1 概述

1.1 S7-1200 的 PROFINET 通信口

S7-1200 CPU 本体上集成了一个 PROFINET 通信口，支持以太网和基于 TCP/IP 的通信标准。使用这个通信口可以实现 S7-1200 CPU 与编程设备的通信，与 HMI 触摸屏的通信，以及与其它 CPU 之间的通信。这个 PROFINET 物理接口是支持 10/100Mb/s 的 RJ45 口，支持电缆交叉自适应，因此一个标准的或是交叉的以太网线都可以用于这个接口。

1.2 S7-1200 支持的协议和最大的连接资源

S7-1200 CPU 的 PROFINET 通信口支持以下通信协议及服务

- TCP
- ISO on TCP（RCF 1006）
- S7 通信 (服务器端)

通信口所支持的最大通信连接数

S7-1200 CPU PROFINET 通信口所支持的最大通信连接数如下：

- 3 个连接用于 HMI (触摸屏) 与 CPU 的通信
- 1 个连接用于编程设备（PG）与 CPU 的通信
- 8 个连接用于 Open IE (TCP, ISO on TCP) 的编程通信，使用 T-block 指令来实现
- 3 个连接用于 S7 通信的服务器端连接，可以实现与 S7-200，S7-300 以及 S7-400 的以太网 S7 通信

S7-1200 CPU 可以同时支持以上 15 个通信连接，这些连接数是固定不变的，不能自定义。

TCP（Transport Connection Protocol）

TCP 是由 RFC 793 描述的标准协议，可以在通信对象间建立稳定、安全的服务连接。如果数据用 TCP 协议来传输，传输的形式是数据流，没有传输长度及信息帧的起始、结束信息。在以数据流的方式传输时接收方不知道一 条信息的结束和下一条信息的开始。因此，发送方必须确定信息的结构让接收方能够识别 。在多数情况下 TCP 应用了 IP (Internet protocol)，也就是“TCP/IP 协议”， 它位于 ISO-OSI 参考模型的第四层。

协议的特点：

- 与硬件绑定的高效通信协议
- 适合传输中等到大量的数据 (<=8192 bytes)
- 为大多数设备应用提供
  – 错误恢复
  – 流控制

– 可靠性

- 一个基于连接的协议
- 可以灵活的与支持 TCP 协议的第三方设备通信
- 具有路由兼容性
- 只可使用静态数据长度
- 有确认机制
- 使用端口号进行应用寻址
- 大多数应用协议，如 TELNET、FTP 都使用 TCP
- 使用 SEND/RECEIVE 编程接口进行数据管理需要编程来实现

1.3 物理网络连接

S7-1200 CPU 的 PROFINET 口有两种网络连接方法：

- 直接连接：当一个 S7-1200 CPU 与一个编程设备，或是 HMI ，或是另一个 PLC 通信时，也就是说只有两个通信设备时，实现的是直接通信。直接连接不需要使用交换机，用网线直接连接两个设备即可，如图 1 所示。

图 1 通

- 网络连接：当多个通信设备进行通 网络连接，如图 2 所示。

多个通信设备的网络连接需要使用以太网 77 的 4 口交 l 设备。CSM1277 交换机是即插即用的，使用前不用做任何设置。

S7-1200 CPU 与 另一台 S7-1200 CPU 直接连接

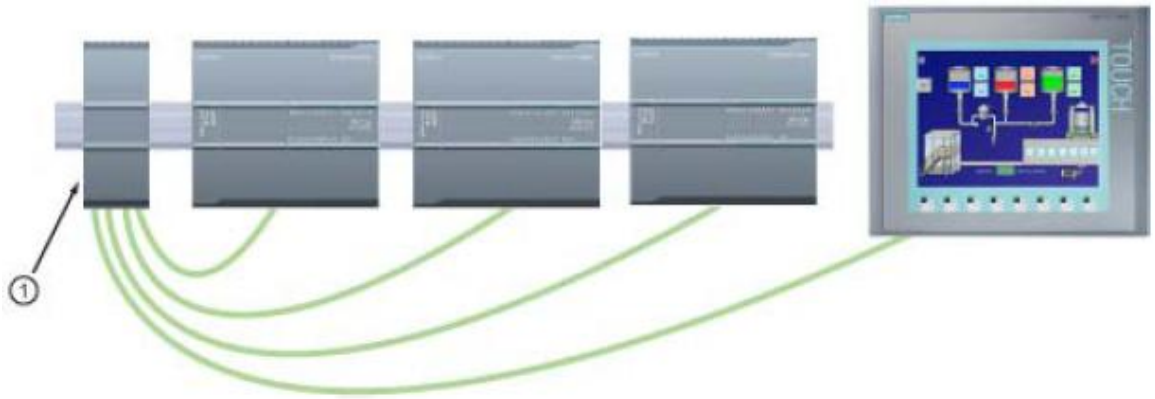


图 2 多个通信设备的网络连接

□ CSM1277 以太网交换机

2 S7-1200 CPU 通过 ETHERNET 与 S7-1200 CPU 通信

S7-1200 与 S7-1200 之间的以太网通信可以通过 TCP 或 ISO on TCP 协议来实现，使用的通信指令是在双方 CPU 调用 T-block (TSEND_C, TRCV_C, TCON, TDISCON, TSEN, TRCV) 指令来实现。通信方式为双边通信，因此 TSEND 和 TRCV 必须成对出现。因为 S7-1200 CPU 目前只支持 S7 通信的服务器（Sever）端，所以它们之间不能使用 S7 这种通信方式。

2.1 硬件和软件需求及所完成的通信任务

硬件：

□ S7-1200 CPU

□ PC （带以太网卡）

□ TP 电缆

软件：

STEP 7 Basic V10.5

所完成的通信任务：

□ 将 PLC_1 的通信数据区 DB 块中的 100 个字节的数据发送到 PLC_2 的接收数据区 DB 块中。

□ PLC_1 的 QB0 接收 PLC_2 发送的数据 IB0 的数据。

2.2 创建新项目及建立逻辑连接

□打开 STEP 7 Basic 软件并新建项目

在 STEP 7 Basic 的 “Portal View” 中选择 “Create new project” 创建一个新项目

□添加硬件并命名 PLC

然后进入 “Project view”，在“Project tree” 下双击 “Add new device”，在对话框中选择所使用的 S7-1200 CPU 添加到机架上，命名为 PLC_1，如图 3 所示。

同样方法再添加通信伙伴的 S7-1200 CPU ，命名为 PLC_2。

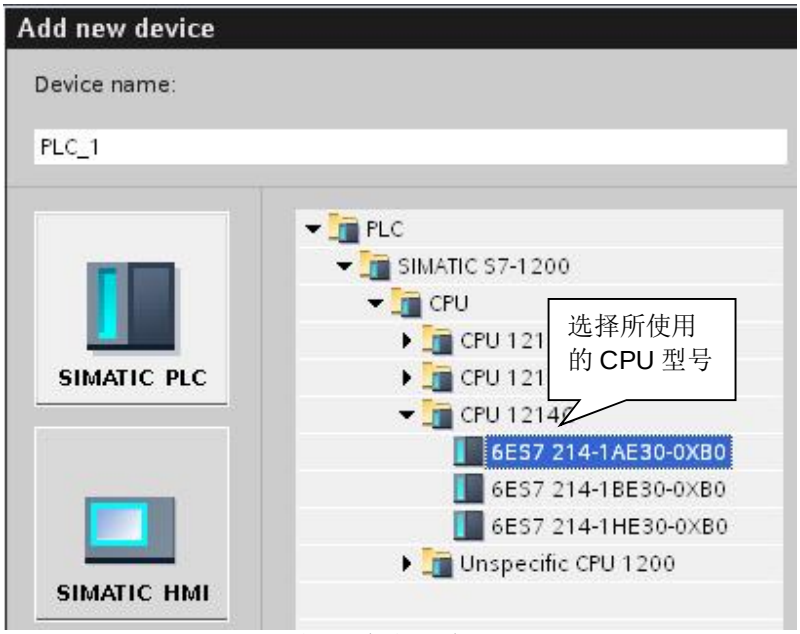


图 3 添加新设备

为了编程方便，我们使用 CPU 属性中定义的时钟位，定义方法如下：

在 “Project tree” > “PLC_1” > “Device configuration” 中，选中 CPU ，然后在下面的属性窗口中，“Properties” > “System and clock memory” 下，将系统位定义在 MB1，时钟位定义在 MB0，如图 4 所示。

