



AU7 149 PNT22-CAN  
PROFINET Slave to CANopen  
Master 网关用户使用手册



## 目录

一、简介 .....	1
1.1 电气规格 .....	1
1.2 外形尺寸图 .....	2
1.3 模块接线图 .....	2
二、模块说明 .....	3
2.1 指示灯说明 .....	3
2.2 电源端口说明 .....	3
2.3 CAN 端口说明 .....	4
三、CANopen Tool 软件说明 .....	4
3.1 CANopen Tool 软件安装 .....	4
3.2 CANopen Tool 软件说明 .....	6
3.3 CANopen Tool 软件添加第三方 EDS 设备文件 .....	7
四、实际组态通讯案例说明 .....	8

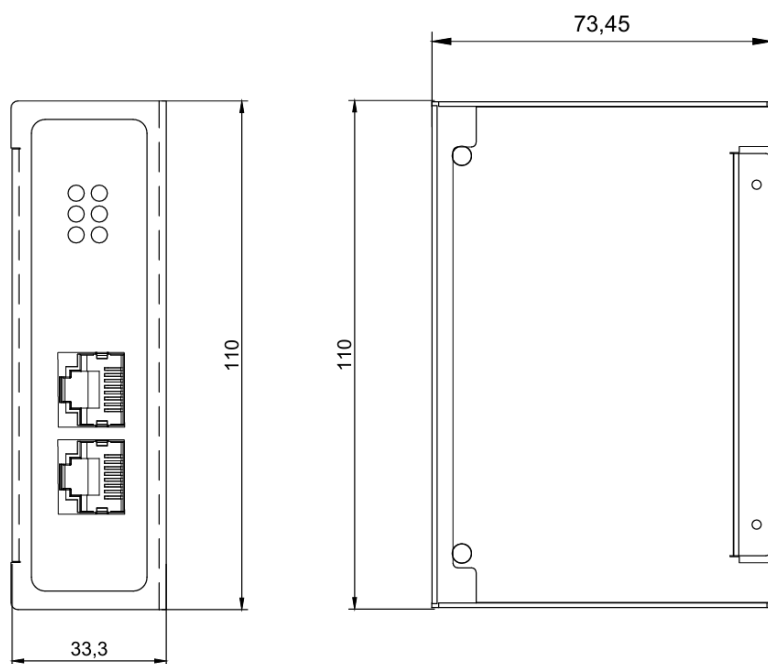
## 一、简介

本说明书描述了 AU7 149 PNT22-CAN PROFINET Slave to CANopen Master (以下简称 PN-CANopen) 网关模块的各项参数, 具体使用方法和注意事项, 为方便工程人员的操作使用。本产品实现 PROFINET 网络与 CANopen 网络之间的数据通讯, 可连接 CANopen 网络到 PROFINET 网络。即将 CANopen 设备连接到 PROFINET 网络。

