Truhigh P500 PLC

通讯手册

V1.4

创恒
Truhigh

目录

目录 1

第 1 章 MODBUS 通讯使用 2

1.1 MODBUS TCP 主站 2
1.2 MODBUS TCP 从站 8
1.3MODBUS RTU 主站 12
1.4 MODBUS RTU 从站 29
1.5 PU510 作为 MODBUS 从站（MBS） 39
1.6 MCGS 通讯连接 51
1.7 MODBUS RTU 硬件组态 62
1.8 MODBUS TCP 硬件组态 66

第 2 章 CI510 通讯配置 70

2.1 添加 I/O 配置组合 70
2.2 手动添加 CI510 以及 IO 模块 71
2.3 网络自动添加 CI510 以及 IO 模块 73

第 3 章 自由口通讯使用 76

3.1 物理端口配置 76
3.2 数据接收 77
3.3 数据发送 79

第 4 章 连接第三方 PLC 82

4.1 西门子 S7-200 连接 82
4.2 三菱 FX3GA 连接 89

第 5 章 MQTT 连接 95

5.1 ALIYUN 连接设置 95
5.2 AZURE 连接设置 98
5.3 ONENET 连接设置 101
5.4 SCHNEIDER 连接设置 103
5.5 TRUHIGH 连接设置 105

第 1 章　Modbus 通讯使用

P500 系列 PLC 的标准 Modbus 通讯分为功能块和硬件组态两种模式，功能块模式必须添加有 TifsFwLib.FWL 固件库，数据存放于 M3 分区或自定义数据类型。

- **1.1　Modbus TCP 主站**：功能块模式，PLC 作为 TCP 主站，访问 TCP 从站
- **1.2　Modbus TCP 从站**：功能块模式，PLC 作为 TCP 从站，被 TCP 主站访问
- **1.3　Modbus RTU 主站**：功能块模式，PLC 作为 RTU 主站，访问 RTU 从站
- **1.4　Modbus RTU 从站**：功能块模式，PLC 作为 RTU 从站，被 RTU 主站访问
- **1.5　PU510 作为 Modbus 从站**：硬件组态模式，PLC 作为 TCP/RTU 从站，被主站访问
- **1.6　MCGS 通讯连接**：硬件组态模式，PLC 作为 TCP 或 RTU 从站，被 MCGS 主站访问
- **1.7　Modbus RTU 硬件组态**：硬件组态模式，PLC 作为 RTU 主站，访问 RTU 从站
- **1.8　Modbus TCP 硬件组态**：硬件组态模式，PLC 作为 TCP 主站，访问 TCP 从站

1.1 Modbus TCP 主站

1.1.1 Modbus TCP 通讯初始化

Modbus TCP 主站通信初始化采用 MBUS_TCP_CTRL 功能块：

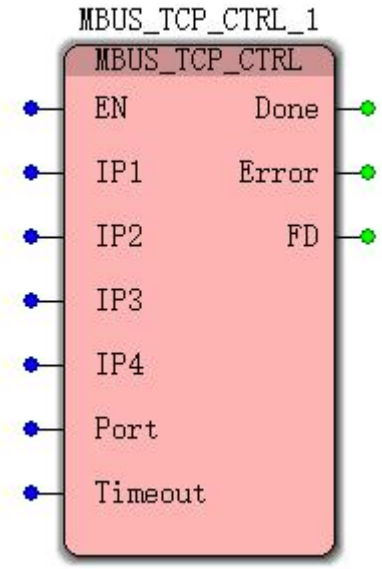


图 1-1-1

EN：为功能块使能，为 TRUE 时该功能块才有效，否则无效。

IP1-IP4：为从站 IP 地址，例如 IP1=192,IP2=168,IP3=0,IP4=100 表示从站地址为 192.168.0.100。

Port：从站网络端口号，典型值为 502。

Timeout：连接超时时间，等待从站做出响应时间，单位 ms。

Done：功能块执行完成标志，只有初始化成功才为 TURE。

Error：执行状态码，执行完成并且无错误时为 0，否则为相应错误码。

FD:Modbus 连接标识号,每个 MBBUS_TCP_CTRL 产生唯一的标识号,在使用 MBUS_TCP_MSG 功能码时需要改标识号。

详细功能块说明参考《Truhigh P500 功能块手册》第三章，Truhigh_TifsFwlib 库，在此只介绍例程。

例程如下：

CTRL_EN 默认值为 1；

time_set 默认值为 1；

IP_ADDR1-IP_ADDR4 为 192.168.1.34；

PORT 为 502；

TIME_OUT 为 1000ms

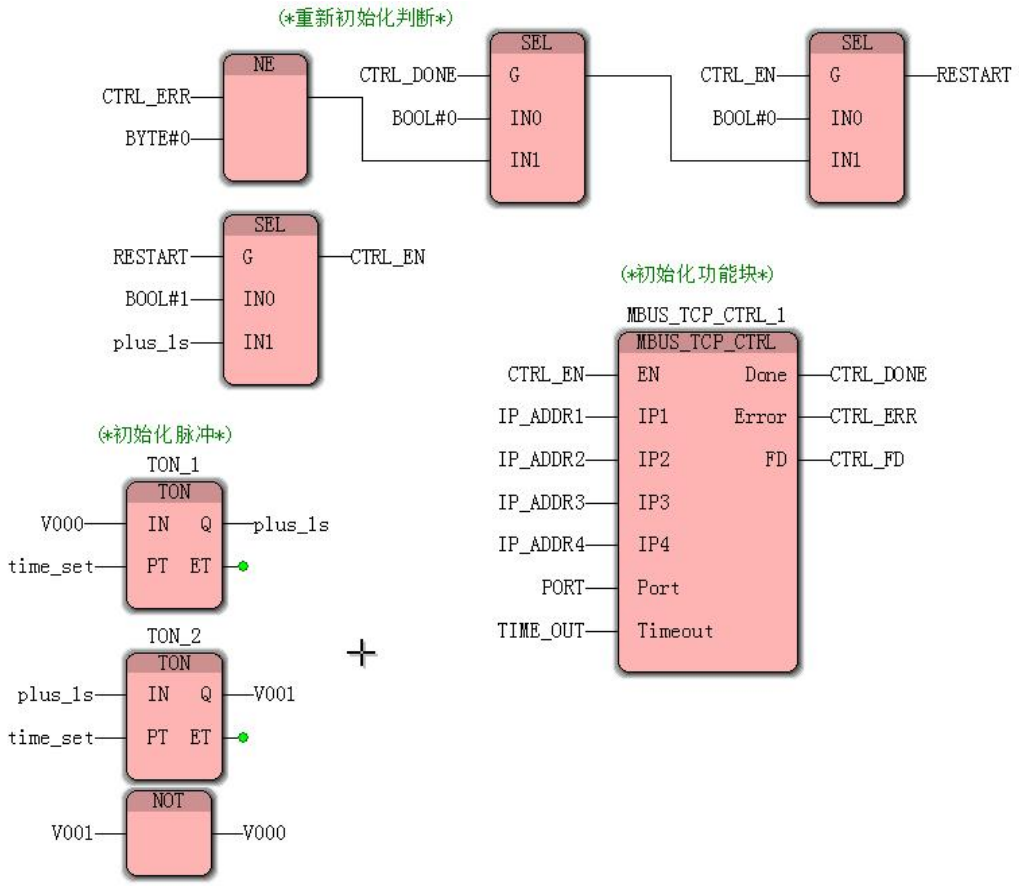


图 1-1-2

重新初始化判断逻辑：如果初始化功能块执行完成后 CTRL_DONE 为 TRUE 并且 CTRL_ERR 不为 0，则需要重新给 CTRL_EN 一个上升沿，重新建立连接。

初始化脉冲逻辑：产生脉冲；

1.1.2 Modbus TCP 数据通讯

Modbus TCP 数据通讯采用 MBUS_TCP_MSG 功能块：

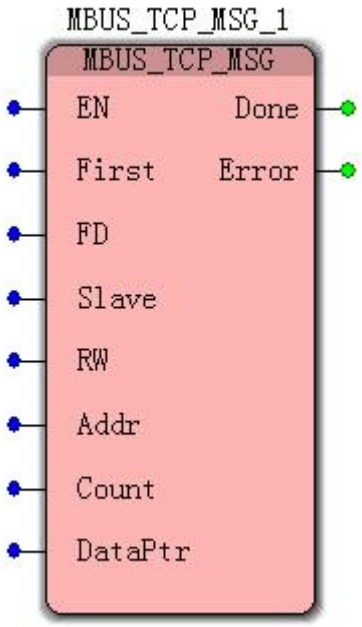


图 1-1-3

EN：使能控制，如果启用功能块，必须为 TRUE。

First：通讯请求脉冲，由 FALSE 变为 TRUE 时，按照给定参数执行一次数据请求；需要连续的给定上升沿脉冲，才能进行连续的数据请求。

FD：Modbus 连接标识号，由 MBUS_TCP_CTRL 产生。

Slave：从站设备地址 ID。

RW：读写指示，0 为读，1 写。

Addr：读写寄存器地址，根据地址不同判断读写哪一类寄存器，1-65535（线圈寄存器 0xxxxx）、100001-165535（离散寄存器 1xxxxx）、300001-365535（输入寄存器 3xxxxx）、400001-465535（保持寄存器 4xxxxx），第一个数字表示寄存器类型，后面 5 位表示寄存器地址；地址为从编号从 1 开始，最大 65535，对应的 modicon（莫迪康）地址从 0 开始。

Count：单次读写从站寄存器个数，线圈寄存器最多：1920 个、离散寄存器最多：1920 个、输入寄存器最多：120 个、保持寄存器最多：120 个。

DataPtr：读写数据存放首地址，类型为 Any，需要指向一块地址连续的存储单元，建议变量为数组变量或者地址为 M3 的共享内存区。

Done：功能块执行完成标志，当 First 通讯请求脉冲由 FALSE 变为 TRUE 时，Done 变为 FALSE，当功能块执行完成（从站正常返回数据或者产生错误终止）后，变为 TRUE。

Error：错误代码，BYTE 类型，仅当 Done 输出为 TRUE 时，Error 输出才有效。

详细功能块说明参考《Truhigh P500 功能块手册》第三章，Truhigh_TifsFwlib 库，在此只介绍例程。

例程如下：

MSG_TCP_CTRL 初始化成功判断逻辑：初始化成功后才能进行数据的读写。

读写脉冲产生逻辑：产生 1 秒读写脉冲，可根据需求自己编写读写脉冲逻辑和读写间隔时间。读写上升沿只有在 EN 为 TRUE（1）时才有效。

slave_id 设置为 1；
rw_r 设置为 0，表示读寄存器；
reg_addr 为 406001，表示读保持寄存器 6001（莫迪康地址为 6000）寄存器。
reg_cnt 为 16，表示读取 16 个寄存器。
reg_data_M3，设置如下：
数据类型为 BYTE，用法为局部变量（VAR）也可设为全局变量（VAR_GLOBAL），I/O 地址为%MB3.100，则从%MB3.100 开始 16 个保持寄存器（32 个字节）存放读取的数据。

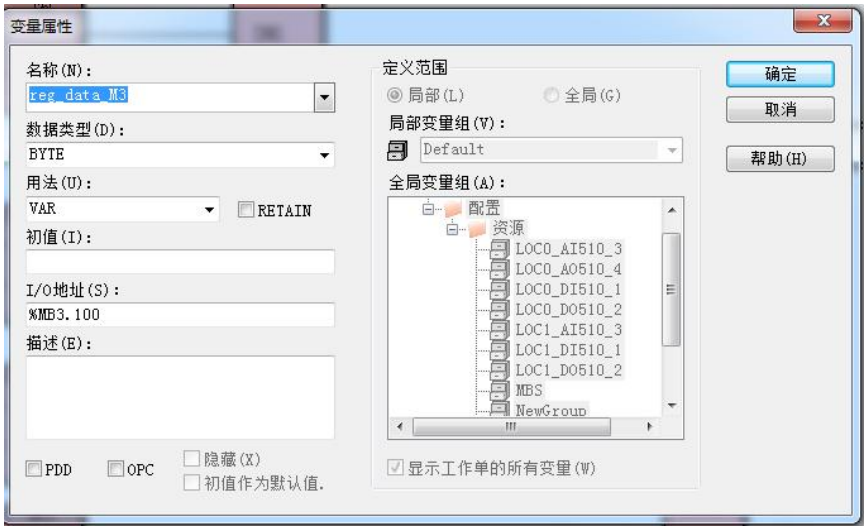


图 1-1-4

当需要获取数据时设置变量地址在此范围内，如例程中变量 hold_reg1 和 hold_reg2 属性如下：

图 1-1-5

表示 hold_reg1 数据内容和保持寄存器 6000 内容一致，hold_reg2 内容和保持寄存器 6001 内容一致。

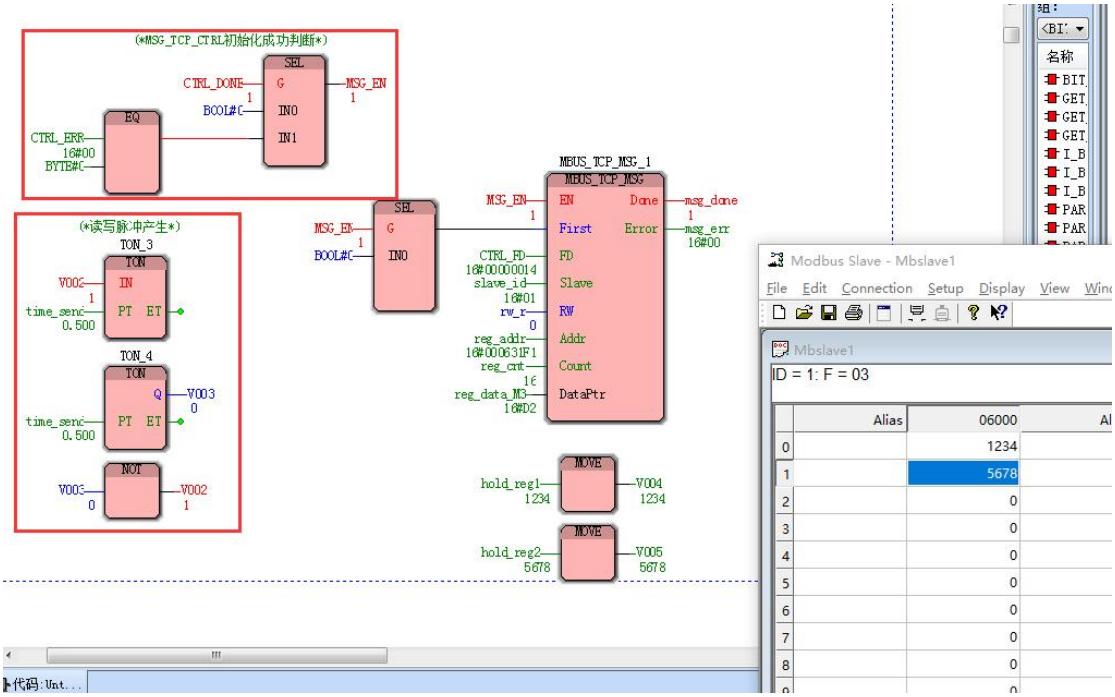


图 1-1-6

DataPtr 如果不采用 M3 区变量，可以用数组形式，例如定义数组类型为

TYPE

　　MB_INT_100 : ARRAY [0..99] OF INT;

END_TYPE

hold_datas 属性如下：

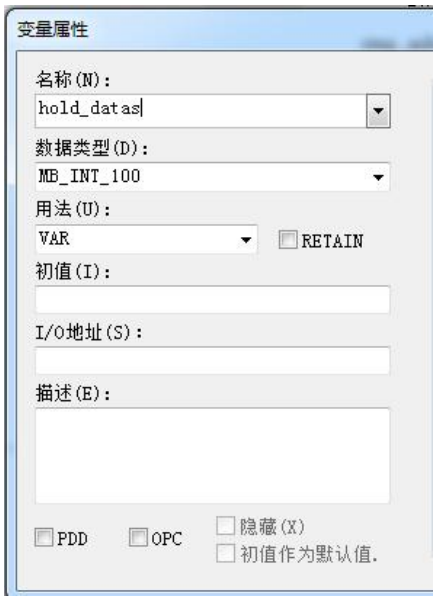


图 1-1-7

则对应数据如下：

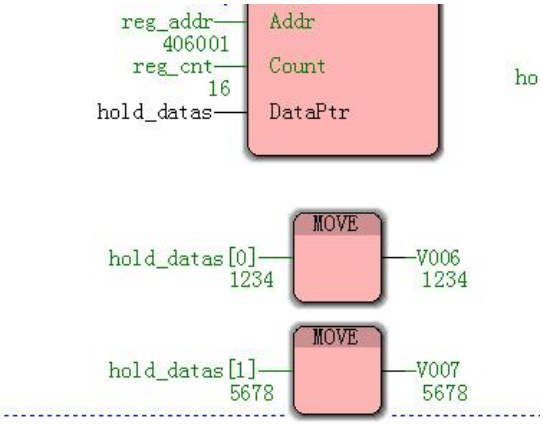


图 1-1-8

每个 hold_datas[x]数组成员代表一个寄存器数据。

如果针对一个从站既有读也有写类型，并且寄存器类型也都多种，则需要采用多个 MBUS_TCP_MSG 功能块，例程如下：

不同 MBUS_TCP_MSG 功能块需要轮训执行，所以需要设计相应的轮训程序。

MBUS_TCP_MSG1 功能为读取保持寄存器数据，起始地址为 6001，个数为 16 个；

MBUS_TCP_MSG2 功能为写保持寄存器数据，起始地址为 6101，个数为 8 个；

MBUS_TCP_MSG3 功能为读线圈寄存器数据，起始地址为 48001，个数为 10 个；

MBUS_TCP_MSG4 功能为写线圈寄存器数据，起始地址为 48101，个数为 10 个；

time_tvl 为每个功能块执行间隔时间，设置为 100ms。

数据类型定义为

TYPE

MB_INT_100 : ARRAY [0..99] OF INT;

MB_BOOL_100 : ARRAY [0..99] OF BOOL;

END_TYPE

hold_data_r 和 hold_data_w 数据类型为 MB_INT_100 ；

coil_data_r 和 coil_data_w 数据类型为 MB_BOOL_100 ；

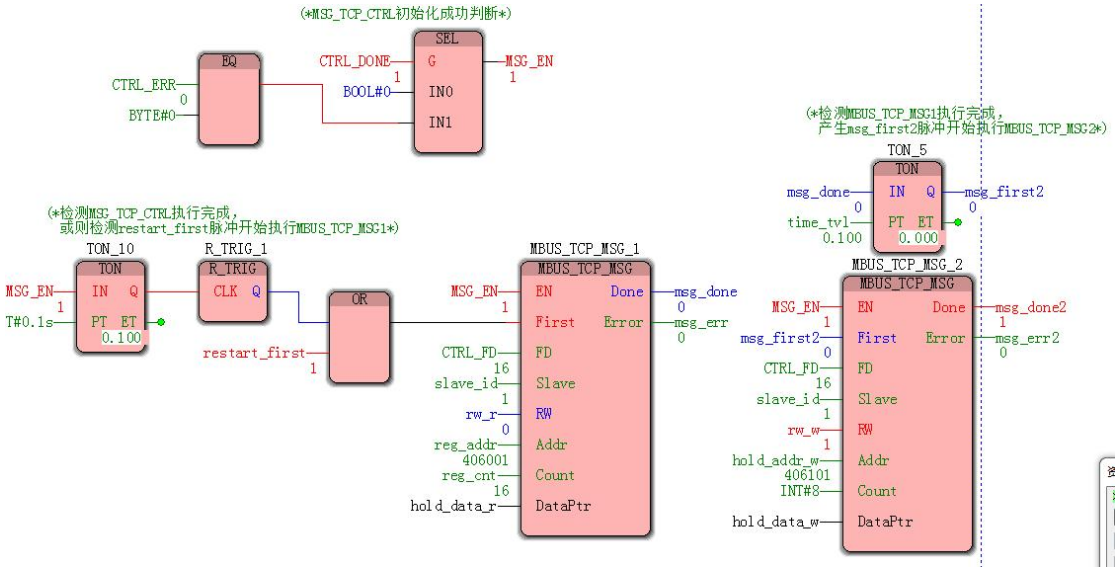


图 1-1-9

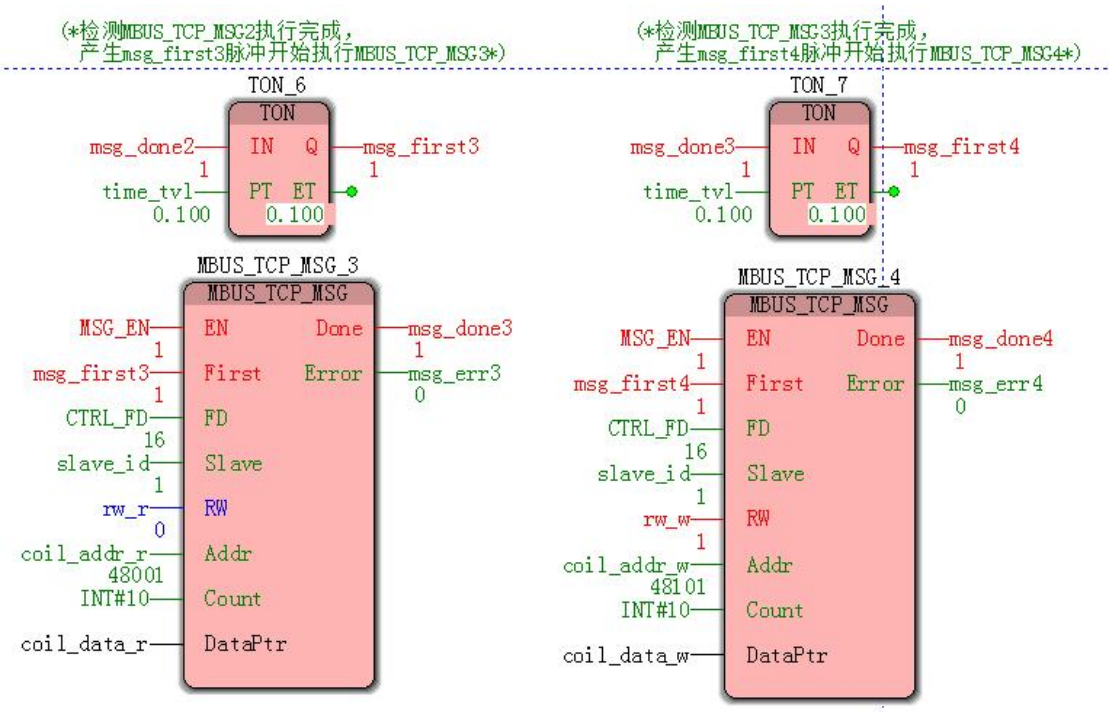


图 1-1-10

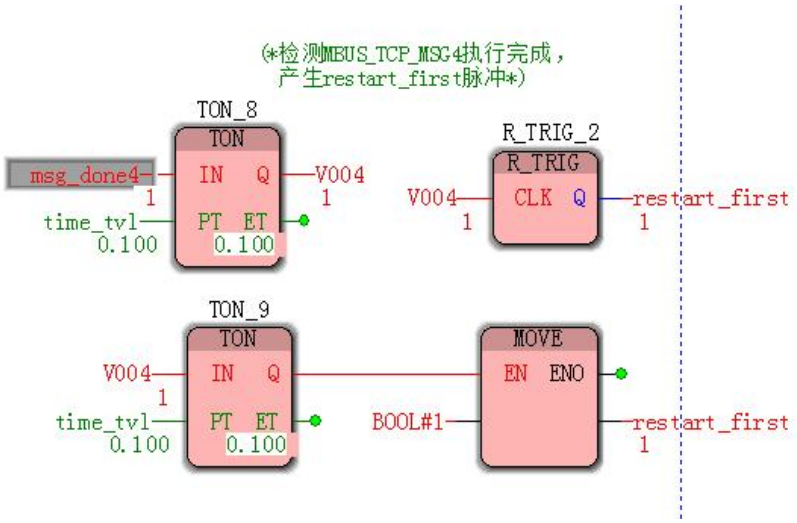


图 1-1-11

1.2 Modbus TCP 从站

1.2.1 Modbus 从站通讯初始化

Modbus 从站通讯初始化采用 MBUS_TCP_INIT 功能块。

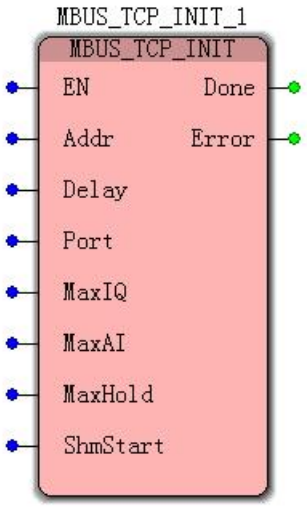


图 1-2-1

EN：使能控制，为 TRUE(1)时该功能块才有效。

Addr：从站地址号 ID；

Delay：延迟相应时间，延迟相应主站的读写操作，单位 ms，典型值为 0。

Port：网络端口号，典型值为 502。

MaxIQ：用于设置 Modbus 寄存器（线圈和离散）可用的点数（寄存器个数）。

MaxAI：用于设置 Modbus 寄存器（输入寄存器）可用的点数（寄存器个数）。

MaxHold：用于设置 Modbus 寄存器（保持寄存器）可用的点数（寄存器个数）。

ShmStart：用于设置 Modbus 从站数据的起始地址，必须存放在 M3 区，合法地址为 0-10239。

Done：功能块执行完成标志。

Error：功能块执行结果，为 0 时表示没有错误。

Modbus 从站通讯状态指示功能块采用 MBUS_TCP_SLAVE 功能块。

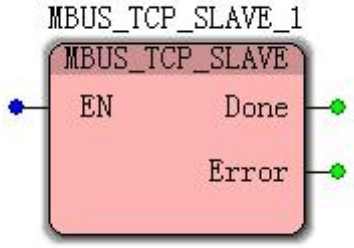


图 1-2-2

EN：使能控制，当为 TRUE（1）时判断从站接收状态。

Done：功能块执行完成标志。

Error：从站接收状态，为 0 时表示连接正常。

详细功能块说明参考《Truhigh P500 功能块手册》第三章，Truhigh_TifsFwlib 库，在此只介绍例程。

例程如下：

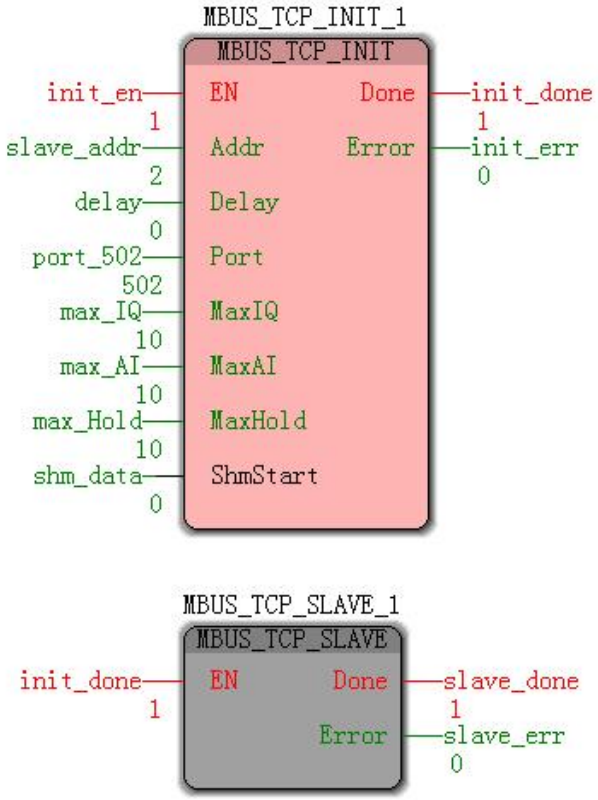


图 1-2-3

slave_addr 设置为 2;

port_502 设置为 502；

max_IQ、max_AI、max_Hold 设置为 10;

shm_data 设置地址为%MB3.1000，数据类型为 BYTE。

各类型寄存器在 M3 区的存储顺序为先是线圈寄存器，然后是离散输入寄存器，之后是只读输入寄存器，最后是保存寄存器，如果长度为 0 则跳过相应寄存器，如上例程：

MaxIQ = 10；

MaxAI = 10；

MaxHold = 10；

ShmStart = %MB3.1000;

则相应寄存器数据起始地址如下：

线圈寄存器的起始地址为%MB3.1000，长度占用(10 + 7)/8 = 2 字节，即 3.1000 和 3.1001；

离散输入寄存器的起始地址为%MB3.1002，长度占用(10 + 7)/8 = 2 字节，即 3.1002 和 3.1003；

只读输入寄存器的起始地址为%MB3.1004，长度占用 10*2 = 20 字节，3.1004 到 3.1023；

保存寄存器的起始地址为%MB3.1024，长度占用 10*2 = 20 字节，3.1024 到 3.1043；

Modbus 从站寄存器在 M3 区存放地址规则如下：

每个线圈和离散寄存器占用 M3 的一个 bit 位，例程中线圈和离散寄存器地址和 M3 区地址对应关系如下：表 1-2-1

线圈寄存器地址	M3 地址	离散寄存器地址	M3 地址
0-7	%MX3.1000.0-%MX3.1000.7	0-7	%MX3.1002.0-%MX3.1002.7

8-9	%MX3.1001.0-%MX3.1001.1	8-9	%MX3.1003.0-%MX3.1003.1

可建立相应变量，填写对应的地址，即可访问对应的 Modbus 寄存器数据，如下表：

15	coil_data_0	BOOL	VAR	线圈寄存器0	%MX3.1000.0		
16	coil_data_1	BOOL	VAR	线圈寄存器1	%MX3.1000.1		
17	coil_data_2	BOOL	VAR	线圈寄存器2	%MX3.1000.2		
18	coil_data_3	BOOL	VAR	线圈寄存器3	%MX3.1000.3		
19	coil_data_4	BOOL	VAR	线圈寄存器4	%MX3.1000.4		
20	coil_data_5	BOOL	VAR	线圈寄存器5	%MX3.1000.5		
21	coil_data_6	BOOL	VAR	线圈寄存器6	%MX3.1000.6		
22	coil_data_7	BOOL	VAR	线圈寄存器7	%MX3.1000.7		
23	coil_data_8	BOOL	VAR	线圈寄存器8	%MX3.1001.0		
24	coil_data_9	BOOL	VAR	线圈寄存器9	%MX3.1001.1		
25	disp_data_0	BOOL	VAR	离散寄存器0	%MX3.1002.0		
26	disp_data_1	BOOL	VAR	离散寄存器1	%MX3.1002.1		
27	disp_data_2	BOOL	VAR	离散寄存器2	%MX3.1002.2		
28	disp_data_3	BOOL	VAR	离散寄存器3	%MX3.1002.3		
29	disp_data_4	BOOL	VAR	离散寄存器4	%MX3.1002.4		
30	disp_data_5	BOOL	VAR	离散寄存器5	%MX3.1002.5		
31	disp_data_6	BOOL	VAR	离散寄存器6	%MX3.1002.6		
32	disp_data_7	BOOL	VAR	离散寄存器7	%MX3.1002.7		
33	disp_data_8	BOOL	VAR	离散寄存器8	%MX3.1003.0		
34	disp_data_9	BOOL	VAR	离散寄存器9	%MX3.1003.1		

图 1-2-4

每个输入和保持寄存器占用 M3 的两个字节，例程中线圈和离散寄存器地址和 M3 区地址对应关系如下：表 1-2-2

输入寄存器地址	M3 地址	保持寄存器地址	M3 地址
0-10	%MW3.1004-%MW3.1022	0-10	%MW3.1024-%MW3.1042

可建立相应变量，填写对应的地址，即可访问对应的 Modbus 寄存器数据，如下表：

35	input_data_0	INT	VAR	输入寄存器0	%MW3.1004	
36	input_data_1	INT	VAR	输入寄存器1	%MW3.1006	
37	input_data_2	INT	VAR	输入寄存器2	%MW3.1008	
38	input_data_3	INT	VAR	输入寄存器3	%MW3.1010	
39	input_data_4	INT	VAR	输入寄存器4	%MW3.1012	
40	input_data_5	INT	VAR	输入寄存器5	%MW3.1014	
41	input_data_6	INT	VAR	输入寄存器6	%MW3.1016	
42	input_data_7	INT	VAR	输入寄存器7	%MW3.1018	
43	input_data_8	INT	VAR	输入寄存器8	%MW3.1020	
44	input_data_9	INT	VAR	输入寄存器9	%MW3.1022	
45	hold_data_0	INT	VAR	保持寄存器0	%MW3.1024	
46	hold_data_1	INT	VAR	保持寄存器1	%MW3.1026	
47	hold_data_2	INT	VAR	保持寄存器2	%MW3.1028	
48	hold_data_3	INT	VAR	保持寄存器3	%MW3.1030	
49	hold_data_4	INT	VAR	保持寄存器4	%MW3.1032	
50	hold_data_5	INT	VAR	保持寄存器5	%MW3.1034	
51	hold_data_6	INT	VAR	保持寄存器6	%MW3.1036	
52	hold_data_7	INT	VAR	保持寄存器7	%MW3.1038	
53	hold_data_8	INT	VAR	保持寄存器8	%MW3.1040	
54	hold_data_9	INT	VAR	保持寄存器9	%MW3.1042	

图 1-2-5

1.3Modbus RTU 主站

1.3.1 确定通讯要求

1、modbus rtu 通讯需要使用 485 进行通讯，故选择扩展端口中的 COM0、COM1、COM2，此处选择 COM0 进用于 modbus rtu 通讯。

2、根据总线物理通讯速率与校验方式确定端口的波特率、校验位，此处选择 115200 波特率、无校验的通讯方式。

3、为保障 modbus rtu 通讯的严谨型，设定 0-65535ms 的通讯超时时间，此处选择 1000ms.

1.3.2 配置物理端口

确定 modbus rtu 通讯要求后，直接在工程中使用 MBUS_RTU_CTRL 功能块对 modbus rtu 通讯所使用的扩展端口配置即可。如下图所示。

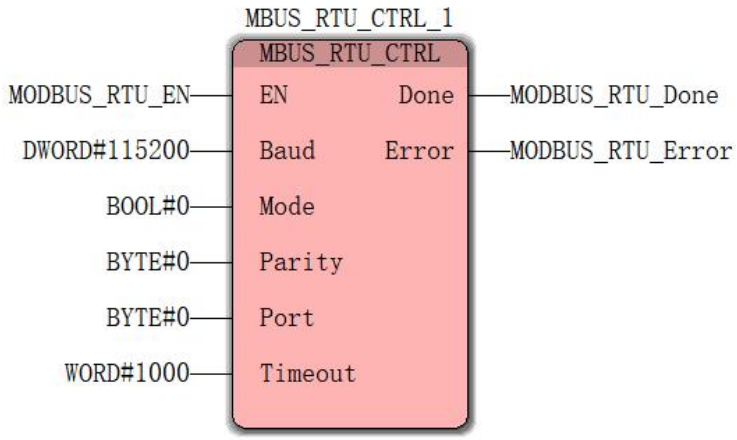


图 1-3-1

EN 位连接 MODBUS_RTU_EN，用于 MBUS_RTU_CTRL_1 功能块的使能与禁用的控制，置为 True，方能够配置扩展端口 COM0 作为 modbus rtu 通讯，置为 False，扩展端口 COM0 上的 modbus rtu 通讯功能禁用。

Baud 传入参数 DWORD#115200，即设置波特率为 115200；

Mode 传入参数 BOOL#0，Mode 用于设置功能块模式，当前为保留选项，该参数设置为 0 即可；

Parity 传入参数 BYTE#0，即表示设置校验位为无校验。0 为无校验，1 为奇校验，2 为偶校验；

Port 传入参数 BYTE#0，即表示使用扩展端口 0。0 为 COM0，1 为 COM1；

Timeout 传入参数 WORD#1000，即表示 modbus rtu 的超时时间设置为 1000ms，通讯延迟 1000ms 则认为 modbus rtu 通讯超时；

Done 为配置结果输出位，结果传入变量 MODBUS_RTU_Done。

Error 为配置错误信息输出，结果传入变量 MODBUS_RTU_Error。

以上仅为 **1.3.1 确定通讯要求** 参数下的配置，对 MBUS_RTU_CTRL 功能块的使用详

见功能块手册第三章 2.1 章节。

工程运行后 MODBUS_RTU_EN 置为 True，之后 MODBUS_RTU_Done 结果为 True，MODBUS_RTU_Error 结果为 0，即表示将 COM0 配置为 modbus rtu 所用的物理端口成功。如下图所示。

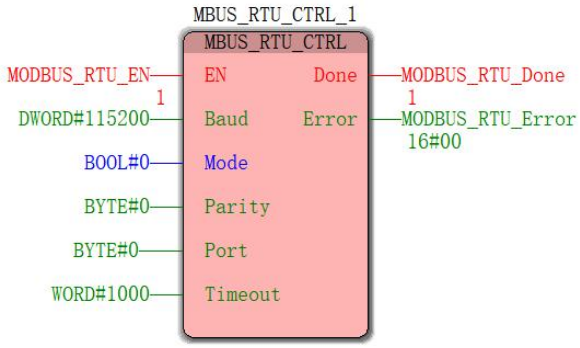


图 1-3-2

1.3.3 modbus rtu 寄存其操作

当端口配置成功后，便可以使用 MBUS_RTU_MSG 功能块对目标 modbus 寄存器进行操作。MBUS_RTU_MSG 功能块的使用方法详见功能块手册第三章 2.2 章节。本文仅对 modbus 的部分寄存器进行举例介绍。

1.3.3.1 线圈寄存器的读写操作

假设目标从站的地址为 1，对其前 10 个线圈寄存器进行读写。配置功能块如下。

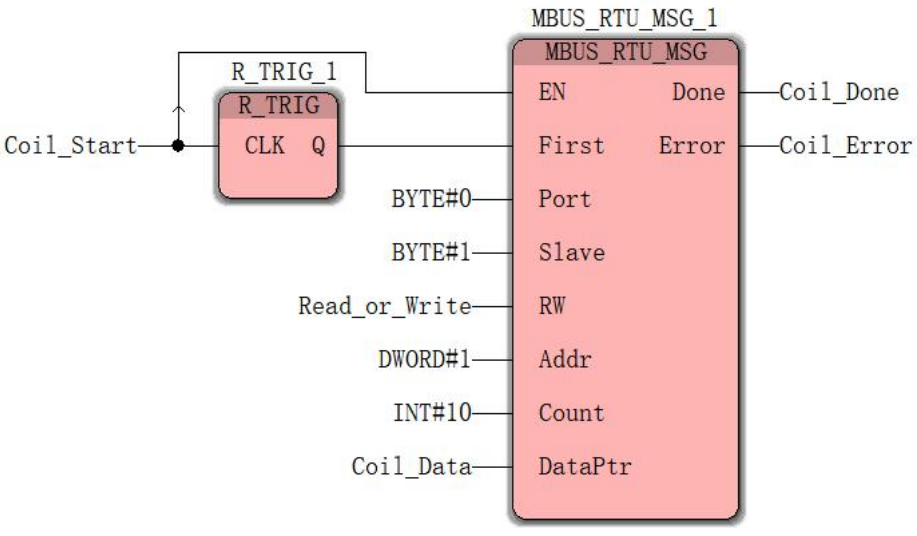


图 1-3-3

Coil_Start 用于对 modbus 线圈的操作控制，其参数由 False 转 True，便会完成一次对从站的指定线圈进行一次读写操作，其中 R_TRIG 功能块用于上升沿检测。

Port 传入的端口号为之前使用 MBUS_RTU_CTRL_1 配置过的端口。

Slave 传入参数即为目标从站的地址。

RW 为灵活控制读写，传入 Read_or_Write 变量中的参数，0 为读操作，1 为写操作。

Addr 传入 modbus 寄存器的地址，此处为 1.

