COTRUST

# CTH300-H 系列可编程逻辑控制器

# 用户手册

手册版本 V1.95
发布日期 04/2021

# 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、抄录或与其他产品捆绑销售，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

感谢您购买深圳市合信自动化技术有限公司的 CTH300-H 系列可编程控制器！

在使用我公司 CTH300-H 系列可编程控制器前，敬请您仔细阅读本手册，以便更清楚地掌握产品的特性，更安全地应用，充分利用本产品丰富的功能。

## 产品特性

CTH300-H 系列可编程逻辑控制器可配合多种扩展模块或单元构成功能强大的可编程逻辑控制器系统，该系统满足高端 OEM 设备及中小型工程项目的控制需求，具备高速、高精度、智能易用、通信互连能力强、中等 I/O 规模等特性。该控制器同时具备高安全性、大规模分布式控制的扩展能力，体积紧凑、接线方便可靠。

## 适用对象

本手册提供关于 CTH300-H 系列可编程控制器的技术规格、安装和调试信息，为工程师、安装人员、维护人员和具有自动化常识的电工而设计。

## 服务支持

深圳市合信自动化技术有限公司建立了售后维修服务中心，并提供电话热线服务。客户在产品使用过程中遇到问题时，可随时与深圳市合信自动化技术有限公司各地的服务支持联系。各地服务支持热线电话请到 http://www.co-trust.com/Company/Contact/index.html 获取。此外，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。

# 安全注意事项

CTH300-H 系列可编程控制器的安装、操作、维护工作仅限于合格人员执行。对于使用本资料所引发的任何后果，合信概不负责。

在尝试使用设备之前，请仔细阅读设备相关注意事项，务必遵守安装调试安全预防措施和操作程序。对错误使用设备而可能带来的危害和损害程度见下述符号说明。



**警告**

该标记表示

“由于没有按要求操作造成的危险，可能导致人身伤亡”



**注意**

该标记表示

“由于没有按要求操作造成的危险，可能会导致人身轻度或中度伤害和设备损坏”



**提示**

该标记表示

“对操作的描述进行必要的补充或说明”

# 版本修订记录

| 发行日期 | 修订后版本 | 修订内容 |
| --- | --- | --- |
| 2021 年 4 月 | V1.95 | • 增加 CPU 升级固件内容，详见章节 2.7 CPU 在线升级固件<br>• 新增连续插补功能，详见章节 13.3.3 轴组指令 相对位移直线插补指令、绝对位移直线插补指令、相对位移圆弧插补指令、绝对位移圆弧插补指令。章节 13.3.5 连续插补功能应用举例<br>• 增加 S7 主从站协议内容，详见章节 14.1.9 西门子 S7 协议通信举例<br>• 新增 C 程序块的使用案例，详见章节 15 C 程序块<br>• 完善原点回归、特殊原点回归指令内容，详见章节 13.3.1 单轴控制指令原点回归、特殊原点回归指令;<br>• 增加回原功能应用举例，详见章节 13.7.2 回原功能应用举例 |
| 2020 年 9 月 | V1.94 | 增加追踪视图功能描述，参见章节 I Trace 追踪功能 |
| 2020 年 5 月 | V1.93 | • 新增编码器轴配置功能，详见章节 13.1.1 轴配置<br>• 新增 CT_TCP_SERVER 库使用说明，详见章节 H CT_tcp_server_lib 库的使用介绍 |
| 2020 年 4 月 | V1.92 | • 新增回原模式描述，详见章节 13.4 回原功能介绍。<br>• 新增以太网 socket 库介绍，详见章节 G 以太网“ct_socket”通信库的使用介绍。<br>• 对章节 14 做了内容结构完善。 |
| 2019 年 12 月 | V1.91 | • 新增 H31-00 和 H32-01 规格参数，参见章节 5 主控模块。<br>• 完善主控模块 Axis 指令说明，参见章节 14.3 主控模块 Axis 指令说明。 |
| 2019 年 10 月 | V1.90 | • 新增 H36-01 规格参数，参见章节 5 主控模块。<br>• 增加章节 13.13 轴配置及电子凸轮和章节 13.14 H3X-01 主控模块 Axis 指令说明。 |
| 2019 年 8 月 | V1.84 | • 新增章节 2.3 硬件组态。<br>• 完善章节 13.5 Modbus TCP 通信和章节 13.6 UDP_PPI 通信，更新部分操作截图。 |

# 目录


权利声明 ..... II

前 言 ..... III

安全注意事项 ..... IV

版本修订记录 ..... V

**1 系统概述 ..... 1**

本章概要 ..... 1

1.1 CTH300-H 系列产品及编程软件 ..... 1

1.1.1 CTH300-H 系列主控模块 ..... 2

1.1.2 CTH300-H 系列扩展模块 ..... 3

1.1.3 MagicWorks PLC 编程软件 ..... 4

1.2 系统架构 ..... 4

1.3 电磁兼容性 ..... 6

1.4 运输、存储和运行的环境条件 ..... 7

**2 使用入门 ..... 8**

2.1 连接主控模块 ..... 8

2.2 设置通讯 ..... 9

2.3 硬件组态 ..... 10

2.4 编写程序 ..... 12

2.5 编译并下载程序 ..... 13

2.6 运行程序 ..... 14

2.7 CPU 在线升级固件 ..... 14

**3 安装 ..... 16**

本章概要 ..... 16

3.1 安装注意事项 ..... 16

3.2 安装尺寸 ..... 18

3.3 使用机架 ..... 19

3.4 安装方法 ..... 21

3.5 接地和布线 ..... 22

3.6 抑制电路 ..... 23

**4 电源模块 ..... 25**

本章概要 ..... 25

4.1 技术规范 ..... 25

4.1.1 常规特性 ..... 25

4.1.2 功能特性 ..... 25

4.2 接线规格 ..... 26
4.2.1 接口示意图 ..... 26
4.2.2 接口定义 ..... 27
**5 主控模块 ..... 28**
本章概要 ..... 28
5.1 技术规范 ..... 28
5.1.1 常规特性 ..... 28
5.1.2 功能特性 ..... 29
5.1.3 通信口规范 ..... 30
5.1.4 数据存储器规格 ..... 31
5.1.5 密码级别及权限控制 ..... 34
5.1.6 实时时钟功能 ..... 34
5.1.7 中断事件 ..... 36
5.2 接线规格 ..... 37
5.2.1 接口示意图 ..... 37
5.2.2 接口定义 ..... 37
5.2.3 制作标准网线 ..... 39
**6 数字量模块 ..... 41**
本章概要 ..... 41
6.1 数字量输入模块 ..... 41
6.1.1 技术规范 ..... 41
6.1.2 接线规格 ..... 42
6.2 数字量输出模块 ..... 43
6.2.1 技术规范 ..... 43
6.2.2 接线规格 ..... 44
**7 模拟量模块 ..... 45**
本章概要 ..... 45
7.1 通用特性 ..... 45
7.1.1 常规特性 ..... 45
7.1.2 功能特性 ..... 46
7.2 模拟量输入模块 ..... 46
7.2.1 技术规范 ..... 46
7.2.2 接线规格 ..... 47
7.3 模拟量输出模块 ..... 48
7.3.1 技术规范 ..... 48
7.3.2 接线规格 ..... 49
7.4 模拟量输入输出模块 ..... 49
7.4.1 技术规范 ..... 49
7.4.2 接线规格 ..... 51

**8 温度模块 ..... 52**

本章概要 ..... 52
8.1 技术规范 ..... 52
8.1.1 常规特性 ..... 52
8.1.2 功能特性 ..... 53
8.1.3 输入特性 ..... 53
8.1.4 热电偶特性 ..... 55
8.1.5 热电阻特性 ..... 56
8.2 接线规格 ..... 57
8.2.1 热电偶接线规格 ..... 57
8.2.2 热电阻接线规格 ..... 58

**9 高速计数模块 ..... 60**

本章概要 ..... 60
9.1 技术规范 ..... 60
9.2 接线图 ..... 61
9.3 软件配置规格 ..... 61
9.3.1 设置计数器指令 ..... 62
9.3.2 设置模式指令 ..... 63
9.3.3 获取当前计数值指令 ..... 65
9.3.4 获取当前计数状态指令 ..... 65
9.3.5 获取当前速度指令 ..... 66
9.3.6 获取当前锁存值指令 ..... 66
9.3.7 清除锁存值指令 ..... 67

**10 脉冲输出模块 ..... 68**

本章概要 ..... 68
10.1 技术规范 ..... 68
10.2 接线图 ..... 69
10.3 软件配置规格 ..... 70
10.3.1 单轴相对运动指令 ..... 70
10.3.2 单轴绝对运动指令 ..... 72
10.3.3 速度控制指令 ..... 74
10.3.4 设置当前绝对坐标 ..... 76
10.3.5 获取当前绝对位置指令 ..... 77
10.3.6 设置电机方向指令 ..... 77
10.3.7 设置轴输出模式 ..... 78

**11 CAN 主站模块 ..... 80**

本章概要 ..... 80
11.1 技术规范 ..... 80
11.1.1 常规特性 ..... 80

11.1.2 功能特性 ..... 81
11.1.3 通信口规范 ..... 81
11.2 接线规格 ..... 82
11.2.1 接口示意图 ..... 82
11.2.2 接口定义 ..... 82
**12 中继模块 ..... 84**
本章概要 ..... 84
12.1 技术规范 ..... 84
12.1.1 常规特性 ..... 84
12.1.2 功能特性 ..... 85
12.2 接线规格 ..... 85
12.2.1 接口示意图 ..... 85
12.2.2 接口定义 ..... 86
12.2.3 中继模块网线制作 ..... 86
**13 轴配置及电子凸轮 ..... 87**
本章概要 ..... 87
13.1 轴配置向导 ..... 87
13.1.1 轴配置 ..... 92
13.1.2 轴组配置 ..... 100
13.2 电子凸轮向导 ..... 101
13.3 轴指令“ct_plcopen_lib(v2.1)”介绍 ..... 105
13.3.1 单轴控制指令 ..... 106
13.3.2 同步控制指令 ..... 124
13.3.3 轴组指令 ..... 134
13.3.4 错误代码 ..... 147
13.3.5 连续插补功能应用举例 ..... 148
13.3.6 电子凸轮程序示例 ..... 154
13.4 回原功能介绍 ..... 156
13.4.1 回原模式模式原理介绍 ..... 157
13.4.2 回原功能应用举例 ..... 160
**14 使用指南 ..... 175**
本章概要 ..... 175
14.1 通信功能应用示例 ..... 175
14.1.1 总线通信 ..... 175
14.1.2 PPI/MPI 通信 ..... 180
14.1.3 Modbus RTU 通信 ..... 186
14.1.4 Modbus TCP 通信 ..... 190
14.1.5 UDP_PPI 通信 ..... 201
14.1.6 远程 EtherNET 通信 ..... 209
14.1.7 CANopen 通信 ..... 212

14.1.8 EtherCAT 通信 ..... 217
14.1.9 西门子 S7 协议通信举例 ..... 223
14.2 为主控模块 RS485 口选择通信协议 ..... 249
14.3 高速计数模块应用示例 ..... 250
14.3.1 示例组件 ..... 250
14.3.2 网络连接 ..... 250
14.3.3 操作步骤 ..... 251
14.4 HSC 中断示例 ..... 255
14.4.1 硬件组态 ..... 255
14.4.2 中断例程 ..... 256
14.5 故障诊断 ..... 257
14.5.1 通过编程软件进行诊断 ..... 257
14.5.2 通过 PLC 状态指示灯进行诊断 ..... 272
**15 C 程序块 ..... 273**
本章概要 ..... 273
15.1 在 Magicworks PLC 中的 C 语言编程步骤 ..... 273
15.2 C 语言基础 ..... 278
15.2.1 头文件 ..... 278
15.2.2 变量 ..... 279
15.2.3 注释的使用 ..... 280
15.2.4 各类数值型数据间的混合运算 ..... 280
15.2.5 强制类型转换 ..... 281
15.2.6 数组 ..... 282
15.3 运算符与表达式 ..... 283
15.3.1 赋值运算符与赋值表达式 ..... 283
15.3.2 算术运算符 ..... 284
15.3.3 自增自减运算符 ..... 284
15.3.4 关系运算符与表达式 ..... 284
15.3.5 逻辑运算符与表达式 ..... 285
15.3.6 条件运算符与条件表达式 ..... 286
15.4 for 循环语句 ..... 286
15.5 选择语句 ..... 287
15.5.1 if 选择结构语句 ..... 287
15.5.2 switch 多分支选择语句 ..... 290
15.6 libplc300.a 库介绍 ..... 291
15.6.1 CF 块参数符号表中可选择的数据类型 ..... 291
15.6.2 大小端转换 ..... 292
15.6.3 STL 系统与 C 语言系统的数据转换操作函数 ..... 292
15.6.4 获取内存基地址的操作函数 ..... 296
**附录 ..... 297**
A 订货信息 ..... 297

B 缩略术语 ..... 298
C 电源预算 ..... 300
D 编程卡（U 盘）使用说明 ..... 301
D.1 编程卡下载 ..... 302
D.2 编程卡上载 ..... 303
E CT_Modbus 库的使用介绍 ..... 304
E.1 CT_Modbus 库介绍 ..... 304
E.2 安装 CT_Modbus 库文件 ..... 304
E.3 CT_Modbus_RTU 库功能说明 ..... 305
E.4 CT_Modbus_master_tcp_single 库功能说明 ..... 311
F PID_T 库的使用介绍 ..... 313
G 以太网“ct_socket”通信库的使用介绍 ..... 316
G.1 ct_socket 库介绍 ..... 316
G.2 使用说明 ..... 316
H CT_tcp_server_lib 库的使用介绍 ..... 320
H.1 TCP_Server ..... 320
H.2 TCP_Connect ..... 321
H.3 TCP_Send ..... 322
H.4 TCP_Recv ..... 323
I Trace 追踪功能 ..... 324
J 特殊存储器说明 ..... 328
K 指令速查 ..... 343

# 1 系统概述

CTH300-H 系列可编程逻辑控制器可配合多种扩展模块或单元构成功能强大的可编程逻辑控制器系统，该系统满足高端 OEM 设备及中小型工程项目的控制需求，具备高速、高精度、智能易用、通信互连能力强、中等 I/O 规模等特性。

该控制器同时具备高安全性、大规模分布式控制的扩展能力，体积紧凑、接线方便可靠。

### 本章概要

本章主要介绍 CTH300-H 系统的常规技术数据，主要内容包括：

- ❑ CTH300-H 系列产品及编程软件
- ❑ 系统架构
- ❑ 电磁兼容性
- ❑ 运输、存储和运行的环境条件

## 1.1 CTH300-H 系列产品及编程软件

本节将分别对 CTH300-H 系列主控模块及其扩展单元、配套编程软件 MagicWorks PLC 进行概述。

CTH300-H 系列主控模块及其扩展单元的功能简述如下：

- **电源模块（PWR）**

为 CTH300-H 系列可编程控制器提供 24V DC 电源。

- **主控模块（CPU）**

CTH300-H 系统的中央处理单元，用于处理各种运算以及用户程序的运行，作为 CTH300-H 系统的核心控制装置。

- **信号模块（SM）**

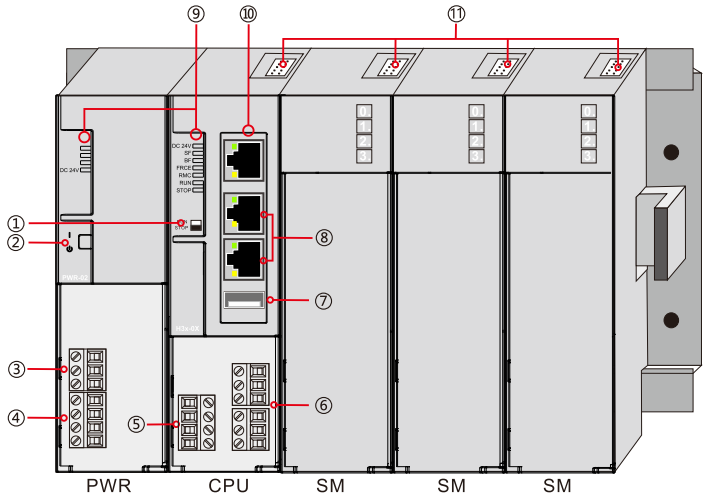
用于输出数字信号/模拟信号给现场设备或者接收现场设备输入的数字信号/模拟信号。

- **CAN 主站通信模块（CP）**

用于与其它 CAN 从站模块进行 CANopen 通信。

- **中继模块（INT）**

用于多机架配置时连接主机架和扩展机架，CTH300-H 系列 PLC 通过分布式中央机架和 3 个扩展机架最多可挂接 32 个模块。




① 系统运行开关　⑤ 电源接线端子　⑨ LED状态指示灯
② 24VDC电源开关　⑥ RS485信号接线端子　⑩ 以太网通信口
③ 220VAC输入电源接线端子　⑦ USB接口　⑪ 总线接口
④ 24VDC输出电源接线端子　⑧ RS485/EtherCAT/CANopen通信口

### 1.1.1 CTH300-H 系列主控模块

CTH300-H 系列主控模块是 CTH300-H 系统的中央处理单元，用于处理各种运算以及用户程序的运行，作为 CTH300-H 系统的核心控制装置。以下为主控模块的外观示意图：

| | CTH300-H 系列主控模块型号 |
|---|---|
|  | CTH3 H35-000S1 |
| | CTH3 H36-000S1 |
| | CTH3 H36-001S2 |
| | CTH3 H31-000S1 |
| | CTH3 H32-001S2 |

CTH300-H 系列主控模块的特性描述如下：

◆ **运算速度快**

CPU 的逻辑指令执行速度为 0.086μs，浮点运算速度为 1.4μs；CPU 采用 Cortex-A8 内核处理器，主频最高可达 1GHz。

◆ **高速总线**

CPU 和扩展模块之间的高速总线采用 M-LVDS 技术，数据传输速率达到 55Mbps。

◆ **强大的系统扩展能力**

可扩展的最大数字量 I/O 点数为 1024 点，可扩展的最大模拟量 I/O 点数为 256 点。

◆ **支持多种通信协议**

CPU 本机支持 EtherNET、EtherCAT、CANopen、PPI/MPI、Modbus，S7 等通信协议，还可以通过扩展模块支持 CANopen 高速现场总线。

◆ **编程方便快捷**

使用 MagicWorks PLC 编程软件（V2.00 或更高版本）进行编程；支持 C 语言编程。CPU 自带 USB 接口，可通过 U 盘下载程序。

◆ **支持远程功能**

支持远程 Ethernet 通信，可进行程序的远程监控、调试，以及支持远程更新固件。

◆ **完善的运控功能**

支持单轴运动功能、多轴插补功能、电子凸轮及电子齿轮功能。

### 1.1.2 CTH300-H 系列扩展模块

CTH300-H 系列主控模块属于 CTH300 系列中型 PLC 家族，它可以支持 CTH300-H 系列扩展模块，其模块类型包括但不仅限于：数字量输入输出、模拟量输入输出、温度采集、高速计数、CAN 通信模块等，具体模块及其型号请参考下表：

表 1-1 CTH300-H 系列扩展模块

| | | | |
|---|---|---|---|
|  | 数字量输入模块<br>CTH3 DIT-080S1<br>CTH3 DIT-160S1<br>CTH3 DIT-320S1 | 数字量输出模块<br>CTH3 DQT-080S1<br>CTH3 DQT-160S1<br>CTH3 DQT-320S1<br>CTH3 DQR-080S1<br>CTH3 DQR-160S1 | |
|  | 模拟量输入模块<br>CTH3 AIS-040S1<br>CTH3 AIV-080S1<br>CTH3 AIC-080S1 | 模拟量输出模块<br>CTH3 AQS-040S1<br>CTH3 AQS-080S1<br>模拟量输入输出模块<br>CTH3 AMS-060S1 | 温度模块<br>CTH3 AIT-040S1<br>CTH3 AIT-080S1<br>CTH3 AIR-040S1<br>CTH3 AIR-080S1 |

| | | | | | |
|---|---|---|---|---|---|
|  | 高速计数模块<br>CTH3 HSC-020S1 |  | 脉冲输出模块<br>CTH3 HSP-040S1 |  | CAN 主站模块<br>CTH3 CAN-1M0S1 |

| | | | |
|---|---|---|---|
|  | 电源模块<br>CTH3 PWR-020S1 |  | 中继模块<br>CTH3 INT-000S1 |

**<备注>** 有关各扩展模块的描述请参考第 6~12 章节。

### 1.1.3 MagicWorks PLC 编程软件

随 CTH300-H 系列产品附赠的 MagicWorks PLC 软件，为用户开发、编辑和监控自己的应用程序提供了良好的编程环境；提供两种程序编辑器以便您快捷高效地开发您的应用程序，集成的在线帮助功能方便您找到所需的信息。

**软件特性**

- 支持 IEC61131 编程语言
- 支持编程语言：LAD（梯形图）、STL（指令列表）、C 语言
- 支持中文/英文编程
- 智能化帮助
- 指令集丰富

**计算机配置要求**

操作系统：Windows 98、Windows 2000、Windows XP、Vista、Windows 7 32bit。

硬件配置：IBM 486 以上兼容机，内存 128MB 以上，至少 500MB 以上硬盘剩余空间。

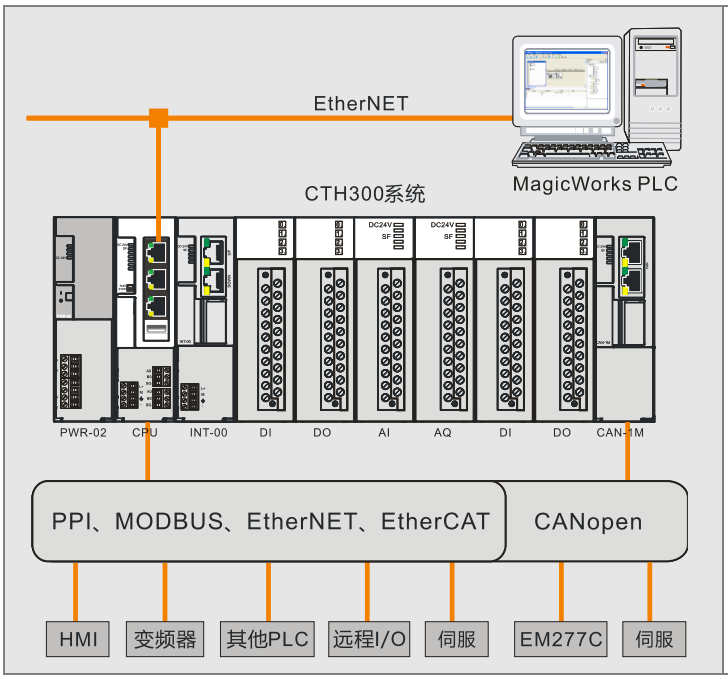


**提示**

具体使用方法请参考 MagicWorks PLC 软件的在线帮助或《MagicWorks PLC 用户手册》。
手册和软件均可于合信官网免费下载，下载地址：http://www.co-trust.com/Download/index.html

## 1.2 系统架构

CTH300-H 系统的典型架构参考以下图所示：




**编程**

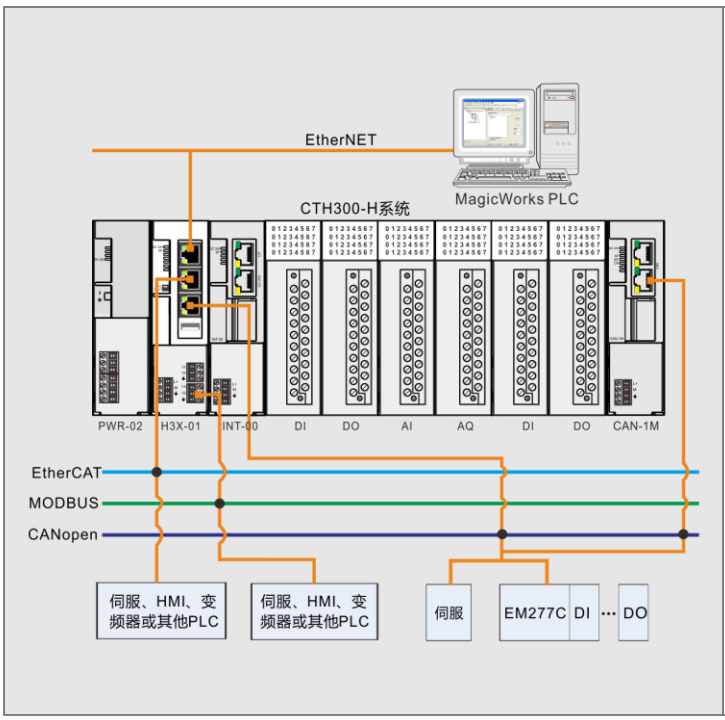
通过 H3X-00 的 EtherNET 通信口或 RS485 通信口与上位机进行通信。

**CTH300-H 系统架构**

最多可由一个中央机架和三个扩展机架组成，最多可安装 32 个扩展单元，机架的扩展详情见 3.3 使用机架

**通信**

通过 H3X-00 的 RS485、EtherCAT、EtherNET 通信口与 HMI、远程 I/O 或其他 PLC 等进行通信；H3X-00 通过总线扩展 CAN-1M 模块进行 CANopen 通信。




**编程**

通过 H3X-01 的 EtherNET 通信口或 RS485 通信口与上位机进行通信。

**CTH300-H 系统架构**

最多可由一个中央机架和三个扩展机架组成，最多可安装 32 个扩展单元，机架的扩展详情见 3.3 使用机架

**通信**

通过 H3X-01 的 RS485、EtherCAT、CANopen、EtherNET 通信口与 HMI、伺服、远程 I/O 或其他 PLC 等进行通信；H3X-01 通过总线扩展 CAN-1M 模块进行 CANopen 通信。

在应用系统中，各种 I/O 扩展模块通过总线接口挂接在 CTH300-H 系列主控模块或中继模块 INT-00 的后方。I/O 扩展模块把采集到的各种数据和诊断信息通过总线发送给主控模块，主控模块再根据用户程序以及相关信息进行处理。

## 1.3 电磁兼容性

电磁兼容性（EMC）是指电气设备在其电磁环境中正常运行且不干扰环境的能力。下表说明了 CTH300-H 系列 PLC 应当遵循的标准或规范。

表 1-2 可编程逻辑控制器标准：IEC61131-2，GB15969

| **电磁兼容性-抗扰度** | |
|---|---|
| 静电放电 IEC61000-4-2 | 接触放电：±4KV<br>空气放电：±8KV |
| 电快速瞬变脉冲群<br>IEC61000-4-4 | 电源线：2KV，5KHz<br>信号线：2KV，5KHz（I/O 耦合夹）<br>1KV，5KHz（通讯耦合夹） |
| 浪涌 IEC61000-4-5 | 电源线：2KV（非对称），1KV（对称） |
| 射频电磁场辐射<br>IEC61000-4-3 | 80MHz~1GHz，10V/m，80%AM（1KHz）<br>1.4GHz~2GHz，3V/m，80%AM（1KHz）<br>2GHz~2.7GHz，1V/m，80%AM（1KHz） |
| 射频场感应传导干扰<br>IEC61000-4-6 | 0.15MHz~80MHz，10V/m，80%AM（1KHz） |
| 直流电源输入端口短时中断和电压变化<br>IEC61000-4-29 | 短时中断：10ms<br>电压变化：80%~120%，100ms |
| **环境测试** | |
| 高温运行 IEC60068-2<br>低温运行 IEC60068-2 | 60℃ 16 小时<br>-10℃ 16 小时 |
| 高温启机 IEC60068-2<br>低温启机 IEC60068-2 | 60℃ 2 小时<br>-10℃ 2 小时 |
| 高低温循环运行 IEC60068-2 | -10℃ ~60℃ 驻留时间 3 小时，温升速率 1℃/min，2 个循环 |
| 高温存储 IEC60068-2<br>低温存储 IEC60068-2 | 70℃ 72 小时<br>-40℃ 72 小时 |
| 冷热冲击 IEC60068-2 | -40℃~70℃ 驻留时间 3 小时，温变时间<1min，5 个循环 |
| 高温高湿 IEC60068-2 | 40℃ 48 小时 |
| 交变湿热 IEC60068-2 | 25℃~55℃ 95% 2 个循环 |
| 正弦振动（裸机）IEC60068-2 | 5~150Hz，0.05G2/Hz<br>150Hz~500Hz -3dB/oct，1 小时/轴，X、Y、Z 总共 3 轴 |
| 冲击（裸机） IEC60068-2 | 15G，11ms 脉冲，3 次/方向 |
| 流动混合气体腐蚀试验<br>IEC60068-2-60 | H2S：0.1ppm，NO2：0.2ppm，CL2：0.02ppm，温度：30℃，湿度：75%，周期：4 天 |
| **高压绝缘测试** | |
| 24V/5V 标称电路间<br>110V/220V 电路对地<br>110V/220V 电路对 110V/220V 电路 | 500 VAC<br>1500 VAC<br>1500 VAC |

| | |
|---|---|
| 110V/220V 电路接到 24V/5V 电路 | 1500 VAC |

## 1.4 运输、存储和运行的环境条件

表 1-3 CTH300-H 系列模块符合的电气规范

| | | |
|---|---|---|
| **环境条件 -- 运输和存贮** | | |
| 温度 | | -40℃~+70℃ |
| 大气压 | | 1080 hPa~660 hPa（对应高度为-1000m~+3500m） |
| 相对湿度 | | 10%~95%，非结露 |
| 跌落 | | 1m，10 次，运输包装 |
| **环境条件 -- 工作** | | |
| 温度 | 水平安装位置 | 0℃~60℃ |
| | 垂直安装位置 | 0℃~40℃ |
| 大气压 | | 1080 hPa~795 hPa（对应高度为-1000m~+2000m） |
| 相对湿度 | | 10%~95%，非结露 |
| 恶劣环境<br>污染物浓度 | | 较低盐雾、潮湿、尘雾等环境<br>SO2<0.5ppm，相对湿度<60%，非结露<br>H2S<0.1ppm，相对湿度<60%，非结露 |

# 2 使用入门

本章基于具体实例，介绍如何创建一个包含 PLC 程序的简单工程，并将此程序下载到目标设备（本例使用的目标设备为 H35-00），运行并监控此程序。

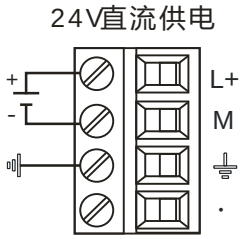
使用 LAD 语言编写一个示例程序：首次上电时，定时器 T33 开始计时，达 1 秒后停止计时；定时器 T37 在 T33 停止后开始计时，达到 1 秒后定时器 T37 停止计时；之后定时器 T33 和定时器 T37 在 1 秒间隔中进行交替计时操作。

## 2.1 连接主控模块

只需给 PLC 供电，然后在编程设备与 PLC 间连接通信电缆即可。

### ❑ 给 PLC 供电

下图为 PLC 的直流供电接线示意图：

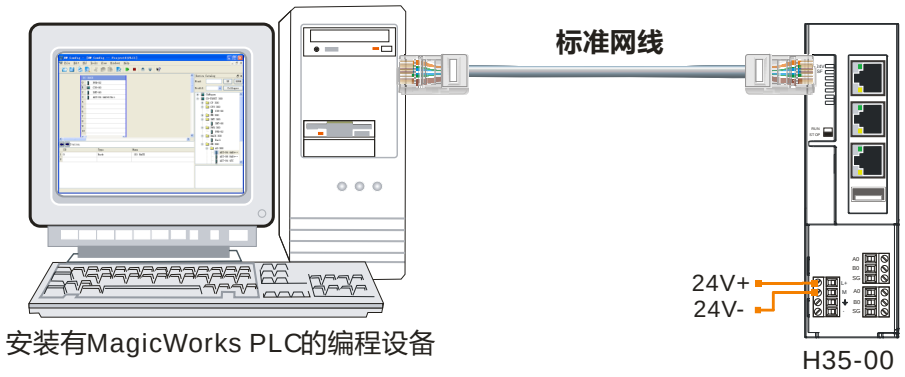



**警告**

安装或拆除 CTH300-H 系列 PLC 之前，必须遵守相应的安全防护规范，并务必将其电源断开。

### ❑ 连接电缆

使用标准网线连接编程设备 PG/PC 与 PLC 的 EtherNET 通信口（以 H35-00 为例）。



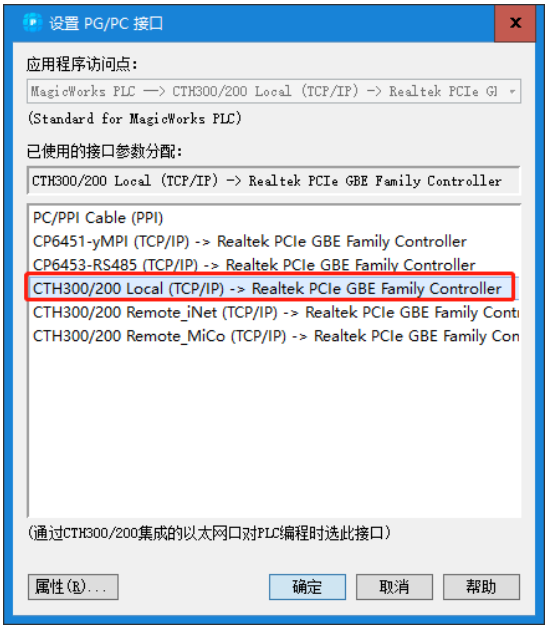

**<备注>** 请参考章节 5.2.3 制作标准网线制作标准网线。

## 2.2 设置通讯

双击桌面图标打开 MagicWorks PLC 软件，选择菜单项“文件”→“新建”新建一个项目，然后右键选中新建的项目选择“插入新对象”→“PLC”即可在该项目中插入一个 H35-00 站点。

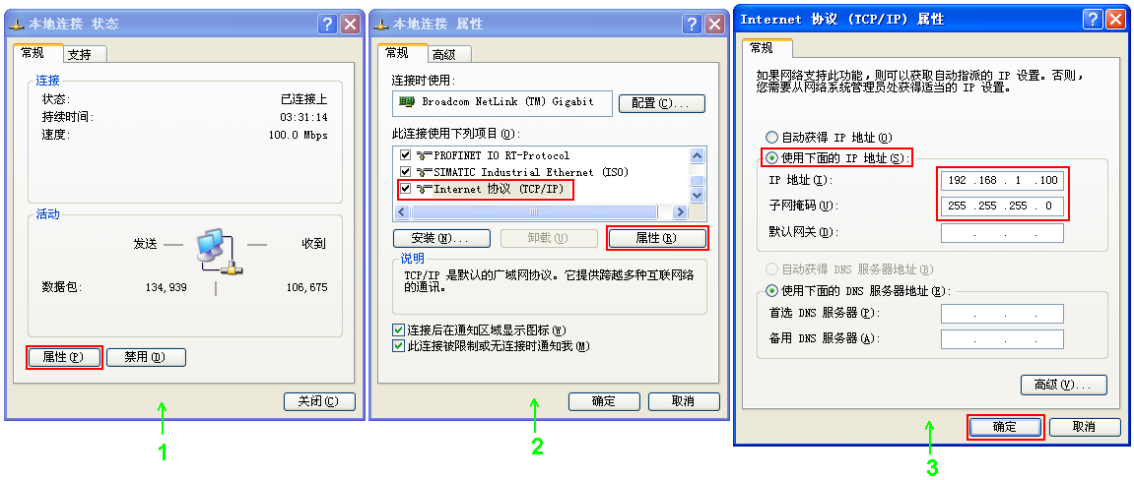
**1、设置 PG/PC 接口**

选择菜单项“工具”→“设置 PG/PC 接口”或在项目管理器界面选择“设置 PG/PC 接口”打开如下窗口。选择“CTH300/200 Local(TCP/IP)->Realtek PCIe GBE Family Controller”接口，点击“确定”。




**2、将编程设备的 IP 与 H35-00 的 IP 设为同一局域网**

通过 MagicWorks PLC 成功搜索到与编程设备相连的 H35-00 后，可记录该 H35-00 的 IP 地址，并将编程设备当前所用本地连接的 IP 地址与该 H35-00 的 IP 地址设置为同一个网段，设置方法可参考如下示例。本例将编程设备当前本地连接的 IP 改为 192.168.1.100，即与 H35-00（出厂 IP：192.168.1.201）处于同一个局域网。

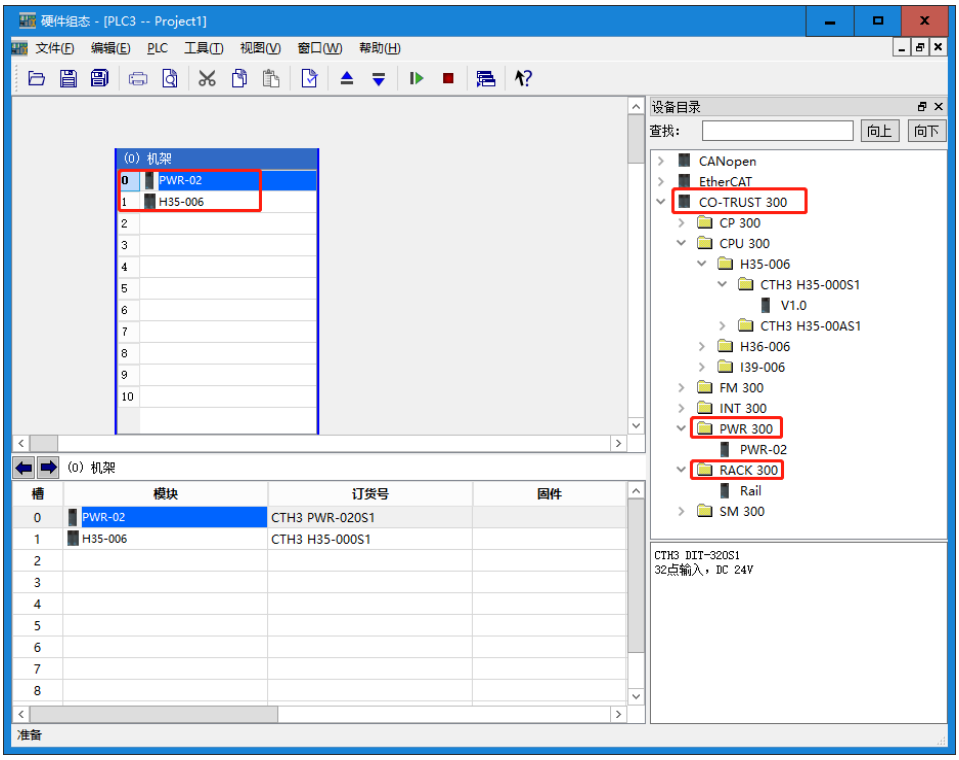


## 2.3 硬件组态

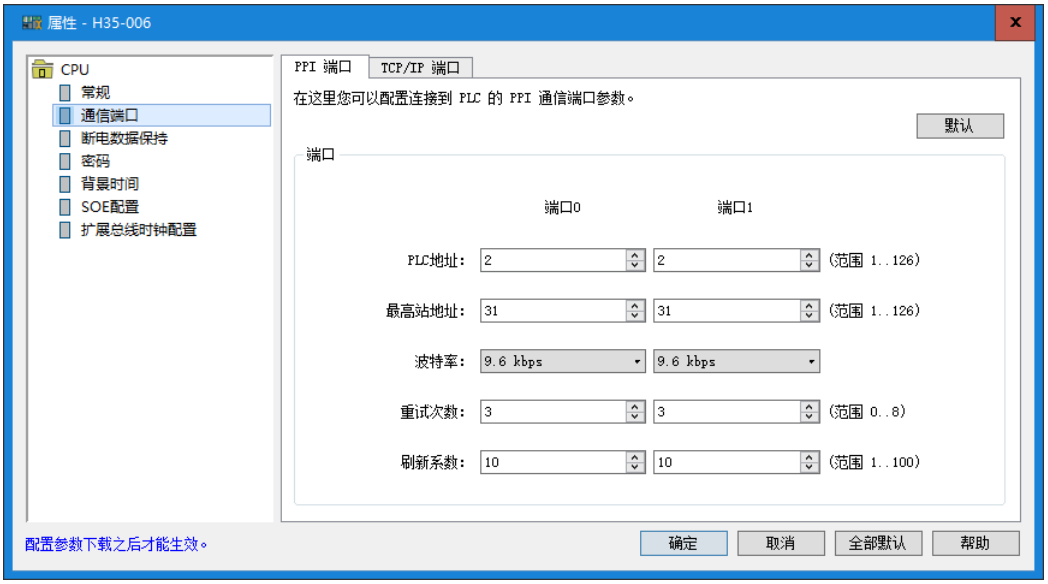
在 300CPU 站点的“项目管理器”界面选择“硬件组态”，进行硬件组态。

**步骤一：添加机架、电源模块和 CPU**

硬件组态界面中，在“CO-TRUST 300”项目树下顺序展开“RACK 300”、“PWR 300”和“CPU 300”文件夹，添加机架“Rail”、电源模块和 CPU，电源模块只能放至机架的 0 号槽内，CPU 只能放至机架的 1 号槽内，如下图所示。

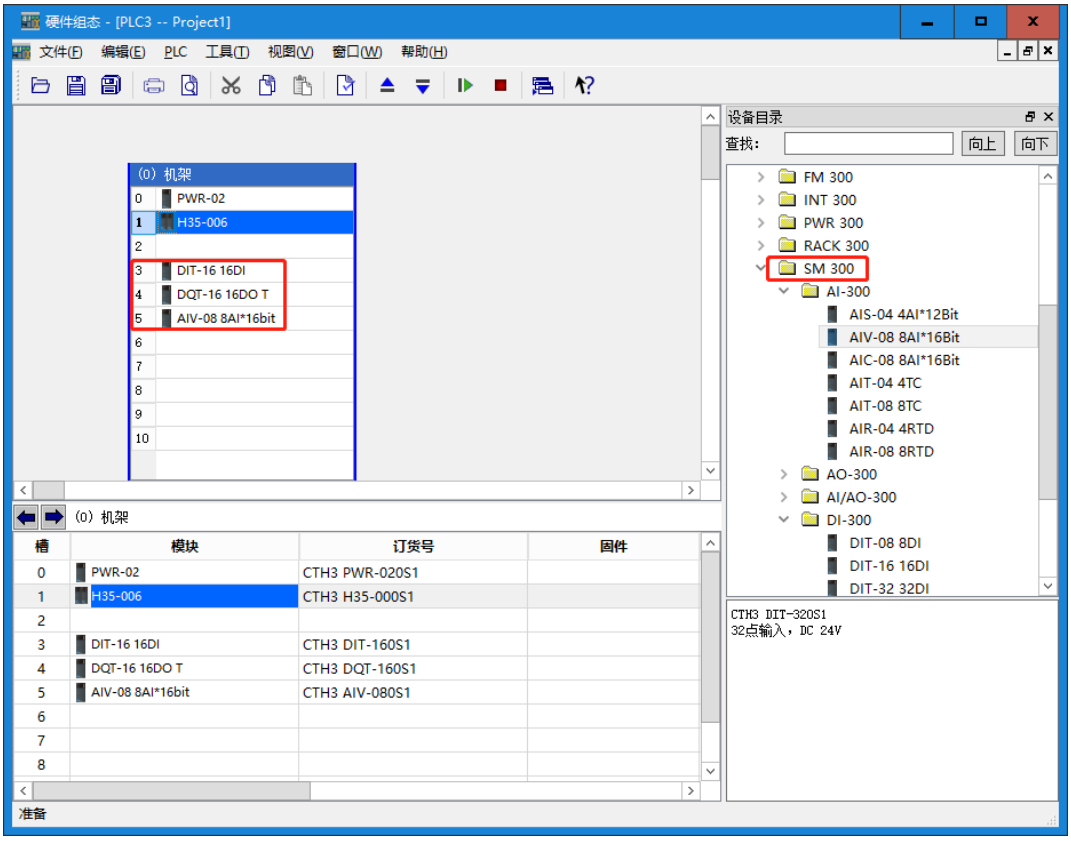



双击机架上的“H35-006”进行 CPU 属性配置，在属性界面选择“通信端口”可进行通信端口属性设置如下图所示。

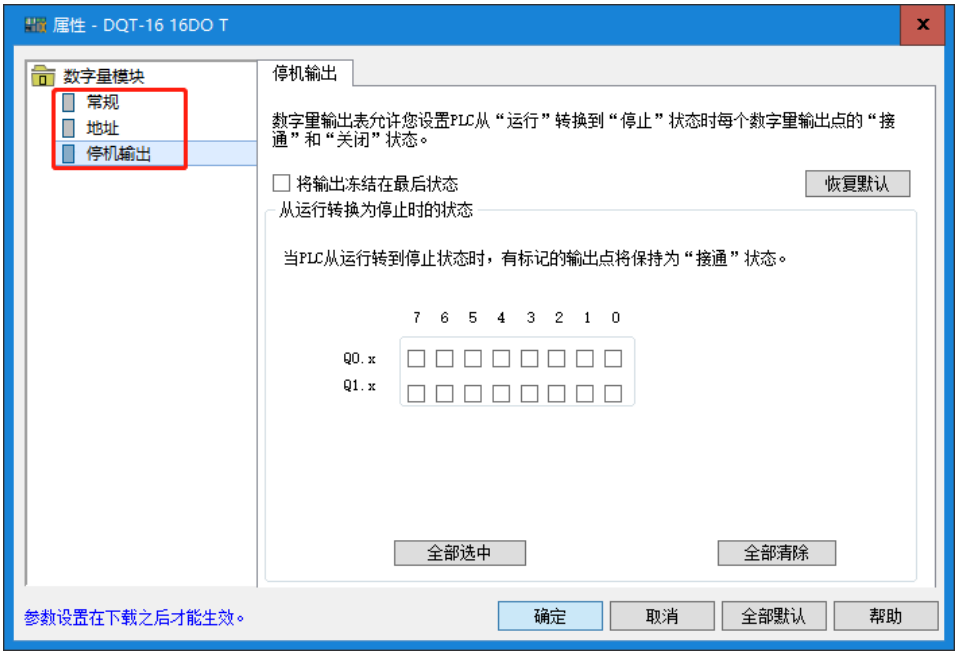


**步骤二：添加功能模块**

展开“SM 300”项目树选择所需模块添加至机架中，如下图所示。

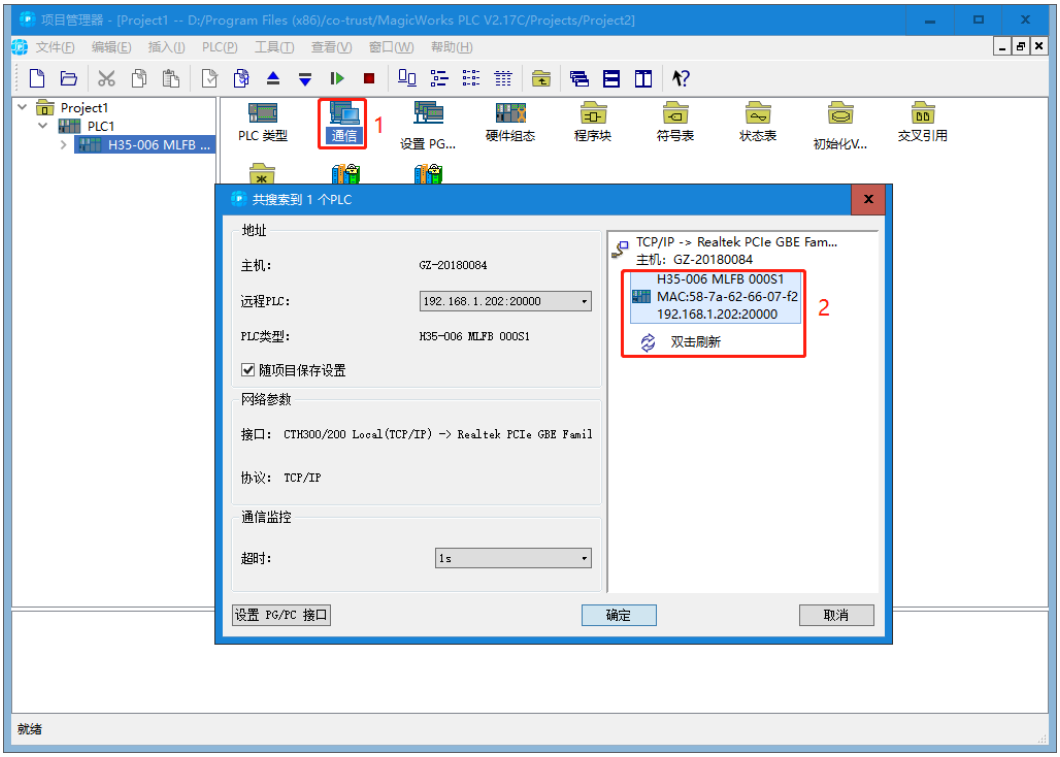



双击机架上的模块，可进行模块属性配置，模块属性配置界面如下。



**与 CPU 建立通讯**

完成以上设置后，在工作窗口双击通信图标弹出如下通信窗口，双击通信对话框中的图标进行搜索，搜索成功的 CPU 即会显示在通信对话框中。



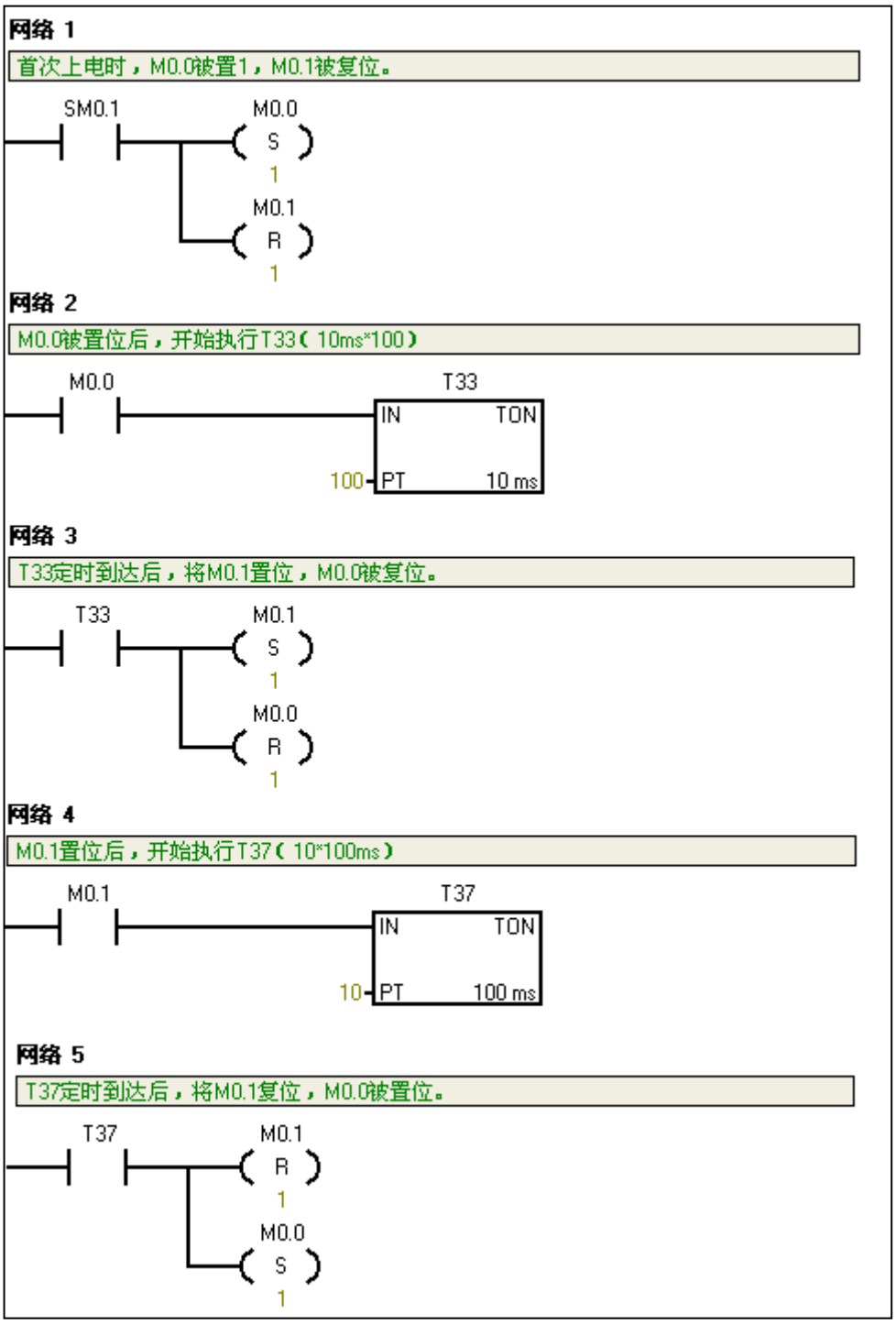

设置完成后，即可通过编程设备对 CPU 执行编译、下载、运行等操作。

## 2.4 编写程序

**本例程序实现目的：定时器 T33 和定时器 T37 在 1 秒间隔中交替计时。**

1）在项目树中展开“H35-006”，然后选中子项目“程序块”，在右侧窗口中双击 OB1 进入程序编辑窗口。

2）使用 LAD 语言编程，最终实现的程序如下图所示：




## 2.5 编译并下载程序

**1、保存并编译组态**

选择菜单项“文件”→“保存”以保存当前组态，然后选择菜单项“PLC”→“编译”对当前工程进行编译；若编译成功，则可以进行下载操作。

**2、将程序下载到 CPU 中**

在主界面选择菜单项“PLC”→“下载”将程序块和硬件组态从编程设备下载到 CPU 中。若您的 CPU 处于运行模式，下载界面将弹出一个对话框提示您将 CPU 置于 STOP 模式。

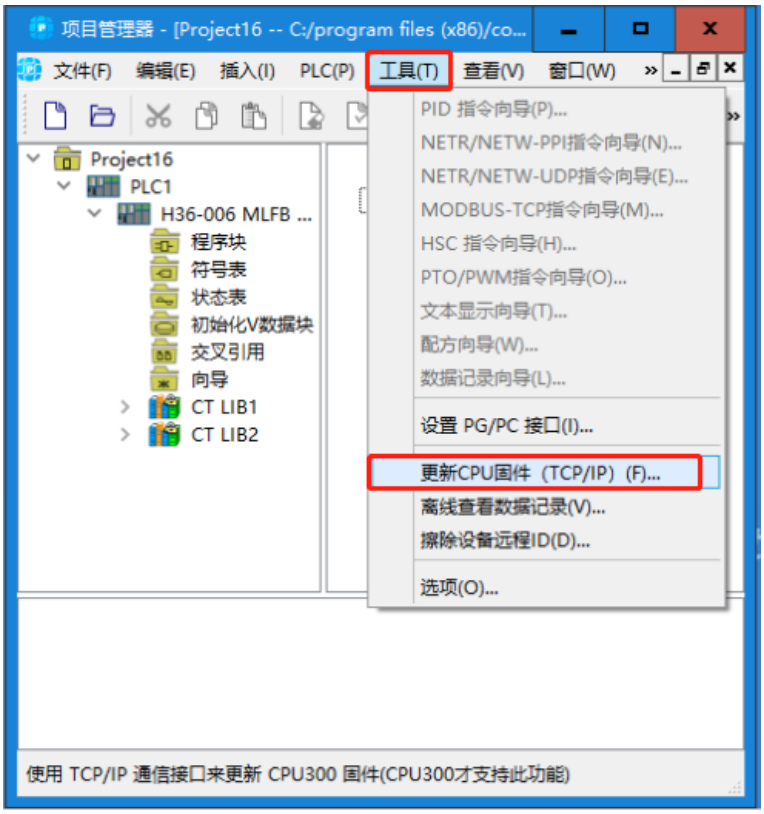
## 2.6 运行程序

程序成功下载至 CPU 后，将 CPU 的系统运行开关拨到 RUN，随后您即可通过状态表监控程序的运行情况。

**<备注>** 当系统出现故障时，请参考章节 14.5 故障诊断获取 CTH300-H 系列主控模块的诊断方法。

## 2.7 CPU 在线升级固件

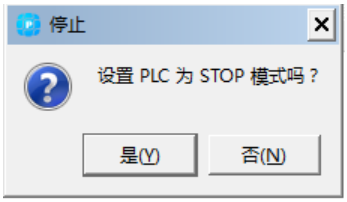
1、CTH300-H 系列 PLC 支持在线升级固件，进行在线更新固件前，需保证 CPU 已连入网络，在编程软件项目管理器界面选择“工具”→“更新 CPU 固件（TCP/IP）”打开“更新 CPU 固件”界面，如下图所示。



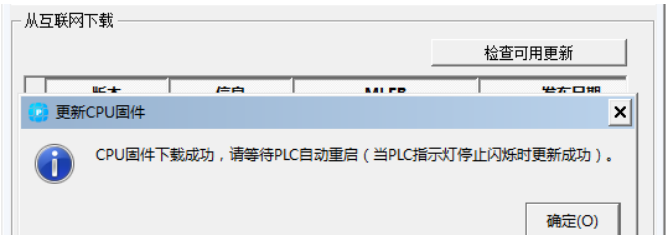

2、更新 CPU 固件”界面选择“从互联网下载”，点击“检查可用更新”，信息窗口将显示当前 PLC 可选用的固件，如下图所示，选中需更新的固件点击“安装”，固件将自动从互联网上下载。




3、开始前将出现如下弹窗，选择“是”，然后 CPU 将自动加载固件。




4、加载完成后，将出现以下提示，点击“确定”。注意，PLC 固件加载完成前切勿对 PLC 进行操作或断开电源！




5、完成后需对 PLC 进行断电重启操作。

# 3 安装

CTH300-H 系列 PLC 的外形设计使其极易安装。在现场可以利用安装孔把模块固定在控制柜的背板上，或者利用设备上的 DIN 夹子，把模块固定在一个标准（DIN）的导轨上。

### 本章概要

本章介绍 CTH300-H 系列 PLC 的安装，主要内容包括：

- ❑ 安装注意事项
- ❑ 安装尺寸
- ❑ 使用机架
- ❑ 安装方法
- ❑ 接地和布线
- ❑ 抑制电路

## 3.1 安装注意事项

CTH300-H 系列 PLC 的安装须注意以下事项：

### ❑ 隔离 PLC 与加热装置、高电压和电子噪声

按照一般惯例，在安装设备器件时，总是把产生高电压和高电子噪声的设备与 CTH300-H 系列 PLC 这样的低压电子型设备分隔开。

在控制柜的背板上安排 CTH300-H 系列 PLC 时，应考虑把电子器件安排在控制柜中温度较低的区域内，因电子器件长期在高温环境下工作会缩短其无故障时间。

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

### ❑ 为散热和接线留出适当的空间

CTH300-H 系列 PLC 的设计采用自然对流散热方式，在模块的上下方都必须留有至少 30mm 的空间，以便于正常的散热。前面板与背板的板间距离也应保持至少 80mm。



**注意**

在垂直安装所允许的最高环境温度要比水平安装时低10℃ ，而且 CPU 应安装在所有扩展模块的下方。

在安装 CTH300-H 系列 PLC 时，应留出足够空间用于接线和连接通讯电缆。

图 3-1 显示的是安装在多个机架上的 CTH300-H CPU，其中显示了各机架与相邻组件、电缆槽、机柜之间的间距。通过电缆槽为模块接线时，屏蔽连接元件底部与电缆槽间的最小间距为 40 mm。

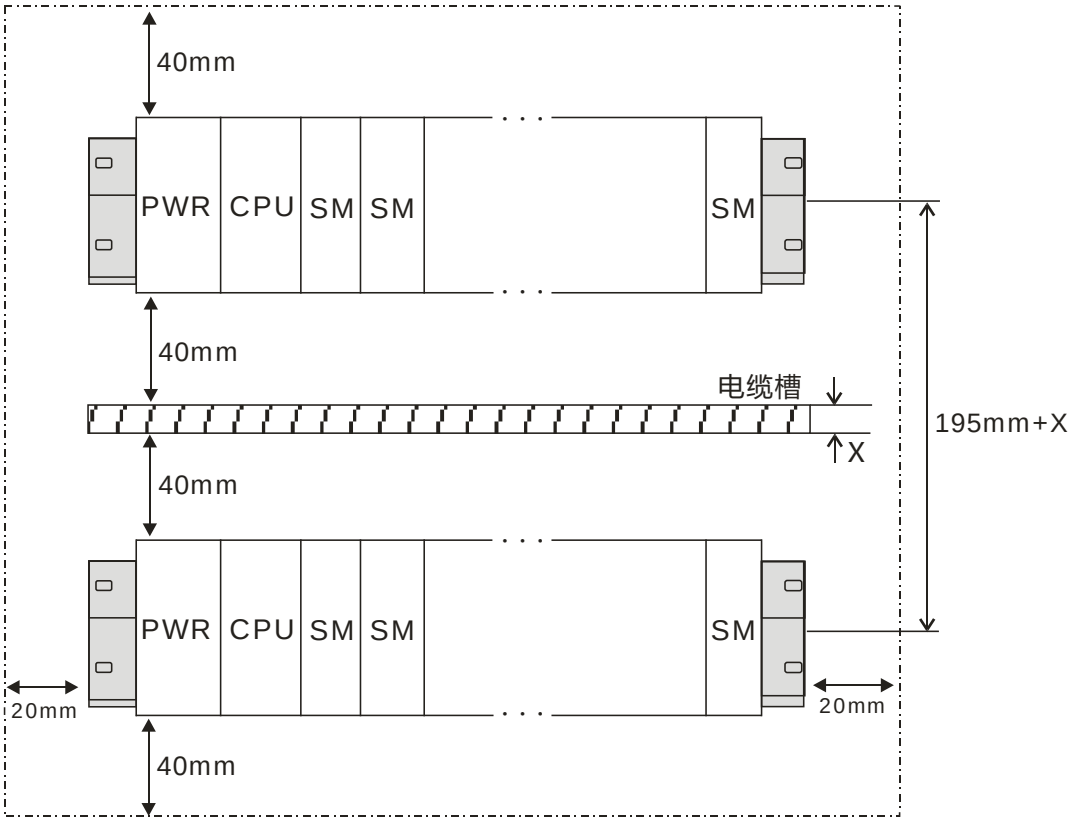



图 3-1　安装示意图

## ❑ 电源预算

选择好各机架的 CPU、电源模块、中继模块和扩展模块后，还需要确认系统总线的电流消耗和功率消耗是否满足以下条件：

**条件 1：总线电流消耗确认**

内部总线电压为 5VDC，电流由 CPU（无中继模块时）或中继模块提供。每个机架上扩展模块总线电流消耗之和不能超过 CPU 或中继模块允许的最大总线电流。

**条件 2：功率消耗确认**

使用电源模块时，每个机架上其它模块的功率消耗之和不能超过电源模块允许的最大功耗。

使用外部电源时，根据所接的功率之和选择合适功率大小的型号。

**<备注>** CTH300-H 系列 PLC 的总线电源电流的预算请参考附录 C 电源预算。

## ❑ 设备断电

在安装和拆卸 CTH300-H 系列 PLC 及其相关设备时，必须预先采取适当的安全措施并且确认 CTH300-H 系列 PLC 的供电被切断。

**警告**

在带电情况下安装或拆卸 CTH300-H 系列 PLC 及其相关设备有可能导致电击或设备误动作，进一步造成严重的人身伤害甚至死亡和设备损坏！

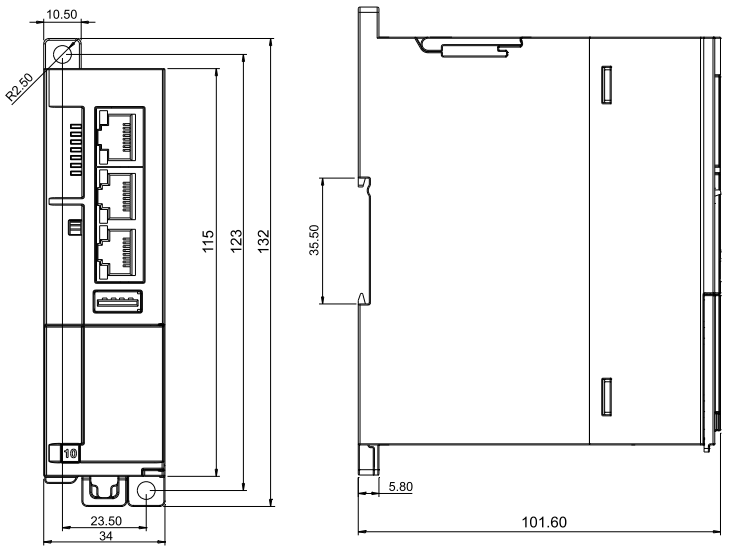
在更换或安装 CTH300-H 系列 PLC 时，要确定使用了正确或等同的模块。在更换 CTH300-H 系列 PLC 时，除了要使用相同的模块外，还要确保安装的方向和位置是正确的。

**注意**

- 如果您安装了不正确的模块，CTH300-H 系列 PLC 的程序可能会产生错误的功能。
- 如果未能使用相同的模块按照相同的方向和顺序替换 CTH300-H 系列 PLC，有可能造成严重的人身伤害和设备损坏。

## 3.2 安装尺寸

CTH300-H 系列 PLC 都有安装孔，可以很方便地安装在背板上，各模块安装尺寸见下表。

| 型号 | 尺寸图（单位：mm） |
|---|---|
| H31-00<br>H32-01<br>H35-00<br>H36-00<br>H36-01<br>CAN-1M<br>PWR-02<br>INT-00 |  |

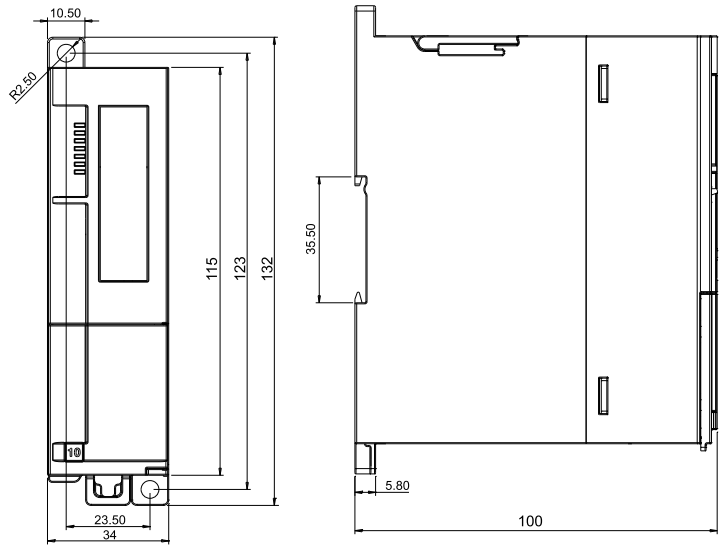
| 型号 | 尺寸图（单位：mm） |
| --- | --- |
| DIT-08<br>DIT-16<br>DIT-32<br>DQT-08<br>DQT-16<br>DQT-32<br>DQR-08<br>DQR-16<br>AIS-04<br>AIC-08<br>AIV-08<br>AQS-04<br>AQS-08<br>AMS-08<br>AIT-04<br>AIT-08<br>AIR-04<br>AIR-08<br>HSC-02<br>HSP-04 |  |

图 3-2 CTH300-H 系列 CPU 及模块尺寸图

## 3.3 使用机架

### ❑ 中央机架（Rack0）和扩展机架（Rack1\Rack2\Rack3）

CTH300-H 系统由一个中央单元和一个或多个扩展模块组成。

包含 CPU 的机架是中央单元。配有模块并连接到中央机架形成了系统的扩展机架。

### ❑ 扩展机架的使用

在您的应用中，如果中央机架的所有插槽均被使用，随后您即可使用扩展机架。

使用扩展机架时，除额外的机架和中继模块（INT）之外，可能还需要更多的电源模块。使用中继模块时，必须确保与扩展站相兼容。

### ❑ 机架上的模块布局

CTH300 的机架是一个装配导轨。可利用此导轨安装 CTH300-H 系统所属的模块。

**① 一个机架上的模块布局**

如果要在一个机架上安装模块，请注意以下几点：

- CPU 右侧安装的模块数不超过八个（SM、FM、CP）。
- 机架上的模块在 CTH300-H 背板总线上的累计功耗不得超过 1.6A。

下图显示在一个 CTH300-H 系统中装配八个信号模块的布局。

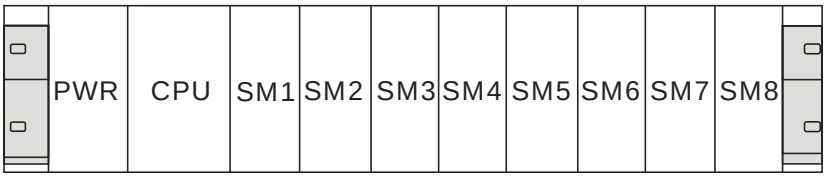



图 3-3 CTH300-H 系列布局图

**② 四个机架的完整装配**

如果计划在多个机架上执行装配，需要使用中继模块（INT），不同机架之间通过中继模块网口连接。

如果要在多个机架上排列模块，请注意以下几点：

- INT 模块始终使用插槽 3（插槽 1：PWR；插槽 2：CPU，插槽 3：INT）
- 在插入第一个信号模块前它始终位于左侧。
- 每个机架上安装的模块数不超过八个（SM、FM、CP）。
- 模块（SM、FM、CP）数量受到 CTH300-H 总线上允许的电流消耗的限制。每个机架的累积功耗不得超过 1600 mA。

下图显示 CTH300-H 系统中的各模块在 4 个机架上的排列情况。

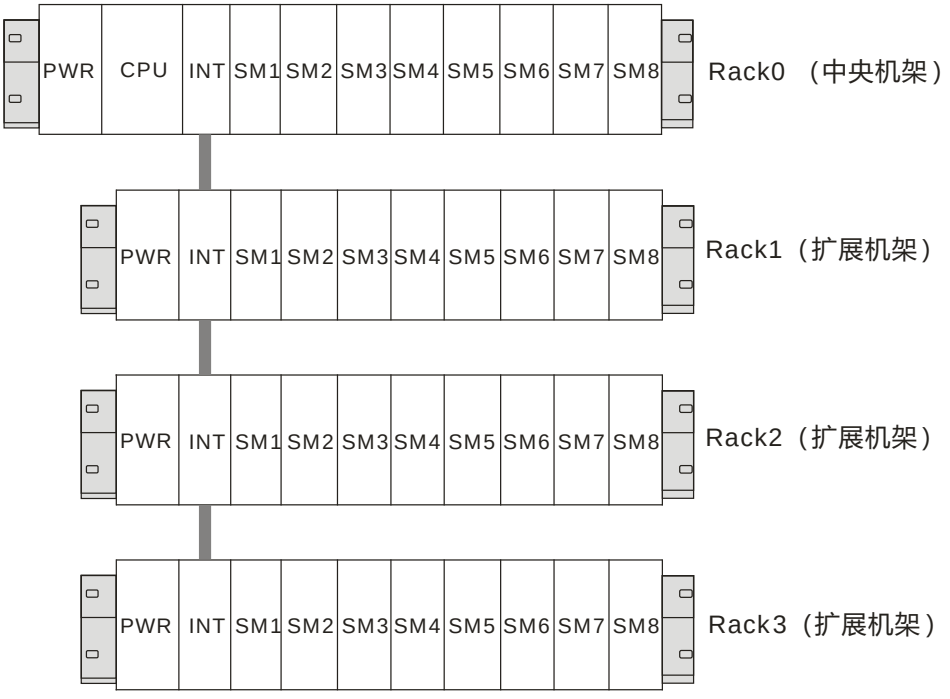



图 3-4 4 个机架的 CTH300-H 系列模块布局

**提示**

多机架系统使用中继模块时，仅第一个机架的中继模块需要通过背板总线挂接在 CPU 上，其余中继间通过网口连接。中继模块间通过网口连接时，请注意 IN/OUT 口的使用顺序，即前一个中继模块的 OUT 口连接下一个中继模块的 IN 口。

## 3.4 安装方法

### ❑ 安装方式

CTH300-H 系列 PLC 既可以安装在控制柜背板上，也可以安装在标准 DIN 导轨上；既可以水平安装，也可以垂直安装。

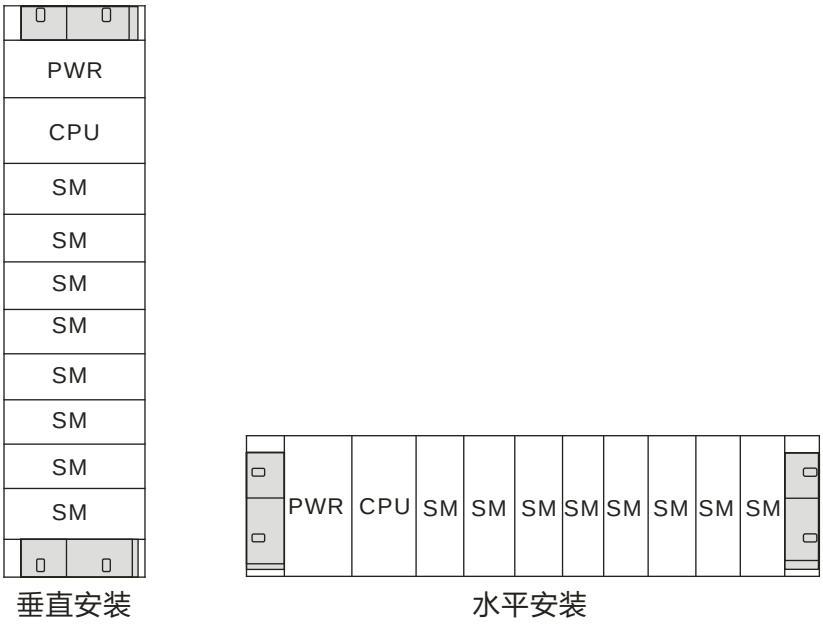



图 3-5 CPU 及模块安装方法

### ❑ 安装与拆除操作

请按照以下方法安装或拆除 CTH300-H 系列 PLC。

- **安装面板**

1）按照尺寸要求定位打孔；

2）用合适的螺钉将模块固定在背板上；

3）如果使用了扩展模块，将扩展模块的扁平电缆连到前盖下面的扩展口。

- **DIN 导轨安装**

1）将导轨固定在背板上；

2）打开模块底部的 DIN 夹子，将模块背部卡在 DIN 导轨上；

3）如果使用了扩展模块，将扩展模块的扁平电缆连到前盖下面的扩展口；

4）旋转模块贴近 DIN 导轨，合上 DIN 夹子；

5）仔细检查模块上 DIN 夹子与 DIN 导轨是否紧密固定好；

6）为避免模块损坏，不要直接按压模块正面，而要按压安装孔的部分。

**注意**

当 CTH300-H 系列 PLC 在震动比较大的使用环境或者采用垂直安装方式时，应该使用 DIN 导轨挡块。如果系统处于高震动环境中，使用背板安装方式可以得到较高的震动保护等级。

- **拆卸 CPU 或者扩展模块**

1）拆除 CTH300-H 系列 PLC 的电源；

2）拆除模块上的所有连线和电缆；

3）如果有其它扩展模块连接在您所拆卸的模块上，请打开前盖，拔掉相邻模块的扩展扁平电缆；

4）拆掉安装螺钉或者打开 DIN 夹子；

5）拆下模块，拆卸和安装端子排。

- **端子排的安装**

1）打开端子排安装位置的上盖；

2）确保模块上的插针与端子排边缘的小孔对正；

3）将端子排向下压入模块。确保端子块对准了位置并锁住。

- **CTH300-H CPU 端子排的拆卸**

1）打开端子排安装位置的上盖；

2）把螺丝刀插入端子排与端子座的的缝隙中；

3）螺丝刀上压并撬出端子排。

- **CTH300-H PLC 端子排的拆卸**

1）打开端子排安装位置的上盖；

2）用螺丝刀将两边的安装螺丝拧松就可以把端子排拆卸下来。

## 3.5 接地和布线

### ❑ CTH300-H 系列 PLC 接地和接线指南

合理的接地和接线对于所有的电器设备是至关重要的，它能够确保您的系统具备最优的操作特性，同时能够为您的系统提供更好的电子噪声保护。

CTH300-H 系列 PLC 及其相关设备的接线遵从所有有效电气编码规则。安装和操作所有设备要符合所有有效国家或地区标准。同地区的权威代表保持联系，以确定符合特殊需要的标准。

在设计 CTH300-H 系统的接地和接线时必须考虑安全因素，否则有可能造成设备的误动作。因此，您应该执行所有的安全规定以避免人员伤害和设备损坏。

**警告**

1、试图在带电情况下进行接地或接线，有可能造成死亡或严重的人身伤害和设备损坏。

2、控制设备有可能造成它所控制设备的误操作，进而导致死亡或者严重的人身伤害和设备损坏。因此 CTH300-H 系统中必须具备独立于 CTH300-H 系列 PLC 的急停功能、机电互锁或者其它冗余的安全设施。

## 3.6 抑制电路

在使用感性负载时，要加入抑制电路来限制输出关断时电压的升高。抑制电路可以保护输出点不至于因为高感抗开关电流而过早的损坏。另外，抑制电路还可以限制感性负载开关时产生的电子噪声。

**注意**

抑制电路的有效性取决于应用，您应该调整其参数以适应您的特殊应用。要确保所有器件参数与实际应用相符合。

### ❑ 晶体管输出和控制直流负载的继电器输出

晶体管输出有内部保护，可以适应多种应用。由于继电器型输出既可以连接直流负载，又可以连接交流负载，因而没有内部保护。

图 3-6 给出了直流负载抑制电路的一个实例。在大多数的应用中，用附加的二极管 A 即可，但如果您的应用中要求更快的关断速度，则推荐您加上齐纳二极管 B。确保齐纳二极管能够满足输出电路的电流要求。

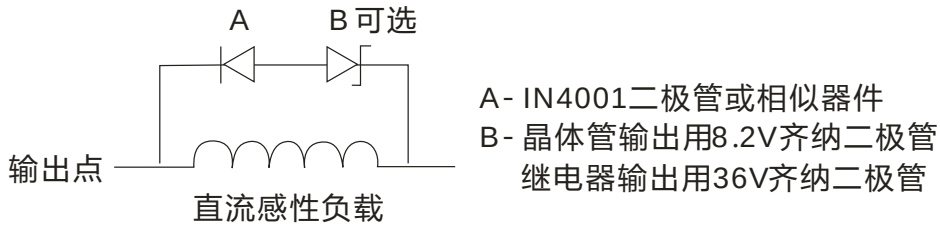



图 3-6 直流负载的抑制电路

### ❑ 交流输出和控制交流负载的继电器输出

交流输出对大部分应用都有内部保护，因为继电器可用于 DC 或 AC 负载所以不提供内部保护。

图 3-7 给出了交流负载抑制电路的一个实例。在大多数的应用中，附加的金属氧化物可变电阻(MOV)可以限制峰值电压，从而保护 CTH300-H 系列 PLC 内部电路。要确保 MOV 的工作电压比正常的线电压至少高出 20%。

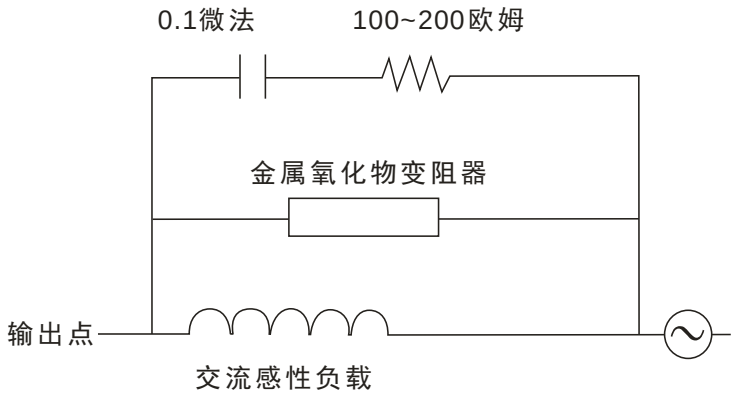



图 3-7 交流负载的抑制电路

# 4 电源模块

PWR-02 电源模块，最佳匹配于 CTH300 系统，专门为 CPU 及扩展模块（除数字量模块外）提供 24V DC 电源。每个机架请选配一个电源模块，数字量输入输出电源和传感器电源请选择其他供电电源。

### 本章概要

本章主要介绍 CTH300-H 系列 PWR-02 电源模块的如下内容：

- ❑ 技术规范
- ❑ 接线规格

表 4-1 电源模块 PWR-02 的基本属性

| 名称 | 规格描述 | 订货号 |
|---|---|---|
| PWR-02电源模块 | 输入：85~264V AC 输出：24V DC/2A | CTH3 PWR-020S1 |

## 4.1 技术规范

### 4.1.1 常规特性

表 4-2 电源模块 PWR-02 的常规特性

| | |
|---|---|
| **物理特性** | |
| 尺寸（W×H×D） | 34×115×101.6 mm |
| **LED 指示灯特性** | |
| 24V 电源指示灯（绿色） | 亮起：有 24V DC 输出，熄灭：无 24V DC 输出 |
| **开关特性** | |
| 开关 | 控制 24V DC 电源输出<br>ON：有 24V DC 输出，OFF：无 24V DC 输出 |

### 4.1.2 功能特性

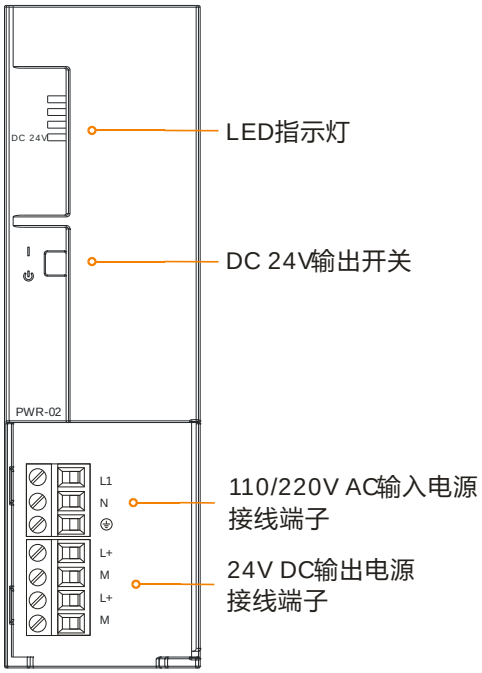
表 4-3 电源模块 PWR-02 的功能特性

| | |
|---|---|
| **输入电压特性** | |
| 电压范围 | 85~264VAC，宽电压输入 |
| 额定频率<br>频率范围 | 50Hz/60Hz<br>47Hz~63Hz |
| 效率 | 75% |

| | |
|---|---|
| 交流电流 | 0.9A/110V、0.5A/220V |
| 浪涌电流（25℃ 最大） | ≤20A/110V、≤35A/220V |
| 泄露电流 | ≤5mA/220VAC |
| **输出电压特性** | |
| 直流电压/额定电流 | 24VDC/2A |
| 额定功率 | 48W |
| 纹波和噪声（最大） | 150mVp-p |
| 电压输出范围 | ±5% |
| 启动/上升/保持时间 | ≤2.5s/≤50ms/≥20ms |
| 隔离（电源输入与输出） | 110V/220V AC 与 24V DC 之间隔离 |
| **保护功能** | |
| 过载保护 | 105%~130%的额定输出功率，切断输出，故障排除自动恢复 |
| 过压保护 | 115%~135%Ue；保护方式：打嗝模式，故障排除自动恢复 |
| 浪涌保护 | 供电电源端提供浪涌吸收功能 |
| 过流保护 | 电源输出端提供过流保护 |
| **安全电磁兼容** | |
| 耐电压 | 输入~输出：1.5KVDC，输入-PE：1.5KVDC，输出-PE：500VDC |
| 隔离电阻 | 输入~输出，输入-PE，输出-PE：100MΩ/500VDC |
| 依据标准 | 安全参照UL60950和UL1950，电磁兼容参照EN55022 |

## 4.2 接线规格

### 4.2.1 接口示意图



### 4.2.2 接口定义

表 4-4 PWR-02 的 220V AC 输入电源接口定义

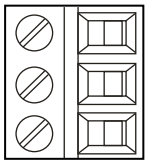
| 3 位可拆卸端子 | 信号 | 信号定义 |
|---|---|---|
|  | L | 火线 |
| | N | 零线 |
| | ⏚ | 接地 |

表 4-5 PWR-02 的 24V DC 输出电源接口定义

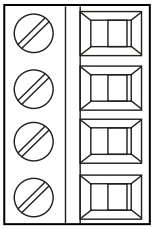
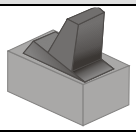
| 4 位可拆卸端子 | 信号 | 信号定义 |
|---|---|---|
|  | L+ | 24V 电源正 |
| | M | 24V 电源负 |
| | L+ | 24V 电源正 |
| | M | 24V 电源负 |

表 4-6 PWR-02 的拨码开关定义

| 两态开关 | 位号 | 拨码方向 | 信号定义 |
|---|---|---|---|
|  | ON | 向上 | 有 24V DC 输出 |
| | OFF | 向下 | 无 24V DC 输出 |

# 5 主控模块

主控模块是 CTH300-H 系统的中央处理单元，用于处理各种运算以及用户程序的运行，作为 CTH300-H 系统的核心控制装置。

### 本章概要

本章主要介绍 CTH300-H 系列主控模块的如下内容：

- 技术规范
- 接线规格

表 5-1 主控模块的基本特性

| 名称 | 规格描述 | 订货号 |
|---|---|---|
| H35-00 主控模块 | 192KB 程序空间，512KB 数据空间，16KB 掉电保持数据空间；双 PPI/自由通信接口，1 个以太网通信口，1 个 USB 接口 | CTH3 H35-000S1 |
| H36-00 主控模块 | 256KB 程序空间，1MB 数据空间，32KB 掉电保持数据空间；1 个 PPI/自由通信接口，1 个以太网通信口，1 个 EtherCAT 通信口，1 个 USB 接口，基本运动控制功能 | CTH3 H36-000S1 |
| H36-01 主控模块 | 256KB+64KB 程序空间，1MB 数据空间，32KB 掉电保持数据空间；1 个 PPI/自由通信接口，1 个以太网通信口，1 个 EtherCAT 通信口，1 个 USB 接口，1 个 CAN 通信接口，支持单轴运动控制，插补、电子齿轮和电子凸轮功能 | CTH3 H36-001S2 |
| H31-00 高性能薄型 | 192KB+64KB 程序空间，512KB 数据空间，16KB 掉电保持数据空间，双 PPI/自由通讯口，1 个以太网通信口，1 个 USB 接口，支持基本运动控制功能，最大机架数 1 个 | CTH3 H31-000S1 |
| H32-01 运动控制薄型 | 256KB+64KB 程序空间，1MB 数据空间，32KB 掉电保持数据空间，1 个 PPI/自由通讯口，1 个以太网通信口，1 个 USB 接口，1 个 CAN 接口，1 个 EtherCAT 接口，支持单轴运动控制，插补、电子齿轮及电子凸轮功能，最大机架数 1 个 | CTH3 H32-001S2 |

## 5.1 技术规范

### 5.1.1 常规特性

表 5-2 常规特性

| 物理特性 | |
|---|---|
| 尺寸（W×H×D） | 34×115×101.6 mm |

| | | |
|---|---|---|
| 功耗 | | 19.5W |
| **电源特性** | | |
| 额定输入电压 | | 24V DC |
| 输入电压范围 | | 20.4~28.8V DC |
| 输入电流 | | 0.8A |
| 极性反接保护 | | 有 |
| 总线电源电压 | | +5V DC |
| 总线电源电流 | | 1.6A |
| **LED 指示灯特性** | | |
| 24V 电源指示灯（绿色） | | 亮：24VDC 供电正常；灭：无 24VDC 供电 |
| SF 指示灯（红色） | | 亮：系统故障；灭：无错 |
| BF 指示灯（红色） | | 亮：总线故障；灭：无错 |
| FRCE 指示灯（黄色） | | 亮：有项目被强制；灭：无项目被强制<br>**<备注>** 也可通过 DLED 指令控制。 |
| RMC 指示灯（绿色） | | 亮:CPU 与远程服务器通讯成功(拨码开关 3 拨为 ON 且 EtherNET 通信口参数已正确配置）<br>灭：CPU 与远程服务器通讯失败或禁止与远程服务器通讯（允许和禁止通过拨码开关控制与服务器通讯） |
| RUN 指示灯（绿色） | | 亮：系统运行；灭：系统停止 |
| STOP 指示灯（橙色） | | 亮：系统停止；灭：系统运行 |
| 以太网指示灯 | 绿色 | 闪烁：连接；灭：未连接 |
| | 黄色 | 亮：100Mbps；灭：10Mbps |
| CAN 口指示灯（仅 H36-01 支持） | 绿色 | 数据接收指示灯 |
| | 黄色 | 数据发送指示灯 |
| EtherCAT 指示灯（仅 H32-01 支持） | 绿色 | 亮：连接；灭：未连接 |
| | 黄色 | 亮：100Mbps；灭：10Mbps |
| 运行开关 RUN/STOP | | 在 3 秒内快速拨动开关 8 次即可复位 IP 配置<br>（从 RUN->STOP 记为一次，从 STOP->RUN 也记为一次） |

### 5.1.2 功能特性

表 5-3 功能特性

| 功能项 | 描述 | | | | |
|---|---|---|---|---|---|
| | **H35-00** | **H31-00** | **H36-00** | **H36-01** | **H32-01** |
| **通信接口** | | | | | |
| 总线接口 | 提供扩展模块接口 | | | | |
| USB 接口 | 提供 USB 主设备接口 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 标配以太网接口 | 提供 1 个以太网接口<br>● 支持以太网编程；<br>● 支持 UDP_PPI 协议：CPU 之间的网络读写操作（指令方式 UDPNETW/UDPNETR 或指令向导方式）；<br>● 支持 Modbus_TCP 协议；<br>● 支持 S7 主从站协议（仅 H32-01、H36-01 支持）。 | | | | |
| RS485 接口 | 2 个 | | 1 个 | | |
| EtherCAT 接口 | 无 | | 1 个 | | |
| CANopen 接口 | 无 | | | 1 个 | |
| **存储功能** | | | | | |
| 程序存储空间 | 192KB | 256KB | | 256KB+64KB | |
| 数据存储空间 | 512KB | | 1MB | | |
| 掉电保持数据 | 16KB | | 32KB | | |
| **扩展 I/O 功能** | | | | | |
| 最多支持机架数 | 4 | 1 | 4 | 4 | 1 |
| 每机架支持最多模块数 | 除了电源、CPU、中继模块外最多支持 8 个模块 | | | | |
| 通过 INT-00 扩展模块 | H35-00/H36-00/H36-01：最大允许扩展 32 个 I/O 模块（4 机架，每机架 8 个模块）<br>H31-00/H32-01：只有 1 个机架，最大允许扩展 8 个 I/O 模块 | | | | |
| 通过 ECT-00 扩展模块 | H36-00/H36-01：最大允许 512 个 I/O 模块（64 个 ECT，每个 ECT 可扩展 8 个模块）<br>H32-01：最大允许 64 个 I/O 模块（8 个 ECT，每个 ECT 可扩展 8 个模块） | | | | |
| **隔离和保护** | | | | | |
| 电源隔离 | 外部电源与系统电源之间隔离 | | | | |
| 通讯隔离 | 以太网、EtherCAT、CAN、RS485 隔离 | | | | |
| 电源保护 | 供电电源端提供反接保护功能及浪涌吸收功能 | | | | |
| 接口保护 | 通讯端口防雷保护 | | | | |
| **编程功能** | | | | | |
| MagicWorks PLC 提供梯形图 LAD、指令列表 STL 两种编程语言，编程软件支持中文/英文。 | | | | | |
| **运控功能** | | | | | |
| H36-01，H32-01：支持单轴运动控制，插补、电子凸轮和电子齿轮功能<br>H36-00，H31-00：支持基本运动控制功能 | | | | | |

### 5.1.3 通信口规范

表 5-4 通讯口规范

| **RS485通讯口** | |
|---|---|
| 通讯接口数及协议 | RS485通讯口（可切换 PPI 与自由口协议） |
| 通讯波特率 | PPI/MPI：9.6Kbps、19.2Kbps、187.5Kbps |
| | 自由口：1.2Kbps~115.2Kbps，内置 Modbus 主站/从站功能 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 每段最大电缆长度 | 使用隔离中继器：1000m（187.5Kbps）/1200m（38.4Kbps）<br>未使用隔离中继器：50m | | | | | | |
| 最大站点数 | 每段32个站，每个网络126个站 | | | | | | |
| 点到点<br>（PPI 主站模式） | 是（NETR/NETW），每条指令最多读/写200字节，同一时刻允许有8条读写指令 | | | | | | |
| MPI 连接 | 一个 MPI 从站最多可连接8个 MPI 主站 | | | | | | |
| 隔离 | 通信口隔离 | | | | | | |
| **以太网通讯口** | | | | | | | |
| 通讯接口 | 1个标配以太网口 | | | | | | |
| 波特率 | 10/100Mbps 自适应 | | | | | | |
| 协议类型 | UDP_PPI 协议、Modbus_TCP/IP 主从站协议，S7主从站协议（仅H32-01、H36-01支持）。 | | | | | | |
| 每段最大电缆长度 | 100m | | | | | | |
| 一个站点最大连接数 | UDP_PPI 最多支持8个连接，Modbus_TCP 最多支持32个连接 | | | | | | |
| DHCP 功能 | 支持 | | | | | | |
| 远程监控/编程功能 | 支持 | | | | | | |
| 隔离 | 通信口隔离 | | | | | | |
| **EtherCAT 通讯口（H36-00/H36-01/H32-01支持）** | | | | | | | |
| 通讯接口 | 1个 EtherCAT 通信主站接口 | | | | | | |
| 波特率 | 100Mbps | | | | | | |
| 协议类型 | EtherCAT 接口协议 | | | | | | |
| 每段最大电缆长度 | 100m | | | | | | |
| 支持的功能 | 配置分布时钟、配置启动参数、配置 PDO 参数和映射、配置总线循环周期、配置启动检查供应商 ID 和产品 ID | | | | | | |
| 隔离 | 通信口隔离 | | | | | | |
| **CANopen 通讯口（H36-01/H32-01支持）** | | | | | | | |
| 通讯接口 | 1个 CAN 通信主站接口 | | | | | | |
| 最大从站点数 | 1个主站后面最多可连接32个从站 | | | | | | |
| 协议类型 | CANopen DS301标准协议 | | | | | | |
| 支持功能 | 自动启动 CANopen manager、可选从站轮询、启动从站、NMT、同步生产、同步消耗、心跳产生、创建激活时间 | | | | | | |
| 传输速率 | 1000 | 800 | 500 | 250 | 125 | 50 | 20 |
| 最大长度 | 25 | 50 | 100 | 250 | 500 | 1000 | 2500 |
| 隔离 | 通信口隔离 | | | | | | |

### 5.1.4 数据存储器规格

表 5-5 数据存储区规格

| **数字量输入/输出映象区（I/Q）** | | |
|---|---|---|
| | **数字输入映象区（I）** | **数字输出映象区（Q）** |
| 位地址范围 | I0.0~I255.7 | Q0.0~Q255.7 |

| | | |
|---|---|---|
| 字节地址范围 | IB0~IB255 | QB0~QB255 |
| 字地址范围 | IW0~IW254 | QW0~QW254 |
| 双字地址范围 | ID0~ID252 | QD0~QD252 |
| 访问属性 | 支持立即访问/直接访问/间接访问 | |
| 保持属性 | 不支持掉电保持 | |
| **模拟输入/输出映象区（AI/AQ）** | | |
| 每通道字长 | 16 bits | |
| 参数地址范围 | 模拟输入映象区（AI）：<br>AIW0~AIW1022（只读） | 模拟输出映象区（AQ）：<br>AQW0~AQW1022（只写） |
| 访问属性 | 支持立即访问/直接访问/间接访问 | |
| 保持属性 | 不支持掉电保持 | |
| **变量内存区（V）** | | |
| 容量（字节） | 65536 bytes | |
| 位地址范围 | V0.0~V65535.7 | 读/写 |
| 字节地址范围 | VB0~VB65535 | 读/写 |
| 字地址范围 | VW0~VW65534 | 读/写 |
| 双字地址范围 | VD0~VD65532 | 读/写 |
| 访问属性 | 支持直接访问/间接访问 | |
| 保持属性 | V0.0~V10239.7支持掉电保持，其他内存区不支持掉电保持 | |
| **特殊内存区（SM）** | | |
| 容量 | 2048 bytes | |
| 位地址范围 | SM0.0~SM2047.7 | SMB0-29只读、之后读/写 |
| 字节地址范围 | SMB0~SMB2047 | SMB0-29只读、之后读/写 |
| 字地址范围 | SMW0~SMW2046 | SMB0-29只读、之后读/写 |
| 双字地址范围 | SMD0~SMD2044 | SMB0-29只读、之后读/写 |
| 访问属性 | 支持直接访问/间接访问 | |
| **特殊寄存器 SM 功能说明参见本文档附录 H 特殊存储器说明** | | |
| **内部内存区（M）** | | |
| 容量（字节） | 32 bytes | |
| 位地址范围 | M0.0~M31.7 | |
| 字节地址范围 | MB0~MB31 | |
| 字地址范围 | MW0~MW30 | |
| 双字地址范围 | MD0~MD28 | |
| 访问属性 | 支持直接访问/间接访问 | |
| 保持属性 | 可配置为全部或部分掉电保持 | |
| **局部变量区（L）** | | |
| 容量（字节） | 128 bytes | |
| 位地址范围 | L0.0~L127.7 | |
| 字节地址范围 | LB0~LB127 | |
| 字地址范围 | LW0~LW126 | |
| 双字地址范围 | LD0~LD124 | |

| | | | | |
|---|---|---|---|---|
| 访问属性 | 支持直接访问 | | | |
| 保持属性 | 开机到停机对同一调用位置子程序保持，不能掉电保持 | | | |
| **累加器寄存器（AC）** | | | | |
| 容量 | 4个 | | | |
| 位地址范围 | 不支持 | | | |
| 字节地址范围 | AC0~AC3 | | | |
| 字地址范围 | AC0~AC3 | | | |
| 双字地址范围 | AC0~AC3 | | | |
| 访问属性 | 支持直接访问 | | | |
| 保持属性 | 不支持掉电保持 | | | |
| **S** | | | | |
| 容量（字节） | 32 | | | |
| 位地址范围 | S0.0~S31.7 | | | |
| 字节地址范围 | SB0~S31 | | | |
| 字地址范围 | SW0~SW30 | | | |
| 双字地址范围 | SD0~SD28 | | | |
| 访问属性 | 支持直接/间接访问 | | | |
| 保持属性 | 不支持掉电保持 | | | |
| **定时器（T）** | | | | |
| **类型** | **分辨率** | **数目** | **编号** | **最大定时值** |
| 有记忆接通延时定时器 TONR | 1ms | 2 | T0，T64 | 32.767s |
| | 10ms | 8 | T1~T4，T65~T68 | 327.67s |
| | 100ms | 54 | T5~T31，T69~T95 | 3276.7s |
| 接通延时定时器 TON/关断延时定时器 TOF | 1ms | 34 | T32，T96，T256~T287 | 32.767s |
| | 10ms | 744 | T33~T36，T97~T100，T288~T1023 | 327.67s |
| | 100ms | 1206 | T37~T63，T101~T255，T1024~T2047 | 3276.7s |
| 访问属性 | 计时寄存器可直接/间接访问，状态位仅可直接访问 | | | |
| 保持属性 | 可配置当前计时值的掉电保持属性，状态位不能保持 | | | |
| **计数器（C）** | | | | |
| 数目 | 2048 | | | |
| 计数方式 | 向上计数/向下计数/向上向下计数 | | | |
| 最大计数值 | 32767 | | | |
| 访问属性 | 计数寄存器可直接/间接访问，状态位仅可直接访问 | | | |
| 保持属性 | 可配置当前计数值的掉电保持属性，状态位不能保持 | | | |
| **上升沿和下降沿** | | | | |
| 最大可用数量总和 | 4096个 | | | |

表 5-6　300CPU 支持的数据类型

| 数据类型 | 大小 | 说明 | 取值范围 |
|---|---|---|---|
| 布尔 | 1位 | 布尔值 | 0~1 |

| | | | |
|---|---|---|---|
| 字节 | 8位 | 无符号字节 | 0~255 |
| 字 | 16位 | 无符号整数 | 0~65535 |
| 整数 | 16位 | 有符号整数 | -32768~+32767 |
| 双字 | 32位 | 无符号双整数 | 0~4294967295 |
| 双整数 | 32位 | 有符号双整数 | -2147483648~+2147483647 |
| 实数 | 32位 | IEEE 32位浮点数 | +1.175495E-38~+3.402823E+38<br>-1.175495E-38~-3.402823E+38 |
| 字符串 | 1~255字节 | ASCII 字符串照原样存储在 PLC 内存中，形式为：1字节字符串长度+ASCII 字符 | 无 |

### 5.1.5 密码级别及权限控制

表 5-7 300CPU 密码级别及权限控制

| 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/硬件配置块 | 允许 | 允许 | 需验证密码 | 不允许 |
| 下载程序块/数据块/硬件配置块 | 允许 | 需验证密码 | 需验证密码 | 需验证密码（不允许下载硬件配置块） |
| 清除程序块/数据块/硬件配置块 | 允许 | 需验证密码 | 需验证密码 | 需验证密码（不允许删除硬件配置块，允许三块一起删除） |
| 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 |

### 5.1.6 实时时钟功能

表 5-8 300CPU 的实时时钟功能

| | |
|---|---|
| 出厂设置 | 未设置，固定值90年1月1日00:00:00星期日 |
| 掉电保持时间 | 约112小时（典型值） |

| | | |
|---|---|---|
| 精度 | 每月偏差<60秒 | |
| 读取时钟功能 | 通过指令 TODR/TODRX 或通过软件读取 | |
| 设置时钟功能 | 通过指令 TODW/TODWX 或通过软件设置 | |
| **常规时钟格式（8字节）** | | |
| **T 字节** | **说明** | **字节数据** |
| 0 | 年（0-99） | 当前年份（BCD 值） |
| 1 | 月（1-12） | 当前月份（BCD 值） |
| 2 | 日期（1-31） | 当前日期（BCD 值） |
| 3 | 小时（0-23） | 当前小时（BCD 值） |
| 4 | 分钟（0-59） | 当前分钟（BCD 值） |
| 5 | 秒（0-59） | 当前秒（BCD 值） |
| 6 | 0 | 保留，始终设置为00 |
| 7 | 星期几（1-7） | 当前是星期几，1=星期日（BCD 值） |
| **扩展时钟格式（19字节）** | | |
| **T 字节** | **说明** | **字节数据** |
| 0 | 年（0-99） | 当前年份（BCD 值） |
| 1 | 月（1-12） | 当前月份（BCD 值） |
| 2 | 日期（1-31） | 当前日期（BCD 值） |
| 3 | 小时（0-23） | 当前小时（BCD 值） |
| 4 | 分钟（0-59） | 当前分钟（BCD 值） |
| 5 | 秒（0-59） | 当前秒（BCD 值） |
| 6 | 0 | 保留，始终设置为00 |
| 7 | 星期几（1-7） | 当前是星期几，1=星期日（BCD 值） |
| 8 | 时区 | 00H-03H, 08H,10H-13H, FFH |
| 9 | 修正小时数（0-23） | 修正数量，小时（BCD 值） |
| 10 | 修正分钟数（0-59） | 修正数量，分钟（BCD 值） |
| 11 | 开始月份（1-12） | 夏时制的开始月份（BCD 值） |
| 12 | 开始日期（1-31） | 夏时制的开始日期（BCD 值） |
| 13 | 开始小时（0-23） | 夏时制的开始小时（BCD 值） |
| 14 | 开始分钟（0-59） | 夏时制的开始分钟（BCD 值） |
| 15 | 结束月份（1-12） | 夏时制的结束月份（BCD 值） |
| 16 | 结束日期（1-31） | 夏时制的结束日期（BCD 值） |
| 17 | 结束小时（0-23） | 夏时制的结束小时（BCD 值） |
| 18 | 结束分钟（0-59） | 夏时制的结束分钟（BCD 值） |

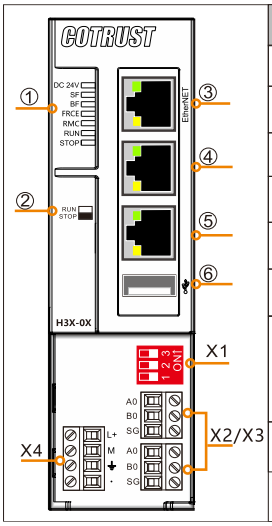
### 5.1.7 中断事件

表 5-9 支持的中断事件

| 优先级别群组 | 中断事件号 | 优先级别组别 | 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） | 12 | 1 | HSC0 CV=PV（须与运控模块配合才能实现） |
| | 27 | 1 | HSC0方向改变（须与运控模块配合才能实现） |
| | 28 | 1 | HSC0外部复原 / Zphase（须与运控模块配合才能实现） |
| | 13 | 1 | HSC1 CV=PV（须与运控模块配合才能实现） |
| | 14 | 1 | HSC1方向改变（须与运控模块配合才能实现） |
| | 15 | 1 | HSC1外部复原（须与运控模块配合才能实现） |
| | 16 | 1 | HSC2 CV=PV（须与运控模块配合才能实现） |
| | 17 | 1 | HSC2方向改变（须与运控模块配合才能实现） |
| | 18 | 1 | HSC2外部复原（须与运控模块配合才能实现） |
| | 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（须与运控模块配合才能实现） |
| | 29 | 1 | HSC4 CV=PV（须与运控模块配合才能实现） |
| | 30 | 1 | HSC4方向改变（须与运控模块配合才能实现） |
| | 31 | 1 | HSC4外部复原 / Z phase（须与运控模块配合才能实现） |
| | 33 | 1 | HSC5 CV=PV（须与运控模块配合才能实现） |
| 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 |

## 5.2 接线规格

### 5.2.1 接口示意图




| | H35-00 | H31-00 | H36-00 | H36-01 | H32-01 |
|---|---|---|---|---|---|
| ① | LED指示灯 | | | | |
| ② | 系统运行开关 | | | | |
| ③ | EtherNET 通信口 | | | | |
| ④ | RS485 通信口 | | EtherCAT 通信口 | EtherCAT 通信口 | |
| ⑤ | RS485 通信口 | | EtherCAT 通信口 | CAN 通信口 | |
| ⑥ | USB接口 | | | | |
| X1 | 3：远程控制开关<br>2：终端电阻开关<br>1：终端电阻开关 | 3：远程控制开关<br>1：终端电阻开关 | | | |
| X2/X3 | PORT0:RS485接线端子 | | | | |
| X4 | 电源接线端子 | | | | |

### 5.2.2 接口定义

表 5-10　电源接口定义

| 4 位可拆卸端子（X4） | 符号 | 信号定义 |
|---|---|---|
| L+<br>M<br>⏚<br>• | L+ | 24V 电源正 |
| | M | 24V 电源负 |
| | ⏚ | 24V 电源地 |
| | -- | -- |

表 5-11　USB 接口定义

| USB 接口 | 位号 | 信号 | 信号定义 |
|---|---|---|---|
| 1 2 3 4 | 1 | V_BUS | +5V 电源 |
| | 2 | Data- | 数据负 |
| | 3 | Data+ | 数据正 |
| | 4 | GND | 地 |

表 5-12　标配以太网接口/EtherCAT 接口定义

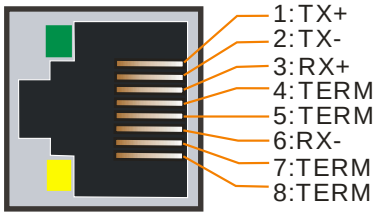
| 单网口接口（EtherNET） | 位号 | 信号 | 信号定义 |
|---|---|---|---|
|  | 1 | TX+ | 数据发送正端 |
| | 2 | TX- | 数据发送负端 |
| | 3 | RX+ | 数据接收正端 |
| | 4 | -- | -- |
| | 5 | -- | -- |
| | 6 | RX- | 数据接收负端 |
| | 7 | -- | -- |
| | 8 | -- | -- |
| | 连接器外壳 | PE | 机壳接地 |

表 5-13　H35-00/H31-00 的 RS485 通信接口定义

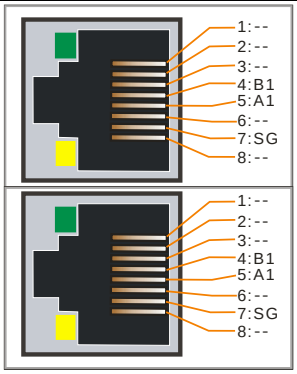
| 双网口接口（PORT1） | 位号 | 信号 | 信号定义 |
|---|---|---|---|
|  | 1 | -- | -- |
| | 2 | -- | -- |
| | 3 | -- | -- |
| | 4 | B1 | RS485+ 信号 B |
| | 5 | A1 | RS485- 信号 A |
| | 6 | -- | -- |
| | 7 | SG | RS485 信号地 |
| | 8 | -- | -- |

表 5-14　CAN 通信接口定义

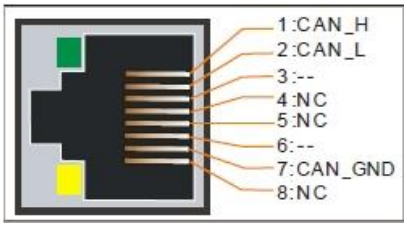
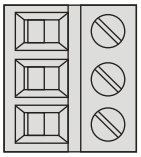
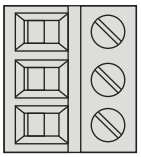
| RJ45 网口 | 位号 | 信号 | 信号定义 |
|---|---|---|---|
|  | 1 | CAN_H | 数据发送正端 |
| | 2 | CAN_L | 数据发送负端 |
| | 3 | -- | -- |
| | 4 | -- | -- |
| | 5 | -- | -- |
| | 6 | -- | -- |
| | 7 | CAN_GND | CAN 的地 |
| | 8 | -- | -- |
| | 连接器外壳 | PE | 机壳接地 |
| 绿色指示灯 | 作为 CAN 数据接收状态指示 | | |
| 黄色指示灯 | 作为 CAN 数据发送状态指示 | | |

表 5-15　RS485 接线端子定义

| 6 位可拆卸端子（PORT0） | | 信号 | 信号定义 |
|---|---|---|---|
| X2 |  | A0 | RS485- 信号 A/- |
| | | B0 | RS485+ 信号 B/+ |
| | | SG | RS485 信号地 |
| X3 |  | A0 | RS485- 信号 A/- |
| | | B0 | RS485+ 信号 B/+ |
| | | SG | RS485 信号地 |

**提示**

一些仪表的 RS485 信号定义为 A 正，B 负，接线的时候需要注意对应正负，正对正，负对负（**全局通用**）。

表 5-16　系统运行开关拨码定义

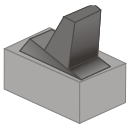
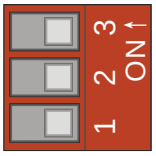
| 系统运行开关 | 符号 | 拨码方向 | 信号定义 |
|---|---|---|---|
|  | RUN | 向上 | 系统运行 |
| | STOP | 向下 | 系统停止 |
| | | 在 3 秒内快速拨动 8 次 | 复位 IP 配置（从 RUN->STOP 记为一次，从 STOP->RUN 也记为一次） |

