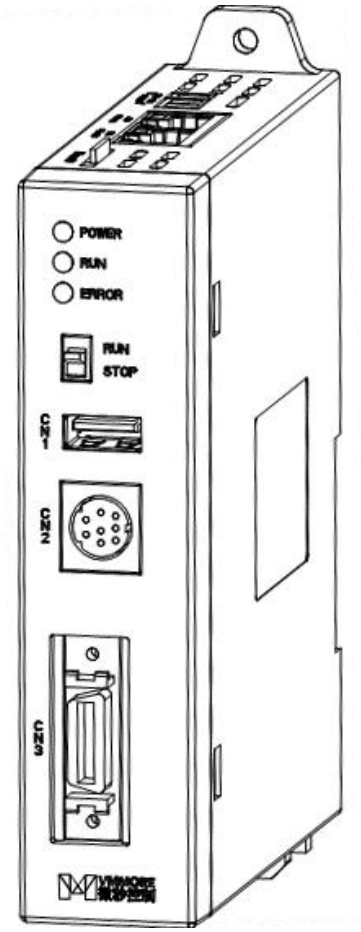
VMMORE

中型 PLC

PC4M 系列

用户手册

PC4M-MC100\101\102EC



PC4M 系列中型 PLC 用户手册

资料版本　V1.1.0
归档日期　2023-5-5

日期	变更后版本	变更内容
2022 年 07 月	V1.0.0	文档发布。
2022 年 11 月	V1.0.1	修改 1.4 章节前 3 行及前 2 段。
2023 年 5 月	V1.1.0	增加 SD 卡槽使用说明，增加 USB 口使用说明，增加 Reset 按键使用说明。

● 安全注意事项 ●

安装、使用、维修、检查之前必须仔细阅读本手册、使用说明书、伺服电机技术资料和相关资料。请在对设备情况、安全信息和注意事项都完全清楚以后再进行使用。

本手册中，安全注意事项的级别分为【危险】和【注意】两种。

【危险】		误操作时，会导致危险情况发生，可能会引起死亡或重伤。
【注意】		误操作时，会导致危险情况发生，可能会引起中等程度的伤害或轻伤，另外可能会引起物品的损坏。另外，即使是【注意】当中记录的事项，在某些情况下也可能导致严重后果。

所以无论是哪种标志都记录了很重要的内容，请务必遵守。

禁止、强制的标志的说明如下。

 表示禁止（不能做的事情）。例如，  表示严禁烟火。

 表示强制（必须做的事情）。例如，  表示必须接地。

1. 启动、维护保养时的危险事项

 危险

- 通电时请勿触碰接线板上的端子，否则可能造成机台误动作。
- 接线和检查要由专门的技术人员进行。
- 进行接线前请务必先断开所有外部电源。
- 请不要损坏电缆，强拉电缆，在电缆上放置重物或挤压电缆，否则可能造成触电。
- 要在运行过程中更改程序或执行强制输出或 RUN/STOP 等操作前，请务必先熟读手册，在充分确定安全的情况下方可进行操作。
- 否则可能由于操作错误引起机械的损坏及事故。

2. 启动、维护保养时的注意事项

 注意

- 请勿擅自拆解、改动产品，否则有可能引起故障、误动作、火灾。
- 请务必断开电源后方可拆装网线等连接线缆。
- 否则有可能引起故障、误动作。
- PLC 内的电解电容由于老化会造成容量降低。为防止由于故障导致的二次危害，在一般环境中使用时，建议 10 年左右进行更换。

3. 运输、保管时的注意

注意
根据产品的重量请采用正确的搬运方法。 不要超过产品堆积的数量限制。 运输可编程控制器时，请务必在运输前对可编程控制器上电，对“BATT 的指示灯处于 OFF 状态”及“电池的寿命”进行确认。如果在 BATT 指示灯处于 ON 状态或是超出寿命的状态下运输，则在运输过程中实时时钟数据可能处于不稳定的状态。 可编程控制器是精密设备，请不要使其跌落或遭受强力冲击。否则可能导致可编程控制器故障。送达后，请务必对可编程控制器进行动作确认。

4. 废弃时的注意

注意
产品废弃时，请按工业废弃物处理。

目录

1 功能和构成

PC4M 系列中型 PLC 是一款基于嵌入式系统的处理单元，采用最新的架构实现。支持开放的工业以太网 EtherCAT 总线，可以满足用户灵活的扩展需求，使用 3S 的 Codesy3.5 作为调试开发环境，方便易用。配合微秒特有的分布式控制方案，可以极大地提升您的生产效率和降低系统成本。

产品特点： 高效、简洁、经济

1> 使用微秒的分布式控制方案，32 轴 PDO 同步 1ms
2> 更快的执行速度，单条 BOOL 指令 1ns
3> 高扩展性，可使用 EtherCAT 总线扩展第三方 IO、计数、温控等模块
4> 简洁、经济，无本地总线设计,不需电源模块，直接外部 24V 供电。
5> 丰富的低成本接口，3 路 RS485，分别支持、自由口、Modbus 协议
6> 主流的编程环境 Codesys3.5，方便易用

1.1 PC4M 标准规格

PC4 M – MC 10 0 EC

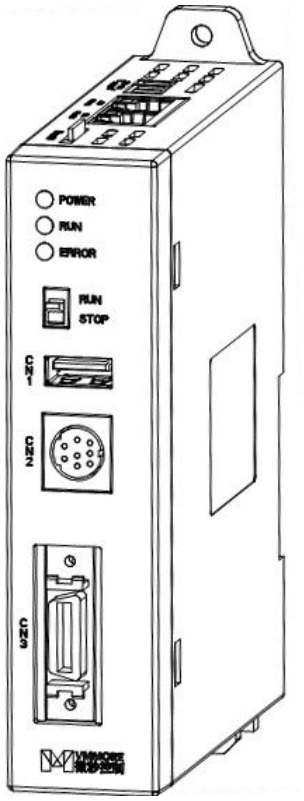
重要控制总线：
EC： 支持 EtherCAT 总线主站

软件版本：
0： 支持PN 方案
1： 支持PN 方案和 SoftMotion Basic
2： 支持PN 方案和 SoftMotion Basic& CNC

硬件版本：
10: 第一代通用硬件平台

产品功能：
MC: 运动控制型

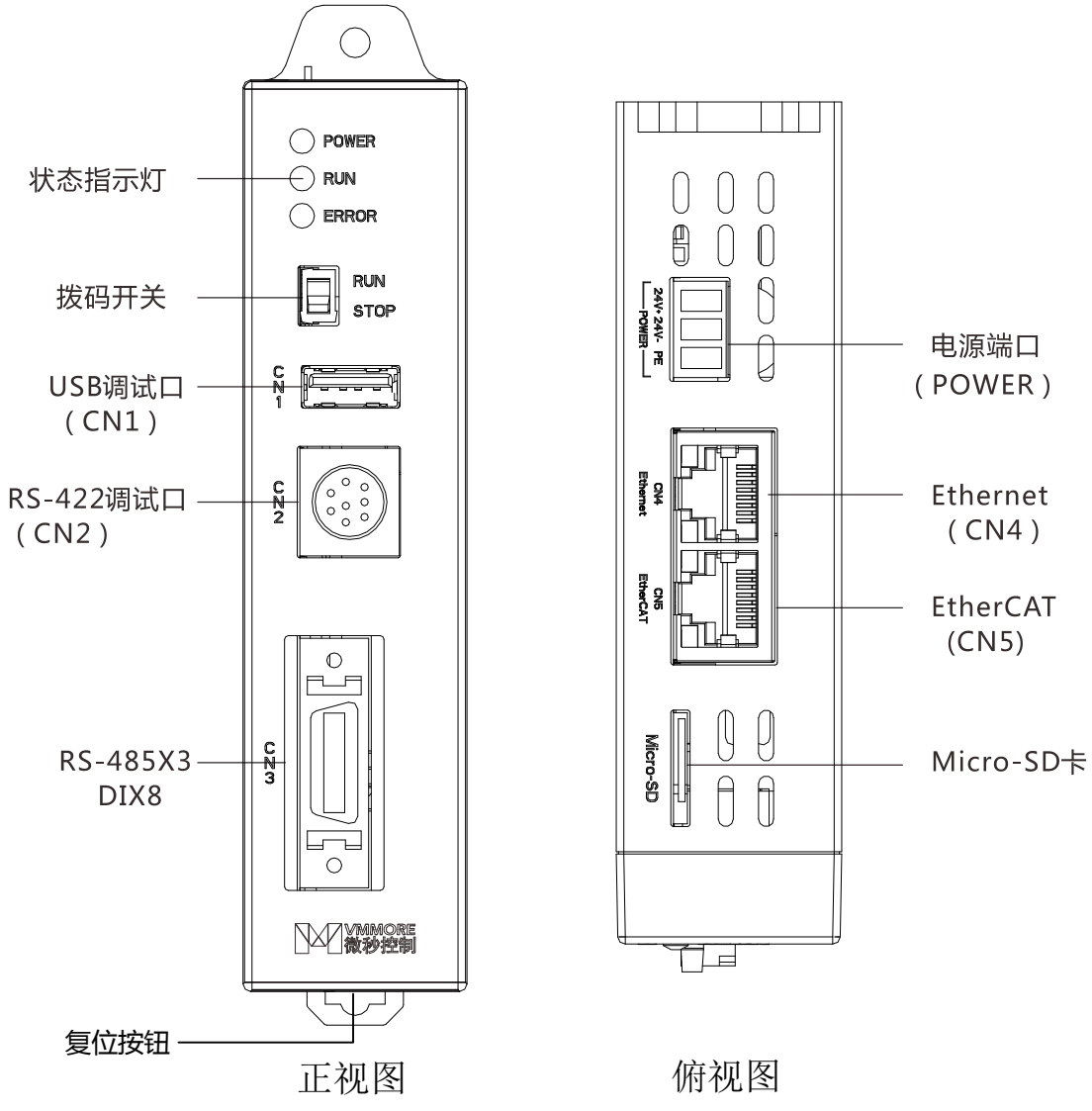
产品系列：
经济型中型PLC 系列



1

项目	规格
编程语言	IEC61131-3 编程语言（LD，FBD，IL，ST，SFC，CFC）
程序执行方式	编译执行
用户程序空间	16MByte
用户数据容量	16Mword
掉电保持容量	512KByte
I 输入区域大小	64KWord
Q 输出区域大小	64KWord
M 内存区域大小	240KWord
SD 卡存储容量	最多 32G 通用 SD 卡
指令性能	布尔指令 1 ns/step 字操作指令 2 ns/step 双字操作指令 2 ns/step 浮点指令 10 ns/step

1.2 各部分名称


状态指示灯
POWER
RUN
ERROR
拨码开关
RUN
STOP
USB调试口
（CN1）
CN1
RS-422调试口
（CN2）
CN2
RS-485X3
DIX8
CN3
VMMORE
微秒控制
复位按钮
正视图
24V- 24V+ PE
POWER
电源端口
（POWER）
CN4
Ethernet
Ethernet
（CN4）
CN5
EtherCAT
EtherCAT
(CN5)
Micro-SD
Micro-SD卡
俯视图

1.3 接口定义

1.3.1 POWER 电源端子

信号定义如下表：。

Pin	信号
1	24V+
2	24V-
3	PE

DC24V，0.5A，支持反接保护。

1.3.2 CN1 USB-TypeA

CN1 的信号定义如下表：

Pin	信号
1	USB0_VBUS
2	USB0_DM
3	USB0_DP
4	GND

功能：更新 PLC 程序

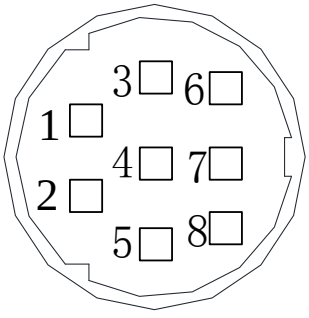
将 Application.app、Application.crc 放置于 U 盘根目录。将 U 盘插到 PLC 的 CN1（USB 接口）上，重启后，程序会被更新。程序会自动更新到 PLC 上。

PLC 程序 U 盘访问路径：/mnt/usb/

1.3.3 CN2 RS422 通讯口

功能：Codesys 后台通讯调试

CN2 接口形状与针脚排列如下图所示：

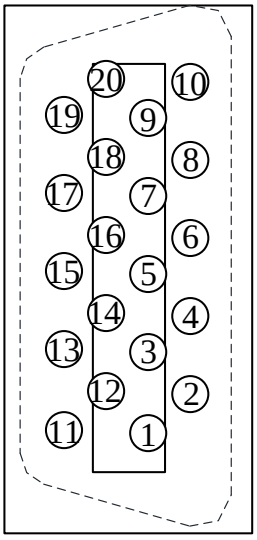


CN2 接口可以连接增量型编码器（左图），也可以连接绝对值型编码器（右图），具体信号定义如下表：

针脚号	信号	描述
1	RX-	串行数据接收引脚，RS422 差分信号负端
2	RX+	串行数据接收引脚，RS422 差分信号正端
3	GND	信号地
4	TX-	串行数据发送引脚，RS422 差分信号负端
5	+5V	+5V 电源信号
6	保留	未作定义的引脚，禁止用户连接
7	TX+	串行数据发送引脚，RS422 差分信号正端
8	保留	未作定义的引脚，禁止用户连接

1. 3. 4 CN3 SCSI-20 DI 输入和通讯口

插头核心针脚排列如下图所示：

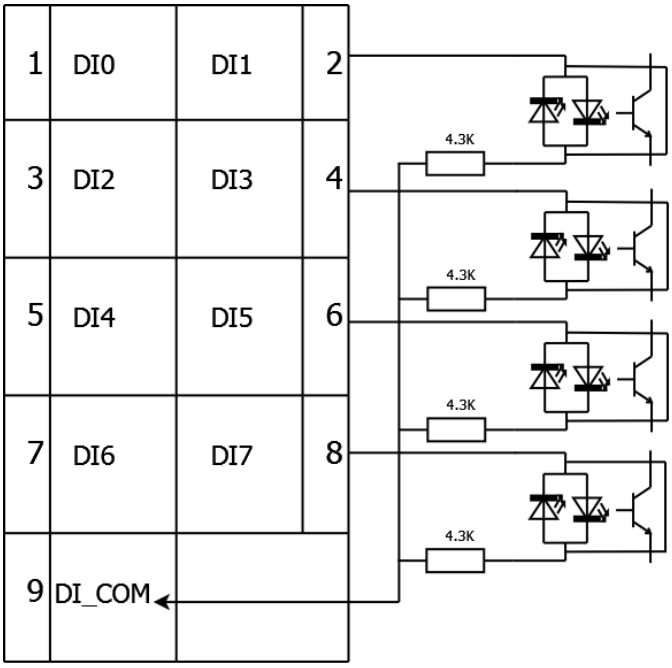


CN3 接口的信号定义如下表：

针脚号	名称	描述
1	DI0	数字量输入 1
2	DI1	数字量输入 2
3	DI2	数字量输入 3
4	DI3	数字量输入 4
5	DI4	数字量输入 5
6	DI5	数字量输入 6
7	DI6	数字量输入 7
8	DI7	数字量输入 8
9	DI_COM	数字量输入公共端
10	GND	信号地
11	GND	信号地
12	RS485_P1+	RS485 差分信号正端 1
13	RS485_P1-	RS485 差分信号负端 1
14	RS485_P2+	RS485 差分信号正端 2
15	RS485_P2-	RS485 差分信号负端 2
16	RS485_P3+	RS485 差分信号正端 3
17	RS485_P3-	RS485 差分信号负端 3
18	GND	信号地

RS485_PX+: 表示第 X 个 485 口的 RS485 A
RS485_PX-: 表示第 X 个 485 口的 RS485 B

DI 输入接线方式



1.3.5 CN4 Ethernet 通讯口

协议：EtherNet TCP/UDP
功能：MODBUS TCP 通讯、Codesys 后台通讯调试

1.3.6 CN5 EtherCAT 通讯口

功能：EtherCAT 协议，连接 EtherCAT 从站设备

1.3.7 系统运行拨码开关定义

信号	拨码方向	信号行为
RUN	向上	PLC 程序运行
STOP	向下	PLC 程序停止

1.3.8 Reset 按键

1.PLC 格式化
PLC 断电状态下，长按 Reset 按键（建议单手操作），直到 PLC 上电，Error 灯闪烁后释放按键。此时 Error 灯会熄灭，PLC 格式化完成。

2.PLC IP 复位
PLC 运行状态下，长按 Reset 按键，直到 Error 灯长亮后释放按键。重启 PLC 后，IP 将复位成“192.168.1.250”。

1.3.9 SD 卡接口

将 Application.app、Application.crc 放置于 SD 卡的根目录。将 SD 卡插到 PLC 顶面的 SD 卡插槽上，重启后，程序会被更新。
PLC 程序 SD 卡访问路径：/mnt/sd1/

1.4 PLC 功能

1.4.1 PC4M 提供三类运动控制功能

- PC4M-MC100EC 通过 PN 参数配置功能配合 300N 系列伺服驱动器实现运动控制功能。
- PC4M-MC101EC 通过 Softmotion 功能配合 300N 系列或 700 系列伺服驱动器实现运动控制功能。
- PC4M-MC102EC 开放 Softmotion CNC 功能，运动控制功能更加丰富。

● 关于分布式运动控制方式与集中式运动控制方式

分布式控制方式采用带有运动控制功能的从站： MSD 型伺服驱动器实现，通过使用微秒提供 IEC 库，可以形成独立的控制小系统。

集中式运动控制方式时，推荐使用 3S 的 SoftMotion 运动控制库实现，使用该库可以实现对基于 CIA402 的 EtherCAT 伺服实现单轴主从轴控制，详细使用及概念可参考 Codesys3.5 帮助文档和 PLCOpen 文档。

- PC4M 总线协议支持

物理层：以太网
应用层协议： EtherCAT 、MODBUS TCP 主站、从站

物理层： RS485
应用层协议：MODBUS RTU 主站、从站

- 逻辑控制功能

三种程序组织单元

- 程序（Program）
- 功能块（Function Block）
- 功能（Function）

六种编程语言

- 指令表（IL）
- 结构化文本（ST）
- 顺序功能图表（SFC）
- 功能模块图（FBD）
- 梯形图（LD）
- 连续功能图表（CFC）

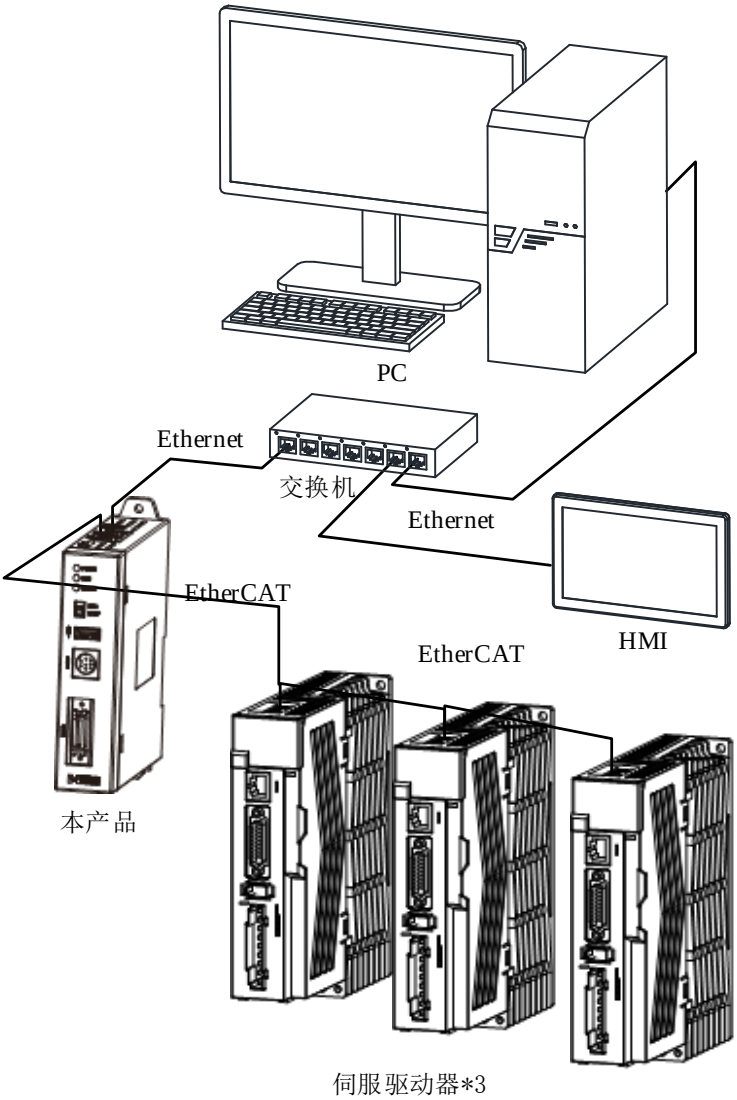
标准化的 PLC 功能

类型转换、数值功能、算术功能、移位功能、布尔运算功能、选择功能、比较功能、字符串功能、定时器、计数器、边沿检测、双稳态元素

<备注>关于编程使用请参考第二章

1.5 连接、配线、安装

1.5.1 连接



典型系统应用

1.5.2 安装注意事项

将 PC4M 系列 PLC 与加热装置、高电压和电子噪声隔离开

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 PC4M 系列 PLC 这样的低压电子型的设备分隔开。

在控制柜的背板上安排 PC4M 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。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。

