

# RS485/RS422-EC 使用手册

-- V1.2



目录

一、 产品概述 ..... 1

    1.1、 产品简介 ..... 1

    1.2、 特点功能 ..... 1

    1.3、 应用场景 ..... 1

二、 产品规格 ..... 2

    2.1、 产品参数 ..... 2

    2.2、 各部分说明 ..... 3

        2.2.1、 端子说明 ..... 3

        2.2.2、 指示灯说明 ..... 3

        2.2.3、 网口指示灯 ..... 3

三、 产品功能 ..... 4

    3.1、 Modbus RTU 主站功能 ..... 4

    3.2、 透传 ..... 4

    3.3、 按键复位功能 ..... 4

    3.4、 Boot 功能（固件升级） ..... 5

四、 参数配置说明 ..... 6

    4.1、 配置前准备 ..... 6

    4.2、 对象字典 ..... 6

    4.3、 使用说明 ..... 9

五、 Twincat 使用入门指导 ..... 10

    5.1、 添加模块 ..... 10

帮助 100 万家企业实现智能制造

## 一、产品概述

### 1.1、产品简介

RS485/RS422-EC 是一款 EtherCAT 转 MODBUS RTU 通讯功能的模块，是一款经济稳定、安装简易，适用性强的产品。最多支持 10 个从站模块，每个从站模块可支持 64 位线圈、64 位离散输入、16 个保持寄存器、16 个输入寄存器当用户从站 IO 点数大于单一从站模块可设置 IO 点数时，可进行扩展使用，即支持单一模块最大 640 位线圈，640 位离散输入、160 个保持寄存器、160 个输入寄存器。

### 1.2、特点功能

- 支持 Modbus RTU 主站功能，采用标准 Modbus RTU 通讯协议，通讯接口可设置
- 支持 RS485 或 RS422 透明传输功能
- 模块波特率、数据格式、从站相关信息可通过 EtherCAT 上位机进行设置
- 相关用户信息可永久保存，上电即可用
- 采用符合 DIN 35MM 标准导轨安装方式，方便安装
- 电源电路采用防反接设计
- 广泛用于工业现场设备的信号采集和控制

### 1.3、应用场景

RS485/RS422-EC 模块可应用范围很广，如：PLC 控制、工业自动化、楼宇自控、POS 系统、电力监控、门禁医疗、考勤系统、自助银行系统、电信机房监控、信息家电、LED 信息显示设备、测量仪表及环境动力监控系统、售饭系统等需要 RS485 或 RS422 串口转 EtherCAT 总线的设备或系统。



## 二、产品规格

### 2.1、产品参数

|                  |                                     |
|------------------|-------------------------------------|
| 主要参数             |                                     |
| 网口参数（RS422 通讯参数） |                                     |
| 接口类型             | RJ45                                |
| 通讯协议             | EtherCAT                            |
| 串口参数（RS422 通讯参数） |                                     |
| 接口类型             | RS422（5.08mm 间距工业级接线端子）             |
| 波特率              | 1200~4.6875Mbps                     |
| 通信格式             | 默认 8 位数据，1 位停止，无校验                  |
| 传输距离             | 波特率 100kb/s 时，422 串口通讯 1200 米，以实际为准 |
| 串口参数（RS485 通讯参数） |                                     |
| 接口类型             | RS485（5.08mm 间距工业级接线端子）             |
| 波特率              | 1200~4.6875Mbps                     |
| 通信格式             | 默认 8 位数据，1 位停止，无校验                  |
| 传输距离             | 波特率 9600 时，485 串口通讯 1200 米，以实际为准    |
| 电源参数             |                                     |
| 工作电压             | DC 24V;带防反接保护                       |
| 功耗               | 2W~4W                               |
| 工作环境             |                                     |
| 工作温度             | -10℃~50℃                            |
| 存储温度             | -20℃~70℃                            |
| 其他               |                                     |
| 安装方式             | 导轨                                  |
| 尺寸               | 29MM(长)*92MM(宽)*65MM(高)，以实物为准       |

2.2、各部分说明

2.2.1、端子说明

| 端子标   | 功能说明            |
|-------|-----------------|
| 24V+  | 12-28V 直流供电电源正极 |
| 0V    | 12-28V 直流供电电源负极 |
| B-    | RS485 反向端       |
| A+    | RS485 正向端       |
| T-    | RS422 接收反向端     |
| T+    | RS422 接收正向端     |
| R-    | RS422 发送反向端     |
| R+    | RS422 发送正向端     |
| Reset | 复位按钮            |

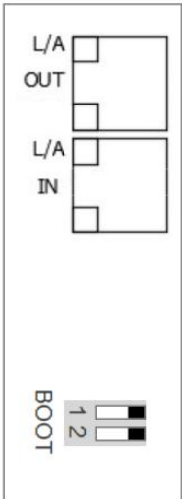
2.2.2、指示灯说明

| 名称    | 说明               |
|-------|------------------|
| SYS   | 系统指示灯            |
| RUN   | EtherCAT 通讯状态指示灯 |
| ERR   | EtherCAT 通讯错误指示灯 |
| RS422 | RS422 工作指示灯      |
| RS485 | RS485 工作指示灯      |

2.2.3、网口指示灯

IN 为输入接上级设备

OUT 为输出接下级设备



### 三、产品功能

#### 3.1、Modbus RTU 主站功能

本模块 RTU 主站最多可支持 10 个从站模块，每个模块可设置 64 路数字（线圈）输出、64 路离散输入、16 个输入寄存器和 16 个保持寄存器。

使用 Modbus RTU 功能时，每个从站会生成各自读写命名，每条命令以一定周期（可设）进行轮询，当周期过小时，主站会给出警告，但是仍然会在超时后尽快将下一条指令发出。

使用 Modbus RTU 功能时，写功能调用不能小于轮询周期的 2 倍，不然会出现一直进行写功能，回读数据不刷新，详细设置方式见第七章节。

Modbus 通讯可选使用 RS422 或者 RS485 接口。

#### 3.2、透传

当使用透传模式时，本模块在收到 EtherCAT 主站发出的数据后，将直接通过所选串口发出，接收到串口数据后，将数据直接发送到 EtherCAT 主站。

#### 3.3、按键复位功能

模块上电时，按住 Reset 复位按钮，直到模块 SYS 灯出现双闪后松开按钮，再将模块断电至少 3S 后上电，模块即恢复出厂参数，如下表。

| 参数名称 | 参数默认值                     |
|------|---------------------------|
| 串口参数 | 波特率 9600，校验位 None,停止位 1 位 |

|      |          |
|------|----------|
| 从站参数 | 恢复默认，全 0 |
| 选用接口 | RS485    |
| 选用模式 | 透传模式     |


模块上电之前或者上电后 3 秒内，先将 SW1 拨至“1”并保持 3 秒以上，此时 SYS 灯、RS422、RS485 灯均以 1 秒间隔闪烁，3 秒后 SYS 灯由 1 秒间隔闪烁变为 0.2 秒间隔闪烁，进入 boot 状态成功。



|  |   |                   |          |        |                     |   |
|--|---|-------------------|----------|--------|---------------------|---|
|  |   |                   |          |        |                     | <p>用于选择串口数据格式<br/>第一位是数据长度，第二位是校验，第三位是停止位</p> <p>1: 8E1<br/>2: 8O1<br/>3: 8N1<br/>4: 8E1.5<br/>5: 8O1.5<br/>6: 8N1.5<br/>7: 8E2<br/>8: 8O2<br/>9: 8N2<br/>10: 7E1<br/>11: 7O1<br/>12: 7E1.5<br/>13: 7O1.5<br/>14: 7E2<br/>15: 7O2<br/>16: 9N1<br/>17: 9N1.5<br/>18: 9N2</p> <p>当 8000: 1 设置为 0x10 时，波特率使用本子索引中的值，直接输入十进制数即可，如 115200 对应 115.2kbps 波特率</p> |
|  | 2 | dataframe         | enum     | Pre-Op | min:1<br>max:18     |   |
|  | 3 | explicit baudrate | uint32_t | Pre-Op | --                  | <p>轮询时间，在 modbus 模式下，每一条指令轮询设置的时间，关于轮询时间详细设置方式见第九章</p>  |
|  | 4 | polling time      | uint16_t | Pre-Op | min:50<br>max:65535 |   |
|  | 5 | Slave Reset       | bool     | Pre-Op | min:0<br>max:1      | <p>将该子索引给 1，清除保存的用户参数，该位自复位</p>   |
|  | 6 | Error Reset       | bool     | Op     | min:0<br>max:1      | <p>该子索引给 1，将清除 4070 中保存的错误代码，该位自复位</p>  |
|  | 7 | Device Mode       | enum     | Pre-Op | min:0<br>max:1      | <p>选择模块工作模式，<br/>0: modbus 模式<br/>1: 透明传输模式</p>   |
|  | 8 | Device Interface  | enum     | Pre-Op | min:0<br>max:1      | <p>选择模块工作接口<br/>0: RS485 接口<br/>1: RS422 接口</p>   |



|                 |   |                              |         |        |                    |                                       |
|-----------------|---|------------------------------|---------|--------|--------------------|---------------------------------------|
| 800x<br>(x=1-A) | 1 | Slave addr                   | Byte    | Pre-Op | min:0<br>max:255   | 从站接口地址, 1-255, 不可为 0, 为 0 时从站无效, 关闭从站 |
|                 | 2 | coil is readable             | bool    | Pre-Op | min:0<br>max:1     | 线圈是否需要回读<br>0: 关闭回读<br>1: 开启回读功能      |
|                 | 3 | hold reg readable            | bool    | Pre-Op | min:0<br>max:1     | 保持寄存器是否需要回读,<br>0: 关闭回读<br>1: 开启回读功能  |
|                 | 5 | coil start address           | uint6_t | Pre-Op | min:0<br>max:65535 | 从站 modbus 协议中线圈起始地址                   |
|                 | 6 | the number of coil           | Byte    | Pre-Op | min:0<br>max:64    | 从站 modbus 协议中线圈数量                     |
|                 | 8 | Discrete input start addr    | uint6_t | Pre-Op | min:0<br>max:65535 | 从站 modbus 协议中离散输入起始地址                 |
|                 | 9 | the number of DI             | Byte    | Pre-Op | min:0<br>max:64    | 从站 modbus 协议中离散输入数量                   |
|                 | B | Input register start addr    | uint6_t | Pre-Op | min:0<br>max:65535 | 从站 modbus 协议中输入寄存器起始地址                |
|                 | C | the number of Input register | Byte    | Pre-Op | min:0<br>max:16    | 从站 modbus 协议中输入寄存器数量                  |
|                 | E | Hold register start addr     | uint6_t | Pre-Op | min:0<br>max:65535 | 从站 modbus 协议中保持寄存器起始地址                |
|                 | F | the number of Hold register  | Byte    | Pre-Op | min:0<br>max:16    | 从站 modbus 协议中保持寄存器数量                  |

 **Note:** 在单一对象字典无法将某一个从站数据读取完毕时, 可以使用多个对象字典读取同一个从站中的值, 将寄存器和线圈的起始地址进行相应偏移即可。

4.3、使用说明

DataValid 功能描述:

DataValid 为自定义的功能配置标志，数据类型为 byte (8-bit)，高 4 位用来区分不同的 modbus 从站；低 4 位区分 modbus 不同功能码。

DataValid 修改说明:

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 第 7 位 | 第 6 位 | 第 5 位 | 第 4 位 | 第 3 位 | 第 2 位 | 第 1 位 | 第 0 位 |
| x4    | x3    | x2    | x1    | a     | b     | c     | d     |

读功能 Bits 7:4      x4x3x2x1:取对应对象字典 (0x8001~0x800a) & 0x00ff 后的值，即 1-10。

- Bit    3

a: 读输入寄存器的值

0: 此次显示的值是之前通过 modbus 读取的输入寄存器的值

1: 此次显示的值是刚通过 modbus 读取上来的输入寄存器的值
- Bit    2

b: 读离散输入的值

0: 此次显示的值是之前通过 modbus 读取的离散输入的值

1: 此次显示的值是刚通过 modbus 读取上来的离散输入的值
- Bit    1

c: 读保持寄存器的值

0: 此次显示的值是之前通过 modbus 读取的保持寄存器的值

1: 此次显示的值是刚通过 modbus 读取上来的保持寄存器的值
- Bit    0

d: 读线圈的值

0: 此次显示的值是之前通过 modbus 读取的线圈的值

1: 此次显示的值是刚通过 modbus 读取上来的线圈的值



写功能 Bits 7:4 x4x3x2x1:取对应对象字典 (0x8001~0x800a) & 0x00ff 后的值, 即 1-10。

Bit 3 a: 保留

Bit 2 b: 保留

Bit 1 c: 写保持寄存器, 0 和 1 交替写值, 开始一次写保持寄存器

Bit 0 d: 写线圈, 0 和 1 交替写值, 开始一次写线圈

**示例:** 写一次对象字典 0x8003 所设置的从站保持寄存器, 第一次 DataValid=0x32, 第二次 DataValid=0x30, 第三次 DataValid=0x32。

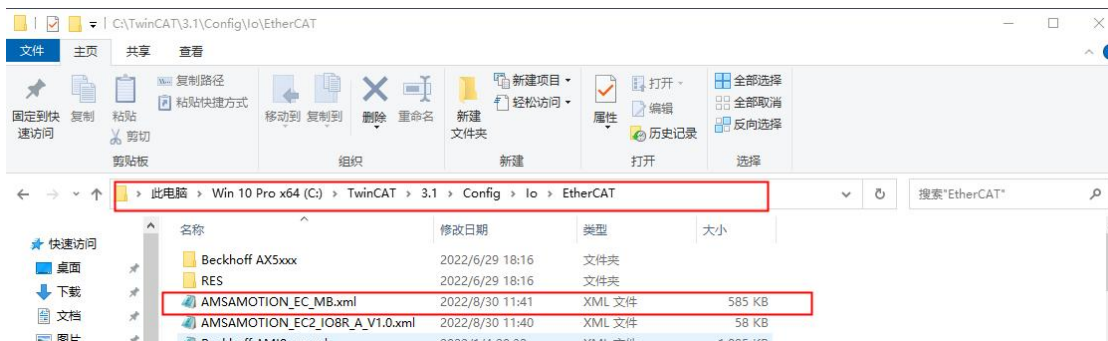
写一次对象字典 0x8003 所设置的从站线圈, 第一次 DataValid=0x31, 第二次 DataValid=0x30, 第三次 DataValid=0x31。

写一次对象字典 0x8003 所设置的从站线圈和保持寄存器, 第一次 DataValid=0x33, 第二次 DataValid=0x30, 第三次 DataValid=0x33 (默认从上电开始)。

## 五、Twincat 使用入门指导

### 5.1、添加模块

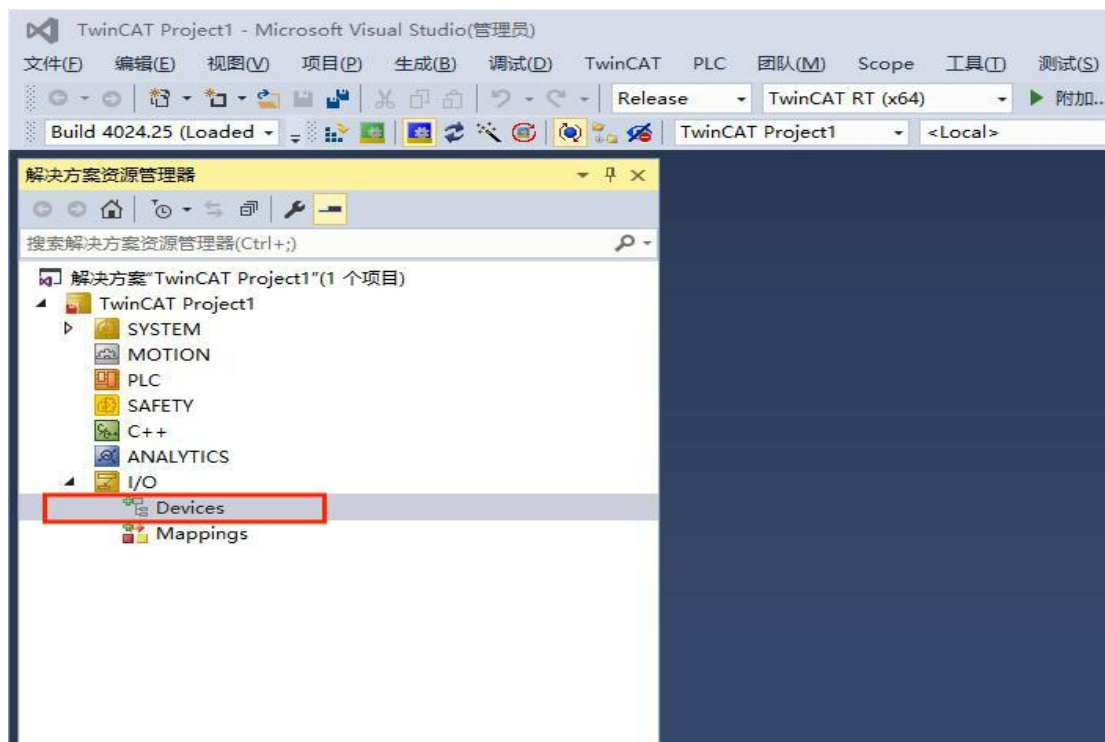
将 EtherCAT 所需 XML 拷贝至主站安装文件夹下的模块 xml 文件夹下, 以 TwinCAT 3 为例, 如下图:



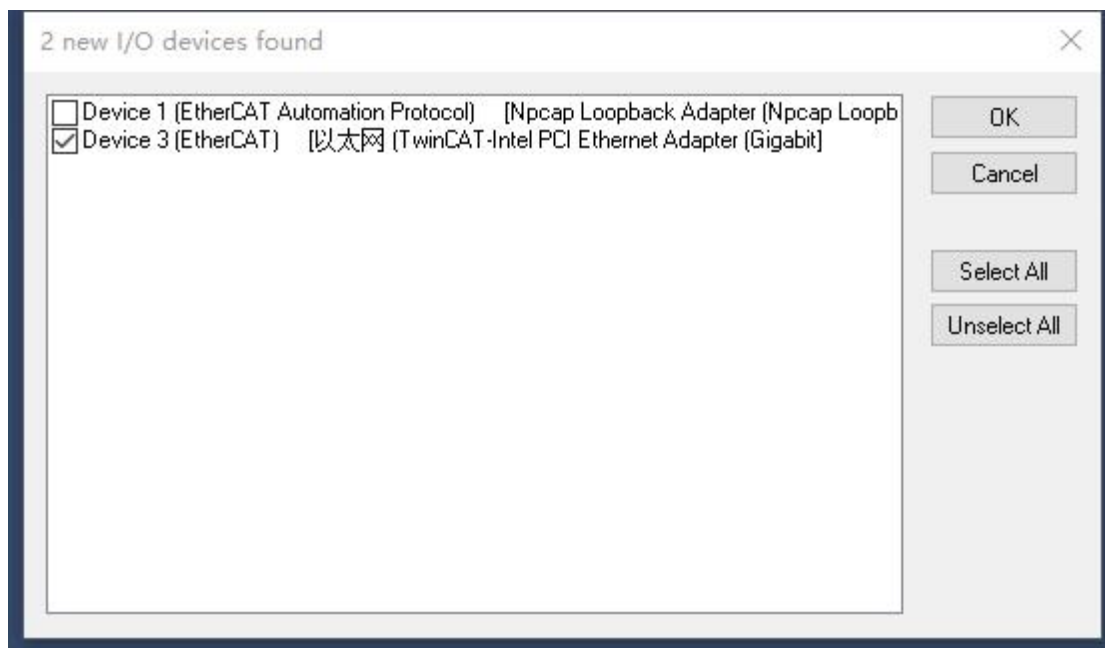
右击桌面右下角 TwinCAT 图标，选择“System Manager”，打开 TwinCAT 软件，如下图所示：



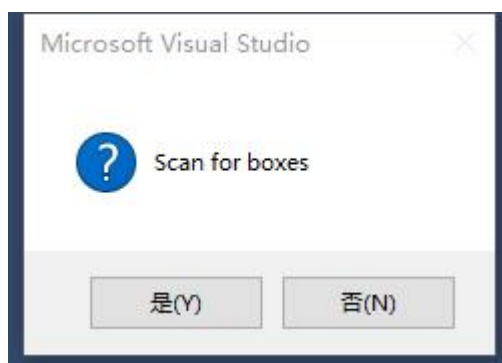
打开 TwinCAT 软件后，新建工程或打开已有工程，在“I/O → Devices”处右击选择“Scan Devices”



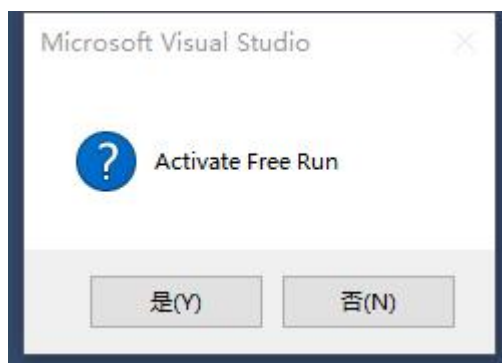
勾选当前扫描到的“本地连接”网卡



弹出窗口“Scan for boxes”选择“是”

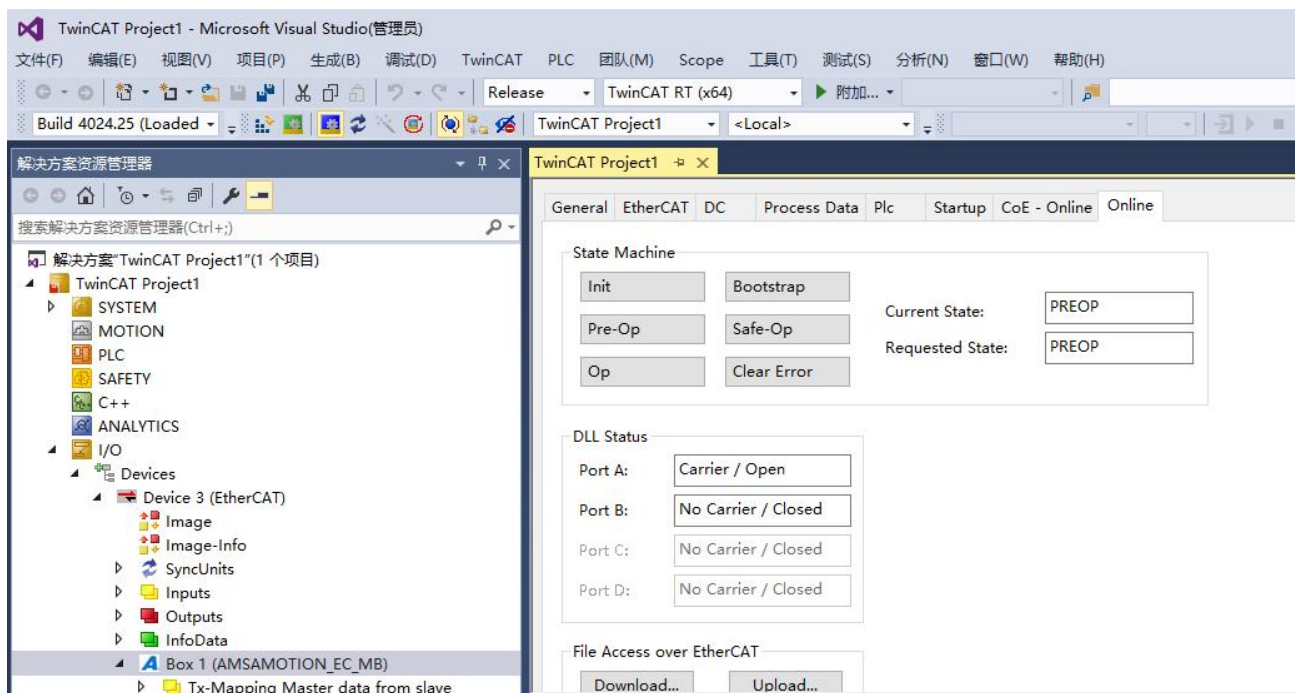


弹出窗口“Activate Free SYS”选择“是”

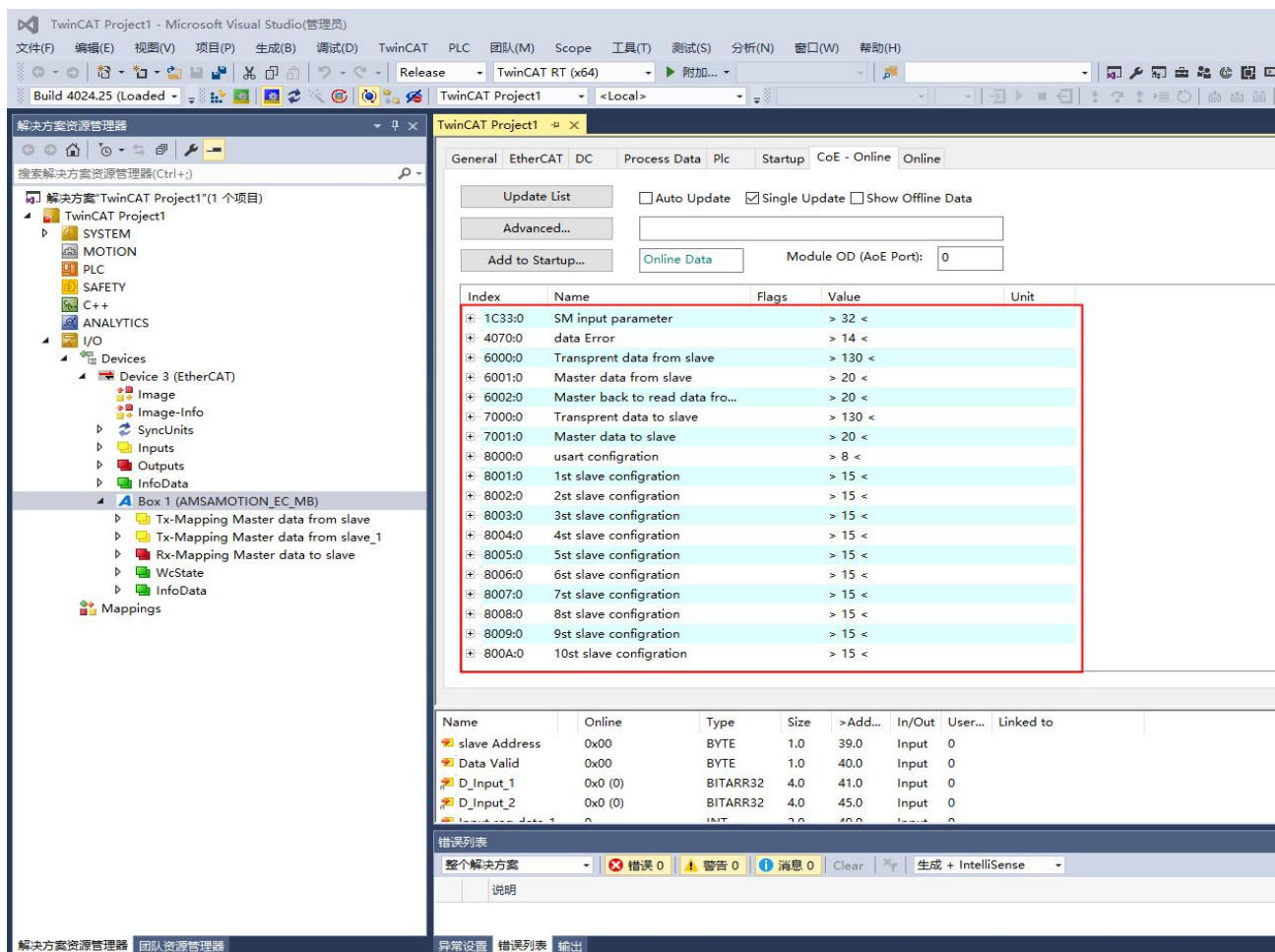


单击 Box1，进入 Online 选项，当前状态 Pre-Op，如果在其他状态，请切回 Pre-Op：

⚠Note: 如果不是处于 Pre-Op 模式下，大部分对象是不可修改的，若修改，TwinCAT 会报错。



单击 CoE-Online，下拉可以见到本模块所有对象字典：



## 5.2、ModBus 功能

本例将其设置为 Modbus 模式，从站使用一个 16 位数字输入输出模块，示例用模块参数如下：

|                 |       |
|-----------------|-------|
| Modbus 从站地址     | 1     |
| Modbus 从站接口     | RS485 |
| Modbus 从站波特率    | 9600  |
| Modbus 从站数据格式   | 8N1   |
| Modbus 从站线圈数量   | 16    |
| Modbus 从站离散输入数量 | 16    |

△Note：在单一对象字典无法将某一个从站数据读取完毕时，可以使用多个对象字典读取同一个从站中的值，将寄存器和线圈的起始地址进行相应偏移即可。

1)单击展开对象字典 8000 对象字：

|         |                     |                     |
|---------|---------------------|---------------------|
| 8000:0  | usart configuration | > 8 <               |
| 8000:01 | Baudrate            | RW 9600 Baud (3)    |
| 8000:02 | dataframe           | RW 8N1 (3)          |
| 8000:03 | Explicit baudrate   | RW 9600             |
| 8000:04 | Polling time        | RW 0x0032 (50)      |
| 8000:05 | Slave Reset         | RW FALSE            |
| 8000:06 | Error Reset         | RW FALSE            |
| 8000:07 | Device Mode         | RW pass-through (1) |
| 8000:08 | DEVICE Interface    | RW RS485 (0)        |

2)选择对象字 8000：1，双击下图“1”处，然后选择“2”处选项，然后点击 OK：

The screenshot shows the 'Set Value Dialog' box with the following fields:

- Dec: 3
- Hex: 0x0003
- Enum: 9600 Baud (selected)
- Bool: (empty)
- Binary: (empty)
- Bit Size: 115.2 kBaud (selected)

The 'Enum' dropdown list includes the following options: 2400 Baud, 4800 Baud, 9600 Baud, 12.2 kBaud, 14.4 kBaud, 19.2 kBaud, 38.4 kBaud, 57.6 kBaud, and 115.2 kBaud. The '115.2 kBaud' option is highlighted with a red box.



