希望向导建议正确大小的未使用内存块地址，则点击“建议地址”按钮。

<第 5 步> 生成项目组件

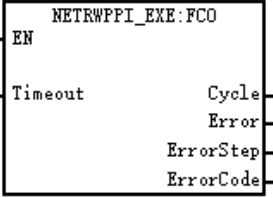


最后一步，指令向导将为您指定的配置生成相关的程序组件，包括：

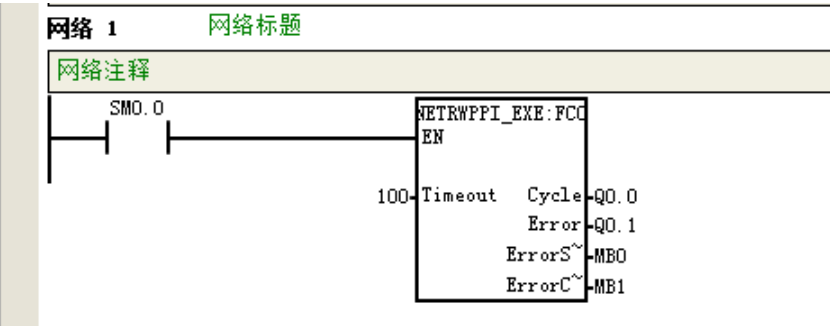
- 网络读写符号表，包含配置中每个操作每个网络读写操作的状态符号地址。
- 功能 NET_EXE 包含了操作的具体代码，要在程序中使用网络通信，需要在主程序块中调用执行功能功能（NET_EXE）。每次扫描周期时，使用 SM0.0 调用该功能。这样会启动配置网络操作执行。为每项网络操作建立的数据处理功能会在适当时间被自动调用。

PPI 向导生成的功能为具体操作指令，具体指令说明如下表所示：

PPI 通信指令	
指令名称	功能块图

NETRWPPI_EXE			
符号	变量类型	数据类型	注释
EN	IN	BOOL	使用 SM0.0 调用
Timeout	IN	INT	0 = 不计时的；1-32767 = 计时值 (秒)。
Cycle	OUT	BOOL	每周周期处理完所有的通信操作时置位。
Error	OUT	BOOL	0 = 无错误；1 = 出错 (检查 NETR/W-PPI 指令缓冲区状态字节以获取错误代码)。
ErrorStep	OUT	BYTE	发生错误的操作步。
ErrorCode	OUT	BYTE	错误详情： 0 无错 1 超时错误，远程站不应答 2 接收错误，应答中存在校验、帧或校验和错误 3 脱机错误，重复站地址或故障硬件引起的冲突 4 队列溢出错误，8 个以上 NETR/NETW 指令被激活 5 违反协议，未启用 SMB30/SMB130 将端口设为 PPI 主站模式即尝试执行 NETR/NETW 6.非法参数，NETR/NETW 表格包含一个非法或无效数值 7 无资源，远程站繁忙（正在上载或下载序列） 8 层 7 错误，违反应用程序协议 9 信息错误，数据地址或数据长度不正确

要在程序中执行 PPI 向导配置的网络操作，您需要在主程序块中使用 SM0.0 调用该功能（NETRWPPI_EXE）。该功能在编程软件中的调用示例如下图所示：



对于本说明中配置生成的功能，相关 PPI 通信操作如下表所述：

操作步	本地 PLC 数据表位置	远程设备操作状态存储位置	数据长度
写操作	VB0-VB1 -> VB0-VB1	NETR1_Status:VB46	2 Bytes
读操作	VB2-VB3 <- VB0-VB1	NETR2_Status:VB55	2 Bytes

通过状态监控表查看配置的各项操作的执行状态时，请参考下面的 UDP 网络读写操作状态字节

定义表：

位编号	定义	说明
1	D	完成（功能完成）位，0 = 未完成，1 = 完成；
2	A	激活位，指令正在执行中为 1，不在执行中为 0；
3	E	错误状态位，0 = 无错，1 = 有错误；
4	n4	保留，始终为 0；
5~8	错误代码	详细代码说明请参见本通信指令注释；

<注意> PPI 通信采用普通网络读写，执行读写操作时可组态多种寄存器，从站地址映射为直接映射方式。

5.5.2 NETR/W-UDP 指令向导

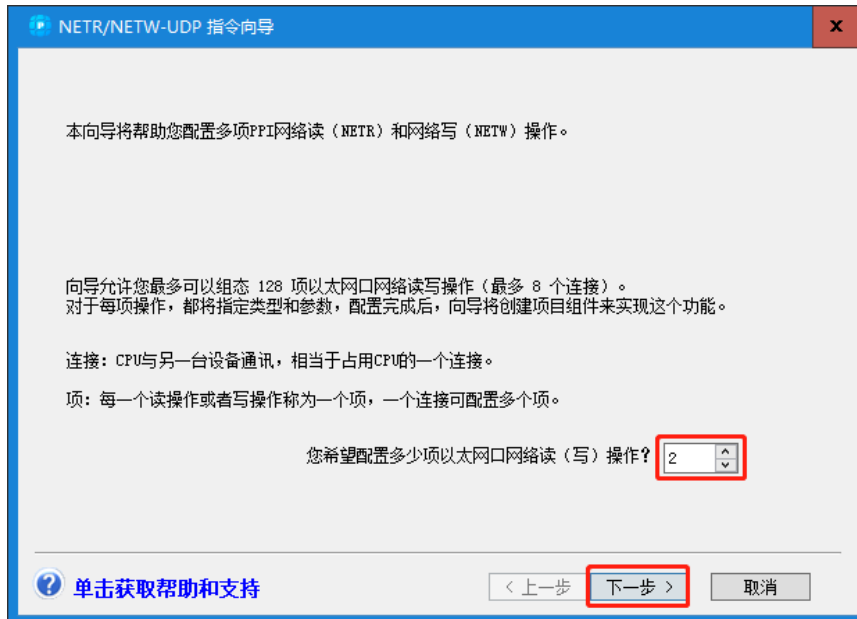
“NETR/W UDP 指令向导”可以创建项目代码，用于实现网络中的两个 PLC 之间的数据交互。

欲使用 NETR/NETW 指令向导，请遵循以下基本步骤：

- 1) 选择菜单命令“工具”→“NETR/W-UDP 指令向导”或点击项目树中的向导结点，在右边的工作区域双击 NETR/W-UDP 向导图标，启动 NETR/W-UDP 指令向导。并随后打开此向导或某现有配置。
- 2) 点击“下一步”按钮前往 NETR/W-UDP 向导的每个新页。
- 3) 完成向导中的步骤后，点击“完成”按钮。

具体操作步骤如下：

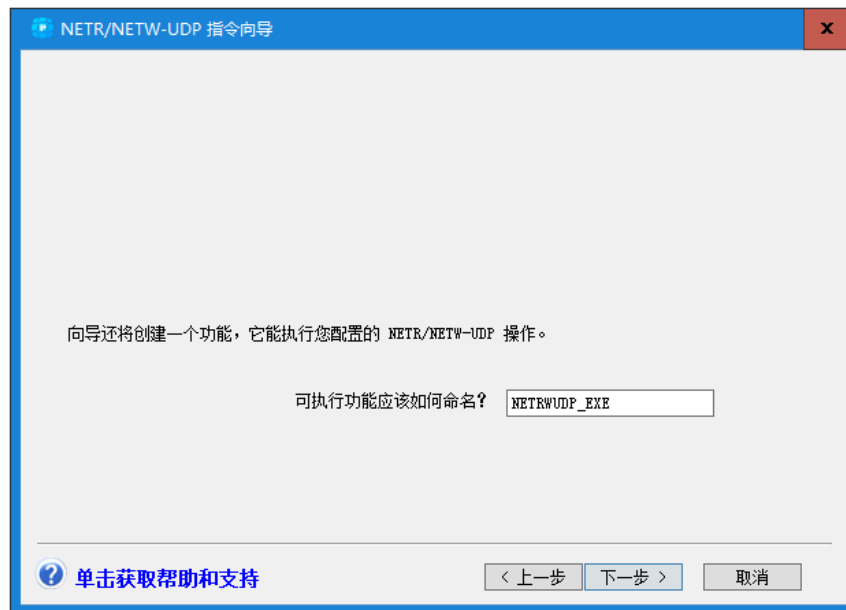
<第 1 步> 选择以太网通信端口进行网络读写操作，指定要配置的网络操作数：



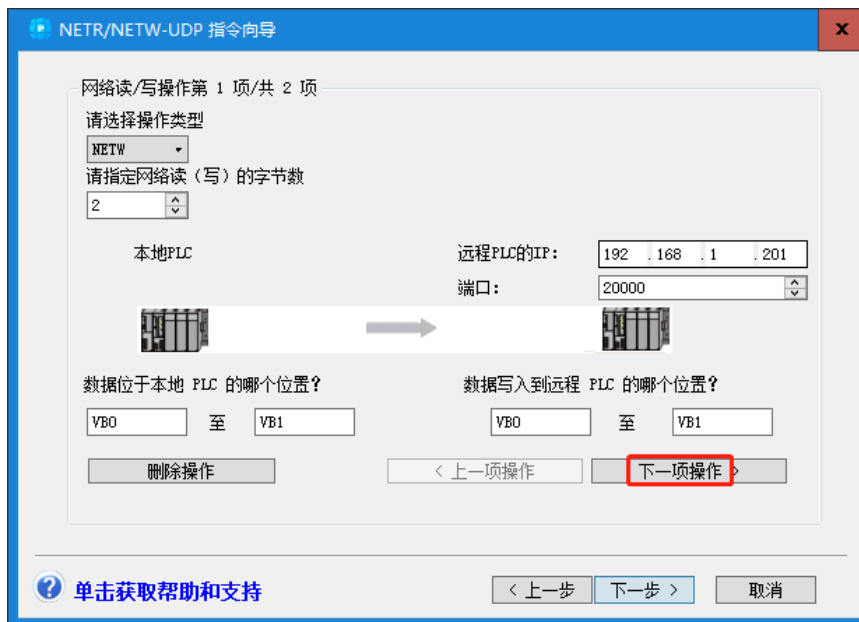
在对话框“您希望配置多少项以太网网络读（写）操作？”中设定要配置的网络操作数，目前允许的操作数范围是 1~24。

<备注> 选择 TCP/IP 通信口（EtherNET 通信口）可以进行 UDP_PPI 通信。

<第 2 步> 编辑向导生成的功能符号名。配置完成后，向导会建立一个用于执行具体网络操作的以此为名的参数化功能。



<第 3 步> 配置网络操作





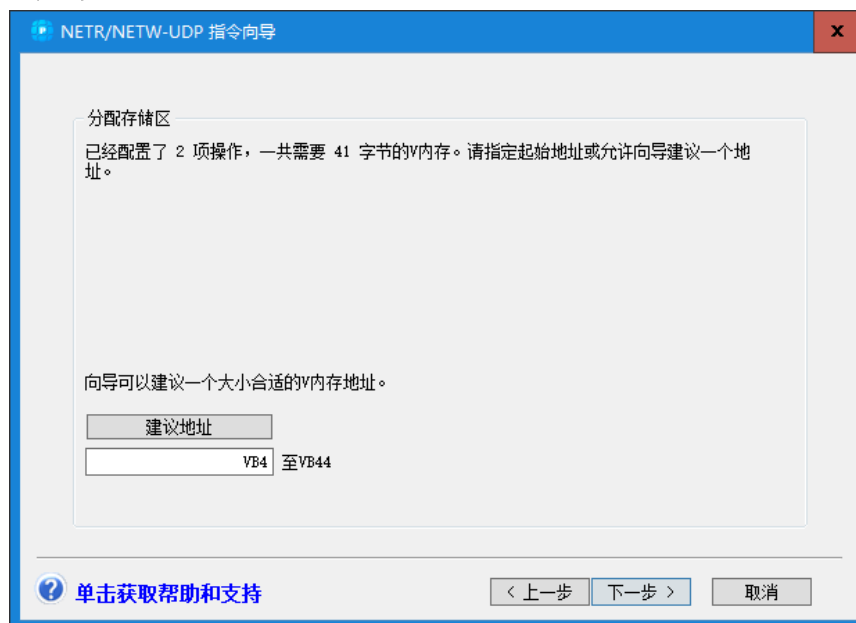
网络操作（PPI 或 TCP/IP 通信口）要配置的信息包括：

- 1) 此操作是网络读还是写
- 2) 从从站读入或写入从站多少个字节
- 3) 要交互的 PLC 从站 IP 地址及端口号
- 4) 远程 PLC 地址映射（只能是 V 内存），即从远程 PLC 读取或向远程 PLC 写入 V 内存位置

此界面上三个按钮：

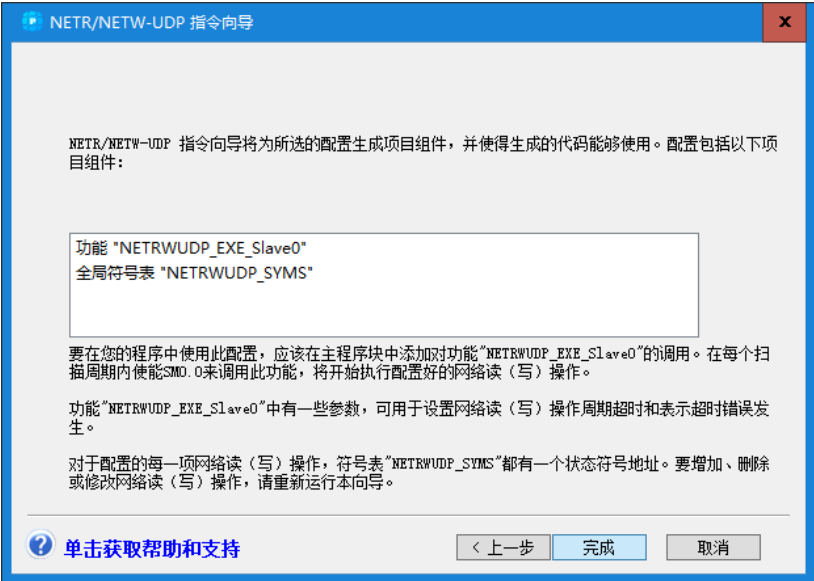
- 删除操作：删除当前的操作，此操作导致已配置的操作数目减一
- 上一个操作：将界面切换到上一个操作
- 下一个操作：将界面切换到下一个操作

<第 4 步> 分配 V 存储区



对于配置的每一项网络操作，要求有 27 个字节的 V 存储区。此页中您可以选择想要存放该配置块的 V 内存地址。若您希望向导建议正确大小的未使用内存块地址，则点击“建议地址”按钮。

<第 5 步> 生成项目组件



最后一步，指令向导将为您指定的配置生成相关的程序组件，包括：

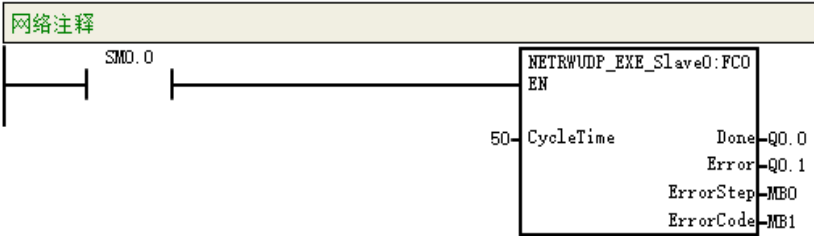
- 网络读写符号表，包含配置中每个操作每个网络读写操作的状态符号地址。
- 功能 NETRWUDP_EXE 包含了操作的具体代码，要在程序中使用网络通信，需要在主中调用执行功能功能（NETRWUDP_EXE）。每次扫描周期时，使用 SM0.0 调用该功能。这样会启动配置网络操作执行。为每项网络操作建立的数据处理功能会在适当时间被自动调用。

UDP 读写指令向导生成的功能为具体操作指令，具体指令说明如下表所示：

UDP 读写指令			
指令名称		功能块图	
NETRWUDP_EXE			
符号	变量类型	数据类型	注释
EN	IN	BOOL	使用 SM0.0 调用
CycleTime	IN	DWORD	周期时间，单位：ms，范围：50~10000
Done	OUT	BOOL	每完成一次所有网络读(写)操作时切换状态。
Error	OUT	BOOL	0 = 无错误；1 = 出错 (检查 NETR/W-UDP 指令缓冲区状态字节以获取错误代码)。
ErrorStep	OUT	BYTE	发生错误的操作步。

ErrorCode	OUT	BYTE	错误详情。 0: 没有错误; 1: 超时错误, 远程站不应答; 2: 接收错误, 应答中存在校验、帧或校验和错误; 3: 脱机错误, 重复站址或故障硬件引起的冲突; 4: 队列溢出错误, 8 个以上 UDP_NETR/UDP_NETW 方框被激活; 6: 非法参数, UDP_NETR/UDP_NETW 表格包含一个非法或是无效数值; 7: 无资源, 远程站繁忙 (正在上载或下载序列); 8: 第 7 层错误, 违反应用程序协议; 9: 信息错误, 数据地址错误或数据长度不正确; 10: 已达到最大连接数 129: 内部错误, 表指针为空

要在程序中执行 UDP 向导配置的网络操作, 您需要在主程序块中使用 SM0.0 调用该功能 (NETRWUDP_EXE)。该功能在编程软件中的调用示例如下图所示:



对于本向导配置生成的功能, 相关 UDP 读写操作如下表所述:

操作步	本地 PLC 数据表位置	远程设备操作状态存储位置	数据长度
写操作	VB0-VB1 -> VB0-VB1	NETR1_Status:VB4	2 Bytes
读操作	VB2-VB3 <- VB0-VB1	NETR2_Status:VB18	2 Bytes

通过状态监控表查看配置的各项操作的执行状态时, 请参考下面的 UDP 网络读写操作状态字节定义表:

位编号	定义	说明
1	D	完成 (功能完成) 位, 0 = 未完成, 1 = 完成;
2	A	激活位, 指令正在执行中为 1, 不在执行中为 0;
3	E	错误状态位, 0 = 无错, 1 = 有错误;
4	n4	保留, 始终为 0;
5~8	错误代码	详细代码说明请参见本通信指令注释;

<注意> UDP 通信采用普通网络读写, 执行读写操作时可组态多种寄存器, 因此从站地址映射为直接映射方式。

5.5.3 MODBUS-TCP 指令向导

使用 MODBUS-TCP 指令向导配置 MODBUS TCP 通信的基本操作:

- 1) 选择菜单命令 “工具” → “MODBUS-TCP 向导” 或选中项目树中的向导结点并在右边的工

作窗口双击 MODBUS-TCP 向导图标, 启动 MODBUS-TCP 向导。

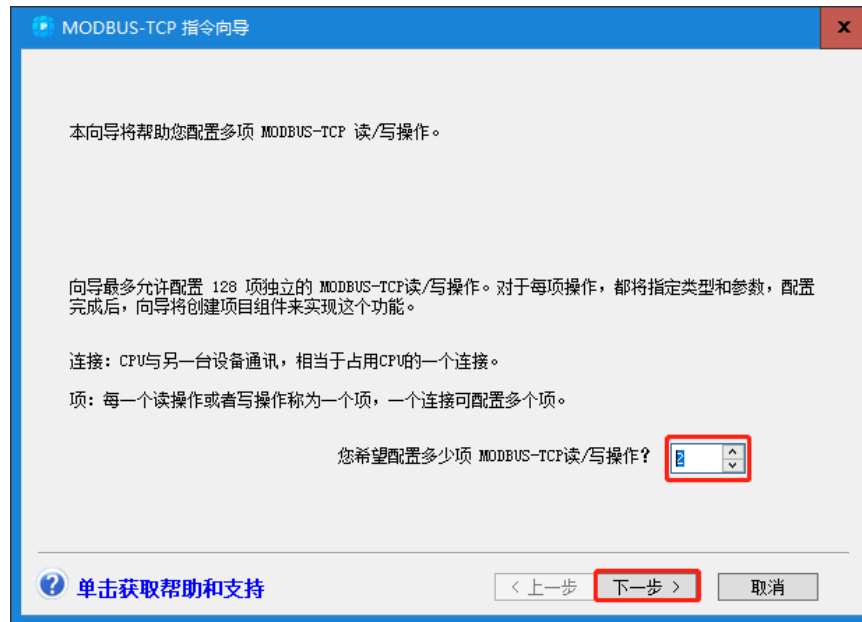
2) 点击“下一步”按钮前往 MODBUS-TCP 向导的每个新部分。

3) 在完成配置时, 点击“完成”按钮。

在向导配置完成时, 会在您的项目创建一个功能 FC, 其中包含您在 MODBUS TCP 向导中指定的信息和地址。供 MODBUS TCP 通信使用的参数和信息存储于 V 存储区中。如果想要在完成后查看已配置的参数块和信息, 您可以打开向导编辑器, 然后选择相应的向导加以检查。

具体操作步骤如下:

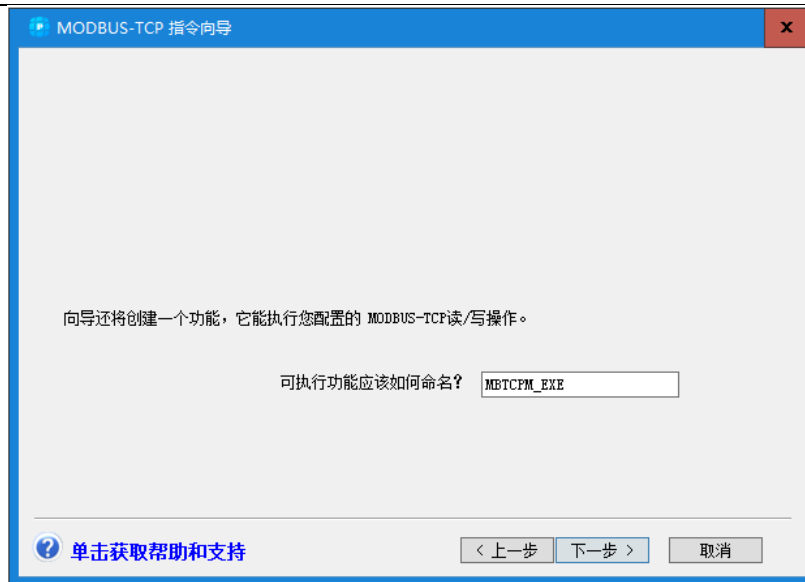
<第 1 步> 指定要配置的网络操作数



在对话框“您希望配置多少项 MODBUS-TCP 读/写操作?”中设定要配置的网络操作数, 目前允许的操作数范围是 1~32。

如果项目中包含有已配置的 MODBUS-TCP 指令向导, 此页面会出现一个删除旧有配置的选择框“移除此 MODBUS-TCP 配置”, 你可以选择删除已有配置或者点击下一步编辑它。如果是编辑现在配置, 打开配置页面时, 所有以前的配置参数都会自动载入。

<第 2 步> 指定生成的功能名称



在此页面中你可以更改以下信息：

编辑向导生成的功能符号名。配置完成后，向导会建立一个用于执行具体网络操作的以此为名的参数化功能。

<第 3 步> 配置网络操作





网络操作要配置的信息包括：

- 1) 此操作是网络读还是写
- 2) 从从站读入或写入从站多少个字节
- 3) 要进行远程通信的 PLC 从站地址、端口号（若为从站设备为 H35-00，此值默认为 502）、单元 ID。
- 4) 本地 PLC 地址映射（只能是 V 内存），即读回的数据或待写入的数据存储在本地 PLC V 内存中的位置
- 5) 远程 PLC 地址映射（只能是 V 内存），即从远程 PLC 读取或向远程 PLC 写入的 V 内存位置

此界面上三个按钮：

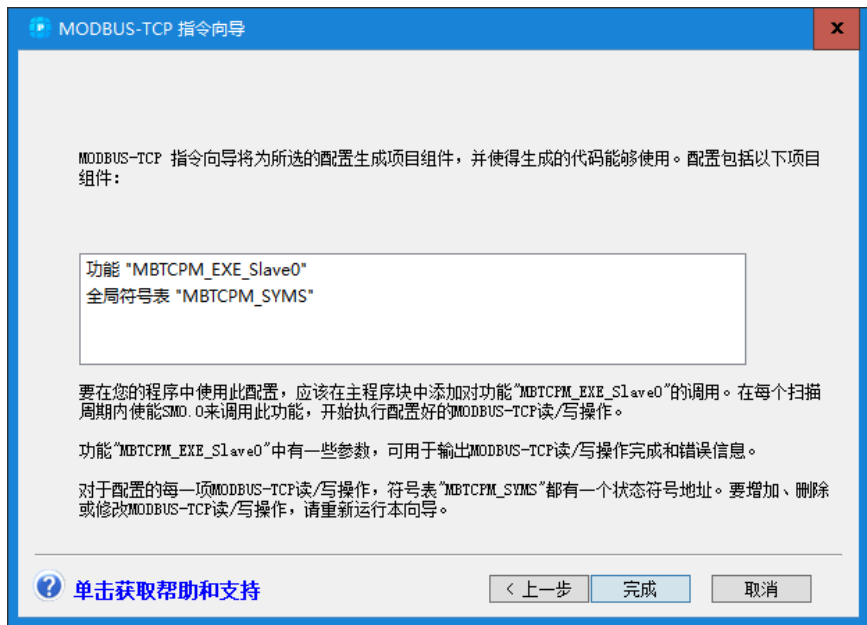
- 删除操作：删除当前的操作，此操作导致已配置的操作数目减一
- 上一个操作：将界面切换到上一个读/写操作
- 下一个操作：将界面切换到下一个读/写操作

<第 4 步> 分配 V 存储区



对于您配置的每一项网络操作，要求有 34 个字节的 V 存储区。在此页中您可以选择想要存放该配置块的 V 内存地址。如果您希望向导建议正确大小的未使用内存块地址，则点击“建议地址”按钮。

<第 5 步> 生成项目组件

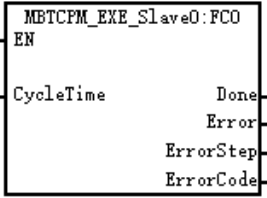


最后一步，指令向导将为您指定的配置生成相关的程序组件，包括：

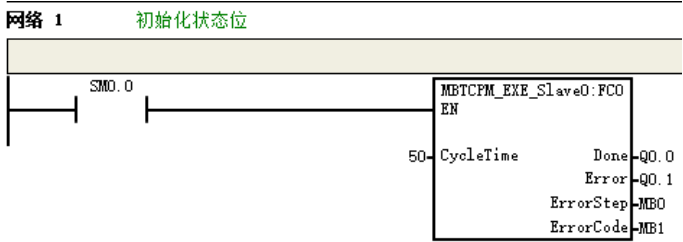
- 网络读写符号表，包含配置中每个操作每个网络读写操作的状态符号地址。
- 功能 MBTCPM_EXE 包含了操作的具体代码，要在程序中使用网络通信，需要在主程序块中调用执行功能功能（MBTCPM_EXE）。每次扫描周期时，使用 SM0.0 调用该功能。这样会启动配置网络操作执行。为每项网络操作建立的数据处理功能会在适当时间被自动调用。

ModBus_TCP 向导生成的功能为具体操作指令，指令说明如下表所示：

ModBus_TCP 通信指令

指令名称		功能块图	
MBTCPM_EXE			
符号	变量类型	数据类型	注释
EN	IN	BOOL	使用 SM0.0 调用
CycleTime	OUT	DWORD	每周期处理完所有的通信操作时置位。
Done	OUT	BOOL	完成标志位。1：完成，0：未完成。
Error	OUT	BYTE	0：无错；1：有错误（详情请查看错误字节）
ErrorStep	OUT	BYTE	发生错误的操作步。
ErrorCode	OUT	BYTE	错误详情。 0：没有错误； 1：连接数达到最大值； 2：正在建立连接； 3：超时错误； 4：请求的参数有错误； 5：指令没有使能； 6：连接正忙于处理其它请求； 7：应答错误； 8：创建连接失败； 9：连接已存在； 10：连接不存在。

要在程序中执行 ModBus_TCP 向导配置的网络操作，您需要在主程序块中使用 SM0.0 调用该功能（MBTCPM_EXE）。该功能在编程软件中的调用示例如下图所示：



对于本说明中配置生成的功能示例，相关 ModBus_TCP 通信操作如下表所述：

操作步	本地 PLC 数据表位置	远程设备操作状态存储位置	数据长度
写操作	VW0-VW2 -> 40001-40002	MBTCPR1_Status:VB64	2 Words
读操作	VW4-VW6 <- 40001-40002	MBTCPR2_Status:VB84	2 Words

通过状态监控表查看配置的各项操作的执行状态时，请参考下面的 MODBUS-TCP 网络读写操作状态字节定义表：

位编号	定义	说明
1	D	完成（功能完成）位，0=未完成，1=完成；

2	A	激活位，指令正在执行中为 1，不在执行中为 0；
3	E	错误状态位，0=无错，1=有错误；
4	n4	保留，始终为 0；
5~8	错误代码	详细代码说明请参见本通信指令注释；

ModBus 地址通常是包含数据类型和偏移量的 5 个或 6 个字符值。前一个或两个字符决定数据类型，后几个字符是一个符合当前数据类型的相应值。ModBus-TCP 从站支持以下地址：

TCP 协议对应的从站地址映射

ModBus 从站地址	CTSC-200 地址
000001	Q0.0
000002	Q0.1
000003	Q0.2
...	...
002047	Q255.6
002048	Q255.7
010001	I0.0
010002	I0.1
010003	I0.2
...	...
012047	I255.6
012048	I255.7
030001	AIW0
030002	AIW2
030003	AIW4
...	...
030512	AIW1022
040001	VW0
040002	VW0+2
...	...
04xxxx	VW0+2 x (xxxx-1)

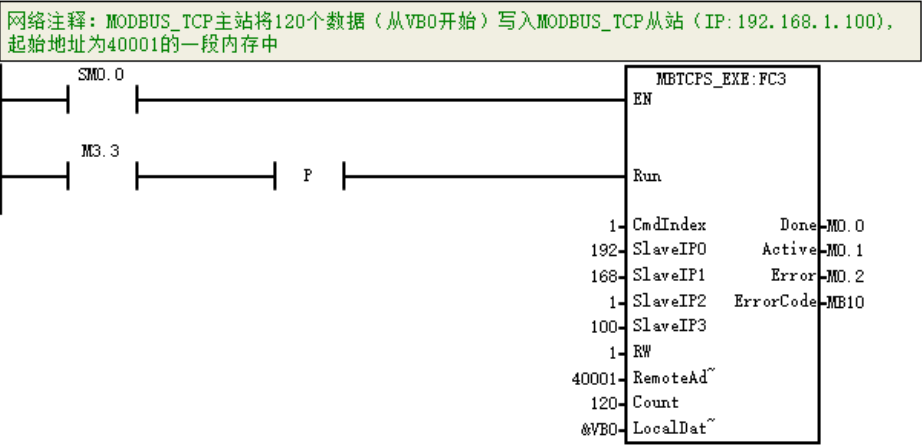
MODBUS-TCP 单条读写指令库

如需执行单次 MODBUS-TCP 网络读写操作，可使用单条读写指令库，请访问合信官网下载 Modbus_TCP 单条读写库文件 ct_mbus_master_tcp_single.ctmwl。下面介绍该库的具体调用方法：

ModBus_TCP 单条读写指令	
指令名称	功能块图

MBTCPS_EXE		MBTCPS_EXE:FC1 EN Run CmdIndexDone SlaveIP0Active SlaveIP1Error SlaveIP2ErrorCode SlaveIP3 RW RemoteAddr Count LocalData~	
符号	变量类型	数据类型	注释
EN	IN	BOOL	指令使能
RUN	IN	BOOL	通信使能位，请使用沿触发
CmdIndex	IN	BYTE	调用 MBTCPS_EXE 的序号，每处调用序号都不能重复，范围 1~255
SlaveIP0	IN	BYTE	从站 IP 地址的第一个字节
SlaveIP1	IN	BYTE	从站 IP 地址的第二个字节
SlaveIP2	IN	BYTE	从站 IP 地址的第三个字节
SlaveIP3	IN	BYTE	从站 IP 地址的第四个字节
RW	IN	BYTE	读写标志，0=读；1=写
RemoteAddr	IN	DWORD	远程数据地址（如 40001）
Count	IN	WORD	元素数目（1-120 个字或 1-1920 位）
LocalDataPtr	IN	DWORD	本地数据指针（如&VB0）
Done	OUT	BOOL	完成标志位（0=未完成；1=已完成）
Active	OUT	BOOL	激活位（0=激活；1=未激活）
Error	OUT	BYTE	错误位，0=无错，1=有错
Errorcode	OUT	BYTE	错误代码，完成位为 1 时有效

主站编程示例如下图所示：

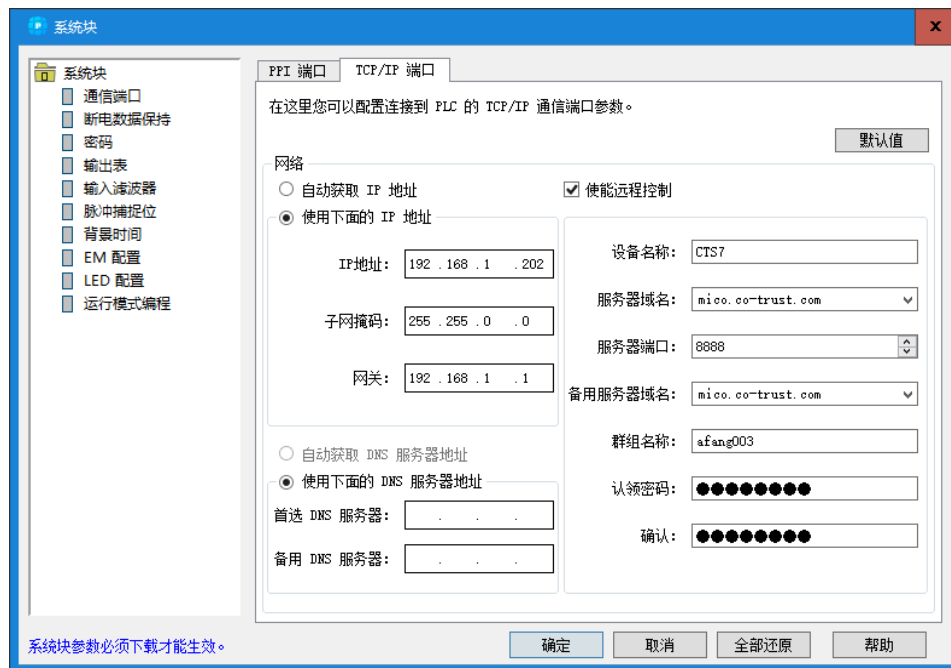


通过 MagicWorks PLC 软件的状态表读取 Modbus_TCP 从站以 40001 起始的 120 个数据，若该段内容与 Modbus_TCP 主站中对应内存中的数据一致，则表示 Modbus TCP 通信成功。

Error 输出端的错误代码定义如下：

错误代码	说明
0	无错
1	连接数达到最大值
2	正在建立连接
3	接收超时，从站无响应
4	请求的参数有错误
5	指令没有使能
6	连接正忙于处理其它请求（如同时激活多条 MBTCPS_EXE）

<注意> 从站 IP 必须对应于主站编程设定的 IP，如需更改从站 IP，请打开 MagicWorks PLC 中的系统块，修改其中的 IP 地址，如下图所示。另外，PLC 之间进行 ModBus TCP 通信时，从站端口号应为 1024 以上，此为系统默认规定。



5.5.4 文本显示向导

请使用文本显示向导配置下列 TD 版本：

- TD 200 V2.1 或之前版本，具有 9 个固定功能按钮。
- TD 200 V3.0 支持多个语言集、自定义用户菜单、用字符串显示的 PLC 数据，并且带有 9 个固定功能按钮（与 TD 200 V1.0 的项目向后兼容）。
- TD 400C V2.0 支持 TD 200 中包含的所有国际字体，并可以显示 4 行文本。键盘设计工具可以为 15 个按钮分配不同的功能。

如果您想了解 TD 200 和 TD400C 的详细功能，请参考对应的用户手册。文本显示向导会引导您逐步完成 TD 的配置，协助您设置 TD 性能的参数，以及输入用户信息。在向导配置完成时，您在向导中指定的信息和地址将用于生成数据页、符号表及指令功能。通过调用文本显示向导为新的配置指定新的起始地址，您可以创建多个文本显示配置。

1) 通过选择菜单命令“工具”→“文本显示向导”或点击项目树中的向导结点，在右边的工作区域双击文本显示向导图标 ，启动文本显示向导。

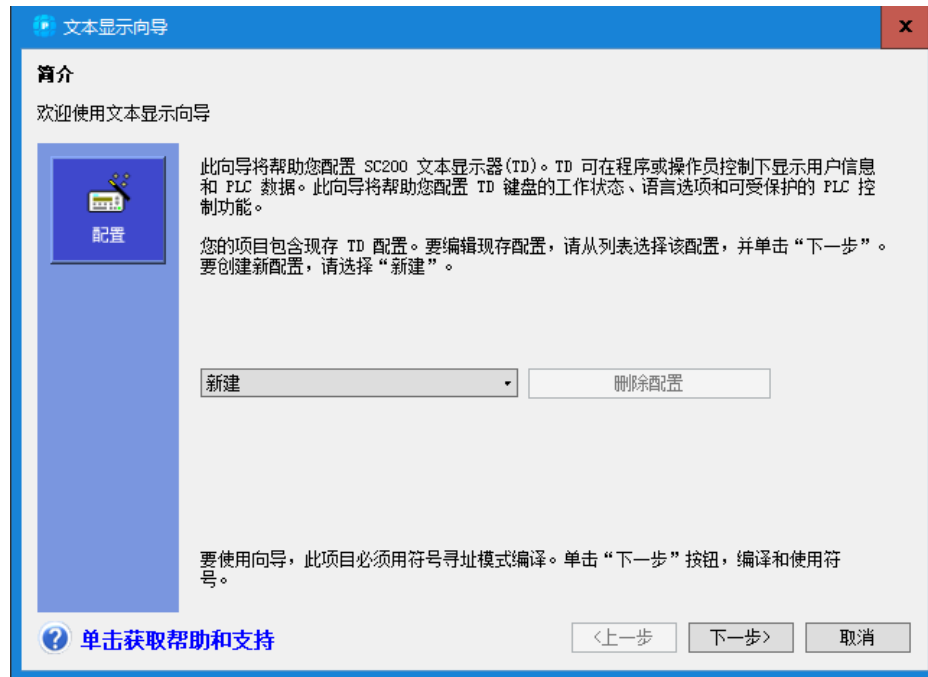
2) 点击“下一步”按钮前往文本显示向导的每个新部分。

3) 在完成配置时，点击“完成”按钮。

在向导配置完成时，会在您的项目创建一个数据页，其中包含您在文本显示向导中指定的信息和地址。供 TD 使用的参数和信息信息存储于 V 存储区中。如果想要在完成后查看已配置的参数块和信息，您可以打开数据块编辑器，然后选择相应的数据页加以检查。

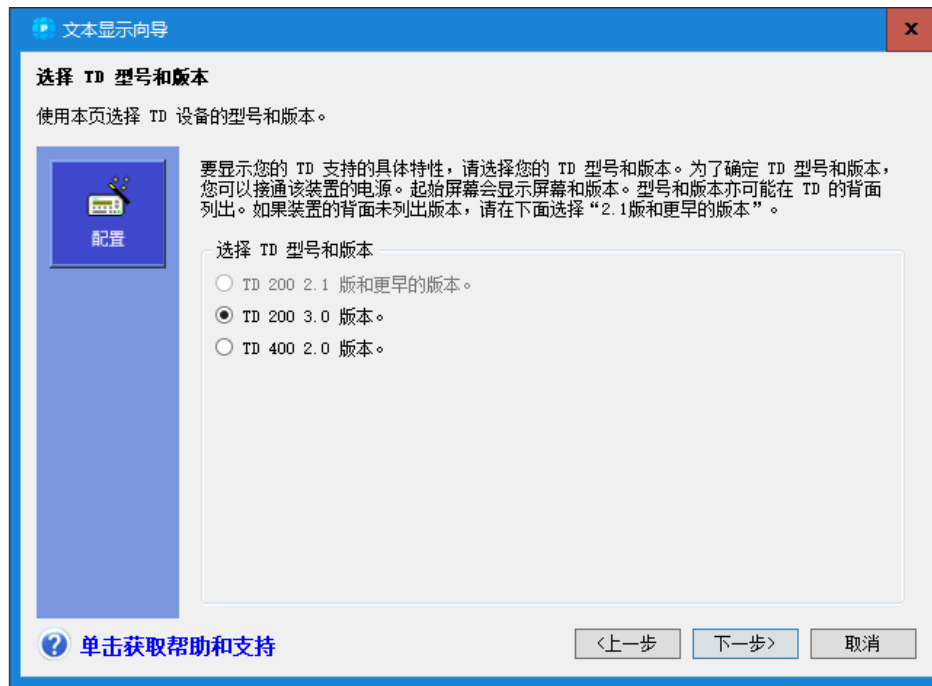
TD 200 V3.0 和 TD 400C 的向导步骤

<第 1 步> 编辑现存配置（TD 200 V3.0 和 TD 400C）



选择您要编辑的配置。如果您的项目没有现存配置，那么点击“下一步”前往第 2 步。

<第 2 步> 选择 TD 型号和版本（TD 200 V3.0 和 TD 400C）



不同的 TD 型号和版本支持不同的功能。为了正确配置 TD，您必须选择与您所使用相符的 TD 型号和版本。TD 200 V2.1 使用与 TD 200 V3.0 和 TD 400C 不同的向导步骤。您可以通过点击适当按钮直接前往现有配置的各个部分。

基本配置

第 1~5 步配置部分使您可设定 TD 设备的参数，包括指定按键行为、语言选项，以及对 PLC 功能的保护程度。这属于必需部分。

菜单

第 6 步（可选）这个部分使您可为该 TD 配置定义一个用户菜单。此用户菜单可用于提供指示性文字信息，或协助机器的故障排除。配置用户菜单并非必需。

报警

第 7 步（可选）这个部分使您可为该 TD 配置定义位触发报警。报警是具有优先顺序的用户信息，会在程序控制下显示。对报警的配置并非必需。

语言

第 7~8 步（可选）如果您定义了报警或用户菜单，“语言集”图标就会出现；这就让您可以定义用户信息的不同语言翻译。

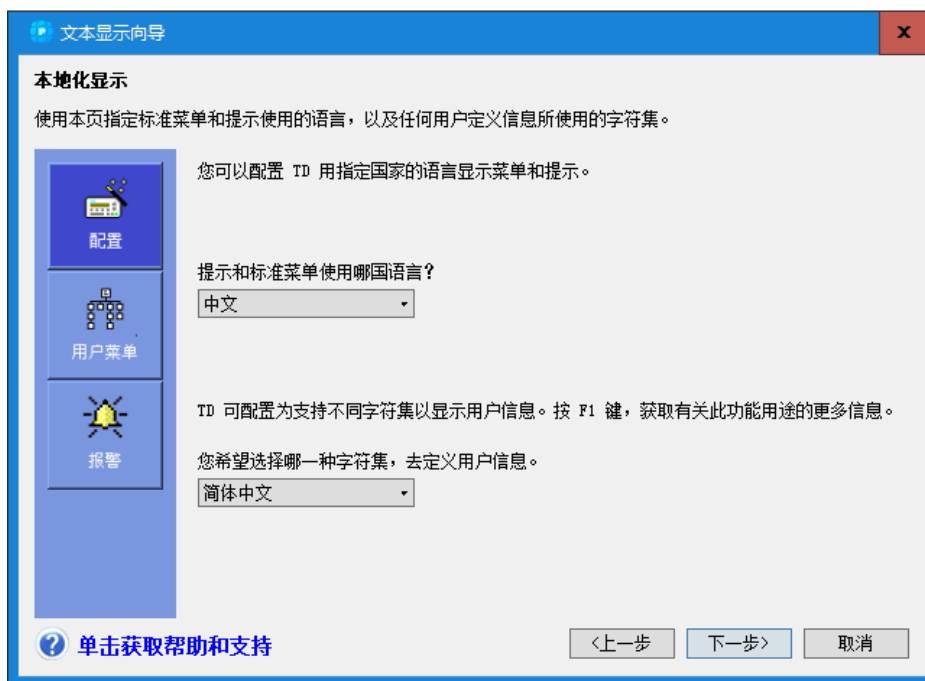
<第 3 步> 使能标准菜单、密码及设置更新速率



此页使用户可以指定显示哪些 TD 菜单选项。这些菜单可因方便用户或出于安全目的而使能或禁止。一旦在向导中使能了标准菜单选择，当 PLC 程序运行时，用户将可以在 TD 中使用该菜单。倘若某个菜单选择已在向导中被禁止，那么用户将无法使用该菜单。

“密码保护”使您可使能一个四位数密码（从 0000 到 9999）。此密码控制着用户编辑嵌入信息中的变量和使用菜单模式的能力。如果使能密码保护，向导中会出现一个域，供设置密码之用。此密码不是 CPU 密码。此密码存储于 TD 配置内，只影响对该 TD 内编辑功能的使用。

<第 4 步> 本地化显示



国家语言：请选择 TD 应为标准提示和菜单使用的国家语言。

字符集：请选择您创建的用户信息应使用的字符集。请注意，选择某字符集并非一定意指特定国家语言。但是，许多国家语言需要特定字符集，方能显示该语言中使用的全部字符。

若是定义了多个语言集（第 8 步）的配置，则“本地化显示”页会针对每个语言集出现。每语言集可指定一个不同的国家语言和（或）字符集，从而更好地支持用户信息的翻译和切换。

- TD 400C V1.0 的标准提示和菜单仅支持英语和简体中文。只有简体中文字符集可以用于用户信息。TD 400C V1.0 也仅在中国范围内供货。
- TD 400C V2.0 支持 TD 200 中包含的所有国际字体。TD 400C V2.0 可在世界范围内供货。
- TD 200 每行可显示多达 20 个单字节字符 TD 400 每行可显示 16 或 24 个单字节字符
- 当您在文本显示向导中输入或由 MagicWorks PLC 的数据块编辑器检查信息时，**Latin 1 Bold**（拉丁 1 粗体）字符不会显示为粗体；但这些字符会在 TD 显示单元上正确显示（即以粗体出现）。
- 简体中文字符为双字节字符。TD 200 每行可显示多达 10 个双字节字符 TD 400 每行可显示 8 或 12 个双字节字符
- 简体中文字符集中有一些 **Latin 1**（拉丁 1）字符。取决于您的键盘和（或）键盘仿真器，**Latin 1** 字符既可能为单字节格式，也可能为如同简体中文字符的双字节字符。如果您在同一条信息中将单字节字符与双字节字符混用，请切记单字节字符的后面只能接单字节字符或空格。

信息编辑器会显示双字节字符位置。每个字符位置可容纳下列组合之一：

- * 一个双字节字符
- * 两个单字节字符
- * 一个单字节字符和一个单字节空格

一个字符位置中不能容纳一个单字节字符加一个双字节字符。

如用户菜单或报警中已有文本被定义，那么修改字体尺寸将删除该用户菜单或报警文本。修改字体尺寸后，必须再次输入该文本。

<注释> 中文字符必须使用小号字体。如果选择中文作为“国家语言”，请进入第 5 步。

<第 5 步> 配置键盘按键



TD 型号提供了用于在 PLC 中设置 V 存储区的“置位”功能按钮。程序逻辑可以监视这些 V 存储区，以检测操作员何时按下 TD 按钮。这些赋给 TD 按钮的 V 存储区在 V 存储区中 TD 参数块的保留区域进行分配。

TD 200: 配合 shift 按钮的 4 个标准“置位”按钮可以控制 8 个 V 存储区

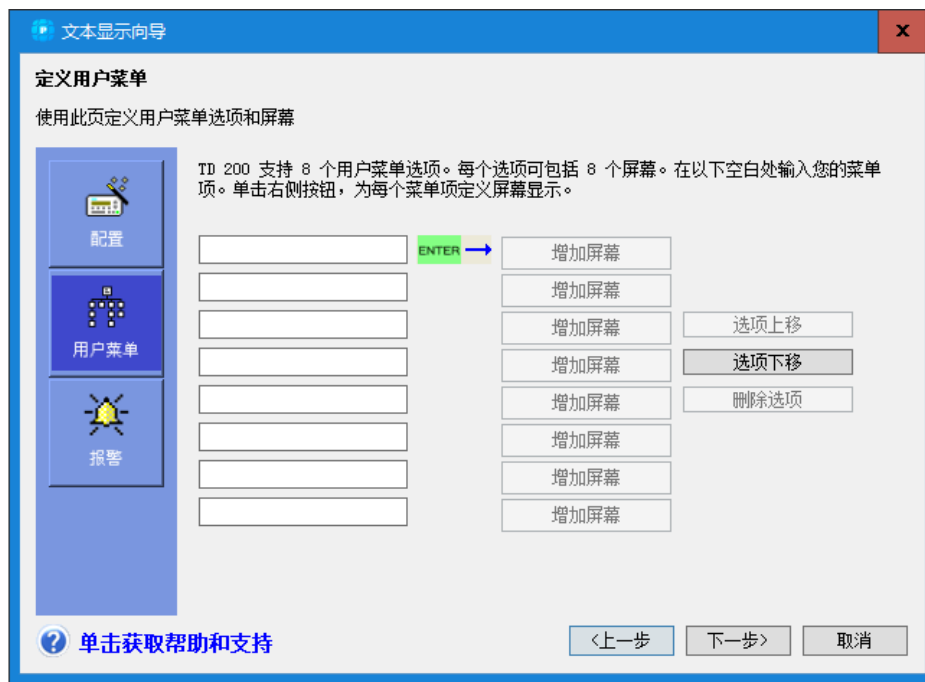
TD 400C: 配合 shift 按钮的 8 个标准“置位”按钮可以控制 16 个 V 存储区

TD 400C: 自定义的键区至多可以拥有 15 个按钮，可控制 28 个 V 存储区（配合 shift 键）

由向导的这个页面，您可以为键盘按钮控制的每个位指定一个符号名。此向导将基于每个按钮的名称建议一个符号名。您可以接受这些建议，或点击每个符号名进行编辑。在完成文本显现向导后，这些符号将会在向导创建的符号表中出现。这些符号名即可用于您的程序逻辑，以检测用户对 TD 按钮的操作。

每个按钮都可配置为置位相应的 PLC 位或作为一个瞬动触点使用。一旦选择“置位”，每当用户按下 TD 按钮上的按钮时，相应的 V 存储区就会置位。该位会保持在已置位状态，只能使用程序逻辑加以清除。倘若某按钮被选择为“瞬动触点”，每当用户按下 TD 键区上的按钮时，相应的 V 存储区就会置位。该位只在此 TD 面板按钮按下期间处于置位状态。当用户放开该按钮时，与其关联的 V 存储区就会被复位。

<第 6 步> 定义用户菜单和文字



TD 200 和 TD 400C 支持一个运行时可由用户命令控制其显示的用户菜单。用户能够使用 TD 按键上的标准按键滚动此菜单，然后选择要显示的标题。一旦选择某菜单选项，用户能够使用箭头按键在该菜单项的屏幕之间滚动。每个 TD 屏幕可包含您自定义的 ASCII 字符、可编辑的 PLC 变量和图标。

您可以配置多达 8 个用户菜单选项，其中每个菜单选项可有 1~8 个屏幕。此用户菜单的显示由 TD 用户而不是 PLC 程序加以控制。如要创建菜单选项，请为您希望使用的每个选择输入文本；然后点击“编辑屏幕”按钮，定义该菜单选项的文本屏幕。

请针对此用户菜单的选择，编辑要显示的信息屏幕。请使用屏幕编辑器创建用户信息。您可用下列方式的组合编写您的信息：

- 点击工具栏上的某个图标按钮，在屏幕内的当前光标位置插入一个图标（只限 TD 200 和 TD 400C）。请注意，此图标不会在 TD 200 向导显示中实际出现。由这个图标占据的位置将用一特殊颜色显示。
- 使用键盘为屏幕输入文字。
- 嵌入一个 PLC 存储区值；方法为点击按钮，然后指定您想要在该信息中显示的数据之地址。每个屏幕最多可包含 6 个 PLC 数据引用。

您可以使用复选框，指定任意屏幕作为默认的“启动”屏幕。如果您删除了已指定的默认屏幕，那么将没有默认屏幕。在没有其他用户操作活动时，默认的启动屏幕将会在 TD 上显示。此默认屏幕也可包括 PLC 数据和图标。当输入屏幕文字时，您可以按键盘的 Insert 键在插入模式和覆盖模式之间切换。

只限 TD 400C：

- 您可以为信息中的每行文字选择大号字体或小号字体。如果选择 TD 400C 作为目标设备，则将显示“字体”按钮。单击“字体”按钮可更改字体大小。
- 新信息的默认字体是小字号（每行 24 个 ASCII 字符或 12 个中文字符）。也可以为这些行

选择大号字体。然而，更改字体将导致已包含文字的信息行重新调整文字。如果更改字体后 PLC 数据参考及信息文本超出信息末尾，则它们将会被删除。

- 在 TD 400C 上最多可以显示两行的大号字体或四行小号字体。
- 如果一个 MagicWorks PLC 项目具有 TD 400C 向导配置，那么必须使用 MagicWorks PLC 4.0 SP4 及以上版本来上传或打开项目。

屏幕编辑器所包括的微型工具栏，提供了剪切、复制、粘贴及删除整个屏幕的功能。信息内容亦可在菜单编辑器和报警编辑器之间移动：

将屏幕的一个副本放在剪贴板内，然后删除此屏幕；屏幕的总数就会减少一个。

将当前屏幕的一个副本放在剪贴板上。

当剪贴板上有 TD 信息时，‘粘贴’按钮就可供使用。粘贴信息不会将该信息从剪贴板上删除。此粘贴的信息会被插入到屏幕列表中当前显示的屏幕上面。粘贴信息会令屏幕的总数增加一个；因此用户如果已有最大数目（8）的屏幕，此操作就无法进行。

删除此屏幕并返回到菜单或屏幕定义页面，屏幕的总数就会减少一个。

倘若此向导配置定义了多个语言集，那么用户菜单简介页会允许您选择想要编辑的语言集。在这种情况下，您会看到现有语言集的一个下拉列表。当前语言集将为默认选择。请注意，您不能由此创建一个语言集，您只能选择针对某现有语言集编辑此用户菜单。新的语言需在‘语言集’页添加。请用您的主要语言创建菜单和屏幕信息，然后编辑此菜单和屏幕信息文字的替代语言版本。您必须使用主要语言增加或删除菜单选项、信息屏幕及 PLC 数据引用。

<第 7 步> 定义报警选项与文字



报警是具有优先顺序的位触发信息，会在 PLC 程序的控制下显示。

选择报警文本长度。文本长度将影响可在单个屏幕上显示的报警的最大数目。

选择默认的显示模式。默认的显示模式将确定是否在通电后或用户不操作 30 秒后显示用户屏幕

或报警。

您可为 TD 200 和 TD 400C 配置多达 80 条报警。如要创建新报警，请点击‘新报警’按钮，修改识别该报警的默认符号名，然后输入信息。请使用‘前一个报警’和‘下一个报警’按钮修改现有报警。新报警的创建顺序就是其优先顺序。此优先顺序由‘转到’下拉列表中括号内的报警号表明。

您可用下列方式的组合创建报警屏幕的用户报警信息：

- 点击工具栏上的某个图标按钮，在屏幕内的当前光标位置插入一个图标（只限 TD 200 和 TD 400C）。请注意，此图标不会在 TD 200 向导显示中实际出现。由这个图标占据的位置将用一特殊颜色显示。
- 用键盘输入文字。
- 嵌入一个 PLC 存储区值；方法为点击按钮，然后指定您想要在该信息中显示的数据之地址。每条报警屏幕最多可包含 6 个 PLC 数据引用。

当输入屏幕文字时，您可以按键盘的 Insert 键在插入模式和覆盖模式之间切换。

只限 TD 400C：

- 您可以为报警信息中的每行文字选择大号字体或小号字体。如果选择 TD 400C 作为目标设备，则将显示“字体”按钮。单击“字体”按钮可更改字体大小。
- 新的报警信息的默认字体是小字号（每行 24 个 ASCII 字符或 12 个中文字符）。也可以为这些行选择大号字体。然而，更改字体将导致已包含文字的信息行重新调整文字。如果更改字体后 PLC 数据参考及信息文本超出信息末尾，则它们将会被删除。
- 在 TD 400C 上最多可以显示两行的大号字体或四行小号字体。
- 如果一个 MagicWorks PLC 项目具有 TD 400C 向导配置，那么必须使用 MagicWorks PLC 4.0 SP4 及以上版本来上传或打开项目。

屏幕编辑器所包括的微型工具栏，提供了剪切、复制、粘贴及删除整个报警屏幕的功能。信息内容亦可在菜单标题编辑器和报警编辑器之间移动：

将报警屏幕的一个副本放在剪贴板内，然后删除此报警屏幕；报警的总数就会减少一个。

将当前报警屏幕的一个副本放在剪贴板上。

在执行粘贴操作时，您将有机会选择插入或覆盖当前报警屏幕。粘贴插入动作会插入一条新报警屏幕，其优先顺序比当前显示之报警高一个。报警的总数会增加一个。粘贴覆盖动作将覆盖当前显示的报警，不会创建新报警屏幕。请注意，剪切和粘贴可用于重新排列报警，以改变其优先顺序。如要变更某报警屏幕的优先顺序，请前往该报警屏幕，然后选择‘剪切’图标；就可将该报警屏幕从其当前位置拆卸并放在剪贴板上。然后前往该报警的新位置（通过视需要点击‘前一个报警’或‘下一个报警’），并且选择‘粘贴’按钮。当信息框出现，询问您是要插入或覆盖时，请选择插入。该报警屏幕现就已在列表中得到移动并改变其优先顺序。

删除此报警屏幕，报警屏幕的总数就会减少一个。

倘若此向导配置定义了多个语言集，那么用户报警简介页会允许您选择想要编辑的语言集。在这种情况下，您会看到现有语言集的一个下拉列表。当前语言集将为默认选择。请注意，您不能由此创建一个语言集，您只能选择针对某现有语言集编辑此用户菜单。新的语言需在“语言集”页添加。请用您的主要语言创建报警，然后编辑此报警文字的替代语言版本。您必须使用

主要语言增加或拆卸报警及 PLC 数据引用。

<第 8 步> 选择要编辑、添加或删除的语言集



如果您定义了用户菜单或位触发报警，“语言集”图标就会出现。此功能可帮助您制作 TD 200 或 TD 400C 项目的国际版本。如果您增加某个替代语言并随后翻译屏幕文字，则用户可用该替代语言显示 TD 文字。

如要增加新语言，请点击下拉列表并选择“新语言集”。

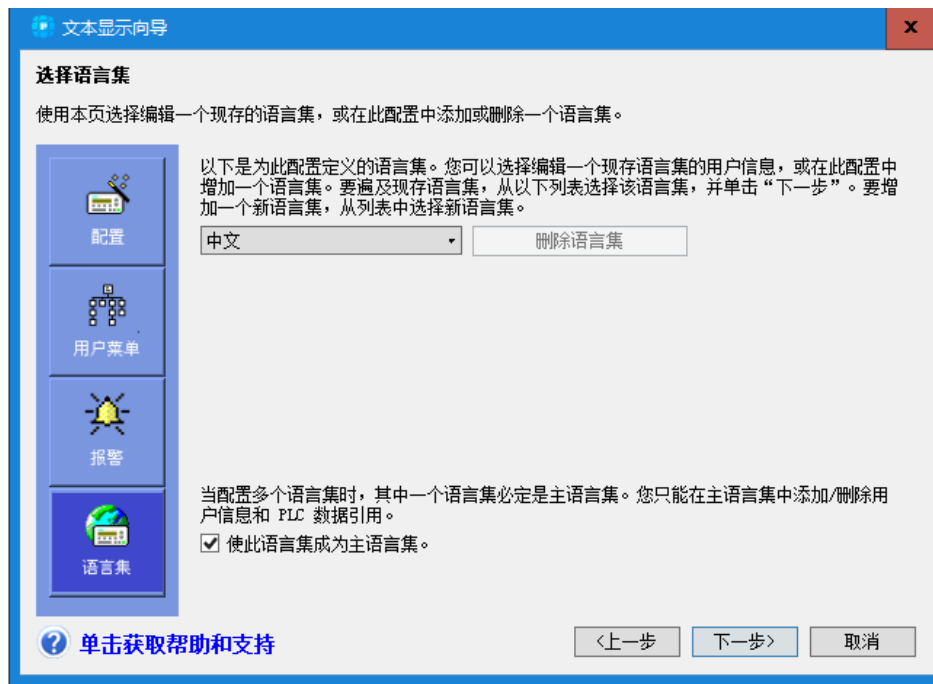
然后选择要复制的语言集。您可以将新语言集作为任何现有语言集的副本加以初始化。与此语言集关联的用户信息就可供翻译。

倘若此配置定义了多个语言集，其中一个语言集必须被指定为‘主语言集’。您只能用主语言集执行下列操作：

- 更改报警选项和添加或删除报警屏幕。
- 添加或删除用户菜单项。
- 添加或删除用户菜单屏幕。
- 增加、删除或编辑报警或菜单屏幕中的 PLC 数据引用。

<注释> 与报警和 PLC 数据引用关联的符号名适用于所有语言集。举例来说，报警 0 的符号名对所有语言集都一样。

<第 9 步> 本地化显示（具有多个语言集）



国家语言：此设定会影响 TD 200 的预置菜单和提示。请选择您希望 TD 显示预置菜单和提示的语言。

字符集：此设定决定了您在创建用户信息时可以使用的字符类型。

- TD 400C V1.0 的标准提示和菜单仅支持英语和简体中文。只有简体中文字符集可以用于用户信息。TD 400C V1.0 也仅在中国范围内供货。
- TD 400CV2.0 支持 TD 200 中包含的所有国际字体。TD 400C V2.0 可在世界范围内供货。
- TD 200 每行可显示多达 20 个单字节字符 TD 400 每行可显示 16 或 24 个单字节字符
- 当您在文本显示向导中输入或由 MagicWorks PLC 的数据块编辑器检查信息时，**Latin 1 Bold**（拉丁 1 粗体）字符不会显示为粗体；但这些字符会在 TD 显示上正确显示（即以粗体出现）。
- 简体中文字符为双字节字符。TD 200 每行可显示多达 10 个双字节字符 TD 400 每行可显示 8 或 12 个双字节字符
- 简体中文字符集中有一些 **Latin 1**（拉丁 1）字符。取决于您的键盘和（或）键盘仿真器，**Latin 1** 字符既可能为单字节格式，也可能为如同简体中文字符的双字节字符。如果您在同一条信息中将单字节字符与双字节字符混用，请切记单字节字符的后面只能接单字节字符或空格。

信息编辑器会显示 10 个双字节字符位置。每个字符位置可容纳下列组合之一：

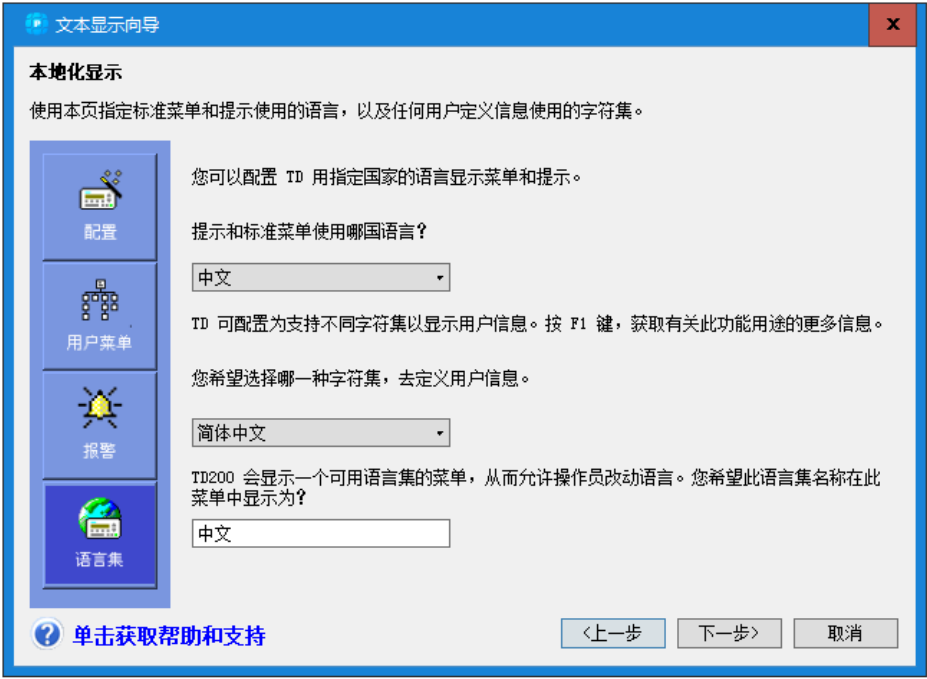
- * 一个双字节字符
- * 两个单字节字符
- * 一个单字节字符和一个单字节空格

一个字符位置中不能容纳一个单字节字符加一个双字节字符。

语言集名称

任何配置的第一个语言集将得到一个默认名称；此名称与在向导的‘配置’部分指定的 TD 语言相符。当您增加新的语言集时，可以修改此默认名。

<第 10 步> 向导部分完成



基本配置 第 1~5 步此配置部分使您可设定 TD 设备的参数，包括指定键盘行为、语言选项，以及对 PLC 功能的保护程度。这属于必需部分。

用户菜单

第 6 步（可选）这个部分使您可为该 TD 配置定义一个用户菜单。用户菜单并非必需。

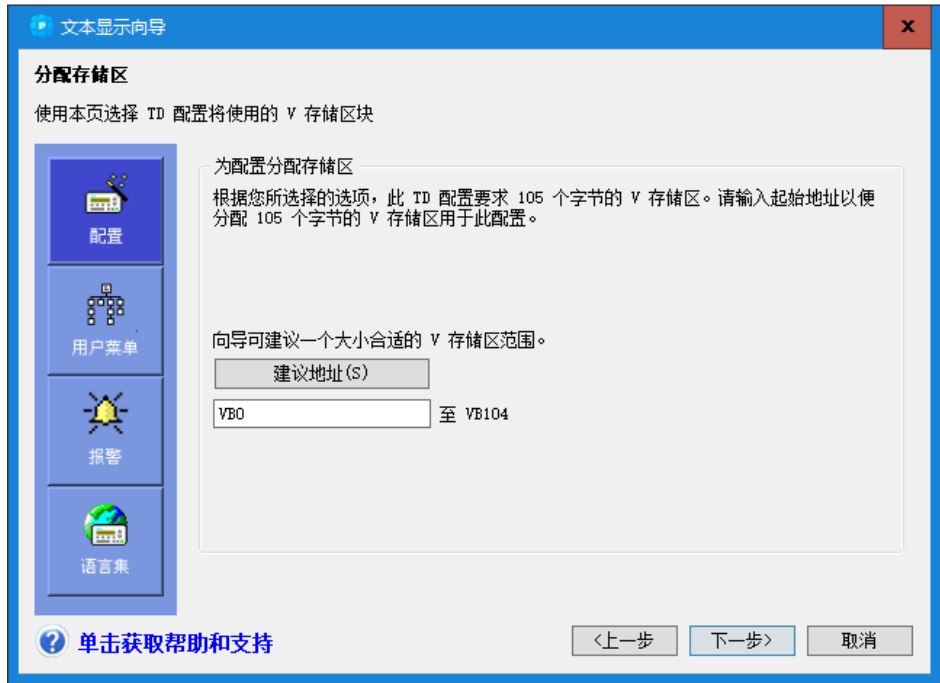
报警

第 7 步（可选）这个部分使您可定义存储位触发的报警。报警并非必需。

语言集

第 8~9 步（可选）如果您定义了报警或用户菜单，“语言集”图标就会出现；这就让您可以定义用户信息的不同语言翻译。

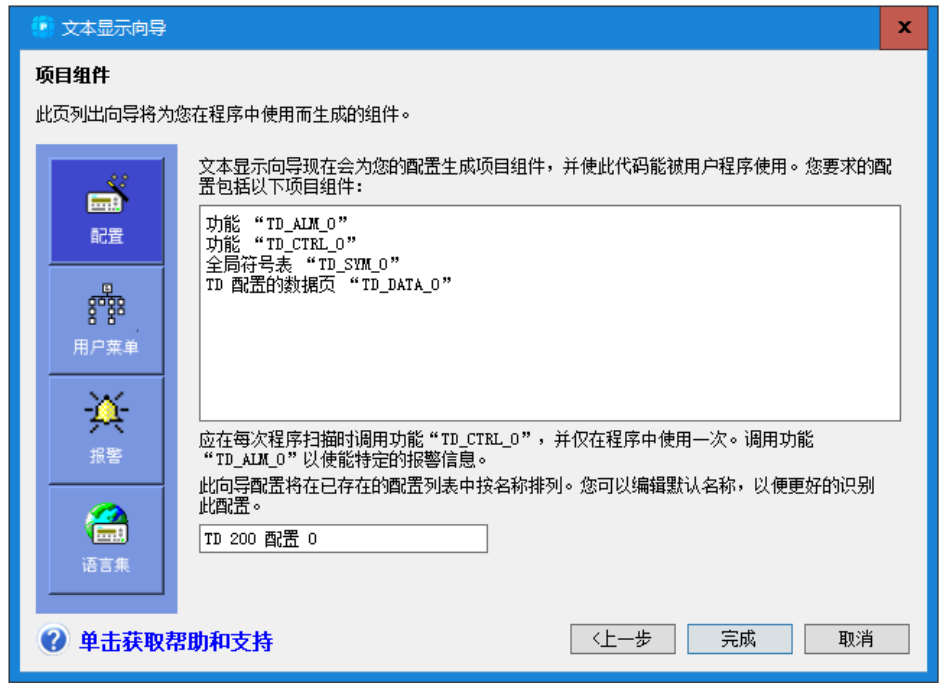
<第 11 步> 为配置分配存储区



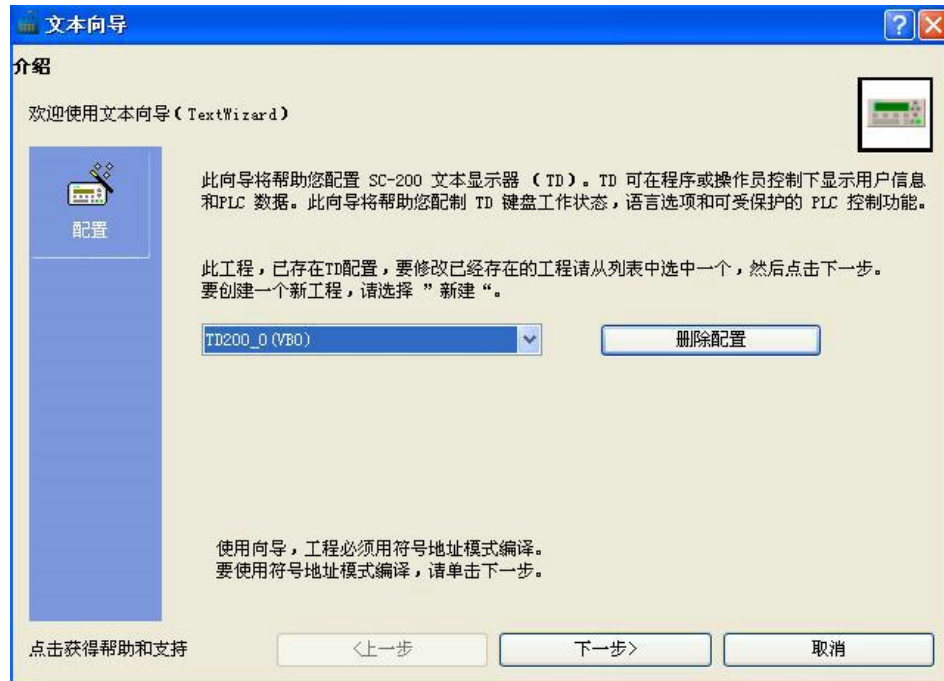
当您就特定配置初次到达此页时，该配置会自动得到建议的 V 存储区地址。此建议地址已针对程序的交叉引用得到检查，确定是未被程序使用的存储区。

点击“建议地址”按钮，向导就会寻找下一个（具有足够容量的）可用 V 存储区块。举例来说，假如您输入地址 VB300，并且此配置的大小是 132 个字节，那么‘建议地址’功能将会在 VB432 开始寻找一个 V 存储区块。下一个具有足够容量的可用存储区块将得到使用，并且‘至’地址会被更新。

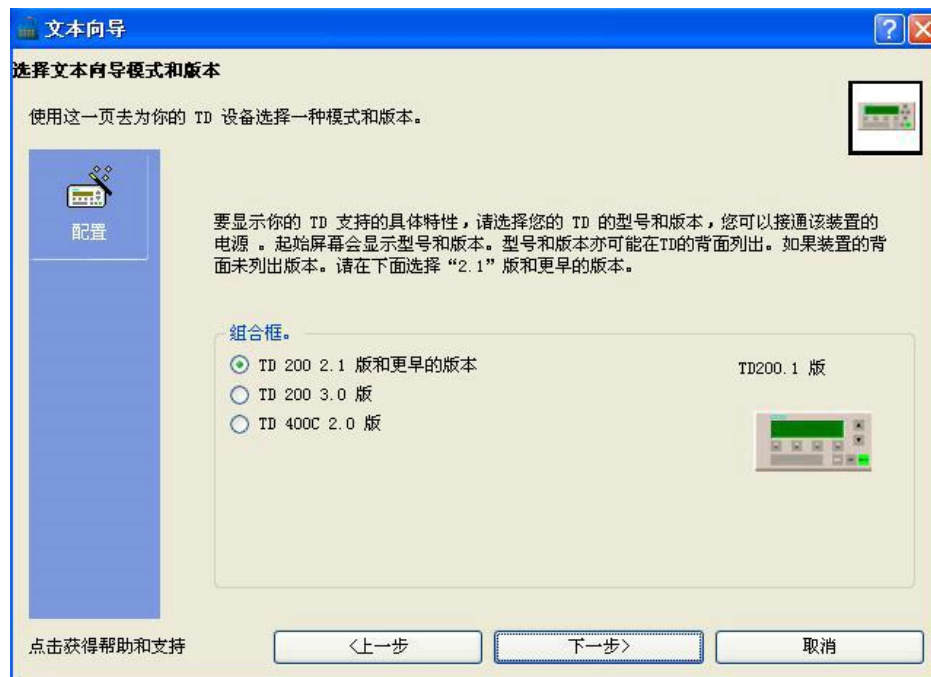
<第 12 步> 生成项目组件



TD 200 V2.1 或之前版本的文本显示向导步骤

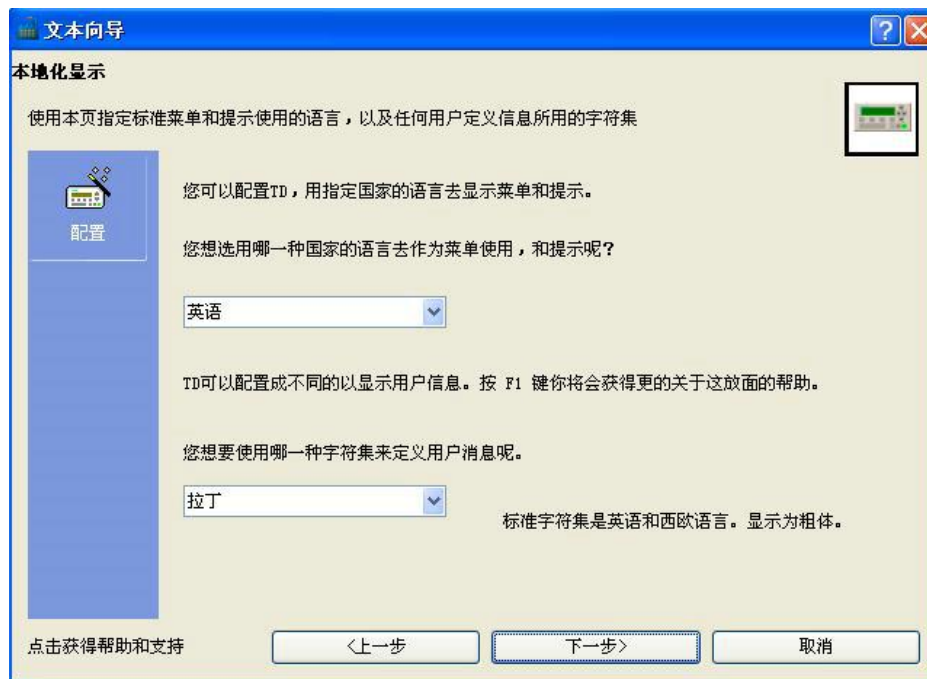
<第 1 步> 编辑现有配置（TD 200 V2.1 或之前版本）

选择您想要编辑的配置。如果您的项目没有现存的配置，那么点击“下一步”前往第 2 步。

<第 2 步> 选择 TD 型号和版本（TD 200 V2.1 或之前版本）

不同的 TD 型号和版本支持不同的功能。为了正确配置 TD，您必须选择所用的型号和版本。TD 200 V3.0 和 TD 400C 使用与 TD 200 V2.1 不同的向导步骤。

<第 3 步> 本地化显示



国家语言：此设定会影响 TD 的预置菜单和提示。请选择您希望用于预置菜单和提示的语言。

字符集：此设定决定了您在创建用户信息时可以使用的字符类型。

- 每行信息最多可显示 20 个单字节字符。

<注释>

当您在文本显示向导中键入或由 MagicWorks PLC 的数据块编辑器检查信息时，Latin 1 Bold（拉丁 1 粗体）字符不会显示为粗体；但这些字符会在 TD 显示单元上正确显示（即以粗体出现）。

简体中文字符集中有一些拉丁 1 字符。取决于您的键盘和（或）仿真器，拉丁 1 字符既可能为单字节格式，也可能为如同简体中文字符的双字节字符。如果您在同一条信息中将单字节字符与双字节字符混用，请切记单字节字符的后面只能接单字节字符或空格。

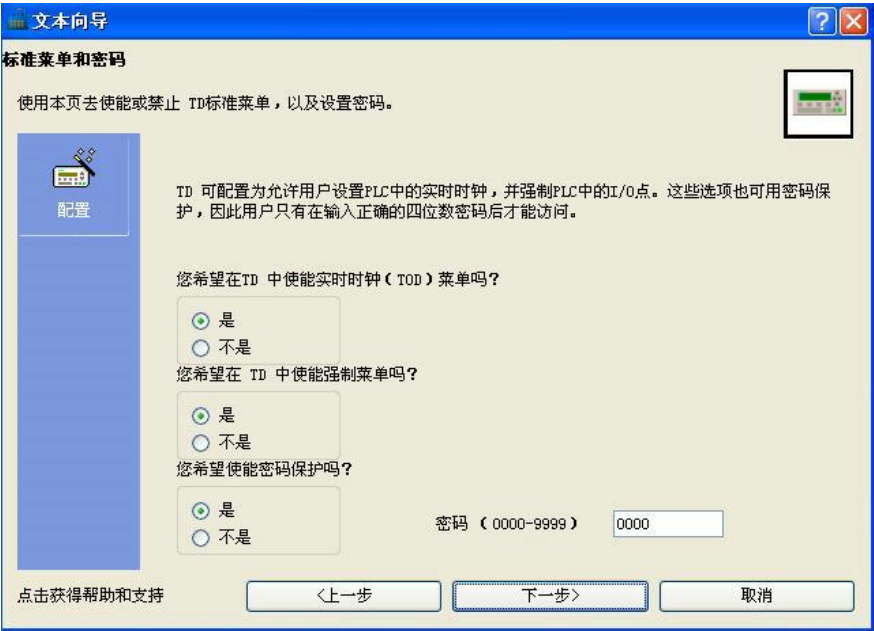
信息编辑器会显示 10 个双字节字符位置。每个字符位置可容纳下列组合之一：

- * 一个双字节字符
- * 两个单字节字符
- * 一个单字节字符和一个单字节空格

一个字符位置中不能容纳一个单字节字符加一个双字节字符。

提示：如果您在使用 2.0 版之前的 TD200 版本，请勿使用默认字符集（拉丁 1）。请只使用原版 TD 200 字符集或条形图字符集。倘若选择了所用 TD 版本不支持的字符集，在下次使用新配置启动 TD 时，您将收到一个关于参数块的未知错误信息。如要纠正此错误，请更改字符集选择，然后下载修正后的配置。如果您不知道所用 TD 的版本，请将它关闭后重新启动。版本号会在接通电源时显示。

<第 4 步> 标准菜单（实时时钟、强制、密码）

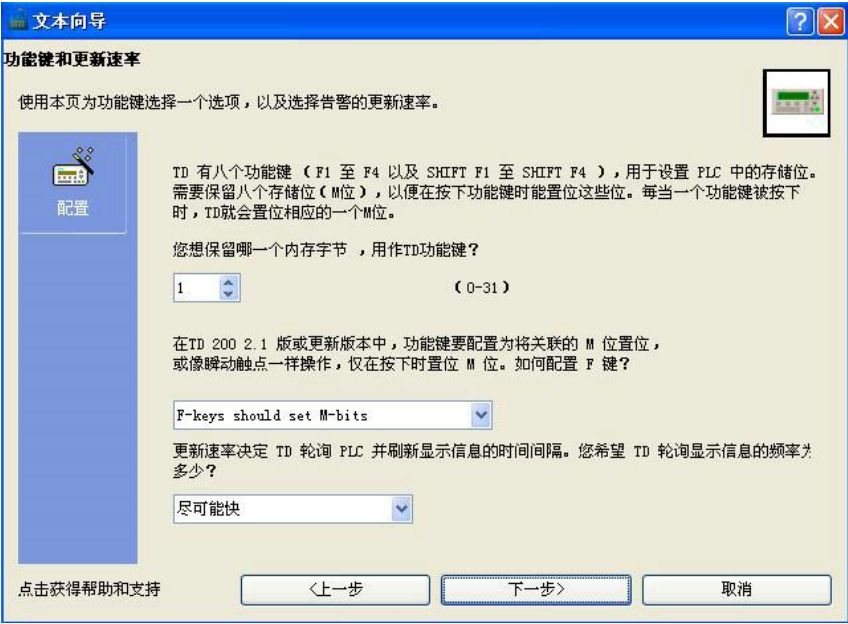


“实时时钟（TOD）”和“强制”菜单选择可用来选择是否使能“TOD 时钟”菜单和“强制”菜单。一旦使能某个选择，用户就可在 TD 中使用相应菜单。若某个选择未使能，该菜单不会在 TD 的菜单模式中出现。

“密码保护”可使能一个四位数密码（从 0000 到 9999）保护。此密码控制着用户编辑嵌入信息中的变量和使用菜单模式的能力。如果使能密码保护，向导中会出现一个域，供设置密码之用。此密码不是 CPU 密码。此密码存储于 TD 配置块内，只影响对该 TD 内编辑功能的使用。

提示：嵌入数据在向导中后面出现的一个屏幕上加以配置，但此处的密码保护选项会影响到您在后面能否选择用密码保护“用户编辑数据功能”。

<第 5 步> 功能键和更新速率



TD 具有 8 个用于把存储区位置位或作为瞬动触点使用的功能键（F1 至 F4 和 SHIFT F1 至

SHIFT F4)。为了读取 TD 功能键，必须为 TD 保留存储区的八个位（M 位）。每当某个功能键被按下时，TD 就会设置相应 M 位的状态。

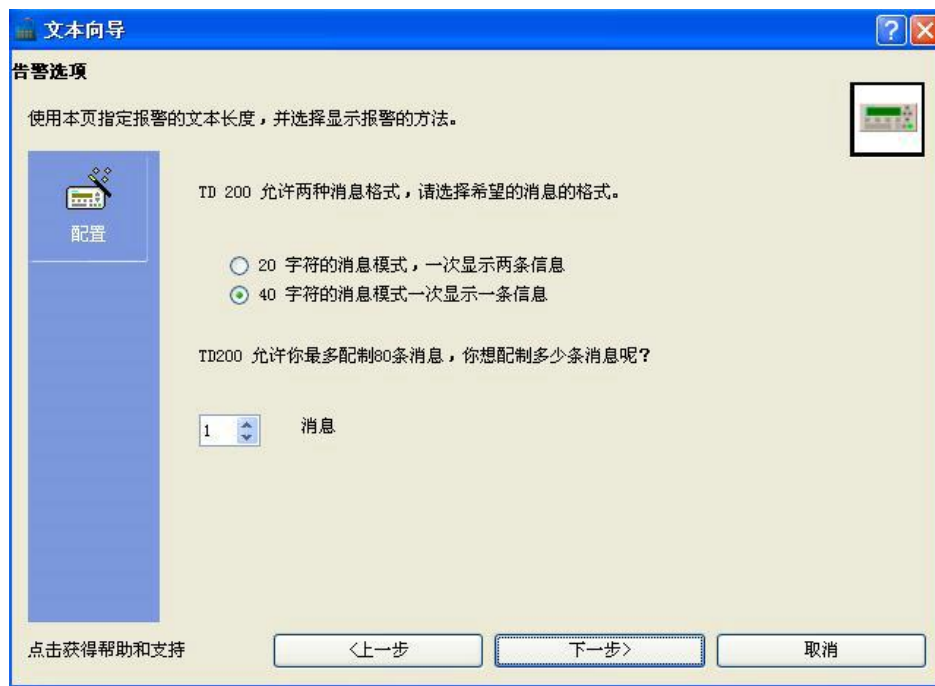
- 请为 TD 输入一个 M 存储区字节地址。
- 选择 TD 的 F 键行为。这些 F 键能够将 M 位如同锁存触点或瞬动触点那样操作。

警告每当某个功能键被按下时，TD 就会设置相应的 M 位。如果您不想使用功能键，并且因此没有指定 M 字节地址，TD 为这些功能键指定的默认值为字节 M0。如果您的程序使用 M0 中的位，并且某用户按下任何功能键，TD 就会设置 M0 中的相应位，从而重写已由程序赋予该位的数值。对 M 位的无意改变可能会使您的程序有意外动作。不可预知的控制器操作可能造成人员伤亡或严重伤害，以及设备损坏。即使您的程序不使用功能键，也请总是保留一个 M 区地址。

更新速率决定着 TD 轮询 PLC 以查看有没有要显示信息的频率。

请由列表框中选择一个更新速率。

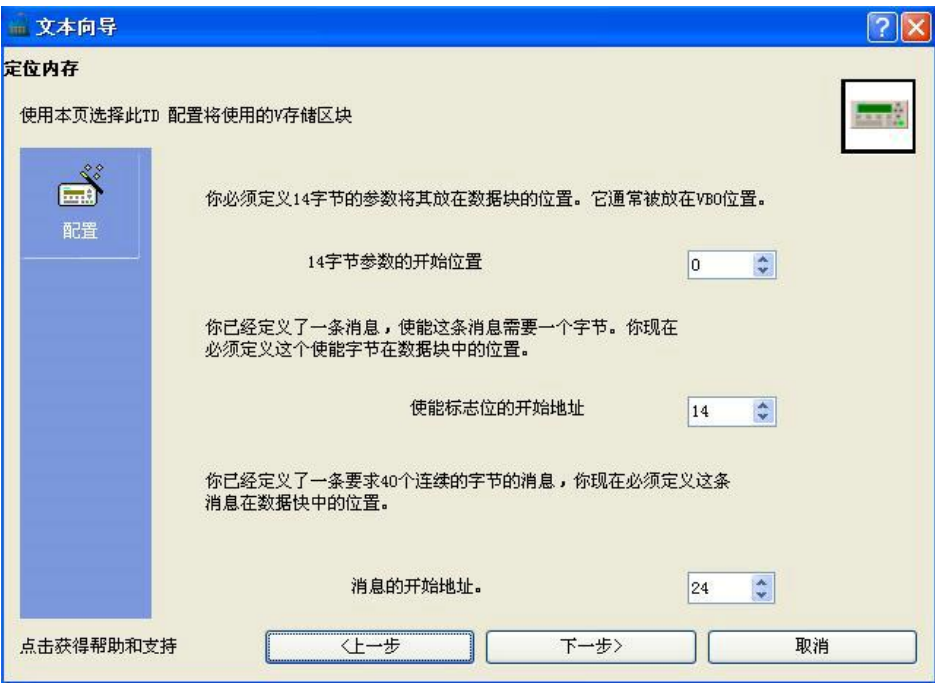
<第 6 步> 报警选项



TD 允许两种信息大小。请选择要用的大小：

- 20 字符信息模式：每次显示两条信息
- 40 字符信息模式：每次显示一条信息
- TD 允许您配置多达 80 条信息。
- 请输入您想要创建的信息数目。

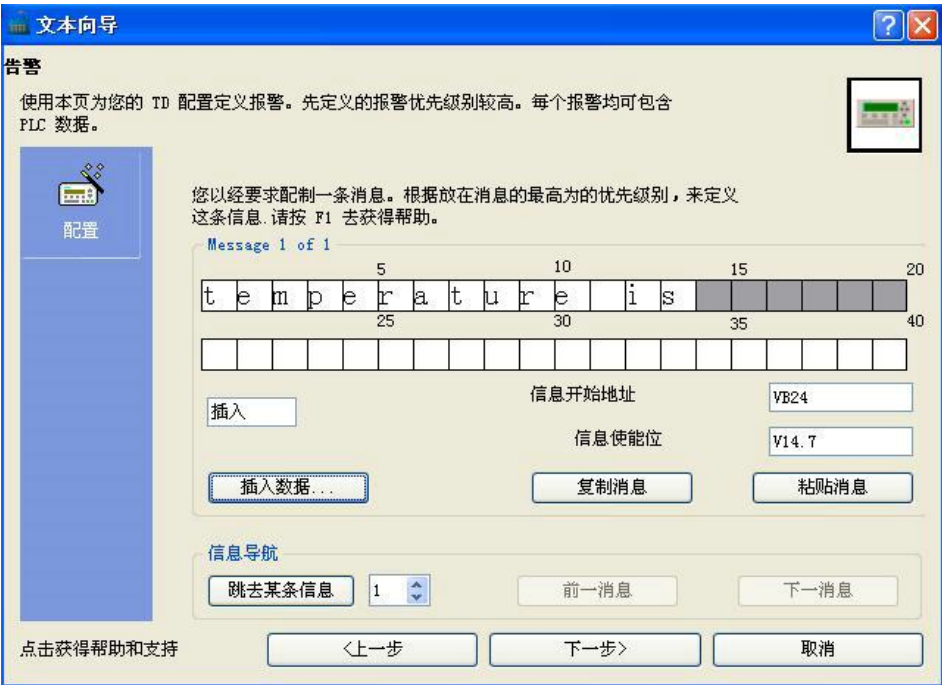
<第 7 步> 分配存储区



必须指定存储参数定义的位置，通常起始地址为 VB0。

- 请输入参数块的起始字节地址（最多可包含 14 个字节）。
- 请输入信息使能标志的起始字节地址。
- 请输入信息内容的起始字节地址。

<第 8 步> 报警（输入信息）



在 20 字符域中输入您的信息。

<注释> 原版 TD 200 字符集和棒图字符集必须使用特别 ALT 键组合键入信息字符。所有其他字

符集可用普通键盘和（或）键盘仿真器键入。

您能够复制当前可见的信息并粘贴到下一条信息。

在“信息浏览”页中，您可以用下列两种方式之一转到另一条信息：

- 点击“上一个信息”和“下一个信息”按钮，或者
- 从“转到信息”框中选择适当信息，然后点击“转到信息”按钮，即可直接提取任何信息。

原版 TD 200 字符集的 ALT 键组合

TD 200 支持对特定国际字符和特殊字符的显示，这些字符是在文本显示向导中使用 ALT 键组合加以输入。如要使用 ALT 键组合：

- 1) 确定键盘上的 NUM LOCK 已激活。
- 2) 按住 ALT 键。
- 3) 使用键盘上的数字按键，键入该字符的 4 位数代码。
- 4) 放开 ALT 键。

请注意，您在文本显示向导中输入的某些字符，在数据块编辑器中有不同的表示。但是，这些字符在 TD 中的显示将与您在文本显示向导中的输入一样。

如要在文本显示向导中输入棒图字符，您必须使用下列 ALT 键组合。请注意，这些字符在此向导（及数据块编辑器）中的表示，与在 TD 本身中不同，但会在 TD 200 中正确显示。

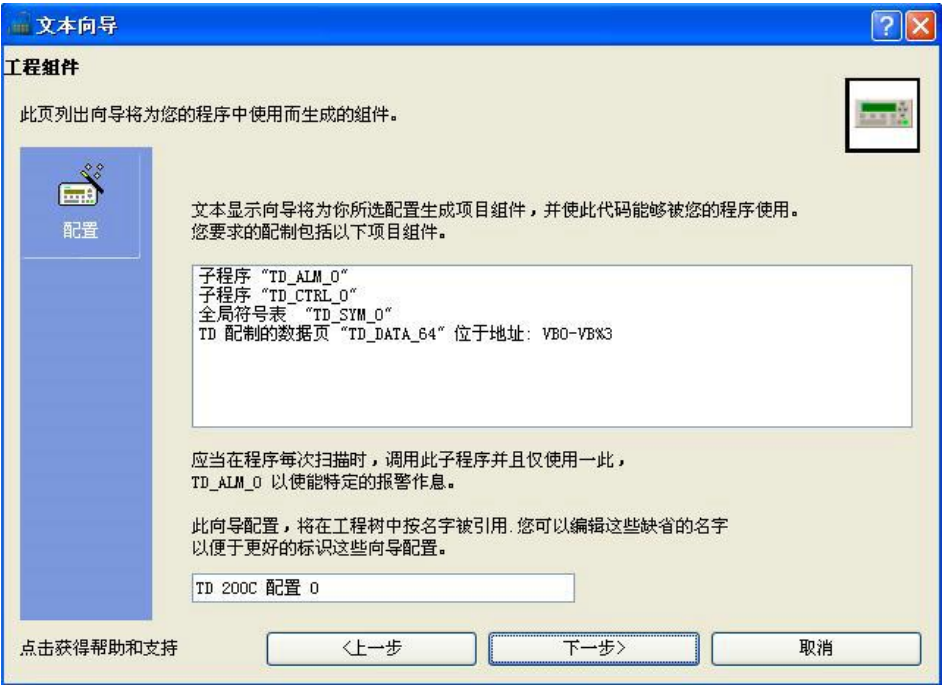
嵌入数据

- a) 将光标放在内的正确位置，然后点击“嵌入数据”按钮。
- b) 选择数据格式和显示格式。
- c) 输入小数点右边要显示的位数。

当心：嵌入数据值右端的字符具有该数值在 TD 显示中的锚固点的作用。在 TD 显示上的信息中，数据值始终以该锚固点为基准向右对齐。随着数据值的加大，它将在锚固点的左边使用更多的空间，并且可能开始使用信息文字已占用的空间。请确定在文字结尾处和该锚固点之间为该数据值的预期范围留出足够空间。

- d) 选择用户是否必须确认收到该信息。
- e) 选择是否允许用户编辑此数据。（如果您在第 3 步中未使能“密码保护”，则可使用“上一步”按钮返回至该屏幕，然后更改您的选择。使用“上一步”按钮不会丢失在前面的屏幕上已做出的选择。）
- f) 如果允许编辑，请选择“用户编辑数据”是否应受密码保护。此外，请选择当信息被使能后，该数据是否应变成可编辑数据。

<第 9 步> 生成项目组件



文本显示向导会为您选定的配置（程序块和数据块）生成项目组件，并令此代码可供您的程序使用。您必须下载 TD 配置块（数据块）、系统块及程序块至 CTSC-200 CPU，才能更新此配置。

TD 项目组件

TD_CTRL_x（控制 TD 操作）这个功能用于监视和控制 TD 设备的操作，在您的程序中应只调用一次，并且应在每个程序扫描时调用。

TD_ALM_x（使能报警）如果 TD 配置定义了报警，那么向导将会生成这个功能，用于使能特定的报警。

TD_SYM_x（全局符号表）包含与 TD 按钮、报警及 PLC 数据关联的符号。

TD_DATA_x（数据块页）用于存储文本显示向导配置的 V 存储区数据。

项目	使能报警指令		控制 TD 操作指令
功能块图			
符号	EN	ALM_EN	EN
变量类型	IN	IN_OUT	IN
数据类型	BOOL	BOOL	BOOL
注释		希望使用的报警使能位	用 SM0.0 调用,监视和控制 TD 设备的操作

5.5.5 配方向导



提示

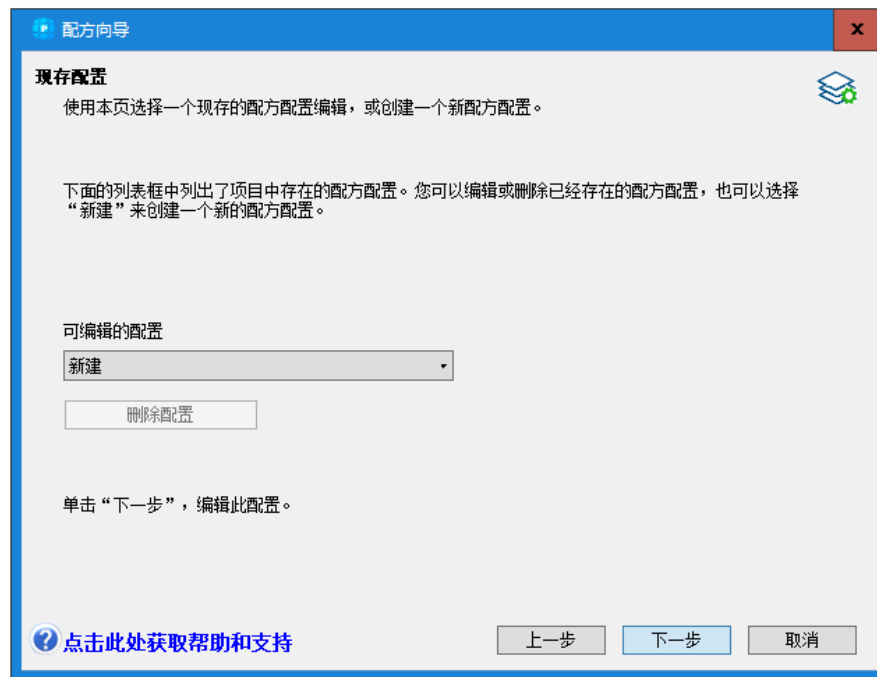
- CTH3 系列 PLC 采用 U 盘作为配方的存储设备。
- “配方向导”可以创建项目代码，用于在 64K 字节或 256K 字节永久性内存（EEPROM）盒中存储和获取配方数据。配方向导用于在存储卡中组织和载入自动化程序配方。将配方数据值转移到存储卡内，可以解放否则必需用来存储配方数据的 V 内存地址。新的配方定义和配方数值分配可在配方向导中直接键入。对配方数据的后续改动可通过重新运行配方向导或采用 RCPx_WRITE 功能的编程来完成。

欲使用配方向导，请遵循以下基本步骤：

- 1、选择菜单命令“工具”→“配方向导”或点击项目树中的向导结点，在右边的工作区域双击配方向导图标，启动配方向导。并随后打开此向导或某现有配置。
- 2、点击“下一步”按钮前往配方向导的每个新页。
- 3、完成向导中的步骤后，点击“完成”按钮。