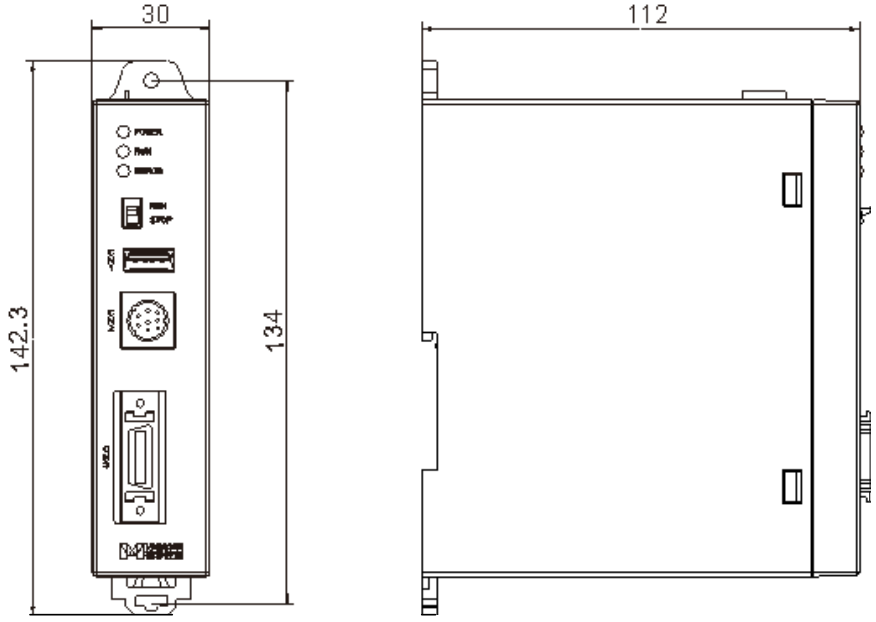
为散热和接线留出适当的空间

PC4M 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

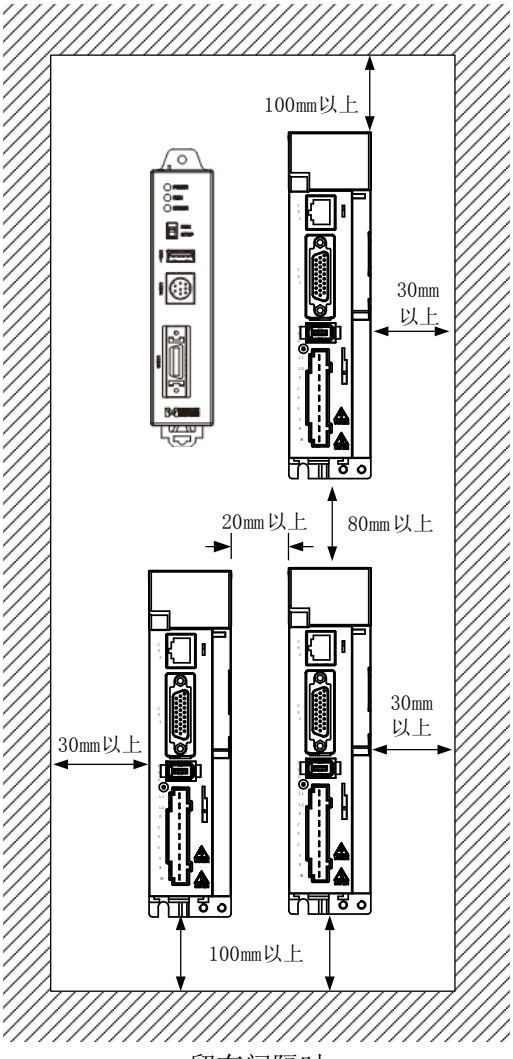
在安装 PC4M 系列 PLC 时，应留出足够空间用于接线和连接通讯电缆。

1.5.3 安装尺寸

PC4M 系列 PLC 配置有安装孔，可以很方便地安装在背板上，安装尺寸见下图。

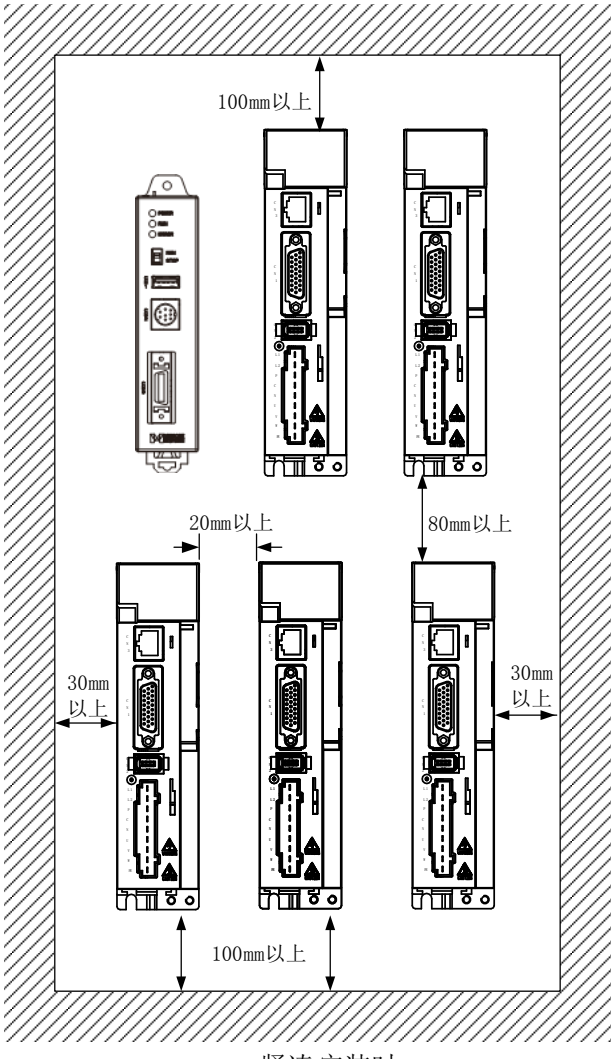


控制柜

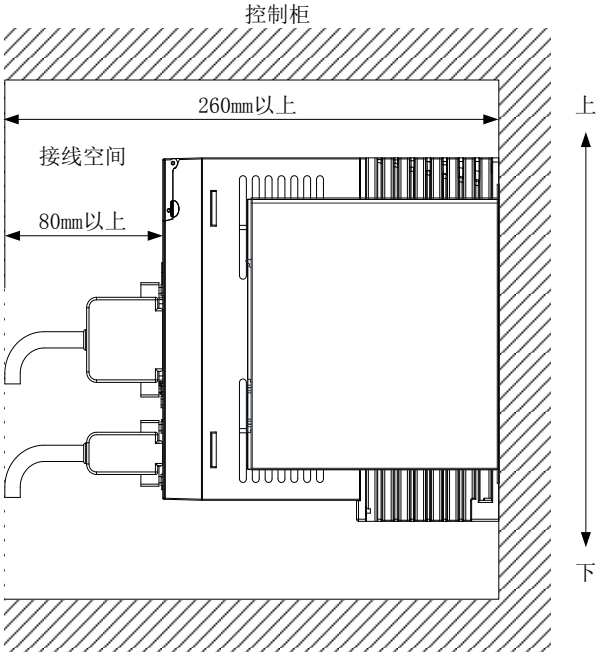


留有间隔时

控制柜



紧凑安装时


控制柜
260mm以上
接线空间
80mm以上
上
下

2 编程环境安装及配置

2.1 Codesys3.5 SP11 安装

PC 硬件及系统要求：操作系统 WIN7 SP1 及以上版本，CPU 主频 2GHZ 及以上，硬盘预留存储空间 5GB 及以上，电脑 RAM 内存至少 2GB。

安装软件及 PLC 设备描述文件、学习资料例程请从微秒网站或代理商处获取。

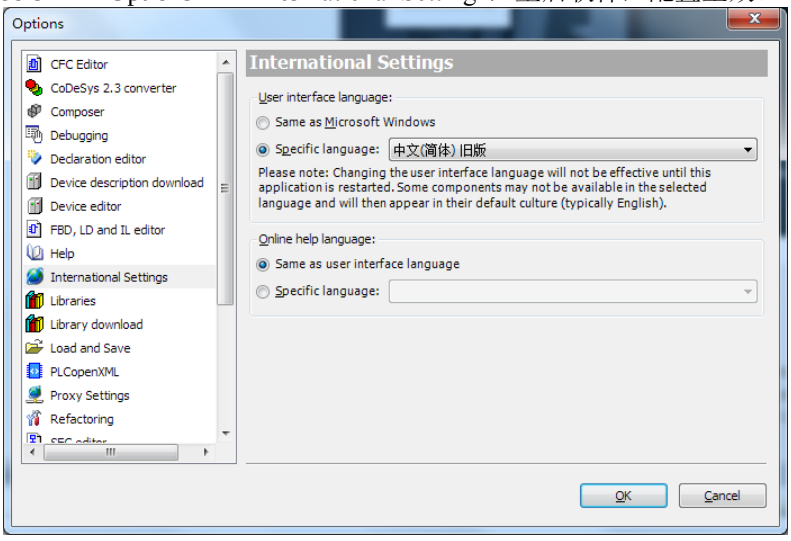
双击 Setup_CODESYSV35SP11.exe 根据提示操作点击下一步即可安装。

2.2 编程环境配置及使用

双击 CODESYS V3.5 SP11 软件图标 启动软件

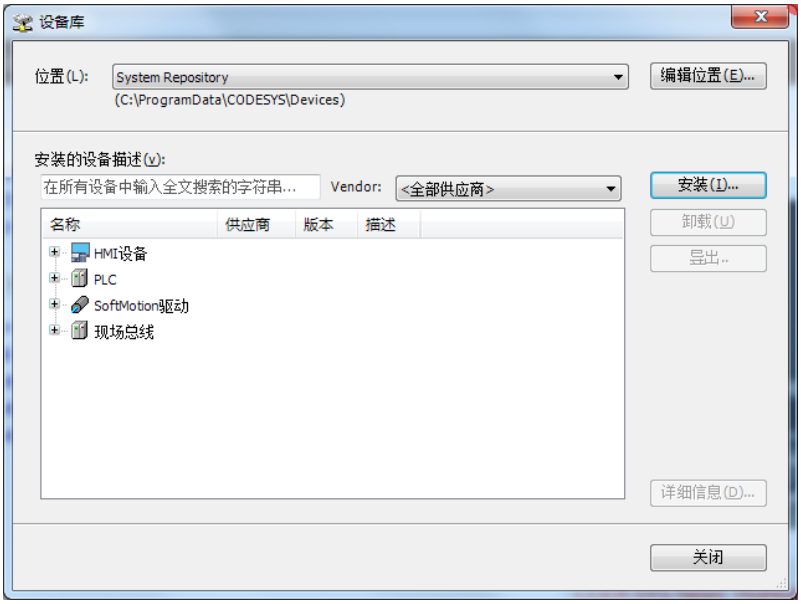
2.2.1 软件中英文设置

英文切换成中文“Tools”→“Options”→“International Setting”，重启软件，配置生效。



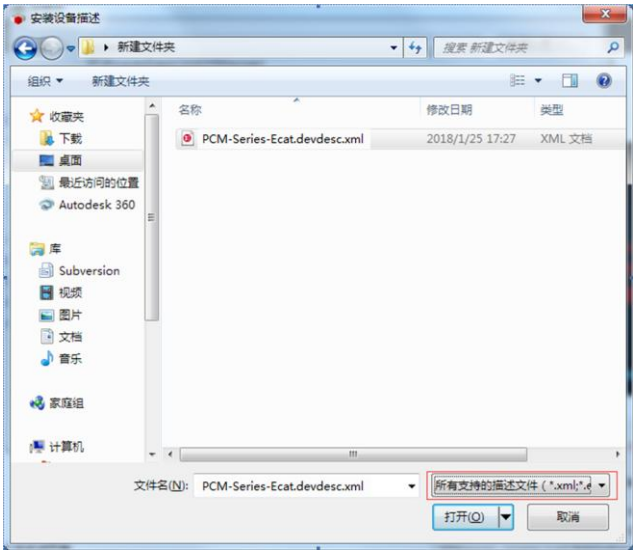
2.2.2 安装 PLC 设备描述文件

PCM-Series-Ecat.devdesc.xml：点击，“工具”→“设备库”→“安装”



在打开的对话框中选择所需的 PCM-Series-Ecat.devdesc.xml，点击“打开”按钮确认选项后，新设备随即添加到“设备库”中的设备目录。

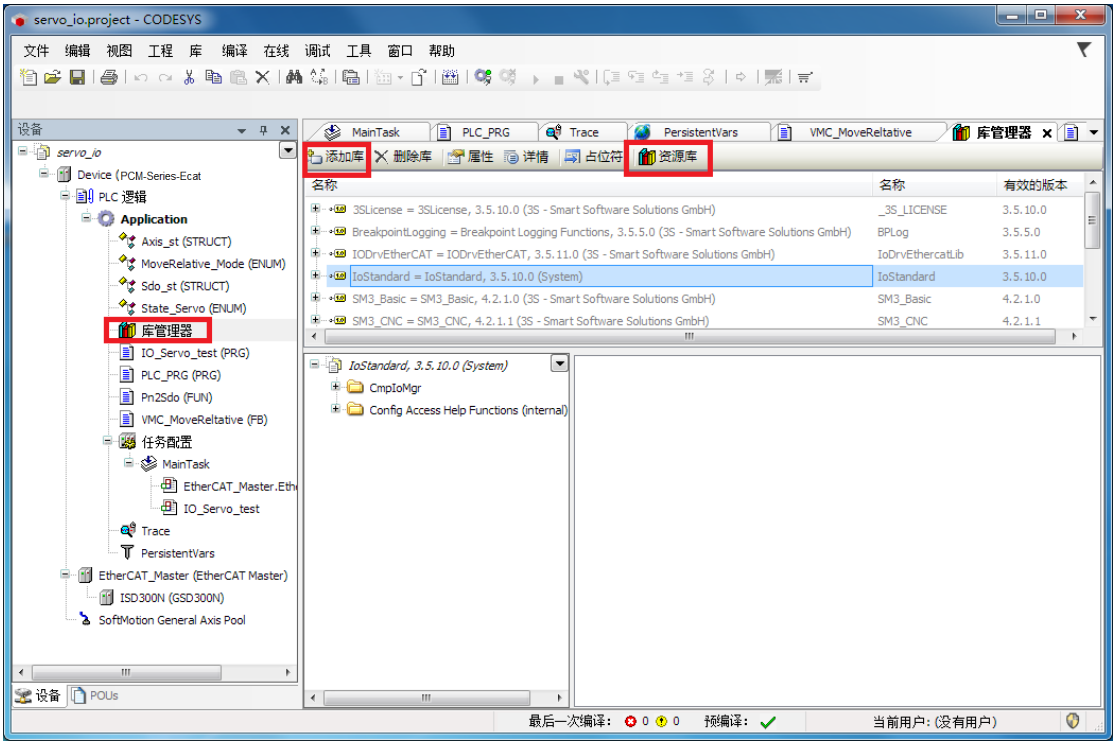
注：Codesys 编程体系中，所有的设备都有一个设备描述文件，此处注意选择右下角红色框中选择“所有支持的描述文件”，用户可以将自己项目中用到的所有设备文件一次添加完成，选中所有要安装的设备文件，点击“打开”即可完成安装。



2.2.3 安装 PLC 库文件

用户可以根据不同的应用需求选择安装所需的库文件。PCM-Series-Ecat.devdesc.xml 提供如下表的库文件，关于库的使用详情参考第五章使用样例。

VM_FreeProtocol.compiled-library	自由口协议 IEC 库
VM_MODBUSMaster.compiled-library	MODBUS 主站 IEC 库
VM_MODBUSSlave.compiled-library	MODBUS 从站 IEC 库
VM_Standard.compiled-library	微秒标准 IEC 库
VM_Common.compiled-library	通用库



Codesys IEC 库安装包含两层含义：

➢ 资源库：本地软件仓库

添加后对整个 Codesys3.5 有效。相当于编程环境的 IEC 库仓库。

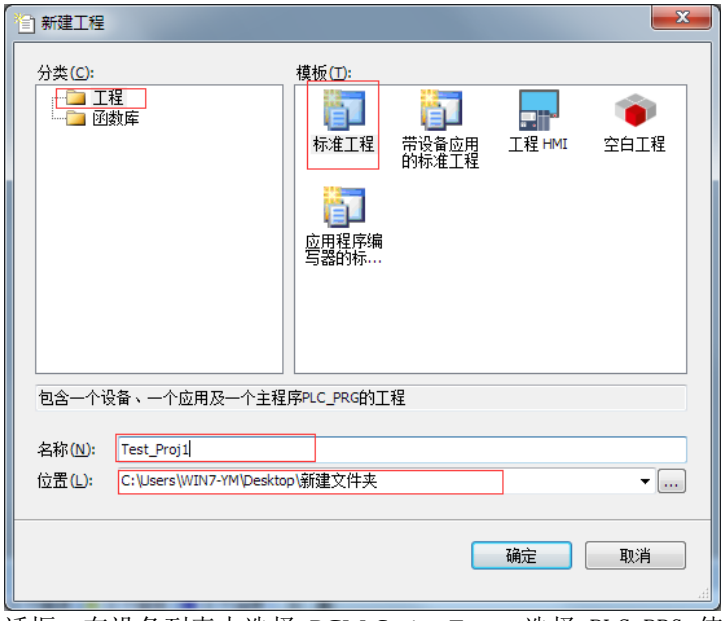
点击“资源库”-“安装”选择微秒提供的所有库并安装。

➢ 库管理器：仅对当前项目有效。

通过“库管理器”-“添加库”选择安装 VM_Standard.compiled-library，在整个项目中即可使用 MODBUSTCP、MODBUS 主站、MODBUS 从站、自由口等协议。

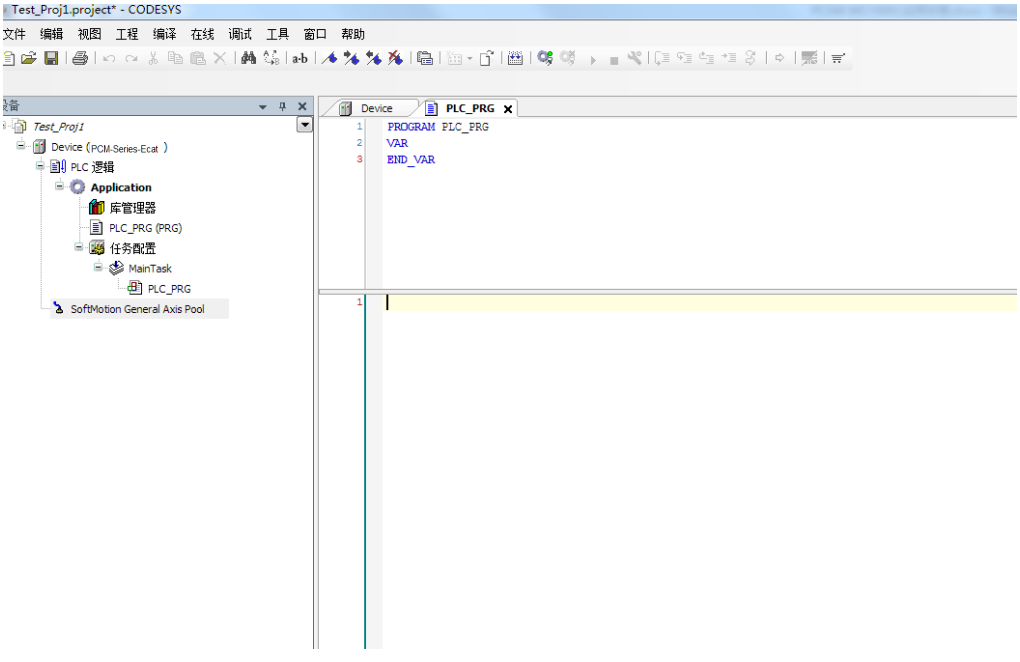
2.2.4 新建一个 PLC 项目

打开 Codesys 软件，依次点击，“文件”-“新建工程”出现如下图对话框
选择“工程”、标准工程、输入 PLC 工程名称、选择项目保存位置。



点击确认，出现如下对话框，在设备列表中选择 PCM-Series-Ecat，选择 PLC_PRG 使用什么语言。此处我们选择 LD 梯形图，点击确认，默认会添加一个采用 LD 编写的程序。

编程界面如下图：



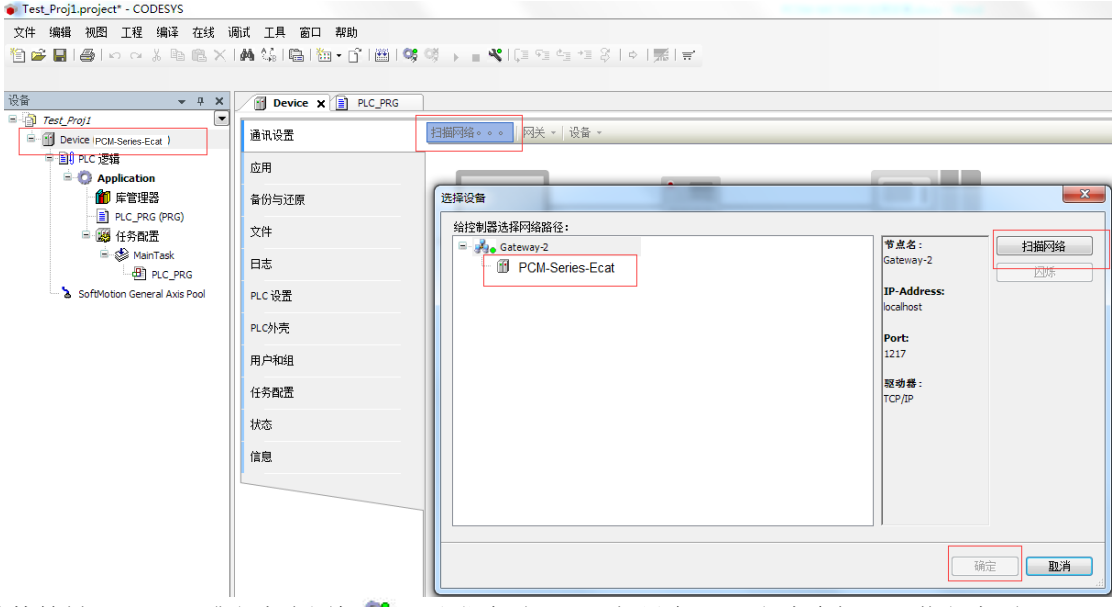
2.2.5 连接到 PLC

将 PC 的 IP 地址和 PLC 的 IP（192.168.1.250）设为同一个网段

将 PC 机的本地连接属性打开，双击 TCP/IP 协议，将“自动获得 IP 地址”更改为“使用下面的 IP 地址”，然后在 IP 地址中填写“192.168.1.X”即可（X 不可以为 250）。

双击左侧 Device（PCM-Series-Ecat）

依次点击“扫描网络”-“ PCM-Series-Ecat”-“确认”

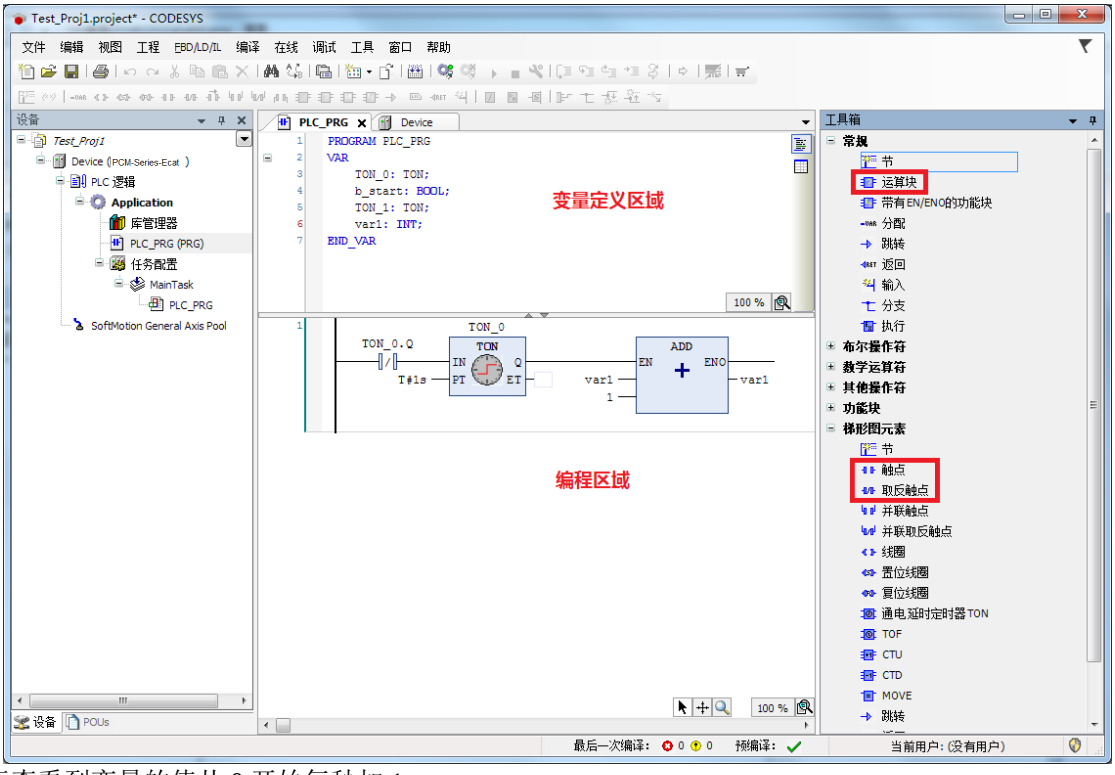


此时按快捷键 Alt + F8 或者点击图标 可以登陆到 PLC，如果有 PLC 程序会提示下载程序到 PLC。

当处于登陆状态下，按 Ctrl + F8 或点击图标 即可退出登陆状态。

2.2.6 编写一个简单程序到下载到 PLC 运行

示例程序功能，使用定时器 1s 间隔加一个变量。



登陆运行查看到变量的值从 0 开始每秒加 1。

3 IEC61131-3 术语及语法介绍

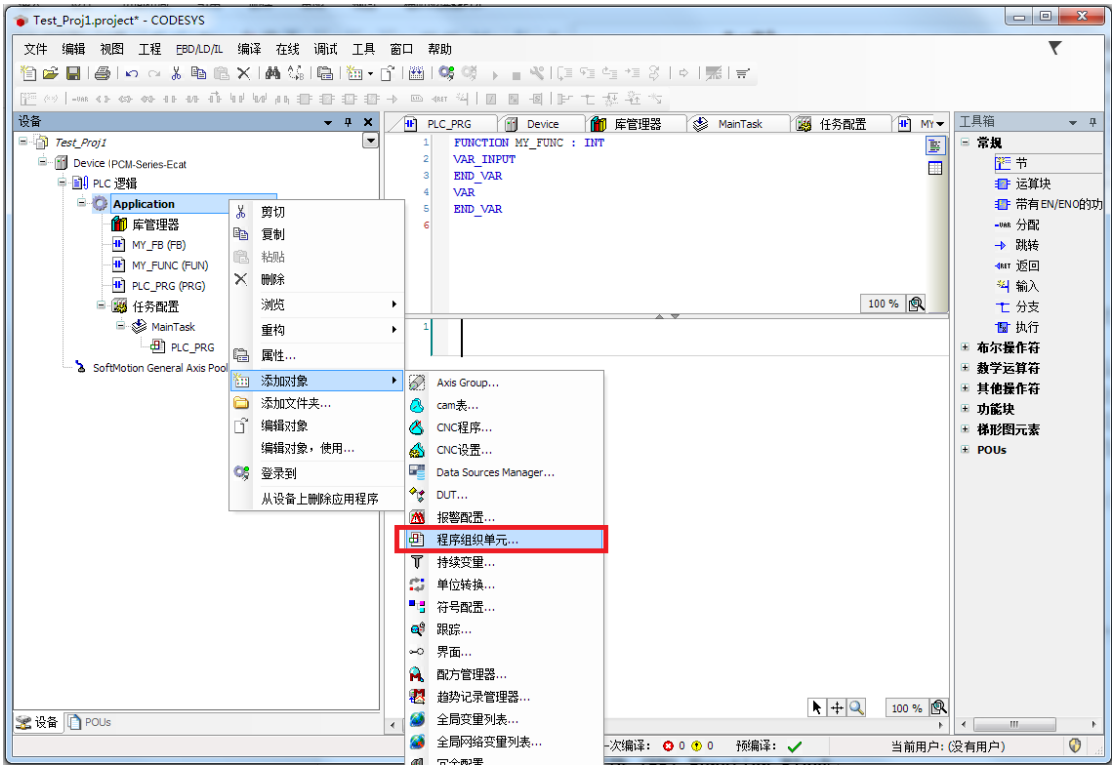
3.1 术语

IEC61131-3 是一个致力于工业自动化编程语言的标准，在世界范围内被广泛支持。

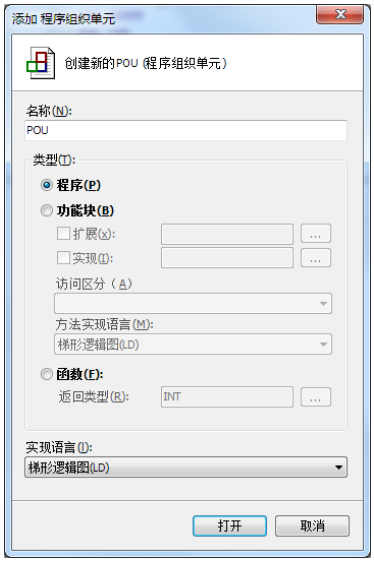
PLCopen 是使 PLC 软件不依靠于供应商，而使其独立于产品的世界组织. 它通过发布和强化 IEC 61131-3 软件开发标准，给工业控制系统用户带来更大的价值，保证软件的兼容性。PLCopen 组织制定了基于 IEC61131-3 框架下常见的运动控制功能，进而拆分成标准的的功能块（例如 3S 的 SoftMotion Basic）。

