深圳市雷赛控制技术有限公司
SHENZHEN LEADSHINE CONTROL TECHNOLOGY CO.,LTD

PMC600 中型 PLC 用户手册（二）

编程软件篇

2021 年 06 月

版权说明

本手册版权归深圳市雷赛控制技术有限公司所有，未经本公司书面许可，任何人不得翻印、翻译和抄袭本手册中的任何内容。

本手册中的信息资料仅供参考。由于改进设计和功能等原因，雷赛公司保留对本资料的最终解释权，内容如有更改，恕不另行通知。

调试机器要注意安全！用户必须在机器中设计有效的安全保护装置，在软件中加入出错处理程序。否则所造成的损失，雷赛公司没有义务或责任负责。

目　　录

第1章　ST 语言编程入门

ST 语言即结构化文本编程语言（Structured Text）是 IEC 61131-3 标准支持的五种语言之一，专为可编程逻辑控制器（PLC）设计。它是一种块结构的高级语言，在语法上与 Pascal 语言相同，和 C 语言相似。

注意：ST 语言中字母不分大小写！

每条语句要由分号“;”结尾。

1.1. ST 语言的数据类型

A. 常用的数据类型有：

BOOL:	逻辑量,TRUE 或 FALSE
INT:	16 位有符号整数
DINT:	32 位有符号整数
REAL:	32 位实数
LREAL:	64 位实数
STRING:	字符串
TIME:	时间值

B. 常量

数字常量可以定义为一个二进制、十六进制和十进制数。二进制和 16 进制的数有前缀 2#、16#。

```
例：   2#1001_0011    // 二进制数
       16#A2          // 十六进制
       3.14159        // 十进制
```

字符串常量用单引号定义。例：’ enable ’

时间常量：以 t#为前缀，时间单位有：d – 天、h – 时、m – 分、s – 秒、ms – 毫秒

```
例：   T#100S12ms
       t#12h34m15s
```

C. 变量

变量名必须以字母开头，可以加入数字。

使用变量必须先定义变量。本地变量只能在一个子程序、或一个函数中使用；全局变量可在所有子程序和函数中使用。定义变量时可赋初值。

本地变量定义的格式：

```
VAR
    i: INT;
    Speed: LREAL := 20;
END_VAR
```

全局变量定义的格式：

```
VAR_GLOBAL
    IN1: BOOL;
    Wcount: INT ;
    motorState: STRING := 'Stop';
END_VAR
```

变量类型转换的函数有很多。通用的转换函数为：ANY_TO_<任意数据类型>

示例：

```
// 从 REAL 数据类型到 INT 类型的转化
re : REAL := 1.234;
i : INT := ANY_TO_INT(re);
```

D. 数组

一维数组的定义格式：数组名：ARRAY[0..n] OF 类型；　　//该数组的长度为 n+1

二维数组的定义格式：数组名：ARRAY[0..m,0..n] OF 类型；//（m+1）行×（n+1）列

例：

定义：

```
LineEnd: ARRAY[0..2] OF LREAL := [0,0,0]; //定义时可以赋初值
answer:ARRAY[0..3,0..3] OF INT;
```

使用：

```
LineEnd[1]:= 10;
answer[i,1]:= hen;
```

1.2. 运算符

常用运算符及其优先级如表 1.1 所示。

表 1.1.　运算符及其优先级

优先级	运算符号	描述
1	（……）	括号
2	函数名（……）；	函数的调用
3	- NOT	负号 非
4	* / MOD	乘 除 求余数
5	+ -	加 减
6	<，<=，>，>=	小于，小于等于，大于，大于等于
7	=，<>	等于，不等于
8	AND XOR OR	与 异或 或

1.3. 常用的数学函数

常用的数学函数如表 1.2 所示。

表 1.2. 常用的数学函数

函数名	描述	单位
ABS（x）	绝对值	
SQRT（x）	平方根	
LN（x）	自然对数	
LOG（x）	10 为底的对数	
EXP（x）	指数	
EXPT（y，x）	幂	计算 y 的 x 次方
SIN（x）	正弦	角度单位为弧度
COS（x）	余弦	
TAN（x）	正切	
ASIN（x）	反正弦	函数值的单位为弧度
ACOS（x）	反余弦	
ATAN（x）	反正切	

1.4. 常用语句

A. 赋值语句

:=

注：ST 语言中的赋值语句和 C 语言不同，等号前多一个冒号。

例：DelayOn := TRUE;

L := r0 - r0*COS(PI/4);

B. 条件语句

```
IF 条件表达式 THEN
    ……
  ELSE
    ……
END_IF;
```

例：

```
IF sum = 100.0  THEN  // 注意：不要写成 sum := 100 ！否则软件不会报错，程序不对。
    i:=i+1;
  ELSE
    i:=i-1;
END_IF;
```

C. 循环语句

FOR 循环变量 := 初值 TO 终值 BY 步长 DO

……

```
END_FOR;
```

注：如果步长为 1，BY1 可省略

D. 条件循环语句

```
WHILE 条件表达式 DO
……
END_WHILE;
```

E. 开关语句

开关语句是一种多分支选择语句。格式如下：

```
CASE <变量> OF
      <数值 1>: <指令块 1>
      <数值 2>: <指令块 2>
      <数值 3, 数值 4, 数值 5>: <指令块 3>
      <数值 6 .. 数值 10>: <指令块 4>
      ……
      <数值 n>: <指令块 n>
   ELSE <其他指令块>
END_CASE;
```

例：

```
CASE INT1 OF
   1, 5: BOOL1 := TRUE;     // 如果 INT1 = 1 或 5，BOOL1 = TRUE，BOOL3 = FALSE
         BOOL3 := FALSE;
      2: BOOL2 := FALSE;  // 如果 INT1 = 2，BOOL2 = FALSE，BOOL3 = TRUE
         BOOL3 := TRUE;
 10..20: BOOL1 := TRUE;    // 如果 INT1 在 10~20 内，BOOL1 = TRUE，BOOL3 = TRUE
         BOOL3:= TRUE;
   ELSE
      BOOL1 := NOT BOOL1;       // 否则，BOOL1 取反，BOOL2 = BOOL1 OR BOOL2
      BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

F. EXIT 语句

在 FOR、WHILE 或 REPEAT 循环语句中执行 EXIT 语句，循环将立即停止。

F. CONTINUE 语句

在 FOR、WHILE 或 REPEAT 循环语句中执行 CONTINUE 语句，循环将立即开始下一次循环。

```
FOR Counter:=1 TO 5 BY 1 DO
    INT1:=INT1/2;
    IF INT1=0 THEN
        CONTINUE;     // 避免除数为零
    END_IF
    Var1:=Var1/INT1;  (* 除数不为零时执行 *)
END_FOR;
```

G. RETURN 语句

退出子程序或函数。

H. 调用功能块、子程序、函数的语句

格式： 功能块或子程序名（参数 1，参数 2，……）；

变量 := 函数名（参数 1，参数 2，……）；

I. 注释符号

单行注释符号为：“//”；多行注释符号为“(*” 语句块 “*)”。

J. iStudio 软件帮助文件

在雷赛的 iStudio 软件中，有完整的帮助文件。其中有关 ST 语言比较完整的内容在目录的“CODESYS Development System”→“参考编程”→“编程语言和他们的编辑器”中，如图 1.1 所示。用户有必要仔细阅读。

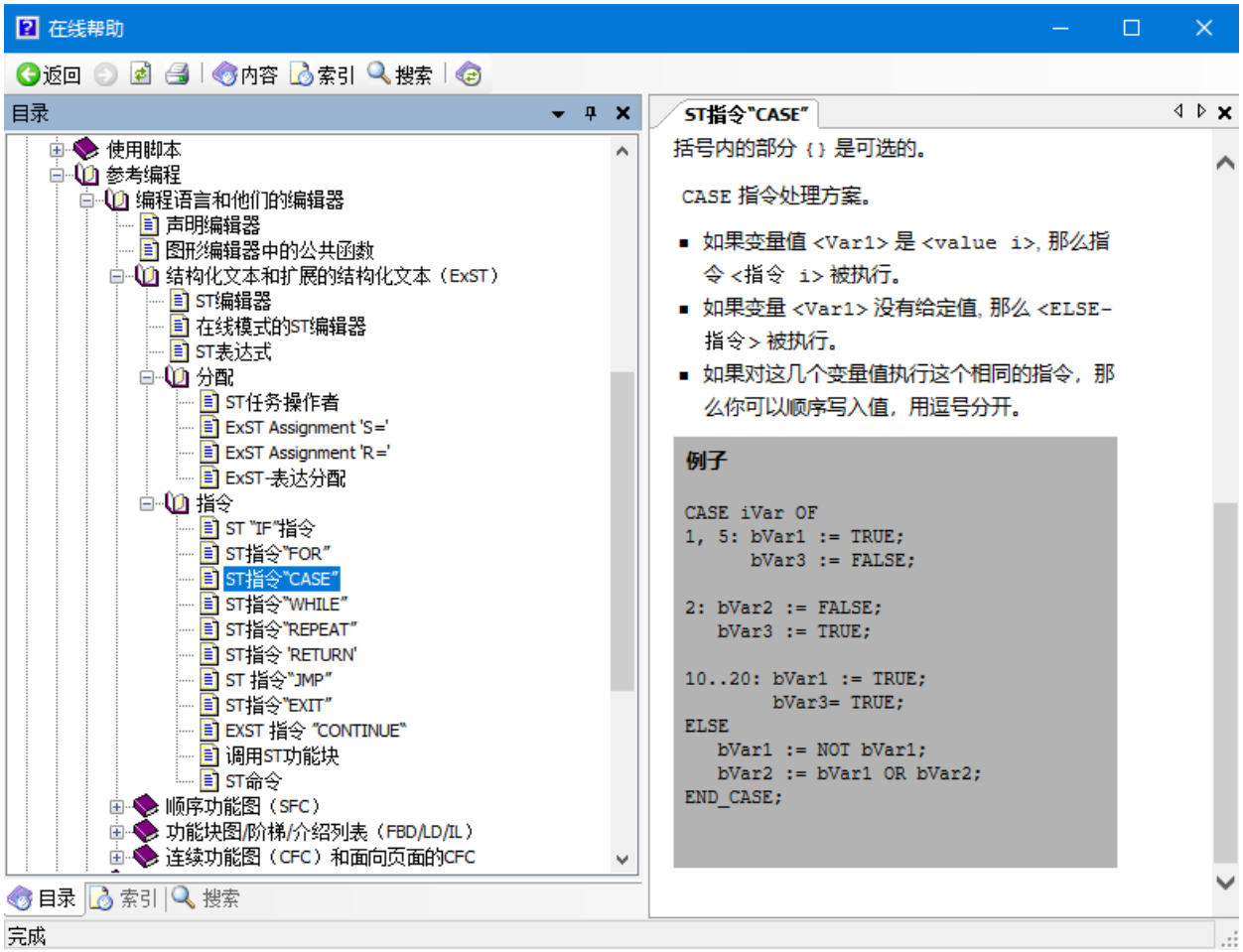

图 1.1 雷赛 iStudio 软件帮助文件中与 ST 语言相关内容的目录

1.5. ST 语言与 C 语言的区别

求解“百钱百鸡”问题。鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

ST 程序代码如下：

```
PROGRAM PLC_PRG
VAR
    cock,hen,chick,i:INT;                // 定义整数型变量
    sum:REAL;                            // 定义实数型变量
    answer:ARRAY[0..3,0..3] OF INT;      // 定义整数型二维数组 4X4
    flag:BOOL:=TRUE;                     // 定义布尔型变量且赋初值
END_VAR

IF flag=TRUE THEN                                  // 只计算一次
        flag:=FALSE;
        i:=0;
        FOR cock := 1 TO 19 DO                     // 公鸡数量在 1～19 之间变化
            FOR hen := 1 TO 31 DO                  // 母鸡数量在 1～31 之间变化
                chick := 100-cock-hen;             // 小鸡数量为 100 - cock - hen
                sum := cock*5+hen*3+chick/3.0;     // 计算购鸡的总价格
                IF sum = 100.0  THEN               // 如果总价格 sum = 100
                    answer[i,0]:=cock;             // 保存结果
                    answer[i,1]:=hen;
                    answer[i,2]:=chick;
                    i:=i+1;                        // 解的数量加 1
                END_IF;
            END_FOR;
        END_FOR;
END_IF;
```

该程序是一个死循环程序。如果没有 If flag = true 和 flag := false 这几条语句，PMC 运动控制器将反复计算该段程序。

如果将上述 ST 语言编写的百钱百鸡程序翻译成 C 语言，结果如下：

```
long cock,hen,chick;
double sum;
string result;

while(true)
{
  If (flag == true )
     {
      flag = false;
      for (cock = 1; cock <= 19; cock = cock + 1)      // 公鸡数量在 1～19 之间变化
         {
          for (hen = 1; hen <= 31; hen++)             // 母鸡数量在 1～31 之间变化
             {
              chick = 100 - cock - hen;               // 小鸡数量为 100- cock-hen
              sum = cock * 5 + hen * 3 + chick / 3;   // 计算购鸡的总价格
              if (sum == 100)                         // 如果总价格 sum  = 100
                {
```