表 5-17　DIP 开关拨码定义

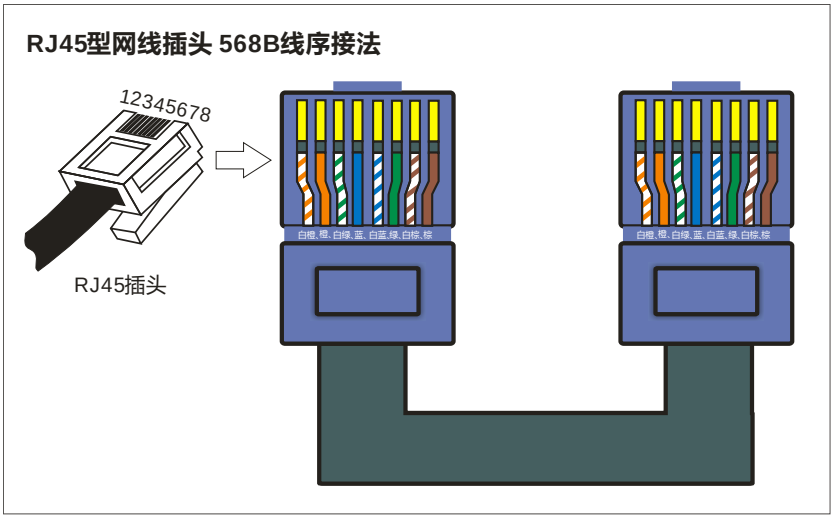
| DIP 开关 | 位号 | 符号 | 信号定义 | 备注 |
|---|---|---|---|---|
|  | 1 | Term-PORT0 | 终端电阻开关 1（PORT0） | ON：有终端电阻<br>OFF：无终端电阻 |
| | 2 | Term-PORT1 (H35-00) | 终端电阻开关 2（PORT1） | ON：有终端电阻<br>OFF：无终端电阻 |
| | | NC (H36-0X) | -- | -- |
| | 3 | RMC-SW | 远程控制开关 | ON：远程控制允许<br>OFF：远程控制禁止 |

## 5.2.3　制作标准网线

CTH300-H 系列可编程控制器进行以下通信时，推荐使用标准网线：

- 主控模块与上位机进行 EtherNET 通信
- 主控模块进行 EtherCAT 通信
- 主控模块进行 CANopen 通信
- CAN-1M 主站模块与 E10/H1A/H2A/EM277C 等 CAN 从站进行 CANopen 通信

参考如下示意图制作标准网线：


RJ45型网线插头 568B线序接法
12345678
RJ45插头
白橙,橙,白绿,蓝,白蓝,绿,白棕,棕
白橙,橙,白绿,蓝,白蓝,绿,白棕,棕

# 6 数字量模块

数字量模块是 CTH300-H 系统扩展模块中的成员，用于输出数字信号给现场设备或者接收现场设备输入的数字信号。

### 本章概要

本章主要介绍 CTH300-H 系列数字量模块的如下内容：

- ❑ 数字量输入模块技术规范和接线规格
- ❑ 数字量输出模块技术规范和接线规格

表 6-1 数字量模块的基本属性

| 名称 | 规格描述 | 订货号 |
|---|---|---|
| DIT-080S1 数字模块 | 数字量输入 8 x 24VDC | CTH3 DIT-080S1 |
| DIT-160S1 数字模块 | 数字量输入 16 x 24VDC | CTH3 DIT-160S1 |
| DIT-320S1 数字模块 | 数字量输入 32 x 24VDC | CTH3 DIT-320S1 |
| DQT-080S1 数字模块 | 数字量输出 8 x 24VDC | CTH3 DQT-080S1 |
| DQT-160S1 数字模块 | 数字量输出 16 x 24VDC | CTH3 DQT-160S1 |
| DQT-320S1 数字模块 | 数字量输出 32 x 24VDC | CTH3 DQT-320S1 |
| DQR-080S1 数字模块 | 数字量输出 8 x 继电器 | CTH3 DQR-080S1 |
| DQR-160S1 数字模块 | 数字量输出 16 x 继电器 | CTH3 DQR-160S1 |

## 6.1 数字量输入模块

### 6.1.1 技术规范

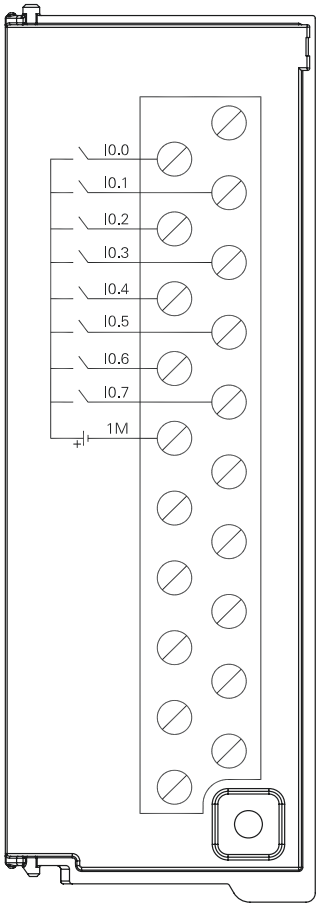
表 6-2 数字量输入模块规范

| 特性 | | DIT-080S1 | DIT-160S1 | DIT-320S1 |
|---|---|---|---|---|
| 尺寸（W×H×D） | | 34×115×100 mm | | |
| 输入点数 | | 8 | 16 | 32 |
| 电流消耗 | 24V DC | 4mA/通道 | | |
| | +5V 总线 | 60mA | 80mA | 130mA |
| 输入类型 | | 漏型/源型（IEC 1 类漏型） | | |
| 输入额定电压 | | 24V DC，6mA | | |
| 输入电压范围 | | 20.4~28.8V DC | | |
| 浪涌电压 | | 35V DC，持续 0.5s | | |
| 逻辑 1（最小） | | 15V DC，2.5mA，翻转电平：10.5V DC±15% | | |

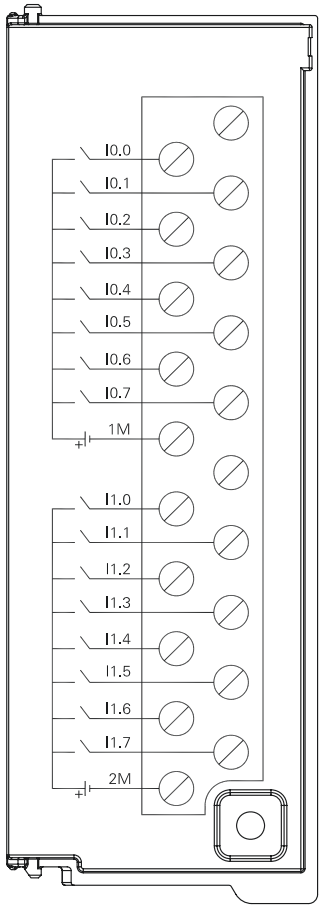
| | | | | |
|---|---|---|---|---|
| 逻辑 0（最大） | | 5V DC，1mA | | |
| 连接 2 线接近开关传感器（BERO）<br>允许的漏电流（最大） | | 1mA | | |
| 输入滤波 | | 可配置，支持 0.2ms、0.4ms、0.8ms、1.6ms、3.2ms、6.4ms（默认）、12.8ms | | |
| 输入频率 | | 1.5KHz，占空比 50% | | |
| 输入阻抗 | | 6.6KΩ | | |
| 隔离 | | 500V AC，持续 1min | | |
| 每组隔离点数 | | 8 | | |
| 同时 ON 点数 | | 8 | 16 | 32 |
| 电缆长度 | 屏蔽 | 500m | | |
| | 非屏蔽 | 300m | | |

## 6.1.2 接线规格

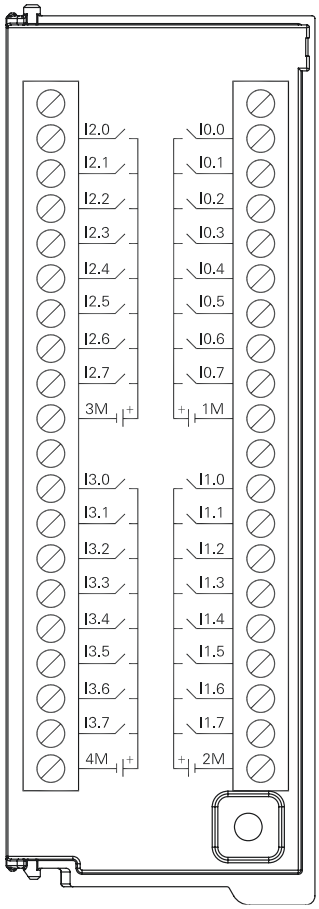
◆ DIT-080S1 接线图




◆ DIT-160S1 接线图




◆ DIT-320S1 接线图



## 6.2 数字量输出模块

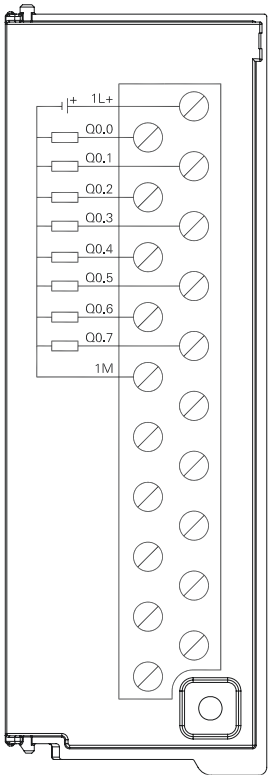
### 6.2.1 技术规范

表 6-3 数字量输出模块规范

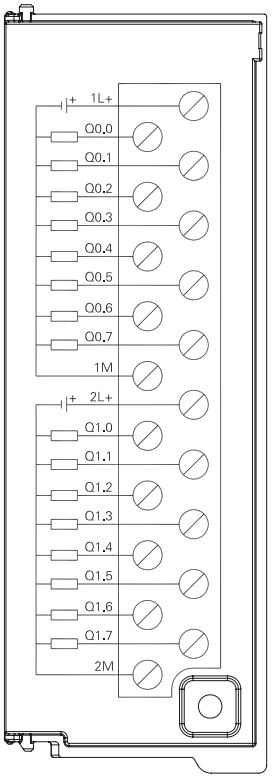
| 特性 | | DQT-080S1 | DQT-160S1 | DQT-320S1 | DQR-080S1 | DQR-160S1 |
|---|---|---|---|---|---|---|
| 尺寸（W×H×D） | | 34×115×100 mm | | | | |
| 输出点数 | | 8 | 16 | 32 | 8 | 16 |
| 电流消耗 | 24V DC | 50mA | 95mA | 180mA | 64mA | 130mA |
| | +5V 总线 | 70mA | 120mA | 210mA | 45mA | 60mA |
| 输出类型 | | 固态-MOSFET，源型 | | | 继电器-干触点 | |
| 输出额定电压 | | 24V DC | | | DC：24V，AC：110V/220V | |
| 输出电压范围 | | 20.4~28.8V DC | | | DC：5~30V，AC：5~250V | |
| 逻辑 1（最小） | | 20V DC | | | - | |
| 逻辑 0（最大） | | 0.1V DC，10KΩ 负载 | | | - | |
| 最大输出电流 | | 0.5A | | | 2A | |
| 每个公共端电流 | | 最大 4A | | | 最大 16A | |
| 最大输出漏电流 | | 15uA | | | - | |
| 浪涌电流 | | 8A，100ms | | | 5A，4s@10%占空比 | |
| 灯负载 | | 5W | | | DC：30W /AC：200W | |
| 接触电阻 | | 0.3Ω，最大 0.6Ω | | | 新设备最大为 0.2Ω | |
| 输出延迟 | | 断开到接通（最大）：50us<br>接通到断开（最大）：200us | | | 最大 10ms | |
| 最大输出频率 | | 1KHz | | | 阻性负载：10Hz<br>灯负载：1Hz<br>感性负载：0.5Hz | |
| 机械寿命（无负载） | | - | | | 10，000，000 | |
| 触点寿命（额定负载） | | - | | | 100，000 | |
| 隔离 | 现场到逻辑 | 500V AC，持续 1min | | | | |
| | 线圈到触点 | - | | | 1500V AC，持续 1min | |
| 每组隔离点数 | | 8 | | | | |
| 同时 ON 点数 | | 8 | 16 | 32 | 8 | 16 |
| 两个输出并联 | | 支持同组内两个输出并联 | | | | |
| 电缆长度 | 屏蔽 | 500m | | | | |
| | 非屏蔽 | 150m | | | | |

## 6.2.2 接线规格

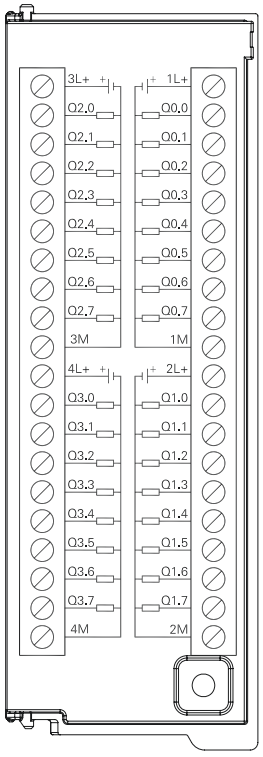
◆ DQT-080S1 接线图



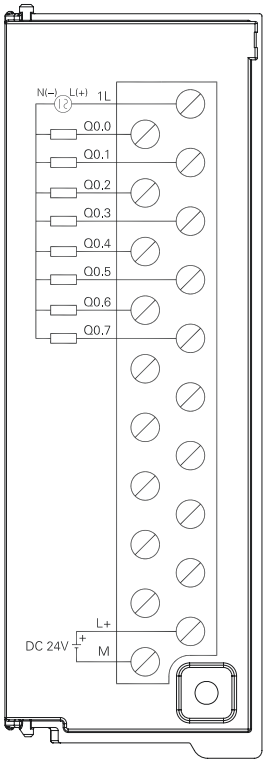

◆ DQT-160S1 接线图



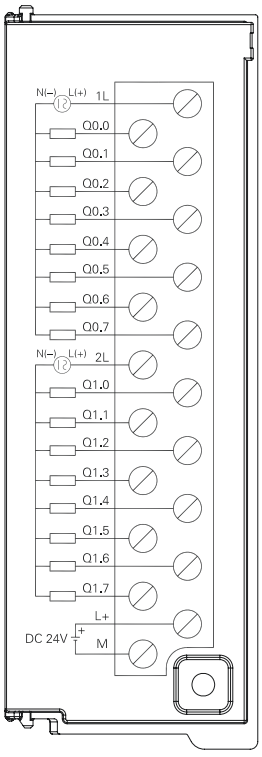

◆ DQT-320S1 接线图




◆ DQR-080S1 接线图




◆ DQR-160S1 接线图



# 7 模拟量模块

模拟量模块是 CTH300-H 系统扩展模块中的成员，用于输出模拟信号给现场设备或者接收现场设备输入的模拟信号。

### 本章概要

本章主要介绍 CTH300-H 系列模拟量模块的如下内容：

- ❑ 模拟量模块通用特性
- ❑ 模拟量输入模块技术规范和接线规格
- ❑ 模拟量输出模块技术规范和接线规格
- ❑ 模拟量输入/输出模块技术规范和接线规格

表 7-1 模拟量模块的基本属性

| 名称 | 规格描述 | 订货号 |
|---|---|---|
| AIS-04模拟模块 | 模拟量电压电流输入，4AI x 12 bits | CTH3 AIS-040S1 |
| AMS-06模拟模块 | 模拟量电压电流输入输出，4AIx12 bits，2AQx12 bits | CTH3 AMS-060S1 |
| AIV-08 模拟模块 | 模拟量电压输入，8AI x 16 bits | CTH3 AIV-080S1 |
| AIC-08 模拟模块 | 模拟量电流输入，8AI x 16 bits | CTH3 AIC-080S1 |
| AQS-04模拟模块 | 模拟量电压电流输出，4AQ x 12 bits | CTH3 AQS-040S1 |
| AQS-08模拟模块 | 模拟量电压电流输出，8AQ x 12 bits | CTH3 AQS-080S1 |

## 7.1 通用特性

### 7.1.1 常规特性

表 7-2 模拟量模块的常规特性

| 特性 | AIS-040S1 | AMS-060S1 | AIV-080S1/ AIC-080S1 | AQS-040S1 | AQS-080S1 |
|---|---|---|---|---|---|
| **物理特性** | | | | | |
| 尺寸（W×H×D） | 34×115×100 mm | | | | |
| **电源特性** | | | | | |
| 额定输入电压 | 24V DC | | | | |
| 输入电压范围 | 20.4V~28.8V DC | | | | |
| 输入电流 | 65mA | 110mA | 50mA | 110mA | 200mA |
| 极性反接保护 | 有 | | | | |
| 总线电源电压 | +5V DC | | | | |

| | | | |
|---|---|---|---|
| 总线电源电流 | 50mA | 30mA | 40mA |
| **LED 指示灯特性** | | | |
| 24V 电源指示灯 | 亮：24VDC 供电正常；灭：无 24VDC 供电 | | |
| SF 指示灯 | 亮：模块故障；灭：无错<br>闪烁：输入电流信号超限报警（仅针对 4-20mA） | | |

### 7.1.2 功能特性

表 7-3 模拟量模块功能特性

| 功能类别 | 描述 |
|---|---|
| 扩展功能 | 提供总线扩展功能 |
| 信号隔离 | 现场与总线之间隔离 |
| 电源保护 | 供电电源端提供反接保护功能及浪涌吸收功能 |
| 滤波功能 | 采用硬件滤波与软件滤波相结合的方式 |
| 电源功能 | 模块采用 24V DC 供电 |

## 7.2 模拟量输入模块

### 7.2.1 技术规范

表 7-4 模拟量输入模块规范

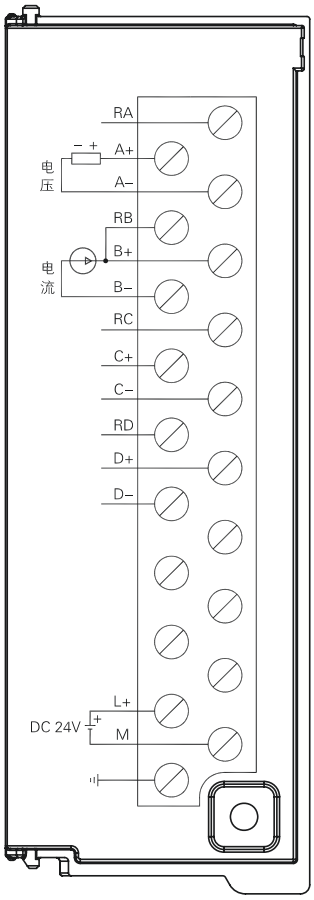
| 特性 | | AIS-040S1 | AIV-080S1 | AIC-080S1 |
|---|---|---|---|---|
| 输入类型 | | 电压或电流（差分输入） | 电压输入 | 电流输入 |
| 输入点数 | | 4 | 8 | |
| 输入量程 | 电压 | 单极性：0~5V，0~10V，双极性：±2.5V，±5V | | |
| | 电流 | 0~20mA，4~20mA | | |
| 最大输入 | 电压 | 30V DC | | |
| | 电流 | 40mA | | |
| 输入阻抗 | 电压 | ≥2MΩ | | |
| | 电流 | 250Ω | | |
| 数据格式 | 电压 | 0~+32000（单极性），-32000~+32000（双极性） | | |
| | 电流 | 0~+32000 | | |
| 输入阶跃响应 | | 4 通道 5ms（最快） | 8 通道 50ms（最快） | |
| 模块更新频率（所有通道） | | 4 通道支持 200Hz、100Hz、50Hz、20Hz、10Hz 配置<br>默认：50Hz 所有通道 | 8通道支持50Hz、20Hz、10Hz、5Hz、2Hz 配置<br>默认：10Hz 所有通道（50Hz 仅满足4通道） | |
| 共模抑制 | | 大于 40dB | | |
| 通道串扰 | | 大于 60dB | | |
| 共模电压 | | -12V≤信号电压+共模电压≤+12V | | |

| | | | |
|---|---|---|---|
| 分辨率 | 单极性 | 12 bits | 16 bits |
| | 双极性 | 11 bits +符号位 | 15 bits +符号位 |
| 测量原理 | | 逐次逼近 | Sigma-delta（Σ-△） |
| 测量误差 | | 0.5%（最大） | 0.1%（最大） |
| 断线检测（仅针对 4~20mA） | | 断线标定：-32768，32767 两值可选 | |
| 隔离 | 现场至逻辑 | 500V AC | |
| | 24VDC 至逻辑 | | |
| 诊断 | 超负量程 | 单极性：0，双极性：-32768 | 单极性：0，双极性：-32768 |
| | 超正量程 | 单极性：32760，双极性：32752 | 单极性：32767，双极性：32767 |
| | 没有电源 | 32736 | 32766 |

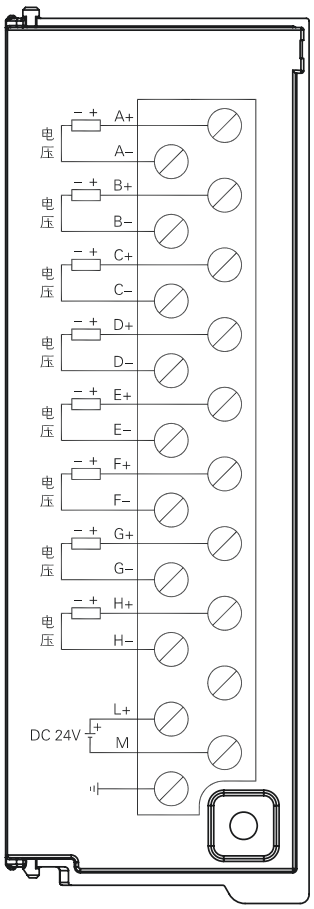
备注：CTH3 AIC-080S1 的输入量程为 4~20mA，当 1.2ma≤输入电流<4ma 时，数值会由 0 到-32767 线性变化，当小于 1.2ma，数值显示-32767。

### 7.2.2 接线规格

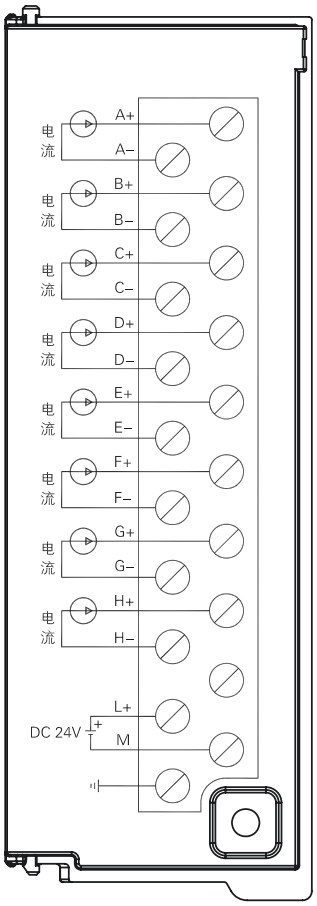
◆ AIS-040S1 接线图




◆ AIV-080S1 接线图




◆ AIC-080S1 接线图



## 7.3 模拟量输出模块

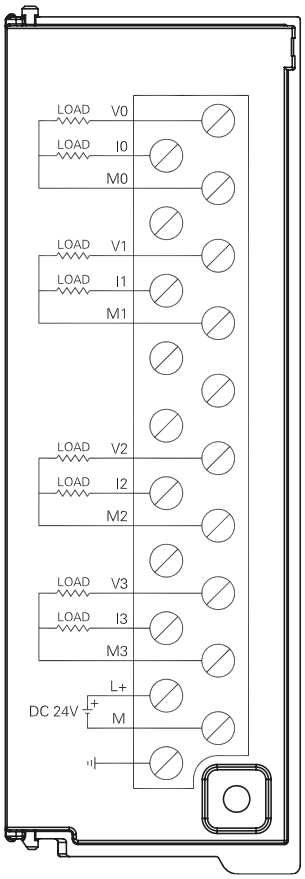
### 7.3.1 技术规范

表 7-5 模拟量输出模块规范

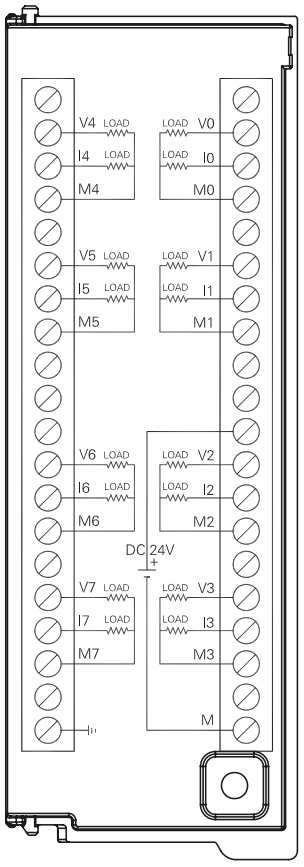
| 特性 | | AQS-040S1 | AQS-080S1 |
|---|---|---|---|
| 输出类型 | | 电压或电流 | |
| 输出点数 | | 4 | 8 |
| 输出量程 | 电压 | ±10V | |
| | 电流 | 0~20mA， 4mA~20mA | |
| 保护 | 输出误接电压 | 最大 30V DC | |
| | 电压短路保护 | 有 | |
| 数据格式 | 电压 | -32000~+32000（满量程时） | |
| | 电流 | 0~+32000（满量程时） | |
| 建立时间 | 电压输出 | 100us | |
| | 电流输出 | 2ms | |
| 负载阻抗 | 电压输出 | 5000Ω（最小） | |
| | 电流输出 | 500Ω（最大） | |
| 分辨率 | 单极性 | 12 bits | |
| | 双极性 | 11 bist +符号位 | |
| 精度 | 电压 | 典型值：满量程的±0.5%，最坏情况：满量程的±2% | |
| | 电流 | 典型值：满量程的±0.6%，最坏情况：满量程的±2% | |
| 隔离 | 电源隔离 | 500V AC | |
| | 现场至逻辑 | | |

### 7.3.2 接线规格

◆ AQS-040S1 接线图




◆ AQS-080S1 接线图




## 7.4 模拟量输入输出模块

### 7.4.1 技术规范

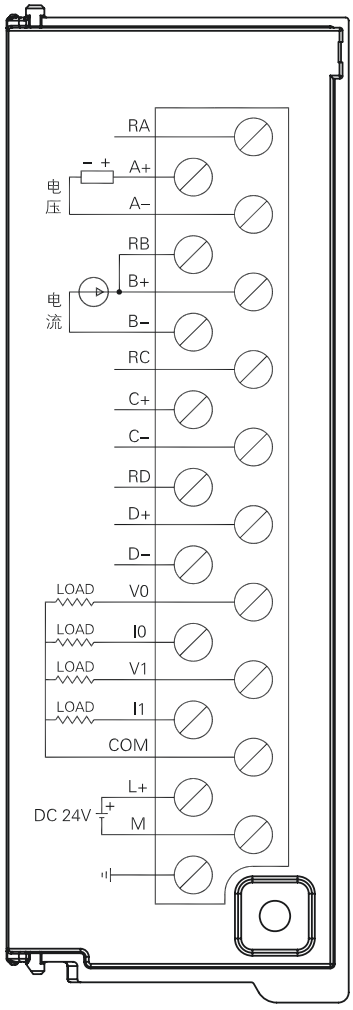
表 7-6 模拟量输入输出模块规范

| **输入特性** | | **AMS-060S1** |
|---|---|---|
| 输入类型 | | 电压或电流（差分输入） |
| 输入点数 | | 4 |
| 输入量程 | 电压 | 单极性：0~5V，0~10V；双极性：±2.5V，±5V |
| | 电流 | 0~20mA，4~20mA |
| 最大输入 | 电压 | 30VDC |
| | 电流 | 40mA |
| 输入阻抗 | 电压 | ≥2MΩ |
| | 电流 | 250Ω |

| | | | |
|---|---|---|---|
| 数据格式 | 电压 | 0~+32000（单极性），-32000~+32000（双极性） | |
| | 电流 | 0~+32000 | |
| 输入阶跃响应 | | 4 通道 5ms（最快） | |
| 模块更新频率（所有通道） | | 4 通道支持 200Hz、100Hz、50Hz、20Hz、10Hz 配置<br>默认：50Hz 所有通道 | 8 通道支持 50Hz、20Hz、10Hz、5Hz、2Hz 配置<br>默认：10Hz 所有通道 |
| 共模抑制 | | 大于 40dB | |
| 通道串扰 | | 大于 60dB | |
| 共模电压 | | -12V≤信号电压+共模电压≤+12V | |
| 分辨率 | 单极性 | 12 bits | |
| | 双极性 | 11 bits+符号位 | |
| 测量原理 | | 逐次逼近 | |
| 测量误差 | | 0.5%（最大） | |
| 断线检测（仅针对 4~20mA） | | 断线标定：-32768，32767 两值可选 | |
| 隔离（现场至逻辑/24VDC 至逻辑） | | 500V AC | |
| 诊断 | 超负量程 | 单极性：0，双极性：-32768 | |
| | 超正量程 | 单极性：32760，双极性：32752 | |
| | 无电源 | 32736 | |
| **输出特性** | | **AMS-060S1** | |
| 输出类型 | | 电压或电流 | |
| 输出点数 | | 2 | |
| 输出量程 | 电压 | ±10V | |
| | 电流 | 0~20mA， 4mA~20mA | |
| 保护 | 输出误接电压 | 最大 30V DC | |
| | 电压短路保护 | 有 | |
| 数据格式 | 电压 | -32000~+32000（满量程时） | |
| | 电流 | 0~+32000（满量程时） | |
| 建立时间 | 电压输出 | 100us | |
| | 电流输出 | 2ms | |
| 负载阻抗 | 电压输出 | 5000Ω（最小） | |
| | 电流输出 | 500Ω（最大） | |
| 分辨率 | 单极性 | 12bit | |
| | 双极性 | 11bit+符号位 | |
| 精度 | 电压 | 典型值：满量程的±0.5%，最坏情况：满量程的±2% | |
| | 电流 | 典型值：满量程的±0.6%，最坏情况：满量程的±2% | |
| 隔离 | 电源隔离 | 500V AC | |
| | 现场至逻辑 | | |

## 7.4.2 接线规格

◆ AMS-060S1 接线图



# 8 温度模块

CTH300-H 系列温度模块用于热电偶、热电阻温度测量，集成 4 个或 8 个输入点，用户可配置温度传感器类型，满足用户多点温度测量需求。

### 本章概要

本章主要介绍 CTH300-H 系列温度模块的如下内容：

- ❑ 技术规范
- ❑ 接线规格

表 8-1 温度模块的基本属性

| 名称 | 规格描述 | 订货号 |
|---|---|---|
| AIT-04 温度模块 | 热电偶输入模块，4 路 TC，隔离型 16 bits 精度 | CTH3 AIT-040S1 |
| AIT-08 温度模块 | 热电偶输入模块，8 路 TC，隔离型 16 bits 精度 | CTH3 AIT-080S1 |
| AIR-04 温度模块 | 热电阻输入模块，4 路 RTD，隔离型 16 bits 精度 | CTH3 AIR-040S1 |
| AIR-08 温度模块 | 热电阻输入模块，8 路 RTD，隔离型 16 bits 精度 | CTH3 AIR-080S1 |

## 8.1 技术规范

### 8.1.1 常规特性

表 8-2 CTH300-H 系列温度模块的常规特性

| 特性 | AIT-040S1 | AIT-080S1 | AIR-040S1 | AIR-080S1 |
|---|---|---|---|---|
| **物理特性** | | | | |
| 尺寸（W×H×D） | 34×115×100 mm | | | |
| **电源特性** | | | | |
| 额定输入电压 | 24V DC | | | |
| 输入电压范围 | 20.4V~28.8V DC | | | |
| 输入电流 | 50mA | | 60mA | 80mA |
| 极性反接保护 | 有 | | | |
| 总线电源电压 | +5V DC | | | |
| 总线电源电流 | 50mA | | | |
| **LED 指示灯特性** | | | | |
| 24V 电源指示灯 | 亮：24VDC 供电正常；灭：无 24VDC 供电 | | | |
| SF 指示灯 | 亮：模块故障；灭：无错；闪烁：有通道断线或超量程 | | | |

### 8.1.2 功能特性

表 8-3 CTH300-H 系列温度模块的功能特性

| 功能类别 | 功能名称 | 描述 |
|---|---|---|
| I/O 功能 | I/O 接口 | 提供 4 路/8 路温度传感器输入接口 |
| 扩展功能 | 扩展功能 | 提供总线扩展功能 |
| 隔离功能 | 信号隔离 | 现场与总线之间隔离 |
| | 电源隔离 | 外部电源与系统电源之间隔离 |
| 保护功能 | 电源保护 | 供电电源端提供反接保护功能及浪涌吸收功能 |
| | 断线检测 | 输入提供断线检测功能 |
| 滤波功能 | 滤波功能 | 采用硬件滤波与软件滤波相结合的方式 |
| 电源功能 | 电源功能 | 模块采用 24VDC 供电 |

### 8.1.3 输入特性

表 8-4 CTH300-H 系列温度模块的输入特性

| 特性 | AIT-040S1 | AIT-080S1 | AIR-040S1 | AIR-080S1 |
|---|---|---|---|---|
| 输入类型 | 悬浮型热电偶 | | 模块参考接地热电阻 | |
| 输入点数 | 4 | 8 | 4 | 8 |
| 接线方式 | - | | 支持 2 线制、3 线制、4 线制；<br>默认：3 线制 | |
| 输入范围 | 热电偶类型（选一种）：<br>S、T、R、E、N、K、J<br>电压范围：±80mV<br>默认：K | | 热电阻类型（选一种）：<br>Pt-100Ω，200Ω，500Ω，1000Ω（α=3850ppm，3920ppm，3850.55ppm，3916ppm，3902ppm）<br>Pt-10000Ω（α=3850ppm）<br>Cu-9.035Ω（α=4720ppm）<br>Ni-100Ω，120Ω，1000Ω（α=6720ppm，6178ppm）<br>R-150Ω，300Ω，600ΩFS<br>默认：Pt-100Ω（α=3850ppm） | |
| 温度测量范围（℃） | S：-50~1768<br>T：-270~400<br>R：-50~1768<br>E：-270~1000<br>N：-270~1300<br>K：-270~1372<br>J：-210~1200<br><备注>测量超范围则报错，诊断详情参考表 8-5。溢出范围参考章节 8.1.4 热电偶特性。 | | Pt-100Ω，200Ω，500Ω，1000Ω：-200~850<br>Pt-10000Ω：-200~600<br>Cu-9.035Ω：-200~260<br>Ni-0.00672：-80~260<br>Ni-0.006178：-60~300<br><备注> 测量超范围则报错，诊断详情参考表 8-5。 | |

| 特性 | | AIT-040S1 | AIT-080S1 | AIR-040S1 | AIR-080S1 |
|---|---|---|---|---|---|
| 隔离 | 现场至逻辑 | 500VAC | | | |
| | 现场至 24VDC | | | | |
| | 24VDC 至逻辑 | | | | |
| 共模抑制 | | ＞100dB@120VAC | | | |
| 输入分辨率 | 温度 | 0.1℃ / 0.1℉ | | | |
| | 电压 | 15 位+符号位 | | - | |
| | 电阻 | - | | 15 bits +符号位 | |
| 测量原理 | | Sigma-Delta | | | |
| 模块更新频率（所有通道） | | 4 通道支持 8Hz、4Hz、2Hz、1Hz 配置，默认：2Hz 所有通道 | | | |
| | | 8 通道支持 4Hz、2Hz、1Hz、0.5Hz 配置，默认：1Hz 所有通道 | | | |
| 到传感器的导线长度 | | 最大 100m | | | |
| 导线回路电阻 | | 100Ω | | 20Ω，Cu 型 2.7Ω | |
| 噪声抑制 | | 85dB@50Hz/60Hz/400Hz | | | |
| 数据字格式 | | 电压：-27648~+27648 | | | |
| 输入阻抗 | | ＞10MΩ | | | |
| 最大输入电压 | | 输入端能支持最大电压为 30V DC 的误接 | | | |
| 分辨率 | | 15 位+符号位 | | | |
| 输入滤波衰减 | | -3dB@21KHz | | -3dB@3.6KHz | |
| 基本误差 | | 0.1%Fs（电压） | | 0.1%Fs（电阻） | |
| 重复性 | | 0.05%Fs | | | |
| 冷端补偿 | | 可配置，默认有冷端补偿 | | - | |
| 冷端误差 | | ±1.5℃ | | - | |
| 温度单位 | | 摄氏度/华氏度可配置，默认摄氏度 | | | |
| 断线检测 | | 热电偶：可配置，默认有断线检测 | | 热电阻：一直有断线检测，不可配置 | |
| | | 支持正负两个方向标定，默认正向标定 | | | |
| PID 控制功能 | | 无 | | - | |

表 8-5　CTH300-H 系列温度模块诊断信息

| 出错类型 | 通道数据 | SF 指示灯 | 24V 指示灯 | 范围状态位 | 24V 电源故障 |
|---|---|---|---|---|---|
| 无模块电源 | 32766 | 熄灭 | 熄灭 | 0 | 1 |
| 断线 | 32767（正向标定）<br>-32768（负向标定） | 闪烁 | 亮起 | 1 | 0 |
| 超出温度范围 | 32767（正向标定）<br>-32768（负向标定） | 闪烁 | 亮起 | 1 | 0 |

**<备注>** 范围状态位是模块出错寄存器字节中的位，可以根据模块顺序查看 SMB500~531，即可查询对应模块的错误代码。SMB500~531 状态位的描述请参考章节 14.5.1 通过编程软件进行诊断。

## 8.1.4 热电偶特性

各种类型热电偶的温度范围（°C）和准确度

| 数据字 (1个数字位=0.1°C) | | 类型J | 类型K | 类型T | 类型E | 类型R,S | 类型N | ±80mV | |
|---|---|---|---|---|---|---|---|---|---|
| 十进制 | 十六进制 | | | | | | | | |
| 32767 | 7FFF | >1200.0°C | >1372.0°C | >400.0°C | >1000.0°C | >1768.0°C | >1300.0°C | >94.071mV | OF |
| ↑ | ↑ | | | | | | | ↑ | ↑ |
| 32511 | 7EFF | | | | | | | 97.071mV | OR |
| : | : | | | | | | | | |
| 27649 | 6C01 | | | | | | | 80.0029mV | |
| 27648 | 6C00 | | | | | | | 80mV | |
| : | : | | | | | ↑ | | | |
| 17680 | 4510 | | ↑ | | | 1768.0°C | | | |
| : | : | | | | | | | | |
| 13720 | 3598 | | 1372.0°C 超出范围 | | | | ↑ | | |
| : | : | | | | | | | | |
| 13000 | 32C8 | ↑ | 1300.0°C | | | | 1300.0°C | | NR |
| : | : | | | | | | | | |
| 12000 | 2EE0 | 1200.0°C | | | ↑ | | | | |
| : | : | | | | | | | | |
| 10000 | 2710 | | | ↑ | 1000.0°C | | | | |
| : | : | | | | | | | | |
| 4000 | 0FA0 | | | -400.0°C | | 400.0°C | | | |
| : | : | | | | | | | | |
| 1 | 0001 | 0.1°C | 0.1°C | 0.1°C | 0.1°C | 0.1°C | 0.1°C | 0.0029mV | |
| 0 | 0000 | 0.0°C | 0.0°C | 0.0°C | 0.0°C | 0.0°C | 0.0°C | 0.0mV | |
| -1 | FFFF | -0.1°C | -0.1°C | -0.1°C | -0.1°C | -0.1°C | -0.1°C | -0.0029mV | |
| : | : | | | | | 低于范围 | | | |
| -500 | FE0C | | | | | -50.0°C | | | |
| -1500 | FA24 | -150.0°C | | | | ↓ | | | |
| : | : | | | | | | | | |
| -2000 | F830 | 低于范围 | -200.0°C | | | | | | |
| : | : | | | | | | | | |
| -2100 | F7CC | -210.0°C | | | | | | | |
| : | : | | 低于范围 | | | | | | |
| -2550 | F60A | | | -255.0°C | -255.0°C | | | | |
| : | : | | | 低于范围 | 低于范围 | | | | |
| -2700 | F574 | ↓ | -270.0°C | -270.0°C | -270.0°C | | -270.0°C | | |
| : | : | | | | | | | | |
| -27648 | 9400 | | ↓ | ↓ | ↓ | | ↓ | -80mV | |
| -27649 | 93FF | | | | | | | -80.0029mV | UR |
| : | : | | | | | | | | |
| -32512 | 8100 | | | | | | | -94.071mV | |
| # | # | | | | | | | ↓ | ↓ |
| -32768 | 8000 | <-210.0°C | <-270.0°C | <-270.0°C | <-270.0°C | <-50.0°C | <-270.0°C | <-94.07mV | UF |
| 全量程范围的精度 | | S0.1% | S0.3% | S0.6% | S0.1% | S0.6% | S0.1% | S0.1% | |
| 精度(无冷端补偿的额定范围) | | S1.5°C | S1.7°C | S1.4°C | S1.3°C | S3.7°C | S1.6°C | S0.10°C | |
| 冷端误差 | | S1.5°C | S1.5°C | S1.5°C | S1.5°C | S1.5°C | S1.5°C | NA | |
| *OF=上溢 OR=超出范围 NR=额定范围 VR=低于范围 UF=下溢 | | | | | | | | | |
| ↑表示所有大于该值但小于断线阀值的模拟量都报告为上溢出值，32767（10x7FFF）<br>↓表示所有小于该值但大于断线阀值的模拟量都报告为下溢出值，-32768（0x8000） | | | | | | | | | |

## 8.1.5 热电阻特性

用于热电阻类型的温度范围（°C）和精度

| 系统字 (1个数字位=0.1°C) | | Pt10000 | Pt100<br>Pt200<br>Pt500<br>Pt1000 | Ni100<br>Ni120<br>Ni1000 | Cu9.035 | 0-150Ω | 0-300Ω | 0-600Ω | |
|---|---|---|---|---|---|---|---|---|---|
| 十进制 | 十六进制 | | | | | | | | |
| 32767 | 7FFF | | | | | | | | |
| 32766 | 7FFE | | | | | ↑ | ↑ | ↑ | |
| 32511 | 7EFF | | | | | 176.383Ω | 352.767Ω | 705.534Ω | |
| 29649 | 6C01 | | | | | 150.005Ω | 300.011Ω | 600.022Ω | |
| 27648 | 6C00 | | | | | 150.000Ω | 300.000Ω | 600.000Ω | |
| 25000 | 61AB | | | | | | | | ↑ |
| 18000 | 4650 | | | | | | | | OR |
| 15000 | 3A98 | | | | | | | | |
| 13000 | 32C8 | ↑ | ↑ | | | | | | |
| 10000 | 2710 | 1000.0°C | 1000.0°C | | | | | | |
| | | | | | | | | | |
| 8500 | 2134 | | 850.0°C | | | | | | |
| 6000 | 1770 | 600.0°C | | | ↑ | | | | |
| 3120 | 0C30 | | | ↑ | 312.0°C | | | | |
| 2950 | 0B86 | | | 295.0°C | | | | | |
| 2600 | 0A28 | | | | 260.0°C | | | | |
| 2500 | 09C4 | | | 250.0°C | | | | | |
| | | | | | | | | | |
| 1 | 0001 | 0.1°C | 0.1°C | 0.1°C | 0.1°C | 0.005Ω | 0.011Ω | 0.022Ω | |
| 0 | 0000 | 0.0°C | 0.0°C | 0.0°C | 0.0°C | 0.000Ω | 0.000Ω | 0.000Ω | |
| -1 | FFFF | -0.1°C | -0.1°C | -0.1°C | -0.1°C | | （负值是不可能的） | | |
| | | | | | | ↓ | ↓ | ↓ | |
| -600 | FDA8 | | | -60.0°C | | | | | |
| | | | | | | | | | |
| -1050 | FBE6 | | | -105.0°C | | | | | |
| | | | | ↓ | | | | | |
| -2000 | F830 | -200.0°C | -200.0°C | | -200.0°C | | | | |
| -2400 | F6A0 | | | | -240.0°C | | | | |
| -2430 | F682 | -243.0°C | -243.0°C | | ↓ | | | | |
| | | ↓ | ↓ | | | | | | |
| -5000 | EC78 | | | | | | | | |
| -6000 | E890 | | | | | | | | UR |
| -10500 | D6FC | | | | | | | | ↓ |
| -12000 | D120 | | | | | | | | |
| -20000 | 4E20 | | | | | | | | |
| | | | | | | | | | |
| -32767 | 8001 | | | | | | | | |
| -32768 | 8000 | | | | | | | | |
| 全量程范围的精度 | | ±0.4% | ±0.1% | ±0.2% | ±0.5% | ±0.1% | ±0.1% | ±0.1% | |
| 精度（额定范围） | | ±4°C | ±1°C | ±0.6°C | ±2.8°C | ±0.15°C | ±0.3°C | ±0.6°C | |

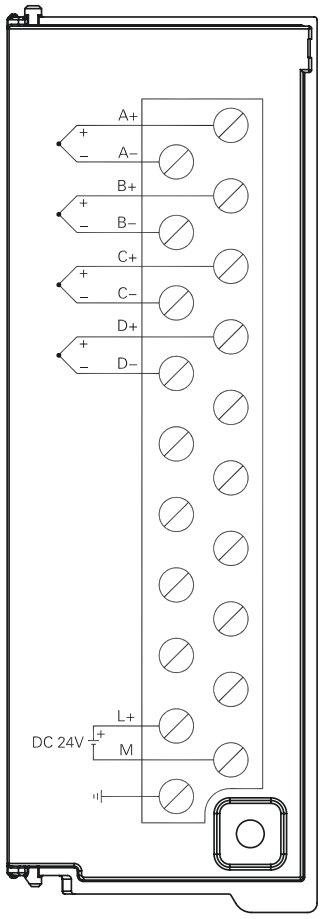
*OF=上溢出 OR=超出范围 NR=额定范围 UR=低于范围 UF=下溢出

↑或↓表示所有超过或低于这个限制值的模拟量其测量值都报告为所选的熔断标定值，32767（0x7FFF）或-32768（0x8000）

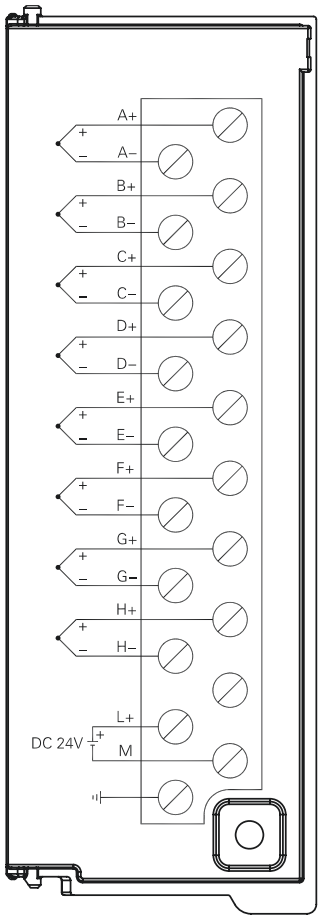
## 8.2 接线规格

### 8.2.1 热电偶接线规格

◆ AIT-040S1 接线规格

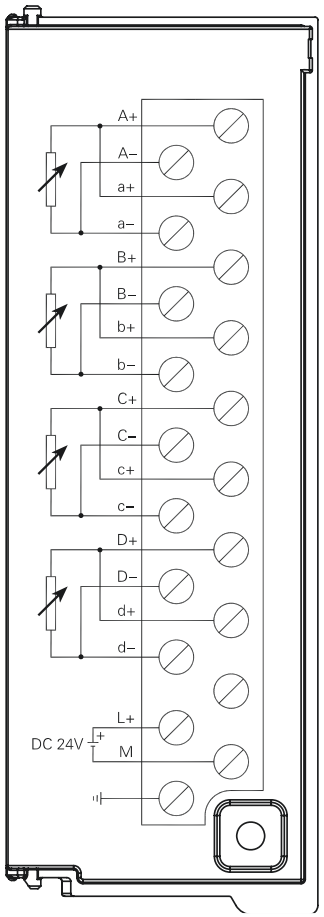



◆ AIT-080S1 接线规格

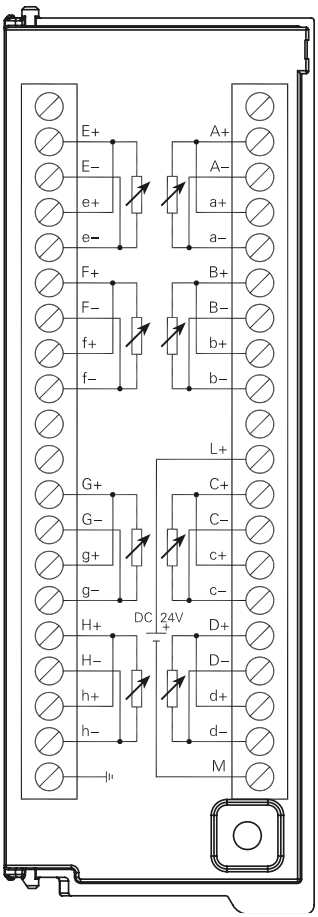


## 8.2.2 热电阻接线规格

◆ AIR-040S1 接线规格



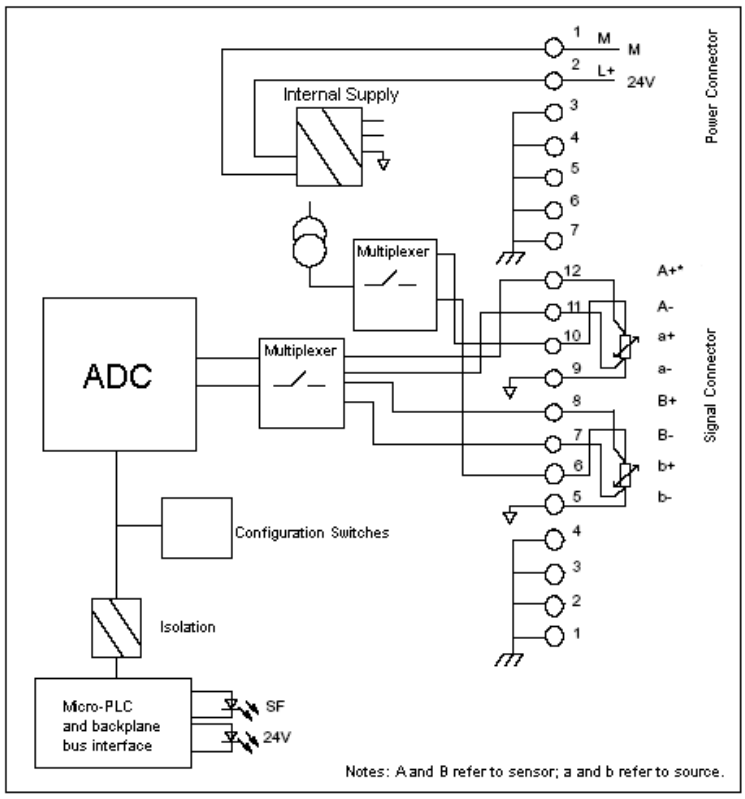

◆ AIR-080S1 接线规格

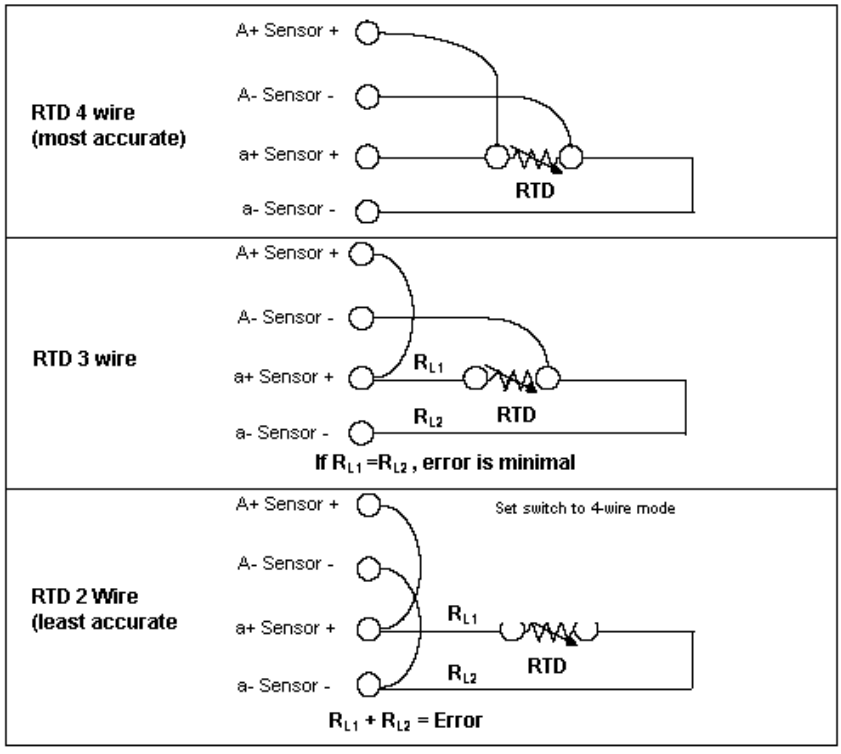



**RTD 与传感器的接线**

如下图所示，用户可采用直接接线或使用扩展线的方式将热电阻传感器接到热电阻模块上。使用屏蔽线可达到最好的抗躁性，若使用屏蔽线，应将屏蔽线接到信号连接器的 1 至 4 针接地点上。该接地点与电源连接器的 3 至 7 针共地。如果有的热电阻输入通道没有使用，用户应将一个电阻器与未使用的输入通道相连，以防止由于浮地输入信号产生的误差影响有效通道的错误显示。

用户需将电源连到电源连接器的 1 和 2 针上，必须将电源连接器的针 3 接到附近的机壳地。可任选下图三种接线方式（2 线制，3 线制，4 线制）之一将热电阻模块与传感器相连。精度最高的是 4 线接法，精度最低的是 2 线接法，建议只有在应用中不在乎导线误差时才用 2 线接法。


1 M
M
2 L+
24V
Internal Supply
3
4
5
6
7
Power Connector
Multiplexer
12
11
10
9
8
7
6
5
4
3
2
1
A+*
A-
a+
a-
B+
B-
b+
b-
Signal Connector
ADC
Multiplexer
Configuration Switches
Isolation
Micro-PLC
and backplane
bus interface
SF
24V
Notes: A and B refer to sensor; a and b refer to source.


RTD 4 wire
(most accurate)
A+ Sensor +
A- Sensor -
a+ Sensor +
a- Sensor -
RTD
RTD 3 wire
A+ Sensor +
A- Sensor -
a+ Sensor +
a- Sensor -
RL1
RL2
RTD
If RL1 =RL2 , error is minimal
RTD 2 Wire
(least accurate
A+ Sensor +
A- Sensor -
a+ Sensor +
a- Sensor -
Set switch to 4-wire mode
RL1
RL2
RTD
RL1 + RL2 = Error
Notes: A refers to sensor; a refers to source

# 9 高速计数模块

## 本章概要

本章主要介绍 CTH300-H 系列 HSC-02 高速计数模块的如下内容：

- ❑ 技术规范
- ❑ 接线规格
- ❑ 软件配置规格

表 9-1 高速计数模块 HSC-02 的基本属性

| 规格描述 | 订货号 |
|---|---|
| 高速计数模块 HSC-02，2路差分/单端信号输入 | CTH3 HSC-020S1 |

## 9.1 技术规范

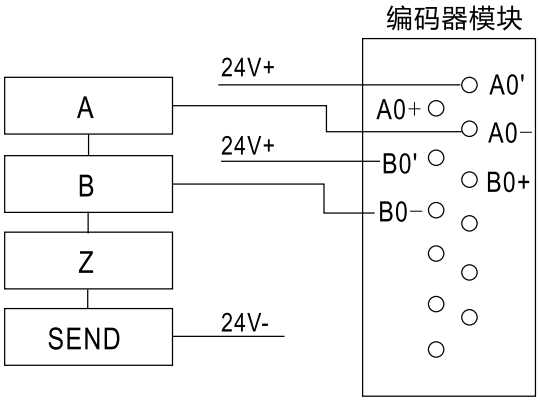
表 9-2 HSC-02 的常规特性

| | | |
|---|---|---|
| **物理特性** | | |
| 尺寸（W×H×D） | | 34×115×100 mm |
| **电源特性** | | |
| 总线电源电压 | | +5V DC |
| 总线电源电流 | | 130mA |
| **LED 指示灯特性** | | |
| 信号指示灯 | | 亮：有输入信号；灭：无输入信号 |
| **传感器连接** | | |
| 输入通道数 | | 2 |
| 信号类型 | 差分输入 | 信号电压：5VDC<br>最高输入频率：2MHz |
| | 单端输入 | 信号电压：24VDC<br>最高输入频率：500KHz<br>信号占空比允许范围：40%-60% |
| 信号输入最大保护电压 | | 30VDC |
| 输入滤波 | | 可配置，125KHz/250KHz/500KHz/1MHz/2MHz |
| 正交易码 | | 1、2、4 倍频 |
| 计数器格式 | | 32 位 |
| 计数器清零功能 | | 支持，Z 信号 |
| 计数器捕捉功能 | | 支持，Z 信号 |
| 多计数器同步计数功能 | | 支持，INT 信号 |

| | |
|---|---|
| INT 信号电压 | 24VDC |
| INT 信号最高输入频率 | 500KHz |
| INT 信号输入滤波 | 可配置，125KHz/250KHz/500KHz |
| 光电隔离 | 500VAC，1min |

## 9.2 接线图

高速计数模块的接线如图右所示，该模块可采用单端信号输出和差分信号输出两种方式，连接不同的驱动器时具体接法如下：



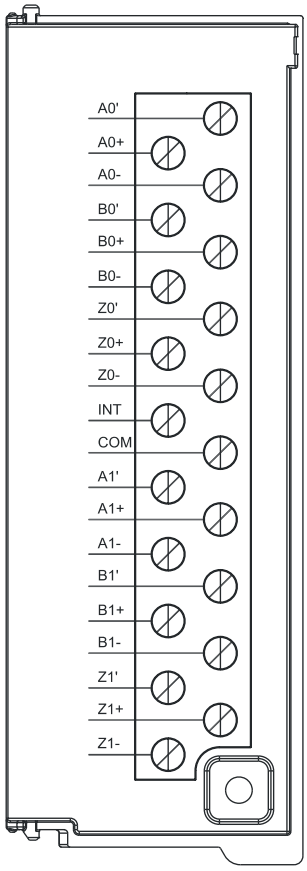




图 9-1　E10 伺服单端信号输出连接编码器模块

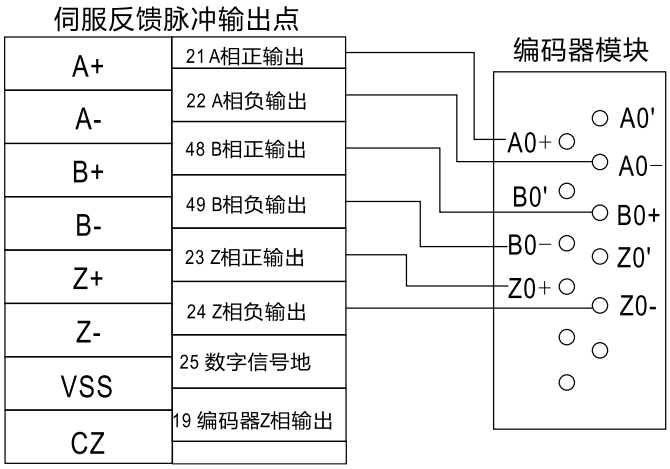



图 9-2　H1A 伺服差分信号输出连接编码器模块

## 9.3 软件配置规格

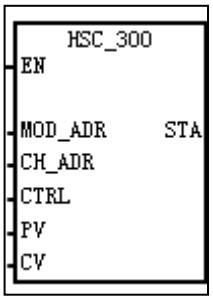
高速计数模块 HSC-02 支持指令库 HSC_300_LIB，以下各节为该库的指令说明。

表 9-3　中断事件

| 计数器 | 中断事件号 | 优先级 | 中断说明 |
|---|---|---|---|
| HSC0 | 12 | 1 | CV=PV 中断 |
| | 27 | 1 | 计数方向改变中断 |
| | 28 | 1 | 外部复位 Z 信号中断 |
| HSC1 | 13 | 1 | CV=PV 中断 |
| | 14 | 1 | 计数方向改变中断 |
| | 15 | 1 | 外部复位 Z 信号中断 |

### 9.3.1　设置计数器指令

函数名：HSC_300




功能：配置计数器参数。

表 9-4　HSC_300 指令的参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>BIT0~BIT3：槽号<br>BIT4~BIT7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）号上的模块<br>例如：Rack16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道<br>0：HSC0；1：HSC1 | BYTE | 0~1 | |
| CTRL | IN | 控制字 | BYTE | | 见下表9-5 |
| PV | IN | 预设值 | DINT | | |
| CV | IN | 当前值 | DINT | | |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |



**提示**

MOD_ADR 模块地址说明了机架号和槽号的取值范围，该参数在多条 HSC 库指令中通用，因此相关设置都是一样。

**示例**

下图两个 HSC 挂在第二个中继的第 4 和第 5 个模块位置，而 MOD_ADR 是要看硬件组态中的模块所在的机架号以及对应的槽号（16#16 和 16#17），而不是模块位置（16#14 和 16#15）。

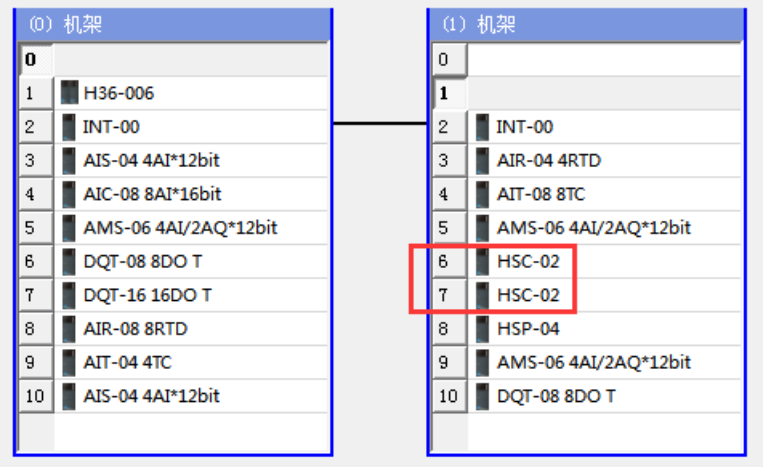



表 9-5　控制字（R/W）

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| hsc_en | hsc_cv_update | hsc_pv_update | hsc_dir_update | hsc_dir | quad_rate | | reset_level |

reset_level：复位电平，1--高电平复位，0--低电平复位

quad_rate[1:0]：正交计数选择，00--4x 倍数，01--2x 倍数，10--1x 倍数

hsc_dir：计数方向，0--减计数，1--增计数

hsc_dir_update：计数方向更新，0--不更新，1--更新

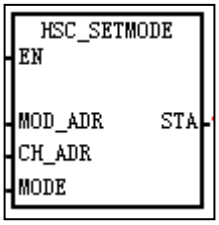
hsc_pv_update：预设值更新，0--不更新，1--更新

hsc_cv_update：当前值更新，0--不更新，1--更新

hsc_en：计数使能，0--不使能，1--使能

## 9.3.2　设置模式指令

函数名：HSC_SETMODE



功能：设置计数器模式。

表 9-6　HSC_SETMODE 指令的参数说明

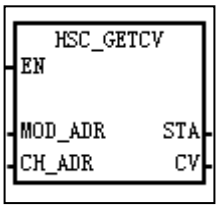
| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>Bit 0~ Bit 3：槽号<br>Bit 4~ Bit 7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）槽号上的模块。例如：Rack16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道 | BYTE | 0~1 | |
| MODE | IN | 计数模式 | BYTE | | Bit0~Bit3：HSC 计数模式（见表 9-7）<br>Bit4：Z 信号锁存功能，0：锁存，1：不锁存<br>Bit5：Z 信号清零功能，0：清零，1：不清零<br>Bit6：预留<br>Bit7：锁存值清零 0：无效，1：有效 |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |

表 9-7　计数器模式

| 模式 | 描述 | 输入 | | | 软件控制 |
|---|---|---|---|---|---|
| | HSC0 | A0 | B0 | Z0 | INT |
| | HSC1 | A1 | B1 | Z1 | |
| 0 | 具有内部方向控制的单相计数器 | 时钟 | - | - | - |
| 1 | | | - | 重设 | - |
| 2 | | | - | | 启动（外部同步） |
| 3 | 具有外部方向控制的单相计数器 | 时钟 | 方向 | - | - |
| 4 | | | | 重设 | - |
| 5 | | | | | 启动（外部同步） |
| 6 | 具有 2 个时钟输入的双相计数器 | 向上时钟 | 向下时钟 | - | - |
| 7 | | | | 重设 | - |
| 8 | | | | | 启动（外部同步） |
| 9 | A/B 相正交计数器 | 时钟 A | 时钟 B | - | - |
| 10 | | | | 重设 | - |
| 11 | | | | | 启动（外部同步） |

### 9.3.3 获取当前计数值指令

函数名：HSC_GETCV




功能：获取当前计数值

表 9-8 HSC_GETCV 指令的参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>Bit0~Bit3：槽号<br>Bit4~Bit7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）槽号上的模块。例如：Rack16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道 | BYTE | 0~1 | |
| CV | OUT | 当前计数值 | DINT | | 当前计数值 |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |

### 9.3.4 获取当前计数状态指令

函数名：HSC_GETSTA




功能：获取当前计数状态。

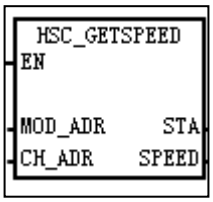
表 9-9 HSC_GETSTA 指令的参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>Bit 0~ Bit 3：槽号<br>Bit 4~ Bit 7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）槽号上的模块例如：Rack16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道 | BYTE | 0~1 | |
| HSC_STA | OUT | 计数器状态 | BYTE | | Bit0~Bit3：当前模式 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Bit4：预留<br>Bit5：HSC0当前计数方向位：1=增计数<br>Bit6=1：当前值等于预设值位<br>Bit7=1：当前值大于预设值位 |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |

### 9.3.5 获取当前速度指令

函数名：HSC_GETSPEED



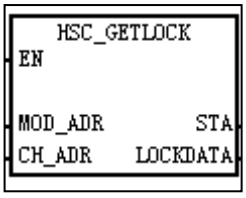

功能：获取当前速度。

表 9-10 HSC_GETSPEED 指令的参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>Bit 0~ Bit 3：槽号<br>Bit 4~ Bit 7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）槽号上的模块。例如：Rack16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道 | BYTE | 0~1 | |
| SPEED | OUT | 当前速度 | DWORD | | Hz |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |

### 9.3.6 获取当前锁存值指令

函数名：HSC_GETLOCK



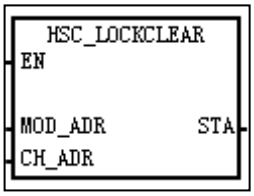

功能：获取当前锁存值。

表 9-11　HSC_GETLOCK 指令的参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>Bit 0~ Bit 3：槽号<br>Bit 4~ Bit 7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）槽号上的模块<br>例如：16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道 | BYTE | 0~1 | |
| LOCKDATA | OUT | 当前锁存值 | DINT | | 当前锁存值 |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |

## 9.3.7　清除锁存值指令

函数名：HSC_CLEARLOCK




功能：清除锁存值。

表 9-12　HSC_CLEARLOCK 指令的参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址<br>Bit 0~ Bit 3：槽号<br>Bit 4~ Bit 7：机架号 | BYTE | | Rackx（0~3）的 y（3~10）槽号上的模块。例如：Rack16 为第 1 个机架的 6 号模块。 |
| CH_ADR | IN | 通道 | BYTE | 0~1 | |
| STA | OUT | 返回状态 | BYTE | 0~255 | 0：OK<br>1：参数错<br>2：访问模块出错 |

# 10 脉冲输出模块

### 本章概要

CTH300-H 系列 HSP-04 脉冲输出模块用于多轴运动控制，支持总线扩展，每个 CPU 或 EtherCAT 从站后面最多可以挂 8 个 HSP 模块（1 个 CPU 后面挂 4 个机架，4 个机架总共可以挂 8 个 HSP 模块）。

本章主要介绍如下内容：

- ❑ 常规特性
- ❑ 接线规格
- ❑ 软件配置规格

表 10-1 脉冲输出模块 HSP-04 的基本属性

| 产品名称 | 规格描述 | 订货号 |
|---|---|---|
| 脉冲输出模块 | 4路差分/单端信号输出 | CTH3 HSP-040S1 |

## 10.1 技术规范

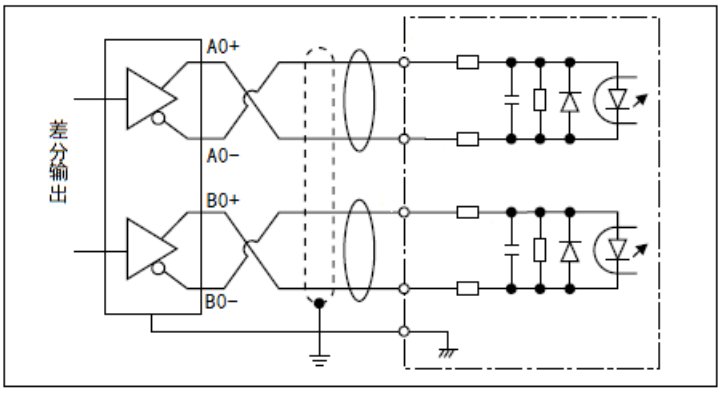
表 10-2 HSP-04 的常规特性

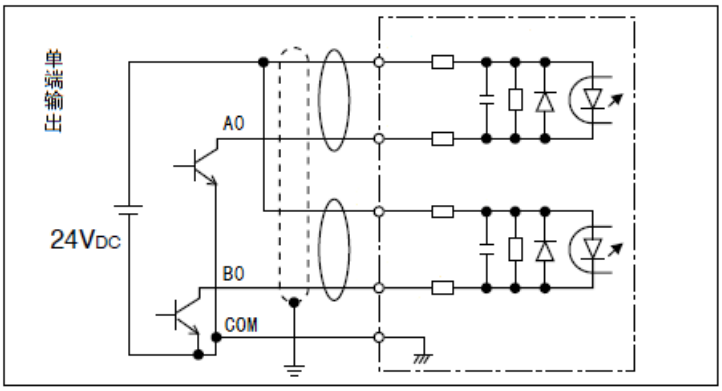
| | | |
|---|---|---|
| **物理特性** | | |
| 尺寸（W×H×D） | 34×115×101.6 mm | |
| **电源特性** | | |
| 额定输入电压 | 24V DC | |
| 输入电压范围 | 20.4V~28.8V DC | |
| 输入电流 | 100mA | |
| 极性反接保护 | 有 | |
| 总线电源电压 | +5V DC | |
| 总线电源电流 | 100mA | |
| **LED 指示灯特性** | | |
| 信号指示灯 | 亮：有输入信号；灭：无输入信号 | |
| **输出特性** | | |
| 输出通道数 | 4 | |
| 输出类型 | 差分信号 | 单端（NPN）信号 |
| 最高输出频率 | 4MHz | 500KHz |
| 输出信号占空比 | - - | 50% |
| 额定输出电压 | 5VDC | 5~24VDC |

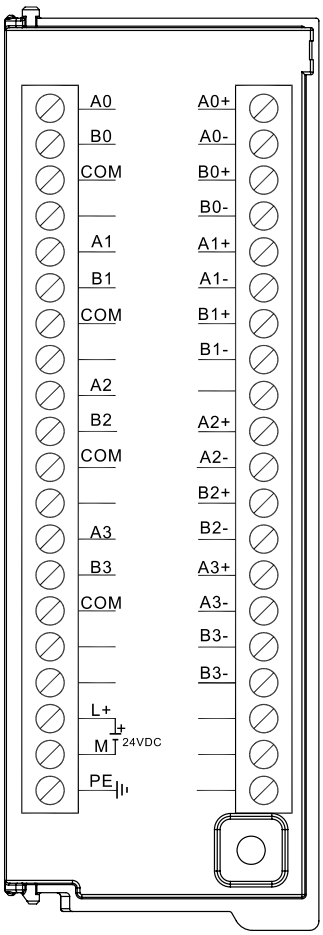
| | | |
|---|---|---|
| 输出电压范围 | 0~5.5VDC | 5~28.8VDC |
| 输出信号逻辑“1” | 3.8V（最小） | 0.5V（最大） |
| 输出信号逻辑“0” | 0.3V（最大） | Vcc-0.5V（最小） |
| 浪涌电流 | 8A，持续 100ms | |
| 每点电流（最大） | 20mA | |
| 每个公共端最大电流 | 无 | 160mA |
| 漏电流（最大） | 10μA | |
| 隔离 | 500VAC，持续 1min | |

## 10.2 接线图

1、接线图右半侧为差分输出，共 4 轴，每轴有两对差分输出，
分别为脉冲及方向（如 A0+、A0-为脉冲，B0+、B0-为方向），
输出频率最高 4MHz；
2、接线图左半侧为单端输出及 24VDC 电源，单端输出共 4 轴，
每轴有两组输出，分别为脉冲及方向（如 A0 为脉冲、B0 为方向，
COM 为公共端），输出频率最高 500KHz。
接线方式分别如下：



## 10.3 软件配置规格

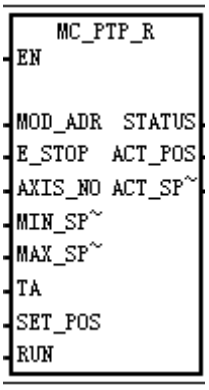
HSP-04 脉冲输出模块配合 MagicWorks PLC 编程软件中的脉冲输出指令库 Hsp_libv1.4 进行使用，包含以下控制指令：

表 10-3 脉冲输出控制指令

| 指令名称 | 指令功能 |
|---|---|
| MC_INIT_DIR | 配置电机的方向指令 |
| MC_PTP_R | 单轴相对运动指令 |
| MC_PTP_A | 单轴绝对位置指令 |
| MC_SPEED_CTRL | 速度控制指令 |
| MC_SET_POS | 设置当前绝对位置 |
| MC_READPOS | 读取当前绝对坐标 |
| MC_SET_MODE | 设置输出模式指令 |

### 10.3.1 单轴相对运动指令

函数名：MC_PTP_R




功能：

用作单轴点对点控制（单轴定长驱动）。

调用一次可输出固定脉冲，通过最大、最小速度和加减速时间的设定，输出的脉冲在启动时会逐渐的加速到最大的速度，当脉冲数快要跑完时，脉冲的频率会自动减下来，以防止在启动或停止时的机器的惯性太大而引起振动或卡死。

表 10-4 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | Byte | 高4位 0~3<br>低4位 3~10 | Bit4~Bit7：机架号<br>Bit3~Bit0：槽号 |
| E_STOP | IN | 紧急停止位。<br>1：有效<br>0：无效 | Bool | 0/1 | 1、只有Run为1与E_Stop为0时运行。<br>2、当E_STOP 为1时，RUN 内部复位。 |

| | | | | | |
|---|---|---|---|---|---|
| AXIS_NO | IN | 设置轴号 | Byte | 0~3 | 该参数在运行过程中不能修改。 |
| MIN_SPEED | IN | 最小速度，即启动时或停止时的速度。单位：HZ | Dword | 100~4000000 | 1、最小速度的设定要小于最大速度。<br>2、此参数在运行过程中可以修改。<br>3、单轴最大速度设为500K，差分最大速度4M。 |
| MAX_SPEED | IN | 最大速度，即运行中的最大速度。单位：HZ | Dword | 100~4000000 | |
| TA | IN | 加速/减速时间。<br>单位：ms | Dword | 0~10000 | 该参数在运行过程中可以修改 |
| SET_POS | IN | 输出的脉冲数，分正负。正脉冲数表示沿X轴的正方向，负脉冲数表示沿着 X 轴的负方向。 | Dint | -2147483648 ~ +2147483647 | 该参数在运行过程中可以修改。<br>当新设定值大于已输出的脉冲数，那么最后输出的脉冲会以新设定值为准。<br>当新设定值小于已输出脉冲数，那么会马上停止脉冲输出。 |
| RUN | IN/OUT | 运行使能位。<br>1：有效<br>0：无效 | Bool | 0/1 | 1、只有 RUN 为1与E_STOP为0时运行。<br>2、当运行完成后，RUN 内部复位。<br>3、当 E_STOP 为 1 时，RUN 内部复位。<br>4. 当指令处于运行状态时，设RUN为0，则实现软停功能。 |
| STATUS | OUT | 输出状态字节：<br>7 6 5 4 3 2 1 0<br>Bit0：参数配置错误标志<br>1—参数配置错误<br>0—参数配置正常<br>Bit1：运行标志<br>1—正在运行，该指令正在输出脉冲，且指令未执行完。<br>0—不运行，因公共资源被其他指令占用，所以指令还没得以运行；或者指令已经运行完毕。<br>Bit2：完成标志 | Byte | 0~255 | Bit0：<br>1、只对轴参数和TA,MOD_ADR配置错误进行判断；<br>2、MIN_SPEED/MAX_SPEED等参数不作报错，会自动设置成一个最接近的合理值。 |

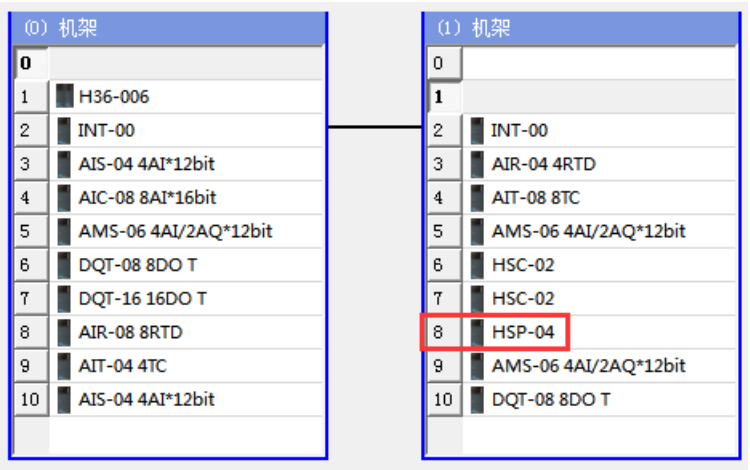
| | | | | | |
|---|---|---|---|---|---|
| | | 1—完成，指令执行完毕。<br>0—未完成，指令未执行或指令正在执行中但未完成。<br>Bit3：忙标志<br>1—忙标志有效，该轴正在被其它指令占用。<br>0—忙标志无效，指令正在执行或此执行已完成。<br>Bit4：模块状态。<br>1：模块访问出错。<br>0：模块访问没有出错。<br>Bit6：被其它指令中止 | | | |
| ACT_POS | OUT | 当前的相对坐标或本指令已输出的脉冲数。 | Dint | -2147483648 ~ +2147483647 | |
| ACT_SPEED | OUT | 当前实际运行速度。 | Dword | 100~4000000 | |

**提示**

MOD_ADR 模块地址说明了机架号和槽号的取值范围，该参数在多条 HSP 库指令中通用，因此相关设置都是一样。

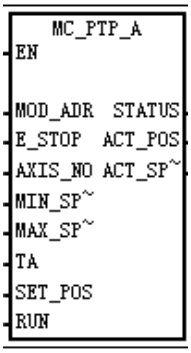
**示例**

下图 HSP 挂在第二个中继的第 6 个模块位置，而 MOD_ADR 是要看硬件组态中的模块所在的机架号以及对应的槽号（16#18），而不是模块所在位置（16#16）。




## 10.3.2 单轴绝对运动指令

函数名：MC_PTP_A




功能：用作单轴点对点控制（非定长，而是定点）。调用一次可在原脉冲数基础上输出脉冲至指定脉冲数，通过最大、最小速度和加减速时间的设定，输出的脉冲在启动时会逐渐的加速到最大的速度，当脉冲数快要跑完时，脉冲的频率会自动减下来，以防止在启动或停止时的机器的惯性太大而引起振动或卡死。

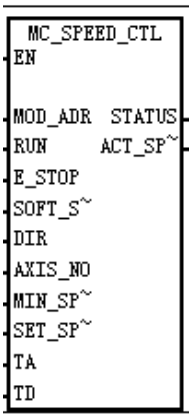
表 10-5 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | Byte | 高4位0~3<br>低4位3~10 | Bit4~Bit7:机架号<br>Bit3~Bit0:槽号 |
| E_STOP | IN | 紧急停止位。<br>1：有效<br>0：无效 | Bool | 0/1 | 1、只有RUN为1与E_STOP为0时运行。<br>2、当E_STOP 为1时，RUN内部复位。 |
| AXIS_NO | IN | 设置轴号 | Byte | 0~3 | 该参数在运行过程中不能修改。 |
| MIN_SPEED | IN | 最小速度，即启动时或停止时的速度。单位：HZ | Dword | 100~4000000 | 1、最小速度的设定要小于最大速度。<br>2、此参数在运行过程中可以修改。<br>3、单轴最大速度设为500K，差分最大速度4M。 |
| MAX_SPEED | IN | 最大速度，即运行中的最大速度。单位：HZ | Dword | 100~4000000 | |
| TA | IN | 加速/减速时间。<br>单位：ms | Dword | 0~10000 | 该参数在运行过程中可以修改 |
| SET_POS | IN | 输出的脉冲数，分正负。正脉冲数表示沿X轴的正方向，负脉冲数表示沿着 X轴的负方向。 | Dint | -2147483648<br>~<br>+2147483647 | 该参数在运行过程中可以修改。<br>当新设定值大于已输出的脉冲数，那么最后输出的脉冲会以新设定值为准；当新设定值小于已输出脉冲数，那么会马上停止 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 脉冲输出。 |
| RUN | IN/OUT | 运行使能位。<br>1：有效<br>0：无效 | Bool | 0/1 | 1、只有 RUN ==1与E_STOP ==0 时才能运行。<br>2、当运行完成后，RUN 内部复位。<br>3、当E_STOP为1时，RUN内部复位。<br>4、当指令处于运行状态时，设RUN为0，则实现软停功能。 |
| STATUS | OUT | 输出状态字节：<br>7 6 5 4 3 2 1 0<br>Bit0：参数配置错误标志<br>1—参数配置错误<br>0—参数配置正常<br>Bit1：运行标志<br>1—正在运行，该指令正在输出脉冲，且指令未执行完。<br>0—不运行，因公共资源被其他指令占用，所以指令未运行；或者指令已经运行完毕。<br>Bit2：完成标志<br>1—完成，指令执行完毕。<br>0—未完成，指令未执行或指令正在执行中但未完成。<br>Bit3：忙标志<br>1—忙标志有效，该轴正在被其它指令占用。<br>0—忙标志无效，指令正在执行或此执行已完成。<br>Bit4：模块状态。<br>1—模块访问出错。<br>0—模块访问没有出错。<br>Bit6:被其它指令中止 | Byte | 0~255 | Bit0：<br>1、只对轴参数和TA,MOD_ADR配置错误进行判断；<br>2、MIN_SPEED/MAX_SPEED等参数不作报错，会自动设置成一个最接近的合理值。 |
| ACT_POS | OUT | 当前的绝对坐标 | Dint | -2147483648~+2147483647 | |
| ACT_SPEED | OUT | 当前实际运行速度。 | Dword | 100~4000000 | |

### 10.3.3 速度控制指令

函数名：MC_SPEED_CTL




功能：控制单轴输出脉冲的频率，可任意时候改变输出脉冲的频率（速度）。当接收到软停止命令时，会自动减速停止。当收到紧急停止命令时，会马上停止脉冲输出，不经过减速。

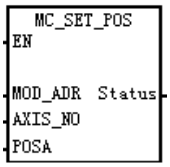
表 10-6 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | Byte | 高4位 0~3<br>低4位 3~10 | Bit4~Bit7：机架号<br>Bit3~Bit0：槽号 |
| RUN | IN | 运行使能位。<br>1：有效，<br>0：无效。 | Bool | 0/1 | 1、只有 RUN为1与E_Stop为0与SOFT_STOP为0时才能运行。<br>2、当运行完成后，RUN内部复位。 |
| E_STOP | IN | 紧急停止位。1:有效，0:无效。当收到有效紧急停止命令后，输出脉冲会马上停止，不经过减速。 | Bool | 0/1 | 只有RUN为1 与E_Stop为0与Soft_Stop为0时才能运行。 |
| SOFT_STOP | IN | 软停止位。1：有效，0：无效。<br>当收到有效软停止命令时，输出脉冲会减速停止。 | Bool | 0/1 | 只有RUN为1与E_Stop为0 与SOFT_STOP为0时才能运行。 |
| DIR | IN | 脉冲的方向位 | Bool | 0/1 | 该参数在运行过程中不能修改。 |
| AXIS_NO | IN | 设置轴号 | Byte | 0~3 | |
| MIN_SPEED | IN | 最小速度，即启动时或停止时的速度。单位：HZ | Dword | 0~4000000 | 1、设定速度为0时，没有脉冲输出；设定最小速度非0且小于100时，模块默认为100；若设定速度小于最小速度，模块默认将最小速度值修改为设定速度的值。 |

| | | | | | |
|---|---|---|---|---|---|
| SET_SPEED | IN | 设定速度，在收到停止命令前，输出脉冲会加速或减速到此速度。 | Dword | 0~4000000 | 2、SET_SPEED在运行过程中可修改。MIN_SPEED在运行过程中可修改。 |
| TA | IN | 加速时间，从最小速度到设定速度的加速时间。<br>单位：毫秒 | Dword | 0~10000 | 该参数在运行过程中可以修改。 |
| TD | IN | 减速时间，从设定速度到最小速度的减速时间。<br>单位：毫秒 | Dword | 0~10000 | |
| STATUS | OUT | 输出状态字节：<br>7 6 5 4 3 2 1 0<br>Bit0：参数配置错误标志<br>1—参数配置错误<br>0—参数配置正常<br>Bit1：运行标志<br>1—正在运行，该指令正在输出脉冲，且指令未执行完。<br>0—不运行，因公共资源被其他指令占用，所以指令还未运行；或者指令已运行完毕。<br>Bit2：完成标志<br>1—完成，指令执行完毕。<br>0—未完成，指令未执行或指令正在执行中但未完成。<br>Bit3：忙标志<br>1—忙标志有效，该轴正在被其它指令占用。<br>0—忙标志无效，指令正在执行或此执行已完成。<br>Bit5~Bit6：预留<br>Bit7：指令通信状态标志<br>1—通信超时；0—无超时 | BYTE | 0~255 | Bit0：<br>1、只对轴参数和TA/TD,MOD_ADR配置错误进行判断；<br>2、MIN_SPEED/SET_SPEED等参数不作报错，会自动设置成一个最接近的合理值。 |
| ACT_SPEED | OUT | 当前速度（频率）输出。 | Dword | 100~400000 | 该值可能跟实际值会有一点偏差。 |

### 10.3.4 设置当前绝对坐标

函数名：MC_SET_POS



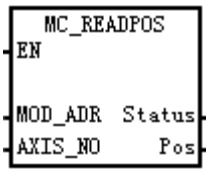

功能：设置当前绝对坐标，本指令只能在轴停止时使用。

表 10-7 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | BYTE | 高4位0~3<br>低4位3~10 | Bit4~Bit7：机架号<br>Bit3~Bit0：槽号 |
| AXIS_NO | IN | 设置轴号 | BYTE | 0~3 | |
| POSA | IN | 设定的坐标 | | -2147483648~<br>+2147483647 | - |
| Status | OUT | 状态 | BYTE | 0~255 | 0：OK<br>1：参数出错<br>2：访问模块出错 |

### 10.3.5 获取当前绝对位置指令

函数名：MC_READPOS




功能：读取当前位置的绝对坐标值指令。

表 10-8 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | Byte | 高4位0~3<br>低4位3~10 | Bit4~Bit7：机架号<br>Bit3~Bit0：槽号 |
| AXIS_NO | IN | 设置轴号 | Byte | 0~3 | 该参数在运行过程中不能修改。 |
| STATUS | OUT | 状态 | Byte | 0~255 | 0：OK<br>1：参数出错<br>2：访问模块出错 |
| POS | OUT | 当前的绝对坐标 | Dint | -2147483648~<br>+2147483647 | |

### 10.3.6 设置电机方向指令

函数名：MC_INIT_DIR




功能：配置电机的方向。

**提示**

执行此指令只在 CPU 上电第一个扫描周期执行一次。

表 10-9 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | Byte | 高4位0~3<br>低4位3~10 | Bit4~Bit7：机架号<br>Bit3~Bit0：槽号 |
| AXIS_NO | IN | 设置轴号 | Byte | 0~3 | 该参数在运行过程中不能修改。 |
| DIR | IN | 配置方向信号为正向时的有效电平<br>DIR为1时，设置对应方向轴输出“1”时为电机正转<br>DIR为0时，设置对应方向轴输出“0”时为电机反转 | Bool | 0~1 | 默认值：0，即默认方向轴输出为“1”，为电机正转。 |
| STATUS | OUT | 状态 | Byte | 0~255 | 0：OK<br>1：参数出错<br>2：访问模块出错 |

## 10.3.7 设置轴输出模式

函数名：MC_SET_MODE




功能：设置轴输出模式

表 10-10 指令参数说明表：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|---|
| MOD_ADR | IN | 模块地址 | Byte | 高4位0~3<br>低4位3~10 | Bit4~Bit7：机架号<br>Bit3~Bit0：槽号 |
| AXIS_NO | IN | 设置轴号 | Byte | 0~3 | |

| | | | | | |
|---|---|---|---|---|---|
| MODE | IN | 设定的模式 | Byte | 0，1，2 | 0：方向 + 脉冲<br>1：正负脉冲<br>2： AB相模式 |
| Status | OUT | 状态 | Byte | 0~255 | 0：OK<br>1：参数出错<br>2：访问模块出错 |

# 11 CAN 主站模块

CAN 主站通信模块 CAN-1M 是 CTH300-H 系统扩展模块中的一员，属于 CANopen 主站模块，用于与其它从站模块进行 CANopen 通信。

### 本章概要

本章主要介绍 CTH300-H 系列 CAN-1M 主站模块的如下内容：

- 技术规范
- 接线规格

表 11-1 主站模块 CAN-1M 的基本特性

| 名称 | 规格描述 | 订货号 |
|---|---|---|
| CAN-1M 主站模块 | CANopen 主站通信模块，1 个 CAN 通信口 | CTH3 CAN-1M0S1 |

## 11.1 技术规范

### 11.1.1 常规特性

表 11-2 CAN-1M 的常规特性

| | |
|---|---|
| **物理特性** | |
| 尺寸（W×H×D） | 34×115×101.6 mm |
| 功耗 | 2.5W |
| **电源特性** | |
| 额定输入电压 | 24V DC |
| 输入电压范围 | 20.4V~28.8V DC |
| 输入电流 | 100mA |
| 极性反接保护 | 有 |
| 总线电源电压 | +5V DC |
| 总线电源电流 | 100mA |
| **LED 指示灯特性** | |
| 24V 电源指示灯 | 亮：24VDC 供电正常；灭：无 24VDC 供电 |
| SF 指示灯 | 亮：主站没有初始化；闪烁：初始化中；灭：初始化完成 |
| BF 指示灯 | 亮：从站有诊断；灭：从站没有诊断 |
| DX 指示灯 | 亮：组态数据有错。如从站个数超过 32 个，20K 波特率从站超过 8 个 |

### 11.1.2 功能特性

表 11-3 CAN-1M 的功能特性

| 功能类别 | 功能项 | 描述 |
|---|---|---|
| 扩展功能 | 扩展功能 | 提供总线扩展功能 |
| 通讯功能 | CAN 接口 | 提供 1 个 CANopen 通信接口 |
| 隔离功能 | 电源隔离 | 外部电源与系统电源之间隔离 |
| | 通讯隔离 | CAN 通信口隔离 |
| 保护功能 | 电源保护 | 供电电源端提供反接保护功能及浪涌吸收功能 |
| | 接口保护 | 通讯端口防雷保护 |

<备注> CAN_1M 模块做主站时，最多可以配置写 300 个字节和读 300 个字节的数据。

### 11.1.3 通信口规范

表 11-4 CAN-1M 的通信口规范

| **CANopen 通讯** | | | | | | | |
|---|---|---|---|---|---|---|---|
| 通讯接口 | 1 个 CAN 通讯口 | | | | | | |
| 最大主站点数 | 1 个 CPU 后面最多可连接 8 个主站 | | | | | | |
| 最大从站点数 | 1 个主站后面最多可连接 32 个从站 | | | | | | |
| 传输速率（kbit/s） | 1000 | 800 | 500 | 250 | 125 | 50 | 20 |
| 最大长度（m） | 25 | 50 | 100 | 250 | 500 | 1000 | 2500 |
| 隔离 | 通信口隔离 | | | | | | |

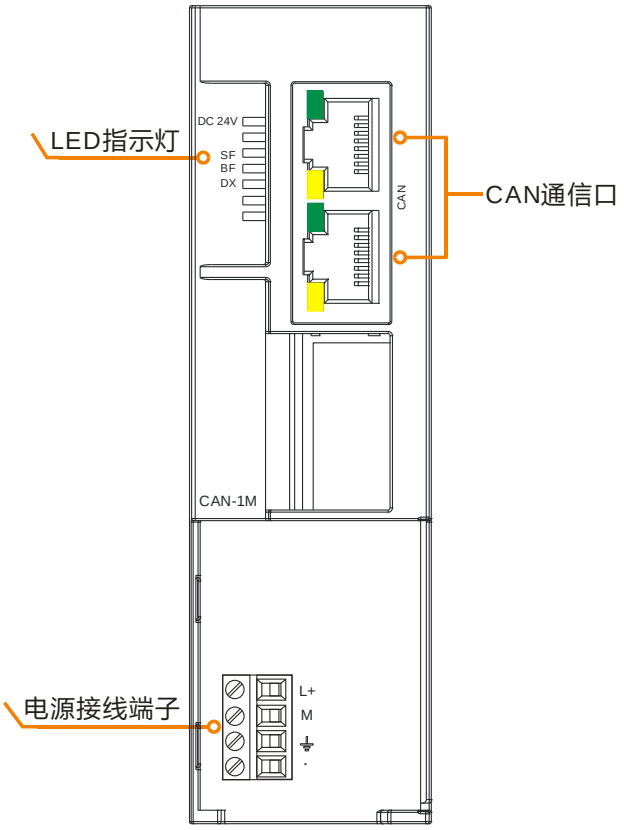
CAN 通信口采用带屏蔽的网线作为通信线，可供选择的网线类型为 22AWG~25AWG，其规格标准如下表，电阻值为单根导线的直流电阻值，推荐使用全屏蔽五类线或者全屏蔽超五类线，24AWG。

表 11-5 通信口线材规格

| AWG | 外径公制 mm | 外径英制 inch | 截面积 $mm^2$ | 电阻值 W/km |
|---|---|---|---|---|
| **22** | 0.643 | 0.0253 | 0.3247 | 54.3 |
| **23** | 0.574 | 0.0226 | 0.2588 | 48.5 |
| **24** | 0.511 | 0.0201 | 0.2047 | 89.4 |
| **25** | 0.44 | 0.0179 | 0.1624 | 79.6 |

## 11.2 接线规格

### 11.2.1 接口示意图

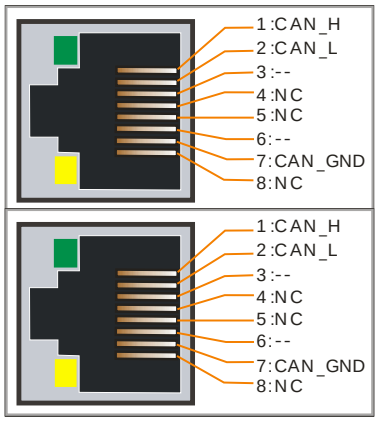



### 11.2.2 接口定义

表 11-6 CAN-1M 的电源接口定义

| 4 位可拆卸端子 | 位号 | 信号 | 信号定义 |
|---|---|---|---|
| L+<br>M<br>⏚<br>• | 1 | L+ | 24V 电源正 |
| | 2 | M | 24V 电源负 |
| | 3 | ⏚ | 大地 |
| | 4 | -- | -- |

表 11-7　CAN-1M 的 CAN 通信口定义




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

# 12 中继模块

中继模块 INT-00 是 CTH300-H 系统的一个重要组成部分，为多机架、多点数的扩展提供了保证。

### 本章概要

本章主要介绍 CTH300-H 系列 INT-00 中继模块的如下内容：

- ❑ 常规特性
- ❑ 接线规格

## 12.1 技术规范

表 12-1 中继模块 INT-00 的基本属性

| 产品名称 | 规格描述 | 订货号 |
|---|---|---|
| 中继模块 | INT-00中继扩展模块 | CTH3 INT-000S1 |

### 12.1.1 常规特性

表 12-2 INT-00 的常规特性

| **物理特性** | |
|---|---|
| 尺寸（W×H×D） | 34×115×101.6 mm |
| 功耗 | 19.5W |
| **电源特性** | |
| 额定输入电压 | 24V DC |
| 输入电压范围 | 20.4V~28.8V DC |
| 输入电流 | 0.8A |
| 极性反接保护 | 有 |
| 总线电源电压 | +5V DC |
| 总线电源电流 | 1.6A |
| **LED 指示灯特性** | |
| 24V 电源指示灯 | 亮：24VDC 供电正常；灭：无 24VDC 供电 |
| SF 指示灯 | 亮：模块故障；灭：无错 |

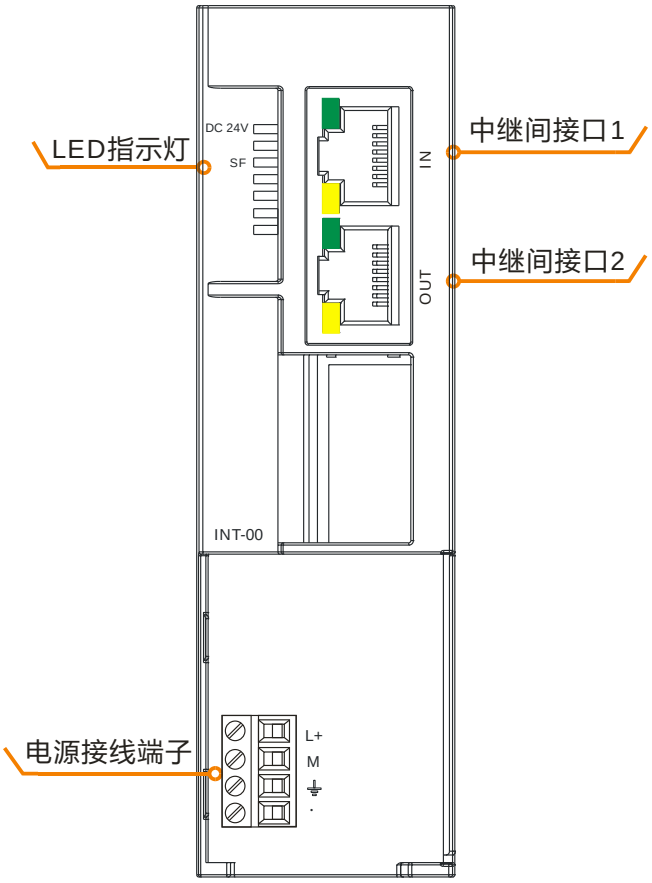
### 12.1.2 功能特性

表 12-3 INT-00 的功能特性

| 功能类别 | 功能项 | 描述 |
|---|---|---|
| 扩展功能 | 扩展总线接口 | 提供总线扩展功能 |
| 通讯功能 | 中继间接口 | 提供中继之间通信接口 |
| 扩展功能 | 扩展功能 | 提供允许扩展 8 个 I/O 模块 |
| 隔离功能 | 电源隔离 | 外部电源与系统电源之间隔离 |
| 保护功能 | 电源保护 | 供电电源端提供反接保护功能及浪涌吸收功能 |

## 12.2 接线规格

### 12.2.1 接口示意图




请按照以下说明连接中继模块，如果接错则会导致通信失败。

中继间接口 1（IN）：连接上一个中继模块的接口（若为第一个中继模块，此口不接）。

中继间接口 2（OUT）：连接下一个中继模块的接口（若为最后一个中继模块，此口不接）。

注意：第一个中继模块要挂在 CPU 后面并插上总线排插，第二个中继模块总线辫悬空。

### 12.2.2 接口定义

表 12-4 INT-00 的电源接口定义

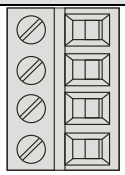
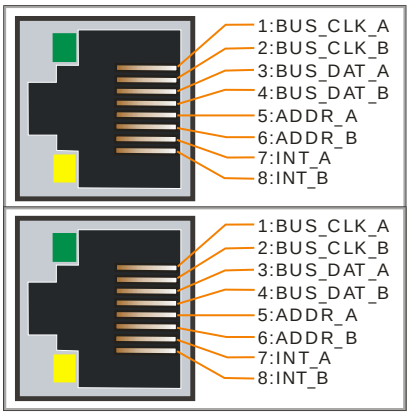
| 4 位可拆卸端子 | 位号 | 信号 | 信号定义 |
|---|---|---|---|
| L+<br>M<br>⏚<br>•  | 1 | L+ | 24V 电源正 |
| | 2 | M | 24V 电源负 |
| | 3 | ⏚ | 大地 |
| | 4 | -- | -- |

表 12-5 INT-00 的双 RJ45 接口定义




| 位号 | 信号 | 信号定义 |
|---|---|---|
| 1 | BUS_CLK_A | 总线时钟 |
| 2 | BUS_CLK_B | |
| 3 | BUS_DAT_A | 总线数据 |
| 4 | BUS_DAT_B | |
| 5 | ADDR_A | 配置地址 |
| 6 | ADDR_B | |
| 7 | INT_A | 中断 |
| 8 | INT_B | |
| 连接器外壳 | PE | 机壳接地 |

### 12.2.3 中继模块网线制作

CTH300 系列 PLC 中继模块所使用的网线为非标准网线，制作网线时必须确保将网线中的信号线成对组合，即：白橙-橙、白绿-绿、白蓝-蓝、白棕-棕，而这四者之间的顺序可随意组合。



# 13 轴配置及电子凸轮

H3X-01 系列 CPU 支持轴配置及电子凸轮功能，用户可以结合编程软件 MagicWorks PLC（V2.17 及以上版本）进行相关配置。

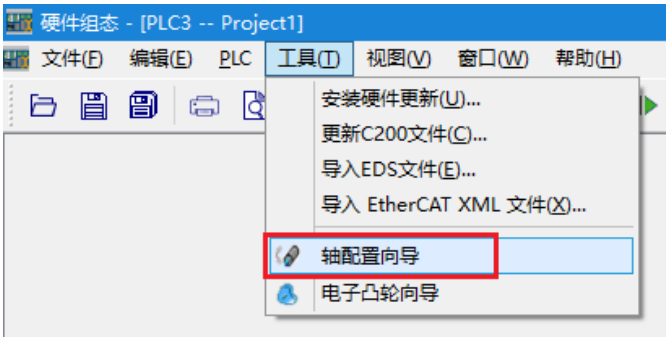
### 本章概要

本章主要介绍 CTH300-H 系列可编程控制器的使用示例，具体内容如下：

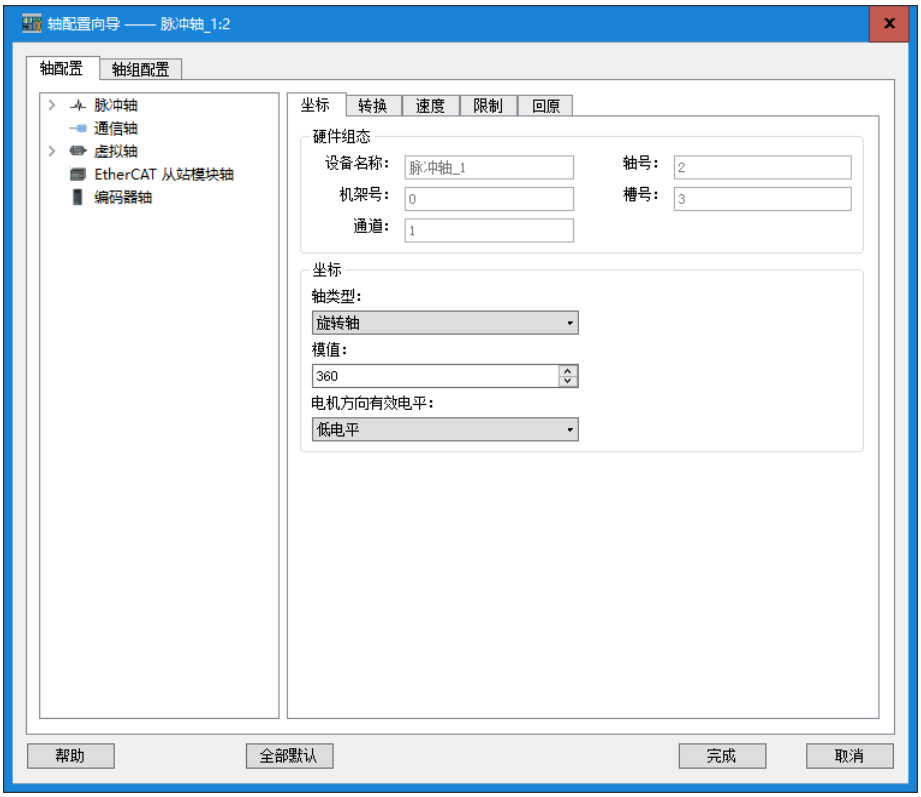
- 轴配置
- 电子凸轮
- 主控模块 Axis 指令说明
- 回原功能介绍

## 13.1 轴配置向导

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**

1、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。

2、PLC 程序中可通过指令 MC_Readstatus 获取轴当前状态；若想获取轴通信状态，直接调用 SMB400、SMB401 和 SMB402。

**【坐标】**

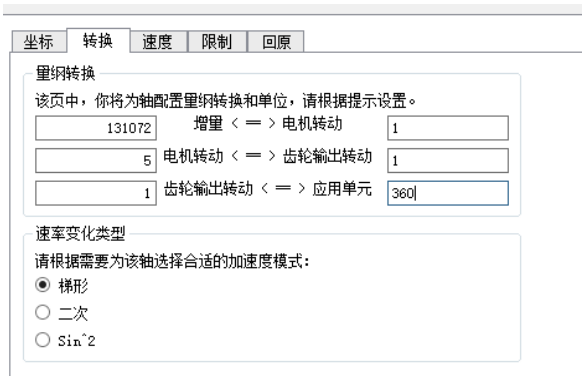



**硬件组态：**“设备名称”为添加该轴时的轴名称；“轴号”为该轴添加的顺序 ID，不可修改，随轴的添加依次递增，若删除某条轴，则新添加的轴轴号为上一次删除轴轴号；“机架号”及“槽号”为该轴对应 HSP 模块所在的机架号及槽号；“通道号”为该轴在 HSP 模块中对应的通道号，范围 0~3（仅脉冲轴及 EtherCAT 轴有通道选项）。

**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，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”。

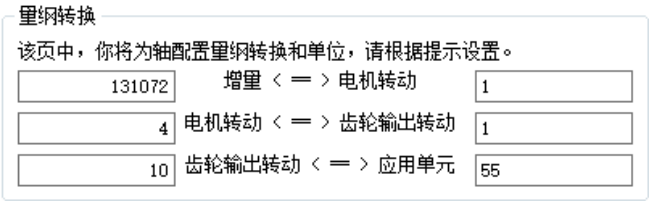
**【转换】**



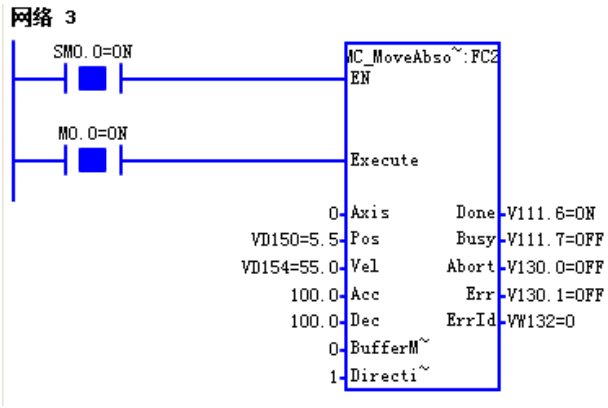

在“转换”界面可根据提示设置纲量转换和速率变化类型。将上图中的“应用单元”设为 360，当程序指令要求伺服运行 1 个单位时，齿轮端将会选择 1/360 圈（轴运动 1 度），伺服电机端会旋转 5*1/360 圈（电机旋转 5 度），以此类推。按照实际机械结构设定对应参数（即电子齿轮比）之后，就可按照应用系统的运动距离物理单位输入距离命令，使参数控制直观易懂。

**<备注>** 量纲转换的任一变量取负值，则电机方向与默认的方向相反。

**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：




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：



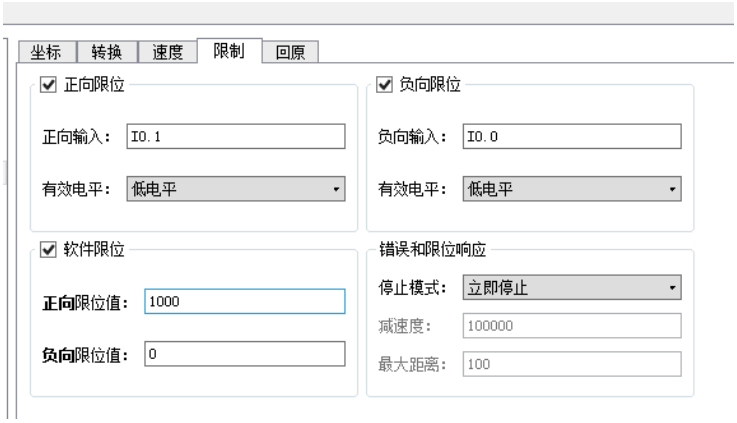
**【速度】**




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：

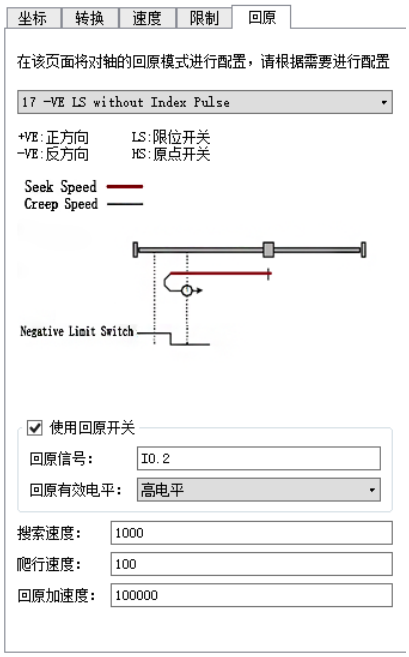
| Errorcode | 意义 |
|---|---|
| 5 | 速度取值太小（大于最小速度） |
| 6 | 速度取值太大（大于最大速度） |
| 7 | 加速度取值太小 |
| 8 | 加速度取值太大 |
| 9 | 减速度取值太小 |
| 10 | 减速度取值太大 |