Count 传入参数为操作寄存器的个数，即自 Addr 传入的地址，连续操作寄存器的个数，此处为自 modbus 地址 1（线圈 1）连续操作 10 个寄存器，线圈 1 至线圈 10。

DataPtr 为 modbus 线圈寄存器 1 至 10 存放的首地址，此处选择自 M3 分区的首地址。配置如下。

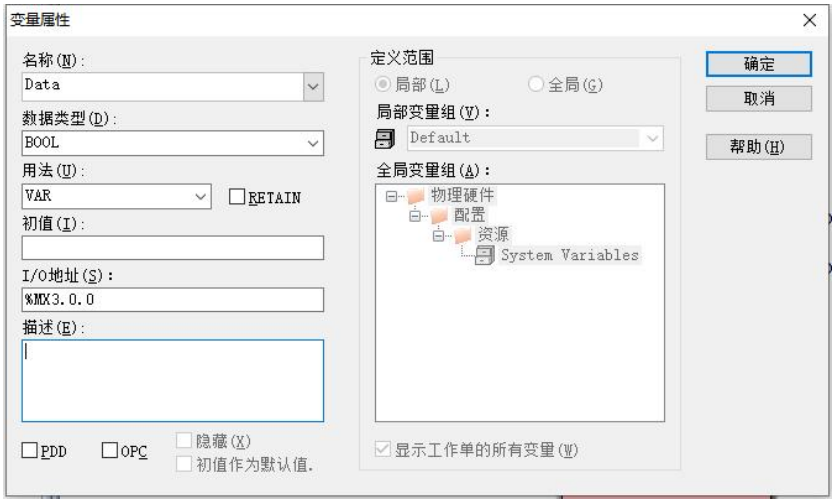


图 1-3-4

为方便观察，建立线圈寄存器的变量表，如下图所示。

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Coil											
	Coil_1	BO...	VAR_GL...	线圈寄存器1	%MX3.0.0							
	Coil_2	BO...	VAR_GL...	线圈寄存器2	%MX3.0.1							
	Coil_3	BO...	VAR_GL...	线圈寄存器3	%MX3.0.2							
	Coil_4	BO...	VAR_GL...	线圈寄存器4	%MX3.0.3							
	Coil_5	BO...	VAR_GL...	线圈寄存器5	%MX3.0.4							
	Coil_6	BO...	VAR_GL...	线圈寄存器6	%MX3.0.5							
	Coil_7	BO...	VAR_GL...	线圈寄存器7	%MX3.0.6							
	Coil_8	BO...	VAR_GL...	线圈寄存器8	%MX3.0.7							
	Coil_9	BO...	VAR_GL...	线圈寄存器9	%MX3.1.0							
	Coil_10	BO...	VAR_GL...	线圈寄存器10	%MX3.1.1							

图 1-3-5

Done 与 Error 为功能块执行结果，分别传入 Coil_Done 与 Coil_Error 变量，若 Coil_Done 为 True，Coil_Error 为 0，表示此次操作成功。

1、读取线圈寄存器

（1）使用 Modbus Slave 软件，设置站地址为 1，并把 10 个线圈寄存器全部置为 1。

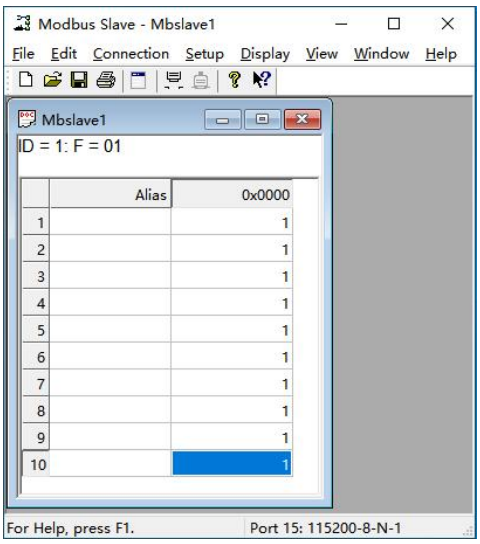


图 1-3-6

（2）使用功能块进行读操作，即 Coil_Start 由 False 转 True，如下图所示；

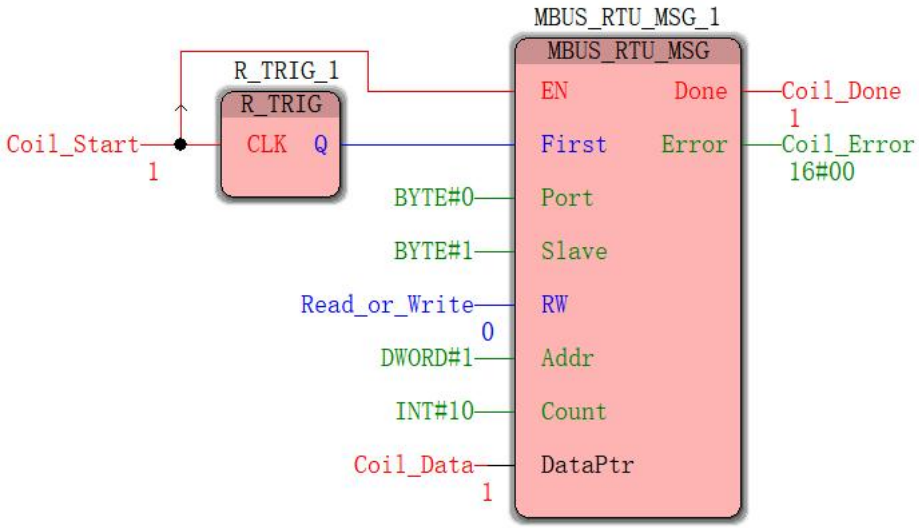


图 1-3-7

操作成功，查看变量表，相关联的变量均置为 True，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Coil												
	Coil_1	TRUE	BO...	VAR_GL...	线圈寄存器1	%MX3.0.0		□	□	□	□	□	
	Coil_2	TRUE	BO...	VAR_GL...	线圈寄存器2	%MX3.0.1		□	□	□	□	□	
	Coil_3	TRUE	BO...	VAR_GL...	线圈寄存器3	%MX3.0.2		□	□	□	□	□	
	Coil_4	TRUE	BO...	VAR_GL...	线圈寄存器4	%MX3.0.3		□	□	□	□	□	
	Coil_5	TRUE	BO...	VAR_GL...	线圈寄存器5	%MX3.0.4		□	□	□	□	□	
	Coil_6	TRUE	BO...	VAR_GL...	线圈寄存器6	%MX3.0.5		□	□	□	□	□	
	Coil_7	TRUE	BO...	VAR_GL...	线圈寄存器7	%MX3.0.6		□	□	□	□	□	
	Coil_8	TRUE	BO...	VAR_GL...	线圈寄存器8	%MX3.0.7		□	□	□	□	□	
	Coil_9	TRUE	BO...	VAR_GL...	线圈寄存器9	%MX3.1.0		□	□	□	□	□	
	Coil_10	TRUE	BO...	VAR_GL...	线圈寄存器10	%MX3.1.1		□	□	□	□	□	

图 1-3-8

2、写线圈寄存器

（1）修改变量表中的线圈寄存器关联变量的值为 False

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Coil												
	Coil_1	FALSE	BO...	VAR_GL...	线圈寄存器1	%MX3.0.0		☐	☐	☐	☐	☐	
	Coil_2	FALSE	BO...	VAR_GL...	线圈寄存器2	%MX3.0.1		☐	☐	☐	☐	☐	
	Coil_3	FALSE	BO...	VAR_GL...	线圈寄存器3	%MX3.0.2		☐	☐	☐	☐	☐	
	Coil_4	FALSE	BO...	VAR_GL...	线圈寄存器4	%MX3.0.3		☐	☐	☐	☐	☐	
	Coil_5	FALSE	BO...	VAR_GL...	线圈寄存器5	%MX3.0.4		☐	☐	☐	☐	☐	
	Coil_6	FALSE	BO...	VAR_GL...	线圈寄存器6	%MX3.0.5		☐	☐	☐	☐	☐	
	Coil_7	FALSE	BO...	VAR_GL...	线圈寄存器7	%MX3.0.6		☐	☐	☐	☐	☐	
	Coil_8	FALSE	BO...	VAR_GL...	线圈寄存器8	%MX3.0.7		☐	☐	☐	☐	☐	
	Coil_9	FALSE	BO...	VAR_GL...	线圈寄存器9	%MX3.1.0		☐	☐	☐	☐	☐	
	Coil_10	FALSE	BO...	VAR_GL...	线圈寄存器10	%MX3.1.1		☐	☐	☐	☐	☐	

图 1-3-9

（2）更改 Read_or_Write 变量为 1，将 Coil_Start 由 False 转 True 一次，结果如下；

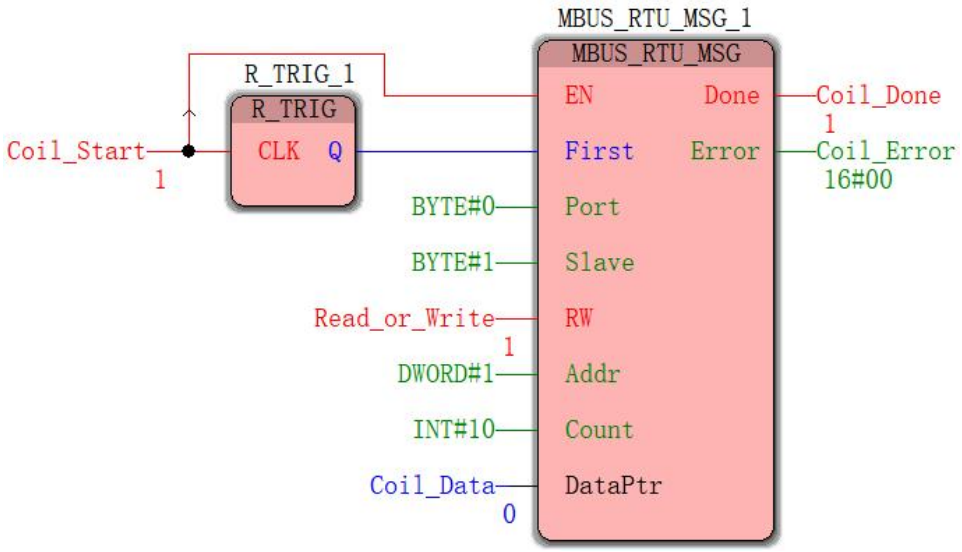


图 1-3-10

Modbus Slave 软件中表示线圈寄存器的结果如下

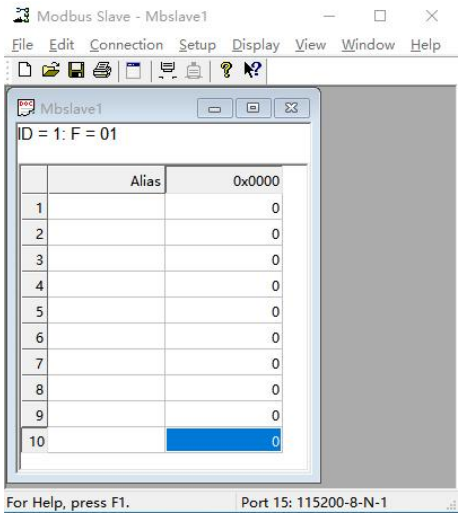


图 1-3-11

如此，读写线圈寄存器操作完成。

1.3.3.2 离散寄存器的读操作

离散寄存器操作与线圈寄存器类似，只是没有了写操作。

假设目标从站的地址为 1，对其前 10 个离散寄存器进行读操作。配置功能块如下。

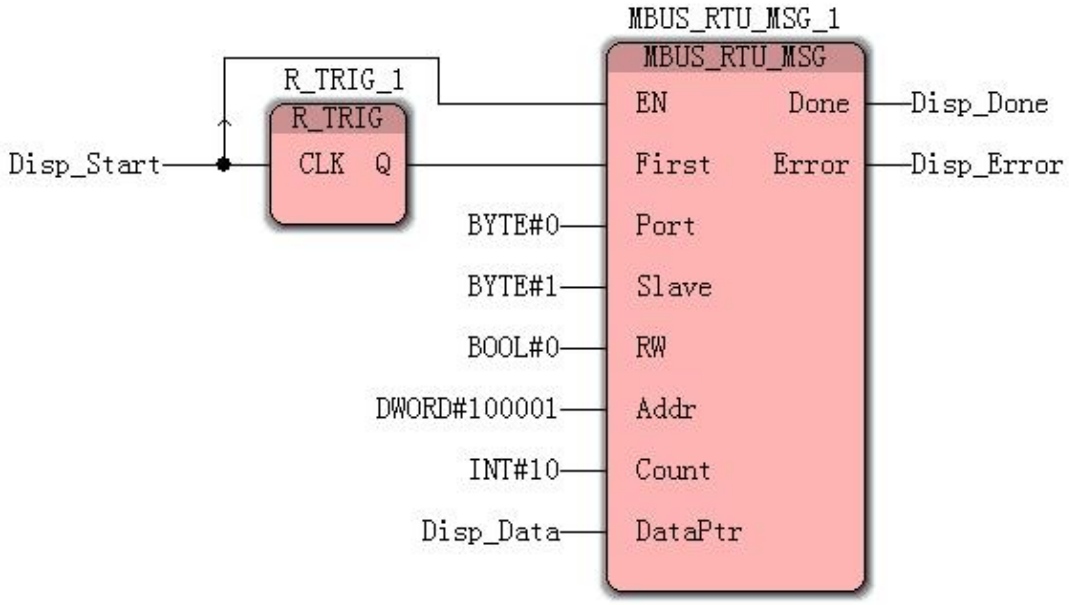


图 1-3-12

Disp_Start 用于对 modbus 离散寄存器的操作控制，其参数由 False 转 True，便会完成一次对从站的指定离散寄存器进行一次读写操作，其中 R_TRIG 功能块用于上升沿检测。

Port 传入的端口号为之前使用 MBUS_RTU_CTRL_1 配置过的端口。

Slave 传入参数即为目标从站的地址。

RW 为灵活控制读写，传入 BOOL#0，设为读操作。

Addr 传入 modbus 寄存器的地址，此处为 1.

Count 传入参数为操作寄存器的个数，即自 Addr 传入的地址，连续操作寄存器的个数，此处为自 modbus 地址 100001（离散寄存器 1）连续操作 10 个寄存器，离散寄存器 1 至离散寄存器 10。

DataPtr 为 modbus 离散寄存器 1 至 10 存放的首地址，此处选择 M3 分区，首地址选择%MX3.10.0。配置如下。

图 1-3-13

为方便观察，同样建立离散寄存器的变量表，如下图所示。

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Disp											
	Disp_1	BOOL	VAR_GL...	离散寄存器1	%MX3.10.0		☐	☐	☐	■	■	
	Disp_2	BOOL	VAR_GL...	离散寄存器2	%MX3.10.1		☐	☐	☐	■	■	
	Disp_3	BOOL	VAR_GL...	离散寄存器3	%MX3.10.2		☐	☐	☐	■	■	
	Disp_4	BOOL	VAR_GL...	离散寄存器4	%MX3.10.3		☐	☐	☐	■	■	
	Disp_5	BOOL	VAR_GL...	离散寄存器5	%MX3.10.4		☐	☐	☐	■	■	
	Disp_6	BOOL	VAR_GL...	离散寄存器6	%MX3.10.5		☐	☐	☐	■	■	
	Disp_7	BOOL	VAR_GL...	离散寄存器7	%MX3.10.6		☐	☐	☐	■	■	
	Disp_8	BOOL	VAR_GL...	离散寄存器8	%MX3.10.7		☐	☐	☐	■	■	
	Disp_9	BOOL	VAR_GL...	离散寄存器9	%MX3.11.0		☐	☐	☐	■	■	
	Disp_10	BOOL	VAR_GL...	离散寄存器10	%MX3.11.1		☐	☐	☐	■	■	

图 1-3-14

Done 与 Error 为功能块执行结果，分别传入 Disp_Done 与 Disp_Error 变量，若 Disp_Done 为 True，Disp_Error 为 0，表示此次操作成功。

1、读取线圈寄存器

（1）使用 Modbus Slave 软件，设置站地址为 1，并把 10 个离散寄存器设置如下。

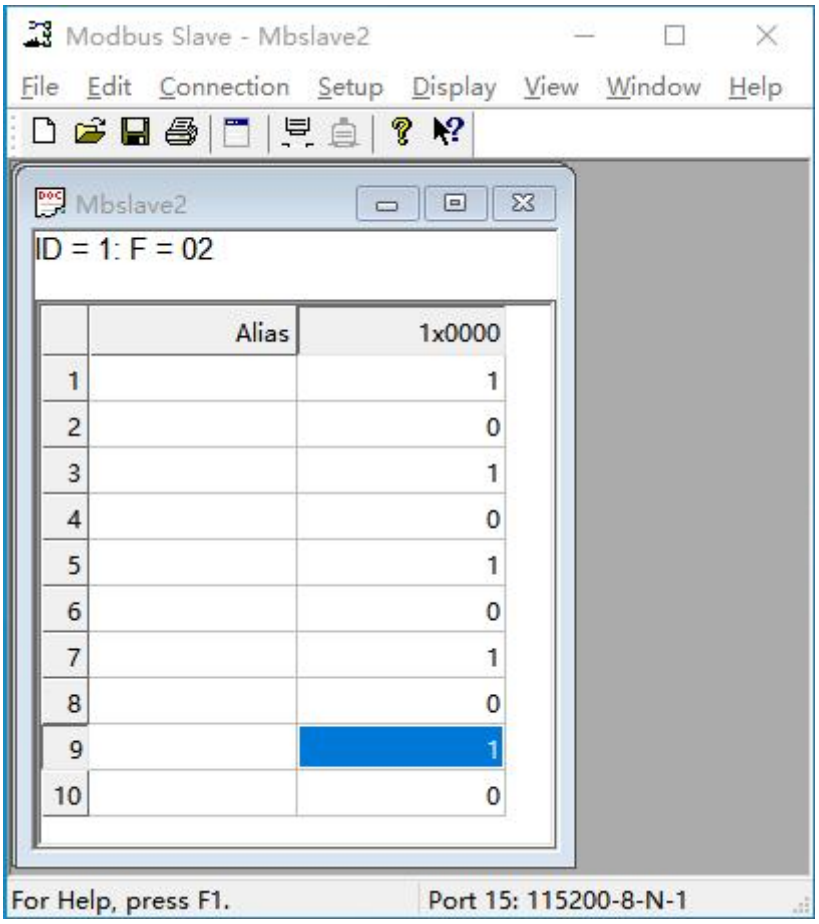


图 1-3-15

（2）使用功能块进行读操作，即 Disp_Start 由 False 转 True，如下图所示；

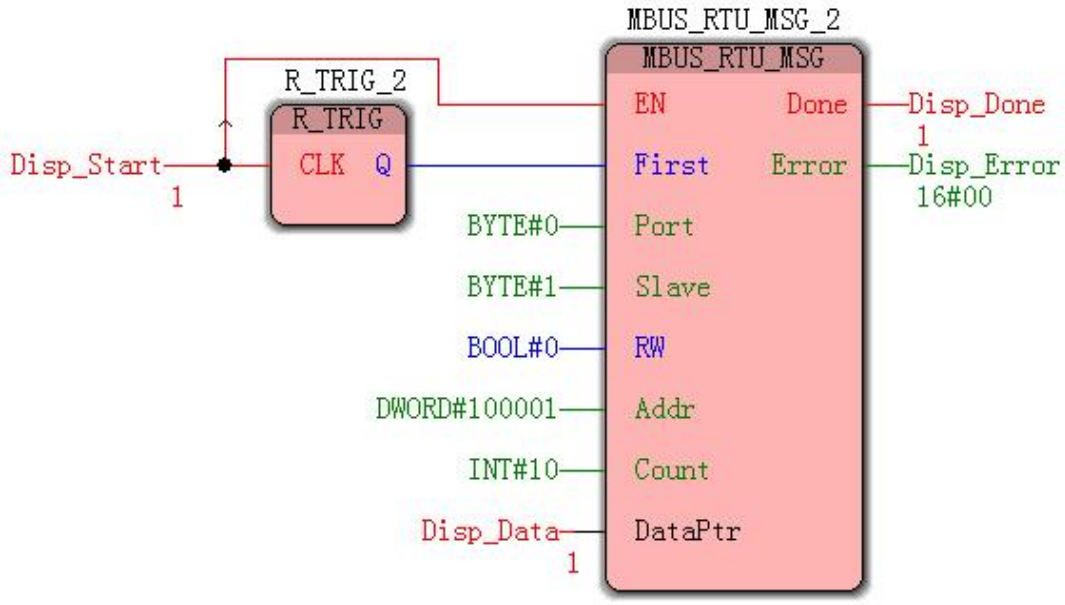


图 1-3-16

操作成功，查看变量表，相关联的变量均置为 True，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Disp												
	Disp_1	TRUE	BOOL	VAR_GL...	离散寄存器1	%MX3.10.0		□	□	□	□	□	
	Disp_2	FALSE	BOOL	VAR_GL...	离散寄存器2	%MX3.10.1		□	□	□	□	□	
	Disp_3	TRUE	BOOL	VAR_GL...	离散寄存器3	%MX3.10.2		□	□	□	□	□	
	Disp_4	FALSE	BOOL	VAR_GL...	离散寄存器4	%MX3.10.3		□	□	□	□	□	
	Disp_5	TRUE	BOOL	VAR_GL...	离散寄存器5	%MX3.10.4		□	□	□	□	□	
	Disp_6	FALSE	BOOL	VAR_GL...	离散寄存器6	%MX3.10.5		□	□	□	□	□	
	Disp_7	TRUE	BOOL	VAR_GL...	离散寄存器7	%MX3.10.6		□	□	□	□	□	
	Disp_8	FALSE	BOOL	VAR_GL...	离散寄存器8	%MX3.10.7		□	□	□	□	□	
	Disp_9	TRUE	BOOL	VAR_GL...	离散寄存器9	%MX3.11.0		□	□	□	□	□	
	Disp_10	FALSE	BOOL	VAR_GL...	离散寄存器10	%MX3.11.1		□	□	□	□	□	

图 1-3-17

结果与 Modbus Slave 软件中离散寄存器设置一致，如此，读离散寄存器操作完成。

1.3.3.2 输入寄存器的读操作

输入寄存器操作与线圈寄存器及离散寄存器操作类似，仅在映射本地地址上有所不同。同样假设目标从站的地址为 1，对其前 10 个输入寄存器进行读操作。配置功能块如下。

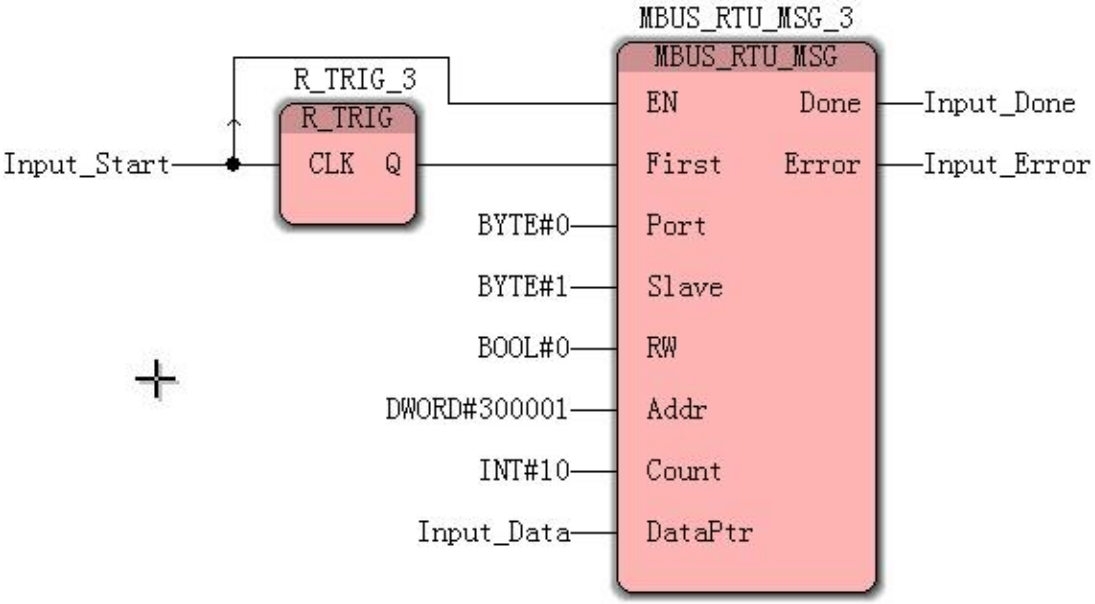


图 1-3-18

Input_Start 用于对 modbus 输入寄存器的操作控制，其参数由 False 转 True，便会完成一次对从站的指定输入寄存器进行一次读写操作，其中 R_TRIG 功能块用于上升沿检测。

Port 传入的端口号为之前使用 MBUS_RTU_CTRL_1 配置过的端口。

Slave 传入参数即为目标从站的地址。

RW 为灵活控制读写，传入 BOOL#0，设为读操作。

Addr 传入 modbus 寄存器的地址，此处为 1.

Count 传入参数为操作寄存器的个数，即自 Addr 传入的地址，连续操作寄存器的个数，此处为自 modbus 地址 300001（输入寄存器 1）连续操作 10 个寄存器，输入寄存器 1 至输入寄存器 10。

DataPtr 为 modbus 输入寄存器 1 至 10 存放的首地址，此处选择 M3 分区，首地址选择%MW3.20。配置如下。

图 1-3-19

为方便观察，同样建立离散寄存器的变量表，如下图所示。

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Input											
	Input_1	INT	VAR_GL...	输入寄存器1	%MW3.20		□	□	□	■	■	
	Input_2	INT	VAR_GL...	输入寄存器2	%MW3.22		□	□	□	■	■	
	Input_3	INT	VAR_GL...	输入寄存器3	%MW3.24		□	□	□	■	■	
	Input_4	INT	VAR_GL...	输入寄存器4	%MW3.26		□	□	□	■	■	
	Input_5	INT	VAR_GL...	输入寄存器5	%MW3.28		□	□	□	■	■	
	Input_6	INT	VAR_GL...	输入寄存器6	%MW3.30		□	□	□	■	■	
	Input_7	INT	VAR_GL...	输入寄存器7	%MW3.32		□	□	□	■	■	
	Input_8	INT	VAR_GL...	输入寄存器8	%MW3.34		□	□	□	■	■	
	Input_9	INT	VAR_GL...	输入寄存器9	%MW3.36		□	□	□	■	■	
	Input_10	INT	VAR_GL...	输入寄存器10	%MW3.38		□	□	□	■	■	

图 1-3-20

Done 与 Error 为功能块执行结果，分别传入 Input_Done 与 Input_Error 变量，若 Input_Done 为 True，Input_Error 为 0，表示此次操作成功。

1、读取输入寄存器

（1）使用 Modbus Slave 软件，设置站地址为 1，并把 10 个输入寄存器设置如下。

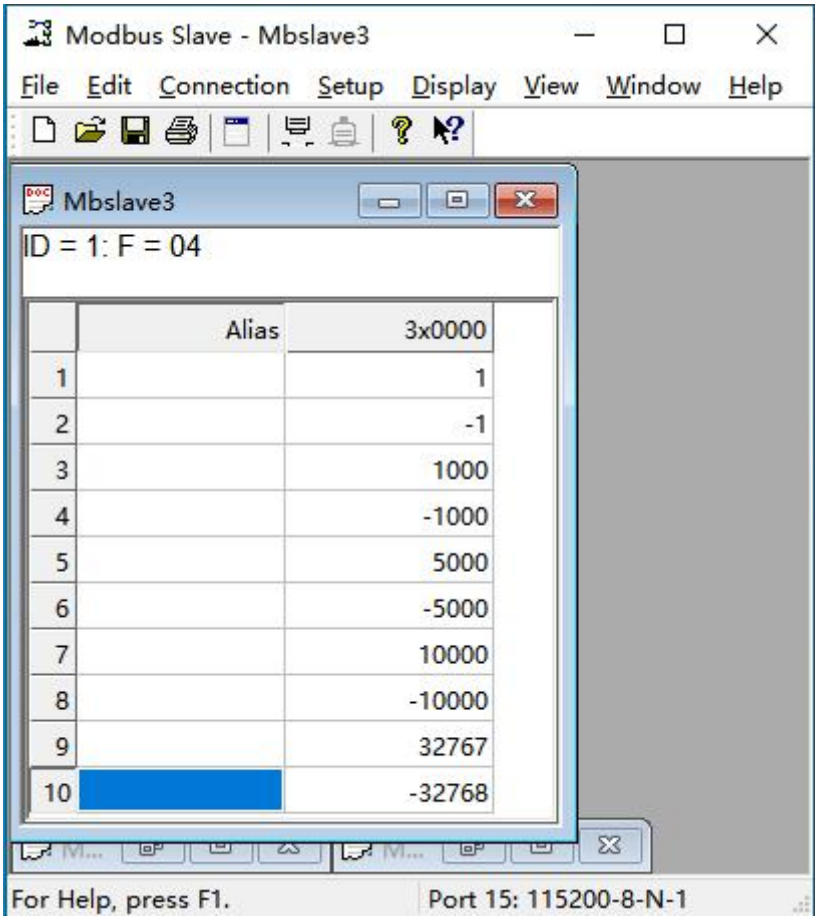


图 1-3-21

（2）使用功能块进行读操作，即 Input_Start 由 False 转 True，如下图所示；

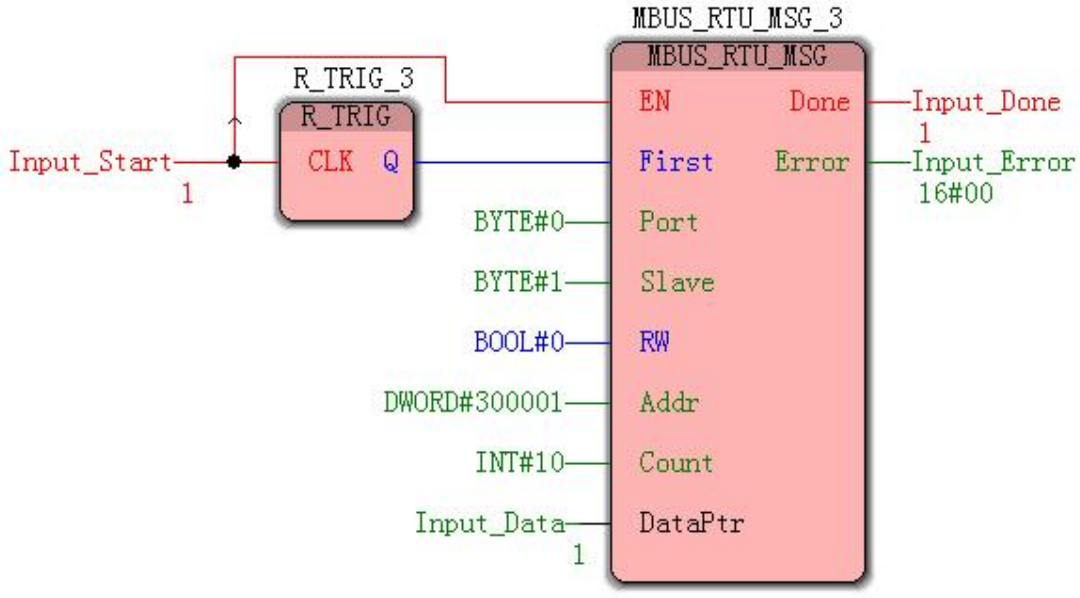


图 1-3-22

操作成功，查看变量表，相关联的变量结果，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Input												
	Input_1	1	INT	VAR_GL...	输入寄存器1	%MW3.20		□	□	□	□	□	
	Input_2	-1	INT	VAR_GL...	输入寄存器2	%MW3.22		□	□	□	□	□	
	Input_3	1000	INT	VAR_GL...	输入寄存器3	%MW3.24		□	□	□	□	□	
	Input_4	-1000	INT	VAR_GL...	输入寄存器4	%MW3.26		□	□	□	□	□	
	Input_5	5000	INT	VAR_GL...	输入寄存器5	%MW3.28		□	□	□	□	□	
	Input_6	-5000	INT	VAR_GL...	输入寄存器6	%MW3.30		□	□	□	□	□	
	Input_7	10000	INT	VAR_GL...	输入寄存器7	%MW3.32		□	□	□	□	□	
	Input_8	-10000	INT	VAR_GL...	输入寄存器8	%MW3.34		□	□	□	□	□	
	Input_9	32767	INT	VAR_GL...	输入寄存器9	%MW3.36		□	□	□	□	□	
	Input_10	-32768	INT	VAR_GL...	输入寄存器10	%MW3.38		□	□	□	□	□	

图 1-3-23

结果与 Modbus Slave 软件中输入寄存器设置一致，如此，读输入寄存器操作完成。

1.3.3.4 保持寄存器的读写操作

假设目标从站的地址为 1，对其前 10 个保持寄存器进行读写。配置功能块如下。

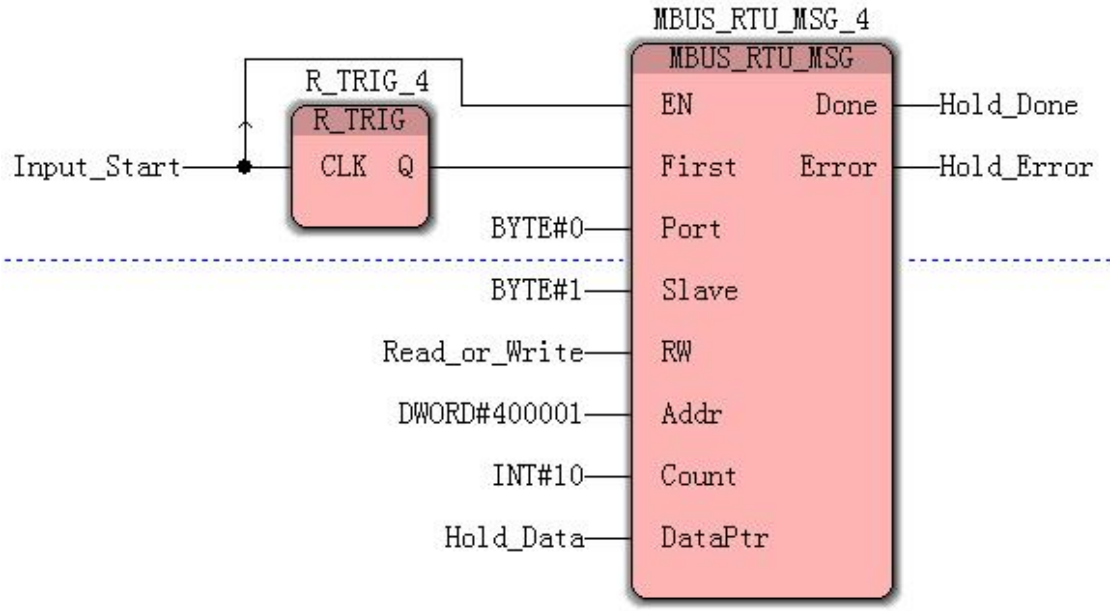


图 1-3-24

Coil_Start 用于对 modbus 保持寄存器的操作控制，其参数由 False 转 True，便会完成一次对从站的指定保持寄存器进行一次读写操作，其中 R_TRIG 功能块用于上升沿检测。

Port 传入的端口号为之前使用 MBUS_RTU_CTRL_1 配置过的端口。

Slave 传入参数即为目标从站的地址。

RW 为灵活控制读写，传入 Read_or_Write 变量中的参数，0 为读操作，1 为写操作。

Addr 传入 modbus 寄存器的地址，此处为 1.

