CTS7-200与西门子 S7-200 CPU模块区别使用手册

前言

本手册的用途

本手册专门用来介绍 TrustPLC CTS7-200 CPU和西门子 SIMATIC S7-200 CPU模块的区别及区别的使用。

CTS7-200 的详细使用手册请参考 CT的“ TrustPLC CTS7-200 用户手册 ”。

为了便于表述，本文使用 CT表示 Co-Trust 的简称。

为了便于说明，本文中使用部分软件截图。

目录

1. CTS7-200 和西门子 S7-200 CPU 模块的主要区别

CTS7-200 和西门子 S7-200 CPU 模块的主要区别有以下方面：

1. 暂不支持在线升级 STL程序。
2. 暂不支持在线编程功能（ Program Edit In Run ）。
3. System Block 的增加内存设置无效（默认为允许使用最大的内存空间）；
4. 暂不支持西门子的通信模块（如 EM277 、特殊模块（如 EM241 EM243 EM253 ）及部分模拟量模块（如 EM231-0HC 232-0HB）。
5. 不支持西门子的电池卡和存储卡，只支持 CT的存储卡和电池卡。
6. 通信端口的区别， CPU集成 2 个或 3 个通信口。
7. 提高了网络读写的通信能力：单次网络读写可扩大到 200 个字节。
8. CTS7-200内置 MODBUS主站和从站协议，不占用用户程序空间及数据空间
9. 增加了 1 个扩展 100K数据块；
10. 增加了 1 个永久保存 V内存功能库；
11. 增加了 2 个自整定智能温控 PID 库；
12. 扩大了 CPU的程序空间和增加了程序保密性： 通过调用 CT提供的动态库， CPU226M / L/H/PSC266的程序空间可扩展到 72K， CPU224的程序空间可扩展到 16K；
13. 关于远程维护，西门子有 2 种方式，但我们的 CPU只支持其中的一种方式；
14. 只支持 CT专用的存储卡和电池卡。
15. CPU运算速度更快。

和西门子 CPU 不同部分的使用说明见以下各章节。

2. CPU 通信端口的使用

2.1. CPU通信口介绍

CTCPU PPI口采用白色接头，自由口采用黑色接头。

2.1.1. CPU224+

通讯端口有 2 个逻辑口， PORT0 逻辑口为 PPI 口，包含 RS232 和 RS485 两种物理接口； FPORT 包含 RS485 自由口和 RS232 微打 2 种物理接口，内部集成 MODBUS 等通信协议，不支持 PPI、MPI 等通信协议。

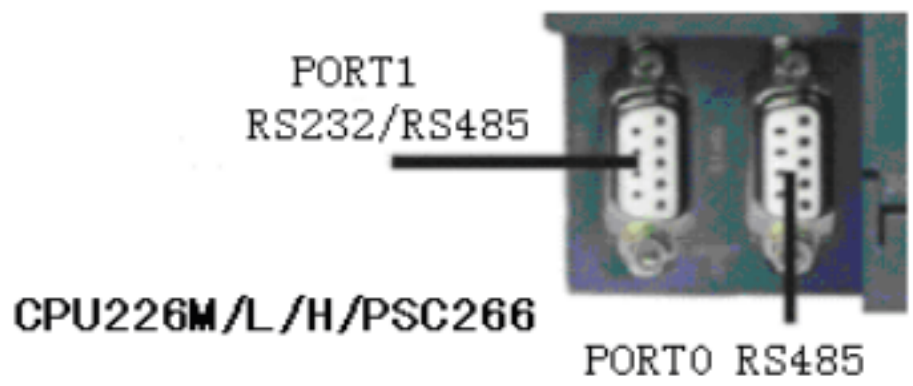
CPU224+ 的 PORT0 和 FPORT 口如下图所示：

控制 PORT0 为 RS232 还是 RS485 通讯的拨码开关如下图所示：

2.1.2. CPU226M/CPU226L/CPU226H/PSC266

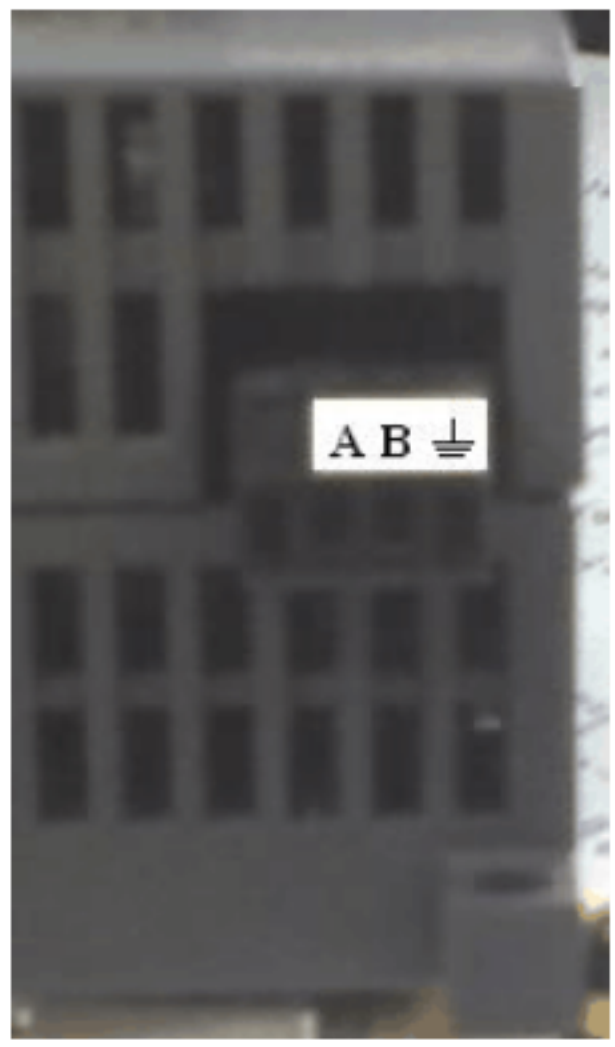
有 3 个 RS485 接头，逻辑口 3 个，分别对应两个网络端口（ Port 0/Port 1）和 FPORT。PORT0 和 PORT1 都为 PPI 口, PORT0 只有 RS485 接口，PORT1 包含 RS232 和 RS485 两种物理接口； FPORT 为 RS485 自由口，在 CPU 的左上角。

CPU226M/L/H/PSC266 的 PORT0 和 PORT1 口如下图所示：



CPU226M/CPU226L/CPU226H/PSC266　的 FPORT 口如下二图所示：

CPU226L 的 FPORT 图



CPU226M/CPU226H/PSC266 的 FPORT 图

2.2. CPU通信口的使用

2.2.1. CPU224+

PORT0 为 PPI 口，支持 RS232 和 RS485 2 种通信方式。使用 RS485 口通信时将拨码开关拨到 485 一侧，支持西门子的 PPI 和 MPI 编程电缆；使用 RS232 口通信时将 CPU 侧面的拨码开关拨到 232 一侧，只支持 CT-RS232 专用编程电缆。

FPORT 为自由口，支持 RS485 和 RS232 微打 2 种。

2.2.1.1.　PORT0口和西门子　PPI\MPI 编程电缆的连接

PC　　CPU 224+ PORT0

[RS232]　　[RS232] [RS485]

|　　|

|　　西门子　PPI/MPI 编程电缆　　|

A 端 <----------> B　　端 ----------|

2.2.1.2.　PORT0口和 CT-RS232专用编程电缆的连接

PC　　CPU 224+ PORT0

[RS232]　　[RS232] [RS485]

|　　|

|　　CT-RS232 专用编程电缆　　|

A 端 <----------> B　　端 ----------|

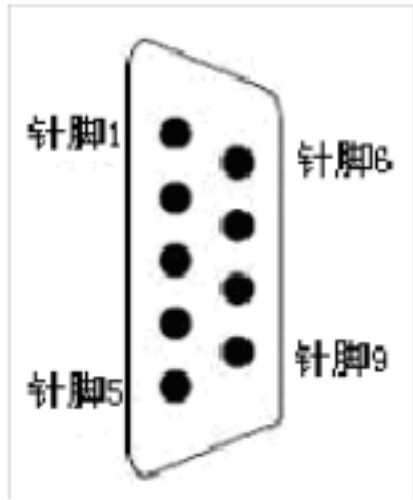
CT-RS232 编程电缆的　2 端以下　2 图所示：

接口 (A) ，连接　PC :

接口 (B)，　连接　CPU224

2.2.1.3.　和自制的　RS232电缆的连接

CPU224+通讯端口引脚定义：

连接器	插针号	**FPORT （RS485）**	**PORT0 （RS232）**
	1	机壳接地	机壳接地
	2	RXD （RS232）	RXD（RS232 或明或 24V 地）#
	3	RS-485 信号 B	RS-485 信号 B
	4	CTS（RS232）	RTS（TTL ）
	5	逻辑地	逻辑地
	6	+5 V、100Ω 串联电阻器	+5 V、100Ω 串联电阻器
	7	RTS（RS232）	+24V
	8	RS-485 信号 A	RS-485 信号 A
	9	TXD （（RS232）	TXD （RS232）
	连接器外壳	机壳接地	机壳接地

自制 RS232 电缆的使用和 CT-RS232 完全相同。

2.2.2. CPU226M/CPU226L/CPU226H/PSC266

PORT0 为 PPI 口，其中 PORT0 为 RS485 口，支持西门子的 PPI 和 MPI 编程电缆。

PORT1 也为 PPI 口， 支持 RS232 和 RS485 2 种通信方式。 由于此 PPI 口不带电源， 所以不能支持西门子的 PPI/MPI 编程电缆。

FPORT 为自由口，内部集成 MODBUS 等通信协议。

2.2.2.1. PORT0 口和西门子 PPI\MPI 编程电缆的连接

PC CPU 226M/L/H/PSC266

[RS232] [PORT1] [PORT0]

| |

| 西门子 PPI/MPI 电缆 |

A 端<----------> B 端 ----------|

2.2.2.2. PORT1口和 CT-RS232专用编程电缆的连接

PC CPU 226M/L/H/PSC266 PORT1

[RS232] [RS232] [RS485]

| |

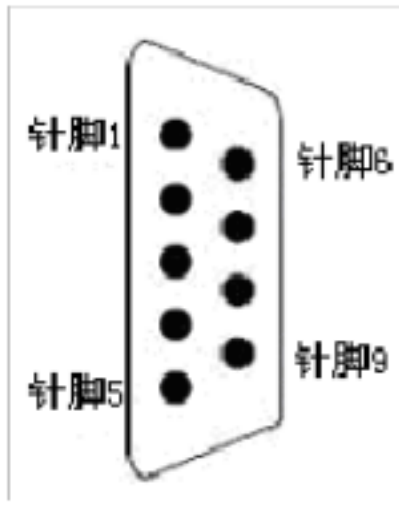
| CT-RS232 专用编程电缆 |

A 端 <----------> B 端 ----------|

连接方法和 **CPU224+** 相同。

2.2.2.3. 和自制的 RS232电缆的连接

CPU226M/CPU226L/CPU226H/PSC266 通讯端口引脚定义：

连接器	插针号	**PORT0**	**PORT1**		**FPORT**
	1	机壳接地	机壳接地		1(地)：机壳接地
	2	逻辑地	RS232 信号 RXD		2(B)：RS485 信号 B/+
	3	RS485 信号 B	RS485 信号 B		3(A)：RS485 信号 A/-
	4	发送申请	发送申请		4(空)：需用终端匹配时将 3、4 有连在一起。(CPU226L 无此项)
	5	逻辑地	逻辑地		
	6	+5V，100Ω	+5V，100Ω		
	7	+24V	空/+24V		
	8	RS485 信号 A	RS485 信号 A		
	9	10 位协议选择（输入）	RS232 信号 TXD		
	连接器外壳	机壳接地	机壳接地		

自制 RS232 电缆的使用和 CT-RS232 完全相同。

3. 提高网络读写能力的使用

3.1. 如何提高网络读写能力的介绍

有 2 个方面的提高：

1. 原来 NETR/NETW 指令，只能读写 1－16 字节的数据。新版本增强型的网络读写指令，单次可以支持 1－200 个字节。
2. 增强型的 CPU 型号，对网络读写操作进行了优化， 优化后的性能提高体现在 19.2 或者 187.5 两种波特率下， 当网络中存在某个站被多个主站读写访问的情况时， 增强型的 CPU 在当前读写操作过程中还能继续接受其它读写操作， 也就是说增强型 CPU 能够同时处理多条网络操作（CPU 最多能同时处理 8 个网络读写操作），所以这样网络的效率将得到了极大的提高和改善。原来从站 CPU 同时只能接受 1 个网络读写操， 不能同时处理多个网络操作 ，只有当当前网络读写操作完成后，才能处理下一个网络读写操作。

3.2. 增强型网络读写的使用

增强型的网络读写指令，和原来的的 NETR/NETW 指令完全一样，唯一的区别就是 TBL 的“数据长度”可以达到 200 字节，网络读写的 TBL 参数和原来不一样。

增强型的网络读写的 TBL 参数表如下：

TBL 内容					字节顺序
D	A	E	0	错误代码	0
远程站地址					1
远程站的数据区指针（I，Q，M，V）					2
					3
					4
					5
数据长度（ 1－200）					6
数据字节 0					7
数据字节 1					8
,,					
数据字节 198					205
数据字节 199					206

4. CT-MODBUS 主站和从站库的使用

4.1. CT_MODBUS 库介绍

一共有 4 个库，分别是 PORT0 口的主站、从站库， PORT1 口的主站和从站库。

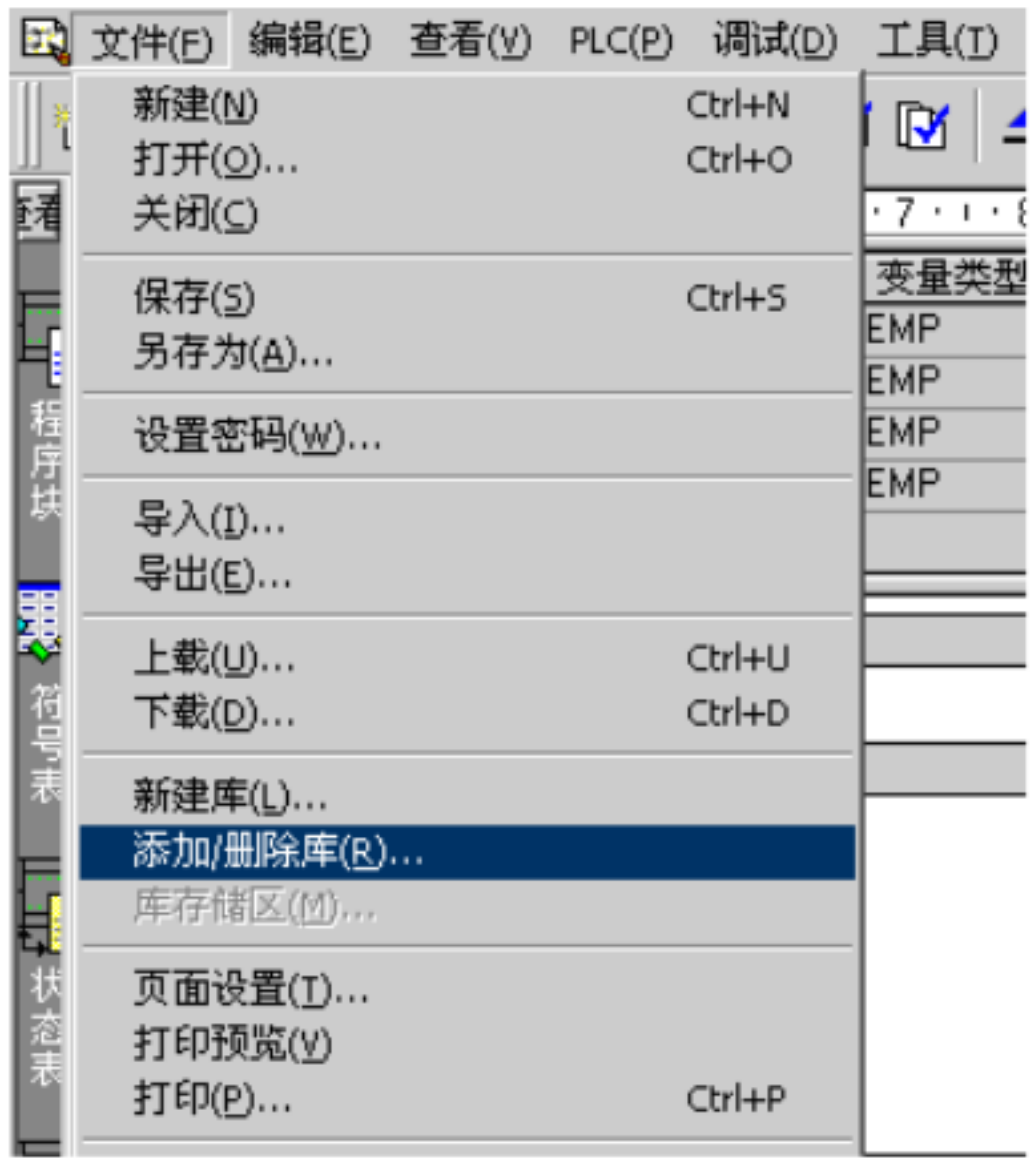
CT_MODBUS 功能块主要针对西门子 MODBUS 功能块占用 CPU 大量程序空间和数据空间而提供的一组内嵌的简单易用的 MODBUS 协议库。 CT_MODBUS 功能块是集成在 CPU 内部，不占用用户程序空间和数据空间，作为一组库函数提供给用户使用。

注意：用 modbus 协议和台达变频器通信时， 不支持 7 位数据位格式， 如 P92 01 资料格式 <7,E,1>。

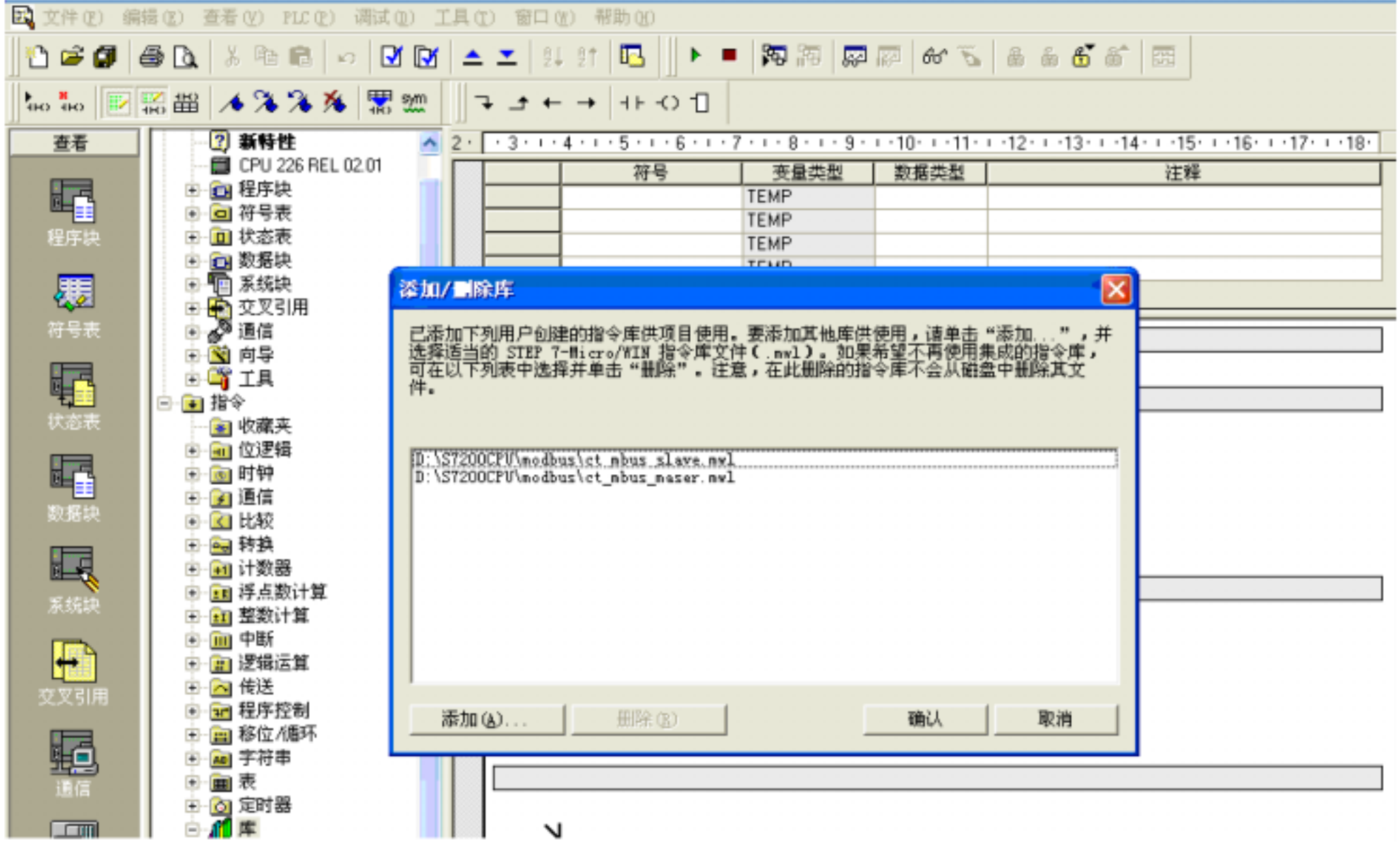
4.2. 安装说明

4.2.1. 添加库文件

在“文件” ----“添加 /删除库”，找到库文件“ ct_mbus_master.mwl ”和“ ct_mbus_slave.mwl ”,如下图所示。



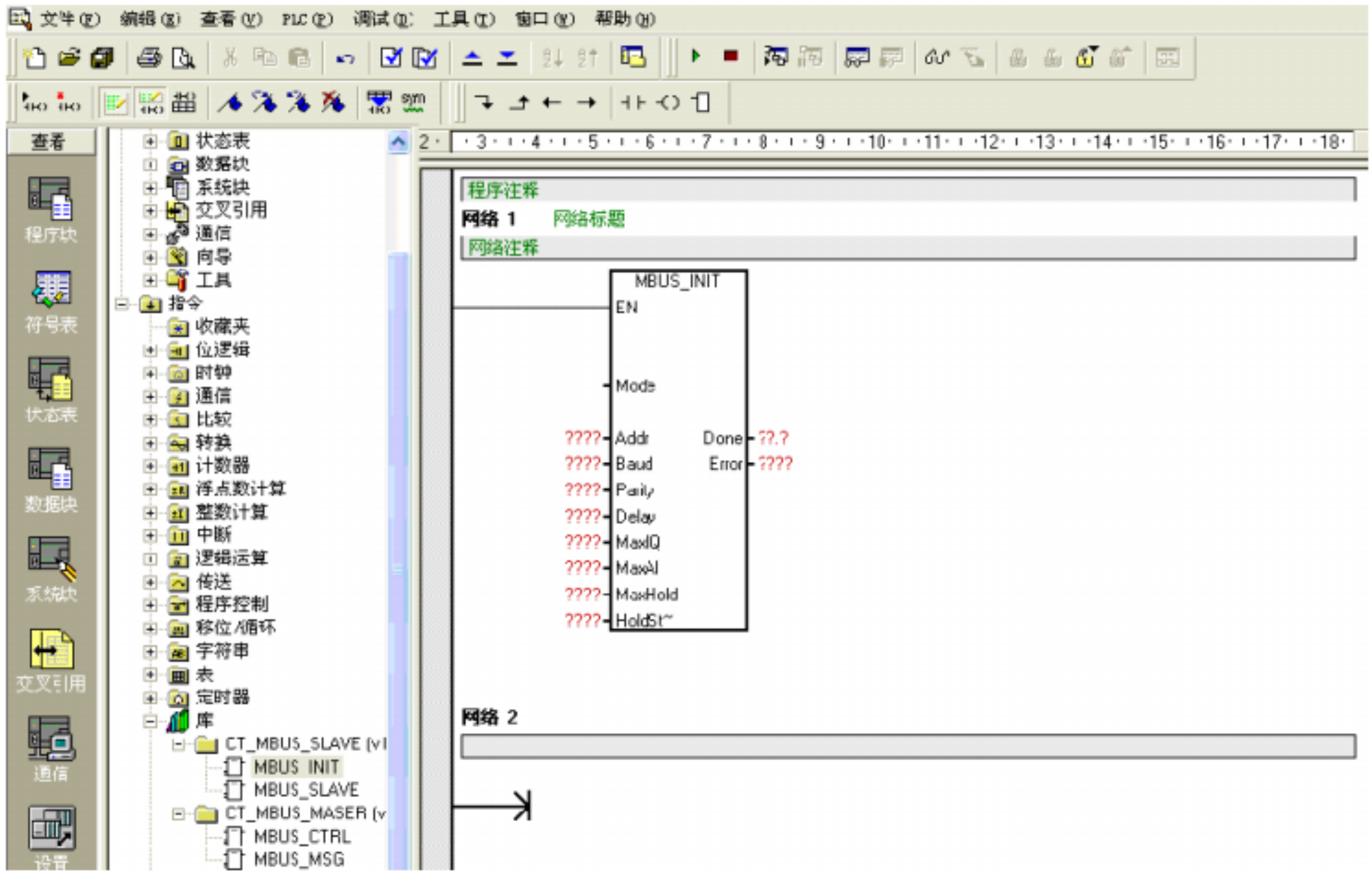
在你存放的 “ ct_mbus_master.mwl ”和“ ct_mbus_slave.mwl ”文件的位置， 找到文件， 如下图所示， 点“ 添加 “ 按钮。