**【限制】**




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】**



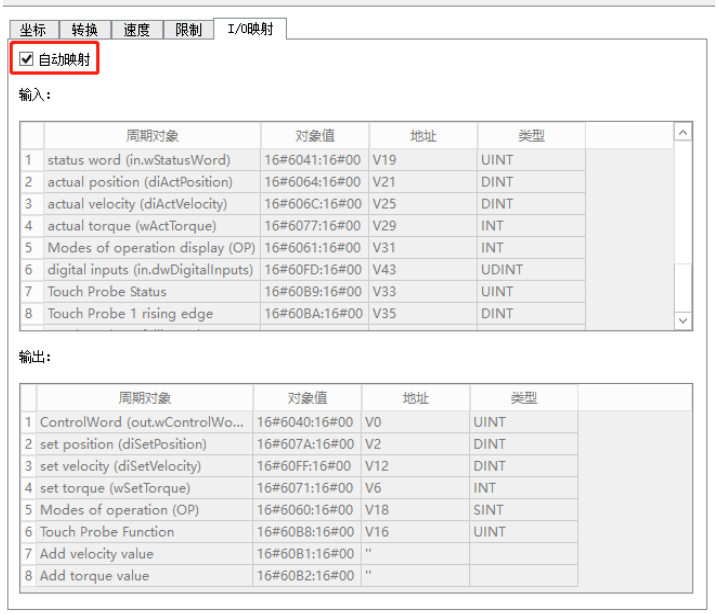

在此页面可对轴的回原模式进行配置，此配置仅脉冲轴支持，选择下拉选项可查看回原模式，共有 17 种回原模式可选。

搜索速度是指还没找到原点之前，寻找原点的速度；

爬行速度是指找到原点之后，从某个方向靠近原点的速度；

回原加速度是指搜索速度的加速度，为了靠近原点的精度，爬行速度通常都很低，不用加速。

**【I/O 映射】**

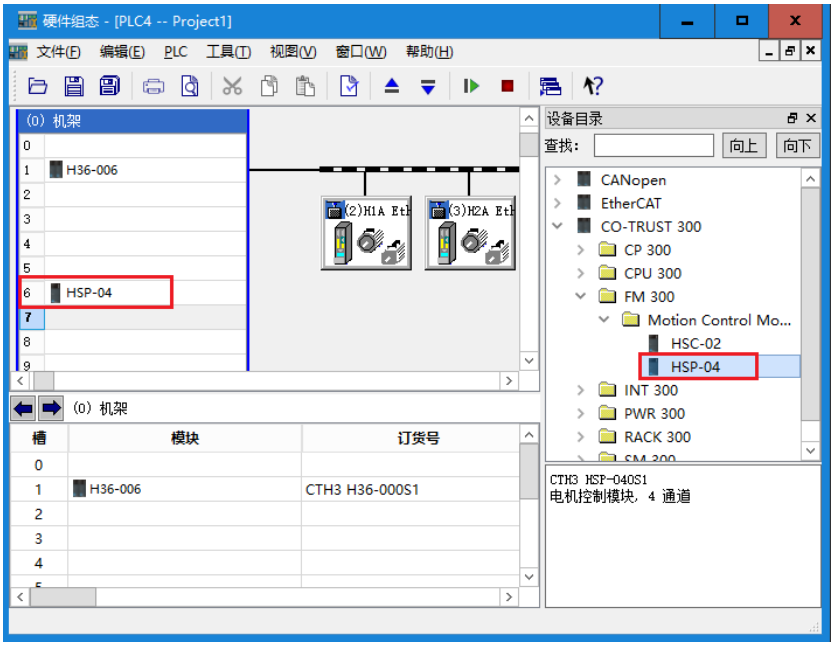



此配置仅通信轴和 EtherCAT 轴支持，在此页面勾选“自动映射”则输入输出均不可修改，建议勾选，因带和不带 CAI402 轴的伺服索引地址不同。

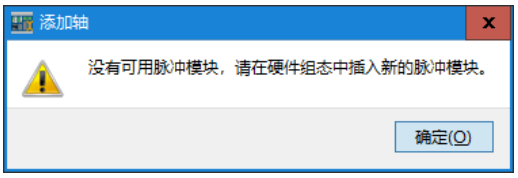
### 13.1.1 轴配置

#### 1、脉冲轴配置

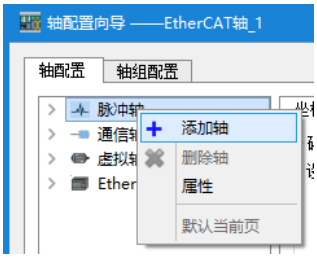
添加脉冲轴前，需先在主站所在机架中添加运动控制模块“HSP-04”，如下图所示：

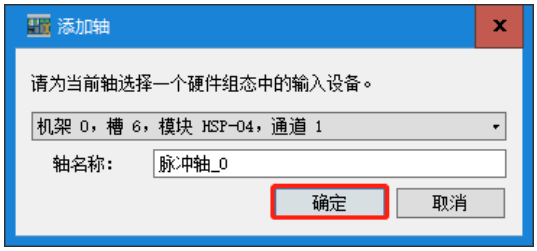



若未添加“HSP-04”，将出现以下提示：

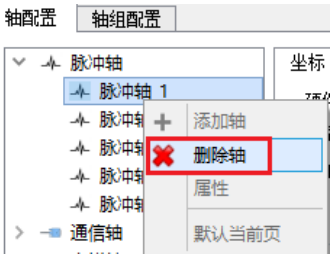



完成以上配置后在“工具”选项中选择“轴向导配置”，在轴向导配置界面鼠标右键点击脉冲轴即可选择添加轴，轴名称可自由更改，点击“确定”即成功添加轴，如下所示：



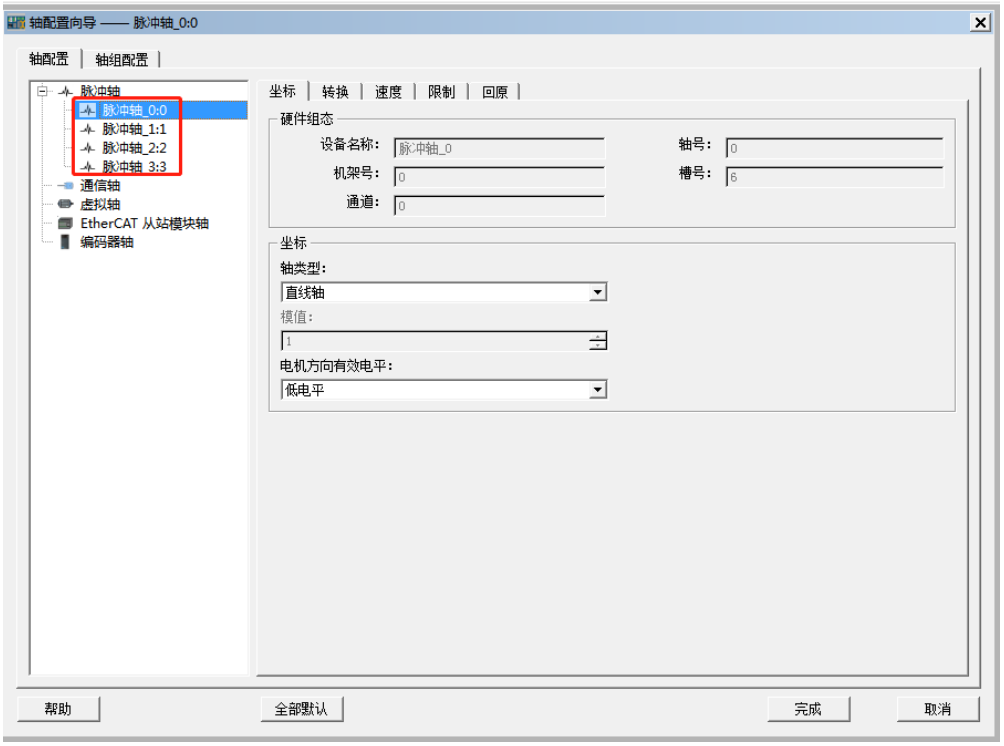

右键选中添加的轴，可选择删除轴，如下图：



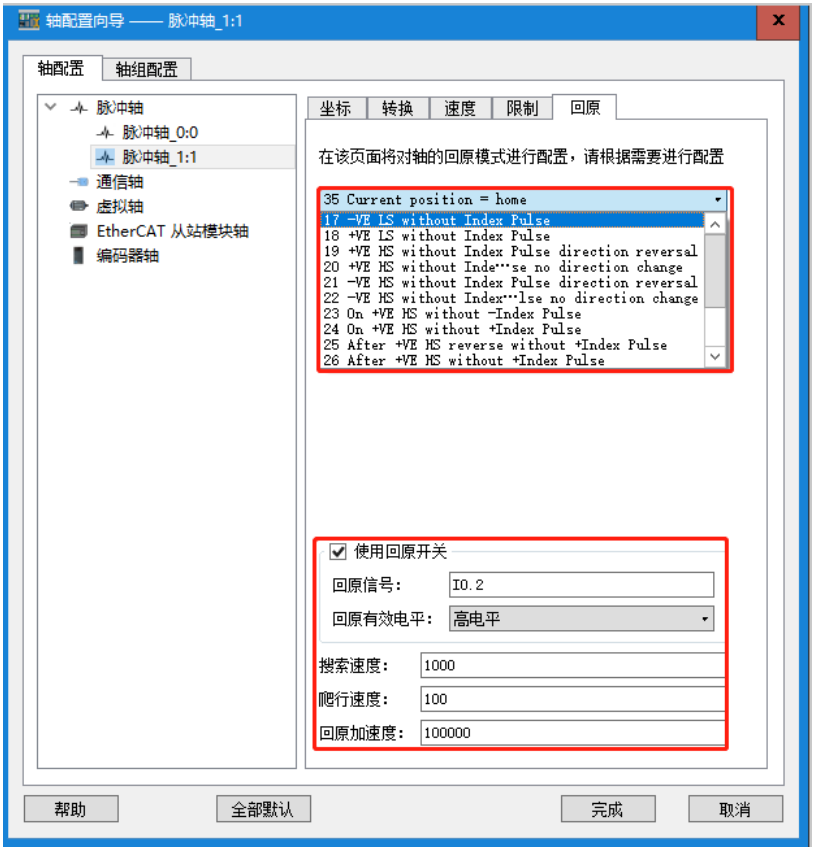
**提示**

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：



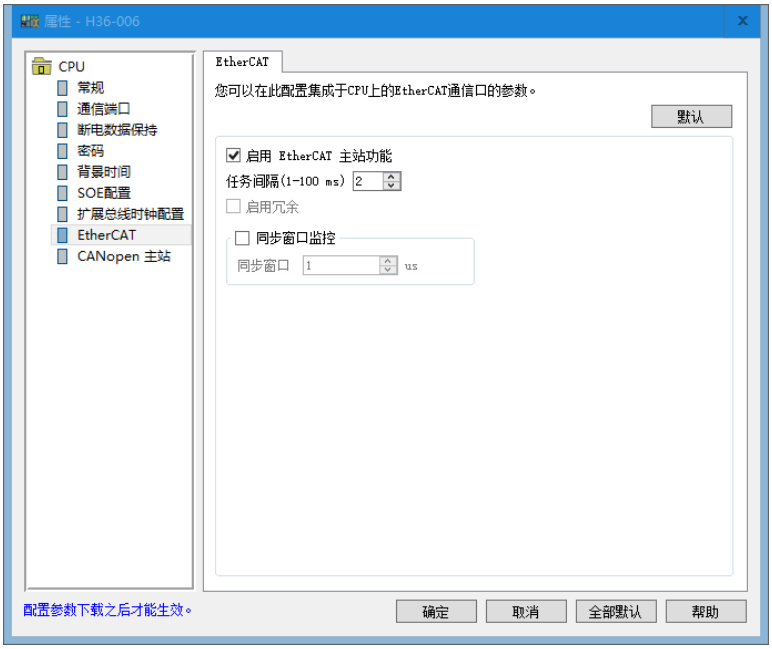

若需要使用回原功能，则点击“回原”，选择回原模式（共有 17 种回原模式），勾选“使用回原开关”，设置回原参数、回原有效电平、搜索速度、爬行速度、回原加速度。



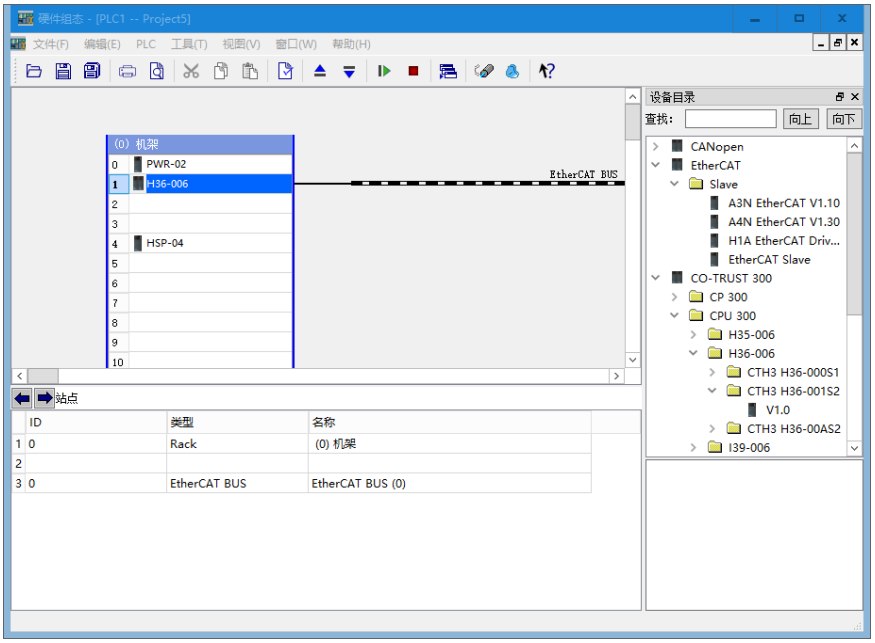

## 2、通信轴配置

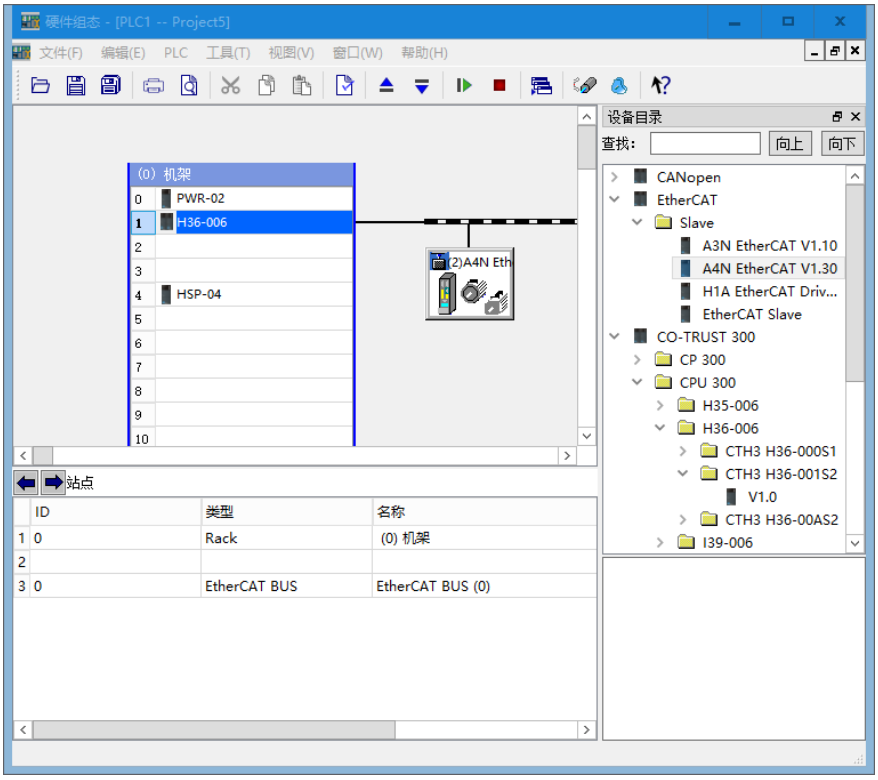
添加通信轴前需在硬件组态界面添加相应的 EtherCAT 从站，否则将无法添加通信轴。具体操作如下：

1）右击机架上的 H36，选择“属性”→“EtherCAT”，点击页面中的“启用 EtherCAT 主站功能”。

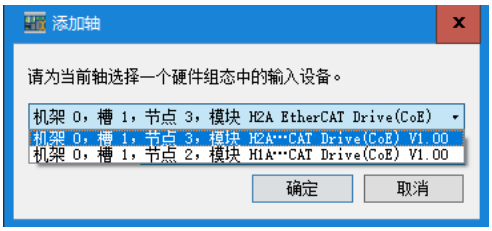


2）确定后可看到硬件组态界面的“EtherCAT BUS”，点击右边“EtherCAT”，选择从站并将从站拖到“EtherCAT BUS”下。




3）完成以上配置后在“工具”选项中选择“轴向导配置”，在轴向导配置界面双击“通信轴”即可添加通信轴，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。



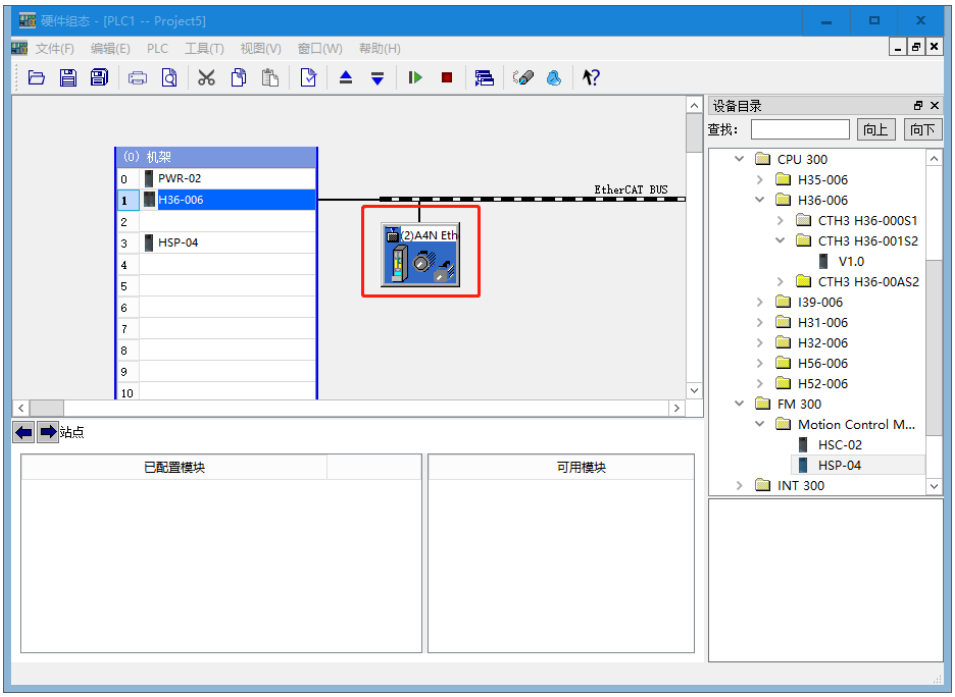

**注意：**一个从站只可对应添加一个通信轴。

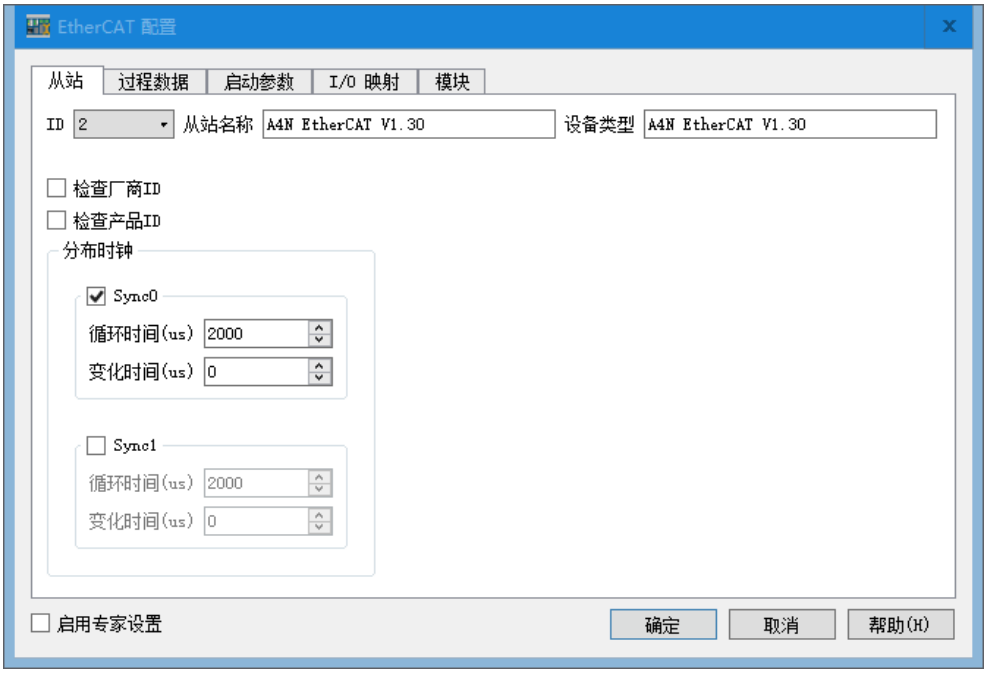
4）添加的通信轴界面如下所示：



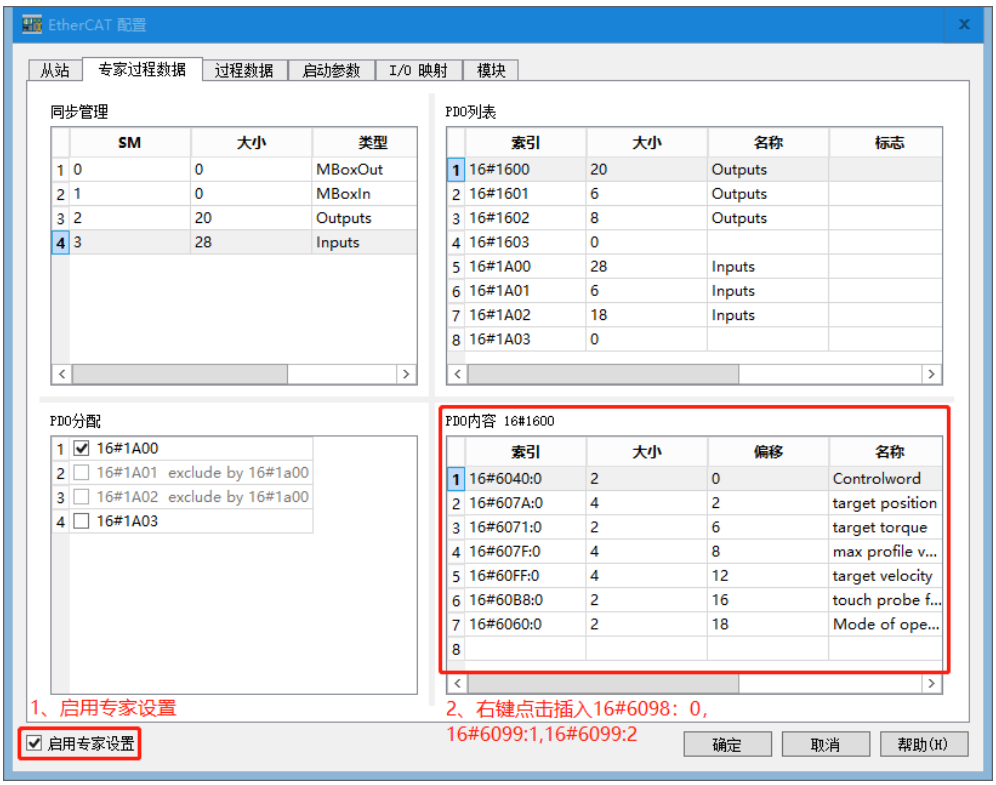

5）若需要使用驱动器回原（MC_Home 指令），在配置通信轴回原功能时，需配置回原模式 16#6098:0 以及回原搜索速度 16#6099:1，回原爬行速度 16#6099:2。操作如下：

双击挂接在“EtherCAT BUS”下的从站。出现 EtherCAT 配置界面。



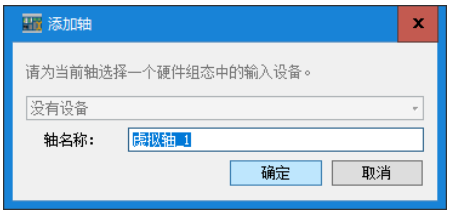



6）点击“启用专家设置”，右键点击后插入参数 16#6098:0（回原模式），16#6099:1（回原搜索速度），16#6099:2（回原爬行速度）。

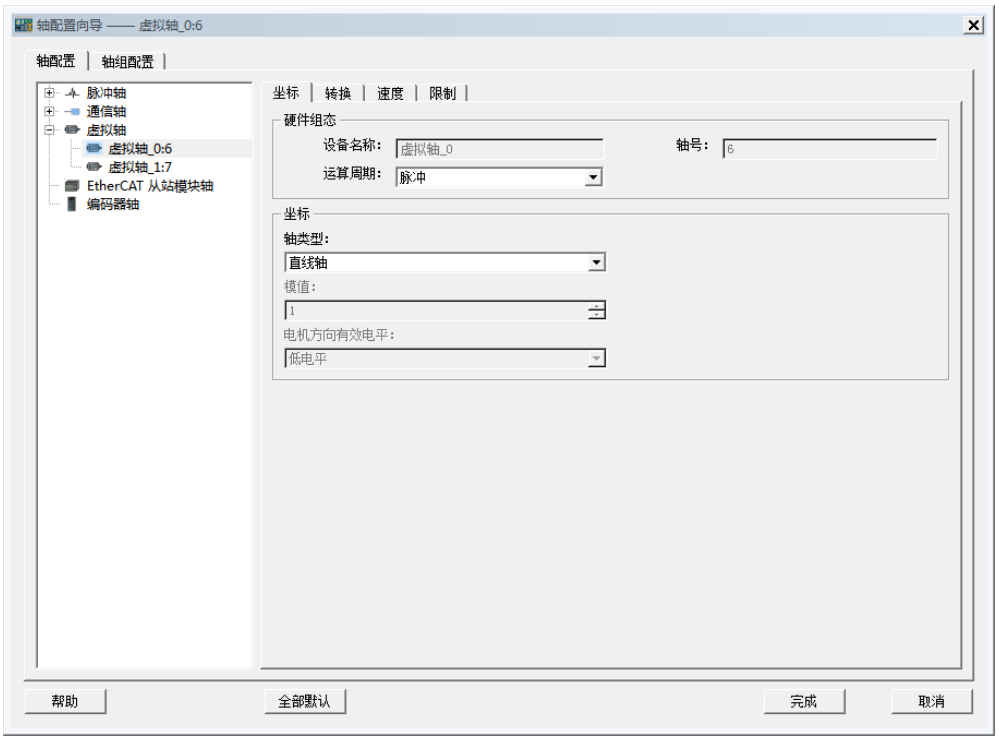



## 3、虚拟轴配置

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。

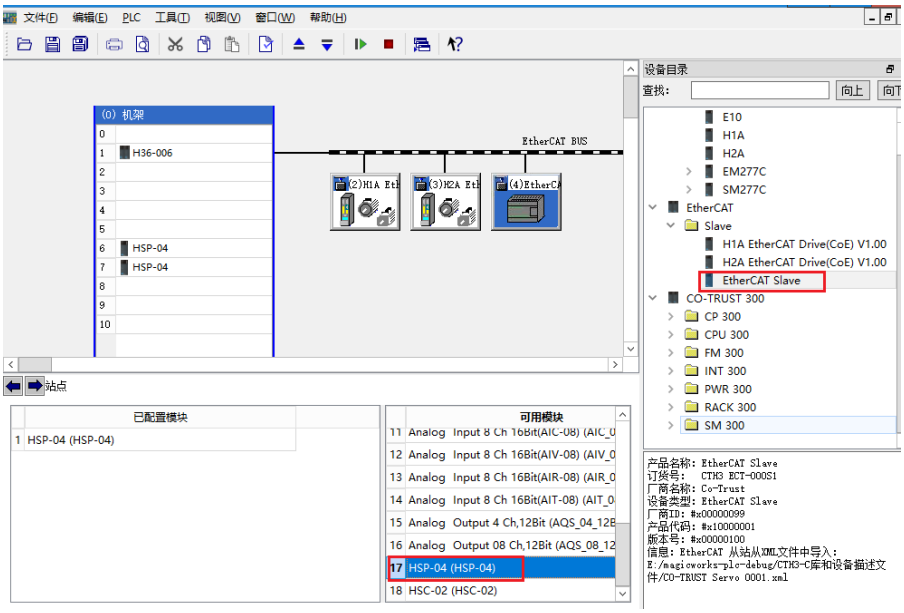



添加的虚拟轴界面如下所示：

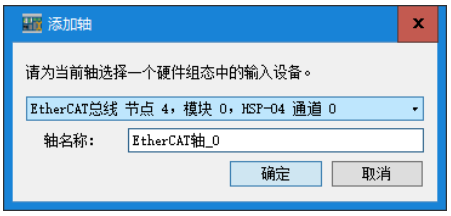



## 4、EtherCAT 从站模块轴配置

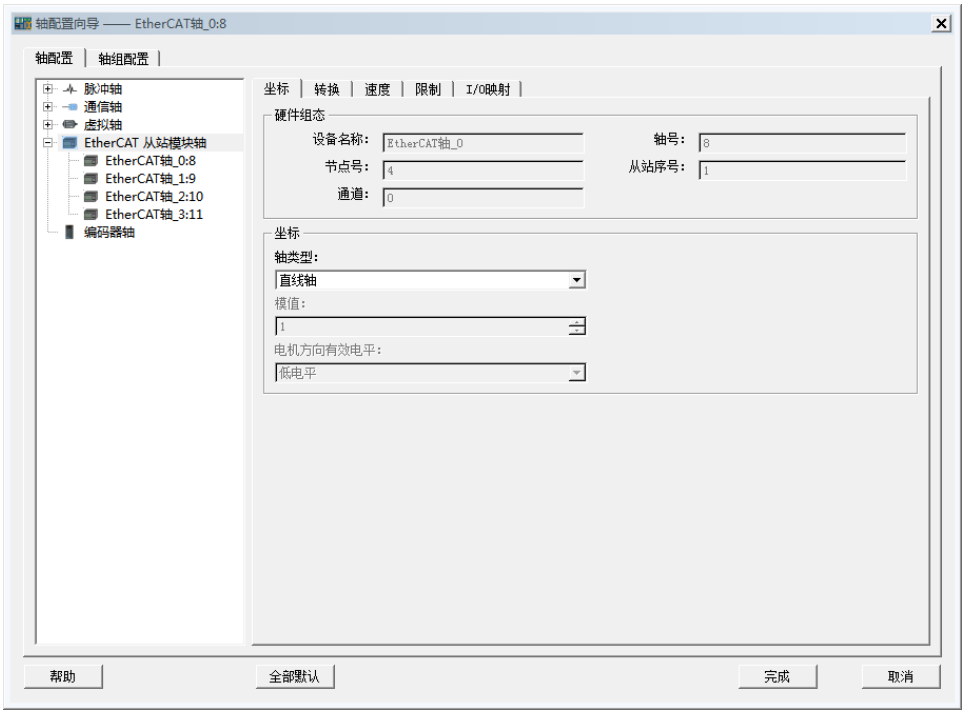
添加 EtherCAT 从站模块轴前，需先在硬件组态界面添加 EtherCAT 从站，并为所添加的 EtherCAT 从站选择配置 HSP 模块。如下图所示：



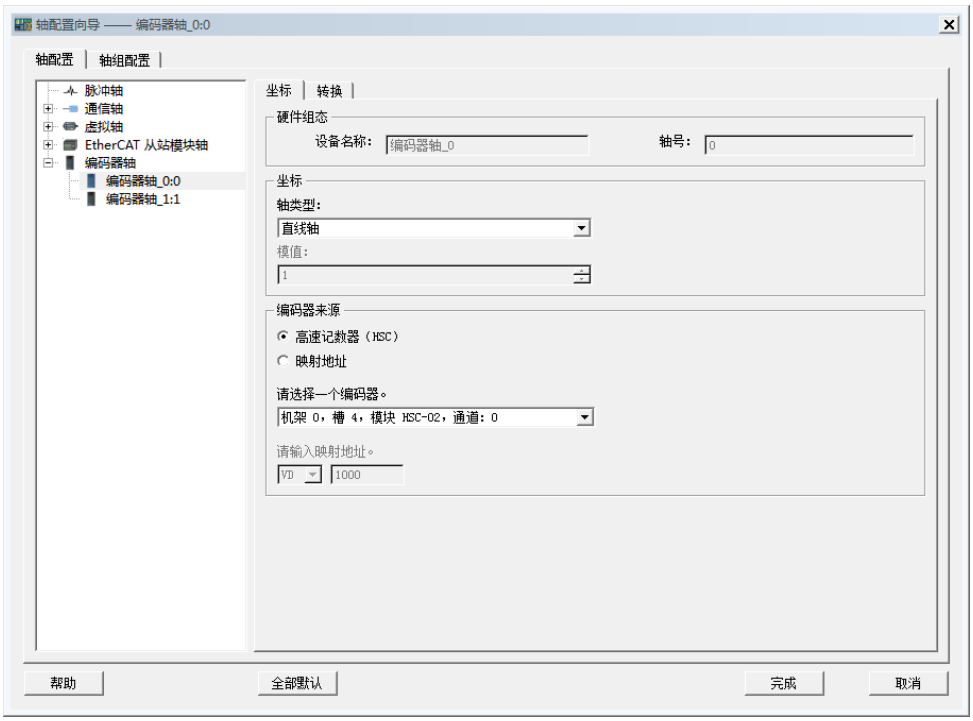

在“轴配置向导”界面右键点击“EtherCAT 轴”选择“添加轴”，在弹出的以下界面可进行轴配置。配置完点击“确定”完成添加。




添加的 EtherCAT 从站模块轴界面如下所示：



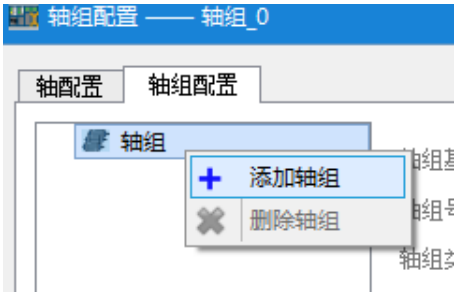

## 5、编码器轴配置



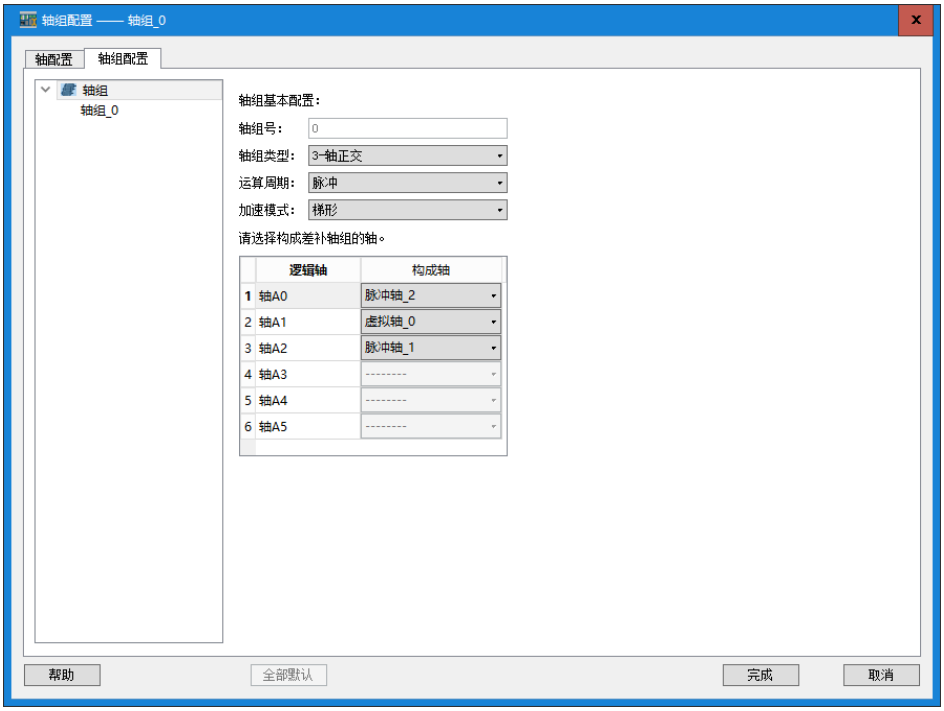

在上述界面选中“编码器轴”，右键可选择添加新的编码器轴。在“编码器来源”选项下为该轴选择对应编码器，可选择输入相应映射地址（默认 V 区）确定编码器。

### 13.1.2 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：

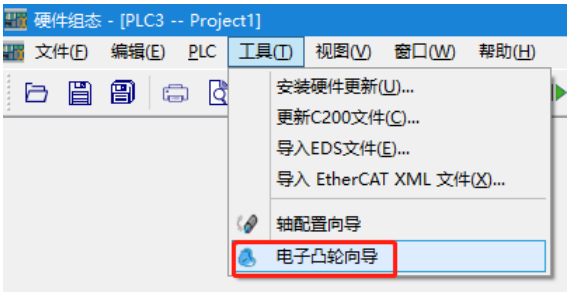



- “轴组基本配置”：“轴组号”不可更改，“轴组类型”下拉选项可选择“2-轴正交”或“3-轴正交”，分别对应可选两组或三组“构成轴”；“运算周期”可选“脉冲”或“EtherCAT”；“加速模式”可选“梯形”，“sin*2”和“二次”。
- “逻辑轴”：默认六组，不可更改。
- “构成轴”：“2-轴正交”对应两个构成轴，“3-轴正交”对应三个构成轴，下拉选项可选择“脉冲轴”或“虚拟轴”。

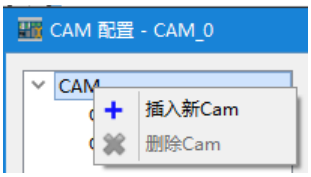
## 13.2 电子凸轮向导

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。

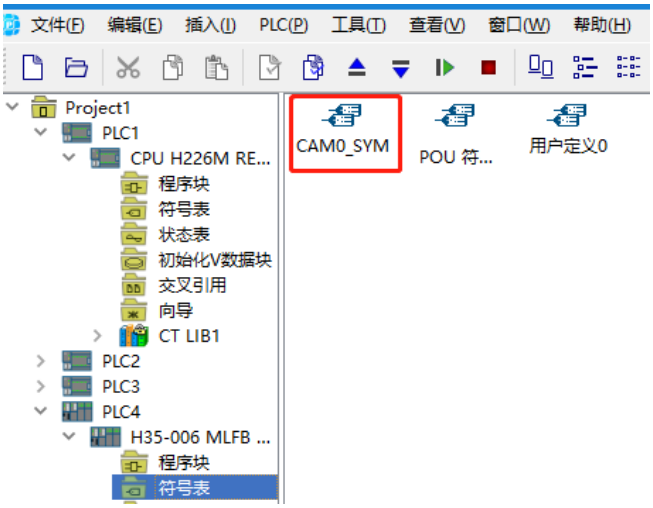
1、在硬件组态界面选择“工具”->“电子凸轮向导”即可进行电子凸轮配置。



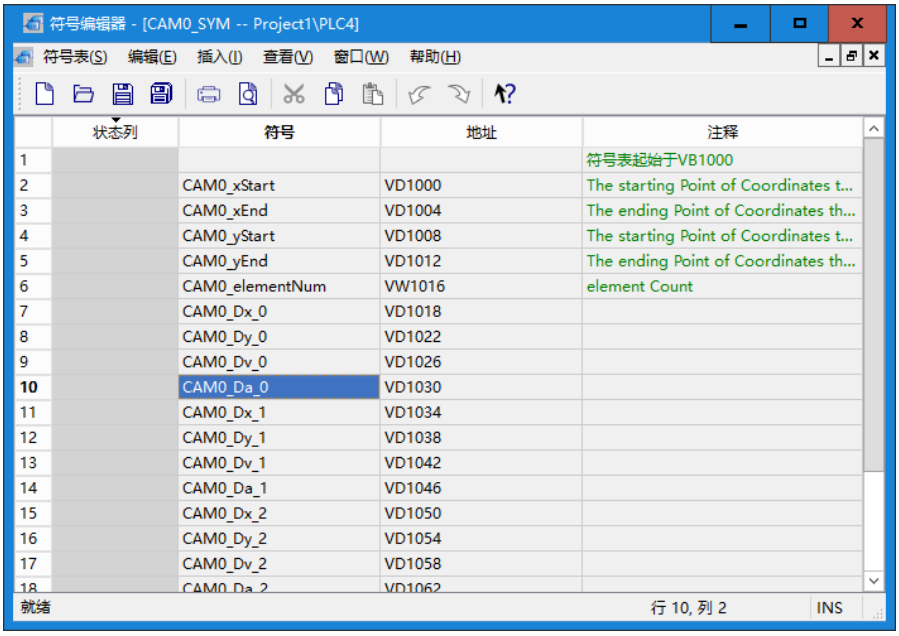

2、电子凸轮向导配置界面如下，选中“CAM”右键可插入新的 CAM：



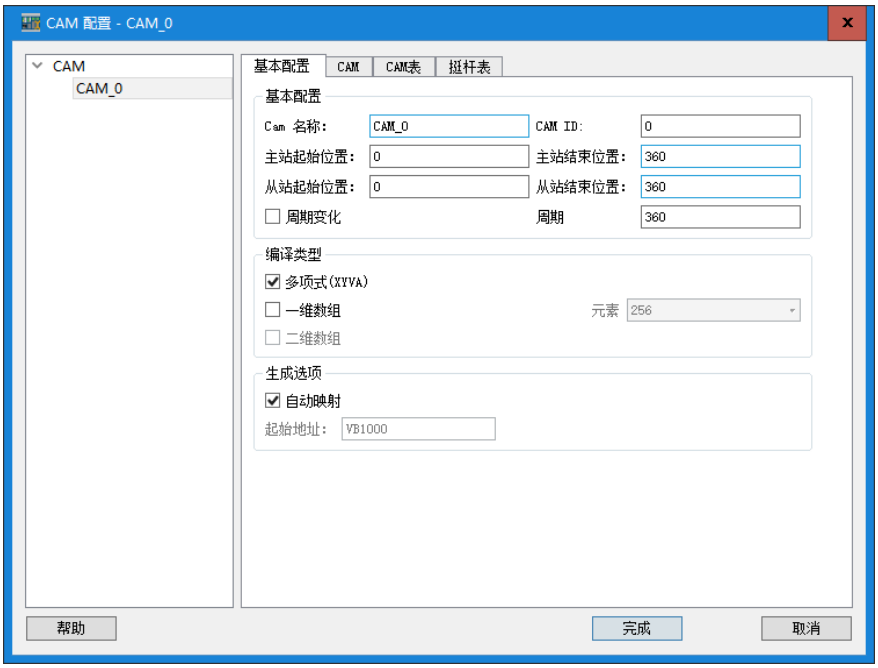

成功插入 CAM 后，在项目管理器界面选择“符号表”，可看到新增的 CAM。




3、点击“CAM0_SYM”出现以下界面，可查看对应 CAM 的符号地址。

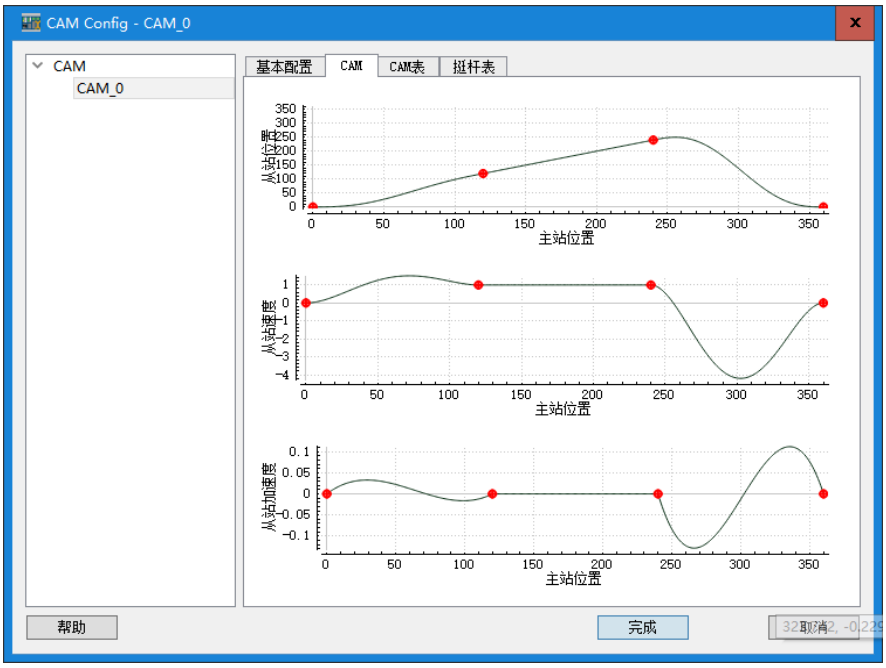



4、插入 CAM 后，配置界面如下所示。



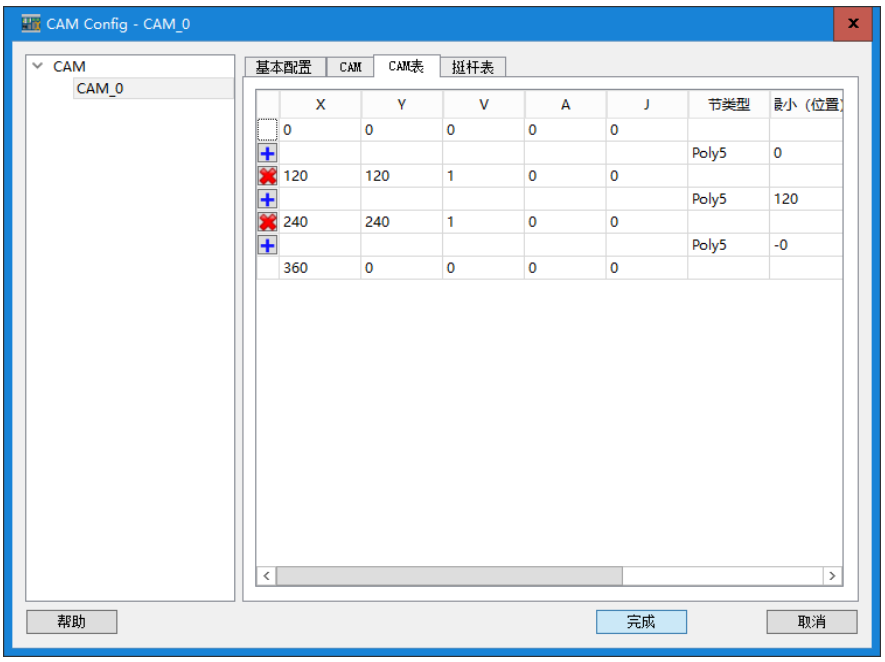

**【基本配置】**

- “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，目前已支持“一维数组”和“二维数组”的编译类型。
- “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。



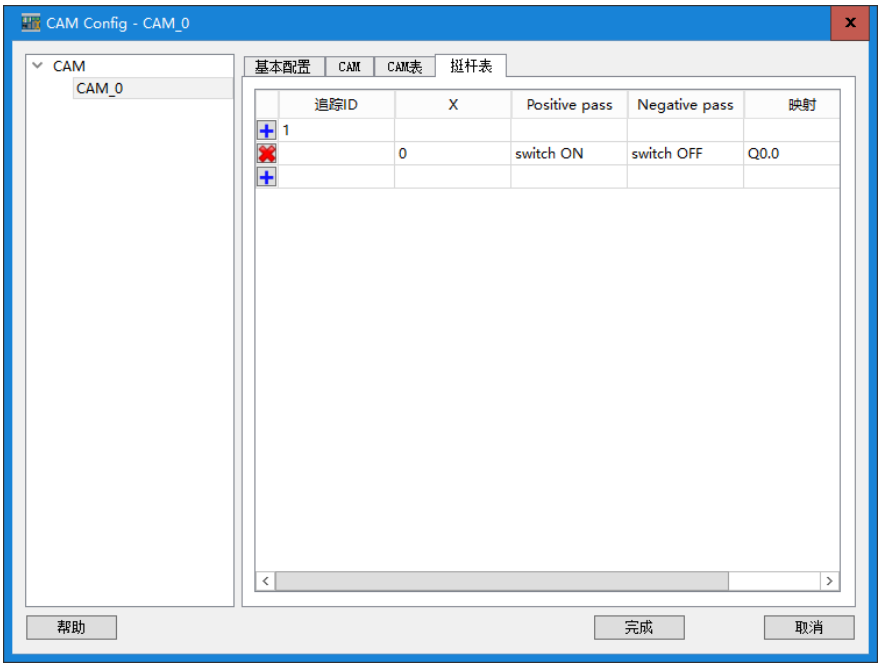

**【CAM】**

此页面以曲线图的形式分别显示“主站位置”与“从站位置”、“从站速率”以及“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“”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。

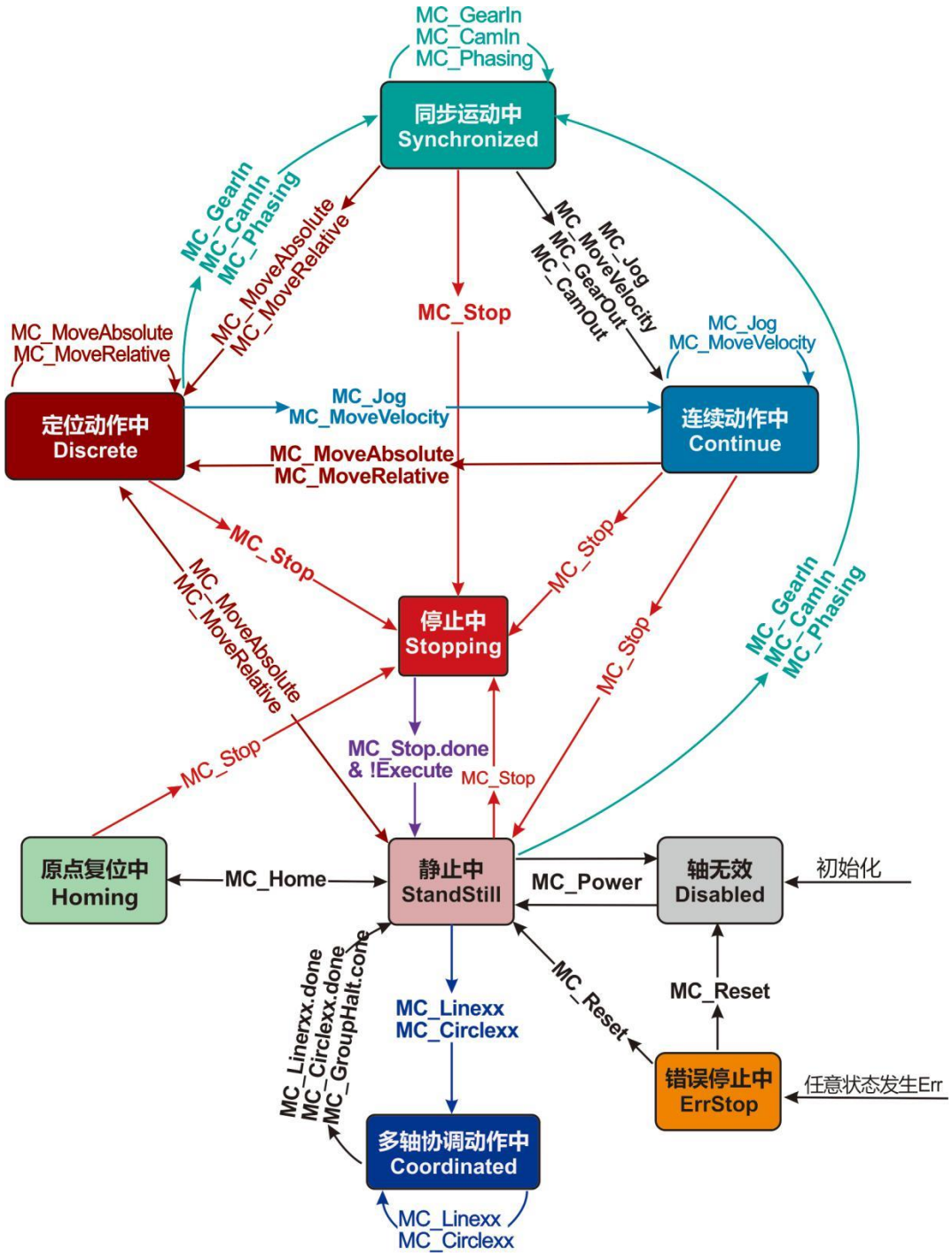



**【挺杆表】**

功能保留，PLC 暂不支持此功能。

## 13.3 轴指令“ct_plcopen_lib(v2.1)”介绍

Axis 指令依据 PLCOpen 标准设计，轴状态指令流程图如下图所示：



**提示**

1、脉冲轴不能与 ct_hsp300_lib 库里 MC_PTP_R、MC_PTP_A、MC_SPEED_CTRL 同时使用。

2、MagicWorks PLC(V2.19 及以上版本)支持 ct_plcopen_lib(v2.1)指令库。

ct_plcopen_lib(v2.1)指令分三类，分别是单轴指令、多轴指令和轴组指令，以下各章节将介绍这三类轴指令。

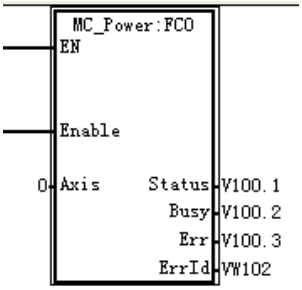
### 13.3.1 单轴控制指令

单轴控制指令一般用于定位控制，即伺服电机拖动外部机构运动到指定位置。
单轴控制常用到的 MC 功能块如下：

| 函数名 | 指令功能 | 备注 |
|---|---|---|
| MC_Power | 轴使能指令 | |
| MC_MoveAbsolute | 绝对位移指令 | |
| MC_MoveRelative | 相对位移指令 | |
| MC_MoveVelocity | 速度指令 | |
| MC_Stop | 停止指令 | |
| MC_Halt | 可打断的停止指令 | |
| MC_Home | 原点回归指令 | 驱动器回原方式 |
| SMC_Home | 特殊原点回归指令 | 控制器回原方式 |
| MC_Reset | 复位错误指令 | |
| MC_MoveFeed | 中断事件触发运动指令 | |
| MC_MoveJog | 点动指令 | |
| MC_SetPosition | 设置当前位置指令 | |
| SMC_CtrlByActPos | 位置跟随指令 | |
| MC_ReadStatus | 读取轴状态指令 | |
| MC_ReadSetPos | 读取轴设定位置指令 | |
| MC_WriteParameter | 写参数指令 | |
| MC_ReadParameter | 读取参数指令 | |

***轴使能指令***

函数名：MC_Power



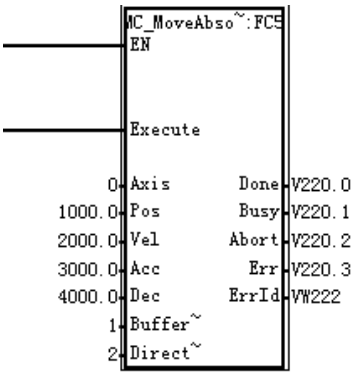
功能：此指令用于对相应的轴使能或解除使能。

参数说明：

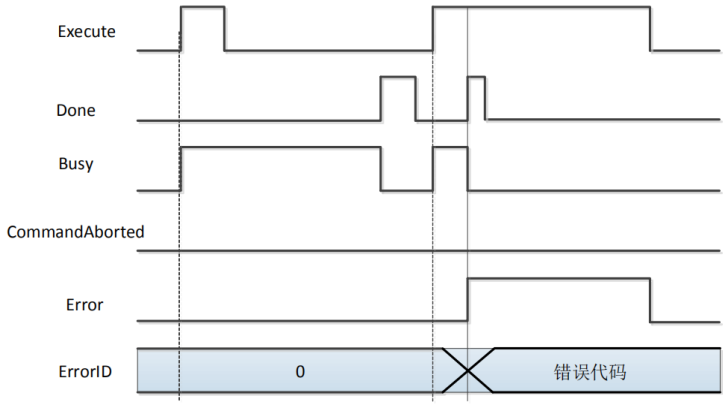
| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Enable | IN | 轴使能 | BOOL | 设为TRUE则进入可运行状态，设为FALSE则解除可运行状态。 |
| Axis | IN | 轴ID号 | BYTE | |
| Status | OUT | 状态位 | BOOL | 进入可运行状态时变为 TRUE。<br>进入不可运行状态变为 FALSE。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件由On变Off时，Error位被复位。 |
| ErrId | OUT | 错误码 | WORD | 错误码，详见章节13.3.4 错误代码。 |

## *绝对位移指令*

函数名：MC_MoveAbsolute




功能：此指令用于控制终端执行机构按照给定速度，加减速移动到相对于零点的目标位置，此指令在运行过程中一旦被指令终止，剩余未完成的距离将被丢弃，执行新的指令功能。



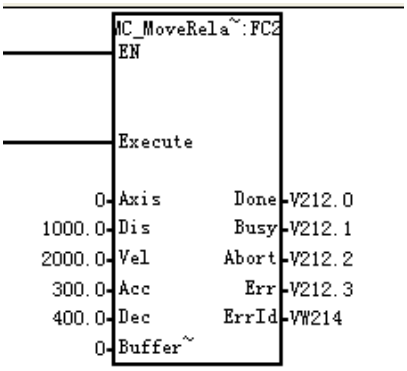

参数说明：

| 参数名 | 输入输 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|

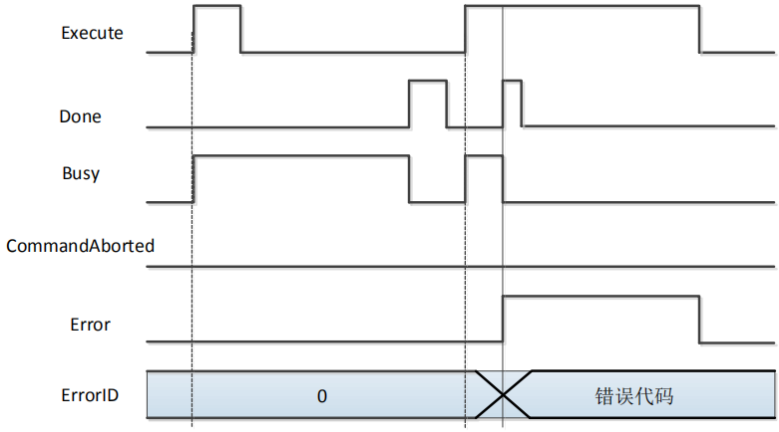
| | 出属性 | | | |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Pos | IN | 位置 | REAL | 终端执行机构相对零点的目标位置，单位：单元 |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 定义轴的动作：<br>0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Direction | IN | 方向 | INT | 运转方向：<br>0：距离最短的方向；<br>1：正方向；<br>-1：反方向；<br>2：延续当前的方向；<br>该参数针对旋转轴时生效，对直线轴无效。 |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位置位；<br>当指令的执行条件Off时，Done位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 指令执行过程中被终止时，Abort位置位；<br>当指令的执行条件Off时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *相对位移指令*

函数名：MC_MoveRelative




功能：此指令用于控制终端执行机构以当前位置为参考点，按照给定的速度，加减速移动给定的距离，此指令在运行过程中一旦被终止，剩余未完成的距离将被丢弃，执行新的指令功能。



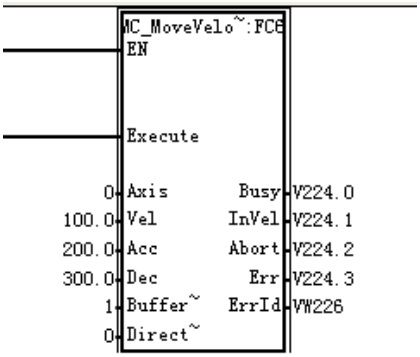

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Dis | IN | 距离 | REAL | 以当前位置为参考，终端执行机构要移动的目标距离，该值设置为负数代表轴将会反转。<br>单位：单元。 |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。<br>（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。<br>（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。<br>（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |

| | | | | |
|---|---|---|---|---|
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *速度指令*

函数名：MC_MoveVelocity




功能：此指令用于控制终端执行机构按给定的加减速运动至给定速度，并匀速运行。当终端机构到达给定速度后，此指令执行完成，但终端机构仍以此速度继续运行。

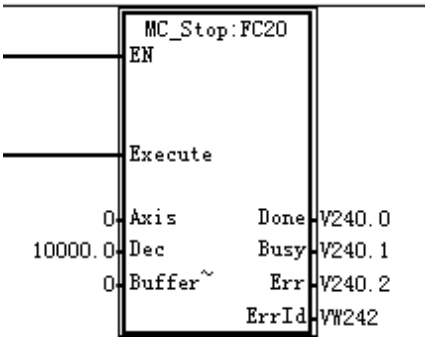
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Direction | IN | 方向 | INT | 1：正方向<br>-1：负方向 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |

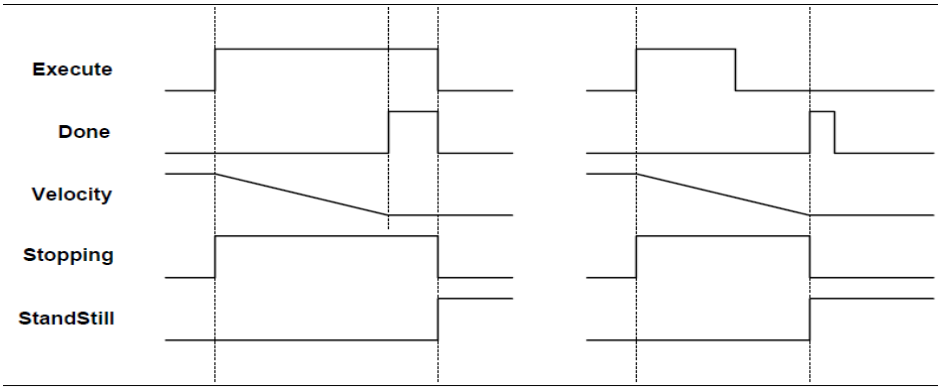
| | | | | |
|---|---|---|---|---|
| Invel | OUT | 速度到达位 | BOOL | 到达目标速度后，Invelocity位被置位；当指令的<br>执行条件由On变Off 时，Invelocity位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *停止指令*

函数名：MC_Stop




功能：此指令用于控制终端执行机构按给定的减速度减速，直到停止。此指令运行时，不能被其它任何指令终止。速度降为 0 后，轴变 Stopping 状态，直到 Execute 变为 0，才会切到 StandStill（静止状态）。



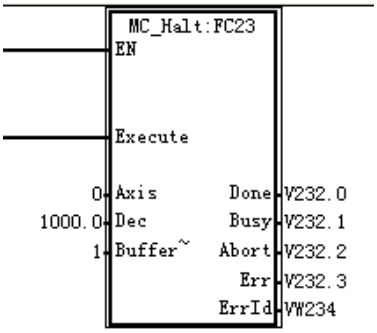

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。 |

| | | | | |
|---|---|---|---|---|
| | | | | （单位：单元/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Done | OUT | 完成位 | BOOL | 速度降为0后，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

***可打断的停止指令***

函数名：MC_Halt




功能：此指令用于控制终端执行机构按给定的减速度减速，直到停止。此指令运行时，能被其它指令终止。

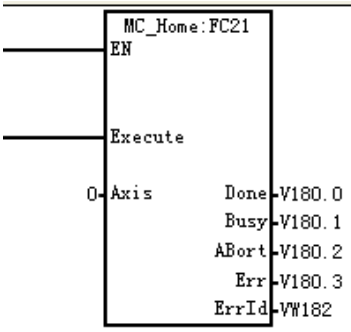
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。<br>（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Done | OUT | 完成位 | BOOL | 速度降为0后，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位； |

| | | | | |
|---|---|---|---|---|
| | | | | 当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *原点回归指令*

函数名：MC_Home




功能：此指令用于控制轴按轴参数给定的模式和速度执行回原点动作，回原点模式和速度在轴参数设置界面中设定。

**注意：**脉冲轴与通信轴的配置有所不同

• 脉冲轴是在参数设置界面设定回原模式和回原速度，详情见章节 13.1.1 轴配置 脉冲轴配置。

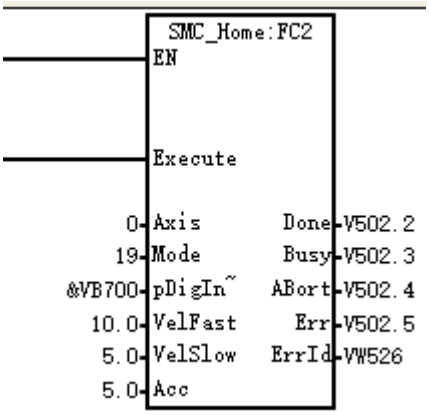
• 通信轴，则是要改写回原模式 16#6098:0 的值以及配置回原速度 16#6099:0，16#6099:1；具体操作见章节 13.1.1 轴配置通信轴配置。

参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Done | OUT | 完成位 | BOOL | 速度降为0后，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *特殊原点回归指令*

函数名：SMC_Home




功能：此指令由控制器控制轴按参数给定的模式和速度执行回原点动作，回原模式和速度在指令中设定；注意，一般情况下，轴的原点开关/限位信号是接至 PLC 系统。回原完成后，控制器会偏置轴的坐标为 0.0，轴本身的 16#6064:0(position actual value)不会因为操作指令而发生变化。

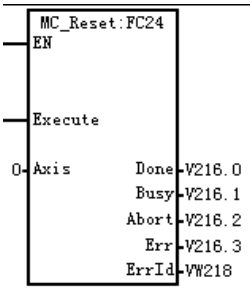
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Mode | IN | 回零模式 | BYTE | 参见脉冲轴回零13.4 回原功能介绍。 |
| pDigInput | IN | 输入信号指针 | 指针 | 原点开关，正向，反向限位开关位号指向的内存地址。<br>内存里<br>bit0: 负向开关<br>bit1: 正向开关<br>bit2: 原点开关<br>开关信号1为有效，0为无效。<br>例如，pDigInput等于 &IB0时，<br>I0.0：负向开关<br>I0.1：正向开关<br>I0.2：原点开关。 |
| VelFast | IN | 快速回零速度 | REAL | 查找原点时的快速速度，此参数总为正。（单位：单元/秒） |
| VelSlow | IN | 慢速回零速度 | REAL | 查找原点时的慢速速度，此参数总为正。（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。（单位：单元/秒/秒） |
| Done | OUT | 完成位 | BOOL | 回零完成后，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |

| | | | | |
|---|---|---|---|---|
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见13.3.4 错误代码 |

### *复位错误指令*

函数名：MC_Reset



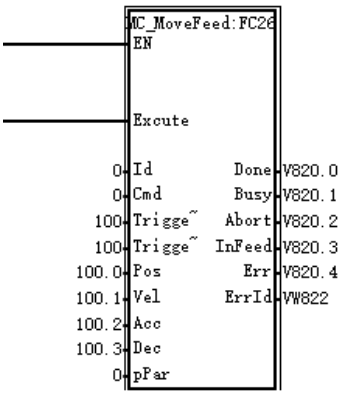

功能：用于解除轴的异常，解除异常后，轴状态将从 ErrorStop 状态转换到 StandStill 状态。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Done | OUT | 完成位 | BOOL | 当轴状态被复位至准备执行状态后，Done 位被置位；<br>当指令的执行条件由 On 变 Off 时，Done 位被复位 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *中断事件触发运动指令*

函数名：MC_MoveFeed




功能：指定输入中断事件发生后，轴进行 Cmd 指定的动作。在指定中断发生之前，Execute 变为 0，可取消当前指令。运动完成后，需要再次触发才能再次响应中断。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Id | | 轴或轴组Id | BYTE | 见下表2 |
| CMD | IN | 发生的动作 | BYTE | 0：停止。<br>1：相对运动。<br>2：绝对运动。<br>3：速度控制。<br>16#80：轴组停止 |
| TriggerInput | IN | 中断事件 | BYTE | 见下附表1 |
| TriggerVar | IN | 中断事件参数 | DWORD | |
| Pos | IN | 位置 | REAL | 见下附表2 |
| Vel | IN | 速度 | REAL | |
| Acc | IN | 加速度 | REAL | |
| Dec | IN | 减速度 | REAL | |
| pPar | IN | 其它参数指针 | DWORD | 预留。当上述参数不足描述参数时，将参数放置于pPar指示的内存里面。 |
| Done | IN | 完成位 | BOOL | 当轴状态被复位至准备执行状态后，Done 位被置位；<br>当指令执行条件由 On 变 Off 时，Done 位复位 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| InFeed | OUT | 运动进行位 | BOOL | 当中断发生,并触发运动时，InFeed被置位。<br>当指令完成进InFeed被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

附表 1 中断触发事件

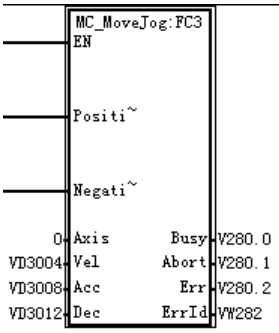
| TriggerInput | 意义 | TriggerVar |
|---|---|---|
| 0 | 上升沿，I0.0 | 发生中断事件的模块<br>TriggerVar=机架号*256 + 槽号<br>如模块放置于机架号为 1，槽号为 3 的位置，则 TriggerVar =16#0103 |
| 2 | 上升沿，I0.1 | |
| 4 | 上升沿，I0.2 | |
| 6 | 上升沿，I0.3 | |
| 1 | 下降沿，I0.0 | |
| 3 | 下降沿，I0.1 | |
| 5 | 下降沿，I0.2 | |
| 7 | 下降沿，I0.3 | |
| 12 | HSC0 CV=PV | |
| 27 | HSC0 方向改变 | |
| 28 | HSC0 外部复原 / Zphase | |
| 13 | HSC1CV=PV | |
| 14 | HSC1 方向改变 | |
| 15 | HSC1 外部复原 / Zphase | |
| 128 | V 内存上升沿中断 | TriggerVar =内存偏移地址* 8 + 内存的位。<br>如 V0.0 此值为 0；V127.7 此值为 1023（127*8+7）。 |
| 129 | V 内存下降沿中断 | |

附表 2 轴运动

| CMD | 参数 | 详细描述 |
|---|---|---|
| 0：停止。 | Id：轴 ID 号<br>Dec：减速度 | 相当于中断触发后执行 MC_Stop 指令。 |
| 1：相对运动。 | Id：轴 ID 号<br>Vel：速度<br>Acc：加速度<br>Dec：减速度<br>Pos：运动的距离 | 相当于中断触发后执行 MC_MoveRelative |
| 2：绝对运动。 | Id：轴 ID 号<br>Vel：速度<br>Acc：加速度<br>Dec：减速度<br>Pos：位置<br>当轴为旋转轴时：<br>VEL > 0 方向为正向旋转<br>VEL < 0 方向为反向旋转 | 相当于中断触发后执行 MC_MoveAbsolute |
| 3：速度运动。 | Id：轴 ID 号<br>Vel：速度<br>Acc：加速度<br>Dec：减速度<br>Vel > 0.0 正向<br>Vel < 0.0 反向 | 相当于中断触发后执行 MC_MoveVelocity |
| 16#80：轴组停止 | Id：轴组 ID 号<br>Dec：减速度 | 相当于中断触发后执行 MC_GroupHalt |

## *点动指令*

函数名：MC_MoveJog



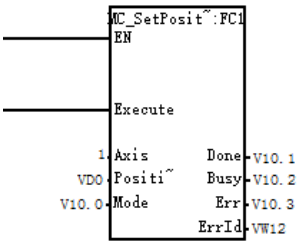

功能：根据指定 Velocity(目标速度)执行微动移动。需要正方向微动移动时，将 PositiveEnable(正方向有效)设为 TRUE，需要负方向微动移动时，将 NegativeEnable(负方向有效)设为 TRUE。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| PositiveEnable | IN | 正方向有效 | BOOL | 设为TRUE，则开始正方向移动。设为FALSE，则结束移动。 |
| NegativeEnable | IN | 负方向有效 | BOOL | 设为 TRUE，则开始负方向移动。设为 FALSE，则结束移动。 |
| Axis | IN | 轴ID号 | DWORD | 轴号 |
| Vel | IN | 目标速度 | REAL | 指定目标速度。单位为［指令单位/秒］。 |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。（单位：单元/秒/秒）<br>当设为0时，实现紧急停止 |
| Busy | OUT | 忙位 | BOOL | 指令正在执行为TRUE，执行完成为FLASE |
| Abort | OUT | 中止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *设置当前位置指令*

函数名：MC_SetPosition




功能：此指令设定当前坐标的值(单元)。

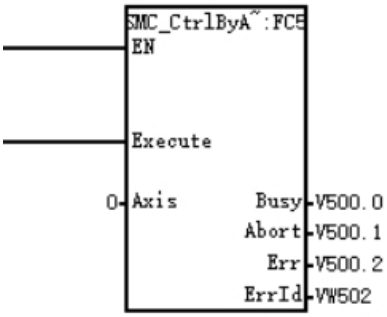
**注意：本指令不能在轴运动时使用，当轴作为同步运动的主轴，也不可以使用，不然可能发生不可预知的后果。**

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | 轴号 |
| Position | IN | 指定的坐标位置 | REAL | 指定的坐标位置（单元）<br>取值-1E+13<=Position<=1E+13 |
| Mode | IN | 位置模式 | BOOL | 0：绝对位置模式.指定为此模式，则轴的绝对位置为Position<br>1：相对位置模式.指定为此模式，则轴的绝对位置为当前位置加Position |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位；<br>当指令的执行条件 Off 时，Done 位被复位。 |
| Busy | OUT | 忙位 | BOOL | 指令正在执行为TRUE，执行完成为FLASE |
| Error | OUT | 错误位 | OBOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件由On变Off时，Error位被复位。 |
| ErrId | OUT | 错误码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *位置跟随指令*

函数名：SMC_CtrlByActPos




功能：调用此指令后，轴位置将由外部指令控制，轴设定位置将跟随当前位置。

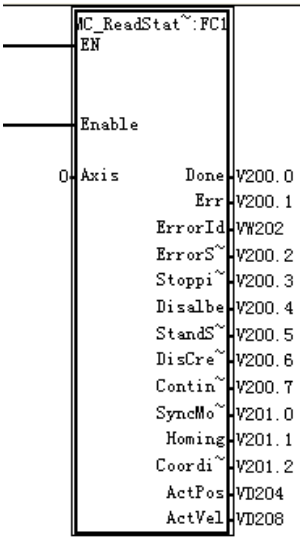
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由 Off 变 On 时，执行该指令。<br>当执行条件由 ON 变 Off 时，取消当前指令 |
| Axis | IN | 主轴 ID 号 | BYTE | 轴号 |
| Busy | OUT | 指令处理中 | BOOL | True，指令正在处理中 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort 位被置位；<br>当指令的执行条件 Off 时，Abort 位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error 位被置位； |

| | | | | |
|---|---|---|---|---|
| | | | | 当指令的执行条件 Off 时，Error 位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节 13.3.4 错误代码。 |

## *读取轴状态指令*

函数名：MC_ReadStatus




功能：此指令用于读取相应轴状态参数的值。

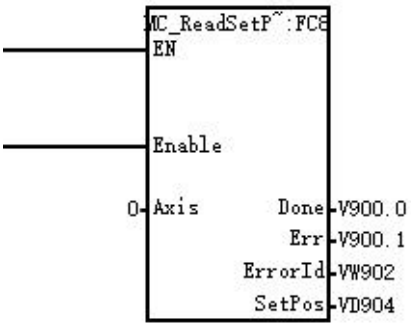
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| enable | IN | 指令执行条件 | BOOL | 当为On时，执行该指令。 |
| Axis | IN | 轴Id号 | BYTE | 轴号 |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位；<br>当指令的执行条件 Off 时，Done 位被复位。 |
| Error | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件由On变Off时，Error位被复位。 |
| ErrId | OUT | 错误码 | WORD | 错误码<br>当Error为TRUE时，ErrId为指令的错误。<br>当ErrStop为TRUE时，ErrId为轴当前错误。 |
| ErrStop | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态ErrStop |
| Stopping | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态Stopping |
| Disabled | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态Disabled |
| StandStill | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态StandStill |
| DisCreteMotion | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态DisCreteMotion |
| Continous Motion | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态ContinousMotion |
| SyncMotion | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态SyncMotion |

| | | | | |
|---|---|---|---|---|
| Homing | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态Homing |
| Coordinate | OUT | 状态指示位 | BOOL | TRUE，如果轴位于状态Coordinate |
| ActPos | OUT | 当前坐标 | REAL | 单元 |
| ActVel | OUT | 当前速度 | REAL | 单元/秒 |

### *读取轴设定位置指令*

函数名：MC_ReadSetPos



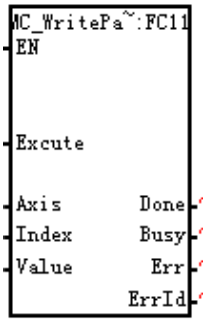

功能：此指令用于读取轴的当前设定位置。

参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| enable | IN | 指令执行条件 | BOOL | 当为 On 时，执行该指令。 |
| Axis | IN | 轴 ID 号 | BYTE | 轴号 |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位；<br>当指令的执行条件 Off 时，Done 位被复位。 |
| Error | OUT | 错误位 | BOOL | 如果检测到有错误，Error 位被置位；<br>当指令的执行条件由 On 变 Off 时，Error 位被复位。 |
| ErrId | OUT | 错误码 | WORD | 当 Error 为 TRUE 时，ErrId 为指令的错误。<br>当 ErrStop 为 TRUE 时，ErrId 为轴当前错误。 |
| Setpos | OUT | 当前设定位置 | REAL | 单元 |

### *写参数指令*

函数名：MC_WriteParameter

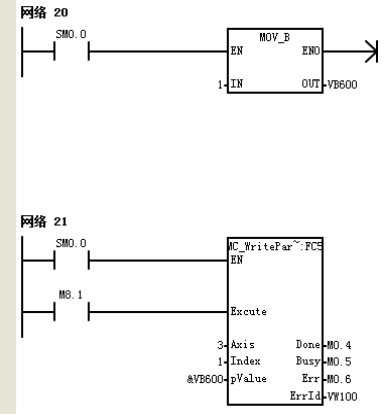



功能：此指令用于写入相应的轴的配置参数的值，写入成功后，相当于修改了硬件组态，永久有效。只能在轴未使能的情况下运行该指令。

参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Index | IN | 参数的序号 | WORD | 见下表**参数描述表** |
| pValue | IN | 参数值的指针 | DWORD | 参数指针 |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位；当指令的执行条件 Off 时，Done 位被复位。 |
| Error | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；当指令的执行条件由On变Off时，Error位被复位。 |
| ErrId | OUT | | | 错误码 |

**参数描述表**

| Index | 参数 | pValue | 说明 |
|---|---|---|---|
| 0 | 量纲转换 | pValue 为指向量纲描述的指针。量纲描述见下附表 3。 | |
| 1 | 加减速方式 | pValue：为指向加减速方式的指针。加减速方式为一个 BYTE<br>加速模式，0：梯形；1：S 形；2：Sin 加减速 | 轴 3 加减速方式为 S 形加减速<br> |
| 2 | 速度 | pValue：为指向速度设定的指针。指向结构见附表 4 | |

附表 3 纲量转换数据结构说明

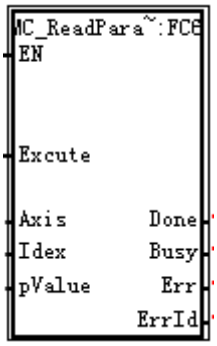
| 名称 | 类型 | 说明 |
|---|---|---|
| ScalPulseInc | REAL | 量纲转换，脉冲增量<br>（硬件组态量纲转换左 1） |
| ScalMotorTurns | REAL | 量纲转换，电机圈数<br>（硬件组态量纲转换右 1） |
| ScalMotorTurnsInc | REAL | 量纲转换，电机圈数增量<br>（硬件组态量纲转换左 2） |
| ScalGearTurnsInc | REAL | 量纲转换，齿轮圈数增量<br>（硬件组态量纲转换右 2） |
| ScalGearTurnsInc | REAL | 量纲转换，齿轮圈数增量<br>（硬件组态量纲转换左 3） |
| ScalAppUnitsc | REAL | 量纲转换，工程单位<br>（硬件组态量纲转换左 3） |

附表 4 速度设定说明

| 名称 | 类型 | 说明 |
|---|---|---|
| VelMax | REAL | 最大速度 |
| VelMin | REAL | 最小速度 |
| VelStartStop | REAL | 启动停止速度 |
| AccMax | REAL | 最大加速度 |
| AccMin | REAL | 最小加速度 |

## *读取参数指令*

函数名：MC_ReadParameter




功能：此指令用于读取相应的轴的配置参数的值。

参数说明

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Index | IN | 参数的序号 | WORD | 见参数描述表 |

| | | | | |
|---|---|---|---|---|
| pValue | OUT | 参数指针 | DWORD | |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位；当指令的执行条件 Off 时，Done 位被复位。 |
| Error | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；当指令的执行条件由On变Off时，Error位被复位。 |
| ErrId | OUT | 错误码 | | 错误码 |

### 13.3.2 同步控制指令

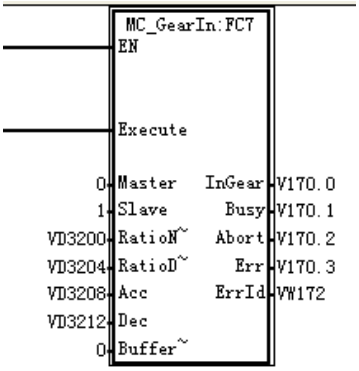
同步控制一般用于电子齿轮、电子凸轮、相位偏移指令等。

同步控制指令常用到的 MC 功能块如下：

| 函数名 | 指令功能 | 备注 |
|---|---|---|
| MC_GearIn | 电子齿轮耦合指令 | |
| MC_GearOut | 电子齿轮脱离指令 | |
| MC_CamTableSelect | 凸轮表选择指令 | |
| MC_CamIn | 电子凸轮关联指令 | |
| MC_CamOut | 解除电子凸轮关联指令 | |
| MC_Phasing | 相位偏移指令 | |
| MC_MoveSuperImposed | 叠加相对位移指令 | |

#### *电子齿轮耦合指令*

函数名：MC_GearIn




功能：此指令用于建立主从轴间的齿轮关系。建立齿轮关系时，可设定齿轮比等参数。齿轮关系建立后从轴以给定的比例关系跟随主轴运动，实现主从轴同步控制。

同步完成之后从轴速度=主轴速度 * RatioNum/RatioDen。

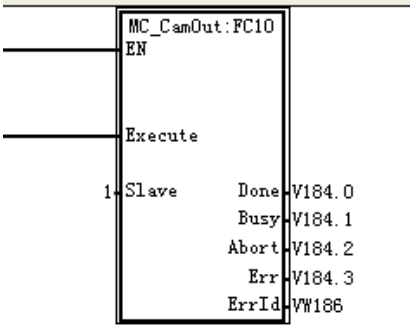
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off变On时，执行该指令。 |

| | | | | |
|---|---|---|---|---|
| Master | IN | 主轴ID号 | BYTE | |
| Slave | IN | 从轴ID号 | BYTE | |
| RatioNum | IN | 电子齿轮的分子 | REAL | 非0 |
| RetioDen | IN | 电子齿轮的分母 | REAL | 非0 |
| Acc | IN | 加速度 | REAL | |
| Dec | IN | 减速度 | REAL | |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| InGear | OUT | 耦合完成位 | BOOL | 耦合完成变InGear置位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *电子齿轮脱离指令*

函数名：MC_GearOut




功能：此指令用于解除主从轴间的齿轮关系，关系解除后，从轴会以脱离前的速度继续运行。

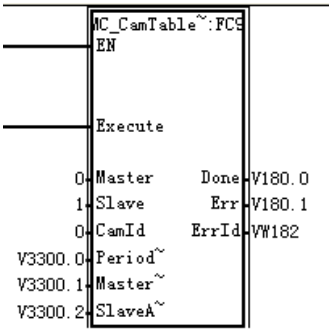
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Slave | IN | 从轴ID号 | BYTE | |
| Done | OUT | 完成位 | BOOL | 当取消齿轮关联指令执行成功时，Done位被置位。<br>当指令执行条件Off时，Done位复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位； |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |

| | | | | |
|---|---|---|---|---|
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

## *凸轮表选择指令*

函数名：MC_CamTableSelect




功能：此指令用于选择凸轮曲线，同时指定主从轴关联时的模式。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Master | IN | 主轴ID号 | BYTE | |
| Slave | IN | 从轴ID号 | BYTE | |
| CamId | IN | 凸轮表的ID号 | BYTE | |
| Periodic | IN | 周期 | BOOL | 此参数为1时，从轴会将周期性地执行凸轮运动。<br>此参数为0时，从轴只执行一个周期的凸轮运动。 |
| MasterAbsolute | IN | 主轴绝对位置模式 | BOOL | 当此参数为1时，主轴为绝对位置模式。<br>当此参数为0时，主轴为相对位置模式 |
| SlaveAbsolute | IN | 从轴绝对位置模式 | BOOL | 此参数为1时，从轴为绝对位置模式。<br>此参数为0时，从轴为相对位置模式。 |
| Done | OUT | 完成位 | BOOL | 当凸轮参数设置成功时，Done位被置位。<br>当指令执行条件Off时，Done位复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

在基本配置中指定了凸轮数据的起始地址，配置之后将生成数据块（功能暂时未实现），或指定（V 内存，符号表）起始地址。起始地址之后的一块内存存放 CAM 表数据。

当凸轮表选择多项式方式时，数据如下：

XYVZ 方式

| 序号 | 名称 | 类型 | 注释 |
|---|---|---|---|
| 1 | | STRUCT | |
| 2 | xStart | REAL | X 轴坐标起点 |

| | | | |
|---|---|---|---|
| 3 | xEnd | REAL | X 轴坐标终点 |
| 4 | yStart | REAL | Y 轴坐标起点 |
| 5 | yEnd | REAL | Y 轴坐标终点 |
| 6 | nElements | WORD | 元素个数 |
| 7 | dX0 | REAL | 第0个点 X 坐标 |
| 8 | dY0 | REAL | 第0个点 Y 坐标 |
| 9 | dV0 | REAL | 第0个点速度 |
| 10 | dA0 | REAL | 第0个点加速度 |
| 11 | dX1 | REAL | 第1个点 X 坐标 |
| 12 | dY1 | REAL | 第1个点 Y 坐标 |
| 13 | dV1 | REAL | 第1个点速度 |
| 14 | dA1 | REAL | 第1个点加速度 |
| 15 | dX2 | REAL | 第2个点 X 坐标 |
| | ... | ... | ... |
| | | END_STRUCT | |

**一维数组方式**

| 序号 | 名称 | 类型 | 注释 |
|---|---|---|---|
| 1 | | STRUCT | |
| 2 | xStart | REAL | X 轴坐标起点 |
| 3 | xEnd | REAL | X 轴坐标终点 |
| 4 | yStart | REAL | Y 轴坐标起点 |
| 5 | yEnd | REAL | Y 轴坐标终点 |
| 6 | nElements | WORD | 元素个数 |
| 7 | dY0 | REAL | 第0个点 Y 坐标 |
| 8 | dY1 | REAL | 第1个点 Y 坐标 |
| 9 | dY2 | REAL | 第2个点 Y 坐标 |
| | ... | ... | ... |
| | | END_STRUCT | |

**二维数组方式**

| 序号 | 名称 | 类型 | 注释 |
|---|---|---|---|
| 1 | | STRUCT | |
| 2 | xStart | REAL | X 轴坐标起点 |
| 3 | xEnd | REAL | X 轴坐标终点 |
| 4 | yStart | REAL | Y 轴坐标起点 |
| 5 | yEnd | REAL | Y 轴坐标终点 |
| 6 | nElements | WORD | 元素个数 |
| 8 | dX0 | REAL | 第0个点 X 坐标 |
| 12 | dY0 | REAL | 第0个点 Y 坐标 |
| 15 | dX1 | REAL | 第1个点 X 坐标 |
| | dY1 | REAL | 第1个点 Y 坐标 |
| | dX2 | REAL | 第2个点 X 坐标 |
| | dY2 | REAL | 第2个点 Y 坐标 |
| | ... | ... | ... |
| | | END_STRUCT | |

用户可以通过修改此内存里数据，从而改变凸轮表数据，修改数据后，调用 MC_CamTableSelect，通知 PLC 凸轮表数据改变，如果此时系统处于循环凸轮表运行之中，改变后的数据将在下一个凸轮周期生效，需要立即生效则在调用 MC_CamTableSelect 后，立即调用 MC_CamIn 指令。

**凸轮偏移 Offset 和缩放比例 Scaling：**

主轴输入转换的位置是根据以下公式进行的，并且使用转化后的 X 作为作为凸轮的输出：

计算公式：**X=MasterScaling*MasterPosition+MasterOffset**

因此，如果主轴的比例大于 1，所述凸轮将会运行在一个更高的速率，如果比例值小于 1，速率将会随之降低。

从轴 SlaveOffset，SlaveScaling：

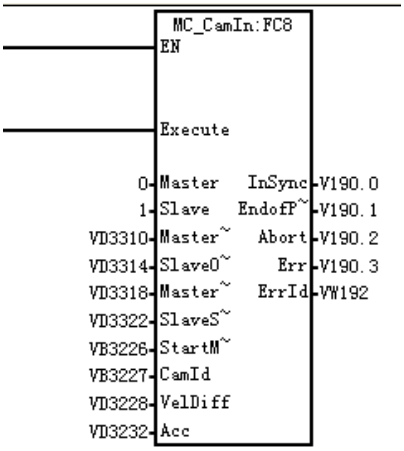
计算公式：**Y=SlaveScaling*CAM(X)+SlaveOffset**

如果 SlaveScaling>1 导致凸轮效果的拉伸，从轴的范围将会增加；如果 SlaveScaling<1 将会导致一个收缩。

CAM 表数据不变时， 此指令只需调用一次，后面不需要调用，当 CAM 表数据变化时，才需要重新调用本指令。

## *电子凸轮关联指令*

函数名：MC_CamIn




功能：此指令用于建立主从轴间的凸轮关系，建立凸轮关系时，可根据应用需求指定主从轴的偏移值，缩放比和启动模式。当指令执行完毕，从轴根据凸轮曲线跟随主轴运动。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Master | IN | 主轴ID号 | BYTE | |
| Slave | IN | 从轴ID号 | BYTE | |
| MasterOffset | IN | 主轴偏移 | REAL | 主轴凸轮位置偏移。单位：单元 |
| SlaveOffset | IN | 从轴偏移 | REAL | 从轴凸轮位置偏移。单位：单元 |
| MasterScaling | IN | 主轴缩放 | REAL | 主轴缩放配置参数。此参数不为0 |
| SlaveScaling | IN | 从轴缩放 | REAL | 从轴缩放配置参数。此参数不为0 |

| | | | | |
|---|---|---|---|---|
| StartMode | IN | 启动模式 | BYTE | 0：正向立即跳变<br>1：加速（走最短距离）跳变<br>2：斜坡正转<br>3：斜坡反转<br>注：正转反转只是针对从轴是旋转轴的情况，直线轴StartMode为0，1时为跳变，2，3为斜坡启动。该参数的四种模式只是针对从站来说的，从轴是绝对还是相对模式由MC_CamTableSelect指令配置。 |
| CamId | IN | 凸轮表的ID号 | BYTE | |
| VelDiff | IN | 凸轮表耦合速度 | REAL | |
| Acc | IN | 凸轮表加速度 | REAL | |
| InSync | OUT | 凸轮关系建立 | BOOL | 主轴和从轴建立关系后，InSync被置位，<br>当指令的执行条件Off 时，InSync位被复位。 |
| EndOfProfile | OUT | 凸轮关系结束 | BOOL | 如果MC_CamTableSelect指令执行时，Perioc参数为0，当凸轮曲线执行完成一次后，EndOfProfile被置位。<br>Perioc参数为1，当凸轮曲线执行完成一次后，EndOfProfile被置位，一周期后复位。<br>当指令执行条件Off时，EndOfProfile位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见13.3.4 错误代码 |

凸轮偏移 Offset 和缩放比例 Scaling：

主轴输入转换的位置是根据以下公式进行的，并且使用转化后的 X 作为作为凸轮的输出：

X=MasterScaling*MasterPosition+MasterOffset

因此，如果主轴的比例大于 1，所述凸轮将会运行在一个更高的速率，如果比例值小于 1，速率将会随之降低。

从轴 SlaveOffset,SlaveScaling：

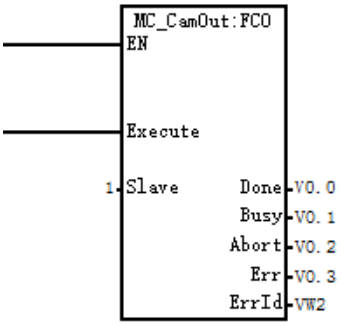
计算公式：

Y=SlaveScaling*CAM(X)+SlaveOffset

如果 SlaveScaling>1 导致凸轮效果的拉伸，从轴的范围将会增加；如果 SlaveScaling<1 将会导致一个收缩。

### *解除电子凸轮关联指令*

函数名：MC_CamOut



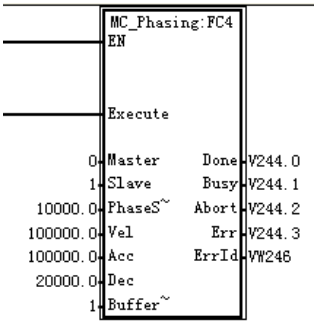

功能：此指令用于解除主从轴间的凸轮关系，关系解除后，从轴会以脱离前的速度继续运动。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Slave | IN | 从轴ID号 | BYTE | |
| Done | OUT | 完成位 | BOOL | 当取消凸轮关联指令执行成功时，Done位被置位。<br>当指令执行条件Off时，Done位复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位； |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

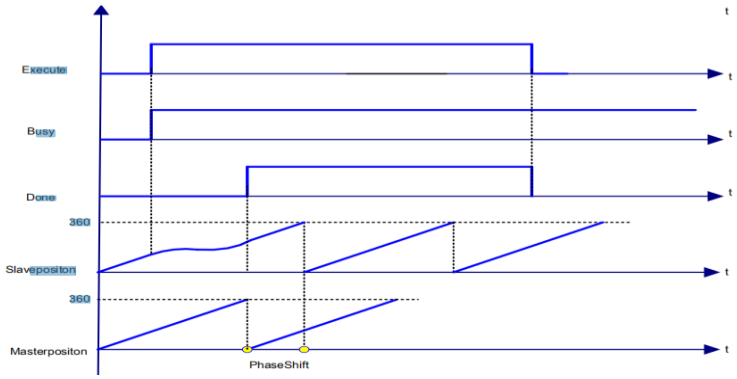
## *相位偏移指令*

函数名：MC_Phasing




功能：此指令用于调整主轴和从轴的相位差，当指令完调用完成之后，主轴和从轴之间相位差为 PhaseShift，即从轴位置=主轴位置-PhaseShift。

如下图所示，主从轴都按 360 周期运动，Execute 信号上升沿执行调整，调整完成后从轴与主轴之间相位偏差为 PhaseShift 设定的值。

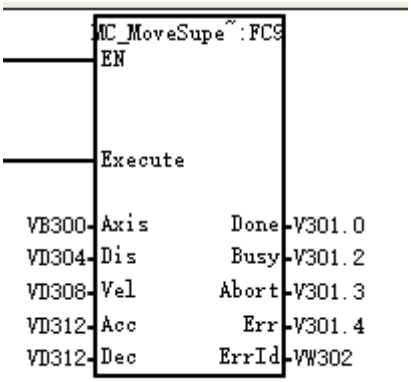



参数说明：

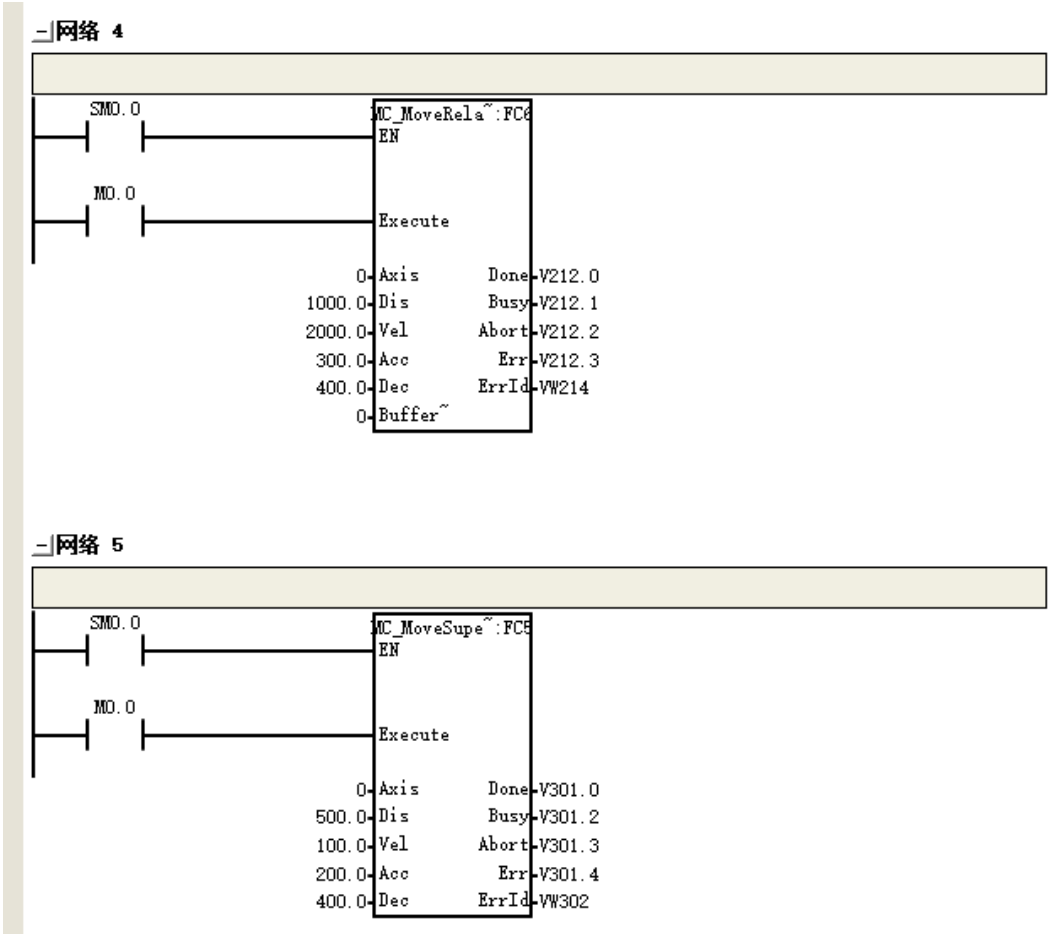
| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Master | IN | 主轴ID号 | BYTE | |
| Slave | IN | 从轴ID号 | BYTE | |
| PhaseShift | IN | 相位偏移 | REAL | 相位偏移 |
| Vel | IN | 速度 | REAL | 相位调整速度。 |
| Acc | IN | 加速度 | BOOL | 相位调整加速度。 |
| Dec | IN | 减速度 | BOOL | 相位调整减速度。 |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Done | OUT | 相位偏移调整完成 | BOOL | 相位调整完成Done被置位。<br>当指令的执行条件Off 时，Done位被复位 |
| Busy | OUT | 指令处理中 | BOOL | True，指令正在处理中 |
| Abort | OUT | 命令终止位 | BOOL | 当指令执行过程中被终止时，Abort位置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

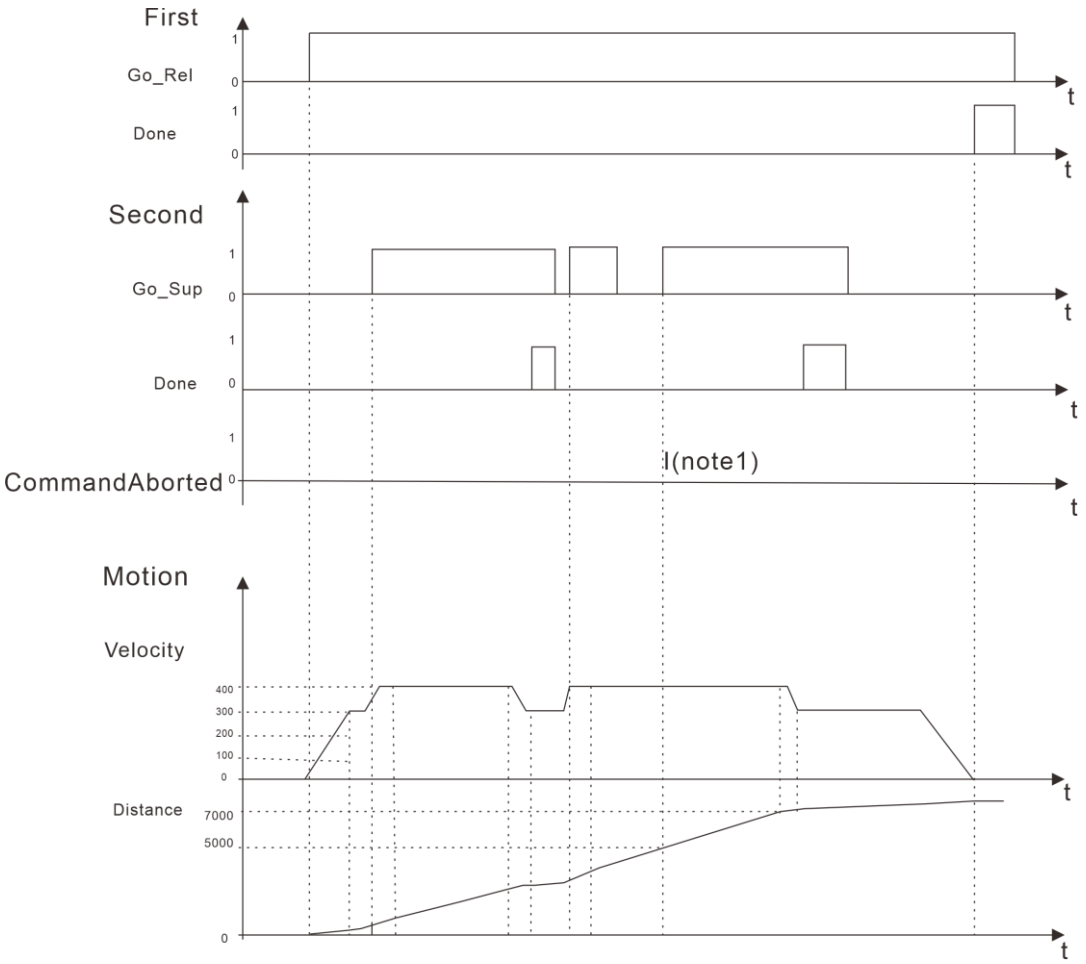
## *叠加相对位移指令*

函数名：MC_MoveSuperImposed




功能：此指令用于控制终端执行机构在当前运动状态下按给定速度，加减速独立的追加一段给定距离，此指令执行时，不会终止前一个指令执行，两条指令同时执行，距离、速度、加减速会实时叠加。当其中一个指令执行完毕，将不再叠加其速度、加减速，另一指令仍然独立运行。





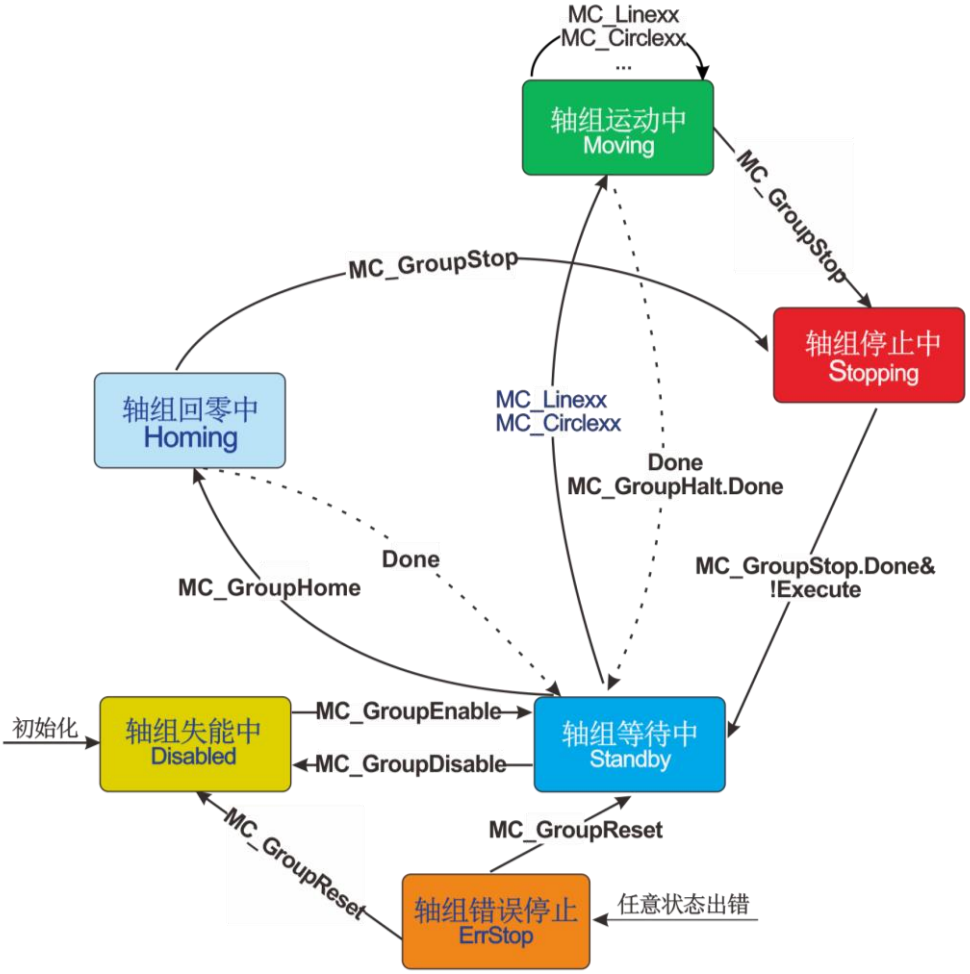

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| Axis | IN | 轴ID号 | BYTE | |
| Dis | IN | 距离 | REAL | 以当前位置为参考，终端执行机构要移动的目标距离，该值设置为负数代表轴将会反转。<br>单位：单元。 |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。<br>（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。<br>（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。<br>（单位：单元/秒/秒） |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |

| | | | | |
|---|---|---|---|---|
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见13.3.4 错误代码 |

### 13.3.3 轴组指令

轴组的运动控制指令，一般用于多轴的直线插补、圆弧插补指令等。轴组指令状态流程图如下图所示：




**提示**

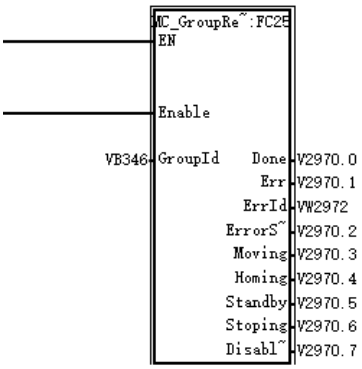
图中 MC_GroupHome，GroupStop 等函数还没有实现。因此 Homing 和 Stopping 两种轴组状态暂时保留。

轴组指令常用到的 MC 功能块如下：

| 函数名 | 指令功能 | 备注 |
|---|---|---|
| MC_GroupReadState | 读取轴组状态指令 | |
| MC_GroupEnable | 轴组有效指令 | |
| MC_GroupDisable | 轴组无效指令 | |
| MC_GroupReset | 轴组错误复位指令 | |
| MC_GroupHalt | 轴组停止指令 | |
| MC_MoveLinerRelative | 相对位移直线插补指令 | |
| MC_MoveLinerAbsolute | 绝对位移直线插补指令 | |
| MC_MoveCircularRalative | 相对位移圆弧插补指令 | |
| MC_MoveCircularAbolute | 绝对位移圆弧插补指令 | |

## *读取轴组状态指令*

函数名：MC_GroupReadState




功能：此指令用于读取相应的轴组状态参数的值。

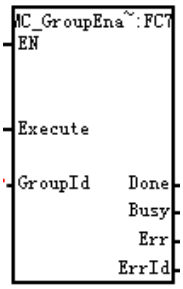
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Enable | IN | 指令执行条件 | BOOL | 当为On时，执行该指令。 |
| GroupId | IN | 轴组Id号 | BYTE | 轴组号 |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位； |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Err位被置位；<br>当指令的执行条件由On变Off时，Err位被复位。 |
| ErrId | OUT | 错误码 | WORD | 当Err为TRUE时，ErrId为指令的错误。<br>当ErrStop为TRUE时，ErrId为轴组当前错误。 |
| ErrStop | OUT | 状态指示位 | BOOL | TRUE，如果轴组位于状态ErrStop |
| Stopping | OUT | 状态指示位 | BOOL | TRUE，如果轴组位于状态Stopping |
| Moving | OUT | 状态指示位 | BOOL | TRUE，如果轴组位于状态Moving |
| Standby | OUT | 状态指示位 | BOOL | TRUE，如果轴组位于状态Standby |
| Homing | OUT | 状态指示位 | BOOL | TRUE，如果轴组位于状态Homing |

| | | | | |
|---|---|---|---|---|
| Disabled | OUT | 状态指示位 | BOOL | TRUE，如果轴组位于状态Disabled |

### *轴组有效指令*

函数名：MC_GroupEnable



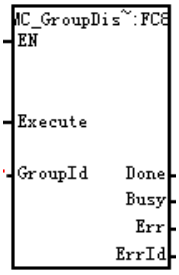

功能：使轴组有效。轴组状态从 GroupDisalbed 转变为 GroupStandby。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off变On时，执行该指令。 |
| GroupId | IN | 轴组Id号 | BYTE | 轴组号 |
| Busy | OUT | 忙位 | BOOL | 指令正在执行为TRUE，执行完成为FLASE |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位； |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *轴组无效指令*