Count 传入参数为操作寄存器的个数，即自 Addr 传入的地址，连续操作寄存器的个数，此处为自 modbus 地址 1（保持寄存器 1）连续操作 10 个寄存器，保持寄存器 1 至保持寄存器 10。

DataPtr 为 modbus 保持寄存器 1 至 10 存放的首地址，此处选择自 M3 分区，首地址为%MW3.40。配置如下。

图 1-3-25

为方便观察，建立线圈寄存器的变量表，如下图所示。

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Hold											
	Hold_1	INT	VAR_GL...	保持寄存器1	%MW3.40		☐	☐	☐	■	■	
	Hold_2	INT	VAR_GL...	保持寄存器2	%MW3.42		☐	☐	☐	■	■	
	Hold_3	INT	VAR_GL...	保持寄存器3	%MW3.44		☐	☐	☐	■	■	
	Hold_4	INT	VAR_GL...	保持寄存器4	%MW3.46		☐	☐	☐	■	■	
	Hold_5	INT	VAR_GL...	保持寄存器5	%MW3.48		☐	☐	☐	■	■	
	Hold_6	INT	VAR_GL...	保持寄存器6	%MW3.50		☐	☐	☐	■	■	
	Hold_7	INT	VAR_GL...	保持寄存器7	%MW3.52		☐	☐	☐	■	■	
	Hold_8	INT	VAR_GL...	保持寄存器8	%MW3.54		☐	☐	☐	■	■	
	Hold_9	INT	VAR_GL...	保持寄存器9	%MW3.56		☐	☐	☐	■	■	
	Hold_10	INT	VAR_GL...	保持寄存器10	%MW3.58		☐	☐	☐	■	■	

图 1-3-26

Done 与 Error 为功能块执行结果，分别传入 Input_Done 与 Input_Error 变量，若 Input_Done 为 True，Input_Error 为 0，表示此次操作成功。

1、读取保持寄存器

（1）使用 Modbus Slave 软件，设置站地址为 1，并把 10 个保持寄存器设置如下。

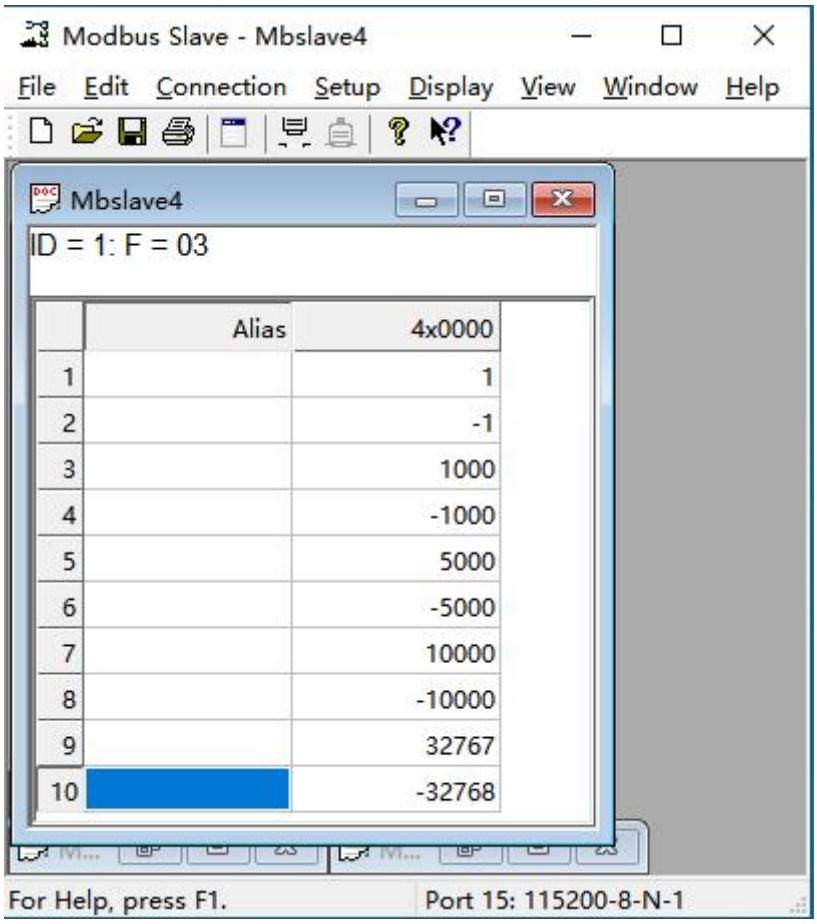


图 1-3-27

（2）使用功能块进行读操作，即 Input_Start 由 False 转 True，如下图所示；

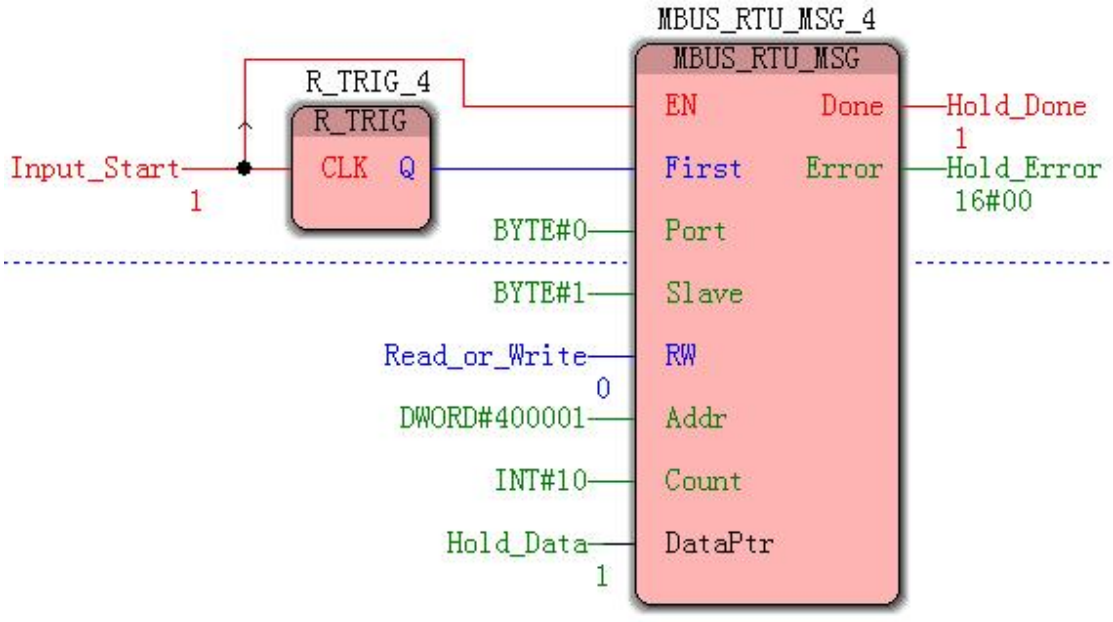


图 1-3-28

操作成功，查看变量表，相关联的变量均置为 True，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Hold												
	Hold_1	1	INT	VAR_GL...	保持寄存器1	%MW3.40		☐	☐	☐	☐	☐	
	Hold_2	-1	INT	VAR_GL...	保持寄存器2	%MW3.42		☐	☐	☐	☐	☐	
	Hold_3	1000	INT	VAR_GL...	保持寄存器3	%MW3.44		☐	☐	☐	☐	☐	
	Hold_4	-1000	INT	VAR_GL...	保持寄存器4	%MW3.46		☐	☐	☐	☐	☐	
	Hold_5	5000	INT	VAR_GL...	保持寄存器5	%MW3.48		☐	☐	☐	☐	☐	
	Hold_6	-5000	INT	VAR_GL...	保持寄存器6	%MW3.50		☐	☐	☐	☐	☐	
	Hold_7	10000	INT	VAR_GL...	保持寄存器7	%MW3.52		☐	☐	☐	☐	☐	
	Hold_8	-10000	INT	VAR_GL...	保持寄存器8	%MW3.54		☐	☐	☐	☐	☐	
	Hold_9	32767	INT	VAR_GL...	保持寄存器9	%MW3.56		☐	☐	☐	☐	☐	
	Hold_10	-32768	INT	VAR_GL...	保持寄存器10	%MW3.58		☐	☐	☐	☐	☐	

图 1-3-29

2、写线圈寄存器

（1）修改变量表中的保持寄存器关联变量的值为 0

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Hold												
	Hold_1	0	INT	VAR_GL...	保持寄存器1	%MW3.40		☐	☐	☐	☐	☐	
	Hold_2	0	INT	VAR_GL...	保持寄存器2	%MW3.42		☐	☐	☐	☐	☐	
	Hold_3	0	INT	VAR_GL...	保持寄存器3	%MW3.44		☐	☐	☐	☐	☐	
	Hold_4	0	INT	VAR_GL...	保持寄存器4	%MW3.46		☐	☐	☐	☐	☐	
	Hold_5	0	INT	VAR_GL...	保持寄存器5	%MW3.48		☐	☐	☐	☐	☐	
	Hold_6	0	INT	VAR_GL...	保持寄存器6	%MW3.50		☐	☐	☐	☐	☐	
	Hold_7	0	INT	VAR_GL...	保持寄存器7	%MW3.52		☐	☐	☐	☐	☐	
	Hold_8	0	INT	VAR_GL...	保持寄存器8	%MW3.54		☐	☐	☐	☐	☐	
	Hold_9	0	INT	VAR_GL...	保持寄存器9	%MW3.56		☐	☐	☐	☐	☐	
	Hold_10	0	INT	VAR_GL...	保持寄存器10	%MW3.58		☐	☐	☐	☐	☐	

图 1-3-30

（2）更改 Read_or_Write 变量为 1，将 Input_Start 由 False 转 True 一次，结果如下；

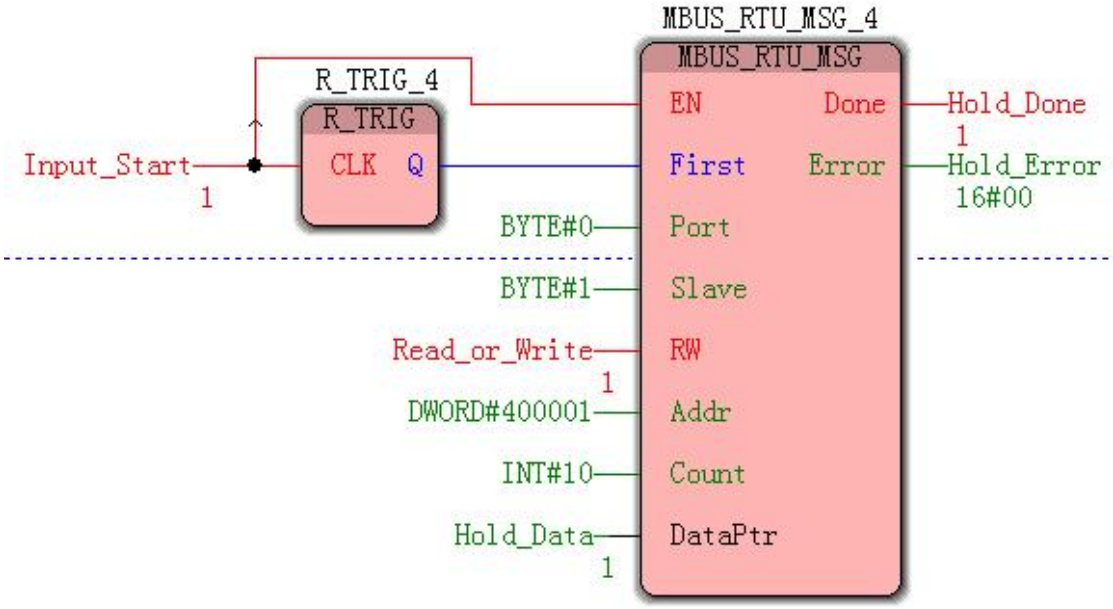


图 1-3-31

Modbus Slave 软件中表示保持寄存器的结果如下，结果全部被设置为 0。

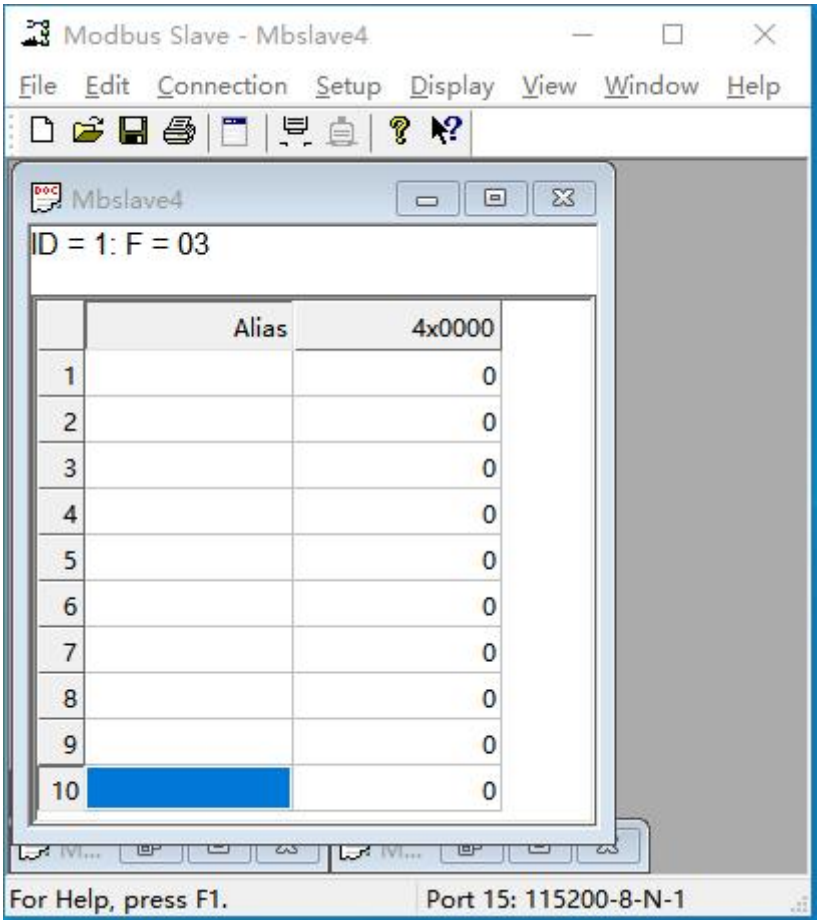


图 1-3-32

如此，读写保持寄存器操作完成。

1.3.3.5 寄存器操作总结

对于 modbus RTU 通讯所需的扩展端口配置，仅需使用 MBUS_RTU_CTRL 功能块配置一次即可，但要保证 modbus 通讯期间功能块一直使能，使用同一种 MBUS_RTU_MSG 功能块，进行不同的配置，便可完成对于具体 modbus 寄存器的操作。

1、完整工程如下：

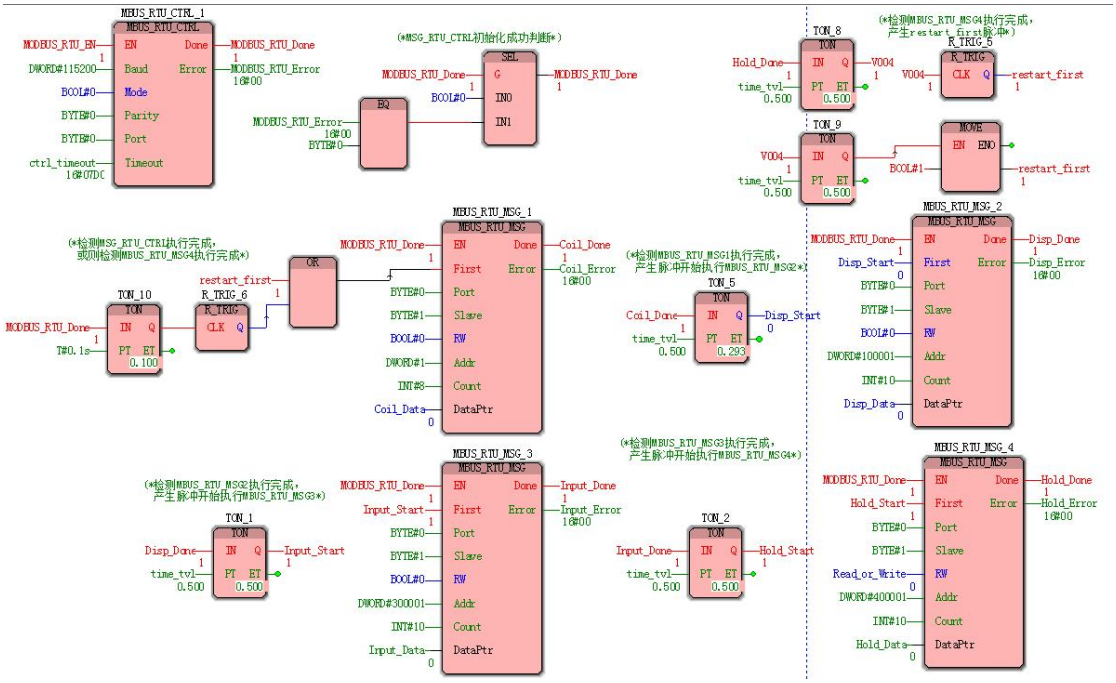


图 1-3-33

2、modbus 参数变量表如下：

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Hold											
	Hold_1	INT	VAR_GL...	保持寄存器1	%MW3.40		☐	☐	☐	☐	☐	
	Hold_2	INT	VAR_GL...	保持寄存器2	%MW3.42		☐	☐	☐	☐	☐	
	Hold_3	INT	VAR_GL...	保持寄存器3	%MW3.44		☐	☐	☐	☐	☐	
	Hold_4	INT	VAR_GL...	保持寄存器4	%MW3.46		☐	☐	☐	☐	☐	
	Hold_5	INT	VAR_GL...	保持寄存器5	%MW3.48		☐	☐	☐	☐	☐	
	Hold_6	INT	VAR_GL...	保持寄存器6	%MW3.50		☐	☐	☐	☐	☐	
	Hold_7	INT	VAR_GL...	保持寄存器7	%MW3.52		☐	☐	☐	☐	☐	
	Hold_8	INT	VAR_GL...	保持寄存器8	%MW3.54		☐	☐	☐	☐	☐	
	Hold_9	INT	VAR_GL...	保持寄存器9	%MW3.56		☐	☐	☐	☐	☐	
	Hold_10	INT	VAR_GL...	保持寄存器10	%MW3.58		☐	☐	☐	☐	☐	
	Input											
	Input_1	INT	VAR_GL...	输入寄存器1	%MW3.20		☐	☐	☐	☐	☐	
	Input_2	INT	VAR_GL...	输入寄存器2	%MW3.22		☐	☐	☐	☐	☐	
	Input_3	INT	VAR_GL...	输入寄存器3	%MW3.24		☐	☐	☐	☐	☐	
	Input_4	INT	VAR_GL...	输入寄存器4	%MW3.26		☐	☐	☐	☐	☐	
	Input_5	INT	VAR_GL...	输入寄存器5	%MW3.28		☐	☐	☐	☐	☐	
	Input_6	INT	VAR_GL...	输入寄存器6	%MW3.30		☐	☐	☐	☐	☐	
	Input_7	INT	VAR_GL...	输入寄存器7	%MW3.32		☐	☐	☐	☐	☐	
	Input_8	INT	VAR_GL...	输入寄存器8	%MW3.34		☐	☐	☐	☐	☐	
	Input_9	INT	VAR_GL...	输入寄存器9	%MW3.36		☐	☐	☐	☐	☐	
	Input_10	INT	VAR_GL...	输入寄存器10	%MW3.38		☐	☐	☐	☐	☐	
	Disp											
	Disp_1	BOOL	VAR_GL...	离散寄存器1	%MX3.10.0		☐	☐	☐	☐	☐	
	Disp_2	BOOL	VAR_GL...	离散寄存器2	%MX3.10.1		☐	☐	☐	☐	☐	
	Disp_3	BOOL	VAR_GL...	离散寄存器3	%MX3.10.2		☐	☐	☐	☐	☐	
	Disp_4	BOOL	VAR_GL...	离散寄存器4	%MX3.10.3		☐	☐	☐	☐	☐	
	Disp_5	BOOL	VAR_GL...	离散寄存器5	%MX3.10.4		☐	☐	☐	☐	☐	
	Disp_6	BOOL	VAR_GL...	离散寄存器6	%MX3.10.5		☐	☐	☐	☐	☐	
	Disp_7	BOOL	VAR_GL...	离散寄存器7	%MX3.10.6		☐	☐	☐	☐	☐	
	Disp_8	BOOL	VAR_GL...	离散寄存器8	%MX3.10.7		☐	☐	☐	☐	☐	
	Disp_9	BOOL	VAR_GL...	离散寄存器9	%MX3.11.0		☐	☐	☐	☐	☐	
	Disp_10	BOOL	VAR_GL...	离散寄存器10	%MX3.11.1		☐	☐	☐	☐	☐	
	Coil											
	Coil_1	BOOL	VAR_GL...	线圈寄存器1	%MX3.0.0		☐	☐	☐	☐	☐	
	Coil_2	BOOL	VAR_GL...	线圈寄存器2	%MX3.0.1		☐	☐	☐	☐	☐	
	Coil_3	BOOL	VAR_GL...	线圈寄存器3	%MX3.0.2		☐	☐	☐	☐	☐	
	Coil_4	BOOL	VAR_GL...	线圈寄存器4	%MX3.0.3		☐	☐	☐	☐	☐	
	Coil_5	BOOL	VAR_GL...	线圈寄存器5	%MX3.0.4		☐	☐	☐	☐	☐	
	Coil_6	BOOL	VAR_GL...	线圈寄存器6	%MX3.0.5		☐	☐	☐	☐	☐	
	Coil_7	BOOL	VAR_GL...	线圈寄存器7	%MX3.0.6		☐	☐	☐	☐	☐	
	Coil_8	BOOL	VAR_GL...	线圈寄存器8	%MX3.0.7		☐	☐	☐	☐	☐	
	Coil_9	BOOL	VAR_GL...	线圈寄存器9	%MX3.1.0		☐	☐	☐	☐	☐	
	Coil_10	BOOL	VAR_GL...	线圈寄存器10	%MX3.1.1		☐	☐	☐	☐	☐	

图 1-3-34

1.4 Modbus RTU 从站

1.4.1 确定通讯要求

1、modbus rtu 通讯需要使用 485 进行通讯，故选择扩展端口中的 COM0 或 COM1，本次选择 COM0 进用于 modbus rtu 通讯。

2、根据总线物理通讯速率与校验方式确定端口的波特率、校验位，此处选择 115200 波特率、无校验的通讯方式。

3、确定本机的 modbus 地址、modbus 寄存器使用状况及存放本地数据位置，本次演示选择地址 1，开启 80 个线圈寄存器、80 个离散寄存器、10 个输入寄存器、10 个保持寄存器。

1.4.2 配置物理端口及 modbus 参数

确定 modbus rtu 通讯要求后，直接在工程中使用 MBUS_RTU_INIT 功能块对 modbus rtu 通讯所使用的扩展端口配置即可。如下图所示。

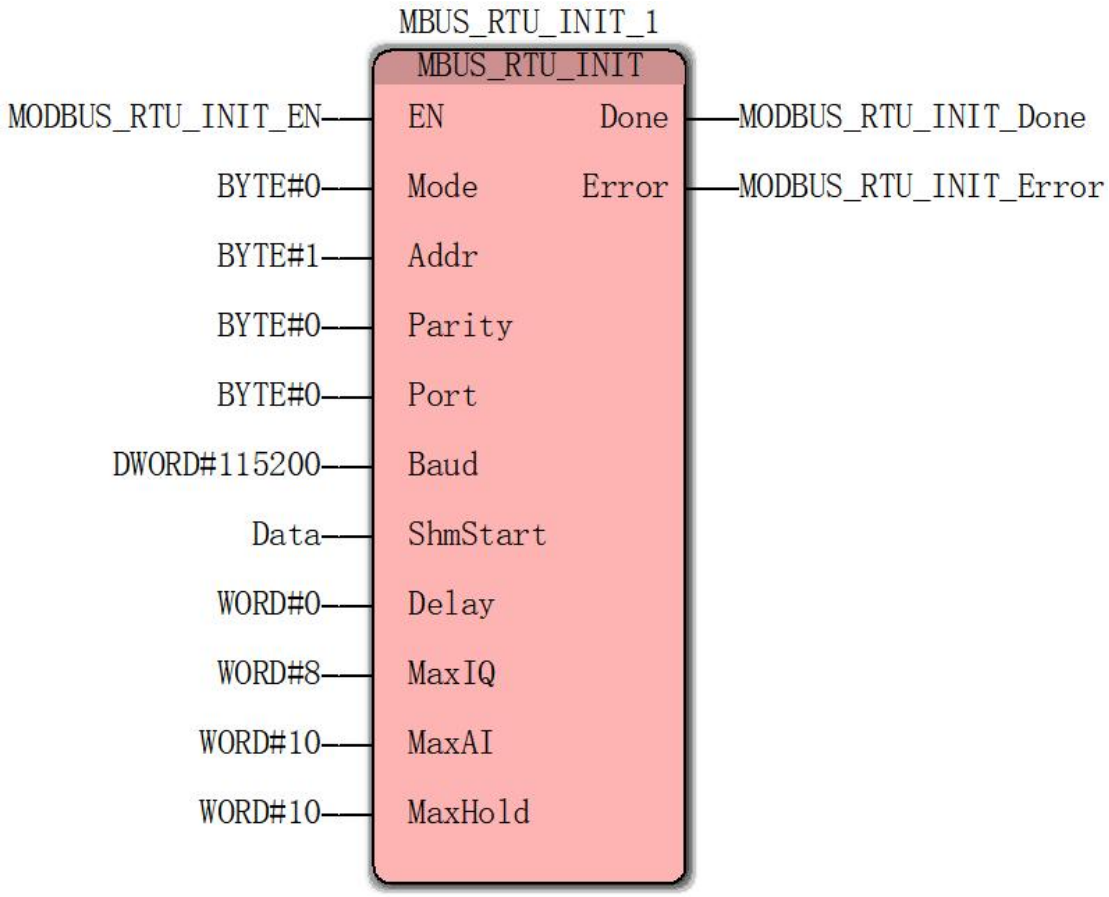


图 1-4-1

EN 位连接 MODBUS_RTU_INIT_EN，用于 MBUS_RTU_INIT_1 功能块的使能与禁用的控制，置为 True，方能够配置扩展端口 COM0 作为 modbus rtu 通讯并开启相关 modbus 寄存器，置为 False，扩展端口 COM0 上的 modbus rtu 通讯功能禁用。

Mode 传入参数 BOOL#0，Mode 用于设置功能块模式，当前为保留选项，该参数设置为 0 即可；

Addr 传入参数为本机地址，即 modbus 从站地址，本次使用地址 1；

Port 传入参数 BYTE#0，即表示使用扩展端口 0。0 为 COM0，1 为 COM1；

Parity 传入参数 BYTE#0，即表示设置校验位为无校验。0 为无校验，1 为奇校验，2 为偶校验；

Baud 传入参数 DWORD#115200，即设置波特率为 115200；

Data 传入本地地址，用以映射 modbus 寄存器数据，本次选择自%MX3.0.0 开始存放；

图 1-4-2

Delay 传入参数 WORD#0，即表示回应 modbus rtu 请求时不延时，即收即发；

MaxIQ 传入参数 WORD#8，表示开启 8 个线圈寄存器与 8 个离散寄存器；

MaxAI 传入参数 WORD#10，表示开启 10 个输入寄存器；

MaxHold 传入参数 WORD#10，表示开启 10 个保持寄存器；

Done 为配置结果输出位，结果传入变量 MODBUS_RTU_INIT_Done。

Error 为配置错误信息输出，结果传入变量 MODBUS_RTU_INIT_Error。

以上仅为 **1.4.1 确定通讯要求** 参数下的配置，对 MBUS_RTU_INIT 功能块的使用详见功能块手册第三章 3.1 章节。

工程运行后 MODBUS_RTU_INIT_EN 置为 True，之后 MODBUS_RTU_INIT_Done 结果为 True，MODBUS_RTU_INIT_Error 结果为 0，即表示使用 COM0 并配置 modbus rtu 从站成功。如下图所示。

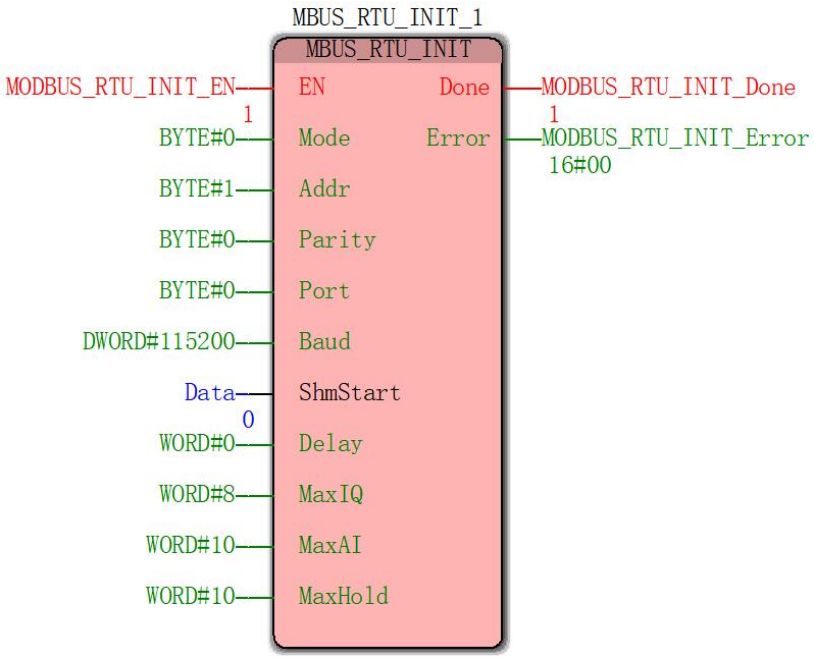


图 1-4-3

1.4.3 开启通讯

modbus 从站配置完成后，通讯功能并未完全开启，主站还不能对从站进行操作，还需要使用 MBUS_RTU_SLAVE 功能块开启从站的 modbus 通讯功能。配置如下：

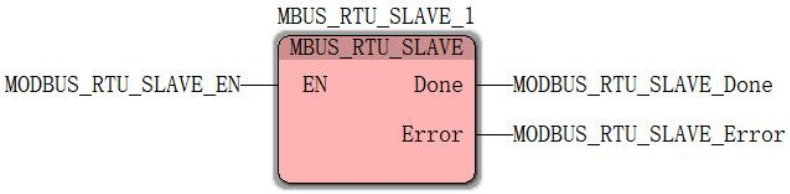


图 1-4-4

将 MBUS_RTU_SLAVE_1 功能块的 EN 位使能，若 MODBUS_RTU_Slave_Done 结果为 True，MODBUS_RTU_Slave_Error 结果为 0，即表示 modbus rtu 从站功能开启并被主站连接成功。

从站开启通讯

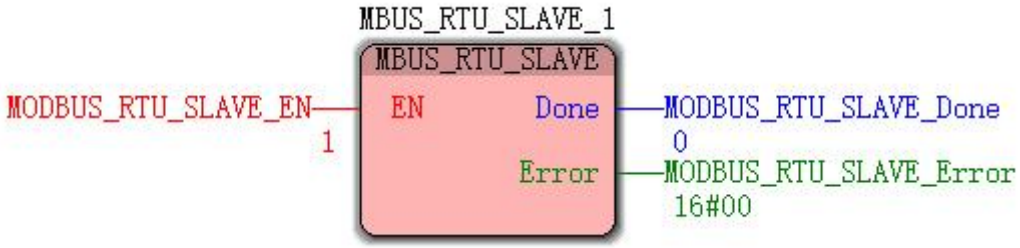


图 1-4-5

被主站连接

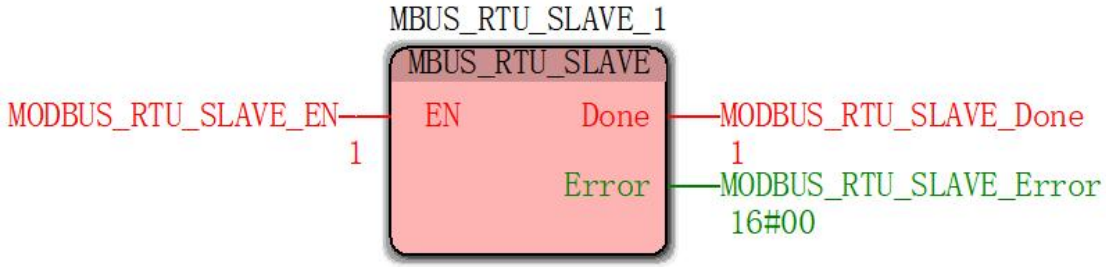


图 1-4-6

1.4.4 建立 modbus 变量表

根据 modbus rtu 从站的配置，对使用的 modbus 寄存器建立对应的变量表，此操作用于数据展示，实际使用仅对 M3 分区地址操作即可。

modbus 数据与本地地址映射（针对本次配置），表 1-4-1

modbus 寄存器类型	modbus 寄存器地址	modbus 寄存器个数	本地地址（M3 区）
线圈寄存器	1-8	8	%MX3.0.0
离散寄存器	100001-100008	8	%MX3.1.0
输入寄存器	300001-300010	10	%MW3.2

保持寄存器	400001-400010	10	%MW3.22

变量表如下：

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Coil											
	Coil_1	BOOL	VAR_GL...	线圈寄存器1	%MX3.0.0		☐	☐	☐	☐	☐	
	Coil_2	BOOL	VAR_GL...	线圈寄存器2	%MX3.0.1		☐	☐	☐	☐	☐	
	Coil_3	BOOL	VAR_GL...	线圈寄存器3	%MX3.0.2		☐	☐	☐	☐	☐	
	Coil_4	BOOL	VAR_GL...	线圈寄存器4	%MX3.0.3		☐	☐	☐	☐	☐	
	Coil_5	BOOL	VAR_GL...	线圈寄存器5	%MX3.0.4		☐	☐	☐	☐	☐	
	Coil_6	BOOL	VAR_GL...	线圈寄存器6	%MX3.0.5		☐	☐	☐	☐	☐	
	Coil_7	BOOL	VAR_GL...	线圈寄存器7	%MX3.0.6		☐	☐	☐	☐	☐	
	Coil_8	BOOL	VAR_GL...	线圈寄存器8	%MX3.0.7		☐	☐	☐	☐	☐	
	⊟ Disp											
	Disp_1	BOOL	VAR_GL...	离散寄存器1	%MX3.1.0		☐	☐	☐	☐	☐	
	Disp_2	BOOL	VAR_GL...	离散寄存器2	%MX3.1.1		☐	☐	☐	☐	☐	
	Disp_3	BOOL	VAR_GL...	离散寄存器3	%MX3.1.2		☐	☐	☐	☐	☐	
	Disp_4	BOOL	VAR_GL...	离散寄存器4	%MX3.1.3		☐	☐	☐	☐	☐	
	Disp_5	BOOL	VAR_GL...	离散寄存器5	%MX3.1.4		☐	☐	☐	☐	☐	
	Disp_6	BOOL	VAR_GL...	离散寄存器6	%MX3.1.5		☐	☐	☐	☐	☐	
	Disp_7	BOOL	VAR_GL...	离散寄存器7	%MX3.1.6		☐	☐	☐	☐	☐	
	Disp_8	BOOL	VAR_GL...	离散寄存器8	%MX3.1.7		☐	☐	☐	☐	☐	
	⊟ Input											
	Input_1	INT	VAR_GL...	输入寄存器1	%MW3.2		☐	☐	☐	☐	☐	
	Input_2	INT	VAR_GL...	输入寄存器2	%MW3.4		☐	☐	☐	☐	☐	
	Input_3	INT	VAR_GL...	输入寄存器3	%MW3.6		☐	☐	☐	☐	☐	
	Input_4	INT	VAR_GL...	输入寄存器4	%MW3.8		☐	☐	☐	☐	☐	
	Input_5	INT	VAR_GL...	输入寄存器5	%MW3.10		☐	☐	☐	☐	☐	
	Input_6	INT	VAR_GL...	输入寄存器6	%MW3.12		☐	☐	☐	☐	☐	
	Input_7	INT	VAR_GL...	输入寄存器7	%MW3.14		☐	☐	☐	☐	☐	
	Input_8	INT	VAR_GL...	输入寄存器8	%MW3.16		☐	☐	☐	☐	☐	
	Input_9	INT	VAR_GL...	输入寄存器9	%MW3.18		☐	☐	☐	☐	☐	
	Input_10	INT	VAR_GL...	输入寄存器10	%MW3.20		☐	☐	☐	☐	☐	
	⊟ Hold											
	Hold_1	INT	VAR_GL...	保持寄存器1	%MW3.22		☐	☐	☐	☐	☐	
	Hold_2	INT	VAR_GL...	保持寄存器2	%MW3.24		☐	☐	☐	☐	☐	
	Hold_3	INT	VAR_GL...	保持寄存器3	%MW3.26		☐	☐	☐	☐	☐	
	Hold_4	INT	VAR_GL...	保持寄存器4	%MW3.28		☐	☐	☐	☐	☐	
	Hold_5	INT	VAR_GL...	保持寄存器5	%MW3.30		☐	☐	☐	☐	☐	
	Hold_6	INT	VAR_GL...	保持寄存器6	%MW3.32		☐	☐	☐	☐	☐	
	Hold_7	INT	VAR_GL...	保持寄存器7	%MW3.34		☐	☐	☐	☐	☐	
	Hold_8	INT	VAR_GL...	保持寄存器8	%MW3.36		☐	☐	☐	☐	☐	
	Hold_9	INT	VAR_GL...	保持寄存器9	%MW3.38		☐	☐	☐	☐	☐	
	Hold_10	INT	VAR_GL...	保持寄存器10	%MW3.40		☐	☐	☐	☐	☐	

图 1-4-7

1.4.5 modbus 通讯验证

使用 Modbus Poll 软件连接 P500 modbus rtu 从站设备，从而进行寄存器的读写操作，验证通讯功能。

1、读线圈寄存器

修改变量表中的 Coil_1 值，将其由 False 变为 True。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Coil												
	Coil_1	TRUE	BOOL	VAR_GL...	线圈寄存器1	%MX3.0.0		☐	☐	☐	☐	☐	

图 1-4-8

使用 modbus 功能码 01 读线圈寄存器，观察 Modbus Poll 软件中线圈寄存器 1 的值，发现线圈寄存器 1 的值由 0 变为 1，证明读线圈寄存器成功。如下图所示。

