HCFa
禾川科技

M系列 可编程逻辑控制器

通讯指令

指令手册

2024年3月V1.00
资料编码:ATC/MMXC032410

※ 目录

※ 前言

非常感谢您购买 M 系列控制器。该手册主要介绍控制器通讯指令，如以太网、RS485、RS232、CAN 等通讯指令。

- **面向的用户**

本手册面向 M 系列控制器编程和调试的技术人员。读者需要具备一定可编程控制器相关的基础知识和编程思维。

- **版本改版记录**

版本号	变更时间	更新说明
V1.00	2024/03/06	初版

版权声明

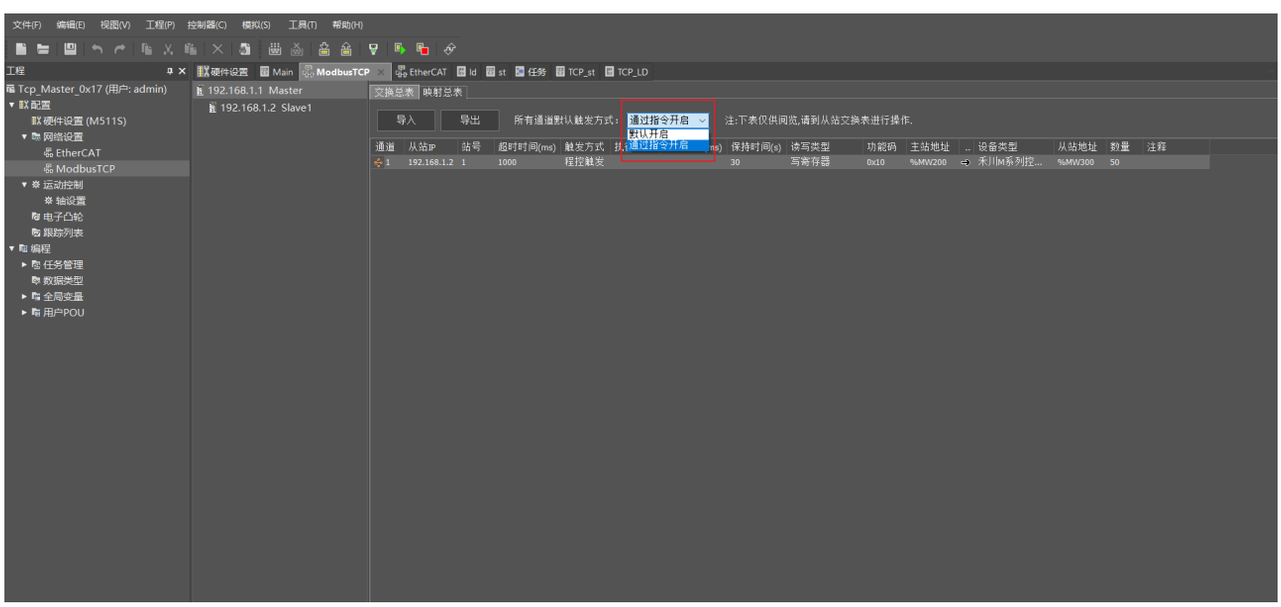
第 1 章　以太网通讯

1.1 ModbusTCP通讯使用概览

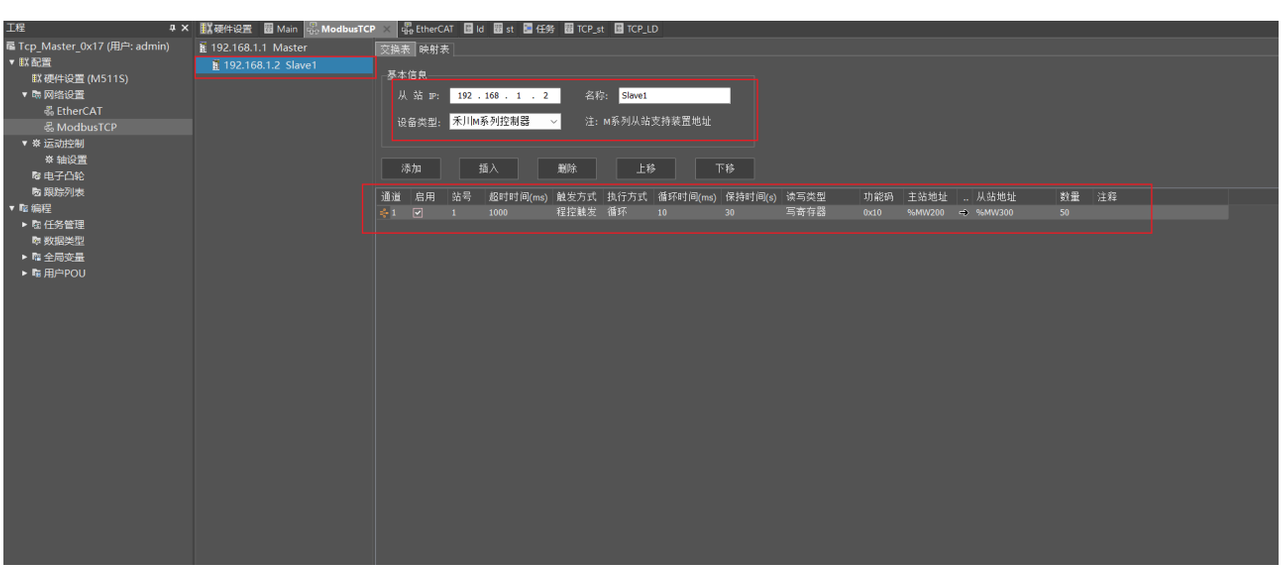
软件配置使用 ModbusTCP 功能：

步骤	项目	使用情况		说明
1	控制主站	软件配置界面选择通过指令开启	ModbusTCP_MasterRun	开启 ModbusTCP 主站功能
		软件配置界面选择默认开启	默认开启	开启 ModbusTCP 主站功能
		ModbusTCP_MasterStop		关闭 ModbusTCP 主站功能
		ModbusTCP_Masterstatus		获取 ModbusTCP 主站状态
2	配置通道	软件配置界面		配置指定数据交换通道
3	控制通道	软件配置界面选择程控触发	ModbusTCP_LinkRun	开启指定数据交换通道
		软件配置界面选择默认触发	默认开启	开启指定数据交换通道
		ModbusTCP_LinkStop		关闭指定数据交换通道
		ModbusTCP_GetLinkStatus		获取指定数据交换通道状态

步骤 1：对 ModbusTCP 主站功能总开关的操作，开启后可以使用 ModbusTCP 相关功能，软件配置可选择使用默认开启（默认运行）或通过 ModbusTCP_MasterRun 指令开启（指令触发后执行），通过 ModbusTCP_MasterStop 指令关闭，通过 ModbusTCP_Masterstatus 指令获取 ModbusTCP 主站状态。

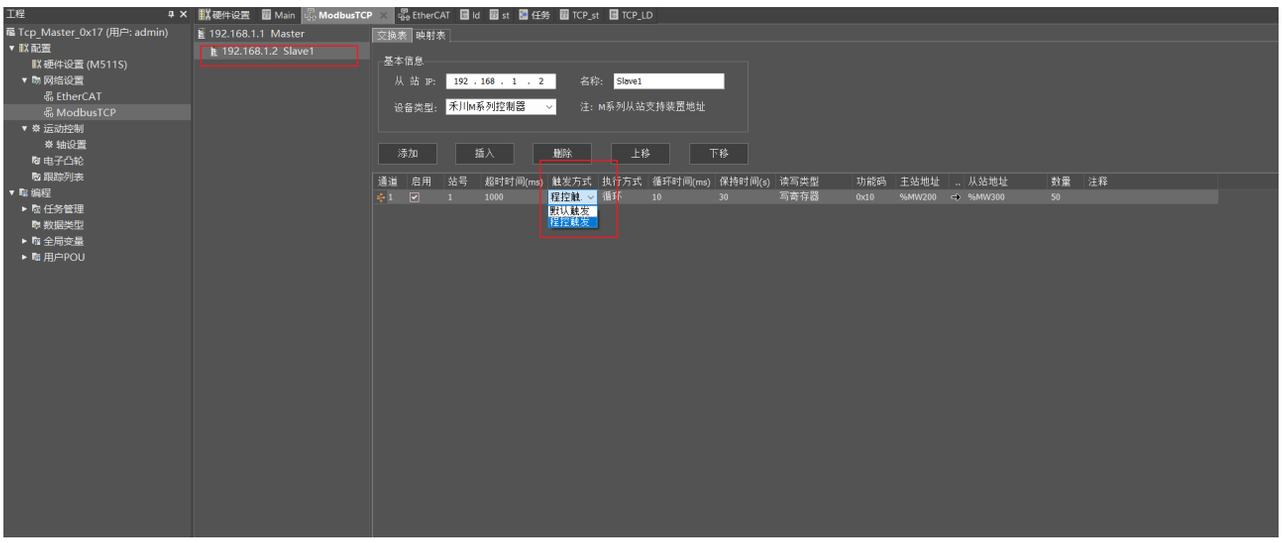


步骤 2：ModbusTCP 主站内有多个数据交换通道，之间相互独立，在软件配置界面可以分别配置通道参数



步骤 3：ModbusTCP 主站内有多个数据交换通道，之间相互独立，通过软件配置的通道可使用默认触发（默认运行）或程控触发（ModbusTCP_LinkRun 指令触发后执行）对指定通道进行开启。通过 ModbusTCP_LinkStop 指令关闭，通过 ModbusTCP_

GetLinkStatus 指令获取指定数据交换通道状态。



指令配置使用 ModbusTCP 功能：

步骤	流程	使用指令	说明
1	控制主站	ModbusTCP_MasterRun	开启 ModbusTCP 主站功能
		ModbusTCP_MasterStop	关闭 ModbusTCP 主站功能
		ModbusTCP_Masterstatus	获取 ModbusTCP 主站状态
2	配置通道	ModbusTCP_LinkConfig	配置指定数据交换通道
3	控制通道	ModbusTCP_LinkRun	开启指定数据交换通道
		ModbusTCP_LinkStop	关闭指定数据交换通道
		ModbusTCP_GetLinkStatus	获取指定数据交换通道状态

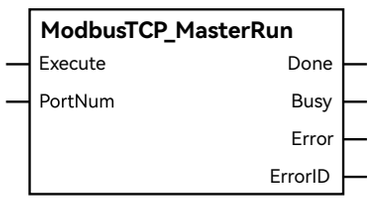
步骤 1：对 ModbusTCP 主站功能总开关的操作，开启后可以使用 ModbusTCP 相关功能，通过 ModbusTCP_MasterRun 指令开启，通过 ModbusTCP_MasterStop 指令关闭，通过 ModbusTCP_Masterstatus 指令获取 ModbusTCP 主站状态。

步骤 2：ModbusTCP 主站内有多个数据交换通道，之间相互独立，通过 ModbusTCP_LinkConfig 指令配置通道参数。

步骤 3：ModbusTCP 主站内有多个数据交换通道，之间相互独立，通过 ModbusTCP_LinkRun 指令开启。通过 ModbusTCP_LinkStop 指令关闭，通过 ModbusTCP_GetLinkStatus 指令获取指定数据交换通道状态。

1.2 ModbusTCP_MasterRun（ModbusTCP主站开启指令）

开启 ModbusTCP 主站功能。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ModbusTCP_ MasterRun	以太网口 ModbusTCP 管理	FB		ModbusTCP_MasterRun0 (Execute:= 参数 , PortNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令
PortNum	保留	UINT	保留	保留	保留

◆ 输出变量

名称	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。 值的含义请参阅“指令错误代码描述”。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

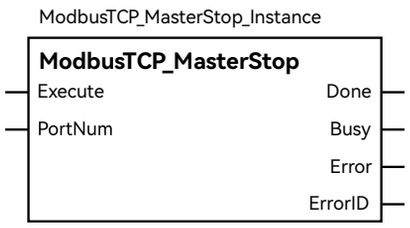
◆ 功能说明

• 基本功能说明

该指令用来启动 ModbusTCP 的主站功能。

1.3 ModbusTCP_MasterStop（ModbusTCP主站停止指令）

关闭 ModbusTCP 主站功能。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ModbusTCP_ MasterStop	以太网口 ModbusTCP 管理	FB		ModbusTCP_MasterStop_Instance (Execute:= 参数 , PortNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	保留	UINT	保留	保留	保留

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。 值的含义请参阅“指令错误代码描述”。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

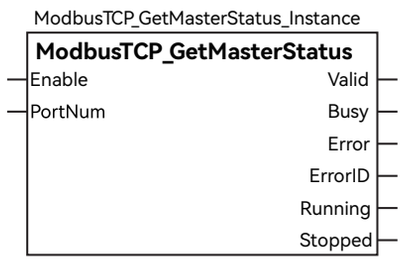
◆ 功能说明

• 基本功能说明

该指令用来关闭 ModbusTCP 主站功能。

1.4 ModbusTCP_GetMasterstatus（ModbusTCP主站状态获取指令）

获取主站状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ModbusTCP_ Masterstatus	读取主站状态	FB		ModbusTCP_GetMasterStatus(Enable:= 参数 , PortNum:= 参数 , Valid=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数 , Running=> 参数 , Stopped=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行，设为 FALSE，该指令不执行。
PortNum	保留	UINT	保留	保留	保留

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	指令正常执行时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。 值的含义请参阅“指令错误代码描述”。
Running	主站功能在开启状态	BOOL	TRUE / FALSE	主站在开启的状态下为 TRUE
Stopped	主站功能在关闭状态	BOOL	TRUE / FALSE	主站在关闭的状态下为 TRUE

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时 Error 由 FALSE 变为 TRUE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时
Running	ModbusTCP_MasterRun 指令开启时	ModbusTCP_MasterStop 指令开启时 Enable 由 TRUE 变为 FALSE 时
Stopped	ModbusTCP_MasterStop 指令开启时	ModbusTCP_MasterRun 指令开启时 Enable 由 TRUE 变为 FALSE 时

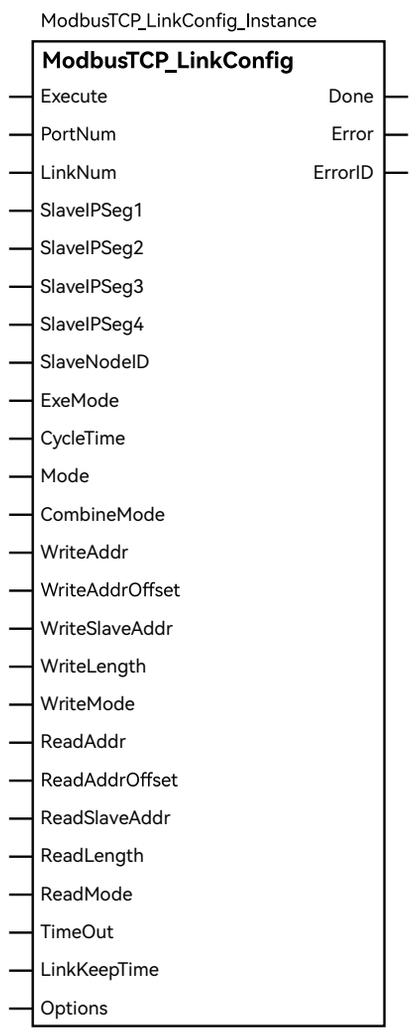
◆ 功能说明

• 基本功能说明

该指令用来读取 ModbusTCP 主站状态。通过该指令可以读取到 ModbusTCP 主站功能处于开启还是关闭状态。

1.5 ModbusTCP_LinkConfig（ModbusTCP参数配置指令）

参数配置指令。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ModbusTCP_LinkConfig	ModbusTCP 参数配置	FB		ModbusTCP_LinkConfig_Instance(Execute:= 参数 , PortNum:= 参数 , LinkNum:= 参数 , SlaveIPSeg1:= 参数 , SlaveIPSeg2:= 参数 , SlaveIPSeg3:= 参数 , SlaveIPSeg4:= 参数 , SlaveNodeID:= 参数 , ExeMode:= 参数 , CycleTime:= 参数 , Mode:= 参数 , CombineMode:= 参数 , WriteAddr:= 参数 , WriteAddrOffset:= 参数 , WriteSlaveAddr:= 参数 , WriteLength:= 参数 , WriteMode:= 参数 , ReadAddr:= 参数 , ReadAddrOffset:= 参数 , ReadSlaveAddr:= 参数 , ReadLength:= 参数 , ReadMode:= 参数 , TimeOut:= 参数 , LinkKeepTime:= 参数 , Options:= 参数 , Done=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	保留	UINT	保留	保留	保留
LinkNum	数据交换通道编号	UINT	参考通讯指令规格	不可缺省	设定 Modbus TCP 周期性数据交换的编号。
SlaveIPseg1	目标设备的 IP 地址（第一段）	USINT	0~255	0	设定目标设备 IP 地址的第一段。 如 IP 地址为 192.168.1.2，则第一段为 192
SlaveIPseg2	目标设备的 IP 地址（第二段）	USINT	0~255	0	设定目标设备 IP 地址的第二段。 如 IP 地址为 192.168.1.2，则第二段为 168
SlaveIPseg3	目标设备的 IP 地址（第三段）	USINT	0~255	0	设定目标设备 IP 地址的第三段。 如 IP 地址为 192.168.1.2，则第三段为 1
SlaveIPseg4	目标设备的 IP 地址（第四段）	USINT	0~255	0	设定目标设备 IP 地址的第四段。 如 IP 地址为 192.168.1.2，则第四段为 2

SlaveNodeID	目标设备的 Modbus 站号	USINT	0~255	0	设定目标设备的 Modbus 站号
ExeMode	循环模式	USINT	0~1	0	设定发送的模式 0：循环发送 1：仅发送一次
CycleTime	循环时间	USINT	0~65535	10	设定在循环发送的模式下，上一次数据发送和下一数据发送的间隔时间：单位 ms
Mode	读写地址的类型	USINT	0~1	0	设定读写从站地址的类型，通过该参数选择读写 Word 型地址或 Bit 型地址。 0：Word（字） 1：Bit（位）
Comebine-Mode	启用读写整合模式	BOOL	TRUE 或 FALSE	FALSE	该参数用于设定是否启用读写整 合功能 0：不启用（读和写数据分开进行）1：启用（读和写数据整合为一笔报 文且使用 0x17 功能码） 注意：仅在 Mode 为 0 时有效。
WriteAddr	写操作数据缓存的起始地址	UINT	%MW0~%MW32767 %QW0~%QW63	0	设定欲写入目标的数据在控制器 中的存放地址。写操作时，控制器 将该地址中的内容写入到目标设 备 中（该 地 址 由 WriteAddr 和 WriteAddrOffset 共同决定）。 如果本参数对应的输入为变量，则 需在该变量声明时为其指定正确 的地址。
WriteAddrOff-set	写操作数据缓存起始地址的偏移量	USINT	0~255	0	设定写操作缓存地址的偏移量。 实际写入的内容将从以 WriteAdd 为基础按 WriteAddrOffset 进行偏移后的地址中取出。 如果为 Word 读写，偏移单位为 1Word。例如，WriteAddr 指定地 址为 %MW0，WriteAddrOffset 为 1，表示起始地址为 %MW1。 如果为 Bit 读写，偏移单位为 1Bit。例如，WriteAddr 指定地址 为 %MW1，WriteAddrOffset 为 1，表示起始地址为 %MX2.1。
WriteSlaveAd-dr	写操作的目标地址	UINT	16#0~16#FFFF	0	设定写入操作的起始地址，该地址 为目标设备中的 Modbus 地址。
Writelength	写操作长度	UINT	字装置：0~100 位装置：0~256	0	设定写入数据的长度。 如果为 Word 读写，单位为 Word； 如果为 Bit 读写，单位为 Bit。
WriteMode	写操作模式（功能码）	USINT	16#0、16#06、16#10、16#05、16#0F	0	设定写操作时使用的功能码。 如果设定为 0，控制器将自主选择。
ReadAddr	读操作数据缓存的起始地址	UINT	%MW0~%MW32767 %QW0~%QW63	0	设定从目标设备中读取的数据在 控制器中的存放起始地址，该地址 由 ReadAddr 和 ReadAddrOffset 共同决定。 如果本参数对应的输入为变量，则 需在该变量声明时为其指定正确 的地址。
ReadAddrOff-set	读操作数据缓存起始地址的偏移量	USINT	0~255	0	设定读操作缓存地址的偏移量。 读取的内容将存放在以 ReadAddr 为基础按 ReadAddrOffset 进行偏移后的地 址中。 如果为 Word 读写，偏移单位为 1Word。例如，ReadAddr 指定地 址为 %MW100，ReadAddrOffset 为 1，表示起始地址 为 %MW101。 如果为 Bit 读写，偏移单位为 1Bit。例如，ReadAddr 指定地址 为 %MW100，ReadAddrOffset 为 1，表示起始地址 为 %MX200.1。
ReadSlaveAd-dr	读操作的目标地址	UINT	16#0~16#FFFF	0	设定读操作的起始地址，该地址 为目标设备中的 Modbus 地址。

Readlength	读操作长度	UINT	字装置：0~100 位装置：0~256 （0）	0	设定读取数据的长度。 如果为 Word 读写，单位为 Word； 如果为 Bit 读写，单位为 Bit。
ReadMode	读操作模式 (功能码)	USINT	16#0、16#03、16#04、16#01、16#02	0	设定读操作时使用的功能码。 如果设定为 0，控制器将自主选择。
TimeOut	通讯超时时间	UINT	0~65535	1000	设定等待目标设备响应的时间，单 位：ms。
LinkKeepTime	链接保持时间	UINT	0~65535	30	设定本设备和目标设备建立通讯连接可以保持的时间
Options	保留	保留	保留	保留	保留

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。 值的含义请参阅“指令错误代码描述”。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Execute 从 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时

◆ 功能说明

• 基本功能说明

该指令用于配置以太网 ModbusTCP 周期性数据交换（写和读）的相关参数，例如目标设备的 IP 地址、Modbus 站号、数据读写的目标地址、缓存地址、数据长度、功能码等。

注意：该指令仅进行参数配置，在 ModbusTCP 主站开启，通道数据交换下更改配置，重新触发该指令会立即生效。

• LinkNum

控制器中内建有多个 ModbusTCP 周期性数据交换通道，这些数据交换通道之间相互独立，可分别配置其参数，该指令的 LinkNum 参数则用于指定欲对哪一个通道进行参数配置。

• EnableLink

通过参数 EnableLink 可将 LinkNum 指定的 Modbus TCP 周期性数据交换通道配置为启用或关 闭，当 EnableLink 为 TRUE 时为启动，为 FALSE 为关闭。 注意：EnableLink 与其他参数一样，在指令触发执行时刻（Execute 上升沿时刻）被锁定，在 此时刻之前或之后改变 EnableLink 的值均无效。

• WriteAddr、WriteAddrOffset、WriteLength

周期性数据交换包含写操作和读操作。写操作是将写操作缓存地址中指定长度的数据写入到目标设备，写操作缓存地址由参数 WriteAddr、WriteAddrOffset 指定，写操作的数据长度由 WriteLength 指定。 参数 WriteAddr，WriteAddrOffset 指定是缓存区域的起始地址，该地址以基准地址 + 偏移的形式 进行指定，其中基准地址为 WriteAddr，偏移为 WriteAddrOffset。 Word 读写时（Mode=0），WriteAddrOffset 的单位为 Word，例如 WriteAddr 指定的基准地址 为 %MW1000，WriteLength 指定的长度为 5: 如果 WriteAddrOffset 为 0，则缓存起始地址为 %MW1000，

• 软件配置与指令配置的优先级说明

该指令配置的通道生效后优先级高于软件配置的通道。软件配置的通道仅会在程序下载和上电时生效一次，指令配置的通道不具有断电保持性，需要重新执行该指令。

1.6 ModbusTCP_LinkRun（ModbusTCP配置通道开启指令）

启动 ModbusTCP 指定的数据交换通道。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ModbusTCP_LinkRun	以太网口从站 link 管理	FB	ModbusTCP_LinkRun_Instance ModbusTCP_LinkRun Execute PortNum LinkNum Done Busy Error ErrorID	ModbusTCP_LinkRun1(Execute:= 参数 , PortNum:= 参数 , LinkNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令
PortNum	保留	UINT	保留	保留	保留
LinkNum	数据交换通道编号	UINT	参考通讯指令规格	不可缺省	设定 Modbus TCP 周期性数据交换的编号

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	指令执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。 值的含义请参阅“指令错误代码描述”。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

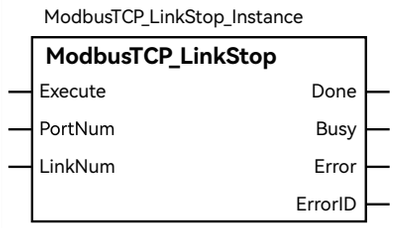
◆ 功能说明

• 基本功能说明

该指令用来启动指定的数据交换通道。

1.7 ModbusTCP_LinkStop（ModbusTCP配置通道关闭指令）

关闭 ModbusTCP 指定的数据交换通道。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ModbusTCP_LinkStop	以太网口从站 link 管理	FB		ModbusTCP_LinkStop(Execute:= 参数 , PortNum:= 参数 , LinkNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令
PortNum	保留	UINT	保留	保留	保留
LinkNum	数据交换通道编号	UINT	参考通讯指令规格	不可缺省	设定 Modbus TCP 周期性数据交换的编号

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	指令执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。 值的含义请参阅“指令错误代码描述”。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

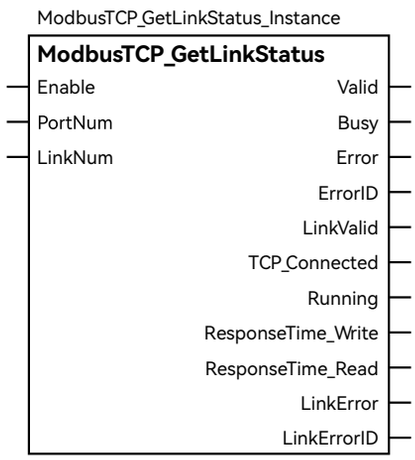
◆ 功能说明

• 基本功能说明

该指令关闭指定的数据交换通道。

1.8 ModbusTCP_GetLinkStatus（ModbusTCP通道状态取获指令）

获取 ModbusTCP 指定的数据交换通道状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Manage	ModbusTCP 指定的数据交换通道状态	FB		ModbusTCP_GetLinkStatus1(Enable:= 参数 , PortNum:= 参数 , LinkNum:= 参数 , Valid=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数 , LinkValid=> 参数 , TCP_Connected=> 参数 , Running=> 参数 , ResponseTime_Write=> 参数 , ResponseTime_Read=> 参数 , LinkError=> 参数 , LinkErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行
PortNum	保留	UINT	保留	保留	保留
LinkNum	通道	UINT	参考通讯指令规格	不可缺省	设定 Modbus TCP 周期性数据交换的编号

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD		指令执行异常时，输出错误代码。*1
LinkValid	配置完成	BOOL	TRUE / FALSE	LinkNum 指定通道的配置完成
TCP_Connected	TCP 连接	BOOL	TRUE / FALSE	和指定通道的从站建立连接时为 TRUE。
Running	数据交换正常	BOOL	TRUE / FALSE	指定通道数据交换正常时为 TRUE，异常时为 FALSE。
ResponseTime_Write	写回复时间	UINT	0~65535	从主站发出写命令至接收到从站回复主站命令的时间，单位：ms
ResponseTime_Read	读回复时间	UINT	0~65535	从主站发出读命令至接收到从站回复主站命令的时间，单位：ms
LinkError	数据交换异常	BOOL	BOOL/FALSE	指定通道数据交换异常时为 TRUE，正常时为 FALSE。如从站回复数据异常或者超时时，该位变为 TRUE，从站回复正常时，自动变为 FALSE
LinkErrorID	数据交换异常错误代码	WORD	0~65535	指定通道数据交换异常时，即 LinkError 为 TRUE 时，输出对应的错误代码。值的含义请参阅“指令错误代码描述”。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	指令正常执行时变为 TRUE
Busy	Enable 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时 Error 由 FALSE 变为 TRUE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时
LinkValid	LinkNum 指定通道的配置有效为 TRUE	LinkNum 指定通道的配置无效为 FALSE
TCP_Connected	和指定通道的从站建立连接时为 TRUE	和指定通道的从站断开连接时为 FALSE
Running	指定通道数据交换正常时为 TRUE	指定通道数据交换异常时为 FALSE
LinkError	指定通道数据交换异常时为 TRUE	指定通道数据交换正常时为 FALSE

◆ 功能说明

• 基本功能说明

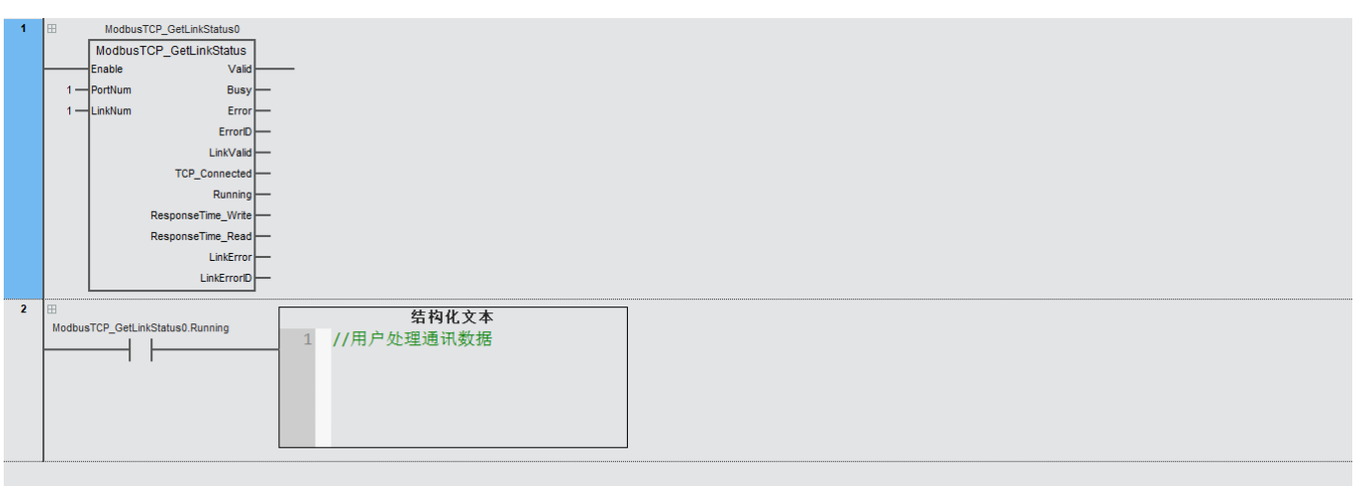
该指令用来获取 ModbusTCP 指定数据交换通道的状态，如获取指定通道当前数据交换正常或异常，或者主站和从站是否建立连接等。该指令的输出位 Running 为 TRUE 时表示数据交换正常；该指令的输出位 LinkError 为 TRUE 时表示数据交换异常，LinkErrorID 输出对应的错误码。如从站回复数据异常或者超时时，该指令的输出位 LinkError 位变为 TRUE，从站回复正常时，自动变为 FALSE。

◆ 示例程序

如下图所示，在 ModbusTCP 配置界面为通道 1 配置一笔数据，站号 1，默认触发，功能码 0x10。通过该指令获取通道 1 的状态。

类别	名称	分配到	数据类型	初始值	注释
VAR	ModbusTCP_GetLinkStatus0		BOOL		

梯形图 (LD):



构化文本 (ST):

```
ModbusTCP_GetLinkStatus1(Enable:=1 ,
    PortNum:= 1,
    LinkNum:=1 ,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    LinkValid=> ,
    TCP_Connected=> ,
    Running=> ,
    ResponseTime_Write=> ,
    ResponseTime_Read=> ,
    LinkError=> ,
    LinkErrorID=>
    );
    IF ModbusTCP_GetLinkStatus1.Running THEN
        //用户处理通讯数据
        ;
    END_IF;
```

- **程序说明:**

Running 为 TRUE 之后表示指定的通道数据正在运行交换，用户可以根据实际需求处理通讯数据。

1.9 ModbusTCP通讯使用范例

1.9.1 TCP数据交换范例一：软件配置ModbusTCP

• 目标需求

两个 M 系列 PLC 通过 TCP 协议的 0x10 功能码和 0x03 功能码进行数据交换

控制器的 IP 地址和端口如下:

项目	主站控制器	项目	从站控制器
IP 地址	192.168.1.1	IP 地址	192.168.1.2
端口号	502	端口号	502
地址	MW200	地址	MW300
地址	MW500	地址	MW400

• 需求分析

根据需求，需要主站控制器利用通道 1 的 0x10 功能码向从站控制器发送数据，0x03 功能码向从站控制器读取数据

步骤	项目	使用情况		说明
1	控制主站	软件配置界面选择通过指令开启	ModbusTCP_MasterRun	开启 ModbusTCP 主站功能
		软件配置界面选择默认开启	默认开启	开启 ModbusTCP 主站功能
		ModbusTCP_MasterStop		关闭 ModbusTCP 主站功能
		ModbusTCP_Masterstatus		获取 ModbusTCP 主站状态
2	配置通道	软件配置界面		配置指定数据交换通道
3	控制通道	软件配置界面选择程控触发	ModbusTCP_LinkRun	开启指定数据交换通道
		软件配置界面选择默认触发	默认开启	开启指定数据交换通道
		ModbusTCP_LinkStop		关闭指定数据交换通道
		ModbusTCP_GetLinkStatus		获取指定数据交换通道状态

• 软件配置

步骤一：将 ModbusTCP 主站启动方式选择为默认开启，下载后自动开启 ModbusTCP 主站功能

步骤二：添加从站，设置通道 1 的数据参数

步骤三：触发方式设置为默认触发，下载后或者控制器上电后自动开启通道 1 的数据开始交换

1.9.2 TCP数据交换范例二：软件与指令配置ModbusTCP

- **目标需求**

两个 M 系列 PLC 通过 TCP 协议的 0x10 功能码和 0x06 功能码进行数据交换

控制器的 IP 地址和端口如下：

项目	主站控制器	项目	从站控制器
IP 地址	192.168.1.1	IP 地址	192.168.1.2
端口号	502	端口号	502
地址	MW200	地址	MW300
地址	MW500	地址	MW400

- **需求分析**

根据需求，需要主站控制器利用通道 1 的 0x10 功能码向从站控制器发送数据，0x03 功能码向从站控制器读取数据

步骤	项目	使用情况		说明
1	控制主站	软件配置界面选择通过指令开启	ModbusTCP_MasterRun	开启 ModbusTCP 主站功能
		软件配置界面选择默认开启	默认开启	开启 ModbusTCP 主站功能
		ModbusTCP_MasterStop		关闭 ModbusTCP 主站功能
		ModbusTCP_Masterstatus		获取 ModbusTCP 主站状态
2	配置通道	软件配置界面		配置指定数据交换通道

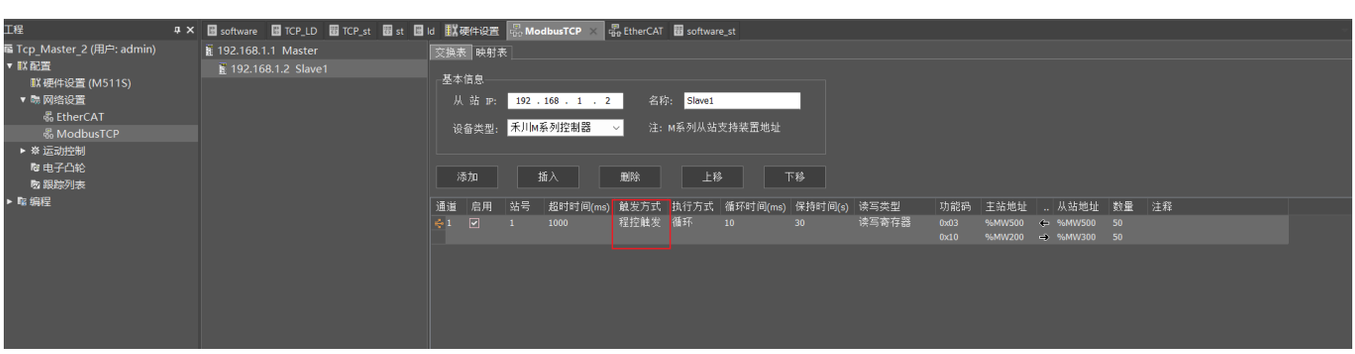
3	控制通道	软件配置界面选择程控触发	ModbusTCP_LinkRun	开启指定数据交换通道
		软件配置界面选择默认触发	默认开启	开启指定数据交换通道
		ModbusTCP_LinkStop		关闭指定数据交换通道
		ModbusTCP_GetLinkStatus		获取指定数据交换通道状态

- **软件配置**

步骤一：将 ModbusTCP 主站启动方式选择为通过指令启动，由指令启动 ModbusTCP 主站功能。

步骤二：添加从站，设置通道 1 的数据参数。

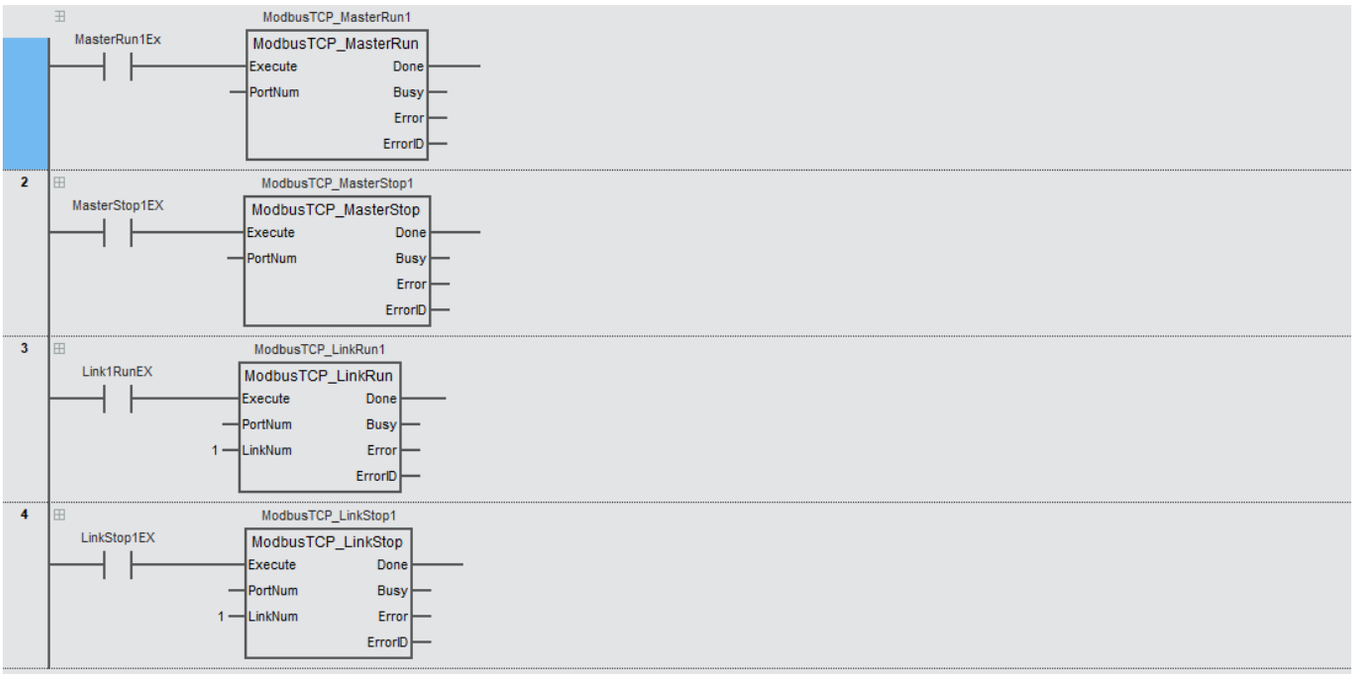
步骤三：触发方式设置为程控触发，由指令控制通道 1 的数据交换开启。



- **变量表**

类别	名称	分配到	数据类型	初始值	注释
VAR	MasterRun1Ex		BOOL		
VAR	MasterStop1EX		BOOL		
VAR	ModbusTCP_MasterRun1		ModbusTCP_MasterRun		
VAR	ModbusTCP_MasterStop1		ModbusTCP_MasterStop		
VAR	Link1RunEX		BOOL		
VAR	LinkStop1EX		BOOL		
VAR	ModbusTCP_LinkRun1		ModbusTCP_LinkRun		
VAR	ModbusTCP_LinkStop1		ModbusTCP_LinkStop		

梯形图（LD）



ST

```
ModbusTCP_MasterRun1(Execute:= MasterRun1Ex,
    PortNum:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
ModbusTCP_MasterStop(Execute:=MasterStop1EX ,
    PortNum:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
ModbusTCP_LinkRun1(Execute:= Link1RunEX,
    PortNum:= ,
    LinkNum:= 1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
ModbusTCP_LinkStop1(Execute:=LinkStop1EX ,
    PortNum:= ,
    LinkNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
```

- **程序说明**

第一步：当变量 MasterRun1Ex 设置为 TRUE 触发 ModbusTCP_MasterRun1 指令开启 ModbusTCP 主站功能。

第二步：当变量 LinkConfigEX 设置为 TRUE 触发 ModbusTCP_LinkConfig1 指令将配置好的数据参数写入到通道 1。

第三步：如果要停止通道 1 的数据交换，将变量 LinkStop1EX 设置为 TRUE 触发 ModbusTCP_LinkStop1 指令关闭通道 1 的数据交换，或将变量 MasterStop1EX 设置为 TRUE 触发 ModbusTCP_MasterStop1 指令关闭 ModbusTCP 主站功能。使用 ModbusTCP_LinkStop 指令关闭指定通道数据交换，用户如果配置其他通道，其他通道还可以进行数据交换；使用 ModbusTCP_MasterStop 关闭的是 ModbusTCP 主站功能，所有数据通道交换都会被停止。

1.9.3 TCP数据交换范例三：指令配置ModbusTCP

• 目标需求

两个 M 系列 PLC 通过 TCP 协议的 0x10 功能码和 0x06 功能码进行数据交换

控制器的 IP 地址和端口如下：

项目	主站控制器	项目	从站控制器
IP 地址	192.168.1.1	IP 地址	192.168.1.2
端口号	502	端口号	502
地址	MW200	地址	MW300
地址	MW500	地址	MW400

• 需求分析

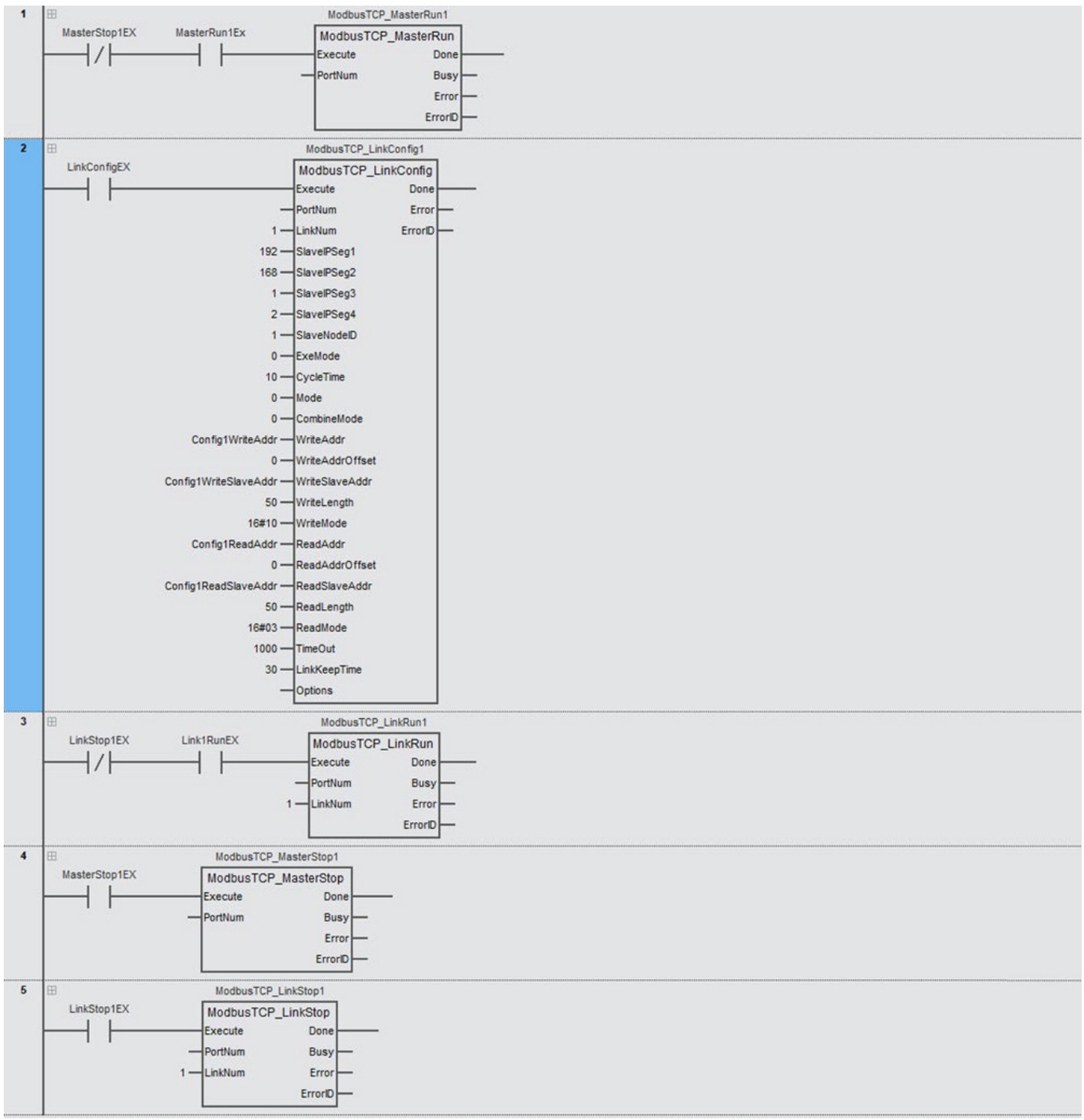
根据需求，需要主站控制器利用通道 1 的 0x10 功能码向从站控制器发送数据，0x03 功能码向从站控制器读取数据

步骤	流程	使用指令	说明
1	控制主站	ModbusTCP_MasterRun	开启 ModbusTCP 主站功能
		ModbusTCP_MasterStop	关闭 ModbusTCP 主站功能
		ModbusTCP_Masterstatus	获取 ModbusTCP 主站状态
2	配置通道	ModbusTCP_LinkConfig	配置数据交换通道 1（具体配置见指令）
3	控制通道	ModbusTCP_LinkRun	开启数据交换通道 1
		ModbusTCP_LinkStop	关闭数据交换通道 1
		ModbusTCP_GetLinkStatus	获取数据交换通道 1

• 变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	ModbusTCP_LinkConfig1		ModbusTCP_LinkConfig		
VAR	LinkConfigEX		BOOL		
VAR	Config1WriteAddr	%MW200	UINT		
VAR	Config1WriteSlaveAddr	%MW300	UINT		
VAR	Config1ReadAddr	%MW500	UINT		
VAR	Config1ReadSlaveAddr	%MW400	UINT		
VAR	ModbusTCP_MasterRun1		ModbusTCP_MasterRun		
VAR	MasterRun1Ex		BOOL		
VAR	ModbusTCP_MasterStop1		ModbusTCP_MasterStop		
VAR	MasterStop1EX		BOOL		
VAR	ModbusTCP_LinkRun1		ModbusTCP_LinkRun		
VAR	Link1RunEX		BOOL		
VAR	ModbusTCP_LinkStop1		ModbusTCP_LinkStop		
VAR	LinkStop1EX		BOOL		

梯形图（LD）


1
ModbusTCP_MasterRun1
MasterStop1EX
MasterRun1Ex
ModbusTCP_MasterRun
Execute
PortNum
Done
Busy
Error
ErrorID
2
ModbusTCP_LinkConfig1
LinkConfigEX
ModbusTCP_LinkConfig
Execute
PortNum
1 LinkNum
192 SlaveIPSeg1
168 SlaveIPSeg2
1 SlaveIPSeg3
2 SlaveIPSeg4
1 SlaveNodeID
0 ExeMode
10 CycleTime
0 Mode
0 CombineMode
Config1WriteAddr WriteAddr
0 WriteAddrOffset
Config1WriteSlaveAddr WriteSlaveAddr
50 WriteLength
16#10 WriteMode
Config1ReadAddr ReadAddr
0 ReadAddrOffset
Config1ReadSlaveAddr ReadSlaveAddr
50 ReadLength
16#03 ReadMode
1000 TimeOut
30 LinkKeepTime
Options
Done
Error
ErrorID
3
ModbusTCP_LinkRun1
LinkStop1EX
Link1RunEX
ModbusTCP_LinkRun
Execute
PortNum
1 LinkNum
Done
Busy
Error
ErrorID
4
ModbusTCP_MasterStop1
MasterStop1EX
ModbusTCP_MasterStop
Execute
PortNum
Done
Busy
Error
ErrorID
5
ModbusTCP_LinkStop1
LinkStop1EX
ModbusTCP_LinkStop
Execute
PortNum
1 LinkNum
Done
Busy
Error
ErrorID