3)选择对象字 8000: 7，双击下图“1”处，然后选择“2”处选项，然后点击 OK：

| Index   | Name                      | Flags | Value            | Unit |
|---------|---------------------------|-------|------------------|------|
| 7000:0  | Transporent data to slave |       | > 130 <          |      |
| 7001:0  | Master data to slave      |       | > 20 <           |      |
| 8000:0  | usart configuration       |       | > 8 <            |      |
| 8000:01 | Baudrate                  | RW    | 115,2 kBaud (9)  |      |
| 8000:02 | dataframe                 | RW    | 8N1 (3)          |      |
| 8000:03 | Explicit baudrate         | RW    | 9600             |      |
| 8000:04 | Polling time              | RW    | 0x0032 (50)      |      |
| 8000:05 | Slave Reset               | RW    | FALSE            |      |
| 8000:06 | Error Reset               | RW    | FALSE            |      |
| 8000:07 | Device Mode               | RW    | pass-through (1) |      |
| 8000:08 | DEVICE Interface          | RW    | RS485 (0)        |      |
| 8001:0  | 1st slave configuration   |       | > 15 <           |      |
| 8002:0  | 2st slave configuration   |       | > 15 <           |      |
| 8003:0  | 3st slave configuration   |       | > 15 <           |      |
| 8004:0  | 4st slave configuration   |       | > 15 <           |      |
| 8005:0  | 5st slave configuration   |       | > 15 <           |      |
| 8006:0  | 6st slave configuration   |       | > 15 <           |      |
| 8007:0  | 7st slave configuration   |       | > 15 <           |      |

Set Value Dialog

Dec:     
Hex:   
Enum: 

pass-through  
Modbus  
pass-through

  
Boot: ☐ 0 ☐ 1   
Binary:   
Bit Size: ☒ 1 ☐ 8 ☐ 16 ☐ 32 ☐ 64 ☐ ?

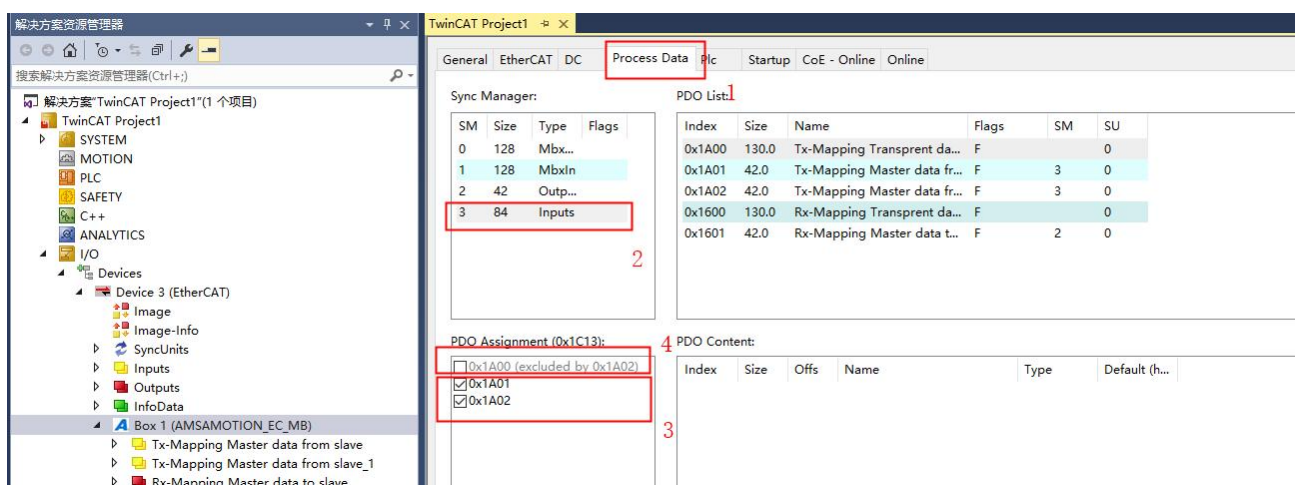
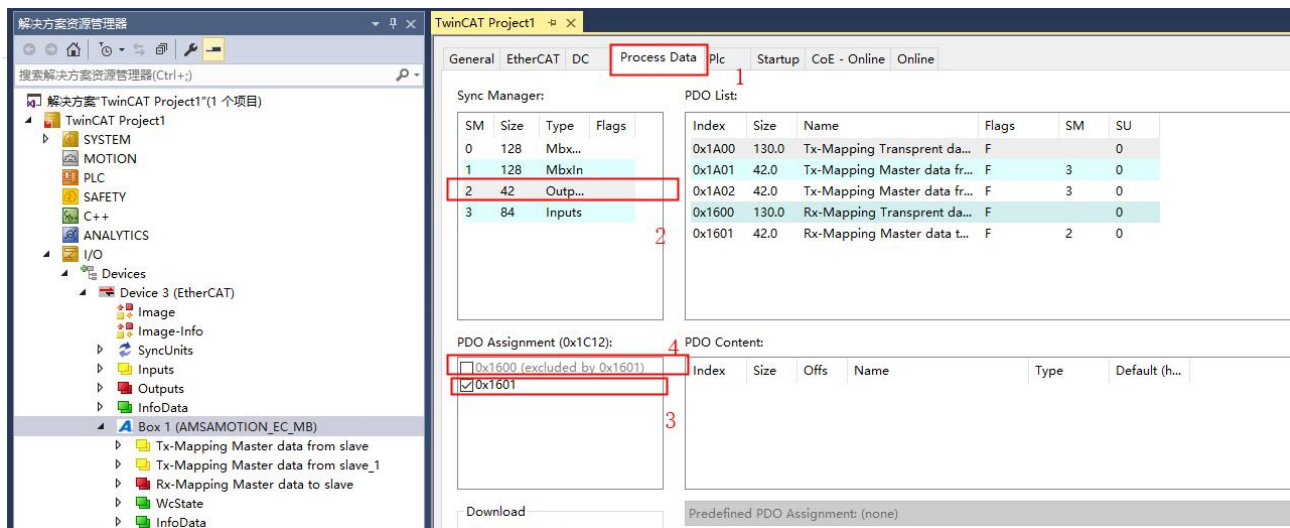
4)完成后对象字 8000 所有子索引如下图：

|         |                       |                    |
|---------|-----------------------|--------------------|
| 8000:0  | usart configuration   | > 8 <              |
| 8000:01 | Baudrate              | RW 115,2 kBaud (9) |
| 8000:02 | dataframe             | RW 8N1 (3)         |
| 8000:03 | Explicit baudrate     | RW 9600            |
| 8000:04 | Polling time          | RW 0x0032 (50)     |
| 8000:05 | Slave Reset           | RW FALSE           |
| 8000:06 | Error Reset           | RW FALSE           |
| 8000:07 | Device Mode           | RW Modbus (0)      |
| 8000:08 | DEVICE Interface      | RW RS485 (0)       |
| 8001:0  | 1st slave confiration | > 15 <             |

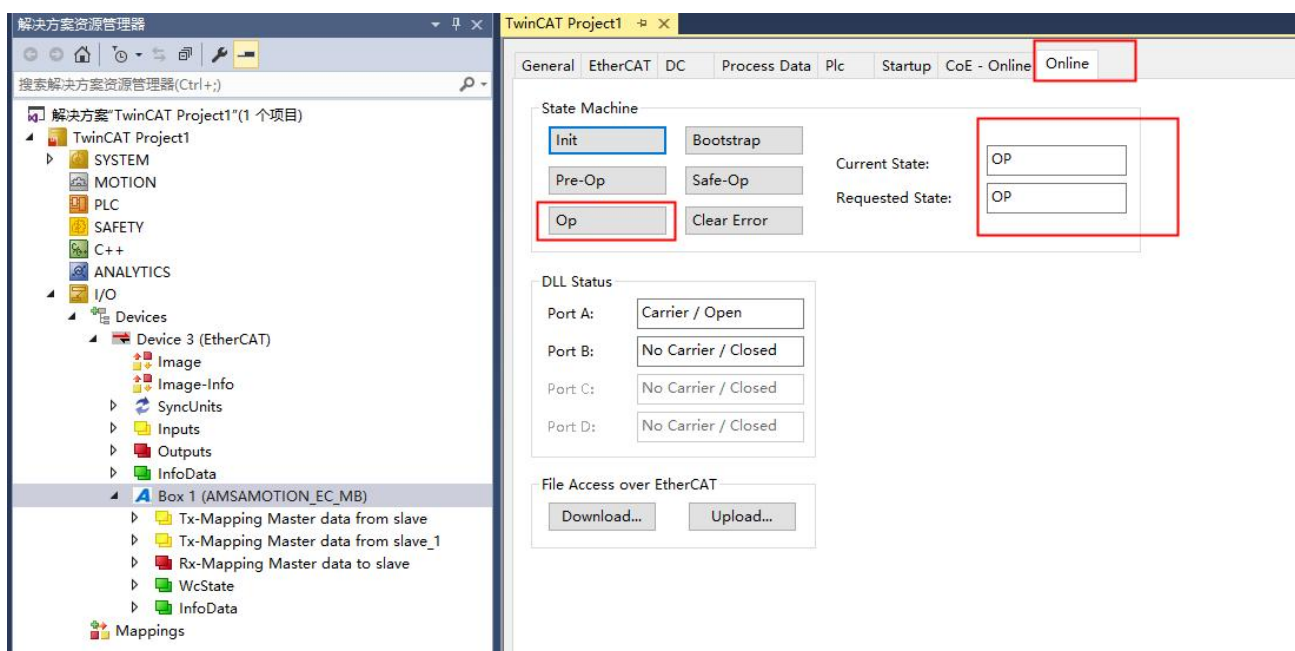
5)展开对象字 8001，同样将其设置为下图所示：

|          |                                    |               |
|----------|------------------------------------|---------------|
| 8001:0   | 1st slave configuration            | > 15 <        |
| 8001:01  | slave Addr                         | RW 0x01 (1)   |
| 8001:02  | coil is readable                   | RW TRUE       |
| 8001:03  | Keeps the register readable        | RW FALSE      |
| 8001:05  | Slave coil start address           | RW 0x0000 (0) |
| 8001:06  | The number of slave coil           | RW 0x10 (16)  |
| 8001:08  | slave Discrete input start address | RW 0x0000 (0) |
| 8001:09  | The number of slave Discrete input | RW 0x10 (16)  |
| 8001:... | Slave input register start address | RW 0x0000 (0) |
| 8001:... | The number of Slave input register | RW 0x00 (0)   |
| 8001:0E  | Slave hold register start address  | RW 0x0000 (0) |
| 8001:0F  | The number of Slave hold register  | RW 0x00 (0)   |

6)单击标题栏中 Process Data（图中“1”处），然后分别单击下两张图中的“2”处，检查图中“3”是否为图中所示，如果不是，先点击图中“4”所示位置，将其取消，然后点击“3”选中：



7) 点击标题栏，将页面切回 Online，将状态更改为 Op，此时已经在开始运行，RUN 灯以 1 秒周期闪烁，RS485 灯以 0.5 秒周期闪烁：



8)此时单击下图中的“1”或者“2”，可以看到 Data Valid 位开始更新，如果有数据，相应的位会置 0 或者置 1，其中 DI 或者回读的 DO 在当前窗口无法直接看到变化，按下图 2 所示进行操作（显示的是回读的 DO），可看到每一位的状态。

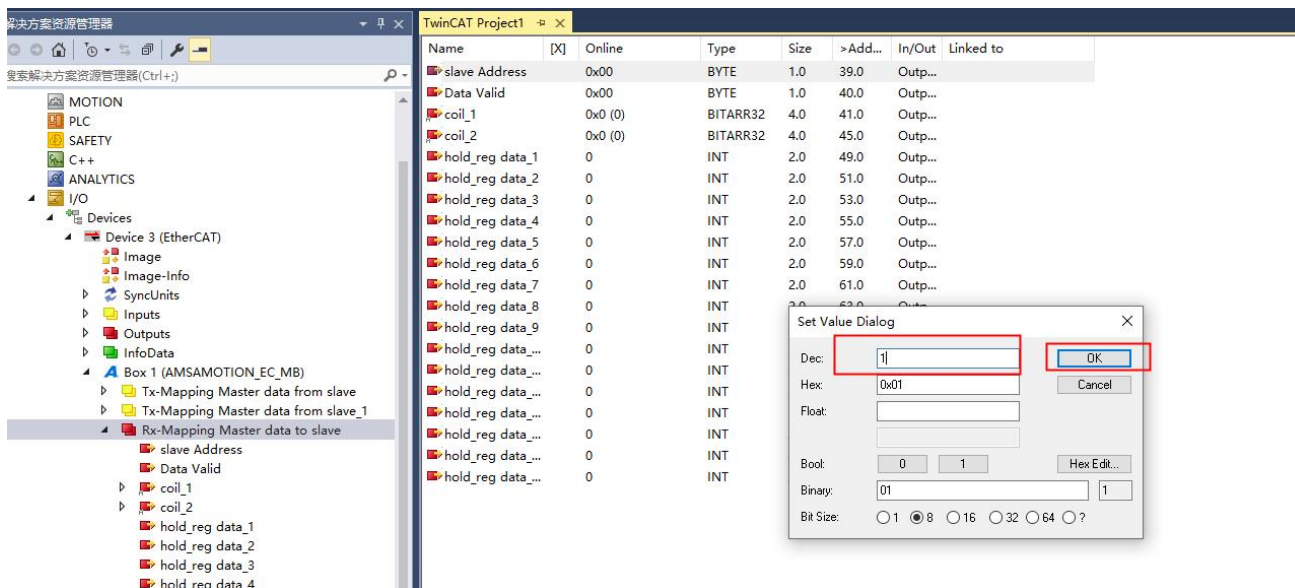
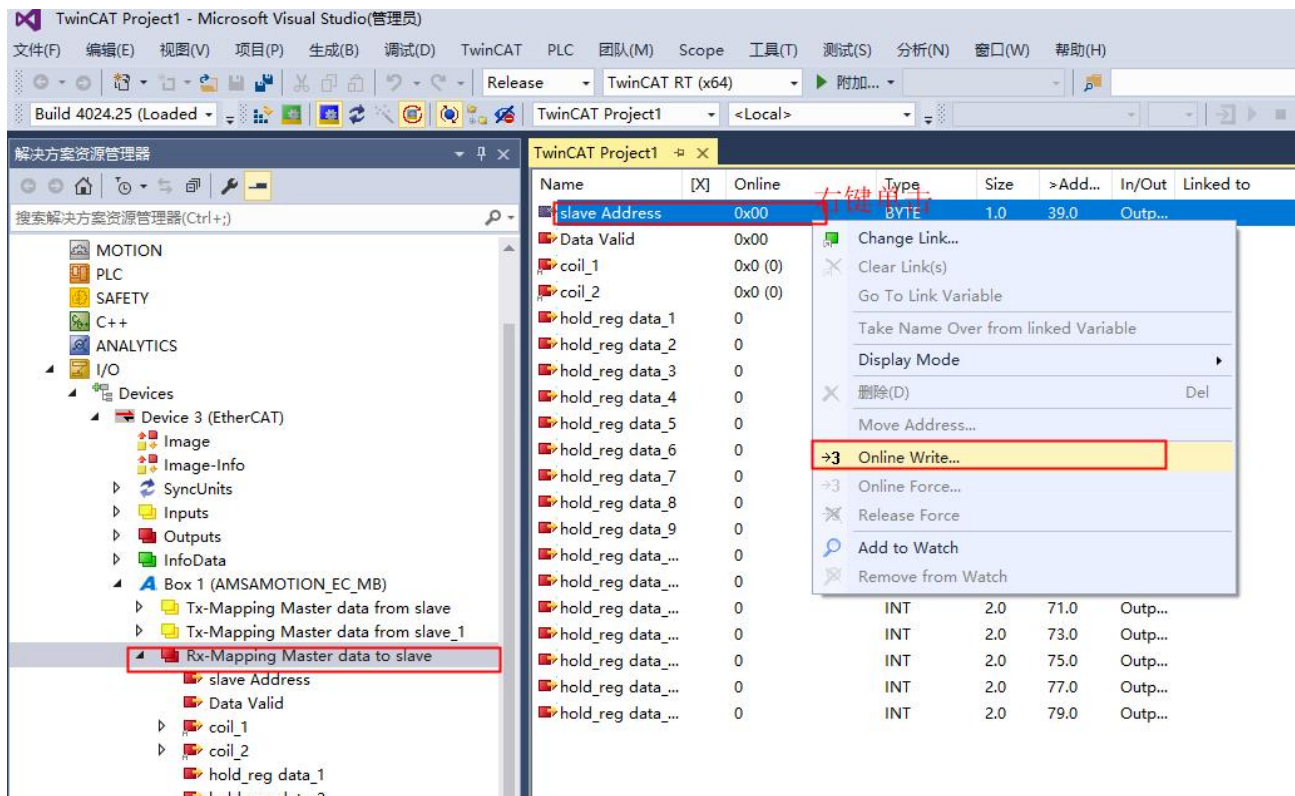
The top screenshot shows the 'Object Explorer' on the left with 'Tx-Mapping Master data from slave 1' highlighted (labeled 1) and 'Data Valid' highlighted (labeled 2). The right pane shows a list of variables including 'slave Address', 'Data Valid', and various input registers.

The bottom screenshot shows the 'Online List' window with a table of coil addresses and their values. A red box highlights the 'Online' column in the table.

| Name   | Online | Value | T |
|--------|--------|-------|---|
| coil_1 |        |       | B |
| [0]    | 1      | 1     | B |
| [1]    | 1      | 1     | B |
| [2]    | 1      | 1     | B |
| [3]    | 1      | 1     | B |
| [4]    | 1      | 1     | B |
| [5]    | 1      | 1     | B |
| [6]    | 1      | 1     | B |
| [7]    | 1      | 1     | B |
| [8]    | 1      | 1     | B |
| [9]    | 1      | 1     | B |
| [10]   | 1      | 1     | B |
| [11]   | 1      | 1     | B |
| [12]   | 1      | 1     | B |
| [13]   | 1      | 1     | B |
| [14]   | 1      | 1     | B |
| [15]   | 1      | 1     | B |
| [16]   | 0      | 0     | B |
| [17]   | 0      | 0     | B |
| [18]   | 0      | 0     | B |
| [19]   | 0      | 0     | B |
| [20]   | 0      | 0     | B |
| [21]   | 0      | 0     | B |

9)如果需要写线圈输出或者写保持寄存器，将分别将要写的从站地址，需要写的数字写入下图中的地方，然后更改 Data Valid 位，modbus 将会把需要更新的从站数据发出。





**Variable Declaration Table:**

| Name              | Online  | Type     | Size | >Add... | In/Out  | Linked to |
|-------------------|---------|----------|------|---------|---------|-----------|
| slave Address     | 0x01    | BYTE     | 1.0  | 39.0    | Outp... |           |
| Data Valid        | 0x00    | BYTE     | 1.0  | 40.0    | Outp... |           |
| coil_1            | 0x0 (0) | BITARR32 | 4.0  | 41.0    | Outp... |           |
| coil_2            | 0x0 (0) | BITARR32 | 4.0  | 45.0    | Outp... |           |
| hold_reg data_1   | 0       | INT      | 2.0  | 49.0    | Outp... |           |
| hold_reg data_2   | 0       | INT      | 2.0  | 51.0    | Outp... |           |
| hold_reg data_3   | 0       | INT      | 2.0  | 53.0    | Outp... |           |
| hold_reg data_4   | 0       | INT      | 2.0  | 55.0    | Outp... |           |
| hold_reg data_5   | 0       | INT      | 2.0  | 57.0    | Outp... |           |
| hold_reg data_6   | 0       | INT      | 2.0  | 59.0    | Outp... |           |
| hold_reg data_7   | 0       | INT      | 2.0  | 61.0    | Outp... |           |
| hold_reg data_8   | 0       | INT      | 2.0  | 63.0    | Outp... |           |
| hold_reg data_9   | 0       | INT      | 2.0  | 65.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 67.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 69.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 71.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 73.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 75.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 77.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 79.0    | Outp... |           |

**Set Value Dialog:**

Dec:

Hex:

Float:

Bool: ☐ 0 ☐ 1

Binary:

Bit Size: ☐ 1 ☐ 8 ☐ 16 ☒ 32 ☐ 64 ☐ ?

右键单击

同输入一样，此处写入数据后，显示是不会变化的，如果需要在此页面查看，展开图中1，所示，查看每一位的数据

**Variable Declaration Table:**

| Name              | Online  | Type     | Size | >Add... | In/Out  | Linked to |
|-------------------|---------|----------|------|---------|---------|-----------|
| slave Address     | 0x01    | BYTE     | 1.0  | 39.0    | Outp... |           |
| Data Valid        | 0x00    | BYTE     | 1.0  | 40.0    | Outp... |           |
| coil_1            | 0x0 (0) | BITARR32 | 4.0  | 41.0    | Outp... |           |
| coil_2            | 0x0 (0) | BITARR32 | 4.0  | 45.0    | Outp... |           |
| hold_reg data_1   | 0       | INT      | 2.0  | 49.0    | Outp... |           |
| hold_reg data_2   | 0       | INT      | 2.0  | 51.0    | Outp... |           |
| hold_reg data_3   | 0       | INT      | 2.0  | 53.0    | Outp... |           |
| hold_reg data_4   | 0       | INT      | 2.0  | 55.0    | Outp... |           |
| hold_reg data_5   | 0       | INT      | 2.0  | 57.0    | Outp... |           |
| hold_reg data_6   | 0       | INT      | 2.0  | 59.0    | Outp... |           |
| hold_reg data_7   | 0       | INT      | 2.0  | 61.0    | Outp... |           |
| hold_reg data_8   | 0       | INT      | 2.0  | 63.0    | Outp... |           |
| hold_reg data_9   | 0       | INT      | 2.0  | 65.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 67.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 69.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 71.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 73.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 75.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 77.0    | Outp... |           |
| hold_reg data_... | 0       | INT      | 2.0  | 79.0    | Outp... |           |

**Set Value Dialog:**

Dec:

Hex:

Float:

Bool: ☐ 0 ☐ 1

Binary:

Bit Size: ☐ 1 ☒ 8 ☐ 16 ☐ 32 ☐ 64 ☐ ?