```
                    result = "cock = " + Convert.ToString(cock);
                    result = result + "  hen = " + Convert.ToString(hen);
                    result = result + "  chick = " + Convert.ToString(chick) + "\n";
                    richTextBox1.AppendText(result);    // 输出符合“百钱百鸡”条件的结果
                }
            }
        }
    }
}
```

注意：ST 编写的程序执行过程和 PLC 的梯形图一样，是反复扫描运行的。

在编写 PMC600 运动控制器的程序时，一定要注意程序的运行时间，尤其是使用循环语句的时候要特别注意，程序执行一遍的时间尽量不要超过 4 毫秒，否则，EtherCAT 总线型伺服电机有可能出错。

1.6. ST 与 FBD 混合编程

编写 ST 语言时，为了便于程序阅读、简化代码输入，有些人习惯以 FBD、LD 的形式调用功能块。

FBD（功能块图 Function Block Diagram）是一种图形化编程语言。它由一些“节”组成，每个“节”由一些功能块组成。如图 1.2 所示。

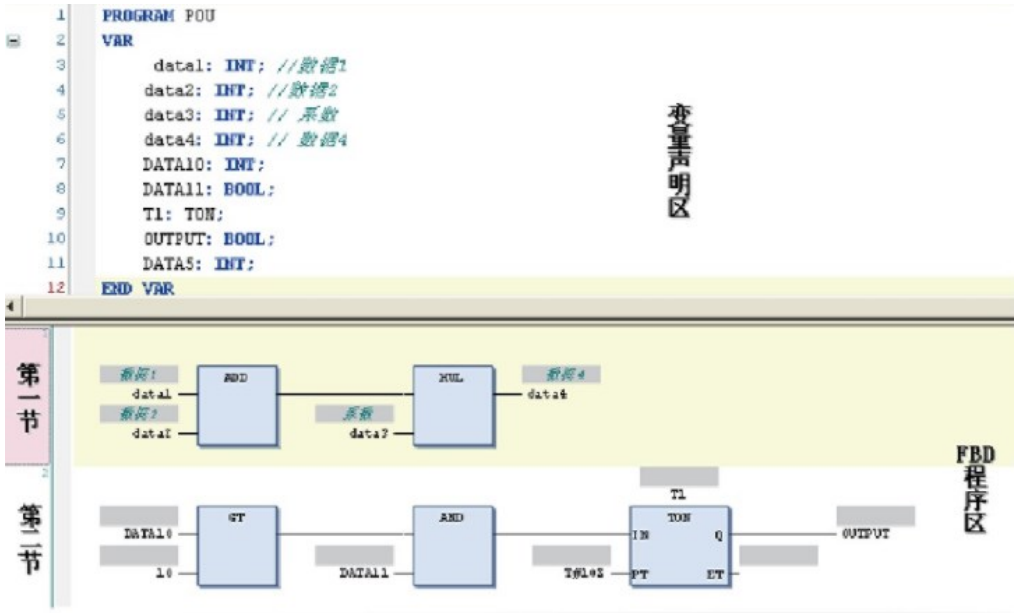

图 1.2 FBD 语言格式

如图 1.3 所示，以 FBD 的形式设置了电机使能模块 MC_Power，并在主程序中调用，如图 1.4 所示。

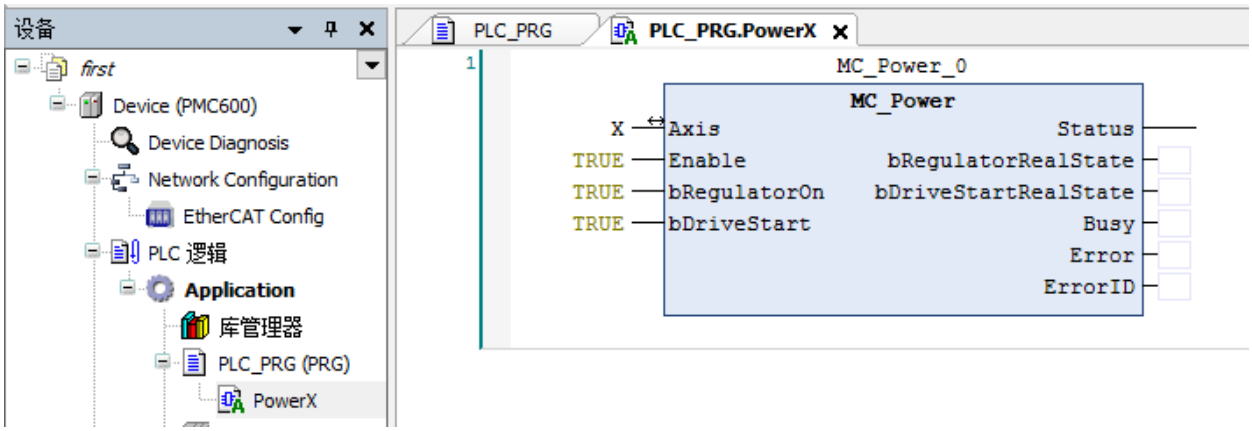

图 1.3　用 FBD 语言设置功能模块 PowerX

图 1.4　主程序中调用 FBD 语言的模块 PowerX

第2章　iStudio 软件简介与安装

2.1　iStudio 特色

iStudio 软件基于 CODESYS V3.5 开发，包括 PLC 编程、可视化 HMI、安全 PLC、控制器实时核、现场总线及运动控制，是一个完整的自动化软件。

iStudio 软件还结合了雷赛特有的运动控制功能，将多种复杂运动控制功能封装成工艺库，供用户使用。

- 采用国际标准 IEC61131-3 架构，支持梯形图 LD、指令表 IL、结构化文本 ST、功能块图 FBD、顺序功能流程图 SFC、控制流程框图 CFC 等多种编程语言；
- 强大的软件仿真、在线调试及程序检查能力，不需要连接 PMC 硬件即可实现程序的调试和仿真
- 具备方便的产品配置功能，可以轻松快速的实现包括 CPU 配置、通信配置、本地 IO 功能配置、EtherCAT 总线配置等。
- 提供丰富的运动控制功能库和行业工艺库，涵盖多种过程控制和运动控制的应用场景。

2.2　iStudio 软件架构介绍

iStudio 功能强大，可靠性高，开放性好，并且集成了 PLC、可视化、运动控制及安全 PLC。iStudio 软件从架构上可以分为 3 层：应用开发层、通信层和设备层，如图 2.1 所示。

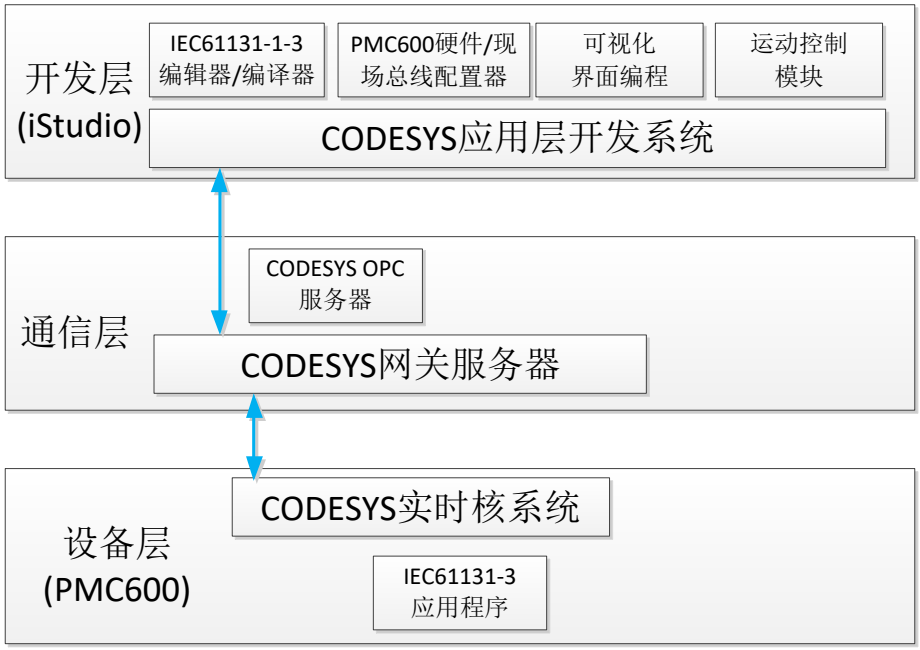

图 2.1　iStudio 架构示意图

2.2.1　开发层

iStudio 软件中集成了 IEC61131-3 编辑器（具有在线编程和离线编程功能）、编译器及其配件组件、可视化界面编程组件、同时集成的运动控制模块可使其功能更加完整和强大。

（1）IEC61131-3 编辑器：iStudio 提供了所有 IEC61131-3 所定义的 6 种编程语言：梯形图 LD、指令表 IL、结构化文本 ST、功能块图 FBD、顺序功能流程图 SFC、控制流程框图 CFC；

（2）编译器：负责将应用程序转换为机器代码并且优化可编程控制器的性能。当用户输入了错误的应用程序代码时，立刻回接收到编译器发出的语法错误警告及错误信息，让编程人员可以迅速做出相应纠正。

（3）硬件/现场总线配置器：针对 PMC600 的硬件设备及不同现场总线协议，对相应的参数进行设定。

（4）可视化界面编程；直接在软件中即可实现可视化编程（人机界面）,系统已经集成了可视化编辑器。

（5）运动控制模块：运动控制功能已经集成在软件中，形成了 SoftMotion(CNC)软件包。基于 PLCopen 的工具包可以实现单轴、多轴运动，以及电子凸轮传动、电子齿轮传动及复杂多轴 CNC 控制等。基于雷赛的运动控制库，可以获得多种成熟的行业应用功能。

2.2.2 通信层

应用开发层和设备层之间的通信是由网关服务器来实现的，网关服务器中安装了 OPC 服务器。

CODESYS 网关服务器

作用在应用开发层和设备硬件层之间，可以使用 TCP/IP 或通过 CAN 等总线实现远程访问，是 CODESYS 开发工具包中不可分割的一部分。

CODESYS OPC 服务器

对基于 CODESYS 进行编程的控制器，无需考虑所使用的的硬件 CPU，已经集成并实现了 OPC V2.0 规范的多客户端功能，并且能同时访问多个控制器。

2.2.3 设备层

PMC600 系列中型 PLC，是系统的设备层，已经预先安装 CODESYS 实时核，能保证进行实时响应和精确的控制。

（1）实时核系统；

（2）IEC61131-3 应用程序：用户在开发层编写完的程序通过以太网或串口下载至设备层中，最终该应用程序的文件被转换为二进制存放在 PMC600 中，根据用户设定的执行方式循环执行对应的程序。

2.3 iStudio 支持的功能汇总

iStudio 支持雷赛多款 PMC 设备，为了方便用户查询，此章节内容将会把 iStudio 支持的 PMC 型号种类，编程语言种类以及 PMC 系列所有支持的功能做一个汇总说明。

2.3.1 iStudio 支持的 PMC 主机种类

表 2.1 iStudio 支持的 PMC 主机种类

	PMC600	PMC610
iStudio	✔	✔

2.3.2 iStudio 支持的编程语言种类

iStudio 支持 IEC61131-3 标准的编程语言，共分为两大类编程语言：文本化编程语言和图形化编程语言。文本化编程语言包括指令表编程语言（Instruction List，IL）和结构化

文本编程语言（Struct Text,ST），图形化编程语言包括梯形图编程语言（Ladder Diagram，LD）和功能块图编程语言（Function Block Diagram，FBD）以及顺序功能图（Sequence Function Chart，SFC）、连续功能图（Continuous Function Chart，CFC）。

表 2.2　iStudio 支持的编程语言种类

	PMC600	PMC610
IL	√	√
ST	√	√
LD	√	√
FBD	√	√
SFC	√	√
CFC	√	√

2.3.3　iStudio 支持的功能列表

表 2.3　iStudio 支持的功能种类

	PMC600	PMC610
SM3_Basic	√	√
电子齿轮	√	√
电子凸轮	√	√
SM3_CNC	√	√
PLCOpen PART 1	√	√
PLCOpen PART 2	√	√
PLCOpen PART4	√	√
高速锁存	X	√
高速比较	X	√
PWM	X	√

2.4 iStudio 的安装与管理

2.4.1 安装 iStudio

由于 iStudio 软件是基于 CODESYS V3.5 软件开发，需要处理的数据较多，因此对 PC 的硬件及系统环境有一定要求，其要求的最低配置及推荐配置见表 2.4。

表 2.4 软件安装最低配置及推荐配置

描述	最低配置	推荐配置
操作系统	Windows 7/8/10	Windows 10(64 位)
内存	2GB	4GB
硬盘空间	2GB	4GB
处理器	Pentium V，Centrino > 1.8GHz，Pentium M > 1.0GHz	Pentium V，Centrino > 3.0GHz，Pentium M > 1.5GHz
注意：为了能有更好的操作体验，我们建议您使用 64 位操作系统，在 32 位系统中可能会出现部分界面显示异常。		

直接双击运行”istudio”安装文件即可进入安装，整个安装过程中安装助手都会引导用户进行安装，图 2.2 为双击安装文件后弹出安装向导界面。

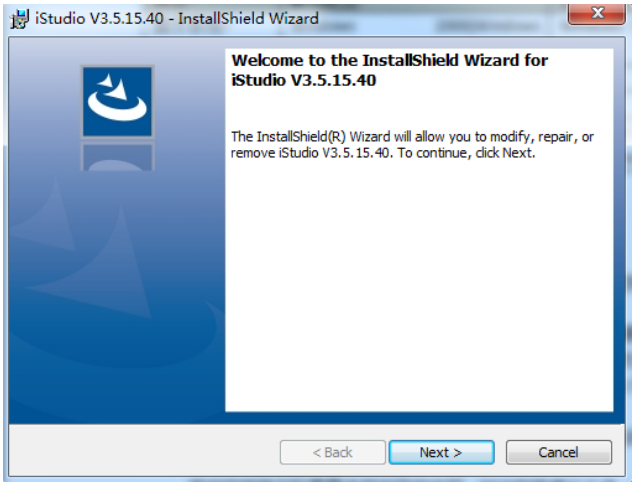

图 2.2 iStudio 安装向导

在软件正式安装之前，程序会对安装环境进行检测，必须要有 Microsoft Visual C++ 及 Microsoft .NET Framework 4.6 安装环境，如之前没有安装过，改应用程序会自动安装这部分软件。安装前需要同意遵守 3S 公司的软件使用规范，如图 2.3 所示，同意后即可进入下一步。

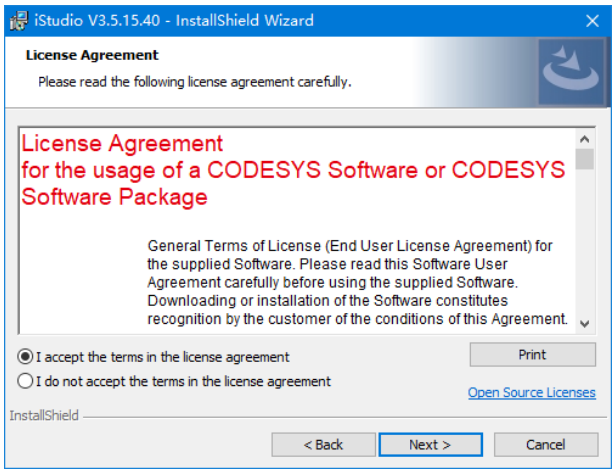

图 2.3　软件使用规范

程序默认的安装路径为“C:\Program Files\iStudio 3.5.15.40\”，如果需要修改，则可通过单击“Change…”按钮重新定义，如图 2.4 所示。

用户可根据实际需要选择安装的内容，初次安装建议全选。当然，用户可以根据实际需求进行安装，鼠标选中特定功能后，在右侧的“Feature Description”中会找到对应的介绍内容，如图 2.5 所示。

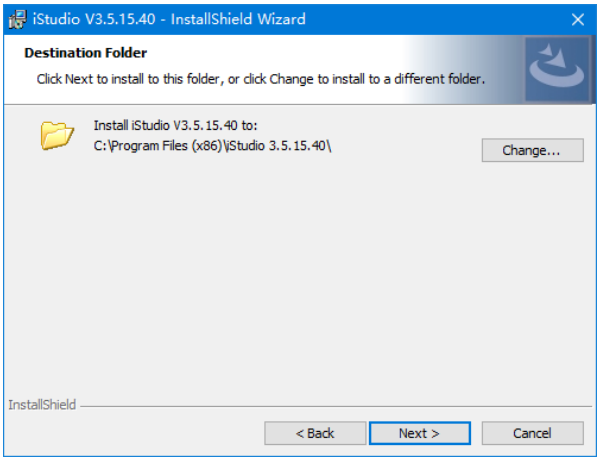

图 2.4　软件安装路径

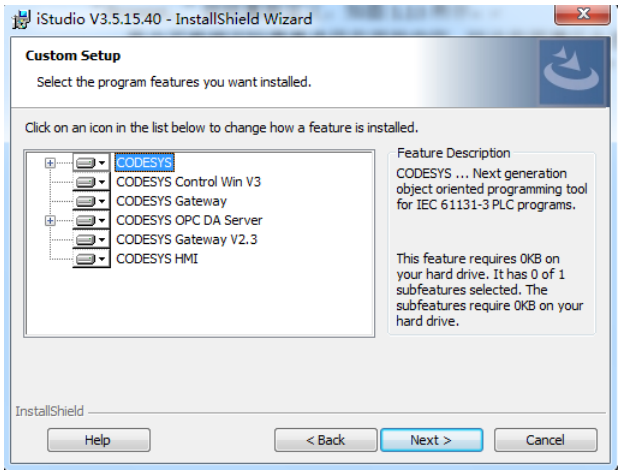

图 2.5　iStudio 可选功能

完成上一步骤后，即可等待软件安装完成，如图 2.6 所示。

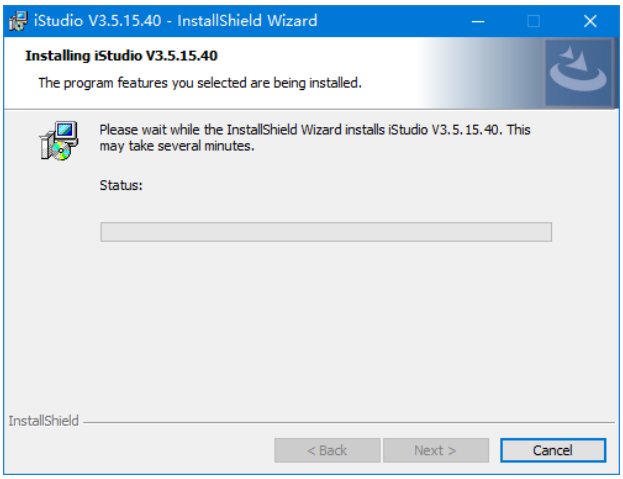

图 2.6　软件安装过程

2.4.2　iStudio 版本管理

在 iStudio 中可以同时安装一个组件的多个不同版本，并且可以组合使用。程序编译器也可以安装与使用多个版本，而且无需更新整个版本就可以新增独立的功能。

2.5　开始使用 iStudio

进入“开始”菜单，选择“程序”->“Leadshine” ->“iStudio v3.5.15.4”，或者，当安装完成之后，可以直接在桌面找到 iStudio 的图标  ，双击打开。

PLC Development System 是整个自动化开发平台的核心，它几乎包含了一个先进的自动化开发工具所应具有的所有功能。图 2.7 为 iStudio 的用户界面，标准组件主要有菜单

栏、工具栏、编辑窗口、设备窗口、监视窗口、消息窗口和在线模式等。现面对用户开发环境进行详细介绍。

在软件中，所有的窗口及视图都不是固定的，用户可以根据自己的习惯将窗口和视图通过鼠标拖曳的方式移动到目标位置，将窗口和视图进行重新排列。

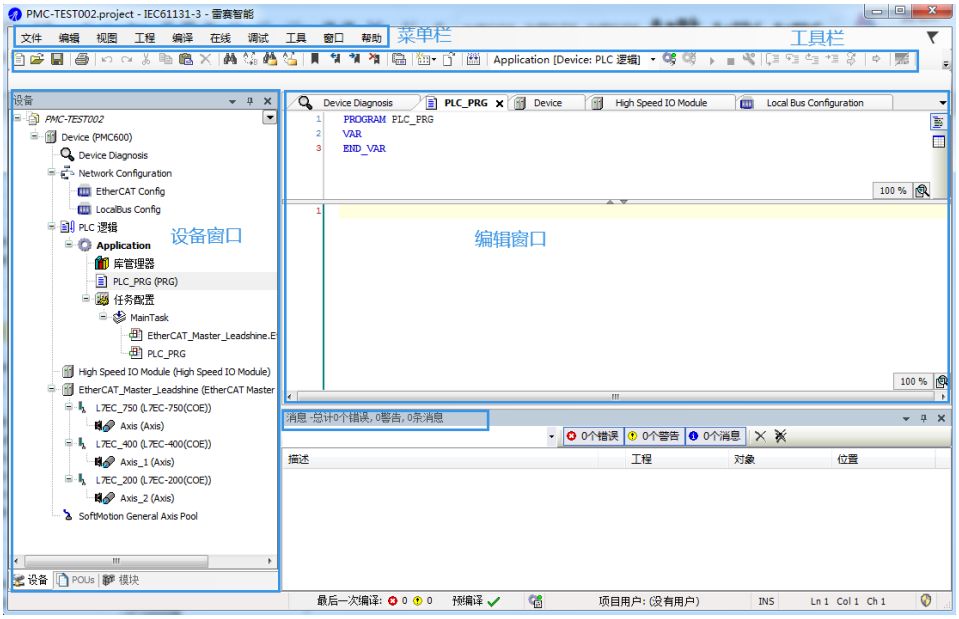

图 2.7　istudio 用户界面

2.5.1　菜单栏

在 iStudio 中，菜单栏是使用最为频繁的操作选项，项目的新建及保存，程序编译、登入及下载，调试时的设置断点及强制写入等操作都需要通过菜单栏里的功能来实现。在 iStudio 中，菜单栏能实现的功能见表 2.5。

表 2.5 菜单栏功能列表

菜单名	内容
文件	对工程文件进行操作（打开、关闭、保存、打印、页面设置、下载/上传源代码等）
编辑	编辑器（如语言编辑器、声明编辑器）操作
视图	激活特定的标准视图，如在用户界面的某个窗口中显示视图，与窗口菜单功能类似
工程	编辑工程对象和工程基本信息、复制工程、合并工程、导出工程、配置库以及用户管理
编译	编译工程，例如：（1）包含语法检查的预编译运行；（2）当采用在线修改和离线代码生成的方法时，可以删除上一次的编译信息（清空）
在线	登入、退出控制器，加载控制器上的工程和复位
调试	控制运行在控制器上程序（启动、停止）和调试操作（断点、单步、写入、强制）
工具	该菜单包含的命令可以打开工具，这些工具用来配置工程的操作环境（如库的设备的安装、用户界面自定义、编辑器选项、加载和保存）
窗口	操作用户界面中的各个窗口（如排列、打开、关闭等命令）。与视图菜单功能类似

帮助	打开在线帮助，获取系统帮助信息

2.5.2 工具栏

通过单击图标，用户可以更快地选择相应的命名。可以选择的图标将自动与激活的窗口相适应。仅当鼠标在图标上单击然后释放时，才能执行命令。如果用户将鼠标指针短时停留在工具栏的一个图标上时，则会在工具提示中显示该图标的名称。工具栏如图 2.8 所示。

图 2.8 工具栏

当用户选择不同的对象时，工具栏里面的内容会略微发生改变，如使用梯形图（LD）的程序组织单元（POU）时，工具栏里会有触点、线圈等功能，而当选用功能块图（FBD）时，会出现添加功能块等选项。工具栏中的图标也不是固定的，用户可以根据自己的使用习惯自定义自其中的内容。

2.5.3 编辑窗口

编辑窗口用于在相应的编辑器中创建特定的对象。一般所说的语言编辑诶（如 ST-编辑器、CFC-编辑器）是指语言编辑器和声明编辑器的组合，通常在下部是语言编辑器，上部是声明编辑器。在其他编辑器中，还可以提供对话框（如任务的编辑器、设备编辑器）。POU 或资源对象的名称时钟显示在窗口的标题栏中。在离线或在线模式下通过“编辑对象”命令可以在编辑器窗口中打开对象。

2.5.4 设备窗口

以树型结构管理工程中的资源对象。在项目中，数据分层结构中以对象的形式进行保存。

2.5.5 监视窗口

显示一个 POU 的监视视图，当程序登入后，可以用来监视 POU 中任意的表达式。

2.5.6 消息窗口

消息窗口显示预编译、编译、生成器、下载和程序检查等信息。如果用鼠标双击消息窗口内的一条消息或按右键选择“转移到源代码处”，编辑器就会打开锁选择的对象。通过右键快捷菜单中的选项“后一个消息”（<F4>键）及“前一个消息”（<Shift+F4>组合键），可以在各个错误消息之间快速跳转。消息窗口的显示是滚动的。消息窗口示例如图 2.9 所示。

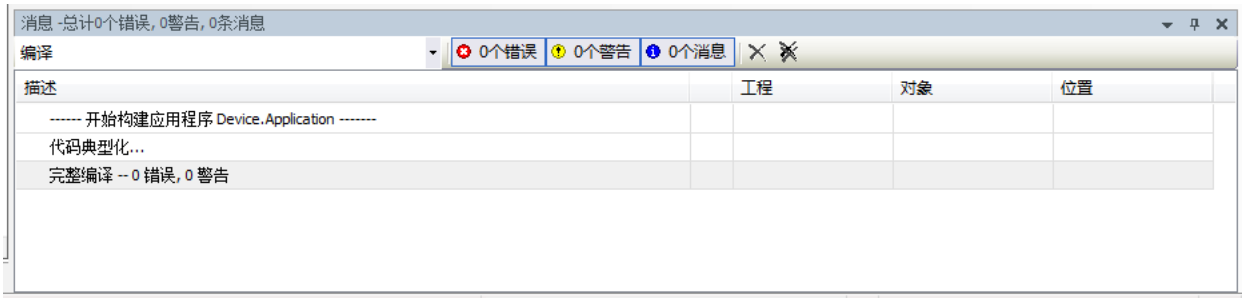

图 2.9　消息窗口

2.5.7 在线模式

表 2.6　在线模式状态描述

状态	描述
运行	程序正在运行
停止	程序已经停止
停止到 BP	程序停止于断点
仿真	iStudio 目前正处于无硬件仿真状态
程序下载	程序已经下载到设备上
程序未变	设备上的程序与编程系统中的相符
修改程序（在线改变）	设备上的程序与编程系统中的不同，需要在线更改

2.5.8 帮助

用户在打开应用程序后，可以找到“帮助”菜单，单击“目录”或者直接在主界面按“F1”可以打开在线帮助。用户可以根据索引或者搜索关键字找到所需要的内容，如图 2.10 所示。

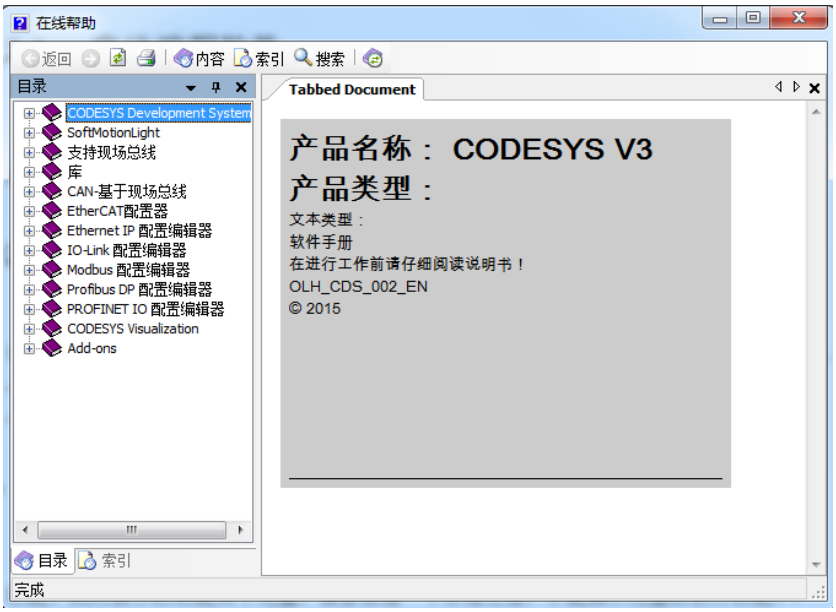
在线帮助
返回
内容
索引
搜索
目录
CODESYS Development System
SoftMotionLight
支持现场总线
库
CAN-基于现场总线
EtherCAT配置器
Ethernet IP 配置编辑器
IO-Link 配置编辑器
Modbus 配置编辑器
Profibus DP 配置编辑器
PROFINET IO 配置编辑器
CODESYS Visualization
Add-ons
目录
索引
Tabbed Document
产品名称： CODESYS V3
产品类型：
文本类型：
软件手册
在进行工作前请仔细阅读说明书！
OLH_CDS_002_EN
© 2015
完成

图 2.10　在线帮助

第3章 iStudio 工程结构

本章从一个 iStudio 工程的角度介绍工程的主体结构和各个对象属性。

3.1 iStudio 软件模型

3.1.1 软件模型简介

iStudio 软件的模型和 CODESYS 软件一致，采用分层结构表示，每一层隐含其下面层的许多特性。软件模型描述了基本的软件元素机器相互关系，这些软件元素包含设备、应用、任务、全局变量、访问路径和应用对象，它们是现代 PLC 的软件基础，其内部结构如图 3.1 所示，该软件模型与 IEC61131-3 标准的软件模型保持一致。

上述软件模型从原理上描述了如何将一个复杂设备分解为若干小的可管理部分，并在各部分之间有清晰规范的接口方法。软件模型描述了一台可编程控制器如何实现多个独立程序的同时运行，如何实现对程序执行的完全控制。

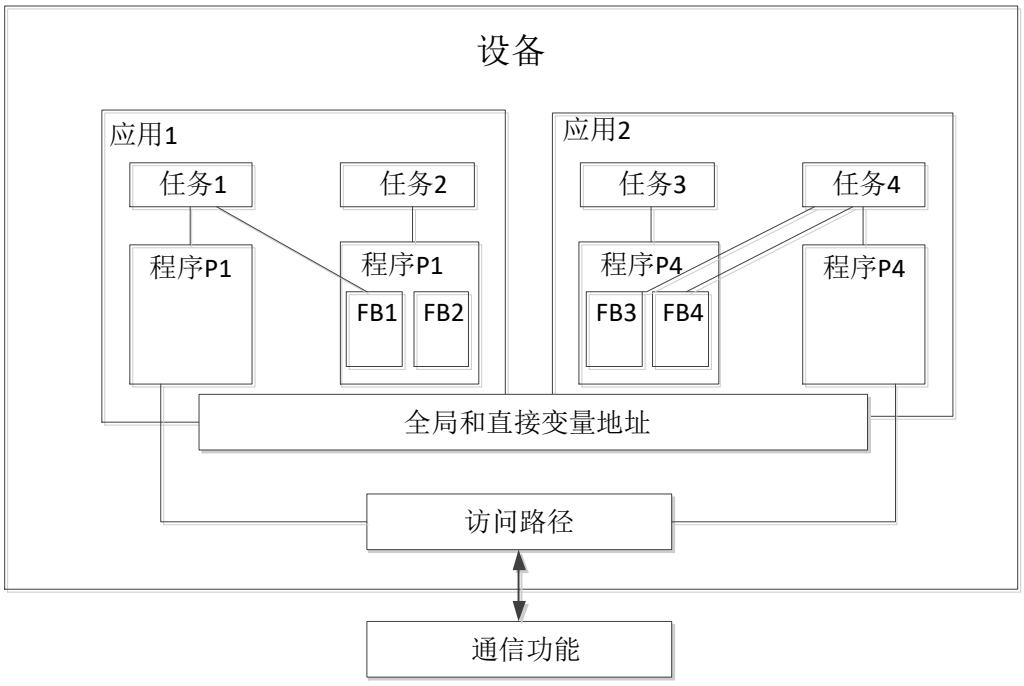

图 3.1 iStudio 软件模型结构

3.1.2 设备

在模型的最上层，设备可以等效于一个 PLC 所需要的所有软件的集合。针对大型复杂

的应用系统，如整个产品线的自动化，可能需要多个 PLC 联机通信，可将一个 PLC 与其他多个设备接口实现总线通信。这是设备可以理解为一个特定类型的控制系统，它具备硬件装置、处理资源、IO 地址映射和系统内存存储的能力，即等同于一个 PLC。

3.1.3 应用

在 PLC 系统中，设备将所有“应用”结合成组，为“应用”提供数据交互的手段。在每一个设备中都有一个或多个应用，应用位于软件模型的第二层。“应用”不仅为运行程序提供了一个支持系统，而且它反映 PLC 的物理结构，在程序和 PLC 物理 I/O 通道之间提供了一个接口。

应用被分配在一个 PLC 的 CPU 中，因此，可将应用理解为一个 PLC 中的微处理器单元。在应用内定义的全局变量在改应用内部是有效的。应用主要成员包括全局变量、任务和程序组织单元（POU）等。

3.1.4 访问路径

访问路径的主要功能是将全局变量、直接表示变量和程序组织单元（POU）的输入/输出变量联系起来，从而实现信息的存储。它提供在不同应用之间交互数据和信息的方法，每一个应用内的变量可通过其他远程配置来存取。

3.1.5 通信功能

提供与其他系统，如其他可编程控制器系统、机器人控制器、计算机等装置的数据通信，用于实现程序传输、数据文件传输、监视、诊断等功能。通常采用复合国际标准的通信方式（如 RS232、RS485）或工业现场总线（如 EtherCAT、Modbus、EtherNet/IP 等）。

3.1.6 软件模型的特点

该软件模型具有如下特点：

（1）在一台 PLC 中能同时装载、启动和执行多个独立的程序。

（2）实现对程序执行的完全控制能力：标准的任务机制，保证了 PLC 系统对程序执行的完全控制能力。

（3）支持程序组织单元的重用特性。

（4）支持分层设计：一个复杂的软件通常可以通过一层层的分解过程，最终可以分解为可管理的程序单元。

3.2 iStudio 工程

3.2.1 iStudio 工程简介

iStudio 的工程结构继承于 CODESYS 的软件模型，包括了设备、应用、通信等主要对象，同时结合了控制器的特性，新增了设备诊断、网络配置、高速 IO 配置、EtherCAT 主站等对象。图 3.2 表示了 CODESYS 原版工程的内容和 iStudio 软件集成的内容（以 PMC600 作为设备为例）的对比。

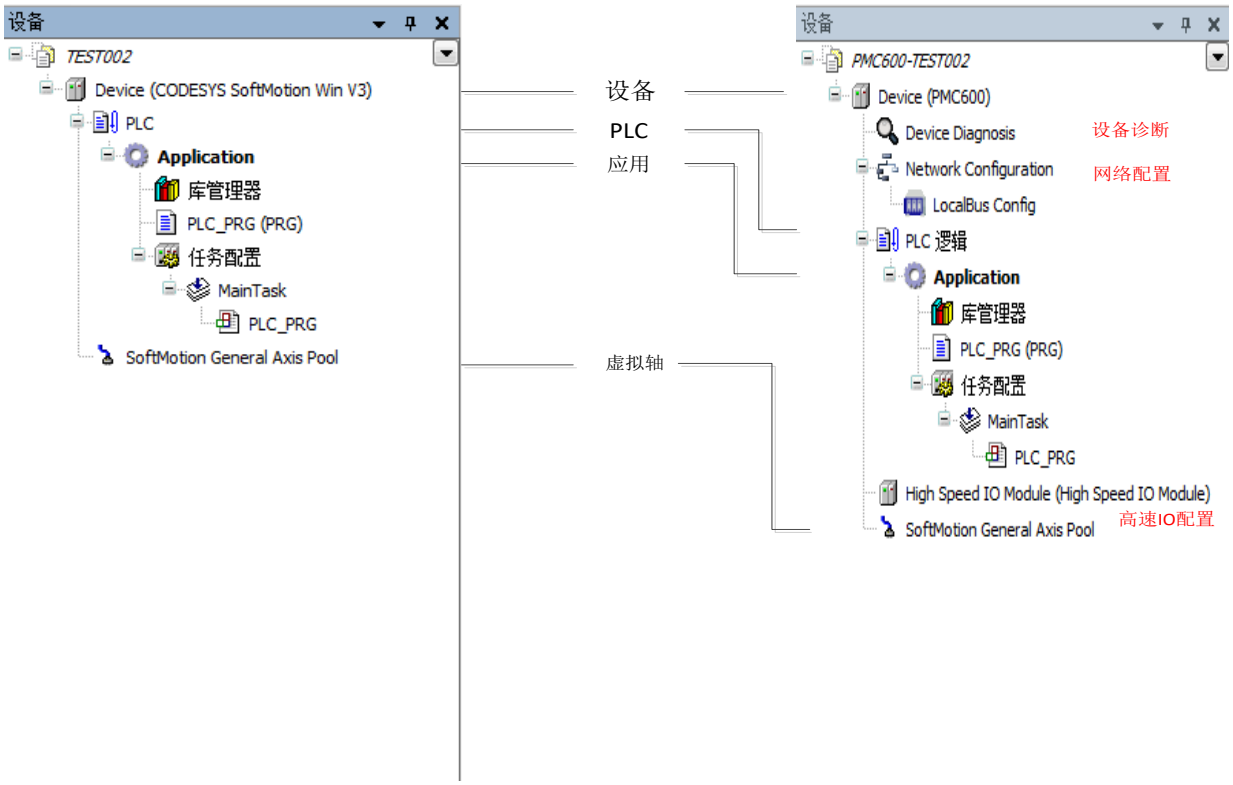

图 3.2　CODESYS 原版工程和 iStudio 软件集成内容对比图

3.2.2 新建工程

单击菜单栏的“文件”选项，选择“新建工程（Ctrl+N）”，软件会弹出如图 3.3 所示的新建工程对话框。选择“分类”中的“工程”，然后选择“模版”中的“标准工程”，之后确定工程名称以及工程所在的位置。

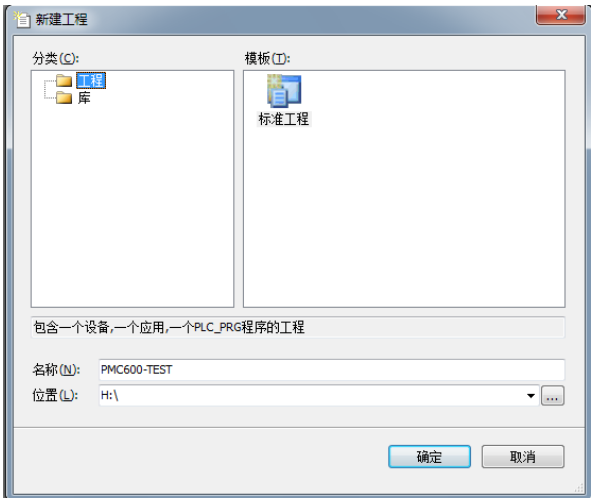

图 3.3 新建工程对话框

单击“确定”，即可弹出“标准工程向导”，在“设备”栏选择“PMC600”，选择编程语言，如图 3.4 和图 3.5 所示、此处以“结构化文本”为例进行说明。

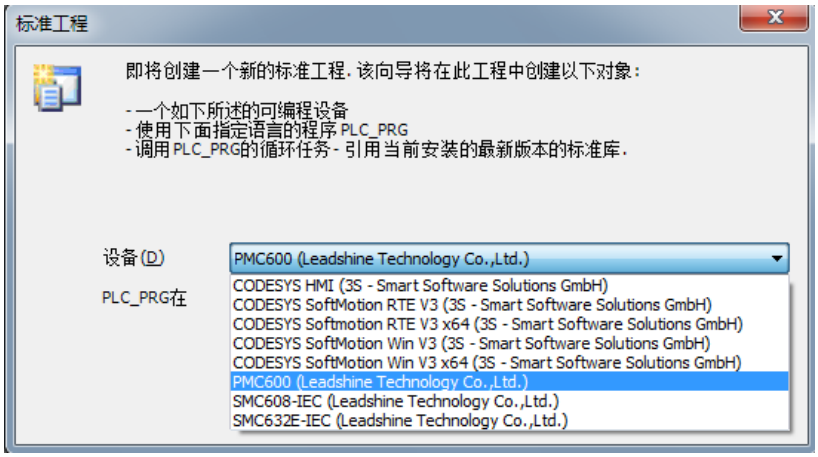

图 3.4 选择 PMC600 设备

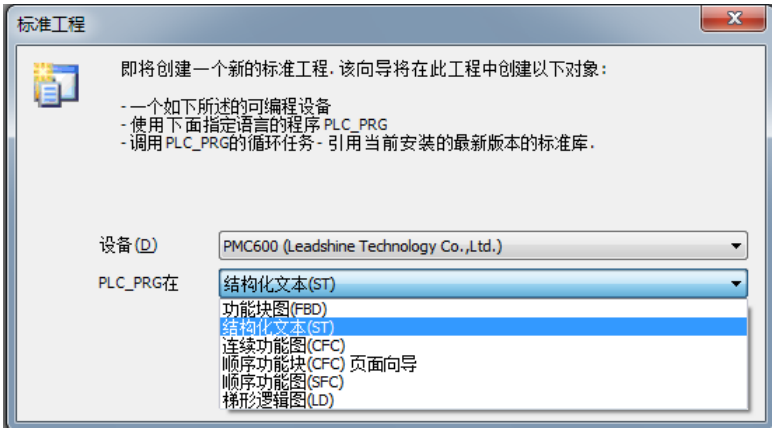

图 3.5 选择编程语言

单击“确定”按钮之后，系统会进行工程的创建，一个标准的 iStudio 工程（PMC600）如图 3.6 所示。工程以“Device”（即设备）为单位，由“设备”的各种不同类型的子项目组成。

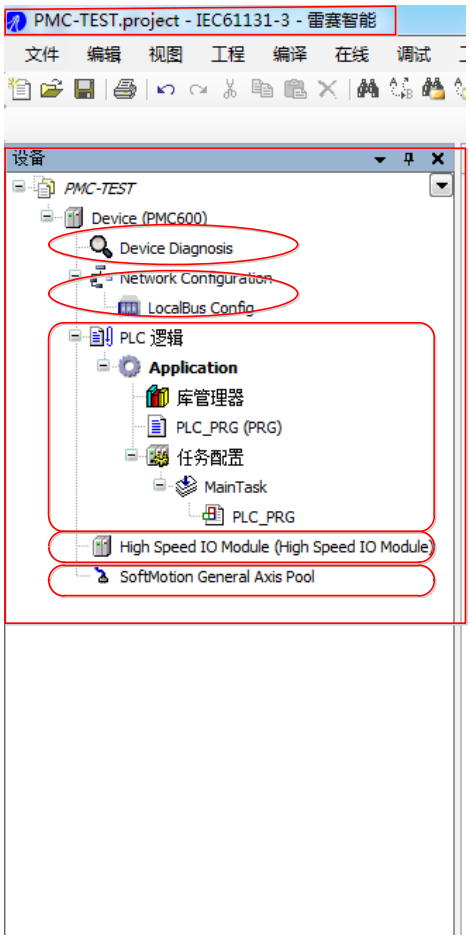

图 3.6 iStudio 标准工程界面

在设备（Device）栏，可以看到有以下子项目：

（1）Device Diagnosis：即设备诊断

（2）Network Configuration：即网络配置

（3）PLC 逻辑：

（4）High Speed IO Module：高速 IO 模块

（5）SoftMotion General Axis Pool：虚拟轴配置

以上，是一个 iStudio 标准工程里的基本配置，每一个工程都必须包含这些配置信息。

3.2.3 设备（Device）属性

“设备”（Device）是工程的操作主体，是一系列对象的集合，包括硬件设备诊断（Device Diagnosis）、网络配置（Network Configuration）、PLC 逻辑（库管理、POU、任务配置等）、本地高速 IO 配置、虚拟轴配置等。

双击“Device(PMC600)”栏，可以看到如图 3.7 所示的设备相关属性的内容。

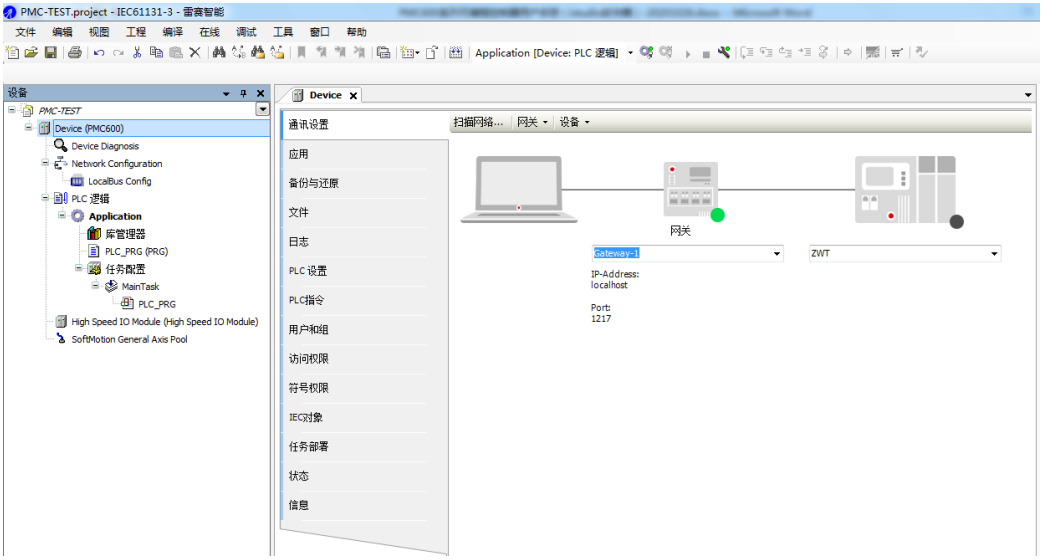

图 3.7 设备相关属性

在设备属性列表里，可以看到如图 3.7 所示子项目：

- 通讯设置
- 应用
- 备份与还原
- 文件
- 日志
- PLC 设置
- PLC 指令
- 用户和组
- 访问权限
- 符号权限
- IEC 对象
- 任务部署
- 状态
- 信息

3.2.4 通讯设置

在软件工程列表中双击“Device(PMC600)”，点击“通讯设置”，可以看到如图 3.8 所示的通讯连接界面。

图 3.8　通讯设置

3.2.5 连接 PLC

在进行软件下载，控制器状态监控等操作之前，需要先将 iStudio 软件与 PLC 进行连接。请按照以下步骤进行连接：

（1） 连接 PLC：PMC600 的四个网口中的第一个网口（从上到下的顺序）是以太网接口，对应的的接口表示是“Ethernet”，如图 3.9 所示，请使用网线连接 PC 和控制器。

图 3.9　连接 PLC

（2）扫描网络：可以通过“扫描网络”搜索在同一网段内的 PLC,找到后单击确定，如图 3.10 所示的 PLC 设备名为“PMC-600” 。

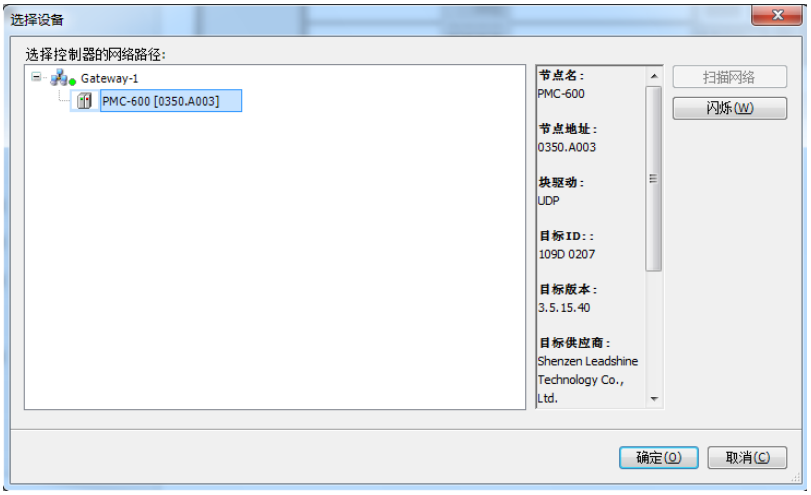

图 3.10　扫描网络

（3）连接成功后，会出现“网关”、“设备”图片的右下角出现绿色的圆形标志，说明此时网关和 PLC 设备已经处于理解状态。在设备端可以看到节点名、节点地址、目标 ID、目标类型、目标供应商、目标版本等详细信息。如图 3.11 所示。

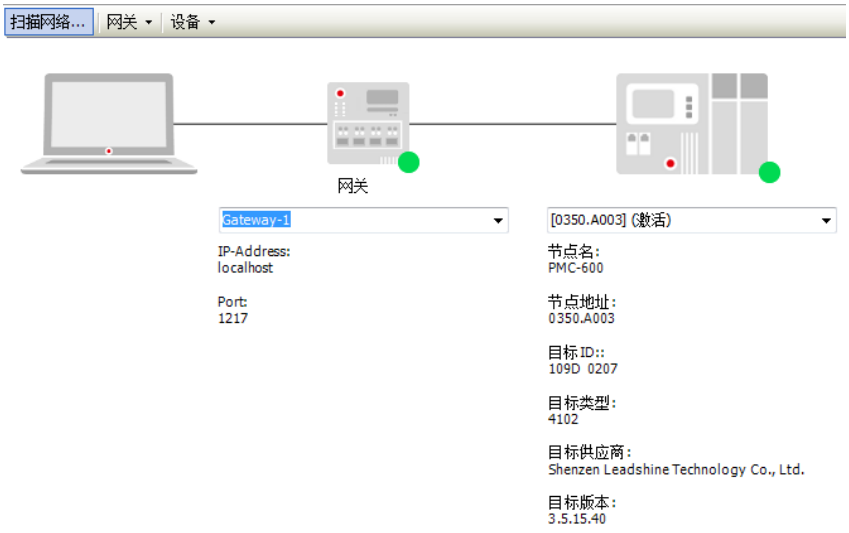

图 3.11　PLC 连接成功

用户需要注意：

（1）PC 和 PMC600 的 IP 地址需要在同一个网段，否则会出现连接不上的现象；

（2）PMC600 的 Ethernet 端口出厂默认 IP 地址是 192.168.1.3，与之相连的 PC 端的网口 IP 地址应该设置为 192.168.1.XXX（XXX 表示 1~254，XXX 不能等于 3），当控制器 IP 地址修改后，PC 端的网口地址也应做对应修改。

（3）PMC600 可以通过网线和 PC 进行一对一连接，也可以通过网络交换设备（如路

由器）实现一台 PMC600 设备和多台 PC 连接，或者多台 PMC600 设备和一台 PC 连接。

3.2.6 网络结构配置（Network Configuration）