POU 指程序组织单元，可以是程序（PRG）、功能块（FB）、函数（FUN）

Codeys3.5 创建 POU 时会指示 POU 类型。



点击程序组织单元，会让用户选择是创建程序、还是功能块、或者函数，并且需要选择实现语言。



程序 Program

1> 程序，是根据控制器过程的需要，包含了函数和功能块的一个逻辑组的 POU.
2> 任务调用程序
3> 程序调用功能块和函数
4> 程序调用程序

功能块 (FB) Function Block

1> 程序调用功能块
2> 功能块可调用功能块或函数
3> FB 有输入，输出变量
4> FB 有运算法则:每次 FB 被执行,就是运行一段程序编码

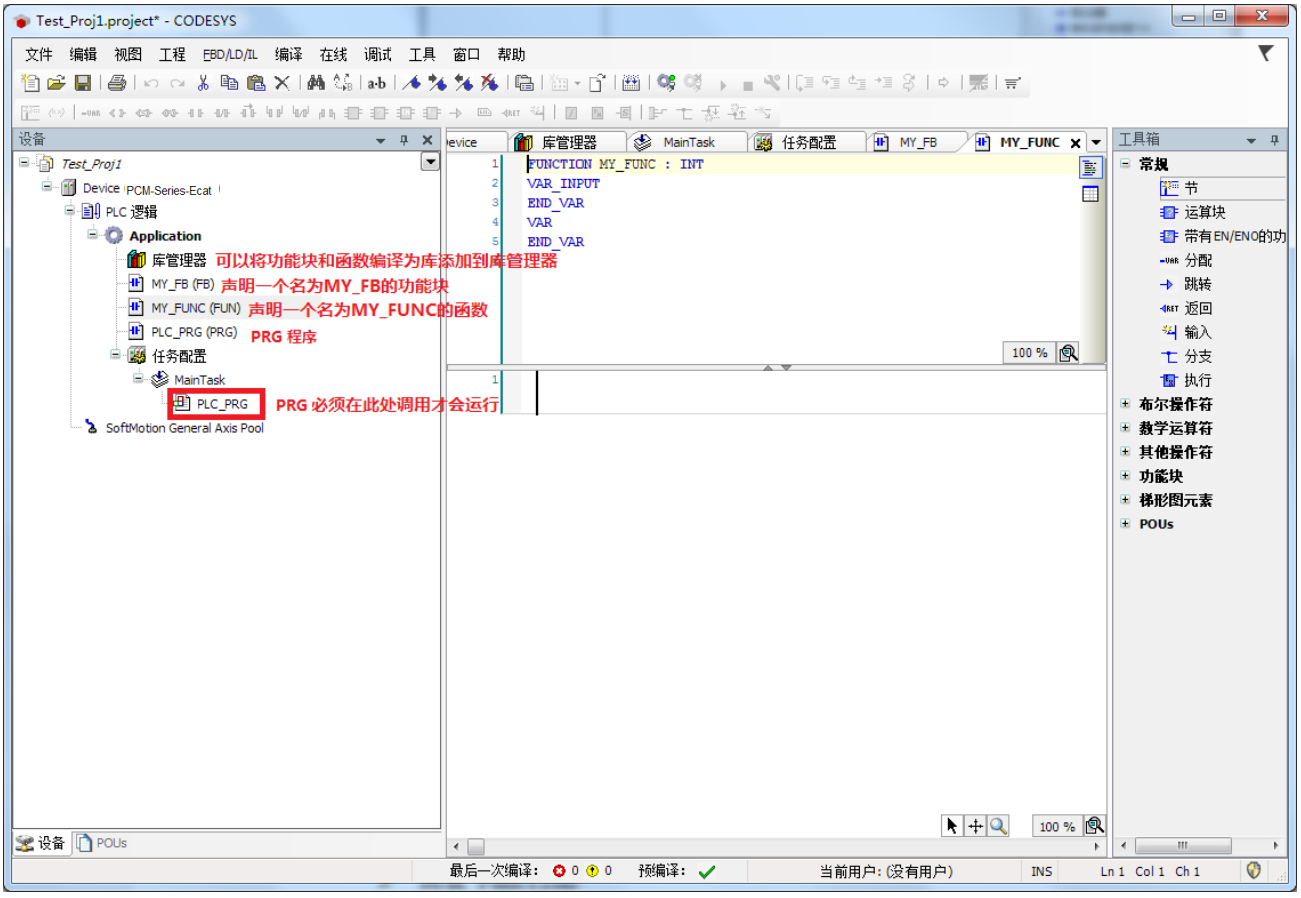
函数 Function

1> 程序或功能块可调用函数
2> 函数有输入变量，和一个输出变量
3> 函数有运算法则:每次函数被执行,就是运行一段程序编码
4> 函数可以调用另外的函数，但不能调用功能块

功能块和函数之间的区别

1> FB: 例程，全部数据分配内存地址
函数: 没有指定的内存分配地址
2> FB: 多个输出变量或没有输出变量
函数: 一个输出变量
3> FB：可调用功能块或函数
函数:可调用函数，但不能调用功能块

各部分在软件中的表示如下图：



3.2 语法及编程

3.2.1 程序运行方式

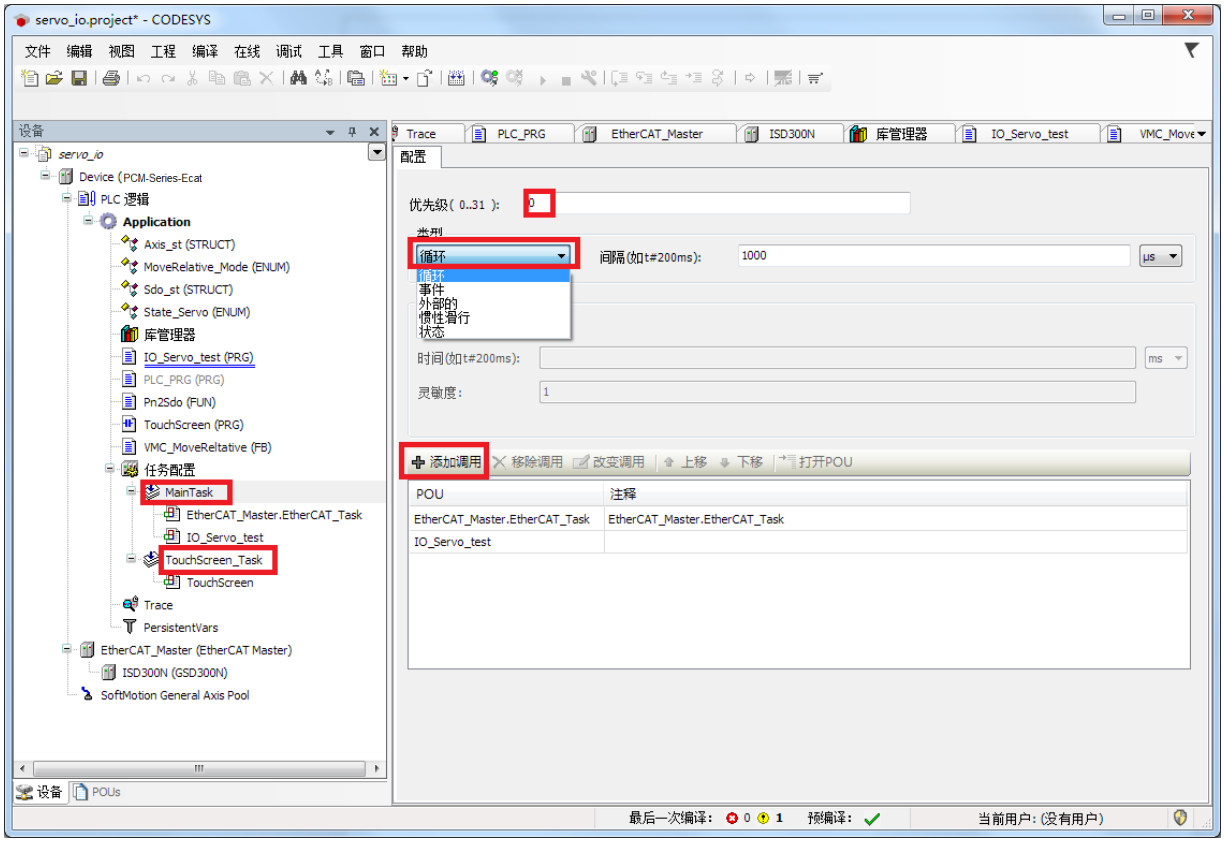
Codesys3.5 中的任务配置如下图所示：

一个任务配置中可以添加多个任务，如下图我们一共添加了两个任务，任务优先级采用默认即可，一般不需修改，MainTask 和 TouchScreen_Task,两个任务分别采用不同的优先级。

MainTask：任务采用循环调用，优先级为 0，循环间隔 1000us，下方的添加调用列表会显示该任务执行哪些程序，在这里由于我们添加了 EtherCAT 主站，所以默认添加 EtherCAT_Master.EtherCAT_Task,该程序主要运行 EtherCAT 主站程序，IO_Servo_test 是我们自己添加的程序。其中包含运动控制相关代码，所以和 EtherCAT 主站处于同一运行周期。

TouchScreen_Task：任务也采用循环调用，优先级为 1，循环间隔此处自动设置为 20ms，这个时间和用户软件处理任务相关，此处由于我们的该任务只做和触摸屏通讯相关处理的实时性要求不高的通讯，也可设置为其他更大的周期，保证通讯的前提下可以更好的节省 CPU 的计算资源

注：EtherCAT 总线的控制在第 4 章详细介绍。



➢ 运动控制器可通过五种扫描方式执行它的任务：循环、事件、外部的、惯性滑行、状态。

1> 循环：按照给定的时间间隔循环执行程序
循环执行是中型 PLC 执行最主要的运行方式，常见的关于运动控制 IEC 任务及 EtherCAT 总线任务就是以此种方式运行。

2> 事件：只要全局变量输入区域事件有一个上升沿时，CODESYS 就开始处理任务
可以将程序中的 BOOL 变量链接到这个事件域，程序中若该变量产生上升沿，则调用相应添加的程序。

3> 外部的：只要在输入域定义的事件发生，CODESYS 就开始处理任务。
支持 DI0 – DI7 8 个外部触发通道，触发信号。
注：详细的外部 DI 使用见 2.6 节。

4> 惯性滑行：工程继续开始或者在一个完整的过程结束之后，CODESYS 将会自动再次开始处理任务。该种方式任务优先级最低。

5> 状态：如果变量值在输入域事件被定义为 TRUE， CODESYS 就会开始处理任务。
例如程序中的 BOOL 变量可以链接到这个状态域，该变量为 TRUE 可以调用程序的执行。

3.2.2 语法基础

变量

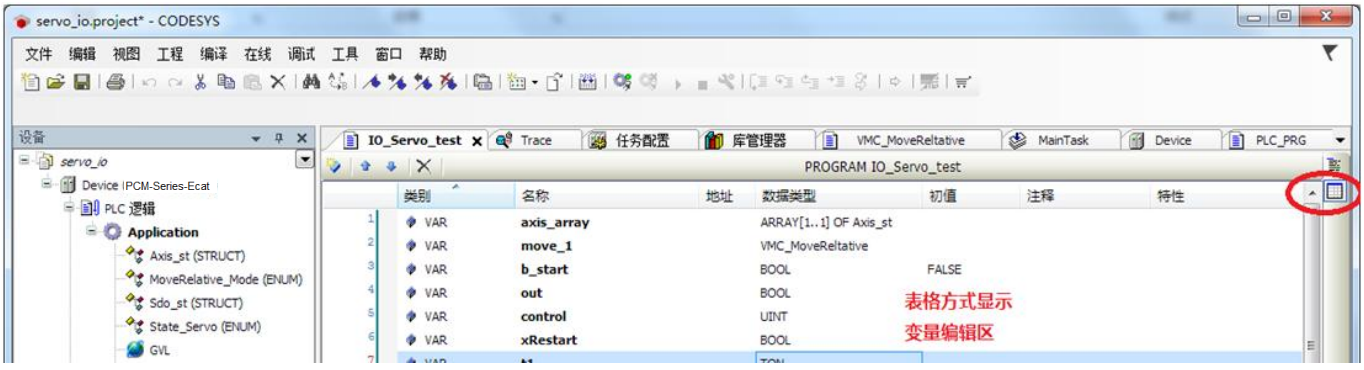
IEC61131-3 编程体系中，通信和数据交换都是基于变量。例如 EtherCAT 过程数据也是通过用户或程序自定义变量链接到通讯参数映射表来实现的，这样在 PLC 程序中就可以访问这些通讯的过程数据了。

同时，编程体系里变量从不同的角度来说又具备不同的属性。

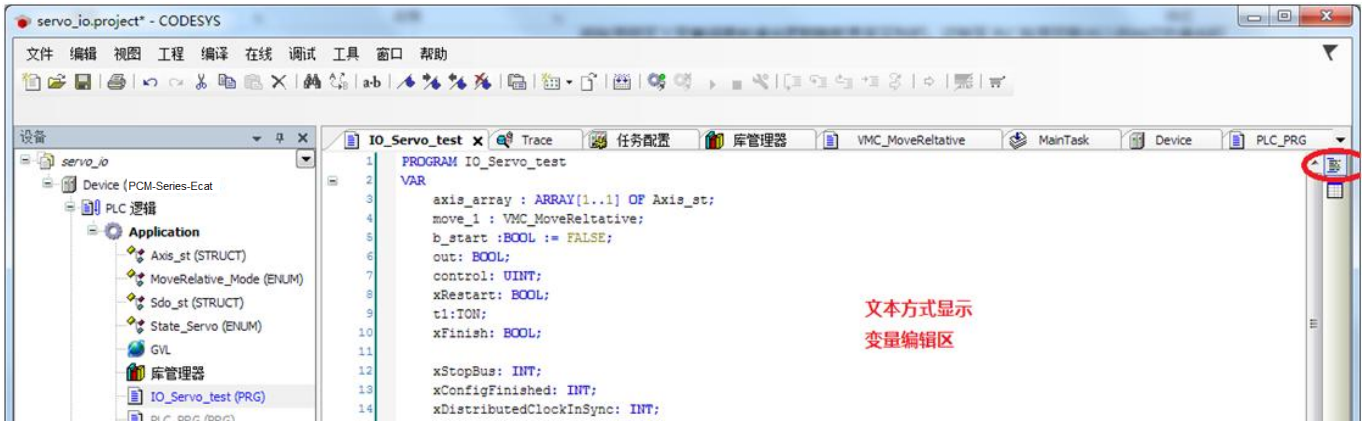
Codesys 中变量显示的两种方式：文本方式和表格方式。

表格方式，点击右侧红色椭圆形圈住的按钮，变量定义切换为表格形式。

表格形式的变量定义清晰、明了，方便编辑，如下图。



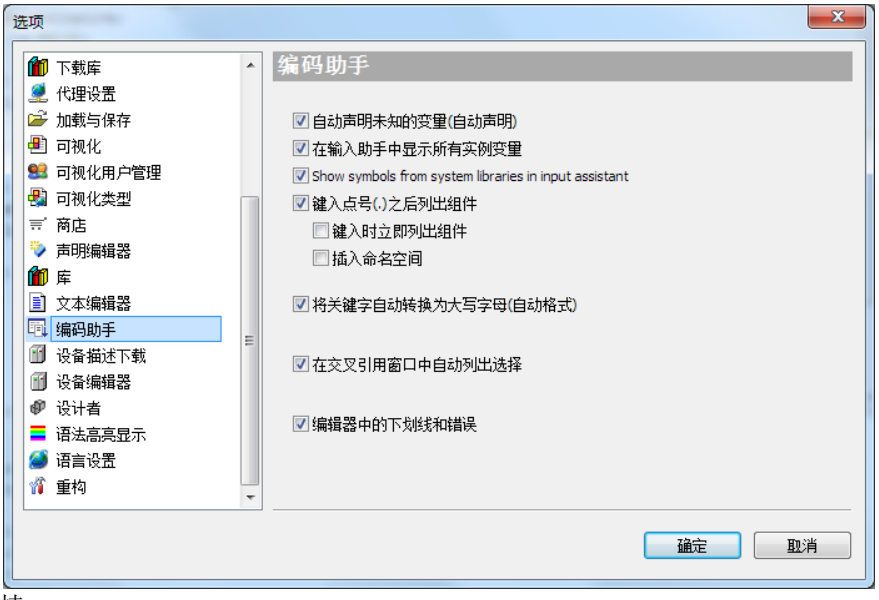
文本形式的变量定义，如下图，灵活，方便，例如对于一些复杂数据结构初始化，如下图。



技巧：

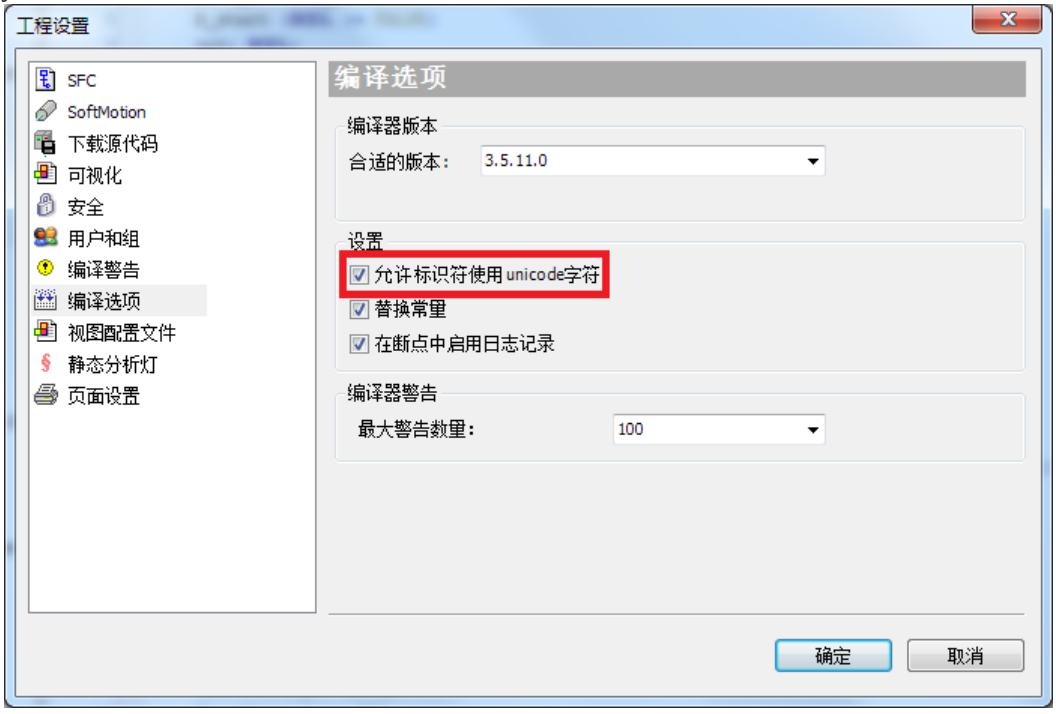
√变量名自动补充：

打开 Codesys 编程软件，依次点击：“工具”-“选项”-“编码助手”，勾选成如下图所示。



√中文变量名支持

打开 Codesys 编程软件，依次点击：“工程”-“工程设置”-“编译选项”-点击下图中红色框选部分。



1>按照内存宽度划分的变量属性

类型名	描述	占用内存宽度（位数）
BYTE	字节	8
WORD	字	16
DWORD	双字	32
LWORD	长字	64
SINT	短整型	8
USINT	无符号短整型	8
INT	整型	16
UINT	无符号整型	16
DINT	双整型	32
UDINT	无符号双整型	32
LINT	长整型	64
ULINT	无符号长整型	64
REAL	单精度浮点数	32
LREAL	双精度浮点数	64

变量的定义并赋初始值：
16 进制、8 进制、二进制：

```
Var1:UINT:= 16#FFFF;
Var1:UINT:= 8#177777;
Var1:UINT:= 2#1111_1111_1111_1111;
```

注：不同进制的数据表示方法：16#，8#，2# 可以使用“_”作为单元分隔符。

变量的定义：

```
Var1:UINT;   (**此处变量只定义未赋初始值，默认会赋初始值 0**)
```

访问变量中的每个位：

```
Var1:UINT:= 16#FFFF;     (**Var1.0 → Var1.15 表示 Var1 从最低位到最高位**)
```

特殊数据类型

STRING 类型可以包含任意字符串，如果没有定义大小，则 CODESYS 默认分配 80 个字符。作为规定，CODESYS 不限制字符串的长度，但是字符串函数处理的长度只能从 1 到 255。如果用一个数据类型过长的字符串进行初始化，则 CODESYS 可以从右边适当地截断该字符串。

带有 35 个字符的字符串声明示例（ASCII 编码格式）：

```
Str:STRING(35) := “这是一个字符串”;
```

WSTRING 类型基于被解码为 Unicode 格式，WSTRING 类型区别于 STRING 类型，其内存要求是每个字符 2 个字节再加上 2 个额外的字节。一个 STRING 只需要一个字节。

```
Wstr:WSTRING:=”这是一个 Wstring”;
```

时间数据类型

数据类型	下限	上限	内存
TIME	0	4294967295	32 位
TIME_OF_DAY(TOD)	0 (00:00:00:000)	4294967295 (23:59:59:999)	32 位
DATE	0 (01.01.1970)	4294967295 (2106-02-07)	32 位
DATE_AND_TIME(DT)	0 (1970-01-01, 00:00:00)	4294967295 (2106-02-07, 06:28:15)	32 位

2>按照作用范围划分的变量属性

局部变量:只在本 POU、FB 范围内有效

```
VAR
END_VAR
```

全局变量：在整个用户工程内有效

```
VAR_GLOBAL
g_var1: INT := 10
END_VAR
```

在 POU 中引用全局变量需要指定全局变量列表的名字：

```
GVL.g_var1
```

Codesys 中添加全局变量：

右键单击“Applation”-“添加对象”-“全局变量列表”

输入全局变量列表名字（加入我们使用自动命名 GVL），创建成功，在此文件里可以添加全局变量。

3>按照输入输出特性划分的变量属性

输入类型变量：

```
VAR_INPUT
END_VAR
```

输出类型变量：

```
VAR_OUTPUT
END_VAR
```

复合数据类型（有输入有输出）：

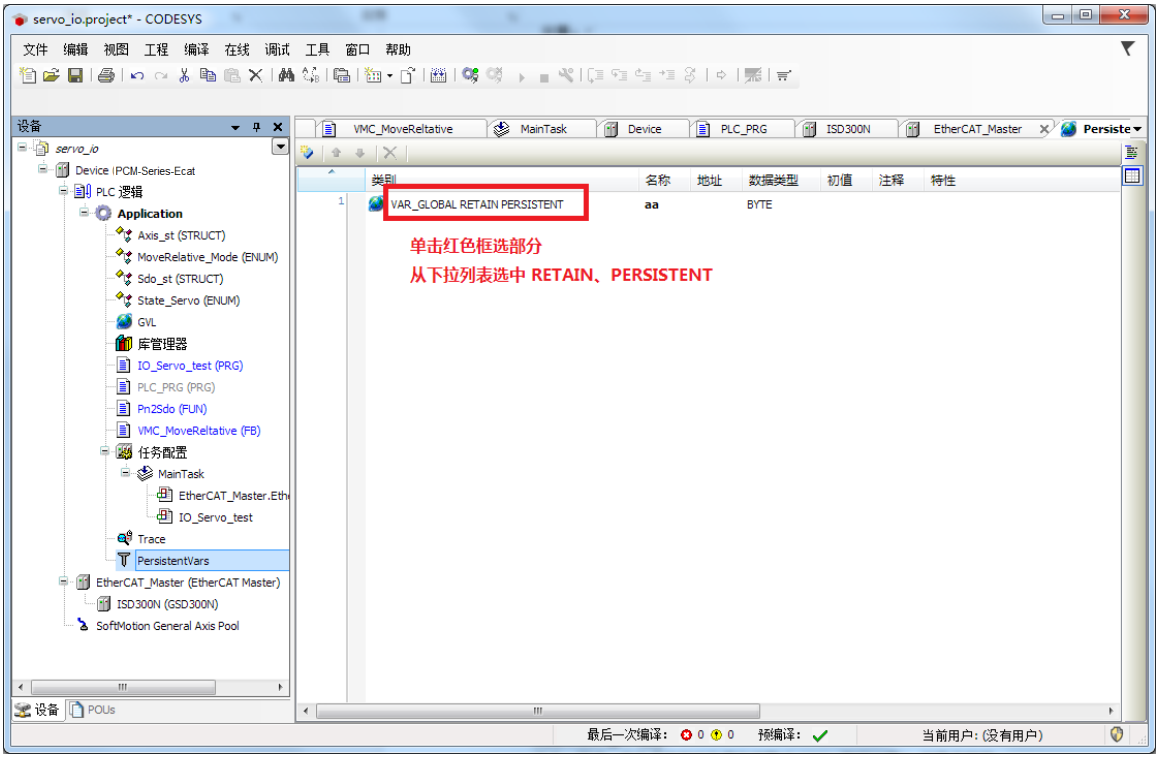
```
VAR_IN_OUT
END_VAR
```

4>按照变量的存储特性划分的变量属性

保持变量：RETAIN
PLC 热复位，该类型变量值不变。
PLC 冷复位，该类型变量值变为初始值。

永久变量：PERSISTENT
PLC 热复位，该类型变量值不变。
PLC 冷复位，该类型变量值不变。
程序擦除，才会销毁该变量。
该类型变量使用最多。
Codesys 软件永久变量使用方法：“Applation”右键单击-“添加对象”-“持续变量”

弹出菜单提示输入文件名，此处我们使用默认的文件名 PersistentVars。
注意：保持变量列表中保持变量类别需要按下图进行设置，否则变量不具备保持属性。



进阶：数组、结构体、枚举、联合、子范围、指针。
参考 Codesys 软件帮助文档，或微秒提供的学习例程。

3.3 语言特征

主要分为两个大类：文本语言、图形语言

运算符

IL，FBD，LD	ST
ADD	+
SUB	-
MUL	*
DIV	/
MOD	MOD
EQ	=
NE	<>
GE	>=
GT	>
LE	<=
LT	<

3.3.1 结构文本（ST - Structured Text）

是一种高级语言 (类似 PASCAL)，可以进行复杂公式的(赋值命令)编写，具备分支和循环流控指令，简洁、高效，方便实现算法（例如 PID 控制、数学计算）、抽象工艺等。封装函数功能块方便灵活。

ST 运算符结合性

操作	符号	约束力
括号	（）	最强
函数调用	函数名（）	
求幂	EXPT	
按位取反	NOT	
乘法	*	
除法	/	
取模	MOD	
加	+	
减	-	
与	AND	
异或	XOR	
或	OR	最弱

赋值操作符：

赋值符号左边是一个操作数（变量，地址）， ":="右边是赋予它的表达式的值，例如：

```
var1 := var2+100;
```

在运算结束后，变量 var1 就变为 var2 的值加上 100.