具体操作步骤如下：

<第 1 步> 编辑现有配置



点击配置下拉列表，从中选一个现有或新的配方定义。您最多可以有四个不同的配方定义。每个配方定义中可保存数百个配方。

如果不存在现有配方配置，配方向导将由第 2 步开始。

<第 2 步> 配方定义

配方向导

配方定义

使用本页为此配方定义数据域。

为此配方配置定义数据域。每个域将成为您项目中的一个符号名。您必须为每个域指定一个数据类型以及一个在新建配方时将使用的默认值。

	域名	数据类型	默认值	注释
1	AAA	BYTE	0	第一个
2	BBB	BOOL	2#0	
3	CCC	WORD	0	
4	DDD	INT	+0	
5	EEE	DWORD	0	
6	FFF	DINT	+0	
7	GGG	REAL	0.0	最后一个
8				

单击“下一步”，为此配置编辑配方。

点击此处获取帮助和支持

上一步

下一步

取消

用鼠标点击以选取一个表数据单元，然后编辑该域。键入一个符号名作为域名。由下拉式列表中选取一个数据类型。为每个配方数据值键入一个默认值和注释。使用定义配方中全部数据项目所必需的表行数。您最多可以有四个不同的配方定义。

<第 3 步> 创建和编辑配方数据值

配方向导

创建和编辑配方

每个配方是与各配方——对应的一套数据值。新添加的配方会自动初始化为您指定的默认值。配方名称将成为您项目中的一个符号。

目前已为此配置定义了 3 个配方。

	域名	数据类型	DEF0_RCP2	DEF0_RCP1	DEF0_RCP0
1	AAA	BYTE	0	0	0
2	BBB	BOOL	2#0	2#0	2#0
3	CCC	WORD	0	0	0
4	DDD	INT	+0	+0	+0
5	EEE	DWORD	0	0	0
6	FFF	DINT	+0	+0	+0
7	GGG	REAL	0.0	0.0	0.0

1

增加配方

删除

恢复默认值

点击此处获取帮助和支持

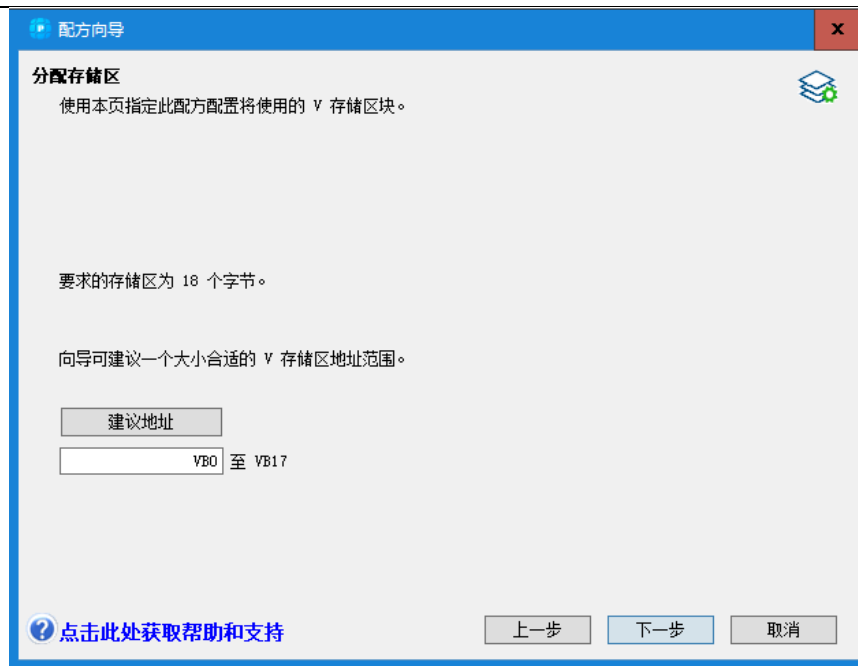
上一步

下一步

取消

双击列标题，重新命名该配方，然后编辑默认值以创建新配方。点击配方数据的任何位置或注释列启用“新”按钮，然后点击此“新”按钮插入新的配方。重复此过程即可基于当前配方定义插入、重新命名、编辑所有配方。

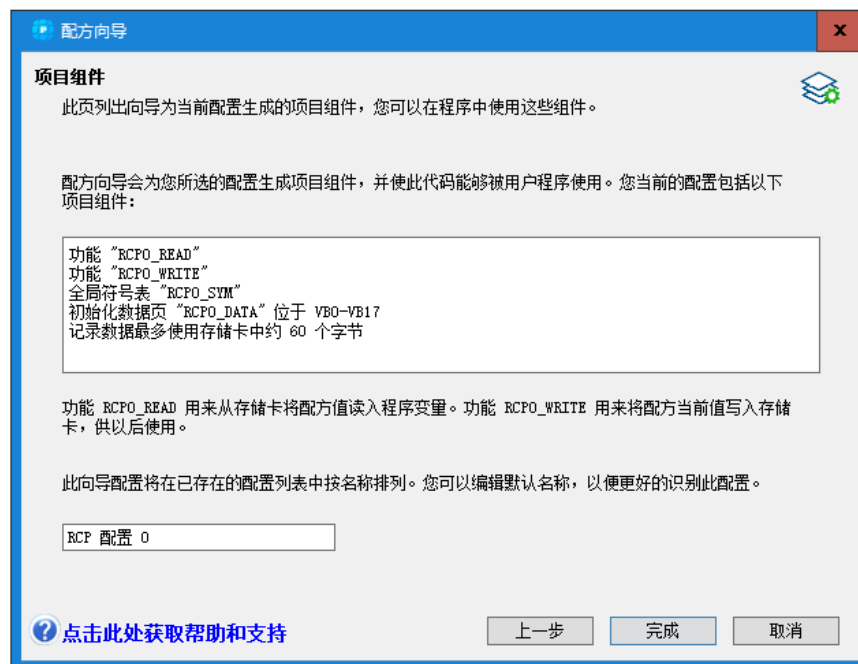
<第 4 步> 分配 V 内存



此向导将分配一个“数据块”标记以在 V 内存中存储配方配置。在数据块中指定您希望安放该配置的起始地址。您可以选择想要存放该配置块的 V 内存地址；如果您希望向导建议正确大小的未使用内存块地址，则点击“建议地址”按钮。

配置块的大小根据您在向导中所作的具体选择不同而异。

<第 5 步> 生成程序代码



配方向导会为您的配方（符号表、数据块、功能、存储卡数据）生成项目组件，并令您的程序可以使用这些组件。您必须将带有配方向导配置的项目下载至 CPU 才能加以使用。如果某项目有配方配置，则下载窗口的默认设置为“配方”选项已被选中。

配方下载选项必须选中，才能将配方数据载入存储卡。

您的程序能够由用配方数据符号名指明的 V 内存地址读取配方数据。每执行一次 RCPx_READ 就会将一个配方从存储卡复制到 V 内存。每执行一次 RCPx_WRITE 就会将一个配方从 V 内存复制到存储卡。

配方向导通过生成下列项目组件而创建存取存储卡接口的程序：

- 配方符号表，每个配方定义一个，用于关联配方符号名、V 内存地址、配方法注解
- 新的数据块标记，每个配方定义一个，用于为每个配方定义分配 V 内存地址和数值。您的程序可使用对这些 V 内存地址的符号名援引来存取配方数据。
- 功能 RCPx_READ 将指定的配方数据值从存储卡复制到 V 内存。RCPx_READ 功能用于将不同的配方载入 V 内存或刷新当前配方的 V 内存数值。
- 功能 RCPx_WRITE 将指定的配方数据值从 V 内存复制到存储卡。如果程序在 V 内存配方数据值中写入改动，并且希望保存为永久配方更改，请使用 RCPx_WRITE。

下表列出了配方 RCPx_READ/RCPx_WRITE 指令定义及相关说明

项目	读配方指令			写配方指令		
功能块图						
符号	EN	Rcp	Error	EN	Rcp	Error
变量类型	IN	IN	OUT	IN	IN	OUT
数据类型	BOOL	WORD	BYTE	BOOL	WORD	BYTE
注释		从存储卡读 取的配方	读取配方 的错误码		写入存储 卡的配方	写配方的 错误状态

针对上述指令，错误代码和错误状态说明如下：

错误代码	描述
0	无错误
132	存储卡访问失败

5.5.6 数据记录向导



提示

- CTH3 系列 PLC 采用 U 盘作为数据记录的存储设备。
- “数据归档向导”可以创建项目代码，用于在 64K 字节或 256K 字节电可擦可编程只读存储器（EEPROM）存储卡中存储数据。数据归档向导可帮助您在存储卡内存储过程测量数据。将过程数据的存放转移到存储卡内，可以解放否则必需用来存储此数据的 V 内存地址。记录定义会被输入到数据归档向导内。对记录定义的改动可通过重新运行数据归档向导来完成。

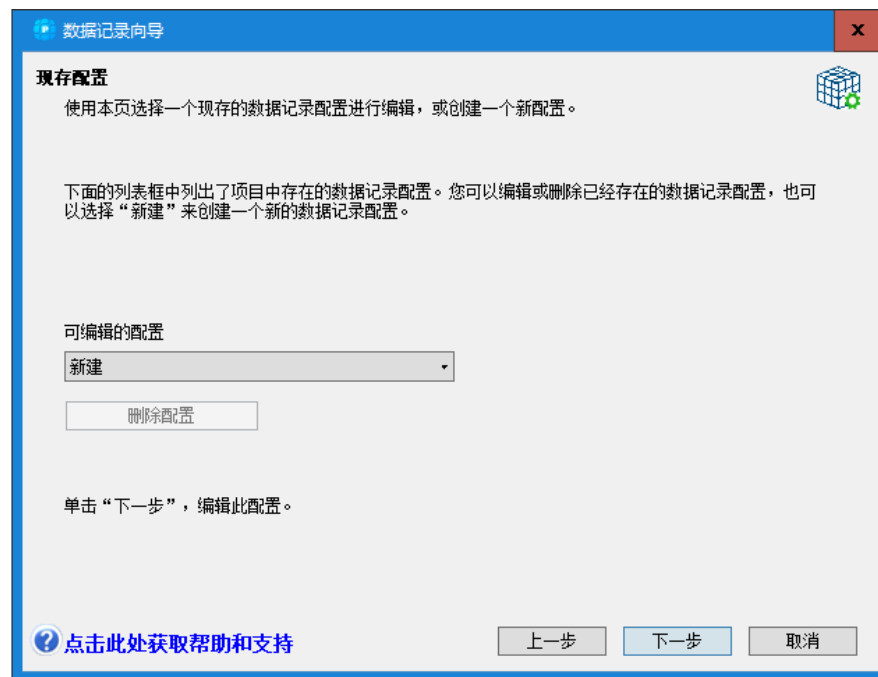
欲使用数据归档向导，请遵循以下基本步骤：

- 1、选择菜单命令“工具”→“数据归档向导”或点击项目树中的向导结点，在右边的工作区域双击数据记录向导图标, 启动数据记录向导。并随后打开此向导或某现有配置。
- 2、单击“下一步”按钮前往数据归档向导的每个新页。

3、完成向导中的步骤后，单击“完成”按钮。

具体操作步骤如下：

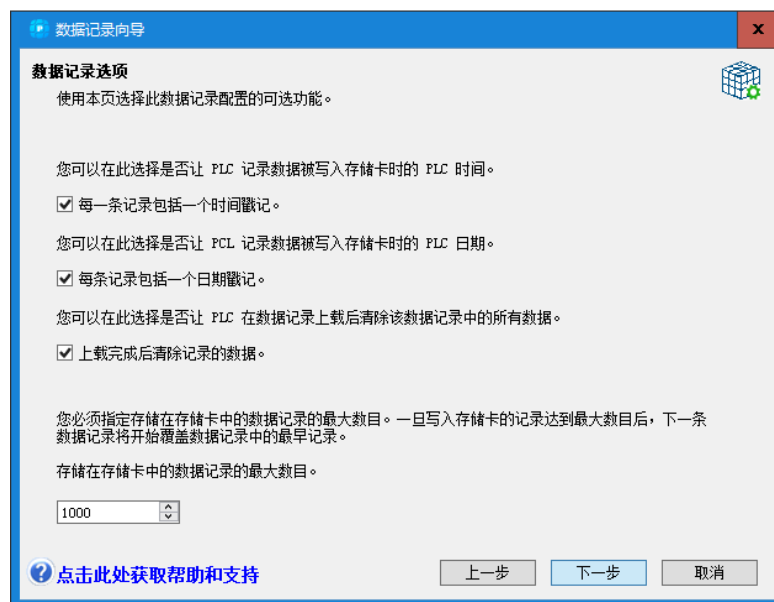
<第 1 步> 编辑现有配置



单击配置下拉列表，从中选一个现有或新的数据归档定义。您最多可以有四个不同的记录定义。

如果不存在现有数据归档配置，数据归档向导将由第 2 步开始。

<第 2 步> 数据归档选项



您可以让每份记录包括一个时间戳记。在选中这个选项后，每当用户程序发出数据归档写入命令，CPU 就会在每份记录中自动包括一个整数时间戳记。选中此选项会令数据归档记录定义的最大长度缩短 3 个字节。

每份数据归档记录可添加一个日期戳记。在选中这个选项后，每当用户程序发出数据归档写入命令，CPU 就会在每份记录中自动包括一个整数日期戳记。选中此选项会令数据归档记录定义的最大长度缩短 3 个字节。

您可选择每当数据归档被上载时，清除该数据归档中的全部记录。

数据归档的形式为循环缓冲区，您必须指定存储在存储卡中数据归档记录的最大数目。此数目为 CPU 中的一个 WORD (字) 值，且必须选取范围 (1-65535)。此选项的默认值为 1000。在您下载具有数据归档配置的项目时，存储卡空间会被保留以存储最大数目的记录。

<第 3 步> 数据记录定义

数据记录向导

数据记录定义

使用这一页为数据记录定义数据域。

为此数据记录指定数据域。每个域将成为您项目中的一个符号名。您必须为每个域指定一个数据类型。存储在存储卡中的每个数据记录最多有203个字节，包括3个开销字节，3个日期戳记字节（如果已经使能），3个时间戳记字节（如果已经使能）以及其他用于存储数据值的字节。

此数据记录定义的大小为 15 个字节，还有 188 个字节可供使用。

	域名	数据类型	注释
1	Year	BYTE	年
2	Month	BYTE	月
3	Day	BYTE	日
4	Hour	BYTE	时
5	Minute	BYTE	分
6	Second	BYTE	秒
7			

单击“下一步”，为此数据记录配置分配存储区。

点击此处获取帮助和支持

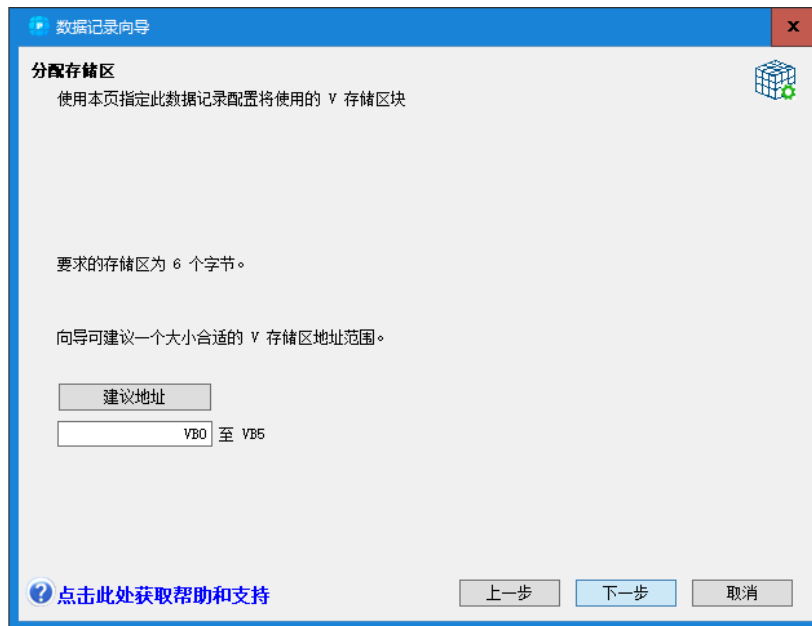
上一步

下一步

取消

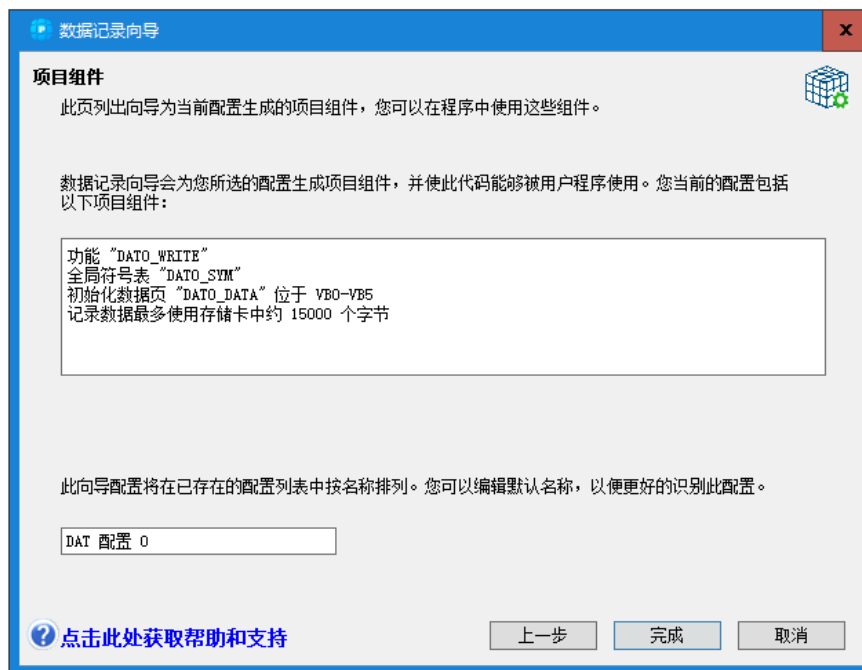
用鼠标点击以选取一个表数据单元，然后编辑该域。键入一个符号名作为域名。由下拉式列表选取一个数据类型。为记录中的每个项目键入一条注释。使用定义全部数据项目所必需的表行数。您最多可以有四个不同的数据归档定义。

<第 4 步> 分配 V 内存



此向导将分配一个"数据块"标记以在 V 内存中存储数据归档配置。在数据块中指定您希望安放该配置的起始地址。您可以选择 V 内存地址，如果您希望向导建议正确大小的未使用内存块地址，则单击"建议地址"按钮。配置块的大小根据您在向导中所作的具体选择不同而异。

<第 5 步> 生成程序代码



数据归档向导会为您的归档数据定义 (符号表、数据块、功能、存储卡数据) 生成项目部件，并令您的程序可以使用这些部件。

您必须将带有数据归档向导配置的项目下载至 CTSC-200 才能加以使用。如果某项目有数据归档配置，则下载窗口的默认设置为“数据归档配置”选项已被选中。

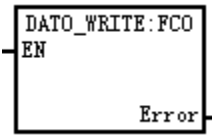
数据归档下载选项必须选中，以为数据归档数据准备存储卡。

您的程序能够将新的过程数据移动到由该记录符号名指明 V 内存地址，执行一次 DATx_WRITE 功能以取得全部数据值的快照并在存储卡内保存一个新记录。

数据归档向导通过生成下列项目部件而创建存取存储卡接口的程序：

- 数据归档符号表，每个数据归档定义一个，用于关联数据归档符号名、V 内存地址、数据归档注解
- 新的数据块标记，每个数据归档定义一个，用于为每个记录定义分配 V 内存地址。您的程序使用这些 V 内存地址来积累当前归档记录。
- 功能 DATx_WRITE 将指定的记录从 V 内存复制到存储卡。每执行一次 DATx_WRITE 就会为已储存在存储卡内的归档数据添加一个新的数据记录。

下表列出了配方 RCPx_READ/RCPx_WRITE 指令定义及相关说明

数据记录向导指令			
指令名称		功能块图	
DAT0_WRITE			
符号	变量类型	数据类型	注释
EN	IN	BOOL	使用 SM0.0 调用
Error	OUT	BYTE	写数据记录的错误状态

针对上述指令，错误代码和错误状态说明如下：

错误代码	描述
0	无错误
132	存储卡访问失败

5.5.7

PID 指令向导



提示

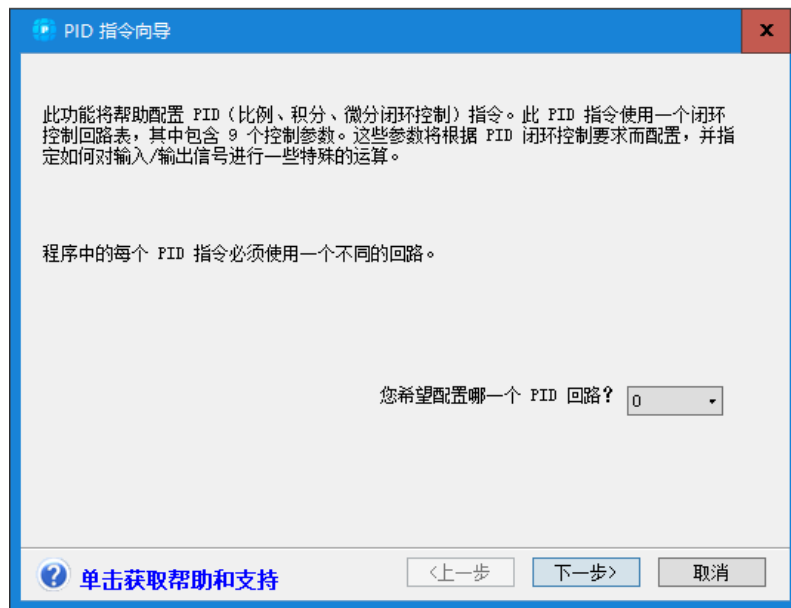
- 在使用 PID 向导之前，程序必须被编译并位于符号编址模式。如果您尚未编译，向导会在 PID 配置过程开始提示您进行编译。
- CTH-200 指令向导的 PID 功能可用于简化 PID 操作配置。向导向您询问初始化选项，然后为指定配置生成程序代码和数据块代码。

欲使用 PID 向导，请遵循以下基本步骤：

- 1、选择菜单命令“工具”→“PID 向导”或点击项目树中的向导结点，在右边的工作区域双击 PID 向导图标，启动 PID 向导。并随后打开此向导或某现有配置。
- 2、单击“下一步”按钮前往 PID 向导的每个新页。
- 3、完成向导中的步骤后，单击“完成”按钮。

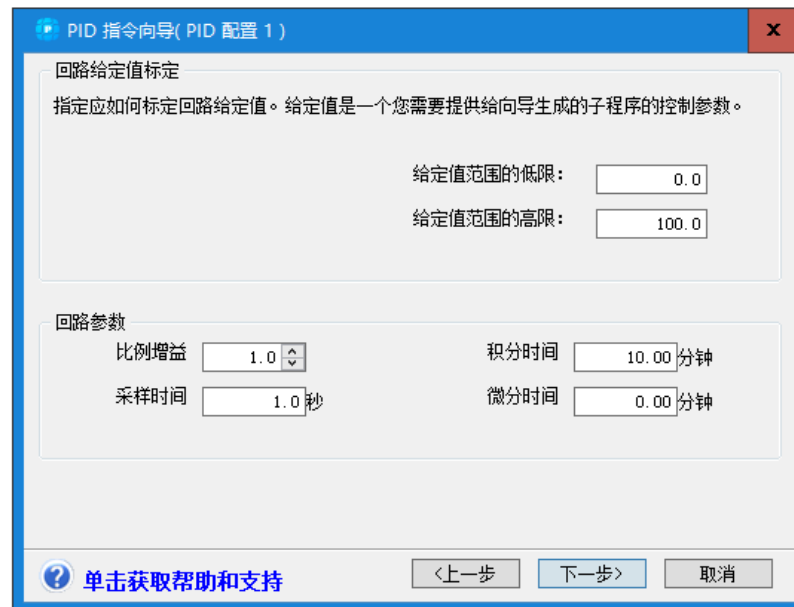
具体操作步骤如下：

<第 1 步> 指定回路号码



<注意> 如果项目包含使用 MagicWork PLC 建立的现有 PID 配置，您必须在继续执行第 1 步之前选择编辑其中一个现有配置或建立一个新配置。您指定配置哪一个 PID 回路。

<第 2 步> 设置回路参数



<注意> 参数表地址的符号名已经由向导指定。PID 向导生成的代码使用相对于参数表中的地址的偏移量建立操作数。如果您为参数表地址建立了符号名，然后又改变为该符号指定的地址，由 PID 向导生成的代码则不再能够正确执行。回路给定是您为向导生成子程序提供的一个参数。

设置选项如下：

- 1) 指定回路给定 (SP) 应当如何标定。为“范围低限”和“范围高限”选择任何实数。默认值是 0.0 和 100.0 之间的一个实数。
- 2) 指定下列回路参数：
 - 比例增益 - 积分时间
 - 微分时间 - 采样时间

要了解回路参数如何影响 PID 计算性能，请参阅 PID 计算。

<第 3 步> 回路输入和输出选项

PID 指令向导 (PID 配置 1)

回路输入选项
指定如何标定回路过程变量 (PV)。回路 PV 是一个您需要提供给向导生成子程序的控制参数。

标定	过程变量	回路给定值
单极性	范围低限: 0	0.0
☐ 使用 20% 的偏移量	范围高限: 32000	100.0

回路输出选项
指定如何标定回路输出。回路输出是一个您需要提供给向导生成子程序的控制参数。

输出类型	标定	范围低限	范围高限
模拟量	单极性	0	32000
	☐ 使用 20% 的偏移量		

单击获取帮助和支持 <上一步 下一步> 取消

回路过程变量 (PV) 是您为向导生成的子程序指定的一个参数。

向导向您询问以下回路输入选项：

1) 指定回路过程变量 (PV) 应当如何标定。您可以选择：

- 单极性（可编辑，默认范围 0 至 32000）
- 双极性（可编辑，默认范围 -32000 至 32000）
- 20% 偏移量（设置范围 6400 至 32000，不可变更）

<注意> 环路设定值 (SP) 的下限必须对应于过程变量 (PV) 的下限，

环路设定值的上限必须对应于过程变量的上限，以便 PID 算法能正确按比例缩放。

您还必须输入以下回路输出选项：

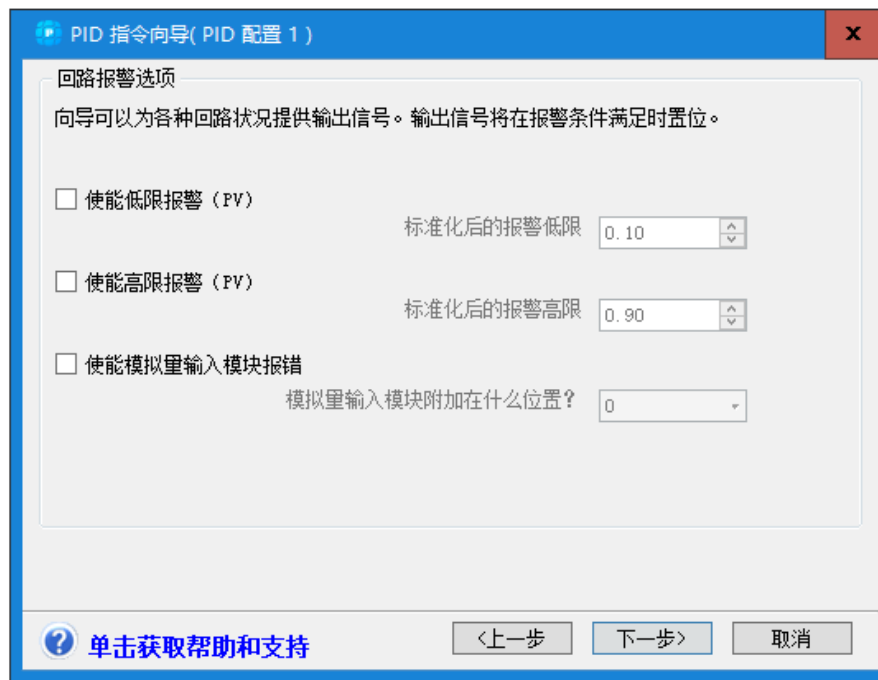
2) 指定回路输出应当如何标定。您可以选择：

- 输出类型（模拟量或数字量）

<注意> 如果您选择配置数字量输出类型，则必须以秒为单位输入“占空比周期”。

- 标定（单极、双击或 20% 偏移量）

<第 4 步> 回路报警选项

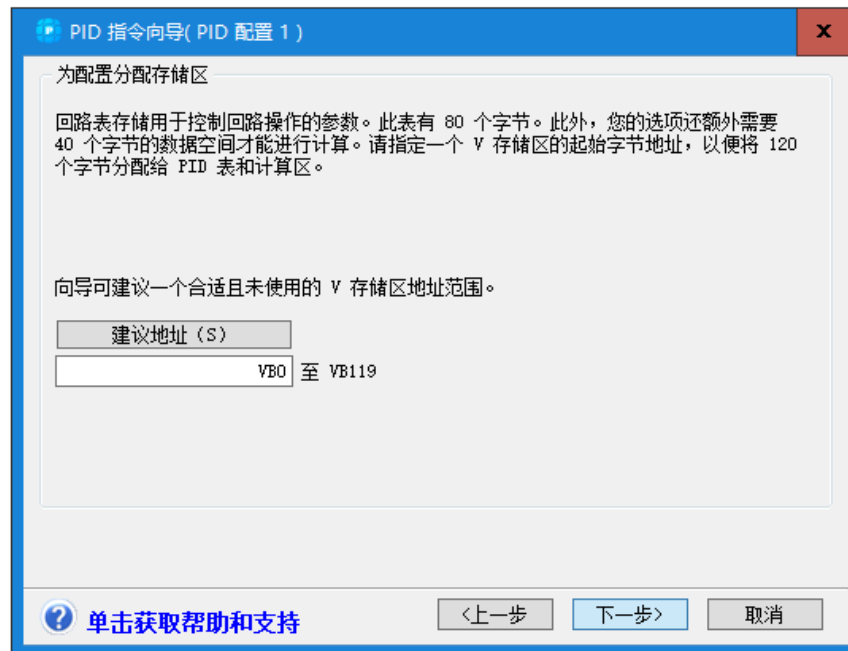


该向导为各种回路条件提供输出。当达到报警条件时，输出被置位。

指定您希望使用报警输入的哪些条件：

- 使能低限报警（PV），并在 0.0 到报警高限之间设置标准化的报警低限
- 使能高限报警（PV），并在报警低限和 1.0 之间设置标准化的报警高限
- 使能模拟量输入模块错误报警，并指定输入模块附加在 PLC 上的位置

<第 5 步> 为计算指定存储区域



PID 指令使用 V 存储区中的一个 36 个字节的参数表，存储用于控制回路操作的参数。PID 计算还要求一个“暂存区”，用于存储临时结果。您需要指定该计算区开始的 V 存储区字节地址。

您还可以选择增加 PID 的手动控制。位于手动模式时，PID 计算不执行，回路输出不改变。

<注意> 当 PID 位于手动模式时，输出应当通过向“手动输出”参数写入一个标准化数值（0.00 至 1.00）的方法控制输出，而不是用直接改变输出的方法控制输出。这样会在 PID 返回自动模式时提供无扰动转换。

<第 6 步> 指定初始化子程序和中断程序

<注意> 如果项目包含一个激活 PID 配置，已经建立的中断程序名被设为只读。因为项目中的所有配置共享一个公用中断程序，项目中增加的任何新配置不得改变公用中断程序的名称。向导为初始化子程序和中断程序指定了默认名称。您可以编辑默认名称。

<第 7 步> 生成代码

该屏幕显示 PID 向导生成的 POU 列表，并对如何把它们集成到您的用户程序作出简要说明。回答这些询问后，点击“完成”，CTH-200 指令向导将为您指定的配置生成程序代码和数据块代码。由向导建立的子程序和中断程序成为项目的一部分。要在程序中使能该配置，每次扫描周期时，

使用 SM0.0 从主程序块调用该子程序。该代码配置 PID 0。该子程序初始化 PID 控制逻辑使用的变量，并启动 PID 中断“PID_EXE”程序。根据 PID 采样时间循环调用 PID 中断程序。

<注意> 具有 PID 向导配置的项目不直接使用标准 PID 指令。如果您使用 PID 向导配置，您的程序必须使用 CALL PIDxINIT 来激活 PID 向导子程序。

5.5.8 PTO 指令向导

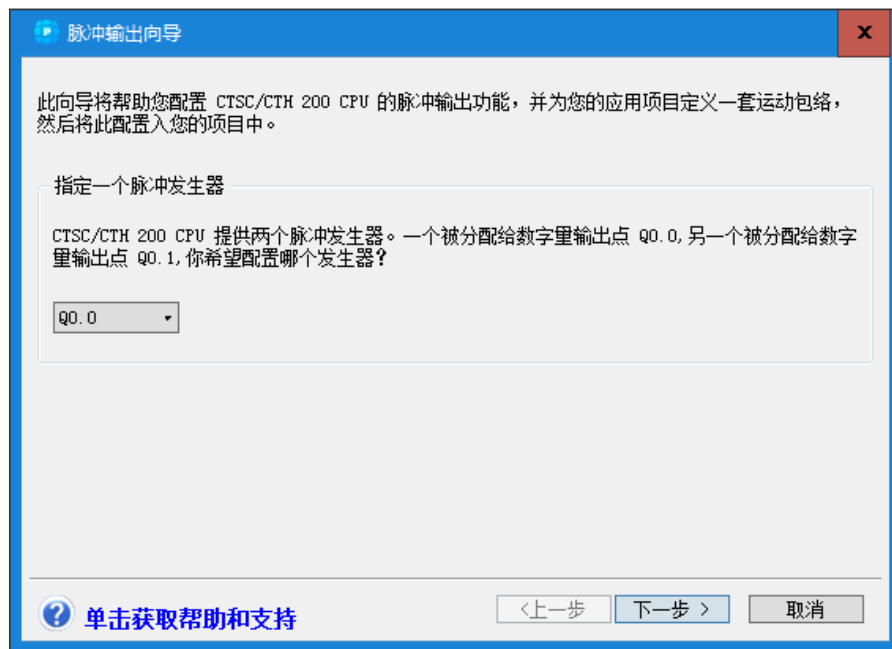
注意：本向导仅限用于第二代 CPU。

欲使用 PTO/PWM 指令向导，请遵循以下基本步骤：

- 1、选择菜单命令“工具”→“PTO/PWM 指令向导”或点击项目树中的向导结点，在右边的工作区域双击 PTO/PWM 指令向导图标，启动 PTO/PWM 指令向导。并随后打开此向导或某现有配置。
- 2、点击“下一步”按钮前往 PTO/PWM 指令向导的每个新页。
- 3、完成向导中的步骤后，点击“完成”按钮。

具体操作步骤如下：

<第 1 步> 指定一个脉冲发生器

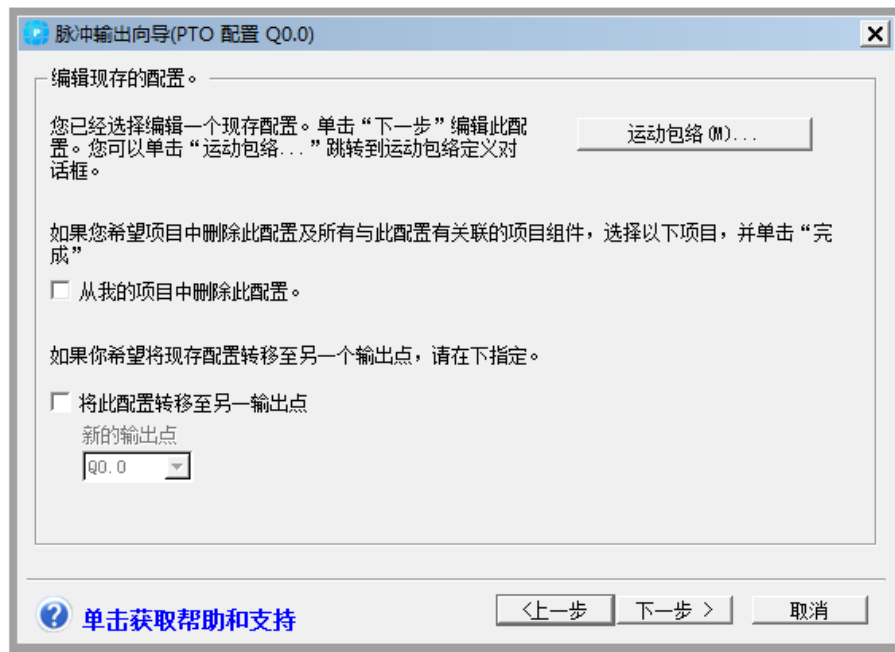


CTH-200 PLC 有两个脉冲发生器：

- 一个指定给数字输出点 Q0.0
- 一个指定给数字输出点 Q0.1

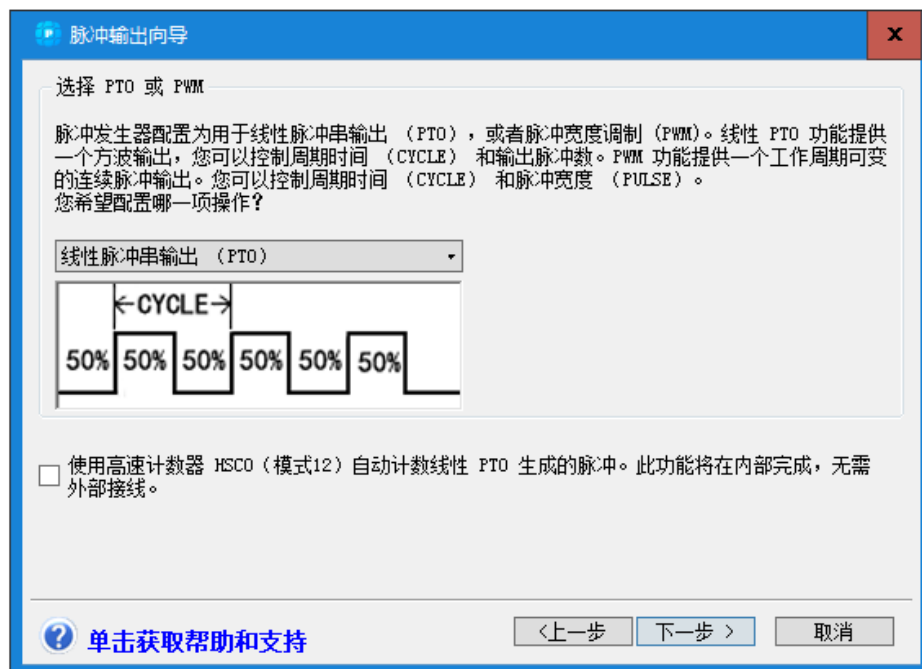
指定您希望配置的发生器。

<第 2 步> 编辑现有 PTO 或 PWM 配置



如果项目中已有一个配置，您可以从项目中删除该配置，或者将现有配置移至另一个脉冲发生器。如果项目中没有配置，则继续执行第 3 步。

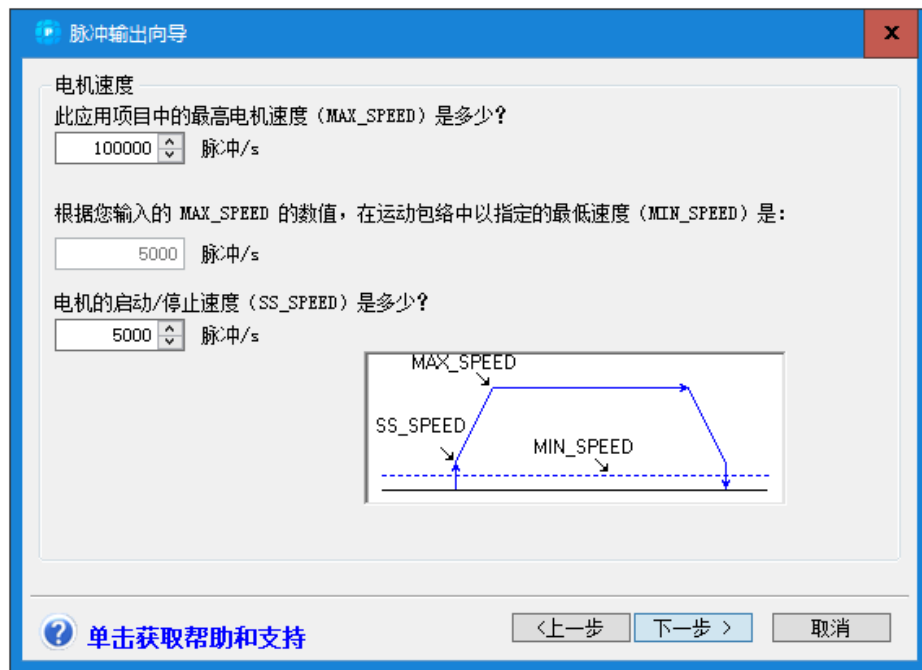
〈第 3 步〉选择 PTO 或 PWM，并选择时间基准



选择为脉冲串输出 (PTO) 或脉冲宽度调制 (PWM) 配置脉冲发生器。

- 就 PTO 模式而言，您可以启用高速计数器，计算输出脉冲数目。
- 就 PWM 模式而言，请为周期时间和脉冲宽度选择一个时间基准（微秒或毫秒）。

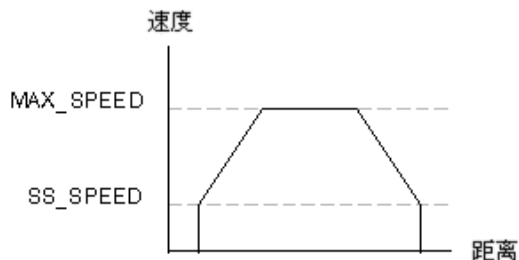
〈第 4 步〉指定电机速度



为您的应用指定最高速度 (MAX_SPEED) 和开始 / 停止速度 (SS_SPEED)。

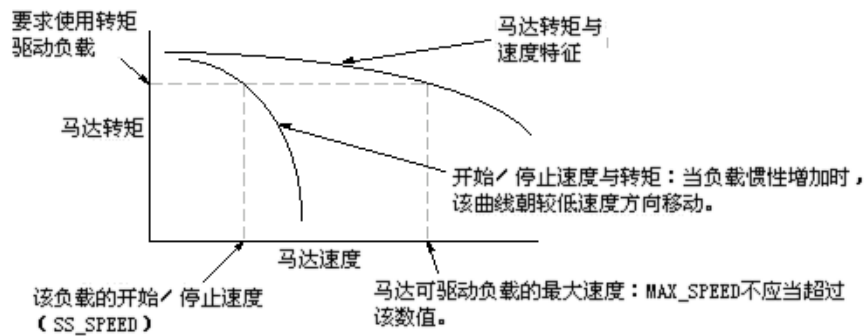
- **MAX_SPEED:** 在电机扭矩能力范围内输入您的应用的最佳工作速度。驱动负载所需的转矩由摩擦力、惯性和加速 / 减速时间决定。位置控制向导会计算和显示由位控模块为指定的 MAX_SPEED 所能够控制的最低速度。
- **SS_SPEED:** 在电机的能力范围内输入一个数值，以低速驱动负载。如果 SS_SPEED 数值过低，电机和负载可能会在运动开始和结束时颤动或跳动。如果 SS_SPEED 数值过高，电机可能在启动时丧失脉冲，并且在尝试停止时负载可能过度驱动电机。

<注意> MIN_SPEED 值由计算得出。您不能在此域中输入其他数值。



电机数据单有指定电机和给定负载开始 / 停止（或拉入 / 拉出）速度的不同方法。通常，有用的 SS_SPEED 数值是 MAX_SPEED 数值的 5% 至 15%。SS_SPEED 数值必须大于由您对 MAX_SPEED 的规定所显示的最低速度。

为了帮助您就您的应用选择正确的速度，请参阅电机的数据单。下面的数字显示一条典型的电机扭矩 / 速度曲线。

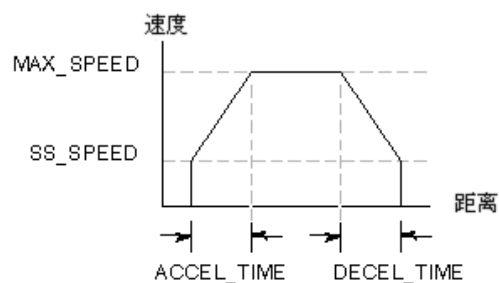


<第 5 步> 设置加速和减速时间

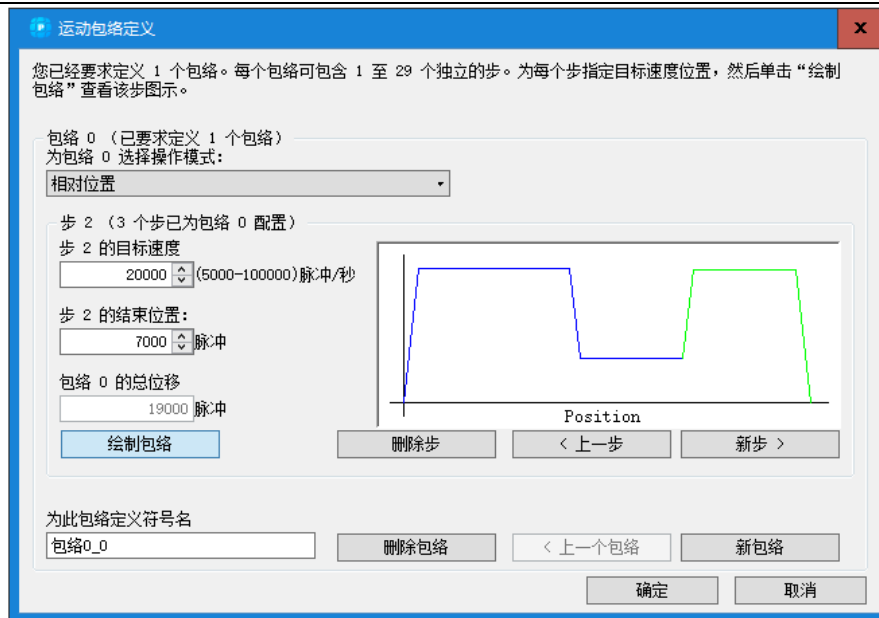
以毫秒 (ms) 为单位指定下列时间:

- **ACCEL_TIME**: 电机从 SS_SPEED 加速至 MAX_SPEED 所需要的时间
默认值 = 1000 ms
- **DECEL_TIME**: 电机从 MAX_SPEED 减速至 SS_SPEED 所需要的时间
默认值 = 1000 ms

加速时间和减速时间的默认设置均为 1 秒。通常, 电机所需时间不到 1 秒。电机加速和减速时间由反复试验决定。您应当在开始时用位置控制向导输入一个较大的数值。当测试您的应用时, 请根据要求调整有关数值。请通过逐渐减少时间直至电机开始停顿的方法, 优化该应用的设置。



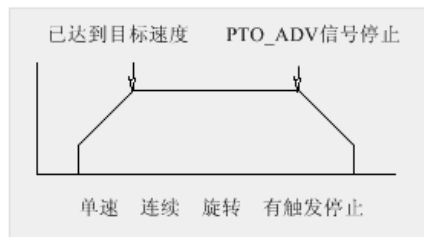
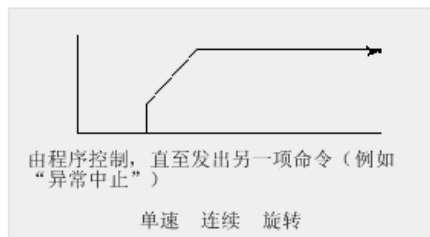
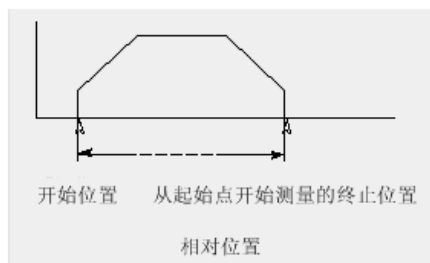
<第 6 步> 定义每个已配置的轮廓



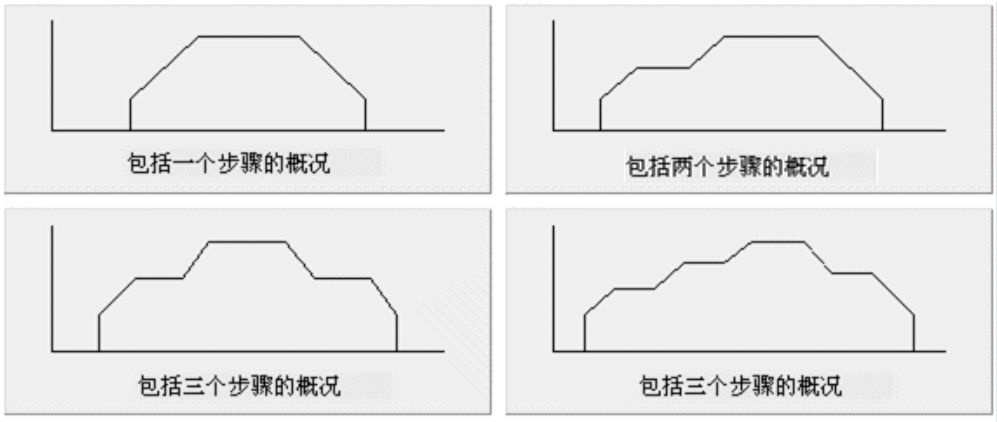
您选定要配置的每个轮廓会被指定一个符号名。在此定义的符号名是您在 PTOx_RUN 子程序中输入的参数。

针对每个轮廓，您必须选取下列参数：

- **操作模式：**您根据操作模式（相对位置或单速连续旋转）配置此轮廓。如果您选择单速连续旋转，您必须输入一个目标速度。请启用 PTO_ADV 复选框和一个停止距离，为此轮廓发起一个停止。下图显示了不同的操作模式。



- **轮廓的步骤：**步骤是工具移动的固定距离，包括在加速时间和减速时间所走过的距离。每个轮廓最多可有 4 个单独步骤。您为每个步骤指定目标速度和结束位置。如果您有不止一个步骤，请单击“新步骤”按钮，然后为轮廓的每个步骤输入此信息。下图显示了四个可能的轮廓，但还可能其他组合。



您只需点击"步骤绘图"按钮，即可检视根据位置控制向导的计算做出的此步骤之图形表示。这令您可轻易且互动地查看和编辑每个步骤。

位置控制向导也令您可通过在定义轮廓时输入一个符号名，为每个轮廓定义符号名。

在完成轮廓的配置后，您可以将它保存至配置。

您的所有配置和轮廓信息都存储于数据块 V 内存赋值的 PTOx_data 页内。

〈第 7 步〉 设定轮廓数据的起始 V 内存地址

脉冲输出向导

为配置分配存储区

线性 PTO 配置要求23个字节的 V 存储区。另外您还配置了 1 个运动包络。全部配置需要的 V 存储区总是 70 个字节。请输入起始地址，以便使用 V 存储区的 70 个字节保存配置和包络信息。向导可以建议一个大小合适的 V 存储区地址范围。

向导可以建议一个大小合适的 V 存储区地址范围。

建议地址 (S)

VB14 至 VB83

单击获取帮助和支持

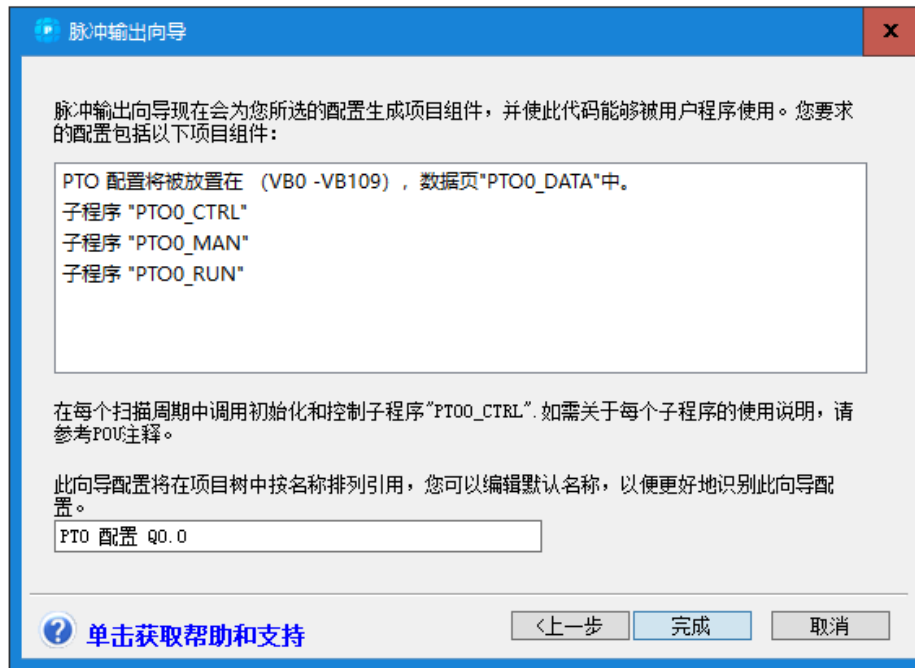
<上一步

下一步 >

取消

PTO 向导在 V 内存中以受保护的数据块页形式生成 PTO 轮廓模板。PWM 向导不使用 V 内存模板。

〈第 8 步〉 生成项目代码



PTO / PWM 组件

PTO 配置中可能存在下列项目组件。在下面的组件名中，'x'将用为脉冲发生而选择的输出位取代：'0'代表 Q0.0，'1'代表 Q0.1。此外，向导将通过附加字符 '_Z' ,确保这些组件名在用户的项目中是唯一的；在此 Z 是一个基于零的指标。Z 将不断递增，直至生成唯一的名称。请点击组件名，取得详细说明。

● PTO 配置的项目组件

PTOx_CTRL（初始化和控制 PTO 操作）应在每次程序扫描时（于 EN 输入处）启用，并且由子程序调用而在您的程序中只执行一次。

PTOx_RUN（运行 PTO 轮廓）用于执行特定运动轮廓。当用户定义了一个或多个运动轮廓后，此子程序将为 PTO 向导配置生成。

PTOx_MAN（手动 PTO 模式）子程序可用来在程序控制下指挥脉冲发生。

PTOx_LDPOS（载入位置）子程序用于将某当前位置参数载入 PTO 操作。当用户选取了脉冲计数的高速计数器时，PTO 向导会创建此子程序。

PTOx_ADV（前进）PTOx_ADV 子程序会停止当前的连续运动轮廓并按照在向导轮廓定义中规定的脉冲数前进。如果您已在位置控制向导中指定了至少一个启用 PTOx_ADV 选项的单速连续旋转，此子程序就会得到创建。

PTOx_SYM（全局符号表）用于 PTO 向导配置中使用的变量。

PTOx_Data（数据块页）由向导配置使用的 V 内存数据。此数据包含参数表和运动轮廓定义。

● PWM 配置的项目组件

PWMx_RUN（手动 PWM 模式）用于在程序控制下执行 PMW 的子程序。

5.5.9 HSC 向导



提示

- 在使用 HSC 向导之前，必须编译程序。如果您尚未编译程序，向导在 HSC 配置过程开始时提示您编译程序。
- HSC 向导向您询问初始化选项，然后为指定的配置生成程序代码和数据块代码。

MagicWorks PLC 提供 LAD 或 STL 中的“高速计数器”指令（HSC）用于控制速率超过 PLC 扫描速率的事件。HSC 操作十分灵活，有多种确定计数器行为方式的初始化选项。要以适当的方式配置 HSC，您需要在 SM 存储区中设置几个参数，配置 HSC 为一个具体的操作模式，将中断程序连接在具体的计数器事件上，最后使能计数器操作。

欲使用 HSC 指令向导，请遵循以下基本步骤：

- 1、选择菜单命令“工具”→“HSC 指令向导”或点击项目树中的向导结点，在右边的工作区域双击 HSC 指令向导图标，启动 HSC 指令向导。并随后打开此向导或某现有配置。
- 2、单击“下一步”按钮前往 HSC 指令向导的每个新页。
- 3、完成向导中的步骤后，单击“完成”按钮。

具体操作步骤如下：

<第 1 步> 选择计数器类型和操作模式

HSC指令向导

选择的 PLC 类型提供一下高速计数器。不同的计数器最多可支持12种操作模式。

您希望配置哪个计数器？

HCO

高速计数器的操作模式决定使用哪些时钟脉冲和计数方向输入，以及是否使用启动和复位输入。您希望选择哪种操作模式？

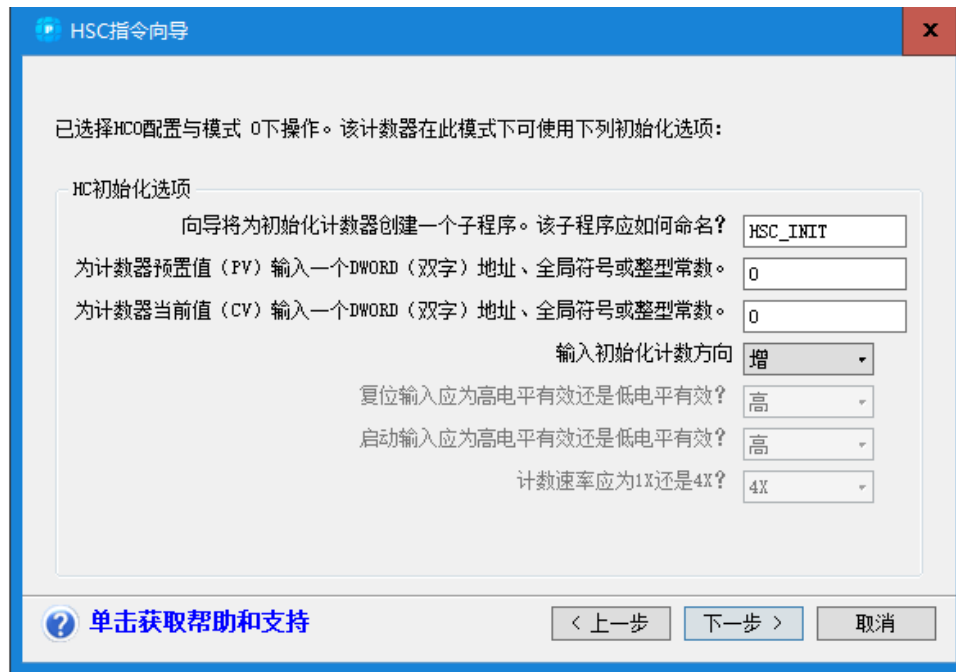
模式	模式说明
模式 0	带内部方向控制的单相增/减计数器。无启动输入；无复位输入。

单击获取帮助和支持

< 上一步 下一步 > 取消

从列表框（根据 PLC）选择须配置的计数器。从列表框选择操作模式，您选择的计数器决定它可用的模式。

<第 2 步> 指导初始参数

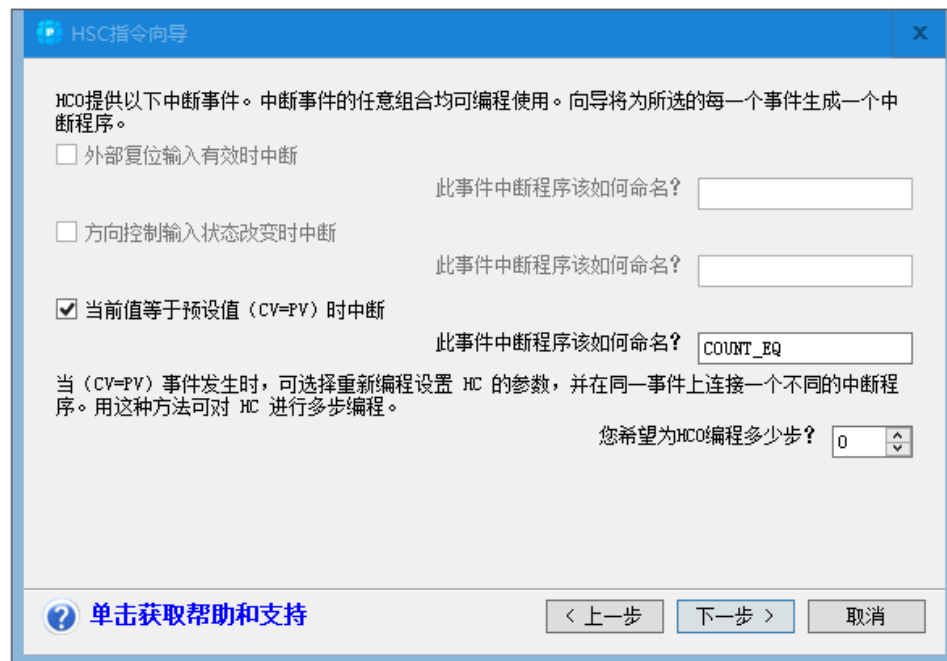


向导为子程序指定一个默认名称。您可以指定一个不同的名称，但请勿使用现有子程序名称。

为计数器 CV 和 PV 指定一个双字地址、全局符号或整数常数。

<注意> 如果您为 PV 或 CV 输入一个未定义的符号，则在定义符号之前无法执行第 3 步。向导会提示您在“定义符号”对话框中定义符号。当您为 PV 和 CV 选择地址或符号时，更新 PV 和 CV 的选项被禁止。您必须修改生成的代码，才能修改或传递数值。根据定义，这意味着代码是不可重新装载的。指定初始计数方向。

<第 3 步> 程序中断事件/编程多步操作

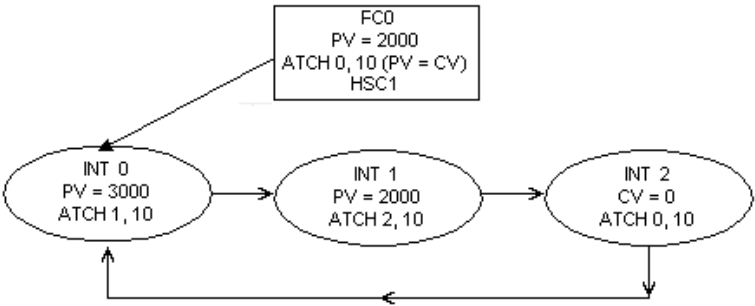


您的计数器类型和操作模式选择决定可用的中断事件。当您选择对当前数值等于预设值事件 (CV=PV) 进行编程时，向导允许您指定多步计数器操作。

为了理解多步 HSC 操作，请参考下列内容：



在已编程的中断程序中，您可以选择将相同的中断事件重新连接至另一个中断程序。这相当于在计数器操作中增加了一个“步”。高速计数器的常见应用包括跟随某些循环事件的操作（例如旋转杆）。因此，您可能需要为计数器编程几个步，其中每个步改变 HSC 的一些动态参数，并将一个不同的中断程序连接至相同的中断事件。最后一个步骤中可以重新连接原始中断程序（构成一个循环），当然也可能并非如此。下例说明一个简化的、三个步骤的 HSC 应用。再后面是每个步的说明。



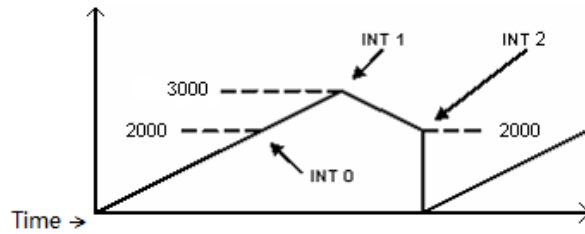
FC 0 该子程序包含计数器初始化。计数器的当前值被指定为 0（CV=0）。计数器的预置值被指定为 2000（PV=2000）。计数方向为 UP（向上）。事件 10（HSC0 CV=PV）被连接至 INT0，计数器被启动。

INT 0 当计数器达到第一个预置值 2000 时，执行 INT0。计数器预置值被更改为 3000，方向不变。事件 10（HSC0 CV=PV）被重新连接至 INT1，计数器被重新启动。

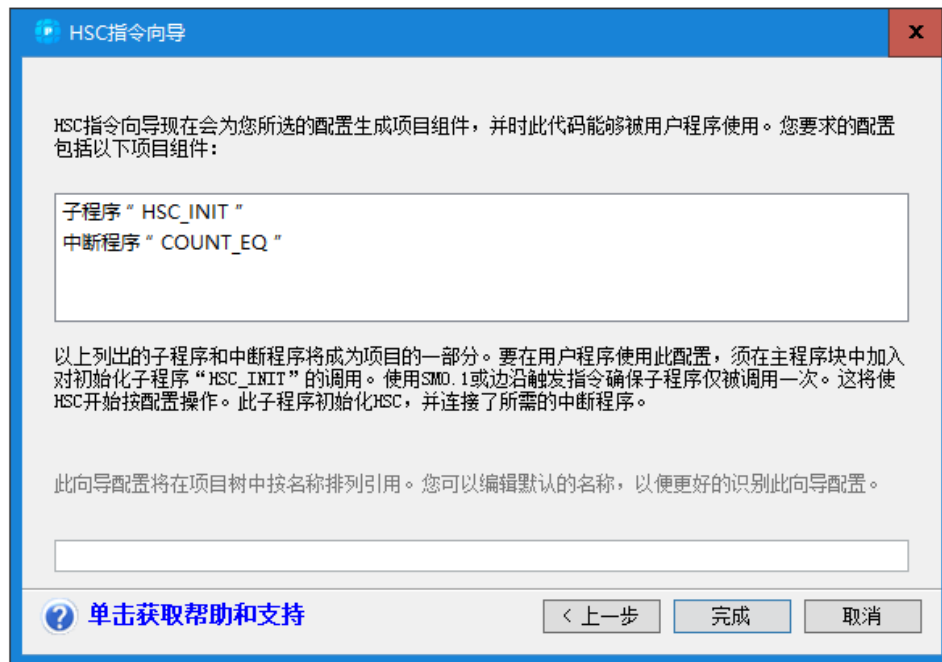
INT 1 计数器再次达到预置值（3000）时，执行 INT1。此时，我们将预置值更改成 2000（PV=2000），将计数方向更改为 DOWN（减），将 INT1 连接至事件 10（HSC0 CV=PV），并重新启动计数器。

INT 2 当计数器向下计数至预置值 2000 时，执行 INT2。此时，我们将当前值设为 0（CV=0），并将计数方向更改为 UP（增）。事件 10 被重新连接至 INT0，至此完成了计数器操作的循环。

下图显示计数器当前值随时间的变化。每个（CV=PV）中断事件均标有该事件调用的 INT 程序。



<第 4 步> 生成代码



完成 HSC 参数配置后，您可以检查计数器使用的子程序/中断程序列表。该列表仅供参考。

点击“完成”按钮，允许向导为 HSC 生成必要的程序代码。代码包括用于高速计数器初始化的子程序。另外，为您选择编程的每一个事件生成一个中断程序。对于多步应用，则为每一个步生成一个中断程序。

<注意> 要能使计数器操作，您必须从主程序中调用包含初始化代码的子程序。使用 SM0.1 或沿触发指令确保该子程序仅被调用一次。

5.6 库的使用

MagicWorks PLC 总共有两种不同类型的库：

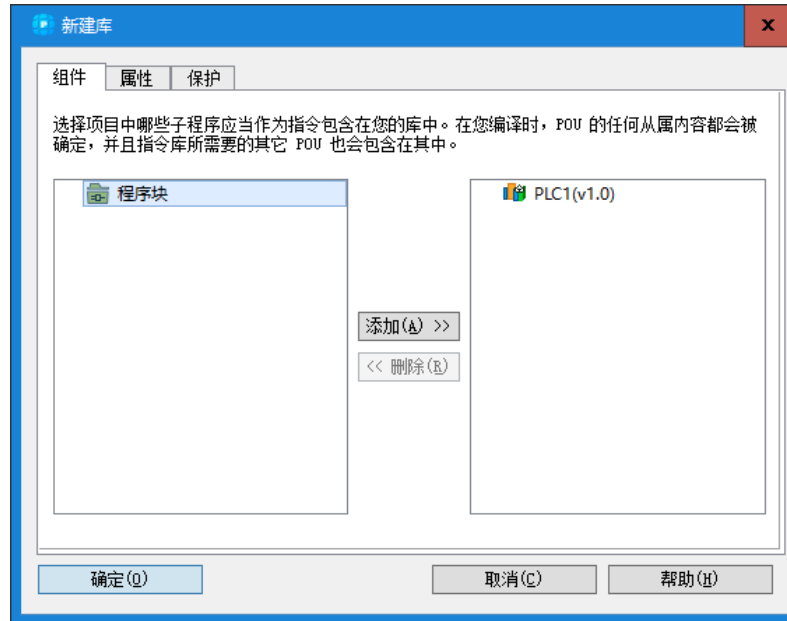
- COTRUST 公司提供的库（如：USS 库、Modbus 库、Socket 库、Motion_ctrl_lib 运控库、EM231_7WB 称重库等）
- 用户自定义的库

<备注> COTRUST 公司的库文件可从本公司网站免费下载：<http://www.co-trust.com>

新建库

用户可以从 MagicWorks PLC 项目管理器项目建立一个指令库：

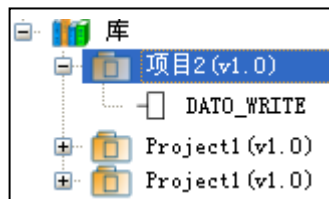
选择菜单项“文件”→“新建库”，会打开新建库对话框，如下所示：



此对话框中您可以设置以下内容：

组件：选择项目中的哪些子程序作为指令包括在库中。要包括子程序，选中左侧列表中的子程序，并点击“添加”按钮。欲删除指令，选择右侧的指令，并点击“删除”按钮。

属性：指定指令库名称、建立和存储库的路径和文件名。库名可以包含大小写和空格。一旦库被添加并使用，您在此处提供的名称以后会在指令树中显示。



安装时，会建立一个默认 Lib（库）目录。您应当在此处放置指令，以便在添加指令库和使用时方便查找。您还可以指定建立的库的版本。

保护：指定库中的代码是否应当受密码保护，防止被查看和编辑。选择“是”复选框，并为库输入密码（四个字符）。

添加/删除库

本操作用于在系统中添加新库，或删除系统中已经添加的库。

欲在项目中增加指令库：

- 1) 选择菜单命令“文件”→“添加 / 删除库”。
- 2) 点击“添加...”，在项目中添加库。

- 3) 浏览至您保存库文件的目录。选择.ctmwl 文件，并点击“打开”。
- 4) 点击“确定”，库添加成功之后，查看一下指令树中的库节点，可以看到新库已经被添加到指令树中。

欲从项目中删除指令库：

- 1) 选择“文件”→“添加 / 删除库”菜单命令。
- 2) 此时可以看到当前存在的库列表，选择您需要删除的库。
- 3) 点击“确定”，库删除成功之后，查看一下指令树中的库节点，可以看到选择删除的库已经从指令树中移除。

调用库指令

成功添加库之后，即可使用程序编辑窗口的指令树中的库指令，调用库指令的步骤如下：

- 1) 在程序编辑器窗口打开所需的 POU，并滚动至您希望插入调用的网络。
- 2) 在指令树中，以双击的方式打开“库”文件夹。您可以从指令树拖动适当的库指令，并将指令放置在程序编辑器中的适当网络单元格中，或将光标放在程序编辑器的单元格上，然后双击指令树中的库指令。
- 3) 在程序中编辑调用指令参数，并为每个参数指定有效的操作数。

当您调用库指令时，可能会在项目中增加一个或多个 POU。库指令实际上是包含在库中的子程序。当您调用库指令时，执行该指令的子程序会自动从库复制至 MagicWorks PLC 项目。当您继续调用时，就不会再出现此种情形，因为所要求的子程序已经成为项目的一部分。

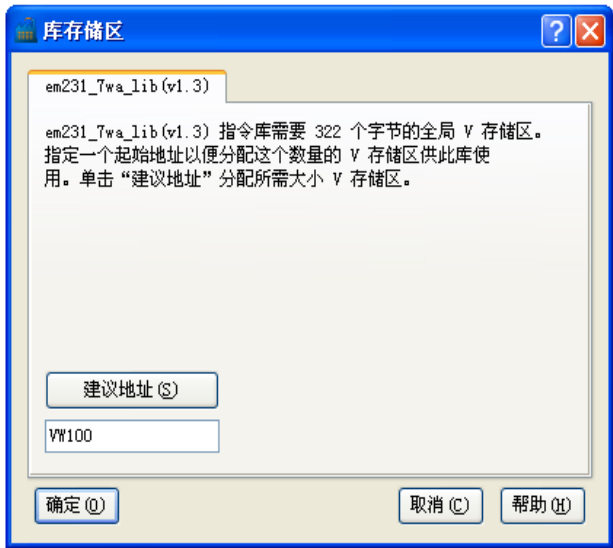
您会注意到已经自动在项目中增加一个新 POU (EM231_7WA_LIB)。该 POU 在项目树中限制，在程序编辑器中有一个标签。您放置的指令现在成为对该子程序的调用。

库存储区

如果库要求使用部分 V 内存，您为库可以使用的内存块指定一个起始地址。没有必要在调用其中一条库指令后立即为库分配内存，但您必须分配库内存，才能成功地编译使用库的项目。为当前项目使用的指令库分配存储区：

- 1) 在程序块中调用需要分配存储区的库指令
- 2) 选择菜单命令“文件”>“库存储区”

您建立的库不要求使用 V 内存，对话框会进行说明。如果库要求使用 V 内存，您必须为库的使用指定一个地址。以下图示以称重库 EM231_7WA_LIB 为例，要求使用 V 存储区即可点击“建议地址”按钮，输入库需要使用的地址。



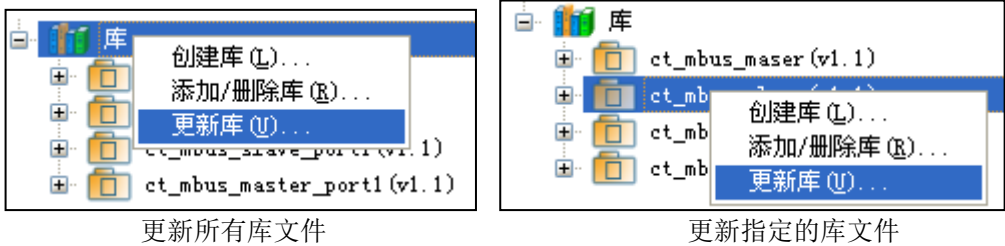
<注释>

- 该操作只有在项目使用指令库时才被启用。
- 需要在成功编译使用库的项目之前为库分配存储区。

在线更新库

若想将库文件更新至最新版本，参考如下步骤：

打开程序块，在程序编辑器窗口用鼠标右键点击指令树的“库”条目并选择“更新库”，即可在线更新 COTRUST 所有库文件。或者，选中某个库文件然后选择“更新库”操作，即可将该库更新为最新版本。



<备注> “创建库”和“添加/删除库”请参考上文的操作。

5.7 编译功能

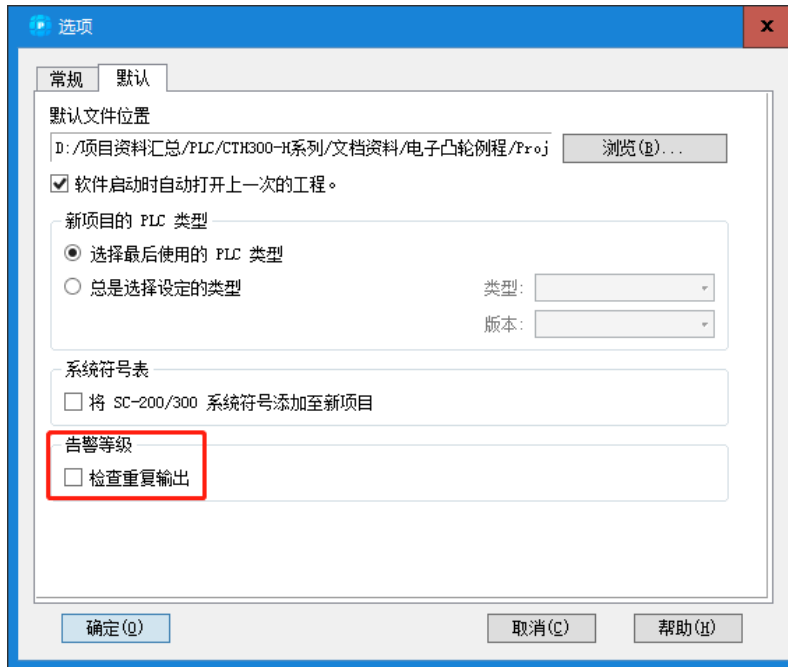
编译过程中支持程序语法检查和错误定位。STL 编译错误定位到行，LAD 编译错误定位到行和列，数据块编辑编译错误定位到行。

双击编译错误信息自动跳转到错误位置的功能，STL 和 LAD 跳转到错误网络，数据块跳转到错误行。

LAD 和 STL 程序可以相互转换，程序块、系统块、数据块都能针对站点型号编译成对应 PLC 支持的目标码。

从 PLC 上载的目标码能够还原成程序块、系统块、数据块组成一个完整工程。

MagicWorksPLC V2.18 及以上版本支持编译时检查重复输出，在项目管理器界面选择“工具”→“选项”，在弹出的“选项”窗口的默认选项下可勾选“检查重复输出”，选中表示检查，默认不检查。



项目管理器界面的编译（PLC 菜单）

在项目管理器界面，可使用以下几种方法来启动项目编译：

- 选择菜单命令“PLC”→“编译”或点击“编译”图标, 编译当前选中并修改过的块（程序块、数据块、EasyCAN 组态和硬件组态）。
- 选择菜单命令“PLC”→“全部编译”或点击“全部编译”图标, 编译全部项目组件（程序块、数据块、系统块、EasyCAN 组态和硬件组态，保密库只编译修改过的程序块）。
- 选择菜单命令“PLC”→“重新编译程序块”，清理并编译所有编译所有程序块。在使用编译遇到不明错误或是改动较多时建议使用此功能。

编辑错误全部显示在 MagicWorks PLC Manager 位于底部的输出窗口中。

<注意> 编译前 MagicWorks PLC 会自动保存您选择编译的块。

程序编辑器界面的编译（文件菜单）

- 在程序编辑器窗口，选择菜单命令“PLC”→“编译”或点击“编译”图标, 编译当前工程当前 PLC 站点下修改过的所有程序块，编译信息显示在底部的输出窗口。
- 在程序编辑器窗口，选择菜单命令“PLC”→“重新编译程序块”，清理并编译所有编译所有程序块。在使用编译遇到不明错误或是改动较多时建议使用此功能。

<注意> 编译前程序编辑器会自动保存您当前工程 PLC 下所有程序块的修改。

初始化 V 内存数据块界面的编译（文件菜单）

在初始化 V 内存数据块编辑窗口，选择菜单命令“PLC”→“编译”或点击“编译”图标, 编译当前工程当前 PLC 站点下的所有数据块，编译信息显示在底部的输出窗口。

<注意> 编译前初始化 V 内存数据块编辑器会自动保存您当前工程 PLC 下所有数据块的修改。

EasyCAN 组态软件的编译（文件菜单）

在 EasyCAN 组态软件窗口，选择菜单命令“PLC”→“编译”或点击“编译”图标，编译当前工程的 PLC 站点下的 EasyCAN 组态，编译信息以弹出对话框的形式显示。

<注意> 编译前 EasyCAN 组态软件会自动保存您当前工程 PLC 下 EasyCAN 组态的修改。

硬件组态界面的编译（文件菜单）

在硬件组态窗口界面，选择菜单命令“PLC”→“编译”或点击“编译”图标，编译当前工程的 PLC 站点下的硬件组态，编译信息以弹出对话框的形式显示。

<注意> 编译前硬件组态编辑器会自动保存您当前工程 PLC 下硬件组态的修改。

DB 编辑器界面的编译（文件菜单）

在 DB 参数编辑器窗口，选择菜单命令“PLC”→“编译”或点击“编译”图标，编译当前工程的 PLC 站点下的所有数据块，编译信息显示在底部的输出窗口。

<注意>编译前 DB 编辑器会自动保存您当前工程 PLC 下所有数据块的修改。

6 通信设置

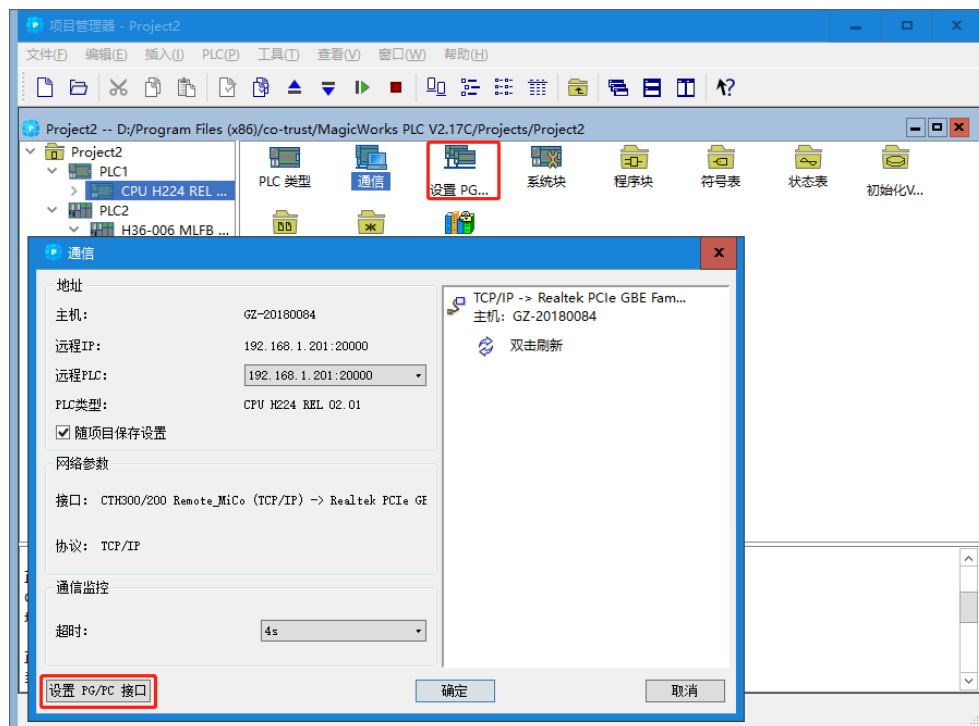
本章概要

- ☐ 通过串口与 PLC 建立 PPI 通信
- ☐ 通过 CP6451 网关与串口 PLC 建立通信
- ☐ 通过 CP6453 网关与串口 PLC 建立通信
- ☐ 通过 CTH200/300 集成以太网口与 PLC 建立通信
- ☐ 通过 iNet 与远程 CTH200/300 建立通信
- ☐ 通过 Mico 与远程 CTH200/300 建立通信

6.1 通过串口与 PLC 建立 PPI 通信

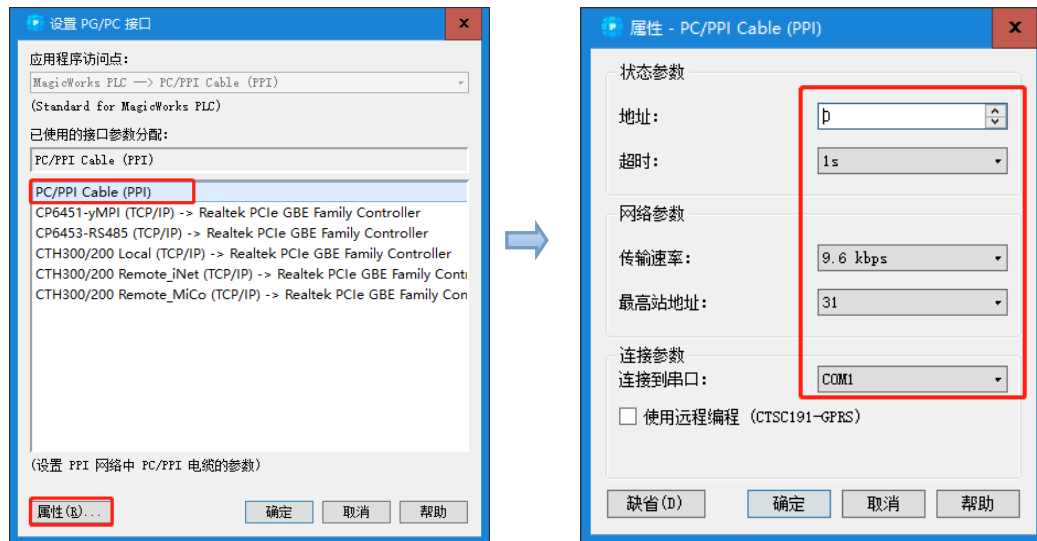
通过串口直接与 PLC 相连并通信时选此协议。连接方法：使用 PPI 电缆将 PC 与 PLC 的 PPI 通信口直连。连接过程如下：

- 1、在项目管理器界面点击设置 PG/PC 接口图标“”，或在通信界面选择“设置 PG/PC 接口”，进入设置界面。



- 2、在“设置 PG/PC 接口”界面选中“PC/PPI Cable(PPI)”，点击“属性”可调整编程设备或

PC 的 PPI 接口，使其与所使用的 PPI 网络相匹配。确保正确区分站参数和网络参数。站参数是接口的本地参数，而网络参数必须与网络中的其他站相匹配。



可以设置下列站参数和网络参数：

◆ 地址

在“状态参数”区域中选择“地址”下拉框中的内容。该内容指向已调用接口参数分配的编程设备（PG）或 PC。

在此，设置编程设备/ PC 的本地站地址。本地站地址在网络中必须唯一。在 PPI 网络中通过该地址访问编程设备/PC。

◆ 超时

“超时”表示您希望通讯驱动程序建立连接所消耗的时间。

<备注> 当网络上出现故障，例如慢响应时间误报通信超时错误时，请增加通信超时时间。

◆ 传输速率

按照您希望 MagicWorks PLC 网络上通讯的速率设置传送速率。PPI 电缆支持 9.6 kbps、19.2 kbps 和 187.5 kbps。

◆ 最高站地址

选择已组态的最高站地址。确保 PPI 网络中所有站的最高站地址都相同。

◆ 连接参数

在编程设备上或 PC 上选择要连接 PC/PPI 电缆的接口（COM 串行接口或 USB 口）。

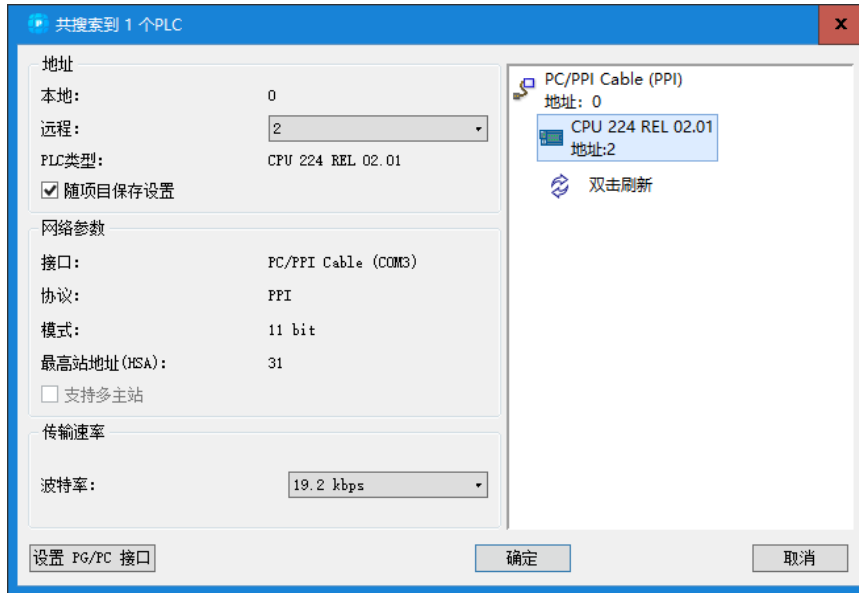
◆ 使用远程编程（CTSC191-GPRS）

- 如果您正在使用 GPRS 转 RS485 模块（CTSC191-GPRS）进行远程编程，请勾选此选项，否则有可能通信失败。选择此选项后，超时参数会自动变为默认的 8s，用户可以根据网络情况自行改变超时值。传输速率选择框会变得不可用，因为此时通信接口的波特率由 CTSC191-GPRS 的配置软件来设置，此界面上显示的波特率无意义。
- 这种模式下，只有当 CTSC191-GPRS 的波特率与 PLC 通信端口的波特率一致时，编程软件才能与 PLC 保持连接。

提示： CTSC191-GPRS 的详细使用方法请参考《MagicWorks PLC 远程编程模块使用说明》

书》，下载地址：<http://www.co-trust.com>

3、设置完成后，点击“确定”返回“通信”界面，双击“双击刷新”即可搜索到 PLC，如下图所示：



6.2 通过 CP6451 网关与串口 PLC 建立通信

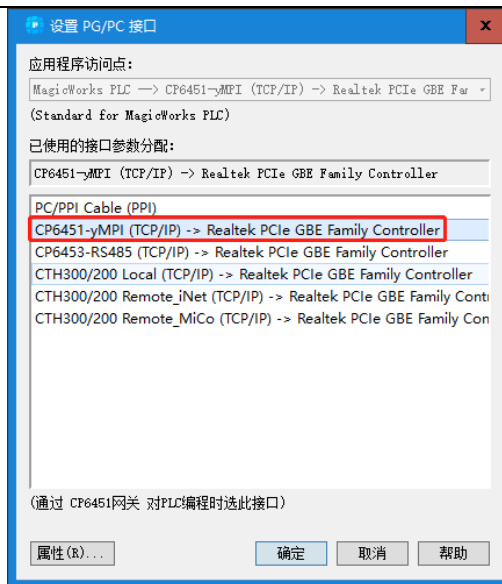
通过 CP6451 网关对 PLC 编程时选此接口（本地模式/远程模式）。

连接方法：将网关模块 CP645x-yMPI 的 RJ45 接口与 PC 或路由器连接，485 通信线连接网关模块的 RS485 接口与 PLC 的 RS485 接口，局域网或互联网。

本地模式（PC 跟 CP6451 在同一局域网）：通信对话框——TCP/IP 端口属性页面的参数应该设置成 CP6451 网关的内部 IP 和端口（出厂设置是 192.168.1.200：20000）；

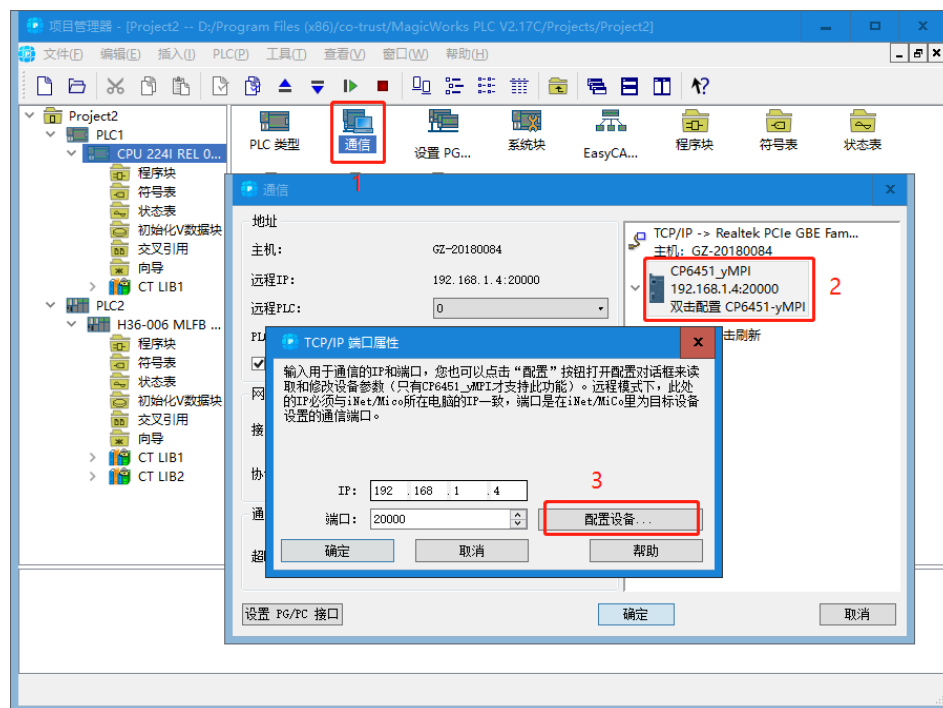
配置 CP6451-yMPI 模块

步骤 1：在“设置 PG/PC 接口”对话框中选择 CP6451-yMPI（TCP/IP）协议：



步骤 2: MagicWorks PLC 连接当前 CP6451-yMPI 网关模块

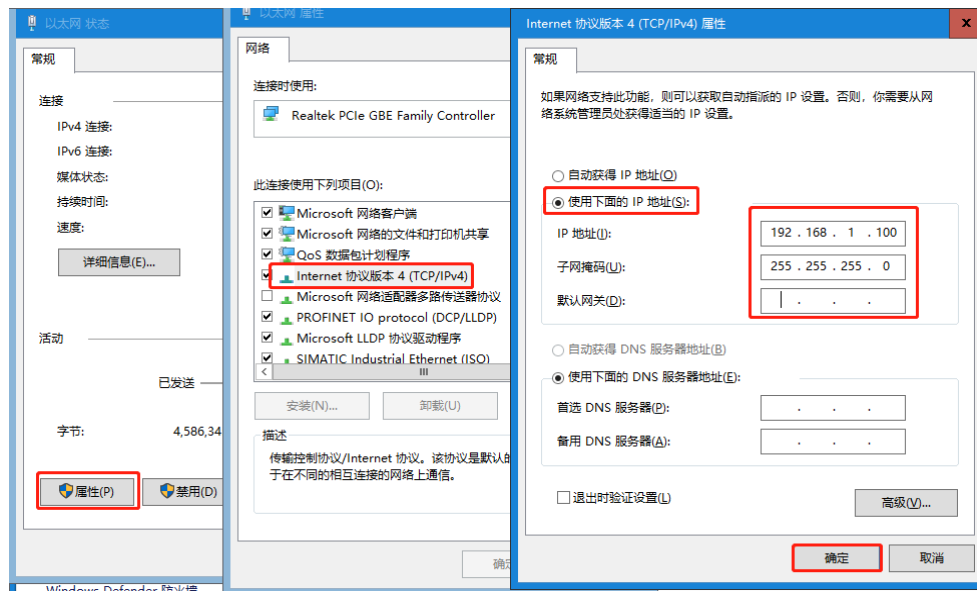
打开 MagicWorks PLC 后，展开 PLC 站点，双击“通信”→“CP6451-yMPI”→“配置设备”
打开网关配置界面，具体操作顺序参考下图中的序号标记：



提示

配置参数前需将电脑 IP 设置与 CP6451 模块的 IP 处于同一网段，CP6451 模块默认端口号为 20000，不能修改。

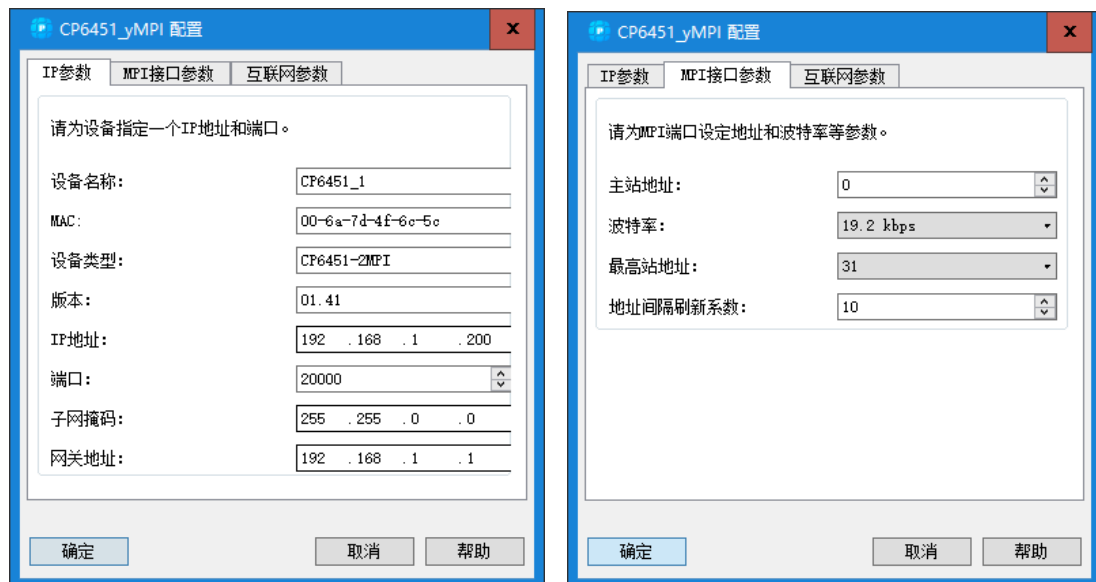
在网络设置里可按如下步骤修改电脑 IP（示例图为 Win10 操作系统）。



步骤 3: 通过以上步骤打开配置 CP6451-yMPI 网关模块参数配置界面。

1) 配置“IP 参数”

2) 配置“MPI 接口参数”



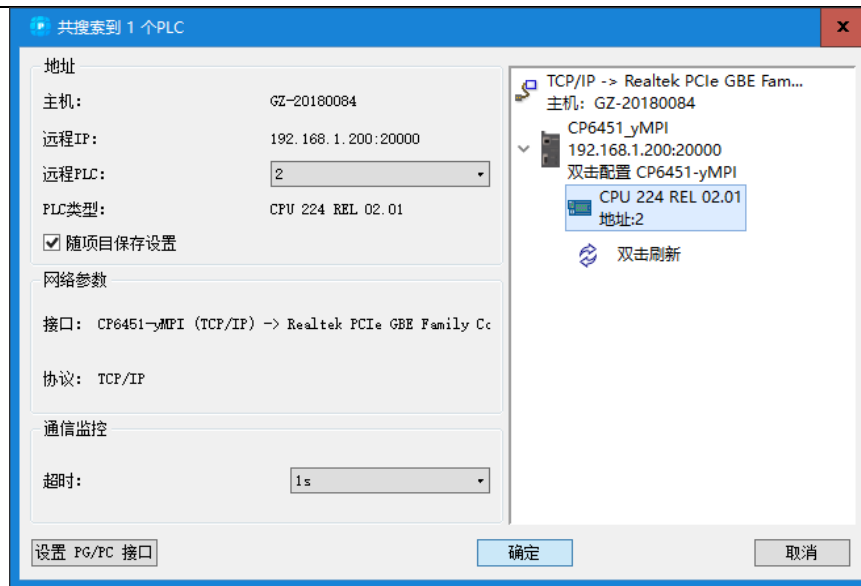
“主站地址”、“最高站地址”和“地址间隔刷新系数”建议采用默认值。

“波特率”：按照所连接 PLC 的波特率进行设置。

2) CP6451-yMPI 网关模块的 IP 参数、MPI 接口参数配置完成后，点击“确定”完成配置。

搜索 PLC

完成以上配置后，在“通信”界面双击“双击刷新”即可搜索到 PLC 进行通信，如下图所示：



6.3 通过 CP6453 网关与串口 PLC 建立通信

通过 CP6453 网关对 PLC 编程时选此接口（本地模式/远程模式）。

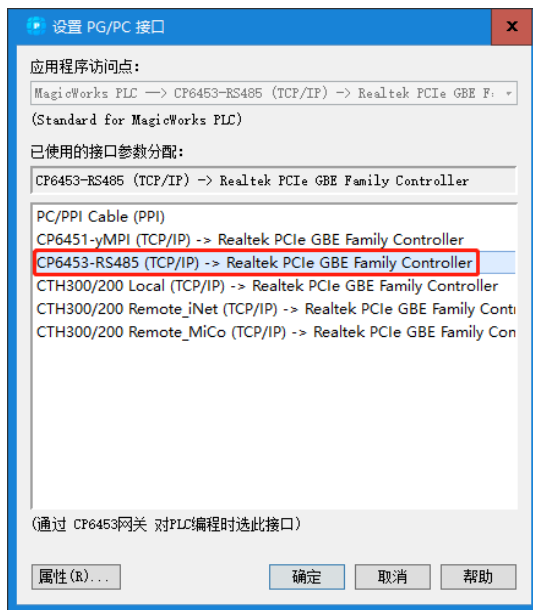
连接方法：将网关模块 CP6453-RS485 的 RJ45 接口与 PC 或路由器连接，485 通信线连接网关模块的 RS485 接口与 PLC 的 RS485 接口，局域网或互联网。