安装成功后， 在目录树的“ 库“ 下可以看到新增加的 CT_MBUS_MASTER 和 CT_MBUS_SLAVE 的库：

4.2.2. 调用 CT_MODBUS 库

点要添加功能块的“网络”，双击“库”下面的“ MBUS_INIT ”,“ MBUS_SLAVE ”,“ MBUS_CTRL ”,“ MBUS_MASTER ”就会在“网络”里出现相应的功能块。结果如下图所示：



4.3. CT_MODBUS_S 功能说明

4.3.1. Modbus地址

Modbus地址通常是包含数据类型和偏移量的 5个或6个字符值。第一个或前两个字符决定数据类型，最后的四个字符是符合数据类型的一个适当的值。 Modbus主站则将这个地址对应到正确的功能上。 Modbus从站指令支持以下地址：

00001 至00128是实际输出，对应于Q0.0--Q15.7;

10001 至10128是实际输入，对应于I0.0 —I15.7;

30001至30032是模拟输入寄存器，对应于 AIW0至AIW62;

40001 至4XXXX是保持寄存器，对应于 V内存区。

所有Modbus地址都是从 1开始编号的。右表所示为 Modbus地址与S7--200 地址的对应关系。 Modbus从站协议允许您对 Modbus主站可访问的输入、输出、模拟输入和保持寄存器 V区）的数量进行限定。

Modbus地址	S7-200地址
000001	Q0.0
000002	Q0.1
000003	Q0.2
...	...
000127	Q15.6
000128	Q15.7
010001	I0.0
010002	I0.1
010003	I0.2
...	...
010127	I15.6
010128	I15.7
030001	AIW0
030002	AIW2
030003	AIW4
...	...
030032	AIW62
040001	HoldStart
040002	HoldStart+2
040003	HoldStart+4
...	...
04xxxx	HoldStart+2 x（xxxx-1）

4.3.2. 使用 Modbus从站协议指令

4.3.2.1. CT_MODBUS_S从站协议指令占用 CTS7-200CPU的资源

1. 根据使用不同的 Modbus协议库占用自由口 0 或者 1 作为 Modbus从站协议通讯。当 Port 0 或者 Port 1 作为 Modbus协议通讯时，它不能再作为其它任何目的使用， 包括与 SETP7-Micro/WIN 通讯，自由口通讯。 MBUS_INIT指令控制 Port 的设定是 Modbus协议还是 PPI。
2. 与选用 Port 自由口通讯相关的所有的 SM。
3. 占用 92 字节程序空间。

4.3.2.2. 在 CTS7-200 程序中使用 Modbus从站协议指令遵循的步骤

1. 在您的程序中插入 MBUS_INIT指令并且只在一个循环周期中执行该指令， MBUS_INIT指令可用于对 Modbus通讯参数的初始化或修改。 当您插入 MBUS_INIT 指令时，几个隐藏的子程序和中断

服务程序会自动地添加到您的程序中。

2. 在您的程序中只使用一个 MBUS_SLAVE 指令。该指令在每个循环周期中执行，为接收到的所有请求提供服务。
3. 用通讯电缆将 S7--200 通讯口与 Modbus主站连接起来。

4.3.2.3. Modbus 从站协议指令所支持的功能

Modbus 从站协议指令支持 Modbus RTU 协议。这些指令使用 S7--200 的自由口功能，支持大部分常用 Modbus 功能。以下是所支持的 Modbus 功能：

功能	描述
1	读单个/多个线圈（实际输出）状态，功能1返回任意数量输出点的接通/断开状态（Q）
2	读单个/多个接触器（实际输入）状态，功能2返回任意数量的输入点的接通/断开状态（I）
3	读单个/多个保持寄存器，功能3返回V存储器的内容，保持寄存器在Modbus下是字类型，在一个请求中最多可读120个字。
4	读单个/多个输入寄存器，功能4返回模拟输入值。
5	写单个线圈（实际输出），功能5将实际输出点设置为指定值，该输出点不是被强制，用户程序可以重写由Modbus的请求而写入的值。
6	写单个保持寄存器，功能6写一个单个保持寄存器的值到S7-200的V存储区。
15	写多个线圈（实际输出），功能15写多个实际输出值到S7-200的Q映象区，起始输出点必须是一个字节的开始（如，Q0.0或Q2.0），并且要写的输出的数量是8的倍数，这是Modbus从站协议指令的限定，这些点不是被强制，用户程序可以重写由Modbus的请求而写入的值。
16	写多个保持寄存器，功能16 写多个保持寄存器到S7-200的V区，在一个请求中最多可写120字。

4.3.2.4. MBUS_INIT 指令

MBUS_INIT 指令用于使能和初始化或禁止 Modbus通讯。MBUS_INIT 指令必须无错误的执行，然后才能够使用 MBUS_SLAVE 指令。在继续执行下一条指令前，MBUS_INIT 指令必须执行完并且 Done位被立即置位。MBUS_INIT 指令应该在每次通讯状态改变时只执行一次。因此，EN输入端应使用边沿检测元素以脉冲触发，或者只在第一个循环周期内执行一次。

参数说明：

参数地址	说明	类型	数值范围	备注
Mode	选择通讯协议：输入 1值将 Port 定义为 Modbus协议并使能该协议，输入值为 0将 Port 定义为 PPI并禁止 Modbus协议。	位		
Addr	设置本站地址	字节	1到247之间	
Baud	设置波特率。	双字	1200、2400、4800、9600、19200、38400、57600、115200	
Parity	设置校验。	字节	0-- 无校验	所有设置使用一

			1-- 奇校验 2-- 偶校验	个停止位。
Delay	通过为标准 Modbus信息超时增加指定数量的毫秒，扩展标准 Modbus信息结束超时条件	整型	0到32767	单位：毫秒
MaxIQ	设置可使用的 I 和Q点数	整型	其数值可为 0到128。数值为 0则禁止对输入和输出的读写	建议 MaxIQ的取值为128，即允许访问S7--200 的所有 I 点和 Q点。
MaxAI	设置可使用的字输入寄存器（AI ）的个数	整型	0到32。值为 0则禁止读模拟输入。	MaxAI的建议值如下： - CPU221为 0 - CPU222为16 - CPU224,CPU226和CPU224XP为32。
MaxHold	设置可以使用的 V存储区字保持寄存器的个数	整型	0到32767	单位：字
HoldStart	设置可以使用的 V存储区的保持寄存器的起始地址	双字	指向 V存储区的指针	
Done	当MBUS_INIT指令完成时，Done输出接通	位		
Error	Error 输出字节包含该指令的执行结果。	字节		

4.3.2.5. MBUS_SLAVE指令

MBUS_SLAVE指令用于服务来自 Modbus主站的请求，必须在每个循环周期都执行，以便检查和响应 Modbus请求。当 EN输入接通时，该指令在每一循环周期内执行。 MBUS_SLAVE指令无输入参数。

参数说明 ：

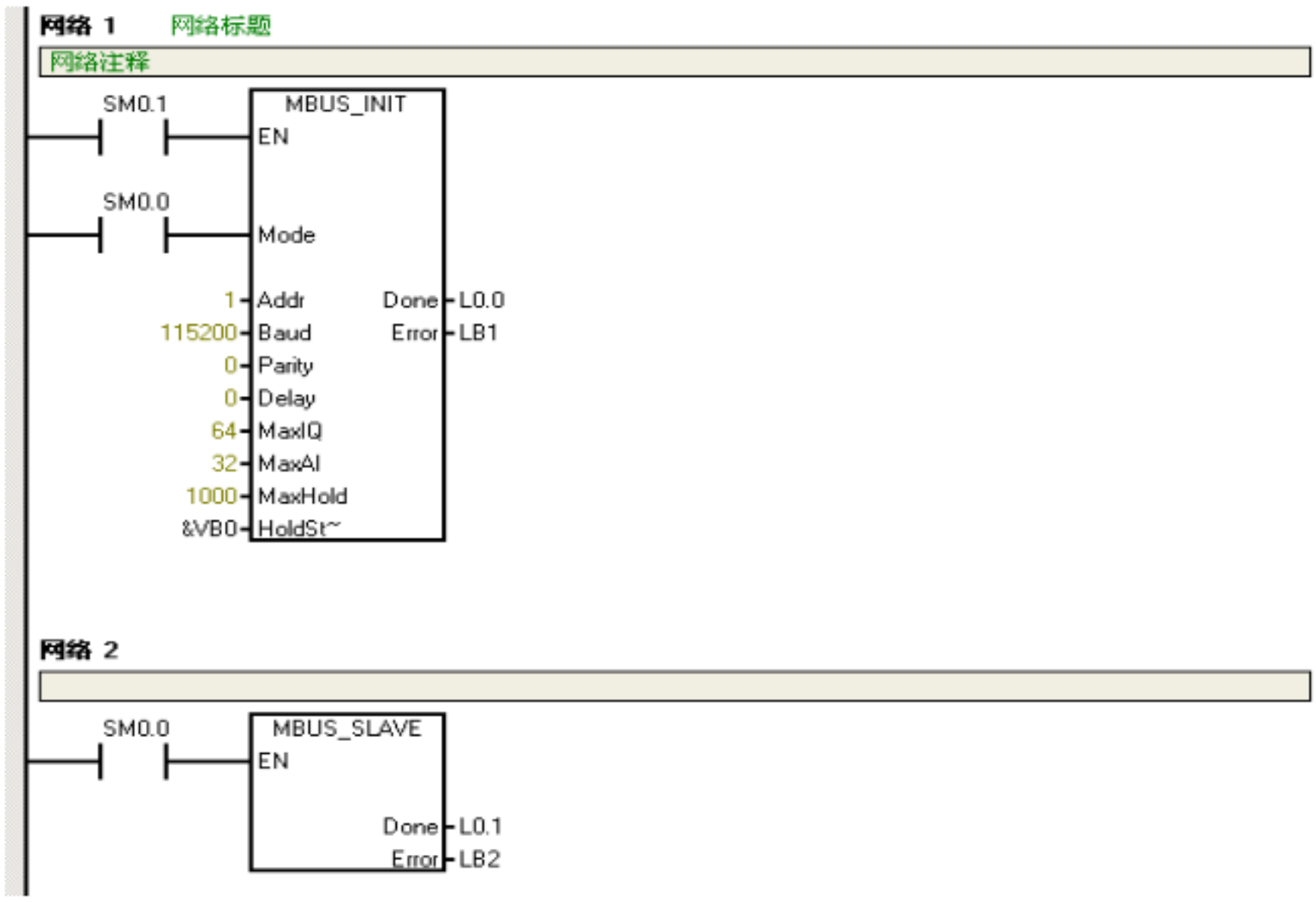
参数地址	说明	类型	数值范围	备注
Done	当MBUS_SLAVE指令响应Modbus请求时 Done输出接通。如果没有服务的请求， Done输出会断开。	位		
Error	输出包含该指令的执行结果。	字节	错误代码如下表	该输出只有 Done接通时才有效。 如果 Done断开，错误代码不会改变。

错误码如下表所示：

0	无错误
1	存储区范围错误
2	非法波特率或校验
3	非法从站地址
4	Modbus参数的非法值
5	保持寄存器与Modbus从站符号地址重叠
6	接收校验错误
7	接收CRC错误
8	非法功能请求/不支持的功能
9	请求中有非法存储区地址
10	从站功能未始能.

4.3.2.6. Modbus 从站协议指令使用实例

下面的梯形图程序就是建立了一个从站地址为 1，波特率为 115200，无校验的 MODBUS 从站：



4.3.3. 使用 Modbus主站协议指令

4.3.3.1. Modbus 主站协议指令占用 CTS7-200CPU的资源

1. 根据使用不同的 Modbus协议库占用自由口 Port 0 或者 Port 1 作为 Modbus从站协议通讯。当 Port 0 或者 Port 1 作为 Modbus 协议通讯时，它不能再作为其它任何目 的使用，包括与 SETP7-Micro/WIN 通讯，自由口通讯。MBUS_INIT指令控制 Port 的设 定是 Modbus协议还是 PPI。
2. 与选用 Port 自由口通讯相关的所有的 SM

3. 占用 119 子节程序空间。

4.3.3.2. MBUS_CTRL 指令

使用 SM0.0 调用 MBUS_CTRL 指令完成主站的初始化，并启动其功能控制。

参数说明 ：

参数地址	说明	类型	数值范围	备注
Mode	设置通讯模式 ：为 1 时，使能 Modbus 协议功能；为 0 时恢复系统为 PPI 协议	位		
Baud	设置波特率	双字	11200， 2400， 4800，9600， 19200， 38400，57600， 115200	
Parity	设置校验。	字节	0-- 无校验 1-- 奇校验 2-- 偶校验	所有设置使用一个停止位。
Timeout	主站等待从站响应的时间，以毫秒为单位	整型	1 - 32767	典型的设置值为 1000 毫秒（1 秒）
Done	完成位 ， 初始化完成，此位会自动置 1	位		
Error	初始化错误代码	字节	0-- 无错误 1-- 校验选择非法 2-- 波特率选择非法 3-- 模式选择非法	只有在 Done 位为 1 时有效

4.3.3.3. MBUS_MSG 指令

使用 SM0.0 调用 Modbus RTU 主站读写子程序 MBUS_MSG 指令，First 接通发送一个 Modbus 请求。同一时刻只能有一个读写功能（即 MBUS_MSG 使能。

各参数如下 ：

参数地址	说明	类型	数值范围	备注
First	读写请求位	位		每一个新的读写请求必须使用脉冲触发
Slave	设置从站地址	字节	1-247	
RW	操作命令	字节	0-- 读 1-- 写	
Addr	选择读写的数据类型	双字	0000 至 0xxxx-- 开关	

			量输出 1000 至 1xxxx-- 开关 量输入 3000 至 3xxxx-- 模拟 量输入 4000 至 4xxxx-- 保持 寄存器	
Count	通讯的数据个数（位或字的个数）	整型		Modbus 主站每一个 MBUS_MSG 指令可读 / 写的最大数据量为 120 个字
DataPtr	数据指针 ， 如果是读指令，读回的数据放到这个数据区中；如果是写指令，要写出的数据放到这个数据区中	双字		
Done	完成位，读写功能完成位	位		
Error	错误代码		错误代码如下	只有在 Done 位为 1 时，错误代码才有效

错误码如下：

0 ＝ 无错误

1 ＝ 响应校验错误

2 ＝ 未用

3 ＝ 接收超时（从站无响应）

4 ＝ 请求参数错误（ slave address, Modbus address, count, RW ）

5 ＝ Modbus/ 自由口未使能

6 ＝ Modbus 正在忙于其它请求

7 ＝ 响应错误（响应不是请求的操作）

8 ＝ 响应 CRC校验和错误

101 ＝ 从站不支持请求的功能

102 ＝ 从站不支持数据地址

103 ＝ 从站不支持此种数据类型

104 ＝ 从站设备故障

105 ＝ 从站接受了信息，但是响应被延迟

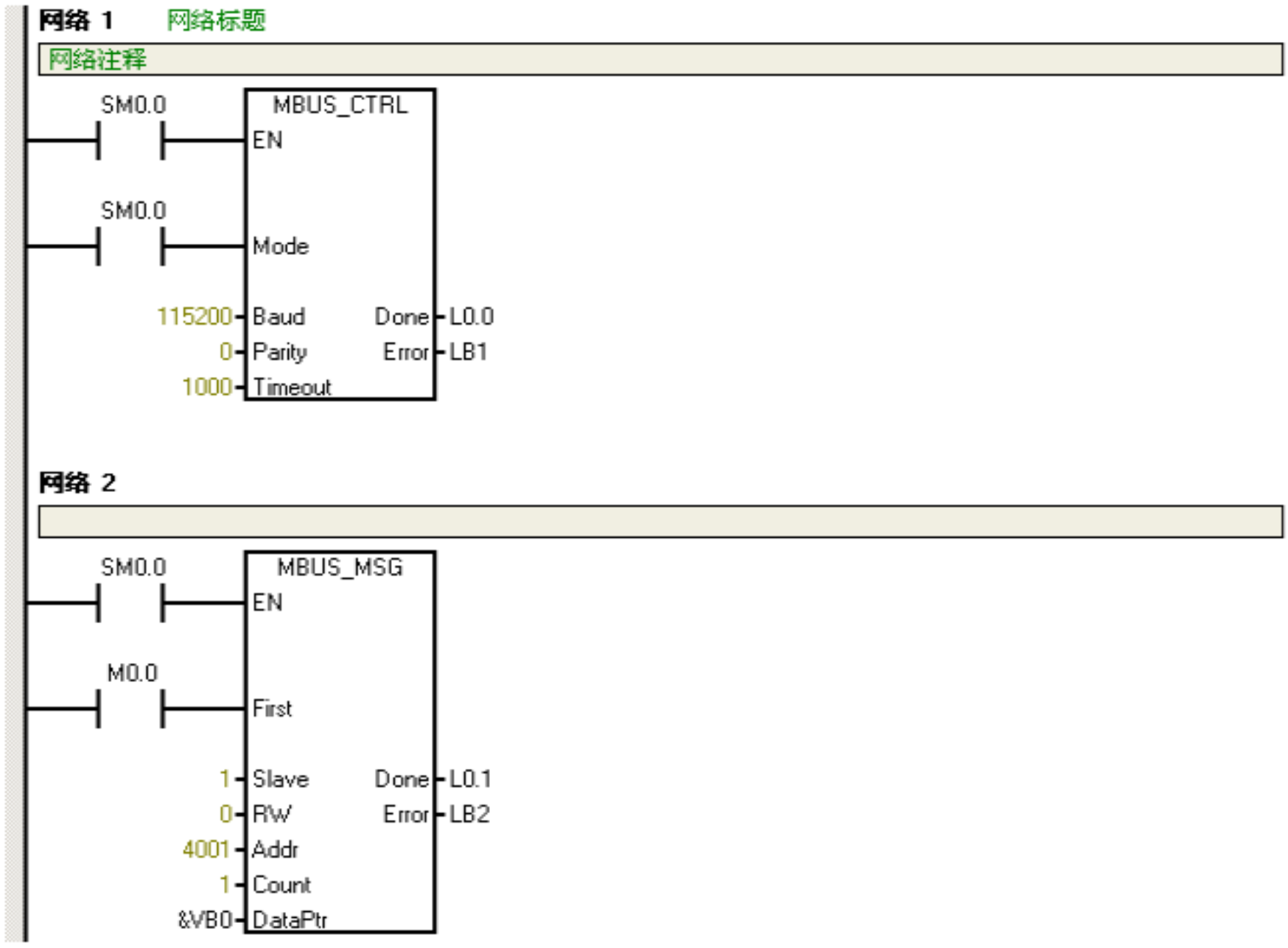
106 ＝ 从站忙，拒绝了该信息

107 ＝ 从站拒绝了信息

108 ＝ 从站存储器奇偶错误

4.3.3.4. Modbus 主站协议指令使用实例

下面的梯形图程序就是建立了一个从站地址为 1，波特率为 115200，无校验的 MODBUS 主站，First 接通发送一个 Modbus 请求：



5. CT 扩展 100K 数据块库的使用

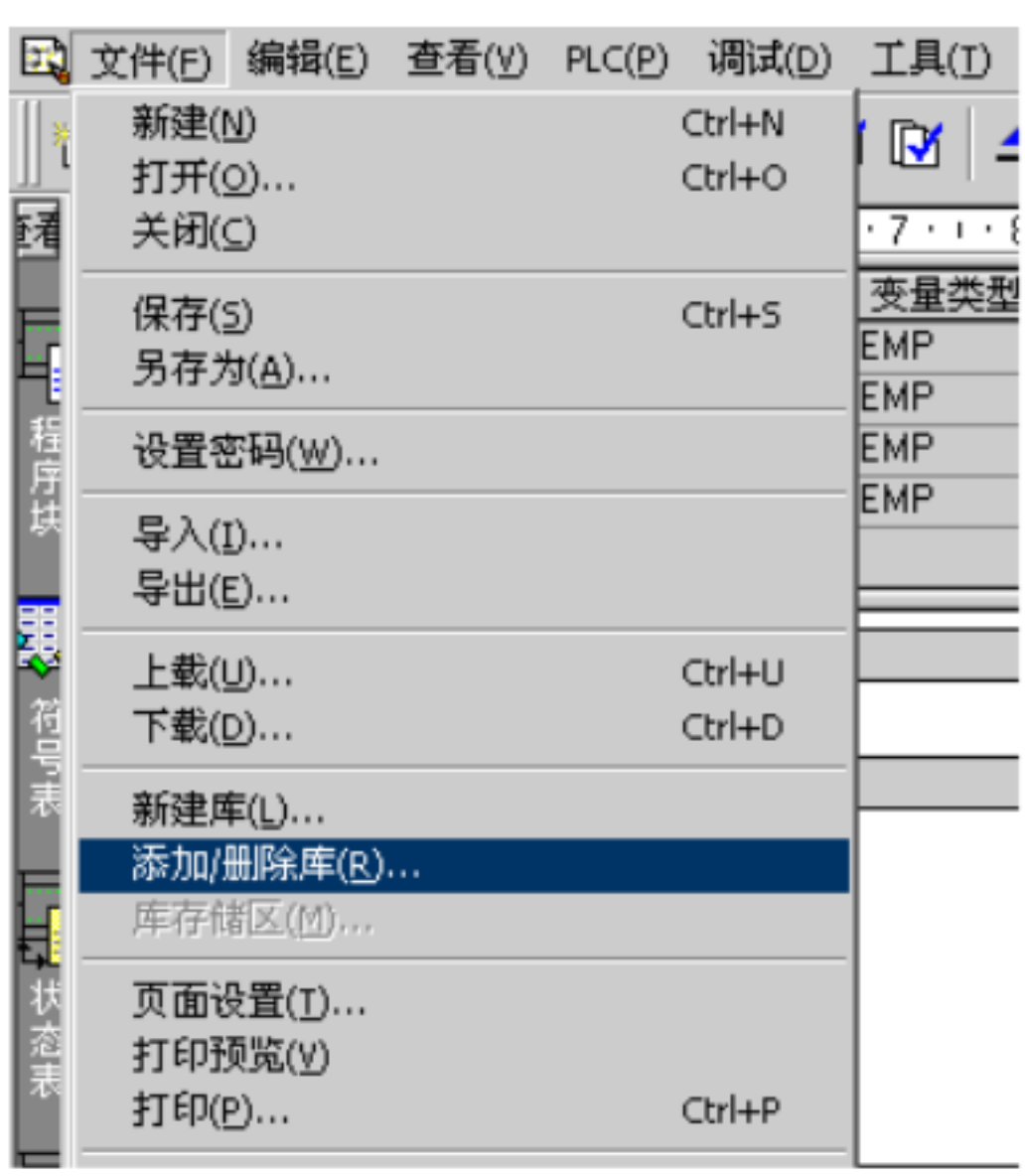
5.1. 功能介绍

扩展数据空间是 CTS7-200 CPU 在已有的存储区域之外扩展的 100K 用户可访问的数据存储空间，这块存储空间的数据保持特性同 V 内存数据空间，在 CPU 断电情况下通过超级电容来保持数据，数据最长保持时间可达 100 个小时，数据存储量为 100K 字节。CO-TRUST 专门为这块数据的访问提供了库指令（请从 www.co-trust.com 网站免费下载），用户可以将 CO-TRUST 提供的库添加到 MicroWin 中，通过库中提供的读写指令访问这块数据空间，实现扩展数据空间和其他数据空间的数据交换。