ST

```
ModbusTCP_LinkConfig1(Execute:=LinkConfigEX ,
    PortNum:=  ,
    LinkNum:=1 ,
    SlaveIPSeg1:=192 ,
    SlaveIPSeg2:=168 ,
    SlaveIPSeg3:= 1,
    SlaveIPSeg4:= 1,
    SlaveNodeID:= 1,
    ExeMode:= 0,
    CycleTime:=10 ,
    Mode:= 0,
    CombineMode:= 0,
    WriteAddr:=Config1WriteAddr ,
    WriteAddrOffset:= 0,
    WriteSlaveAddr:= 0,
    WriteLength:=50 ,
    WriteMode:= 16#10,
    ReadAddr:= Config1ReadAddr,
    ReadAddrOffset:=0 ,
    ReadSlaveAddr:= Config1ReadSlaveAddr,
    ReadLength:=50 ,
    ReadMode:=16#05 ,
    TimeOut:= 1000,
    LinkKeepTime:=30 ,
    Options:= ,
    Done=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP MasterRun1(Execute:=  MasterRun1Ex AND (NOT MasterStop1EX) ,
    PortNum:=  ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP_LinkRun1(Execute:=  Link1RunEX AND (NOT LinkStop1EX) ,
    PortNum:=  ,
    LinkNum:=1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP_MasterStop1(Execute:= MasterStop1EX,
    PortNum:=  ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

    ModbusTCP_LinkStop1(Execute:= LinkStop1EX,
    PortNum:=  ,
    LinkNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );

```

- **程序说明**

第一步：将变量 MasterRun1Ex 设置为 TRUE 触发 ModbusTCP_MasterRun1 指令配置开启 ModbusTCP 主站功能。

第二步：将变量 LinkConfigEX 设置为 TRUE 触发 ModbusTCP_LinkConfig1 指令将配置好的数据参数写入到通道 1。

第三步：将变量 Link1RunEX 设置为 TRUE 触发 ModbusTCP_LinkRun1 指令开启通道 1 配置的数据开始交换。

第四步：如果要停止通道 1 的数据交换，将变量 LinkStop1EX 设置为 TRUE 触发 ModbusTCP_LinkStop1 指令关闭通道 1

的数据交换，或将变量 MasterStop1EX 设置为 TRUE 触发 ModbusTCP_MasterStop1 指令关闭 ModbusTCP 主站功能。使用 ModbusTCP_LinkStop 指令关闭指定通道数据交换，用户如果配置其他通道，其他通道还可以进行数据交换；使用 ModbusTCP_MasterStop 关闭的是 ModbusTCP 主站功能，所有数据通道交换都会被停止。

1.10 Socket通讯使用概览

指令配置使用 Socket 功能：

步骤	流程	使用指令	说明
1	配置 Socket 参数	Socket_Config	配置通信参数，包括 IP 地址、端口号、通信方式等。确保 PLC 与外部设备的通信参数相匹配。
2	建立连接	Socket_Open	配置 Socket 的工作模式，打开 Socket 数据交换启动功能。
		Socket_GetStatus	监控 Socket 状态。
3	发送和接收数据	Socket_Send	发送数据。
		Socket_Receive	接收数据。
4	关闭连接	关闭 Socket 数据交换	关闭 Socket 连接。

步骤 1：配置 Socket 参数，通过执行 Socket_Config 指令配置通信参数，设置目标设备的 IP 地址、端口号、连接模式（TCP 或 UDP）、连接保持时间等参数。

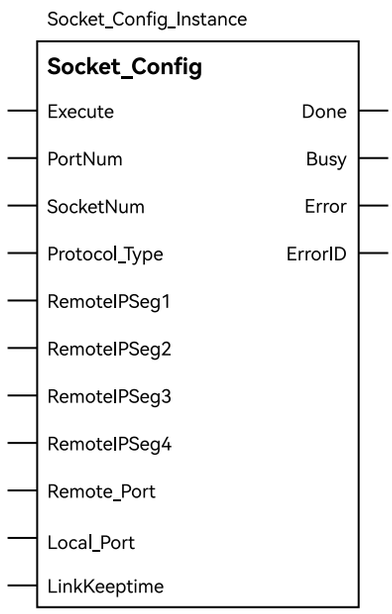
步骤 2：建立连接，通过执行 Socket_Open 设置工作模式并建立连接，Socket_Open 指令执行后，通过 Socket_GetStatus 指令获取 Socket 连接状态，Socket 连接判断。

步骤 3：发送数据，通过执行 Socket_Send 指令发送数据。 接收数据，通过执行 Socket_Receive 指令接收数据。

步骤 4: 关闭连接, 通过执行 Socket_Closed 指令关闭连接。

1.11 Socket_Config（Socket参数配置指令）

配置以太网口 Socket 功能相关参数。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Config	Socket 参数配置	FB		Socket_Config_Instance(Execute:= 参数 , PortNum:= 参数, SocketNum:= 参数 , Protocol_Type:= 参数 , RemoteIPSeg1:= 参数 , RemoteIPSeg2:= 参数 , RemoteIPSeg3:= 参数 , RemoteIPSeg4:= 参数 , Remote_Port:= 参数 , Local_Port:= 参数 , LinkKeeptime:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
SocketNum	Socket 编号	UINT	参考通讯指令规格	不可缺省	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。
Protocol_Type	Socket 连接模式	USINT	0 或 1	0	设定 Socket 连接模式。 0：UDP 1：TCP
RemoteIPSeg1	目标设备的 IP 地址（第一段）	USINT	0~255	0	设定目标设备 IP 地址的第一段。 如 IP 地址为 192.168.1.2，则第一段为 192
RemoteIPSeg2	目标设备的 IP 地址（第二段）	USINT	0~255	0	设定目标设备 IP 地址的第二段。 如 IP 地址为 192.168.1.2，则第二段为 168
RemoteIPSeg3	目标设备的 IP 地址（第三段）	USINT	0~255	0	设定目标设备 IP 地址的第三段。 如 IP 地址为 192.168.1.2，则第三段为 1
RemoteIPSeg4	目标设备的 IP 地址（第四段）	USINT	0~255	0	设定目标设备 IP 地址的第四段。 如 IP 地址为 192.168.1.2，则第四段为 2
Remote_Port	目标设备的端口号	UINT	0~65535	0	设定目标设备的端口号
Local_Port	本地端口号	UINT	0~65535	0	设定控制器本地端口号
LinkKeeptime	连接保持时间	UINT	0~65535	0	设置该 Socket 连接保持时间，超出该时间没有数据交换，将自动关闭该连接（单位：秒）

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。

Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Execute 从 TRUE 变为 FALSE 时
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时 Error 由 FALSE 变为 TRUE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时

◆ 功能说明

• 基本功能说明

本指令用于配置 Socket 功能的相关参数，例如目标设备的 IP 地址、端口号、连接保持时间等。

• Local_Port

UDP 模式时，Local_Port 可设置为 0，此时控制器将自动分配端口号。

TCP 模式时，Local_Port 不可设置为 0。

• 注意事项

Socket 数据交换启动后，不可执行本指令，待 Socket 数据交换关闭后，可执行本指令。

1.12 Socket_ConfigFrameLength（Socket数据长度配置指令）

配置以太网口 Socket 数据长度。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Config-FrameLength	Socket 数据长度配置	FB	Socket_ConfigFrameLength_Instance Socket_ConfigFrameLength Enable / PortNum / SocketNum / Length Valid / Busy / Error / ErrorID / SendCount / ReceiveCount / ReceiveLength	Socket_ConfigFrameLength_Instance(Enable:= 参数 , PortNum:= 参数, SocketNum:= 参数 , Length:= 参数 , Valid=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数 , SendCount=> 参数 , ReceiveCount=> 参数 , ReceiveLength=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数为 TRUE 时执行该指令。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
SocketNum	Socket 编号	UINT	参考通讯指令规格	不可缺省	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。
Length	Socket 数据长度	ARRAY[1..100] OF UINT	0~1460	0	设定 Socket 数据长度 (单位：Byte)

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	TRUE: 指令输出变量有效。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD		指令执行异常时，输出错误代码。*1
SendCount	Socket 发送次数	UINT		Socket 发送次数
ReceiveCount	Socket 接收次数	UINT		Socket 接收次数
ReceiveLength	Socket 发送数据长度	ARRAY[1..100] OF UINT		Socket 接收数据长度 (单位：Byte)

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	指令执行完成时。	Enable 由 TRUE 变为 FALSE 时

Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时 Error 由 FALSE 变为 TRUE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时

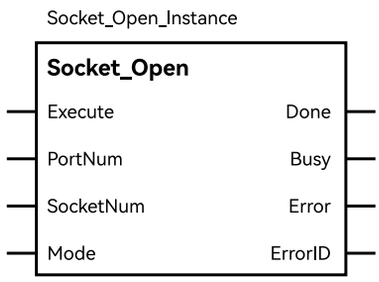
◆ 功能说明

• 基本功能说明

本指令为 Socket_Config 指令的扩展，用于配置以太网口 Socket 发送数据长度。

1.13 Socket_Open（Socket功能开启指令）

开启 Socket 功能。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Open	Socket 功能开启	FB		Socket_Open_Instance(Execute:= 参数 , PortNum:= 参数, SocketNum:= 参数 , Mode:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
SocketNum	Socket 编号	UINT	参考通讯指令规格	不可缺省	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。
Mode	Socket 工作模式	BOOL	TRUE 或 FALSE	FALSE	设定 Socket 工作模式。 TRUE：客户端 FALSE：服务器

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Execute 从 TRUE 变为 FALSE 时
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时 Error 由 FALSE 变为 TRUE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时

◆ 功能说明

• 基本功能说明

本指令用于开启 Socket 功能，即建立 TCP 连接或启用 UDP 功能。

启动后的状态可通过 Socket_GetStatus 指令获取。

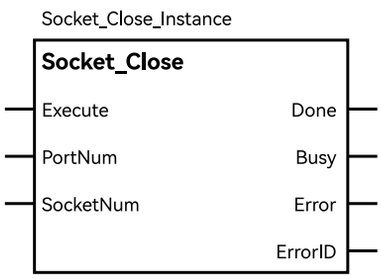
- **工作模式**

服务器模式 (Mode=FALSE)：控制器处于等待目标设备与其建立连接。

客户端模式（Mode=TRUE）：控制器主动与目标设备建立连接。

1.14 Socket_Close（Socket功能关闭指令）

关闭 Socket 功能。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Close	关闭 Socket 功能	FB		Socket_Close_Instance(Execute:= 参数 , PortNum:= 参数, SocketNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
SocketNum	Socket 编号	UINT	参考通讯指令规格	不可缺省	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Execute 从 TRUE 变为 FALSE 时
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时 Error 由 FALSE 变为 TRUE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时

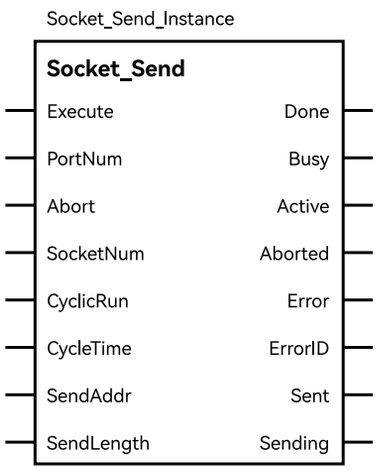
◆ 功能说明

• 基本功能说明

本指令用于关闭 Socket 功能，即断开 TCP 连接或关闭 UDP 功能。

1.15 Socket_Send（Socket数据发送指令）

发送 Socket 数据。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Send	发送 Socket 数据	FB		Socket_Send_Instance(Execute:= 参数 , PortNum:= 参数, Abort:= 参数 , SocketNum:= 参数 , CyclicRun:= 参数 , CycleTime:= 参数 , SendAddr:= 参数 , SendLength:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Aborted=> 参数 , Error=> 参数 , ErrorID=> 参数 , Sent=> 参数 , Sending=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
Abort	中断数据发送	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时中断数据发送。
SocketNum	Socket 编号	UINT	参考通讯指令规格	不可缺省	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。
CyclicRun	循环发送	BOOL	TRUE 或 FALSE	FALSE	设定是否循环发送缓存区中的数据。 TRUE: 循环发送 FALSE: 单次发送
CycleTime	循环发送时间间隔	UINT	0~65535	0	设定循环发送的时间间隔。(单位：ms)
SendAddr	发送数据缓存地址	USINT	%MB0~%MB65535	不可缺省	设定存放发送数据的起始地址。
SendLength	发送数据长度	UINT	1~1000	0	设定发送数据长度。(单位：Byte)

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	发送完成 (仅单次发送)	BOOL	TRUE / FALSE	单次发送模式下，数据发送完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令正在控制 Socket	BOOL	TRUE / FALSE	指令正常控制 Socket 时为 TRUE。
Aborted	指令被中断	BOOL	TRUE / FALSE	指令被中断时为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。

ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Sent	发送完成	BOOL	TRUE / FALSE	数据发送完成时变为 TRUE。
Sending	发送中	BOOL	TRUE / FALSE	数据发送中为 TRUE。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	发送完成时。	Execute 从 TRUE 变为 FALSE 时。
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时。 Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。
Active	指令开始控制 Socket 时。	Done 由 FALSE 变为 TRUE 时。 Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。
Aborted	指令被中断时。	Execute 从 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时。
Sent	发送完成时。	Execute 从 TRUE 变为 FALSE 时。 数据发送中还没有完成时。
Sending	数据发送中还没有完成时。	数据发送完成时。

◆ 功能说明

• 基本功能说明

本指令用于发送 Socket 数据。指令触发执行时，将缓存区域中的 SendLength 长度的数据通过 Socket 发送给目标设备，其中缓存区域的起始地址由参数 SendAddr 指定。

• 单次发送

当参数 CyclicRun 为 FALSE 时即为单次发送。指令每次触发执行仅发送一次数据，发送完成后，输出变量 Done 变为 TRUE，指令执行完成，如需再次发送，需要重新触发指令执行。

• 循环发送

当参数 CyclicRun 为 TRUE 时即为循环发送。指令触发执行后，将按 CycleTime 设定的间隔时间循环发送数据；如果 CycleTime 设置为 0，则发送完一次数据后，马上发送下一次。每发送完成一次，输出变量 Sent 变为 TRUE 一个周期，然后再变为 FALSE。准备发送数据时，Sending 为 TRUE，Sent 为 FALSE；发送数据完成时，Sending 为 FALSE，Sent 为 TRUE。

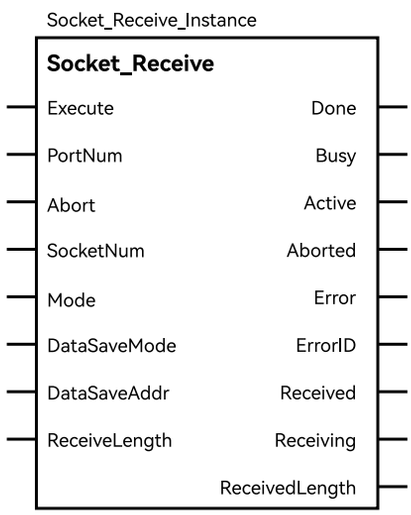
循环发送过程中，将 Abort 由 FALSE 设置为 TRUE，即可中断循环发送，同时输出变量 Aborted 将变为 TRUE。

• 注意事项

本指令需要在 Socket 功能开启用后执行，可通过 Socket_Open 指令开启 Socket 功能。

1.16 Socket_Receive（Socket数据接收指令）

接收 Socket 数据。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Receive	接收 Socket 数据	FB		Socket_Receive_Instance(Execute:= 参数 , PortNum:= 参数, Abort:= 参数 , SocketNum:= 参数 , Mode:= 参数 , DataSaveMode:= 参数 , DataSaveAddr:= 参数 , ReceiveLength:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Aborted=> 参数 , Error=> 参数 , ErrorID=> 参数 , Received=> 参数 , Receiving=> 参数 , ReceivedLength=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
Abort	中断数据接收	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时中断数据接收。
SocketNum	Socket 编号	UINT	参考通讯指令规格	0	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。
Mode	接收模式	USINT	0、1	0	设定接收数据的方式。 0: 循环接收 1: 单次接收
DataSaveMode	接收数据存储模式	USINT	0、1	0	设定接收数据向缓存区中存储的方式。 0: 接续 (Mode 为 0，循环接收时有效) 1: 覆盖
DataSaveAddr	接收数据缓存起始地址	USINT	%MB0~%MB65535	不可缺省	设定存放接收数据的起始地址。
ReceiveLength	接收数据长度	UINT	1~1000	0	设定接收数据长度。(单位：Byte)

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	数据接收完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令正在控制 Socket	BOOL	TRUE / FALSE	指令正常控制 Socket 时为 TRUE。
Aborted	指令被中断	BOOL	TRUE / FALSE	指令被中断时为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。

ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Received	接收完成	BOOL	TRUE / FALSE	数据接收完成时变为 TRUE。
Receiving	接收中	BOOL	TRUE / FALSE	数据接收中为 TRUE。
Receiv-edLength	一笔报文的数据长度	UINT	0~65535	一笔报文的数据长度，单位 :Byte。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Execute 从 TRUE 变为 FALSE 时。
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时。 Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。
Active	指令开始控制 Socket 时。	Done 由 FALSE 变为 TRUE 时。 Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。
Aborted	指令被中断时。	Execute 从 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时。
Received	数据接收完成时。	Execute 从 TRUE 变为 FALSE 时。 数据准备接收中。
Receiving	数据准备接收中。	数据接收完成时。

◆ 功能说明

• 基本功能说明

本指令用于接收 Socket 数据。指令触发执行时，将接收到的数据按 DataSaveMode 设定模式存入到缓存区域中，其中缓存区域的起始地址由参数 DataSaveAddr 指定。

• 单次接收

当参数 Mode 为 1 时表示单次接收。需要执行单次接收时，先将 Mode 设置为 1，再执行指令（Execute 从 FALSE 变为 TRUE）。

单次接收时，指令每次触发执行仅接收一次数据，接收到数据后，输出变量 Received 和 Done 变为 TRUE，指令执行完成，如需再次接收，需要重新触发指令执行。

如果实际接收数据长度超过输入变量 ReceiveLength 指定的长度，该指令会报错，接收数据缓存地址也接收不到数据。遇到这种情况，可以将输入变量 ReceiveLength 指定的长度变大，然后重新执行该指令。

• 连续接收

当参数 Mode 为 0 时即为连续接收。需要执行连续接收数据时，先将 Mode 设置为 0，再执行指令（Execute 从 FALSE 变为 TRUE）。

连续接收数据时，接收到一笔新的数据，Received 变为 TRUE 一个周期，然后再变为 FALSE。Receiving 和 Received 为互斥，即 Receiving 为 TRUE 时，Received 为 FALSE；Receiving 为 FALSE 时，Received 为 TRUE。

连续接收数据时，可通过参数 DataSaveMode 设置数据存储方式为接续或覆盖。

当设置 DataSaveMode（值为 1）为覆盖模式时，新接收的数据始终从接收数据缓存起始地址开始存储。ReceivedLength 为最新接收一笔数据的数据长度（一笔报文中有多少个字节的数据）, Receiving 和 Received 为互斥。

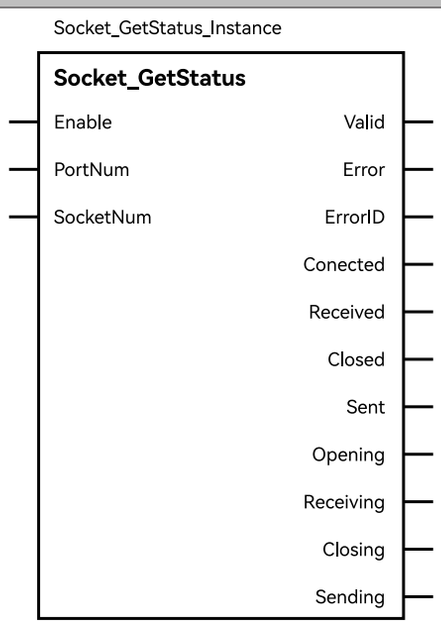
当设置 DataSaveMode（值为 0）为接续模式时，新接收到的数据存放在前一笔数据的后面。当实际接收数据长度小于 ReceiveLength 指定的长度时, 接续接收数据, ReceivedLength 为最新接收一笔数据的数据长度 (一笔报文中有多少个字节的数据), Receiving 和 Received 为互斥。当实际接收数据长度大于或等于 ReceiveLength 指定的长度时，输出变量 Done 变为 TRUE，Receiving 为 FALSE，Received 为 TRUE，不再接收数据。如果需要再接收数据，需要重新触发该指令执行。

• 注意事项

本指令需要在 Socket 功能开启用后执行，可通过 Socket_Open 指令开启 Socket 功能。

1.17 Socket_GetStatus（Socket状态获取指令）

获取 Socket 状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_GetStatus	Socket_GetStatus 状态获取	FB		Socket_GetStatus_Instance(Enable:= 参数 , PortNum:= 参数, SocketNum:= 参数 , Valid=> 参数 , Error=> 参数 , ErrorID=> 参数 , Conected=> 参数 , Received=> 参数 , Closed=> 参数 , Sent=> 参数 , Opening=> 参数 , Receiving=> 参数 , Closing=> 参数 , Sending=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
SocketNum	Socket 编号	UINT	参考通讯指令规格	不可缺省	指定 Socket 的编号。不同的 Socket 通过 SocketNum 进行区分标识。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	指令正常执行时变为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Conected	连接建立完成	BOOL	TRUE / FALSE	连接成功时为 TRUE，反之为 FALSE。
Received	接收成功	BOOL	TRUE / FALSE	接收成功时为 TRUE，反之为 FALSE。
Closed	连接已关闭	BOOL	TRUE / FALSE	连接关闭时为 TRUE，反之为 FALSE。
Sent	数据发送完成	BOOL	TRUE / FALSE	发送完成时为 TRUE，反之为 FALSE。
Opening	Socket 数据交换启用中	BOOL	TRUE / FALSE	Socket 数据交换启用中为 TRUE，反之为 FALSE。
Receiving	数据接收中	BOOL	TRUE / FALSE	Socket 数据接收中为 TRUE，反之为 FALSE。
Closing	Socket 数据交换关闭中	BOOL	TRUE / FALSE	Socket 数据交换关闭中为 TRUE，反之为 FALSE。
Sending	数据发送中	BOOL	TRUE / FALSE	Socket 数据发送中为 TRUE，反之为 FALSE。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时。 错误消除时。
Conected	连接建立成功时。	Enable 从 TRUE 变为 FALSE 时。 连接断开时。
Received	数据接收完成时。	Enable 从 TRUE 变为 FALSE 时。
Closed	连接关闭时。	Enable 从 TRUE 变为 FALSE 时。 连接建立成功时。
Sent	发送完成时。	Enable 从 TRUE 变为 FALSE 时。 重新发送时。
Opening	Socket 数据交换开始启用时。	Enable 从 TRUE 变为 FALSE 时。 Socket 数据交换启用完成时。
Receiving	开始数据接收时。	Enable 从 TRUE 变为 FALSE 时。 数据接收完成时。
Closing	Socket 数据交换开始关闭时。	Enable 从 TRUE 变为 FALSE 时。 Socket 数据交换关闭完成时。
Sending	开始发送数据时。	Enable 从 TRUE 变为 FALSE 时。 数据发送完成时。

◆ 功能说明

• 基本功能说明

本指令用于获取 Socket 的当前状态。

1.18 Socket_Manage（Socket管理指令）

开启或关闭 Socket 功能。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Socket_Manage	以太网口 Socket 管理	FB	Socket_Manage_Instance Socket_Manage Enable — Valid PortNum — Ready EnableSocket — PhysicalLinkBroken	Socket_Manage_Instance(Enable:= 参数 , PortNum:= 参数, EnableSocket:= 参数 , Valid=> 参数 , Ready=> 参数 , PhysicalLinkBroken=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
PortNum	以太网硬件接口编号	UINT	保留	保留	保留
EnableSocket	开启或关闭 Socket 功能	BOOL	TRUE 或 FALSE	FALSE	启用或关闭 Socket 功能。 TRUE: 开启 FALSE: 关闭

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	指令正常执行时变为 TRUE。
Ready	Socket 功能已开启	BOOL	TRUE / FALSE	TRUE:Socket 已开启; FALSE:Socket 已关闭。
PhysicalLinkBroken	物理连接已断开	BOOL	TRUE / FALSE	TRUE: 物理连接已断开; FALSE: 物理连接正常。

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时
Ready	Enable 为 TRUE 且 Socket 功能成功开启时。	Enable 由 TRUE 变为 FALSE 时 EnableSocket 由 TRUE 变为 FALSE 时 以太网口物理连接断开时
PhysicalLinkBroken	EnableSocket 为 TRUE 且主机以太网口物理连接断开时。	Enable 由 TRUE 变为 FALSE 时 以太网口物理连接重新连接时

◆ 功能说明

• 基本功能说明

本指令用于开启或关闭 Socket 功能。通过 Socket 功能可实现自定义以太网数据的发送和接收。

注意 1：M511S 控制器版本为 1.01.07 以及之前的版本，M312 控制器版本为 1.01.06 以及之前的版本，需要先通过执行 Socket_Manage 指令打开 Socket 功能，才能使用其他 Socket 相关指令。

注意 2：M511S 控制器版本为 1.01.07 之后的版本，M312 控制器版本为 1.01.06 之后的版本时，不建议使用本指令。不使用本指令时，其他 Socket 相关指令可以直接使用；可以使用 Socket_Open 开启 Socket 功能，可以使用 Socket_Close 关闭 Socket 功能。

如果使用本指令，本指令执行（Enable 为 TRUE）时，仍可以通过输入变量 EnableSocket 开启或关闭 Sockset 功能。EnableSocket 为 TRUE 时，Socket 功能开启；EnableSocket 为 FALSE 时，Socket 功能关闭。

1.19 以太网Socket通讯范例TCP

1.19.1 Socket数据交换范例

• 目标需求

两个 M 系列 PLC 通过 Socket 功能进行自定义数据交换。

控制器的 IP 地址和端口如下:

项目	客户端控制器	项目	服务端控制器
IP 地址	192.168.1.1	IP 地址	192.168.1.10
端口号	10000	端口号	20000

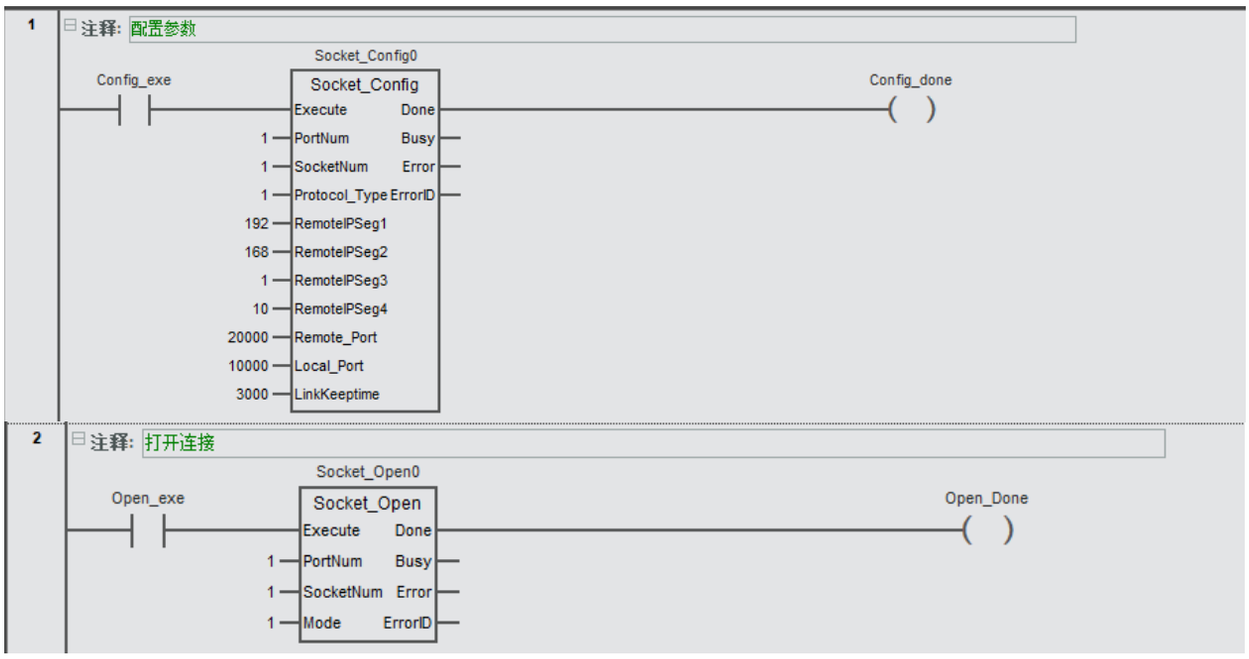
• 需求实现

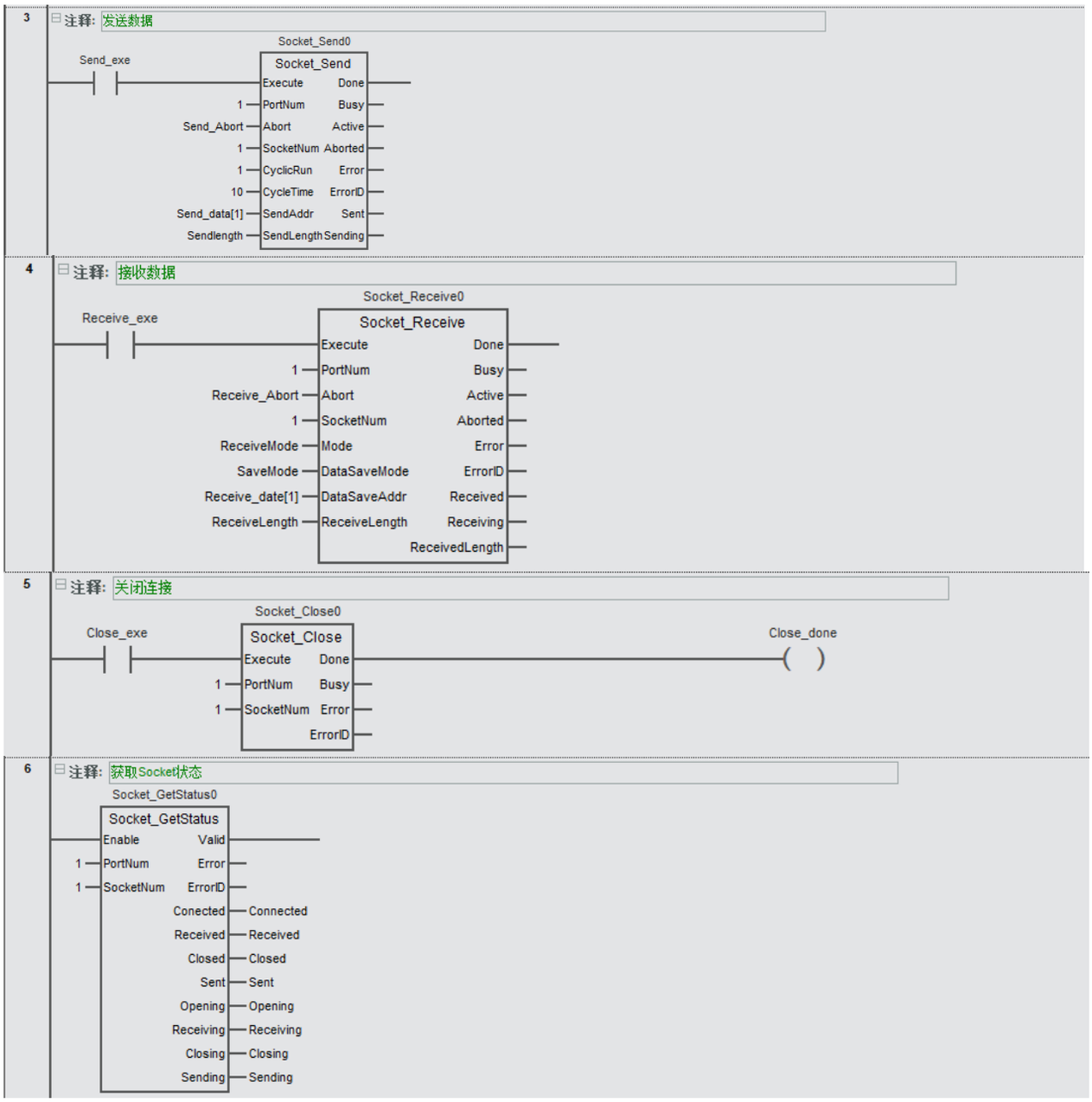
根据需求分析编写程序。

客户端控制器程序变量规划如下表:

范围	名称	地址	类型	初始值	注释
VAR	Socket_Config0		Socket_Config		Socket 配置指令
VAR	Config_exe		BOOL		Socket 配置指令执行条件
VAR	Config_done		BOOL		Socket 配置参数完成位
VAR	Socket_Open0		Socket_Open		Socket 端口打开指令
VAR	Open_exe		BOOL		Socket 端口打开执行条件
VAR	Open_Done		BOOL		Socket 端口打开完成位
VAR	Socket_Send0		Socket_Send		Socket 发送数据指令
VAR	Send_Exe		BOOL		Socket 发送数据执行条件
VAR	Send_Abort		BOOL		中断 Socket 发送数据
VAR	Send_data	%MB1000	ARRAY[1..100] OF USINT	[100(0)]	Socket 发送缓存地址
VAR	SendLength		UINT	100	Socket 发送长度
VAR	Socket_Receive0		Socket_Receive		Socket 数据接收指令
VAR	Receive_Exe		BOOL		Socket 数据接收执行条件
VAR	Receive_Abort		BOOL		中断 Socket 数据接收
VAR	ReceiveMode		USINT		Socket 数据接收方式
VAR	SaveMode		USINT	1	Socket 接收数据存储模式，0 为接续存储数据 ,1 为覆盖存储数据
VAR	ReceiveBuffer	%MB2000	ARRAY[1..100] OF USINT	[100(0)]	Socket 数据接收缓存地址
VAR	ReceiveLength		UINT	100	Socket 接收数据长度
VAR	Socket_Close0		Socket_Close		Socket 端口关闭指令
VAR	Close_exe		BOOL		Socket 端口关闭执行条件
VAR	Close_Done		BOOL		Socket 端口关闭完成位
VAR	Socket_GetStatus0		Socket_GetStatus		Socket 工作状态获取指令
VAR	Connected		BOOL		连接建立成功
VAR	Received		BOOL		接收成功
VAR	Closed		BOOL		连接已关闭
VAR	Sent		BOOL		数据发送完成
VAR	Opening		BOOL		Socket 连接端口打开中
VAR	Receiving		BOOL		数据接收中
VAR	Closing		BOOL		Socket 连接端口关闭
VAR	sending		BOOL		数据发送中

- **客户端控制器梯形图程序如下：**





- **客户端控制器ST程序如下：**

```
// 配置参数
Socket_Config0(Execute:=Config_exe ,
    PortNum:= 1,
    SocketNum:=1 ,
    Protocol_Type:=1 ,
    RemoteIPSeg1:=192 ,
    RemoteIPSeg2:=168 ,
    RemoteIPSeg3:=1 ,
    RemoteIPSeg4:=10 ,
    Remote_Port:=20000 ,
    Local_Port:=10000 ,
    LinkKeeptime:=3000 ,
```

```
    Done=>Config_done
    );
// 打开连接
Socket_Open0(Execute:= Open_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Mode:=1 ,
    Done=>Open_Done
    );
// 发送数据
Socket_Send0(Execute:=Send_Exe ,
    PortNum:= 1,
    Abort:=Send_Abort ,
    SocketNum:=1 ,
    CyclicRun:=1 ,
    CycleTime:=10 ,
    SendAddr:= Send_data[1] ,
    SendLength:=SendLength ,
    );
// 接收数据
Socket_Receive0(Execute:=Receive_Exe ,
    PortNum:=1 ,
    Abort:= Receive_Abort,
    SocketNum:=1 ,
    Mode:=ReceiveMode ,
    DataSaveMode:=SaveMode ,
    DataSaveAddr:= Receive_date[1] ,
    ReceiveLength:=ReceiveLength ,
    );
// 关闭连接
Socket_Close0(Execute:=Close_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Done=>Close_done ,
    );
// 获取 Socket 状态
Socket_GetStatus0(Enable:=1,
    PortNum:=1 ,
    SocketNum:= 1,
    Conected=>Connected,
```

```
Received=>Received,
Closed=>Closed,
Sent=>Sent,
Opening=>Opening,
Receiving=>Receiving,
Closing=>Closing,
Sending=>Sending
);
```

• 程序流程说明：

步骤 1: 配置 Socket 参数 (Socket_Config) , 包括连接方式、远端设备 IP、端口号、连接保持时间。将变量 Config_exe 设置为 TRUE 以触发 Socket 参数配置指令执行, 变量 Config_done 为 TRUE 即配置成功。

步骤 2：打开 Socket 端口建立连接（Socket_Open）, 包括 Socket 工作模式 , 工作模式为客户端（Mode=1）需要确保外部服务器已经打开并处于可连接的状态, 然后将 Open_exe 变量设置为 TRUE, 以触发 Socket_Open 指令来打开 Socket 端口并建立连接。可以使用指令 Socket_GetStatus 获取 Socket 的状态，通过检查 Connected 变量是否为 TRUE 来判断 Socket 连接是否成功建立。

步骤 3: 发送和接收 Socket 数据。在 Socket 端口打开且连接建立成功后, 方可执行发送和接收指令, 本例中, 将变量 Send_Exe 设置为 TRUE 即可触发 Socket 发送, (Socket_Send1.CyclicRun) 设置为 1 为循环发送, 将变量 Receive1_Exe 设置为 TRUE 即可触发 Socket 接收。

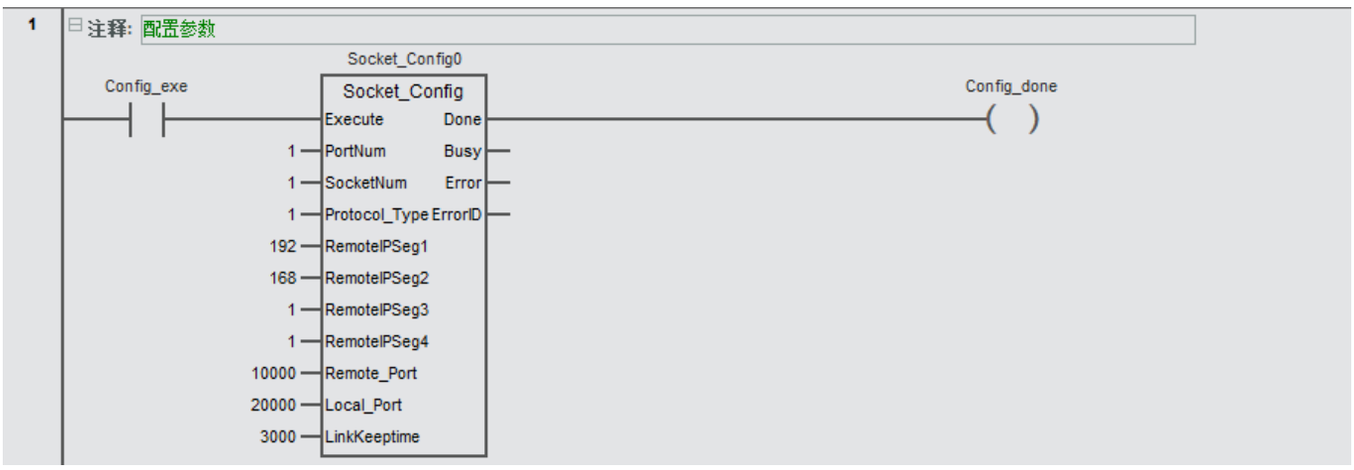
步骤 4: 关闭 Socket 端口以断开连接 (Socket_Close) , 变量 Close_exe 为 TRUE 触发关闭 Socket 端口指令, 变量 Close_done 为 TRUE 即关闭成功。

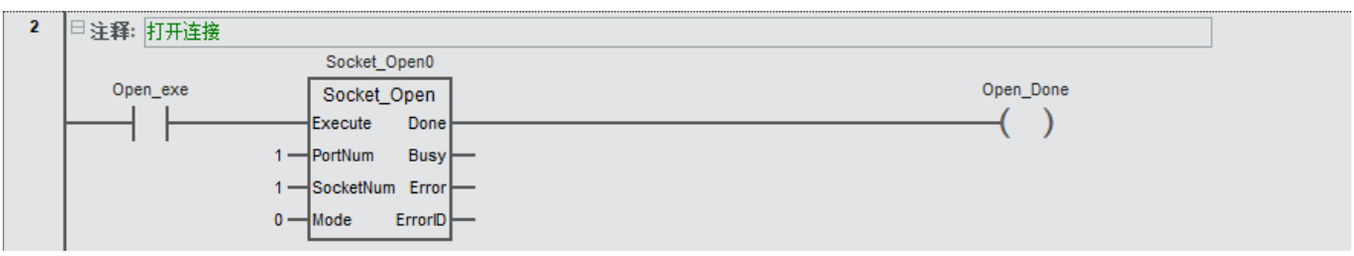
• 服务器端控制器程序变量规划如下表：

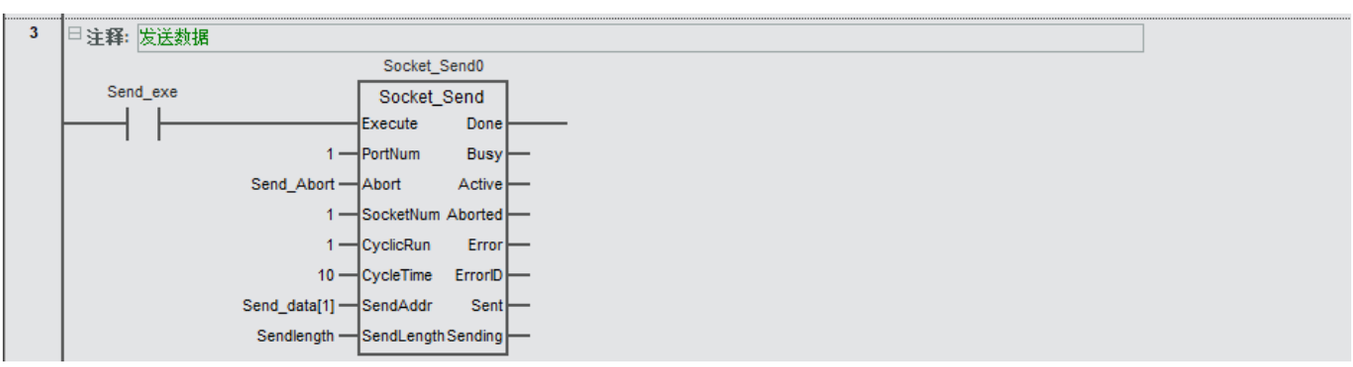
范围	名称	地址	类型	初始值	注释
VAR	Socket_Config0		Socket_Config		Socket 配置指令
VAR	Config_exe		BOOL		Socket 配置指令执行条件
VAR	Config_done		BOOL		Socket 配置参数完成位
VAR	Socket_Open0		Socket_Open		Socket 端口打开指令
VAR	Open_exe		BOOL		Socket 端口打开执行条件
VAR	Open_Done		BOOL		Socket 端口打开完成位
VAR	Socket_Send0		Socket_Send		Socket 发送数据指令
VAR	Send_Exe		BOOL		Socket 发送数据执行条件
VAR	Send_Abort		BOOL		中断 Socket 发送数据
VAR	Send_data	%MB3000	ARRAY[1..100] OF USINT	[100(0)]	Socket 发送缓存地址
VAR	SendLength		UINT	100	Socket 发送长度
VAR	Socket_Receive0		Socket_Receive		Socket 数据接收指令
VAR	Receive_Exe		BOOL		Socket 数据接收执行条件
VAR	Receive_Abort		BOOL		中断 Socket 数据接收
VAR	ReceiveMode		USINT		Socket 数据接收方式
VAR	SaveMode		USINT	1	Socket 接收数据存储模式，0 为接续存储数据 ,1 为覆盖存储数据
VAR	ReceiveBuffer	%MB4000	ARRAY[1..100] OF USINT	[100(0)]	Socket 数据接收缓存地址
VAR	ReceiveLength		UINT	100	Socket 接收数据长度
VAR	Socket_Close0		Socket_Close		Socket 端口关闭指令
VAR	Close_exe		BOOL		Socket 端口关闭执行条件
VAR	Close_Done		BOOL		Socket 端口关闭完成位
VAR	Socket_GetStatus0		Socket_GetStatus		Socket 工作状态获取指令

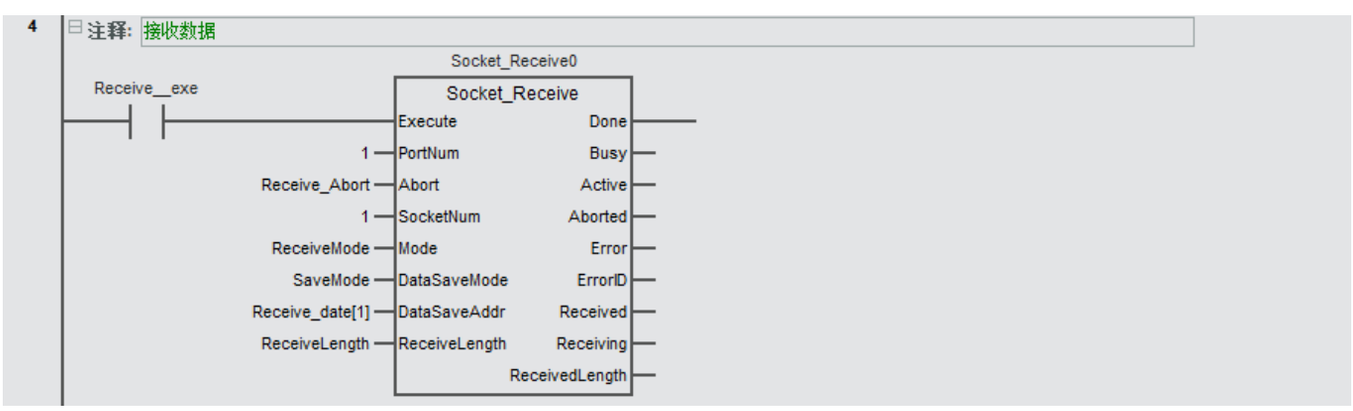
VAR	Connected		BOOL		连接建立成功
VAR	Received		BOOL		接收成功
VAR	Closed		BOOL		连接已关闭
VAR	Sent		BOOL		数据发送完成
VAR	Opening		BOOL		Socket 连接端口打开中
VAR	Receiving		BOOL		数据接收中
VAR	Closing		BOOL		Socket 连接端口关闭
VAR	Sending		BOOL		数据发送中

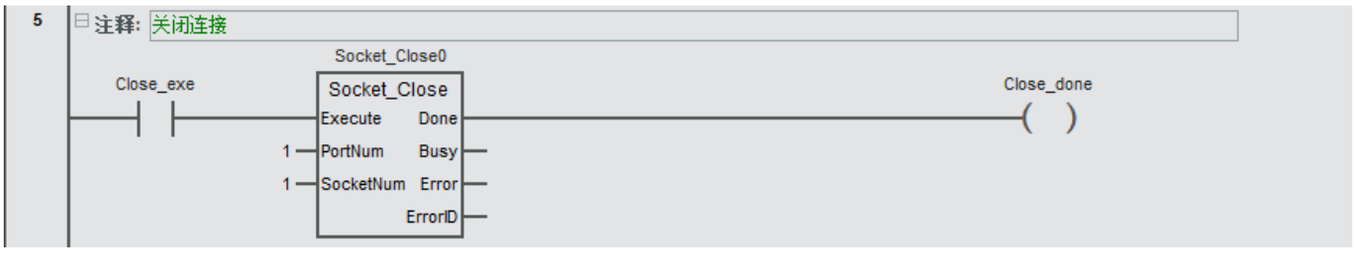
- **服务端控制器梯形图程序如下：**











- **服务端控制器ST程序如下：**

```
// 配置参数
Socket_Config0(Execute:=Config_exe ,
    PortNum:= 1,
    SocketNum:=1 ,
    Protocol_Type:=1 ,
    RemoteIPSeg1:=192 ,
    RemoteIPSeg2:=168 ,
    RemoteIPSeg3:=1 ,
    RemoteIPSeg4:=1 ,
    Remote_Port:=10000 ,
    Local_Port:=20000 ,
    LinkKeeptime:=3000 ,
    Done=>Config_done
    );
// 打开连接
Socket_Open0(Execute:= Open_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Mode:=0 ,
    Done=>Open_Done
    );
// 发送数据
Socket_Send0(Execute:=Send_Exe ,
```

```
    PortNum:= 1,
    Abort:=Send_Abort ,
    SocketNum:=1 ,
    CyclicRun:=1 ,
    CycleTime:=10 ,
    SendAddr:= Send_data[1] ,
    SendLength:=SendLength ,
    );
// 接收数据
Socket_Receive0(Execute:=Receive_Exe ,
    PortNum:=1 ,
    Abort:= Receive_Abort,
    SocketNum:=1 ,
    Mode:=ReceiveMode ,
    DataSaveMode:=SaveMode ,
    DataSaveAddr:= Receive_date[1] ,
    ReceiveLength:=ReceiveLength ,
    );
// 关闭连接
Socket_Close0(Execute:=Close_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Done=>Close_done ,
    );
/// 获取 Socket 状态

Socket_GetStatus0(Enable:=1,
    PortNum:=1 ,
    SocketNum:= 1,
    Conected=>Connected,
    Received=>Received,
    Closed=>Closed,
    Sent=>Sent,
    Opening=>Opening,
    Receiving=>Receiving,
    Closing=>Closing,
    Sending=>Sending
    );
```

- **程序流程说明：**

步骤 1：配置 Socket 参数（Socket_Config），包括连接方式、远端设备 IP、端口号、连接保持时间。将变量 Config_exe 设

置为 TRUE 以触发 Socket 参数配置指令执行，变量 Config_done 为 TRUE 即配置成功。

步骤 2：打开 Socket 端口建立连接（Socket_Open），包括 Socket 工作模式，工作模式为客户端（Mode=0）将 Open_exe 变量设置为 TRUE，以触发 Socket_Open 指令来打开 Socket 端口并建立连接。可以使用指令 Socket_GetStatus 获取 Socket 的状态，通过检查 Opening 变量是否为 TRUE 来判断 Socket 端口在打开状态。