本地模式（PC 跟 CP6453 在同一局域网）：通信对话框——TCP/IP 端口属性页面的参数应该设置成 CP6453 网关的内部 IP 和端口（出厂设置是 192.168.1.201：端口固定是 3636）；

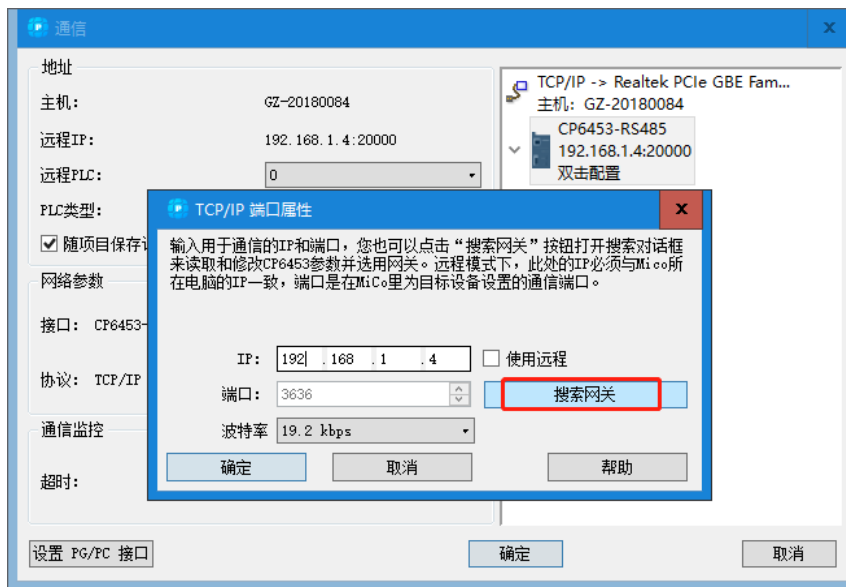
远程模式（PC 跟 CP6453 不在同一局域网，访问 MiCo 上的在线网关设备）：通信对话框——TCP/IP 端口属性页面的 IP 应该设置成 MiCo 软件所在电脑的 IP，端口改成 MiCo 软件"我的设备"列表里要访问设备的通信端口（如果 MagicWorks PLC 跟 MiCo 安装在同一台电脑，通信对话框——TCP/IP 端口属性的 IP 可以设置成 127.0.0.1）；

本地模式配置 CP6453 模块

- 1、在“设置 PG/PC 接口”界面选择 CP6453-RS485(TCP/IP)协议。



2、在通信界面双击 CP6453 设置端口属性，CP6453 模块的“IP”和“端口号”为默认，不修改，波特率与通信 PLC 保持一致，点击“搜索网关”进入网关配置界面。



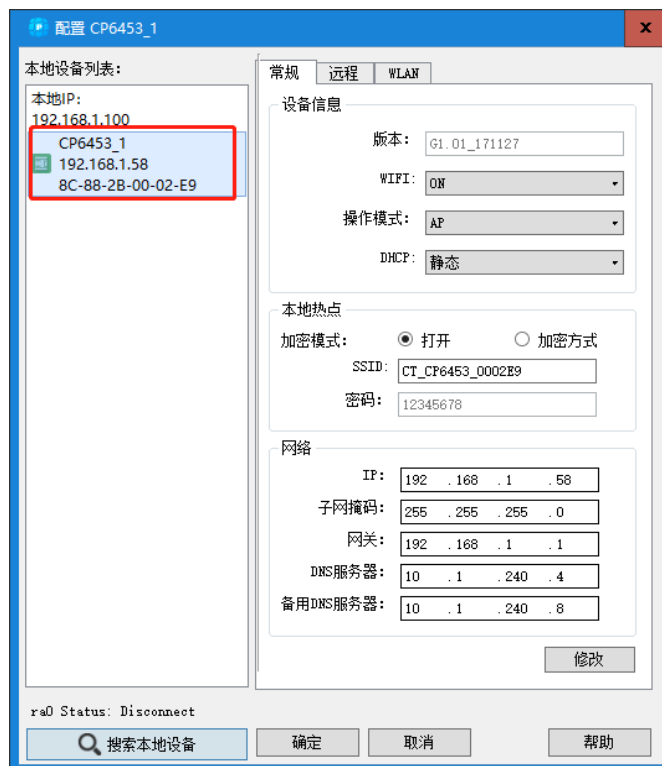
提示

配置参数前需将电脑 IP 设置与 CP6453 模块的 IP 处于同一网段，CP6453 模块端口号默认为 3636，不能修改。

4、在如下配置界面，点击“搜索本地设备”。

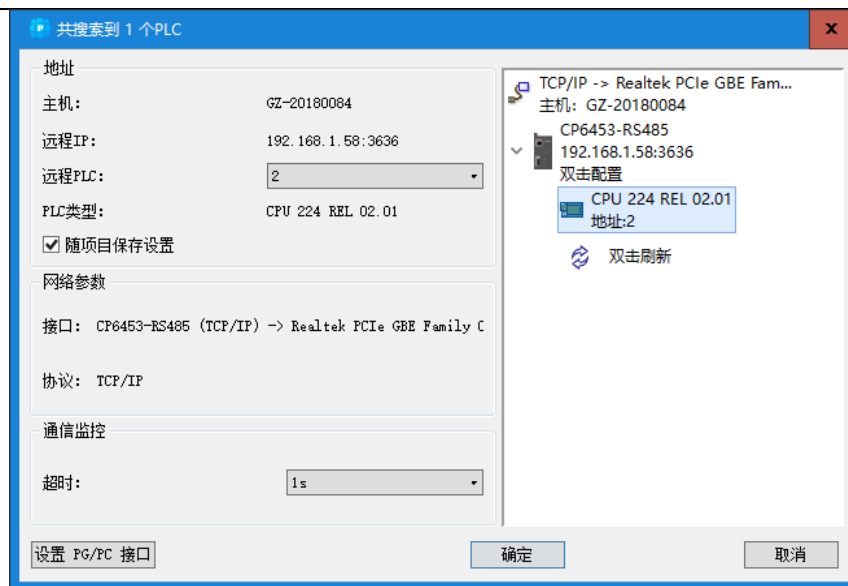


5、完成以上操作即可搜索到连接的 CP6453 网关并完成自动配置，点击“确定”完成常规配置。



搜索 PLC

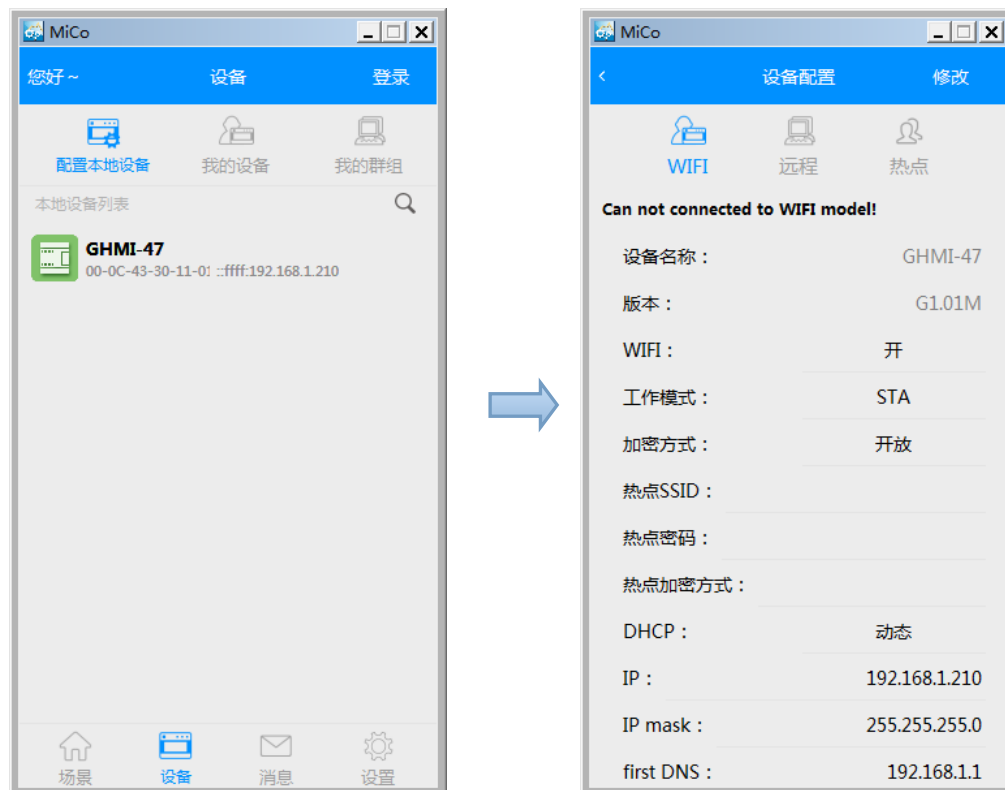
完成以上操作后，在通信界面点击“双击刷新”即可搜索到 PLC 进行通信。如下图所示：



使用远程时 CP6453 网关模块的配置

- 1) 将网关模块与上位机和路由器连接到同一网段;
- 2) 打开客户端 MiCo, 在“设备”页面点击搜索按钮搜索本地网关设备, 随即显示本地连接的在线网关, 点击即可查看并配置相关信息, 如下图所示:

网关模块基本配置



重要参数说明:

- **WIFI 开关:** 控制网关模块发出的 WIFI 信号, 选择“关”则无法接收或发出热点信号。
- **工作模式:** 网关模块 WIFI 模式分为 STA 和 AP 两种, STA 模式下, 网关设备接收热点; AP 模式下, 网关设备发出热点。
- **网络参数:** 其中包括 DHCP (动态/静态)、网关模块本机 IP、IP 掩码、首选 DNS 服务

器和备用 DNS 服务器，务必确保模块本机 IP 与上位机 IP 处于同一网段。

用户可根据需要修改上述界面中的配置参数，相关配置变更在点击右上角的“修改”并单击“确认”后才能生效。

注意事项：

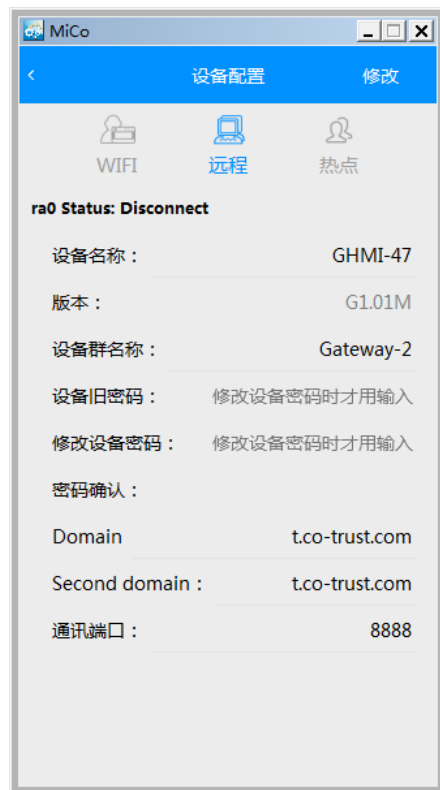
❑ 在 WIFI 的 STA 模式下，用户可设置网关设备接收的热点信号，包括热点 SSID 和密码以及加密方式，随后在“热点”标签下点击“连接”，即可将该网关模块连接到指定热点。

❑ 在 WIFI 的 AP 模式下，用户可以设置网关设备发出的热点信号，包括本机热点 SSID 和密码以及加密方式，然后单击“修改”即可保存本机热点信息，在网关模块 WIFI 打开的情况下供其他无线设备连接。

网关模块远程配置

网关模块需要加入 Mico 系统服务器之后才可供远程使用，下面介绍如何将网关模块配置到远程服务器中：

在 Mico 系统中连接本地网关设备后，点击“远程”标签设置设备信息和服务器信息。



设备名称：网关设备在服务器设备群组中的显示名称，由用户自定义。

设备群名称：将网关设备配置到其中的具体设备名称。

设备旧密码：认领设备所需的初始密码。

修改设备密码：根据需要修改设备密码。

密码确认：修改设备密码时进行确认，要求与修改的密码一致。

Domain：麦控星云服务器 - t.co-trust.com

Second domain：t.co-trust.com

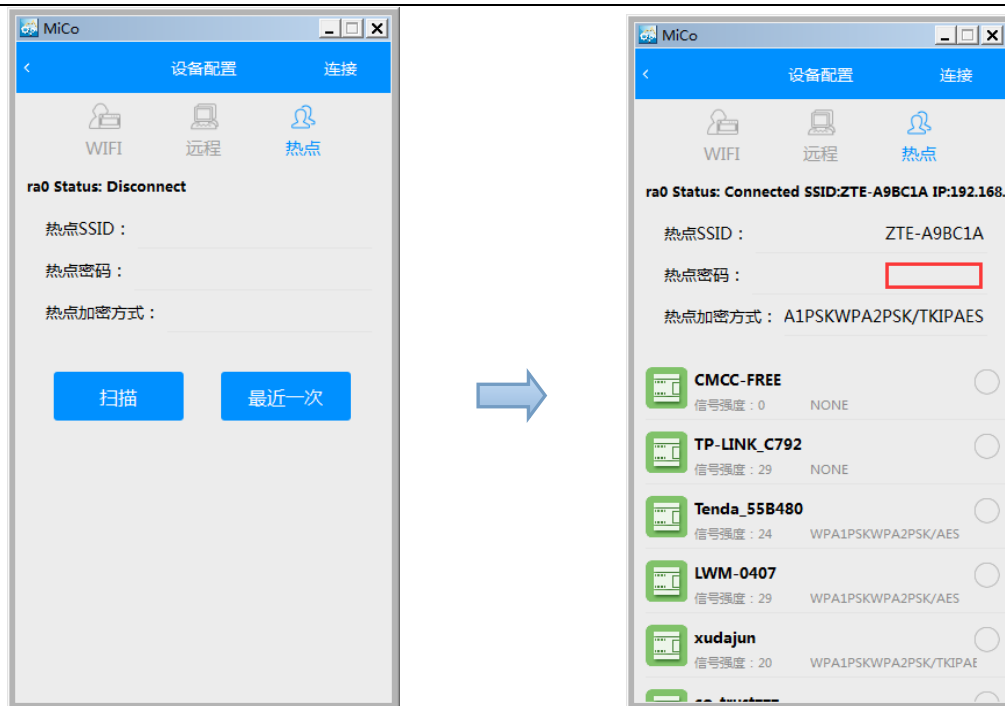
通讯端口：麦控星云服务器端口号，默认为 8888。

完成上述参数设置后，点击“修改”即可将远程参数配置到服务器中，该网关设备随即出现在指定群组中，可供用户认领使用。

热点连接

■ 在 STA 模式下网关模块可连接无线热点，具体操作如下：

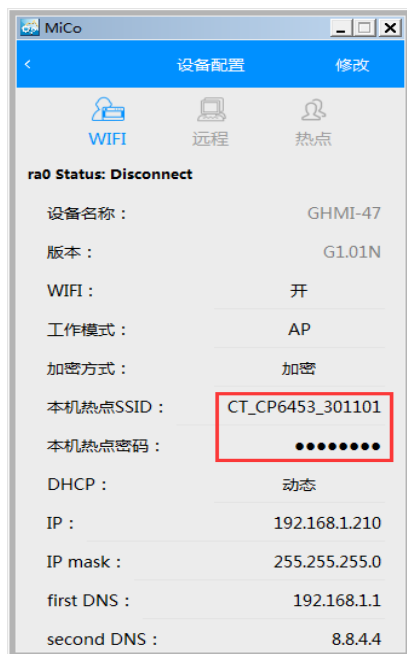
设备配置 -> 热点 -> 扫描 -> 选择所需热点信号 -> 输入对应的热点热点密码 -> 点击“连接”并确认，即可连接到指定热点。连接成功后，热点信息将显示在 WIFI 标签下。



在 AP 模式下连接网关模块无线热点，具体操作如下：

设备配置 -> WIFI -> 工作模式 AP -> 通过移动设备或笔记本电脑连接网关热点。

用户随后可在移动设备或笔记本电脑上登录客户端 MiCo，对具体场景和设备进行监控和操作。



注意：以上各参数配置完成后，将 CP6453 网关模块的网线重新接入网络。若 CP6453 网关模块的 LINK 指示灯亮起，则表示 CP6453 网关模块配置成功，即网关模块成功连接到服务器。

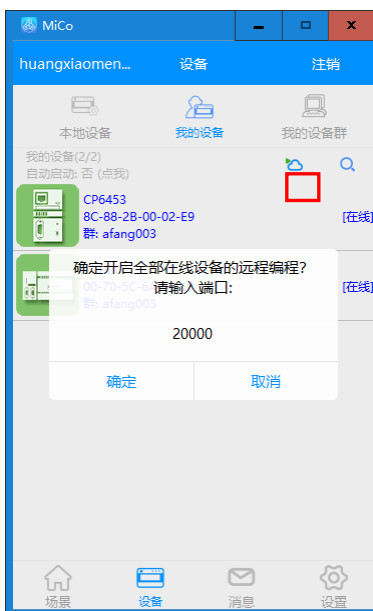
配置 Mico

登录 MiCo，并在 MiCo 主页面的“我的设备”里启动所有设备或部分设备的远程编程连接。如何将设备加入设备群，详细参考手册《Mico 远程监控系统手册》，下载地址：

<http://www.co-trust.com/>

通过如下方法启动所有设备的远程编程：

双击图标，打开 MiCo 并登录，在“我的设备”界面点击按钮，弹出对话框“确定开启全部在线设备的远程编程？”，选择“确定”，即可一键启动所有在线远程设备，如下图所示：



提示



- 1、若远程设备为离线状态则无法启动远程编程。
- 2、一键启动远程编程时，若不修改端口，则所有设备按上次开启时的端口开启，新设备按顺序按规则分配端口。

通过如下方法启动部分设备的远程编程：

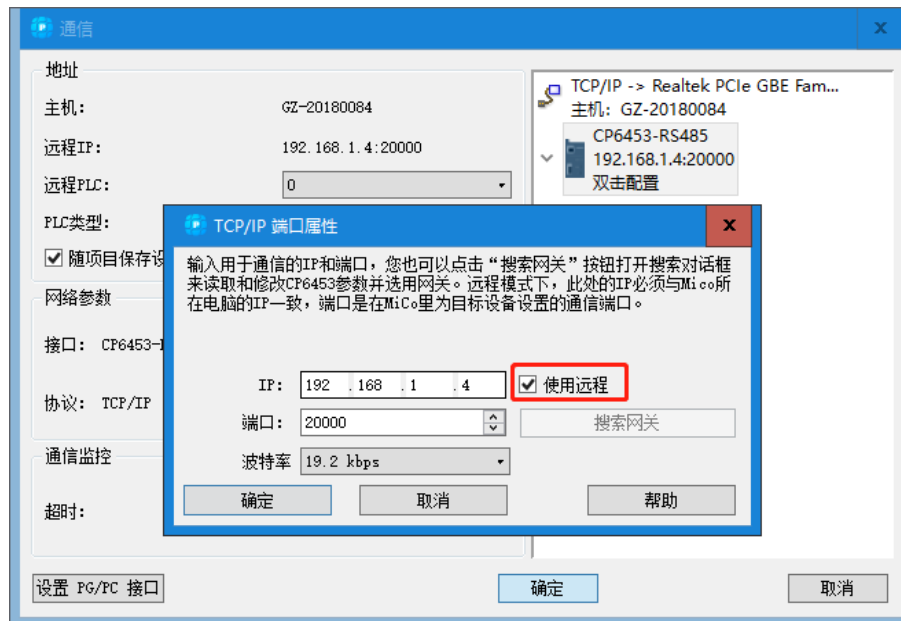
- 1、在 MiCo 的“我的设备”中找到要进行远程控制设备，右键点击该设备并选择“启动远程编程”选项，如下图所示：



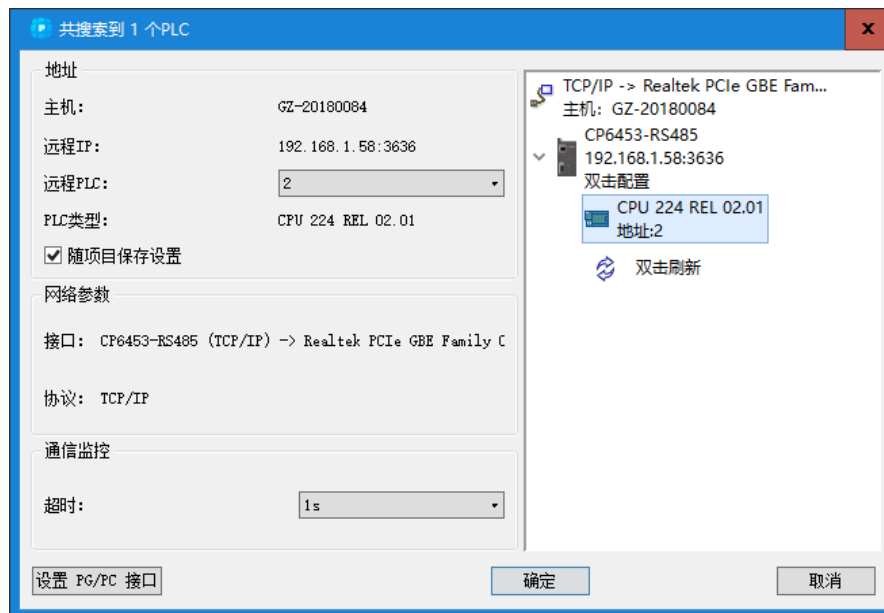
- 2、选择“启动远程编程”后，弹出如下窗口。在该窗口中设置本地端口（默认 20000），然后点击“启动远程编程连接”按钮，即可启动该设备的远程编程功能。

搜索 PLC

在 CP6453 “TCP/IP 端口属性” 界面勾选“使用远程”，再点击“确定”。



完成以上操作后，在通信界面点击“双击刷新”即可搜索到 PLC 进行通信。如下图所示：



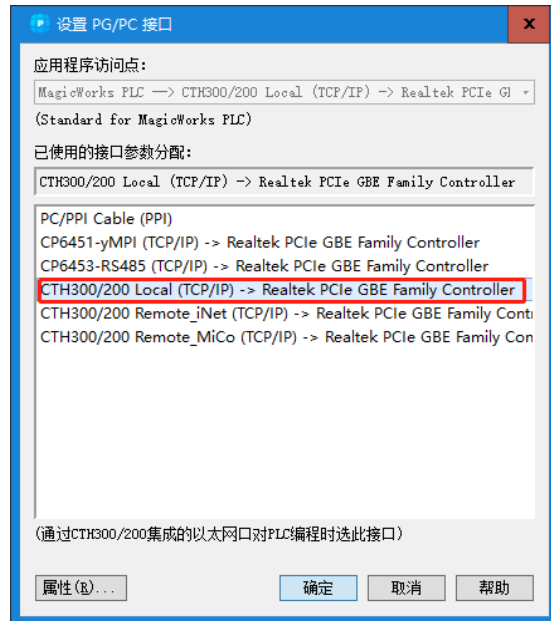
6.4 通过 CTH200/300 集成以太网口与 PLC 建立通信

通过 CTH300/200 集成的以太网口对 PLC 编程时选此接口。

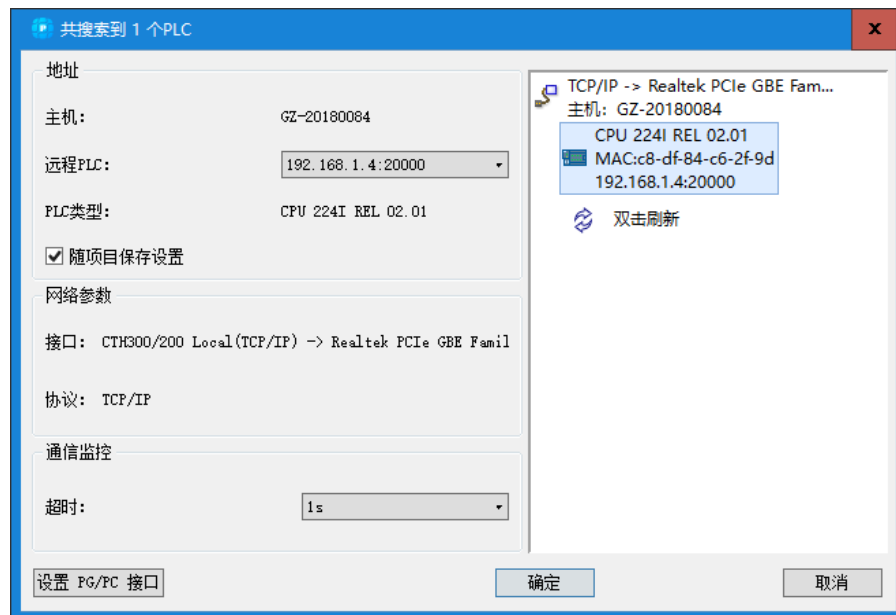
连接方法：通过标准网线将 PC 与 PLC 接入同一个局域网。

仅限本地模式（PC 跟 CTH200/300 在同一局域网），通信对话框搜索时即使 PLC 跟 PC 不在同一网段也能被搜索到，但如果要选定一个 PLC 与之进行通信时，必须要将编程设备的 IP 与该 PLC 设置为同一网段，否则通信不上。

在“设置 PG/PC 接口”界面选择 CTH300/200 Local(TCP/IP)->Realtek PCIe GBE Family Controller 协议，点击确定，如下图：



在通信界面点击“双击刷新”即可搜到 PLC 进行通信。如下图所示：

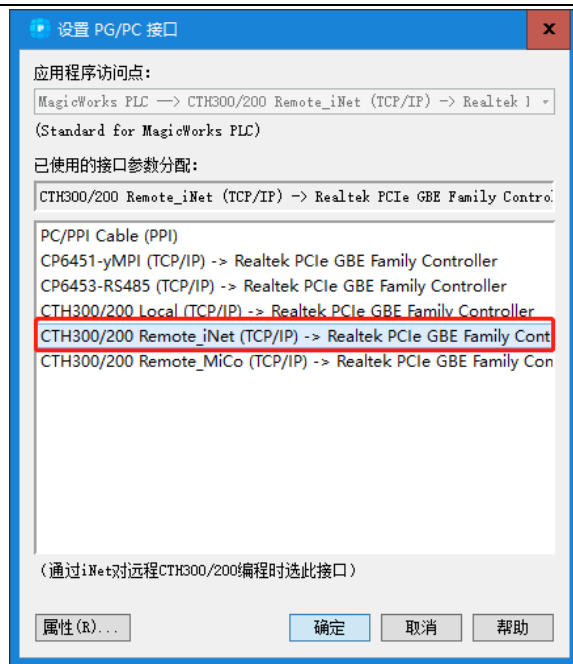


6.5 通过 iNet 与远程 CTH200/300 建立通信

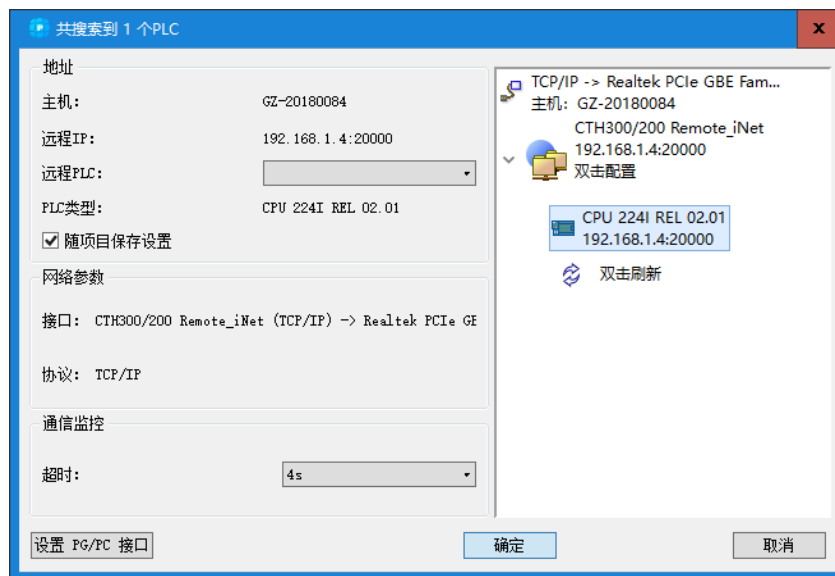
通过 iNet 对远程 CTH300/200 编程时选此接口。

连接方法：通过标准网线将 PLC 接入互联网中且与 PC 处于不同局域网，iNet 中我的设备设置本地端口使之进入就绪或通信中状态，iNet 需和编程软件在同一台 PC 或同一局域网中。

在“设置 PG/PC 接口”界面选择 CTH300/200 Remote_iNet(TCP/IP)-> Realtek PCIe GBE Family Controller 协议



在“通信”界面双击“双击刷新”即可搜索到 PLC 并通信，如下图所示：



6.6 通过 Mico 与远程 CTH200/300 建立通信

通过 MiCo 对远程 CTH300/200 编程时选此接口。

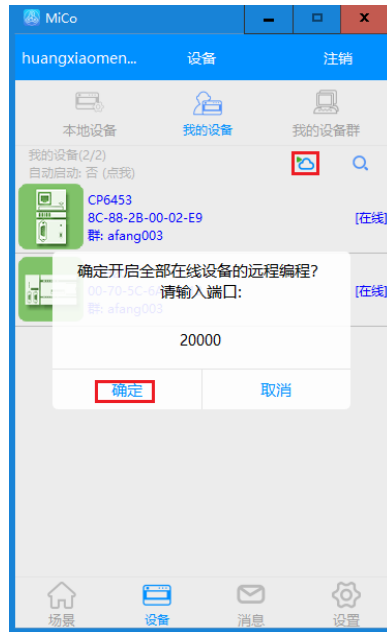
连接方法：通过标准网线将 PLC 接入互联网中，Mico 客户端开启 PLC 设备的远程编程端口，客户端和编程软件需在同一局域网中。

配置 Mico

登录 MiCo，并在 MiCo 主页面的“我的设备”里启动所有设备或部分设备的远程编程连接。如何将设备加入设备群，详细参考手册《Mico 远程监控系统手册》，下载地址：
<http://www.co-trust.com/>

通过如下方法启动所有设备的远程编程：

双击图标，打开 MiCo 并登录，在“我的设备”界面点击按钮，弹出对话框“确定开启全部在线设备的远程编程？”，选择“确定”，即可一键启动所有在线远程设备，如下图所示：



提示



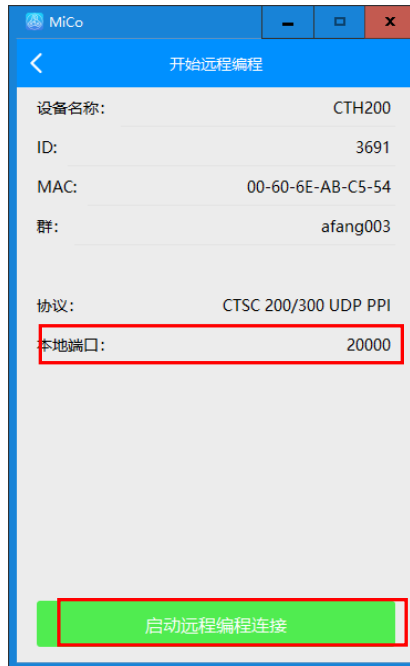
- 1、若远程设备为离线状态则无法启动远程编程。
- 2、一键启动远程编程时，若不修改端口，则所有设备按上次开启时的端口开启，新设备按顺序按规则分配端口。

通过如下方法启动部分设备的远程编程：

- 1、在 MiCo 的“我的设备”中找到要进行远程控制设备，右键点击该设备并选择“启动远程编程”选项，如下图所示：

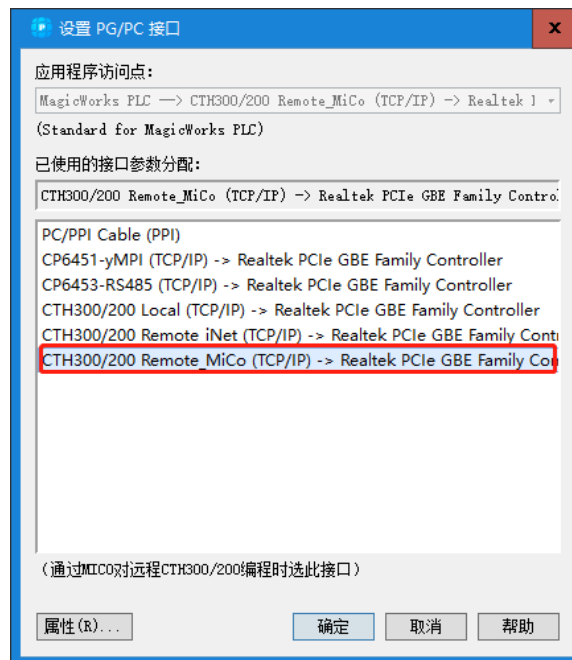


2、选择“启动远程编程”后，弹出如下窗口。在该窗口中设置本地端口（默认 20000），然后点击“启动远程编程连接”按钮，即可启动该设备的远程编程功能。



搜索 PLC

在“设置 PG/PC 接口”界面选择 CTH300/200 Remote_Mico(TCP/IP)协议。



在“通信”界面双击“双击刷新”，即可搜索到 PLC 并进行通信。如下图所示：

共搜索到 1 个PLC

地址

主机: GZ-20180084

远程IP: 10.1.10.66:20001

远程PLC: 10.1.10.66:20001

PLC类型: CPU H224X REL 02.01

☒ 随项目保存设置

网络参数

接口: CTH300/200 Remote_MiCo (TCP/IP) -> Realtek PCIe GE

协议: TCP/IP

通信监控

超时: 4s

TCP/IP -> Realtek PCIe GBE Fam...

主机: GZ-20180084

CPU H224X REL 02.01

10.1.10.66:20001

双击刷新

设置 PG/PC 接口

确定

取消

7 PLC 配置

本章主要对 PLC 功能块配置进行相关说明，您可以参考本章节内容进行 PLC 配置。

本章概要

- ☐ 系统块配置
- ☐ 数据块配置
- ☐ EasyCAN 组态
- ☐ 硬件组态
- ☐ 其余功能项

7.1 系统块配置

◆ **300 系列 PLC 的系统块通过如下方式进行配置：**

打开硬件组态界面→将 CPU 添加至机架→双击 CPU 节点→在弹出的窗口配置 CPU。

<备注> 具体配置方法请参考章节 [8.6.1 一般操作步骤](#)。

◆ **200 系列 PLC 的系统块可在项目管理器窗口进行配置：**

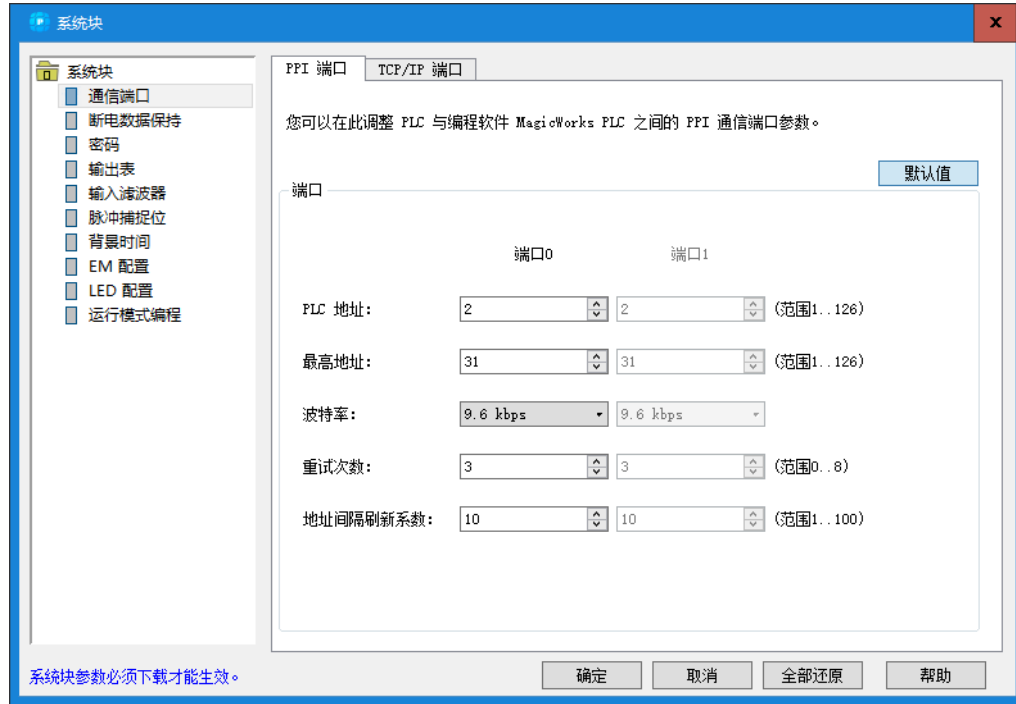
在项目管理器窗口的项目树展开 CPU 节点，然后在右侧的工作区域双击图标  打开系统块对话框，即可查看和编辑当前 200 系列 CPU 的系统配置。

在下载或上载系统块之前，您必须成功地建立 PC（安装有 MagicWorks PLC）与 CPU 之间的通信。

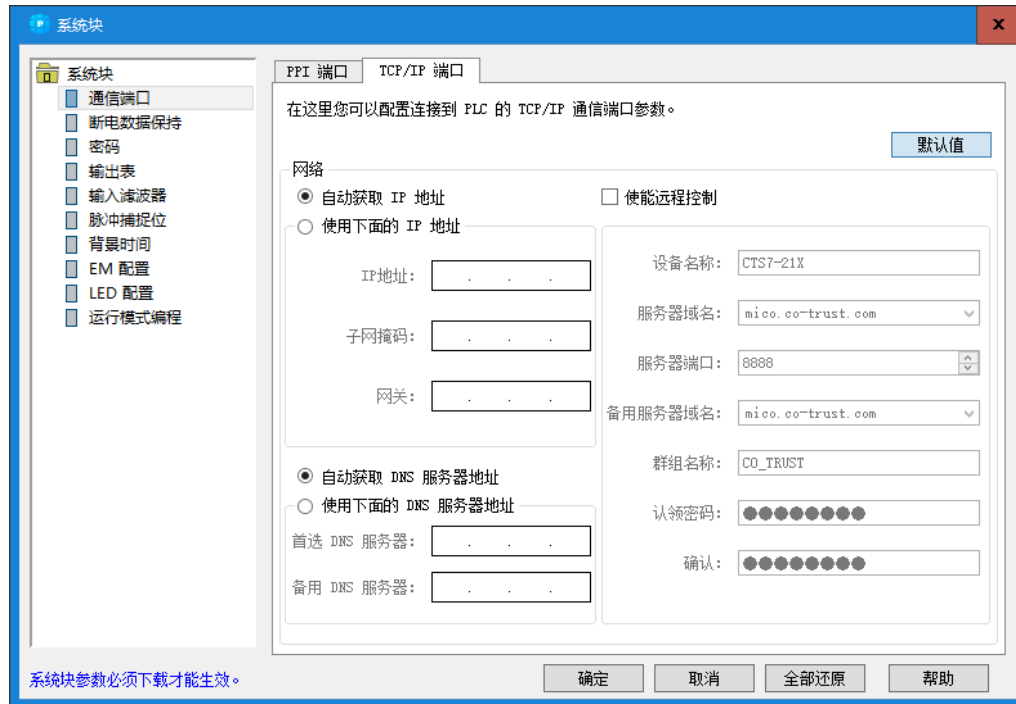
7.1.1 通信端口

在系统块对话框左侧项目树点击“通信端口”节点，您就可以查看或更改 CPU 的通信参数。

PPI 端口参数设置：



TCP/IP 端口参数设置：



<备注> CPU 224 只有一个通信端口 PORT0，CPU 226L/P/M 有两个通信口 PORT0 和 PORT1，PORT0 和 PORT1 都支持三种波特率：9.6 kbps、19.2 kbps 和 187.5 kbps。

7.1.2 断电数据保持

选择系统块对话框左侧目录的“断电数据保持”，选择需要断电保持的数据区域并下载至 PLC。

您最多可定义六个保持范围，选择您希望在上电周期过程中保持的存储区。您可将下列存储区中的地址范围定义为保持：**V**、**M**、**C** 和 **T**。默认情况下，所有的 **V**、**T** 和 **C** 存储区均被设为断电保持。**M** 存储区的前 14 个字节的默认设置是非保持。对于定时器，只能保持有记忆定时器（**TONR**），而且只有定时器和计数器的当前值可定义为保持。每次上电时定时器和计数器位均被清除。您可以手动更改保持范围，取消对某些区域的断电保持。

CPU 具有超级大电容，可在 **CPU** 断电后保存 **RAM** 数据。有些 **CPU** 型号支持延长可保持 **RAM** 数据时间的选用电池。电池只有在超级电容完全放电后才提供电源。

断电时：**M** 存储区的前 14 个字节（**MB0** 至 **MB13**），如果被配置为保持，在 **CPU** 模块失去电源时被永久性保存在 **EEPROM** 中。

上电时：**CPU** 检查 **RAM** 存储区，检查超级电容或电池是否已成功地将数据保存在 **RAM** 中的数据。如果 **RAM** 数据已成功保存，**RAM** 存储区的保持区不变。永久 **V** 存储区（在 **EEPROM** 中）的相应区域被复制至 **CPU RAM** 中的非保持区。用户程序和 **CPU** 配置也从 **EEPROM** 恢复。**CPU RAM** 的所有其他非保持区均被清零。如果未保存住 **RAM** 的内容（例如长时间断电后），**CPU** 清除 **RAM** 的保持和非保持区域，并为上电后的首次扫描设置保持数据丢失标志位（**SM0.2**）为 1。用户程序和 **CPU** 配置然后从 **EEPROM** 复制至 **CPU RAM**。此外，**EEPROM** 中的 **V** 存储区永久区域和 **M** 存储区永久区域（如果被定义为保持）从 **EEPROM** 复制至 **CPU RAM**。**CPU RAM** 的所有其他区域均被清零。



提示

- 断电保持区域的地址范围跟 **PLC** 类型与版本有关，系统块在编译时会根据 **PLC** 类型对此页的参数进行检查，如果设置超出了 **PLC** 内存范围，输出窗口会给出一条警告信息。
- 将范围 **MB0** 至 **MB13** 更改为保持范围会启用一种特殊功能，该功能会在断电时自动将这些位置保存在 **EEPROM** 中。

7.1.3 密码保护

在系统块对话框点击左侧目录的“密码”结点。选择密码等级，输入并验证密码（密码区分大小写），然后将修改下载至 **PLC**。

密码授权访问功能和存储区：如果没有设置密码，**CTSC-200** 提供不受限制的访问。受密码保护时，**CTSC-200** 根据授权级别来提供操作功能限制。

- CTSC-200** 的默认密码级别是级别 1（不受限制的访问）。
- 在网络中输入密码并不影响 **CTSC-200** 的密码保护。授权一位用户访问受限制的功能并不意味着授权其他用户访问这些功能。在某一时刻，**CTSC-200** 只允许一位用户执行无限制访问。
- 因为级别 1 允许不受限制的访问，如果你将其它密码级别更改成级别 1，也就是等于取消密码保护。

下表列出不同授权级别允许的不同访问功能。

操作说明	权限级别1	权限级别2	权限级别3	权限级别4
读写用户数据	允许	允许	允许	允许
运行/停止/上电复位	允许	允许	允许	允许
读写实时时钟	允许	允许	允许	允许

STOP 模式下写 Q 点	允许	需验证密码	需验证密码	需验证密码
强制数据	允许	需验证密码	需验证密码	需验证密码
上载程序块/数据块/系统块	允许	允许	需验证密码	不允许
下载程序块/数据块/系统块	允许	需验证密码	需验证密码	需验证密码(系统块不允许下载)
清除程序块/数据块/系统块	允许	需验证密码	需验证密码	需验证密码(系统块不允许删除,允许三个块一起删除)
运行时编辑	允许	需验证密码	需验证密码	不允许
首次或多次扫描	允许	需验证密码	需验证密码	需验证密码
刷新扫描周期	允许	需验证密码	需验证密码	需验证密码
项目比较	允许	允许	需验证密码	不允许
程序状态监控(时间戳比较通过)	允许	允许	允许	允许
程序状态监控(时间戳比较不通过)	允许	需验证密码	需验证密码	不允许

提示：输入密码后，当编程设备从 CTSC-200 上断开后，该密码的授权级别最多可保持一分钟有效时间。始终在断开电缆之前退出 MagicWorks PLC，以防另一位用户使用编程设备特权。

忘记 PLC 密码怎么办？

如果忘记 PLC 密码，您必须清除 PLC 存储区，重新载入程序。清除 PLC 存储区使 PLC 进入 STOP 模式，并将 PLC 复原为工厂设置的默认值，PLC 地址、波特率和实时时钟除外。

欲清除 PLC 中的程序：

- 1) 选择菜单命令“PLC”→“清除”，显示“清除”对话框。
 - 2) 选择所有的复选框，并点击“确认”按钮核实采取的措施。
 - 3) 如果密码已被配置，MagicWorks PLC 会显示一个密码验证对话框。欲清除密码，在密码验证对话框中输入 CLEARPLC，继续执行“全部清除”操作。（CLEARPLC 密码不区分大小写字母）
- “全部清除”操作不从存储卡中拆卸程序。因为存储卡存储密码和程序，您必须重新对存储卡进行编程，才能拆卸丢失的密码。



警告

清除 PLC 存储区影响数字量和模拟量输出状态。工厂默认值是当 PLC 处于 STOP（停止）模式时，将数字量输出复位为零（关断）。但是，如果您已经使用系统块的“输出表”标签定义数字量输出的“安全状态”，您需要注意删除系统块时也将删除该“安全状态”定义。下一次您将 PLC 从 RUN 模式转为 STOP 模式时，数字量输出会被设为零（工厂默认值）。另外，当您删除程序块时，模拟量输出被冻结为当前值。直到您下载新的程序块，能够对模拟量输出状态进行更改的唯一方法是使用状态表。清除 PLC 存储区时，如果 CTSC-200 PLC 与设备相连，对数字量输出状态的改动可传送至该装置。而且，您删除程序块后，因为输出被“冻结”，与模拟量输出相连的装置不再能够正常响应。如果清除了 PLC 存储区，却没有仔细考虑对数字量和模拟量输出的影响，装置操作可能出现无法预料的情况，可能导致人员死亡和严重伤害及/或装置损坏。请始终采用适当的安全预防措施，并在清除 PLC 存储区之前确保程序处于安全状态。

7.1.4 数字量输出表

在系统块对话框点击左侧目录的“输出表”结点，然后在右边的窗口中选择数字量标签，您就可以在此查看和设定 CPU 停机时数字量输出值，修改下载至 PLC 后设定生效。

“数字量输出表”令您可在 RUN-to-STOP（运行至停止）转换后将数字量输出设置为安全状态，或保持在转换为 STOP 模式之前所存在的输出状态。数字量输出表是下载和存储到 CTSC-200 CPU 的系统块之一部分。

将输出冻结在最后的狀態：点击此复选框，就可在 PLC 进行 RUN-to-STOP（运行至停止）转换时将所有数字量输出冻结在其最后的状态。

冻结状态：此表允许您在 RUN-to-STOP（运行至停止）转换时使输出进入已知状态。请点击您希望设置为“接通”（1）的每个输出的复选框。RUN-to-STOP（运行至停止）转换时，所有输出的默认状态为“关断”（0）。

<备注> 对于 CPU 224 和 CPU 226，您可以设置 Q 位值的 16 个字节。

7.1.5 模拟量输出表

在系统块对话框点击左侧目录的“输出表”结点，然后在右边的窗口中选择模拟量标签，您就可以在此查看和设定 CPU 停机时模拟量输出值，修改下载至 PLC 后设定生效。

模拟量输出表令您可在 RUN-to-STOP（运行至停止）转换后将模拟量输出设置为安全数值，或保持在转换为 STOP（停止）模式之前存在的输出数值。模拟量输出表是下载和存储到 CTSC-200 CPU 的系统块之一部分。

将输出冻结在最后的狀態：点击此复选框，就可在 PLC 进行 RUN-to-STOP（运行至停止）转换时将所有模拟量输出冻结在其最后的数值。

冻结数值：此表允许您在 RUN-to-STOP（运行至停止）转换时将模拟量输出设置为某已知数值（-32768 ~ 32767）。

7.1.6 数字量输入滤波

在系统块对话框点击左侧目录的“输入滤波器”结点，然后在右边的窗口中选择数字量标签，您就可以在此修改您的设定，然后将修改下载至 PLC。

CTSC-200 允许您为数字量输入点选择一个定义时延的输入滤波器，输入滤波延时的范围为 0.2 毫秒至 12.8 毫秒。

通过设置输入延时，您可以过滤掉数字量输入接线上可能对输入状态造成不良影响的噪音。输入状态改变时，输入必须在延时期限内一直保持在新状态，才被认为是有效输入，否则将作为噪音被过滤掉。

输入滤波延时的默认值是 6.4 毫秒。

设置输入延迟操作如下：

- 1) 从下拉框中选择延时值。
- 2) 点击“确定”按钮。

**提示**

- 每相邻的 4 个数字量输入点共用一个延时参数；
- 当指令读取输入中断和脉冲捕捉时，数字量输入滤波器会影响输入值。根据您的滤波器选项，可能会造成程序错过一个中断事件或脉冲捕捉。高速计数器只记录未经过滤的输入事件。

7.1.7 模拟量输入滤波

在系统块对话框点击左侧树的“输入滤波器”结点，然后在右边的窗口中选择模拟量标签，您就可以在此选择需要滤波的模拟量，然后将修改下载至 PLC。

隔离和单点接地

模拟输入通道没有相互隔离，也没有与 CPU 隔离。如果传感器使用与 CPU/模拟 EM 不同的电源，可能会导致较高的波动的共模电压，从而影响到所报告的模拟输入值的稳定性。

要纠正这种状况，可将电源回路共同连接到同一点上。可以使用模拟 EM 上的 M 接线端。

信号质量和响应

传感器的响应时间或传送模拟量信号至模拟 EM 的信号线的长度和状况，也可能引起模拟输入值的波动。在这种情况下，波动值可能变化太快，导致程序逻辑无法有效响应。可在 CPU 中组态滤波器，以平滑模拟输入信号，从而将一个平均值传送给程序逻辑。

对于 CPU 22x，可在单个模拟输入上选择软件滤波器。过滤值是预选数目的模拟输入采样的平均值。对于所有启用过滤的模拟输入，滤波器采用相同的规范（采样数目和死区）。

滤波器具有快速响应特性，使得较大的变化可以在过滤值中得到快速反映。当输入相对于平均值超出指定的变化值时，滤波器将使一个阶跃函数改变为最新的模拟输入值。该变化值即为死区，它在模拟输入数字值的计数器中指定。

当 CPU 的模拟滤波器打开时，模拟值将在每次 CPU 扫描时自动读入。在 AI 字（用于程序逻辑）中报告的实际值通过对这些测量到的模拟值求平均值而得出。在系统块编辑器的输入滤波器分支的“模拟”标签的“采样数目”域中，指定用于计算平均值的数值的数目。采样数目可以设置为下列数值：2、4、8、16、32、64、128 和 256。采样数目多将使结果稳定但响应较慢，采样数目少将使结果不稳定但响应较快。

死区

在新值适度接近于当前平均值的许多应用场合中，使用平均值十分有效。然而，在某些应用场合下，当模拟输入值大幅改变时，平均计算可能会导致无法接受的响应时间。也就是说，在大幅改变时，可能既需要过滤又需要快速响应。

在这类应用中，可以组态一个死区。死区定义了当前平均值周围的一个范围（当前平均值 $\pm$ 死区值）。如果最新模拟输入值位于该范围内，则平均值如上所述进行计算。然而，如果新的模拟数值位于该范围之外，则不进行平均计算。而是立即将平均值设置为新的模拟输入值。这样可使滤波器对于大幅改变的模拟输入值也有快速响应。

在系统块编辑器的输入滤波器分支的“模拟”标签的“死区”域中指定死区。可以将死区值设置为 0 来禁用死区。当死区被禁用后，不管输入值的变化有多大，平均值都将如上所述进行计算。要启用死区，可将该域的值设置为 16 ~ 4080 范围内的数值。

**提示**

不应通过模拟量字中传递数字量信息或报警指示的模块使用模拟量过滤。**AS-i** 主站模块、热电偶模块和 **RTD** 模块要求禁止 **CPU** 模拟量输入滤波功能。模拟量输入滤波的默认系统块设置是“打开”（选择）。您必须禁止所有相关模拟量输入过滤（取消选择），并下载修改的系统块，才能操作这些 **I/O** 模块。

7.1.8 脉冲捕捉位

在系统块对话框点击左侧树的“脉冲捕捉位”结点，编辑属性，然后下载至 **PLC**。

CTSC-200 为 **CPU** 的某些或全部数字量输入点提供脉冲捕捉功能。脉冲捕捉功能允许您捕捉高电平脉冲或低电平脉冲。此类脉冲出现的时间极短，**PLC** 在扫描周期开始读取数字量输入时，可能无法始终看到此类脉冲。

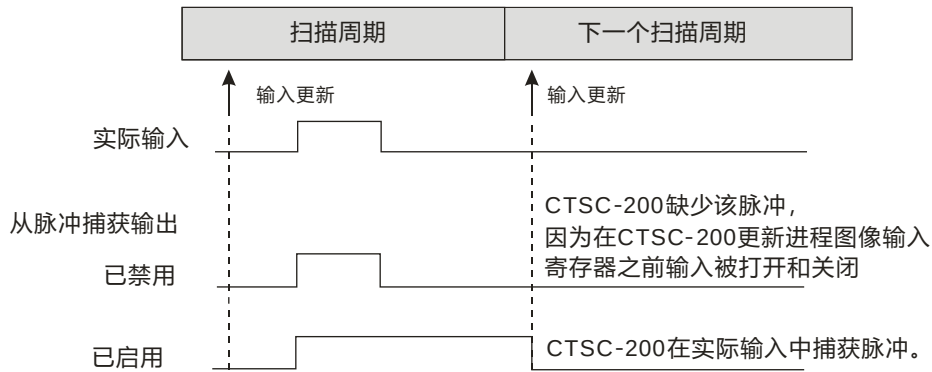
当为某一输入点启用脉冲捕捉时，输入状态的改变被锁定，并保持至下一次输入循环更新。这样可确保延续时间很短的脉冲被捕捉，并保持至 **CTSC-200** 读取输入。

该功能可使用的最大数字量输入数目取决于 **PLC** 的型号：

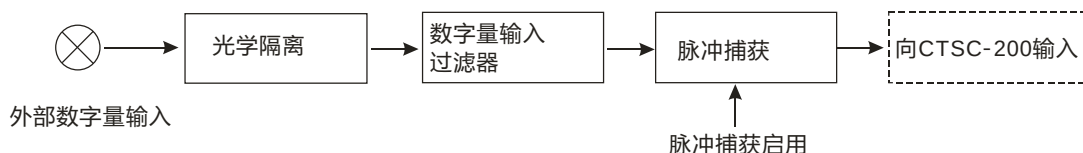
CPU 类型	最大数字量输入
CPU 122/222	8 个
CPU 124/224/224I/224X/224XP/H224/H224X/226M	14 个
CPU126/226/226I/H226XM/H226XC/H226XL/H226M/H226L/H228XL	24 个

可为每个局部数字量输入分别启用脉冲捕捉操作。

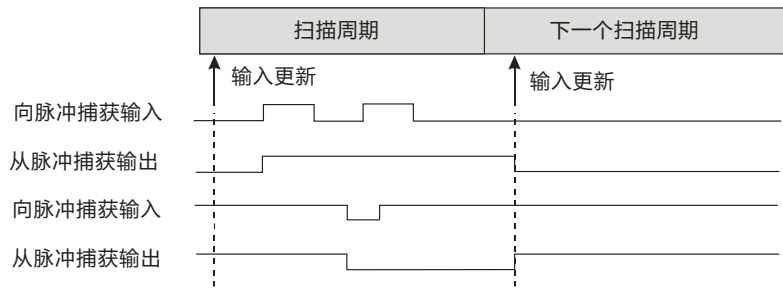
以下数字量显示（脉冲捕捉启用和禁用）**CTSC-200** 的基本操作状况。



由于脉冲捕捉功能在输入通过输入滤波器后对输入进行操作，您必须调整输入滤波时间，以防滤波器过滤掉脉冲。下图显示数字量输入电路方框图：



下图显示启用脉冲捕捉功能时对各种不同输入条件的响应。如果在某一特定扫描中存在一个以上脉冲，仅读取第一个脉冲。如果在某一特定扫描中有多个脉冲，则应当使用上升 / 下降沿中断事件。



7.1.9 背景时间

在系统块对话框点击左侧树的“背景时间”结点。在右边的窗口设置背景时间参数，下载至 PLC。

在此，您可以设置 PLC 专门用于处理通信操作（运行模式编辑编译或执行状态监控等）所花的时间占扫描周期的百分比。如果增加专门用于处理通信的时间百分比，扫描周期时间也会增加，导致整个控制过程的运行速度变慢。

背景时间百分比的默认值为 10%，这个选择为快速处理编译 / 状态监控操作同时尽量减小对控制过程的影响进行了平衡。当然，您也可以根据系统的具体需求调整该参数，增幅为 5%，范围为 5% ~ 50%。

7.1.10 EM 配置

在系统块对话框点击左侧树的“EM 配置”结点。在此，您可以查看当前项目中已组态的智能模块所占用的 V 存储区情况，此页面的文本框不可编辑。

目前，CTSC-200 PLC 支持的智能模块只有 EM277A 模块。由于 EM277A 模块的配置参数由 DP 主站发送，故在此处不显示其使用的 V 存储区地址。

7.1.11 LED 配置

选择系统块对话框左侧的“LED 配置”结点。选择您所需要的选项，然后将修改下载至 PLC。

在右边的窗口中使用复选框设置 PLC 面板上标签为“SF/DIAG”的 LED 在何时点亮，有两种情况可以让 LED 亮起黄色：

- 在 CPU 中存在强制的地址；
- 系统中存在 I/O 错误；

<注意> 系统故障 SF（红色 LED 发光）优先于诊断 DIAG（黄色 LED 发光）。

7.1.12 运行模式编程

选择系统块对话框左侧的“运行模式编程”结点，选择是否禁用运行模式编程功能，然后将修改下载至 PLC。此项设置不影响 PLC 的存储区大小。

<备注> 运行模式编程功能的具体说明请参考章节 [9.10 运行模式编程](#)。

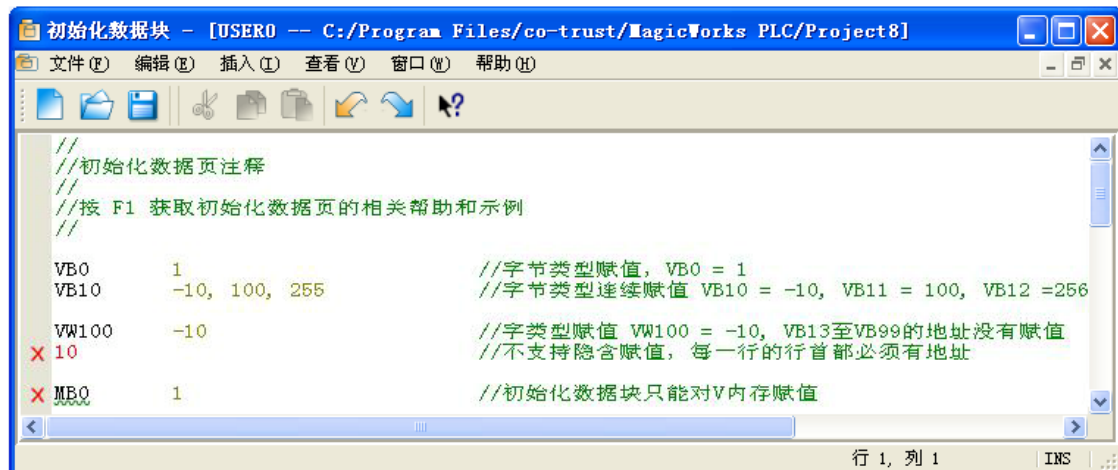
7.2 数据块配置

如何使用数据块初始化 V 内存

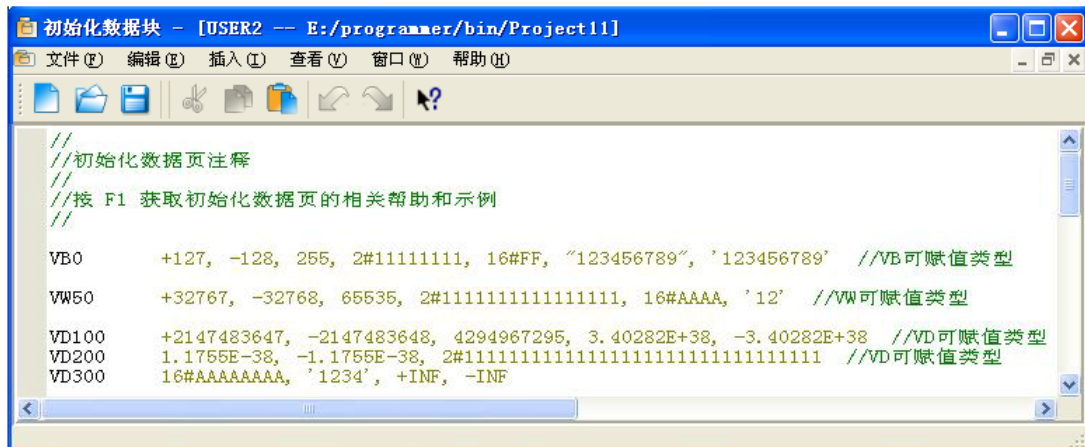
初始化 V 内存数据块的功能是供用户对 PLC 的 V 存储区进行初始化。用户可以对 V 存储区的字节（VB）、字（VW）或双字（VD）地址赋值，可赋给数据常数或 ASCII 字符常数。

- 数据块不支持隐含赋值，支持一行连续多个赋值。即数据块的每一行的行首都必须包含一个显性地址赋值（绝对或符号地址），一个地址后可输入多个数据值，多个数据值之间必须用逗号或空格隔开。当一行有多个赋值时，编辑器会自动进行隐性地址赋值，根据行首的地址分配及数据值大小（字节、字或双字）指定适当的 V 存储区数量。
- 数据块编辑器具备实时语法检查功能和自动格式化功能。用户输入时，文本以高亮显示，不同颜色的字体表示不同的意义，具体情况参见数据块的语法检查一节。键入一行光标离开后（按 ENTER 键，或鼠标点击使光标换行），数据块编辑器会自动对此行进行格式化（对齐地址和注释、统一大小写，统一逗号和空格，符号显示等）。
- 在完成一赋值行后按 CTRL+ENTER 键组合，会令地址自动增加至下一个可用地址。
- 注释以双斜线//指定，注释可选。

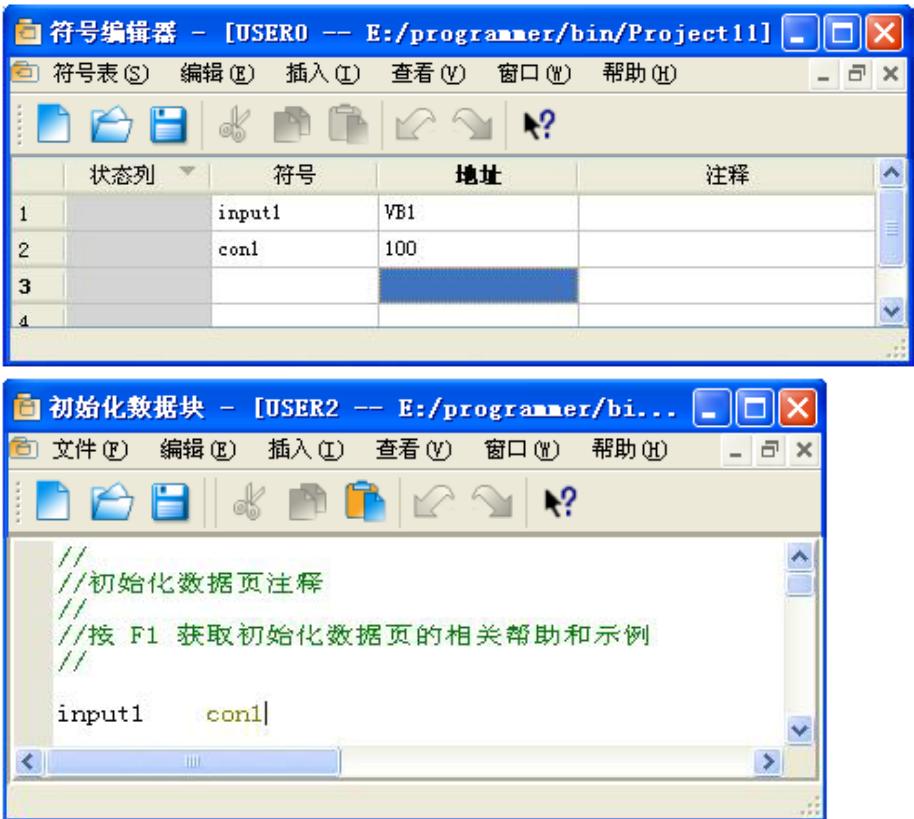
数据块赋值的一般规则



直接地址和数值赋值



符号地址和符号数值赋值



有效的 ASCII 赋值

VB 类型地址可以使用单引号（长度不限）或双引号（长度不能超过 254 个字符）字符串赋值：

VB0 'A', 'ABCDE', "ABCDEFGHIJK"

VW 类型地址只能使用单引号且长度为 2 个字符的字符串赋值：

VW0 'AB'

VD 类型地址只能使用单引号且长度为 4 个字符的字符串赋值：

VD0 'ABCD'

双引号 ASCII 常数字符串格式将在字符串前存储一个包含串长度的前导字节。

数据块的语法检查

一旦在包含错误的行尾按 ENTER 键，立即会在数据块左页边显示一个红色的叉号。在下载至 PLC 之前，您必须纠正全部输入错误。

X	ybbQ	255	//没有定义的符号错误
X	&&	255	//非法语法错误
X	VB0	256	//数值对该地址无效

引起输入错误的条件包括：

- 指定了错误的存储区（非 V 存储区）
- 在一行的行首没有指定地址
- 使用非法语法或无效数值
- 未能正确地指定注释（双前斜线必须位于注释之前：//注释样本）

数据块字体显示：

- 地址：黑色字体
- 数据：黄色字符

- 注释：绿色字体
- 语法错误：红色字体（字符串变量在行首，数字在行首，不合法的字符串变量，双引号字符串变量长度超过 255，不合法的字符如&@等）
- 地址非法使用：黑色字体红色波浪线（地址超出范围，如：VB10240）
- 数值非法使用：黄色字体红色波浪线（数值超出范围，VW/VD 后接双引号字符串，VW/VD 后单引号字符串的长度不为 2/4（如果字符串内有非法字符显示为红色））
- 未定义的符号：黑色字体绿色波浪线（符合符号名称的规则但在符号表未定义，关键字同样也显示为未定义的符号，如：LD MB0）

数据块的编译

可以编译数据块：在项目管理器使用菜单命令“PLC”→“全部编译”或鼠标选中项目树中的初始化 V 内存数据块结点然后使用菜单命令“PLC”→“编译”。

编译数据块时，如果编译程序发现错误，会在“输出窗口”显示错误的类型与存在的行号。

数据块编译中包含的错误有：

- ERROR 2770：地址重复赋值
- ERROR 2771：地址范围对当前 PLC 类型无效
- ERROR 2772：非法语法
- ERROR 2773：数值大小对地址无效
- ERROR 2774：字符串长度超过 255
- ERROR 2779：没有定义的符号

将数据块下载至 PLC

如果编辑了数据块，则需要将数据块下载至 PLC。在下载前软件会自动对数据块进行编译，如果存在编译错误，会在下载对话框中给出提示，不允许选择数据块下载。

数据块只有下载成功后才会生效。

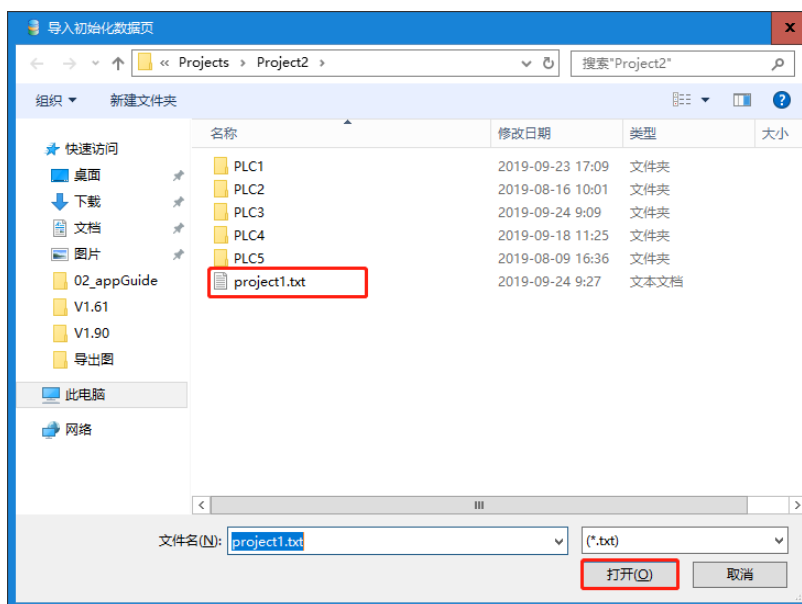
从 PLC 上载数据块

您必须在项目管理器中打开一个项目，才能上载数据块。

您只能上载该 PLC 的数据块地址部分和数据值部分，注释部分会被丢掉。

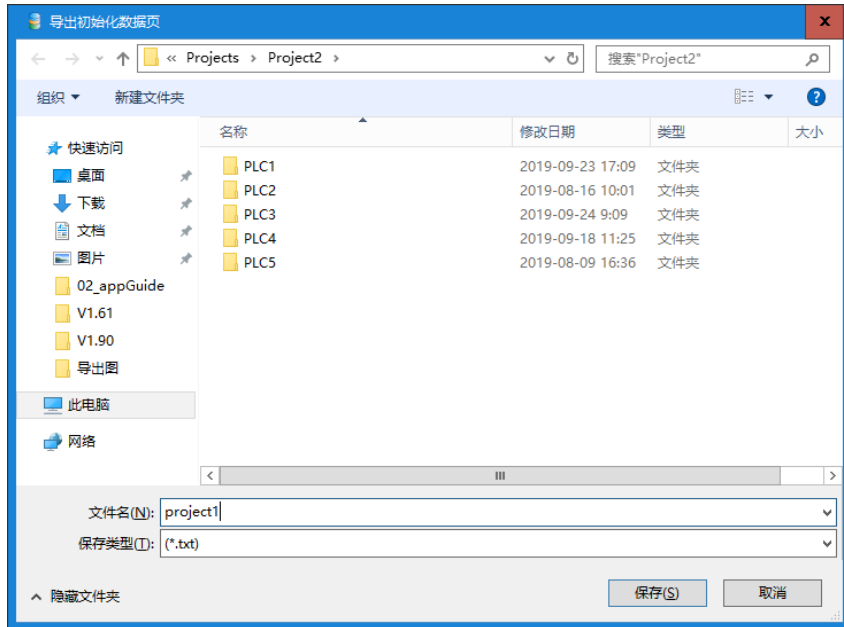
导入初始化数据页文件

将 ASCII 文件格式的初始化数据页文件（后缀.txt）内容导入到当前激活的初始化数据页面窗口。



导出初始化数据页文件

将当前初始化数据页的内容以 ASCII 文件格式导出到后缀为.txt 的文本文件。



查看 ASCII 文本文件格式的一种简单方法是导出一个现有的初始化数据页文件，然后使用 Windows Notepad（记事本）编辑器检查文件内容。

如果导入 ASCII 文本文件（例如，test.txt）因语法错误失败，初始化数据编辑器会以对话框的形式弹出提示。如果文件通过语法测试，则被添加至项目。文件中的注释也一起被导入。

7.3 EasyCAN 组态

EasyCAN 组态软件是用来配置 COTRUST CANopen 通信产品的工具软件，能图形化的组态复杂的 CANopen 网络。

7.3.1 EasyCAN 组态软件特点

- 1) 给用户提供了图形化的 CANopen 通信配置界面；
- 2) 支持主站类型为 CPU226M-CAN，支持从站类型为 EM277C/E10/H1A/H2A/Others；
- 3) 每个主站最多可连接 32 个从站，每个 EM277C 最多可组态 7 个模块，每个 E10 最多可组态 14 个参数，读写最多各 7 个；
- 4) 允许用户自定义从站和从站模块的映射地址，允许映射到 V 内存；
- 5) 支持对组态工程进行编译；

以下三种情况允许保存，但编译时会给出错误提示，并且不允许下载：

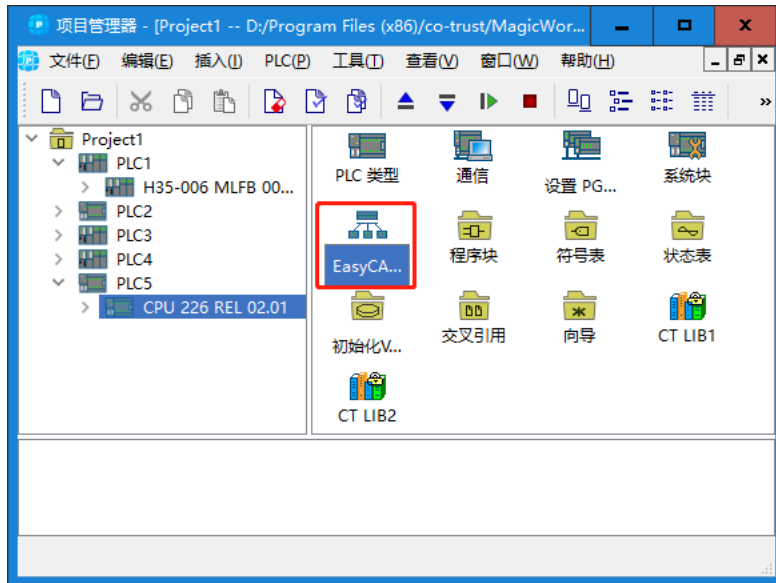
- 主站选择了使用 CAN 总线但从站数目为 0；
- 主站的节点 ID 与从站有重复，主站所连接的服务器或客户端的节点 ID 与从站的节点 ID 有重复，或一个从站节点 ID 与另一从站的节点 ID 有重复；

- 组态从站模块时有空的插槽。

7.3.2 组态并通信

EasyCAN 组态软件是用来配置 COTRUST CANopen 通信产品的工具软件，能图形化的组态复杂的 CANopen 网络。组态 EasyCAN 步骤如下：

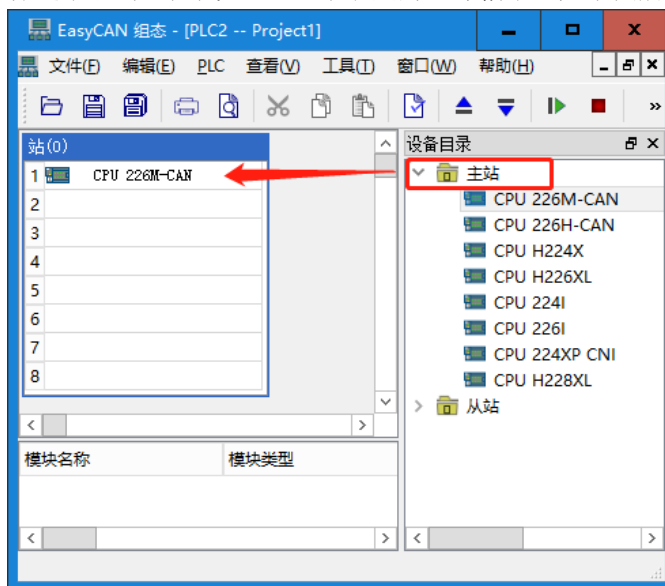
1、在 200CPU 站点下的项目管理器界面双击 EasyCAN 组态图标，，打开组态界面。



2、在 EasyCAN 组态界面中，为当前项目组态一个 EasyCAN 网络，请遵循以下步骤：

1) 添加主站

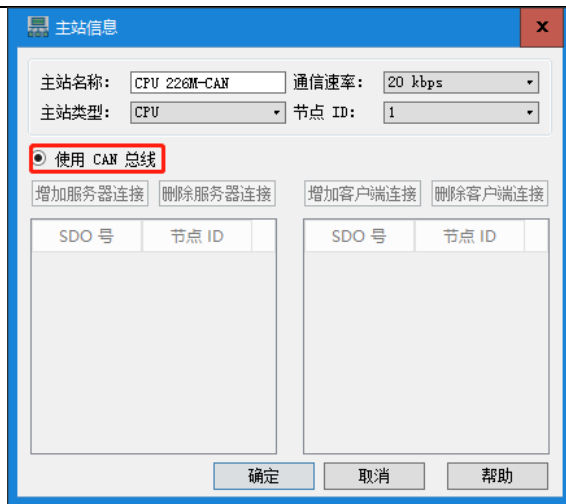
鼠标单击展开右边的设备目录树，打开主站节点，选中与你当前实际设备相符的 CPU 型号，鼠标选中双击或拖拽 CPU 至站节点的 1 号槽中，如下图所示。



<注意> 主站只能放置于 1 号插槽中。

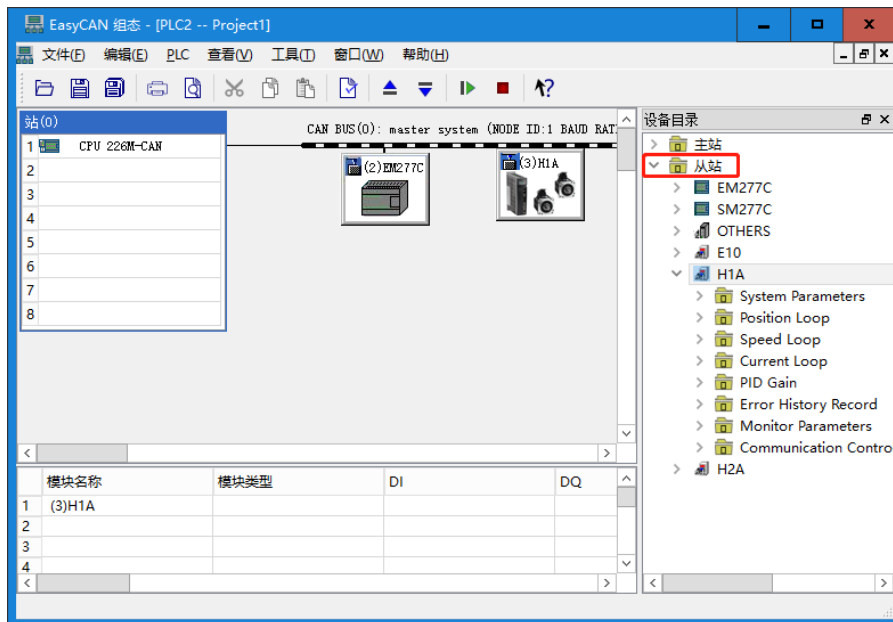
2) 修改主站（CPU）参数

双击插槽 1 中的 CPU，会弹出主站信息对话框，如下图。在此，您可以修改主站名称、类型，选择通信速率和节点 ID，勾选“使用 CAN 总线”即可添加 CAN BUS。



3) 添加从站

鼠标单击展开右边的设备目录树，打开从站节点，选中与你当前实际设备相符的从站设备型号，鼠标选中拖拽进入组态界面 CAN 总线区域放下，系统会自动将从站与 CAN 总线相连。



4) 修改从站参数

双击 CAN 总线下连接的从站图标，即弹出从站信息对话框，如下图：

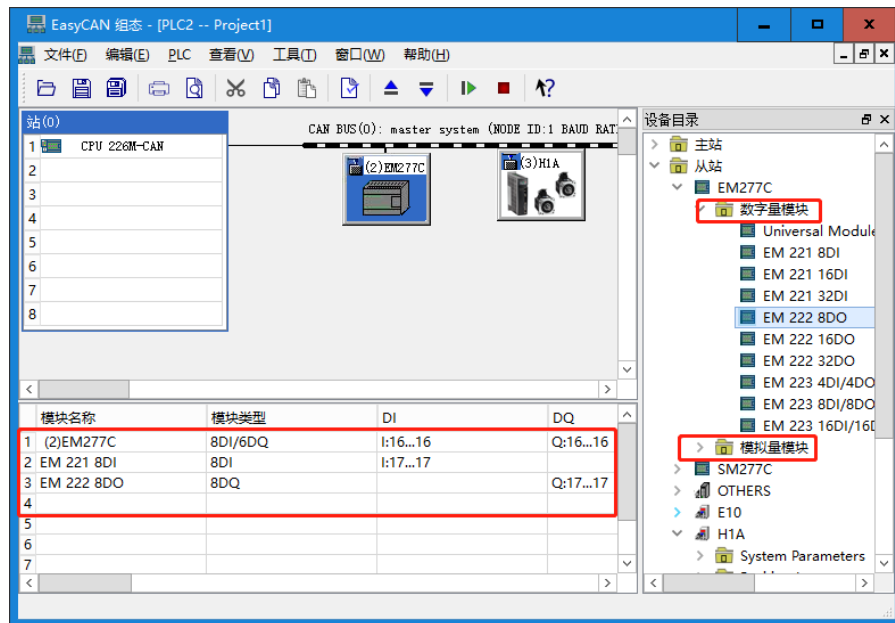


在此，您可以修改从站的名称，节点 ID，选择从站的类型。

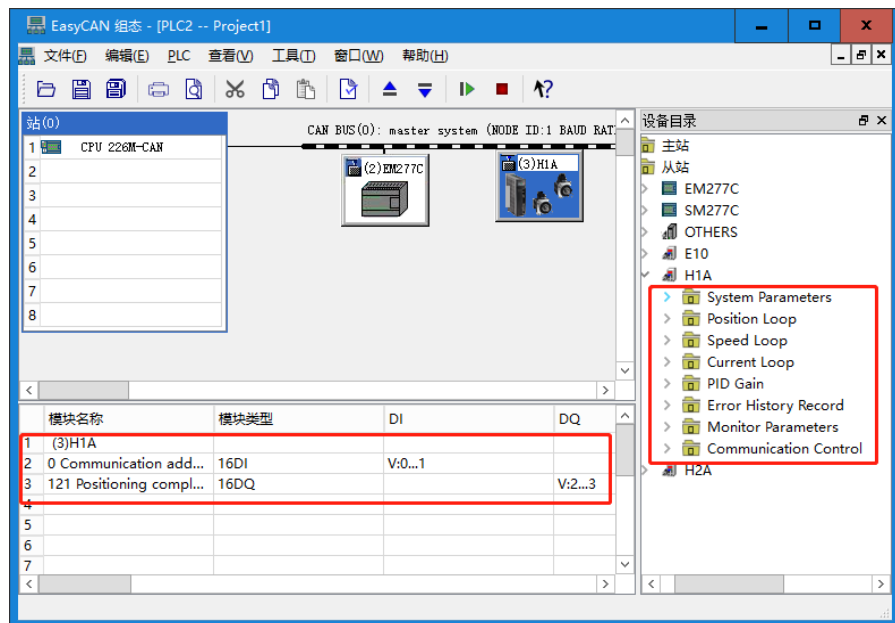
5) 为从站组态模块或为从站组态参数

- 为 EM277C 添加模块：展开设备目录→EM277C 从站节点→数字量模块或模拟量模块节点→

选中与你当前实际设备相符的模块型号双击或使用鼠标拖拽进入模块插槽放下→模块即被添加到当前从站设备下，成功添加的模块会出现在下方的模块列表中。



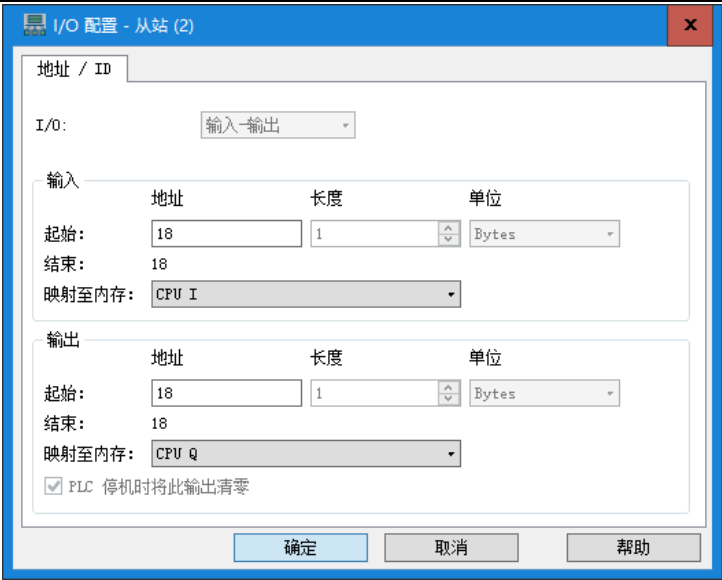
• 为伺服从站组态参数：展开设备目录→伺服从站节点（E10/H1A/H2A/Others）→参数组节点→鼠标双击你需要操作的参数→相关参数即被成功添加。



<备注> 如果您在模块列表找不到对应的模块型号或参数，您可以选择 Universal Module，它允许用户自己定义模块的 IO。

6) 修改 IO 配置

EasyCAN 组态（MagicWorks PLC V1.72 及以上版本）支持用户自定义 IO 的起始地址，并在映射内存中增加了 V 内存，允许用户自己选择映射内存。双击模块列表中的模块，会弹出如下对话框：



在此，您可以修改模块（对 EM277C 从站）或参数（对伺服类型从站）对应 IO 的起始地址和映射内存，不同类型 IO 可映射的内存区域为：

- 数字量输入模块或伺服读类型参数：可映射到 CPU 的 I 内存和 V 内存
- 数字量输出模块或伺服写类型参数：可映射到 CPU 的 Q 内存和 V 内存
- 模拟量输入：可映射到 CPU 的 AI 内存和 V 内存
- 模拟量输出：可映射到 CPU 的 AQ 内存和 V 内存

各内存有效的地址范围为：

内存类型	有效地址范围
I	IB16~IB79
Q	QB16~QB79
AI	AIW64~AIW386（CPU H224X/CPU H226XL/CPU 224I/CPU 226I/CPU 224XP CNI/CPU H228XL） AIW64~AIW318（CPU 226M/H-CAN）
AQ	AQW64~AQW386（CPU H224X/CPU H226XL/CPU 224I/CPU 226I/CPU 224XP CNI/CPU H228XL） AQW64~AQW318（CPU 226M/H-CAN）
V	VB0~Vmax

<说明> 对确定类型的模块或参数，I/O 类型和输入/输出长度是不可修改的，除非您编辑的模块类型是 Universal Module，点确定时，系统会自动检查所选地址的合法性。

7) 编译 EasyCAN 组态

EasyCAN 组态完成后，您可以编译当前工程，生成可下载的数据。有两种方法实现编译：

- （1）保存组态，在 EasyCAN 组态界面单击菜单项“PLC”→“编译”，或点击工具栏上的编译按钮，系统会自动保存并编译组态数据，并且弹出对话框显示编译信息，在此您可以了解到编译中的错误和报警的详细信息。
- （2）保存组态后，回到项目管理器主界面，选中 EasyCAN 组态图标，然后点击编译按钮来编译保存的 EasyCAN 组态，也可以点击全部编译依次编译程序块、数据块、系统块和

EasyCAN 组态。编译信息会显示在下方的输出窗口中。

8) 下载 EasyCAN 组态

连接 PLC 并成功通信的情况下，在项目管理器界面点击下载按钮，打开下载对话框，在选项中勾选 EasyCAN 组态项，点下载，EasyCAN 组态数据就会被下载至目标 PLC 中。

9) 如何将 PLC 中的 EasyCAN 组态恢复至工程

连接 PLC 并成功通信的情况下，在项目管理器界面点击上载按钮，打开上载对话框，在选项中勾选 EasyCAN 组态项，点上载，保存在 PLC 中的 EasyCAN 组态数据就会被上载至工程中。上载完成后，双击 EasyCAN 组态图标，即可查看上载的组态。

10) 清除 PLC 中保存的 EasyCAN 组态

连接 PLC 并成功通信的情况下，在项目管理器界面点击菜单项“PLC”→“清除”，即可打开清除对话框，在选项中勾选 EasyCAN 组态项，点“清除”按钮，保存在 PLC 中的 EasyCAN 组态数据随即被删除。

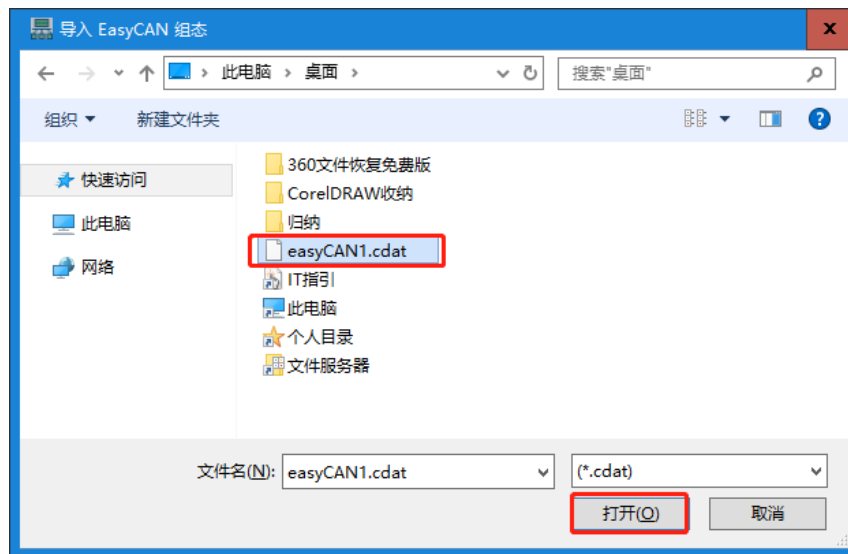
11) EasyCAN 比较

连接 PLC 并成功通信的情况下，在项目管理器界面点击菜单项“PLC”→“比较”，即可打开项目比较对话框，在选项中勾选 EasyCAN 组态项，点击“开始比较”按钮，系统就会比较当前工程中的与保存在 PLC 中的 EasyCAN 组态数据，并将比较的结果显示在下拉框中。

7.3.3 EasyCAN 组态文件的导入导出

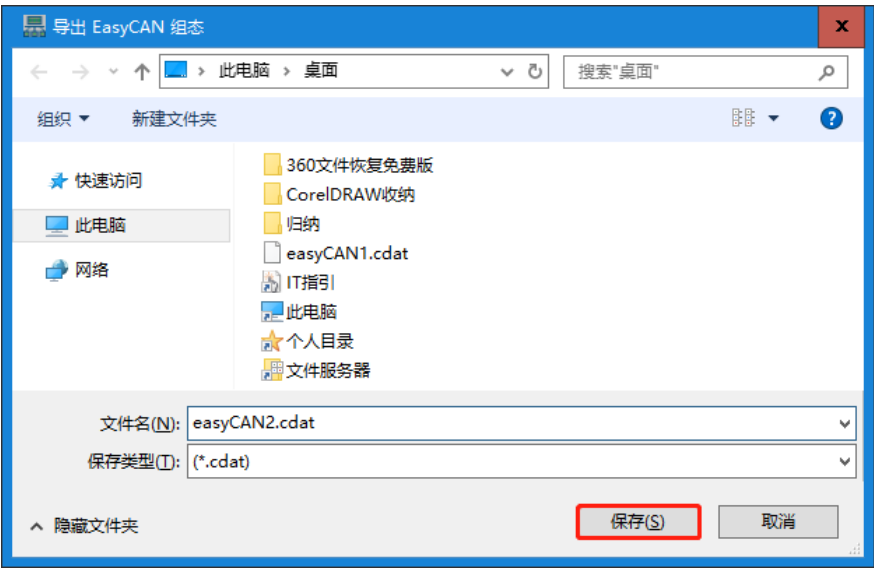
导入 EasyCAN 组态

将 ASCII 文件格式的 EasyCAN 组态文件（后缀为.cdat）内容导入到当前激活的 EasyCAN 组态子窗口。



导出 EasyCAN 组态

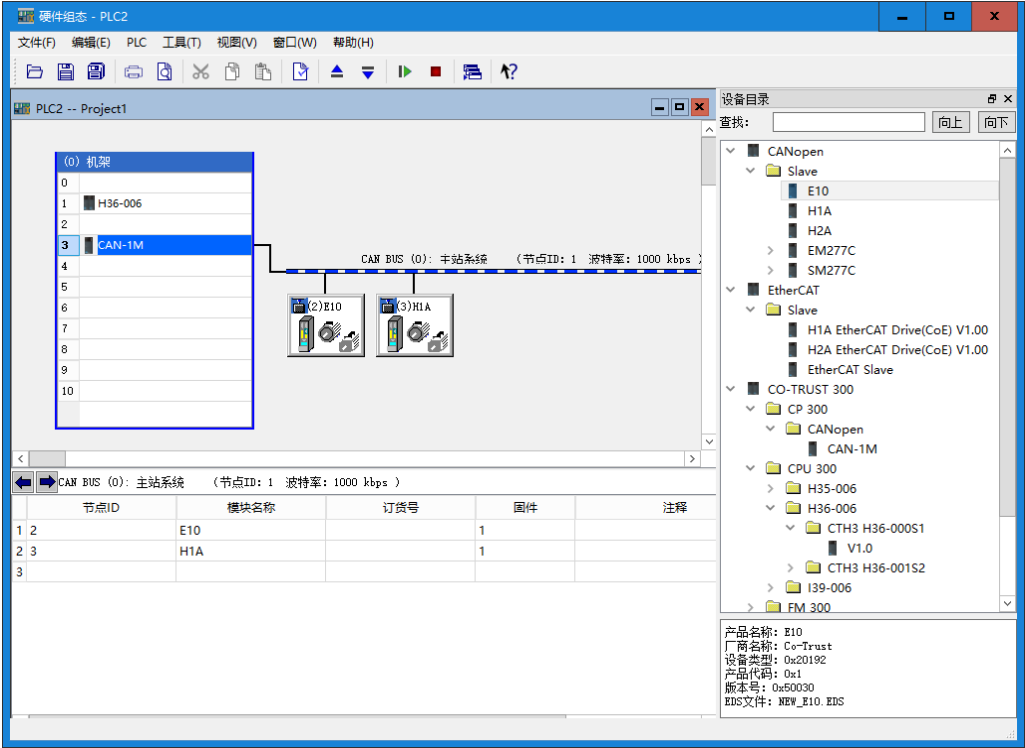
将当前 EasyCAN 组态窗口的内容以 ASCII 文件格式导出到后缀为.cdat 的二进制文件。



7.4 硬件组态

硬件组态软件是用来配置 MagicWorks PLC 通信产品的工具软件，能图形化的组态复杂的 PLC 硬件网络。支持 CTH3 系列所有 PLC 及 200 运控系列 PLC 的硬件组态，给用户提图形化的硬件配置界面，具有良好的扩展性。

硬件组态主界面示意图如下：



7.4.1 一般操作步骤

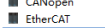
操作步骤:

1) 打开硬件组态界面

展开项目树中的 CPU300 节点，然后在其右侧工作区域中双击硬件组态图标 ，即可进入硬件组态界面。

在硬件组态软件中，您可以为当前项目组态一个硬件网络，请遵循以下步骤：

2) 添加机架

鼠标单击展开硬件组态界面的设备目录树，打开 COTRUST 300 节点 ，然后展开机架

节点 RACK 300  Rail，鼠标选中机架“Rail”拖拽进入工作区域放下，即可创建一个机架。

<备注> 最多允许创建四个机架。

3) 添加电源模块

鼠标单击展开设备目录树，打开 PWR300 节点，鼠标选中 PWR-02 并拖拽进入机架的插槽 0 放下，机架表格中即会出现你插入的模块。

<备注> 电源模块只能放置于 0 号插槽中。

4) 添加 CPU

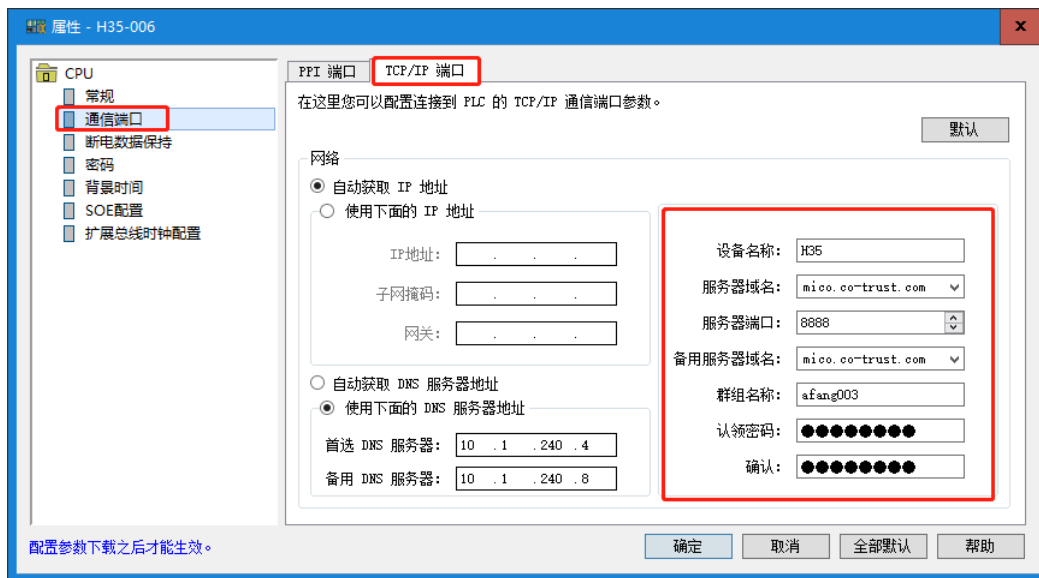
鼠标单击展开设备目录树，打开 CPU 300 节点，选中与你当前实际设备相符的 CPU 型号，鼠标选中拖拽进入机架的插槽 1 放下，机架表格中即会出现你选择的 CPU 信息。