写和当前值不同即可，0-255

10)按上面写入后，读回线圈状态，第0位和第2位是1，其他均为0，如下图：

**Online List Table:**

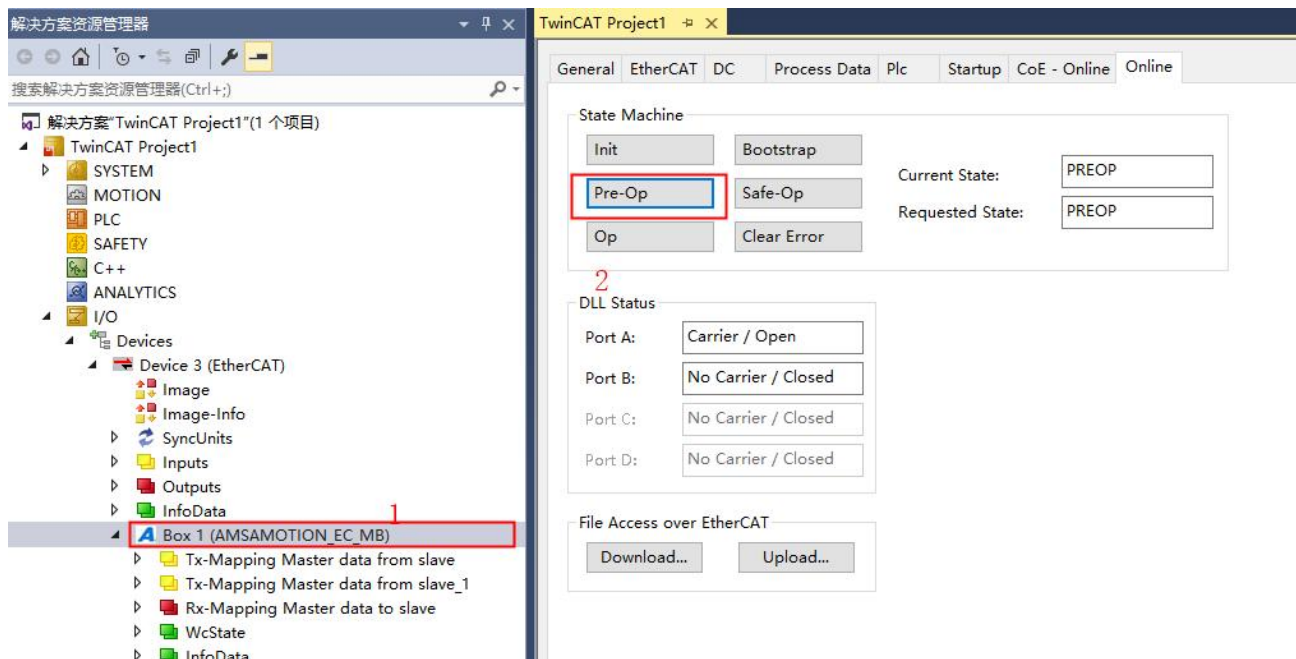
| Name   | Online | Value | Type     |
|--------|--------|-------|----------|
| coil_1 |        |       | BITARR32 |
| [0]    | 1      | 1     | BIT      |
| [1]    | 0      | 0     | BIT      |
| [2]    | 1      | 1     | BIT      |
| [3]    | 0      | 0     | BIT      |
| [4]    | 0      | 0     | BIT      |
| [5]    | 0      | 0     | BIT      |
| [6]    | 0      | 0     | BIT      |
| [7]    | 0      | 0     | BIT      |
| [8]    | 0      | 0     | BIT      |
| [9]    | 0      | 0     | BIT      |
| [10]   | 0      | 0     | BIT      |
| [11]   | 0      | 0     | BIT      |
| [12]   | 0      | 0     | BIT      |
| [13]   | 0      | 0     | BIT      |
| [14]   | 0      | 0     | BIT      |
| [15]   | 0      | 0     | BIT      |
| [16]   | 0      | 0     | BIT      |
| [17]   | 0      | 0     | BIT      |
| [18]   | 0      | 0     | BIT      |
| [19]   | 0      | 0     | BIT      |
| [20]   | 0      | 0     | BIT      |
| [21]   | 0      | 0     | BIT      |

### 5.3、透明传输功能

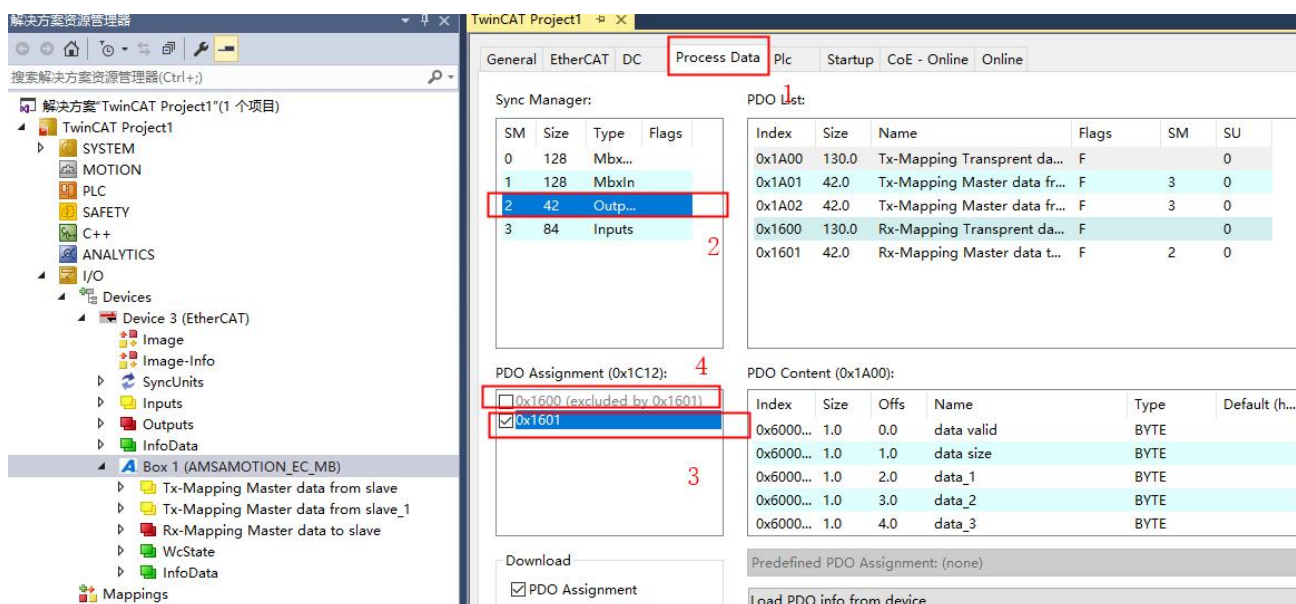
本章节在 5.1 章节基础上进行修改。

1)按下图单击“1”处，然后单击“2”，将运行模式切换回 Pre-Op。

**Note：**如果不是处于 Pre-Op 模式下，大部分对象字是不可修改的，否则会报错。



2)按下图将同步管理器输入输出更换，单击“1”处，将界面切换回 Process Data，然后单击“2”处，取消“3”处的勾选，后将“4”勾选上。





**Sync Manager:**

| SM | Size | Type    | Flags |
|----|------|---------|-------|
| 0  | 128  | Mbx...  |       |
| 1  | 128  | MbxIn   |       |
| 2  | 130  | Outp... |       |
| 3  | 84   | Inputs  |       |

**PDO List:**

| Index  | Size  | Name                         | Flag |
|--------|-------|------------------------------|------|
| 0x1A00 | 130.0 | Tx-Mapping Transp...         | F    |
| 0x1A01 | 42.0  | Tx-Mapping Master data fr... | F    |
| 0x1A02 | 42.0  | Tx-Mapping Master data fr... | F    |
| 0x1600 | 130.0 | Rx-Mapping Transp...         | F    |
| 0x1601 | 42.0  | Rx-Mapping Master data t...  | F    |

**PDO Assignment (0x1C13):**

|                                     |                             |
|-------------------------------------|-----------------------------|
| <input type="checkbox"/>            | 0x1A00 (excluded by 0x1A02) |
| <input checked="" type="checkbox"/> | 0x1A01                      |
| <input checked="" type="checkbox"/> | 0x1A02                      |

**PDO Content (0x1A00):**

| Index     | Size | Offs | Name       |
|-----------|------|------|------------|
| 0x6000... | 1.0  | 0.0  | data valid |
| 0x6000... | 1.0  | 1.0  | data size  |
| 0x6000... | 1.0  | 2.0  | data_1     |
| 0x6000... | 1.0  | 3.0  | data_2     |
| 0x6000... | 1.0  | 4.0  | data_3     |

**Download:**

☒ PDO Assignment

☐ PDO Configuration

Predefined PDO Assignment: (none)

Load PDO info from device

**Sync Manager:**

| SM | Size | Type    | Flags |
|----|------|---------|-------|
| 0  | 128  | Mbx...  |       |
| 1  | 128  | MbxIn   |       |
| 2  | 130  | Outp... |       |
| 3  | 84   | Inputs  |       |

**PDO List:**

| Index  | Size  | Name                         | Flags | SM | SU |
|--------|-------|------------------------------|-------|----|----|
| 0x1A00 | 130.0 | Tx-Mapping Transp...         | F     |    | 0  |
| 0x1A01 | 42.0  | Tx-Mapping Master data fr... | F     | 3  | 0  |
| 0x1A02 | 42.0  | Tx-Mapping Master data fr... | F     | 3  | 0  |
| 0x1600 | 130.0 | Rx-Mapping Transp...         | F     | 2  | 0  |
| 0x1601 | 42.0  | Rx-Mapping Master data t...  | F     |    | 0  |

**PDO Assignment (0x1C12):**

|                                     |                             |
|-------------------------------------|-----------------------------|
| <input checked="" type="checkbox"/> | 0x1600                      |
| <input type="checkbox"/>            | 0x1601 (excluded by 0x1600) |

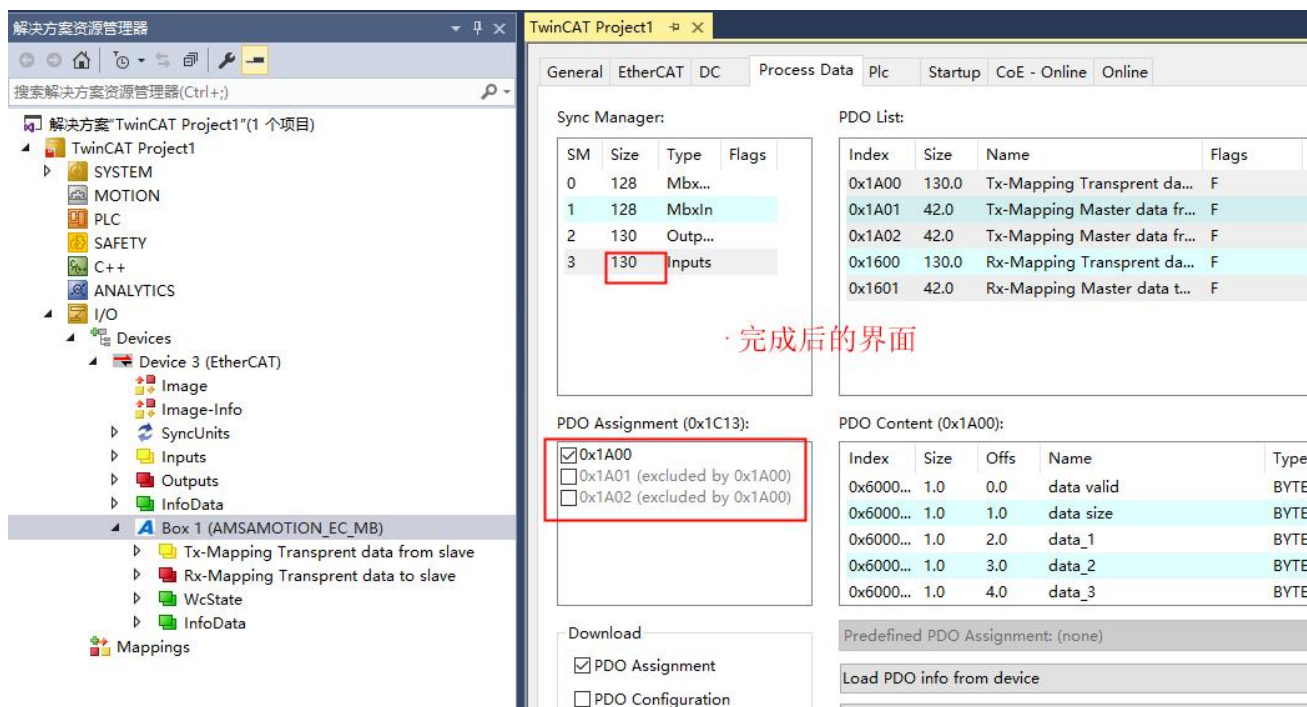
**PDO Content (0x1A00):**

| Index     | Size | Offs | Name       | Type | Default (h) |
|-----------|------|------|------------|------|-------------|
| 0x6000... | 1.0  | 0.0  | data valid | BYTE |             |
| 0x6000... | 1.0  | 1.0  | data size  | BYTE |             |
| 0x6000... | 1.0  | 2.0  | data_1     | BYTE |             |
| 0x6000... | 1.0  | 3.0  | data_2     | BYTE |             |
| 0x6000... | 1.0  | 4.0  | data_3     | BYTE |             |

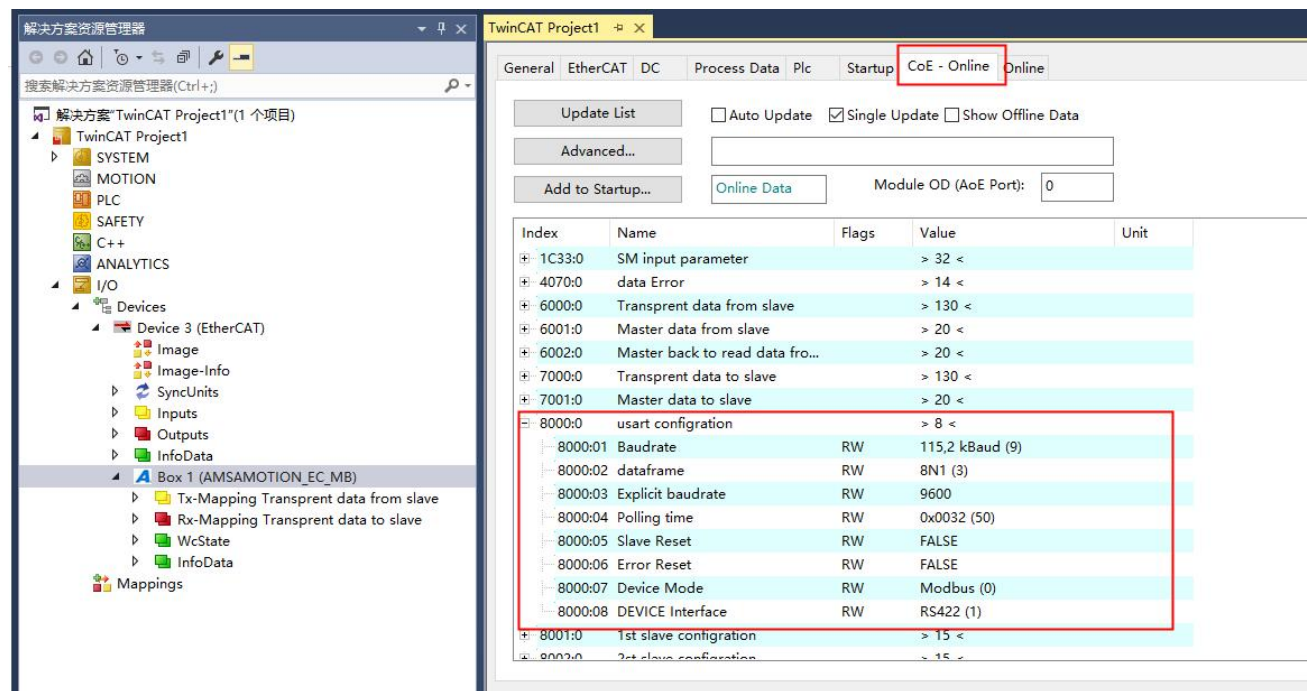
**Download:**

Predefined PDO Assignment: (none)

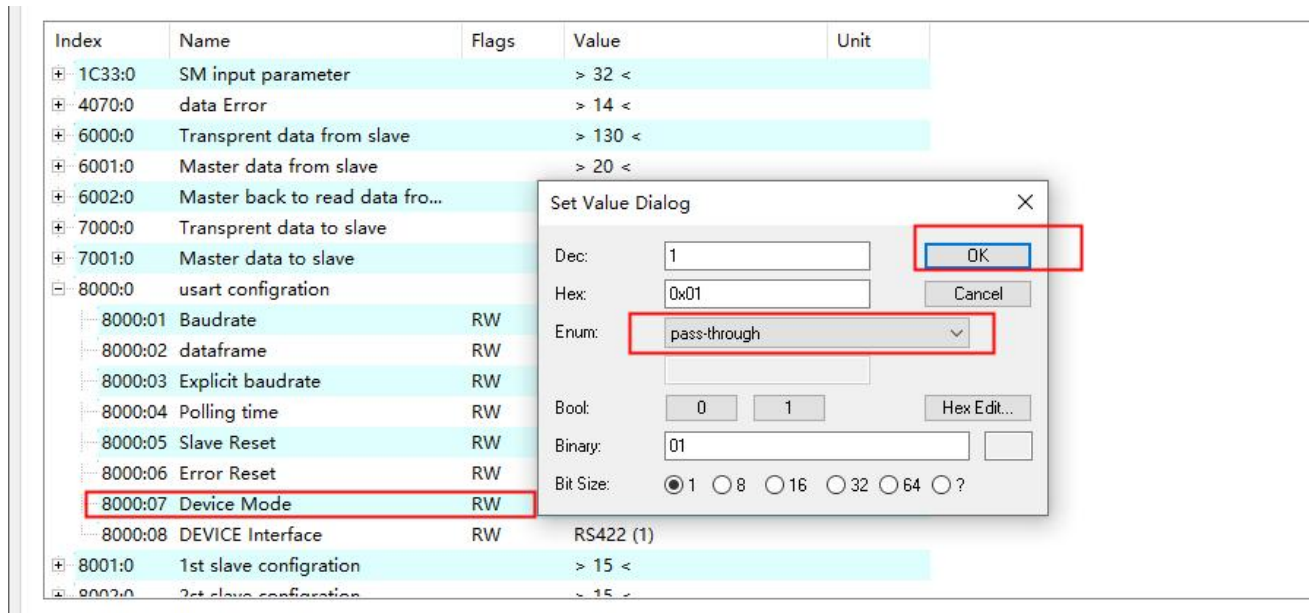
完成操作后的界面



3)按下图点击标题栏，将界面切换到 CoE-Online,找到对象字典索引 8000，展开：

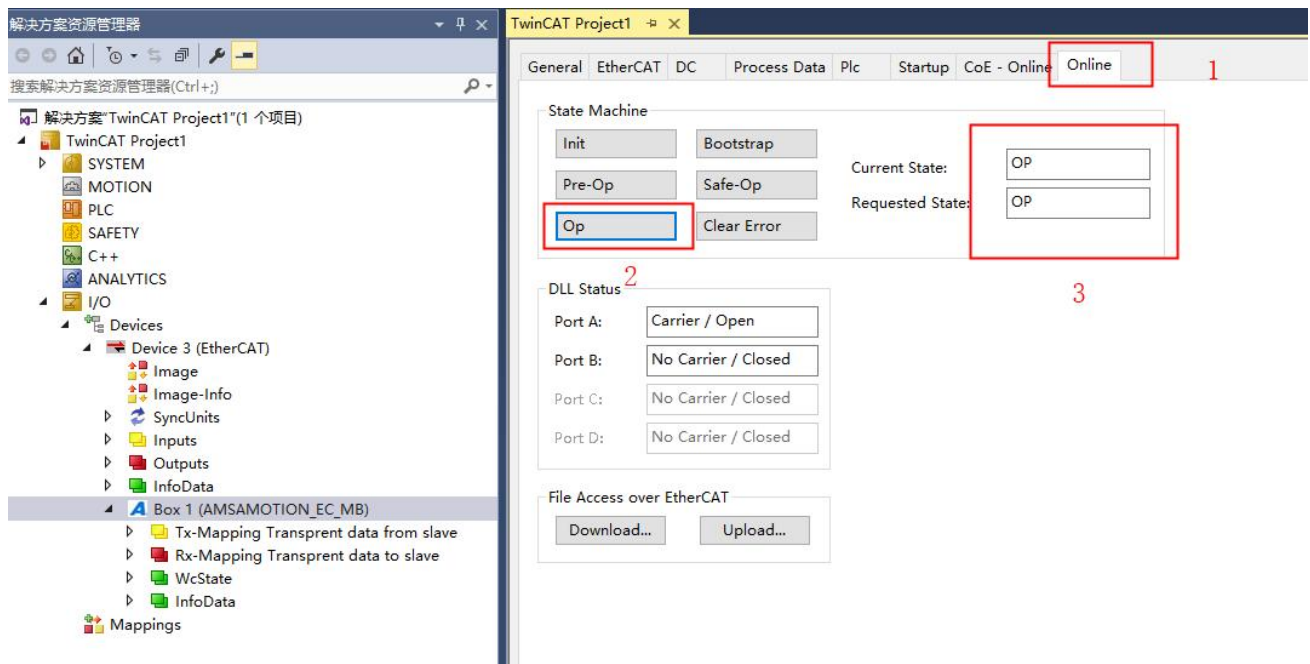


4)将索引 8000 下的子索引 7 修改为 pass-through 模式，如下：



5)波特率、数据格式、接口等按实际需求进行设置，完成后将运行状态切换到 Op，此时 RUN 灯以 1 秒周期闪烁，RS485 或者 RS422 灯以 0.2 秒周期闪烁：

**Note:** 使用透传功能时，索引 8001-800A 从站设置未使用，可以是任意合理的值。





6)将界面切换到输输出，如果 Online 行无数据，按下图操作：

如此行无数据，点击图中“2”处刷新。

| Name       | Online | Type | Size | >Add... | In/Out | Linked to |
|------------|--------|------|------|---------|--------|-----------|
| data valid | 0x01   | BYTE | 1.0  | 39.0    | Input  |           |
| data size  | 0x07   | BYTE | 1.0  | 40.0    | Input  |           |
| data_1     | 0x01   | BYTE | 1.0  | 41.0    | Input  |           |
| data_2     | 0x02   | BYTE | 1.0  | 42.0    | Input  |           |
| data_3     | 0x02   | BYTE | 1.0  | 43.0    | Input  |           |
| data_4     | 0x00   | BYTE | 1.0  | 44.0    | Input  |           |
| data_5     | 0x00   | BYTE | 1.0  | 45.0    | Input  |           |
| data_6     | 0xb9   | BYTE | 1.0  | 46.0    | Input  |           |
| data_7     | 0xb8   | BYTE | 1.0  | 47.0    | Input  |           |
| data_8     | 0x00   | BYTE | 1.0  | 48.0    | Input  |           |
| data_9     | 0x00   | BYTE | 1.0  | 49.0    | Input  |           |
| data_10    | 0x00   | BYTE | 1.0  | 50.0    | Input  |           |
| data_11    | 0x00   | BYTE | 1.0  | 51.0    | Input  |           |
| data_12    | 0x00   | BYTE | 1.0  | 52.0    | Input  |           |
| data_13    | 0x00   | BYTE | 1.0  | 53.0    | Input  |           |
| data_14    | 0x00   | BYTE | 1.0  | 54.0    | Input  |           |
| data_15    | 0x00   | BYTE | 1.0  | 55.0    | Input  |           |
| data_16    | 0x00   | BYTE | 1.0  | 56.0    | Input  |           |
| data_17    | 0x00   | BYTE | 1.0  | 57.0    | Input  |           |
| data_18    | 0x00   | BYTE | 1.0  | 58.0    | Input  |           |
| data_19    | 0x00   | BYTE | 1.0  | 59.0    | Input  |           |
| data_20    | 0x00   | BYTE | 1.0  | 60.0    | Input  |           |
| data_21    | 0x00   | BYTE | 1.0  | 61.0    | Input  |           |
| data_22    | 0x00   | BYTE | 1.0  | 62.0    | Input  |           |
| data_23    | 0x00   | BYTE | 1.0  | 63.0    | Input  |           |
| data_24    | 0x00   | BYTE | 1.0  | 64.0    | Input  |           |
| data_25    | 0x00   | BYTE | 1.0  | 65.0    | Input  |           |

7)打开一个串口助手，连接好硬件，发送任意字符，完成后可以看到模块显示和串口发送的数据一致：

每收到一包数据加1，到255后回0，判断是否收到数据  
收到数据总数

OD 0A是换行符，实际发送的数据是字符串，方便比较转换成了HEX显示，字符串发送时末尾带有换行符。

41 4D 53 41 4D 4F 54 49 4F 4E |

8)发送数据时如下图，切换到输出界面，将每一位数据写入 data\_x 中，x=1~128，将要发送的数据长度写入 data\_size 中,写完后如下图 2，完成后改变 data valid 位的值，将启动发送。

Note：写入数据时注意进制转换。

将数据每一位依次写入到data\_x中，x=1~128

2, 右键单击

3

| Name       | [X] | Online | Type | Size | >Add... | In/Out  | Linked to |
|------------|-----|--------|------|------|---------|---------|-----------|
| data valid |     | 0x00   | BYTE | 1.0  | 39.0    | Outp... |           |
| data_size  |     | 0x00   | BYTE | 1.0  | 40.0    | Outp... |           |
| data_1     |     | 0x00   | BYTE | 1.0  | 41.0    | Outp... |           |
| data_2     |     | 0x00   | BYTE | 1.0  | 42.0    | Outp... |           |
| data_3     |     | 0x00   | BYTE | 1.0  | 43.0    | Outp... |           |
| data_4     |     | 0x00   | BYTE | 1.0  | 44.0    | Outp... |           |
| data_5     |     | 0x00   | BYTE | 1.0  | 45.0    | Outp... |           |
| data_6     |     | 0x00   | BYTE | 1.0  | 46.0    | Outp... |           |
| data_7     |     | 0x00   | BYTE | 1.0  | 47.0    | Outp... |           |
| data_8     |     | 0x00   | BYTE | 1.0  | 48.0    | Outp... |           |
| data_9     |     | 0x00   | BYTE | 1.0  | 49.0    | Outp... |           |
| data_10    |     | 0x00   | BYTE | 1.0  | 50.0    | Outp... |           |
| data_11    |     | 0x00   | BYTE | 1.0  | 51.0    | Outp... |           |
| data_12    |     | 0x00   | BYTE | 1.0  | 52.0    | Outp... |           |
| data_13    |     | 0x00   | BYTE | 1.0  | 53.0    | Outp... |           |
| data_14    |     | 0x00   | BYTE | 1.0  | 54.0    | Outp... |           |
| data_15    |     | 0x00   | BYTE | 1.0  | 55.0    | Outp... |           |
| data_16    |     | 0x00   | BYTE | 1.0  | 56.0    | Outp... |           |
| data_17    |     | 0x00   | BYTE | 1.0  | 57.0    | Outp... |           |
| data_18    |     | 0x00   | BYTE | 1.0  | 58.0    | Outp... |           |
| data_19    |     | 0x00   | BYTE | 1.0  | 59.0    | Outp... |           |
| data_20    |     | 0x00   | BYTE | 1.0  | 60.0    | Outp... |           |
| data_21    |     | 0x00   | BYTE | 1.0  | 61.0    | Outp... |           |
| data_22    |     | 0x00   | BYTE | 1.0  | 62.0    | Outp... |           |
| data_23    |     | 0x00   | BYTE | 1.0  | 63.0    | Outp... |           |
| data_24    |     | 0x00   | BYTE | 1.0  | 64.0    | Outp... |           |
| data_25    |     | 0x00   | BYTE | 1.0  | 65.0    | Outp... |           |