时钟位我们主要使用 M0.3，它是以 2Hz 的速率在 0 和 1 之间切换的一个位。可以使用它去自动激活发送任务。

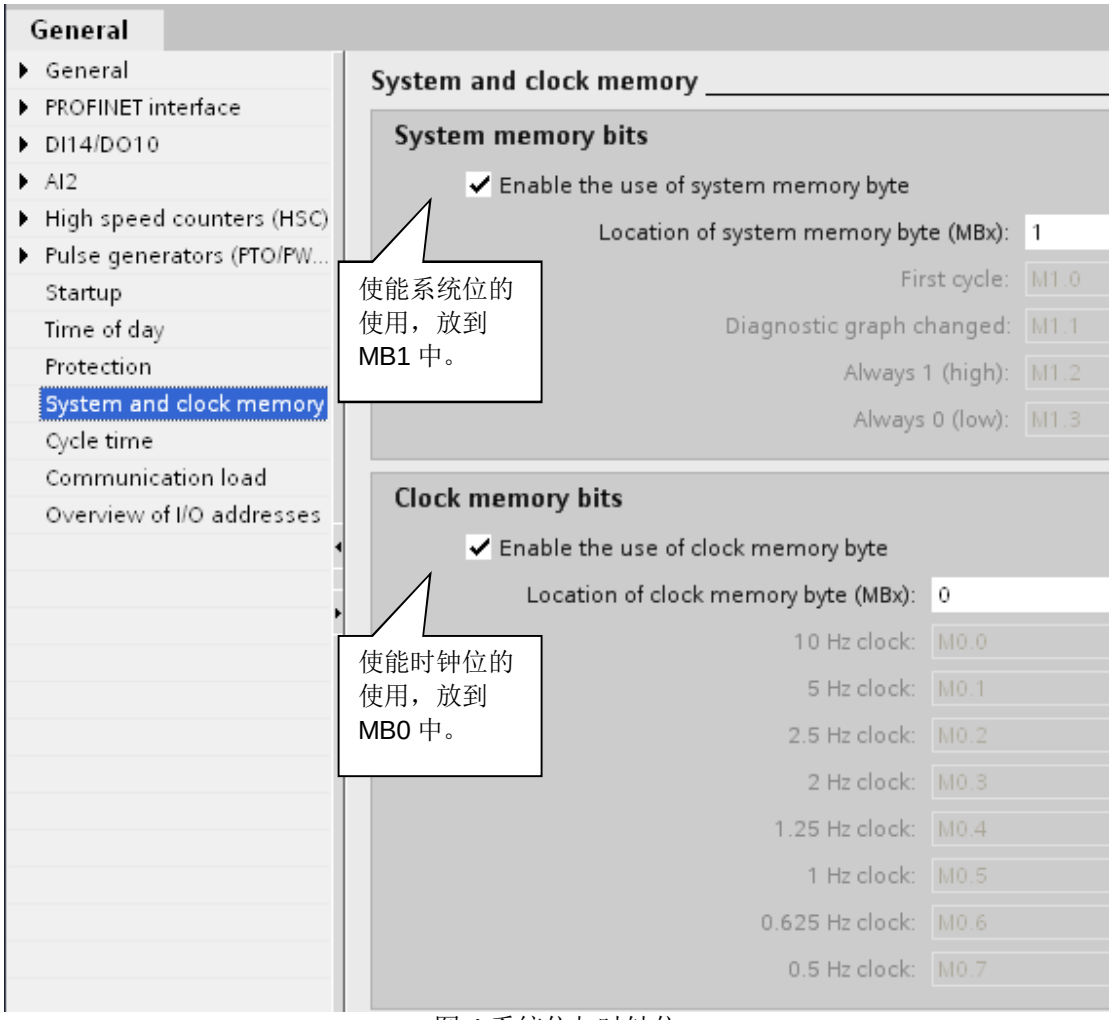


图 4 系统位与时钟位

□为 PROFINET 通信口分配以太网地址

在 “Device View”中点击 CPU 上代表 PROFINET 通信口的绿色小方块，在下方会出现 PROFINET 接口的属性，在 “Ethernet addresses” 下分配 IP 地址为 192.168.0.1 ，子网掩码为 255.255.255.0，如图 5 所示。

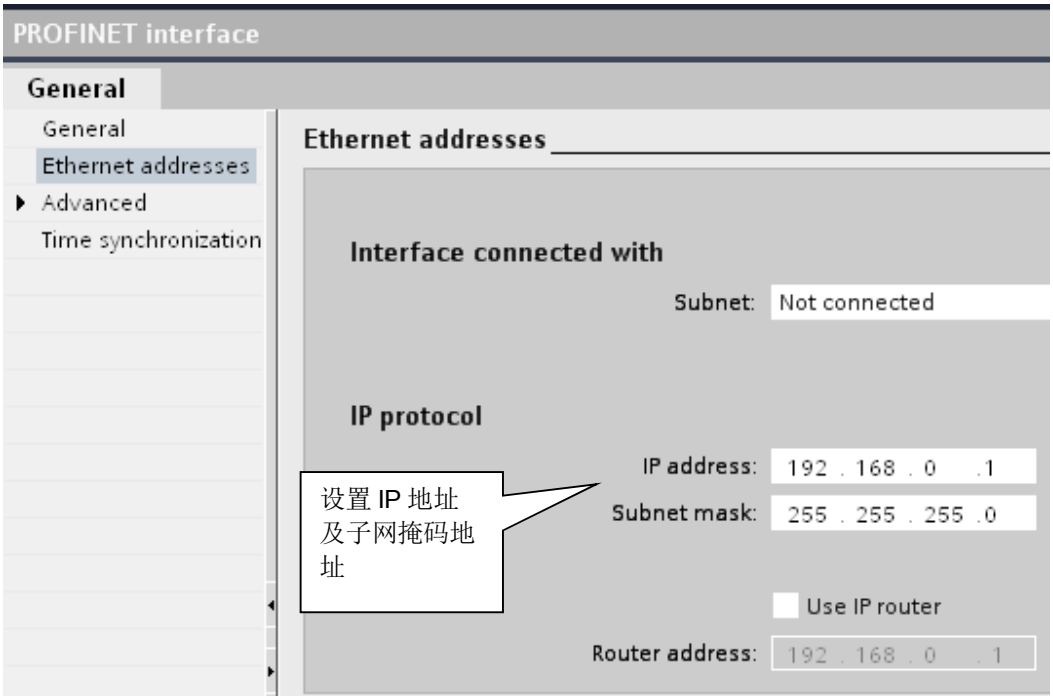


图 5 分配 IP 地址

□同样方法，在同一个项目里添加另一个新设备 S7-1200 CPU 并为其分配 IP 地址为 192.168.0.2

□创建 CPU 之间的逻辑网络连接

在项目树 “Project tree”>“Devices & Networks” >“Networks view” 视图下，创建两个设备的连接。用鼠标点中 PLC_1 上的 PROFINET 通信口的绿色小方框，然后拖拽出一条线，到另外一个 PLC_2 上的 PROFINET 通信口上，松开鼠标，连接就建立起来了，如图 6 所示。