5.2. 安装说明

5.2.1. 添加库文件

在“ 文件 ” ----“ 添加 /删除库 ”，找到库文件“ pid_t.mwl ”,如下图所示。

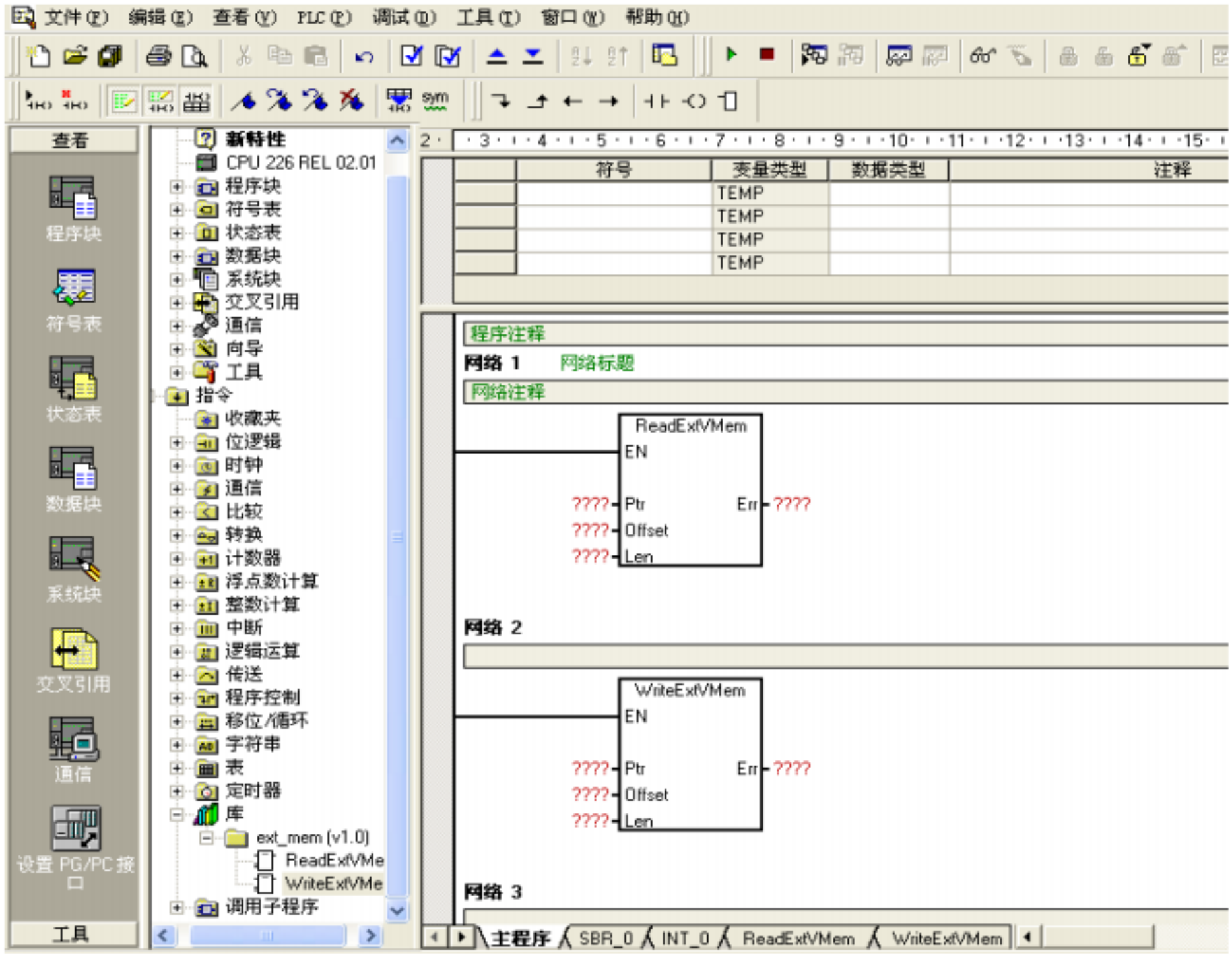


在你存放的 ext_mem.mwl 文件的位置，找到此文件，如下图所示，点“ 添加“ 按钮。

安装成功后，在目录树的“ 库“ 下可以看到新增加的 ext_mem 的库：

5.2.2. 调用 ext_mem库

点要添加功能块的“ 网络 ”， 双击“ 库 ”下面的“ ReadExtVMem” 和“ WriteExtVMem ”,就会在“ 网络 ”里出现相应的功能块。结果如下图所示：



5.3. ext_mem库功能说明

5.3.1. 使用读函数 ReadExtVMem从扩展数据空间读取数据

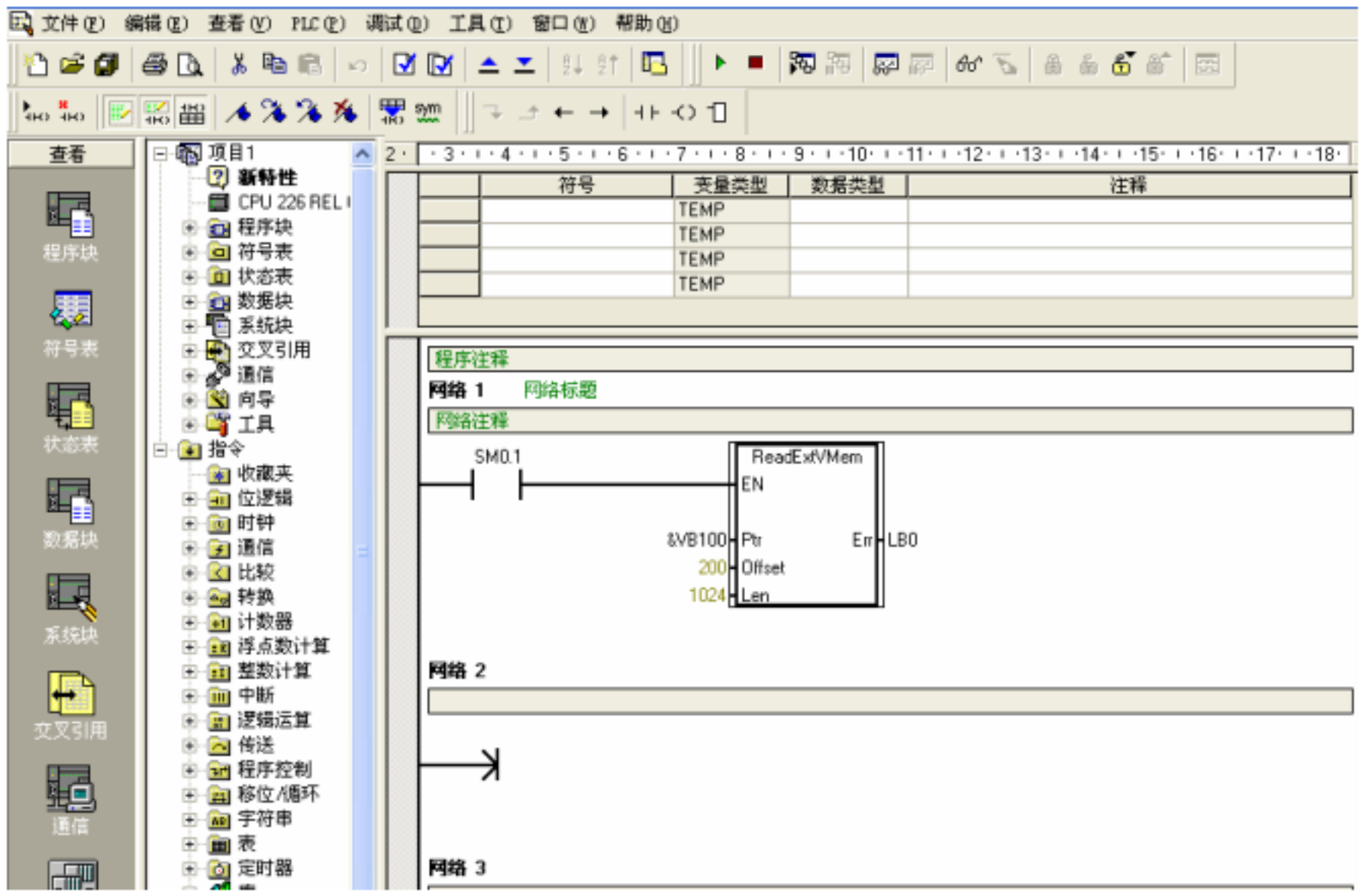
参数地址	说明	类型	数值范围	备注
Ptr	表示要读到的内存地址指针	DWORD		如：&VB0,&IB0
Offset	表示读取扩展内存中的起始偏移地址	DWORD	0-102399	
Len	表示读取内存长度(字节数)	DWORD		
Err	返回值表示读取是否成功	BYTE		0 表示读取成功，其它表示失败。

5.3.2. 使用写函数 WriteExtVMem 向扩展数据空间写入数据

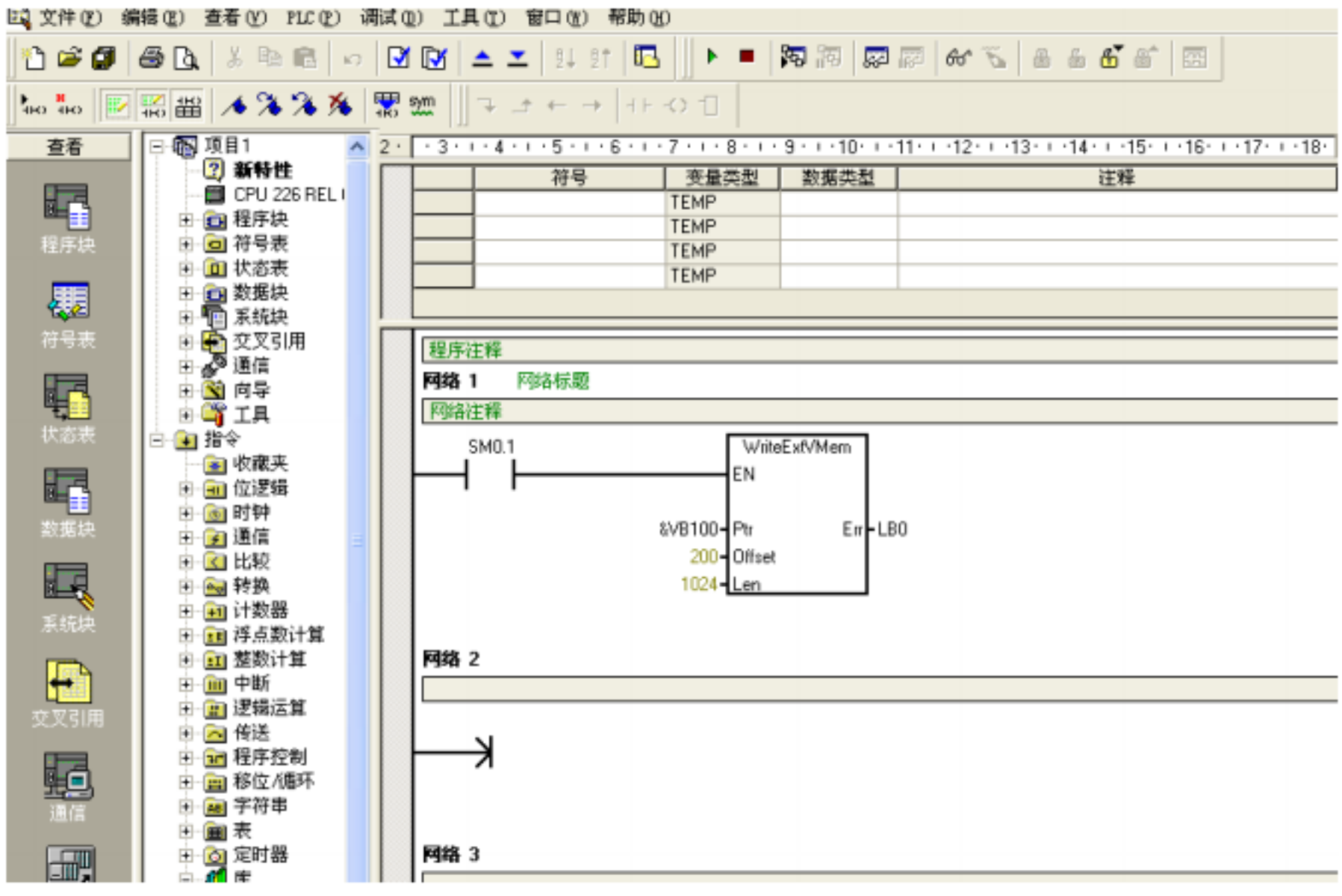
参数地址	说明	类型	数值范围	备注
Ptr	表示要写到扩展数据空间的源数据内存地址指针	DWORD		如：&VB0,&IB0
Offset	表示写到扩展内存中的起始偏移地址	DWORD	0-102399	
Len	表示写入内存长度(字节数)	DWORD		
Err	返回值表示写入是否成功	BYTE		0 表示写入成功，其它表示失败。

5.4. 应用例子

5.4.1. 把扩展内存中从偏移量 200 开始的 1024 个字节读到 VB100开始的内存中



5.4.2. 把 VB100开始的 1024 个字节写到扩展内存中从偏移量 200 开始的内存中



6. CT 永久保存 V 内存功能库的使用

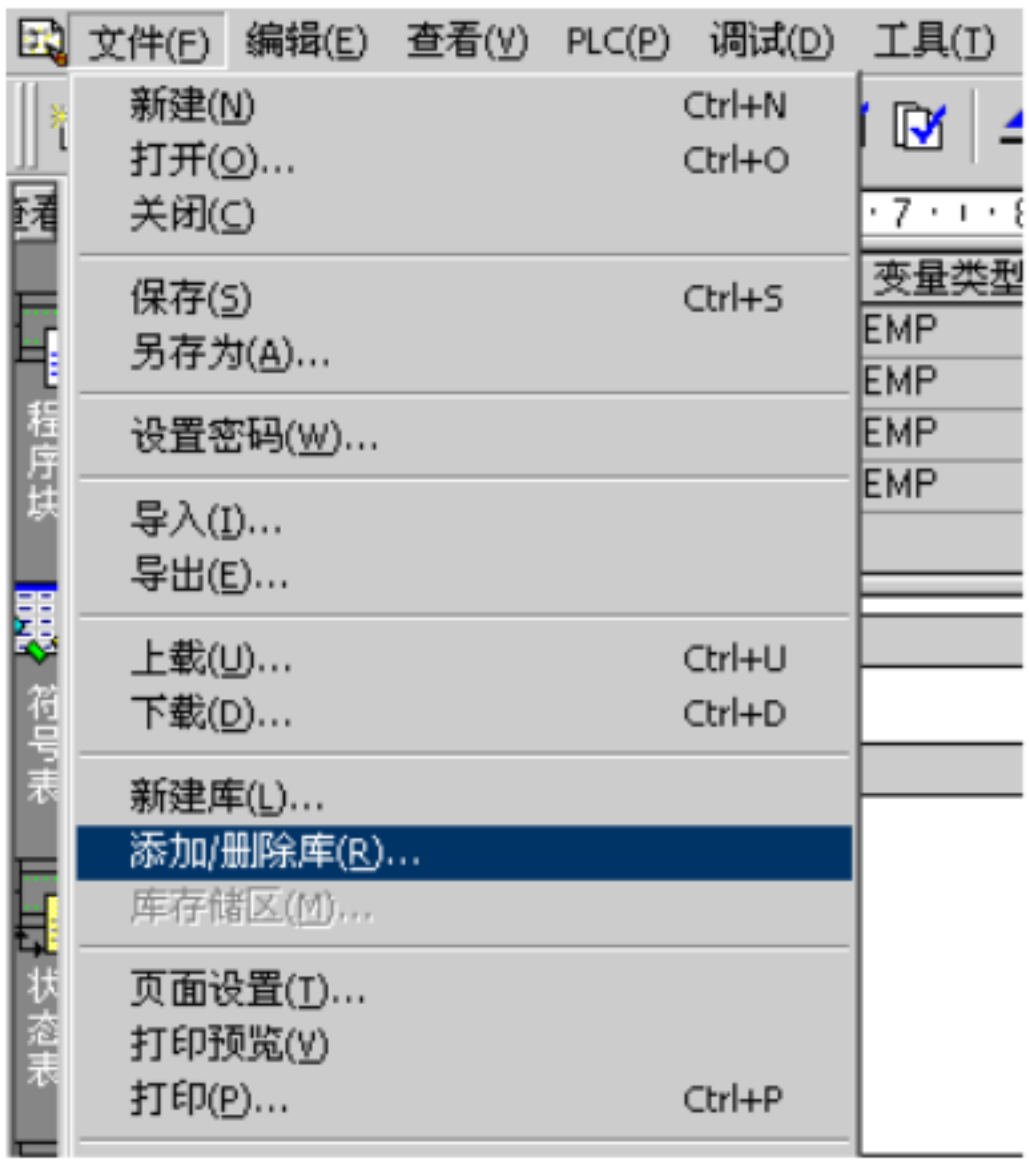
6.1. 功能介绍

CT_SAVEVMEM 做为一个库函数提供给用户使用。其功能是将用户需要保存的一段 V 内存的数据保存到永久性内存中。

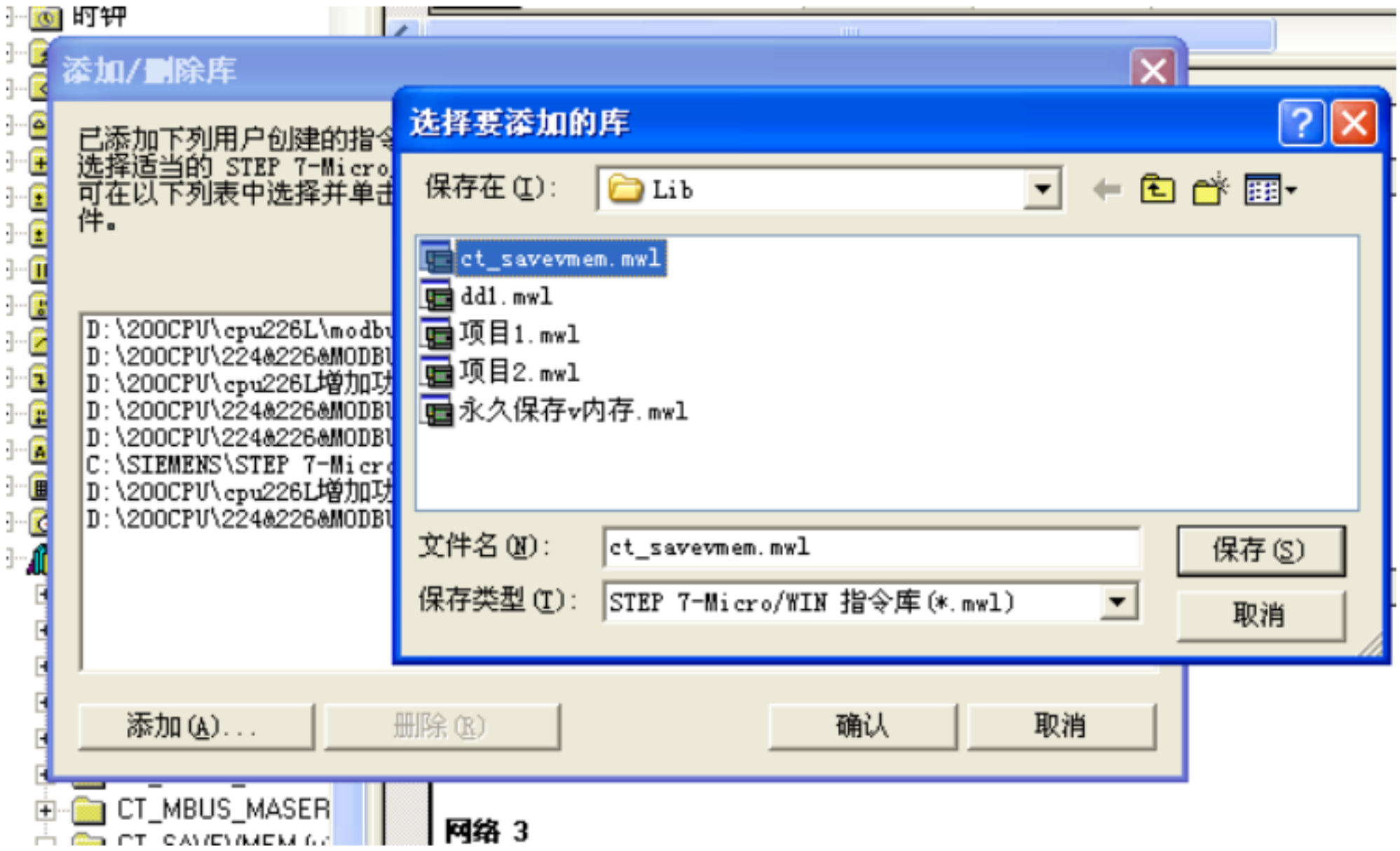
6.2. 安装说明

6.2.1. 添加库文件

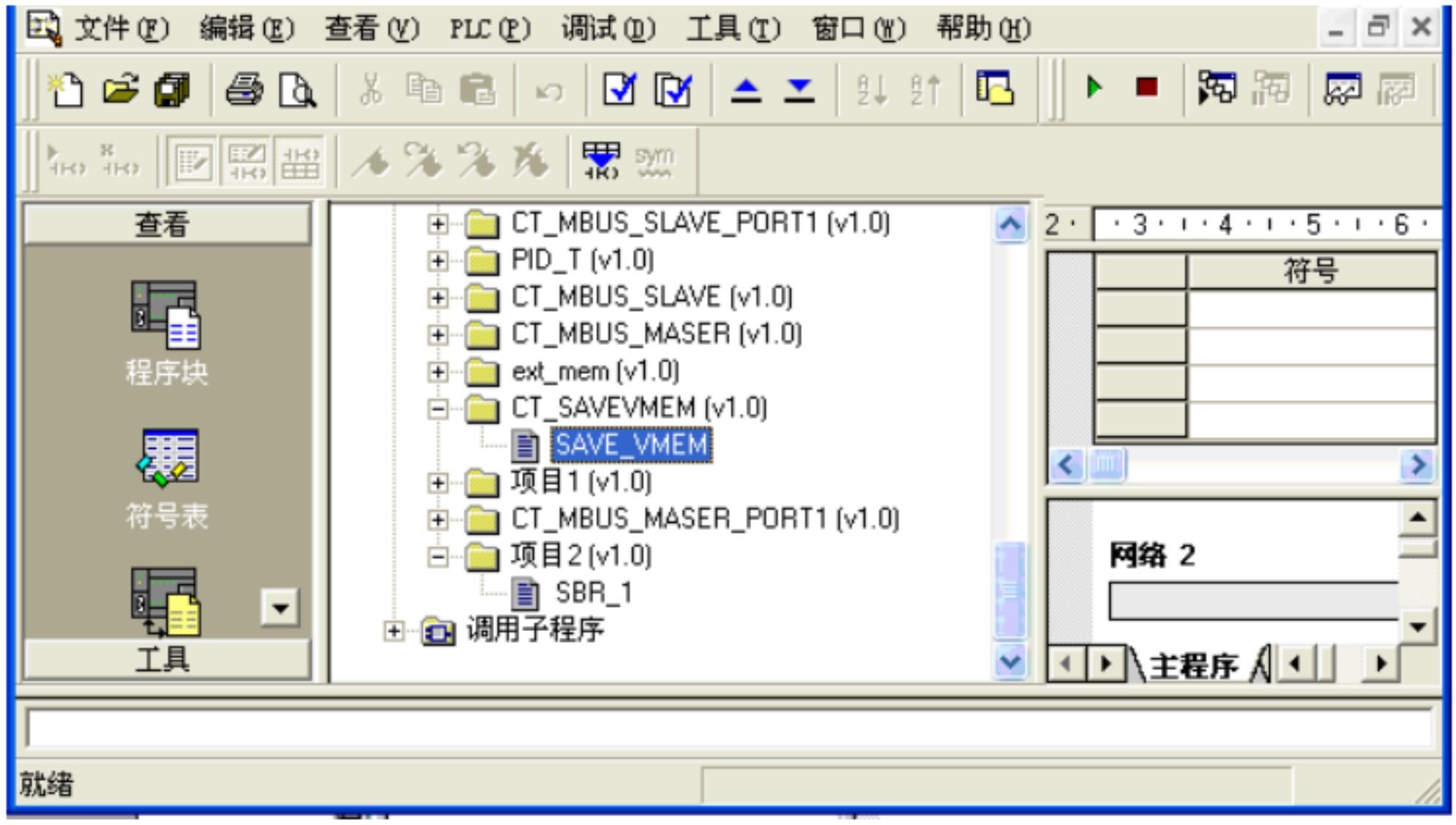
在“ 文件 ” ----“ 添加 /删除库 ”，找到库文件“ ct_savevmem ”,如下图所示。



在弹出的对话框中点“ 添加 ”，找到你存放的“ ct_savevmem.mwl ”文件的位置，找到此文件，选中此文件后点“ 保存 ”，再在“ 添加 /删除库 ”中选中刚添加的“ ct_savevmem.mwl ”文件，点“ 确认 ”。