步骤 3：发送和接收 Socket 数据。在 Socket 端口打开且连接建立成功后，方可执行发送和接收指令，本例中，将变量 Send_Exe 设置为 TRUE 即可触发 Socket 发送，(Socket_Send1.CyclicRun) 设置为 1 为循环发送，将变量 Receive1_Exe 设置为 TRUE 即可触发 Socket 接收。

步骤 4：关闭 Socket 端口以断开连接（Socket_Close），变量 Close_exe 为 TRUE 触发关闭 Socket 端口指令，变量 Close_done 为 TRUE 即关闭成功。

1.20 以太网Socket通讯范例UDP

1.20.1 Socket数据交换范例

• 目标需求

两个 M 系列 PLC 通过 Socket 功能进行自定义数据交换。

控制器的 IP 地址和端口如下:

项目	控制器1	项目	控制器2
IP 地址	192.168.1.1	IP 地址	192.168.1.10
端口号	10000	端口号	20000

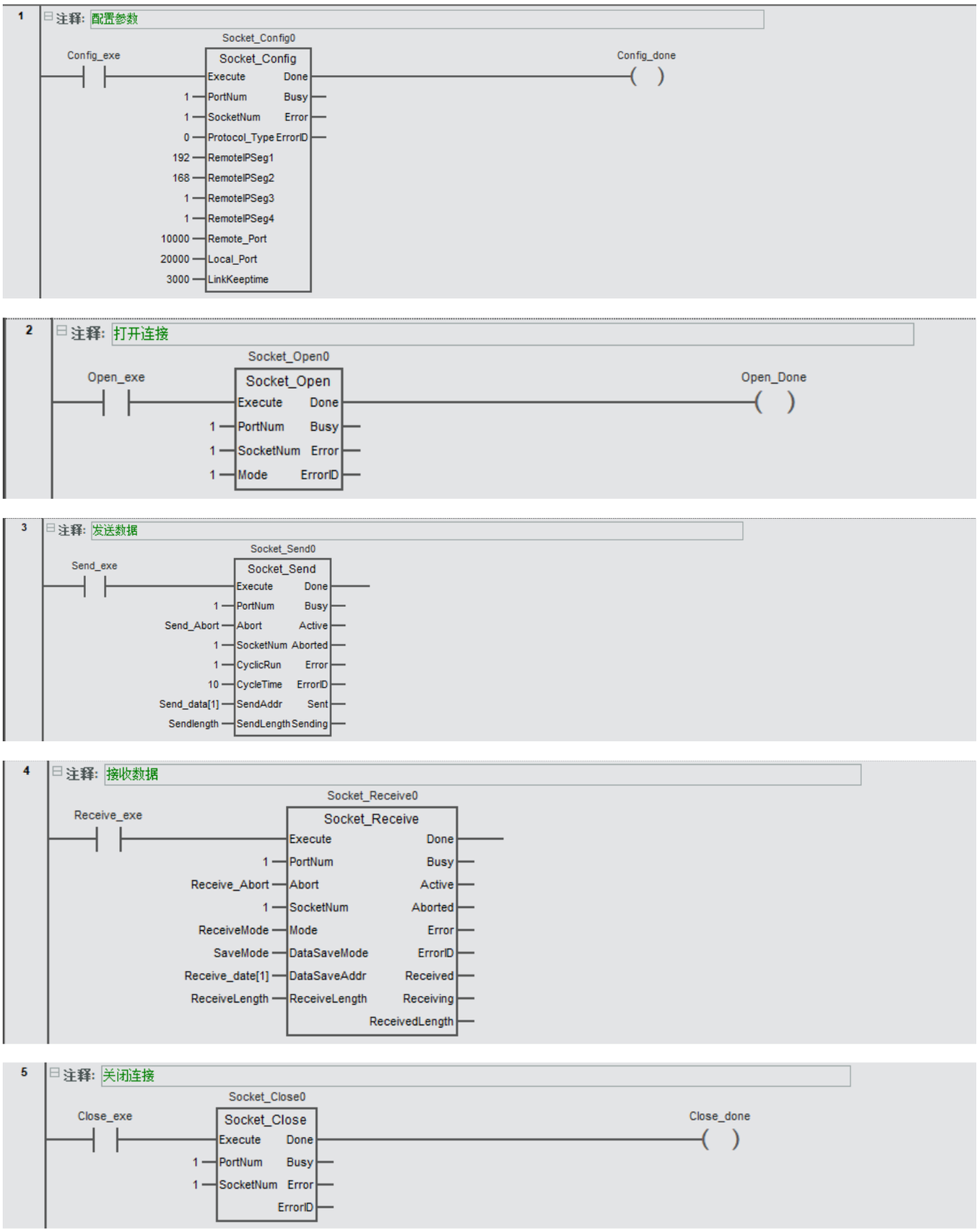
• 需求实现

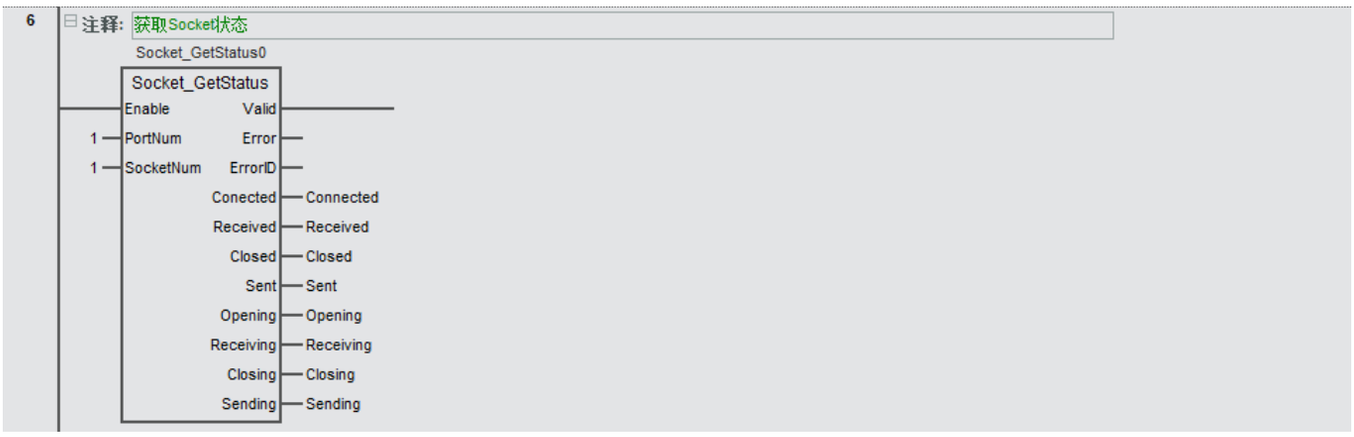
根据需求分析编写程序。

控制器 1 程序变量规划如下表:

范围	名称	地址	类型	初始值	注释
VAR	Socket_Config0		Socket_Config		Socket 配置指令
VAR	Config_exe		BOOL		Socket 配置指令执行条件
VAR	Config_done		BOOL		Socket 配置参数完成位
VAR	Socket_Open0		Socket_Open		Socket 端口打开指令
VAR	Open_exe		BOOL		Socket 端口打开执行条件
VAR	Open_Done		BOOL		Socket 端口打开完成位
VAR	Socket_Send0		Socket_Send		Socket 发送数据指令
VAR	Send_Exe		BOOL		Socket 发送数据执行条件
VAR	Send_Abort		BOOL		中断 Socket 发送数据
VAR	Send_data	%MB1000	ARRAY[1..100] OF USINT	[100(0)]	Socket 发送缓存地址
VAR	SendLength		UINT	100	Socket 发送长度
VAR	Socket_Receive0		Socket_Receive		Socket 数据接收指令
VAR	Receive_Exe		BOOL		Socket 数据接收执行条件
VAR	Receive_Abort		BOOL		中断 Socket 数据接收
VAR	ReceiveMode		USINT		Socket 数据接收方式
VAR	SaveMode		USINT	1	Socket 接收数据存储模式，0 为接续存储数据 ,1 为覆盖存储数据
VAR	ReceiveBuffer	%MB2000	ARRAY[1..100] OF USINT	[100(0)]	Socket 数据接收缓存地址
VAR	ReceiveLength		UINT	100	Socket 接收数据长度
VAR	Socket_Close0		Socket_Close		Socket 端口关闭指令
VAR	Close_exe		BOOL		Socket 端口关闭执行条件
VAR	Close_Done		BOOL		Socket 端口关闭完成位
VAR	Socket_GetStatus0		Socket_GetStatus		Socket 工作状态获取指令
VAR	Connected		BOOL		连接建立成功
VAR	Received		BOOL		接收成功
VAR	Closed		BOOL		连接已关闭
VAR	Sent		BOOL		数据发送完成
VAR	Opening		BOOL		Socket 连接端口打开中
VAR	Receiving		BOOL		数据接收中
VAR	Closing		BOOL		Socket 连接端口关闭
VAR	sending		BOOL		数据发送中

- **控制器1梯形图程序如下：**





- **控制器1ST程序如下：**

```
// 配置参数
Socket_Config0(Execute:=Config_exe ,
    PortNum:= 1,
    SocketNum:=1 ,
    Protocol_Type:=0 ,
    RemoteIPSeg1:=192 ,
    RemoteIPSeg2:=168 ,
    RemoteIPSeg3:=1 ,
    RemoteIPSeg4:=10 ,
    Remote_Port:=20000 ,
    Local_Port:=10000 ,
    LinkKeeptime:=3000 ,
    Done=>Config_done
    );
// 打开连接
Socket_Open0(Execute:= Open_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Mode:=0 ,
    Done=>Open_Done
    );
// 发送数据
Socket_Send0(Execute:=Send_Exe ,
    PortNum:= 1,
    Abort:=Send_Abort ,
    SocketNum:=1 ,
    CyclicRun:=1 ,
    CycleTime:=10 ,
    SendAddr:= Send_data[1] ,
    SendLength:=SendLength ,
```

```
    );
// 接收数据
Socket_Receive0(Execute:=Receive_Exe ,
    PortNum:=1 ,
    Abort:= Receive_Abort,
    SocketNum:=1 ,
    Mode:=ReceiveMode ,
    DataSaveMode:=SaveMode ,
    DataSaveAddr:= Receive_date[1] ,
    ReceiveLength:=ReceiveLength ,
    );
// 关闭连接
Socket_Close0(Execute:=Close_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Done=>Close_done ,
    );
// 获取 Socket 状态
Socket_GetStatus0(Enable:=1,
    PortNum:=1 ,
    SocketNum:= 1,
    Conected=>Connected,
    Received=>Received,
    Closed=>Closed,
    Sent=>Sent,
    Opening=>Opening,
    Receiving=>Receiving,
    Closing=>Closing,
    Sending=>Sending
    );
```

- **程序流程说明：**

步骤 1：配置 Socket 参数（Socket_Config），包括连接方式、远端设备 IP、端口号、连接保持时间。将变量 Config_exe 设置为 TRUE 以触发 Socket 参数配置指令执行，变量 Config_done 为 TRUE 即配置成功。

步骤 2：打开 Socket 端口建立连接（Socket_Open），UDP 模式下不需要设置连工作模式 , Open_exe 变量设置为 TRUE，以触发 Socket_Open 指令来打开 Socket 端口并建立连接。可以使用指令 Socket_GetStatus 获取 Socket 的状态，通过检查 Opening 变量是否为 TRUE 来判断 Socket 端口是否打开，当 UDP 两端同时处于 Opening 状态则建立连接。

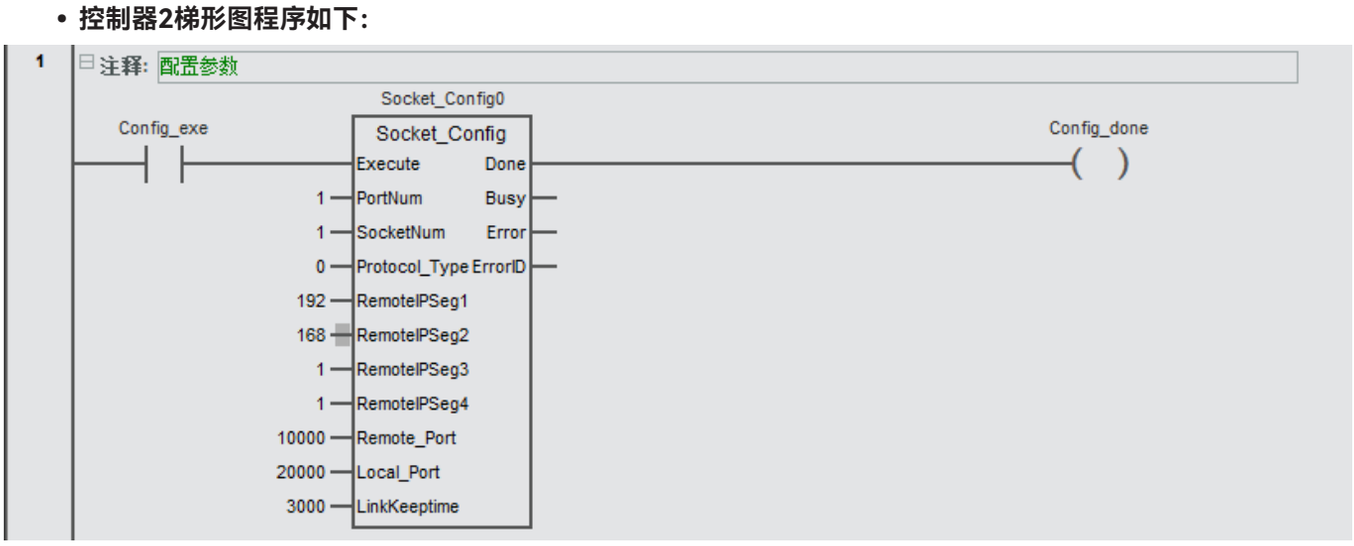
步骤 3：发送和接收 Socket 数据。在 Socket 端口打开且连接建立成功后，方可执行发送和接收指令，本例中，将变量 Send_Exe 设置为 TRUE 即可触发 Socket 发送，(Socket_Send1.CyclicRun) 设置为 1 为循环发送，将变量 Receive1_Exe 设置为 TRUE 即可触发 Socket 接收。

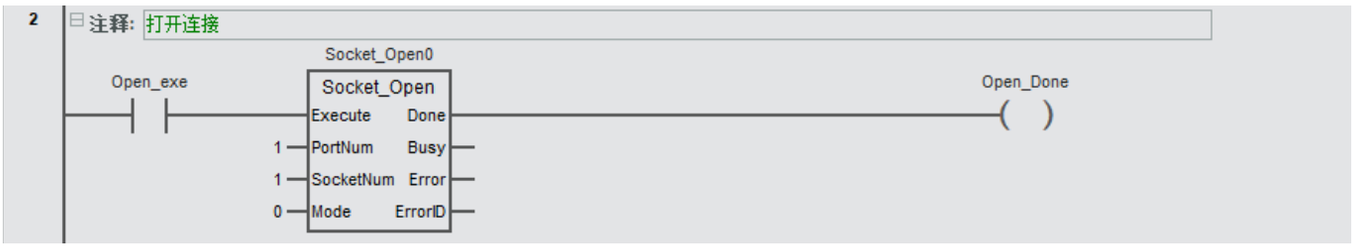
步骤 4：关闭 Socket 端口以断开连接（Socket_Close）, 变量 Close_exe 为 TRUE 触发关闭 Socket 端口指令，变量 Close_done 为 TRUE 即关闭成功。

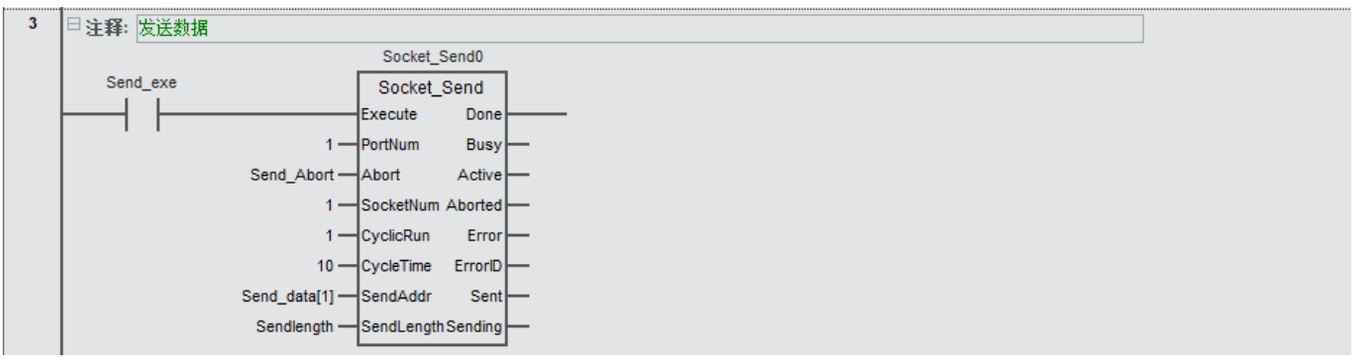
- **控制器2程序变量规划如下表：**

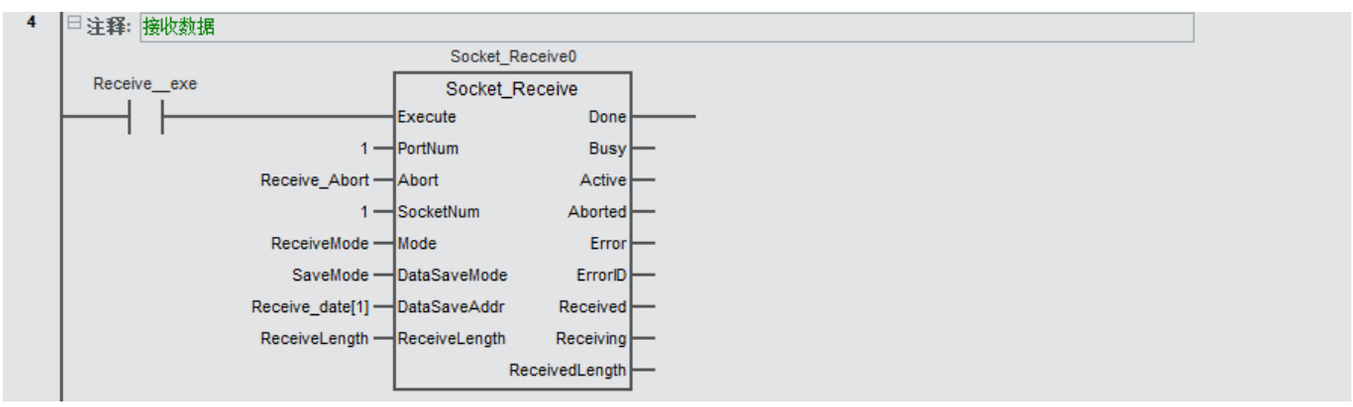
范围	名称	地址	类型	初始值	注释
VAR	Socket_Config0		Socket_Config		Socket 配置指令
VAR	Config_exe		BOOL		Socket 配置指令执行条件
VAR	Config_done		BOOL		Socket 配置参数完成位
VAR	Socket_Open0		Socket_Open		Socket 端口打开指令
VAR	Open_exe		BOOL		Socket 端口打开执行条件
VAR	Open_Done		BOOL		Socket 端口打开完成位
VAR	Socket_Send0		Socket_Send		Socket 发送数据指令
VAR	Send_Exe		BOOL		Socket 发送数据执行条件
VAR	Send_Abort		BOOL		中断 Socket 发送数据
VAR	Send_data	%MB3000	ARRAY[1..100] OF USINT	[100(0)]	Socket 发送缓存地址
VAR	SendLength		UINT	100	Socket 发送长度
VAR	Socket_Receive0		Socket_Receive		Socket 数据接收指令
VAR	Receive_Exe		BOOL		Socket 数据接收执行条件
VAR	Receive_Abort		BOOL		中断 Socket 数据接收
VAR	ReceiveMode		USINT		Socket 数据接收方式
VAR	SaveMode		USINT	1	Socket 接收数据存储模式，0 为接续存储数据 ,1 为覆盖存储数据
VAR	ReceiveBuffer	%MB4000	ARRAY[1..100] OF USINT	[100(0)]	Socket 数据接收缓存地址
VAR	ReceiveLength		UINT	100	Socket 接收数据长度
VAR	Socket_Close0		Socket_Close		Socket 端口关闭指令
VAR	Close_exe		BOOL		Socket 端口关闭执行条件
VAR	Close_Done		BOOL		Socket 端口关闭完成位
VAR	Socket_GetStatus0		Socket_GetStatus		Socket 工作状态获取指令
VAR	Connected		BOOL		连接建立成功
VAR	Received		BOOL		接收成功
VAR	Closed		BOOL		连接已关闭
VAR	Sent		BOOL		数据发送完成
VAR	Opening		BOOL		Socket 连接端口打开中
VAR	Receiving		BOOL		数据接收中
VAR	Closing		BOOL		Socket 连接端口关闭
VAR	Sending		BOOL		数据发送中

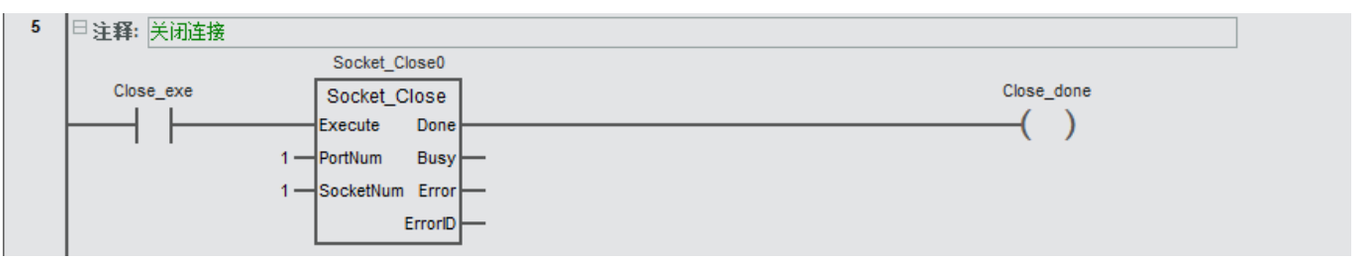
- **控制器2梯形图程序如下：**

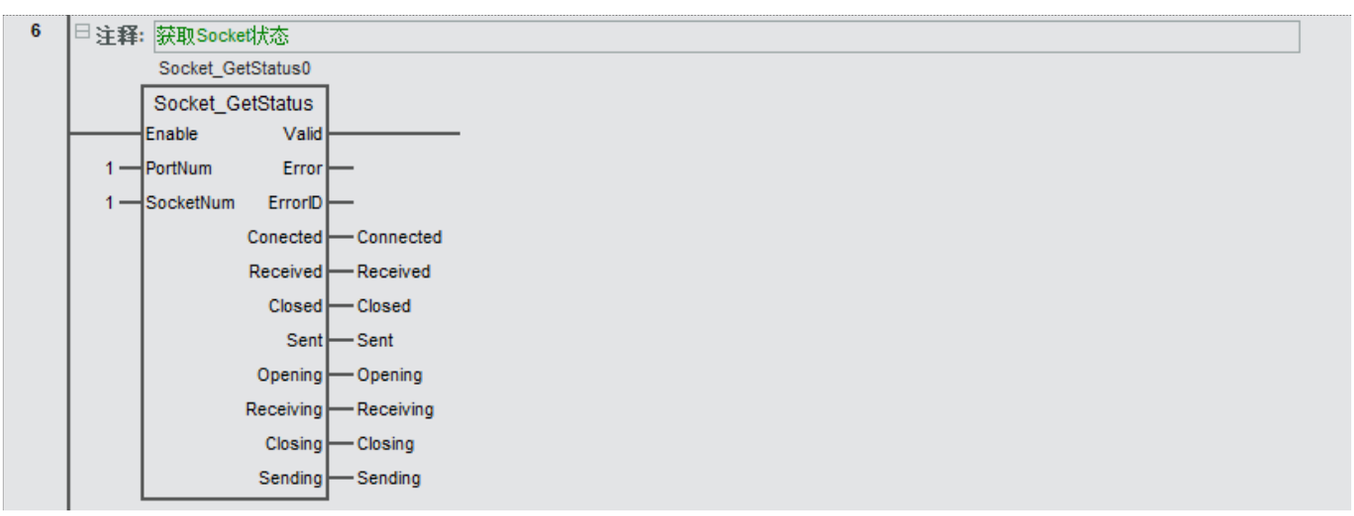



2
注释: 打开连接
Socket_Open0
Socket_Open
Open_exe
Execute
Done
Open_Done
1
PortNum
Busy
1
SocketNum
Error
0
Mode
ErrorID


3
注释: 发送数据
Socket_Send0
Socket_Send
Send_exe
Execute
Done
1
PortNum
Busy
Send_Abort
Abort
Active
1
SocketNum
Aborted
1
CyclicRun
Error
10
CycleTime
ErrorID
Send_data[1]
SendAddr
Sent
Sendlength
SendLength
Sending


4
注释: 接收数据
Socket_Receive0
Socket_Receive
Receive__exe
Execute
Done
1
PortNum
Busy
Receive_Abort
Abort
Active
1
SocketNum
Aborted
ReceiveMode
Mode
Error
SaveMode
DataSaveMode
ErrorID
Receive_date[1]
DataSaveAddr
Received
ReceiveLength
ReceiveLength
Receiving
ReceivedLength


5
注释: 关闭连接
Socket_Close0
Socket_Close
Close_exe
Execute
Done
Close_done
1
PortNum
Busy
1
SocketNum
Error
ErrorID


6
注释: 获取Socket状态
Socket_GetStatus0
Socket_GetStatus
Enable
Valid
1
PortNum
Error
1
SocketNum
ErrorID
Conected
Connected
Received
Received
Closed
Closed
Sent
Sent
Opening
Opening
Receiving
Receiving
Closing
Closing
Sending
Sending

- **控制器2ST程序如下：**

```
// 配置参数
Socket_Config0(Execute:=Config_exe ,
    PortNum:= 1,
    SocketNum:=1 ,
    Protocol_Type:=0 ,
    RemoteIPSeg1:=192 ,
    RemoteIPSeg2:=168 ,
    RemoteIPSeg3:=1 ,
    RemoteIPSeg4:=1 ,
    Remote_Port:=10000 ,
    Local_Port:=20000 ,
    LinkKeeptime:=3000 ,
    Done=>Config_done
    );
// 打开连接
Socket_Open0(Execute:= Open_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Mode:=0 ,
    Done=>Open_Done
    );
// 发送数据
Socket_Send0(Execute:=Send_Exe ,
    PortNum:= 1,
    Abort:=Send_Abort ,
    SocketNum:=1 ,
    CyclicRun:=1 ,
    CycleTime:=10 ,
    SendAddr:= Send_data[1] ,
    SendLength:=SendLength ,
    );
// 接收数据
Socket_Receive0(Execute:=Receive_Exe ,
    PortNum:=1 ,
    Abort:= Receive_Abort,
    SocketNum:=1 ,
    Mode:=ReceiveMode ,
    DataSaveMode:=SaveMode ,
    DataSaveAddr:= Receive_date[1] ,
    ReceiveLength:=ReceiveLength ,
```

```
    );
// 关闭连接
Socket_Close0(Execute:=Close_exe ,
    PortNum:=1 ,
    SocketNum:=1 ,
    Done=>Close_done ,
    );
/// 获取 Socket 状态

Socket_GetStatus0(Enable:=1,
    PortNum:=1 ,
    SocketNum:= 1,
    Conected=>Connected,
    Received=>Received,
    Closed=>Closed,
    Sent=>Sent,
    Opening=>Opening,
    Receiving=>Receiving,
    Closing=>Closing,
    Sending=>Sending
    );
```

- **程序流程说明：**

步骤 1：配置 Socket 参数（Socket_Config），包括连接方式、远端设备 IP、端口号、连接保持时间。将变量 Config_exe 设置为 TRUE 以触发 Socket 参数配置指令执行，变量 Config_done 为 TRUE 即配置成功。

步骤 2：打开 Socket 端口建立连接（Socket_Open），UDP 模式下不需要设置连工作模式，Open_exe 变量设置为 TRUE，以触发 Socket_Open 指令来打开 Socket 端口并建立连接。可以使用指令 Socket_GetStatus 获取 Socket 的状态，通过检查 Opening 变量是否为 TRUE 来判断 Socket 端口是否打开，当 UDP 两端同时处于 Opening 状态则建立连接。

步骤 3：发送和接收 Socket 数据。在 Socket 端口打开且连接建立成功后，方可执行发送和接收指令，本例中，将变量 Send_Exe 设置为 TRUE 即可触发 Socket 发送，(Socket_Send1.CyclicRun) 设置为 1 为循环发送，将变量 Receive1_Exe 设置为 TRUE 即可触发 Socket 接收。

步骤 4：关闭 Socket 端口以断开连接（Socket_Close），变量 Close_exe 为 TRUE 触发关闭 Socket 端口指令，变量 Close_done 为 TRUE 即关闭成功。

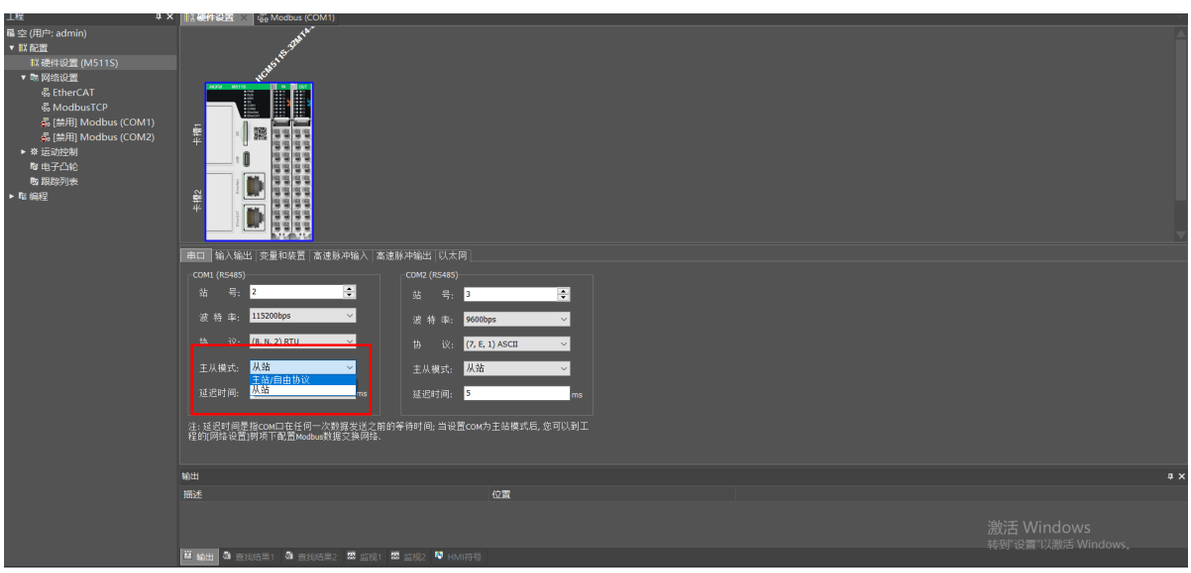
第 2 章　串口通讯

2.1 串口通讯使用概览

软件配置使用 modbus 功能

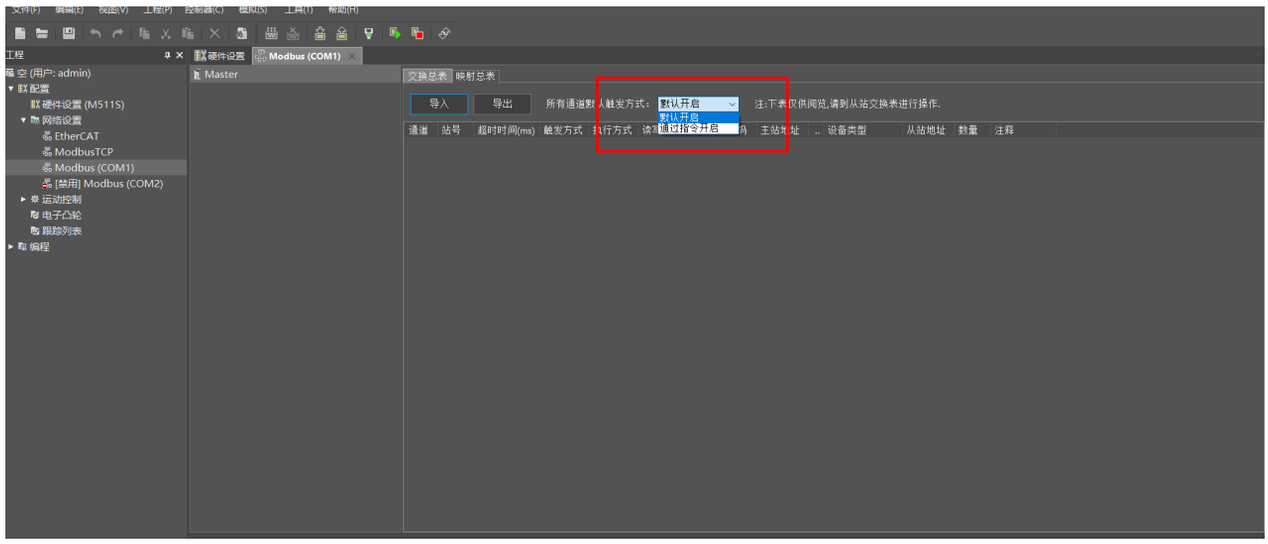
步骤	项目	使用情况		说明
1	设置串口主从模式	通过指令开启	Serial_Manage	设置串口主从模式
		硬件配置界面将串口主从模式改为 主站 / 自由协议	默认从站	
2	设置通讯协议	软件配置界面	默认（7,E,1）ASCII	设置通讯协议
3	控制主站	软件配置界面选择通过指令开启	Modbus_MasterRun	开启 Modbus 主站功能
		软件配置界面选择默认开启	默认开启	开启 Modbus 主站功能
		Modbus_MasterStop		关闭 Modbus 主站功能
		Modbus_Masterstatus		获取 Modbus 主站状态
4	配置通道	软件配置界面		配置指定数据交换通道
5	控制通道	软件配置界面选择程控触发	Modbus_LinkRun	开启指定数据交换通道
		软件配置界面选择默认触发	默认开启	开启指定数据交换通道
		Modbus_LinkStop		关闭指定数据交换通道
		Modbus_GetLinkStatus		获取指定数据交换通道状态

步骤 1：对控制器的串口主从模式进行设置，硬件设置 - 串口 中将主从模式设置为 主站 / 自由协议，或通过 Serial_Manage 指令更改串口主从模式。

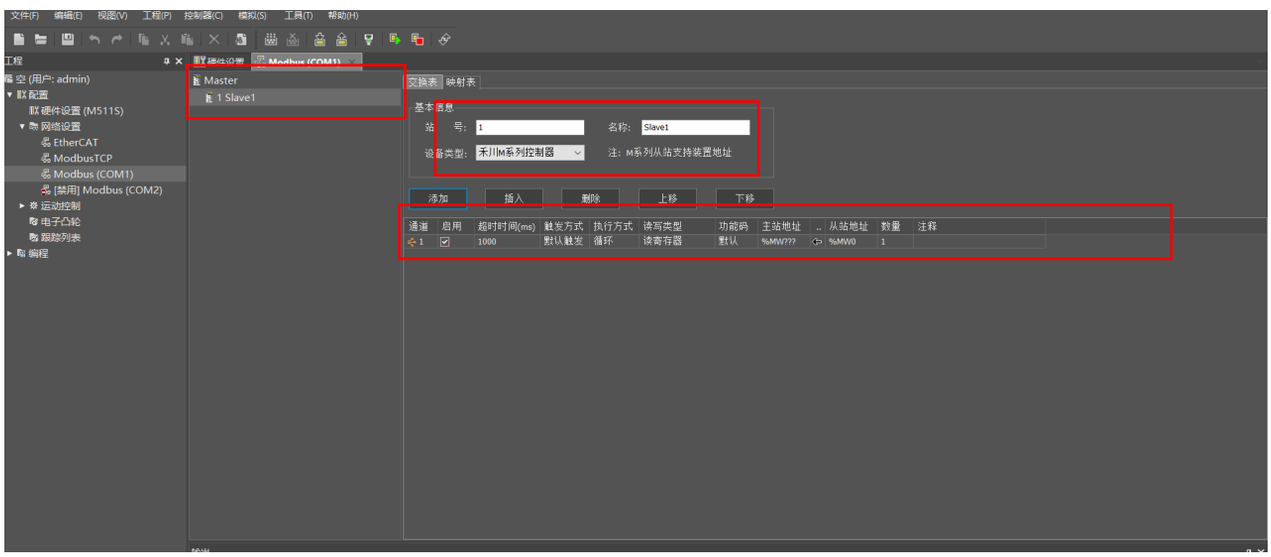


步骤 2：更改通讯协议，确保主从站的波特率和协议一致。

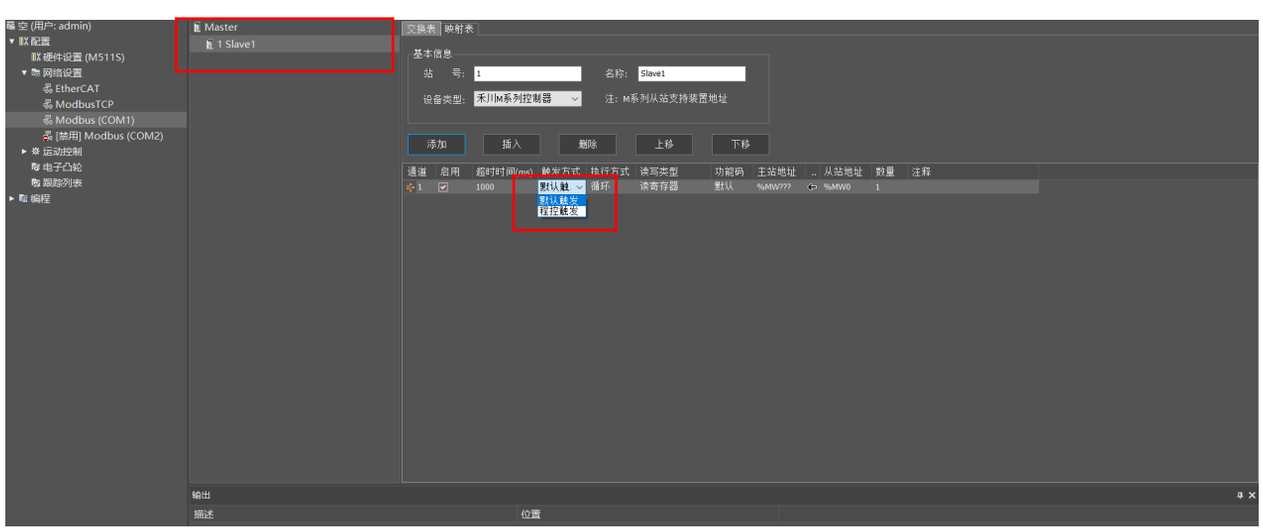
步骤 3：对 Modbus 主站功能总开关的操作，开启后可以使用 Modbus 相关功能；软件配置可选择使用默认开启（默认运行）或通过 Modbus_MasterRun 指令开启（指令触发后执行），通过 Modbus_MasterStop 指令关闭；通过 Modbus_GetMasterstatus 指令获取 Modbus 主站状态。



步骤 4：Modbus 主站内有多个数据交换通道，之间相互独立，在软件配置界面可以分别配置通道参数。



步骤 5：Modbus 主站内有多个数据交换通道，之间相互独立，通过软件配置的通道可使用默认触发（默认运行）或程控触发（Modbus_LinkRun 指令触发后执行）对指定通道进行开启。通过 Modbus_LinkStop 指令关闭，通过 Modbus_GetLinkStatus 指令获取指定数据交换通道状态。



指令配置使用 Modbus 功能：

步骤	流程	使用指令	说明
1	设置主从模式	Serial_Manage	设置串口的主从模式
2	控制主站	Modbus_MasterRun	开启 Modbus 主站功能
		Modbus_MasterStop	关闭 Modbus 主站功能
		Modbus_GetMasterstatus	获取 Modbus 主站状态
3	配置通道	Modbus_LinkConfig	配置指定数据交换通道
4	控制通道	Modbus_LinkRun	开启指定数据交换通道
		Modbus_LinkStop	关闭指定数据交换通道
		Modbus_GetLinkStatus	获取指定数据交换通道状态

步骤 1：对控制器的串口主从模式进行设置，通过 Serial_Manage 指令更改串口主从模式为“主站 / 自由协议”。

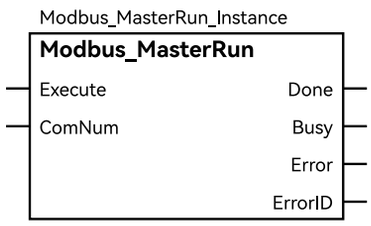
步骤 2：对 Modbus 主站功能总开关的操作，开启后可以使用 Modbus 相关功能，通过 Modbus_MasterRun 指令开启，通过 Modbus_MasterStop 指令关闭，通过 Modbus_GetMasterstatus 指令获取 Modbus 主站状态。

步骤 3：Modbus 主站内有多个数据交换通道，之间相互独立，通过 Modbus_LinkConfig 指令配置通道参数。

步骤 4：Modbus 主站内有多个数据交换通道，之间相互独立，通过 Modbus_LinkRun 指令开启。通过 Modbus_LinkStop 指令关闭，通过 Modbus_GetLinkStatus 指令获取指定数据交换通道状态。

2.2 Modbus_MasterRun(串口Modbus主站开启指令)

控制主站开启指令。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_MasterRun	串口 Modbus 管理	FB		Modbus_MasterRun(Execute:= 参数 , ComNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数为 TRUE 时执行该指令。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数用于指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

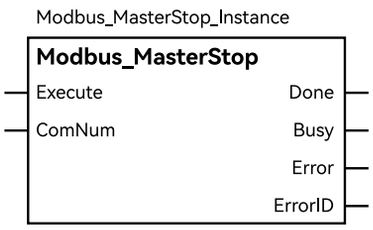
该指令用于开启串口 Modbus 周期性数据交换。在指令执行时（Execute 为 TRUE），通过参数 Execute 控制开启，如果 Execute 为 FALSE，指令不执行。

Modbus_LinkConfig 指令设定的参数，在 Open 由 FALSE 变为 TRUE 时生效，所以 Modbus 周期性数据交换的操作流程为 :

Step1(指令配置方式): Modbus_LinkConfige 指令配置参数 ; Step2: Modbus_Mange 指令执行（Enable 为 TRUE）, 通过 Modbus_MasterRun 和 Modbus_LinkRun 参数控制数据交换的开启通过 Modbus_MasterStop 和 Modbus_LinkStop 参数控制数据交换的关闭。

2.3 Modbus_MasterStop(串口Modbus主站停止指令)

关闭 Modbus 主站功能。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_MasterStop	串口 Modbus 管理	FB		Modbus_MasterStop(Execute:= 参数 , ComNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数为 TRUE 时执行该指令。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

Modbus_MasterStop 指令在配置时使用，当 Modbus 主站进行周期性数据交换时 Execute 为 FASLE。当需要停止 Modbus 数据交换时 Execute 置为 TRUE，即停止主站与从站间的数据交换。

2.4 Serial_Manage（串口主站从站模式设定指令）

设置指定的串口做 Modbus 主站或从站。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Serial_Manage	串口自定义主从站模式	FB	Serial_Manage_Instance Serial_Manage Enable Valid ComNum Busy Mode Error ErrorID	Serial_Manage_Instance(Enable:= 参数 , ComNum:= 参数 , Mode:= 参数 , Valid=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。
Mode	串口主从模式设置	INT	0~1	0	设定串口为主站模式或从站模式。 0: 串口为从站模式 1: 串口为主站模式

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	指令正常执行时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Busy	Enable 从 FALSE 变为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。 Error 由 FALSE 变为 TRUE 时。
Valid	Enable 为 TRUE 时。	Error 由 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时

◆ 功能说明

• 基本功能说明

该指令用于设置指定的串口做 Modbus 主站或从站。在指令执行时 (Enable 为 TRUE)， 通过参 Mode 设置串口做主站或从站 (0: 串口设置为从站、1: 串口设置为主站)；如果 Enable 为 FALSE，指令不执行，此时参数 Mode 无效。

2.5 Modbus_GetMasterStatus（串口Modbus主站状态获取指令）

读取主站状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_Get-MasterStatus	Modbus 主站状态获取	FB	Modbus_GetMasterStatus_Instance Modbus_GetMasterStatus Enable ComNum Valid Busy Error ErrorID Running Stopped	Modbus_GetMasterStatus _Instance(Enable:= 参数 , ComNum:= 参数 , Valid=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数 , Running=> 参数 , Stopped=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出有效参数	BOOL	TRUE / FALSE	指令正常执行时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Running	指令正在运行	BOOL	TRUE / FALSE	主站通讯执行中为 TRUE
Stopped	指令停止	BOOL	TRUE / FALSE	主站通讯停止为 TRUE

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。
Busy	Enable 从 FALSE 变为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。 Error 由 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时
Running	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。
Stopped	Enable 由 TRUE 变为 FALSE 时。 Error 由 FALSE 变为 TRUE 时。	Enable 为 TRUE 时。

◆ 功能说明

• 基本功能说明

该指令用于获取串口 Modbus 周期性数据交换的状态。如果数据交换发生错误，输出变量 Error、ErrorID 将输出对应错误状态和错误代码。

注意：输出变量 Error 和 ErrorID 表示该指令是否执行错误，与数据交换是否报错无关。

2.6 Modbus_LinkConfig（串口Modbus数据交换通道参数配置指令）

配置串口作为 Modbus 主站进行数据交换所需参数。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_Link-Config	Modbus 数据交换通道参数配置	FB	Modbus_LinkConfig_Instance **Modbus_LinkConfig** Execute — Done ComNum — Error LinkNum — ErrorID SlaveNodeID ExeMode CycleTime Mode CombineMode WriteAddr WriteAddrOffset WriteSlaveAddr WriteLength WriteMode ReadAddr ReadAddrOffset ReadSlaveAddr ReadLength ReadMode TimeOut Options	Modbus_LinkConfig (Execute:= 参数 , ComNum:= 参数 , LinkNum:= 参数 , SlaveNodeID:= 参数 , ExeMode:= 参数 , CycleTime:= 参数 , Mode:= 参数 CombineMode:= 参 WriteAddr:= 参数 , WriteAddrOffset:= 参数 , WriteSlaveAddr:= 参数 WriteLength:= 参数 , WriteMode:= 参数 , ReadAddr:= 参数 , ReadAddrOffset:= 参数 , ReadSlaveAddr:= 参数 , ReadLength:= 参数 , ReadMode:= 参数 TimeOut:= 参数 , Options:= 参数 , Done=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	输出变量有效	BOOL	TRUE / FALSE	FALSE	TRUE: 指令输出变量有效。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。
LinkNum	数据交换通道编号	UINT	1~24	0	设定串口 Modbus 数据交换通道的编号。可同时支持最多 24 笔不同的数据交换通道，通过 LinkNum 进行区分标识。
SlaveNodeID	目标设备的 Modbus 站号	USINT	0~255	0	设定目标设备的 Modbus 站号
ExeMode	执行模式	USINT	0~1	0	设定执行的模式 0：循环发送 1：仅发送一次
CycleTime	循环时间	USINT	0~65535	10	ExeMode 设定为循环发送时，上一次数据发送和下一数据发送的间隔时间：单位 ms
Mode	读写地址的类型	USINT	0~1	0	设定读写从站地址的类型，通过该 参数选择读写 Word 型地址或 Bit 型地址。 0：Word（字） 1：Bit（位）

Comebine-Mode	读写整合模式选择	BOOL	TRUE 或 FALSE	FALSE	该参数用于设定是否启用读写整 合功能 0：不启用 (读和写数据分开进行) 1：启用（使用 0x17 功能码读和写数据整合为一笔报文） 注意：仅在 Mode 为 0 时有效
WriteAddr	写操作数据缓存的 起始地址	UINT	%MW0~%MW32767 %QW0~%QW63	不可缺省	设定欲写入目标的数据在控制器 中的存放地址。写操作时，控制器 将该地址中的内容写入到目标设备中（ 该地 址 由 WriteAddr 和 WriteAddrOffset 共同决定）。如果本参数对应的输入为变量，则 需在该变量声明时为其指定正确 的地址。
WriteAddrOff-set	写操作数据缓存起 始地址的偏移量	USINT	0~255	0	设定写操作缓存地址的偏移量。实际写入的内容将从以 WriteAdd 为基础按 WriteAddrOffset 进行偏 移后的地址中取出。 如果为 Word 读写，偏移单位为 1Word。例如，WriteAddr 指定地 址为 %MW0，WriteAddrOffset 为 1，表示起始地址为 %MW1。 如果为 Bit 读写，偏移单位为 1Bit。例如，WriteAddr 指定地址 为 %MW1，WriteAddrOffset 为 1，表示起始地址为 %MX2.1。
WriteSlaveAd-dr	写操作的目标地址	UINT	16#0~16#FFFF	0	设定写入操作的起始地址，该地址 为目标设备中的 Modbus 地址。
Writelength	写操作长度	UINT	字装置：0~100 位装置：0~256	0	设定写入数据的长度。如果为 Word 读写，单位为 Word；如果为 Bit 读写，单位为 Bit。
WriteMode	写操作功能码	USINT	16#0、16#06、16#10、16#05、16#0F	0	设定写操作时使用的功能码。 如果设定为 0，控制器将自主选择。
ReadAddr	读操作数据缓存的 起始地址	UINT	%MW0~%MW32767 %QW0~%QW63	不可缺省	设定从目标设备中读取的数据在 控制器中的存放起始地址，该地址 由 ReadAddr 和 ReadAddrOffset 共同决定。如果本参数对应的输入为变量，则需在该变量声明时为其指定正确 的地址。
ReadAddrOff-set	读操作数据缓存起 始地址的偏移量	USINT	0~255	0	设定读操作缓存地址的偏移量。读取的内容将存放在以 ReadAddr 为基础按 ReadAddrOffset 进行偏移后的地址中。如果为 Word 读写，偏移单位为 1Word。例如，ReadAddr 指定地 址为 %MW100，ReadAddrOffset 为 1，表示起始地址 为 %MW101。 如果为 Bit 读写，偏移单位为 1Bit。例如，ReadAddr 指定地址 为 %MW100，ReadAddrOffset 为 1，表示起始地址 为 %MX200.1。
ReadSlaveAddr	读操作的目标地址	UINT	16#0~16#FFFF	0	设定读操作的起始地址，该地址 为目标设备中的 Modbus 地址。
Readlength	读操作长度	UINT	字装置：0~100 位装置：0~256	0	设定读取数据的长度。如果为 Word 读写，单位为 Word；如果为 Bit 读写，单位为 Bit。
ReadMode	读操作功能码	USINT	16#0、16#03、16#04、16#01、16#02	0	设定读操作时使用的功能码。如果设定为 0，控制器将自主选择。
TimeOut	通讯超时时间	UINT	0~65535	0	设定等待目标设备响应的时间，单位：ms。
Options	保留	Option_ Modbus_Link	保留		保留