新建一个 PMC600 的工程之后，点开“Network Configuration”节点，可以看到如图 3.12 所示 PMC600 拓扑结构。“Network Configuration”栏下有一栏默认的“EtherCAT Config”。

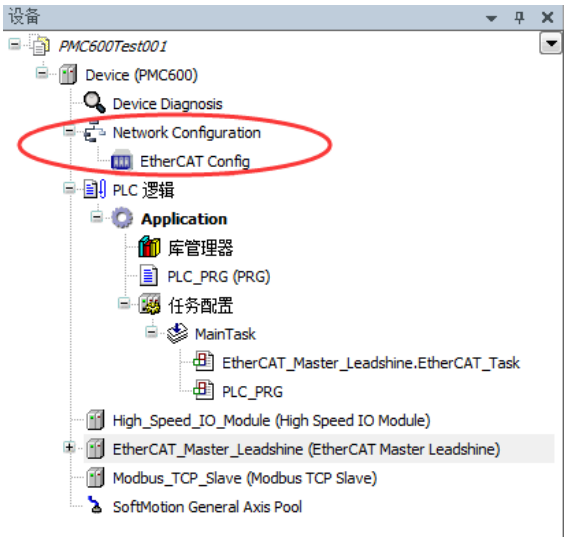

图 3.12　网络结构配置

双击“Network Configuration”，显示的是 PMC600 主机的网络配置界面，在“网络设备”栏里可以看到允许添加的从站类型，包括雷赛总线伺服、雷赛总线步进、雷赛扩展模块等，如图 3.13 所示。

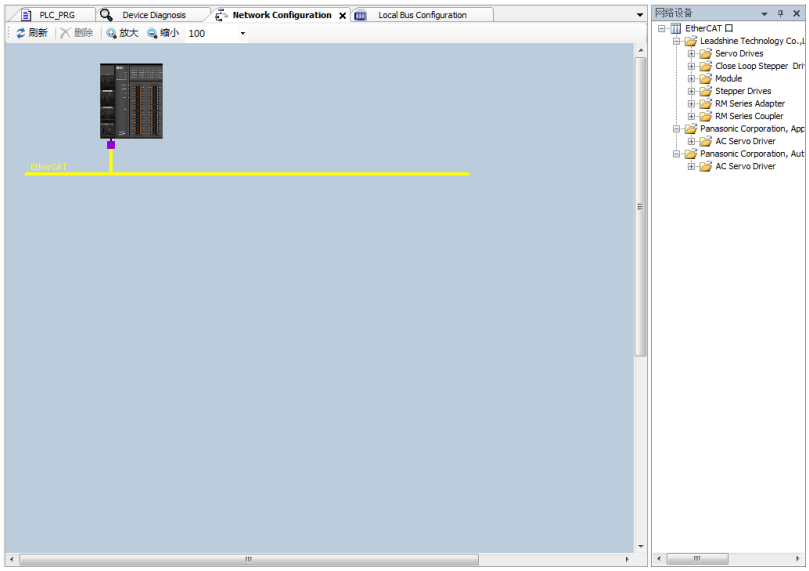

图 3.13　网络配置界面

鼠标左键单击 PMC600 的主机图片，可以看到 PMC600 主机当前支持的网络配置类型，

包括 RS232 配置、RS485 配置、Ethernet 配置、EtherNet/IP 配置、EtherCAT 配置。

- RS232：包括 Modbus 主站、Modbus 从站、自由协议；
- RS485：包括 Modbus 主站、Modbus 从站、自由协议；
- Ethernet：包括 Ethernet 主站、Modbus TCP 从站；
- EtherNet/IP：包括扫描器和适配器；
- EtherCAT：EtherCAT 主站，默认包含。

如图 3.14 所示。

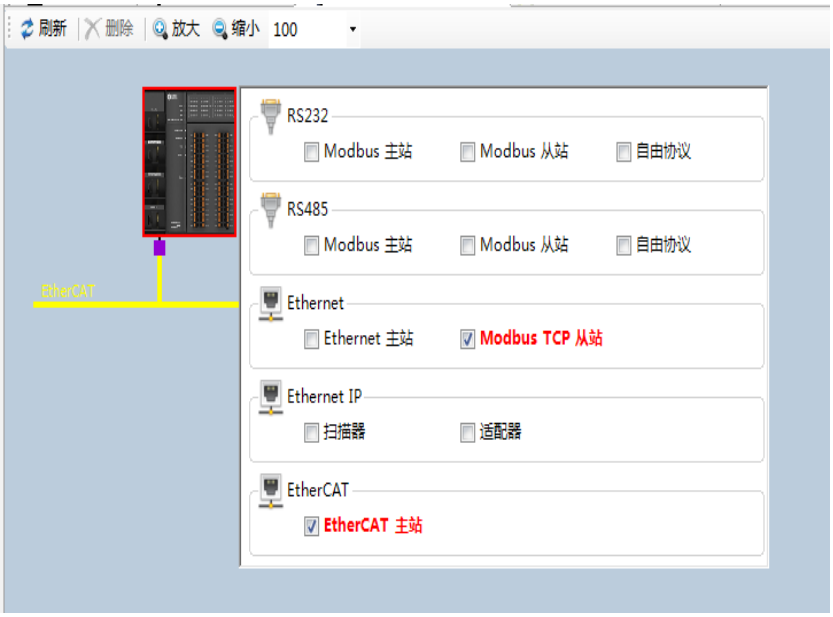

图 3.14　网络配置

PMC600 是 EtherCAT 总线型控制系统，在网络配置栏下会默认生成一个“EtherCAT Config”配置，双击“EtherCAT Config”，可以看到 PMC600 的 EtherCAT 总线网络配置，如图 3.15 所示。

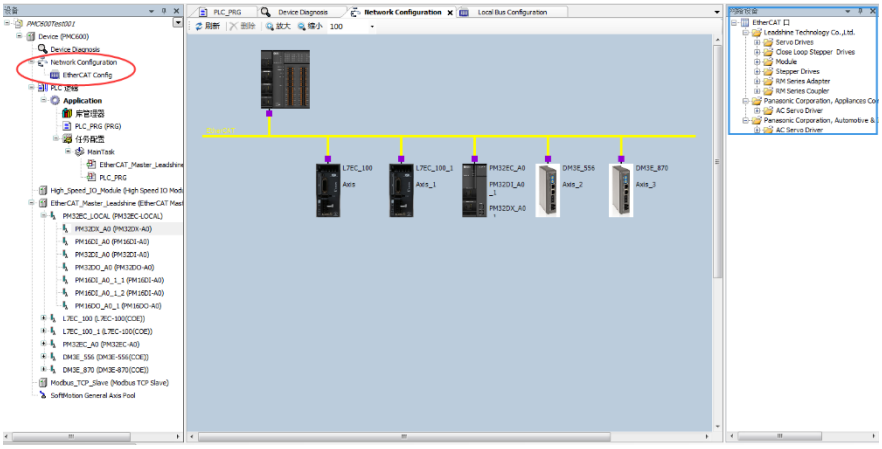

图 3.15　PMC600 总线网络配置

PMC600 主站支持右侧扩展 EtherCAT 总线扩展模块，也支持通过 EtherCAT 总线接口扩展伺服、步进或者 PM 系列总线适配器以及 EtherCAT 总线扩展模块。如图 3.16 所示。

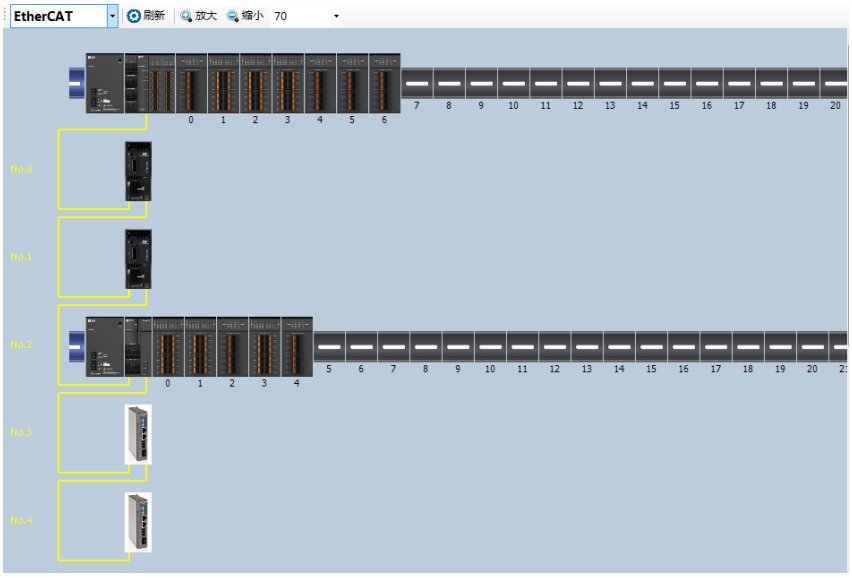

图 3.16　PMC600 总线网络扩展

3.2.7PLC 逻辑

PLC 逻辑指在硬件设备（PLC）上运行程序是所需要的对象集合。它的核心是应用（Application），是实现整个 PLC 控制的核心部分，在新建工程之后，PLC 逻辑中的 Application 缺省有“库管理器”、“PLC_PRG”、任务配置等选项，如图 3.17 所示。

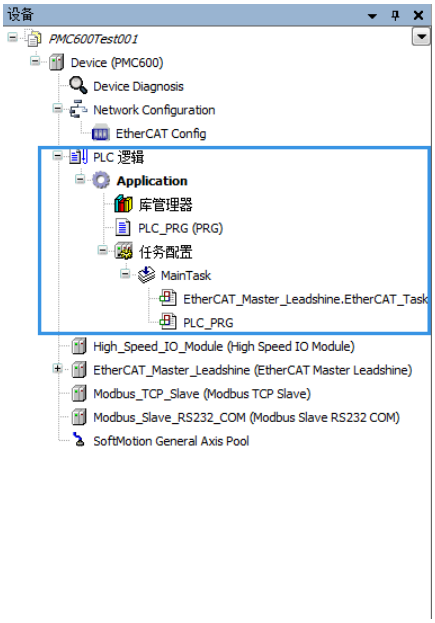

图 3.17　PLC 逻辑

右键单击“Application”，在弹出的快捷菜单中选择“添加对象”，或通过工具栏中的调按钮 实现对多种对象的添加操作，如图 3.18 所示。

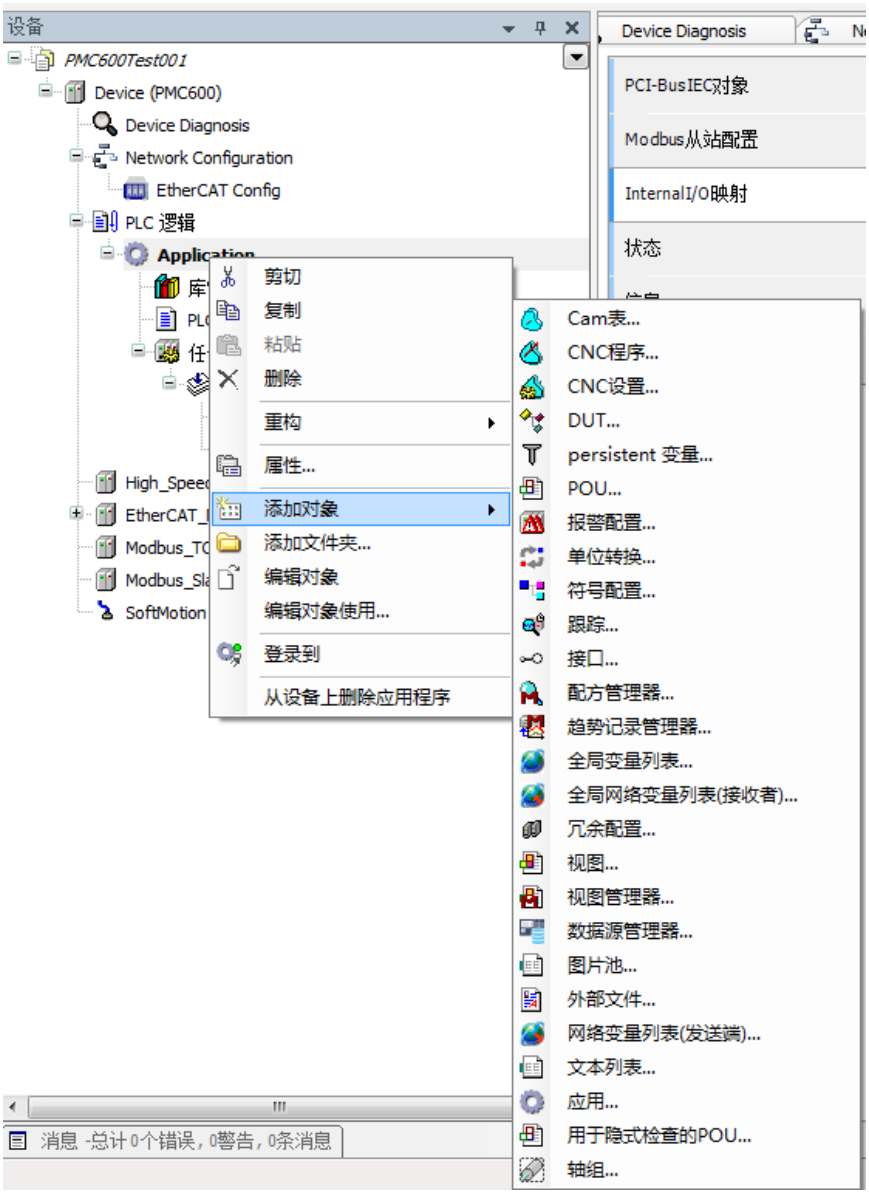

图 3.18　添加对象

常见的应用对象包括任务、程序组织单元、任务配置、全局变量、库管理器和采样追踪等。这些对象与硬件设备平台无关，用户可以在程序组织单元（POU）中管理它们，然后在设备窗口中将它们实例化，再分配到具体的设备中。

一个设备内可以有一个或者多个应用，如图 3.19 所示，该设备有 Application1 和 Application2 两个应用。

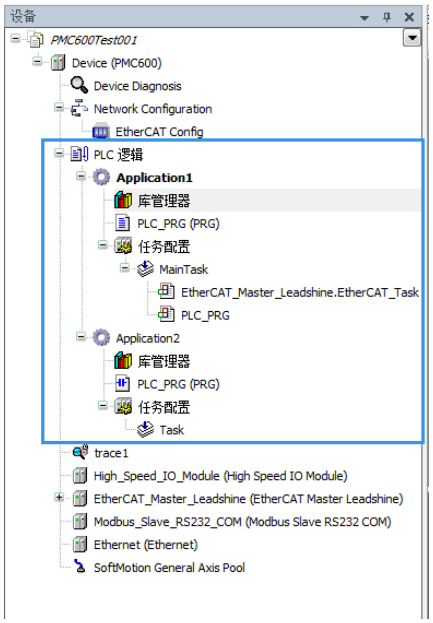

图 3.19　设备内的应用

3.2.7.1 库管理器

库文件用于存放程序中可以多次使用的程序组织单元（POU）。这些 POU 可以从已有的项目复制到库中，也可以由用户通过新建库项目自己定义库。

库函数除了是函数、功能块和程序的集合，其中还包含一些特殊定义的结构体、枚举类型等。从功能上划分，可以将库文件分为系统库文件、应用库文件、厂商自定义库文件及 IEC 动作库。默认的函数库文件是“.library”。

在“库管理器”中，显示与当前项目有关或者调用的所有库，如图 3.20 所示。库的 POU、数据类型和全局变量，都可以像普通定义的 POU、数据类一样。

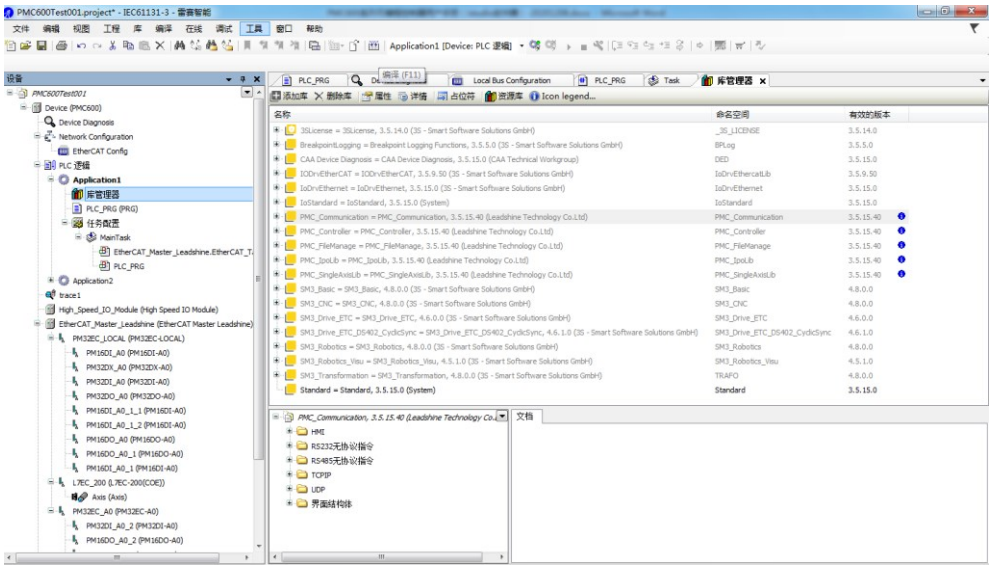

图 3.20　库管理器

在库文件中，可以进行“添加库”、“删除库”等操作，如图 3.21 所示。在 istudio 软件中，已经打包安装好雷赛 PMC 系列控制器库和 SMC 系列控制器库，封装了逻辑控制和运动控制相关功能，可以直接使用。

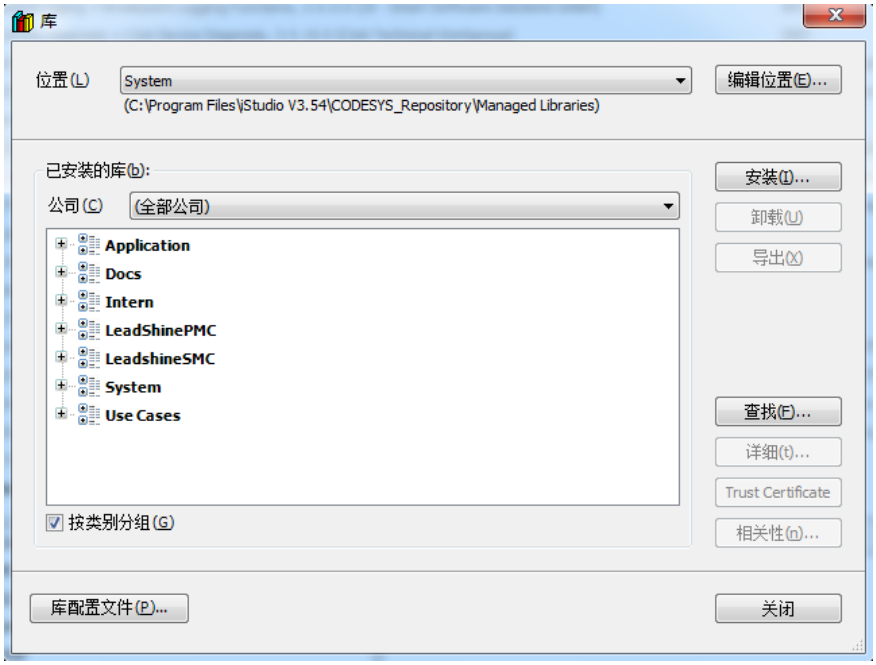

图 3.21　添加库和删除库

3.2.7.2 POU

新建 PMC600 工程之后，会在 Application 中自动新建一个 PLC_PRG(PRG)，即默认的 POU，POU 是程序组织单元，由声明区和代码区两部分组成，是用户程序的最小软件单元，它相当于传统编程系统中的块（Block），是全面理解新语言概念的基础。按功能划分，POU 可以分为函数（FUN）、功能块（FB）和程序（PRG）。

一个完整的 POU 由如下三大部分组成，结构如图 3.22 所示：

（1）POU 类型及命名

（2）变量声明部分

（3）代码指令部分（POU 主体）

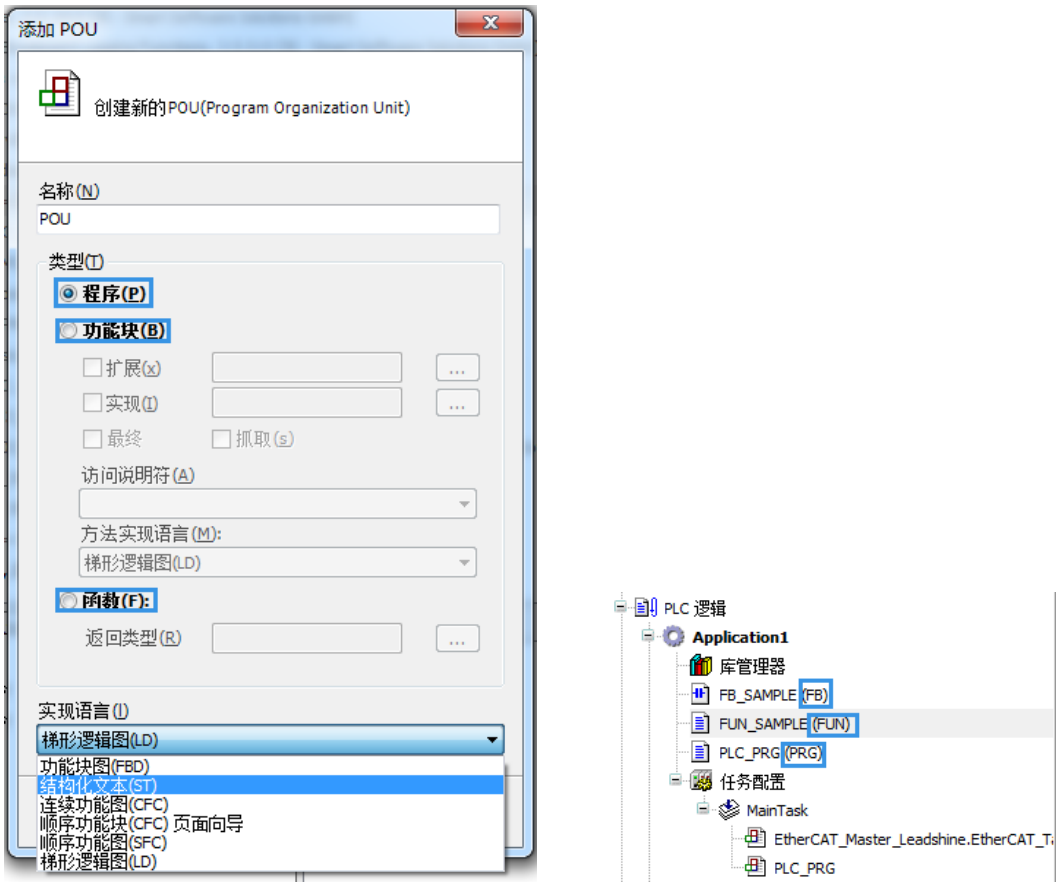

图 3.22　POU 的结构组成

3.2.7.3 任务

3.2.7.3.1　任务概述

一个程序可以用不同的编程语言来写。一个典型的程序通常由许多功能块组成，各功能块之间可以互相交互数据。在一个程序中，不同部分的执行通过“任务”来控制。“任务”配置后，可以使一系列程序或功能块周期性或由一个特定的事件触发开始执行程序。

在任务配置中，用名称、优先级和任务的启动类型来定义它。启动类型可以通过时间（周期、随机）或通过内部或外部的触发任务时间来定义，如用一个布尔型全局变量的上升沿或系统中的某一特定事件。对每个任务，可以设定一串由任务启动的程序，如果在当前周期内执行次任务，那么这些程序会在一个周期的长度内被处理。优先权和条件的结合将决定任务执行的时序。任务具体的设置界面如图 3.23 图所示。

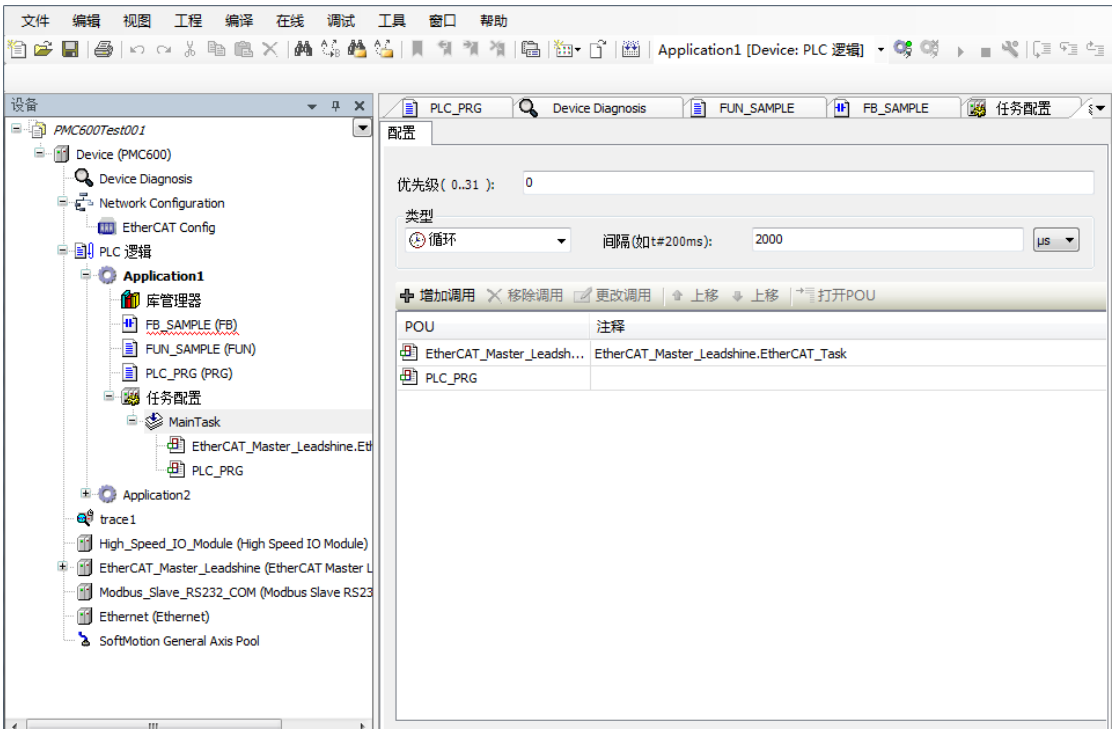

图 3.23　任务设置

任务有相应属性，程序设计人员在配置时需要遵循以下规则：

- 循环任务的最大数为 100
- 惯性滑行任务的最大数为 100
- 事件触发任务的最大数为 100
- 主程序“PLC_PRG”可能会在任何情况下作为一个惯性滑行程序执行，而不用手动插入任务配置中。
- 处理和调用程序是根据任务编辑器内自上而下的顺序所执行的

3.2.7.3.2　PLC 执行程序过程

图 3.24 详细地描述了在 PLC 内部执行程序的完整流程，主要包括输入采样、程序执行和输出刷新。

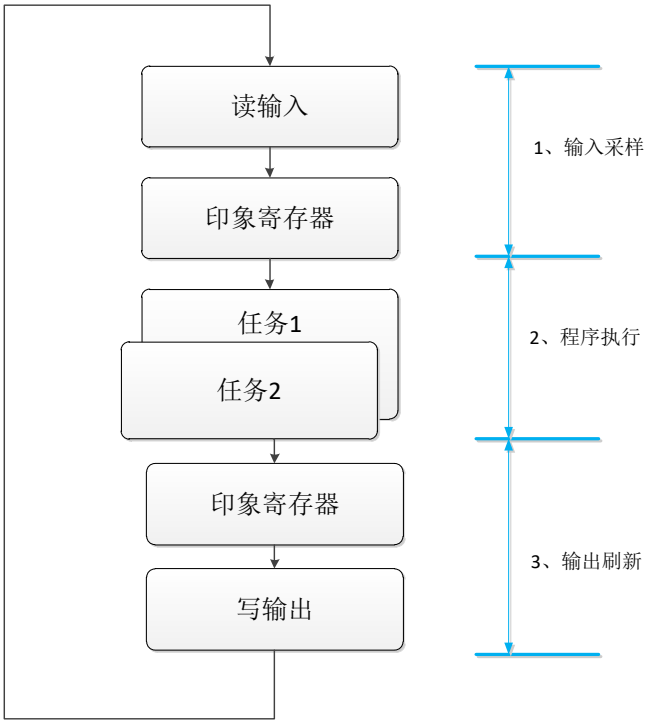

图 3.24　PLC 程序执行过程

（1）输入采样：每次扫描周期开始时，PLC 检测输入设备（开关、按钮）的状态，将状态写入输入印象寄存区内。在程序执行阶段，运行系统从输入印象区内读取数据并进行程序运算。需要特别注意的是，输入的刷新只发生在扫描开始阶段，在扫描过程中，即使输出状态改变，输入状态也不会发生变化。

（2）程序执行：在扫描周期的程序执行阶段，PLC 从输入映像区或输出映像区内读取状态和数据，并依照指令进行逻辑和算数运算，运算的结果保存在输出映像区相应的单元中。在这一阶段，只有输入映像寄存器的内容保持不变，其他映像寄存器的内容会随着程序的执行而变化。

（3）输出刷新：输出刷新也叫写输出阶段，PLC 将输出印象区的装填和数据传送到输出点上，完成输出动作。

PLC 在一个扫描周期内除了完成上述 3 个阶段的任务，还要完成内部诊断、通信、公共处理以及输入\输出服务等辅助任务。

PLC 重复执行上述过程，每重复一次的时间就是一个工作周期（扫描周期）。由扫描方式可知，为了迅速响应输入/输出数据的变化，完成控制任务，扫描时间较短，扫描周期一般都控制在毫秒（ms）级。

3.2.7.3.3　任务的执行类型

针对每个独立的任务，可以对其执行的类型进行编辑，包括固定周期循环、事件触发、惯性滑行和状态 4 种类型，如图 3.25 所示。

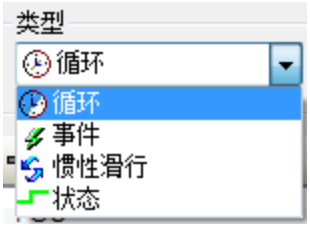

图 3.25　任务的执行类型

3.2.7.3.4　PMC600 的 EtherCAT 任务

PMC600 项目中会自动生成一个 EtherCAT 任务，如图 3.26 所示，并且占据最高优先级，与总线有关的操作，特别是轴控制，需要在这个任务下运行。

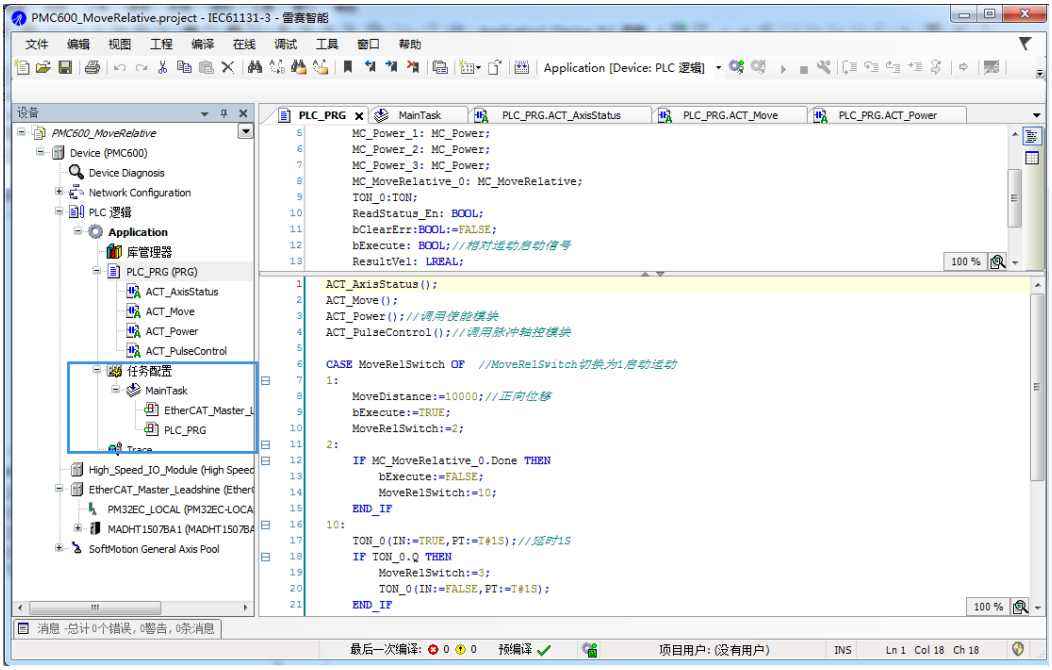

图 3.26　EtherCAT 任务

3.2.7.4 高速 IO 配置

高速 IO 配置是为了方便设置 PMC600 主机上的高速 IO 口，分配各个复用 IO 口的资源。PMC600 产品的共有 64 点 IO，区分为 32 入 32 出。

其中 IN0 – IN15 作为通用高速输入口，可以支持复用功能。复用功能包括：通用输入、单相计数器、AB 相计数器以及轴原点限位信号。IN16 – IN31 作为低速输入口，仅做普通输入口使用，不复用。

输出口 OUT0-OUT15 作为高速输出口，还可以支持单端脉冲输出，包括脉冲方向输出和 CW/CCW 输出。OUT16 – OUT31 仅做普通输出口使用，不复用。

新建工程后，在左侧模型树下会自动生成“High Speed IO Module”，双击此栏，可以

看到高速 IO 模块对应的配置界面，如下图 3.27 所示。

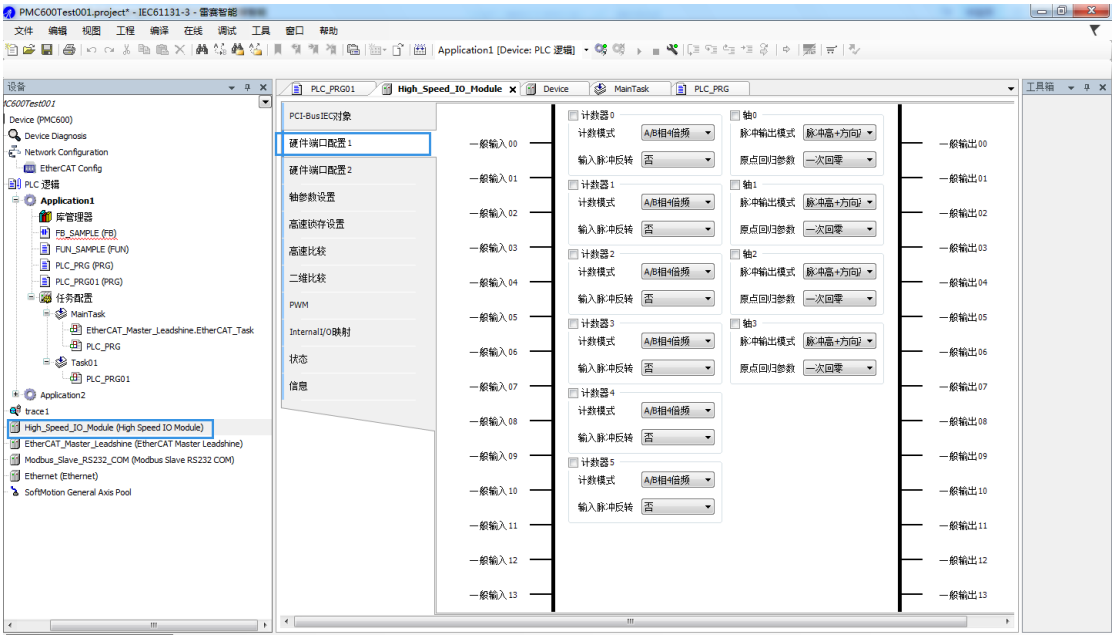

图 3.27　高速 IO 模块配置界面

详细配置方法请参考本文 4.2.2 PMC 主机 IO 模块配置部分。

3.2.7.5 虚拟轴配置

在工程的左侧目录树下，可以看到“SoftMotion General Axis Pool”,即虚拟轴配置信息，可以在此栏中添加虚拟轴相关操作，选中“SoftMotion General Axis Pool”，右键选择“添加设备”即可以添加虚拟轴，如图 3.28 所示。

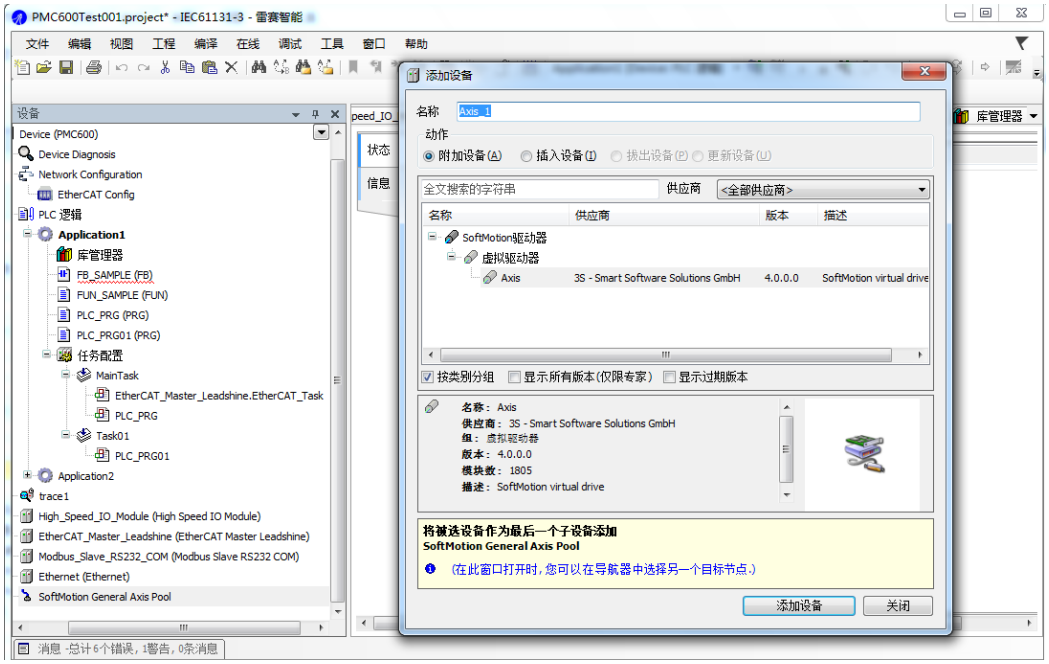

图 3.28　添加虚拟轴

当在 IO 配置界面里，将高速输出口选择为脉冲输出功能时，会将轴添加到虚拟轴列表。如图 3.29 所示所示。

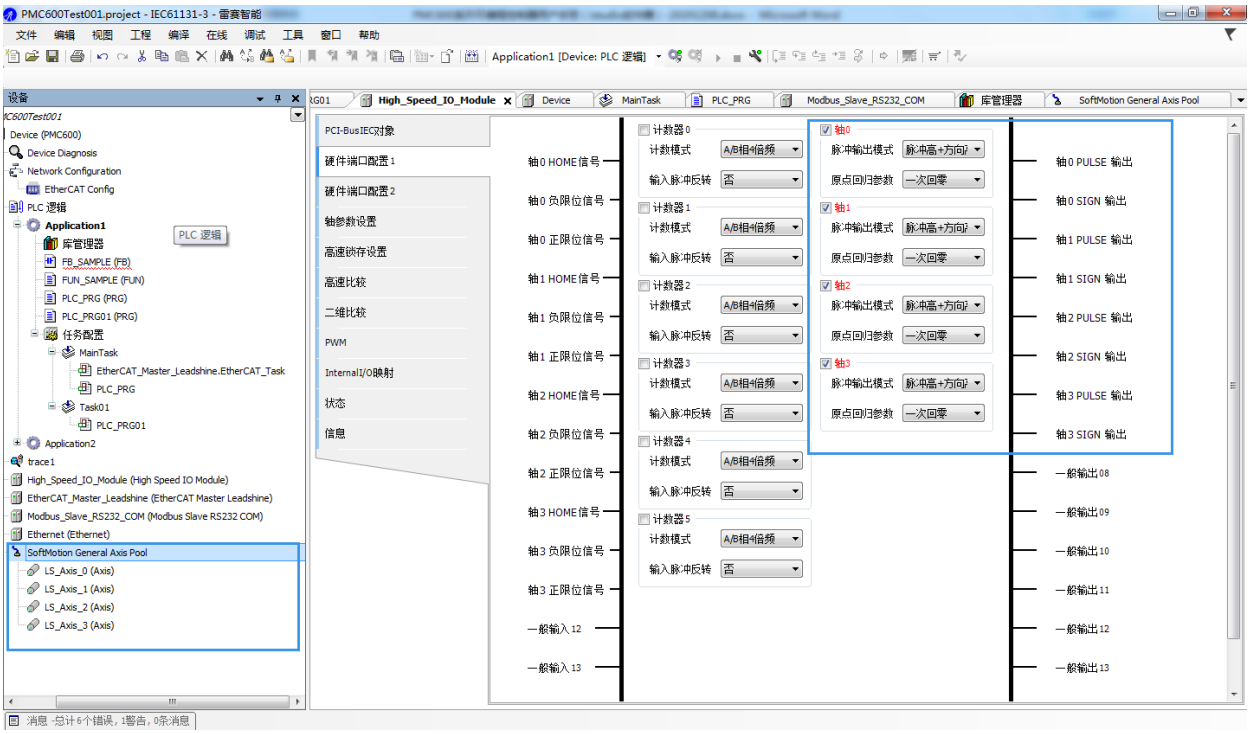

图 3.29　虚拟轴列表

第4章　参数配置

4.1 PMC600 主机配置

主机的配置主要是各接口协议配置和高速 IO 功能配置、CPU 的系统参数如 IP 地址、RTC 等。

PMC600 系列支持 EtherCAT 总线、EtherNet/IP、Modbus RTU、Modubs TCP、自由协议等，另外还提供高速 IO 功能，可以将高速 IO 口配置成通用输入输出或者有特殊功能的 IO 口，如高速计数口、脉冲输出口等。

4.1.1 主机系统设置

系统设置主要用于配置 CPU 故障停机、掉电位置保存、网络地址和系统时间等，如图 4.1 所示。

图 4.1　系统配置

错误时的运行模式：

（1）组态错误时停机：出现组态不一致时 CPU 是否停止运行，例如 CPU 后挂接的组态 IO 和实际的 IO 不匹配是否停机。

（2）系统错误时停机：出现系统错误时 CPU 是否停止运行，如中断错误、堆栈溢出等。

（3）FLASH 错误时停机：出现 FLASH 错误时 CPU 是否停止运行，暂不支持。

（4）SD 卡错误时停机：出现 SD 卡错误时 CPU 是否停止运行，如 SD 卡内存满，SD 卡丢失等。

掉电保存：设置掉电保存位置，如本地存储器或 SD 扩展卡。

网络设置包括以下方面：

（1）网口：PMC600 系列产品使用的 Ethernet 通讯网口名。PMC600 支持两个 Ethernet 口，两个 Ethernert 口使用不同的网络名称，可以对这两个网口设置不同的网络信息。

（2）使用下面的 IP：PMC 的 IP 地址既可以通过手动修改，也可以自动获取，此配置用于手动修改 PLC 的网络信息。

（3）自动获取 IP：PMC 的 IP 地址由路由器或者交换机自动分配。

（4）IP 地址：PMC 的 IP 地址。

（5）子网掩码：PMC 的子网掩码。

（6）网关：设置 PMC 的网关。

（7）读取：读取 PLC 的 IP 地址和子网掩码，显示在 IP 地址和子网掩码编辑框中。

（8）写入：将编辑框中的 IP 地址和子网掩码写入 PLC。如果在登录状态并且使用网络连接，系统会退出登录，此时用户需要重连设备。

系统时间配置包括以下方面：

（1）PLC 时间：显示读取的当前 PLC 时间。

（2）读取：读取 PLC 时间。

（3）写入：写入 PLC 当前设置的日期和时间，当前设置的日期和时间在左侧的日期和时间编辑框内。

（4）同步到本地日期/时间：写入当前电脑对应的时间到 PLC。

升级：点击“工具”选项，可以选择进行固件升级，包括固件文件、数据文件、配置文件、Application 等。可以提供下载到控制器、上传到本地、删除功能，如图 4.2 所示。。

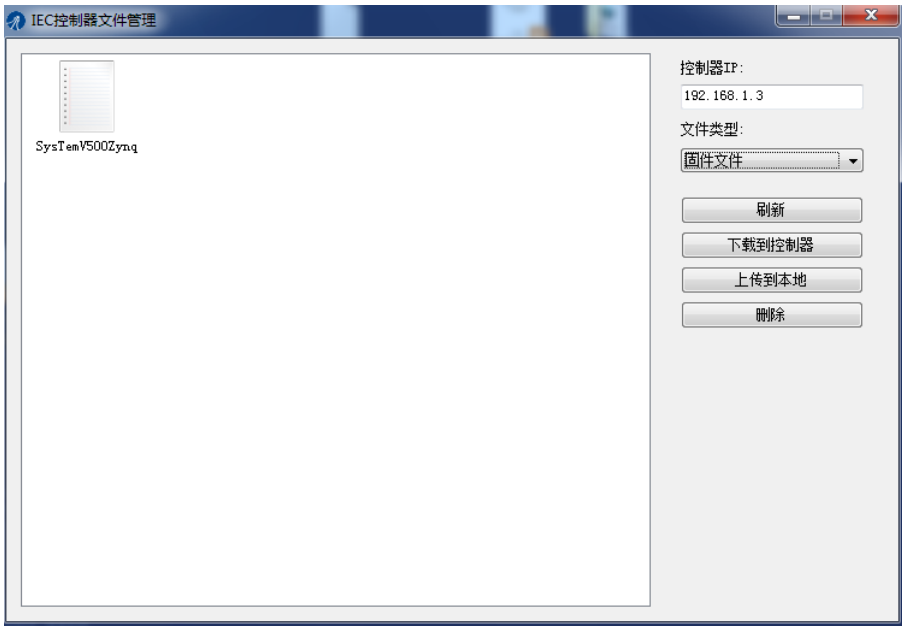

图 4.2　升级界面

控制器 IP：目标控制器的 IP 地址；

文件类型：包括固件文件、数据文件、配置文件、Application；

固件文件：PMC 控制器的固件文件；

Application：应用程序；

刷新：刷新当前选择文件类型的状态；

下载到控制器：将当前选择的文件下载到控制器；

上传到本地：将当前选择的文件类型上传到 PC；

删除：删除当前选择的文件。

PMC 控制器信息：PMC 的信息包括名称、厂商、组、类型、ID、版本、模型号以及描述，如图 4.3 所示。

图 4.3　PMC 控制器信息

4.1.2 PMC600 主机 IO 配置

如图 4.4 所示，在左侧工程目录树下，双击“High_Speed_IO_Module”,可以看到弹出的高速 IO 模块界面，该界面中主要包括硬件端口配置（PMC 系列支持）、轴参数配置（PMC610 支持）、高速锁存配置（PMC610 支持）、高速比较（PMC610 支持）、二维比较（PMC610 支持）、PWM（PMC610 支持）、IO 映射（PMC 系列支持）等。

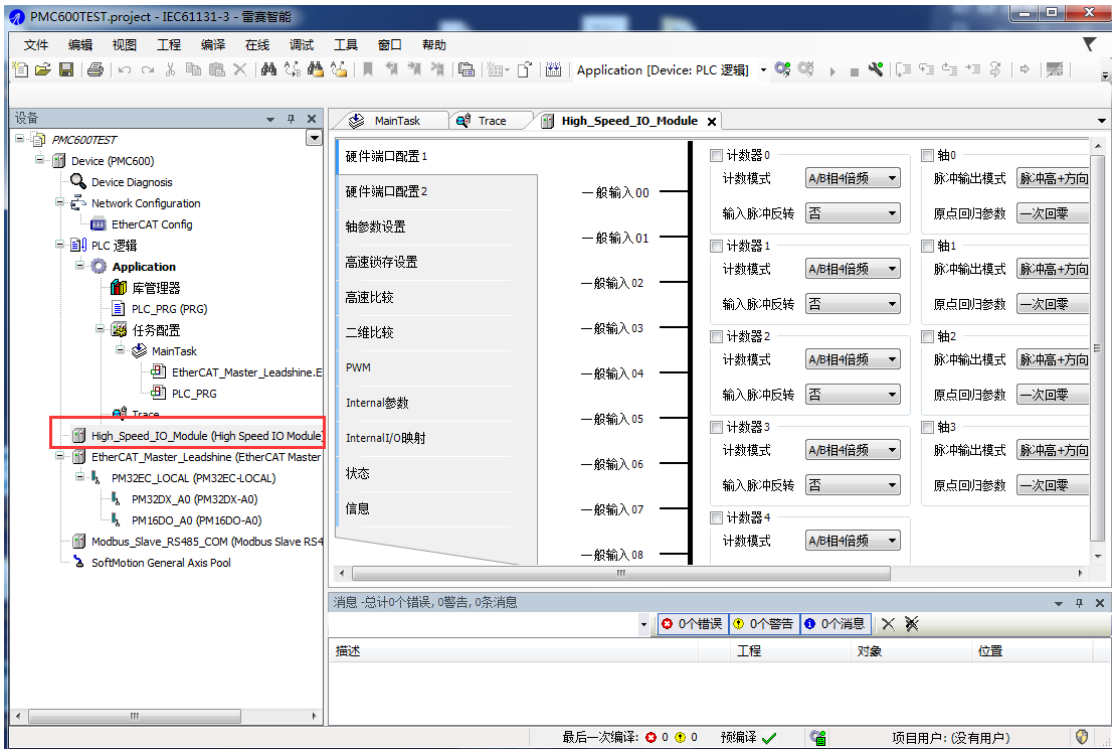

图 4. 4　PMC 主机 IO 配置

（1）硬件 IO 端口配置 1

PMC600 系列控制器的 IO 支持多种复用功能，包括通用输入、单相计数器、AB 相计数器以及轴原点限位信号。IN16 – IN31 作为低速输入口，仅做普通输入口使用，不复用。

输出口 OUT0-OUT15 作为高速输出口，还可以支持单端脉冲输出，包括脉冲方向输出和 CW/CCW 输出。OUT16 – OUT31 仅做普通输出口使用，不复用。

具体复用功能详见《PMC600 中型 PLC 用户手册 1-硬件篇》。

计数器和轴功能在勾选之后，表示会使能计数器功能和轴功能，此时计数器功能会占用一般输入口，轴功能会占用一般输出口。所以当需要使用一般输入输出口时，不能勾选“计数器 0-计数器 5”和“轴 0-轴 4”的选项。

一般输入支持 IN0 到 IN15 共 16 个输入口，一般输出支持 OUT0 到 OUT15 共 16 个输出口。选择一般输入输出后的界面如图 4.5 所示，计数器 n、轴 n 之前的方框内无勾。

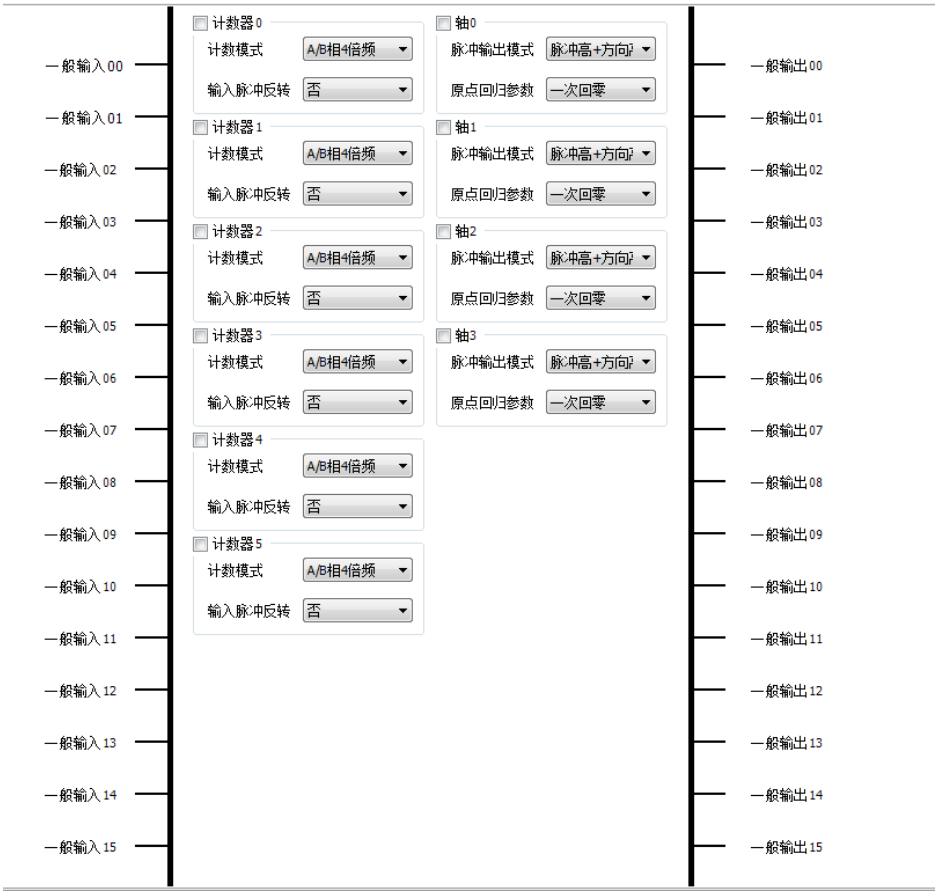

图 4.5　输入输出口配置界面

（2）计数器

如果需要使用高速计数器，则需要勾选对应的计数器编号，选中的计数器会显示红色，如图 4.6 所示。每个计数器的计数模式包括“A/B 相 4 倍频”、“脉冲方向”、“单相计数”三种模式：

（1）A/B 相 4 倍频：会占用两个输入口，支持输入脉冲翻转选项。

（2）脉冲方向：会占用两个输入口。

（3）单相计数：会占用一个输入口。

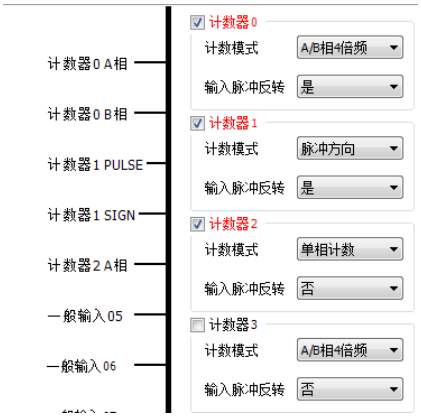

图 4.6　计数模式

（3）轴

如果需要使用轴相关功能，则需要勾选对应的轴号，选中的轴会显示红色，每个选中的轴会自动配置对应的输入口作为原点和正负限位，也会自动匹配对应的输出口作为脉冲输出口（仅支持单端输出）。

图 4.7　轴配置

如图 4.7 所示，勾选“轴 0”，轴 0 会显示红色，输入口 IN0 会被配置为轴 0 HOME 信号，IN1 会被配置为轴 0 负限位信号，IN2 会被配置为轴 0 正限位信号，输出口 OUT0 会被配置为轴 0 PULSE 输出，输出口 OUT1 会被配置为轴 0 SIGN 输出。勾选“轴 1”，轴 1 会显示红色，输入口 IN3 会被配置为轴 1 HOME 信号，IN2 会被配置为轴 1 负限位信号，IN5 会被配置为轴 1 正限位信号，输出口 OUT2 会被配置为轴 1 脉冲 PULSE 输出，输出口 OUT2 会被配置为轴 1 方向 SIGN 输出。以此类推。

请注意，当勾选轴功能时，如果已经勾选了计数器功能，可能会导致端口配置出现冲突，如图 4.8 所示，此时请注意合理分配计数器和轴的功能。

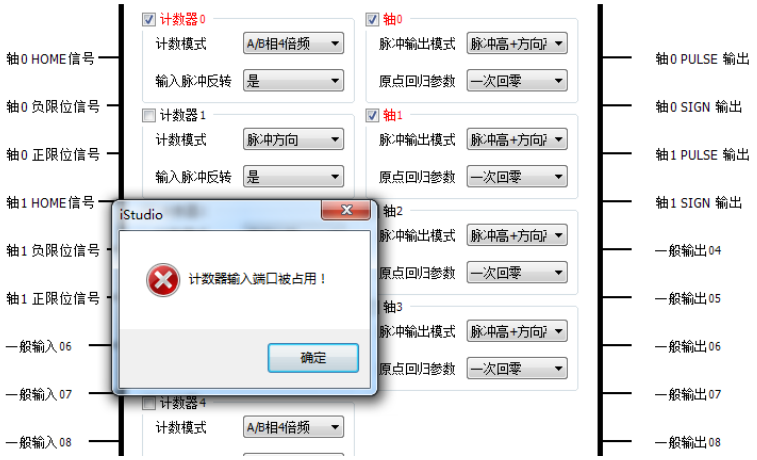

图 4.8　计数器和轴功能冲突

轴功能选择后，还可以配置脉冲输出模式参数和回零参数。

（4）脉冲输出模式：

脉冲+方向：信号分高电平有效和低电平有效，故有：脉冲高+方向高、脉冲高+方向低、脉冲低+方向高、脉冲低+方向低等组合方式。

双脉冲：支持双脉冲高和双脉冲低两种模式。

AB 相：支持 AB 相输出模式。

（5）回原点模式：

PMC 主机上的回零模式支持多种方式，需要配合轴参数中的回零方向来完整定义具体的回零的动作，包括一次回零，一次回零加反找、二次回零、原点锁存、二次回零加锁存，回零的方向包括负方向和正方向。具体的回零动作含义如图 4.9~4.18 所示。

一次回零+负方向回零：

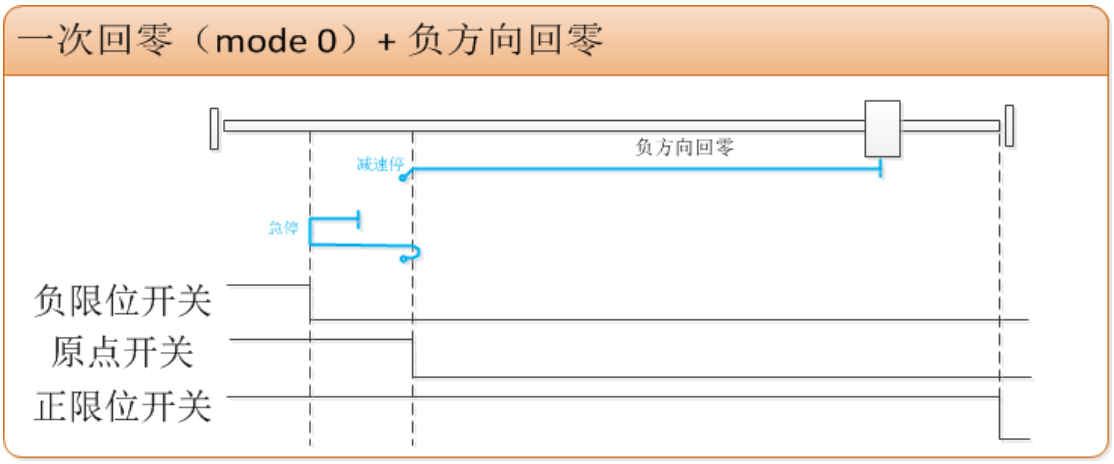

图 4.9　一次回零+负方向回零

一次回零+正方向回零：

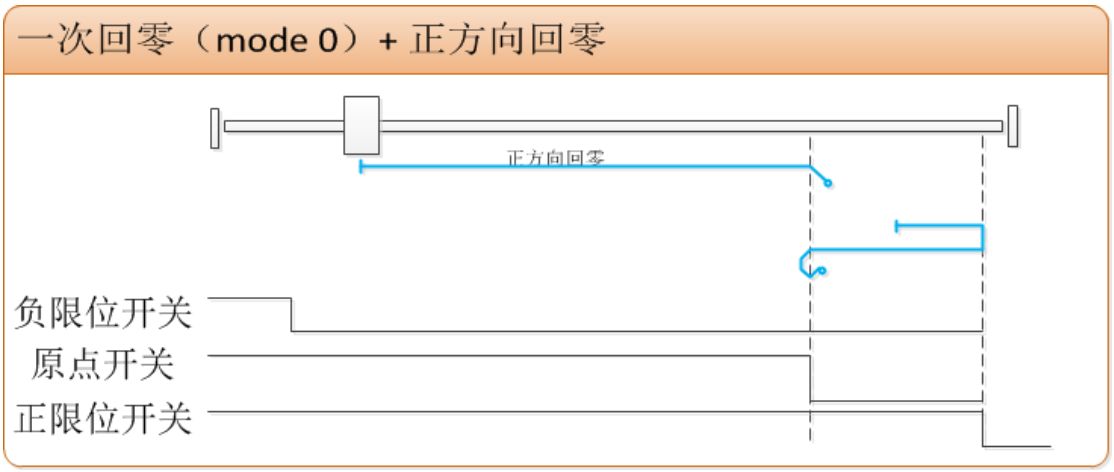

图 4.10　一次回零+正方向回零

一次回零反找+负方向回零：

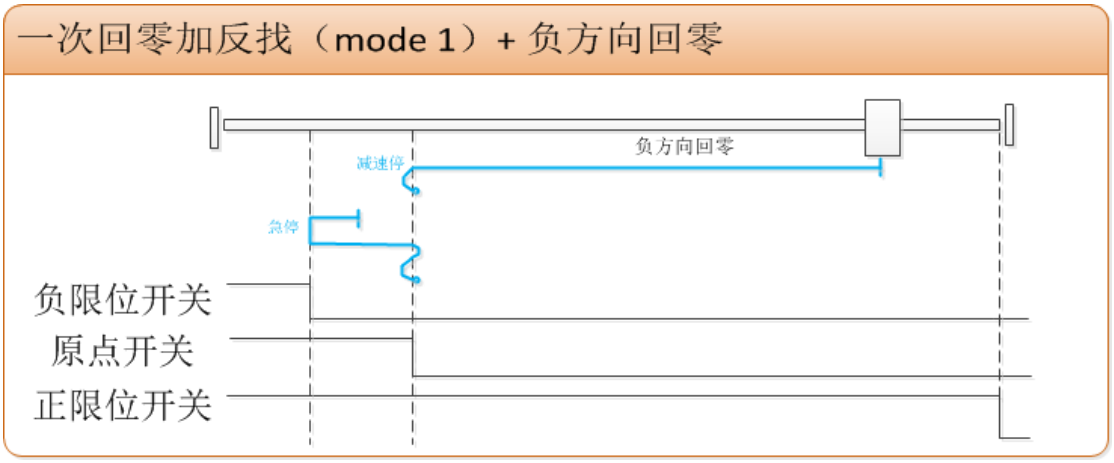

图 4.11　一次回零反找+负方向回零

一次回零反找+正方向回零：

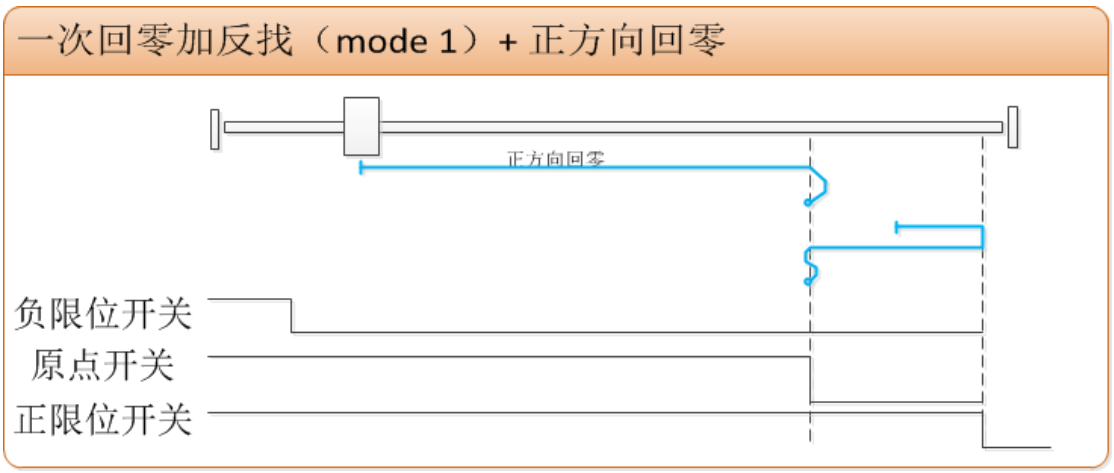

图 4.12　一次回零反找+正方向回零

二次回零+负方向回零：

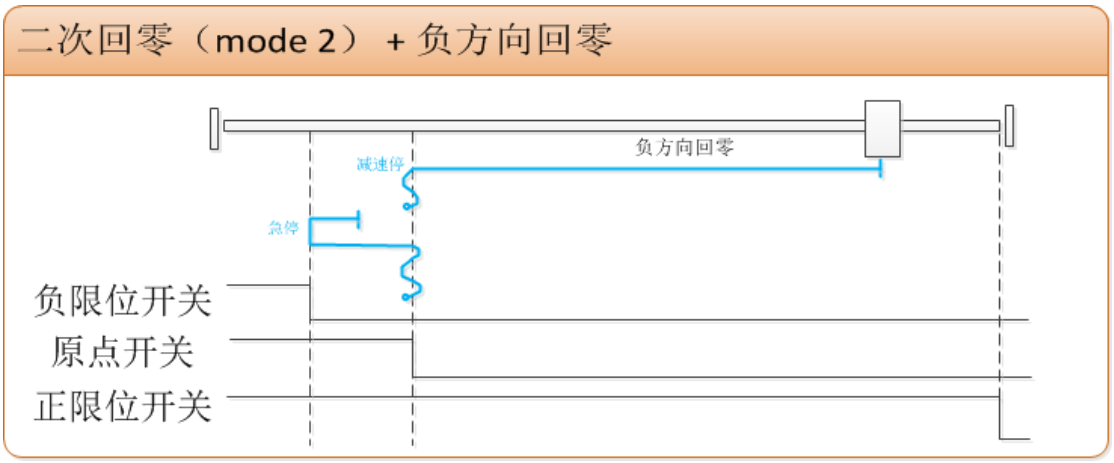

图 4.13　二次回零+负方向回零

二次回零+正方向回零：

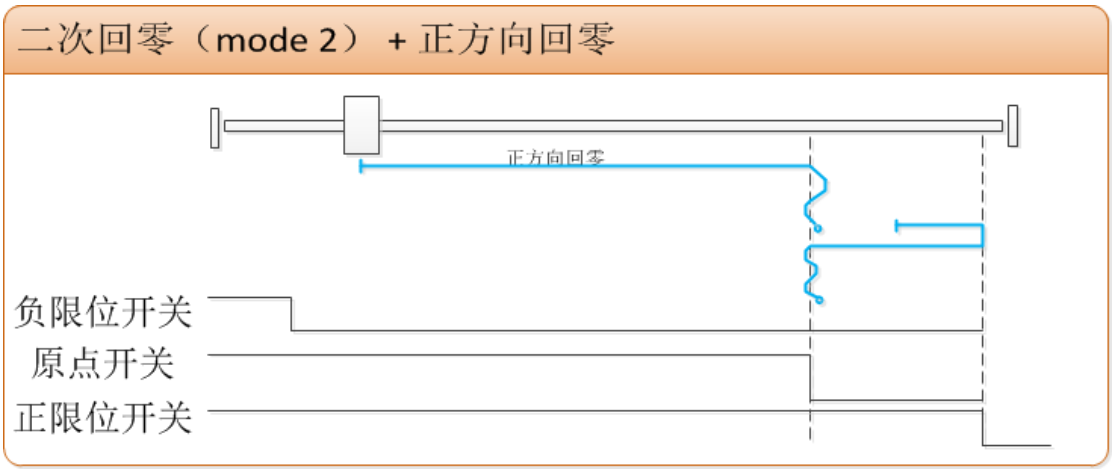

图 4.14　二次回零+正方向回零

一次回零加原点锁存+负方向回零：

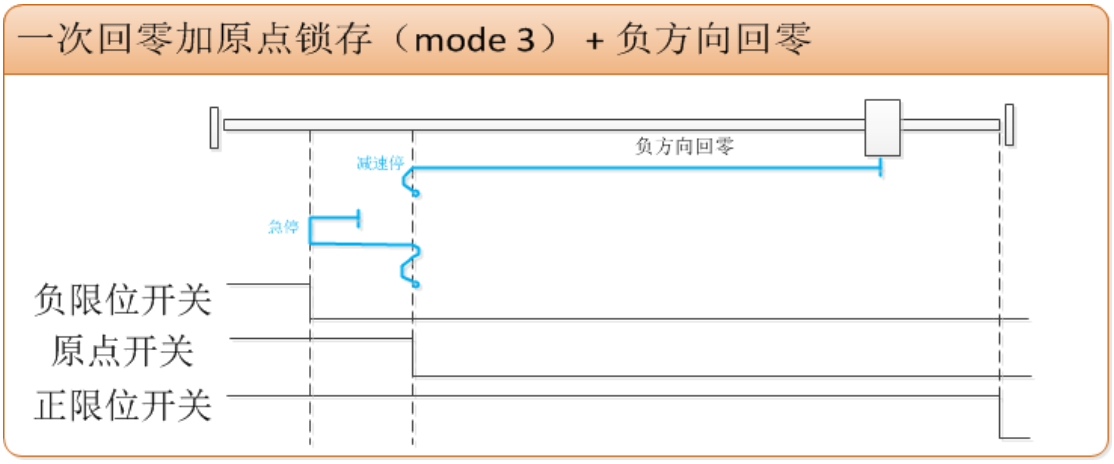

图 4.15　一次回零加原点锁存+负方向回零

一次回零加原点锁存+正方向回零：

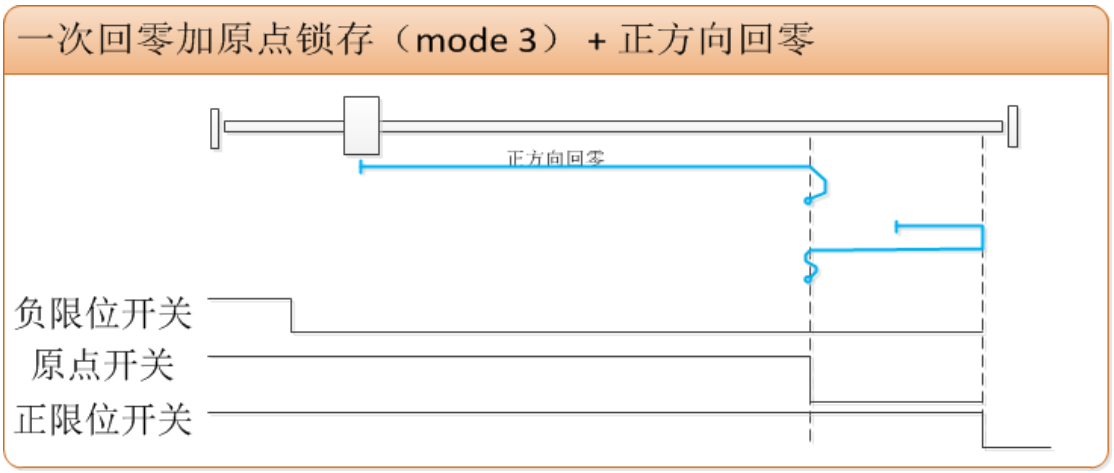

图 4.16　一次回零加原点锁存+正方向回零

二次回零加锁存+负方向回零：

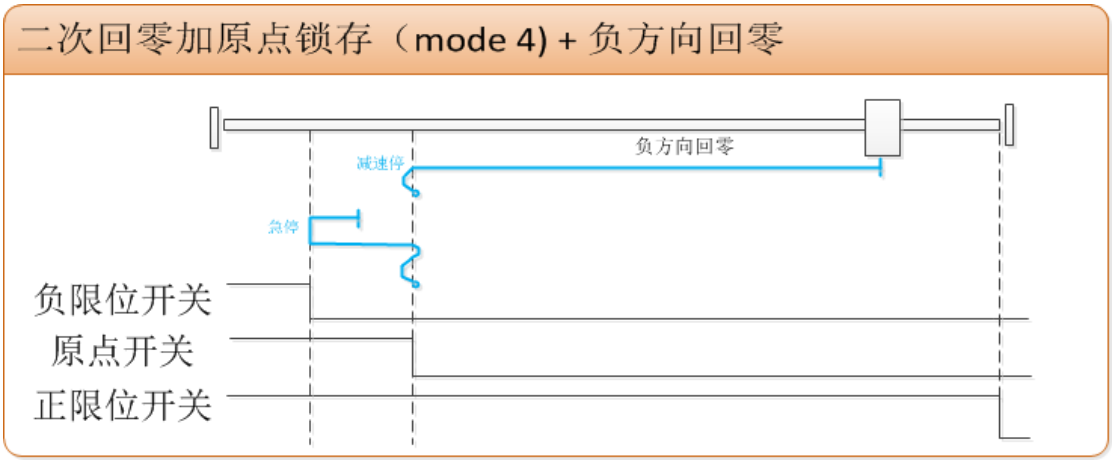

图 4.17　二次回零加锁存+负方向回零

二次回零加锁存+正方向回零：

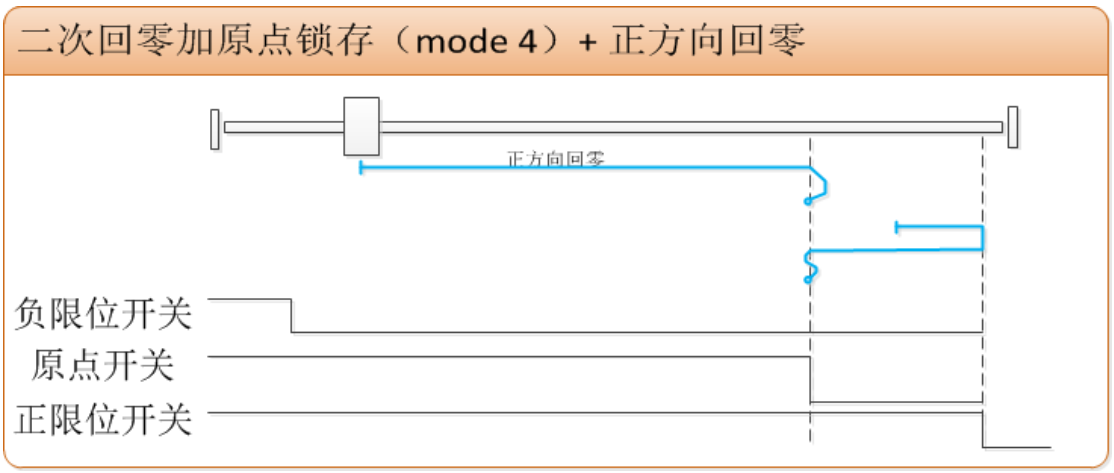

图 4.18　二次回零加锁存+正方向回零

硬件端口配置 2：

PMC610 控制器支持 32 路输入和 32 路输出，在“端口配置 1”中分配了 IN0 – IN15 以及 OUT0 – OUT15 的引脚定义，在“端口配置 2”中分配了 IN16 – IN31 以及 OUT16 – OUT31 的引脚定义，如图 4.19 所示。

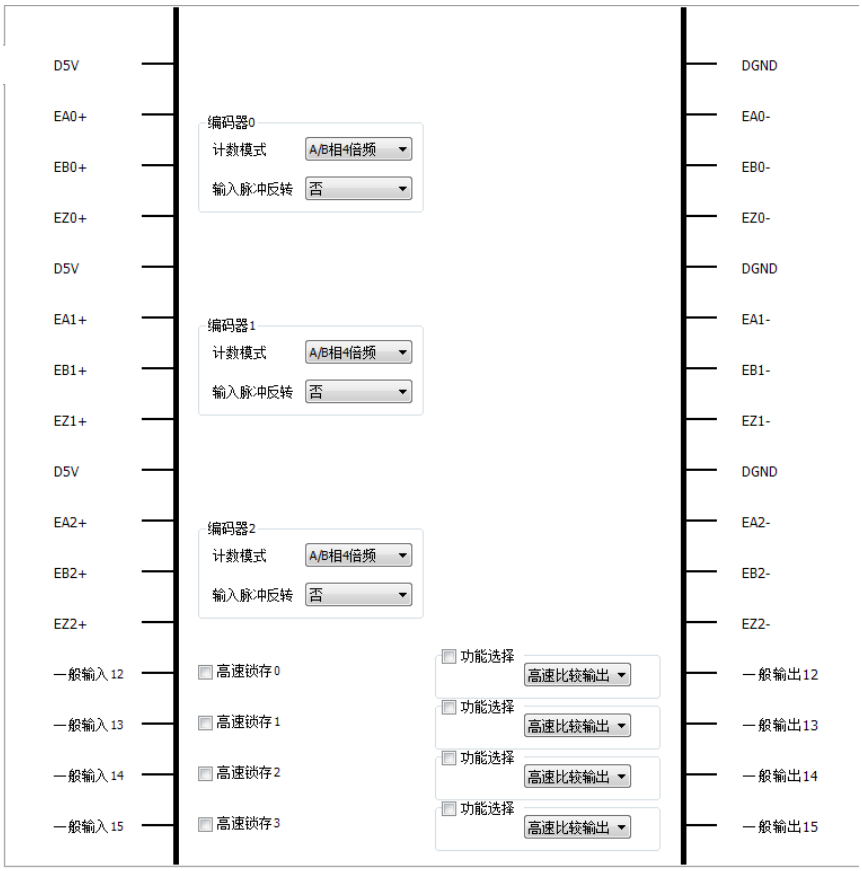

图 4.19　PMC610 的端口配置 2

轴参数设置：

当使能了轴控制功能之后，轴参数中对应的界面会变成可编辑状态，当去掉使能轴功能之后，轴参数中对应的界面会变成不可编辑状态。如勾选了“轴 0”和“轴 1”，轴 0 和轴 1 对应的界面中的参数设置变成可编辑状态，轴 2 和轴 3 对应的界面中的参数设置仍然是不可编辑状态。

以下分别介绍轴参数配置界面中的各项参数的含义：

轴名称：对应的轴的名称，在程序中可以直接使用，作为参数传递到功能块中。

传动比分子：

传动比分母：

回零一般参数：

（1）回零模式，具体模式参考图 4.9~4.18；

（2）回零方向：正方向或负方向；

（3）有效电平：可选择低有效下降沿触发和高有效上升沿触发；

（4）回零位置偏移：回零完成后的位置偏移值，正值表示正向，负值表示负向；

（5）回零完成方式：

（6）原点回归速度：高速找原点开关的速度；

（7）爬行速度：低速离开原点开关的速度；

（8）加速度：加速度；

（9）减速度：减速度。

回零特殊参数：

（1）HOME 锁存源：原点锁存源，包括指令位置和编码器反馈位置；

（2）HOME 触发边沿：原点触发方式，下降沿触发锁存和上升沿触发锁存；

（3）停止模式：急停模式或者减速停模式；

（4）停止周期数：减速停模式下，停下来花费的周期数。

高速锁存设置：

高速锁存是指接入到控制器的高速输入信号有效后，控制器器自动锁定当前指定运动轴的位置并保存，可以实现高精度的定位控制。高速锁存支持单次锁存模式和连续锁存模式，单次锁存是指开启了锁存功能之后锁存一次，连续锁存是指可以多次进行位置锁存。

PMC610 支持 4 个高速锁存器，需要在硬件配置端口界面进行配置，当使能了对应的高速锁存器之后，此界面才处于可配置状态，否侧此界面里的参数无法编辑。如图 4.20 所示。

图 4.20　高速锁存界面

锁存模式：支持单次锁存和连续锁存；

锁存轴号：支持锁存 PMC600 的本地 4 个轴，轴 0 到轴 3；

锁存源：支持指令位置和反馈位置两种方式；

触发边沿：下降沿触发、上升沿触发和双边沿触发；

滤波时间：设置输入信号的滤波时间，单位为毫秒，可以过滤掉干扰信号。

高速比较：

高速比较是指在 PMC 底层实时的将当前位置和目标位置进行比较，当满足触发条件时，对指定的输出口进行指定电平的输出。常用于飞拍等高速运行场合。

在硬件配置选择将最后 4 个高速输出口设置为高速输出口，PMC 软件会将高速比较设置界面激活，如图 4.21 所示。

图 4.21　高速比较设置界面

比较模式：支持禁止、小于模式、等于模式、大于模式、线性模式、队列模式，各个模式的具体含义如下所示。

（1）禁止：禁止使用此功能；

（2）小于：指令位置或者编码器位置小于目标位置时触发输出口输出；

（3）大于：指令位置或者编码器位置大于目标位置时触发输出口输出；

（4）等于：指令位置或者编码器位置等于目标位置时触发输出口输出；

（5）线性：目标位置是一系列递增或递减的位置点，从起始位置开始按照设置的比较点数递增或者递减，当当前位置达到目标位置时触发输出口输出；

（6）队列：目标位置是一系列位置点，但该位置点是应用程序设置的，可以任意设置，当当前位置到达目标位置时出发赎回粗口输出。

比较轴号：需要比较位置的轴号。

比较源：选择是使用指令位置还是编码器位置作为比较的参考源。

输出电平：选择位置比较触发之后指定输出口的电平状态，可选高电平或低电。

注意：

每一个比较器对应一个指定的输出口，不能更改，高速比较器 0 绑定的输出口是