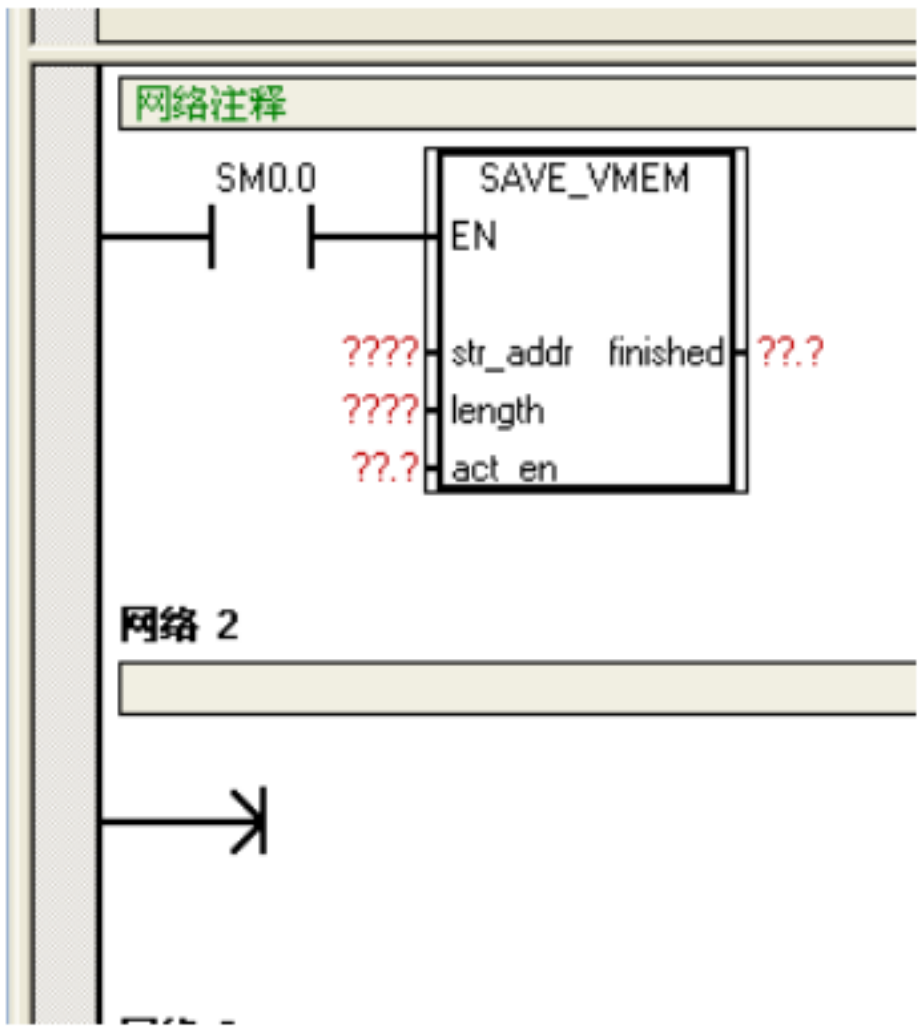


安装成功后，在目录树的“ 库“ 下可以看到新增加的“ ct_savevmem ”库：



6.2.2. 调用 CT_SAVEVMEM 库

点要添加功能块的“ 网络 ”，双击“ 库 ”下面的“ SAVE_VMEM ”,就会在“ 网络 ”里出现相应的功能块，结果如下图所示：



注意：确保在写完成前 EN 一直处于接通状态。

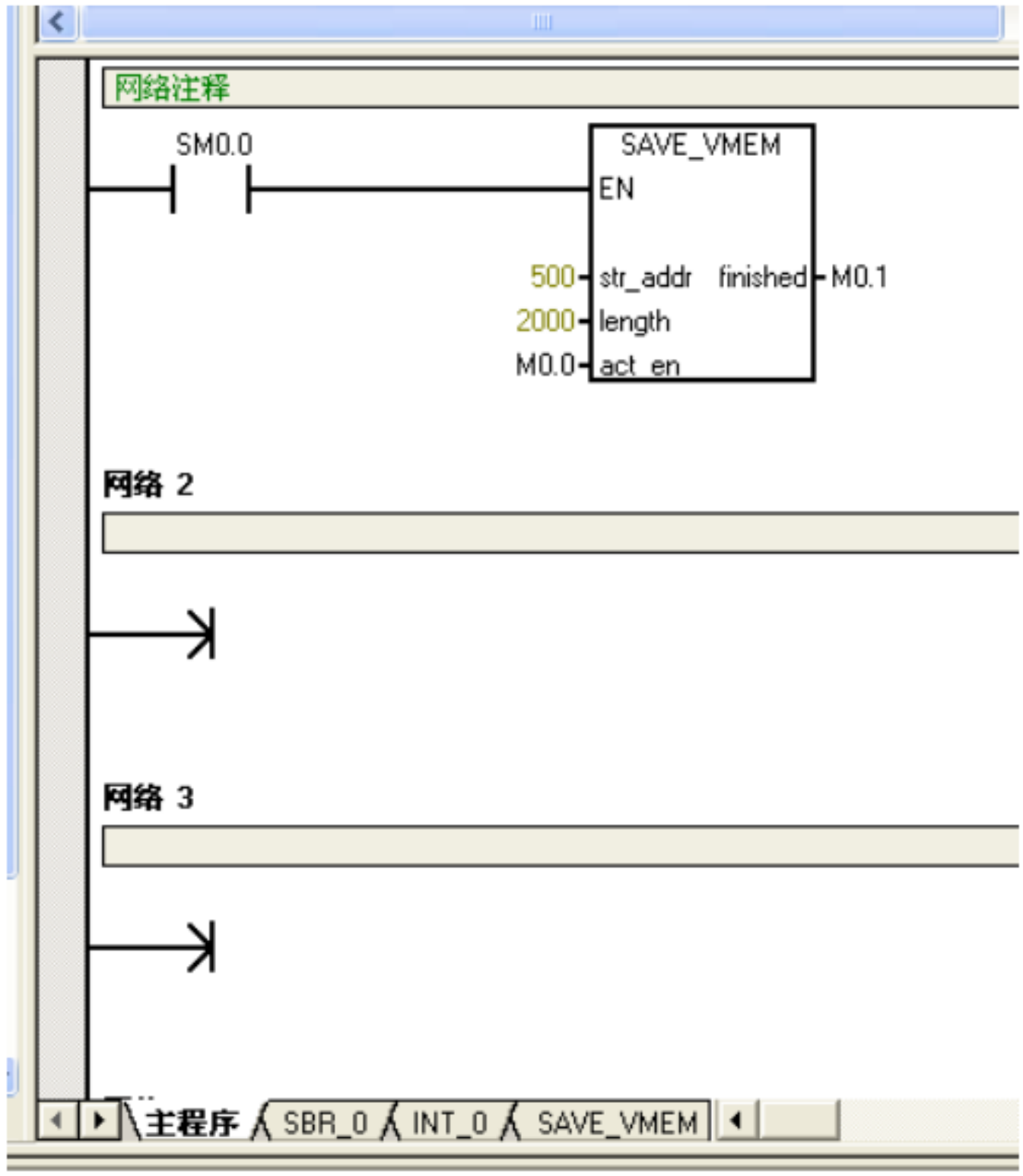
6.3. CT_SAVEVMEM 库功能说明

参数地址	说明	类型	备注
str_addr	V 内存的起始地址	WORD	可以是常量或变量（如 MW0 ）。
length	偏移量（以字为单位）	WORD	要永久保存连续 V 内存的长度。
act_en	写允许位	BOOL	此位为 1 时开始写永久内存，写结束后次位自动复位。
finished	写结束标志位	BOOL	写开始时此位自动复位，写结束时此位自动置位。
注：写的总长度为字的整数倍。			

6.4. 应用实例

应用要求：将 VW500 开始的连续 2000 个字保存到永久内存中， M0.0 作为保存允许位， M0.1 为保存完成位。

应用程序如下：



7. CT PID 控制库的使用

7.1. CPU内嵌的 PID-T 库的使用

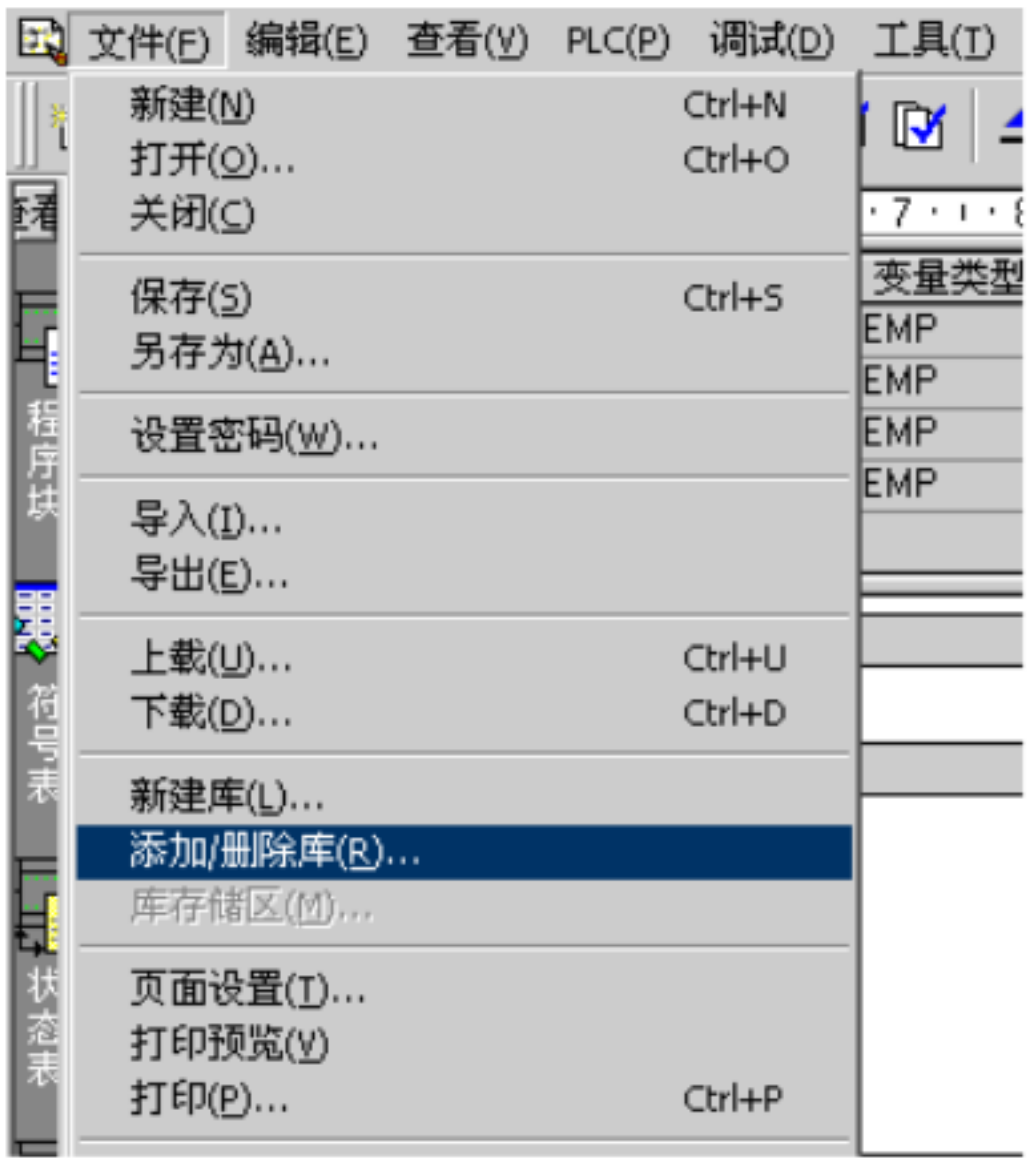
7.1.1. 功能介绍

PID_T 功能块是集成在 CPU 内部，不占用用户程序空间， 作为一个库函数提供给用户使用。 PID_T 主要针对温度控制的智能 PID 功能，带有自整定、自适功能，用户无需复杂编程，只需调用和设置一些简单的参数就可以使用，温度控制准确。

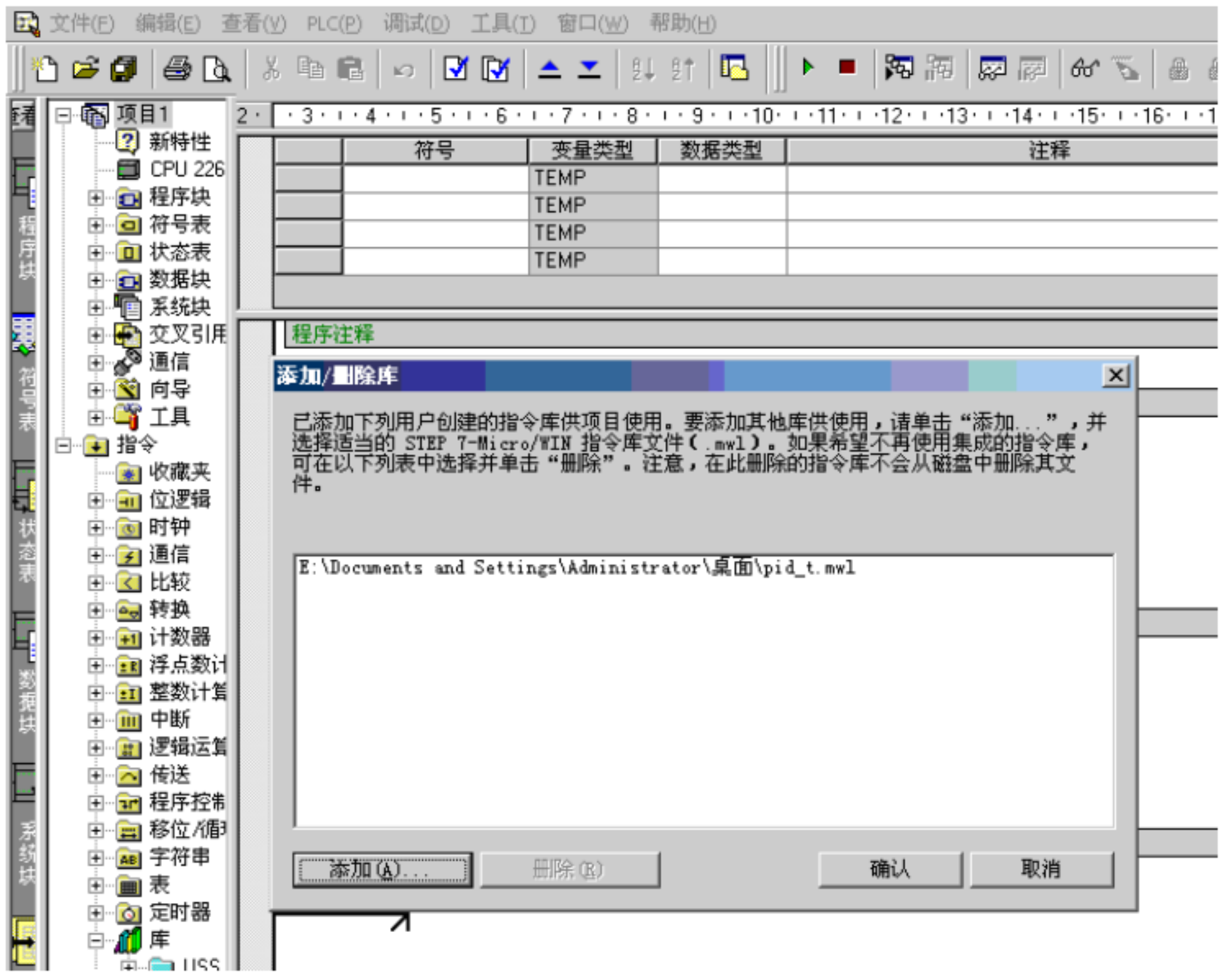
7.1.2. 安装说明

7.1.2.1. 添加库文件

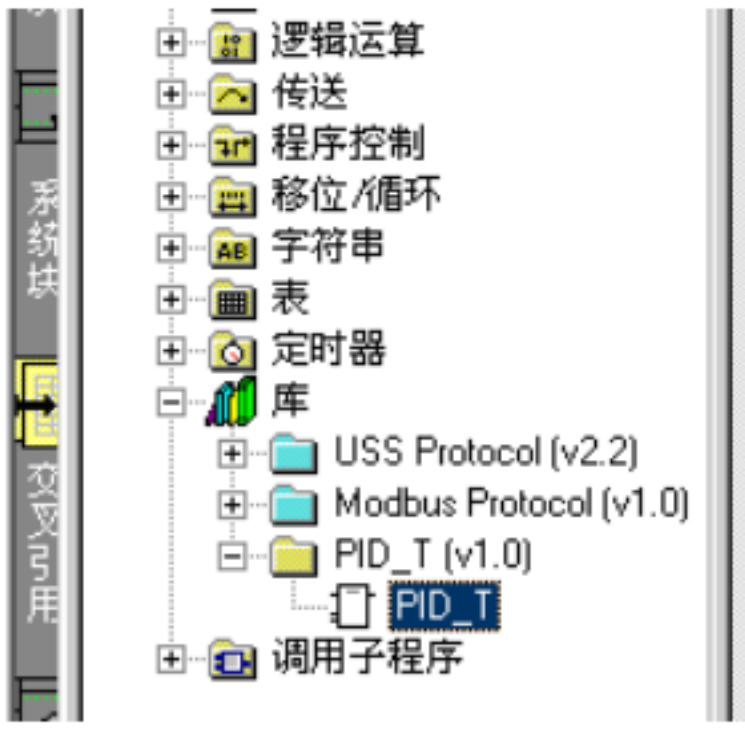
在“ 文件 ” ----“ 添加 /删除库 ”，找到库文件“ pid_t.mwl ”,如下图所示。



在你存放的 pid_t.mwl 文件的位置，找到此文件，如下图所示，点“ 添加“ 按钮。

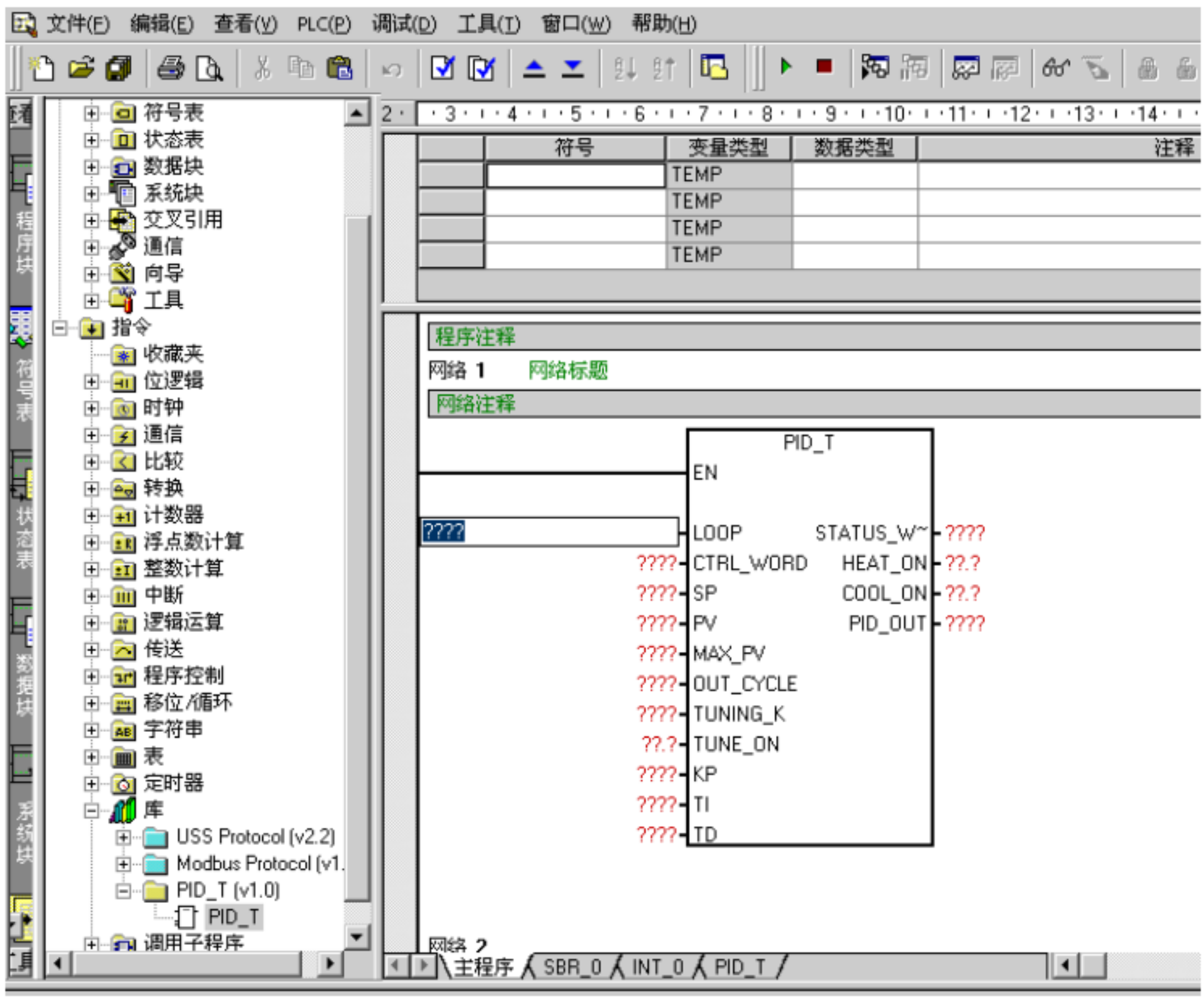


安装成功后，在目录树的“ 库“ 下可以看到新增加的 PID_T 的库：



7.1.2.2. 调用 PID 库

点击要添加功能块的 “ 网络 ”， 双击“ 库 ” 下面的 “ PID_T ”,就会在“ 网络 ” 里出现相应的功能块。结果如下图所示：



7.1.3. PID 库功能说明

7.1.3.1. 地址参数说明

参数地址	说明	类型	数值范围	备注
LOOP	属于第几个回路 PID, 不能重复。从 0 开始。	字，常数或变量	0－63	
CTRL_WORD	控制字，控制 PID 运行	字，常数或变量		常用控制字： 1、16#03(只有加热输出，带自适应功能) 2、 16#07(加热冷却输出，带自适应功能)
SP	设定值	字，常数或变量	-32767－32767 0--65535	
PV	测量值（反馈值）	字，变量	-32767－32767 0--65535	
MAX_PV	测量值的最大值	字，常数或变量	-32767－32767 0--65535	
OUT_CYCLE	脉冲输出周期	字，常数或变量	1--255	单位：秒
TUNING_K	自整定系数	双字，浮点数	0.5---2.0	0.5: 表示滞后很大的系统。 1.0:正常的系统 2.0:
TUNING_ON	启动自整定	位，变量		自整定结束后自动复位
Kp	比例系数	字，整数，常数或变量		如果为常数，不能执行自整定功能
Ｔi	积分时间	字，整数，常数或变量	1--3600	单位：秒 如果为常数，不能执行自整定功能
Td	微分时间	字，整数，常数或变量	0--3600	单位：秒 如果为常数，不能执行自整定功能
STATUS_WORD	状态字	字，变量		运行状态及报警

				状态
HEAT_ON	加热输出	位		
COOL_ON	冷却输出	位		
PID_OUT	PID 模拟输出	字，整数，变量	当定义为只有加热输出时，输出范围为：0—32000，带冷却输出时：-32000---32000	

7.1.3.2. 控制字、状态字位地址

控制字的位地址意义如下：

控制字位	设置	备注
0	0	PID 停止
	1	PID 运行
1	0	积分一直起作用，比例系数 Kp 不自动调整
	1	积分分离及比例系数自动调整
2	0	PID 单极输出
	1	PID 双极输出
3	0	保留
	1	保留
4	0	积分起作用
	1	积分不起作用
5	0	微分起作用
	1	微分不起作用
6		保留
7		保留

状态字位地址的意义：

状态字位	值	备注
0	0	无断线故障
	1	断线故障
1	0	自整定未进行
	1	正在自整定

2	0	无自整定故障
	1	自整定故障
3	0	不加热
	1	正在加热
4	0	不冷却
	1	正在冷却
5	0	PID 停止状态
	1	PID 运行状态
6		保留
7		

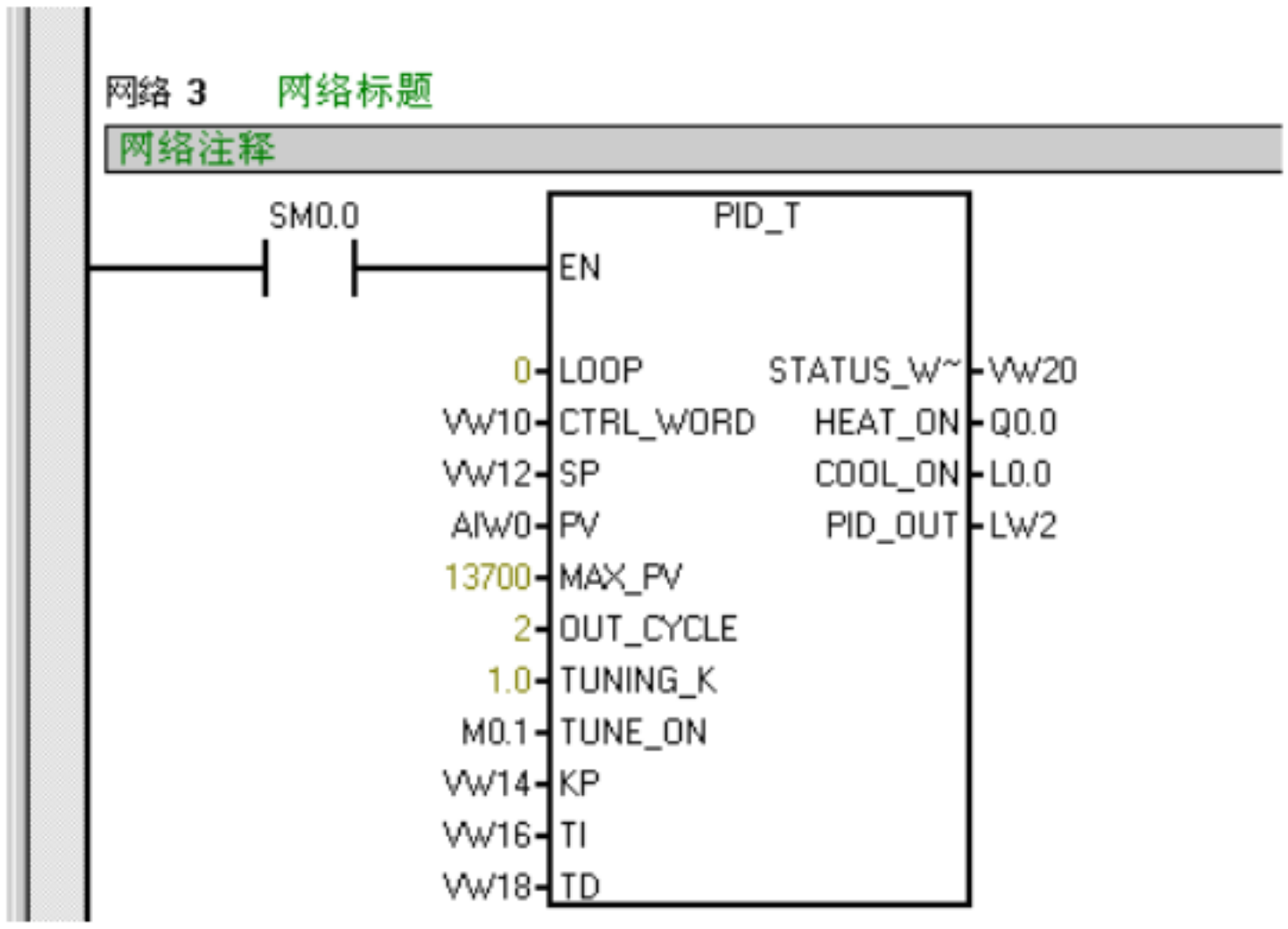
7.1.4. 应用例子

7.1.4.1. 系统需求

在这个例子里，我们以一个温区为例，系统及 I/O 配置表如下：

系统配置	CPU224+231-7PD22	温度采集采用四路热电偶模块	
控制要求	1、 只有加热输出，没有冷却输出 2、 要求自整定参数 3、 K 型热电偶		
I/O 点分布			
Q0.0	加热输出		
AIW0	温度输入	K 型热电偶	
M0.0	PID 运行 /停止位		
M1.0	自整定启动位		