函数名：MC_GroupDisable




功能：使轴组无效。将轴组状态由 GroupStandby 转变为 GroupDisalbed。

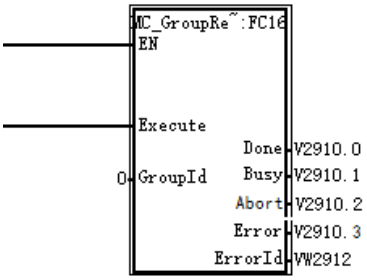
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| GroupId | IN | 轴组ID号 | BYTE | 轴组号 |
| Done | OUT | 状态位 | BOOL | 指令执行成功后，Done 位被置位； |
| Busy | OUT | 忙位 | BOOL | 指令正在执行为TRUE，执行完成为FLASE |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位； |

| | | | | |
|---|---|---|---|---|
| | | | | 当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *轴组错误复位指令*

函数名：MC_GroupReset



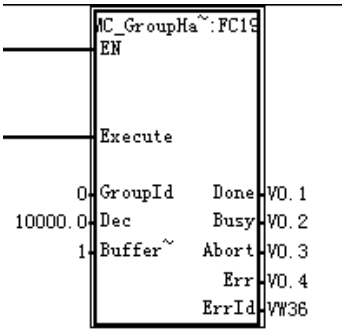

功能：轴组处于 ErrStop 状态时，调用本指令，清除轴组和轴的错误，并将轴组状态切换到 Disabled（出错前轴组未使能）或 Standby。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| GroupId | IN | 轴组Id号 | BYTE | 轴组号 |
| Done | IN | 指令完成位 | BOOL | 完成位 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 保留接口，不会出现 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *轴组停止指令*

函数名：MC_GroupHalt




功能：使插补动作中的所有轴减速停止。

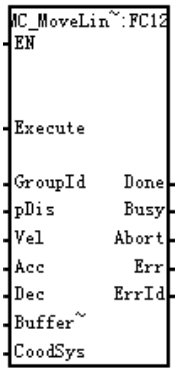
参数说明：

| 参数名 | 输入输 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|

| | 出属性 | | | |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| GroupId | IN | 轴组Id号 | BYTE | 轴组号 |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。（单位：单元/秒），当设为0时，实现紧急停止 |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令)<br>其他：非法 |
| Done | OUT | 完成位 | BOOL | 指令执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Abort | OUT | 中止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *相对位移直线插补指令*

函数名：MC_MoveLinerRelative




功能：此指令按照相对的位移进行直线插补

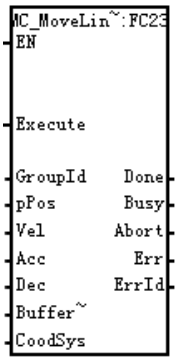
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| GroupId | IN | 轴组ID号 | BYTE | |
| pDis | IN | 相对位置指针 | DWORD | 指向描述目标距离的指针。<br>目标距离为描述为6个REAL类型的数据。代表（MCS下的空间位移）<br>X：REAL;<br>Y：REAL;<br>Z：REAL;<br>A：REAL;<br>B：REAL: |

| | | | | |
|---|---|---|---|---|
| | | | | C：REAL,<br>或（ACS下的每个轴的位移）<br>A0：REAL<br>A1：REAL;<br>A2：REAL;<br>A3：REAL:<br>A4：REAL;<br>A5：REAL;<br>（单位：单元） |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。<br>（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。<br>（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。<br>（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令，禁用过渡)<br>1：Buffered(前一段减速完成开始执行缓冲的功能块)<br>2：BendingPrevious(以当前速度走到前一段结束并按照前一段的速率开始执行下一段，禁用过渡)<br>16#12：TMCorner（附加角过渡，在前一段开始执行减速时以附加角过渡到下一段）；<br>详情见 BufferMode 连续插补具体介绍 |
| CoordSystem | IN | 坐标系统 | BYTE | 0：ACS<br>1：MCS (暂时只有这种)<br>2：WCS<br>3：PCS_1<br>4：PCS_2 |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *绝对位移直线插补指令*

函数名：MC_MoveLinerAbsolute




功能：此指令按照绝对的位移进行直线插补

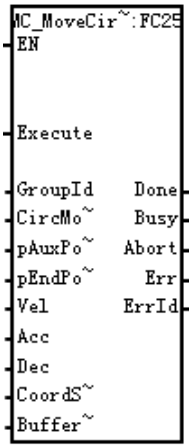
参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off变On时，执行该指令。 |
| GroupId | IN | 轴组ID号 | BYTE | |
| pPos | IN | 绝对位置指针 | DWORD | 指向描述目标位置的指针。<br>目标位置为描述为6个REAL类型的数据。代表（MCS下的空间位移）<br>X：REAL;<br>Y：REAL;<br>Z：REAL;<br>A：REAL;（保留）<br>B：REAL: (保留)<br>C：REAL,（保留）<br>或（ACS下的每个轴的位移）<br>A0：REAL<br>A1：REAL;<br>A2：REAL; (对两轴保留)<br>A3：REAL:（保留）<br>A4：REAL;（保留）<br>A5：REAL:（保留）<br>（单位：单元） |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令，禁用过渡)<br>1：Buffered(前一段减速完成开始执行缓冲的功能块)<br>2：BendingPrevious(以当前速度走到前一段结束并按照前一段的速率开始执行下一段，禁用过渡)<br>16#12：TMCorner（附加角过渡，在前一段开 |

| | | | | |
|---|---|---|---|---|
| | | | | 始执行减速时以附加角过渡到下一段）；详情见BufferMode连续插补具体介绍 |
| CoordSystem | IN | 坐标系统 | BYTE | 0：ACS<br>1：MCS (暂时只有这种)<br>2：WCS<br>3：PCS_1<br>4：PCS_2 |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *相对位移圆弧插补指令*

函数名：MC_MoveCircularRalative




功能：此指令按照相对的位移进行圆弧插补。

参数说明：

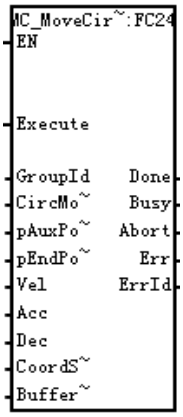
| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| GroupId | IN | 轴组ID号 | BYTE | |

| | | | | |
|---|---|---|---|---|
| CircMode | IN | 圆弧插补方式 | BYTE | Bit0~Bit3<br>0：三点式<br>1：中心点式。<br>8：三点式全圆<br>9：中心点式全圆<br>Bit4：插补方向（中心点式和全圆有效）<br>0：为反转(CCW)<br>1：正转(CW)<br>圆弧的方向是根据右手定则选择的：CCW方向是沿着手指弯曲的方向。<br>当为中心点式时，如果中心点与起点和终点的距离不完全相同（这通常是由于编程中心点的精度有限），则将其投影到起点和终点的垂直平分线上。一旦中心点离得太远（超过半径的1%），就会返回一个错误。 |
| pAuxPoint | IN | 参考位置指针 | DWORD | 指向描述参考位置的指针。<br>三点式时，pAuxPostion指的是经过的参考点。<br>当为中心点式时，pAuxPostion指的是圆心。<br>参考位置为描述为6个REAL类型的数据。代表（MCS下的空间位移）<br>X：REAL<br>Y：REAL<br>Z：REAL<br>A：REAL（保留）<br>B：REAL（保留)<br>C：REAL（保留）<br>（单位：单元） |
| pEndPoint | IN | 目标位置指针 | DWORD | 指向描述目标位置的指针。<br>目标位置为描述为6个REAL类型的数据。代表（MCS下的空间位移）<br>X：REAL<br>Y：REAL<br>Z：REAL<br>A：REAL（保留）<br>B：REAL（保留）<br>C：REAL（保留） |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。<br>（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。<br>（单位：单元/秒/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。<br>（单位：单元/秒/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令，禁用过渡)<br>1：Buffered(前一段减速完成开始执行缓冲的功能块)<br>2：BendingPrevious(以当前速度走到前一段结束并按照前一段的速率开始执行下一段，禁用过渡)<br>16#12：TMCorner（附加角过渡，在前一段开 |

| | | | | |
|---|---|---|---|---|
| | | | | 始执行减速时以附加角过渡到下一段）；详情见BufferMode连续插补具体介绍 |
| CoordSystem | IN | 坐标系统 | BYTE | 0：ACS<br>1：MCS (暂时只有这种)<br>2：WCS<br>3：PCS_1<br>4：PCS_2 |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

### *绝对位移圆弧插补指令*

函数名: MC_MoveCircularAbolute




功能：此指令按照绝对的位移进行圆弧插补。

参数说明：

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Execute | IN | 指令执行条件 | BOOL | 当执行条件由Off 变On时，执行该指令。 |
| GroupId | IN | 轴组ID号 | BYTE | |
| CircMode | IN | 圆弧插补方式 | BYTE | Bit0-Bit3<br>0：三点式<br>1：中心点式。<br>8：三点式全圆<br>9：中心点式全圆<br>Bit4: 插补方向（中心点式有效）<br>0：为反转(CCW)<br>1：正转(CW)<br>圆弧的方向是根据右手定则选择的：CCW方向 |

| | | | | |
|---|---|---|---|---|
| | | | | 是沿着手指弯曲的方向。<br>当为中心点式时，<br>如果中心点与起点和终点的距离不完全相同（这通常是由于编程中心点的精度有限），则将其投影到起点和终点的垂直平分线上。一旦中心点离得太远（超过半径的1%），就会返回一个错误。 |
| pAuxPoint | IN | 参考位置指针 | DWORD | 指向描述参考位置的指针。<br>当三点式时，pAuxPostion指的是经过的参考点。<br>当为中心点式时，pAuxPostion指的是圆心。<br>参考位置距离为描述为6个REAL类型的数据。代表（MCS，PCS，WCS下的）<br>X：REAL<br>Y：REAL<br>Z：REAL<br>A：REAL（保留）<br>B：REAL（保留）<br>C：REAL（保留）<br>（单位：单元） |
| pEndPoint | IN | 结束位置指针 | DWORD | 指向描述结束位置的指针。<br>结束位置距离为描述为6个REAL类型的数据。<br>X：REAL<br>Y：REAL<br>Z：REAL<br>A：REAL（保留）<br>B：REAL（保留）<br>C：REAL（保留）（单位：单元） |
| Vel | IN | 速度 | REAL | 终端执行机构的运转速度，此参数总为正。<br>（单位：单元/秒） |
| Acc | IN | 加速度 | REAL | 终端执行机构的加速度，此参数总为正。<br>（单位：单元/秒） |
| Dec | IN | 减速度 | REAL | 终端执行机构的减速度，此参数总为正。<br>（单位：单元/秒） |
| BufferMode | IN | 中止模式 | BYTE | 0：Abort (直接打断正在进行的指令，禁用过渡)<br>1：Buffered(前一段减速完成开始执行缓冲的功能块)<br>2：BendingPrevious(以当前速度走到前一段结束并按照前一段的速率开始执行下一段，禁用过渡)<br>16#12：TMCorner（附加角过渡，在前一段开始执行减速时以附加角过渡到下一段）；详情见BufferMode连续插补具体介绍 |
| CoordSystem | IN | 坐标系统 | BYTE | 0：ACS；1：MCS (暂时只有这种)；2：WCS<br>3：PCS_1；4：PCS_2 |
| Done | OUT | 完成位 | BOOL | 绝对位移动作执行完成时，Done位被置位；<br>当指令的执行条件Off 时，Done位被复位。 |
| Abort | OUT | 命令终止位 | BOOL | 当该指令执行过程中被终止时，Abort位被置位；<br>当指令的执行条件Off 时，Abort位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。 |

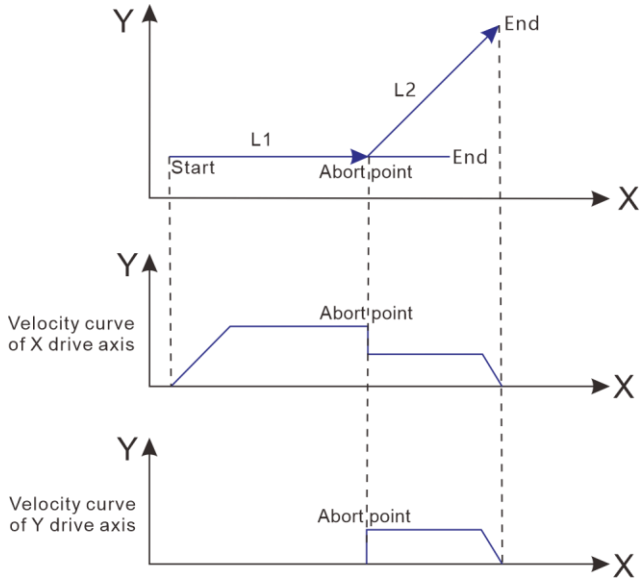
| | | | | |
|---|---|---|---|---|
| | | | | 当指令完成Busy复位。<br>当指令的执行条件Off 时，Busy位被复位。 |
| Err | OUT | 错误位 | BOOL | 如果检测到有错误，Error位被置位；<br>当指令的执行条件Off 时，Error位被复位。 |
| ErrId | OUT | 错误代码 | WORD | 错误代码，详见章节13.3.4 错误代码。 |

***BufferMode 连续插补具体介绍***

| 取值 | 描述 | 详细 |
|---|---|---|
| 0 | Abort | 直接打断正在进行的指令，禁用过渡 |
| 1 | Buffered | 前一段减速完成，开始执行缓冲的指令 |
| 2 | BendingPrevious | 以前一段速度走到前一段结束并按照前一段的速率开始执行下一段，禁用过渡 |
| 16#12 | TMCorner | 附加角过渡，在前一段开始执行减速时以附加角过渡到下一段 |

**Abort**

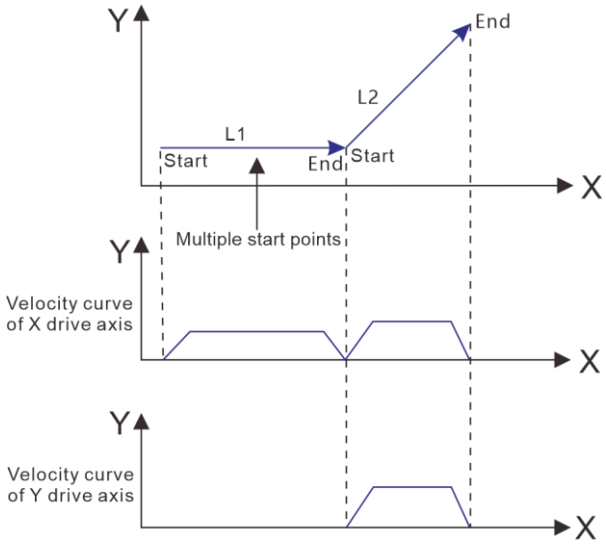
直线 L1 先执行第一条插补指令，在 L1 走完之前触发第二条插补指令，如果第二条插补指令的缓冲模式设置为“Abort“，则第二条插补指令立即打断第一条插补指令并开始执行新的插补曲线。




**Buffered**

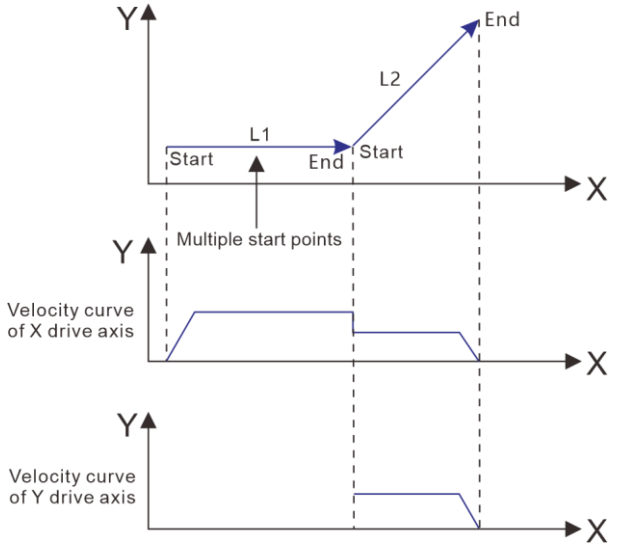
直线 L1 首先执行第一条插补指令，在 L1 走完之前触发第二条插补指令，如果第二条插补指令的缓冲模式设置为“Buffered”，插补器将继续执行第一条插补指令。

等第一条插补指令执行完，Done 信号输出有效后，第二条插补指令开始执行，如下图所示：

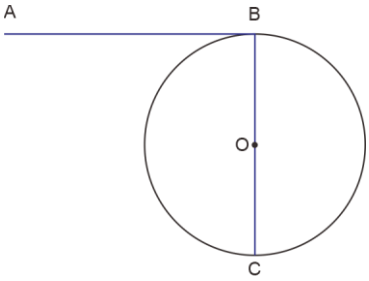



**BendingPrevious**

直线 L1 先执行第一条插补指令，在 L1 走完之前触发第二条插补指令。如果第二条插补指令的缓冲模式设置为“BendingPrevious”，插补器将尝试保持第一条指令的目标速度走完整条直线。等待第一条插补指令执行完，Done 信号输出有效后，第二条插补指令开始执行，切换点将保持速率不变，坐标轴的分速度将重新分配。如下图所示：

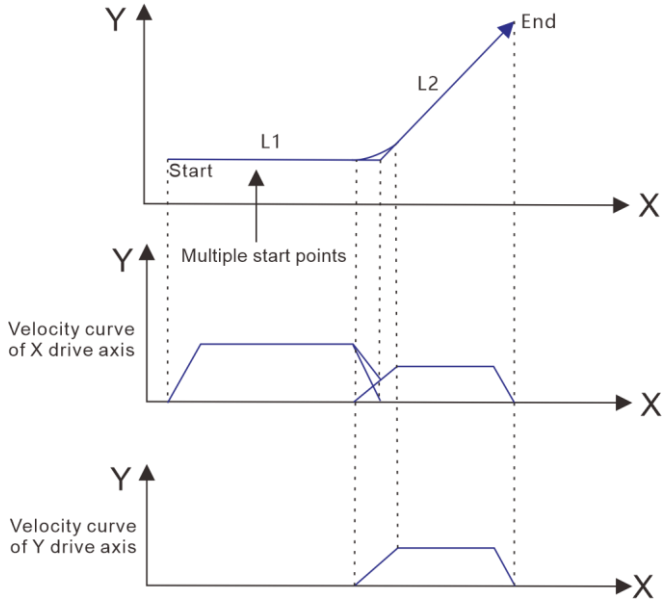



该模式适合在直线与圆弧相互切换且直线位于圆弧的切线上，采用该模式可以保持插补曲线的速度连续。



**TMCorner**

直线 L1 首先执行第一条插补指令，在 L1 走完之前触发第二条插补指令。如果第二条插补指令的缓冲模式设置为“TMCorner”，插补器检测到第一条直线 L1 开始执行减速时，启动第二条插补指令，以附加角的方式过渡到第二条插补，保持各方向上速度连续，如下图所示：




## 13.3.4 错误代码

错误代码说明：

| 错误码 | 说明 | 错误码 | 说明 |
|---|---|---|---|
| 0 | 没有错误 | 44 | Position 参数设置过小 |
| 1 | 轴号取值错误 | 45 | 主轴坐标超范围 |
| 2 | 轴状态错误 | 46 | 运行超速（运行速度超过最大速度） |
| 3 | 指令丢失 | 50 | 通讯错误 |
| 4 | 轴忙 | 51 | 伺服内部错误 |
| 5 | 速度取值太小（小于最小速度） | 52 | 没有错误需要清除 MC_Reset |
| 6 | 速度取值太大（大于最大速度） | 53 | 没有上电 |
| 7 | 加速度取值太小 | 60 | RetioNumerator 取值错误 |
| 8 | 加速度取值太大 | 61 | RetioMenominator 取值错误 |
| 9 | 减速度取值太小 | 65 | 凸轮表和 MC_TableSelect 不一致。 |
| 10 | 减速度取值太大 | 66 | 凸轮表 ID 错误（或没有创建） |
| 11 | 方向取值错误 | 67 | 凸轮表数据错误 |
| 12 | HSP 轴取值错误 | 68 | MasterScaling 取值错误 |
| 13 | HSP 模块号错误 | 69 | SlaveScaling 取值错误 |
| 15 | 轴未准备好 | 70 | StartMode 取值错误 |
| 16 | 复位错误失败 | 80 | 达到左限位开关 |
| 17 | GroupId 错误 | 81 | 达到右限位开关 |
| 19 | 距离指针错误。 | 82 | 到达负向软件限位 |

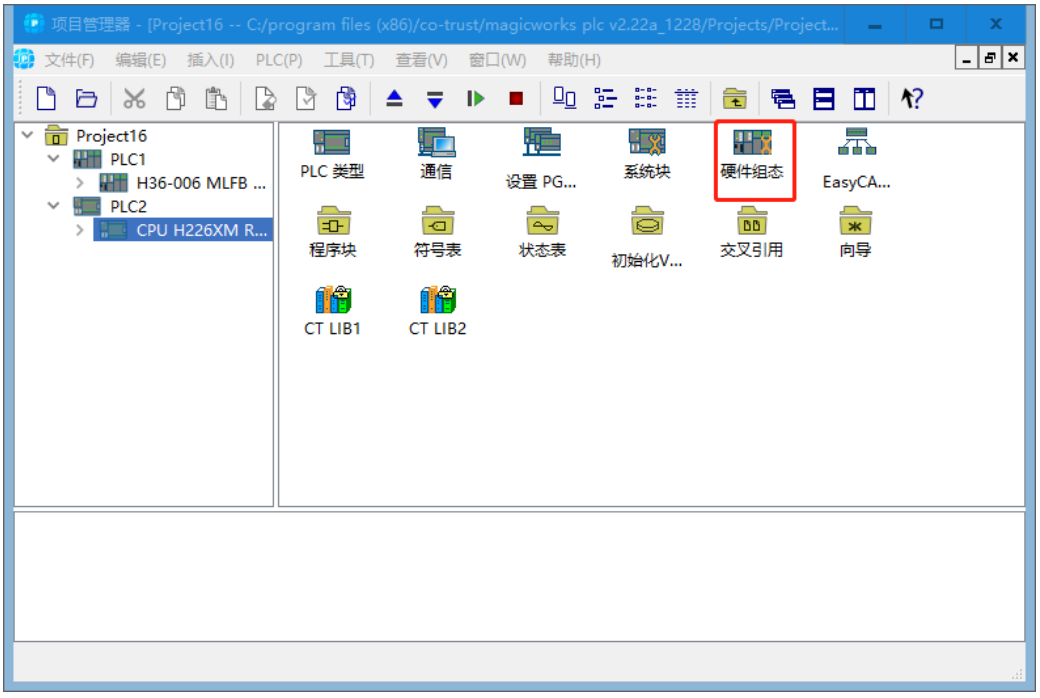
| | | | |
|---|---|---|---|
| 20 | BufferMode 取值错误 | 83 | 到达正向软件限位 |
| 21 | CoordSys 取值错误 | 90 | cmd 取值错误 |
| 22 | 内部软件错误 | 91 | TriggerInput 取值错误 |
| 23 | 内存申请错误 | 92 | TriggerVar 取值错误 |
| 24 | CircMode 设置错误 | 93 | MC_Feed 执行失败 |
| 25 | 圆弧三点成一线 | 94 | MC_Feed 指令正在执行 |
| 26 | 两点到圆心距离不相等 | 95 | 模块中没有此中断，或中断没有使能。 |
| 30 | 回原模式缺少 Z 相配置 | 100 | 轴组中的轴错误 |
| 31 | 回原模式缺少正向开关配置 | 101 | 轴组状态设置错误 |
| 32 | 回原模式缺少负向开关配置 | 102 | 轴组中有轴已处于别的轴组控制之中。 |
| 33 | 回原模式缺少原点开关配置 | 110 | 超过最大缓冲指令条数。 |
| 34 | 回原模式取值错误 | 1000 | 此配置将来才会支持 |
| 35 | 伺服内部回原错误 | 2001~2022 | 模块错误 |
| 43 | Position 参数设置过大 | 3000 | 组态中存在错误 |

<备注> 连续插补最大缓冲指令条数为 8 条

### 13.3.5 连续插补功能应用举例

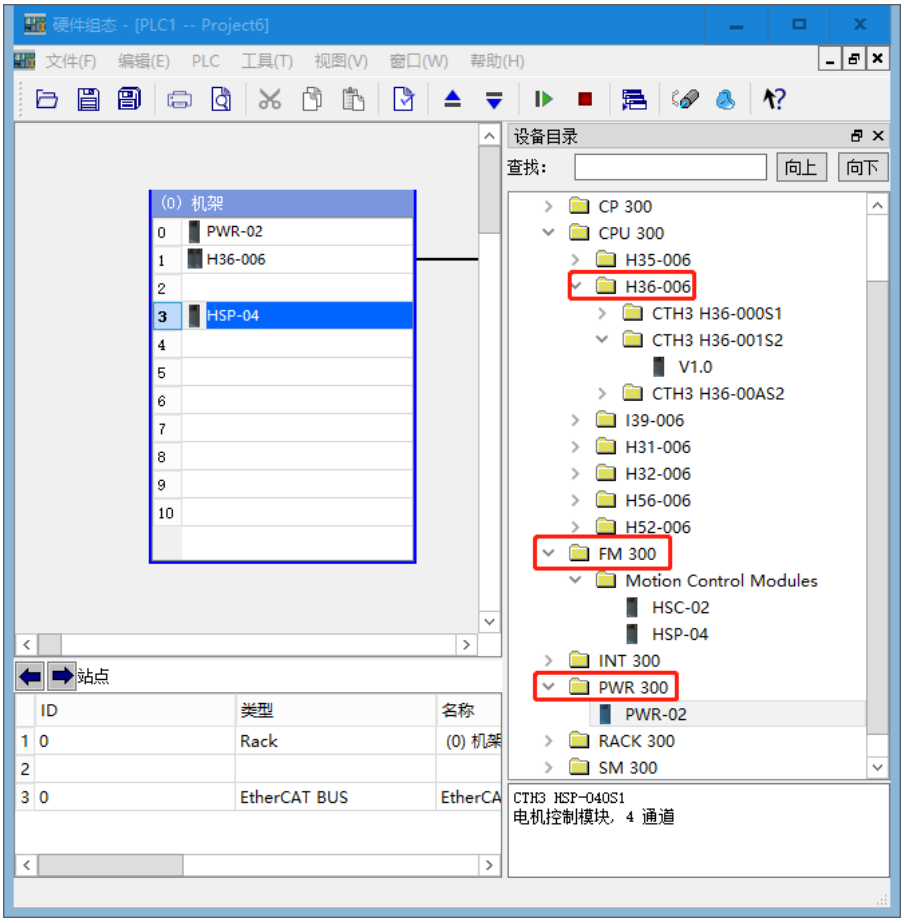
下面以 CPU H36-006 为例子对连续插补功能进行具体操作介绍。

在 H36-006 CPU 站点的项目管理器界面选择“硬件组态”，进行硬件组态，然后在主程序中调用插补指令进行编程。本例调用的插补指令为相对位移直线插补指令。



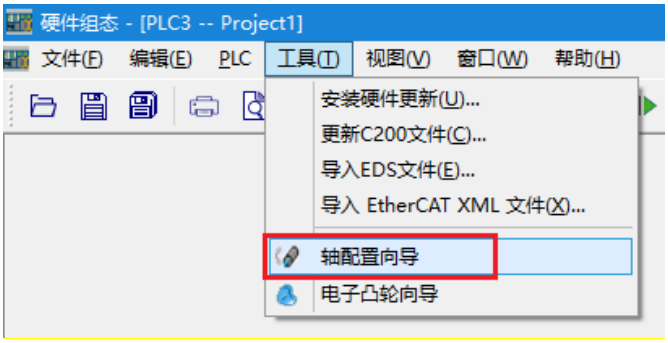

**步骤一：添加电源模块、CPU 和 HSP 模块**

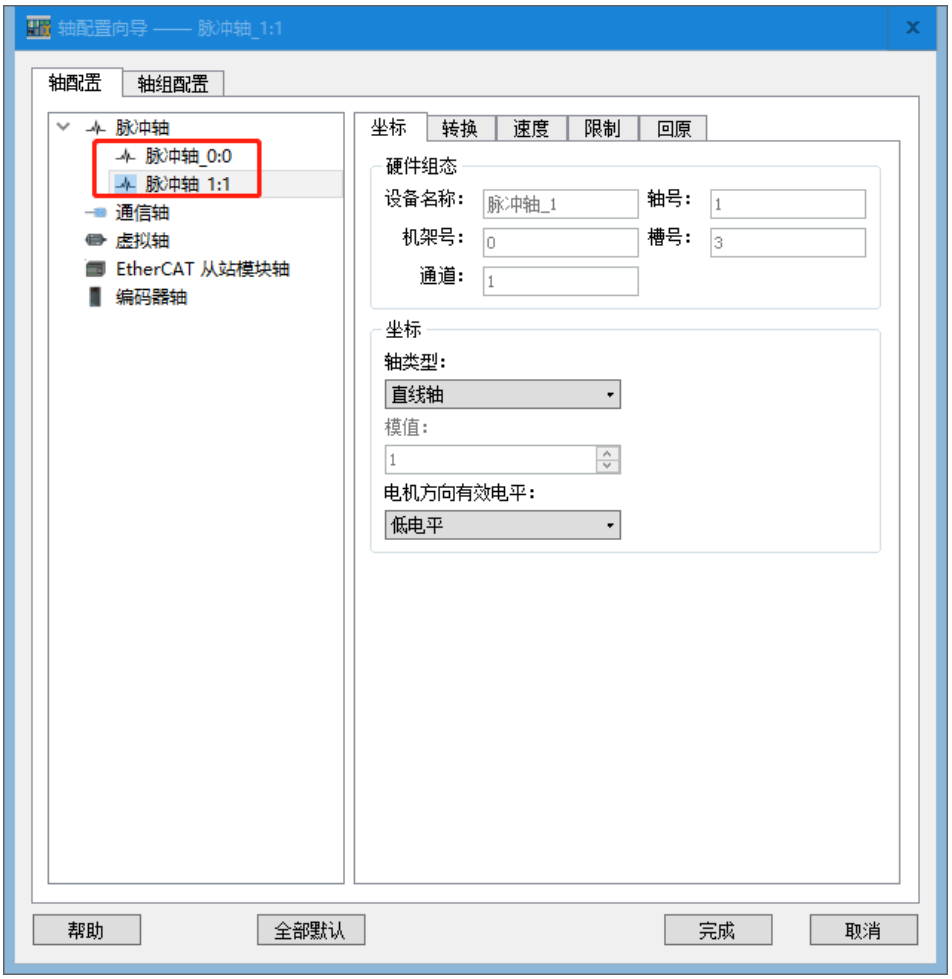
硬件组态界面中，在“CO-TRUST 300”项目树下顺序展开“PWR 300”、“CPU 300”和“FM 300”文件夹，添加电源模块、CPU 和 HSP 模块，电源模块只能放至机架的 0 号槽内，CPU 只能放至机架的 1 号槽内，HSP 模块可放在 3-10 号卡槽内，如下图所示。




**步骤二：配置轴**

在硬件组态界面选择“工具”→“轴配置向导”，添加两个脉冲轴，如下图所示。

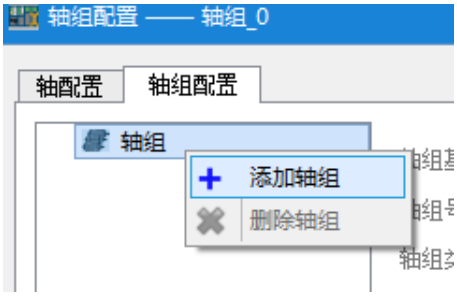




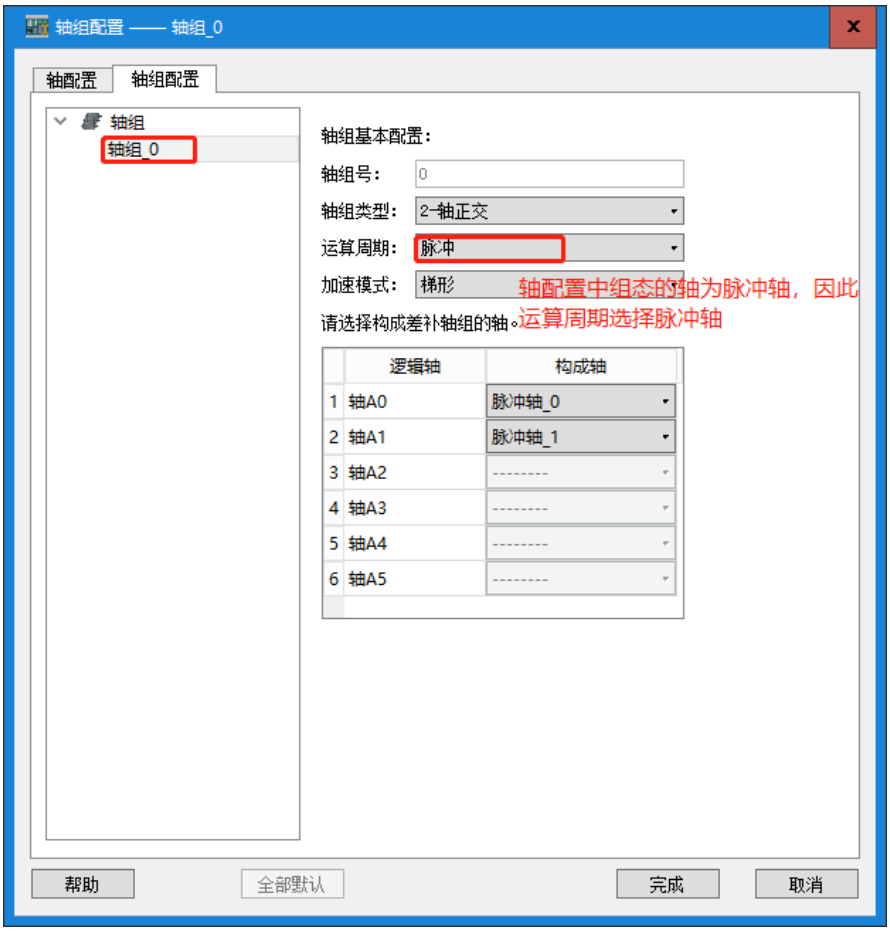

<**备注**> 使用插补功能需要配置两个或两个以上的脉冲轴或 EtherCAT 轴。

**步骤三：配置轴组**

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，本例配置的轴有两个；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：



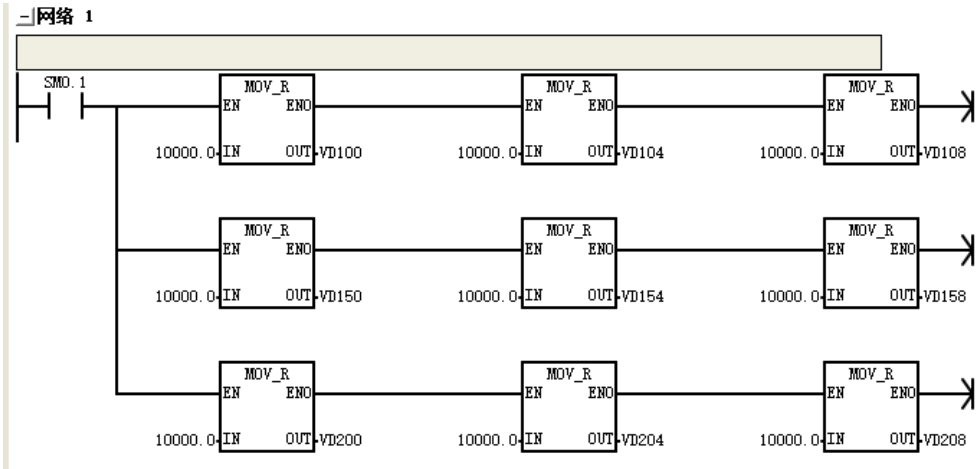

添加成功的轴组界面如下所示：



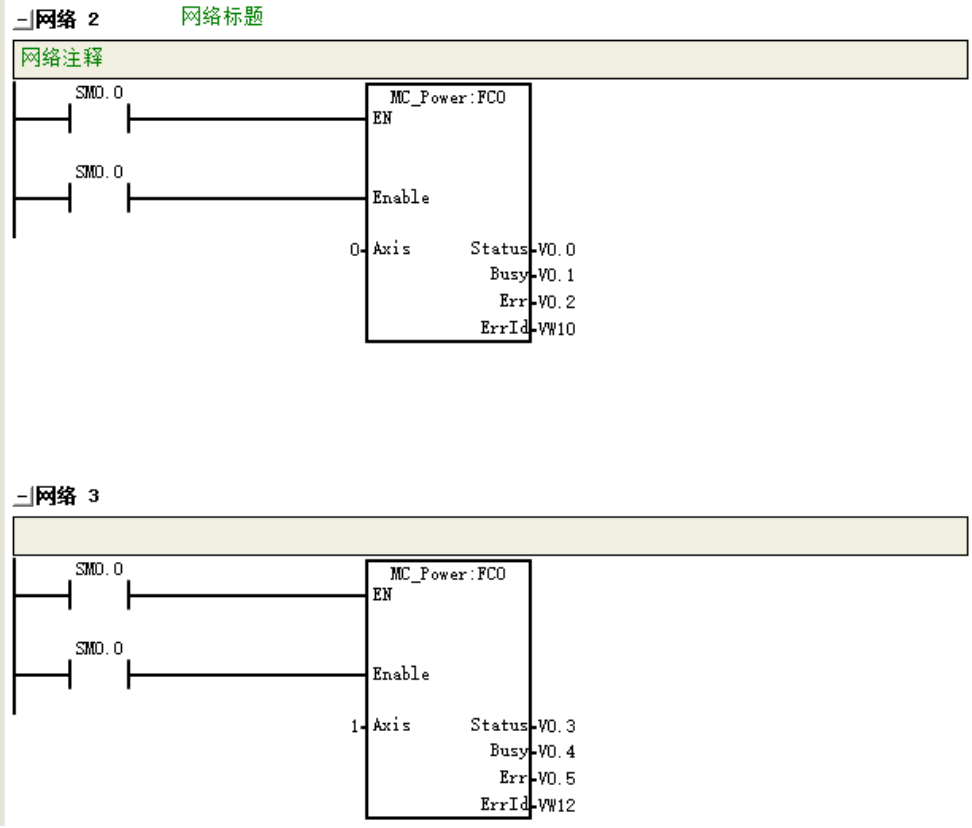

- “轴组基本配置”：“轴组号”不可更改，“轴组类型”下拉选项可选择“2-轴正交”或“3-轴正交”，分别对应可选两组或三组“构成轴”；“运算周期”可选“脉冲”或“EtherCAT”；“加速模式”可选“梯形”，“sin*2”和“二次”。
- “逻辑轴”：默认六组，不可更改。
- “构成轴”：“2-轴正交”对应两个构成轴，“3-轴正交”对应三个构成轴，下拉选项可选择“脉冲轴”或“虚拟轴”。

**步骤四：在主程序中调用插补指令进行编程。**

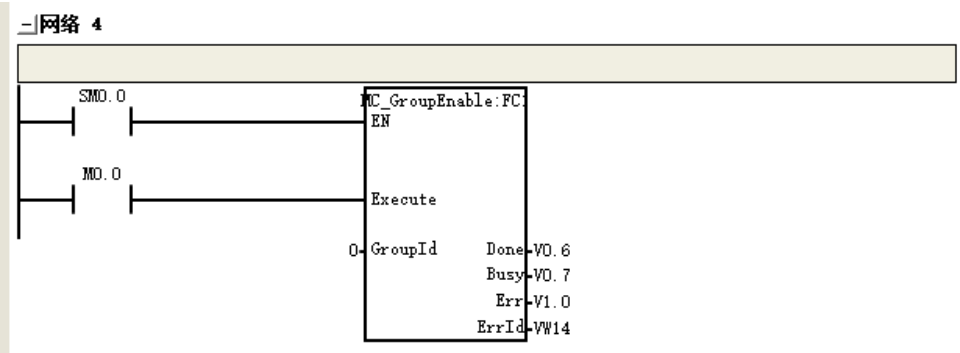
网络 1：设置插补指令的速度



网络 2、网络 3：分别对两个脉冲轴进行使能操作。

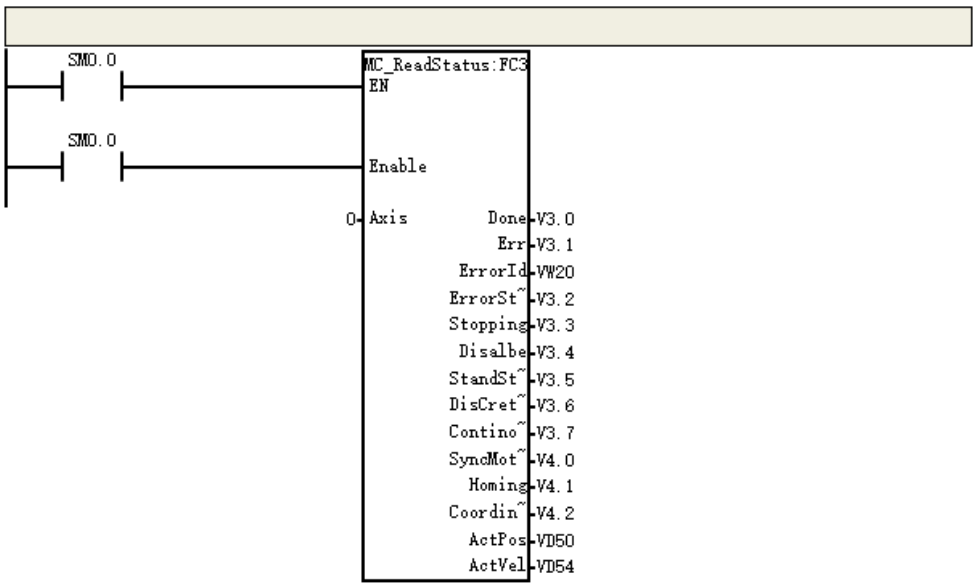



网络 4：调用轴组有效指令，使轴组有效

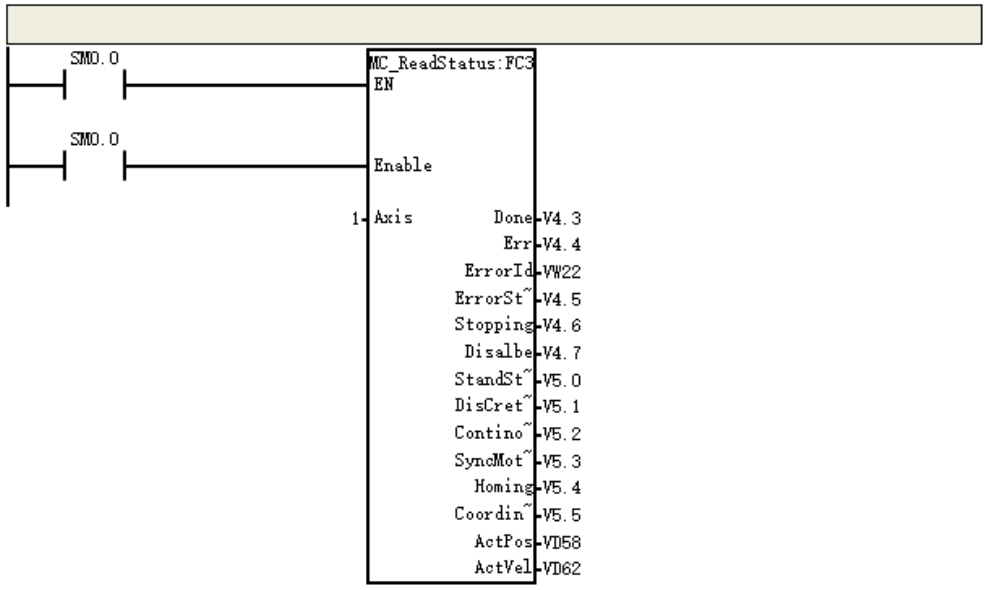


网络 5、网络 6：读取相应的轴状态。

网络 5

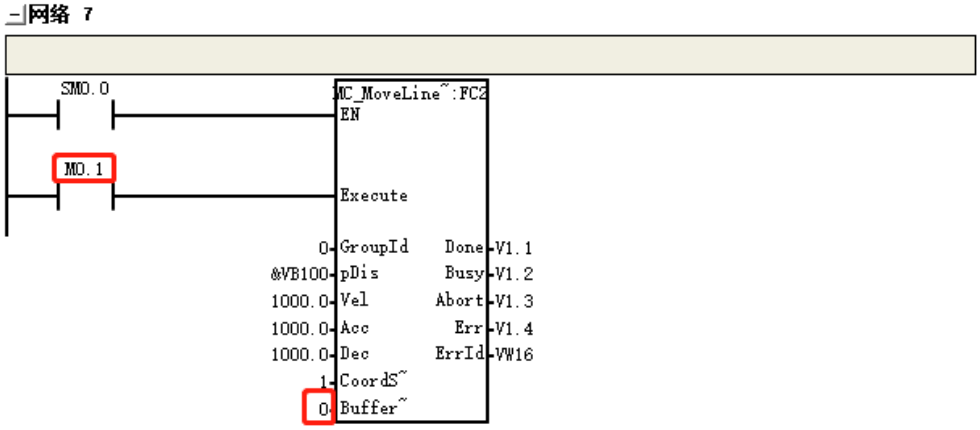


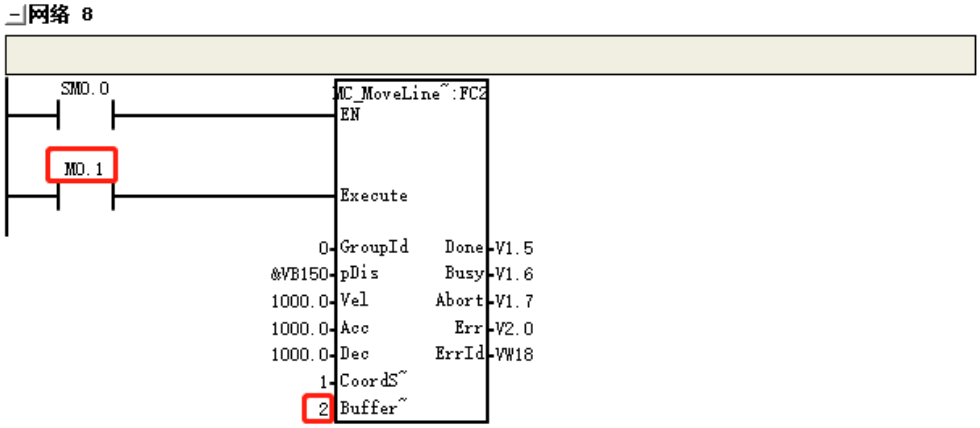

网络 6

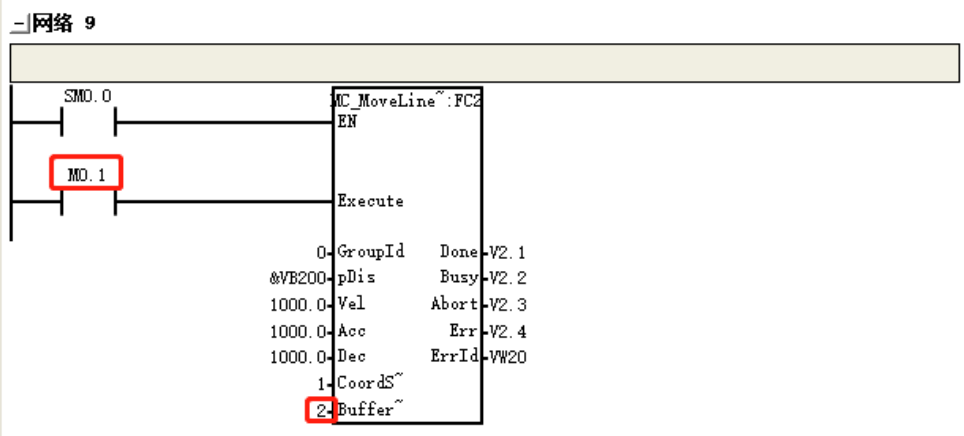


网络 7、8、9:设置运行速度、加速度、减速度，插补模式等参数，指令使能位可以用同一个位来同时触发（程序中缓存 2 条连续插补指令），然后按照指令先后顺序进行连续直线插补运动。

**<备注>：** 连续插补一次最大缓存指令条数为 8 条，如果想缓存第 9 条，当第 1 条指令运行完成后再使能缓存第 9 条。



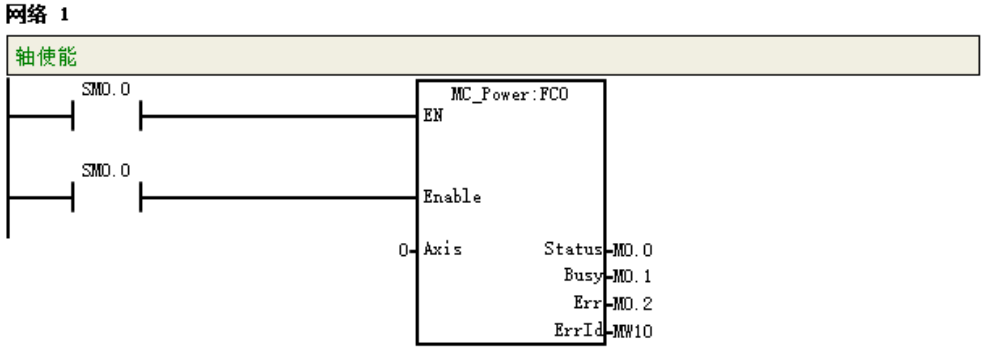

## 13.3.6 电子凸轮程序示例

以下程序为 CTH300-H 系列 Ethercat CPU 使用轴指令 ct_plcopen_lib(v2.0)实现电子凸轮功能

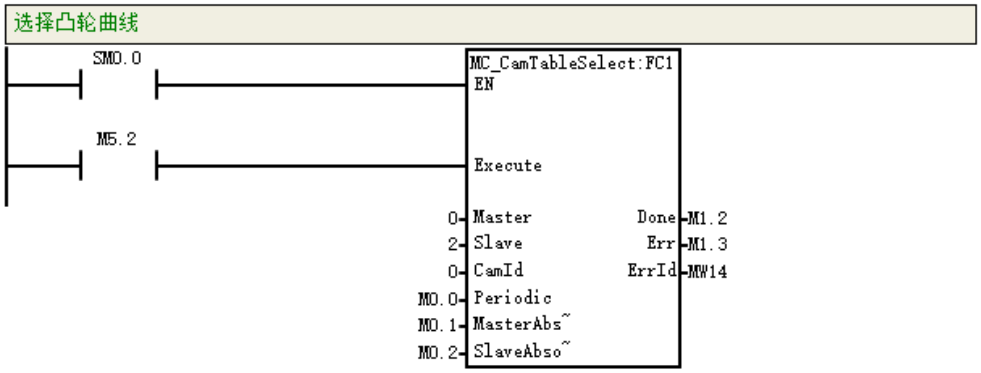
的示例。

**程序实现的网络示意图**

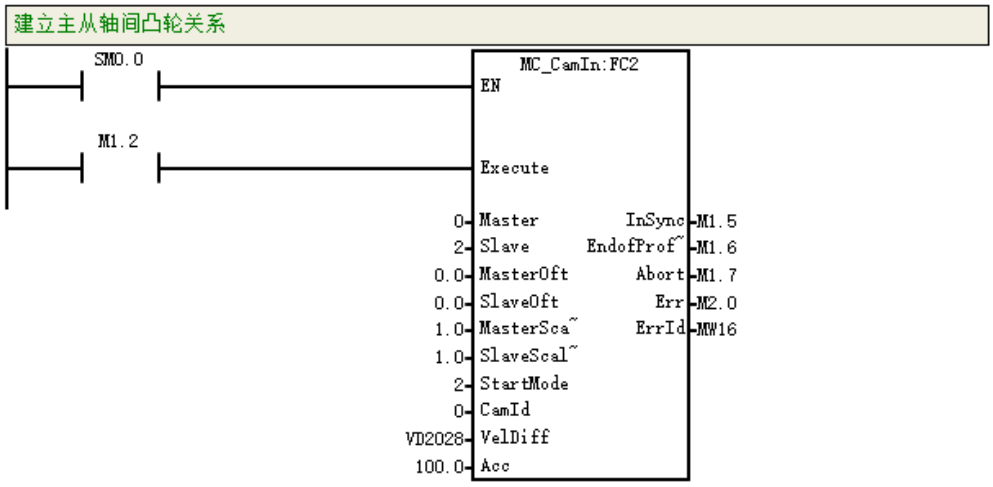
网络 1：对轴进行使能操作。



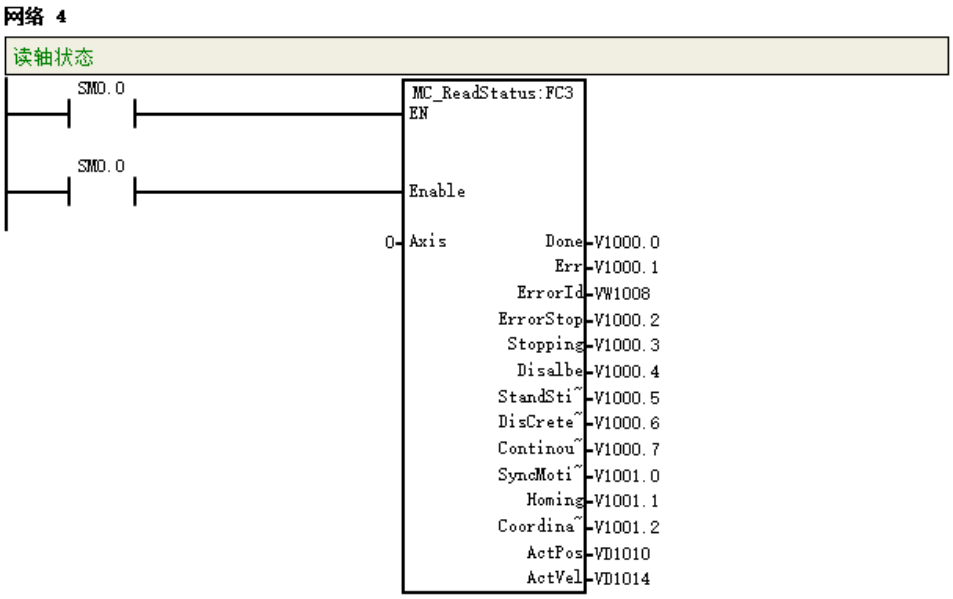

网络 2：选择凸轮曲线，设置主站、从站信息、状态信息等参数。



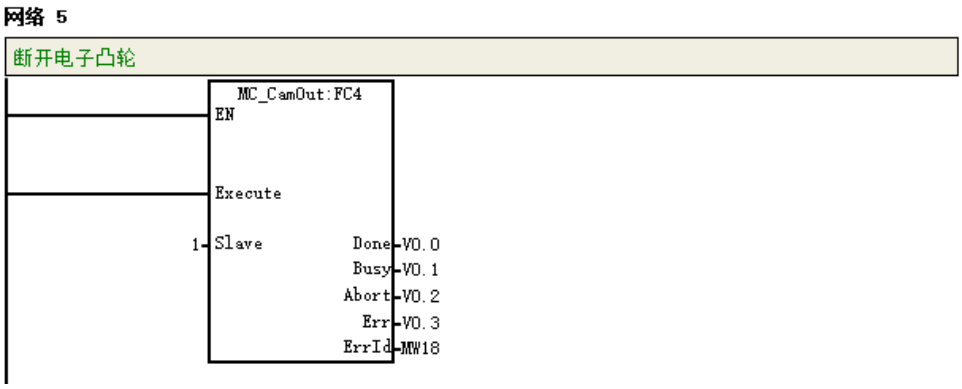

网络 3：建立主从轴间的凸轮关系，建立凸轮关系时，根据应用需求指定主从轴的偏移值，缩放比和启动模式。当指令执行完毕，从轴根据凸轮曲线跟随主轴运动。



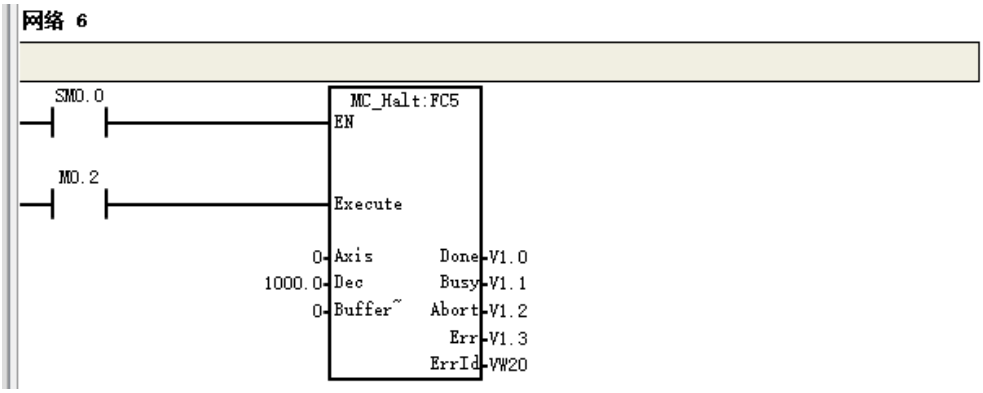

网络 4：读取相应轴状态。




网络 5：解除主从轴间的凸轮关系，关系解除后，从轴会以脱离前的速度继续运动。




网络 6：轴按给定的减速度减速，直到停止。




## 13.4 回原功能介绍

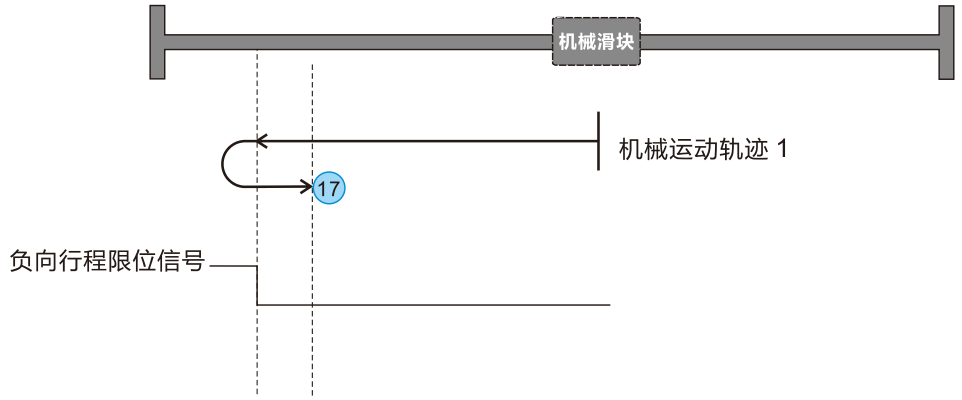
### 13.4.1 回原模式模式原理介绍

用户可根据各自身对精度的要求及实际应用需求从以下 17 种回原模式进行选择。

| 模式 | 说明 |
|---|---|
| 17 | 参考负向行程限位信号 |
| 18 | 参考正向行程限位信号 |
| 19 | 参考正向原点信号开关 |
| 20 | 参考正向原点信号开关 |
| 21 | 参考负向原点信号开关 |
| 22 | 参考负向原点信号开关 |
| 23 | 参考原点开关和正限位的原点模式 |
| 24 | 参考原点开关和正限位的原点模式 |
| 25 | 参考原点开关和正限位的原点模式 |
| 26 | 参考原点开关和正限位的原点模式 |
| 27 | 参考原点开关和负限位的原点模式 |
| 28 | 参考原点开关和负限位的原点模式 |
| 29 | 参考原点开关和负限位的原点模式 |
| 30 | 参考原点开关和负限位的原点模式 |
| 35 | 设置当前位置为原点 |

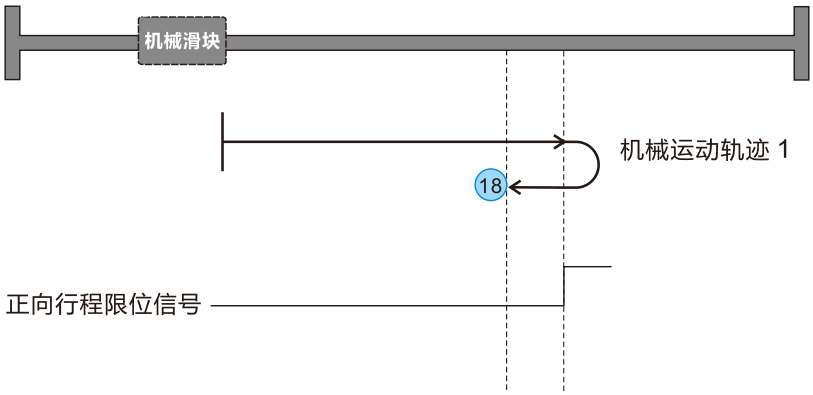
无论机械初始处于什么位置，当设备（原点开关、正向行程限位开关、负向行程限位开关）安装完好，伺服所寻找的设备原点总是唯一的。以下各模式示意图中的竖线“|”代表机械初始位置，圆圈“X”代表原点位置。

**回原模式 17：**参考负限位下降沿



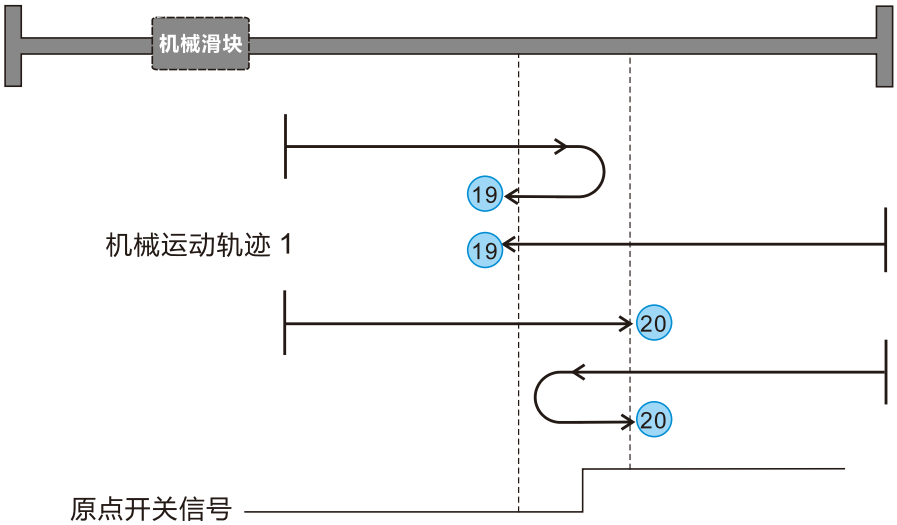

回原开始时，负向行程限位信号为 0，反向高速进行回原，遇到行程限位信号上升沿后进行减速，反向，进行正向低速运行，直到找到原点停机。

**回原模式 18：**参考正限位下降沿



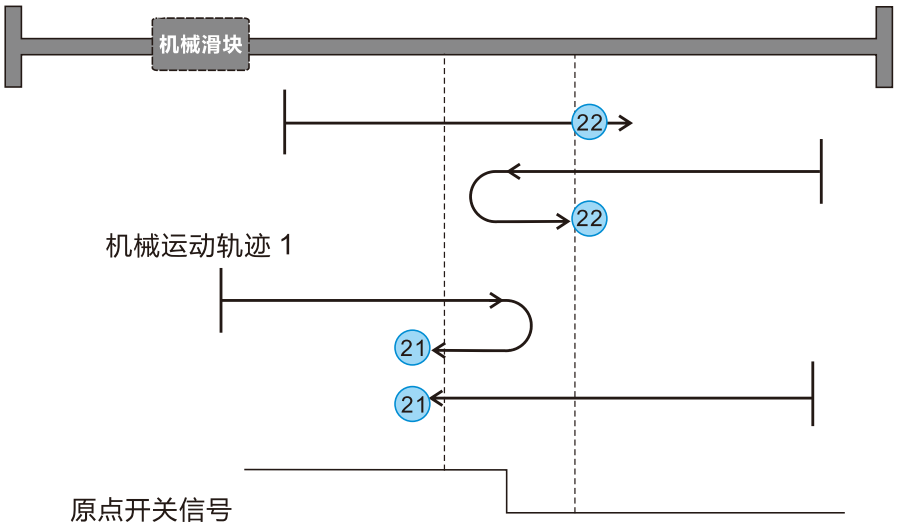

回原开始时，正向行程限位信号为 0，遇到行程限位信号上升沿后，减速并反向，进行低速运行，直到找到原点停机。

**回原模式 19、20：**参考正向原点信号开关



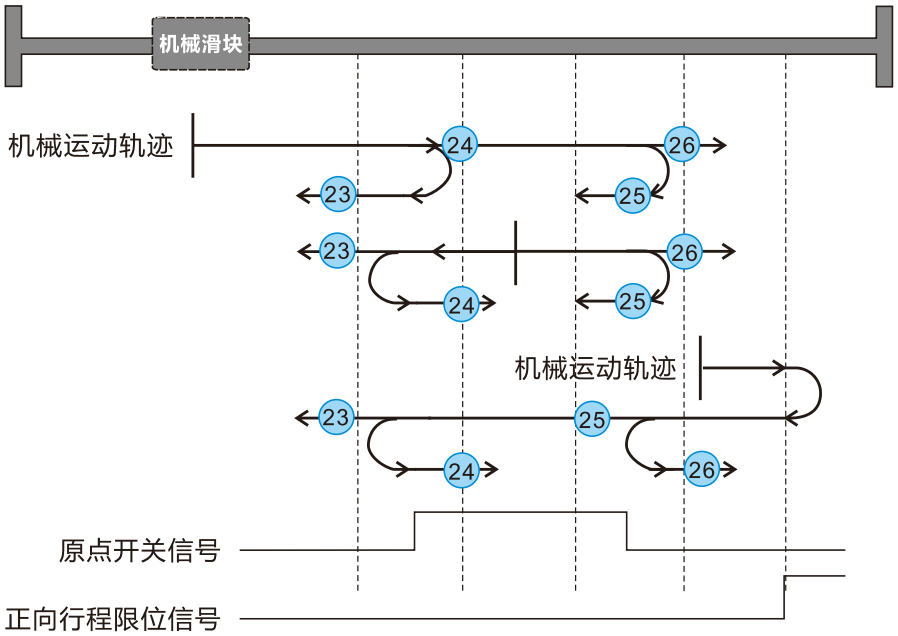

原点开关位于机械正方向。机械往原点开关方向运动，在检测到原点开关后减速停止，再反向低速运行，直到找到原点，电机立即停止。

**回原模式 21、22：**参考负向原点信号开关



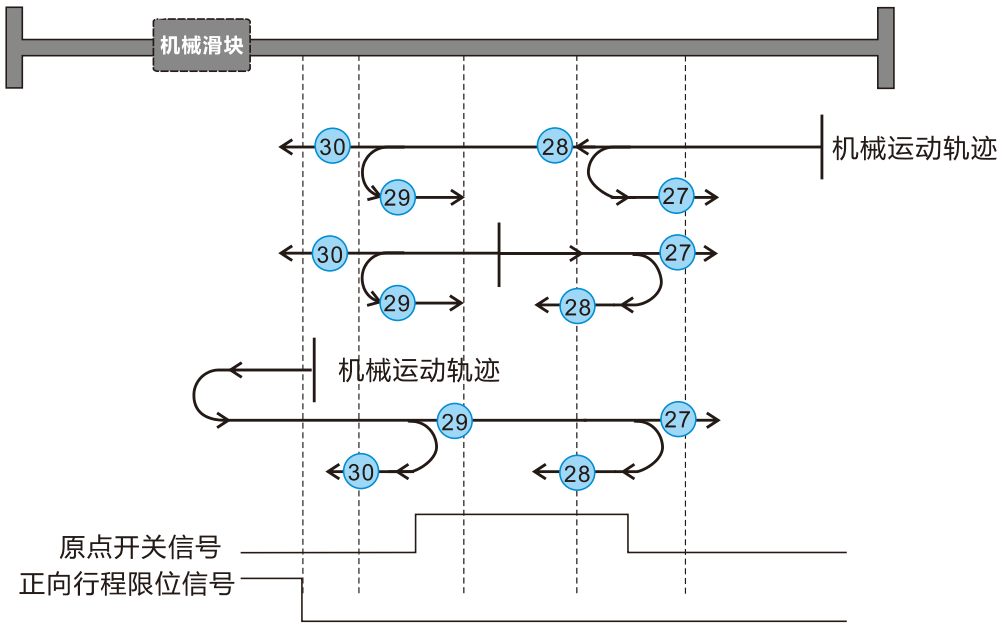

原点开关位于机械负方向。机械往原点开关方向运动，在检测到原点开关后减速停止，再反向低速运行，直到找到原点，电机立即停止。

**回原模式 23~26：**参考原点信号开关和正向行程限位



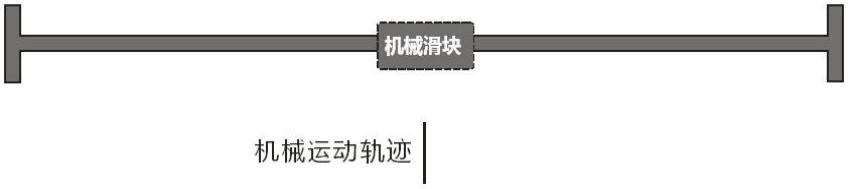

当原点定义在正向原点开关的左边沿时，如果机械滑块起始位置在的原点开关范围之外，初始运动方式是正向运动；如果起始位置在原点开关范围之内，向负方向运动，遇到原点开关下降沿时认为找到原点。当原点定义在正向原点开关的右边沿时，如果机械滑块起始位置在的原点开关范围之外，初始运动方式是正向运动；如果起始位置在原点开关范围之内，正方向运动，遇到原点开关下降沿时认为找到原点。

**回原模式 27~30：**参考负向原点信号开关和负向行程限位




当原点定义在正向原点开关的右边沿时，如果机械滑块起始位置在的原点开关范围之外，初始运动方式是负向运动；如果起始位置在原点开关范围之内，向正方向运动，遇到原点开关下降沿时认为找到原点。当原点定义在正向原点开关的左边沿时，如果机械滑块起始位置在的原点开关范围之外，初始运动方式是负向运动；如果起始位置在原点开关范围之内，向负方向运动；遇到原点开关下降沿时认为找到原点。

**回原模式 35：**以当前位置为原点




电机停在当前位置不动，给定回原信号即完成回原。

## 13.4.2 回原功能应用举例

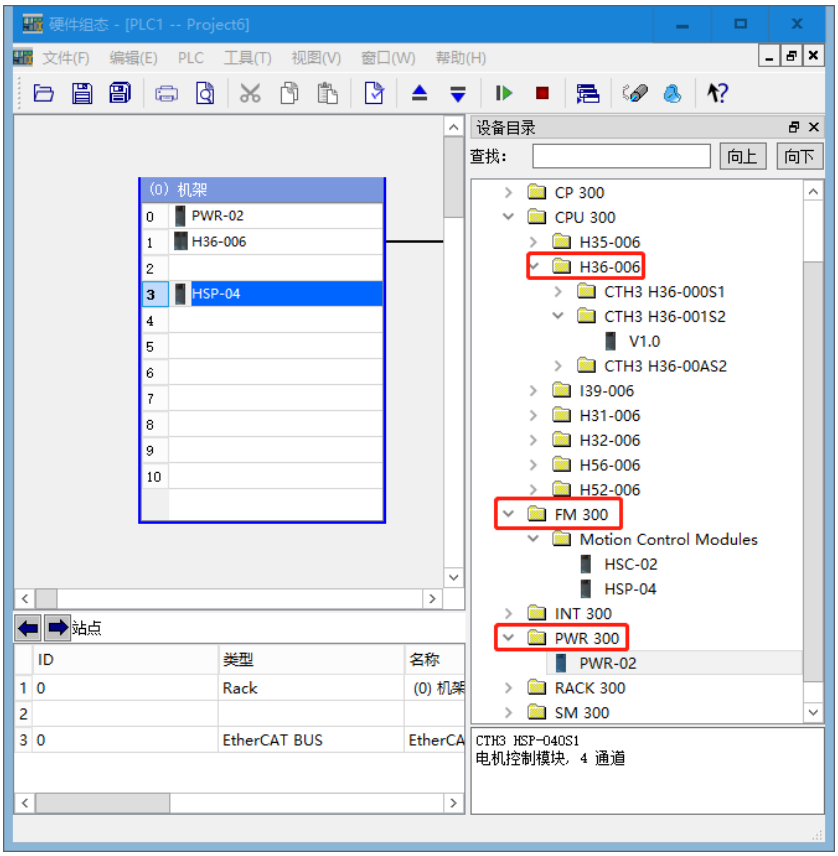
下面以 H36-006 CPU 为例对回原功能进行具体介绍。

**示例 1：脉冲轴原点回归示例**

在 H36-006 CPU 站点的项目管理器界面选择“硬件组态”，进行硬件组态。

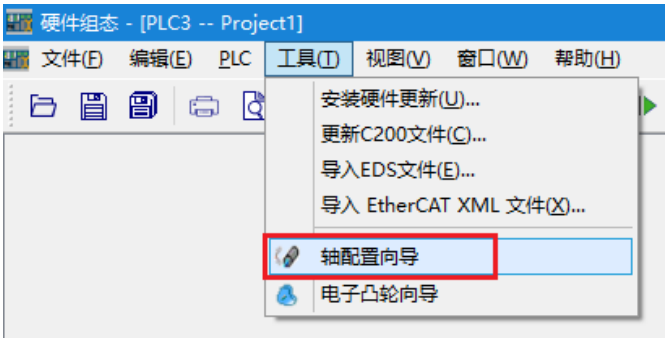
**步骤一：添加电源模块、CPU 和 HSP 模块**

硬件组态界面中，在“CO-TRUST 300”项目树下顺序展开“PWR 300”、“CPU 300”和“FM 300”文件夹，添加电源模块、CPU 和 HSP 模块，电源模块只能放至机架的 0 号槽内，CPU 只能放至机架的 1 号槽内，HSP 模块可放在 3-10 号卡槽内，如下图所示。

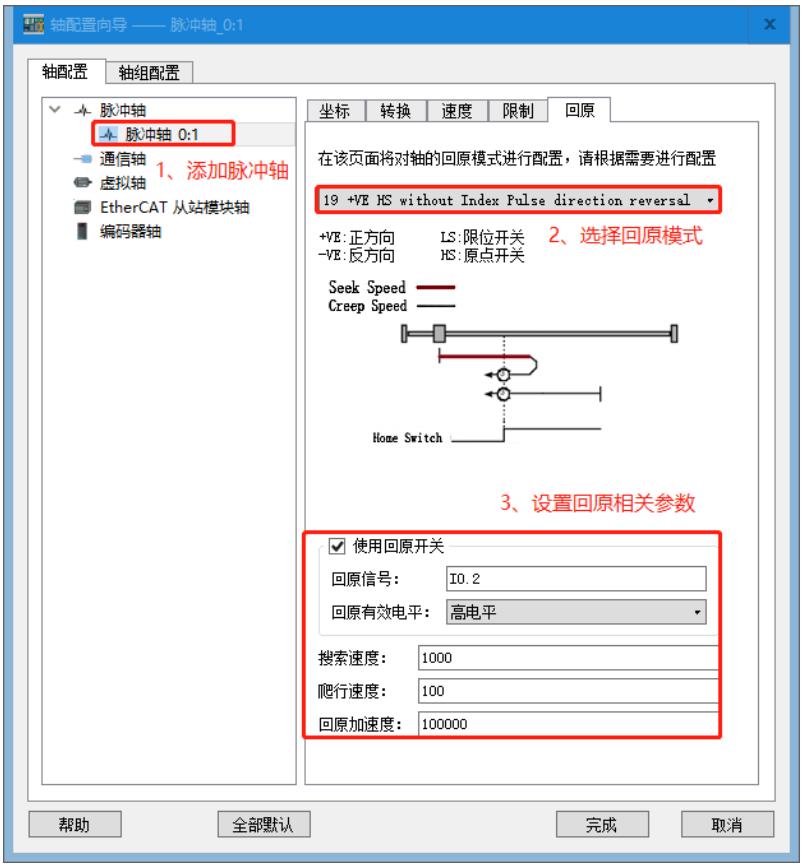



**步骤二：配置轴**

在硬件组态界面选择“工具”→“轴配置向导”，添加脉冲轴，即可进行轴配置，如下图所示。

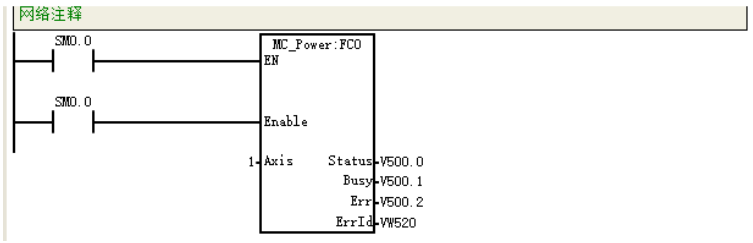



点击“回原”，在此页面可对轴的回原模式进行配置，此配置仅脉冲轴支持，选择下拉选项可查看回原模式，共有 17 种回原模式可选。

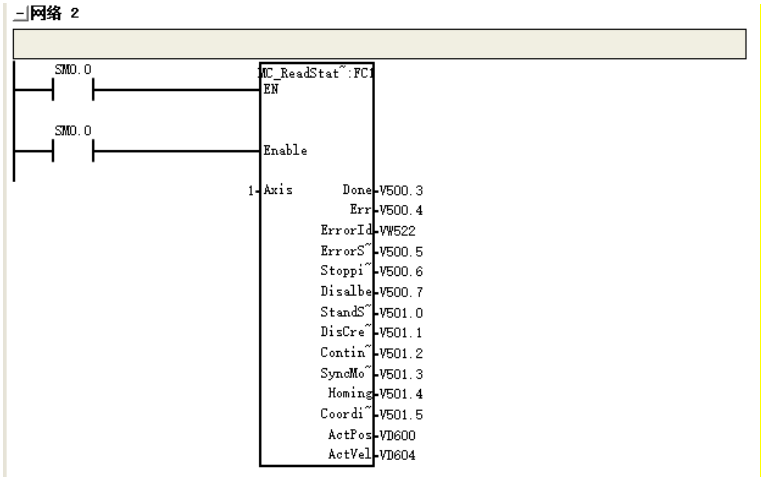



**步骤三：在主程序中调用 MC_Home 指令（原点回归指令）进行编程。**

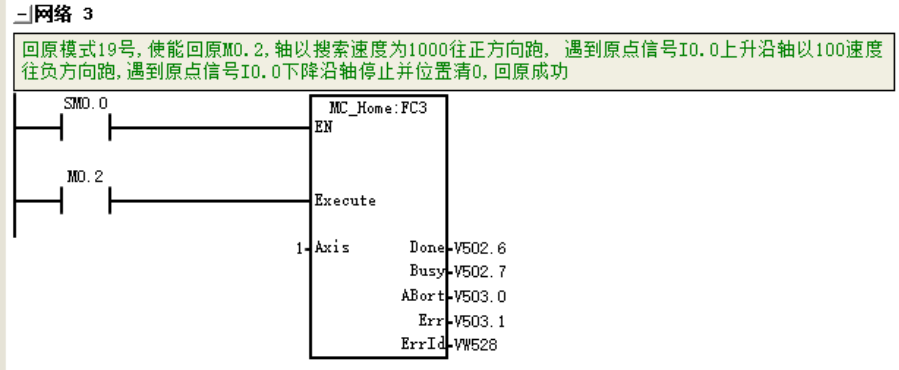
网络 1：对轴进行使能操作，轴 ID 号 Axis 设置为 1。




网络 2：读取轴状态



网络 3：回原模式为 19 号，回原信号为 I0.0，轴的搜索速度为 1000。

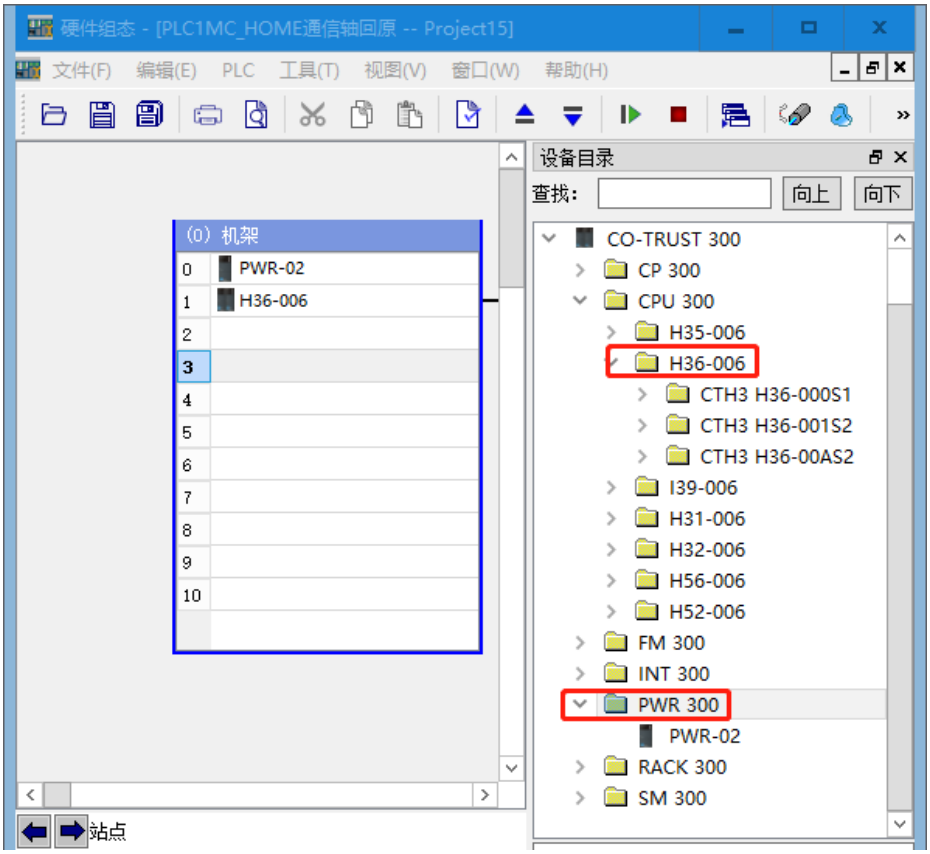



## 示例 2：通信轴原点回归示例

在 H36-006 CPU 站点的项目管理器界面选择“硬件组态”，进行硬件组态。

**步骤一：添加电源模块、CPU**

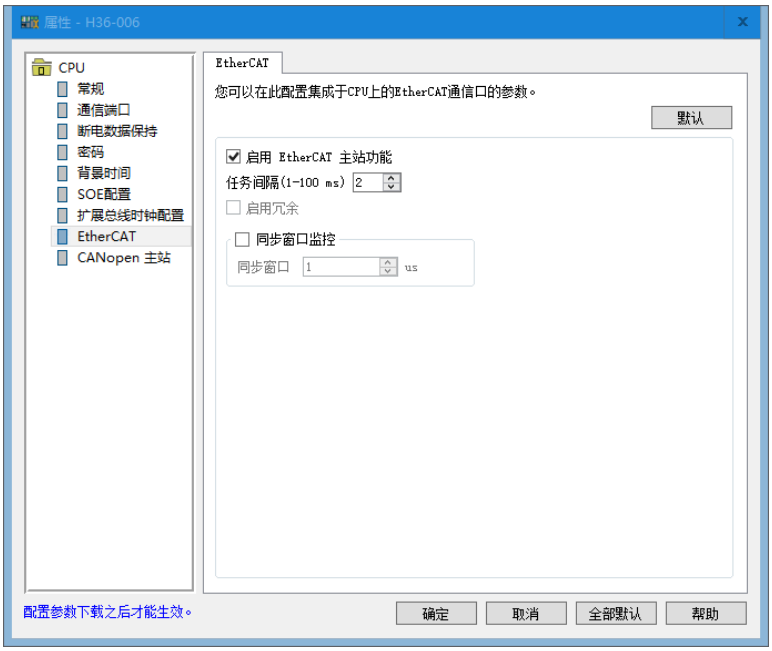
硬件组态界面中，在“CO-TRUST 300”项目树下顺序展开“PWR 300”和“CPU 300”文件夹，添加电源模块、CPU，电源模块只能放至机架的 0 号槽内，CPU 只能放至机架的 1 号槽内，如下图所示。



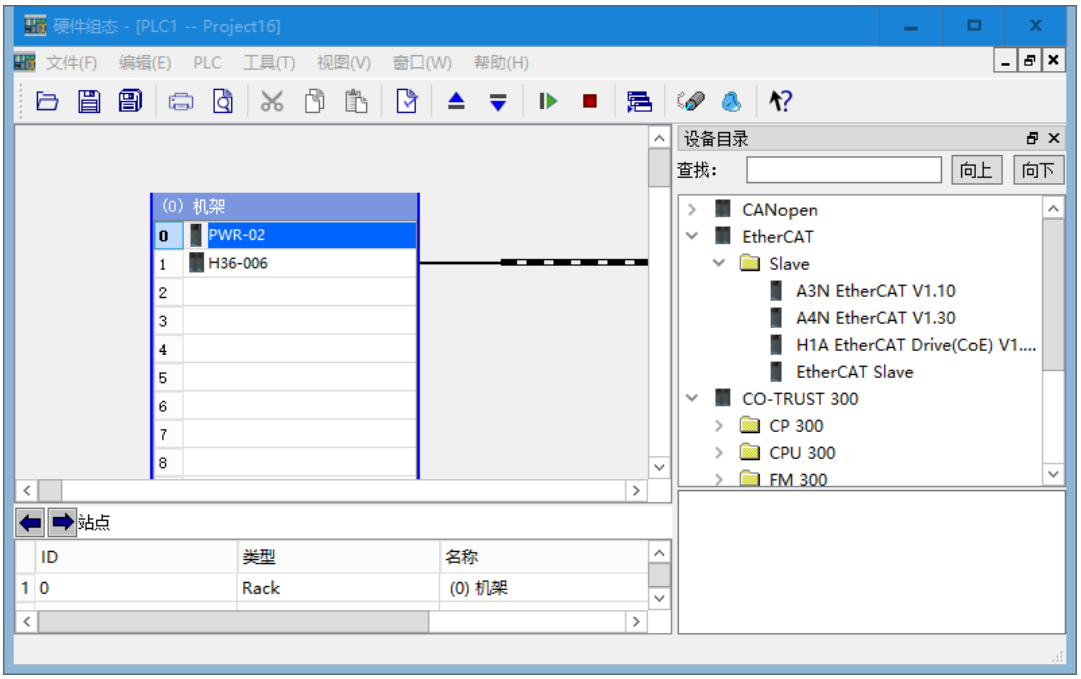

**步骤二：配置轴**

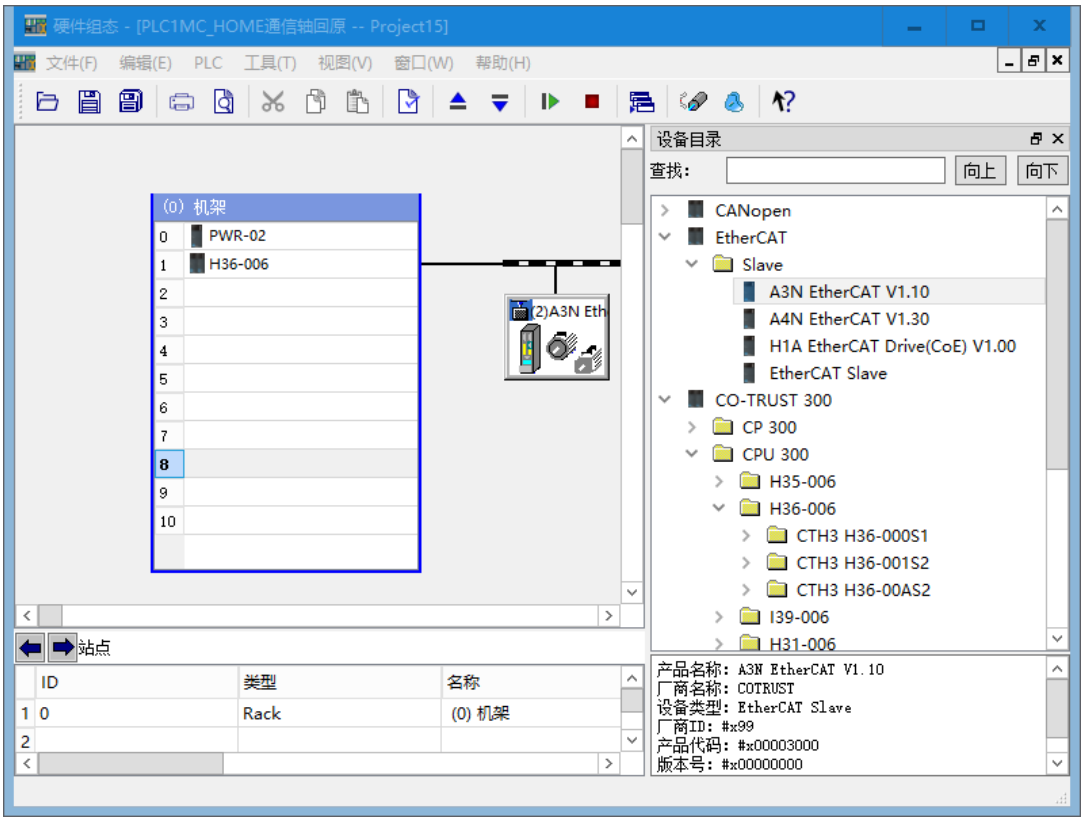
添加通信轴前需在硬件组态界面添加相应的 EtherCAT 从站，否则将无法添加通信轴。具体操作如下：

1）右击机架上的 H36，选择“属性”→ “EtherCAT”，点击页面中的“启用 EtherCAT 主站功能”。

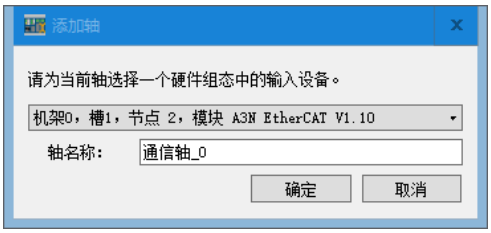



2）确定后可看到硬件组态界面的“EtherCAT BUS”，点击右边项目树下的“EtherCAT”，选择从站并将从站拖到“EtherCAT BUS”下。



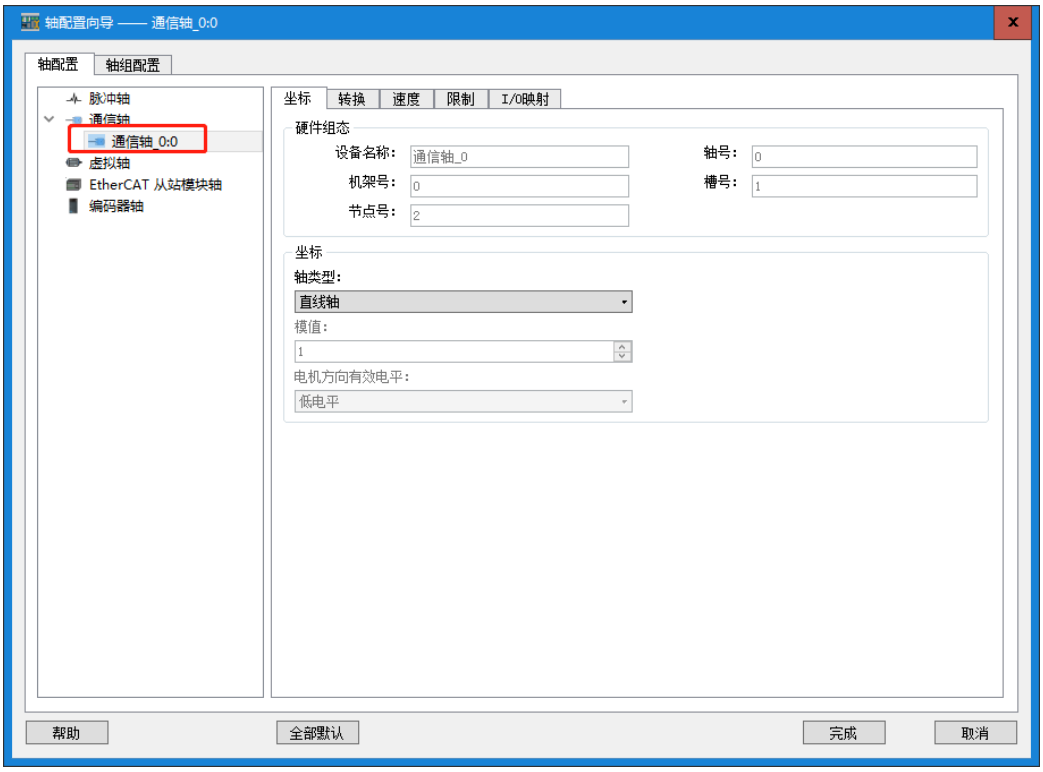



3）完成以上配置后在“工具”选项中选择“轴向导配置”，在轴向导配置界面双击“通信轴”，在弹出的“添加轴”界面中可为该轴选择输入设备，可选设备为硬件组态中所添加的从站。“轴名称”可修改，点击“确定”完成添加。



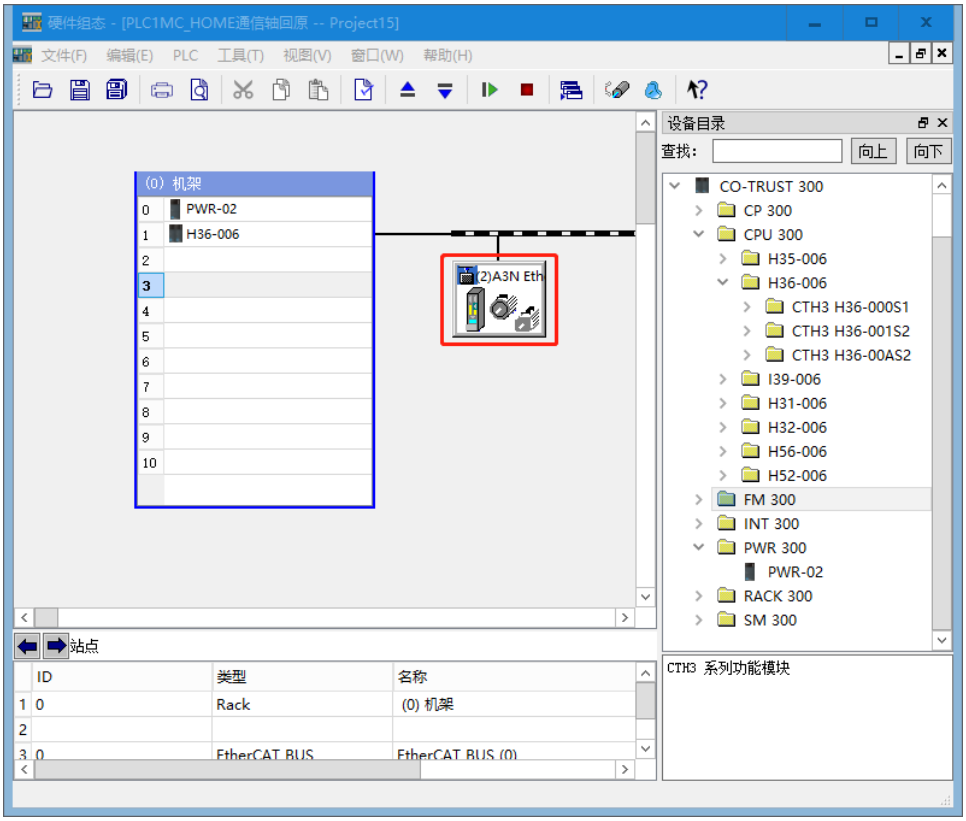

**注意：**一个从站只可对应添加一个通信轴。

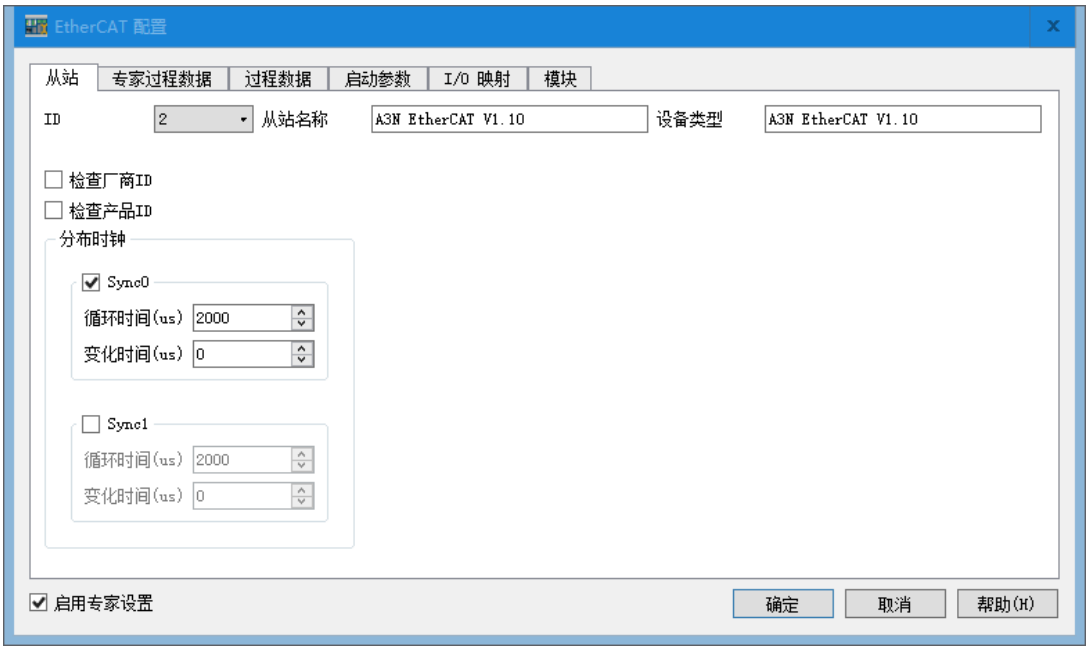
4）添加的通信轴界面如下所示：




5）使用驱动器回原（MC_Home 指令），在配置通信轴回原功能时，需配置 16#6098:0（回原模式）、16#6099:1（回原搜索速度）、16#6099:2（回原爬行速度）；操作如下：

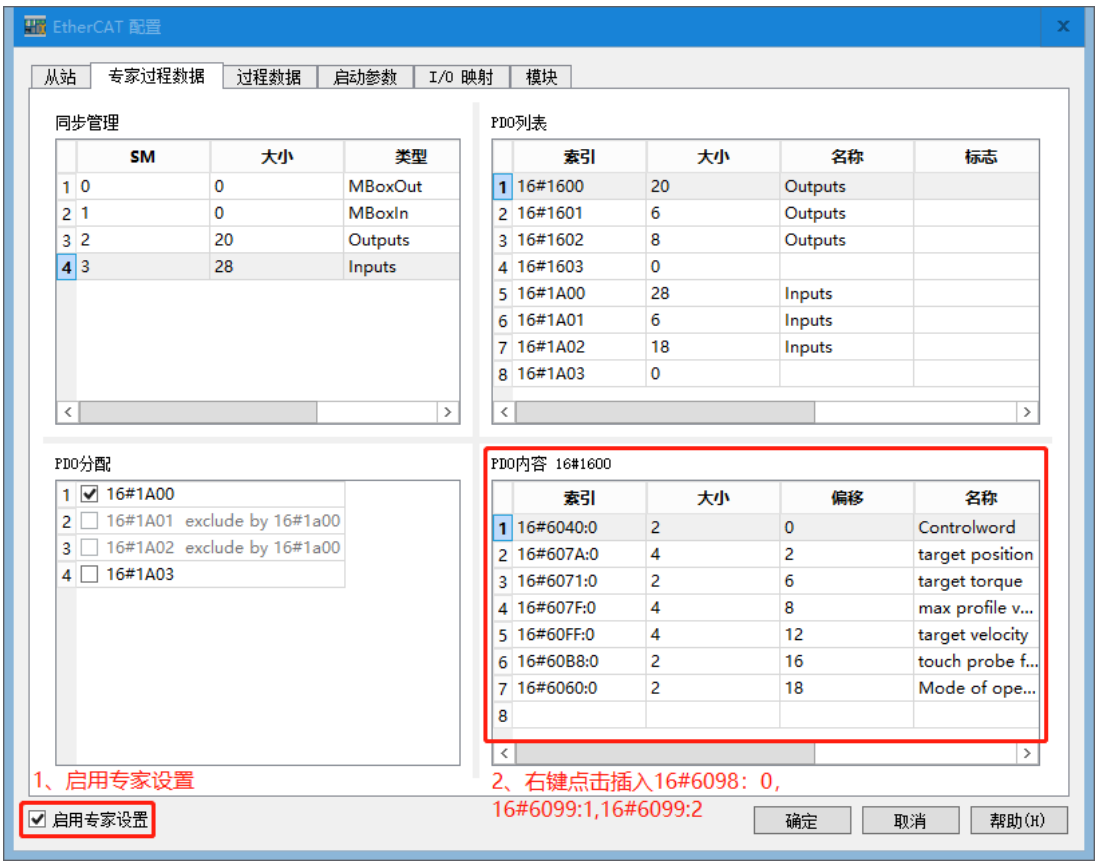
首先双击挂接在“EtherCAT BUS”下的从站，出现 EtherCAT 配置界面。





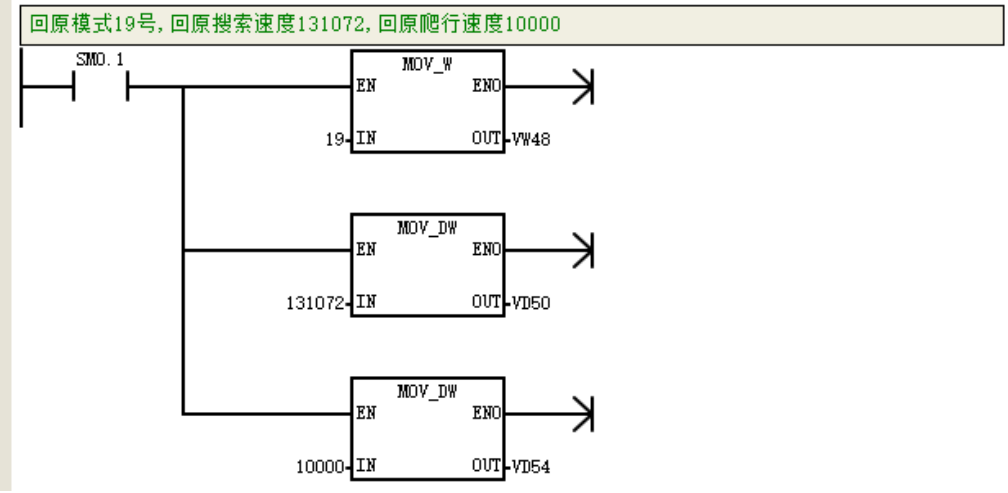

6）点击“启用专家设置”，右键点击后插入 16#6098:0（回原模式）、16#6099:1（回原搜索速度）、16#6099:2（回原爬行速度）。

**<备注>** 回原爬搜索速度与爬行速度的指令单位有两种，分别为：pps(脉冲数/每秒)、rpm(转数/每分)请参阅驱动器厂商说明书，具体请查阅驱动器厂商说明书。

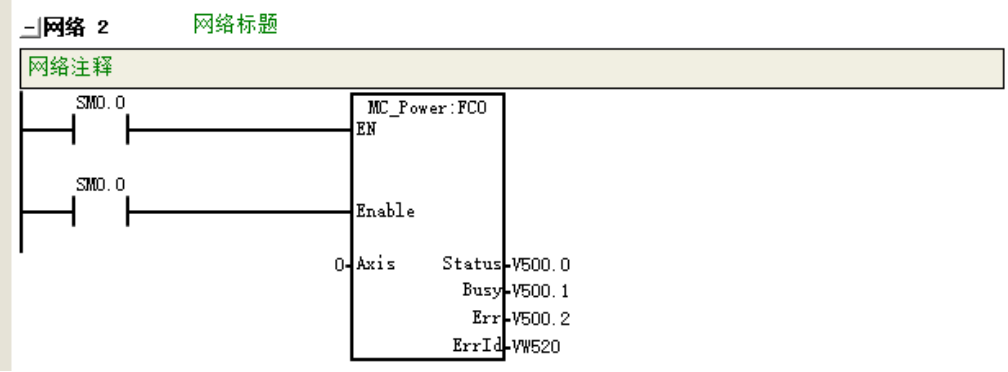


**步骤三：在主程序中调用 MC_Home 指令（原点回归指令）进行编程。**

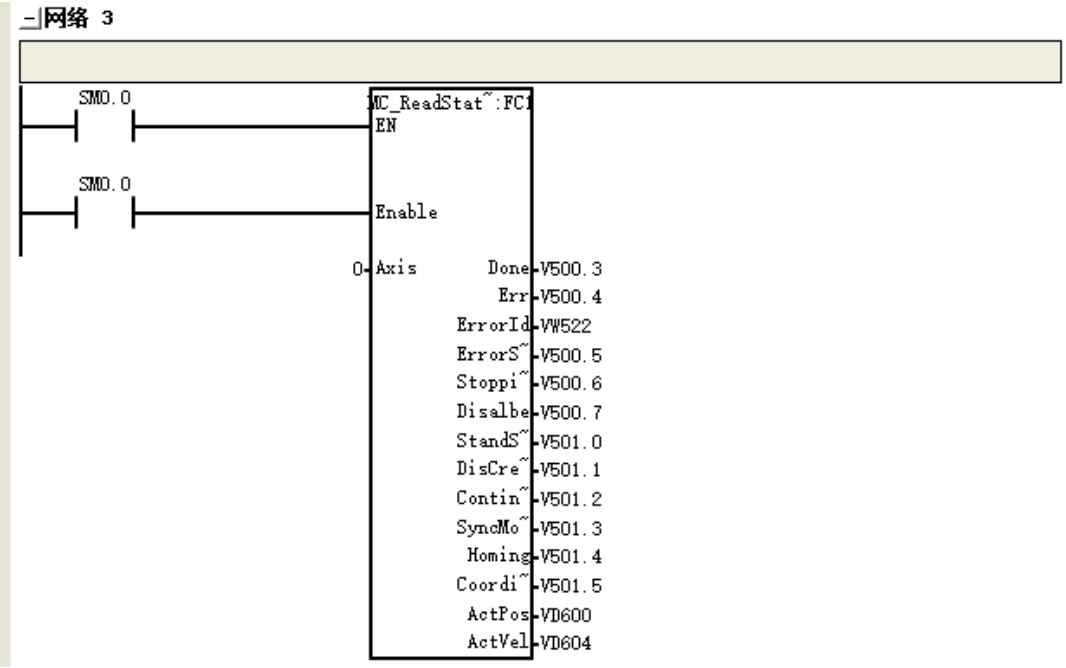
网络 1：回原模式为 19 号，回原搜索速度为 131072，回原爬行速度为 1000。




网络 2：使轴进行使能操作。

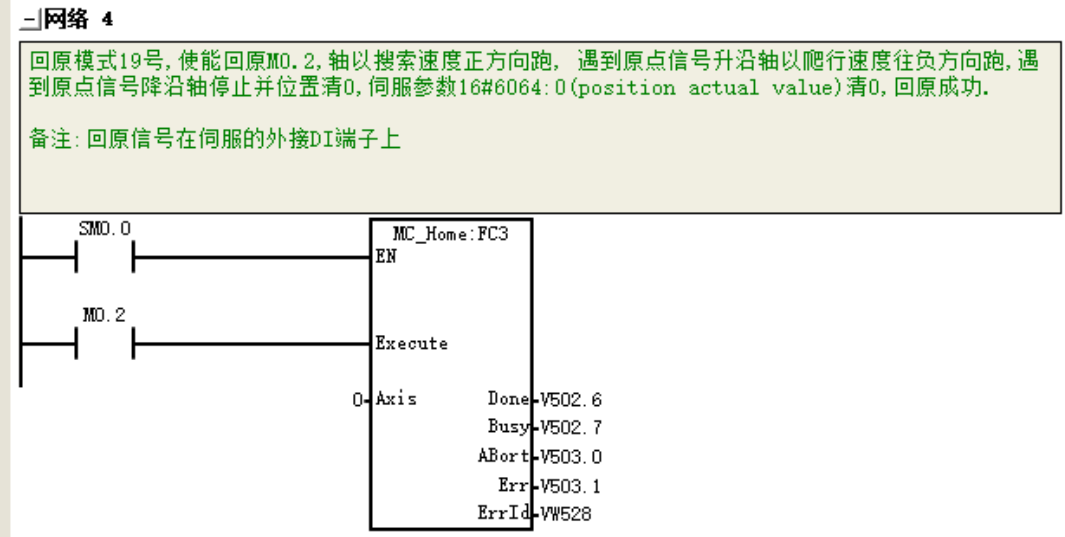



网络 3：读取轴状态



回原模式为 19 号，回原信号 I0.2，轴以搜索速度正方向跑，遇到原点信号升沿轴以爬行速度往负方向跑，遇到原点信号降沿轴停止并位置清 0，伺服参数 16#6064:0(position actual value)清 0。