<备注> CPU 只能放置于 1 号插槽中。

5) 修改 CPU 参数

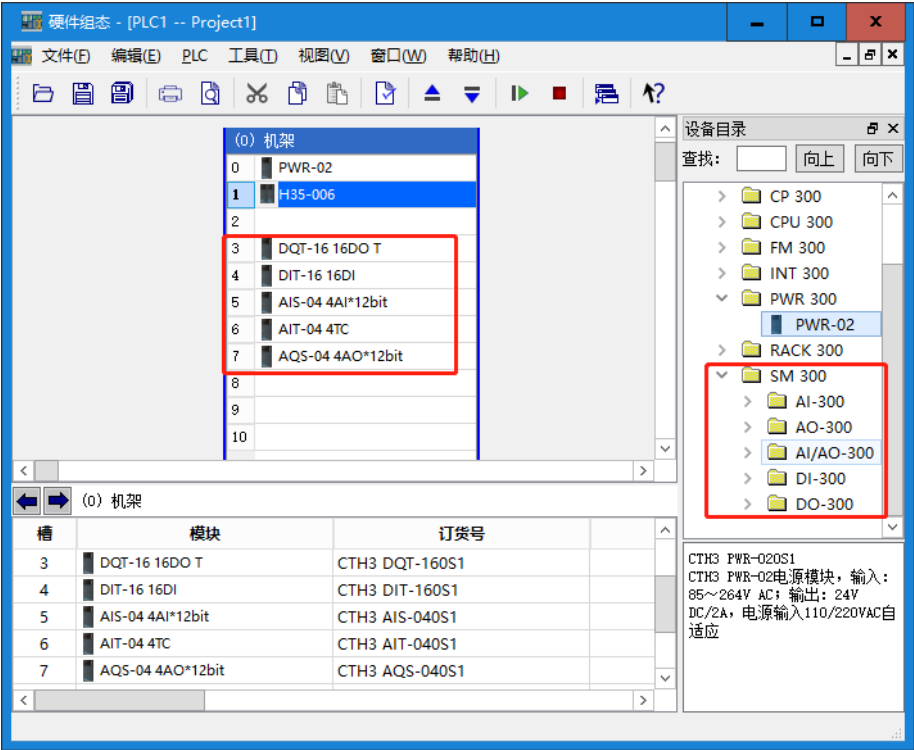
双击插槽 1 中的 CPU300 图标，会弹出 CPU 信息对话框，在该对话框中，您可以为 CPU 设置通信、断电保持、密码、背景时间、SOE 等参数。

若需要进行远程以太网通信，则需在 TCP/IP 通信端口进行参数设置，如下图所示：



6) 添加其他模块

展开设备目录树，可以将 COTRUST 300 下的模块添加到机架上。将模块添加到机架上时，如果不能放置，机架列表会有相应的提示。



硬件诊断:

1、在线/离线状态显示

1) 界面规格

硬件组态里增加菜单 PLC--在线子菜单，工具栏上增加对应的按钮，点击按钮进行在线/离线切换，在线时，页面不允许编辑（禁止插入、粘贴、删除模块，禁止修改模块属性），实时显示每个模块的错误状态，无故障模块所在名称前图标左下方显示一个√，有错误的显示×（如），模块实际不存在或与组态的不一致的显示？。离线时恢复正常状态，页面允许编辑。

2) 每种模块的状态显示

- ◆ 如果模块类型是 CPU
 - 判断系统信息区 514~515 和 SMB500~SMB531 的值（系统信息区 514~515 是判断 CPU 自身的致命错误，SMB500~SMB531 是判断所有模块的状态，模块类型不对或找不到的时候 CPU 的 BF 灯会亮，此时诊断状态应该显示红色）
 - 系统信息区 514~515 为 0 且 SMB500~SMB531 全部都不等于 0x03 也不等于 0x07: √; 其它: ×
- ◆ 如果模块类型是电源或中继
 - 始终为√
- ◆ 如果模块类型是 slot3~10 上的模块
 - 判断 SMB500~SMB531 的值（一个模块对应一个字节）:
 - 0x00: √; 0x03 或 0x07: ?; 其它: ×
- ◆ 如果对象类型是 EtherCAT 总线
 - 判断 SMB401 的值:

0x00: ✓; 其它: ✗

- ◆ 如果对象类型是 CANopen 总线

判断 SMB550/590/630/670/710/750/790/830（一个从站对应其中一个字节）的值：

0x05: ✓; 其它: ✗

- ◆ 如果模块类型是 EtherCAT 从站

判断 SMB402~SMB465 的值：

0x08: ✓; 0x80 或 0x81: ?; 其它: ✗

- ◆ 如果模块类型是 CANopen 从站

判断 SMB551~582 /591~622 /631~662 /671~702 /711~742 /751~782 /791~822 /831~862（一个从站对应一个字节）的值：

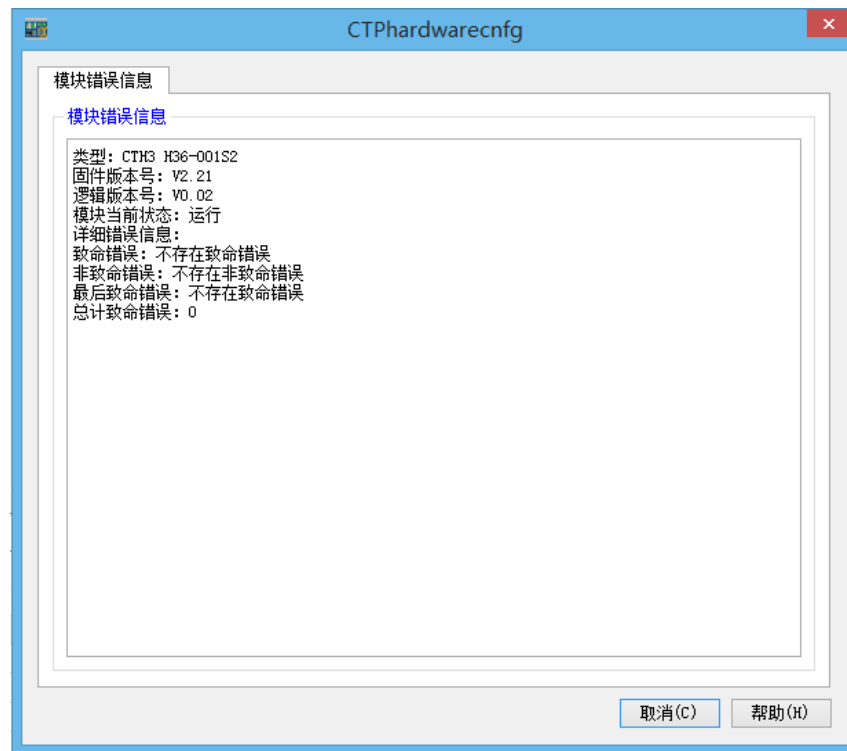
0x05: ✓; 其它: ✗

2、查看诊断信息

界面规格

每个模块右键菜单下增加诊断子菜单，在线模式下使能，离线模式下禁止，点击时诊断菜单弹出对应模块的诊断对话框，如下：

- ◆ 没有使能 EtherCAT 的 CPU 的诊断对话框如下：



类型: CPU 的订货号 (ModuleCommonData.mlfb)

固件版本号: SMW102

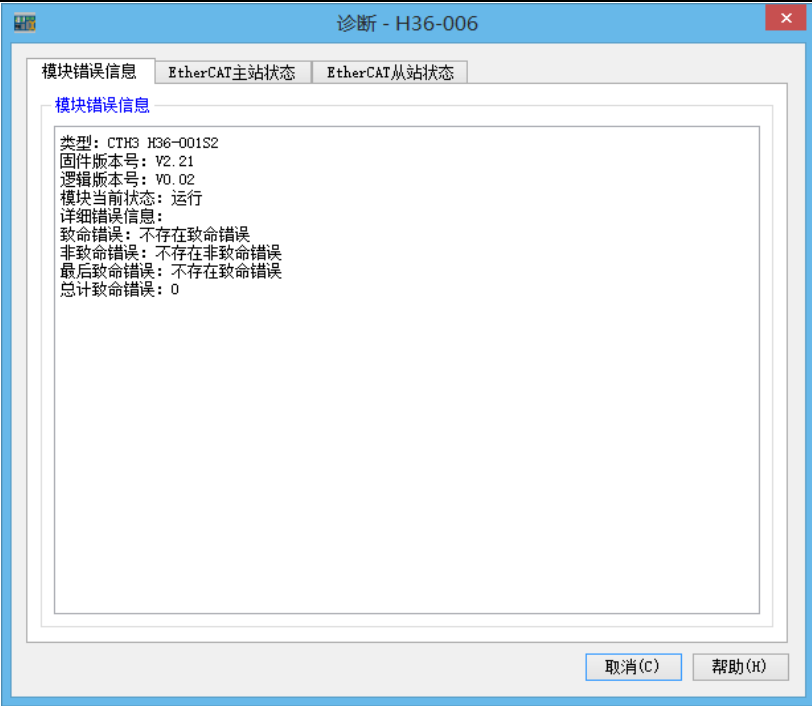
硬件版本号: SMW104

模块当前状态: 运行/停止

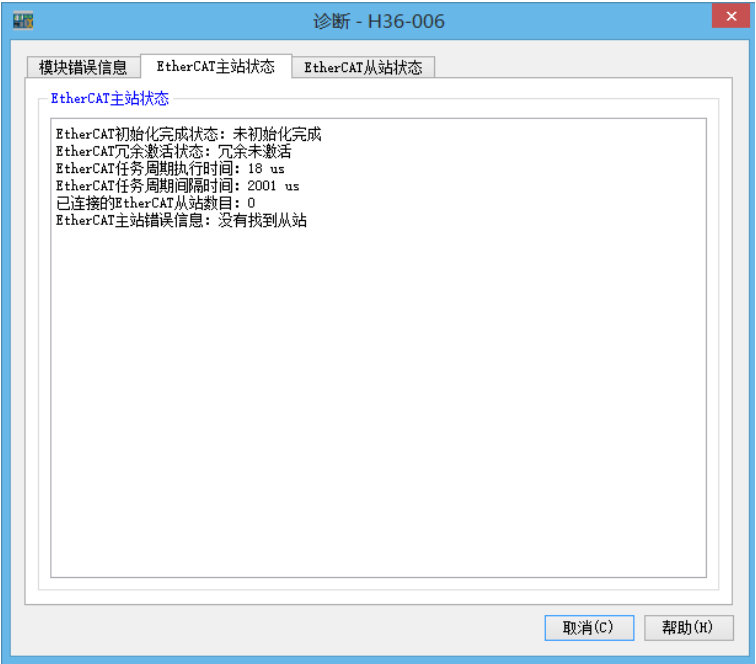
详细错误信息: 系统信息区 470-477 和 512~515 的值（具体解析可参见 plcinfodlg.cpp 文件）

- ◆ 使能了 EtherCAT 主站功能的 CPU 诊断对话框：

模块错误信息页面包含信息如下，此页面的数据解析跟普通 CPU 一样：

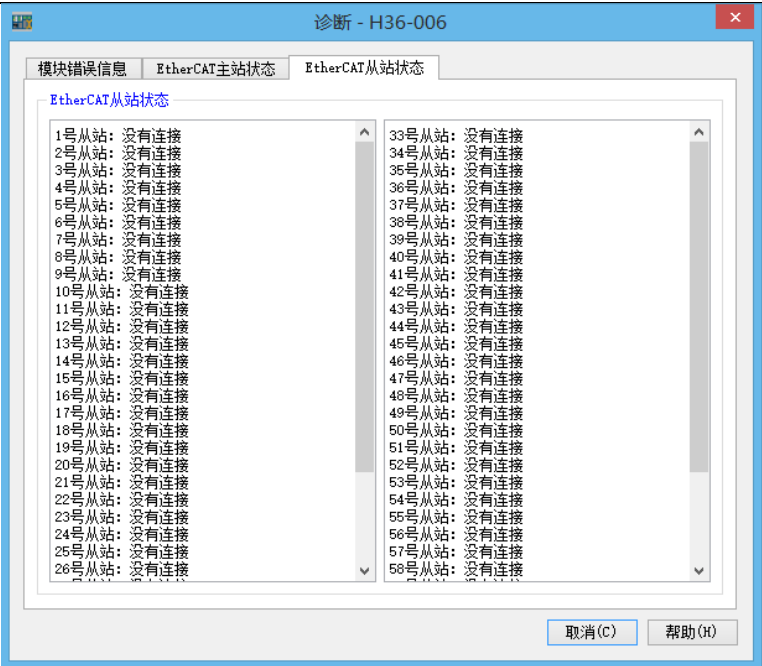


EtherCAT 主站状态页面包含信息如下：



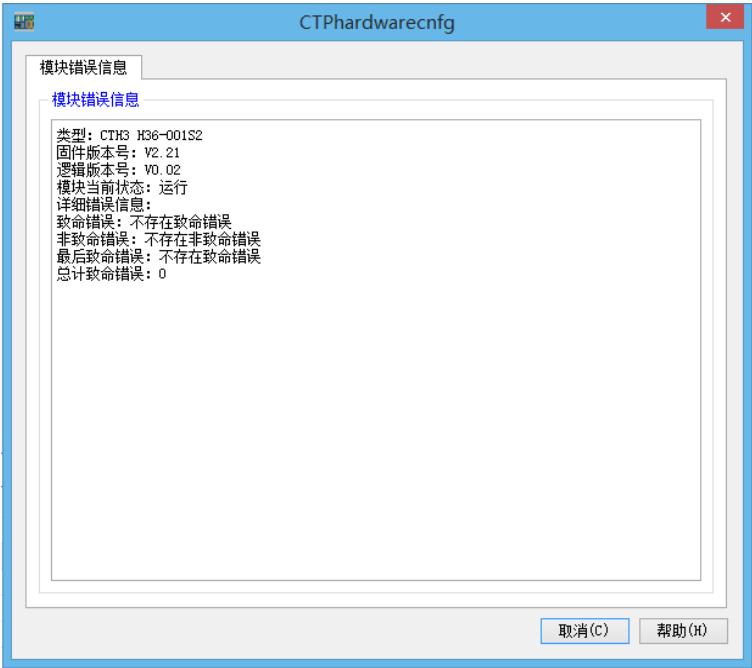
主站状态	说明	主站状态	说明
初始化完成状态	SM390.0	任务周期间隔时间	SMD396
冗余激活状态	SM390.1	找到的从站个数	SMB400
任务周期执行时间	SMD392	主站错误信息	SMB401

EtherCAT 从站状态页面包含信息如下（包含当前主站下的所有从站的状态，每个主站最多可组态 64 个从站）：



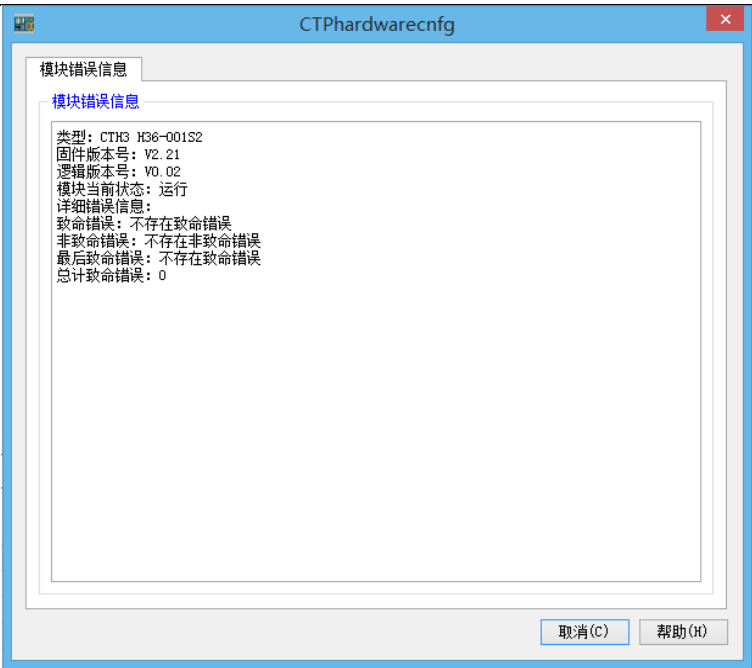
解析格式见 SMB402~SMB465，每个字节对应一个从站。

◆ 位于插槽 slot3~10 除 CAN 主站以外的模块的诊断对话框如下：



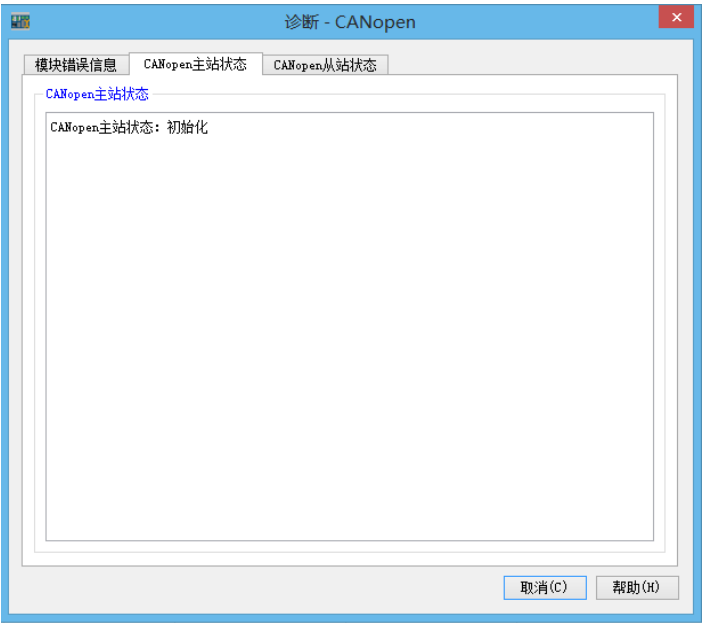
参数	说明
类型	模块的订货号（ModuleCommonData.mlfb）
固件版本号	SMB1003
逻辑版本号	模块没有硬件版本号，固定为 V1.00
错误计数（累计）	SMD1004
详细错误信息	SMB500
内部诊断信息	SMB872~SMB905，诊断信息的格式跟模块类型有关，见规格文档“内部诊断信息”章节的说明

◆ CANopen 主站模块诊断对话框，模块错误信息页面如下：



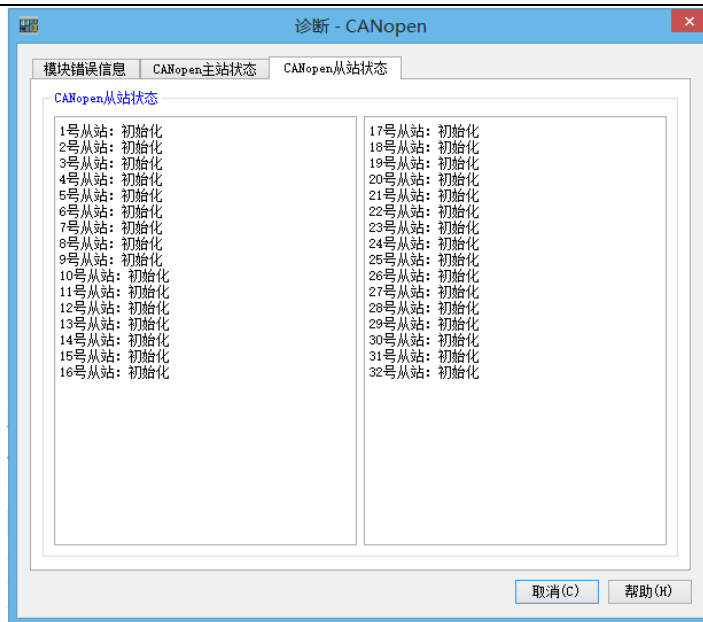
参数	说明
类型	模块的订货号（ModuleCommonData.mlfb）
固件版本号	SMB583
逻辑版本号	模块没有硬件版本号，固定为 V1.00
错误计数（累计）	SMD1004
详细错误信息	SMB500
内部诊断信息	SMB872~SMB905，诊断信息的格式跟模块类型有关

CANopen 主站状态页面包含信息如下：



主站状态：SMB550

CANopen 从站状态页面包含信息如下（包含当前主站下的所有从站的状态，每个主站最多可组态 32 个从站）：



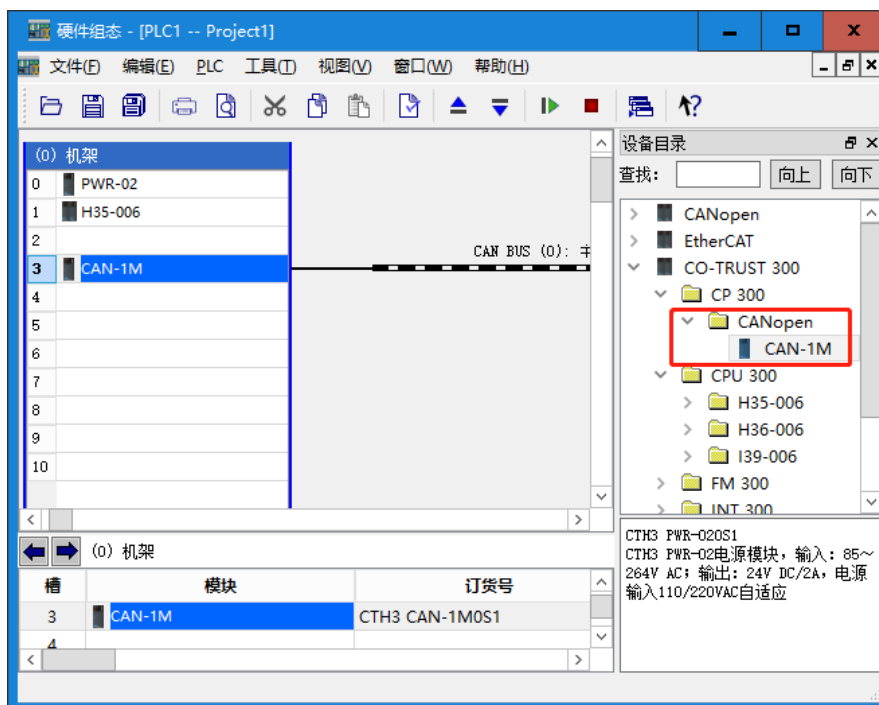
从站状态：SMB551

特别说明：EtherCAT 从站和 CANopen 从站暂不支持诊断（在线时要显示错误状态，但不支持右键——查看诊断信息），从站的诊断信息需要在所属的主站里查看。

7.4.2 组态 CANopen 通信

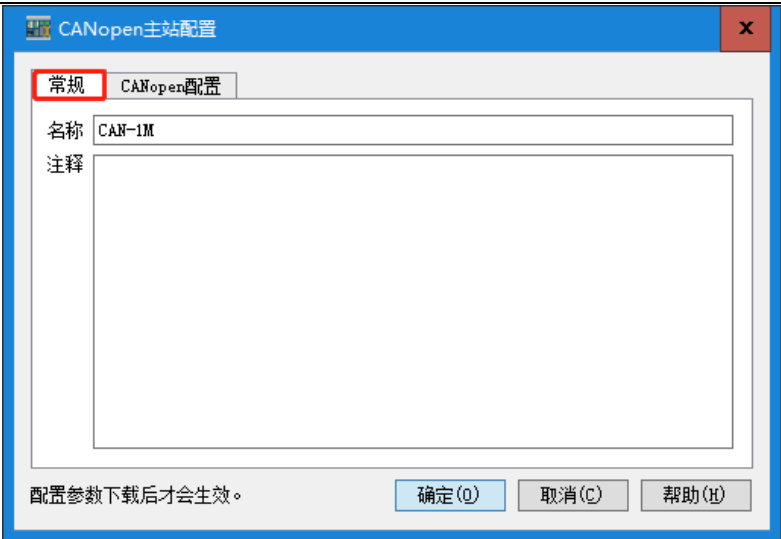
1、添加 CAN 主站模块

展开设备目录树，打开 COTRUST 300 下的 CP300 主站，鼠标选中 CAN-1M 并拖拽进入机架模块列表的 3~10 槽放下，系统会自动添加总线。

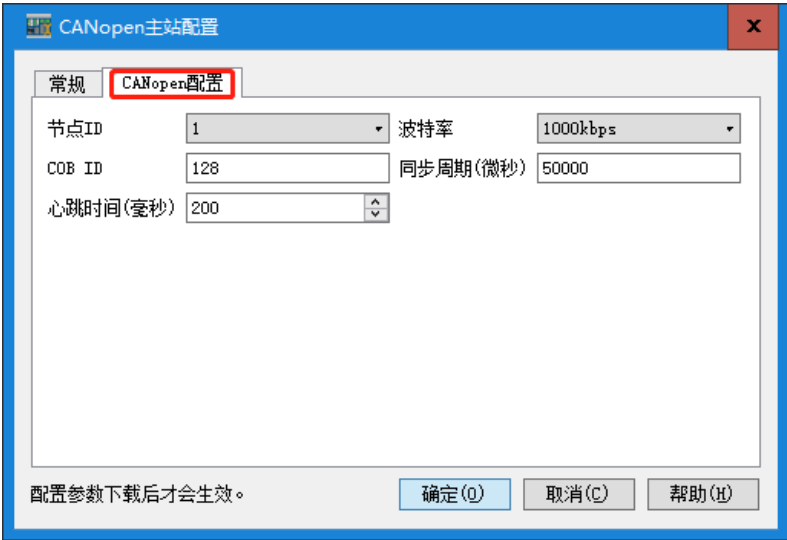


2、配置 CAN 主站模块

在机架上双击已经成功添加的 CAN-1M 模块，弹出如下对话框，即可配置该模块：
在“常规”选项卡中可以编辑主站模块的名称，以及注释。



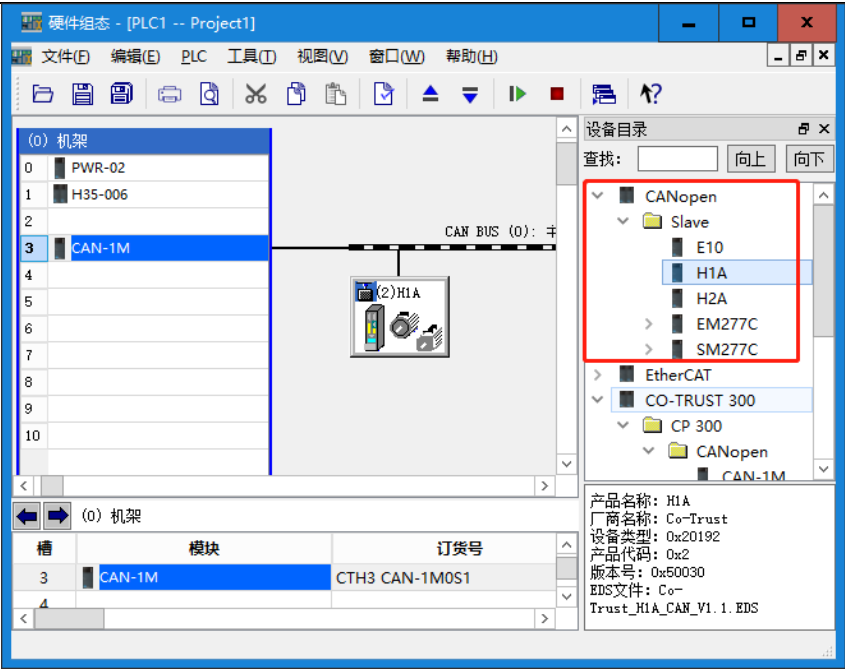
在“CANopen 配置”选项卡中可以配置主站模块通信参数：



参数	说明
节点 ID	主站在 CANopen 网络上的节点地址
波特率	设置 CANopen 总线的通讯速率
COB ID	CAN 对象标识符地址信息
同步周期（微秒）	设定发送同步信息的周期。当从站 PDO 传输类型为同步传输时，可根据需要设置其同期周期，默认为 50000us
心跳时间（毫秒）	设定 CAN 主站发送心跳报文的周期

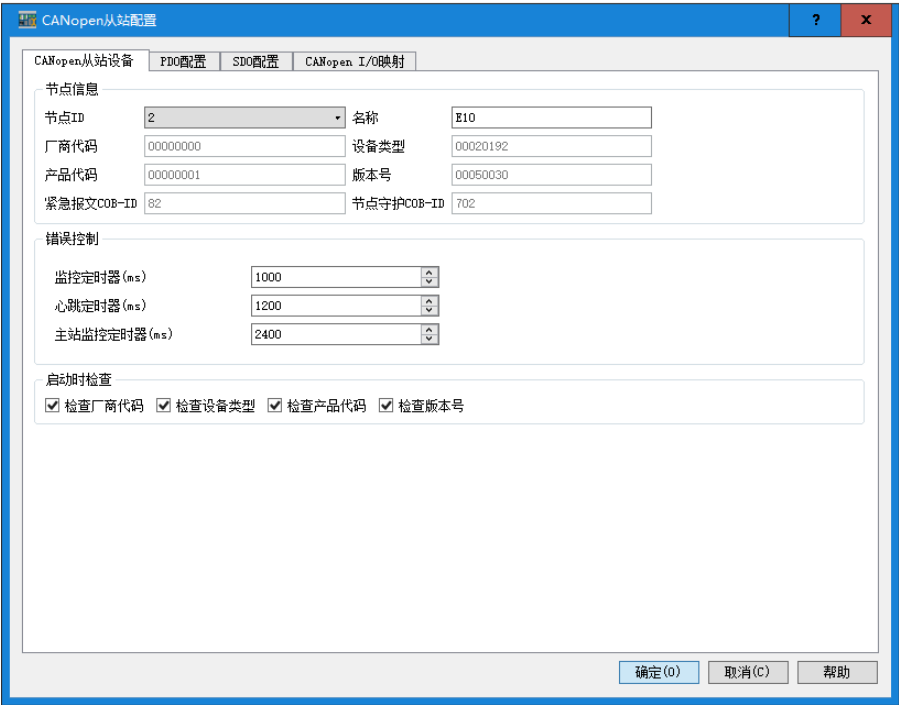
3、添加 CAN 从站

展开设备目录树的 CANopen 节点，打开从站节点 Slave，选中与你当前实际设备相符的从站设备型号鼠标选中拖拽进入组态界面 CAN 区域放下，系统会自动将从站与 CAN 总线相连。



4、为 CAN 从站（伺服）组态参数

双击 CAN 总线上的从站（E10/H1A/H2A）图标，即弹出如下图所示的从站信息对话框，具体的参数配置参考如下描述。



（1）“CANopen 从站设备”选项卡

在“CANopen 从站配置”对话框的“CANopen 从站设备”选项卡中可以设置节点 ID、错误控制参数及 EDS 一致性检查，参考如下描述：

节点信息

参数	说明
节点 ID	节点在 CANopen 网络上的地址，用户可修改此节点 ID
名称	节点的设备名称，只读参数

产商代码	提供商的 ID 代码，只读参数
设备类型	节点的设备类型代码，只读参数
产品代码	节点的产品代码，其中 1 表示 E10 伺服驱动器、2 表示 H1A 伺服驱动器、3 表示 H2A 伺服驱动器，只读参数
版本号	节点 CANopen 功能的修订版本，只读参数
紧急报文 COB-ID	用来标识 CANopen 网络中各节点的紧急报文。节点紧急报文 COB-ID=0x80+节点站号，只读参数
节点守护 COB-ID	用来标识 CANopen 网络中各节点的节点管理报文，只读参数

错误控制协议：设置错误控制的方式，支持 Heartbeat 协议。参数说明如下：

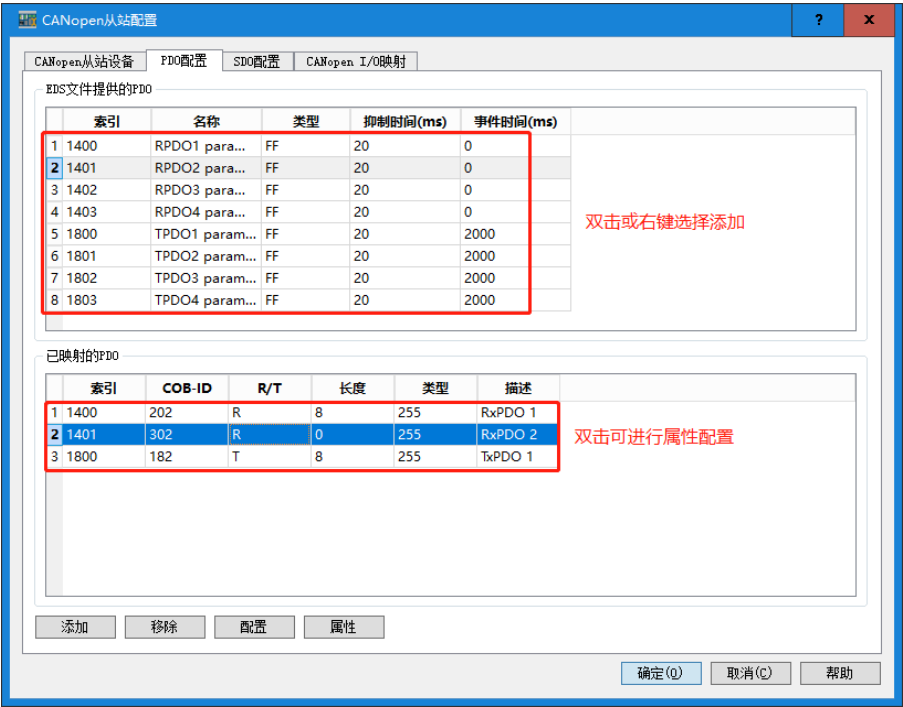
参数（ms）	说明
监控定时器	从站监控时间，指从站在本监控时间内没有接到主站的心跳即认为掉线
心跳定时器	从站心跳时间，从站按“从站心跳时间”周期发送 Heartbeat 报文给主站
主站监控定时器	主站监控时间，如果主站在“主站监控时间”内没有收到从站的 Heartbeat 报文，主站就认为该从站掉线

选择 EDS 参数一致性检查

您可以选择要不要进行厂商代码、设备类型、产品代码和版本号的一致性检查，默认要检查。如果这四项有勾选，组态下载至 PLC 后，CANopen 从站会将组态中（EDS 文件中）的这四个参数与从站固件中相应参数进行对比，如果不一致，会停止 CANopen 通信，同时在 SM 内存给出错误信息。如果去掉勾选，组态下载后不会进行检查，这样即使组态软件中使用的 EDS 文件版本与固件中不一致，通信成功但可能会导致读写参数的结果不对。

（2）“PDO 配置”选项卡

在“CANopen 从站配置”对话框的“PDO 配置”选项卡进行 PDO 配置，请按照图示步骤操作：



<第 1 步>：在 PDO 配置对话框中通过双击或右键选择“添加”即可将所选 PDO 添加到“已映射的 PDO”列表中。

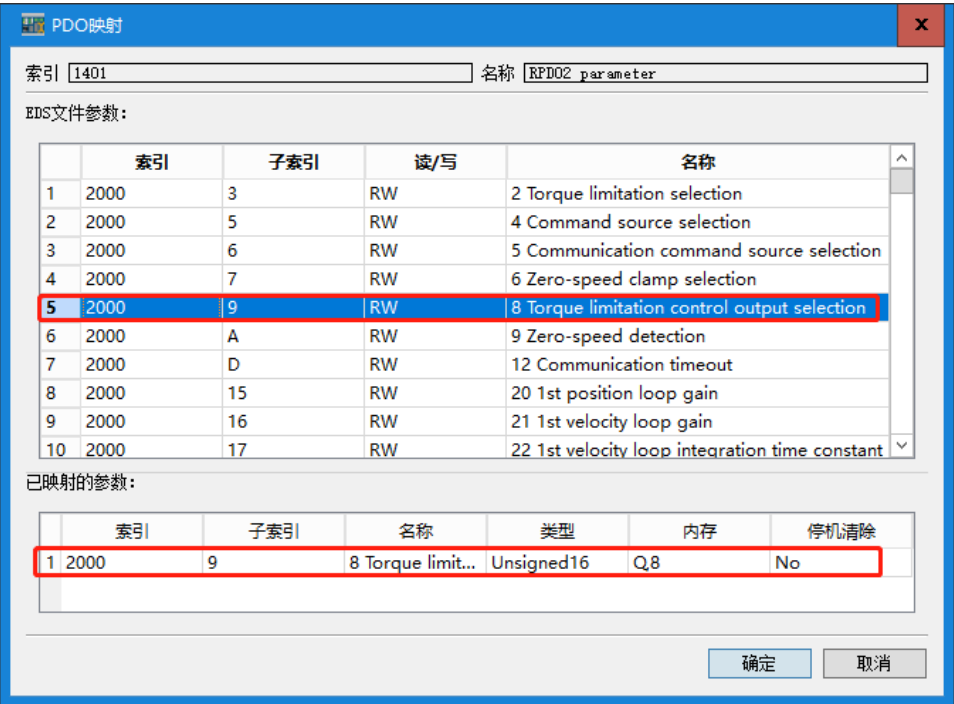
<第 2 步>: 在“已映射的 PDO”列表中编辑 PDO 属性或配置 PDO。



- 索引 1400 ~ 1403: 接收 PDO 通信参数
- 索引 1800 ~ 1803: 传送 PDO 通信参数
- PDO 映射: 从站节点中需要和主站进行通信的参数
- RxPDO: CANopen 主站去写从站映射在 RxPDO 中的参数
- TxPDO: CANopen 主站去读从站映射在 TxPDO 中的参数

“配置”操作

在“已映射的 PDO”中选择一个对象后右键选择“配置”或点击该对话框下方的“配置”按钮即可打开该项的配置对话框，在该对话框中可以配置该组的参数如下图所示：



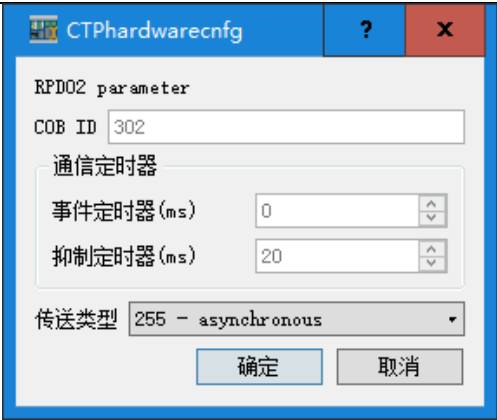
PDO 配置完成后，您可以在 CANopen I/O 映射页面查看所有 PDO 参数的映射地址。

“移除”操作

在“已映射的 PDO”中选择一个对象后右键选择“移除”或点击该对话框下方的“移除”按钮即可将该项从列表中删除。

“属性”操作

在“已映射的 PDO”中选择一个对象后右键选择“属性”或点击该对话框下方的“属性”按钮即可打开该项的属性对话框，如下所示：



COB ID: CAN 对象标识符地址信息。

事件定时器: 设定事件触发定时器的时间。

抑制定时器: 设定两个连续调用数据对象的传输服务之间的最小时间间隔。

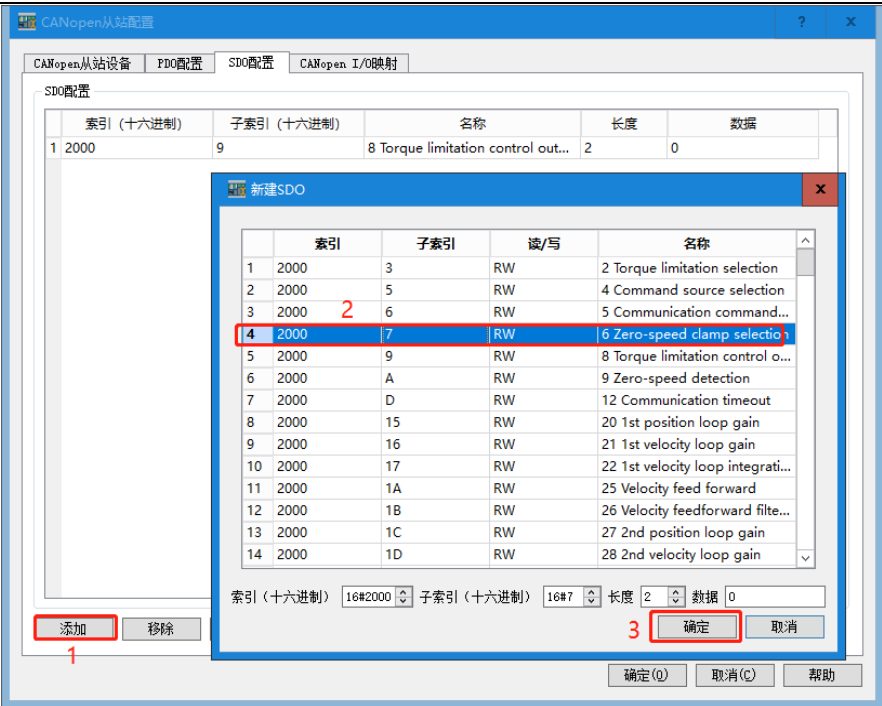
传送类型: 选择 PDO 数据传输类型（参考下表）。

传输类型	触发 PDO 的条件 (B = both needed, O = one or both)			PDO 传输
	SYNC	RTR	Event	
0	B		B	同步，非循环
1~240	O		-	同步，循环
241~251	-		-	Reserved
252	B	B	-	同步，在 RTR 之后
253	-	O		异步，在 RTR 之后
254	-	O	O	异步，制造商特定事件
255	-	O	O	异步，设备子协议特定事件
说明： SYNC 一接收到 SYNC-object。 RTR 一接收到远程帧。 Event 一例如数值改变或者定时器中断。 传输类型为 1 到 240 时，该数字代表两个 PDO 之间的 SYNC 对象的数目。				

（3）“SDO 配置”选项卡

自动 SDO 配置: 在 CANopen 组态中可以编辑 SDO 报文，下载至主站后，在每次初始化时完成固有的初始化流程后，紧接着就会执行自动 SDO。

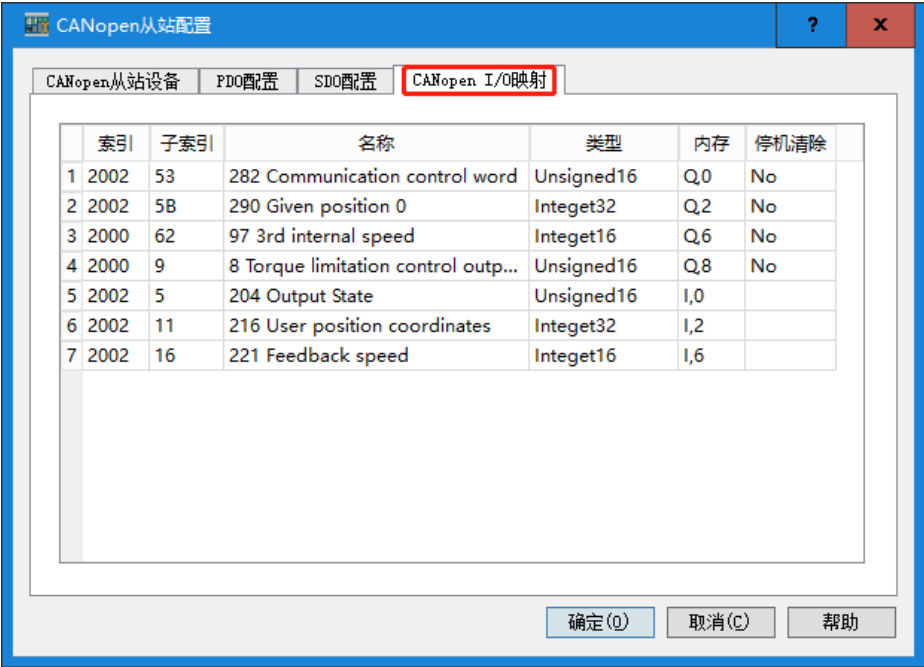
在“CANopen 从站配置”对话框的“SDO 配置”选项卡中可以配置 SDO，操作如下：



点击“添加”在弹出的界面中选中要添加的 SDO，点击“确定”即可添加一条 SDO，点击“编辑”按钮可以对选中的自动 SDO 进行修改，点击“移除”按钮可以删除配置的 SDO。自动 SDO 只能写参数，不能读参数，自动 SDO 只在从站由预运行状态进入运行状态前对从站写一次。此功能常用来初始化从站参数。

(4) “CANopen I/O 映射” 选项卡

“CANopen 从站配置”对话框的“CANopen I/O 映射”选项卡中将显示已经配置好的 IO/参数及映射的内存。



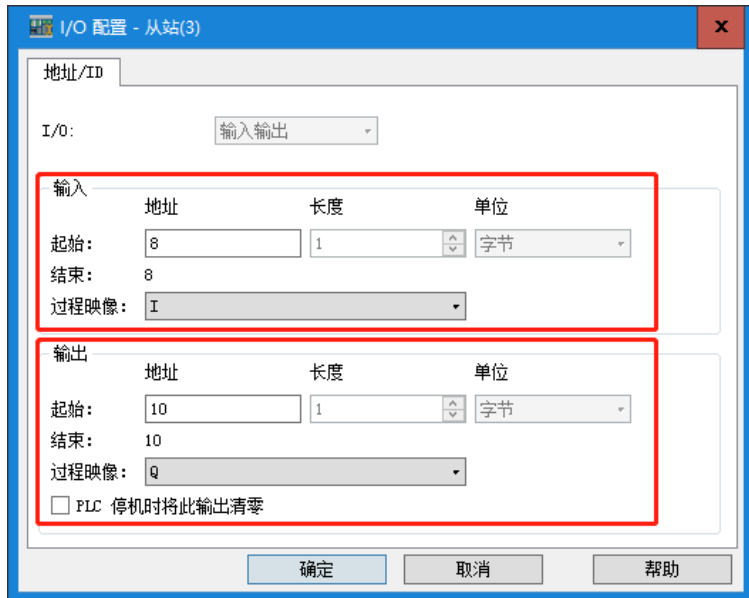
5、为 CAN 从站（EM277C）添加模块

展开设备目录树，打开所选从站 EM277C 节点下的数字量模块或模拟量模块节点，选择与你当前实际设备相符的模块型号，使用鼠标拖拽进入模块插槽放下，模块即被添加到当前从站下。

您也可以通过双击设备目录树上的模块节点将模块按顺序插入到当前从站下。如果您在模块列表中找不到对应的模块型号，您可以选择 **Universal Module**，它允许用户自己定义模块的 IO。

6、修改 IO 配置

硬件组态软件支持用户自定义 IO 的起始地址，允许用户自己选择映射内存。双击模块列表中的非空白行，会弹出如下对话框：



在此，您可以修改模块（对 EM277C 从站）或参数（对伺服类型从站）对应 IO 的起始地址和映射内存，不同类型 IO 可映射的内存区域为：

数字量输入：可映射到 CPU 的 I 内存

数字量输出：可映射到 CPU 的 Q 内存

模拟量输入：可映射到 CPU 的 AI 内存

模拟量输出：可映射到 CPU 的 AQ 内存

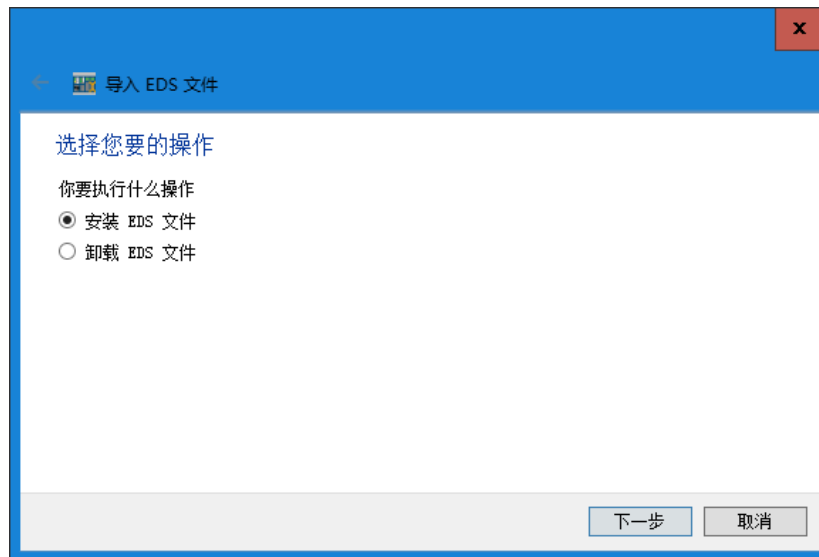
伺服读类型参数：可映射到 CPU 的 I/AI 内存

伺服写类型参数：可映射到 CPU 的 Q/AQ 内存

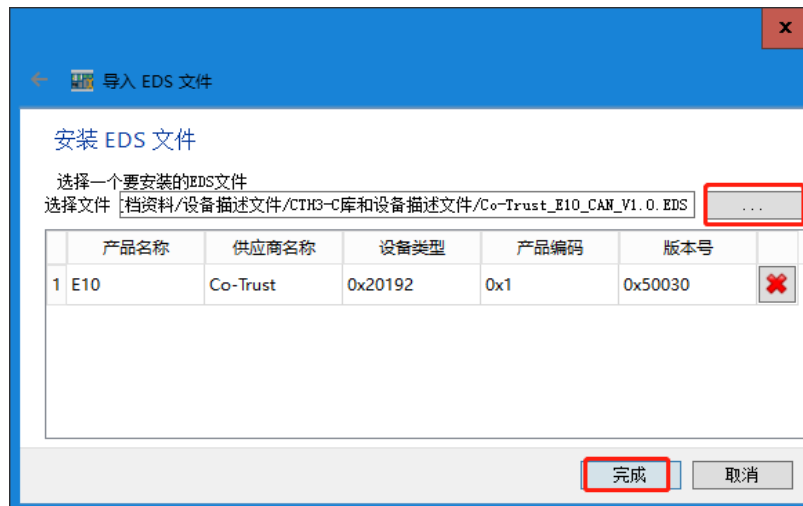
<说明> 对确定类型的模块或参数，I/O 类型和输入/输出长度是不可修改的，除非您编辑的模块类型是 **Universal Module**，点确定时，系统会自动检查所选地址的合法性。

7、导入 EDS 文件

安装 CANopen 从站设备的 EDS 文件：选择菜单项“工具”→“导入 EDS 文件（E）”打开如下对话框：

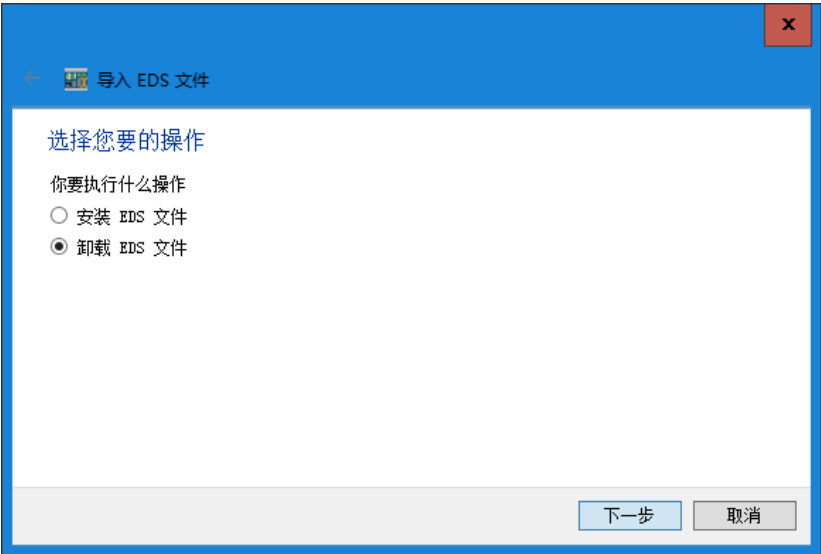


在以上窗口中选择“安装 EDS 文件”，然后点击“下一步”：



点击“...”选择需要安装的 EDS 文件，然后点击“添加”按钮，即可完成 EDS 文件安装。

卸载 CANopen 从站设备的 EDS 文件：选择菜单项“工具”→“导入 EDS 文件（E）”，打开如下对话框，在该对话框中选中“卸载 EDS 文件”，然后点击“下一步”：



勾选需要删除的 EDS 文件，如下图所示，然后点击“完成”按钮即可成功删除该 EDS 文件。

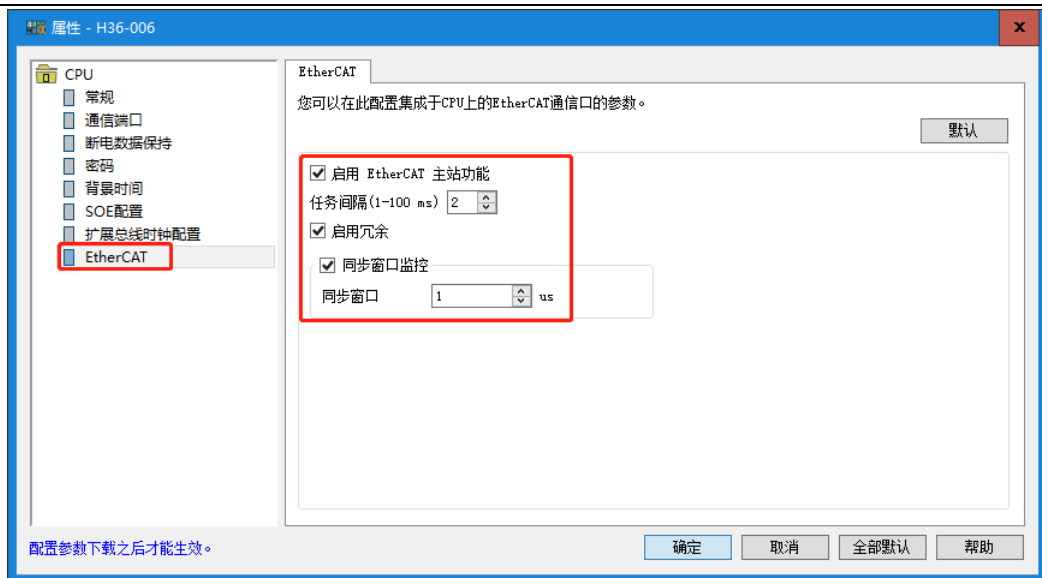


7.4.3 组态 EtherCAT 通信

以下将介绍 EtherCAT 组态的一般操作步骤：

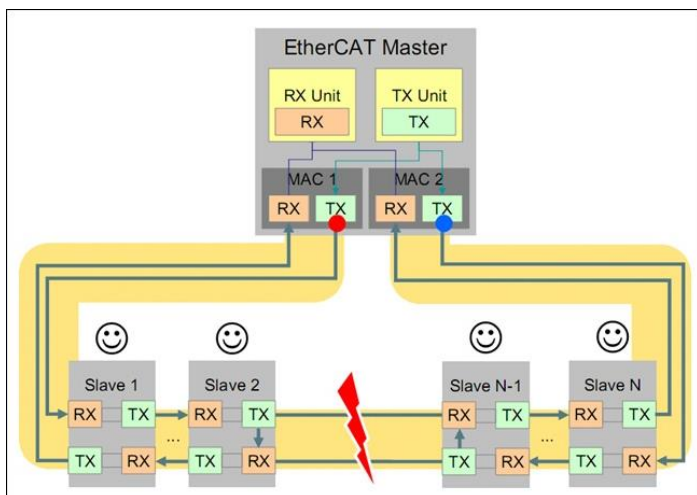
1、添加 EtherCAT 主站

展开设备目录树，打开 COTRUST 300 下的 CPU300 节点，打开 H36-006 节点，选择 H36-000S1 或 H36-001S2 并拖拽进入机架模块列表的 1 号槽放下，然后双击机架上的 H36-000S1 或 H36-001S2 打开其属性对话框，选择 EtherCAT 标签，勾选“启用 EtherCAT 主站功能”后点击“确定”，即可启用 EtherCAT 功能。



【启用冗余】：如果使用环拓扑支持冗余，则激活此选项。使用此功能，即使电缆破碎，EtherCAT网络功能依然可以保持。

EtherCAT 环拓扑（冗余）见下图描述：



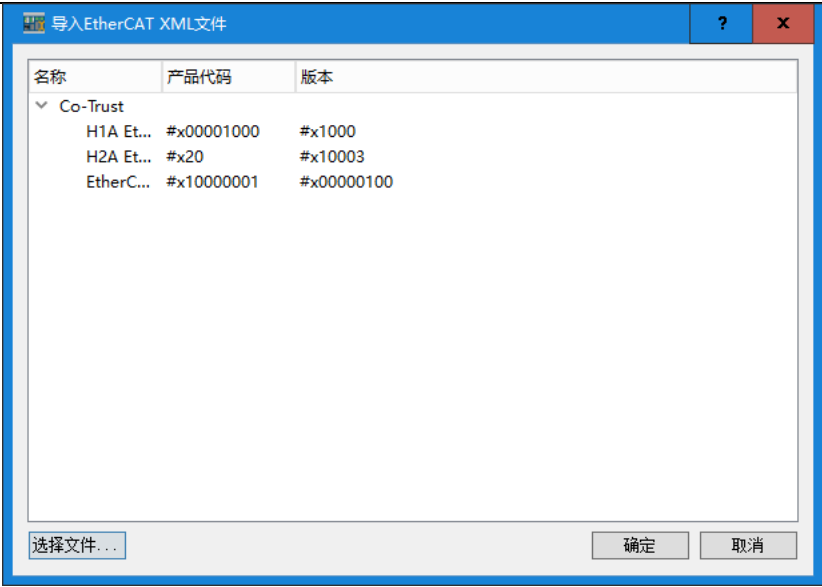
【周期时间】：指在此之后可以传送新的数据报文的时间段。如果‘分布时钟’功能被激活，则主站循环时间将被传至从站时钟。因此可以实现精确的数据交换同步，这在分布式过程中要求同步动作时尤其重要（例如多个伺服轴执行同时联动任务）。因此可以在网络范围内得到非常精确的、信号抖动小于 1 微妙的时钟基。

【同步偏移量】：这个值允许在 EtherCAT 从站的 PLC 循环周期内修正同步中断的偏移。通常情况下 PLC 的任务周期比从站同步中断晚启动 20%。这意味着 PLC 任务会滞后循环周期的 80% 并且没有数据丢失。

【同步窗口】：同步窗口监控的时间。

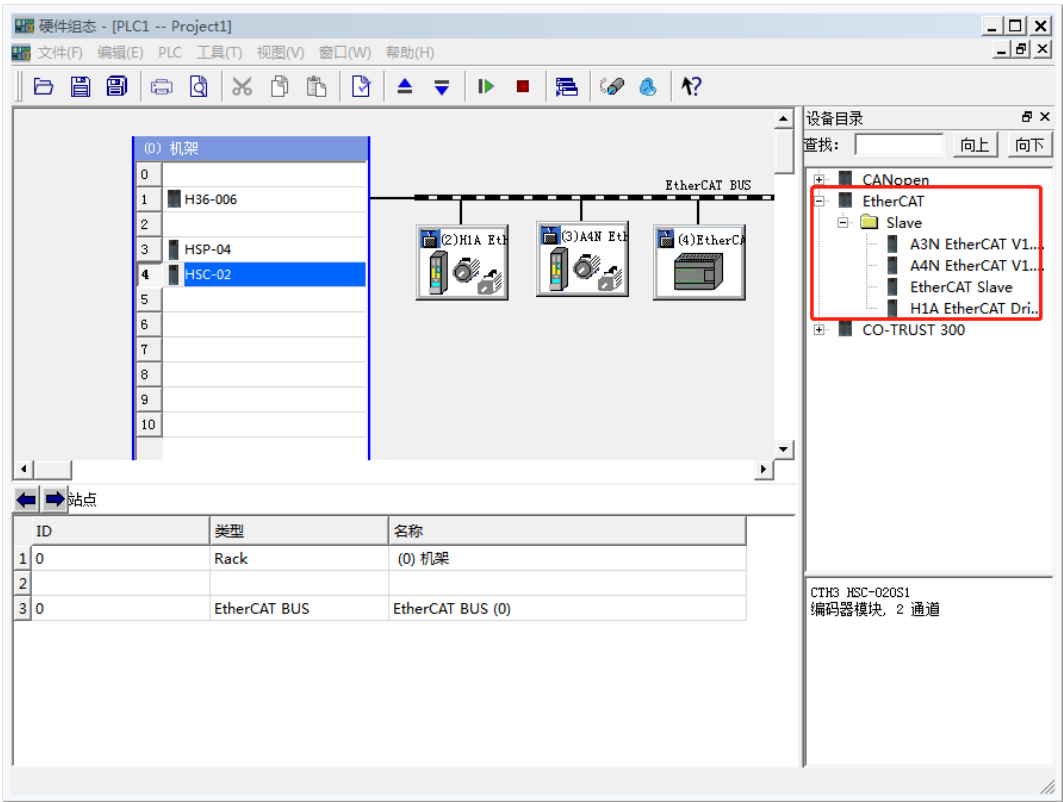
2、导入 XML 文件

添加第三方 EtherCAT 标准从站，选择菜单项“工具”→“导入 EtherCAT XML 文件(X)”打开如下对话框，点击“选择文件...”，选择要导入的 XML 文件，点击确定即完成添加。



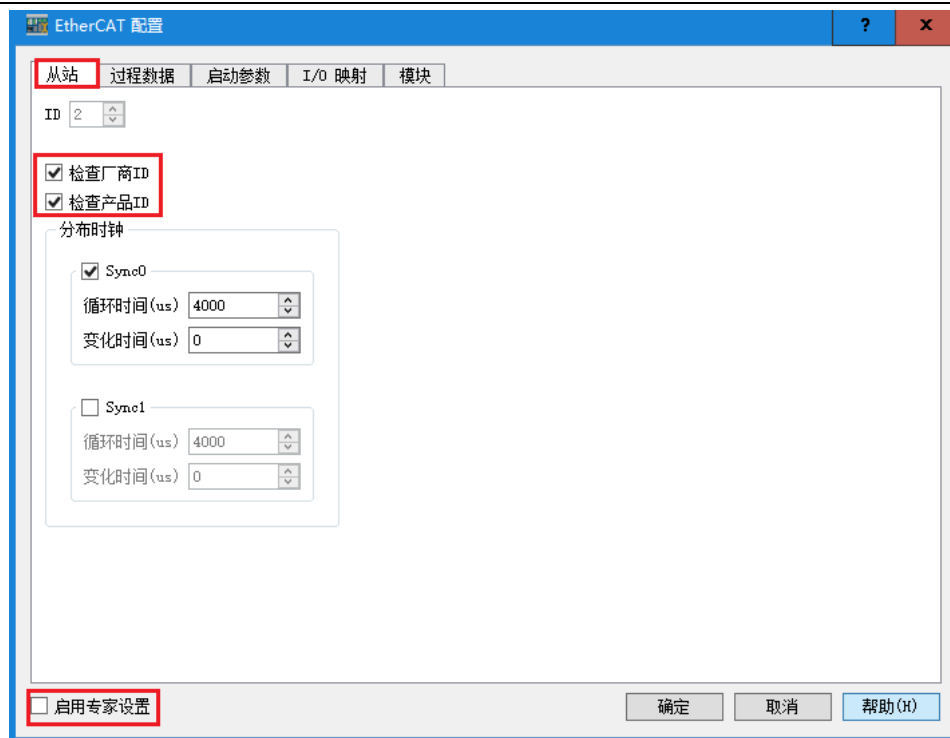
3、添加 EtherCAT 从站

展开设备目录树的 EtherCAT 节点，打开从站节点 Slave，选中与你当前实际设备相符的从站设备型号，鼠标选中拖拽进入组态界面的 EtherCAT 总线区域放下，从站即被成功添加。



4、配置 EtherCAT 从站

双击 EtherCAT 总线上的从站图标，即弹出如下图所示的从站信息对话框，具体的参数配置参考如下描述。



“从站”选项卡

1) 检查厂商 ID/检查产品 ID

勾选状态项默认在系统启动时，根据当前的配置，检查供应商 ID 和/或者产品 ID。可以选择不启用该选项，以关掉检查。

2) Sync0（同步 0）

使能同步 0：如果该选项被选中，使用“sync0”同步单元运用。一个同步单元描述了一套同步交换的过程数据。

循环时间(us)：显示当前设置的周期时间。

变化时间(us)：输入微秒制的期望时间。

3) Sync1（同步 1）

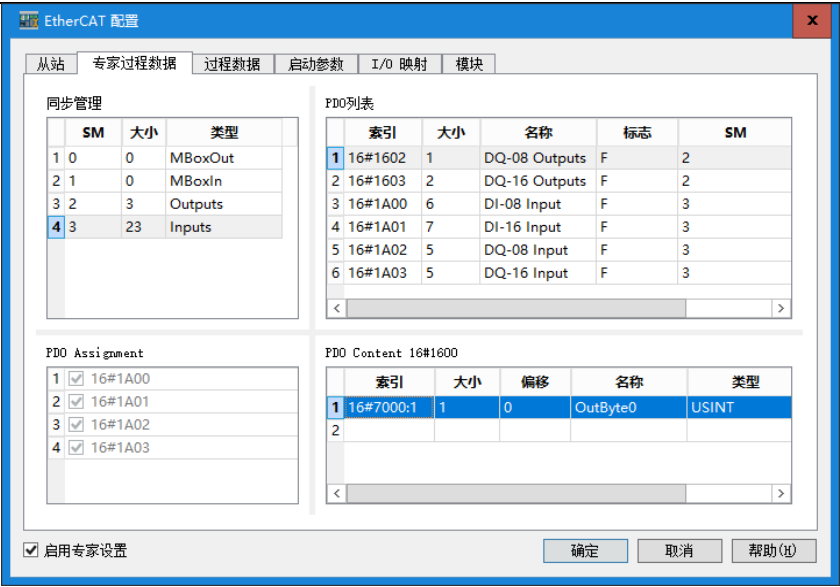
使能同步 1：如果该选项被选中，使用“sync1”同步单元运用。一个同步单元描述了一套同步交换的过程数据。

循环时间(us)：显示当前设置的周期时间。

变化时间(us)：输入微秒制的期望时间。

4) 启用专家设置：勾选该项即可激活专家过程数据功能。

“专家过程数据”选项卡



- **同步管理：** PDO 数据（大小和类型）的同步管理器列表。
- **PDO Assignments：** 分配给所选择的同步管理器的 PDO 列表。创建的 PDO 和 IO 通道可以被激活。此处仅 PDO 可以被使能或不使能。
- **PDO 列表：** 在设备描述文件中的所有 PDO 列表。可以通过上下文菜单命令删除、编辑或添加单个 PDO。

若添加的从站为 H1A/H2A，在 PDO 列表中选择一项，通过右键菜单选择“编辑”，在弹出的对话框中可以编辑 PDO 属性：

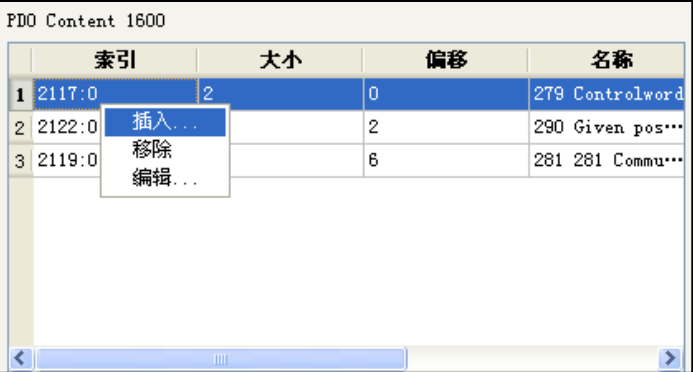


参数	说明
名称	PDO 项的名称
索引	PDO 的索引
SyncUnit	PDO 应当分配给同步管理器的 ID
Exclude PDOs	定义的一个排除列表。如果 PDO Assignment 中的一个 PDO 被使能，则其它将显示为灰色且不被检测
RxPDO (Output)	从从站传送到主站的 PDO
TxPDO (Input)	从主站传送到从站的 PDO
强制	PDO 是必要的，且需要在 PDO Assignment 中强制
固定	PDO 内容已经固定且不能更改。不能在 PDO Content 中添加项

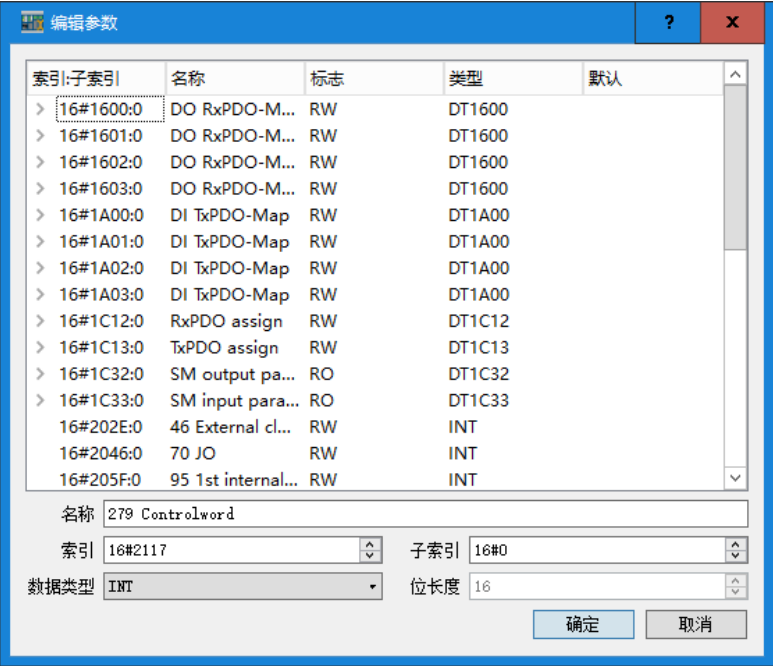
- **PDO Content：** 显示 PDO 列表中所选的 PDO 内容。可以通过上下文菜单命令执行删除、

编辑或添加项。

在 PDO 列表中选择一项后，在 PDO Content 表格中会显示此 PDO 的参数。在此表格中通过右键菜单可以插入或者编辑参数：



选择“编辑”或“插入”后在弹出对话框中配置参数的属性

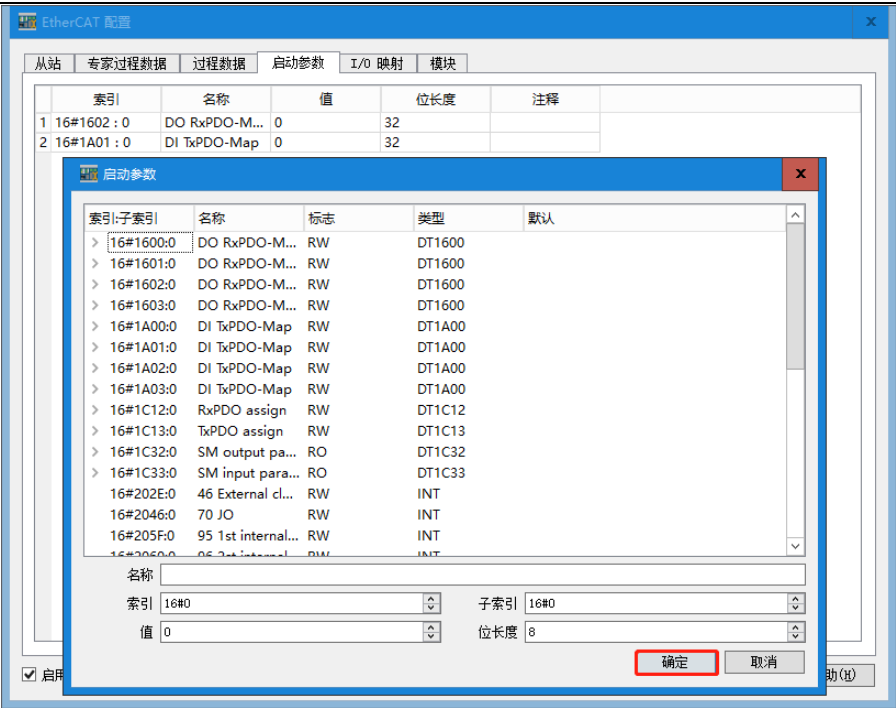


“过程数据”选项卡

它显示了设备描述文件所描述的从站输入和输出过程数据。设备中被选择的输入（可读）和输出（可写）。将在 I/O 映射对话框中作为 PLC 的输出和输入获取。为了更改当前的选择，必须先用鼠标点击当前选择的数据前面的勾选框，以取消选择，此后可以设置另一个。

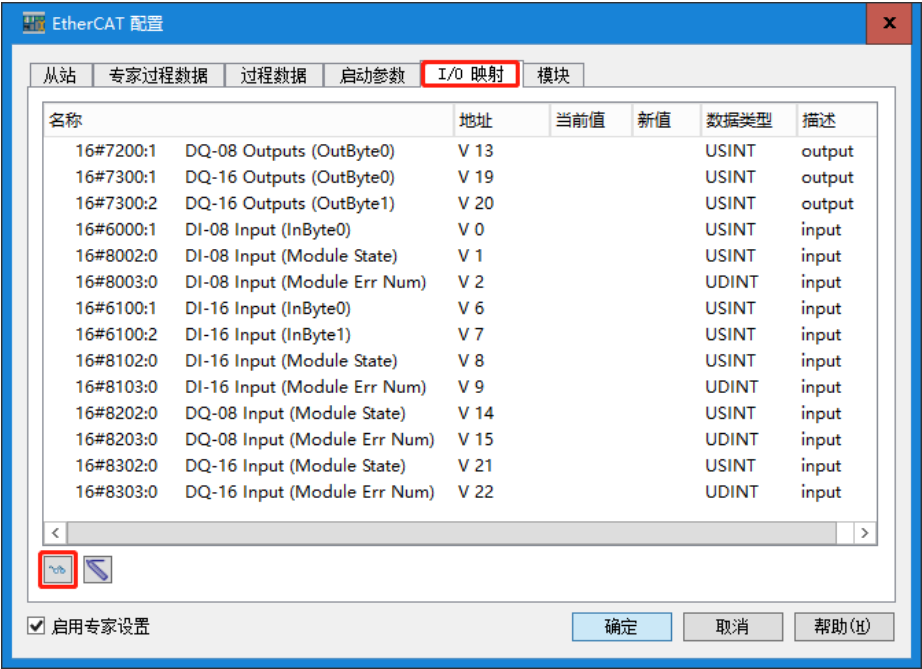
“启动参数”选项卡

在此选项卡中可以插入启动参数。设备启动时，将以“启动参数”中设置的值执行操作。
如下图，在“启动参数”选项卡下右键选择“插入”，在弹出的“启动参数”窗口中选中要添加的参数项，点击“确定”，即可插入参数：



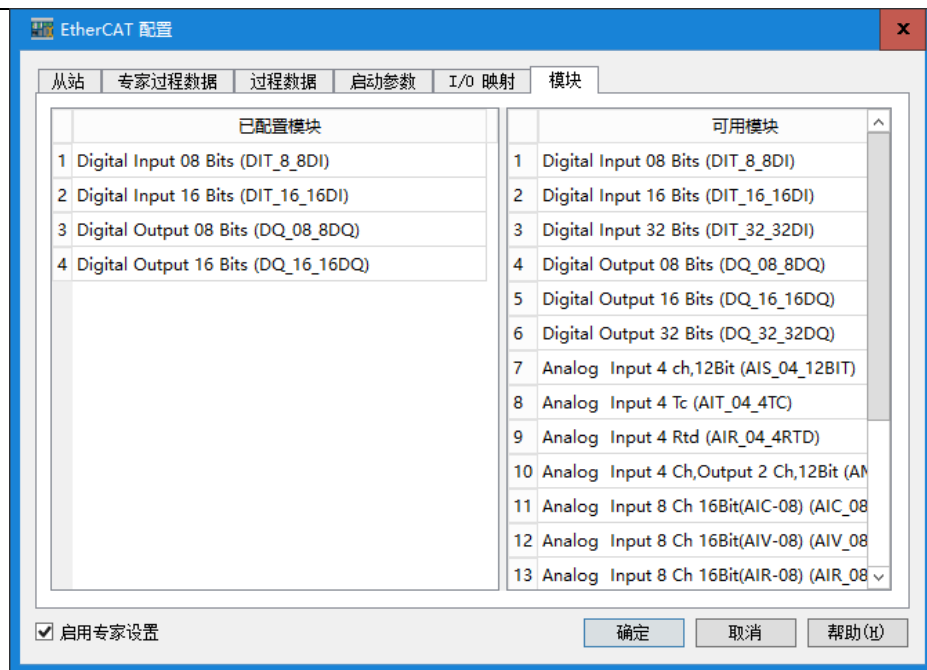
“I/O 映射” 选项卡

显示已经配置的 IO 参数及其映射，双击地址列可以编辑其地址：



点击 “I/O 映射” 页面下方的监控按钮可以监控已经配置的参数。

“模块” 选项卡



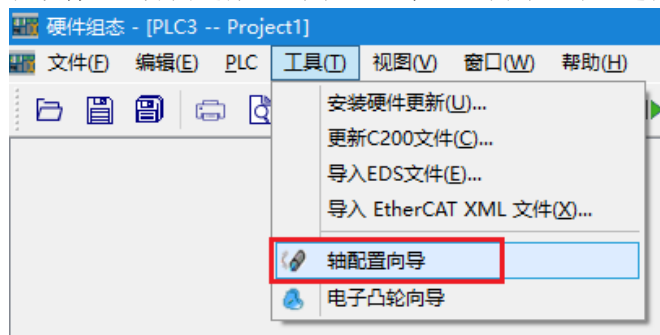
“模块”选项卡显示已配置模块及可用模块，双击“可用模块”栏目的模块选项或右键点击选择“添加”，可将相应模块配置到左侧“已配置模块”栏目。

备注：伺服类从站无需添加模块。

7.4.4 轴配置向导

MagicWorksPLC V2.17 及以上版本支持轴配置。

在硬件组态界面选择“工具”->“轴配置向导”即可进行轴配置，如下图所示。



轴配置向导界面如下所示：



提示

轴配置中所有轴所配置轴个数总和上限为 64，单类轴个数上限也为 64。

【坐标】



- 硬件组态：“设备名称”为添加该轴时的轴名称；“轴号”为该轴添加的顺序 ID，不可修改，随轴的添加依次递增，若删除某条轴，则新添加的轴轴号为上一次删除轴轴号；“机架号”及“槽号”为该轴对应 HSP 模块所在的机架号及槽号；“通道号”为该轴在 HSP 模块中对应的通道号，范围 0~3（仅脉冲轴及 EtherCAT 轴有通道选项）。
- 坐标：“轴类型”可选择“直线轴”或“旋转轴”，对于丝杆类型的往复运行机构，其行程是有限的，而需要知道其在丝杆行程范围内的绝对位置，此时选择“直线轴”较好；若是单方向运转类型的旋转轴，采用线性模式容易出现位置计数溢出，从而计算错误，故选择“旋转轴”较好。其余选项均为默认不可选；“用户坐标来源”固定可选“指令输出”；“电机方向有效电平”可选择“低电平”或“高电平”。

坐标

转换

速度

限制

回原

量纲转换

该页中，你将为轴配置量纲转换和单位，请根据提示设置。

131072

增量 < == > 电机转动

1

5

电机转动 < == > 齿轮输出转动

1

1

齿轮输出转动 < == > 应用单元

360

速率变化类型

请根据需要为该轴选择合适的加速度模式：

☒ 梯形

☐ 二次

☐ Sin^2

【转换】

在“转换”界面可根据提示设置纲量转换和速率变化类型。将上图中的“应用单元”设为 360，当程序指令要求伺服运行 1 个单位时，齿轮端将会选择 1/360 圈（轴运动 1 度），伺服电机端会旋转 5*1/360 圈（电机旋转 5 度），以此类推。按照实际机械结构设定对应参数（即电子齿轮比）之后，就可按照应用系统的运动距离物理单位输入距离命令，使参数控制直观易懂。

举例：伺服电机经过减速机 4:1 的机械减速后，驱动导程为 5.5mm 的丝杆，即丝杆转动一圈，丝杆滑块运动 5.5mm。量纲转换设置如下：

量纲转换

该页中，你将为轴配置量纲转换和单位，请根据提示设置。

131072

增量 < == > 电机转动

1

4

电机转动 < == > 齿轮输出转动

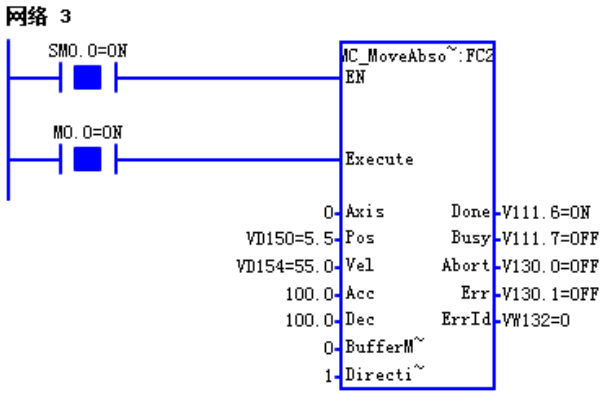
1

10

齿轮输出转动 < == > 应用单元

55

设置后的量纲即可作为 MC 控制指令的物理参数量纲，将上述设置下载到程序块中，编程使得工件运动到 5.5mm 坐标处，位置命名单位即为设备物理坐标单位，程序示例如下：



【速度】

坐标

转换

速度

限制

I/O映射

电机最大速度(MAX_SPEED):

4000000

电机最小速度(MIN_SPEED):

100

停/启速度:

100

最大加/减速度

10000000

最小加/减速度

100

在此页面可设置电机的最大/最小速度、停止/启动速度以及最大和最小加/减速度。程序运行过程中，若设定速度超过组态设置，库指令将报警。错误代码如下：

Errorcode	意义
5	速度取值太小（大于最小速度）
6	速度取值太大（大于最大速度）
7	加速度取值太小
8	加速度取值太大
9	减速度取值太小
10	减速度取值太大

【限制】

坐标

转换

速度

限制

回原

☒ 正向限位

正向输入：

I0.1

有效电平：

低电平

☒ 负向限位

负向输入：

I0.0

有效电平：

低电平

☒ 软件限位

正向限位值：

1000

负向限位值：

0

错误和限位响应

停止模式：

立即停止

减速度：

100000

最大距离：

100

勾选“正向限位”和“负向限位”可对正/负向输入进行修改设置，“有效电平”可选择“高电平”或“低电平”；勾选“软件限位”可对正/负向限位值进行修改设置；“错误和限位响应”的“停止模式”可选“立即停止”或“减速停止”。

【回原】

坐标

转换

速度

限制

回原

在该页面将对轴的回原模式进行配置，请根据需要进行配置

17 -VE LS without Index Pulse

+VE: 正方向

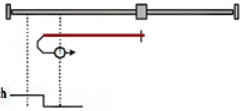
LS: 限位开关

-VE: 反方向

MS: 原点开关

Seek Speed

Creep Speed



☒ 使用回原开关

回原信号：

I0.2

回原有效电平：

高电平

搜索速度：

1000

爬行速度：

100

回原加速度：

100000

在此页面可对轴的回原模式进行配置，此配置仅脉冲轴支持，选择下拉选项可查看回原模式，

共有 17 中回原模式可选。
以上配置完成后点击“完成”按钮在弹出的窗口（如下图所示）选择“是”，完成轴配置。

【I/O 映射】

坐标转换速度限制I/O映射

☒ 自动映射

输入：

	周期对象	对象值	地址	类型
1	status word (in.wStatusWord)	16#6041:16#00	V19	UINT
2	actual position (diActPosition)	16#6064:16#00	V21	DINT
3	actual velocity (diActVelocity)	16#606C:16#00	V25	DINT
4	actual torque (wActTorque)	16#6077:16#00	V29	INT
5	Modes of operation display (OP)	16#6061:16#00	V31	INT
6	digital inputs (in.dwDigitalInputs)	16#60FD:16#00	V43	UDINT
7	Touch Probe Status	16#60B9:16#00	V33	UINT
8	Touch Probe 1 rising edge	16#60BA:16#00	V35	DINT

输出：

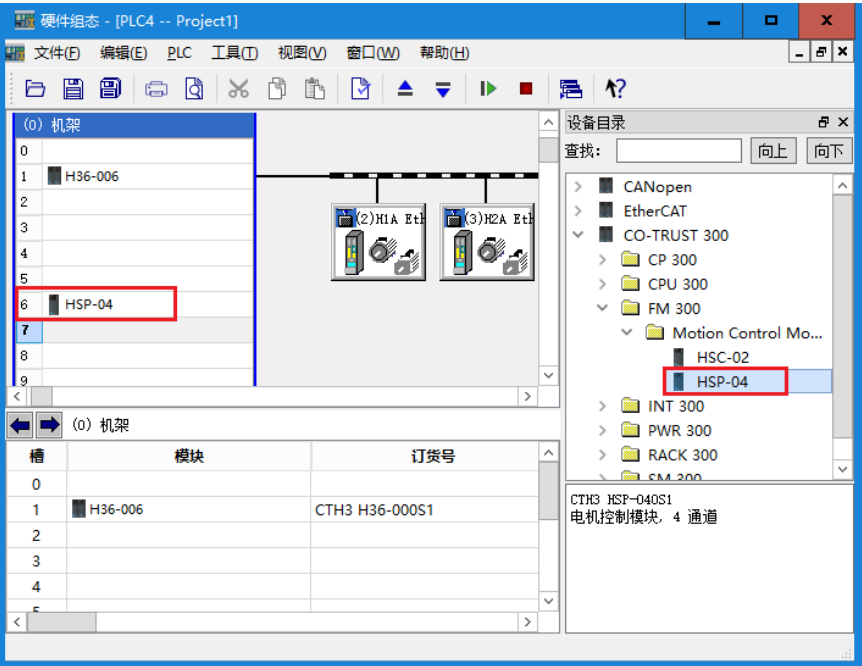
	周期对象	对象值	地址	类型
1	ControlWord (out.wControlWo...	16#6040:16#00	V0	UINT
2	set position (diSetPosition)	16#607A:16#00	V2	DINT
3	set velocity (diSetVelocity)	16#60FF:16#00	V12	DINT
4	set torque (wSetTorque)	16#6071:16#00	V6	INT
5	Modes of operation (OP)	16#6060:16#00	V18	SINT
6	Touch Probe Function	16#60B8:16#00	V16	UINT
7	Add velocity value	16#60B1:16#00	"	
8	Add torque value	16#60B2:16#00	"	

此配置仅通信轴和 EtherCAT 轴支持，在此页面勾选“自动映射”则输入输出均不可修改，建议勾选，因带和不带 CAI402 轴的伺服索引地址不同。

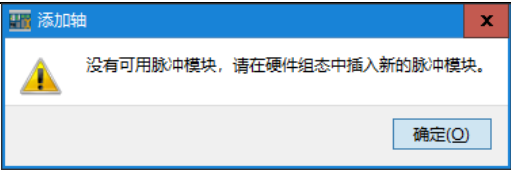
7.4.4.1 轴配置

1、脉冲轴配置

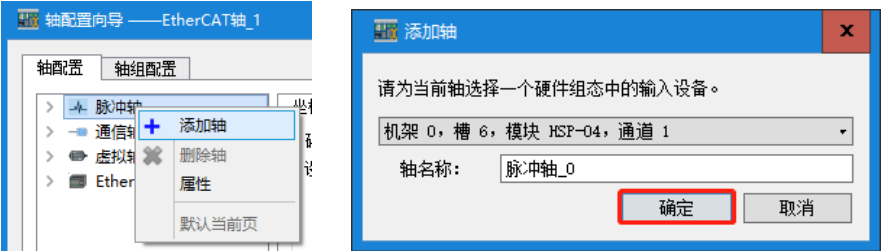
添加脉冲轴前，需先在主站所在机架中添加运动控制模块“HSP-04”，如下图所示：



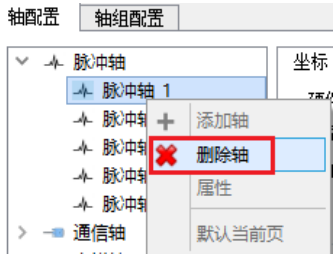
若未添加“HSP-04”，将出现以下提示：



完成以上配置后在“工具”选项中选择“轴向导配置”，在轴向导配置界面鼠标右键点击脉冲轴即可选择添加轴，轴名称可自由更改，点击“确定”即成功添加轴，如下所示：



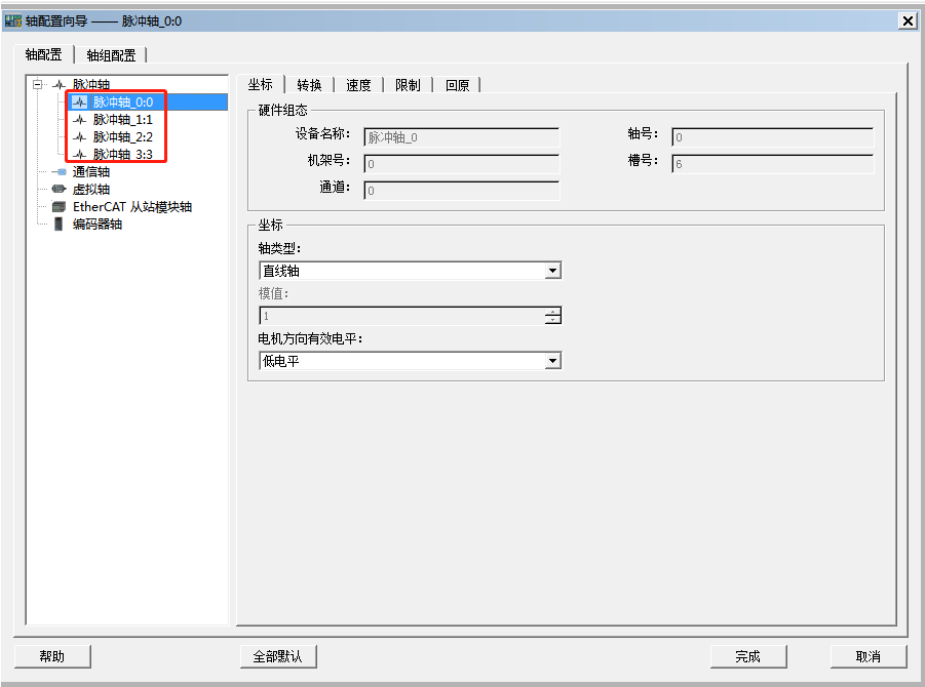
右键选中添加的轴，可选择删除轴，如下图：



提示

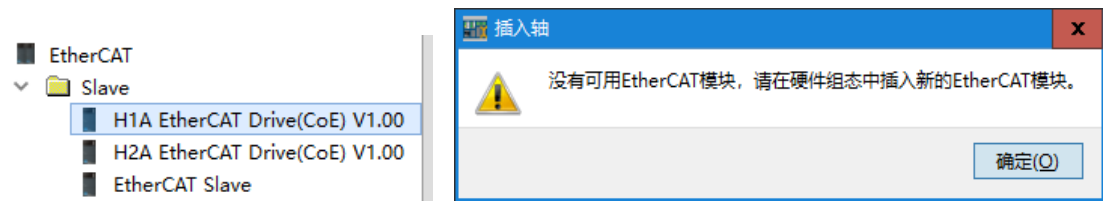
一个 HSP 模块对应可添加 4 个脉冲轴，若需添加更多脉冲轴需在机架内相应添加 HSP 模块。

添加的脉冲轴界面如下所示：

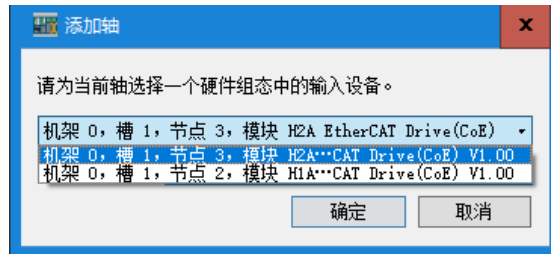


2、通信轴配置

添加通信轴前需在硬件组态界面添加相应的 EtherCAT 从站（如下左图），否则将无法添加通信轴，将出现如下右图所示提示。

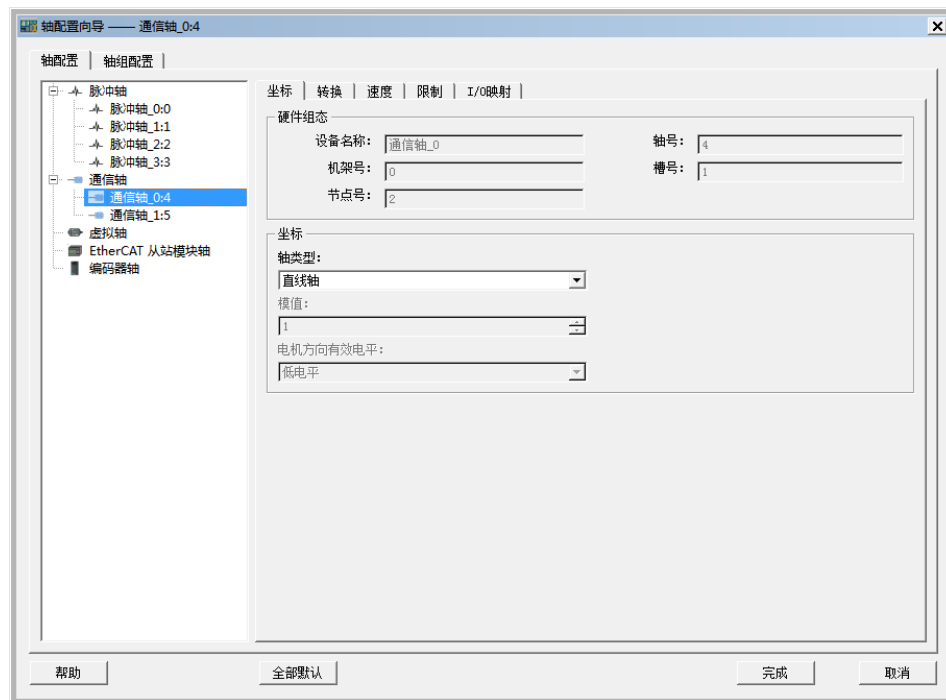


鼠标右键点击“通信轴”，在弹出的“添加轴”界面中可为该轴选择输入设备，可选设备为硬件组态中所添加的从站。“轴名称”可修改，点击“确定”完成添加。



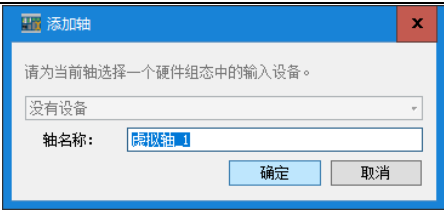
注意：一个从站只可对应添加一个通信轴。

添加的通信轴界面如下所示：

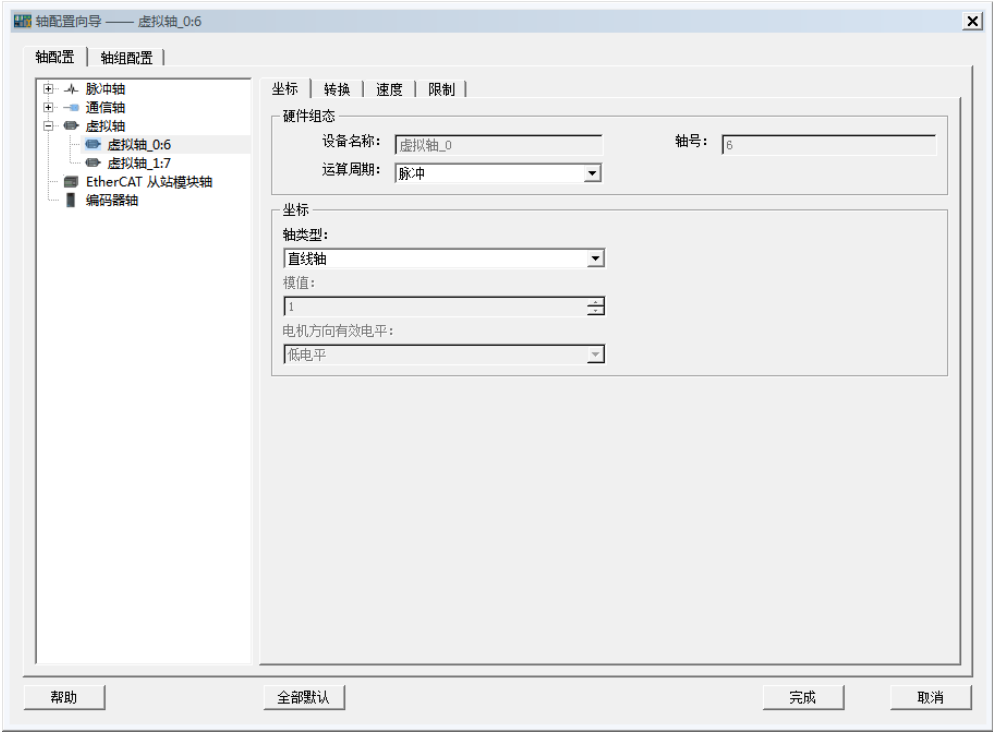


3、虚拟轴配置

在轴配置向导界面，鼠标右键点击“虚拟轴”，选择添加轴，在弹出的以下界面可选择修改轴名称，点击“确定”完成虚拟轴添加。

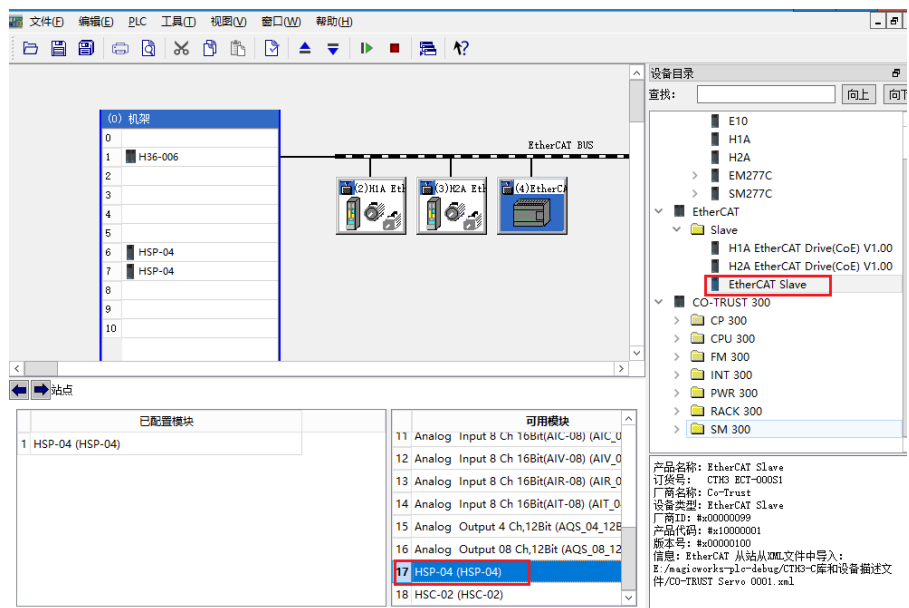


添加的虚拟轴界面如下所示：

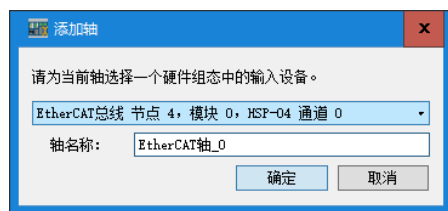


4、EtherCAT 从站模块轴配置

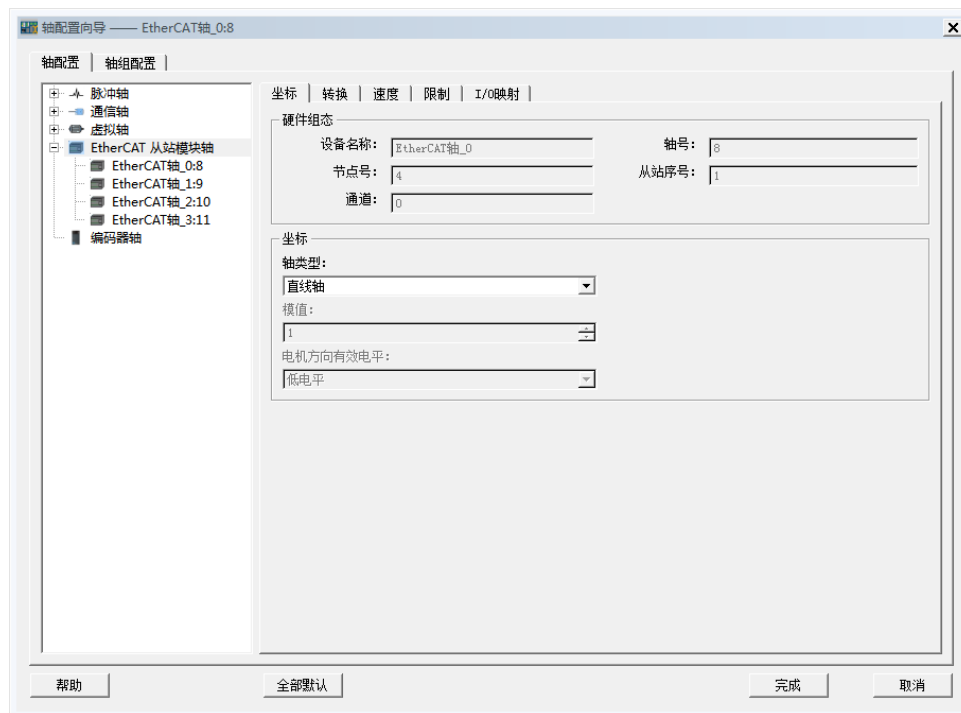
添加 EtherCAT 从站模块轴前，需先在硬件组态界面添加 EtherCAT 从站，并为所添加的 EtherCAT 从站选择配置 HSP 模块。如下图所示：