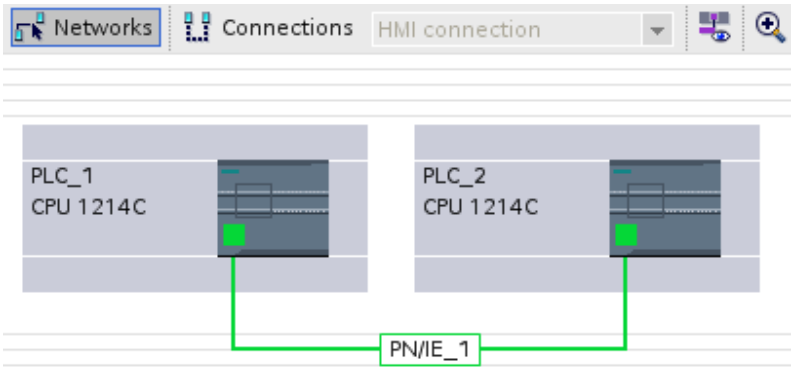


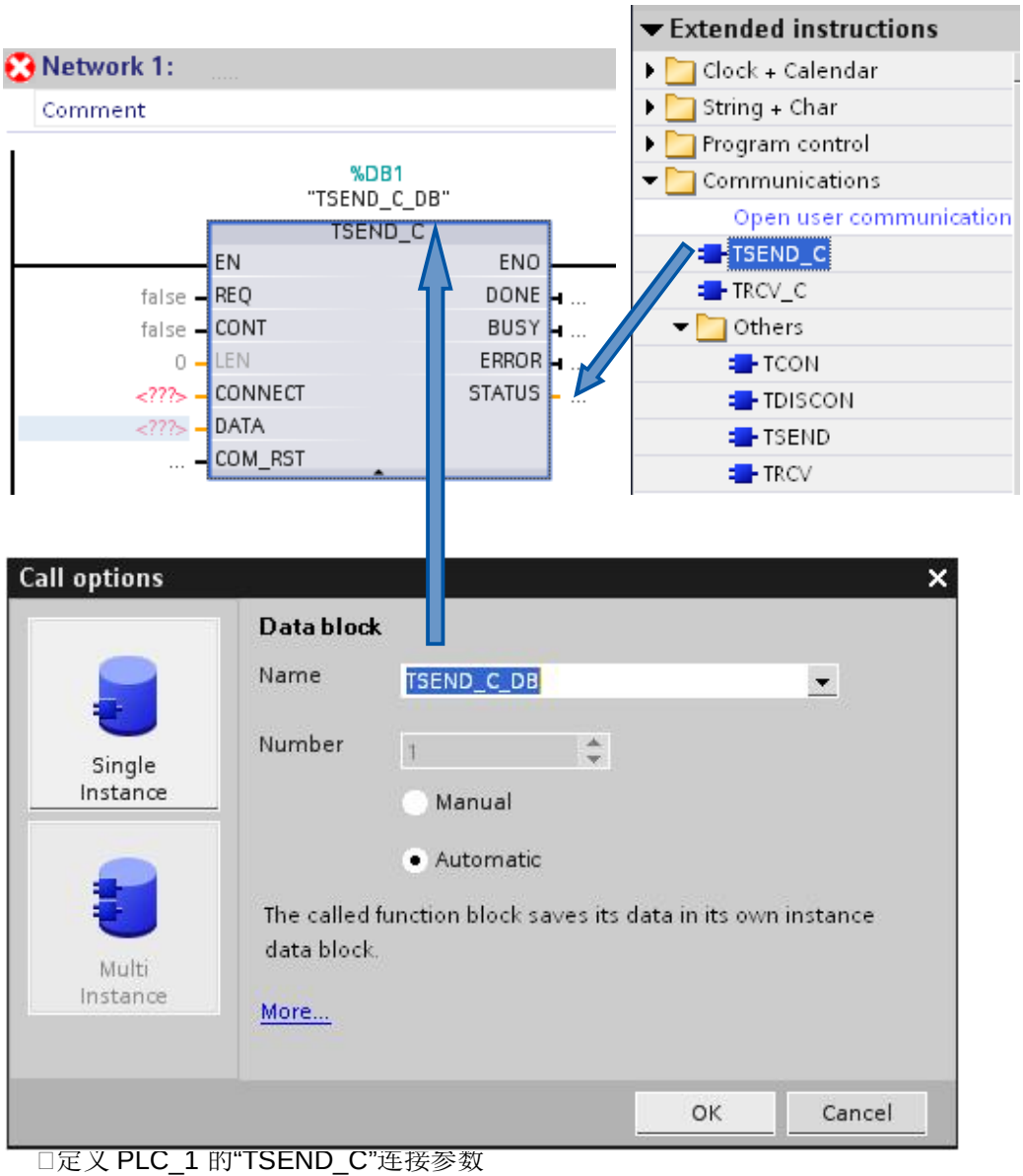
图 6 建立两个 CPU 的逻辑连接

2.3 TCP 通信

2.3.1 在 PLC_1 中调用并配置“TSEND_C”、“T_RCV” 通信指令

□ 在 PLC_1 的 OB1 中调用“TSEND_C” 通信指令

在第一个 CPU 中调用发送通信指令，进入 “Project tree” > “ PLC_1” > “Program blocks” > “OB1” 主程序中，从右侧窗口 “Instructions” > “Extended Instructions” > “Communications” 下调用 “TSEND_C” 指令，并选择 “Single Instance” 生成背景 DB 块。然后单击指令块下方的“下箭头”，使指令展开显示所有接口参数，如图 7 所示。



□定义 PLC_1 的“TSEND_C”连接参数

PLC_1 的 TSEND_C 指令的连接参数需要在指令下方的属性窗口“Properties”> “Configuration”>“Connection parameter”中设置，如图 8 所示。

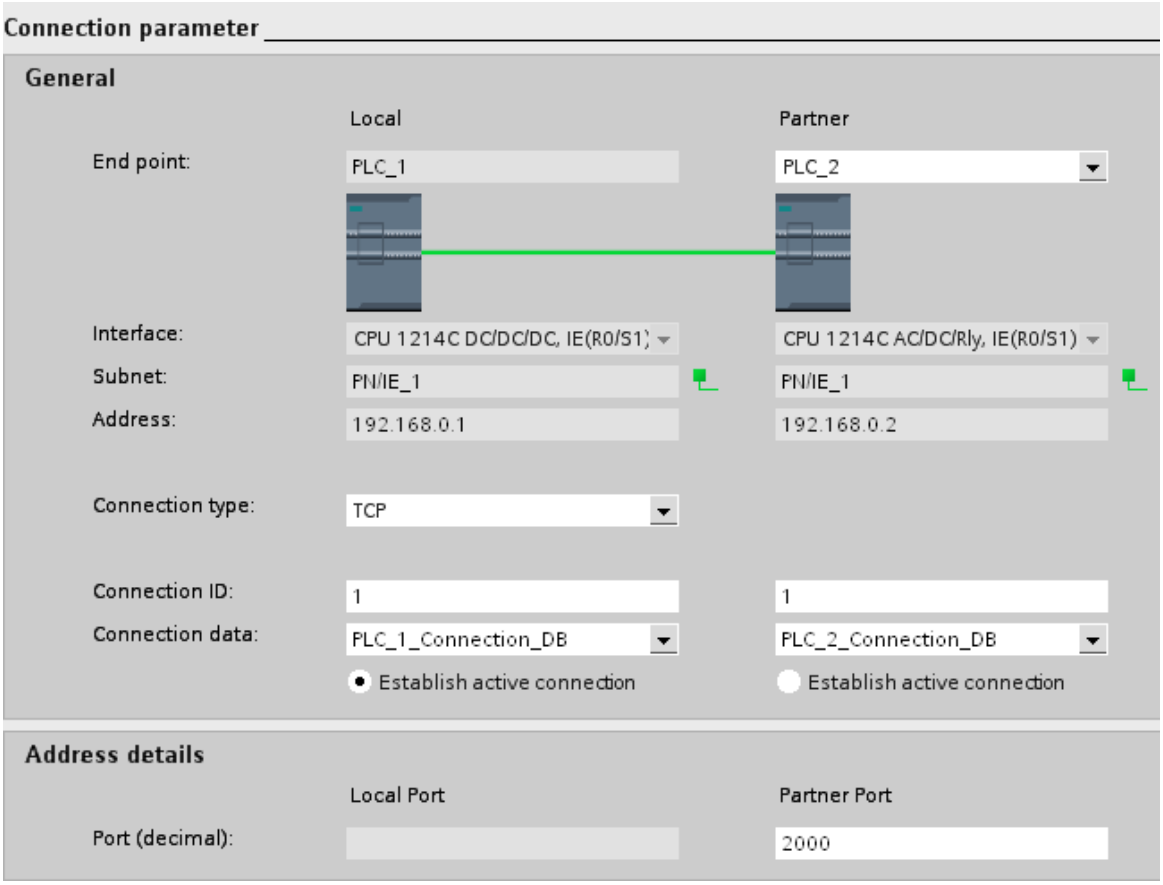


图8 定义 TSEND_C 连接参数

连接参数说明：

End point ：可以通过点击选择按钮选择伙伴 CPU ：PLC_2

Connection type ：选择通信协议为 TCP

Connection ID ：连接的地址 ID 号，这个 ID 号在后面的编程里会用到

Connection data ：创建连接时，系统会自动生成本地的连接 DB 块，所有的连接数据都会存在这个 DB 块中。通信伙伴的连接 DB 块，只有在对方（PLC_2）建立连接后才能生成，然后在本地 (PLC_1) 中才能通过选择按钮选择。

Active connection setup ：选择本地 PLC_1 作为主动连接

Address details ：定义通信伙伴方的端口号为：2000；如果选用的是 ISO on TCP 协议，则需要设定的是 TSAP 地址（ASCII 形式），本地 PLC_1 可以设置成“PLC1”，伙伴方 PLC_2 可以设置成“PLC2”。

□ 定义 PLC_1 的“TSEND_C”发送通信块接口参数

首先，根据所使用的接口参数定义符号表

在“Project tree”> “PLC_1” > “PLC tags”中定义所使用的符号名，如图 9 所示。

PLC tags

	Name	Data type	Address	Retain	Comment
1	2Hz_clock	Bool	%M0.3		
2	Input_byte0	Byte	%IB0		
3	T_C_COMR	Bool	%M10.0		
4	TSENDC_DONE	Bool	%M10.1		
5	TSENDC_BUSY	Bool	%M10.2		
6	TSENDC_ERROR	Bool	%M10.3		
7	TSENDC_STATUS	Word	%MW12		
8	Output_byte0	Byte	%QB0		
9	TRCV_NDR	Bool	%M10.4		
10	TRCV_BUSY	Bool	%M10.5		
11	TRCV_ERROR	Bool	%M10.6		
12	TRCV_RCVD_LEN	UInt	%MW16		
13	TRCV_STATUS	Word	%MW14		