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。

ErrorID	错误代码	BOOL	0~65535	指令执行异常时，输出错误代码。 *1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于配置 Modbus 数据交换通道相关的参数，例如从站设备的站号、数据读写的目标地址、缓存地址、数据长度、功能码等。注意：该指令仅进行参数配置，任何时候如要配置参数生效，需使用 Modbus_MasterRun 和 Modbus_LinkRun 指令启动数据交换。

• ComNum

该参数用于指定使用的串口硬件接口编号，根据使用的硬件接口，通过该参数设定对应的编号。本体串口硬件接口编号从 1 和 2 开始，扩展卡扩展串口硬件接口编号从 11 和 12 开始。

• LinkNum

控制器中内建有多个 Modbus 数据交换通道，这些数据交换通道之间相互独立，可分别配置其参数，该指令的 LinkNum 参数则用于指定欲对哪一个通道进行参数配置。

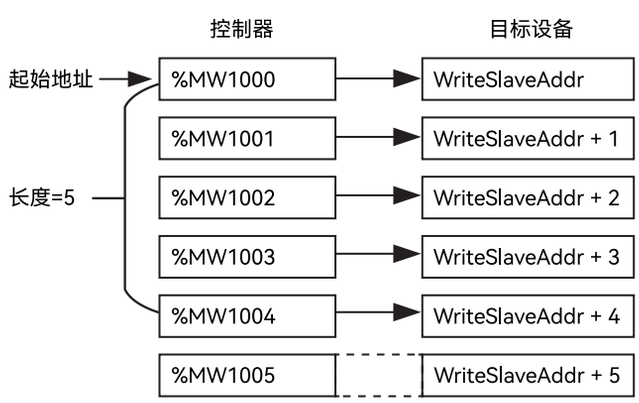
• WriteAddr、WriteAddrOffset、WriteLength

数据交换配置包含写操作和读操作。写操作是将写操作缓存地址中指定长度的数据写入到从站设备，写操作缓存地址由参数 WriteAddr、WriteAddrOffset 指定，写操作的数据长度由 WriteLength 指定。

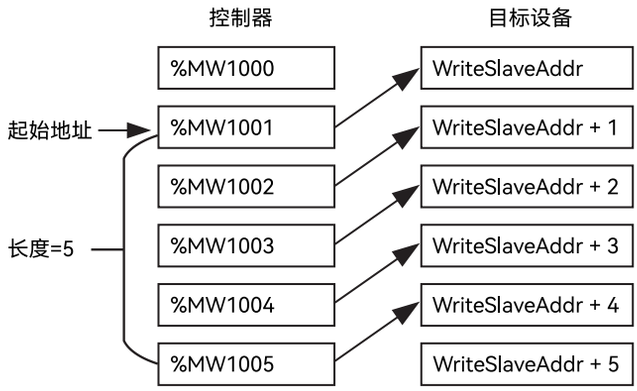
参数 WriteAddr、WriteAddrOffset 指定是缓存区域的起始地址，该地址以基准地址 + 偏移的形式进行指定，其中基准地址为 WriteAddr，偏移为 WriteAddrOffset。

Word 读写时（Mode=0），WriteAddrOffset 的单位为 Word，例如 WriteAddr 指定的基准地址为 %MW1000，WriteLength 指定的长度为 5:

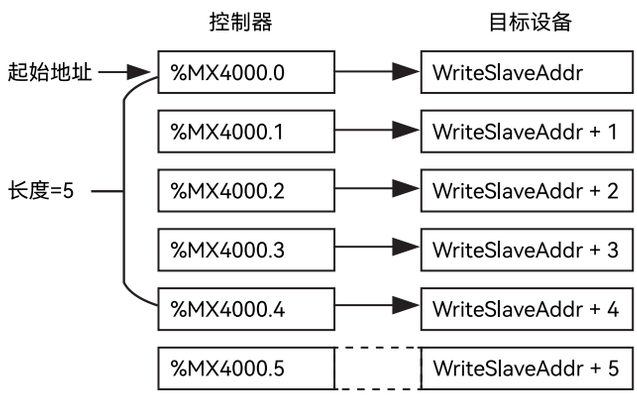
如果 WriteAddrOffset 为 0，则缓存起始地址为 %MW1000，如下图所示:



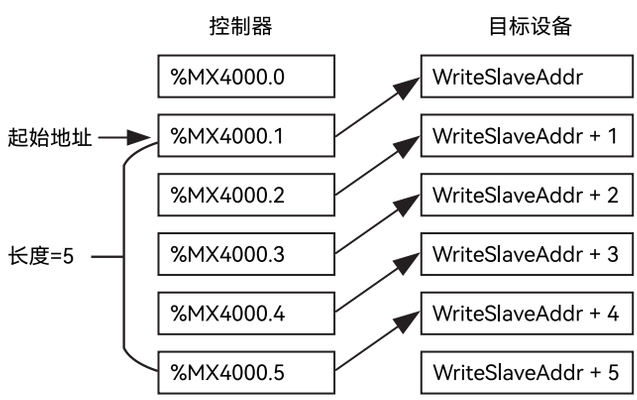
如果 WriteAddrOffset 为 1，则缓存起始地址为 %MW1001，如下图所示:



Bit 读写时（Mode=1），WriteAddrOffset 的单位为 Bit，例如 WriteAddr 指定的基准地址为 %MW2000（%MX4000.0），WriteLength 指定的长度为 5：如果 WriteAddrOffset 为 0，则缓存起始地址为 %MX4000.0，如下图所示：



如果 WriteAddrOffset 为 1，则缓存起始地址为 %MX4000.1，如下图所示：

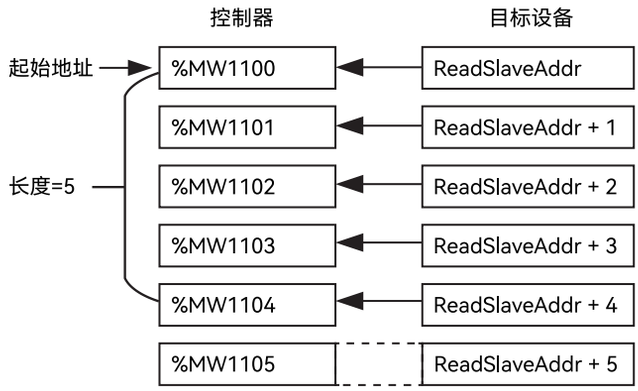


- **ReadAddr、ReadAddrOffset、ReadLength**

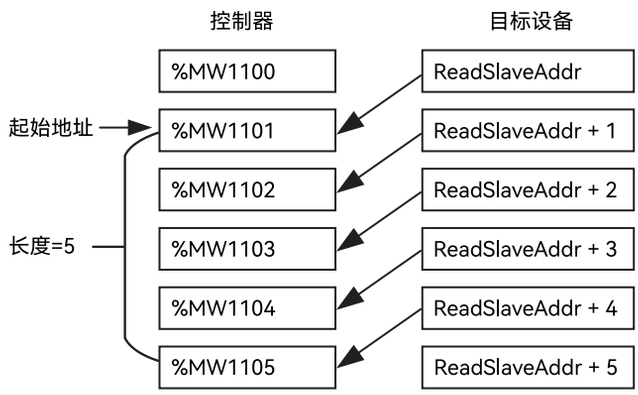
数据交换配置包含写操作和读操作。读操作是将从站设备中指定地址开始指定长度的数据读取到控制器，并将该数据存放于指定的缓存地址中。读操作缓存地址由参数 ReadAddr、ReadAddrOffset 指定，读操作的数据长度由 ReadLength 指定。

参数 ReadAddr、ReadAddrOffset 指定是缓存区域的起始地址，该地址以基准地址 + 偏移的形式进行指定，其中基准地址为 ReadAddr，偏移为 ReadAddrOffset。

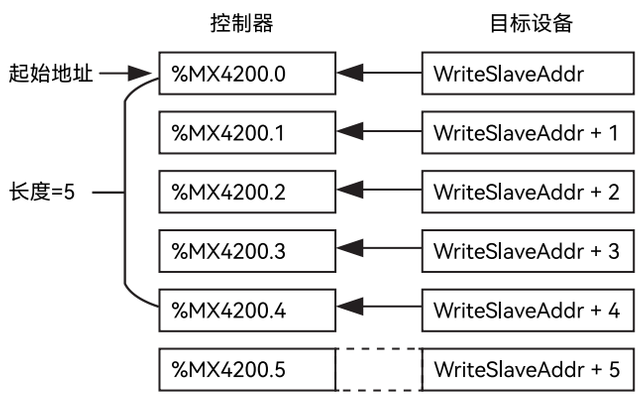
Word 读写时（Mode=0），ReadAddrOffset 的单位为 Word，例如 ReadAddr 指定的基准地址为 %MW1100，ReadLength 指定的长度为 5：如果 ReadAddrOffset 为 0，则缓存起始地址为 %MW1100，如下图所示：



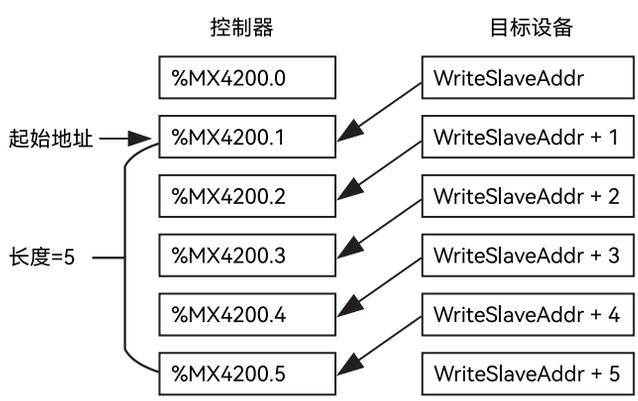
如果 ReadAddrOffset 为 1，则缓存起始地址为 %MW1101，如下图所示：



Bit 读写时（Mode=1），ReadAddrOffset 的单位为 Bit，例如 ReadAddr 指定的基准地址为 %MW2100（%MX4200.0），ReadLength 指定的长度为 5：如果 ReadAddrOffset 为 0，则缓存起始地址为 %MX4200.0，如下图所示：

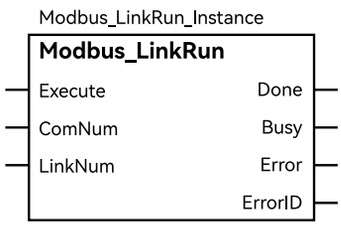


如果 ReadAddrOffset 为 1，则缓存起始地址为 %MX4200.1，如下图所示：



2.7 Modbus_LinkRun(串口Modbus数据交换通道开启指令)

启动串口 Modbus 指定的数据交换通道。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_Link-Run	串口从站 link 管理	FB		Modbus_LinkRun1(Execute:= 参数 , ComNum:= 参数 , LinkNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数为 TRUE 时执行该指令。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。
LinkNum	数据交换通道编号	USINT	1-24	不可缺省	设定串口 Modbus 数据交换通道的编号。可同时支持最多 24 笔不同的数据交换通道，通过 LinkNum 进行区分标识。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	指令完成标志	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

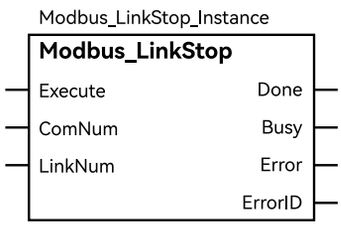
◆ 功能说明

• 基本功能说明

该指令用于开启 Modbus 指定的数据交换通道。

2.8 Modbus_LinkStop(串口Modbus数据交换通道停止指令)

关闭串口 Modbus 指定的数据交换通道。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_Link-Stop	串口从站 link 管理	FB		Modbus_LinkStop(Execute:= 参数 , ComNum:= 参数 , LinkNum:= 参数 , Done=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数为 TRUE 时执行该指令。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。
LinkNum	数据交换通道编号	USINT	1-24	不可缺省	设定串口 Modbus 数据交换通道的编号。可同时支持最多 24 笔不同的数据交换通道，通过 LinkNum 进行区分标识。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	指令完成标志	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

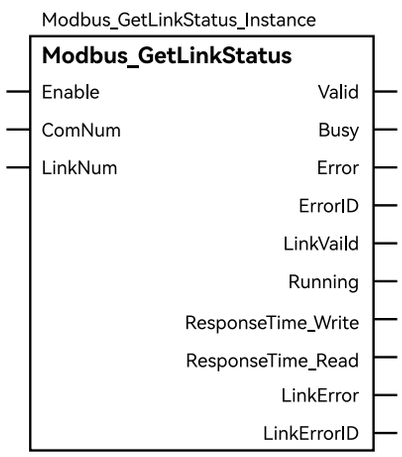
◆ 功能说明

• 基本功能说明

该指令关闭串口 Modbus 指定的数据交换通道，即停止指定通道主站与从站间的数据交换。

2.9 Modbus_GetLinkStatus（串口Modbus数据交换通道状态获取指令）

读取 Modbus 的状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Modbus_ GetLinkStatus	Modbus 从站状态读取	FB		Modbus_GetLinkStatus(Enable:= 参数 , ComNum:= 参数 , LinkNum:= 参数 , Valid=> 参数 , Busy=> 参数 , Error=> 参数 , ErrorID=> 参数 , LinkVaild=> 参数 , Running=> 参数 , ResponseTime_Write=> 参数 , ResponseTime_Read=> 参数 , LinkError=> 参数 , LinkErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。
LinkNum	数据交换通道编号	UINT	1~24	不可缺省	设定串口 Modbus 数据交换通道的编号。可同时支持最多 24 笔不同的数据交换通道，通过 LinkNum 进行区分标识。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE/FASLE	指令正常执行时变为 TRUE。
Busy	执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常或参数错误时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时或参数错误，输出错误代码。*1
LinkVaild	通道有效	BOOL	TRUE / FALSE	LinkNum 指定的通道的配置完成。
Running	指令正在运行	BOOL	TRUE/FALSE	指定通道数据交换正常时为 TRUE，异常时为 FALSE。
ResponseTime_ Write	写回复时间	TIME	0~65535	从主站发出写命令至接收到从站回复主站命令的时间，单位：ms
ResponseTime_ Read	读回复时间	TIME	0~65535	从主站发出读命令至接收到从站回复主站命令的时间，单位：ms
LinkError	配置通道错误	BOOL	TRUE/FASLE	指定通道数据交换异常时为 TRUE，正常时为 FALSE。如从站回复数据异常或者超时时，该位变为 TRUE，从站回复正常时，自动变为 FALSE。
LinkErrorID	配置通道错误代码	WORD	0~65535	指定通道数据交换异常时，即 LinkError 为 TRUE 时，输出对应的错误代码。值的含义请参阅“指令错误代码描述”。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。
Busy	Enable 从 FALSE 变为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。 Error 由 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时。 Error 从 FASLE 变为 TRUE 时。
LinkVaild	Enable 为 TRUE 时 ,Error 未变为 TRUE，通道开启正常时为 TRUE。	LinkNum 指定的通道配置异常时为 FALSE。
Running	Enable 为 TRUE 时指令运行 Error 未变为 TRUE。	指定通道数据交换异常时为 FALSE。
LinkError	Enable 为 TRUE, 通道状态异常时为 TRUE。	指定通道数据交换正常时为 FALSE。

◆ 功能说明

• 基本功能说明

Modbus_GetLinkStatus 指令用于检测配置通道的状态是否正常，读写数据的回复时间是否正常。

Enable 指令开启参数，ComNum 为工程使用的串口号，Linknum 为配置的通道编号，ResponseTime_Write 为主站向从站写数据等从站回复的时间，ResponseTime_Read 为主站向从站读数据等从站回复的时间。在使用时主要用来检测软件配置通道是否开启正常，通常配合 Modbus_GetMsterStatus 指令来使用检测主站通讯状态。

2.10 串口通讯范例

2.10.1 串口数据交换范例一：软件配置Modbus数据交换

- **目标需求**

两个 M 系列 PLC 通过串口协议的 0x10 功能码和 0x03 功能码进行数据交换

控制器的设置如下：

项目	主站控制器	项目	从站控制器
通讯协议	115200,(8,N,2),RTU	通讯协议	115200,(8,N,2),RTU
主从模式	主站 / 自由协议	主从模式	从站
地址	MW200	地址	MW300
地址	MW500	地址	MW400

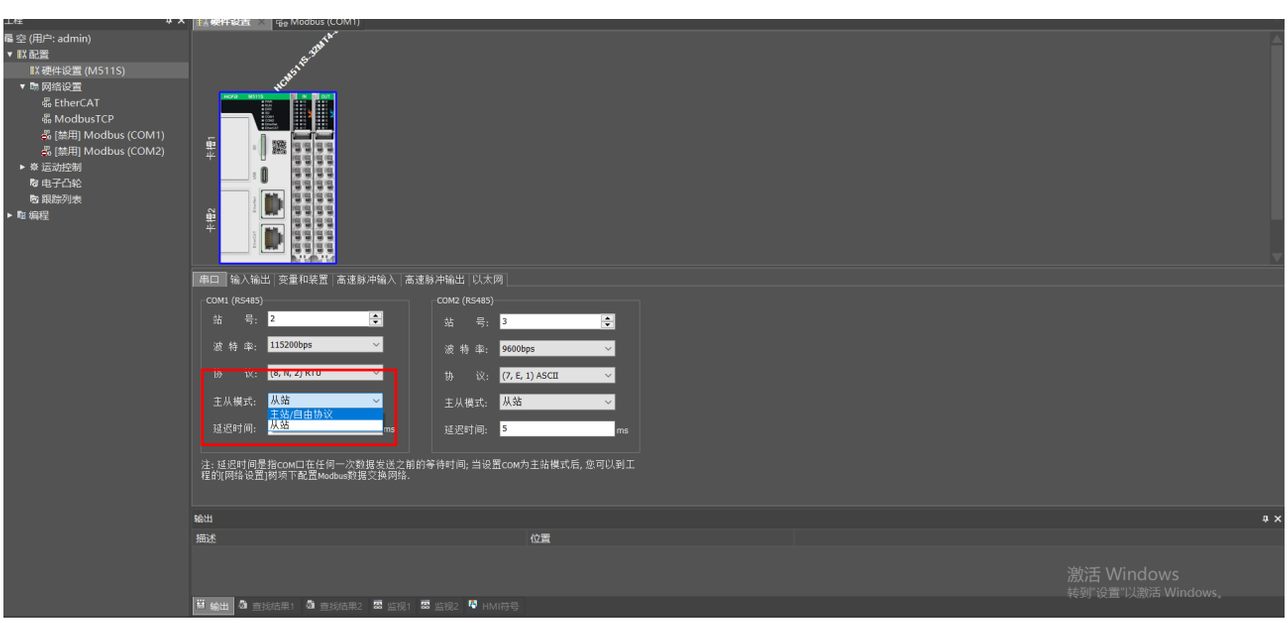
- **需求分析**

根据需求，需要主站控制器利用通道 1 的 0x10 功能码向从站控制器发送数据，0x03 功能码向从站控制器读取数据

步骤	项目	使用情况		说明
1	设置串口主从模式	硬件配置界面将串口主从模式改为 主站 / 自由协议	默认从站	设置串口主从模式
2	设置通讯协议	软件配置界面	默认（7,E,1）ASCII	设置通讯协议
3	控制主站	软件配置界面选择默认开启	默认开启	开启 Modbus 主站功能
		Modbus_MasterStop		关闭 Modbus 主站功能
		Modbus_GetMasterstatus		获取 Modbus 主站状态
4	配置通道	软件配置界面		配置指定数据交换通道
5	控制通道	软件配置界面选择默认触发	默认开启	开启指定数据交换通道
		Modbus_LinkStop		关闭指定数据交换通道
		Modbus_GetLinkStatus		获取指定数据交换通道状态

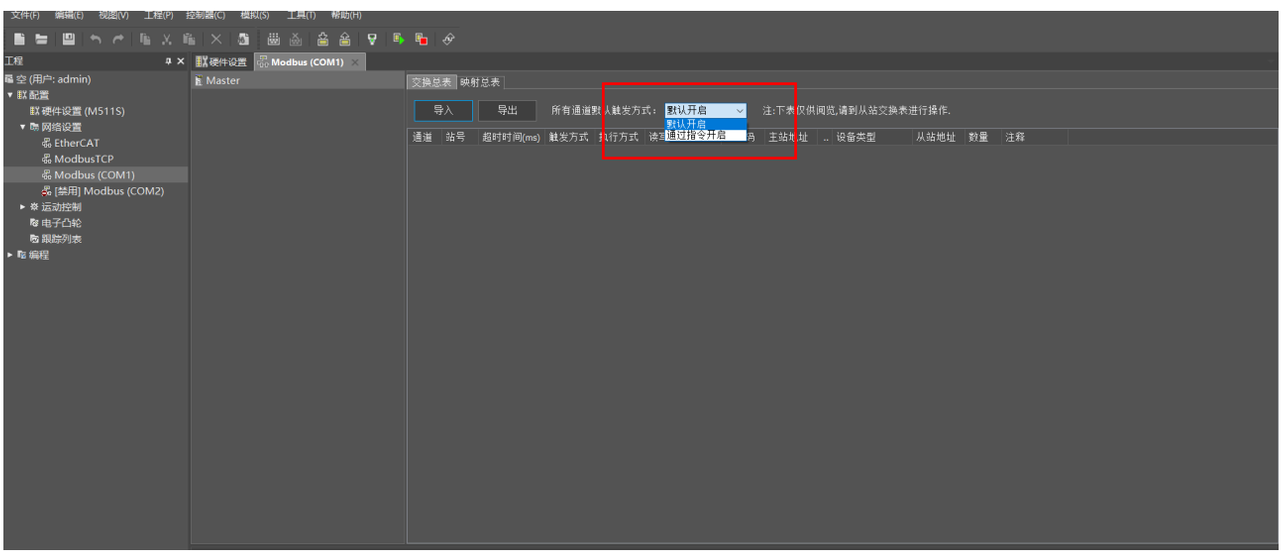
- **软件配置**

步骤一：对控制器的串口主从模式进行设置，硬件设置 - 串口 中将主从模式设置为 主站 / 自由协议

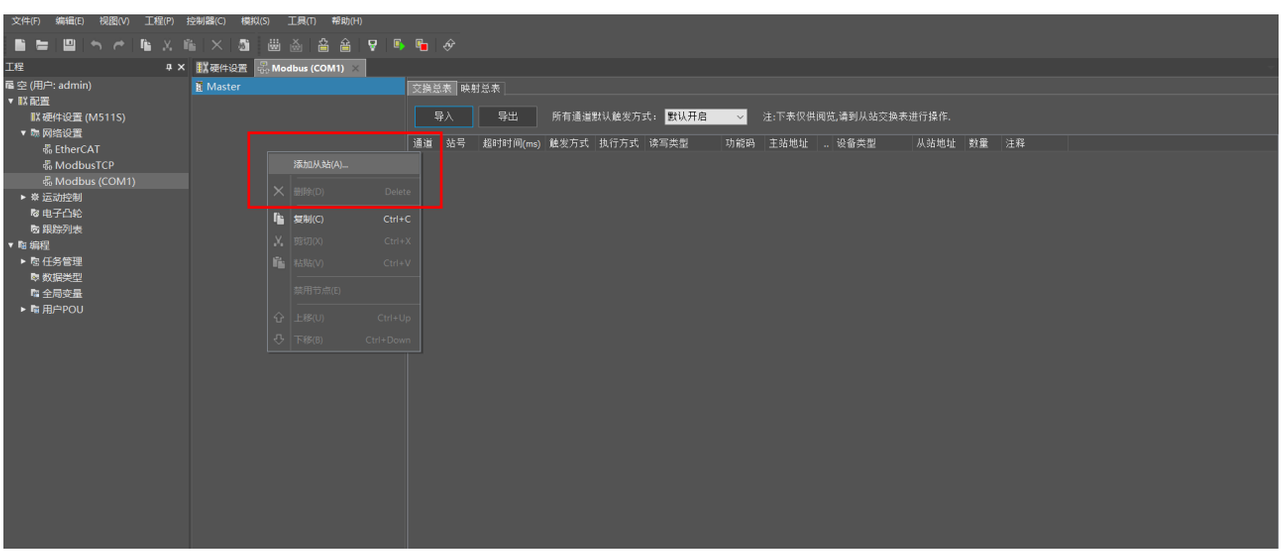


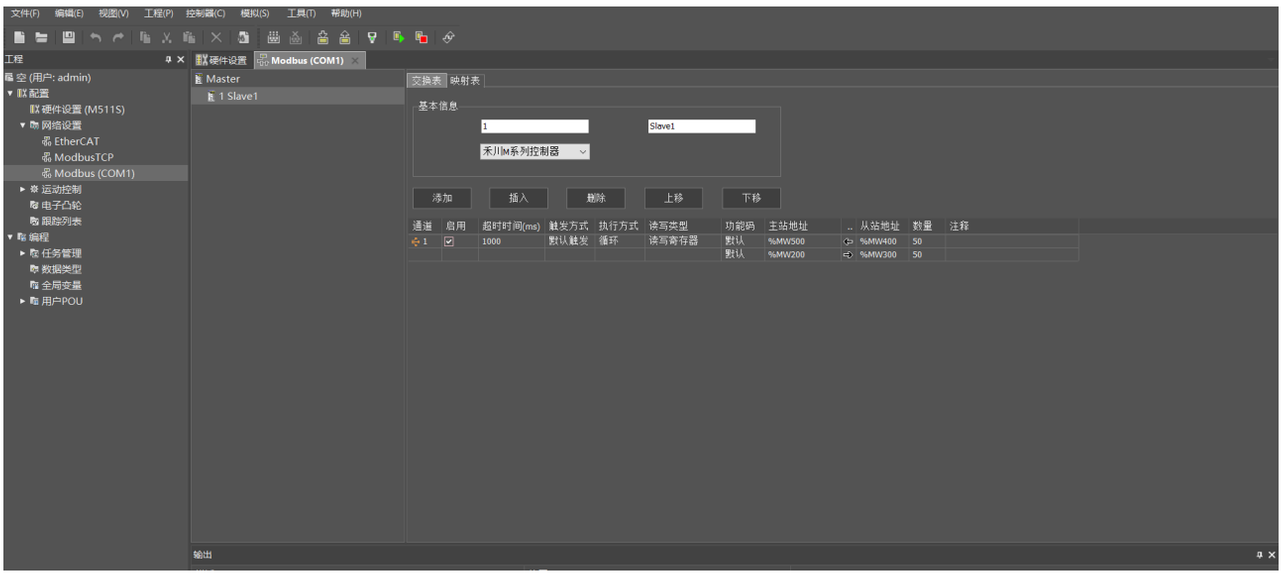
步骤二：更改通讯协议，需要确保主从站的波特率和协议一致

步骤三：将 Modbus 主站启动方式选择为默认开启，下载后自动开启 Modbus 主站功能

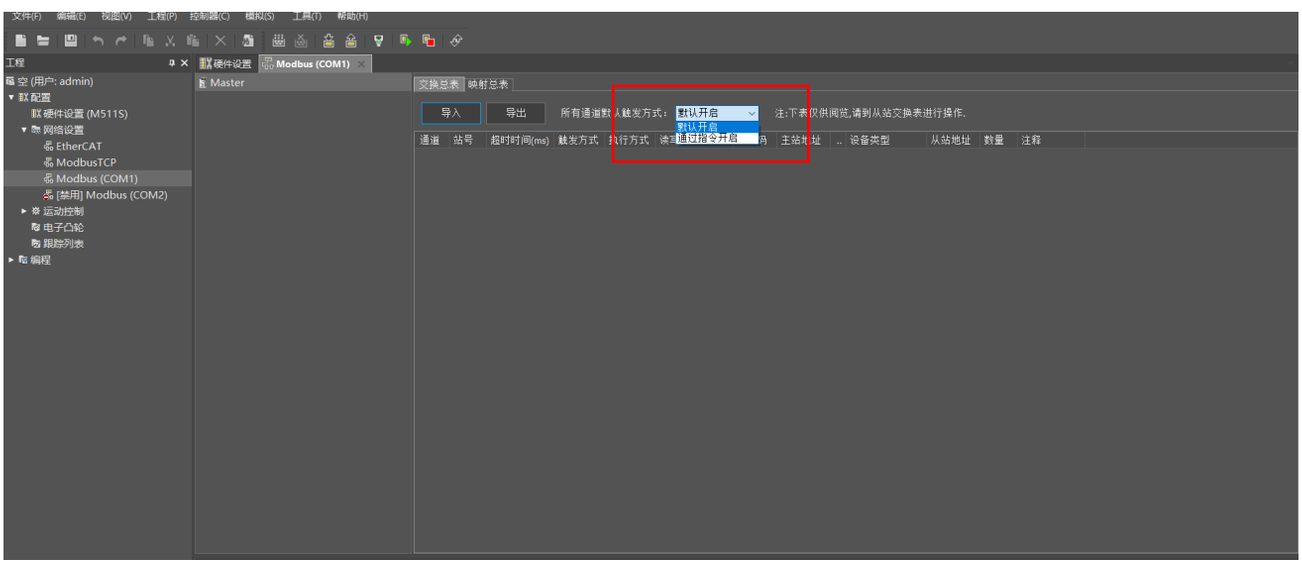


步骤四：添加从站，设置通道 1 的数据参数





步骤五：触发方式设置为默认触发，下载后自动开启通道 1 的数据开始交换



2.10.2 Modbus数据交换范例二：软件与指令配置Modbus数据交换

- **目标需求**

两个 M 系列 PLC 通过串口协议的 0x10 功能码和 0x03 功能码进行数据交换

控制器的 IP 地址和端口如下：

项目	主站控制器	项目	从站控制器
通讯协议	115200,(8,N,2),RTU	通讯协议	115200,(8,N,2),RTU
主从模式	主站 / 自由协议	主从模式	从站
地址	MW200	地址	MW300
地址	MW500	地址	MW400

- **需求分析**

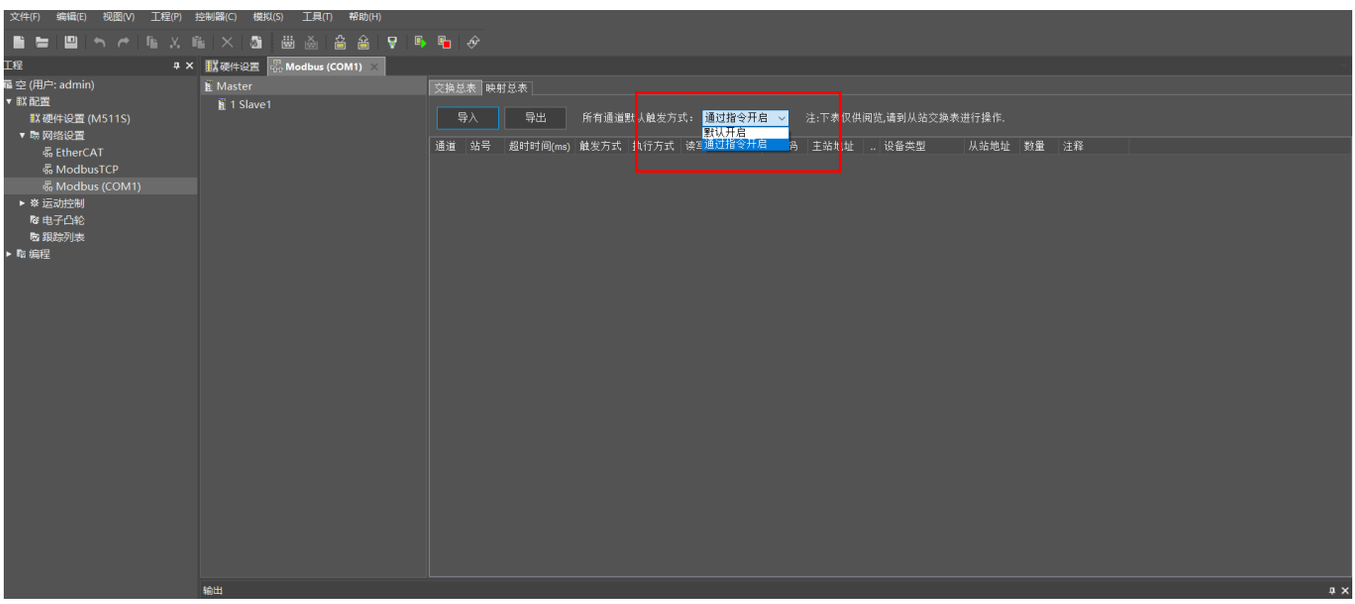
根据需求，需要主站控制器利用通道 1 的 0x10 功能码向从站控制器发送数据，0x03 功能码向从站控制器读取数据

步骤	项目	使用情况		说明
1	设置串口主从模式	通过指令开启	Serial_Manage	设置串口主从模式
		硬件配置界面将串口主从模式改为 主站 / 自由协议	默认从站	

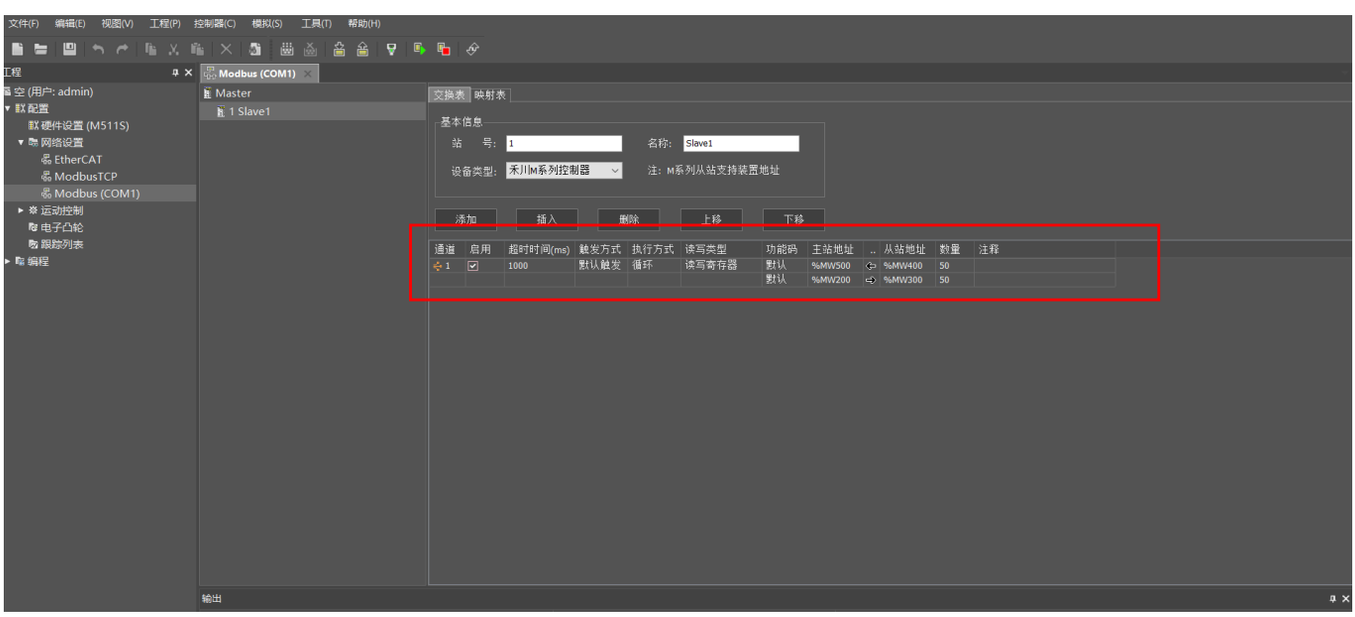
2	设置通讯协议	软件配置界面	默认（7,E,1）ASCII	设置通讯协议
3	控制主站	软件配置界面选择通过指令开启	Modbus_MasterRun	开启 Modbus 主站功能
		软件配置界面选择默认开启	默认开启	开启 Modbus 主站功能
		Modbus_MasterStop		关闭 Modbus 主站功能
		Modbus_Masterstatus		获取 Modbus 主站状态
4	配置通道	软件配置界面		配置指定数据交换通道
5	控制通道	软件配置界面选择程控触发	Modbus_LinkRun	开启指定数据交换通道
		软件配置界面选择默认触发	默认开启	开启指定数据交换通道
		Modbus_LinkStop		关闭指定数据交换通道
		Modbus_GetLinkStatus		获取指定数据交换通道状态

- **软件配置**

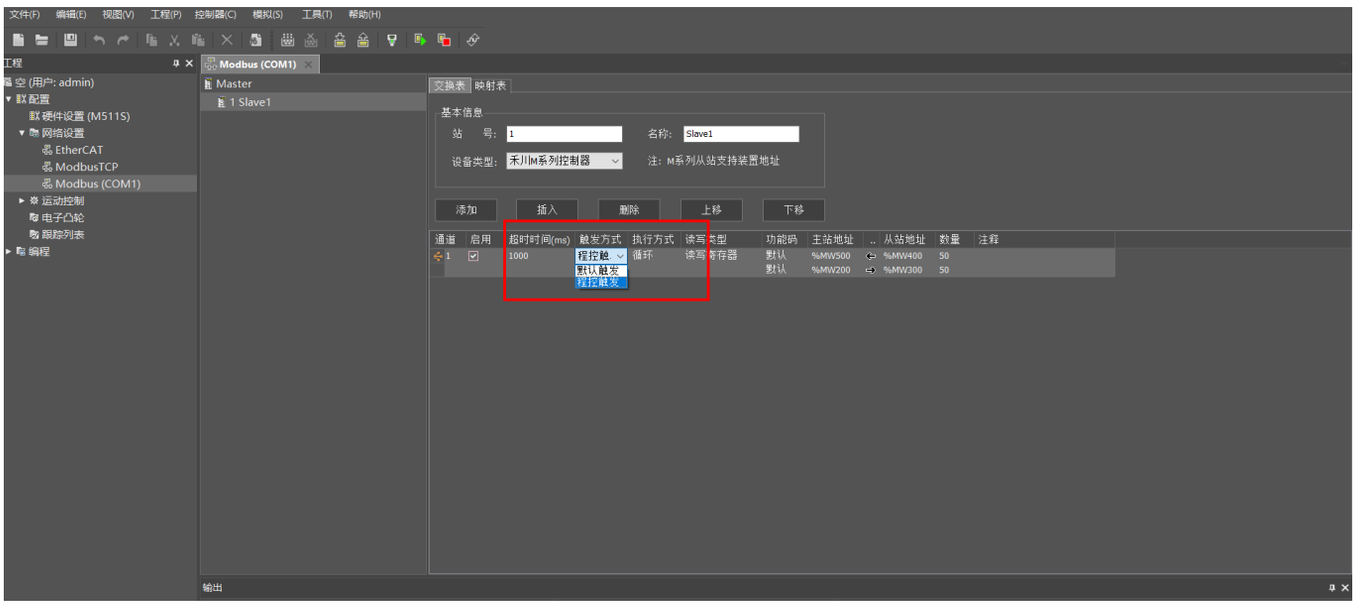
步骤一：将 ModbusTCP 主站启动方式选择为通过指令启动，下载后自动开启 ModbusTCP 主站功能



步骤二：添加从站，设置通道 1 的数据参数



步骤三：触发方式设置为程控触发，由程序控制



类别	名称	分配到	数据类型	初始值	注释
VAR	MasterRun1Ex		BOOL		
VAR	MasterStop1EX		BOOL		
VAR	Modbus_MasterRun1		Modbus_MasterRun		
VAR	Modbus_MasterStop1		Modbus_MasterStop		
VAR	Link1RunEX		BOOL		
VAR	LinkStop1EX		BOOL		
VAR	Modbus_LinkRun1		Modbus_LinkRun		
VAR	Modbus_LinkStop1		Modbus_LinkStop		

梯形图（LD）

ST

```
Modbus_MasterRun1(Execute:= MasterRun1Ex,
    ComNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
Modbus_MasterStop1(Execute:= MasterStop1EX,
    ComNum:=1 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
Modbus_LinkRun1(Execute:= Link1RunEX,
    ComNum:=1 ,
    LinkNum:=1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
Modbus_LinkStop1(Execute:=LinkStop1EX ,
    ComNum:= 1,
    LinkNum:= 1,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=>
    );
```

第一步：当变量 MasterRun1Ex 设置为 TRUE 触发 Modbus_MasterRun1 指令配置开启 Modbus 主站功能。

第二步：当变量 LinkRun1EX 设置为 TRUE 触发 Modbus_LinkRun1 指令配置开启 Modbus 从站功能。

第三步：当变量 LinkConfigEX 设置为 TRUE 触发 Modbus_LinkConfig1 指令将配置好的数据参数写入到通道 1。

第四步：如果要停止，将变量 LinkStop1EX 设置为 TRUE 触发 Modbus_LinkStop1 指令关闭通道 1 的数据交换，或将变量 MasterStop1EX 设置为 TRUE 触发 Modbus_MasterStop1 指令配置关闭 Modbus 功能。使用 Modbus_LinkStop 指令关闭指定通道数据交换，用户如果配置其他通道，其他通道还可以进行数据交换；使用 Modbus_MasterStop 关闭的是 Modbus 功能，所有数据通道交换都会被停止。

2.11 Mds_UserDefine（串口自定义协议数据发送和接收指令）

按自定义协议控制串口发送和接收数据。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
Mds_UserDefine	串口自定义协议数据收发	FB	Mds_UserDefine_Instance Mds_UserDefine Execute / Done ComNum / Busy Abort / Active CyclicRun / Aborted SendAddr / Error SendLength / ErrorID ReceiveAddr / NewCycle ReceiveLength / Received Add_STX_ETX / ReceiveTimeOut STX / ReceiveOverflow ETX1 / ReceivedLength ETX2 TimeOut	Mds_UserDefine_Instance(Execute:= 参数 , ComNum:= 参数 , Abort:= 参数 , CyclicRun:= 参数 , SendAddr:= 参数 , SendLength:= 参数 , ReceiveAddr:= 参数 , ReceiveLength:= 参数 , Add_STX_ETX:= 参数 , STX:= 参数 , ETX1:= 参数 , ETX2:= 参数 , TimeOut:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Aborted=> 参数 , Error=> 参数 , ErrorID=> 参数 , NewCycle=> 参数 , Received=> 参数 , ReceiveTimeOut=> 参数 , ReceiveOverflow=> 参数 , ReceivedLength=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
ComNum	串口硬件接口编号	UINT	1~255	不可缺省	该参数为指定使用的串口硬件接口编号。本体串口硬件接口编号从 1 和 2 开始。 扩展卡扩展串口硬件接口编号从 11 和 12 开始。
Abort	中断数据发送和接收	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时中断数据发送和接收。
CyclicRun	循环发送和接收	BOOL	TRUE 或 FALSE	FALSE	设定是否循环发送和接收缓存区中的数据。 TRUE: 循环发送和接收 FALSE: 单次发送和接收
SendAddr	发送数据缓存起始地址	USINT	%MB0~%MB65535	不可缺省	设定存放发送数据的起始地址。
SendLength	发送数据长度	UINT	0~200	0	设定发送数据长度。(单位：Byte)
ReceiveAddr	接收数据缓存起始地址	USINT	%MB0~%MB65535	不可缺省	设定存放接收数据的起始地址。

ReceiveLength	接收数据长度	UINT	0~200	0	设定接收数据长度。(单位：Byte)
Add_STX_ETX	添加头码和尾码	BOOL	TRUE 或 FALSE	FALSE	设定是否添加头码和尾码。 TRUE: 添加 FALSE: 不添加
STX	头码	USINT	16#0~16#7F	16#3A	设定欲添加的头码。
ETX1	尾码 1	USINT	16#0~16#7F	16#0D	设定欲添加的第一个尾码。
ETX2	尾码 2	USINT	16#0~16#7F	16#0A	设定欲添加的第二个尾码。
TimeOut	超时时间	UINT	0~32767	0	设定接收数据超时时间。（单位：ms）

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成 (仅单次发送)	BOOL	TRUE / FALSE	单次发送模式下（CyclicRun 设置为 TRUE 时），数据发送完成、数据接收完成、数据发送和接收完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令正在控制串口	BOOL	TRUE / FALSE	指令正常控制串口时为 TRUE。
Aborted	指令被中断	BOOL	TRUE / FALSE	指令被中断时为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
NewCycle	新收发周期开始	BOOL	TRUE / FALSE	循环发送模式下，新收发周期将开始。
Received	接收完成	BOOL	TRUE / FALSE	数据接收完成时为 TRUE。
ReceiveTime-Out	接收超时	BOOL	TRUE / FALSE	数据接收超时为 TRUE。
ReceiveOver-flow	接收数据超长	BOOL	TRUE / FALSE	接收数据超过设定长度时为 TRUE。
Receiv-edLength	实际接收长度	UINT	0~200	实际接收长度。(单位：Byte)

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Execute 从 TRUE 变为 FALSE 时。
Busy	Execute 从 FALSE 变为 TRUE 时。	Done 由 FALSE 变为 TRUE 时。 Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。
Active	指令开始控制串口时。	Done 由 FALSE 变为 TRUE 时。 Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。
Aborted	指令被中断时。	Execute 从 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Execute 从 TRUE 变为 FALSE 时
NewCycle	循环发送模式下，上一收发周期结束时。	Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。 循环发送模式下，新收发周期开始时。
Received	接收完成时。	Aborted 由 FALSE 变为 TRUE 时。 Error 由 FALSE 变为 TRUE 时。 循环发送模式下，新收发周期开始时。

ReceiveTime-Out	接收超时时。	Execute 从 TRUE 变为 FALSE 时。
ReceiveOver-flow	接收数据长度超过设定值时。	当实际接收长度不超过设定长度时。

◆ 功能说明

• 基本功能说明

该指令用于控制串口收发自定义数据。

指令触发执行时，将发送缓存区域中的 SendLength（发送数据长度）指定长度的数据发送给对应的设备，然后将接收到数据存储到接收缓存区域中（超过 ReceiveLength 的部分会被丢弃）。发送缓存区域的起始地址由参数 SendAddr 指定，接收缓存区域的起始地址由参数 ReceiveAddr 指定。可以通过 CyclicRun 指定单次发送和接收或者循环发送和接收。

• SendLength（发送数据长度）

STX 为 FALSE(不启用头码和尾码) 时，指令执行时，将发送缓存区域中的 SendLength（发送数据长度）指定长度的数据发送给对应的设备；STX 为 TRUE(启用头码和尾码) 时，指令执行时，将发送缓存区域中 SendLength（发送数据长度）指定长度的数据以及头码和尾码的数据发送给对应的设备，SendLength（发送数据长度）不计算头码和尾码的数据。

• ReceiveLength（接收数据长度）

ReceiveLength（接收数据长度）为接收一帧报文中所有数据的长度。STX 为 TRUE(启用头码和尾码) 时， ReceiveLength（接收数据长度）将头码和尾码的数据也计算在内。

• 单次发送和接收

当参数 CyclicRun 为 FALSE 时表示单次发送、单次接收或者单次发送和接收。

需要执行单次发送和接收时，先将 CyclicRun 设置为 FALSE，再执行指令（Execute 从 FALSE 变为 TRUE）。

单次发送：当 SendLength（发送数据长度）不为 0，ReceiveLength（接收数据长度）为 0，CyclicRun 为 FALSE 时，单次发送；指令执行后，SendLength（发送数据长度）指定数据长度的数据发送后，Done 变为 TRUE，Done 变为 TRUE 后，该指令不再发送数据。如果需要再次发送数据，需要重新执行该指令。

单次接收：当 SendLength（发送数据长度）为 0，ReceiveLength（接收数据长度）不为 0 时，CyclicRun 为 FALSE 时，在 TimeOut 指定的时间内单次接收；接收到数据后，Done 变为 TRUE，Done 变为 TRUE 后，该指令不再接收数据。如果需要再次接收数据，需要重新执行该指令。在 TimeOut 指定的时间内没有接收到数据，该指令报错。如果需要再次接收数据，需要重新执行该指令。

单次发送和接收：当 SendLength（发送数据长度）不为 0，ReceiveLength（接收数据长度）不为 0，CyclicRun 为 FALSE 时，单次发送和接收；发送完数据并接收到数据后，Done 变为 TRUE，Done 变为 TRUE 后，该指令不再发送和接收数据。如果需要再次发送和接收数据，需要重新执行该指令。