完成后改变data valid值，将启动发送

| Name       | [X] | Online | Type | Size | >Add... | In/Out  | Linked to |
|------------|-----|--------|------|------|---------|---------|-----------|
| data valid |     | 0x00   | BYTE | 1.0  | 39.0    | Outp... |           |
| data_size  |     | 0x00   | BYTE | 1.0  | 40.0    | Outp... |           |
| data_1     |     | 0x41   | BYTE | 1.0  | 41.0    | Outp... |           |
| data_2     |     | 0x4d   | BYTE | 1.0  | 42.0    | Outp... |           |
| data_3     |     | 0x53   | BYTE | 1.0  | 43.0    | Outp... |           |
| data_4     |     | 0x41   | BYTE | 1.0  | 44.0    | Outp... |           |
| data_5     |     | 0x4d   | BYTE | 1.0  | 45.0    | Outp... |           |
| data_6     |     | 0x4f   | BYTE | 1.0  | 46.0    | Outp... |           |
| data_7     |     | 0x54   | BYTE | 1.0  | 47.0    | Outp... |           |
| data_8     |     | 0x49   | BYTE | 1.0  | 48.0    | Outp... |           |
| data_9     |     | 0x4f   | BYTE | 1.0  | 49.0    | Outp... |           |
| data_10    |     | 0x4e   | BYTE | 1.0  | 50.0    | Outp... |           |
| data_11    |     | 0x0d   | BYTE | 1.0  | 51.0    | Outp... |           |
| data_12    |     | 0x0a   | BYTE | 1.0  | 52.0    | Outp... |           |
| data_13    |     | 0x00   | BYTE | 1.0  | 53.0    | Outp... |           |
| data_14    |     | 0x00   | BYTE | 1.0  | 54.0    | Outp... |           |
| data_15    |     | 0x00   | BYTE | 1.0  | 55.0    | Outp... |           |
| data_16    |     | 0x00   | BYTE | 1.0  | 56.0    | Outp... |           |
| data_17    |     | 0x00   | BYTE | 1.0  | 57.0    | Outp... |           |
| data_18    |     | 0x00   | BYTE | 1.0  | 58.0    | Outp... |           |
| data_19    |     | 0x00   | BYTE | 1.0  | 59.0    | Outp... |           |
| data_20    |     | 0x00   | BYTE | 1.0  | 60.0    | Outp... |           |
| data_21    |     | 0x00   | BYTE | 1.0  | 61.0    | Outp... |           |
| data_22    |     | 0x00   | BYTE | 1.0  | 62.0    | Outp... |           |
| data_23    |     | 0x00   | BYTE | 1.0  | 63.0    | Outp... |           |
| data_24    |     | 0x00   | BYTE | 1.0  | 64.0    | Outp... |           |
| data_25    |     | 0x00   | BYTE | 1.0  | 65.0    | Outp... |           |



9) 发送完成后如下图:

The screenshot shows the TwinCAT Project1 interface. On the left is the '解决方案资源管理器' (Solution Explorer) showing the project structure. The main area displays a table of data points. A blue box highlights the 'data\_1' through 'data\_12' entries in the 'Name' and 'Online' columns. The 'Type' column shows 'ATX XCOM V2.6'. The 'Size' column shows values from 0x01 to 0x0a. The 'Data' column shows a sequence of hexadecimal values: 41 4D 53 41 4D 4F 54 49 4F 4E 0D 0A. At the bottom, there is a status bar with buttons for '单条发送' (Send Single), '多条发送' (Send Multiple), '协议传输' (Protocol Transfer), and '帮助' (Help). The status bar also displays the same sequence of hexadecimal values: 41 4D 53 41 4D 4F 54 49 4F 4E.

| Name       | [X] | Online | Type          | Size | >Add... | In/Out | Linked to                           |
|------------|-----|--------|---------------|------|---------|--------|-------------------------------------|
| data valid |     | 0x01   | ATX XCOM V2.6 |      |         |        |                                     |
| data size  |     | 0x0c   |               |      |         |        |                                     |
| data_1     |     | 0x41   |               |      |         |        | 41 4D 53 41 4D 4F 54 49 4F 4E 0D 0A |
| data_2     |     | 0x4d   |               |      |         |        |                                     |
| data_3     |     | 0x53   |               |      |         |        |                                     |
| data_4     |     | 0x41   |               |      |         |        |                                     |
| data_5     |     | 0x4d   |               |      |         |        |                                     |
| data_6     |     | 0x4f   |               |      |         |        |                                     |
| data_7     |     | 0x54   |               |      |         |        |                                     |
| data_8     |     | 0x49   |               |      |         |        |                                     |
| data_9     |     | 0x4f   |               |      |         |        |                                     |
| data_10    |     | 0x4e   |               |      |         |        |                                     |
| data_11    |     | 0x0d   |               |      |         |        |                                     |
| data_12    |     | 0x0a   |               |      |         |        |                                     |
| data_13    |     | 0x00   |               |      |         |        |                                     |
| data_14    |     | 0x00   |               |      |         |        |                                     |
| data_15    |     | 0x00   |               |      |         |        |                                     |
| data_16    |     | 0x00   |               |      |         |        |                                     |
| data_17    |     | 0x00   |               |      |         |        |                                     |
| data_18    |     | 0x00   |               |      |         |        |                                     |

单条发送 多条发送 协议传输 帮助

41 4D 53 41 4D 4F 54 49 4F 4E

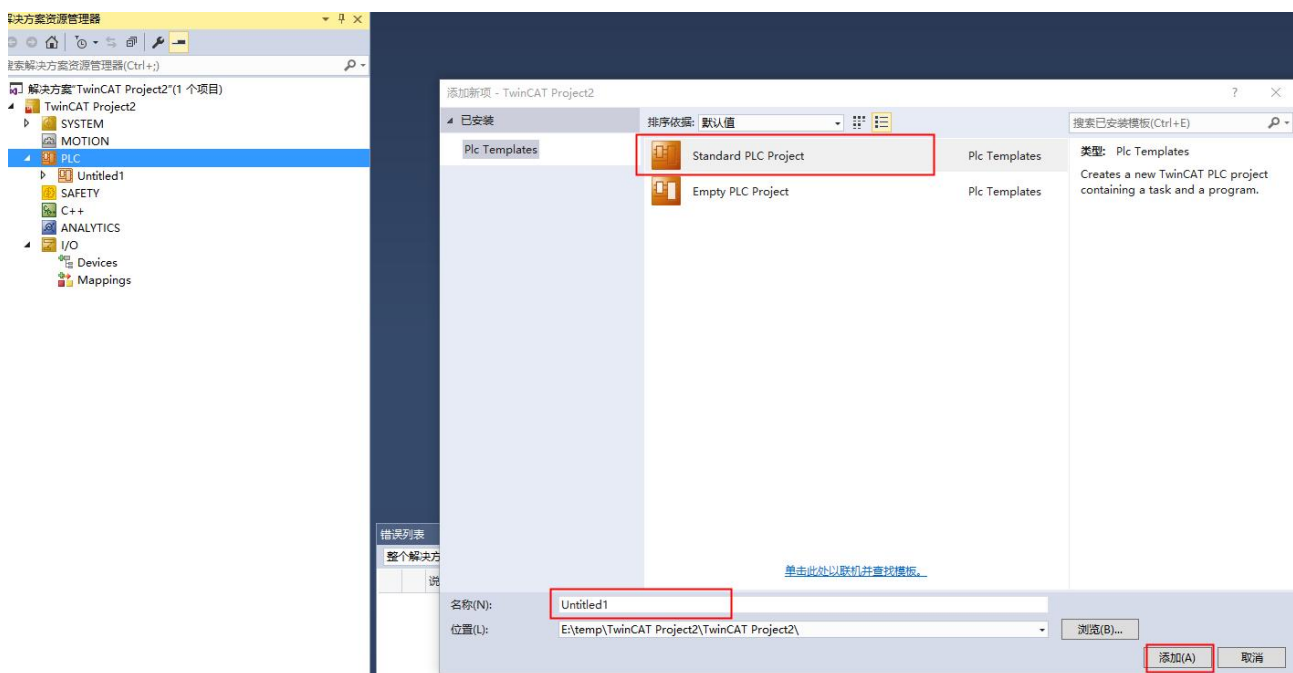
## 5.4、配合 PLC 使用

本章节利用 10 个 16 位输入输出模块配合本模块，利用简单流水灯工程示例，演示添加 PLC 工程及映射变量，并保存每个从站中回读数据。

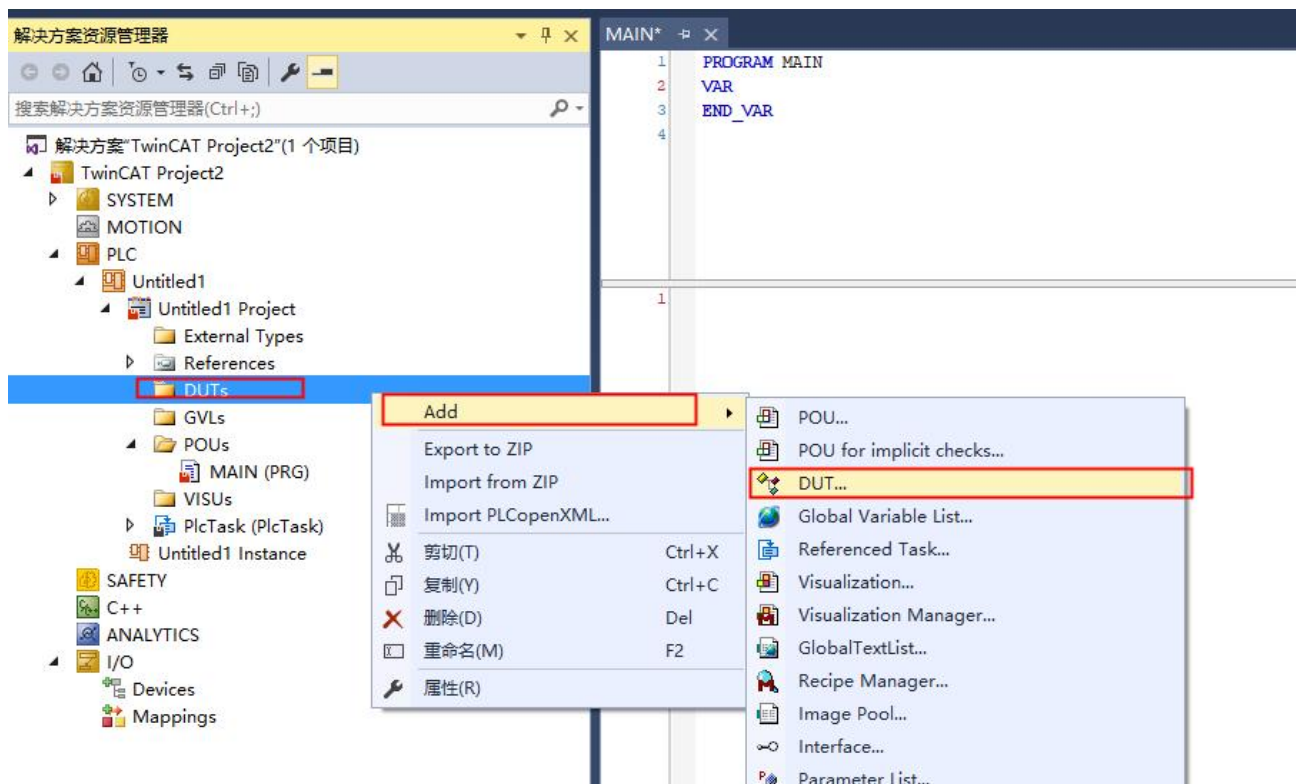
1) 如下图，找到 PLC 选项，右键单击，然后选择添加新项：



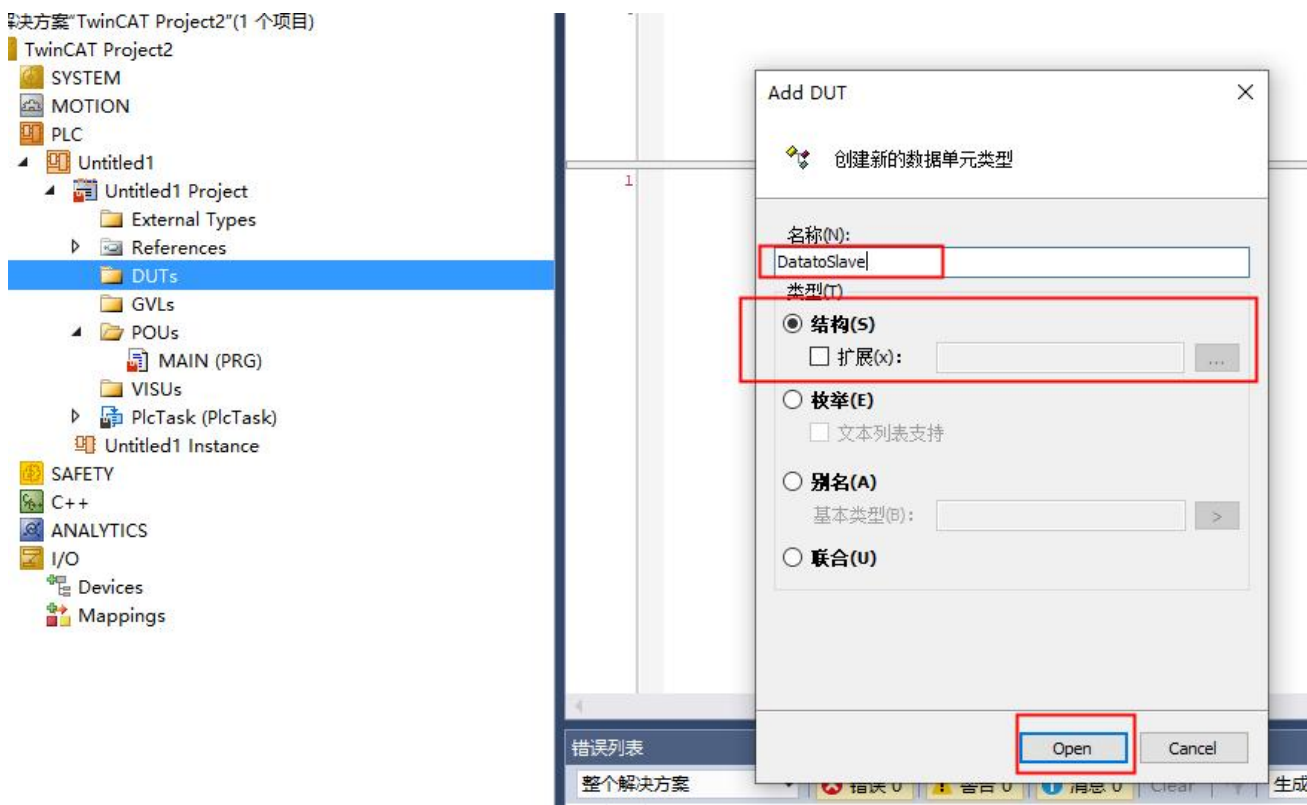
2) 选择标准 PLC 项目，输入名称，点击添加：



3) 找到 DUTs 文件夹，右键单击，天机 DUT 文件，如下图。



4) 输入数据结构体名称“DatatoSlave”,点击打开:



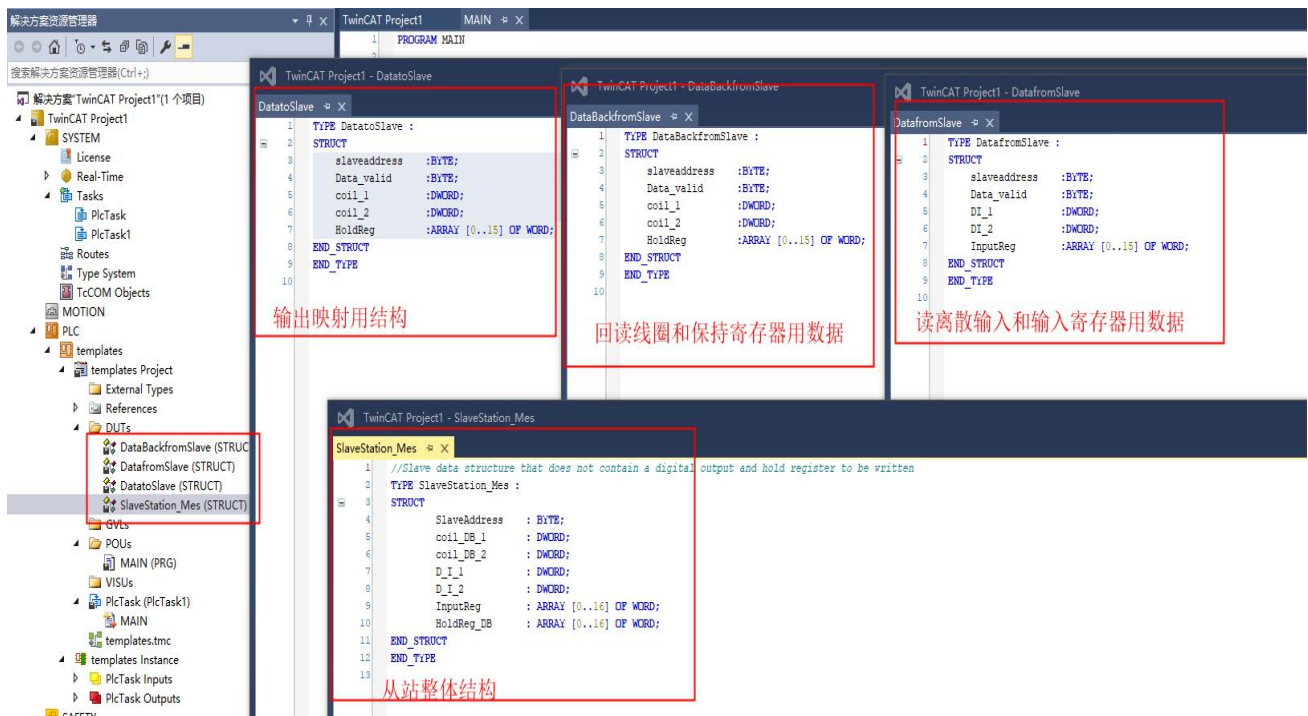
5) 输入以下代码：

```

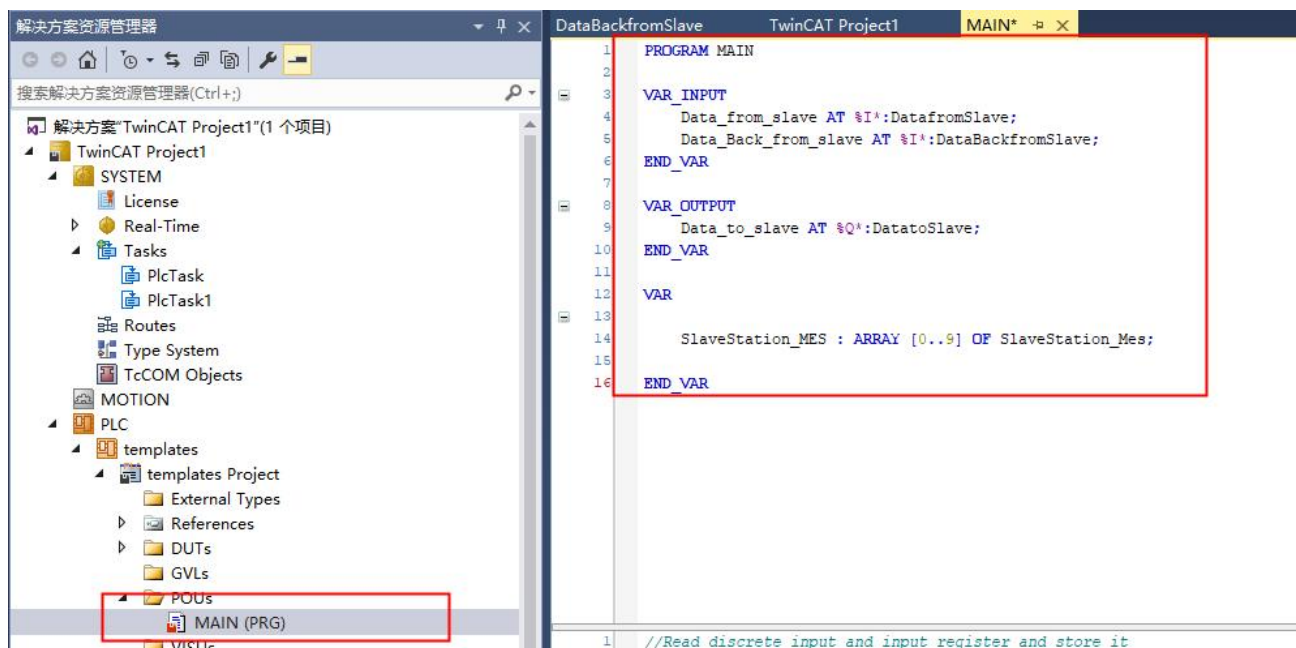
1  TYPE DatatoSlave :
2  STRUCT
3      slaveaddress      :BYTE;
4      Data_valid        :BYTE;
5      coil_1            :DWORD;
6      coil_2            :DWORD;
7      HoldReg           :ARRAY [0..15] OF WORD;
8  END_STRUCT
9  END_TYPE
10

```

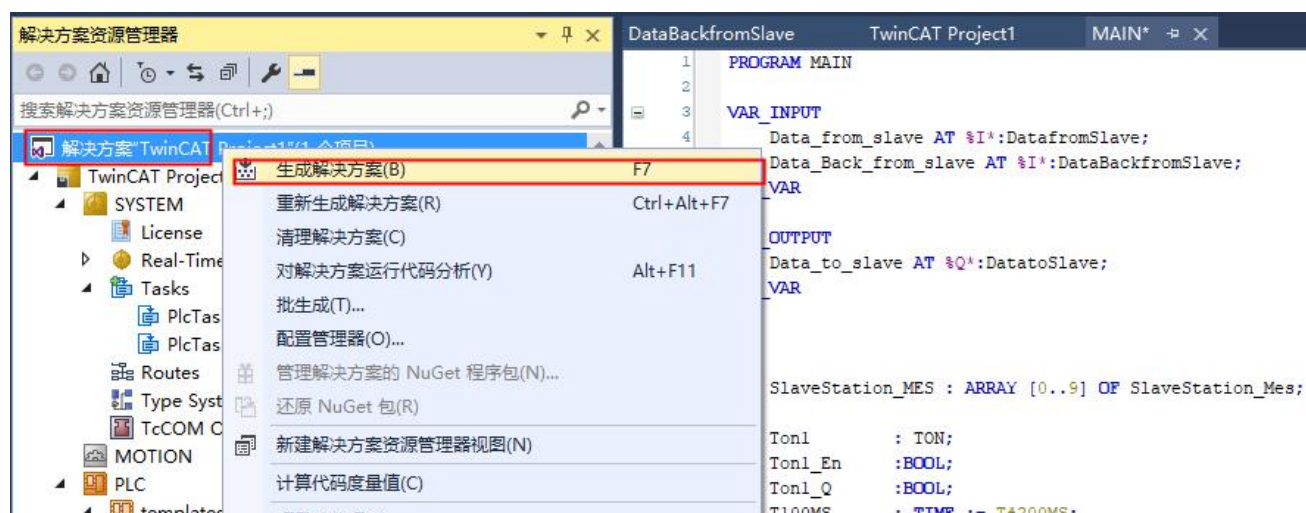
6) 按照步骤 3、4、5 新建其他如下图中结构体：



7) 找到 MAIN (PRG) 文件，双击打开，在声明栏输入以下变量声明，其中“Data\_from\_slave”用于读输入映射，“Data\_Back\_from\_slave”用于回读输出的映射，“Data\_to\_slave”用于输出映射，“SlaveStation\_MES”保存了 10 个从站的输入和回读数据：

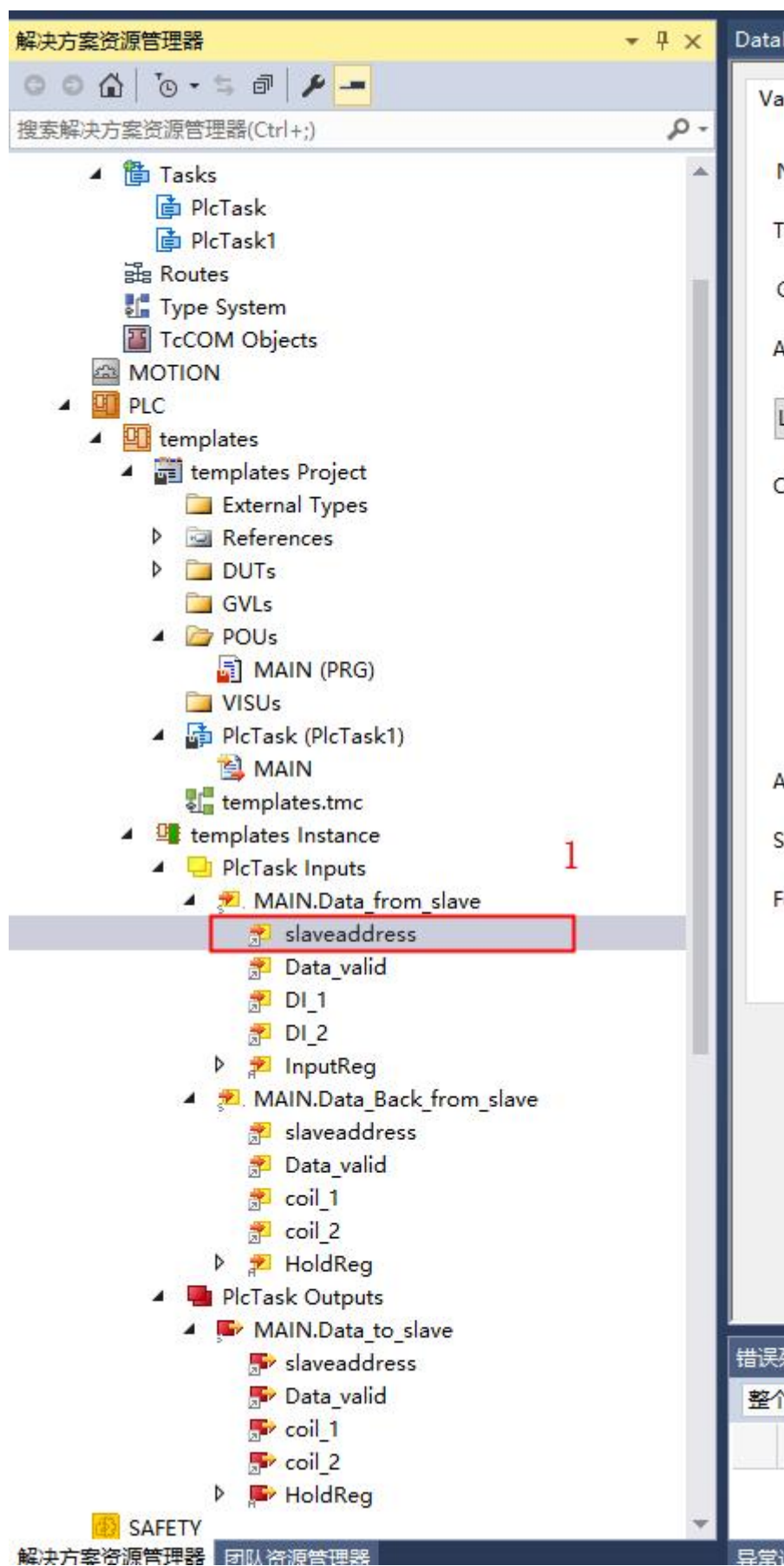


8) 按下 F7 或者按下图右键点击“解决方案”后单击“生成解决方案”