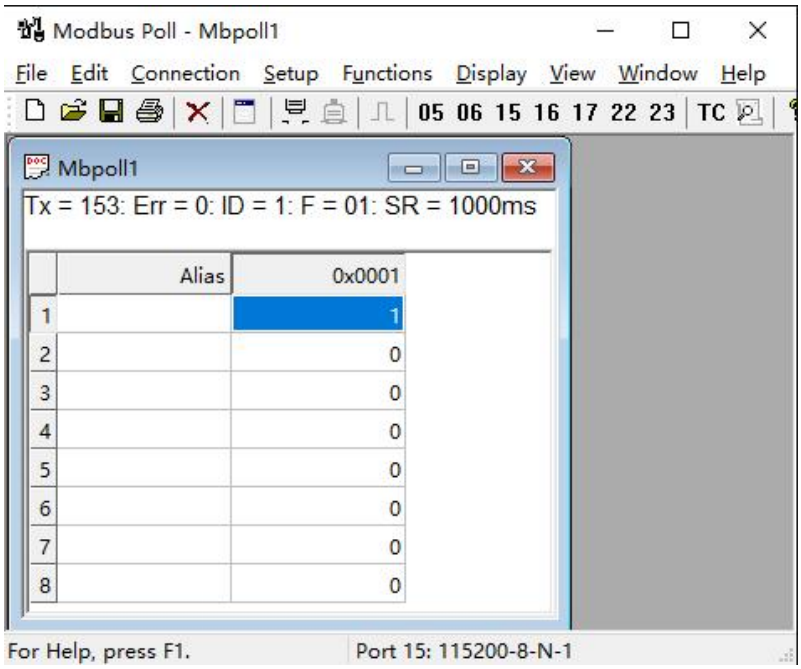


图 1-4-9

2、写单个线圈寄存器

修改 Modbus Poll 软件中线圈寄存器 1 的值，由 1 变为 0，使用 modbus 功能码 05 写单个线圈寄存器，如下图所示。

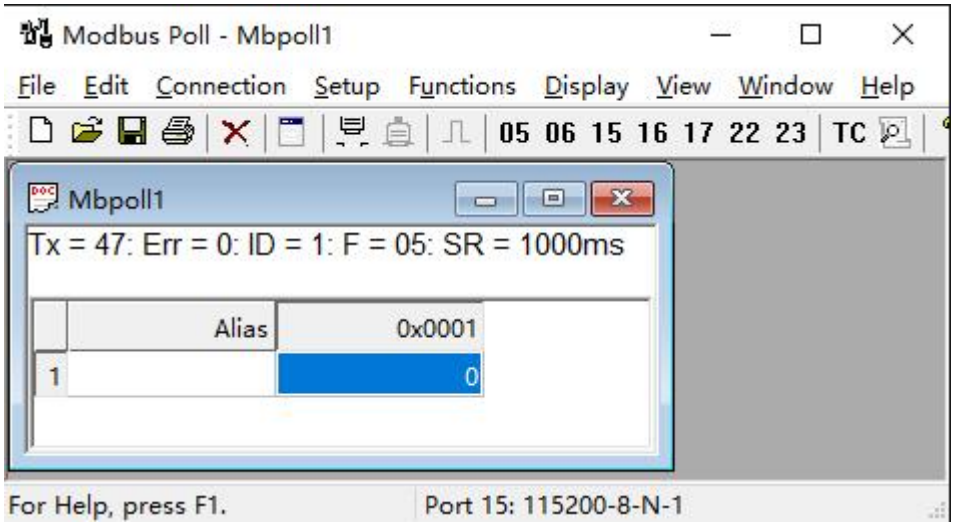


图 1-4-10

观察变量表中的 Coil_1 结果发生改变，由 True 变为 False，证明写单个线圈寄存器写成功。如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Coil												
	Coil_1	FALSE	BOOL	VAR_GL...	线圈寄存器1	%MX3.0.0		☐	☐	☐	☐	☐	

图 1-4-11

3、写多个线圈寄存器

修改 Modbus Poll 软件中线圈寄存器 1 至线圈寄存器 8 的值，全部由 1 变为 0，使用 modbus 功能码 15 写多个线圈寄存器，如下图所示。

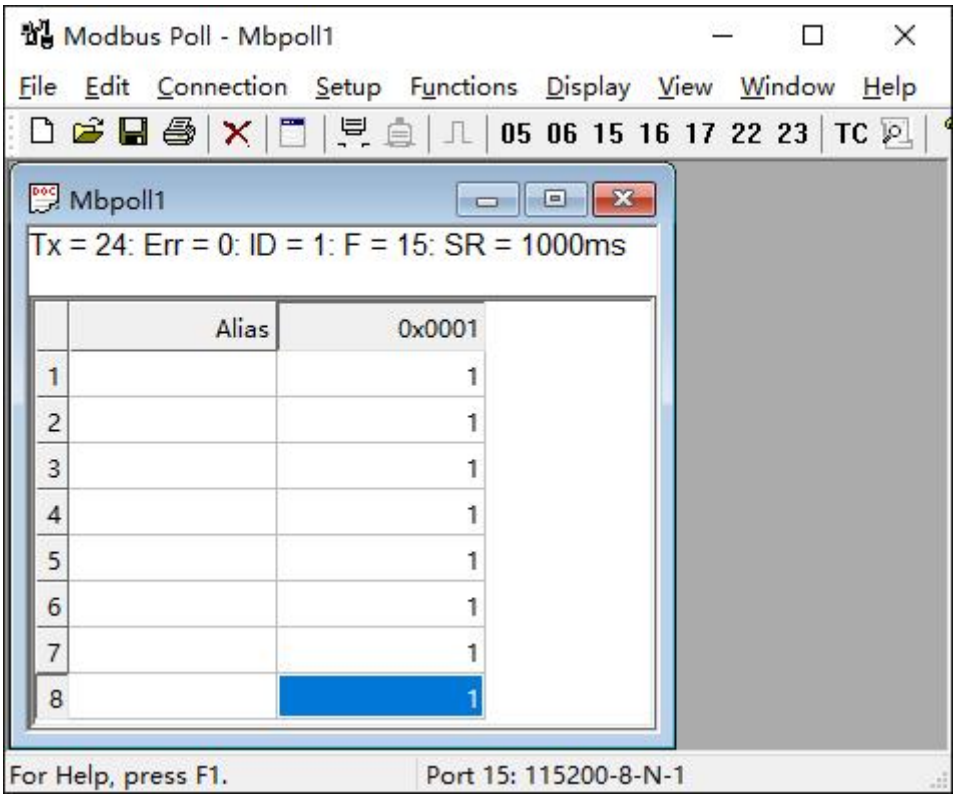


图 1-4-12

观察变量表中的 Coil_1 至 Coil_8 结果发生改变，均由 False 变为 True，证明写单多个线圈寄存器写成功，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Coil												
	Coil_1	TRUE	BOOL	VAR_GL...	线圈寄存器1	%MX3.0.0		□	□	□	□	□	
	Coil_2	TRUE	BOOL	VAR_GL...	线圈寄存器2	%MX3.0.1		□	□	□	□	□	
	Coil_3	TRUE	BOOL	VAR_GL...	线圈寄存器3	%MX3.0.2		□	□	□	□	□	
	Coil_4	TRUE	BOOL	VAR_GL...	线圈寄存器4	%MX3.0.3		□	□	□	□	□	
	Coil_5	TRUE	BOOL	VAR_GL...	线圈寄存器5	%MX3.0.4		□	□	□	□	□	
	Coil_6	TRUE	BOOL	VAR_GL...	线圈寄存器6	%MX3.0.5		□	□	□	□	□	
	Coil_7	TRUE	BOOL	VAR_GL...	线圈寄存器7	%MX3.0.6		□	□	□	□	□	
	Coil_8	TRUE	BOOL	VAR_GL...	线圈寄存器8	%MX3.0.7		□	□	□	□	□	

图 1-4-13

4、读离散寄存器

修改变量表中的 Disp_1 至 Disp_8 值，展示如下。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Disp												
	Disp_1	TRUE	BOOL	VAR_GL...	离散寄存器1	%MX3.1.0		□	□	□	□	□	
	Disp_2	FALSE	BOOL	VAR_GL...	离散寄存器2	%MX3.1.1		□	□	□	□	□	
	Disp_3	TRUE	BOOL	VAR_GL...	离散寄存器3	%MX3.1.2		□	□	□	□	□	
	Disp_4	FALSE	BOOL	VAR_GL...	离散寄存器4	%MX3.1.3		□	□	□	□	□	
	Disp_5	TRUE	BOOL	VAR_GL...	离散寄存器5	%MX3.1.4		□	□	□	□	□	
	Disp_6	FALSE	BOOL	VAR_GL...	离散寄存器6	%MX3.1.5		□	□	□	□	□	
	Disp_7	TRUE	BOOL	VAR_GL...	离散寄存器7	%MX3.1.6		□	□	□	□	□	
	Disp_8	FALSE	BOOL	VAR_GL...	离散寄存器8	%MX3.1.7		□	□	□	□	□	

图 1-4-14

使用 modbus 功能码 02 读离散寄存器，观察 Modbus Poll 软件中离散寄存器 1 至离散寄存器 8 的值，如下图所示。

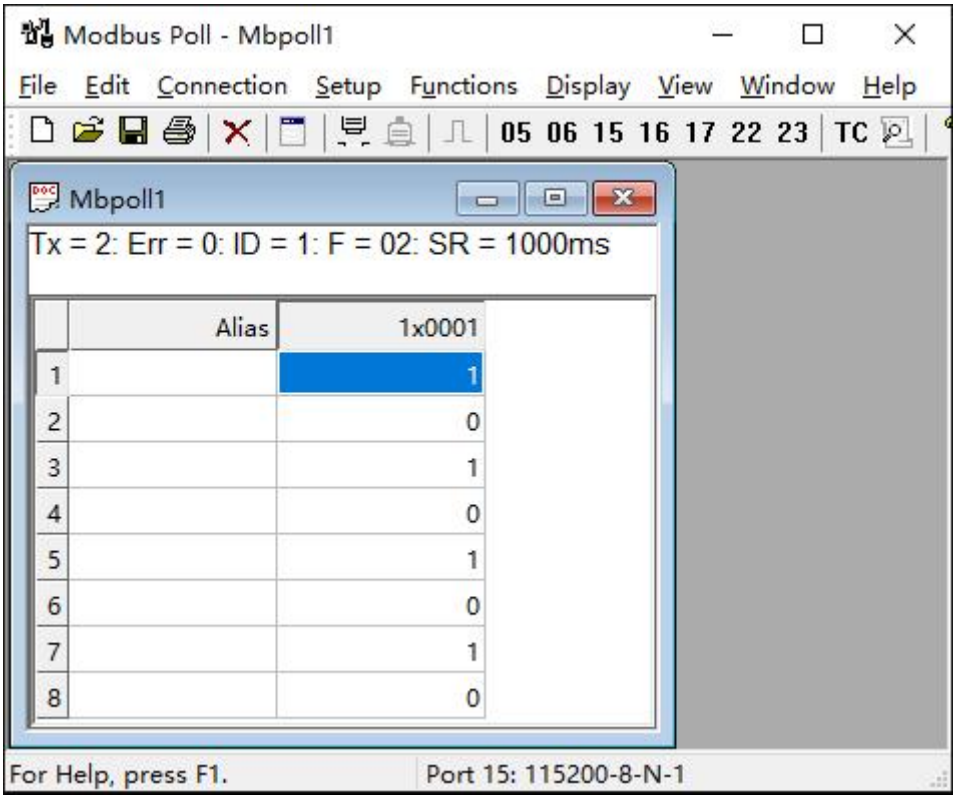


图 1-4-15

发现离散寄存器 1 至离散寄存器 8 的值与变量表一致，证明读离散寄存器成功。

5、读输入寄存器

修改变量表中的 Input_1 至 Input_10 值，展示如下。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Input												
	Input_1	1	INT	VAR_GL...	输入寄存器1	%MW3.2		☐	☐	☐	☐	☐	
	Input_2	-1	INT	VAR_GL...	输入寄存器2	%MW3.4		☐	☐	☐	☐	☐	
	Input_3	1000	INT	VAR_GL...	输入寄存器3	%MW3.6		☐	☐	☐	☐	☐	
	Input_4	-1000	INT	VAR_GL...	输入寄存器4	%MW3.8		☐	☐	☐	☐	☐	
	Input_5	0	INT	VAR_GL...	输入寄存器5	%MW3.10		☐	☐	☐	☐	☐	
	Input_6	0	INT	VAR_GL...	输入寄存器6	%MW3.12		☐	☐	☐	☐	☐	
	Input_7	10000	INT	VAR_GL...	输入寄存器7	%MW3.14		☐	☐	☐	☐	☐	
	Input_8	-10000	INT	VAR_GL...	输入寄存器8	%MW3.16		☐	☐	☐	☐	☐	
	Input_9	32767	INT	VAR_GL...	输入寄存器9	%MW3.18		☐	☐	☐	☐	☐	
	Input_10	-32768	INT	VAR_GL...	输入寄存器10	%MW3.20		☐	☐	☐	☐	☐	

图 1-4-16

使用 modbus 功能码 04 读输入寄存器，观察 Modbus Poll 软件中输入寄存器 1 至输入寄存器 10 的值，发现输入寄存器 1 至输入寄存器 10 的值与变量表一致，证明读输入寄存器成功。如下图所示。

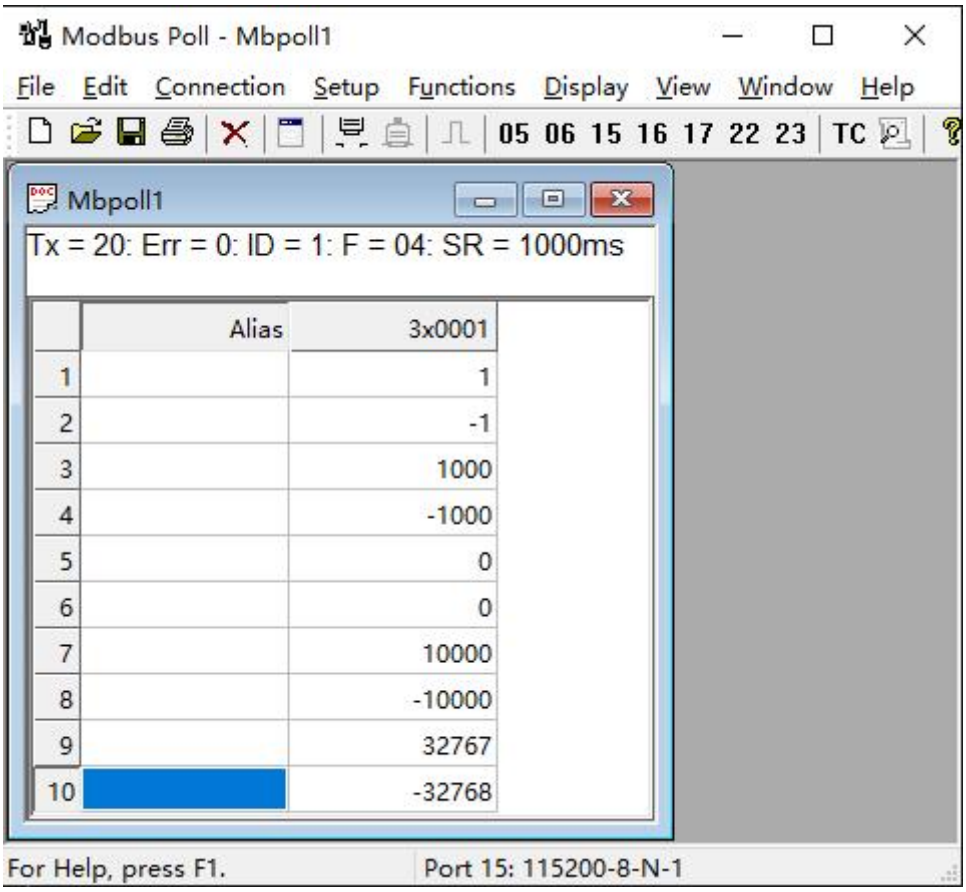


图 1-4-17

6、读保持寄存器

修改变量表中的 Hold_1 至 Hold_10 值，展示如下。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Hold												
	Hold_1	1	INT	VAR_GL...	保持寄存器1	%MW3.22		☐	☐	☐	☐	☐	
	Hold_2	2	INT	VAR_GL...	保持寄存器2	%MW3.24		☐	☐	☐	☐	☐	
	Hold_3	3	INT	VAR_GL...	保持寄存器3	%MW3.26		☐	☐	☐	☐	☐	
	Hold_4	4	INT	VAR_GL...	保持寄存器4	%MW3.28		☐	☐	☐	☐	☐	
	Hold_5	5	INT	VAR_GL...	保持寄存器5	%MW3.30		☐	☐	☐	☐	☐	
	Hold_6	6	INT	VAR_GL...	保持寄存器6	%MW3.32		☐	☐	☐	☐	☐	
	Hold_7	7	INT	VAR_GL...	保持寄存器7	%MW3.34		☐	☐	☐	☐	☐	
	Hold_8	8	INT	VAR_GL...	保持寄存器8	%MW3.36		☐	☐	☐	☐	☐	
	Hold_9	9	INT	VAR_GL...	保持寄存器9	%MW3.38		☐	☐	☐	☐	☐	
	Hold_10	10	INT	VAR_GL...	保持寄存器10	%MW3.40		☐	☐	☐	☐	☐	

图 1-4-18

使用 modbus 功能码 03 读保持寄存器，观察 Modbus Poll 软件中保持寄存器 1 至保持寄存器 10 的值，发现保持寄存器 1 至保持寄存器 10 的值与变量表一致，证明读保持寄存器成功。如下图所示。

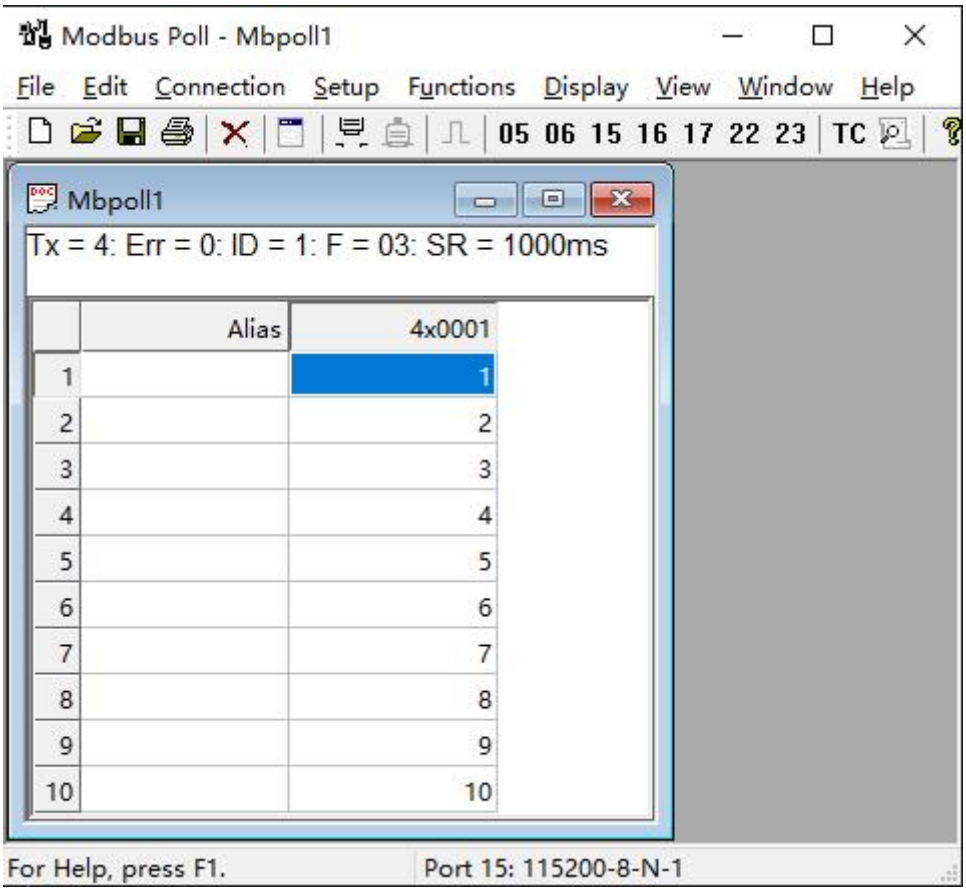


图 1-4-19

7、写单个保持寄存器

修改 Modbus Poll 软件中保持寄存器 1 的值为 32767，使用 modbus 功能码 06 写单个保持寄存器，如下图所示

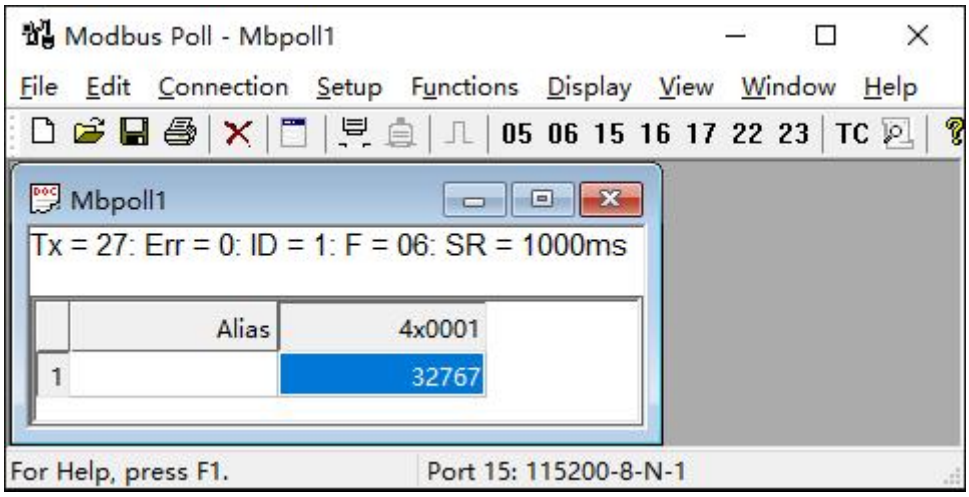


图 1-4-20

观察变量表中的 Hold_1 结果变为 32768，证明写单个保持寄存器写成功，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ Hold												
	Hold_1	32767	INT	VAR_GL...	保持寄存器1	%MW3.22		☐	☐	☐	☐	☐	

图 1-4-21

8、写多个保持寄存器

修改 Modbus Poll 软件中保持寄存器 1 至保持寄存器 10 的值，使用 modbus 功能码 16 写多个保持寄存器，如下图所示

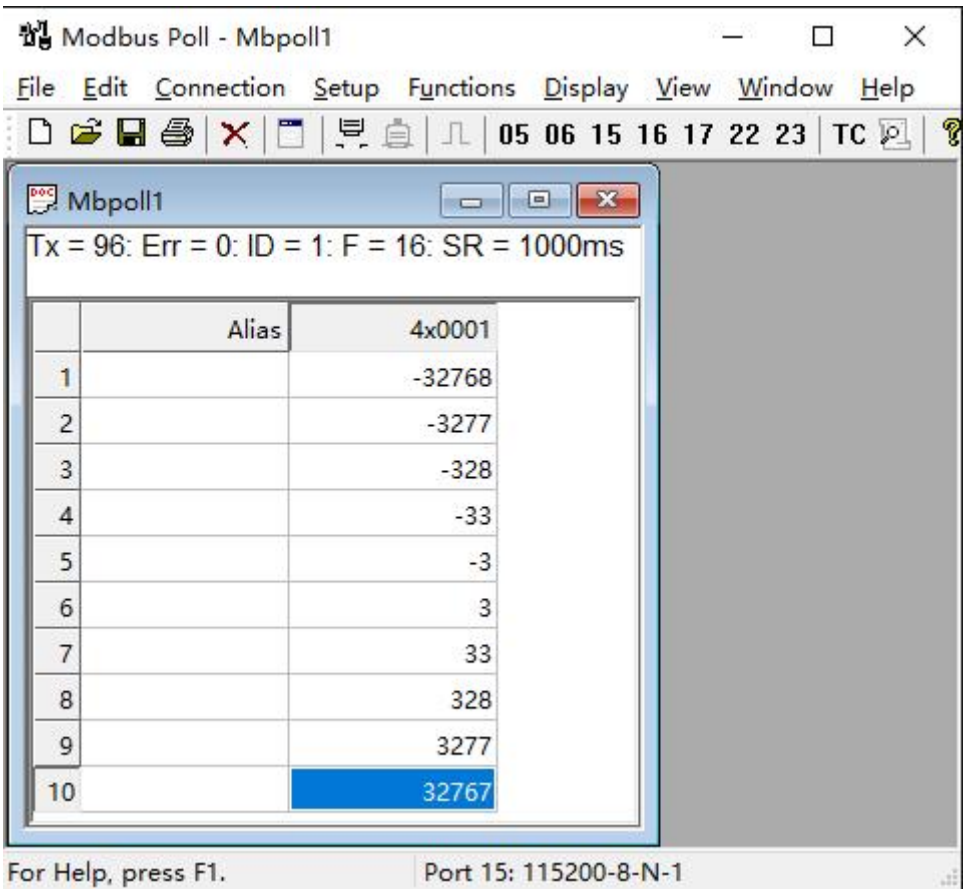


图 1-4-22

观察变量表中的 Hold_1 至 Hold_10 结果发生变化，证明写多个保持寄存器写成功。如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	Hold												
	Hold_1	-32768	INT	VAR_GL...	保持寄存器1	%MW3.22		☐	☐	☐	☐	☐	
	Hold_2	-3277	INT	VAR_GL...	保持寄存器2	%MW3.24		☐	☐	☐	☐	☐	
	Hold_3	-328	INT	VAR_GL...	保持寄存器3	%MW3.26		☐	☐	☐	☐	☐	
	Hold_4	-33	INT	VAR_GL...	保持寄存器4	%MW3.28		☐	☐	☐	☐	☐	
	Hold_5	-3	INT	VAR_GL...	保持寄存器5	%MW3.30		☐	☐	☐	☐	☐	
	Hold_6	3	INT	VAR_GL...	保持寄存器6	%MW3.32		☐	☐	☐	☐	☐	
	Hold_7	33	INT	VAR_GL...	保持寄存器7	%MW3.34		☐	☐	☐	☐	☐	
	Hold_8	328	INT	VAR_GL...	保持寄存器8	%MW3.36		☐	☐	☐	☐	☐	
	Hold_9	3277	INT	VAR_GL...	保持寄存器9	%MW3.38		☐	☐	☐	☐	☐	
	Hold_10	32767	INT	VAR_GL...	保持寄存器10	%MW3.40		☐	☐	☐	☐	☐	

图 1-4-23

1.5 PU510 作为 Modbus 从站（MBS）

P500 可作为标准 Modbus 从站与第三方主站通讯，支持 ModbusTCP/RTU。

1、启用 MBS 功能，右键点击 MBS，选择启用；

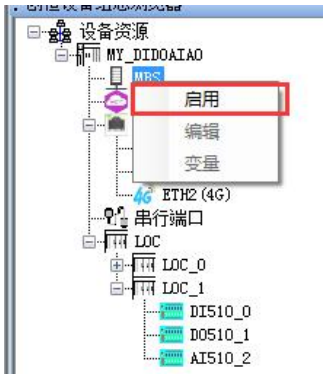


图 1-5-1

2、配置通讯参数，右键点击 MBS，选择编辑，弹出 MBS 配置界面；

图 1-5-2

从站 ID:PU510 作为从站时的 Modbus 地址 1-255；

延迟时间：响应主站延时返回时间，单位毫秒；

超时时间：判断主站连接超时时间，单位毫秒；

通信协议：Modbus TCP/RTU；

作为 Modbus TCP 从站时参数：

端口号：网络端口号；

作为 Modbus RTU 从站时参数：

串行端口：COM0-COM2，RTU 通讯端口。

波特率：1200~115200，波特率可选，默认 9600；

数据位：通讯数据位；

校验位：数据校验位：None/Odd/Even;

停止位：1、2 个停止位；

3、变量，右键点击 MBS，选择变量，弹出变量配置界面；

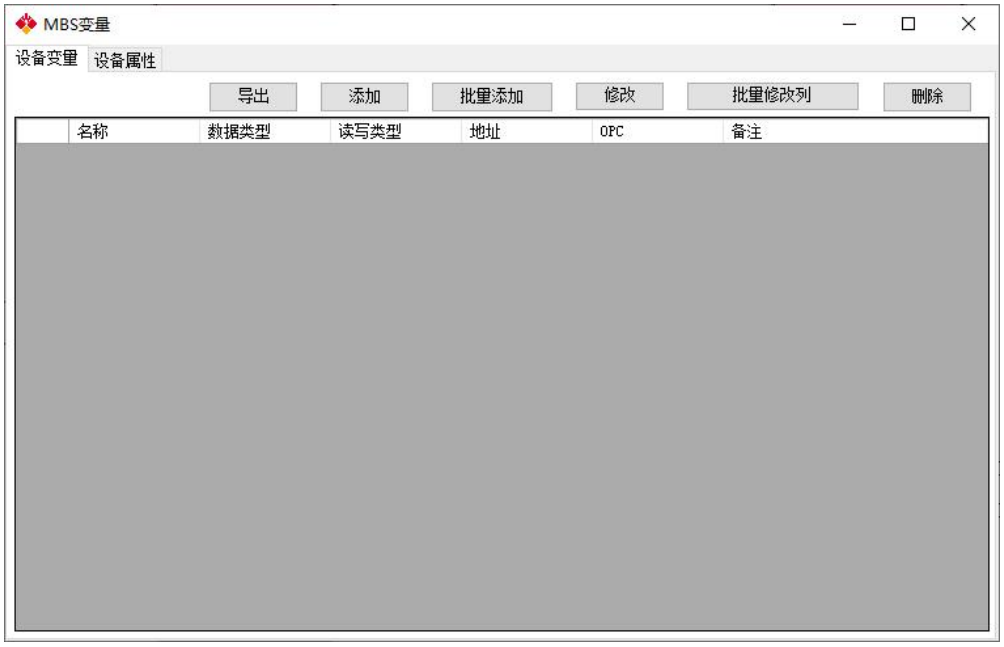


图 1-5-3

导出：可把当前配置的变量点表导出到 csv 格式表格中，以便其他应用使用或者查看点表信息。

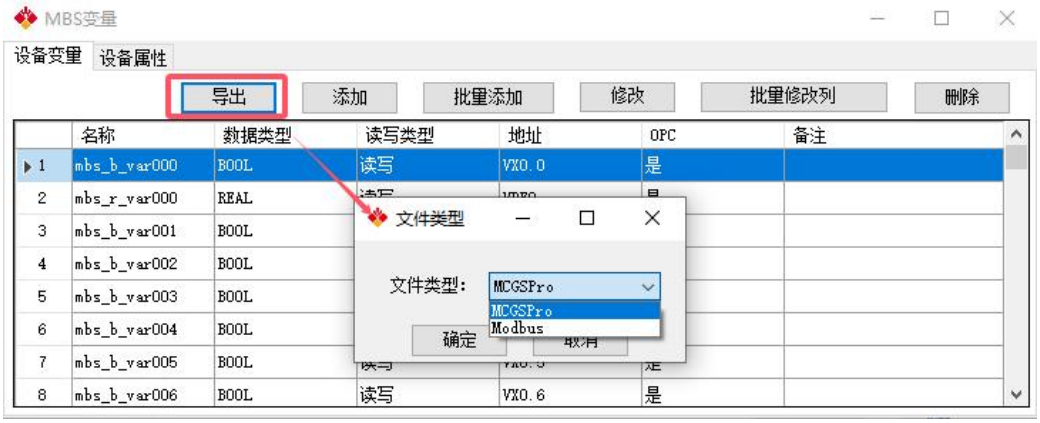


图 1-5-4

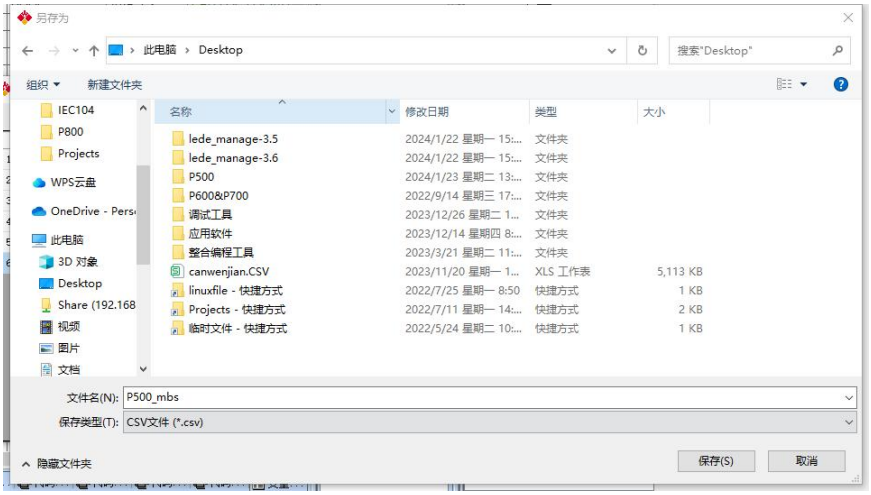
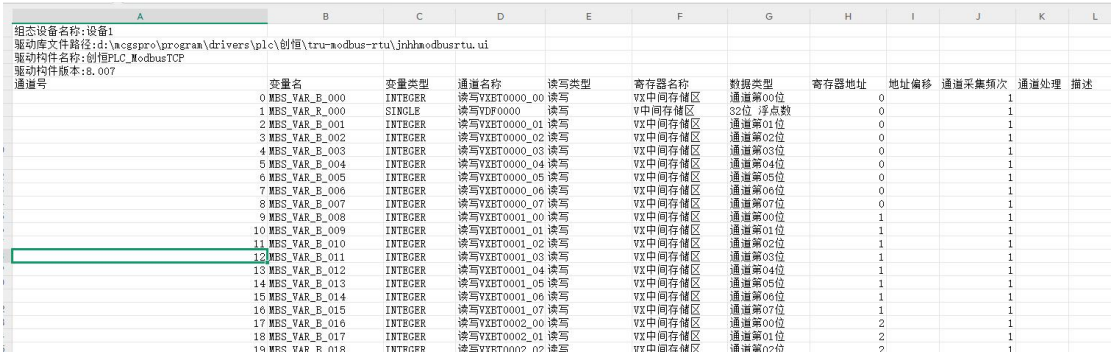


图 1-5-5

导出类型分为 MCGSPro 和 Modbus，MCGSPro 格式文件符合 MCGS_Pro 软件导入设备格式，可以在 MCGS_Pro 软件里面快速建立点表。

MCGSPro 内容格式如下：



组态设备名称:设备1
驱动库文件路径:d:\mcgspro\program\drivers\plc\创恒\tru-modbus-rtu\jnhhmodbusrtu.ui
驱动构件名称:创恒PLC_ModbusTCP
驱动构件版本:8.007

通道号	变量名	变量类型	通道名称	读写类型	寄存器名称	数据类型	寄存器地址	地址偏移	通道采集频次	通道处理	描述
0	MBS_VAR_B_000	INTEGER	读写VXBT0000_00	读写	VX中间存储区	通道第00位	0		1		
1	MBS_VAR_R_000	SINGLE	读写VDF0000	读写	V中间存储区	32位 浮点数	0		1		
2	MBS_VAR_B_001	INTEGER	读写VXBT0000_01	读写	VX中间存储区	通道第01位	0		1		
3	MBS_VAR_B_002	INTEGER	读写VXBT0000_02	读写	VX中间存储区	通道第02位	0		1		
4	MBS_VAR_B_003	INTEGER	读写VXBT0000_03	读写	VX中间存储区	通道第03位	0		1		
5	MBS_VAR_B_004	INTEGER	读写VXBT0000_04	读写	VX中间存储区	通道第04位	0		1		
6	MBS_VAR_B_005	INTEGER	读写VXBT0000_05	读写	VX中间存储区	通道第05位	0		1		
7	MBS_VAR_B_006	INTEGER	读写VXBT0000_06	读写	VX中间存储区	通道第06位	0		1		
8	MBS_VAR_B_007	INTEGER	读写VXBT0000_07	读写	VX中间存储区	通道第07位	0		1		
9	MBS_VAR_B_008	INTEGER	读写VXBT0001_00	读写	VX中间存储区	通道第00位	1		1		
10	MBS_VAR_B_009	INTEGER	读写VXBT0001_01	读写	VX中间存储区	通道第01位	1		1		
11	MBS_VAR_B_010	INTEGER	读写VXBT0001_02	读写	VX中间存储区	通道第02位	1		1		
12	MBS_VAR_B_011	INTEGER	读写VXBT0001_03	读写	VX中间存储区	通道第03位	1		1		
13	MBS_VAR_B_012	INTEGER	读写VXBT0001_04	读写	VX中间存储区	通道第04位	1		1		
14	MBS_VAR_B_013	INTEGER	读写VXBT0001_05	读写	VX中间存储区	通道第05位	1		1		
15	MBS_VAR_B_014	INTEGER	读写VXBT0001_06	读写	VX中间存储区	通道第06位	1		1		
16	MBS_VAR_B_015	INTEGER	读写VXBT0001_07	读写	VX中间存储区	通道第07位	1		1		
17	MBS_VAR_B_016	INTEGER	读写VXBT0002_00	读写	VX中间存储区	通道第00位	2		1		
18	MBS_VAR_B_017	INTEGER	读写VXBT0002_01	读写	VX中间存储区	通道第01位	2		1		
19	MBS_VAR_B_018	INTEGER	读写VXBT0002_02	读写	VX中间存储区	通道第02位	2		1		

图 1-5-6

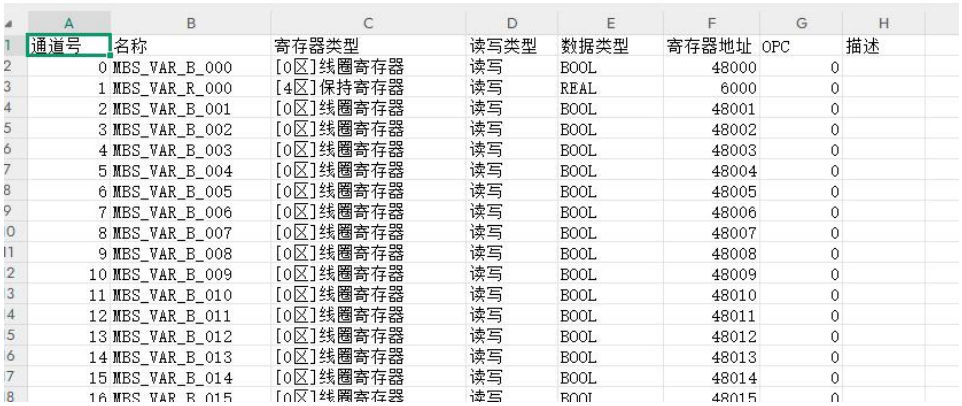
组态设备名称：需要和 MCGSPro 软件里面建立的设备名称同名，根据需要自行修改，否则会导入失败。

驱动库文件路径：需要和 MCGSPro 软件安装的创恒 MCGS 驱动路径相同，注意驱动类型 TCP/RTU，根据需要自行修改。

驱动构件名称：需要和 MCGSPro 软件里面驱动名称相同，根据需要自行修改。

驱动构件版本：需要和 MCGSPro 软件里面驱动版本相同，根据需要自行修改。

Modbus 内容格式如下：



通道号	名称	寄存器类型	读写类型	数据类型	寄存器地址	OPC	描述
0	MBS_VAR_B_000	[0区]线圈寄存器	读写	BOOL	48000	0	
1	MBS_VAR_R_000	[4区]保持寄存器	读写	REAL	6000	0	
2	MBS_VAR_B_001	[0区]线圈寄存器	读写	BOOL	48001	0	
3	MBS_VAR_B_002	[0区]线圈寄存器	读写	BOOL	48002	0	
4	MBS_VAR_B_003	[0区]线圈寄存器	读写	BOOL	48003	0	
5	MBS_VAR_B_004	[0区]线圈寄存器	读写	BOOL	48004	0	
6	MBS_VAR_B_005	[0区]线圈寄存器	读写	BOOL	48005	0	
7	MBS_VAR_B_006	[0区]线圈寄存器	读写	BOOL	48006	0	
8	MBS_VAR_B_007	[0区]线圈寄存器	读写	BOOL	48007	0	
9	MBS_VAR_B_008	[0区]线圈寄存器	读写	BOOL	48008	0	
10	MBS_VAR_B_009	[0区]线圈寄存器	读写	BOOL	48009	0	
11	MBS_VAR_B_010	[0区]线圈寄存器	读写	BOOL	48010	0	
12	MBS_VAR_B_011	[0区]线圈寄存器	读写	BOOL	48011	0	
13	MBS_VAR_B_012	[0区]线圈寄存器	读写	BOOL	48012	0	
14	MBS_VAR_B_013	[0区]线圈寄存器	读写	BOOL	48013	0	
15	MBS_VAR_B_014	[0区]线圈寄存器	读写	BOOL	48014	0	
16	MBS_VAR_B_015	[0区]线圈寄存器	读写	BOOL	48015	0	

图 1-5-7

可自行修改导出的文件内容，方便创恒其他设备导入点表。

注意：

名称：字符长度不超 26 字符；

寄存器类型：分为[0 区]线圈寄存器、[1 区]离散输入寄存器、[3 区]只读输入寄存器、[4 区]保持寄存器；

读写类型：分为只读、只写、读写；

数据类型：支持 BOOL、INT、UINT、DINT、UDINT、REAL、LREAL、WORD、DORD；

寄存器地址：为 Modbus 寄存器地址，没有偏移，起始地址为 0 地址；

OPC：是否具有 OPC 属性。

添加：添加单个变量，添加界面如下：

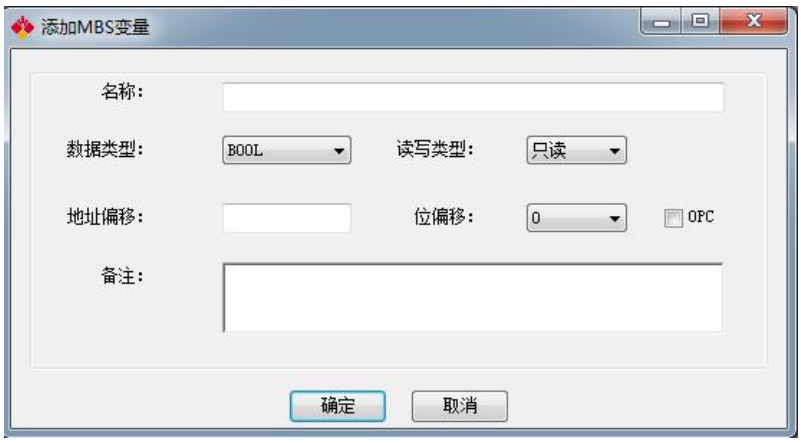


图 1-5-8

名称：变量名称，最长支持 26 个字符；

数据类型：系统基本数据类型。

读写类型：只读和读写，以主站角度定义读写类型。

OPC：是否作为 OPC 变量；

地址偏移：V 变量区寄存器地址

表 1-5-1

数据类型	地址偏移 X	位偏移 Y	最多变量个数
BOOL	0-124	0-7	1000
INT UINT WORD DINT UDINT DWORD REAL STRING	0-3998	0	1000

注意：

1、V 区所有变量累加最多 1000 个变量，BOOL 类型数据地址和非 BOOL 类型数据地址相互独立，所有非 BOOL 类型地址为公用地址，并且不可重复使用。

2、非 BOOL 变量地址偏移加上变量类型字节长度不得超过 4000；

例如 INT/UINT/WORD 变量字节长度为 2，则偏移地址最大为 3998；

DINT/UDINT/DWORD/REAL 变量字节长度为 4，则偏移地址最大为 3996；

STRING 变量字节长度为 80，则偏移地址最大为 3920.