• 循环发送和接收

当参数 CyclicRun 为 TRUE 时表示循环发送、循环接收或者循环发送和接收。

需要执行循环发送和接收时，先将 CyclicRun 设置为 TRUE，再执行指令（Execute 从 FALSE 变为 TRUE）。

循环发送：当 SendLength（发送数据长度）不为 0，ReceiveLength（接收数据长度）为 0 时，循环发送。

循环接收：当 SendLength（发送数据长度）为 0，ReceiveLength（接收数据长度）不为 0 时，在 TimeOut 指定的时间内循环接收。在 TimeOut 指定的时间内没有接收到数据，该指令报错。如果需要再次接收数据，需要重新执行该指令。

循环发送和接收：当 SendLength（发送数据长度）不为 0，ReceiveLength（接收数据长度）不为 0 时，循环发送和接收，发送完数据并接收到数据后，执行下次数据发送和接收。在 TimeOut 指定的时间内没有接收到数据，该指令报错。如果需要再次接收数据，需要重新执行该指令。

当接收到数据时，ReceivedLength 为实际接收到的数据长度，Received 变为 TRUE 一个任务周期，然后自动变为 FALSE。

循环发送和接收数据过程中，将 Abort 位由 FALSE 变为 TRUE 时，中断循环环发送和接收数据，同时输出变量 Aborted 变为 TRUE。

- **头码和尾码**

如果参数 Add_STX_ETX 为 TRUE，发送数据时将自动添加设置的头码和尾码，接收数据时将比对实际接收数据中的头码和尾码与设定值是否相符，如果不符，则丢弃接收数据并认为没有收到数据。

注意：当发送数据和接收的头码尾码不相同时，不可启用添加头码尾码功能。

- **注意事项**

该指令发送数据时不会自动添加校验码，可自行按照校验规则计算并放置于发送区对应地址中。

2.12 自定义协议范例

- **目标需求**

M511S 主机 COM1 口发送数据（通过串口自定义协议发送和接收指令发送标准 Modbus 报文数据），M511S 主机 COM2 口接收数据。

COM1 通信口发送标准 Modbus 报文数据	16#[01,10,15,00,00,01,02,00,08, 校验码]
COM2 通信口接收 COM1 发送的数据	16#[01,10,15,00,00,01,02,00,08, 校验码]

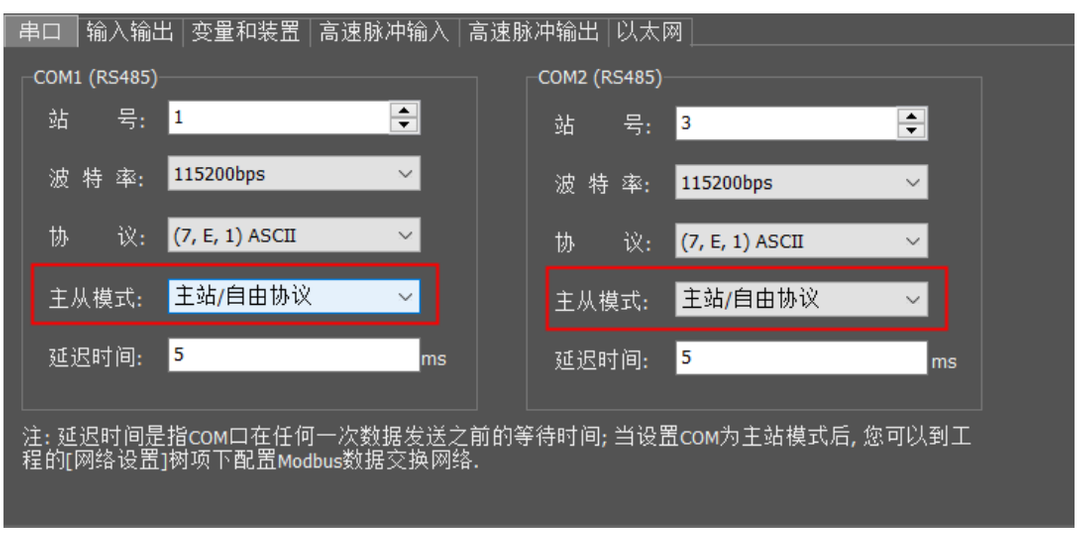
ASCII模式：

- **需求分析**

根据需求，ASCII 模式下，头码为 16#3A，尾码为 16#0D 和 16#0A。该范例中启用头码和尾码，请求数据的顺序为：3A 30 31 31 30 31 35 30 30 30 30 30 31 30 32 30 30 30 38 43 46 0D 0A，以上数据均为 16 进制数据。在发送数据前，需要指定发送数据缓存起始地址及发送数据长度，将头码和尾码外的数据依序写入到发送缓存区域中，发送数据长度为 20（不包含头码尾码）。在接收数据前，需要指定接收数据缓存起始地址及接收数据长度，接收数据长度为 23（包含头码尾码）。

- **软件设置**

串口使用自定义协议相关指令发送或接收数据时，需要将对应的串口设置为“主站 / 自由协议”，设置如下图红色方框处所示。



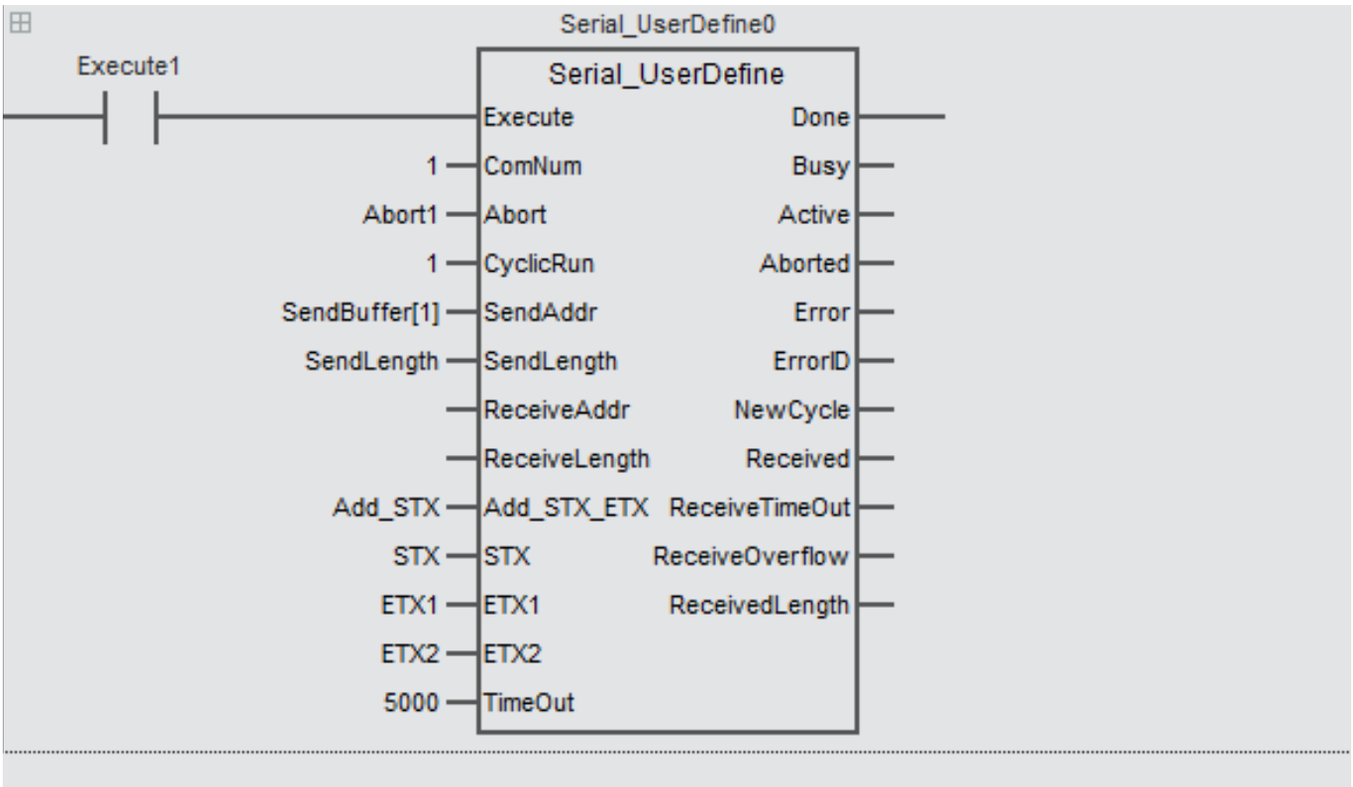
- **程序**

根据需求分析编写程序。

COM1 通讯口发送数据时，程序变量规划如下表所示：

变量名称	分配地址	数据类型	初始值	说明
Serial_UserDefine0		Serial_UserDefine		自定义协议收发指令
Execute1		BOOL		自定义协议收发指令执行条件
Abort1		BOOL		中断循环发送
SendBuffer	%MB0	ARRAY[1..20] OF USINT		发送数据缓存地址
SendLength		UINT	20	发送长度
Add_STX		BOOL	TRUE	添加头码和尾码
STX		USINT	16#3A	头码
ETX1		USINT	16#0D	尾码 1
ETX2		USINT	16#0A	尾码 2

主站梯形图程序如下：



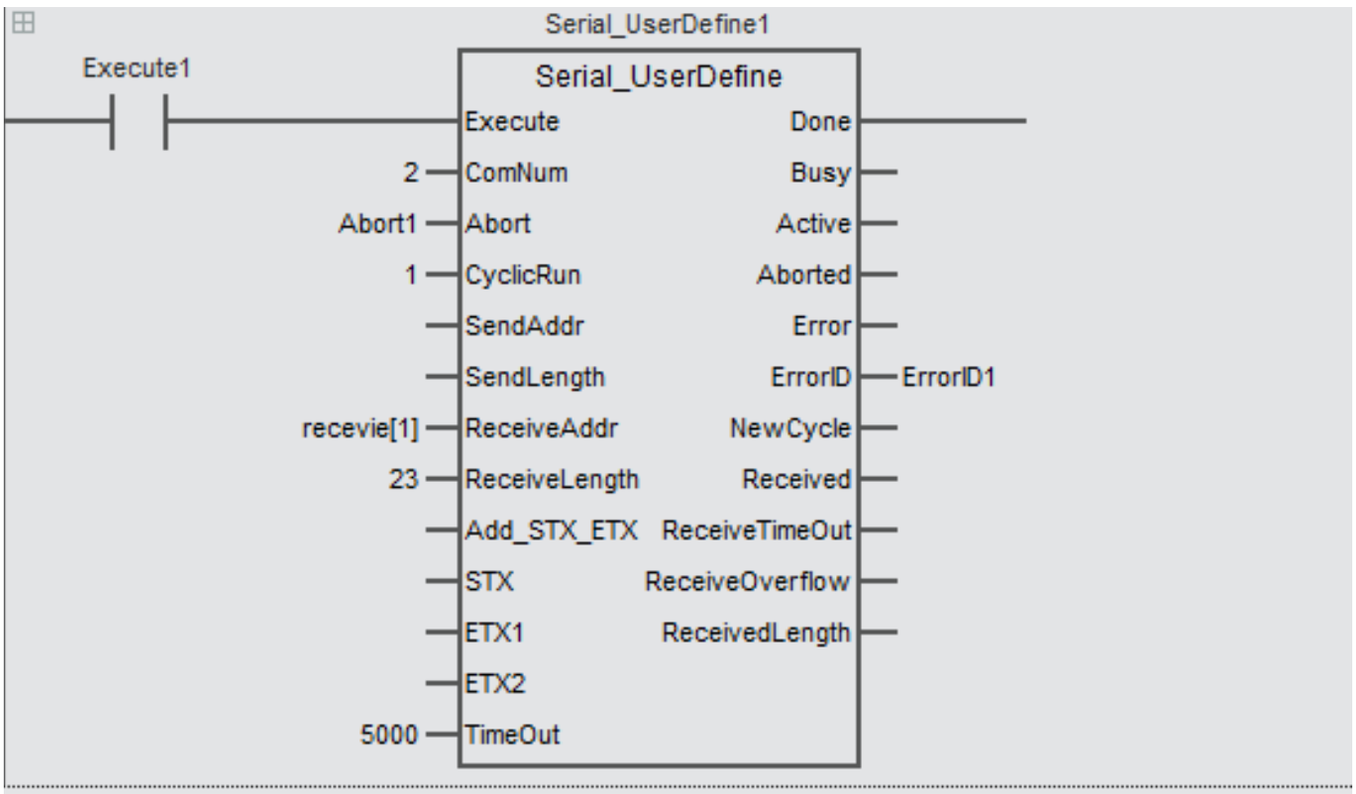
主站 ST 程序如下：

```
Serial_UserDefine0(Execute:= Execute1,
    ComNum:=1 ,
    Abort:= Abort1,
    CyclicRun:= 1,
    SendAddr:=SendBuffer[1] ,
    SendLength:= SendLength,
    ReceiveAddr:= ,
    ReceiveLength:= ,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:=5000 ,);
```

COM2 通讯口接收数据时，程序变量规划如下表所示：

变量名称	分配地址	数据类型	初始值	说明
Serial_UserDefine1		Serial_UserDefine		自定义协议收发指令
Execute1		BOOL		自定义协议收发指令执行条件
Abort1		BOOL		中断循环接收
recevie	%MB100	ARRAY[1..23] OF USINT		接收数据缓存地址
ErrorID1		WORD		错误代码

从站梯形图程序如下：



从站 ST 程序如下：

```
Serial_UserDefine1(Execute:= Execute1,
    ComNum:= 2,
    Abort:= Abort1,
    CyclicRun:=1 ,
    SendAddr:= ,
    SendLength:= ,
    ReceiveAddr:= recevie[1],
    ReceiveLength:= 23,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:= 5000,);
```

- **程序流程说明：**

第一步：将欲发送的数据写入到发送缓存区域中。在此范例中，需要将 16 进制数 30 31 31 30 31 35 30 30 30 30 30 31 30 32 30 30 30 38 43 46 依序写入到 %MB0 至 %MB20 中（SendBuffer[1]~SendBuffer[20]），接收数据存入到 %MB100 至 %MB122 中（recevie [1]~ recevie[23]）。根据发送和接收的数据，则发送数据长度为 20（不包含头码尾码），接收数据长度为 23（包含头码尾码）。输入参数 SendBuffer（发送数据缓存起始地址）指定的数组长度要等于或者大于 20，输入参数 recevie（接收数据缓存起始地址）指定的数组长度要等于或者大于 23。

第二步：输入变量 CyclicRun 设置为 TRUE（循环发送和接收），输入参数执行由 FALSE 变为 TRUE 后，UserDefine1 和 UserDefine2 指令触发执行，UserDefine1 指定 COM1 发送数据，UserDefine2 指定 COM2 接收数据。

RTU模式：

- **需求分析**

根据需求，RTU 模式下，请求数据的顺序为：01 10 15 00 00 01 02 00 08 E3 57，以上数据均为 16 进制数据。在发送数据前，需要指定发送数据缓存起始地址及发送数据长度，将发送数据依序写入到发送缓存区域中，发送数据长度为 11。在接收数据前，需要指定接收数据缓存起始地址及接收数据长度，接收数据长度为 11。

- **软件设置**

串口使用自定义协议相关指令发送或接收数据时，需要将对应的串口设置为“主站 / 自由协议”，设置如下图红色方框处所示。

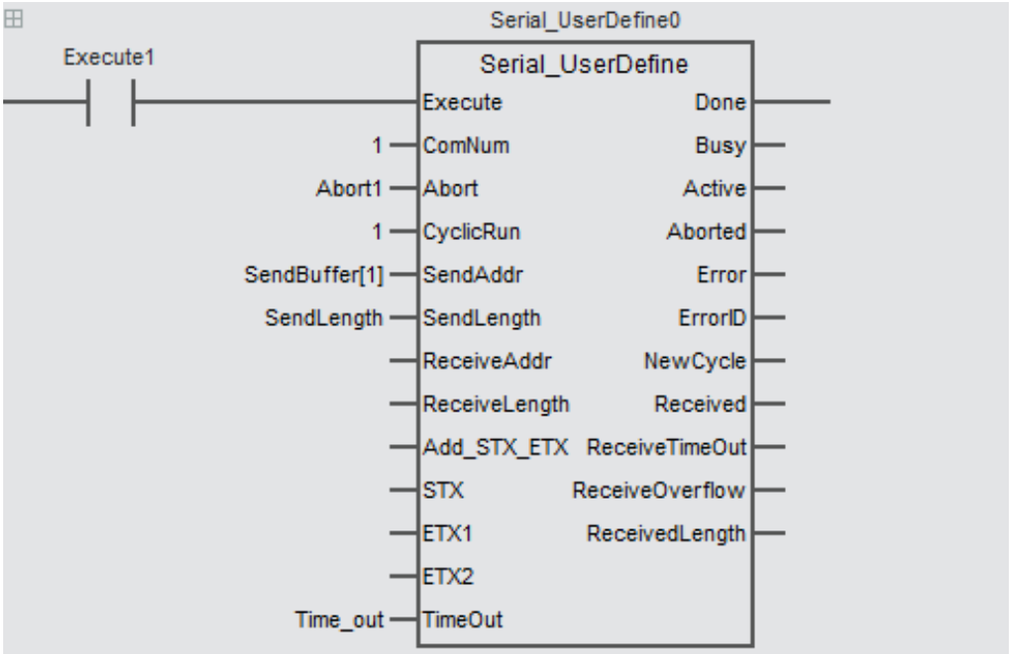
- **程序**

根据需求分析编写程序。

COM1 通讯口发送数据时，程序变量规划如下表所示：

变量名称	分配地址	数据类型	初始值	说明
Serial_UserDefine0		Serial_UserDefine		自定义协议收发指令
Execute1		BOOL		自定义协议收发指令执行条件
Abort1		BOOL		中断循环发送
sendbuffer	%MB0	ARRAY[1..11] OF USINT		发送数据缓存地址
Time_out		UINT	5000	设定接收数据超时时间

主站梯形图程序如下：



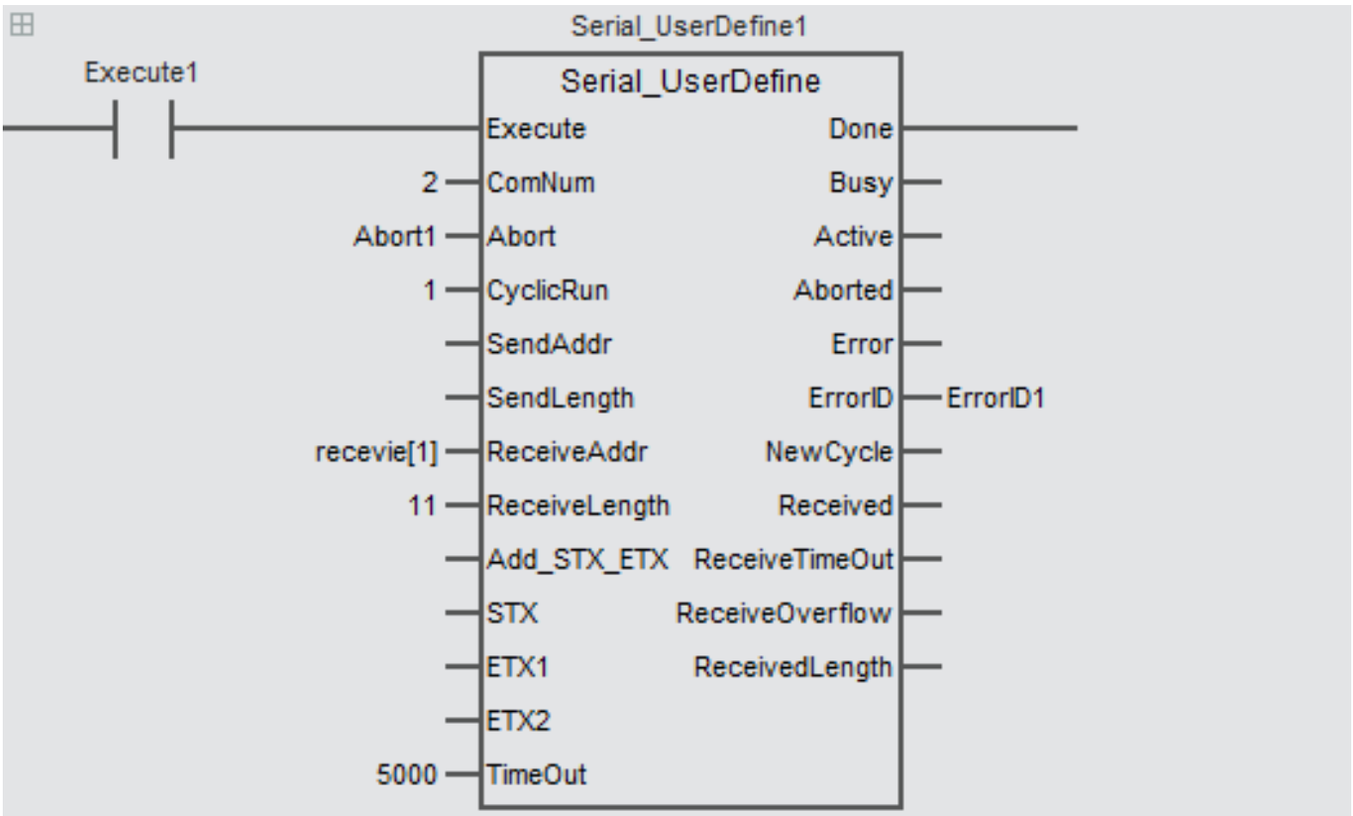
主站 ST 程序如下：

```
Serial_UserDefine0(Execute:= Execute1,
    ComNum:=1 ,
    Abort:= Abort1,
    CyclicRun:= 1,
    SendAddr:=SendBuffer[1] ,
    SendLength:= SendLength,
    ReceiveAddr:= ,
    ReceiveLength:= ,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:=5000 ,);
```

COM2 通讯口接收数据时，程序变量规划如下表所示：

变量名称	分配地址	数据类型	初始值	说明
Serial_UserDefine1		Serial_UserDefine		自定义协议收发指令
Execute1		BOOL		自定义协议收发指令执行条件
Abort1		BOOL		中断循环发送
recevie	%MB100	ARRAY[1..11] OF USINT		发送数据缓存地址

从站梯形图程序如下：



从站 ST 程序如下：

```
Serial_UserDefine1(Execute:= Execute1,
    ComNum:= 2,
    Abort:= Abort1,
    CyclicRun:=1 ,
    SendAddr:= ,
    SendLength:= ,
    ReceiveAddr:= recevie[1],
    ReceiveLength:= 11,
    Add_STX_ETX:= ,
    STX:= ,
    ETX1:= ,
    ETX2:= ,
    TimeOut:= 5000,);
```

- **程序流程说明：**

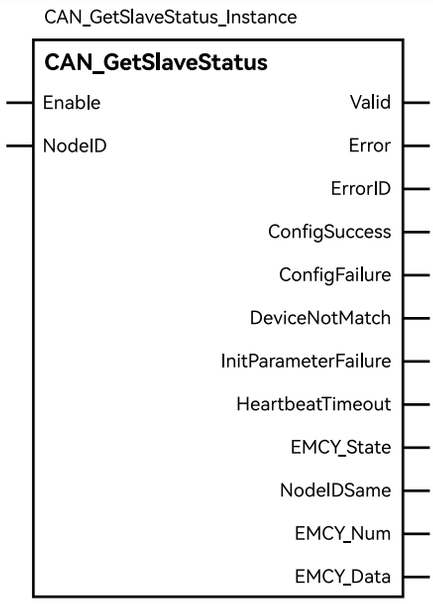
第一步：将欲发送的数据写入到发送缓存区域中。在此范例中，需要将 16 进制数 01 10 15 00 00 01 02 00 08 E3 57 依序写入到 %MB0 至 %MB10 中（SendBuffer[1]~SendBuffer[11]），接收数据存入到 %MB100 至 %MB110 中（recevie [1]~recevie[11]）。根据发送和接收的数据，则发送数据长度为 11，接收数据长度为 11。输入参数 SendBuffer（发送数据缓存起始地址）指定的数组长度要等于或者大于 11，输入参数 recevie（接收数据缓存起始地址）指定的数组长度要等于或者大于 11。

第二步：输入变量 CyclicRun 设置为 TRUE（循环发送和接收），输入参数执行由 FALSE 变为 TRUE 后，UserDefine1 和 UserDefine2 指令触发执行，UserDefine1 指定 COM1 发送数据，UserDefine2 指定 COM2 接收数据。

第 3 章　CAN通讯

3.1 CAN_GetSlaveStatus（CANopen从站状态获取指令）

获取 CANopen 从站状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
CAN_Get-SlaveStatus	CANopen 从站状态获取	FB		CAN_GetSlaveStatus_Instance(Enable:= 参数 , NodeID:= 参数 , Valid=> 参数 , Error=> 参数 , ErrorID=> 参数 , ConfigSuccess=> 参数 , ConfigFailure=> 参数 , DeviceNotMatch=> 参数 , InitParameterFailure=> 参数 , HeartbeatTimeout=> 参数 , EMCY_State=> 参数 , NodeIDSame=> 参数 , EMCY_Num=> 参数 , EMCY_Data=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
NodeID	从站站号	USINT	参考通讯指令规格	不可缺省	指定 CANopen 从站的站号。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	TRUE: 指令输出变量有效。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
ConfigSuccess	配置从站成功	BOOL	TRUE / FALSE	TRUE: 配置从站成功。
ConfigFailure	配置从站失败	BOOL	TRUE / FALSE	TRUE: 配置从站失败。
DeviceNot-Match	实际连接从站与配置从站不符	BOOL	TRUE / FALSE	TRUE: 实际连接从站与配置从站不符。
InitParameter-Failure	参数初始化失败	BOOL	TRUE / FALSE	TRUE: 参数初始化失败。
HeartbeatTim-eout	从站超时掉线	BOOL	TRUE / FALSE	TRUE: 从站超时掉线。
EMCY_State	从站发送紧急报文	BOOL	TRUE / FALSE	TRUE: 从站发送紧急报文时为 TRUE。
NodeIDSame	从站站号和主站站号相同	BOOL	TRUE / FALSE	TRUE: 从站站号和主站站号相同。
EMCY_Num	从站发送紧急报文次数	USINT	0~255	从站发送紧急报文次数。从站发送一次紧急报文，该参数的值加一。

EMCY_Data	从站发送紧急报文数据	ARRAY[1..5] OF CAN_EMCY_Type		从站发送紧急报文数据。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时。 错误消除时。
ConfigSuccess	配置从站成功时。	Enable 从 TRUE 变为 FALSE 时。
ConfigFailure	配置从站失败时。	Enable 从 TRUE 变为 FALSE 时。
DeviceNot-Match	实际连接从站与配置不符时。	Enable 从 TRUE 变为 FALSE 时。
InitParameter-Failure	参数初始化失败时。	Enable 从 TRUE 变为 FALSE 时。
HeartbeatTim-eout	HeartBeat 超时。	Enable 从 TRUE 变为 FALSE 时。
EMCY_State	从站发生报警时。	Enable 从 TRUE 变为 FALSE 时。
NodeIDSame	站号重复时。	Enable 从 TRUE 变为 FALSE 时。

◆ 功能说明

• 基本功能说明

该指令用于获取 CANopen 网络中站号为 NodeID 的从站的当前状态。

注意：该指令仅在控制器的 CANopen 端口配置为主站模式时有效。

3.2 CAN_GetSlaveStatus使用范例

- **目标需求**

通过 CAN_GetSlaveStatus 指令获取 M500S Series 的 CANopen 从站状态。

- **软件配置**

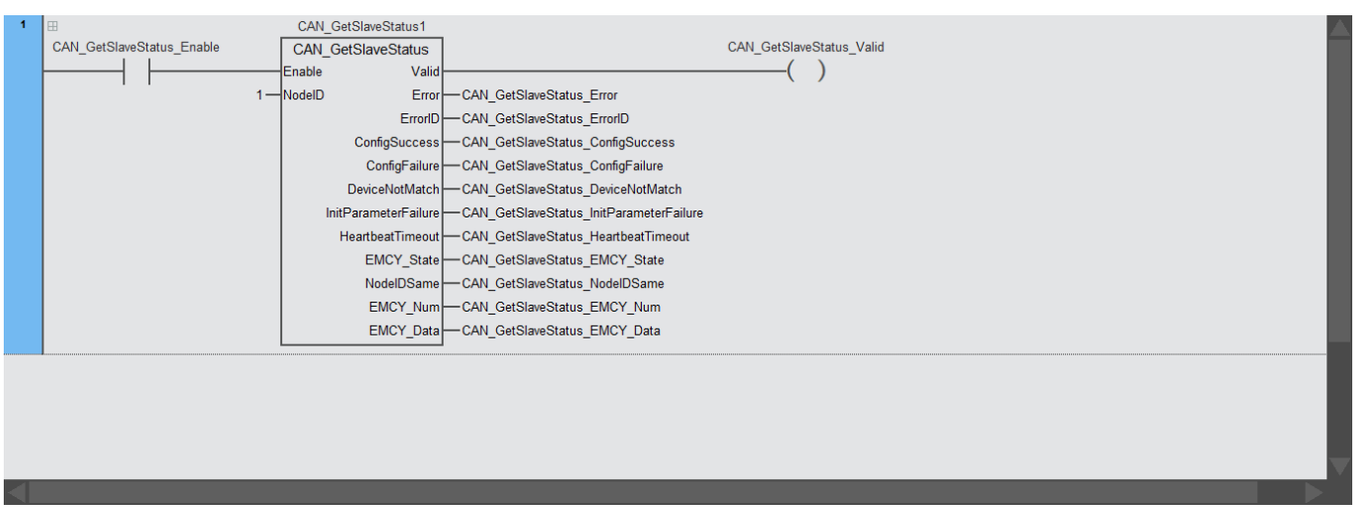
参考“CANopen 网络设置方法”章节说明。

- **指令配置**

变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	CAN_GetSlaveStatus1		CAN_GetSlaveStatus		
VAR	CAN_GetSlaveStatus_Enable		BOOL		
VAR	CAN_GetSlaveStatus_Valid		BOOL		
VAR	CAN_GetSlaveStatus_Error		BOOL		
VAR	CAN_GetSlaveStatus_ErrorID		WORD		
VAR	CAN_GetSlaveStatus_ConfigSuccess		BOOL		
VAR	CAN_GetSlaveStatus_ConfigFailure		BOOL		
VAR	CAN_GetSlaveStatus_DeviceNotMatch		BOOL		
VAR	CAN_GetSlaveStatus_InitParameterFailure		BOOL		
VAR	CAN_GetSlaveStatus_HeartbeatTimeout		BOOL		
VAR	CAN_GetSlaveStatus_EMCY_State		BOOL		
VAR	CAN_GetSlaveStatus_NodeIDSame		BOOL		
VAR	CAN_GetSlaveStatus_EMCY_Num		USINT		
VAR	CAN_GetSlaveStatus_EMCY_Data		ARRAY[1..5] OF CAN_EMCY_Type		

梯形图 LD:



结构化文本 ST:

```
CAN_GetSlaveStatus1(Enable:=CAN_GetSlaveStatus_Enable ,
    NodeID:=1 ,
    Valid=>CAN_GetSlaveStatus_Valid ,
    Error=>CAN_GetSlaveStatus_Error ,
    ErrorID=>CAN_GetSlaveStatus_ErrorID ,
    ConfigSuccess=>CAN_GetSlaveStatus_ConfigSuccess ,
    ConfigFailure=>CAN_GetSlaveStatus_ConfigFailure ,
    DeviceNotMatch=>CAN_GetSlaveStatus_DeviceNotMatch ,
    InitParameterFailure=>CAN_GetSlaveStatus_InitParameterFailure ,
    HeartbeatTimeout=>CAN_GetSlaveStatus_HeartbeatTimeout ,
    EMCY_State=>CAN_GetSlaveStatus_EMCY_State ,
    NodeIDSame=>CAN_GetSlaveStatus_NodeIDSame ,
    EMCY_Num=>CAN_GetSlaveStatus_EMCY_Num ,
    EMCY_Data=>CAN_GetSlaveStatus_EMCY_Data
    );
```

- **程序流程说明**

步骤一：NodeID 写入从站节点号。

步骤二 : 对 CAN_GetSlaveStatus_Enable 写入 True 后通过 CAN_GetSlaveStatus_ConfigSuccess 查看配置从站是否成功。

步骤三：当 CAN_GetSlaveStatus_Enable 为 True 时，CAN_GetSlaveStatus_Error 从 False 变成 True 后可以通过 CAN_GetSlaveStatus_ErrorID 查看错误代码。

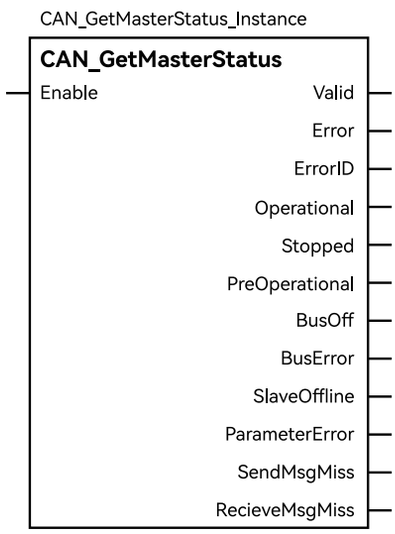
步骤四：当 CAN_GetSlaveStatus_Enable 为 True 时，当 CAN_GetSlaveStatus_DeviceNotMatch 从 False 变成 True 后检查配置的从站是否和实物一致以及站号是否一致。

步骤五：当 CAN_GetSlaveStatus_Enable 为 True 时，当 CAN_GetSlaveStatus_EMCY_State 从 False 变成 True 后检查配置的从站站号是否冲突。

步骤六：当 CAN_GetSlaveStatus_Enable 为 True 时，可以通过 CAN_GetSlaveStatus_EMCY_Num 查看从站报警数量，CAN_GetSlaveStatus_EMCY_Data 为从站报警内容。

3.3 CAN_GetMasterStatus（CANopen主站状态获取指令）

获取 CANopen 主站状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
CAN_GetMas-terStatus	CANopen 主站状态获取	FB		CAN_GetMasterStatus_Instance(Enable:= 参数 , Valid=> 参数 , Error=> 参数 , ErrorID=> 参数 , Operational=> 参数 , Stopped=> 参数 , PreOperational=> 参数 , BusOff=> 参数 , BusError=> 参数 , SlaveOffline=> 参数 , ParameterError=> 参数 , SendMsgMiss=> 参数 , RecieveMsgMiss=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数为 TRUE 时执行该指令。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	TRUE: 指令输出变量有效。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Operational	运行	BOOL	TRUE / FALSE	主站在 CANopen 网络中的状态。 TRUE: 主站处于运行状态。
Stopped	停止	BOOL	TRUE / FALSE	主站在 CANopen 网络中的状态。 TRUE: 主站处于停止状态。
PreOperational	预运行	BOOL	TRUE / FALSE	主站在 CANopen 网络中的状态。 TRUE: 主站处于预运行状态。
BusOff	总线不能通讯	BOOL	TRUE / FALSE	TRUE: 总线不能通讯。 该变量为 TRUE 时，需要检测现场硬件接线是否正确，现场环境是否有干扰，设备是否接地等。
BusError	总线异常	BOOL	TRUE / FALSE	TRUE: 总线发生错误。如现场环境差，有干扰等情况时，总线数据传送多次才可以传送成功时可能会发生这种情形。
SlaveOffline	从站掉线	BOOL	TRUE / FALSE	TRUE: 主站内配置的任何一个从站，和主站建立连接后掉线时。
ParameterError	配置从站参数时出错	BOOL	TRUE / FALSE	TRUE: 主站配置从站参数时发生错误，即主站不能和从站建立连接，如软件配置的从站参数，从站不支持时会发生该错误。
SendMsgMiss	发送数据错误	BOOL	TRUE / FALSE	TRUE: 主站发送数据时出错，如发送数据过快导致发送缓存区满时，该变量值为 TRUE。
RecieveMs-gMiss	接收数据错误	BOOL	TRUE / FALSE	TRUE: 主站接收数据时出错，如接收数据快导致接收缓存区满时，该变量值为 TRUE。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时。 错误消除时。
Operational	主站处于运行状态。	Enable 从 TRUE 变为 FALSE 时。 主站离开运行状态时。
Stopped	主站处于停止状态。	Enable 从 TRUE 变为 FALSE 时。 主站离开停止状态时。
PreOperational	主站处于预运行状态。	Enable 从 TRUE 变为 FALSE 时。 主站离开预运行状态时。
BusOff	总线不能通讯。	Enable 从 TRUE 变为 FALSE 时。 总线恢复时。
BusError	总线异常时。	Enable 从 TRUE 变为 FALSE 时。 总线异常消除时。
SlaveOffline	有从站掉线时。	Enable 从 TRUE 变为 FALSE 时。
ParameterError	配置从站参数出错时。	Enable 从 TRUE 变为 FALSE 时。
SendMsgMiss	发送数据错误时。	Enable 从 TRUE 变为 FALSE 时。
RecieveMs-gMiss	接收数据错误时。	Enable 从 TRUE 变为 FALSE 时。

◆ 功能说明

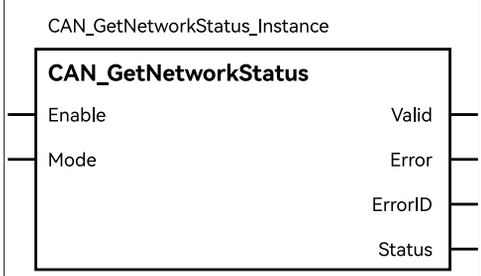
• 基本功能说明

该指令用于获取 CANopen 网络中主站的当前状态。

注意：该指令仅在控制器的 CANopen 端口配置为主站模式时有效。

3.4 CAN_GetNetworkStatus（获取CANopen网络中所有从站状态）

获取 CANopen 网络中所有从站的状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
CAN_GetNet-workStatus	获取 CANopen 网络中所有从站状态	FB		CAN_GetNetworkStatus_Instance(Enable:= 参数 , Mode:= 参数 , Valid=> 参数 , Error=> 参数 , ErrorID=> 参数 , Status=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
Mode	状态类别	USINT	1-3	不可缺省	设定获取 CANopen 网路中所有从站状态的类别。 1：是否配置从站； 2：从站是否与主站建立连接； 3：从站是否发送紧急报文。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	TRUE: 指令输出变量有效。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Status	状态	ARRAY[1..32] OF BOOL		所有从站的状态。 数组元素 1 对应站号为 1 的从站的状态，元素 2 对应站号为 2 的从站的状态。 值为 TRUE 表示与输入变量 Mode 设定的类别一致；反之为不一致。详细参考该指令的功能说明部分。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时。 错误消除时。

◆ 功能说明

• 基本功能说明

该指令用于获取 CANopen 网络中所有从站的状态。从站的状态类别通过输入变量 Mode 设定，获取的状态通过 Status 参数输出。如 Mode 设定值为 2 时，Status[1] 的值表示站号为 1 的从站与主站通讯是否正常，值为 True 则表示通讯正常，值为 Flase 则表示通讯异常；Status[2] 的值表示站号为 2 的从站与主站通讯是否正常，值为 True 则表示通讯正常，值为 Flase 则表示通讯异常；以此类推。如 Mode 设定值为 1 时，Status[1] 的值表示站号为 1 的从站是否被配置，值为 True 则表示成功配置，值为 Flase 则表示未配置；Status[2] 的值表示站号为 2 的从站是否被配置，值为 True 则表示成功配置，值为 Flase 则表示未配置；以此类推。

注意：该指令仅在控制器的 CANopen 端口配置为主站模式时有效。

3.5 CAN_GetNetworkStatus使用范例

- **目标需求**

通过 CAN_GetNetworkStatus 指令获取 CANopen 网络中所有从站状态。

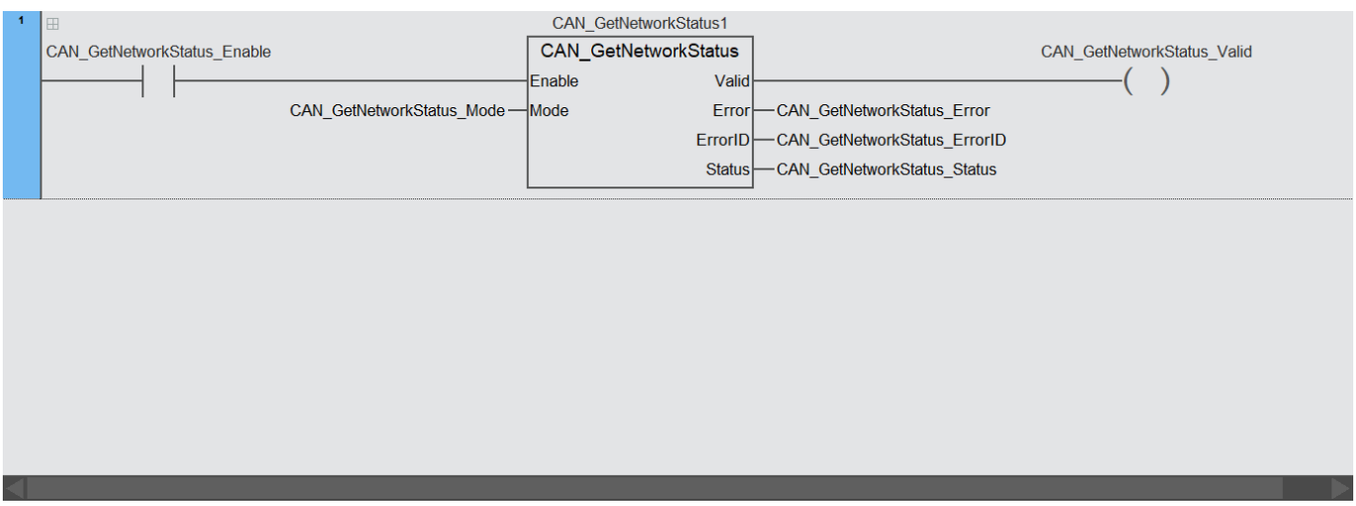
- **软件配置**

参考“CANopen 网络设置方法”章节说明。

变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	CAN_GetNetworkStatus1		CAN_GetNetworkStatus		
VAR	CAN_GetNetworkStatus_Enable		BOOL		
VAR	CAN_GetNetworkStatus_Mode		USINT		
VAR	CAN_GetNetworkStatus_Valid		BOOL		
VAR	CAN_GetNetworkStatus_Error		BOOL		
VAR	CAN_GetNetworkStatus_ErrorID		WORD		
VAR	CAN_GetNetworkStatus_Status		ARRAY[1..32] OF BOOL		

梯形图 LD:



结构化文本 ST:

```
CAN_GetNetworkStatus1(Enable:=CAN_GetNetworkStatus_Enable ,
    Mode:=CAN_GetNetworkStatus_Mode ,
    Valid=>CAN_GetNetworkStatus_Valid ,
    Error=>CAN_GetNetworkStatus_Error ,
    ErrorID=>CAN_GetNetworkStatus_ErrorID ,
    Status=>CAN_GetNetworkStatus_Status
    );
```

- **程序流程说明**

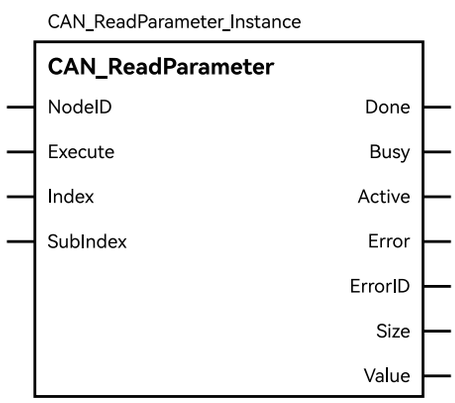
步骤一：CAN_GetNetworkStatus_Mode 写入 1 时可以通过 CAN_GetNetworkStatus_Status 查看是否配置从站，CAN_GetNetworkStatus_Mode 写入 2 时可以通过 CAN_GetNetworkStatus_Status 查看是否与从站建立连接，CAN_GetNetworkStatus_Mode 写入 3 时可以通过 CAN_GetNetworkStatus_Status 查看从站是否报警 。

步骤二：CAN_GetNetworkStatus_Mode 写入值后触发 CAN_GetNetworkStatus_Enable 后通过 CAN_GetNetworkStatus_Status 检测从站状态。

步骤三：当 CAN_GetNetworkStatus_Enable 为 True 时，CAN_GetNetworkStatus_Error 从 False 变成 True 后可以通过 CAN_GetNetworkStatus_ErrorID 查看错误代码。

3.6 CAN_ReadParameter（CANopen从站参数读取指令）

读取 CANopen 从站中的参数。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
CAN_ReadPar-ameter	CANopen 从站参数读取	FB		CAN_ReadParameter_Instance(NodeID:= 参数 , Execute:= 参数 , Index:= 参数 , SubIndex:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Error=> 参数 , ErrorID=> 参数 , Size=> 参数 , Value=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
NodeID	从站站号	USINT	1~63	不可缺省	指定从站站号。
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
Index	参数的索引	UINT	0~65535	0	设定欲读取参数的索引。
SubIndex	参数的子索引	USINT	0~255	0	设定欲读取参数的子索引。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令控制中	BOOL	TRUE / FALSE	指令控制中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Size	参数的数据类型	USINT		欲读取参数的类型。 1：Byte（1 字节）； 2：Word（2 字节）； 4：DWord（4 字节）。
Value	读取的参数值	UDINT		读取到的参数值

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Active	指令控制读参数时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。

Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于读取 CANopen 从站的参数值，欲读取的参数通过 Index 和 SubIndex 进行指定。从站参数的 Index 和 SubIndex 可通过从站相关资料获得。

该指令在 Execute 由 FALSE 变为 TRUE 时，按照输入变量设定的值读取从站的参数。

该指令执行时，将指定从站参数的值读取到输出变量 Value 内。输出变量 Value 的数据类型为 UDINT，当 Value 的数据类型和读取从站参数的数据类型不一致时，可通过数据类型转换指令将 Value 变量的数据类型转换为从站参数的数据类型。如读取从站参数的数据类型为 DINT，可以通过 UDINT_TO_DINT 指令将 Value 的值转换到 DINT 类型的变量内；如读取从站参数的数据类型为 INT，可以通过 UDINT_TO_INT 指令将 Value 的值转换到 INT 类型的变量内。

• 指令完成时机

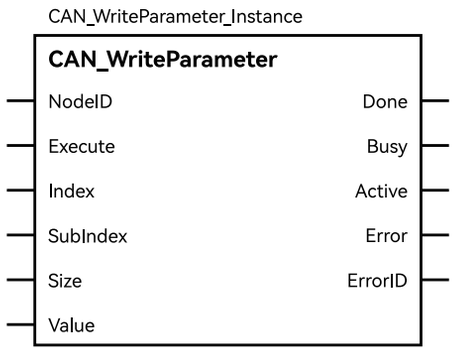
当读取到从站的参数值后，该指令完成，Done 位由 FALSE 变为 TRUE。

• 重启该指令

当指令执行完成后，Execute 再次由 FALSE 变为 TRUE 时，该指令可以重新执行；当指令正在执行中，Execute 再次由 FALSE 变为 TRUE 时，对指令的执行不会产生影响，指令仍按照未执行完成的输入变量执行指令。

3.7 CAN_WriteParameter（CANopen从站参数设置指令）

设置 CANopen 从站中的参数。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
CAN_WritePa-rameter	CANopen 从站参数设置	FB		CAN_WriteParameter_Instance(NodeID:= 参数 , Execute:= 参数 , Index:= 参数 , SubIndex:= 参数 , Size:= 参数 , Value:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
NodeID	从站站号	USINT	1~63	不可缺省	指定从站站号。
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
Index	参数的索引	UINT	0~65535	0	设定欲设置参数的索引。
SubIndex	参数的子索引	USINT	0~255	0	设定欲设置参数的子索引。
Size	参数的数据类型	USINT	1~4	不可缺省	欲设置参数的类型。 1：Byte（1 字节）； 2：Word（2 字节）； 4：DWord（4 字节）。
Value	设定值	UDINT	0 ~ 4294967295	0	设定值

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令控制中	BOOL	TRUE / FALSE	指令控制中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Active	指令控制写参数时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。

Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于设置 CANopen 从站的参数值，从站的参数通过 Index 和 SubIndex 进行指定。从站参数的 Index 和 SubIndex 可通过从站相关资料获得。设置的参数值，为输入变量 Value 的值。

该指令在 Execute 由 FALSE 变为 TRUE 时，按照输入变量设定的值设置从站的参数。

该指令执行时，将输入变量 Value 中的值设置到指定的从站参数中。输入变量 Value 的数据类型为 UDINT，当 Value 的数据类型和从站参数的数据类型不一致时，可通过数据类型转换指令将 Value 变量的数据类型转换为从站参数的数据类型。如从站参数的数据类型为 DINT，可以通过 UDINT_TO_DINT 指令将 Value 的值转换到 DINT 类型的变量内；如从站参数的数据类型为 INT，可以通过 UDINT_TO_INT 指令将 Value 的值转换到 INT 类型的变量内。

• 指令完成时机

当发出写参数值的命令后，该指令完成，Done 位由 FALSE 变为 TRUE。

• 重启该指令

当指令执行完成后，Execute 再次由 FALSE 变为 TRUE 时，该指令可以重新执行；当指令正在执行中，Execute 再次由 FALSE 变为 TRUE 时，对指令的执行不会产生影响，指令仍按照未执行完成的输入变量执行指令。

3.8 CAN参数读写使用范例

- **目标需求**

M511S CANopen 主站通过 CAN_ReadParameter 指令和 CAN_WriteParameter 指令对 M511S CANopen 从站参数 16#2000:01（索引：子索引）进行读取和写入。