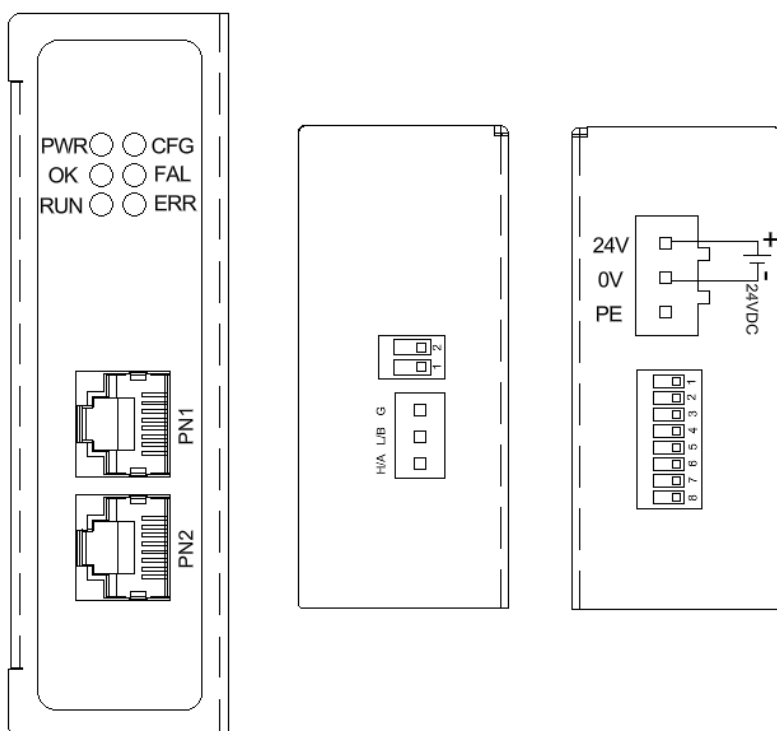
### 1.1 电气规格

工作电源	24VDC ( $\pm 10\%$ )
工作电流损耗	130mA@24V DC
通信接口	2 个 RJ45 网口, 1 个 CAN 接口
网口通信速率	100Mbit/s
支持最大数据	1440bytes IN+1440bytes OUT
协议转换	Profinet-RT 从站转 CANopen 主站
从站 ID/IP 设置	由编程软件配置或者主站分配
最大连接从站数	8 个 CAN 从站
CANopen 规范	支持 DS301 v4.0.2
CANopen 接口	3 口端子
CANopen 支持波特率	10kBits/s、20kBits/s、50kBits/s、100kBits/s、125kBits/s、250kBits/s、500kBits/s、800kBits/s、1Mbits/s
参数设置	软件配置
电源与总线隔离	有
保护	反向电压/短路保护
系统电源和通讯诊断	支持
防护等级	IP20
外壳材料	金属
工作环境	工作环境温度: $-20^{\circ}\text{C} \sim 70^{\circ}\text{C}$ ; 相对湿度: 10% ~ 95%(无凝露)
尺寸 (长×宽×高)	34×110×74mm
重量	大约 330g
安装	DIN 35mm 导轨安装

## 1.2 外形尺寸图



## 1.3 模块接线图



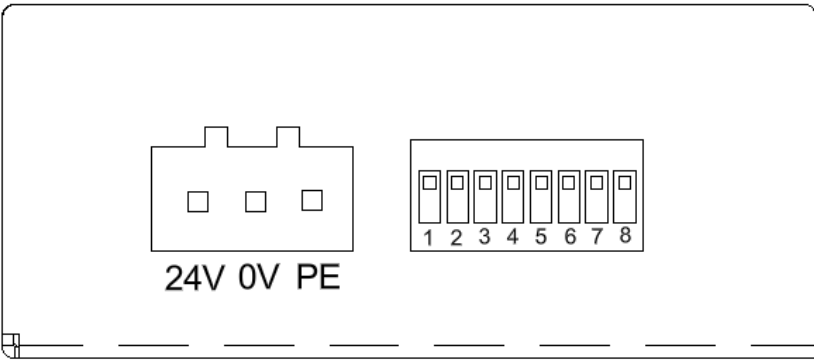


二、模块说明

2.1 指示灯说明

指示灯	状态	含义
PWR	绿灯亮	电源正常
	绿灯灭	电源故障
CFG	绿灯亮	下载过正确有效的配置工程
	绿灯灭	无配置工程
OK	绿灯亮	Profinet 通讯正常
	绿灯灭	Profinet 通讯断开
RUN	绿灯亮	节点处于运行状态
	绿灯周期性亮 200ms、灭 1000ms	节点处于停止状态
	绿灯周期性亮 200ms、灭 200ms	节点处于预运行状态
ERR	红灯灭	CANopen 网络正常
	红灯闪烁	至少一个 CANopen 从站离线