备注：回原信号在伺服的外接 DI 端子上。

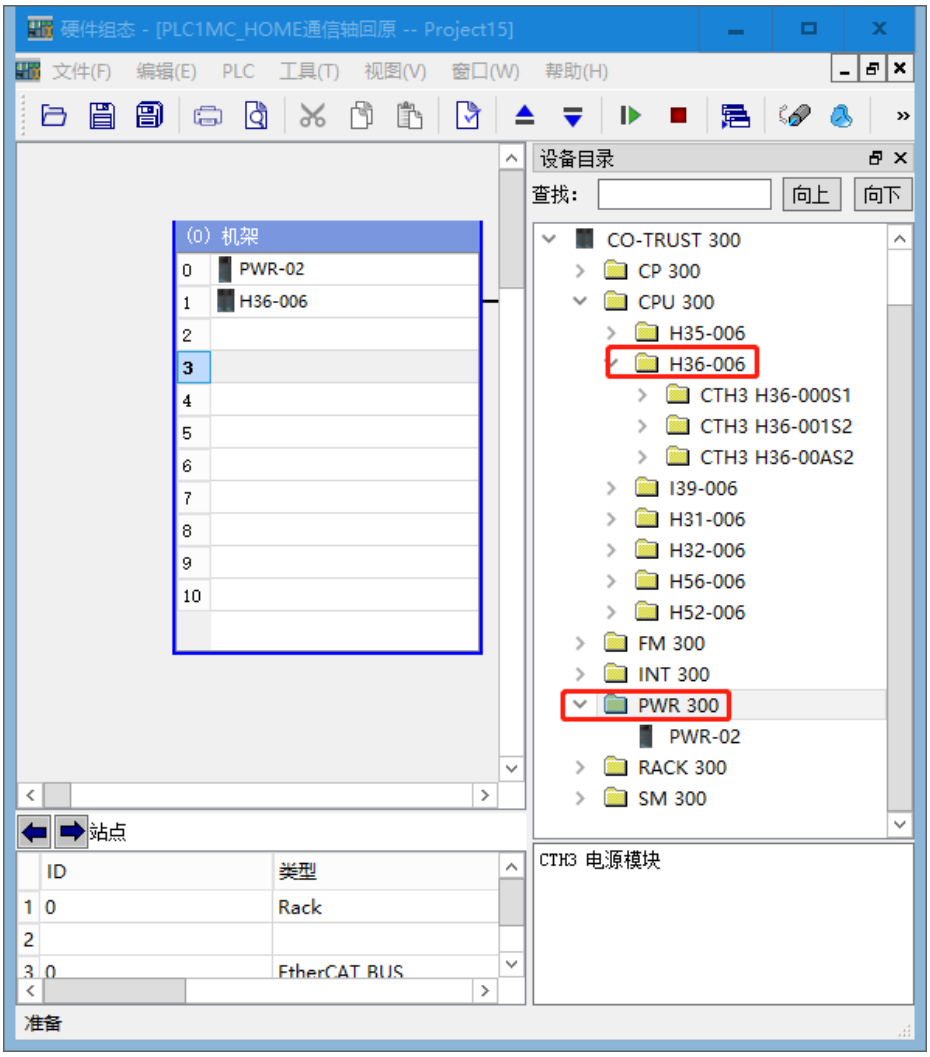



**示例 3：通信轴特殊原点回归示例**

在 H36-006 CPU 站点的项目管理器界面选择“硬件组态”，进行硬件组态。

**步骤一：添加电源模块、CPU**

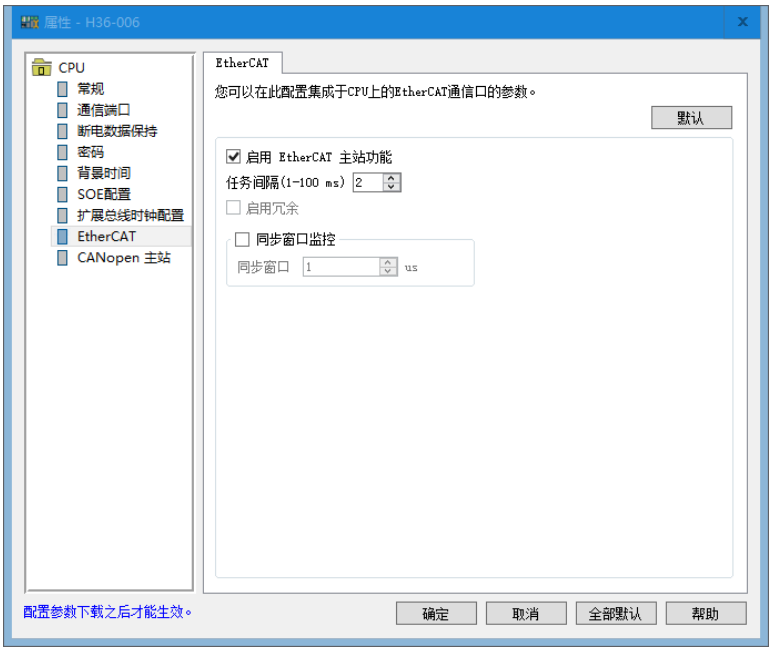
硬件组态界面中，在“CO-TRUST 300”项目树下顺序展开“PWR 300”和“CPU 300”文件夹，添加电源模块、CPU，电源模块只能放至机架的 0 号槽内，CPU 只能放至机架的 1 号槽内，如下图所示。



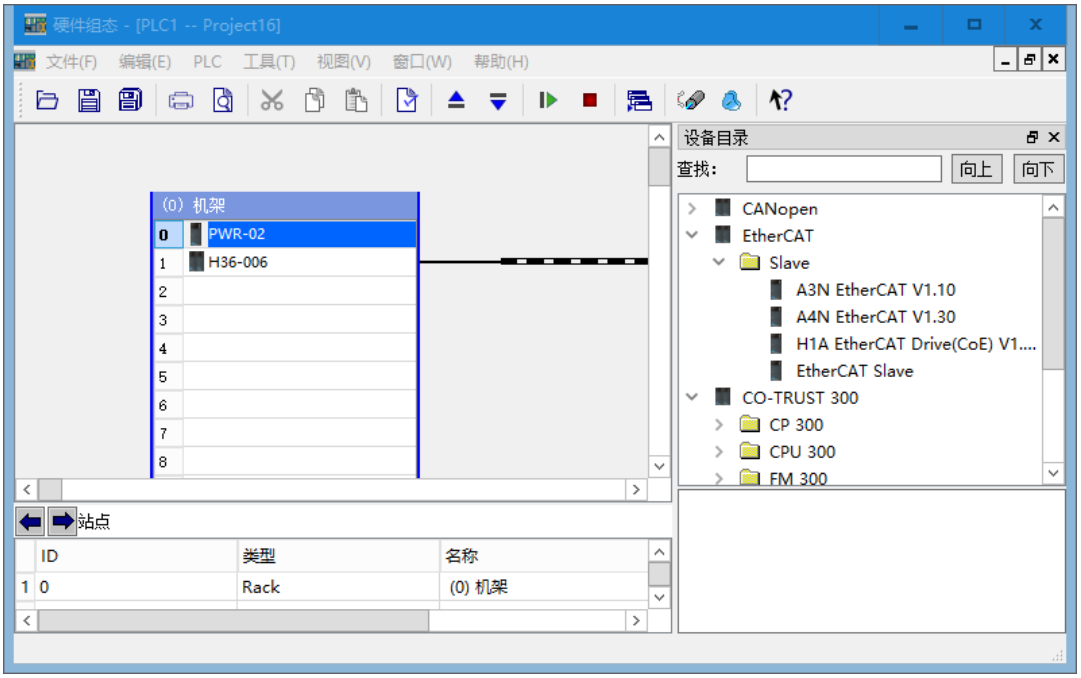

**步骤二：配置轴**

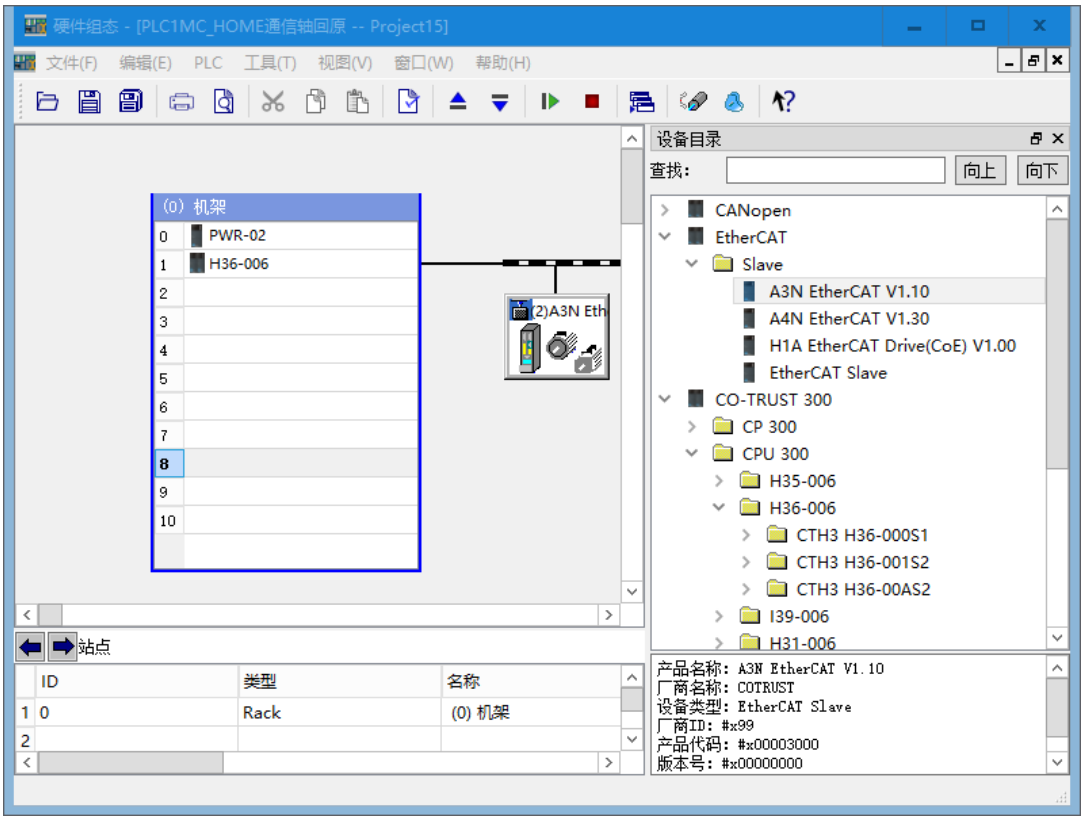
添加通信轴前需在硬件组态界面添加相应的 EtherCAT 从站，否则将无法添加通信轴。具体操作如下：

1）右击机架上的 H36，选择“属性”→“EtherCAT”，点击页面中的“启用 EtherCAT 主站功能”。

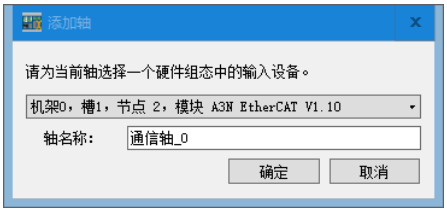



2）确定后可看到硬件组态界面的“EtherCAT BUS”，点击右边“EtherCAT”，选择从站并将从站拖到“EtherCAT BUS”下。



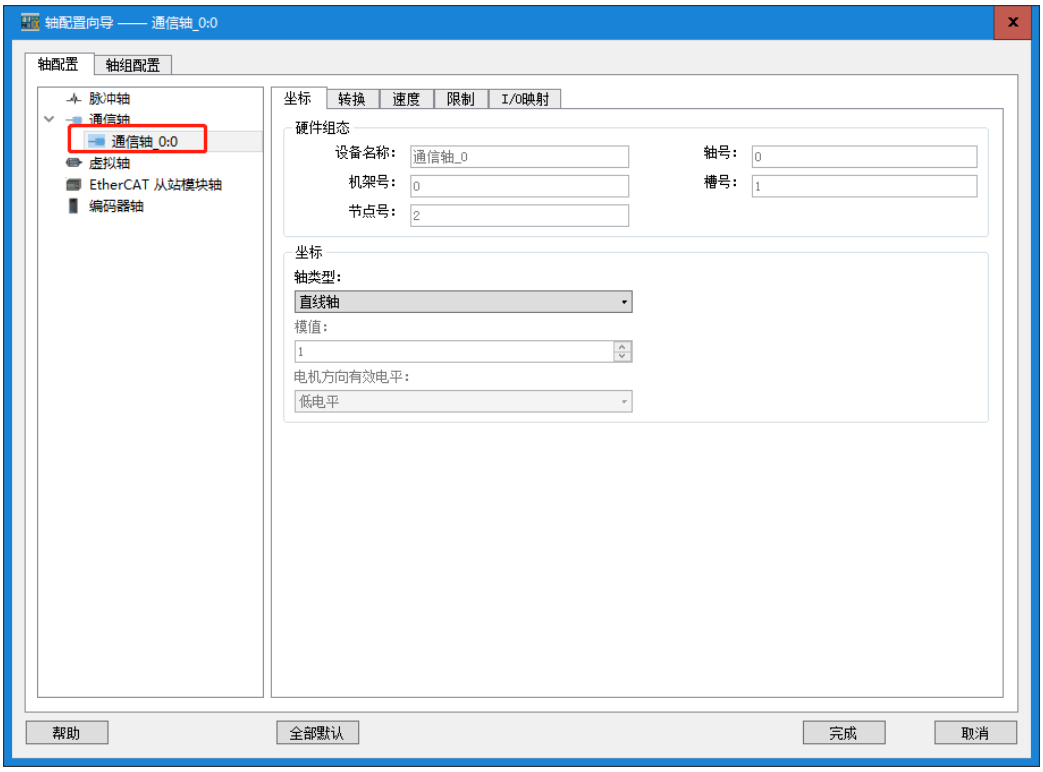



3）完成以上配置后在“工具”选项中选择“轴向导配置”，在轴向导配置界面双击“通信轴”即可添加通信轴，“轴名称”可修改，点击“确定”完成添加。



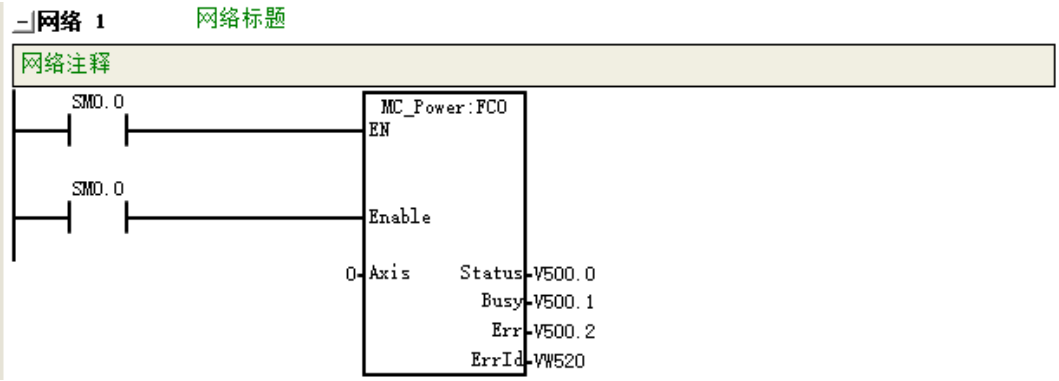

**注意：**一个从站只可对应添加一个通信轴。

4）添加的通信轴界面如下所示：

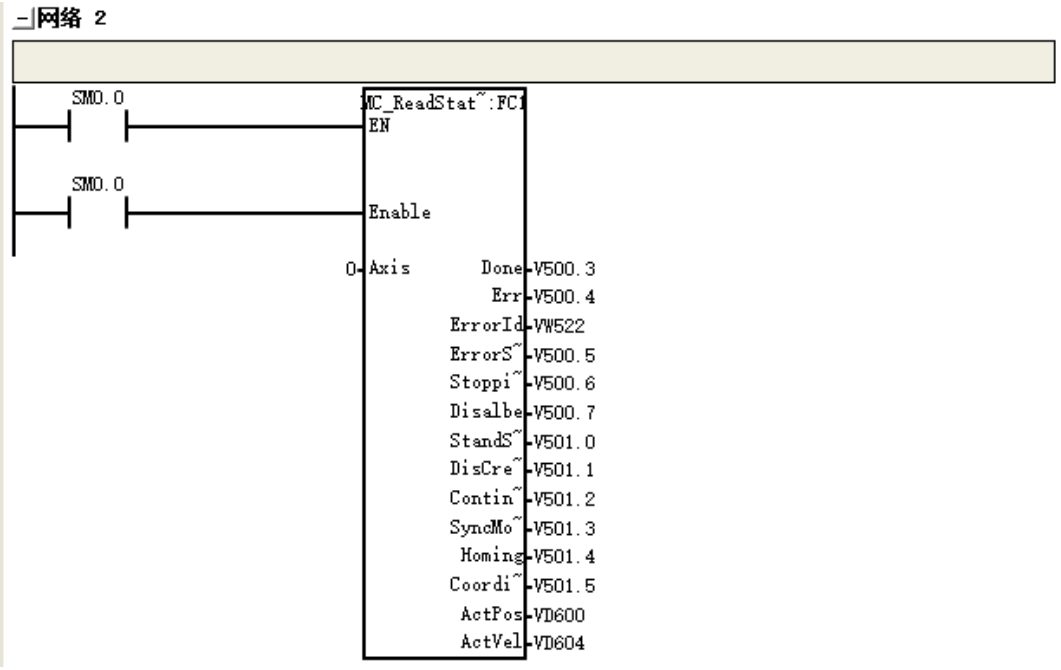



**步骤三：在主程序中调用 SMC_Home 指令（特殊原点回归指令）进行编程。**

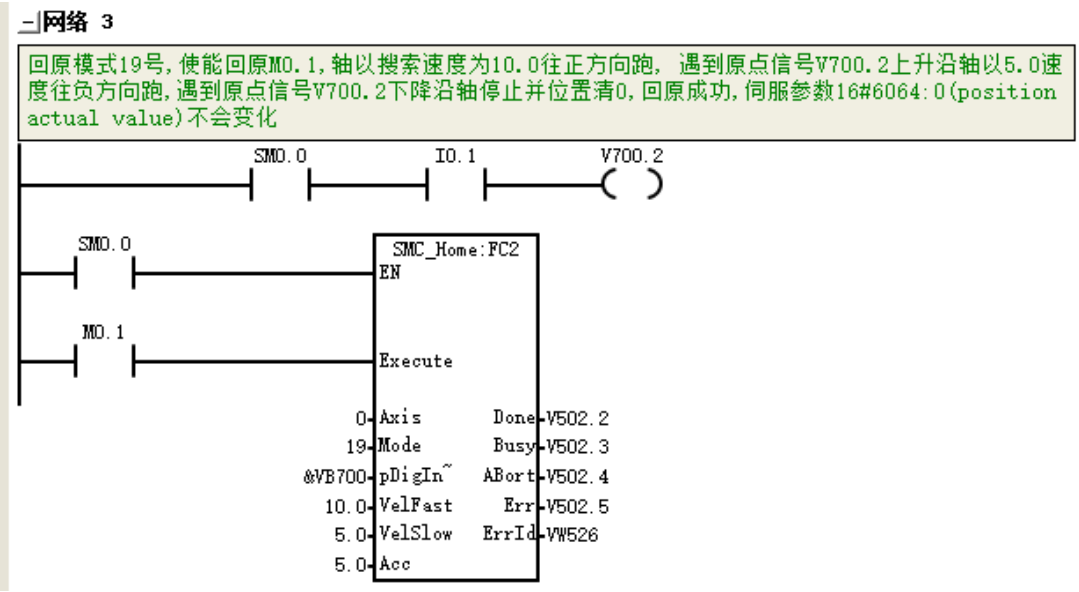
网络 1：使轴进行使能操作，轴 ID 号 Axis 设置为 0。




网络 2：读取轴状态。




网络 3：调用 SMC_Home 特殊原点回归指令，选用回原模式 19，轴的搜索速度为 10.0，加速度为 5。



# 14 使用指南

本章将通过一些具体实例展示 CTH300-H 系列可编程逻辑控制器的多种功能应用。

### 本章概要

本章主要介绍 CTH300-H 系列可编程控制器的使用示例，具体内容如下：

- ❑ 通信功能应用示例
- ❑ 为主控模块的 RS485 接口选择通信协议
- ❑ 高速计数模块应用示例
- ❑ HSC 中断示例
- ❑ 故障诊断

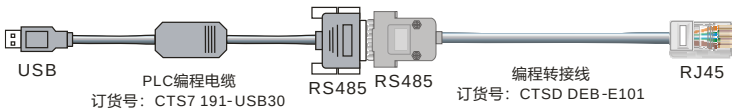
## 14.1 通信功能应用示例

本节将通过一个具体实例展示 H35-00 的总线通信功能即如何通过总线接口传送数据。

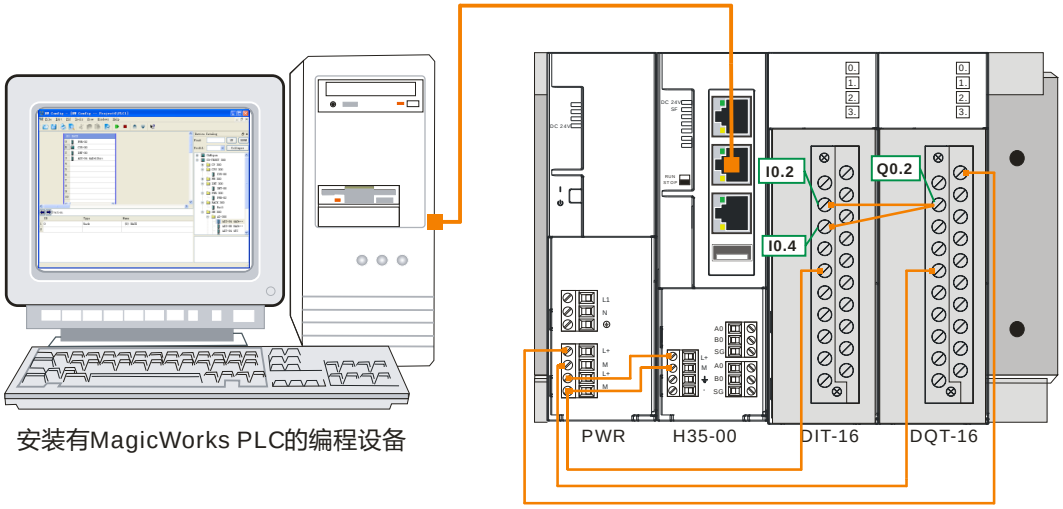
### 14.1.1 总线通信

**1、示例组件**

表 14-1 总线通信的示例组件

| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 进行组态、编程和调试 |
| 装配导轨 | CTH300-H 机架，用于固定系统中的各模块 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电 |
| H35-00 主控模块 | CPU 执行用户程序，向 CTH300-H 背板总线提供 5V 电压，并通过 RS485 接口与其它模块进行 PPI 通讯 |
| CTH300-H 扩展模块 | SM 转换不同的过程信号电平，使其与 CTH300-H 相匹配。<br>本示例中使用数字量输入模块 DIT-16、数字量输出模块 DQT-16 |
| 编程电缆 | 将CPU连接到PC/PG：PLC编程电缆 + 编程转接线<br> |

**2、网络连接**




安装有MagicWorks PLC的编程设备

**先决条件**

- 拥有一个由电源模块和 H35-00 组成的 CTH300 系统。串行连接将通过 H35-00 的 RS485 通信口（PORT1）实现。
- 编程设备上已经正确安装 MagicWorks PLC 软件（V2.00 或更高版本）。
- 编程设备已经与 H35-00 正确连接。
- 已经准备好要进行串行数据传送的连接对象，并使用了必需的电缆进行连接。
- CPU 已经正确连接到电源模块。

**实现功能**

点亮输出点 Q0.2，输入点 I0.2 和 I0.4 同时被点亮。

## 3、操作步骤

**步骤 1：安装导轨和模块**

1）各模块安装顺序（从左到右）：PWR-02→H35-00→DIT-16→DQT-16

2）固定导轨的安装和接地

用螺钉将固定导轨安装在底板上，使固定导轨的上下边缘保留至少 40MM 的间隙。然后，将固定导轨连接到保护性导体上。

3）将模块安装到导轨上

将电源模块挂接到导轨上，滑动到固定导轨的接地螺钉位置，然后用螺钉将其固定。依次将 H35-00、数字量模块 DIT-16、DQT-16 挂到导轨上，滑动致其碰到左侧的模块，然后在底部用力将其推入，最后使用螺钉将所有模块固定。

**步骤 2：接线**

打开 H35-00、电源模块、数字量输入（DIT-16）和输出模块（DQT-16）的前面板，然后参考如下连线：

1）使用编程电缆连接 PC 与 H35-00

2）DIT-16 通过总线方式连接至 H35-00

3）DIQ-16 通过总线方式连接至 DQT-16

**警告**

电源模块上电后或将电源电缆连接到主电源后不得触摸通电电缆，任何接线工作须在断电情况下进行！

**步骤 3：设置通信**

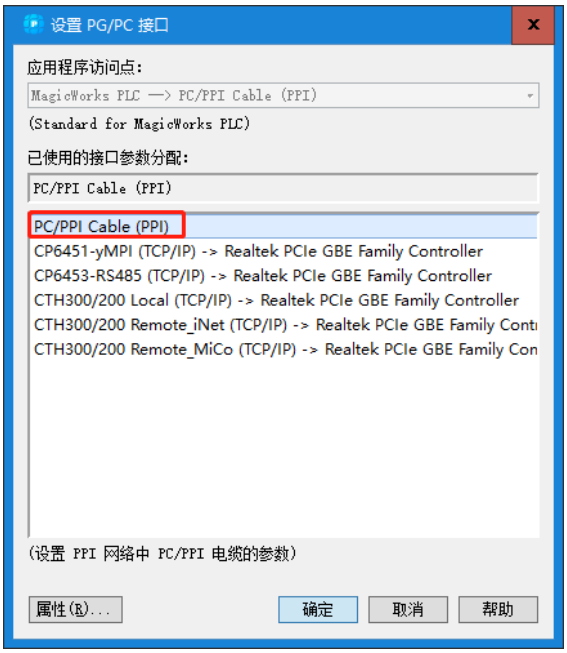
1）创建新项目

双击桌面图标打开 MagicWorks PLC 软件，然后选择菜单项“文件”→“新建”→输入项目名称→点击“确定”按钮完成新项目的创建。

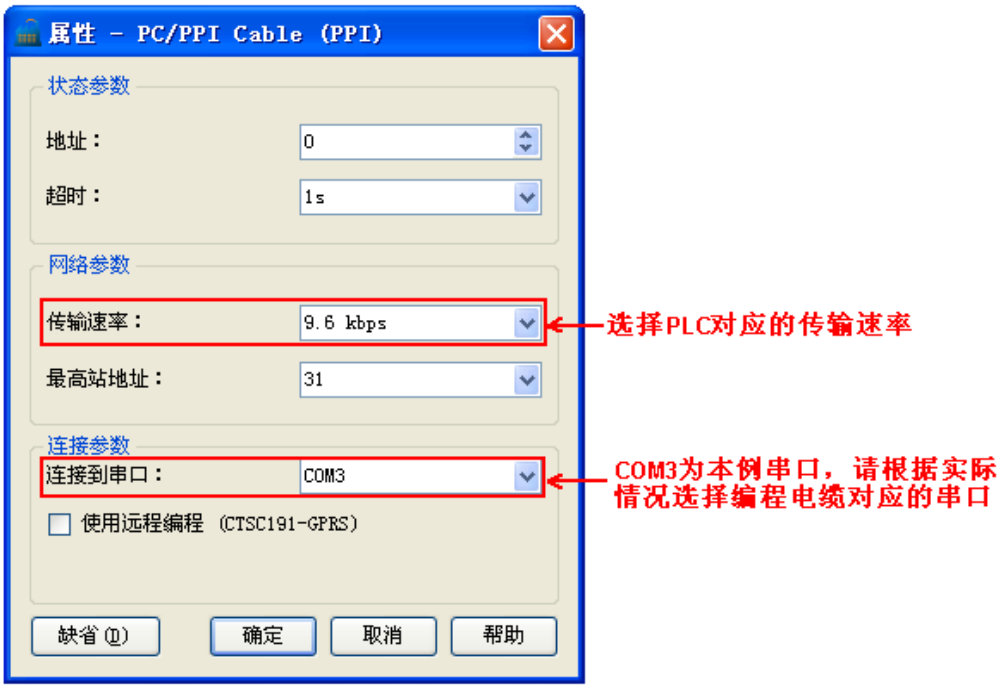
2）插入 PLC 站

在项目视图右键点击站点选择“插入新对象”→“PLC 站”→“H35-00”。

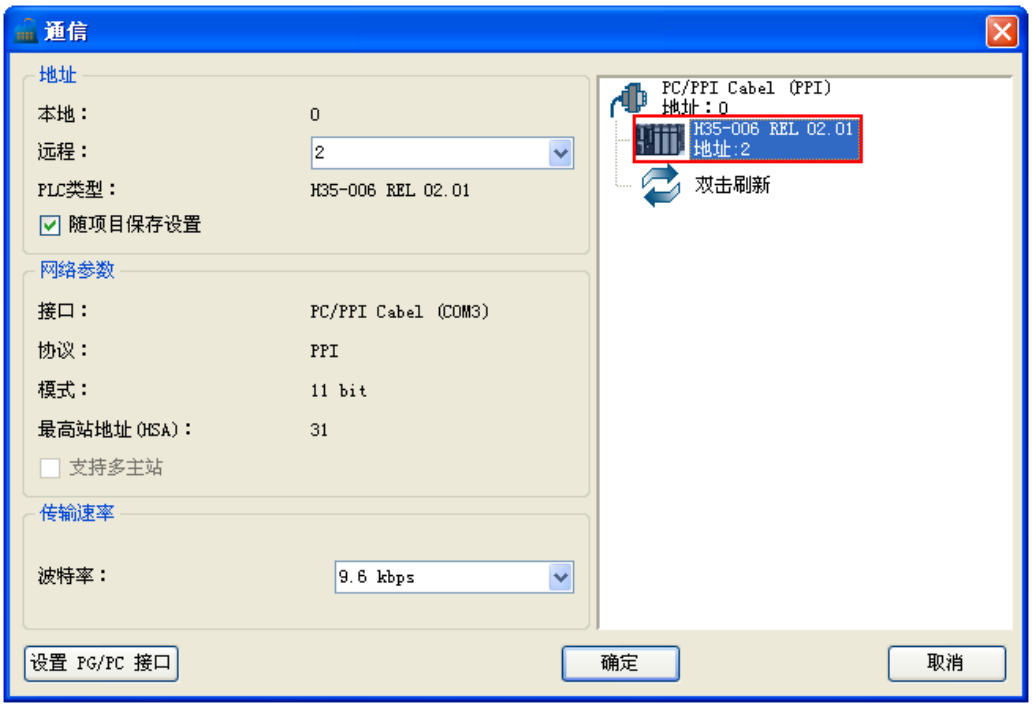
3）在 MagicWorks PLC 主界面，双击“设置 PG/PC 接口”或选择菜单项“工具”→“设置 PG/PC 接口”弹出如下窗口，选择“PC/PPI Cable(PPI)”，再点击“属性”按钮。




在如下“属性”配置界面按照硬件组态中 PPI 端口配置以上窗口中的参数，配置完成后点击“确定”完成配置。




4）在主界面的工作窗口双击图标打开通信对话框，在对话框右侧双击图标进行搜索，搜索到的 PLC 显示在通信对话框中，即表示搜索成功。




**步骤 4：在 MagicWorks PLC 中进行硬件组态**

1）进入硬件组态界面

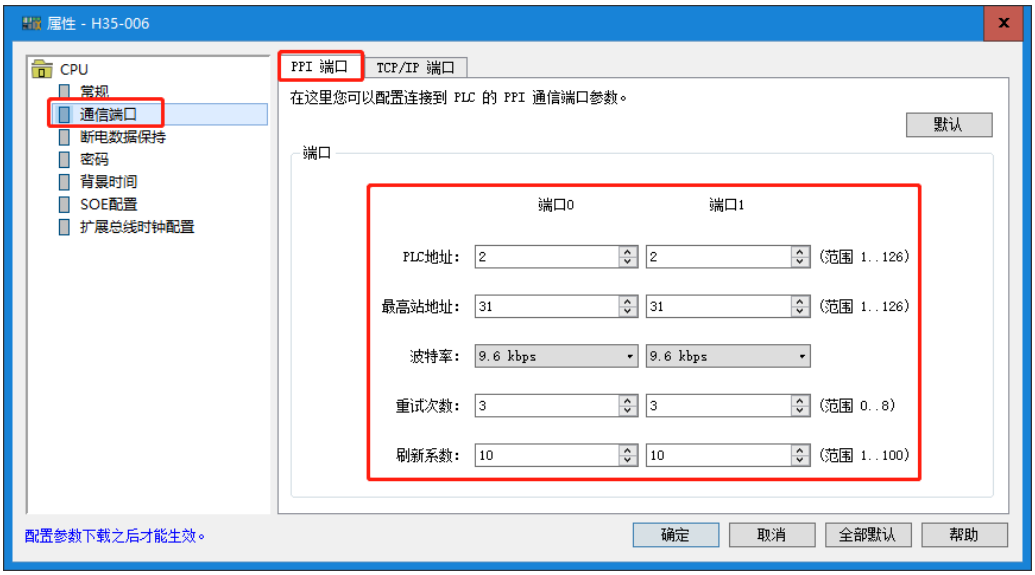
在项目管理器的项目树中选中 H35-00 站点，然后双击右侧工作窗口的硬件组态图标，进入硬件组态界面。

2）添加电源

在硬件组态界面，选中机架 Rack 0，然后展开设备目录的 CO-TRUST 300 节点选择 PWR-300→PWR-02，并将其拖放到机架（Rack 0）的插槽 0 中。

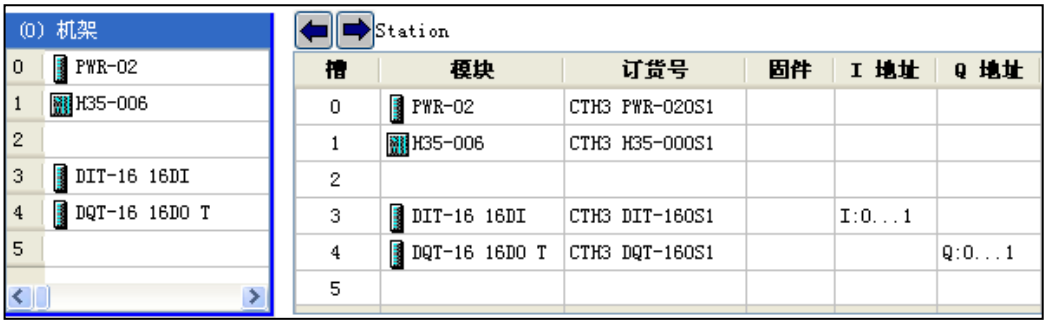
3）添加 CPU 并为其配置 PPI 通信

从设备目录中展开 CPU300→H35-00，并将其拖放到导轨插槽 1 中。然后，双击插槽 1 中的 H35-00，弹出如下窗口：




4）添加模块

从硬件目录中展开“SM 300”→“DI-300”，将 DIT-16 插入到机架插槽 3 中。然后展开 DO-300，将 DQT-16 插入到机架插槽 4 中。




5）保存并编译组态

选择菜单项“文件”→“保存”以保存当前组态，然后选择菜单项“PLC”→“编译”对当前工程进行编译。

**步骤 5：调试**

1）选择菜单项“PLC”→“下载”将程序块和硬件组态从编程设备下载到 H35-00 中。若您的 H35-00 处于运行模式，下载界面将弹出一个对话框提示您将 H35-00 置于 STOP 模式，点击“确定”将 H35-00 置于 STOP 模式。

2）下载完成后，将 H35-00 的系统运行开关拨到 RUN。

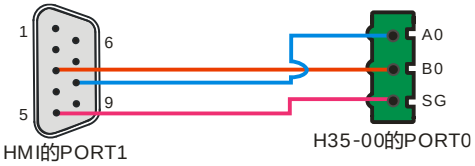
3）在 MagicWorks PLC 的项目窗口打开状态表，在状态表地址栏填写 Q0.2，并在其新值列写入 1，数字量模块 DQT 的 Q0.2 即被点亮。同时，输入点 I0.2 和 I0.4 被点亮。

**<备注>** 关于 CTH300-H 系列主控模块及扩展模块的诊断方法请参考 14.5 故障诊断。

## 14.1.2 PPI/MPI 通信

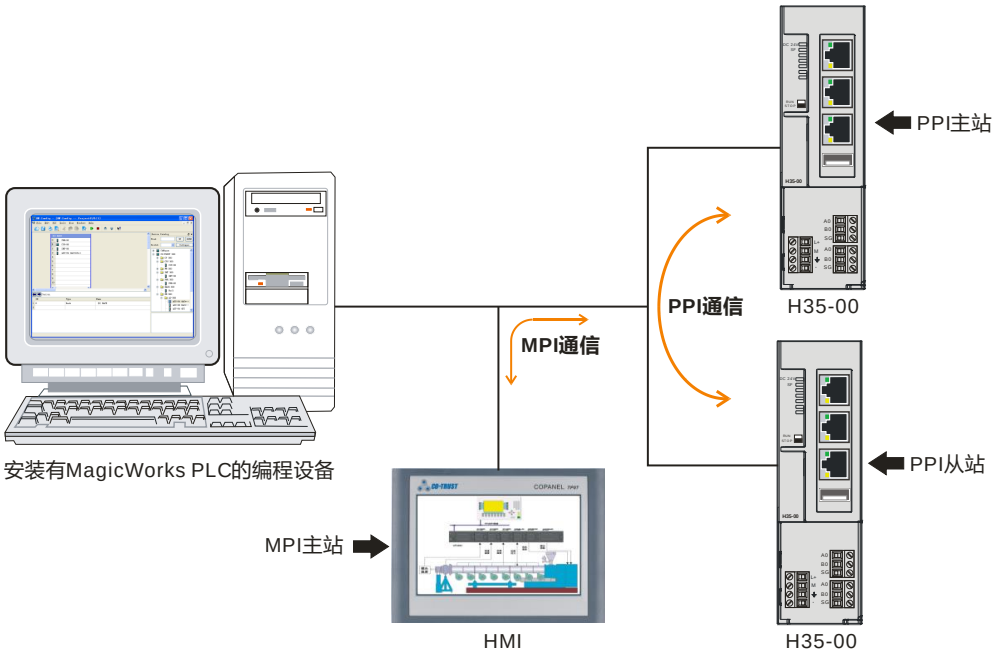
### 1、示例组件

表 14-2 PPI/MPI 通信的示例组件

| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300 系统进行编程 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电 |
| H35-00 主控模块 | 本例使用两个 H35-00，一个作为 PPI 主站，另一个作为 PPI 从站，它们通过 RS485 接口进行 PPI 通讯 |
| Copanel HMI | 作为 MPI 主站，与 H35-00 相连进行 MPI 通信 |
| 标准网线 | 将H35-00连接到编程设备<br>**<备注>** 请参考5.2.3 制作标准网线制作标准网线。 |
| HMI 下载电缆 | 将Copanel HMI连接到PC，执行HMI的下载任务（订货号：CTS6291-USB01）<br><br> |
| H35-00 与 Copanel HMI 的通信线 | DB9转RS485接线端子<br><br> |

### 2、网络连接

- PPI 通信：网络中的两个 H35-00 使用网络读写指令 NETR/NETW 互相读写数据。
- MPI 通信：网络中的 Copanel HMI 作为 MPI 主站监控 H35-00（MPI 从站）。



### 3、操作步骤

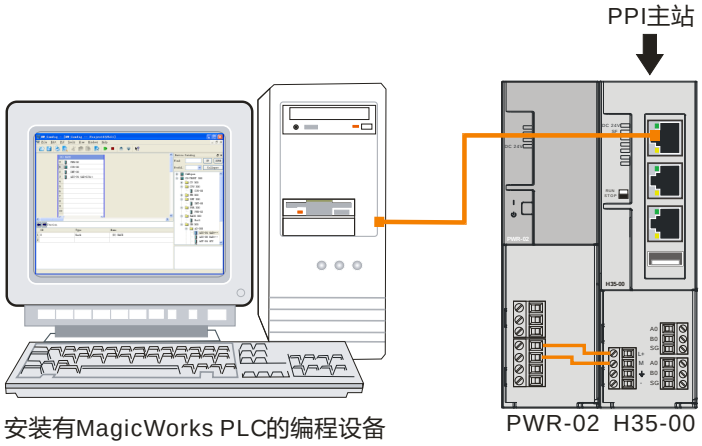
**步骤 1：连接 PPI 主站（H35-00）并为其供电**

1）打开 H35-00、电源模块 PWR-02 的前面板为它们接线。

2）使用标准网线连接 PC 与 PPI 主站（EtherNET 通信口）。

**步骤 2：对 PPI 主站（H35-00）进行编程**

本例以网络读写指令向导为例，对作为 PPI 主站的 H35-00 进行编程：将网络读写指令通过以太网通信口下载到 H35-00 中。

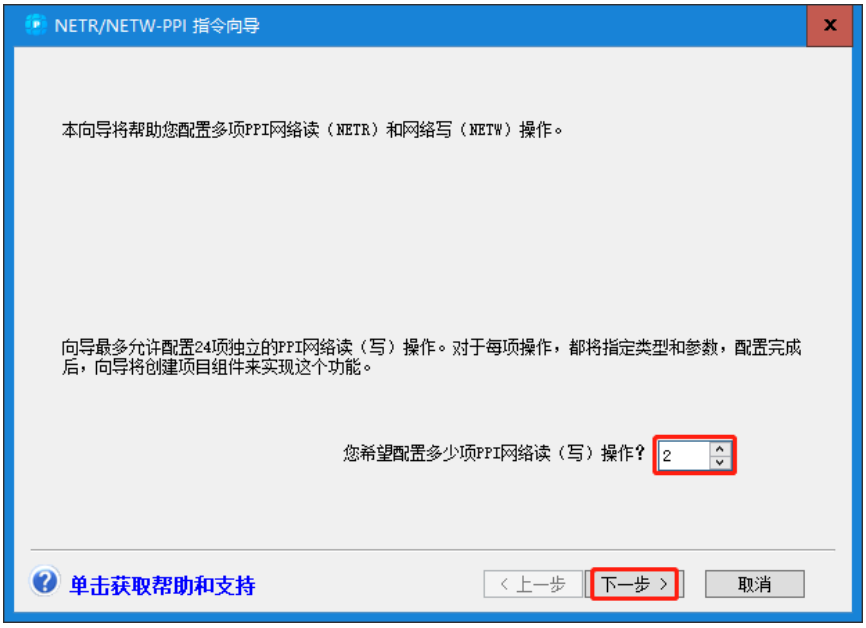



1）在 MagicWorks PLC 中新建一个工程，在该工程中添加 H35-00 站点，然后参考上述网络连接将 H35 与 PC 进行通信连接。

2）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向导进行配置。

**NETR/W-PPI 向导配置具体操作步骤如下：**

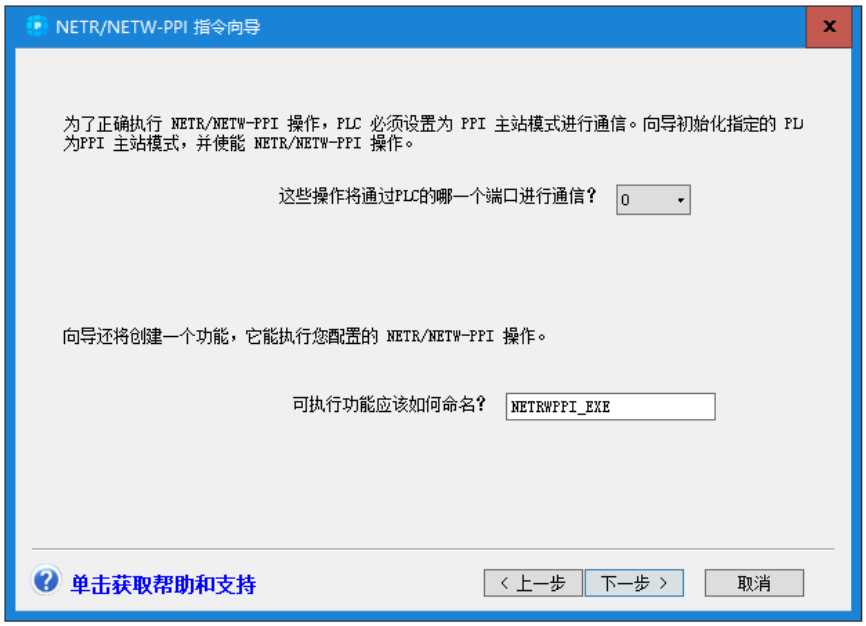
**<第 1 步>** 选择 PPI 通信端口进行网络读写操作，指定要配置的网络操作数：




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：

- 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。
- 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





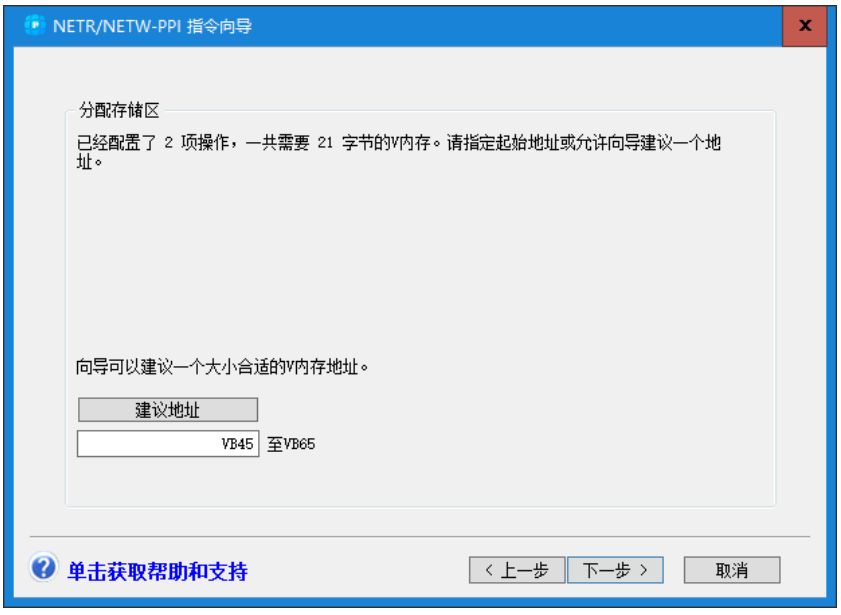

网络操作要配置的信息包括：

- 此操作是网络读还是写
- 从从站读入或写入从站多少个字节
- 要交互的 PLC 从站地址
- 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：

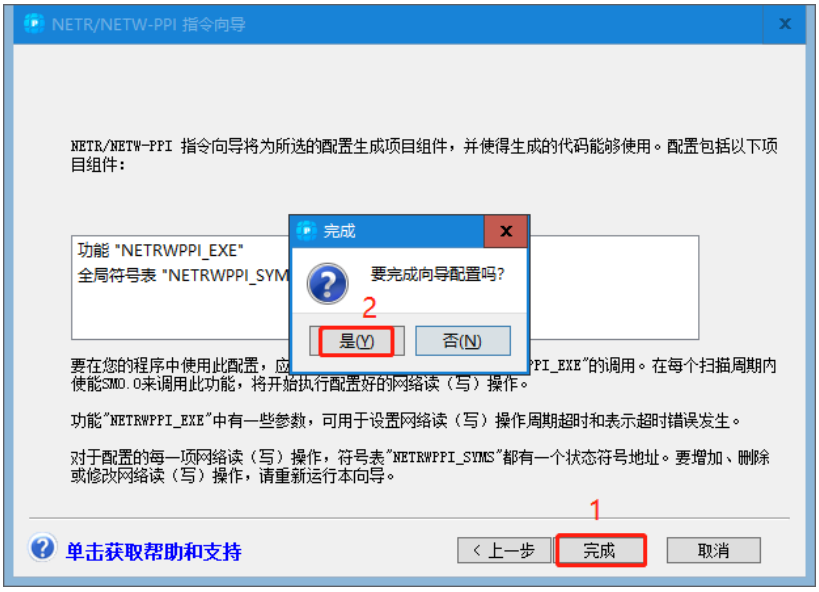
- 删除操作：删除当前的操作，此操作导致已配置的操作数目减一
- 上一个操作：将界面切换到上一个操作
- 下一个操作：将界面切换到下一个操作

**<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

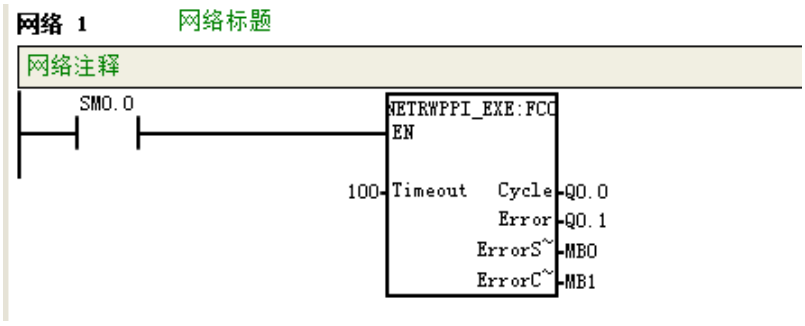



在以上对话框中点击“完成”按钮，在弹出的界面点击“是”完成 NETR/W_PPI 指令向导配置。最后一步，指令向导将为您指定的配置生成相关的程序组件，包括：

- 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，要在程序中使能网络通信，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。

3）在程序块中调用该向导生成的功能块

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 通信操作如下表所示。

表 14-3　PPI 通信操作说明

| 操作步 | 本地控制器数据表位置 | 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:VB48 | 2 Bytes |
| 读操作 | VB2–VB3 <- VB0-VB1 | NETR2_Status:VB57 | 2 Bytes |

通过状态监控表查看配置的各项操作的执行状态时，请参考下表

表 14-4　UDP 网络读写操作状态字节定义

| 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 | 详细错误代码说明参见下表 |

表 14-5　错误代码说明

| 错误代码 | 定义 |
|---|---|
| 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 | 队里溢出错误，8 个以上 NETW/NETR 指令被激活 |
| 5 | 违反协议，未启用 SMB30/SMB130 将端口设置为 PPI 主站模式即尝试进行 NETW/NETR |
| 6 | 非法参数，NETW/NETR 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 |

**<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。

4）程序编辑完成后，保存当前程序并进行编译。然后在“项目管理器界面”的工具栏点击下载按钮或选择菜单项“PLC”→“下载”将当前程序下载到控制器中。

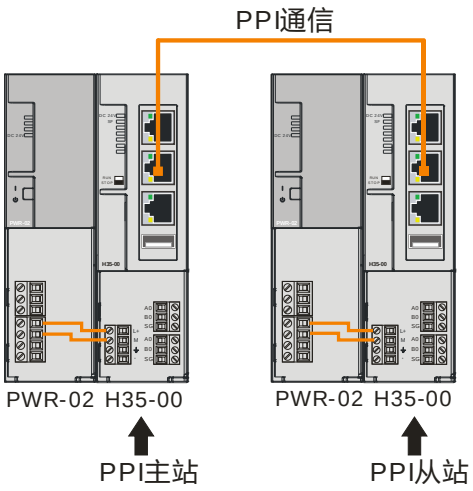
**<注意>** 为保证程序块、功能块同时被下载，请务必返回到项目管理器主界面执行下载操作。

**步骤 3：在 PPI 从站（H35-00）指定地址中写入值**

使用标准网线将 PPI 从站（H35-00）连接到编程设备，在 MagicWorks PLC 通信连接成功后，在状态表中为 PPI 从站的指定地址写入新值。

**步骤 4：执行 PPI 主从站通信**

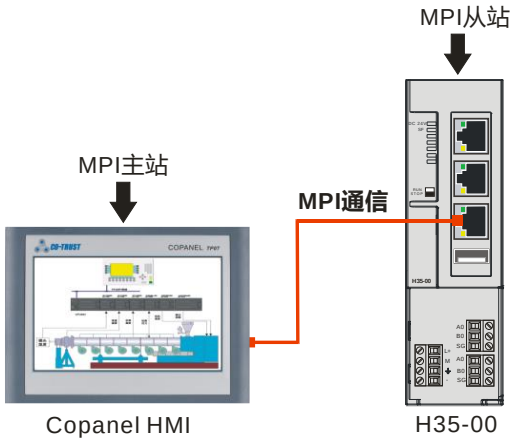
本例的 PPI 通信目的：PPI 主站读 PPI 从站内存中指定地址中的数值。




使用标准网线连接 PPI 主从站的 RS485 通信口（PORT1），然后将 PPI 主站的系统运行开关拨至 RUN。随后 PPI 主从站的 RS485 通信口（PORT1）指示灯开始闪烁，即表示 PPI 通信正常进行。

**步骤 5：执行 MPI 主从站通信**

将 Copanel HMI 与一个 H35-00 相连，然后通过 MPI 方式对 H35-00 进行访问。




1）通过上位机软件 MagicWorks HMI 为 Copanel HMI 组态工程。

首先在 MagicWorks HMI 中建立一个连接（通讯协议：CO-TRUST CTH 300），选择与 H35-00 匹配的波特率、协议、站地址。然后组态一个工程，并将工程下载到 Copanel HMI 中。

2）使用通信电缆连接 Copanel HMI 与 H35-00，随后 Copanel HMI 与 H35-00 即可进行 MPI 通信。

**<备注>** 当系统出现故障时，请参考章节 14.5 故障诊断获取 CTH300-H 系列主控模块及扩展模块的诊断方法。

### 14.1.3 Modbus RTU 通信

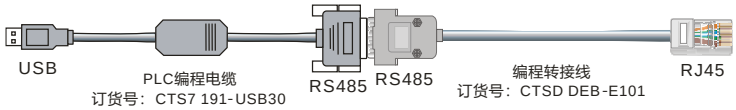
本指南通过一个具体实例来引导您建立一个应用程序，通过本例，您将熟悉 H35-00 的 Modbus RTU 通信功能。

选择 H35-00 的 PORT0 或 PORT1 进行 Modbus RTU 通信时，需要使用 CT_Modbus 库文件配置通信，该库文件包括 4 个库，分别是 PORT0 的主从站库和 PORT1 的主从站库。本例使用 PORT0 的 Modbus RTU 库文件。

**<备注>** 请参考附录 E CT_Modbus 库的使用介绍获取 CT_Modbus 库文件的相关使用说明。

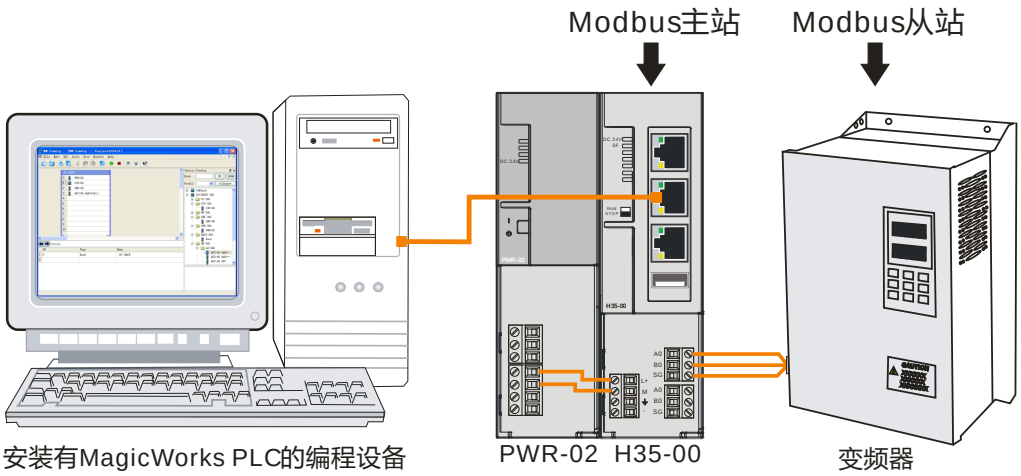
## 1、示例组件

表 14-6　Modbus RTU 通信的示例组件

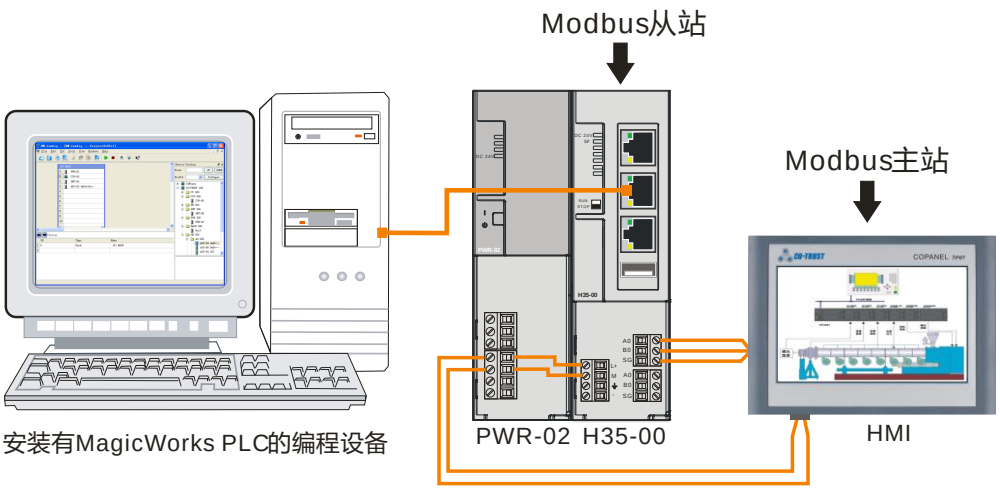
| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 系列可编程控制器进行组态、编程和调试 |
| 电源模块 PWR-02 | 提供 H35-00 及 Copanel HMI 的 24VDC 工作电源 |
| H35-00 主控模块 | CPU 执行用户程序，向 CTH300 系统总线提供 5V 电压，并通过以太网接口与其它模块进行通讯。 |
| Copanel HMI | Copanel 系列 HMI，作为 Modbus 主站与 H35-00 通信 |
| 变频器 | 支持 Modbus 协议的变频器，作为 Modbus 从站与 H35-00 通信 |
| 编程电缆 | 将H35-00连接到编程设备：PLC编程电缆+编程转接线<br> |

## 2、网络连接

◆ H35-00 作为 Modbus RTU 主站与变频器通信




◆ H35-00 作为 Modbus RTU 从站与 Copanel HMI 通信




## 3、操作步骤

**步骤 1：为 H35-00、电源模块、HMI 和变频器接线**

为 H35-00、电源模块 PWR-02、Copanel HMI 和变频器接线。具体步骤如下：

1）使用编程电缆连接 PC 与 H35-00。

2）当 H35-00 作为 Modbus RTU 主站与变频器通信时，使用通信线将 H35-00 的 RS485 接线端子的 1A、1B 连接至变频器的 Modbus 通信口。

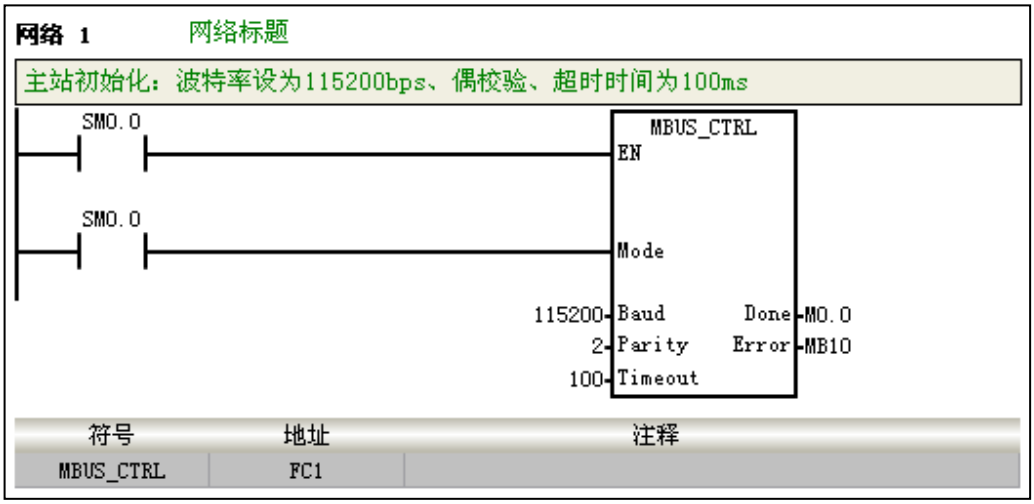
3）当 H35-00 作为 Modbus RTU 从站与 Copanel HMI 通信时，使用通信线将 H35-00 的 RS485 接线端子的 1A、1B 连接至 HMI 的 Modbus 通信口。

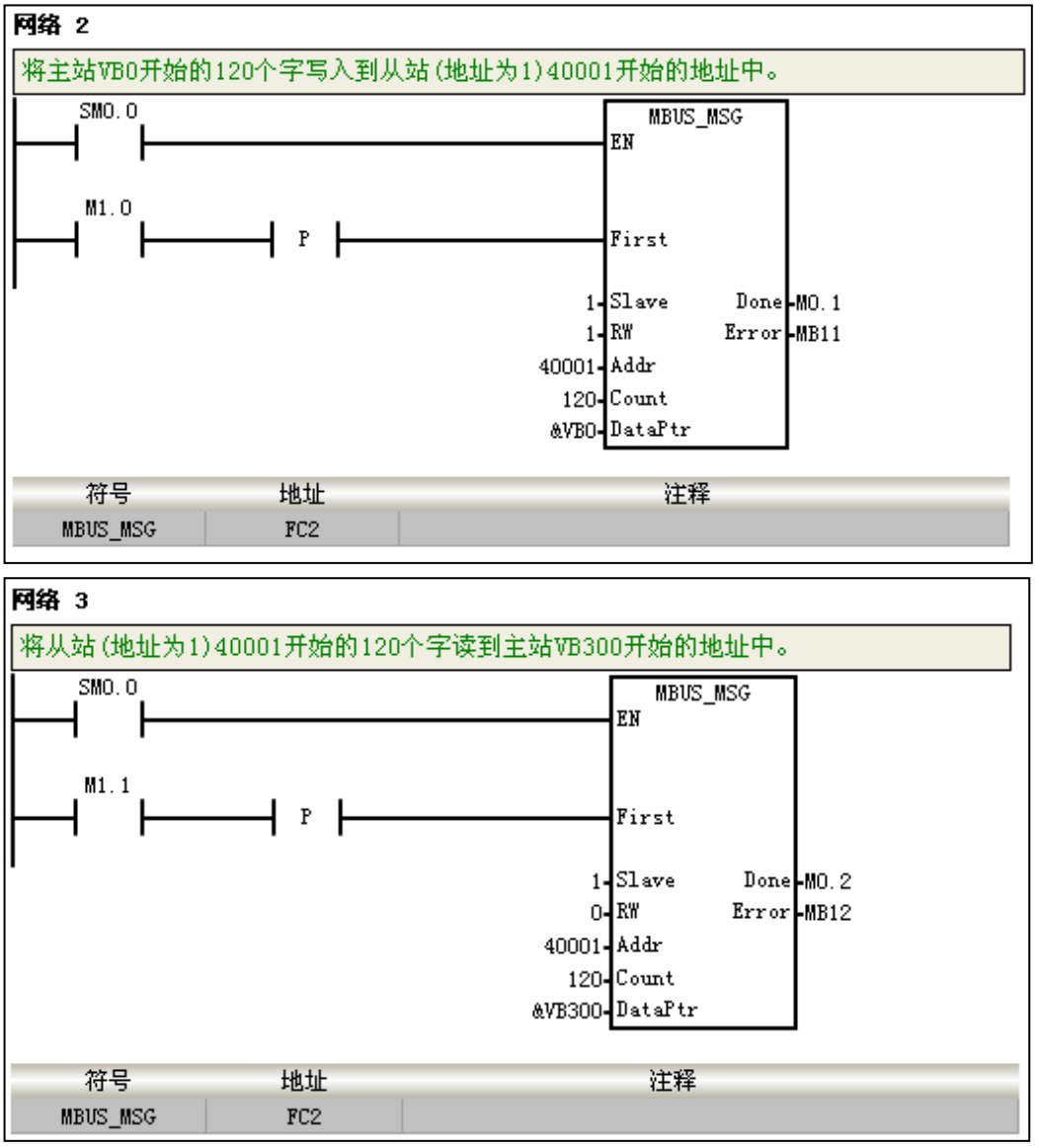
**步骤 2：设置通信**

在 MagicWorks PLC 中新建一个工程，在该工程中添加 H35-00 站点，然后参考网络连接将 H35-00 与 PC 进行通信连接。

**步骤 3：当 H35-00 作为 Modbus 主站与变频器通信**

1）为 Modbus 主站（H35-00）编辑的程序如下：





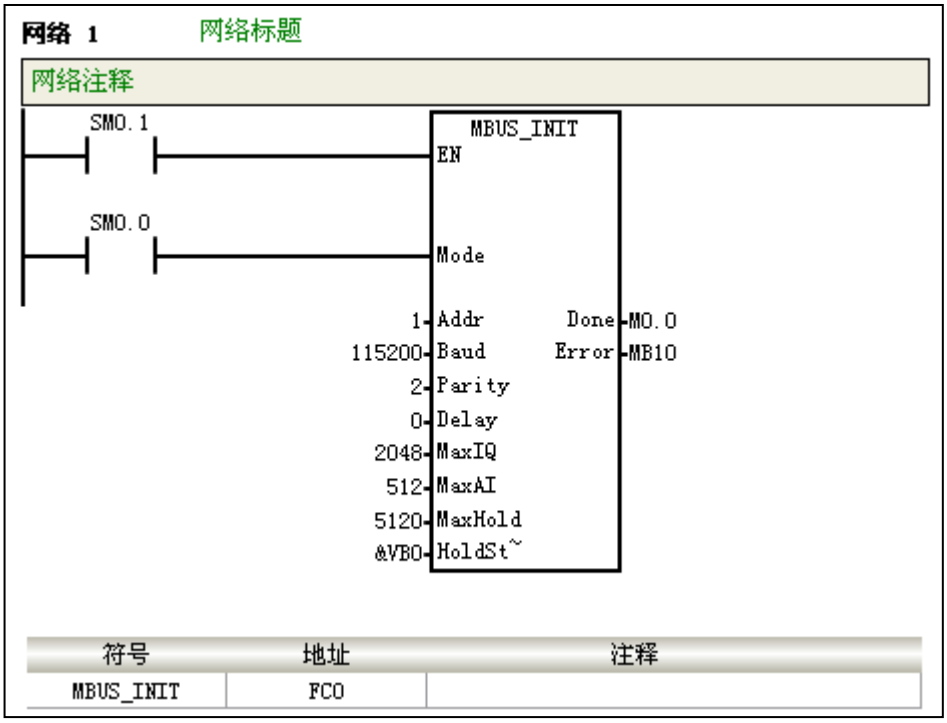

2）程序编辑成功后，保存当前程序并进行编译。选择菜单项“PLC”→“下载”将程序块和硬件组态从编程设备下载到 Modbus 主站（H35-00）中。若您的 H35-00 处于运行模式，下载界面将弹出一个对话框提示您将 H35-00 置于 STOP 模式，点击“确定”将 H35-00 置于 STOP 模式。

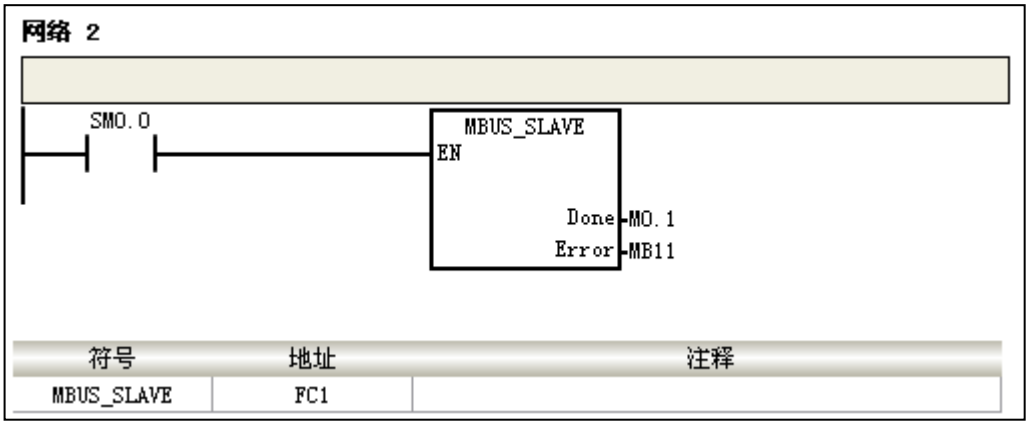
**<注意>** 为保证程序块、功能块同时被下载，请务必返回到项目管理器主界面执行下载操作。

3）Modbus 主站（H35-00）开始与变频器进行 Modbus RTU 通信。

**步骤 4：当 H35-00 作为 Modbus 从站与 Copanel HMI 通信**

1）为 Modbus 从站（H35-00）编辑的程序如下：




2）程序编辑成功后，保存当前程序并进行编译。将 Modbus 从站（H35-00）的系统运行开关拨到 STOP，然后返回 MagicWorks PLC 主界面，选择菜单项“PLC”→“下载”将当前程序下载到 Modbus 从站中。

**<注意>** 为保证程序块、功能块同时被下载，请务必返回到项目管理器界面执行下载操作。

3）Modbus 从站（H35-00）开始与 Copanel HMI 进行 Modbus RTU 通信。

### 14.1.4 Modbus TCP 通信

本节将通过一个具体实例来展示 CTH300-H 系列 CPU 的 Modbus TCP 通信功能。

在 Modbus TCP 通信中，CPU 作为从站时，通讯独立于整个大循环周期，即收即回；CPU 作为主站时，其收发由用户程序控制。

使用 H35-00 的 EtherNET 通信口进行 Modbus TCP 通信时，无需任何配置，H35-00 即可作为 Modbus TCP 从站实现 Modbus TCP 通信。MODBUS-TCP 通信默认端口号为 502。

当 H35-00 作为 Modbus TCP 主站与其他从站进行通信时，可通过三种方式完成 Modbus TCP 通信配置，Modbus TCP 单条读写指令（ct_mbus_master_tcp_single）、MBTCPSND 指令以及 Modbus TCP 向导。

**提示**

请参考《MagicWorks PLC 用户手册》获取 Modbus TCP 向导的相关使用说明；

请参考本文档附录 E.4 CT_Modbus_master_tcp_single 库功能说明获取 Modbus TCP 单条读写指令库文件的相关使用说明；

请访问合信官网下载相关手册和库文件，网址：http://www.co-trust.com/Download/index.html

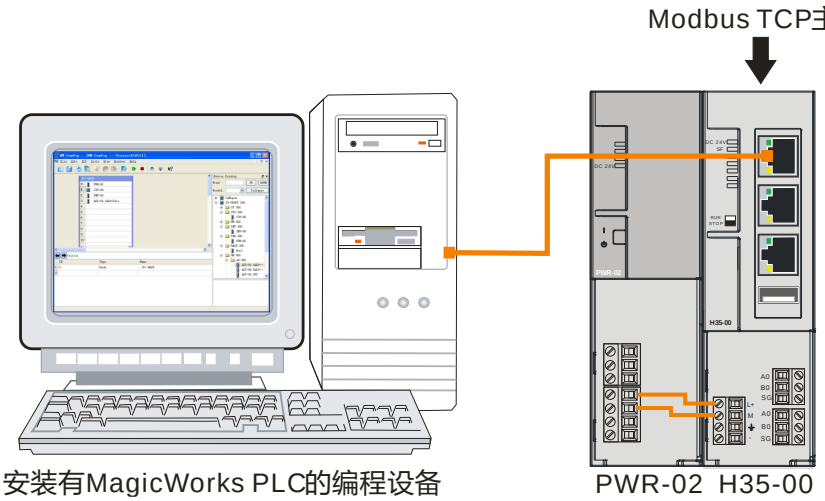
### 1、示例组件

表 14-7　Modbus TCP 通信的示例组件

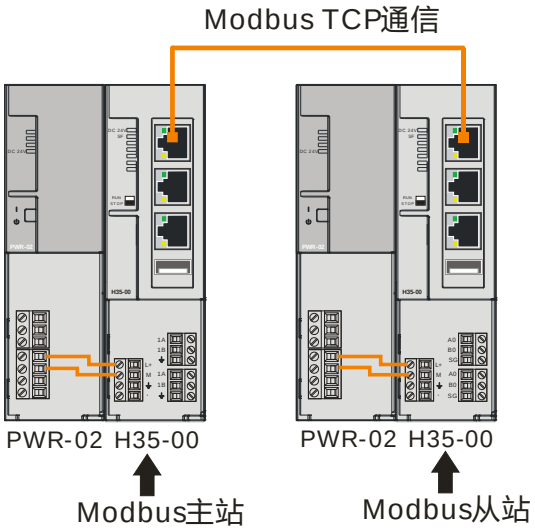
| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 可编程控制器进行组态、编程和调试。 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电 |
| H35-00 主控模块 | 本例使用两个 H35-00，一个 H35-00 作为 Modbus 主站，另一个 H35-00 作为 Modbus 从站，它们通过 EtherNET 接口进行 Modbus TCP 通讯。 |
| 标准网线 | 连接H35-00与编程设备<br>连接Modbus主站（H35-00）与Modbus从站（H35-00） |

### 2、网络连接

使用标准网线连接编程设备与 H35-00，使用 CT_Modbus_TCP 库进行编程：




使用标准网线连接 Modbus 主站和从站，主站将进行一个写操作，将指定地址中的数据写入到从站中，然后，通过从站读取相应地址中的数据，继而实现 Modbus TCP 通信：




## 3、操作步骤

**步骤 1：为 H35-00 与电源模块接线**

打开 H35-00、电源模块 PWR-02 的前面板，然后参照网络连接为它们接线。

**步骤 2：连接电缆**

使用一条标准网线连接 PC 与 Modbus 主站。

**步骤 3：设置通信**

在 MagicWorks PLC 中新建一个工程，在该工程中添加 H35-00 站点，然后参考网络连接将 H35-00 与 PC 进行通信连接。

**步骤 4：对 Modbus 主站（H35-00）进行编程**

1）通过 Modbus TCP 库进行 Modbus TCP 通信设置，在 MagicWorks PLC 的程序块中添加库指令，以下指令为“Ct_Mbus_master_tcp_single”库文件中的“MBTCPS_EXE”库指令：

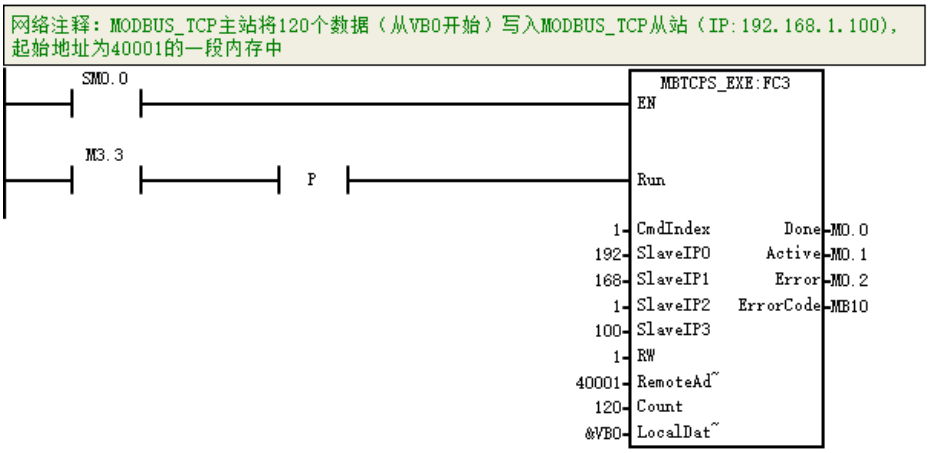



表 14-8 “MBTCPS_EXE”库指令定义

| 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 | WORD | 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 |

表 14-9　输出端的错误代码定义

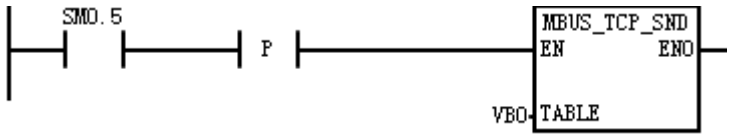
| 错误代码 | 说明 |
|---|---|
| 0 | 无错 |
| 1 | 连接数达到最大值 |
| 2 | 正在建立连接 |
| 3 | 接收超时，从站无响应 |
| 4 | 请求的参数有错误 |
| 5 | 指令没有使能 |
| 6 | 连接正忙于处理其它请求（如同时激活多条 MBTCPS_EXE） |

2）通过 Modbus_TCP 网络读写指令进行 Modbus TCP 通信设置，在 MagicWorks PLC 的程序块中插入该指令，库指令定义如下：

**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。 |

表 14-10　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

表 14-11　TABLE 参数错误代码（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：**

- 同时可有多个从站消息读写，但是每个从站只能同时有一条消息读写。
- MBUS_ TCP_SND 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。

**MODBUS_TCP 网络读写指令应用举例：**

**网络 4**

网络注释：SM0.5的上升沿，将从VB16开始的120个字写到远程IP为192.168.1.202，端口号为502，从VW0开始的一段CPU内存中|

```
LD        SM0.1
MOVB      192, VB1
MOVB      168, VB2
MOVB      1, VB3
MOVB      202, VB4
MOVW      502, VW5
MOVB      1, VB7
MOVD      40001, VD8
MOVB      1, VB12
MOVW      120, VW14
```

**网络 5**

```
LD        SM0.5
EU
MBTCPSND  VB0
```

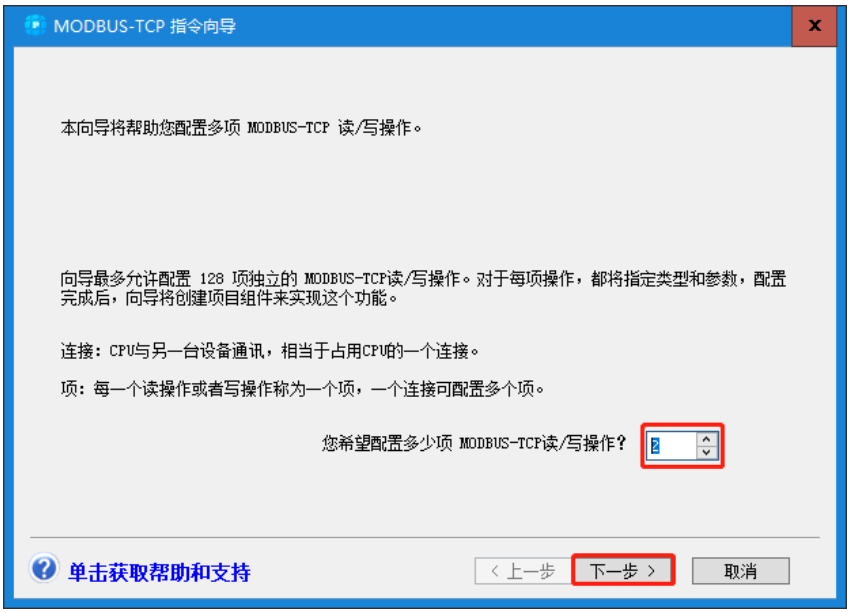
3）通过 Modbus TCP 指令向导为 Modbus_TCP 主站编程

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 向导进行向导配置。

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：**

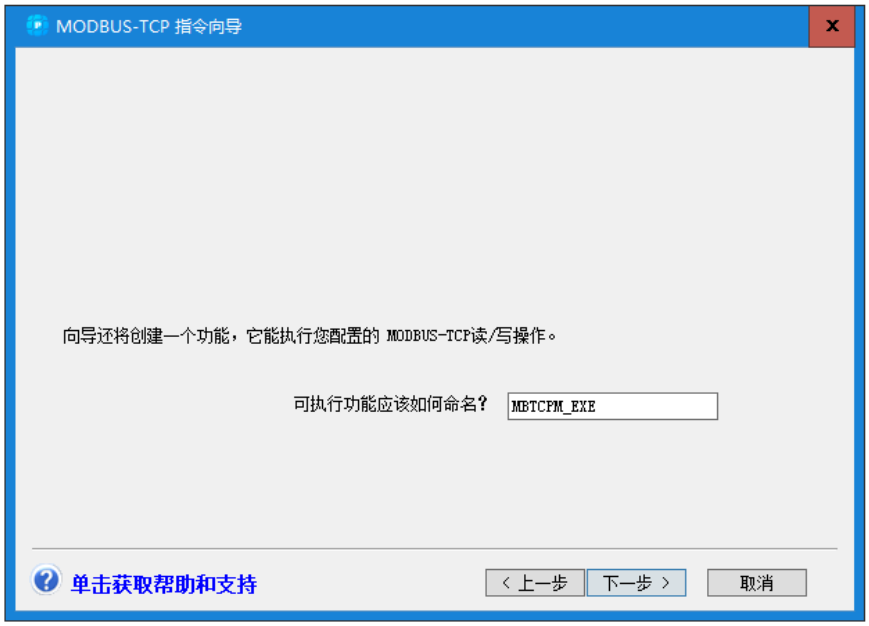
**<第 1 步>** 指定要配置的网络操作数




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。

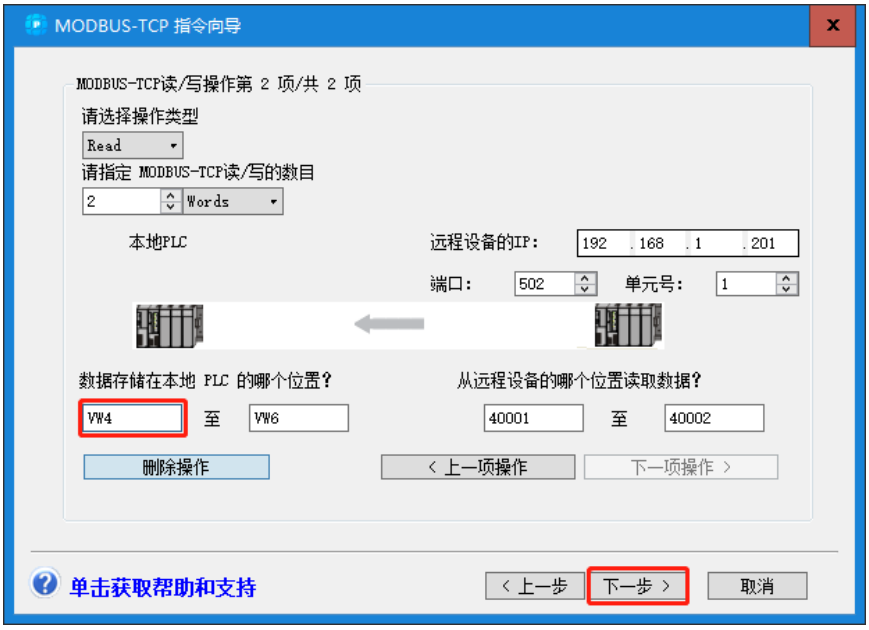
**<第 2 步>** 指定生成的功能名称




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：

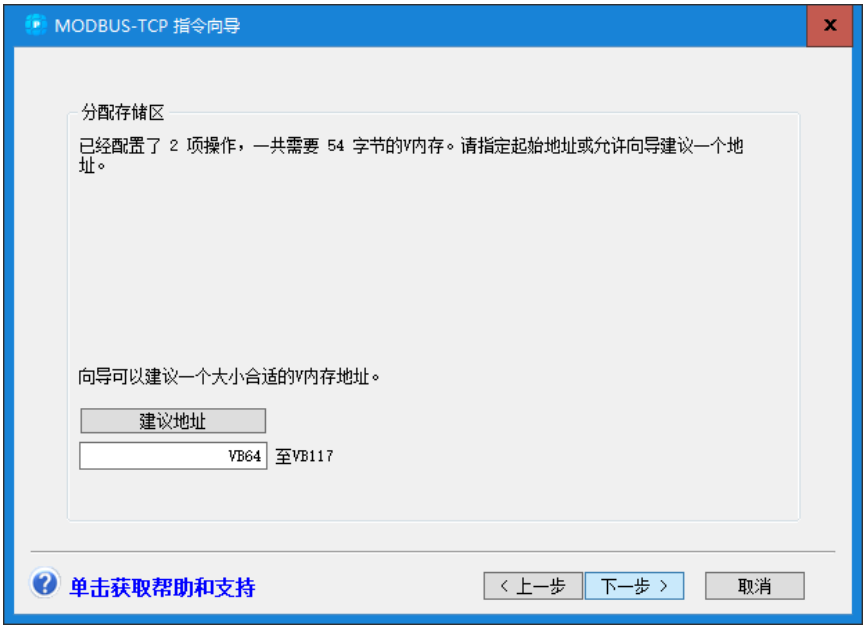
- 此操作是网络读还是写
- 从从站读入或写入从站的数目，单位可选字或位
- 要进行远程通信的 PLC 从站地址、端口号（从站为控制器时此值默认为 502）、单元 ID。
- 本地 PLC 地址映射（只能是 V 内存），即读回的数据或待写入的数据存储在本地 PLC V 内存中的位置
- 远程 PLC 地址映射（只能是 V 内存），即从远程 PLC 读取或向远程 PLC 写入的 V 内存位置

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

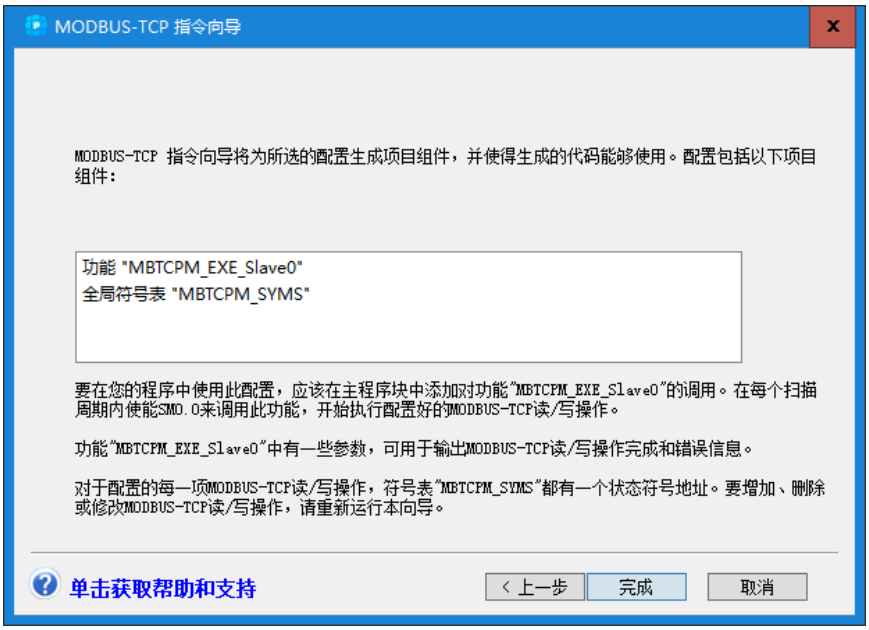
若需配置多个操作，在配置完的操作界面点击“下一个操作”即可进入下一个操作的配置界面，配置完最后一个操作后，点击“下一步”。

**<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。

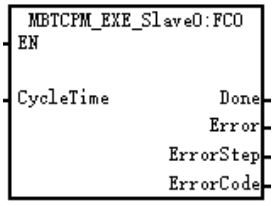
**<第 5 步>** 生成项目组件



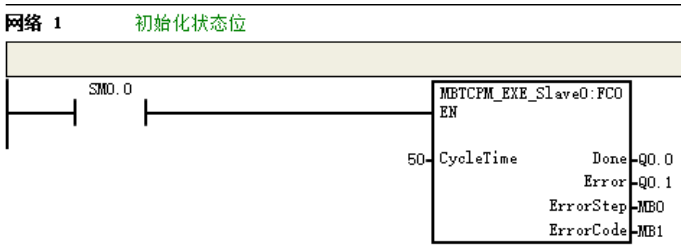

最后一步，点击“完成”，在弹出的确定界面点击“确定”即可完成 MODBUS TCP 指令向导配置，指令向导将为您指定的配置生成相关的程序组件，包括：

- 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 包含了操作具体代码，要在程序中使能网络通信，需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。

表 14-12 “MBTCPM_EXE”指令说明

| 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。<br>0：没有错误；<br>1：连接数达到最大值；<br>2：正在建立连接；<br>3：超时错误；<br>4：请求的参数有错误；<br>5：指令没有使能；<br>6：连接正忙于处理其它请求；<br>7：应答错误；<br>8：创建连接失败；<br>9：连接已存在；<br>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。

表 14-13 ModBus_TCP 通信操作说明

| 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，请参考下表：

表 14-14　ModBus_TCP 网络读写操作状态字节定义

| 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； |

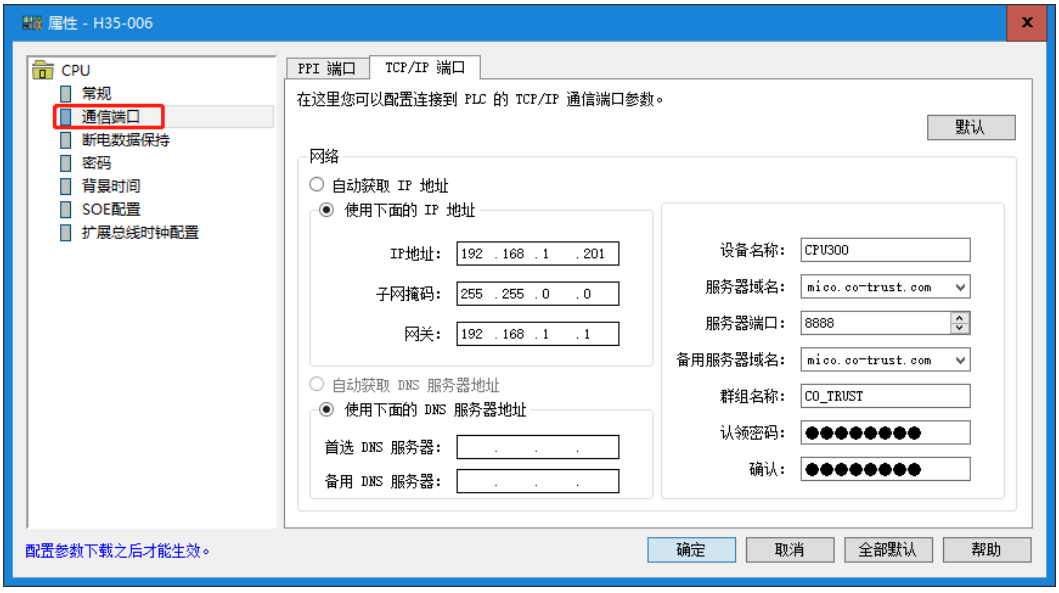
ModBus 地址通常是包含数据类型和偏移量的 5 个或 6 个字符值。前一个或两个字符决定数据类型，后几个字符是一个符合当前数据类型的相应值。

**备注：**最多可配置 32 个服务器。

在 MagicWorks PLC 向导中打开 Modbus TCP 指令向导，逐步配置后即可通过 SM0.0 调用生成的 FC。更多详细操作请访问合信官网参下载 MagicWorks PLC 用户手册进行参考，地址：http://www.co-trust.com/Download/index.html

**步骤 5：Modbus TCP 主站与 Modbus TCP 从站互连通信**

组态 Modbus TCP 从站时，从站 IP 必须对应于主站编程设定的 IP，如需更改从站 IP，请在硬件组态中双击 CPU 模块打开属性修改其的 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。




**步骤 6：调试**

按顺序执行以上步骤后，使用标准网线连接 Modbus 主站与从站，然后通过 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。

## 4、ModBus TCP 从站地址映射

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：

表 14-15 TCP 协议对应的从站地址映射

| ModBus 从站地址 | 控制器地址 |
|---|---|
| 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) |

### 14.1.5 UDP_PPI 通信

本节将通过一个具体实例来展示 H35-00 的 UDP_PPI 通信功能（Ethernet 通信口）。

在 UDP PPI 通信中，CPU 作为从站时，通讯独立于整个大循环周期，即收即回；CPU 作为主站时，其收发由用户程序控制。

H35-00 作为主站或从站，使用网络读写指令 UDP_NETR/UDP_NETW 或 NETW/NETR 指令向导与局域网内的其他设备进行 UDP PPI 通信。

**提示**

请参考《MagicWorks PLC 用户手册》获取 UDP 读写指令向导相关使用说明。相关手册和库文件请访问合信官网，网址：http://www.co-trust.com/Download/index.html

**1、示例组件**

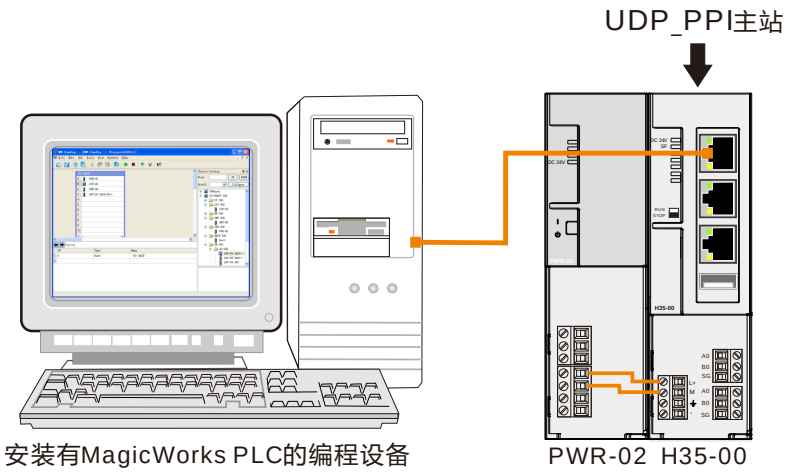
表 14-16　UDP PPI 通信的示例组件

| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 可编程控制器进行组态、编程和调试 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电 |

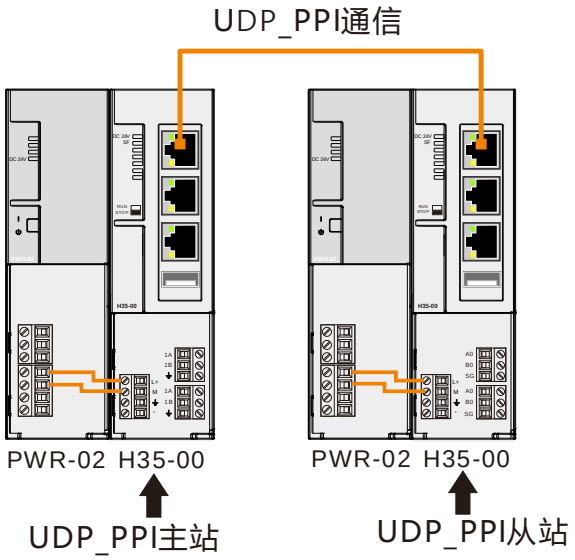
| | |
|---|---|
| H35-00 主控模块 | 本例使用两个H35-00，一个H35-00作为UDP_PPI主站，另一个H35-00作为 UDP_PPI 从站，它们通过 EtherNET 接口进行 UDP_PPI 通讯 |
| 标准网线 | 连接H35-00与编程设备<br>连接UDP_PPI主站（H35-00）与UDP_PPI从站（H35-00） |

## 2、网络连接

使用标准网线连接编程设备与 H35-00，通过编程设备对 UDP_PPI 主站（H35-00）进行编程：




使用标准网线连接 UDP_PPI 主站和 UDP_PPI 从站，UDP_PPI 主站将进行一个写操作，将指定地址中的数据写入到 UDP_PPI 从站中，然后通过 UDP_PPI 从站读取相应地址中的数据，继而实现 UDP_PPI 通信：




## 3、操作步骤

**步骤 1：为 H35-00 与电源模块接线**

打开 H35-00、电源模块 PWR-02 的前面板，然后参考网络连接为它们接线。

**步骤 2：连接电缆**

使用一条标准网线连接 PC 与 UDP_PPI 主站（H35-00）。

**步骤 3：设置通信**

在 MagicWorks PLC 中新建一个工程，在该工程中添加 H35-00 站点，然后参考章节 2.3 设置通讯将 H35-00 与 PC 进行通信连接。

**步骤 4：对 UDP_PPI 主站（H35-00）进行编程**

以 UDP 网络读写为例，您可以通过以下两种方法之一对 UDP_PPI 主站进行网络读写操作。通过 UDP 网络读写指令 UDP_NETR/UDP_NETW 为 UDP_PPI 主站编程

表 14-17 “UDP_NETR/UDP_NETW”指令 TABLE 参数表

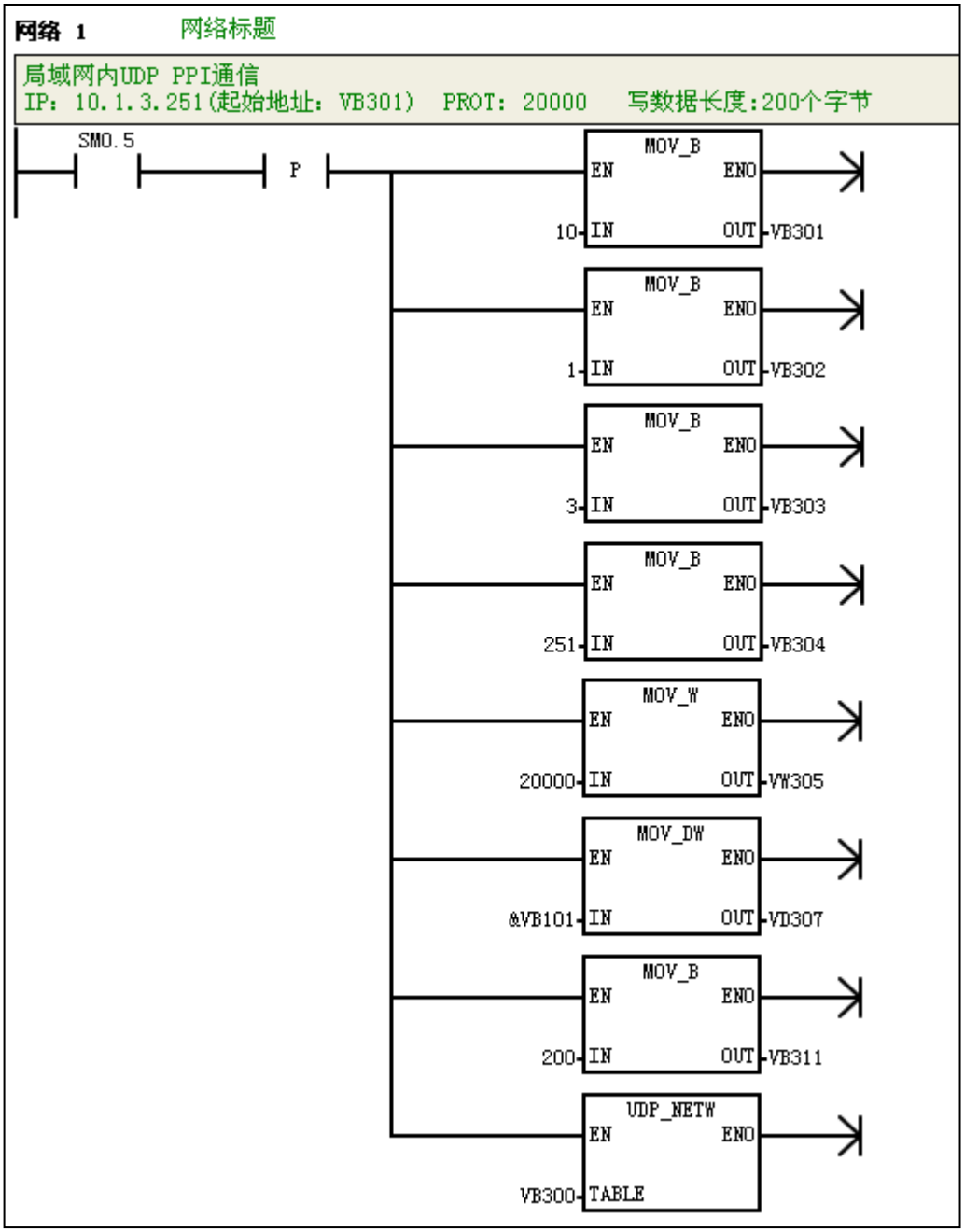
| D | A | E | 0 | 错误代码 | 0 |
|---|---|---|---|---|---|
| IP 地址第 1 个字节 | | | | | 1 |
| IP 地址第 1 个字节 | | | | | 2 |
| IP 地址第 1 个字节 | | | | | 3 |
| IP 地址第 1 个字节 | | | | | 4 |
| 端口号高字节 | | | | | 5 |
| 端口号低字节 | | | | | 6 |
| 指向远程站指针<I，Q，M，V，DB>（共 4 个字节）第一个字节 | | | | | 7 |
| 指向远程站指针第 2 个字节 | | | | | 8 |
| 指向远程站指针第 3 个字节 | | | | | 9 |
| 指向远程站指针第 4 个字节 | | | | | 10 |
| 数据长度 | | | | | 11 |
| 数据字节 0 | | | | | 12 |
| 数据字节 1 | | | | | 13 |
| ... | | | | | ... |
| 数据字节 199 | | | | | 211 |

D：完成（功能完成），0=未完成、1=完成

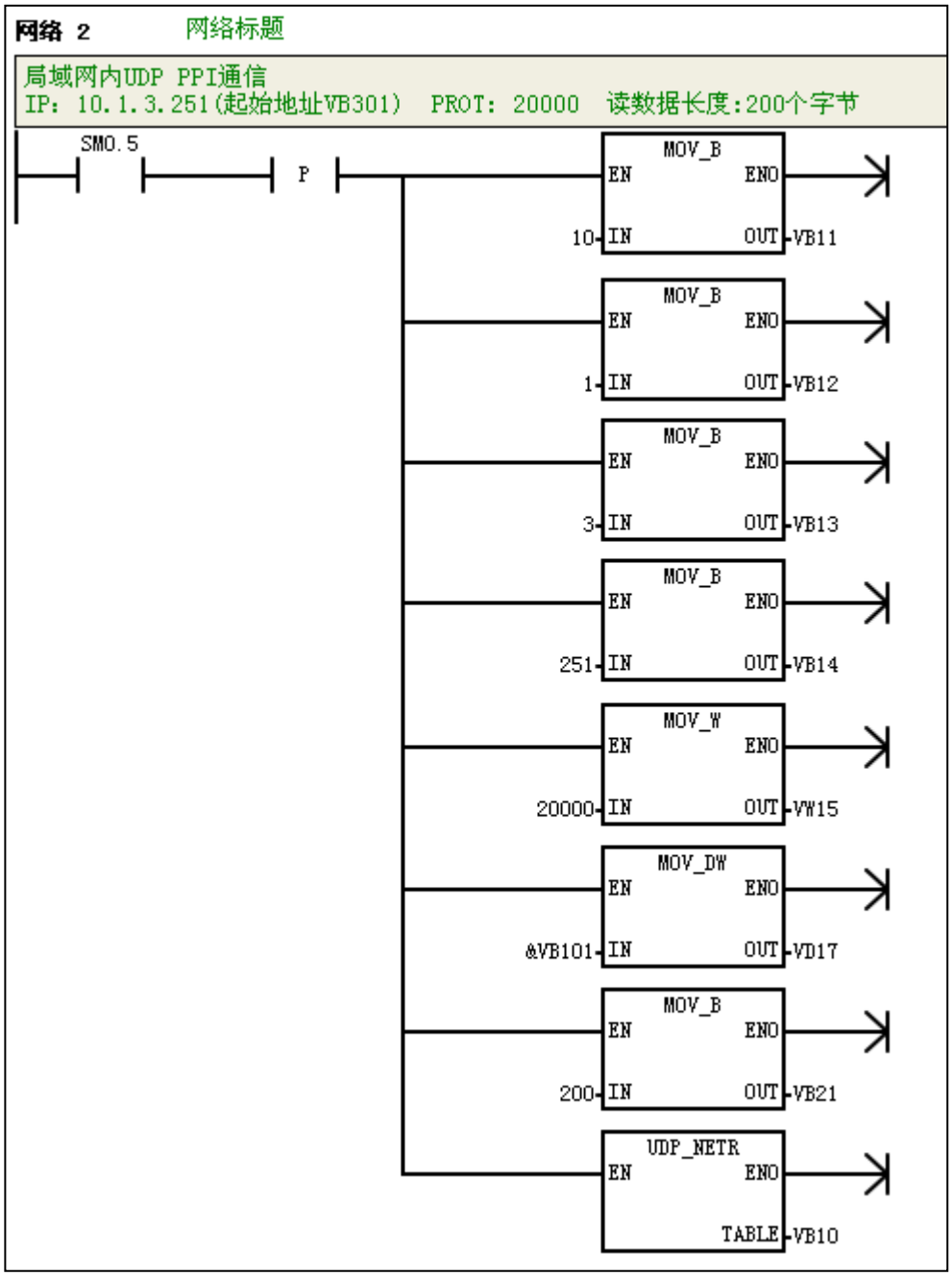
A：现用（功能入队），0=非现用、1=现用

E：错误，0=无错、1=错误

1）**网络 1：**将 UDP_PPI 主站（起始地址：VB312）中的 200 个字节数据写入到 UDP_PPI 从站（IP：10.1.3.251、起始地址：VB101）中。

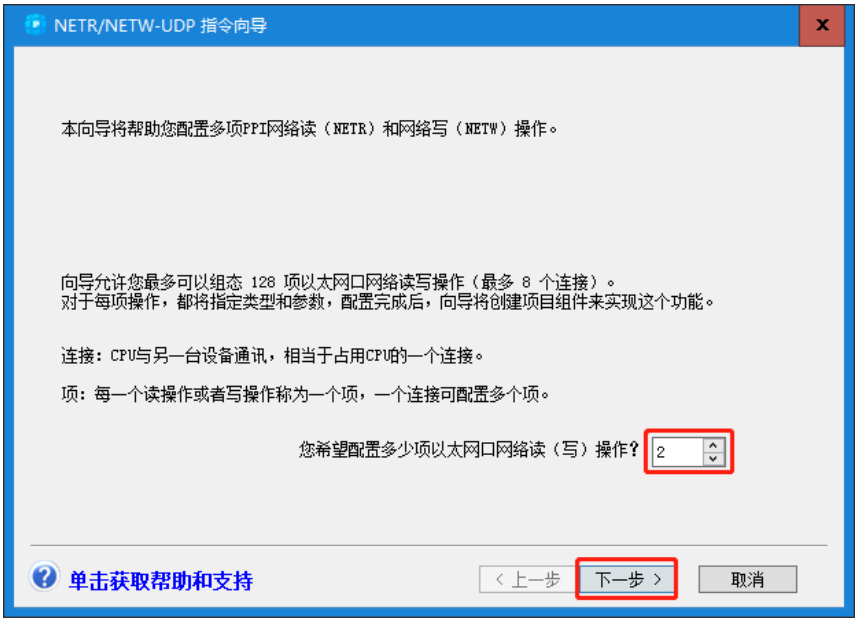



**网络 2：**将 UDP_PPI 从站（IP：10.1.3.251、起始地址：VB101）中的 200 个字节数据读入到 UDP_PPI 主站（起始地址：VB22）中。

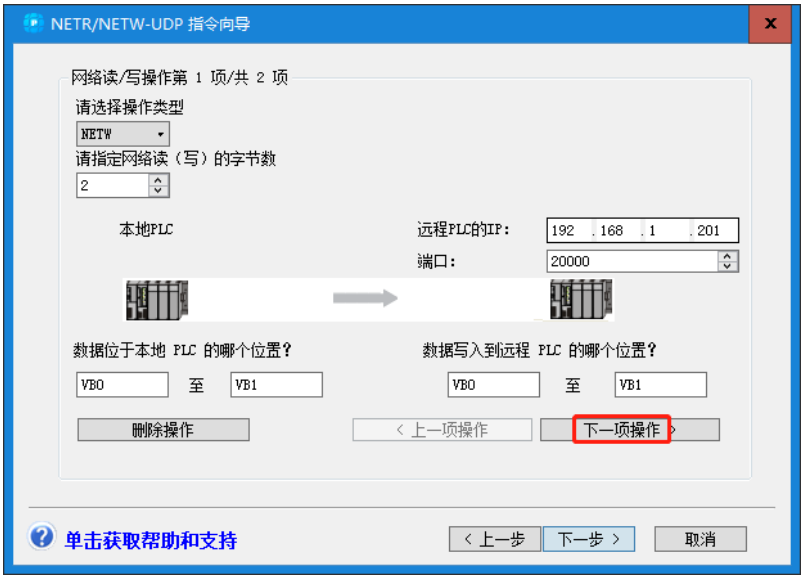



2）通过 NETR/NETW 指令向导为 UDP_PPI 主站编程

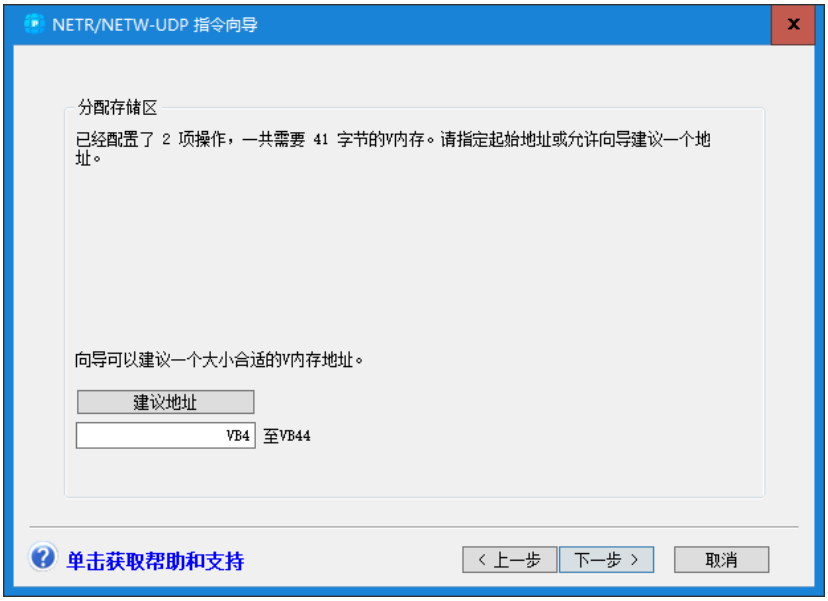
在 H35-00 站点下的项目管理器界面菜单项选择“工具”→“NETR/NETW-UDP 指令向导”或选择“向导”，然后在右侧的工作区域双击“NETR/NETW-UDP 指令向导”打开如下向导对话框，逐步进行配置，最后点击“完成”，在弹出的确定窗口中点击“确定”完成配置。

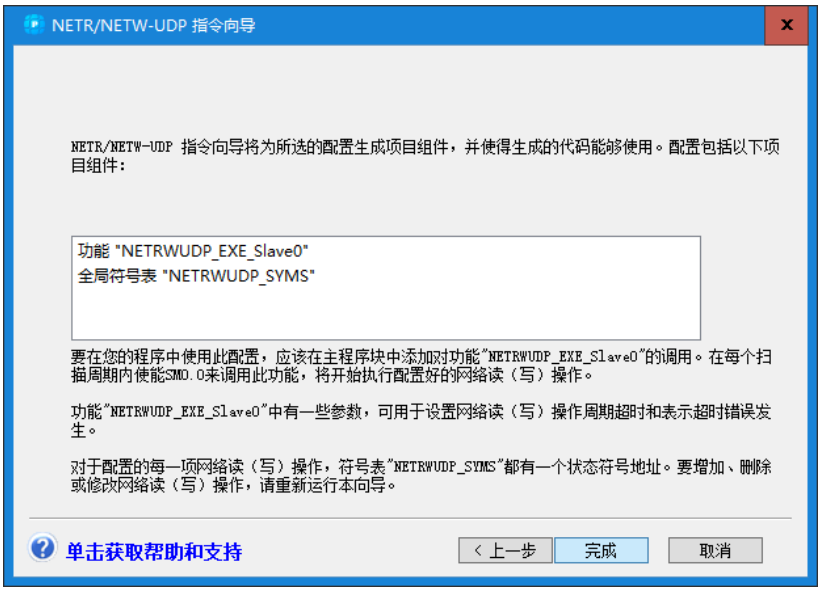

NETR/NETW-UDP 指令向导
本向导将帮助您配置多项PPI网络读（NETR）和网络写（NETW）操作。
向导允许您最多可以组态 128 项以太网口网络读写操作（最多 8 个连接）。
对于每项操作，都将指定类型和参数，配置完成后，向导将创建项目组件来实现这个功能。
连接：CPU与另一台设备通讯，相当于占用CPU的一个连接。
项：每一个读操作或者写操作称为一个项，一个连接可配置多个项。
您希望配置多少项以太网口网络读（写）操作？
2
单击获取帮助和支持
< 上一步
下一步 >
取消