条件执行（IF）

```
IF <Boolean_Expression1>Then
<instruction(s)>;
ELSIF <Boolean_Expression2> Then
<instruction(s)>;
ELSIF <Boolean_Expression3> Then
<instruction(s)>;
ELSE
<else_instruction>;
END_IF
```

如果布尔运算表达式<Boolean expression1>返回 TRUE，只有 if 指令部分执行，其它部分不执行。否则，布尔运算表达式从<Boolean expression 2>开始，一个接一个的计算，直到某个布尔表达式返回为 TRUE，直到计算到表达式 3，如果没有任何一个布尔运算表达式返回 TRUE，那么只计算 ELSE 下的指令。

条件执行（CASE）

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

```
CASE <Var1>OF(**Var1 为整数类型**)
<ValueA>:
<Instruction1>;
<ValueB>:
<Instruction2>;
<ValueC,ValueD,ValueE>:
<Instruction3>;
<ValueF..ValueK>:
<Instruction4>;
<ValueN>:
<Instruction5>;
ELSE
ELSE Instruction;
END_CASE
```

CASE 指令根据下面的模式来处理：

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

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

如果有多个变量值要执行同一个指令，那么这些条件执行一个公共指令 **Instruction3**。

如果对于一个变量在一个值的范围内执行同一个指令 **Instruction4**，那么在初始值和最后值之间用两个句点隔开，所以可以规定公共条件。

循环（FOR）

```
FOR <Var1>:= <INIT_Value>TO <END_Value>BY <Stepwidth>
DO
<Instruction(s)>;
END_FOR
```

只要计数器 Var1 不大于 END_Value，指令 Instructions 就一直执行，在执行 Instructions 之前首先检查计数器的值，如果 INIT_Value 比 END_Value 大的话 Instructions 将不在执行。当 Instructions 执行后，Var1 通常要增加一个 Step size， Step size 可以是任何整型值，如果没有 Step size，它将设置为 1，当 Var1 大到一定值时，循环结束。

循环（WHILE）

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

```
WHILE <BooleanExpression>DO
<Instruction(s)>;
END_WHILE
```

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

注：如果一直重复执行(死循环) PLC 将会出现异常。

3.3.2 指令表（IL - Instruction List）

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

限定符：

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

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

N 用于其它情况：取操作数的反（不包括累加器）。

下面是操作符和它们可能的限定符以及相关的意义：

操作符及限定符意义	
LD N	使当前的值等于操作数
ST N	在操作数的位置保存当前值
S	当前的值为 TRUE 时，把布尔型操作数置为 TRUE
R	当前的值为 TRUE 时，把布尔型操作数置为 FALSE
AND N，（	位逻辑运算符号“与”
OR N，（	位逻辑运算符号“或”
XOR N，（	位逻辑运算符号“异或”
ADD （	加法
SUB（	减法
MUL（	乘法
DIV（	除法
GT（	>
GE（	>=
EQ（	=
NE（	<>
LE（	<=
LT（	<
JMP CN	跳转到标号
CAL CN	调用程序功能块
RET CN	离开 POU 并返回到调用的地方
）执行延时操作	

3.3.3 功能块图（FBD - Function Block Diagram）

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

3.3.4 梯形图（LD - Ladder Diagram）

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

梯形图包含了一系列的网络，左右两边各有一个垂直的电流线，网络图仅限制于左右两母线之间的范围内，在中间是由线圈触点和连接线组成的电路图。每一个网络包含左边的一系列触点，这些触点根据布尔变量值的 TRUE 和 FALSE 来传递从左到右的开和关的状态。每一个触点是一个布尔变量，如变量值为 TRUE，电路从左到右通过连接线就连通。否则右边接收到“关”的值。

触点

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

这些状态与布尔变量 TRUE 和 FALSE 相一致。布尔变量属于每一个触点。如果变量值为 TRUE，那么状态可以通过连接线从左传到右边。否则，右边接收到的是“断开”。触点可以并联使用，其中的一个并联分支必须传递“开”状态时，并联分支才能传递“开”。或者触点串联连接，此时，触点必须传递“开”状态时，最后的触点才传递“开”，这些与串并联连接的电路一致。

线圈

在梯形网络图的右边有一些所谓的线圈，它们用（）表示，并且只能通过水平线来连接。线圈传递从左到右的连接状态，并且复制状态到布尔变量中，可以描述入口线的状态为“开”（对应布尔变量的 TRUE）或者“关闭”状态（对应布尔变量的 FALSE）。梯形图中的功能块可以在网络图中添加功能块和程序，但它们必须具有布尔型值的输入和输出，并且可以象触点那样用在梯形图的左边。

触点的设置/复位

触点可以被定义为设置/复位触点。设置触点在触点符号中用“S”表示，它从不覆盖相应的布尔变量中的 TRUE 值，也就是说，如果变量一旦设置为 TRUE，它将保持状态不变。复位触点用“R”来表示，它从不覆盖相应布尔变量中 FALSE 值，如果变量已经设置了 FALSE，它将保持这种状态不变。当使用梯形图时候，可以用触点开关的结果来控制其它的 POU。一方面可以用线圈把结果输出给全局变量，这个全局变量可以在其它的地方使用。也可以通过引入一个带 EN 输入的 POU 来直接在梯形网络图中插入可能的调用。这些 POU 是完整的正常的操作数、功能、程序或功能块。它们都有一个附加的输入标志符 EN。EN 输入是一个布尔变量，只有当 EN 值为 TRUE 时，带有 EN 输入的 POU 才会被计算。

3.3.5 顺序功能图（SFC - Sequential Function Chart）

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

步（Step）

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

有两种类型的步：

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

动作（Action）

一个动作可以包含一系列的指令表或结构化文本指令，功能模块图或梯形图许多的网络，或者又包含另外顺序功能图。在简单步中，动作经常是和步连接在一起的，为了能编辑一个动作，在步上双击鼠标或选择此步，再选择菜单命令“Extras”“Zoom Action / Transition”。另外，每一个步中只允许一个输入或输出动作。IEC 步的动作是附加在顺序功能图- POU 内的对象管理器中，通过双击或者在它的编辑器中按 Enter 键可以加载它。也可以通过“Project”“Add Action”来创建一个新的动作。可以为一个 IEC 步分配最多九个动作

进入和退出动作

可以额外的为一个步添加一个进入和退出的动作，在一个步激活后，一个进入动作只能执行一次。退出动作只在步失效之前执行一次。带进入动作的步左下角一个“E”来表示，退出动作用右下角的“X”表示。

转换/转换条件

在步和步之间有所谓的转换。

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

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

激活步

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

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

IEC 步

在顺序功能图中可以使用标准的 IEC 步。为了能使用 IEC 步，必须在工程文件中联接 Iesfc.lib 库文件。一个 IEC 步中不能分配超过九个动作， IEC 的动作不象简单步那样固定地作为输入或输出到某个步的动作，而是和步分开存储并且能够在一个 POU 中重复使用多次。因此，它们必须用命令“Extras Associate action”和单个步联系在一起。除了动作，布尔变量也能分配给步。能够使用所谓的限定词来控制激活和未激活的动作和布尔变量。可能有时间延迟，如果一个动作依然激活中，而下一个步已经开始处理了，通过限定词 S（设置），可以取得并发的过程。随着每一个顺序功能模块的调用，相关联的布尔变量被设置或复位，也就是说，随着每一次调用，这个值将在 TRUE 到 FALSE 之间来回变化。IEC 步的关联动作在步右边的两长方形中表示，左边的区域包含了限定词，可能带有时间常量，右边的区域包含了动作名和各自的布尔变量名。

限定符

为了关联动作和 IEC 步，用到下面的限定词

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

注意：当一个动作失去激活时，它会再执行一次。这就是说每个动作至少执行两次。

可选分支

在 SFC 中可以定义两个或两个以上的可选择分支。每一个分支的开始和结束必须带有一个转换。可选分支可以包含并行的分支和其它的选择分支，一个可选分支开始于一个水平线并终止于一个水平线（选择结束），或是一个跳跃。

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

并行分支

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

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

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

更详细内容请参考 Codesys 帮助文档中参考编程章节介绍。

4 EtherCAT 总线运动控制介绍

4.1 总线原理及术语简介

EtherCAT 总线系统中主要分为两种设备，EtherCAT 主站设备（PC5M）、EtherCAT 从站设备（带 EtherCAT 从站的伺服、IO 模块、计数模块….）。

EtherCAT 总线系统从启动到和所有从站同步需要一定的时间，这个时间值和系统中连接的从站设备数量、主站和 PLC 任务周期运行时间值的设置都有关系，PLC 中有相应的功能块或函数、变量可以供编程者（参考 4.4 节）获取总线中各个设备的状态。通常情况下我们的主站和运动控制程序一起运行，程序逻辑因此需要关心主站和各个从站的状态。从而进一步编写时序严谨的程序。

4.1.1 通信模式

在实际自动化控制系统中，应用程序之间通常有两种数据交换形式：时间关键的过程数据（PDO）和非时间关键的邮箱数据(SDO)。时间关键表示特定动作必须在确定的时间窗口内完成。如果不能在要求的时间窗口内完成通信，则可能引起控制失效。时间关键的数据通常周期性发送，称为周期性过程数据通信。非时间关键数据可以非周期性发送，在 EtherCAT 中采用非周期性邮箱（mailbox）数据通信。

EtherCAT 从站支持三种同步模式：

自由运行

在自由运行模式下 EtherCAT 通信和应用程序彼此独立运行。

同步于数据输入或输出事件

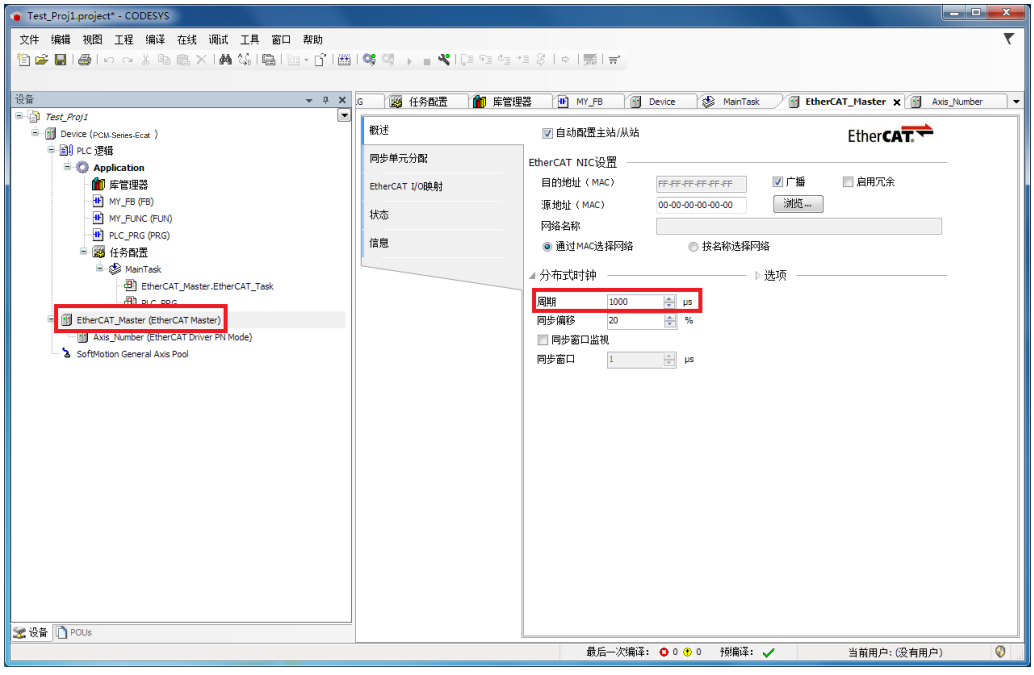
从站应用程序同步于数据输出事件，如果没有数据输出则使用数据输入事件作为同步信号。

同步于分布时钟（DC）同步事件（EtherCAT 伺服使用该模式）

本地周期由 SYNC 事件触发。主站必须在 SYNC 事件之前完成数据帧的发送，为此也要求主站时钟也要同步于参考时钟。

我们绝大多数情况下只会用到第三种模式。

主站同步周期设置：对于常见的使用分布式控制方案的系统而言，主站周期可以设置为 1ms，主站的运行周期和 PLC 的运动控制任务一致，且同步运行。主站的周期时间设置如下图所示



4.1.2 PDO 和 SDO 的使用

PDO（Process Data Objects）过程数据对象

过程数据在每个 EtherCAT 总线周期都会实时更新，因此和运动控制相关时间敏感的参数需要归类为过程数据。

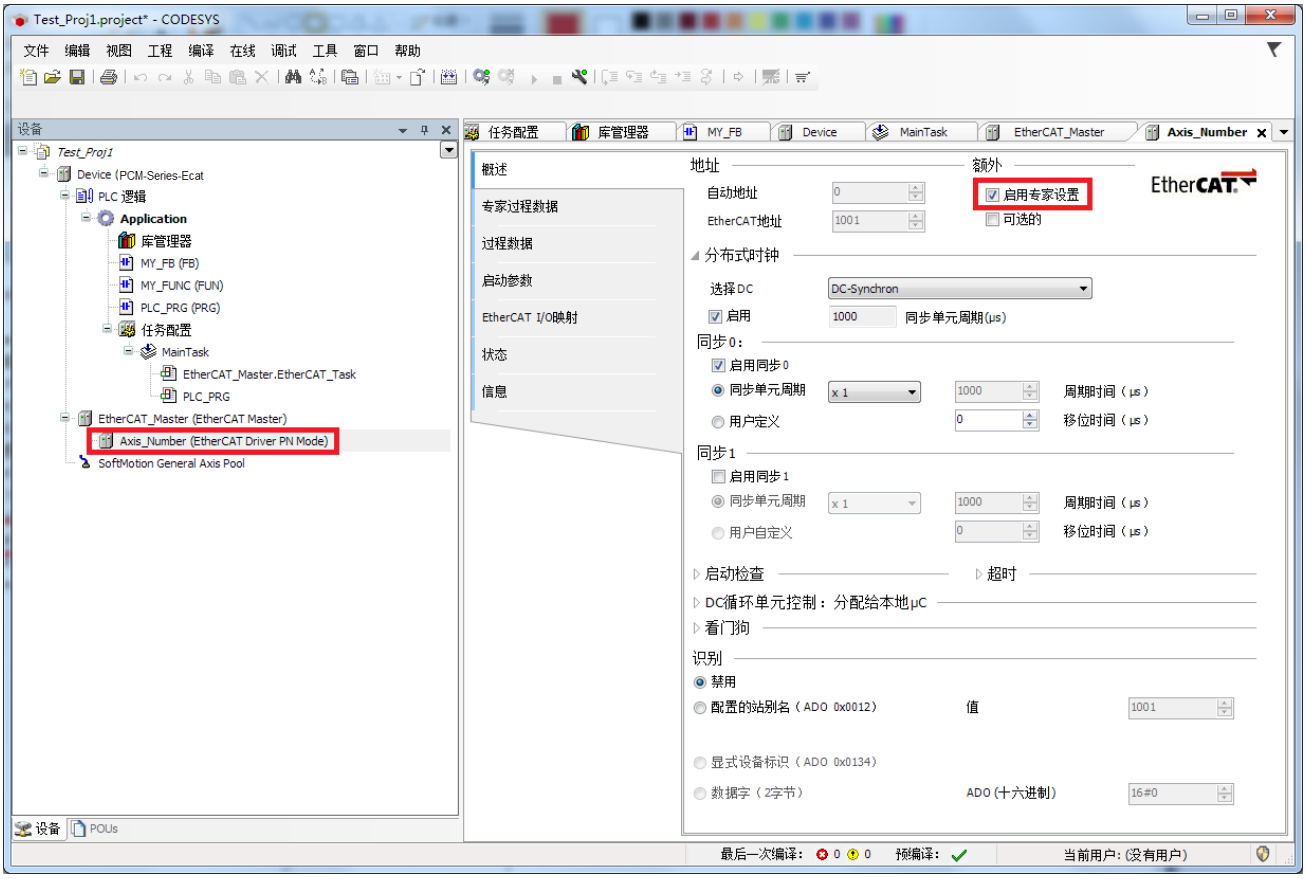
在 Codesys3.5 编程软件中，当用户添加了主站和从站设备。用户就可以访问或者选择使用哪些过程数据了。过程数据组的选择可参考 4.3.1。

过程数据组的编辑

某些时刻，固定的 PDO 参数无法满足需求，此时用户需要自己添加过程数据到对应的组（哪些参数可映射为过程数据由从站设备决定），下面以微秒从站伺服的过程数据编辑来介绍 Codesys3.5 中修改过程数据的方法。

注意：可编辑的过程数据组只有 16#1600（输出）和 16#1A00（输入），过程数据都是单向的。

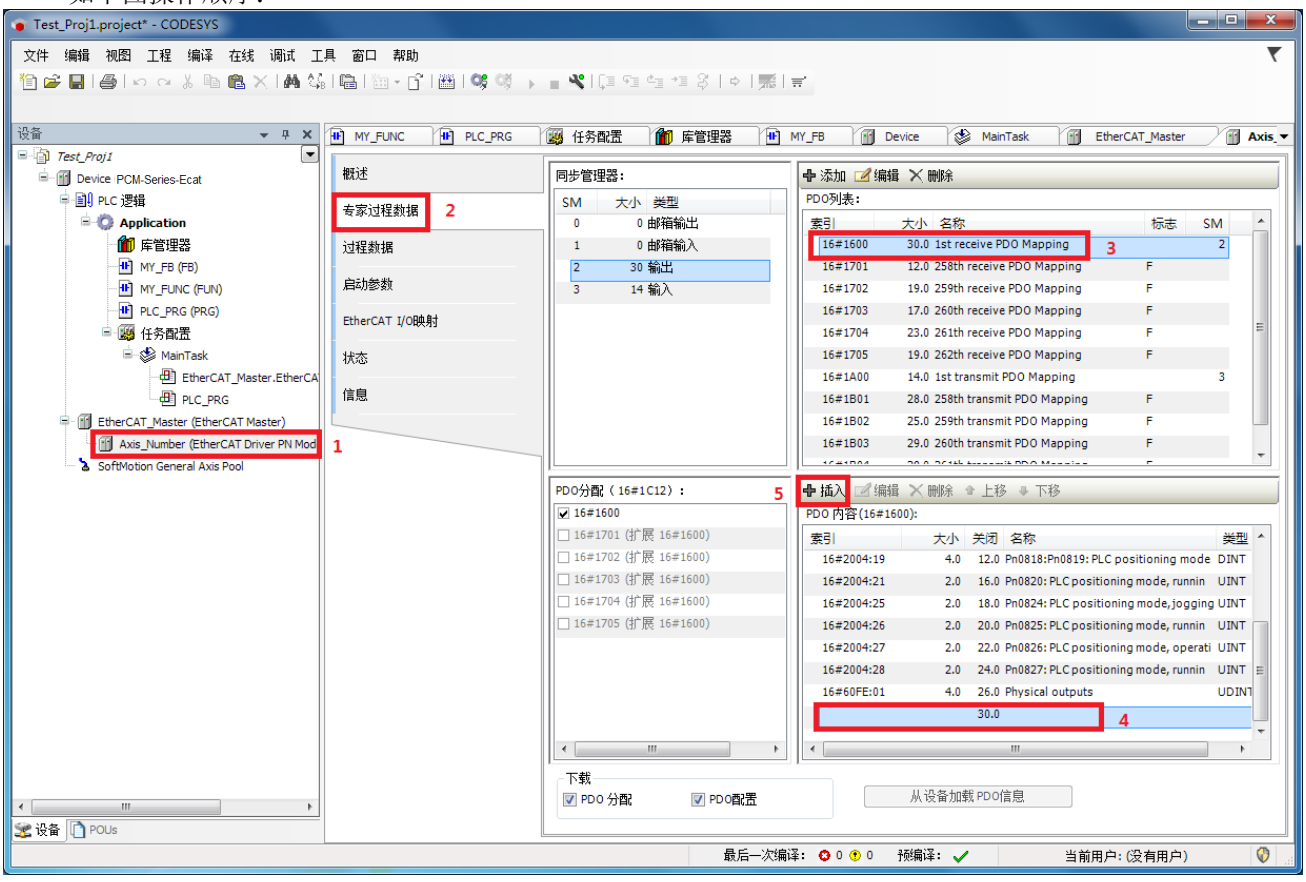
1> 启用专家设置



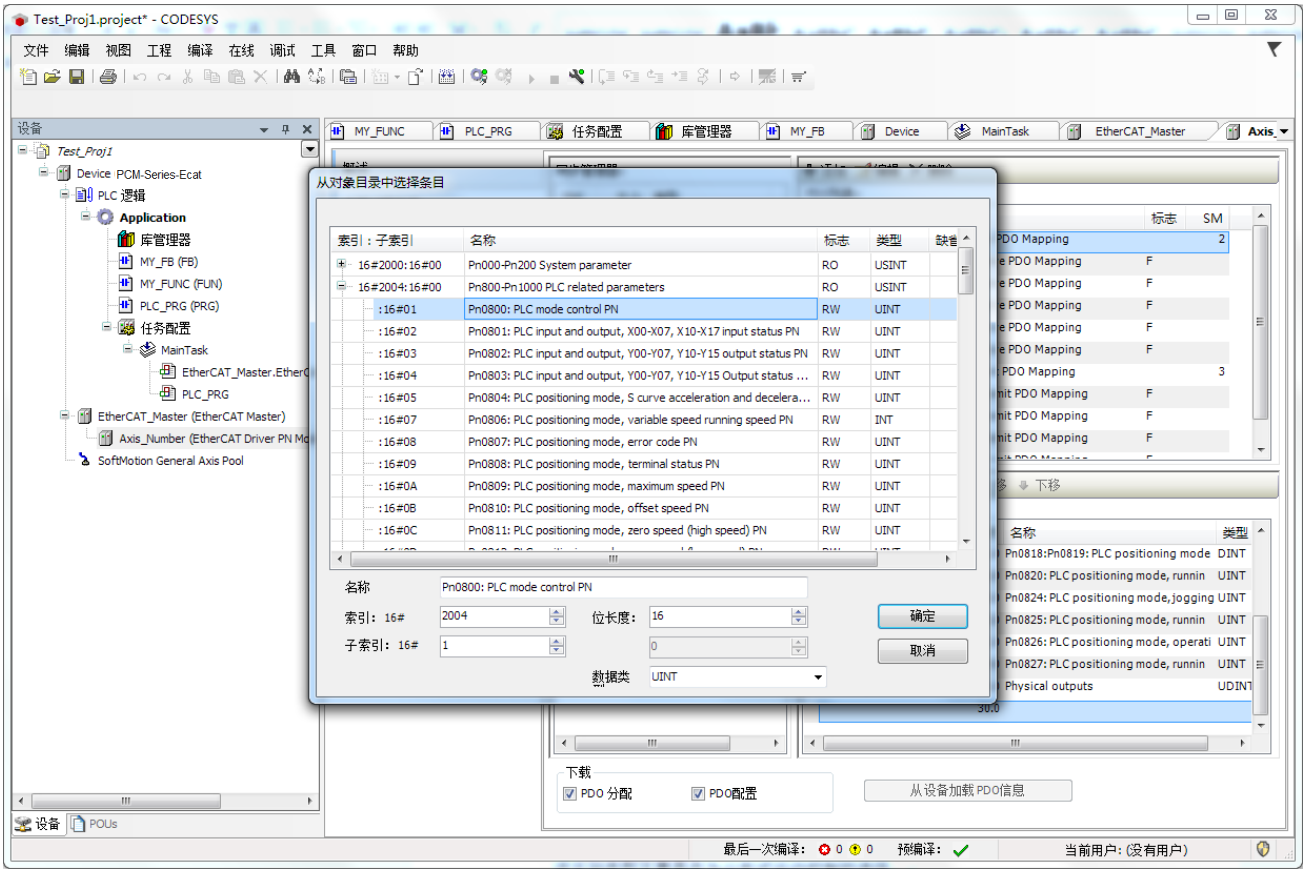
4

2> 专家过程数据

如下图操作顺序：



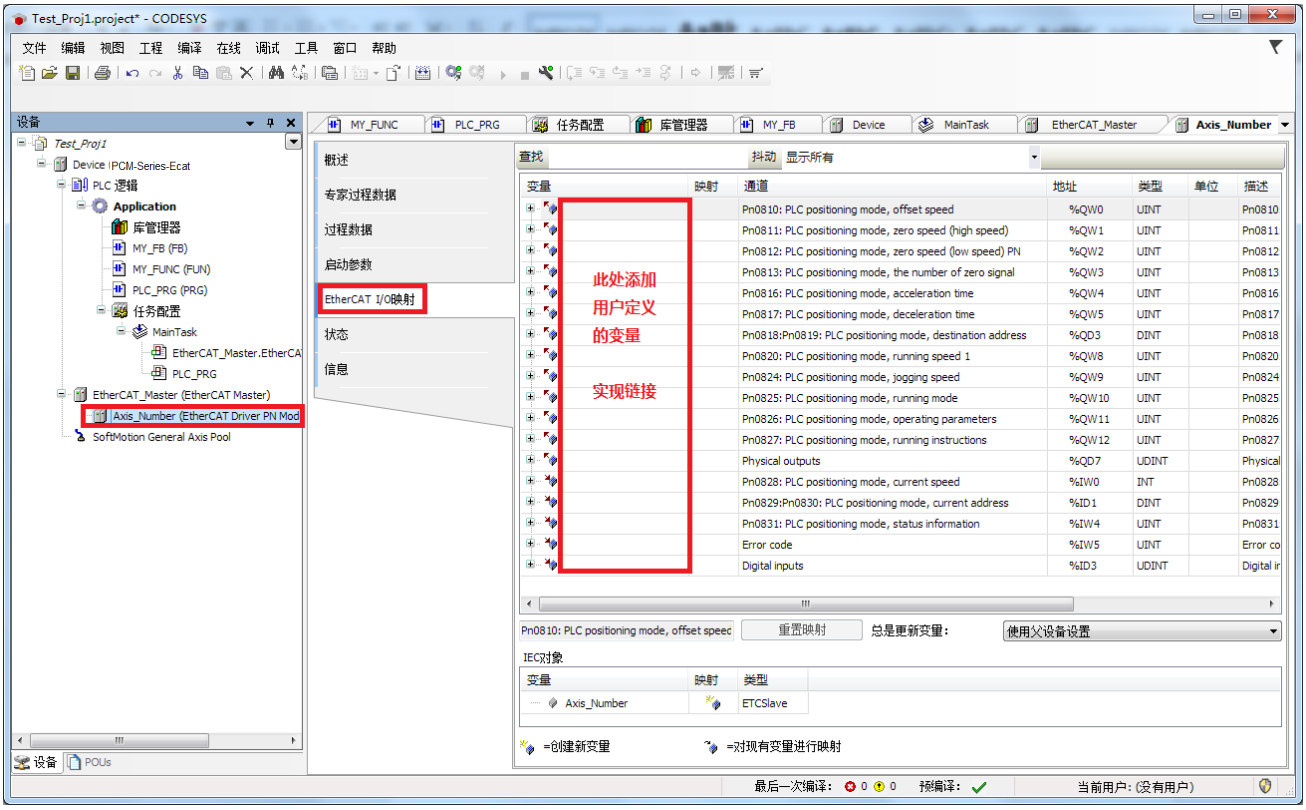
点击“插入”后出现如下图的表格，在其中选择想要添加到的组中。



EtherCAT IO 映射

编辑完过程数据后，还需要用户实现变量的链接，只有完成链接后，才可以在 POU 中使用该变量。

链接变量需要注意变量类型的匹配。



SDO（Service Data Objects）

SDO 的使用相对简单，前面章节安装的库中已经对微秒从站伺服实现了封装，用户只需要调用库里面的功能块即可实现对伺服内部参数的修改。实际应用中，我们一般将不需要周期性修改的参数以 SDO 方式读写参数。