7.1.4.2. 应用程序



7.1.4.3. 程序说明

程序地址参数说明如下：

PID_T 的参数说明			
参数	地址或数值	说明	备注
LOOP	0	因为这是第一个回路，所以是 0	
CTRL_WORD	VW10		
SP	VW12		
PV	AIW0		
MAX_PV	13700	因为 K 型热电偶的最大输入为 13700	
OUT_CYCLE	2	2 秒，这是脉冲输出周期	
TUNING_K	1.0		
TUNING_ON	M0.1	置 1 时开始整定，整定结束后复位	
Kp	VW14	比例系数，整定结束后，整定值自动写到此变量，用户还可以自行调整	
Ｔi	VW16	积分时间，整定结束后，整定值自动写到此变量，用户还可以自行调整	
Td	VW18	微分时间，整定结束后，整定	能

		值自动写到此变量，用户还可以自行调整	
STATUS_WORD	VW20	状态字	
HEAT_ON	Q0.0	加热输出点	
COOL_ON	L0.0	因为没有用到，所以用了一个局部变量	
PID_OUT	LW2	因为没有用到，所以用了一个局部变量	

7.2. CT PID 模块控制库“ em231 pid lib ”的使用

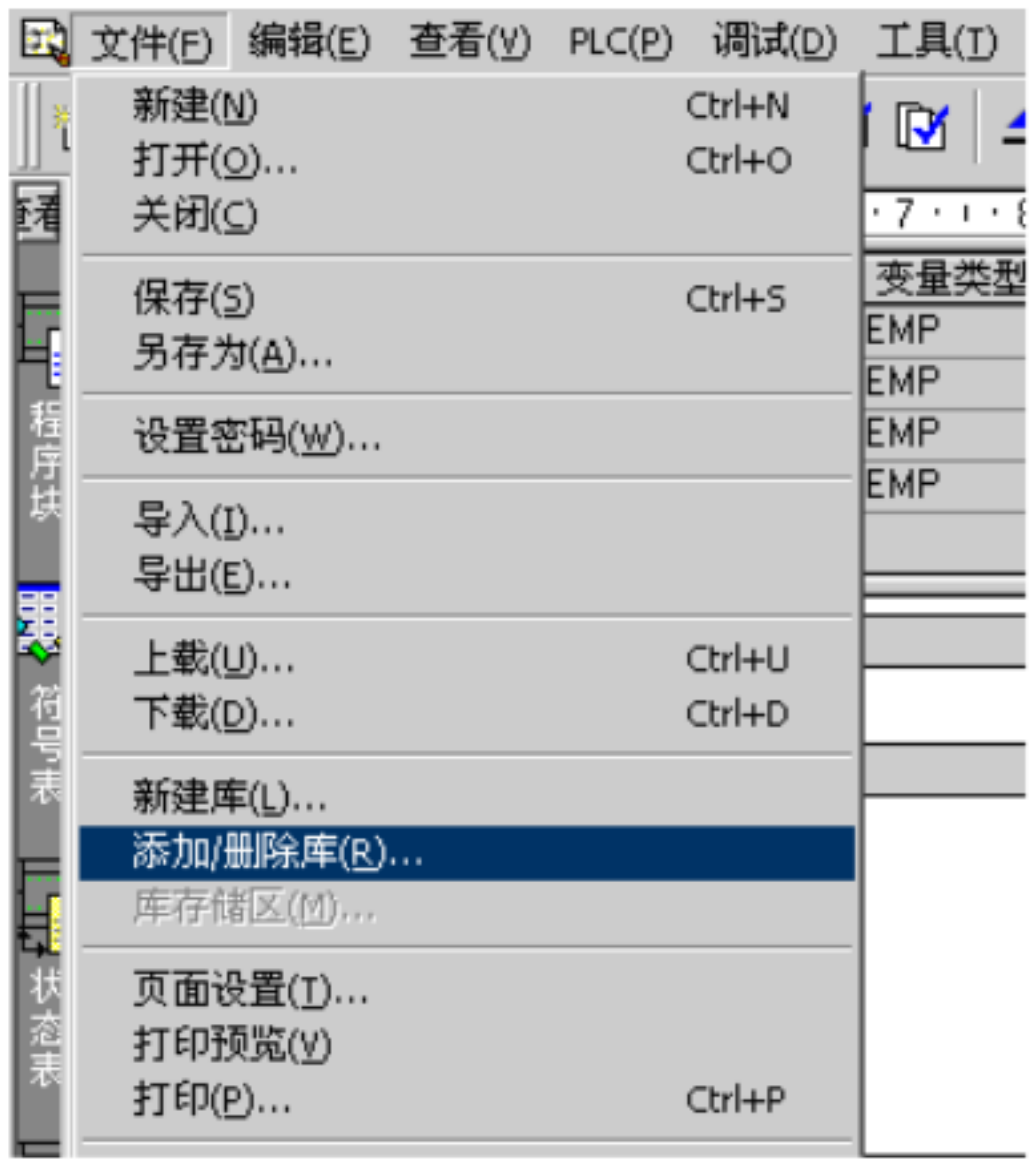
7.2.1. 功能介绍

PID_setting 功能块是专门为 CT 的 PID 模块而提供的，作为一个库函数给用户使用。 PID_setting 主要针对温度控制的智能 PID 模块，用户无需复杂编程， 只需调用和设置一些简单的参数就可以使用，温度控制准确。

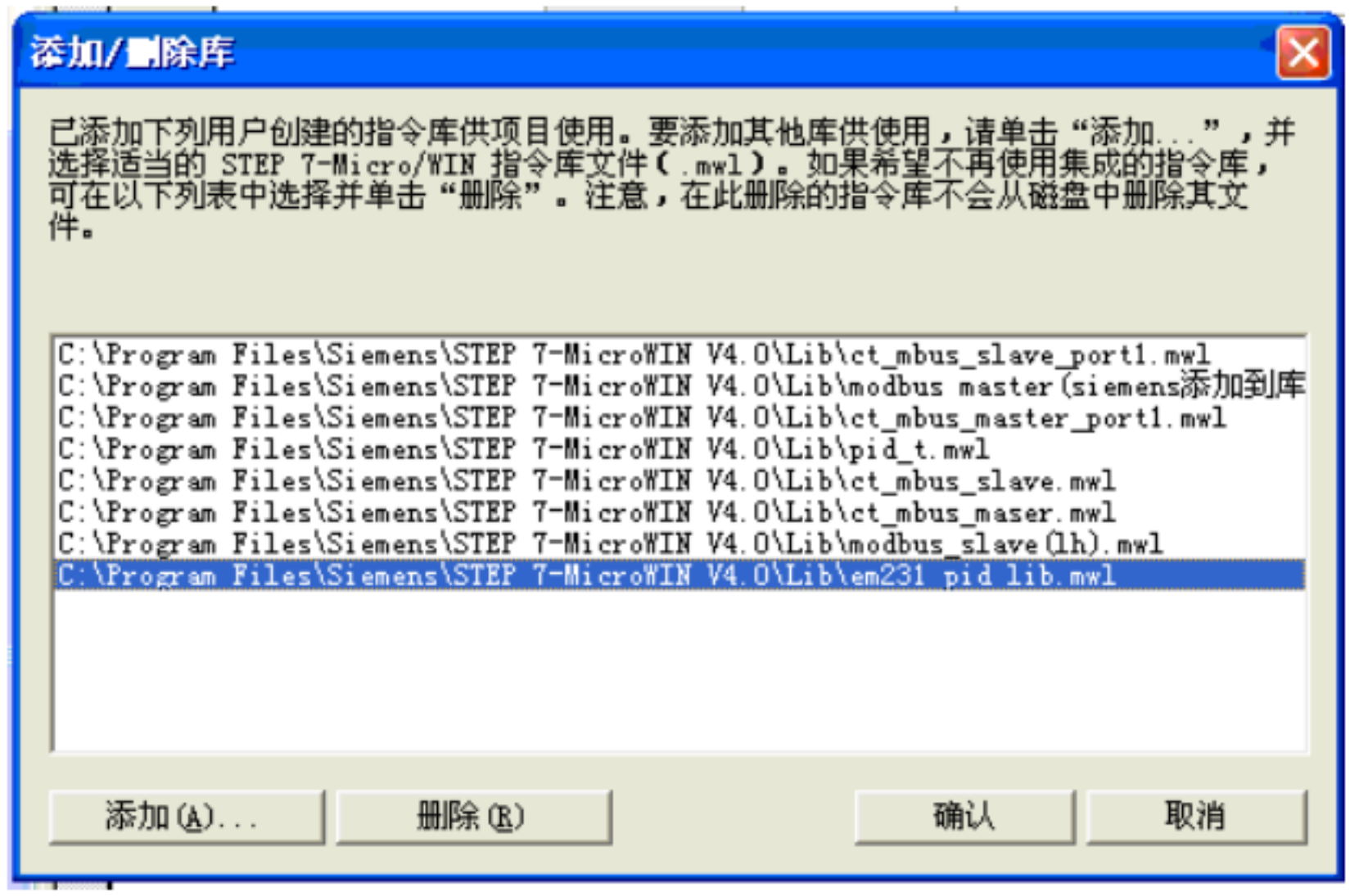
7.2.2. 安装说明

7.2.2.1. 添加库文件

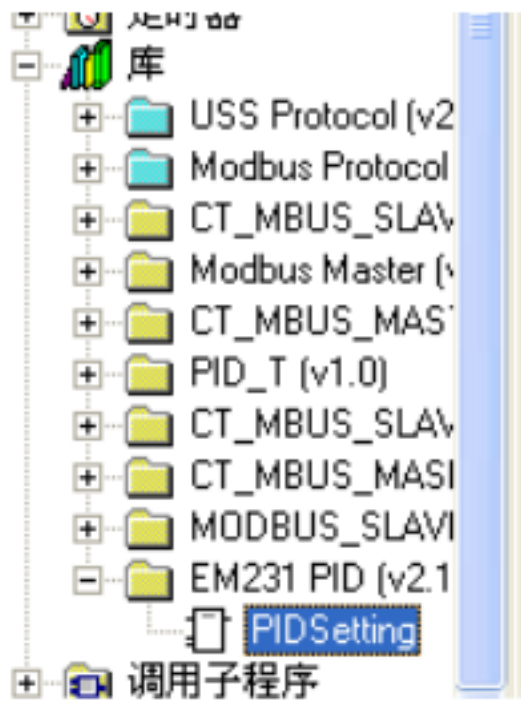
在“ 文件” ----“ 添加 /删除库 ”，找到库文件“ em231 pid lib.mwl ”,如下图所示。



在你存放的“ em231 pid lib.mwl ”文件的位置，找到此文件，如下图所示，点“ 添加“ 按钮。

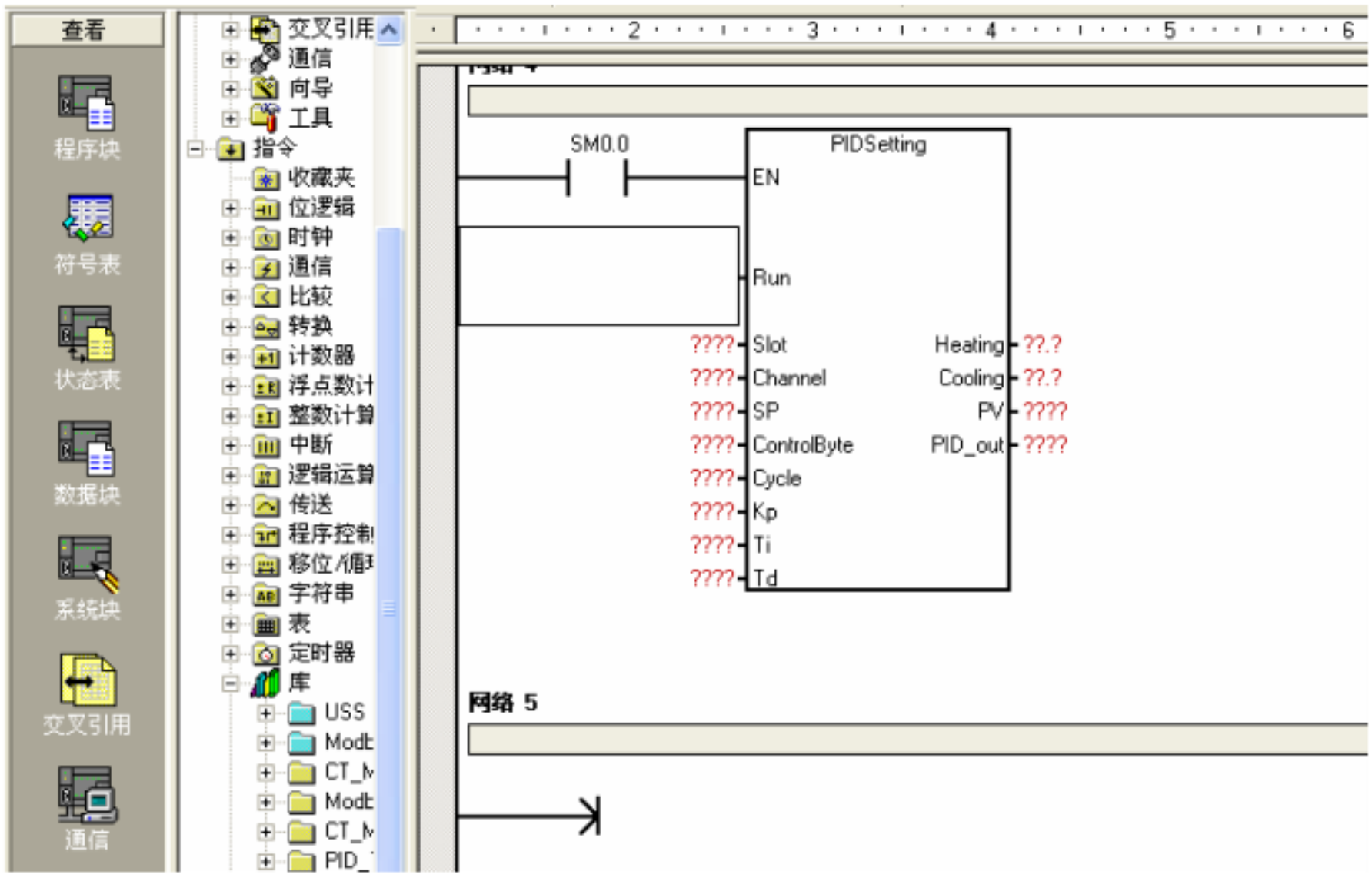


安装成功后，在目录树的“ 库“ 下可以看到新增加的　　PID_setting 的库：



7.2.2.2.　调用 PID 库

点击要添加功能块的“ 网络 ”，　　双击“ 库 ”下面的“　PIDSetting ”,就会在“ 网络 ”里出现相应的功能块。结果如下图所示：



7.2.3. PID 库功能说明

7.2.3.1. 地址参数说明

参数地址	说明	类型	数值范围	备注
Run	运行	位	0 或 1	
Slot	槽号从 0 开始。	字，常数或变量	0 － 6	
Channel	通道号	字，常数或变量	0 － 7	
SP	设定值	字，常数或变量	-32767 ─ 32767 0--65535	
CTRLByte	控制字节，控制 PID 运行	常数或变量		常用控制字节： 1、 16#03(只有加热输出，带自适应功能) 2、 16#07(加热冷却输出，带自适应功能)
Cycle	脉冲输出周期	字，常数或变量	1--255	单位：秒
Kp	比例系数	字，整数，常数或变量		如果为常数，不能执行自整定功能
Ｔi	积分时间	字，整数，常数	1--3600	单位：秒

		或变量		如果为常数，不能执行自整定功能
Td	微分时间	字，整数，常数或变量	0--3600	单位：秒 如果为常数，不能执行自整定功能
Heating	加热输出	位		
Cooling	冷却输出	位		
PV	测量值（反馈值）	字，变量	-32767—32767 0--65535	
PID_out	PID 模拟输出	字，整数，变量	当定义为只有加热输出时，输出范围为：0—32000，带冷却输出时：-32000---32000	

7.2.3.2. 控制位地址

控制字的位地址意义如下：

控制字位	设置	备注
0	0	PID 停止
	1	PID 运行
1	0	积分一直起作用，比例系数 Kp 不自动调整
	1	积分分离及比例系数自动调整
2	0	PID 单极输出
	1	PID 双极输出
3	0	保留
	1	保留
4	0	积分起作用
	1	积分不起作用
5	0	微分起作用
	1	微分不起作用
6		保留
7		保留

7.2.4. 应用例子

7.2.4.1. 系统说明

本例设置第 1 个扩展模块（插槽号为 0）上的 EM231PID 模块第 1 个 PID 回路（通道号为 0）的参数。调用 PIDSetting 来设定该回路的参数, 无需计算 PID 参数地址, 只需输入回路所在的插槽号和通道号，再使能 Run 来运行该回路。

Q0.0 为正向脉冲输出, Q0.1 为负向脉冲输出。

VW0为实际温度 , VW2为 PID 模拟量输出。

修改 PID 设定参数使用其他地址,

设定温度： VW120,

控制字： VB122,

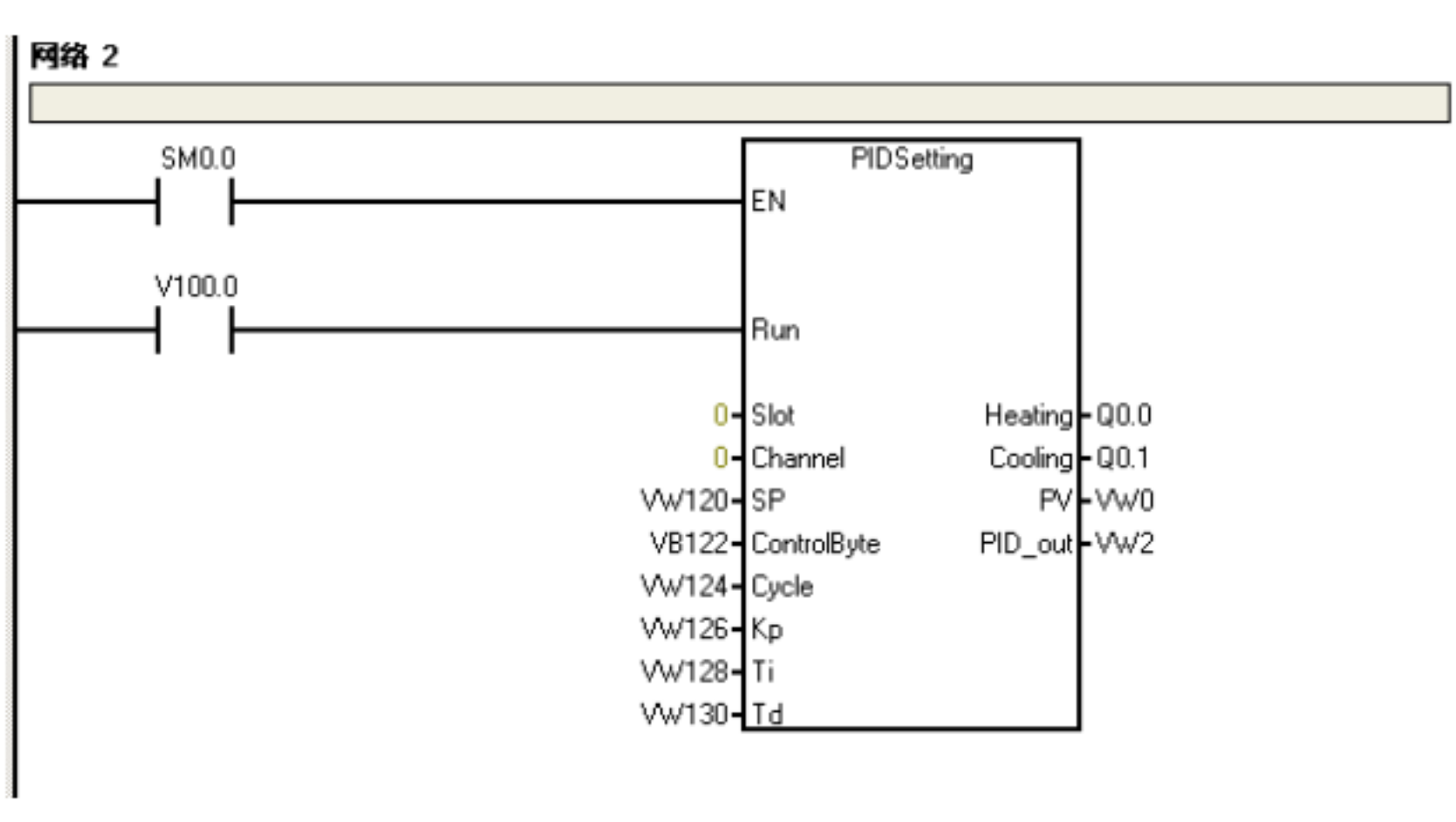
脉冲输出周期： VW124,

比例系数： VW126,

积分时间： VW128,

微分时间： VW130

7.2.4.2. 应用程序



8. CT 扩展 CPU 程序空间和增强程序保密性库的使用

8.1. 功能介绍

动态库功能块是 CTS7200CPU 为扩大用户编程程序空间和增加程序保密性给用户提供的一个特殊功能。 CTS7200CPU 动态库是提前下载到 PLC 中，应用程序下载时再编译到程序中的功能独立的程

序块。 CTS7-200CPU 可加载两个大小各个 24K 的动态库（“ ct_lib1 ”和“ ct_lib2 ”)。

8.2. 使用说明

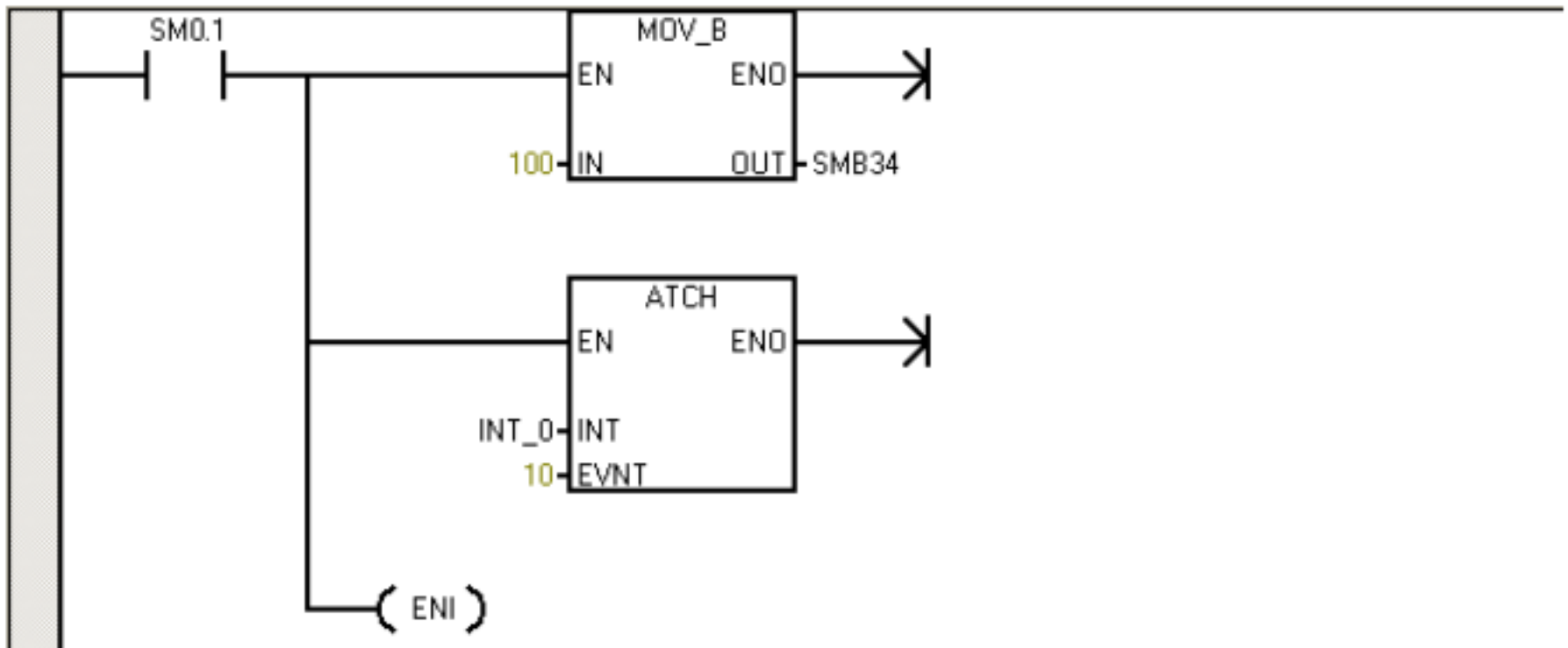
8.2.1. 动态库的使用范围

CTS7200CPU 最多可加载两个动态库 (“ ct_lib1 ”和“ ct_lib2 ”)，不同类型的 CPU 所支持的动态库及动态库的大小如下表所示：

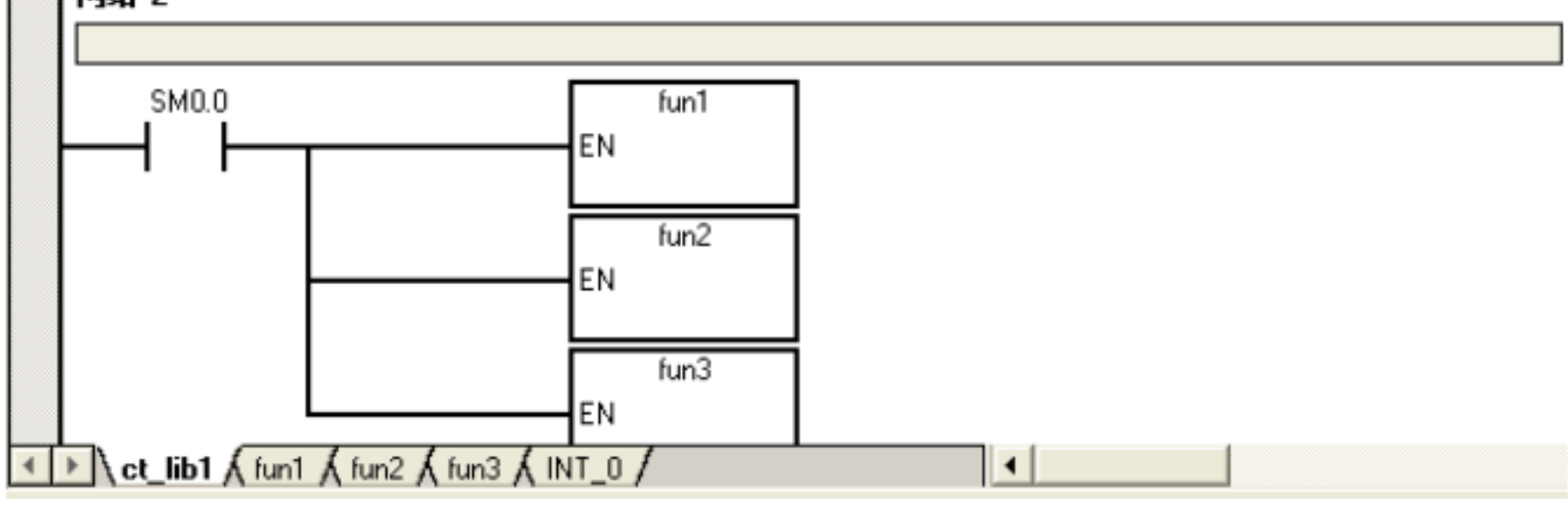
CPU 型号	ct_lib1	ct_lib2
CPU224+	允许 4K	不支持
CPU226M/226L/226H	允许 24K	允许 24K
PSC266T	允许 24K	允许 24K