NETR/NETW-UDP 指令向导
向导还将创建一个功能，它能执行您配置的 NETR/NETW-UDP 操作。
可执行功能应该如何命名？
NETRWUDP_EXE
单击获取帮助和支持
< 上一步
下一步 >
取消


NETR/NETW-UDP 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的IP：
192 . 168 . 1 . 201
端口：
20000
数据位于本地 PLC 的哪个位置？
VB0
至
VB1
数据写入到远程 PLC 的哪个位置？
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-UDP 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的IP：
192 . 168 . 1 . 201
端口：
20000
数据存储在本地 PLC 的哪个位置？
VB2
至
VB3
从远程 PLC 的哪个位置读取数据？
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




完成以上操作后，即可在 OB1 网络中通过 SM0.0 调用该 FC，如下图所示：

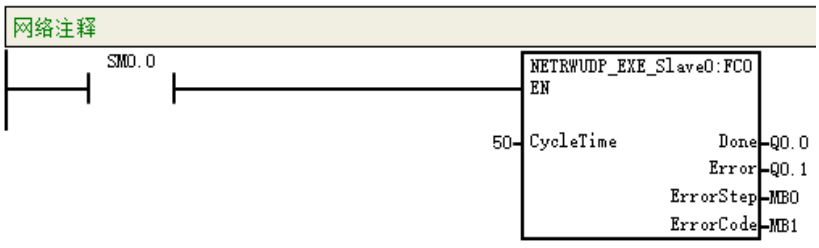



表 14-18　UDP_PPI 通信网络操作说明

| 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 | MBTCPR1_Status：VB4 | 2 Bytes |
| 读操作 | VB2-VB3 <- VB0-VB1 | MBTCPR2_Status：VB18 | 2 Bytes |

3）按照上述方法 1）或 2）完成程序编辑后，编译程序并将其下载到 UDP_PPI 主站设备中。

**步骤 5：UDP_PPI 主站与 UDP_PPI 从站互联通信**

1）给 UDP_PPI 主站、从站接通电源。

2）使用标准网线连接 UDP_PPI 主站与从站的 EtherNET 通信口，随后它们即可进行 UDP_PPI 通信。

### 4、UDP PPI 通信从站地址映射

UDP_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。

## 14.1.6 远程 EtherNET 通信

本节将通过一个具体实例来展示 H35-00 的远程 Ethernet 通信功能。

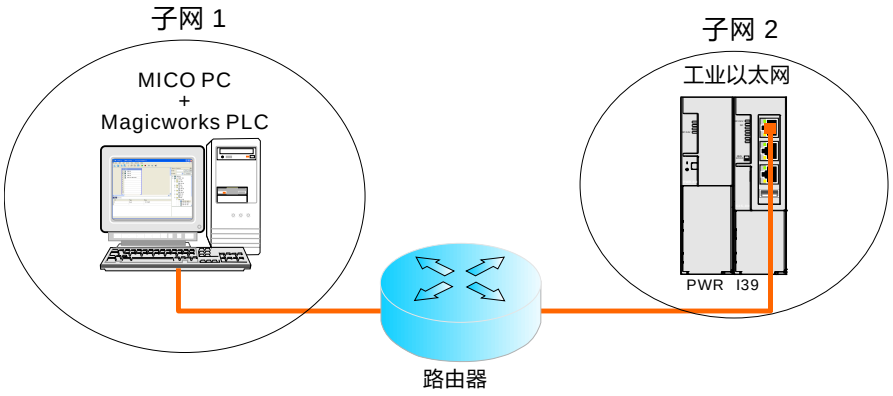
选择进行远程以太网通讯时，需要先由 MiCo 与远程 H35-00 建立通信，随后方可通过 MagicWorks PLC 编程软件对远程 H35-00 进行编程、监控等操作。

### 1、示例组件

表 14-19　远程 EtherNET 通信的示例组件

| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 系列可编程控制器进行组态、编程和调试；<br>安装有 MiCo，与 CTH300-H 系列主控模块进行通信。 |
| 装配导轨 | CTH300 系统机架，用于固定系统中的各模块。 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电 |
| H35-00 主控模块 | CPU 执行用户程序，向 CTH300 系统背板总线提供 5V 电压，并通过以太网接口与其它模块通信。 |
| 路由器 | 连接子网 1 和子网 2 中的设备，以使子网 1 和子网 2 中的设备能够读写对方的数据。 |
| 标准网线 2 根 | 连接 PC 与路由器；连接 H35-00 与路由器 |

### 2、网络连接




子网 1 中包含远程编程设备（安装有软件 MiCo 和 MagicWorks PLC），该编程设备用于对远

程 H35-00 通信和编程；子网 2 中包含远程编程对象 H35-00，H35-00 的 EtherNET 通信口与路由器相连；子网 1 和子网 2 中的设备都与同一个路由器相连。

**<备注>** MiCo 和 MagicWorks PLC 须安装于同一局域网中的同一台编程设备。

## 3、操作步骤

**步骤 1：接线**

打开 H35-00 和电源模块 PWR-02 的前面板，为它们接线。

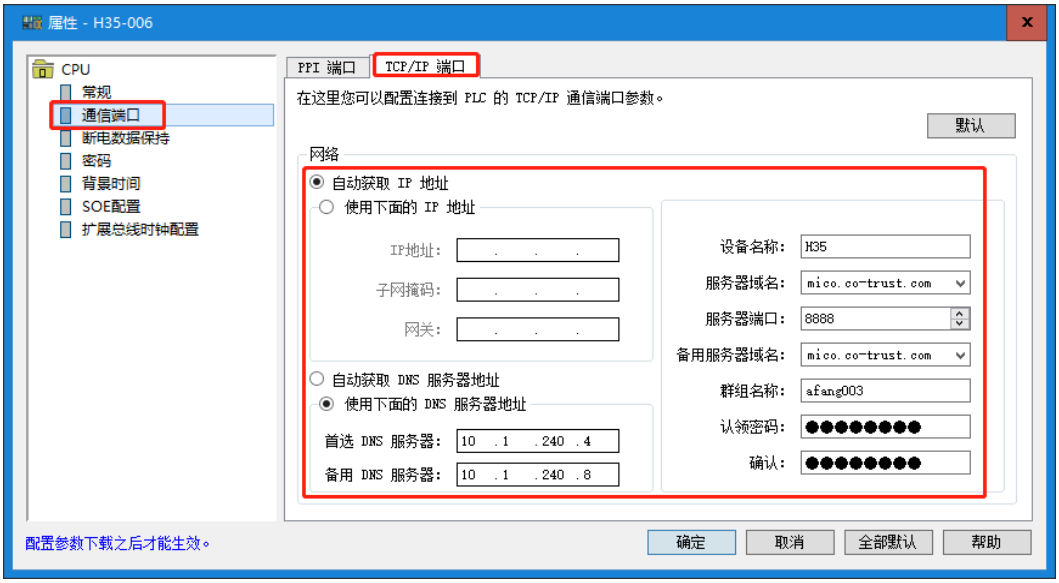
**步骤 2：连接电缆**

参考网络连接，使用标准网线连接 H35-00、编程设备与路由器。

**步骤 3：CPU 属性配置**

1）H35-00 站点下双击图标进入“硬件组态”界面。有关硬件组态详细步骤，请参见章节 2.3 硬件组态。

2）在“硬件组态”界面双击机架上的 H35-00 进行 CPU 属性配置。在属性配置界面选择“通信端口”→“TCP/IP 端口”进行 IP 和远程功能配置。可选择“自动获取 IP 地址”和“自动获取 DNS 服务器地址”，前提是保证控制器和编程设备处于同一网段。




**提示**

1、远程功能的设置及 MiCo 端远程编程操作请参考《MiCo 远程监控系统手册》，手册下载地址：http://www.co-trust.com/Download/index.html

2、现场需确认能接入外网，才能和 MiCo 服务器进行通讯。

**步骤 4：下载硬件配置块到 PLC**




**步骤 5：MiCo 端配置**

完成以上操作后可在 MiCo 客户端认领设备，将 CPU 加入到我的设备中。并启动远程编程。



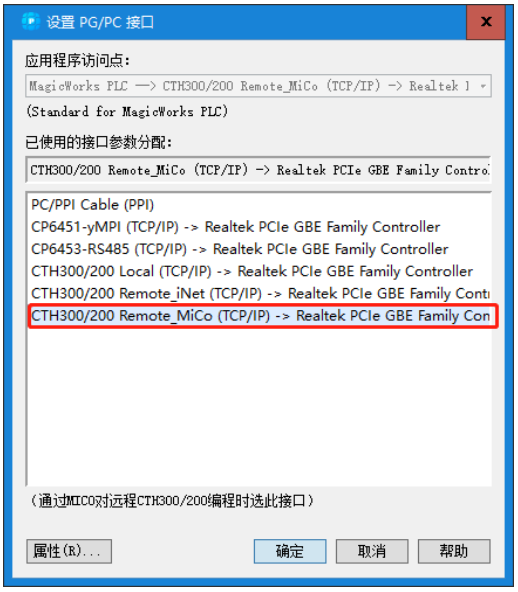

**备注：**有关设备在 MiCo 端更多详细操作步骤，请参考《MICO 远程监控系统手册》，手册下载地址：http://www.co-trust.com/Download/index.html

完成以上步骤后，将 H35 的拨码开关 3 拨为 ON，启动远程控制，若 PLC 的 RM 指示灯亮起则表示远程通信成功，设备已连接至 MiCo 服务器。

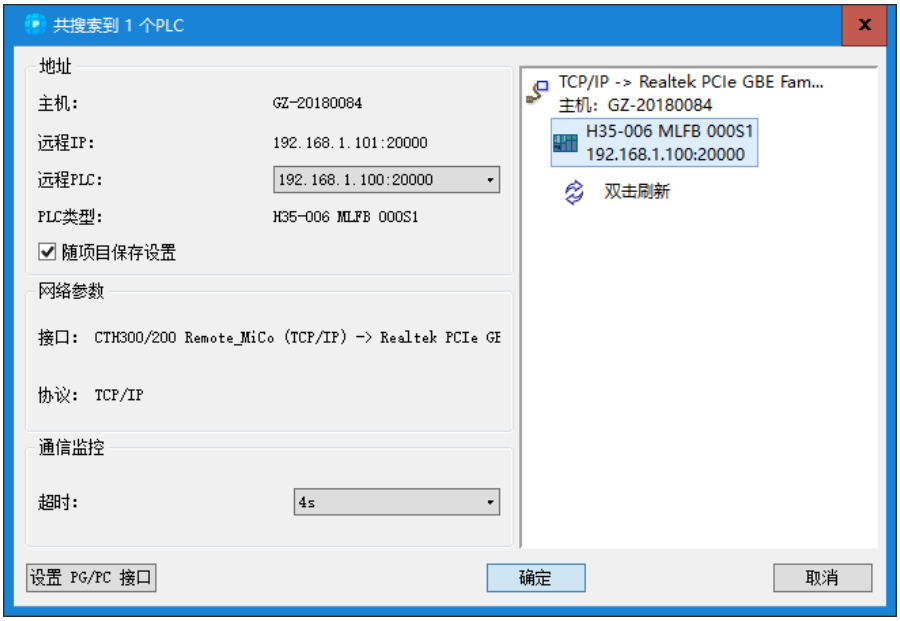
**步骤 6：设置通信连接**

1）在 MagicWorks PLC 项目管理器界面选择“工具”→“设置 PG/PC 接口”或直接双击“设

置 PG/PC 接口”，即可打开如下窗口，选择“CTH300/200 Remote_Mico(TCP/IP)->Realtek Pcle GBE Family Controller”，然后点击“确定”完成配置。




2）在 MagicWorks PLC 项目管理器界面双击“通信图标”打开通信界面，再双击图标进行 CPU 搜索，成功连接的控制器将以“H35-00”的形式显示在通信对话框中，如下图所示。




3）在以上通信界面成功搜索到 PLC 后，即表示通信成功，可进行远程编程或监控等操作。

**<备注>** 当系统出现故障时，请参考 14.5 故障诊断获取 CTH300-H 系列主控模块及扩展模块的诊断方法。

### 14.1.7 CANopen 通信

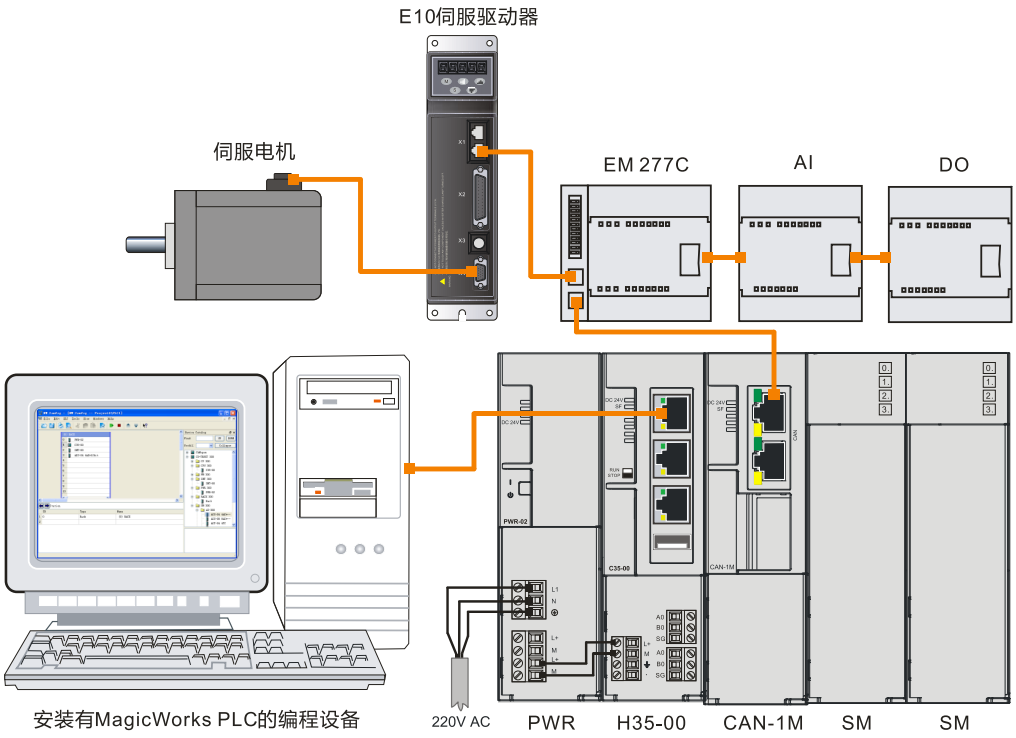
本节将通过一个具体实例来展示 CTH300-H 系列 PLC 的 CANopen 通信功能。

## 1、示例组件

表 14-20　CANopen 通信的示例组件

| 组件 | 功能 |
| --- | --- |
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 系列可编程控制器进行组态、编程和调试。 |
| 装配导轨 | CTH300 系统机架，用于固定系统中的各模块。 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电 |
| H35-00 主控模块 | CPU 执行用户程序，向 CTH300 系统背板总线提供 5V 电压，并通过以太网接口与以太网网络中的其它节点通讯。 |
| CAN-1M 主站模块 | 在 CANopen 通信中作为 CAN 主站，与 CPU 及 CAN 从站进行通信。 |
| CAN 从站设备 | EM277C、E10 伺服驱动器。 |
| EM277C 的扩展模块 | EM277C 的扩展模块 EM231 AI4、EM222 16DO。 |
| E10 伺服电机 | 与 E10 伺服驱动器相连。 |
| 标准网线 3 根 | ● 连接H35-00与编程设备<br>● 连接CAN-1M与EM 277C<br>● 连接CAN-1M与E10伺服驱动器 |
| 编码器电缆 | 连接 E10 伺服驱动器与电机 |

## 2、网络连接




## 3、操作步骤

**步骤 1：接线**

打开 H35-00、电源模块、CAN-1M 的前面板，然后参照网络连接为它们接线。具体步骤如下：

1）使用标准网线连接 PC 与 H35-00
2）H35-00 与 CAN 主站模块通过总线进行连接
3）使用标准网线连接 CAN-1M 模块与 EM277C
4）使用标准网线连接 EM277C 与 E10
5）扩展模块通过总线与 EM277C 进行连接

**步骤 2：设置通信**

在 MagicWorks PLC 中新建一个工程，在该工程中添加 H35-00 站点，然后参考章节 2.3 设置通讯将 H35-00 与 PC 进行通信连接。

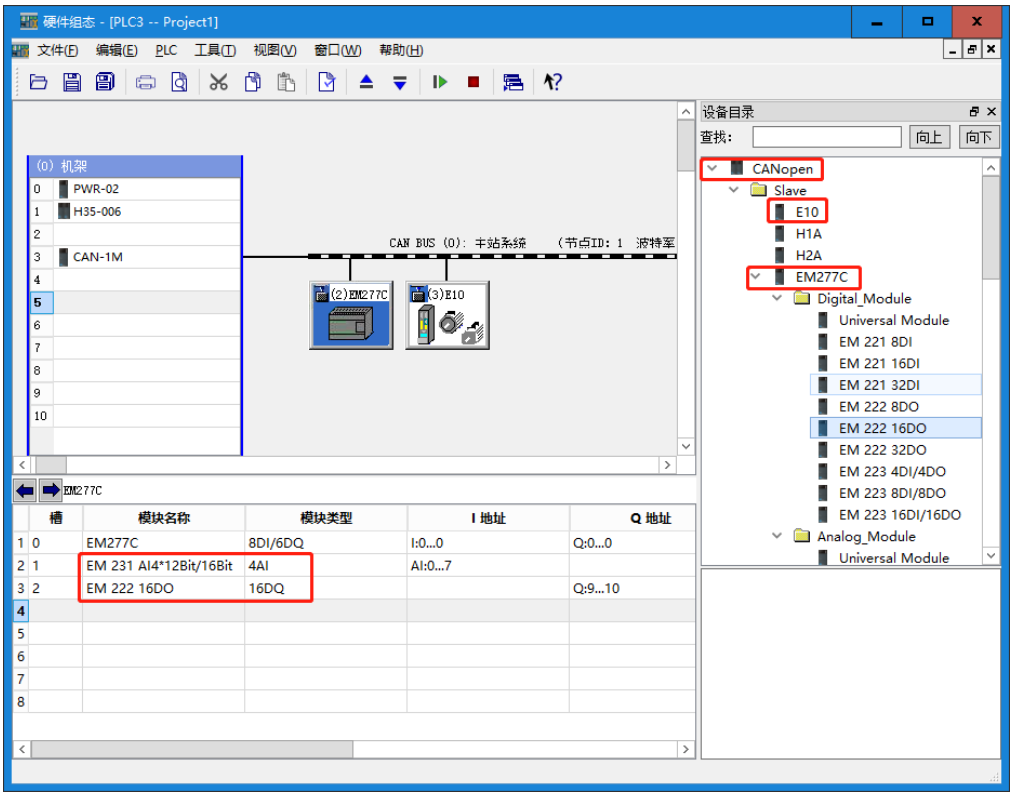
**步骤 3：在 MagicWorks PLC 中进行硬件组态**

在 MagicWorks PLC 项目视图中单击选中 H35-00 站点，然后在其右侧工作窗口双击硬件组态图标，进入硬件组态界面。

**1** 添加电源、CPU、CAN-1M 主站模块、CAN 从站模块及从站扩展模块

在硬件组态界面，选中机架 Rack 0，然后展开设备目录中的 CO-TRUST 300 节点，首先选择电源模块 PWR-02，将其拖放到机架（Rack 0）的插槽 0 中，然后选择 H35-00，将其拖放到导轨插槽 1，最后选择 CAN-1M，将其拖放到导轨插槽 3。

展开设备目录的“CANopen”→“Slave”，采用拖拽或双击的方式将 EM277C 和 E10 放置于 CAN-1M 主站模块的 CAN 总线上。然后为 EM277C 添加扩展模块 EM231 AI4、EM222 16DO。组态完成的工程如下图所示：

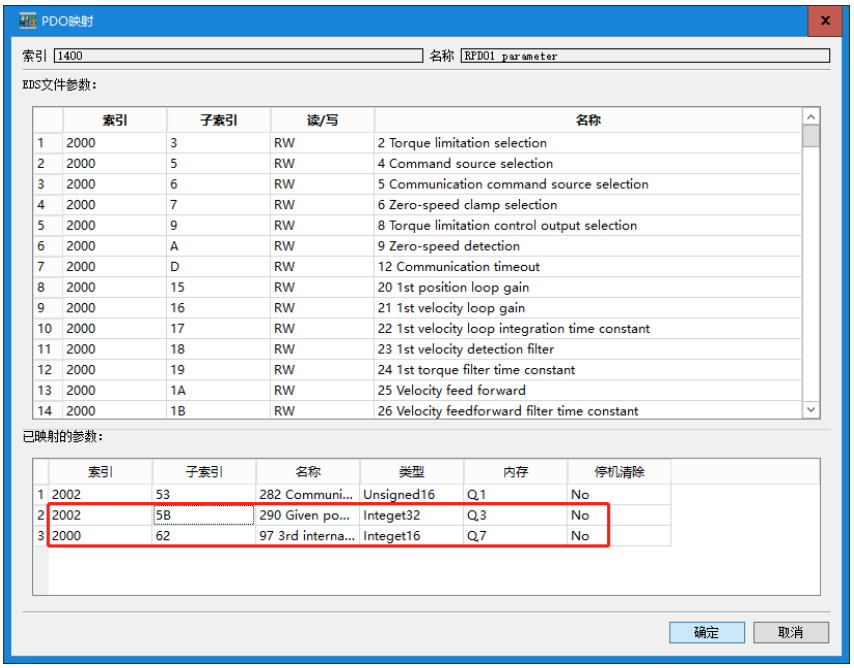



**2** 为 CAN 从站（E10）配置参数

在 CAN 总线上双击从站 E10 打开从站配置对话框，然后在从站配置对话框中双击“已映射的 PDO”中的索引 1400 打开对应的 PDO 映射（如下图所示），然后在对应的 EDS 文件参数中

分别双击 P97（设置电机的运行速度）、P290（设置给定的位置大小）。P97 和 P290 随即被添加到映射的参数列表中。

**<备注>** 双击或右键点击“已映射的参数”中的参数，即可选择配置该参数或修改其属性。

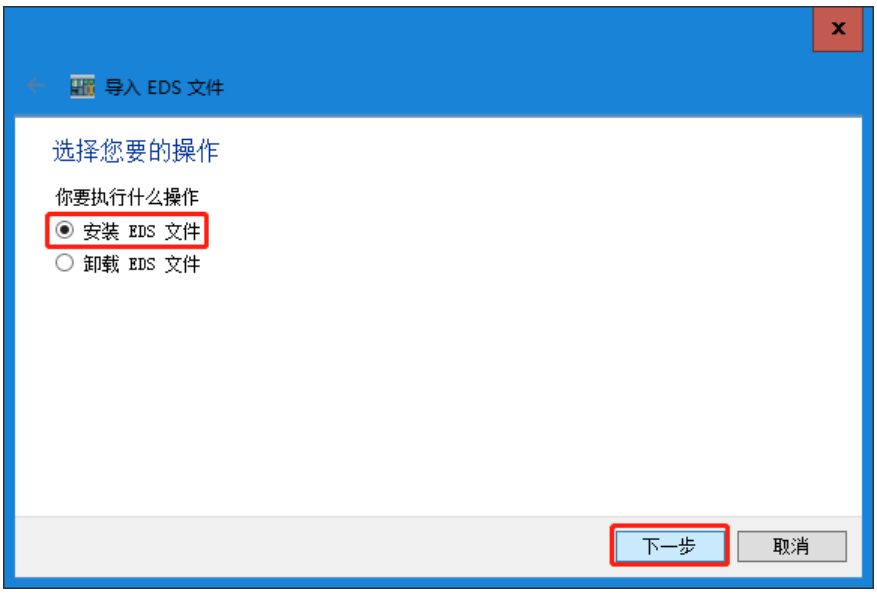



**3** 保存并编译组态

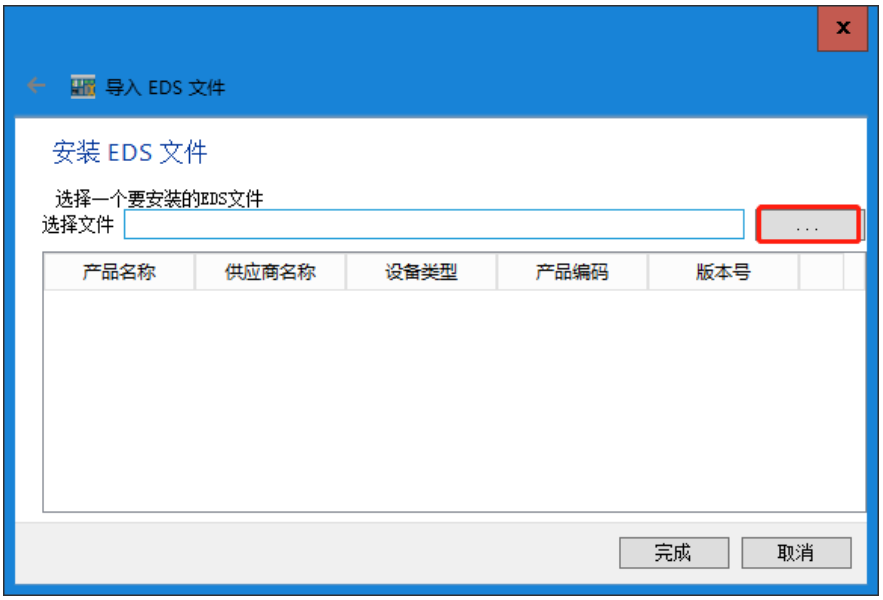
选择菜单项“文件”→“保存”以保存当前组态，然后选择菜单项“PLC”→“编译”对当前工程进行编译，若编译无误即可执行以下调试步骤。

**<注意>** 如果选用第三方 CAN 从站，则需先导入相应的 EDS 文件，以施耐德 Lexium 15 系列伺服驱动器为例，请按如下步骤导入其 EDS 文件：

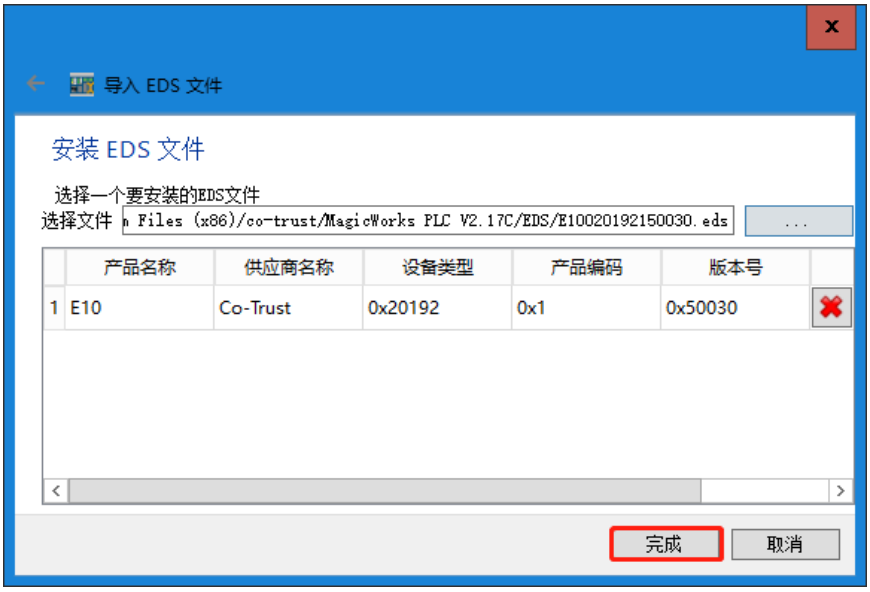
1）打开组态工程下的“硬件组态”→“工具”→“导入 EDS 文件(E)”打开如下窗口，选择“安装 EDS 文件”再点击“下一步”，如下图所示。




2）在弹出的窗口中点击“…”按钮打开 EDS 文件所在位置，如下图所示。




3）选择存储路径并添加后，改 EDS 文件将显示在窗口中，如下图所示，点击“完成”即完成导入 EDS 文件操作。




4）完成以上操作后，该文件被列入 CANopen 从站列表，用户可根据实际需求进行配置和调用。

**步骤 4：调试**

1、调试 EM 277C 及其扩展模块

在状态表中为 EM277C 的扩展模块 EM222 16DO（Q7.0~Q7.7、Q8.0~Q8.7）所有输出点写 1，成功点亮 EM222 16DO 的所有输出点则表示调试成功。

2、调试 E10 伺服驱动器及电机

1）参考《E10 使用说明书》对当前 E10 从站进行参数配置：Pr01=1（通信位置控制模式）、Pr11=1（CANopen 波特率设置为 1000Kbps）、Pr0=3（通信地址设置为 3），然后将 E10 伺服驱动器断电重启。

2）参考《E10 使用说明书》中的参数详解，在状态表中对 P97（QW4）和 P290（QW6）进行

读写。此处选择 P97=500、P290=10000，即在 QW4 中写入新值 500、QW6 中写入新值 10000。

3）选择 MagicWokrs PLC 菜单项“PLC”→“下载”将当前组态下载到 H35-00 中；然后，选择菜单项“PLC”→“运行”使 H35-00 中的应用程序开始运行，随即 E10 伺服驱动器的电机开始运转，执行完 10000 个给定位置后停止。

**<备注>** 当系统出现故障时，请参考章节 14.5 故障诊断获取 CTH300-H 系列可编程控制器的诊断方法。

### 14.1.8 EtherCAT 通信

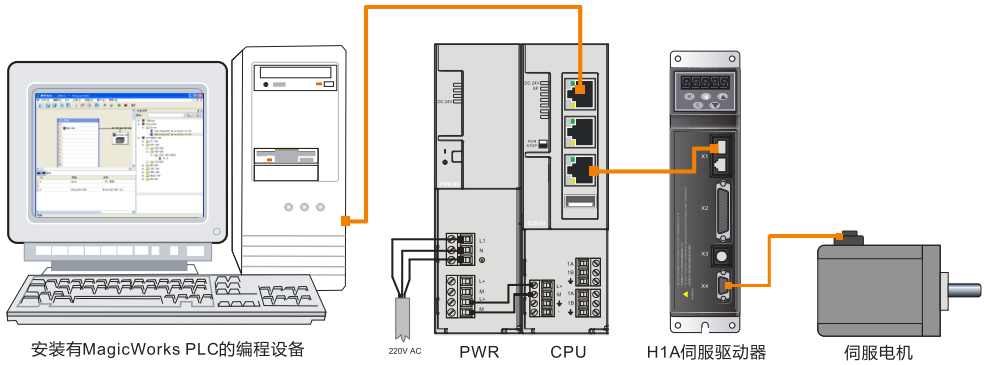
本节将通过一个具体实例来展示 H36-0X 的 EtherCAT 通信功能（以 H36-00 为例）。

#### 1、示例组件

表 14-21 EtherCAT 通信的示例组件

| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 系列可编程控制器进行组态、编程和调试。 |
| 装配导轨 | CTH300 系统机架，用于固定系统中的各模块。 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24VDC 负载电路供电 |
| H36-00 主控模块 | CPU 执行用户程序，向 CTH300 系统背板总线提供 5 V 电压，并通过以太网接口与以太网网络中的其它节点通讯。 |
| EtherCAT 从站设备 | H1A 伺服驱动器。 |
| H1A 伺服电机 | 与 H1A 伺服驱动器相连。 |
| 标准网线 2 根 | • 连接H36-00与编程设备<br>• 连接H36-00与H1A伺服驱动器 |
| 编码器电缆 | 连接 H1A 伺服驱动器与电机 |

#### 2、网络连接




**<备注>** 请使用 EtherCAT 通信口的 OUT 接口进行通信，在非冗余情况下 IN 接口不可用。

#### 3、操作步骤

**步骤 1：接线**

打开 H36-00、电源模块的前面板，然后参考网络连接接线。具体接线步骤如下：

1）使用标准网线连接 PC 与 H36-00
2）使用标准网线连接 H36-00 与 H1A 伺服驱动器
3）使用编码器电缆连接 H1A 伺服驱动器与电机

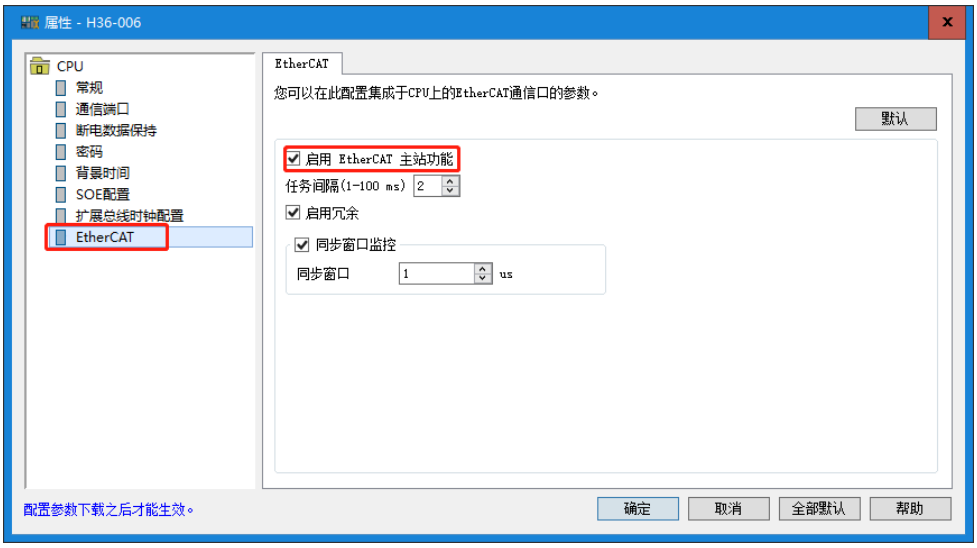
**步骤 2：设置通信**

在 MagicWorks PLC 中新建一个工程，在该工程中添加 H36-00 站点，然后参考章节 2.3 设置通讯将 H36-00 与 PC 进行通信连接。

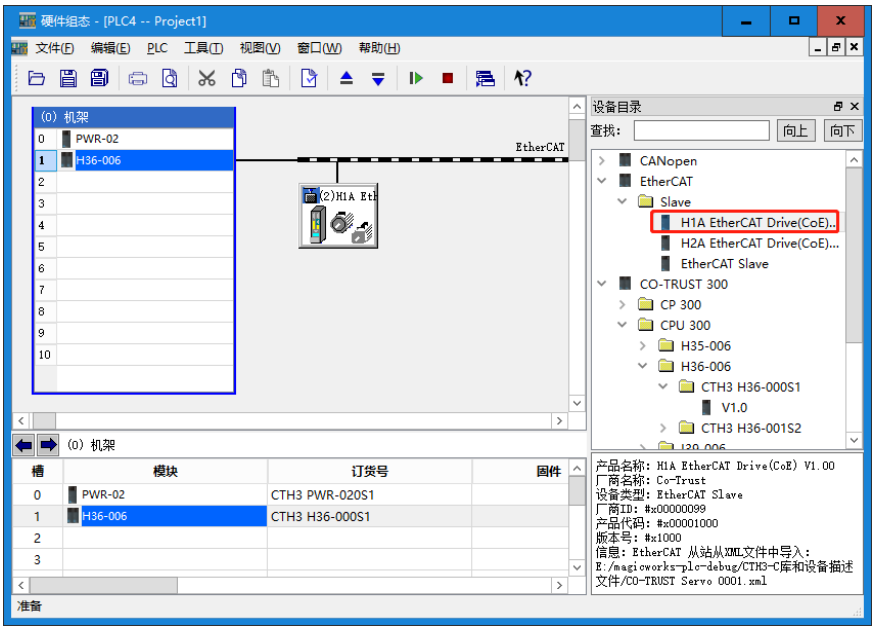
**步骤 3：在 MagicWorks PLC 中进行硬件组态**

在 MagicWorks PLC 项目视图中单击选中 H36-00 站点，然后在其右侧工作窗口双击硬件组态图标，进入硬件组态界面。

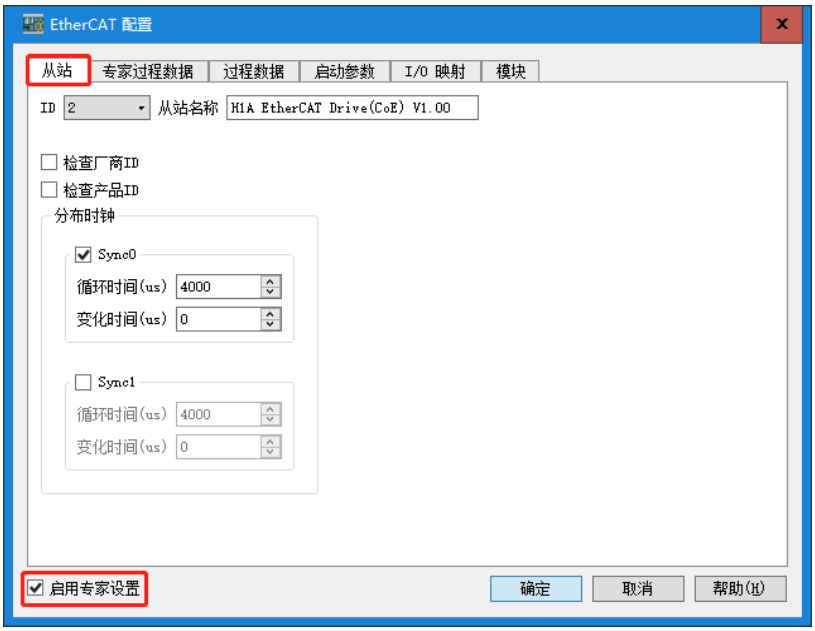
1）在硬件组态界面，通过设备目录将电源、CPU 添加到机架上，然后双击已插入的 H36-006 打开其属性对话框，选择左列的“EtherCAT”选项卡，并勾选“启用 EtherCAT 主站功能”，即可启用 EtherCAT 功能。




2）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 H1A，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。

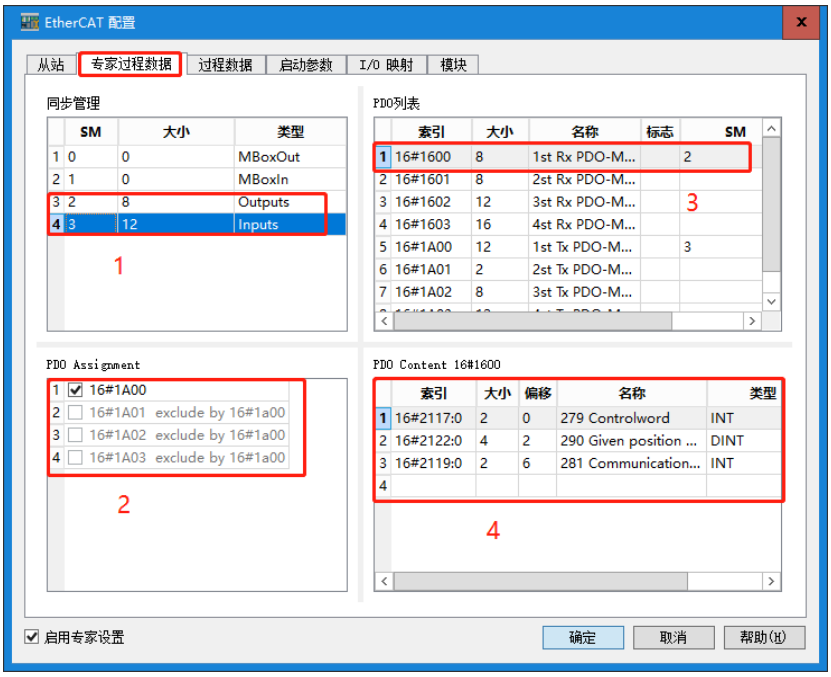



3）双击 EtherCAT 总线上的 H1A 从站图标，即弹出如下图所示的 EtherCAT 配置对话框，具体的参数配置参考如下描述。勾选“启用专家设置”即可配置专家过程数据。

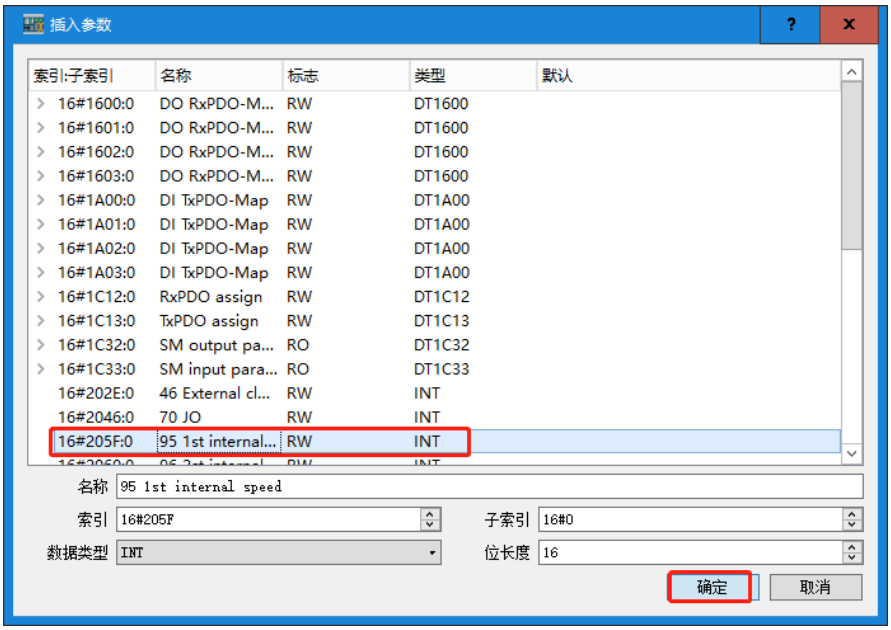



◆ “专家过程数据”选项卡

通过同步管理选择输入/输出，然后在 PDO Assignment 中勾选所需的输入/输出组，在 PDO 列表选中对应的组，最后在 PDO Content 中为该组添加或删除参数。

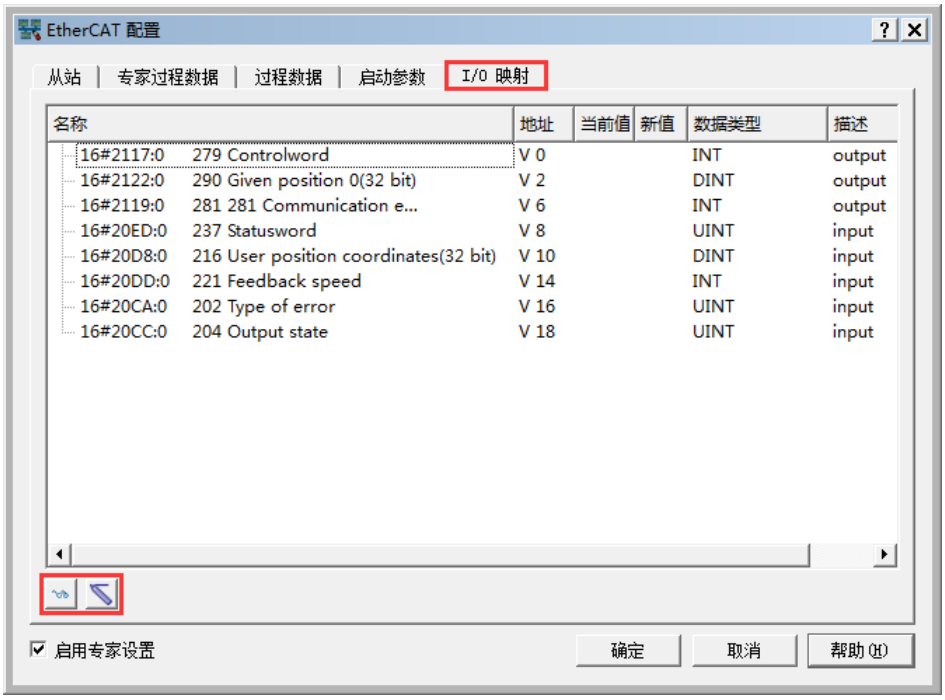



如需为特定 PDO 参数添加变量，请选中该参数，在 PDO Content 区域单击右键后选择“插入”，随即显示添加变量的设置窗口，选中要添加的变量，点击“确定”，如下图所示：




◆ “I/O 映射”选项卡

配置成功的 I/O 参数将显示在 I/O 映射中，双击地址列可以编辑其地址。

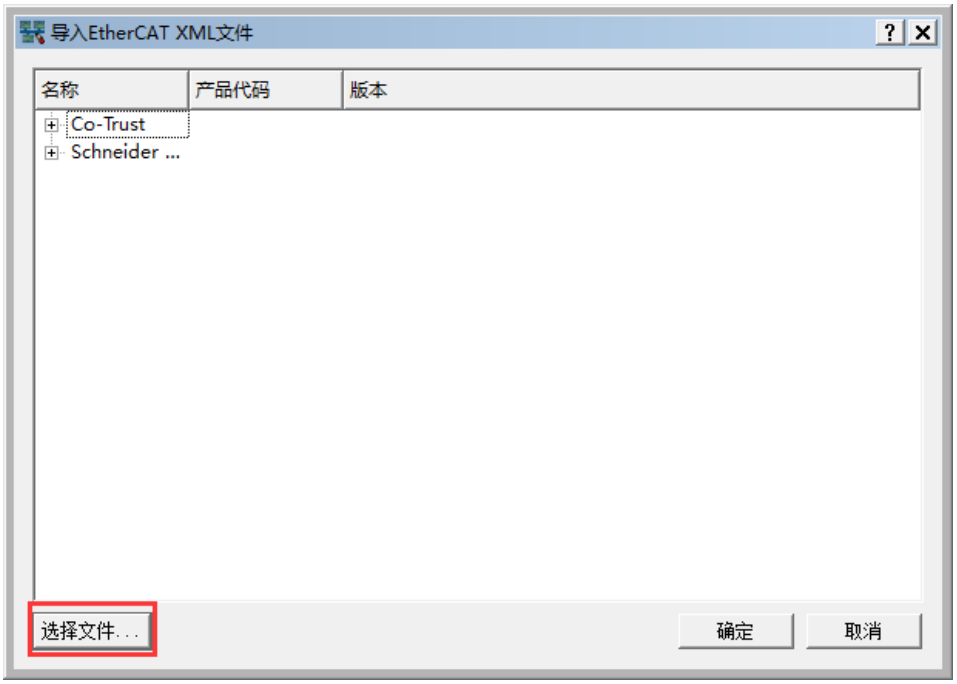



4）在 EtherCAT 配置对话框的“I/O 映射”选项卡中点击监控按钮，可以开始监控已配置的参数，然后在“新值”列输入参数值，最后点击写入按钮，即可成功写入新值。

**导入第三方从站**

如果选用第三方 EtherCAT 从站，则需先导入相应的 XML 设备描述文件，以欧姆龙 R88D-KN 系列伺服为例，请按如下步骤导入其 XML 文件：

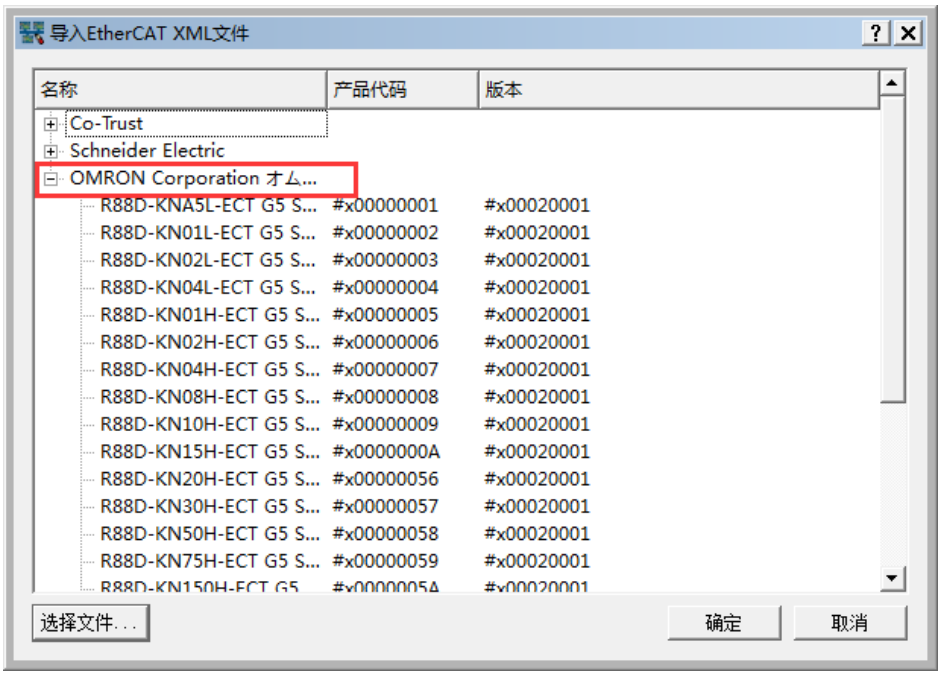
◆ 打开组态工程下的“硬件组态”→“工具”→“导入 EtherCAT XML 文件(X)”，随即显示以下界面：



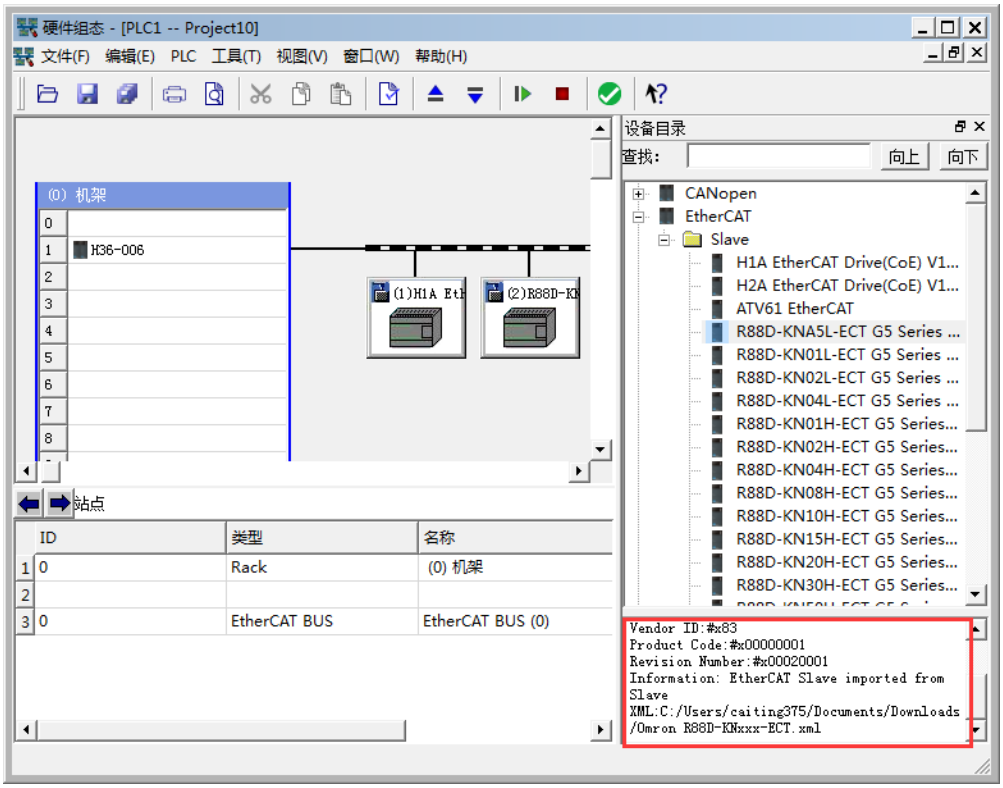
◆ 在打开的窗口中点击“选择文件”，并在相应的存储路径下选择 XML 文件，弹出以下窗口：




◆ 点击“确定”后，前一个界面中将列出添加的 XML 文件和所属制造商：




◆ 点击“确定”后相关伺服将被列入 EtherCAT 从站列表，窗口中会显示产品所属制造商和相关信息，单击“完成”结束导入操作，之后用户可以根据实际需求对第三方从站进行配置和调用，如下图所示。




**提示**

当系统出现故障时，请参考章节 14.5 故障诊断获取 CTH300-H 系列主控模块的诊断方法（EtherCAT 寄存器：SMB400~SMB465）。

### 14.1.9 西门子 S7 协议通信举例

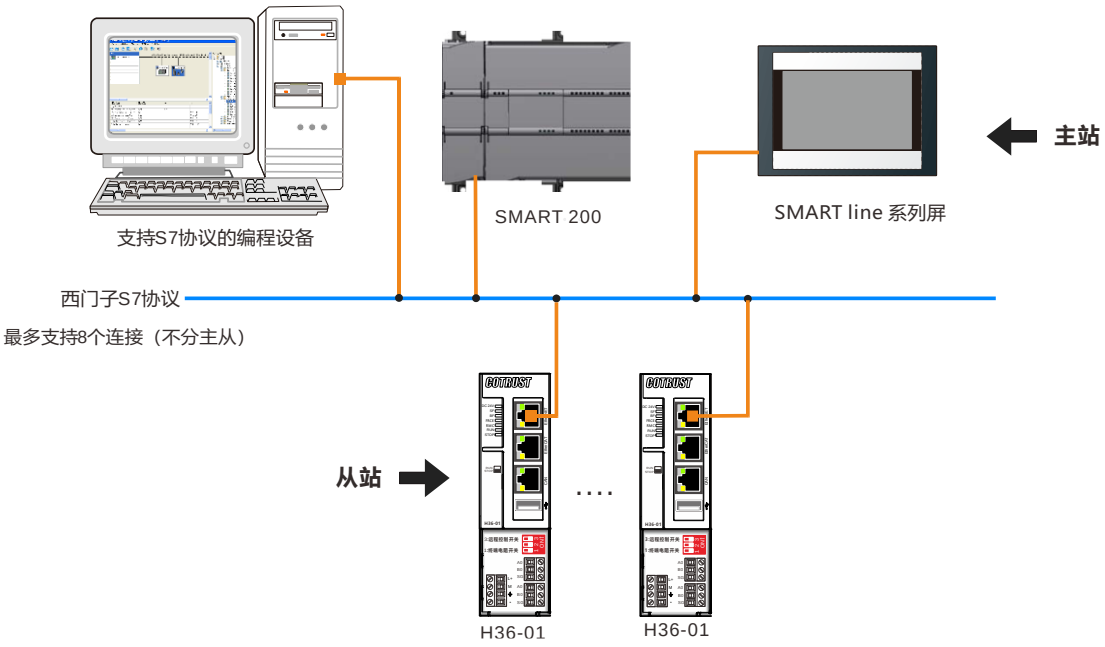
CTH300-H 系列 H32-01、H36-01 支持西门子 S7 协议主从站功能，最多支持 8 个连接（不分主从）。西门子 PLC 或屏或支持 S7 协议的上位机软件作为主站，CTH300-H 系列 H32-01、H36-01 作为从站，即可进行 S7 协议通信，此外 H32-01、H36-01 也可以作主站。本节将通过具体实例来展示合信 CTH300-H 系列 H32-01、H36-01 与西门子触摸屏 PLC 及第三方支持西门子 S7 协议的触摸屏与上位机通讯的参数配置操作。

**1、示例组件**

| 组件 | 功能 |
|---|---|
| 编程设备 | 安装有 STEP 7-MicroWin SMART 软件，对西门子 SMART200 PLC 进行组态、编程和调试。 |
| CTH300-H 系列 PLC | 作为 S7 协议从站，与 S7 协议主站进行通信。 |
| 西门子 SMART200 PLC | 作为 S7 协议主站，与 S7 协议从站进行通信。 |
| 组态王 | 软件版本 V6.55，作为 S7 协议主站，与 S7 协议从站进行通信。 |
| 昆仑通态触摸屏 | 作为 S7 协议主站，与 S7 协议从站进行通信。 |
| 西门子触摸屏 | SMART 700 IE V3，作为 S7 协议主站，与 S7 协议从站进行通信。 |
| 威纶通触摸屏 | 作为 S7 协议主站，与 S7 协议从站进行通信。 |

## 2、网络连接

如下为 H36-01 配合西门子等设备或上位机软件进行 S7 协议通信的典型连接：




### *西门子产品与 CTH300-H 系列产品 CPU 的网口通讯*

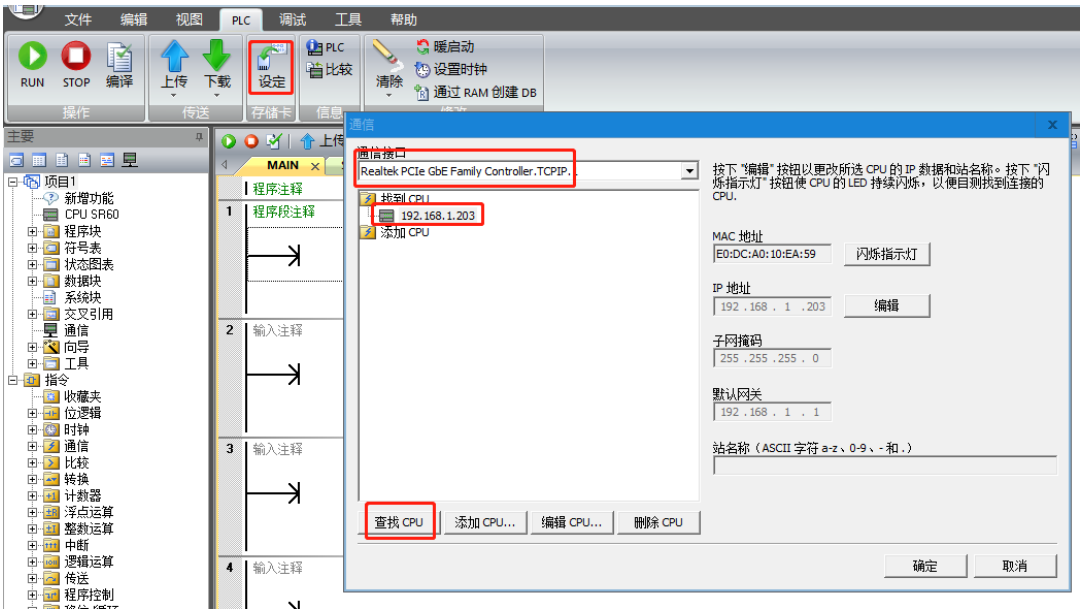
**示例 1：SMART200 与 H36-01 的 S7 协议通信**

本例以西门子 SMART200 为主站，CTH300-H 系列 H36-01 做从站，进行 S7 协议的通信。

**<备注>：**西门子触摸屏也可以作为主站与 H36-01 进行 S7 协议的通信。

1、SMART200 PLC 与 STEP 7-MicroWin SMART 编程软件通信设置

打开 STEP 7-MicroWin SMART 软件，将 SMART200 PLC 与电脑用 RJ45 网线连接（电脑与 PLC 保持在同一个网段内），打开 SMART 软件设置界面，选择电脑对应的本地网卡，然后点击查找 CPU 即可找到 PLC。

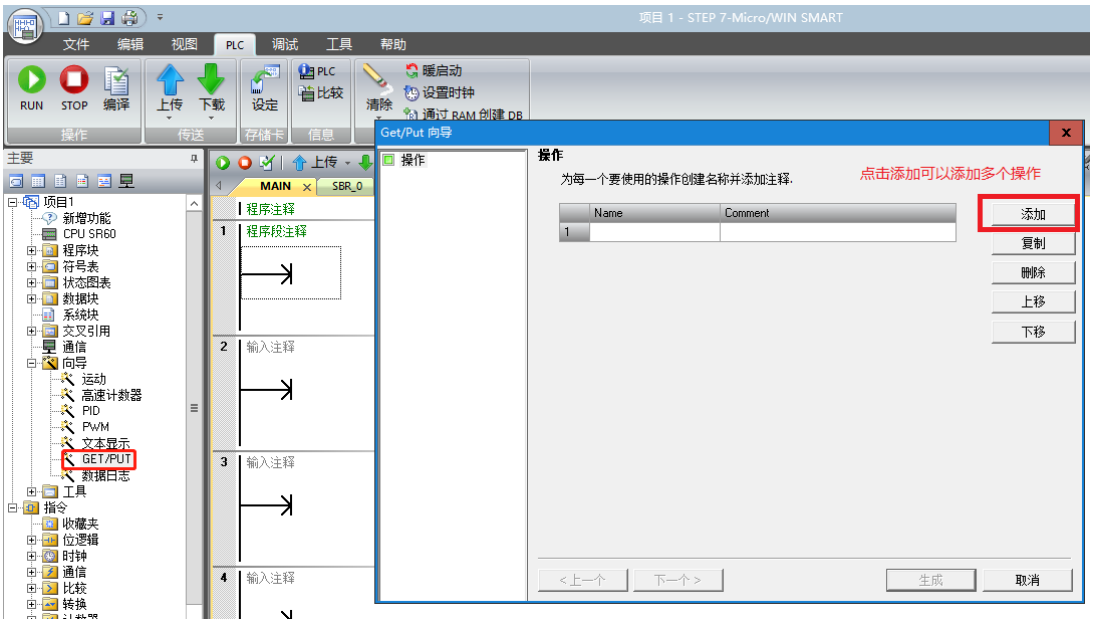


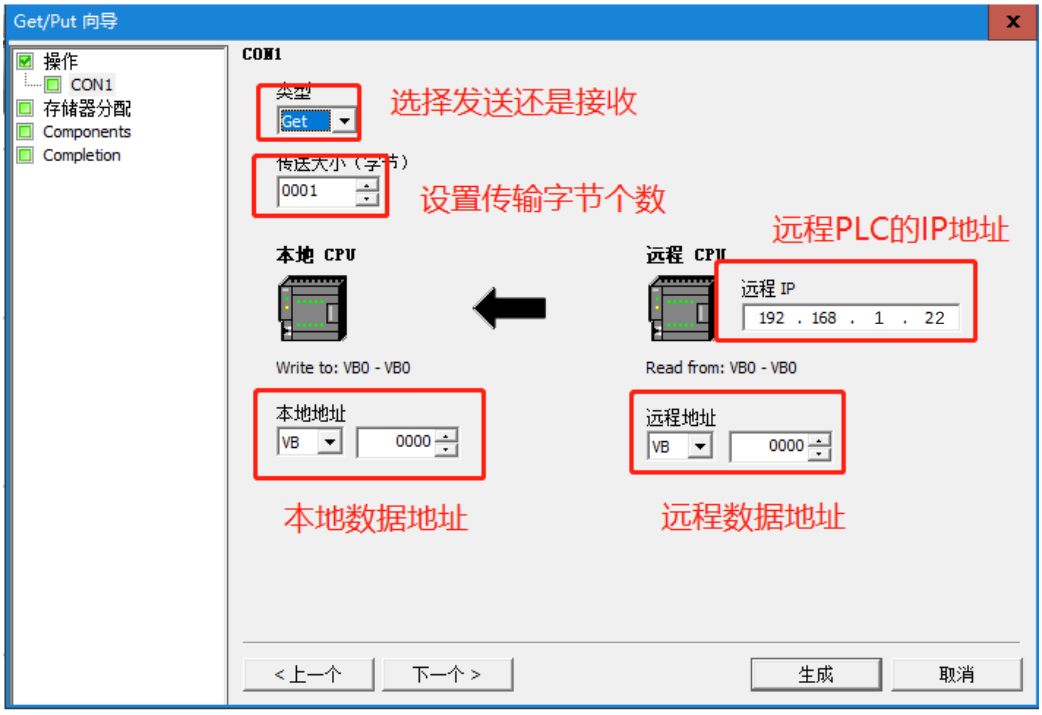

2、SMART200 PLC 做主站，CTH300-H 系列 PLC 做从站程序设置。

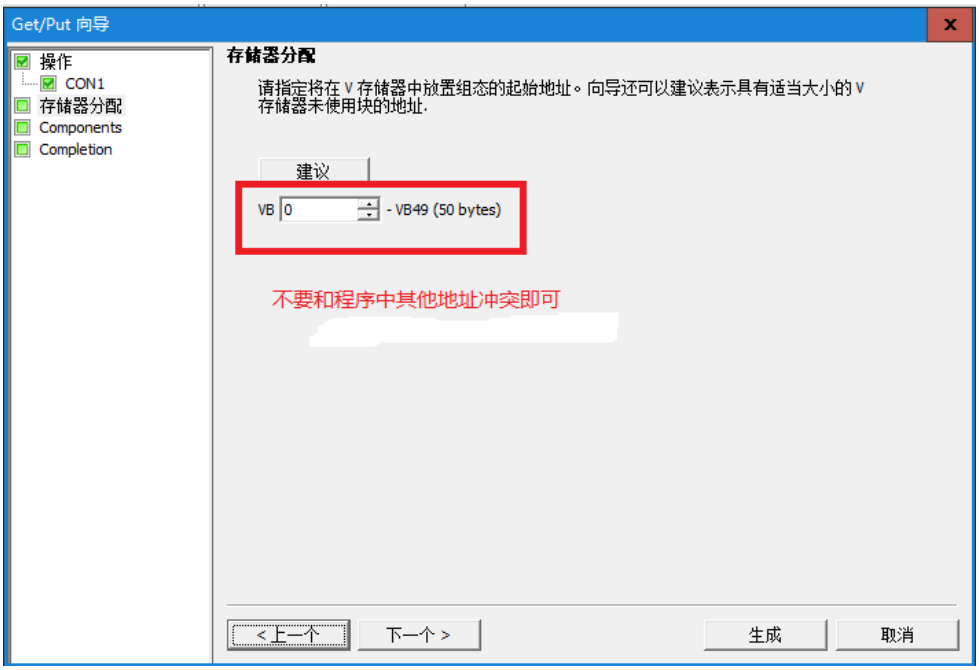
**主站配置**

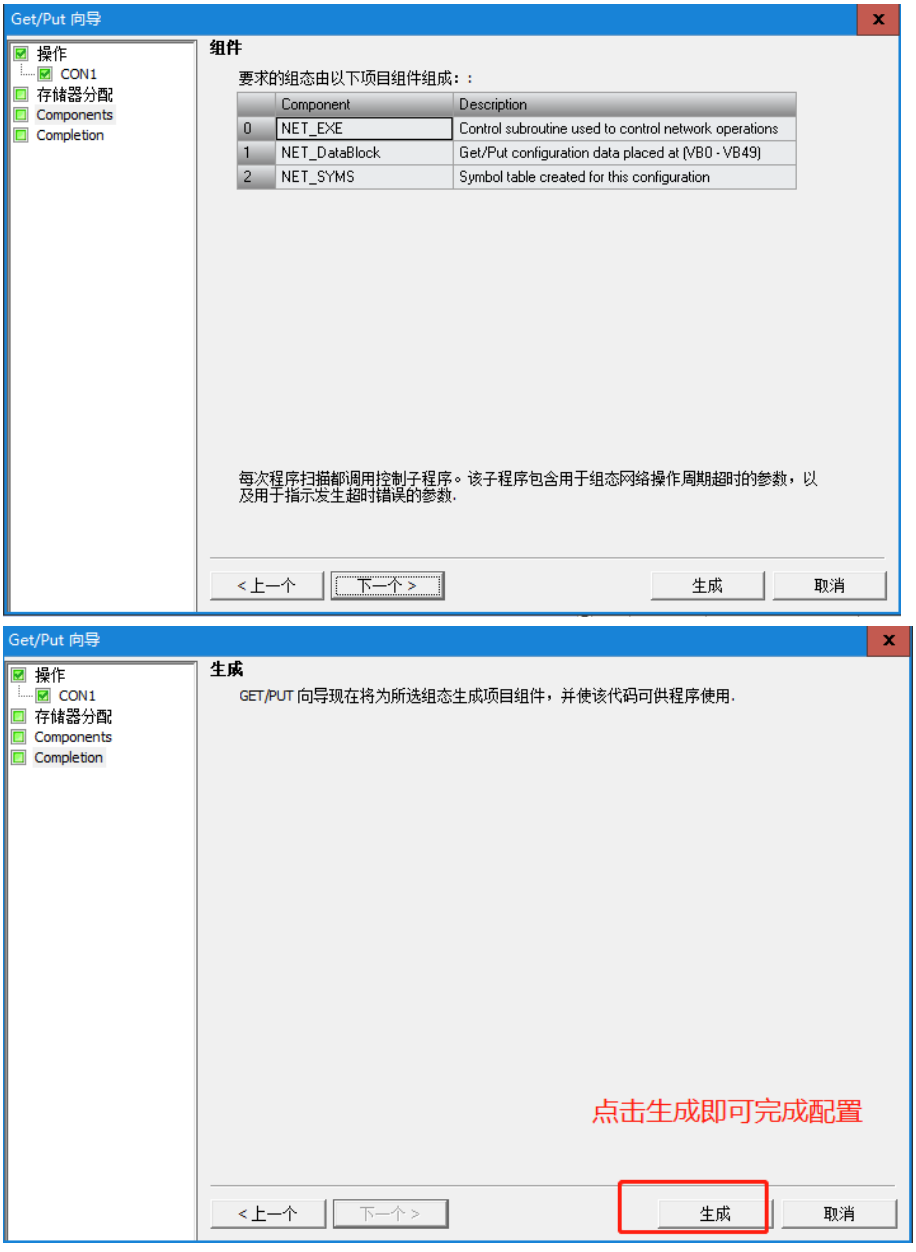
1）建立 GET/PUT 向导

在 STEP 7-MicroWin SMART 软件中打开 GET/PUT 向导，并进行如下设置：

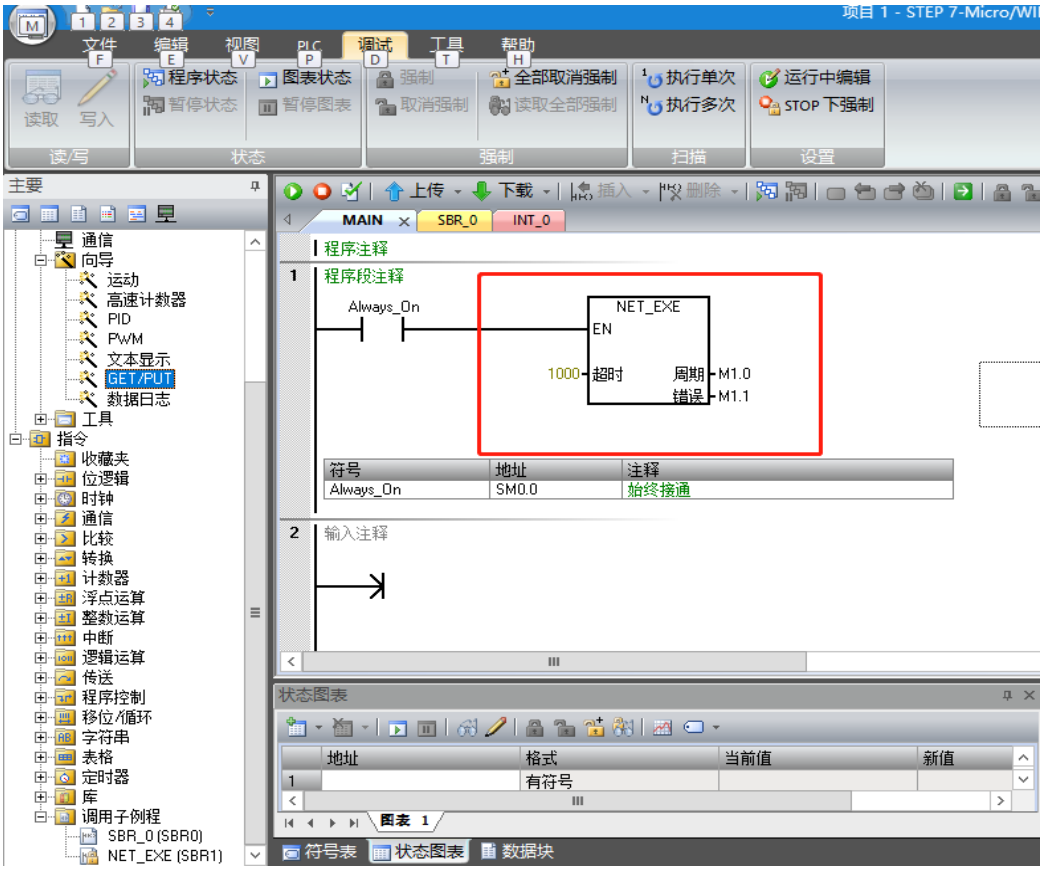


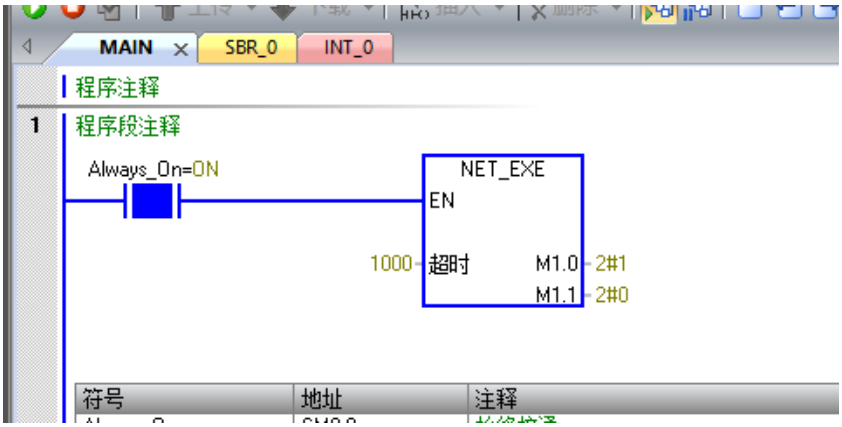

Get/Put 向导
操作
CON1
存储器分配
Components
Completion
CON1
类型
Get
选择发送还是接收
传送大小（字节）
0001
设置传输字节个数
远程PLC的IP地址
本地 CPU
远程 CPU
远程 IP
192 . 168 . 1 . 22
Write to: VB0 - VB0
Read from: VB0 - VB0
本地地址
VB
0000
远程地址
VB
0000
本地数据地址
远程数据地址
< 上一个
下一个 >
生成
取消


Get/Put 向导
操作
CON1
存储器分配
Components
Completion
存储器分配
请指定将在 V 存储器中放置组态的起始地址。向导还可以建议表示具有适当大小的 V 存储器未使用块的地址.
建议
VB 0 - VB49 (50 bytes)
不要和程序中其他地址冲突即可
< 上一个
下一个 >
生成
取消




2）在 SMART 软件的程序里面用 sm0.0 调用向导生成的块。

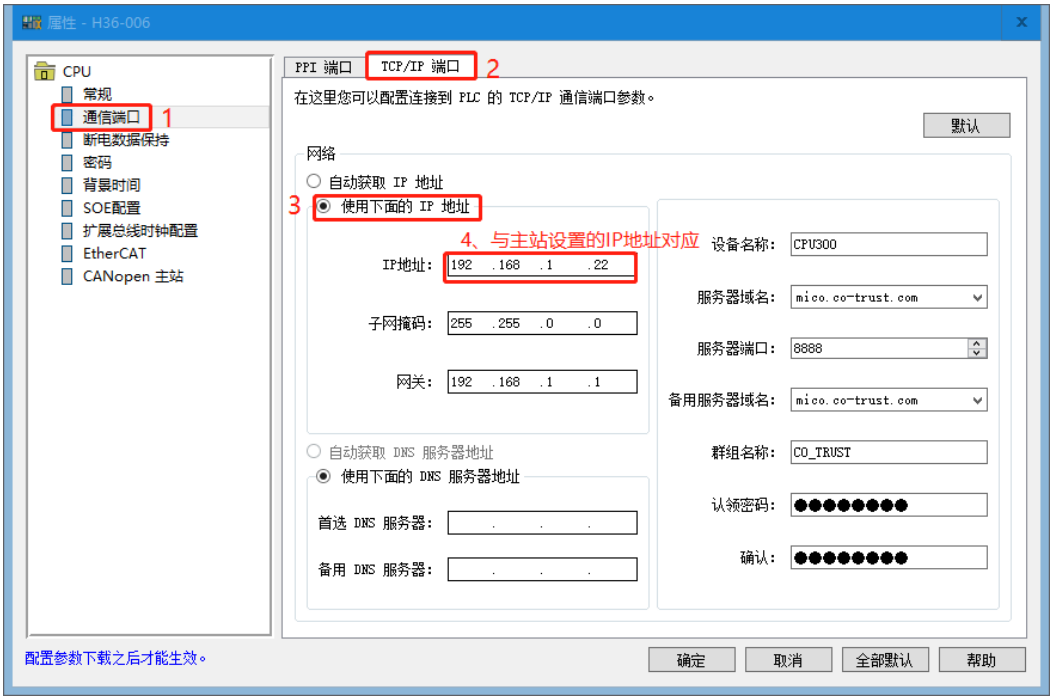



**从站配置**

在 CTH300-H 系列 PLC 的硬件组态中配置好从站 IP 地址即可，操作示意如下：

在机架上添加一个 H36-01CPU，双击该模块以后进行以下操作即可。




通过以上步骤将 S7 协议主从站设置完成后，即可实现西门子 SMART200 与 H36-01 进行的 S7 协议通信。

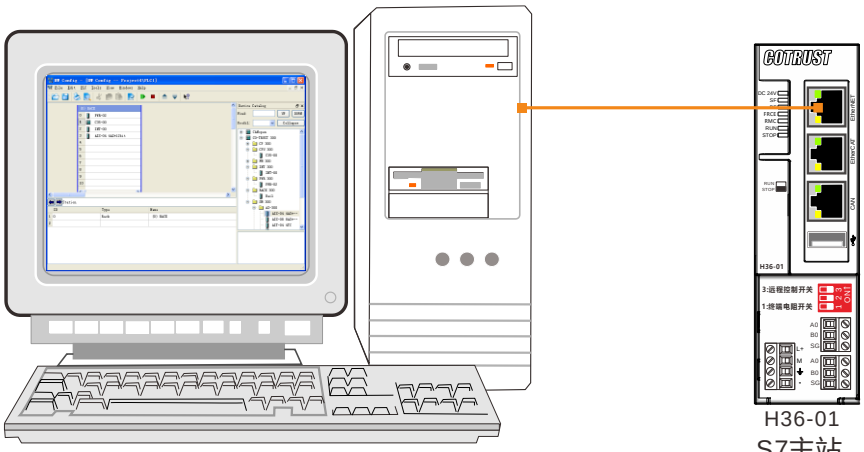
## 示例 2：CTH300-H 系列 CPU 与 SMART200 的 S7 协议通信

本节以 CTH300-H 系列 H36-01 为主站，西门子 SMART200 做从站，通过具体实例来展示主站 H36-01 PLC 与从站西门子 SMART200 PLC 通讯的参数配置操作。

示例组件

| 组件 | 功能 |
| --- | --- |
| 编程设备 | 安装有 Magicworks PLC，对 CTH300-H 系列 CPU H36-01 进行组态、编程和调试。 |
| H36-01 CPU | 作为 S7 协议主站，与 S7 协议从站进行通信。 |
| 西门子 SMART200 PLC | 作为 S7 协议从站，与 S7 协议主站进行通信。 |
| 标准网线 | • 连接 H36-01 PLC 与编程设备。<br>• 连接 H36-01 PLC（S7 主站）与西门子 SMART200 PLC（S7 从站）。 |

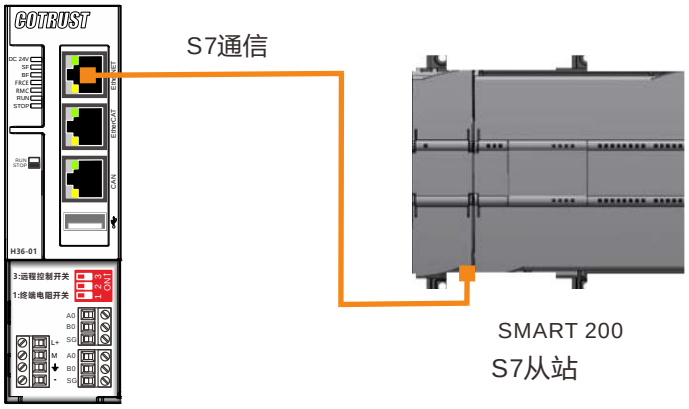
使用标准网线连接编程设备与 H36-01 PLC，通过编程设备对 S7 主站（H36-01 PLC）进行编程：




安装有Magicworks PLC的编程设备

H36-01
S7主站

使用标准网线连接 S7 主站和 S7 从站，S7 主站将进行一个写操作，将指定地址中的数据写入到 S7 从站中，然后通过 S7 从站读取相应地址中的数据，继而实现 S7 通信：

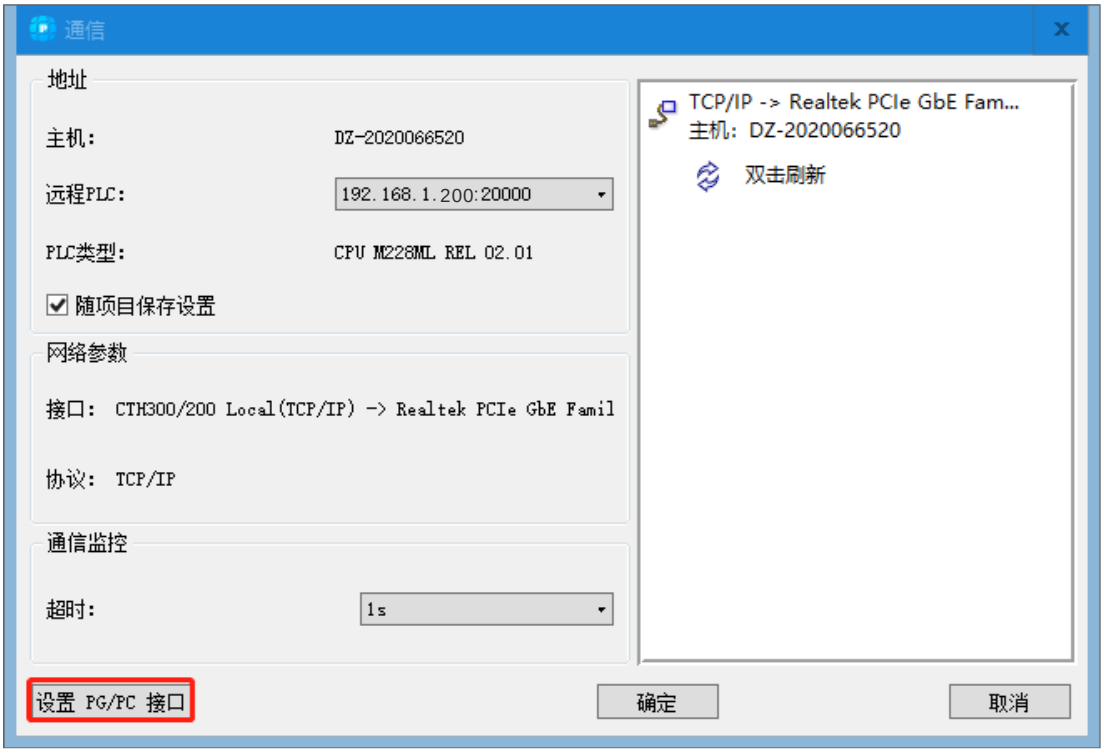



SMART 200
S7从站

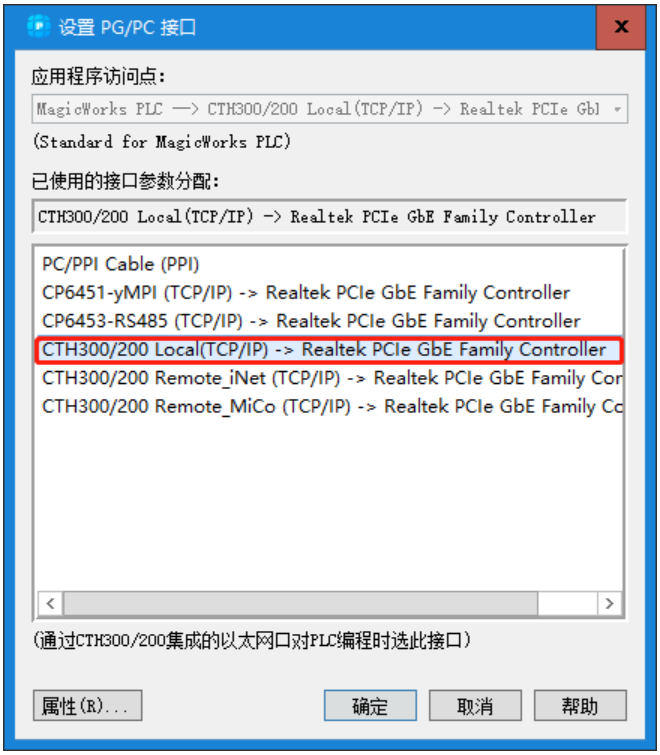
H36-01
S7主站

**1、主站 H36-01 PLC 与 Magicworks PLC 编程软件通信设置**

打开 Magicworks PLC 软件，将 H36-01 PLC 与电脑用 RJ45 网线连接，电脑与 PLC 保持在同一个网段内，打开通信界面，点击设置 PG/PC 接口。



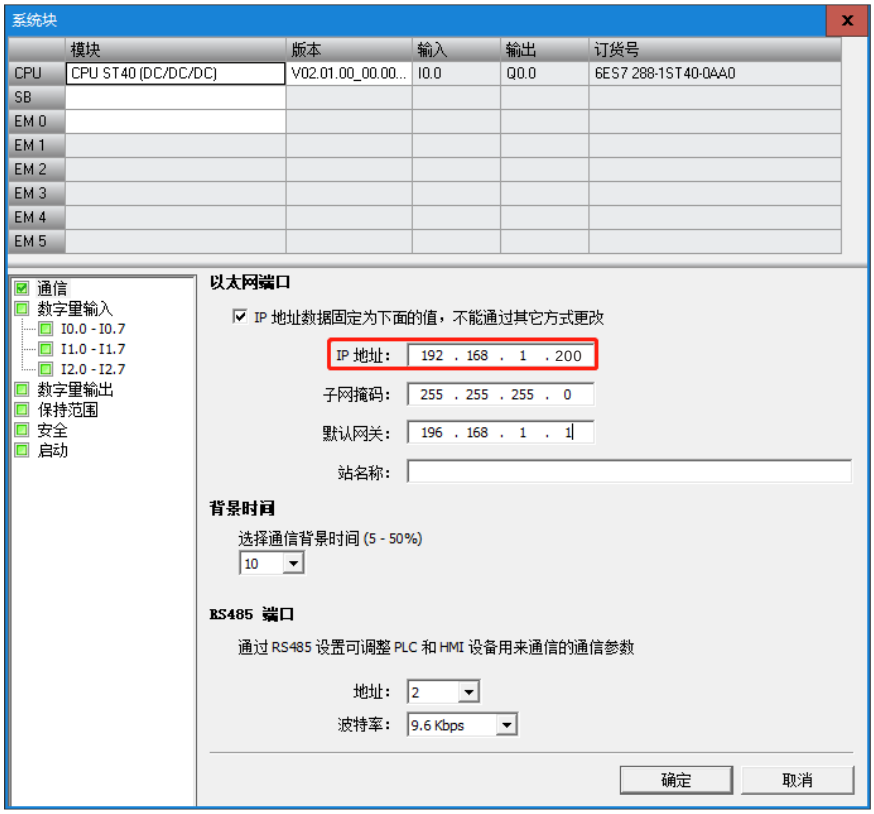

选择“CTH300/200Local（TCP/IP）->Realtek PLCe GbE Family Controller”，然后确定。




在通信界面双击刷新，即可出现 CPU H36-01，点击 CPU H36-01 确定即可。

**从站配置**

打开 STEP7-MicroWin SMART，点击项目树下的“系统快”，在系统块中设置从站 IP，IP 地址与主站对应好即可。




**2、对 S7 主站（CPU H36-01）进行编程**

以 S7 网络读写为例，您可以通过以下方法对 S7 主站进行网络读写操作。

通过 S7 网络读写指令 S7_read/S7_write 为 S7 主站编程。

S7_read/S7_write 指令的 TABLE 参数表：

| D | A | E | 0 | 错误代码 | 0 |
|---|---|---|---|---|---|
| 远程站 IP 第一个字节 | | | | | 1 |
| 远程站 IP 第二个字节 | | | | | 2 |
| 远程站 IP 第三个字节 | | | | | 3 |
| 远程站 IP 第四个字节 | | | | | 4 |
| 保留（必须设为 0） | | | | | 5 |
| 保留（必须设为 0） | | | | | 6 |
| 指向远程站目标区域指针的第 1 个字节 | | | | | 7 |
| 指向远程站目标区域指针的第 2 个字节 | | | | | 8 |
| 指向远程站目标区域指针的第 3 个字节 | | | | | 9 |
| 指向远程站目标区域指针的第 4 个字节 | | | | | 10 |
| 长度（不大于 200） | | | | | 11 |
| 指向本地站目标区域指针的第 1 个字节 | | | | | 12 |
| 指向本地站目标区域指针的第 2 个字节 | | | | | 13 |
| 指向本地站目标区域指针的第 3 个字节 | | | | | 14 |
| 指向本地站目标区域指针的第 4 个字节 | | | | | 15 |

D:完成位，0 未完成，1 完成

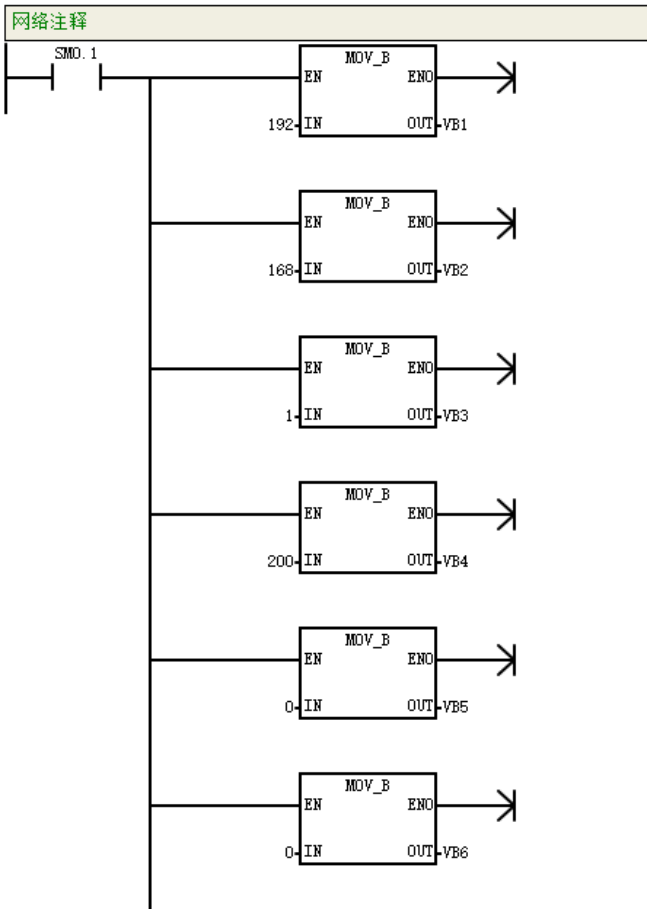
A:激活位，指令正在执行中为 1，未执行为 0

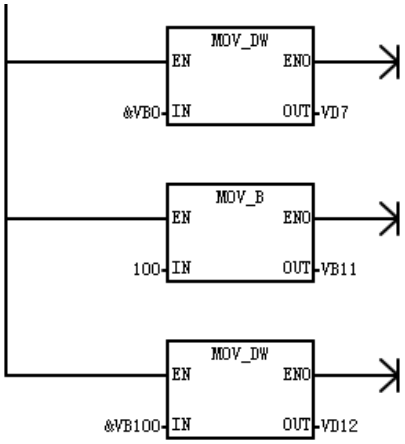
E:位，0 无错，1 出错

**错误代码表**

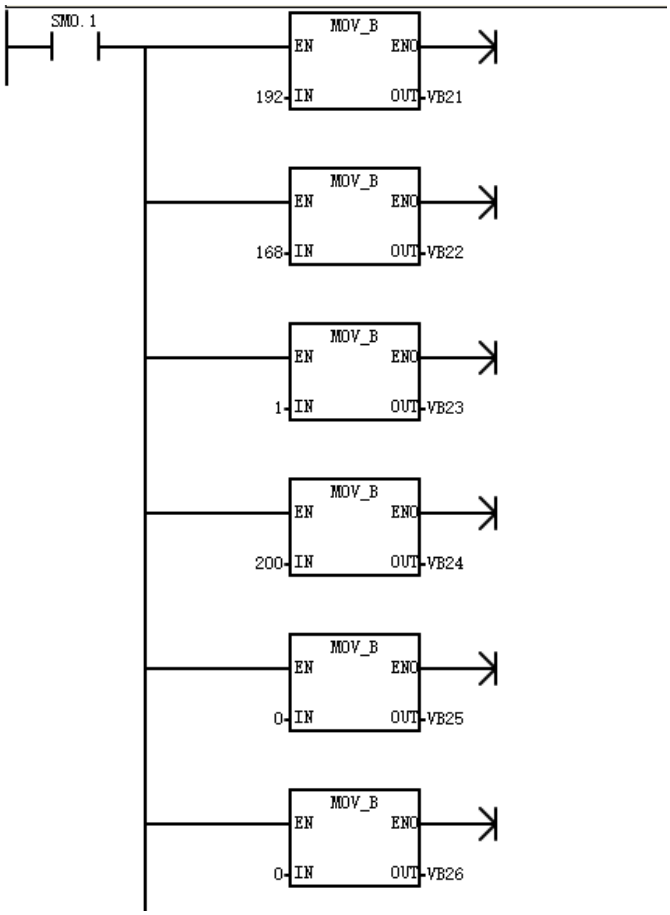
| 代码 | 说明 |
|---|---|
| 0 | 无错 |
| 1 | 超时错误：远程站不应答 |
| 2 | 保留 |
| 3 | 脱机错误：重复站址或故障硬件引起的冲突 |
| 4 | 队列溢出错误：8 个 S7_write/S7_read 方框被激活 |
| 5 | 接收错误：收到的回复有错误 |
| 6 | 非法参数：S7_write/S7_read 表格包含一个非法或无效数值 |
| 7 | 保留 |
| 8 | 保留 |
| 9 | 信息错误：数据长度不正确 |
| 10 | 连接数超过 8 个 |
| 11 | 网线没插 |

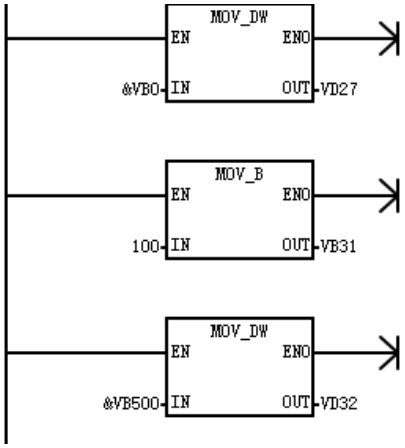
***网络 1：***根据 S7_read/S7_write 指令的 TABLE 参数表将远程站的 IP 设为 192.168.1.200，远程站目标区域指针设为&VB0（一定要用指针）,读的长度设置为 100 字节,本地站目标区域指针设为&VB100（一定要用指针），配置好参数表。



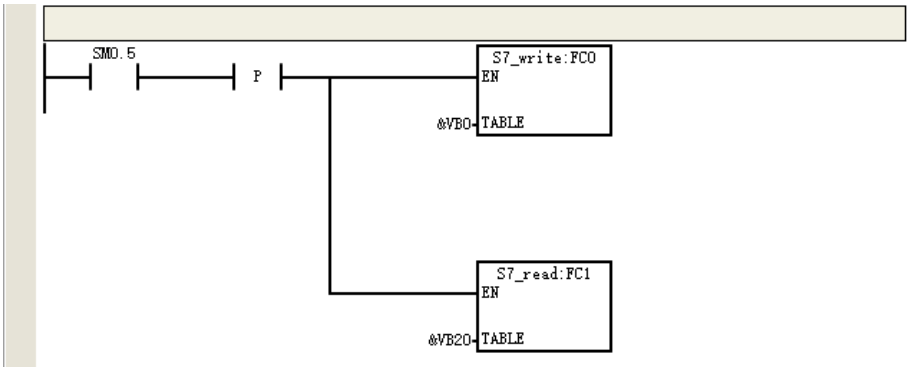



***网络 2：***根据 S7_read/S7_write 指令的 TABLE 参数表，配置好参数表。将远程站的 IP 设为 192.168.1.200，远程站目标区域指针设为&VB0（一定要用指针）,写的长度设置为 100 字节, 本地站目标区域指针设为&VB500（一定要用指针）。





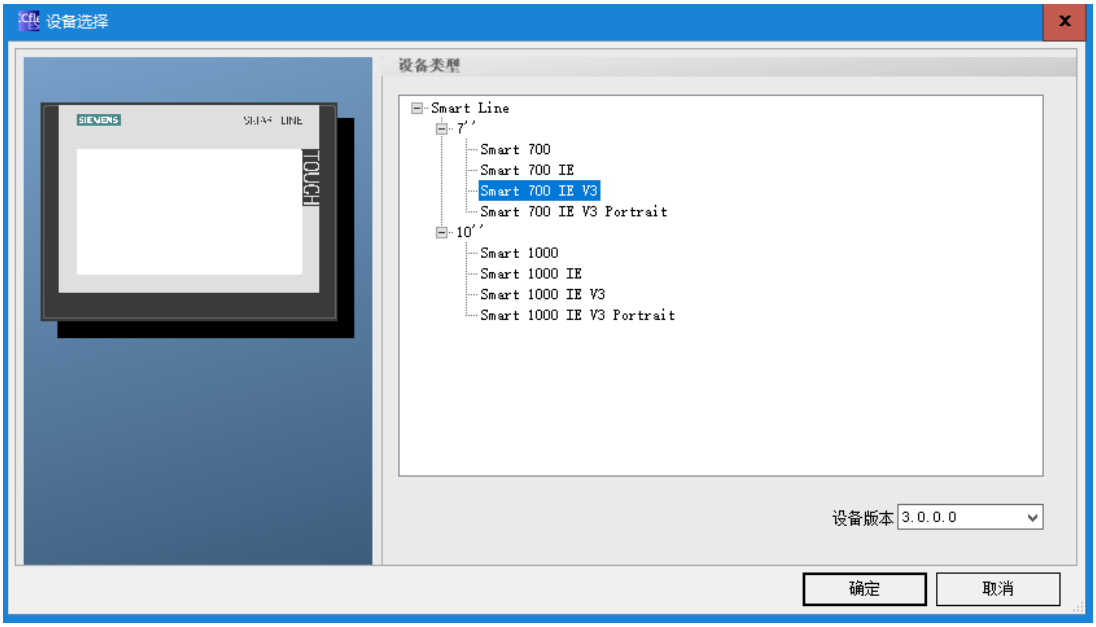

***网络 3：***调用 S7_read/S7_write 指令，通过参数表的参数进行读写数据，指令的 TABLE 参数务必使用指针。

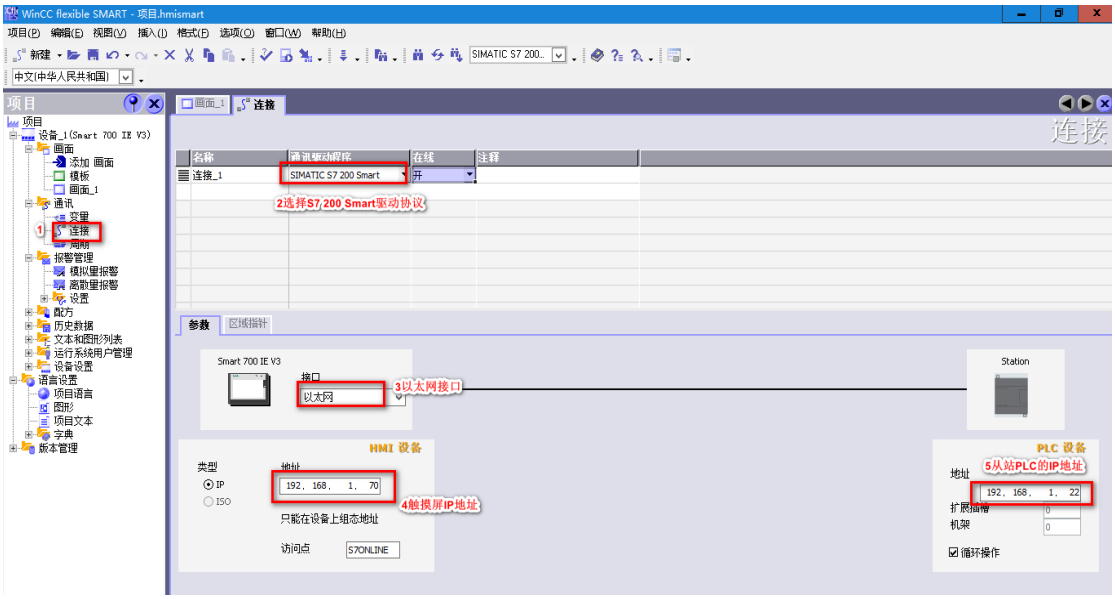



## 示例 3：西门子触摸屏与 H36-01 的 S7 协议通信

本例以西门子触摸屏为主站，CTH300-H 系列 PLC 做从站，进行 S7 协议的通信。

**主站配置**

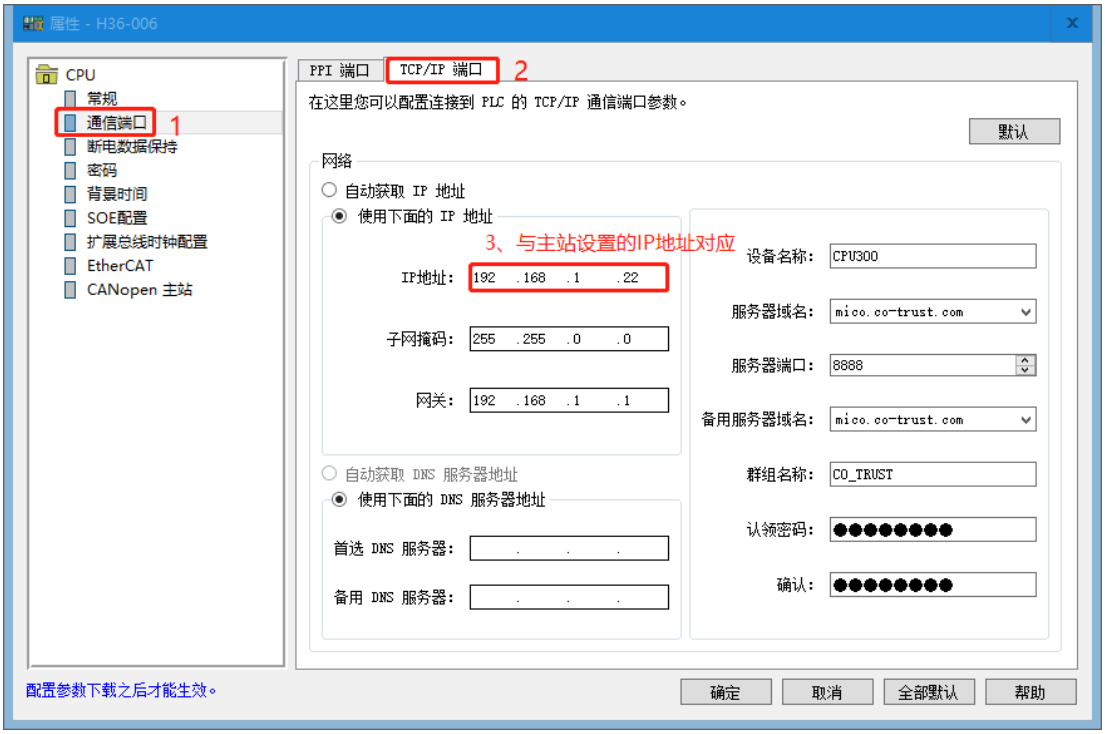



## 从站配置

在 CTH300-H 系列 PLC 的硬件组态中配置好从站 IP 地址即可，操作示意如下：

在机架上添加一个 H36-01CPU，双击该模块以后进行以下操作即可。




## 示例 4：S7-1200/1500 与 CTH300-H 系列 PLC 的的 S7 协议通信

S7-1200/1500 CPU 与 CTH300-H 系列 PLC 的的 S7 协议通信（S7-1200/1500 作为主站），合信 CTH300-H 系列 PLC（H32-01、H36-01）支持西门子 S7 协议从站功能，西门子 PLC 或屏或支持 S7 协议的上位机软件作为主站，CTH300-H 系列 PLC（H32-01、H36-01）作为从站，即可进行 S7 协议通信。

| 硬件和软件需求及所完成的通信任务 | |
|---|---|
| **硬件** | ● S7-1200 CPU 硬件版本 V2.0 或更高<br>● H36-01<br>● PC （带以太网卡）<br>● TP 以太网电缆<br>● 工业交换机 |
| **软件** | ● TIA portal V15<br>● MagicWorks PLC |

**所完成的通信任务：**

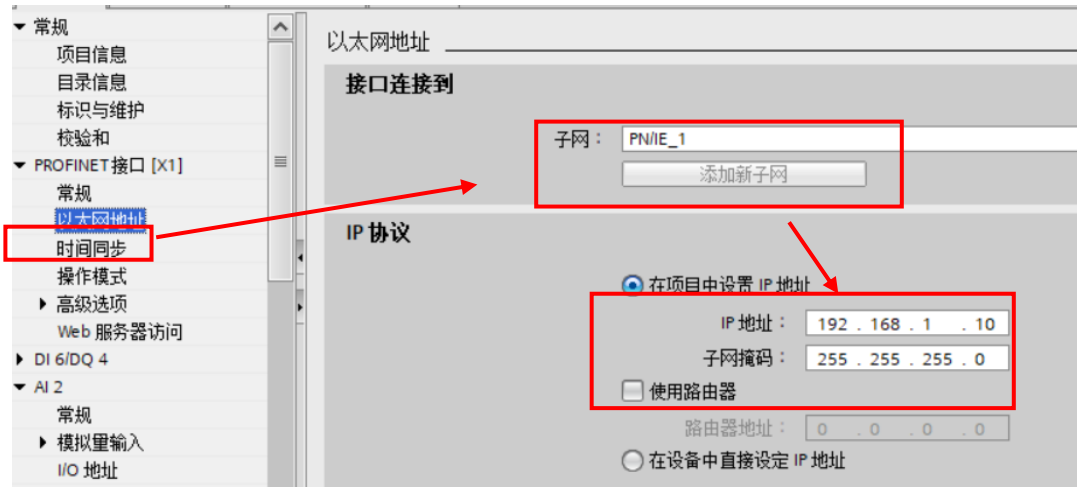
S7-1200 将通信数据区 DB1 中的 100 个字节发送到 H36-01 的 VB 数据区，S7-1200 读取 H36-01 中的 VB 数据区存储到 S7-1200 的数据区 DB1 数据区中。

S7-1200 M 区中的 MB0-MB9 连续 10 个字节发送到 H36-01 的 M 区中 MB0-MB9，S7-1200 读取 H36-01 区中的 MB10-MB19 连续 10 个字节到 S7-1200 的 M 区中 MB10-MB19。