OUT12，高速比较器 1 绑定的输出口是 OUT13，高速比较器 2 对应的输出口是 OUT14，高速比较器 3 对应的输出口是 OUT15。

二维比较：

二维高速比较是在一维高速比较的基础上增加了一个轴进行比较，当进行两轴位置比较时，PMC600 根据目标位置设置一个误差带，由两轴组成的平面矩形误差带，如果当前的实际位置进入到误差带，则进行比较，当满足条件时，会进行指定输出口的输出。二维比较设置界面如图 4.22 所示。

图 4.22　二维比较设置界面

比较模式：支持进入误差带触发和进入误差带以及等于触发，两者的区别在于进入误差带之后，是否要比对轴的位置严格等于设置的目标位置，当严格要求设置的目标位置和当前位置相等时，会提高最终加工的精确性，但也可能会出现因为速度过快导致的位置不触发现象。

PWM：

PWM 即“脉宽调节”功能，和位置比较功能复用，当在“高速 IO 配置”界面勾选“PWM”功能即可激活 PWM 参数设置界面，如图 4.23 所示。

图 4.23　PWM 设置界面

PWM 界面中的设置参数，包括功能的使能与禁止，PWM 占空比和 PWM 频率。

Internal IO 映射：

Internal IO 映射，是指将 PMC600 控制器中的通用 IO 资源和变量绑定在一起，方便在应用程序中使用。可以使用 DWORD 类型的变量对所有的输入口或者输出口进行映射，也可以使用 BOOL 类型的变量对单个输入或者输出点进行映射。如图 4.24 所示。

变量	映射	通道	地址	类型	单元	描述
Application.PLC_PRG.I...		Genral_IO_IN	%ID0	DWORD		
Application.PLC_PRG.I...		Genral_IO_Out	%QD0	DWORD		
		Bit0	%QX0.0	BOOL		
		Bit1	%QX0.1	BOOL		
		Bit2	%QX0.2	BOOL		
		Bit3	%QX0.3	BOOL		
		Bit4	%QX0.4	BOOL		
		Bit5	%QX0.5	BOOL		
		Bit6	%QX0.6	BOOL		
		Bit7	%QX0.7	BOOL		
		Bit8	%QX1.0	BOOL		
		Bit9	%QX1.1	BOOL		
		Bit10	%QX1.2	BOOL		
		Bit11	%QX1.3	BOOL		
		Bit12	%QX1.4	BOOL		
		Bit13	%QX1.5	BOOL		
		Bit14	%QX1.6	BOOL		
		Bit15	%QX1.7	BOOL		
		Bit16	%QX2.0	BOOL		
		Bit17	%QX2.1	BOOL		
		Bit18	%QX2.2	BOOL		
		Bit19	%QX2.3	BOOL		
		Bit20	%QX2.4	BOOL		
		Bit21	%QX2.5	BOOL		
		Bit22	%QX2.6	BOOL		

图 4.24　Internal IO 映射

4.2 串口协议配置

在网络配置（Network Configuration）栏，可以对主机的串口功能进行配置，包括 RS232 串口，RS485 串口，其中串口都支持 modbus RTU、MODBUS ASCII 以及串口自由协议。如图 4.25 所示，单击 PMC600 的主机，会弹出串口及网络功能配置的界面。

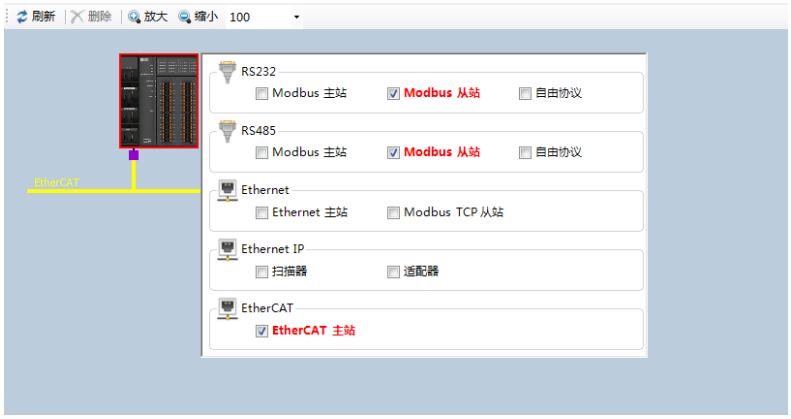

图 4.25　串口及网络功能配置

选择对应的功能，对应的指示文字会变成红色，同时在左侧设备栏，会增加对应的设备信息，如选择 232 作为从站时，会弹出“Modbus_Slave_RS232_COM”，以及选择 485 作为从站时“Modbus_Slave_RS485_COM”。

Modbus 从站：

参数设置：在串口配置中可以选择常见的串口参数配置，包括波特率，奇偶校验，数据位，停止位等，如图 4.26 所示。也可以设置 Modbus 从站的站号。

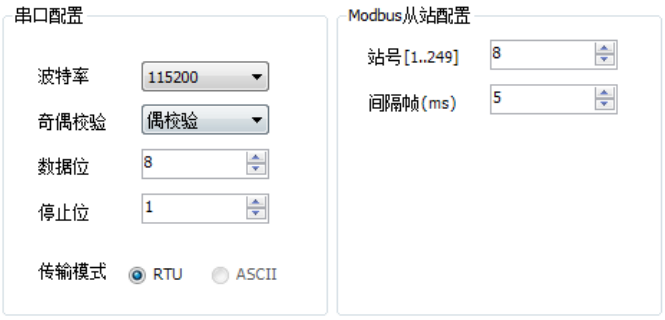

图 4.26　Modbus 从站参数设置界面

设置完串口参数之后，随工程一起下载到 PMC 控制器中生效。

Modbus 地址：在 PMC 的通信库“PMC_Communication”中，已经定义了多种不同数据类型的全局变量数组，如表 4.1 所示。

表 4.1 不同数据类型的全局变量数组

数据类型	数据长度	类型名称	数量
BOOL	1 位	M	4000
BYTE	8 位	ByteToM	500
INT	16 位	DWS	5000
UINT	16 位	DWU	5000
DINT	32 位	DDS	5000
UDINT	32 位	DDU	5000
REAL	32 位	DR	2500
String	31 个字符	DS	150

从上面的表格可以看出，在 PMC_Communication 通信库中定义了 4000 个 BOOL 型的全局变量，变量名称为 M0 – M3999，对应的 Modbus 地址为 0x1 – 0x4000。

ByteToM 型数组是 1 个大小为 500 的 BYTE 型数组，变量名称为 ByteToM，数组首地址对应的 Modbus 首地址为 4x4001。

DWS 型数组是一个包含了 5000 个 INT 数据类型的数组，变量名称为 DWS[0] – DWS[4999]，对应的 Modbus 地址为 4x1 – 4x5000。

DWU 型数组是一个包含了 5000 个 UINT 数据类型的数组，变量名称为 DWU[0] – DWU[4999]，对应的 Modbus 地址为 4x5001 – 4x10000。

DDS 型数组是一个包含了 5000 个 DINT 数据类型的数组，变量名称为 DDS[0]-DDS[4999]，对应的 Modbus 地址为 4x10001 – 4x20000。

DDU 型数组是一个包含了 5000 个 UDINT 数据类型的数组，变量名称为 DDU[0]-DDU[4999]，对应的 Modbus 地址为 4x20001 – 4x30000。

DR 型数组是一个包含了 2500 个 REAL 数据类型的数组，变量名称为 DR[0] – DR[2499]，起始地址为 4x30001 – 4x35000。

DS 型数组是一个包含了 150 个 STRING 类型的数组，每个 STRING 字符串包含 31 个字符，变量名称为 DS[0] – DS[149]，对应的 Modbus 地址为 4x35001 – 4X37385。

除了全局变量设置以外，PMC_Communication 库中还提供专门的功能块对上述全局变量进行复位或者设置值的操作，如“LS_Modbus_ResetM”是对 M 变量进行复位操作，“LS_Modbus_SetM”是对 M 变量进行置位操作。“LS_Modbus_SetByteToM”，是对 ByteToM 变量进行批量或者单个变量进行值设置。其他类型的全局变量也有类似的功能块。

自由协议的参数设置：

对串口的功能配置为“自由协议”后，在左侧的功能目录树下，会新增“Free_Protocol_RS232_Com”，或者是“Free_Protocol_RS485_Com”。

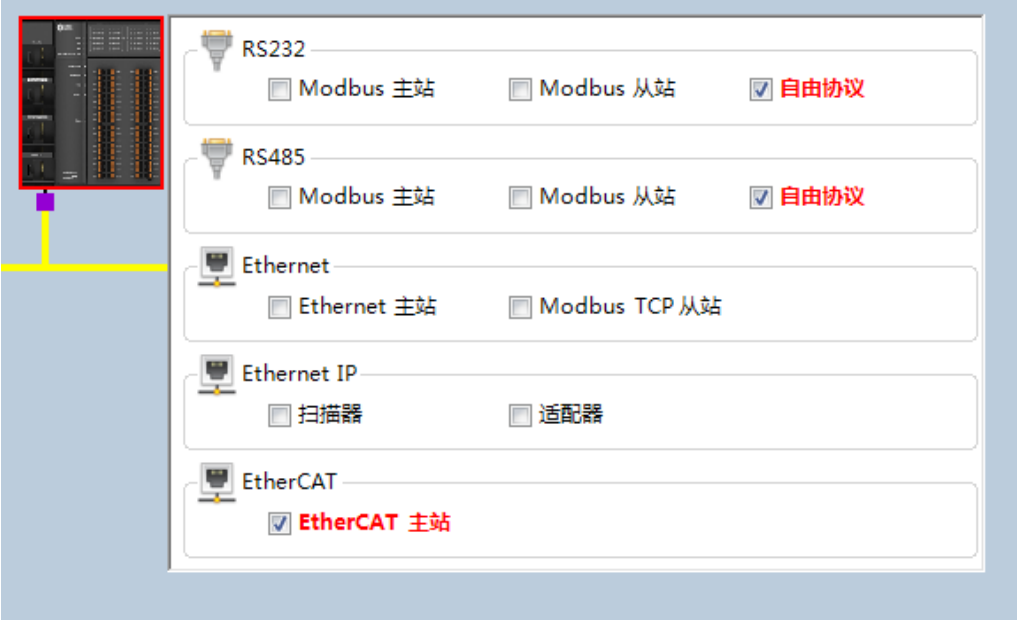

图 4.27　自由协议选择界面

以 RS232 串口设置为自由协议为例，双击“Free_Protocol_RS232_Com”，可以在参数设置界面设置相关的参数，如图 4.28 所示。

图 4.28　自由协议设置界面

在进行自由协议通信之前，需要设置常见的串口参数，包括波特率、奇偶校验方式、数据位、停止位等。

自由协议中的 IO 映射：

如图 4.29 所示，在 Internal IO 映射界面中，可以对串口自由协议进行变量映射，可以很方便的将通信的通道映射到程序中的变量中去，在程序中直接对变量进行操作即可实现对串口自由协议的操作。

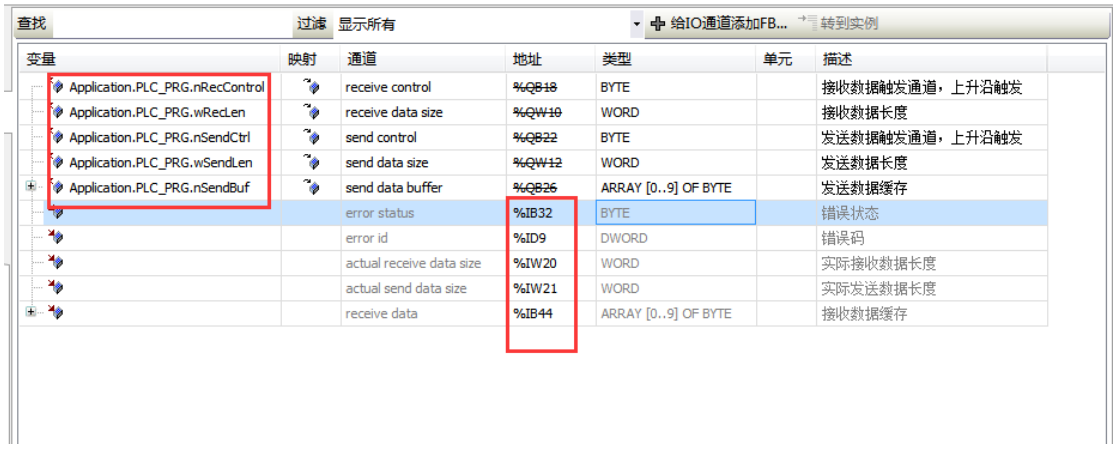

图 4.29　Internal IO 映射界面

自由协议中的功能块：

串口自由协议中定义了功能块，来实现串口的打开关闭，以及串口的读写操作，包括针对 232 串口的“LS_RS232_Open”、“LS_RS232_Close”、“LS_RS232_Read”、“LS_RS232_Write”，以及针对 485 串口的“LS_RS485_Open”、“LS_RS485_Close”、“LS_RS485_Read”、“LS_RS485_Write”。详细信息及使用方法请参考《PMC600 中型 PLC 用户手册基础篇》”中的“Communication 通讯库”章节。

4.3 以太网协议配置

PMC600 支持以太网 Modbus TCP 主站（Master）和 Modbus TCP 从站（Slave）模式，在左侧的工程列表中增加主站或者从站，并对其参数进行设置即可实现 Modbus TCP 的通信。

在功能图中，勾选“Modbus TCP 从站”，左侧工程项目列表会增加“Modbus_TCP_Slave(Modbus TCP Slave)”，如图 4.30 所示。

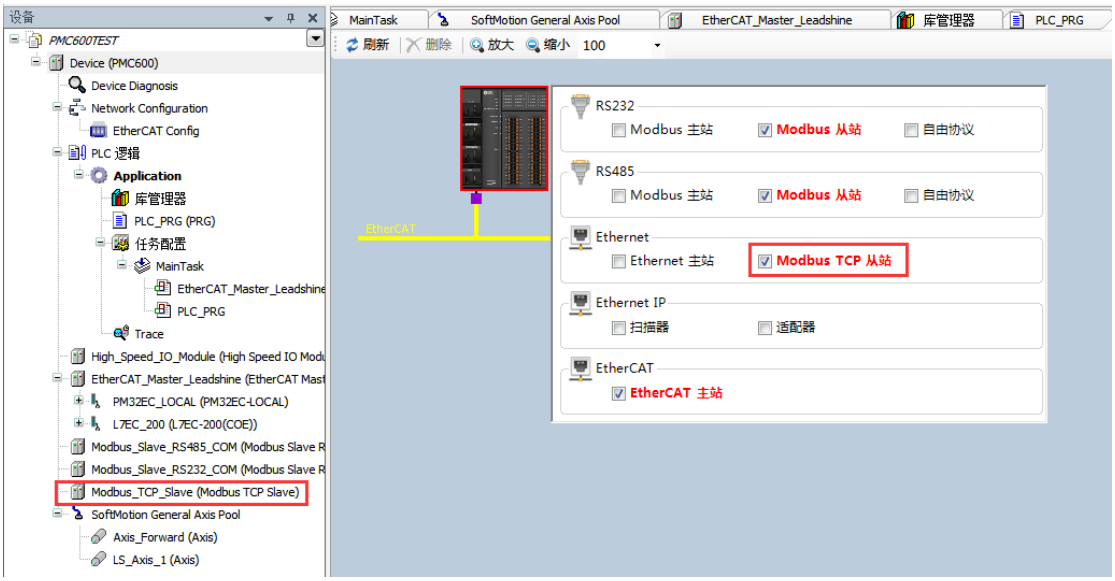

图 4.30　以太网通讯协议选择界面

双击“Modbus_TCP_Slave”，在弹出的界面中可以进行从站参数设置，如图 4.31 所示。

图 4.31　以太网通讯协议从站参数设置

从站端口：当前控制器的从站端口号。

帧间隔：设置数据帧之间的间隔时间。

注意：网络通信首先会对 IP 地址进行设置，PMC600 提供了对应的功能块“LS_TCP_SetIpAddress”进行 IP 地址设置。在应用程序中调用即可实现对 PMC600 的 IP 地址修改。

如图 4.32 所示，在功能图中勾选“Ethernet 主站”，左侧工程项目列表会增加“Ethernet”主站，选中“Ethernet”，右键选择“添加设备”，选择“Modbus TCP Master”，左侧列表中增加此项内容后，选中“Modbus TCP Master”，右键选择“Modbus TCP Slave”。就设置了 PMC600 中的以太网作为 Modbus TCP 主站的功能，并且在主站下添加了一个 Modbus TCP 从站。

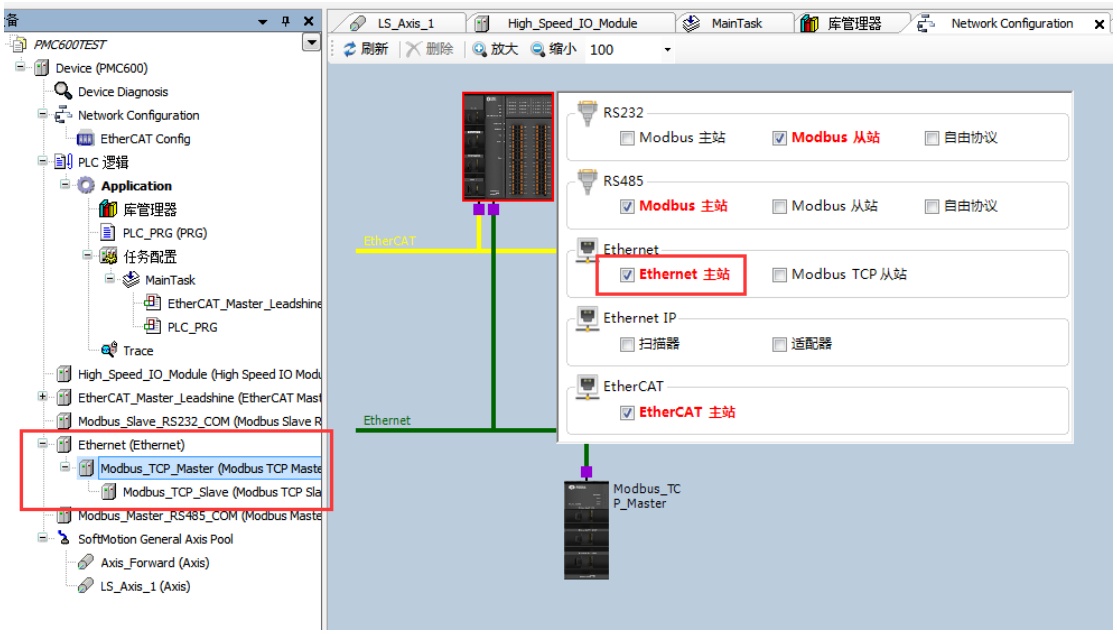

图 4.32　Modbus TCP 主站

双击“Ethernet”选项，可以在“通用”栏中，设置主站的 IP 地址，通过“接口”后的选择按钮，来找到对应的网口信息，如图 4.33 所示。

图 4.33　网口信息

图 4.34 为弹出的“网络适配器”界面，可以看到当前连接的 PMC 控制器的网络信息，其中 etho 是 EtherCAT 网口的地址，eth1 是第一个以太网口，eth2 是第二和第三个网口。选中 eth1 作为主站，确定后即完成了对网口的选择。

图 4.34　网络适配器界面

在 Ethernet Device I/O 映射中，将总线循环任务选择为“使用 MainTask”。

双击主站设备列表下的从站设备“Modbus_TCP_Slave”，在通用栏中修改对应从站的 IP 地址，响应超时时间以及端口号。在“Modbus_TCP_Slave 参数”中选中并修改“Unit_ID”选项，此参数即从站的节点号。然后在“Modbus 从站通道”中添加对应的 MODBUS 通信通道，如图 4.35 所示。

	名称	访问类型	触发器	读偏移	长度	错误处理	写偏移	长度	注释
0	Channel 0	Read Coils (函数代码 01)	循环, t#100ms	16#0000	1	保持最后的值			
1	Channel 1	Write Single Coil (函数代码 05)	循环, t#100ms				16#0000	1	
2	Channel 2	Read Input Registers (函数代码 04)	循环, t#100ms	16#0000	1	保持最后的值			

图 4.35　MODBUS 通信通道

第5章　iStudio 编程实例

本章通过一个实验系统，详细介绍 PMC600 中型 PLC 的基本使用方法。

5.1 实验系统硬件结构

本实验系统由运动控制器 PMC600、网络适配器 PM32EC-A0、扩展 IO 模块 PM32DX-A0、脉冲式步进电机及驱动器 M415B、总线式步进电机及驱动器 DM3E-556、电源模块 PM02PS-A0、按键和 LED 灯组成。硬件结构如图 5.1 所示；电路图参见硬件手册。

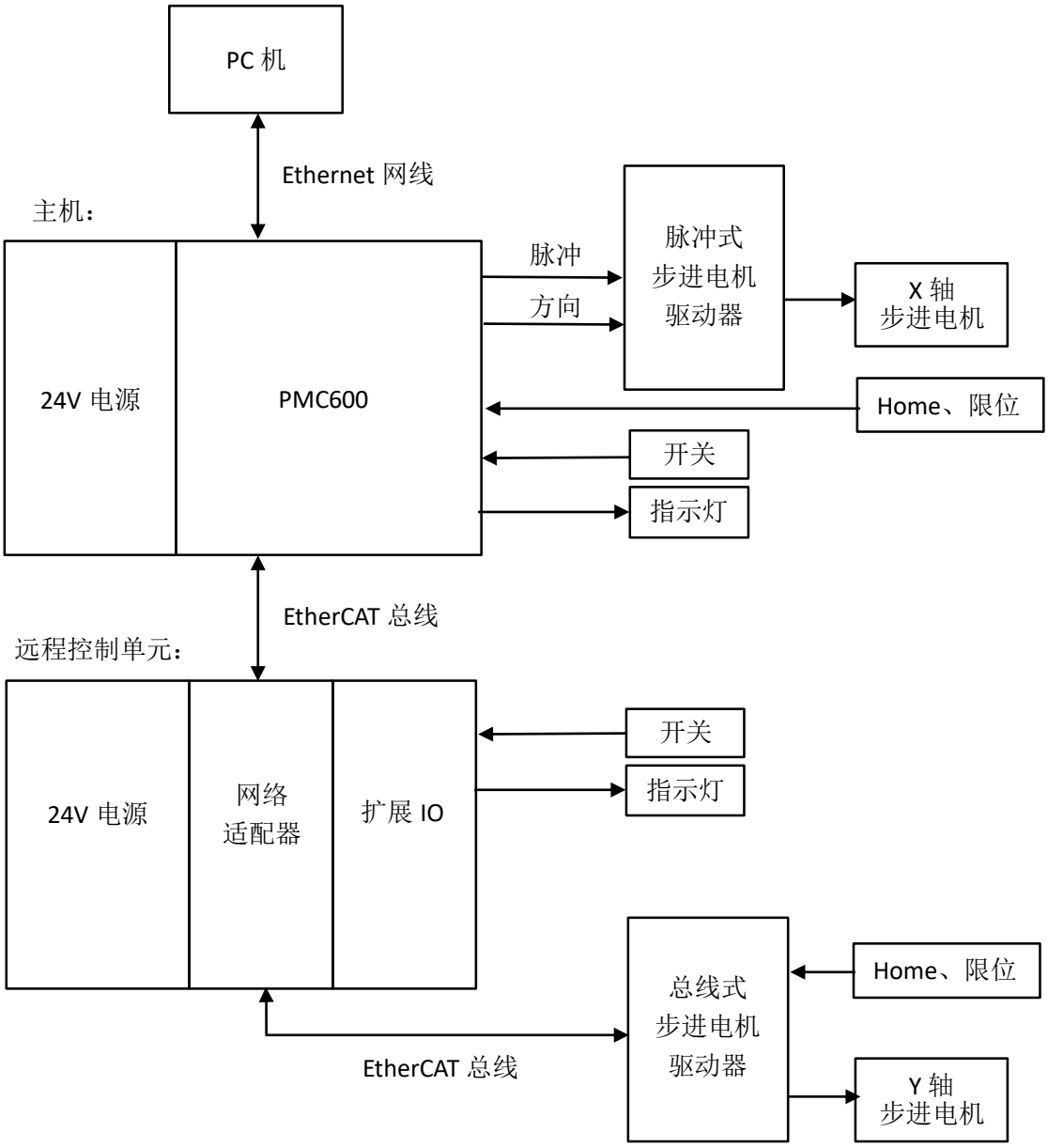

图 5.1　实验系统的硬件结构

5.2 设置 PC 机的 IP 地址

将 PMC600 的 EtherNet 口与 PC 机的网口连接，然后上电。

打开 Windows 的“网络和共享中心”界面，如图 5.2 所示；点击与 PMC600 连接的“以太网”，打开了“以太网状态”界面，如图 5.3 所示；点击上面的“属性”按钮，打开了“以太网属性”界面，如图 5.4 所示。

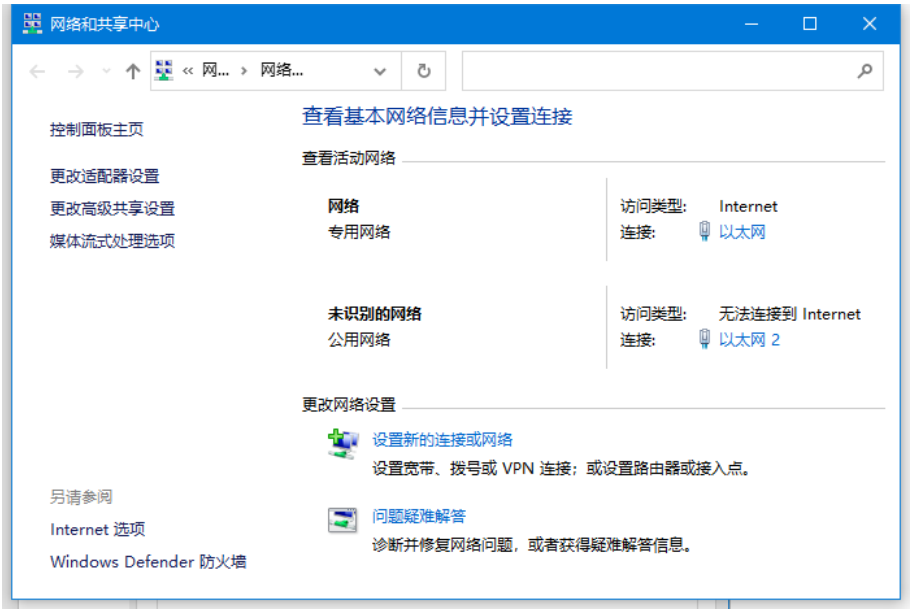

图 5.2 “网络和共享中心”界面

图 5.3 “以太网状态”界面

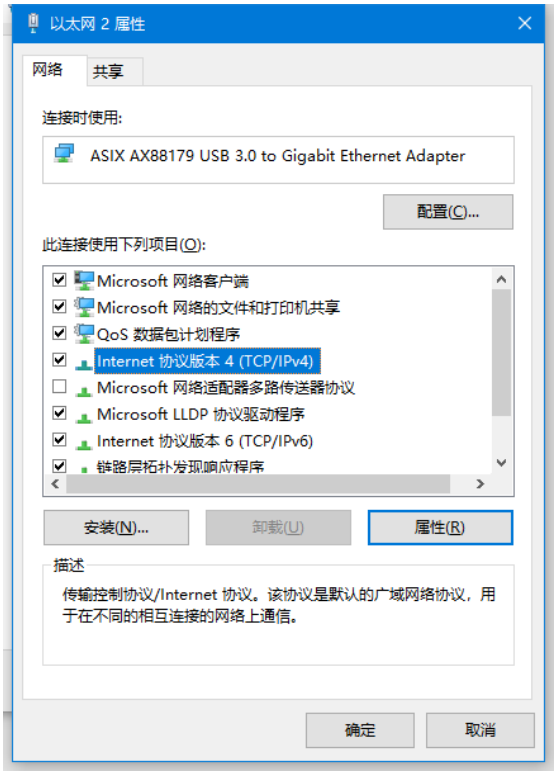

图 5.4 “以太网属性”界面

选择“Internet 协议版本 4（TCP、IPv4）”，再点击“属性”按钮，进入 Internet 协议版本 4（TCP、IPv4）属性界面，按图 5.5 所示，修改其参数，最后点击“确定”按钮。

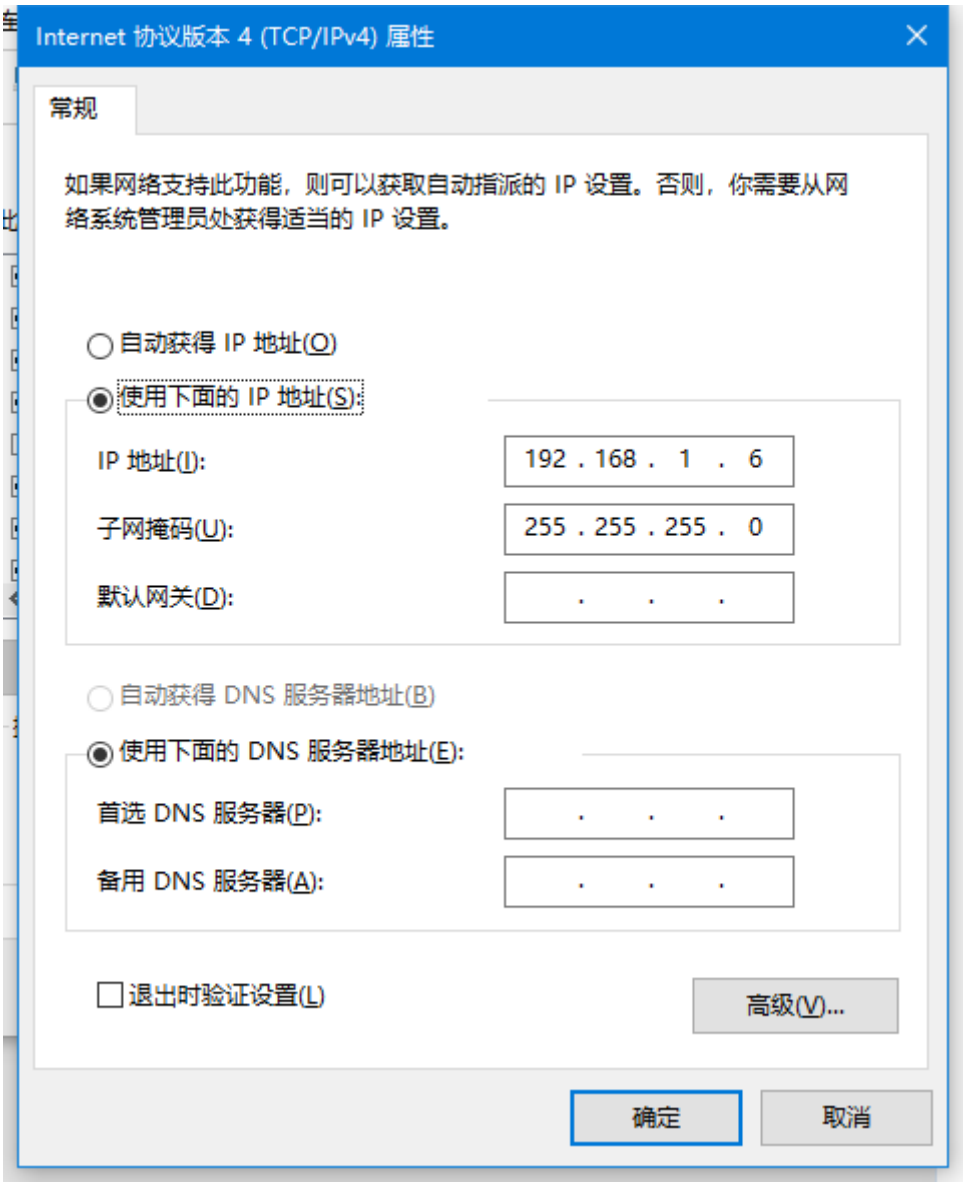

图 5.5 “Internet 协议版本 4（TCP、IPv4）”界面

5.3 启动编程环境

双击 iStudio 软件图标如图 5.6 所示，启动 iStudio 软件，界面显示如图 5.7 所示。

图 5.6 iStudio 软件图标

图 5.7　启动 iStudio 软件

点击“新建工程”快捷栏或者点击“文件”→“新建工程”，新建一个工程项目；在如图 5.8 所示的界面中选择“Standard project”(标准工程)，输入工程名称和保存位置，点击“确定”；然后在如图 5.9 所示界面中选择“设备”为 PMC600，编程语言选择“结构化文本（ST）”。

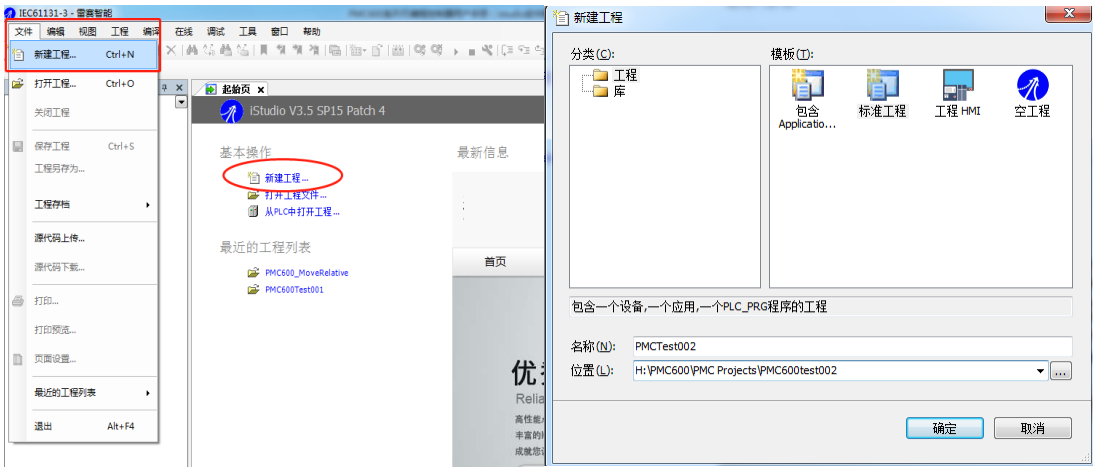

图 5.8　新建工程界面

图 5.9　“标准工程选择”界面

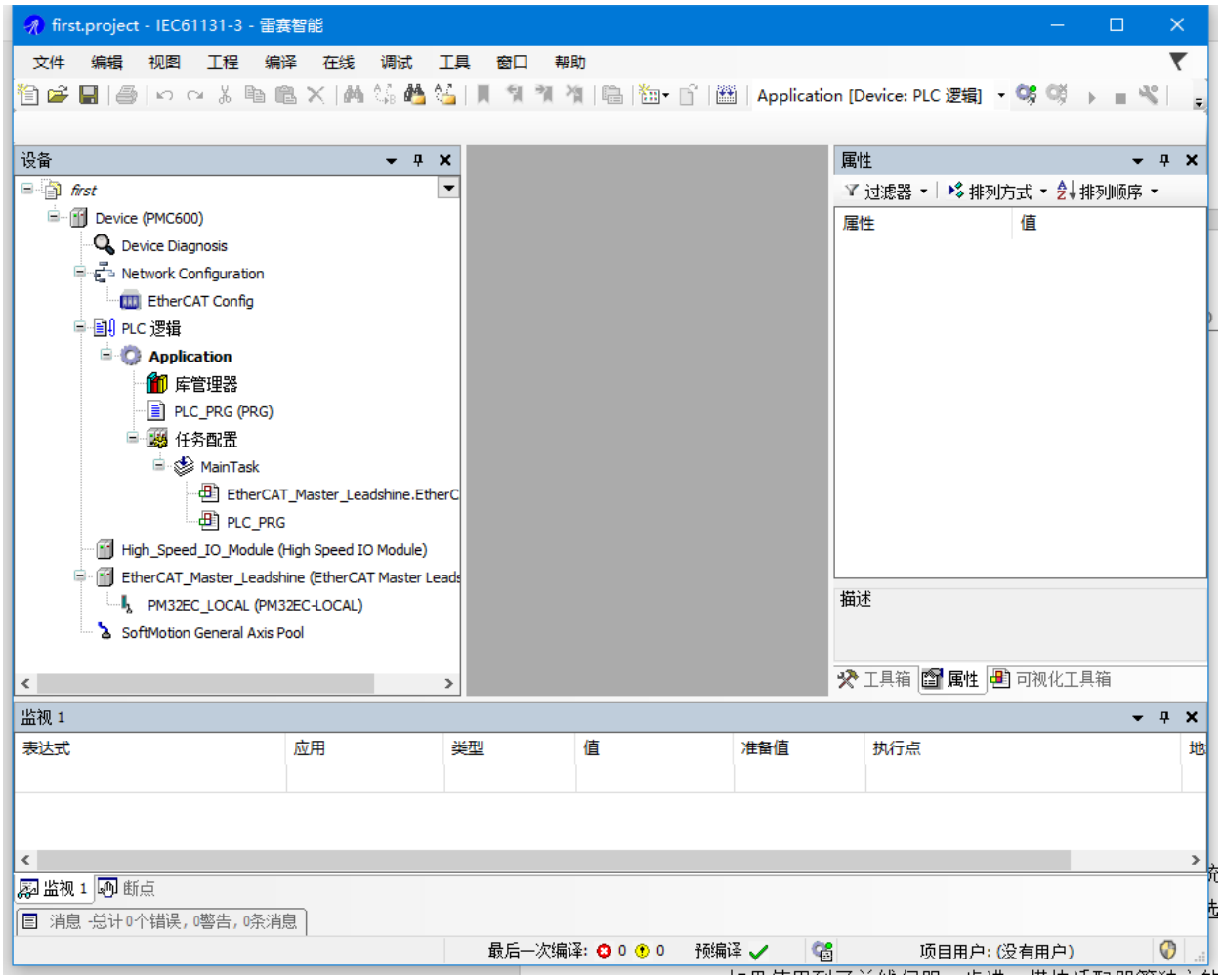

图 5.10　工程界面

5.4 网络拓扑的配置

如图 5.10 所示，在工程目录树中已经加入了 PMC600 主机中的高速 IO 单元，但网络适配器及其上面的 IO 模块和 2 个步进电机驱动器都还没有出现。

双击工程目录树中的“Device”，出现“Device”界面如图 5.11 所示。点击“扫描网络”，出现“选择设备”界面，选中“PMC-600”后，点“确定”。这时，“网关”和“节点”都显示绿点，表示软件和 PMC600 已连接。

用鼠标右键点击目录树中的“EtherCAT_Master_Leadshine”，选择“扫描设备”，如图 5.12 所示。几秒钟后出现扫描结果如图 5.13 所示。点击按钮“复制所有设备到工程中”，扩展 IO 模块、总线步进电机驱动器都自动地加入到软件系统中。参见图 5.14、5.15。

双击目录树中的“Network Configuration”，进入总线配置界面，如图 5.14 所示。在此界面可以手动配置总线伺服电机、步进电机和扩展模块。用鼠标拖动界面右侧中的模块总线上即可。

双击目录树中的“EtherCAT Configuration”，进入 IO 模块配置界面，如图 5.14 所示。在此界面可以手动配置 PMC 系列的 IO 模块。用鼠标拖动界面右侧中的 IO 模块到 PMC6

00 主机或网络适配器的右侧即可。

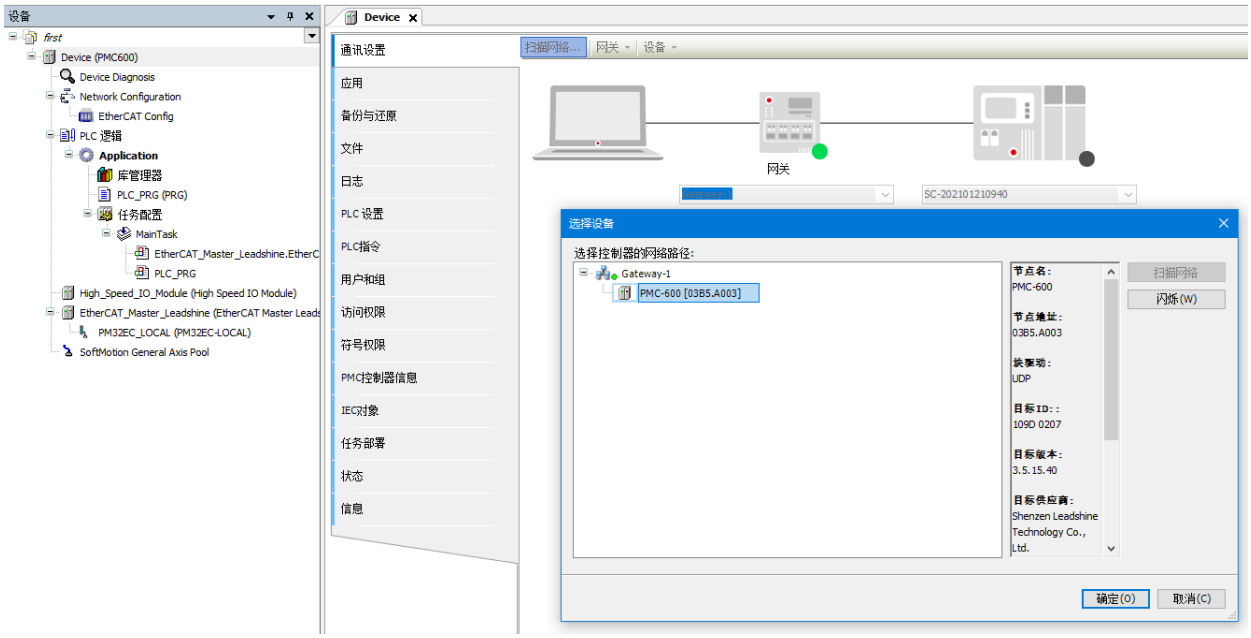

图 5.11　扫描网络，连接控制器

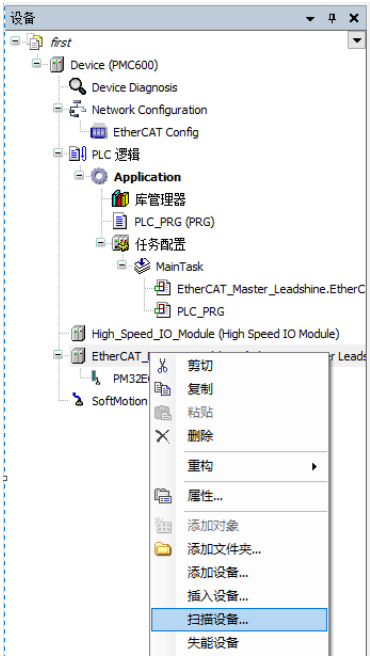

图 5.12　选择“扫描设备”

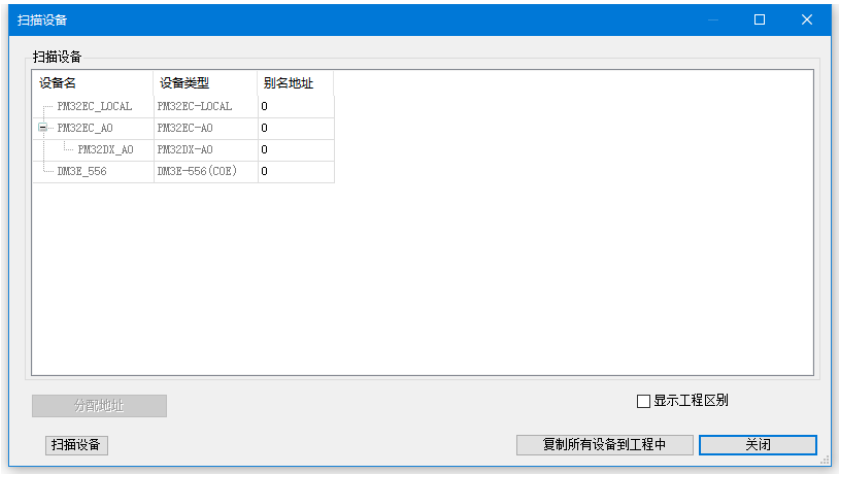

图 5.13　扫描设备的结果

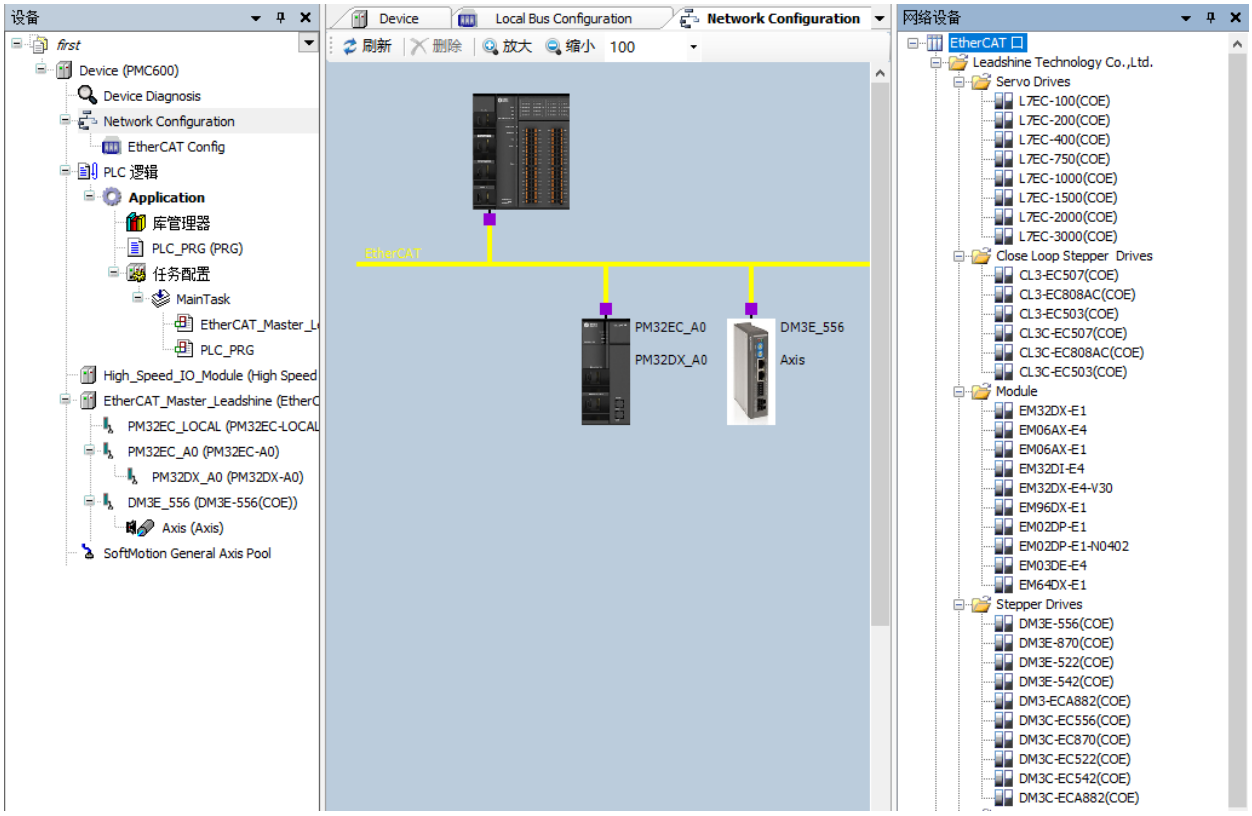

图 5.14　网络设备配置界面

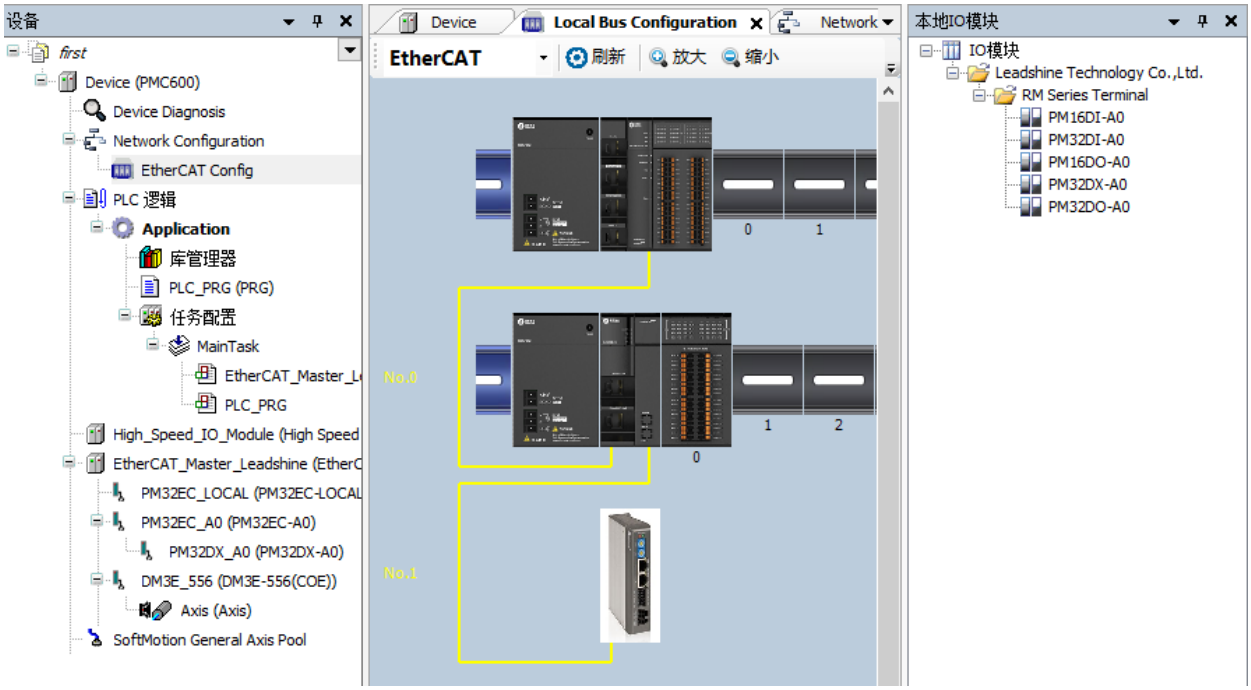

图 5.15　IO 模块配置界面

5.5 IO 的映射与测试

双击目录树中的“PLC_PRG”，进入编程界面，如图 5.16 所示，在变量定义区定义一个输入变量 In5、一个输出变量 Out7。

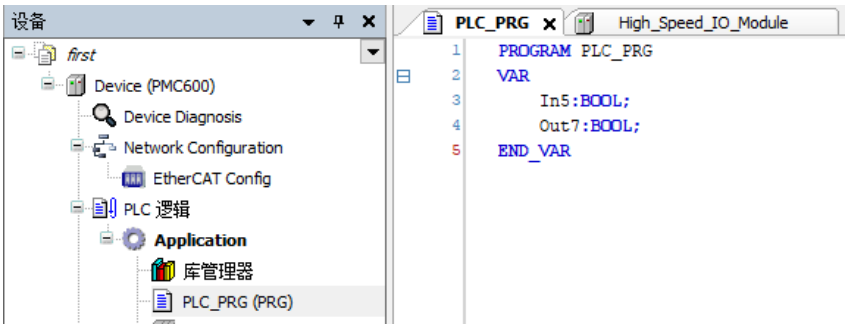

图 5.16　IO 模块配置界面

双击目录树中的“High_Speed_IO_Module”，进入 IO 设置界面，如图 5.17 所示；点击“Internal I/O 映射”栏，点击“变量”下的加号，显示 Genral_IO_IN 的所有变量。

在“通道”栏找到 Bit5，将鼠标移至该行“变量”栏中双击，出现符号“…”，如图 5.17 所示；点击该符号，出现“输入助手”界面；以次点击“Application”和“PLC_PRG”前的加号，显示已定义的变量；选择“In5”后，点击“确定”键，完成输入型号的映射。

以同样的方法，可完成输出变量 Out7 的映射，参见图 5.18。

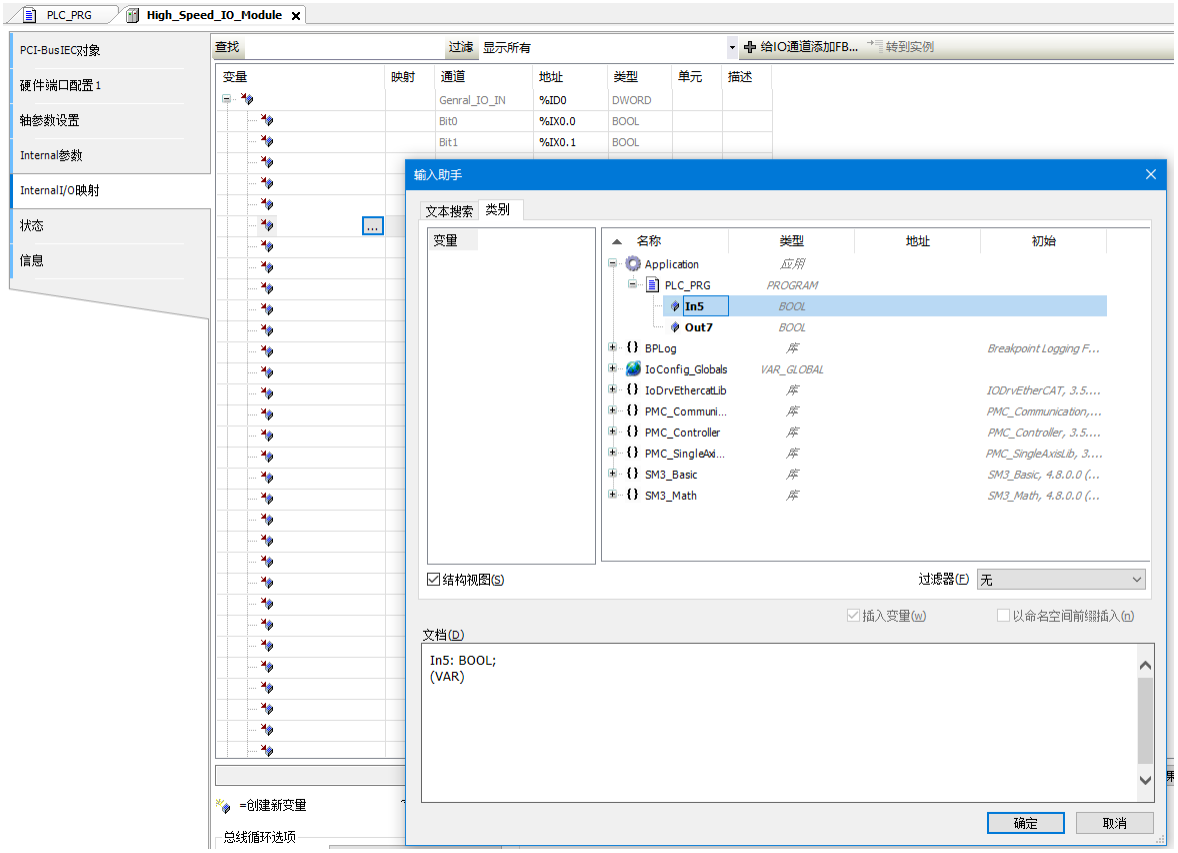

图 5.17　输入信号的映射

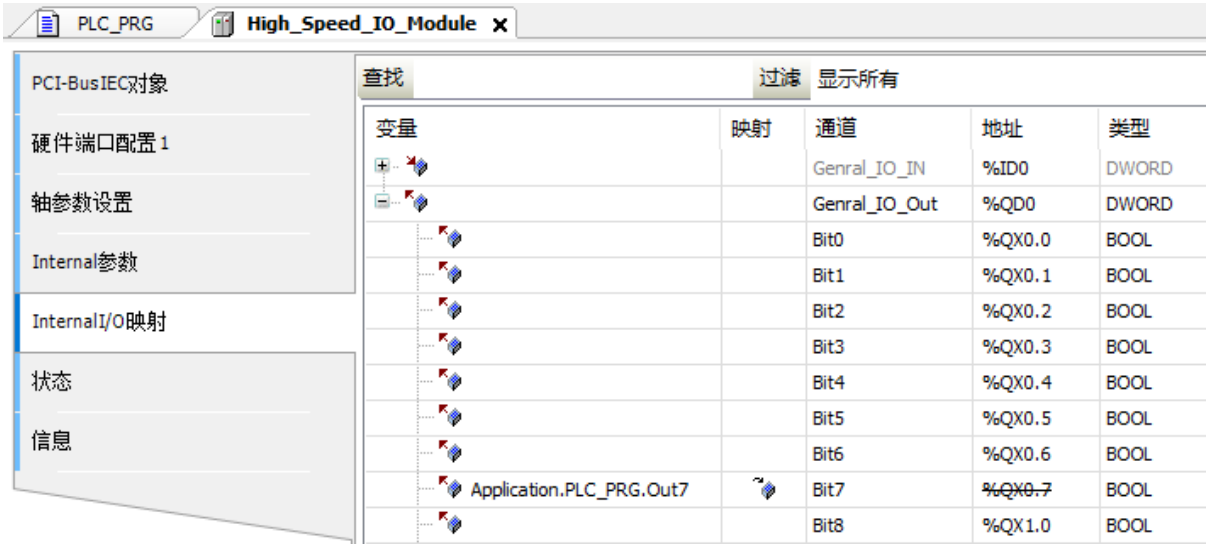

图 5.18　输出信号的映射

如图 5.19 所示，点击“登录”快捷键连接运动控制器，弹出一个对话框，选“是”；再点击“启动”快捷键，启动运动控制器中的程序。

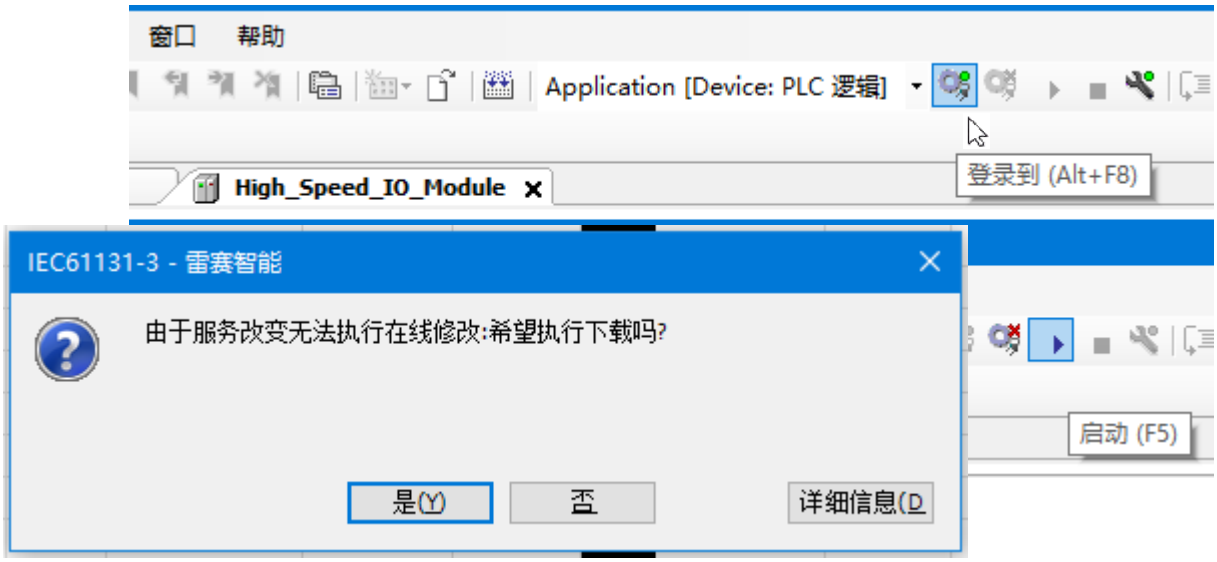

图 5.19　登录运动控制器并启动

此时，按下按钮 button5，可在“当前值”栏中看到与 Bit5 对应的值变为 TRUE，如图 5.20 所示。说明运动控制器接收到 button5 的输入信号，输入口电路连接正确。

鼠标点击 Genral_IO_Out 之下的 Bit7 行的“预备值”栏，出现“TRUE”，如图 5.21 所示；按 Ctrl+F7 键，“当前值”变为 TRUE，LED7 灯亮；再点击“预备值”栏，出现“FALSE”，按 Ctrl+F7 键，“当前值”变为 FALSE，LED7 灯灭。说明运动控制器发出 Out7 输出信号，输出口电路连接正确。

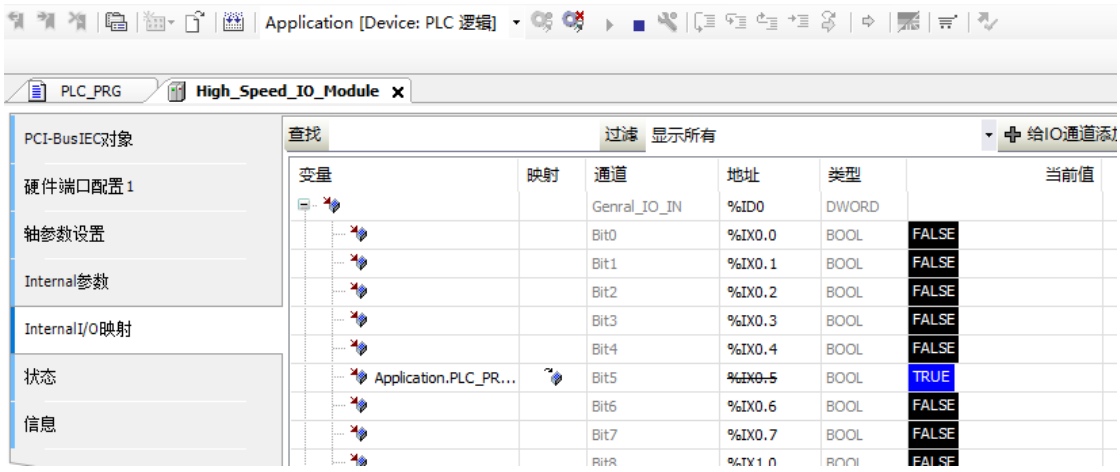

图 5.20　测试输入信号

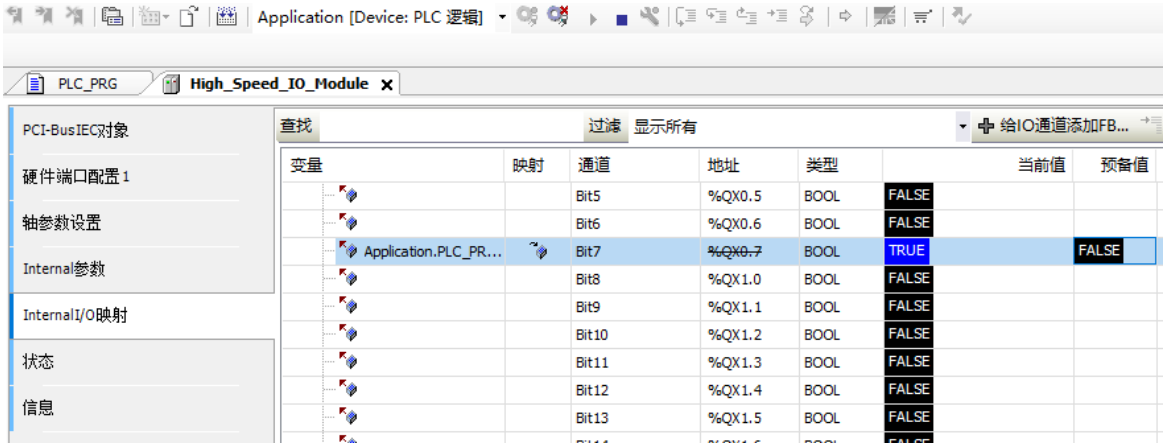

图 5.21　测试输出信号

扩展 IO 模块 PM32EC_A0 的输入、输出变量的映射方法和主机的 IO 映射方法相同。如图 5.22 所示，鼠标双击目录树中的“PM32EC_A0”，即可看到“EtherCAT I/O 映射”。

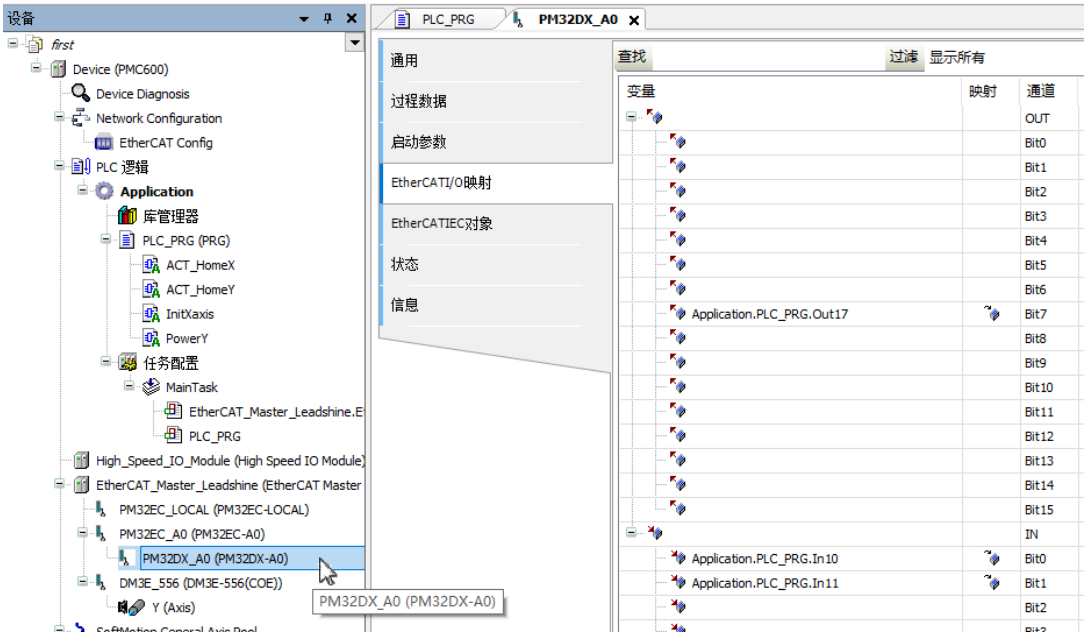

图 5.22　扩展 IO 模块的 IO 映射

5.6 PMC600 的电机接口设置及测试

1. 设置轴的端口

双击目录树中的“High_Speed_IO_Module”，进入 IO 设置界面；点击“硬件端口配置 1”栏，显示如图 5.23 所示界面。点击轴 0 前的方框，出现符号“√”，即选中轴 0，输出口 Out00、Out01 为电机的脉冲和方向信号；IN00、IN01、IN02 为原点信号、负限位和正限位信号。详见硬件手册。

点击“轴参数设置”栏，如图 5.24 所示界面，设置轴的参数。

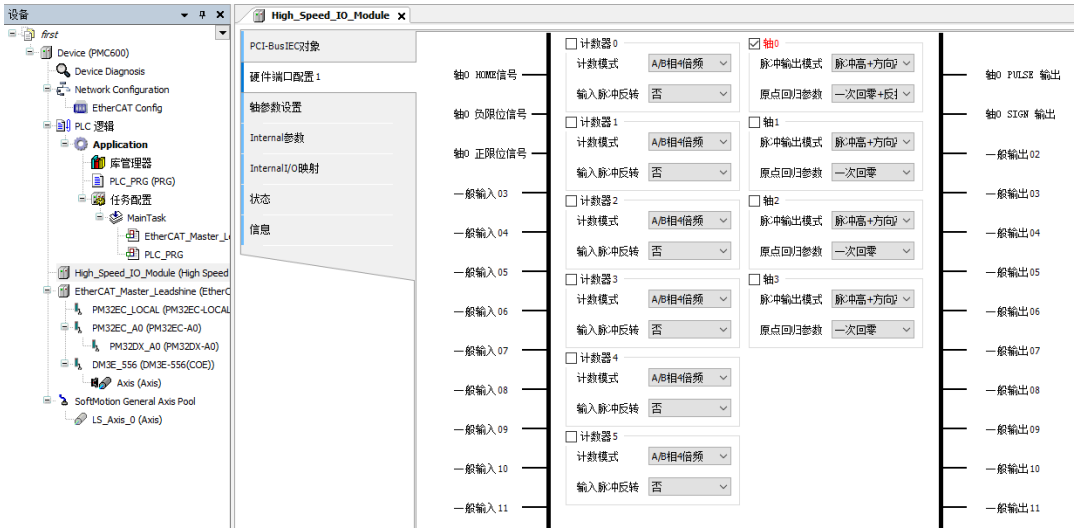

图 5.23　轴控制信号的设置

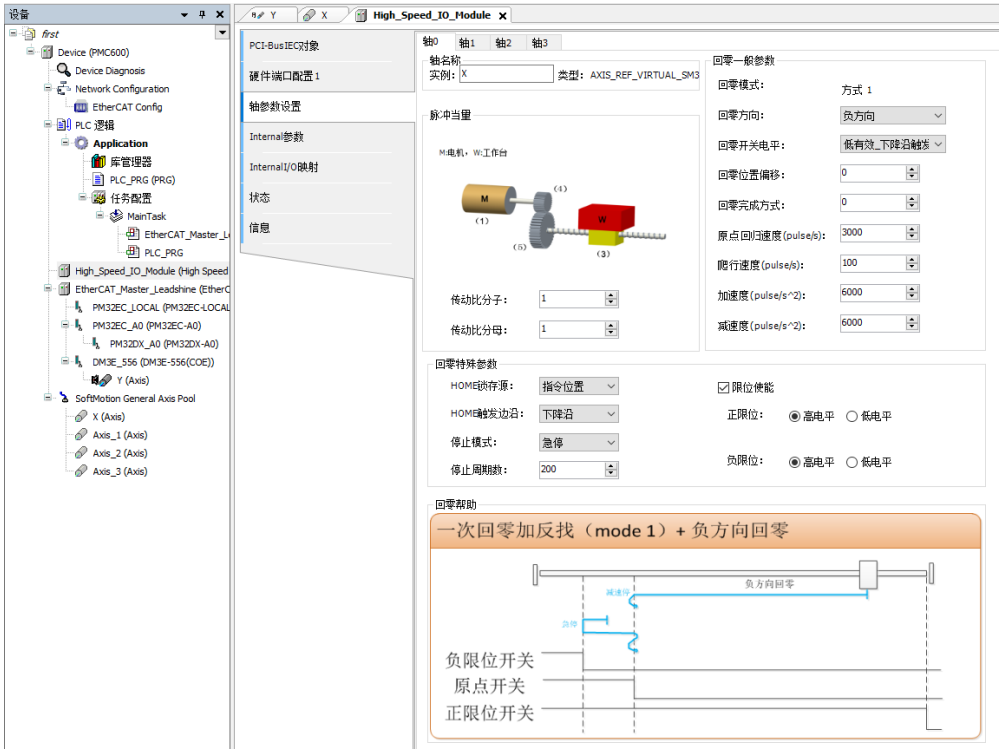

图 5.24　轴参数设置界面

单击目录树中的“SoftMotion General Axis Pool”下的轴号，可以修改轴的名称，如图 5.25 所示。

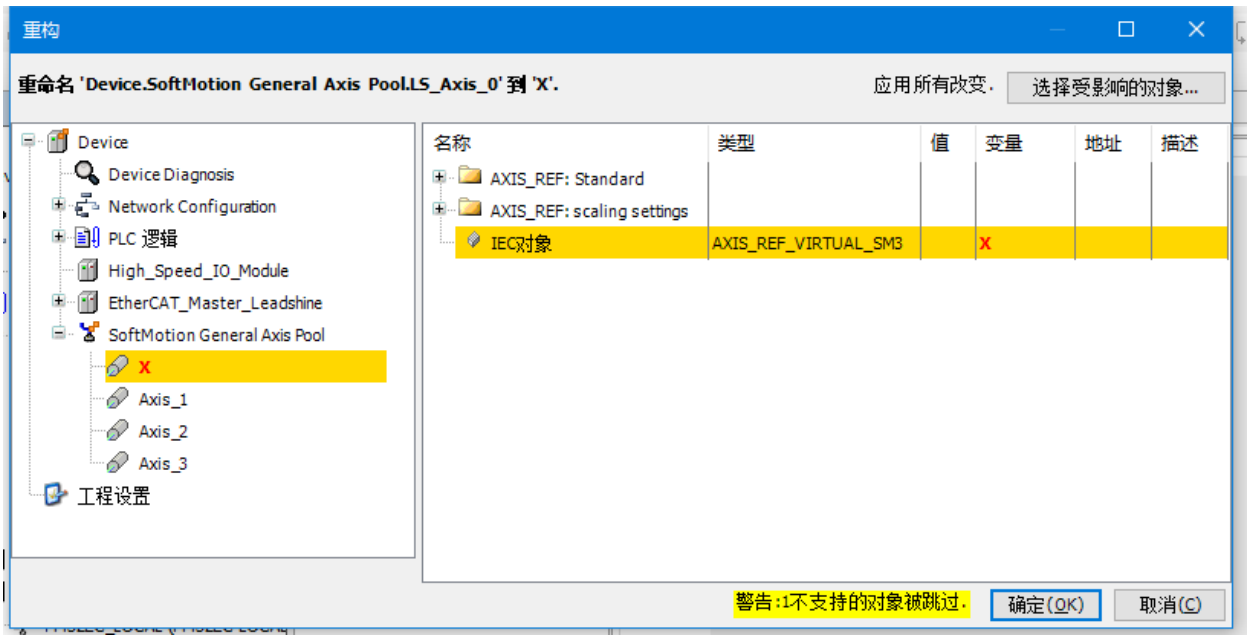

图 5.25　修改轴的名称

2.　轴初始化

除了完成上述设置外，还要在程序中调用子程序 LS_PulseAxis_MachineState 初始化脉冲轴。该子程序要求设置 4 个轴，所有还要再添加 3 个虚拟轴。

右键点击目录树中的“SoftMotion General Axis Pool”，选择“添加设备”，打开如图 5.26 所示的界面。

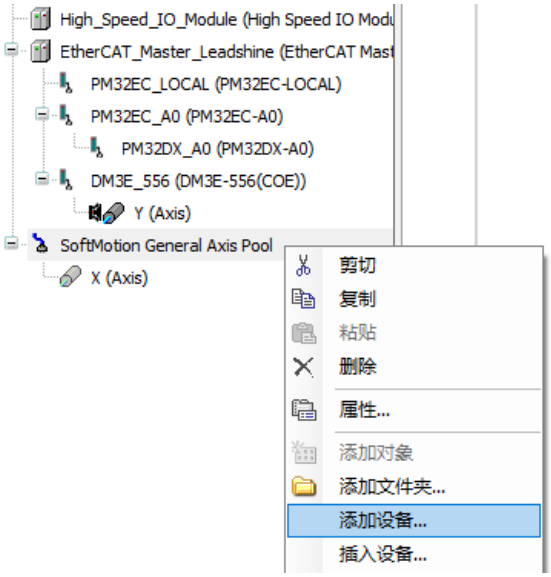

图 5.26　选择“添加设备”

如图 5.27 所示，单击“虚拟驱动器”下的“Axis”，再点击“添加设备”按钮，即添

加了一个虚拟轴。然后按此方法再添加 2 个虚拟轴。

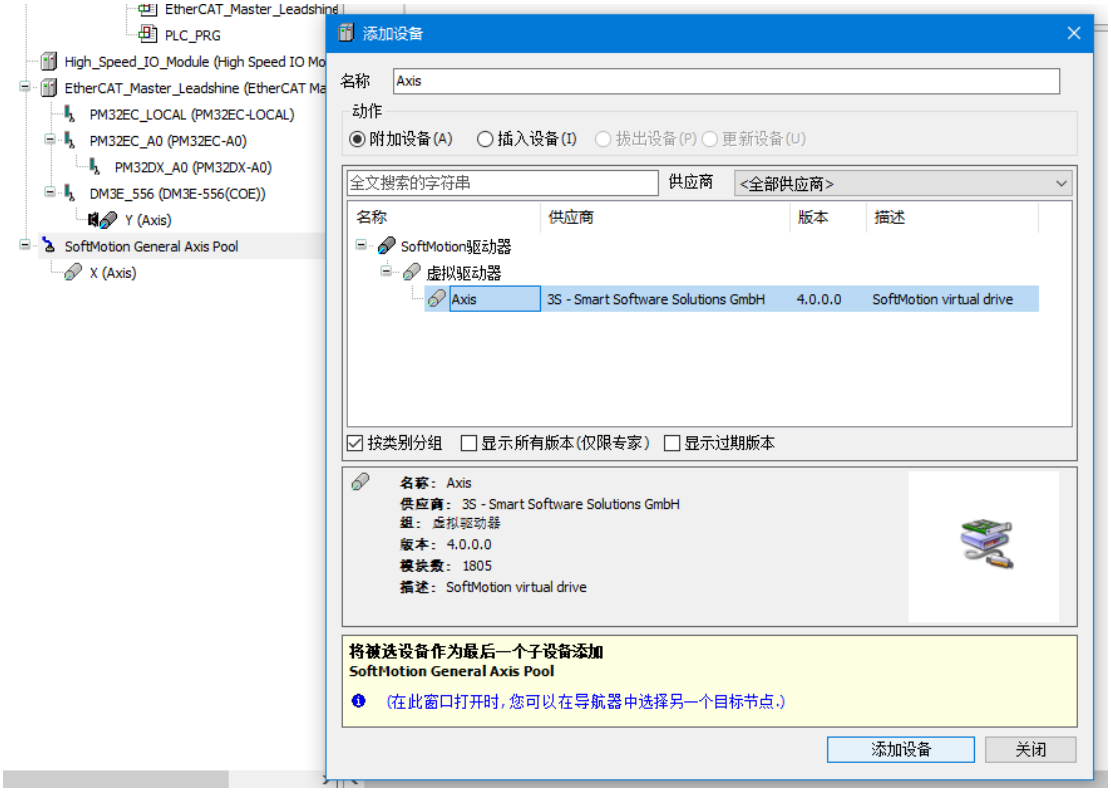

图 5.27　添加虚拟轴

在工程目录树中，用鼠标右键单击“PLC_PRG”，选“添加对象”→“动作”，如图 5.28 所示。所谓“动作”，可以理解为子程序。

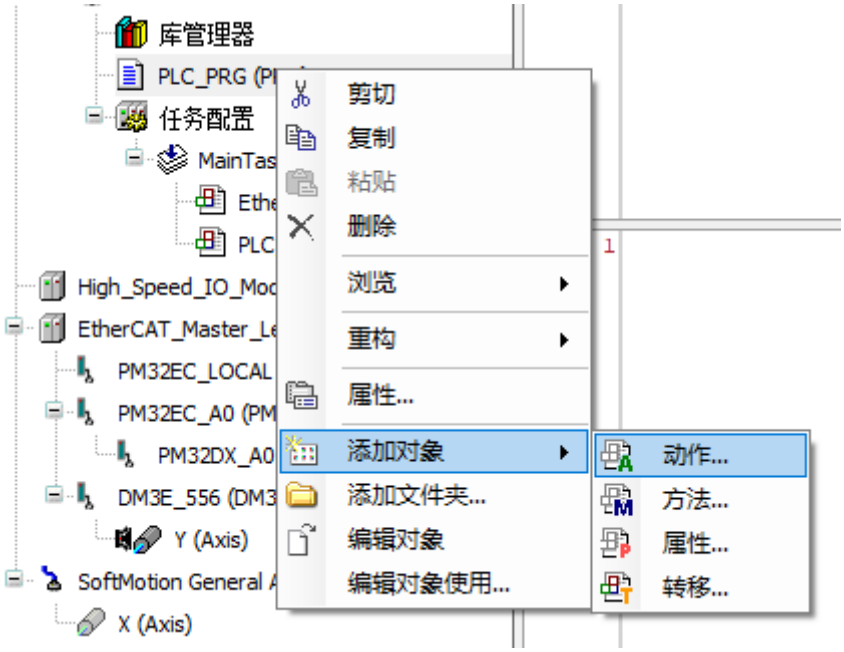

图 5.28　添加动作

设置新动作的名称、选择实现语言，如图 5.29 所示。

在功能块编辑区里点击鼠标右键，选“插入运算块”，如图 5.30 所示。在输入助手中的“文本搜索”栏里，输入“LS_PulseAxis_MachineState”，在搜索结果里选中该子程序，如图 5.31 所示，然后点击“确定”。

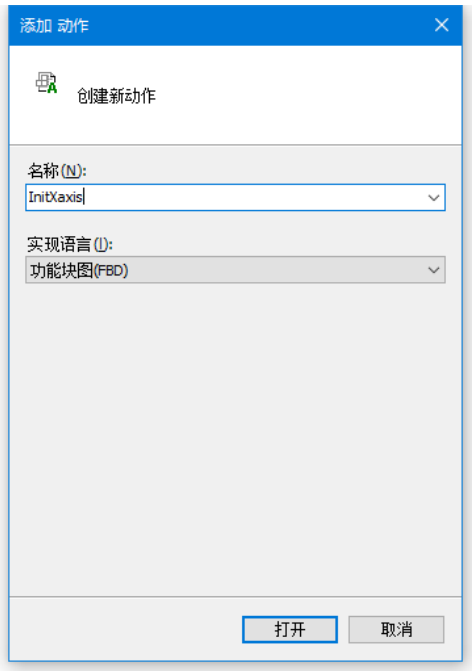

图 5.29　设置“动作”的名称

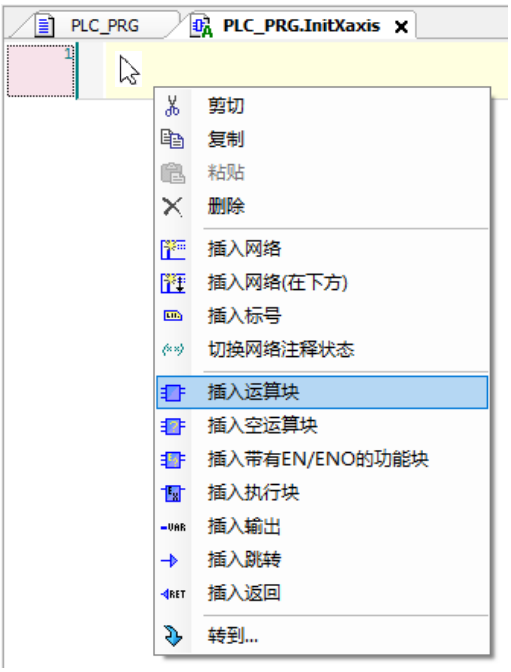

图 5.30　插入运算块

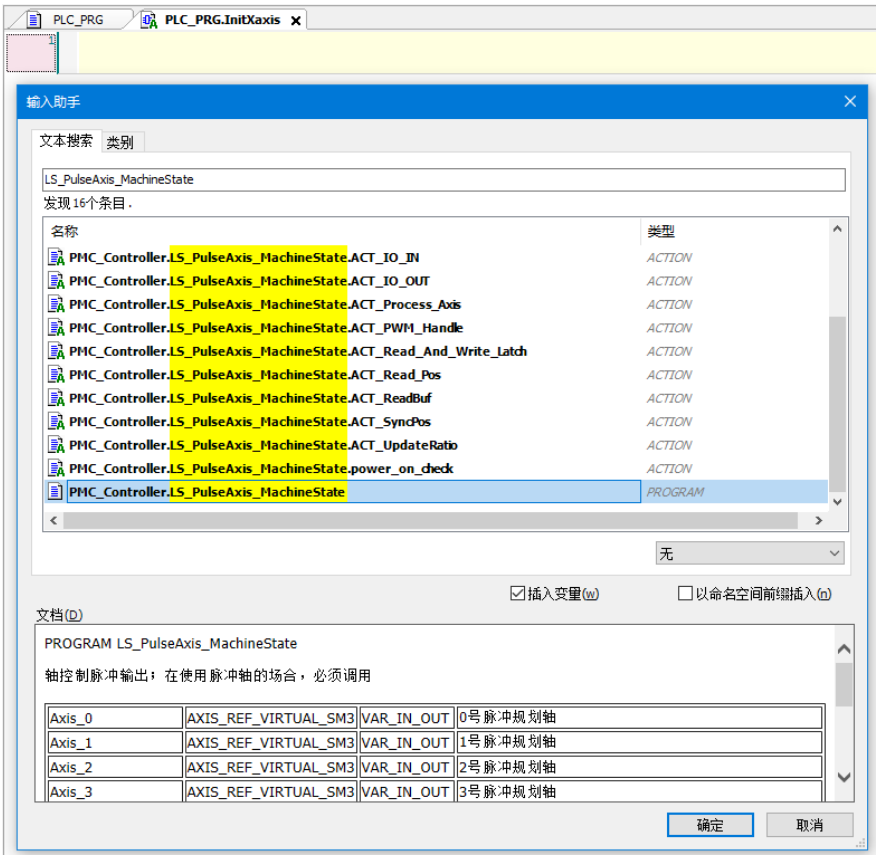

图 5.31　搜索要插入的模块

按照图 5.32 所示，设置 LS_PulseAxis_MachineState 的参数。有问号处，用退格键清除。LS_PulseAxis_MachineState 是程序不是功能块，故不需要声明模块名称。

在已调用的模块下方点击鼠标右键，选择“插入网络”，如图 5.32 所示。然后添加电机使能模块 MC_Power；并自动声明模块的名称，如图 5.33 所示。

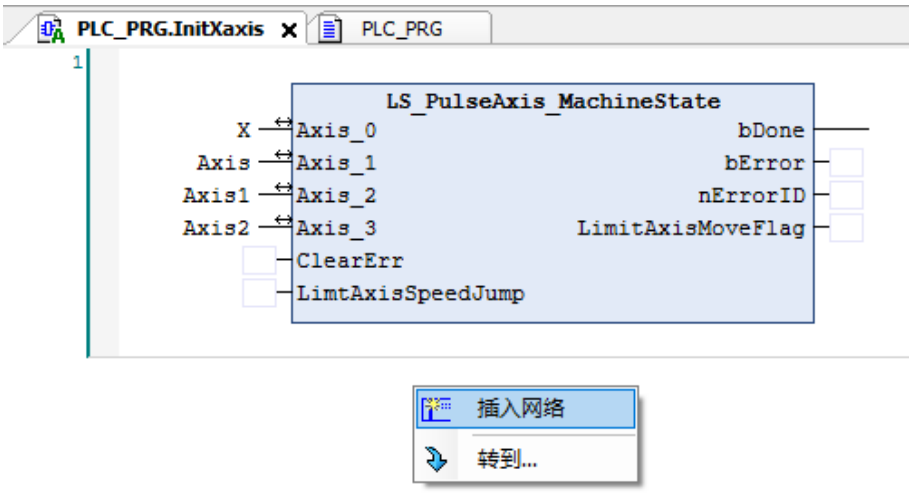

图 5.32　搜索要插入的模块

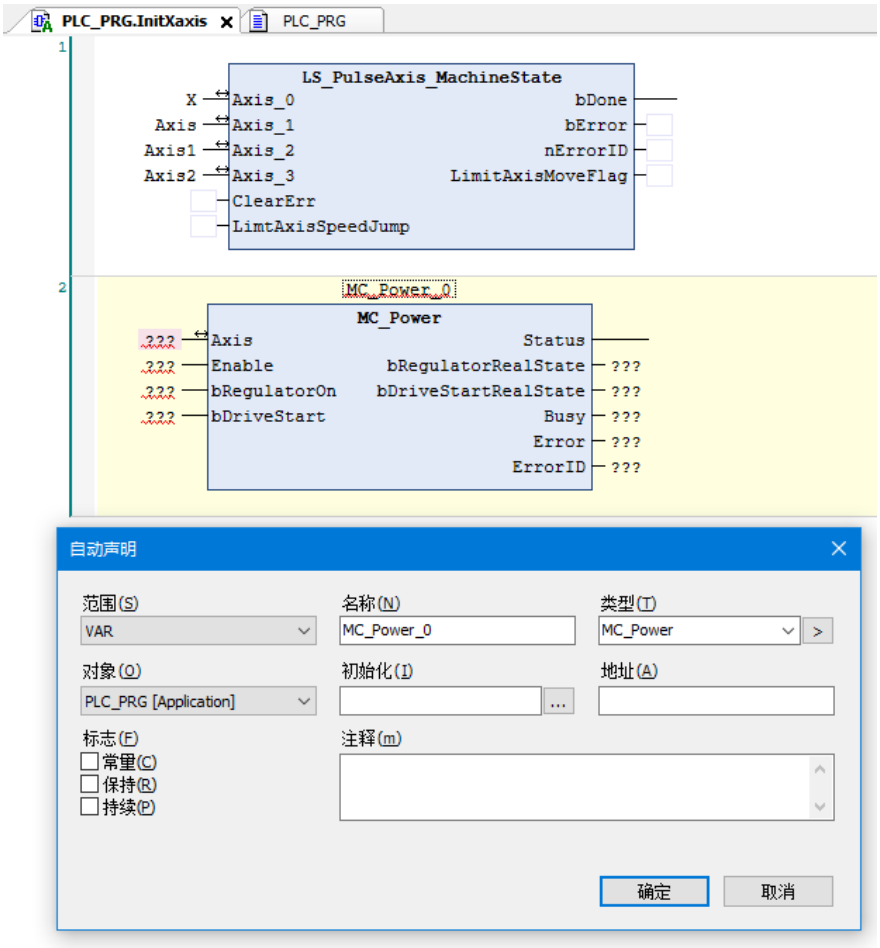

图 5.33　添加新模块并自动声明名称

完成后的“动作”InitXaxis 功能块程序如图 5.34 所示。

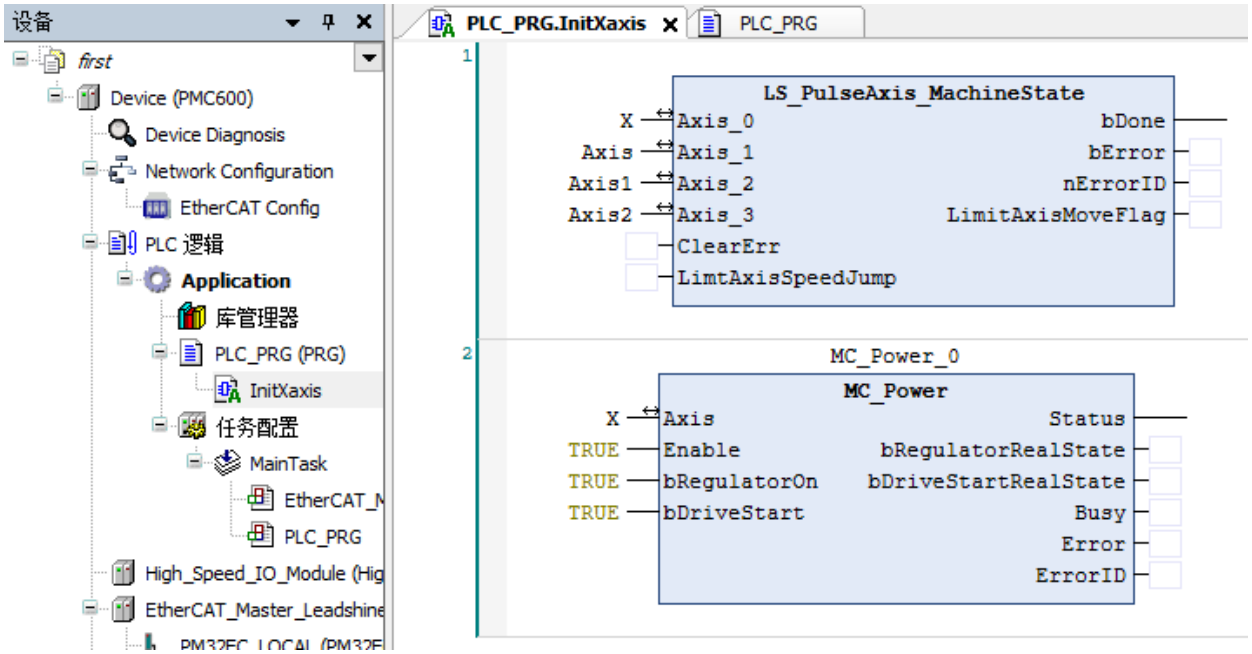

图 5.34　功能块程序 InitXaxis

3.　调用点位运动模块

在 PLC_PRG 编辑区点击右键，选“输入助手”，如图 5.35 所示。在“文本搜索”栏里，输入“MC_MoveRelative”，在搜索结果里选中该模块，并设置模块的名称，如图 5.36 所示；得到的功能模块代码如图 5.37 所示。

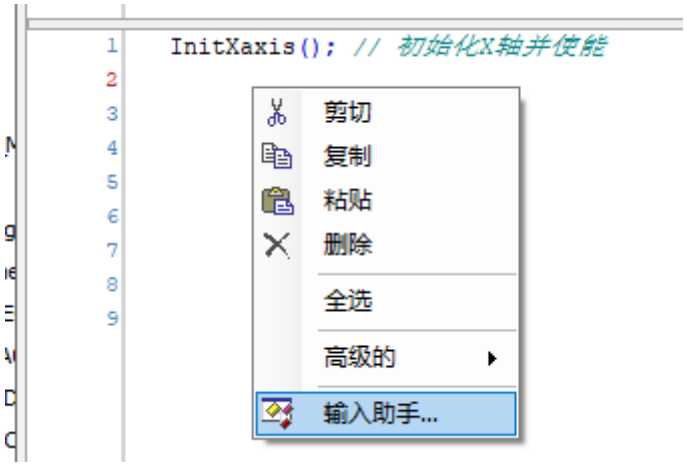

图 5.35　选中“输入助手”

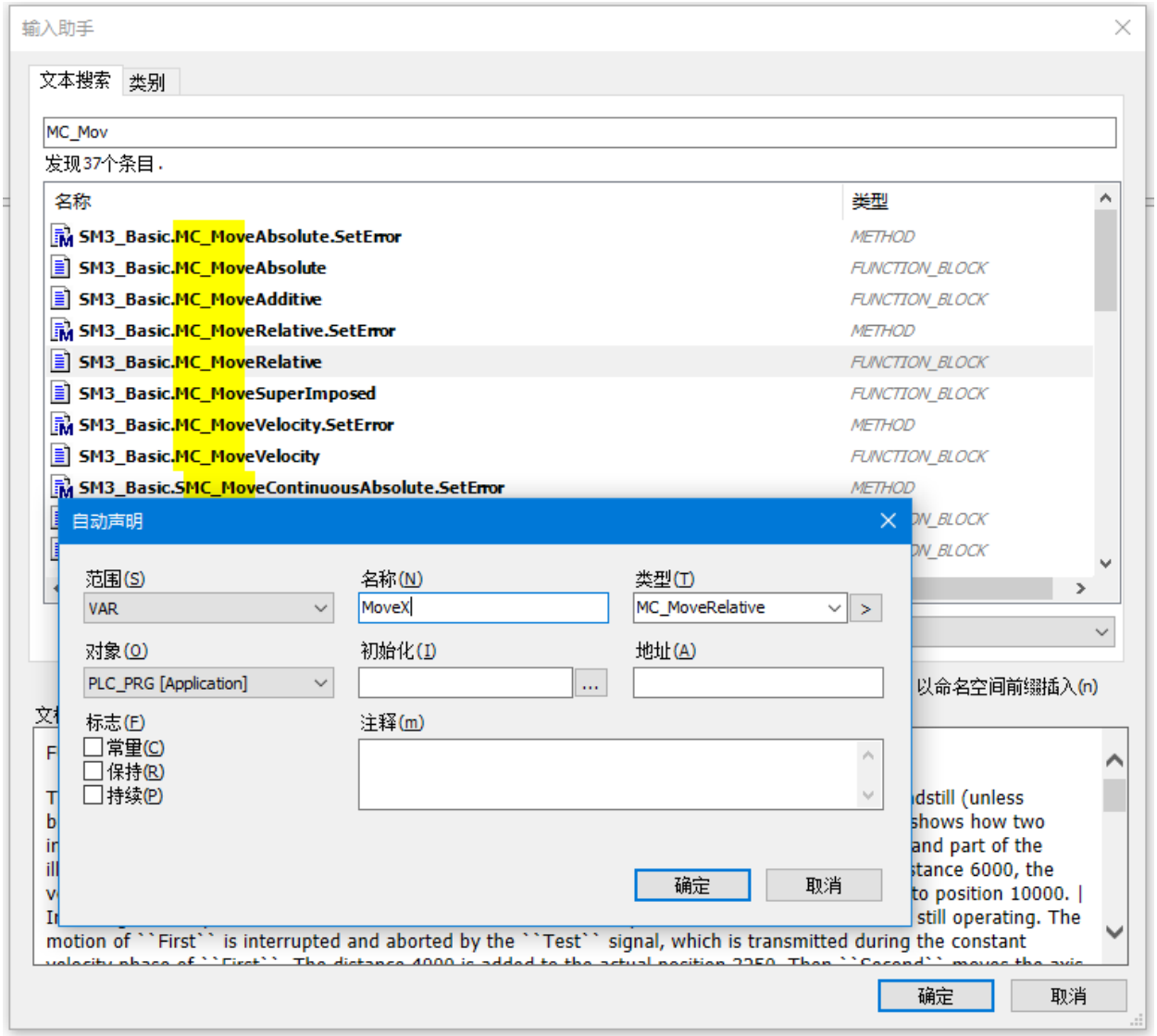

图 5.36　用输入助手输入功能块并命名