4.1.3 EtherCAT 主站、从站的控制查询

实际项目中，尤其是在使用分布式方式控制各个从站伺服轴时，用户 PLC 程序必须确保总线中的各个设备状态可知可控，以保证正确的操作时序。

例如一个正确的操作流程如下：

1. PLC 程序检测到各个从站已经进入到同步状态
2. PLC 程序开始执行分布式控制运动控制相关功能（使用 SDO、PDO 通讯），若此处所有从站还未建立到同步状态，则 PLC 程序执行就会出现错误。
3. 监测各个伺服报警代码，如果监测异常则停机

从站状态监测

```
bstart : BOOL;
start_prepare : BOOL;

pSlave := Ethercat_Master.FirstSlave;
start_prepare := TRUE;

WHILE pSlave <> 0 DO

pSlave^();

IF pSlave^.wState = ETC_SLAVE_STATE.ETC_SLAVE_OPERATIONAL THEN
 bstart := TRUE;
ELSE
 bstart := FALSE;
END_IF
start_prepare := start_prepare AND bstart;
pSlave := pSlave^.NextInstance;

END_WHILE
```

上面的代码可以实现所有从站状态的扫描检测，若 start_prepare 变量变为 TRUE，则表示所有从站已经切换到运行状态。

主站状态监测及主站重启

使用下面代码可以实现主站的监测和重启。

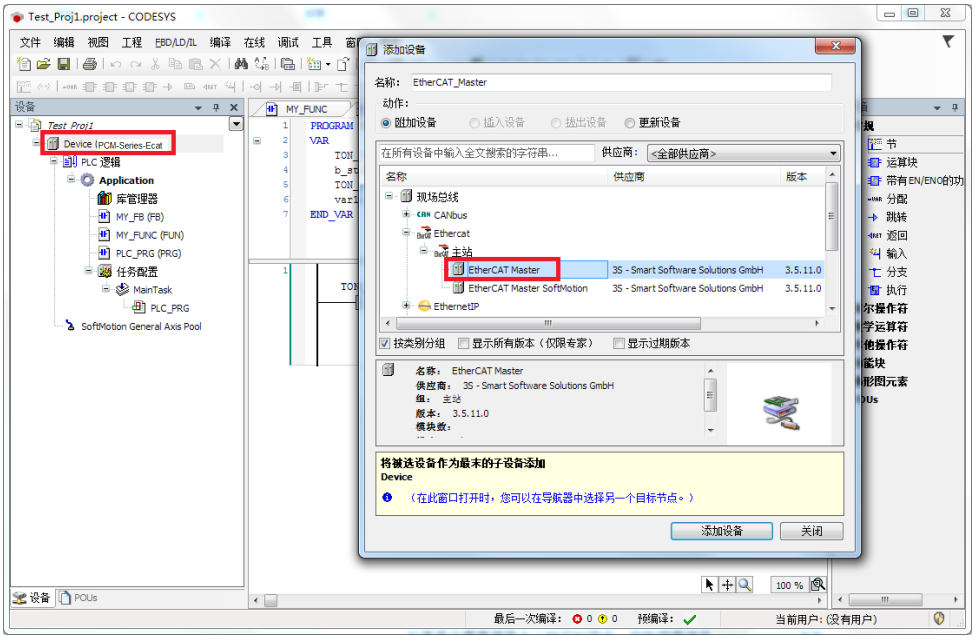
```
EtherCAT_Master(
xRestart:= bRestart, //主站在上升沿重启
xStopBus:= FALSE,    //TRUE : 主站停机  FALSE：主站正常工作
xConfigFinished=> ,
xDistributedClockInSync=> bSync, //数据输出 TRUE：主站同步时钟和第一个从站实现同步
xError=> ,                       //主站出现错误
xSyncInWindow=> ); //重启后仍然需要判断每个从站状态
```

4.2 Codesys3.5 添加主站

需要在项目工程里添加主站设备才可以使用从站设备。
主站设备包含两种

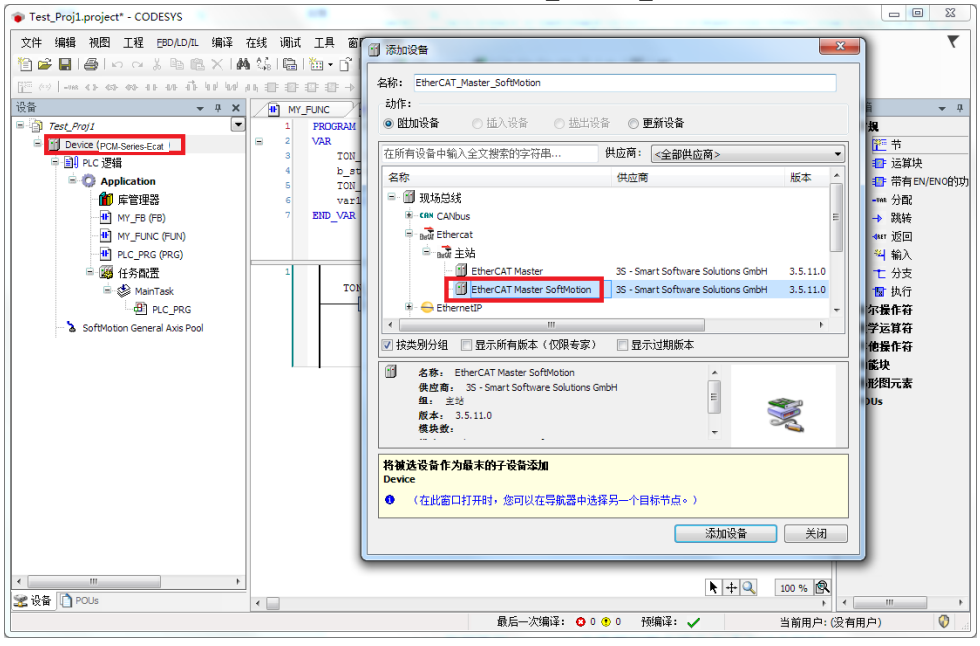
4.2.1 EtherCAT_Master

该主站类型主要是作为分布式运动控制的选择。
选中“PLC 设备”- 右键单击 -“添加设备”-“EtherCAT Master”



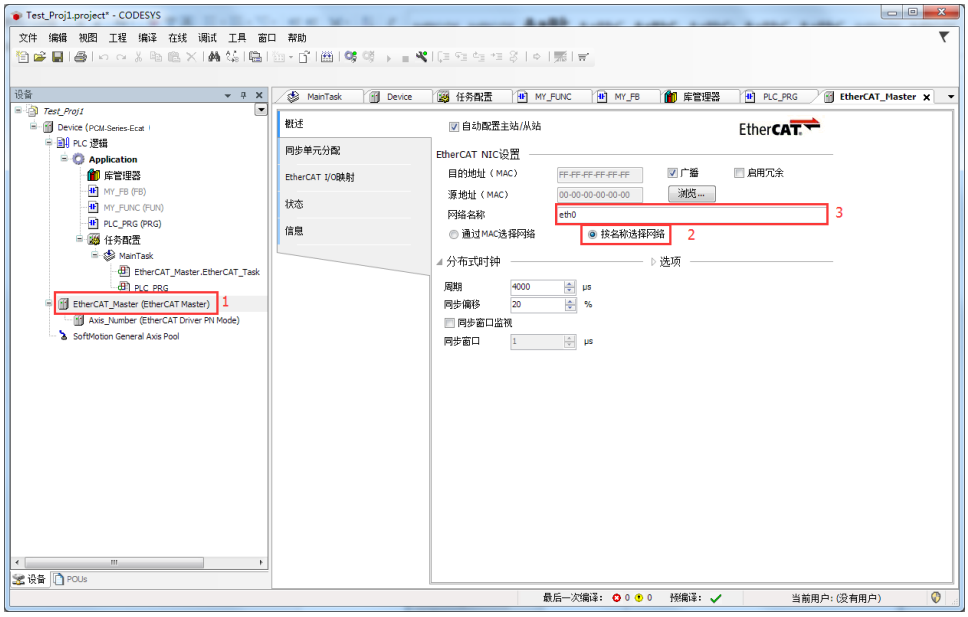
4.2.2 EtherCAT_Master_SoftMotion

该主站类型主要是作为 SoftMotion 多轴同步使用方式的选择。
选中“PLC 设备”- 右键单击 -“添加设备”-“EtherCAT_Master_SoftMotion”



4.2.3 设置 EtherCAT 主站的网络名称

依次单击“EtherCAT_Master”-“按名称选择网络”-“网络名称”
网络名称修改为:eth0



4.3 使用微秒 EtherCAT 总线型伺服实现分布式点到点控制

微秒的分布式控制方案可以有效减轻 PLC 的计算负载，同时配合 1ms 的 EtherCAT 运行周期，可以实现理想的多轴分布式控制方案。

IO 分配：微秒总线型伺服包含 12 个输入，8 个输出，对于一般设备，点数足够，可免除用户额外购买 IO 模块的成本。

注：关于总线伺服的详细资料请参考微秒控制的网站 www.vmmore.com 或联系代理商获取。

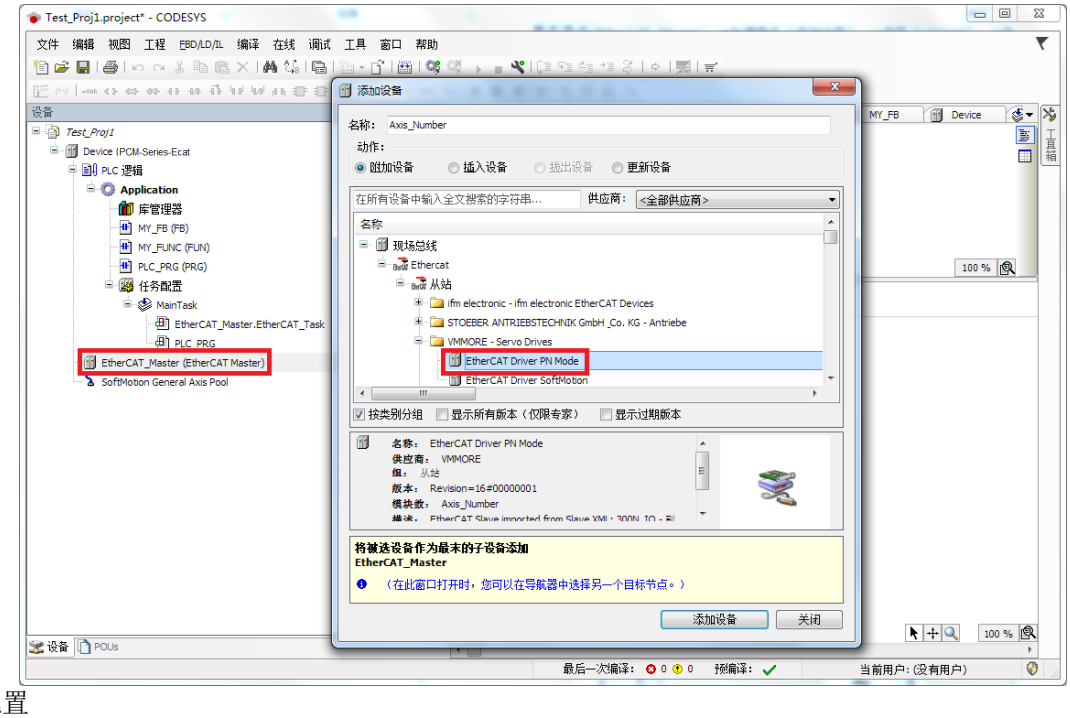
使用分布式控制需要使用 VM_Motion.compiled-library 库文件。

4.3.1 使用准备

添加设备

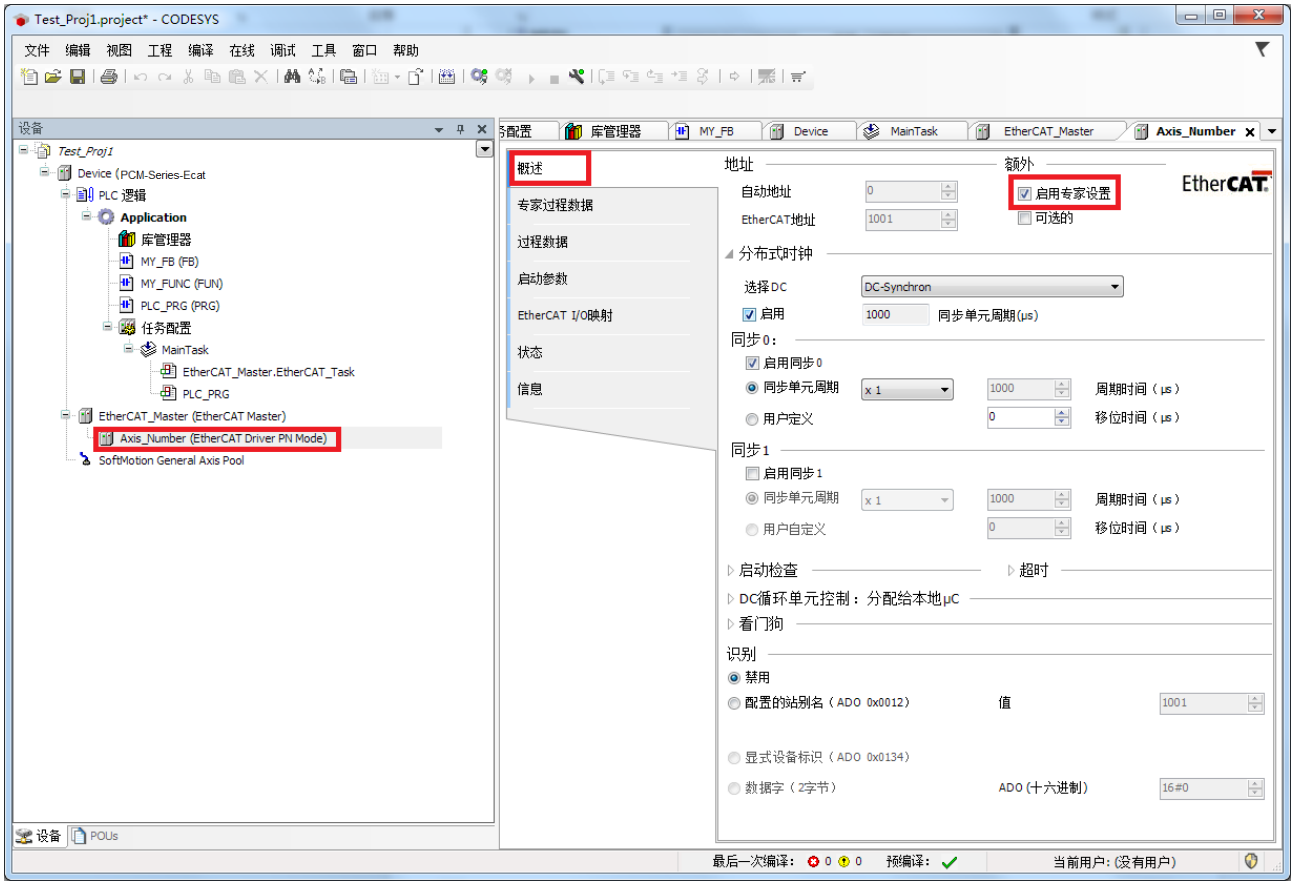
依据使用的轴数添加对应的设备，EtherCAT 网络中连接的设备数量必须和此处添加的从站伺服轴设备数量一致。

单击选中“EtherCAT_Master”→右键单击“添加设备” →选择“EtherCAT Driver PN Mode” →选择对话框下方左侧“添加设备按钮”

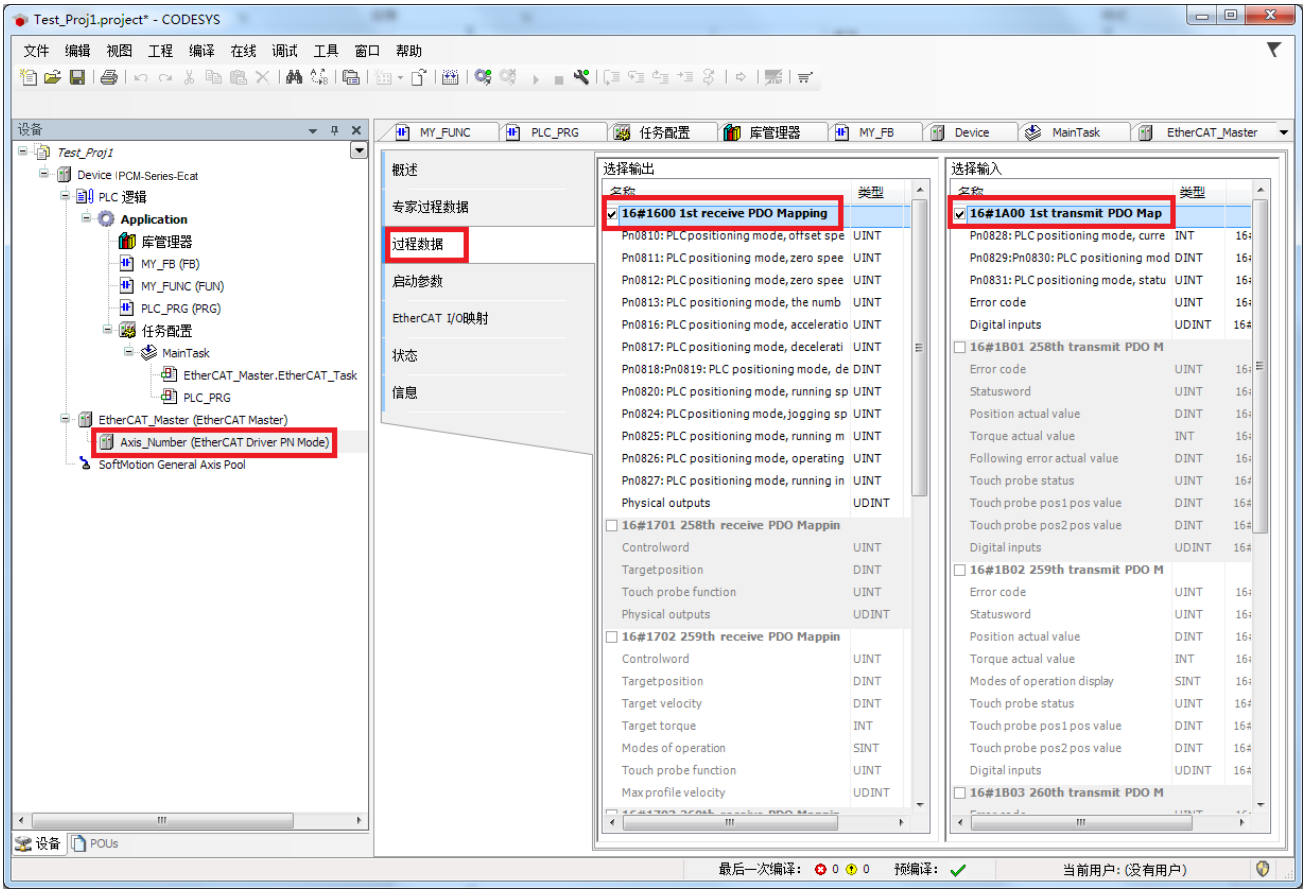


PDO 配置

过程数据配置参考如下过程，单击 EtherCAT Driver PN Mode 轴设备，选择“概述”选项卡，勾选“启用专家设置”选项卡。



过程数据的组需要使用如下配置：



过程数据链接必须保证已经安装完成库 VM_Motion.compiled-library。

4.3.2 微秒 EtherCAT 伺服轴 PN 参数的修改及编程

从站伺服内部的参数 PN 可以映射为 PDO 或使用 PN 读写函数以 SDO 方式进行修改。一般 PDO 的编辑不推荐用户自己进行，因此对于不是要求高度实时的参数修改推荐使用 SDO 进行修改参数。

关于 PN 读写的相关库函数使用方法请参考文档《VMMORE IEC library user guide》其中对应章节。

4.3.3 运动控制功能块

请参考《VMMORE IEC library user guide》文档。

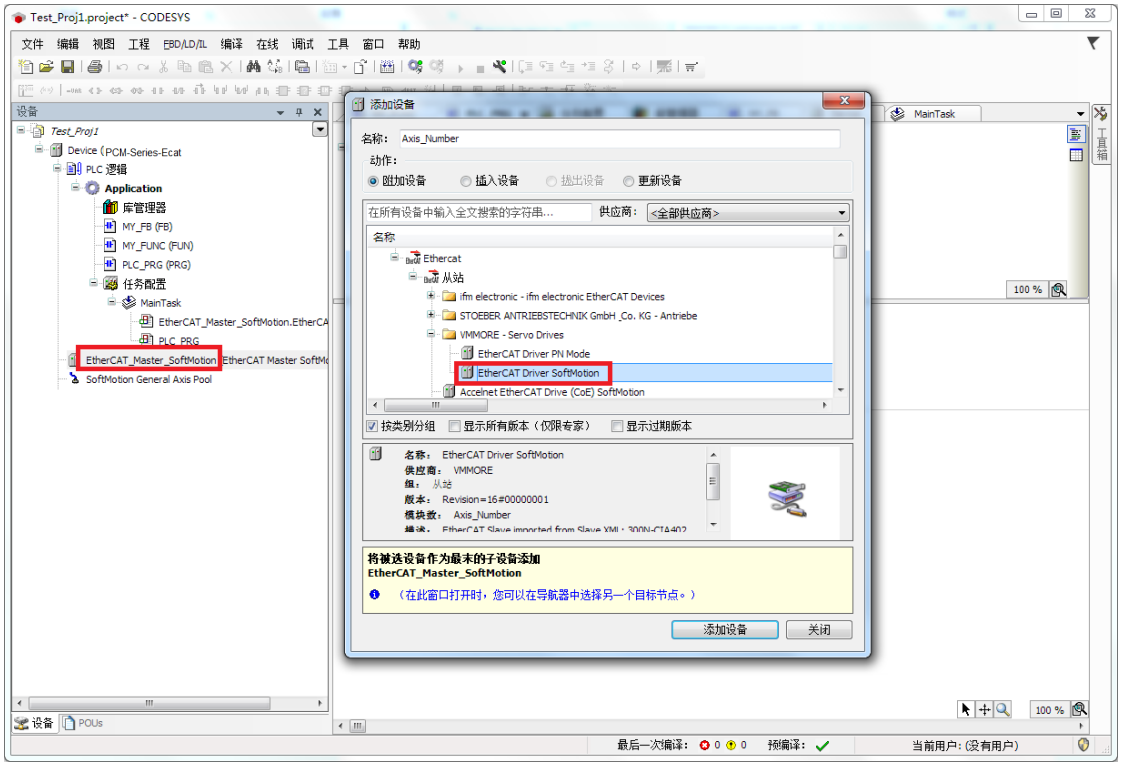
4.4 使用 SoftMotion PLCOpen 功能块

注：主站需要使用 EtherCAT_Master SoftMotion 类型。

如果用户需要使用第三方标准 Cia402 伺服，控制器可使用本产品，例如此处我们使用 300N 系列从站伺服作为控制对象。

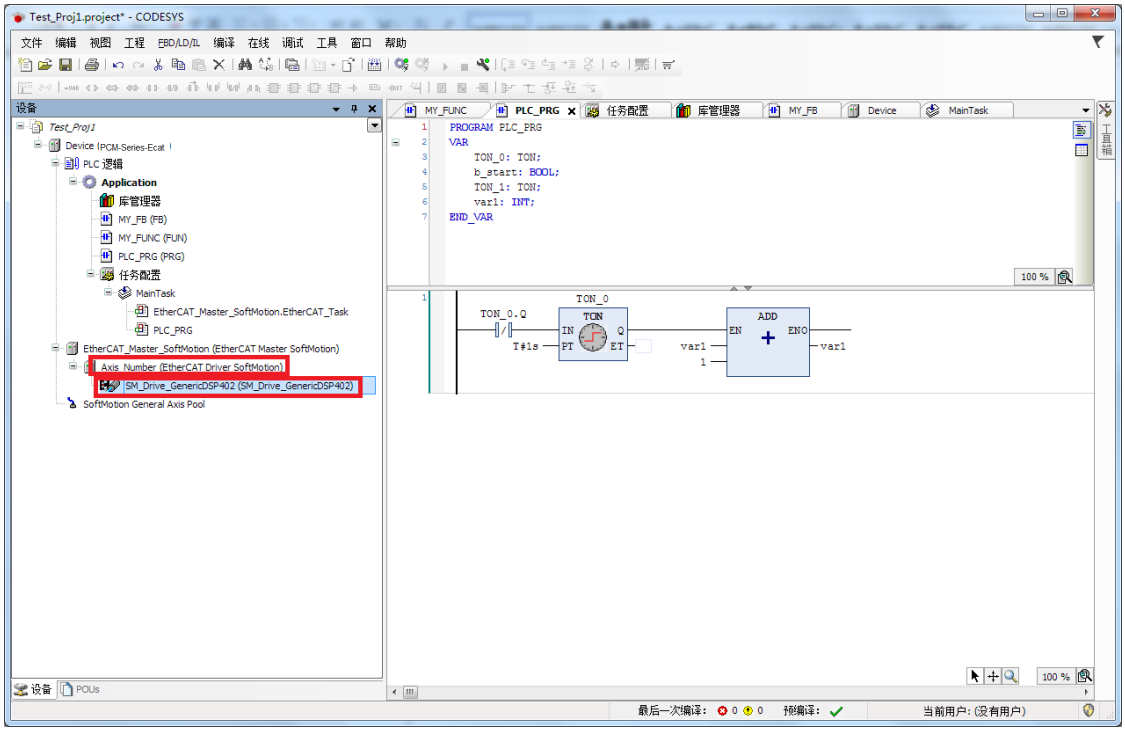
4.4.1 添加 CIA402 轴设备，使用 SoftMotion 实现伺服轴的运动控制。