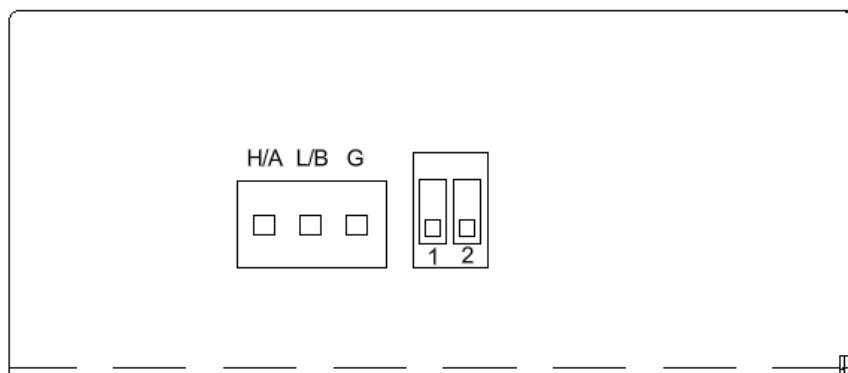
2.2 电源端口说明



引脚	功能
1	24V+, 直流 24V 电源正, 范围 9-30V
2	0V, 直流 24V 电源负
3	PE, 地

注：拨码开关暂时不用

## 2.3 CAN 端口说明



引脚	功能
1	CAN-H
2	CAN-L
3	GND,保护地

注：红色拨码开关用于设置终端匹配功能，当开关拨到“ON”时，启用终端匹配功能，当开关拨到“OFF”时，禁止终端匹配功能。终端匹配电阻为 120 欧姆。

## 三、CANopen Tool 软件说明

### 3.1 CANopen Tool 软件安装

在安装 CANopen Tool 软件时，推荐使用的计算机配置如表所示。

环境	类型	型号
硬件环境	显示器	彩色 CRT
	输入输出	标准键盘，鼠标
	USB 接口	至少一个 2.0 接口
	显卡	分辨率支持 1280×1024
	CPU	Intel Pentium 2.4GHz 以上
	内存	512M 以上
	硬盘	10G 以上
软件环境	操作系统	Windows7 以上
	应用软件	CANopen Tool