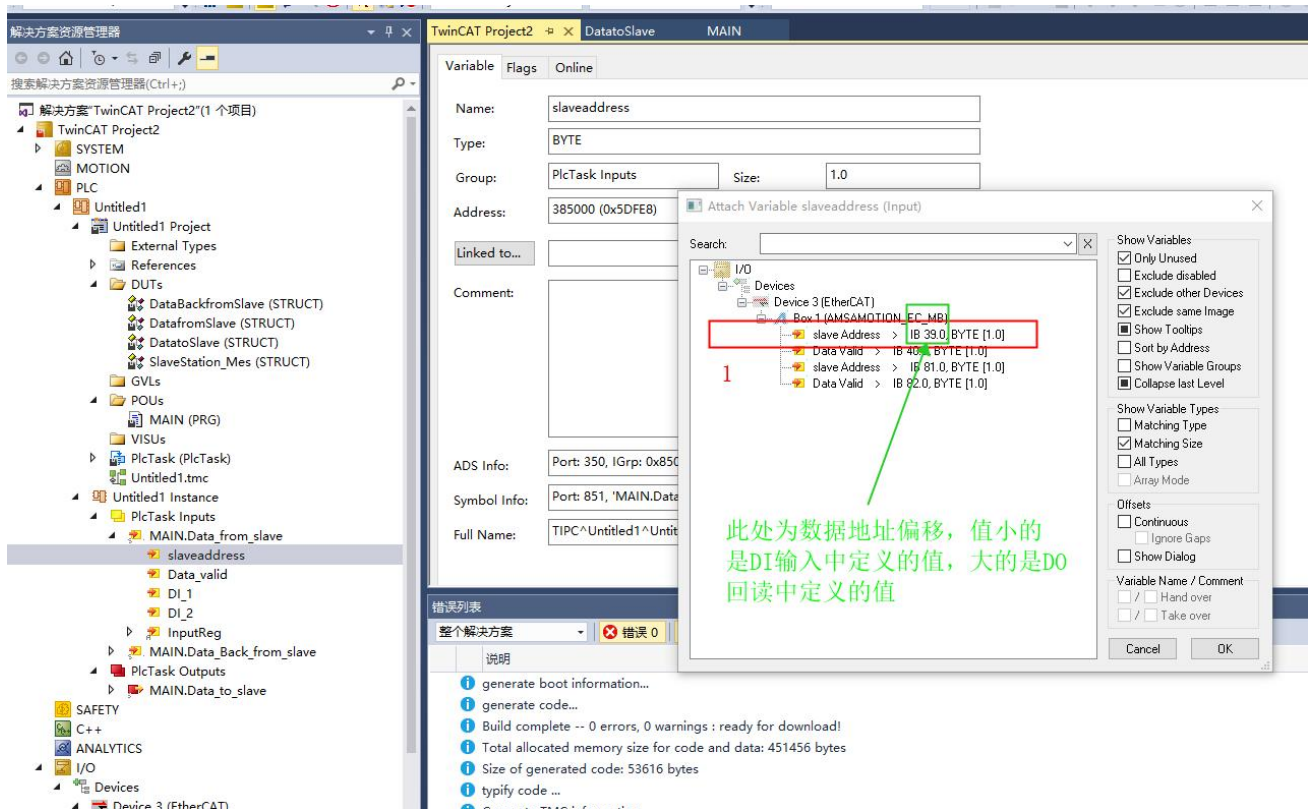


9) 编译后会生成部分数据，在 PLC 栏目下找到“PLCTask Inputs “和” PLCTask Outputs”并展开，双击图中“1”所示位置。





10) 双击下图所示位置，第 9 步中所示其他变量也是一样设置。



11) 如果需要保存回读的每个从站数据，可参考以下代码：

```

DataBackfromSlave  MAIN  -P X
1 PROGRAM MAIN
2
3 VAR_INPUT
4   Data_from_slave AT %I^:DatafromSlave;
5   Data_Back_from_slave AT %I^:DataBackfromSlave;
6 END_VAR
7
8 VAR_OUTPUT
9   Data_to_slave AT %Q^:DatatoSlave;
10 END_VAR
11
12 VAR
13   SlaveStation_MES : ARRAY [0..9] OF SlaveStation_Mes;
14
15   Ton1      : TON;
16   Ton1_En   : BOOL;
17   Ton1_Q    : BOOL;
18   T100MS    : TIME := T#200MS;
19
20   Data_from_slave_Data_valid : BYTE := 0;
21   Data_Back_from_slave_Data_valid : BYTE := 0;
22
23   i : INT := 0;
24 END_VAR
25
26 //Read discrete input and input register and store it
27 IF (Data_from_slave.Data_valid <> Data_from_slave_Data_valid) THEN
28   SlaveStation_MES[Data_from_slave.slaveaddress-2].SlaveAddress := Data_from_slave.slaveaddress;
29   SlaveStation_MES[Data_from_slave.slaveaddress-2].DI_1 := Data_from_slave.DI_1;
30   SlaveStation_MES[Data_from_slave.slaveaddress-2].DI_2 := Data_from_slave.DI_2;
31
32   FOR i:=0 TO 15 BY 1 DO
33     SlaveStation_MES[Data_from_slave.slaveaddress-2].InputReg[i] := Data_from_slave.InputReg[i];
34   END_FOR
35   Data_from_slave_Data_valid := Data_from_slave.Data_valid;
36 END_IF
37
38 //Read back digital output and hold register and store
39 IF(Data_Back_from_slave.Data_valid <> Data_Back_from_slave_Data_valid) THEN
40   SlaveStation_MES[Data_Back_from_slave.slaveaddress-2].SlaveAddress := Data_Back_from_slave.slaveaddress;
41   SlaveStation_MES[Data_Back_from_slave.slaveaddress-2].coil_DB_1 := Data_Back_from_slave.coil_1;
42   SlaveStation_MES[Data_Back_from_slave.slaveaddress-2].coil_DB_2 := Data_Back_from_slave.coil_2;
43
44   FOR i:=0 TO 15 BY 1 DO
45     SlaveStation_MES[Data_Back_from_slave.slaveaddress-2].HoldReg_DB[i] := Data_Back_from_slave.HoldReg[i];
46   END_FOR
47   Data_Back_from_slave_Data_valid := Data_Back_from_slave.Data_valid;
48 END_IF
49
50 //Running water lamp application

```

12) 以下是实现 10 个从站 DO 输出流水灯代码，供参考：

```

24
25 //Running water lamp application
26 Ton1(IN:= Ton1_En, PT:= T100MS, Q=> Ton1_Q, ET=> );
27
28 IF Ton1_Q THEN
29   Data_to_slave.coil_1 := Data_to_slave.coil_1 * 2 + 1;
30   IF (Data_to_slave.coil_1 > 65535) THEN
31     Data_to_slave.slaveaddress := Data_to_slave.slaveaddress + 1;
32     Data_to_slave.coil_1 := 0;
33   END_IF
34   IF(Data_to_slave.slaveaddress > 11) OR (Data_to_slave.slaveaddress < 2) THEN
35     Data_to_slave.slaveaddress := 2;
36   END_IF
37   Data_to_slave.Data_valid := Data_to_slave.Data_valid + 1;
38   Ton1_En := FALSE;
39 END_IF
40
41 IF NOT(Ton1_Q) AND NOT(Ton1_En) THEN
42   Ton1_En :=TRUE;
43 END_IF
44

```

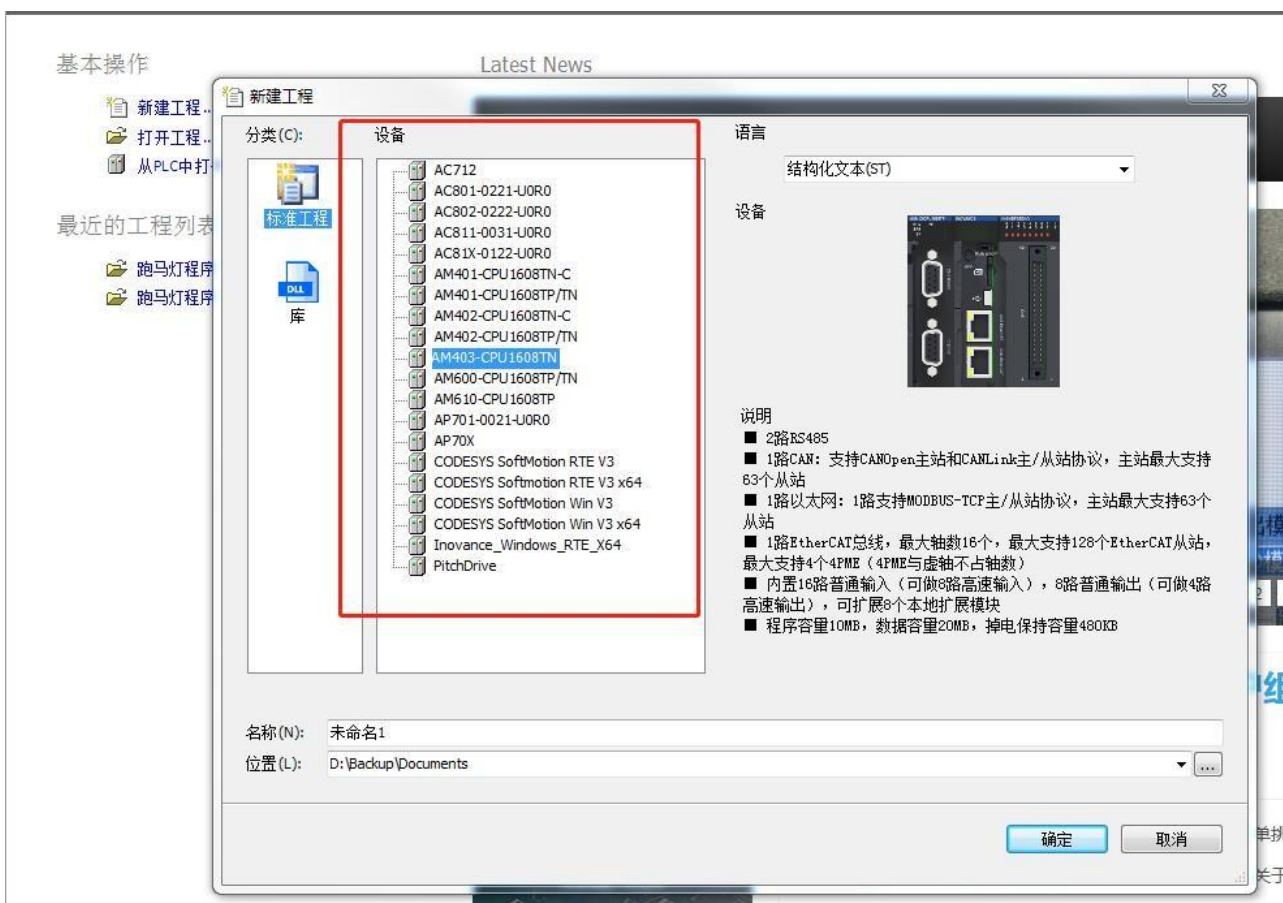


## 六、连接汇川 AM401

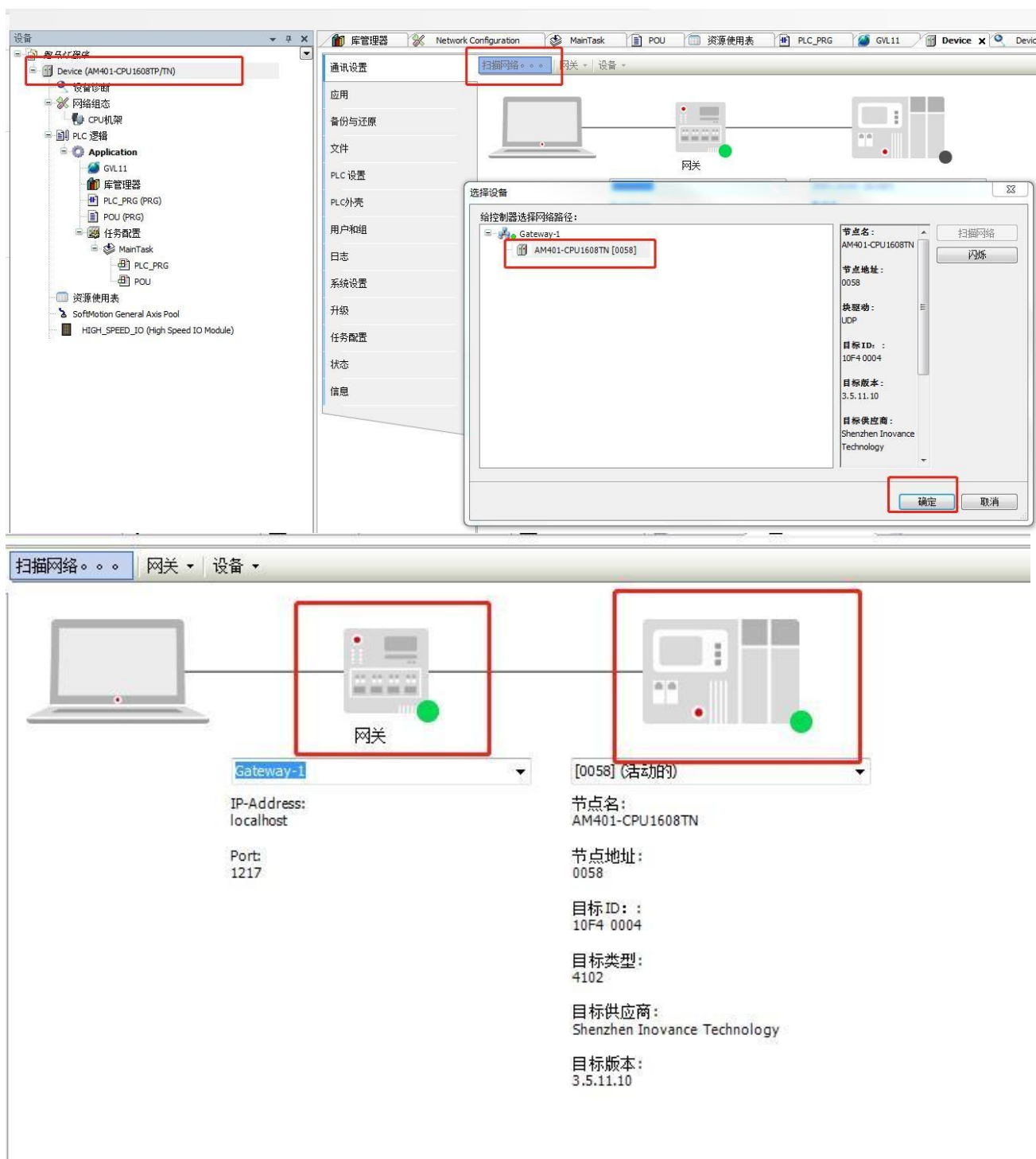
本章节针对 RS485/RS422-EC 与汇川 PLC 的 CODESYS 使用为例以实现相应功能需求。

### 6.1、添加模块

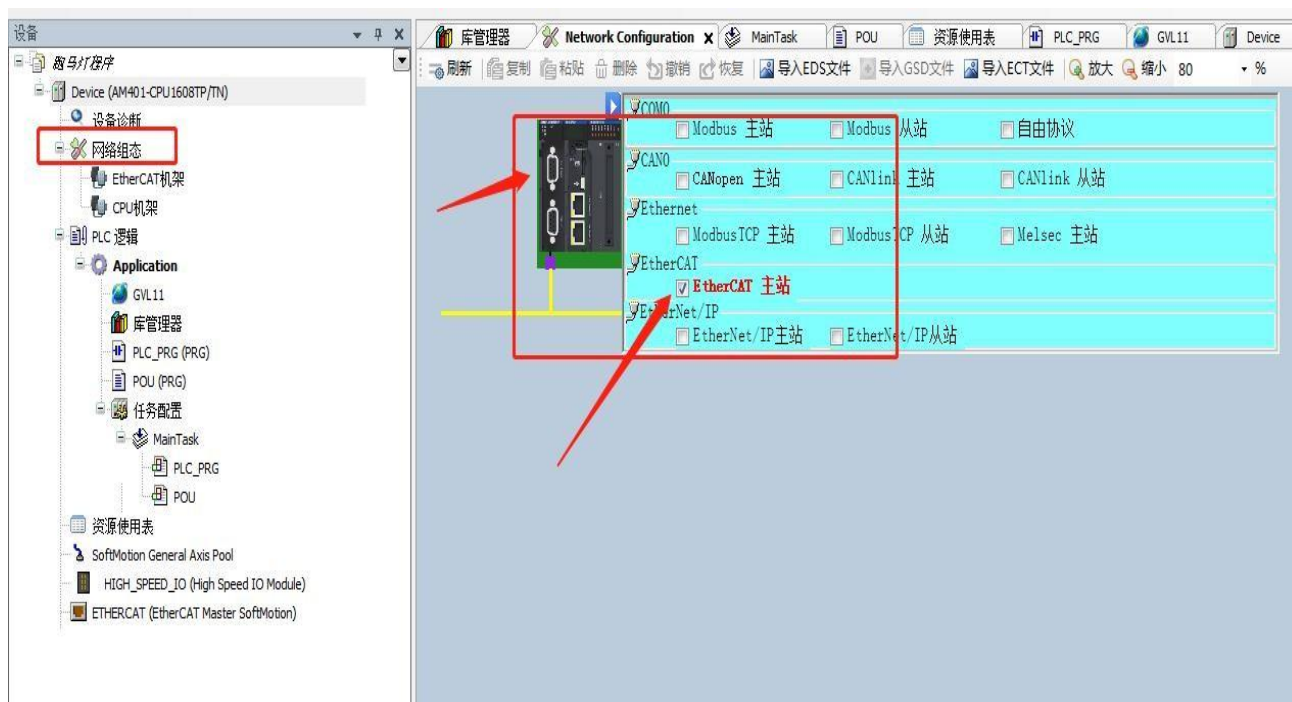
- 1) 打开软件新建一个工程，找到对应的 PLC 型号，如下图：



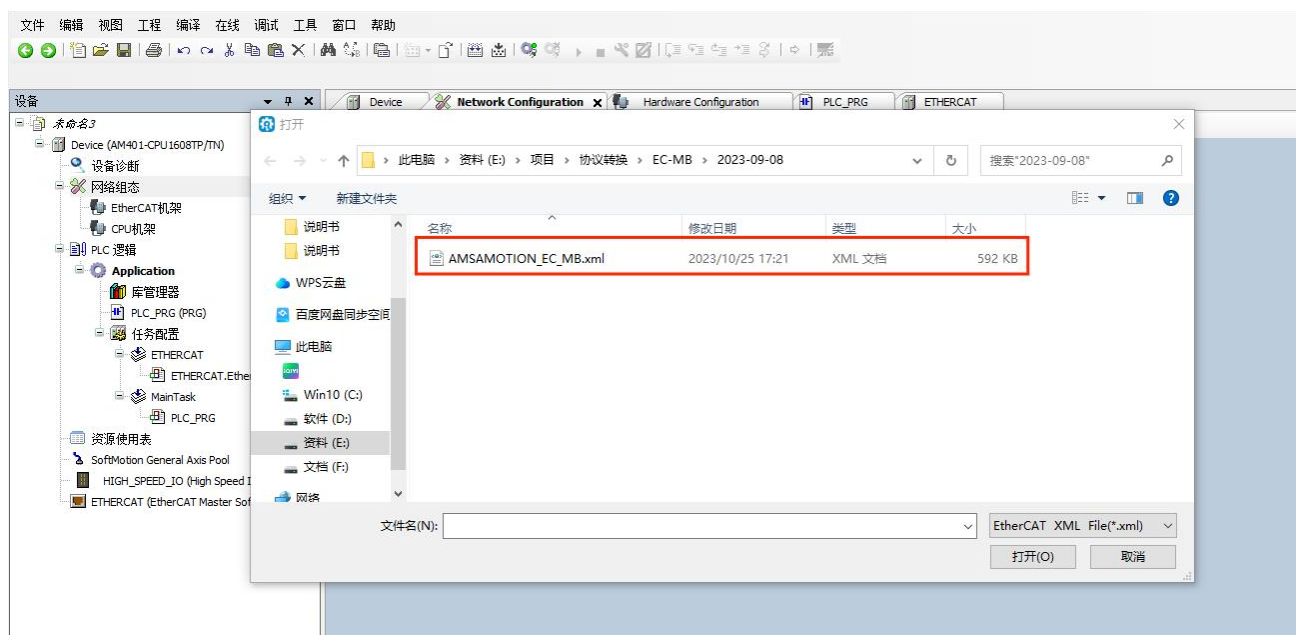
2) 打开 Device, 扫描网络, 选择扫描到的 PLC, 点击确定, 网关和节点都是绿色小点时说明 PLC 已经连接正常



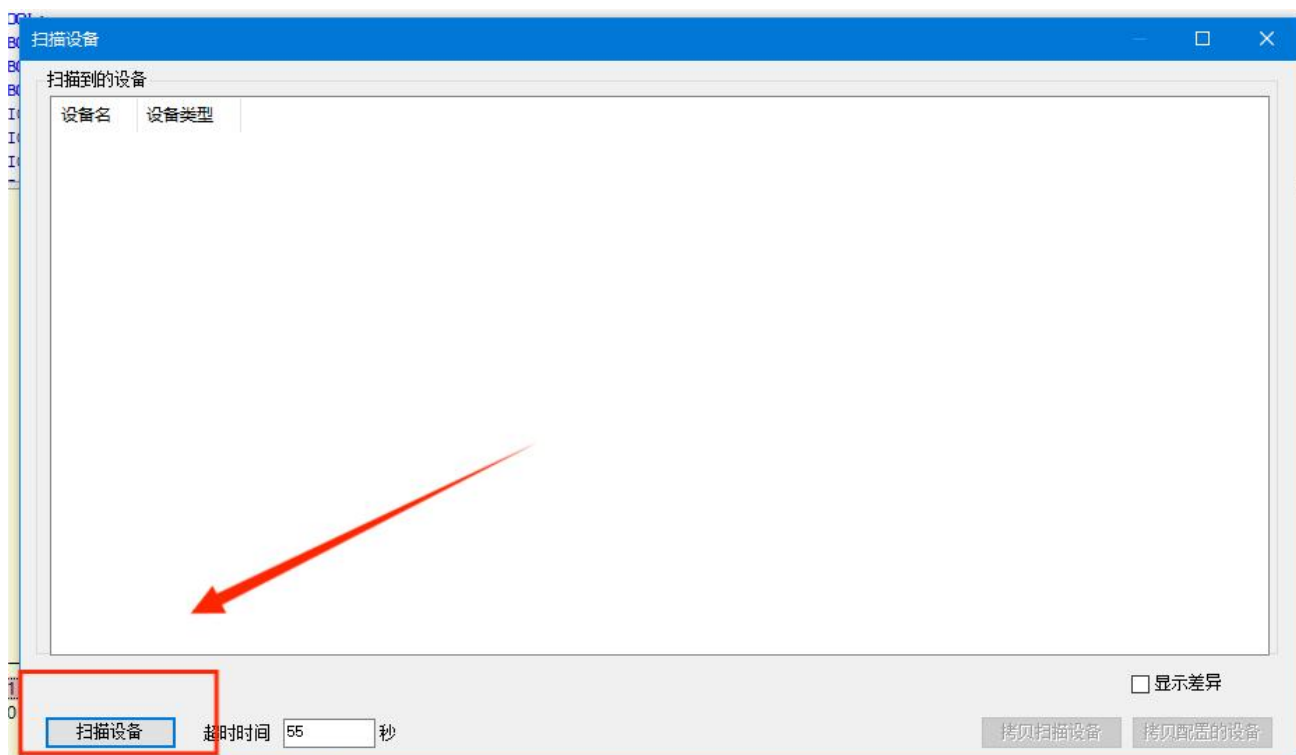
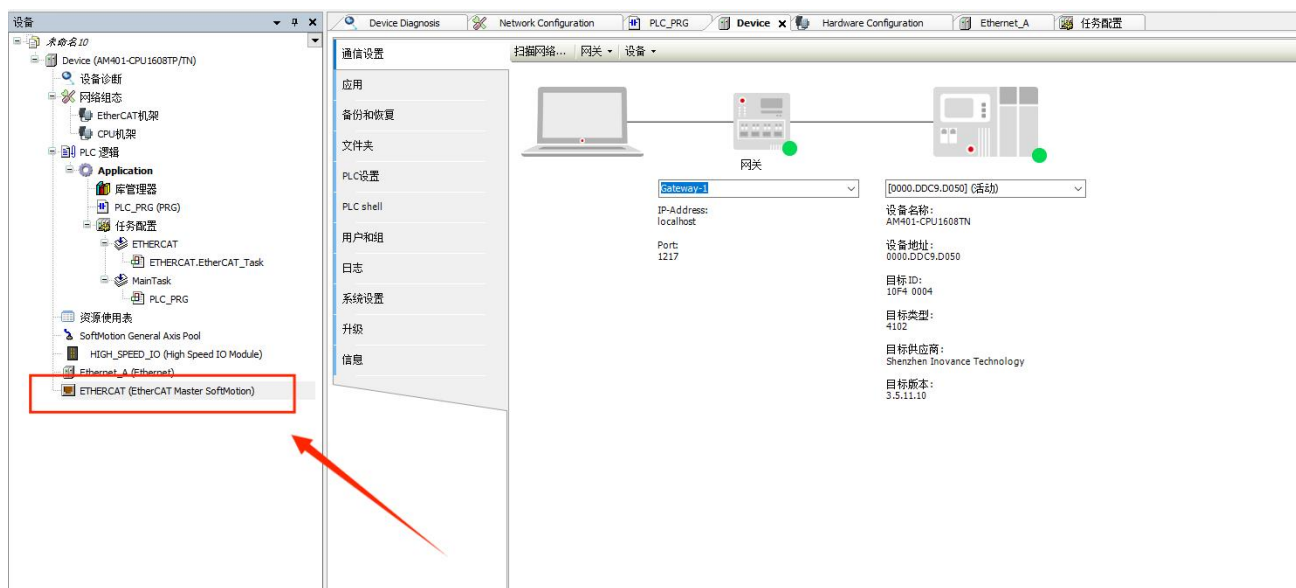
3) 点击网络组态，击 PLC 模型，勾选 EtherCAT 主站