单击选中“EtherCAT_Master SoftMotion”→右键单击“添加设备”→选择“EtherCAT Driver SoftMotion”→选择对话框下方左侧 “添加设备按钮”



选中 Axis_Number 右键单击→从菜单中选择“添加 SoftMotion 的 CiA402 轴”

添加完成后如下图所示。

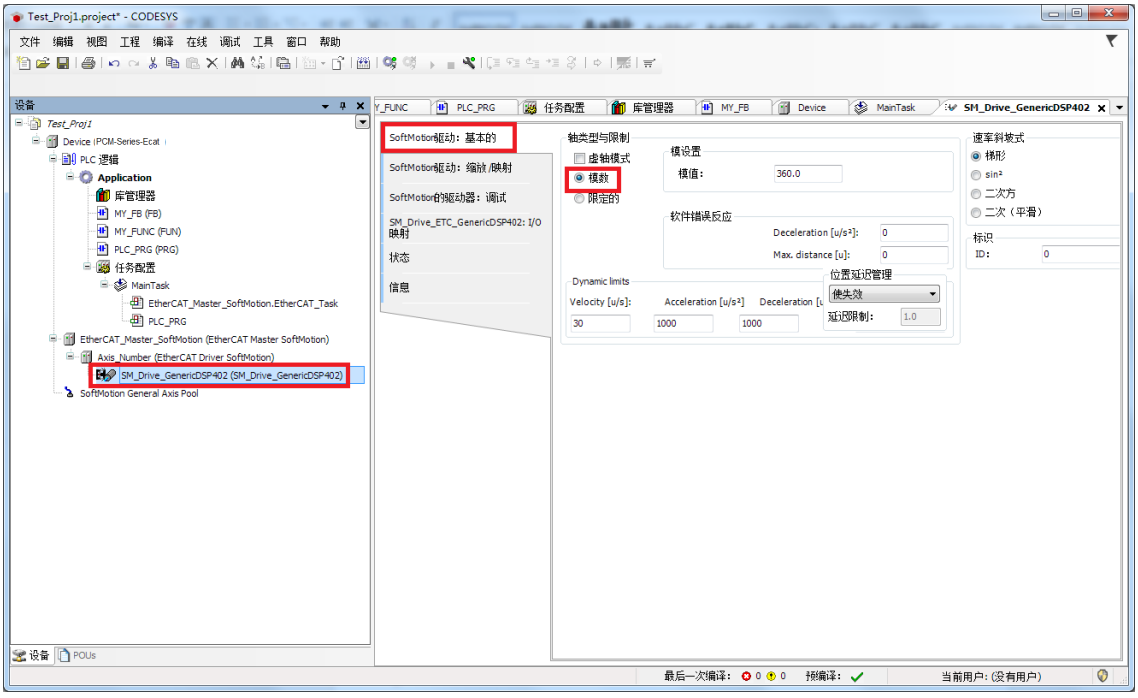


使用 SoftMotion 控制伺服轴参数配置，请参考微秒 300n 系列用户手册_v1.0 中 14 章的介绍。

4.4.2 SoftMotion 轴的参数配置

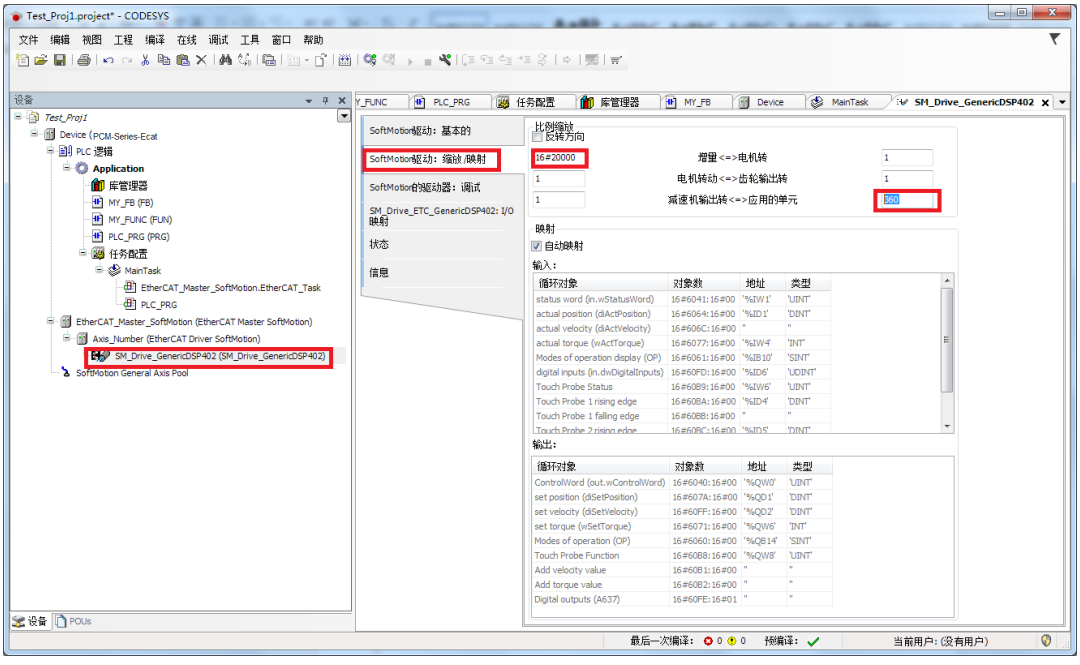
需要修改两个选项卡：

“基本的”



“缩放/映射”-打开如下图所示的选项卡。

“增量”：此处填写伺服对应的单圈脉冲数；应用单元：PLCOpen 功能块没有量纲，使用应用单元作为单位，例如转速=应用单位/秒，我们推荐使用 360 作为单位。

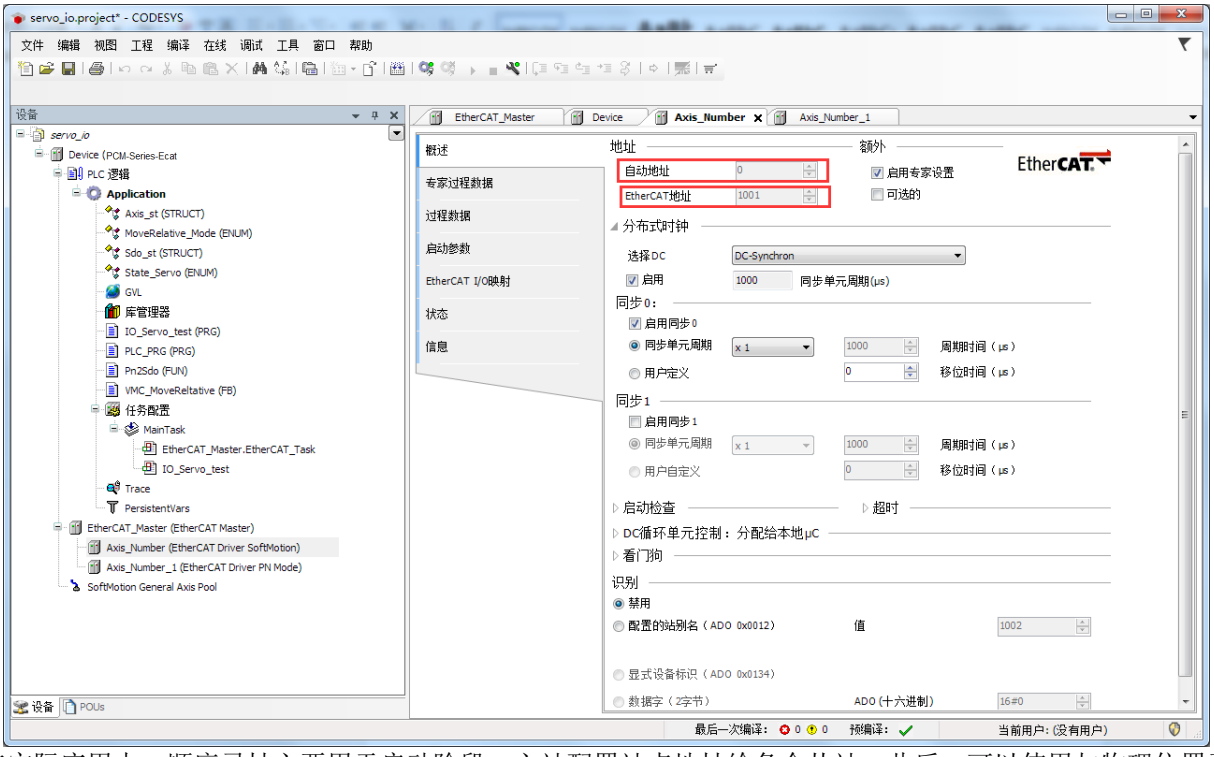


4.5 PLC 程序中 EtherCAT 主站、从站控制及查询

4.5.1 顺序寻址

在 Codesys3.5 中默认添加从站设备时，使用的是该寻址方式。

顺序寻址时，从站的地址由其在网段内的连接位置确定，第一个靠近主站的 EtherCAT 地址为 1001，后续的依次加一。自动地址：用一个负数来表示每个从站在网段内由接线顺序决定的位置。顺序寻址子报文在经过每个从站设备时，其顺序地址加一；从站在接收报文时，顺序地址为 0 的报文就是寻址到自己的报文。由于这种机制在报文经过时更新设备地址，所以又被称为“自动增量寻址”。



在实际应用中，顺序寻址主要用于启动阶段，主站配置站点地址给各个从站。此后，可以使用与物理位置无关的站点地址来寻址从站。使用顺序寻址机制能自动为从站设定地址。

4.5.2 设置固定的从站地址

设置寻址时，从站的地址与其在网段内的连接顺序无关。地址可以由主站在数据链路启动阶段配置给从站，也可以由从站在上电初始化的时候从自身的配置数据存储区装载，然后由主站在链路启动阶段使用顺序寻址方式读取各个从站的设置地址，并在后续运行中使用。

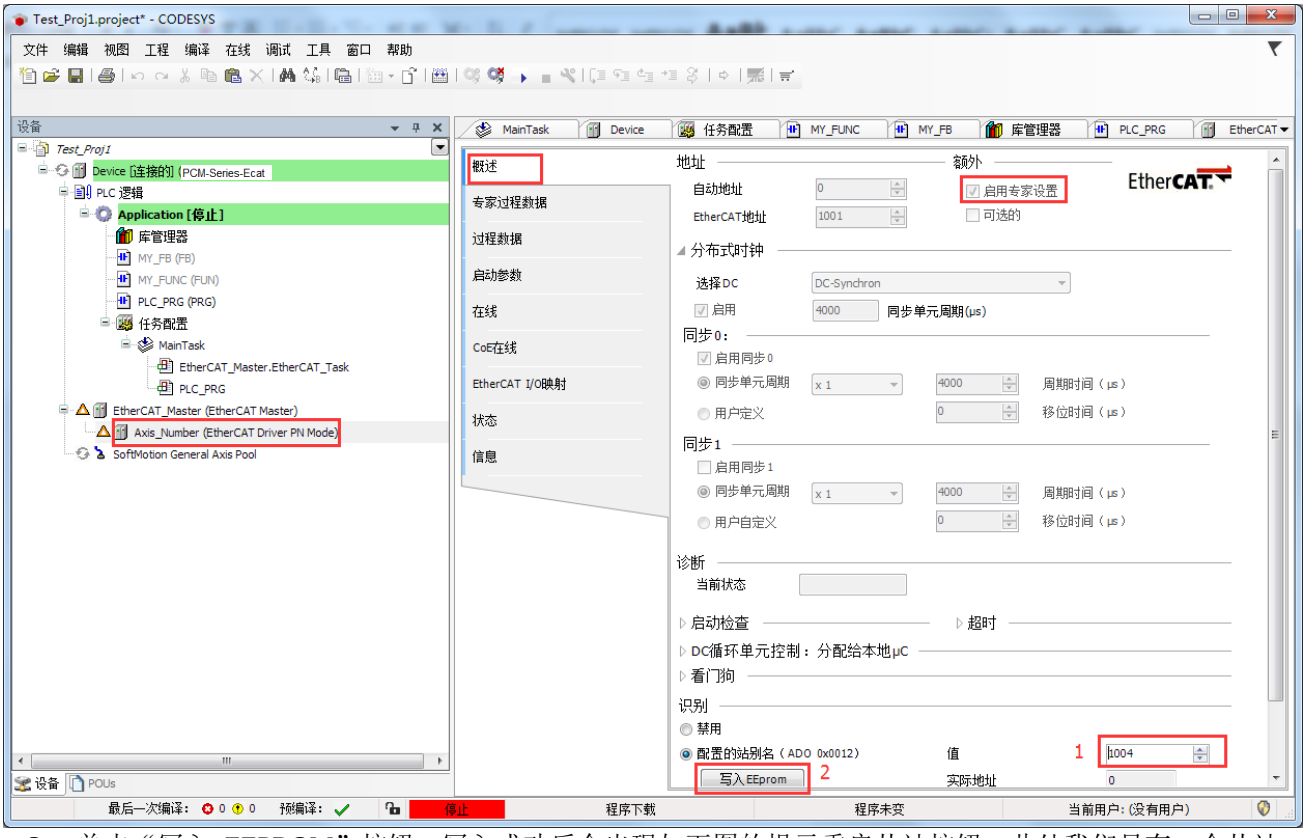
Codesys3.5 中固定从站地址设置方法：

1> 使能从站“启用专家设置”

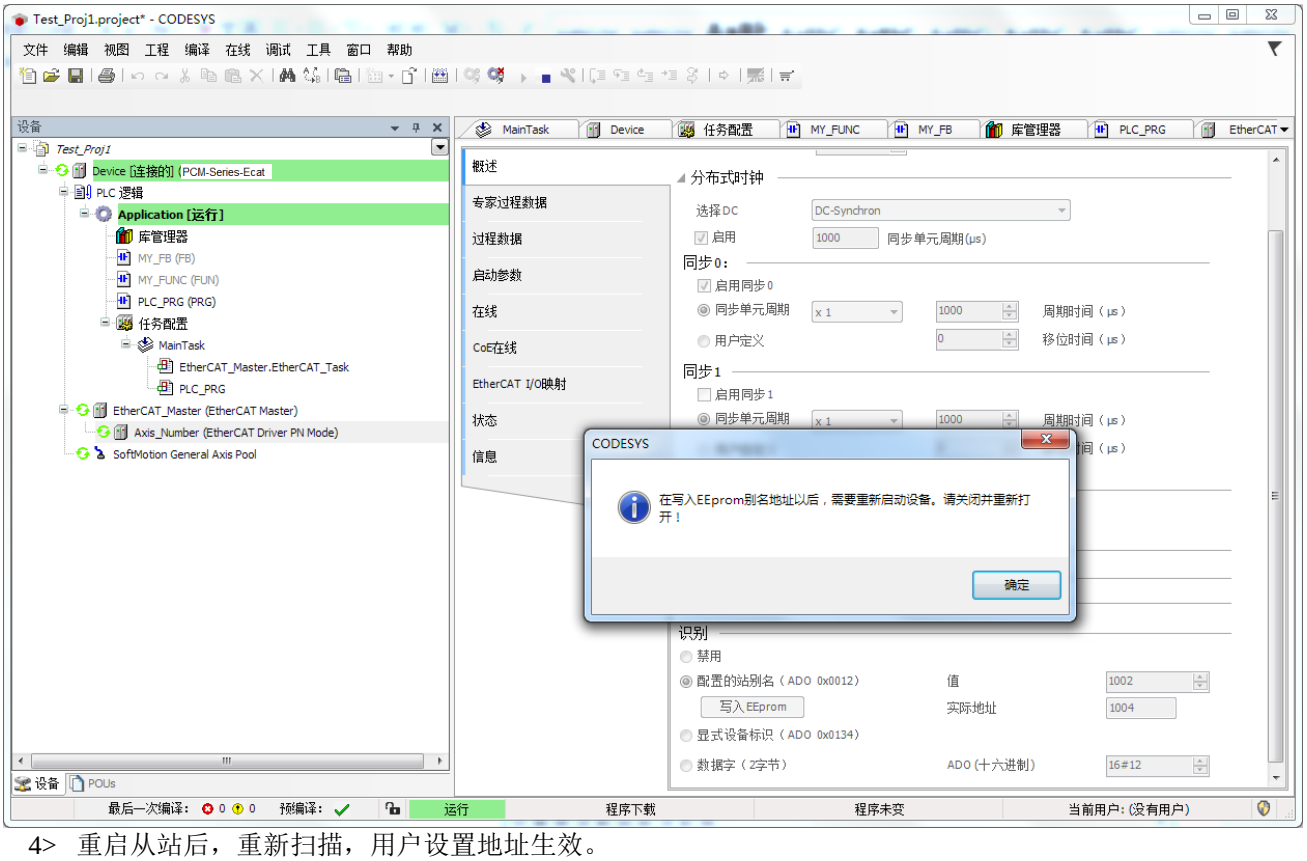
2> 登陆到 PLC，进入在线模式。

在线设置从站伺服的地址

在线模式下，进入从站配置选项卡“概述”，修改下方的“配置的站别名”，将“值”修改为用户定义的地址。



3> 单击“写入 EEPROM”按钮，写入成功后会出现如下图的提示重启从站按钮，此处我们只有一个从站，所以只设置了一个固定地址，当用户项目中用到了多个从站时，需要逐个设置地址，并且每个从站地址不可重复。



4> 重启从站后，重新扫描，用户设置地址生效。

5 控制器其它接口使用

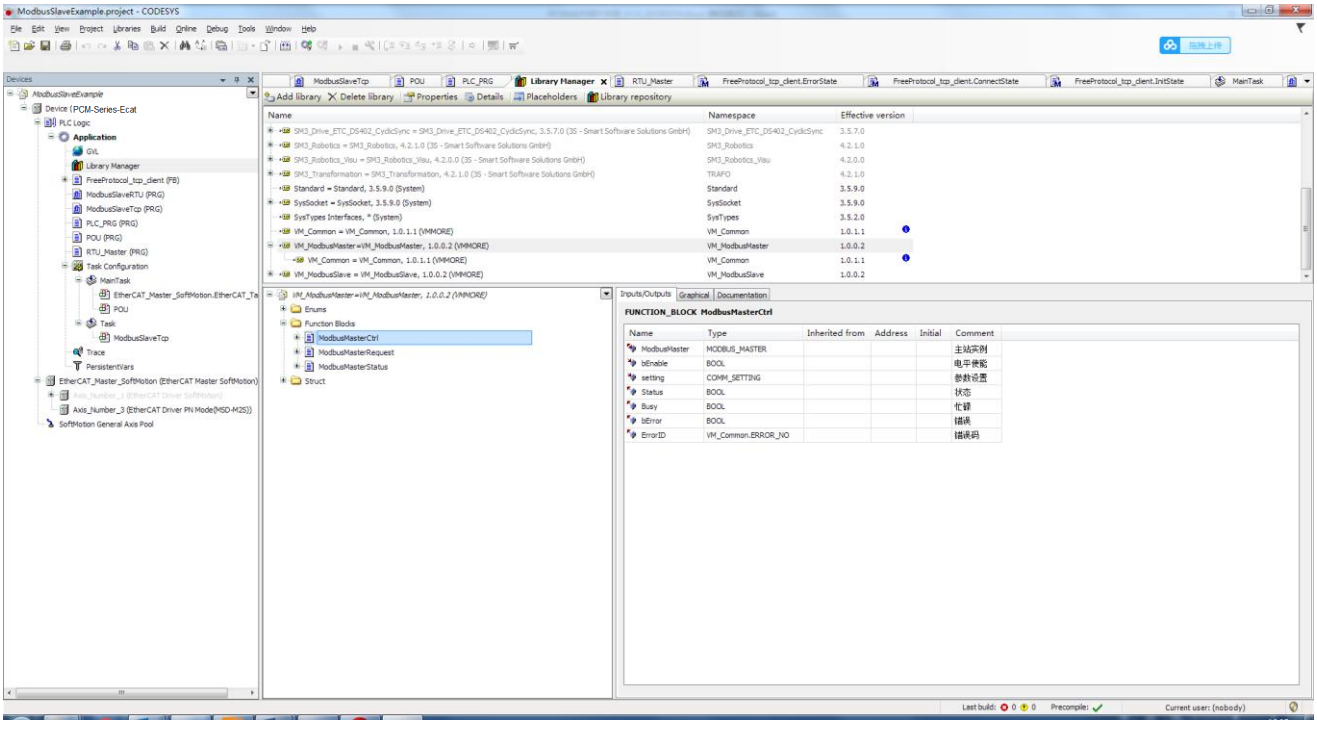
使用 PLC 的如下接口需要保证前面 2.2.2 章节所介绍的库文件的安装正确，及 PLC 编程环境的正确配置。

5.1 MODBUS 主从站使用

5.1.1 MODBUS 主站介绍

MODBUS 主站库提供 TCP/RTU 两种通信模式，在使用 MODBUS 主站之前，需要先安装并加载 VM_Common 库和 VM_ModbusMaster 库，库的安装和加载请参考上述章节。下面详细介绍 MODBUS 主站库的接口：

点开 MODBUS 主站库，可以看到主站提供的配置和功能块，如下图



上图中，Function Blocks 文件夹中的是库提供的功能块；Functions 文件夹中包含的是可供调用的函数，包括读写位、字的函数；Struct 文件夹中包含的是通信配置参数。这些功能块和参数都配有中文注释。下面分开介绍。

1) MODBUSMasterCtrl，主要功能是初始化主站实例，生成主站设备节点，进一步给 ModbusMasterRequest 提供使用。

ModbusMaster：Modbus 主站设备实例，节点请求需要使用该实例。

Enable：使能功能块，上升沿触发，下降沿复位功能块（关闭连接）

setting：参数设置，串口和网口参数配置

Status：指示当前硬件线路上的主站实例的初始化状态

Busy：忙碌标志位，正在执行时会置位

bError: 错误标志位，连接断开或通信出错都会置位标志位

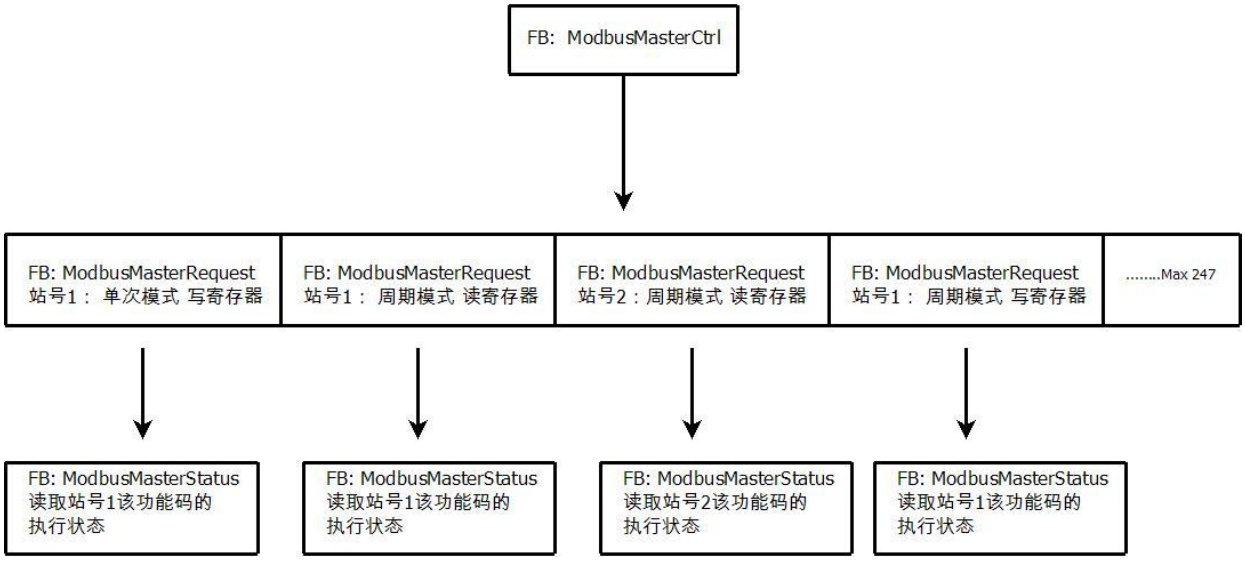
ErrorID：错误码，错误时会输出错误码

2) ModbusMasterRequest，生成有效的请求节点，分为单次和周期读写功能码，函数参数参考注释

3) ModbusMasterStatus，获取请求节点的功能码执行状态，函数参数参考注释

注：此处 pNode 参数需要为 ModbusMasterRequest 执行成功请求的节点。

比较完整的调用关系如下图



4) COMM_SETTING，参数设定结构体，详细请参考注释

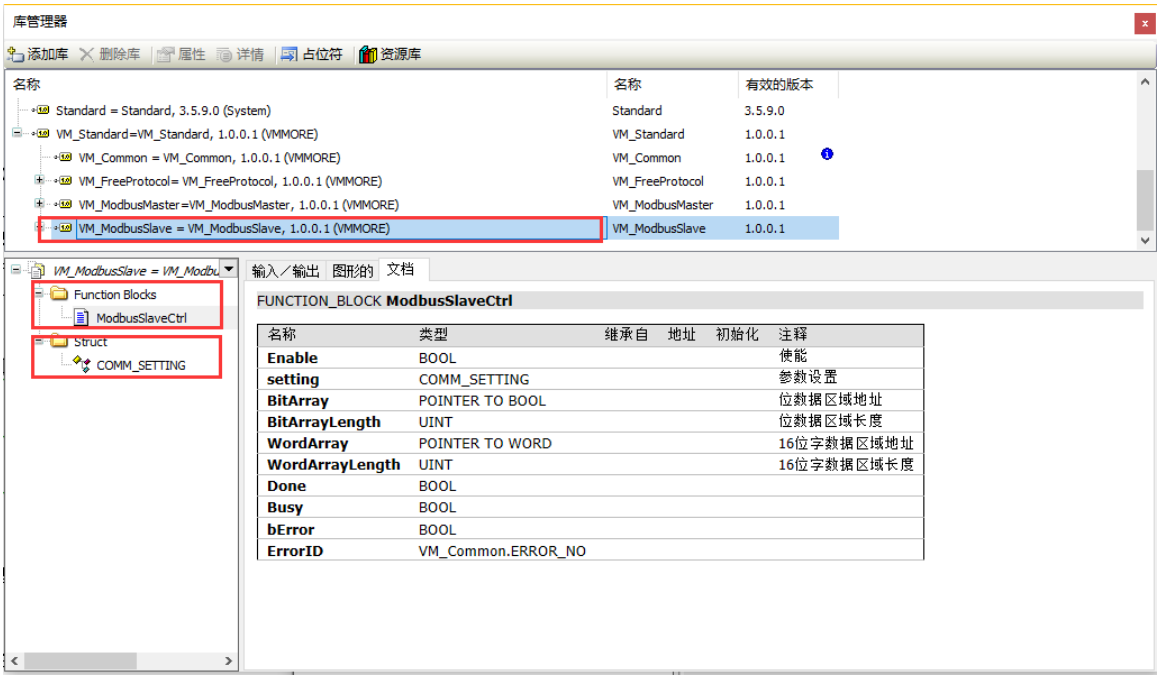
- commType：通讯类型，TCP/RTU，默认为 TCP
- serial：串口号，默认 COM1，PC5M 提供 COM1、COM2、COM3 三个 485 物理接口，详情请参考前面章节
- baudRate：波特率，通信类型 RTU 有效
- parity: 校验，默认偶校验，通信类型 RTU 有效
- stopBit: 停止位，默认 1 位，通信类型 RTU 有效
- ScanRate: 如果 ModbusMasterRequest 使用周期模式，此处定义扫描周期默认 1s
- ResponseTime：ModbusMaster 主站超时时间，单位 ms，默认 1s
- frame_byte_timeout:单位 ms ,modbus 帧和帧之间的分辨时间,默认前后 100ms 内的字节流算一帧