图 9 定义所使用的符号表

然后，创建并定义 PLC_1 的发送数据区 DB 块。

通过“Project tree”>“PLC_1”>“Program blocks”>“Add new block”，选择“Data block” 创建 DB 块，选择绝对寻址，点击“OK”键，定义发送数据区为 100 个字节的数组，如图 10 及图 11 所示。

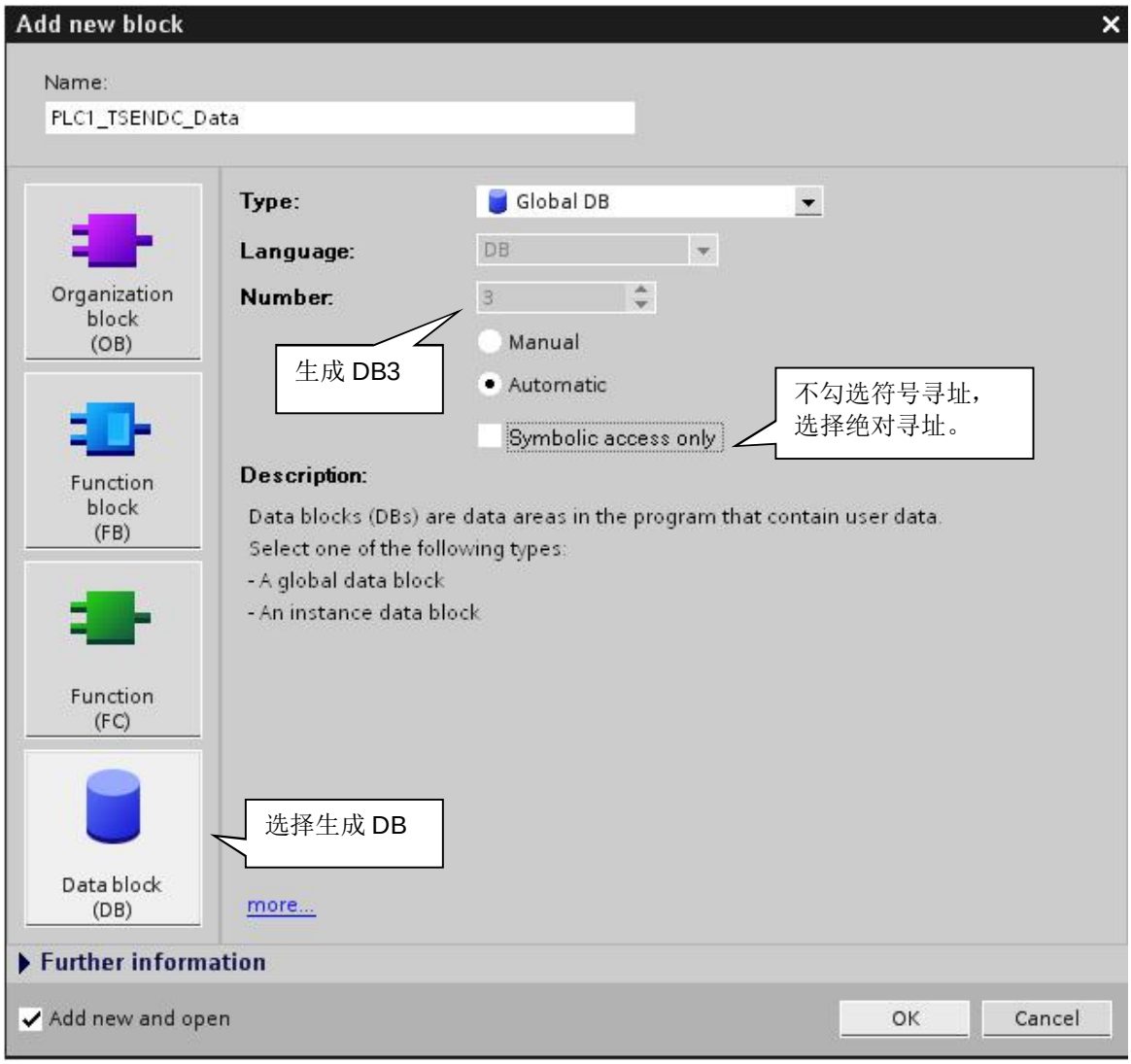


图 10 创建发送数据区 DB 块

注意：对于双边编程通信的 CPU，如果通信数据区使用 DB 块，既可以将 DB 块定义成符号寻址，也可以定义成绝对寻址。使用指针寻址方式，必须创建绝对寻址的 DB 块。

PLC1_TSENDC_Data

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Static_1	Array [0 .. 100] of byte	0.0		☑	