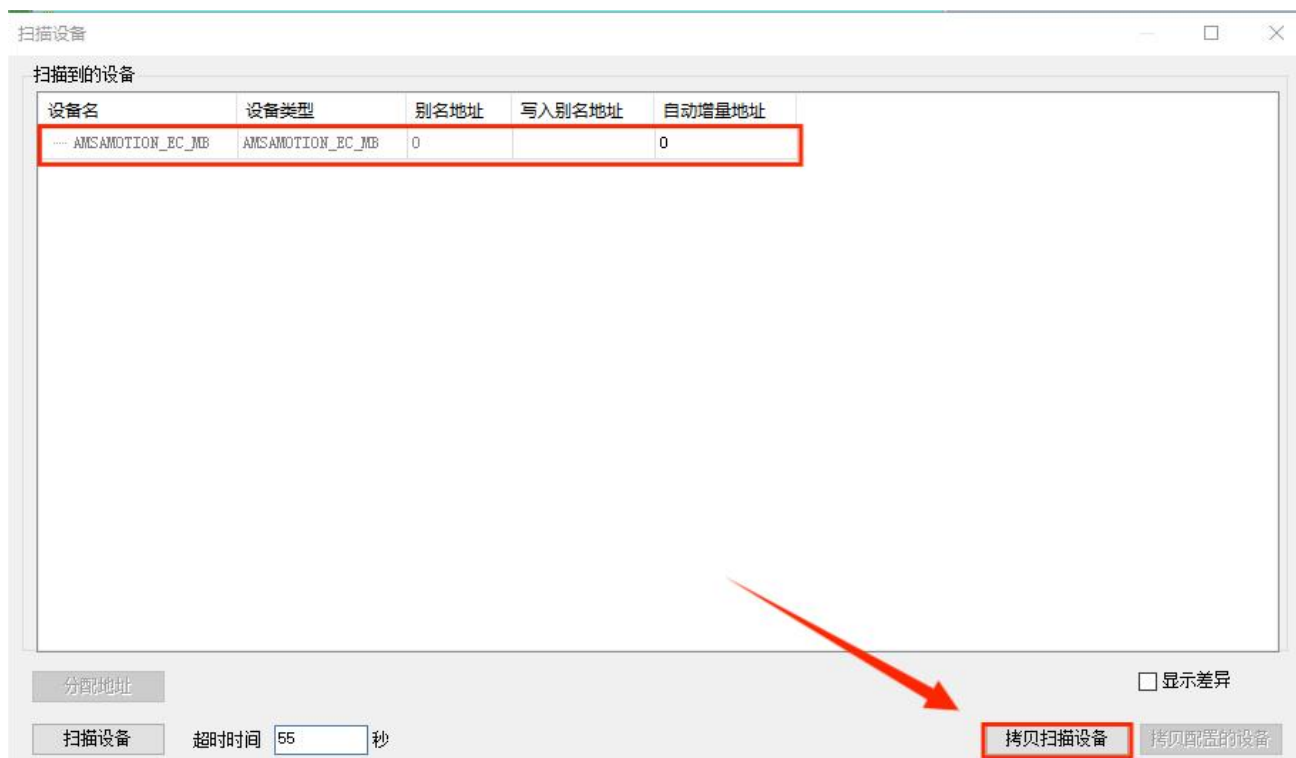
4) 点击左侧网络组态，点击导入 ECT 文件，导入对应模组的 XML



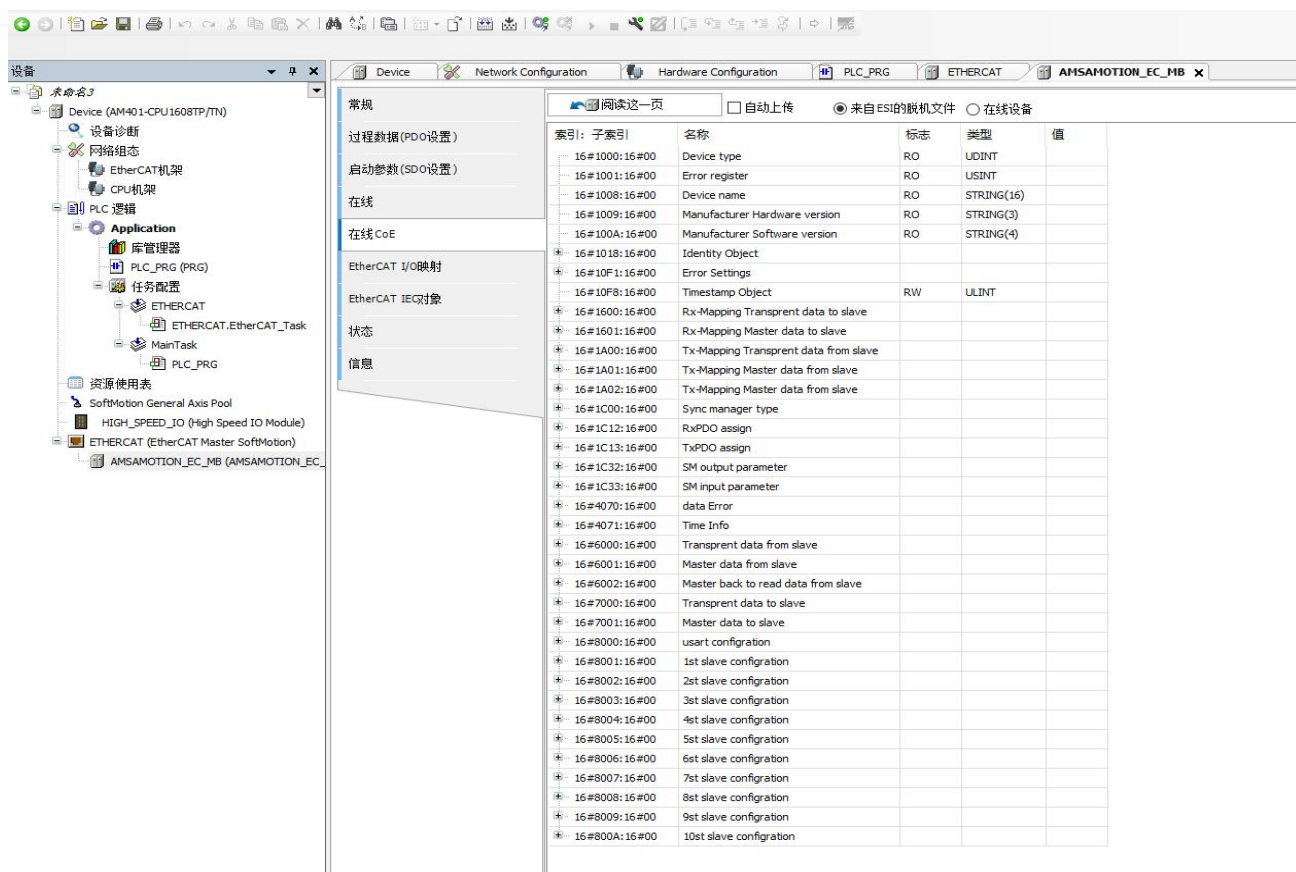
4) 点击 ETHERCAT(EtherCAT Master SoftMotion) 并右键选择扫描设备



## 5) 扫出设备后点击拷贝扫描设备



## 6) 单击在线 COE，下拉可以见到本模块所有对象字典：



## 6.2、ModBus 功能

本例将其设置为 Modbus 模式，从站使用一个 16 位数字输入输出模块，示例用模块参数如下：

|                 |       |
|-----------------|-------|
| Modbus 从站地址     | 1     |
| Modbus 从站接口     | RS485 |
| Modbus 从站波特率    | 9600  |
| Modbus 从站数据格式   | 8N1   |
| Modbus 从站线圈数量   | 16    |
| Modbus 从站离散输入数量 | 16    |

△Note：在单一对象字典无法将某一个从站数据读取完毕时，可以使用多个对象字典读取同一个从站中的值，将寄存器和线圈的起始地址进行相应偏移即可。

所有参数需要在启动参数（SDO 设置）修改

1)单击展开对象字典 8000 对象字：

从对象字典中选择项

| 索引: 子索引       | 名称                      | 标志 | 类型    | 缺省      |
|---------------|-------------------------|----|-------|---------|
| 16#7001:16#00 | Master data to slave    |    |       |         |
| 16#8000:16#00 | usart configuration     |    |       |         |
| :16#01        | Baudrate                | RW | UINT  | 16#0... |
| :16#02        | dataframe               | RW | UINT  | 16#0... |
| :16#03        | Explicit baudrate       | RW | DINT  | 16#0... |
| :16#04        | Polling time            | RW | UINT  | 16#0... |
| :16#05        | Slave Reset             | RW | BOOL  | 16#00   |
| :16#06        | Error Reset             | RW | BOOL  | 16#00   |
| :16#07        | Device Mode             | RW | USINT | 16#01   |
| :16#08        | DEVICE Interface        | RW | USINT | 16#00   |
| 16#8001:16#00 | 1st slave configuration |    |       |         |
| 16#8002:16#00 | 2st slave configuration |    |       |         |
| 16#8003:16#00 | 3st slave configuration |    |       |         |
| 16#8004:16#00 | 4st slave configuration |    |       |         |

名称:

索引: 16#  位长度:

子索引: 16#  值:

☐ 完全访问 ☐ 字节数组



2)选择对象字 8000: 1, 双击下图“1”处, 然后选择“2”处选项, 然后点击 OK:

从对象字典中选择项

| 索引: 子索引        | 名称                        | 标志 | 类型    | 缺省      |
|----------------|---------------------------|----|-------|---------|
| 16#1C13: 16#00 | TxPDO assign              |    |       |         |
| 16#1C32: 16#00 | SM output parameter       |    |       |         |
| 16#1C33: 16#00 | SM input parameter        |    |       |         |
| 16#7000: 16#00 | Transporent data to slave |    |       |         |
| 16#7001: 16#00 | Master data to slave      |    |       |         |
| 16#8000: 16#00 | usart configuration       |    |       |         |
| 16#01          | Baudrate                  | RW | UINT  | 16#0... |
| 16#02          | dataframe                 | RW | UINT  | 16#0... |
| 16#03          | Explicit baudrate         | RW | DINT  | 16#0... |
| 16#04          | Polling time              | RW | UINT  | 16#0... |
| 16#05          | Slave Reset               | RW | BOOL  | 16#00   |
| 16#06          | Error Reset               | RW | BOOL  | 16#00   |
| 16#07          | Device Mode               | RW | USINT | 16#01   |
| 16#08          | DEVICE Interface          | RW | USINT | 16#00   |

名称: Baudrate

索引: 16# 8000 位长度: 16

子索引: 16# 1 值: 9600 Baud

☐ 完全访问 ☐ 字节数组

确定 取消

3)选择对象字 8000: 7, 双击下图“1”处, 然后选择“2”处选项, 然后点击 OK:

从对象字典中选择项

| 索引: 子索引        | 名称                      | 标志 | 类型    | 缺省      |
|----------------|-------------------------|----|-------|---------|
| 16#01          | Baudrate                | RW | UINT  | 16#0... |
| 16#02          | dataframe               | RW | UINT  | 16#0... |
| 16#03          | Explicit baudrate       | RW | DINT  | 16#0... |
| 16#04          | Polling time            | RW | UINT  | 16#0... |
| 16#05          | Slave Reset             | RW | BOOL  | 16#00   |
| 16#06          | Error Reset             | RW | BOOL  | 16#00   |
| 16#07          | Device Mode             | RW | USINT | 16#01   |
| 16#08          | DEVICE Interface        | RW | USINT | 16#00   |
| 16#8001: 16#00 | 1st slave configuration |    |       |         |
| 16#8002: 16#00 | 2st slave configuration |    |       |         |
| 16#8003: 16#00 | 3st slave configuration |    |       |         |
| 16#8004: 16#00 | 4st slave configuration |    |       |         |
| 16#8005: 16#00 | 5st slave configuration |    |       |         |
| 16#8006: 16#00 | 6st slave configuration |    |       |         |

名称: Device Mode

索引: 16# 8000 位长度: 8

子索引: 16# 7 值: Modbus

☐ 完全访问 ☐ 字节数组

确定 取消

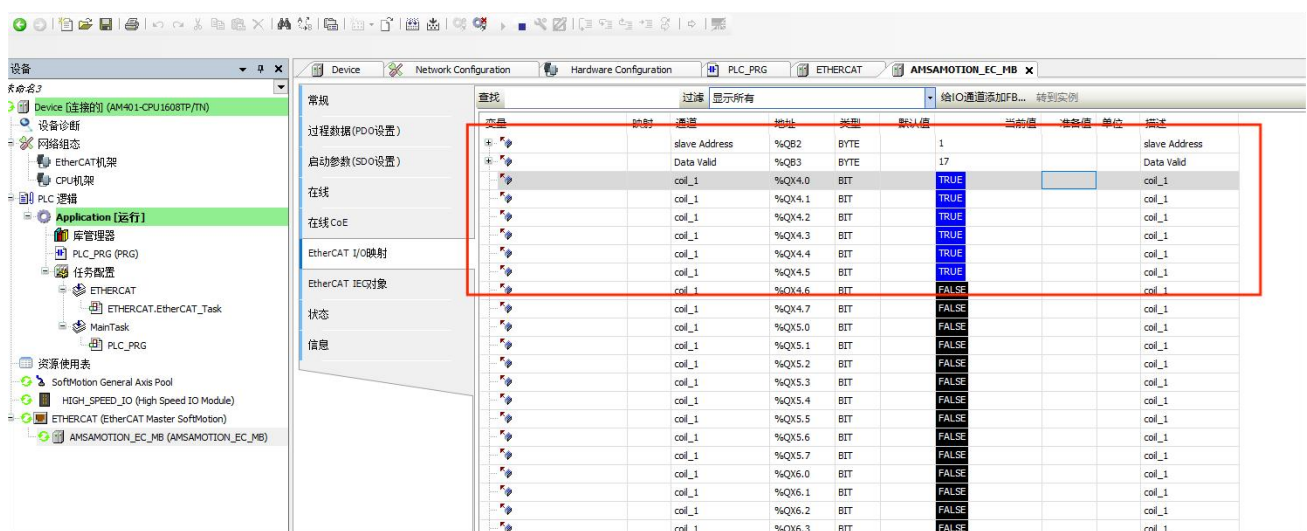
4) 完成后对象字 8000 所有子索引如下图：

|               |                     |    |       |           |
|---------------|---------------------|----|-------|-----------|
| 16#8000:16#00 | usart configuration | RO | USINT | 8         |
| :16#01        | Baudrate            | RW | UDINT | 9600 Baud |
| :16#02        | dataframe           | RW | UDINT | 8N1       |
| :16#03        | Explicit baudrate   | RW | DINT  | 9600      |
| :16#04        | Polling time        | RW | UINT  | 100       |
| :16#05        | Slave Reset         | RW | BOOL  | False     |
| :16#06        | Error Reset         | RW | BOOL  | False     |
| :16#07        | Device Mode         | RW | USINT | Modbus    |
| :16#08        | DEVICE Interface    | RW | USINT | RS485     |

5) 展开对象字 8001，同样将其设置为下图所示：

|               |                                    |    |       |       |
|---------------|------------------------------------|----|-------|-------|
| 16#8001:16#00 | 1st slave configuration            | RO | USINT | 15    |
| :16#01        | slave Addr                         | RW | UINT  | 1     |
| :16#02        | coil is readable                   | RW | BOOL  | True  |
| :16#03        | Keeps the register readable        | RW | BOOL  | False |
| :16#05        | Slave coil start address           | RW | UINT  | 0     |
| :16#06        | The number of slave coil           | RW | UINT  | 16    |
| :16#08        | slave Discrete input start address | RW | UINT  | 0     |
| :16#09        | The number of slave Discrete input | RW | UINT  | 0     |
| :16#0B        | Slave input register start address | RW | UINT  | 0     |
| :16#0C        | The number of Slave input register | RW | UINT  | 0     |
| :16#0E        | Slave hold register start address  | RW | UINT  | 0     |
| :16#0F        | The number of Slave hold register  | RW | UINT  | 0     |

6) 如果需要写线圈输出或者写保持寄存器，将分别将要写的从站地址，需要写的数据写入下图中的地方，然后更改 Data Valid 位，modbus 将会把需要更新的从站数据发出。具体请看 4.3 使用说明





## 6.3、透明传输功能

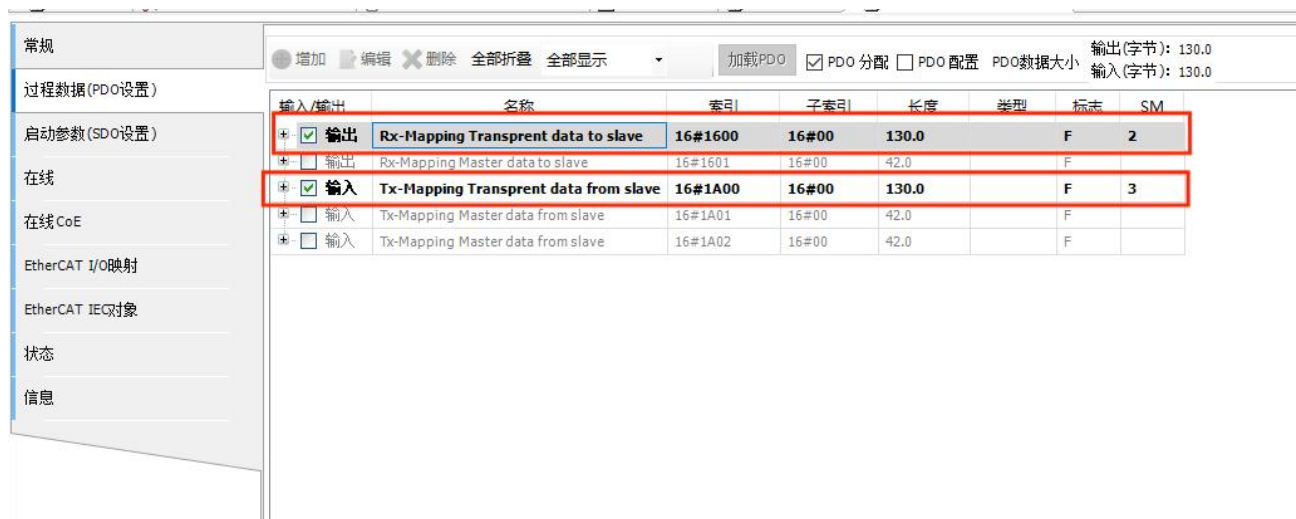
本章节在 6.1 章节基础上进行修改。

- 1) 将运行模式切换回初始化。

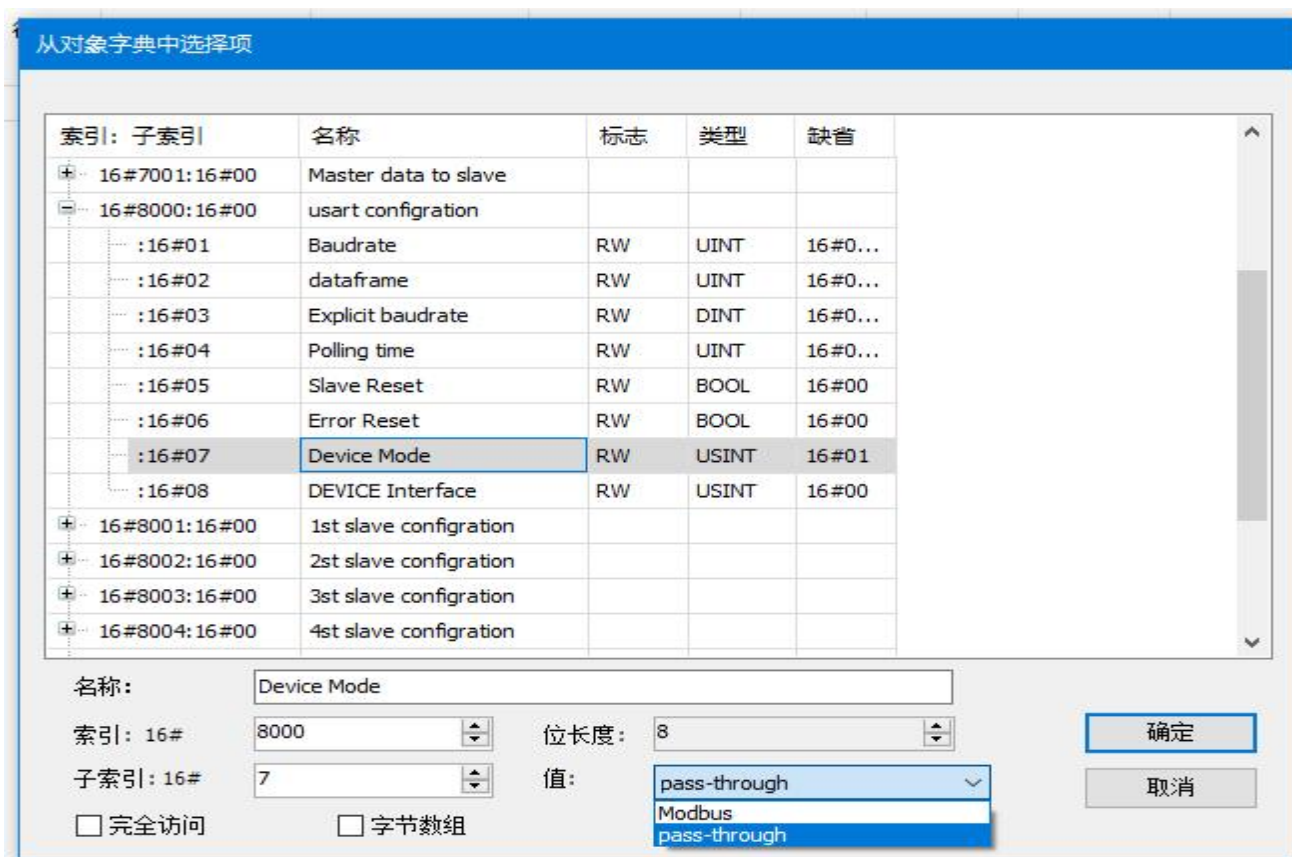
**Note:** 如果不是处于初始化模式下，大部分对象字是不可修改的，否则会报错。



- 2) 按下图将同步管理器输入输出更换

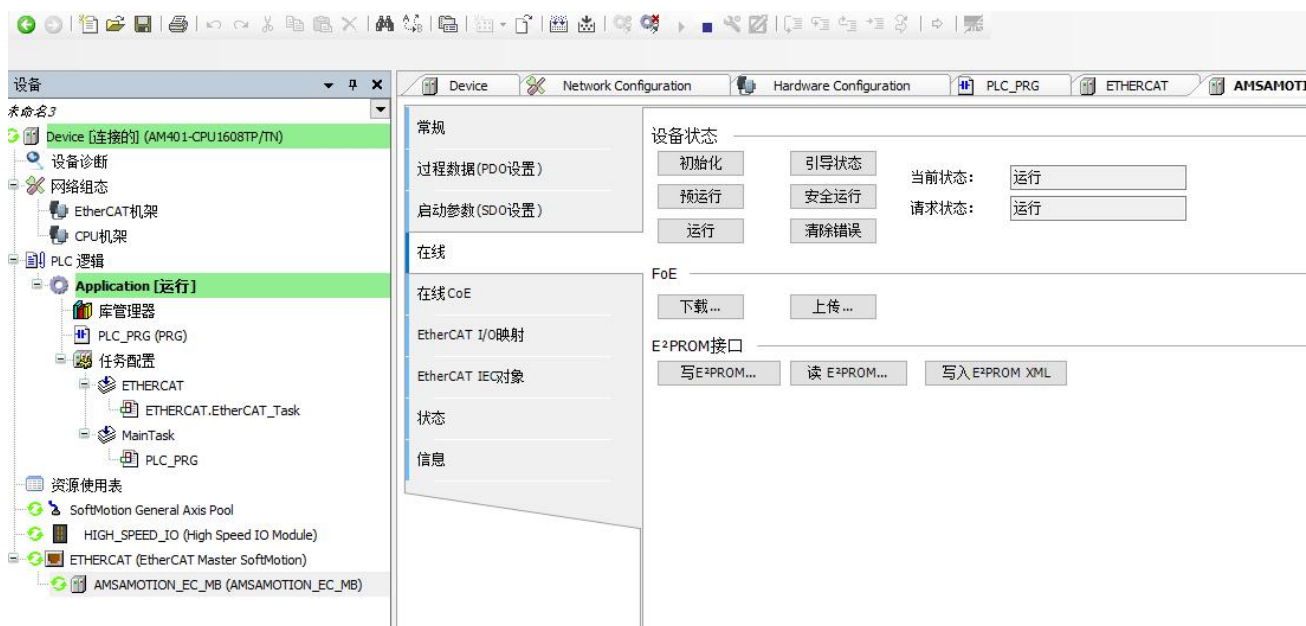


- 3) 按下图点击标题栏，将界面切换到启动参数（SDO 设置）并修改成透传模式



- 4) 波特率、数据格式、接口等按实际需求进行设置，完成后登录到设备并运行设备，此时 RUN 灯以 1 秒周期闪烁，RS485 或者 RS422 灯以 0.2 秒周期闪烁：

**Note:** 使用透传功能时，索引 8001-800A 从站设置未使用，可以是任意合理的值。



5) 将界面切换到 Ether CAT IO 映射

常规

过程数据(PDO设置)

启动参数(SDO设置)