使用示例，请参考随机的 MODBUS 主站示例教程。

5.1.2 MODBUS 从站

MODBUS 从站库同样提供 TCP/RTU 两种通信模式，在使用 MODBUS 从站之前，同样需要先安装并加载 VMMORE 标准库，即 VM_Standard 库，标准库的安装和加载请参考上述章节。下面详细介绍 MODBUS 从站库的接口：

点开 MODBUS 从站库，可以看到从站提供的配置和功能块，如下图。



上图中，Function Blocks 文件夹中的是库提供的功能块，Struct 中包含的是通信配置参数。
这些功能块和参数都配有中文注释，下面分开介绍：

1) ModbusSlaveCtrl，主要负责和主站建立连接，并监听主站命名，处理后返回结果
- Enable：使能功能块，上升沿触发，下降沿复位功能块（关闭连接）
- Setting：参数设置，串口和网口参数配置
- BitArray：位数据区域地址
- BitArrayLength：位数据区域长度
- WordArray：16 位字数据区域地址
- WordArrayLength：16 位字数据区域长度
- Done：完成标志位，通信正常打开，此标志位会被置位
- Busy：忙碌标志位，正在连接时会置位
- bError: 错误标志位，连接断开或通信出错都会置位标志位
- ErrorID：错误码，错误时会输出错误码

2) COMM_SETTING，参数设定结构体，详细请参考注释
- commType：通讯类型，TCP/RTU，默认为 TCP
- slaveID：本机从站站号，默认 1
- serial：串口号，默认 COM1，PC5M 提供 COM1、COM2、COM3 三个 485 物理接口，详情请参考前面章节
- baudRate：波特率
- parity: 校验，默认偶校验
- stopBit: 停止位，默认 1 位
- ipAddr: 允许连接的 MODBUS 主站地址，默认不赋值(所有地址的 MODBUS 主站都可以连接)
- port：TCP 端口,默认 502
- timeout：相邻的两个字节接收间隔时间，超过该设定时间，返回数据认定为一帧

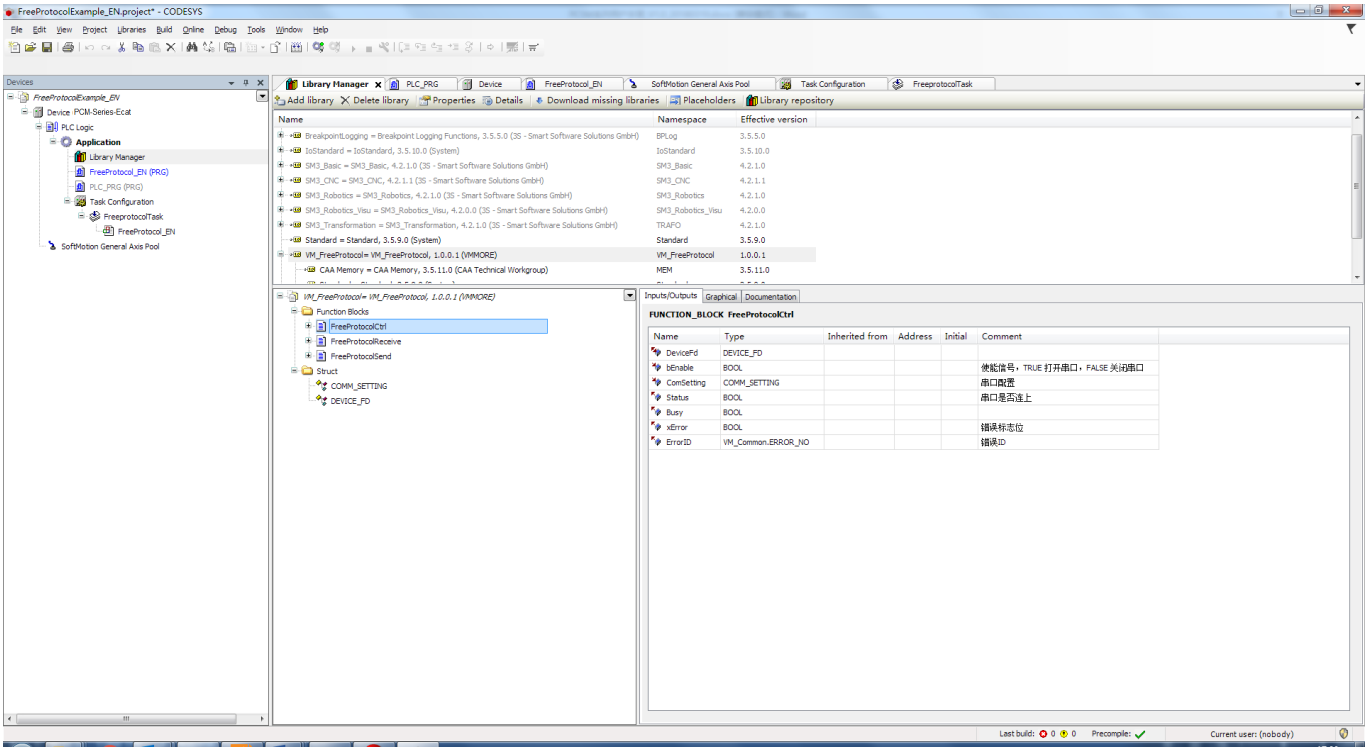
使用示例，请参考随机的 MODBUS 从站示例教程。

5.2 自由口的使用

自由口库只提供串口通信模式，在使用自由口库之前，同样需要先安装并加载 VMMORE 标准库和 VM_FreeProtocol.compiled-library 库，库的安装和加载请参考上述章节。

5.2.1 自由口库介绍

下面详细介绍自由口库英文版的接口，点开自由口库，可以看到参数配置和功能块，如下图。



上图中，Function Blocks 文件夹中的是库提供的功能块。
这些功能块和参数都配有中文注释，下面分开介绍：

1) FreeProtocolCtrl，主要负责和自由口的另一端连接，发送命令，并接收返回数据
- Enable：使能功能块，上升沿触发，置位打开串口连接，复位关闭串口连接
- DeviceFd：自由口设备描述符，供自由口发送接收使用
- ComSetting：自由口 UART 参数配置
- bstatus：相应的硬件设备打开状态
- xError：指示打开设备是否出错

2) FreeProtocolSend，主要负责通过自由口发送数据
- DeviceFd：给将要发送的数据指定自由口设备描述符
- bExcute：上升沿使能一次发送
- SendArray：发送数组，填充将要发送的数据，限制每次发送的有效数据小于等于 512Byte
- SendLength：指定要发送数据的长度，最大 512
- Done：发送完成标志

3) FreeProtocolReceive，主要负责自由口数据的接收
- DeviceFd：指示将要接收数据的串口设备指定有效的设备描述符
- bExcute：上升沿使能一次接收
- ReceiveArray：接收数组，每次接收都会覆盖之前的内容，接收缓冲区最大 512Byte
- ReceiveLength：接收到的字节数，最大为 512
- Done：成功接收到一次数据，该变量为 TRUE
- xError：指示接收错误

使用示例，请参考随机的 FreeProtocol 示例教程。

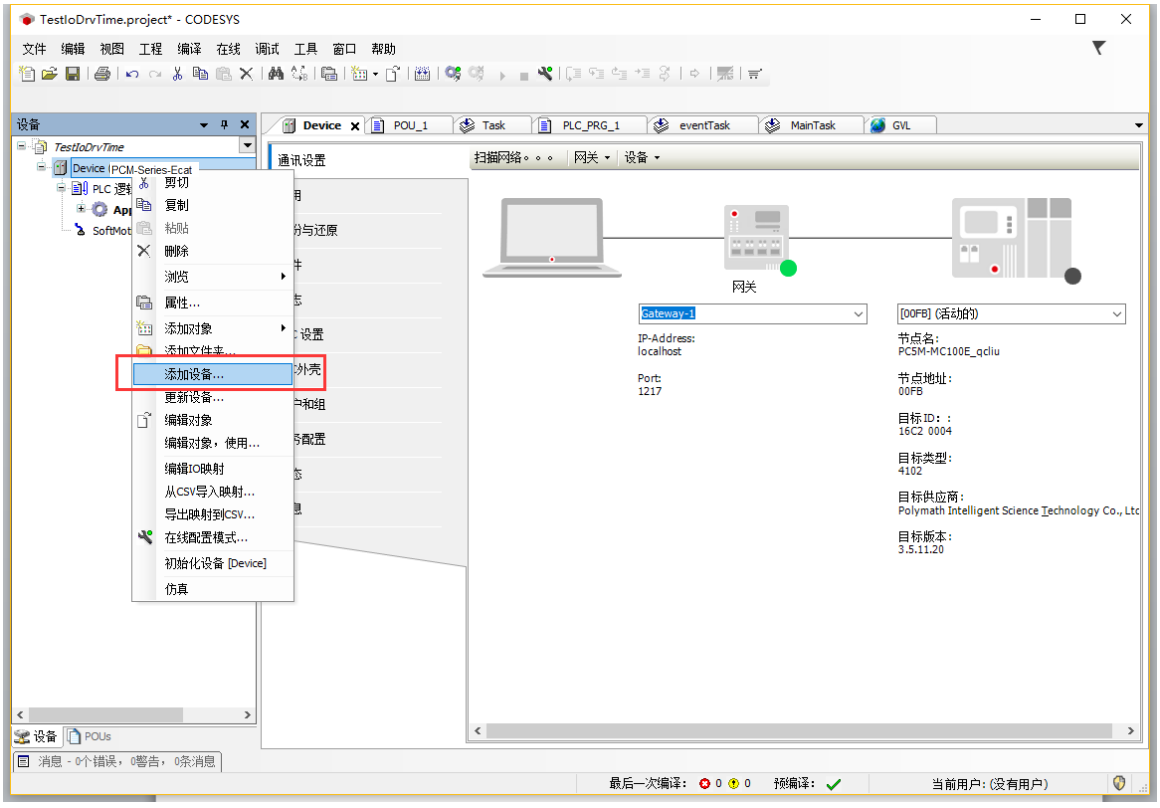
5.3 本机 8 路 DI 的使用

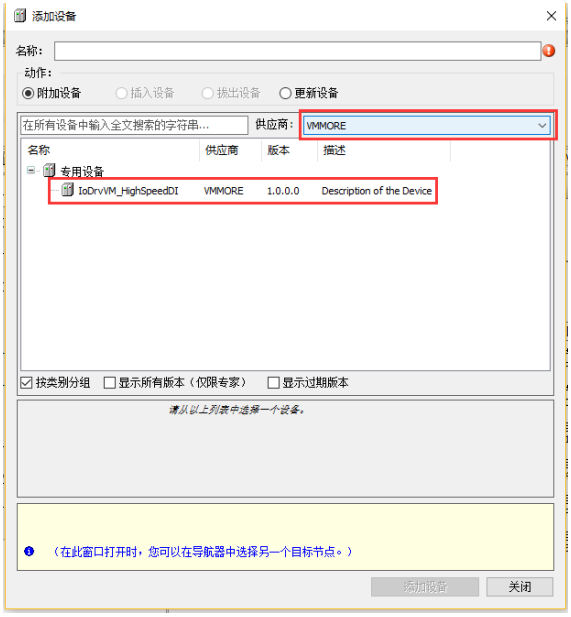
5.3.1 DI 设备的添加和配置

要使用 DI，必须要先安装相应的设备描述文件到设备库，然后在工程中添加设备，才能使用。

安装设备描述文件的方法请参考前面章节。

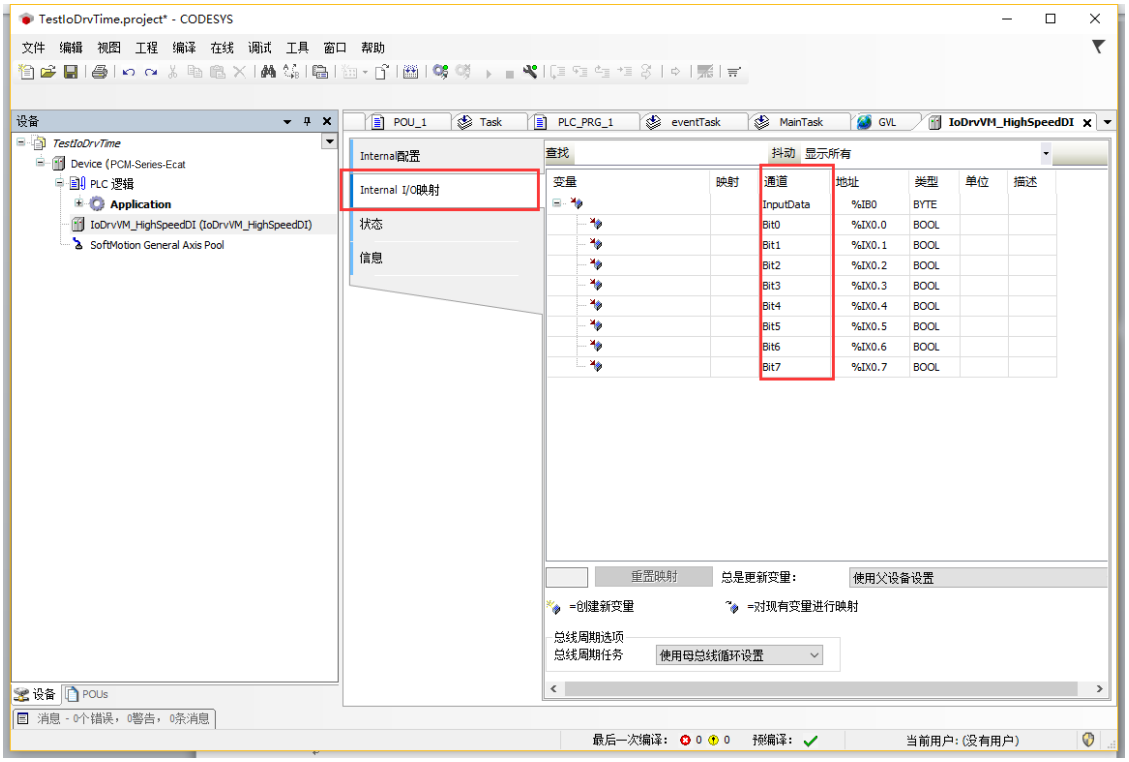
将设备添加到工程，右键单击 PCM-Series-Ecat 设备文件，选择添加设备，供应商选择 VMMORE，选择设备 IoDrvVM_HighSpeedDI。



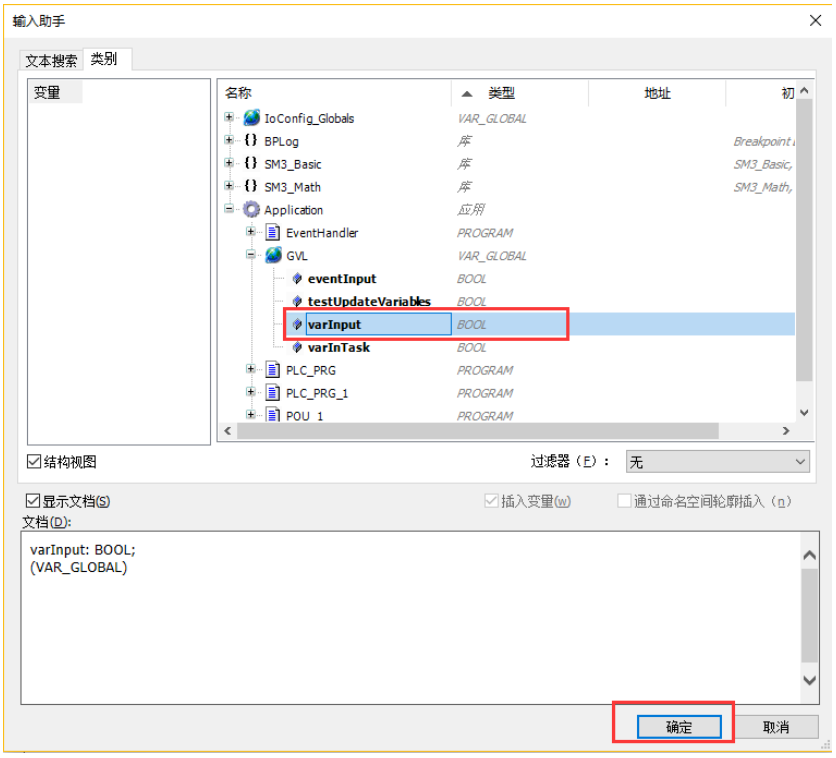


添加设备完成，如下图。

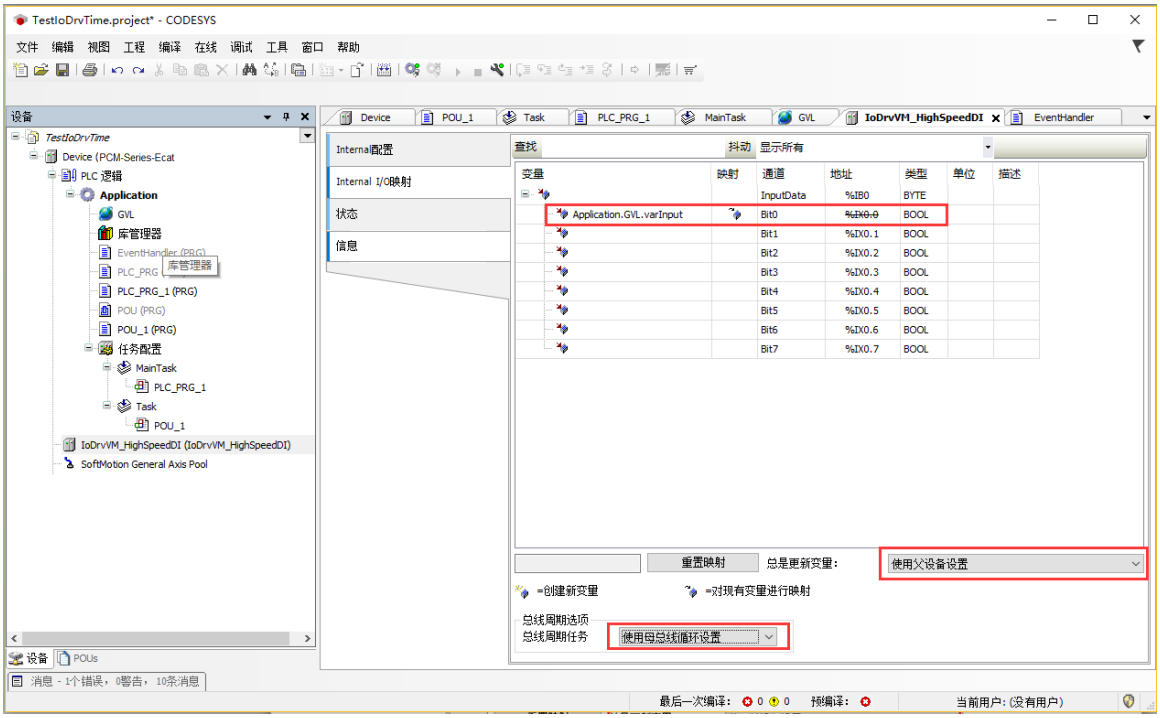
添加设备完成后，即可使用DI映射到程序中的变量。鼠标左键双击设备，选择“Internal I/O映射”即可看到8路DI。



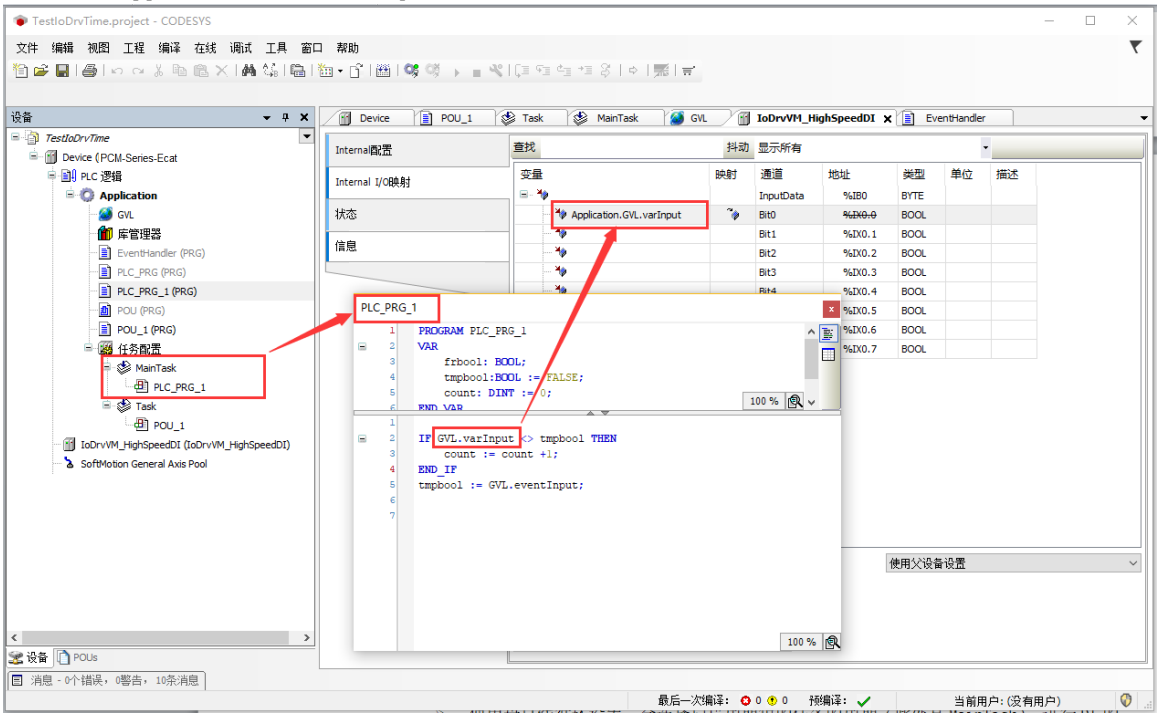
鼠标左键双击其中一路 DI 的变量栏，可以选择要映射的变量，请注意映射的变量只能是 BOOL 类型。也可以将整个 BYTE 类型的输入映射到一个 BYTE 类型的变量。