在“轴配置向导”界面右键点击“EtherCAT 轴”选择“添加轴”，在弹出的以下界面可进行轴配置。配置完点击“确定”完成添加。

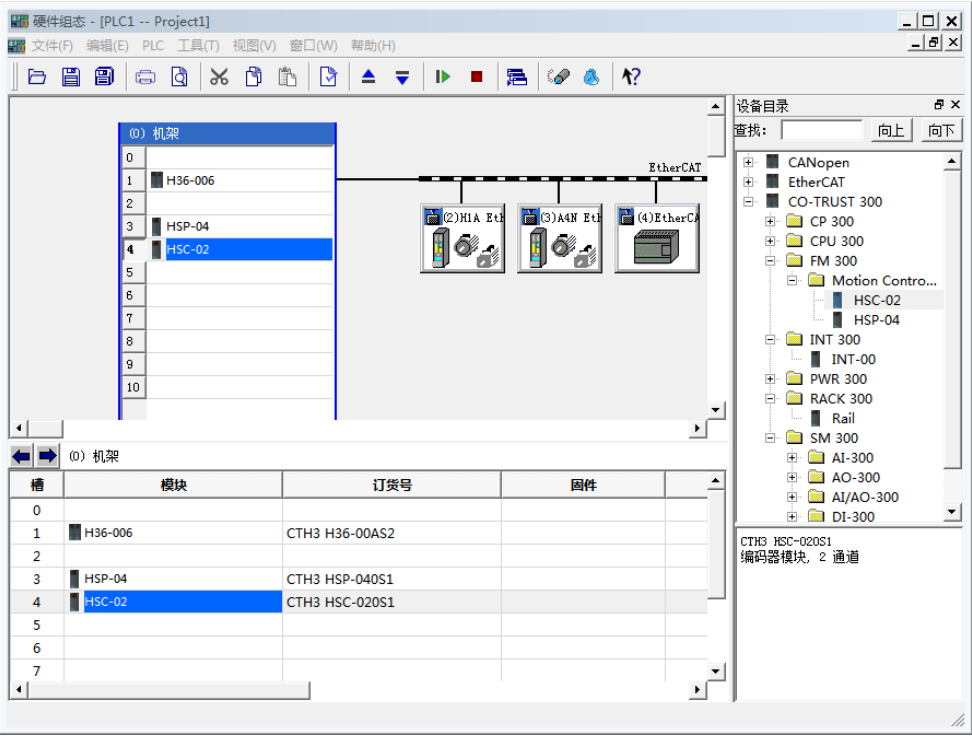


添加的 EtherCAT 从站模块轴界面如下所示：

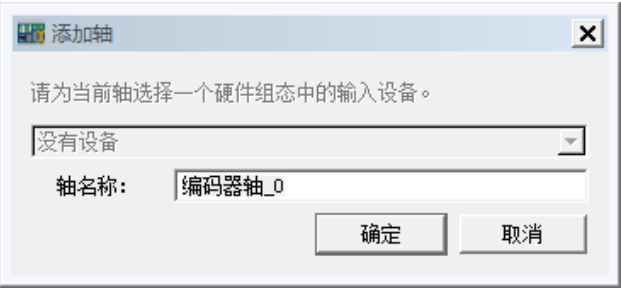


5、编码器轴配置

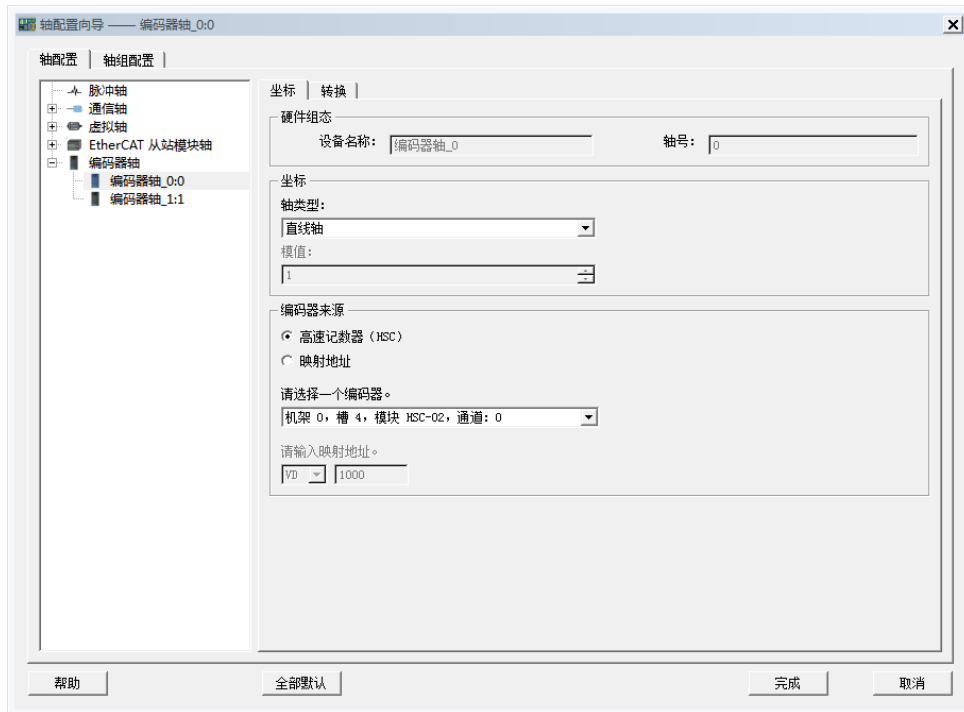
添加脉冲轴前，需先在主站所在机架中添加运动控制模块“HSC-02”，如下图所示：



完成以上配置后在“工具”选项中选择“轴向导配置”，在“轴配置向导”界面右键点击“编码器轴”选择“添加轴”，在弹出的以下界面可进行轴配置。配置完点击“确定”完成添加。



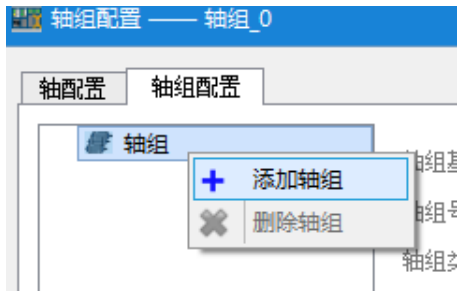
添加的编码器轴界面如下所示：



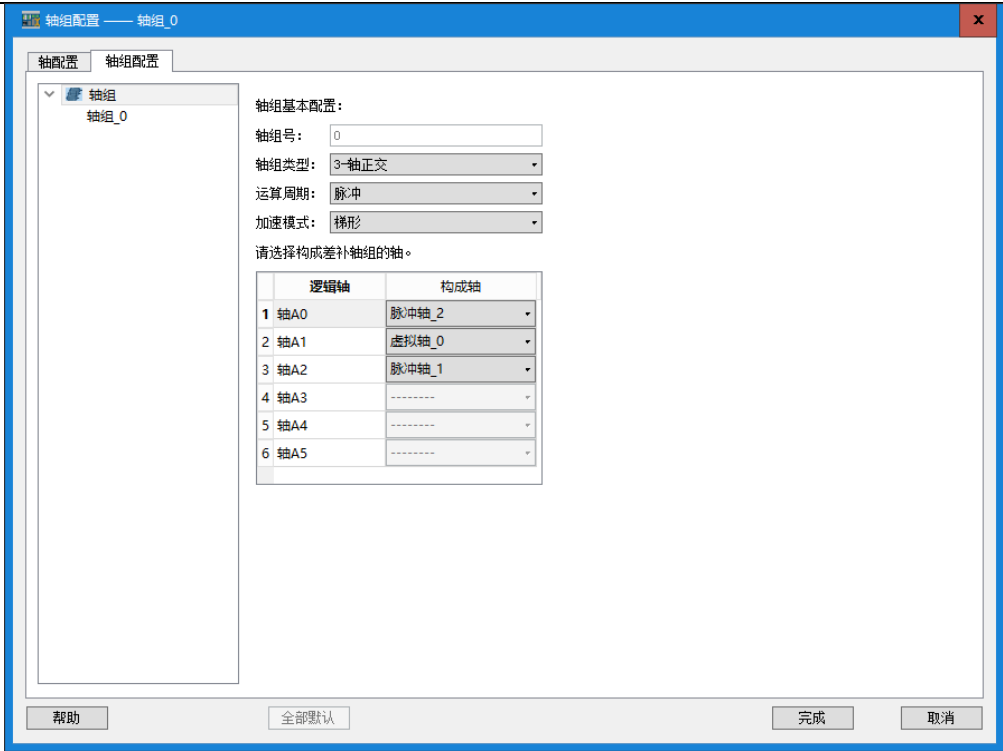
- 编码器来源：为该轴选择对应编码器，可选择输入相应映射地址（默认 V 区）确定编码器。如编码器来源选择高速计数器，需要事先在硬件组态的机架中添加 HSC 模块。选择映射地址时，需要将编码器的位置传送到映射地址中。

7.4.4.2 轴组配置

轴组配置即将配置好的轴两个或三个一组以轴组的形式自由组合，在“轴组配置”界面，右键点击“添加轴组”进行轴组添加，如下图所示：



添加成功的轴组界面如下所示：



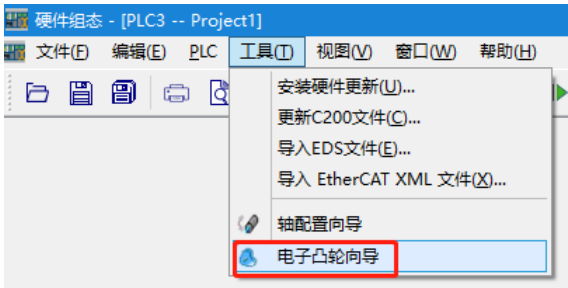
- “轴组基本配置”：“轴组号”不可更改，“轴组类型”下拉选项可选择“2-轴正交”或“3-轴正交”，分别对应可选两组或三组“构成轴”；“运算周期”可选“脉冲”或“EtherCAT”；“加速模式”可选“梯形”，“ $\sin^2$ ”和“二次”。
- “逻辑轴”：默认六组，不可更改。
- “构成轴”：“2-轴正交”对应两个构成轴，“3-轴正交”对应三个构成轴，下拉选项可选择“脉冲轴”或“虚拟轴”。

7.4.5 电子凸轮向导

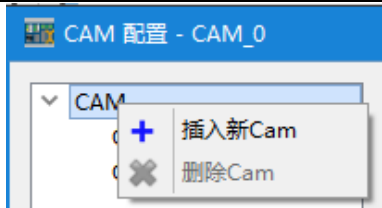
MagicWorksPLC V2.17 及以上版本支持电子凸轮配置。

在电子凸轮编辑器中可以通过图形或列表的方式实现电子凸轮盘(或者凸轮开关功能)。当根据相关的应用程序产生代码时，将创建各种全局数据结构(CAM 数据)，这些数据结构能够被 IEC 程序访问。

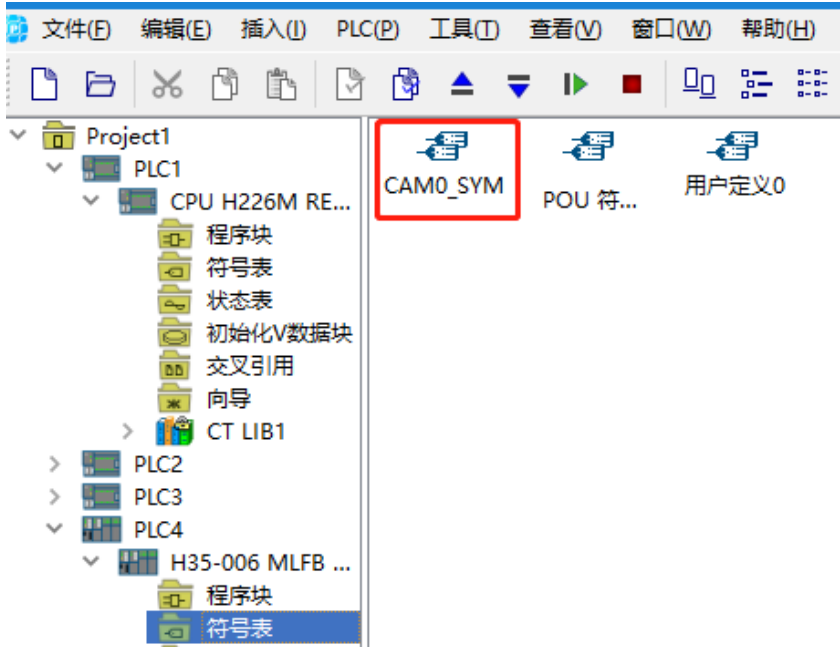
在硬件组态界面选择“工具”->“电子凸轮向导”即可进行电子凸轮配置。



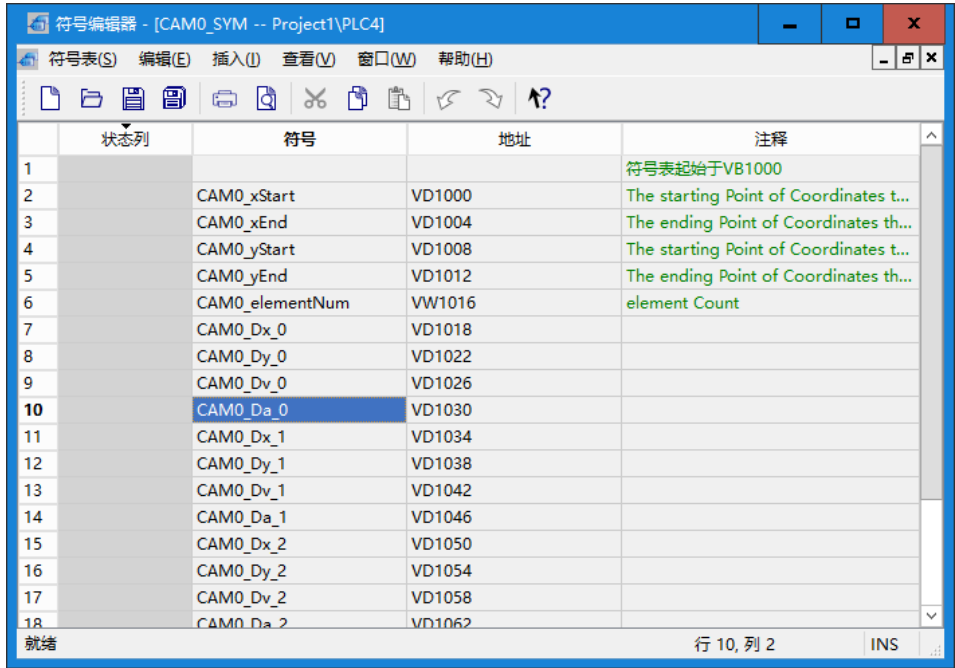
电子凸轮向导配置界面如下，选中“CAM”右键可插入新的 CAM：



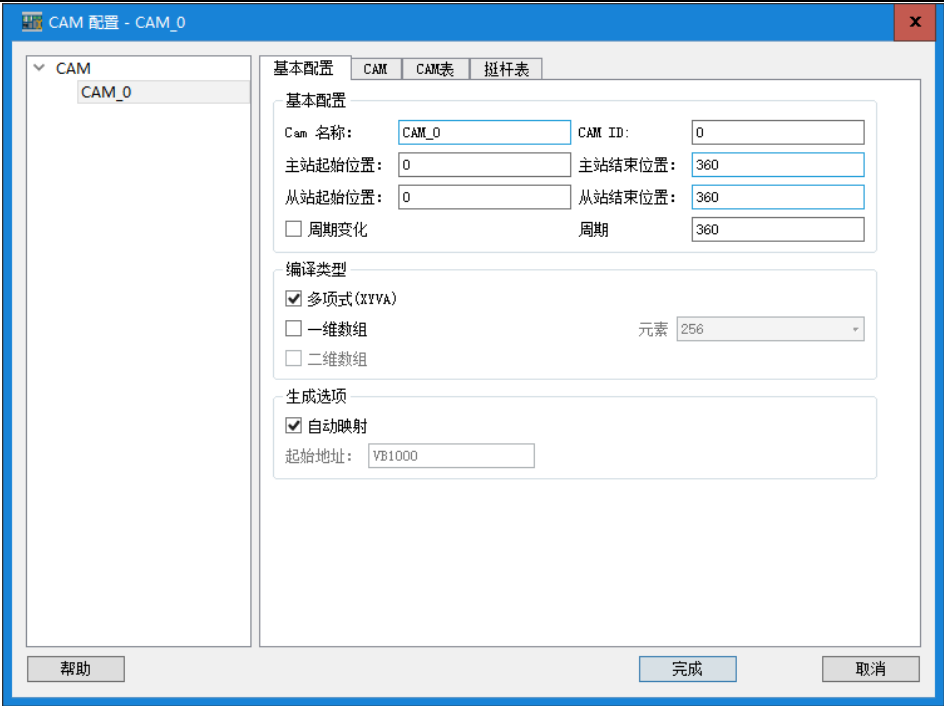
成功插入 CAM 后，在项目管理器界面选择“符号表”，可看到新增的 CAM。



点击“CAM0_SYM”出现以下界面，可查看对应 CAM 的符号地址。

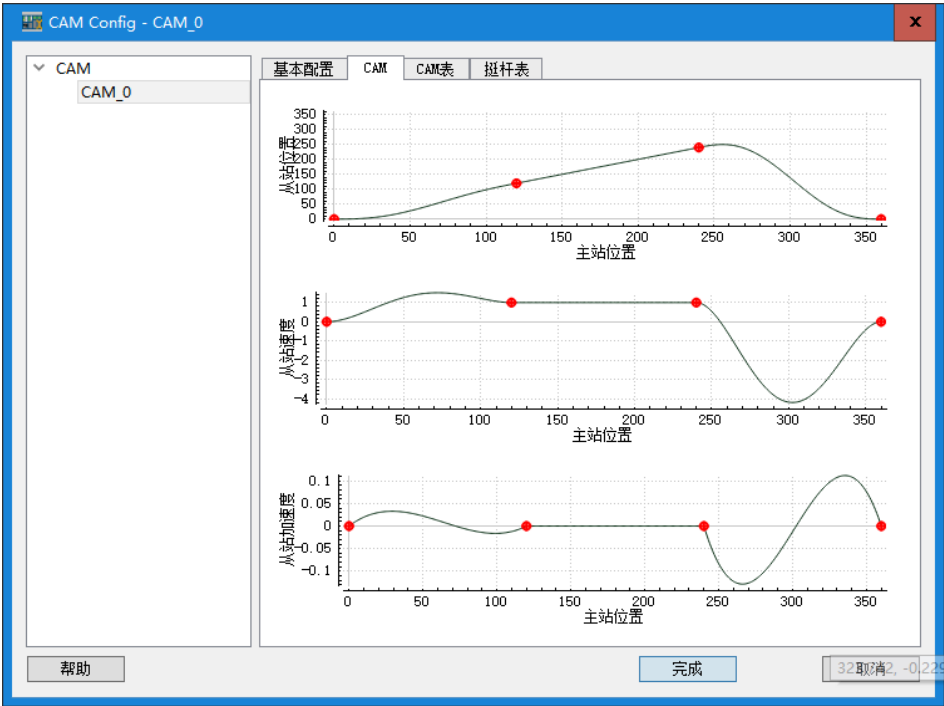


插入 CAM 后，配置界面如下所示。



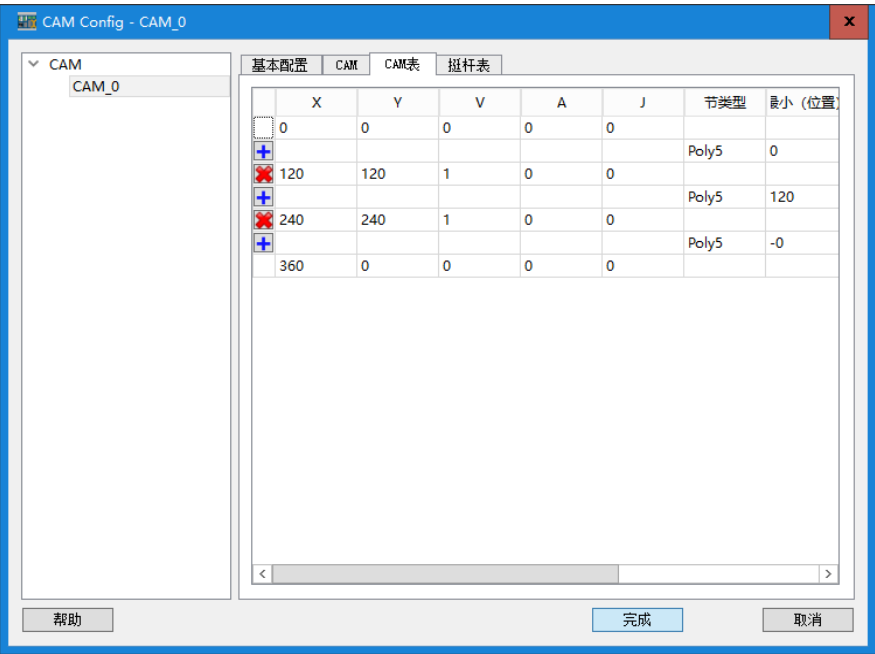
【基本配置】

- “基本配置”：用户可手动更改“Cam 名称”以及主从站的起始和结束位置（主站位置对应 CAM 曲线图中的横坐标，从站位置对应 CAM 曲线图中的纵坐标），勾选“周期变化”，CAM 中曲线图将呈周期变化，主站结束位置即为周期。
- “编译类型”：勾选“多项式(XYVA)”，CAM 在状态表中将以 x,y,a,v 的符号显示横纵坐标、速度与加速度状态。
- “生成选项”：勾选“自动映射”，将默认起始地址为 VB1000，取消勾选，可对起始地址进行修改，起始地址对应符号表中的起始地址。



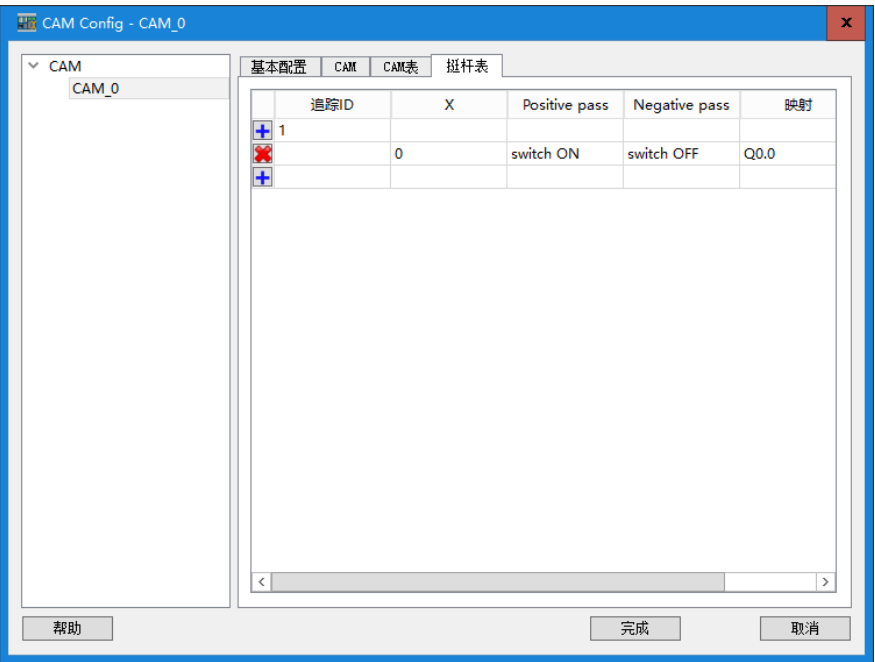
【CAM】

此页面以曲线图的形式显示“主站位置”与“从站位置”，“主站位置”与“从站速率”，“主站位置”与“从站加速度”的关系。横坐标“主站位置”最大值对应“基本配置”界面的“主站结束位置”设置值，纵坐标“从站位置”最大值对应“基本配置”界面“从站结束位置”的设置值。点击红色节点，可将其拖动。



【CAM 表】

此页面对应显示 CAM 页面曲线图中红色节点的信息，一个红色节点对应一组数据，点击“+”可添加新节点，点击“X”可删除已添加的节点。双击“X、Y、V、A”对应的数值，可进行手动修改。



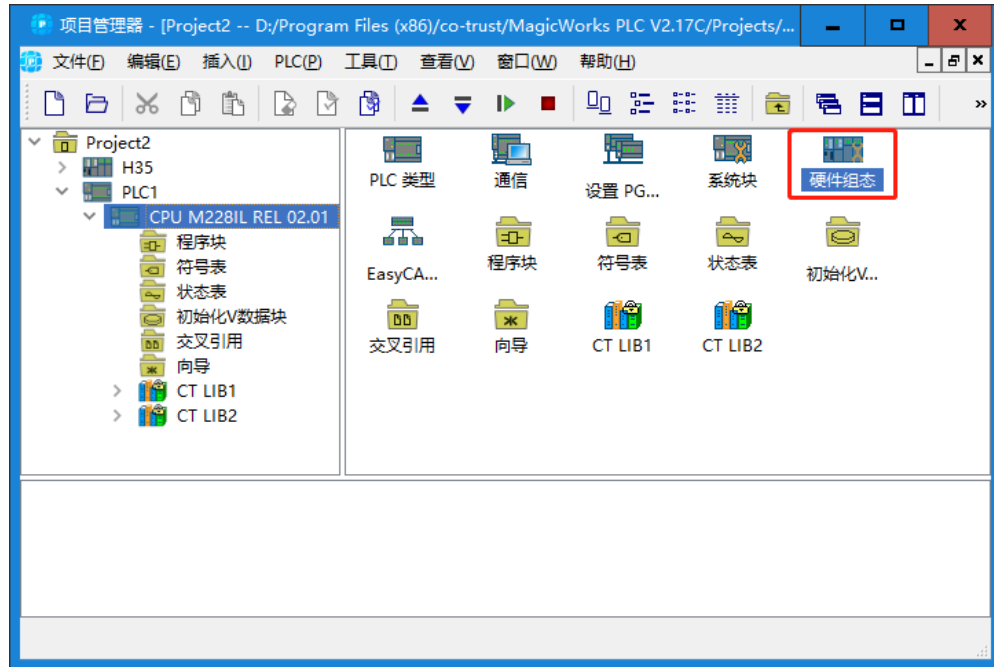
【挺杆表】

功能保留，PLC 暂不支持此功能。

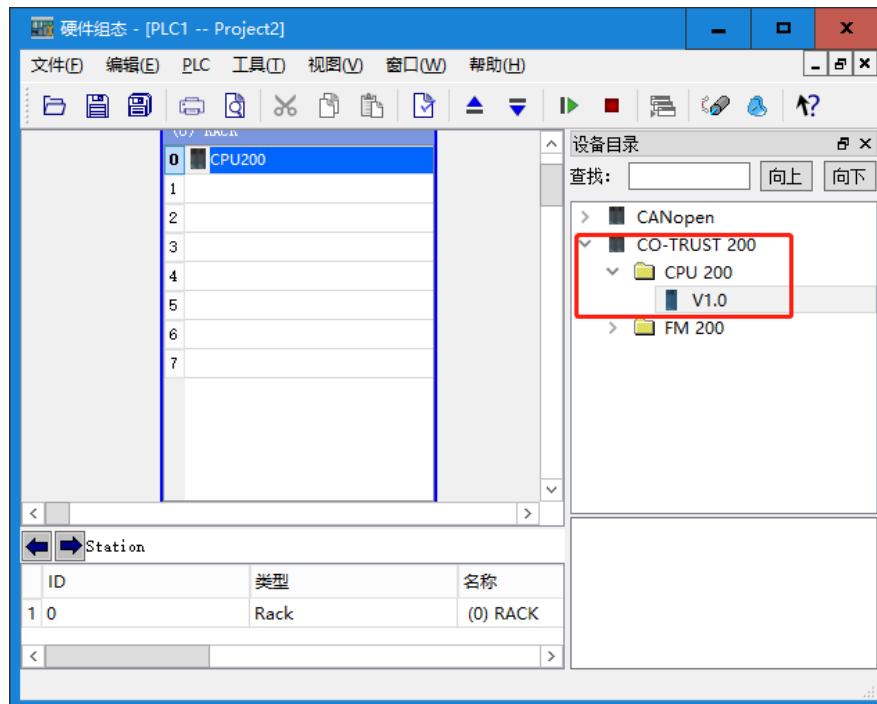
7.4.6 200 系列 PLC 硬件组态

MagicWorks PLC V2.18 及以上版本支持 200 系列 PLC（CTMC 运控系列及 V5 全系列）的硬件组态功能，主要为 CANopen 通信组态，具体操作步骤如下：

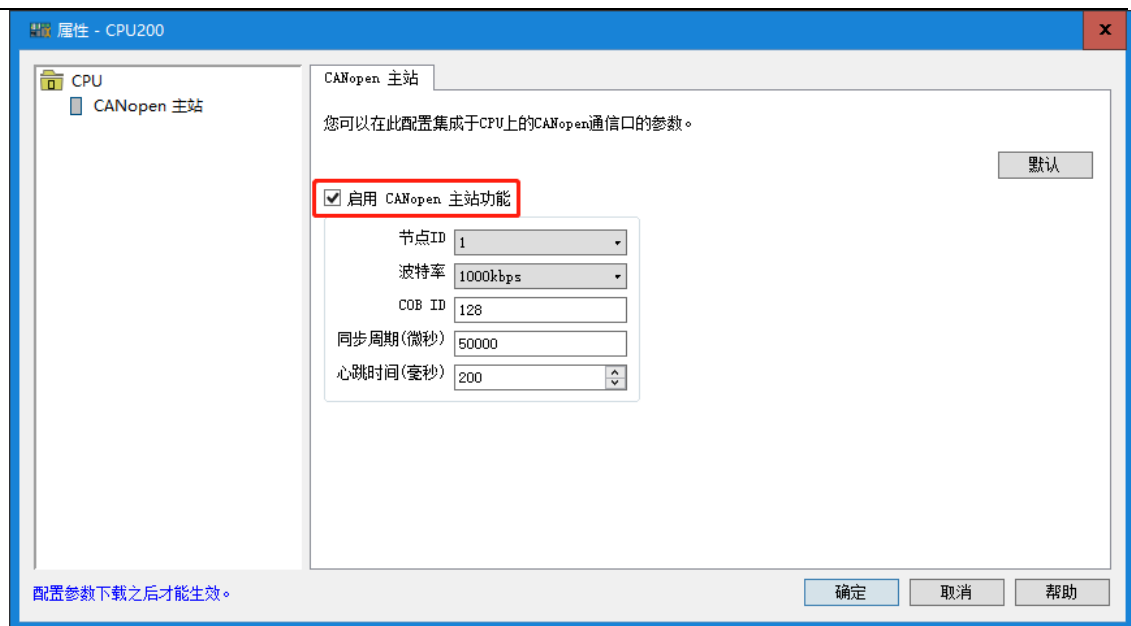
1、在 200 系列 CPU 的项目管理器界面双击“硬件组态”，打开硬件组态界面。



2、在硬件组态界面，依次展开“CO-TRUST 200”→“CPU 200”，将 CPU 拖拽至机架的 0 号槽内，即完成 CPU 主站的添加。



3、双击机架上的“CPU200”，在弹出的窗口中勾选“启用 CANopen 主站功能”即可添加 CAN 总线。



4、完成以上操作后可在 CAN 总线上添加 CANopen 从站并进行配置,详细步骤可参考章节 [7.4.2 组态 CANopen 通信](#)。

200 系列 CPU 硬件组态同样支持轴配置和电子凸轮,操作步骤与 300 系列类似,详细步骤请参考章节 [7.4.4 轴配置向导](#)和章节 [7.4.5 电子凸轮向导](#)。

7.4.7 其它操作

1、编译硬件组态

硬件组态完成后,您可以编译当前工程,生成可下载的数据。有两种方法实现编译:

- 保存组态,在硬件组态软件界面单击“文件”→“编译”菜单,或点击工具栏上的编译按钮,系统会自动保存并编译组态数据,并且弹出对话框显示编译信息,在此您可以了解到编译中的错误和报警的详细信息。
- 保存组态后,回到项目管理器主界面,选中硬件组态图标,然后点击编译按钮来编译保存的硬件组态,也可以点击全部编译依次编译程序块,数据块,硬件组态。编译信息会显示在下方的输出窗口中。

2、下载硬件组态

连接 PLC 的情况下,在项目管理器界面点击下载按钮,打开下载对话框,然后在选项中勾选硬件组态项,点击“下载”按钮,硬件组态数据就会被下载至目标 PLC 中。

3、将 PLC 中的硬件组态恢复至工程

连接 PLC 的情况下,在项目管理器界面点击上载按钮,打开上载对话框,在选项中勾选硬件组态项,点击“上载”按钮,保存在 PLC 中的硬件组态数据就会被上载至工程中。上载完成后,双击硬件组态图标,即可查看上载的组态。

4、清除 PLC 中保存的硬件组态

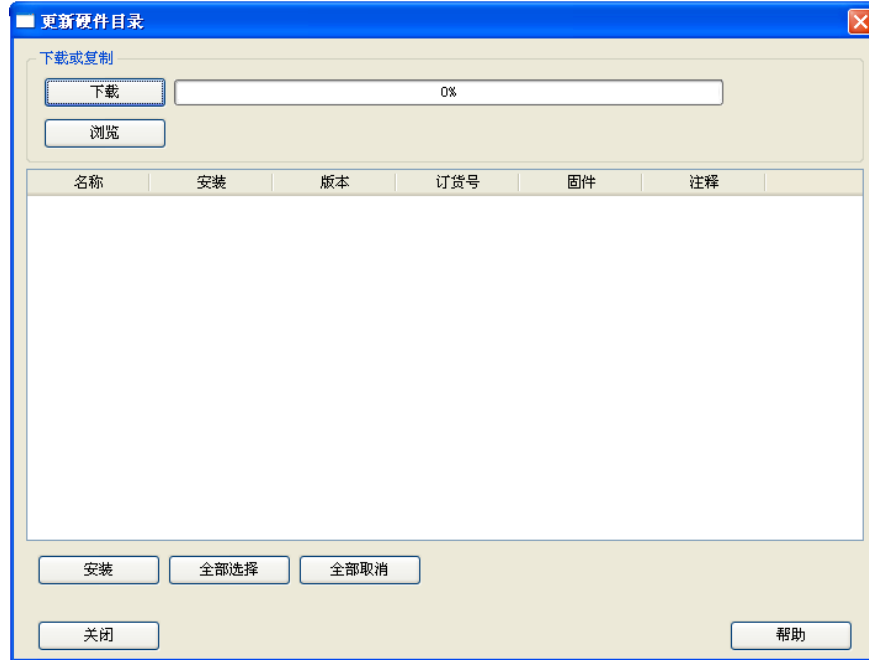
连接 PLC 的情况下,在项目管理器界面点击菜单项“PLC”→“清除”,在清除对话框选项中勾选硬件组态项,然后点击“清除”按钮,保存在 PLC 中的硬件组态数据就会被删除。

5、硬件比较

连接 PLC 的情况下，在项目管理器界面点击菜单项“PLC”→“比较”，即可打开比较对话框，在选项中勾选硬件组态项，然后点击“开始比较”，系统就会比较当前工程中的与保存在 PLC 中的硬件组态数据，并将比较的结果显示在下拉框中。

6、安装硬件更新（此功能暂时保留不启用）

下载安装硬件更新：选择菜单项“工具”→“安装硬件更新（U）”对话框：

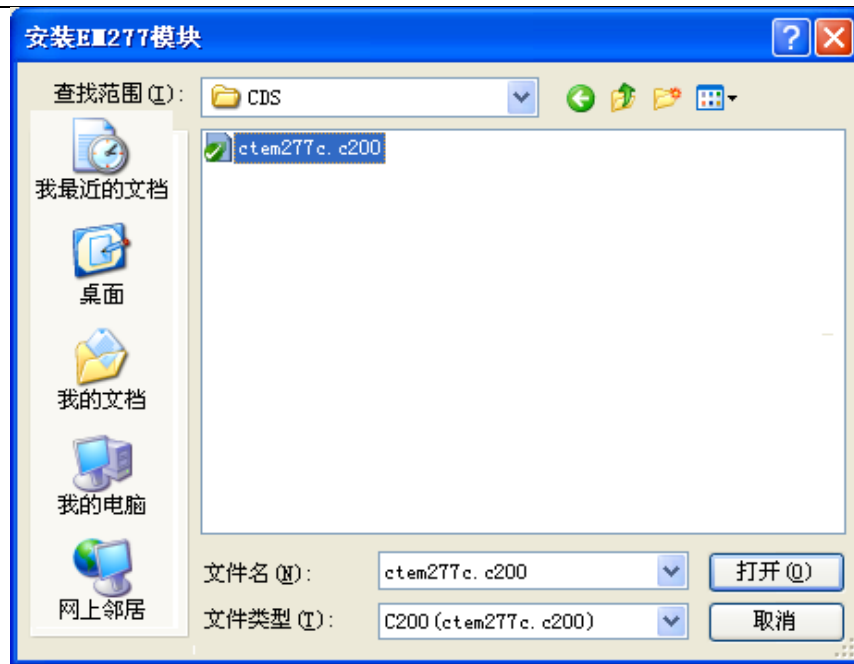


“下载”：从互联网上下载硬件更新文件（名称固定为 hspcontents_ct.xml）。

“浏览”：从硬盘上拷贝硬件更新文件（名称固定为 hspcontents_ct.xml）。

7、更新 C200 文件

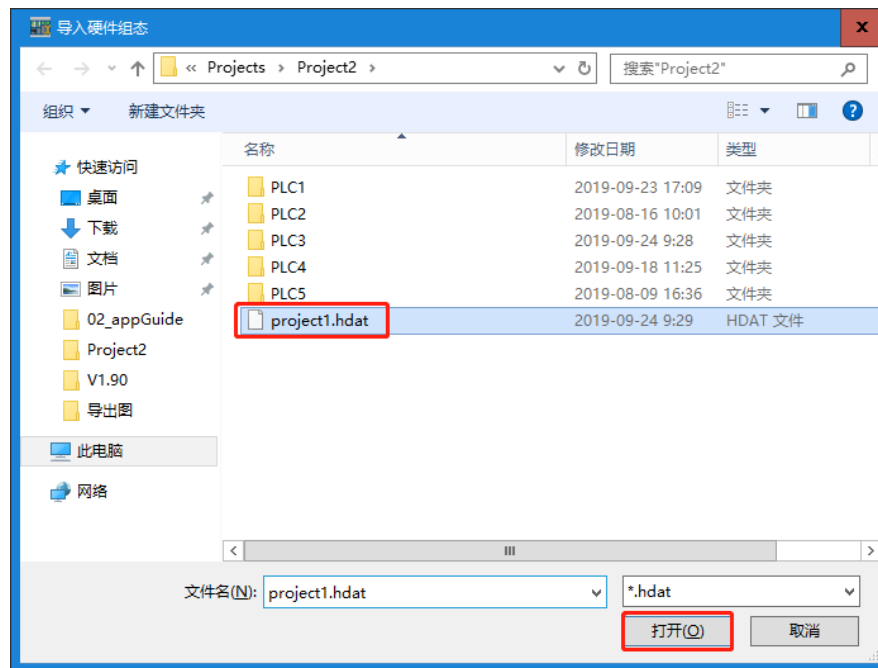
若在当前 EM227C 节点下找不到需要组态的数字量或模拟量模块，则可以通过“更新 C200 文件”来解决，在“硬件组态”界面点击“工具”选择“更新 C200 文件”。在弹出的“安装 EM277 模块”界面找到需要的文件，点击“打开”，即可导入需要的 C200 文件，如下图所示：



8、导入导出硬件组态文件

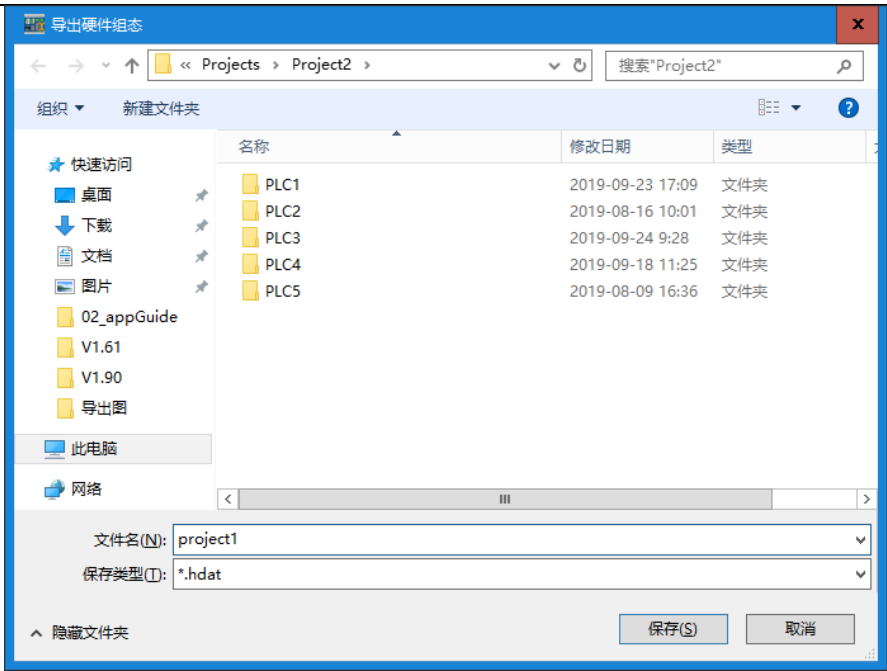
◆ 导入硬件组态

将 ASCII 文件格式的硬件组态文件（后缀为.hdat）内容导入到当前激活的硬件组态子窗口。



◆ 导出硬件组态

将当前硬件组态窗口的内容以 ASCII 文件格式导出到后缀为.hdat 的二进制文件。



7.5 安全保护

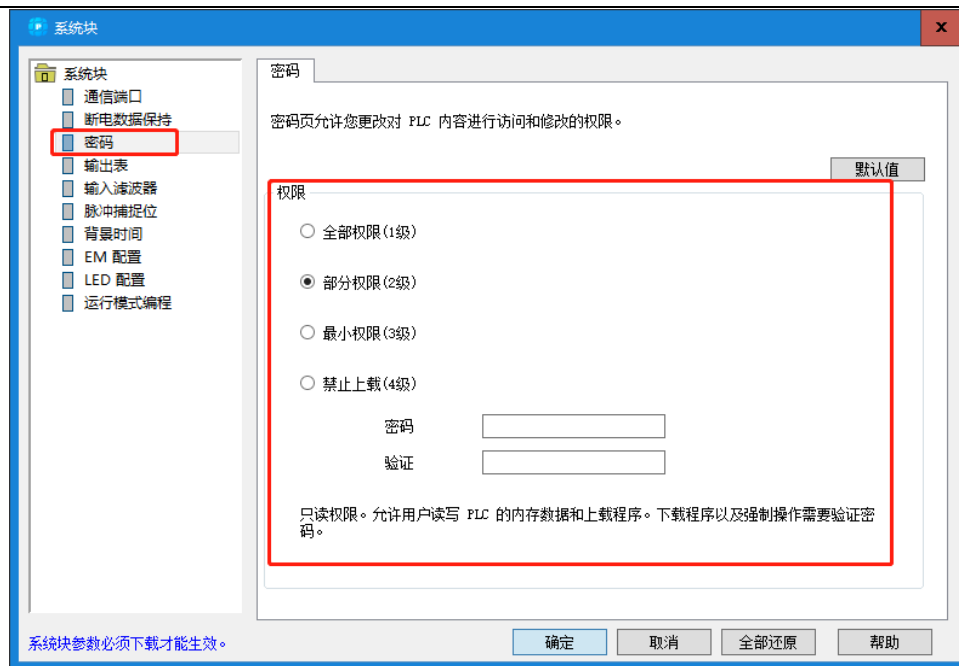
MagicWorks PLC 的以下项目支持加密保护功能：

- CPU 加密
- 项目文件加密
- 工程程序块加密
- 工程数据块加密
- 保密库功能

CPU 加密

小型 PLC（CTSC-100/200 PLC）通过系统块设置对 CPU 进行加密：在项目管理器主界面的项目树展开 CPU 节点，然后在右侧工作区域双击“系统块”，即弹出如下配置窗口，在该窗口中选择“密码”，即可对 CPU 进行加密操作。

中型 PLC（CTH3 系列 PLC）通过硬件组态界面的 CPU 节点属性进行加密：在中型 PLC 项目的硬件组态界面，双击导轨中 CPU 所在的插槽（Slot 1），弹出如下对话框，在该对话框中选择“密码”即可对 CPU 进行加密操作。

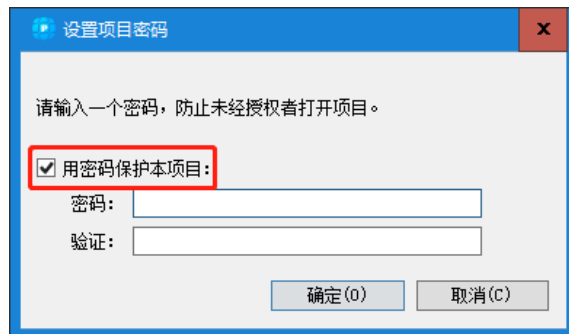


<备注> 详细描述请参考章节 [5.6.3 密码保护](#)。

项目文件加密

使用此命令，用户可以为项目设置一个密码保护，这样其他用户未经密码验证无法查看和编辑项目。

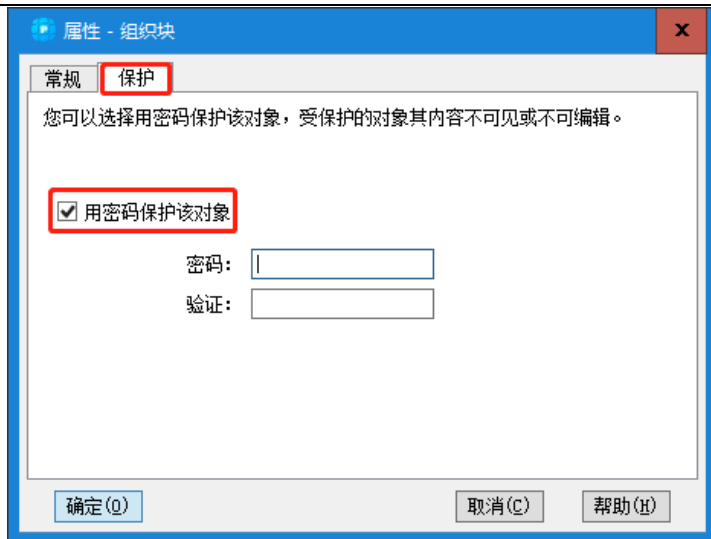
选择菜单项“文件”→“设置密码”打开如下对话框，勾选“用密码保护本项目”后即可设置项目密码，项目密码的长度最长为 8 位。



工程程序块（OB/FC/FB）加密

程序块属性的“保护”标签允许您选择用密码保护程序块，以防其他用户看到该 POU，并在下载时加密。为工程的程序块（OB/FC/FB）设置密码的操作如下：

在项目树选中程序块节点，然后在右侧的工作区域中右键点击你需要加密的程序块（OB/FC/FB）并选择“对象属性”，即会弹出如下程序块属性对话框，在属性对话框中选中“保护”，最后勾选“用密码保护该 POU”并输入密码，则该程序块加密成功。

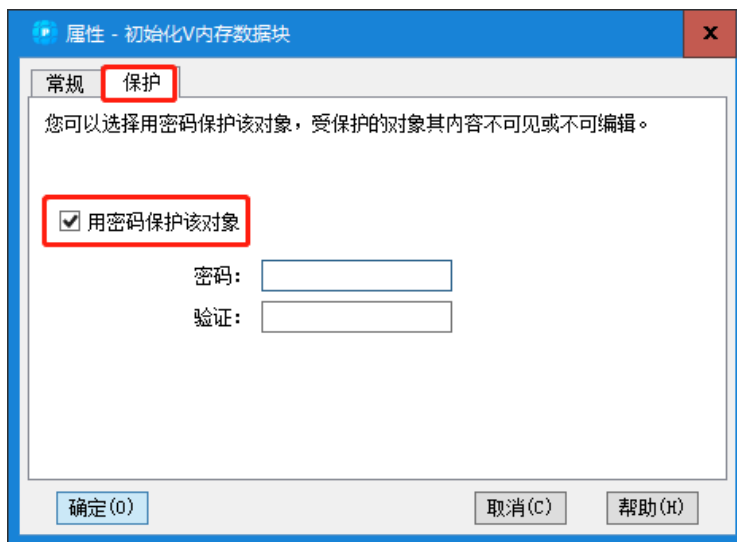


欲取消对程序块的锁定：在程序块的属性对话框选择“保护”标签->输入密码->点击“验证”。

工程数据块加密

数据块属性的“保护”标签允许您选择用密码保护数据块，以防其他用户看到该 POU，并在下载时加密。为初始化 V 内存数据块设置密码的操作如下：

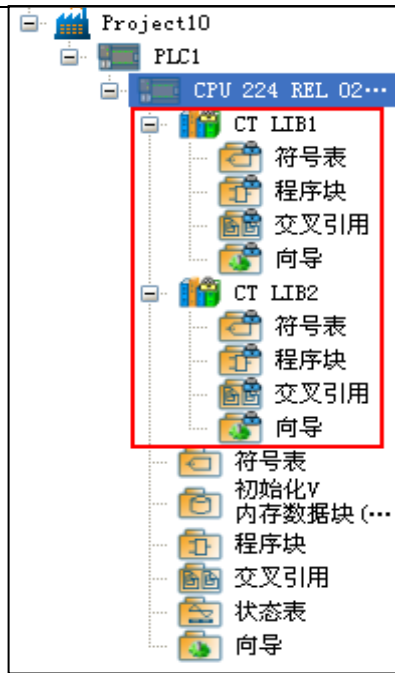
在项目树选中初始化 V 内存数据块节点，然后在右侧的工作区域中右键点击你需要加密的数据块并选择“对象属性”，即会弹出如下程序块属性对话框，在属性对话框中选中“保护”，最后勾选“用密码保护该 POU”并输入密码，则该数据块加密成功。



欲取消对数据块的锁定：在数据块的属性对话框选择“保护”标签->输入密码->点击“验证”。

保密库功能

与 CPU200 保持一致，保密库的入口与主程序在同一个 PLC 节点下，如下图所示，CT LIB1 和 CT LIB2 即为保密库，展开 CT LIB1 或 CT LIB2 即可配置相应的子项目。



<备注> 保密库不能够执行上载和程序监控功能。

8 PLC 存储区

本章概要

- 内存类型
- CPU 内存地址范围
- 寻址内存
- 常数数值范围
- 关键字

8.1 内存类型

区域	说明	作为位存取	作为字节存取	作为字存取	作为双字存取	可保留	可强迫
I	离散输入和映像寄存器	读取/写入	读取/写入	读取/写入	读取/写入	否	是
Q	离散输出和映像寄存器	读取/写入	读取/写入	读取/写入	读取/写入	否	是
M	内部内存位	读取/写入	读取/写入	读取/写入	读取/写入	是	是
SM	特殊内存位	读取/写入	读取/写入	读取/写入	读取/写入	否	否
V	变量内存	读取/写入	读取/写入	读取/写入	读取/写入	是	是
T	定时器当前值和定时器位	T 位 读取/写入	否	T 当前 读取/写入	否	T 当前 -是否	T 位-否
C	计数器当前值和计数器位	C 位 读取/写入	否	C 当前 读取/写入	否	C 当前 -是否	C 位-否
HC	高速计数器当前值	否	否	否	只读	否	否
AI	模拟输入	否	否	只读	否	否	是
AQ	模拟输出	否	否	只写	否	否	是
AC	累加器寄存器	否	读取/写入	读取/写入	读取/写入	否	否
L	局部变量内存	读取/写入	读取/写入	读取/写入	读取/写入	否	否
S	SCR	读取/写入	读取/写入	读取/写入	读取/写入	否	否
DI/D B	数据块	读取/写入	读取/写入	读取/写入	读取/写入	是	否

8.2 CPU 内存地址范围

当您执行任何需要与 CPU 通讯的操作时，MagicWorks PLC 首先会识别 CPU 型号。创建程序时，必须确保输入的 I/O 和内存范围对目标 CPU 类型有效。

如果下载的程序中存取的 I/O 或内存位置超出 CPU 的允许范围，系统会提示错误信息。下表列举了部分 CPU 内存地址范围，若需了解更多 CPU 内存范围，请至合信官网查看，网址：

<http://www.co-trust.com/Products/Programmable/CTH200/CPU/>

数据类型	内存类型	CPU224	CPU226	CTH3 H35-000S1 / CTH3 I39-000S1	CTH3 H36-000S1	CTH3 H35-00AS1
位	I	0.0 - 15.7	0.0 - 15.7	0.0 - 255.7	0.0 - 255.7	0.0 - 255.7
	Q	0.0 - 15.7	0.0 - 15.7	0.0 - 255.7	0.0 - 255.7	0.0 - 255.7
	V	0.0 - 8191.7	0.0 - 10239.7	0.0 - 10239.7	0.0 - 10239.7	0.0 - 10239.7
	M	0.0 - 31.7	0.0 - 31.7	0.0 - 31.7	0.0 - 31.7	0.0 - 31.7
	SM	0.0 - 549.7	0.0 - 549.7	0.0 - 549.7	0.0 - 549.7	0.0 - 549.7
	S	0.0 - 31.7	0.0 - 31.7	0.0 - 31.7	0.0 - 31.7	0.0 - 31.7
	L	0.0 - 59.7	0.0 - 59.7	0.0 - 127.7	0.0 - 127.7	0.0 - 127.7
	T	0 - 255	0 - 255	0 - 1024	0 - 1024	0 - 1024
	C	0 - 255	0 - 255	0 - 1024	0 - 1024	0 - 1024
	AC	0 - 3	0 - 3	0 - 3	0 - 3	0 - 3
	DB	不支持	不支持	DB1.DBX0.0 - DB254.DBX6553 5.7(总大小最多 118KBytes)	DB1.DBX0.0 - DB254.DBX655 35.7(总大小最多 64KBytes)	DB1.DBX0.0 - DB254.DBX65 535.7(总大小最 多 256KBytes)
字节	IB	0 - 15	0 - 15	0 - 255	0 - 255	0 - 255
	QB	0 - 15	0 - 15	0 - 255	0 - 255	0 - 255
	VB	0 - 8191	0 - 10239	0 - 10239	0 - 10239	0 - 10239
	MB	0 - 31	0 - 31	0 - 31	0 - 31	0 - 31
	SMB	0 - 549	0 - 549	0 - 549	0 - 549	0 - 549
	SB	0 - 31	0 - 31	0 - 31	0 - 31	0 - 31
	LB	0 - 59	0 - 59	0 - 127	0 - 127	0 - 127
	DB	不支持	不支持	DB1.DBB0 - DB254.DBB6553 5(总大小最多 118KBytes)	DB1.DBB0 - DB254.DBB655 35(总大小最多 64KBytes)	DB1.DBB0 - DB254.DBB65 535(总大小最多 256KBytes)
字	IW	0 - 14	0 - 14	0 - 254	0 - 254	0 - 254
	QW	0 - 14	0 - 14	0 - 254	0 - 254	0 - 254
	VW	0 - 8190	0 - 10238	0 - 10238	0 - 10238	0 - 10238
	MW	0 - 30	0 - 30	0 - 30	0 - 30	0 - 30
	SMW	0 - 548	0 - 548	0 - 548	0 - 548	0 - 548
	SW	0 - 30	0 - 30	0 - 30	0 - 30	0 - 30

双字	LW	0 - 58	0 - 58	0 - 126	0 - 126	0 - 126
	AIW	0 - 62	0 - 62	0 - 1022	0 - 1022	0 - 1022
	AQW	0 - 62	0 - 62	0 - 1022	0 - 1022	0 - 1022
	T	0 - 255	0 - 255	0 - 1024	0 - 1024	0 - 1024
	C	0 - 255	0 - 255	0 - 1024	0 - 1024	0 - 1024
	DB	不支持	不支持	DB1.DBW0 - DB254.DBW6553(总大小最多118KBytes)	DB1.DBW0 - DB254.DBW6553(总大小最多64KBytes)	DB1.DBW0 - DB254.DBW6553(总大小最多256KBytes)
	ID	0 - 12	0 - 12	0 - 252	0 - 252	0 - 252
	QD	0 - 12	0 - 12	0 - 252	0 - 252	0 - 252
	VD	0 - 8188	0 - 10236	0 - 10236	0 - 10236	0 - 10236
	MD	0 - 28	0 - 28	0 - 28	0 - 28	0 - 28
	SMD	0 - 546	0 - 546	0 - 546	0 - 546	0 - 546
	SD	0 - 28	0 - 28	0 - 28	0 - 28	0 - 28
	LD	0 - 56	0 - 56	0 - 124	0 - 124	0 - 124
	HC	0 - 5	0 - 5	不支持	不支持	不支持
	DB	不支持	不支持	DB1.DBW0 - DB254.DBW6553(总大小最多118KBytes)	DB1.DBW0 - DB254.DBW6553(总大小最多64KBytes)	DB1.DBW0 - DB254.DBW6553(总大小最多256KBytes)

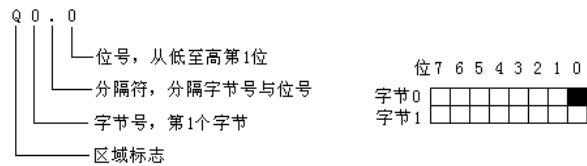
8.3 寻址内存

程序中，指令操作数有直接寻址、符号寻址和间接寻址三种寻址方式。

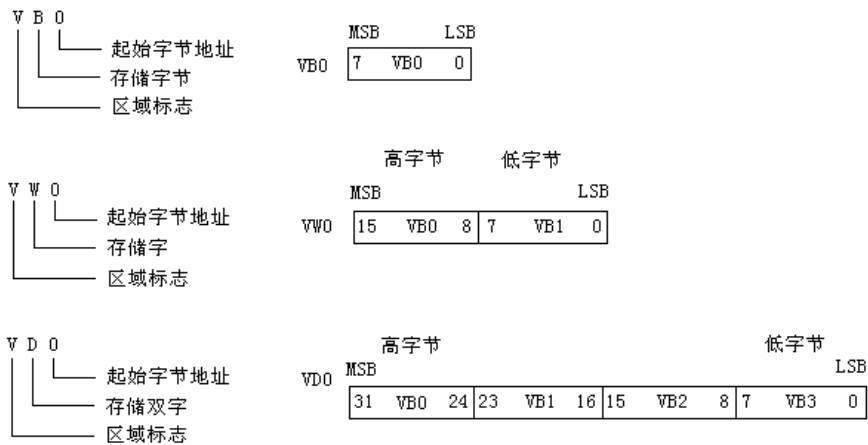
8.3.1 直接寻址

CT CPU 将信息存储于不同的存储器单元，每个单元都有唯一的地址。您可以明确指定希望存储的内存地址，允许程序直接存取信息。直接寻址需要指定内存区域、大小和位置。

欲存取内存区中的某个位，您需要指定内存区标识符、字节地址和前面带一个点号的位数。如 Q0.0 指向 Q 内存区中的第一个字节的最低位。字节号与位号中间的点 (".") 用于分隔位址。



您可以使用字节地址格式将大多数内存区 (V、I、Q、M、S、L 和 SM) 的数据存取为字节、字或双字。欲存取内存中数据的字节、字或双字，您需要指定区域标识符、数据大小指定和字节、字或双字的字节地址，例如，VB0 指向 V 内存区中的第一个字节位置。



其他内存区中的数据（例如，T、C、HC 和累加器）可使用地址格式存取，地址格式包括区域标识符和设备号码，如 T33 指向第 33 号定时器。

指令直接寻址数据区类型及可寻址空间大小

直接寻址数据区类型	可寻址空间大小		
	CPU224	CPU226	H35-000S1
V	8192 字节	10240 字节	10240 字节
DI/DB	不支持	不支持	65536 字节
I	16 字节	16 字节	256 字节
Q	16 字节	16 字节	256 字节
M	32 字节	32 字节	32 字节
S	32 字节	32 字节	32 字节
L	60 字节	60 字节	128 字节
SM	550 字节	550 字节	2048 字节
T	256 位, 256 字	256 位, 256 字	256 位, 256 字
C	256 位, 256 字	256 位, 256 字	256 位, 256 字
HC	6 双字	6 双字	6 双字
AC	4 双字	4 双字	4 双字
AI	32 字	32 字	512 字
AQ	32 字	32 字	512 字

<备注> CPU226M-CAN 的 I、Q 在 16 字节基础上增加了 64 字节，AI、AQ 增加了 128 字，SM 内存增加了 100 字节，扩展的内存给 CAN 功能专用，详情可参考《CANopen 产品手册》。

支持的数据类型

数据类型	描述	位宽
BOOL	布尔	1
BYTE	字节	8
WORD	字	16
INT	整型	16
DWORD	双字	32
DINT	双整型	32
REAL	实数	32

STRING	字符串指针	32
STRINGCHAR (仅共享数据块支持该类型)	字符串	256 字节 (最多可输入 255 字符, 首字节为长度)

8.3.2 符号寻址

为了使程序简单易读, 有时采用符号寻址, 使用地址的符号名 (字母数字和下划线的组合) 来识别地址。符号常数使用符号名识别常数或 ASCII 字符值。

对于 PLC 程序, 您用符号表进行全局符号赋值。如果您在符号表或局部变量表中有指定的符号地址, 您可以通过“查看”→“符号寻址”菜单命令切换显示的方式: 仅显示地址、仅显示符号、显示符号和地址。

符号寻址的三种方式:

- 1) 仅显示地址。如果在全局或局部符号表中为某地址定义了符号, 使用该地址时只显示地址, 不显示符号。
- 2) 仅显示符号。对全局或局部符号表中为定义了符号的某地址, 程序中显示其符号; 未定义符号直接显示地址。
- 3) 符号地址同时显示。例如: Always_On: SM0.0

<提示> 程序中局部变量和全局变量的名称相同时, 局部用法优先。即如果在程序中检测到有符号变量, 程序编辑器首先会去查找当前 POU 对应的局部变量表, 如果局部变量表中不存在该符号的定义, 再去查找全局变量表。例如, 您在符号表中将 PumpOn 定义为全局符号, 同时也在 FC1 中将其定义为局部变量。编译程序时, 在 FC1 中使用将局部定义用于 PumpOn, 在其它 POU 中 PumpOn 使用全局定义。

8.3.3 间接寻址

支持 I、Q、V、M、S、SM、T (仅限当前值) 和 C (仅限当前值) 的间接寻址;

中型 PLC 支持背景数据块内存空间的间接寻址, 支持 DI、DB 存储区的间接寻址。

间接寻址使用指针来访问内存。指针是包含另一个内存地址的双字内存位置。您只能将 V 内存、L 内存或累加器寄存器 (AC1、AC2、AC3) 用作指针。欲建立指针, 您必须使用“移动双字”指令, 将间接寻址内存位置移至指针位置。指针还可以作为参数传递至子程序。

CTSC-200 允许使用指针存取以下内存区: I、Q、V、M、S、T * (仅限当前值) 和 C (仅限当前值)。您不能使用间接寻址存取位地址或存取 AI、AQ、HC、SM 或 L 内存区。

间接寻址中必须使用到两个特殊符号: * 指定该操作数是一个指针; & 取内存位置的地址而非内存中储存的内容。

	VB0	VB1	VB2	VB3
	12	34	56	78
AC1	VB0的地址			MOVD &VB0, AC1 将VB0的地址放入AC1建立指针
AC0	1234			MOVW *AC1, AC0 将AC1里存的地址移入AC0
AC1	VB2的地址			+D, +2, AC1 累加器AC1加2, 指向下一个内存位置VB2
AC0	5678			MOVW *AC1, AC0 将AC1里存的地址移入AC0

提示:

- 欲间接存取字节, 将指针增一或减一, 这样就可访问下一个字节或上一个字节。
- 欲间接存取字, 将指针增二或减二, 这样就可访问下一个字或上一个字。
- 欲间接存取双字, 将指针增四或减四, 这样就可访问下一个双字或上一个双字。
- 欲存取定时器或计数器当前值(此为字数值), 将指针增二或减二。
- 如果在指针递增过程中指向的地址超出了V内存上限, 程序执行时会收到一则运行时错误信息。

8.4 常数数值范围

数据大小	不带符号的整数范围		带符号的整数范围		实数	
	十进制数	十六进制数	十进制数	十六进制数	十进制实数(正数)	十进制实数(负数)
字节	0~255	0~FF	-128~+127	80~7F	—	—
字	0~65535	0~FFFF	-32768~+32767	8000~7FFF	—	—
双字	0~4294967295	0~FFFFFFFF	-2147483648~+2147483647	80000000~7FFFFFFF	+1.175495E-38~+3.402823E+38	-1.175495E-38~-3.402823E+38

常数格式标识符

程序可在许多指令中使用字节、字或双字大小的数据常数。格式标识符控制如何显示常数数值(二进制数字、十进制数字、十六进制数字或 ASCII 字符)。

假设程序常数为十进制数字, 除非您使用格式标识符:

2# 二进制数字

16# 十六进制数字

" ASCII 常数, 使用单引号字符起始与结束 ASCII 字符。对在参数列表中指定常数的绝大多数指令有效。作为数据字节存储。

"" ASCII 常数字符串, 使用双引号字符起始与结束 ASCII 字符。对在参数列表中指定常数字符串的绝大多数指令有效。作为长度字节后接数据字节存储。

二进制常数举例

例如...	数字基数	分隔符	常数数值

2#1101	2	#	1101
2#1101_1111	2	#	11011111

十六进制常数举例

例如...	数字基数	分隔符	常数数值
16#3FB2	16	#	3FB2

ASCII 常数字符范围

ASCII 常数字符的有效范围为 ASCII 32 至 ASCII 255，但不包括 DEL（删除）字符、单引号字符、双引号字符。在此范围之外的 ASCII 字符必须使用特殊\$字符格式。

ASCII 原义字符常数和字符串常数使用说明

单引号和双引号中的每个中文字符编码长度为 2，\$作为转义字符时不算长度。

ASCII 原义字符串常数（双引号）只能赋给 V 内存字节型地址（VB），但双引号内的字符长度不能超过 255。

ASCII 原义字符常数（单引号）可以赋给字节、字和双字型地址（IB, QB, MB, SMB, VB, SB, LB, AC），赋给字节型地址时单引号中的字符长度没有限制，赋给字型地址时字符长度只能为 2，赋给双字型地址时字符长度只能为 4。

地址 举例	字符(串)常量	字符(串)常量对该 地址的有效性	有效字符(串)常量在内存中的格式					
			VB0	VB1	VB2	VB3	VB4	VB5
VB0	'a'	有效	a					
	'a\$1f\$\$\$'\$"	有效	a	0x1F	0x24	0x27	0x22	
	"a 字符"	有效	5	a	0xD7	0xD6	0xB7	0xFB
VW0	'a'	无效，因为长度等于1，不为2						
	'ab'	有效	a	b				
	'\$1f\$'\$"	有效	0x1F	0x24				
	'字'	有效	0xD7	0xD6				
	'a\$1f\$'\$"	无效，因为长度等于3，不为2						
	'a 字'	无效，因为长度等于3，不为2						
	"ab"	无效，字符串常量只能赋给地址 VB						
VD0	'a'	无效，因为长度等于1，不为4						
	'abcd'	有效	a	b	c	d		
	'\$1f\$\$\$'\$"	有效	0x1F	0x24	0x27	0x22		
	'字符'	有效	0xD7	0xD6	0xB7	0xFB		
	'a\$1f\$\$\$'\$"	无效，因为长度等于5，不为4						
	'a 字符'	无效，因为长度等						

		于5，不为4						
	"abcd"	无效，字符串常量只能赋给地址 VB						

使用转义字符\$的字符串说明

当下载或上载数据块及主程序、子程序和中断程序时，使用\$格式字符修饰 ASCII 原义常数。

代码	解释	存储在 PLC 内存中的数据
\$\$	单美元符号	\$(0x24)
\$'	单引号字符	'(0x27)
\$"	双引号字符	"(0x22)
\$L 或 \$l	换行字符	0x0A
\$N 或 \$n	新行字符	0x0A
\$P 或 \$p	进纸，新页	0x0C
\$R 或 \$r	回车字符	0x0D
\$T 或 \$t	制表符，即制表字符	0x09
\$1f	ASCII 码值为 0x1f 的字符	0x1F

8.5 关键字

这些关键字是 ASCII 程序文件的保留字，不能用作符号名或 POU 名称。

下列所有关键字均用大写字母标记，但不区分大小写，可键入小写、大写或大小写混合字母。

关键字	含义	备注
ANY	任何	预留
ARRAY	阵列	预留
BEGIN	开始	
BOOL	布尔	
BYTE	字节	
CHAR	字符	
DATA_BLOCK_TAB		
DATE	日期	预留
DATE_AND_TIME	日期_和_时间	预留
DINT	双整数	
DT		预留
DWORD	双字	
END_DATA_BLOCK_TAB		
END_FUNCTION	结束_功能	
END_FUNCTION_BLOCK	结束_功能_块	
END_ORGANIZATION_BLOCK	结束_组织_块	
END_STRUCT	结束_结构	
END_TYPE	结束_类型	

END_VAR	结束_变量	
FALSE	假	
FUNCTION	功能	
FUNCTION_BLOCK	功能_块	
INT	整数	
NETWORK	网络	
OF		预留
OFF	关闭	
ON	打开	
ORGANIZATION_BLOCK	组织_块	
REAL	实数	
STRING	字符串	
STRUCT	结构	预留
TIME	时间	预留
TIME_OF_DAY	当日时间	预留
TITLE	标题	
TOD	实时时钟	
TRUE	真	
UDT		不支持 - 保留作为将来用
VAR	变量	
VAR_IN_OUT	变量_入_出	
VAR_INPUT	变量_输入	
VAR_OUTPUT	变量_输出	
WORD	字	
DB	数据块	
FC	功能	
FB	功能块	
SFB	系统功能块	
OB	组织块	
SDB		
SBR		
SUBROUTINE_BLOCK	子程序_块	
END_SUBROUTINE_BLOCK	结束_子程序_块	
INTERRUPT_BLOCK	中断_块	
END_INTERRUPT_BLOCK	结束_中断_块	

所有语句列表指令助记符均为关键字。所有带大小标识符的内存类型助记符均为关键字。

9 PLC 在线控制

本章概要

- ☐ PLC 运行/停止模式
- ☐ PLC 上电复位
- ☐ 上载程序
- ☐ 下载程序
- ☐ 清除 PLC 存储区
- ☐ PLC 信息
- ☐ 存储卡配置
- ☐ 存储卡擦除
- ☐ 从 RAM 建立数据块
- ☐ 实时时钟
- ☐ 将 PLC 与当前项目中的块进行比较
- ☐ 运行模式编程
- ☐ PLC 类型选择

9.1 PLC 运行/停止模式

使用下列一种方法改变 PLC 操作模式：

- 手动切换 PLC 的模式开关，可以在两种模式中切换：运行（RUN）、停止（STOP）。
- 选择菜单命令“PLC”→“运行”，将 PLC 切换至运行状态，如果此时模式开关在 STOP 状态，系统会给出错误信息提示开关不在正确位置。选择菜单命令“PLC”→“停止”，将 PLC 切换至停机状态。
- 点击工具条上的“运行”按钮，将 PLC 切换至运行状态，如果此时模式开关在 STOP 状态，系统同样会给出一则错误信息。
- 点击工具条上的“停止”按钮，将 PLC 切换至停机状态。
- 在程序中插入 STOP 指令，使 PLC 在一定条件下停机。

CTSC-200 模式开关说明：

- 将模式开关打在停止模式，停止程序的执行；
- 将模式开关打在运行模式，启动程序的执行；

- 若模式开关为 STOP，且电源状态发生变化，则当电源恢复时，CPU 会进入 STOP 模式；
- 若模式开关为 RUN，且电源状态发生变化，则当电源恢复时，CPU 会进入 RUN 模式。

CTSC-200 操作模式说明：

CTSC-200 有停止和运行两种操作模式。CPU 前面板上的 LED 状态显示了当前的操作模式。

在停止模式下，CTSC-200 不执行程序，不执行输入更新，用户中断条件被禁止，CPU 处于半空闲状态。

您可以新建、编辑、下载、上载或删除用户程序，也可以使用调试协助，例如首次扫描和多次扫描功能。STOP 模式下发生通信中断时，PLC 收到信息并在适当的情况下执行请求。CPU 模式下也会执行 CPU 自诊断程序，包括定期检查操作系统 EEPROM、I/O 模块状态检查和在每次访问扩展 I/O 时执行的 I/O 扩展总线检查。PLC 操作系统会记录致命错误，并强制从运行模式更改为停止模式。如果 PLC 检测到致命错误，在致命错误条件依然存在时不允许从停止模式更改为运行模式。PLC 操作系统功能也记录非致命错误，用于检查，但不会停机。

在运行模式下，CPU200 将执行用户程序。PLC 读取输入、执行程序、写入输出、对通信请求作出应答、更新智能模块、执行内部管理工作以及对中断条件作出应答。在运行模式中，您可以新建、编辑和监控程序操作和数据，使用提供的调试功能可增强追踪程序操作和识别编程问题的能力。

每次扫描周期开始时先读取当前输入位数值，并将这些数值写入输入映像寄存器。与具有实际输入位位于同一个字节中，但无相应的实际物理输入点的输入位，会在每次输入更新循环时在映像寄存器中被清零，除非它被强制。读取输入后，程序从第一条指令开始执行，直至遇到结束指令。遇到结束指令时，PLC 检查系统的智能模块是否需要服务。如果需要，信息将被读取并缓存，以用于循环周期的下一阶段。在扫描周期的信息处理阶段，从通信端口接收的信息被处理。相关的应答完成后被暂存，等待在适当的时间传送给通信请求方。自我诊断检查包括对操作系统 EEPROM 和用户程序存储区以及 I/O 模块状态进行定期检查。

最后，输出映像寄存器数值被写入输出模块，从而完成一次扫描周期。

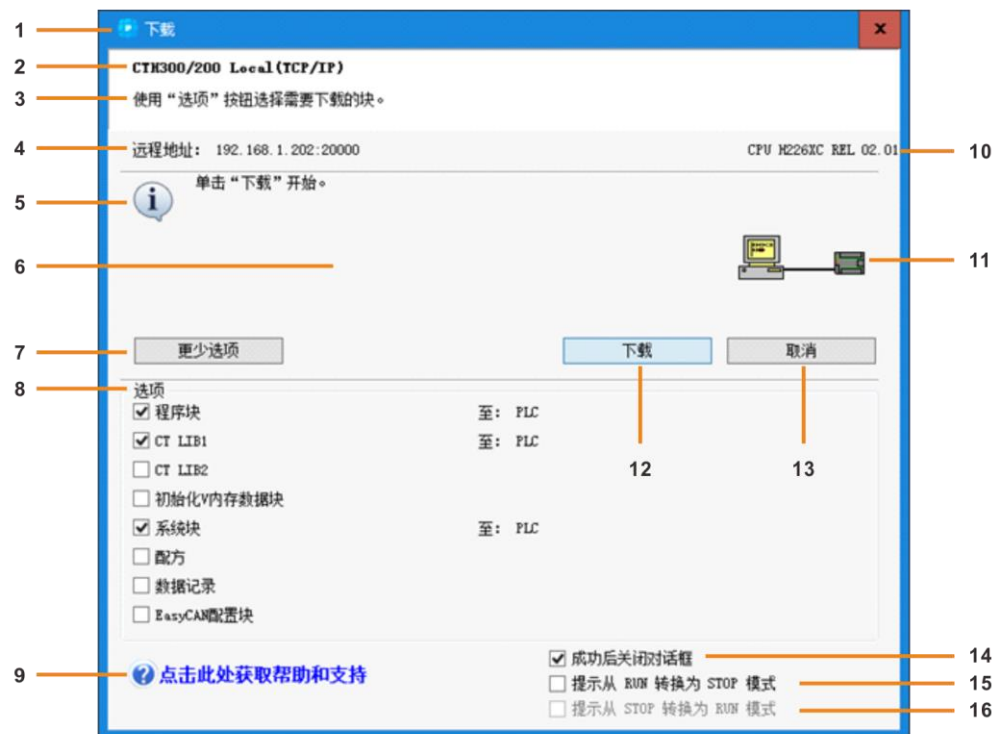
9.2 PLC 上电复位

要从 PLC 清除致命错误并返回 RUN 模式，请选择菜单命令“PLC”→“上电复位”。

如果操作 PLC 存在致命错误，系统错误指示灯 SF 将点亮，程序停止执行。此时必须将 PLC 模式复位为 STOP，然后再设置为 RUN，才能清除错误。可以远程使用“PLC”→“上电复位”命令或以手动方式复位 PLC 模式开关完成此项任务。

9.3 上载/下载/清除

上载/下载/清除对话框示意图



- 1、标题栏: 对话框名称
- 2、连接类型: 通信连接类型
- 3、操作说明与指示: 正在进行的操作说明
- 4、CPU 地址: 通信地址
- 5、状态图标: 表明进程的总体状态

- 6、状态信息区：显示错误信息的区域
- 7、选项按钮：点此按钮可选择选项框可见项的内容
- 8、选项内容：选择要操作的块
- 9、帮助：打开 MagicWorks PLC 帮助（与按 F1 键相同）
- 10、CPU 类型：目标 CPU 的型号和版本
- 11、动画图标：表明操作处理是否在进行中
- 12、选项内容：选择要操作的块操作按钮：如果没有错误存在，此按钮可见，点击可开始操作
- 13、取消按钮：不进行任何操作关闭对话框，上载/下载/清除操作中此按钮不可用
- 14、成功操作后关闭对话框：用于选择是否在成功操作后关闭此窗口的选项
- 15、提示从“RUN”转换为“STOP”模式：用于选择当 CPU 从运行模式转换为停止模式时是否询问用户
- 16、提示从“STOP”转换为“RUN”模式：用于选择当 CPU 从停止模式转换为运行模式时是否询问用户

9.3.1 上载程序

打开上载对话框

可以使用以下几种方法之一将 PLC 的项目组件上载至 MagicWorks PLC 项目管理器：

- 点击工具栏的上载按钮
- 选择菜单命令“PLC”→“上载”
- 按快捷键组合 Ctrl+U

要上载（PLC 至编辑器），PLC 通信必须正常运行。确保网络硬件和 PLC 连接电缆正常操作。

在选项框中选择想要上载的块（程序块、数据块、系统块、配方、数据记录或 CANopen 配置块），如果选项框不可见，请点击“更多选项”按钮使其可见。点击对话框上的上载按钮，然后选定要上载的程序组件就会从 PLC 复制到当前打开的项目，随后您即可保存已上载的程序。

上载全部块

您可以上载全部的块，PLC 不包含符号或状态图信息，因此，您无法上载符号表或状态图。

上载至新的空项目

这是获得程序组件的保险方法。由于项目空置，所以您不必担心上载上来的内容覆盖您的项目数据。如果您希望使用为该项目建立的状态图或符号表内容，请您打开另一个 MagicWorks PLC 工程，并从另一个项目文件复制该信息。

上载至现有项目

如果您希望改写自下载至 PLC 以来对程序进行的全部修改，这是一个好办法。

如果您需要保留下载至 PLC 之后对当前工程所作的任何修改，则不应采用这种方法，因为上载会改写这些块。

执行上载的步骤：

- 1) 打开 MagicWorks PLC Manager 中的一个项目，容纳您将从 PLC 上载的块。

* 如果您希望上载至一个空项目，选择菜单项“文件”→“新建”，或使用“新项目”工具条按钮。

* 如果您希望上载至现有项目，选择菜单项“文件”→“打开”，或使用“打开项目”工具条按钮。

2) 选择菜单项“PLC”→“上载”，或使用工具条中的“上载”按钮，初始化上载程序。

3) “选项”对话框显示程序块、数据块、系统块、配方、数据记录和 CANopen 配置块复选框。请核实已选择您希望上载的块复选框，并取消选择您不希望上载的任何块，然后点击“上载”按钮。

9.3.2 下载程序

下载程序的要求

- 1) 编程设备和可编程控制器中的 CPU 之间必须存在一个连接。
- 2) 必须可以访问可编程控制器。
- 3) 为将块下载至 PLC，在项目的对象属性对话框中必须已经为“使用”选择了条目。
- 4) 您正在下载的程序已经完成了编译，且编译成功。
- 5) CPU 必须处于允许进行下载的工作模式下（停止-STOP 模式）；
- 6) 如果离线打开块，并希望对其进行下载，则 CPU 必须链接到管理器中的在线用户程序。
- 7) 在您下载用户程序之前请复位 CPU，以确保 CPU 上没有任何其他的块。

打开下载对话框

可以使用以下几种方之一法从 MagicWorks PLC 管理器向 PLC 下载项目组件：

- 点击工具栏的下载按钮；
- 选择菜单命令“PLC”→“下载”；
- 按 Ctrl+D 快捷键组合；

要成功的将项目组件下载至 PLC，PLC 通信必须正常运行。确保网络硬件和 PLC 连接电缆正常操作。

在选项框中选择想要下载的块（程序块、数据块、系统块、配方、数据记录或 CANopen 配置块），如果选项框不可见，请点击“更多选项”按钮使其可见。点击对话框上的下载按钮，您选定要下载的程序组件就会从当前项目复制到 PLC 的内存。

执行下载的步骤

如果您已经成功地在运行 MagicWorks PLC 的个人计算机和 PLC 之间建立通讯，您可以将程序下载至该 PLC。请遵循下列步骤：

- 1) 下载至 PLC 之前，您必须将 PLC 设为“STOP”模式。
- 2) 点击工具条中的“下载”按钮或选择菜单项“文件”→“下载”打开“下载”对话框。
- 3) “选项”框显示程序块、数据块、系统块、配方、数据记录和 CANopen 配置块复选框。请核实已选择您希望下载的块复选框，并取消选择您不希望下载的任何块。
- 4) 点击“下载”按钮，开始下载程序。
- 5) 如果下载成功或有错误，信息框会给出相应的提示。

<注意> 从当前工程将程序块、初始化 V 内存数据块、系统块和 CANopen 配置块下载至 PLC 时，下载的块内容会覆盖 PLC 中的当前块内容（如果 PLC 中存在这些块），而下载的配方与数据记录则会覆盖存储卡的相应内容。在您开始下载之前，请核实您希望覆盖的块。

9.3.3 清除 PLC 存储区

要清除 PLC 存储区的指定区域，选择菜单命令“PLC”→“清除”。

<备注> 清除之前必须将 CPU 切换至 STOP 状态。

在选项框里设置要清除的块，如果选项框不可见，点击“更多选项”按钮使其可见。

忘记 PLC 密码怎么办

如果您忘记了 PLC 密码，则必须将程序块、数据块和系统块三块一起清除，然后重新载入程序。清除 PLC 存储区使 PLC 处于 STOP 模式，并将 PLC 复原为工厂默认值，但 PLC 地址、波特率和实时时钟除外。

清除 PLC 中程序的步骤

- 1) 要清除 PLC 中的程序，选择菜单命令“PLC”→“清除”打开清除对话框。
- 2) 选择所有的复选框，并点击“确定”按钮，核实执行的步骤。
- 3) 如果已经配置密码，应用程序会弹出一个密码验证对话框，要清除密码，请在密码验证对话框中输入 CLEARPLC（不区分大小写），继续执行“清除”操作。

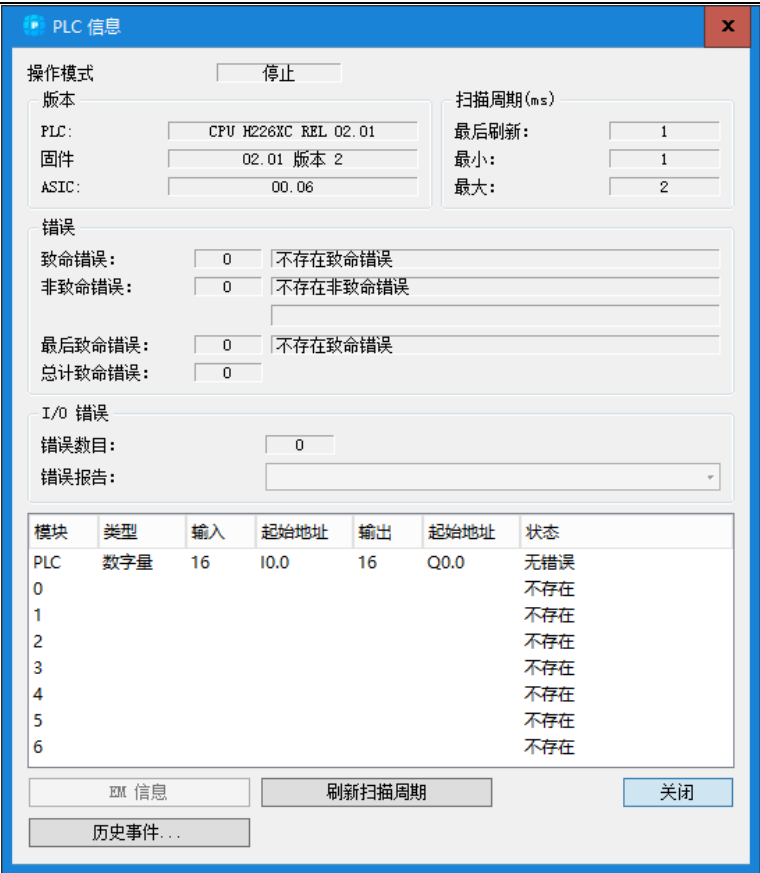


警告

清除 PLC 存储区影响数字量和模拟量输出的状态。工厂默认值是在 PLC 位于 STOP（停止）模式时，将数字量输出复位为零（关闭）。但是，如果您已经使用系统块的“输出表”标签定义数字量输出的“安全状态”，您需要注意删除系统块会删除该“安全状态”定义。下一次您将 PLC 从 RUN（运行）模式转入 STOP（停止）模式时，数字量输出会被设为零（工厂默认值）。此外，当您删除程序块时，模拟量输出被冻结为当前值。直至您下载新的程序块，对模拟输出状态进行更改的唯一方法是使用状态表清除 PLC 存储区时，如果 CTSC-200 PLC 与设备相连，对数字输出状态的更改可传送至装置。而且，您删除程序块后，因为输出被“冻结”，与模拟输出相连的装置无法按照指定的方式响应。如果清除了 PLC 存储区，却没有仔细考虑对数字量和模拟量输出的影响，设备可能按照无法预料的方式操作，可能导致人员死亡或严重伤害及 / 或设备损坏。始终遵循正确的安全预防措施，在清除 PLC 存储区之前确保程序处于安全状态。

9.4 PLC 信息

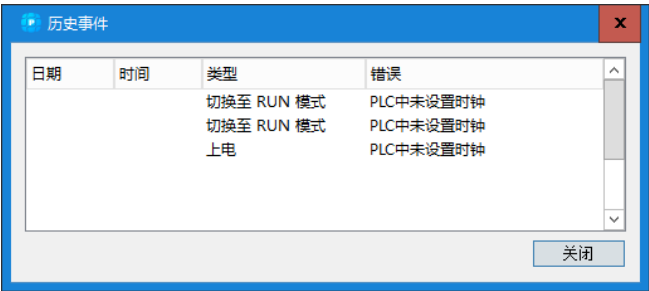
选择菜单命令“PLC”→“信息”或在通信界面双击搜索到的 PLC 可查看 PLC 信息，例如 PLC 型号和版本号码、操作模式、扫描速率、I/O 模块配置、CPU 和 I/O 模块错误及事件历史日志。



如果您的 CPU 上连着智能模块，要查看其信息，请选择合适的模块列表项，然后点击 **EM 信息** 按钮。如此就会出现一个单独的对话框，指明模块类型、模块修订版本，以及任何模块错误或其他相关信息。

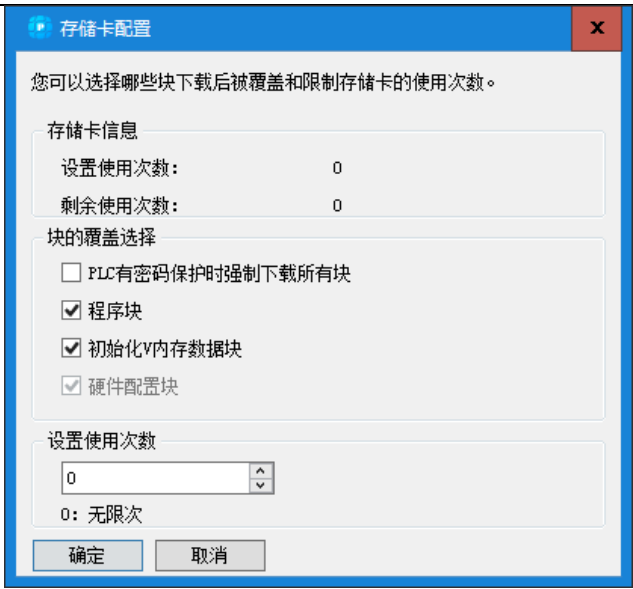
刷新扫描周期按钮允许您刷新三个“扫描周期”域中的次数。

如果您的 PLC 类型支持“事件历史”日志，该按钮就会被使能。请点击“事件历史”按钮查看上电、模式转换及致命错误的日志记录。



9.5 存储卡配置

在项目管理器界面菜单项选择“PLC”→“存储卡配置”打开如下对话框，随后即可对存储卡进行配置。



选择将程序块或初始化 V 内存数据块进行下载后被覆盖
设置存储卡的使用次数，设置为 0 即表示可以无限次的使用。
存储卡的使用次数（设置的使用次数、剩余的使用次数）将显示在“编程卡信息”中。

9.6 存储卡擦除

可选的存储卡为您的程序和项目数据提供了便携的永久性存储区（EEPROM）能力。您可以使用菜单命令“PLC”→“存储卡擦除”擦除已存储的全部数据。
存储卡的全部内容都会擦除，包括：

- 用户程序
- 存储于永久 V 存储区的数据
- 系统块（PLC 配置）
- 配方和数据记录配置
- 作为执行数据记录指令之结果而存储的数据
- 用户文件



提示

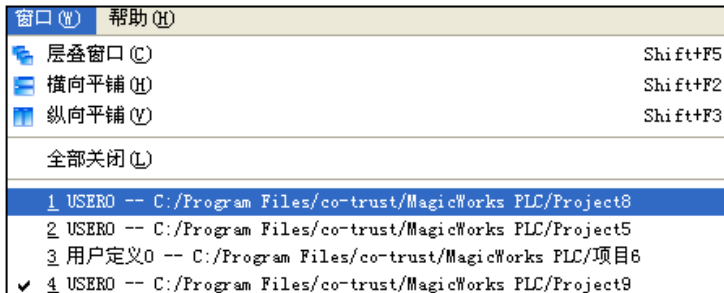
- 只有当 PLC 已上电和处于 STOP 模式，并且适当的存储卡已安装后才能擦除存储卡的内容。
- 擦除命令只可用于 64K 或 256K 存储卡，即使将 32K 存储卡插入上述的 CPU，依然不能擦除。
- 您可以在 PLC 已上电的情况下安装或拆卸存储卡。如要安装，请取下 PLC 上的塑料盖，然后将存储卡插入 PLC。
- 当心静电放电能够损坏存储卡或 PLC 上的接口。在拿放存储卡时，请与已接地的导电垫接触或使用接地腕带。请将存储卡存放在导电容器内。

9.7 从 RAM 建立数据块

初始化 V 内存数据块

在项目管理器中您可以为一个项目建立一个或多个初始化数据页，使用以下一种方法打开初始化数据页：

- 在项目管理器中初始化 V 内存数据块文件夹中双击任一数据页图标或是在图标上单击鼠标右键选择“打开”菜单项即可启动数据块编辑器进入编辑状态。
- 如果在数据块编辑器中有一个以上数据页，使用数据块编辑器窗口菜单中的排列窗口菜单项或是选择其中的已打开窗口列表菜单项切换数据页。



导入/导出初始化数据页至 ASCII 文本文件

- 从 ASCII 文本文件导入初始化数据页；
- 导出初始化数据页到 ASCII 文本文件；

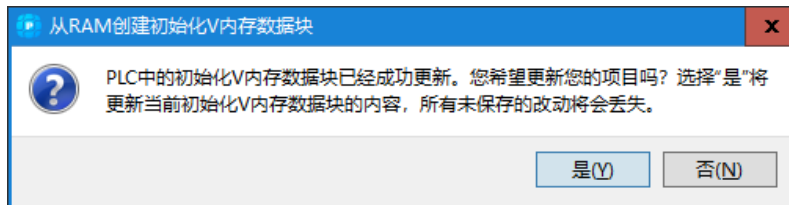
从 RAM 建立数据块

在项目管理器中选择“PLC”→“从 RAM 建立数据块”菜单命令，将 PLC 的 V 存储区保存至 PLC 的 EEPROM。

此操作的包括三个步骤：

- 从 PLC 读取 V 存储区中的当前值；
- 从 PLC 上载数据块，然后将 V 存储区数据作为数据块的赋值部分生成一个新的数据块；
- 将新生成的数据块下载至 PLC；

下载完成后，系统会弹出一个对话框，询问要不要更新项目，如下图所示：



选择“是”，则用新生成的数据块替换原来项目中的数据块，选择“否”，结束操作。



提示

- 进行此操作时，CPU 必须处于 STOP（停止）模式。
- 倘若 CPU RAM 内有 V 存储区地址的内容已经修改，并且这些地址在上一次数据块下载时未得到指定，那么在您执行“从 RAM 建立数据块”命令时，这些已修改的地址会被指定给一个新的标签名（例如 PLC_DATA1）。当程序的执行向新的 V 地址写入数据时，或用状态表修改新的 V 地址时，此情形就可能发生。

9.8 实时时钟

要查看或设置存储在 PLC 中的当前时间和日期，选择菜单命令“PLC”→“实时时钟”。

您可以在“实时时钟”对话框中设置时间和日期。PLC 型号 214、215、216、221、222、224、226、H35-00 支持“实时时钟”及 TODR/TODW 程序指令。

CPU 222、CPU 224、CPU 226（2.0 版或更高版本）、H35-00 支持夏时制时间的自动调整。

<备注> 请参考附录 [B 指令集](#) 中的 TODR / TODW 指令。

9.9 将 PLC 与当前项目中的块进行比较

要将 MagicWorks PLC 中当前打开的项目与 PLC 进行比较，选择菜单命令“PLC”→“比较”即可。请点击有关复选框，选择要比较的项目组件，比较结果会显示在下拉框中。

9.10 运行模式编程



警告

- 在 PLC 运行模式编程时，向 PLC 进行下载操作，会使得程序立即在 PLC 执行。因此没有防范错误的余地；
- 程序中的错误可能会引起对人员的严重伤害和对机器设备的损害，仅限合格的操作员使用此功能，并在使用前阅读本节有关运行模式编程的细节。

在运行（RUN）模式下执行程序编辑功能，可以使您不必将 PLC 切换到 STOP 状态，而直接将改动的程序下载到 PLC，比如：

您可以更快的对程序进行调试，对当前程序进行较小的改动（更改参数值）直接下载到 PLC，而不需停机。

对实际的控制过程进行下载时，必须考虑对机器设备和操作员造成的伤害。

启动运行模式编程

在执行运行模式编程前，请检查：

- 1) 您的 PLC 类型支持在线运行模式编程；
- 2) 在菜单项“系统块”→“运行模式编程”中，不勾选“禁用运行模式编程”；
- 3) 您的 PLC 已经和 MagicWorks PLC 软件之间建立了通讯；
- 4) 确保当前的 CPU 运行程序中不包括 CT 的保密库；

如果上述条件满足，你可以通过菜单命令“PLC”→“运行模式编程”，进入运行模式编程。



提示

请在执行运行模式编程前保存当前的项目工程，进入运行模式编程首先进入上载程序阶段，在此之前，MagicWorks PLC 不会对当前项目是否进行保存进行询问，而是在您点击上载按钮后直接覆盖当前项目程序。

运行模式编程模式正确启动后，鼠标将变为  形状。在程序块完成所需的编辑后，请先保存程序，再执行下载操作，更改的程序才能在 PLC 中生效；若没有执行保存操作，MagicWorks PLC 将下载之前的程序。

运行模式编程下载

执行运行模式编程下载时，只能下载程序块（同时也只对程序块进行编译），系统块和程序块不会下载。

开始下载后，直到 MagicWorks PLC 下载完成之前，您不能执行其他任务。

如果程序中存在编译错误，将会在输出窗口提示出错的位置和其他信息。

指定背景时间

在运行模式编程开始下载程序时，PLC 需要一些时间对修改的程序进行背景编译（PLC 同时执行当前的载入程序）。

您可通过“系统块”→“背景时间”项进行设置，默认值为 10%，最高可设置为 50%。更高的背景时间意味着更快的编译速度，但会延长程序总体的扫描时间，可能会对当前执行的程序产生不良影响。

选择运行模式编程的依据

使用运行模式编程还是使用在 CPU 在 STOP 模式的下载编程方式，请考虑在运行模式编程下的一些因素：

- 1) 如果在运行模式编程中删除输出的控制逻辑，输出保持其最后状态，直到 CPU 断电停止运行或进入 STOP（停止）模式。
- 2) 如果在运行模式编程中删除正在运行的 HSC 或 PTO/PWM 功能，这些功能继续运行，直到 CPU 断电停止运行或进入 STOP（停止）模式。
- 3) 如果在运行模式编程中删除 ATCH 指令但中断程序本身并未删除，则中断程序继续执行，直到 CPU 断电停止运行或进入 STOP（停止）模式。
- 4) 如果在运行模式编程中删除 DTCH 指令但中断程序本身并未删除，则中断程序不会停止执行，直到 CPU 断电停止运行或进入 STOP（停止）模式。
- 5) 如果在运行模式编程中增加 ATCH 指令，并以遇到首次扫描标志为条件，这些事件不会被激活，直到 CPU 断电重启或出现 STOP-to-RUN（停止至运行）模式转换。
- 6) 如果在运行模式编程中删除 ENI 指令，则中断程序继续操作，直到 CPU 断电停止运行或出现 RUN-to-STOP（运行至停止）模式转换。
- 7) 如果在运行模式编程中修改通讯（RCV/XMT/NETR/NETW）指令的内存表地址，而且接收指令在从旧程序向新程序转换的过程中被激活，则接收的数据被写入旧表地址。只有在（向旧地址的）当前接收请求完成后才开始使用新地址，但因为已经对程序进行了编辑，程序将在新地址中寻找数据（但却不存在）。
- 8) 首次扫描标志不受运行模式编程的影响，只有在 CPU 重新上电运行或执行 STOP-RUN（停止到运行）操作后，才会被执行。