```
1   InitXaxis(); // 初始化X轴并使能
2   MoveX(
3       Axis:= ,
4       Execute:= ,
5       Distance:= ,
6       Velocity:= ,
7       Acceleration:= ,
8       Deceleration:= ,
9       Jerk:= ,
10      BufferMode:= ,
11      Done=> ,
12      Busy=> ,
13      Active=> ,
14      CommandAborted=> ,
15      Error=> ,
16      ErrorID=> );
17
```

图 5.37　程序中的功能块代码

设置相对点位运动模块的参数，结果如图 5.38 所示。

图 5.38　相对点位运动模块的参数设置

4.　测试电机做点位运动

首先，点击快捷键编译程序，如图 5.38 所示；然后，登录控制器，选择“登录并下载”，如图 5.39 所示；再运行程序。

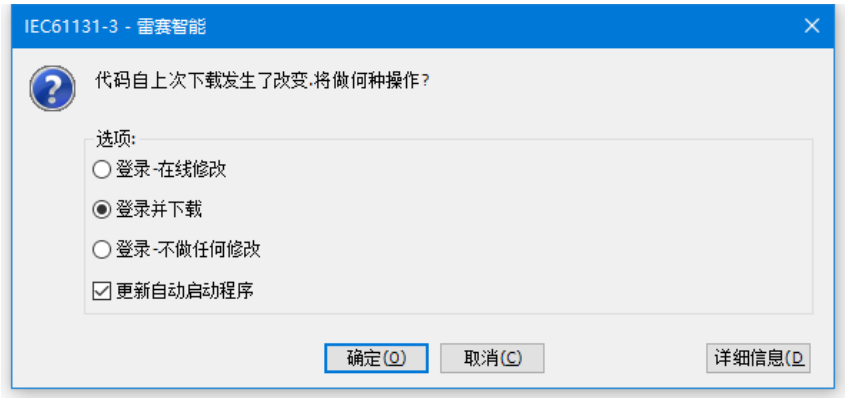

图 5.39　登录并下载程序

鼠标左键双击点位运动功能块 MoveX 中的 Execute 后面的变量，使其准备值为 TRUE，如图 5.40 所示；然后按 Ctrl+F7 键，写入 TRUE，如图 5.41 所示。这时，电机开始转动。

点击项目目录树中的 X（Axis），打开 X 轴的参数显示界面，位移参数显示位移 1600 个脉冲，如图 5.42 所示。

如果这样操作，电机不转动，说明电机接线有错误！

总线电机的测试方法和脉冲式电机的相同。

图 5.40　手动设置 Execute 的输入值

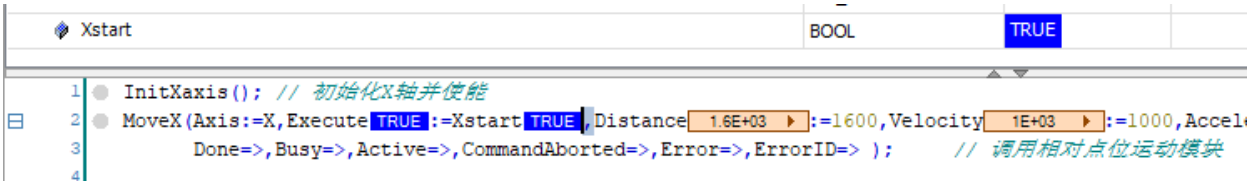

图 5.41　按 Ctrl+F7 键，写入 TRUE

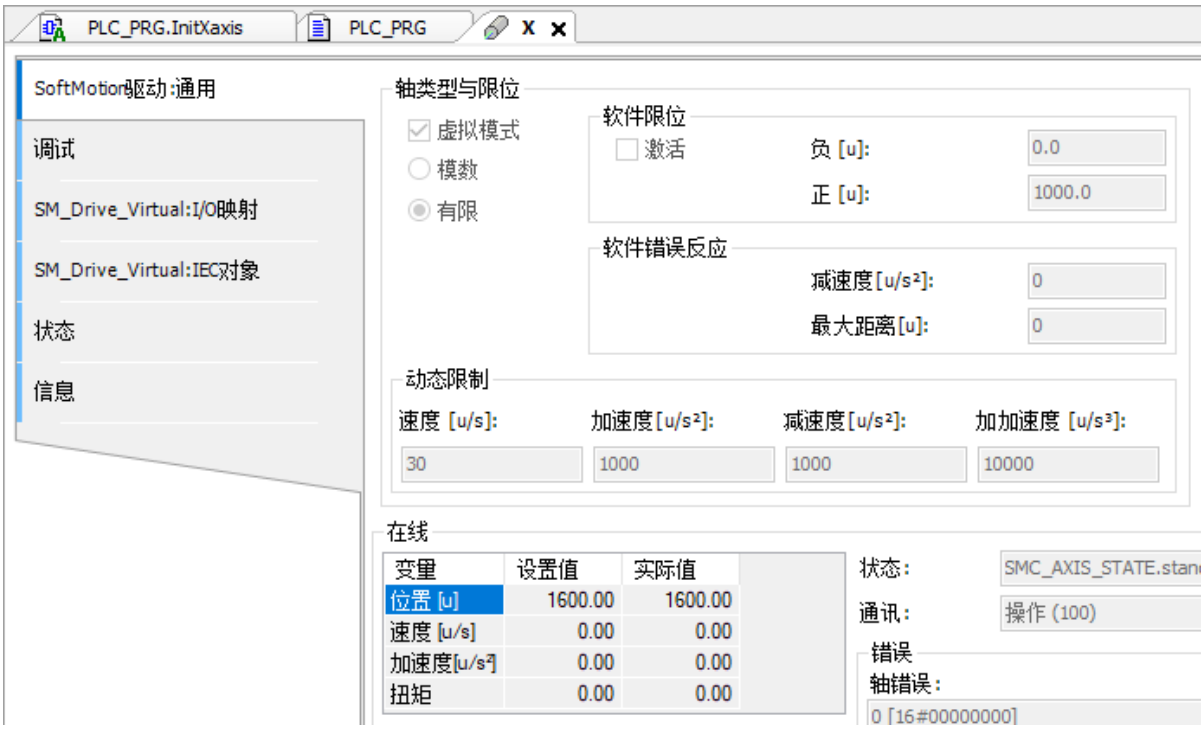

图 5.42　位置参数：位移 1600 个脉冲

5.7 程序设计与运行

硬件测试完成后，可以正式开始编写控制程序。

1. IO 控制

读输入口、写输出口都不需要特殊指令，直接对定义的输入、输出变量读、写即可。代码如下。