- **软件配置**

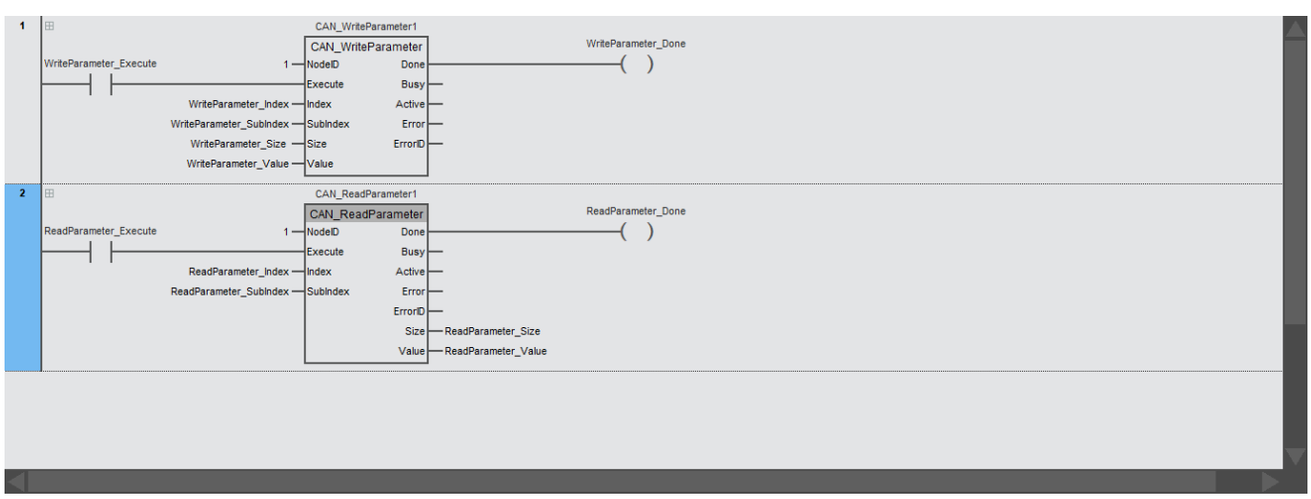
参考“CANopen 网络设置方法”章节说明。

- **指令配置**

变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	CAN_WriteParameter1		CAN_WriteParameter		
VAR	CAN_ReadParameter1		CAN_ReadParameter		
VAR	WriteParameter_Execute		BOOL		
VAR	ReadParameter_Execute		BOOL		
VAR	WriteParameter_Done		BOOL		
VAR	ReadParameter_Done		BOOL		
VAR	WriteParameter_Value		udint		
VAR	ReadParameter _Value		udint		
VAR	WriteParameter_Index		uint		
VAR	ReadParameter _Index		uint		
VAR	WriteParameter_SubIndex		usint		
VAR	ReadParameter_SubIndex		usint		
VAR	WriteParameter_Size		usint		
VAR	ReadParameter_Size		usint		

梯形图 LD:



结构化文本 ST:

```
CAN_WriteParameter1
(NodeID:=1 ,
    Execute:=WriteParameter_Execute ,
    Index:=WriteParameter_Index ,
    SubIndex:=WriteParameter_SubIndex ,
    Size:=WriteParameter_Size ,
    Value:=WriteParameter_Value,
    Done=>WriteParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=>
    );
CAN_ReadParameter1
(NodeID:=1 ,
    Execute:=ReadParameter_Execute ,
    Index:=ReadParameter_Index ,
    SubIndex:=ReadParameter_SubIndex ,
    Done=>ReadParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    Size=>ReadParameter_Size ,
    Value=>ReadParameter_Value
    );
```

- **程序流程说明**

步骤一：设置输入变量 NodeID 的值为 1，表示对站号为 1 的从站进行写参数。

步骤二：设置输入变量 WriteParameter_Index 的值为 16#2000，输入变量 WriteParameter_SubIndex 的值为 1，设置输入变量 WriteParameter_Size 的值为 2，输入变量 WriteParameter_Value 的值为 1。

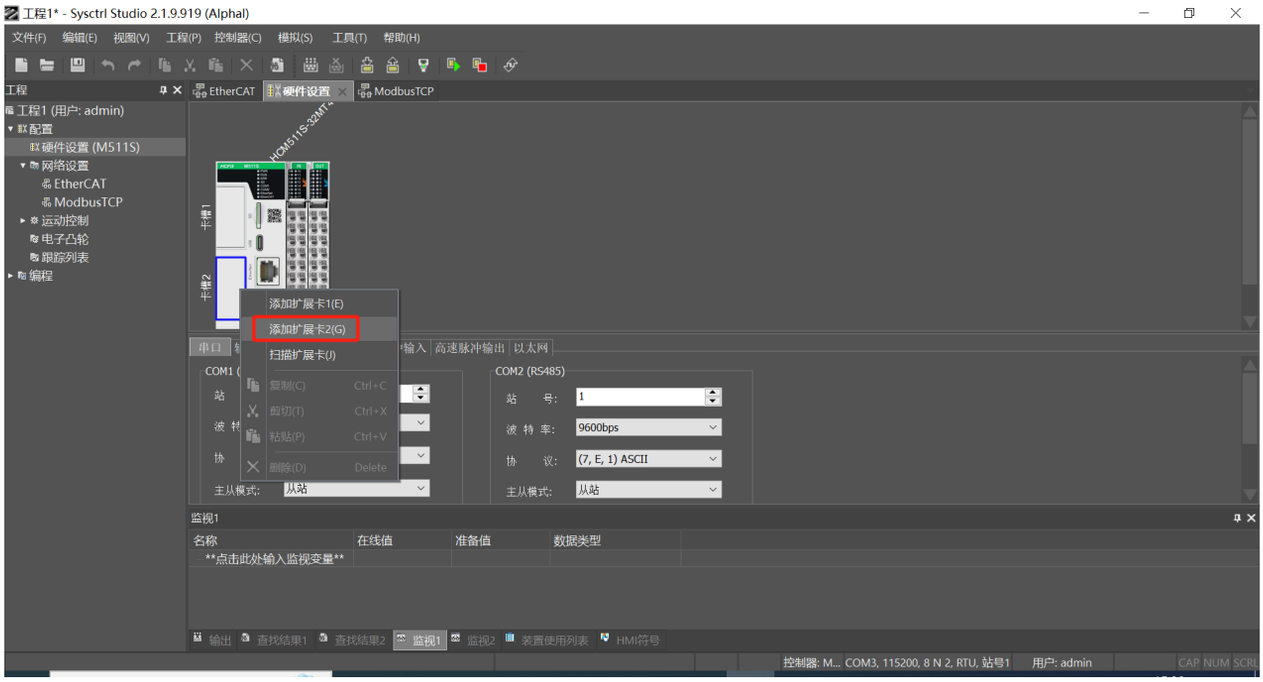
步骤三：输入变量 WriteParameter_Execute 由 FALSE 变为 TRUE 后，触发 CAN_WriteParameter 指令执行，WriteParameter_Done 位变为 TRUE 时表示参数写入完成。

步骤四：设置输入变量 ReadParameter_Index 的值为 16#2000，输入变量 ReadParameter_SubIndex 的值为 1。

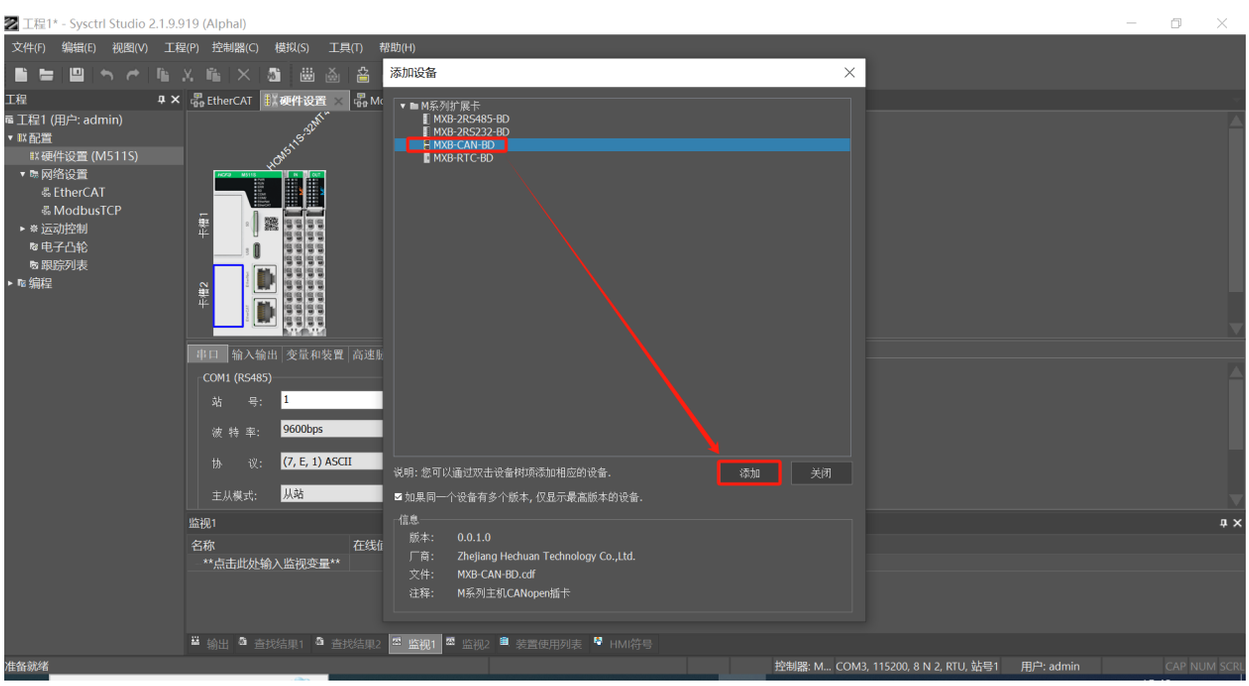
步骤五：输入变量 ReadParameter_Execute 由 FALSE 变为 TRUE 后，触发 CAN_ReadParameter 指令执行，ReadParameter_Done 位变为 TRUE 时表示参数读取完成，读取完成后，ReadParameter_Size 和 ReadParameter_Value 的值分别为 2 和 1，Read-Parameter_Value 的值为读取的参数值。

3.9 CANopen网络设置方法

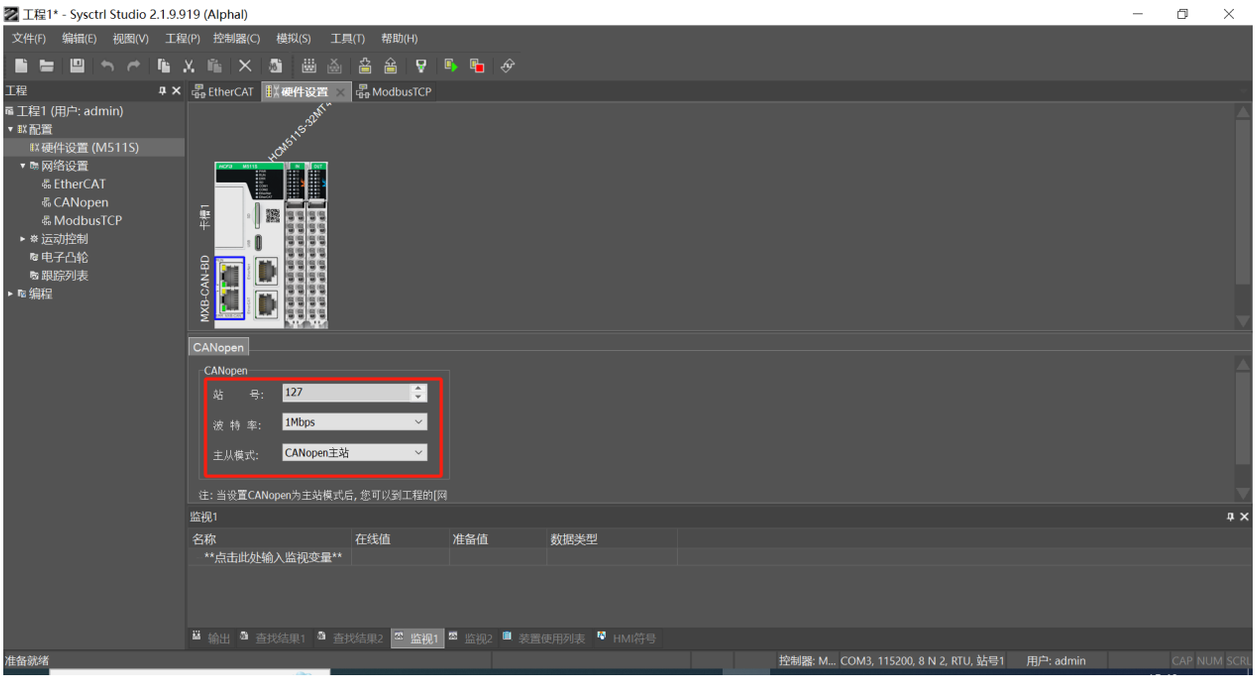
步骤一：双击硬件设置一单击卡槽后右击添加扩展卡。双击“硬件设置”，单击控制器对应的卡槽后右击，然后单击“添加扩展卡2”。



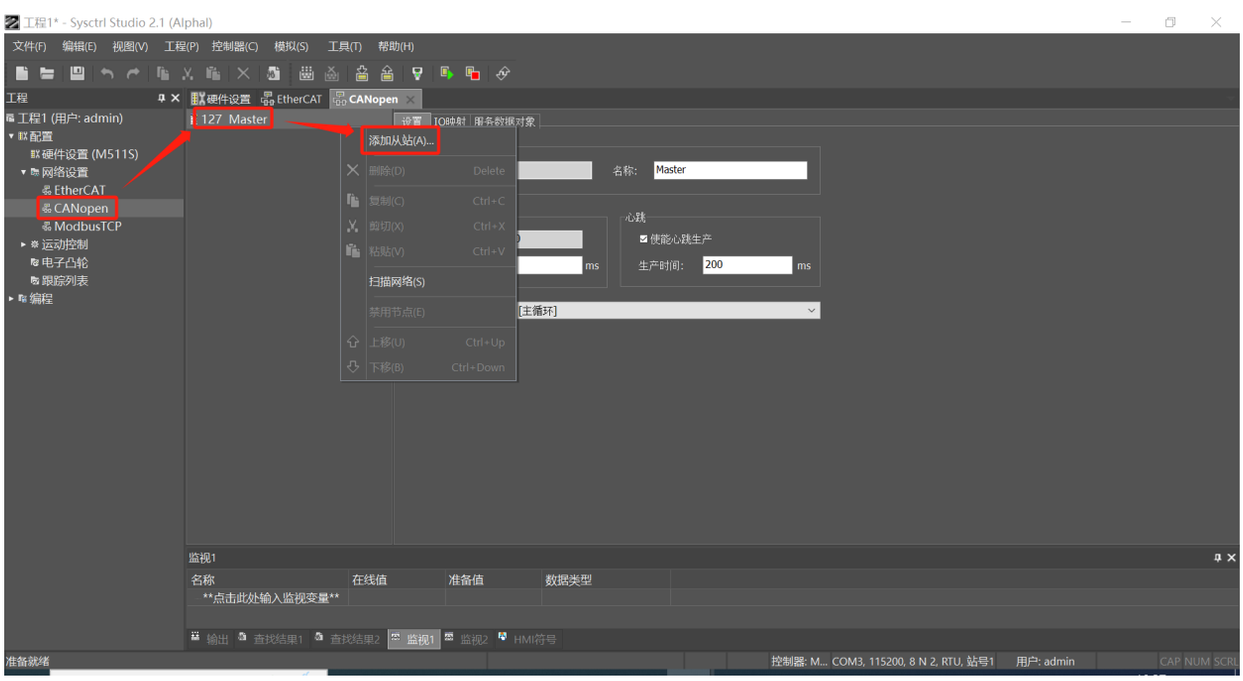
步骤二：双击“MXB-CAN-BD”或者单击选择“MXB-CAN-BD”后单击下图红色方框处“添加”按钮添加从站



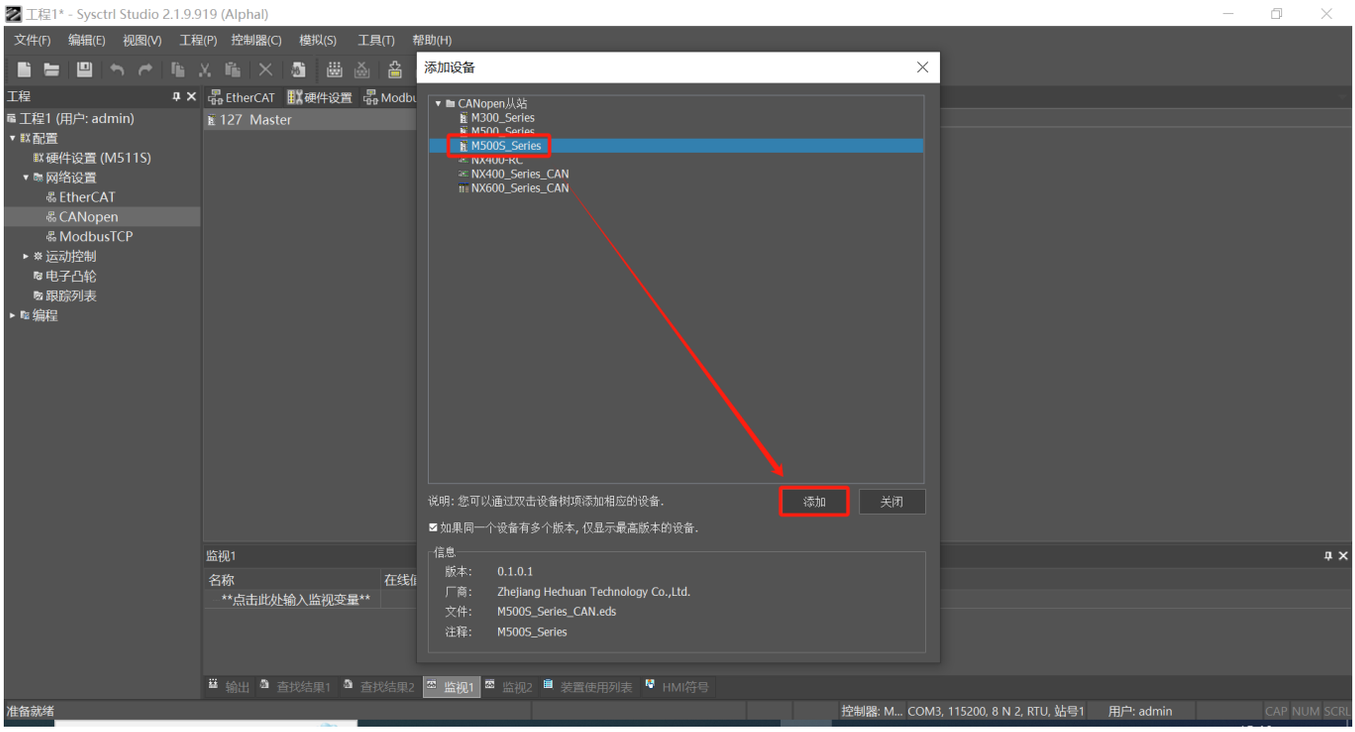
步骤三：设置主站站号、波特率和主从模式（主站和从站站号不能重复，波特率需保持一致）。



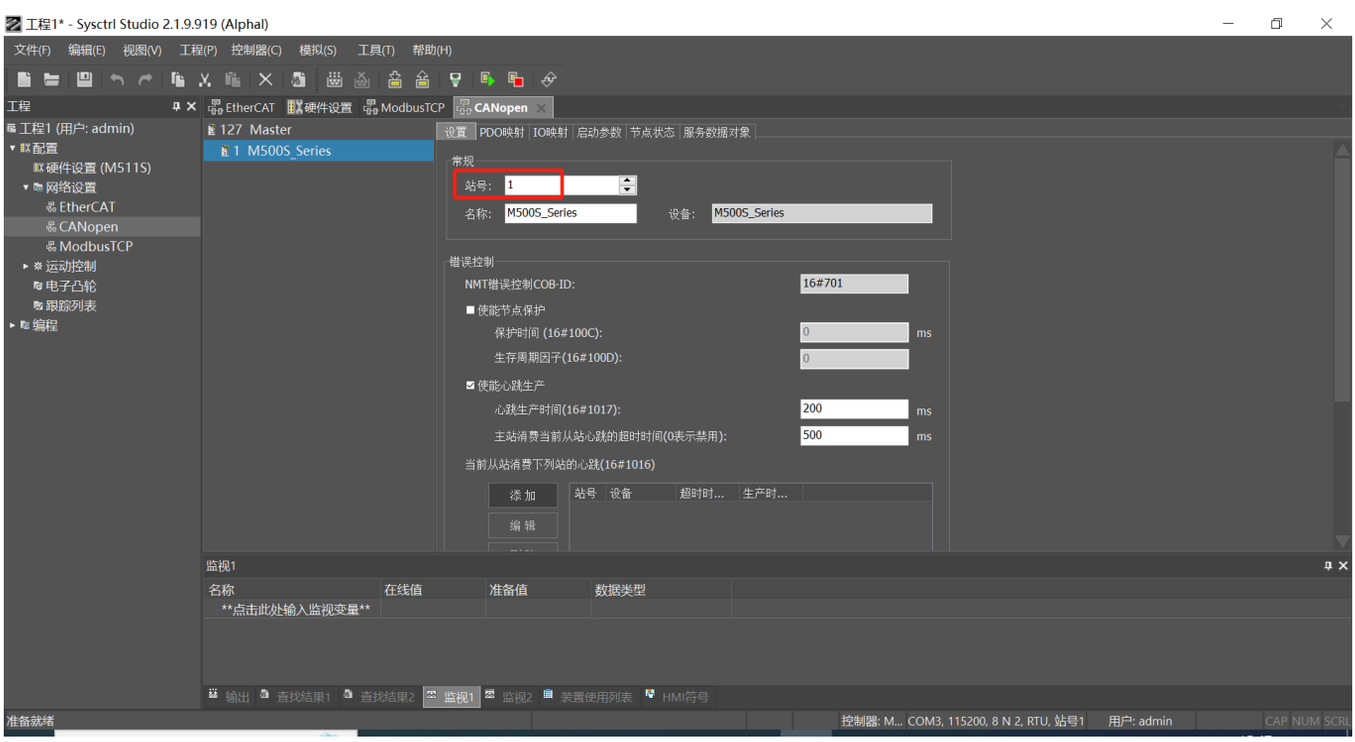
步骤四：单击“网络设置”，然后双击“CANopen”，单击“127 Master”后右击，单击“添加从站”。



步骤五：双击 M500S Series 添加从站；或者单击选择 M500S Series 后，单击下图红色方框处“添加”按钮添加从站。



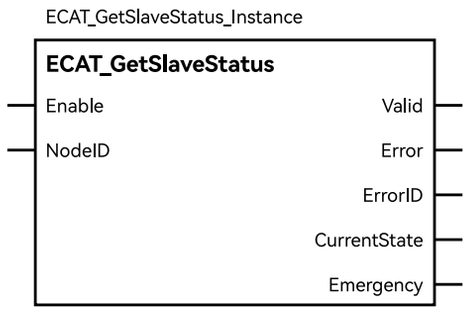
步骤六：指定从站站号（此处指定的从站站站号须和实际连接的从站站号一致）。



第 4 章　EtherCAT通讯

4.1 ECAT_GetSlaveStatus（EtherCAT从站状态获取指令）

获取 EtherCAT 从站状态。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ECAT_Get-SlaveStatus	EtherCAT 从站状态获取	FB		ECAT_GetSlaveStatus_Instance(Enable:= 参数 , NodeID:= 参数 , Valid=> 参数 , Error=> 参数 , ErrorID=> 参数 , CurrentState=> 参数 , Emergency=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Enable	启动	BOOL	TRUE 或 FALSE	FALSE	设为 TRUE，该指令执行；设为 FALSE，该指令不执行。
NodeID	从站 EtherCAT 地址	UINT	参考通讯指令规格	不可缺省	指定 EtherCAT 从站的地址。例如第一台从站的地址为 1001，第二台从站的地址为 1002，以此类推。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Valid	输出变量有效	BOOL	TRUE / FALSE	TRUE: 指令输出变量有效。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
CurrentState	从站当前状态	ECAT_State_Type		EtherCAT 从站的当前状态。 0：Unknown（未知状态）； 1：Init（初始化）； 2：PreOP（预运行）； 3：SafeOP（安全运行）； 4：OP（运行）。
Emergency	从站紧急报文中错误代码的值	UINT	0~65535	从站紧急报文中错误代码的值。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Valid	Enable 为 TRUE 时。	Enable 由 TRUE 变为 FALSE 时
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Enable 从 TRUE 变为 FALSE 时。 错误消除时。

◆ 功能说明

• 基本功能说明

该指令用于获取 EtherCAT 网络中指定从站的当前状态，从站通过输入变量 NodeID 指定。

主站（控制器）与 EtherCAT 从站的建立连接时，正常的从站状态变化过程为：Init（初始化）=> PreOP（预运行）=> SafeOP（安全运行） => OP（运行）。当从站状态处于 OP（运行）状态时，表示主站与该从站通讯连接建立完成，即主站与从站通讯正常。

如对 EtherCAT 从站进行控制或读取参数时，可以通过该指令判断从站状态处于 OP（运行）状态后，再进行相关的操作。

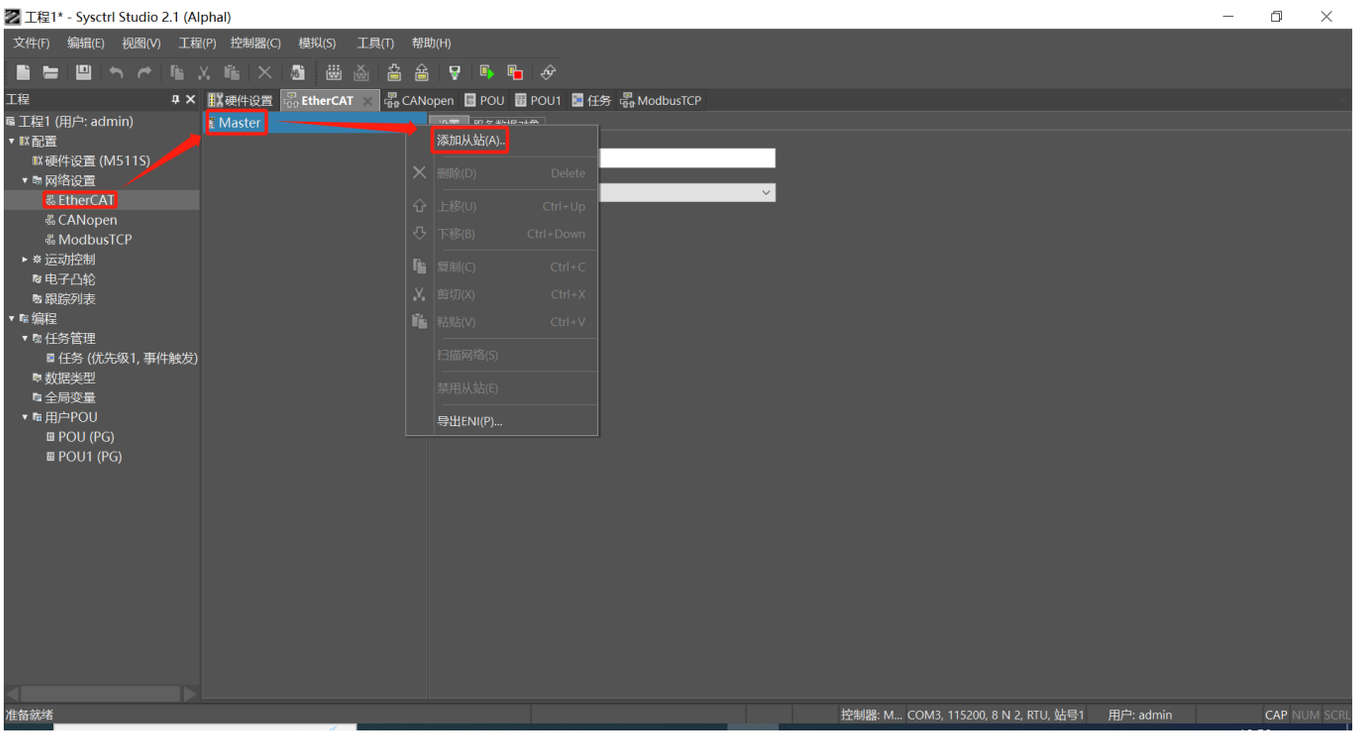
4.2 ECAT_GetSlaveStatus使用范例

- **目标需求**

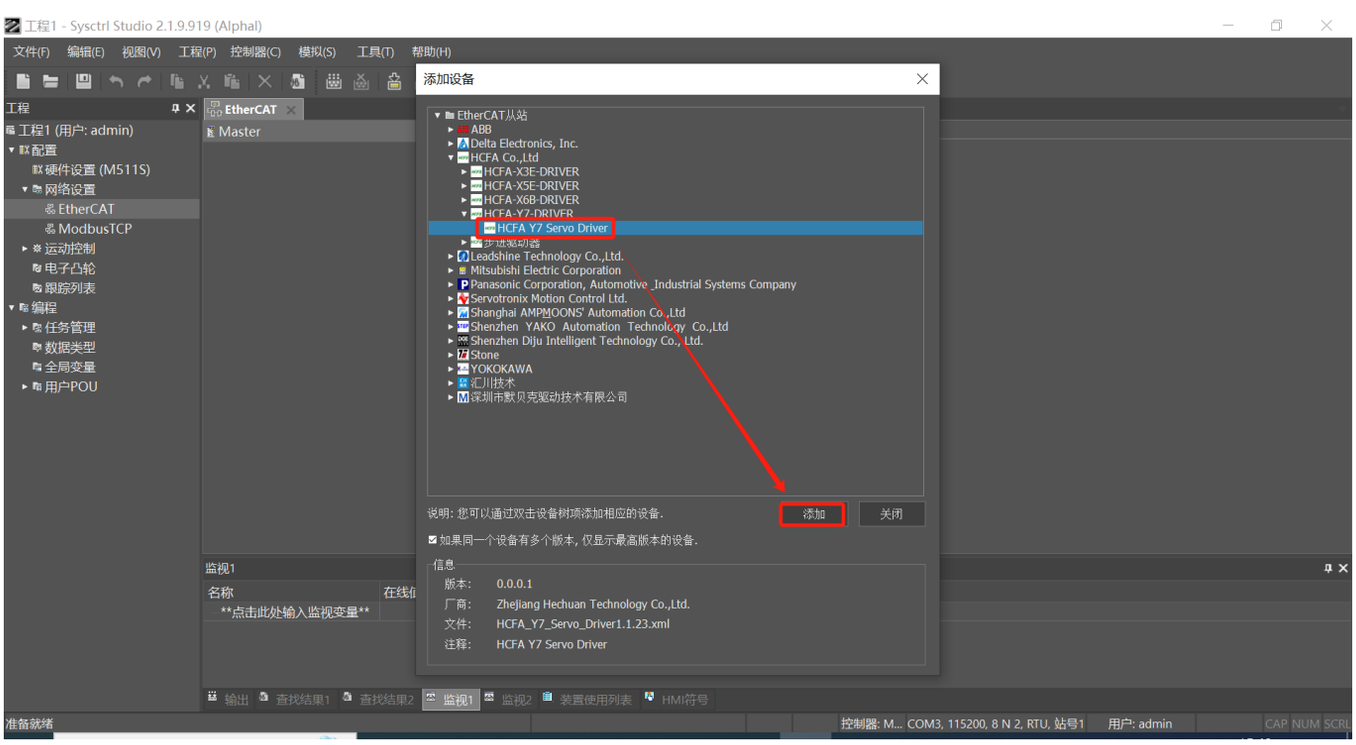
通过 ECAT_GetSlaveStatus 指令获取禾川 Y7 系列伺服驱动器 EtherCAT 从站状态。

- **软件配置**

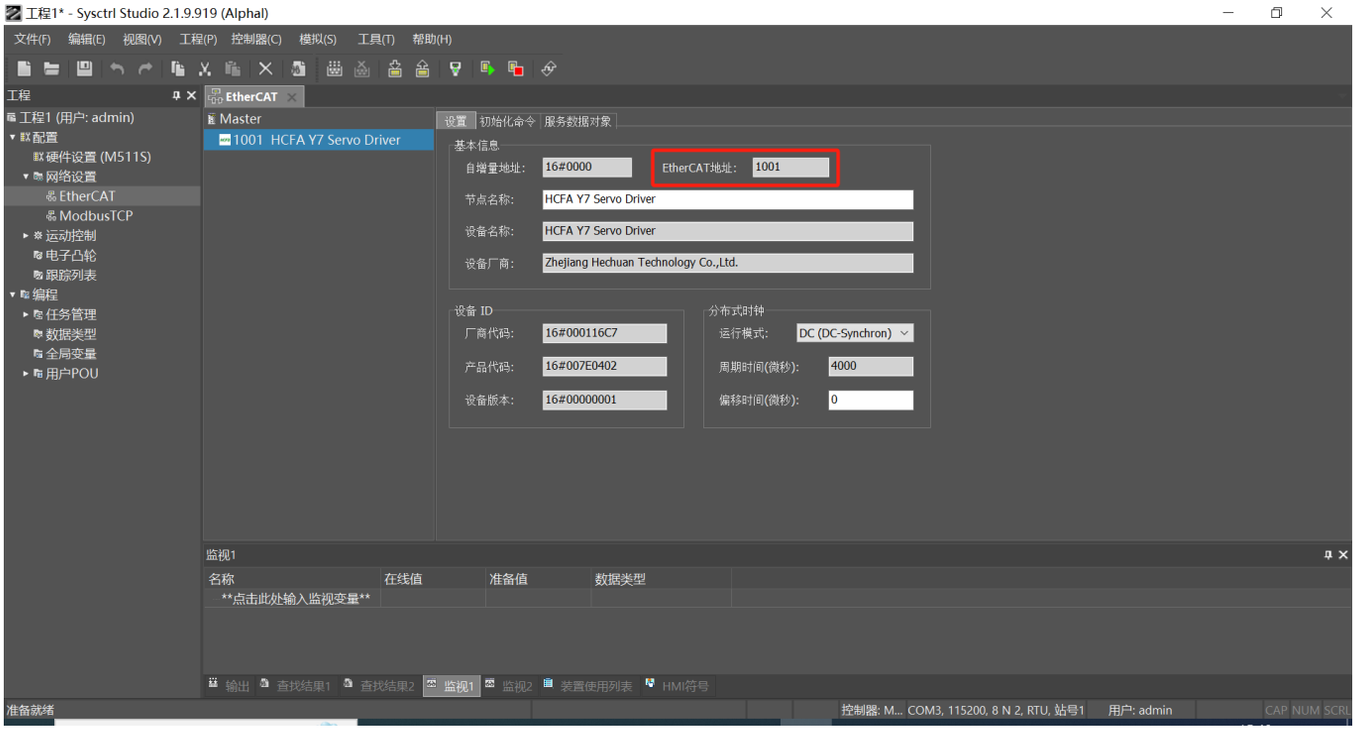
步骤一：单击“网络设置”，然后双击“EtherCAT”，单击 Master 后右击，单击添加从站。



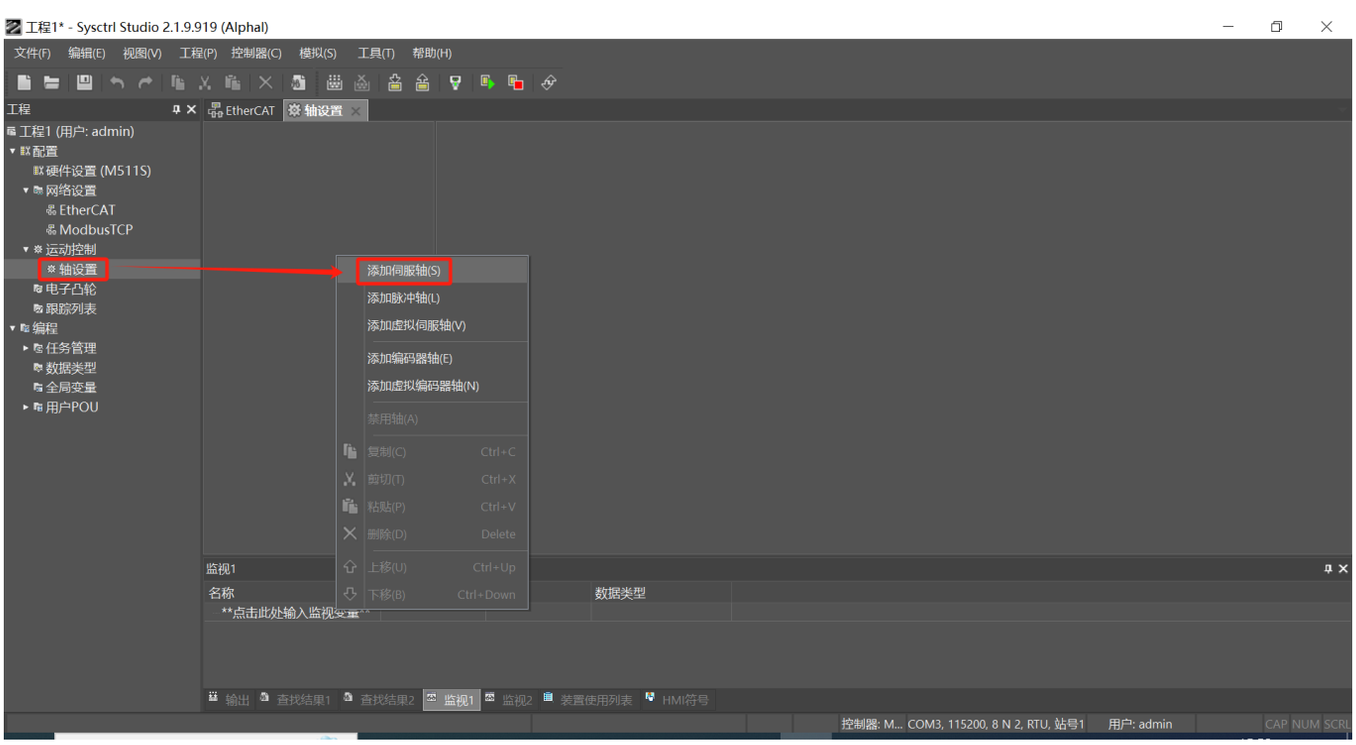
步骤二：双击 HCFA Y7 Servo Driver 添加从站，或者单击选择 HCFA Y7 Servo Driver 后，单击下图红色方框处添加按钮添加从站。



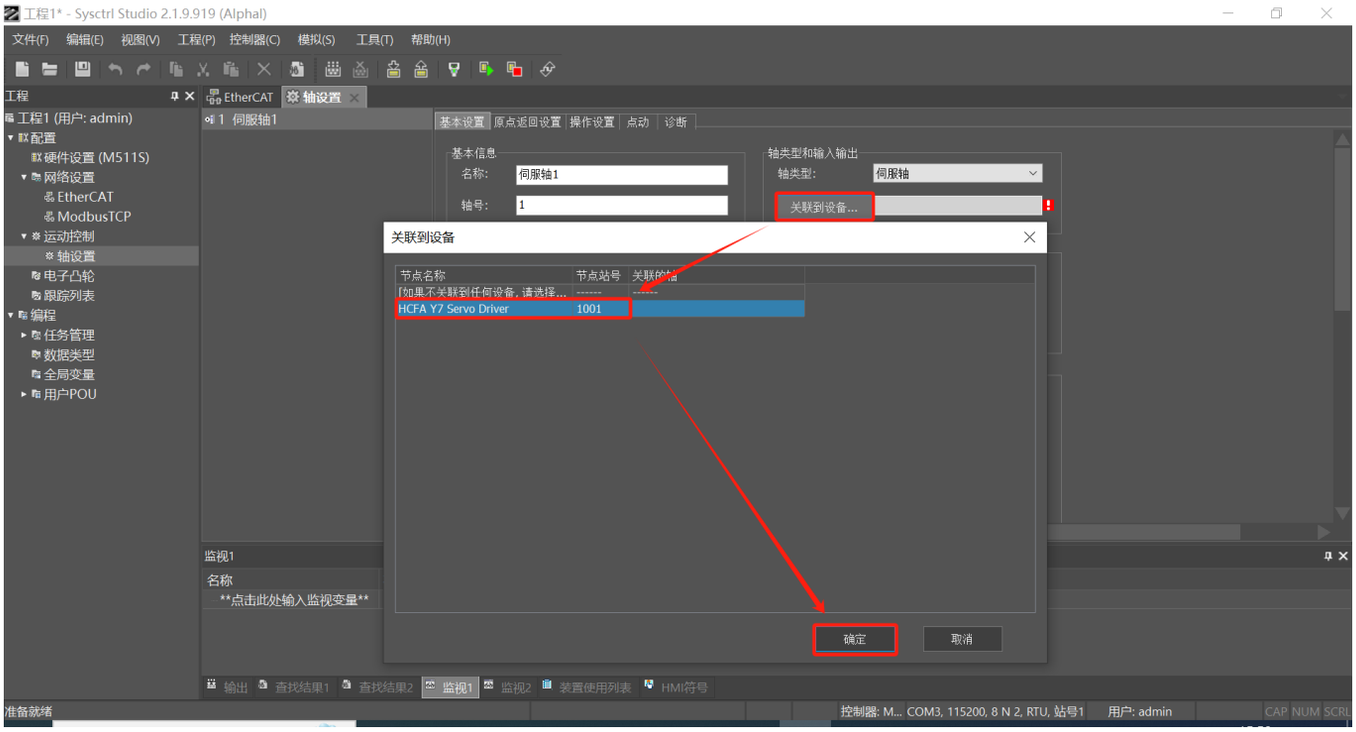
步骤三：第一台 EtherCAT 从站的 EtherCAT 地址为 1001，第二台 EtherCAT 从站的 EtherCAT 地址为 1002，以此类推。



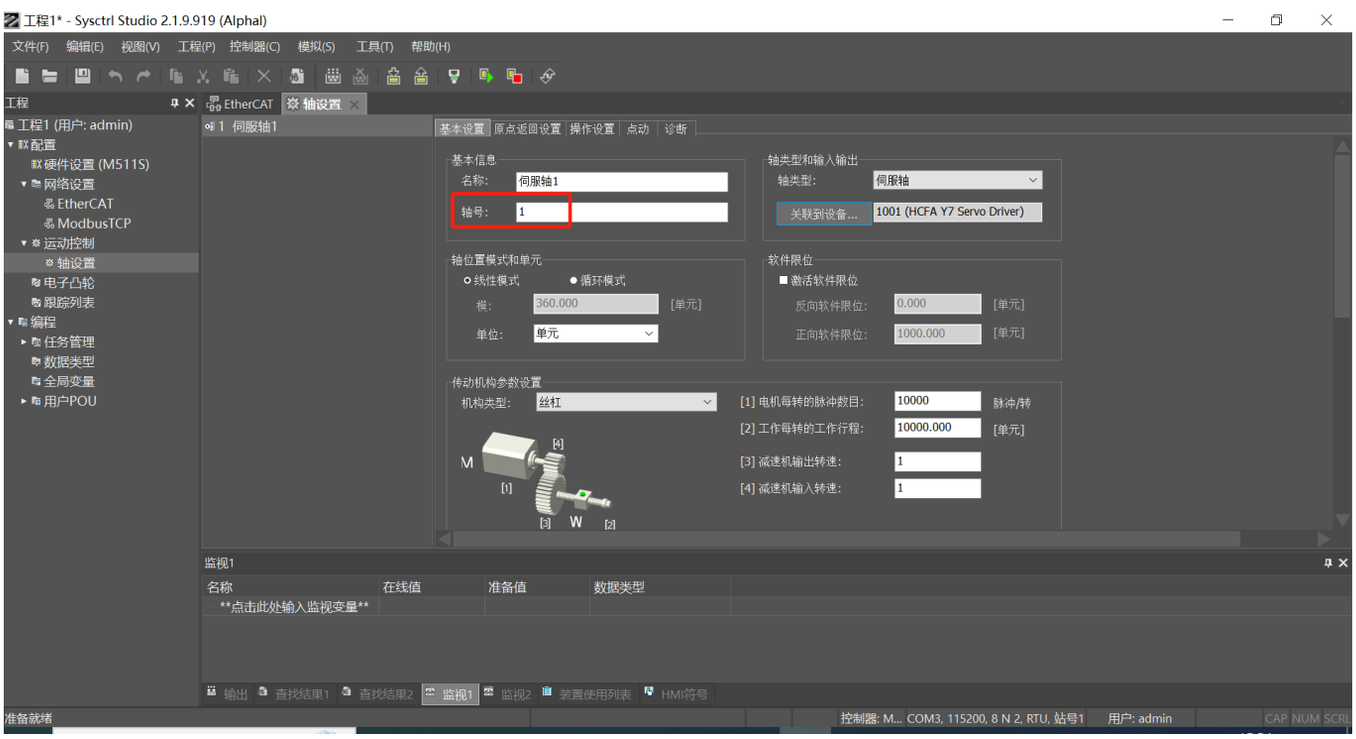
步骤四：双击轴设置 - 右击中间空白地方 - 单击添加伺服轴。



步骤五：单击“关联到设备”-> 单击“HCFA Y7 Servo Driver” -> 单击“确定”按钮。通过此步操作，轴和 EhterCAT 从站建立一一对应关系。



步骤六：在下图红色方框处设置轴号。



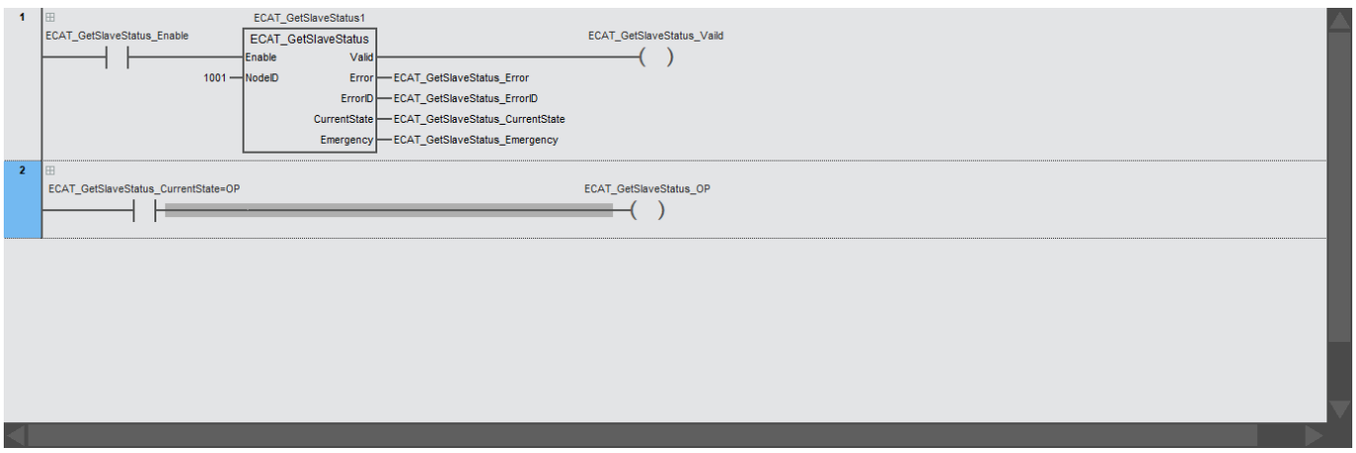
- **指令配置**

变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	ECAT_GetSlaveStatus1		ECAT_GetSlaveStatus		
VAR	ECAT_GetSlaveStatus_Enable		BOOL		
VAR	ECAT_GetSlaveStatus_Vaild		BOOL		
VAR	ECAT_GetSlaveStatus_Error		BOOL		
VAR	ECAT_GetSlaveStatus_ErrorID		WORD		

VAR	ECAT_GetSlaveStatus_CurrentState		ECAT_State_Type		
VAR	ECAT_GetSlaveStatus_Emergency		UINT		
VAR	ECAT_GetSlaveStatus_OP		BOOL		

梯形图 LD:



结构化文本 ST:

```
ECAT_GetSlaveStatus1(Enable:=ECAT_GetSlaveStatus_Enable ,
    NodeID:=1001 ,
    Valid=>ECAT_GetSlaveStatus_Vaild ,
    Error=>ECAT_GetSlaveStatus_Error ,
    ErrorID=>ECAT_GetSlaveStatus_ErrorID ,
    CurrentState=>ECAT_GetSlaveStatus_CurrentState ,
    Emergency=>ECAT_GetSlaveStatus_Emergency
    );

IF ECAT_GetSlaveStatus_CurrentState=OP THEN
    ECAT_GetSlaveStatus_OP:=TRUE;
ELSE
    ECAT_GetSlaveStatus_OP:=FALSE;
END_IF;
```

- **程序流程说明**

步骤一：设置输入变量 NodeID 的值为 1001，表示读取第 1 台 EtherCAT 从站的状态。

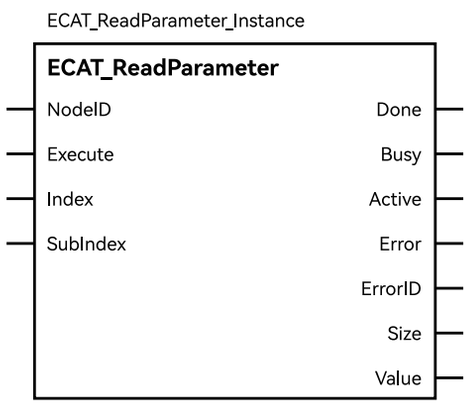
步骤二：输入变量 ECAT_GetSlaveStatus_Enable 为 True 时，ECAT_GetSlaveStatus_Vaild 变为 TRUE 后，通过 ECAT_GetSlaveStatus_CurrentState 查看从站当前状态。当 ECAT_GetSlaveStatus_CurrentState 的值为 OP 时，说明从站状态处于 OP（运行）状态，表示主站与从站通讯连接建立完成，即主站与从站通讯正常。