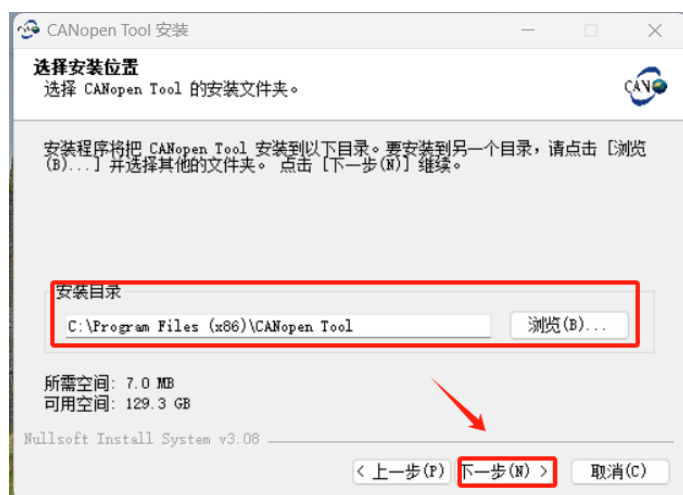
安装 CANopen Tool 软件的主要步骤如下所述。

#### 第 1 步 启动安装向导

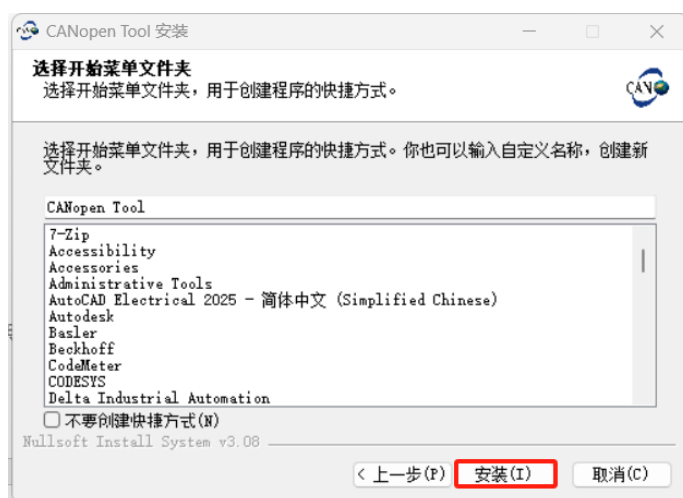
双击安装包，弹出框如下图所示，点击下一步：



弹出对话框如下图所示，选择安装位置，点击下一步，如下图所示：



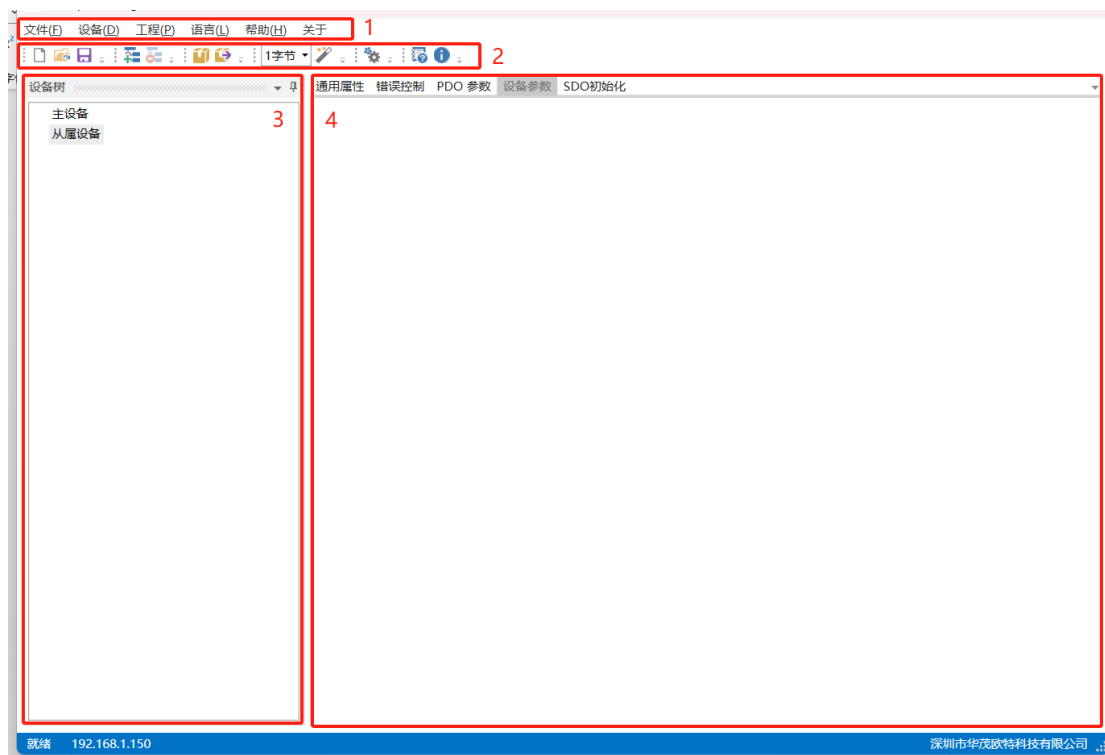
选择是否创建桌面快捷方式，点击安装，如下图所示：



安装完成，如下图所示：



### 3.2 CANopen Tool 软件说明



软件界面整体可分为个部分：菜单栏、工具栏、设备树，以及配置区。上图中的1表示的是菜单栏，2表示的是工具栏，3表示的是设备树，4表示的是配置区。

其中：

菜单栏：是提供软件所支持的工程文件操作，比如打开、保存；以及工程的下装与通讯设置等。

工具栏：用于快速访问菜单栏中的各个功能。



设备树：用于组态需要的硬件设备，一共分为两个区域，分别是主设备和从属设备，其中主设备只能添加一个，从属设可以添加多个。

配置区：用于显示详细配置主设备与从设备的信息，当选择了对应的设备时，才会在“配置区”显示其信息，该区域包含通用属性、错误控制、PDO 参数、设备参数、SDO 参数这 5 项选项卡在里面。其中：