8.2.2. 创建动态库

在工程中，创建所有要作为动态库的程序块，将主程序块的名称命名为 ct_lib1 或 ct_lib2，下载到 PLC 中，在 PLC 中就生成了库函数包括工程中的所有子程序块的动态连接库。



符号	地址	注释
INT_0	INT0	中断程序注释

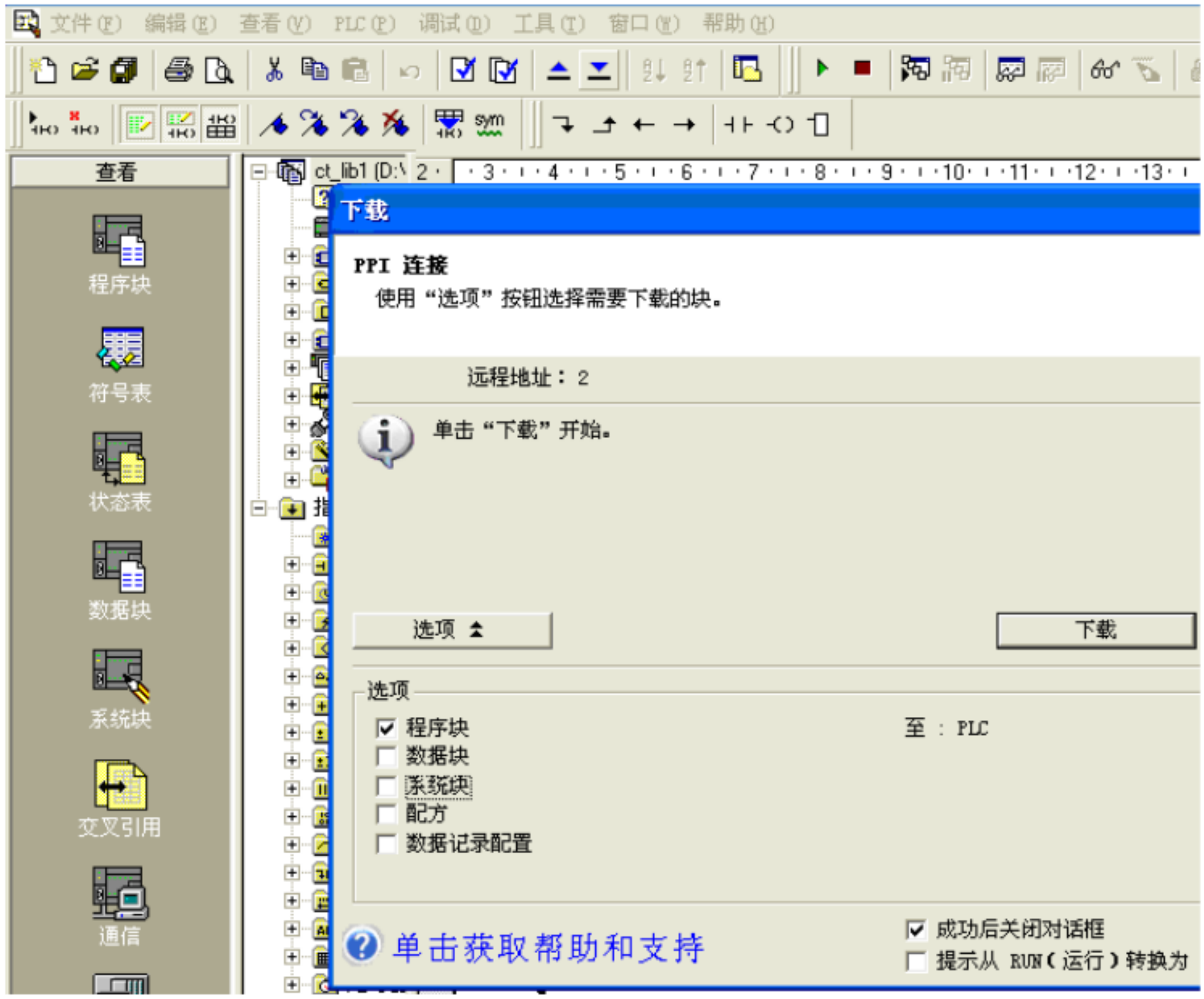


8.2.3. 下载动态库

将编辑好的动态库程序下到 PLC 中（如下图所示），在 PLC 中就生成了库函数包括工程中的所有子程序块的动态连接库。

每次成功下载了新的动态库之后, PLC 中原来的库和程序块完全被清除。 PLC 中生成了名为 ct_lib1 的动态库。

注意：在下载动态库时确保只下载程序块。

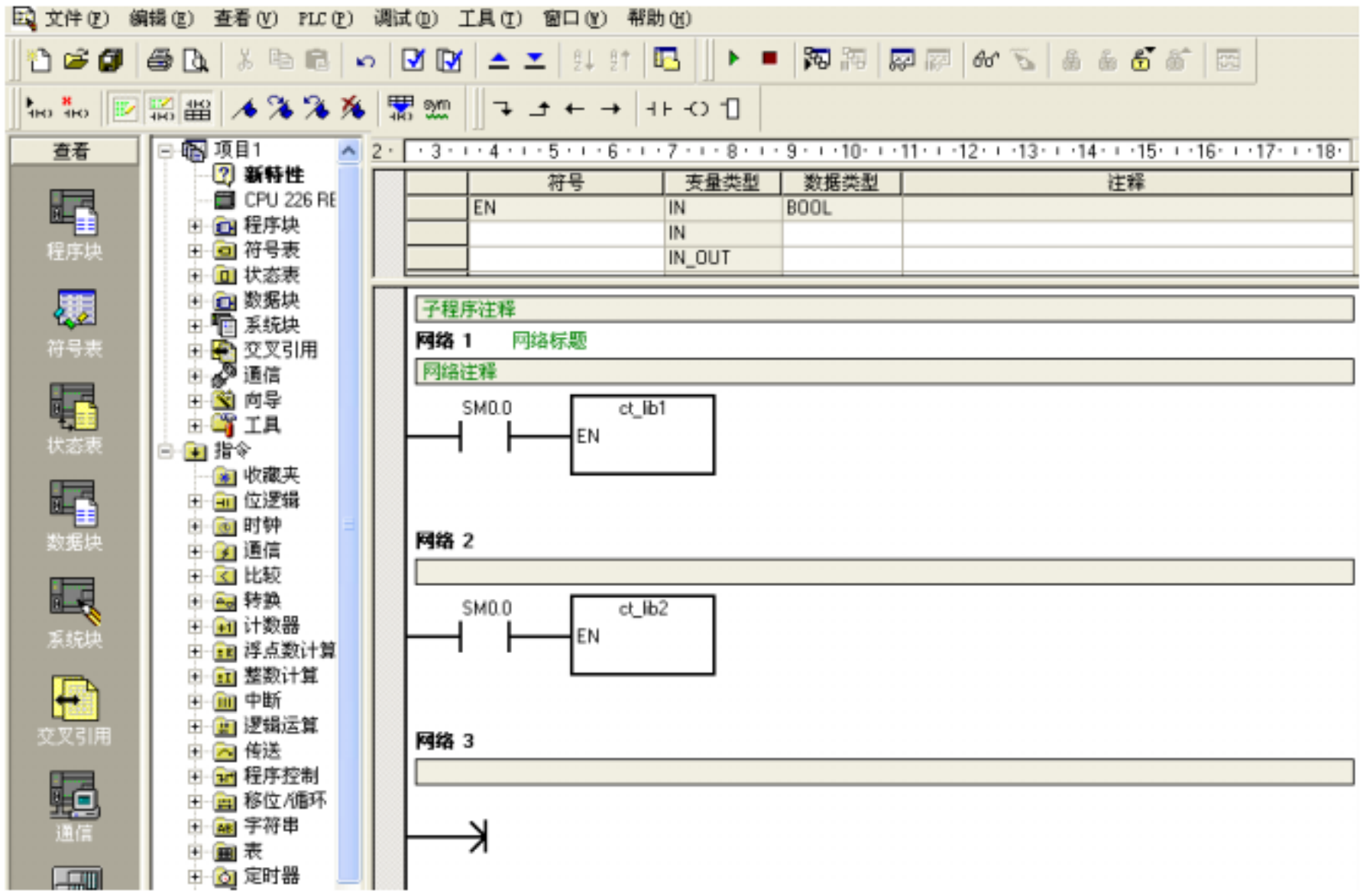


同样的流程可以下载名为 ct_lib2 的动态库到 PLC 中。

8.2.4. 使用动态库

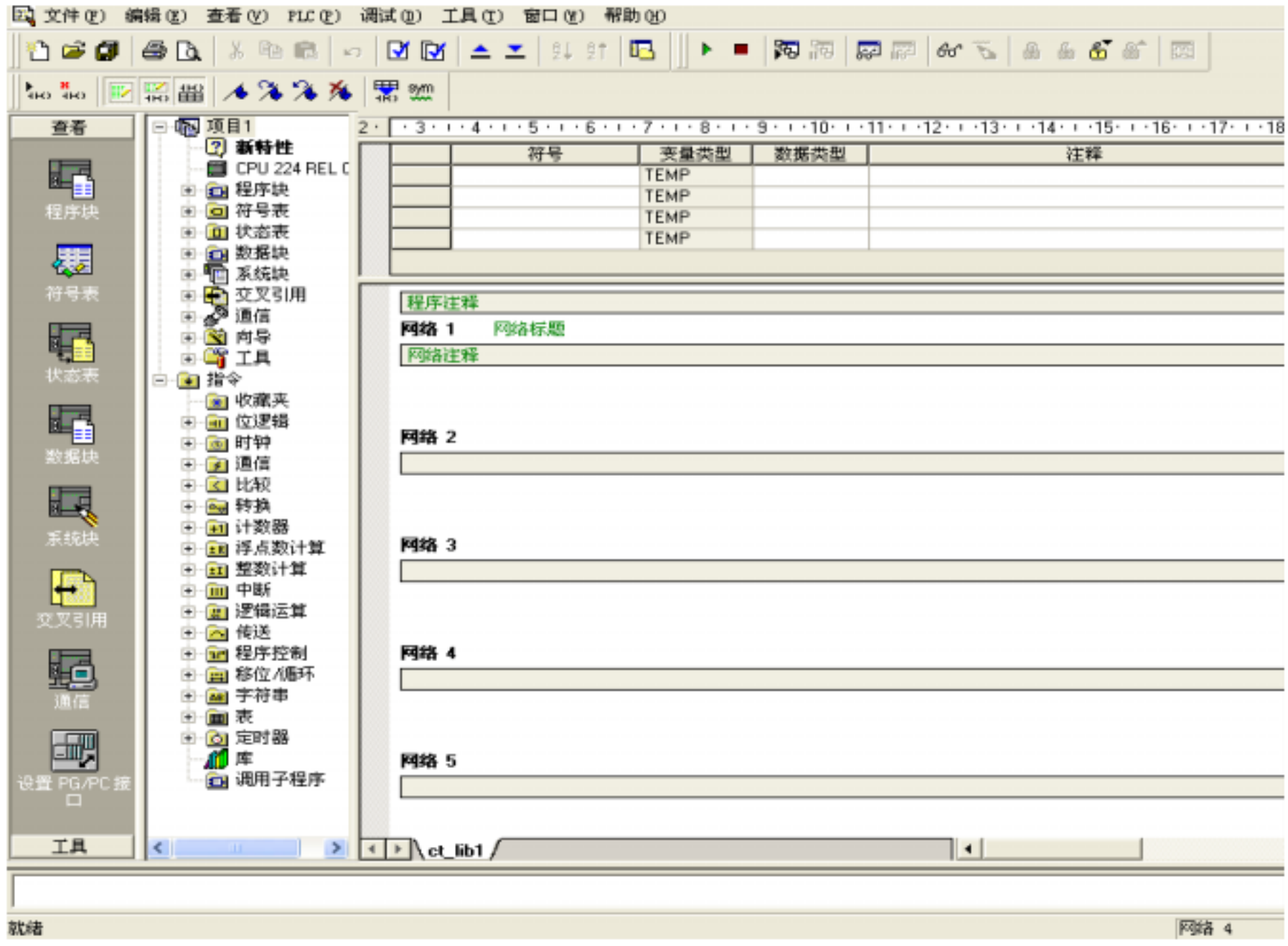
在工程中，创建和已经下载到 PLC 中的动态库同名的空的子程序块，如 ct_lib1 或 ct_lib2，在工程中可以调用这些子程序块，程序下载到 PLC 时，工程中的空程序块被替换成同名相应的库函数，运行时执行相应动态库的程序。

注意：使用时最好先加载动态库，然后再加载使用动态库的程序。



8.2.5. 清除动态库

下载新的动态库时， PLC 中原有的动态库被清除。下载一个主程序名称为 ct_lib1 或 ct_lib2 的空的程序块的工程到 PLC 中， PLC 中的相应的动态库就被完全清除。



9. CT CPU 远程维护的使用

9.1. 简介

本章节是用来指导读者使用 RS-232/PPI 多台主设备电缆（ RS232/PPI Multi-Master Cable ）通过 moden 以及电话网络，用本地 PC 运行的 STEP7-Micro/WIN 实现对远程的 PLC 进行维护（包括下载、上载等功能）的方法。

要实现这种远程维护，需要的资源：

远程端：一个 moden、一个被维护的 CPU、一台装有 RS232 串口且按装了西门子“ STEP MICRO/WIN ”软件的 PC 机。

本地端：一个 moden，一台装有 RS232 串口且按装了西门子“ STEP MICRO/WIN ”软件的 PC 机。

完成 CPU 远程维护要有两个方面的配置：

1. 对远程端 RS-232/PPI 多台主设备电缆的配置。
2. 对本地 PC 机上运行的“ STEP7-Micro/WIN ”软件的配置。

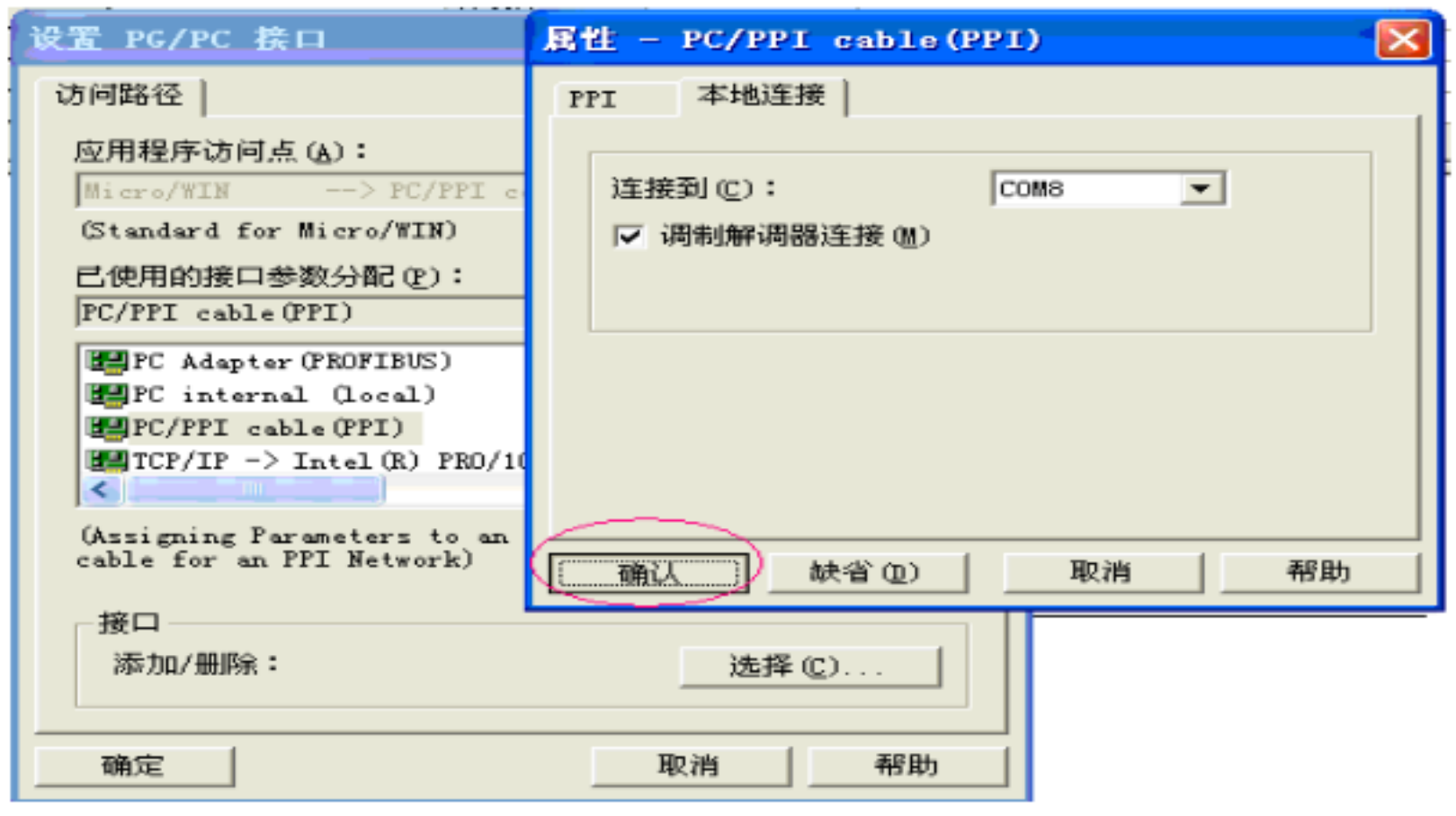
9.2. 配置操作

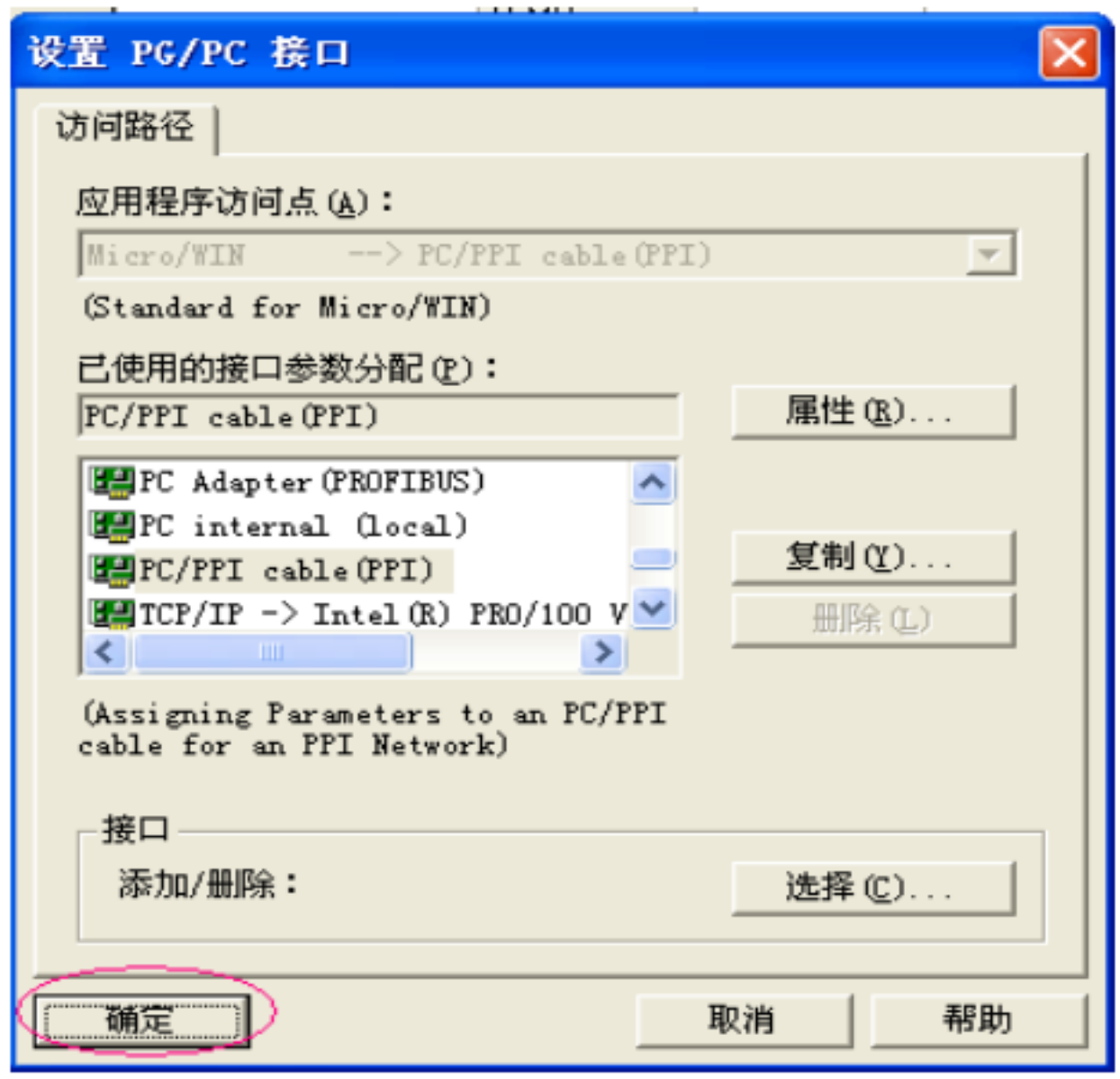
9.2.1. 对远程端 RS232-485 PC/PPI 多主设备电缆的配置

9.2.1.1. 初始化配置

1） 选择初始化 PC/PPI 电缆的 COM 口

打开 STEP7-Micro/WIN 的“ 设置 PG/PC 接口 ”，设置电缆和 PC 机连接的接口参数为“ PC/PPI cable（PPI） ”，点击右边的“ 属性 ”按钮，在弹出属性对话框中选择电缆要连接的串口，并选上“ 调制解调器连接 ”， 先后 2 次按 “ 确认 ”完成初始化配置。如下图所示：