例如：添加以下变量时地址偏移和位偏移分别为

AA:地址偏移为 0，位偏移为 0，对应 Modbus 地址为线圈区 48000；

BB:地址偏移为 0，位偏移为 1，对应 Modbus 地址为线圈区 48001；

CC:地址偏移为 0，位偏移为 2，对应 Modbus 地址为线圈区 48002；

DD:地址偏移为 0，位偏移为 0，对应 Modbus 地址为保持区 6000；

EE:地址偏移为 2，位偏移为 0，对应 Modbus 地址为保持区 6001；

FF:地址偏移为 6，位偏移为 0，对应 Modbus 地址为保持区 6003；

	名称	数据类型	读写类型	地址	OPC	备注
1	AA	BOOL	读写	VX0.0	否	
2	BB	BOOL	读写	VX0.1	否	
3	CC	BOOL	读写	VX0.2	否	
4	DD	INT	读写	VW0	否	
5	EE	REAL	读写	VDF2	否	
6	FF	INT	读写	VW6	否	

图 1-5-9

V 区变量地址与 Modbus 地址对应关系如下：

x 为地址偏移，y 为位偏移。BOOL 类型数据为线圈寄存器，其他为保持寄存器。

表 1-5-2

数据类型		Modbus 地址	通信功能码	系统显示地址
线圈（BOOL）	BOOL	x*8+y+48000	读 1，写 5/15	VXx.y
16 位 无符号二进制	UINT	x/2+6000	读 3，写 6/16	VWUx
16 位 有符号二进制	INT	x/2+6000	读 3，写 6/16	VWx
16 位 4 位 BCD	WORD	x/2+6000	读 3，写 6/16	VWDx
32 位 无符号二进制	UDINT	x/2+6000	读 3，写 6/16	VDUx
32 位 有符号二进制	DINT	x/2+6000	读 3，写 6/16	VDx
32 位 8 位 BCD	DWORD	x/2+6000	读 3，写 6/16	VDDx
32 位 浮点数	REAL	x/2+6000	读 3，写 6/16	VDFx
ASCII 字符串	STRING	x/2+6000	读 3，写 6/16	VBSx

说明：ASCII 字符串变量固定为 80 个字节长度。

I/Q 区变量（IO 模块对应通道）与 Modbus 地址对应关系如下：

x 为地址偏移，y 为位偏移。

IX 输入存储区为 DIxxx；I 输入存储区 AIxxx；QX 输出存储区为 DOxxx；Q 输出存储区 AOxxx。

I/Q 区变量不需要单独添加，通讯时按照 IO 组态时所添加的地址进行通信。

表 1-5-3：

MCGS 通道类型	数据类型		Modbus 地址	通信功能码	PLC 地址
IX 输入存储区	DI 开关量	BOOL	x*8+y	2	IXx.y
I 输入存储区	16 位 无符号二进制	UINT	x/2	4	IWx
	16 位 有符号二进制	INT	x/2	4	IWx
	16 位 4 位 BCD	WORD	x/2	4	IWx
	32 位 无符号二进制	UDINT	x/2	4	IDx
	32 位 有符号二进制	DINT	x/2	4	IDx
	32 位 8 位 BCD	DWORD	x/2	4	IDx
	32 位 浮点数	REAL	x/2	4	IDx
QX 输出存储区	DO 开关量	BOOL	x*8+y	读 1，写 5/15	QXx.y

Q 输出存储区	16 位 无符号二进制	UINT	x/2	读 3，写 6/16	QWx
	16 位 有符号二进制	INT	x/2	读 3，写 6/16	QWx
	16 位 4 位 BCD	WORD	x/2	读 3，写 6/16	QWx
	32 位 无符号二进制	UDINT	x/2	读 3，写 6/16	QDx
	32 位 有符号二进制	DINT	x/2	读 3，写 6/16	QDx
	32 位 8 位 BCD	DWORD	x/2	读 3，写 6/16	QDx
	32 位 浮点数	REAL	x/2	读 3，写 6/16	QDx

批量添加：点击批量添加按钮，弹出批量添加界面:

图 1-5-10

批量添加分为手动添加、批量生成、导入文件三种模式：

1、手动添加

在批量添加界面表格中手动填入变量各项属性，然后点击“确定”按钮，则逐条添加到全局变量表中。

图 1-5-11

2、批量生成

点击批量生成按钮，弹出批量生成界面：

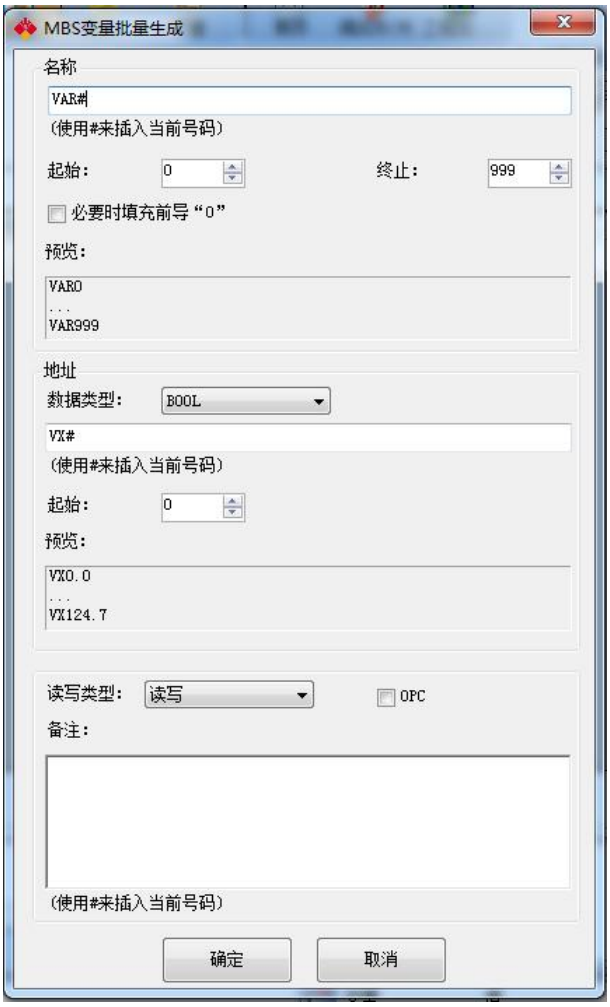


图 1-5-12

名称：变量名称，使用#来表示号码插入位置；

起始/终止：号码的起始和结束（包含结束号码）；

必要时填充前导“0”：是否在号码前填充“0”；

数据类型：所批量添加的变量数据类型；

起始：变量起始地址；

读写类型：只读/读写；

OPC：所添加变量是否具有 OPC 属性。

例如填写如下：

图 1-5-13

点击确定按钮，则在批量添加界面预生成相应的变量，可以查看和修改相应的变量属性：

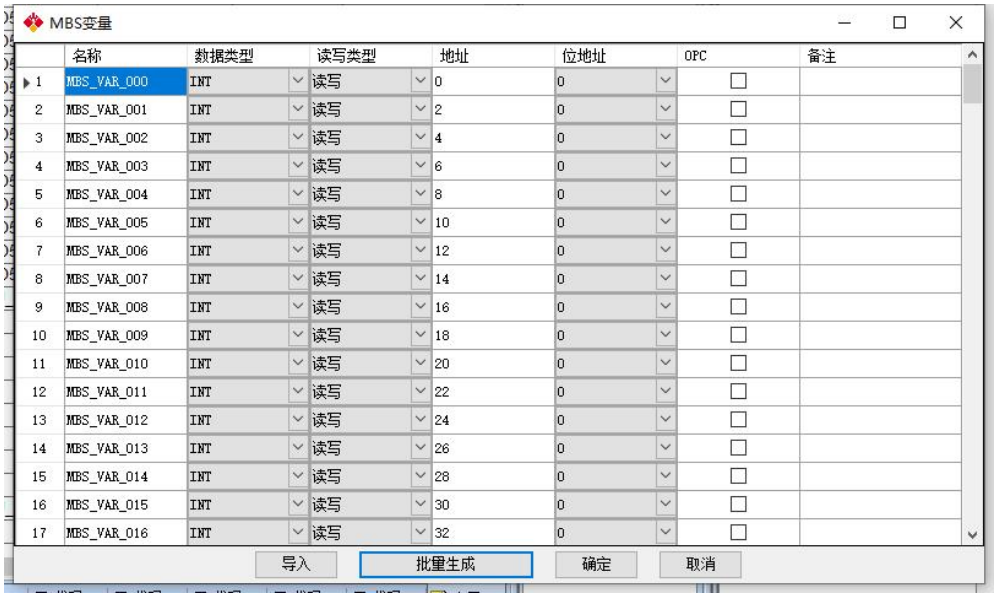


图 1-5-14

点击“确定”按钮，会自动逐条添加到全局变量表。

3、导入

点击“导入”按钮，选择导入文件类型

图 1-5-15

选择合法的变量点表文件：

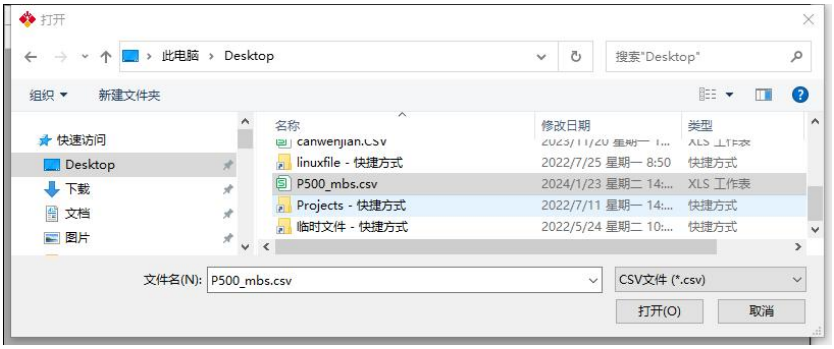


图 1-5-16

显示导入的变量点表：

图 1-5-17

并且可对点表查看修改，点击“确定”按钮，会自动逐条添加到全局变量表。

打开全局变量表如下：

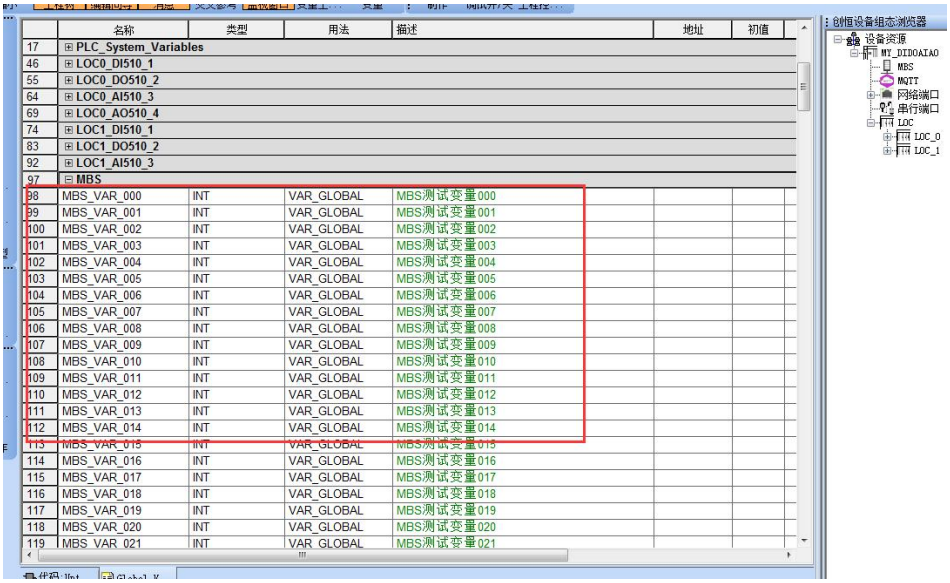


图 1-5-18

批量修改列：拖选变量后点击批量修改列，弹出批量修改界面如下：

图 1-5-19

可批量更改读写属性和 OPC 属性。

删除：拖选变量后点击删除按钮弹出确认提示界面，如下图：

图 1-5-20

设备属性：

可修改设备变量在通讯中的数据格式，方便不同字节序系统中数据的对应。

图 1-5-21

表 1-5-4

数据类型	字节序	默认字节序
16 位字节序	0 - 21	1 - 12
	1 - 12	
32 位字节序	0 - 4321	1 - 3412
	1 - 3412	
	2 - 2143	
	3 - 1234	
64 位字节序	0 - 8765_4321	1 - 7856_3412
	1 - 7856_3412	
	2 - 6587_2143	
	3 - 5678_1234	
	4 - 4321_8765	
	5 - 3412_7856	
	6 - 2143_6587	
	7 - 1234_5678	
字符串字节序	0 - 21	1 - 12
	1 - 12	
字符串编码格式	0 - GBK	0 - GBK
	1 - UTF8	
	2 - UNICODE	

1.6 MCGS 通讯连接

P500 PLC 支持 MCGS（昆仑通态触摸屏）驱动，可通过简单的配置进行通讯。

1.6.1 Multiprog 变量添加

1、启用 MBS 功能，右键点击 MBS，选择启用；

图 1-6-1

2、配置通讯参数，右键点击 MBS，选择编辑，弹出 MBS 配置界面；

图 1-6-2

从站 ID:PU510 作为从站时的 Modbus 地址 1-255；

延迟时间：响应主站延时返回时间，单位毫秒；

超时时间：判断主站连接超时时间，单位毫秒；

通信协议：Modbus TCP/RTU；

作为 Modbus TCP 从站时参数：

端口号：网络端口号；

作为 Modbus RTU 从站时参数：

串行端口：COM0-COM2，RTU 通讯端口。

波特率：1200~115200，波特率可选，默认 9600；

数据位：通讯数据位；

校验位：数据校验位：None/Odd/Even;

停止位：1、2 个停止位；

3、变量，右键点击 MBS，选择变量，弹出变量配置界面；

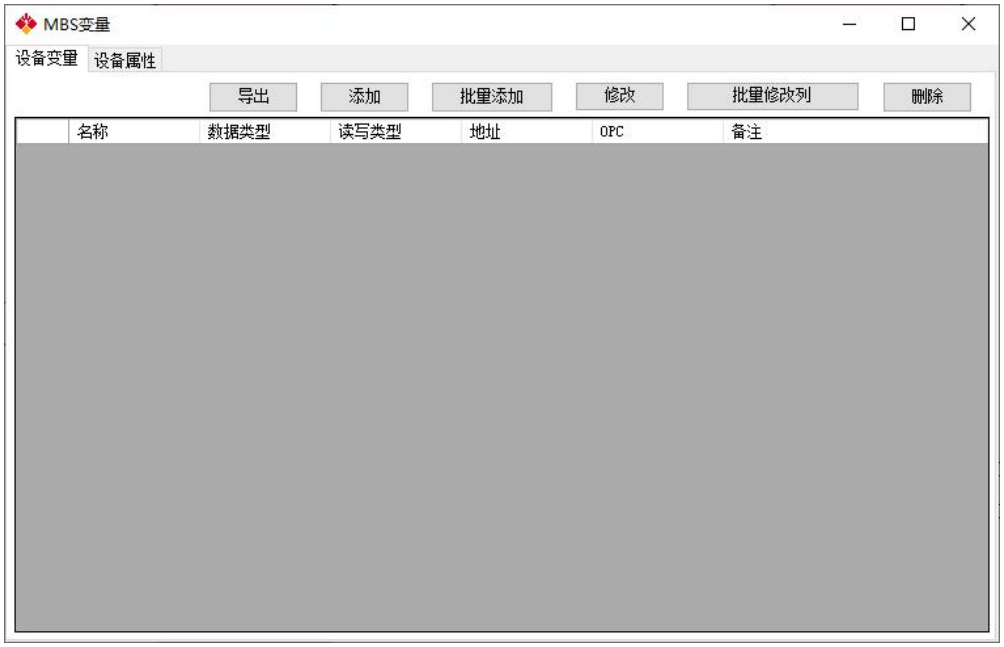


图 1-6-3

添加：添加单个变量，添加界面如下：

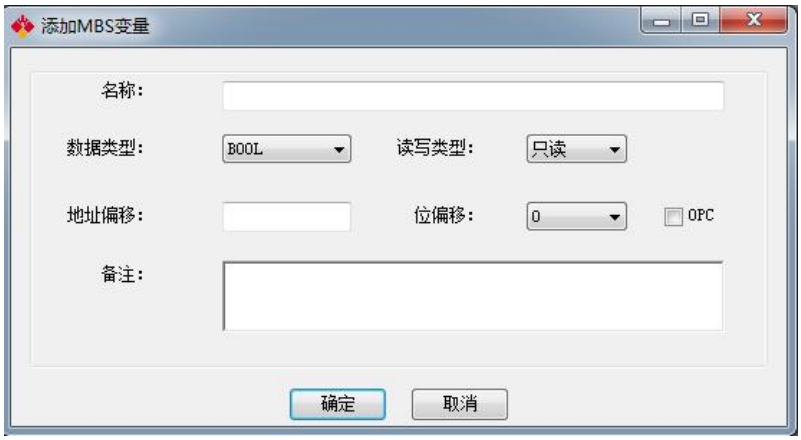


图 1-6-4

名称：变量名称，最长支持 32 个字符（中文 16 个）；

数据类型：系统基本数据类型。

读写类型：只读和读写，以主站角度定义读写类型。

OPC：是否作为 OPC 变量；

地址偏移：V 变量区寄存器地址；

位偏移：针对 BOOL 类型在 V 区字节中的 bit 位偏移，范围 0-7。

地址偏移范围如下表，表 1-6-1

数据类型	地址偏移 X	位偏移 Y	最多变量个数
BOOL	0-124	0-7	1000
INT	0-3998	0	1000

UINT WORD DINT UDINT DWORD REAL STRING			

说明：

1、V 区所有变量累加最多 1000 个变量，BOOL 类型数据地址和非 BOOL 类型数据地址相互独立，所有 非 BOOL 类型地址为公用地址，并且不可重复使用。

2、非 BOOL 变量地址偏移加上变量类型字节长度不得超过 4000；

例如 INT/UINT/WORD 变量字节长度为 2，则偏移地址最大为 3998；

DINT/UDINT/DWORD/REAL 变量字节长度为 4，则偏移地址最大为 3996；

STRING 变量字节长度为 80，则偏移地址最大为 3920.

例如：添加以下变量时地址偏移和位偏移分别为

AA:地址偏移为 0，位偏移为 0；

BB:地址偏移为 0，位偏移为 1；

CC:地址偏移为 0，位偏移为 2；

DD:地址偏移为 0，位偏移为 0；

EE:地址偏移为 2，位偏移为 0；

FF:地址偏移为 6，位偏移为 0；

	名称	数据类型	读写类型	地址	OPC	备注
1	AA	BOOL	读写	VX0.0	否	
2	BB	BOOL	读写	VX0.1	否	
3	CC	BOOL	读写	VX0.2	否	
4	DD	INT	读写	VW0	否	
5	EE	REAL	读写	VDF2	否	
6	FF	INT	读写	VW6	否	

图 1-6-5

在 MCGS 中变量对应设置如下：

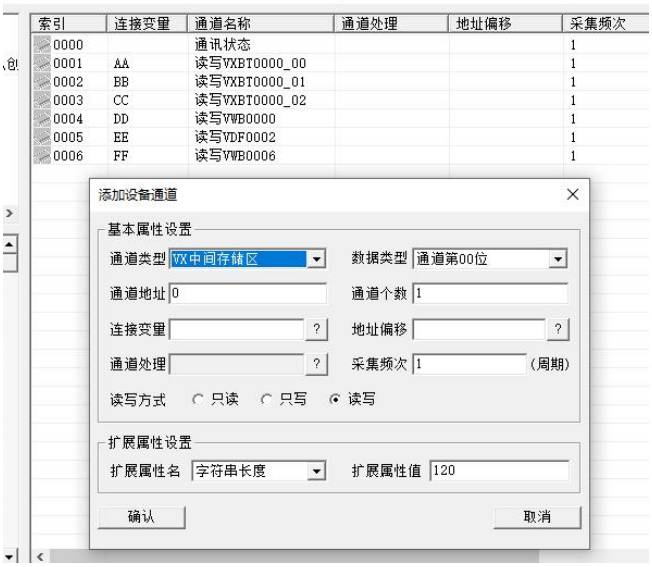


图 1-6-6

表 1-6-2

变量	通道类型	通道地址	数据类型	通道名称	PLC 地址
AA	VX 中间存储区	0	通道第 00 位	读写 VXBT0000_00	VX0.0
BB	VX 中间存储区	0	通道第 01 位	读写 VXBT0000_01	VX0.1
CC	VX 中间存储区	0	通道第 02 位	读写 VXBT0000_00	VX0.2
DD	V 中间存储区	0	16 位 有符号二进制	读写 VWB0000	VW0
EE	V 中间存储区	2	32 位 浮点数	读写 VDF0002	VDF2
FF	V 中间存储区	6	16 位 有符号二进制	读写 VWB0006	VW6

PLC V 区变量与 MCGS 变量对应关系如下：x 为地址偏移，y 为位偏移。

表 1-6-3

MCGS 参数				PLC 参数	
通道类型	通道地址	数据类型	通道名称	PLC 数据类型	PLC 地址
VX 中间存储区	x	通道第 y 位	VXBTx_y	BOOL	VXx.y
V 中间存储区	x	16 位 无符号二进制	VWUBx	UINT	VWUx
V 中间存储区	x	16 位 有符号二进制	VWBx	INT	VWx
V 中间存储区	x	16 位 4 位 BCD	VWDx	WORD	VWDx
V 中间存储区	x	32 位 无符号二进制	VDUBx	UDINT	VDUx
V 中间存储区	x	32 位 有符号二进制	VDBx	DINT	VDx
V 中间存储区	x	32 位 8 位 BCD	VDDx	DWORD	VDDx
V 中间存储区	x	32 位 浮点数	VDFx	REAL	VDFx
V 中间存储区	x	GBK 字符串	VGBKx_80	STRING	VBSx

说明：GBK 字符串变量固定为 80 个字节长度。

PLC I/Q 区变量（IO 模块对应通道）不需要单独添加，在硬件组态时已生成：

	名称	类型	用法	描述	地址
54	⊟ LOC0_DI510_1				
55	LOC0_1_DI510_CH0	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(0)	%IX0.0
56	LOC0_1_DI510_CH1	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(1)	%IX0.1
57	LOC0_1_DI510_CH2	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(2)	%IX0.2
58	LOC0_1_DI510_CH3	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(3)	%IX0.3
59	LOC0_1_DI510_CH4	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(4)	%IX0.4
60	LOC0_1_DI510_CH5	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(5)	%IX0.5
61	LOC0_1_DI510_CH6	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(6)	%IX0.6
62	LOC0_1_DI510_CH7	BOOL	V..	LOC LOC(0) IO DI510(1) Channel(7)	%IX0.7
63	⊟ LOC0_DO510_2				
64	LOC0_2_DO510_CH0	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(0)	%QX0.0
65	LOC0_2_DO510_CH1	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(1)	%QX0.1
66	LOC0_2_DO510_CH2	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(2)	%QX0.2
67	LOC0_2_DO510_CH3	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(3)	%QX0.3
68	LOC0_2_DO510_CH4	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(4)	%QX0.4
69	LOC0_2_DO510_CH5	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(5)	%QX0.5
70	LOC0_2_DO510_CH6	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(6)	%QX0.6
71	LOC0_2_DO510_CH7	BOOL	V..	LOC LOC(0) IO DO510(2) Channel(7)	%QX0.7
72	⊟ LOC0_AI510_3				
73	LOC0_3_AI510_CH0	UINT	V..	LOC LOC(0) IO AI510(3) Channel(0)	%IW1
74	LOC0_3_AI510_CH1	UINT	V..	LOC LOC(0) IO AI510(3) Channel(1)	%IW3
75	LOC0_3_AI510_CH2	UINT	V..	LOC LOC(0) IO AI510(3) Channel(2)	%IW5
76	LOC0_3_AI510_CH3	UINT	V..	LOC LOC(0) IO AI510(3) Channel(3)	%IW7
77	⊟ LOC0_AO510_4				
78	LOC0_4_AO510_CH0	UINT	V..	LOC LOC(0) IO AO510(4) Channel(0)	%QW1
79	LOC0_4_AO510_CH1	UINT	V..	LOC LOC(0) IO AO510(4) Channel(1)	%QW3
80	LOC0_4_AO510_CH2	UINT	V..	LOC LOC(0) IO AO510(4) Channel(2)	%QW5
81	LOC0_4_AO510_CH3	UINT	V..	LOC LOC(0) IO AO510(4) Channel(3)	%QW7

图 1-6-7

在 MCGS 中设置如下：

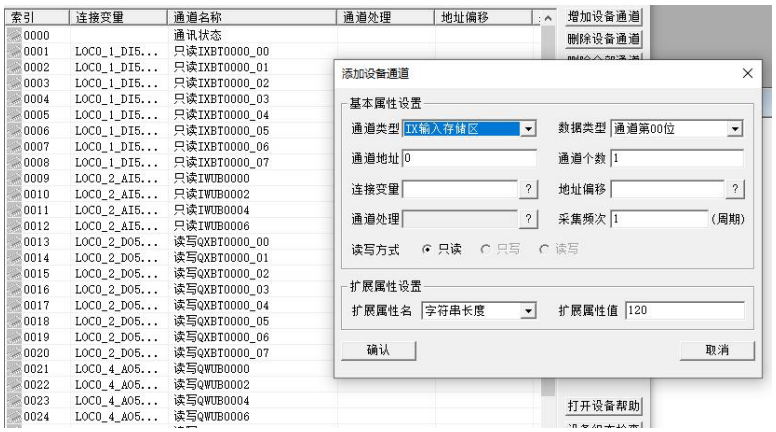


图 1-6-8

表 1-6-4

变量	通道类型	通道地址	数据类型	通道名称	PLC 地址
LOC0_1_DI510_CH0	IX 输入存储区	0	通道第 00 位	只读 IXBT0000_00	IX0.0
LOC0_1_DI510_CH1	IX 输入存储区	0	通道第 01 位	只读 IXBT0000_01	IX0.1
.....	IX 输入存储区	x	通道第 y 位	只读 IXBTx_y	IXx.y
LOC0_2_AI510_CH0	I 输入存储区	0	16 位 无符号二进制	只读 IWUB0002	IW1
LOC0_2_AI510_CH1	I 输入存储区	2	16 位 无符号二进制	只读 IWUB0004	IW3

.....	I 输入存储区	x	16 位 无符号二进制	只读 IWUBx	IW(x-1)
LOCO_3_DO510_CH0	QX 输出存储区	0	通道第 00 位	只写 QXBT0000_00	QX0.0
LOCO_3_DO510_CH1	QX 输出存储区	0	通道第 01 位	只写 QXBT0000_01	QX0.1
.....	QX 输出存储区	x	通道第 y 位	只写 QXBTx_y	QXx.y
LOCO_4_AO510_CH0	Q 输出存储区	0	16 位 无符号二进制	只写 QWUB0002	QW1
LOCO_4_AO510_CH1	Q 输出存储区	2	16 位 无符号二进制	只写 QWUB0004	QW3
.....	Q 输出存储区	x	16 位 无符号二进制	只写 QWUBx	QW(x-1)

说明：

针对 I 输入存储区（AI）和 Q 输出存储区(AO)地址，因为 Modbus 只读输入寄存器和保持寄存器不支持单字节寻址，所以只能是偶数字节地址，而 PLC 的 IO 地址支持单字节地址，所以如果 PLC 地址为奇数（1、3、5、7...），则对应的 MCGS 通道地址需要减去 1，如果 PLC 地址为偶数则不需要处理，可以直接对应。

PLC IO 区变量与 MCGS 变量对应关系如下：

x 为通道地址，y 为通道位偏移，z 为 PLC 模拟量通道地址。

表 1-6-5

MCGS 参数				PLC 参数		
通道类型	通道地址	数据类型	通道名称	PLC 数据类型	PLC 地址	模块类型
IX 输入存储区	x	通道第 y 位	IXBTx_y	BOOL	IXx.y	DI5xx
QX 输出存储区	x	通道第 y 位	QXBTx_y	BOOL	QXx.y	DO5xx
I 输入存储区	x	16 位 无符号二进制	IWUBx	UINT	IWz	AI5xx
Q 输出存储区	x	16 位 无符号二进制	QWUBx	UINT	QWz	AO5xx

说明：

若 z 值为偶数，则 x=z；若 z 值为奇数，则 x=z-1；

1.6.2 MCGS 变量添加

1、添加创恒 PLC_Modbus 驱动

创恒 PLC_Modbus 驱动只支持 McgsPro 组态软件，新建项目后打开设备工具箱页面，找到设备管理，双击设备管理：

图 1-6-9

在设备管理界面依次选择所有设备-->PLC-->创恒-->TUR-MODBUS-RTU/TCP-->创恒 PLC_ModbusTCP/RTU，然后点击增加，即可把设备驱动添加到右侧，如下图：

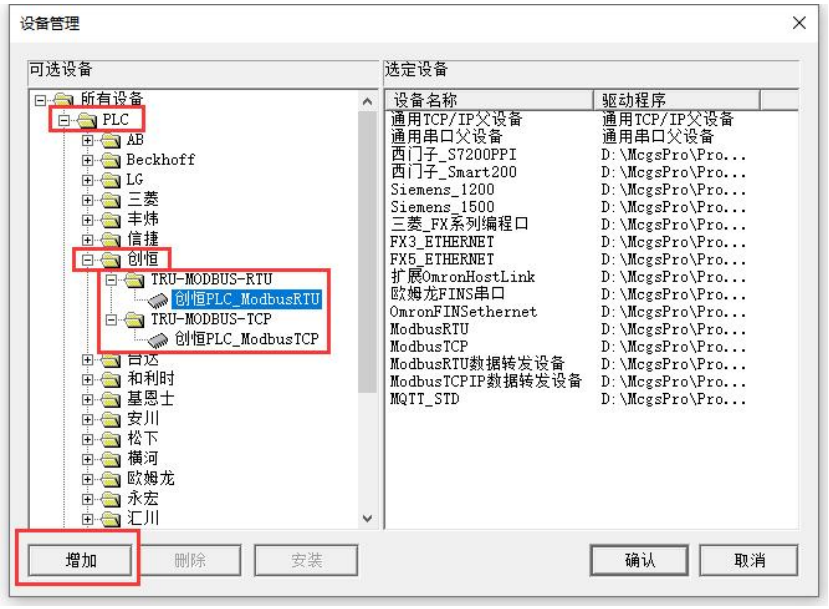


图 1-6-10

如果在 PLC 中找不到创恒分支，则需要联系 PLC 厂家，手动添加，添加方法如下：在 McgsPro 安装目录中依次选择 Program-->Drivers-->PLC，在目录中新建创恒文件夹，把厂家提供的驱动放入文件夹内，重启 McgsPro 软件即可。