图 11 定义发送数据区为字节类型的数组

定义完通信数据区，继续定义 PLC_1 的“TSEND_C”发送通信块接口参数，如图 12 所示。

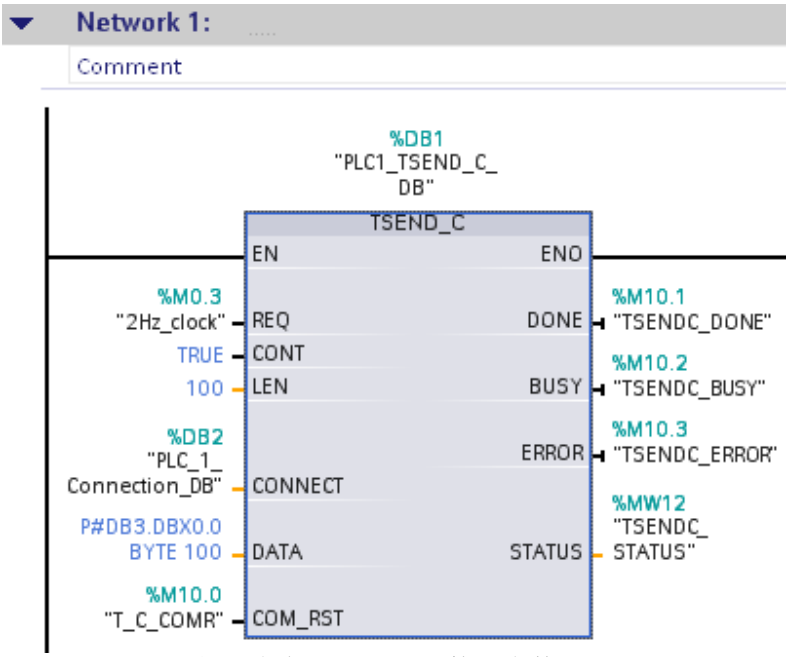


图 12 定义 TSEND_C 接口参数

参数说明：

输入接口参数：

REQ ：= “2Hz_clock” // 使用 2Hz 的时钟脉冲，上升沿激活发送任务

CONT ：= TRUE // 建立连接并一直保持连接

LEN ：= 100 //发送数据长度

CONNECT ：=“PLC_1_Connection_DB”//连接数据 DB 块

DATA ：= P#DB3.DBX0.0 BYTE 100 // 发送数据区的数据，使用指针寻址时，DB 块要选用绝对寻址

COM_RST ：=“T_C_COMR” //为 1 时，完全重启动通信块，现存的连接会中断

输出接口参数：

DONE ：=“TSENDC_DONE” //任务执行完成并且没有错误，该位置 1

BUSY ：=“TSENDC_BUSY” // 该位为 1，代表任务未完成，不能激活新任务

ERROR ：=“TSENDC_ERROR” //通信过程中有错误发生，该位置 1

STATUS ：=“TSENDC_STATUS” //有错误发生时，会显示错位信息号

□ 在 PLC_1 的OB1中调用接收指令T_RCV 并配置基本参数

为了实现 PLC_1 接收来自 PLC_2 的数据，则在 PLC_1 中调用接收指令 T_RCV 并配置基本参数。

接收数据与发送数据使用同一连接，所以使用不带连接管理的 T_RCV 指令。根据所使用的接口参数定义符号表，如图 2-15 所示。配置接口参数，如图 13 所示：

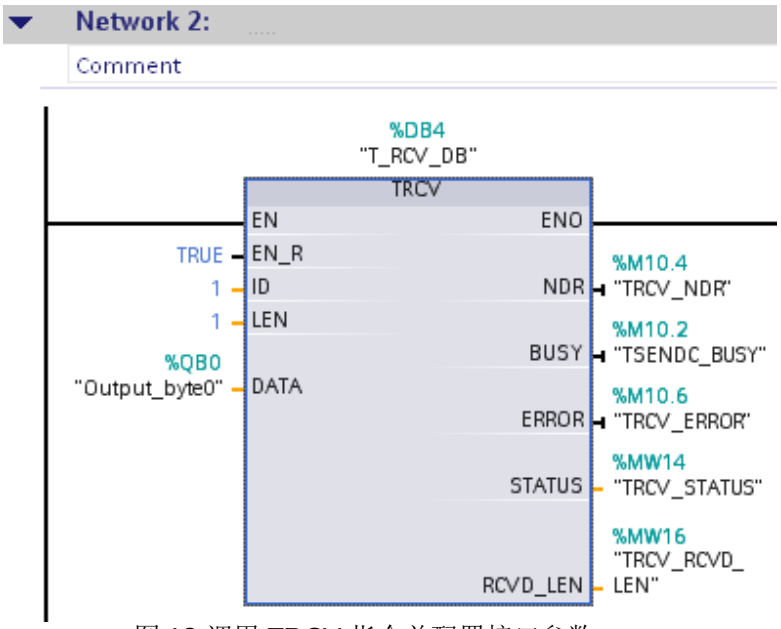


图 13 调用 TRCV 指令并配置接口参数

参数说明：

输入接口参数：

EN_R ：= TRUE // 准备好接收数据

ID ：= 1 // 连接号，使用的是 TSEND_C 的连接参数中 Connection ID 的参数地址

LEN ：= 1 // 接收数据长度为 1 个字节

DATA ：= “Output_byte0” // 接收数据区的符号地址

输出接口参数：

NDR ：= “TRCV_NDR” // 该位为 1，接收任务成功完成

BUSY ：= “TSENDC_BUSY” // 该位为 1，代表任务未完成，不能激活新任务

ERROR ：= “TRCV_ERROR” // 通信过程中有错误发生，该位置 1

STATUS ：= “TRCV_STATUS” // 有错误发生时，会显示错误信息号

RCVD_LEN ：= “TRCV_RCVD_LEN” // 实际接收数据的字节数

2.3.2　在 PLC_2 中调用并配置“TRCV_C”通信指令

□ 同样方法，在 PLC_2 中调用“TRCV_C”通信指令，进入“Project tree”> “ PLC_2”>“Program blocks”>“Main”主程序中，从右侧窗口“Instructions”> “Extended Instructions”>“Communications”下调用“TRCV_C”指令，并选择 “Single Instance” 生成背景 DB 块。

□ 定义连接参数， PLC_2 的“TRCV_C”指令的连接参数需要在指令下方的属性窗口“Properties”> “Connection parameter”中设置，如图 14 所示。

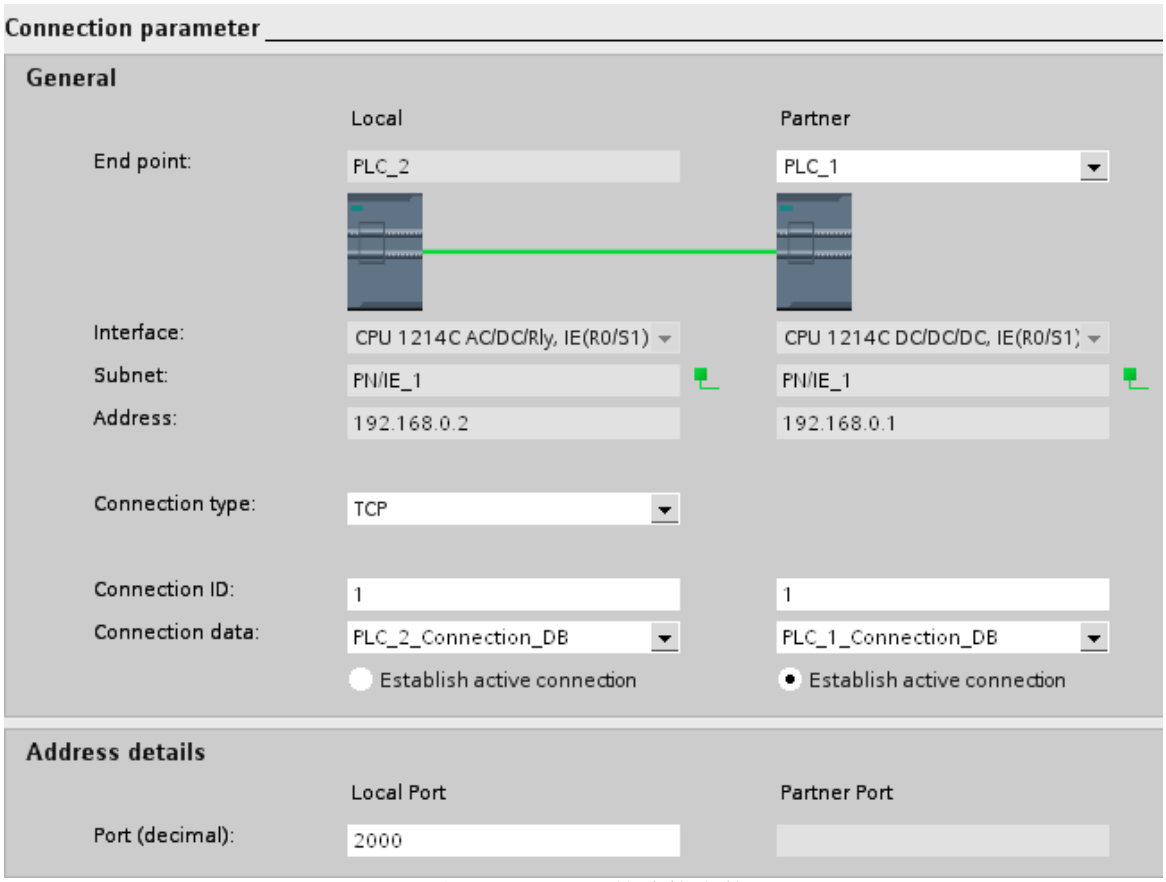


图 14 TRCV_C 的连接参数配置

连接参数的配置与 TSEND_C 的连接参数配置基本相似，各参数要与通信伙伴 CPU 对应设置。

□ 定义接收通信块参数

首先，创建并定义接收数据区 DB 块。

通过“Project tree”>“ PLC_2”>“Program blocks”>“Add new block” ，选择 “Data block”创建 DB 块，选择符号寻址，点击“OK”键，定义接收数据区为 100 个字节的数组，如图 15 及图 16 所示。

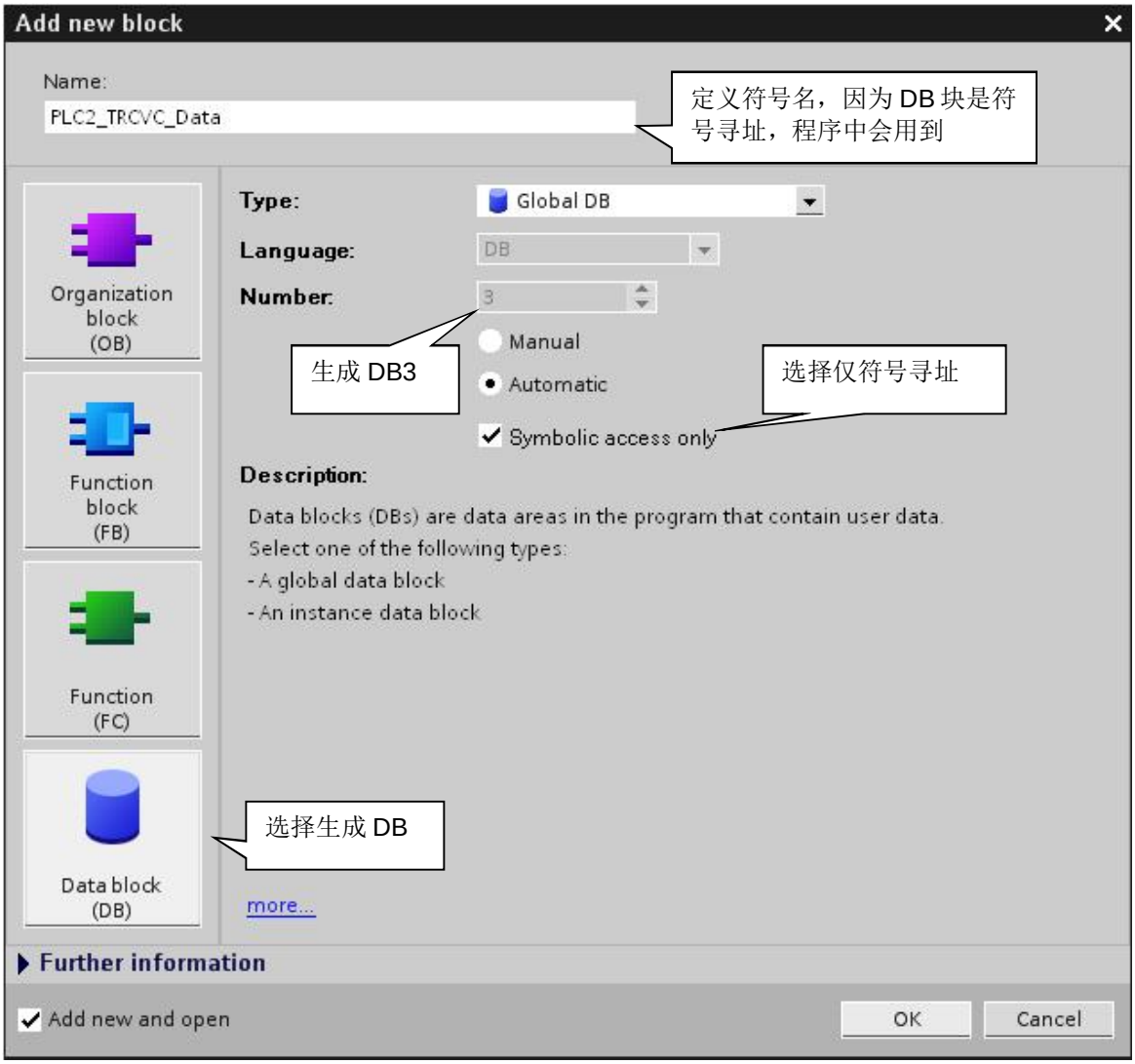


图 15 创建接收数据区 DB 块

PLC2_TRCVC_Data

	Name	Data type	Initial value	Retain	Comment
1	Static				
2	Static_1	Array [0 .. 100] of byte		☑	