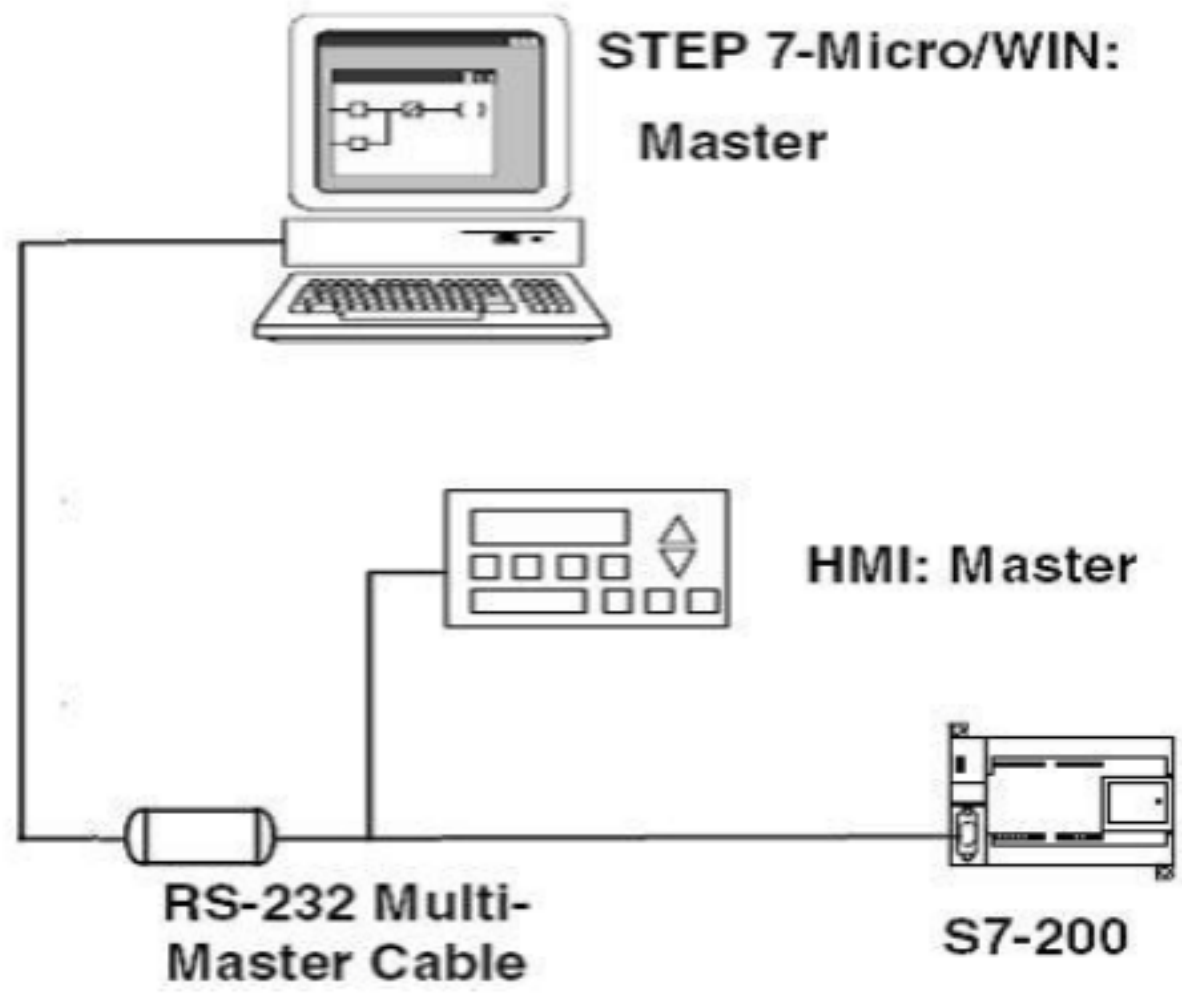




2） 连接好 PC/PPI 电缆

串口配置完后，将 PPI 电缆的拨码开关配置成 

（白色为拨码开关位置），再把电缆一端连接到刚配置的 PC 机的串口上，另一端连接到 CPU 的 485 口上（接 cpu 的目的是让电缆通过 CPU加电），让 CPU 上电，如下图所示：

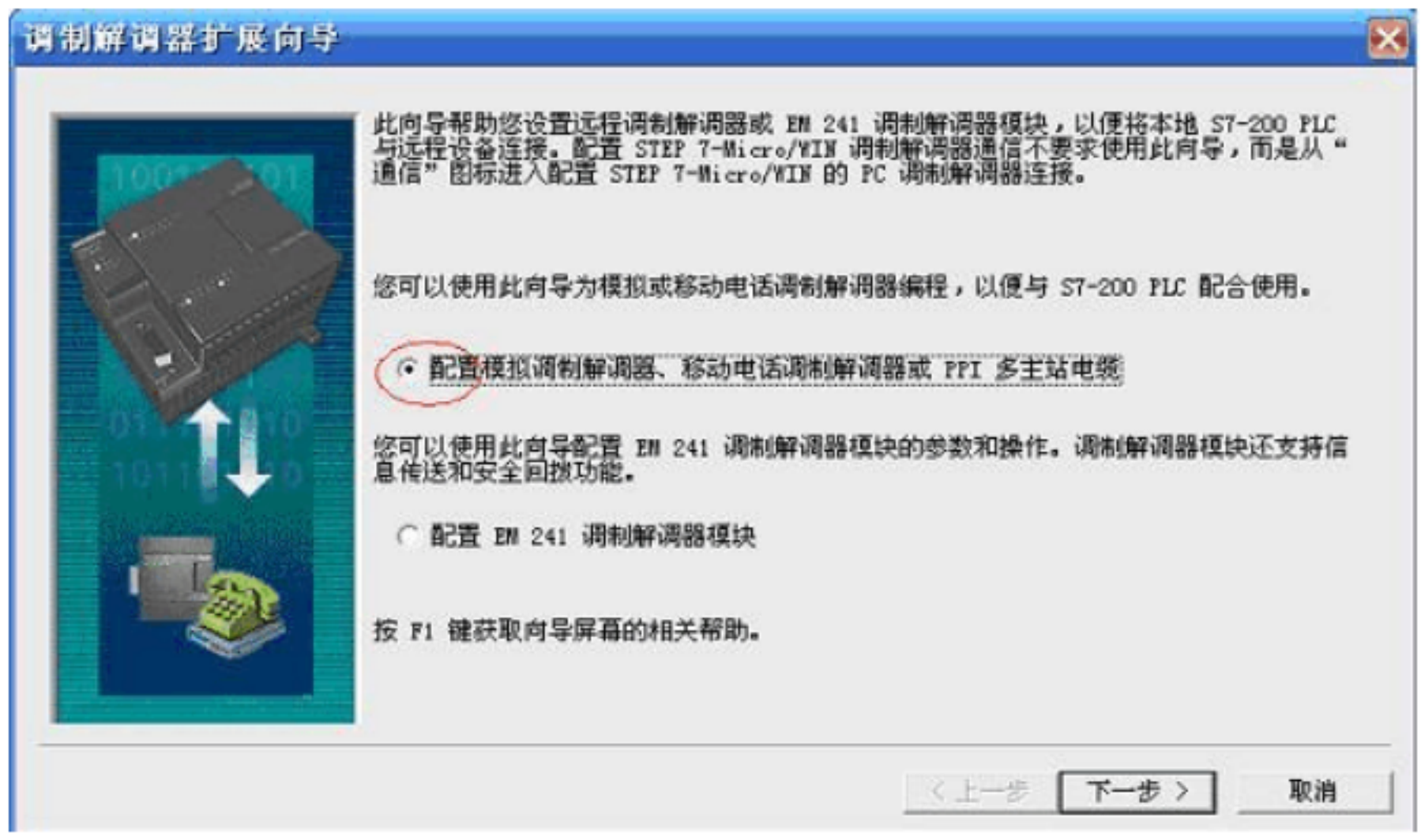


图中的 HMI ：Master 可以是其它主站，也可以是多个主站，也可以没有。

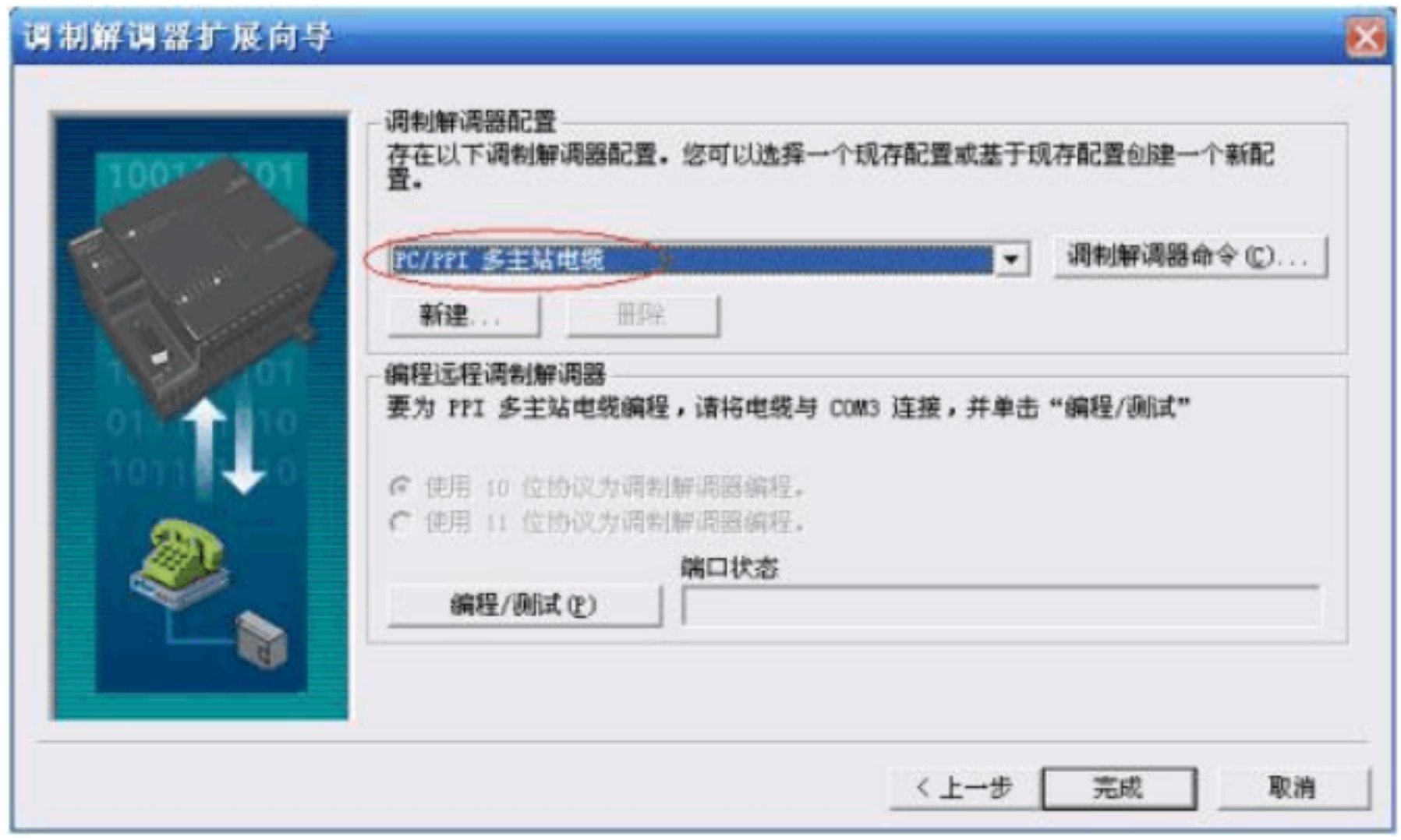
3） 配置“ 远程调制解调器 ”向导

连接好 PPI 电缆后，在向导中配置“ 远程调制解调器 ”向导，如下图所示：

进入“ 调制解调器扩展向导 ”，选中“ 配置模拟调制解调器、移动电话调制解调器或 PPI 多主站电缆 ”项，如下图红圈所示：



单击“ 下一步 ”配置调制解调器（如下图），选从下拉框中选中“ PC/PPI 多主站电缆 ”（如下图红圈所示）。

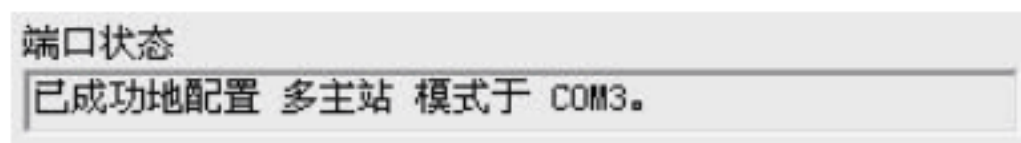


选中“ PC/PPI 多主站电缆 ”单击右边的“ 调制解调器命令 ”按钮。进入下图所示的对话框。

选中红圈圈出的“ 多主站 ”模式，下面的 AT 命令采用默认方式，点击确认。

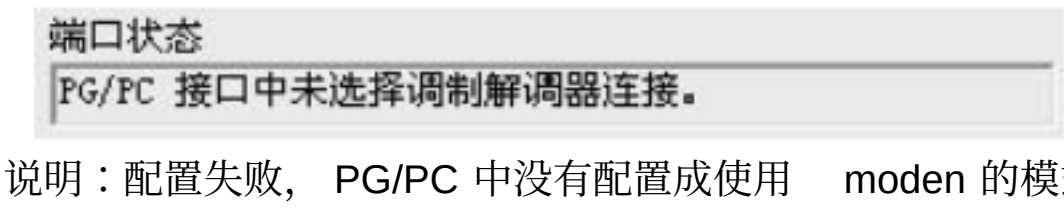
4） 测试配置的 PC/PPI 电缆

完成了 PC/PPI 多主站电缆的模式配置后，回到前面调制解调器配置画面，单击“ 编程 / 测试 ”按钮， 对配置的 PC/PPI 多台主设备电缆进行测试。 当配置成功或者失败时会在右边的文本框中提示不同的信息。分别如下：



说明：配置成功（ COM3 会随读者所使用的串口而变化）。

说明：配置失败，造成的可能原因有， 1：电缆的拨码开关组配置不对； 2：电缆于 PC 的连接有问题，检查电缆的拨码开关组及和 PC 串口的连接。



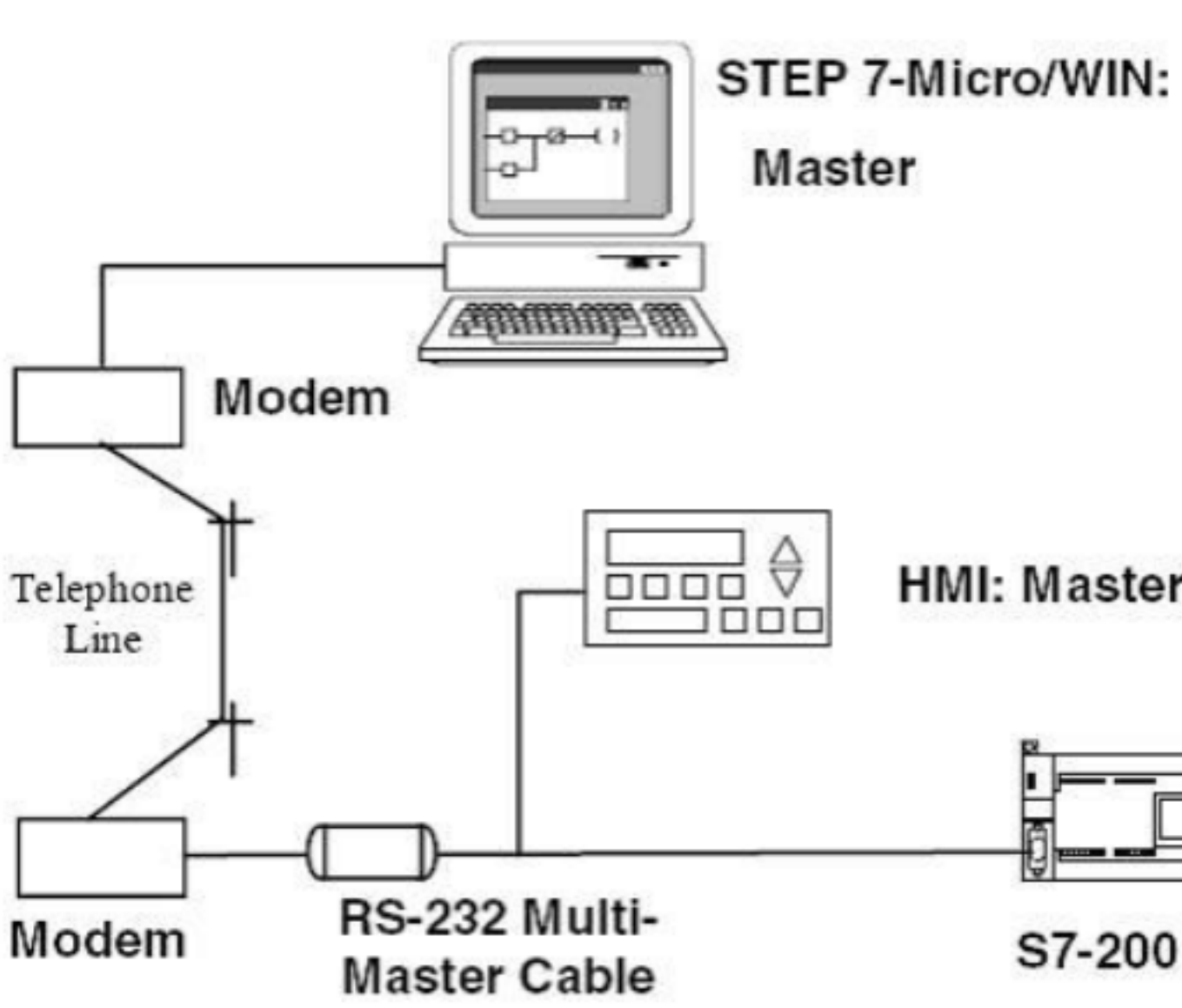
说明：配置失败， PG/PC 中没有配置成使用 moden 的模式， 重新配置“ 设置 PC/PG 接口 ”。

配置成功后，初始化配置完成。

9.2.1.2. 使用配置

初始化配置成功后， 将 PC/PPI 电缆断电， 并将从 PC 机串口上取下的一端连接到 moden 串口上, 连接好后让 PC/PPI 电缆和 moden 上电，再将 PC/PPI 电缆的拨码开关组配置成 

。（白色为拨码开关位置），连接如下图所示：



图中的 HMI ： Master 可以是其它主站，也可以是多个主站，也可以没有。

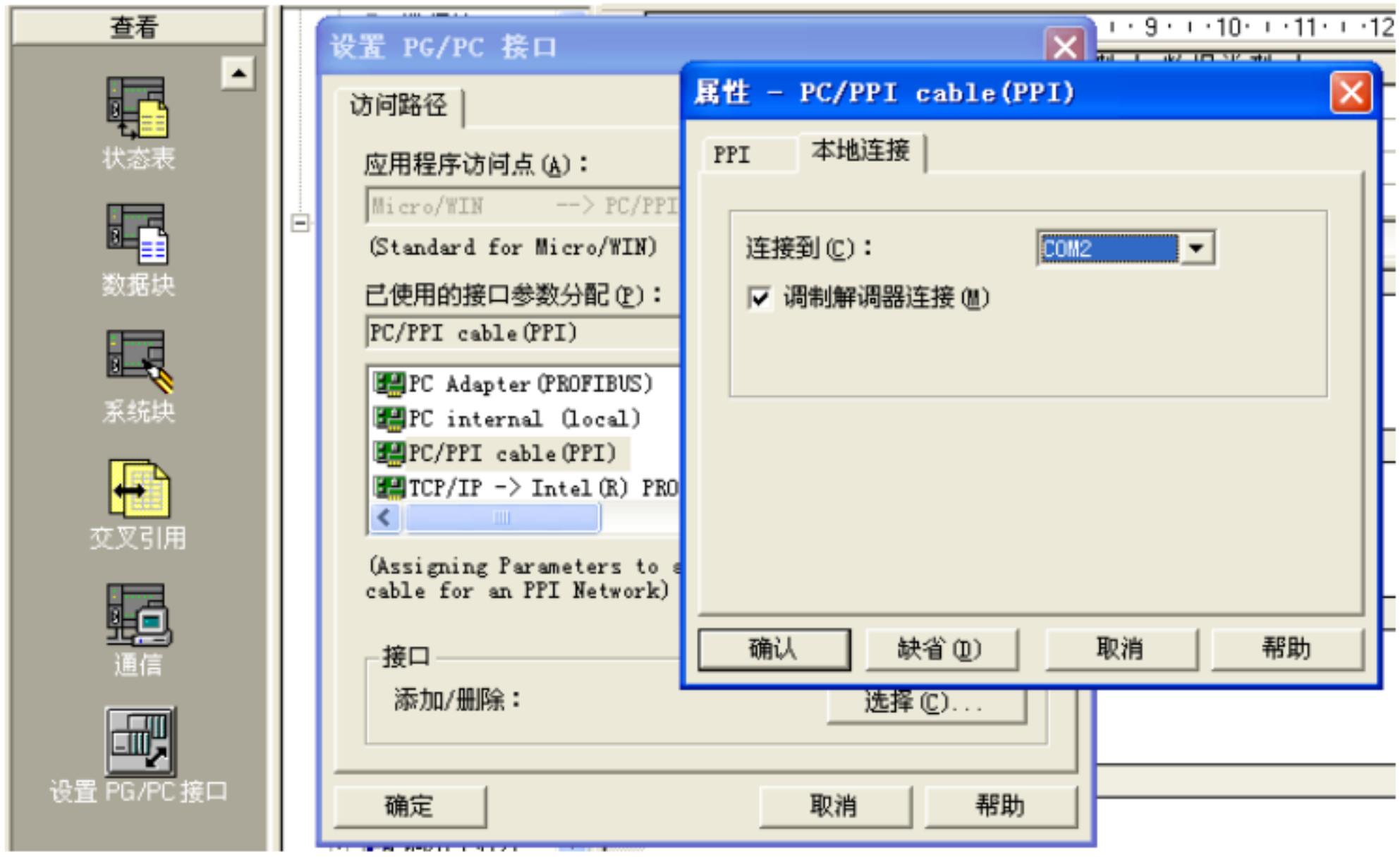
注意： RS--232/PPI 多台主设备电缆在每次加电时，将会配置调制解调器。请确保在电缆加电之前或几乎在同一时间时，调制解调器也已加电。此外，如果调制解调器循环加电，则确保电缆也循环加电。这允许电缆正确配置调制解调器，并以可用的最高波特率运行。

9.2.2. 对本地 PC调制解调器配置连接

要求最好先安装好 moden 的驱动。

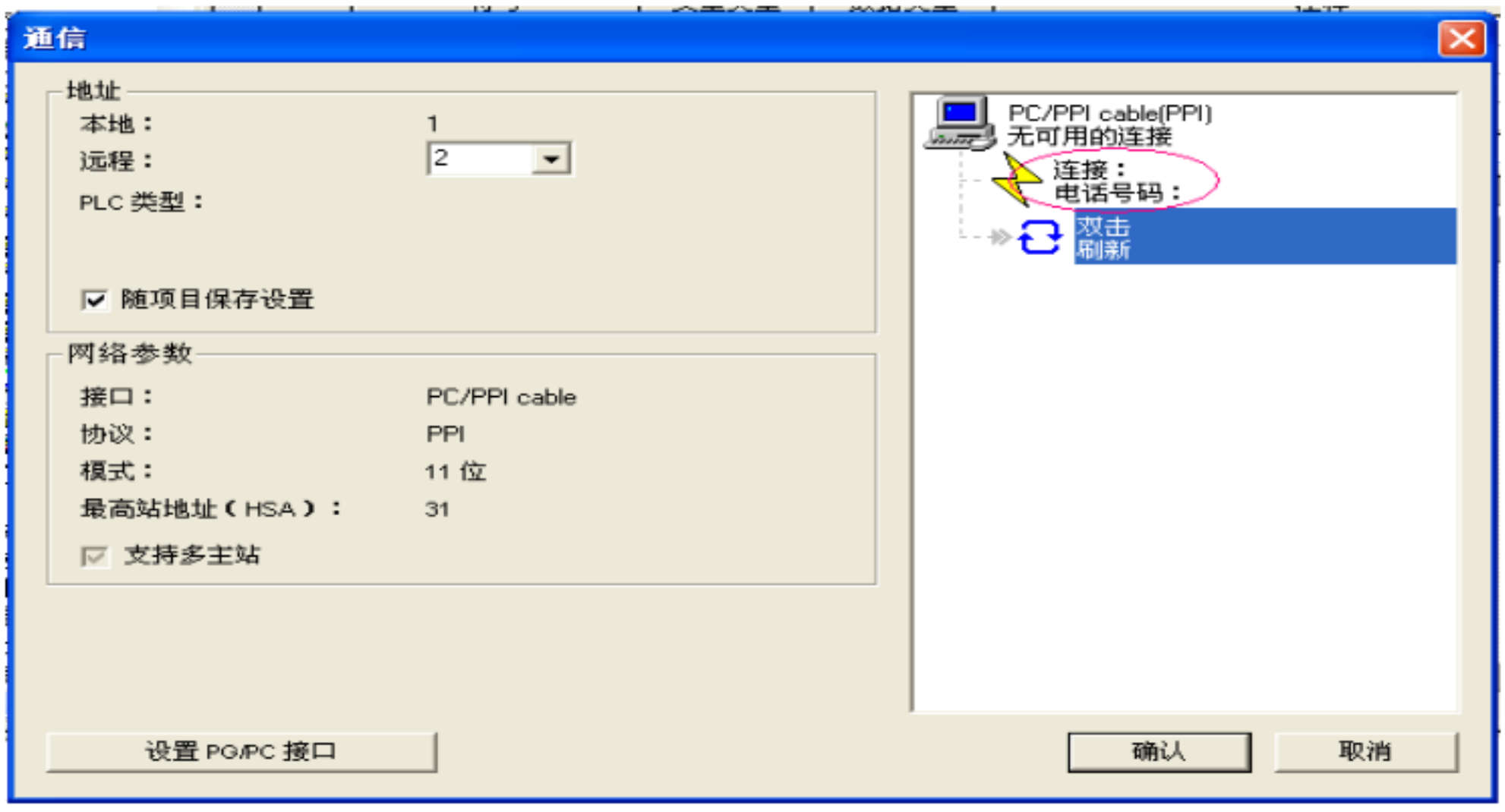
1） 配置 moden 和 PC 连接的串口

通过在配置 STEP7-Micro/WIN 使用那一个串口的对话框中选中 Moden connection 来配置 STEP7-Micro/WIN 使用 moden 模式，如下图所示。