```
IF IN5=TRUE THEN
    OUT7:=TRUE;    // 如果输入口开关闭合，输出口 LED 灯亮
  ELSE
    OUT7:=FALSE;   // 如果输入口开关断开，输出口 LED 灯灭
END_IF
```

编译程序后登录控制器，下载程序，运行程序即可。

2. 点位运动和 JOG 运动

暂停程序、退出登录，可继续编写程序。让 X 轴做点位运动，Y 轴做 JOG 运动。Y 轴为总线步进驱动器 DM3E-556 控制的步进电机。

以同样的方法调用 Y 轴的使能指令，如图 5.43 所示。

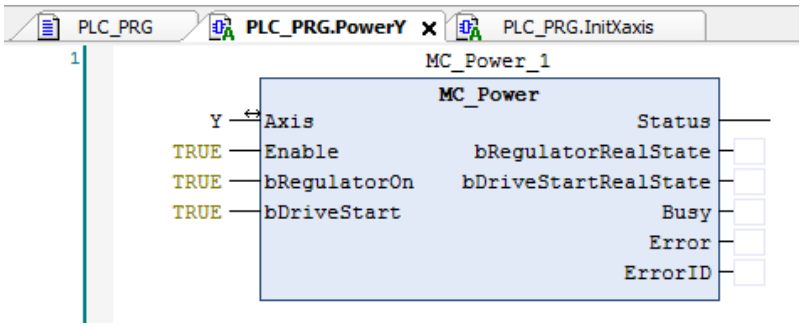

图 5.43　Y 轴使能模块

定义 X 轴和 Y 轴运动的启动信号：Xstart，JogFdY 和 JogBdY。点位运动 MC_MoveRelative 指令之前已经调用；调用 JOG 运动的 MC_Jog 指令的方法也一样。MC_MoveRelative 和 MC_Jog 指令的参数介绍，详见手册 4。

程序代码如下。