图 16 定义接收区为 100 个字节的数组

然后，定义所使用参数的符号地址，如图 17 所示。

PLC tags

		Name	Data type	Address	Retain	Comment
1		T_C_COMR	Bool	%M10.0		
2		TRCVC_DONE	Bool	%M10.1		
3		TRCVC_BUSY	Bool	%M10.2		
4		TRCVC_ERROR	Bool	%M10.3		
5		TRCVC_STATUS	Word	%MW12		
6		TRCVC_RCVLEN	UInt	%MW14		
7		Input_byte0	Byte	%IB0		
8		TSEND_DONE	Bool	%M10.4		
9		TSEND_BUSY	Bool	%M10.5		
10		TSEND_ERROR	Bool	%M10.6		
11		TSEND_STATUS	Word	%MW16		
12		2Hz_clock	Bool	%M0.3		
13		Tag_1	Bool	%M0.4		

图 17 TRCV_C 指令所使用的符号地址

最后，定义接收通信块接口参数，如图 18 所示。

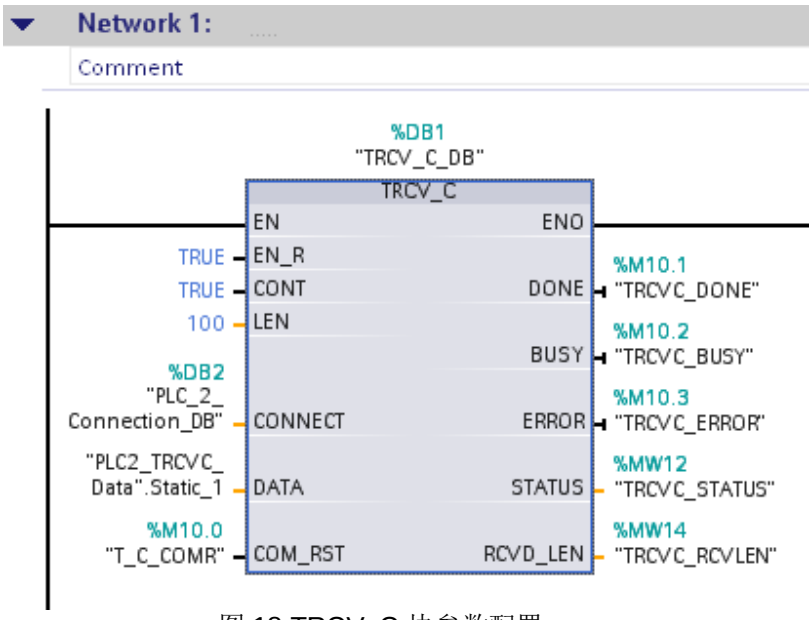


图 18 TRCV_C 块参数配置

参数配置：

输入接口参数：

EN_R ：= TRUE // 准备好接收数据

CONT ：= TRUE // 建立连接并一直保持连接

LEN ：= 100 // 接收的数据长度为 100 个字节

CONNECT ：=“PLC_2_Connection_DB” // 连接数据 DB 块

DATA ：=“PLC2_TRCVC_Data”.Static_1 //接收数据区，DB 块选用的是符号寻址

COM_RST ：=“T_C_COMR” // 为 1 时，完全重启动通信块，现存的连接会中断

输出接口参数：

DONE ：= “TRCVC_DONE” // 任务执行完成并且没有错误，该位置 1

BUSY ：= “TRCVC_BUSY” // 该位为 1，代表任务未完成，不能激活新任务

ERROR ：= “TRCVC_ERROR” // 通信过程中有错误发生，该位置 1

STATUS ：= “TRCVC_ERROR” // 有错误发生时，会显示错误信息号

RCVD_LEN ：= “TRCVC_RCVLEN” // 实际接收数据的字节数

□ PLC_2 将 I/O 输入数据 IB0 发送到 PLC_1 的 输出 QB0 中，则在 PLC_2 中调用发送指令并配置块参数，发送指令与接收指令使用同一个连接，所以使用不带连接的发送指令 T_SEND，如图 19 所示。

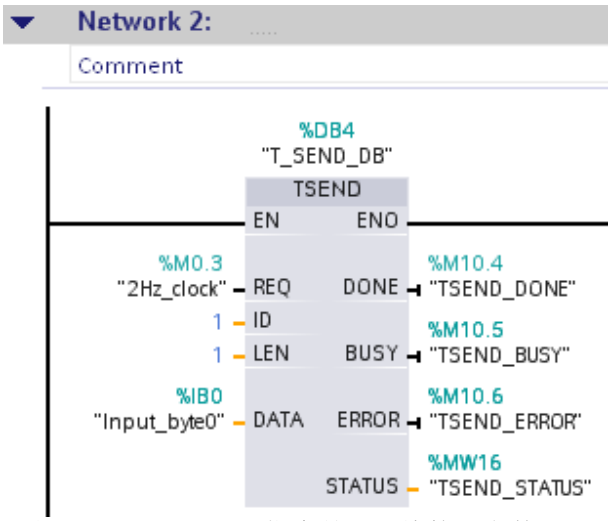


图 19 调用 T_SEND 指令并配置块接口参数

参数说明：

输入接口参数：

REQ ：= “2Hz_clock” // 使用 2Hz 的时钟脉冲，上升沿激活发送任务

ID ：= 1 // 连接号，使用的是 TRCV_C 的连接参数中 Connection ID 的参数地址

LEN ：= 1 // 接收数据长度为 1 个字节

DATA ：= “Input_byte0” // 接收发送数据区的符号地址

输出接口参数：

DONE ：“TSEND_DONE” //任务执行完成并且没有错误，该位置 1

BUSY ：“TSEN_BUSY” // 该位为 1，代表任务未完成，不能激活新任务

ERROR ：“TSEND_ERROR” //通信过程中有错误发生，该位置 1

STATUS ：“TSEND_STATUS” //有错误发生时，会显示错误信息号

2.3.3 下载硬件组态及程序并监控通信结果

下载两个 CPU 中的所有硬件组态及程序，从监控表中可以看到，PLC_1 的 TSEND_C 指令发送数据：“11”，“22”，“33”数据，PLC_2 接收到数据：“11”，“22”，“33”。而 PLC_2 发送数据 IB0 为“0001_0001”，PLC_1 接收数据到 QB0 也是“0001_0001”，如图 20 所示。