上升沿或下降沿的处理

在运行模式编程下，您可以对包含有上升沿或下降沿（EU 或 ED 指令）的网络进行操作。

MagicWorks PLC 暂时会为每一个 EU/ED 指令分配一个号码，您可也更改此号码的顺序，但不允许使用重复的 EU/ED 号码，并只能最多使用 255 个 EU/ED 指令。

局部变量保持功能衔接

在运行模式编辑时，如果用到局部变量保持功能，尽量不要再程序中插入新的 CALL 调用。因为局部变量保持功能要依赖于 CALL 指令在程序中的排序来定位内存。如果在程序中插入了新的 CALL 指令，则新插入的 CALL 指令之后的一切调用的子程序局部变量无法和之前衔接；而新插入的 CALL 指令之前的一切调用的子程序局部变量可以保持。

CT 的运动控制库就用到了局部变量保持功能，因此在运行模式编辑中尽量不要增加运控指令。

CT 保密库不支持运行模式编程

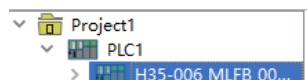
包含有 CT 保密库的程序执行上载时（运行模式编程需要上载），如果运行模式编程时，程序名 MAIN 更改为 ct_lib1 或 ct_lib2，则会下载失败。

运行模式编程的硬件/固件支持

CTSC-200 CPU V3.92 及以后版本所有型号都支持运行模式下编辑程序，CTSC-100 CPU 不支持运行模式下程序编辑。

9.11 PLC 类型选择

若需要使用“PLC 类型”对话框，请选择菜单命令“PLC”→“类型”，您可以在左侧的项目树中查看当前项目的 PLC 类型。

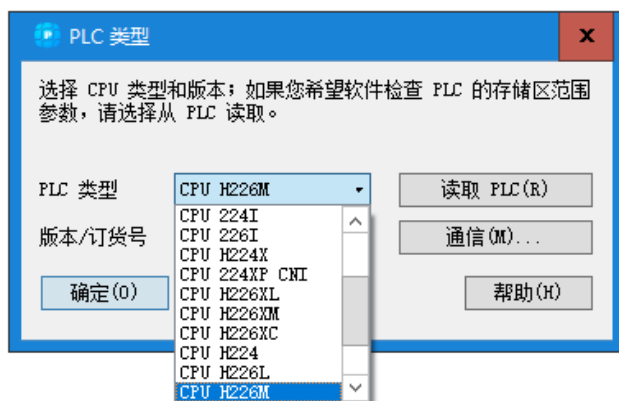


通过选择 PLC 类型，MagicWorks PLC 会自动启动语法检查，可防止在建立程序时发生错误。此外，在您为项目指定 PLC 类型后，指令树用红色标签 X 表示对当前 PLC 无效的指令。

如果您打开一个新项目，没有进行 PLC 类型选择即开始编辑程序，MagicWorks PLC 编辑器和编译程序允许您对并非在所有的 CTSC-200 PLC 型号中均受支持的指令、地址和 PLC 功能进行编程。如果您在不受目标 PLC 支持的项目中使用指令、地址或 PLC 功能，当您尝试下载项目时，会被 PLC 拒绝。

指定 PLC 类型选择

点击“PLC 类型”的下拉框以选择一种 PLC，列表框中会列出软件支持的所有 PLC 类型。

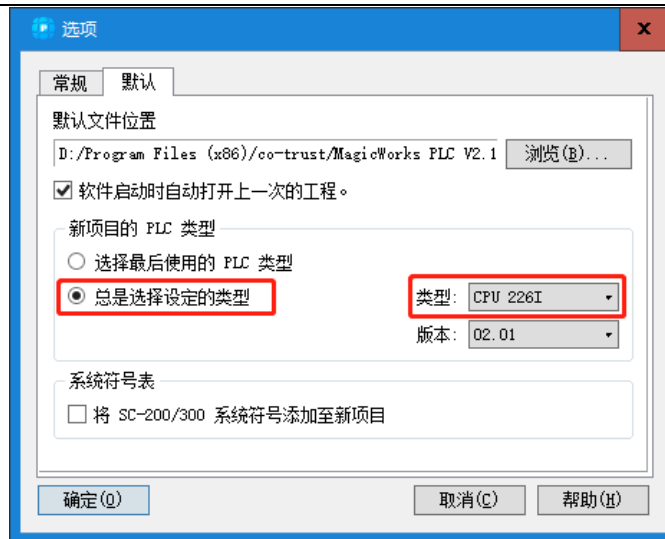


作这项选择时，CPU 226 最新硬件版本的范围和功能限制就得到应用。为了确保在执行范围检查时考虑到您的 PLC 型号和您的硬件版本，您可以让 MagicWorks PLC 从 PLC 直接读取信息。

<备注> 当您指定 PLC 类型选择时，不需要配置通信，只有在希望 MagicWorks PLC 为您读取远程 PLC 类型时，您才需要建立通信。

设置默认 PLC 类型

您可以选择让新项目始终使用最后选择的 PLC 类型，或者让新项目始终将默认值设为指定 PLC 类型。选择“工具”→“选项”并点击“常规”标签，设置默认 PLC 类型。



- 如果您选择将默认值设为最后选择的 PLC 类型，您每次建立新项目时，该项目的 PLC 类型均与上一个项目的 PLC 类型相同。
- 如果您选择“总是选择设置的类型”选项，建立的每个新项目均设为您从下拉列表中选择
的 PLC 类型。当前 PLC 类型包含在该列表中。如果当前 PLC 类型还有具体发布版本
号（系使用“读取 PLC”功能获得），您还是可以选择让每个新项目使用该 PLC 类型。

<备注> 在新安装 MagicWorks PLC 时，默认选项被设为“选择最后使用的 PLC 类型”。

读取远程 PLC 类型

欲使 MagicWorks PLC 读取 PLC 型号和硬件版本，点击“PLC 类型”对话框中的按钮。

<注意> 您必须已经成功地建立通信才能读取 PLC 型号和硬件信息。您可以使用“PLC 类型”对话框中的“通信”按钮，建立与 PLC 的通信，或者在有多个 PLC 与通信网络连接时选择一个目标 PLC。

类型不匹配

如果您尝试向 PLC 下载程序，但 PLC 与项目的 PLC 类型选择不匹配，会出现警告信息。您可以通过点击下载页面的“改动项目”按钮将当前项目 PLC 类型设为读取的 PLC 类型。

10 调试与监控程序

当您在运行 MagicWorks PLC 的编程设备和 PLC 之间建立通信并向 PLC 成功下载程序后，即可使用状态监控与调试功能。

“状态监控”是指显示程序在 PLC 中执行时的有关 PLC 数据的当前值和状态的信息。您可以使用状态表监控和程序状态监控窗口读取、写入和强制 PLC 数据值。

在控制程序的执行过程中，PLC 数据的动态改变可用三种不同方式查看：

- **程序状态监控**

在程序编辑器窗口中显示状态数据：当前 PLC 数据值会显示在引用该数据的 STL 语句或 LAD 图形旁边。LAD 图形也显示能流，由此可看出哪个图形分支在活动中。

- **状态表监控**

在表格中显示状态数据：每行指定一个要监视的 PLC 数据。您可以指定一个存储区地址、格式、当前值及新值。

- **趋势图显示**

用随时间而变的 PLC 数据绘图跟踪状态数据：您可以将现有的状态表在表格视图和趋势视图之间切换；新的趋势数据亦可在趋势视图中直接生成。

本章概要

- ☐ 如何模拟过程条件
- ☐ 如何使用程序状态监控
- ☐ 如何使用状态表监控

10.1 如何模拟过程条件

您可以在程序状态监控或者状态表监控过程中将新数值写入或强制至操作数，以模拟控制过程的状态。

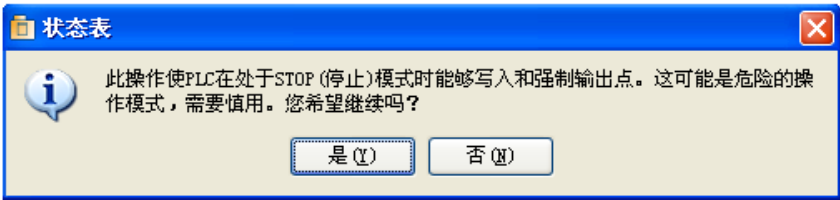
使用程序编辑器和状态表编辑器的“调试”菜单命令，可以开始以下“调试”功能：

图标	功能	描述
	单次读取（仅限状态表）	要想获得全部变量的一次更新值，可使用“单次读取”功能。在状态表监控开启情况下，状态表从 PLC 连续读取状态值更新。监控关闭后，可以使用“单次读取”按钮进行一次更新。

	全部写入（仅限状态表）	使用“全部写入”功能，可将新值列中输入的值写入 PLC 中对应的地址。
	强制	要将地址强制为某一数值，您必须首先规定所需的数值，可通过读取数值（如果您希望强制为当前数值）或键入数值（如果您希望将地址强制为新数值）来完成。一旦某地址被强制后，PLC 每次扫描时都会将强制值重新应用于该地址，直至对该地址执行取消强制操作。 “强制”功能覆盖立即读取或立即写入指令。“强制”功能还覆盖被在转换为 STOP（停止）模式时输出指定数值的配置，意思就是，如果 CPU 进入 STOP（停止）模式，输出被设为强制数值，而并非输出表中配置的数值。 用户可以以字节、字或双字强制 V 或 M 存储区；以偶数字节起始的字强制 AI 或 AQ 存储区（例如 AIW6 或 AIW14）；或强制 I/O 点（I 和 Q 位址）以模拟逻辑条件。您可以一次强制 16 个（V、M、AI 或 AQ）地址和所有的 I/O 位（所有的 I 和 Q 位地址）。
	显性强制	该图标表示该地址已经被显性强制。该单元格数值在地址被取消强制之前无法改变。
	隐性强制	该图标表示该地址已被隐性强制。即包含这个地址的另一个较大范围的地址已经被显性强制。
	部分显性强制	该图标表示该地址的一部分区域被显性强制。
	取消强制	对于程序状态监控和状态表监控，选择一个地址并使用“取消强制”从该特定地址移除强制。您也可以选中一个参数，然后用鼠标右键点击该参数，调出“强制”和“取消强制”功能弹出菜单。
	取消全部强制（仅限状态表）	如果您希望从 PLC 中的全部地址清除强制，可使用“取消全部强制”功能。应用“取消全部强制”之前不必选择单个地址。
	读取全部强制（仅限状态表）	使用“读取全部强制”功能时，状态表的“当前值”会显示已经被显性强制、隐性强制或部分显性强制的所有地址。
	查看趋势图	趋势图用随时间而变的 PLC 数据绘制图形以跟踪状态数据：您可以把现有的状态表在表格视图和趋势视图之间切换。新的趋势数据亦可在趋势视图中直接定义查看。趋势显示的行号与状态表的行号对应。新的趋势数据属性亦可在趋势图中直接指定。
	暂停趋势图	该图标表示暂停状态表的趋势视图显示。

在 STOP 模式下写入和强制输出

要在 STOP（停止）模式中启用“写入”和“强制”输出功能，选择菜单命令“调试”→“STOP（停止）模式下写入-强制输出”，出于安全考虑，系统会弹出确认对话框，如下图所示：

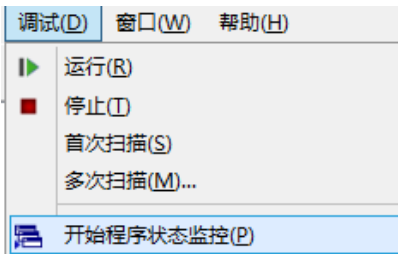


- 警告**
- 在写入或强制输出时，如果 COTRUST PLC 与其他设备相连，这些改动可能被传送到该装置，可能引起该装置无法预料的操作，亦可能造成人员死亡和严重伤害及 / 或装置损坏。
 - 每次打开程序编辑器或状态表编辑器时，菜单选项默认为不选用，防止您在 PLC 处于 STOP（停止）模式时写入或强制输出。选择菜单选项会为当前项目的当前编辑操作启用写入和强制功能。

10.2 如何使用程序状态监控

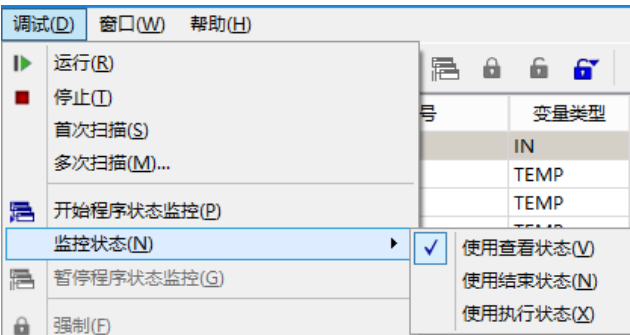
在程序编辑器窗口与程序状态监控相关的几个操作：

- 1、开始/停止程序状态监控：“调试”>“开始/停止程序状态监控”或点击工具栏“”图标可开始或停止程序监控。



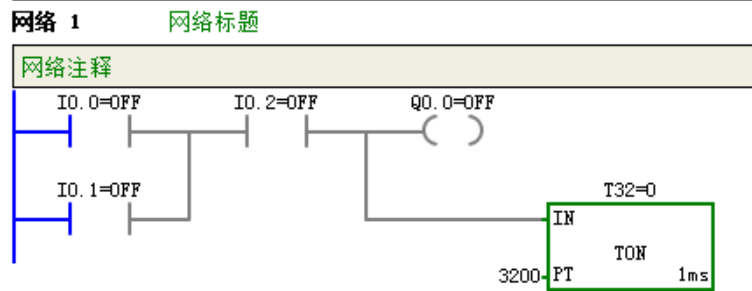
在开始监控和停止监控之间切换，如果当前已经开启程序状态监控，调试菜单下会出现停止程序状态监控命令，对应的工具按钮会显示被按下状态，此时选择停止程序状态监控菜单命令或点击处于按下状态的开始/停止状态监控按钮，程序状态监控就会停止，按钮弹起。

- 2、监控状态



“监控状态”是指查看显示程序在 PLC 中执行有关 PLC 数据当前值和功率流的信息。目前 MagicWorks PLC 支持三种监控状态：查看状态、结束状态、执行状态。

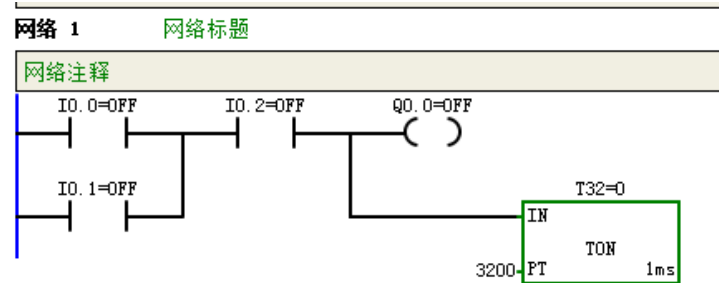
- 1) 查看状态：



查看状态下，显示指令/能流的导通状态，同时也显示未导通指令的参数值。导通指令的参数值与执行状态一致，未导通指令的参数值取指令扫描结束时的变量值，即与状态表中监控的一致。

特别说明：查看状态监控时需要从 PLC 读取的数据比较多，导致通信压力增大和界面卡顿，因此不建议一直使用此状态监控，只在必要时切换至此方式监控。

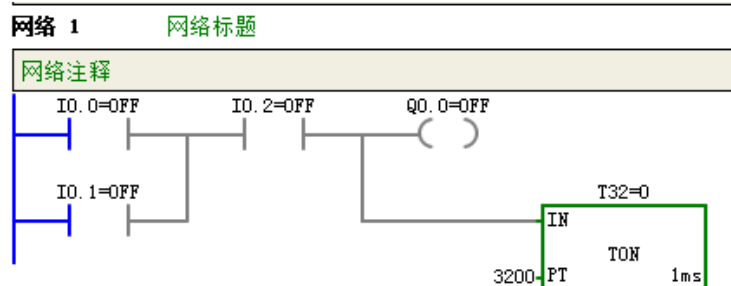
2) 结束状态:



接通电源或逻辑真的节点和线圈显示为蓝色。

“结束状态”显示在程序扫描结束时读取的状态结果。这些结果可能不会反映 PLC 数据地址的所有数值变化，因为随后的程序指令在程序扫描结束之前可能会写入和重新写入数值。由于快速的 PLC 扫描周期和相对慢速的 PLC 状态数据通信之间存在的速度差别，“结束状态”显示的是几个扫描周期结束时采集的数据值。

3) 执行状态:



执行状态下颜色显示:

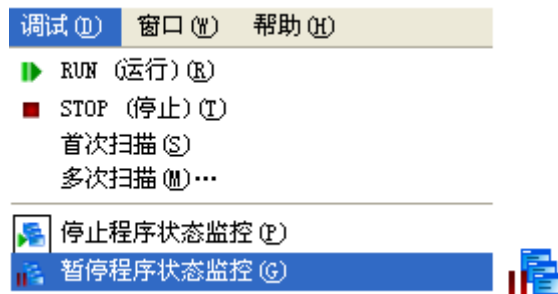
- 程序被扫描时，电源横杆会带有色彩。
- 图形中的功率流用色彩表示。
- 触点接通时，指令会变色显示。
- 线圈输出接通时，指令会变色显示。
- 绿色计时器和计数器表示计时器和计数器包含有效数据。
- 灰色（默认值）表示无功率流、指令未扫描（跳过或未调用）或位于 STOP 模式的 PLC。

- 红色表示执行的指令有误。
- 方框和 SUBR 指令，指令接通并准确成功执行时，方框和 SUBR 指令带有色彩。
- 跳接和标签指令现用时，显示为功率流色彩。如果为非现用，则显示为灰色。

执行状态视图显示操作数数值，并表示每条指令在位于扫描循环执行程序阶段时的功率流。执行状态可以显示可能被执行后续程序指令改写的中间数据值。所有显示的 PLC 数据值均从一个程序循环扫描收集。

注意：标准型 CTSC200 和 CTH200 每次最多允许监控 100 条 STL 指令，因此建议使用这类型 PLC 时，同一个网络的 STL 条数不要超过 100，否则监控时会有部分指令取不到正确结果。

3、暂停程序状态监控



在暂停监控和重新连续监控之间切换，更新暂停之后，触发网络中的状态值在您查看时保持不变，直到重新连续监控后数据才连续更新。此功能只有在开启了程序状态监控后才能使用。如果当前已经暂停监控，调试菜单下暂停程序状态监控的命令周围会有一个选中的方框标记，对应的工具按钮会显示被按下状态，此时再选择暂停程序状态监控菜单命令或点击已按下的按钮，按钮弹起。

程序状态监控的说明

- 锁图标表示该数值已经被“显性”或直接强制为当前正在显示的数值。
- 灰色锁图标表示该数值已经被“隐性”强制，即不对地址进行直接强制，但存储区落入另一个被显性强制的较大区域中。例如，如果您显性强制 VW0，则 VB0 和 VB1 被隐性强制，因为它们包含在 VW0 中。
- 半锁图标表示数值被“部分”强制。例如，如果您显性强制 VB1，则 VW0 被部分强制，因为其中的一个字节 VB1 被强制。
- 正在扫描程序时，电源母线显示为蓝色。
- 图形中的能流用蓝色表示。
- 触点接通时，指令会显示为蓝色。
- 线圈输出接通时时，指令会显示为蓝色。
- 方框类指令接通电源并准确无误地成功执行时，指令显示为蓝色。
- 绿色定时器和计数器表示定时器和计数器包含有效数据。
- 红色表示指令执行有误。
- 跳转和标签指令激活时，显示为能流的颜色。如果为非激活，则显示为灰色。
- 灰色表示无能流、指令未扫描（跳过或未调用）或位于 STOP（停止）模式的 PLC。

- 如果您将地址当作指针使用，但该地址包含的数值是有效的存储区地址，则用红色显示错误。

LAD 和 STL 程序状态视图显示操作数数值，并显示出每条指令在扫描周期中执行程序阶段时的能流状态。程序状态可以显示可能被后续执行的程序指令覆盖的中间数据值。所有的程序状态数据值均从一个程序扫描周期采集。

程序状态监控提示：

- PLC 位于 RUN（运行）模式时，会显示彩色状态值和元素。
- PLC 位于 STOP（停止）模式时，状态值和元素变成灰色。
- 只有在 PLC 位于 RUN 模式并实际采集状态数据时才会显示代表强制的锁图标。
- 写入、强制和取消强制只有在 PLC 位于 RUN 模式并实际采集状态数据时才能使用。
- 当在 STOP 模式中采集状态时，写入、强制和取消强制弹出菜单选项被禁用。
- 当状态被“暂停”时，写入、强制和取消强制弹出菜单选项可以使用。
- 如果输入的是常数，写入、强制和取消强制弹出菜单选项被禁止。
- 强制和取消强制工具栏按钮和弹出菜单选项仅在用于带有 V、M、AI 和 AQ 存储区类型的字节、字和双字长度时才有效。
- 强制和取消强制工具栏按钮和弹出菜单选项仅对位、字节、字和双字长度的 I 和 Q 存储区类型有效。

指令状态位说明：

	使能位输出（ENO）
	SCR 位
	SCR 区标志位
	结果为零（SM1.0）
	溢出（SM1.1）
	结果为负（SM1.2）
	除数为零（SM1.3）
	操作超出表长度（SM1.4）
	空表读取（SM1.5）
	非 BCD 至整数转换（SM1.6）
	ASCII 至十六进制数错误（SM1.7）

STL 程序中程序状态监控举例：

当您打开 STL 中的状态监控时，程序编辑器窗口被分为一个代码区（左侧）和一个状态区（右侧）。可以根据您希望监控的数据类型定制状态区。

在 STL 状态中共有三个可监控的数据类型：

- 操作数：每条指令最多可监控三个操作数。

- 逻辑堆栈：最多可监控四个来自逻辑堆栈的最新数值。
- 指令状态位：最多可监控十二个状态位。

LAD 图执行状态监控举例

- 当您下载程序至 PLC 时，会被提示将 PLC 转换为 STOP（停止）模式。如果您希望连续更新程序状态，请务必将 PLC 转换回 RUN（运行）模式。您可以选中 MagicWorks PLC 状态栏，查看 PLC 的当前模式。该状态栏每 20~30 秒更新一次。
- 当您启动程序状态监控时，很多其他 MagicWorks PLC 功能会被禁止。例如，除非您再次关闭程序状态监控，否则不能编辑程序。其他功能（例如从一个程序编辑器的视图改变为另一个程序编辑器的视图）会使程序状态监控自动关闭。如果您希望继续查看状态，则必须重新选择适当的状态选项。
- 如果您位于程序不在执行的区域（例如，子程序、中断程序或由于 JMP 指令被跳过的区域），就不会有状态显示，因为代码没有被执行扫描，所有的指令操作数和能流线均显示为灰色。

状态监控通信与扫描周期

PLC 在连续循环中读取输入、执行程序逻辑、写入输出和执行系统操作和通信。该扫描周期速度极快，每秒执行多次。虽然 CT Program Editor 会快速发出状态请求，但还是应当认识到并不能查看到 PLC 中出现的每一个事件，这一点十分重要。

由于 PLC 和监控程序状态的编程设备间存在通信延时，显示的操作数数值总是在状态显示中改变之前即在 PLC 中改变。虽然更新显示无须很长时间，但可能出现这种状况，即向操作数发出一条强制指令，其数值实际已经在 PLC 中改变，但没有在程序状态监控中改变。

<备注> 一次只允许一个状态监控窗口运行。如果您尝试在任何其他与同一个 PLC 连接的 MagicWorks PLC 应用程序中运行状态监控，则会收到错误信息。



提示

- 当您强制操作数时，始终注意状态显示，直至显示锁图标。一旦显示锁图标，则表示数值已经正确强制（否则您会看到被强制的数值，可以纠正错误的强制操作）。
- 如您希望强制一个数值起伏波动的操作数，请使用状态表。这样，您就无须“捕捉”来自不断变化的显示的数值，而直接将所需的数值键入“新值”列，然后发出“强制”命令。

用程序状态监控模拟过程条件（读取、强制、取消强制和全部取消强制）

您可以模拟过程条件，方法是在程序状态监控过程中从程序编辑器向操作数写入或强制新值。
要写入操作数：

- 直接点击操作数，然后用鼠标右键直接点击操作数，并从弹出菜单选择“写入”。

要强制单个操作数：

- 直接点击操作数，然后从“调试”工具栏点击“强制”图标。
- 直接用鼠标右键点击操作数，并从弹出菜单选择“强制”。

要取消强制单个操作数：

- 直接点击操作数，然后从“调试”工具栏点击“取消强制”图标。
- 直接用鼠标右键点击操作数，并从弹出菜单选择“取消强制”。

要取消强制全部强制数据：

- 从“调试”工具栏点击“取消全部强制”图标。

强制功能覆盖立即读取或立即写入指令。强制功能还覆盖转换为 **STOP**（停止）模式时配置的指定输出值：如果 **CPU** 进入 **STOP**（停止）模式，输出被设为强制数值，而不是预先配置的数值。



警告

在程序中强制数据时，在程序每次扫描时将操作数复原为该数值，而会忽略输入 / 输出条件或其他正常情况下应该影响操作数的程序逻辑无关。强制可能导致程序操作无法预料，无法预料的程序操作可能导致人员死亡或严重伤害和 / 或设备损坏。强制功能只是调试程序的辅助工具，切勿为了弥补过程装置的故障而执行强制。仅限合格人员使用强制功能。强制程序数值后，务必通知所有有权维修或调试程序的人员。

使用“程序状态监控”按钮或“调试”菜单命令，开始状态数据通信，并使能调试工具。

您还可以使用硬件的输入模拟器开关以手动方式接通和关断数字量输入，并通过 **PLC** 上的 **LED** 指示灯观察数字量输出的状态。

从“调试”工具栏或“调试”菜单列表访问以下程序状态监控功能，参考章节 [10.1 如何模拟过程条件](#)。

<注释>

以上范例显示 **LAD** 执行状态结果。如果您在调试程序并发现带有强制操作数的网络没有按照您期待的方式执行，可使用“交叉引用”或“编辑”→“查找”命令，查找其他可能引用并改变操作数数值的网络。

当您使用垂直滚动按钮时，程序代码行和程序状态显示中的行会同步前进。滚动时，由于需要从 **PLC** 获取数据，程序状态显示可能需要一点时间才能更新。如果您将程序状态显示更改为较小的尺寸，“操作数”列也会显示水平滚动按钮。您还可以重新调整“操作数”列尺寸，使之能够纳入较小的显示区域。您无法更改“逻辑堆栈”或“指令状态位”列，也无法沿水平方向滚动这些列。如果这些列无法纳入程序状态显示区域，则应当扩大水平方向程序状态显示。

10.3 如何使用状态表监控

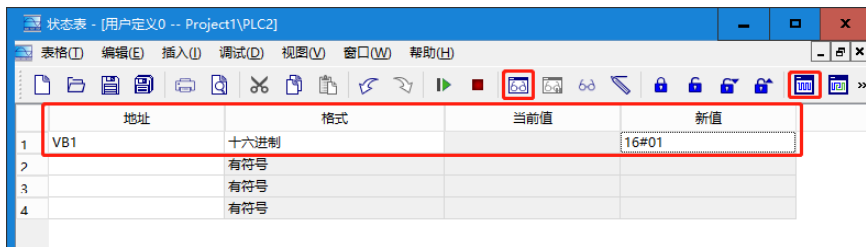
在项目管理器中您可以为一个项目建立一个或多个状态表，在将程序下载至 PLC 之后监控和调试程序操作。使用以下方法之一打开状态表：

- 在资源管理器中状态表目录下双击任一状态表页图标或是在图标上单击鼠标右键选择“打开”菜单项即可启动状态表编辑器进入该状态。
- 如果在状态表编辑器中有一个以上状态表，使用状态表编辑器窗口菜单中的排列窗口菜单项或者选择其中的已打开窗口列表菜单项切换状态表。

10.3.1 状态表监控

如果您已经打开一个空状态表，编辑行，增加您希望监控的 PLC 数据地址。在状态表任意行点击右键选择“插入”→“插入行”即可在选中行上方插入新行。

打开状态表，可查看或编辑表内容，您可在地址列和新值列手动键入内容，在格式列下拉项选择格式。点击图标即可启动状态表监控，进而采集状态信息。点击图标即可查看趋势视图。



在控制程序的执行过程中，状态表中数据的动态改变可用两种不同方式查看：

表格显示：在一表格中显示状态数据，每行指定一个要监视的 PLC 数据值。您可以指定存储区地址、格式、当前值及新值。

趋势图显示：用随时间而变的 PLC 数据绘图以跟踪状态数据，您可以就现有的状态表在表格视图和趋势视图之间切换。新的趋势数据亦可在趋势视图中直接赋值。

使用以下一种方法启动在状态表中载入 PLC 数据的通信：

- 要连续采集状态表信息：开启状态表→使用菜单命令“调试”→“状态表监控”或使用“状态表监控”工具栏按钮。
- 要获得单个数值的“快照”，可使用“单次读取”功能：使用菜单命令“调试”→“单次读取”或使用“单次读取”工具栏按钮（已经开启状态表监控，“单次读取”功能被禁止）。



提示

- 打开状态表并不意味着开始自动查看状态。您必须启动状态表监控，才能采集状态信息；
- 如果表是空的，启动状态表监控没有任何意义：您必须首先将程序数据（操作数）放在“地址”列中，并为“格式”列中的每一个数据选择数据类型，以“建立”一个表；
- 您可以采集 STOP（停止）模式的 PLC 状态，检查程序运行的初始或最终状态。PLC 必须位于 RUN（运行）模式，才能从连续执行的程序采集数据。

如果 PLC 位于 RUN（运行）模式，程序在连续扫描的状况下执行。您可以启动状态表监控，

连续更新状态表数值。您还可以使用“单次读取”功能，采集状态表数值的单个“快照”。

您在查看状态表时，还可以将 PLC 设为 STOP（停止）模式，使用“首次扫描”或“多次扫描”功能，监控有限次数扫描的程序操作。

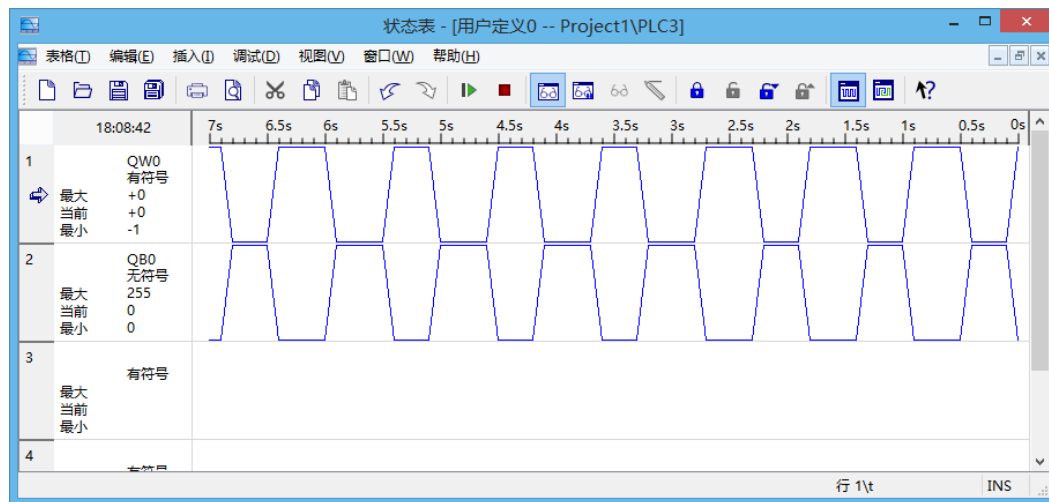
状态表监控提示

- 无论 PLC 位于何种操作模式，锁图标始终会显示。
- 如果 PLC 位于 RUN（运行）或 STOP（停止）模式，可使用写入、强制和取消强制功能。但是，在 STOP（停止）模式中，写入输出（Q 和 AQ）选项只有在“调试”→停止模式下写入-强制输出的项目被选中时才能。
- 状态值在 PLC 位于 RUN（运行）或 STOP（停止）模式时都被显示。
- 强制和取消强制工具栏按钮选项仅对字节、字和双字长度的 V、M、AI 和 AQ 存储区类型可用。工具栏按钮只有在所有选项可以被强制 / 取消强制时才会变成可用状态。
- 强制和取消强制工具栏按钮和弹出菜单选项仅对位、字节、字和双字长度的 I 和 Q 存储区类型可用。工具栏按钮只有在所有选项可以被适当强制 / 取消强制（这是选择多行的情况）时才会变成可用。
- 强制和取消强制工具栏按钮和弹出菜单选项对所有完全或部分隐性强制数据无效。显性强制数值可以被强制为一个新数值，而无须取消强制后再强制新数值。

在状态表中使用调试功能

您可以在程序状态监控过程中从程序编辑器，或者在状态表监控中将新数值写入或强制至操作数，以模拟控制过程的状态。

10.3.2 使用趋势视图



趋势图显示细节

使用下列方法在状态表的表格视图和趋势视图之间切换：

- 1) 使用状态表编辑器菜单命令“视图”→“查看趋势视图”。
- 2) 用鼠标右键点击状态表，然后复选或取消查看趋势图命令。
- 3) 点击调试工具栏的趋势图按钮 .

趋势图用随时间而变的 PLC 数据绘制图形以跟踪状态数据：您可以把现有的状态表在表格视图和趋势视图之间切换。新的趋势数据亦可在趋势视图中直接定义查看。趋势显示的行号与状态表的行号对应。新的趋势数据属性亦可在趋势图中直接指定。

倘若在选取“切换趋势图”按钮之前没有使用“状态表监控”按钮，供趋势功能使用的状态采集将不会开始。如要启动 PLC 数据采集，您必须选取状态表监控按钮。如果状态表已启动，趋势图将会立即开始显示已采集的数据。

通过点击“暂停趋势图”按钮或选择菜单命令“调试”→“暂停趋势图”可以冻结趋势绘图。

趋势视图词汇表

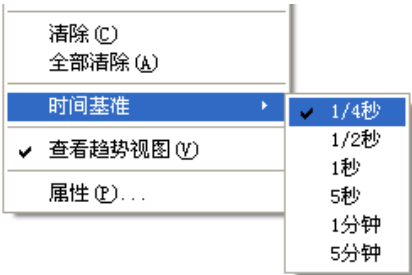
名称	含义
趋势显示	基于时间而绘制的 PLC 数据图形，可通过选取“切换趋势图”按钮从预先配置的状态表创建。
趋势绘图	对于状态表中已配置的每个地址，PLC 数据图形之屏幕位置会以时间为自变量得到显示。您不能调整趋势绘图的宽度。
数据点或趋势点	数据点意指出自 PLC 中某地址的单个数据。数据点是在相应地址的趋势绘图中赋予最后一个像素时的数值。
范围	趋势数据的绘图分布在最小值和最大值之间。此范围通常定义了数值的某个固定跨度。
自由范围	如果没有给趋势绘图选取范围，则最小值和最大值之间的数据范围会动态变化。
采样速率或采集速率	定义由 PLC 采集数据的速率。此速率将是固定的，并且将基于选定的趋势显示时间单位计算得出。
趋势窗口	用于描述对趋势数据进行绘图的时间窗口；即由趋势绘图屏幕宽度代表的时间跨度。
实时趋势	实时趋势意指以实时方式采集和在趋势窗口中显示数据。一旦此数据填满趋势窗口，最先的数据就会从趋势绘图上消失。

趋势图的单次读取和写入按钮与状态表上的相应按钮作用相同。“强制”、“取消强制”、“全部取消强制”及“读取全部强制”的作用与在状态表中相似。

实时趋势功能不支持历史趋势，即超出一个趋势窗口之时间跨度的趋势数据不会得到保留。

将光标放在分隔趋势行的横线上直至出现双箭头光标。在光标为双箭头时，向上拖动以减少或向下拖动以增加高度。

您可以用秒或分为单位设置趋势绘图的时间基准，其默认值为 1s。配置的时间基准会在趋势图上用 s（秒）或 m（分）显示，其位置为此图顶部附近的趋势窗口时间刻度内。如果您更改趋势图的时间基准，整个图的数据都会被清除并用新的时间基准重新显示。



在选定一个趋势行后，用鼠标右键点击会弹出菜单，点击“属性”可指定此行的趋势数据属性。



通过“趋势数据属性”对话框，您可以对趋势绘图做出下列变更：

- 编辑地址（例如：将 VW0 更改为 VW10）。
- 增加一个新地址，如果选定的点以前没有得到配置。
- 更改显示格式类型。
- 基于选定的显示格式设置范围。

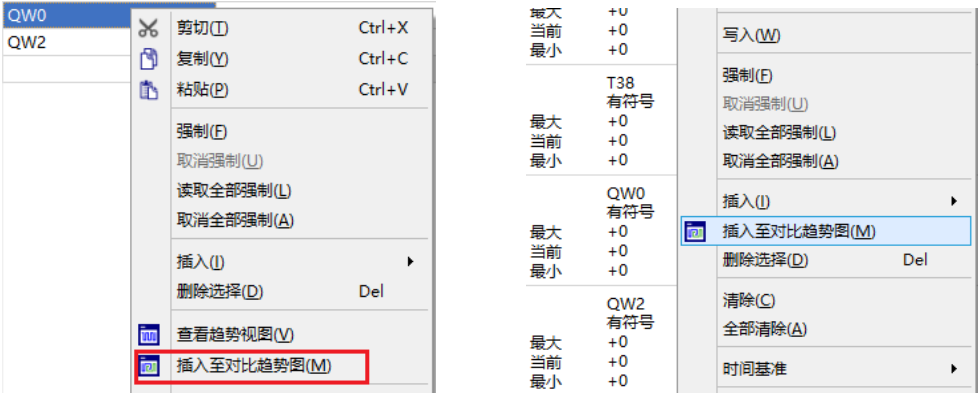
如果显示格式被修改，所有数据将会重新显示，并且可能因选定的新格式而改变。如果这些格式在数值上是兼容的，例如有符号或无符号数，趋势绘图将只会进行可能的范围变更。如果有些数值现在是在旧范围之外，并且范围已锁定，那么此绘图将以最大或最小范围值处中断。如果此绘图配置为自由范围，那么如有必要，这些数据将使用新的范围值重新显示。

按照默认规定，固定范围跨度不会启用；因此，这些趋势图绘图将为自由范围。如要能够针对给定趋势点选取范围值，请复选“锁定范围”复选框；这将令此复选框下面的域可接受输入。由此对话框对地址或显示格式做出的任何更改将同时影响状态表和趋势图。

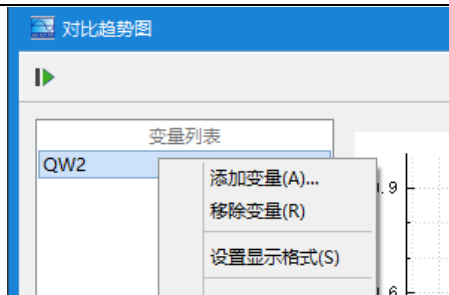
10.3.3 使用对比趋势视图

提示：执行“插入对比趋势图”操作前需在状态表中写入变量，位变量不允许插入。

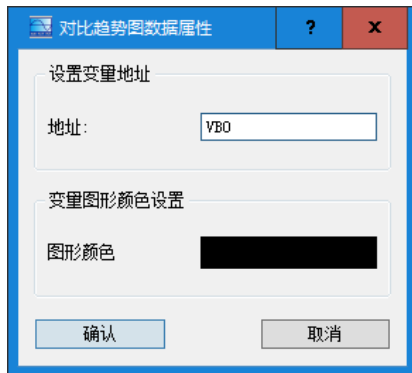
1、插入对比趋势图：在“状态表”界面右键单击变量，选择“插入至对比趋势图”如下左图，或在“查看趋势图”界面右键选择“插入至对比趋势图”如下右图，打开对比趋势图视图界面。



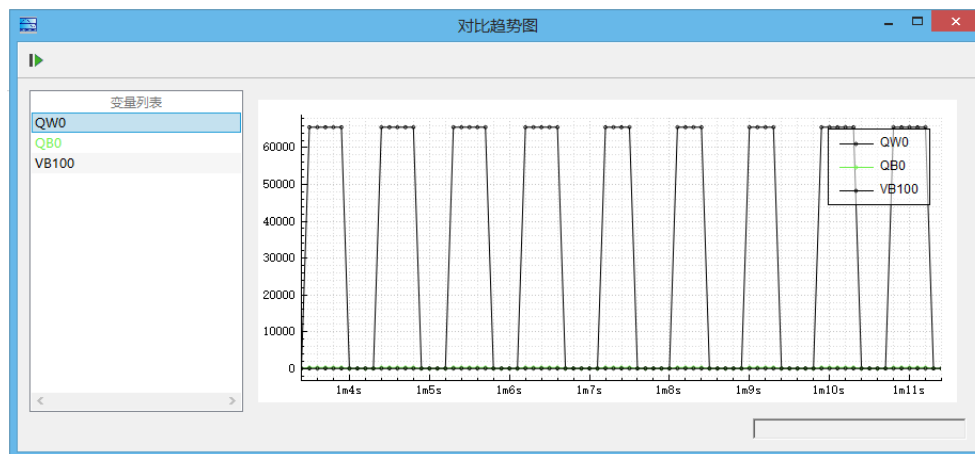
2、添加/移除变量：在“对比趋势图”界面右键点击列表可添加或删除变量，最多可添加三组变量，对已添加的变量可选“移除变量”，选择后将删除该项变量。



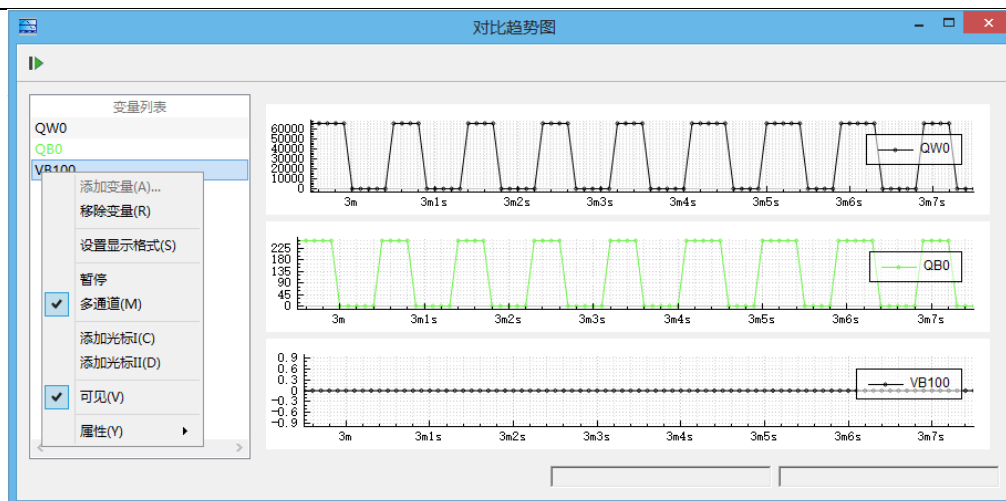
选择“添加变量”后，将出现如下左图所示界面，输入要进行添加的变量地址，点击颜色条可进行图形颜色（列表中变量及右侧坐标中图形颜色）的设置，如下右图所示，设置完成后点击“确认”完成变量的添加。



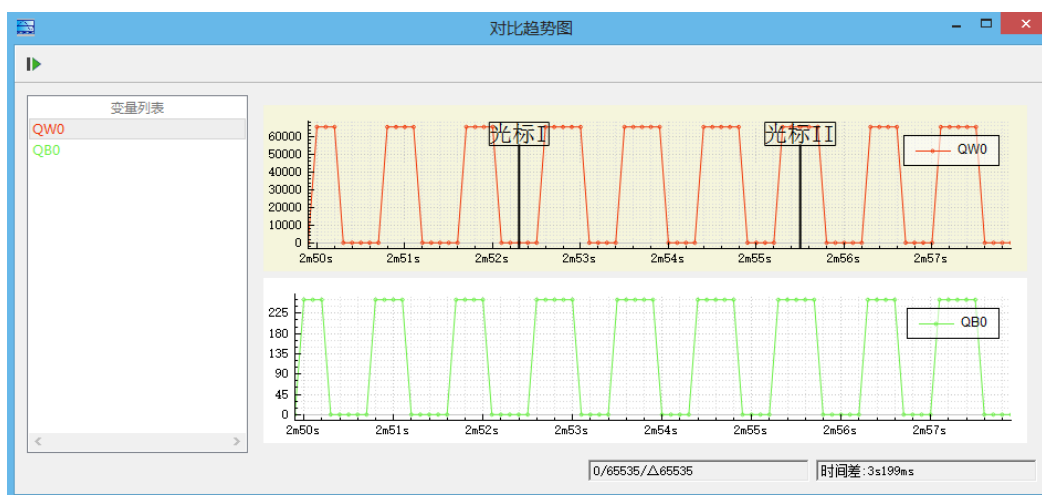
添加变量后的对比趋势图如下图所示，界面右侧为变量点数据对比折线图。添加变量后坐标系默认为自动跟踪显示最新数据曲线的数据，按住鼠标左键并移动可暂停跟踪功能，单击鼠标右键恢复跟踪功能。鼠标滑轮键在暂停跟踪功能时可缩放横坐标系统，“Alt 加滚轮”可缩放纵坐标，跟踪状态无法缩放。鼠标右键点击列表中的变量也可选择“暂停”或“恢复”画面。



3、多通道：右键点击列表中的变量，勾选“多通道”选项，变量图线将以单个变量一组的形式在坐标图中显示出来。若变量列表变量点个数少于 2 个时多通道功能不可用。若监控变量个数为两个，且处于多通道状态时，作“移除变量”操作后会自动改变为单通道状态。如下图所示：

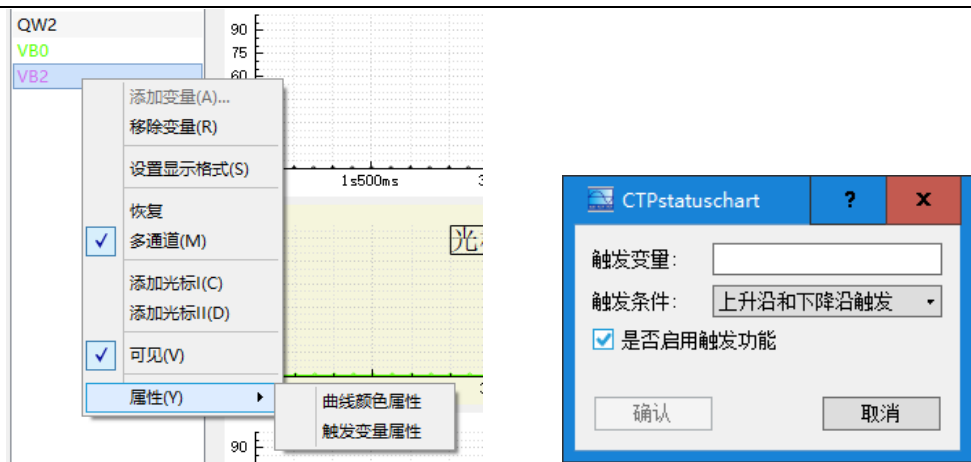


4、添加光标：右键点击列表中的变量，选择“添加光标”，可为变量图线添加两组光标。鼠标左键双击光标可选中/取消选中光标，光标线选中后为红色。光标选中后，鼠标左键单击坐标系任意位置，光标会移动至点击位置。单次只能选中一个光标。当两个光标都已添加时，移动任何一个光标，底部状态栏会显示两个光标之间的时间差，而变量列表中会显示每个变量的数据差，两个光标位置可自由放置。若坐标系失去焦点会暂停计算光标间的时间差和数据差。



对比趋势图处于多通道模式时，变量列表中一个变量对应一个坐标系统，因此选中变量添加光标后会添加至对应的坐标系统，坐标系统右上角标签文字与变量名一致。选中变量取消光标也只会取消对应坐标系统的光标，不会影响其他变量的光标状态。当变量对应坐标系统两个光标都添加后，底部状态栏前面会显示光标间的时间差，后面会显示光标间的该变量的数据差。只有当该坐标系统被选中时才开始计算时间差和数据差并显示。选中的坐标系统背景颜色为淡黄色，若坐标系统失去焦点会暂停计算坐标系统时间差和数据差，状态栏会维持显示最后一次计算的时间差和数据差。

5、可见和属性：右键点击列表中的变量，“可见”选项设置该变量对应图线在右图中的显示情况，勾选“可见”则该变量对应图线将显示在右侧，反之不显示，最多可选择两组变量“不可见”。选择“属性”选项，可设置图线颜色及触发变量。



勾选“是否启用触发功能”，可设置“触发变量”和“触发条件”，触发条件可选“上升沿和下降沿触发”、“上升沿触发”、“下降沿触发”。

趋势图清除

在“查看趋势图”界面，点击右键，在弹出的菜单项选择“清除”，将清除趋势图内当前选定的已收集数据。选择“全部清除”则清除此趋势图界面内全部条目的全部数据，包括未被滚动到可见范围内的数据。

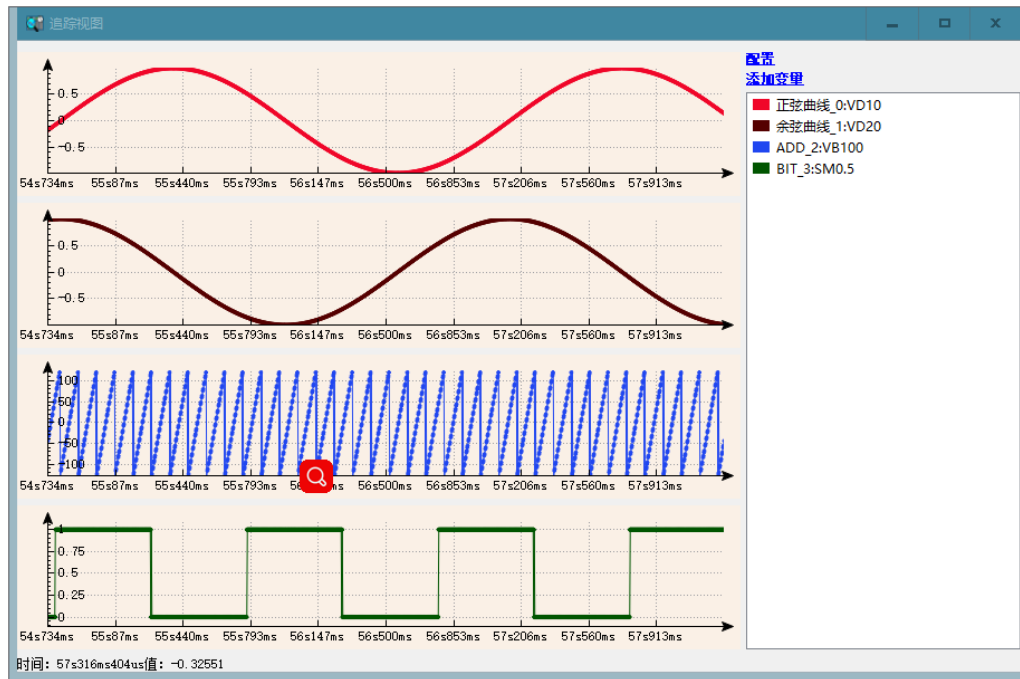
10.3.4 追踪视图

追踪功能用于在一个或多个图表中配置和显示特定于应用程序的跟踪数据。

在应用程序运行时，可以在“追踪视图”中查看跟踪变量的值曲线记录。要求是设置跟踪配置，将跟踪配置传输到 PLC，并开始跟踪记录。记录的数据将传输到上位机，并根据配置以图表形式显示。跟踪时可以浏览数据。

启动“追踪”功能，用户可在状态表 ->菜单 ->视图 ->追踪 或在状态表工具栏中点击图标启动追踪视图。

如下图所示：左边是数据显示图表，右边是变量列表，目前可添加最多 8 个变量和 8 个图表。



配置对话框

对话框包括了对追踪工程的数据记录的追踪配置，点击“追踪视图”右上方的“配置”或鼠标右键是弹出菜单中的“配置”可以进入到配置对话框。

添加变量：添加一个要监控的变量，目前规格：最多可添加 8 个变量。

- 可在“追踪视图”右上点击添加变量添加
- 可在“追踪记录”视图中右键菜单中添加
- 在图表视图中选中其中一个图表的“显示变量”结点在右键弹出菜单中添加

删除变量：在追踪视图中选中一个变量右键删除变量，弹出确认对话框，确认后删除变量，注意：变量删除后不能撤消

分配变量：在追踪视图中选中一个变量在右键弹出菜单中选中“分配到图表”，可以把变量分配到指定的图表中。在添加变量的时候会默认分配到第一个图表中。目前可最多添加 8 个图表。

添加图表：图表视图中可通过右键菜单“添加图表”可在视图中插入一个新的图表。

删除图表：图表视图中可选择要删除的图表通过右键菜单“删除图表”，并在对话框中确认后可删除图表。注意：图表删除后不能撤消。

记录设置：点击追踪记录中的“追踪”可在右边进行“记录设置”

启用触发器：通过使能一个变量来记录一段时间内的数据，点击右边复选框可以使能/不使能，当使能触发器时，还需要配置触发变量，触发沿和触发后采样时间。

触发变量：当使能触发器的时候有效，被用作触发器的信号，有效的触发信号是一个变量，PLC 中的一个位地址。例如 M0.0。

触发沿：当使能触发器的时候有效，定义触发边沿检测：

上升沿：对于 BOOL 触发变量，触发在值从 FALSE 变成 TRUE 的时候发生。

下降沿：对于 BOOL 触发变量，触发发生在当值从 TRUE 变成 FALSE 的时候。

上升沿和下降沿：对于 BOOL 触发变量，触发发生在值改变的时候。

触发后采样时间：触发器在触发后继续采样指定的时间后停止采样。

采样周期：定义的 PLC 采样的时间周期，可在下拉框中选择 PLC 周期、EtherCAT 周期、脉冲周期和 1ms 中断。

变量设置：对添加的变量或已经存在的变量进行配置

变量地址：输入要监控的 PLC 中的有效地址，当地址无效时会报错，必须配置。

变量名称：当前变量的别名，可供用户更好地识别变量，可以不配置。

显示格式：共有 4 种显示格式供用户选择：有符号、无符号、浮点型和位。

曲线颜色：添加变量时会随机生成一种颜色，用户也可自行修改。

线类型：两点之间连线的类型是，可选择直线、阶梯和没有连线。

点类型：有圆点、交叉和无三种样式选择。

时间轴配置：点击图表视图中的时间轴结点可配置对时间轴进行设置。

显示方式：监控数据的过程中时间轴（X 轴）的变化方式：

自动：时间轴自动缩放

固定长度：显示一段常数长度。

固定：固定显示一段数值从最小到最大。

最小值：段的初始值。显示模式为“固定”时有效。

最大值：段的结束值。显示模式为“固定”时有效。

长度：常数段长度。

网格：时间轴方向图的网格线。

Y 轴配置

点击图表视图中的 Y 轴结点可配置对 Y 轴进行设置。

显示方式：监控数据的过程中 Y 轴的变化方式：

自动：时间轴自动缩放。

固定：固定显示一段数值从最小到最大

最小值：段的初始值。显示模式为“固定”时有效。

最大值：段的结束值。显示模式为“固定”时有效。

网格：Y 轴方向图的网格线。

描述：Y 轴被标记为“描述”。

跟踪图的预览图：可以预览表配置好的图表，也可进行图表的颜色。

追踪视图右键菜单

配置：可打开配置对框，如果当前处于监控状态，将会自动暂停

下载配置：将配置好的变量下载到 PLC，如果数据有误将会报错，否则会自动开始读取数据。

如果修改过变量就必须重新下载数据之后才能读：如添加、删除过变量，修改过变量的地址，显示格式等。

注意：下载前要确保软件已边上 PLC 并选用的是“CTH300/200 Local(TCP/IP)”通信接口，否则无法边接到 PLC。

开始/暂时读数：下载数据后，可以开始/暂停读数。

光标：  光标是一个跟踪光标，平行于 y 轴中垂直的黑线。可以用于标识光标位置的时间和对应的 y 值，如果光标所在的时间点没有对应的 Y 值，将显示离光标最近的一个点的值。

在没有跟踪光标可用时，添加一个光标到跟踪图。

当一个跟踪光标可用时，再添加一个光标到跟踪图。

当两个跟踪光标可用时，删除显示的跟踪光标。

无任何痕迹光标跟踪图：在这种模式下，你可以通过鼠标指针轨迹图运行。集中于光标的 X 值，会以普通风格显示在状态栏中（例如，时间: 1m23s456ms Value: 1）。

一个跟踪光标的跟踪图：在状态栏和 y 值，输出结果主要标志是跟踪光标的时间。例子：

Time:1m23s456ms。

两个跟踪光标的跟踪图：在状态栏中，输出两次，由该两个跟踪光标（例如标记的时间间隔，Time: 1m23s456ms - Time: 1m24s456ms (Δ 1s)）。

跟踪图的用户输入：如果一个或两个跟踪光标可用，那么你可以沿着 X 轴移动。

鼠标：拖动跟踪光标的三角形到其它位置。在状态栏中同步更新 y 值。

键盘：方向键左右可移动黑色跟踪光标。

显示/隐藏变量：如果不想查看某个变量，可以在右边变量列表中选中变量右键弹出菜单中“隐藏变量”，变量就不会在图表中显示。如果某个变量已经被隐藏了又想重新查看变量的曲线，可以在右边变量列表中选中变量右键弹出菜单中“显示变量”，该变量就会重新显示到图表上。

重置视图：根据时间轴和 Y 轴配置的显示模式,使视图恢复默认刷新状态。

多通道：追踪视图中以第一个图表为模板为每一个变量分配一个图表并重新初始化图表。

注意：使用的“多通道”功能后旧的图表配置数据将被删除。

单通道：追踪视图把所有变量都放到第一个图表中并重新初始化图表。

注意：使用的“单通道”功能后旧的图表配置数据将被删除。

保存轨迹：将读取的数据保存到一个*.cttrace 的文件中。

装载轨迹：从*.cttrace 文件中读装载保存的数据。

11 诊断

本章概要

- 诊断信息
- 时间戳不匹配

11.1 诊断信息

通过出现的诊断符号，可查看是否有可供模块使用的诊断消息。诊断符号说明了相应模块的状态，对于 CPU 则说明了其工作模式。

打开 MagicWorks PLC 软件→选择菜单项“PLC”→在项目管理器界面打开“硬件组态”→“获取诊断信息”即可打开诊断窗口。

表 11-1 诊断信息格式

列名	意义	备注
日期	诊断事件发生的日期，格式为：年.月.日	
时间	诊断事件发生的时间，格式为：时：分：秒：毫秒	
类型	诊断事件的类别	
过程变量	SOE 事件发生时，过程变量的值（仅 SOE 事件才有过程变量）	最多可组态16个 SOE 事件；过程变量的类型和显示格式可在 CPU “属性” → “SOE 配置” 中组态

11.1.1 PLC 非致命错误

非致命错误可能降低 PLC 的某些性能，但不会使 PLC 无法执行用户程序和更新 I/O。

表 11-2 内存非致命错误信息

支持的事件类型	类型编码	编码	事件描述
CPU 非致命错误	0x7	0x00	无错误
		0x01	在执行 HDEF 方框之前启用 HSC
		0x02	输入中断分配冲突，分配至已经分配给 HSC 的点
		0x03	输入分配冲突，分配至已经分配给输入中断或其他 HSC 的 HSC
		0x04	尝试在中断程序中执行 ENI、DISI 或 HDEF 指令
		0x05	尝试在完成第一个 HSC/PLS 之前执行第二个带有

				相同号码的 HSC/PLS，与主程序中的 HSC/PLS 发生中断程序冲突
			0x06	间接编址错误
			0x07	TODW（日写入时间）或 TODR（日读取时间）数据错误
			0x08	超出最大用户子程序嵌套层数
			0x09	在端口0中同时执行 XMT/RCV 指令
			0x0A	尝试通过执行另一条用于相同 HSC 的 HDEF 指令重新定义 HSC
			0x0B	在端口1中同时执行 XMT/RCV 指令
			0x0C	保留
			0x0D	保留
			0x0E	保留
			0x0F	在比较触点指令中遇到非法数字数值
			0x10	保留
			0x11	保留
			0x12	保留
			0x13	非法 PID 回路表
			0x80	程序过大，CPU 无法生成可执行代码；请缩小程序大小
			0x81	保留
			0x82	非法指令；检查指令助记符0
			0x83	缺少 MEND，或主程序中不允许存在指令；增加 MEND 指令或移除不正确的指令
			0x85	缺少 FOR；增加 FOR 指令或删除 NEXT 指令
			0x86	缺少 NEXT；增加 NEXT 指令或删除 FOR 指令
			0x87	缺少标签（LBL、INT、子程序）；增加适当的标签
			0x88	缺少 RET 或子程序中不允许存在指令；在子程序结尾处增加 RET 或移除不正确的指令
			0x89	缺少 RETI 或中断程序中不允许存在指令；在中断程序结尾处增加 RETI 或移除不正确的指令
			0x8B	至 SCR 段非法 JMP 或从 SCR 段非法 JMP
			0x8C	重复标签（LBL、INT、SBR）；为其中一个标签重新命名
			0x8D	非法标签（LBL、INT、SBR）；核实未超出允许使用的标签数
			0x90	非法参数；核实指令允许使用的参数
			0x91	范围错误（包括地址信息）；检查操作数范围
			0x92	指令计数域错误（包括计数信息）；核实最大计数
			0x93	超过 FOR/NEXT 嵌套层数
			0x94	用地址信息向非易失性内存写入范围错误
			0x95	缺少 LSCR 指令（载入 SCR）
			0x96	缺少 SCRE 指令（SCR 结束）或在 SCRE 前出现

				不允许使用的指令
			0x97	用户程序包含不带号码及带号码的 EU/ED 指令
			0x98	尝试在配备不带号码 EU/ED 指令的程序中执行运行时间编辑
			0x99	过多隐含程序段
			0x9A	在用户中断中尝试转入自由端口模式
			0x9B	非法索引（字符串操作，已指定该操作中的一个起始位置数值0）

11.1.2 PLC 致命错误

严重错误导致 PLC 停止执行程序。取决于错误的严重程度，严重错误可以使 PLC 无法执行任何一个或全部功能。处理严重错误的目的在于使 PLC 进入安全状态，使 PLC 能对现有错误条件的询问作出应答。

检测到严重错误时，PLC 执行下列任务。

- 更改为 STOP（停止）模式；
- 打开系统故障 LED 和停止 LED；
- 关闭输出；

MagicWorks PLC 在 PLC 信息对话框中显示由 PLC 生成的错误代码以及简要说明。欲打开该对话框，选择菜单命令“PLC”→“信息”。

一旦纠正了引起严重错误的条件后，必须采用下列方法之一重新启动 PLC。

- 关闭 PLC 电源后再重新打开；
- 将模式开关从 TERM（有条件）或 RUN（运行）转至 STOP（停止）位置。（只有在 PLC 出现严重错误和故障指示灯亮起时，模式开关才以此种方式工作。）

重新启动 PLC 会清除严重错误条件并开始启动诊断测试。如果发现另一个严重错误条件，PLC 再次设置系统故障 LED；否则，PLC 开始正常操作。

有几种可能的错误条件会使 PLC 无法通讯，在这种情况下，您无法检视 PLC 错误代码。此类错误表示硬件故障，要求修理 PLC 模块；改变程序或清除 PLC 内存不能解决这一问题。

PLC 一直处于该状态，直到严重错误得到纠正。下表列举可从 PLC 读取的严重错误代码说明。

表 11-3 PLC 致命错误信息（CTH300-H 系列 PLC 适用）

支持的事件类型	类型编码	编码	事件描述
CPU 致命错误	0x8	0x00	不存在严重错误
		0x01	保留
		0x02	保留
		0x03	扫描看门狗超时错误
		0x04	保留
		0x05	保留
		0x06	保留

		0x07	保留
		0x08	保留
		0x09	保留
		0x0A	保留
		0x0B	保留
		0x0C	保留
		0x0D	保留
		0x0E	保留
		0x0F	保留
		0x10	内部软件错误
		0x11	比较触点间接编址错误
		0x12	比较节点非法浮点数值错误
		0x13	保留
		0x14	比较节点范围错误

11.1.3 模块诊断事件（CTH300 系列 PLC 适用）

表 11-4 EtherCAT 寄存器 SMB400~SMB465 诊断

SMB	错误描述
SMB400	找到的 EtherCAT 从站的个数
SMB401	EtherCAT 错误： 0：没有错误 1：组态参数错误 2：没有找到从站 3：状态切换错误 4：写组态时发生错误 5：从站个数错误 6：从站不匹配
SMB402	第1个从站的状态： 0：没有连接 1：初始化状态 2：预操作状态 4：安全操作 8：操作状态 16#80：产品 ID 不匹配 16#81：厂商 ID 不匹配 16#82：SDO 写入出错 其它：错误的状态
SMB403~SMB465	从站2状态~从站64状态

表 11-5 模块诊断事件

支持的事件类型	类型编码	编码	事件描述
模块诊断事件（在特殊存储区 SMB500 ～ SMB531中显示）	0xC	0x00	模块无故障
		0x01	模块忙
		0x02	模块超时未响应
		0x03	模块类型不匹配
		0x04	模块版本不匹配
		0x05	软件错误
		0x06	模块等待标志超时
		0x07	总线应答错误
		0x08	总线 CRC 校验错误
		0x10	内存偏移量超范围
		0x11	模块没有准备好
		0x12	模块组态错误
		0x13	模块不支持本条指令
		0x15	模块内部诊断
		0x16	模块没有电源

表 11-6 特殊存储区 SMB500~SMB531 诊断功能

SMB	错误
SMB500	第1个模块(Rack0, Slot3)错误
SMB501	第2个模块(Rack0, Slot4)错误
SMB502	第3个模块(Rack0, Slot5)错误
SMB503	第4个模块(Rack0, Slot6)错误
SMB504	第5个模块(Rack0, Slot7)错误
SMB505	第6个模块(Rack0, Slot8)错误
SMB506	第7个模块(Rack0, Slot9)错误
SMB507	第8个模块(Rack0, Slot10)错误
SMB508	第9个模块(Rack1, Slot3)错误
SMB509	第10个模块(Rack1, Slot4)错误
SMB510	第11个模块(Rack1, Slot5)错误
SMB511	第12个模块(Rack1, Slot6)错误
SMB512	第13个模块(Rack1, Slot7)错误
SMB513	第14个模块(Rack1, Slot8)错误
SMB514	第15个模块(Rack1, Slot9)错误
SMB515	第16个模块(Rack1, Slot10)错误
SMB516	第17个模块(Rack2, Slot3)错误
SMB517	第18个模块(Rack2, Slot4)错误
SMB518	第19个模块(Rack2, Slot5)错误
SMB519	第20个模块(Rack2, Slot6)错误
SMB520	第21个模块(Rack2, Slot7)错误
SMB521	第22个模块(Rack2, Slot8)错误
SMB522	第23个模块(Rack2, Slot9)错误

SMB523	第24个模块(Rack2, Slot10)错误
SMB524	第25个模块(Rack3, Slot3)错误
SMB525	第26个模块(Rack3, Slot4)错误
SMB526	第27个模块(Rack3, Slot5)错误
SMB527	第28个模块(Rack3, Slot6)错误
SMB528	第29个模块(Rack3, Slot7)错误
SMB529	第30个模块(Rack3, Slot8)错误
SMB530	第31个模块(Rack3, Slot9)错误
SMB531	第32个模块(Rack3, Slot10)错误
SMB1512~SMB1549	保留
SMB1550~SMB1589	H36-000S2本机 CAN 口信息
SMB1550	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB1551	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB1552 ~SMB1581	从站2状态（按节点 ID 排列）~ 从站31状态（按节点 ID 排列）
SMB1582	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB1583	CAN 模块软件版本号
SMB1584 ~SMB1589	保留
SMB1590~SMB2047	保留

11.1.4 运行模式转换

表 11-7 运行模式转换诊断信息

支持的事件类型	类型编码	编码	事件描述
运行模式转换	0x5	0x01	上电事件
		0x02	开机事件

		0x03	停机事件

11.1.5 同步错误

表 11-8 同步错误诊断信息

支持的事件类型	类型编码	编码	事件描述
同步错误	0x6	0x00	无错误
		0x01	使用的 DB 块不存在或者使用 DB 越界

11.1.6 SOE 事件诊断

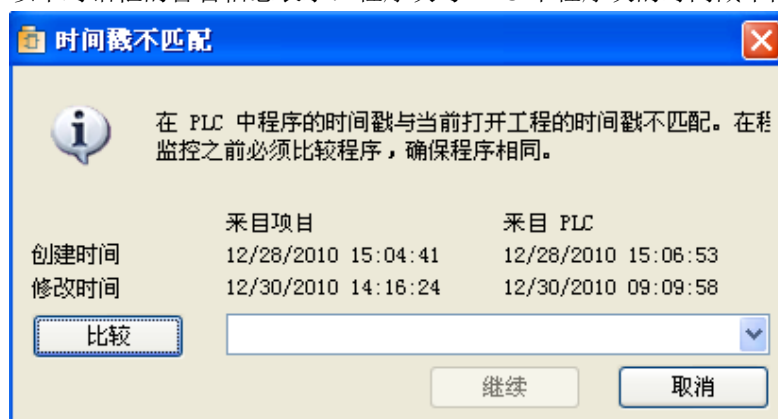
SOE 监视记录方式：每个扫描周期监视记录一次。

表 11-9 SOE 事件诊断信息

支持的事件类型	类型编码	编码	事件描述
SOE 事件	0xB	BOOL 变量状态变化事件： (1) DI、DQ 状态 ON→OFF, OFF→ON (2) BOOL 型 ON→OFF, OFF→ON	扫描周期结束监视事件，最多组态16个 SOE 事件，按顺序编码1-16，可组态过程变量（支持有符号整型，无符号整型，有符号双整型，无符号双整型，浮点型便测量固态，能组态对应事件文本并在诊断信息显示。

11.2 时间戳不匹配

以下对话框的警告信息表示：程序块与 PLC 中程序块的时间戳不同，继续操作可能有危险。



时间戳不同只是说明程序的最后修改时间不同，并不能说明程序块的内容不相同，此时您可以点击对话框上的“比较”按钮将 PLC 中的程序块上载上来进行比较，一旦比较通过，继续按钮将会使能。或者您可以退出此对话框，选择保存菜单保存项目中的程序块文件，然后回到项目管理器界面执行下载程序块操作，这样可使 PLC 中的程序块和项目中的程序块时间戳相同，从而允许您运行状态。

程序块的时间戳说明：

每个程序包含两个不同的时间戳：创建时间戳和最近修改时间戳。利用“新建”菜单建立程序

块时，系统会同时设置块的创建时间戳。用户其后的保存操作会影响最近修改时间戳而不会影响创建时间戳。

备注：加密程序块时间戳一致时可进行程序状态监控。

12 打印

MagicWorks PLC 程序创建成功后，即可使用“打印”功能输出项目文档。

本章概要

- 支持打印的项目
- 打印功能的基本操作

12.1 支持打印的项目

通过打开相关项目，支持打印的项目内容包括：

- 程序块（DB 块、OB 块、FB 块、FC 块）
- 初始化 V 内存数据块
- EasyCAN 组态
- 硬件组态
- 状态表
- 交叉引用（交叉引用、字节引用、位引用、沿使用）
- 符号表

12.2 基本操作

参考下文执行打印的基本步骤：

- 1、打开相应的对象，在屏幕上显示要打印的信息。
- 2、在应用程序窗口中，使用以下一种方法，打印程序和项目文档。
 - 点击“打印”按钮 。
 - 菜单项选择“文件”→“打印”菜单命令。
 - 按 Ctrl+P 捷径键组合。

根据当前所在的应用程序，菜单栏中出现的第一个条目可能不是“文件”，而是该应用程序处理的对象，例如“符号表”。

- 3、选择打印功能后，弹出打印对话框，用于设置一些参数并打印当前项目文件。

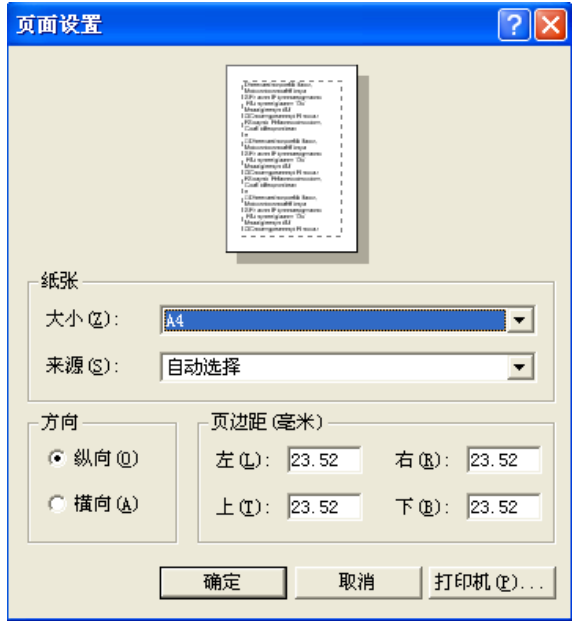


在此，您可以“选择打印机”和设置要打印的“页面范围”，“首选项”用于修改打印选项，“查找打印机”用于查找需要使用的打印机，“份数”用于选择打印的总份数。

<备注> 暂不支持选择组件打印。

页面设置

想要设定打印的格式，可以使用页面设置功能。选择“文件”→“页面设置”菜单命令，即可打开页面设置对话框，界面如下图：

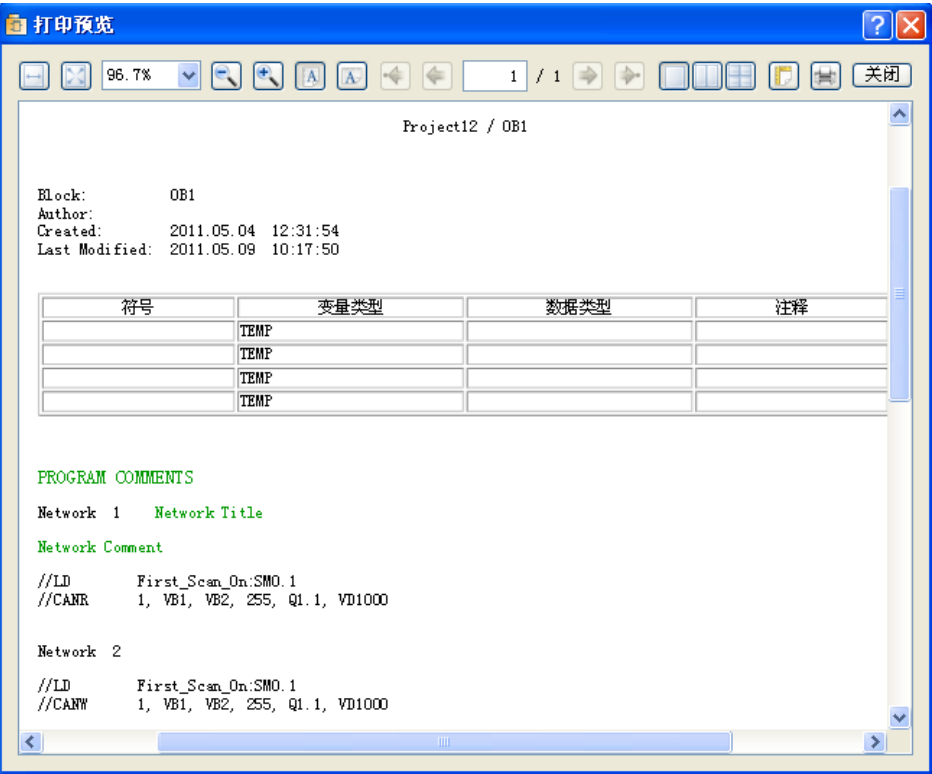


“页面设置”对话框允许您配置以下选项：

- 纸张：选择纸张尺寸和纸张来源。
- 方向：调整页面方向，可选择纵向或横向。
- 页边距：调整打印件的左、右、上、下与页边的距离。

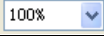
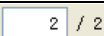
打印预览

如果想在纸张上实际打印之前先查看打印的效果，可以使用打印预览功能。选择“文件”→“打印预览”菜单命令或点击工具栏按钮，即可启动打印预览。



通过窗口上方的工具按钮，您可以实现预览页面的放大、缩小，有多页时可以翻页查看，如果效果满意，还可以点击右上方的打印按钮将预览内容打印到纸张上。

预览窗口中的各按钮功能如下：

功能	图形	说明
宽度适应		预览文档的显示宽度根据当前预览窗口的大小自动调整
页面适应		预览文档的显示大小不随预览窗口的大小调整
设置显示比例		选择文档预览时显示的比例（相对文档的实际大小）
缩小		每点击此按钮一次，预览文档显示的大小按一定比例缩小
放大		每点击此按钮一次，预览文档显示的大小按一定比例放大
拉长		预览文档的高度增加
拉宽		预览文档的宽度增加
功能	图形	说明
第一页		切换到文档的第一页预览，当文档数量只有一页时，此按钮无效
上一页		切换到当前页的上一页预览，当文档数量只有一页时，此按钮无效
转到指定页		切换到指定页预览
下一页		切换到当前页的下一页预览，当文档数量只有一页时，此按钮无效

最后一页		切换到文档的最后一页预览，当文档数量只有一页时，此按钮无效
单页显示		预览窗口中每次显示一页文档
双页显示		预览窗口中每次显示两页文档，左右排列
页面总览		预览窗口中显示所有的文档，自动排列
页面设置		点击进入页面设置对话框
打印		点击进入打印对话框
关闭		关闭预览窗口

附录

通过本附录您可以快速查阅以下内容：

- ☐ 术语解释
- ☐ 指令集
- ☐ 特殊存储器（SM）标志位
- ☐ 中断事件优先级列表
- ☐ 国际助记符

A 术语解释

表 A-1 专业术语解释

缩写	全称	中文释义
AI	Analog input	模拟量输入
AQ	Analog output	模拟量输出
CAN	Controller Area Network	控制器局域网
COB-ID	Communication Object Identifier	通信对象 ID
DB	Data Block	数据块
FB	Function Block	功能块
FBD	Function Block Diagram	功能块编程语言
FC	Function	功能
HSC	High-speed counter	高速计数器
PTO	Pulse Train Output	脉冲串输出
PDO	Process Data Object	过程数据对象
PLC	Programmable Logic Controller	可编程控制器
STL	Statement List	指令表编程语言
LAD	Ladder Logic	梯形图编程语言
IEC	International Electro Technical Commission	国际电工委员会
MES	Manufacturing Execution System	制造执行系统
NETWORK		梯形图编程中的程序单元，网络
OB	Organization Block	组织块
OEM	Original Equipment Manufacturer	原厂委托制造
OPC	OLE for Process Control	用于过程控制的 OLE
PID	Proportion Integration Differentiation	比例、积分、微分

SDO	Service Data Object	服务数据对象
SOE	Sequence Of Event	事件顺序记录
TD2X		文本显示器，显示屏为两行
TD4X		文本显示器，显示屏为四行

B 指令集

B.1 指令一览表

类别	STL	指令名称
位 逻 辑 指 令	LD	常开触点载入指令
	A	常开触点与指令
	O	常开触点或指令
	LDN	常闭触点载入指令
	AN	常闭触点与指令
	ON	常闭触点或指令
	LDI	常开触点立即指令
	AI	常开触点与立即指令
	OI	常开触点或立即指令
	LDNI	常闭触点立即指令
	ANI	常闭触点与立即指令
	ONI	常闭触点或立即指令
	NOT	逻辑栈顶取反指令
	EU	上升沿检测指令
	ED	下降沿检测指令
	ALD	与载入指令
	OLD	或载入指令
	LPS	逻辑进栈指令
	LDS	载入堆栈指令
	LRD	逻辑读取指令
	LPP	逻辑出栈指令
	=	输出指令
	=I	立即输出指令
	S	置位指令
	SI	立即置位指令
	R	重设指令
	RI	立即重设指令
	AENO	功率流与指令
	NOP	空操作指令
比	LDBx	字节比较（=,<,>,<=,>=,<>）载入指令

较 指 令	LDWx	整数比较 (=,<,>,<=,>=,<>) 载入指令
	LDDx	长整数比较 (=,<,>,<=,>=,<>) 载入指令
	LDRx	浮点数比较 (=,<,>,<=,>=,<>) 载入指令
	ABx	字节操作数比较 (=,<,>,<=,>=,<>) 与指令
	AWx	整数比较 (=,<,>,<=,>=,<>) 与指令
	ADx	双整数比较 (=,<,>,<=,>=,<>) 与指令
	ARx	浮点数比较 (=,<,>,<=,>=,<>) 与指令
	OBx	字节操作数比较 (=,<,>,<=,>=,<>) 或指令
	OWx	整数比较 (=,<,>,<=,>=,<>) 或指令
	ODx	双整数比较 (=,<,>,<=,>=,<>) 或指令
	ORx	浮点数比较 (=,<,>,<=,>=,<>) 或指令
	LDSx	字符串比较 (=,<>) 载入指令
	ASx	字符串比较 (=,<>) 与指令
	OSx	字符串比较 (=,<>) 或指令
传 送 指 令	MOVB	字节移动指令
	MOVW	字移动指令
	MOVD	双字移动指令
	MOVR	浮点数移动指令
	BMB	块移动字节指令
	BMW	块移动字指令
	BMD	块移动双字指令
	SWAP	高低字节交换指令
	BIR	移动字节立即读指令
	BIW	移动字节立即写指令
	SRCP	存储配方到存储卡
	LRCP	从存储卡装载配方
	SDL	存储数据记录到存储卡
整 数 计 算 指 令	+I	整数加法指令
	-I	整数减法指令
	*I	整数乘法指令
	/I	整数除法指令
	+D	双整数加法指令
	-D	双整数减法指令
	*D	双整数乘法指令
	/D	双整数除法指令
	MUL	整数与双整数相乘
	DIV	整数与双整数相除
类别	STL	指令名称
整 数	INCB	字节递增指令
	DECB	字节递减指令

	计 算 指 令	INCW	整数递增指令
		DECW	整数递减指令
		INCD	长整数递增指令
		DECD	长整数递减指令
	浮 点 数 运 算 指 令	+R	实数加法指令
		-R	实数加法指令
		*R	实数乘法指令
		/R	实数除法指令
		SQRT	平方根指令
		SIN	正弦运算指令
		COS	余弦运算指令
		TAN	正切运算指令
		LN	自然对数运算指令
		EXP	自然指数运算指令
		PID	PID 环路运算指令
	转 换 指 令	BTI	字节至整数转换
		ITB	整数至字节转换
		ITD	整数至双整数转换
		ITS	整数转换为字符串
		DTI	双整数至整数转换
		DTR	双整数至实数转换
		DTS	双整数至字符串
		ROUND	进位取整指令
		TRUNC	截位取整指令
		RTS	实数至字符串转换
		BCDI	BCD 至整数转换
		IBCD	整数至 BCD 转换
		ITA	整数至 ASCII 转换
		DTA	双整数至 ASCII 转换
		RTA	实数至 ASCII 码转换指令
		ATH	ASCII 至16进制转换
		HTA	16进制至 ASCII 转换
		STI	字符串至整数转换
		STD	字符串至双整数转换
		STR	字符串至实数转换
		DECO	解码指令
		ENCO	编码指令
		SEG	段指令
	实 时 时	TODR	读取实时时钟
		TODW	设置实时时钟
		TODRX	读取扩展的实时时钟

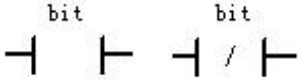
	钟	TODWX	设置扩展的实时时钟
	移位循环指令	SLB	向左移位字节
		SLW	向左移位字
		SLD	向左移位双字
		SRB	向右移位字节
		SRW	向右移位字
		SRD	向右移位双字
		RLB	向左旋转字节
		RLW	向左旋转字
		RLD	向左旋转双字
		RRB	向右旋转字节
		RRW	向右旋转字
		RRD	向右旋转双字
		SHRB	移位寄存器位指令
	逻辑计算指令	INVB	反转字节指令
		INWV	反转字指令
		INVD	反转双字指令
		ANDB	与字节指令
		ANDW	与字指令
		ANDD	与双字指令
		ORB	或字节指令
		ORW	或字指令
		ORD	或双字指令
		XORB	异或字节指令
		XORW	异或字指令
		XORD	异或双字指令
	计数器指令	CTU	向上计数指令
		CTD	向下计数指令
		CTUD	向上/向下计数指令
		HDEF	高速计数器定义
		HSC	高速计数器指令
		PLS	脉冲输出指令
		PTO	脉冲输出指令
	定时器指令	TON	打开延时计时器
		TONR	保留性打开延时计时器
		TOF	关闭延时计时器
		BITIM	读取开始时间计时器
		R_UTIM	读取当前微秒值指令
		CITIM	计算间隔时间计时器
	字	SLEN	取字符串长度指令

	符 串 指 令	SCPY	复制字符串指令
		SSCPY	从字符串复制子字符串指令
		SCAT	并置字符串指令
		SFND	在字符串中查找字符串指令
		CFND	在字符串中查找一个字符指令
	表 指 令	FILL	内存填充指令
		ATT	增加至表格指令
		FNDx	表格查找 (=,<,>,<=, >=,<>) 指令
		FIFO	先入先出指令
		LIFO	后入先出指令
	中 断 指 令	CRETI	从中断程序有条件返回指令
		ENI	启用中断指令
		DISI	禁用中断指令
		ATCH	附加中断指令
		DTCH	分离中断指令
		CEVNT	清除中断事件指令
	程 序 控 制 指 令	FOR	FOR-NEXT 循环开始
		NEXT	FOR-NEXT 循环结束
		JMP	跳转至标签指令
		LBL	标签指令
		WDR	监视器重设指令
		DLED	诊断 LED 指令
		CALL	调用子例行程序指令
		LSCR	载入顺序控制中继
		SCRE	顺序控制中继结束
		SCRT	顺序控制中继转换
		CSCRE	有条件顺序控制中继结束
		CRET	从子程序有条件返回指令
		END	有条件结束
		STOP	停机指令
	通 信 指 令	CANW	CANopen SDO 写指令
		CANR	CANopen SDO 读指令
		XMT	自由口传输指令
		RCV	自由口接收指令
		NETR	网络读取指令
		NETW	网络写入指令
		UDPNETR	以太网网络读功能
		UDPNETW	以太网网络写入功能
		MBSNDMSG	发送 Modbus 消息
		SPA	设置端口指令
		GPA	获得端口地址指令

	EBUSR	读取模块信息指令
	EBUSW	写入模块信息指令
	EBUSGETDIA	获取模块内部诊断信息指令
	EBUSSNDCMD	发送模块操作命令指令

B.2 指令详解

位逻辑指令

标准触点指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： LD bit A bit O bit LDN bit AN bit ON bit		
梯形图： 		
输入输出参数	数据类型	适用操作数
位	BOOL	I, Q, M, SM, T, C, V, S, L

操作数说明：如果参数 bit 的数据类型为 I 或 Q，标准触点指令从内存或过程映象寄存器读取引用值。

功能说明：

当参数位变量 bit 为 1 时，常开触点（LD、A、O）接通。

当参数位变量 bit 为 0 时，常闭触点（LDN、AN、ON）接通。

在 STL 中，常开触点用载入（LD）、与（A）、或（O）指令表示，常开指令将地址位数值置于栈顶。

常闭触点由负装载（LDN）、与非（AN）、或非（ON）指令表示，常闭指令将地址位数值的逻辑非置于栈顶。

在 LAD 中，常开和常闭指令用触点表示。

示例程序：

STL:

ORGANIZATION_BLOCK 主程序: OB1

Network 1

LD I0.0
A I0.1
O I0.2
= Q0.0 //如果 I0.0 与 I0.1 同时为 1，或者 I0.2 为 1，则 Q0.0 点亮

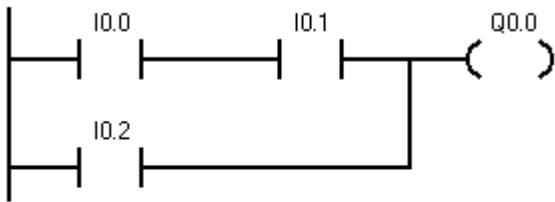
Network 2

LDN I1.0
AN I1.1
ON I1.2
= Q1.0 //如果 I1.0 与 I1.1 同时为 0，或者 I1.2 为 0，则 Q1.0 点亮

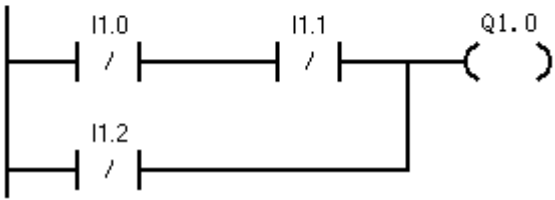
梯形图：

ORGANIZATION_BLOCK 主程序：OB1

Network 1



Network 2



立即触点指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： LDI bit AI bit OI bit LDNI bit ANI bit ONI bit		
梯形图： 		
输入输出参数	数据类型	适用操作数
位	BOOL	I

功能说明：

当实际输入点为 1 时，立即常开触点（LDI、AI、OI）接通。

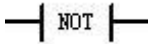
当实际输入点为 0 时，立即常闭触点（LDNI、ANI、ONI）接通。

立即指令执行时获取实际输入值，不更新 CPU 进程映象寄存器。立即触点的参数值会立即更新，而不依赖于 CPU 的扫描周期。

在 STL 中，立即常开触点用立即载入（LDI）、立即与（AI）和立即或（OI）指令表示，立即常开指令立即将实际输入值装载、与、或至栈顶。

立即常闭触点用立即载入非（LDNI）、立即与非（ANI）和立即或非（ONI）指令表示，立即常闭指令立即将实际输入点数值的逻辑非装载、与、或至栈顶。

在 LAD 中，立即常开和立即常闭指令用触点表示。

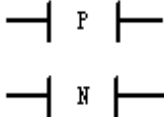
取反指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： NOT		
梯形图： 		
输入输出参数	数据类型	适用操作数
无	无	无

功能说明：

取反（NOT）触点将输入的使能位状态取反。当使能位为 1，取反（NOT）触点断开，当使能位为 0，取反（NOT）触点接通。

在 STL 中，取反（NOT）指令将堆栈顶部的数值从 0 改变为 1，或从 1 改变为 0。

在 LAD 中，取反（NOT）指令用触点表示。

边沿检测指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： EU ED		
梯形图： 		
输入输出参数	数据类型	适用操作数
无	无	无

功能说明：

上升沿检测（EU）指令比较本次扫描与上次扫描的输入能流的变化，如能流有上升沿变化时

(OFF→ON)，将能流置为 ON 状态一个扫描间隔；其他情况下将能流置为 OFF 状态。

下降沿检测 (ED) 指令比较本次扫描与上次扫描的输入能流的变化，如能流有下降沿变化时 (ON→OFF)，将能流置为 ON 状态一个扫描间隔；其他情况下将能流置为 OFF 状态。

在 STL 中，上升沿检测指令一旦在堆栈顶部数值中检测到 0 至 1 转换时，则将堆栈顶值设为 1；否则，将其设为 0。

下降沿检测指令一旦在堆栈顶部数值中检测到 1 至 0 转换时，则将堆栈顶值设为 1；否则，将其设为 0。

对于运行时编辑操作，您必须为边沿检测指令输入一个参数。有关详情，请参阅在运行模式中进行程序编辑。

在 LAD 中，边沿检测指令用触点表示。



提示

因为边沿检测指令要求执行“打开至关闭”或“关闭至打开”转换，所以无法在首次扫描时检测上升沿或下降沿。在首次扫描中，CTSC-200 设置由这些指令指定的位状态。在其后的扫描中，这些指令才能检测到指定位的转换。

逻辑堆栈指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表：		
ALD		
OLD		
LPS		
LDS n		
LRD		
LPP		
梯形图： 梯形图无此指令		
输入输出参数	数据类型	适用操作数
LDS 指令操作数 n	无	常数 (0~8)

功能说明：

与载入指令 (ALD) 采用逻辑与操作将堆栈第一级和第二级中的数值组合，并将结果载入堆栈顶部。执行 ALD 后，堆栈深度减 1。

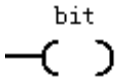
或载入指令 (OLD) 采用逻辑或操作将堆栈第一级和第二级中的数值组合，并将结果载入堆栈顶部。执行 OLD 后，堆栈深度减 1。

逻辑进栈指令 (LPS) 复制堆栈中的顶值并使该数值进栈。堆栈底值被推出栈并丢失。

逻辑出栈指令 (LPP) 将堆栈中的一个数值出栈。第二个堆栈数值成为堆栈新顶值。

逻辑读取指令 (LRD) 将第二个堆栈数值复制至堆栈顶部。不执行进栈或出栈，但旧堆栈顶值被复制破坏。

载入堆栈指令 (LDS) 复制堆栈中的堆栈位 n，并将该数值置于堆栈顶部。堆栈底值被推出栈并丢失。

输出指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： = bit		
梯形图： 		
输入输出参数	数据类型	适用操作数
输出位 bit	BOOL	I, Q, M, SM, T, C, V, S, L
使能位 (LAD)	BOOL	使能位

功能说明：

输出指令 (=) 将输出位的新数值写入过程映像寄存器。

在 STL，位于堆栈顶端的数值被复制至指定的位。

在 LAD 中，当输出指令被执行时，CPU 将过程映像寄存器中的输出位打开或关闭，指定的位被设为等于使能位。

立即输出指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： =I bit		
梯形图： 		
输入输出参数	数据类型	适用操作数
输出位 bit	BOOL	Q
使能位 (LAD)	BOOL	使能位

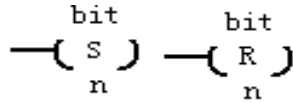
功能说明：

执行指令时，立即输出指令 (=I) 将新值写入实际输出和对应的过程映像寄存器位置。执行“立即输出”指令时，实际输出点的值被立即设为等于使能位。“I”表示立即输出；执行指令时，新值被写入实际输出和对应的过程映像寄存器位置。这与非立即输出不同，非立即输出仅将新值写入过程映像寄存器而不马上更新实际输出。

对于 STL，立即输出指令将位于堆栈顶端的数值复制至指定的实际输出位 bit。

置位复位指令	设置 ENO = 0 的错误条件	0006（间接地址）、0091（操作数超出范围）
	影响的特殊内存位	无
指令列表： S bit, n R bit, n		

梯形图:



输入输出参数	数据类型	适用操作数
位 bit	BOOL	I, Q, M, SM, T, C, V, S, L
位数 n	字节	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *AC, *LD

功能说明:

当能流有效时，置位打开或复原从指定地址 **bit** 开始的 **n** 个点，最多可以置位和复位 255 个点。

如果“复位”指令指定一个定时器位（T）或计数器位（C），指令将复位定时器或计数器位，并清除定时器或计数器的当前值。

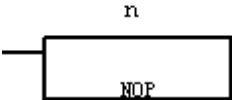
立即置位复位指令	设置 ENO = 0 的错误条件	0006（间接地址）、0091（操作数超出范围）
	影响的特殊内存位	无
指令列表: SI bit, n RI bit, n		
梯形图:		
输入输出参数	数据类型	适用操作数
位 bit	BOOL	Q
位数 n	字节	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *AC, *LD

功能说明: 能流有效时，立即置位指令（SI）和立即复位指令（RI）将立即置位打开或复位清零从指定地址 **bit** 开始的 **n** 个点，可以置位或复位 1 至 128 个点。执行立即置位复位指令时，新值被写入实际输出点和相应的过程映像寄存器位置。这与非立即指令不同，非立即指令只将新值写入过程映像寄存器而不会在扫描周期中立即更新物理输出点。

功率流与指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表: AENO		
梯形图: 梯形图无此指令		
输入输出参数	数据类型	适用操作数
无	无	无

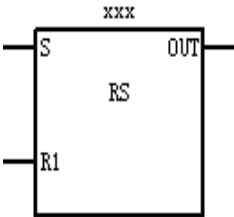
功能说明: AENO 是 ENO 位的逻辑与运算，运算的结果放入栈顶。ENO 是 LAD 中指令盒的布尔输出。如果指令盒在 EN 输入位置被使能而且执行时无错，则 ENO 输出向下一个指令盒传递能流。ENO 位用于堆栈顶部，影响其后指令执行的使能位。STL 指令没有 EN 输入；堆栈顶值必须为逻辑 1，有条件指令才能执行，STL 中也没有 ENO 输出，但采用 AENO 指令可存取 ENO

输出位。AENO 可用于生成与方框 ENO 位相同的效果。

空操作指令	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： NOP n		
梯形图： 		
输入输出参数	数据类型	适用操作数
n	字节	常数（0~255）

功能说明：

空操作指令（NOP）对用户程序执行空操作，可以用在用户程序需要短暂延时的地方。

复位主双稳态触发器	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： STL 中不存在此指令		
梯形图： 		
输入输出参数	数据类型	适用操作数
S	BOOL	Power Flow
R1	BOOL	Power Flow
xxx	BOOL	I, Q, M, V, S
OUT	BOOL	Power Flow

功能说明：复原主双稳态触发器（RS）是一种复原主要位的锁存器。如果设置（S）和复原（R）信号均为真实，则输出（OUT）为虚假。“位”参数指定被设置或复原的布尔参数。供选用输出反映位参数的信号状态。

设置主双稳态触发器	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： STL 中不存在此指令		
梯形图：		

xxx S1 OUT SR R		
输入输出参数	数据类型	适用操作数
S1	BOOL	Power Flow
R	BOOL	Power Flow
xxx	BOOL	I, Q, M, V, S
OUT	BOOL	Power Flow

功能说明：设置主双稳态触发器（SR）是一种设置主要位的锁存器。如果设置（S1）和复原（R）信号均为真实，则输出（OUT）为真实。“位”参数指定被设置或复原的布尔参数。供选用输出反映位参数的信号状态。

比较指令

字节比较指令	设置 ENO = 0 的错误条件	非法间接地址
	影响的特殊内存位	无
指令列表： LDB= IN1, IN2 AB= IN1, IN2 OB= IN1, IN2 LDB<> IN1, IN2 AB<> IN1, IN2 OB<> IN1, IN2 LDB>= IN1, IN2 AB>= IN1, IN2 OB>= IN1, IN2 LDB<= IN1, IN2 AB<= IN1, IN2 OB<= IN1, IN2 LDB> IN1, IN2 AB> IN1, IN2 OB> IN1, IN2 LDB< IN1, IN2 AB< IN1, IN2 OB< IN1, IN2		
梯形图：		

IN1 ==B IN2 IN1 >=B IN2 IN1 >B IN2 IN1 <>B IN2 IN1 <=B IN2 IN1 <B IN2

功能说明：

无论使能位状态如何，"比较"指令均会执行。字节比较指令用于比较两个字节型的值，字节比较不带符号。

在 STL 中，比较结果为真时，栈顶为 1。

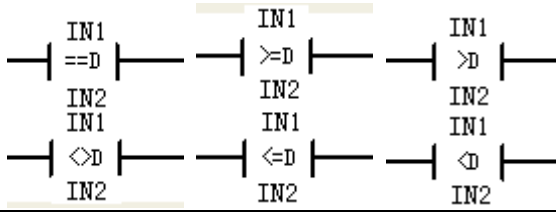
在 LAD 中，比较结果为真时，触点打开。

<注意> 执行比较指令时如出现以下错误为致命错误，会使 CTSC-200 立即停止执行程序：

- 非法间接地址
- 比较实数指令中存在非法实数（例如，NAN）

为防止出现以上错误，请确保在使用比较指令前指针已正确初始化并且参数中不存在非法实数。

双整数比较指令	设置 ENO = 0 的错误条件	非法间接地址
	影响的特殊内存位	无
指令列表：		
LDD= IN1, IN2		
AD= IN1, IN2		
OD= IN1, IN2		
LDD<> IN1, IN2		
AD<> IN1, IN2		
OD<> IN1, IN2		
LDD>= IN1, IN2		
AD>= IN1, IN2		
OD>= IN1, IN2		
LDD<= IN1, IN2		
AD<= IN1, IN2		
OD<= IN1, IN2		
LDD> IN1, IN2		
AD> IN1, IN2		
OD> IN1, IN2		
LDD< IN1, IN2		
AD< IN1, IN2		