```
PROGRAM PLC_PRG

VAR                        // 定义变量
  ......
  In5: BOOL;
  In16: BOOL;
  In17: BOOL;
  MC_Power_0: MC_Power;              // X 轴使能模块
  MC_Power_1: MC_Power;              // Y 轴使能模块
```

```
  MoveX: MC_MoveRelative;          // 命名点位运动模块
  JogY: MC_Jog;                    // 命名 JOG 运动模块
  Xstart:BOOL;          // X 轴点位运动启动信号
  JogFdY: BOOL;          // Y 轴正转启动信号
  JogBdY: BOOL;          // Y 轴反转启动信号
   ......
END_VAR

InitXaxis(); // 初始化 X 轴并使能
PowerY();    // Y 轴使能，总线电机使能只需要调用 MC_Power 指令

IF IN5=TRUE THEN   // 如果按键 5 按下
    Xstart:=TRUE;   // X 轴开始点位运动
    ELSE
    Xstart:=FALSE; // 否则，X 轴不动
END_IF
IF In16=TRUE THEN  // 如果按键 16 按下
   JogFdY:=TRUE;   // Y 轴正转
   ELSE
   JogFdY:=FALSE; // 否则，Y 轴不转
END_IF

IF In17=TRUE THEN  // 如果按键 17 按下
   JogBdY:=TRUE;   // Y 轴反转
ELSE
   JogBdY:=FALSE; // 否则，Y 轴不转
END_IF

MoveX(Axis:=X,Execute:=Xstart,Distance:=1600,Velocity:=1000,Acceleration:=10000,
      Deceleration:=10000,Jerk:=,BufferMode:=,Done=>,Busy=>,Active=>,CommandAborted=>,
      Error=>,ErrorID=> );          // 相对点位运动模块

JogY(Axis:=Y,JogForward:=JogFdY,JogBackward:=JogBdY,Velocity:=5000,
      Acceleration:=10000,Deceleration:=10000,Jerk:= ,Busy=>,
      CommandAborted=>,Error=>,ErrorId=> );     // JOG 运动模块
```

编译程序后登录控制器、下载程序，运行程序，当项目树中的 X、Y 轴图标前都变为绿色时，如图 5.44 所示，可以开始控制电机运动。

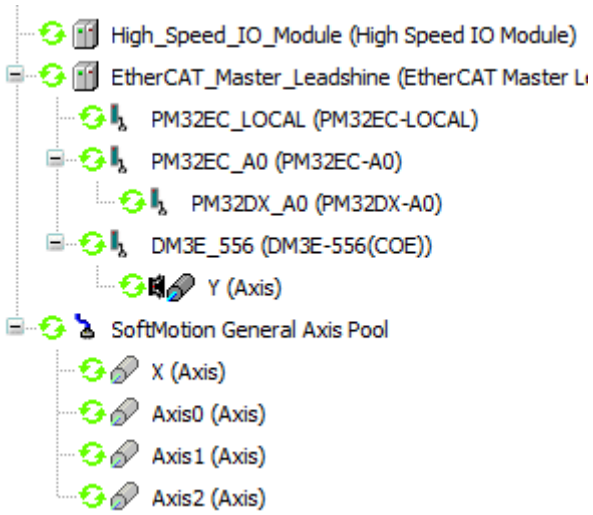

图 5.44　电机已使能

3. 回原点运动

添加 SMC_Home 指令用于脉冲轴 X 回原点，如图 5.45 所示；添加 LS_EtherCATAxis _HomeMove 指令用于总线轴 Y 回原点，如图 5.46 所示；控制回零的代码如下。

回原点指令的参数设置，详见手册 4。

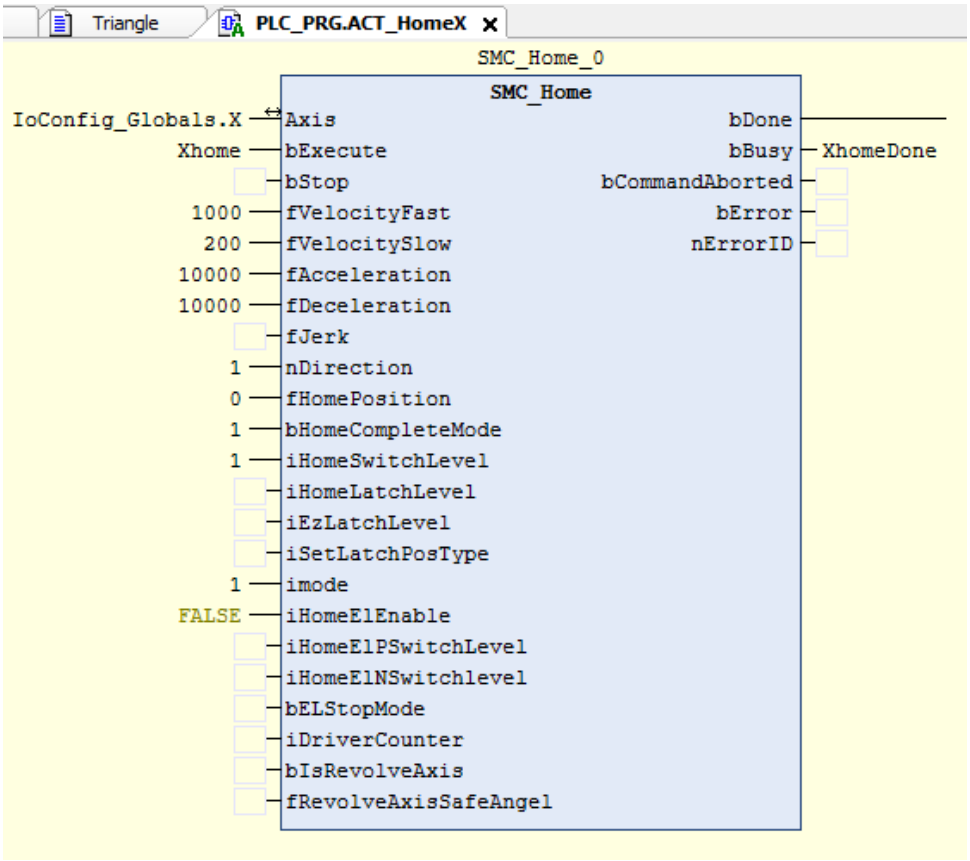

图 5.45　脉冲轴回零指令模块

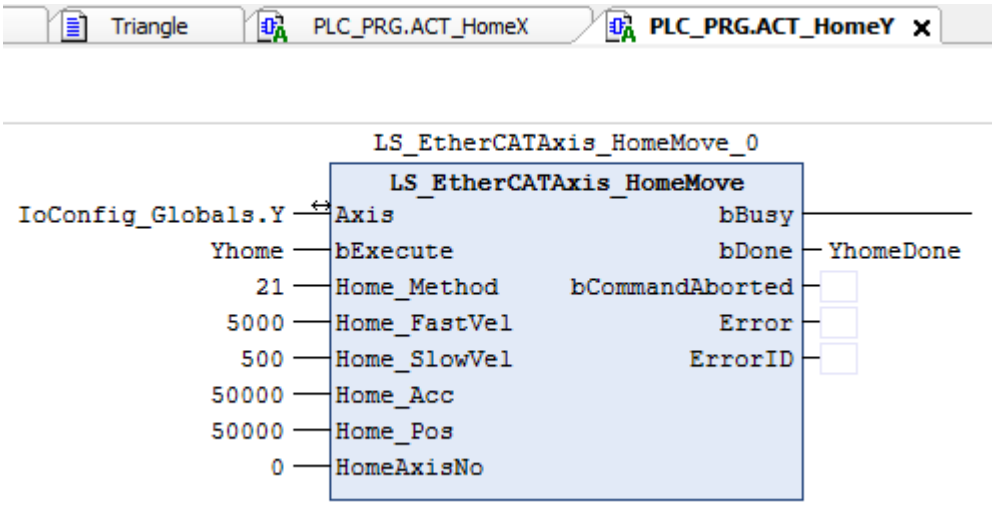

图 5.46　总线轴回零指令模块

```
PROGRAM PLC_PRG

VAR                          // 定义变量
   ......
  SMC_Home_0: SMC_Home;                                    // 命名脉冲电机 Home 模块
  LS_EtherCATAxis_HomeMove_0: LS_EtherCATAxis_HomeMove;    // 命名总线电机 Home 模块
  Xhome:BOOL;                // X 轴 Home 启动
  XhomeDone:BOOL;            // X 轴 Home 结束
  Yhome:BOOL;                // Y 轴 Home 启动
  YhomeDone:BOOL;            // Y 轴 Home 结束
   ......
END_VAR


      IF IN10=TRUE THEN
          Xhome:=TRUE;         // 开关 IN10 闭合，X 轴回原点
      END_IF
      IF XhomeDone=TRUE THEN
          Xhome:=FALSE;        // X 轴回原点结束
      END_IF

      IF IN11=TRUE THEN
          Yhome:=TRUE;         // 开关 IN11 闭合，Y 轴回原点
      END_IF
      IF YhomeDone=TRUE THEN   // Y 轴回原点结束
      Yhome:=FALSE;
      END_IF

      ACT_HomeX();      // 调用 X 轴回原点运动模块
      ACT_HomeY();      // 调用 Y 轴回原点运动模块
```

4. 分段插补运动

用直线插补、圆弧插补指令控制 X、Y 轴完成如图 5.47 的轨迹运动：起点在原点，第一段轨迹的终点坐标为 5000，1000；第二段轨迹的坐标为 10000，0；第三段轨迹为圆弧，半径为 10000，终点在原地。

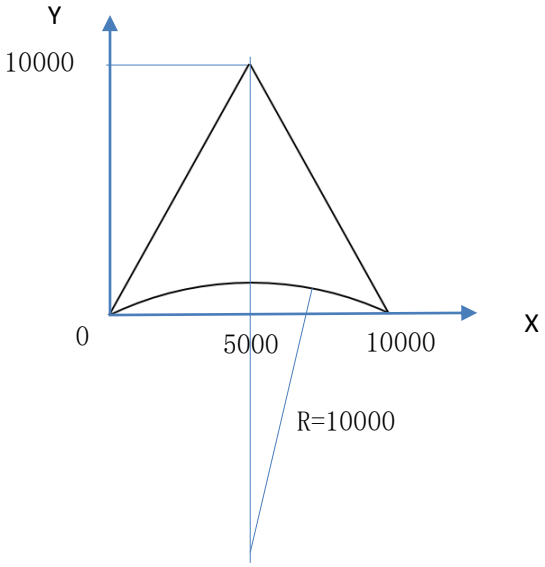

图 5.47　XY 轴的运动轨迹

编写子程序 Triangle，实现分段插补运动的控制过程。步骤如下：

1）全局变量的定义

鼠标右键点击项目树中的“Application”，选择“全局变量列表”，如图 5.48 所示。给“全局变量列表”一个名称：GVL。

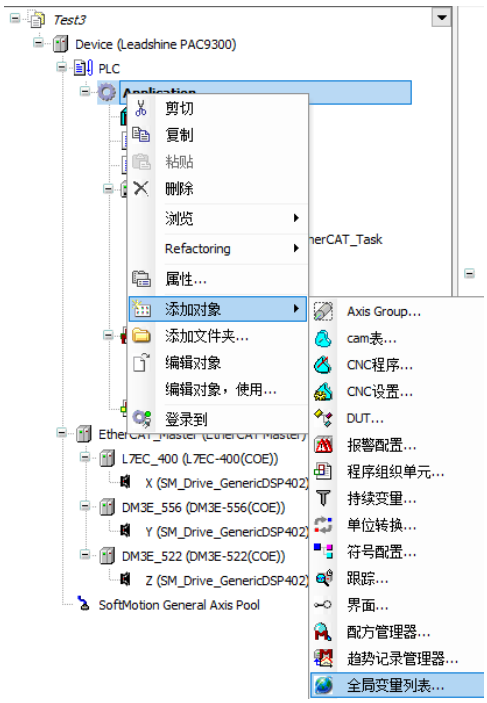

图 5.48　添加全局变量

然后，在项目树中鼠标双击“GVL”打开全局变量列表的编辑界面。
全局变量既可以在主程序中使用，也可以在子程序中使用。

```
VAR_GLOBAL
    TriangleEnable: BOOL;    // 子程序 Triangle 运行标志
    CamEnable: BOOL;
    TriangleState : INT;    // 子程序 Triangle 运行状态
    CamState : INT;
END_VAR
```

注意：全局变量 VAR_GLOBAL 前的{attribute 'qualified_only'}要删除！

2) 子程序的定义
鼠标右键点击项目树中的“Application”，选择“POU”，如图 5.49 所示。

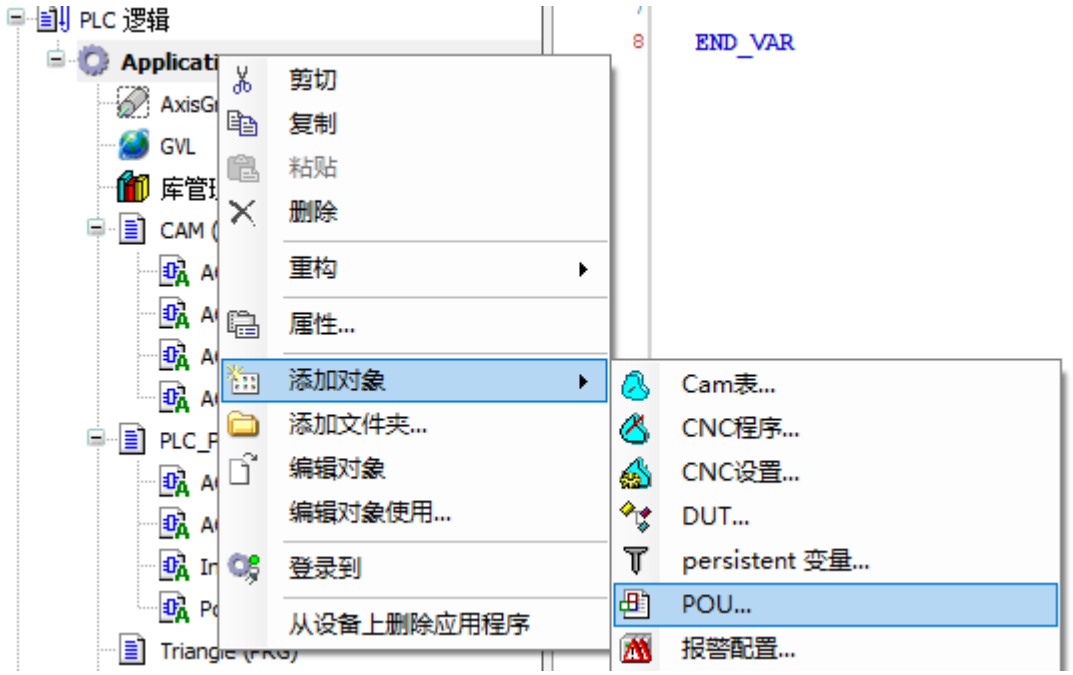

图 5.49　添加子程序

在弹出的对话框中选择“程序”、填写子程序的名称：Triangle，选子程序的语言为“结构化文本”，然后点“打开”。在项目树中鼠标双击“Triangle”，可打开子程序的编辑界面。子程序代码如下。

```
PROGRAM Triangle
VAR                                          // 子程序的变量定义
  LineInterpolate: LS_2AxisLineRel;          // 命名相对坐标的直线插补运动模块
  CircleInterpolate:LS_2AxisCircleRel;       // 命名相对坐标的圆弧插补运动模块
  LineExe: BOOL;                             // 启动直线插补
  CircleExe: BOOL;                           // 启动圆弧插补
```

```
  MyEndPos: ARRAY [0..1] OF LREAL;       // 终点坐标
  MyR: REAL;                             // 圆弧半径
  MyDir: INT;                            // 圆弧插补运动方向
END_VAR


CASE TriangleState OF
1:                                       // 第 1 条轨迹：直线插补
  MyEndPos[0]:=5000;
  MyEndPos[1]:=10000;                    // 终点坐标为(5000，10000)
  LineExe:=TRUE;                         // 启动直线插补
  TriangleState:=2;                      // 进入状态 2

2:                                       // 等待直线插补结束
  IF LineInterpolate.Done THEN
      LineExe:=FALSE;                    // 关闭直线插补
      TriangleState:=3;
  END_IF

3:                                       // 第 2 条轨迹：直线插补
  MyEndPos[0]:=5000;
  MyEndPos[1]:=-10000;                   // 终点坐标为(5000，-10000)，是相对坐标！
  LineExe:=TRUE;                         // 启动直线插补
  TriangleState:=4;

4:                                       // 等待直线插补结束
  IF LineInterpolate.Done THEN
      LineExe:=FALSE;
      TriangleState:=5;
  END_IF

5:                                       //第 3 条轨迹：圆弧插补
  MyEndPos[0]:=-10000;
  MyEndPos[1]:=0;                        // 终点坐标为(-10000，0)，是相对坐标！
  MyR :=-10000;                          // 半径=10000，圆弧角小于 180 度，故为负
  MyDir := 0;                            // 圆弧插补方向为逆时针，故为 0
  CircleExe:=TRUE;                       // 启动圆弧插补
  TriangleState:=6;

6:                                       // 等待圆弧插补结束
    IF CircleInterpolate.Done THEN
      CircleExe:=FALSE;
      TriangleState:=7;
  END_IF

7:                                       // 子程序结束
  TriangleEnable := FALSE;

END_CASE
```

```
// 相对坐标的直线插补指令
LineInterpolate(Axis0:=X, Axis1:=Y, Exec:=LineExe, Halt:=, Stop:=, IpoCycle:=2000,
                EndPos:=MyEndPos, VelocityMode:=1, Velocity:=3000,
                Acceleration:=10000, Deceleration:=10000, Jerk:= , Vel=>, Done=>,
                Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=> );
// 相对坐标的圆弧插补指令
CircleInterpolate(Axis0:=X,   Axis1:=Y, Exec:=CircleExe, Halt:=, Stop:=,
                  IpoCycle:=2000, ArcIpo_dir:=MyDir, Circle_Mode:=2,
                  StopAngle:=, CircleRadiual:=MyR, AuxPos:=, EndPos:=MyEndPos,
                  RunCycle:=1, VelocityMode:=1, Velocity:=3000,
                  Acceleration:=10000, Deceleration:=10000, Jerk:=, Vel=>, Done=>,
                  Busy=>, CommandAborted=>, CommandHalt=>, Error=>, ErrorID=> );
```

注意：执行一条插补指令后，要等到该指令结束后，才能执行下一条插补指令。

为了调试方便，在调用子程序 Triangle 之前，主程序将 X、Y 轴的当前位置清零，使用的是设置坐标指令 MC_SetPosition。为了显示插补运动过程中的 X、Y 轴的速度和 XY 轴的合速度，使用了读轴速度指令 MC_ReadActualVelocity。由于速度值有波动，为了更好地显示速度曲线，采用数字低通滤波算法对合速度进行了处理，使速度曲线较平滑。主程序的相关代码如下：

```
PROGRAM PLC_PRG

VAR                                    // 定义变量
   ......
In12: BOOL;                            // 子程序 Triangle 的启动开关
runFlag: BOOL;
SetPosX: MC_SetPosition ;              // 命名设置坐标模块
SetPosY: MC_SetPosition ;
GetSpeedX: MC_ReadActualVelocity;      // 命名读轴速度模块
GetSpeedY: MC_ReadActualVelocity;
realSpeedX: LREAL ;                    // X 轴的速度
realSpeedY: LREAL ;                    // Y 轴的速度
absSpeed0: LREAL;
absSpeed1: LREAL;
absSpeed: LREAL;                       // XY 轴的合速度
   ......
END_VAR

......
IF In12 = TRUE THEN // 如果开关 IN12 闭合，准备开始运行子程序 Triangle
   runFlag := TRUE;  // 启动设置坐标模块
   SetPosX(Axis:=X,Execute:=runFlag,Position:=0,Mode:=FALSE,
            Done=>,Busy=>,Error=>,ErrorID=>);                  // X 轴坐标清零
   SetPosY(Axis:=Y,Execute:=runFlag,Position:=0,Mode:=FALSE,
            Done=>,Busy=>,Error=>,ErrorID=>);                  // Y 轴坐标清零
   runFlag := FALSE;              // 关闭设置坐标模块
```

```
    TriangleEnable := TRUE;      // 可以调用子程序 Triangle
    TriangleState := 1;          // 子程序 Triangle 的状态值为 1
END_IF;