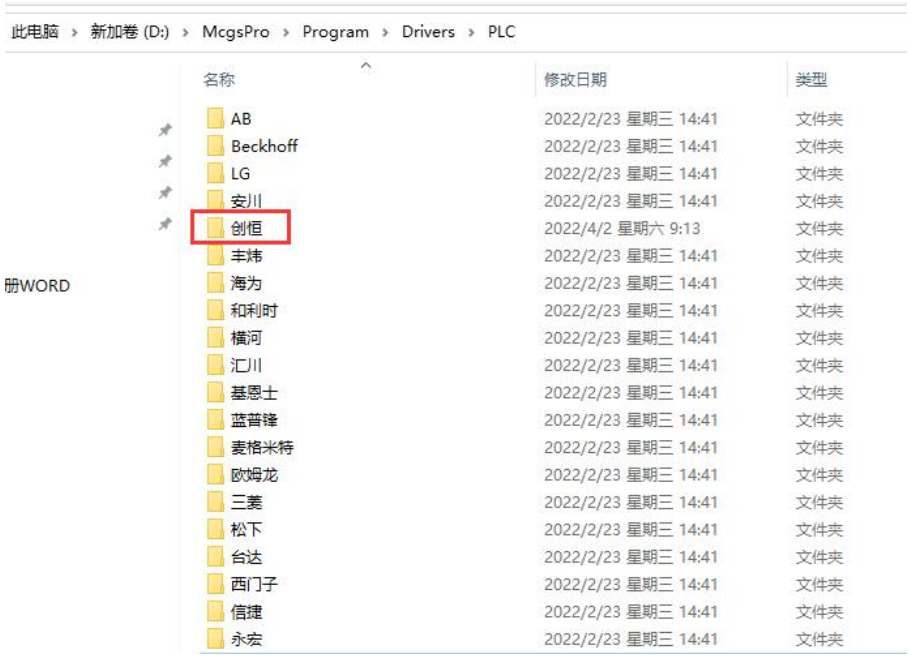


图 1-6-11

2、添加创恒 PLC_Modbus 设备

驱动添加完成后添加 Modbus 设备，以添加 ModbusRTU 为例：

先添加通用串口父设备，再添加创恒 PLC_ModbusRTU：

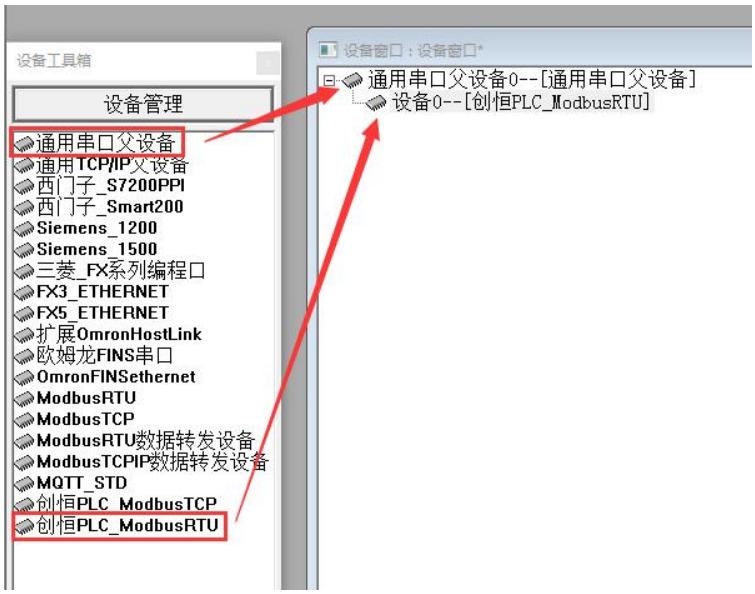


图 1-6-12

双击父设备，弹出通用串口设备属性编辑界面，根据需求选择端口号和通讯参数。

图 1-6-13

3、添加通道变量

双击设备进入设备编辑窗口。

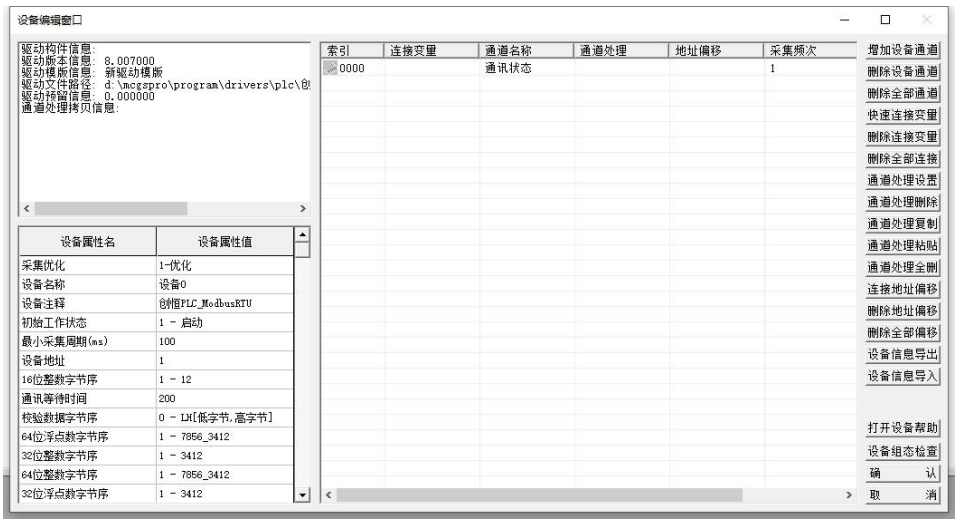


图 1-6-14

可根据需求修改左侧设备属性，除了设备地址和通讯等待时间其他默认即可。

点击增加设备通道：

图 1-6-15

通道类型、通道地址和数据类型，参考 1.6.1 章节中 PLC V 区变量与 MCGS 通道变量对应关系表（表 1-6-3）；IO 变量与 MCGS 通道变量对应关系（表 1-6-5）。

或者通过 Multiprog 软件 MBS 导出的 MCGSPro 格式的文件，批量导入到 MCGS，在设备编辑窗口选择“设备信息导入”，如图 1-6-16：

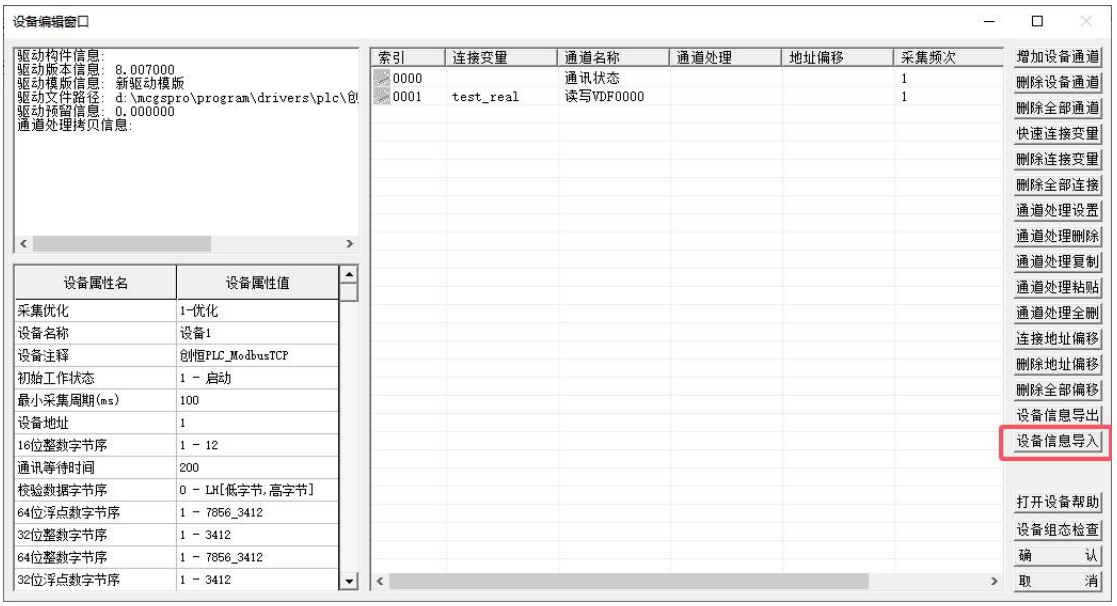


图 1-6-16

在弹出界面中选择 Multiprog 导出的 csv 文件，如下图：

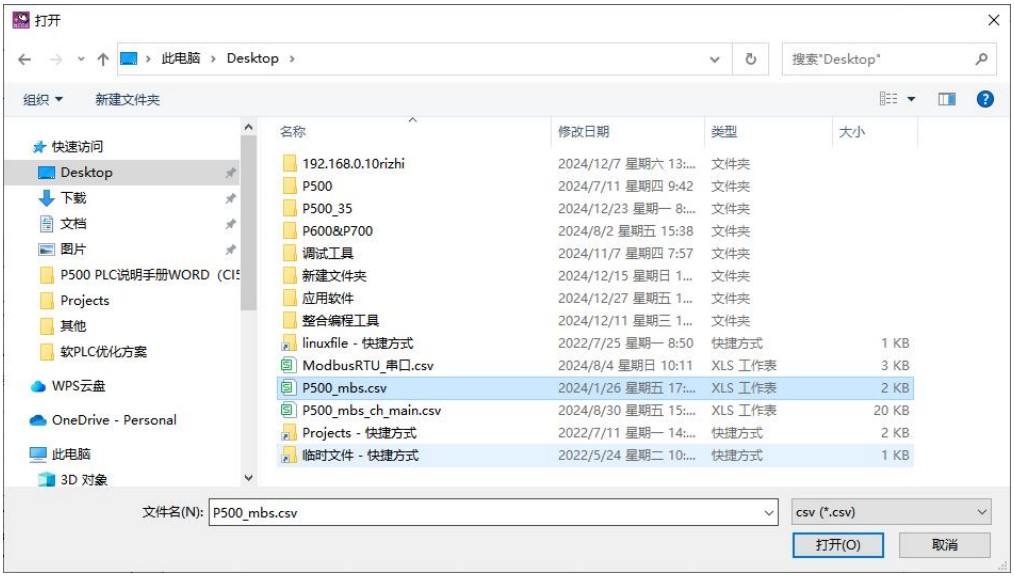
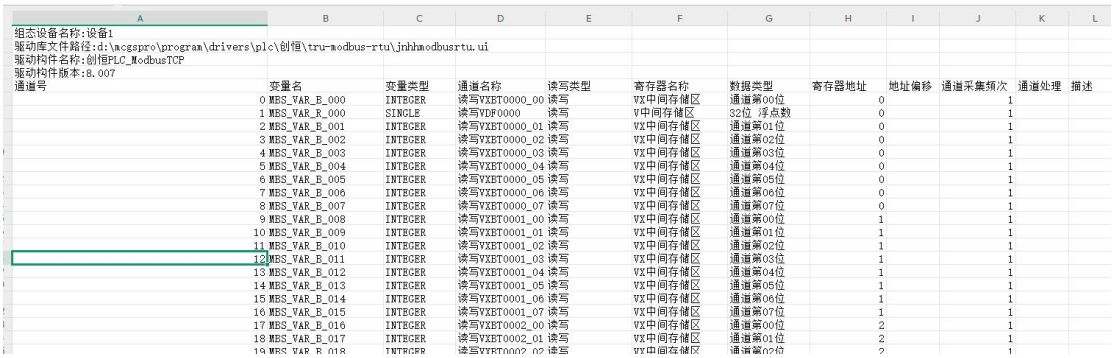


图 1-6-17

MCGSPro 内容格式如下：



组态设备名称:设备1
驱动库文件路径:d:\mcgspro\program\drivers\plc\创恒\tru-modbus-rtu\jnhhmodbusrtu.ui
驱动构件名称:创恒PLC_ModbusTCP
驱动构件版本:8.007

通道号	变量名	变量类型	通道名称	读写类型	寄存器名称	数据类型	寄存器地址	地址偏移	通道采集频次	通道处理	描述
0	MBS_VAR_B_000	INTEGER	读写VXBT0000_00	读写	VX中间存储区	通道第00位	0		1		
1	MBS_VAR_R_000	SINGLE	读写VDF0000	读写	V中间存储区	32位 浮点数	0		1		
2	MBS_VAR_B_001	INTEGER	读写VXBT0000_01	读写	VX中间存储区	通道第01位	0		1		
3	MBS_VAR_B_002	INTEGER	读写VXBT0000_02	读写	VX中间存储区	通道第02位	0		1		
4	MBS_VAR_B_003	INTEGER	读写VXBT0000_03	读写	VX中间存储区	通道第03位	0		1		
5	MBS_VAR_B_004	INTEGER	读写VXBT0000_04	读写	VX中间存储区	通道第04位	0		1		
6	MBS_VAR_B_005	INTEGER	读写VXBT0000_05	读写	VX中间存储区	通道第05位	0		1		
7	MBS_VAR_B_006	INTEGER	读写VXBT0000_06	读写	VX中间存储区	通道第06位	0		1		
8	MBS_VAR_B_007	INTEGER	读写VXBT0000_07	读写	VX中间存储区	通道第07位	0		1		
9	MBS_VAR_B_008	INTEGER	读写VXBT0001_00	读写	VX中间存储区	通道第00位	1		1		
10	MBS_VAR_B_009	INTEGER	读写VXBT0001_01	读写	VX中间存储区	通道第01位	1		1		
11	MBS_VAR_B_010	INTEGER	读写VXBT0001_02	读写	VX中间存储区	通道第02位	1		1		
12	MBS_VAR_B_011	INTEGER	读写VXBT0001_03	读写	VX中间存储区	通道第03位	1		1		
13	MBS_VAR_B_012	INTEGER	读写VXBT0001_04	读写	VX中间存储区	通道第04位	1		1		
14	MBS_VAR_B_013	INTEGER	读写VXBT0001_05	读写	VX中间存储区	通道第05位	1		1		
15	MBS_VAR_B_014	INTEGER	读写VXBT0001_06	读写	VX中间存储区	通道第06位	1		1		
16	MBS_VAR_B_015	INTEGER	读写VXBT0001_07	读写	VX中间存储区	通道第07位	1		1		
17	MBS_VAR_B_016	INTEGER	读写VXBT0002_00	读写	VX中间存储区	通道第00位	2		1		
18	MBS_VAR_B_017	INTEGER	读写VXBT0002_01	读写	VX中间存储区	通道第01位	2		1		

图 1-6-18

组态设备名称：需要和 MCGSPro 软件里面建立的设备名称同名，根据需要自行修改，否则会导入失败。

驱动库文件路径：需要和 MCGSPro 软件安装的创恒 MCGS 驱动路径相同，注意驱动类型 TCP/RTU，根据需要自行修改。

驱动构件名称：需要和 MCGSPro 软件里面驱动名称相同，根据需要自行修改。

驱动构件版本：需要和 MCGSPro 软件里面驱动版本相同，根据需要自行修改。

以上信息需要和 MCGS 安装目录和设备信息对应，否则会添加失败。

1.7 Modbus RTU 硬件组态

1.7.1 使能配置串口

使能串口，右键选择“串行端口”，选择“使能串行端口”，选择需要的串口名称：

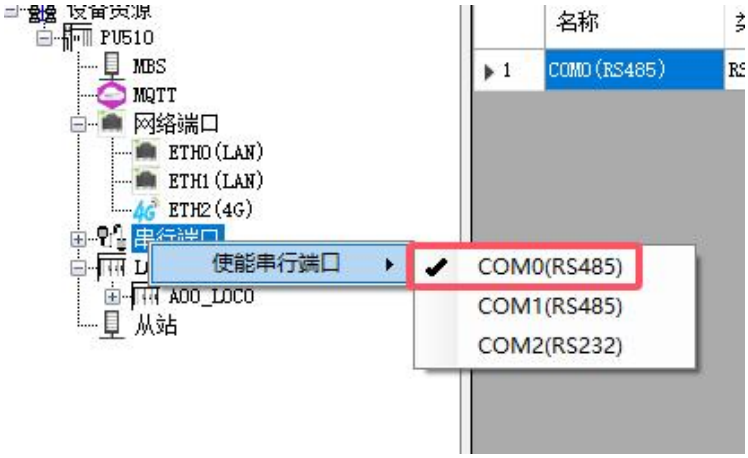


图 1-7-1

配置串口属性，右键选择添加的串口，选择“编辑”：

图 1-7-2

在串口编辑界面设备品牌和设备类型选择“Modbus”：

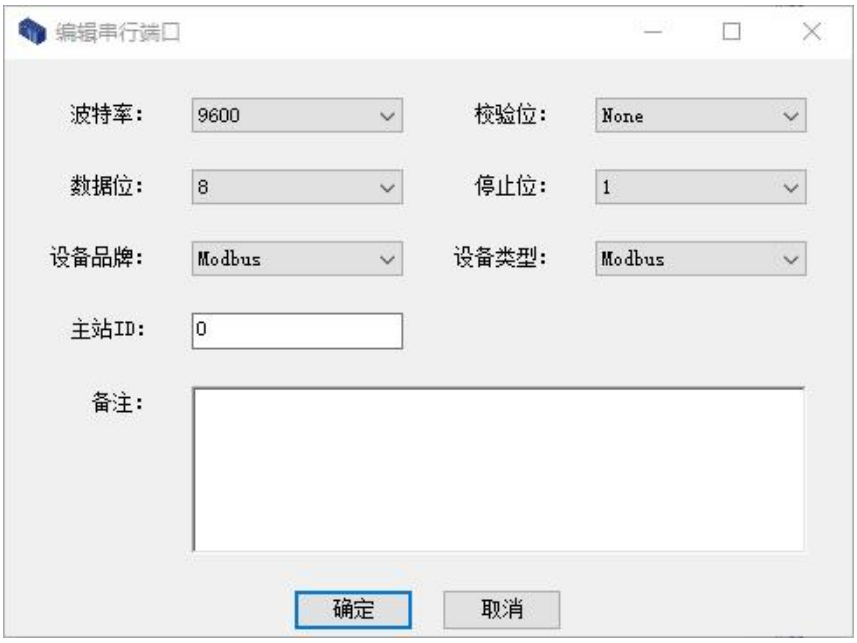


图 1-7-3

Modbus 支持的参数如下：
波特率：1200、2400、4800、9600、19200、38400、57600、115200 可选；
校验位：数据校验位：None/Odd/Even 可选；
数据位：7、8 可选；
停止位：1、2 可选；
设备品牌：固定 Modbus；
设备类型：固定 Modbus；
主站 ID：忽略；
备注：可填写备注。

1.7.2 添加设备

右键选择添加的串口，选择“添加设备”：

图 1-7-4

弹出添加设备界面：

图 1-7-5

名称：设备名称，字符长度由变量和设备名称长度决定，变量最长长度和设备名称长度总

长不超过 30 字符；例如设备名称为 10 字符，则变量名称不能超过 20 字符。
品牌、设备类型：在编辑串口属性的时候确定，在此页面不可修改；
从站 ID:从站站地址，范围 1-255；
超时时间：判断从站超时时间，单位毫秒。
通信间隔：每条数据包发送间隔时间，单位毫秒。
扫描周期：每个站通讯周期，单位毫秒。
保持寄存器变化发送：写保持寄存器是否执行变化发送。

1.7.3 添加变量

右键已添加的设备，选择“变量”：

图 1-7-6

弹出设备变量界面如下

图 1-7-7

导入、导出：均支持 MCGSPro 和 Modbus 内容格式的 CSV 文件，可方便的建立点表，格

式介绍参考 1.5 章节。

名称：字符长度由设备名称和变量长度决定，变量最长长度和设备名称长度总长不超过 30 字符；例如设备名称为 10 字符，则变量名称不能超过 20 字符。

寄存器类型：分为[0 区]线圈寄存器、[1 区]离散输入寄存器、[3 区]只读输入寄存器、[4 区]保持寄存器；

读写类型：分为只读、只写、读写；

数据类型：支持 BOOL、INT、UINT、DINT、UDINT、REAL、LREAL、WORD、DORD；

地址：Modbus 寄存器地址，起始地址为 0 地址；

OPC：是否具有 OPC 属性。

备注：可添加变量备注。

设备属性选项页：

可修改设备变量在通讯中的数据格式，方便不同字节序系统中数据的对应。

图 1-7-8

表 1-7-1

数据类型	字节序	默认字节序
16 位字节序	0 - 21	1 - 12
	1 - 12	
32 位字节序	0 - 4321	1 - 3412
	1 - 3412	
	2 - 2143	
	3 - 1234	
64 位字节序	0 - 8765_4321	1 - 7856_3412
	1 - 7856_3412	
	2 - 6587_2143	
	3 - 5678_1234	
	4 - 4321_8765	
	5 - 3412_7856	

	6 - 2143_6587	
	7 - 1234_5678	
字符串字节序	0 - 21	1 - 12
	1 - 12	
字符串编码格式	0 - GBK	0 - GBK
	1 - UTF8	
	2 - UNICODE	
写单个线圈功能码	5	5
	15	
写单个保持功能码	6	16
	16	

1.8 Modbus TCP 硬件组态

1.8.1 添加从站

右键点击“从站”，选择“添加从站”，选择“Modbus”：

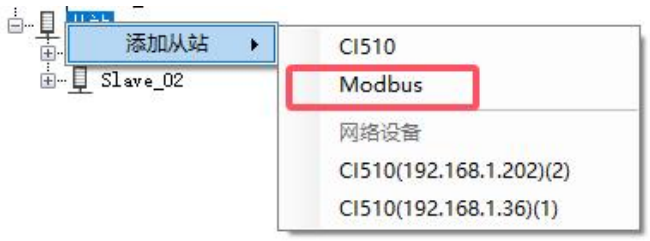


图 1-8-1

弹出添加 Modbus 从站界面：

图 1-8-2

名称：从站名称可修改。
从站 ID：1-99 可选。
通信协议：固定 TRUBUS_TCP。
IP 地址 1、2：通信协议是 TCP 时从站 IP 地址，支持两个 IP 地址，至少填一个。
端口号：TCP 端口号，默认 502，可修改。
通信间隔：每条报文间隔时间，单位 ms，可修改。
超时时间：判断从站超时时间，单位 ms，可修改。

1.8.2 添加变量

右键已添加的从站，选择“变量”：

图 1-8-3

弹出设备变量界面如下

图 1-8-4

导入、导出：均支持 MCGSPro 和 Modbus 内容格式的 CSV 文件，可方便的建立点表，格式介绍参考 1.5 章节。

名称：名称长度不超过 26 字符。

寄存器类型：分为[0 区]线圈寄存器、[1 区]离散输入寄存器、[3 区]只读输入寄存器、[4 区]保持寄存器；

读写类型：分为只读、只写、读写；

数据类型：支持 BOOL、INT、UINT、DINT、UDINT、REAL、LREAL、WORD、DORD；

地址：Modbus 寄存器地址，起始地址为 0 地址；

OPC：是否具有 OPC 属性。

备注：可添加变量备注。

设备属性选项页：

可修改设备变量在通讯中的数据格式，方便不同字节序系统中数据的对应。

图 1-8-5

表 1-8-1

数据类型	字节序	默认字节序
16 位字节序	0 - 21	1 - 12
	1 - 12	
32 位字节序	0 - 4321	1 - 3412
	1 - 3412	
	2 - 2143	
	3 - 1234	
64 位字节序	0 - 8765_4321	1 - 7856_3412
	1 - 7856_3412	
	2 - 6587_2143	
	3 - 5678_1234	
	4 - 4321_8765	
	5 - 3412_7856	
	6 - 2143_6587	
	7 - 1234_5678	
字符串字节序	0 - 21	1 - 12
	1 - 12	
字符串编码格式	0 - GBK	0 - GBK
	1 - UTF8	
	2 - UNICODE	
写单个线圈功能码	5	5
	15	
写单个保持功能码	6	16
	16	

第 2 章　CI510 通讯配置

P500 系统可通过 CI510 扩展远程 IO 模块，增加实际 IO 控制点数量。

2.1 添加 I/O 配置组合

右键点击设备资源，选择“配置”。

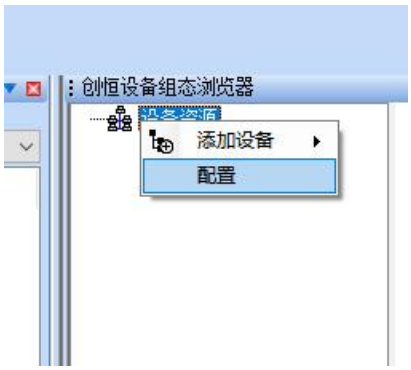


图 2-1-1

弹出自定义硬件配置界面，分为 3 种类型硬件配置，CPU 为 PU510 本体 IO 模块，CI 为 CI510 本体模块，LOC 为扩展 IO 模块，如下图：

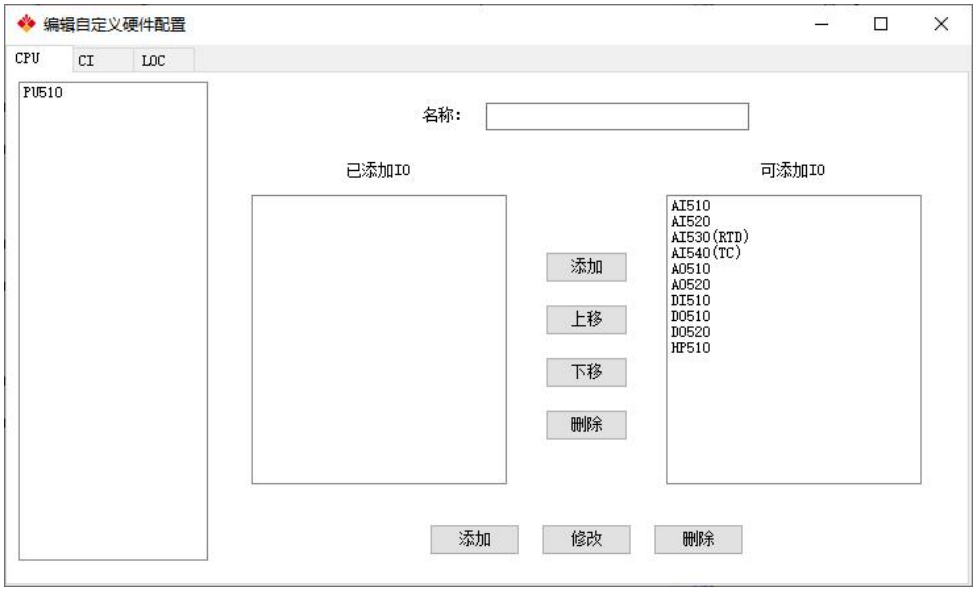


图 2-1-2

添加方法如下：

首先在“可添加 IO”列表中找到对应的模块，点击中间“添加”按钮，加入到“已添加 IO”列表里面，CPU 和 CI 最多可添加 5 个模块，LOC 最多可添加 3 个模块，然后通过上移和下移按钮调整模块顺序；

添加完成后给当前配置填写名称，填写完成后，点击底部“添加”按钮，即可把当前配置保存，并且显示在左侧列表里面。添加完成后，如下图所示：

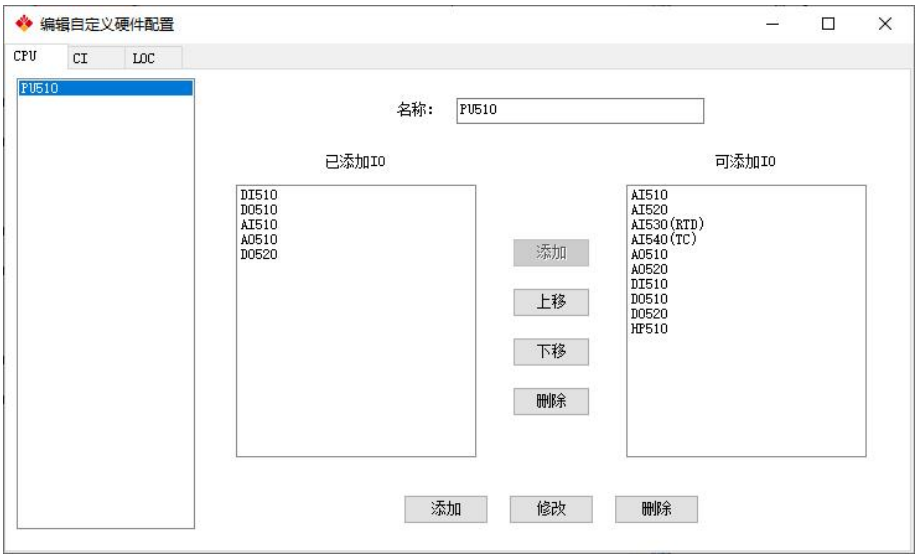


图 2-1-3

添加完成后关闭配置界面。

2.2 手动添加 CI510 以及 IO 模块

右键点击“从站”节点，选择“添加从站”，如图：

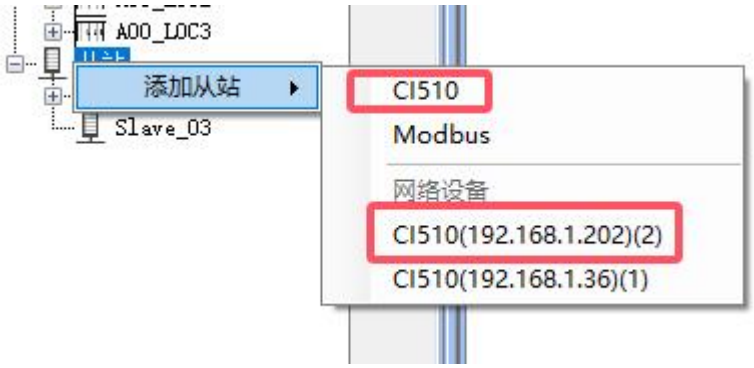


图 2-2-1

选择 CI510 后弹出“添加 CI510 从站”界面：

图 2-2-2

名称：从站名称可修改。
从站 ID：1-99 可选。
通信协议：TRUBUS_TCP 和 TRUBUS_RTU 可选。
IP 地址 1、2：通信协议是 TCP 时从站 IP 地址，支持两个 IP 地址，至少填一个。
端口号：TCP 端口号，默认 502，可修改。
通信间隔：每条报文间隔时间，单位 ms，可修改。
超时时间：判断从站超时时间，单位 ms，可修改。
串行端口：通讯协议是 RTU 时，串口选择，COM0-COM2。
波特率：串口通讯速率，1200、2400、4800、9600、19200、38400、57600、115200。
数据位：5-8
校验位：数据校验位：None/Odd/Even;
停止位：1、2 个停止位。
添加 IO 模块，首先添加本体模块，右键选择 LOC，选择“添加 LOC”,选择预先定义好的 CI510 配置。

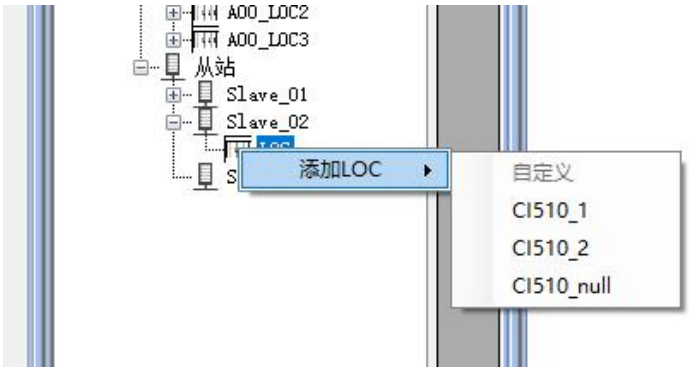


图 2-2-3

然后添加扩展 IO，右键选择“LOC”，“添加 LOC”，选择预先定义的扩展 IO 配置：

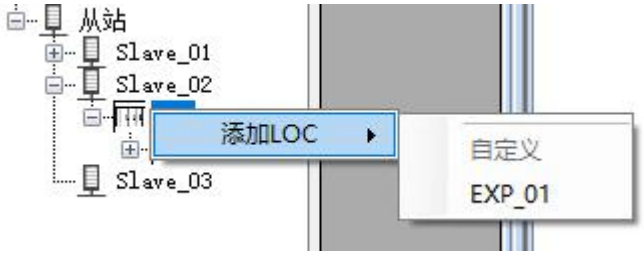


图 2-2-4

最多添加 4 个扩展 IO 分组。

添加完成后会在全局变量表里面生成相应的分组和通道变量。

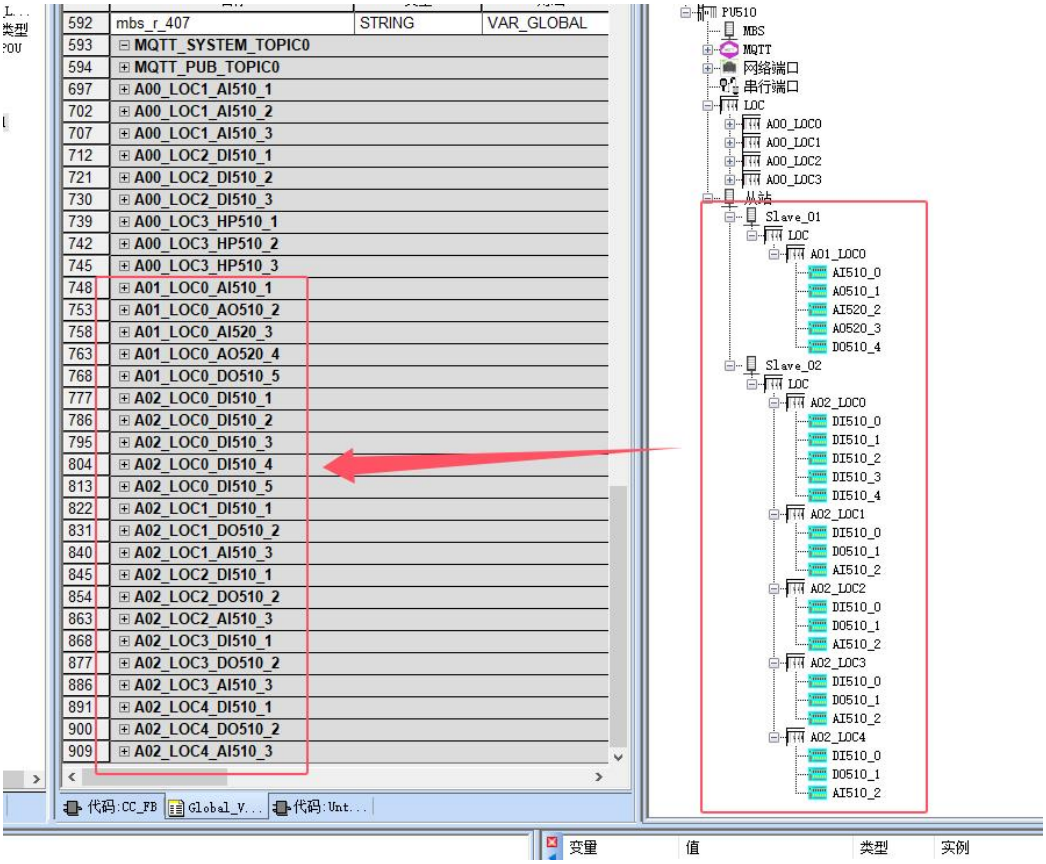


图 2-2-5

2.3 网络自动添加 CI510 以及 IO 模块

右键点击从站，选择“添加从站”，从网络设备中选择可用 CI510。

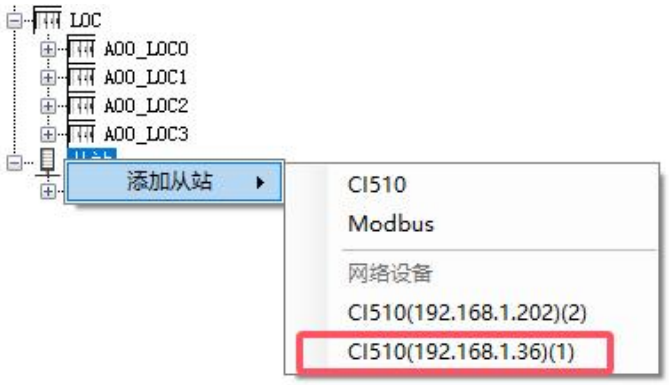


图 2-3-1

如果没有找到可用设备可检查网卡是否选择正确，电脑 IP 地址是否和 CI510 在同一个网段内，然后重新扫描网络设备。

网卡选择：

图 2-3-2

电脑 IP 地址设置：

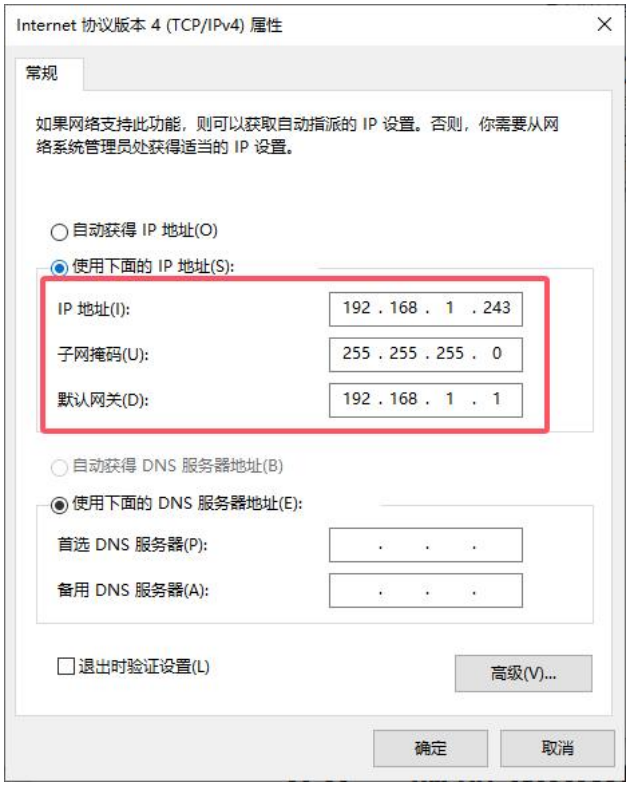


图 2-3-3

扫描网络设备：

图 2-3-4

添加 CI510 后，右键选择添加的设备，选择“自动组态”：

图 2-3-5

系统会自动从 CI510 中获取 IO 模块信息，添加到系统中，如果获取失败请检查网络是否正常，CI510 的 IP 地址是否填写正确。

第 3 章　自由口通讯使用

3.1 物理端口配置

自由口的物理端口配置仅是对所用串行端口进行初始化配置由 FREE_PORT_INIT 功能块完成，需确定所使用的扩展端口、波特率、数据位、奇偶校验位、停止位。本次举例使用扩展端口 COM0，设置 115200 波特率、8 位数据位、无校验、1 位停止位。编写如下。

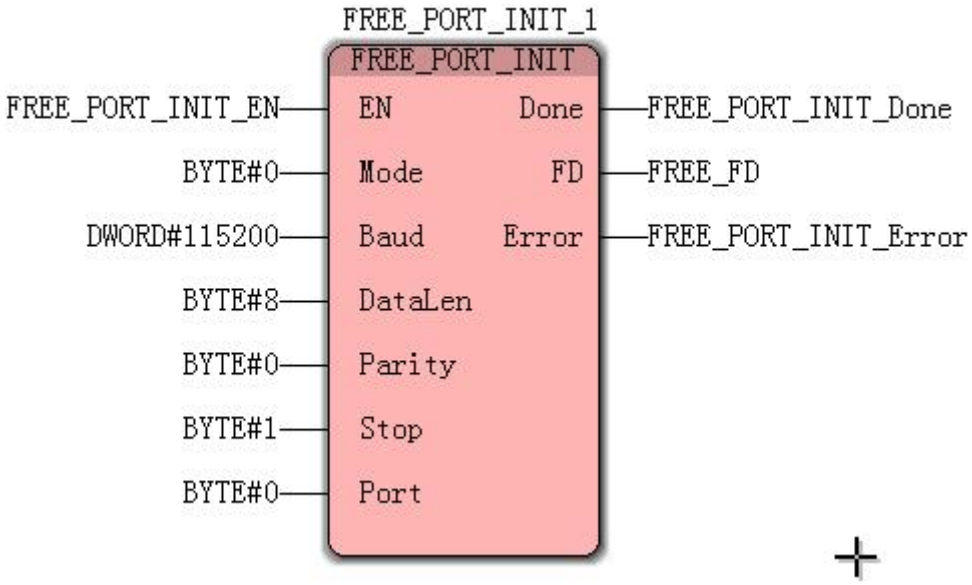


图 3-1-1

EN 位连接 FREE_PORT_INIT_EN，用于 FREE_PORT_INIT_1 功能块的使能与禁用的控制，置为 True，方能够配置扩展端口 COM0 作为自由口通讯，置为 False，扩展端口 COM0 上的自由口通讯功能禁用。