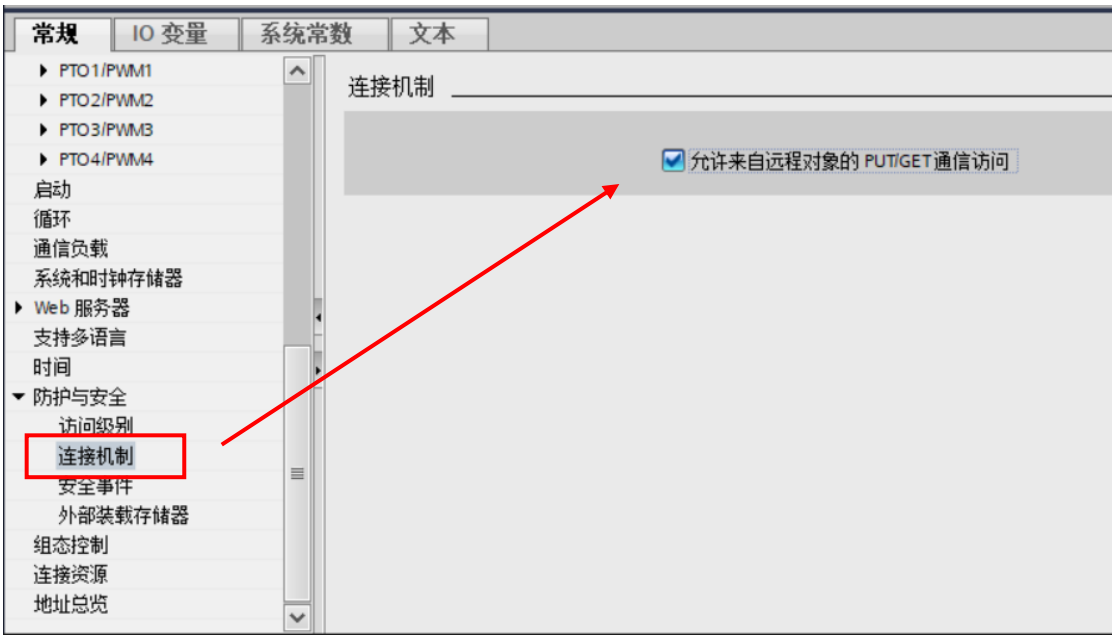
**S7-1200 主站的配置编程**

步骤 1：使用 TIA portal V15 软件新建一个项目并完成硬件配置，网络组态

1）新建子网 PN/IE_1，修改 IP 地址与 PC 和 H36-01 所在同一个网段

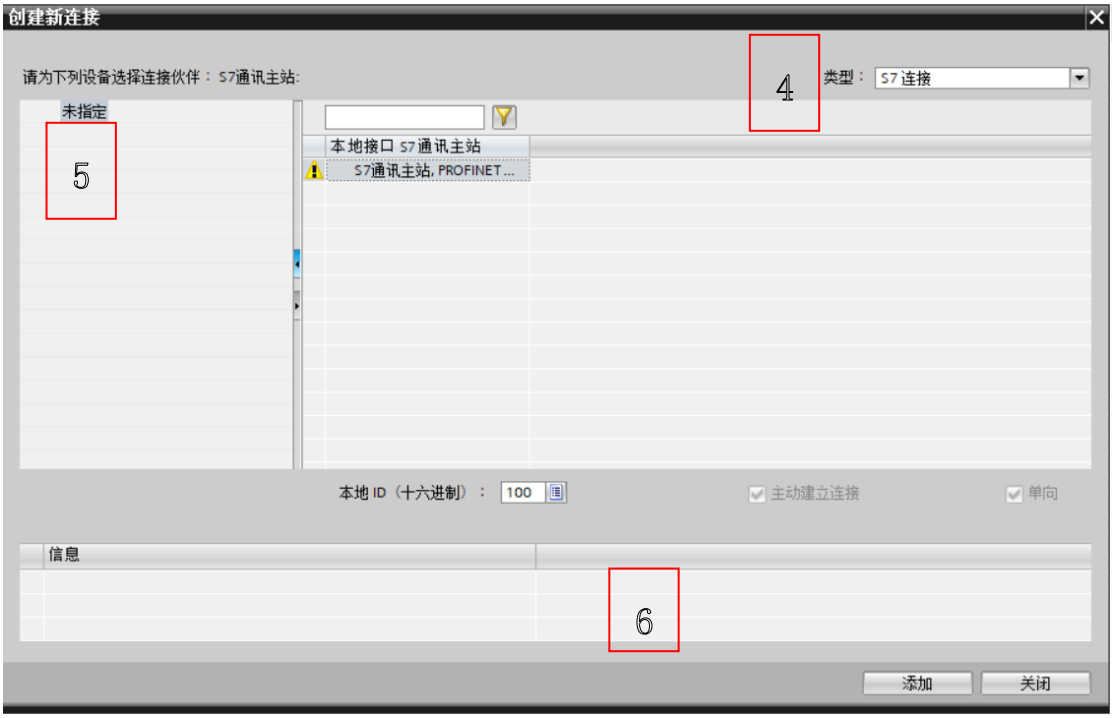



2）将“防护与安全”>“连接机制”>“允许来自远程对象的 PUT/GET 通信访问”勾选

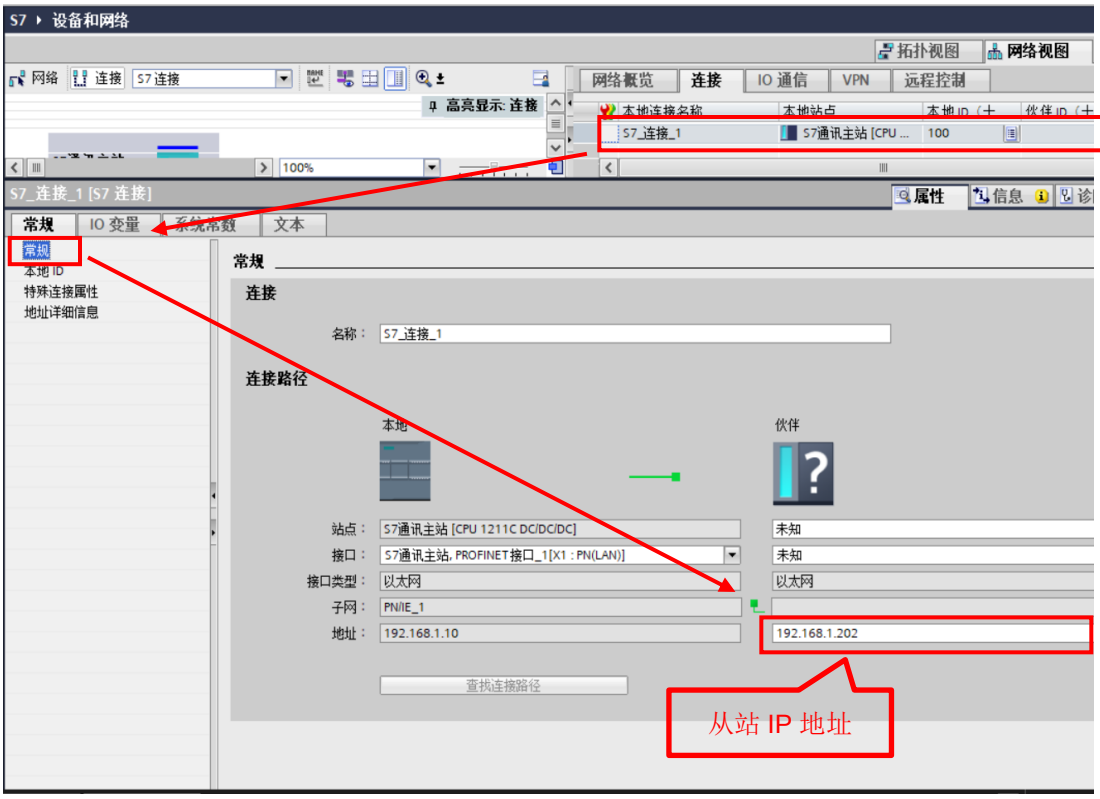



3）在“项目树”>“设备组态”>“网络视图”下，按如下图片中 1~6 的步骤建立 S7 连接

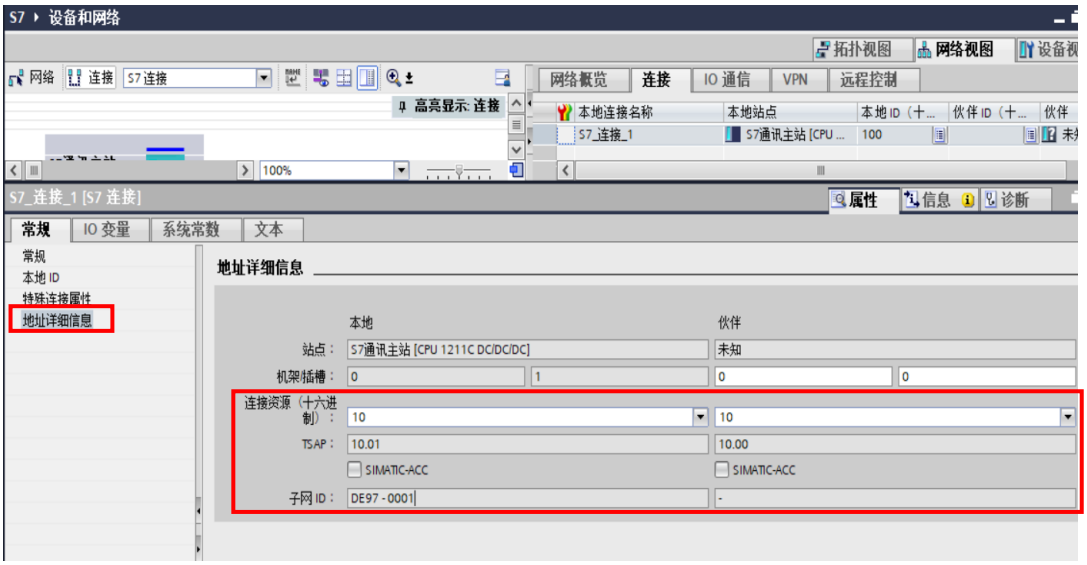


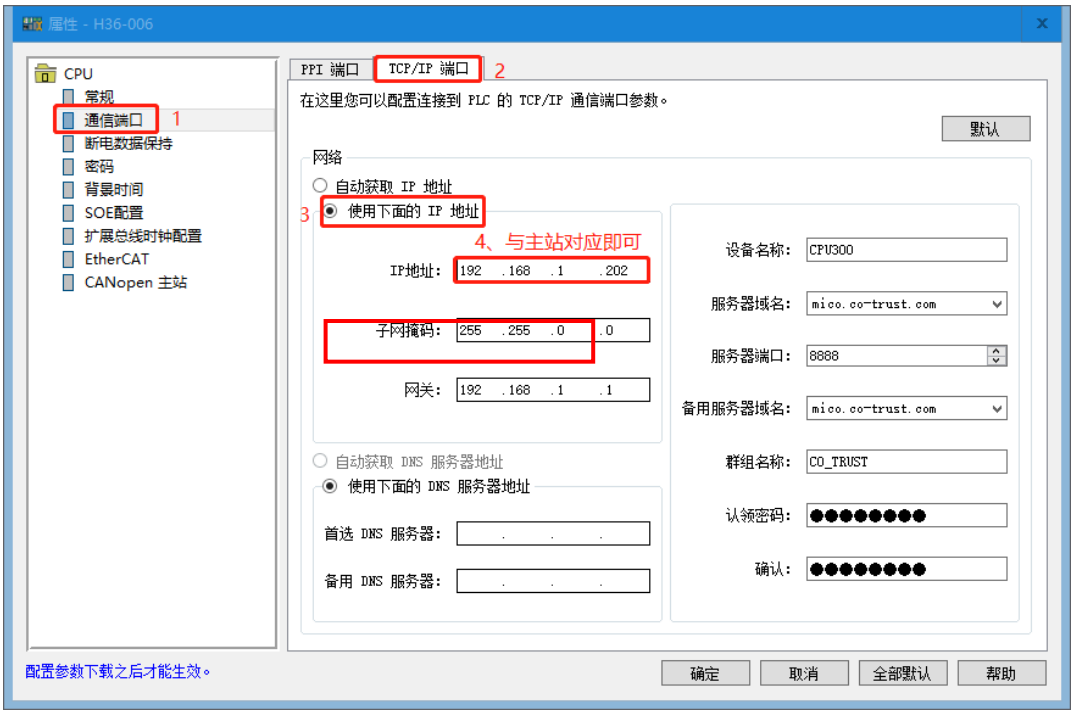

4）按如下图片中的步骤填写连接参数




5）在“地址详细信息”中设置通信伙伴的 TSAP 地址

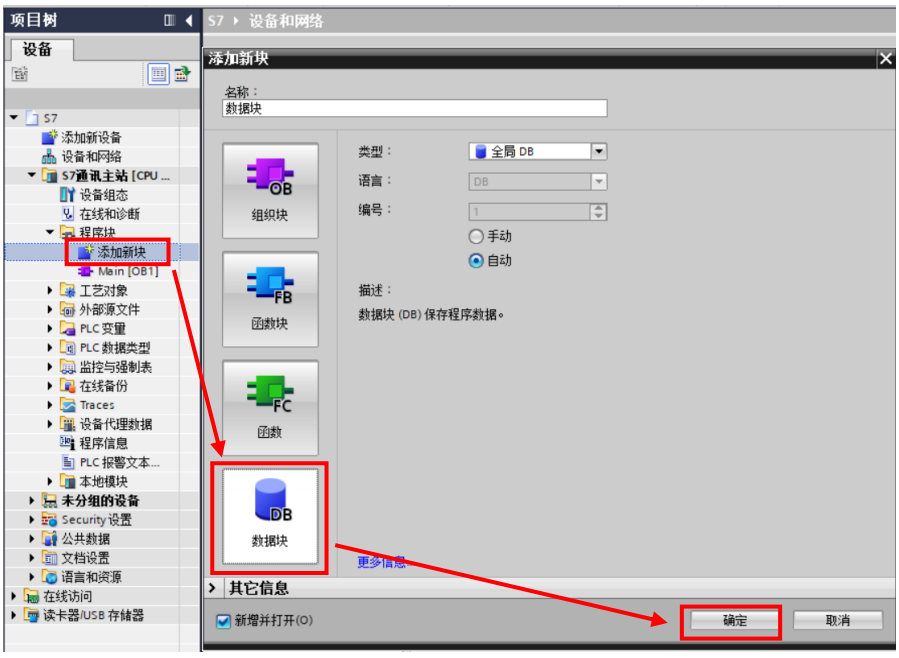



6）在 CTH300-H 系列 PLC 的硬件组态中配置好从站 IP 地址即可，操作示意如下：
在机架上添加一个 H36-01CPU，双击该模块以后进行一下操作即可。

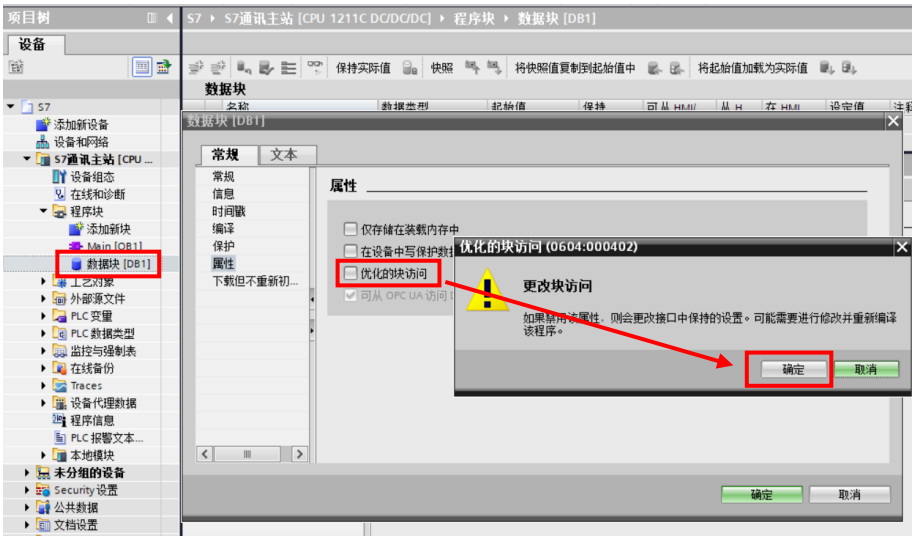



步骤 2：软件编程

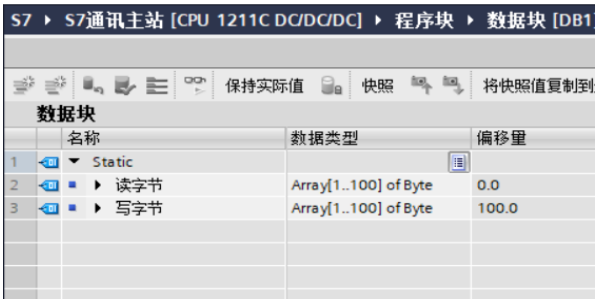
1）创建数据块 DB1,定义两组为 100 个字节的数组



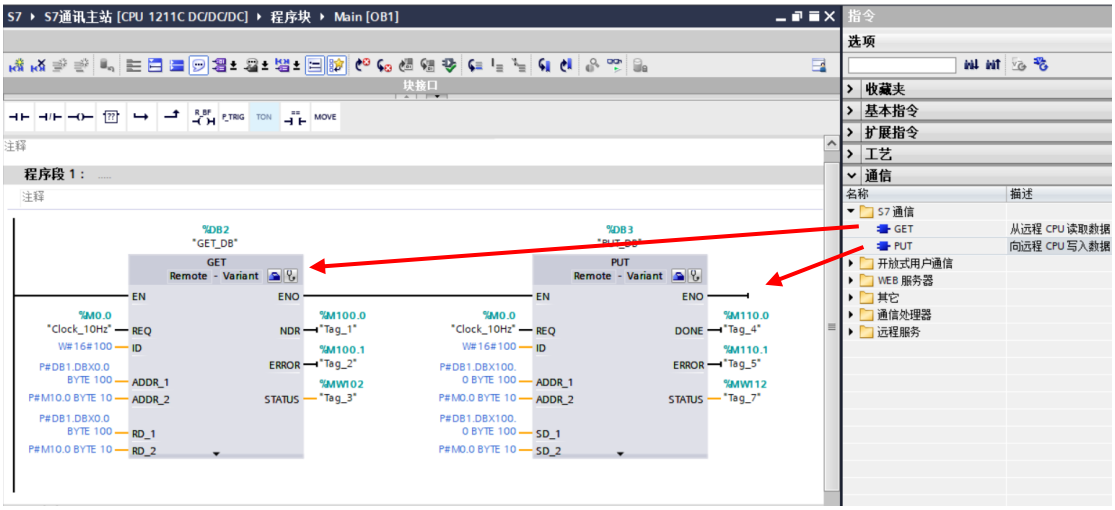

2）在“项目树”>“数据块”>“属性”下，将优化块访问勾选去掉。




3）创建数据块 DB1 的写数据和读数据的缓冲区，定义两组为 100 个字节的数组




4）在 OB1 中，从“指令”-->“S7 通信”下，调用 Get、Put 通信指令，程序编写如下：




CALL “GET” , %DB2 //调用 PUT，使用背景 DB 块：DB2

REQ : =%M0.0 //系统时钟 0.1 秒脉冲

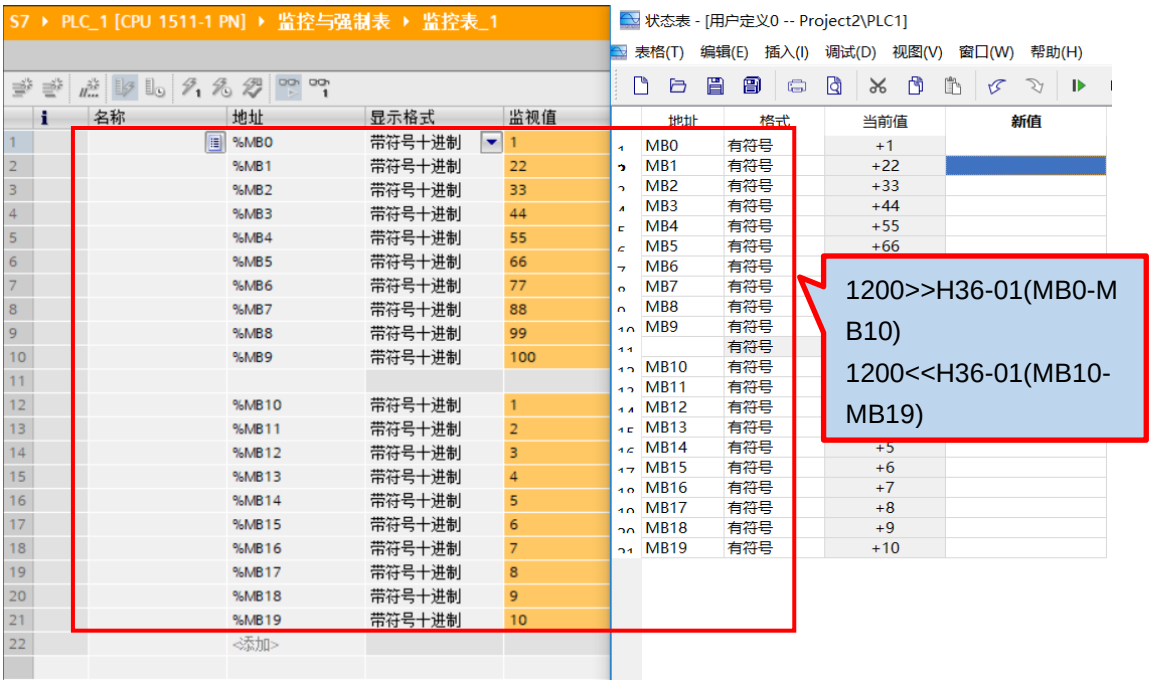
ID : =W#16#0100 //连接号，要与连接配置中一致，创建连接时的连接号

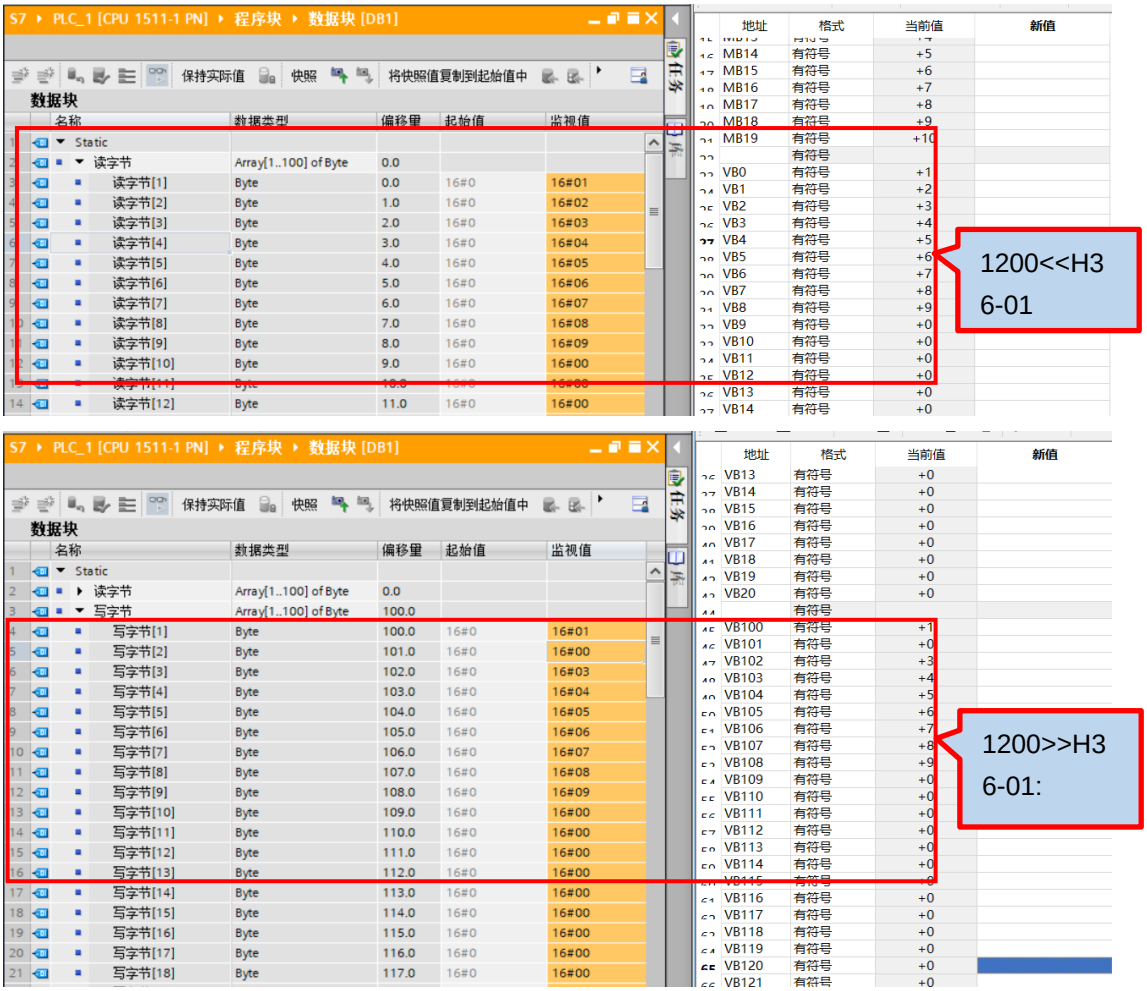
DONE : =%M100.0 //为 1 时，发送完成

```
ERROR      ：=%M100.1          //为 1 时，有故障发生
STATUS     ：=%MW102           //状态代码
ADDR_1     ：=P#DB1.DBX0.0 BYTE 100    //从通信伙伴数据区读取数据的地址
ADDR_2     ：=P#M10.0 BYTE 10          //从通信伙伴数据区读取数据的地址
SD_1       ：=P#DB1.DBX0.0 BYTE 100    //本地接收数据地址
SD_2       ：= P#M10.0 BYTE 10         //本地发送数据区
CALL “PUT” , %DB3              //调用 GET，使用背景 DB 块：DB3
REQ        ：=%M0.0            //系统时钟 0.1 秒脉冲
ID         ：=W#16#0100        //连接号，要与连接配置中一致，创建连接时的连接号
NDR        ：=%M110.0          //为 1 时，接收到新数据
ERROR      ：=%M110.1          //为 1 时，有故障发生
STATUS     ：=%MW112           //状态代码
ADDR_1     ：=P#DB1.DBX100.0 BYTE 100   //发送到通信伙伴数据区的地址
ADDR_1     ：=P#M0.0 BYTE 10            //发送到通信伙伴数据区的地址
RD_1       ：=P#DB1.DBX100.0 BYTE 100   //本地发送数据区
RD_1       ：=P#M0.0 BYTE 10            //本地发送数据区
```

步骤 3：监控结果

通过在 S7-1200 侧编程进行 S7 通信，实现两个 CPU 之间的数据交换，监控结果：






注意：H36-01 中 V 区对应于 DB1，即在 PUT 指令中使用的通信伙伴数据区 ADDR_1=P#DB1.DBX0.0 BYTE 100 在 H36-01 中对于为 VB0~VB99。即在 GET 指令中使用的通信伙伴数据区 ADDR_1=P#DB1.DBX100.0 BYTE 100 在 H36-01 中对于为 VB100~VB199。

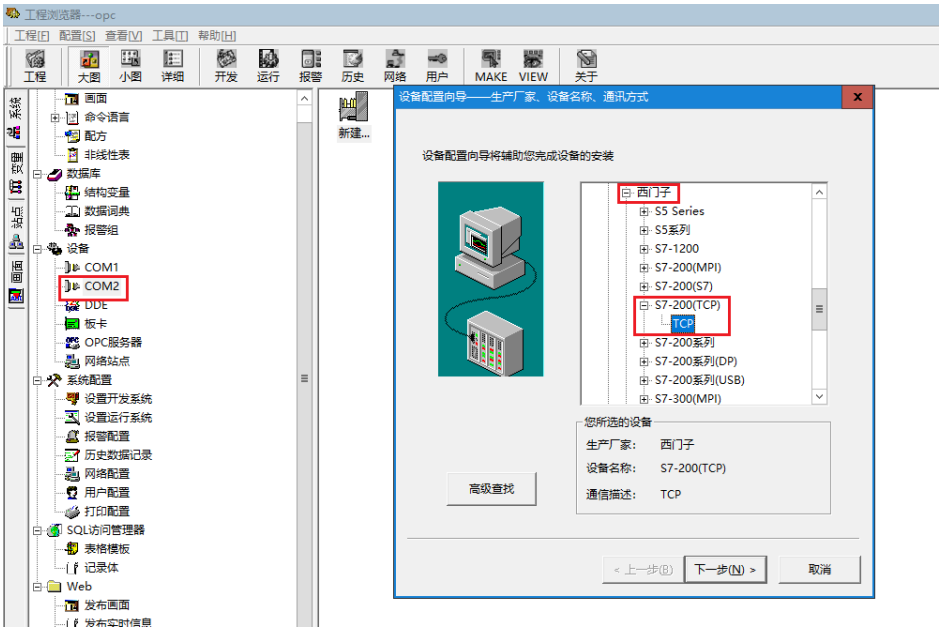
## *其它产品与 CTH300-H 系列 PLC 的网口通讯*

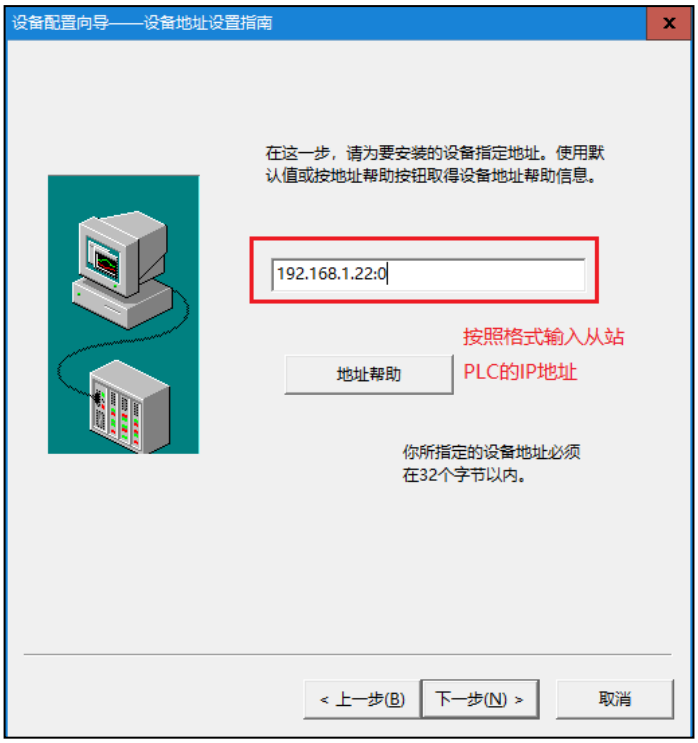
### 示例 1：组态王与 CTH300-H 系列 PLC 的 S7 协议通信

本例以组态王为主站，CTH300-H 系列 PLC 做从站，进行 S7 协议的通信。

**主站配置**

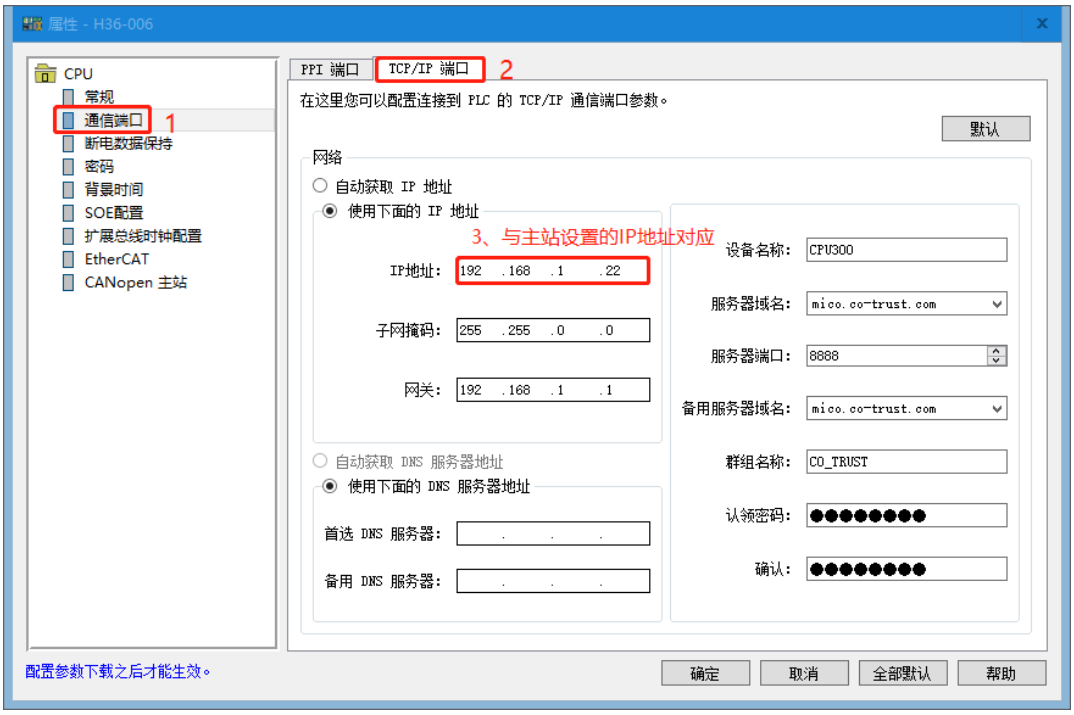
组态王软件的串口设备配置向导中选择 S7 通信协议：




**从站配置**

在 CTH300-H 系列 PLC 的硬件组态中配置好从站 IP 地址即可，操作示意如下：在机架上添加一个 H36-01CPU，双击该模块以后进行以下操作即可。



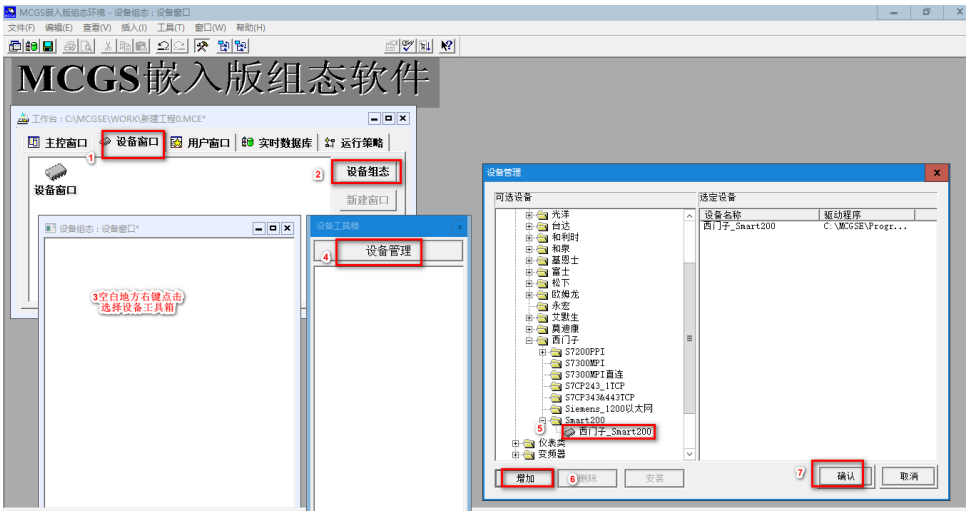

**示例 2：昆仑通态触摸屏与 CTH300-H 系列 PLC 的 S7 协议通信**

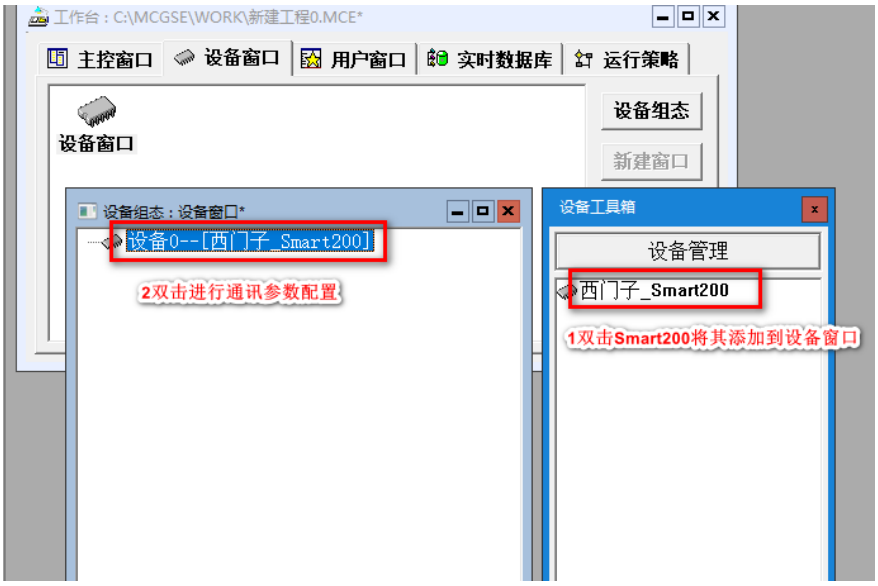
本例以昆仑通态触摸屏为主站，CTH300-H 系列 PLC 做从站，进行 S7 协议的通信。

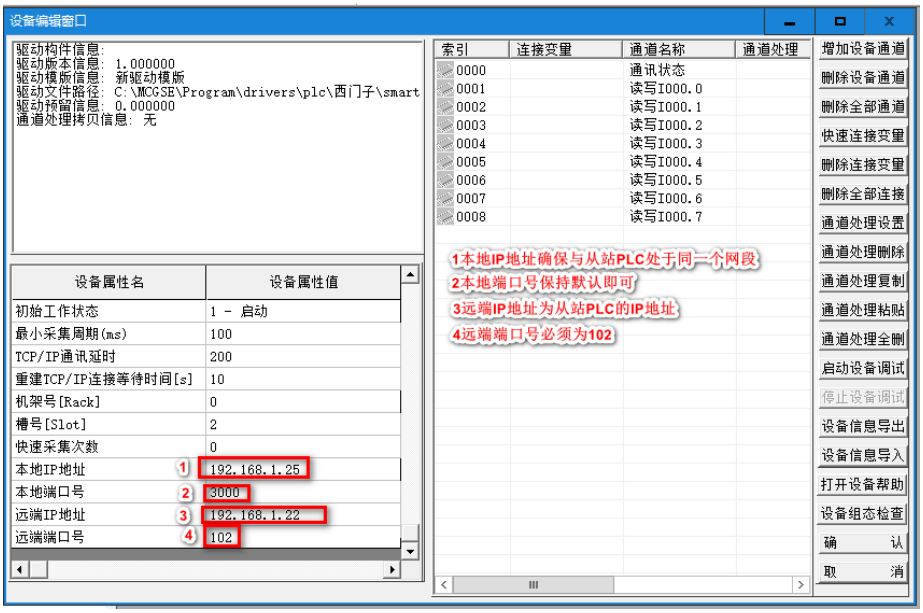
**<备注>**：昆仑通态上位机也可以作为主站与 CTH300-H 系列 PLC 进行 S7 协议的通信。

**主站配置**

参考如下步骤对昆仑通态触摸屏进行 S7 协议的主站配置：

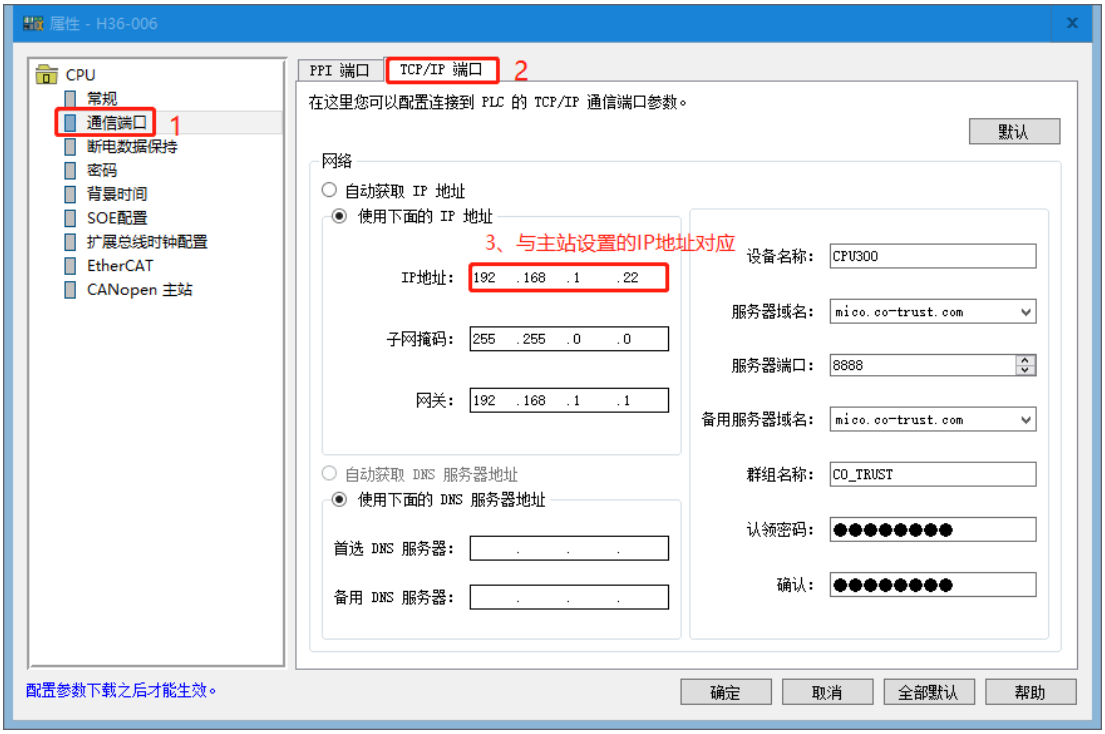





### 从站配置

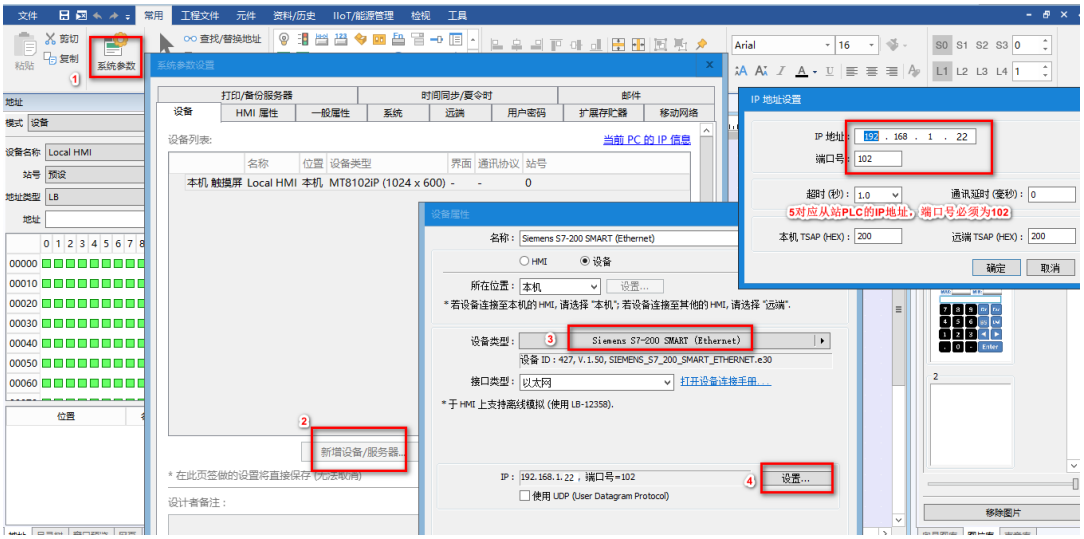
在 CTH300-H 系列 PLC 的硬件组态中配置好从站 IP 地址即可，操作示意如下：在机架上添加一个 H36-01CPU，双击该模块以后进行以下操作即可。




### 示例 3：威纶通触摸屏与 CTH300-H 系列 PLC 的 S7 协议通信

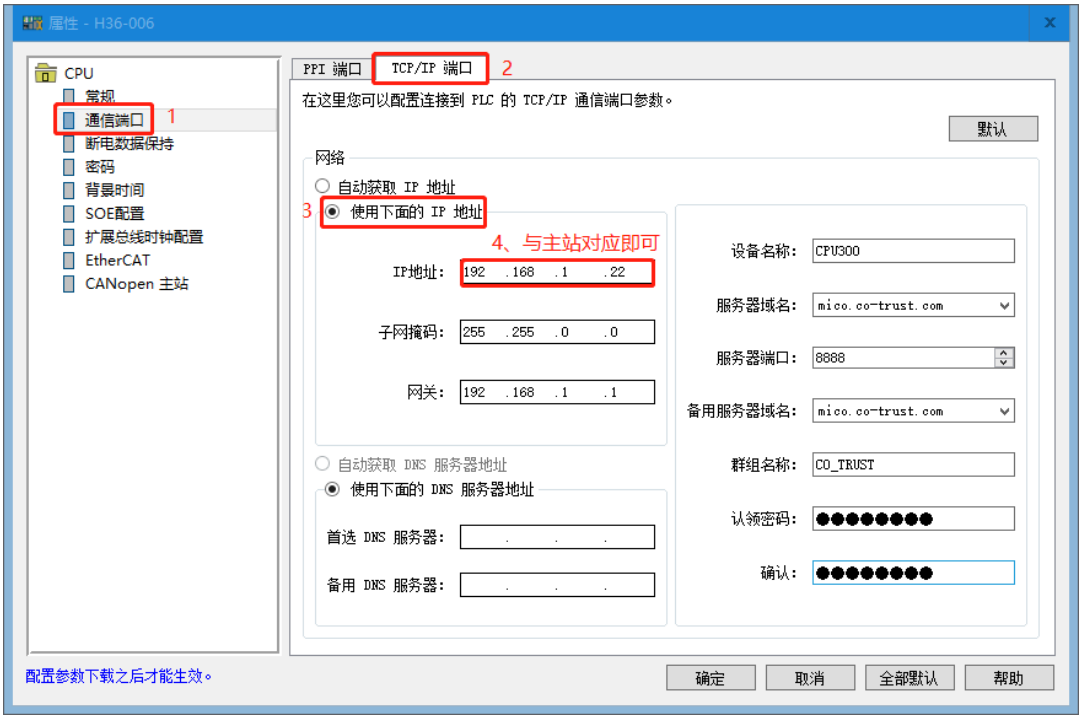
本例以威纶通触摸屏为主站，CTH300-H 系列 PLC 做从站，进行 S7 协议的通信。

**主站配置**




**从站配置**

在 CTH300-H 系列 PLC 的硬件组态中配置好从站 IP 地址即可，操作示意如下：
在机架上添加一个 H36-01CPU，双击该模块以后进行一下操作即可。




## 14.2 为主控模块 RS485 口选择通信协议

300 CPU 的 RS485 通信口可切换 PPI（默认）与自由口模式，若要切换 PPI 与自由口协议，则需要使用特殊存储器字节 SMB30（PORT0）和 SMB130（PORT1）。

**SMB30 控制 PORT0 的自由口通讯，SMB130 控制 PORT1 的自由口通讯。**您可以从 SMB30 和 SMB130 读取或向 SMB30 和 SMB130 写入。这些字节配置各自的通讯端口，进行自由口操作，并提供自由口或系统协议支持选择。

表 14-22　位格式

| 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<br>00--无校验<br>01--偶校验<br>10--无校验<br>11--奇校验 | | **d**：每个字符的数据位<br>0--每个字符 8 个数据位<br>1--每个字符 7 个数据位 | **bbb**：波特率<br>000--38400bps<br>001--19200bps<br>010--9600bps<br>011--4800bps<br>100--2400bps<br>101--1200bps<br>110--115200bps<br>111--57600bps | | | **mm**：协议选择<br>00--PPI/ 从站模式（默认）<br>01--自由口模式<br>10--PPI/主站模式<br>11--保留 | |

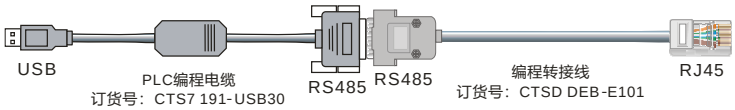
例如：此处将 PORT0 选择为自由口模式、波特率为 19200bps、无校验、每个字符 8 个数据位，那么应该将 CPU 的 SMB30 设置为：00000101。

## 14.3 高速计数模块应用示例

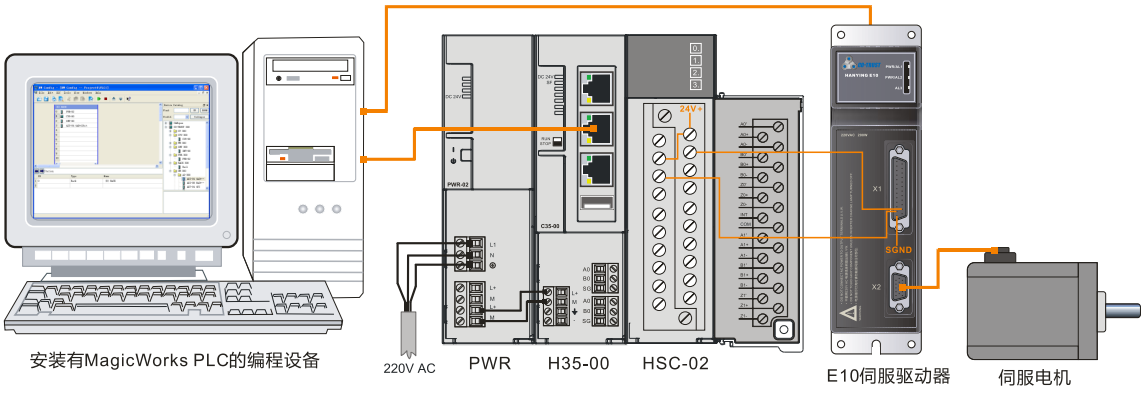
本节将通过一个具体实例来展示高速计数模块如何对伺服电机编码器反馈的脉冲信号进行计数和测速。

### 14.3.1 示例组件

表 14-23 示例组件

| 组件 | 功能 |
|---|---|
| 编程设备 PG\PC | 安装有 MagicWorks PLC 软件（V2.00 或更高版本），对 CTH300-H 系列可编程控制器进行组态、编程和调试。 |
| 装配导轨 | CTH300 系统机架，用于固定系统中的各模块。 |
| 电源模块 PWR-02 | 给 CTH300-H 系列主控模块及其 24 VDC 负载电路供电。 |
| 外部电源 | 提供 E10 伺服驱动器工作电源 |
| H35-00 主控模块 | CPU 执行用户程序，向 CTH300 系统背板总线提供 5V 电压。 |
| HSC-02 高速计数模块 | 通过总线与 H35-00 进行连接，对电机速度、位置等进行计数。 |
| 驱动设备 | E10 伺服驱动器。 |
| 伺服电机 | 与 E10 伺服驱动器相连。 |
| 编程电缆 | 将H35-00连接到编程设备：PLC编程电缆+编程转接线<br><br> |
| 标准网线 | 连接 E10 伺服驱动器与编程设备。 |
| 编码器电缆 | 连接 E10 伺服驱动器与电机。 |

### 14.3.2 网络连接



### 14.3.3 操作步骤

**步骤 1：接线**

打开 H35-00 和电源模块 PWR-02 的前面板，为它们接线。具体步骤如下：

1）使用编程电缆（PLC 编程电缆+编程转接线）连接 PC 与 H35-00

2）H35-00 与高速计数模块通过总线进行连接

3）使用编程电缆连接 PG/PC 与 E10 驱动器

4）使用编码器电缆连接 E10 驱动器与电机

5）为高速计数模块和 E10 驱动器接线

**步骤 2：运行驱动系统**

通过设置 E10 伺服驱动器参数使电机开始正常运转，具体操作参考《E10 系列交流伺服驱动器使用说明书》。

**步骤 3：设置 PLC 通信**

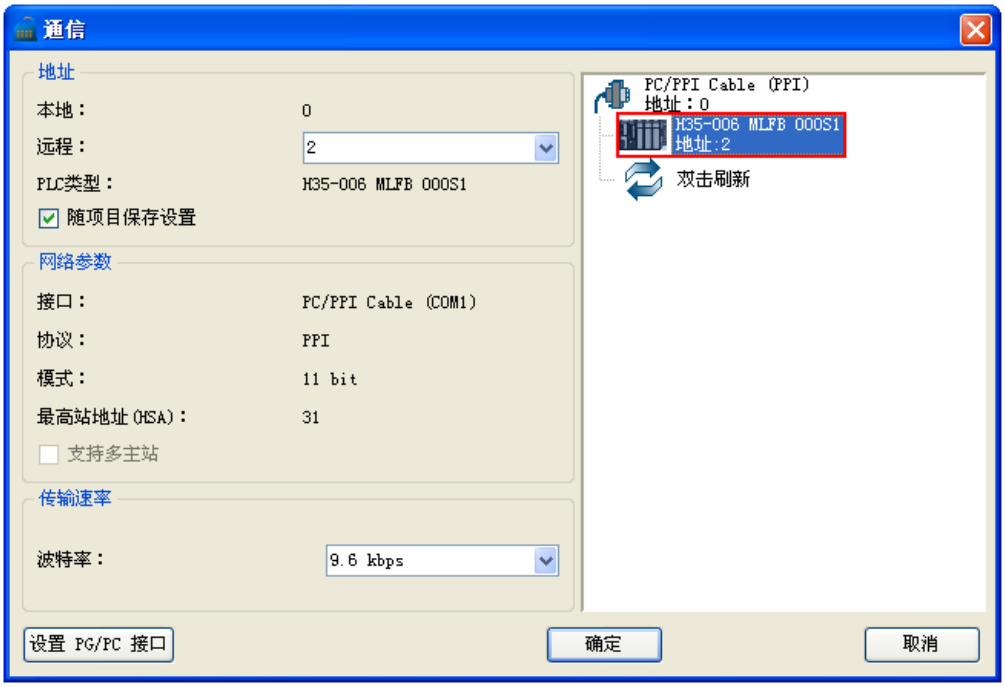
在 MagicWorks PLC 中新建一个工程，在该工程中添加 H35-00 站点，随后即可参考如下步骤对 H35-00 进行通讯设置。

1）设置 PG/PC 接口

选择菜单项“工具”→“设置 PG/PC 接口”打开如下窗口，在“设置 PG/PC 接口”窗口中选择使用接口“PC/PPI Cable（PPI）”，然后点击“属性”按钮打开属性对话框，即可设置通讯波特率、串口，最后点击“确定”按钮完成 PG/PC 接口设置。

2）与 H35-00 建立通讯

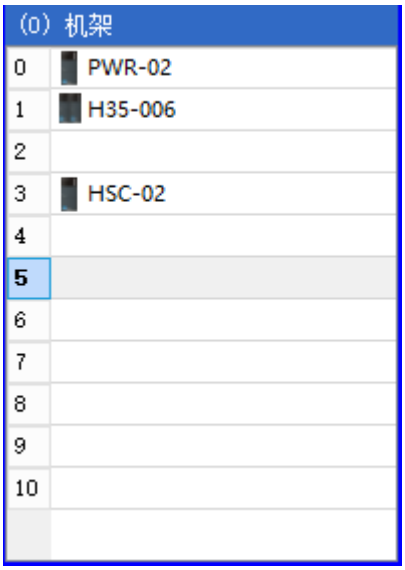
在工作窗口双击通信图标弹出如下通信窗口，双击通信对话框中的图标进行搜索，搜索成功的 H35-00 即会显示在通信对话框中。




**步骤 4：在 MagicWorks PLC 中进行硬件组态**

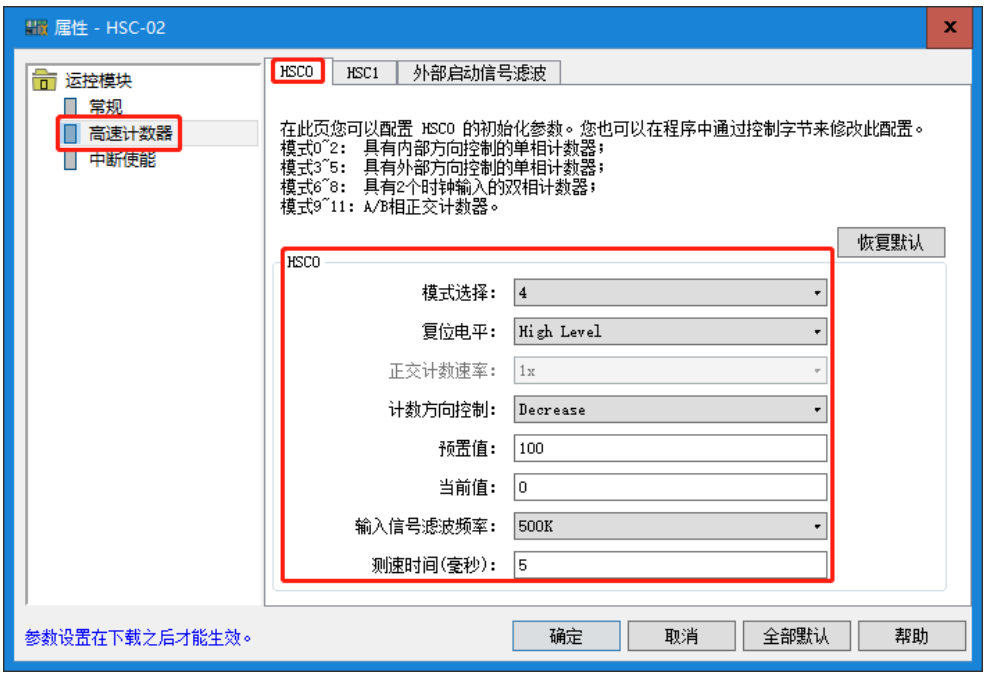
在 MagicWorks PLC 项目视图中单击选中 H35-00 站点，然后在其右侧工作窗口双击硬件组态图标，进入硬件组态界面。

1）在硬件组态界面，通过设备目录将电源、CPU、高速计数模块添加到机架上，组态完成的工程如下图所示：

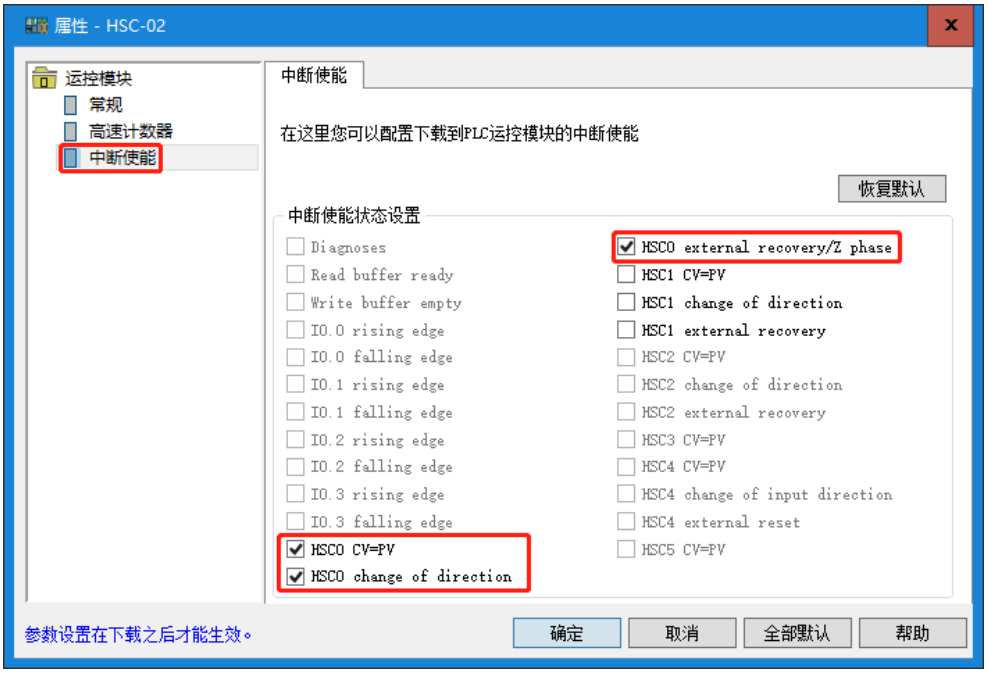



2）配置高速计数模块（HSC-02）属性

在机架中双击模块 HSC-02 打开其配置窗口，在选项卡“高速计数器”中为计数通道 HSC0 配置相关参数（本例只使用了一个通道，故仅配置 HSC0）：



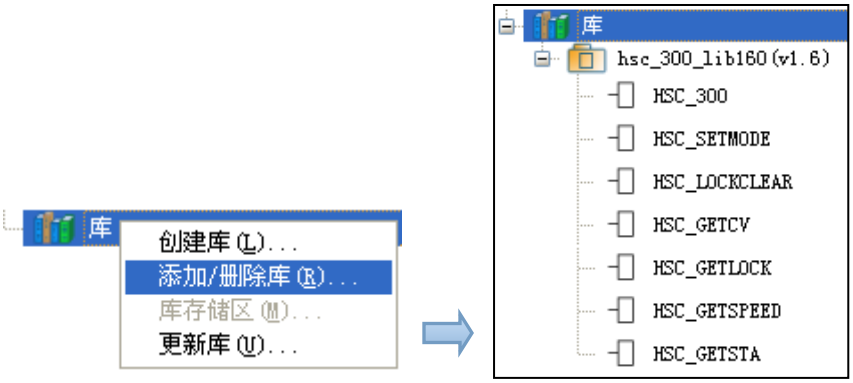

然后勾选需要中断使能的通道（本例只使用了一个通道，故仅使能 HSC0）：




**步骤 5：在 MagicWorks PLC 中编程**

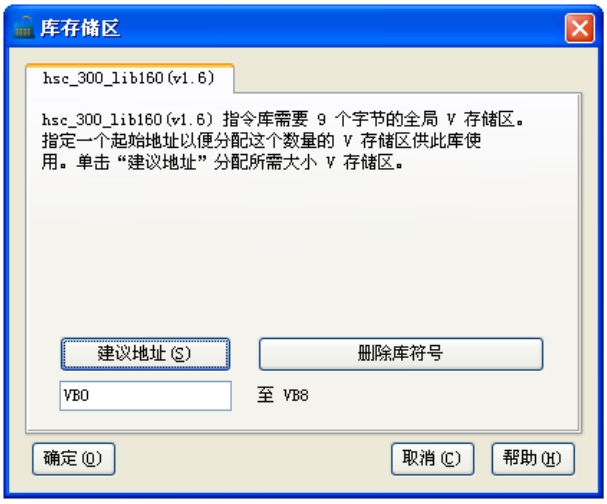
1）添加高速计数模块配置库 hsc_300_lib

在 MagicWorks PLC 主界面的项目窗口中打开程序块对话框，然后在程序对话框的指令树中右键选择“库”->“添加/删除库”，然后在“添加/删除库”对话框中点击“添加”按钮选择添加库文件hsc_300_lib，添加成功的库文件则显示在指令树中：



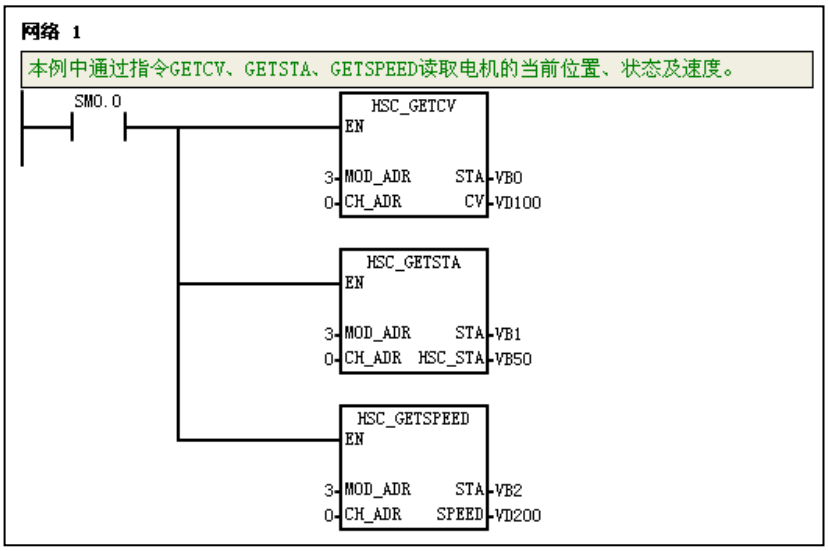

2）为 hsc_300_lib 库分配库存储区

在 MagicWorks PLC 主界面选择菜单项“文件”→“库存储区”，然后在弹出的对话框中填写存储区地址：




3）使用 hsc_300_lib 库指令进行编程

由于已经在硬件组态中对 HSC-02 模块进行配置，则可以直接使用指令（HSC_GETCV、HSC_GETSPEED、HSC_STA）读取该模块的速度、位置及状态等信息。




**<备注>** 若未配置 HSC-02 模块，则可以通过指令 HSC_300 配置。

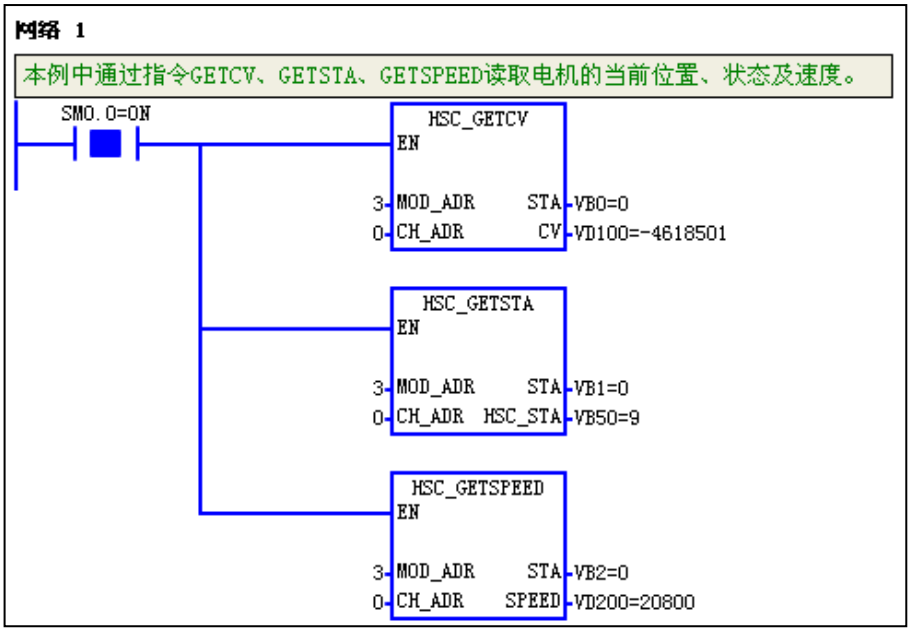
**步骤 6：调试与监控程序**

1）编译、下载程序

选择菜单项“文件”→“保存”以保存当前组态，然后选择菜单项“PLC”→“编译”对当前工程进行编译；若编译成功，则可以进行下载操作，在主界面选择菜单项“PLC”→“下载”将程序块和硬件组态从编程设备下载到 H35-00 中。

2）调试程序

程序下载成功后，将 PLC 置于运行模式，然后点击按钮开始进行程序状态监控：




通过监控以上指令可以获得电机的当前速度（频率）及当前位置。

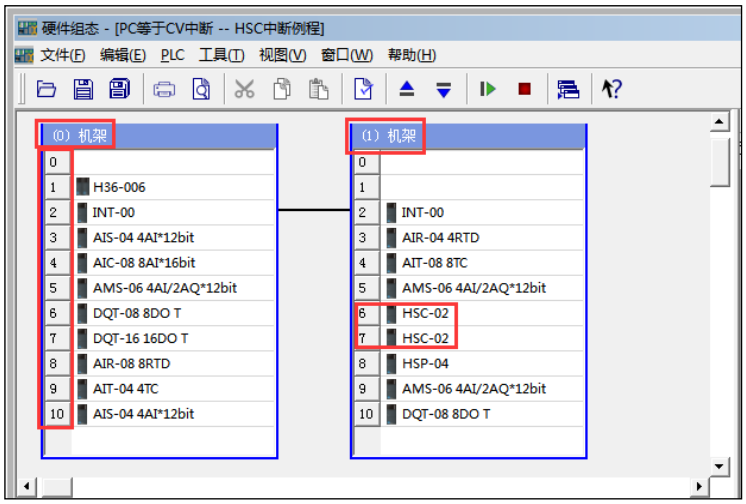
**<备注>** 系统出现故障时，请参考章节 14.5 故障诊断获取诊断方法。

## 14.4 HSC 中断示例

本节以 HSC 中断事件 CV=PV 为例，介绍如何连接中断，使能中断以及判定具体中断模块。

### 14.4.1 硬件组态

- 在 MagicWorks PLC 中新建一个工程，在该工程中添加 H36-00 站点，然后参考章节 2.3 设置通讯将 H36-00 与 PC 进行通信连接。
- 在 MagicWorks PLC 项目视图中单击选中 H36-00 站点，然后在其右侧工作窗口双击硬件组态图标，进入硬件组态界面。
- 在硬件组态界面，通过设备目录将电源、CPU 和所需的扩展模块添加到机架上。



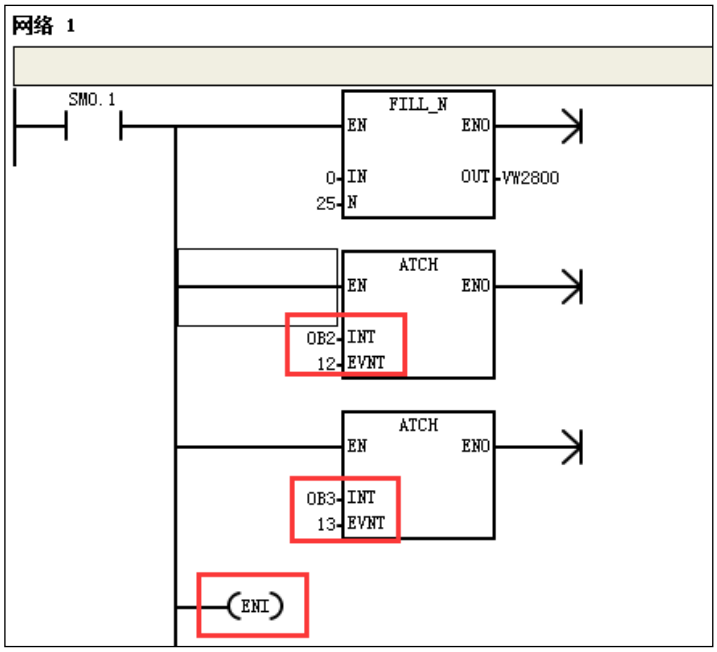
**说明：**如上图所示，硬件组态中有两个机架，分别为 0 号和 1 号，两个 HSC 模块分别插入 1 号机架的插槽 6/7。

## 14.4.2 中断例程

如上文“硬件组态”所述，机架 1 的 6/7 号插槽分别搭载一个 HSC 模块。下面以 HSC 的中断事件 CV=PV 为例说明如何使用 HSC 中断。

**连接中断**

如下图程序所示，将中断事件代码 12/13（HSC0/HSC1 的 CV=PV 中断）分别与中断子程序 OB2 和 OB3 相关连，当触发对应中断号的中断条件时就会执行相应的中断程序。




**使能中断**

通过 ENI 指令使能中断程序的执行。

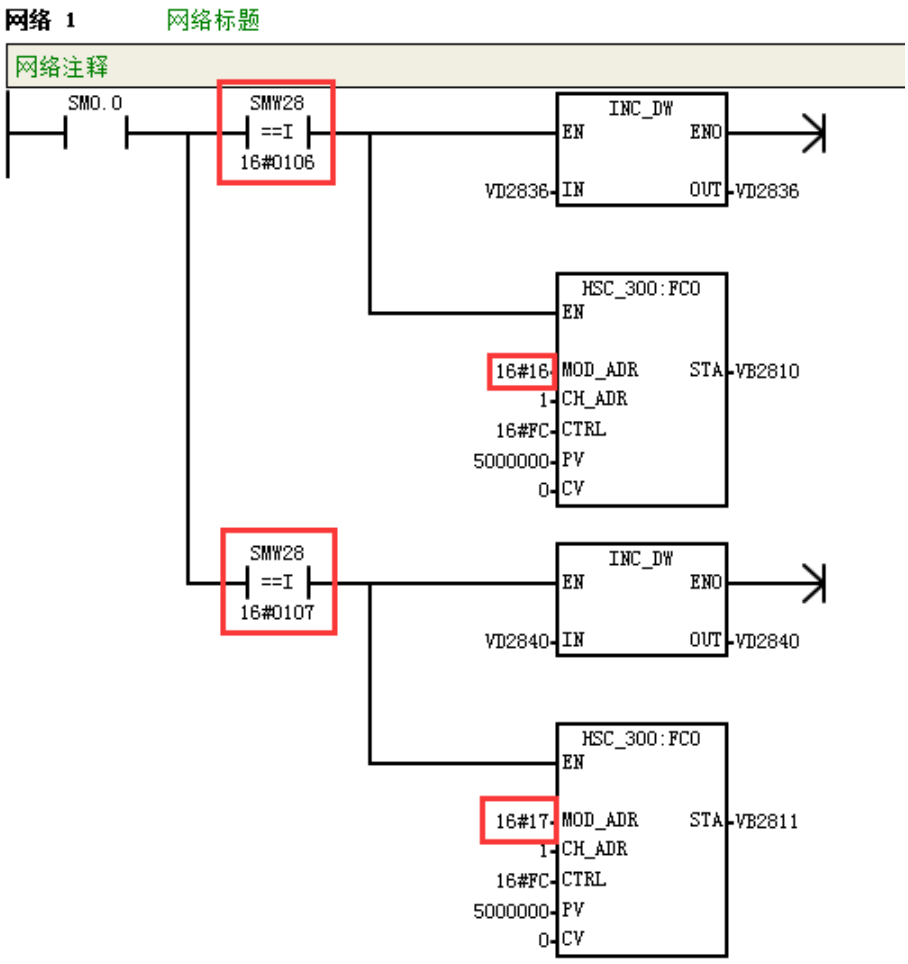
**判定具体中断模块**

在中断子程序 OB2 和 OB3 中，根据特殊存储器 SMW28（双字）确定产生诊断中断(发生故障)或者硬件中断的模块逻辑地址。

如下图所示，当 SMW28 的值等于 16#0106 时，表示 1 号机架上 6 号插槽的 HSC 模块发生中断，当 SMW 的值等于 16#0107 时，表示 1 号机架上 7 号插槽的 HSC 模块发生中断。

**提示**

使用时务必注意，特殊存储器 SMW28为字，MOD_ADR 模块地址参数为字节格式，下图示例中的 SMW 16#0106与 MOD_ADR 16#16是指同一个模块地址。




## 14.5 故障诊断

当系统出现故障时，请先检查以下条件是否满足：

1）CTH300-H 系列主控模块及扩展模块是否正常供电（**注：**主控模块和扩展模块（除数字量模块外）建议采用 PWR-02 独立电源供电，电源不稳会导致停机）。

2）CTH300-H 系列主控模块及扩展模块 I/O 端子的螺丝和接插件是否松动。

3）检查通信电缆的连接情况，确保无误。

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

除以上方法外，还可通过 MagicWorks PLC 读取诊断信息，或者通过 PLC 的 LED 指示灯状态检查 PLC 自身和外部有无异常。

### 14.5.1 通过编程软件进行诊断

诊断信息读取方式：在编程软件 MagicWorks PLC 项目管理器界面菜单项选择“PLC”→“获取诊断信息”，即可打开诊断窗口。

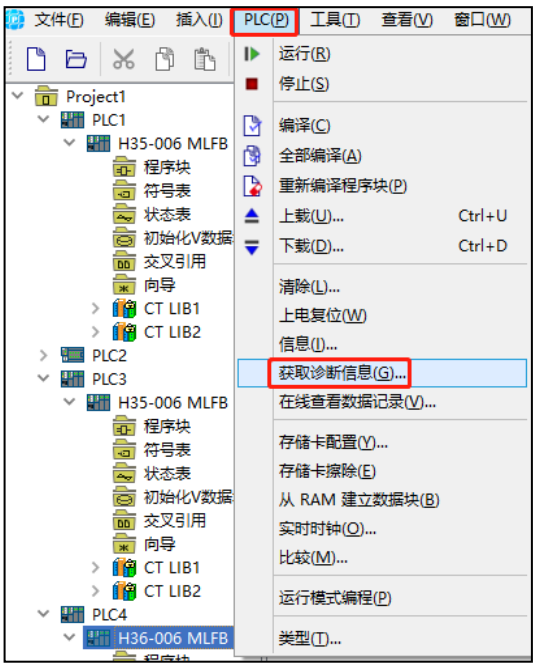



表 14-24　诊断信息窗口描述

| 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。 |

表 14-25　诊断功能规格

| 支持的事件类型 | 类型编码 | 具体事件编码及描述 | |
|---|---|---|---|
| | | 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 |
| 同步错误 | 0x6 | 0x00 | 无错误（保留） |
| | | 0x01 | 使用的 DB 块不存在或者使用 DB 越界（保留） |
| CPU 非致命错误 | 0x7 | 0x00 | 无错误 |
| | | 0x01 | 在执行 HDEF 方框之前启用 HSC（保留） |
| | | 0x02 | 输入中断分配冲突，分配至已分配给 HSC 的点（保留） |
| | | 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（保留） |
| | | 0x04 | 尝试在中断例行程序中执行 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 发生中断例行程序冲突（保留） |
| | | 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（保留） |
| | | 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 |
| | | 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 或中断例行程序中不允许存在指令；在中断例行程序结尾处增加 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） | |
| 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 | |
| 通信错误 | 0xA | 保留 | | |
| SOE 事件 | 0xB | BOOL 变量状态变化事件：<br>（1）DI、DQ 状态 ON→OFF，OFF→ON<br>（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。 |
| SOE 监视记录方式 | | 每个扫描周期监视记录一次 | | |
| H36-0X 的 EtherCAT 寄存器 SMB400~SMB465 | | 各状态位的描述参考表14-11 | | |

| | | | |
|---|---|---|---|
| 模块诊断事件<br>SMB500～<br>SMB531<br>参考表14-12 | 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 14-26　H36-0X 特殊存储区 SMB400~SMB465 诊断功能

| SMB | 错误描述 |
|---|---|
| SMB400 | 找到的 EtherCAT 从站的个数 |
| SMB401 | EtherCAT 错误：<br>0：没有错误<br>1：组态参数错误<br>2：没有找到从站<br>3：状态切换错误<br>4：写组态时发生错误<br>5：从站个数错误<br>6：从站不匹配 |
| SMB402 | 第1个从站的状态：<br>0：没有连接<br>1：初始化状态<br>2：预操作状态<br>4：安全操作<br>8：操作状态<br>16#80：产品 ID 不匹配<br>16#81：厂商 ID 不匹配<br>16#82：SDO 写入出错<br>其它：错误的状态 |
| SMB403~SMB465 | 从站2状态~从站64状态 |

表 14-27　特殊存储区 SMB500~SMB531 诊断功能

| SMB | 错误描述 |
|---|---|

| | |
|---|---|
| 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 |

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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行 |

| | |
|---|---|
| | 0x7F：预运行<br>0xFF：组态数据出错 |
| SMB551 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB591 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB631 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB671 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB711 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB751 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化 |

| | |
|---|---|
| | 0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB791 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB831 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>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<br>0~3位：槽号 SlotID；<br>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 | 第二十三个模块(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 | 第二十四个模块(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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB1551 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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 |

### 14.5.2 通过 PLC 状态指示灯进行诊断

表 14-29 H35-00 的 LED 状态指示灯功能描述

| 指示灯 | 颜色 | 描述 |
|---|---|---|
| 24V 电源指示灯 | 绿色 | 亮起：24VDC 供电正常，熄灭：无 24VDC 供电 |
| SF 指示灯 | 红色 | 亮起：系统故障，熄灭：无错 |
| BF 指示灯 | 红色 | 亮起：总线故障，熄灭：无错 |
| FRCE 指示灯 | 黄色 | 亮起：有项目被强制，熄灭：无项目被强制(也可通过 DLED 指令控制) |
| RMC 指示灯 | 绿色 | 亮起：CPU 与远程服务器通讯成功（拨码开关 3 拨为 ON 且 Ethernet 通信口参数已正确配置）<br>熄灭：与远程服务器通讯失败或禁止与远程服务器通讯（允许和禁止和服务器通讯通过拨码开关控制） |
| RUN 指示灯 | 绿色 | 亮起：系统运行，熄灭：系统停止 |
| STOP 指示灯 | 橙色 | 亮起：系统停止，熄灭：系统运行 |
| 以太网指示灯 | 绿色 | 亮起：连接，闪烁：未连接 |
| | 黄色 | 亮起：100Mbps，熄灭：10Mbps |

# 15 C 程序块

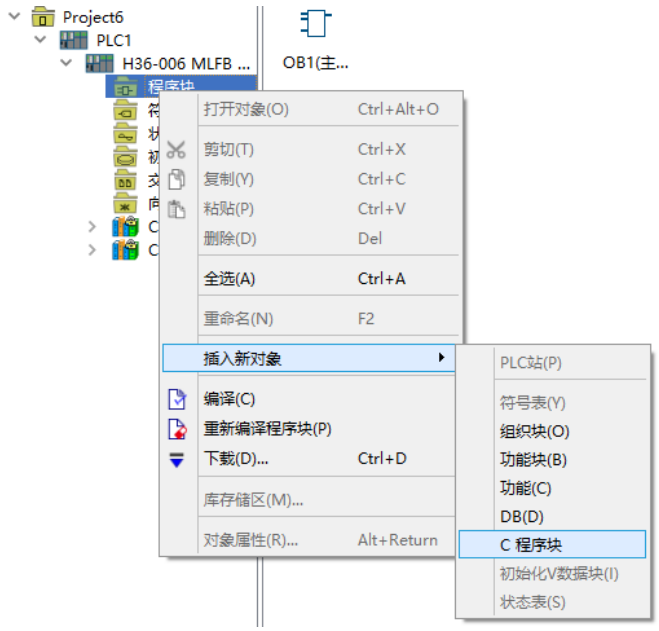
## 本章概要

C 函数编程语言是 MagicWorks PLC 软件功能的升级加强。在原有梯形图和 STL 两种编程语言基础上，允许 C 函数编程与梯形图等编程语言一起工作。利用 C 语言编写功能块，增强程序的可读性和维护性，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。目前有 H56-10/H52-10/H36-01/H32-01 支持 C 函数编程语言。
本章介绍 MagicWorks PLC 的 C 语言编程功能的使用，具体内容如下：

- 在 Magicworks PLC 中的 C 语言编程步骤
- C 语言基础
- 运算符与表达式
- for 循环语句
- 选择语句
- libplc300.a 库的介绍

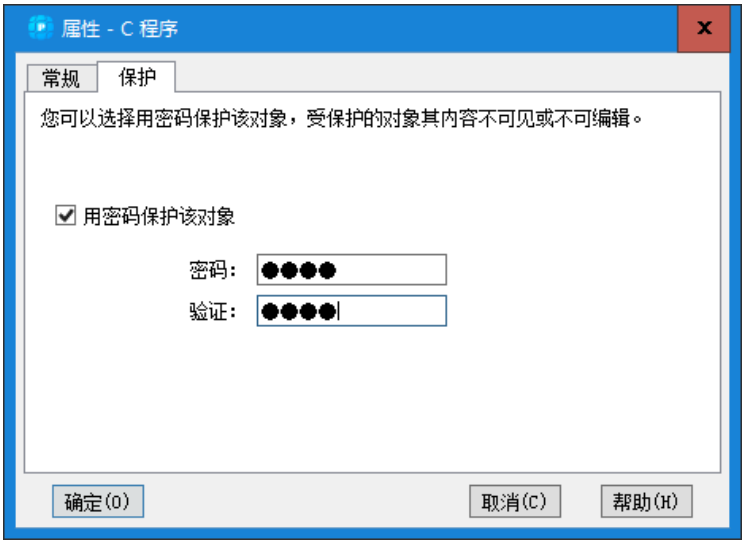
## 15.1 在 Magicworks PLC 中的 C 语言编程步骤

1、在 MagicWorks PLC 中新建项目，选择“程序块”后右键点击空白处，选择“插入新对象”→“C 程序块”。

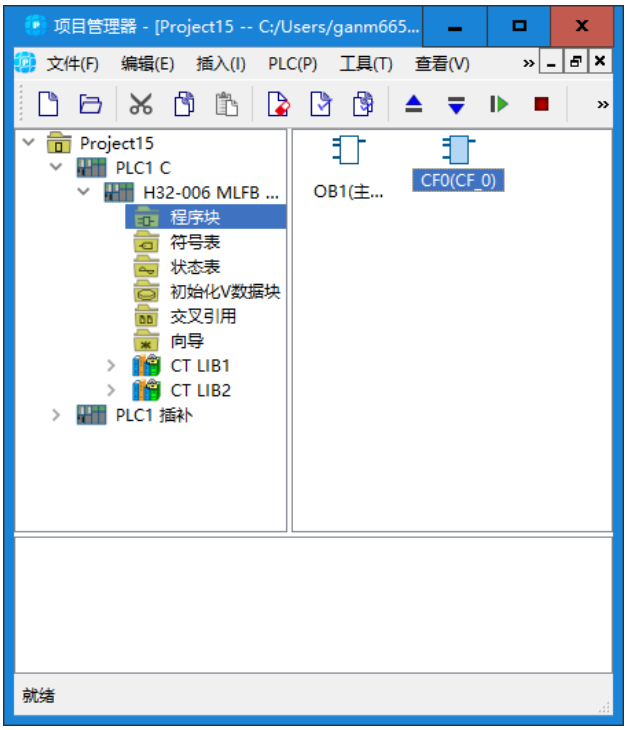


2、编辑 C 程序属性，此处定义的符号名将自动作为 C 函数名，建议不要用中文定义，复制代码时 CF 块的符号名与 C 语言函数名一定要相同，否则将不执行。如果工程师希望进行代码保护，也可以参考以下步骤增加密码保护。

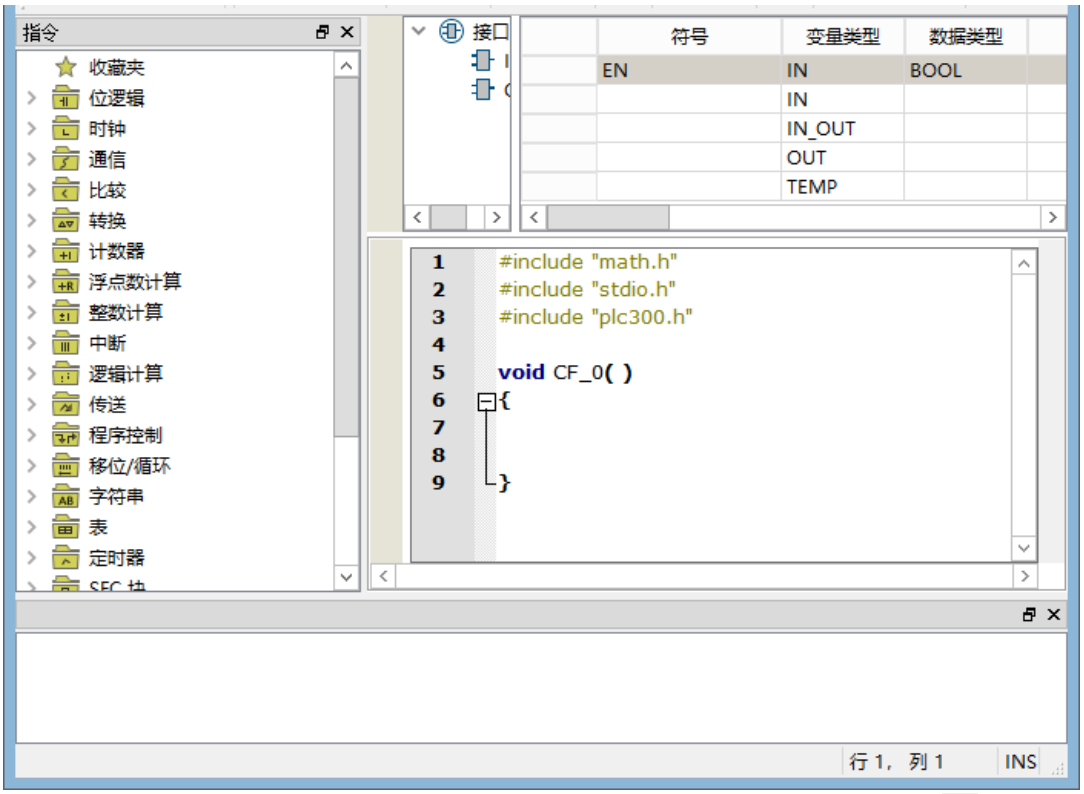


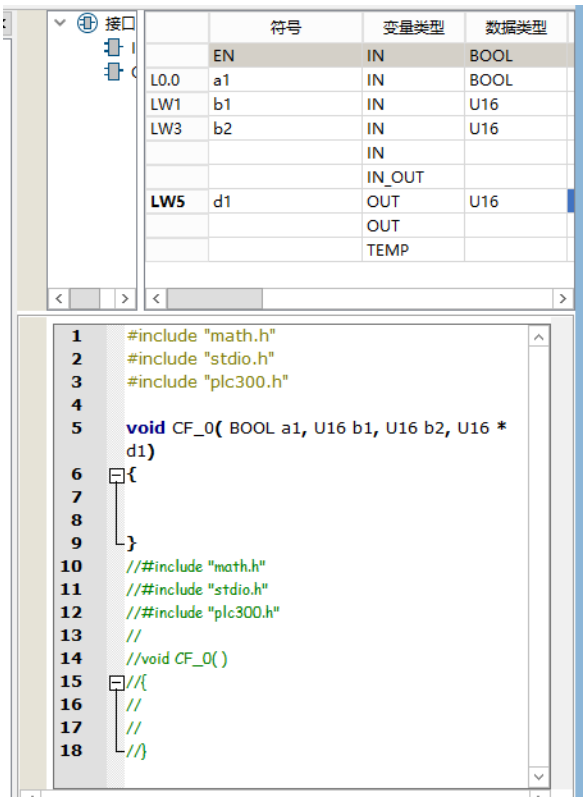

3、函数创建成功后，可在程序块中看到 CF0。




4、双击 CF0，进入 C 函数程序编程窗口。MagicWorks PLC 新建 CF0 后会默认加入以下三个头文件。

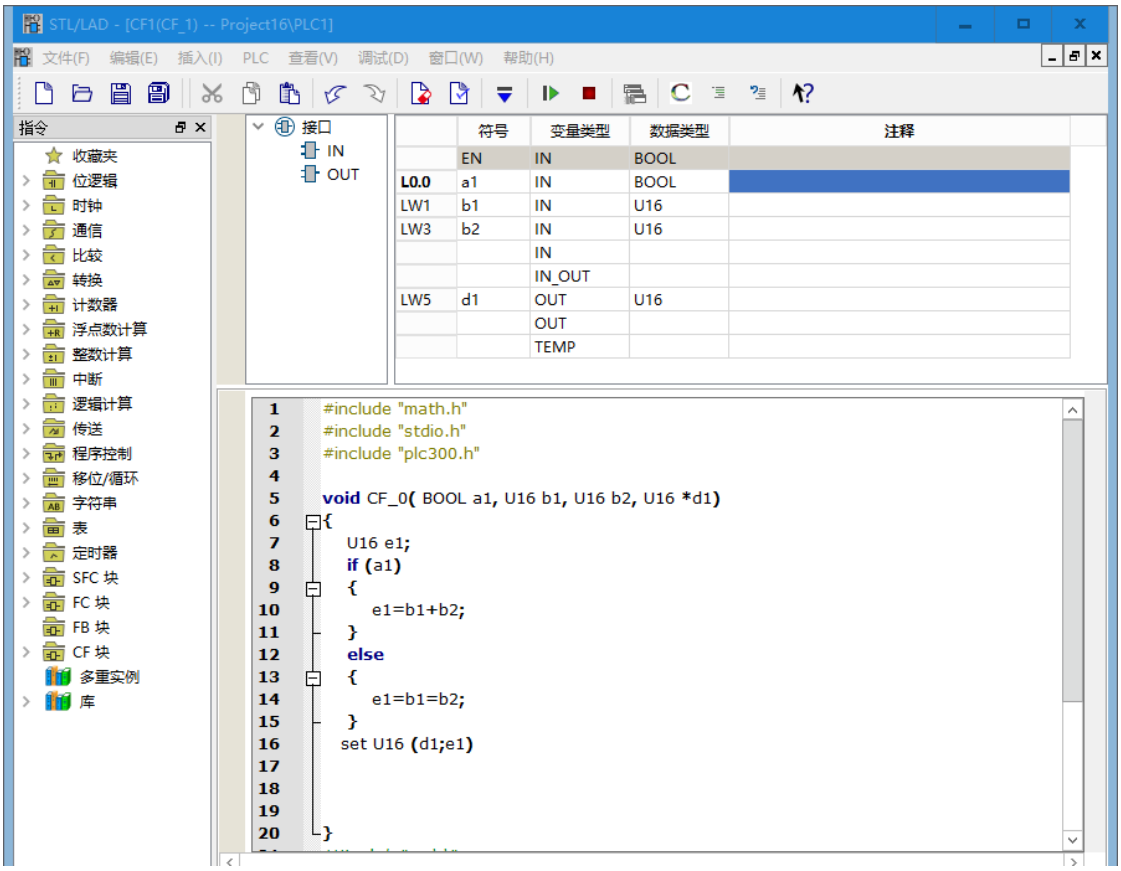



5、在局部变量窗口中定义需要外部输入或输出的参数值，全部定义完成后点击“C”初始化 C 程序块，重新生成函数模板。C 函数的名称和函数参数不要随便改动。

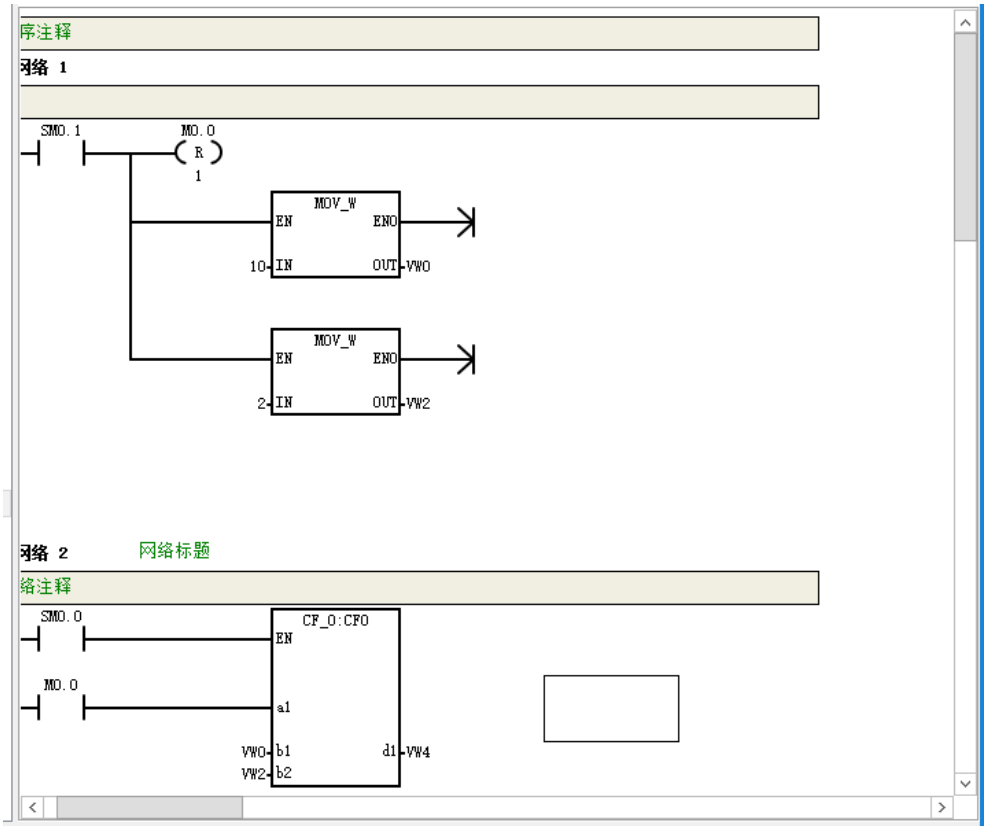



**<备注>** 局部变量窗口最多只能定义 16 个变量。

6、在生成的模板中，编写程序，编程完成后点击执行编绎操作，输出窗口中若无提示错误，即可被 OB1 或其他 FC 调用。从参数符号表定义的参数，输入参数会转变 C 语言相应类型的输入参数，参数可以直接使用；而输入输出和输出参数，会变成相应类型的指针，通过指针在 C 函数里将结果写回 STL 系统，即 C 函数无返回值，返回值只能通过指针传回。




7、编写完成的 C 函数子程序，需在 OB1 主程序或者其它执行的 FC 中调用。当执行程序块下载时，C 子程序也将一同下载到 PLC 中。（注：C 函数内部不可在线监控，但是在梯形图调用处是可以监控输入输出的）。以下程序执行的功能为：如果 M0.0=0，则 vw4=vw0-vw2；如果 M0.0=1，则 vw4=vw0+vw2。




## 15.2 C 语言基础

### 15.2.1 头文件

编译器提供了许多在头文件中声明的标准函数。头文件也称为包含文件，头文件作为一种包含功能函数、数据接口声明的载体文件，主要用于保存程序的声明。

- <stdio.h> 输入输出函数；
- <stdarg.h> 支持变元个数可变函数的宏；
- <math.h> 数学浮点函数；
- <stdlib.h> 内存分配函数；
- <string.h> 字符串处理函数；
- <stdbool.h> bool 类型和布尔值 true 和 false；
- <complex.h> 支持复数；
- <ctype.h> 字符分类函数；
- <wctype.h> 宽字符转换函数。

头文件包含了各种函数的声明以及各种变量的声明，MagicWorks PLC 新建 C 程序块后会默认加入以下三个头文件。

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

其中，头文件<stdio.h>含有大量高级输入/输出(I/0)函数的声明，是 C 语言中的标准输入输出头文件，stdio 即 stand input output 的缩写，程序中含有输入输出操作，就必须使用该头文件。头文件<math.h>中声明了常用的一些数学运算，比如乘方，开方运算、求三角函数、对数之类的函数，可以直接调用。

plc300.h 是合信自己开发的库 libplc300.a 的头文件，库 libplc300.a 主要使用户能够调用底层函数和 STL PLC 系列进行交互，用户可以使用的函数可在头文件 plc300.h 中查看，头文件 plc300.h 在编程软件安装目录\ccompile\include\plc300.h 里。

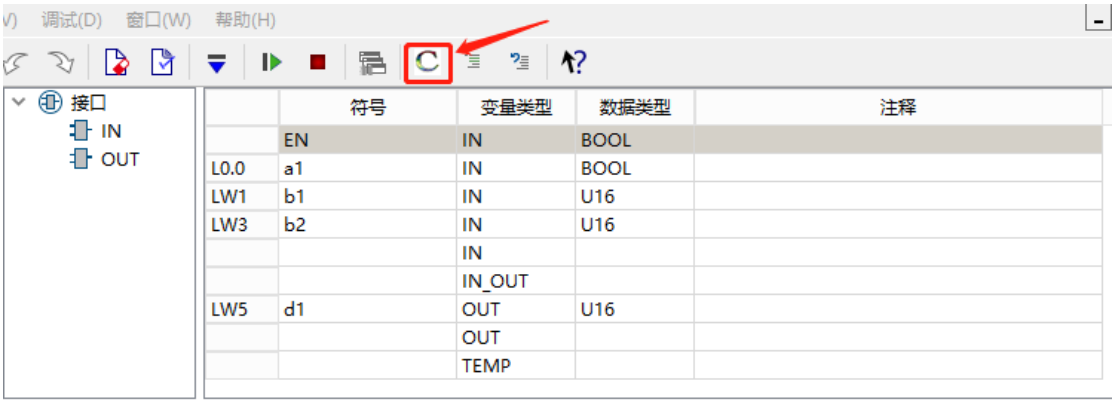
### 15.2.2 变量

**1、定义变量**

变量定义的一般形式：数据类型　变量名

```
例如：FP32     real1;              //定义 32 位浮点数
      S16      int1;               //定义 16 位有符号整型
 const FP32 PI=3.1415926;          //定义 32 位浮点数常量 PI
```

局部变量窗口定义的参数值，执行“初始化 C 程序块”，程序编程窗口会自动将参数定义到子程序中。




```
1  #include "math.h"
2  #include "stdio.h"
3  #include "plc300.h"
4
5  void CF_0( BOOL a1, U16 b1, U16 b2, U16 *d1)
```

注意：

- 变量名必须以字母或下划线开头，名字中间只能由字母、数字和下划线“_”组成；最后一个字符可以是类型说明符。
- 变量必须先定义，后使用。
- 一个变量名只能定义一次，变量建议定义在程序头部，变量定义不允许使用中文变量，且严格区分大小写。
- 一些较简单的字符变量名容易跟一些原有计数器混淆，例如 c 表示计数器，c1 就不允许使用。

**2、static 局部变量与普通变量**

局部变量与普通变量的区别

1) 内存分配和释放

• 普通局部变量只有执行到变量定义的语句时才分配空间。
• static 局部变量在编译阶段（函数还没有执行），变量的空间已经分配。
• 普通局部变量离开作用域{}，自动释放其空间，也就无法使用此变量。
• static 局部变量只有在整个程序结束的时候才将其自动释放。

2) 初始化
• 普通局部变量不初始化，为随机值。
• static 局部变量不初始化，为 0。
• static 局部变量初始化语句只有第一次执行时有效，但是可以赋值多次。
• static 局部变量只能用它常量初始化。
变量在全局数据区分配内存空间；编译器自动对其初始化其作用域为局部作用域，当定义它的函数结束时，其作用域随之结束。
C 语言编程举例如下：只要 var1 不等于 0，则 var2 加一。

```
void CF_3( S16 var1, S16 *var2 )
{
  static S16 temp;
  if(var1!=0)
  {
    temp=temp+1;
  }
  setS16(var2,temp);
}    //temp 变量用 static 修饰符
```

### 15.2.3 注释的使用

编写 C 语言时使用注释有助于对代码的理解，注释不进行编绎，不会影响程序的执行。在 C 语言中有两种注释方式：

- 以/*开始、以*/结束的块注释（block comment）
- 以//开始、以换行符结束的单行注释(line comment)

Magicworks PLC 中既可手动注释，也可以采用注释键进行注释。点击菜单栏中的注释键 即可注释，点击菜单栏中的取消注释键 即可取消注释。

```
void CF_3( S16 i, S16 *j )//令i=3
{
    S16 temp;
    temp=++i;       //i,temp的值为4
    temp=i++;       //temp的值为4，然后i的值变为5
    temp=-(i++);    //temp的值为-5，然后i的值变为6
    setS16(j,temp);

/* 这段代码为测试代码
    演示注释的方法
        —2020-02-16*/
}
```

### 15.2.4 各类数值型数据间的混合运算

在进行运算时，不同数据的类型需要转换成同一类型数据，然后进行运算。

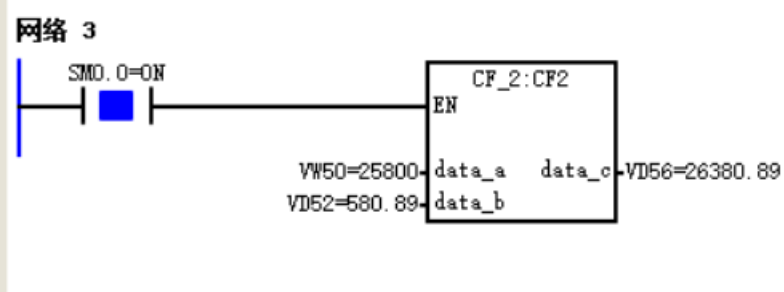
运算过程：先自动转换成同类数据，再进行运算。
转换规则：低字节类型转换成高字节类型。

C 语言编程例程：c = S16 a + FP32 b，a 将自动转换为 FP32 的数据类型，再与 b 相加，结果得到一个 FP32 型的 c。

C 语言编程如下：

```
#include "math.h"
#include "stdio.h"
#include "plc300.h"
void CF_2( S16 data_a, FP32 data_b, FP32 *data_c )
{
   FP32 temp;
   temp=data_a+data_b;
   setFP32(data_c,temp);
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：




## 15.2.5 强制类型转换

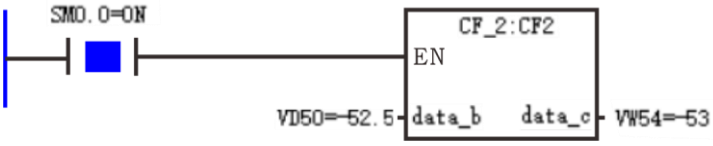
将指定表达式的值转换为指定类型，需要编程者指定需要转换的类型。
形式：（类型名）（表达式）
例如：(S16)(a+b)
说明：1)类型名和表达式需加括号
　　2)在强制类型转换时，得到一个所需类型的中间变量，原来变量类型未发生变化。

C 语言编程例程：将一个 32 位浮点数四舍五入取整然后转换成 16 位整数。

```
void CF_2(FP32 data_b, S16 *data_c )
{
   S16 temp;
   If (data_b>0)
   {
      temp=(S16)(data_b+0.5);
   }
   else
   {
```

```
    temp=(S16)(data_b-0.5);
  }
  setS16(data_c,temp);
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：




## 15.2.6 数组

数组是一组有序数据的集合，若将有限个类型相同的变量的集合命名，这个名称为数组名；数组中每个元素的数据类型都相同，用户通过数组名和下标来确定数组中的元素。

一维数组的定义方式：类型说明符 数组名[整形常量表达式]
例如：FP32 real0[10];
说明：定义了一个数组，数组名为 real0，有 10 个元素，每个元素的类型均为 FP32 型；这 10 个元素分别是 real[0]、real[1]、real[2]、……real[8]、real[9]。

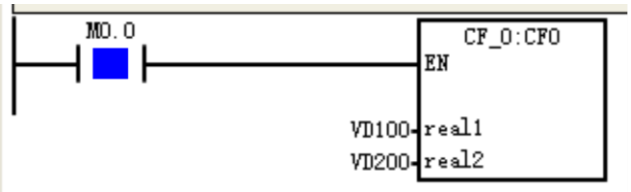
初始化数组：数组可在声明的初始化，初始化数组时，各个索引变量的值要封闭在{}内，并用逗号分隔，例如：S16 aa[3]={2,6,1};等价于：S16 aa[3];aa[0]=2;aa[1]=6;aa[2]=1;

注意：
• 下标从 0 开始。如：数组第 1 个元素为 real1[0]，第 8 个元素为 real1[7]。
• C 语言一般是不允许对数组的大小做动态定义的，如 FP32 real1[n]，这是不允许的。
• 语言不检查数组下标越界，因此使用时不要越界使用，本例中数组元素个数是 10 个，下标范围为：0-9，不能超出 9。
• 数组中的各元素的存储是有先后顺序的，它们在内存中按照这个先后顺序连续存放在一起。

C 语言编程示例：将 VD100 开始连续 10 个值，复制到 VD200 后。

```
void CF_0( void* real1, void* real2 )
{
  S16 i,j;
  FP32 temp[10] ={0,0,0,0,0,0,0,0,0,0};    //定义数组
  for (i=0;i<10;i++)
    {
      temp[i]=getFP32(real1+4*i);          //数组地址
    }
  for (j=0;j<10;j++)
    {
      set FP32((real2+4*j),temp[j]);
    }
}
```

调用生成的 CF0




状态表结果如下所示：

| | 地址 | 格式 | 当前值 |
|---|---|---|---|
| 1 | VD100 | 浮点数 | 0.1111 |
| 2 | VD104 | 浮点数 | 0.2222 |
| 3 | VD108 | 浮点数 | 0.3333 |
| 4 | VD112 | 浮点数 | 0.4444 |
| 5 | VD116 | 浮点数 | 0.5555 |
| 6 | VD120 | 浮点数 | 0.6666 |
| 7 | VD124 | 浮点数 | 0.7777 |
| 8 | VD128 | 浮点数 | 0.8888 |
| 9 | VD132 | 浮点数 | 0.9999 |
| 10 | VD136 | 浮点数 | 1.0 |
| 11 | | 有符号 | |
| 12 | VD200 | 浮点数 | 0.1111 |
| 13 | VD204 | 浮点数 | 0.2222 |
| 14 | VD208 | 浮点数 | 0.3333 |
| 15 | VD212 | 浮点数 | 0.4444 |
| 16 | VD216 | 浮点数 | 0.5555 |
| 17 | VD220 | 浮点数 | 0.6666 |
| 18 | VD224 | 浮点数 | 0.7777 |
| 19 | VD228 | 浮点数 | 0.8888 |
| 20 | VD232 | 浮点数 | 0.9999 |
| 21 | VD236 | 浮点数 | 1.0 |

## 15.3 运算符与表达式

### 15.3.1 赋值运算符与赋值表达式

1、赋值运算符=

赋值表达式：将一个数值赋给一个变量或将一个变量的值赋给另一个变量。

赋值形式：变量名=常量或表达式；

赋值作用：将右边常量或表达式的值赋给左边的变量。如: a=3、a=b*c

注意：

- 所有的语句，包括赋值语句，必须要以分号“；”结尾。
- 赋值后，原来左边的值被右边表达式的值替换
- 如果右边表达式的值类型与左边变量的类型不一致，以左边变量的类型为准，将右边表达式值的类型自动转换为左边变量的类型。

2、多重赋值运算符：多重赋值是将表达式的值同时赋给几个变量，在这种情况下，要把被赋值的变量名按次序写在赋值号的左边。
赋值形式：变量 1=变量 2=…=变量 n=表达式
等价于：变量 1=(变量 2=(…=(变量 n=表达式)))
例如:a=b=c=7;

3、复合赋值运算符：在赋值符“=”之前加上其它运算符。常见的有：+=、-=、*=、/=、%=
复合赋值运算的功能：与左边的变量进行复合运算后再赋给左边的变量。
例如:

```
a+=3;                          等价于 a=a+3
x*=y+8;                        等价于 x=x*(y+8)
x%=3;                          等价于 x=x%3
int a=1,b=2,c=2;   c*=a-b;     等价于 c=c*(a-b)=2*(-1)=-2
```

### 15.3.2 算术运算符

C 语言中的算术运算符有:
+：加或取正值运算符
-：减或取负值运算符
*：乘运算符
/：除运算符
%：取余运算符
其中，加减乘除均为取余运算符，它们要求有两个运算对象。

### 15.3.3 自增自减运算符

运算符说明：++表示自增，变量值加 1，
--表示自减，变量值减 1。
用法 1：单独作为一个表达式被使用。例如：i++或者++i 等价于 i=i+1

用法 2：出现在其他表达式中参与运算，++i/--i 表示 i 自增/自减 1 后再参与其它运算；而 i++/i--则是 i 参与运算后，i 的值再自增/自减 1。

例如：S16 j,i=3;

```
j=++i;      // i,j 的值为 4
j=i++;      // j 的值为 4，然后 i 的值变为 5
j=-(i++);   // j 的值为-5，然后 i 的值变为 6
```

### 15.3.4 关系运算符与表达式

关系运算：比较两个操作数是否满足给定的关系，用于简单的条件判断。在 C 语言中有以下关系运算符：
（1） < 小于
（2） <= 小于或等于
（3） > 大于

（4） >= 大于或等于
（5） == 等于
（6） != 不等于
关系运算符的结果只能是 0 或者 1，值为真时结果为 1，值为假时结果为 0。
例如：3.24<=2.98; //值为假，运算结果为 0
0.5!=3+1; //值为真，运算结果为 1

关系表达式：用关系运算符将两个表达式（算术表达式、关系表达式、逻辑表达式、赋值表达式、字符表达式）连接起来的式子，称为关系表达式
关系表达式的一般形式为：表达式 关系运算符 表达式
关系运算符的优先级低于算术运算符，高于赋值运算符。
例如：a=1+2>3-1; //等价于 a=((1+2)>(3-1))，结果为 a=1

### 15.3.5 逻辑运算符与表达式

逻辑运算也称布尔运算，用于复杂的条件判断。

| 运算符 | 含义 | 优先级 | ST 描述 |
|---|---|---|---|
| ! | 逻辑非 | 1 | NOT |
| && | 逻辑与 | 2 | AND |
| ll | 逻辑或 | 3 | OR |

逻辑表达式的一般形式：表达式 逻辑运算符 表达式

逻辑运算真值表

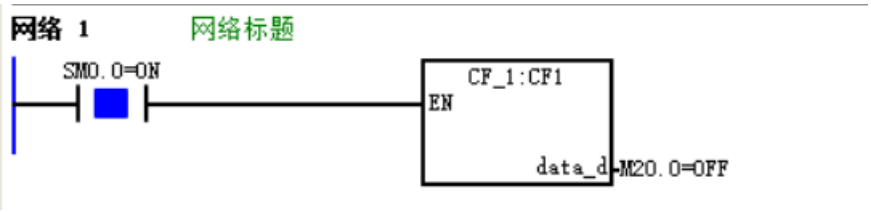
| A 取值 | B 取值 | ！A | A&&B | AllB |
|---|---|---|---|---|
| 非 0 | 非 0 | 0 | 1 | 1 |
| 非 0 | 0 | 0 | 0 | 1 |
| 0 | 非 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

C 语言编程例程：当(a>b)&&(b>c)结果为真时 d=1，当(a>b)&&(b>c)结果为假时 d=0。

```
void CF_1( BOOL *data_d )
{
  S16 data_a=3,data_b=2,data_c=1;
  if ((data_a>data_b)&&(data_b>data_c))
  {
    *data_d=1;
  }
  else
  {
```

```
    *data_d=0;
  }
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：



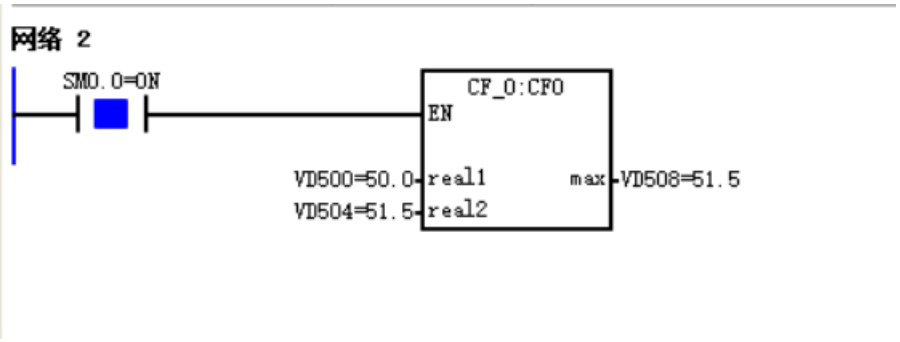

### 15.3.6 条件运算符与条件表达式

条件运算符与条件表达式 有两个符号：问号和分号，它与三个操作数组成三元运算。
一般形式：<表达式 1>？<表达式 2>：<表达式 3>
优先级：逻辑>条件>赋值
例如：S16 max，a=5;b=3;
　　max=a>b?a:b
结果：max=5

C 语言编程例程：比较 2 个值的大小，并将最大值输出。