IF TriangleEnable = TRUE THEN
    Triangle();                  // 调用子程序 Triangle
END_IF;
……
GetSpeedX(Axis:=X,Enable:=TRUE,Valid=>,Busy=>,Error=>,ErrorID=>,
          Velocity=>realSpeedX);                                  // 读 X 轴的速度
GetSpeedY(Axis:=Y,Enable:=TRUE,Valid=>,Busy=>,Error=>,ErrorID=>,
          Velocity=>realSpeedY);                                  // 读 Y 轴的速度
absSpeed1:=SQRT(realSpeedX*realSpeedX+realSpeedY*realSpeedY);     // 计算 XY 轴的合速度
absSpeed:= 0.98*absSpeed0+0.02*absSpeed1;                         // 对合速度低通滤波
absSpeed0:=absSpeed;
```

注意：执行插补运动之前，要修改各轴的动态限制：4 个参数都要改为一个大数值，如图 5.50 所示。

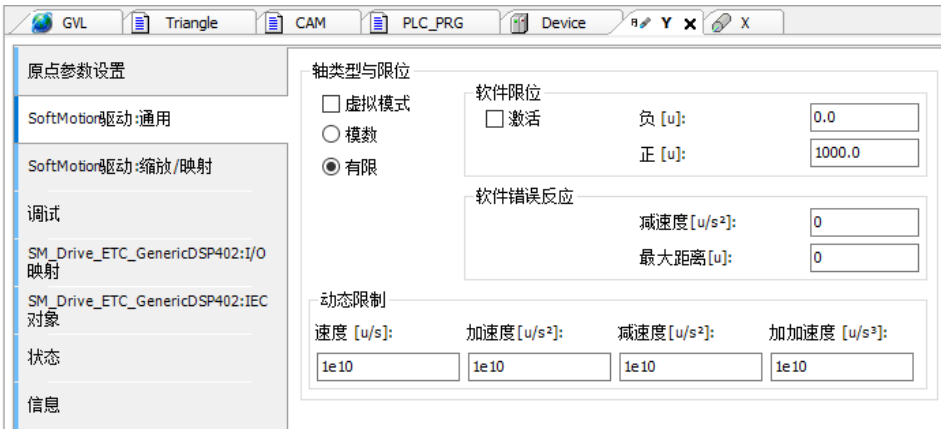

图 5.50　修改轴的动态限制参数

3）变量的跟踪曲线

在程序调试过程中，雷赛 iStudio 软件可显示变量变化的跟踪曲线。使用方法如下。

- 鼠标右键点击项目树中的“Application”，选择“跟踪”，如图 5.51 所示。
- 在弹出的对话框中填写一个跟踪曲线的名称，如图 5.52 所示的 Trace。
- 在项目树中用鼠标双击跟踪曲线名，打开跟踪曲线的界面。
- 在跟踪曲线界面的右上角，点击“配置”，弹出对话框如图 5.53 所示。将其中的“任务”选“Main Task”。
- 再点击界面右上角的“添加变量”，弹出如图 5.54 所示的对话框。在“跟踪变量”中选“IOConfig_Globals”下的“X”，找到 X 轴的位置变量 fActPosion，点“确定”。再添加 X 轴的速度变量 fActVelocity。

  然后在“Y”中选其位置和速度变量。

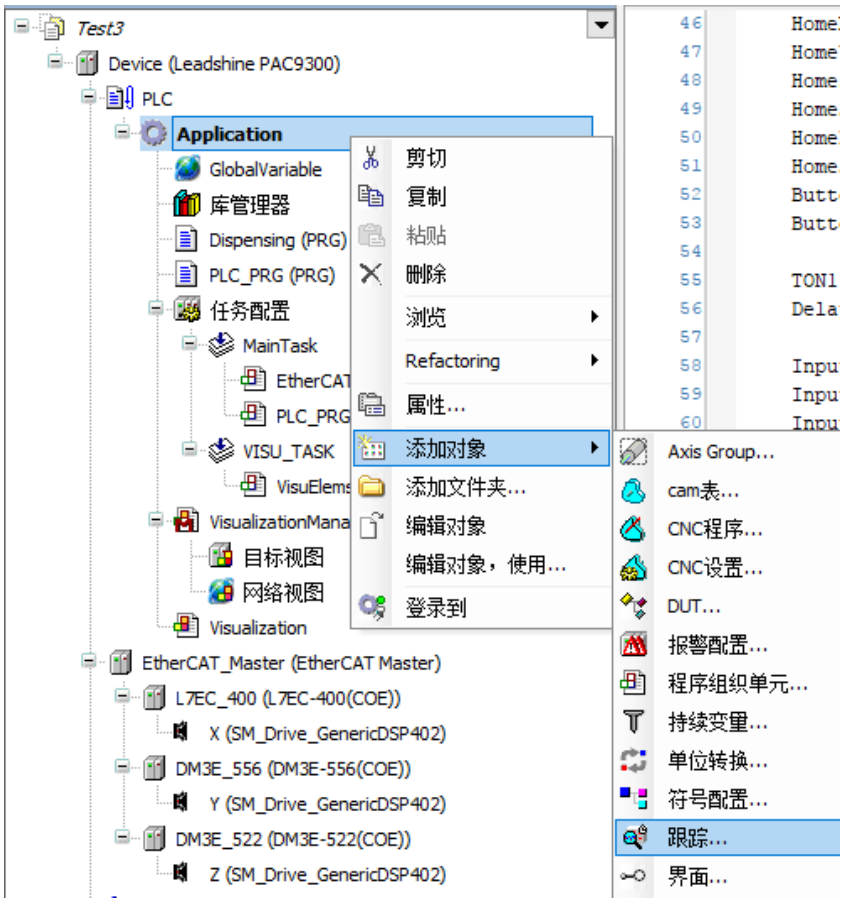

图 5.51　添加跟踪曲线

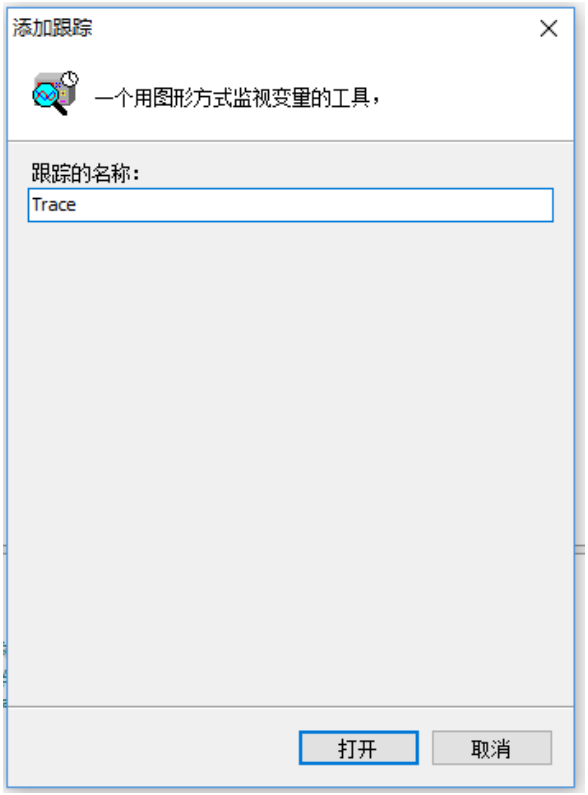

图 5.52　跟踪曲线的名称

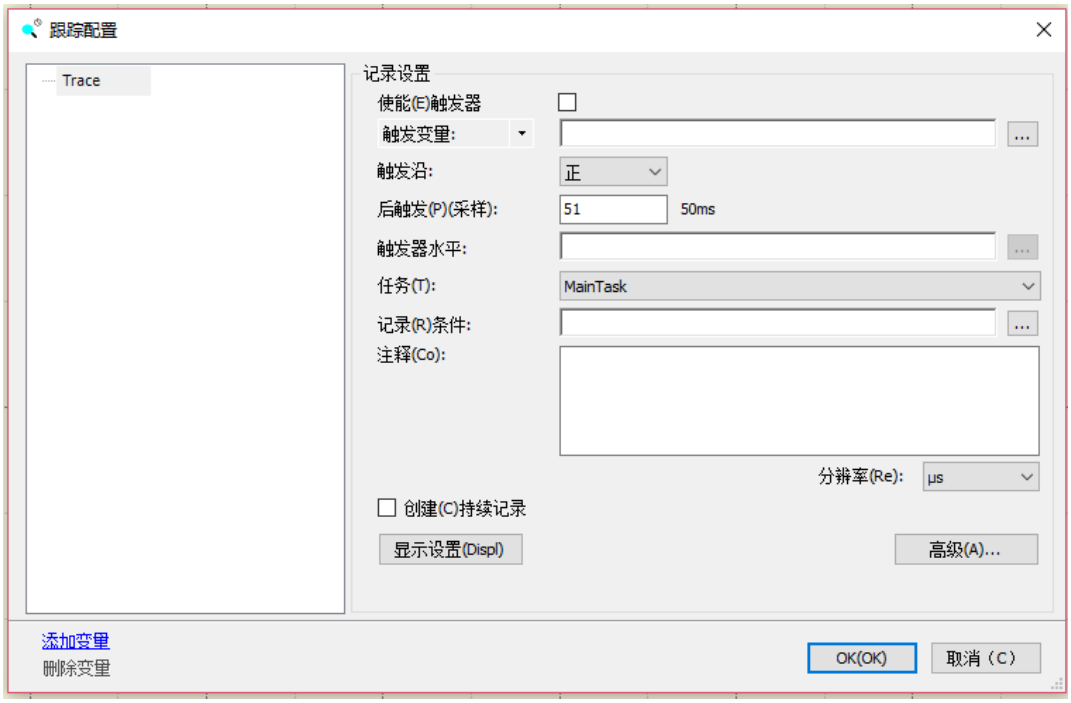

图 5.53　跟踪配置对话框

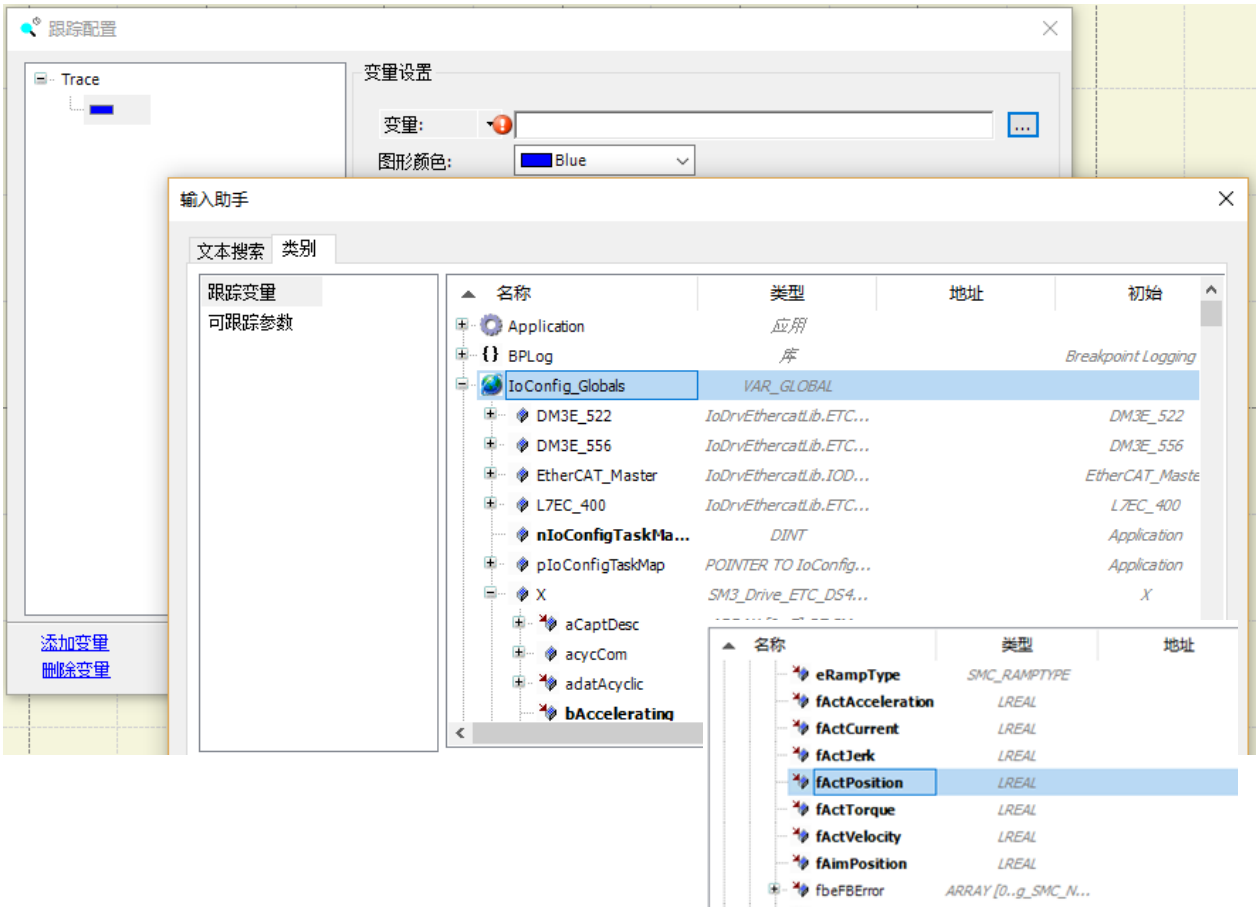

图 5.54　添加变量

- 再添加 XY 轴的合速度 absSpeed。该变量在“Application”下的“PLC_PRG”中。
- 在“跟踪配置”界面中左下方，点时间轴，添加图表；再添加现有变量。X、Y 轴的位移变量各占一个图表；三个速度变量占用一个图表。如图 5.55 所示。
- 运行程序后，在变量跟踪界面上点击鼠标右键，选“下载跟踪”，如图 5.56 所示，即可开始记录所添加变量的变化曲线。

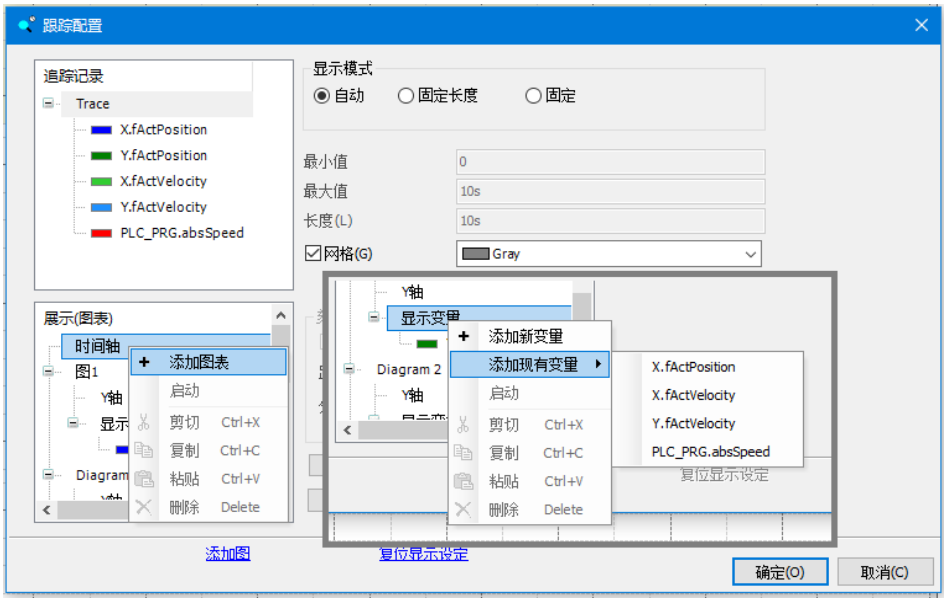

图 5.55　设置变量的图表

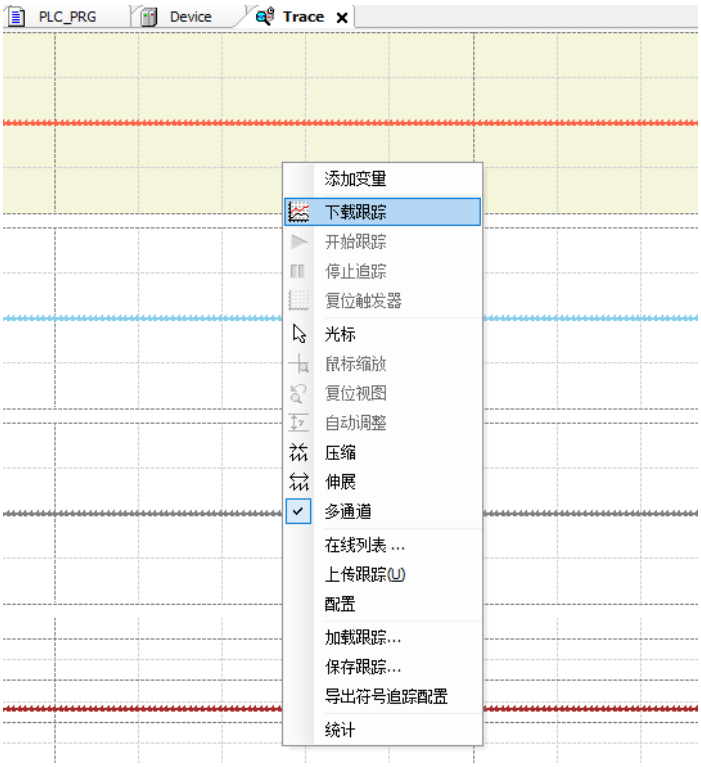

图 5.56　下载跟踪曲线

- 要暂停曲线跟踪，在界面上点击右键，选“停止跟踪”。
- 停止跟踪后，转动鼠标滚轮可放大、缩小曲线，按住鼠标左键可拖拽曲线。

4）程序运行结果

从图 5.57 的插补运动参数曲线可看出：X、Y 轴的每段插补运动都有加速过程和减速过程；因此，合速度每段插补运动也都有加速和减速过程。

这对于图 5.47 所示的轨迹而言是合理的。因为该轨迹的 2 个转弯处转角较小，X 和 Y 轴都有换向过程，如果没有加、减速过程，运动轨迹很难保证位置精度。

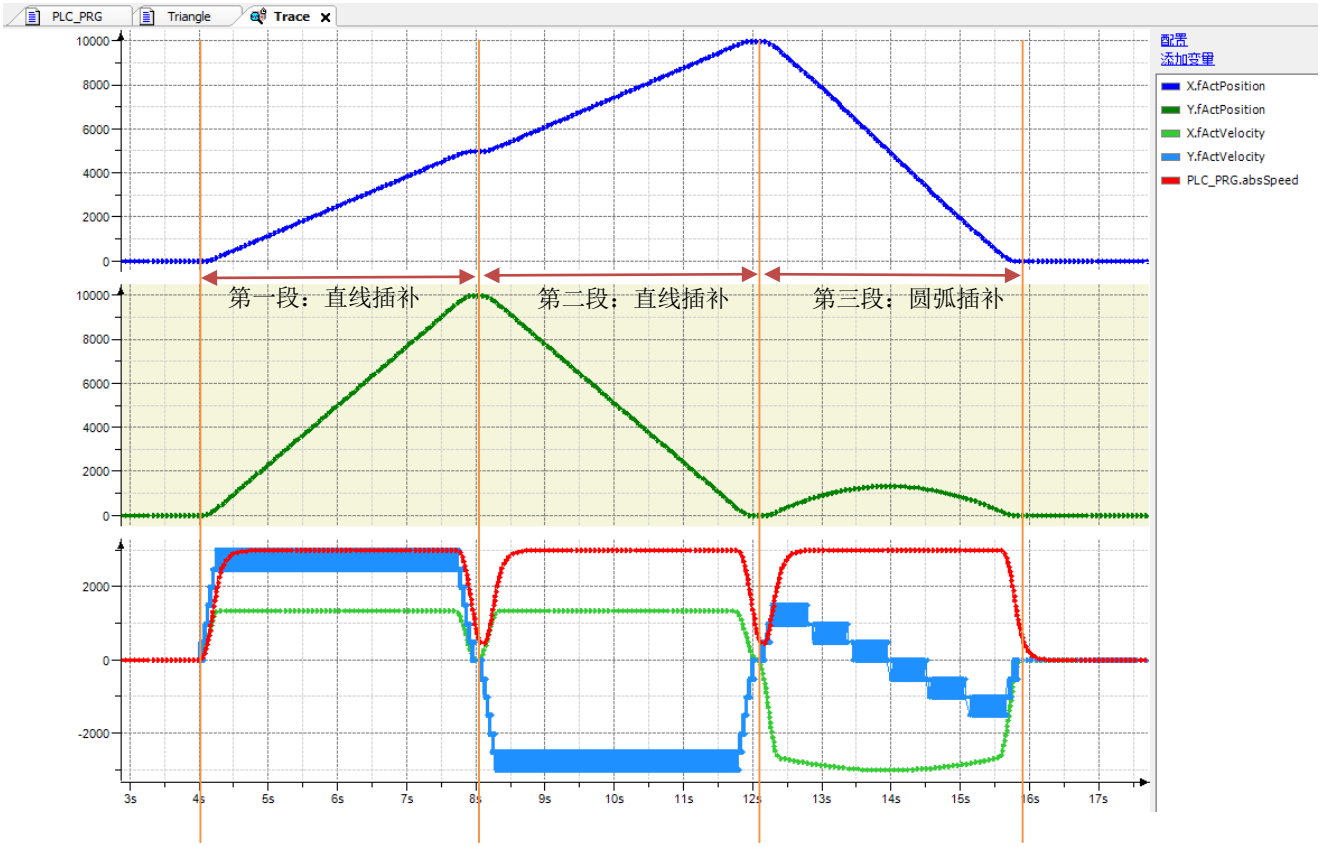

图 5.57　分段插补运动的位置、速度曲线

5. 连续插补运动

轨迹如图 5.58 所示：从原点快速移动到点 P0（200，800）；直线插补到点 P1（400，1300）；圆弧插补经过点 P2a（750，1500）到点 P2（1100，1300）；直线插补到点 P3（1300，800）；圆弧插补经过点 P4a（750，0）回到点 P0。

PMC600 运动控制器有 2 种实现连续插补的指令：轴组连续插补指令和雷赛连续插补指令。

轴组控制功能十分强大，不但可以控制直角坐标系的运动，还可以控制各种机器人的运动，详见手册 4 的介绍。

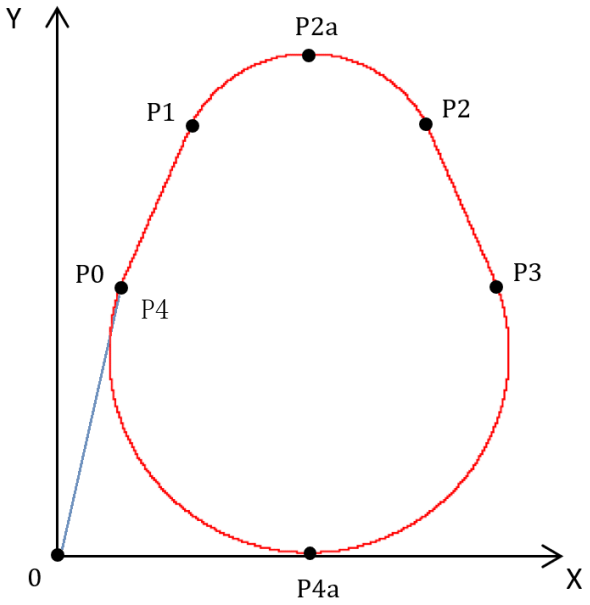

图 5.58　凸轮轨迹

编写子程序 CAM，实现轴组连续插补功能。编程步骤如下：

第一步：建立连续插补轴组，配置运动学模型

首先在工程中，右键单击工程“设备窗口”中的“Application”，选择“添加对象”，然后点击添加“轴组”，并将轴组命名为 AxisGroup。通过“运动学模型”界面，选择“TRAFO.Kin_Gantry3”运动学模型（三维直角坐标系），并选择轴组的轴号为 Axis0、Axis1、Axis2，将轴组规划任务设置为“Softmotion_PlanningTask”，如图 5.59 所示。

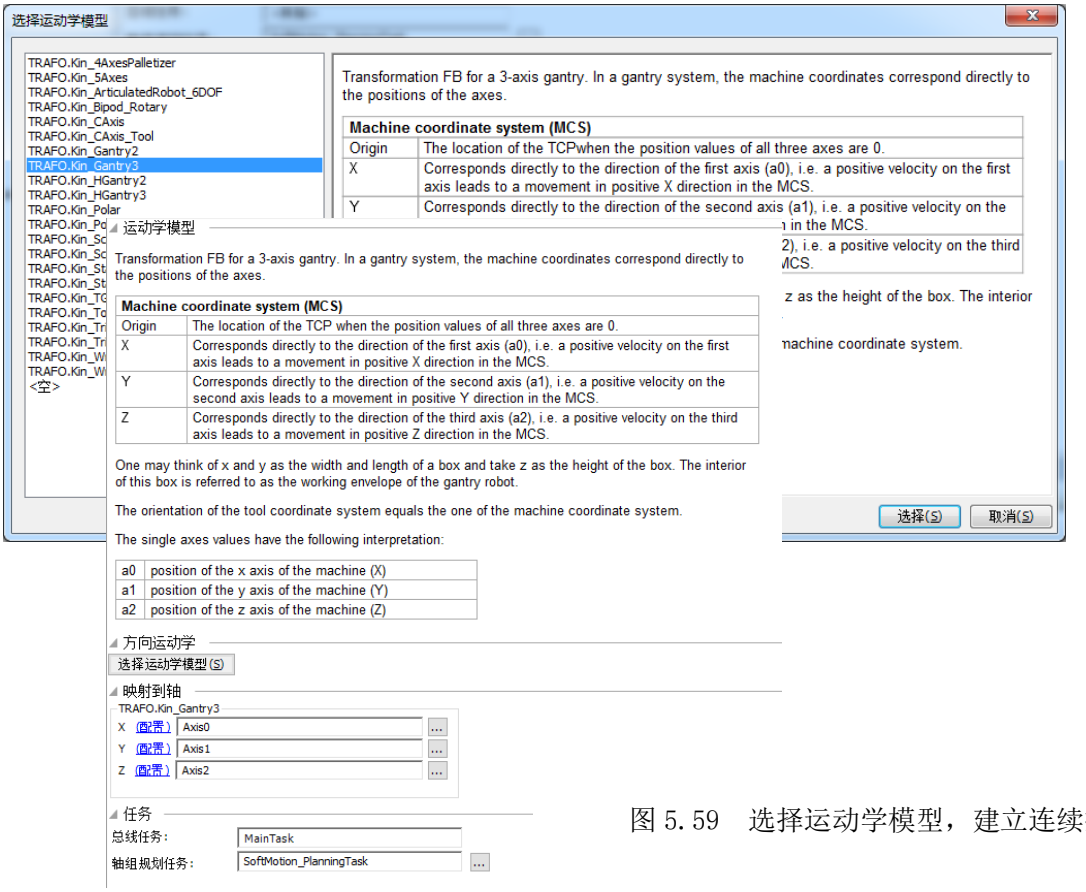

图 5.59　选择运动学模型，建立连续插补轴组

第二步：初始化轴组

轴组初始化包括三个部分：轴组使能；坐标系转换；轴组启用。

在子程序 CAM 中分别调用 SMC_GroupPower、和 MC_SetCoordinateTransform 三个模块，如图 5.60~5.62 所示。

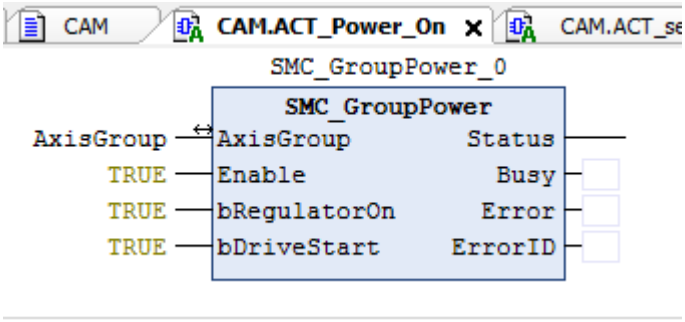

图 5.60　轴组使能模块

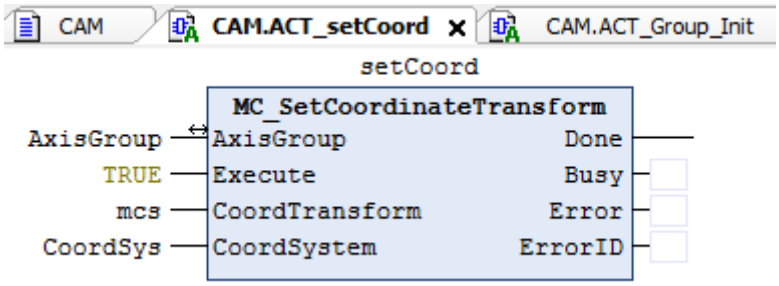

图 5.61　坐标系转换模块

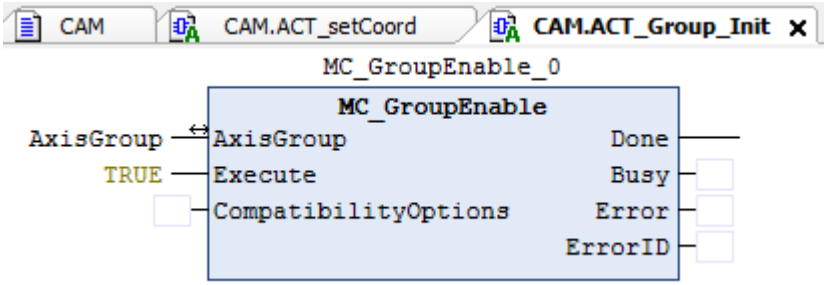

图 5.62　轴组启用模块

第三步：调用插补指令

根据图 5.58 所示的运动轨迹，在子程序 CAM 中调用轴组的直线插补、圆弧插补指令，如图 5.63 所示。各指令的参数定义详见手册 4。

注意：和分段插补指令不同的是：从第二条插补指令开始，启动信号不是上一条插补指令的完成信号 Done，而是指令已接收信号 CommandAccepted。

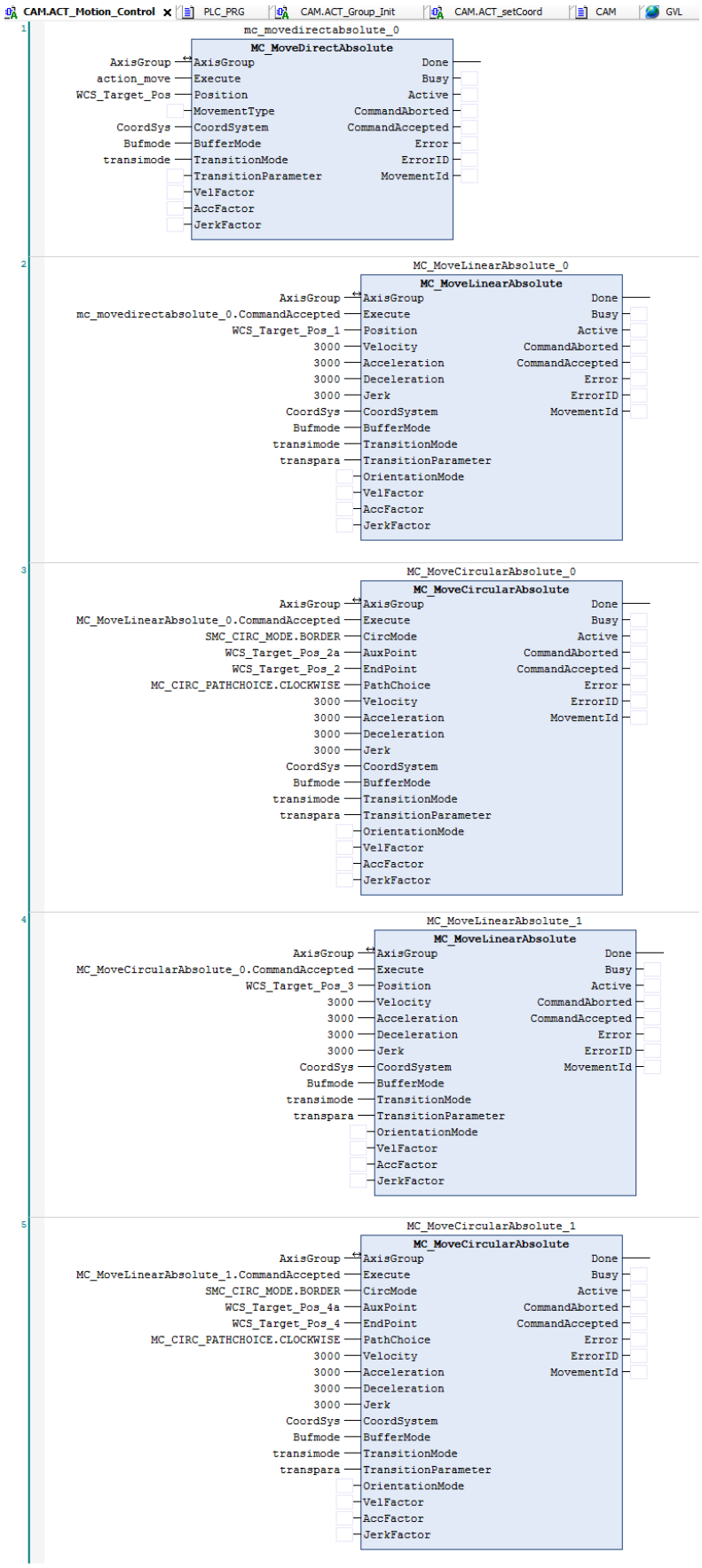

图 5.63　轴组的插补指令

第四步：设置插补参数

根据图 5.58 所示的运动轨迹，在子程序 CAM 的变量定义中定义笛卡尔坐标系（直角坐标）直线插补的终点位置，圆弧插补的终点位置和中间点位置。然后再生成工件坐标系的位置。

轴组连续插补的速度交接模式采用 mcBlendingLow 模式（低速滤波处理）；轴组连续插补采用基于速度参数的拐角模式，拐角过渡因子 TransitionParameter[0]设为 1。

子程序 CAM 的代码如下：

```
PROGRAM CAM
VAR
  //  定义笛卡尔坐标位置
  org_pos : mc_coord_ref := (x:=200, y:= 800, z:= 100);
  firt_pos : mc_coord_ref := (x:=400 , y:= 1300 , z:= 100);
  second_pos : mc_coord_ref := (x:=1100, y:= 1300, z:= 100);
  second_pos2 : mc_coord_ref := (x:=750, y:= 1500, z:= 100);
  third_pos : mc_coord_ref := (x:=1300, y:= 800, z:=100);
  forth_pos : mc_coord_ref := (x:=200, y:= 800, z:=100);
  forth_pos2 : mc_coord_ref := (x:=750, y:= 0, z:=100);

  //  生成工件坐标的位置
  WCS_Target_Pos: SMC_POS_REF := (c := org_pos);
  WCS_Target_Pos_1: SMC_POS_REF := (c := firt_pos);
  WCS_Target_Pos_2: SMC_POS_REF := (c := second_pos);
  WCS_Target_Pos_2a: SMC_POS_REF := (c := second_pos2);
  WCS_Target_Pos_3: SMC_POS_REF := (c := third_pos);
  WCS_Target_Pos_4: SMC_POS_REF := (c := forth_pos);
  WCS_Target_Pos_4a: SMC_POS_REF := (c := forth_pos2);

  SMC_GroupPower_0: SMC_GroupPower;       //命名轴组使能模块
  setCoord : MC_SetCoordinateTransform;  //命名坐标系转换模块
  MC_GroupEnable_0:MC_GroupEnable;        //命名轴组启用模块

  mcs : MC_COORD_REF;                                    //定义坐标系
  CoordSys: SMC_COORD_SYSTEM := SMC_COORD_SYSTEM.MCS;
  Bufmode: MC_BUFFER_MODE := MC_BUFFER_MODE.BlendingLow;        //轴组运动速度交接模式
  transimode: MC_TRANSITION_MODE := MC_TRANSITION_MODE.TMStartVelocity; //拐角过渡模式
  transpara: ARRAY[0..(SMC_RCNST.MAX_TRANS_PARAMS-1)] OF LREAL :=[1.0]; //拐角过渡因子

  mc_movedirectabsolute_0: mc_movedirectabsolute;       //命名快速移动模块
  MC_MoveLinearAbsolute_0: MC_MoveLinearAbsolute;       //命名绝对坐标直线插补模块
  MC_MoveLinearAbsolute_1: MC_MoveLinearAbsolute;
  MC_MoveCircularAbsolute_0: MC_MoveCircularAbsolute; //命名绝对坐标圆弧插补模块
  MC_MoveCircularAbsolute_1: MC_MoveCircularAbsolute;
  action_move: BOOL;

END_VAR
```

```
CASE CamState OF
    1:                                      //轴组使能
        act_power_on();                     //调用轴组使能模块
        IF SMC_GroupPower_0.Status THEN
            CamState:=2;                    //轴组使能完成，进入下一状态
        END_IF

    2:                                       //坐标变换
        setCoord(AxisGroup := AxisGroup);   //调用坐标系转换模块
        CamState :=3;                        //进入下一个状态

    3:                                      //轴组启用
        ACT_Group_Init();
        IF MC_GroupEnable_0.Done THEN
            CamState := 4;                 //轴组启动完成，进入下一状态
        END_IF

    4:                                      // 连续插补
       action_move := TRUE;
       ACT_Motion_Control();                      //开始连续插补
        IF MC_MoveCircularAbsolute_1.Done THEN
            CamState:=5;                        //连续插补结束，进入下一状态
        END_IF
    5:                                      // 子程序结束
        CamEnable := FALSE;
END_CASE
```

相关主程序代码如下：

```
……
IF In12 = TRUE OR In13 = TRUE THEN    // 如果开关 IN12 或 IN13 闭合
   runFlag := TRUE;
   SetPosX(Axis:=X,Execute:=runFlag,Position:=0,Mode:=FALSE,
            Done=>,Busy=>,Error=>,ErrorID=>);
   SetPosY(Axis:=Y,Execute:=runFlag,Position:=0,Mode:=FALSE,
            Done=>,Busy=>,Error=>,ErrorID=>);                  // 位置坐标清零
   runFlag := FALSE;
   IF In12 = TRUE THEN          // 如果开关 IN12 闭合
      TriangleEnable := TRUE;  // 开始子程序 Triangle
      TriangleState := 1;
   END_IF;
   IF In13 = TRUE THEN          // 如果开关 IN13 闭合
      CamEnable := TRUE;        // 开始子程序 CAM
      CamState := 1;
   END_IF;
END_IF;

IF CamEnable = TRUE THEN
   CAM();                        // 调用子程序 CAM
END_IF;
```

```
GetSpeedX(Axis:=Axis0,Enable:=TRUE,Valid=>,Busy=>,Error=>,ErrorID=>,
          Velocity=>realSpeedX);
GetSpeedY(Axis:=Axis1,Enable:=TRUE,Valid=>,Busy=>,Error=>,ErrorID=>,
          Velocity=>realSpeedY);                                    // 读轴的速度
absSpeed1=SQRT(realSpeedX*realSpeedX+realSpeedY*realSpeedY);        // XY 轴的合速度
……
```

第四步：编译程序、下载执行

连续插补的结果如图 5.64 所示。

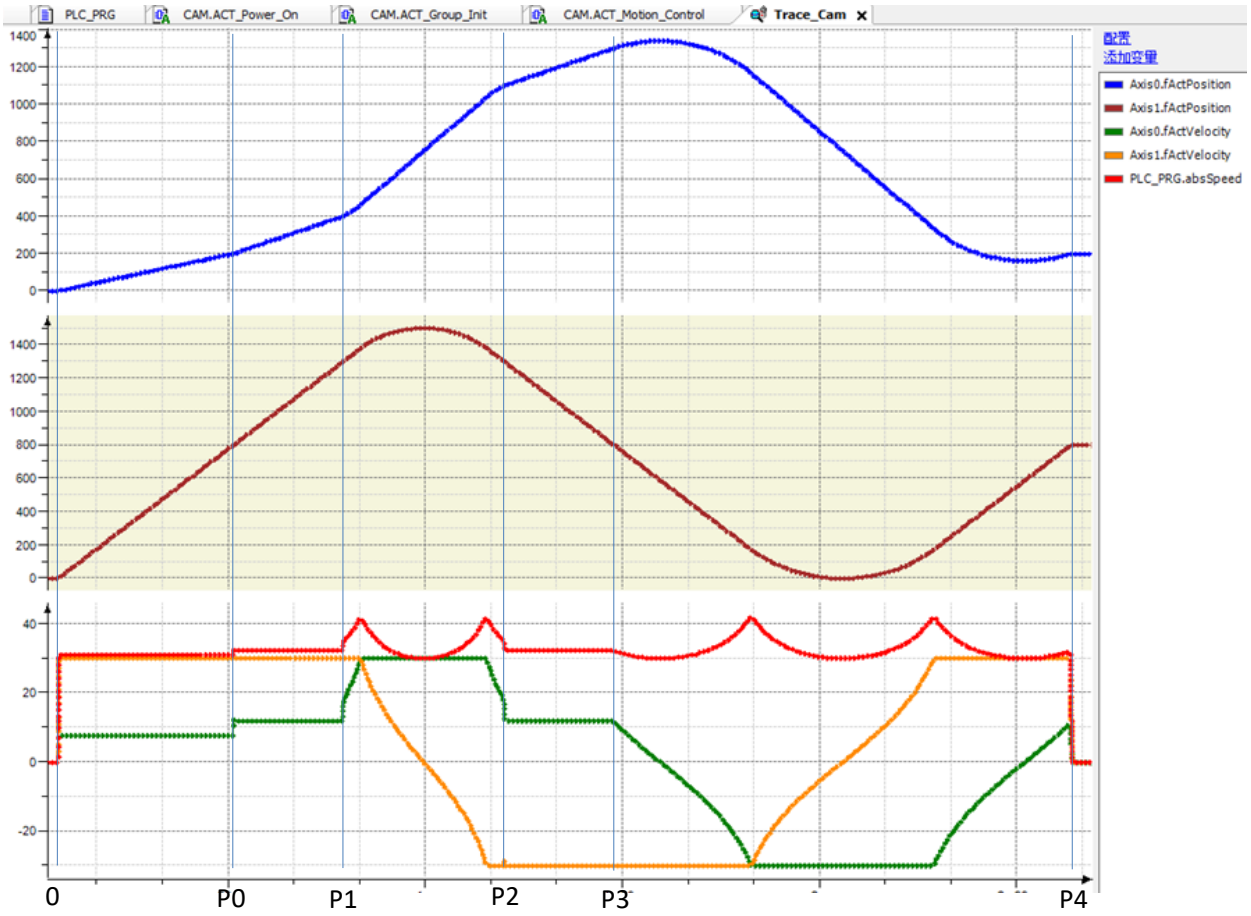

图 5.64　连续插补各参数曲线

从图 5.64 可看出，轴 Axis0、Axis1 的速度曲线在各线段的衔接处没有先减速、后加速的过程；插补运动的合速度基本恒定，即插补速度是连续的。

本例程完整的项目树如图 5.65 所示。

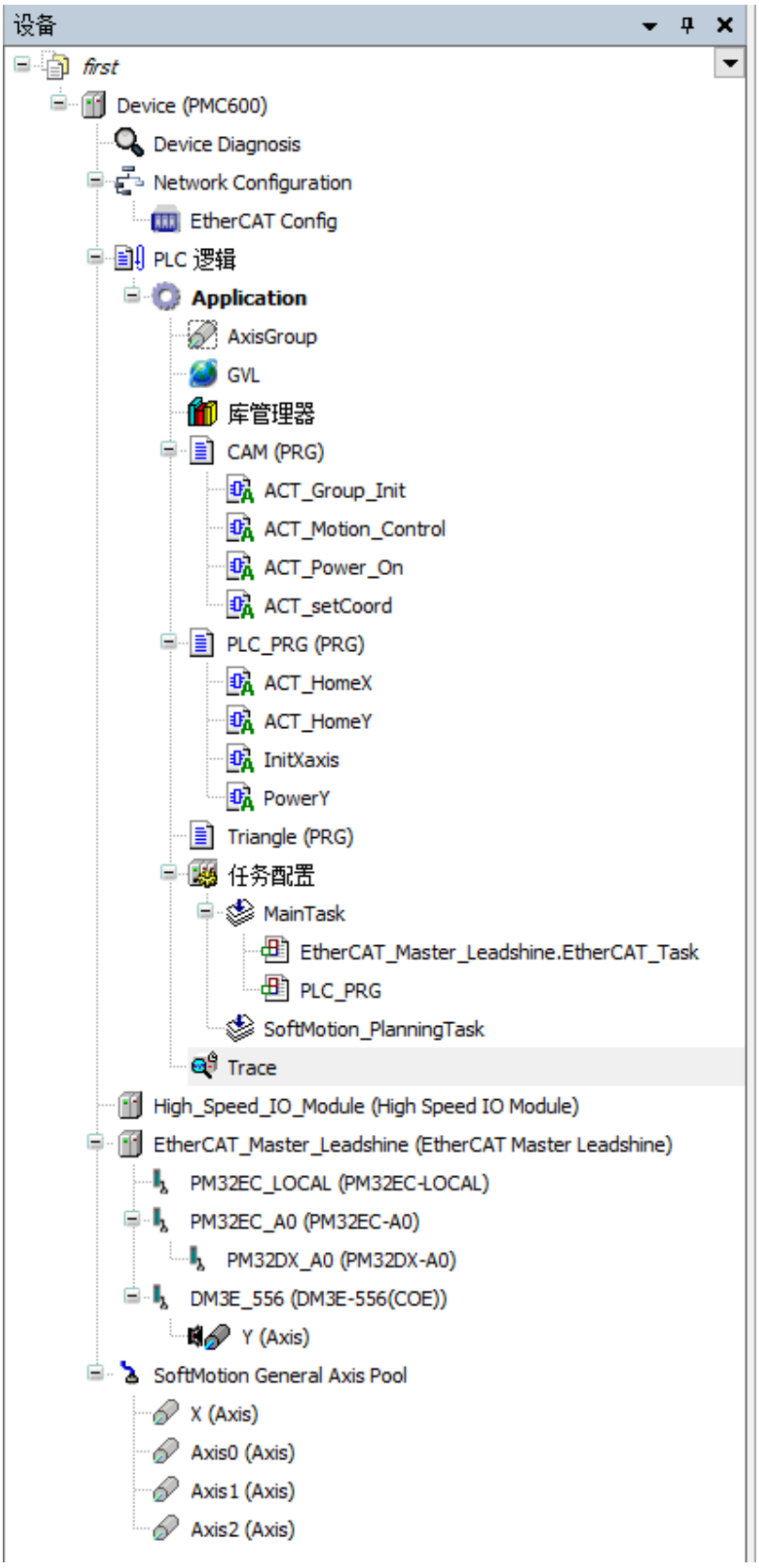
设备
first
Device (PMC600)
Device Diagnosis
Network Configuration
EtherCAT Config
PLC 逻辑
Application
AxisGroup
GVL
库管理器
CAM (PRG)
ACT_Group_Init
ACT_Motion_Control
ACT_Power_On
ACT_setCoord
PLC_PRG (PRG)
ACT_HomeX
ACT_HomeY
InitXaxis
PowerY
Triangle (PRG)
任务配置
MainTask
EtherCAT_Master_Leadshine.EtherCAT_Task
PLC_PRG
SoftMotion_PlanningTask
Trace
High_Speed_IO_Module (High Speed IO Module)
EtherCAT_Master_Leadshine (EtherCAT Master Leadshine)
PM32EC_LOCAL (PM32EC-LOCAL)
PM32EC_A0 (PM32EC-A0)
PM32DX_A0 (PM32DX-A0)
DM3E_556 (DM3E-556(COE))
Y (Axis)
SoftMotion General Axis Pool
X (Axis)
Axis0 (Axis)
Axis1 (Axis)
Axis2 (Axis)

图 5.65　例程的项目树

5.8 配置任务的执行方式和周期

双击“MainTask”界面，可以进入到任务配置界面，可以根据实际情况对任务的类型、任务执行周期、优先级进行修改。

在 MainTask 中，需要保证 EtherCAT 主站相关的功能处于最高优先级，能够在任务中第一时间被执行到，如图 5.66 所示，需要将“EtherCAT_Master_Leadshine.EtherCAT_Task”排列在首位。同时将需要优先执行的代码块往前排列。

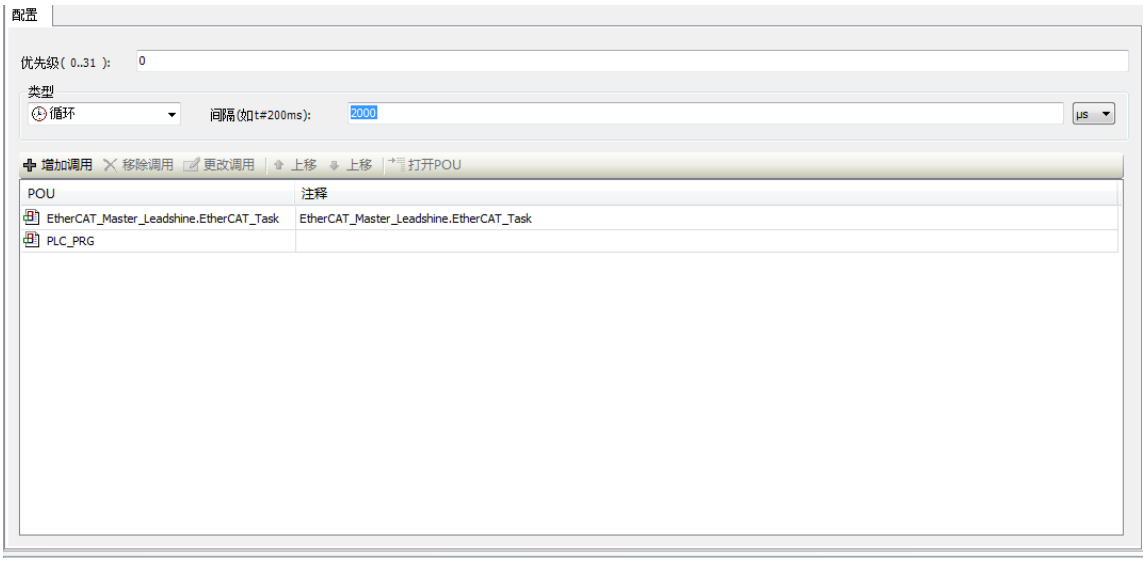

图 5.66　任务配置界面

5.9 程序独立运行

当一个应用程序在 PC 机上用 iStudio 软件编写、下载、调试完成后，该程序已经保存在 SMC600 运动控制器的 Flash 中，调电后不会消失。

拔掉 SMC600 与 PC 机连接的网线，开机后，应用程序就在 SMC600 上独立运行。

深圳市雷赛控制技术有限公司
SHENZHEN LEADSHINE CONTROL TECHNOLOGY CO.,LTD

深圳市雷赛控制技术有限公司
地　址：深圳市深圳市南山区学苑大道 1001 号南山智园 A 3 栋 9 楼
邮　编：518052
电　话：0755-26415968
传　真：0755-26417609
Email：　info@szleadtech.com.cn
网　址：http：//www.szleadtech.com.cn