Mode 传入参数 BYTE#0，Mode 用于设置功能块模式，当前为保留选项，该参数设置为 0 即可；

Baud 传入参数 DWORD#115200，即设置波特率为 115200；

DataLen 传入参数 BYTE#8，即表示设置数据位为 8。

Parity 传入参数 BYTE#0，即表示设置校验位为无校验。0 为无校验，1 为奇校验，2 为偶校验；

Stop 传入参数 BYTE#1，即表示设置停止位为 1。

Port 传入参数 BYTE#0，即表示使用扩展端口 0。0 为 COM0，1 为 COM1，2 为 COM2；

Done 为配置结果输出位，结果传入变量 FREE_PORT_INIT_Done。

FD 为自由口初始化连接标识符。

Error 为配置错误信息输出，结果传入变量 FREE_PORT_INIT_Error。

工程运行后将 FREE_PORT_INIT_EN 置为 True，若 FREE_PORT_INIT_Done 结果为 True，FREE_PORT_INIT_Error 结果为 0，即表示将 COM0 配置为自由口通讯所需的物理端口成功。如下图所示。

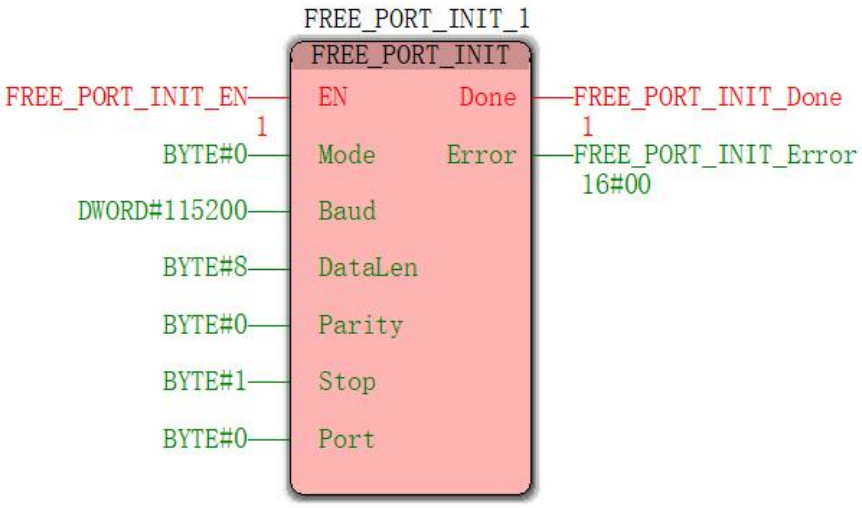


图 3-1-2

3.2 数据接收

3.2.1 数据接收配置

自由口的数据接收使用 FREE_PORT_RCV 功能块完成，功能块编写如下。

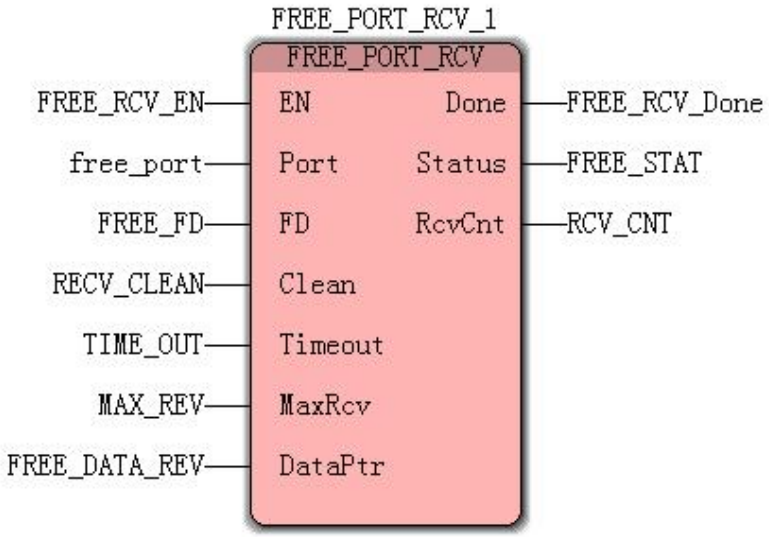


图 3-2-1

EN 位连接 FREE_RCV_EN，用于开启一次自由口接收，当置为 True 时，功能块会根据起始条件接收数据，置为 False，接收停止，并清空功能块的输出内容。

Port 传入参数 free_port 设为 1，即表示使用扩展端口 COM1，该端口已使用 FREE_PORT_INIT_1 功能块配置过；

FD 传入参数为 FREE_FD，该标识符为 FREE_PORT_INIT_1 初始化成功后产生的标识符。

Clean 传入参数为 RECV_CLEAN，清空上次接收数据和接收状态，以便接收新的数据

和状态。

Timeout 传入参数为 TIME_OUT，参数预留。

MaxRcv 传入参数 MAX_RCV，设置为 10，设定最大接收 10 个字节；

DataPtr 传入的是一个地址，用来存放自由口接收到的数据，本次使用%MB3.0。

Done 为数据接收结果输出位，结果传入变量 FREE_PORT_RCV_Done。

Status 为数据接收信息输出，结果传入变量 FREE_PORT_RCV_Status。

RcvCnt 传出实际接收数据的大小。

开启一次数据接收，当 Done 被置为 True 后，表示功能块处理完毕，如果 Status 为 0 并且 RcvCnt 不为 0，表示收到数据，否则没有收到数据或者产生错误信息。结果如下：

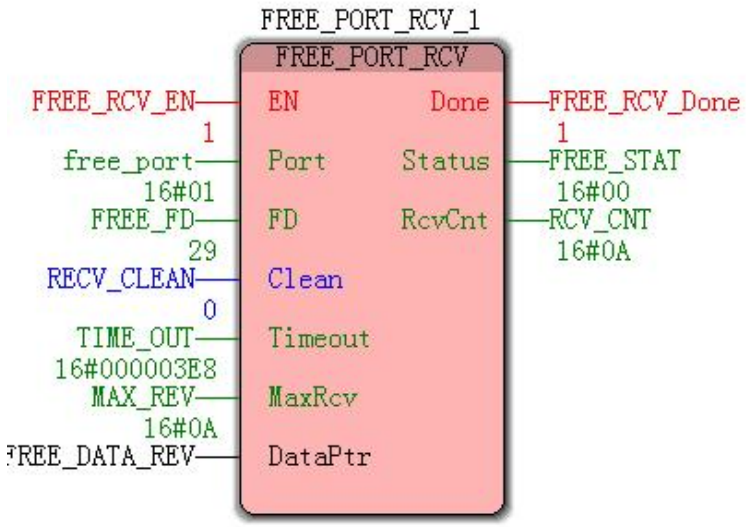


图 3-2-2

3.2.2 应用举例

首先建立变量表展示自由口接收数据。

	名称	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ data											
	data_1	BYTE	VAR_GL...	数据1	%MB3.0		☐	☐	☐	☐	☐	
	data_2	BYTE	VAR_GL...	数据2	%MB3.1		☐	☐	☐	☐	☐	
	data_3	BYTE	VAR_GL...	数据3	%MB3.2		☐	☐	☐	☐	☐	
	data_4	BYTE	VAR_GL...	数据4	%MB3.3		☐	☐	☐	☐	☐	
	data_5	BYTE	VAR_GL...	数据5	%MB3.4		☐	☐	☐	☐	☐	
	data_6	BYTE	VAR_GL...	数据6	%MB3.5		☐	☐	☐	☐	☐	
	data_7	BYTE	VAR_GL...	数据7	%MB3.6		☐	☐	☐	☐	☐	
	data_8	BYTE	VAR_GL...	数据8	%MB3.7		☐	☐	☐	☐	☐	
	data_9	BYTE	VAR_GL...	数据9	%MB3.8		☐	☐	☐	☐	☐	
	data_10	BYTE	VAR_GL...	数据10	%MB3.9		☐	☐	☐	☐	☐	

图 3-2-3

如图所示，功能块 FREE_PORT_RCV_1 的 Done 为被置为 True，表示处理完成，Status 状态码输出 0，表示功能块没有错误，RcvCnt 输出 10，表示接收到 10 个字节大小的字符。

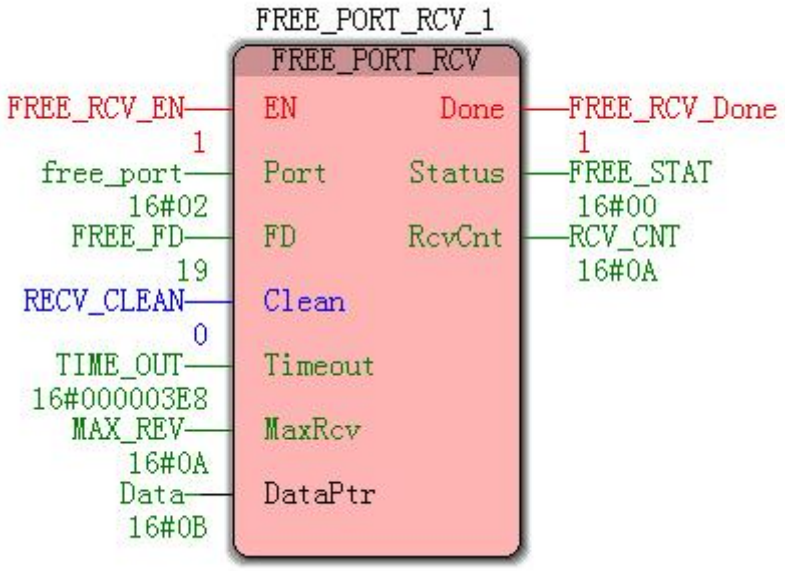


图 3-2-4

查看变量表，接收数据与发送数据一致，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ data												
	data_1	16#00	BYTE	VAR_GL...	数据1	%MB3.0		□	□	□	□	□	
	data_2	16#01	BYTE	VAR_GL...	数据2	%MB3.1		□	□	□	□	□	
	data_3	16#02	BYTE	VAR_GL...	数据3	%MB3.2		□	□	□	□	□	
	data_4	16#03	BYTE	VAR_GL...	数据4	%MB3.3		□	□	□	□	□	
	data_5	16#04	BYTE	VAR_GL...	数据5	%MB3.4		□	□	□	□	□	
	data_6	16#05	BYTE	VAR_GL...	数据6	%MB3.5		□	□	□	□	□	
	data_7	16#06	BYTE	VAR_GL...	数据7	%MB3.6		□	□	□	□	□	
	data_8	16#07	BYTE	VAR_GL...	数据8	%MB3.7		□	□	□	□	□	
	data_9	16#08	BYTE	VAR_GL...	数据9	%MB3.8		□	□	□	□	□	
	data_10	16#09	BYTE	VAR_GL...	数据10	%MB3.9		□	□	□	□	□	

图 3-2-5

3.3 数据发送

3.3.1 数据发送配置

自由口的数据接收使用 FREE_PORT_XMT 功能块完成，相对数接收的功能块配置更为简单，仅需填写扩展端口、发送端口标识符、发送数据的大小、以及需要发送数据在本地的首地址即可，处理结果同样会通过 Done 与 Error 输出。

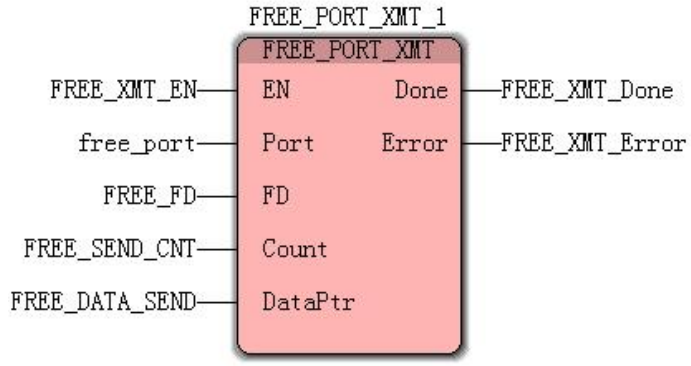


图 3-3-1

EN 位连接 FREE_PORT_XMT_EN，用于开启一次自由口发送，当置为 True 时，功能块会根据配置的数据地址及其个数通过指定扩展端口发出数据，置为 False，功能块使能，并清空功能块的输出内容。

Port 传入参数 free_port 设置为 1，即表示使用扩展端口 1，该端口已使用 FREE_PORT_INIT_1 功能块配置过；

Count 传入参数 FREE_SEND_CNT 设置为 10，设定发送 10 个字节数据；

DataPtr 传入的是一个地址，用来确定发送数据的位置，本次使用%MX3.0.0。

Done 为数据接收结果输出位，结果传入变量 FREE_PORT_XMT_Done。

Status 为数据接收信息输出，结果传入变量 FREE_PORT_XMT_Error。

当 FREE_PORT_XMT_EN 置为 True 后，Done 输出 True，Error 输数 0，表示数据发送成功，功能块状态如下所示。

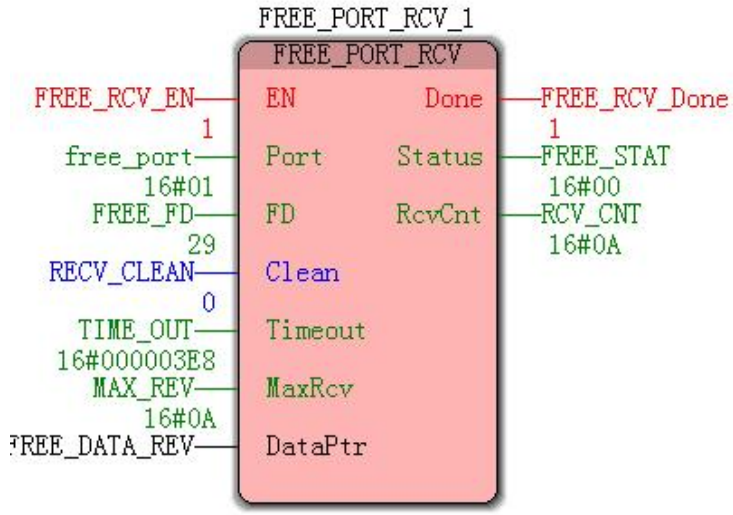


图 3-3-2

3.3.2 应用举例

将 0-9，10 个数据通过扩展端口 COM1 发送出去，并使用串口工具接收。

1、建立变量表，修改变量值如下；

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初	默认的隐藏值
	⊟ data												
	data_1	16#00	BYTE	VAR_GL...	数据1	%MB3.0		☐	☐	☐	☐	☐	
	data_2	16#01	BYTE	VAR_GL...	数据2	%MB3.1		☐	☐	☐	☐	☐	
	data_3	16#02	BYTE	VAR_GL...	数据3	%MB3.2		☐	☐	☐	☐	☐	
	data_4	16#03	BYTE	VAR_GL...	数据4	%MB3.3		☐	☐	☐	☐	☐	
	data_5	16#04	BYTE	VAR_GL...	数据5	%MB3.4		☐	☐	☐	☐	☐	
	data_6	16#05	BYTE	VAR_GL...	数据6	%MB3.5		☐	☐	☐	☐	☐	
	data_7	16#06	BYTE	VAR_GL...	数据7	%MB3.6		☐	☐	☐	☐	☐	
	data_8	16#07	BYTE	VAR_GL...	数据8	%MB3.7		☐	☐	☐	☐	☐	
	data_9	16#08	BYTE	VAR_GL...	数据9	%MB3.8		☐	☐	☐	☐	☐	
	data_10	16#09	BYTE	VAR_GL...	数据10	%MB3.9		☐	☐	☐	☐	☐	

图 3-3-3

2、开启串口工具，等待接收；

3、使能 FREE_PORT_XMT_1 功能块的 EN 位，发出数据；

4、观察串口工具接收结果。

如下图所示，接收数据与发送数据一致，自由口数据发送成功。

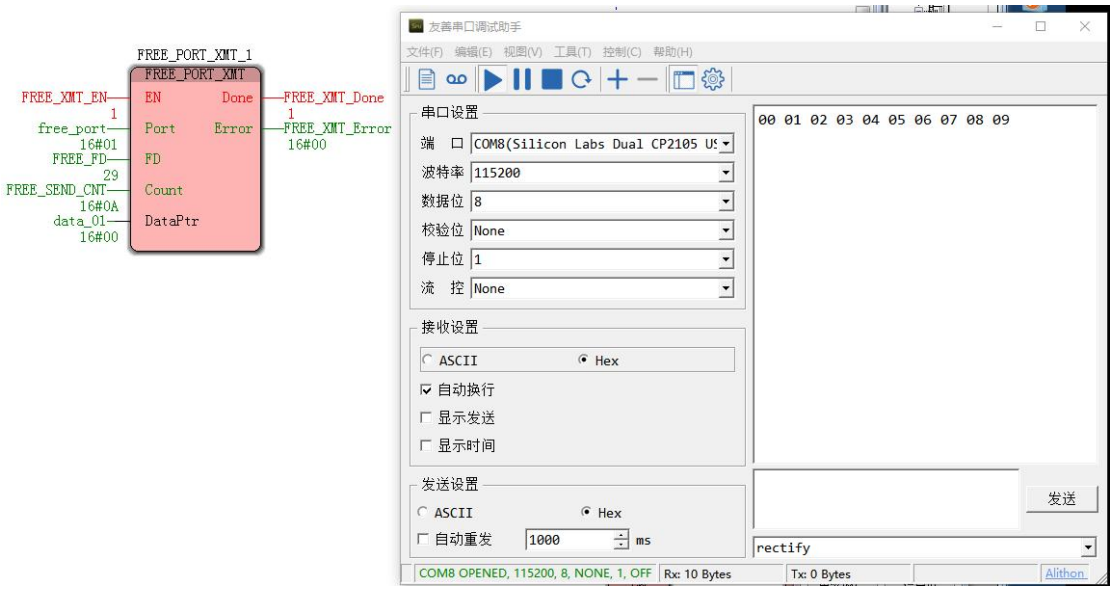


图 3-3-4

第 4 章　连接第三方 PLC

P500 设备具有连接第三方 PLC 的功能，可通过扩展端口，使用相应物理线缆，连接 P500 设备与第三方 PLC 设备，建立物理连接，之后使用 Multiprog 软件进行相应配置，便可以使用 P500 设备监控及控制第三方 PLC 设备。

当前支持西门子 S7-200 的 PPI 接口以及三菱的 FX_Serial 口。

4.1 西门子 S7-200 连接

4.1.1 S7-200 参数确定

要使用 P500 连接 S7-200，必须对 S7-200 的相应参数有所了解，以便在 P500 中进行相应配置。

本例选择 CPU224，

14 路输入，10 路输出，

PORT0 为 19200 波特率、偶检验、8 位数据位、1 位停止位，

站地址为 2。

4.1.2 配置 P500 设备

1、打开工程，在 Truhigh 栏中找打串行端口分支，如下图所示。

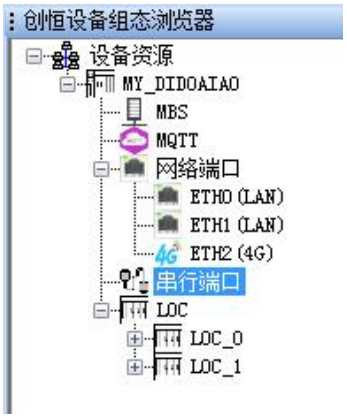


图 4-1-1

2、使能相应扩展端口

本例选择 P500 的扩展端口 0，在串行分支中对 COM0 使能，如下图所示。

图 4-1-2

成功使能 COM 后在串行端口分支下自动创建 COM0（RS485）分支，右键点击 COM0(RS485)，选择编辑，弹出配置界面，如下图所示。

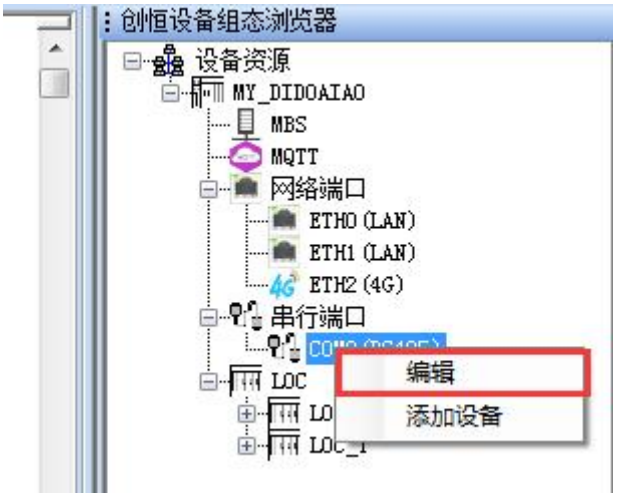


图 4-1-3

图 4-1-4

3、配置扩展端口

扩展端口的配置包含三部分，协议设置、串口设置以及说明。

协议设置用于在指定端口上绑定相应的 PLC 协议，仅需选择目的 PLC 的品牌与型号即可完成，与 S7-200 通讯，需选择“西门子”品牌、“S7-200”型号。主站 ID 选择一个不与目的 PLC 冲突的站地址即可，本例选择 0。如下图所示

串口设置是对扩展端口进行的物理通讯配置，需与第三方 PLC 匹配，根据第三方 PLC 填写如下：

波特率：19200；

检验位：偶检验；

数据位：8 位；

停止位：1 位。

说明为对本扩展端口的解释说明，任意填写即可。

填写完成，最后点击确定即完成配置。

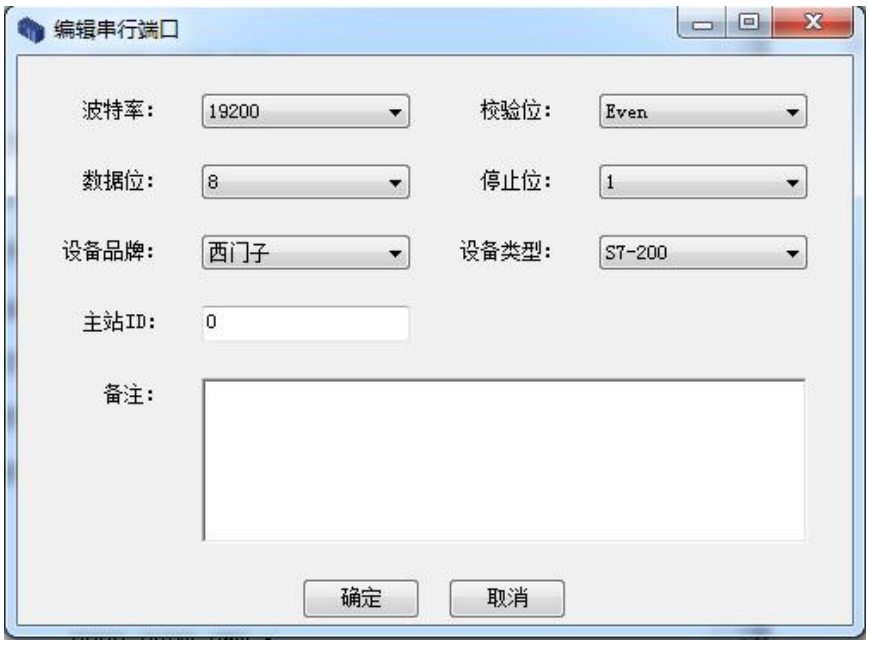


图 4-1-5

4、录入目标 PLC 参数

（1）找到之前添加的扩展端口 COM0（RS485），对其右击，选择设备添加，如下图所示。

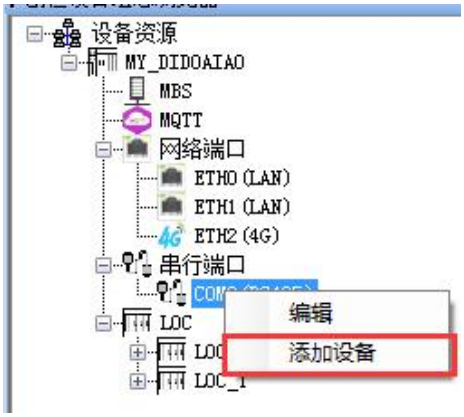


图 4-1-6

（2）添加设备后在右侧弹出 PLC 设备配置栏，如下图所示。

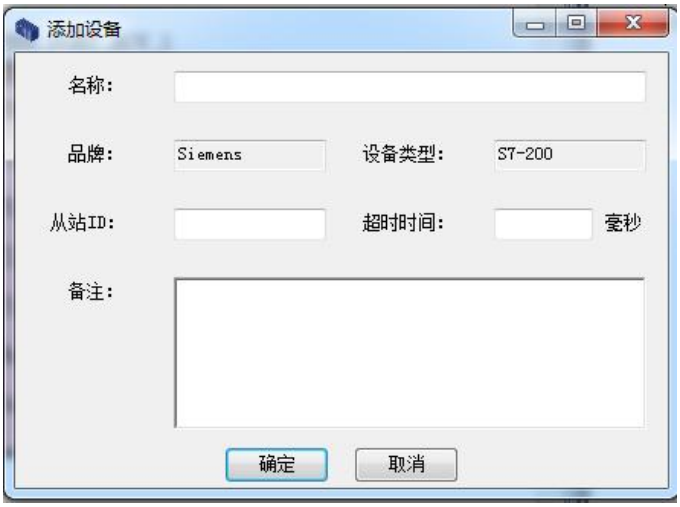


图 4-1-7

（3）录入 PLC 参数

①定义设备名称为“PLC_1”，从站 ID 使用 PLC 本身的站地址，填入“2”，通讯超时填入“200”，单位为 ms。

②点击保存。

图 4-1-8

4、关联 PLC 变量

右键点击“PLC_1”选择“变量”，如下图：

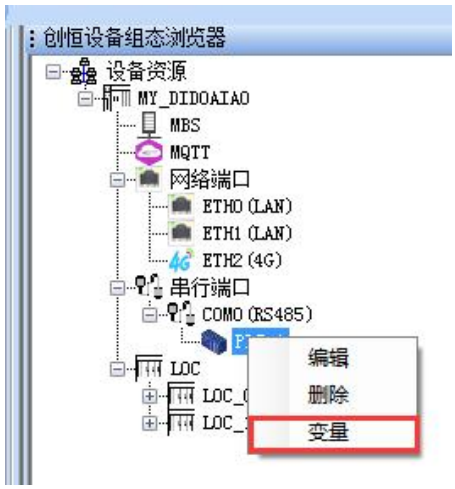


图 4-1-9

选择变量，弹出变量添加界面：

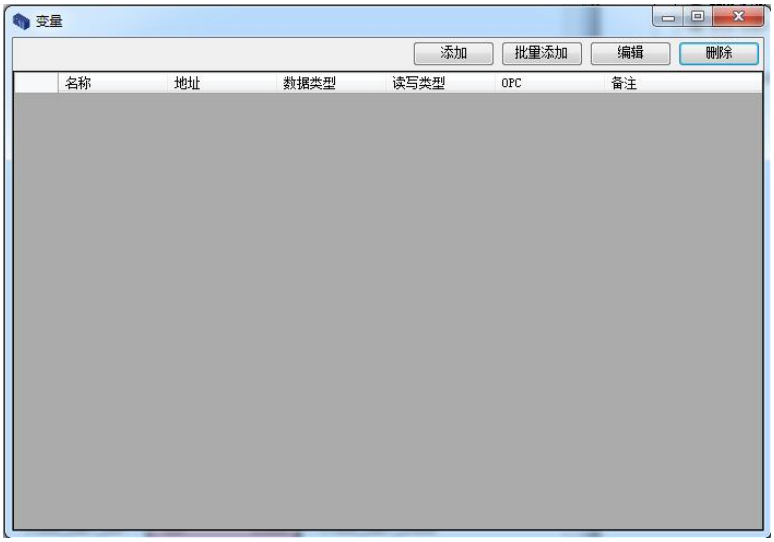


图 4-1-10

点击添加按钮，弹出 PLC 变量填写界面，如下图所示。

图 4-1-11

变量名称是指第三方 PLC 中的变量在 P500 设备上的变量代表值；

OPC 选项，勾选则将该变量加入到 OPC 监控中；

地址是指第三方 PLC 中的变量地址，对于西门子 PLC 有 I、Q、AI、AQ、M、MB、MW、MD、S、SB、SW、SD、SM、SMB、SMW、SMD、V、VB、VW、VD、T、C；地址格式按照第三方 PLC 格式填写；

数据类型是指第三方 PLC 中的变量的数据类型在 P500 中以何种数据类型存放；

读写类型包括只读和读写两种，部分变量仅有只读类型；

详细描述任意填写即可。

例如，关联 S7-200 的 I0.0

根据要求，列出如下内容

变量名称：PLC1_I0_0；

OPC 选项：勾选；

地址:I0.0；

数据类型：BOOL，数据类型根据地址填写自动列出可选类型，如果地址填写非法则数据类型不可选；

读写类型：只读，S7-200 的 I 类型变量不可写；

详细描述：S7-200 的 I0.0。

如下图所示。

图 4-1-12

添加成功后，会在变量表中生成一条关联 I0.0 的信息，全局变量表中“PLC_1”分组下出现“PLC1_I0_0”变量，如下图所示。

图 4-1-13

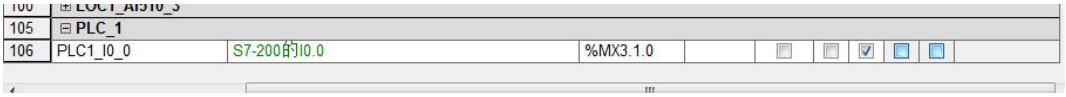


图 4-1-14

至此将 S7-200 的 I0.0 变量与 P500 的 PLC1_I0_0 关联成功，P500 与 S7-200 通讯配置完成，编译工程，下载验证即可。

4.1.3 下装验证

下载工程后，打开“在线调试”功能，监控全局变量表如下：

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初...	默认的隐藏值
	⊟ PLC_1												
	PLC1_I0_0	FALSE	BOOL	VAR_GLOBAL	S7-200 的 I0.0	%MX3.0.0		☐	☐	☑	☐	☐	

图 4-1-15

人为给 S7-200 的 I0.0 接入 24V 点，使其亮灯，观察全局变量表中 PLC1_I0_0 变量被置为 True，如下图所示。

	名称	联机值	类型	用法	描述	地址	初值	保持	P...	O...	隐藏	初...	默认的隐藏值
	⊟ PLC_1												
	PLC1_I0_0	TRUE	BOOL	VAR_GLOBAL	S7-200 的 I0.0	%MX3.0.0		☐	☐	☑	☐	☐	

图 3-1-16

综上，P500 设备连接西门子 S7-200PLC 成功。

4.2 三菱 FX3GA 连接

4.2.1 FX3GA 参数确定

要使用 P500 连接 FX3GA，必须对 FX3GA 的相应参数有所了解，以便在 P500 中进行相应配置。

本例选择 FX3GA-24MR，

14 路输入，10 路输出，

FX_Serial 口配置为 9600 波特率、无校验、8 位数据位、1 位停止位，

站地址为 0。

4.2.2 配置 P500 设备

1、打开工程，在 Truhigh 栏中找打 COM 分支，如下图所示。

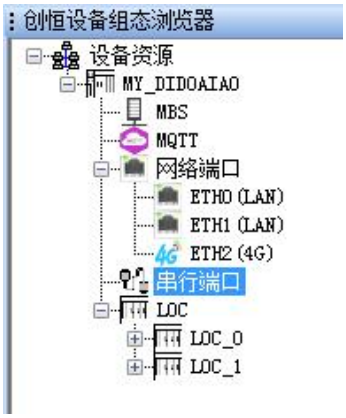


图 4-2-1

2、使能相应扩展端口

本例选择 P500 的扩展端口 0，在 COM 分支中对 COM0 使能，如下图所示。

图 4-2-2

成功使能 COM 后在串行端口分支下自动创建 COM0（RS485）分支，右键点击

COM0(RS485)，选择编辑，弹出配置界面，如下图所示。

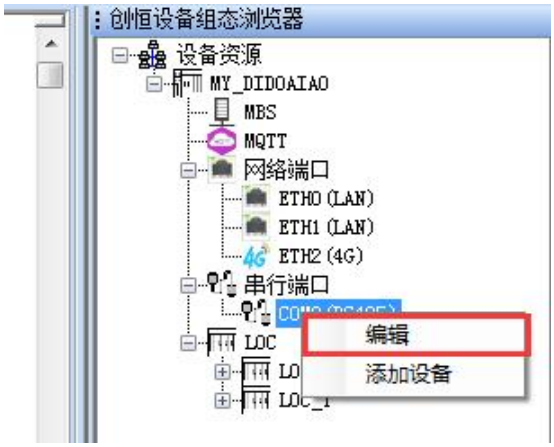


图 4-2-3

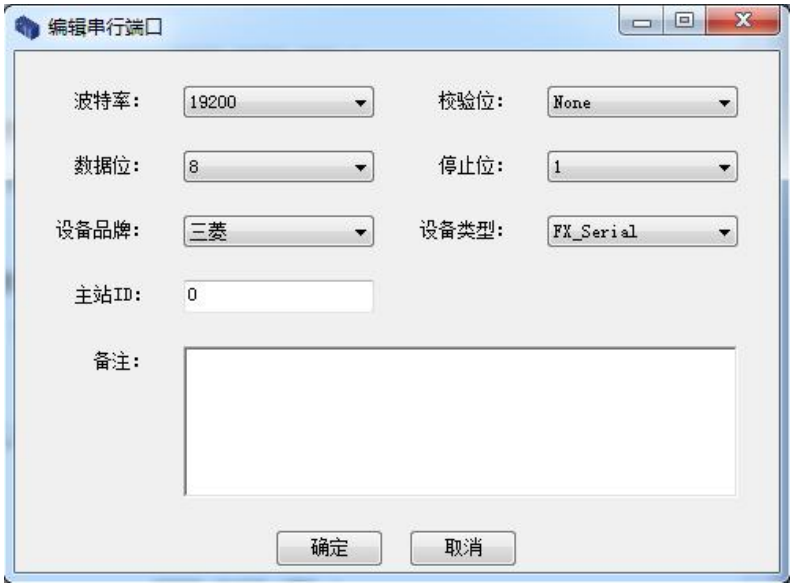


图 4-2-4

3、配置扩展端口

扩展端口的配置包含三部分，协议设置、串口设置以及说明。

协议设置用于在指定端口上绑定相应的 PLC 协议，仅需选择目的 PLC 的品牌与型号即可完成，使用 FX_Serial 口与 FX3GA 通讯，需选择“三菱”品牌、“FX_Serial”通讯方式。主站 ID 选择本例选择 0。如下图所示