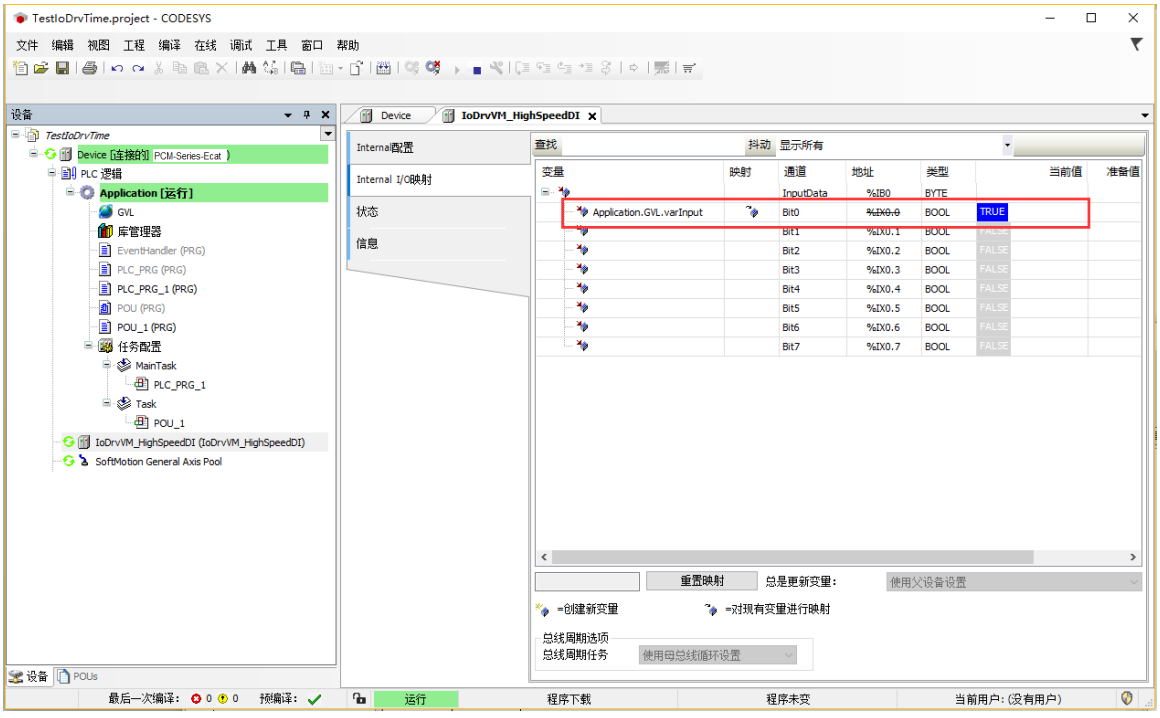
变量映射完成，下面框出的两个选项保持默认值即可。



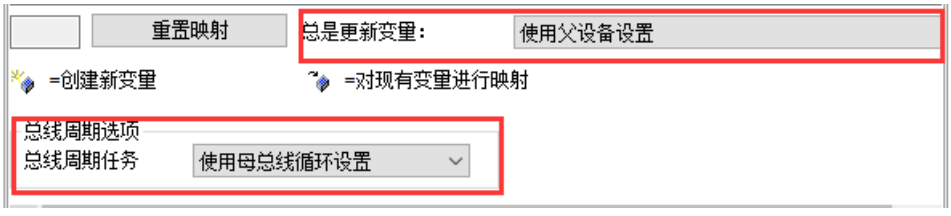
上图中的变量 Application.GVL.varInput 需要在一个任务中使用，不然变量值无法更新，如下图。



配置完成后，登陆下载工程，即可看到变量随着 DI 输入值变化。



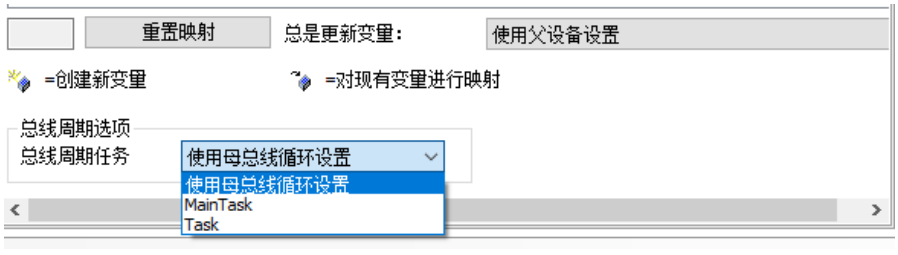
至此，DI 已经可以正常使用，这里需要介绍下两个配置项，“总线周期任务”和“总是更新变量”。



总线周期任务，决定了 DI 组件扫描周期。如下图，工程中有 MainTask 和 Task 两个任务，MainTask 是主任务，扫描时间短，Task 扫描时间长。

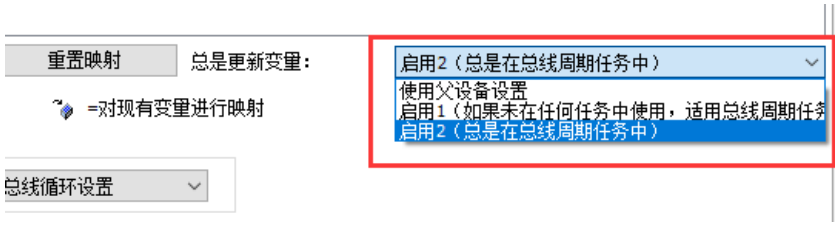
使用母总线循环设置，会选择扫描周期短的任务的周期（此处是 MainTask），进行 DI 的扫描。

选择 MainTask 或者 Task，会按照选择的任务周期进行 DI 扫描。



建议使用默认设置，即“使用母总线循环设置”，同时，建议如果并不需要使用 DI，请不要添加此 DI 设备，因为会产生额外的性能消耗。

“总是更新变量”决定了 DI 通道映射的变量的更新周期，如下图。



使用父设备设置，未映射变量的通道，或者被映射了变量但是变量未被使用的，不会被更新。

启用 1，所有 DI 通道都会被更新，但是映射了变量且变量被使用的通道，将以变量所在的任务的任务周期进行更

新，其他通道会以总线周期进行更新（“总线周期任务” 中设置的周期）。

启用 2，所有 DI 通道都会被更新，更新周期为设定的总线周期。

此处建议使用默认设置，即“使用父设备设置”，详细配置方法请参考 Codesys 帮助文档。

5.3.2 DI 作为外部事件用法

要使用 DI 外部事件，同样需要安装 DI 设备描述文件，安装方法请参考章节 5.3.1。

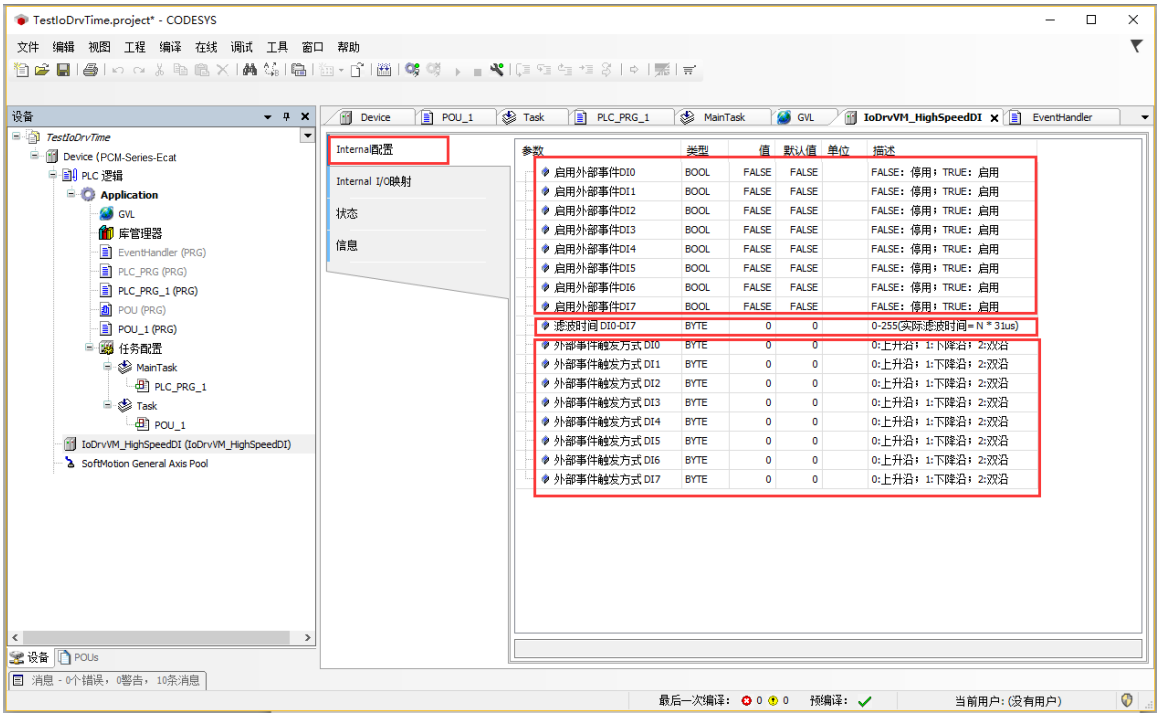
1. 事件配置

下面是事件的配置方法，请在工程中添加 DI 设备后，双击设备→选择 Internal 配置，配置分为三部分，

1>启用事件

2>滤波时间

3>外部事件触发方式

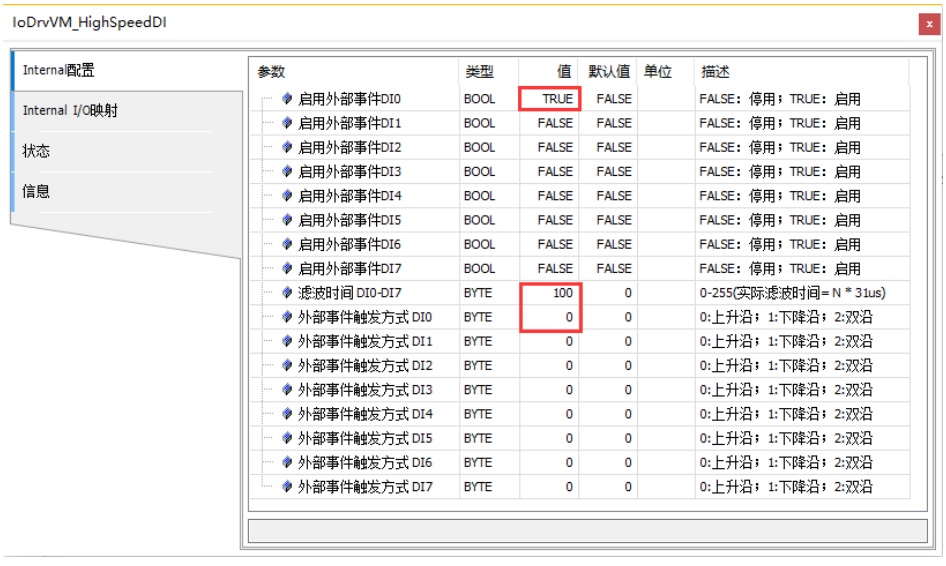


外部事件默认为关闭，需要启用之后，才会生效。

滤波时间对所有通道生效，不提供对单个通道进行滤波时间配置，注意最终的滤波时间=N*31us，建议滤波时间配置值大于 10，因为滤波时间较短时，会因为系统调度等原因，导致事件丢失。

外部事件触发的方式分为上升沿、下降沿或者双沿，请不要配置超过 2 的值。

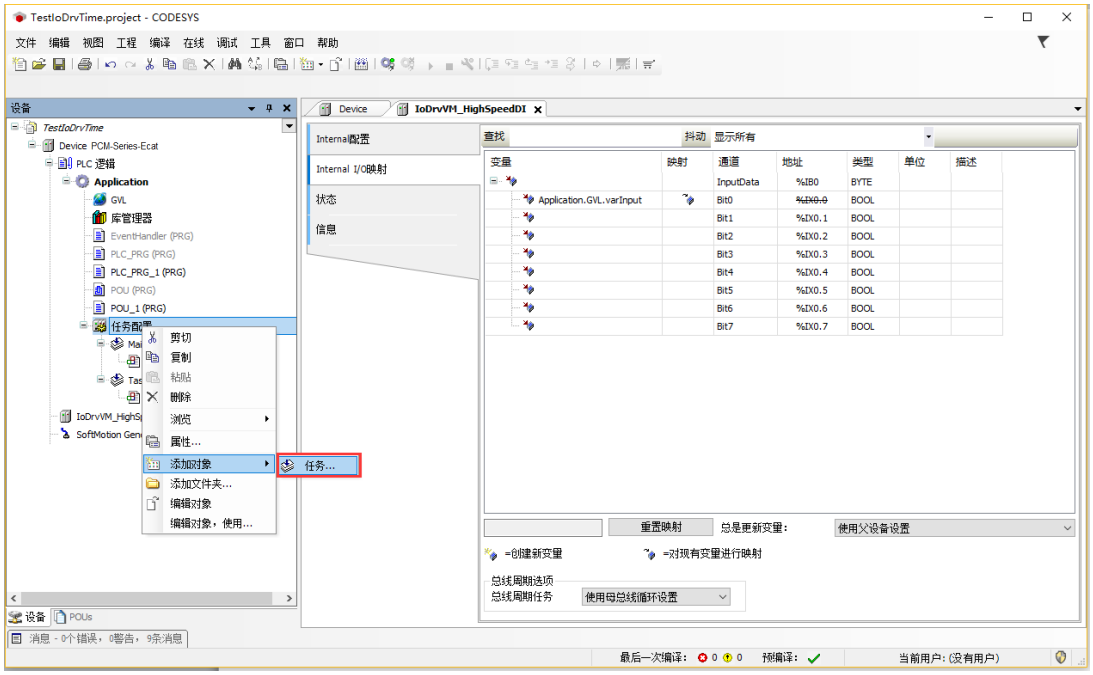
下图中示例，启用了事件 DI0，设置滤波参数为 100，事件触发为上升沿。



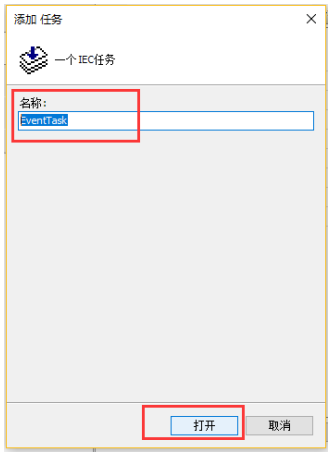
2. 添加事件处理程序

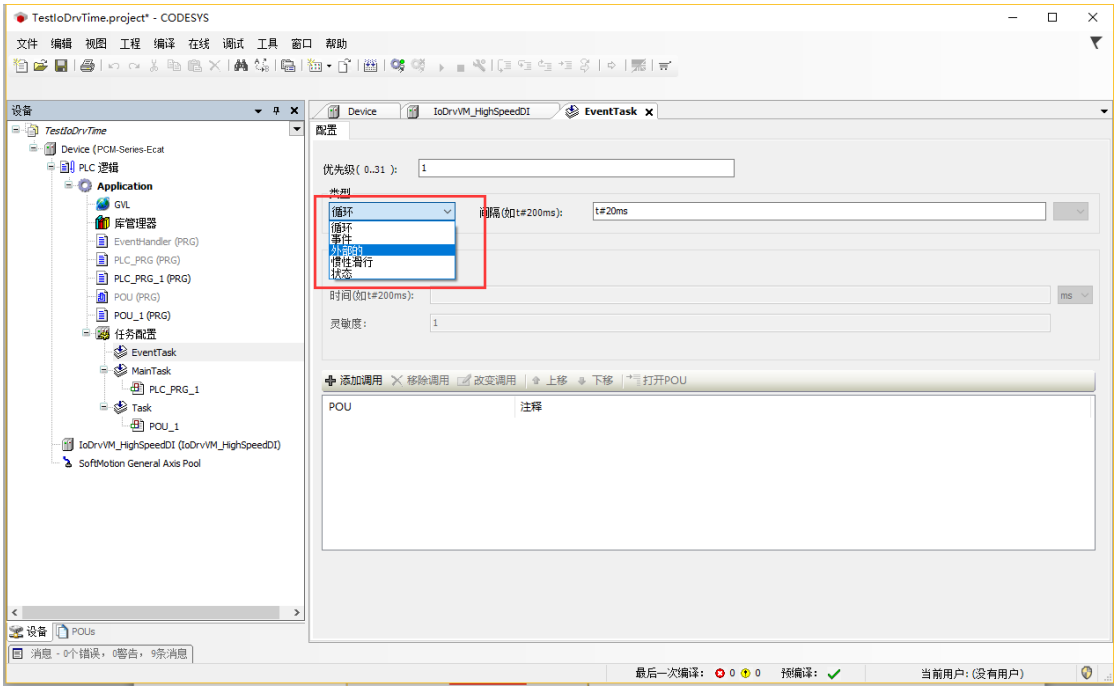
事件启用之后，需要添加事件处理程序，事件触发将会调用事件处理程序。

右键单击“任务配置”-“添加对象”-“任务”。

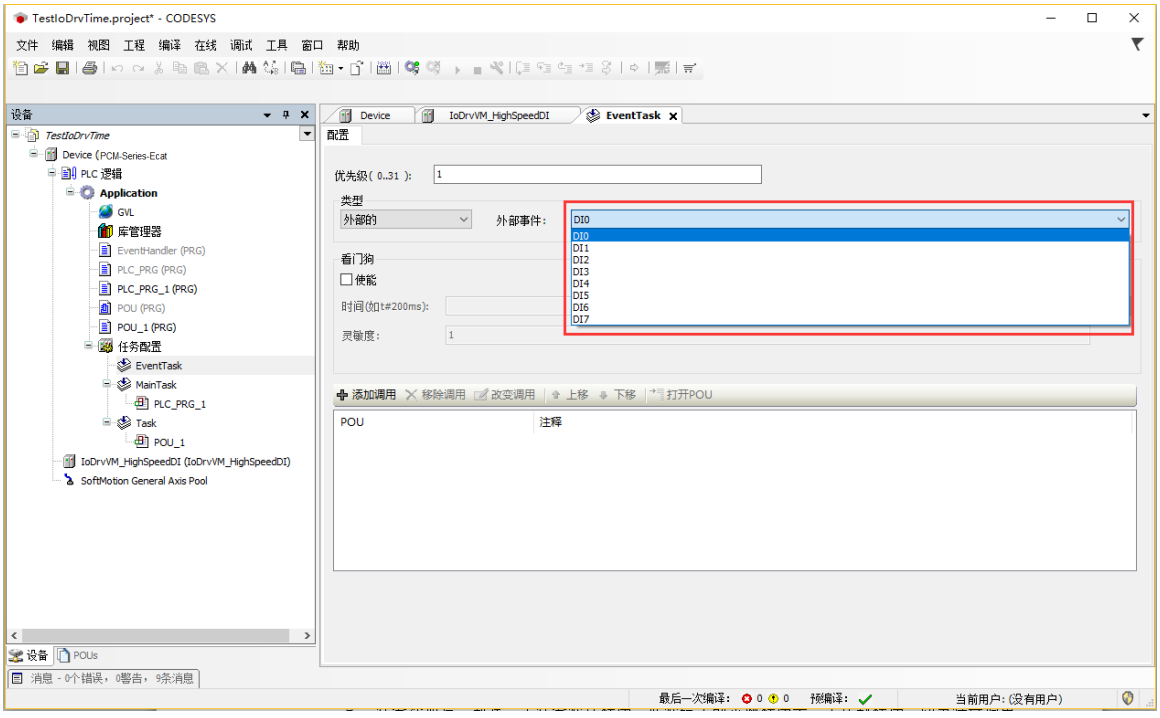


给任务命名，此处命名为“EventTask”，并设置任务类型为：“外部的”。



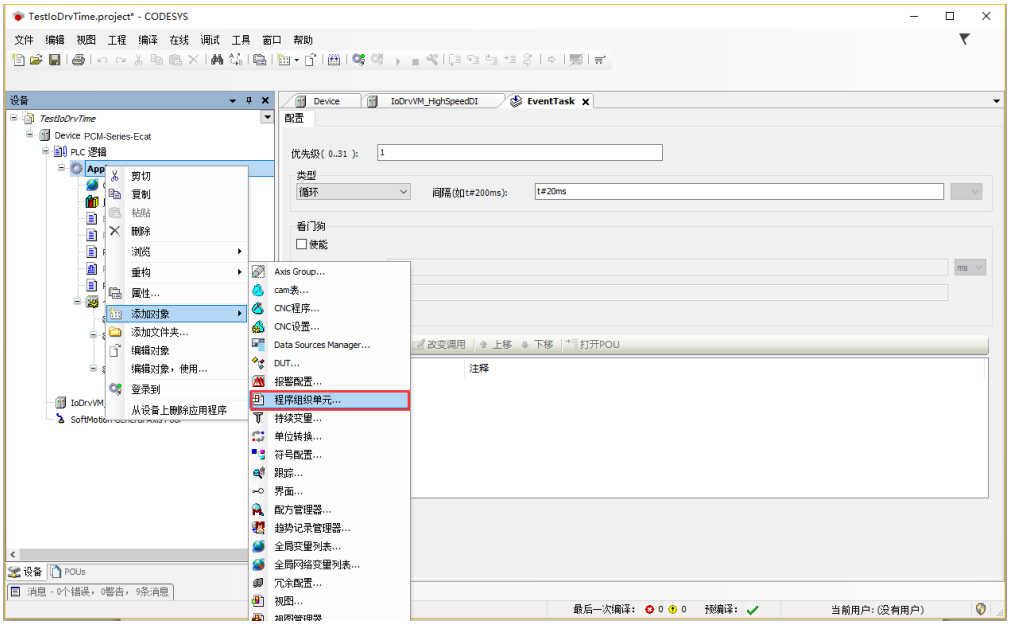


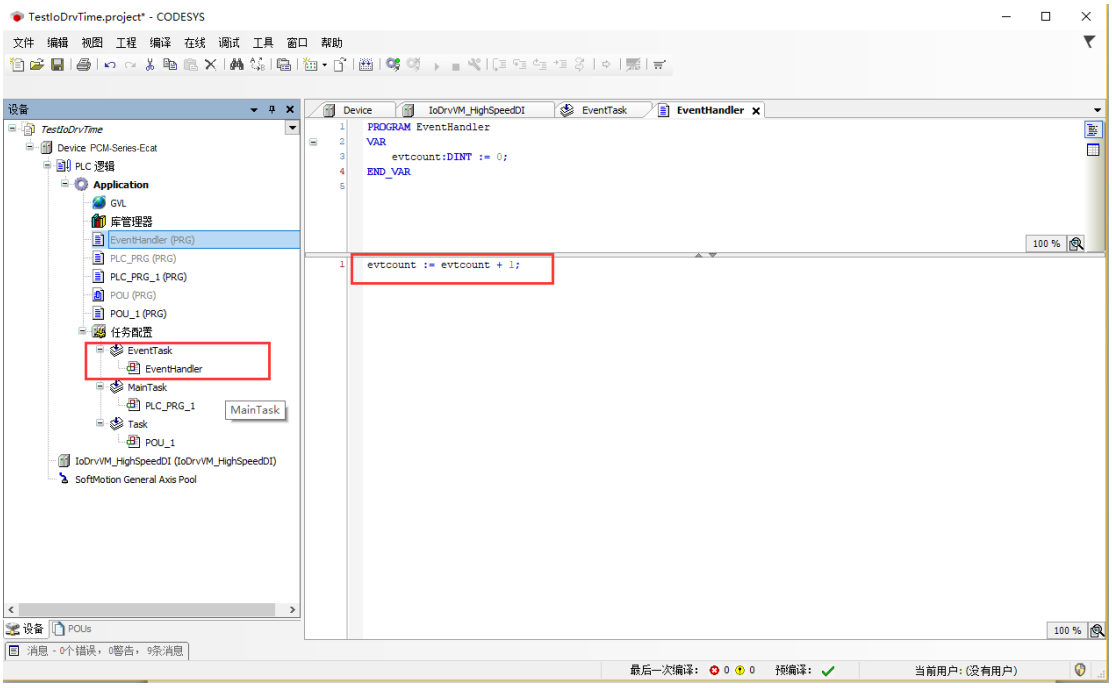
选择触发信号，此处选择 DI0。



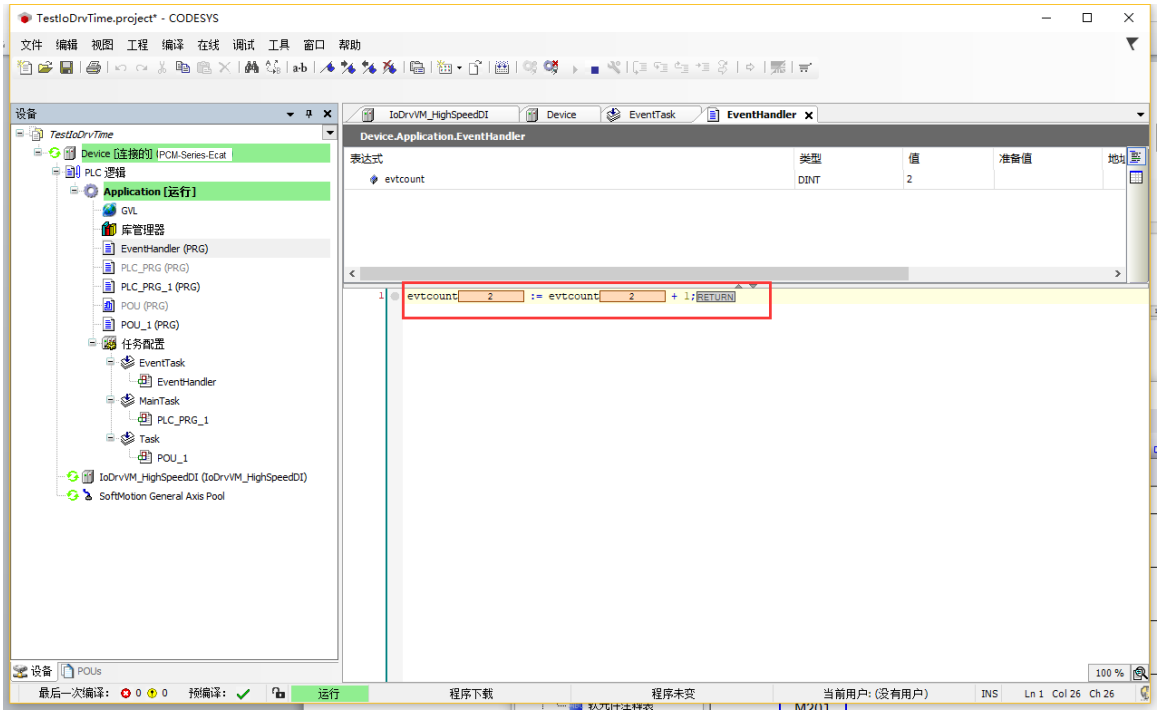
5

任务添加后，新建一个任务处理程序，并将程序添加到任务中。此处给出的示例程序是一个计数程序，即事件每调用一次，计数都会加一。





事件配置完成并添加了事件处理程序后，将工程下载到 PC5M 并运行。如下图，事件被触发了两次。



中型 PLC

PC4M 系列

用户手册

PC4M-MC100\101\102EC

VMMORE

www.vmmore.com