2） 创建 moden 和 PC 的连接。

首先打开通信对话框时就会在右边出现如下图所示：



双击上图中红圈位置，进入　　moden 的连接配置对话框，如下图所示：

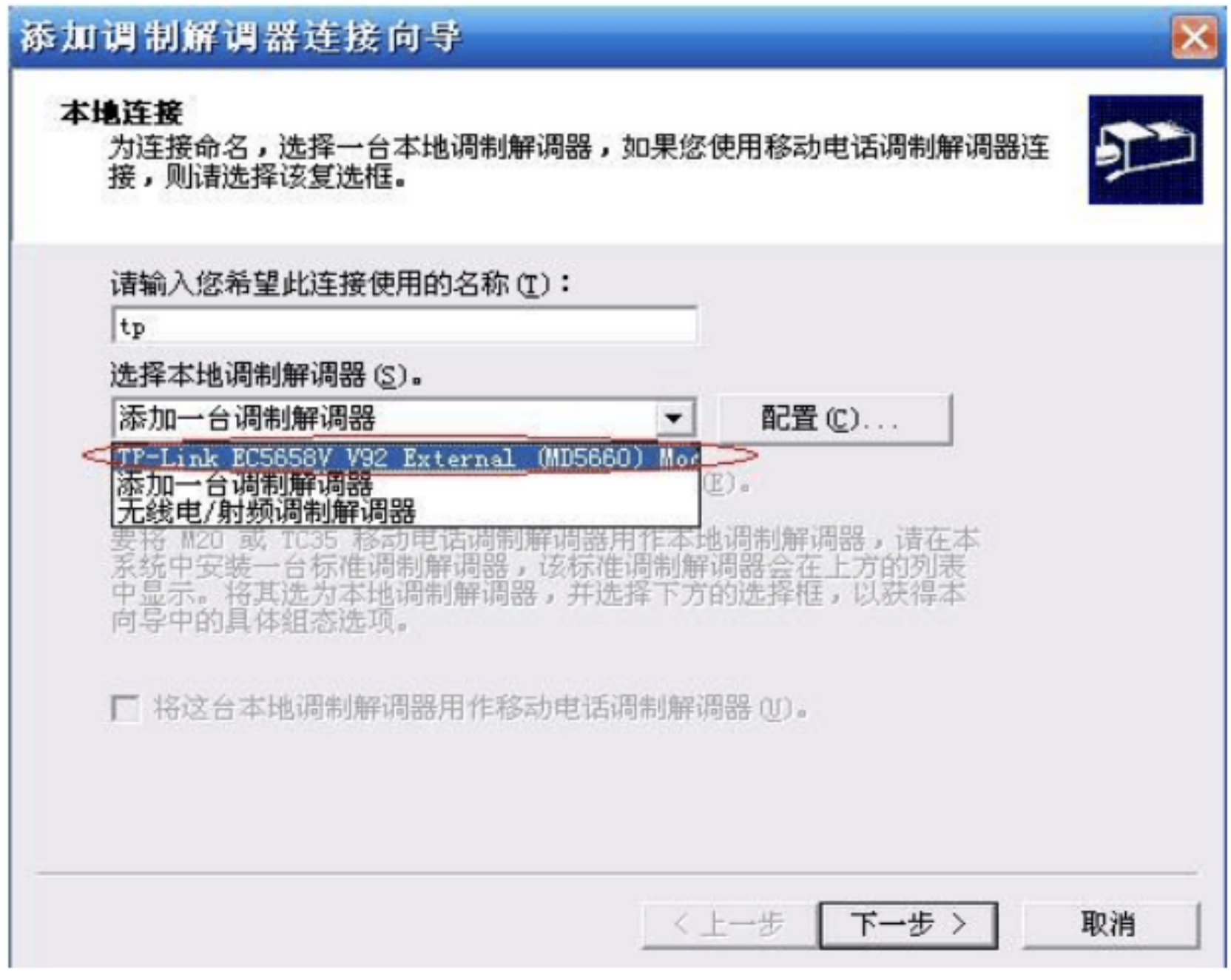
单击“ 设置 ”来添加连接线路。单击后进入“ 调制解调器连接设置 ”对话框，如下图所示：

单击“ 添加 ”或“ 设置 ”按钮来添加一个　　moden 连接，单击后进入“ 添加调制解调器连接向导 ”，如下图所示：

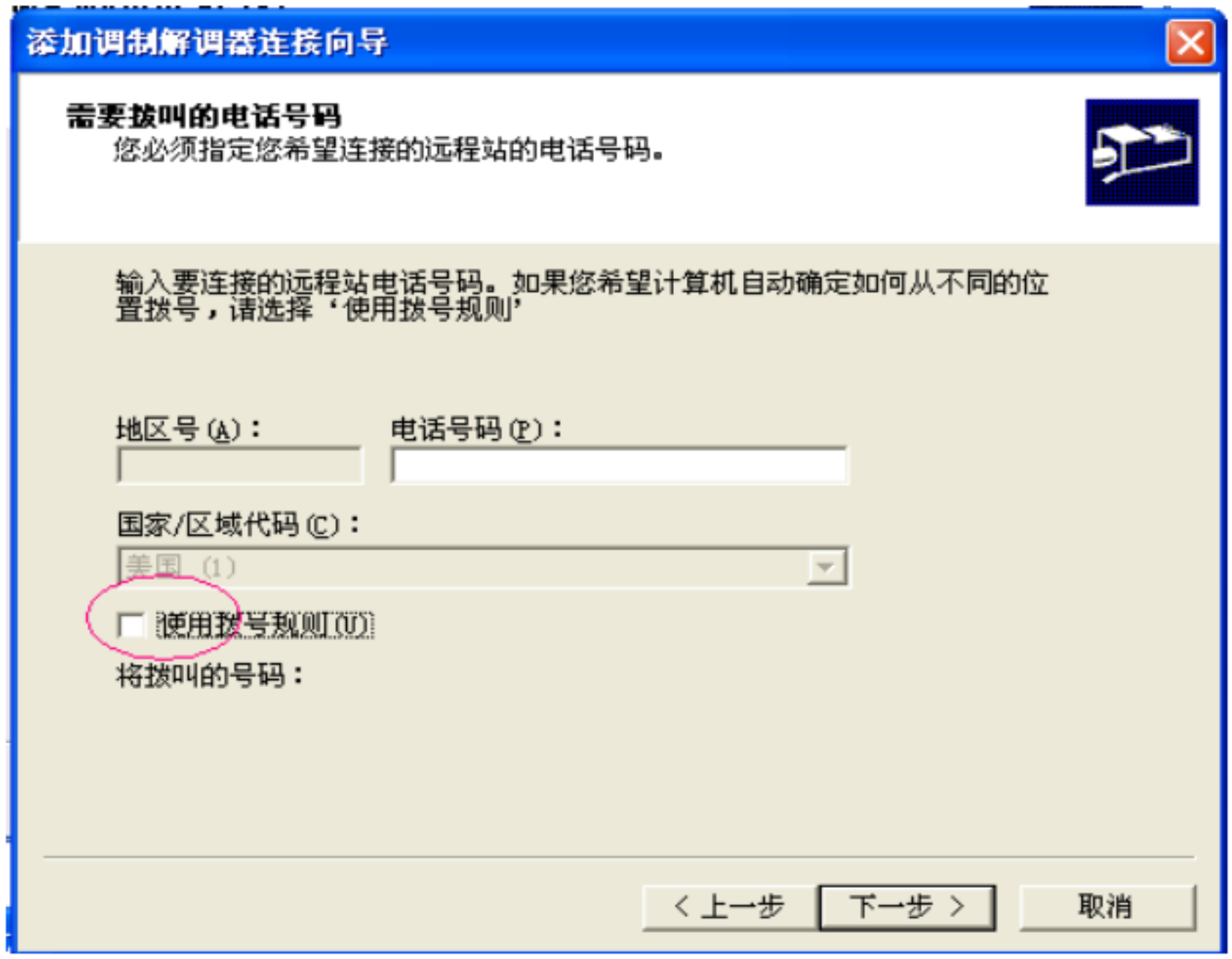
在“ 请输入您希望此连接使用的名称 ”处输入连接名称，再在“ 选择本地调试解调器 ”的列表框中选择已经安装的调制解调器。如下图所示：

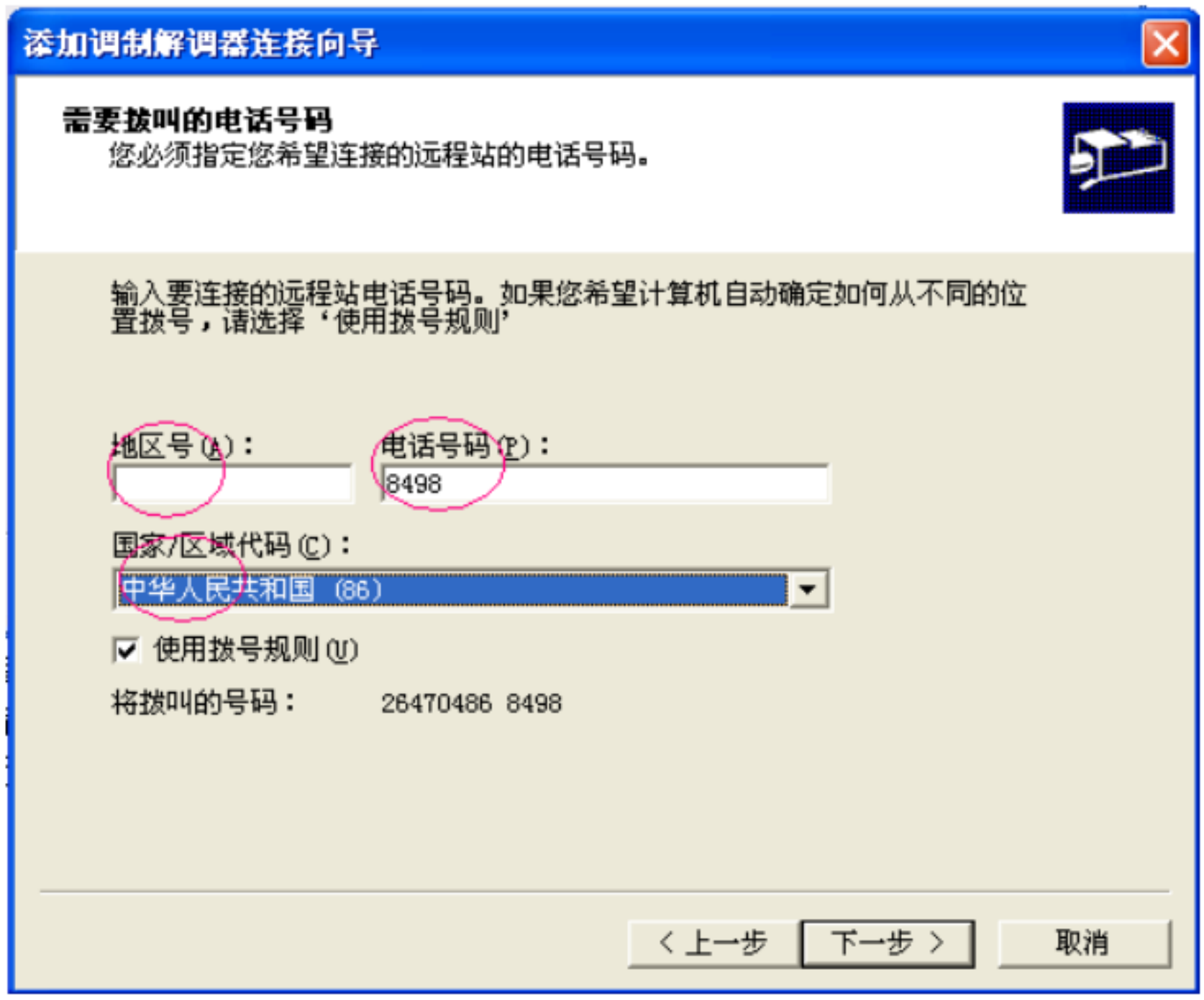
如果没有找到您需要的调制解调器，单击“ 配置 ”对话框中的“ 添加 ”按钮，根据向导来完成添假过程，添加成功后如下图红圈所示显示以添加调制解调器的相关信息：

单击“ 确定 ”后，新添加的调制解调器显示在“ 添加调制解调器向导 ”的“ 选择本地调制解调器 ”下的列表框中，从列表框中选择需要的调制解调器，如下图所示：



单击下一步。设置远程站（即远程　　moden）的电话号码，如下图所示，选中“使用拨号规则”，按向导完成远程电话号码的设置。

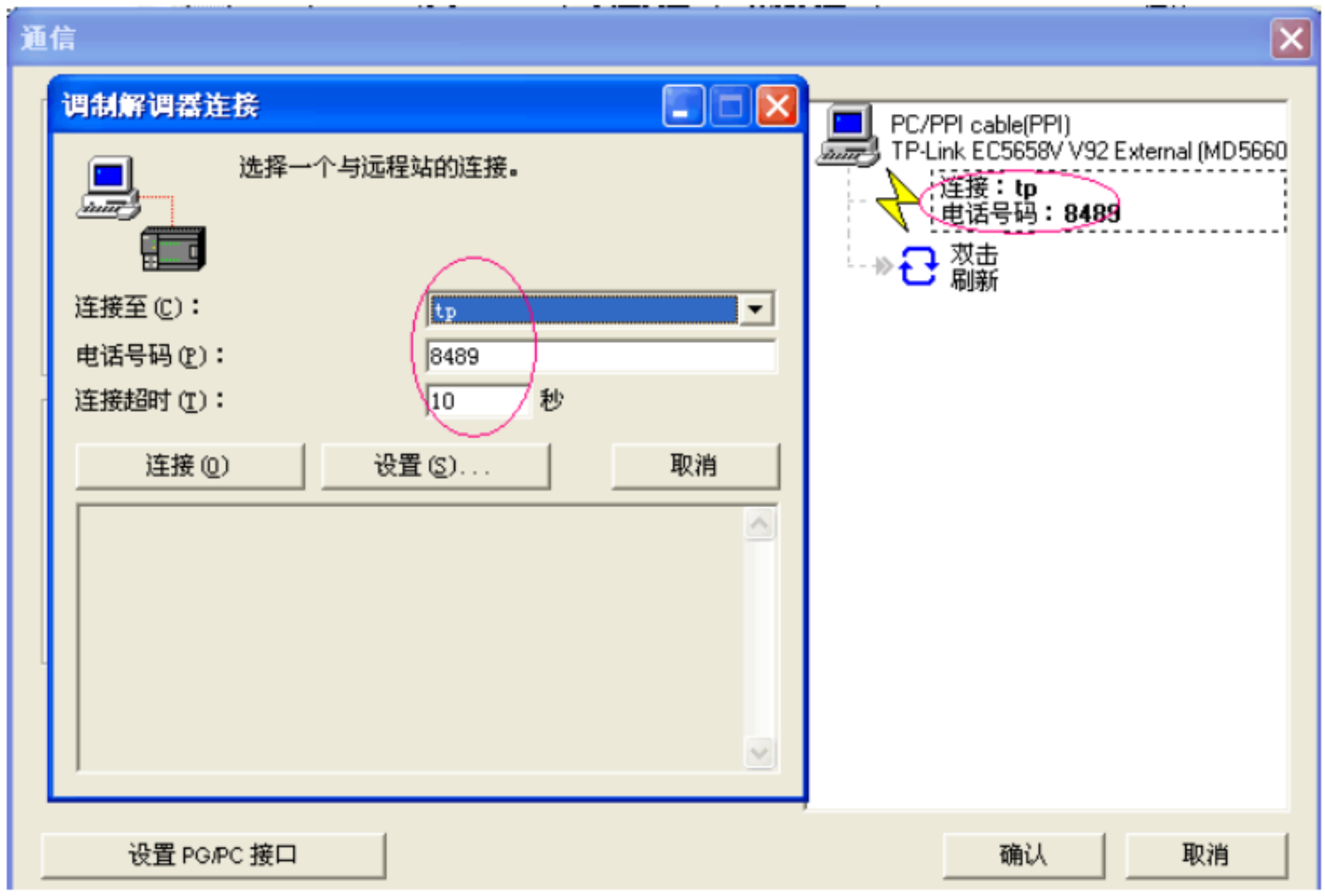




注：如果本地和远程用的是同一网络的不同分机号（即内线），尽量设置为内线号码（如远程端的内线号码为 8494，则“ 电话号码 ”处直接设置为　　8489）。

远程电话号码设置完成后，就会在“ 调制解调器连接设置 ”中出现刚设置的连接名，如下图红圈所示：

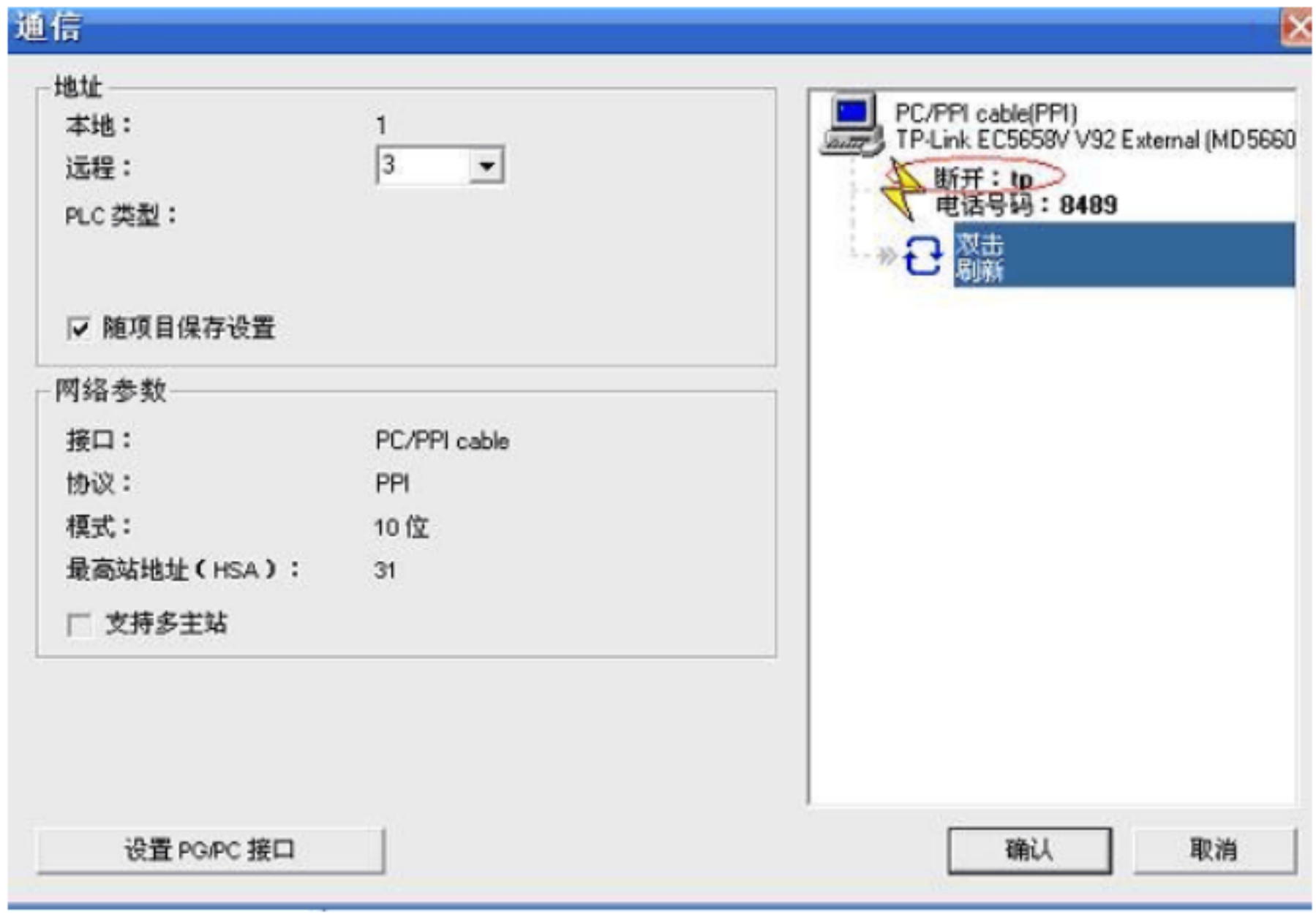
用上面同样的方法可以创建多个　　moden 连接。选中需要使用的连接后单击　“ 关闭 ”，“ 通信 ”中的连接信息及“ 调制解调器连接 ”对话框的信息如下图所示：



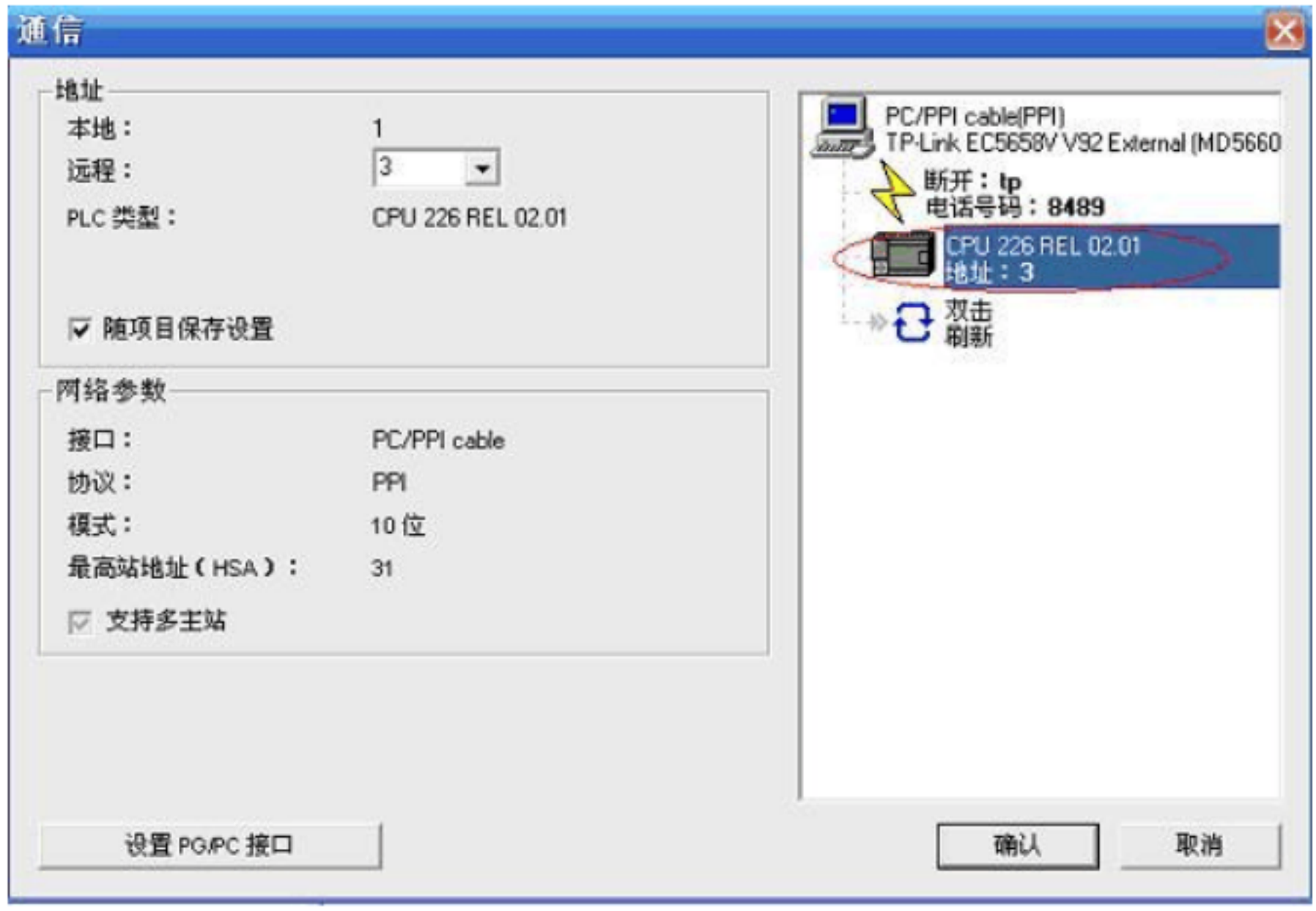
3）建立和远程端调制解调器的连接

单击“ 调制解调器连接 ”的“ 连接 ”按钮，和远程站建立连接，这时 moden 就会发出拨号操作，直到拨号完成为止。拨号过程如下图所示。

拨通后在通信的对话框中会稍有改变。如下图所示：



建立好了远程连接后显示如下：



10. CT 存储卡和电池卡的使用

CT CPU 只支持 CT 专用的存储卡和电池卡。

10.1. CT存储卡

和西门子存储卡使用方法相同。

10.1.1. CT存储卡不同之处

PLC 上电重启后存储卡内容不会自动复制到 PLC 中。

10.2. CT电池卡

和西门子电池卡使用方法相同，在电池卡后备的情况下，数据可掉电保持 200 天左右。

11. CPU 运算速度快可能带来的使用问题

极少数用户应用程序因使用不当，如用扫描周期来累积时间作为定时器的用途，会出现以下 2 种情况：

1. 用 SM0.6 变化的次数累积时间会远远小于西门子 CPU 累计的时间。
2. 用 SMW22 累积时间。如果程序在 CTS7200CPU 上运行循环周期比较长（大于 10ms），不会有问题。否则由于 SMW22 最大误差为 1ms，导致相对误差比较大，累积出的时间会差别比较大，特别是程序在西门子 CPU 上运行周期超过 1ms，但在 Co-trust CPU 上运行小于 1ms 的情况。

以上 2 种用法不提倡。