步骤四：输入变量 ECAT_GetSlaveStatus_Enable 为 True 时，ECAT_GetSlaveStatus_Vaild 变为 TRUE 后，当 ECAT_GetSlaveStatus_Emergency 的值不为 0 时可以通过 ECAT_GetSlaveStatus_Emergency 查看从站发送的紧急报文错误代码。

步骤五：输入变量 ECAT_GetSlaveStatus_Enable 为 True 时，ECAT_GetSlaveStatus_Error 从 False 变成 True 后，可以通过 ECAT_GetSlaveStatus_ErrorID 查看错误代码。

4.3 ECAT_ReadParameter（EtherCAT从站服务数据参数读取指令）

读取 EtherCAT 从站的参数值。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ECAT_ReadPar-ameter	EtherCAT 从站参数读取	FB		ECAT_ReadParameter_Instance(NodeID:= 参数 , Execute:= 参数 , Index:= 参数 , SubIndex:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Error=> 参数 , ErrorID=> 参数 , Size=> 参数 , Value=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
NodeID	从站 EtherCAT 地址	UINT	参考通讯指令规格（0）	不可缺省	指定 EtherCAT 从站的地址。例如第一台从站的地址为 1001，第二台从站的地址为 1002，以此类推。
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
Index	参数的索引	UINT	0~65535	0	设定欲读取参数的索引。
SubIndex	参数的子索引	USINT	0~255	0	设定欲读取参数的子索引。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令控制中	BOOL	TRUE / FALSE	指令控制中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Size	参数的数据类型	USINT	1~4	欲读取参数的类型。 1：Byte（1 字节）； 2：Word（2 字节）； 4：DWord（4 字节）。
Value	读取的参数值	UDINT		读取到的参数值

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。

Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Active	指令控制读参数时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于读取 EtherCAT 从站的参数值，欲读取的参数通过 Index 和 SubIndex 进行指定。从站参数的 Index 和 SubIndex 可通过从站相关资料获得。

该指令在 Execute 由 FALSE 变为 TRUE 时，按照输入变量设定的值读取从站的参数。

该指令执行时，将指定从站参数的值读取到输出变量 Value 内。输出变量 Value 的数据类型为 UDINT，当 Value 的数据类型和读取从站参数的数据类型不一致时，可通过数据类型转换指令将 Value 变量的数据类型转换为从站参数的数据类型。如读取从站参数的数据类型为 DINT，可以通过 UDINT_TO_DINT 指令将 Value 的值转换到 DINT 类型的变量内；如读取从站参数的数据类型为 INT，可以通过 UDINT_TO_INT 指令将 Value 的值转换到 INT 类型的变量内。

• 指令完成时机

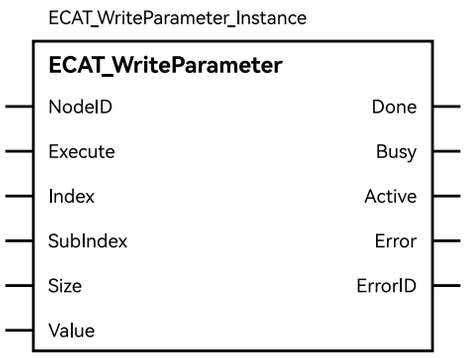
当读取到从站的参数值后，该指令完成，Done 位由 FALSE 变为 TRUE。

• 重启该指令

当指令执行完成后，Execute 再次由 FALSE 变为 TRUE 时，该指令可以重新执行；当指令正在执行中，Execute 再次由 FALSE 变为 TRUE 时，对指令的执行不会产生影响，指令仍按照未执行完成的输入变量执行指令。

4.4 ECAT_WriteParameter（EtherCAT从站服务数据参数设置指令）

设置 EtherCAT 从站中的参数。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
ECAT_WritePa-rameter	EtherCAT 从站参数设置	FB		ECAT_WriteParameter_Instance(NodeID:= 参数 , Execute:= 参数 , Index:= 参数 , SubIndex:= 参数 , Size:= 参数 , Value:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
NodeID	从站 EtherCAT 地址	UINT	参考通讯指令规格	不可缺省	指定 EtherCAT 从站的地址。例如第一台从站的地址为 1001，第二台从站的地址为 1002，以此类推。
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
Index	参数的索引	UINT	0~65535	0	设定欲设置参数的索引。
SubIndex	参数的子索引	USINT	0~255	0	设定欲设置参数的子索引。
Size	参数的数据类型	USINT	1~4	不可缺省	欲设置参数的类型。 1：Byte（1 字节）； 2：Word（2 字节）； 4：DWord（4 字节）。
Value	设定值	UDINT	0 ~ 4294967295	0	设定值

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令控制中	BOOL	TRUE / FALSE	指令控制中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。

Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Active	指令控制写参数时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于设置 EtherCAT 从站的参数值，从站的参数通过 Index 和 SubIndex 进行指定。从站参数的 Index 和 SubIndex 可通过从站相关资料获得。设置的参数值，为输入变量 Value 的值。

该指令在 Execute 由 FALSE 变为 TRUE 时，按照输入变量设定的值设置从站的参数。

该指令执行时，将输入变量 Value 中的值设置到指定的从站参数中。输入变量 Value 的数据类型为 UDINT，当 Value 的数据类型和从站参数的数据类型不一致时，可通过数据类型转换指令将 Value 变量的数据类型转换为从站参数的数据类型。如从站参数的数据类型为 DINT，可以通过 UDINT_TO_DINT 指令将 Value 的值转换到 DINT 类型的变量内；如从站参数的数据类型为 INT，可以通过 UDINT_TO_INT 指令将 Value 的值转换到 INT 类型的变量内。

• 指令完成时机

当发出写参数值的命令后，该指令完成，Done 位由 FALSE 变为 TRUE。

• 重启该指令

当指令执行完成后，Execute 再次由 FALSE 变为 TRUE 时，该指令可以重新执行；当指令正在执行中，Execute 再次由 FALSE 变为 TRUE 时，对指令的执行不会产生影响，指令仍按照未执行完成的输入变量执行指令。

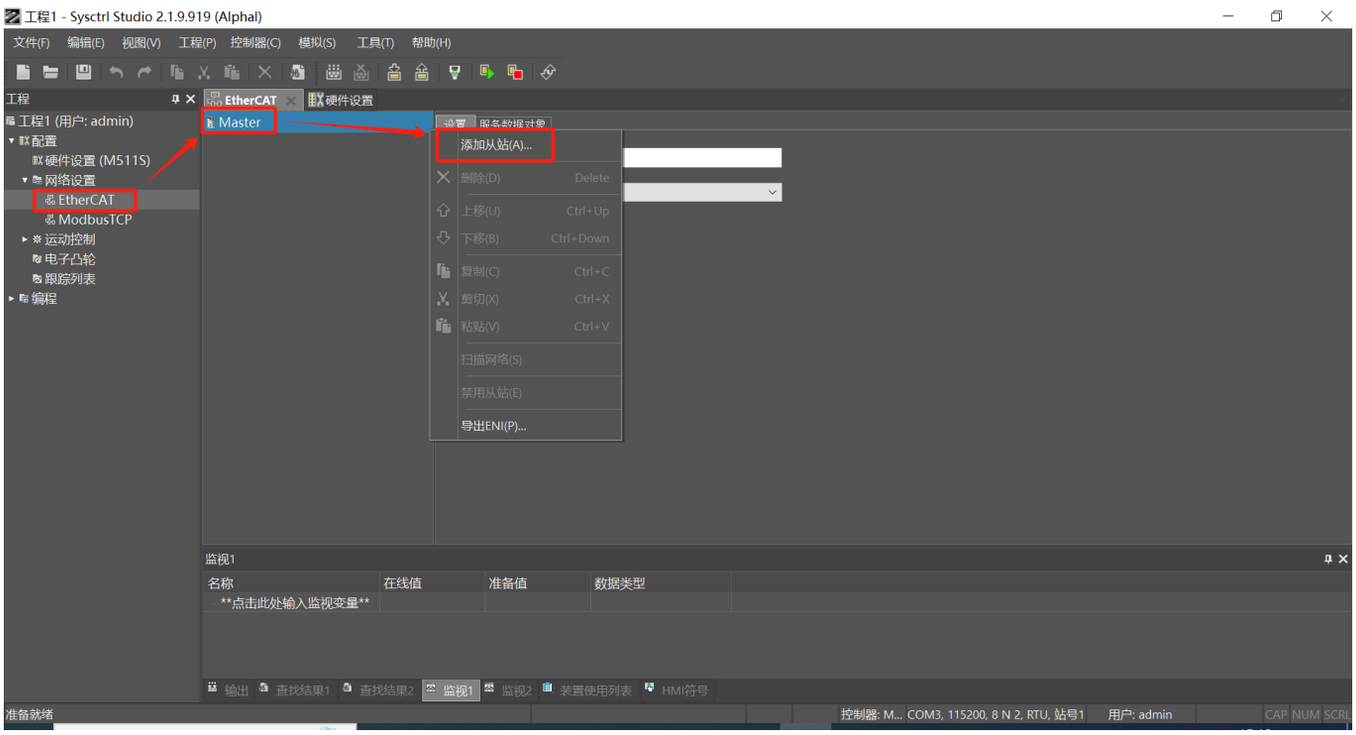
4.5 ECAT参数读写使用范例

- **目标需求**

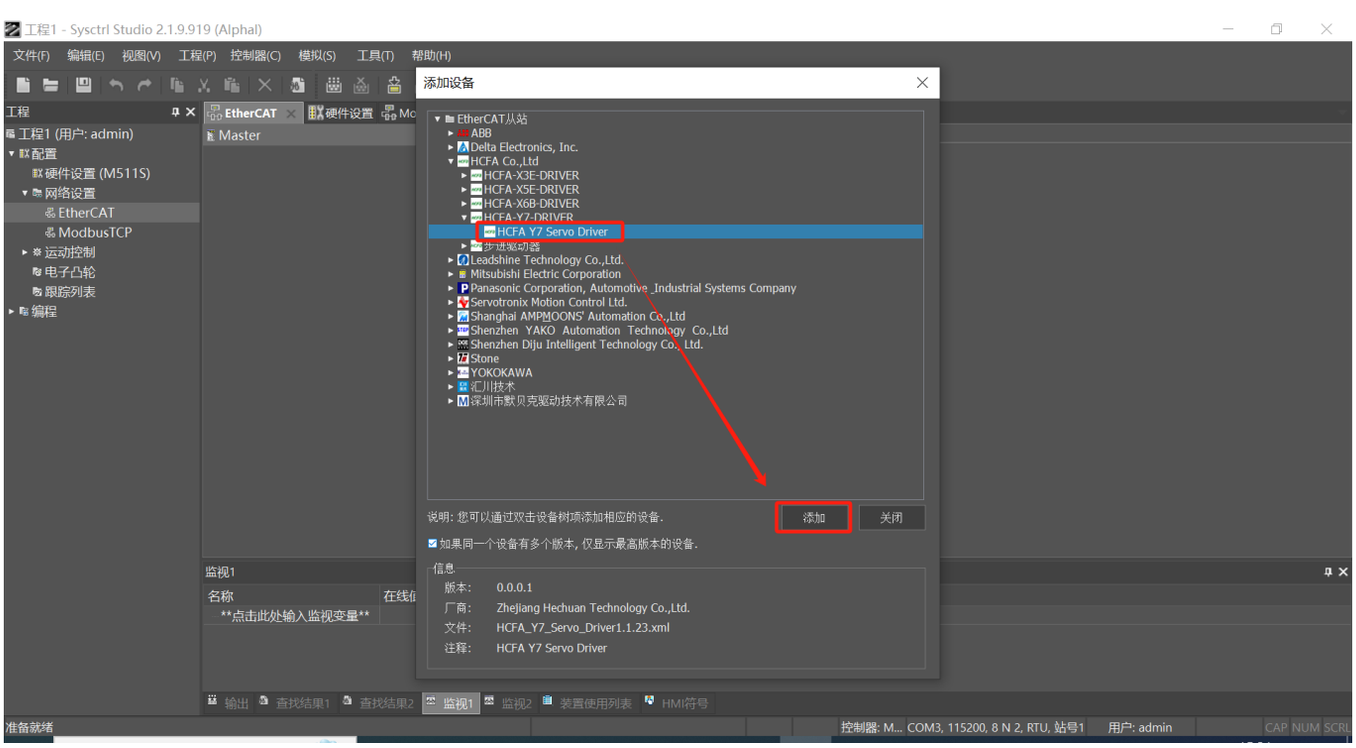
通过 ECAT_ReadParameter 指令和 ECAT_WriteParameter 指令对禾川 Y7 系列伺服驱动器参数 16#607D:02（索引：子索引）进行读取和写入。

- **软件配置**

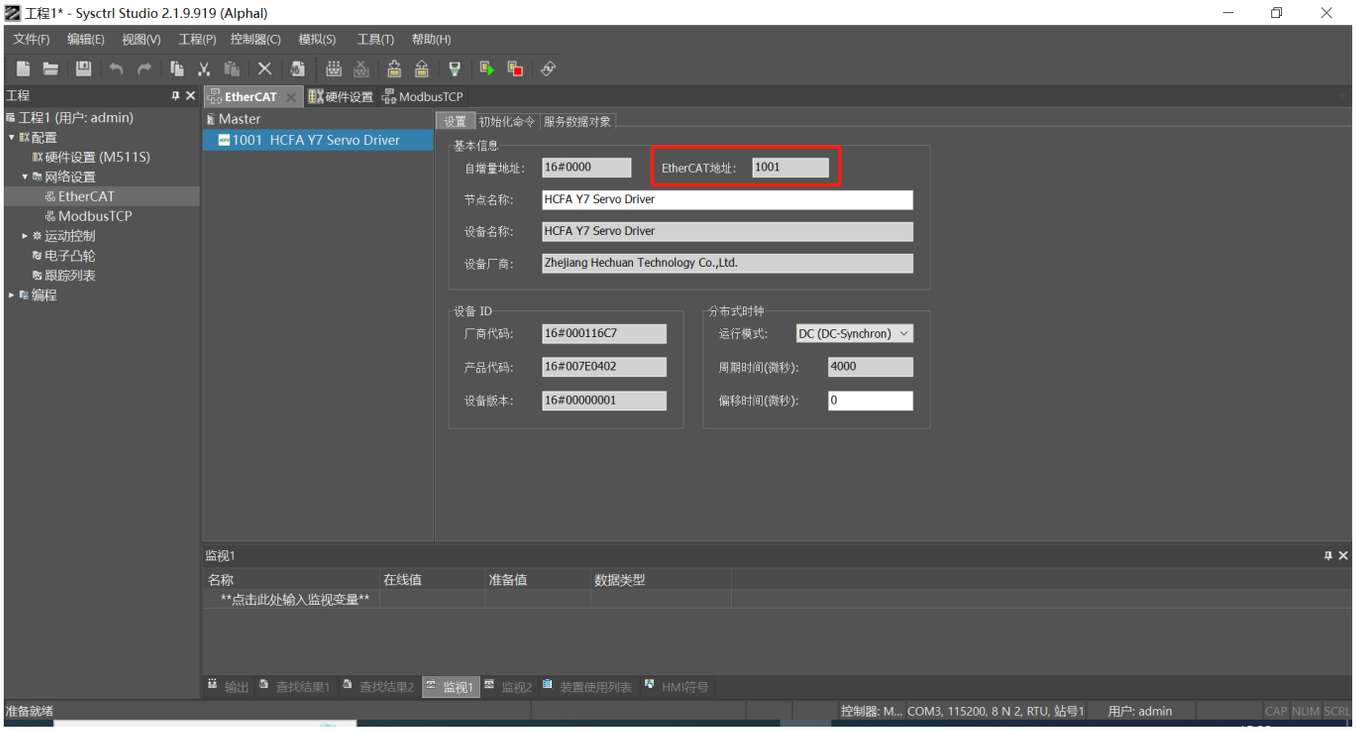
步骤一：单击“网络设置”，然后双击“EtherCAT”，单击 Master 后右击，单击添加从站。



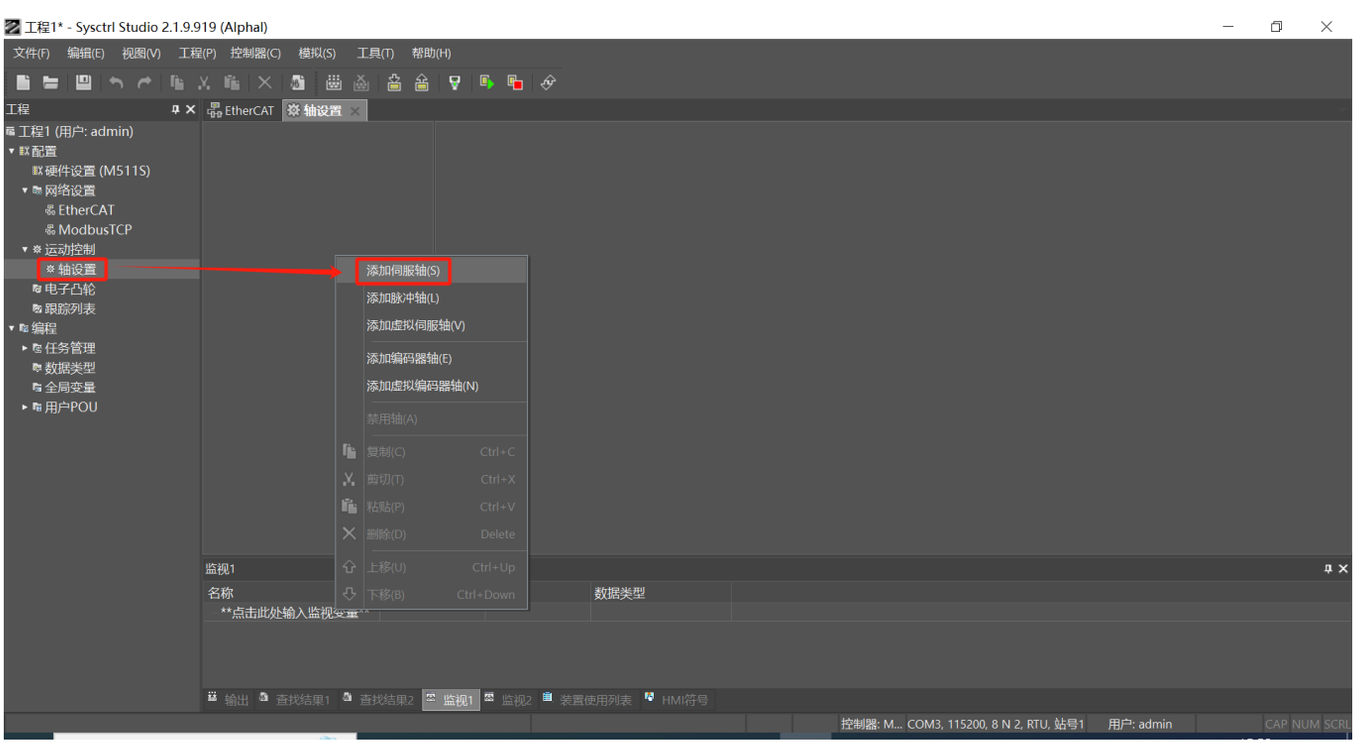
步骤二：双击 HCFA Y7 Servo Driver 添加从站，或者单击选择 HCFA Y7 Servo Driver 后，单击下图红色方框处添加按钮添加从站。



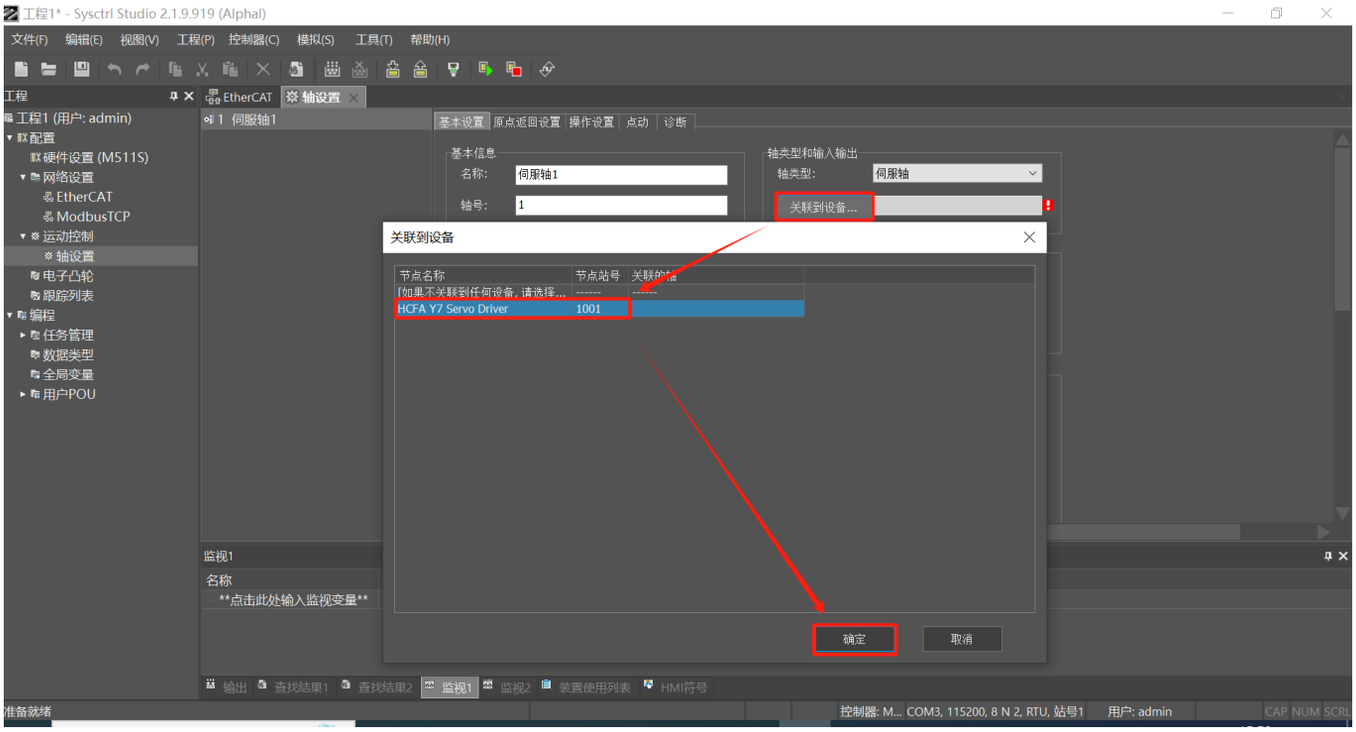
步骤三：第一台 EtherCAT 从站的 EtherCAT 地址为 1001，第二台 EtherCAT 从站的 EtherCAT 地址为 1002，以此类推。



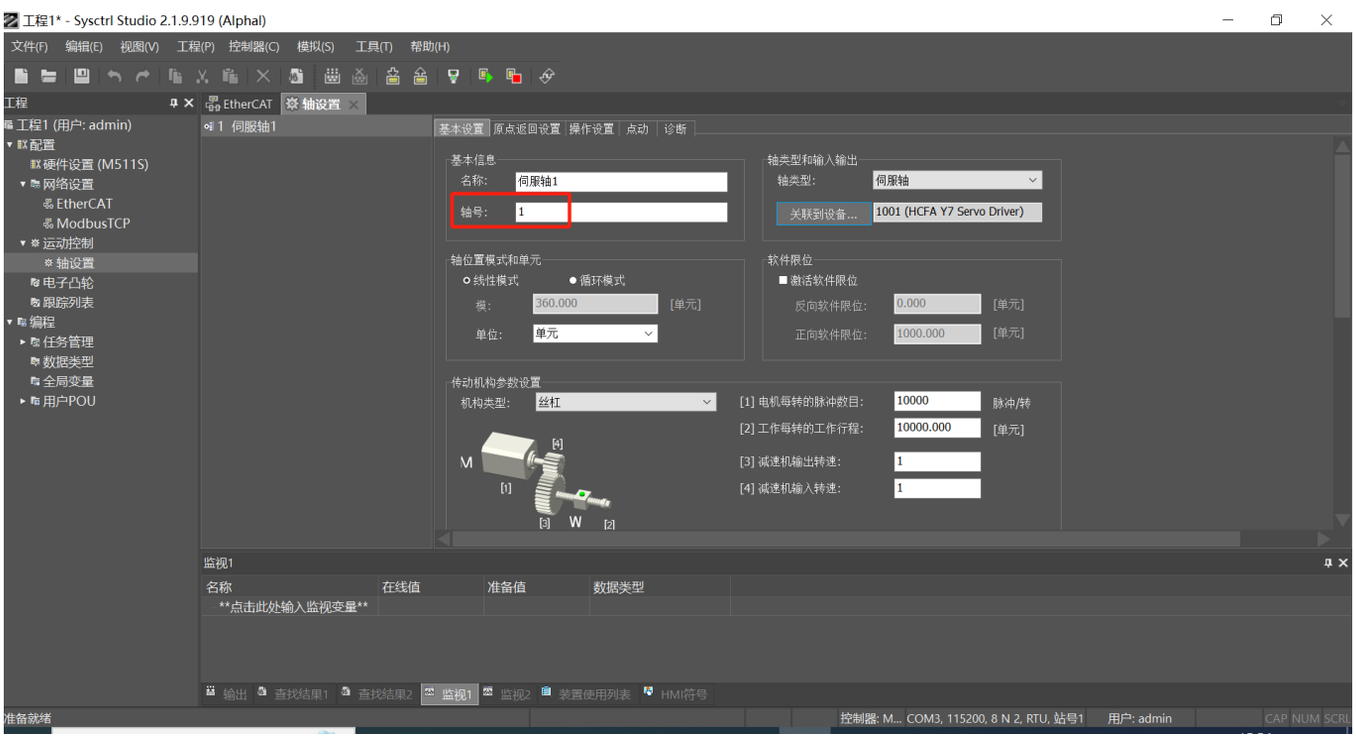
步骤四：双击轴设置 - 右击中间空白地方 - 单击添加伺服轴



步骤五：单击“关联到设备”-> 单击“HCFA Y7 Servo Driver” -> 单击“确定”按钮。通过此步操作，轴和 EhterCAT 从站建立一一对应关系。



步骤六：在下图红色方框处设置轴号。



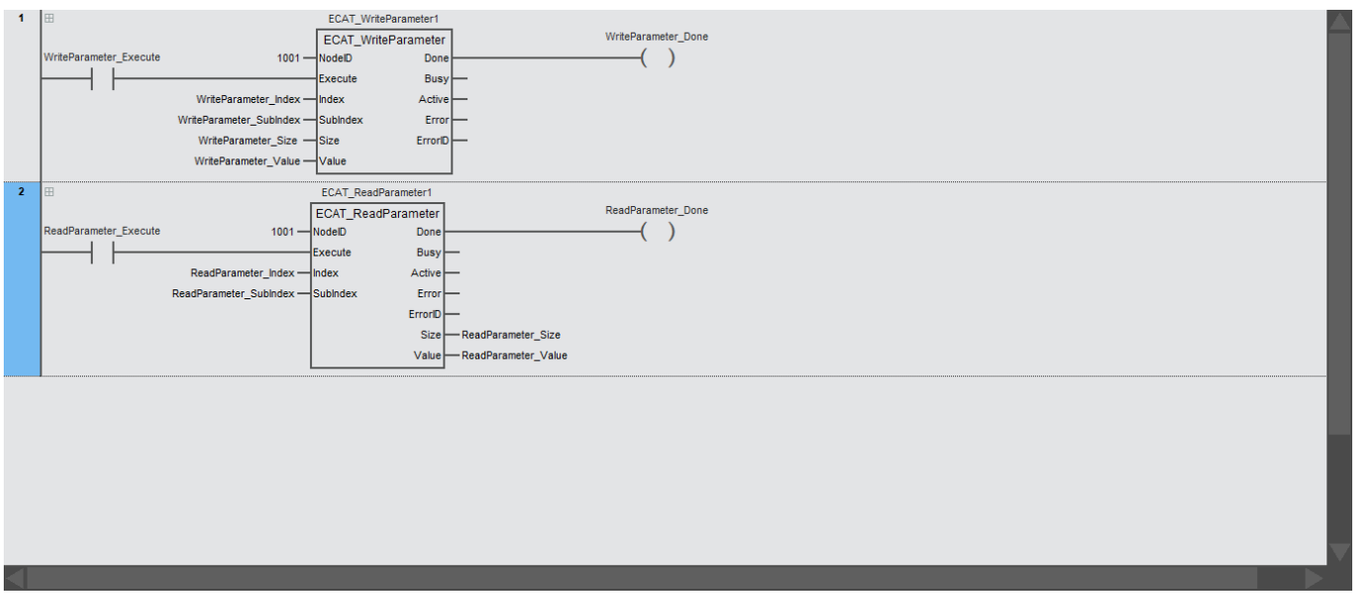
- **指令配置**

变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	ECAT_WriteParameter1		ECAT _WriteParameter		
VAR	ECAT _ReadParameter1		ECAT _ReadParameter		
VAR	WriteParameter_Execute		BOOL		
VAR	ReadParameter_Execute		BOOL		
VAR	WriteParameter_Done		BOOL		

VAR	ReadParameter_Done		BOOL		
VAR	WriteParameter_Value		UDINT		
VAR	ReadParameter _Value		UDINT		
VAR	WriteParameter_Index		UINT	16#607D	
VAR	WriteParameter_SubIndex		UINT	2	
VAR	ReadParameter _Index		USINT	16#607D	
VAR	ReadParameter_SubIndex		USINT	2	
VAR	WriteParameter_Size		USINT		
VAR	ReadParameter_Size		USINT		

梯形图 LD:



结构化文本 ST:

```
ECAT_WriteParameter1
(NodeID:=1001 ,
    Execute:=WriteParameter_Execute ,
    Index:=WriteParameter_Index ,
    SubIndex:=WriteParameter_SubIndex ,
    Size:=WriteParameter_Size ,
    Value:=WriteParameter_Value,
    Done=>WriteParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=>
    );

ECAT_ReadParameter1
(NodeID:=1001 ,
    Execute:=ReadParameter_Execute ,
    Index:=ReadParameter_Index ,
    SubIndex:=ReadParameter_SubIndex ,
    Done=>ReadParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    Size=>ReadParameter_Size ,
    Value=>ReadParameter_Value
    );
```

- **程序流程说明**

步骤一：设置输入变量 NodeID 的值为 1001，表示对主站连接的第 1 台 EtherCAT 从站进行写参数。

步骤二：设置输入变量 WriteParameter_Index 的值为 16#607D，输入变量 WriteParameter_SubIndex 的值为 2，设置输入变量 WriteParameter_Size 的值为 4，输入变量 WriteParameter_Value 的值为 2147483647。

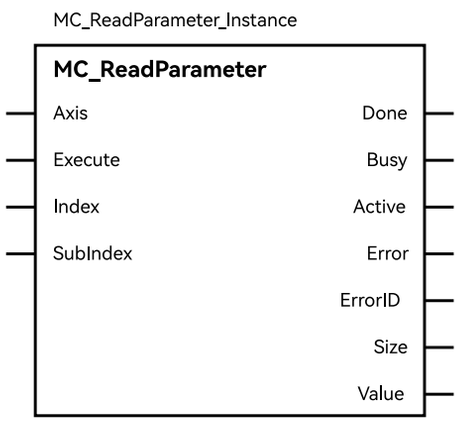
步骤三：输入变量 WriteParameter_Execute 由 FALSE 变为 TRUE 后，触发 ECAT_WriteParameter 指令执行，WriteParameter_Done 位变为 TRUE 时表示参数写入完成。

步骤四：设置输入变量 ReadParameter_Index 的值为 16#607D，输入变量 ReadParameter_SubIndex 的值为 2。

步骤五：输入变量 ReadParameter_Execute 由 FALSE 变为 TRUE 后，触发 ECAT_ReadParameter 指令执行，ReadParameter_Done 位变为 TRUE 时表示参数读取完成，读取完成后，ReadParameter_Size 和 ReadParameter_Value 的值分别为 4 和 2147483647，ReadParameter_Value 的值为读取的参数值。

4.6 MC_ ReadParameter（伺服轴服务数据参数读取指令）

通过 SDO（服务数据对象）读取伺服轴的参数值。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
MC_ReadPar-ameter	EtherCAT 从站参数读取	FB		MC_ReadParameter_Instance(Axis:= 参数 , Execute:= 参数 , Index:= 参数 , SubIndex:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Error=> 参数 , ErrorID=> 参数 , Size=> 参数 , Value=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Axis	轴号	USINT	参考通讯指令规格	不可缺省	指定控制轴的轴号。
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
Index	参数的索引	UINT	0~65535	0	设定欲读取参数的索引。
SubIndex	参数的子索引	USINT	0~255	0	设定欲读取参数的子索引。

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令控制中	BOOL	TRUE / FALSE	指令控制中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1
Size	参数的数据类型	USINT		欲读取参数的类型。 1：Byte（1 字节）； 2：Word（2 字节）； 4：DWord（4 字节）。
Value	读取的参数值	UDINT		读取到的参数值。

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Active	指令控制读参数时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。

Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于读取指定伺服轴的参数值，欲读取的参数通过 Index 和 SubIndex 进行指定。从站参数的 Index 和 SubIndex 可通过从站相关资料获得。

该指令在 Execute 由 FALSE 变为 TRUE 时，按照输入变量设定的值读取从站的参数。

该指令执行时，将指定从站参数的值读取到输出变量 Value 内。输出变量 Value 的数据类型为 UDINT，当 Value 的数据类型和读取从站参数的数据类型不一致时，可通过数据类型转换指令将 Value 变量的数据类型转换为从站参数的数据类型。如读取从站参数的数据类型为 DINT，可以通过 UDINT_TO_DINT 指令将 Value 的值转换到 DINT 类型的变量内；如读取从站参数的数据类型为 INT，可以通过 UDINT_TO_INT 指令将 Value 的值转换到 INT 类型的变量内。

• 指令完成时机

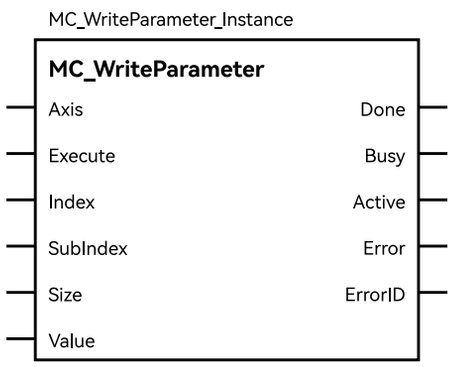
当读取到从站的参数值后，该指令完成，Done 位由 FALSE 变为 TRUE。

• 重启该指令

当指令执行完成后，Execute 再次由 FALSE 变为 TRUE 时，该指令可以重新执行；当指令正在执行中，Execute 再次由 FALSE 变为 TRUE 时，对指令的执行不会产生影响，指令仍按照未执行完成的输入变量执行指令。

4.7 MC_ WriteParameter（伺服轴服务数据参数设置指令）

通过 SDO（服务数据对象）设置伺服轴的参数值。所属库：Communications

指令	名称	FB/FUN	梯形图样式	ST样式
MC_WritePa-rameter	EtherCAT 从站参数设置	FB		MC_WriteParameter_Instance(Axis:= 参数 , Execute:= 参数 , Index:= 参数 , SubIndex:= 参数 , Size:= 参数 , Value:= 参数 , Done=> 参数 , Busy=> 参数 , Active=> 参数 , Error=> 参数 , ErrorID=> 参数);

◆ 输入变量

输入变量	名称	数据类型	设定范围	缺省值	说明
Axis	轴号	USINT	参考通讯指令规格	不可缺省	指定控制轴的轴号。
Execute	启动	BOOL	TRUE 或 FALSE	FALSE	检测到该参数的上升沿时执行该指令。
Index	参数的索引	UINT	0~65535	0	设定欲设置参数的索引。
SubIndex	参数的子索引	USINT	0~255	0	设定欲设置参数的子索引。
Size	参数的数据类型	USINT	1~4	不可缺省	欲设置参数的类型。 1：Byte（1 字节）； 2：Word（2 字节）； 4：DWord（4 字节）。
Value	设定值	UDINT	0 ~ 4294967295	0	设定值

◆ 输出变量

输出变量	名称	数据类型	输出范围	功能
Done	执行完成	BOOL	TRUE / FALSE	指令执行完成时变为 TRUE。
Busy	指令执行中	BOOL	TRUE / FALSE	指令正常执行中为 TRUE。
Active	指令控制中	BOOL	TRUE / FALSE	指令控制中为 TRUE。
Error	执行错误	BOOL	TRUE / FALSE	指令执行发生异常时变为 TRUE。
ErrorID	错误代码	WORD	0~65535	指令执行异常时，输出错误代码。*1

***1: 指令 ErrorID 值的含义请参阅“指令错误代码描述”。**

◆ 输出变量刷新时机

名称	变为TRUE的时机	变为FALSE的时机
Done	指令执行完成时。	Done 为 TRUE，Execute 从 TRUE 变为 FALSE 时。 指令执行且 Execute 为 FALSE 时，Done 变为 TRUE 一个周期后，变为 FALSE。
Busy	检测到 Execute 的上升沿时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。
Active	指令控制写参数时。	Done 从 FALSE 变为 TRUE 时。 Error 从 FALSE 变为 TRUE 时。

Error	指令输入变量值不在允许的范围、不满足指令的执行条件或者指令执行过程中遇到异常。	Error 为 TRUE, Execute 从 TRUE 变为 FALSE 时。 指令已执行且 Execute 变为 FALSE，该指令执行过中遇到异常时，Error 变为 TRUE，一个周期后，变为 FALSE。

◆ 功能说明

• 基本功能说明

该指令用于设置指定伺服轴的参数值，从站的参数通过 Index 和 SubIndex 进行指定。从站参数的 Index 和 SubIndex 可通过从站相关资料获得。设置的参数值，为输入变量 Value 的值。

该指令在 Execute 由 FALSE 变为 TRUE 时，按照输入变量设定的值设置从站的参数。

该指令执行时，将输入变量 Value 中的值设置到指定的从站参数中。输入变量 Value 的数据类型为 UDINT，当 Value 的数据类型和从站参数的数据类型不一致时，可通过数据类型转换指令将 Value 变量的数据类型转换为从站参数的数据类型。如从站参数的数据类型为 DINT，可以通过 UDINT_TO_DINT 指令将 Value 的值转换到 DINT 类型的变量内；如从站参数的数据类型为 INT，可以通过 UDINT_TO_INT 指令将 Value 的值转换到 INT 类型的变量内。

• 指令完成时机

当发出写参数值的命令后，该指令完成，Done 位由 FALSE 变为 TRUE。

• 重启该指令

当指令执行完成后，Execute 再次由 FALSE 变为 TRUE 时，该指令可以重新执行；当指令正在执行中，Execute 再次由 FALSE 变为 TRUE 时，对指令的执行不会产生影响，指令仍按照未执行完成的输入变量执行指令。

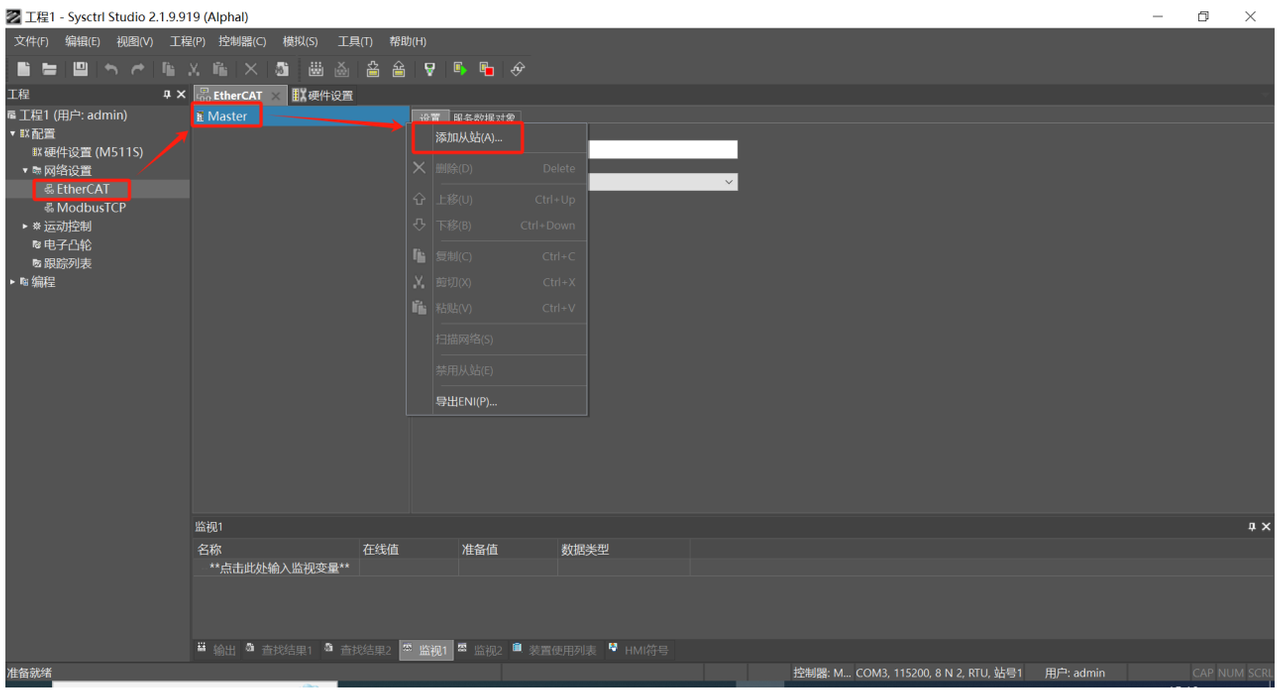
4.8 MC参数读写使用范例

- **目标需求**

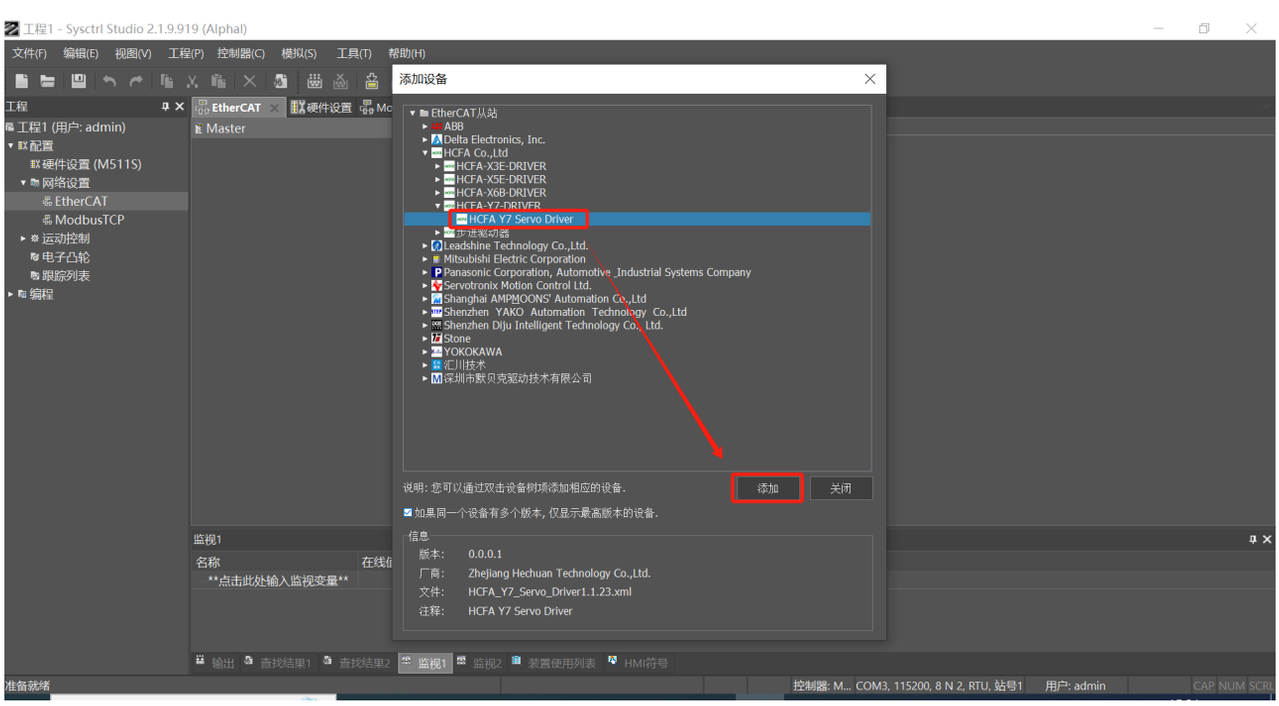
通过 MC_ ReadParameter 指令和 MC_ WriteParameter 指令对禾川 Y7 系列伺服驱动器参数 16#607D:02（索引：子索引）进行读取和写入。

- **软件配置**

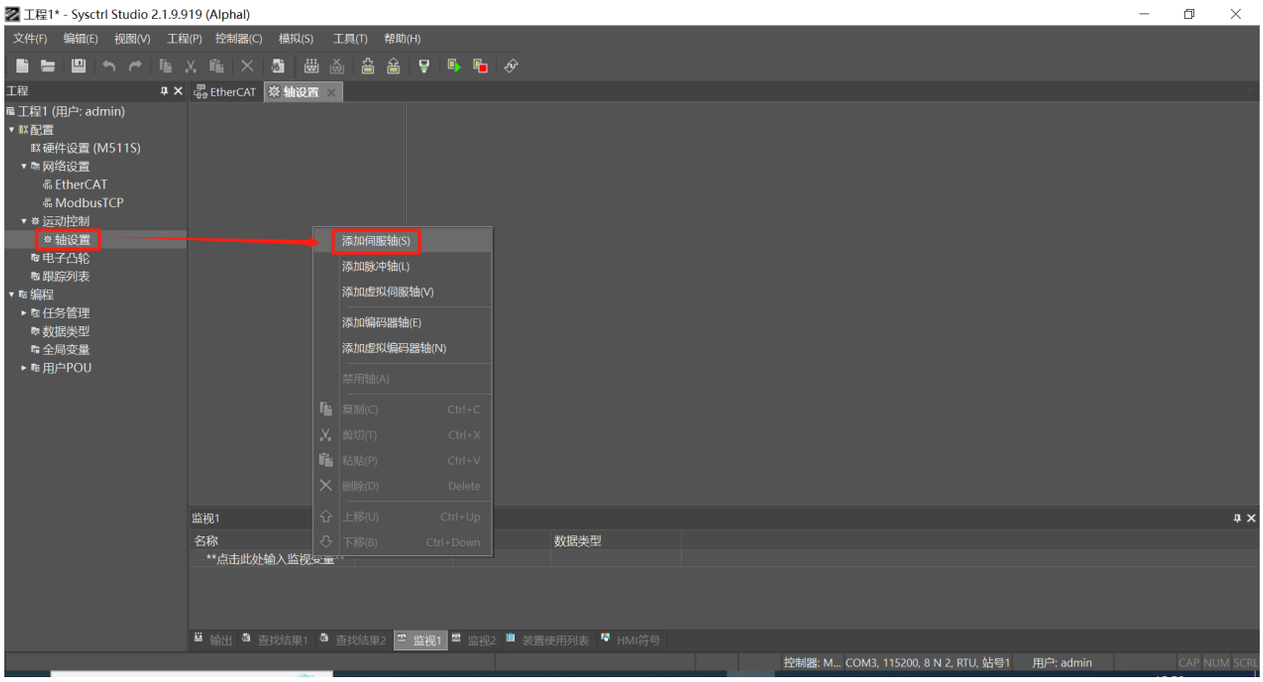
步骤一：单击“网络设置”，然后双击“EtherCAT”，单击 Master 后右击，单击添加从站。



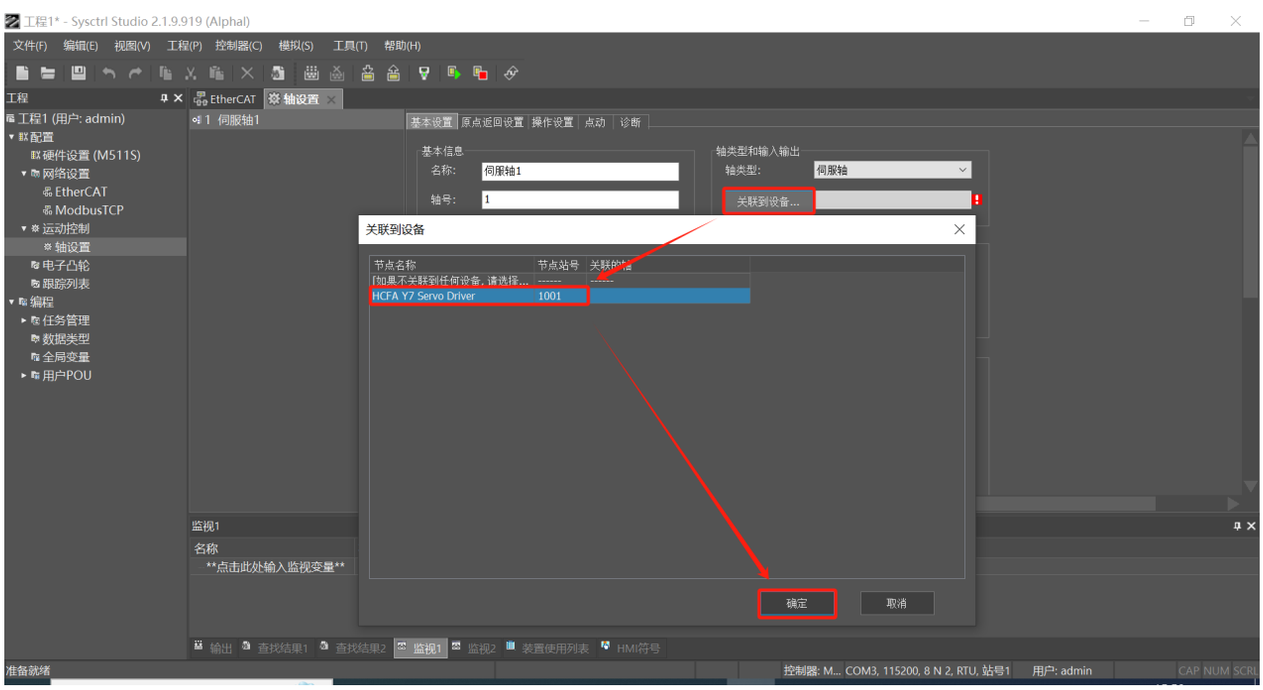
步骤二: 双击 HCFA Y7 Servo Driver 添加从站 , 或者单击选择 HCFA Y7 Servo Driver 后, 单击下图红色方框处“添加”按钮添加从站。