OD< IN1, IN2		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	双整数	ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *LD, *AC
IN2	双整数	ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *LD, *AC

功能说明：

无论使能位状态如何，“比较”指令均会执行。双整数比较指令用于比较两个双整型的值，双整数比较带符号（如 16#7FFFFFFF > 16#80000000，因为 16#80000000 为负）。

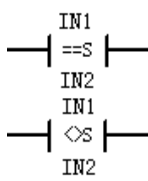
在 STL 中，比较结果为真时，栈顶为 1。

在 LAD 中，比较结果为真时，触点打开。

<注意> 执行比较指令时如出现以下错误为致命错误，会使 CTSC-200 立即停止执行程序：

- 非法间接地址
- 比较实数指令中存在非法实数（例如，NAN）

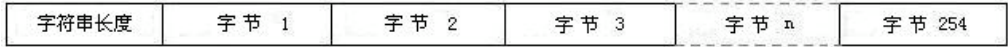
为防止出现以上错误，请确保在使用比较指令前指针已正确初始化并且参数中不存在非法实数。

字符串比较指令	设置 ENO = 0 的错误条件	非法间接地址 遇到长度超过254个字符的字符串 字符串的起始地址和长度无法放入一个指定的内存区
	影响的特殊内存位	无
指令列表： LDS= IN1, IN2 AS= IN1, IN2 OS= IN1, IN2 LDS<> IN1, IN2 AS<> IN1, IN2 OS<> IN1, IN2		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
IN2	字符串	VB, LB, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式:

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节):



功能说明:

无论使能位状态如何，“比较”指令均会执行。字符串比较指令用于比较两个字符串是否相同。

在 STL 中，比较结果为真时，栈顶值为 1。

在 LAD 中，比较结果为真时，触点打开。

<注意> 执行比较指令时如出现以下错误为致命错误，会使 CTSC-200 立即停止执行程序：

- 非法间接地址
- 长度超过 255 个字符的字符串
- 字符串的起始地址和长度无法放入一个指定的内存区

为了防止出现以上错误，请确保在使用比较指令前指针和存放 ASCII 字符串的内存位置已正确初始化并且为 ASCII 字符串保留的缓冲区可完全放置在指定的内存区中。

整数比较指令	设置 ENO = 0 的错误条件	非法间接地址
	影响的特殊内存位	无
指令列表： LDW= IN1, IN2 AW= IN1, IN2 OW= IN1, IN2 LDW<> IN1, IN2 AW<> IN1, IN2 OW<> IN1, IN2 LDW>= IN1, IN2 AW>= IN1, IN2 OW>= IN1, IN2 LDW<= IN1, IN2 AW<= IN1, IN2 OW<= IN1, IN2 LDW> IN1, IN2 AW> IN1, IN2 OW> IN1, IN2 LDW< IN1, IN2		

AW< IN1, IN2 OW< IN1, IN2		
梯形图：		
输入输出参数	数据类型	适用操作数
IN1	整数	IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, 常数, *VD, *LD, *AC
IN2	整数	IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, 常数, *VD, *LD, *AC

功能说明：

无论使能位状态如何，"比较"指令均会执行。整数比较指令用于比较两个整数型的值，整数比较带符号（如 16#7FFF > 16#8000，因为 16#8000 为负）。

在 STL 中，比较结果为真时，栈顶为 1。

在 LAD 中，比较结果为真时，触点打开。

<注意> 执行比较指令时如出现以下错误为致命错误，会使 CTSC-200 立即停止执行程序：

- 非法间接地址
- 比较实数指令中存在非法实数（例如，NAN）

为防止出现以上错误，请确保在使用比较指令前指针已正确初始化并且参数中不存在非法实数。

实数比较指令	设置 ENO = 0 的错误条件	非法间接地址 非法实数
	影响的特殊内存位	无
指令列表： LDR= IN1, IN2 AR= IN1, IN2 OR= IN1, IN2 LDR<> IN1, IN2 AR<> IN1, IN2 OR<> IN1, IN2 LDR>= IN1, IN2 AR>= IN1, IN2 OR>= IN1, IN2 LDR<= IN1, IN2 AR<= IN1, IN2 OR<= IN1, IN2 LDR> IN1, IN2 AR> IN1, IN2 OR> IN1, IN2		

LDR< IN1, IN2 AR< IN1, IN2 OR< IN1, IN2		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	实数	ID, QD, MD, SD, SMD, VD, LD, AC, 常数, *VD, *LD, *AC
IN2	实数	ID, QD, MD, SD, SMD, VD, LD, AC, 常数, *VD, *LD, *AC

功能说明：

无论使能位状态如何，"比较"指令均会执行。实数比较指令用于比较两个实数的值，实数比较带符号。

在 STL 中，比较结果为真时，栈顶值为 1。

在 LAD 中，比较结果为真时，触点打开。

<注意> 执行比较指令时如出现以下错误为致命错误，会使 CTSC-200 立即停止执行程序：

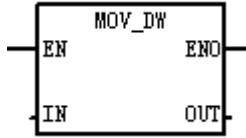
- 非法间接地址
- 比较实数指令中存在非法实数（例如，NAN）

为防止出现以上错误，请确保在使用比较指令前指针已正确初始化并且参数中不存在非法实数。

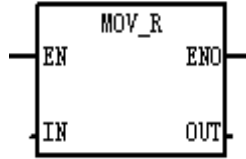
传送指令

字节赋值指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： MOVB IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

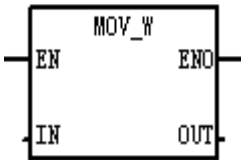
功能说明： 字节赋值指令（MOVB）将输入字节 IN 的值拷贝赋给输出字节型变量 OUT。

双字赋值指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表: MOVD IN OUT		
梯形图: 		
输入输出参数	数据类型	适用操作数
IN	双字	VD, ID, QD, MD, SD, SMD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW AC, 常数, *VD, *LD, *AC
OUT	双字	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

功能说明: 双字赋值指令 (MOVD) 将输入双字 IN 的值拷贝赋给输出双字型变量 OUT。

浮点数赋值指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表: MOVR IN OUT		
梯形图: 		
输入输出参数	数据类型	适用操作数
IN	浮点数	VD, ID, QD, MD, SD, SMD, LD, AC, 常数, *VD, *LD, *AC
OUT	浮点数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

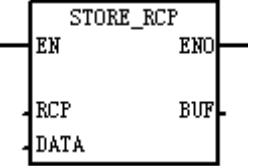
功能说明: 浮点数赋值指令 (MOVR) 将输入实数 IN 的值拷贝赋给输出浮点型变量 OUT。

字赋值指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表: MOVW IN, OUT		
梯形图: 		
输入输出参数	数据类型	适用操作数
IN	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, 常数, AC, *VD, *AC, *LD
OUT	字	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD

功能说明：字赋值指令（MOVW）将输入字 IN 的值拷贝赋给输出字型变量 OUT。

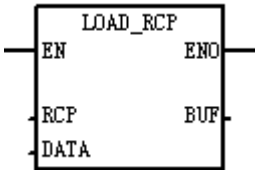
存储数据记录到 存储卡	设置 ENO = 0的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： SDL LOG, BUF		
梯形图： 		
输入输出参数	数据类型	适用操作数
LOG	字节	常数 0~3
BUF	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

功能说明：将当前项目中的数据记录 LOG 拷贝到存储卡中。

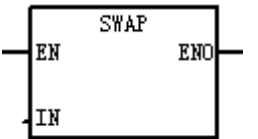
存储配方到存储卡	设置 ENO = 0的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： SRCP RCP, DATA, BUF		
梯形图： 		
输入输出参数	数据类型	适用操作数
RCP	字节	常数 0~3
DATA	字	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD
BUF	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

功能说明：将当前项目中的配方 RCP 拷贝到存储卡中。

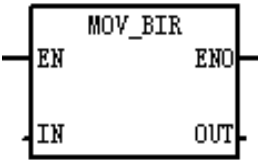
从存储卡装载配方	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： LRCP RCP, DATA, BUF		
梯形图：		

		
输入输出参数	数据类型	适用操作数
RCP	字节	常数 0~3
DATA	字	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD
BUF	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

功能说明：将存储卡上的配方 RCP 拷贝到以 BUF 指定的地址为起始地址的内存中。

字节交换指令	设置 ENO = 0的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： SWAP IN		
梯形图：		
		
输入输出参数	数据类型	适用操作数
IN	字	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD

功能说明：字节交换指令（SWAP）交换输入字 IN 的高位字节和低位字节。

赋值字节立即读取指令	设置 ENO = 0的错误条件	0006 间接地址 无法存取扩充模块（仅限立即指令）
	影响的特殊内存位	无
指令列表： BIR IN, OUT		
梯形图：		
		
输入输出参数	数据类型	适用操作数
IN	字节	IB, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

功能说明：赋值字节立即读取指令（BIR）读取物理端口输入字节 IN 的值，并将结果拷贝赋给 OUT 指定的地址中，不更新进程映像寄存器。

赋值字节立即写入	设置 ENO = 0的错	0006 间接地址

指令	误条件	无法存取扩充模块（仅限立即指令）
	影响的特殊内存位	无
指令列表： BIW IN, OUT		
梯形图：		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD
OUT	字节	QB, *VD, *LD, *AC

功能说明：赋值字节立即写入指令（BIW）从读取输入字节 IN 的值，并将值立即写入指定的物理端口 OUT 和对应的进程映像寄存器。

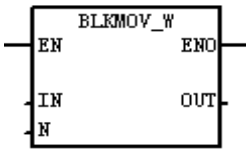
成块赋值字节指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数超出范围
	影响的特殊内存位	无
指令列表： BMB IN, OUT, N		
梯形图：		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD
N	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD

功能说明：成块赋值字节指令（BMB）将从起始地址 IN 开始的 N 个字节的值拷贝赋给 从输出地址 OUT 开始的 N 个字节，N 的范围为 1 至 255。

成块赋值双字指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数超出范围
	影响的特殊内存位	无
指令列表： BMD IN, OUT, N		
梯形图：		
输入输出参数	数据类型	适用操作数
IN	双字	VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD

OUT	双字	VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD
N	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD

功能说明：成块赋值双字指令（BMD）将从起始地址 IN 开始的 N 个双字的值拷贝赋给 从输出地址 OUT 开始的 N 个双字，N 的范围为 1 至 255。

成块赋值字指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数超出范围
	影响的特殊内存位	无
指令列表： BMW IN, OUT, N		
梯形图：		
		
输入输出参数	数据类型	适用操作数
IN	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, *VD, *LD, *AC
OUT	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC
N	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD

功能说明：成块赋值字指令（BMW）将从起始地址 IN 开始的 N 个字的值拷贝赋给 从输出地址 OUT 开始的 N 个字，N 的范围为 1 至 255。

整数计算指令

字节自加自减指令	设置 ENO = 0的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出
指令列表： INCB OUT DECB OUT		
梯形图： INC_B EN ENO IN OUT DEC_B EN ENO IN OUT		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC

OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

功能说明：

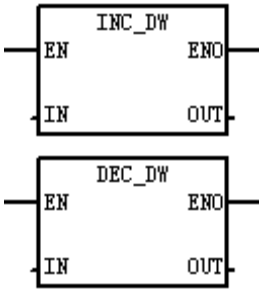
字节自加指令（INCB）在输入字节 IN 上加 1，并将加 1 的结果放入 OUT 指定的地址中。

字节自减指令（DECB）在输入字节 IN 上减 1，并将减 1 的结果放入 OUT 指定的地址中。

字节自加和自减运算不带符号。

在 LAD 中： IN + 1 = OUT
 IN - 1 = OUT

在 STL 中： OUT + 1 = OUT
 OUT - 1 = OUT

双整数自加自减指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果
指令列表： INCD OUT DECD OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	双整数	VD, ID, QD, MD, SD, SMD, LD, AC, HC, 常数, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

功能说明：

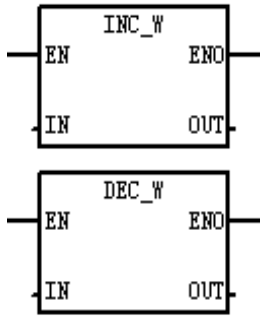
双整数自加指令（INCD）在输入双字 IN 上加 1，并将加 1 的结果放入 OUT 指定的地址中。

双整数自减指令（DECD）在输入双字 IN 上减 1，并将减 1 的结果放入 OUT 指定的地址中。

双整数自加和自减运算带符号（16#7FFFFFFF > 16#80000000）。

在 LAD 中： IN + 1 = OUT
 IN - 1 = OUT

在 STL 中： OUT + 1 = OUT
 OUT - 1 = OUT

整数自加自减指令	设置 ENO = 0的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果
指令列表： INCW OUT DECW OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	整数	VW, IW, QW, MW, SW, SMW, AC, AIW, LW, T, C, 常数, *VD, *LD, *AC
OUT	整数	VW, IW, QW, MW, SW, SMW, LW, AC, T, C, *VD, *LD, *AC 整数

功能说明：

整数自加指令（INCW）在输入字 IN 上加 1，并将加 1 的结果放入 OUT 指定的地址中。

整数自减指令（DECW）在输入字 IN 上减 1，并将减 1 的结果放入 OUT 指定的地址中。

整数自加和自减运算带符号（16#7FFF > 16#8000）。

在 LAD 中：IN + 1 = OUT

IN - 1 = OUT

在 STL 中：OUT + 1 = OUT

OUT - 1 = OUT

整数与双整数的乘除指令	设置 ENO = 0的错误条件	0006 间接地址 SM1.1 溢出 SM1.3 除数为0
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果 SM1.3 除数为0
指令列表： MUL IN1 OUT		

DIV IN1 OUT		
梯形图： MUL EN ENO IN1 IN2 OUT DIV EN ENO IN1 IN2 OUT		
输入输出参数	数据类型	适用操作数
IN1	整数	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, 常数, *VD, *LD, *AC
IN2	整数	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, 常数, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

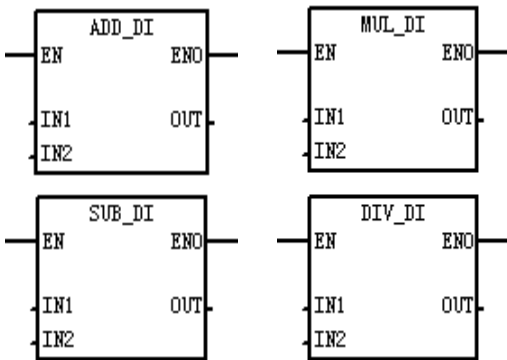
功能说明：

在 STL 乘法指令中，32 位 OUT 的低位字（16 位）被用作乘数之一。在 STL 除法指令中，32 位 OUT 的低位字（16 位）被用作除数。特殊内存（SM）位指示错误和非法数值。如果在除法运算过程中设置 SM1.3（除以零），则其他数学状态位被保留不变。否则，在完成数学运算时，所有受支持的数学状态位均包含有效状态。

在 LAD 中：IN1 * IN2 = OUT
 IN1 / IN2 = OUT

在 STL 中：IN1 * OUT = OUT
 OUT / IN1 = OUT

双整数加减乘除指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出 SM1.3 除数为0(/I)
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果 SM1.3 除数为0(/I)
指令列表： +D IN1 OUT -D IN1 OUT *D IN1 OUT /D IN1 OUT		
梯形图：		

		
输入输出参数	数据类型	适用操作数
IN1	双整数	VD, ID, QD, MD, SMD, SD, LD, AC, HC, 常数, *VD, *LD, *AC
IN2	双整数	VD, ID, QD, MD, SMD, SD, LD, AC, HC, 常数, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

功能说明：

双整数加法运算指令（+D）将两个 32 位整数相加，并将计算结果（32 位整数）放入 OUT 指定的地址中。如果结果超出 32 位长度表示范围，则设置溢出位。

双整数减法运算指令（-D）将两个 32 位整数相减，并将计算结果（32 位整数）放入 OUT 指定的地址中。如果结果超出 32 位长度表示范围，则设置溢出位。

双整数乘法运算指令（*D）将两个 32 位整数相乘，并将乘积结果（32 位整数）放入 OUT 指定的地址中。如果结果超出 32 位长度表示范围，则设置溢出位。

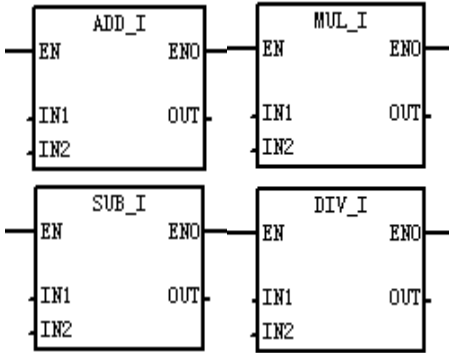
双整数除法运算指令（/D）将两个 32 位整数相除，并将商（32 位整数）放入 OUT 指定的地址中，不保留余数。如果结果超出 32 位长度表示范围，则设置溢出位。

在 LAD 中：IN1 + IN2 = OUT
IN1 - IN2 = OUT
IN1 * IN2 = OUT
IN1 / IN2 = OUT

在 STL 中：IN1 + OUT = OUT
OUT - IN1 = OUT
IN1 * OUT = OUT
OUT / IN1 = OUT

SM1.1 指示溢出错误和非法数值。如果设置 SM1.1，则 SM1.0 和 SM1.2 状态无效，且原来的输入操作数不改动。如果未设置 SM1.3，则数学操作完成，并产生有效的结果，且 SM1.0 和 SM1.2 包含有效状态。如果在除法运算过程中设置 SM1.3，则其他数学状态位将保持不变。

整数加减乘除指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出 SM1.3 除数为0(I)
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果

		SM1.3 除数为0(/I)
指令列表： +I IN1 OUT -I IN1 OUT *I IN1 OUT /I IN1 OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	整数	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, 常数, *VD, *LD, *AC
IN2	整数	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, 常数, *VD, *LD, *AC
OUT	整数	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *LD, *AC

功能说明：

整数加法运算指令（+I）将两个 16 位整数相加，并将计算结果（16 位整数）放入 OUT 指定的地址中。如果结果超出 16 位长度表示范围，则设置溢出位。

整数减法运算指令（-I）将两个 16 位整数相减，并将计算结果（16 位整数）放入 OUT 指定的地址中。如果结果超出 16 位长度表示范围，则设置溢出位。

整数乘法运算指令（*I）将两个 16 位整数相乘，并将乘积结果（16 位整数）放入 OUT 指定的地址中。如果结果超出 16 位长度表示范围，则设置溢出位。

整数除法运算指令（/I）将两个 16 位整数相除，并将商（16 位整数）放入 OUT 指定的地址中，不保留余数。如果结果超出 16 位长度表示范围，则设置溢出位。

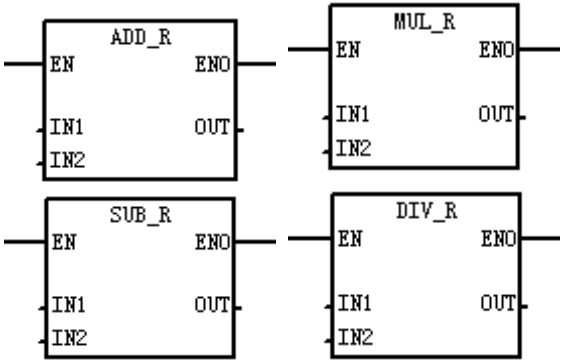
在 LAD 中：IN1 + IN2 = OUT
IN1 - IN2 = OUT
IN1 * IN2 = OUT
IN1 / IN2 = OUT

在 STL 中：IN1 + OUT = OUT
OUT - IN1 = OUT
IN1 * OUT = OUT
OUT / IN1 = OUT

SM1.1 指示溢出错误和非法数值。如果设置 SM1.1，则 SM1.0 和 SM1.2 状态无效，且原来的

输入操作数不改动。如果未设置 **SM1.3**，则数学操作完成，并产生有效的结果，且 **SM1.0** 和 **SM1.2** 包含有效状态。如果在除法运算过程中设置 **SM1.3**，则其他数学状态位将保持不变。

浮点数运算

浮点数加减乘除指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出 SM1.3 除数为0(/R)
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果 SM1.3 除数为0(/R)
指令列表： +R IN1 OUT -R IN1 OUT *R IN1 OUT /R IN1 OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *LD, *AC
IN2	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *LD, *AC
OUT	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

功能说明：

浮点数加法指令（+R）将两个 32 位实数相加，并将计算结果（32 位实数）放入 OUT 指定的地址中。

浮点数减法指令（-R）将两个 32 位实数相减，并将计算结果（32 位实数）放入 OUT 指定的地址中。

浮点数乘法指令（*R）将两个 32 位实数相乘，并将乘积（32 位实数）放入 OUT 指定的地址中。

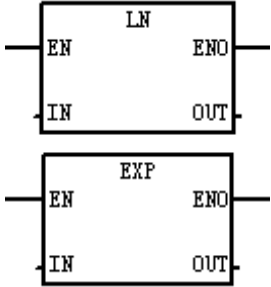
浮点数除法指令（/R）将两个 32 位实数相除，并将商（32 位实数）放入 OUT 指定的地址中。

在 LAD 中：
IN1 + IN2 = OUT
IN1 - IN2 = OUT

$IN1 * IN2 = OUT$
 $IN1 / IN2 = OUT$

在 STL 中： $IN1 + OUT = OUT$
 $OUT - IN1 = OUT$
 $IN1 * OUT = OUT$
 $OUT / IN1 = OUT$

SM1.1 指示溢出错误和非法数值。如果在除法运算中设置 **SM1.3**，则其他数学状态位不变，而且原始输入操作数也不变。**SM1.1** 用于表示溢出错误和非法数值。如果设置 **SM1.1**，则 **SM1.0** 和 **SM1.2** 状态无效，原始输入操作数不变。如果在除法运算过程中未设置 **SM1.1** 和 **SM1.3**，则说明数学运算已完成，得出有效结果，而且 **SM1.0** 和 **SM1.2** 包含有效状态。

自然对数自然指数指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果
指令列表： LN IN, OUT EXP IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *LD, *AC
OUT	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

功能说明：

自然对数指令（LN）对 IN 中的数值进行自然对数计算，并将结果置于 OUT 中。欲从自然对数获得以 10 为底数的对数，用自然对数除以 2.302585（约等于 10 的自然对数）。欲将任何实数提升为另一个实数的乘幂，包括分数指数：将"自然指数"指令与"自然对数"指令相结合。例如，欲将 X 提升为 Y 乘幂，输入以下指令：EXP (Y * LN (X))。

自然指数指令（EXP）进行 e 的 IN 次方指数计算，并将结果置于 OUT 中。欲将任何实数提升为另一个实数的乘幂，包括分数指数：将"自然指数"指令与"自然对数"指令相结合。例如，欲将 X 提升为 Y 乘幂，输入以下指令：

EXP (Y * LN (X))。

举例：

5 的立方 = $5^3=EXP(3*LN(5))=125$

125 的立方根 = $125^{(1/3)}=EXP(1/3)*LN(125))=5$

5 的立方的平方根 = $5^{(3/2)}=EXP(3/2*LN(5))=11.18034$

SM1.1 用于指示溢出错误和非法数值。如果设置 SM1.1，则 SM1.0 和 SM1.2 状态无效，且原来的输入操作数不改动。如果未设置 SM1.3，则数学操作完成，并产生有效的结果，且 SM1.0 和 SM1.2 包含有效状态。

PID 控制指令	设置 ENO = 0的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.1 溢出
	指令列表： PID TBL LOOP	
梯形图：		
输入输出参数	数据类型	适用操作数
TBL	字节	VB
LOOP	字节	常数(0 to 7)

功能说明：

本指令有两个操作数：表示回路表起始地址的 TBL 地址和 0 至 7 常数的"回路"号码。程序中可使用八条 PID 指令。如果两条或多条 PID 指令使用相同的回路号码(即使它们的表格地址不同)，PID 计算会互相干扰，结果难以预料。回路表存储用于控制和监控回路运算的参数，包括程序变量、设置点、输出、增益、样本时间、整数时间(重设)、导出时间(速率)以及整数和(偏差)的当前值及先前值。

如果回路表起始地址或指令中指定的 PID 回路号码操作数超出范围，CPU 编译器将生成一则错误(范围错误)，编译将会失败。

PID 指令不对某些回路表输入值进行范围检查。您必须保证程序变量和设置点(以及作为输入的偏差和先前程序变量)是 0.0 和 1.0 之间的实数。

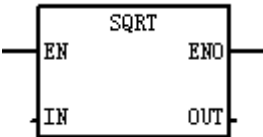
如果进行 PID 计算的数学运算时遇到错误，将设置 SM1.1(溢出或非法数值)并终止 PID 指令的执行。(对回路表中的输出数值的更新可能不完整，因此您应当忽略这些数值，并在执行下一个回路 PID 指令之前纠正引起数学错误的输入值。)

取样速率：

欲按要求的样品速率进行 PID 计算，必须按计时器的控制速率从定时中断程序或从主程序执行 PID 指令。样品时间必须通过回路表作为 PID 指令输入提供。

使用 PID 向导：

MagicWorks PLC 提供 PID 向导，指导您为闭路控制程序定义 PID 算法。选择菜单命令工具 > 指令向导，并从指令向导窗口选择 PID。

平方根指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果
指令列表: Sqrt IN OUT		
梯形图: 		
输入输出参数	数据类型	适用操作数
IN	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *LD, *AC
OUT	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

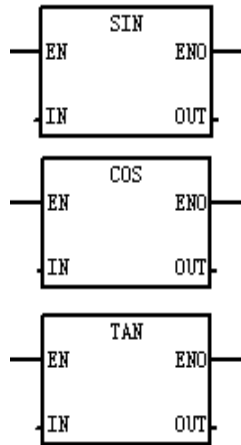
功能说明:

平方根指令 (SQRT) 对 32 位实数 (IN) 取平方根，产生一个 32 位实数结果并将结果放入 OUT 中，如以下等式所示：

$$\sqrt{IN} = OUT$$

SM1.1 用于表示溢出错误和非法数值。如果设置 SM1.1，则 SM1.0 和 SM1.2 状态无效，原始输入操作数不变。如果未设置 SM1.1，则说明数学运算已完成，得出有效结果，而且 SM1.0 和 SM1.2 包含有效状态。

正弦余弦正切指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出
	影响的特殊内存位	SM1.0 零结果 SM1.1 溢出 SM1.2 负结果
指令列表: SIN IN OUT COS IN OUT TAN IN OUT		
梯形图:		



输入输出参数	数据类型	适用操作数
IN	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *LD, *AC
OUT	浮点数	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

功能说明：

正弦指令（SIN）对角度值 IN 进行三角运算，并将结果放置在 OUT 中。输入角以弧度为单位。欲将输入角从角度转换成弧度，用角度乘以 1.745329E-2（约等于 pi/180）。

余弦指令（COS）对角度值 IN 进行三角运算，并将结果放置在 OUT 中。输入角以弧度为单位。欲将输入角从角度转换成弧度，用角度乘以 1.745329E-2（约等于 pi/180）。

正切指令（TAN）对角度值 IN 进行三角运算，并将结果放置在 OUT 中。输入角以弧度为单位。欲将输入角从角度转换成弧度，用角度乘以 1.745329E-2（约等于 pi/180）。

SM1.1 用于指示溢出错误和非法数值。如果设置 SM1.1，则 SM1.0 和 SM1.2 状态无效，且原来的输入操作数不改动。如果未设置 SM1.3，则数学操作完成，并产生有效的结果，且 SM1.0 和 SM1.2 包含有效状态。

转换指令

字符串至整数转换指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 009B 非法指数(指定起始位置值0的字符串操作) SM1.1 溢出或非法值
	影响的特殊内存位	SM1.1(溢出或非法值)
指令列表： STI IN, INDEX, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数

IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, AC, *VD, *LD, *AC
INDEX	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式:

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束（"字符串常数"）。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节):

字符串长度	字节 1	字节 2	字节 3	字节 n	字节 254

STI 指令用以下形式转换字符串:

[spaces] [+ or -] [digits 0-9]

INDEX 值通常设为 1，从字符串的第一个字符开始转换。可将该值设为其他值，在字符串中的不同点开始转换。当输入字符串包含不属于需要转换数字一部分的文本时，可采用此种方法。例如，如果输入字符串是"value:16000"，您可以将 INDEX 设为值 7，跳过字符串开始的字 "value:"。

当达到字符串结束时或遇到第一个无效字符时，转换终止。无效字符是数字（0-9）以外的任何字符。

每当转换产生一个对于输出值过大的整数值时，则设置溢出错误（SM1.1）。例如，如果输入字符串产生一个大于 32767 或小于-32768 的值时，STI 指令设置溢出错误。

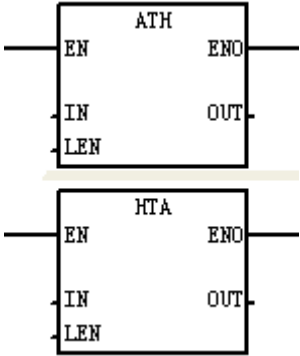
如果当输入字符串未包含有效值而无法执行转换时，也会设置溢出错误（SM1.1）。例如，如果输入字符串包含"A123"，转换指令会设置 SM1.1（溢出），输出值保持不变。

下表是有效和无效字符串转换为整数/双整数/实数时的举例（INDEX = 1）：

输入字符串	字符串有效性	转换为整数	转换为双整数	转换为实数
"65535"	有效	65535	65535	65535.0
"-0032768"	有效	-32768	-32768	-32768.0
" +32767.999 "	有效	32767	32767	32767.0
"00065535FFFF"	有效	65535	65535	65535.0
"65536"	有效	65536(SM1.1=1, 溢出)	65536	65536.0
"-002147483648"	有效	-2147483648(SM1.1=1, 溢出)	-2147483648	-2147483648.0
" +002147483648"	有效	+2147483648(SM1.1=1, 溢出)	+2147483648(SM1.1=1, 溢出)	+2147483648.0
"F255"	无效 (SM1.1=1, 无效)			

"++255"	无 效 (SM1.1=1, 无效)			
" "	无 效 (SM1.1=1, 无效)			

功能说明： 字符串至整数转换指令将字符串数值 IN 转换为整数值，并将转换结果存储在 OUT 指定的地址中，从字符串的第 INDEX 个字符位置开始转换。

ASCII 码与十六进制数字的转换指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 SM1.7 非法 ASCII 值(ATH)
	影响的特殊内存位	SM1.7 (非法 ASCII)
指令列表： ATH IN, OUT, LEN HTA IN, OUT, LEN		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD
LEN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC

操作数说明：

有效 ASCII 输入字符为：

字母数字字符 0～9 和 A～F

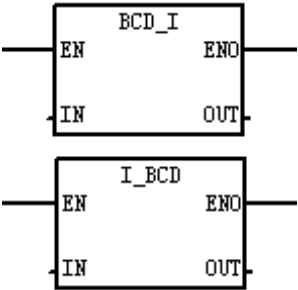
对应十六进制代码值 30～39 和 41～46

功能说明：

ASCII 至 HEX 转换指令将从内存位置 IN 开始的长度为 LEN 的 ASCII 字符码转换十六进制数字，并放入 OUT 指定的地址。ASCII 字符串的最大长度为 255 字符。

HEX 至 ASCII 转换指令将从输入字节 IN 开始的十六进制数字转换成从 OUT 开始的 ASCII 字符。欲转换的十六进制数字位数由长度 LEN 指定，可转换的最大十六进制数字位数为 255。

整数与 BCD 码转换指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.6 无效 BCD 数值

	影响的特殊内存位	SM1.6（无效 BCD）
指令列表： BCDI OUT IBCD OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *AC, *LD
OUT	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC

操作数说明：

BCD 格式的有效范围：

数据类型	最小值			最大值		
	十进制	十六进制	BCD 码	十进制	十六进制	BCD 码
字	0	0000	0000 0000 0000 0000	9999	9999	1001 1001 1001 1001

功能说明：

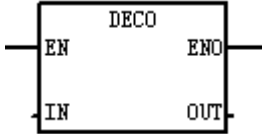
BCD 至整数指令将二进制编码的十进制值 IN 转换成整数值，并将结果载入 OUT 指定的变量中。IN 的有效范围是 0 至 9999 BCD。

整数至 BCD 指令将输入整数值 IN 转换成二进制编码的十进制数，并将结果载入 OUT 指定的变量中。IN 的有效范围是 0 至 9999 BCD。例如，您可以将双整值转换为实数。您还可以在整数和 BCD 格式之间转换。对于 STL，IN 和 OUT 参数使用相同的地址。

字节至整数转换指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： BTI IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *AC, *VD, *LD

OUT	整数	VW, IW, QW, MW, SW, SMW, LW, AQW, T, C, AC, *VD, *LD, *AC

功能说明：字节至整数转换指令将字节数值 IN 转换成整数值，并将结果写入 OUT 指定的变量中。因为字节不带符号，所以无符号扩展。

解码指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： DECO IN, OUT		
梯形图：		
		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SMB, LB, SB, AC, 常数, *VD, *LD, *AC
OUT	字	VW, IW, QW, MW, SMW, LW, SW, AQW, T, C, AC, *VD, *AC, *LD

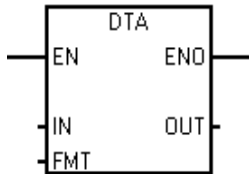
功能说明：解码指令设置输出字 OUT 中与输入字节 IN 低半字节（低 4 位）表示的位数相对应的位为 1，设置输出字 OUT 的所有其它位均为 0。

示例程序：

LD SM0.0

DECO 2#10101111, VW0 //设置 VW0 的第 15(2#1111)位为 1，其它位为 0

运行结果：VW0 = 2#1000 0000 0000 0000

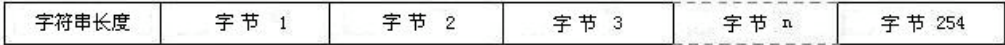
双整数至 ASCII 码转换指令	设置 ENO = 0 的错误条件	0006 间接地址 FMT 位 > 0（用于 FMT 值最高的 4 个位） nnn > 5
	影响的特殊内存位	无
指令列表： DTA IN, OUT, FMT		
梯形图：		
		
输入输出参数	数据类型	适用操作数
IN	双整数	VD, ID, QD, MD, SD, SMD, LD, HC, 常数, AC, *VD, *AC, *LD
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, *VD, *LD, *AC
FMT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC

操作数说明：

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节)：



操作数 FMT 定义：

位号	7	6	5	4	3	2	1	0
值	0	0	0	0	c	n	n	n

c: 整数与小数部分的分隔符，c = 1:使用逗号作为整数与小数部分的分隔符；c = 0:使用小数点作为整数与小数部分的分隔符

nnn: 小数部分的位数,有效范围为 0~5，输出字符串的长度始终为 8 个字符。nnn 等于 0 会使转换结果显示为不带小数点，当 nnn 值大于 5 时，输出显示为 12 个 ASCII 空格字符的字符串。

整数至 ASCII 码的转换遵循下列规则：

- 1) 正值写入输出缓冲区，不带符号。
- 2) 负值写入输出缓冲区，带起始负号(-)。
- 3) 小数点左侧的起首零（与小数点相邻的数字除外）被压缩。
- 4) 输出缓冲区中的数值右对齐。

下表显示几个使用小数点（c = 0）和小数点右面四位小数（nnn = 100）格式的值范例。

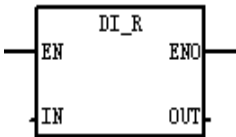
	OUT	字节 OUT										
		+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11
IN=123							0	.	0	1	2	3
IN=-12345						-	1	.	2	3	4	5
IN=1234567					1	2	3	.	4	5	6	7

功能说明：整数至 ASCII 转换指令将整数值 IN 转换成 ASCII 字符数组。格式 FMT 指定转换的精度，以及是将小数点显示为逗号还是点号。转换结果放入从 OUT 开始的 12 个连续字节中。ASCII 字符数组总是 12 个字符。

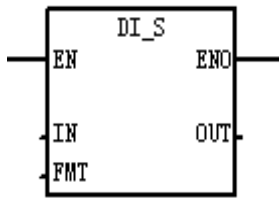
双整数至整数转换指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出或非法数值
	影响的特殊内存位	SM1.1（溢出）
指令列表： DTI IN, OUT		
梯形图：		

输入输出参数	数据类型	适用操作数
IN	双整数	VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *LD, *AC
OUT	整数	VW, IW, QW, MW, SW, SMW, LW, AQW, T, C, AC, *VD, *LD, *AC

功能说明：双整数至整数转换指令将输入的双整数值 IN 转换成整数值，并将结果放入 OUT 指定的变量中。如果转换的值过大，则无法在输出中表示，会导致溢出错误，输出不受影响。

双整数至实数转换指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： DTR IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	双整数	VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *AC, *LD
OUT	实数	VD, ID, QD, MD, SD, SMD, LD, HC, AC, *VD, *AC, *LD

功能说明：双整数至实数转换指令将 32 位带符号的整数 IN 转换成 32 位实数，并将结果放入 OUT 指定的变量中。

双整数至字符串转换指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 非法格式 (nnn > 5)
	影响的特殊内存位	无
指令列表： DTS IN, OUT, FMT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	双整数	VD, ID, QD, MD, SD, SMD, LD, HC, 常数, AC, *VD, *AC, *LD
OUT	字符串	VB, *VD, LB, *AC, *LD
FMT	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数

据类型的格式(一个方格代表一个字节):

字符串长度	字节 1	字节 2	字节 3	字节 n	字节 254

操作数 FMT 定义:

位号	7	6	5	4	3	2	1	0
值	0	0	0	0	c	n	n	n

c: 整数与小数部分的分隔符, c = 1:使用逗号作为整数与小数部分的分隔符; c = 0:使用小数点作为整数与小数部分的分隔符。

nnn: 小数部分的位数,有效范围为 0~5, 输出字符串的长度始终为 12 个字符。nnn 等于 0 会使转换结果显示格式为不带小数点, 当 nnn 值大于 5 时, 输出显示为 12 个 ASCII 空格字符的字符串。

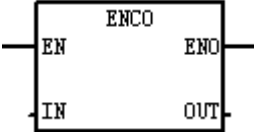
下表显示几个使用小数点 (c = 0) 和小数点右面四位小数 (nnn = 100) 格式的值范例。位于 OUT 位置的值是字符串长度。

	OUT	字节 OUT											
		+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12
IN=123	12							0	.	0	1	2	3
IN=12345	12							1	.	2	3	4	5
IN=1234567	12					1	2	3	.	4	5	6	7

功能说明: 双整数至字符串转换指令将输入的双整数 IN 转换为长度为 12 个字符的 ASCII 字符串。格式 (FMT) 指定小数点右面的转换精度, 小数点是显示为逗号还是句点, 转换结果字符串写入从 OUT 开始的 13 个连续字节中。

提示: 根据以下规则制定格式的输出字符串:

- 整数不带符号写入输出缓冲区。
- 负数带起始减号 (-) 写入输出缓冲区。
- 小数点左面的起始零 (除紧靠小数点的数字外) 被压缩。
- 小数点右面的数值被取整, 使之符合指定的小数点右面的位数。
- 输出字符串的大小最小必须比小数点右面的位数大三个字节。
- 输出字符串中的数值必须右对齐。

编码指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表: ENCO IN, OUT		
梯形图: 		
输入输出参数	数据类型	适用操作数
IN	字	VW, IW, QW, MW, SMW, LW, SW, AIW, T, C, AC, 常数, *VD, *AC, *LD

OUT	字节	VB, IB, QB, MB, SMB, LB, SB, AC, *VD, *LD, *AC

功能说明：编码指令将输入字 IN 位值为 1 的最低位的位数作为输出字节 OUT 的低半字节，写入输出字节的最低 4 位中。

示例程序：

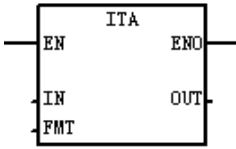
LD SM0.1

MOVB 0, VB0

LD SM0.0

ENCO 2#1100000000, VB0 //2#1100000000 最低不为 0 的位是第 8 位，因此 VB0 的低半字节为 2#1000

运行结果：VB0 = 2#0000 1000

整数至 ASCII 码转换指令	设置 ENO = 0 的错误条件	0006 间接地址 FMT 位 > 0（用于 FMT 值最高的 4 个位） nnn > 5
	影响的特殊内存位	无
指令列表： ITA IN, OUT, FMT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	整数	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, *VD, *LD, *AC
FMT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束（"字符串常数"）。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式（一个方格代表一个字节）：

字符串长度	字节 1	字节 2	字节 3	字节 n	字节 254

操作数 FMT 定义：

位号	7	6	5	4	3	2	1	0
值	0	0	0	0	c	n	n	n

c: 整数与小数部分的分隔符，c = 1:使用逗号作为整数与小数部分的分隔符；c = 0:使用小数点作为整数与小数部分的分隔符。

nnn：小数部分的位数,有效范围为 0～5，输出字符串的长度始终为 8 个字符。nnn 等于 0 会使转换结果显示为不带小数点，当 nnn 值大于 5 时，输出显示为 8 个 ASCII 空格字符的字符串。

整数至 ASCII 码的转换遵循下列规则：

- 1) 正值写入输出缓冲区，不带符号。
- 2) 负值写入输出缓冲区，带起始负号(-)。
- 3) 小数点左侧的起首零（与小数点相邻的数字除外）被压缩。
- 4) 输出缓冲区中的数值右对齐。

下表显示几个使用小数点（c = 0）和小数点右面三位小数（nnn = 011）格式的值范例。

	OUT	字节 OUT						
		+1	+2	+3	+4	+5	+6	+7
IN=12				0		0	1	2
IN=-123				0		1	2	3
IN=1234				1		2	3	4
IN=-12345			1	2		3	4	5

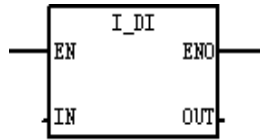
功能说明：整数至 ASCII 转换指令将整数值 IN 转换成 ASCII 字符数组。格式 FMT 指定转换的精度，以及是将小数点显示为逗号还是点号。转换结果放入从 OUT 开始的 8 个连续字节中。ASCII 字符数组总是 8 个字符。

整数至字节转换指令		设置 ENO = 0 的错误条件	0006 间接地址 溢出或非法值
		影响的特殊内存位	SM1.1（溢出）
指令列表： ITB IN, OUT			
梯形图：			
输入输出参数	数据类型	适用操作数	
IN	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *LD, *AC	
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	

功能说明：整数至字节转换指令将输入的整数 IN 转换成字节值，并将结果放入 OUT 指定的变量中。数值 0 至 255 被转换。超过此范围的输入值将导致溢出，输出不受影响。

整数至双整数转换指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： ITD IN, OUT		

梯形图:



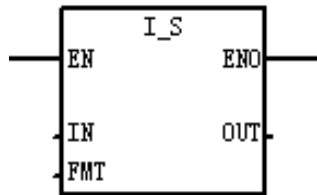
输入输出参数	数据类型	适用操作数
IN	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

功能说明: 整数至双整数转换指令将输入的整数值 IN 转换成双整数值, 并将结果放入 OUT 指定的变量中。符号被扩展。

整数至字符串转换指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 非法格式 (nnn > 5)
	影响的特殊内存位	无

指令列表: ITS IN, OUT, FMT

梯形图:



输入输出参数	数据类型	适用操作数
IN	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, 常数, AC, *VD, *LD, *AC
OUT	字符串	VB, *VD, LB, *AC, *LD
FMT	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式:

字符串是一系列字符和对应的内存地址, 每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度 (即字符数) 的整数。如果常数字符串被直接输入程序编辑器或数据块, 那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节 (254 个字符加上长度字节)。下面的内存图显示了字符串数据类型的格式 (一个方格代表一个字节):

字符串长度	字节 1	字节 2	字节 3	字节 n	字节 254

操作数 FMT 定义:

位号	7	6	5	4	3	2	1	0
值	0	0	0	0	c	n	n	n

c: 整数与小数部分的分隔符, c = 1: 使用逗号作为整数与小数部分的分隔符; c = 0: 使用小数点作为整数与小数部分的分隔符。

nnn：小数部分的位数，有效范围为 0~5，输出字符串的长度始终为 8 个字符。nnn 等于 0 会使转换结果显示格式为不带小数点，当 nnn 值大于 5 时，输出显示为 8 个 ASCII 空格字符的字符串。

下表显示几个使用小数点（c = 0）和小数点右面三位小数（nnn = 011）格式的值范例。位于 OUT 位置的值是字符串长度。

	OUT	字节 OUT							
		+1	+2	+3	+4	+5	+6	+7	+8
IN=12	8				0	.	0	1	2
IN=-123	8				0	.	1	2	3
IN=1234	8				1	.	2	3	4
IN=-12345	8			1	2	.	3	4	5

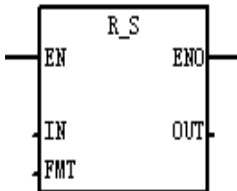
功能说明：整数至字符串转换指令将输入的整数 IN 转换为长度为 8 个字符的 ASCII 字符串。格式（FMT）指定小数点右面的转换精度，小数点是显示为逗号还是句点，转换结果字符串写入从 OUT 开始的 9 个连续字节中。

提示：根据以下规则制定格式的输出字符串：

- 整数不带符号写入输出缓冲区。
- 负数带起始减号（-）写入输出缓冲区。
- 小数点左面的起始零（除紧靠小数点的数字外）被压缩。
- 小数点右面的数值被取整，使之符合指定的小数点右面的位数。
- 输出字符串的大小最小必须比小数点右面的位数大三个字节。
- 输出字符串中的数值必须右对齐。

取整指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出或非法值
	影响的特殊内存位	SM1.1（溢出）
指令列表： ROUND IN, OUT		
梯形图：		
输入输出参数	数据类型	适用操作数
IN	实数	VD, ID, QD, MD, SD, SMD, LD, AC, 常数, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

功能说明：取整指令将输入的实数 IN 转换成双整数值，并将结果存入 OUT 指定的变量中。小数部分采取四舍五入的方法取整。

实数至字符串转换指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 非法格式: nnn > 5 ssss < 3 ssss < 要求的字符数
	影响的特殊内存位	无
指令列表: RTS IN, OUT, FMT		
梯形图: 		
输入输出参数	数据类型	适用操作数
IN	实数	VD, ID, QD, MD, SD, SMD, LD, 常数, AC, *VD, *LD, *AC
OUT	字符串	VB, LB, *VD, *AC, *LD
FMT	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式:

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节):

字符串长度	字节 1	字节 2	字节 3	字节 n	字节 254

操作数 FMT 定义:

位号	7	6	5	4	3	2	1	0
值	s	s	s	s	c	n	n	n

ssss: 输出字符串长度，小于 3 无效。

c: 整数与小数部分的分隔符，c = 1:使用逗号作为整数与小数部分的分隔符；c = 0:使用小数点作为整数与小数部分的分隔符。

nnn: 小数部分的位数,有效范围为 0~5，nnn 等于 0 会使转换结果显示格式为不带小数点，当 nnn 值大于 5 时或当指定的输出字符串长度太小无法存储转换的值时，输出字符串用 ASCII 空格字符填充。

下表显示几个使用小数点（c = 0）和小数点右面有一位数（nnn = 001）以及输出字符串长度为 7 个字符（ssss = 0111）格式的值范例。位于 OUT 位置的值是字符串长度。

	OUT	字节 OUT						
		+1	+2	+3	+4	+5	+6	+7
IN=12345.6	7	1	2	3	4	5	.	6
IN=0.0005	7					0	.	0

IN=-3.6544444	7				-	3	.	7

功能说明：实数至字符串转换指令将实数值 IN 转换为 ASCII 字符串。格式（FMT）指定输出字符串的长度、转换精度以及小数点是显示为逗号还是句点。转换结果放入以 OUT 开始的字符串中。结果字符串长度在格式中指定，可以是 3 至 15 个字符。CTSC-200 使用的实数格式最多可支持 7 个高位数字。设置 7 个以上高位数字会产生取整错误。

提示：根据以下规则制定格式的输出字符串：

- 整数不带符号写入输出缓冲区。
- 负数带起始减号（-）写入输出缓冲区。
- 小数点左面的起始零（除紧靠小数点的数字外）被压缩。
- 小数点右面的数值被取整，使之符合指定的小数点右面的位数。
- 输出字符串的大小最小必须比小数点右面的位数大三个字节。
- 输出字符串中的数值必须右对齐。

段指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： SEG IN, OUT		
梯形图：		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD
OUT	字节	VB, IB, QB, MB, SMB, LB, AC, *VD, *AC, SB, *LD

操作数说明：

下图显示了段指令中用到的七段显示器段编码。

(进) LSD	段显示	(出) -gfe dcba		(进) LSD	段显示	(出) -gfe dcba
0	0	0011 1111		8	8	0111 1111
1	1	0000 0110		9	9	0110 0111
2	2	0101 1011		A	A	0111 0111
3	3	0100 1111		B	b	0111 1100
4	4	0110 0110		C	c	0011 1001
5	5	0110 1101		D	d	0101 1110
6	6	0111 1101		E	E	0111 1001
7	7	0000 0111		F	F	0111 0001

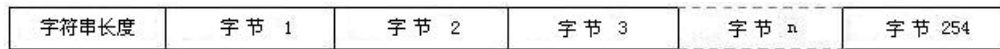
功能说明：段（SEG）指令允许您生成照明七段显示段的位格式。

字符串至双整数 转换指令	设置 ENO = 0 的错误 条件	0006 间接地址 0091 操作数范围 009B 非法指数(指定起始位置值0的字符串操作) SM1.1 溢出或非法值
	影响的特殊内存位	SM1.1(溢出或非法值)
指令列表: STD IN, INDEX, OUT		
梯形图:		
输入输出参数	数据类型	适用操作数
IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC
INDEX	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式:

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束（"字符串常数"）。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式（一个方格代表一个字节）：



STD 指令用以下形式转换字符串：

[spaces] [+ or -] [digits 0-9]

INDEX 值通常设为 1，从字符串的第一个字符开始转换。可将该值设为其他值，在字符串中的不同点开始转换。当输入字符串包含不属于需要转换数字一部分的文本时，可采用此种方法。

例如，如果输入字符串是"value:16000"，您可以将 INDEX 设为值 7，跳过字符串开始的字"value:"。

当达到字符串结束时或遇到第一个无效字符时，转换终止。无效字符是数字（0-9）以外的任何字符。

每当转换产生一个对于输出值过大的整数值时，则设置溢出错误（SM1.1）。例如，如果输入字符串产生一个大于+2147483647 或小于-+2147483648 的值时，STD 指令设置溢出错误。

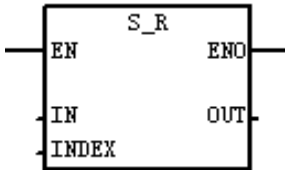
如果当输入字符串未包含有效值而无法执行转换时，也会设置溢出错误（SM1.1）。例如，如果输入字符串包含"A123"，转换指令会设置 SM1.1（溢出），输出值保持不变。

下表是有效和无效字符串转换为整数/双整数/实数时的举例（INDEX = 1）：

输入字符串	字符串有效性	转换为整数	转换为双整数	转换为实数
"65535"	有效	65535	65535	65535.0
"-0032768"	有效	-32768	-32768	-32768.0

"+32767.999 "	有效	32767	32767	32767.0
"00065535FFFF"	有效	65535	65535	65535.0
"65536"	有效	65536(SM1.1=1,溢出)	65536	65536.0
"-002147483648"	有效	-2147483648(SM1.1=1,溢出)	-2147483648	-2147483648.0
"+002147483648"	有效	+2147483648(SM1.1=1,溢出)	+2147483648(SM1.1=1,溢出)	+2147483648.0
"F255"	无效(SM1.1=1,无效)			
"++255"	无效(SM1.1=1,无效)			
" "	无效(SM1.1=1,无效)			

功能说明：字符串至双整数转换指令将字符串数值 IN 转换为双整数值，并将转换结果存储在 OUT 指定的地址中，从字符串的第 INDEX 个字符位置开始转换。

字符串至实数转换指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 009B 非法指数(指定起始位置值0的字符串操作) SM1.1 溢出或非法值
	影响的特殊内存位	SM1.1(溢出或非法值)
指令列表： STR IN, INDEX, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	实数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC
INDEX	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节)：

字符串长度	字节 1	字节 2	字节 3	字节 n	字节 254

STI 指令用以下形式转换字符串：

[spaces] [+ or -] [digits 0-9]

INDEX 值通常设为 1，从字符串的第一个字符开始转换。可将该值设为其他值，在字符串中的不同点开始转换。当输入字符串包含不属于需要转换数字一部分的文本时，可采用此种方法。

例如，如果输入字符串是"value:16000"，您可以将 INDEX 设为值 7，跳过字符串开始的字 "value:"。

字符串至实数转换指令不会使用实数的科学计数法或指数形式转换字符串。该指令不会产生溢出错误(SM1.1)，但只会将字符串转换为直至指数的实数，然后终止转换。例如，字符串"1.234E6"将转换为实数值 1.234，而不产生错误讯息。

当达到字符串结束时或遇到第一个无效字符时，转换终止。无效字符是数字（0-9）以外的任何字符。

每当转换产生一个对于输出值过大的整数值时，则设置溢出错误（SM1.1）。

如果当输入字符串未包含有效值而无法执行转换时，也会设置溢出错误（SM1.1）。例如，如果输入字符串包含"A123"，转换指令会设置 SM1.1（溢出），输出值保持不变。

下表是有效和无效字符串转换为整数/双整数/实数时的举例（INDEX = 1）：

输入字符串	字符串有效性	转换为整数	转换为双整数	转换为实数
"65535"	有效	65535	65535	65535.0
"-0032768"	有效	-32768	-32768	-32768.0
"+32767.999 "	有效	32767	32767	32767.0
"00065535FFFF"	有效	65535	65535	65535.0
"65536"	有效	65536(SM1.1=1, 溢出)	65536	65536.0
"-002147483648"	有效	-2147483648(SM1.1=1, 溢出)	-2147483648	-2147483648.0
"+002147483648"	有效	+2147483648(SM1.1=1, 溢出)	+2147483648(SM1.1=1, 溢出)	+2147483648.0
"F255"	无效 (SM1.1=1, 无效)			
"++255"	无效 (SM1.1=1, 无效)			
" "	无效 (SM1.1=1, 无效)			

功能说明：字符串至实数转换指令将字符串数值 IN 转换为实数值，并将转换结果存储在 OUT 指定的地址中，从字符串的第 INDEX 个字符位置开始转换。

截断指令	设置 ENO = 0 的错误条件	0006 间接地址 SM1.1 溢出或非法值
	影响的特殊内存位	SM1.1（溢出）
指令列表： TRUNC IN, OUT		

梯形图：		
输入输出参数	数据类型	适用操作数
IN	实数	VD, ID, QD, MD, SD, SMD, LD, AC, 常数, *VD, *LD, *AC
OUT	双整数	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD

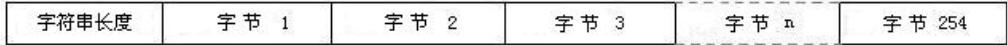
功能说明：截断指令将 32 位实数 IN 转换成 32 位双整数，并将结果放入 OUT 指定的变量中。只有实数的整数部分被转换，小数部分被丢弃。如果输入的实数值过大或无效，将会导致溢出，输出不受影响。

实数至 ASCII 码转换指令	设置 ENO = 0 的错误条件	0006 间接地址 nnn > 5 ssss < 3 ssss < OUT 中的字符数
	影响的特殊内存位	无
指令列表： RTA IN, OUT, FMT		
梯形图：		
输入输出参数	数据类型	适用操作数
IN	实数	VD, ID, QD, MD, SD, SMD, LD, 常数, AC, *VD, *LD, *AC
OUT	字符串	VB, LB, *VD, *AC, *LD
FMT	字节	VB, IB, QB, MB, SB, SMB, LB, 常数, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型格式(一个方格代表一个字节)：



操作数 FMT 定义：

位号	7	6	5	4	3	2	1	0
值	s	s	s	s	c	n	n	n

ssss：输出字符串长度，小于 3 无效。

c：整数与小数部分的分隔符，c = 1:使用逗号作为整数与小数部分的分隔符；c = 0:使用小数点

作为整数与小数部分的分隔符。

nnn: 小数部分的位数,有效范围为 0~5, **nnn** 等于 0 会使转换结果显示格式为不带小数点, 当 **nnn** 值大于 5 时或当指定的输出字符串长度太小无法存储转换的值时, 输出字符串用 ASCII 空格字符填充。

下表显示几个使用小数点 (**c = 0**) 和小数点右面有一位数 (**nnn = 001**) 以及输出字符串长度为 7 个字符 (**ssss = 0111**) 格式的值范例。位于 **OUT** 位置的值是字符串长度。

	OUT	字节 OUT					
		+1	+2	+3	+4	+5	+6
IN=12345.6	1	2	3	4	5	.	6
IN=0.0005					0	.	0
IN=-3.6544444				-	3	.	7

功能说明: 实数至 ASCII 码转换指令将实数值 **IN** 转换为 ASCII 字符。格式 (**FMT**) 指定输出字符串的长度、转换精度以及小数点是显示为逗号还是句点。转换结果放入以 **OUT** 开始的字符串中。结果字符串长度在格式中指定, 可以是 3 至 15 个字符。

提示: 根据以下规则制定格式的输出字符串:

- 整数不带符号写入输出缓冲区。
- 负数带起始减号 (-) 写入输出缓冲区。
- 小数点左面的起始零 (除紧靠小数点的数字外) 被压缩。
- 小数点右面的数值被取整, 使之符合指定的小数点右面的位数。
- 输出字符串的大小最小必须比小数点右面的位数大三个字节。
- 输出字符串中的数值必须右对齐。

实时时钟

读取和设置实时时钟指令	设置 ENO = 0 的错误条件	0006 间接地址 0007 TOD 数据错误(TODW) 000C 不存在时钟
	影响的特殊内存位	SM4.3 (对此时钟有两个同时访问, 非重要错误0007)
指令列表: TODR T TODW T		
梯形图:		

READ_RTC EN ENO T SET_RTC EN ENO T		
输入输出参数	数据类型	适用操作数
时钟 T	字节	VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD

操作数说明：所有日期和时间值必须采用 BCD 格式编码（例如，16#97 代表 2002 年）。请参阅下表：

8 个字节时间缓冲区格式（T）

T 字节	说明	字节数据
0	年 (0-99)	当前年份后两位 (BCD 码)
1	月 (1-12)	当前月份 (BCD 码)
2	日期 (1-31)	当前日期 (BCD 码)
3	小时 (0-23)	当前小时 (BCD 码)
4	分钟 (0-59)	当前分钟 (BCD 码)
5	秒 (0-59)	当前秒 (BCD 码)
6	00	保留字节，始终设置为00
7	星期几 (1-7)	当前是星期几，1=星期日 (BCD 码)

长时间掉电或内存丢失后，实时时钟会被初始化为以下日期和时间：

日期： 90 年 1 月 1 日
时间： 00:00:00
星期： 星期日

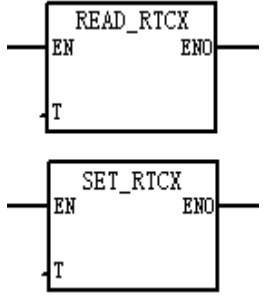
提示：CTSC-200 CPU 无法根据日期和星期是否正确，所以在输入时应当确保日期正确。请勿在主程序和中断程序中使用 TODR/TODW 指令。当另一条 TODR/TODW 指令正在执行时，若尝试执行中断程序中的 TODR/TODW 指令，则该指令不会被执行，同时 SM4.3 被置位。

功能说明：

读取实时时钟指令（TODR）从硬件时钟读取当前时间和日期，并将其写入以地址 T 起始的 8 个字节的缓冲区。

设置实时时钟指令（TODW）设置当前硬件时钟为以 T 起始的 8 个字节的缓冲区的内容。

读取和设置扩展 实时时钟指令	设置 ENO = 0 的错误条件	0006 间接地址 0007 TOD 数据错误(TODW) 000C 不存在时钟 0091 范围错误

	影响的特殊内存位	SM4.3 (对此时钟有两个同时访问, 非重要错误0007)
指令列表: TODRX T TODWX T		
梯形图:		
		
输入输出参数	数据类型	适用操作数
时钟 T	字节	VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD

操作数说明: 所有日期和时间值必须采用 BCD 格式编码 (例如, 16#97 代表 2002 年)。请参阅下表:

8 个字节时间缓冲区格式 (T)

T 字节	说明	字节数据
0	年 (0-99)	当前年份后两位 (BCD 码)
1	月 (1-12)	当前月份 (BCD 码)
2	日期 (1-31)	当前日期 (BCD 码)
3	小时 (0-23)	当前小时 (BCD 码)
4	分钟 (0-59)	当前分钟 (BCD 码)
5	秒 (0-59)	当前秒 (BCD 码)
6	00	保留字节, 始终设置为00
7	星期几 (1-7)	当前是星期几, 1=星期日 (BCD 码)
8	模式(00H-03H, 08H, 10H-13H, FFH)	修正模式: 00H = 修正已禁用 01H = 欧盟 (相对于 UTC 的时区调整 = 0小时) 02H = 欧盟 (相对于 UTC 的时区调整 = +1小时) 03H = 欧盟 (相对于 UTC 的时区调整 = +2小时) 04H-07H = 保留 08H = 欧盟 (相对于 UTC 的时区调整 = -1小时) 09H-0FH = 保留 10H = 美国 11H = 澳大利亚 12H = 澳大利亚 (塔斯马尼亚) 13H = 新西兰 14H-FDH = 保留 FEH = 保留
9	修正小时数 (0-23)	修正数量, 小时 (BCD 码)
10	修正分钟数 (0-59)	修正数量, 分钟 (BCD 码)

11	开始月份 (1-12)	夏时制的开始月份 (BCD 码)
12	开始日期 (1-31)	夏时制的开始日期 (BCD 码)
13	开始小时 (0-23)	夏时制的开始小时 (BCD 码)
14	开始分钟 (0-59)	夏时制的开始分钟 (BCD 码)
15	结束月份 (1-12)	夏时制的结束月份 (BCD 码)
16	结束日期 (1-31)	夏时制的结束日期 (BCD 码)
17	结束小时 (0-23)	夏时制的结束小时 (BCD 码)
18	结束分钟 (0-59)	夏时制的结束分钟 (BCD 码)

欧盟常规：在三月最后一个星期日的 UTC 时间凌晨一点将时间向前调一小时。在十月最后一个星期日的 UTC 时间凌晨两点将时间往回调一小时。(当做出修正时，当地时间因相对于 UTC 的时区调整而不同)

美国常规：在四月第一个星期日的当地时间凌晨两点将时间向前调一小时。在十月最后一个星期日的当地时间凌晨两点将时间往回调一小时。

澳大利亚常规：在十月最后一个星期日的当地时间凌晨两点将时间向前调一小时。在三月最后一个星期日的当地时间凌晨三点将时间往回调一小时。

澳大利亚(塔斯马尼亚)常规：在十月第一个星期日的当地时间凌晨两点将时间向前调一小时。在三月最后一个星期日的当地时间凌晨三点将时间往回调一小时。

新西兰常规：在十月第一个星期日的当地时间凌晨两点将时间向前调一小时。在三月十五日或之后的第一个星期日的当地时间凌晨三点将时间往回调一小时。

长时间掉电或内存丢失后，实时时钟会被初始化为以下日期和时间：

日期：90 年 1 月 1 日
时间：00:00:00
星期：星期日

提示：CTSC-200 CPU 无法根据日期和星期是否正确，所以在输入时应当确保日期正确。请勿在主程序和中断程序中使用 TODRX/TODWX 指令。当另一条 TODRX/TODWX 指令正在执行时，若尝试执行中断程序中的 TODRX/TODWX 指令，则该指令不会被执行，同时 SM4.3 被置位。

功能说明：

读取扩展实时时钟指令（TODRX）从硬件时钟读取当前时间、日期及夏时制配置，并将其写入以地址 T 起始的 19 个字节的缓冲区。

设置扩展实时时钟指令（TODWX）设置当前硬件时钟为以 T 起始的 19 个字节的缓冲区的内容。

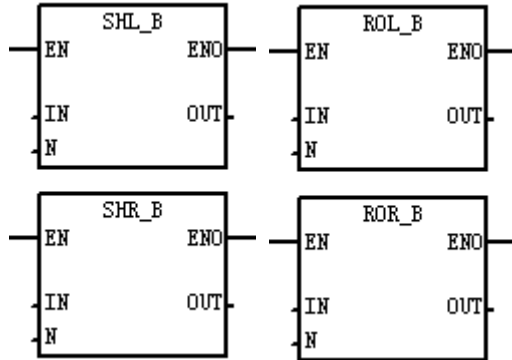
移位/循环指令

字节移位/循环指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	SM1.0 如果移位结果是零，则设置零位 SM1.1 为移出的最后一个位设置溢出位

指令列表:

SLB OUT, N
SRB OUT, N
RLB OUT, N
RRB OUT, N

梯形图:



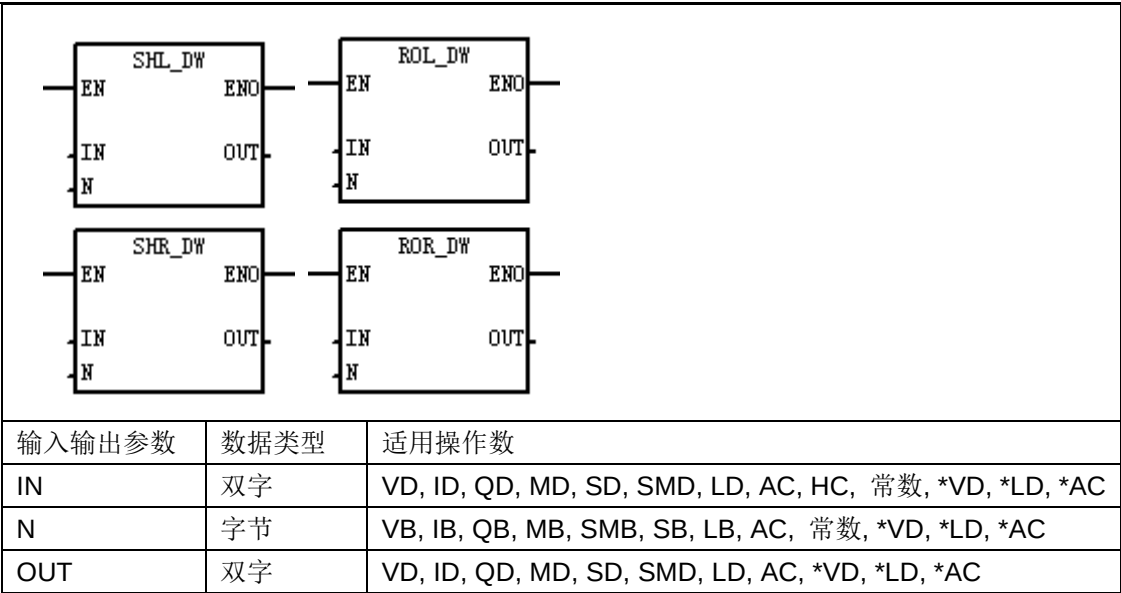
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *LD, *AC
N	字节	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

功能说明:

右移字节（SRB）和左移字节（SLB）指令将输入数值（IN）根据移位计数（N）向右或向左移动，并将结果载入输出字节（OUT）。移位指令对每个移出位补 0。如果移位数目（N）大于或等于 8，则数值最多被移位 8 次。如果移位数目大于 0，溢出内存位（SM1.1）采用最后一次移出位的数值。如果移位操作结果为 0，设置 0 内存位（SM1.0）。右移和向左移字节操作不带符号。

循环右移字节（RRB）和循环左移字节（RLB）指令将输入字节数值（IN）向右或向左旋转 N 位，并将结果载入输出字节（OUT）。旋转具有循环性。如果移位数目（N）大于或等于 8，执行旋转之前先对位数（N）进行模数 8 操作，从而使位数在 0 至 7 之间。如果移动位数为 0，则不执行旋转操作。如果执行旋转操作，旋转的最后一位数值被复制至溢出位（SM1.1）。如果移动位数不是 8 的整倍数，旋转出的最后一位数值被复制至溢出内存位（SM1.1）。如果旋转数值为 0，设置 0 内存位（SM1.0）。循环右移和循环左移字节操作不带符号。

双字移位/循环指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	SM1.0 如果移位结果是零，则设置零位 SM1.1 为移出的最后一个位设置溢出位
指令列表:		
SLD OUT, N SRD OUT, N RLD OUT, N RRD OUT, N		
梯形图:		

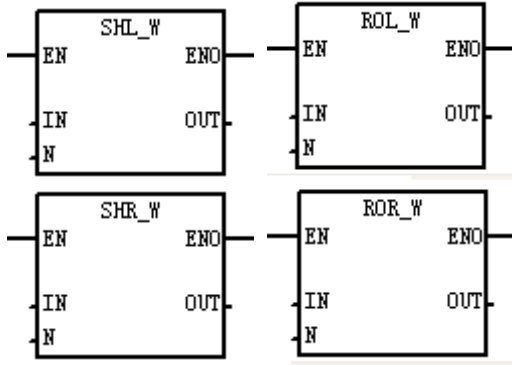


功能说明：

右移双字（SRD）和左移双字（SLD）指令将输入双字数值（IN）向右或向左移动 N 位，并将结果载入输出双字（OUT）。移位指令对每个移出位补 0。如果移位数目（N）大于或等于 32，则数值最多被移位 32 次。如果移位数目大于 0，溢出内存位（SM1.1）采用最后一次移出位数值。如果移位操作结果为 0，设置 0 内存位（SM1.0）。请注意当您使用带符号数据类型时，符号位被移位。

循环右移双字（RRD）和循环左移双字（RLD）指令将输入双字数值（IN）向右或向左旋转 N 位，并将结果载入输出双字（OUT）。旋转具有循环性。如果移位数目（N）大于或等于 32，执行旋转之前在移动位数（N）上执行模数 32 操作。从而使位数在 0 至 31 之间。如果移动位数为 0，则不执行旋转操作。如果执行旋转操作，旋转的最后一位数值被复制至溢出位（SM1.1）。如果移动位数不是 32 的整倍数，旋转出的最后一位数值被复制至溢出内存位（SM1.1）。如果旋转数值为 0，设置 0 内存位（SM1.0）。循环右移和循环左移双字操作不带符号。

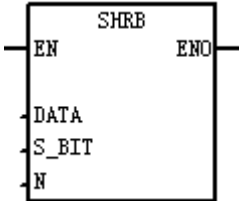
字移位/循环指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	SM1.0 如果移位结果是零，则设置零位 SM1.1 为移出的最后一个位设置溢出位
指令列表： SLW OUT, N SRW OUT, N RLW OUT, N RRW OUT, N		
梯形图：		

		
输入输出参数	数据类型	适用操作数
IN	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, Constant, *VD, *LD, *AC
N	字节	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *LD, *AC
OUT	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC

功能说明：

右移字（SRW）和左移位字（SLW）指令将输入字（IN）数值向右或向左移动 N 位，并将结果载入输出字（OUT）。移位指令对每个移出位补 0。如果移位数目（N）大于或等于 16，则数值最多被移位 16 次。如果移位数目大于 0，溢出内存位（SM1.1）采用最后一次移出位数值。如果移位操作结果为 0，设置 0 内存位（SM1.0）。请注意当您使用带符号的数据类型时，符号位被移位。

循环右移字（RRW）和循环左移字（RLW）指令将输入字数值（IN）向右或向左旋转 N 位，并将结果载入输出字（OUT）。旋转具有循环性。如果移动位数（N）大于或等于 16，在旋转执行之前的移动位数（N）上执行模数 16 操作。从而使移动位数在 0 至 15 之间。如果移动位数为 0，则不执行旋转操作。如果执行旋转操作，旋转的最后一位数值被复制至溢出位（SM1.1）。如果移动位数不是 16 的整倍数，旋转出的最后一位数值被复制至溢出内存位（SM1.1）。如果旋转数值为 0，设置 0 内存位（SM1.0）。循环右移和循环左移字操作不带符号。

移位寄存器指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数超出范围 0092 计数域错误
	影响的特殊内存位	SM1.1 为移出的最后一个位设置溢出位
指令列表： SHRB DATA, S_BIT, N		
梯形图： 		
输入输出参数	数据类型	适用操作数

DATA	布尔	I, Q, M, SM, T, C, V, S, L
S_BIT	布尔	I, Q, M, SM, T, C, V, S, L
N	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC

功能说明：

移位寄存器位（SHRB）指令将 DATA 数值移入移位寄存器。S_BIT 指定移位寄存器的最低位。N 指定移位寄存器的长度和移位方向（移位加 = N，移位减 = -N）。SHRB 指令移出的每个位被放置在溢出内存位（SM1.1）中。该指令由最低位（S_BIT）和由长度（N）指定的位数定义。

使用以下等式计算"移位寄存器"最高位地址（MSB.b）：

$$MSB.b = [(S\_BIT \text{ 字节}) + ([N] - 1 + (S\_BIT \text{ 位})) / 8] . [被 8 除的余数]$$

例如：如果 S_BIT 是 V33.4 和 N is 14，以下计算显示 MSB.b 是 V35.1。

$$\begin{aligned} MSB.b &= V33 + ([14] - 1 + 4) / 8 \\ &= V33 + 17 / 8 \\ &= V33 + 2 \text{ , 余数为 } 1 \\ &= V35.1 \end{aligned}$$

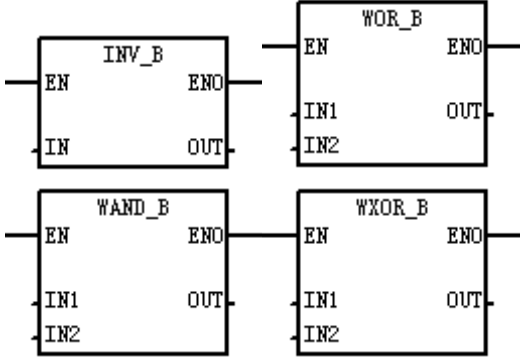
在"移位减"（用长度（N）的负值表示）中，输入数据移入移位寄存器的最高位中，并移出最低位（S_BIT）。移出的数据被放置在溢出内存位（SM1.1）中。

在"移位加"（用长度（N）的正值表示）中，输入数据（DATA）移入移位寄存器的最高位中（由 S_BIT 指定），并移出移位寄存器的最高位。移出的数据被放置在溢出内存位（SM1.1）中。

移位寄存器的最大长度为 64 位（无论正负）。

逻辑计算指令

字节取反、与、或、异或运算指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	SM1.0 操作结果等于零
指令列表： INVB OUT ANDB IN1 OUT ORB IN1 OUT XORB IN1 OUT		
梯形图：		

		
输入输出参数	数据类型	适用操作数
IN1	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD
IN2	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

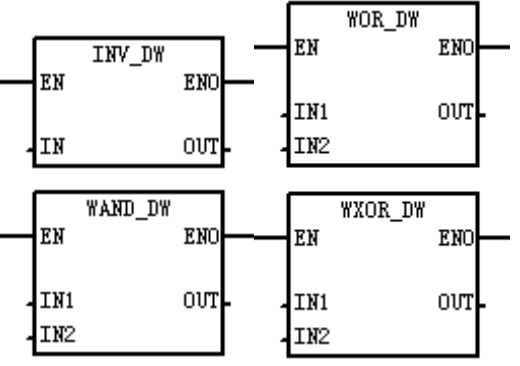
功能说明：

字节取反指令（INVB）对输入字节 IN 执行求补操作，并将结果放入 OUT 指定的地址中。

字节与运算指令（ANDB）对两个输入字节（IN1 和 IN2）的对应位执行与运算操作，并将结果放入 OUT 指定的地址中。

字节或运算指令（ORB）对两个输入字节（IN1 和 IN2）的对应位执行或运算操作，并将结果放入 OUT 指定的地址中。

字节异或运算指令（XORB）对两个输入字节（IN1 和 IN2）的对应位执行异或运算操作，并将结果放入 OUT 指定的地址中。

双字取反、与、或、异或运算指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	SM1.0 操作结果等于零
指令列表： INVD OUT ANDD IN1, OUT ORD IN1, OUT XORD IN1, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	双字	VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *AC, *LD

IN2	双字	VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *AC, *LD
OUT	双字	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD

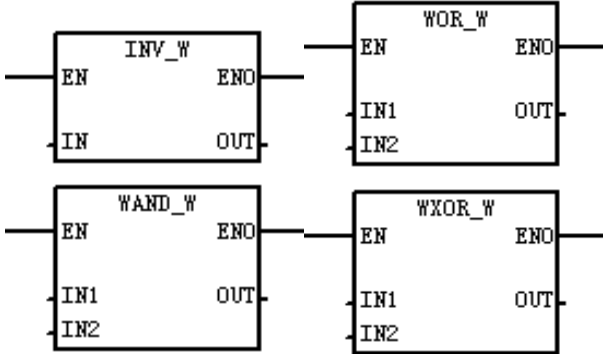
功能说明：

双字取反指令（INVD）对输入双字 IN 执行求补操作，并将结果放入 OUT 指定的地址中。

双字与运算指令（ANDD）对两个输入双字（IN1 和 IN2）的对应位执行与运算操作，并将结果放入 OUT 指定的地址中。

双字或运算指令（ORD）对两个输入双字（IN1 和 IN2）的对应位执行或运算操作，并将结果放入 OUT 指定的地址中。

双字异或运算指令（XORD）对两个输入双字（IN1 和 IN2）的对应位执行异或运算操作，并将结果放入 OUT 指定的地址中。

整数取反、与、或、异或运算指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	SM1.0 操作结果等于零
指令列表： INW OUT ANDW IN1, OUT ORW IN1, OUT XORW IN1, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN1	字	VW, IW, QW, MW, SW, SMW, T, C, AIW, LW, AC, 常数, *VD, *AC, *LD
IN2	字	VW, IW, QW, MW, SW, SMW, T, C, AIW, LW, AC, 常数, *VD, *AC, *LD
OUT	字	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD

功能说明：

字取反指令（INW）对输入字 IN 执行求补操作，并将结果放入 OUT 指定的地址中。

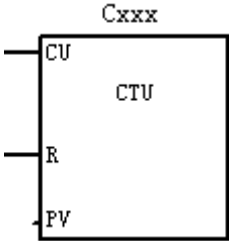
字与运算指令（ANDW）对两个输入字（IN1 和 IN2）的对应位执行与运算操作，并将结果放入 OUT 指定的地址中。

字或运算指令（ORW）对两个输入字（IN1 和 IN2）的对应位执行或运算操作，并将结果放入 OUT

指定的地址中。

字异或运算指令（XORW）对两个输入字（IN1 和 IN2）的对应位执行异或运算操作，并将结果放入 OUT 指定的地址中。

计数器指令

加计数器指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表：CTU Cxxx, PV		
梯形图： 		
输入输出参数	数据类型	适用操作数
Cxxx	字	常数 (C0~C255)
CU	布尔	使能位
R	布尔	使能位
PV	整数	VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, 常数, *VD, *AC, *LD, SW

功能说明：每次加计数输入端 CU 从 OFF 变为 ON 时，加计数器指令 CTU 从当前值计数加 1。当前值（Cxxx）大于或等于预设值（PV）时，计数器位（Cxxx）打开。复原（R）输入打开或执行“复原”指令时，计数器被复原。达到最大值（32767）时，计数器停止计数。计数器范围：Cxxx=C0 至 C255 在 STL 中，CTU 复原输入是堆栈顶值，加计数输入是装载在第二个堆栈位置的值。

<注释> 因为每个计数器有一个当前值，请勿将相同的计数器号码设置给一个以上计数器。（号码相同的加计数器、加 / 减计数器和减计数器存取相同的当前值。）

减计数器指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表：CTD Cxxx, PV		
梯形图：		

Cxxx CD CTD LD PV		
输入输出参数	数据类型	适用操作数
Cxxx	字	常数 (C0~C255)
CD	布尔	使能位
R	布尔	使能位
PV	整数	VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, 常数, *VD, *AC, *LD, SW

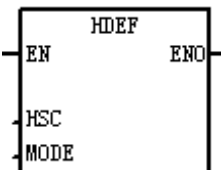
功能说明：每次减计数输入端 CD 从 OFF 变为 ON 时，减计数器指令 CTD 从当前值计数减 1。当前值 Cxxx 等于 0 时，计数器位（Cxxx）打开。载入输入（LD）打开时，计数器复原计数器位（Cxxx）并用预设值（PV）载入当前值。当前值达到零时，减计数器停止计数，计数器位 Cxxx 打开。计数器范围：Cxxx=C0 至 C255 在 STL 中，CTD 载入输入是堆栈顶值，而减计数输入是装载在第二个堆栈位置的数值。

<注释> 因为每个计数器有一个当前值，请勿将相同的计数器号码设置给一个以上计数器。（号码相同的加计数器、加 / 减计数器和减计数器存取相同的当前值）

加減双向计数器指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： CTUD Cxxx, PV		
梯形图： Cxxx CU CTUD CD R PV		
输入输出参数	数据类型	适用操作数
Cxxx	字	常数 (C0~C255)
CU	布尔	使能位
CD	布尔	使能位
R	布尔	使能位
PV	整数	VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, 常数, *VD, *AC, *LD, SW

功能说明：每次加计数输入端 CU 从 OFF 变为 ON 时，加减双向计数器指令 CTUD 从当前值计数加 1。每次减计数输入端 CD 从 OFF 变为 ON 时，加减双向计数器指令 CTUD 从当前值计数减 1。计数器的当前值 Cxxx 保持当前计数。每次执行计数器指令时，预设值 PV 与当前值进行比较。达到最大值（32767），位于加计数输入位置的下一个上升沿使当前值返转为最小值（-32,768）。在达到最小值（-32,768）时，位于减计数输入位置的下一个上升沿使当前计数返转为最大值（32,767）。当当前值 Cxxx 大于或等于预设值 PV 时，计数器位 Cxxx 打开。否则，计数器位关闭。当"复原"（R）输入打开或执行"复原"指令时，计数器被复原。达到 PV 时，CTUD 计数器停止计数。计数器范围：Cxxx=C0 至 C255 在 STL 中，CTUD 复原输入是堆栈顶值，减计数输入是装载在第二个堆栈位置的值，加计数输入是装载在第三个堆栈位置的值。

<注释> 因为每个计数器有一个当前值，请勿将相同的计数器号码设置给一个以上计数器。（号码相同的加计数器、加 / 减计数器和减计数器存取相同的当前值。）

高速计数器定义指令	设置 ENO = 0 的错误条件	0003 输入点冲突 0004 中断中的非法指令 000A HSC 重新定义
	影响的特殊内存位	无
指令列表： HDEF HSC, MODE		
梯形图： 		
输入输出参数	数据类型	适用操作数
HSC	字节	常数（0,1,2,3,4或5）
MODE	字节	常数（0,1,2,3,4,5,6,7,8,9,10或11）

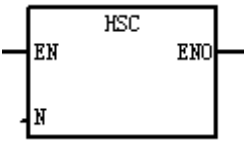
操作数说明：

HSC：高速计数器的 ID，目前 CPU224 和 CPU226 都支持 6 个高速计数器，因此 HSC 的范围为常数 0～5。

MODE：高速计数器的模式，高速计数器有 12 种模式，MODE 的范围为常数 0～11。

功能说明：高速计数器定义指令 HDEF 选择特定的高速计数器（HSCx）的操作模式。模式选择定义高速计数器的时钟、方向、起始和复原功能。您可以为每台高速计数器使用一条"高速计数器定义"指令。

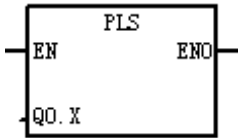
高速计数器指令	设置 ENO = 0 的错误条件	0001 HSC 在 HDEF 之前 0005 HSC/PLS 同步
	影响的特殊内存位	无
指令列表： HSC N		
梯形图：		

		
输入输出参数	数据类型	适用操作数
N	字	常数（0,1,2,3,4或5）

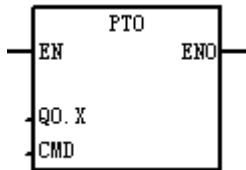
操作数说明：

N：高速计数器的 ID，目前 CPU224 和 CPU226 都支持 6 个高速计数器，因此 N 的范围为常数 0~5。

功能说明：高速计数器指令 HSC 根据 HSC 特殊内存位的状态配置和控制高速计数器。参数 N 指定高速计数器的号码。高速计数器最多可配置为十二种不同的操作模式。每台计数器在功能受支持的位置有专用时钟、方向控制、复原和起始输入。对于双相计数器，两个时钟均可按最高速度运行。在正交模式中，您可以选择一倍（1x）或四倍（4x）的最高计数速率。所有的计数器按最高速率运行，而不会相互干扰。

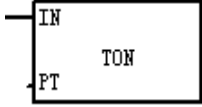
脉冲输出指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： PLS N		
梯形图： 		
输入输出参数	数据类型	适用操作数
N	字	常数0（Q0.0输出脉冲）或1(Q0.1输出脉冲)

功能说明：脉冲输出（PLS）指令被用于控制在高速输入（Q0.0 和 Q0.1）中提供的"脉冲串输出"（PTO）和 "脉宽调制"（PWM）功能。PTO 提供方波（50%占空比）输出，配备周期和脉冲数用户控制功能。PWM 提供连续性变量占空比输出，配备周期和脉宽用户控制功能。

脉冲输出指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： PTO N, CMD		
梯形图： 		
输入输出参数	数据类型	适用操作数
N	字节	常数0（Q0.0输出脉冲）或1(Q0.1输出脉冲)
CMD	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

功能说明：脉冲输出（PLS）指令被用于控制在高速输入（Q0.0 和 Q0.1）中提供的"脉冲串输出"（PTO）和"脉宽调制"（PWM）功能。PTO 提供方波（50% 占空比）输出，配备周期和脉冲数用户控制功能。PWM 提供连续性变量占空比输出，配备周期和脉宽用户控制功能。

定时器指令

接通延时定时器指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： TON Txxx, PT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	布尔	使能位
Txxx	字	常数(0~255)
PT	整数	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *LD, *AC

操作数说明：

定时器类型	分辨率	最大值	定时器 ID
TON/TOF	1ms	32767*1ms	T32, T96
	10ms	32767*10ms	T33-T36, T97-T100
	100ms	32767*100ms	T37-T63, T101-T255
TONR	1ms	32767*1ms	T0, T64
	10ms	32767*10ms	T1-T4, T65-T68
	100ms	32767*100ms	T5-T31, T69-T95

功能说明：接通延时定时器（TON）指令在启用输入为“打开”时，开始计时。当前值（Txxx）大于或等于预设时间（PT）时，定时器位为“打开”。启用输入为“关闭”时，接通延时定时器当前值被清除。达到预设值后，定时器仍继续计时，达到最大值 32767 时，停止计时。TON、TONR 和 TOF 定时器有三种分辨率。分辨率由定时器 ID 决定，详细情况见操作数说明里的表格。

LAD 和 FBD 定时器选择

- 1) 点击定时器号码域，然后键入定时器号码。
- 2) 倘若您键入的定时器号码无效，则时间基准值继续为"???"。
- 3) 将光标放在定时器框内稍等片刻，即可看到定时器工具提示。请查看此类定时器的有效号码列表。
- 4) 一旦键入有效定时器号码，时间基准值就会在定时器框内显示，例如"10 ms"。

<注释>

- TOF 及 TON 不能共享相同的定时器号码。例如，不能有 TON T32 和 TOF T32。
- 您可以将 TON 用于单间隔计时。
- 可用"复原"（R）指令复原任何定时器。"复原"指令执行下列操作：
- 定时器位 = 关闭，定时器当前值 = 0

示例程序：

STL：

ORGANIZATION_BLOCK 主程序：OB1

Network 1

```
LD      I0.0
TON     T33, +50    // 如果 I0.0 为打开状态，在 50*10ms 后定时器 T33 超时，定时器的
                  位为 1

                  // I0.0 打开 T33 启动开始计时，I0.0 关闭 T33 复位并且停止计时
```

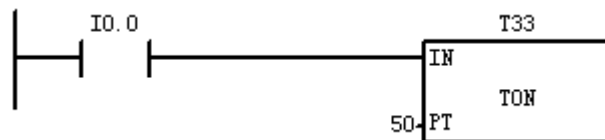
Network 2

```
LD      T33        // 如果 T33 位打开，则点亮 Q0.0，反之关闭 Q0.0
=       Q0.0
```

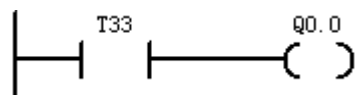
梯形图：

ORGANIZATION_BLOCK 主程序：OB1

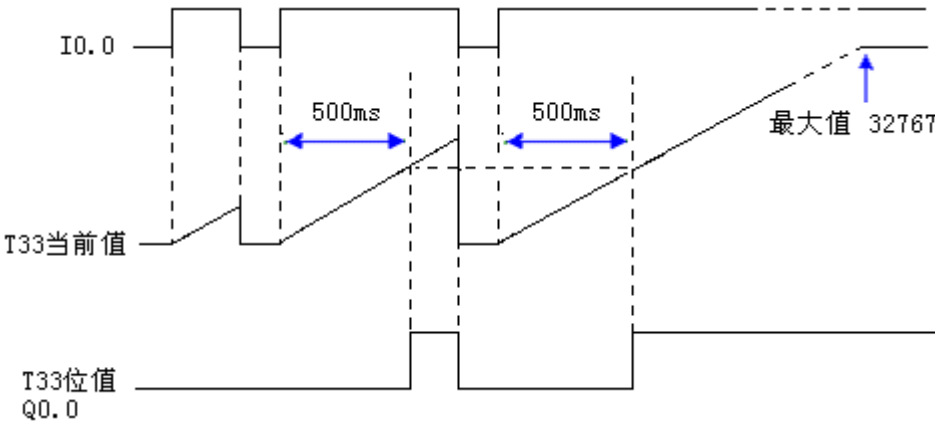
Network 1

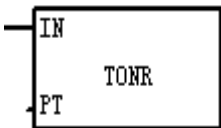


Network 2



时序图：



掉电保持接通延时	设置 ENO = 0 的错误条件	无
定时器指令	影响的特殊内存位	无
指令列表： TONR Txxx, PT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	布尔	使能位
Txxx	字	常数(0~255)
PT	整数	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *LD, *AC

操作数说明：

定时器类型	分辨率	最大值	定时器 ID
TON/TOF	1ms	32767*1ms	T32, T96
	10ms	32767*10ms	T33-T36, T97-T100
	100ms	32767*100ms	T37-T63, T101-T255
TONR	1ms	32767*1ms	T0, T64
	10ms	32767*10ms	T1-T4, T65-T68
	100ms	32767*100ms	T5-T31, T69-T95

功能说明：掉电保护性接通延时定时器（TONR）指令在启用输入为"打开"时，开始计时。当前值（Txxx）大于或等于预设时间（PT）时，计时位为"打开"。当输入为"关闭"时，保持保留性延迟定时器当前值。您可使用保留性接通延时定时器为多个输入"打开"阶段累计时间。使用"复原"指令（R）清除保留性延迟定时器的当前值。达到预设值后，定时器继续计时，达到最大值 32767 时，停止计时。TON、TONR 和 TOF 定时器有三种分辨率。分辨率由定时器 ID 决定，详细情况见操作数说明里的表格。

LAD 和 FBD 定时器选择

1) 点击定时器号码域，然后键入定时器号码。

- 2) 倘若您键入的定时器号码无效，则时间基准值继续为"???"。
- 3) 将光标放在定时器框内稍等片刻，即可看到定时器工具提示。请查看此类定时器的有效号码列表。
- 4) 一旦键入有效定时器号码，时间基准值就会在定时器框内显示，例如 "10 ms"。

<注释>

您可以将 TONR 用于累积多个计时间隔。

可用"复原"（R）指令复原任何定时器。"复原"指令执行下列操作：

定时器位 = 关闭，定时器当前值 = 0

只能用"复原"指令复原 TONR 定时器。

示例程序：

STL:

ORGANIZATION_BLOCK 主程序: OB1

Network 1

```
LD      I0.0
TONR    T0, +500    // 如果 I0.0 为打开状态，在 500*1ms 后定时器 T0 超时，定时器的
位为 1

                        // I0.0 打开 T0 启动开始计时，I0.0 关闭定时器当前值保持，下次 I0.0
打开累加
```

Network 2

```
LD      T0          // 如果 T0 位打开，则点亮 Q0.0，反之关闭 Q0.0
=       Q0.0
```

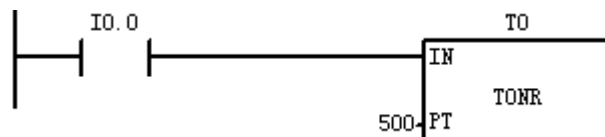
Network 3

```
LD      I1.0        // I1.0 打开时 T0 复位
R       T0, 1
```

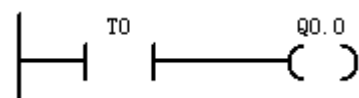
梯形图：

ORGANIZATION_BLOCK 主程序: OB1

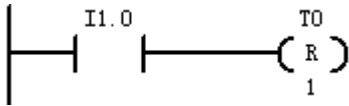
Network 1



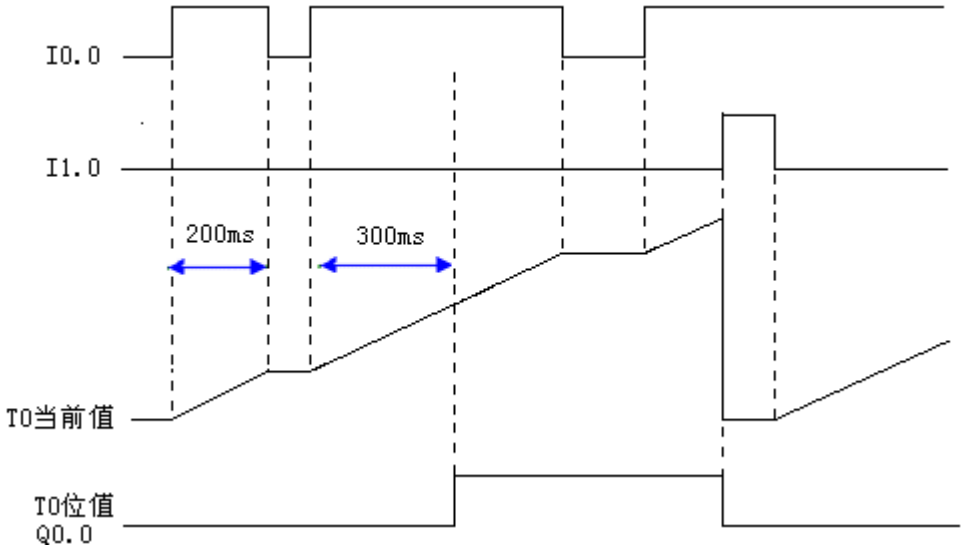
Network 2



Network 3



时序图:



断开延时定时器指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表: TOF Txxx, PT		
梯形图:		
输入输出参数	数据类型	适用操作数
IN	布尔	使能位
Txxx	字	常数(T0~T255)
PT	整数	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *LD, *AC

操作数说明:

定时器类型	分辨率	最大值	定时器 ID
TON/TOF	1ms	32767*1ms	T32, T96
	10ms	32767*10ms	T33-T36, T97-T100
	100ms	32767*100ms	T37-T63, T101-T255
TONR	1ms	32767*1ms	T0, T64
	10ms	32767*10ms	T1-T4, T65-T68
	100ms	32767*100ms	T5-T31, T69-T95

功能说明: 断开延时定时器 (TOF) 用于在输入关闭后, 延迟固定的一段时间再关闭输出。启用输入打开时, 定时器位立即打开, 当前值被设为 0。输入关闭时, 定时器继续计时, 直到消逝

的时间达到预设时间。达到预设值后，定时器位关闭，当前值停止计时。如果输入关闭的时间短于预设数值，则定时器位仍保持在打开状态。**TOF** 指令必须遇到从"打开"至"关闭"的转换才开始计时。如果 **TOF** 定时器位于 **SCR** 区域内部，而且 **SCR** 区域处于非现用状态，则当前值被设为 0，计时器位被关闭，而且当前值不计。TON、TONR 和 TOF 定时器有三种分辨率。分辨率由定时器 ID 决定，详细情况见操作数说明里的表格。

LAD 和 FBD 定时器选择

- 1) 点击定时器号码域，然后键入定时器号码。
- 2) 倘若您键入的定时器号码无效，则时间基准值继续为"???"。
- 3) 将光标放在定时器框内稍等片刻，即可看到定时器工具提示。请查看此类定时器的有效号码列表。
- 4) 一旦键入有效定时器号码，时间基准值就会在定时器框内显示，例如"10 ms"。

<注释>

TOF 及 TON 不能共享相同的定时器号码。例如，不能有 TON T32 和 TOF T32。

您可以将 **TOF** 用于延长时间以超过关闭（或假）条件，例如在电机关闭后使电机冷却。

可用"复原"（R）指令复原任何定时器。"复原"指令执行下列操作：

定时器位 = 关闭，定时器当前值 = 0

复原后，TOF 定时器要求启用输入从"打开"转换为"关闭"，以便重新启动。

示例程序：

STL:

ORGANIZATION_BLOCK 主程序: OB1

Network 1

LD I0.0

TOF T32, +500 // 如果 I0.0 为打开状态，在 500*1ms 后定时器 T32 超时，定时器的位为 0

// I0.0 下降沿 T32 启动开始计时，I0.0 上升沿 T32 复位并且停止计时

Network 2

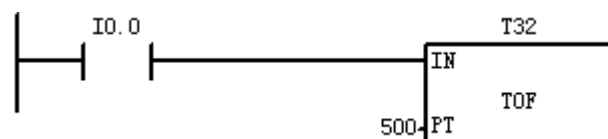
LD T32 // 如果 T32 位打开，则点亮 Q0.0，反之关闭 Q0.0

= Q0.0

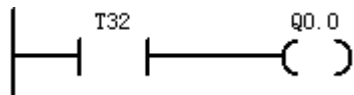
梯形图:

ORGANIZATION_BLOCK 主程序: OB1

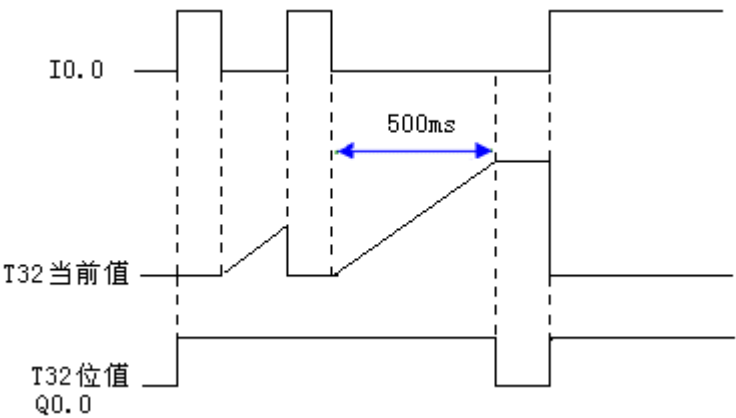
Network 1



Network 2



时序图:



开始间隔时间指令	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： BITIM OUT		
梯形图：		
输入输出参数	数据类型	适用操作数
OUT	双字	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