在线

在线CoE

EtherCAT I/O映射

EtherCAT IEC对象
























状态

信息

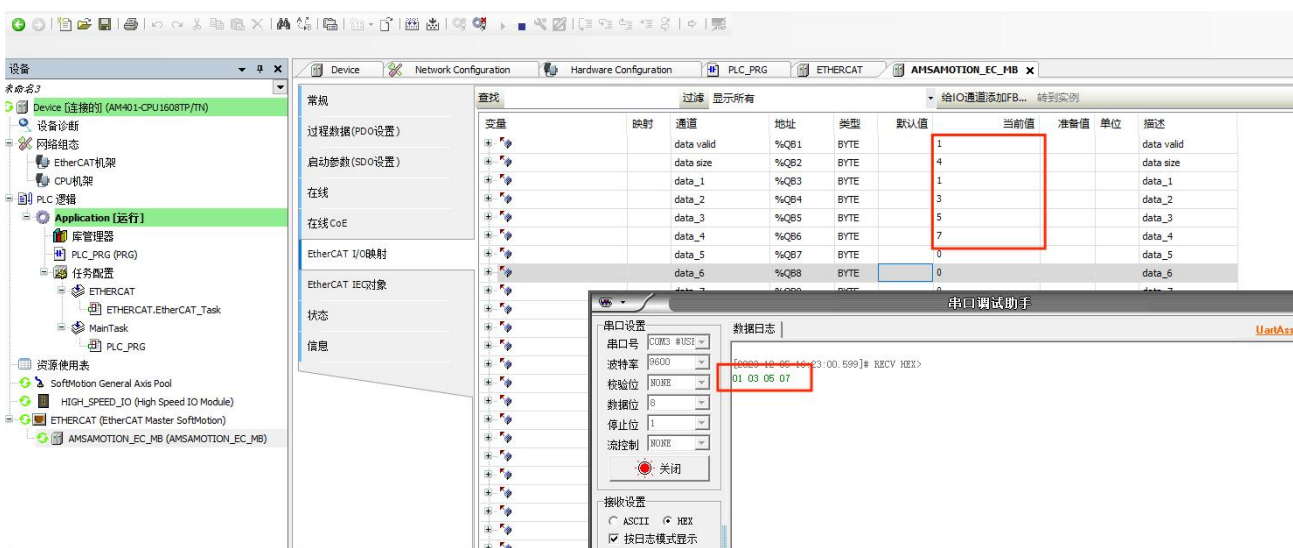
查找

过滤 显示所有

给IO通道添加FB... 转到实例

| 变量  | 映射 | 通道         | 地址    | 类型   | 默认值 | 当前值 | 准备值 | 单位 | 描述         |
|---|----|------------|-------|------|-----|-----|-----|----|------------|
|  |    | data valid | %QB1  | BYTE | 0   |     |     |    | data valid |
|  |    | data size  | %QB2  | BYTE | 0   |     |     |    | data size  |
|  |    | data_1     | %QB3  | BYTE | 0   |     |     |    | data_1     |
|  |    | data_2     | %QB4  | BYTE | 0   |     |     |    | data_2     |
|  |    | data_3     | %QB5  | BYTE | 0   |     |     |    | data_3     |
|  |    | data_4     | %QB6  | BYTE | 0   |     |     |    | data_4     |
|  |    | data_5     | %QB7  | BYTE | 0   |     |     |    | data_5     |
|  |    | data_6     | %QB8  | BYTE | 0   |     |     |    | data_6     |
|  |    | data_7     | %QB9  | BYTE | 0   |     |     |    | data_7     |
|  |    | data_8     | %QB10 | BYTE | 0   |     |     |    | data_8     |
|  |    | data_9     | %QB11 | BYTE | 0   |     |     |    | data_9     |
|  |    | data_10    | %QB12 | BYTE | 0   |     |     |    | data_10    |
|  |    | data_11    | %QB13 | BYTE | 0   |     |     |    | data_11    |
|  |    | data_12    | %QB14 | BYTE | 0   |     |     |    | data_12    |
|  |    | data_13    | %QB15 | BYTE | 0   |     |     |    | data_13    |
|  |    | data_14    | %QB16 | BYTE | 0   |     |     |    | data_14    |
|  |    | data_15    | %QB17 | BYTE | 0   |     |     |    | data_15    |
|  |    | data_16    | %QB18 | BYTE | 0   |     |     |    | data_16    |
|  |    | data_17    | %QB19 | BYTE | 0   |     |     |    | data_17    |
|  |    | data_18    | %QB20 | BYTE | 0   |     |     |    | data_18    |
|  |    | data_19    | %QB21 | BYTE | 0   |     |     |    | data_19    |
|  |    | data_20    | %QB22 | BYTE | 0   |     |     |    | data_20    |
|  |    | data_21    | %QB23 | BYTE | 0   |     |     |    | data_21    |

6) 打开一个串口助手，连接好硬件，发送任意字符，完成后可以看到模块显示和串口发送的数据一致：



注意：data valid 为数据刷新 0 和 1 交替使用

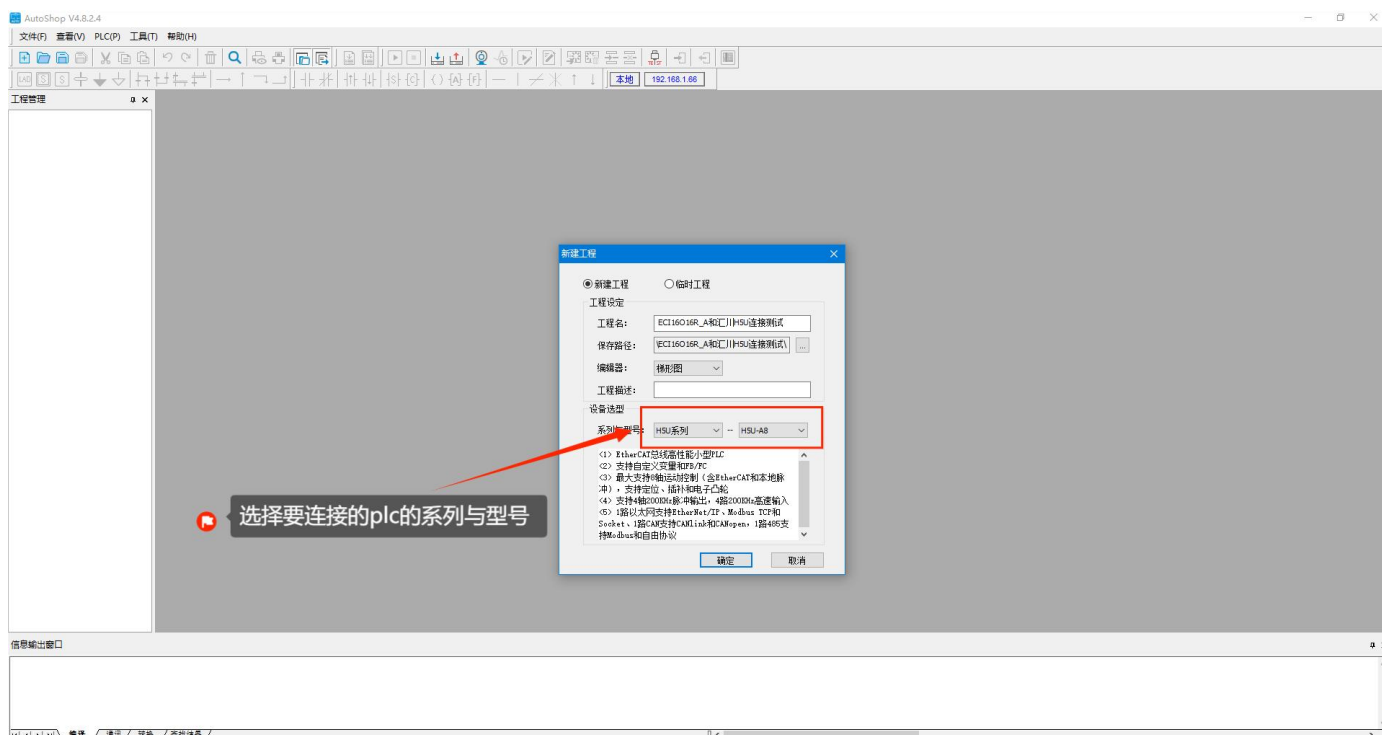
data Size 为数据长度

## 七、连接汇川 H5U-A8

本章节针对 RS485/RS422-EC 与汇川 H5U-A8 使用为例以实现相应功能需求。

### 7.1、添加模块

#### 1) 新建一个工程并且选择相应的系列与型号

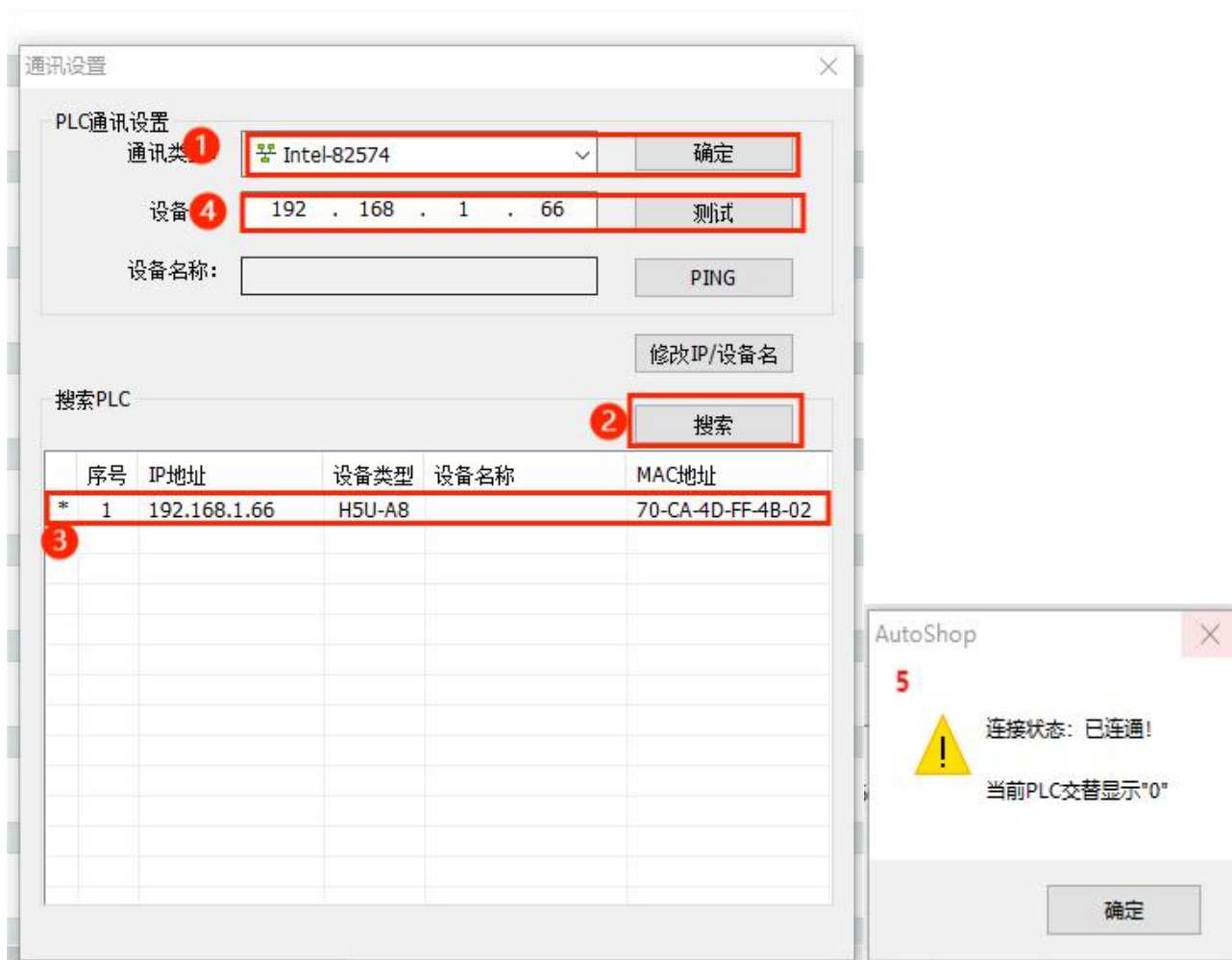


#### 2) 点击测试通讯状态进行通讯设置

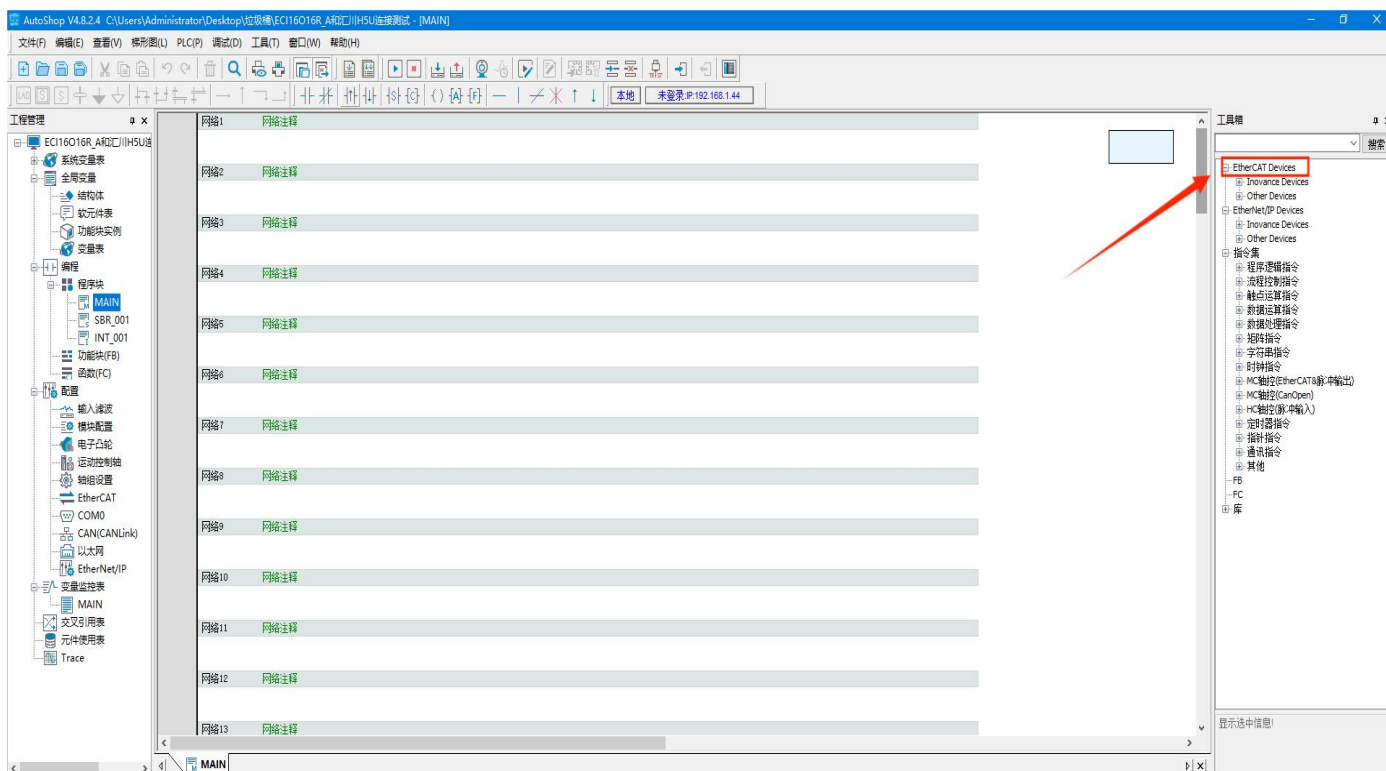


3) 1.通讯类型选择连接 plc 对应的网卡 2.点击搜索 3.查看连接对应设备的详情 4.修改设备的 IP 并且

点击测试(设备 IP 具体请看第 3 步搜索出来的设备 IP) 与 plc 连接成功会出现 5 号图的情况



4) 右键 EtherCAT Devices 选择导入设备 XML 文件 PS: 导入 XML 后需要重启软件





自动扫描

| 序号 | 当前从站列表 |
|----|--------|
|    |        |
|    |        |
|    |        |
|    |        |
|    |        |
|    |        |
|    |        |
|    |        |
|    |        |
|    |        |

| 序号 | 扫描从站列表 | 别名 | 信息 |
|----|--------|----|----|
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |
|    |        |    |    |

2.点击开始扫描

开始扫描

更新组态

退出

自动扫描

| 序号 | 当前从站列表 |  |
|----|--------|--|
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |
|    |        |  |

| 序号 | 扫描从站列表           | 别名 | 信息 |
|----|------------------|----|----|
| 1  | AMSAMOTION_EC_MB | 0  |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |
|    |                  |    |    |

开始扫描  
正在解析扫描数据  
扫描成功!

3.以上是扫出的设备，点击更新组态

开始扫描 更新组态 退出

6) 点击扫出的设备可以查看设备的设置和 I/O 功能映射等功能

工程树: 333 [H5U-A8] > 系统变量表 > 全局变量 > 程序块 > MAIN > SBR\_001 > INT\_001 > 功能块(FB) > 函数(FC) > 配置 > 输入滤波 > 模块配置 > 电子凸轮 > 运动控制轴 > 轴组设置 > EtherCAT > AMSAMOTION\_EC > COM0 > CAN(CANLink) > 以太网 > EtherNet/IP > 变量监控表 > MAIN > 交叉引用表 > 元件使用表 > Trace

十六进制显示当前值

| 变量      | 通道  | 类型               | 当前值      |
|---------|-----|------------------|----------|
| _IQ1_0  | ... | slave Address    | BYTE     |
| _IQ1_1  | ... | Data Valid       | BYTE     |
| _IQ1_2  | ... | coil_1           | BITARR32 |
| _IQ1_3  | ... | coil_2           | BITARR32 |
| _IQ1_4  | ... | hold_reg data_1  | INT      |
| _IQ1_5  | ... | hold_reg data_2  | INT      |
| _IQ1_6  | ... | hold_reg data_3  | INT      |
| _IQ1_7  | ... | hold_reg data_4  | INT      |
| _IQ1_8  | ... | hold_reg data_5  | INT      |
| _IQ1_9  | ... | hold_reg data_6  | INT      |
| _IQ1_10 | ... | hold_reg data_7  | INT      |
| _IQ1_11 | ... | hold_reg data_8  | INT      |
| _IQ1_12 | ... | hold_reg data_9  | INT      |
| _IQ1_13 | ... | hold_reg data_10 | INT      |
| _IQ1_14 | ... | hold_reg data_11 | INT      |
| _IQ1_15 | ... | hold_reg data_12 | INT      |
| _IQ1_16 | ... | hold_reg data_13 | INT      |
| _IQ1_17 | ... | hold_reg data_14 | INT      |
| _IQ1_18 | ... | hold_reg data_15 | INT      |
| _IQ1_19 | ... | hold_reg data_16 | INT      |
| _IQ1_20 | ... | slave Address    | BYTE     |
| _IQ1_21 | ... | Data Valid       | BYTE     |
| _IQ1_22 | ... | D_Input_1        | BITARR32 |
| _IQ1_23 | ... | D_Input_2        | BITARR32 |
| _IQ1_24 | ... | Input reg data_1 | INT      |

## 7.2、ModBus 功能

本例将其设置为 Modbus 模式，从站使用一个 16 位数字输入输出模块，示例用模块参数如下：

|                 |       |
|-----------------|-------|
| Modbus 从站地址     | 1     |
| Modbus 从站接口     | RS485 |
| Modbus 从站波特率    | 9600  |
| Modbus 从站数据格式   | 8N1   |
| Modbus 从站线圈数量   | 16    |
| Modbus 从站离散输入数量 | 16    |

△Note：在单一对象字典无法将某一个从站数据读取完毕时，可以使用多个对象字典读取同一个从站中的值，将寄存器和线圈的起始地址进行相应偏移即可。

**所有参数需要在启动参数修改（根据需求在增加里面添加）**

**PS：H5U 没下拉框所有设置对照 4.2 对象字典进行设置**

1)单击展开对象字典 8000 对象字：

The screenshot shows the '启动参数' (Start Parameters) configuration window. On the left, a tree view shows the project structure with 'AMSAMOTION\_EC' selected. The main area displays a table of parameters for the 16#8000 object dictionary. The '新增/编辑' (Add/Edit) dialog box is open, allowing the user to add or modify parameters. The dialog box contains the following fields:

- 名称 (Name):
- 索引: 16# (Index: 16#):
- 子索引: 16# (Sub-index: 16#):
- 位长度 (Bit length):
- 值 (Value):

The '确定' (OK) button is at the bottom right of the dialog box.

2) 选择对象字 8000 进行参数设置 一般只修改 8000: 01 (波特率) 8000: 07 (设备模式) 8000: 08 (接口模式) 如果有特殊需求按照 4.2 章节进行设置。

|         |   |               |                  |   |     |
|---------|---|---------------|------------------|---|-----|
| 常规设置    | <div><div><div><div><div></div><div>增加</div></div><div><div></div><div>编辑</div></div><div><div></div><div>删除</div></div></div><div><input checked="" type="checkbox"/>隐藏系统参数<input type="checkbox"/>十六进制显示当前值</div></div></div> |               |                  |   |     |
| 过程数据    | 行号  | 索引:子索引        | 名称               | 值 | 位长度 |
| 启动参数    | 1   | 16#8000:16#01 | Baudrate         | 3 | 16  |
|         | 2   | 16#8000:16#07 | Device Mode      | 0 | 8   |
|         | 3   | 16#8000:16#08 | DEVICE Interface | 0 | 8   |
| I/O功能映射 |   |               |                  |   |     |
| 信息      |   |               |                  |   |     |
| 状态      |   |               |                  |   |     |

3)展开对象字 8001，同样将其设置为下图所示：

常规设置

过程数据

启动参数

I/O功能映射

信息

状态

增加

编辑

删除

☒隐藏系统参数

☐十六进制显示当前值

| 行号 | 索引:子索引        | 名称       | 值 | 位长 |
|----|---------------|----------|---|----|
| 1  | 16#8000:16#01 | Baudrate | 3 | 16 |

新增/编辑

| 索引:子索引        | 名称                                 | 标志 | 类型    | 默认值 |
|---------------|------------------------------------|----|-------|-----|
| 16#8000:16#00 | usart configuration                |    | USINT |     |
| 16#8001:16#00 | 1st slave configuration            |    | USINT |     |
| 16:1          | slave Addr                         | RW | BYTE  |     |
| 16:2          | coil is readable                   | RW | BOOL  |     |
| 16:3          | Keeps the register readable        | RW | BOOL  |     |
| 16:5          | Slave coil start address           | RW | UINT  |     |
| 16:6          | The number of slave coil           | RW | BYTE  |     |
| 16:8          | slave Discrete input start address | RW | UINT  |     |
| 16:9          | The number of slave Discrete input | RW | BYTE  |     |
| 16:B          | Slave input register start address | RW | UINT  |     |
| 16:C          | The number of Slave input register | RW | BYTE  |     |

名称:coil is readable

索引: 16#8001

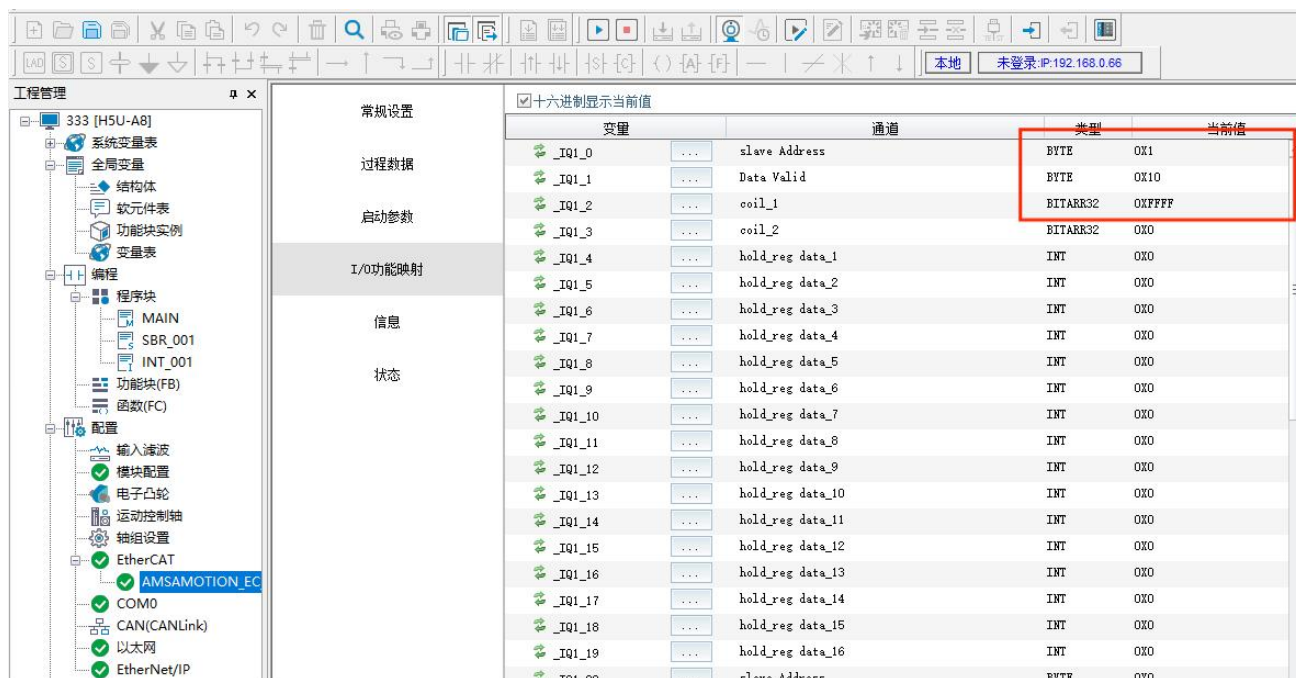
子索引: 16#2

位长度:1

值:1

确定

4) 如果需要写线圈输出或者写保持寄存器，将分别将要写的从站地址，需要写的数据写入下图中的地方，然后更改 Data Valid 位，modbus 将会把需要更新的从站数据发出。具体请看 4.3 使用说明