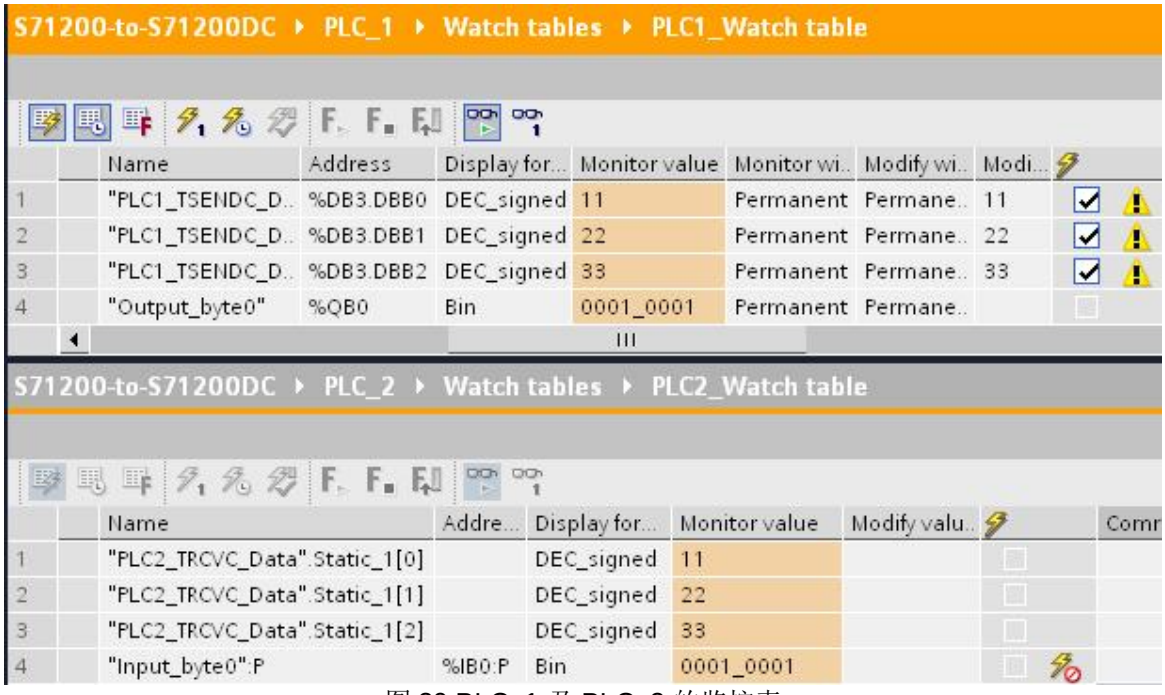


图 20 PLC_1 及 PLC_2 的监控表

2.4 ISO on TCP 通信

使用 ISO on TCP 协议通信，除了连接参数的定义不同，其它组态编程与 TCP 协议通信完全相同。

2.4.1 ISO on TCP 协议通信连接参数的配置

S7-1200 CPU 中，使用 ISO on TCP 协议通信时，PLC_1 的连接参数如图 21 所示。通信伙伴 PLC_2 的连接参数，如图 22 所示。

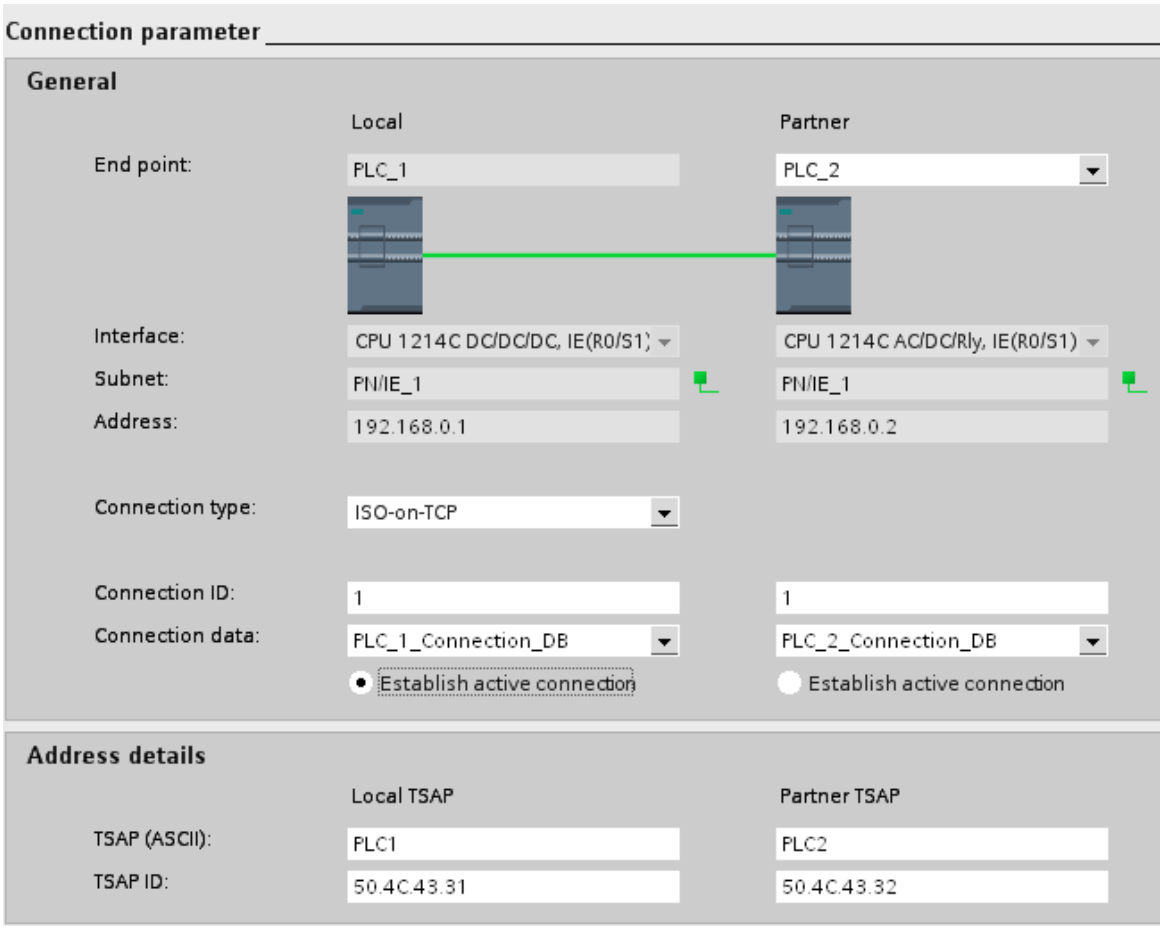


图 21 PLC_1 的 ISO on TCP 协议通信连接参数

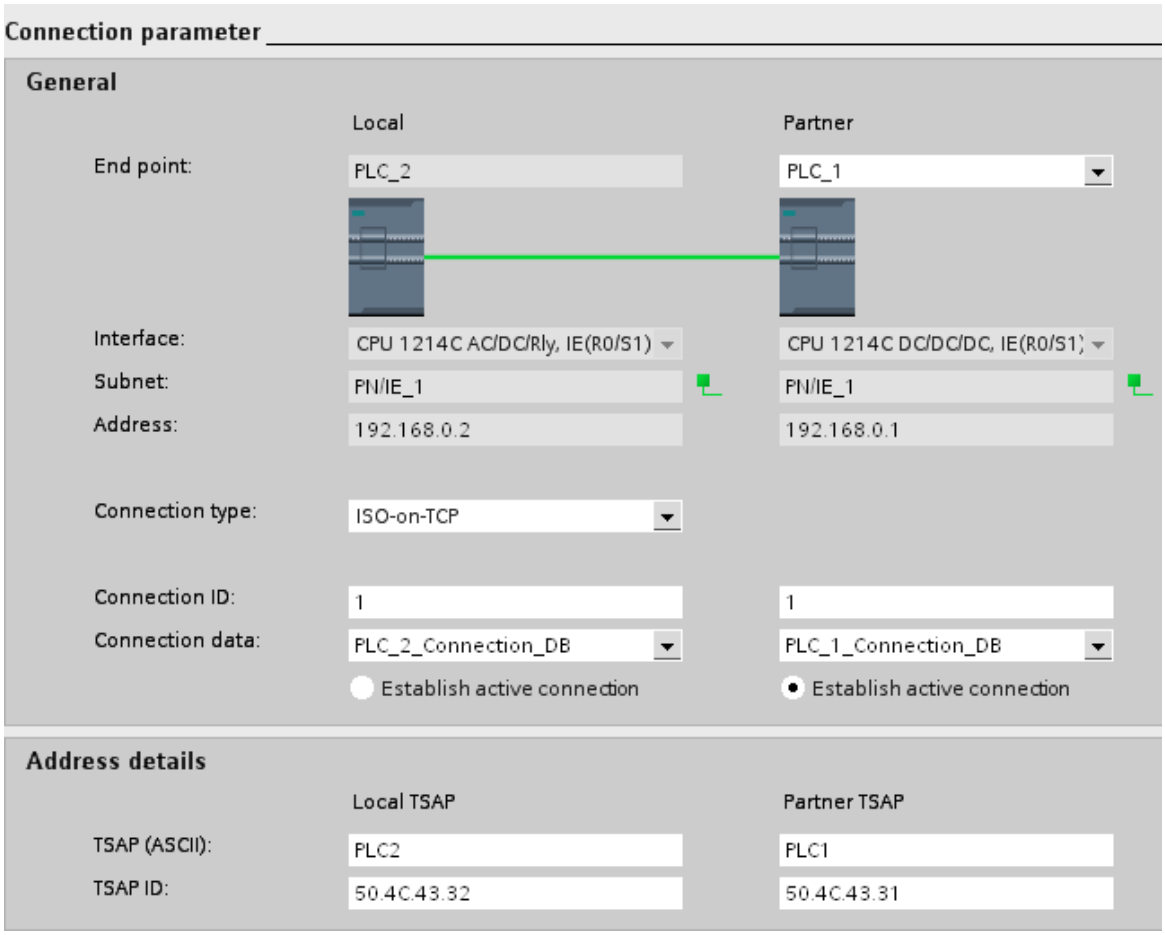


图 22 PLC_2 的 ISO on TCP 协议通信连接参数

2.4.2 ISO on TCP 动态长度数据传输

ISO on TCP 协议支持动态长度的数据传输，而 TCP 协议只支持静态长度的数据传输。 而且只有使用符号寻址的 ISO on TCP 通信才支持动态数据长度传输。