```
void CF_0( FP32 real1, FP32 real2, FP32 *max )
{
 FP32 max_t;
 max_t=real1>real2?real1:real2;
 setFP32(max,max_t);
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：




## 15.4 for 循环语句

在应用程序中，需要多次执行某些步骤，这个过程叫循环；for 循环是编程语言中一种循环语句，循环程序的设计需要建立程序使它能够循环返回并循环执行自身程序。

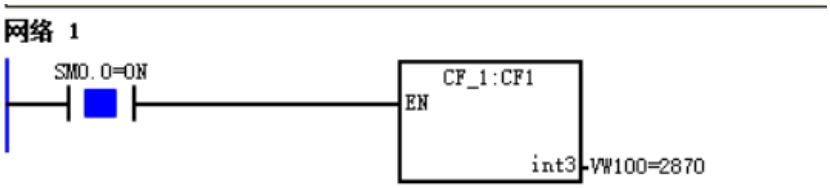
for 循环语句的一般形式：

```
for（单次表达式;条件表达式;末尾循环体）
{
中间循环体;
}
例如：for (i=1;i<=100;i++)
        {
          sum=i+sum;
        }
```

for 循环中的语句反复被执行，每次循环后，变量 i 会自动加 1。i=1 和 i<=100 是循环控制变量的起始值和终止值。当控制变量达到终止值时，程序就执行 for 语句后的下一条指令。

```
例如：求 1-20 的平方和！
void CF_1( S16 *int3 )
{
  S16 int3_t=0;
  S16 i;
  for (i=1;i<=20;i++)
  {
    int3_t=int3_t+i*i;
  }
  setS16(int3,int3_t);
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：




## 15.5 选择语句

### 15.5.1 if 选择结构语句

if 选择结构：判断 if 条件，根据判断结果执行后续操作的选择结构。

**1）单分支 if 选择结构的一般格式：**

```
if（表达式）        //表达式：指判断条件，真为 1，假为 0
{
  执行语句；      //仅当表达式为真时，才执行此处执行语句
}
```

例如：输入一个数，判断能否被 4 或者 7 整除，满足条件输出为 1。

```
void CF_0( S16 int1, S16 *out1 )
{
   S16 out_t=0;
   if (int1%4==0||int1%7==0)
   {
      out_t=1;
   }
   setS16(out1,out_t);
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：



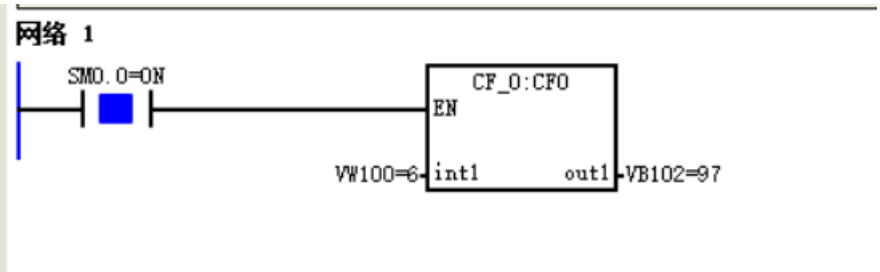

**2）双分支 if 选择结构的一般格式：**

```
if（表达式）      //表达式：指判断条件，真为 1，假为 0
 {
    执行语句；     //仅当表达式为真时，才执行此处执行语句
 }
else
{
     执行语句；     //条件为 0，执行 else 后的语句
}
```

例如：输入一个数，偶数输出 a，奇数输出 b。

```
void CF_0(S16 int1, U8 *out1 )
{
   char out;
   if (int1%2==0)
   {
      out='a';
   }
   else
   {
      out='b';
   }
   setU8(out1,out);
}
```

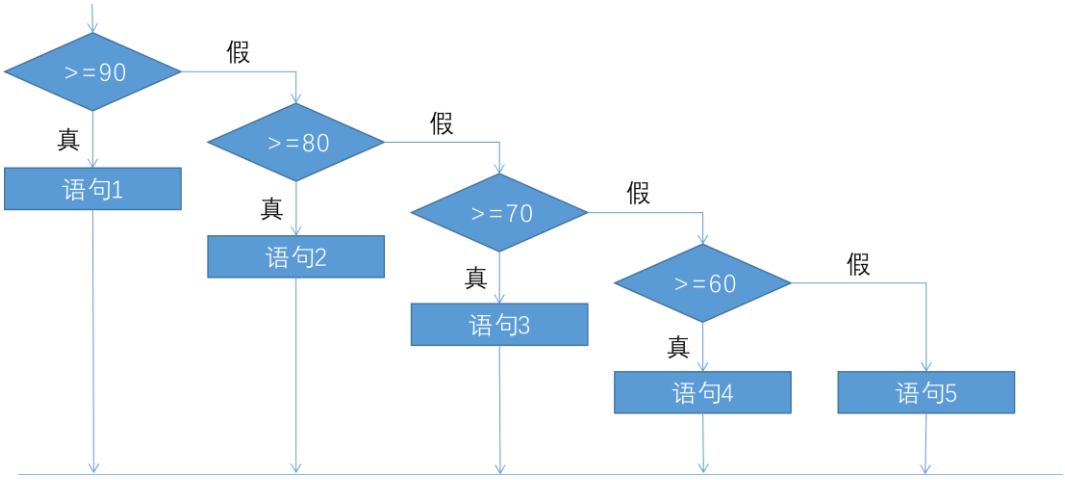
编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：




**3)多重 if 选择结构的一般格式：**

```
if（表达式）
{
   执行语句；
}
else if（表达式 1）
{
   执行语句；
}
else if（表达式 2）
{
   执行语句；
}
……
else
{
     执行语句；
}       //最后一个 else{}可以省略，有则必须放在 else if  后面
```

例如：输入成绩，大于等于 90 为 A，80-89 为 B，70-79 为 C，60 到 69 为 D，低于 60 为 E。多重 if 执行的流程图如下：




C 语言编程如下：

```
void CF_1( S16 score, U8 *level )
{
```

```
    char level_t;
    if (score>=90)
    {
      level_t='A';
    }
    else if (score>=80)
    {
      level_t='B';
    }
    else if(score>=70)
    {
      level_t='C';
    }
    else if(score>=60)
    {
      level_t='D';
    }
    else
    {
      level_t='E';
    }
    setU8(level,level_t);
}
```

### 15.5.2 switch 多分支选择语句

switch 语句是多分支条件判断语句，它根据表达式的值使程序从多个分支中选择一个用于执行的分支。

switch 的一般格式：

```
switch（表达式）
{
   case  常量表达式 1 :   语句 1;
                 break;
    case  常量表达式 2 :   语句 2;
                 break;
    ……
   case  常量表达式 n :   语句 n;
                 break;
    default :   语句 n+1;
}
```

注意：

- case 条件执行语句后需加 break 跳出 switch 结构，否则无法分支；

• 常量表达式的值必须互不相等，否则有二异性；

• case 的顺序无关紧要， default 可有可无。

• switch 语句不等同于 if 语句，switch 只能进行值的相等性检查，不是逻辑判断；if 语句不但可以进行值的相等性判断，还可以计算关系表达式或逻辑表达式，进行逻辑判断的真假。

例如：

```
void CF_0( S16 Level, BOOL *heat1, BOOL *heat2, BOOL *heat3, BOOL *fan )
{
  *heat1=*heat2=*heat3=*fan=0;
  switch (Level)
  {
   case 1:*fan=1;
          break;
   case 2:*fan=0;
          break;
   case 3:*heat1=1;
          break;
   case 4:*heat1=*heat2=1;
          break;
   case 5:*heat1=*heat2=*heat3=1;
          break;
   default:*heat1=*heat2=*heat3=1;
  }
}
```

## 15.6 libplc300.a 库介绍

libplc300.a 库为 Magicworks PLC 自带的库，该库中含有从 STL 取值和将值写回 STL 的函数以及 STL 系统与 C 语言系统的数据转换操作函数，用户可直接调用这些函数。用户可以使用的函数可在头文件 plc300.h 里查看，头文件 plc300.h 在编程软件安装目录\ccompile\include\plc300.h 里。

下面将对 C 程序块的数据类型以及各种操作函数进行介绍。

### 15.6.1 CF 块参数符号表中可选择的数据类型

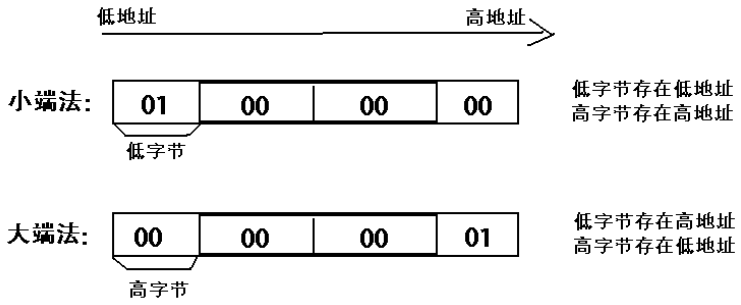
| 关键字 | 数据类型 | 注释 |
|---|---|---|
| unsigned char | BOOL | 位 |
| unsigned char | U8 | 8 位无符号数 |
| signed char | S8 | 8 位有符号数 |
| unsigned short | U16 | 16 位无符号数 |

| | | |
|---|---|---|
| signed short | S16 | 16 位有符号数 |
| unsigned int | U32 | 32 位无符号数 |
| signed int | S32 | 32 位有符号数 |
| float | FP32 | 浮点数 |
| | void* | 指针：指向任何类型的数据 |

**<备注>** void*可以指向任何类型的数据，所有提供指针均使用 void* 型，让用户在 C 代码中进行强制转换，返回值为空，返回结果，可通过指针传递。

### 15.6.2 大小端转换

由于 STL 系统的数据使用的是大端模式，而 C 语言使用的是小端模式，因此直接操作指针赋值会有字节序混乱的问题，因此我们提供了一组通过指针从 STL 取值和将值写回 STL 的函数，函数的具体形式在 MagicWorks PLC 安装目录\ccompile\include\plc300.h 里。
大端模式：数据的高字节保存在内存的低地址中，而数据的低字节保存在内存的高地址中。
小端模式：是指数据的高字节保存在内存的高地址中，而数据的低字节保存在内存的低地址中。



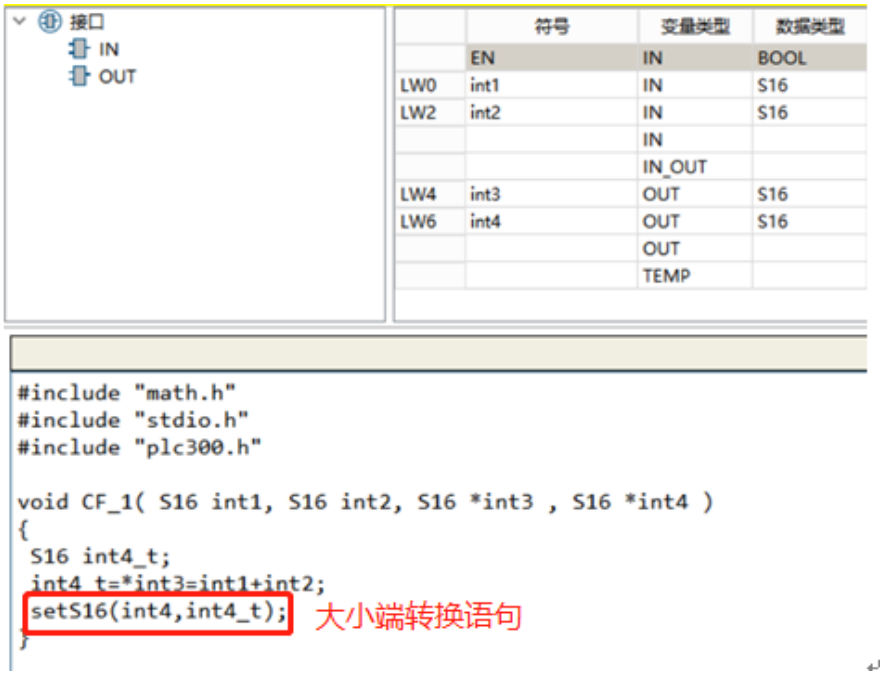

参数通过指针在 C 语言系统和 STL 系统进行传递时，由于 C 语言系统使用的是小端模式，而 STL 系统使用是的大端模式，所以需要调用大小端转换接口。

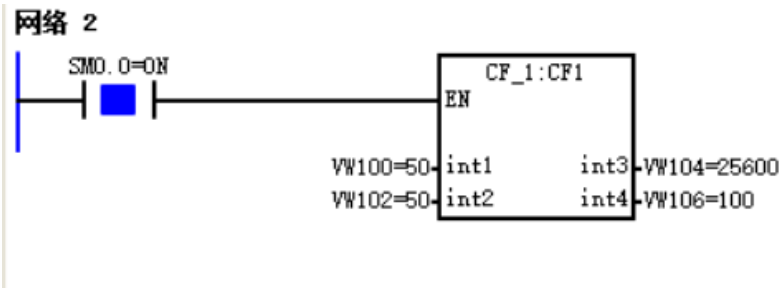
### 15.6.3 STL 系统与 C 语言系统的数据转换操作函数

1、通过指针将值写回 STL 环境：由于输入输出和输出参数会变成相应类型的指针类型，且大小端转换的问题，所有的输入输出或输出参数都要通过以下指令将结果从 C 语言写回到 STL 环境。

```
void setU8(U8 *P, U8 d);          //将一个 U8 数据，从 C 语言写回 STL 系统
void setS8(S8 *P, S8 d);          //将一个 S8 数据，从 C 语言写回 STL 系统
void setU16(U16 *P, U16 d);       //将一个 U16 数据，从 C 语言写回 STL 系统
void setS16(S16 *P, S16 d);       //将一个 S16 数据，从 C 语言写回 STL 系统
void setU32(U32 *P, U32 d);       //将一个 U32 数据，从 C 语言写回 STL 系统
void setS32(S32 *P, S32 d);       //将一个 S32 数据，从 C 语言写回 STL 系统
void setFP32(FP32 *P, FP32 d);    //将一个 FP32 数据，从 C 语言写回 STL 系统
void setBOOL(BOOL *P, BOOL d);    //将一个 BOOL 数据，从 C 语言写回 STL 系统
```

例子如下：输入 int1 与 int2 相加，输出 int4 进行了大小端转换，输出结果正确；输出 int3 没有进行大小端转换，输出结果不正确。




2、通过指针从 STL 取值：如果在 C 函数输入参数中直接调用 PLC 的 V 区、M 区等寄存器，可以将 PLC 寄存器的指针输入给 C 函数，通过取指针值的方式获取 PLC 内存区的值到 C 语言，但必须通过以下函数进行转换。

U8 getU8(U8 *p); //从 p 地址 取得一个 C 语言系统可用的 U8 数据

S8 getS8(S8 *p); //从 p 地址 取得一个 C 语言系统可用的 S8 数据

U16 getU16(U16 *p); //从 p 地址 取得一个 C 语言系统可用的 U16 数据

S16 getS16(S16 *p); //从 p 地址 取得一个 C 语言系统可用的 S16 数据

U32 getU32(U32* p); //从 p 地址 取得一个 C 语言系统可用的 U32 数据

S32 getS32(S32* p); //从 p 地址 取得一个 C 语言系统可用的 S32 数据

FP3 getFP32(FP32* p); //从 p 地址 取得一个 C 语言系统可用的 FP32 数据

FP3 getBOOL(BOOL* p); //从 p 地址 取得一个 C 语言系统可用的 BOOL 数据

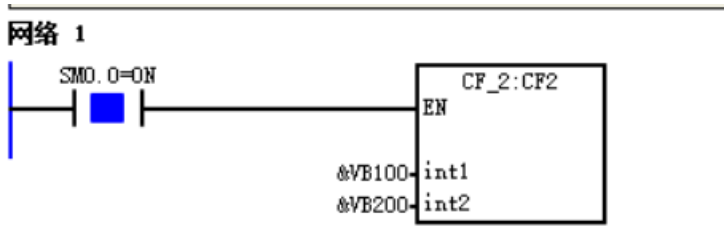
**<备注>** 不通过指针从 STL 取值：输入变量不通过指针的方式，而是输入变量直接获取 PLC 寄存器的值，就不需要使用 get 函数。

C 语言编程如下：

1、将 VW100 开始连续 20 个 VW 值赋给 VW200 地址段。

```
void CF_2( void* int1, void* int2 )
{
   S16 r,i;
   for(i=0;i<20;i++)
   {
      r=getS16(int1+2*i);
      setS16((int2+2*i),r);
   }
}
```

调用生成的 CF 块




状态表结果如下所示：



| | | | |
|---|---|---|---|
| 3 | VW100 | 有符号 | +1 |
| 4 | VW102 | 有符号 | +2 |
| 5 | VW104 | 有符号 | +3 |
| 6 | VW106 | 有符号 | +4 |
| 7 | VW108 | 有符号 | +5 |
| 8 | VW110 | 有符号 | +6 |
| 9 | VW112 | 有符号 | +7 |
| 10 | VW114 | 有符号 | +8 |
| 11 | VW116 | 有符号 | +9 |
| 12 | VW118 | 有符号 | +10 |
| 13 | VW120 | 有符号 | +11 |
| 14 | VW122 | 有符号 | +12 |
| 15 | VW124 | 有符号 | +13 |
| 16 | VW126 | 有符号 | +14 |
| 17 | VW128 | 有符号 | +15 |
| 18 | VW130 | 有符号 | +16 |
| 19 | VW132 | 有符号 | +17 |
| 20 | VW134 | 有符号 | +18 |
| 21 | VW136 | 有符号 | +19 |

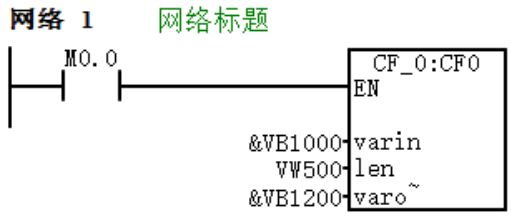
| | | | |
|---|---|---|---|
| 1 | VW200 | 有符号 | +1 |
| 2 | VW202 | 有符号 | +2 |
| 3 | VW204 | 有符号 | +3 |
| 4 | VW206 | 有符号 | +4 |
| 5 | VW208 | 有符号 | +5 |
| 6 | VW210 | 有符号 | +6 |
| 7 | VW212 | 有符号 | +7 |
| 8 | VW214 | 有符号 | +8 |
| 9 | VW216 | 有符号 | +9 |
| 10 | VW218 | 有符号 | +10 |
| 11 | VW220 | 有符号 | +11 |
| 12 | VW222 | 有符号 | +12 |
| 13 | VW224 | 有符号 | +13 |
| 14 | VW226 | 有符号 | +14 |
| 15 | VW228 | 有符号 | +15 |
| 16 | VW230 | 有符号 | +16 |
| 17 | VW232 | 有符号 | +17 |
| 18 | VW234 | 有符号 | +18 |
| 19 | VW236 | 有符号 | +19 |

2、将 V 区开始的 LEN 个 INT 值进行大小排序，显示到另一个 V 区。

```
void CF_0( void* varin, S16 len, void* varout )
{
   S16 i,j,n,k,l;
   S16 temp[len];
   for (l=0;l<len;l++)     //将 V 区的值存入 temp 数组
   {
      temp[l]=getS16(varin+2*l);
   }
```

```
  for (i=len;i>1;i--)        //冒泡法将大小值排序,每次最大就像气泡一样"浮"到数组的上面
  {
    for(j=0;j<i-1;j++)       //依次比较相邻的两个元素,使较大的元素向后移
  if (temp[j]<temp[j+1])
  {
    n=temp[j];
    temp[j]=temp[j+1];
    temp[j+1]=n;
  }
  }
   for(k=0;k<len;k++)    //将 temp 数组值传出到 V 区
  {
    setS16((varout+2*k),temp[k]);
  }
}
```

编译后得到的 CF 块，通过 OB1 或者其他 FC 调用：




状态表结果如下所示：

| 地址 | 格式 | 当前值 |
|---|---|---|
| VW1000 | 有符号 | +24 |
| VW1002 | 有符号 | +231 |
| VW1004 | 有符号 | +2 |
| VW1006 | 有符号 | +141 |
| VW1008 | 有符号 | +4 |
| VW1010 | 有符号 | +23 |
| VW1012 | 有符号 | +1 |
| VW1014 | 有符号 | +7 |
| VW1016 | 有符号 | +6 |
| VW1018 | 有符号 | +4 |
| VW1020 | 有符号 | +0 |
| VW1022 | 有符号 | +578 |
| VW1024 | 有符号 | +98 |
| VW1026 | 有符号 | +74 |
| VW1028 | 有符号 | +15 |
| VW1030 | 有符号 | +97 |
| VW1032 | 有符号 | +88 |
| VW1034 | 有符号 | +17136 |
| VW1036 | 有符号 | +11 |
| VW1038 | 有符号 | +17136 |

| | | |
|---|---|---|
| VW1200 | 有符号 | +17136 |
| VW1202 | 有符号 | +17136 |
| VW1204 | 有符号 | +578 |
| VW1206 | 有符号 | +231 |
| VW1208 | 有符号 | +141 |
| VW1210 | 有符号 | +98 |
| VW1212 | 有符号 | +97 |
| VW1214 | 有符号 | +88 |
| VW1216 | 有符号 | +74 |
| VW1218 | 有符号 | +24 |
| VW1220 | 有符号 | +23 |
| VW1222 | 有符号 | +15 |
| VW1224 | 有符号 | +11 |
| VW1226 | 有符号 | +7 |
| VW1228 | 有符号 | +6 |
| VW1230 | 有符号 | +4 |
| VW1232 | 有符号 | +4 |
| VW1234 | 有符号 | +2 |
| VW1236 | 有符号 | +1 |

### 15.6.4 获取内存基地址的操作函数

```
U8   *getLBase(void);          //获取地址 LB0 的值
U8   *getIBase(void);          //获取地址 IB0 的值
U8   *getQBase(void);          //获取地址 QB0 的值
U16   *getAQBase(void);        //获取地址 AQW0 的值
U16   *getAIBase(void);        //获取地址 AIW0 的值
U8  *getSMBase(void);          //获取地址 SMB0 的值
U8   *getVBase(void);          //获取地址 VB0 的值
U8   *getTVBase(void);         //获取地址 T0 的值（定时器值）
U8   *getTBBase(void);         //获取地址 T0 的值（定时器位）
U8   *getSBase(void);          //获取地址 SB0 的值
U8   *getCVBase(void);         //获取地址 C0 的值（计数器值）
U8   *getCBBase(void);         //获取地址 C0 的值（计数器位）
U8   *getACBase(void)          //获取地址 AC0 的值
```

# 附录

您可以通过本附录快速浏览你所需的内容，本附录包括以下信息：

- 订货信息
- 缩略术语
- 电源预算
- 编程卡的使用
- CT_Modbus 库的使用
- PID_T 库的使用介绍
- 以太网 CT_socket 通信库使用介绍
- CT_tcp_server_lib 库的使用介绍
- Trace 追踪功能
- 特殊存储器说明
- 指令速查

## A 订货信息

表 A-1 订货信息一览表

| 规格描述 | 订货号 |
|---|---|
| **CPU** | |
| H35-00 主控模块，128KB 程序空间，128KB 数据空间，16KB 掉电保持数据空间；双 PPI/自由通信接口，1 个以太网接口，1 个 USB 接口 | CTH3 H35-000S1 |
| H36-00 主控模块，256KB 程序空间，1MB 数据空间，32KB 掉电保持数据空间；1 个 PPI/自由通信接口，1 个以太网通信口，1 个 EtherCAT 通信口，1 个 USB 接口，基本运动控制功能 | CTH3 H36-000S1 |
| H36-01 主控模块，256KB+64KB 程序空间，1MB 数据空间，32KB 掉电保持数据空间；1 个 PPI/自由通信接口，1 个以太网通信口，1 个 EtherCAT 通信口，1 个 CAN 通信接口，1 个 USB 接口，支持单轴运动控制，插补、电子凸轮和电子齿轮功能 | CTH3 H36-001S2 |
| H31-00 高性能薄型 CPU，192KB+64KB 程序空间，512KB 数据空间，16KB 掉电保持数据空间，双 PPI/自由通讯口，1 个以太网通信口，1 个 USB 接口，支持基本运动控制功能，最大机架数 1 个 | CTH3 H31-000S1 |

| | |
|---|---|
| H32-01 运动控制薄型 CPU，256KB+64KB 程序空间，1MB 数据空间，32KB 掉电保持数据空间，1 个 PPI/自由通讯口，1 个以太网通信口，1 个 USB 接口，1 个 CAN 接口，1 个 EtherCAT 接口，支持单轴运动控制，插补、电子齿轮及电子凸轮功能，最大机架数 1 个 | CTH3 H32-001S2 |
| **电源模块** | |
| PWR-02 电源模块，输入 85~264V AC，输出 24V DC/2A | CTH3 PWR-020S1 |
| **中继模块** | |
| INT-00 中继模块 | CTH3 INT-000S1 |
| **CAN 主站模块** | |
| CAN-1M 主站通信模块，1 个 CAN 通信口 | CTH3 CAN-1M0S1 |
| **高速计数模块** | |
| HSC-02高速计数模块，2路差分/单端信号输入 | CTH3 HSC-020S1 |
| **脉冲输出模块** | |
| HSP-04脉冲输出模块，4路差分/单端信号输出 | CTH3 HSP-040S1 |
| **数字量模块** | |
| DIT-08 数字量模块，数字量输入 8 x 24VDC | CTH3 DIT-080S1 |
| DIT-16 数字量模块，数字量输入 16 x 24VDC | CTH3 DIT-160S1 |
| DIT-32 数字量模块，数字量输入 32 x 24VDC | CTH3 DIT-320S1 |
| DQT-08 数字模块，数字量输出 8 x 24VDC | CTH3 DQT-080S1 |
| DQT-16 数字量模块，数字量输出 16 x 24VDC | CTH3 DQT-160S1 |
| DQT-32 数字量模块，数字量输出 32 x 24VDC | CTH3 DQT-320S1 |
| DQR-08 数字量模块，数字量输出 8 x 继电器 | CTH3 DQR-080S1 |
| DQR-16 数字量模块，数字量输出 16 x 继电器 | CTH3 DQR-160S1 |
| **模拟量模块** | |
| AIS-04 模拟量模块，模拟量电压电流输入，4AI x 12bits | CTH3 AIS-040S1 |
| AMS-06 模拟量模块，模拟量电压电流输入输出，4AI x 12bits，2AQ x 12bits | CTH3 AMS-060S1 |
| AIV-08 模拟量模块，模拟量电压输入，8AI x 16bits | CTH3 AIV-080S1 |
| AIC-08 模拟量模块，模拟量电流输入，8AI x 16bits | CTH3 AIC-080S1 |
| AQS-04 模拟量模块，模拟量电压电流输出，4AQ x 12bits | CTH3 AQS-040S1 |
| AQS-08 模拟量模块模拟量电压电流输出，8AQ x 12bits | CTH3 AQS-080S1 |
| **温度模块** | |
| AIT-04 温度模块，热电偶输入模块，4 路 TC，隔离型 16bits 精度 | CTH3 AIT-040S1 |
| AIT-08 温度模块，热电偶输入模块，8 路 TC，隔离型 16bits 精度 | CTH3 AIT-080S1 |
| AIR-04 温度模块，热电阻输入模块，4 路 RTD，隔离型 16bits 精度 | CTH3 AIR-040S1 |
| AIR-08 温度模块，热电阻输入模块，8 路 RTD，隔离型 16bits 精度 | CTH3 AIR-080S1 |

# B 缩略术语

表 B-1 缩略术语解释

| 缩写 | 全称 | 解释 |
|---|---|---|
| AC | Alternating Current | 交流 |
| ADC | Analog to Digital Converter | 模数转换器 |
| COB-ID | Communication Object Identifier | 通信对象ID |
| CPU | Central Processing Unit | PLC 的中央处理单元 |
| DAC | Digital-to-Analog Converter | 数模转换器 |
| DC | Direct Current | 直流 |
| DB | Database | 数据块 |
| FB | Function Block | 功能块 |
| FC | Function | 功能 |
| HMI | Human Machine Interface | 人机界面 |
| HSC | High-speed counter | 高速计数器 |
| IEC | International Electro Technical Commission | 国际电工委员会 |
| IO | Input/Output | 输入/输出 |
| LAD | Ladder Logic | 梯形图编程语言 |
| L+ | -- | 24 VDC 电源 |
| M | -- | 接地 |
| MES | Manufacturing Execution System | 制造执行系统 |
| MPI | Multi Point Interface | 多点接口 |
| NMT | network management | 网络管理 |
| OB | Organization Block | 组织块 |
| OEM | Original Equipment Manufacturer | 原始设备制造商 |
| OPC | OLE (Object Linking and Embedding) for Process Control | 用于过程控制的OLE |
| PDO | Process Data Object | 过程数据对象 |
| PID | Proportion, Integration, Differentiation | 比例、积分、微分 |
| PLC | Programmable Logic Controller | 可编程逻辑控制器 |
| PPI | Point To Point Interface | 点对点接口 |
| RAM | Random Access Memory | 随机存取存储器 |
| RTD | Resistance Temperature Detector | 电阻温度探测器 |
| SDO | Service Data Object | 服务数据对象 |
| SOE | Sequence Of Event | 事件顺序记录 |
| STL | Statement List | 指令表编程语言 |
| TC | Thermocouple | 热电偶 |
| UDP | User Datagram Protocol | 用户数据包协议 |

# C 电源预算

选择好各机架的 CPU、电源模块、中继模块和扩展模块后，还需要确认系统总线的电流消耗和功率消耗是否满足以下条件：

**条件 1：总线电流消耗确认**

内部总线电压为 5VDC，电流由 CPU（无中继模块时）或中继模块提供。每个机架上扩展模块总线电流消耗之和不能超过 CPU 或中继模块允许的最大总线电流。

**条件 2：功率消耗确认**

使用电源模块时，每个机架上其它模块的功率消耗之和不能超过电源模块允许的最大功耗。
使用外部电源时，根据所接的功率之和选择合适功率大小的型号。

表 C-1　5VDC 总线电流供给和消耗

| 产品型号 | 供给电流 | 消耗电流 |
|---|---|---|
| H35-00 | 1600mA | -- |
| H36-00 | 1600mA | -- |
| H36-01 | 1600mA | -- |
| H31-00 | 1600mA | |
| H32-01 | 1600mA | |
| INT-00 | 1600mA | -- |
| DIT-08 | -- | 60mA |
| DIT-16 | -- | 80mA |
| DIT-32 | -- | 130mA |
| DQT-08 | -- | 70mA |
| DQT-16 | -- | 120mA |
| DQT-32 | -- | 210mA |
| DQR-08 | -- | 45mA |
| DQR-16 | -- | 60mA |
| AIS-04 | -- | 50mA |
| AIC-08 | -- | 30mA |
| AIV-08 | -- | 30mA |
| AQS-04 | -- | 40mA |
| AQS-08 | -- | 40mA |
| AMS-06 | -- | 50mA |
| AIT-04 | -- | 50mA |
| AIT-08 | -- | 50mA |
| AIR-04 | -- | 50mA |
| AIR-08 | -- | 50mA |
| HSC-02 | -- | 100mA |
| HSP-04 | -- | 100mA |
| CAN-1M | -- | 100mA |

表 C-2　24VDC 总线电流供给和消耗

| 产品型号 | 供给电流 | 消耗电流 |
|---|---|---|
| PWR-02 | 2000mA | -- |
| H35-00 | -- | 800mA |
| H36-00 | -- | 800mA |
| H36-01 | -- | 800mA |
| H31-00 | -- | 800mA |
| H32-01 | -- | 800mA |
| INT-00 | -- | 800mA |
| DIT-08 | -- | -- |
| DIT-16 | -- | -- |
| DIT-32 | -- | -- |
| DQT-08 | -- | 50mA |
| DQT-16 | -- | 95mA |
| DQT-32 | -- | 180mA |
| DQR-08 | -- | 64mA |
| DQR-16 | -- | 130mA |
| AIS-04 | -- | 65mA |
| AIC-08 | -- | 50mA |
| AIV-08 | -- | 50mA |
| AQS-04 | -- | 110mA |
| AQS-08 | -- | 200mA |
| AMS-06 | -- | 110mA |
| AIT-04 | -- | 50mA |
| AIT-08 | -- | 50mA |
| AIR-04 | -- | 60mA |
| AIR-08 | -- | 80mA |
| HSC-02 | -- | -- |
| HSP-04 | -- | 100mA |
| CAN-1M | -- | 100mA |

# D　编程卡（U 盘）使用说明

CTH300-H 系列 CPU 支持普通 U 盘作为编程卡，免除了用户额外采购特殊编程卡的负担，能够方便地保存程序及数据，提高工作效率。

为了保护用户数据安全，我们不但对编程卡中的数据文件进行了加密，还通过特别的措施保证：数据文件即使被复制至其它 U 盘中，也无法正常使用。另外，用户可以设置限制上载次数来保证自有程序和数据不会被无限制地复制。将 U 盘作为编程卡不会影响 U 盘内的其它文件，可以放心使用。

**<注意>** 目前 300CPU 只支持 FAT32 文件系统的 U 盘，支持热插拔。如果在读写 U 盘数据的过程中拔出 U 盘，会导致相关功能异常，例如下载程序至 CPU 的过程通信超时等。因此，使用时应该谨慎操作。

编程卡（U 盘）可以进行下载、上载、设置、清除等操作，以下各节将分别说明各个操作步骤。

表 D-1 编程卡相关特殊存储器（SM）

| SM 状态位 | 功能描述 |
|---|---|
| SMB100 | 编程卡限制上载次数，只能读，写无意义 |
| SMB101 | 编程卡剩余上载次数，只能读，写无意义 |
| SM7.7 | 0：存储卡（U 盘）存在且工作正常<br>1：存储卡（U 盘）不存在或工作异常 |

**编程卡文件格式**

PLC 的相关数据下载至编程卡（U 盘）后，以文件形式保存在 U 盘中，每类数据文件的名称都是固定的，后缀名为.CTH 或.DTH。

**编程卡热插拔**

U 盘支持热插拔功能，为保证插拔过程不影响 PLC 程序的正常执行，需要遵循以下规定：

1）CPU 处于 RUN 状态时，不检测插入的 U 盘。必须先将系统运行开关拨至 STOP 模式，或通过命令使 CPU 停机，方可检测并识别 U 盘；

2）CPU 在读写 U 盘时，STOP 灯会闪烁，灯闪烁时拔出 U 盘有可能导致异常甚至 U 盘损坏；

3）插入 U 盘后，在 CPU 识别到 U 盘前勿让 CPU 进入 RUN 状态，正确识别到 U 盘的标志是 SM7.7 被清 0。从插入 U 盘到 SM7.7 被清 0，这个过程可能需要 5～10 秒钟。

**编程卡与 CPU 开关状态**

如果编程卡（U 盘）在 CPU 上电时被插在 USB 接口上，CPU 将根据当前开关状态进入相应的工作模式：

1）如果开关处于 RUN 状态，进入正常工作模式；

2）如果开关处于 STOP 状态，进入编程卡上载模式，如果此时编程卡中的程序块、初始化 V 内存数据块、硬件组态块（保存在后缀名为.CTH 的文件中）三者都不存在，CPU 仍然会切换至正常工作模式。

## D.1 编程卡下载

编程卡的下载功能：让用户把 MagicWorks PLC 工程中的程序块、保密库 1/2、硬件组态块、初始化 V 内存数据块、配方和数据记录下载到编程卡中。

下载操作步骤如下：

**步骤 1：**下载任意块至 CPU，CPU 会自动将下载的块写入到 U 盘。写入过程中，STOP 灯会闪烁，写入完毕后，STOP 灯常亮。

**步骤 2：**如果要限制使用次数，确保编程卡中系统块、数据块、程序块都已经下载完毕后，选择 MagicWorks PLC 的菜单项“PLC”→“编程卡配置”写入限制次数值。如果不需要覆盖程序块和数据块，可把“块的覆盖选择”下对应的块勾选去掉（默认覆盖）。点击“确定”，设置成功后会提示“编程卡设置成功”，否则会提示相应错误信息。

**步骤 3：**当写入完毕，为保险起见，STOP 灯从闪烁变常亮后，须等待 3 秒钟再拔出编程卡。



**提示**

1）下载操作前需确认编程卡（U 盘）已经插在 USB 接口上，且已被正确识别（SM7.7 = 0）

2）U 盘可用空间必须大于 64MB 才能成功下载。

3）下载中包含有硬件组态块、程序块（保密库属于程序块）、初始化 V 内存数据块中的任意一块，编程卡中已存在的程序块、初始化 V 内存数据块、硬件组态块都会被覆盖。

4）如果编程卡设置了上载限制次数，再次下载任意一个块到 CPU，都会导致编程卡的上载次数限制失效。

5）下载过程中，若指示灯处于闪烁状态，请勿断电，否则可能导致编程卡的内容丢失。

6）若在未断电的情况下拔掉编程卡，CPU 被再次上电前将可能无法正常工作。

表 D-2 STOP 灯状态

| 指示灯 | 灭 | 闪烁 | 亮 |
|---|---|---|---|
| STOP 灯-橙色 | -- | 传送数据中 | 正常 |

## D.2 编程卡上载

编程卡的上载功能：用户可以将编程卡（U 盘）中的所有块上载到 CPU，并更新 CPU 的 Flash 存储器。

上载操作步骤如下：

**步骤 1：**CPU 断电状态下插入编程卡（U 盘）到 USB 接口。

**步骤 2：**把拨码开关置于“STOP”状态，然后给 PLC 上电。

**步骤 3：**如果编程卡（U 盘）被识别且编程卡中存在 002.CTH 或 003.CTH 或 004.CTH，系统将进入“编程卡上载模式”，否则进入正常运行模式。

**步骤 4：**CPU 检测到编程卡（U 盘），则清除 CPU Flash 存储器中存在的所有块，并用编程卡中的程序块、数据块、硬件组态块等覆盖。如果用户在编程卡中配置了覆盖标志，标志为“不覆盖”的块将不会上载，即保留原来的块内容。只有系统块、数据块、程序块都存在的情况下，才会上载成功。

**步骤 5：**RUN 绿灯在上载过程中会闪烁，等绿灯从闪烁变常亮后超过 3 秒钟，上载完成，CPU 将死机等待。此时用户才可以断电。如果上载失败，SF 灯（红色）会闪烁，直至断电。

**步骤 6：**重新给 PLC 加电，CPU 将执行从编程卡上载的程序和数据。

**<注意>** 如果 USB 接口插有 U 盘，则上电前必须把 CPU 开关拨至“STOP”状态，否则将进入编程卡上载模式，不能正常工作。

以下情况可能会导致上载失败：

1)编程卡使用次数已到限额不影响 CPU 中的原有的数据；

2)编程卡中信息校验失败；

3）CPU 被密码保护；



**提示**

上载过程中，灯闪的时候，不能断电，否则可能会导致编程卡的内容丢失。

上载完成后 CPU 必须重新上电，才能执行上载的程序（请正确设置拨码开关或拔掉 U 盘）。

表 D-3 RUN 灯状态

| 指示灯 | 灭 | 闪烁 | 亮 |
|---|---|---|---|
| RUN 灯-绿色 | -- | 上载进行中 | 上载完成 |
| SF 灯 | 正常 | 出错 | -- |

# E CT_Modbus 库的使用介绍

## E.1 CT_Modbus 库介绍

CT_Modbus 功能块主要针对 Modbus 功能块占用 CPU 大量程序空间和数据空间而提供的一组内嵌的简单易用的 Modbus 协议库。CT_Modbus 功能块是集成在 CPU 内部，不占用用户数据空间，作为一组库函数提供给用户使用。由以下库文件组成：

- **CT_Modbus_RTU 库**

CT_Modbus_RTU 库文件共包含 4 个库，分别是 PORT0 的主从站库和 PORT1 的主从站库。

**<备注>** H35-00 的 PORT0 和 PORT1 支持 CT_Modbus_RTU 的四个库文件（ct_mbus_master、ct_mbus_master_port1、ct_mbus_slave、ct_mbus_slave_port1），H36-00 的 PORT0 支持 CT_Modbus_RTU 的两个库文件（ct_mbus_master 和 ct_mbus_slave）。

- **CT_Modbus_TCP 库**

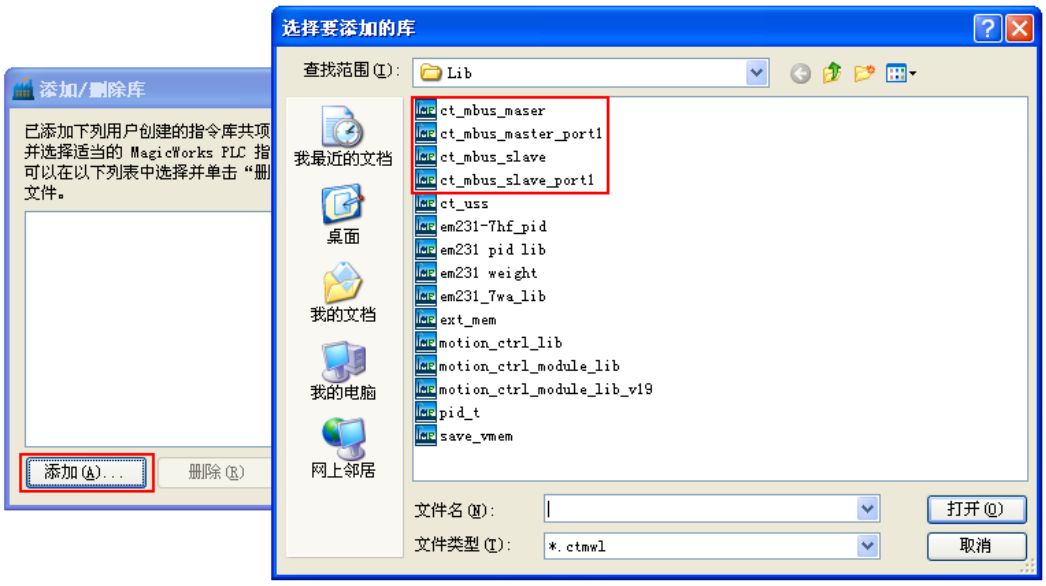
CT_Modbus_TCP 包含 1 个库 CT_Mbus_master_tcp，即为 EtherNET 通信口的 Modbus TCP 主站库。

**<备注>** 您可从合信官网免费下载 CT_Modbus 库：http://www.co-trust.com。

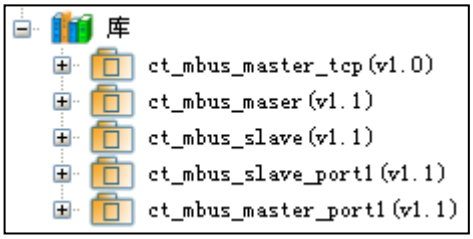
## E.2 安装 CT_Modbus 库文件

**添加库文件**

选择 MagicWorks PLC 菜单项“文件”→“添加/删除库”，在弹出的“添加/删除库”对话框中点击“添加”按钮，找到库文件“ct_mbus_master”、“ct_mbus_master_port1”、“ct_mbus_slave”、“ct_mbus_slave_port1”、“ct_mbus_master_tcp”，如下图所示，选中需要添加的库文件，单击“打开”按钮即可。




安装成功后，在主程序指令树的“库”下方可以看到新增加的库：




**调用 CT_Modbus 库**

单击需要添加功能块的“网络”，双击“库”下方的“MBUS_INIT”，“MBUS_SLAVE”，“MBUS_CTRL”，“MBUS_MASTER”，“Modbus TCP_EXE”就会在“网络”里出现相应的功能块。

## E.3 CT_Modbus_RTU 库功能说明

### Modbus 地址

Modbus 地址通常是包含数据类型和偏移量的 5 个或 6 个 xin 字符值。第一个或前两个字符决定数据类型，最后的四个字符是符合数据类型的一个适当的值。Modbus 主站则将这个地址对应到正确的功能上。

Modbus 从站指令支持以下地址：00001 至 02048 是实际输出，对应于 Q0.0~Q255.7；10001 至 12048 是实际输入，对应于 I0.0~I255.7；30001 至 30512 是模拟输入寄存器，对应于 AIW0 至 AIW1022；40001 至 4XXXX 是保持寄存器，对应于 V 内存区。

所有 Modbus 地址都是从 1 开始编号的。下表所示为 Modbus 地址与 CPU 地址的对应关系。Modbus 从站协议允许您对 Modbus 主站可访问的输入、输出、模拟输入和保持寄存器（V 区）的数量进行限定。

表 E-1　Modbus 地址与 CPU 地址对应关系

| Modbus地址 | CPU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 | HoldStart |
| 040002 | HoldStart+2 |
| 040003 | HoldStart+4 |
| …… | …… |
| 04xxxx | HoldStart+2 x (xxxx-1) |

## 使用 Modbus RTU 从站协议指令

**※ CT_Modbus 从站协议指令占用 CPU 的资源**

1）根据使用不同的 Modbus 协议库占用 PORT0 或 PORT1 作为 Modbus 从站协议通讯。当 PORT 0 或者 PORT 1 作为 Modbus 协议通讯时，它不能再作为其它任何目的使用，包括与 MagicWorks PLC 通讯、自由口通讯。MBUS_INIT 指令控制 Port 的设定是 Modbus 协议还是 PPI。

2）与选用 Port 自由口通讯相关的所有的 SM。

3）占用 92 个字节程序空间。

**※ 在 CPU 程序中使用 Modbus 从站协议指令遵循的步骤**

1）在您的程序中插入 MBUS_INIT 指令并且只在一个循环周期中执行该指令，MBUS_INIT 指令可用于对 Modbus 通讯参数的初始化或修改。当您插入 MBUS_INIT 指令时，几个隐藏的子程序和中断服务程序会自动地添加到您的程序中。

2）在您的程序中只使用一个 MBUS_SLAVE 指令。该指令在每个循环周期中执行，为接收到的所有请求提供服务。

3）用通讯电缆将 CPU 通讯口与 Modbus 主站连接起来。

**※ Modbus 从站协议指令所支持的功能**

Modbus 从站协议指令支持 Modbus RTU 协议。这些指令使用 CPU 的自由口功能，支持大部分常用 Modbus 功能。以下是所支持的 Modbus 功能：

表 E-2　Modbus 功能

| 功能 | 描述 |
|---|---|
| 1 | 读单个/多个线圈（实际输出）状态。功能1返回任意数量输出点的接通/断开状态（Q） |
| 2 | 读单个/多个接触器（实际输入）状态。功能2返回任意数量的输入点的接通/断开状态（I） |
| 3 | 读单个/多个保持寄存器。功能3返回V存储器的内容，保持寄存器在Modbus下是字类型，在一个请求中最多可读120个字。 |
| 4 | 读单个/多个输入寄存器。功能4返回模拟输入值。 |
| 5 | 写单个线圈（实际输出）。功能5将实际输出点设置为指定值。该输出点不是被强制，用户程序可以重写由Modbus的请求而写入的值。 |
| 6 | 写单个保持寄存器。功能6写一个单个保持寄存器的值到CPU的V存储区。 |
| 15 | 写多个线圈（实际输出）。功能15写多个实际输出值到CPU的Q映象区。起始输出点必须是一个字节的开始（如，Q0.0或Q2.0），并且要写的输出的数量是8的倍数。这是Modbus从站协议指令的限定。这些点不是被强制，用户程序可以重写由Modbus的请求而写入的值。 |
| 16 | 写多个保持寄存器。功能16写多个保存寄存器到CPU的V区。在一个请求中最多可写120字。 |

**※ MBUS_INIT 指令**

MBUS_INIT 指令用于使能和初始化或禁止 Modbus 通讯。MBUS_INIT 指令必须无错误的执行，然后才能够使用 MBUS_SLAVE 指令。在继续执行下一条指令前，MBUS_INIT 指令必须执行完并且 Done 位被立即置位。MBUS_INIT 指令应该在每次通讯状态改变时只执行一次。因此，EN 输入端应使用边沿检测元素以脉冲触发，或只在第一个循环周期内执行一次。

表 E-4 MBUS_INT 指令参数说明

| 参数地址 | 说明 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|
| Mode | 选择通讯协议。1：将Port定义为Modbus协议并使能该协议；0：将Port定义为PPI并禁止Modbus协议。 | 位 | | |
| Addr | 设置本站地址。 | 字节 | 1~247 | |
| Baud | 设置波特率。 | 双字 | 1200、2400、4800、9600、19200、38400、57600、115200 | |
| Parity | 设置校验。 | 字节 | 0--无校验<br>1--奇校验<br>2--偶校验 | 所有设置使用一个停止位。 |

| | | | | |
|---|---|---|---|---|
| Delay | 通过为标准Modbus信息超时增加指定数量的毫秒，扩展标准Modbus信息结束超时条件。 | 整型 | 0~32767 | 单位：毫秒 |
| MaxIQ | 设置可使用的I和Q点数。<br>0：禁止对输入和输出的读写。 | 整型 | 0~2048 | 建议MaxIQ的取值为2048，即允许访问所有I点和Q点。 |
| MaxAI | 设置可使用的字输入寄存器（AI）的个数。<br>0：禁止读模拟输入。 | 整型 | 0~512 | MaxAI建议值为512 |
| MaxHold | 设置可以使用的V存储区字保持寄存器的个数。 | 整型 | 0~32767 | 单位：字 |
| HoldStart | 设置可以使用的V存储区的保持寄存器的起始地址。 | 双字 | 指向V存储区的指针。 | |
| Done | 当MBUS_INIT指令完成时，Done输出接通。 | 位 | | |
| Error | Error输出字节包含该指令的执行结果。 | 字节 | | |

**※ MBUS_SLAVE 指令**

MBUS_SLAVE 指令用于服务来自 Modbus 主站的请求，必须在每个循环周期都执行，以便检查和响应 Modbus 请求。当 EN 输入接通时，该指令在每一循环周期内执行。MBUS_SLAVE 指令无输入参数。

表 E-5 MBUS_SLAVE 指令参数说明：

| 参数地址 | 说明 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|
| Done | 当MBUS_SLAVE指令响应Modbus请求时Done输出接通。如果没有服务的请求，Done输出会断开。 | 位 | | |
| Error | 输出包含该指令的执行结果。 | 字节 | 错误代码见下表 | 该输出只有Done接通时才有效。如果Done断开，错误码不会改变。 |

表 E-6　错误代码

| 错误代码 | 描述 |
|---|---|
| 0 | 无错误 |
| 1 | 存储区范围错误 |
| 2 | 非法波特率或校验 |
| 3 | 非法从站地址 |
| 4 | Modbus参数的非法值 |
| 5 | 保持寄存器与Modbus从站符号地址重叠 |
| 6 | 接收校验错误 |
| 7 | 接受CRC错误 |
| 8 | 非法功能请求/不支持的功能 |

| | |
|---|---|
| 9 | 请求中有非法存储区地址 |
| 10 | 从站功能未使能 |

**※ Modbus 从站协议指令使用实例**

使用梯形图编程：建立一个从站地址为 1，波特率为 115200，无校验的 Modbus 从站：

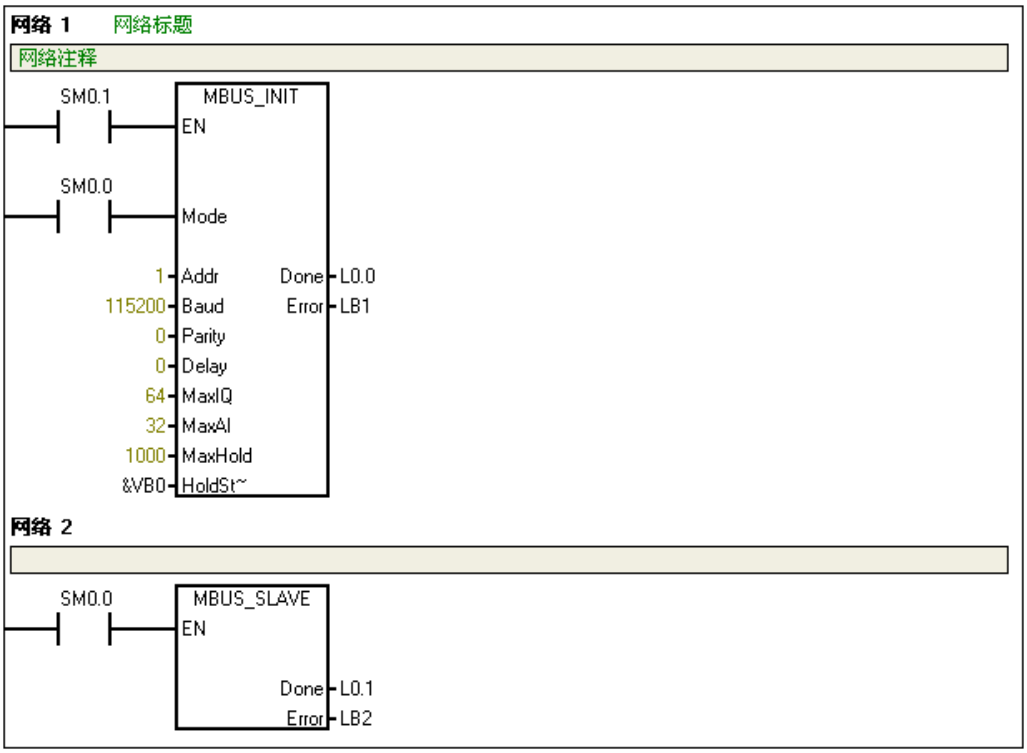



表 E-7　MBUS_INIT 参数配置说明

| | |
|---|---|
| Addr | 设置从站地址为 1 |
| Boud | 设置通信波特率为115200 |
| Party | 设置奇偶校验为无校验 |
| Delay | 设置延迟时间为0毫秒 |
| MaxIQ | 设置最大可使用256个I点和256个Q点（即000001~002048 和 010001~012048） |
| MaxAI | 设置最大可读512路模拟量输入（即030001-030512） |
| MaxHold | 设置最大可使用的V区字保存寄存器的个数（以字为单位）。 |
| StartHold | Modbus主站可以访问CPU的V区保存寄存器的起始地址（如&VB0）。 |

## 使用 Modbus RTU 主站协议指令

**※ Modbus 主站协议指令占用 CPU 的资源**

1）根据使用不同的 Modbus 协议库占用 PORT0 或者 PORT1 作为 Modbus 从站协议通讯。当 PORT0 或者 PORT1 作为 Modbus 协议通讯时，它不能再作为其它任何目的使用，包括与 MagicWorks PLC 通讯、自由口通讯。MBUS_INIT 指令控制 PORT 的设定是 Modbus 协议还是 PPI。

2）与选用 Port 自由口通讯相关的所有的 SM。

3）占用 119 个字节程序空间。

**※ MBUS_CTRL 指令**

使用 SM0.0 调用 MBUS_CTRL 指令完成主站的初始化，并启动其功能控制。

表 E-8 MBUS_CTRL 指令参数说明

| 参数地址 | 说明 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|
| Mode | 设置通讯模式。为 1 时，使能 Modbus 协议功能；为 0 时恢复系统为 PPI 协议。 | 位 | | |
| Baud | 设置波特率。 | 双字 | 11200，2400，4800，9600，19200，38400，57600，115200 | |
| Parity | 设置校验。 | 字节 | 0--无校验<br>1--奇校验<br>2--偶校验 | 所有设置使用一个停止位。 |
| Timeout | 主站等待从站响应的时间，以毫秒为单位。 | 整型 | 1 ~ 32767 | 典型的设置值为 1000 毫秒（1 秒）。 |
| Done | 完成位，初始化完成，此位会自动置 1。 | 位 | | |
| Error | 初始化错误代码。 | 字节 | 0--无错误<br>1--校验选择非法<br>2--波特率选择非法<br>3--模式选择非法 | 只有在 Done 位为1时有效。 |

**※ MBUS_MSG 指令**

使用 SM0.0 调用 Modbus RTU 主站读写子程序 MBUS_MSG 指令，First 接通发送一个 Modbus 请求。同一时刻只能有一个读写功能（即 MBUS_MSG）使能。

表 E-9 MBUS_MSG 指令参数说明

| 参数地址 | 说明 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|
| First | 读写请求位 | 位 | | 每一个新的读写请求必须使用脉冲触发。 |
| Slave | 设置从站地址 | 字节 | 1~247 | |
| RW | 读写操作命令 | 字节 | 0--读<br>1--写 | |
| Addr | 选择读写的数据类型 | 双字 | 00000 至 0xxxx--开关量输出<br>10000 至 1xxxx--开关量输入<br>30000 至 3xxxx--模拟量输入<br>40000 至 4xxxx--保持寄存器 | |
| Count | 通讯的数据个数（位或字的 | 整型 | | Modbus 主站每一个 |

| | | | | |
|---|---|---|---|---|
| | 个数） | | | MBUS_MSG 指令可读/写的最大数据量为 120 个字。 |
| DataPtr | 数据指针，如果是读指令，读回的数据放到这个数据区中；如果是写指令，要写出的数据放到这个数据区中。 | 双字 | | |
| Done | 完成位，读写功能完成位 | 位 | | |
| Error | 错误代码 | | 错误代码见下表 | 只有在 Done 位为 1 时，错误代码才有效。 |

表 E-10 错误码

| 错误代码 | 描述 |
|---|---|
| 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 |
| 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 | 从站接收了信息，但是响应被延迟 |
| 106 | 从站忙，拒绝了该信息 |
| 107 | 从站拒绝了信息 |
| 108 | 从站存储器奇偶错误 |

## E.4 CT_Modbus_master_tcp_single 库功能说明

CT_Modbus_master_tcp_single 指令为 Modbus TCP 主站单条读写指令

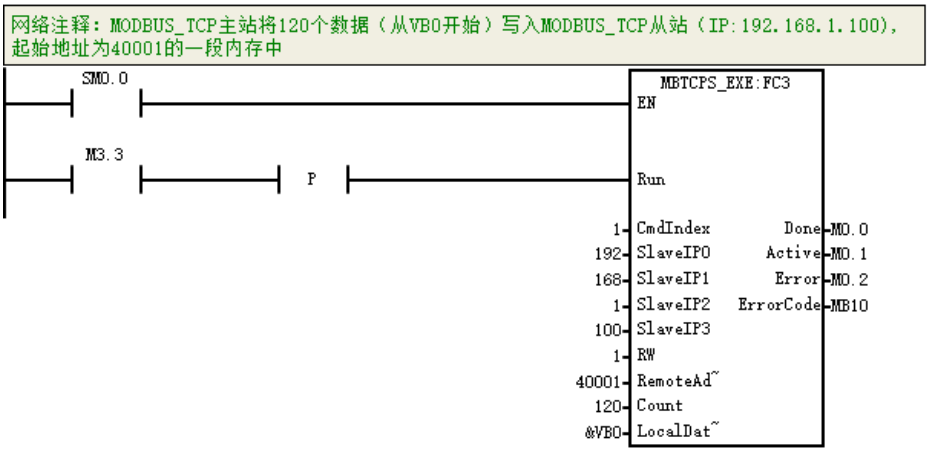



表 E-11 指令参数说明

| 参数地址 | 说明 | 类型 | 备注 |
|---|---|---|---|
| Run | 通信启动位 | BOOL | 请使用沿触发 |
| CmdIndex | 调用 MBTCPS_EXE 的序号，每处调用序号都不能重复 | BYTE | 范围为 1～255 |
| Slave IP0 | 从站 IP 地址的第一位 | BYTE | 假设从站 IP 地址为 192.168.1.201，则 Slave IP0~Slave IP3 应该分别设置为：192/168/1/201 |
| Slave IP1 | 从站 IP 地址的第二位 | BYTE | |
| Slave IP2 | 从站 IP 地址的第三位 | BYTE | |
| Slave IP3 | 从站 IP 地址的第四位 | BYTE | |
| RW | 读写：Read = 0，Write = 1 | BYTE | |
| Count | 读写单元数，有效范围：1~120 字或 1~1920 位 | WORD | |
| RemoteAddr | 远程数据地址（如 40001） | DWORD | |
| LocalDataPtr | 本地数据指针（如 &VB1000） | WORD | |
| Done | 完成标识位：0 = 未完成，1 =已完成 | BOOL | |
| Active | 激活位，0 = 未激活或已完成，1 = 操作已激活且未完成 | BOOL | |
| Error | 错误位，0 = 无错误， 1 = 有错误 | BOOL | |
| ErrCode | 错误代码，完成位为 1 时有效(见下表) | BYTE | |

表 E-12 错误代码表

| 错误代码 | 描述 |
|---|---|
| 0 | 无错误 |
| 1 | 连接的总数达到了允许的最大连接数 |
| 2 | 正在建立连接 |
| 3 | 操作超时 |
| 4 | 请求参数错误 |
| 5 | 指令未被激活 |
| 6 | 连接忙（如同时激活多条 MBTCPS_EXE） |

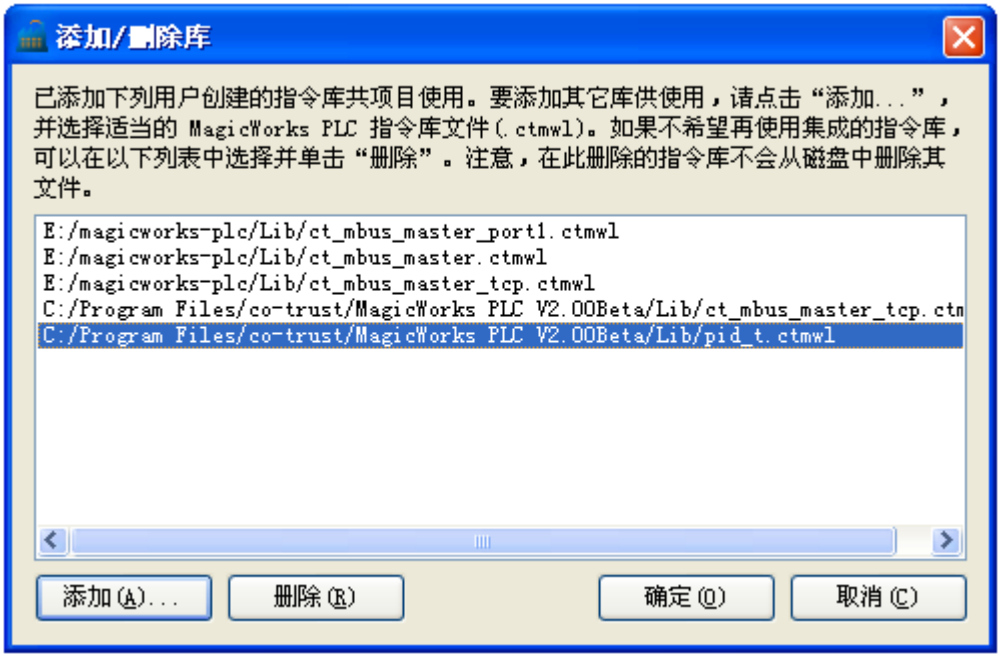
# F PID_T 库的使用介绍

PID_T 功能块是集成在主控模块内部，不占用用户数据空间，作为一个库函数提供给用户使用。PID_T 主要针对温度控制的智能 PID 功能，带有自整定、自适功能，用户无需复杂编程，只需调用和设置一些简单的参数就可以使用，温度控制准确。

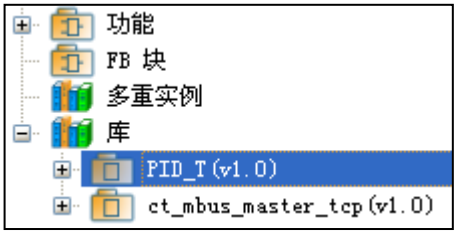
您可以从合信官网免费下载该库。下载地址：http://www.co-trust.com/Download/index.html

**安装说明**

选择 MagicWorks PLC 菜单项“文件”→“添加/删除库”打开如下对话框，然后点击“添加”按钮，在你存放的 pid_t.ctmwl 的位置找到此文件，点击“确定”即添加成功。



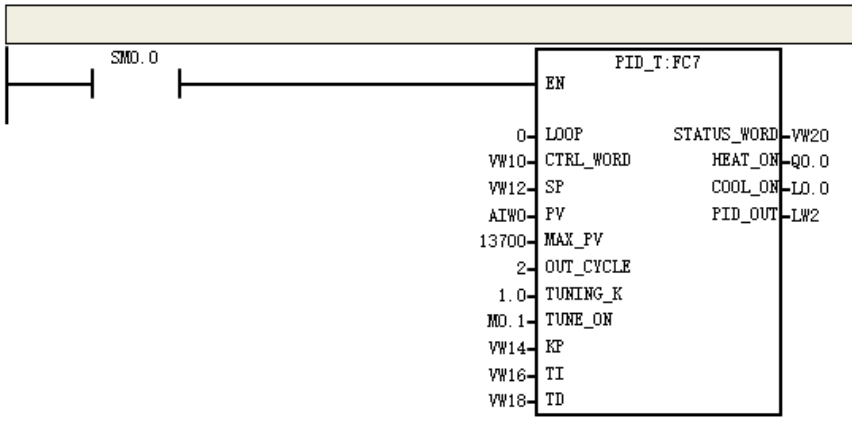

安装成功后，即可在目录树的“库”子目录下看到新增加的 PID_T 库：




**调用 PID_T 库**

点击要添加功能块的“网络”，双击“库”下面的“PID_T”，就会在“网络”里出现相应的功能块，如下图所示：

网络 7




**PID_T 库功能说明**

表 F-1 指令参数说明

| 参数地址 | 说明 | 类型 | 数值范围 | 备注 |
|---|---|---|---|---|
| LOOP | 所属回路PID的编号，不能重复从0开始 | 字，常数或变量 | 0-127 | 表示该控制回路ID。 |
| CTRL_WORD | 控制字<br>控制PID运行 | 字，常数或变量 | | 常用控制字：<br>1）16#03(只有加热输出，带自适应功能)<br>2）16#07(加热冷却输出，带自适应功能) |
| SP | 设定值 | 字，常数或变量 | -32768-32767 | 单位：0.1℃ |
| PV | 测量值（反馈值） | 字，变量 | -32768-32767 | 单位：0.1℃ |
| MAX_PV | 测量值的最大值 | 字，常数或变量 | -32768-32767 | 单位：0.1℃ |
| OUT_CYCLE | 脉冲输出周期 | 字，常数或变量 | 1-255 | 单位：秒 |
| TUNING_K | 自整定系数 | 双字，浮点数 | 0.5-2.0 | 0.5：要求系统控制超调量小。<br>1.0：正常响应。<br>2.0：要求系统控制响应快，超调量大。 |
| TUNING_ON | 启动自整定 | 位，变量 | | 自整定结束后自动复位。 |
| Kp | 比例系数 | 字，整数，常数或变量 | | 如果为常数，不能执行自整定功能。 |
| Ti | 积分时间 | 字，整数，常数或变量 | 1-3600 | 单位：秒<br>如果为常数，不能执行自整定功能。 |
| Td | 微分时间 | 字，整数，常数或变量 | 0-3600 | 单位：秒<br>如果为常数，不能执行自整定功能。 |
| STATUS_WORD | 状态字 | 字，变量 | | 运行状态及报警状态。 |

| | | | | |
|---|---|---|---|---|
| HEAT_ON | 加热输出 | 位 | | |
| COOL_ON | 冷却输出 | 位 | | |
| PID_OUT | PID模拟输出 | 字，整数，变量 | 只有加热输出时，输出范围0-32000。带冷却输出时：-32000-32000 | |

表 F-2 控制字位地址定义

| 控制字位 | 设置 | 备注 |
|---|---|---|
| 0 | 0 | PID 停止 |
| | 1 | PID 运行 |
| 1 | 0 | 积分一直起作用，比例系数 Kp 不自动调整 |
| | 1 | 积分分离及比例系数自动调整 |
| 2 | 0 | PID 单极输出 |
| | 1 | PID 双极输出 |
| 3 | 0 | 保留 |
| | 1 | 保留 |
| 4 | 0 | 积分起作用 |
| | 1 | 积分不起作用 |
| 5 | 0 | 微分起作用 |
| | 1 | 微分不起作用 |
| 6 | | 保留 |
| 7 | | 保留 |

表 F-2 状态字位地址定义

| 状态字位 | 值 | 备注 |
|---|---|---|
| 0 | 0 | 无断线故障 |
| | 1 | 断线故障 |
| 1 | 0 | 自整定未进行 |
| | 1 | 正在自整定 |
| 2 | 0 | 无自整定故障 |
| | 1 | 自整定故障 |
| 3 | 0 | 不加热 |
| | 1 | 正在加热 |
| 4 | 0 | 不冷却 |
| | 1 | 正在冷却 |
| 5 | 0 | PID 停止状态 |
| | 1 | PID 运行状态 |
| 6 | | 保留 |

| | | |
|---|---|---|
| 7 | | 保留 |

# G 以太网“ct_socket”通信库的使用介绍

## G.1 ct_socket 库介绍

目前 CPU 支持 2 个 UDP 连接和 2 个 TCP 客户端连接，提供的指令 SOCK_Open、SOCK_Send、SOCK_Recv、SOCK_Close，共 4 条。

| 约定项 | 范围 | 说明 |
|---|---|---|
| 连接号（SockID） | 0~255 | 255 为无效连接号 |
| 连接类型（SockType） | 0，1 | 0为UDP连接，1为TCP客户端连接 |
| 端口号（Lport/Rport） | 1~65535 | 本地端口号要选用不被其他连接占用的端口号，如系统块里默认端口号20000为PLC监控连接使用。TCP客户端的本地端口号由底层分配，Lport可设置为任何值（UDP模式下不能设为65535），Rport需与服务器端口号一致。 |

**套接字错误码约定：**

| 错误码 | 说明 |
|---|---|
| 0 | 无错误 |
| 1 | 连接已关闭 |
| 2 | 端口号错误。常用，本地端口号被占用，或端口号超出范围报错 |
| 3 | 设备断线 |
| 4 | UDP 建立连接失败。常用，创建连接失败，返回连接号 255 |
| 5 | TCP 客户端连接远方服务端失败 |
| 6 | TCP 服务端监听失败 |
| 7 | 数据长度错误，最大数据长度为 512 字节 |
| 8 | 数据区为空错误 |
| 9 | 连接号错误 |
| 10 | 接收数据错误（没接收到数据） |
| 11 | 连接类型错误 |
| 12 | IP 信息错误 |
| 13 | 超时错误 |
| 14 | TCP 客户端需要重连标志。常用，TCP 重连 |

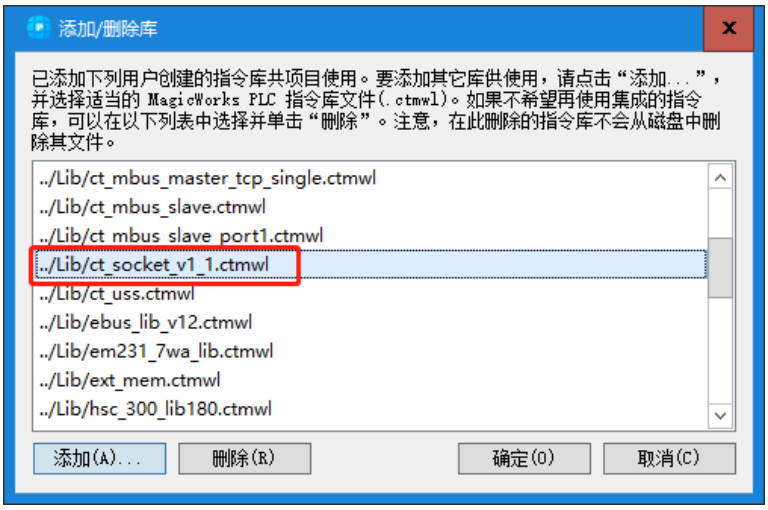
## G.2 使用说明

【添加库文件】

在项目管理器界面或编程界面选择“文件”--“添加/删除库”，如下图所示：



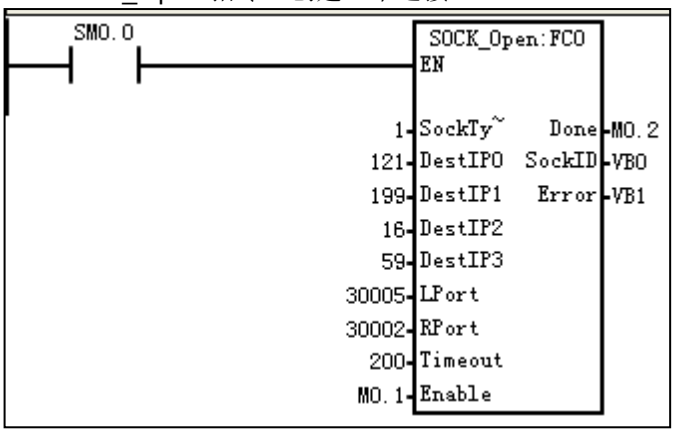

找到“ct_socket_v1_1.ctmwl”如下图所示，点击“确定”按钮。




安装成功后，在程序块界面目录树的“库”下可以看到新增加的“ct_socket（v1.1）”：

【指令功能说明】

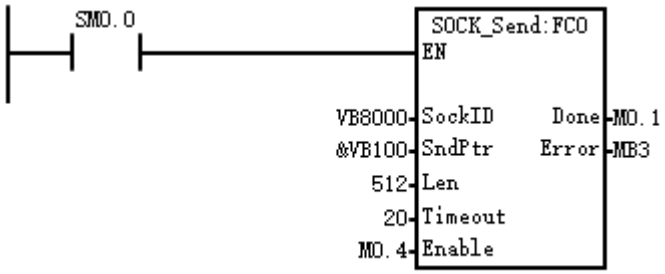
1）SOCK_Open 指令：创建一个连接




| 参数地址 | 说明 | 变量类型 | 数据类型 | 备注 |
|---|---|---|---|---|
| SockTy | 链接类型，UDP(0)和 TCP-CLIENT(1) | IN | BYTE | 不支持 TCP 服务器连接。 |
| DestIP0 | 目标IP地址的第1个字节。 | IN | BYTE | 例如目标地址为192.168.1.202，第一个字节为192。 |
| DestIP1 | 目标IP地址的第2 | IN | BYTE | 例如目标地址为 |

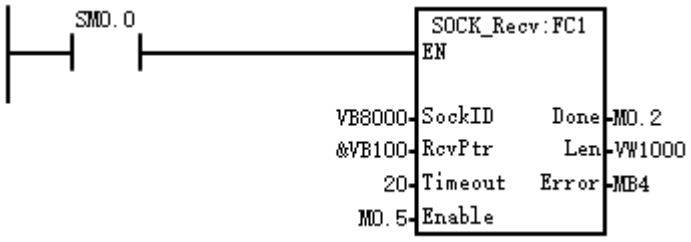
| | | | | |
|---|---|---|---|---|
| | 个字节。 | | | 192.168.1.202，第一个字节为168。 |
| DestIP2 | 目标IP地址的第3个字节。 | IN | BYTE | 例如目标地址为192.168.1.202，第一个字节为1。 |
| DestIP3 | 目标IP地址的第4个字节。 | IN | BYTE | 例如目标地址为192.168.1.202，第一个字节为202。 |
| Lport | 本地端口，UDP使用 | IN | WORD | TCP-CLIENT模式，LPort由底层随机分配 |
| Rport | 目标端口 | IN | WORD | 与远程服务器端口号一致。 |
| Timeout | 超时时间 | IN | BYTE | 单位：100ms。 |
| Enable | 使能位 | I/O | BOOL | |
| Done | 读写功能完成位 | OUT | BOOL | 使能后自动清零。 |
| SockID | 输出连接号 | OUT | BYTE | 底层分配，应用要提供一个全局内存保存连接号。 |
| Error | 错误代码 | OUT | BYTE | 错误码。0：无错 |

2）SOCK_Send 指令：创建一个连接



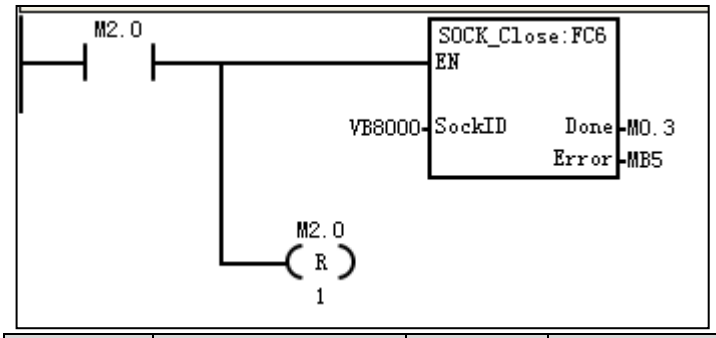

| 参数地址 | 说明 | 变量类型 | 数据类型 | 备注 |
|---|---|---|---|---|
| SockID | 输出连接号 | OUT | BYTE | 底层分配，应用要提供一个全局内存保存连接号。 |
| SndPtr | 数据缓冲区 | IN | DWORD | 指向待发送数据的指针，可以指向I、Q、M或V存储器的指针（如&VB100） |
| Len | 待发送字节数 | IN | WORD | 范围为1~512字节。 |
| Timeout | 超时时间 | IN | BYTE | 单位：100ms。 |
| Enable | 使能位 | I/O | BOOL | 使能 |
| Done | 读写功能完成位 | OUT | BOOL | 使能后自动清零。 |
| Error | 错误代码 | OUT | BYTE | 错误码。0：无错 |

3）SOCK_Recv 指令：接受数据




| 参数地址 | 说明 | 变量类型 | 数据类型 | 备注 |
|---|---|---|---|---|
| SockID | 输出连接号 | OUT | BYTE | 底层分配，应用要提供一个全局内存保存连接号。 |
| RcvPtr | 数据缓冲区 | IN | DWORD | 指向接收数据存储位置的指针，可以指向I、Q、M或V存储器的指针（如&VB100） |
| Timeout | 超时时间 | IN | BYTE | 单位：100ms。 |
| Enable | 使能位 | I/O | BOOL | 使能 |
| Done | 读写功能完成位 | OUT | BOOL | 使能后自动清零。 |
| Len | 实际接收字节数 | IN | WORD | 范围1~512字节。 |
| Rport | 目标端口 | IN | WORD | 与远程服务器端口号一致。 |
| Error | 错误代码 | OUT | BYTE | 错误码。0：无错 |

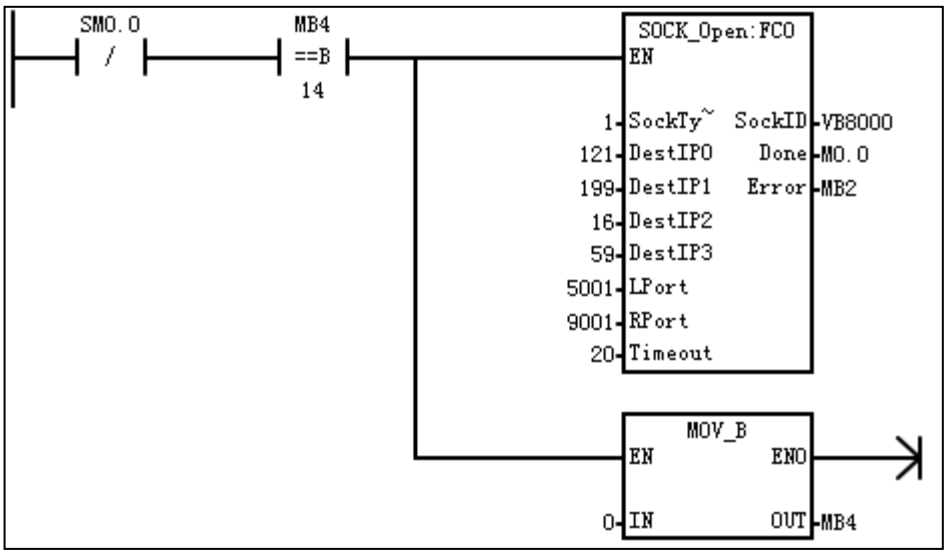
4）SOCK_Close 指令：关闭连接




| 参数地址 | 说明 | 变量类型 | 数据类型 | 备注 |
|---|---|---|---|---|
| SockID | 输出连接号 | OUT | BYTE | 底层分配，应用要提供一个全局内存保存连接号。 |
| Done | 读写功能完成位 | OUT | BOOL | 使能后自动清零。 |
| Error | 错误代码 | OUT | BYTE | 错误码。0：无错 |

【应用实例】

TCP 客户端掉线重连，通过发送出错标志来确定是否重新打开连接：重连标志 14

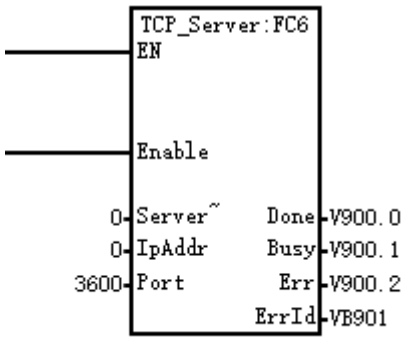



# H CT_tcp_server_lib 库的使用介绍

CT_tcp_server_lib 库文件共包含 4 个指令，分别是：TCP_server、TCP_connect、TCP_send、TCP_recv。

**<备注>** 您可从合信官网免费下载 CT_tcp_server_lib 库：http://www.co-trust.com。

## H.1 TCP_Server




**1.功能**

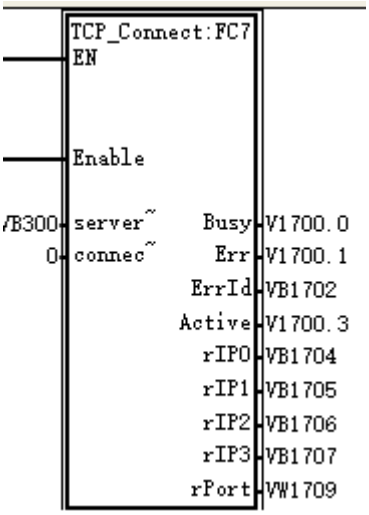
此指令用于启动 Tcp Server。

**2.参数**

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Enable | IN | 指令执行条件 | BOOL | 当Enable为1时启动Tcp server<br>Enable为0时停止Tcp server |
| ServerId | IN | Tcp Server Id号 | BYTE | 提供2个TCP Server，Service Id为（0，1） |

| | | | | |
|---|---|---|---|---|
| IpAddr | IN | 远程连接地址 | BYTE | 暂时只能取0 |
| Port | IN | Tcp Serve监听的端口号 | WORD | |
| Done | OUT | 指令完成位 | BOOL | 当指令完成或出错时Done置位。<br>当指令的执行条件Off时，Done位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令的执行条件Off时，Busy位被复位。 |
| Err | OUT | 指令错误位 | BOOL | 当指令出错时。Err位置位。<br>当指令的执行条件Off 时，Err位被复位。 |
| ErrId | OUT | 错误代码 | BYTE | 0. 无错<br>1. ServerId 错误<br>2. IpAddr 错误<br>3. 端口号被占用<br>当指令的执行条件Off 时，ErrId清0。 |

## H.2 TCP_Connect




**1. 功能**

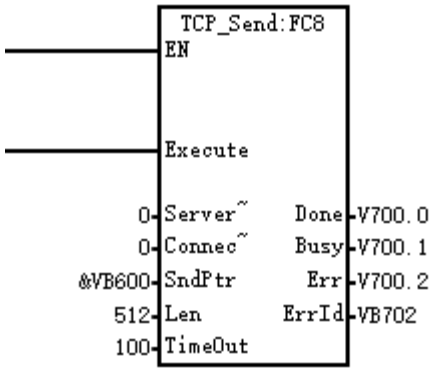
此指令用于得到 TCP 连接的状态。

**2. 参数**

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Enable | IN | 连接使能位 | BOOL | 当Enable为1时得到连接的状态。<br>当Enable下降沿，且处于连接状态，则断开连接。 |
| ServerId | IN | Tcp Server Id号 | BYTE | 取值0，1 |
| ConnectId | IN | 连接Id号 | BYTE | 每个 Server 最多支持4个连接，connect Id 为（0，1，2，3）。 |

| | | | | |
|---|---|---|---|---|
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令的执行条件Off时，连接成功断开，Busy位被复位。 |
| Err | OUT | 指令错误位 | BOOL | 当指令出错时。Err位置位。 |
| ErrId | OUT | 错误代码 | BYTE | 0：无错<br>1：ServerId 错误<br>4：ConnectId 错误 |
| Active | OUT | 连接状态位 | BOOL | 当连接建立时，Active置位。<br>当连接断开时，Active复位。 |
| rIP0 | OUT | 远程连接的IP地址0 | BYTE | |
| rIP1 | OUT | 远程连接的IP地址1 | BYTE | |
| rIP2 | OUT | 远程连接的IP地址2 | BYTE | |
| rIP3 | OUT | 远程连接的IP地址3 | BYTE | |
| rPort | OUT | 远程连接的端口号 | WORD | |

## H.3 TCP_Send




**1.功能**

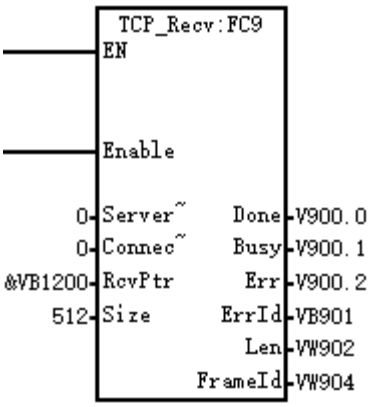
此指令用于 Tcp 发送指令，此指令用于向 TCP 连接发送数据。

**2.参数**

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Excute | IN | 指令触发位 | BOOL | 当Excute上升沿时，触发发送指令。 |
| ServerId | IN | Tcp Server Id号 | BYTE | 取值0，1 |
| ConnectId | IN | 连接Id号 | BYTE | 每个 Server 最多支持 4 个连接，connect Id 为（0，1，2，3）。 |
| SndPtr | IN | 发送指针位 | DWORD | |
| Len | IN | 发送数据长度 | WORD | 最多512字节 |

| | | | | |
|---|---|---|---|---|
| Timeout | IN | 发送超时时间ms | WORD | |
| Done | OUT | 发送完成位 | WORD | 数据发送完成时，Done置位。<br>当指令的执行条件Off时，Done位被复位。<br>在Execute为0时，发送完成，Done将置位一个周期。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令的执行条件Off时,Busy位被复位。 |
| Err | OUT | 指令错误位 | BOOL | 当指令出错时。Err位置位。 |
| ErrId | OUT | 错误代码 | BYTE | 0：无错<br>1：ServerId 错误<br>4：ConnectId 错误<br>5：Server没有打开<br>6：连接状态错误<br>7：发送超时 |

## H.4 TCP_Recv




**1.功能**

此指令用于从 Tcp 连接中接收数据。

**2.参数**

| 参数名 | 输入输出属性 | 参数描述 | 类型 | 备注 |
|---|---|---|---|---|
| Enable | IN | 接收使能位 | BOOL | Enable为1时，指令处于接收数据的状态。<br>Enable为0时，指令不接收数据。 |
| ServerId | IN | Tcp Server Id号 | BYTE | 取值0，1 |
| ConnectId | IN | 连接Id号 | BYTE | 每个 Server 最多支持 4 个连接，connect Id 为（0，1，2，3）。 |
| RcvPtr | IN | 接收数据指针位 | DWORD | |
| Size | IN | 接收数据最大长度 | WORD | 最多512字节 |

| | | | | |
|---|---|---|---|---|
| Done | OUT | 接收完成位 | WORD | 接收到数据时Done置位。<br>当指令的执行条件Off时,Done位被复位。 |
| Busy | OUT | 指令执行位 | BOOL | 当指令执行时。Busy位被置位。<br>当指令的执行条件Off 时,Busy位被复位。 |
| Err | OUT | 指令错误位 | BOOL | 当指令出错时。Err位置位。 |
| ErrId | OUT | 错误代码 | BYTE | 0：无错<br>1：ServerId 错误<br>4：ConnectId 错误<br>5：Server没有打开<br>6：连接状态错误<br>7：发送超时 |
| FrameId | OUT | 接收到数据的帧ID | WORD | 每当接到一帧数据，FrameId会增加1，用户可以依照这个来判断是否有新的数据到来。 |

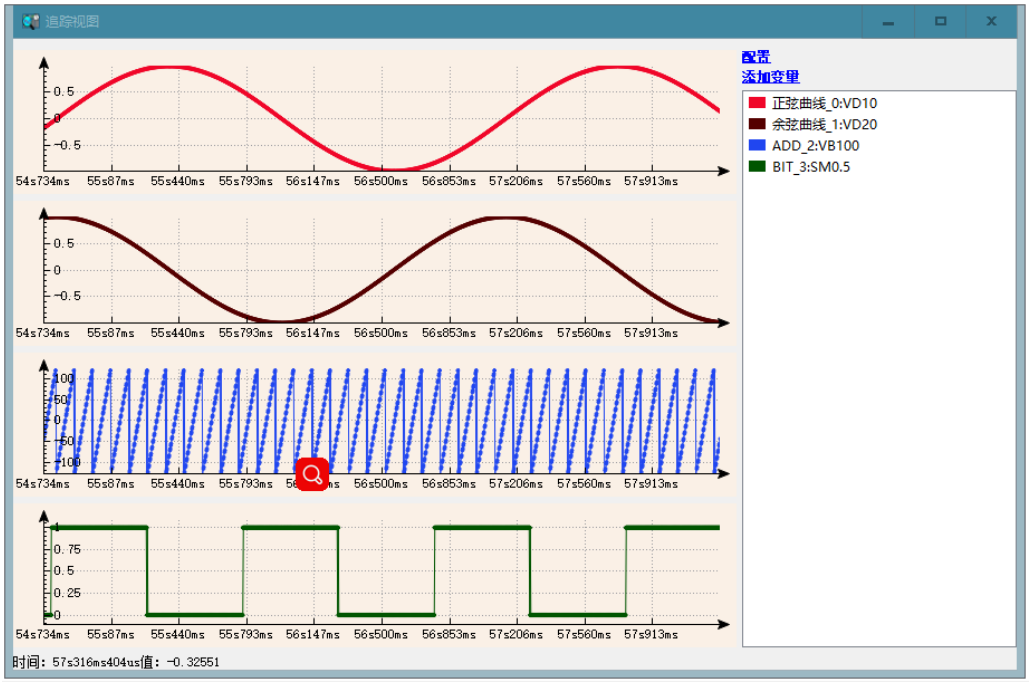
# I Trace 追踪功能

Trace 追踪功能是 MagicWorks PLC 软件的状态表控件中新增的功能，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 MagicWorks PLC 的“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”右上角点击添加变量添加
- 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置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。

# J 特殊存储器说明

表 I-1 特殊存储器功能说明

| 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。 |
| 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~SM3.3 | 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。 |

| 特殊寄存器 | 功能说明 | |
|---|---|---|
| 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 | |
| SMB6 | | |
| SM6.0 | 保留(通道信息存在) | 模块诊断信息 |
| 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：<br>1110：CTH300系列 PLC | |
| SMB7 | | |
| 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 | 0：编程卡存在且工作正常<br>1：编程卡不存在或工作异常 | |
| 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，仅在中断函数里访问有效；<br>SMB28：模块的机架号（0~3），SMB29：模块的槽号（0~10）。 | |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| 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；<br>100：2400bps；101：1200bps；110：115200bps；111：57600bps |
| Bit1:0 | 00：点对点协议（PPI/从属模式）；<br>01：自由口协议；<br>10：PPI/主站模式；<br>11：保留（PPI/从站模式默认值） |
| SMB31 | 永久性内存（EEPROM）写入控制 |
| Bit7 | 0：无执行保存操作请求；<br>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<br>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 |
| SMB100 | 编程卡限制次数（0表示无次数限制） |
| SMB101 | 编程卡剩余使用次数（包括本次），如无限制，显示255。 |
| 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 | MODBUS TCP 通讯在线主站个数 |
| SMD207 | MODBUS TCP 数据包统计：<br>收到的数据包个数 |
| SMD211 | MODBUS TCP 数据包统计：<br>发送的数据包个数 |
| SMB215~SMB222 | 保留 |
| SMB223 | UDP_PPI 通讯在线主站个数 |

| 特殊寄存器 | | 功能说明 |
|---|---|---|
| SMD224 | | UDP_PPI 数据包统计：<br>收到的数据包个数 |
| SMD228 | | UDP_PPI 数据包统计：<br>发送的数据包个数 |
| SMB232~SMB389 | | 保留 |
| SMB390 | | |
| SM390.0 | | EtherCAT 初始化完成位。<br>0：未初始化完成<br>1：初始化完成 |
| SM390.1 | | EtherCAT 冗余激活位。<br>0：冗余未激活<br>1：冗余激活，此时需检查连线 |
| SM390.2~SM390.7 | | 保留 |
| SMB391 | | 保留 |
| SMD392 | | EtherCAT 任务周期执行时间 |
| SMD396 | | EtherCAT 任务周期间隔时间 |
| SMB400~SMB465 (EtherCAT 寄存器) | SMB400 | 找到的 EtherCAT 从站的个数 |
| | SMB401 | EtherCAT 错误：<br>0：没有错误<br>1：组态参数错误<br>2：没有找到从站<br>3：状态切换错误<br>4：写组态时发生错误<br>5: 从站个数错误<br>6: 从站不匹配 |
| | SMB402 | 从站1状态：<br>0：没有连接<br>1：初始化状态<br>2：预操作状态<br>4：安全操作<br>8：操作状态<br>16#80 产品 ID 不匹配<br>16#81 厂商 ID 不匹配<br>16#82 SDO 写入出错<br>其它：错误的状态 |
| | SMB403~SMB465 | 从站2状态~从站64状态： |
| SMB466~SMB499 | | 保留 |
| SMW470 | | EtherCAT 错误包数 |
| SMB500 | | 第一个模块(Rack0，Slot3)错误 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| SMB501 | 第二个模块(Rack0，Slot4)错误 |
| SMB502 | 第三个模块(Rack0，Slot5)错误 |
| SMB503 | 第四个模块(Rack0，Slot6)错误 |
| SMB504 | 第五个模块(Rack0，Slot7)错误 |
| SMB505 | 第六个模块(Rack0，Slot8)错误 |
| SMB506 | 第七个模块(Rack0，Slot9)错误 |
| SMB507 | 第八个模块(Rack0，Slot10)错误 |
| SMB508 | 第九个模块(Rack1，Slot3)错误 |
| SMB509 | 第十个模块(Rack1，Slot4)错误 |
| SMB510 | 第十一个模块(Rack1，Slot5)错误 |
| SMB511 | 第十二个模块(Rack1，Slot6)错误 |
| SMB512 | 第十三个模块(Rack1，Slot7)错误 |
| SMB513 | 第十四个模块(Rack1，Slot8)错误 |
| SMB514 | 第十五个模块(Rack1，Slot9)错误 |
| SMB515 | 第十六个模块(Rack1，Slot10)错误 |
| SMB516 | 第十七个模块(Rack2，Slot3)错误 |
| SMB517 | 第十八个模块(Rack2，Slot4)错误 |
| SMB518 | 第十九个模块(Rack2，Slot5)错误 |
| SMB519 | 第二十个模块(Rack2，Slot6)错误 |
| SMB520 | 第二十一个模块(Rack2，Slot7)错误 |
| SMB521 | 第二十二个模块(Rack2，Slot8)错误 |
| SMB522 | 第二十三个模块(Rack2，Slot9)错误 |
| SMB523 | 第二十四个模块(Rack2，Slot10)错误 |
| SMB524 | 第二十五个模块(Rack3，Slot3)错误 |
| SMB525 | 第二十六个模块(Rack3，Slot4)错误 |
| SMB526 | 第二十七个模块(Rack3，Slot5)错误 |
| SMB527 | 第二十八个模块(Rack3，Slot6)错误 |
| SMB528 | 第二十九个模块(Rack3，Slot7)错误 |
| SMB529 | 第三十个模块(Rack3，Slot8)错误 |
| SMB530 | 第三十一个模块(Rack3，Slot9)错误 |
| SMB531 | 第三十二个模块(Rack3，Slot10)错误 |
| SMD532 | 扩展总线链路层错误总数 |
| SMB536~SMB549 | 保留 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| SMB550~SMB589 | 第一个 CAN 模块信息（按硬件组态排列） |
| SMB550 | 主站状态<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB551 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB552~SMB581 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |
| SMB582 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB591 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB592~SMB621 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| SMB622 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB631 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| SMB671 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB672~SMB701 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |
| SMB702 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB711 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB712~SMB741 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |
| SMB742 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB743 | CAN 模块软件版本号 |
| SMB744~SMB749 | 保留 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| SMB750~SMB789 | 第六个 CAN 模块信息（按硬件组态排列） |
| SMB750 | 主站状态<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB751 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB752~SMB781 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |
| SMB782 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB791 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB792~SMB821 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| SMB822 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB831 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行 |
| SMB832~SMB861 | 从站2状态（按节点 ID 排列）~<br>从站31状态（按节点 ID 排列） |
| SMB862 | 从站32状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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<br>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<br>需要获取内部诊断的模块号，<br>0~3位：槽号 SlotID；<br>4~7位：机架号 RackID。 |
| SMB872~SMB905 | 模块内部诊断 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| 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 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| 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 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| 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 | 第二十三个模块(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 | 第二十四个模块(Rack2，Slot10)描述信息 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| 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 |

| 特殊寄存器 | 功能说明 |
| --- | --- |
| 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<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>0x7F：预运行<br>0xFF：组态数据出错 |
| SMB1551 | 从站1状态（按节点 ID 排列）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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）<br>0x00：初始化<br>0x01：断开连接<br>0x04：停止<br>0x05：运行<br>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 |

# K 指令速查

表 J-1 300CPU 指令速查

| 位逻辑指令 | | | |
|---|---|---|---|
| LD | 常开触点载入指令<br>LD SM0.0 | ALD | 与载入指令<br>ALD |
| A | 常开触点与指令<br>A SM0.0 | OLD | 或载入指令<br>OLD |
| O | 常开触点或指令<br>O SM0.0 | LPS | 逻辑进栈指令<br>LPS |
| LDN | 常闭触点载入指令<br>LDN SM0.0 | LDS | 载入堆栈指令<br>LDS 1 |
| AN | 常闭触点与指令<br>AN SM0.0 | LRD | 逻辑读取指令<br>LRD |
| ON | 常闭触点或指令<br>ON SM0.0 | LPP | 逻辑出栈指令<br>LPP |
| LDI | 常开触点立即载入指令<br>LDI I0.0 | = | 输出指令<br>= Q0.0 |
| AI | 常开触点与立即指令<br>AI I0.0 | =I | 立即输出指令<br>=I Q0.0 |
| OI | 常开触点或立即指令<br>OI I0.0 | S | 置位指令<br>S Q0.0 1 |
| LDNI | 常闭触点立即载入指令<br>LDNI I0.0 | SI | 立即置位指令<br>SI Q0.0 1 |
| ANI | 常闭触点与立即指令<br>ANI I0.0 | R | 重设指令<br>R Q0.0 1 |
| ONI | 常闭触点或立即指令<br>ONI I0.0 | RI | 立即重设指令<br>RI Q0.0 1 |
| NOT | 逻辑栈顶取反指令<br>NOT | AENO | 功率流与指令<br>AENO |
| EU | 上升沿检测指令<br>EU | NOP | 空操作指令<br>NOP 1 |
| ED | 下降沿检测指令<br>ED | -- | -- |

| 比较指令 | | | |
|---|---|---|---|
| LDBx | 字节比较（=,<,>,<=,>=,<>）载入指令<br>LDB= 1, VB0 | OBx | 字节操作数比较（=，<,>,<=,>=,<>）或指令<br>OB= 1, VB0 |
| LDWx | 整数比较（=,<,>,<=,>=,<>）载入指令<br>LDW= 10000, VW0 | OWx | 整数比较（=,<,>,<=，>=,<>）或指令<br>OW= 10000, VW0 |
| LDDx | 长整数比较（=,<,>,<=,>=,<>）载入指令<br>LDD= 100000, VD0 | ODx | 双整数比较（=,<,>，<=,>=,<>）或指令<br>OD= 100000, VD0 |
| LDRx | 浮点数比较（=,<,>,<=,>=,<>）载入指令<br>LDR= 1.0, VD0 | ORx | 浮点数比较（=,<,>，<=,>=,<>）或指令<br>OR= 1.0, VD0 |
| ABx | 字节操作数比较（=,<,>,<=,>=,<>）与指令<br>AB= 1, VB0 | LDSx | 字符串比较（=,<>）载入指令<br>LDS="1234567890",VB0 |
| AWx | 整数比较（=,<,>，<=,>=,<>）与指令<br>AW= 10000, VW0 | ASx | 字符串比较（=,<>）与指令<br>AS="1234567890", VB0 |
| ADx | 双整数比较（=,<,>，<=,>=,<>）与指令<br>AD= 100000, VD0 | OSx | 字符串比较（=,<>）或指令<br>OS="1234567890",VB0 |
| ARx | 浮点数比较（=,<,>，<=,>=,<>）与指令<br>AR= 1.0, VD0 | -- | -- |
| **传送指令** | | | |
| MOVB | 字节移动指令<br>MOVB 1 VB0 | SWAP | 高低字节交换指令<br>SWAP VW0 |
| MOVW | 字移动指令<br>MOVW 1000,VD0 | BIR | 移动字节立即读指令<br>BIR IB0, VB0 |
| MOVD | 双字移动指令<br>MOVD 100000,VD0 | BIW | 移动字节立即写指令<br>BIW VB0, QB0 |
| MOVR | 浮点数移动指令<br>MOVR 1.0,VD0 | SRCP | 存储配方到存储卡 |
| BMB | 块移动字节指令<br>BMB VB0,VB100, 1 | LRCP | 从存储卡装载配方 |
| BMW | 块移动字指令<br>BMW VW0,VW100, 1 | SDL | 存储数据记录到存储卡 |
| BMD | 块移动双字指令<br>BMD VD0,VD100, 1 | -- | -- |

| | | | |
|---|---|---|---|
| **整数计算指令** | | | |
| +I | 整数加法指令<br>+I 10000, VW0 | MUL | 整数与双整数相乘<br>MUL 10000, VD0 |
| -I | 整数减法指令<br>-I 10000, VW0 | DIV | 整数与双整数相除<br>DIV VW0, VD0 |
| *I | 整数乘法指令<br>*I 10000, VW0 | INCB | 字节递增指令<br>INCB VB0 |
| /I | 整数除法指令<br>/I 10000, VW0 | DECB | 字节递减指令<br>DECB VB0 |
| +D | 双整数加法指令<br>+D 100000, VD0 | INCW | 整数递增指令<br>INCW VW0 |
| -D | 双整数减法指令<br>-D 100000, VD0 | DECW | 整数递减指令<br>DECW VW0 |
| *D | 双整数乘法指令<br>*D 100000, VD0 | INCD | 长整数递增指令<br>INCD VD0 |
| /D | 双整数除法指令<br>/D 100000, VD0 | DECD | 长整数递减指令<br>DECD VD0 |
| **浮点数运算指令** | | | |
| +R | 实数加法指令<br>+R 1.0, VD0 | COS | 余弦运算指令<br>COS 1.0, VD0 |
| -R | 实数加法指令<br>-R 1.0, VD0 | TAN | 正切运算指令<br>TAN 1.0, VD0 |
| *R | 实数乘法指令<br>*R 1.0, VD0 | LN | 自然对数运算指令<br>LN 1.0, VD0 |
| /R | 实数除法指令<br>/R 1.0, VD0 | EXP | 自然指数运算指令<br>EXP 1.0, VD0 |
| SQRT | 平方根指令 | PID | PID 环路运算指令<br>PID VB0, 1 |
| SIN | 正弦运算指令<br>SIN 1.0, VD0 | -- | -- |
| **转换指令** | | | |
| BTI | 字节至整数转换<br>BTI 1, VW0 | DTR | 双整数至实数转换<br>DTR 100000, VD0 |
| ITB | 整数至字节转换<br>ITB 10000, VB0 | DTS | 双整数至字符串<br>DTS 100000, VB0, 10 |
| ITD | 整数至双整数转换<br>ITD 10000, VD0 | ROUND | 进位取整指令<br>ROUND 1.0, VD0 |
| ITS | 整数转换为字符串<br>ITS 10000, VB0, 10 | TRUNC | 截位取整指令<br>TRUNC 1.0, VD0 |
| DTI | 双整数至整数转换<br>DTI 100000, VW0 | RTS | 实数至字符串转换<br>RTS 1.0, VB0, 10 |

| 转换指令 | | | |
|---|---|---|---|
| BCDI | BCD 至整数转换<br>BCDI　VW0 | STI | 字符串至整数转换<br>STI"1234567890", 5, VW0 |
| IBCD | 整数至 BCD 转换<br>IBCD　VW0 | STD | 字符串至双整数转换<br>STD"1234567890", 5, VD0 |
| ITA | 整数至 ASCII 转换<br>ITA 10000,VB0,10 | STR | 字符串至实数转换<br>STR"1234567890",5, VD0 |
| DTA | 双整数至 ASCII 转换<br>DTA 100000, VB0, 10 | DECO | 解码指令<br>DECO 1, VW0 |
| RTA | 实数至 ASTII 码转换指令 | ENCO | 编码指令<br>ENCO 10000, VB0 |
| ATH | ASCII 至16进制转换<br>ATH VB0, VB100, 10 | SEG | 段指令<br>SEG 1, VB0 |
| HTA | 16进制至 ASCII 转换<br>HTA VB0, VB100, 10 | -- | -- |
| 实时时钟 | | | |
| TODR | 读取实时时钟<br>TODR VB0 | TODRX | 读取扩展的实时时钟<br>TODRX VB0 |
| TODW | 设置实时时钟<br>TODW VB0 | TODWX | 设置扩展的实时时钟<br>TODWX VB0 |
| 移位循环指令 | | | |
| SLB | 向左移位字节<br>SLB VB0, 4 | RLW | 向左旋转字<br>RLW VW0, 8 |
| SLW | 向左移位字<br>SLW VW0, 8 | RLD | 向左旋转双字<br>RLD VD0, 16 |
| SLD | 向左移位双字<br>SLD VD0, 16 | RRB | 向右旋转字节<br>RRB VB0, 4 |
| SRB | 向右移位字节<br>SRB VB0, 4 | RRW | 向右旋转字<br>RRW VW0, 8 |
| SRW | 向右移位字 | RRD | 向右旋转双字<br>RRD VD0, 16 |
| SRD | 向右移位双字<br>SRD VD0, 16 | SHRB | 移位寄存器位指令<br>SHRB I0.0, V0.0, 8 |
| RLB | 向左旋转字节<br>RLB VB0, 4 | -- | -- |

| | | | |
|---|---|---|---|
| **逻辑计算指令** | | | |
| INVB | 反转字节指令<br>INVB VB0 | ORB | 或字节指令<br>ORB 1, VB0 |
| INVW | 反转字指令<br>INVW VW0 | ORW | 或字指令<br>ORW 10000, VW0 |
| INVD | 反转双字指令<br>INVD VD0 | ORD | 或双字指令<br>OD 100000, VD0 |
| ANDB | 与字节指令<br>ANDB 1, VB0 | XORB | 异或字节指令<br>XORB 1, VB0 |
| ANDW | 与字指令<br>ANDW 10000, VW0 | XORW | 异或字指令<br>XORW 10000, VW0 |
| ANDD | 与双字指令<br>ANDD 100000, VD0 | XORD | 异或双字指令<br>XORD 100000, VD0 |
| **计数器指令** | | | |
| CTU | 向上计数指令<br>CTU C1, 10000 | HDEF | 高速计数器定义<br>HDEF 0, 0 |
| CTD | 向下计数指令<br>CTD C1, 10000 | HSC | 高速计数器指令 |
| CTUD | 向上/向下计数指令<br>CTUD C1, 10000 | PLS | 脉冲输出指令 |
| **定时器指令** | | | |
| TON | 打开延时计时器<br>TON T37, 10000 | BITIM | 读取开始时间计时器<br>BITIM VD0 |
| TONR | 保留性打开延时计时器<br>TONR T31, 10000 | R_UTIM | 开始间隔时间捕捉(us) |
| TOF | 关闭延时计时器<br>TOF T37, 10000 | CITIM | 计算间隔时间(ms)<br>CITIM VD0, VD100 |
| **字符串指令** | | | |
| SLEN | 取字符串长度指令<br>SLEN"1234567890", VB0 | SCAT | 并置字符串指令<br>SCAT"1234567890", VB0 |
| SCPY | 复制字符串指令<br>SCPY"1234567890", VB0 | SFND | 在字符串中查找字符串指令<br>SFND"12345678890", "321", VB0 |
| SSCPY | 从字符串复制子字符串指令<br>SSCPY"1234567890", 1, 10, VB0 | CFND | 在字符串中查找一个字符指令<br>CFND"12345678890", "a", VB0 |
| **表指令** | | | |
| FILL | 内存填充指令<br>FILL 10000, VW0, 10 | FIFO | 先入先出指令<br>FIFO VW200, VW400 |
| ATT | 增加至表格指令<br>ATT 10000, VW0 | LIFO | 后入先出指令<br>LIFO VW200, VW400 |

| | | | |
|---|---|---|---|
| FNDx | 表格查找（=,<,>,<=，>=,<>）指令<br>FND=VW0,9999,VW1000 | -- | -- |
| **中断指令** | | | |
| CRETI | 从中断程序有条件返回指令 | ATCH | 附加中断指令 |
| ENI | 启用中断指令<br>ENI | DTCH | 分离中断指令 |
| DISI | 禁用中断指令<br>DISI | CEVNT | 清除中断事件指令<br>CEVNT 10 |
| **程序控制指令** | | | |
| FOR | FOR-NEXT 循环开始<br>FOR-NEXT<br>FOR VW0, 1, 10<br>NEXT | LSCR | 载入顺序控制中继 |
| NEXT | FOR-NEXT 循环结束 | SCRE | 顺序控制中继结束 |
| JMP | 跳转至标签指令 | SCRT | 顺序控制中继转换 |
| LBL | 标签指令 | CSCRE | 有条件顺序控制中继结束 |
| WDR | 监视器重设指令 | CRET | 从子程序有条件返回指令 |
| DLED | 诊断 LED 指令<br>DLED 1 | END | 有条件结束 |
| CALL | 调用子例行程序指令 | STOP | 停机指令 |
| **通信指令** | | | |
| XMT | 自由口发送指令 | SPA | 设置端口指令 |
| RCV | 自由口接收指令 | GPA | 获得端口地址指令 |
| NETR | 网络读取指令 | EBUSR | 读取模块信息指令 |
| NETW | 网络写入指令 | EBUSW | 写入模块信息指令 |
| UDPNETR | 以太网网络读功能 | EBUSGETDIA | 获取模块内部诊断信息指令 |
| UDPNETW | 以太网网络写入功能 | EBUSSNDCMD | 发送模块操作命令指令 |
| MBSNDMSG | 发送 Modbus 消息 | -- | -- |

**<备注>** 请参考《MagicWorks PLC 用户手册》获取以上指令的详细使用说明，该手册可从合信公司网站免费下载：http://www.co-trust.com/Download/index.html

服务热线

400-700-4858

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