串口设置是对扩展端口进行的物理通讯配置，需与第三方 PLC 匹配，根据第三方 PLC 填写如下：

波特率：9600；

检验位：无校验；

数据位：8 位；

停止位：1 位。

说明为对本扩展端口的解释说明，任意填写即可。

填写完成，最后点击确定即完成配置。

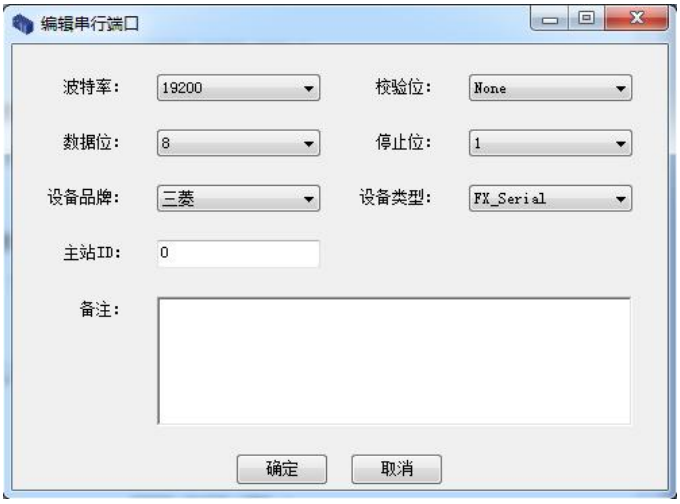


图 4-2-5

4、录入目标 PLC 参数

（1）找到之前添加的扩展端口 COM0（RS485），对其右击，选择设备添加，如下图所示。

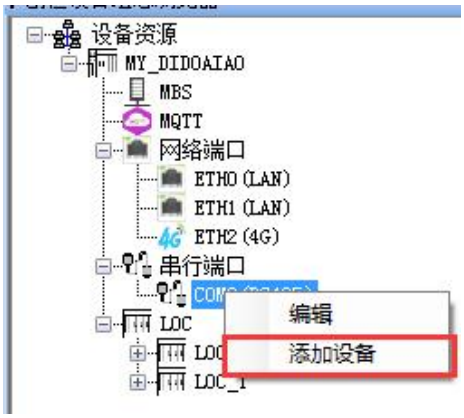


图 4-2-6

（2）添加设备后在右侧弹出 PLC 设备配置栏，如下图所示。

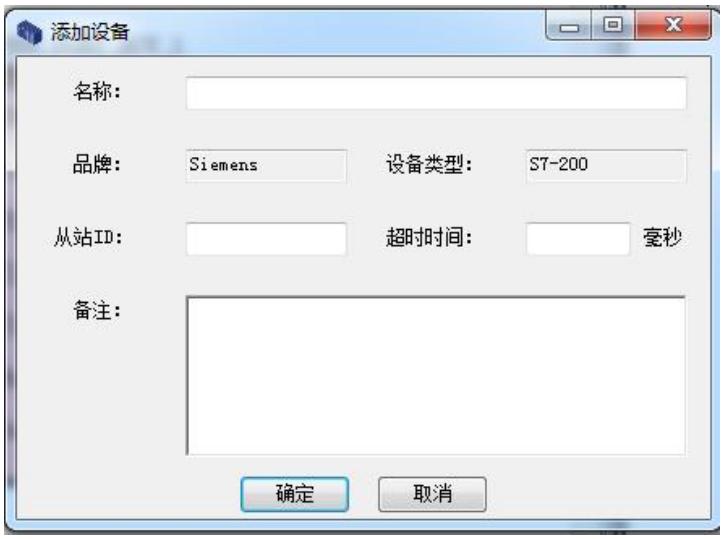


图 4-2-7

（3）录入 PLC 参数

①定义设备名称为“PLC_1”，从站 ID 使用 PLC 本身的站地址，填入“0”，通讯超

时填入“200”，单位为 ms。

②点击确定，设备添加完成。

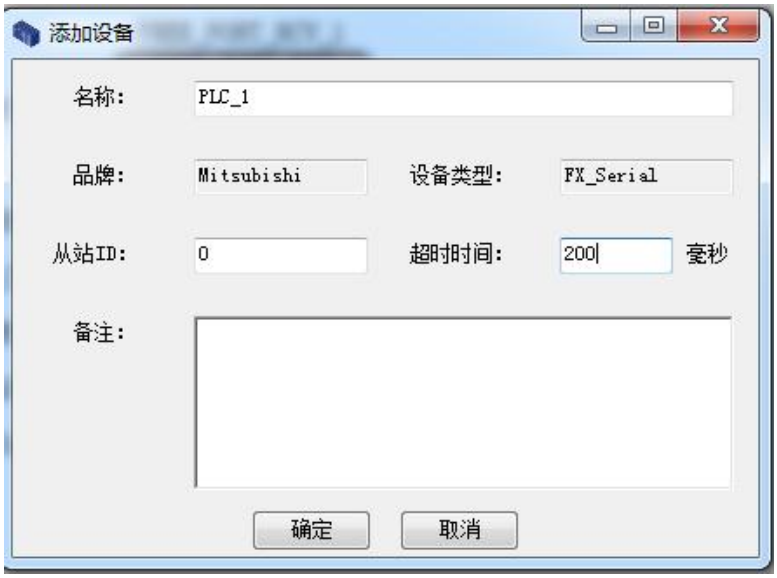


图 4-2-8

4、关联 PLC 变量

右键点击“PLC_1”选择“变量”，如下图：

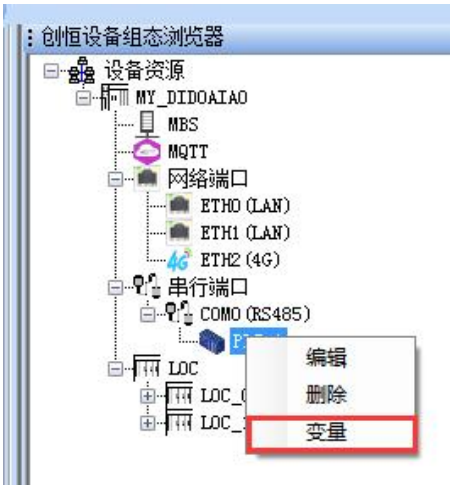


图 4-2-9

选择变量，弹出变量添加界面：

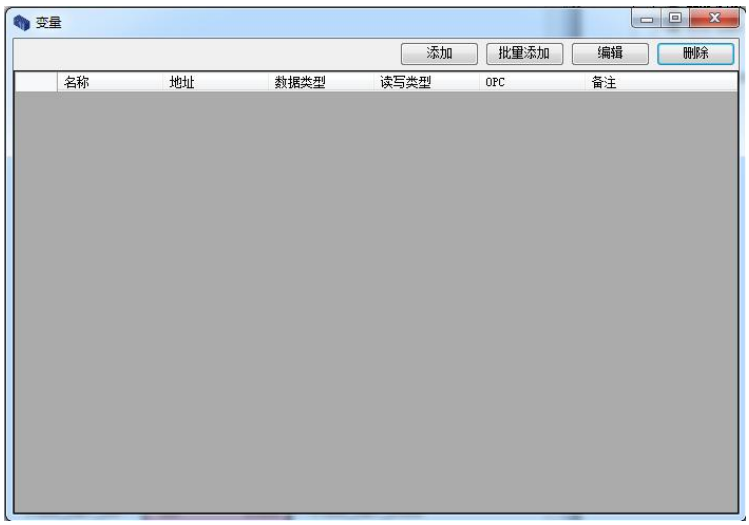


图 4-2-10

点击添加按钮，弹出 PLC 变量填写界面，如下图所示。

图 4-2-11

变量名称是指第三方 PLC 中的变量在 P500 设备上的变量代表值；

OPC 选项，勾选则将该变量加入到 OPC 监控中；

地址是指第三方 PLC 中的变量类型，对于三菱 PLC 有 X、Y、M、D，地址格式按照第三方 PLC 格式填写；

数据类型是指第三方 PLC 中的变量的数据类型在 P500 中以何种数据类型存放；

读写类型包括只读和读写两种，部分变量仅有只读类型；

详细描述任意填写即可。

（1）关联 FX3GA 的 X00

根据要求，列出如下内容

变量名称：PLC1_X00；

OPC 选项：勾选；

地址:X00；

数据类型：BOOL，数据类型根据地址填写自动列出可选类型，如果地址填写非法则数据类型不可选；

读写类型：只读，X 元件不可写；

详细描述：FX3GA 的 X00。

如下图所示。

图 4-2-12

添加成功后，会在变量表中生成一条关联 X00 的信息，全局变量表中“PLC_1”分组下出现“PLC1_I0_0”变量，如下图所示。

图 4-2-13

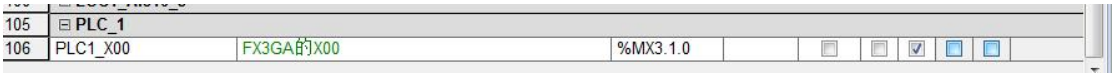


图 4-2-14

至此将 FX3GA 的 X00 变量与 P500 的 PLC1_X00 关联成功，P500 与 FX3GA 通讯配置完成，编译工程，下载验证即可。

4.2.3 下装验证

下载工程后，打开“在线调试”功能，监控全局变量表如下：

图 4-2-15

人为给 FX3GA 的 X00 接入 24V 点，使其亮灯，观察全局变量表中 PLC1_X00 变量被置为 True，如下图所示。

图 4-2-16

综上，P500 设备连接三菱 FX3GAPLC 成功。

第 5 章　MQTT 连接

P500 PLC 设备可通过网络连接互联网云平台，实现远程数据采集及组态控制。目前支持的云平台有 ALiYun，Azure，OneNET，Schneider，Truhigh 同时只能支持一种平台。

5.1 ALiYun 连接设置

1. 首先，添加 MQTT 类型。右键点击 MQTT 在弹出的界面中依次选择“添加 MQTT”，选择 AliYun，如图 5-1-1。

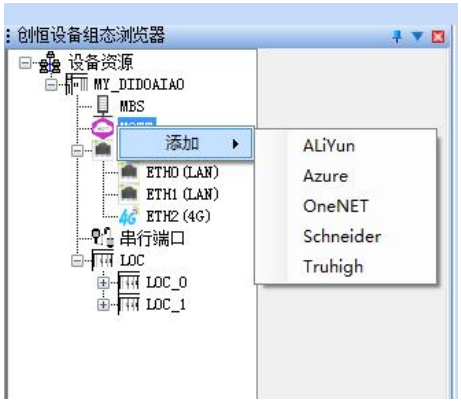


图 5-1-1

2. 然后，编辑 MQTT 属性，如图 5-1-2：

图 5-1-2

连接保持时间：云平台连接保活时间，单位秒。

数据存储时间间隔：

Min：云平台连接失败时，如果数据变化则每隔 Min（单位秒）时　间　保存一条数据到 PLC 数据库;

Max：云平台连接失败时，如果数据一直保持不变则每隔 Max（单　位秒）时间保存一条数据到 PLC 数据库；

数据上传时间间隔：

Min：当数据变化时每隔 Min（单位秒）时间上传一次数据；

Max：当数据一直保持不变则每隔 Max（单位秒）时间上传一次数　据；

连接协议：

Product Key：产品秘钥

Device Name：设备名称

Device Secret：设备秘钥

Geolocation：位置信息

ProductKey、DeviceName、DeviceSecret 分别对应 ALiYun 平台上建立的产品和设备属性参数，如图 5-1-3：

图 5-1-3

3. 设置主题属性。右键选中发布主题-->TOPIC0，选中编辑，弹出界面如图 5-1-4；

图 5-1-4

QoS 是消息的发送方（Sender）和接受方（Receiver）之间达成的一个协议。

QoS0：最多一次，代表，Sender 发送的一条消息，Receiver 最多能收到一次，也就是说 Sender 尽力向 Receiver 发送消息，如果发送失败，也就算了；

QoS1：至少一次，代表，Sender 发送的一条消息，Receiver 至少能收到一次，也就是说 Sender 向 Receiver 发送消息，如果发送失败，会继续重试，直到 Receiver 收到消息为止，但是因为重传的原因，Receiver 有可能会收到重复的消息；

QoS2：只有一次，代表，Sender 发送的一条消息，Receiver 确保能收到而且只收到一次，也就是说 Sender 尽力向 Receiver 发送消息，如果发送失败，会继续重试，直到 Receiver 收到消息为止，同时保证 Receiver 不会因为消息重传而收到重复的消息。

注意：

QoS 是 Sender 和 Receiver 之间的协议，而不是 Publisher 和 Subscriber 之间的协议。换句话说，Publisher 发布了一条 QoS1 的消息，只能保证 Broker 能至少收到一次这个消息；而对于 Subscriber 能否至少收到一次这个消息，还要取决于 Subscriber 在 Subscibe 的时候和 Broker 协商的 QoS 等级。

建立发布（PUB）/订阅（SUB）变量表，右键选中发布主题-->TOPIC0，选择编辑，弹出界面如图 5.1.5；

图 5-1-5

导出物模型：把当前变量表导出到文件中（json 格式），当云平台建立产品和设备并建立物模型时需要导入该文件。

添加：在变量表中按照填写信息插入新的变量，并检查变量合法性。

批量添加：可以通过导入物模型的方式批量导入变量。

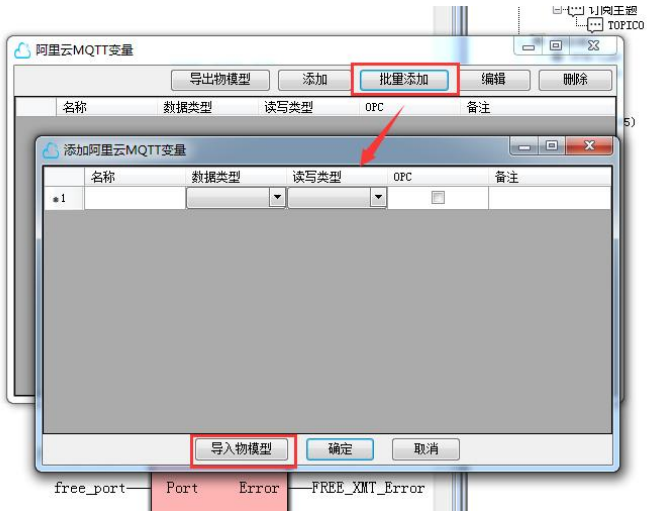


图 5-1-6

导入物模型：可快速导入其他设备物模型数据（json 格式）。

编辑：修改当前选中变量属性。

删除：删除当前选中的变量，并弹出确认删除窗口。

选择添加，弹出变量添加界面:

图 5-1-7

变量表内容说明：

变量名称：在全局变量表里面显示的名称，名字不可重复。

OPC：是否使能 OPC 服务器访问该变量。

数据类型：选择该变量的数据类型，以便分配地址和占用大小；

读写类型：指明该变量的读写类型。

备注：对该变量进行描述。

变量添加及修改时可同步在全局变量表里面生成相应的变量。如图 5-1-8。

至此 MQTT 变量添加完毕。

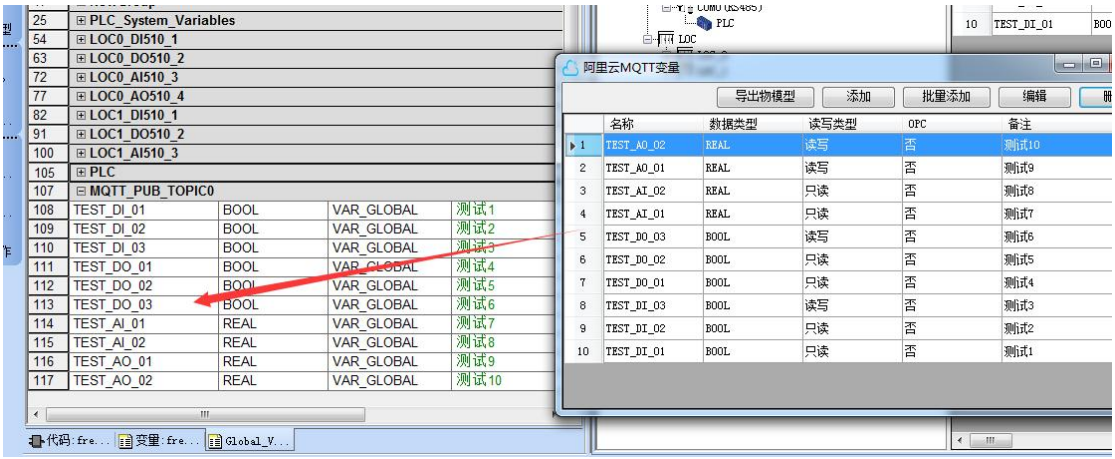


图 5-1-8

5.2 Azure 连接设置

Azure 连接设置同 ALiYun 设置类似。

1. 首先，添加 MQTT 类型。右键点击 MQTT 在弹出的界面中依次选择“添加 MQTT”，选择 Azure，如图 5-2-1。

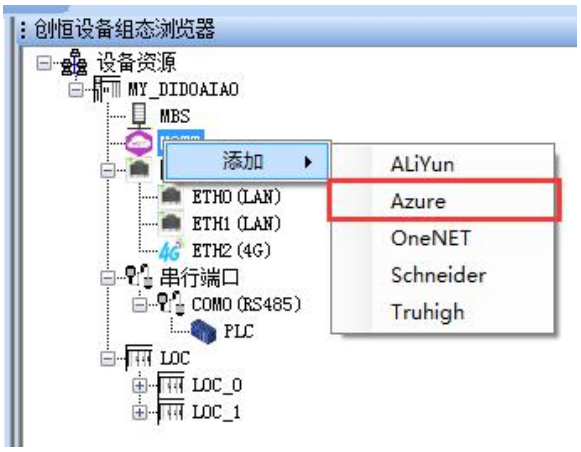


图 5-2-1

2. 然后，编辑 MQTT 属性，如图 5-2-2，

连接保持时间：云平台连接保活时间，单位秒。

数据上传时间间隔：

最小值：当数据变化时每隔 Min（单位秒）时间上传一次数据；

最大值：当数据一直保持不变则每隔 Max（单位秒）时间上传一次数据；

连接字符串：连接 Azure 云平台的字符串秘钥。

图 5-2-2

3. 设置主题属性，右键选中发布主题-->TOPIC0，选中编辑，弹出界面如图 5-2-3；

图 5-2-3

QoS 是消息的发送方（Sender）和接受方（Receiver）之间达成的一个协议。

QoS0：最多一次，代表，Sender 发送的一条消息，Receiver 最多能收到一次，也就是说 Sender 尽力向 Receiver 发送消息，如果发送失败，也就算了；

QoS1：至少一次，代表，Sender 发送的一条消息，Receiver 至少能收到一次，

也就是说 Sender 向 Receiver 发送消息，如果发送失败，会继续重试，直到 Receiver 收到消息为止，但是因为重传的原因，Receiver 有可能会收到重复的消息；

QoS2：只有一次，代表，Sender 发送的一条消息，Receiver 确保能收到而且只收到一次，也就是说 Sender 尽力向 Receiver 发送消息，如果发送失败，会继续重试，直到 Receiver 收到消息为止，同时保证 Receiver 不会因为消息重传而收到重复的消息。

注意：

QoS 是 Sender 和 Receiver 之间的协议，而不是 Publisher 和 Subscriber 之间的协议。换句话说，Publisher 发布了一条 QoS1 的消息，只能保证 Broker 能至少收到一次这个消息；而对于 Subscriber 能否至少收到一次这个消息，还要取决于 Subscriber 在 Subscibe 的时候和 Broker 协商的 QoS 等级。

建立发布（PUB）/订阅（SUB）变量表，右键选中发布主题-->TOPIC0，选择编辑，弹出界面如图 5-2-4；

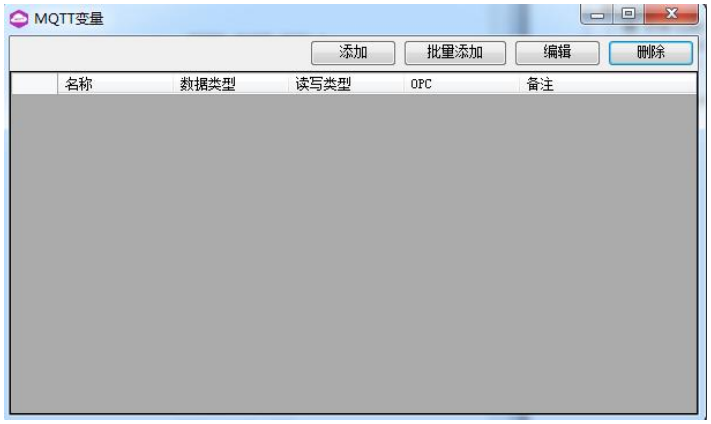


图 5-2-4

添加：在变量表中按照填写信息插入新的变量，并检查变量合法性。

批量添加：可以通过预编辑变量然后批量导入变量。

编辑：修改当前选中变量属性。

删除：删除当前选中的变量，并弹出确认删除窗口。

选择添加，弹出变量添加界面:

图 5-2-5

变量表内容说明：

名称：在全局变量表里面显示的名称，名字不可重复。
OPC：是否使能 OPC 服务器访问该变量。
数据类型：选择该变量的数据类型，以便分配地址和占用大小；
读写类型：指明该变量的读写类型。
描述：对该变量进行描述。
变量添加及修改时可同步在全局变量表里面生成相应的变量。如图 5-2-6。
至此 MQTT 变量添加完毕。

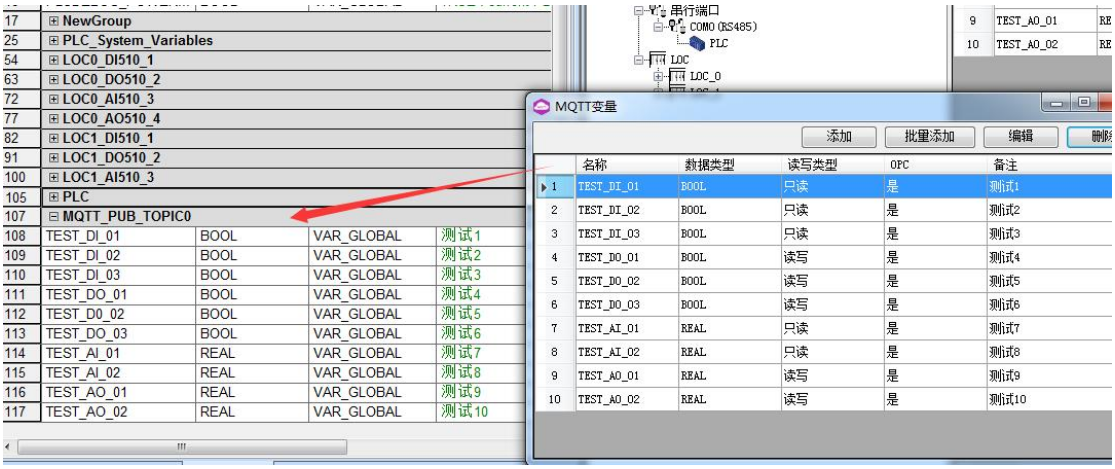


图 5-2-6

5.3 OneNET 连接设置

OneNET 连接设置同 ALiYun 设置类似。

1. 首先，添加 MQTT 类型。右键点击 MQTT 在弹出的界面中依次选择“添加 MQTT”，选择 OneNET，如图 5-3-1。

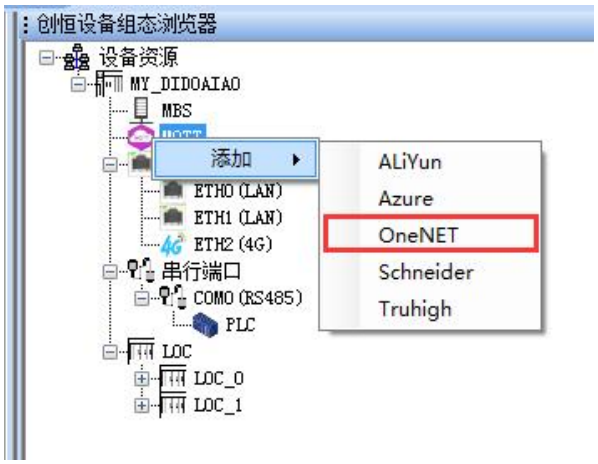


图 5-3-1

2. 然后，编辑 MQTT 属性，如图 5-3-2，

图 5-3-2

连接保持时间：云平台连接保活时间，单位秒。

数据上传时间间隔：

最小值：当数据变化时每隔 Min（单位秒）时间上传一次数据；

最大值：当数据一直保持不变则每隔 Max（单位秒）时间上传一次数据；

连接协议：

设备名称：当前 PLC 的设备 ID，对应 OneNET 平台里面的设备 ID；

产品 ID：平台分配的产品 ID；

设备 KEY：设备秘钥，对应 OneNET 平台的设备秘钥；

过期时间：过期时间参数，设置过期时间。

如图 4-3-3：

图 5-3-3

3. 设置主题属性，右键选中发布主题-->TOPIC0，选中编辑，弹出界面如下：

图 5-3-4

定义同 Azure 连接说明，在此不再重复。

建立发布（PUB）/订阅（SUB）变量表，右键选中发布主题-->TOPIC0，选择编辑，弹出界面如图 5-3-5；

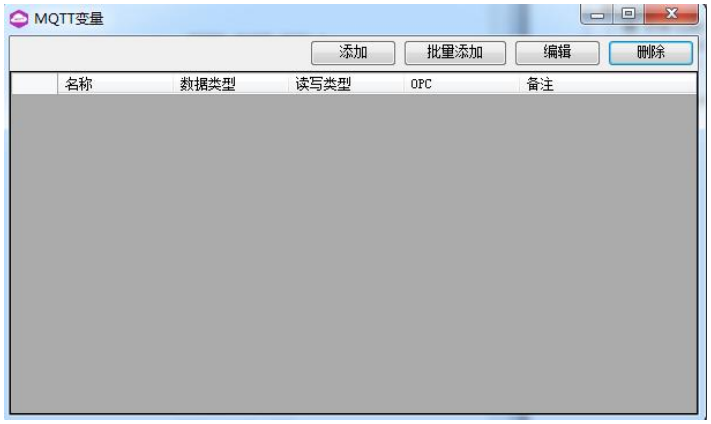


图 5-3-5

添加：在变量表中按照填写信息插入新的变量，并检查变量合法性。
批量添加：可以通过预编辑变量然后批量导入变量。
编辑：修改当前选中变量属性。
删除：删除当前选中的变量，并弹出确认删除窗口。
选择添加，弹出变量添加界面:

图 5-3-6

变量表内容说明：
名称：在全局变量表里面显示的名称，名字不可重复。
OPC：是否使能 OPC 服务器访问该变量。
数据类型：选择该变量的数据类型，以便分配地址和占用大小；
读写类型：指明该变量的读写类型。
描述：对该变量进行描述。

5.4 Schneider 连接设置

Schneider 连接设置同 ALiYun 设置类似。

1. 首先，添加 MQTT 类型。右键点击 MQTT 在弹出的界面中依次选择“添加 MQTT”，选择 Schneider，如图 5-4-1。

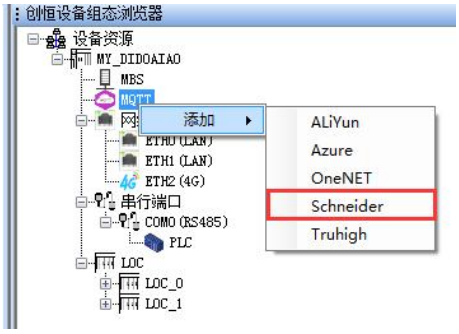


图 5-4-1

2. 然后，编辑 MQTT 属性，如图 5-4-2：

图 5-4-2

连接保持时间：云平台连接保活时间，单位秒。

数据上传时间间隔：

最小值：当数据变化时每隔 Min（单位秒）时间上传一次数据；

最大值：当数据一直保持不变则每隔 Max（单位秒）时间上传一次数据；

Client ID：设备 ID

User Name：用户名

Password：用户密码

3. 主题属性，右键选中发布主题-->TOPIC0，选中编辑，弹出界面如下：

图 5-4-3

定义同 Azure 连接说明，在此不再重复。

建立发布（PUB）/订阅（SUB）变量表，右键选中发布主题-->TOPIC0，选择编辑，弹出界面如图 5-4-4；

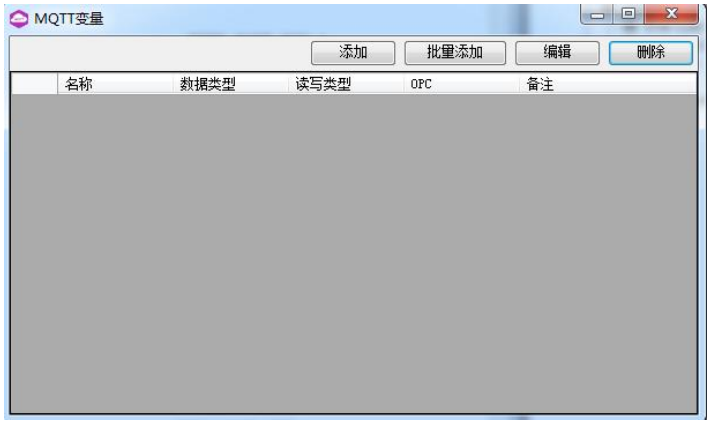


图 5-4-4

添加：在变量表中按照填写信息插入新的变量，并检查变量合法性。
批量添加：可以通过预编辑变量然后批量导入变量。
编辑：修改当前选中变量属性。
删除：删除当前选中的变量，并弹出确认删除窗口。
选择添加，弹出变量添加界面:

图 5-4-5

变量表内容说明：
名称：在全局变量表里面显示的名称，名字不可重复。
OPC：是否使能 OPC 服务器访问该变量。
数据类型：选择该变量的数据类型，以便分配地址和占用大小；
读写类型：指明该变量的读写类型。
描述：对该变量进行描述。

5.5 Truhigh 连接设置

Truhigh 创恒云连接设置如下所述。

1. 首先，添加 MQTT 类型。右键点击 MQTT 在弹出的界面中依次选择“添加 MQTT”，选择 Truhigh，如图 5-5-1。

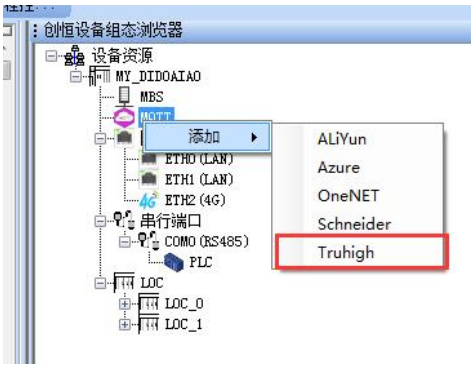


图 5-5-1

在弹出“编辑创恒 MQTT”，如图 5-5-2：

图 5-5-2

IP/URL：云平台 MQTT 服务器地址。

自动获取地址：是否启用自动获取 URL、端口号、用户名和密码，根据自动识别设备 ID 自动获取 URL、端口号、用户名和密码，**自动获取的信息保存在 PLC 内部，不在软件上显示，自动获取功能只针对连接 Truhigh 云平台。**

端口号：服务器端口号。

设备 ID：设备表示 ID，需要和云平台一致。

连接超时：云平台连接超时时间，单位秒。

保活时间：心跳间隔时间（秒），定期向代理服务器发送心跳包的时间间隔，单位秒。

自动清理：表示设备离线并重连后是否接收离线消息。

自动重连：当检测到超时后自动重连服务器。

空中最大报文数量：允许网路链路上未确认的报文的上限，达到上限，消息服务器将不再发送后续的报文。

MQTT 版本：默认 V5.

用户验证：是否启用用户名和密码验证模式。

SSL/TLS：是否启用 SSL/TLS 协议。

协议：TLS 协议版本，默认为 TLSv1.2

双向验证：是否启用服务器和客户端双向验证模式。

根证书：由云平台生成的登录认证根证书文件。

客户端证书：由云平台生成的客户端认证证书文件。

客户端私钥：由云平台生成的客户端认证秘钥文件。

2. MQTT 变量管理，以 Truhigh MQTT 为例。

右键点击变量主题下的“TOPIC0”，选择变量。

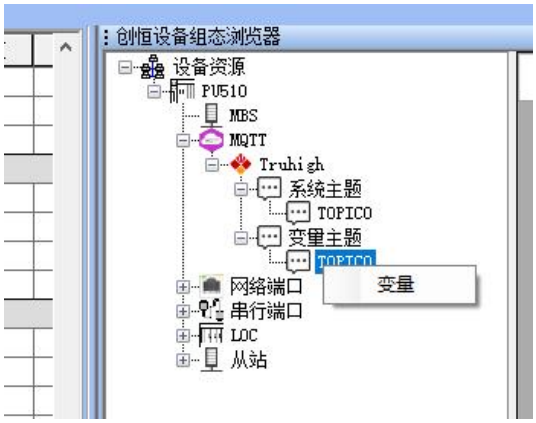


图 5-5-3

弹出 MQTT 变量界面：

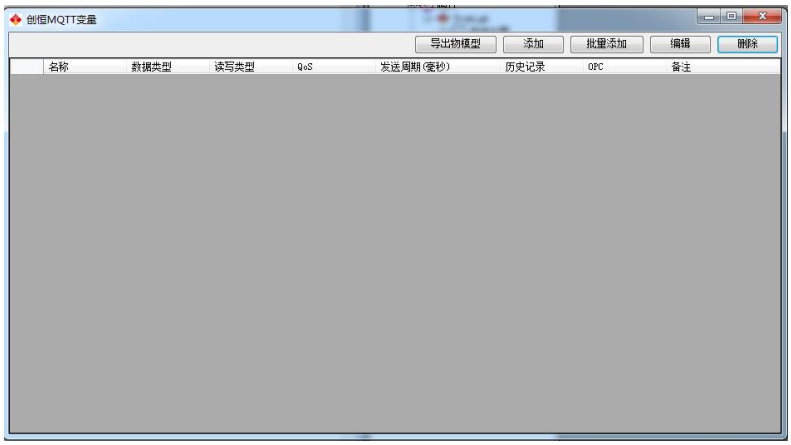


图 5-5-4

1）添加：单个添加变量：

图 5-5-5

名称：变量名称，需要和云平台变量一致。

数据类型：基本数据类型。

读写类型：只读（只发布变量）、只写（只订阅变量）、读写（订阅/发布变量）

只读变量（发布主题）：数据流向为从客户端（设备）到服务器。

只写变量（订阅主题）：数据流向为从服务器到客户端（设备）。

读写变量：数据流向为双向，同时支持订阅和发布，服务器和设备都可以读写该变量。

Qos：服务质量，

最多一次：发送方发送的一条消息，接收方最多能收到一次，也就是说发送方尽力向接收发送消息，如果发送失败，则不再发送。

至少一次：发送方发送的一条消息，接收方至少能收到一次，也就是说发送方向接收方发送消息，如果发送失败，会继续重试，直到接收方收到消息为止，但是因为重传的原因，接收方有可能会收到重复的消息。

只有一次：发送方发送的一条消息，接收方确保能收到而且只收到一次，也就是说发送方尽力向接收方发送消息，如果发送失败，会继续重试，直到接收方收到消息为止，同时保证接收方不会因为消息重传而收到重复的消息。

发送周期：数据发布间隔时间。

历史记录：当连接断开时是否作为历史数据保存，连接正常使重发历史数据。

OPC：是否作为 OPC 变量。

2）批量添加：点击批量添加按钮弹出批量添加界面：

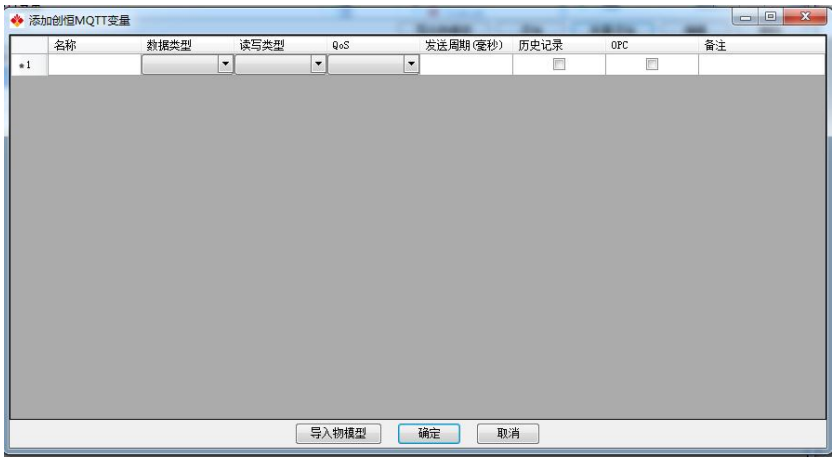


图 5-5-6

图 5-5-7

填写变量后，点击确定按钮，系统自动批量添加到全局变量表。

导入物模型：可选择云平台生成的物模型，进行批量导入变量：

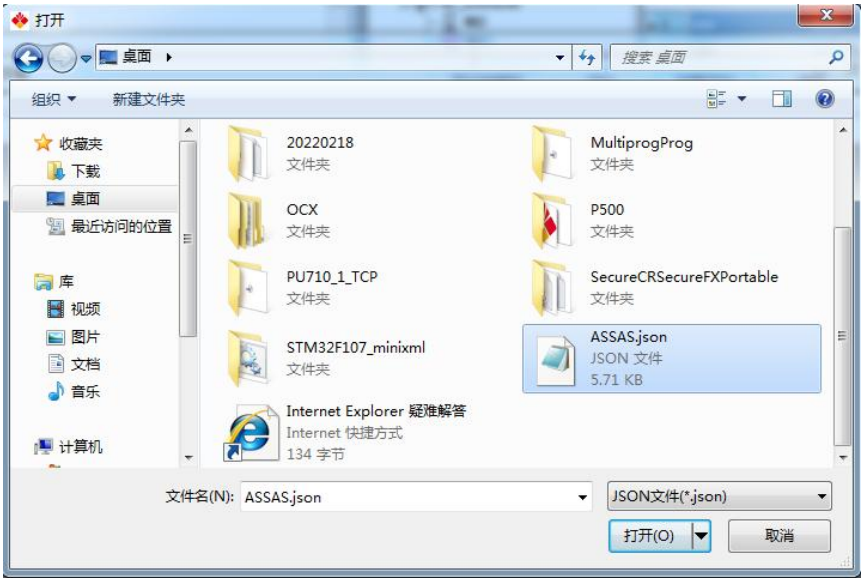


图 5-5-8

点击导入物模型，选择文件->打开，即可导入变量。

3）导出物模型：可把当前 MQTT 变量以云平台物模型格式导出，以便云平台使用。

图 5-5-9

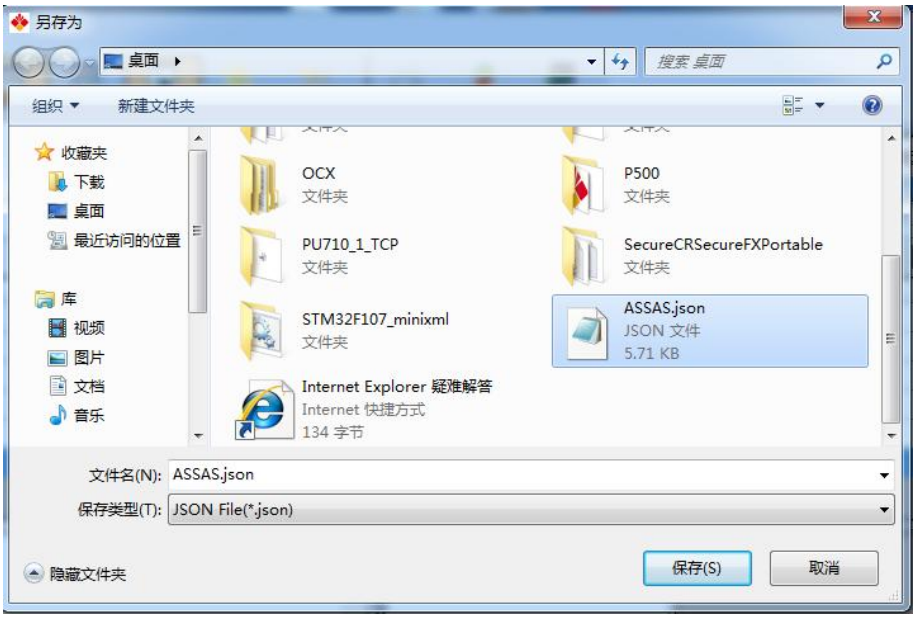


图 5-5-10

3.　系统主题，可选择是否把系统信息作为 MQTT 变量。

右键选择系统主题->TOPIC0->变量。

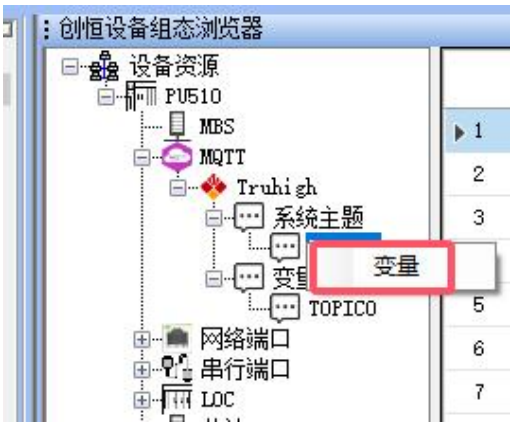


图 5-5-11

弹出系统主题变量编辑界面：

图 5-5-12

启用：是否作为 MQTT 变量。
Qos：服务质量，最多一次、至少一次、只有一次。
发送周期：数据发布间隔时间。
历史记录：当连接断开时是否作为历史数据保存，连接正常使重发历史数据。
OPC：是否作为 OPC 变量。