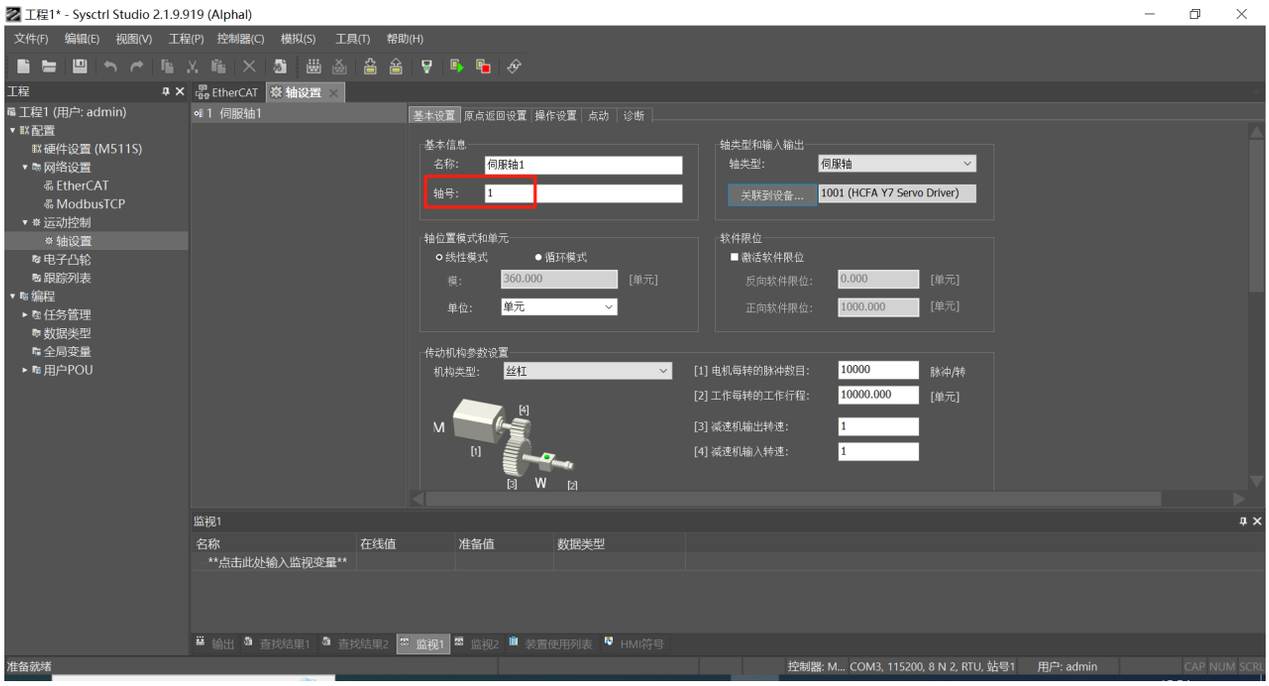
步骤三：双击轴设置 - 右击中间空白地方 - 单击添加伺服轴



步骤四：单击“关联到设备”-> 单击“HCFA Y7 Servo Driver” -> 单击“确定”按钮。通过此步操作，轴和 EhterCAT 从站建立一一对应关系。



步骤五：在下图红色方框处设置轴号。

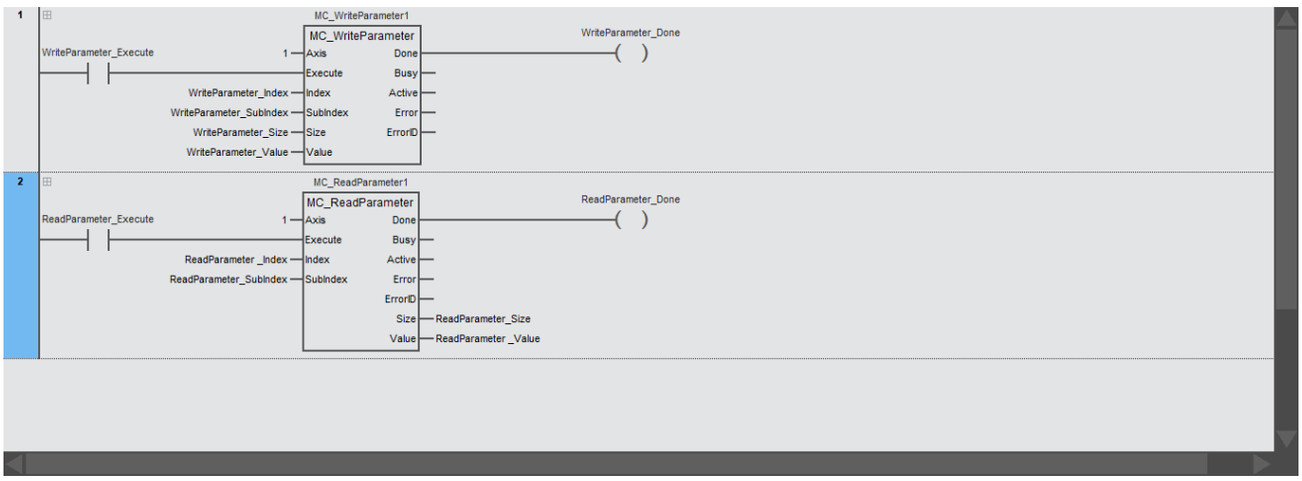


- **指令配置**

变量表

类别	名称	分配到	数据类型	初始值	注释
VAR	MC_WriteParameter1		MC_WriteParameter		
VAR	MC_ReadParameter1		MC_ReadParameter		
VAR	WriteParameter_Execute		BOOL		
VAR	ReadParameter_Execute		BOOL		
VAR	WriteParameter_Done		BOOL		
VAR	ReadParameter_Done		BOOL		
VAR	WriteParameter_Value		UDINT		
VAR	ReadParameter _Value		UDINT		
VAR	WriteParameter_Index		UINT	16#607D	
VAR	WriteParameter_SubIndex		UINT	2	
VAR	ReadParameter _Index		USINT	16#607D	
VAR	ReadParameter_SubIndex		USINT	2	
VAR	WriteParameter_Size		USINT		
VAR	ReadParameter_Size		USINT		

梯形图 LD:



结构化文本 ST:

```
MC_WriteParametel(Axis:=1 ,
    Execute:=WriteParameter_Execute ,
    Index:=WriteParameter_Index ,
    SubIndex:=WriteParameter_SubIndex ,
    Size:=WriteParameter_Size ,
    Value:=WriteParameter_Value ,
    Done=>WriteParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=>
    );

MC_ReadParameterl(Axis:=1 ,
    Execute:=ReadParameter_Execute ,
    Index:=ReadParameter_Index ,
    SubIndex:=ReadParameter_SubIndex ,
    Done=>ReadParameter_Done ,
    Busy=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    Size=>ReadParameter_Size ,
    Value=>ReadParameter_Value
    );
```

- **程序流程说明**

步骤一：设置输入变量 Axis 的值为 1，表示对轴号为 1 的轴进行写参数。

步骤二：设置输入变量 WriteParameter_Index 的值为 16#607D，输入变量 WriteParameter_SubIndex 的值为 2，输入变量 WriteParameter_Size 的值为 4，输入变量 WriteParameter_Value 写入 2147483647。

步骤三：输入变量 WriteParameter_Execute 由 FALSE 变为 TRUE 后，触发 WriteParameter 指令执行，WriteParameter_Done 位变为 TRUE 时表示参数写入完成。

步骤四：设置输入变量 ReadParameter_Index 的值为 16#607D，输入变量 ReadParameter_SubIndex 的值为 2。

步骤五：输入变量 ReadParameter_Execute 由 FALSE 变为 TRUE 后，触发 ReadParameter 指令执行，ReadParameter_Done 位变为 TRUE 时表示参数读取完成，读取完成后，ReadParameter_Size 和 ReadParameter_Value 的值分别为 4 和 2147483647，ReadParameter_Value 的值为读取的参数值。

4.9 通讯指令规格

机种名称	数据交换通道编号（Linknum）范围	Socket编号（SocketNum）范围	CANopen从站站号范围(NodeID)	EhterCAT从站站号范围(NodeID)	轴号范围
HCM511S-32MT4-D	1~4	1~4	1~16	1001~1008	1~16
HCM501S-16MT4-D	不支持	不支持	不支持	1001~1008	1~16
HCM301-16MT4-D	不支持	不支持	不支持	不支持	1~16
HCM302-16MT4-D	不支持	不支持	不支持	不支持	1~16
HCM310-16MT4-D	1~16	1~8	不支持	不支持	1~16
HCM311-16MT4-D	1~16	1~8	不支持	不支持	1~16
HCM312-32MT6-D	1~16	1~8	1~32	不支持	1~16
HCM511-32MT4-D	1~16	1~8	1~32	1001~1016	1~64
HCM512-32MT4-D	1~16	1~8	1~32	1001~1032	1~64
HCM513-32MT4-D	1~16	1~8	1~32	1001~1064	1~64
HCM514-32MT4-D	1~16	1~8	1~32	1001~1128	1~128

第 5 章　Modbus和Modbus TCP通讯相关

5.1 支持的Modbus功能码

M 系列控制器以太网、RS485、RS232、USB 通讯口支持的功能码如下表所示：

种类	功能码	说明	可否广播	读/写最大值	可操作装置
位装置	0x01	定义：读位装置的值。 M 系列控制器位装置的值都可以使用 01 功能码读取。	否	256 个	%IX,%QX
	0x02	定义：读输入位装置的值。 M 系列控制器位装置的值都可以使用 02 功能码读取。	否	256 个	%IX,%QX
	0x05	写单个位装置的值。	是	1 个	%QX
	0x0F	写多个位装置的值。	是	256 个	%QX
字装置	0x03	读单个或多个字装置的值。	否	100 个	%MW,%QW,%IW
	0x04	定义：读单个或多个输入字装置的值。 M 系列控制器字装置的值都可以使用 04 功能码读取。	否	100 个	%MW,%QW,%IW
	0x06	写单个字装置的值。	是	1 个	%MW,%QW
	0x10	写多个字装置的值。	是	100 个	%MW,%QW
	0x17	读写单个或多个字装置的值。	是	100 个	%MW,%QW, %IW（仅读）

5.2 Modbus异常回应码

M 系列控制器以太网、RS485、RS232、USB 通讯口支持的异常回应码如下表所示：

异常回应码	含义	处理方法
1	从站不支持主站指定的功能码	指定从站支持的功能码
2	从站不支持主站指定的 Modbus 地址	指定从站支持的 Modbus 地址
3	读或者写指定的数据长度超出范围	控制器做从站时，操作字（WORD）装置时，一次可以读或者写的最大长度为 100 个 WORD；操作位（bit）装置时，一次可以读或者写的最大长度为 256 个位；超过上述规格，控制器回复该异常回应码。
7	主站和从站计算的校验码不同	确认主站和从站的波特率及通讯格式相同。 检查总线附件是否有干扰源。 检查总线是否为屏蔽线。 检查主站和从站是否都有接地。

5.3 装置对应Modbus地址列表

下表所示为 M 系列控制器支持 Modbus 地址，这些地址可以通过以太网、RS232、RS485 通讯访问，这些地址可以通过对应的功能访问，支持的功能码 :03、04、06、16#10、16#17、01、02、05、16#0F。用户需人机界面等对控制器的位装置进行读写时，可以使用输出装置的位装置作为中间位装置，如可以使用 %QX50.0~%QX127.7 作为中间位装置，没有控制输出点的输出装置，都可以作为中间位装置。

装置名称	装置类型	装置编号	装置地址（hex）	装置属性
I（输入装置）	位装置（bit）	%IX0.0~%IX0.7	6000 ~ 6007	只读
		%IX1.0~%IX1.7	6008 ~ 600F	只读
		.....	.....	只读
		%IX127.0~%IX127.7	63F8 ~ 63FF	只读
	字装置（word）	%IW0~%IW63	8000 ~ 803F	只读
Q（输出装置）	位装置（bit）	%QX0.0~%QX0.7	A000 ~ A007	读 / 写
		%QX1.0~%QX1.7	A008 ~ A00F	读 / 写
		.....	.....	读 / 写
		%QX127.0~%QX127.7	A3F8 ~ A3FF	读 / 写
	字装置（word）	%QW0~%QW63	A000 ~ A03F	读 / 写
M（中间装置）	字装置（word）	%MW0~%MW32767	0000 ~ 7FFF	读 / 写

注意：%MW0~%MW999 地址在没有绑定变量的情况下默认为停电保持。

QX 或者 IX 相关的位装置对应 MODBUS 地址的转换方法如下 :

如 QXA.B，转换规则为 A*8+B 转换为 16 进制 +0xA000

如 %QX50.1 对应的 Modbus 地址为 0xA191，计算方法为 50*8+1=401=0x191

0x191+0xA000=0xA191

第 6 章　Modbus通讯协议说明

6.1 ASCII模式报文结构

RTU 模式下 Modbus 协议的一帧报文的基本格式如下表所示。

起始字符	从站站号	功能码	装置首地址	数据	LRC校验和	结束字符
1 个字符	2 个字符	2 个字符	4 个字符	n 个字符	2 个字符	2 个字符

一帧数据的数据结构

内容	说明
起始字符	起始字符为冒号“ : ”, 对应的 ASCII 码为 16#3A
从站站号	指定需要访问从站的站号。范围：1~255（16#1~26#FF）
功能码	指定对从站读、写或者读写。支持的功能码 :03、04、06、16#10、16#17、01、02、05、16#0F 。
装置首地址	指定对从站读或者写装置首地址。
数据	指定对从站读、写或者读写的相关数据。
LRC 校验和	以“从站站号”开始，至“数据”最后一个字节内容，按照 LRC 校验规则计算 LRC 校验和，LRC 校验和计算方法请参考其他书籍。
结束字符	结束字符 1 （CR）= 16#0D 结束字符 0（LF）=16#0A

字符与 ASCII 码对应关系如下表所示:

字符	“0”	“1”	“2”	“3”	“4”	“5”	“6”	“7”
ASCII码	16#30	16#31	16#32	16#33	16#34	16#35	16#36	16#37
字符	“8”	“9”	“A”	“B”	“C”	“D”	“E”	“F”
ASCII码	16#38	16#39	16#41	16#42	16#43	16#44	16#45	16#46

◆ 功能码及数据

数据的格式取决于功能码。例如，读取控制器以 16#1000 为起始地址的连续 2 个地址的数据 , 控制器的通讯站号为 1。16#1000 为控制器 %MW4096 装置对应的 Modbus 地址。总线数据含义解释如下:

主站→从站：3A 30 31 30 33 31 30 30 30 30 30 32 45 41 0D 0A

从站→主站：3A 30 31 30 33 30 34 30 30 30 31 30 30 30 32 46 35 0D 0A

• 请求信息:

内容	字符	ASCII码
起始字符	“：”	3A
通讯站号：01	“0”	30
	“1”	31
功能码：03	“0”	30
	“3”	33
起始装置地址：16#1000	“1”	31
	“0”	30
	“0”	30
	“0”	30
数据个数（word）：2	“0”	30
	“0”	30
	“0”	30
	“2”	32
LRC 校验码：16#EA	“E”	45
	“A”	41
结束字符 1	CR	0D
结束字符 0	LF	0A

• 回应信息:

内容	字符	ASCII码
起始字符	“：”	3A
通讯站号：01	“0”	30
	“1”	31
功能码：03	“0”	30
	“3”	33
数据个数（Byte）	“0”	30
	“4”	34
16#1000 地址的值	“0”	30
	“0”	30
	“0”	30
	“1”	31
16#1001 地址的值	“0”	30
	“0”	30
	“0”	30
	“2”	32
LRC 校验码 : 16#F5	“F”	46
	“5”	35
结束字符 1	CR	0D
结束字符 0	LF	0A

6.2 RTU模式报文结构

RTU 模式下 Modbus 协议一帧报文的基本格式如下表所示。

起始字符	从站站号	功能码	装置首地址	数据	LRC校验和	结束字符
1 个字符	2 个字符	2 个字符	4 个字符	n 个字符	2 个字符	2 个字符

内容	说明
开始	保持总线无数据≥ 10ms。
从站站号	指定需要访问从站的站号。范围：1~255（16#1~26#FF）。
功能码	指定对从站读、写或者读写。支持的功能码 :03、04、06、16#10、16#17、01、02、05、16#0F 。
装置首地址	指定对从站读或者写装置首地址。
数据	指定对从站读、写或者读写的相关数据。
CRC 校验和	以“从站站号”开始，至“数据”最后一个字节内容，按照 CRC 校验规则计算 CRC 校验和，CRC 校验和计算方法请参考其他书籍。
结束	保持总线无数据≥ 10ms。

◆ 功能码及数据

数据的格式取决于功能码。例如，读取控制器以 16#2000 为起始地址的连续 2 个地址的数据 , 控制器的通讯站号为 1。16#2000 为控制器 %MW8192 装置对应的 Modbus 地址。数据含义解释如下：

主站→从站：01 03 20 00 00 02 CF CB

从站→主站：01 03 04 00 01 00 02 2A 32

• 请求信息：

开始	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	字装置数量高8位	字装置数量低8位	CRC校验和高8位	CRC校验和低8位	结束
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	
总线无数据≥ 10 ms	01	03	20	00	00	02	CF	CB	无输入≥ 10 ms

• 回应信息：

开始	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	字装置数量高8位	字装置数量低8位	CRC校验和高8位	CRC校验和低8位	结束
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	
总线无数据≥ 10 ms	01	03	20	00	00	02	CF	CB	无输入≥ 10 ms

6.3 Modbus功能码介绍

◆ 功能码：03, 读取单个或多个字装置寄存器的值

• 请求信息数据结构：

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	字装置数量高8位	字装置数量低8位	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

备注：寄存器数量单位为 Word

• 回应信息数据结构：

数据含义	从站站号	功能码	数据个数	数据值		数据值	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	……	Byte n	Byte n+1	Byte n+2	Byte n+3

备注：数据个数单位为 Byte

• 异常回应信息数据结构：

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例 :

通过 03 功能码读取控制器 16#A000、16#A001 地址的内容值 ,16#A000、16#A001 为控制器内部 %QW0 、%QW1 的 Modbus 地址，假设 %QW0 的值为 16#1234 、%QW1 的值为 16#5678。

• 请求信息：

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	字装置数量高8位	字装置数量低8位	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	03	A0	00	00	02	E6	0B

• 回应信息：

数据含义	从站站号	功能码	数据个数	数据值		数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte 9	Byte 10
16 进制数值	01	03	04	12	34	56	78	81	07

◆ 功能码：04, 读取单个或多个字装置寄存器的值

• 请求信息数据结构：

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	字装置数量高8位	字装置数量低8位	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

备注：寄存器数量单位为 Word

• **回应信息数据结构：**

数据含义	从站站号	功能码	数据个数	数据值		数据值	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	……	Byte n	Byte n+1	Byte n+2	Byte n+3

备注：数据个数单位为 Byte

• **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例 :

通过 04 功能码读取控制器 16#8000、16#8001 地址的内容值,16#8000、16#8001 为控制器内部 %IW0 、%IW1 的 Modbus 地址，假设 %IW0 的值为 16#1234 、%IW1 的值为 16#5678。

• **请求信息：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	字装置数量高8位	字装置数量低8位	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	04	80	00	00	02	58	0B

• **回应信息：**

数据含义	从站站号	功能码	数据个数	数据值		数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte 9	Byte 10
16 进制数值	01	04	04	12	34	56	78	80	B0

◆ **功能码：06, 写单个字装置寄存器的值**

• **请求信息数据结构：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **回应信息数据结构：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例:

通过 06 功能码将数值 16#1234 写入控制器 16#A000 地址内。

• **请求信息：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	06	A0	00	12	34	A6	BD

• **回应信息：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	06	A0	00	12	34	A6	BD

◆ 功能码：0x10, 写多个字装置寄存器的值

• **请求信息数据结构：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据个数（Word）		数据个数（Byte）	数据值		数据值	数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	……	Byte n	Byte n+1	Byte n+2	Byte n+3

• **回应信息数据结构：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据个数（Word）	CRC校验和高8位		CRC校验和低8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte7

• **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 16#10 功能码将 16#1234 、16#5678 写入控制器 16#A000 、16#A001 地址内，16#A000，16#A001 为控制器内部 %QW0 、%QW1 的 Modbus 地址。

• **请求信息：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据个数（Word）		数据个数（Byte）	数据值		数据值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte 9	Byte 10	Byte 11	Byte 12
16 进制数值	01	10	A0	00	00	02	04	12	34	56	78	70	9C

• **回应信息：**

数据含义	从站站号	功能码	字装置首地址高8位	字装置首地址低8位	数据个数（Word）		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	10	A0	00	00	02	63	C8

◆ 功能码：0x17, 读写单个或多个字装置的值

• 请求信息数据结构:

从站站号	功能码	读字装置首地址	读数据个数（Word）	写字装置首地址	写数据个数（word）		写数据个数（Byte）	写数据值		写数据值	写数据值	CRC校验和
1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	……	2Byte	2Byte	Byte n Byte n+1 Byte n+2	Byte n+3

• 回应信息数据结构:

从站站号	功能码	读数据个数（Byte）	读数据值	读数据值	读数据值	CRC校验和
1Byte	1Byte	1Byte	2Byte	……	2Byte	2Byte

• 异常回应信息数据结构:

从站站号	16#80+功能码	异常回应码	CRC校验和
1Byte	1Byte	1Byte	2Byte

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例:

通过 16#17 功能码将 16#1234 、16#0064 写入控制器 16#0000 、16#0001 地址内，16#0000，16#0001 为控制器内部 %MW0 、%MW1 的 Modbus 地址 , 将从站 16#8000、16#8001 地址内的值读出，16#8000、16#8001 为从站控制器内部 %IW0、%IW1 的 Modbus 地址。

• 请求信息:

从站站号	功能码	读字装置首地址	读数据个数（Word）	写字装置首地址	写数据个数（word）	写数据个数（Byte）	写数据值	写数据值	CRC校验和
1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	2Byte	2Byte
0x01	0x17	0x8000	0x0002	0x0000	0x0002	0x04	0x000C	0x0064	0x7DDC

• 回应信息:

从站站号	功能码	读数据个数（Byte）	读数据值	读数据值	CRC校验和
1Byte	1Byte	1Byte	2Byte	2Byte	2Byte
0x01	0x17	0x04	0x0000	0x0000	0XF927

◆ 功能码：01, 读位装置寄存器的值

• 请求信息数据结构:

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

备注：数据个数单位为 Bit

• 回应信息数据结构:

数据含义	从站站号	功能码	数据个数（Byte）	位装置值	位装置值	位装置值	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

备注：回应信息中数据个数（Byte2）的值由请求信息中的位装置个数（Byte4 和 Byte5）的值来决定。如请求信息中读取位装置的个数为 m，m 除以 8 的商为 n，如果可以整除，回应信息中位装置个数（Byte4 和 Byte5）的值为 n；反之回应信息中位装置个数（Byte4 和 Byte5）的值为 n + 1，详细请参考如下范例。

• **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 01 功能码读取控制器 %QX0.0~%QX1.6 的状态值，%QX0.0 的地址为 16#A000，假设 %QX0.7~%QX0.0= 1000 0001，%QX1.6~%QX1.0=101 0001。

• **请求信息：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	01	A0	00	00	0F	5E	0E

• **回应信息：**

数据含义	从站站号	功能码	数据个数（Byte）	位装置值	位装置值	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
16 进制数值	01	01	02	81	51	18	50

◆ 功能码：02, 读位装置寄存器的值

• **请求信息数据结构：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **回应信息数据结构：**

数据含义	从站站号	功能码	数据个数（Byte）	位装置值	位装置值	位装置值	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

备注：回应信息中数据个数（Byte2）的值由请求信息中的位装置个数（Byte4 和 Byte5）的值来决定。如请求信息中读取位装置的个数为 m，m 除以 8 的商为 n，如果可以整除，回应信息中位装置个数（Byte4 和 Byte5）的值为 n；反之回应信息中位装置个数（Byte4 和 Byte5）的值为 n + 1，详细请参考如下范例。

• **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 02 功能码读取控制器 %IX0.0~%IX1.6 的状态值，%IX0.0 的地址为 16#6000，假设 %IX0.7~%IX0.0= 1000 0001，%IX1.6~%IX1.0=101 0001。

• **请求信息：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	02	60	00	00	0F	26	0E

• **回应信息：**

数据含义	从站站号	功能码	数据个数（Byte）	位装置值	位装置值	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
16 进制数值	01	02	02	81	51	18	14

◆ 功能码：05, 设置单个位装置寄存器的值

• **请求信息数据结构：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

备注：位装置值为 0 时表示将 FALSE 写入位装置，位装置值为 16#FF00 时表示将 TRUE 写入位装置。

• **回应信息数据结构：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

• **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 05 功能码设置控制器 %QX0.7 的值为 TRUE，%QX0.7 的地址为 16#A007。

• **请求信息：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	05	A0	07	FF	00	1F	FB

• **回应信息：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置值		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	05	A0	07	FF	00	1F	FB

◆ 功能码：0x0F, 写多个位装置寄存器的值

• **请求信息数据结构：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数（Bit）		数据个数（Byte）	位装置值	位装置值	位装置值	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	……	Byten	Byte n+1	Byte n+2

备注：请求信息中有多少个 Byte 的数据由请求信息中欲写入位数值的个数决定。

- **回应信息数据结构：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数（Bit）		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7

- **异常回应信息数据结构：**

数据含义	从站站号	16#80+功能码	异常回应码	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 0F 功能码设置控制器 %QX0.7~%QX0.0=1000 0001，%QX1.7~%QX1.0=1010 0011,%QX0.0 的地址为 16#A000。

- **请求信息：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数（Bit）		数据个数（Byte）	位装置值	位装置值	CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte9	Byte 10	Byte 11
16 进制数值	01	0F	A0	00	00	10	02	81	A3	62	03

- **回应信息：**

数据含义	从站站号	功能码	位装置首地址高8位	位装置首地址低8位	位装置个数（Bit）		CRC校验和高8位	CRC校验和低8位
数据顺序	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
16 进制数值	01	0F	A0	00	00	10	76	07

第 7 章　Modbus TCP通讯协议说明

Modbus TCP 是运行在 TCP/IP 上的 Modbus 报文传输协议。Modbus TCP 的端口号为 502。ModbusTCP 协议一帧报文的基本格式如下表所示。

事务处理标识	协议标识符	数据长度	从站站号	功能码	装置地址	数据内容
2Byte	2Byte	2Byte	1Byte	1Byte	2Byte	n 个 Byte

内容	说明
事务处理标识符	Modbus 请求 / 响应事务处理的标识，可以理解为报文的序列号，一般每次通信之后就要加 1 以区别不同的通信数据报文。
协议标识符	00 00 表示 Modbus 协议。
数据长度	以“从站站号”开始，至“数据”最后一个字节内容的字节数，数据长度以字节为单位。
从站站号	指定需要访问从站的站号。范围：1~255（16#1~26#FF）。
功能码	指定对从站读、写或者读写。支持的功能码 :03、04、06、16#10、16#17、01、02、05、16#0F 。
装置地址	指定对从站读或者写装置首地址。
数据内容	指定对从站读或者写装置首地址。

7.1 Modbus功能码介绍

◆ 功能码：03, 读取单个或多个字装置寄存器的值

• 请求信息数据结构:

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• 回应信息数据结构:

事务和协议标识	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

• 异常回应信息数据结构:

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例 :

通过 03 功能码读取控制器 0x1000，0x1001 地址的内容值，0x1000 、0x1001 为控制器内部 %MW4096 、%MW4097 的 Modbus 地址，假设 %MW4096 的值为 0x1234，%MW4097 的值为 0x5678。

• 请求信息:

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x03	0x1000	0x0002

• 回应信息:

事务和协议标识	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0007	0x01	0x03	0x04	0x1234	0x5678

◆ 功能码：04, 读取单个或多个字装置寄存器的值

• 请求信息数据结构:

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• 回应信息数据结构:

事务和协议标识	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

• 异常回应信息数据结构:

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例 :

通过 04 功能码读取控制器 16#8000、16#8001 地址的内容值 ,16#8000、16#8001 为控制器内部 %IW0 、%IW1 的 Modbus 地址，假设 %IW0 的值为 16#1234 、%IW1 的值为 16#5678。

• **请求信息：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x04	0x8000	0x0002

• **回应信息：**

事务和协议标识	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0009	0x01	0x04	0x04	0x1234	0x5678

◆ 功能码：06, 写单个字装置寄存器的值

• **请求信息数据结构：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• **回应信息数据结构：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	……	2Byte

• **异常回应信息数据结构：**

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【 Modbus 和 Modbus TCP 通讯相关】章节介绍

范例 :

通过 06 功能码将数值 16#1234 写入控制器 16#2000 地址内。

• **请求信息：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x06	0x2000	0x1234

• **回应信息：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x06	0x2000	0x1234

◆ 功能码：0x10, 写多个字装置寄存器的值

• **请求信息数据结构：**

事务/协议	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）	数据个数（Byte）	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	2Byte	……	2Byte

• **回应信息数据结构：**

事务/协议	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	……	2Byte

- **异常回应信息数据结构：**

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 16#10 功能码将 16#0020、16#0021 写入控制器 16#1234、16#5678 地址内，16#010，16#0011 为控制器内部 %QW16、%QW17 的 Modbus 地址。

- **请求信息：**

事务/协议	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）	数据个数（Byte）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	2Byte	2Byte
0x00000000	0x000B	0x01	0x10	0x0010	0x0002	0x04	0x0020	0x0021

- **回应信息：**

事务/协议	数据长度	从站站号	功能码	字装置首地址	数据个数（Word）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x10	0x1234	0x5678

◆ 功能码：01, 读位装置寄存器的值

- **请求信息数据结构：**

事务/协议	数据长度	从站站号	功能码	位装置首地址	数据个数（Bit）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

- **回应信息数据结构：**

事务/协议	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

备注：回应信息中数据个数（Byte2）的值由请求信息中的位装置个数（Byte4 和 Byte5）的值来决定。如请求信息中读取位装置的个数为 m，m 除以 8 的商为 n，如果可以整除，回应信息中位装置个数（Byte4 和 Byte5）的值为 n；反之回应信息中位装置个数（Byte4 和 Byte5）的值为 n + 1，详细请参考如下范例。

- **异常回应信息数据结构：**

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 01 功能码读取控制器 %QX0.0~%QX1.6 的状态值，%QX0.0 的地址为 16#A000，假设 %QX0.7~%QX0.0= 1000 0001，%QX1.6~%QX1.0=101 0001。

- **请求信息：**

事务/协议	数据长度	从站站号	功能码	位装置首地址	数据个数（Bit）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x01	0xA000	0x000F

- **回应信息：**

事务/协议	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x01	0x02	81	51

◆ 功能码：02, 读位装置寄存器的值

• 请求信息数据结构：

事务/协议	数据长度	从站站号	功能码	位装置首地址	数据个数（Bit）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• 回应信息数据结构：

事务/协议	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

备注：回应信息中数据个数（Byte2）的值由请求信息中的位装置个数（Byte4 和 Byte5）的值来决定。如请求信息中读取位装置的个数为 m，m 除以 8 的商为 n，如果可以整除，回应信息中位装置个数（Byte4 和 Byte5）的值为 n；反之回应信息中位装置个数（Byte4 和 Byte5）的值为 n + 1，详细请参考如下范例。

• 异常回应信息数据结构：

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 02 功能码读取控制器 %IX0.0~%IX1.6 的状态值，%IX0.0 的地址为 16#6000，假设 %IX0.7~%IX0.0= 1000 0001，%IX1.6~%IX1.0=101 0001。

• 请求信息：

事务/协议	数据长度	从站站号	功能码	位装置首地址	数据个数（Bit）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x02	0x6000	0x000F

• 回应信息：

事务/协议	数据长度	从站站号	功能码	数据个数（Byte）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x02	0x02	81	51

◆ 功能码：05, 设置单个位装置寄存器的值

• 请求信息数据结构：

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

备注：写入值为 16#0000 表示将 FALSE 写入位装置，16#FF00 表示将 TRUE 写入位装置。

• 回应信息数据结构：

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• 异常回应信息数据结构：

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 05 功能码设置控制器 %QX0.7 的值为 TRUE，%QX0.7 的地址为 16#A007。

• **请求信息：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x05	0xA007	0XFF00

• **回应信息：**

事务和协议标识	数据长度	从站站号	功能码	字装置首地址	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x05	0xA007	0XFF00	51

◆ 功能码：0x0F, 写多个位装置寄存器的值

• **请求信息数据结构：**

事务/协议	数据长度	从站站号	功能码	位装置首地址	数据个数（Bit）	数据个数（Byte）	数据值	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	1Byte	……	1Byte

备注：请求信息中有多少个 Byte 的数据由请求信息中欲写入位数值的个数决定。

• **回应信息数据结构：**

事务/协议	数据长度	从站站号	功能码	字装置首地址	数据个数（Bit）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte

• **异常回应信息数据结构：**

事务和协议标识	数据长度	从站站号	16#80+功能码	异常回应码
4Byte	2Byte	1Byte	1Byte	1Byte

备注：异常回应码见【Modbus 和 Modbus TCP 通讯相关】章节介绍

范例：

通过 0F 功能码设置控制器 %QX0.7~%QX0.0=1000 0001，%QX1.7~%QX1.0=1010 0011,%QX0.0 的地址为 16#A000，QX0=1.0 的地址为 16#A008。

• **请求信息：**

事务/协议	数据长度	从站站号	功能码	位装置首地址	数据个数（Bit）	数据个数（Byte）	数据值	数据值
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte	1Byte	1Byte	1Byte
0x00000000	0x0009	0x01	0x0F	0xA000	0x0010	0x02	0x81	0xA3

• **回应信息：**

事务/协议	数据长度	从站站号	功能码	字装置首地址	数据个数（Bit）
4Byte	2Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0x0006	0x01	0x0F	0xA000	0x0010

◆ 功能码：0x17, 读写单个或多个字装置的值

• **请求信息数据结构：**

事务/协议	从站站号	功能码	读字装置首地址	读数据个数（Word）	写字装置首地址	写数据个数（word）	写数据个数（Byte）	写数据值	写数据值	写数据值
4Byte	1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	……	2Byte

备注：请求信息中有多少个 Byte 的数据由请求信息中欲写入位数值的个数决定。

- **回应信息数据结构：**

事务/协议	从站站号	功能码	读数据个数（Byte）	读数据值	读数据值	读数据值
4Byte	1Byte	1Byte	1Byte	2Byte	……	2Byte

- **异常回应信息数据结构：**

事务和协议标识	从站站号	16#80+功能码	异常回应码
4Byte	1Byte	1Byte	1Byte

备注：异常回应码见【B.5 Modbus 异常回应码】章节介绍

范例：

通过 16#17 功能码将 16#000C、16#0064 写入控制器 16#0000、16#0001 地址内，16#0000，16#0001 为控制器内部 %MW0、%MW1 的 Modbus 地址，将从站 16#8000、16#8001 地址内的值读出，16#8000、16#8001 为从站控制器内部 %IW0、%IW1 的 Modbus 地址。

- **请求信息：**

事务/协议	从站站号	功能码	读字装置首地址	读数据个数（Word）	写字装置首地址	写数据个数（word）	写数据个数（Byte）	写数据值	写数据值
4Byte	1Byte	1Byte	2Byte	2Byte	2Byte	2Byte	1Byte	2Byte	2Byte
0x00000000	0xFF	0x17	0x8000	0x0002	0x0000	0x0002	0x04	0x000C	0x0064

- **回应信息：**

事务/协议	从站站号	功能码	读数据个数（Byte）	读数据值	读数据值
4Byte	1Byte	1Byte	1Byte	2Byte	2Byte
0x00000000	0xFF	0x17	0x04	0x0000	0x0000

第 8 章　通讯指令错误代码描述

8.1 通讯指令错误代码描述

错误码		含义	处理方法
十六进制	十进制		
1004	4100	CAN 从站站号超过允许的范围	检查 CAN 从站站号是否在允许的范围内
1301	4865	CANopen 主站给从站通过数据服务（SDO）写或者读参数时，从站没有回应	检查硬件连接正确。 检查指令中操作的从站存在 检查网络中所有站的波特率相同。 检查线缆为屏蔽双绞线。 检查附近是否有干扰源。
1302	4866	CANopen 主站给从站通过数据服务（SDO）写或者读参数时，从站没有正常回复	检查指定的参数存在。 检查写入参数值是否在参数允许的范围内。
1303	4867	CANopen 主站给从站通过数据服务（SDO）写或者读参数时，从站回复错误码	确认从站支持数据服务（SDO） 检查从站工作状态是否正常
4500	17664	CAN 从站站号超过允许的范围	检查 CAN 从站站号是否在允许的范围内
4501	17665	输入变量 Mode 的值超过允许的范围	修改并确保输入变量 Mode 的值在允许的范围内
4502	17666	诊断的从站未在软件中配置	请对软件中配置的从站执行诊断
4503	17667	CAN 通讯口未设置为主站模式	将 CAN 通讯口设置为主站模式
4510	17680	CAN 从站站号超过允许的范围	检查 CAN 从站站号是否在允许的范围内
4511	17681	诊断的从站不存在 (未配置）	对已配置的从站执行诊断功能
6000	24576	以太网 ModbusTCP 数据交换通道编号（LinkNum）设置值不在允许范围内	修改并确保 ModbusTCP 数据交换通道编号（LinkNum）在允许的范围内
6001	24577	以太网 ModbusTCP 数据交换通道配置的写操作长度（Writelength）超过允许的最大值	修改并确保所配置的写操作长度在允许的范围内
6002	24578	以太网 ModbusTCP 数据交换通道配置的读操作长度（Writelength）超过允许的最大值	修改并确保所配置的读操作长度在允许的范围内
6003	24579	以太网物理连接异常	检查并确保以太网硬件连接正常
6004	24580	以太网 Socket 编号设置错误	修改并确保参数 Socket 编号（SocketNum）在允许的范围内
6005	24581	以太网 Socket 功能所配置的发送长度超过上限	修改并确保发送长度在允许的范围内
6006	24582	以太网 Socket 功能所配置的接收长度超过上限	修改并确保接收长度在允许的范围内
6007	24583	以太网 ModbusTCP Link 功能的通讯超时时间 (TimeOut) 设置错误	修改并确保所配置的通讯超时时间在允许的范围内
6008	24584	以太网 Socket 功能所配置的发送接收长度均为 0	修改并确保发送和接收长度不同时为 0
6010	24592	串口 Modbus 数据交换通道参数配置的编号（LinkNum）设置错误	修改并确保参数 LinkNum 在允许的范围内
6011	24593	串口 Modbus 数据交换通道参数配置的写操作长度（Writelength）超过上限	修改并确保所配置的写操作长度在允许的范围内
6012	24594	串口 Modbus 数据交换通道参数配置的读操作长度（Readlength）超过上限	修改并确保所配置的读操作长度在允许的范围内
6013	24595	串口 Modbus 数据交换通道参数配置的发送数据长度（SendLength）超过上限	修改并确保发送数据长度在允许的范围内
6014	24596	串口 Modbus 数据交换通道参数配置的接收数据长度 (ReceiveLength）超过上限	修改并确保接收数据长度在允许的范围内
601A	24602	串口 Modbus 数据交换通道参数配置的通讯超时时间 (TimeOut）错误	修改并确保所配置的通讯超时时间在允许的范围内
601B	24603	串口 Modbus 数据交换通道参数配置的读操作长度（Readlength）和写操作长度（Writelength）均为 0	修改并确保读操作长度和写操作长度不同时为 0

错误码		含义	处理方法
十六进制	十进制		
601C	24604	串口自定义协议数据收发指令功能配置的发送数据长度（SendLength）和接收数据长度 (ReceiveLength）均为 0	修改并确保发送数据长度和接收数据长度不同时为 0
6020	24608	串口 Modbus 数据交换通道参数配置中 WORD 写操作数据缓存的起始地址（WriteAddr）缓存空间不足	修改本地写操作数据缓存的起始地址，并确保有足够空间满足指定的写操作长度
6021	24609	串口 Modbus 数据交换通道参数配置中 WORD 写操作所指定的本地缓存起始地址（WriteAddr）不在允许的范围内	修改并确保本地写操作数据缓存的 起始地址在允许区域内
6022	24610	串口 Modbus 数据交换通道参数配置中 WORD 写操作数据缓存的起始地址（WriteAddr）在允许的区域内，但不满足字地址的对齐关系	修改写操作所指定的本地缓存起始地址或者写操作数据缓存起 始地址的偏移量
6023	24611	串口 Modbus 数据交换通道参数配置中 WORD 读操作数据缓存的起始地址（ReadAddr）缓存空间不足	修改本地读操作数据缓存的起始地址，并确保有足够空间满足指定的读操作长度
6024	24612	串口 Modbus 数据交换通道参数配置中 WORD 读操作数据缓存的起始地址（ReadAddr）不在允许的范围内	修改并确保本地读操作数据缓存的起始地址在允许区域内
6025	24613	串口 Modbus 数据交换通道参数配置中 WORD 读操作数据缓存的起始地址（ReadAddr）在允许的区域内，但不满足字地址的对齐关系	修改读操作所指定的本地缓存起始地址或者读操作数据缓存起 始地址的偏移量
6026	24614	串口 Modbus 数据交换通道参数配置中 Bit 写操作数据缓存的起始地址（WriteAddr）缓存空间不足	修改本地写操作数据缓存的起始地址，并确保有足够空间满足指定的写操作长度
6027	24615	串口 Modbus 数据交换通道参数配置中 Bit 写操作所指定的本地缓存起始地址（WriteAddr）不在允许的范围内	修改并确保本地写操作数据缓存的 起始地址在允许区域内
6028	24616	串口 Modbus 数据交换通道参数配置中 Bit 读操作数据缓存的起始地址（ReadAddr）缓存空间不足	修改本地读操作数据缓存的起始地址，并确保有足够空间满足指定的读操作长度
6029	24617	串口 Modbus 数据交换通道参数配置中 Bit 读操作数据缓存的起始地址（ReadAddr）不在允许的范围内	修改并确保本地读操作数据缓存的起始地址在允许区域内
6030	24624	串口 Modbus 数据交换通道参数配置中中参数读写地址的类型（Mode）设置错误	修改并确保参数在允许的范围内
6031	24625	串口 Modbus 数据交换通道参数配置中读操作所指定的功能码（WriteMode）错误	修改并确保参数在允许的范围内
6040	24640	为发送操作所指定的本地缓存空间不足	修改本地缓存空间的起始地址或者发送的数据长度
6042	24642	为发送操作所指定的本地缓存的起始地址超出了允许的范围	修改并确保本地缓存的起始地址在允许的范围内
6043	24643	为接收操作所指定的本地缓存空间不足	修改本地缓存空间的起始地址或者发送的数据长度
6045	24645	为接收操作所指定的本地缓存的起始地址超出了允许的范围	修改并确保本地缓存的起始地址在允许的范围内
6050	24656	通信端口编号超出范围	修改并确保通信端口编号在允许的范围内
6051	24657	指定的端口不存在	确保指定的端口处于正常状态。例如，指定的端口为扩展卡上的端口，但对应的扩展卡未安装，将产生此错误。
6100	24832	以太网 Socket 功能参数设置错误	修改并确保 Socket 功能相关参数到允许的范围内
6101	24833	以太网物理连接错误	检查以太网物理连接是否正常
6102	24834	以太网 TCP 远端 IP 地址错误	修改并确保远端 IP 地址正确
6103	24835	以太网 TCP 端口错误	修改并确保远端端口正确
6105	24837	以太网 TCP 发送缓存的地址错误	修改并确保发送缓存的地址正确
6106	24838	以太网 TCP/UDP 接收动作已触发	待接收完成后再执行
6107	24839	以太网 TCP 接收缓存的地址错误	修改并确保接收缓存的地址正确
6108	24840	作为 TCP 服务器，实际接收数据长度超出设定长度	修改并确保接收长度大于等于接收到的第一笔数据的长度（以 Byte 为单位）

错误码		含义	处理方法
十六进制	十进制		
6109	24841	以太网 UDP 实际接收数据长度超出设定长度	修改并确保接收长度大于等于接收到的第一笔数据的长度（以 Byte 为单位）
610A	24842	以太网 UDP 远端 IP 地址错误	修改并确保远端 IP 地址正确
610B	24843	以太网 UDP 端口错误	修改并确保本地和远端端口不能同时为 0
610C	24844	以太网 UDP 发送缓存的地址错误	修改并确保发送缓存的地址正确
610D	24845	以太网 UDP 接收缓存的地址错误	修改并确保接收缓存的地址正确
610E	24846	以太网 TCP 连接超时	检查 Socket 配置是否正确或远端设备是否正常
610F	24847	作为 TCP 客户端，实际接收数据长度超出设定长度	修改并确保接收长度大于等于接收到的第一笔数据的长度（以 Byte 为单位）
6110	24848	以太网 TCP 连接被远端设备拒绝	确保远端设备正常后重新建立连接
6111	24849	以太网 TCP/UDP 连接尚未开启	检查连接是否处于开启状态
6112	24850	以太网 TCP/UDP 连接已触发	确保不在连接建立过程中重复触发建立连接
6113	24851	以太网 TCP/UDP 数据发送已触发	等待发送完成后再触发
6114	24852	以太网 TCP/UDP 连接已建立	确保不在连接已建立情形下重复触发建立连接
6115	24853	以太网 TCP/UDP 连接正在关闭	确保不在连接关闭过程中重复触发关闭
6116	24854	以太网 TCP/UDP 连接未关闭	确保在连接关闭的情形下配置 Socket 参数
6117	24855	以太网 TCP/UDP 指定的发送长度错误	修改并确保发送长度在允许范围内
6118	24856	以太网 TCP/UDP 指定的接收长度错误	修改并确保接收长度在允许范围内
6120	24864	ModbusTCP 主站未启动	确保执行本指令时 ModbusTCP 主站已启动，可通过指令启动 ModbusTCP 主站功能
6121	24865	ModbusTCP 数据交换未配置	确保欲操作的数据交换通道已通过软件或指令完成配置
6122	24866	ModbusTCP 数据交换模式错误	检查参数 ExeMode 的范围，确保其在允许的范围内（0~1）
6123	24867	ModbusTCP 数据交换连接保持时间（LinkKeepTime）设置错误	检查参数 LinkKeepTime 的范围，确保其在允许的范围内，LinkKeepTime 须大于 0
6124	24868	TCP 连接已关闭	确认 TCP 连接已打开
6125	24869	TCP 连接超时	未在指定的时间内建立 TCP 连接。
6126	24870	ModbusTCP 消息回复超时	未在指定的时间内回复 ModbusTCP 报文。
6129	24873	ModbusTCP 报文中事务处理标识符错误（报文头部）	ModbusTCP 报文中事务处理标识符不符合规则
612A	24874	ModbusTCP 报文中协议标识符错误（报文头部）	确认 ModbusTCP 报文中协议标识符不为 0。
612B	24875	ModbusTCP 报文长度错误（报文头部）	检查 ModbusTCP 报文长度
612D	24877	TCP 连接未关闭时，又执行建立连接	确保 TCP 连接关闭时，再建立连接
6130	24880	串口读写地址的类型（Mode）设置错误	检查参数 Mode 的范围，确保其在允许的范围内（0 或 1）
6131	24881	Modbus 主站功能未启动	先启动 Modbus 主站功能
6132	24882	Modbus 数据交换未配置	确保欲操作的数据交换通道已通过软件或指令完成配置
6133	24883	串口工作在从站模式，不允许启动 Modbus 主站	确保串口工作在主站模式，再执行串口主站开启指令
6134	24884	Modbus 数据交换模式（ExeMode）错误	检查参数 ExeMode 的范围，确保其在允许的范围内（0~1）
6135	24885	不能识别的功能码	检查从站回复给主站报文数据中的功能码是否正确
6136	24886	Modbus 从站回复数据中的地址与配置地址不同	检查从站回复给主站的报文数据是否正确
6137	24887	Modbus 从站回复数据中的接收数据长度与配置长度不符	检查从站回复给主站的报文数据是否正确
6138	24888	Modbus 接收超时	检查硬件连接正确。 检查网络中所有站的波特率相同。 检查线缆为屏蔽双绞线。 检查附近是否有干扰源。
6139	24889	Modbus 校验错误	
613B	24891	Modbus 实际接收长度超出最大接收长度	检查从站回复给主站的报文数据是否正确
7000	28672	当前设备不支持该功能	检查当前设备支持当前使用的功能

创新 诚信 服务

禾川科技HCFA

禾川自动化中心ATC

浙江禾川科技股份有限公司
浙江省衢州市龙游县工业园区阜财路9号

杭州研发中心
浙江省杭州市临安区青山湖街道励新路299号

400热线电话-400-012-6969

禾川官网网址-www.hcfa.cn