使用符号寻址数据传输，需要数据块发送方和接收方的数据区结构一致。

□通信数据区的定义

发送方的数据块通信数据区定义为 5 个字节，如 23 所示

PLC1_TSENDC_Data

	名称	数据类型	偏移量	初始值
1	Static			
2	Static_1	Array [0 .. 5] of byte	0.0	
3	Static_1[0]	Byte		B#16#00
4	Static_1[1]	Byte		B#16#00
5	Static_1[2]	Byte		B#16#00
6	Static_1[3]	Byte		B#16#00
7	Static_1[4]	Byte		B#16#00
8	Static_1[5]	Byte		B#16#00

图 23 发送方数据块通信数据区的定义

接收方的数据区也定义为 5 个字节，如图 4. 所示

PLC2_TRCVC_Data

	名称	数据类型	初始值
1	Static		
2	Static_1	Array [0 .. 5] of byte	
3	Static_1[0]	Byte	B#16#00
4	Static_1[1]	Byte	B#16#00
5	Static_1[2]	Byte	B#16#00
6	Static_1[3]	Byte	B#16#00
7	Static_1[4]	Byte	B#16#00
8	Static_1[5]	Byte	B#16#00

图 24 接收方数据块通信数据区的定义

□ 编程

发送方的程序如图 25 所示，“LEN”参数要定义成变量。

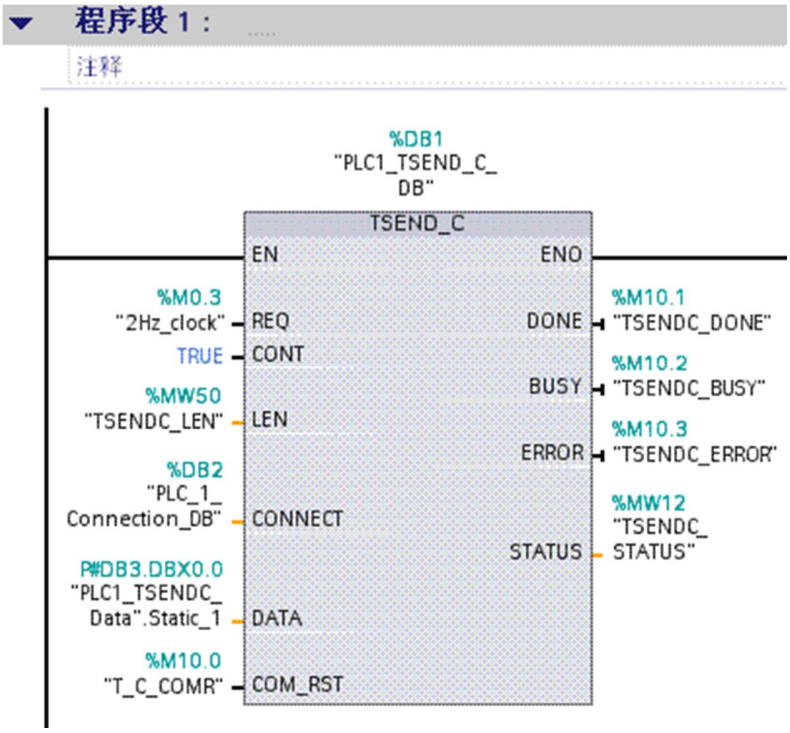


图 25 发送方的编程

接收方的程序如图所示，“LEN”参数赋一个常数“0”，以便实现动态数据长度传输。

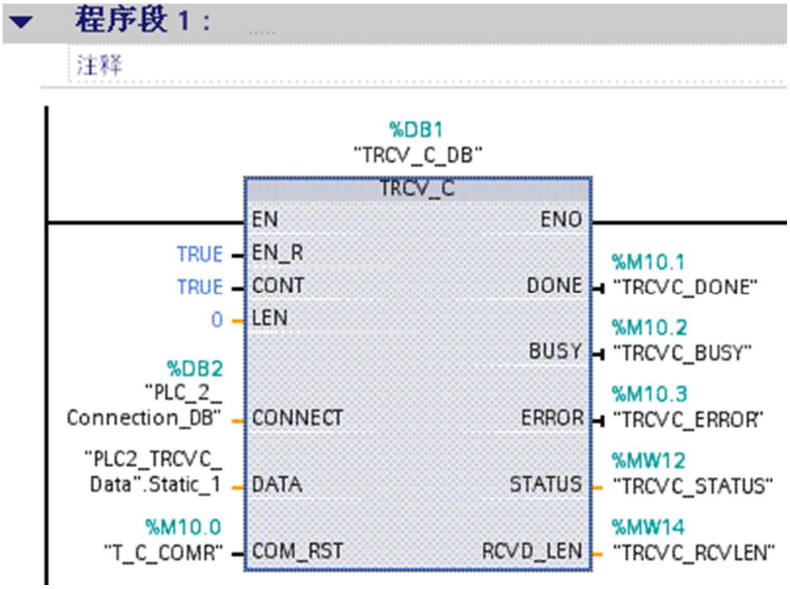


图 26 接收方的编程

□动态长度数据传输

要实现动态长度数据传输，需要将接收方的数据长度设为 0。

如果发送方数据长度“TSENDC_LEN"设为 3，则传送 3 个字节给接收方；

如果要将数据区的全部数据传送,可以将发送方数据长度“TSENDC_LEN"设为 0.

2.5 T-block 通信块的状态及错误代码

2.5.1 T-block 通信块的状态代码

表 1 状态代码

错误	状态（W#16#...）	描述
0	0000	执行任务无错误
0	7000	没有激活的任务
0	7001	启动任务处理，建立连接，等待连接伙伴
0	7002	数据正在发送或接收
0	7003	连接中断
0	7004	连接建立并被监视，无激活的任务

2.5.2 T-block 通信块的错误代码

表 2 错误代码

错误	状态（W#16#...）	描述
1	8070	所有内部背景存储区在使用中
1	8080	输入的通信口 ID 号无效
1	8081	超时，模块错误，内部错误
1	8085	LEN 参数值为 0，或者大于允许值
1	8086	CONNECT 参数在允许范围之外
1	8087	已经到达最大连接数，不允许额外的连接。
1	8088	LEN 参数大于 DATA 中所定义的长度；接收存储区太小。
1	8089	CONNECT 参数未指向 DB 块
1	8090	信息长度非法，模块非法，信息非法。
1	8091	参数化信息版本错误
1	8092	参数化信息中非法的长度记录
1	809A	CONNECT 参数指向的区域不符合连接描述的长度
1	809B	连接描述中的 local_device_id 与 CPU 不符
1	80A1	连接错误 • 定义的连接还未建立 • 定义的连接当前被结束；通过这个连接的传输不允许 • 接口正在重新初始化
1	80A3	试图终止一个不存在的连接
1	80A4	远程伙伴连接的 IP 地址非法。例如，远程 IP 与本地 IP 相同。
1	80A7	通信错误：在 TCON 指令完成前又调用了 TDISCON。
1	80B2	CONNECT 参数指向了一个由关键字 UNLINKED 生成的 DB 块
1	80B3	参数不一致： • 连接描述中有错误 • 本地端口（参数 local_tsap_id）已经在另一个连接中出现 • 连接描述中的 ID 与参数定义的 ID 不同
1	80B4	当使用 ISO on TCP 建立一个被动连接时，错误代码警告你所输入的 TSAP 不符合下面的地址要求： • 对于本地的一个 2 字节的 TSAP ID 值，第一个字节可以是 E0 或是 E1（十六进制），第二个字节是 00 或是 01。 • 对于本地的三个字节或大于三个字节的 TSAP ID 值，第一个字节可以是 E0 或是 E1（十六进制），第二个字节是 00 或是 01，所有其它字节应该是有效的 ASCII 字符。 • 对于本地的三个字节或大于三个字节的 TSAP ID 值，如果第一个字节不是 E0 或 E1（十六进制），那么所有 TSAP ID 必须是有效的 ASCII 字符。
1	80C3	所有连接资源都被使用了
1	80C4	临时通信错误： □□此时无法建立连接 □□接口正在接收新参数 □□TDISCON 当前正在删除已组态连接
1	8722	CONNECT 参数： 源区域无效： DB 中不存在该区域
1	873A	CONNECT 参数： 无法访问连接描述（例如，DB 不可用）
1	877F	CONNECT 参数： 内部错误，如无效 ANY 引用