Slave address 从站地址

Data valid 数据刷新

Coil\_1 线圈 1



## 7.3、透明传输功能

本章节在 7.1 章节基础上进行修改。

1) 停止 plc 让设备断开 OP 状态

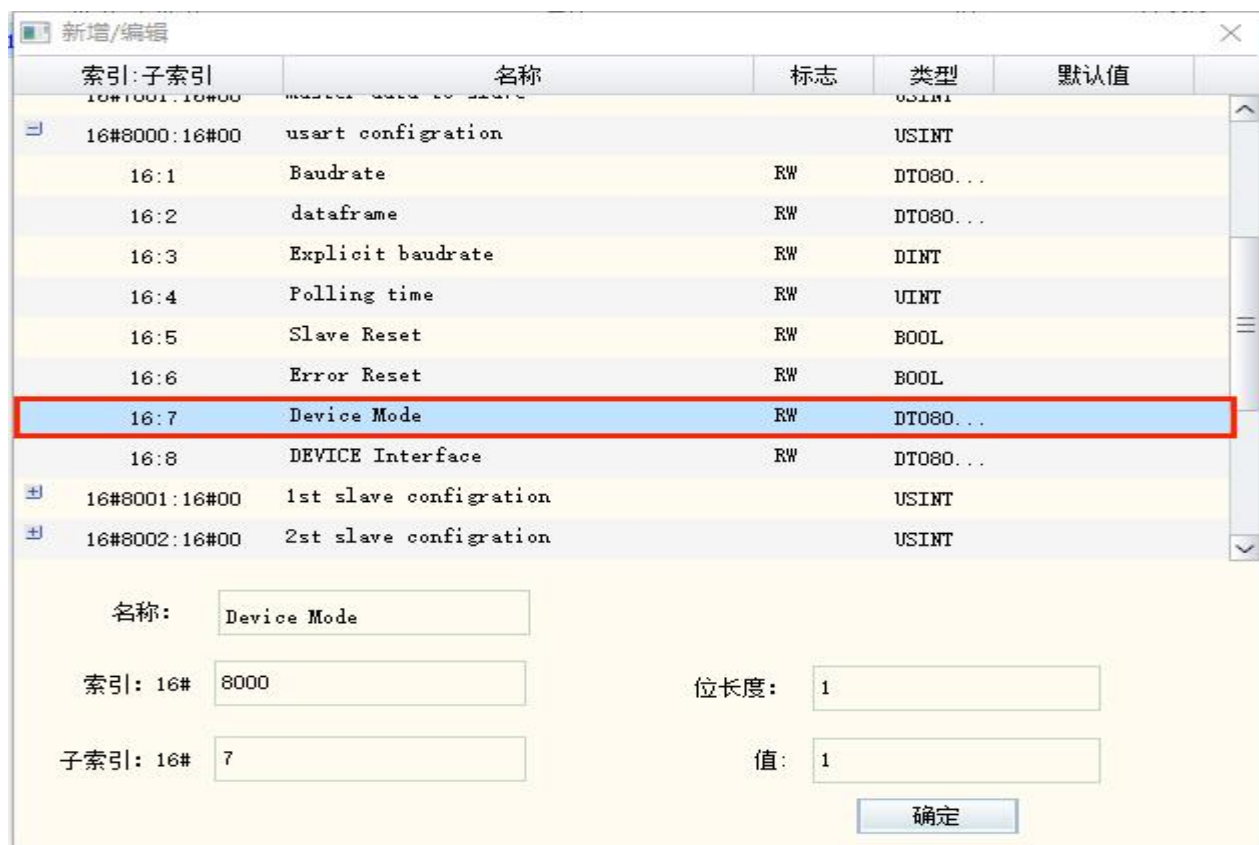
**Note:** 如果不是处于初始化模式下，大部分对象字是不可修改的，否则会报错。

| 输入/输出                                  | 名字                                     | 索引      | 子索引   | 长度    | 标志 | SM | 类型 |
|--|--|---------|-------|-------|----|----|----|
| <input type="checkbox"/> 输出            | Rx-Mapping Transparent data to slave   | 16#1600 | 16#00 | 130.0 | F  | 2  |    |
| <input checked="" type="checkbox"/> 输出 | Rx-Mapping Master data to slave        | 16#1601 | 16#00 | 42.0  | F  | 2  |    |
| <input type="checkbox"/> 输入            | Tx-Mapping Transparent data from slave | 16#1A00 | 16#00 | 130.0 | F  | 3  |    |
| <input checked="" type="checkbox"/> 输入 | Tx-Mapping Master data from slave      | 16#1A01 | 16#00 | 42.0  | F  | 3  |    |
| <input checked="" type="checkbox"/> 输入 | Tx-Mapping Master data from slave      | 16#1A02 | 16#00 | 42.0  | F  | 3  |    |

2) 按下图将同步管理器输入输出更换

| 输入/输出                                  | 名字                                     | 索引      | 子索引   | 长度    | 标志 | SM | 类型 |
|--|--|---------|-------|-------|----|----|----|
| <input checked="" type="checkbox"/> 输出 | Rx-Mapping Transparent data to slave   | 16#1600 | 16#00 | 130.0 | F  | 2  |    |
| <input checked="" type="checkbox"/> 输出 | Rx-Mapping Master data to slave        | 16#1601 | 16#00 | 42.0  | F  | 2  |    |
| <input checked="" type="checkbox"/> 输出 | Tx-Mapping Transparent data from slave | 16#1A00 | 16#00 | 130.0 | F  | 3  |    |
| <input type="checkbox"/> 输入            | Tx-Mapping Master data from slave      | 16#1A01 | 16#00 | 42.0  | F  | 3  |    |
| <input type="checkbox"/> 输入            | Tx-Mapping Master data from slave      | 16#1A02 | 16#00 | 42.0  | F  | 3  |    |

3) 按下图点击标题栏，将界面切换到启动参数并将 8000: 07 修改成透传模式

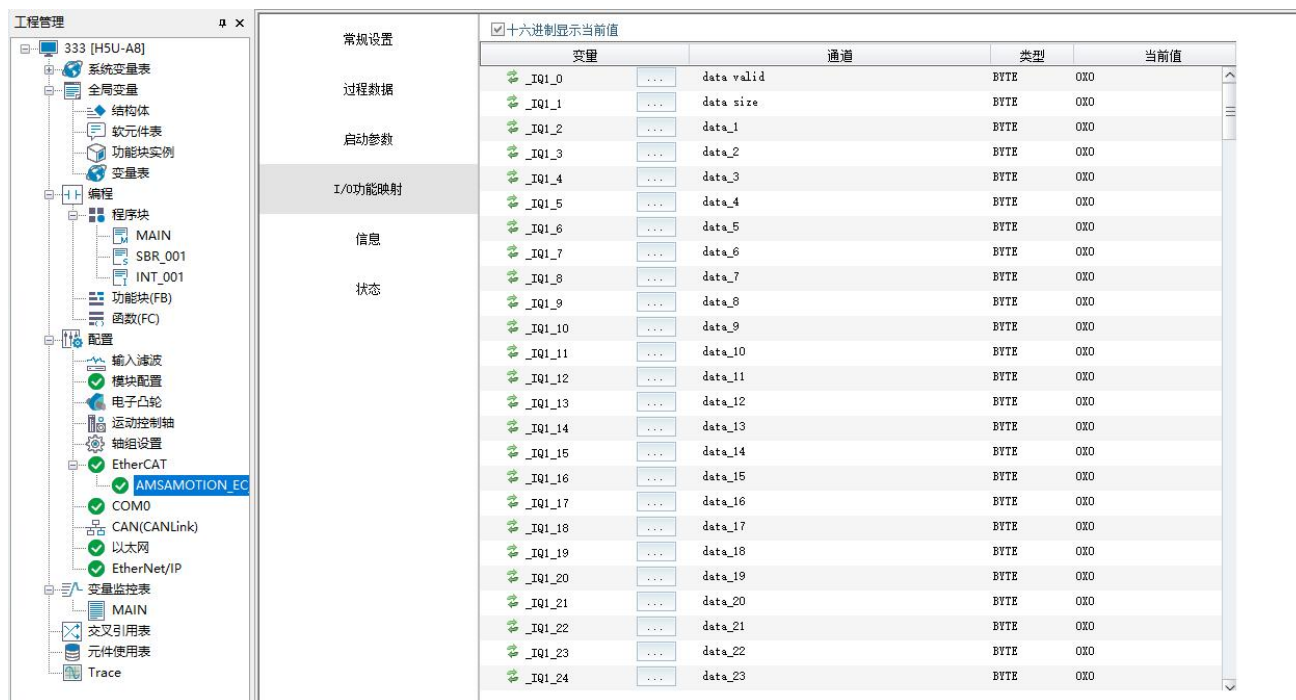


4) 波特率、数据格式、接口等按实际需求进行设置，完成后登录到设备并运行设备，此时 RUN 灯以 1 秒周期闪烁，RS485 或者 RS422 灯以 0.2 秒周期闪烁：

**Note:** 使用透传功能时，索引 8001-800A 从站设置未使用，可以是任意合理的值。

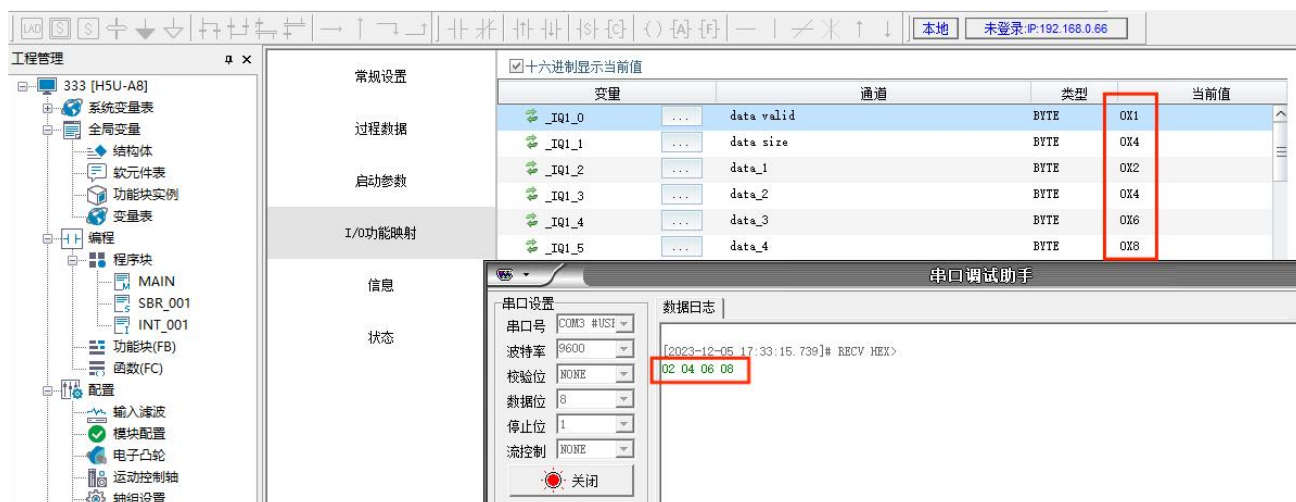


## 5) 将界面切换到 I/O 功能映射



| 变量      | 通道         | 类型   | 当前值 |
|---------|------------|------|-----|
| _IQ1_0  | data valid | BYTE | 0X0 |
| _IQ1_1  | data size  | BYTE | 0X0 |
| _IQ1_2  | data_1     | BYTE | 0X0 |
| _IQ1_3  | data_2     | BYTE | 0X0 |
| _IQ1_4  | data_3     | BYTE | 0X0 |
| _IQ1_5  | data_4     | BYTE | 0X0 |
| _IQ1_6  | data_5     | BYTE | 0X0 |
| _IQ1_7  | data_6     | BYTE | 0X0 |
| _IQ1_8  | data_7     | BYTE | 0X0 |
| _IQ1_9  | data_8     | BYTE | 0X0 |
| _IQ1_10 | data_9     | BYTE | 0X0 |
| _IQ1_11 | data_10    | BYTE | 0X0 |
| _IQ1_12 | data_11    | BYTE | 0X0 |
| _IQ1_13 | data_12    | BYTE | 0X0 |
| _IQ1_14 | data_13    | BYTE | 0X0 |
| _IQ1_15 | data_14    | BYTE | 0X0 |
| _IQ1_16 | data_15    | BYTE | 0X0 |
| _IQ1_17 | data_16    | BYTE | 0X0 |
| _IQ1_18 | data_17    | BYTE | 0X0 |
| _IQ1_19 | data_18    | BYTE | 0X0 |
| _IQ1_20 | data_19    | BYTE | 0X0 |
| _IQ1_21 | data_20    | BYTE | 0X0 |
| _IQ1_22 | data_21    | BYTE | 0X0 |
| _IQ1_23 | data_22    | BYTE | 0X0 |
| _IQ1_24 | data_23    | BYTE | 0X0 |

## 6) 打开一个串口助手，连接好硬件，发送任意字符，完成后可以看到模块显示和串口发送的数据一致：



| 变量     | 通道         | 类型   | 当前值 |
|--------|------------|------|-----|
| _IQ1_0 | data valid | BYTE | 0X1 |
| _IQ1_1 | data size  | BYTE | 0X4 |
| _IQ1_2 | data_1     | BYTE | 0X2 |
| _IQ1_3 | data_2     | BYTE | 0X6 |
| _IQ1_4 | data_3     | BYTE | 0X8 |
| _IQ1_5 | data_4     | BYTE |     |

串口调试助手

串口设置

串口号: COM3 #USB

波特率: 9600

校验位: NONE

数据位: 8

停止位: 1

流控制: NONE

数据日志

[2023-12-05 17:33:15.739]# RECV HEX>

02 04 06 08

注意：data valid 为数据刷新 0 和 1 交替使用

data Size 为数据长度

## 八、错误

### 8.1、模块状态

本模块状态分为四类，等级从低到高分别为正常、从站错误（警告）、主站错误（错误 1）、重启（错误 2）。

不同等级的状态可由 RUN 灯闪烁进行提示，且在对象字 4070 中可查看错误状态。下表是状态和对应的出现原因。

| 状态       | 4070 对象字                          | RUN 灯状态                       | 原因                                 |
|----------|-----------------------------------|-------------------------------|------------------------------------|
| 正常       | 0x00000000                        | 2 秒周期闪烁                       | --                                 |
| 从站<br>错误 | 0x00<<24 (从站索引<<16) (功能码<<8) 0x01 | 1 秒周期闪烁                       | 不合法功能码                             |
|          | 0x00<<24 (从站索引<<16) (功能码<<8) 0x02 |                               | 非法的数据地址                            |
|          | 0x00<<24 (从站索引<<16) (功能码<<8) 0x03 |                               | 非法的数据值或范围                          |
|          | 0x00<<24 (从站索引<<16) (功能码<<8) 0x04 |                               | 从站设备故障                             |
|          | 0x00<<24 (从站索引<<16) (功能码<<8) 0x0D |                               | 接收超时（超过当前波特率接收 300 个字节的时间未接收到返回数据） |
|          | 0x00<<24 (从站索引<<16) (功能码<<8) 0x0E |                               | 接收 CRC 错误，但数据长度>5                  |
|          | 0x00<<24 (从站索引<<16) (功能码<<8) 0x12 |                               | 接收 CRC 错误，且数据长度<5                  |
|          | 0xFF00000F                        |                               | 发送下一条命令时，上一条指令未执行完成                |
| 主站<br>错误 | 0xFF000010                        | 亮 0.5 秒、灭 0.5 秒、亮 0.5 秒、灭 2 秒 | 错误的写从站地址，未设置该从站                    |
|          | 0xFF000011                        |                               | 选择的从站无法使用写功能，或者进入 OP 模式，但未设置任何从站信息 |
|          | 0xFF000014                        |                               | 在透传模式下模块接收到的串口消息过快，出现丢包现象          |
| 重启       | --                                | 亮 0.2 秒、灭 0.2 秒、亮 0.2 秒、灭 1 秒 | 使用了物理 Reset 按钮复位                   |

## 8.2、错误处理

### 1) 从站错误

当出现从站错误，除 0xFF00000F 外，其他均可以先检查从站的状态是否正常，以及本模块的对象字 8001~800A 设置是否合理。

当出现从站错误且主站未停止运行时，0xFF00000F 代码一定会出现。但是，当单独出现 0xFF00000F 时，没有从站错误，请参考第七章内容。

### 2) 主站错误

当出现 0xFF000010 时，先检查 PLC 代码，是否有写从站相应数据，且从站地址未配置到本模块对象字 8001~800A 的子索引 1 中。

当出现 0xFF000011 时，检查是否未配置到本模块对象字 8001~800A 或者设置有误，比如未设置保持寄存器数量，但是 PLC 代码在写相应地址的保持寄存器。

当出现 0xFF000014 时，表明在透明传输模式下，模块接收到的一包长度大于了 128 字节，出现丢包现象。

### 3) 重启

本模块设置有物理 Reset 按钮，当按下该按钮超过三秒时，不管在任何状态，均会见用户数据清除复位成出厂值，此时模块应该需要重新进入 Pre-OP 状态进行初始化，合适的办法是断电 3 秒然后重启模块，如果无法断电，可参考以下两种操作：

- 1) 可将模块运行状态切换回 Pre-OP，然后使用对象字 8000: 6 清除故障，然后重新设置对象字 8000~800A，完成后再将运行模式切换回 OP，开始运行。
- 2) 不使用物理 Reset 按钮，直接将模块运行模式切换成 Pre-Op 模式，将对象字 8000: 5 置 1，然后重新设置对象字 8000~800A，完成后再将运行模式切换回 OP，开始运行。

## 8.3、模块 LED 灯状态

本模块在第 2.2 章节介绍中，有五中 LED 灯，本章节详细介绍各种灯的状态。

### 1) RUN 灯

在模块正常工作时以 2 秒的周期闪烁；

在模块出现从站错误时以 1 秒周期闪烁；

在模块出现主站错误时（错误 1），会以“亮 0.5 秒、灭 0.5 秒、亮 0.5 秒、灭 2 秒”的周期进行双闪；



在模块需要重启时（错误 2），会以“亮 0.2 秒、灭 0.2 秒、亮 0.2 秒、灭 1 秒”的周期进行双闪。

2) EC RUN 灯和 EC ERR 灯

| EC RUN 灯状态 | EC ERR 灯状态 | 模块状态                      |
|------------|------------|---------------------------|
| 灭          | 灭          | 未连接 EtherCAT 主站           |
|            |            | 连接 EtherCAT 主站但处于 Init 状态 |
| 0.2 秒周期闪烁  | 灭          | 模块处于 Pre-Op 状态            |
| 0.5 秒周期闪烁  | 灭          | 模块处于 SAFEOP 状态            |
| 常亮         | 灭          | 模块处于 OP 状态                |
| 灭          | 0.2 秒周期闪烁  | EtherCAT 通讯出现错误           |

3) RS485 灯和 RS422 灯

| RS485 灯   | RS422 灯   | 模块状态                   |
|-----------|-----------|------------------------|
| 1 秒周期闪烁   | 灭         | RS485 接口工作且在 Modbus 模式 |
| 0.5 秒周期闪烁 | 灭         | RS485 接口工作且在透传模式       |
| 灭         | 1 秒周期闪烁   | RS422 接口工作且在 Modbus 模式 |
| 灭         | 0.5 秒周期闪烁 | RS422 接口工作且在透传模式       |
| 灭         | 灭         | 模块未处于 OP 状态            |

## 九、模块轮询时间

在第四章对象字典中有关于轮询时间的设定，具体对象字为 0x8000:4，单位是 ms。该参数设置不当时，即使主站模块未报错，也会出现 0xFF00000F 的错误。

设置时如果不能知道设置多少合适，可先使用默认值运行，在运行时，模块会统计出和从站的通讯时间相关参数，并保存在对象字 4071 中，具体对应见下面说明。

该对象字用于设置模块发送一条指令到发送下一条指令的间隔，设置和应用时应该注意以下几点问题：

1) 发送间隔 = 指令最大长度（字节数） \* 当前波特率每 ms 发送的字节数 + 接收最大长度（字节数） \* 当前波特率每 ms 发送的字节数 + 当前波特率每字节发送消耗的时间 \* 3.5 (向上取整 ms 数) + 从站响应时间。

部分时间计算在对象字 4071 中已经进行粗略计算，可以进行参考对应关系如下：

4071: 3 Total Time : 发送间隔

4071: 2 Tx Time : 指令最大长度（字节数） \* 当前波特率每 ms 发送的字节数

4071: 1 Wait Time : 接收最大长度（字节数） \* 当前波特率每 ms 发送的字节数 + 当前波特率每字节发送消耗的时间 \* 3.5 (向上取整 ms 数) + 从站响应时间

**PS: 默认值为 50，程序上最小设置为 5，当配置小于 5 时会自动按 5 来设置**

2) PLC 发送数据的间隔应该大于该对象字设置时间的两倍，否则模块将一直发送写命令，回读命令将暂停，无法得到执行时间；

3) 在使用读取到的状态时，需要注意每个状态更新的时间 = 每个从站命令条数 \* 从站总数 \* 轮询时间 (0x8000:4) ,每个从站命令条数由对象字 8001~800A 设置生成。比如如下设置：

| 索引   | 子索引 | 名称                           | 设定值 | 索引   | 子索引 | 名称                           | 设定值 |
|------|-----|------------------------------|-----|------|-----|------------------------------|-----|
| 8001 | 1   | Slave addr                   | 2   | 8004 | 1   | Slave addr                   | 4   |
|      | 2   | coil is readable             | 1   |      | 2   | coil is readable             | 0   |
|      | 3   | hold reg readable            | 1   |      | 3   | hold reg readable            | 0   |
|      | 5   | coil start address           | 0   |      | 5   | coil start address           | 0   |
|      | 6   | the number of coil           | 64  |      | 6   | the number of coil           | 64  |
|      | 8   | Discrete input start addr    | 0   |      | 8   | Discrete input start addr    | 0   |
|      | 9   | the number of DI             | 64  |      | 9   | the number of DI             | 64  |
|      | B   | Input register start addr    | 0   |      | B   | Input register start addr    | 0   |
|      | C   | the number of Input register | 16  |      | C   | the number of Input register | 16  |
|      | E   | Hold register start addr     | 0   |      | E   | Hold register start addr     | 0   |

|          | F | the number of Hold register  | 16 |          | F | the number of Hold register  | 16 |
|----------|---|------------------------------|----|----------|---|------------------------------|----|
| 800<br>2 | 1 | Slave addr                   | 3  | 800<br>5 | 1 | Slave addr                   | 5  |
|          | 2 | coil is readable             | 1  |          | 2 | coil is readable             | 1  |
|          | 3 | hold reg readable            | 1  |          | 3 | hold reg readable            | 0  |
|          | 5 | coil start address           | 0  |          | 5 | coil start address           | 0  |
|          | 6 | the number of coil           | 16 |          | 6 | the number of coil           | 64 |
|          | 8 | Discrete input start addr    | 0  |          | 8 | Discrete input start addr    | 0  |
|          | 9 | the number of DI             | 16 |          | 9 | the number of DI             | 64 |
|          | B | Input register start addr    | 0  |          | B | Input register start addr    | 0  |
|          | C | the number of Input register | 0  |          | C | the number of Input register | 16 |
|          | E | Hold register start addr     | 0  |          | E | Hold register start addr     | 0  |
|          | F | the number of Hold register  | 0  |          | F | the number of Hold register  | 16 |
| 800<br>3 | 1 | Slave addr                   | 0  | 800<br>6 | 1 | Slave addr                   | 5  |
|          | 2 | coil is readable             | 1  |          | 2 | coil is readable             | 0  |
|          | 3 | hold reg readable            | 1  |          | 3 | hold reg readable            | 1  |
|          | 5 | coil start address           | 0  |          | 5 | coil start address           | 64 |
|          | 6 | the number of coil           | 16 |          | 6 | the number of coil           | 64 |
|          | 8 | Discrete input start addr    | 0  |          | 8 | Discrete input start addr    | 64 |
|          | 9 | the number of DI             | 16 |          | 9 | the number of DI             | 64 |
|          | B | Input register start addr    | 0  |          | B | Input register start addr    | 16 |
|          | C | the number of Input register | 0  |          | C | the number of Input register | 16 |
|          | E | Hold register start addr     | 0  |          | E | Hold register start addr     | 16 |
|          | F | the number of Hold register  | 0  |          | F | the number of Hold register  | 16 |

在上图示例中，一共设置 5 个从站，对象字 8003: 1 从站地址为空，即使设置从站属性也不计算。

从站 2 使用 64 个线圈输出、64 个离散输入、16 个保持寄存器、16 个输入寄存器，同时支持线圈回读和保持寄存器回读。这是本模块支持的最大数量，在模块运行时，会有 4 条 modbus 协议指令（读线圈、读离散输入、读输入寄存器、读保持寄存器），如果 PLC 编程有按周期写本模块的线圈和保持寄存器，一共有 6 条 modbus 协议指令（加入写线圈、写保持寄存器）。

从站 3 使用 16 个线圈输出、16 个离散输入，同时支持线圈回读和保持寄存器回读，但是保持寄存器数量为零，回读指令无效。在模块运行时，一共 2 条 modbus 协议指令（读线圈、读离散输入），如果 PLC 编程有按周期写本模块的线圈，一共有 3 条 modbus 协议指令（加入写线圈）。

从站 4 使用 64 个线圈输出、64 个离散输入、16 个保持寄存器、16 个输入寄存器，不支持线圈回读和保持寄存器回读。在模块运行时，会有 2 条 modbus 协议指令（读离散输入、读输入寄存器），

如果 PLC 编程有按周期写本模块的线圈和保持寄存器，一共有 4 条 modbus 协议指令（加入写线圈、写保持寄存器）。

从站 5 使用 128 个线圈输出、128 个离散输入、32 个保持寄存器、32 个输入寄存器，支持保持寄存器回读。共占用两个对象字（8005、8006）。在模块运行时，会有 6 条 modbus 协议指令（读离散输入、读输入寄存器、读保持寄存器），如果 PLC 编程有按周期写本模块的线圈和保持寄存器，一共有 10 条 modbus 协议指令（加入写线圈、写保持寄存器）。

综上本次一共使用 14 条读命令，9 条写命令，读命令按照对象字 8000：4 设定时间进行轮询，写命令由 PLC 编程随机插入，当 PLC 需要进行写命令时，本模块在执行完毕上一条读命令后，优先执行写命令。

此时，如果轮询时间（8000：4）设置为 50ms，即使没有任何的写指令，在读到某一个从站状态后，至少需要  $50\text{ms} \times 14 = 700\text{ms}$  才会再次更新这个状态，即假如读取从站 2 离散输入状态后，下一次更新此状态将在 700ms 后，再次期间模块不读此从站离散输入状态。所以🔔在实际使用时，使用 Modbus 功能时，尽量只周期读取需要的从站变量，减少轮询时间和命令条数。

修订历史

| 版本   | 修订日期       | 修订说明          | 维护人   |
|------|------------|---------------|-------|
| V1.0 | 2022.10.06 | 初始版本          | Zhang |
| V1.1 | 2023.03.08 | 修改错误描述        | Zhang |
| V1.2 | 2023.12.06 | 增加使用说明和连接设备使用 | WH    |

关于我们

企业名称：东莞市艾莫迅自动化科技有限公司

官方网站：[www.amsamotion.com](http://www.amsamotion.com)

技术服务：4001-522-518 拨 1

企业邮箱：[sale@amsamotion.com](mailto:sale@amsamotion.com)

公司地址：广东省东莞市南城区袁屋边艺展路 9 号兆炫智造园 B 栋 1 楼



官方公众号



官方抖音