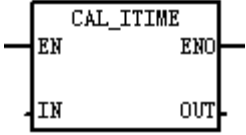
功能说明：读取内置 1 毫秒计数器的当前值，并将该值存储于 OUT。双字毫秒值的最大计时间隔为 2 的 32 次方，即 49.7 日。

取当前微秒值指令	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： R_UTIM OUT		
梯形图：		
输入输出参数	数据类型	适用操作数
OUT	双字	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

功能说明：读取内置 1 毫秒计数器的当前值，并将该值存储于 OUT，单位为毫秒，双字毫秒值

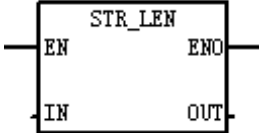
的最大计时间隔为 2 的 32 次方，即 49.7 日。

使用示例：R_UTIM VDO

计算间隔时间指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表：CITIM IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	双字	VD, ID, QD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC
OUT	双字	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

功能说明：计算当前时间与 IN 所提供时间的时差，将该时差存储于 OUT。双字毫秒值的最大计时间隔为 2 的 32 次方，即 49.7 日。取决于 BGN_ITIME 指令的执行时间，CAL_ITIME 指令将自动处理发生在最大间隔内的一毫秒定时器翻转。

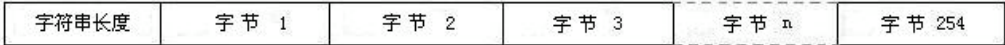
字符串指令

字符串长度指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	0091 操作数范围
指令列表：SLEN IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

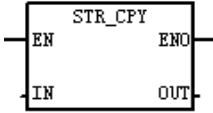
ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节)：



功能说明：字符串长度指令返回 IN 指定的字符串长度。常数字符串参数最长为 255 个字节。

复制字符串指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围
	影响的特殊内存位	无
指令列表： SCPY IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	字符串	VB, *VD, LB, *LD, *AC

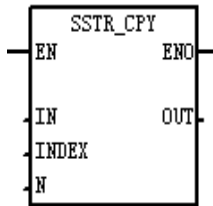
ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节)：



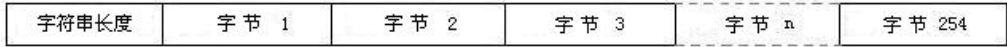
功能说明：复制字符串指令将 IN 指定的字符串复制至 OUT 指定的字符串。常数字符串参数最长为 255 个字节。

从字符串复制子字符串指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 009B 非法索引（指定起始位置0的字符串操作）
	影响的特殊内存位	无
指令列表： SSCPY IN, INDEX, N, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
INDEX	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC
N	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC
OUT	字符串	VB, *VD, LB, *LD, *AC

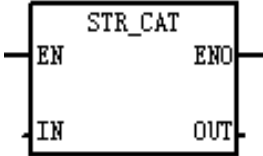
ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节)：



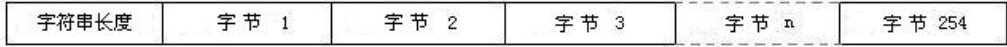
功能说明：从字符串复制子字符串指令将（从索引（INDEX）开始）IN 指定的具体字符串数目复制至 OUT 指定的字符串。

字符串连接指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数范围
	影响的特殊内存位	无
指令列表： SCAT IN, OUT		
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	字符串	VB, *VD, LB, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束 ("字符串常数")。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式(一个方格代表一个字节)：

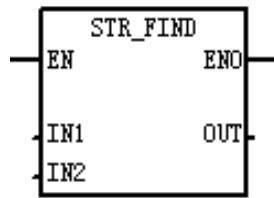


功能说明：字符串连接指令将 IN 指定的字符串附加至 OUT 指定的字符串之后。常数字符串参数最长为 255 个字节。

查找字符串指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数范围 009B 非法索引（指定起始位置0的字符串操作）
	影响的特殊内存位	无

指令列表： SFND IN1, IN2, OUT

梯形图：

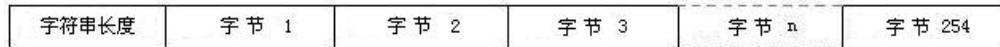


输入输出参数	数据类型	适用操作数
IN1	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
IN2	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束（“字符串常数”）。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型的格式（一个方格代表一个字节）：

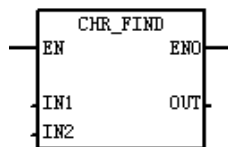


功能说明：在字符串内查找字符串指令在字符串 IN1 中搜索首次出现的字符串 IN2。搜索从 OUT 起始位置开始。如果找到一个与字符串 IN2 完全符合的字符系列，该系列的第一个字符位置被写入 OUT。如果在字符串 IN1 中未找到字符串 IN2，OUT 被设为 0。单个常数字符串最长为 126 个字节，两个常数字符串综合最长为 240 个字节。

在字符串内查找字符串指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数范围 009B 非法索引（指定起始位置0的字符串操作）
	影响的特殊内存位	无

指令列表： CFND IN1, IN2, OUT

梯形图：



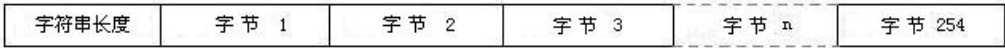
输入输出参数	数据类型	适用操作数
IN1	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
IN2	字符串	VB, 常数字符串, LB, *VD, *LD, *AC
OUT	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

ASCII 常数字符串数据类型的格式：

字符串是一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么

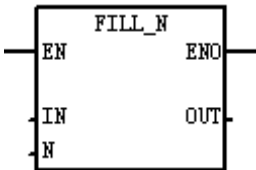
该字符串必须用双引号字符起始和结束（"字符串常数"）。

字符串的最大长度是 255 个字节（254 个字符加上长度字节）。下面的内存图显示了字符串数据类型格式（一个方格代表一个字节）：



功能说明：在字符串中查找字符指令在字符串 IN1 中搜索字符串 IN2 中描述的字符集中的任何字符首次出现的位置。搜索从起始位置 OUT 开始。如果找到一个相符的字符，该字符位置被写入 OUT。如果未找到相符的字符，OUT 被设为 0。 单个常数字符串最长为 126 个字节，两个常数字符串综合最长为 240 个字节。

表指令

内存填充指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数超出范围
	影响的特殊内存位	无
	指令列表： FILL IN, OUT, N	
梯形图： 		
输入输出参数	数据类型	适用操作数
IN	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *LD, *AC
N	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC
OUT	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC

功能说明：内存填充（FILL）指令用包含在地址 IN 中的字值写入 N 个连续字，从地址 OUT 开始。N 的范围是 1 至 255。

先进先出指令	设置 ENO = 0的错误条件	0006 间接地址 0091 操作数超出范围 SM1.5 空表
	影响的特殊内存位	SM1.5 空表
指令列表： FIFO TBL, DATA		
梯形图：		

输入输出参数	数据类型	适用操作数
TBL	字	VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *LD, *AC
DATA	整数	VW, IW, QW, MW, SW, SMW, LW, AC, T, C, AQW, *VD, *LD, *AC

功能说明：先入先出（FIFO）指令通过移除表格（TBL）中的第一个条目，并将数值移至 DATA 指定位置的方法，移动表格中的最早（或第一个）条目。表格中的所有其他条目均向上移动一个位置。每次执行指令时，表格中的条目数减 1。

提示：欲建立表格，首先为最大表条目数建立一个条目。如果您没有这样做，则无法在表格中建立任何条目。此外，所有的表格读取和表格写入指令必须用边缘触发器指令激活。

后进先出指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数超出范围 SM1.5 空表
	影响的特殊内存位	SM1.5 空表

指令列表： LIFO TBL, DATA

梯形图：



输入输出参数	数据类型	适用操作数
TBL	字	VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *LD, *AC
DATA	整数	VW, IW, QW, MW, SW, SMW, LW, AC, T, C, AQW, *VD, *LD, *AC

功能说明：后入先出（LIFO）指令将表格中的最新（或最后）一个条目移至输出内存地址，方法是移除表格（TBL）中的最后一个条目，并将数值移至 DATA 指定的位置。每次执行指令时，表格中的条目数减 1。

提示：欲建立表格，首先为最大表条目数建立一个条目。如果您没有这样做，则无法在表格中建立任何条目。此外，所有的表格读取和表格写入指令必须用边缘触发器指令激活。

表格查找指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数超出范围
	影响的特殊内存位	无

指令列表：

FND= TBL, PIN, INDEX
FND<> TBL, PIN, INDEX
FND< TBL, PIN, INDEX
FND> TBL, PIN, INDEX

梯形图：

TBL_FIND= EN ENO TBL PTN INDEX TBL_FIND> EN ENO TBL PTN INDEX TBL_FIND<> EN ENO TBL PTN INDEX TBL_FIND< EN ENO TBL PTN INDEX		
输入输出参数	数据类型	适用操作数
TBL	字	VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *LD, *AC
PIN	整数	VW, IW, QW, MW, SW, SMW, AIW, LW, T, C, AC, 常数, *VD, *LD, *AC
INDEX	字	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC

功能说明：表格查找（TBL）指令在表格（TBL）中搜索与某些标准相符的数据。"表格查找"指令搜索表，从 INDEX 指定的表格条目开始，寻找与 CMD 定义的搜索标准相匹配的数据数值（PTN）。命令参数（CMD）被指定一个 1 至 4 的数值，分别代表 =、<>、<, and >。如果找到匹配条目，则 INDEX 指向表格中的匹配条目。欲查找下一个匹配条目，再次激活"表格查找"指令之前必须在 INDEX 上加 1。如果未找到匹配条目，INDEX 的数值等于条目计数。一个表格最多可有 100 个条目，数据项目（搜索区域）从 0 排号至最大值 99。

<注释> 当您在用 ATT、LIFO 和 FIFO 指令生成的表格中使用"表格查找"指令时，条目计数与数据条目直接对应。"表格查找"指令并不要求 ATT、LIFO 和 FIFO 所要求的最大条目数。因此，如下所示，您应当将"查找"指令的 SRC 操作数设为一个高于对应的"增加至表格"、"后入先出"或"先入先出"指令 TBL 操作数的一个字地址（两个字节）。

提示：欲建立表格，首先为最大表条目数建立一个条目。如果您没有这样做，则无法在表格中建立任何条目。此外，所有的表格读取和表格写入指令必须用边缘触发器指令激活。

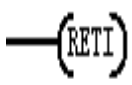
追加至表格指令	设置 ENO = 0 的错误条件	0006 间接地址 0091 操作数超出范围 SM1.4 表溢出
	影响的特殊内存位	SM1.4 表溢出
指令列表： ATT DATA, TABLE		
梯形图： AD_T_TBL EN ENO TBL DATA		

输入输出参数	数据类型	适用操作数
DATA	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *LD, *AC
TBL	字	VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, , *LD *AC

功能说明：增加至表格（ATT）指令向表格（TBL）中加入字值（DATA）。表格中的第一个数值是表格的最大长度（TL）。第二个数值是条目计数（EC），指定表格中的条目数。新数据被增加至表格中的最后一个条目之后。每次向表格中增加新数据后，条目计数加 1。表格最多可包含 100 个条目，不包括指定最大条目数和实际条目数的参数。

提示：欲建立表格，首先为最大表条目数建立一个条目。如果您没有这样做，则无法在表格中建立任何条目。此外，所有的表格读取和表格写入指令必须用边缘触发器指令激活。

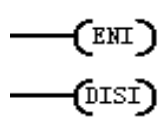
中断指令

从中断程序有条件返回指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： CRETl		
梯形图： 		
输入输出参数	数据类型	适用操作数
无	无	无

功能说明：从中断指令有条件返回（CRETI）指令可根据先前逻辑条件用于从中断返回。欲增加中断程序，在 Project Manager 主界面点菜单编辑(Edit)>插入(Insert)>Organization Block, 。

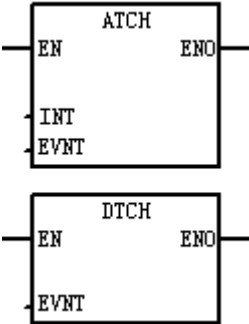
<注释> MagicWorks PLC 自动为每个中断程序增加一个无条件返回。

您应当优化中断程序，执行具体任务，然后使控制返回主例程序。保持中断程序简明扼要，可加快执行速度，并且不会长时间延迟其他过程。否则，无法预测的条件会引起主程序控制的装置操作异常。限制在中断程序中不能使用 DISI、ENI、HDEF、LSCR、END 指令。

中断使能和禁止指令	设置 ENO = 0 的错误条件	0004 尝试在中断程序中执行 ENI、DISI 或 HDEF 指令
	影响的特殊内存位	无
指令列表： ENI DISI		
梯形图： 		
输入输出参数	数据类型	适用操作数

无	无	无

功能说明：中断允许（ENI）指令全局性启用所有附加中断事件进程。中断禁止（DISI）指令全局性禁止所有中断事件进程。转换至 RUN（运行）模式时，中断开始时被禁止。一旦进入 RUN（运行）模式，您可以通过执行全局中断允许指令，启用所有中断进程。执行中断禁止指令会禁止处理中断；但是现用中断事件将继续入队等候。

中断连接与分离指令	设置 ENO = 0的错误条件	0002 配置对 HSC 的输入赋值
	影响的特殊内存位	无
指令列表： ATCH INT, EVENT DTCH EVENT		
梯形图： 		
输入输出参数	数据类型	适用操作数
INT	字节	OBx(x 为常数 2-127)
EVENT	字节	常数 0-33

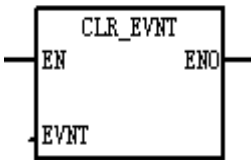
功能说明：

中断连接（ATCH）指令将中断事件（EVNT）与中断程序号码（INT）相联系，并启用中断事件。

中断分离（DTCH）指令取消中断事件（EVNT）与所有中断程序之间的关联，并禁用中断事件。

在激活中断程序之前，必须在中断事件和您希望在事件发生时执行的程序段之间建立关联。使用“中断连接”指令将中断事件（由中断事件号码指定）与程序段（由中断程序号码指定）联系在一起。您可以将多个中断事件附加在一个中断程序上，但一个事件不能同时附加在多个中断程序上。当您将一个中断事件附加在一个中断程序上是，会自动启用中断。如果您用全局禁用中断指令禁用所有的中断，则每次出现的中断事件均入队等候，直至使用全局启用中断指令或中断队列溢出重新启用中断。您可以使用“中断分离”指令断开中断事件与中断程序之间的关联，从而禁用单个中断事件。“中断分离”指令使中断返回至非现用或忽略状态。

清除中断事件指令	设置 ENO = 0的错误条件	无
	影响的特殊内存位	无
指令列表： CEVNT EVENT		
梯形图：		



输入输出参数	数据类型	适用操作数
EVENT	字节	常数 0-33

功能说明：清除中断事件指令会删除中断队列中所有类型为 **EVNT** 的中断事件。此指令用于清除不必要的中断，后者可能由假传感器输出暂态造成。

注：请参考附录 [E 中断事件优先级列表](#)。

程序控制指令

循环指令	设置 ENO = 0 的错误条件	0006 间接地址
	影响的特殊内存位	无
指令列表： FOR INDEX, INIT, FINAL NEXT		
梯形图： The diagram shows a rectangular instruction symbol for the FOR/NEXT loop. It has an input terminal 'EN' on the left and an output terminal 'ENO' on the right. Below the rectangle, there are four input terminals labeled 'INDEX', 'INIT', and 'FINAL'. Below the rectangle, there is a label '(NEXT)' with a small arrow pointing to the bottom of the box.		
输入输出参数	数据类型	适用操作数
INDEX	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC
INIT	整数	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, 常数, *VD, *LD, *AC
FINAL	整数	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, 常数, *VD, *LD, *AC

功能说明：FOR（FOR）指令执行 FOR 和 NEXT 之间的指令。您必须指定索引值或当前循环计数（INDEX）、起始值（INIT）和结束值（FINAL）。NEXT（NEXT）指令标记 FOR 循环结束，并将堆栈顶值设为 1。使用 FOR/NEXT 指令描述为指定计数重复的循环。每条 FOR 指令要求一个 NEXT 指令。您可以复原 FOR/NEXT 循环（在 FOR/NEXT 循环中放置一个 FOR/NEXT 循环），深度可达八。例如，假定 INIT 值等于 1，FINAL 值等于 10，FOR 与 NEXT 之间的指令被执行 10 次，INDEX 值递增：1、2、3、...10。如果起始值大于结束值，则不执行循环。每次执行 FOR 和 NEXT 之间的指令后，INDEX 值递增，并将结果与结束值比较。如果 INDEX 大于结束值，循环则终止。

<注释>

- 如果您启用 **FOR/NEXT** 循环，则将继续循环程序，直至结束反复操作，除非您从循环内部改变结束值。您可以在 **FOR/NEXT** 处于循环过程时改变数值。
- 再次启用循环时，它将初始值复制至索引值（当前循环次数）。下次被启用时，**FOR/NEXT** 指令复原。

跳转指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： JMP n LBL n		
梯形图： 		
输入输出参数	数据类型	适用操作数
n	字节	常数 0~255

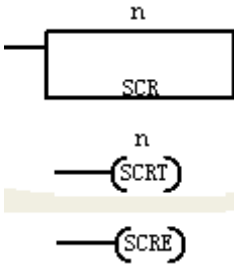
功能说明：

跳转至标签（**JMP**）指令对程序中的指定标签（**n**）执行分支操作。跳转接受时，堆栈顶值始终为逻辑 1。

标签（**LBL**）指令标记跳转目的地（**n**）的位置。

您可以在主程序、子程序或中断程序中使用"跳转"指令。"跳转"及其对应的"标签"指令必须始终位于相同的代码段中（主程序、子程序或中断程序）。您不能从主程序跳转至子程序或中断程序中的标签，与此相似，您也不能从子程序或中断程序跳转至该子程序或中断程序之外的标签。您可以在 **SCR** 段中使用"跳转"指令，但对应的"标签"指令必须位于相同的 **SCR** 段内。

顺序控制继电器指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： LSCR n SCRT n SCRE CSCRE		
梯形图：		

		
输入输出参数	数据类型	适用操作数
n	布尔	S

功能说明：SCR 指令为您提供一种可自然纳入 LAD、FBD 或 STL 程序的简单、强有力的状态控制编程技术。每当应用程序包含一系列必须重复执行的操作时，SCR 可用于为程序安排结构，以便使之直接与应用程序相对应。因而您能够更快速、更方便地编程和调试应用程序。载入顺序控制继电器（LSCR）指令用指令（N）引用的 S 位数值载入 SCR 和逻辑堆栈。SCR 段被 SCR 堆栈的结果数值激励或取消激励。SCR 堆栈数值被复制至逻辑堆栈的顶端，以便方框或输出线圈可直接与左电源杆连接，无须插入触点。顺序控制继电器转换（SCRT）指令识别要启用的 SCR 位（下一个要设置的 n 位）。当使能位进入线圈或 FBD 方框时，打开引用 n 位，并关闭 LSCR 指令（启用该 SCR 段）的 n 位顺序控制继电器结束（SCRE）指令标记 SCR 段的结束。一旦将电源应用于输入，有条件顺序控制继电器结束（CSCRE）指令即标记 SCR 段结束。CSCRE 只有在 STL 编辑器中才能使用。

<注释> CSCRE 只有在第二代（22x）CPU（从 1.20 版开始）才能使用。

下列说明适用于顺序控制继电器指令。

- “载入 SCR”指令（LSCR）标记 SCR 段的开始，“SCR 结束”指令（SCRE）标记 SCR 段的结束。“载入 SCR”和“SCR 结束”指令之间的所有逻辑执行取决于 S 堆栈数值。“SCR 结束”和下一条“载入 SCR”指令之间的逻辑不取决于 S 堆栈数值。
- “SCR 转换”指令（SCRT）提供一种从现用 SCR 段向另一个 SCR 段转换控制的方法。当“SCR 转换”指令有使能位时，该指令会复原当前现用段的 S 位，并设置被引用段的 S 位。在“SCR 转换指令执行时，复原现用段的 S 位不会影响 S 堆栈。因此，SCR 段在退出前保持激励状态。
- “有条件 SCR 结束”指令（CSCRE）提供一种无须执行“有条件 SCR 结束”和“SCR 结束”指令之间的指令即可退出现用 SCR 段的方法。“有条件 SCR 结束”指令不会影响任何 S 位，亦不会影响 S 堆栈。

使用 SCR 的限制

- 您不能在一个以上例行程序中使用相同的 S 位。例如，如果您在主程序中使用 S0.1，则不能在子程序中再使用。
- 您不能在 SCR 段中使用 JMP 和 LBL 指令。这表示不允许跳转入或跳转出 SCR 段，亦不允许在 SCR 段内跳转。您可以使用跳转和标签指令在 SCR 段周围跳转。
- 您不能在 SCR 段中使用“结束”指令。

用 SCR 段控制程序流

CTSC-200 PLC 中的主程序包含每次 PLC 扫描即按顺序执行一次的指令。对于很多应用程序，

可能需要以逻辑方式将主程序分为一系列操作步骤，以便反映控制进程中的步骤（例如，一系列机器操作）。

以逻辑方式将程序分为多个步骤的一种方法是使用 **SCR** 段。通过使用 **SCR** 段，可以将程序分为一个连续步骤流，或分为可以同时现用的多个步骤流。可以将一个步骤流有条件地分为多个步骤流，或将多个步骤流有条件地重新组合为一个步骤流。

看门狗复位指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表: WDR		
梯形图: 		
输入输出参数	数据类型	适用操作数
无	无	无

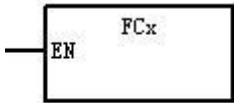
功能说明:

看门狗复原（WDR）指令重新触发 CTSC-200 CPU 的系统监视程序定时器，扩展扫描允许使用的时间，而不会出现看门狗错误。使用“看门狗复原”指令时应当小心。如果使用循环指令阻止扫描完成或严重延迟扫描完成，下列程序只有在扫描周期完成后才能执行：

- 通讯（自由端口模式除外）
- I/O 更新（立即 I/O 除外）
- 强迫更新
- SM 位更新（不更新 SM0、SM5 至 SM29）
- 运行时间诊断程序
- 10 毫秒和 100 毫秒定时器对于超过 25 秒的扫描不能正确地累计时间
- 用于中断程序时的 STOP（停止）指令
- 配备离散输出的扩充模块还包括看门狗定时器，如果模块未被 CTSC-200 写入，监视程序定时器会关闭输出。对每个配备离散输出的扩充模块使用立即写入，在扩展扫描时间期间使正确的输出保持打开。

<注释> 如果您预计扫描时间将超过 500 毫秒，或者您预计会发生大量中断活动，可能阻止返回主扫描超过 500 毫秒，您应当使用 WDR 指令，重新触发看门狗定时器。每次使用“看门狗复原”指令时，您还应当使用对每个离散扩充模块中的一个输出字节（QB）使用立即写入，复原每个扩充模块看门狗。如果您使用“看门狗复原”指令允许执行要求很长扫描时间的程序，将模式开关改变为 STOP（停止）位置会使 CTSC-200 在 1.4 秒内转换为 STOP（停止）模式。

调用	设置 ENO = 0 的错误条件	0008 功能嵌套数超过最大
	影响的特殊内存位	无
指令列表: CALL Function Name, parameter 1, ... parameter n (最多允许16个参数)		
梯形图:		

		
输入输出参数	数据类型	适用操作数
FCx	功能	功能(x = 0 - 127)
使能位	布尔	使能位
输入参数	布尔, 字节, 字, 整数, 双字, 双整数, 字符串	V, I, Q, M, SM, S, T, C, L, VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, VW, T, C, IW, QW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC, VD, ID, QD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ, &SMB, *VD, *LD, *AC, constant
输入 / 输出参数	布尔, 字节, 字, 整数, 双字, 双整数	V, I, Q, M, SM, S, T, C, L, VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, VW, T, C, IW, QW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC, VD, ID, QD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC
输出参数	布尔, 字节, 字, 整数, 双字, 双整数	V, I, Q, M, SM, S, T, C, L, VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, VW, T, C, IW, QW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC, VD, ID, QD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC

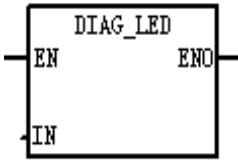
功能说明：调用功能（CALL）指令将控制转换给功能（FCx）。您可以使用带参数或不带参数的“调用功能”指令。在功能完成执行后。控制返回至“调用功能”之后的指令。每个功能调用的输入 / 输出参数最大限制为 16。如果您尝试下载的程序超过此一限制，会返回一则错误信息。如果您为功能指定一个符号名，例如 USR_NAME，该符号名会出现在指令树的“功能”文件夹中。

将参数值指定给功能中的局部内存时应遵守下列规则：

- 1) 参数值指定给局部内存的顺序由 CALL 指定，参数从 L.0 开始。
- 2) 一至八个连续位参数值被指定给从 Lx.0 开始持续至 Lx.7 的单字节。
- 3) 字节、字和双字数值被指定给局部内存，位于字节边界（LBx、LWx 或 LDx）位置。在带参数的“调用功能”指令中，参数个数、类型与顺序必须与功能局部变量表中定义的变量完全匹配。参数顺序必须以输入参数开始，其次是输入 / 输出参数，然后是输出参数。

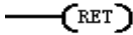
<注释> 位于指令树中的功能名称的工具提示显示每个参数的名称。

诊断 LED 指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表: DLED IN		
梯形图:		

		
输入输出参数	数据类型	适用操作数
IN	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC

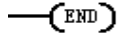
功能说明：

如果输入参数 IN 的数值为零，则诊断 LED 会被设置为不发光。如果输入参数 IN 的数值大于零，则诊断 LED 会被设置为发光（黄色）。标记为 SF/ DIAG 的 CPU 发光二极管（LED）能够配置为：当在系统块内指定的条件为真或当 DIAG_LED 指令以非零 IN 参数得到执行时发黄光。系统块（配置 LED）复选框选项：1) SF/ DIAG LED 会在 CPU 中的某个项被强制时发黄光 2) SF/ DIAG LED 会在某模块有 I/O 错误时发黄光取消对两个“配置 LED”选项的复选，就会让 DIAG_LED 指令独自控制 SF/ DIAG 黄光。CPU 系统故障（SF）用红光表示。

从子程序有条件返回指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： CRET		
梯形图： 		
输入输出参数	数据类型	适用操作数

功能说明：

从子程序有条件返回（CRET）指令根据前一个逻辑终止子程序。从子程序有条件返回指令是供选用指令。MagicWorks PLC 会自动增加要求使用的从子程序无条件返回指令，且不在程序编辑器的“子程序 POU”标记显示的程序逻辑中显示。

有条件结束指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无
指令列表： END		
梯形图： 		
输入输出参数	数据类型	适用操作数

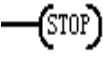
功能说明：

有条件结束（END）指令根据前一个逻辑条件终止主用户程序。

<注释> 您可以在主程序中使用“有条件结束”指令，但不能在子程序或中断程序中使用。MagicWorks PLC 自动在主用户程序中增加无条件结束。

有条件停机指令	设置 ENO = 0 的错误条件	无
	影响的特殊内存位	无

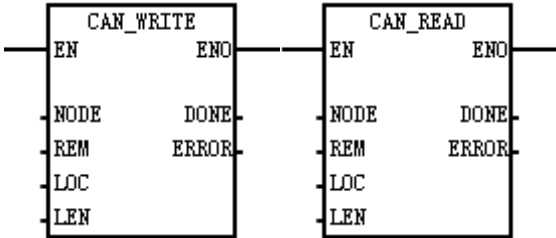
指令列表： STOP

梯形图：

输入输出参数	数据类型	适用操作数

功能说明：在条件满将 CPU 切换至 STOP 状态。

通信指令

CANopen SDO 读写指令	设置 ENO = 0的错误条件	0006 间接地址
	影响的特殊内存位	SMB500~SMB532(显示 CANopen 通信的状态信息)
指令列表： CANW NODE,REM,LOC,LEN,DONE,ERROR CANR NODE,REM,LOC,LEN,DONE,ERROR		
梯形图： 		
输入输出参数	数据类型	适用操作数
NODE	字节	常数（1~127）
REM	字节	VB, *VD, *LD, *AC
LOC	字节	VB, *VD, *LD, *AC
LEN	字节	常数（1~255）
DONE	布尔	Q, M, V
ERROR	双字	VD, QD, MD, AC, *VD, *LD, *AC

SDO 错误信息说明：

错误码	说明
0x00000000	指令执行无错误
0x00000001	主站不处于 Operational 状态，因此不能发送
0x00000002	目标节点故障，因此不发送
0x00000003	目标节点不存在，因此不发送
0x00000004	上一次发送的相同命令尚未得到回应
0x00000005	用户输入的指令参数有错误
0x00000006	指令缓冲区已满
0x00000007	超时没有收到回应
0x00000008	收到的回应报文的数据长度有错误
0x00000009	收到的回应报文不是期望的回应报文

功能说明：

CANopen SDO 通信指令（CANW 和 CANR）是用于实现 CANopen 通信的 SDO 传输。

CANW 将本地 LOC 开始长度为 LEN 字节的 V 内存的值写到 NODE ID 为 NODE 的远程站中，远程站中的写入地址为 REM 开始长度为 LEN 的 V 内存。

CANR 是读入 NODE ID 为 NODE 的远程站的一块 V 内存的值，读取数据的来源为 REM 开始长度为 LEN 的 V 内存，读取数据在本地的存放地址为 LOC 开始长度为 LEN 的 V 内存。

DONE 是完成标志位，读写未完成时为 0，每次读写完成后 DONE 被置 1。

ERROR 是错误标志，用于显示错误信息，ERROR=0 时表示无错误。

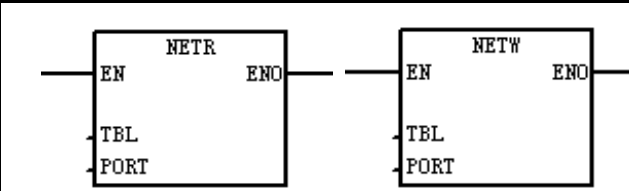
读取设置端口地址指令	设置 ENO = 0 的错误条件	0006 间接地址 0004 尝试在中断程序中执行 SPA(SPA)
	影响的特殊内存位	无
指令列表： GPA ADDR, PORT SPA ADDR, PORT		
梯形图： GET_ADDR EN ENO ADDR PORT SET_ADDR EN ENO ADDR PORT		
输入输出参数	数据类型	适用操作数
ADDR(GPA)	字节	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC
ADDR(SPA)	字节	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC
PORT	字节	常数（0用于 CPU224； 0或1用于 CPU 226/226M/226L）

功能说明：

获得端口地址指令（GPA）读取端口号为 PORT 的通信端口站地址，并将数值置于 ADDR 中指定的地址内。

设置端口地址指令（SPA）将端口号为 PORT 的通信端口的站地址设为 ADDR 中指定的数值，设定的新地址不会被永久性保存。电源重新上电后，设置的端口会返回至最后的地址（系统块里设置并下载的地址）。

网络读写指令	设置 ENO = 0 的错误条件	0006 间接地址 表状态字节 E 位为1
	影响的特殊内存位	无
指令列表： NETR TABLE, PORT NETW TABLE, PORT		
梯形图：		

		
输入输出参数	数据类型	适用操作数
TBL	字节	VB, MB, *VD, *LD, *AC
PORT	字节	常数（0用于 CPU224； 0或1用于 CPU 226/226M/226L）

功能说明：

网络读取指令（NETR）开始一项通讯操作，通过指定的端口 PORT 根据表格 TBL 定义从远程设备收集数据。

网络写入指令（NETW）开始一项通讯操作，通过指定的端口 PORT 根据表格 TBL 定义向远程设备写入数据。

网络读写指令最多可从远程站读取或往远程站写入 16 字节信息，在程序中允许有任意数目的 NETR/NETW 指令，但在任何时间最多只能有 8 条被激活，即激活的 NETR 条数加 NETW 条数之和最多为 8 条。

用户还可以使用“网络读取 / 网络写入指令向导”配置通信程序。选择菜单命令“工具”->“NETR/NETW 指令向导”即可启动此向导。

TBL 参数定义

n7	n6	n5	n4	n3	n2	n1	n0
远程站地址							
指针指向							
数据区域							
远程站							
(I/Q/M/V)							
数据长度							
数据字节0							
数据字节1							
...							
数据字节15							

n7: 完成（功能完成） 0 = 未完成 1 = 完成

n6: 现用（功能入队） 0 = 非现用 1 = 现用

n5: 错误 0 = 无错 1 = 错误

n4: 保留，始终为 0

远程站址：存取数据的 PLC 地址

数据指针：指向 PLC 中数据的间接指针

数据长度：存取的数据字节数目（1-16）

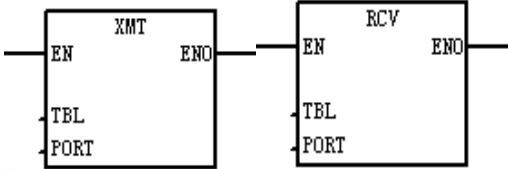
接收或传输数据区域：如下所示，为数据保留的 1-16 个字节（由数据长度指定）。

对于 NETR，该数据区域指执行 NETR 之后从远程站读取的数值存储的区域。

对于 NETW，该数据区域指执行 NETW 之前发送至远程站的数值存储的区域。

n3~n0: 错误代码，含义如下：

错误代码	说明
0	无错
1	超时错误：远程站不应答
2	接收错误：应答中存在校验、帧或校验和错误
3	脱机错误：重复站址或故障硬件引起的冲突
4	队列溢出错误：8 个以上 NETR/NETW 方框被激活
5	违反协议：未启用 SMB30 中的 PPI+即尝试执行 NETR/NETW
6	非法参数：NETR/NETW 表格包含一个非法或无效数值
7	无资源：远程站繁忙（正在上载或下载序列）
8	第 7 层错误：违反应用程序协议
9	信息错误：数据地址错误或数据长度不正确

传送接收指令	设置 ENO = 0的错误条件	0006 间接地址 0009 端口0中同时执行 XMT/RCV 000B 端口1中同时执行 XMT/RCV 接收状态字节显示接收有错误
	影响的特殊内存位	SMB86和 SMB186（接收用户的禁止命令终止接收信息、输入参数错误或无起始或结束条件、收到结束字符、超时、超出最大字符数或奇偶校验错误） SMB3（自由口接收优先级错误） SMB4（中断队列溢出）
指令列表： XMT TABLE, PORT RCV TABLE, PORT		
梯形图： 		
输入输出参数	数据类型	适用操作数
TBL	字节	VB, IB, QB, MB, SB, SMB, *VD, *LD, *AC
PORT	字节	常数（0用于 CPU224； 0或1用于 CPU 226/226M/226L）

功能说明：

传送指令（XMT）在自由端口模式中使用，通过通讯端口传送数据。

接收指令（RCV）开始或终止“接收信息”服务。您必须指定一个开始条件和一个结束条件，“接收”方框才能操作。通过指定端口（PORT）接收的信息存储在数据缓冲区（TBL）中。数据缓冲区中的第一个条目指定接收的字节数目。

示例程序：

STL：

ORGANIZATION_BLOCK 主程序: OB1

// 用串口工具写入以 S 开头 E 结束的字符串正确返回, Q0.0/Q1.0 亮

Network 1

```
LD      SM0.1
MOVB    0, VB500
MOVB    0, VB1000
MOVB    16#09, SMB30  //初始化 PORT0 为自由口, 波特率 9.6k, 8 位数据位, 无校验
MOVB    16#E0, SMB87 //允许接收, 起始结束检测
MOVB    'S', SMB88   //起始字符'S'
MOVB    'E', SMB89   //结束字符'E'
MOVB    200, SMB94   //接收缓冲区为 200 个字节
ATCH    INT0, 23     //启用中断, 接收完成
ATCH    INT1, 9      //发送完成
ENI
RCV      VB100, 0
INTERRUPT_BLOCK INT_0: INT0
```

Network 1

```
LD      SM0.0
XMT      VB100, 0      //将收到的字符发送给 Port 0
AENO
=        Q0.0
INCB     VB1000
```

Network 2

```
LD      SM0.0
CRETI
INTERRUPT_BLOCK INT_1: INT1
```

Network 1

```
LD      SM0.0
RCV      VB100, 0      //发送完成后接收新的字符
AENO
=        Q0.1
INCB     VB500
```

Network 2

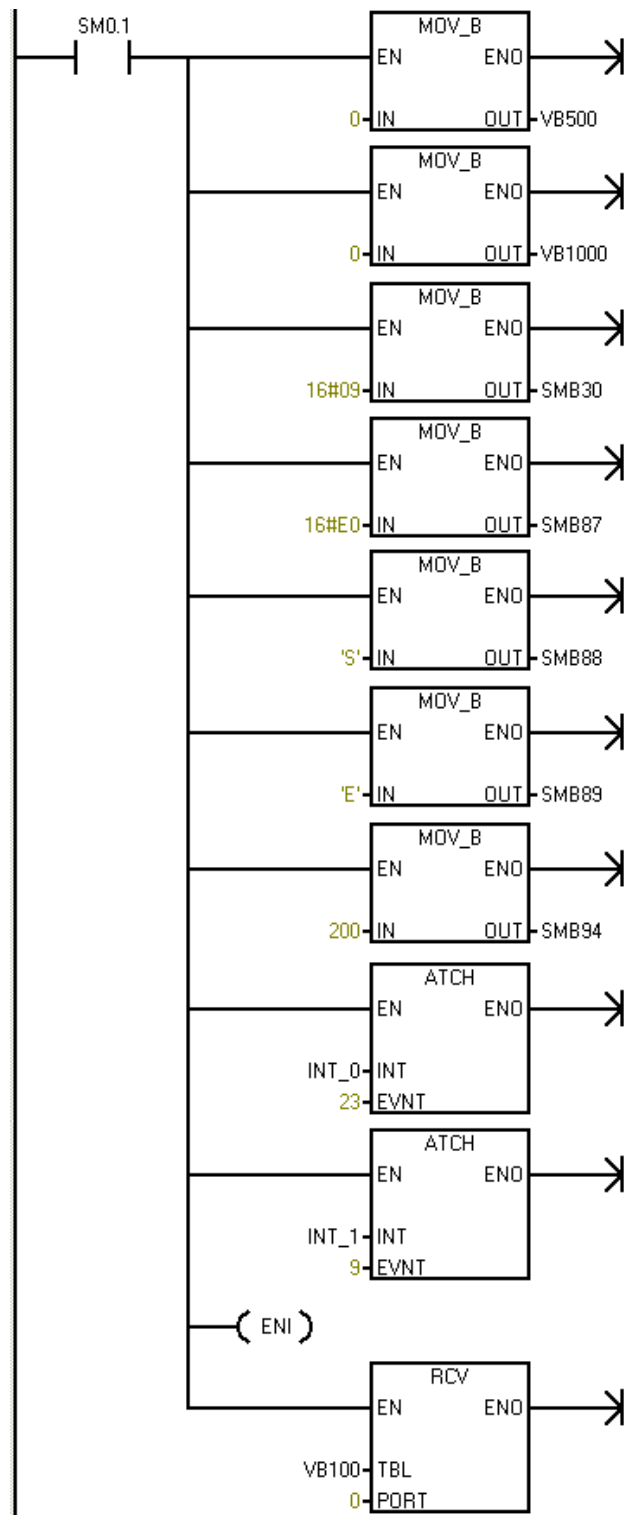
```
LD      SM0.0
CRETI
```

梯形图：

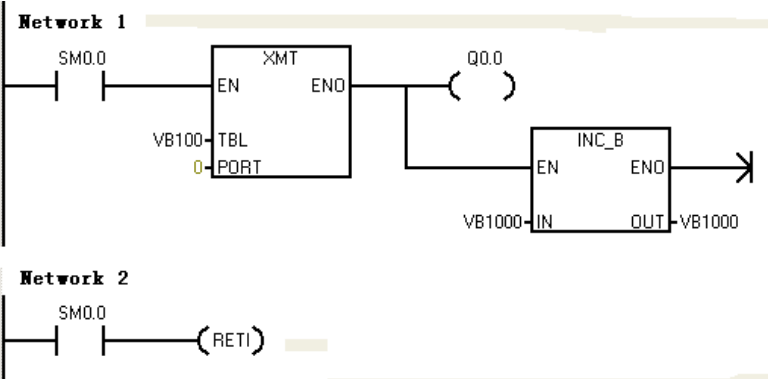
ORGANIZATION_BLOCK 主程序：OB1

// 用串口工具写入以 S 开头 E 结束的字符串正确返回，Q0.0/Q1.0 亮

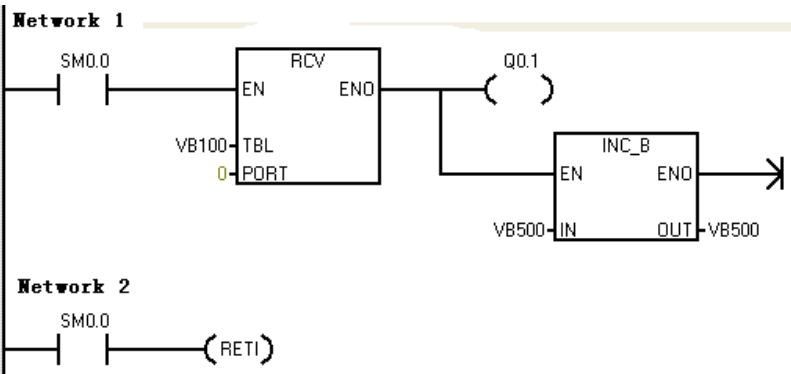
Network 1



INTERRUPT_BLOCK INT_0: INT0



INTERRUPT_BLOCK INT_1: INT1



以太网 PPI 网络读写指令 UDP_NETR/UDP_NETW		
指令列表: UDP_NETR/UDP_NETW EN, TABLE, ENO		
梯形图:		
参数	类型	意义
EN	BOOL	命令使能位，上升沿触发
TABLE	BYTE	包含状态位、IP 地址、指针、数据长度、数据字节，具体描述参考表“UDP_NETR/UDP_NETW 中参数 TABLE 的描述”。

功能说明: 以太网口网络读取指令（UDPNETR）开始一项通讯操作，通过 CPU 上的 Ethernet 通信口，根据表格 TBL 定义从远程设备收集数据。

以太网口网络写入指令（UDPNETW）开始一项通讯操作，通过 CPU 上的 Ethernet 通信口，根据表格 TBL 定义远程设备写入数据。

网络读写指令最多可从远程站读取或从远程站写入 200 字节信息，在程序中允许有任意数目的 NETR/NETW 指令，但在任何时间最多只能有 8 条指令被激活。

TABLE 参数使用说明
字节偏移量 7

D	A	E	n4	错误代码
远程站 IP 地址第 1 个字节				
远程站 IP 地址第 2 个字节				
远程站 IP 地址第 3 个字节				
远程站 IP 地址第 4 个字节				
端口号高字节				
端口号低字节				
指向远程站目标区域指针<I, Q, M, V, DB>（共 4 个字节）第 1 个字节				
指向远程站目标区域指针第 2 个字节				
指向远程站目标区域指针第 3 个字节				
指向远程站目标区域指针第 4 个字节				
数据长度				
数据字节 0				
数据字节 1				
...				
数据字节 199				

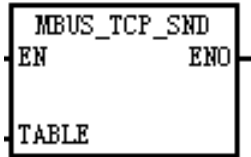
D: 命令完成位，0=未完成，1=完成
A: 激活位，0=指令不在执行中，1=指令正在执行中
E: 错误，0=无错，1=错误
n4: 保留，始终为 0

TABLE 参数中错误代码的描述

错误代码	说明
0	无错误
1	超时错误，远程站不应答
2	接收错误，应答中存在校验、帧或校验和错误
3	脱机错误，重复站地址或故障硬件引起的冲突
4	队列溢出错误，8 个以上 NETR/NETW 指令被激活
6	非法参数，NETR/NETW 表格包含一个非法或无效数值
7	无资源，远程站繁忙（正在上载或下载序列）
8	层 7 错误，违反应用程序协议
9	信息错误，数据地址或数据长度不正确
10	已达到最大连接数
129	内部错误，表指针为空

注释：支持同时 8 个不同 IP 地址（即 8 个连接）的读写，每个 IP 地址可有 32 条读写数据信息，每个信息读写的数据长度最大为 200 字节。

以太网 MODBUS 网络读写指令 MBUS_TCP_SND
指令列表: MBUS_TCP_SND EN, TABLE, ENO
梯形图:



参数	类型	意义
EN		MBUS_TCP_SND 使能位
TABLE	BYTE	包含状态位、IP 地址、端口号、读写类型、指针、数据长度、数据字节，具体描述参考下表 TABLE 参数说明。

功能说明：MODBUS_TCP 网络读写指令开始一项通讯操作，通过 MODBUS_TCP 协议与指定的远程设备交换数据。用户还可以使用“MODBUS_TCP 指令向导”配置通信程序。

TABLE 参数说明

字节偏移量				7	0
D	A	E	n4	错误代码	
远程站 IP 地址第 1 个字节					
远程站 IP 地址第 2 个字节					
远程站 IP 地址第 3 个字节					
远程站 IP 地址第 4 个字节					
端口号高字节					
端口号低字节					
消息读写类型 0：读 1：写					
读写地址第 1 个字节（高字节）					
读写地址第 2 个字节					
读写地址第 3 个字节					
读写地址第 4 个字节（低字节）					
保留					
单元 ID					
数据长度高字节					
数据长度低字节					
数据字节 0					
...					
数据字节 239					

D: 读写功能完成位，0=未完成，1=完成

A: 激活位，1=指令正在执行中，0=指令不在执行中

E: 错误状态位，0=无错；1=有错误

n4: 保留，始终为 0

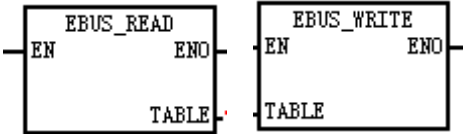
TABLE 参数中错误代码的描述（只有在 Done 位为 1 时，错误代码才有效）

错误代码	说明
0	无错误
1	响应校验错误

2	未用
3	接收超时（从站无响应）
4	请求参数错误（slave address, Modbus address, count, RW）
5	Modbus/自由口未使能
6	Modbus 正在忙于其它请求
7	响应错误（响应不是请求的操作）
8	响应 CRC 校验和错误
101	从站不支持请求的功能
102	从站不支持数据地址
103	从站不支持此种数据类型
104	从站设备故障
105	从站接受了信息，但是响应被延迟
106	从站忙，拒绝了该信息
107	从站拒绝了信息
108	从站存储器奇偶错误

注释：

- 同时可有多从站消息读写，但是每个从站只能同时有一条消息读写。
- TCP_MBUS_MSG 收到应答之后或者出错的时候 Done 才打开。
- TCP MODBUS 最大支持 32 个连接，每条消息一次最大可读写 240 字节。

外部总线读/写指令		
EBUSR TBL		
EBUSW TBL		
梯形图		
		
输入输出参数	数据类型	适用操作数
TBL	字节	VB, MB, *VD, *LD, *AC

功能说明：从模块读取 nBytes 到 TBL 中指向的地址/将 TBL 中指定的 nBytes 写入到模块。

TABLE 参数表定义：

错误码 ErrCode							
R3	R2	R1	R0	S3	S2	S1	S0
偏移量 Offset0							
偏移量 Offset1							
数据长度 Len							
数据字节 Byte[0]							
数据字节 Byte[1]							
...							
数据字节 Byte[Len-1]							

ErrCode：0—成功、其它—失败；

R3 R2 R1 R0：机架号 RackID

S3 S2 S1 S0：槽号 SlotID

Offset0 Offset1：模块参数的偏移量，与总线通信应用层协议表中的偏移量对应

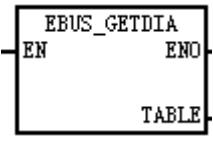
Len：要读取或写入的数据字节数

Byte：读取或写入的数据

获取外部总线的诊断信息

EBUSGETDIA TBL

梯形图



功能说明：获取模块的诊断信息

TABLE 参数表定义：

错误码 ErrCode							
R3	R2	R1	R0	S3	S2	S1	S0
数据长度 Len							
数据字节 Byte[0]							
数据字节 Byte[1]							
...							
数据字节 Byte[Len-1]							

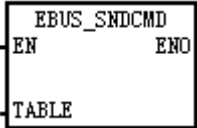
ErrCode：0—成功、其它—失败

R3 R2 R1 R0：机架号 RackID

S3 S2 S1 S0：槽号 SlotID

Len：读取的诊断信息的字节数

Byte：读取的诊断信息内容

外部总线发送命令指令		
EBUSSNDCMD TBL		
梯形图		
		
输入输出参数	数据类型	适用操作数
TABLE	字节	VB, MB, *VD, *LD, *AC

功能说明：向指定模块发送一条命令。

TABLE 参数表定义：

D	E	ErrCode5	ErrCode4	ErrCode3	ErrCode2	ErrCode1	ErrCode0
R3	R2	R1	R0	S3	S2	S1	S0
超时时间（ms）Timeout0							
超时时间（ms）Timeout1							
输入数据长度 LenIn[0]							
输入数据长度 LenIn[1]							
输入数据字节 ByteIn[0]							
输入数据字节 ByteIn[1]							
...							
输入数据字节 ByteIn[LenIn-1]							
输出数据长度 LenOut[0]							
输出数据长度 LenOut[1]							
输出数据字节 ByteOut[0]							
输出数据字节 ByteOut[1]							
...							
输出数据字节 ByteOut[LenOut-1]							

D：完成位（功能完成） 0 = 未完成，1 = 完成

E：错误位 0 = 无错，1 = 错误

错误代码	说明
0x00	没有错误
0x01	发送超时
0x02	模块类型不对
0x03	没有多余的发送控制块
0x04	模块忙
0x05	等待标志超时
0x06	总线标志错误
0x07	软件错误

R3 R2 R1 R0：机架号 RackID

S3 S2 S1 S0: 槽号 SlotID

Timeout0 Timeout1: 指令执行的超时时间

LenIn: 输入数据的字节数

ByteIn: 输入数据的内容, 格式根据具体模块来定

LenOut: 输出数据的字节数

ByteOut: 输出数据的内容, 格式根据具体模块来定

C 特殊存储器 (SM) 标志位 (小型 PLC 专用)

特殊存储器标志位提供了一些具有特殊功能的位, 便于用户查看和控制 CTSC-100/200 CPU 的状态。特殊存储器标志位能以位、字节、字或双字使用。

SM 标志位	功能
SMB0	状态位, 只读
SMB1	状态位, 只读
SMB2	自由口接收字符, 只读
SMB3	自由口奇偶校验错误, 只读
SMB4	队列溢出, 只读
SMB5	I/O 状态, 只读
SMB6	CPU 识别 (ID) 寄存器, 只读
SMB8 至 SMB21	I/O 模块识别和错误寄存器, 只读
SMW22 至 SMW26	扫描时间, 只读
SMB28 和 SMB29	模拟电位器, 只读
SMB30 和 SMB130	自由口控制寄存器
SMB31 和 SMW32	永久存储器 (EEPROM) 写控制
SMB34 和 SMB35	定时中断时间间隔寄存器
SMB36 至 SMB65	HSC0、HSC1 和 HSC2 寄存器
SMB66 至 SMB85	PTO/PWM 寄存器
SMB86 至 SMB94 SMB186 至 SMB194	接收信息控制
SMW98	扩展 I/O 总线错误

C.1 SMB0 至 SMB29 (只读特殊内存)

SMB0: 系统状态位

SMB0 的 8 个位在系统中有特殊作用, 在每次扫描周期结束时 PLC 会更新这 8 个位。用户在程序可以利用这些位的特殊功能来实现一些常用的操作, 具体描述见下表:

SMB0位	功能描述 (只读)
SM0.0	该位始终打开, 如果您希望程序中某段语句每一周期都能执行, 可用此位作使能位。
SM0.1	该位只在首次扫描时为1, 可用作初始化程序段的使能位。
SM0.2	如果断电保持数据丢失, 该位为打开一个扫描周期。该位可用作错误内存位或激活

	特殊启动顺序的使能位。
SM0.3	从电源打开进入 RUN（运行）模式时，该位为接通一个扫描周期。该位可用于在启动操作之前提供机器预热时间。
SM0.4	该位提供一个周期为1分钟30秒打开30秒关闭的方波时钟脉冲。
SM0.5	该位提供一个周期为1秒钟0.5秒打开0.5秒关闭的方波时钟脉冲。
SM0.6	该位为扫描时钟，本次扫描时置1，下次扫描时置0。可用作扫描计数器的使能位。
SM0.7	该位指示 PLC 开关的位置（0：TERM、1：RUN）。可用该位作自由口通信程序段的使能位，这样当开关在 RUN 位置时，自由口通信方式有效，切换至 TERM 位置时，PPI 通信有效。

SMB1：指令执行状态位

SMB1 的 8 个位用于指示某些指令执行中的错误或特殊状态，详细描述见下表：

SMB1位	功能描述（只读）
SM1.0	指令执行结果为0时，该位置1。
SM1.1	指令执行结果溢出或查出非法数值时，该位置1。
SM1.2	指令执行结果为负数时，该位置1。
SM1.3	指令中除数为0，该位置1。
SM1.4	执行 ATT 指令时，如果追加后的表长度超出表范围，该位置1。
SM1.5	执行 LIFO 或 FIFO 指令时，如果表为空，该位置1。
SM1.6	当试图把一个非 BCD 数转换为二进制数时，该位置1。
SM1.7	当 ASCII 码不能转换为有效的十六进制数时，该位置1。

SMB2：自由口接收字符

SMB2 为自由口接收字符缓冲区，在自由口通信方式下，接收到的每个字符都放在这里，便于程序读取。详细描述见下表：

SM 地址	功能描述（只读）
SMB2	该字节包含自由口通讯过程中从端口0或端口1接收的每个字符。

SMB3：自由口奇偶校验错误

SMB3 用于显示自由口奇偶校验错误。自由口通信模式下，在接收字符中检测到校验错误时，SM3.0 被置为 1。通常情况下，在程序接收和读取存储在 SMB2 中的信息字符数值之前，先使用该位检测自由口信息字符是否有传输错误。详细描述见下表：

SMB3位	功能描述（只读）
SM3.0	该位表示在端口0和端口1中是否存在校验错误 0：无错，1：错误
SM3.1～SM3.7	保留

SMB4：队列溢出

SMB4 包含中断队列溢出位，中断是否允许标志位，发送空闲位及是否存在强制标志位。队列溢出表明要么是中断发生的频率高于 CPU，要么是中断已经被全局中断禁止指令所禁止。详细

描述见下表：

SMB4位	功能描述（只读）
SM4.0	通信中断队列溢出时，该位打开。仅限在中断程序中使用，队列空置且控制返回主程序时，该位被复位。
SM4.1	输入中断队列溢出时，该位打开。仅限在中断程序中使用，队列空置且控制返回主程序时，该位被复位。
SM4.2	定时中断队列溢出时，该位打开。仅限在中断程序中使用，队列空置且控制返回主程序时，该位被复位。
SM4.3	检测到运行时间编程错误时，该位打开。
SM4.4	该位反映全局中断启用状态，启用中断时，该位打开。
SM4.5	端口0变送器闲置时，该位打开。
SM4.6	端口1变送器闲置时，该位打开。
SM4.7	当任何内存位置被强制时该位打开。

SMB5: I/O 状态

SMB5 包含系统里发现的 I/O 错误状态位，这些位提供了所发现的 I/O 错误的信息，详细描述见下表：

SMB5位	功能描述（只读）
SM5.0	当有 I/O 错误时，将该位置1。
SM5.1	当 I/O 总线上连接了过多的数字量 I/O 点时，将该位置1。
SM5.2	当 I/O 总线上连接了过多的模拟量 I/O 点时，将该位置1。
SM5.3	当 I/O 总线上连接了过多的智能 I/O 模块时，将该位置1。
SM5.4～SM5.7	保留

SMB6: CPU 识别（ID）寄存器

SMB6 为 CPU 识别（ID）寄存器，用于识别 CPU 的类型，详细描述见下表：

SM 地址	功能描述（只读）
SMB6	CPU 识别寄存器
SM6.0～SM6.3	保留
SM6.4～SM6.7	用于识别 CPU 类型 0000: CPU212, CPU222; 0010: CPU214, CPU224; 0110: CPU221; 1000: CPU215; 1001: CPU216, CPU226, CPU 226M, CPU 226XM, CPU 226XC, CPU226XM; 1110: H35-00;

SMB6 格式：

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
x	x	x	x	r	r	r	r
0000: CPU212, CPU222; 0010: CPU214, CPU224; 0110: CPU221; 1000: CPU215; 1001: CPU216, CPU226, CPU 226M, CPU				保留			

226XM, CPU 226XC, CPU226XM;
1110: H35-00

SMB8 至 SMB21: I/O 模块识别和错误寄存器

SMB8 到 SMB21 是按照字节对形式（相邻两个字节）为扩展模 0 到 6 而准备的。每对字节的偶数位字节为模块识别寄存器，奇数位字节为模块错误寄存器。前者标记着模块类型、I/O 类型、输入和输出点数；后者为对相应模块所测得的 I/O 错误提示。详细描述见下表：

SM 地址	功能描述（只读）
SMB8	模块0识别寄存器
SMB9	模块0错误寄存器
SMB10	模块1识别寄存器
SMB11	模块1错误寄存器
SMB12	模块2识别寄存器
SMB13	模块2错误寄存器
SMB14	模块3识别寄存器
SMB15	模块3错误寄存器
SMB16	模块4识别寄存器
SMB17	模块4错误寄存器
SMB18	模块5识别寄存器
SMB19	模块5错误寄存器
SMB20	模块6识别寄存器
SMB21	模块6错误寄存器

偶数字节（模块 ID 寄存器）格式：

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
m	t	t	a	i	i	q	q
m: 模块是否存在 0--有模块 1--无模块	tt: 模块类型 00--非智能 I/O 模块 01--智能模块 10--保留 11--保留		a: I/O 类型 0--开关量 1--模拟量	ii: 输入点数 00--无输入 01--2 AI 或 8 DI 10--4 AI 或 16 DI 11--8 AI 或 32 DI		qq: 输出点数 00--无输出 01--2 AQ 或 8 DQ 10--4 AQ 或 16 DQ 11--8 AQ 或 32 DQ	

奇数字节（模块错误寄存器）格式：

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
c	0	0	b	r	p	f	t
c: 配置错误 0--无错误 1--有错误			b: 总线错误或校验错误 0--无错误 1--有错误	r: 超范围错误 0--无错误 1--有错误	p: 用户电源错误 0--无错误 1--有错误	f: 熔断器错误 0--无错误 1--有错误	t: 端子块松动错误 0--无错误 1--有错误

SMW22 至 SMW26：扫描时间

SMW22、SMW24 和 SMW26 提供扫描时间信息：以毫秒计的最短扫描时间、最长扫描时间及上次扫描时间。详细描述见下表：

SM 地址	功能描述（只读）
SMW22	上次扫描时间
SMW24	进入 RUN 方式后，所记录的最短扫描时间
SMW26	进入 RUN 方式后，所记录的最长扫描时间

SMB28 至 SMB29：模拟电位器

特殊内存字节 28 和 29 包含与模拟电位器 0 和 1 轴位置对应的数字值。模拟电位器位于 CPU 前方存取门后方。用一把小螺丝刀调整电位计（沿顺时针方向增加，或沿逆时针方向减少）。此类只读数值可被程序用于各种不同的功能，例如，为定时器或计数器更新当前值，输入或改动预设值或设置限制。模拟电位器有一个 0 至 255 的额定范围，以及±2 计数的可靠重复。详细描述见下表：

SM 地址	功能描述（只读）
SMB28	存储模拟调节器0的输入值。在 STOP/RUN 方式下，每次扫描时更新该值。
SMB29	存储模拟调节器1的输入值。在 STOP/RUN 方式下，每次扫描时更新该值。

C.2 SMB30 至 SMB549（读 / 写特殊内存）**SMB30 和 SMB130：自由口控制寄存器**

SMB30 控制自由口 0 的通讯方式，SMB130 控制自由口 1 的通讯方式。您可以对 SMB30 和 SMB130 进行写和读，利用这两个字节设置自由口通信的操作方式，并提供自由口或者系统所支持的协议之间的选择。详细描述见下表：

SM 地址	功能描述
SMB30	自由口0控制寄存器
SMB130	自由口1控制寄存器
SM30.0~SM30.1	端口0通信协议选择
SM130.0~SM130.1	端口1通信协议选择
SM30.2~SM30.4	自由口0波特率
SM130.2~SM130.4	自由口1波特率
SM30.5	自由口0每个字符的数据位
SM130.5	自由口1每个字符的数据位
SM30.6~SM30.7	自由口0的校验选择
SM130.6~SM130.7	自由口1的校验选择

自由口控制字节格式：

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
p	p	d	b	b	b	m	m

pp: 校验选择 00--无校验 01--偶校验 10--无校验 11--奇校验	d: 每个字符的 数据位 0--8 位数据位 1--7 位数据位	bbb: 自由口波特率 000—38.4kbps 001—19.2kbps 010—9.6kbps 011—4.8kbps 100—2.4kbps 101—1.2kbps 110—115.2kbps 111—57.6kbps	mm: 协议设置 00--点到点接口协议 (PPI/从站模式) 01--自由口协议 10--PPI/主站模式 11--保留(缺省是 PPI/从站模式)

<注意> 当选择 mm = 10 (PPI 主站), PLC 将成为网络的一个主站, 可以执行 NETR 和 NETW 指令。PPI 模式下忽略 2 至 7 位。

SMB31 和 SMW32: 永久存储器 (EEPROM) 写控制

在用户程序的控制下, 您可以把 V 存储器中的数据存入永久存储器, 亦称非易失存储器。先把被存数据的地址存入 SMW32 中, 然后把存入命令存入 SMB31 中。一旦您发出存储命令, 则直到 CPU 完成存储操作 SM31.7 被置 0 之前, 您不可以改变 V 存储器的值。

在每次扫描周期末尾, CPU 检查是否有向永久存储器区中存数据的命令。如果有, 则将该数据存入永久存储器中。详细描述见下表:

SM 地址	功能描述
SMB31	永久存储器命令寄存器
SM31.0~SM31.1	要保存的数据大小: 00 = 字节; 01 = 字节; 10 = 字; 11 = 双字
SM31.2~SM31.6	保留, 始终为 00000
SM31.7	存入永久存储器, 每次存储操作完成后, S7-200 复位该位。 0 = 无执行存储器操作的请求 1 = 用户程序申请向永久存储器存储数据
SMW32	所存数据的 V 存储器地址, 该值是相对于 V0 的偏移量。当执行存储命令时, 把该数据存到永久存储器中相应的位置。

SMB34 和 SMB35: 定时中断的时间间隔寄存器

SMB34 和 SMB35 分别定义了定时中断 0 和中断 1 的时间间隔。您可以指定从 1 毫秒至 255 毫秒的时间间隔 (以 1 毫秒为增量)。如果相应的定时中断事件被连接到一个中断服务程序, CTSC-200CPU 就会获取该时间间隔值。若要改变该时间间隔, 您必须把定时中断事件再分配给同一或另一中断程序, 或通过中断分离来终止定时中断事件。

SM 地址	功能描述
SMB34	定时中断 0 的时间间隔数值
SMB35	定时中断 1 的时间间隔数值

SMB36 至 SMB65: HSC0、HSC1 和 HSC2 寄存器

SMB36 至 SMB65 用于监控和控制高速计数器 HSC0、HSC1 和 HSC2 操作。详细描述见下表:

SM 地址	功能描述
HSC0	

SMB36	HSC0计数器状态
SM36.0~SM36.4	保留
SM36.5	HSC0当前计数方向位：1=增计数
SM36.6	HSC0当前值等于预设值位：1=等于
SM36.7	HSC0当前值大于预设值位：1=大于
SMB37	HSC0控制字节
SM37.0	复位的有效控制位：0=高电平复位有效，1=低电平复位有效
SM37.1	保留
SM37.2	正交计数器的计数速率选择：0=4x 计数速率；1=1x 速率
SM37.3	HSC0方向控制位：1=增计数
SM37.4	HSC0更新方向：1=更新方向
SM37.5	HSC0更新预设值：1=向 HSC0写新的预设值
SM37.6	HSC0更新当前值：1=向 HSC0写新的初始值
SM37.7	HSC0有效位：1=有效
SMD38	HSC0新的初始值，把新的当前值写入 SMD38后将 SM37.6设为1，则当前值被更新
SMD42	HSC0新的预置值，把新的预设值写入 SMD42后将 SM37.5设为1，则预设值被更新
HSC1	
SMB46	HSC1计数器状态
SM46.0~SM46.4	保留
SM46.5	HSC1当前计数方向位：1=增计数
SM46.6	HSC1当前值等于预设值位：1=等于
SM46.7	HSC1当前值大于预设值位：1=大于
SMB47	HSC1控制字节
SM47.0	复位的有效控制位：0=高电平复位有效，1=低电平复位有效
SM47.1	HSC1启动有效电平控制位：0=高电平，1=低电平
SM47.2	正交计数器的计数速率选择：0=4x 计数速率；1=1x 速率
SM47.3	HSC1方向控制位：1=增计数
SM47.4	HSC1更新方向：1=更新方向
SM47.5	HSC1更新预设值：1=向 HSC1写新的预设值
SM47.6	HSC1更新当前值：1=向 HSC1写新的初始值
SM47.7	HSC1有效位：1=有效
SMD48	HSC1新的初始值，把新的当前值写入 SMD48后将 SM47.6设为1，则当前值被更新
SMD52	HSC1新的预设值，把新的预设值写入 SMD52后将 SM47.5设为1，则预设值被更新
HSC2	
SMB56	HSC2计数器状态
SM56.0~SM56.4	保留
SM56.5	HSC2当前计数方向位：1=增计数
SM56.6	HSC2当前值等于预设值位：1=等于
SM56.7	HSC2当前值大于预设值位：1=大于

SMB57	HSC2控制字节
SM57.0	复位的有效控制位：0=高电平复位有效，1=低电平复位有效
SM57.1	HSC1启动有效电平控制位：0=高电平，1=低电平
SM57.2	正交计数器的计数速率选择：0=4x 计数速率；1=1x 速率
SM57.3	HSC2方向控制位：1=增计数
SM57.4	HSC2更新方向：1=更新方向
SM57.5	HSC2更新预设值：1=向 HSC2写新的预设值
SM57.6	HSC2更新当前值：1=向 HSC2写新的初始值
SM57.7	HSC2有效位：1=有效
SMD58	HSC2新的初始值，把新的当前值写入 SMD58后将 SM57.6设为1，则当前值被更新
SMD62	HSC2新的预置值，把新的预设值写入 SMD62后将 SM57.5设为1，则预设值被更新

<注释>

- 1) 计数器状态位仅限在执行由高速计数器事件触发的中断程序里有效。
- 2) 当您使用 HSC 外部复位中断事件时，请勿在程序里重新设置新的当前值，然后在连接于该事件的中断程序里重新使能计数器，此操作会引起严重错误。

SMB66 至 SMB85：PTO/PWM 寄存器

SMB66 至 SMB85 被用于监控和控制 PLC（脉冲）指令的脉冲链输出和脉冲宽度调制功能。详细描述见下表：

SM 地址	功能描述
PTO0	
SMB66	PTO0状态字节
SM66.0~SM66.3	保留
SM66.4	PTO0包络终止：0=无错，1=由于增量计算错误而终止
SM66.5	PTO0包络终止：0=不由用户命令终止；1=由用户命令终止
SM66.6	PTO0管道溢出（当使用外部包络时由系统清除，否则由用户程序清除）：0=无溢出，1=有溢出
SM66.7	PTO0空闲位：0=PTO 忙，1=PTO 空闲
SMB67	Q0.0的控制脉冲串输出和脉冲宽度调制
SM67.0	PTO0/PWM0更新周期：1=写新的周期值
SM67.1	PWM0更新脉冲宽度值：1=写新的脉冲宽度
SM67.2	PTO0更新脉冲量：1=写新的脉冲量
SM67.3	PTO0/PWM0基准时间单元：0=1 μs/格，1=1 ms/格
SM67.4	同步更新 PWM0：0=异步更新，1=同步更新
SM67.5	PTO0操作：0=单段操作（周期和脉冲数存在 SM 存储器中），1=多段操作（包络表存在 V 存储器区）
SM67.6	PTO0/PWM0模式选择：0=PTO，1=PWM
SM67.7	PTO0/PWM0有效位：1=有效
SMW68	PTO0/PWM0周期（2~65，535个时间基准）

SMW70	PWM0脉冲宽度值（0~65，535个时间基准）
SMD72	PTO0脉冲计数值（1~232 --1）
PTO1	
SMB76	PTO1状态字节
SM76.0~SM76.3	保留
SM76.4	PTO1包络终止：0=无错，1=由于增量计算错误而终止
SM76.5	PTO1包络终止：0=不由用户命令终止；1=由用户命令终止
SM76.6	PTO1管道溢出（当使用外部包络时由系统清除，否则由用户程序清除）：0=无溢出，1=有溢出
SM76.7	PTO1空闲位：0=PTO 忙，1=PTO 空闲
SMB77	Q0.1的控制脉冲串输出和脉冲宽度调制
SM77.0	PTO1/PWM1更新周期：1=写新的周期值
SM77.1	PWM1更新脉冲宽度值：1=写新的脉冲宽度
SM77.2	PTO1更新脉冲量：1=写新的脉冲量
SM77.3	PTO1/PWM1基准时间单元：0=1 μs/格，1=1 ms/格
SM77.4	同步更新 PWM1：0=异步更新，1=同步更新
SM77.5	PTO1操作：0=单段操作（周期和脉冲数存在 SM 存储器中），1=多段操作（包络表存在 V 存储器区）
SM77.6	PTO1/PWM1模式选择：0=PTO，1=PWM
SM77.7	PTO1/PWM1有效位：1=有效
SMW78	PTO1/PWM1周期（2~65535个时间基准）
SMW80	PWM1脉冲宽度值（0~65535个时间基准）
SMD82	PTO1脉冲计数值（1~232 --1）

SMB86 至 SMB94 和 SMB186 至 SMB194：接收信息控制

SMB86 至 SMB94 以及 SMB186 至 SMB194 被用于控制和读取有关“接收信息”指令的状态。

详细描述见下表：

SM 地址	功能描述
P0接收信息	
SMB86	P0接收信息状态字节
SMB87	P0接收信息控制字节
SMB88	P0信息字符的开始
SMB89	P0信息字符的结束
SMW90	P0空闲行时间间隔单位为毫秒。空闲行时间结束后接收的首个字符为新信息开始。
SMW92	P0字符间/信息间定时器超时值（用毫秒表示）。如果超过时间，就停止接收信息。
SMB94	P0接收字符的最大数（1~255字节）。 <注意> 这个区一定要设为希望的最大缓冲区，即使不使用字符计数信息终止。
SMB110	用于启动强制上载保护功能
SMB111	编程卡覆盖选项： 0 -覆盖程序块和数据块；1 -不覆盖程序块，覆盖数据块； 2 -覆盖程序块，不覆盖数据块；3 -不覆盖程序块和数据块；4 -与为0时相同
SMB112	始终正确显示当前 CPU 版本信息

P1接收信息	
SMB186	P1接收信息状态字节
SMB187	P1接收信息控制字节
SMB188	P1信息字符的开始
SMB189	P1信息字符的结束
SMW190	P1空闲行时间间隔单位为毫秒。空闲行时间结束后接收的首个字符为新信息开始。
SMW192	P1字符间/信息间定时器超时值（用毫秒表示）。如果超过时间，就停止接收信息。
SMB194	P1接收字符的最大数（1到255字节）。<注意> 这个区一定要设为希望的最大缓冲区，即使不使用字符计数信息终止。

接收信息状态字节格式：

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
n	r	e	0	0	t	c	p

n = 1: 接收用户的禁止命令终止接收信息

r = 1: 输入参数错误或无起始或结束条件导致接收信息终止

e = 1: 收到结束字符

t = 1: 接收信息终止：超时

c = 1: 接收信息终止：超出最大字符数

p = 1: 接收信息终止：奇偶校验错误

接收信息控制字节格式：

MSB				LSB			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
en	sc	ec	il	c/m	tmr	bk	0

en: 0 = 禁止接收信息功能。

1 = 允许接收信息功能。

每次执行 RCV 指令时检查允许/禁止接收信息位。

sc: 0 = 忽略 SMB88 或 SMB188。

1 = 使用 SMB88 或 SMB188 的值检测起始信息。

ec: 0 = 忽略 SMB89 或 SMB189。

1 = 使用 SMB89 或 SMB189 的值检测结束信息。

il: 0 = 忽略 SMW90 或 SMW190。

1 = 使用 SMW90 或 SMW190 的值检测空闲状态。

c/m: 0 = 定时器是内字符间定时器。

1 = 定时器是信息定时器。

tmr: 0 = 忽略 SMW92 或 SMW192。

1 = 当 SMW92 或 SMW192 中的定时时间超出时终止接收。

bk: 0 = 忽略中断条件。

1 = 用中断条件作为信息检测的开始。

SMW98: 扩展 I/O 总线错误

SMW98 给出有关扩展 I/O 总线的错误数的信息。详细描述见下表:

SM 地址	功能描述
SMW98	当扩展总线出现校验错误时, 该处每次增加1。当系统得电时或用户程序写入零, 可以进行清零。

SMB136 至 SMB165: HSC3、HSC4 和 HSC5 寄存器

SMB136 至 SMB165 用于监视和控制高速计数器 HSC3、HSC4 和 HSC5 的操作。详细描述见下表:

SM 地址	功能描述
HSC3	
SMB136	HSC3计数器状态
SM136.0~SM136.4	保留
SM136.5	HSC3当前计数方向位: 1=增计数
SM136.6	HSC3当前值等于预设值位: 1=等于
SM136.7	HSC3当前值大于预设值位: 1=大于
SMB137	HSC3控制字节
SM137.0~SM137.2	保留
SM137.3	HSC3方向控制位: 1=增计数
SM137.4	HSC3更新方向: 1=更新方向
SM137.5	HSC3更新预设值: 1=向 HSC3写新的预设值
SM137.6	HSC3更新当前值: 1=向 HSC3写新的初始值
SM137.7	HSC3有效位: 1=有效
SMD138	HSC3新的初始值, 把新的当前值写入 SMD138后将 SM137.6设为1, 则当前值被更新
SMD142	HSC3新的预置值, 把新的预设值写入 SMD142后将 SM137.5设为1, 则预设值被更新
HSC4	
SMB146	HSC4计数器状态
SM146.0~SM146.4	保留
SM146.5	HSC4当前计数方向位: 1=增计数
SM146.6	HSC4当前值等于预设值位: 1=等于
SM146.7	HSC4当前值大于预设值位: 1=大于
SMB147	HSC4控制字节
SM147.0	复位的有效控制位: 0=高电平复位有效, 1=低电平复位有效
SM147.1	保留

SM147.2	正交计数器的计数速率选择：0=4x 计数速率；1=1x 速率
SM147.3	HSC4方向控制位：1=增计数
SM147.4	HSC4更新方向：1=更新方向
SM147.5	HSC4更新预设值：1=向 HSC4写新的预设值
SM147.6	HSC4更新当前值：1=向 HSC4写新的初始值
SM147.7	HSC4有效位：1=有效
SMD148	HSC4新的初始值，把新的当前值写入 SMD148后将 SM147.6设为1，则当前值被更新
SMD152	HSC4新的预设值，把新的预设值写入 SMD152后将 SM147.5设为1，则预设值被更新
HSC5	
SMB156	HSC5计数器状态
SM156.0~SM156.4	保留
SM156.5	HSC5当前计数方向位：1=增计数
SM156.6	HSC5当前值等于预设值位：1=等于
SM156.7	HSC5当前值大于预设值位：1=大于
SMB157	HSC5控制字节
SM157.0~SM157.2	保留
SM157.3	HSC5方向控制位：1=增计数
SM157.4	HSC5更新方向：1=更新方向
SM157.5	HSC5更新预设值：1=向 HSC5写新的预设值
SM157.6	HSC5更新当前值：1=向 HSC5写新的初始值
SM157.7	HSC5有效位：1=有效
SMD158	HSC5新的初始值，把新的当前值写入 SMD158后将 SM157.6设为1，则当前值被更新
SMD162	HSC5新的预置值，把新的预设值写入 SMD162后将 SM157.5设为1，则预设值被更新

<注释>

- 1) 计数器状态位仅限在执行由高速计数器事件触发的中断程序里有效。
- 2) 当您使用 HSC 外部复位中断事件时，请勿在程序里重新设置新的当前值，然后在连接于该事件的中断程序里重新使能计数器，此操作会引起严重错误。

SMB166 至 SMB185: PTO0, PTO1 包络定义表

SMB166 到 SMB194 用来显示包络步的数量和包络表的地址和 V 存储器区中表的地址。详细描述见下表：

SM 地址	功能描述
PTO0	
SMB166	PTO0的包络步当前计数值
SMB167	保留
SMW168	PTO0的包络表 V 存储器地址（从 V0开始的偏移量）
SMB170	线性 PTO0状态字节

SMB171	线性 PTO0结果字节
SMD172	指定线性PTO0发生器在手动模式时产生的频率。频率是以 Hz 为单位的双整型值。
PTO1	
SMB176	PTO1的包络步当前计数值
SMB177	保留
SMW178	PTO1的包络表 V 存储器地址（从 V0开始的偏移量）
SMB180	线性 PTO1状态字节
SMB181	线性 PTO1结果字节
SMD182	指定线性PTO1发生器在手动模式时产生的频率。频率是以 Hz 为单位的双整型值。

SMB200 至 SMB549：智能模块状态

SMB200 到 SMB549 预留存储智能扩展模块的信息。详细描述见下表：

槽号0的 智能模块	槽号1的 智能模块	槽号2的 智能模块	槽号3的 智能模块	槽号4的 智能模块	槽号5的智 能模块	槽号6的 智能模块	功能描述
SMB200 ~ SMB215	SMB250 ~ SMB265	SMB300 ~ SMB315	SMB350 ~ SMB365	SMB400 ~ SMB415	SMB450 ~ SMB465	SMB500 ~ SMB515	模块名称（16个 ASCII 字符）
SMB216 ~ SMB219	SMB266 ~ SMB269	SMB316 ~ SMB319	SMB366 ~ SMB369	SMB416 ~ SMB419	SMB466 ~ SMB469	SMB516 ~ SMB519	SW 修订号（4 个 ASCII 字符）
SMW220 ~ SMB222	SMW270 ~ SMB272	SMW320 ~ SMB322	SMW370 ~ SMB372	SMW420 ~ SMB422	SMW470 ~ SMB472	SMW520 ~ SMB522	错误代码
SMB249	SMB299	SMB349	SMB399	SMB449	SMB499	SMB549	与特定模块类 型相关的信息

D 特殊存储器（SM）标志位（中型 PLC 专用）

特殊存储器标志位提供了一些具有特殊功能的位，便于用户查看和控制 CTH3 系列 CPU 的状态。特殊存储器标志位能以位、字节、字或双字使用。

容量	2K
位地址范围	SM0.0~SM549.7
字节地址范围	SMB0~SMB549
字地址范围	SMW0~SMW548
双字地址范围	SMD0~SMD546
访问属性	支持直接访问/间接访问
保持属性	不能掉电保持

特殊寄存器定义

特殊寄存器	功能说明
SMB0	系统状态位
SM0.0	该位总是打开。
SM0.1	首次扫描周期时该位打开，一种用途是调用初始化子程序。
SM0.2	如果保留性数据丢失，该位为一次扫描周期打开。该位可用作错误内存位或激活特殊启动顺序的机制。

特殊寄存器	功能说明
SM0.3	从电源开启条件进入 RUN（运行）模式时，该位为一次扫描周期打开。该位可用于在启动操作之前提供机器预热时间。
SM0.4	该位提供时钟脉冲，该脉冲在1分钟的周期时间内 OFF（关闭）30秒，ON（打开）30秒。该位提供便于使用的延迟或1分钟时钟脉冲。
SM0.5	该位提供时钟脉冲，该脉冲在1秒钟的周期时间内 OFF（关闭）0.5秒，ON（打开）0.5秒。该位提供便于使用的延迟或1秒钟时钟脉冲。
SM0.6	该位是扫描周期时钟，为一次扫描打开，然后为下一次扫描关闭。该位可用作扫描计数器输入。
SM0.7	该位表示“模式”开关的当前位置（关闭 =“终止”位置，打开 =“运行”位置）。开关位于 RUN（运行）位置时，您可以使用该位启用自由口模式，可使用转换至“终止”位置的方法重新启用带 PC / 编程设备的正常通讯。
SMB1	指令执行状态位
SM1.0	当操作结果为零时，某些指令的执行打开该位。
SM1.1	当溢出结果或检测到非法数字数值时，某些指令的执行打开该位。
SM1.2	数学操作产生负结果时，该位打开。
SM1.3	尝试除以零时，该位打开。
SM1.4	“增加至表格”指令尝试过度填充表格时，该位打开。
SM1.5	LIFO 或 FIFO 指令尝试从空表读取时，该位打开。
SM1.6	尝试将非 BCD 数值转换为二进制数值时，该位打开。
SM1.7	当 ASCII 数值无法转换成有效的十六进制数值时，该位打开。
SMB2	自由口接收字符
SMB3	自由口校验错误
SM3.0	该位表示在端口0和端口1中出现校验错误。（0 = 无错；1 = 错误）
SM3.1-3	保留
SM3.4	如果存在任何 I/O 错误，该位打开。
SM3.5	如果过多数字 I/O 模块与 I/O 总线连接，该位打开。
SM3.6	如果过多模拟 I/O 模块与 I/O 总线连接，该位打开。
SM3.7	如果过多智能 I/O 模块与 I/O 总线连接，该位打开。
SMB4	中断队列溢出、运行时间程序错误、中断启用、自由口变送器被强制
SM4.0	通讯中断队列溢出时，该位打开。
SM4.1	输入中断队列溢出时，该位打开。
SM4.2	定时中断队列溢出时，该位打开。
SM4.3	检测到运行时间编程错误时，该位打开。
SM4.4	该位反映全局中断启用状态。启用中断时，该位打开。
SM4.5	发送器闲置（端口0）时，该位打开。
SM4.6	发送器闲置（端口1）时，该位打开。
SM4.7	当任何内存位置被强制时该位打开。
SMB5	触发当前 OB 的中断事件号

特殊寄存器	功能说明
SMB6	模块诊断信息
SM6.0	保留(通道信息存在)
SM6.1	保留(用户信息存在)
SM6.2	保留(来自替代的诊断中断)
SM6.3	保留
SM6.4~SM6.7	PLC 类型: 0000: CPU212、CPU222; 0010: CPU214、CPU224; 0110: CPU221; 1000: CPU215; 1001: CPU216, CPU226, CPU 226M, CPU 226XM, CPU 226XC, CPU226XM; 1110: H35-00;
SMB7	模块诊断信息
SM7.0	CPU 内部错误
SM7.1	CPU 外部错误
SM7.2	通信错误
SM7.3	操作方式(0: 运行, 1: 停机)
SM7.4	看门狗定时器超时
SM7.5	保留(内部电源故障)
SM7.6	保留(背板故障)
SM7.7	子模块丢失或有故障,无存储卡或存储卡错误
SMD8	中型 PLC 子型号
SMB12~SMB21	保留
SMW22	最后一次扫描时间
SMW24	自进入 RUN 模式以来最小扫描时间
SMW26	自进入 RUN 模式以来最大扫描时间
SMW28	产生诊断中断(发生故障)或者硬件中断的模块逻辑地址
SMB30、SMB130	自由口控制寄存器
Bit7:6	00: 无校验; 01: 偶校验; 10: 无校验; 11: 奇校验
Bit5	0: 每个字符8个数据位; 1: 每个字符7个数据位
Bit4:2	000: 38400bps、001: 19200bps、010: 9600bps、011: 4800bps、 100: 2400bps、101: 1200bps、110: 115200bps、111: 57600bps
Bit1:0	00: 点对点协议 (PPI/从属模式) 01: 自由口协议 10: PPI/主站模式 11: 保留 (PPI/从站模式默认值)

特殊寄存器	功能说明
SMB31	永久性内存（EEPROM）写入控制
Bit7	0: 无执行保存操作请求; 1: 用户程序请求 CPU 将数据保存至固定存储区, 每次保存完成后, CPU 复位该位
Bit6:2	保留
Bit1:0	需要保存的数值大小 00: 字节; 01: 字节; 10: 字; 11: 双字
SMW32	需要保存的 V 内存位置地址
SMB34	毫秒定时中断0时间间隔数值 (ms), 对应中断事件号10
SMB35	毫秒定时中断1时间间隔数值 (ms), 对应中断事件号11
SMB36~SMB65	保留
SMB66~SMB85	保留
SMB86~SMB94 SMB186~SMB194	自由口接收信息控制
SMW96	微秒中断0时间间隔数值 (100us), 对应中断事件号34
SMW98	微秒中断1时间间隔数值 (100us), 对应中断事件号35
SMW100	保留
SMW102	CPU 固件版本
SMW104	CPU 硬件版本
SMB106~SMB135	保留
SMB136~SMB165	保留
SMB166~SMB185	保留
SMB200~SMB205	保存并显示本机的 MAC 地址
SMB206~SMB499	保留
SMB500	第1个模块(Rack0, Slot3)错误
SMB501	第2个模块(Rack0, Slot4)错误
SMB502	第3个模块(Rack0, Slot5)错误
SMB503	第4个模块(Rack0, Slot6)错误
SMB504	第5个模块(Rack0, Slot7)错误
SMB505	第6个模块(Rack0, Slot8)错误
SMB506	第7个模块(Rack0, Slot9)错误
SMB507	第8个模块(Rack0, Slot10)错误
SMB508	第9个模块(Rack1, Slot3)错误
SMB509	第10个模块(Rack1, Slot4)错误
SMB510	第11个模块(Rack1, Slot5)错误
SMB511	第12个模块(Rack1, Slot6)错误
SMB512	第13个模块(Rack1, Slot7)错误
SMB513	第14个模块(Rack1, Slot8)错误
SMB514	第15个模块(Rack1, Slot9)错误
SMB515	第16个模块(Rack1, Slot10)错误
SMB516	第17个模块(Rack2, Slot3)错误

特殊寄存器	功能说明
SMB517	第18个模块(Rack2, Slot4)错误
SMB518	第19个模块(Rack2, Slot5)错误
SMB519	第20个模块(Rack2, Slot6)错误
SMB520	第21个模块(Rack2, Slot7)错误
SMB521	第22个模块(Rack2, Slot8)错误
SMB522	第23个模块(Rack2, Slot9)错误
SMB523	第24个模块(Rack2, Slot10)错误
SMB524	第25个模块(Rack3, Slot3)错误
SMB525	第26个模块(Rack3, Slot4)错误
SMB526	第27个模块(Rack3, Slot5)错误
SMB527	第28个模块(Rack3, Slot6)错误
SMB528	第29个模块(Rack3, Slot7)错误
SMB529	第30个模块(Rack3, Slot8)错误
SMB530	第31个模块(Rack3, Slot9)错误
SMB531	第32个模块(Rack3, Slot10)错误
SMB532~SMB549	保留

表 14-28 特殊存储区 SMB550~SMB2047 诊断功能

SMB	错误描述
SMB550~SMB589	第一个 CAN 模块信息（按硬件组态排列）
SMB550	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB551	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB552~SMB581	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB582	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB583	CAN 模块软件版本号
SMB584~SMB589	保留

SMB590~SMB629	第二个 CAN 模块信息（按硬件组态排列）
SMB590	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB591	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB592~SMB621	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB622	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB623	CAN 模块软件版本号
SMB624~SMB629	保留
SMB630~SMB669	第三个 CAN 模块信息（按硬件组态排列）
SMB630	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB631	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB632~SMB661	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB662	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行

	0x7F: 预运行
SMB663	CAN 模块软件版本号
SMB664~SMB669	保留
SMB670~SMB709	第四个 CAN 模块信息（按硬件组态排列）
SMB670	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB671	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB672~SMB701	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB702	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB703	CAN 模块软件版本号
SMB704~SMB709	保留
SMB710~SMB749	第五个 CAN 模块信息（按硬件组态排列）
SMB710	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB711	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB712~SMB741	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB742	从站32状态（按节点 ID 排列） 0x00: 初始化

	0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB743	CAN 模块软件版本号
SMB744~SMB749	保留
SMB750~SMB789	第六个 CAN 模块信息（按硬件组态排列）
SMB750	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB751	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB752~SMB781	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB782	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB783	CAN 模块软件版本号
SMB784~SMB789	保留
SMB790~SMB829	第七个 CAN 模块信息（按硬件组态排列）
SMB790	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB791	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行

SMB792~SMB821	从站2状态（按节点 ID 排列）~ 从站31状态（按节点 ID 排列）
SMB822	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB823	CAN 模块软件版本号
SMB824~SMB829	保留
SMB830~SMB869	第八个 CAN 模块信息（按硬件组态排列）
SMB830	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB831	从站1状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB832~SMB861	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB862	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB863	CAN 模块软件版本号
SMB864~SMB869	保留
SMB870	模块内部诊断控制字 1: 获取 SMB871指定的模块的内部诊断并将其写入 SMB872后面的字节里头
SMB871	内部诊断模块号，需要获取内部诊断的模块号 0~3位: 槽号 SlotID; 4~7位: 机架号 RackID。
SMB872~SMB905	模块内部诊断
SMB906~SMB999	保留
SMB1000~SMB1015	第一个模块(Rack0, Slot3)描述信息
SMB1000~SMB1002	实际模块信息
SMB1003	模块版本号

SMD1004	模块错误计数
SMB1008~SMB1015	保留
SMB1016~SMB1031	第二个模块(Rack0, Slot4)描述信息
SMB1016~SMB1018	实际模块信息
SMB1019	模块版本号
SMD1020	模块错误计数
SMB1024~SMB1031	保留
SMB1032~SMB1047	第三个模块(Rack0, Slot5)描述信息
SMB1032~SMB1034	实际模块信息
SMB1035	模块版本号
SMD1036	模块错误计数
SMB1040~SMB1047	保留
SMB1048~SMB1063	第四个模块(Rack0, Slot6)描述信息
SMB1048~SMB1050	实际模块信息
SMB1051	模块版本号
SMD1052	模块错误计数
SMB1056~SMB1063	保留
SMB1064~SMB1079	第五个模块(Rack0, Slot7)描述信息
SMB1064~SMB1066	实际模块信息
SMB1067	模块版本号
SMD1068	模块错误计数
SMB1072~SMB1079	保留
SMB1080~SMB1095	第六个模块(Rack0, Slot8)描述信息
SMB1080~SMB1082	实际模块信息
SMB1083	模块版本号
SMD1084	模块错误计数
SMB1088~SMB1095	保留
SMB1096~SMB1111	第七个模块(Rack0, Slot9)描述信息
SMB1096~SMB1098	实际模块信息
SMB1099	模块版本号
SMD1100	模块错误计数
SMB1104~SMB1111	保留
SMB1112~SMB1127	第八个模块(Rack0, Slot10)描述信息
SMB1112~SMB1114	实际模块信息
SMB1115	模块版本号
SMD1116	模块错误计数
SMB1120~SMB1127	保留
SMB1128~SMB1143	第九个模块(Rack1, Slot3)描述信息
SMB1128~SMB1130	实际模块信息
SMB1131	模块版本号
SMD1132	模块错误计数
SMB1136~SMB1143	保留

SMB1144~SMB1159	第十个模块(Rack1, Slot4)描述信息
SMB1144~SMB1146	实际模块信息
SMB1147	模块版本号
SMD1148	模块错误计数
SMB1152~SMB1159	保留
SMB1160~SMB1175	第十一个模块(Rack1, Slot5)描述信息
SMB1160~SMB1162	实际模块信息
SMB1163	模块版本号
SMD1164	模块错误计数
SMB1168~SMB1175	保留
SMB1176~SMB1191	第十二个模块(Rack1, Slot6)描述信息
SMB1176~SMB1178	实际模块信息
SMB1179	模块版本号
SMD1180	模块错误计数
SMB1184~SMB1191	保留
SMB1192~SMB1207	第十三个模块(Rack1, Slot7)描述信息
SMB1192~SMB1194	实际模块信息
SMB1195	模块版本号
SMD1196	模块错误计数
SMB1200~SMB1207	保留
SMB1208~SMB1223	第十四个模块(Rack1, Slot8)描述信息
SMB1208~SMB1210	实际模块信息
SMB1211	模块版本号
SMD1212	模块错误计数
SMB1216~SMB1223	保留
SMB1224~SMB1239	第十五个模块(Rack1, Slot9)描述信息
SMB1224~SMB1226	实际模块信息
SMB1227	模块版本号
SMD1228	模块错误计数
SMB1232~SMB1239	保留
SMB1240~SMB1255	第十六个模块(Rack1, Slot10)描述信息
SMB1240~SMB1242	实际模块信息
SMB1243	模块版本号
SMD1244	模块错误计数
SMB1248~SMB1255	保留
SMB1256~SMB1271	第十七个模块(Rack2, Slot3)描述信息
SMB1256~SMB1258	实际模块信息
SMB1259	模块版本号
SMD1260	模块错误计数
SMB1264~SMB1271	保留
SMB1272~SMB1287	第十八个模块(Rack2, Slot4)描述信息
SMB1272~SMB1274	实际模块信息

SMB1275	模块版本号
SMD1276	模块错误计数
SMB1280~SMB1287	保留
SMB1288~SMB1303	第十九个模块(Rack2, Slot5)描述信息
SMB1288~SMB1290	实际模块信息
SMB1291	模块版本号
SMD1292	模块错误计数
SMB1296~SMB1303	保留
SMB1304~SMB1319	第二十个模块(Rack2, Slot6)描述信息
SMB1304~SMB1306	实际模块信息
SMB1307	模块版本号
SMD1308	模块错误计数
SMB1312~SMB1319	保留
SMB1320~SMB1335	第二十一个模块(Rack2, Slot7)描述信息
SMB1320~SMB1322	实际模块信息
SMB1323	模块版本号
SMD1324	模块错误计数
SMB1328~SMB1335	保留
SMB1336~SMB1351	第二十二个模块(Rack2, Slot8)描述信息
SMB1336~SMB1338	实际模块信息
SMB1339	模块版本号
SMD1340	模块错误计数
SMB1344~SMB1351	保留
SMB1352~SMB1367	第二十三模块(Rack2, Slot9)描述信息
SMB1352~SMB1354	实际模块信息
SMB1355	模块版本号
SMD1356	模块错误计数
SMB1360~SMB1367	保留
SMB1368~SMB1383	第二十四模块(Rack2, Slot10)描述信息
SMB1368~SMB1370	实际模块信息
SMB1371	模块版本号
SMD1372	模块错误计数
SMB1376~SMB1383	保留
SMB1384~SMB1399	第二十五个模块(Rack3, Slot3)描述信息
SMB1384~SMB1386	实际模块信息
SMB1387	模块版本号
SMD1388	模块错误计数
SMB1392~SMB1399	保留
SMB1400~SMB1415	第二十六个模块(Rack3, Slot4)描述信息
SMB1400~SMB1402	实际模块信息
SMB1403	模块版本号
SMD1404	模块错误计数

SMB1408~SMB1415	保留
SMB1416~SMB1431	第二十七个模块(Rack3, Slot5)描述信息
SMB1416~SMB1418	实际模块信息
SMB1419	模块版本号
SMD1420	模块错误计数
SMB1424~SMB1431	保留
SMB1432~SMB1447	第二十八个模块(Rack3, Slot6)描述信息
SMB1432~SMB1434	实际模块信息
SMB1435	模块版本号
SMD1436	模块错误计数
SMB1440~SMB1447	保留
SMB1448~SMB1463	第二十九个模块(Rack3, Slot7)描述信息
SMB1448~SMB1450	实际模块信息
SMB1451	模块版本号
SMD1452	模块错误计数
SMB1456~SMB1463	保留
SMB1464~SMB1479	第三十个模块(Rack3, Slot8)描述信息
SMB1464~SMB1466	实际模块信息
SMB1467	模块版本号
SMD1468	模块错误计数
SMB1472~SMB1479	保留
SMB1480~SMB1495	第三十一个模块(Rack3, Slot9)描述信息
SMB1480~SMB1482	实际模块信息
SMB1483	模块版本号
SMD1484	模块错误计数
SMB1488~SMB1495	保留
SMB1496~SMB1511	第三十二个模块(Rack3, Slot10)描述信息
SMB1496~SMB1498	实际模块信息
SMB1499	模块版本号
SMD1500	模块错误计数
SMB1504~SMB1549	内部使用
SMB1550~SMB1589	本机 CANOpen 信息
SMB1550	主站状态 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行 0xFF: 组态数据出错
SMB1551	从站1状态 (按节点 ID 排列) 0x00: 初始化 0x01: 断开连接 0x04: 停止

	0x05: 运行 0x7F: 预运行
SMB1552~SMB1581	从站2状态（按节点 ID 排列）~从站31状态（按节点 ID 排列）
SMB1582	从站32状态（按节点 ID 排列） 0x00: 初始化 0x01: 断开连接 0x04: 停止 0x05: 运行 0x7F: 预运行
SMB1583~SMB2019	内部使用
SMD2020	扩展模块中断次数
SMB2024~SMB2027	内部使用
SMD2028	本地中断次数
SMB2032~SMB2047	内部使用

E 中断事件优先级列表

优先级别群组	中断事件号码	优先级别组	说明
通讯和诊断事件 （最高优先级）	8	0	端口0: 接收字符
	9	0	端口0: 传输完成
	23	0	端口0: 接收信息完成
	24	0	端口1: 接收信息完成
	25	0	端口1: 接收字符
	26	0	端口1: 传输完成
	34	0	100微秒时基定时中断0
	35	0	100微秒时基定时中断1
	36	0	模块诊断事件中断（预留）
离散（中等优先级）	0	1	上升边缘, I0.0
	2	1	上升边缘, I0.1
	4	1	上升边缘, I0.2
	6	1	上升边缘, I0.3
	1	1	下降边缘, I0.0
	3	1	下降边缘, I0.1
	5	1	下降边缘, I0.2
	7	1	下降边缘, I0.3
	12	1	HSC0 CV=PV（预留）
	27	1	HSC0方向改变（预留）
	28	1	HSC0外部复原 / Zphase（预留）
	13	1	HSC1 CV=PV（预留）
	14	1	HSC1方向改变（预留）
	15	1	HSC1外部复原（预留）
	16	1	HSC2 CV=PV（预留）
	17	1	HSC2方向改变（预留）

	18	1	HSC2外部复原（预留）
	19	1	PTO 0 完成中断（预留）
	20	1	PTO 1 完成中断（预留）
	32	1	HSC3 CV=Pv（预留）
	29	1	HSC4 CV=Pv（预留）
	30	1	HSC4方向改变（预留）
	31	1	HSC4外部复原 / Z phase（预留）
	33	1	HSC5 CV=Pv（预留）
定时（最低优先级）	10	2	定时中断0
	11	2	定时中断1
	21	2	定时器 T32 CT=PT 中断
	22	2	定时器 T96 CT=PT 中断

F 国际助记符

术语	国际助记符
语句表	STL
梯形图	LAD
功能块图	FBD
输入	I
输出	Q
模拟量输入	AI
模拟量输出	AQ
定时器	T
计数器	C
高速计数器	HC
累加器	AC
SCR	S
变量存储区	V
位存储区	M
特殊存储区	SM
局部变量存储区	L



深圳市合信自动化技术有限公司



深圳市南山区打石一路深圳国际创新谷 6 栋 A 座 9 层



0755-86226822



sales@co-trust.com



www.co-trust.com



官方微信公众号