通用属性：主要展示设备的概述与总体信息，比如设备名称、厂商信息等。

错误控制：用于配置设备是工作在心跳模式或者监听模式。

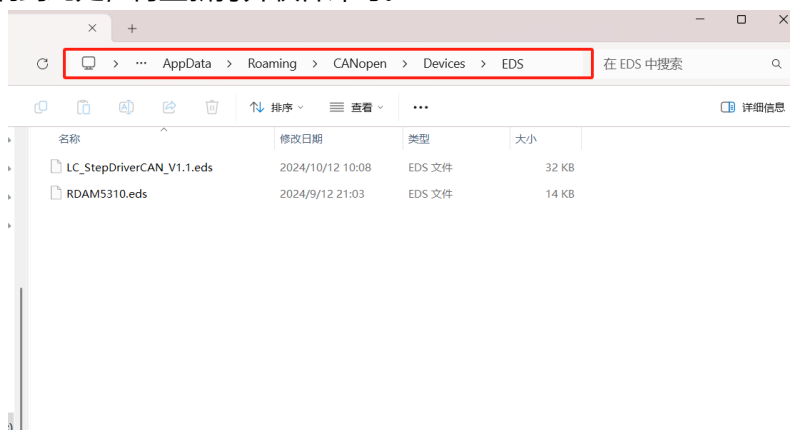
PDO 参数：提供对 RPDO 与 TPDO 对象命令配置。

设备参数：提供对设备的通信区域、制造商区域，以及标准化区域等内容的组态配置。

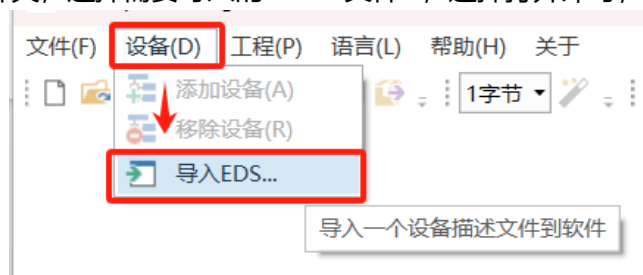
SDO 参数：提供对非周期性数据 SDO 参数的启动命令初始化控制。

### 3.3 CANopen Tool 软件添加第三方 EDS 设备文件

添加第三方的 EDS 设备文件一共有两种添加方式，第一种是通过修改安装目录下的目标文件夹添加第三方设备,例如：C:\Users\WELLAUTO\AppData\Roaming\CANopen\Devices\EDS) 如上所示目录下，存放所有的从属设备描述文件，将用户自定义的 EDS 文件直接复制到此处，再重新打开软件即可。

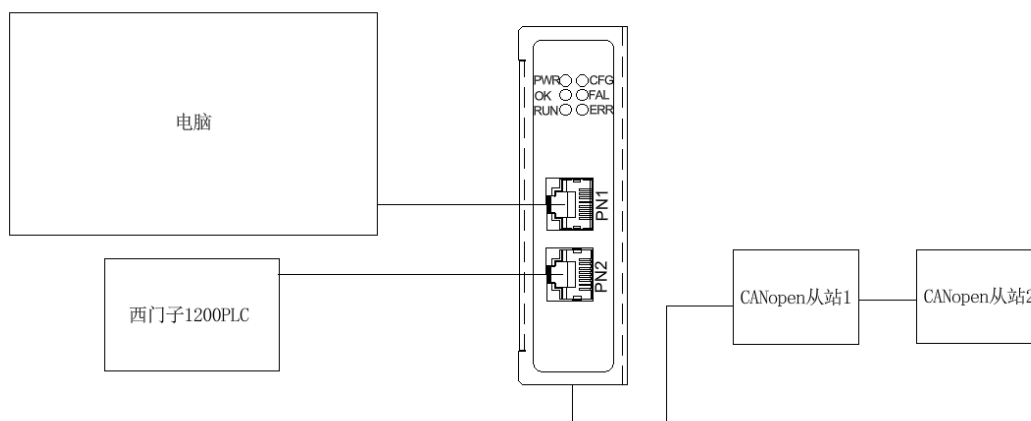


第二种是通过 CANopen Tool 软件导入第三方设备描述文件。打开 CANopen Tool 软件点击<菜单栏>，选择<设备>，点击<导入 EDS...>，选择存放第三方（用户自定义）的 EDS 文件的文件夹，选择需要导入的<EDS 文件>，选择打开即可，如下图所示

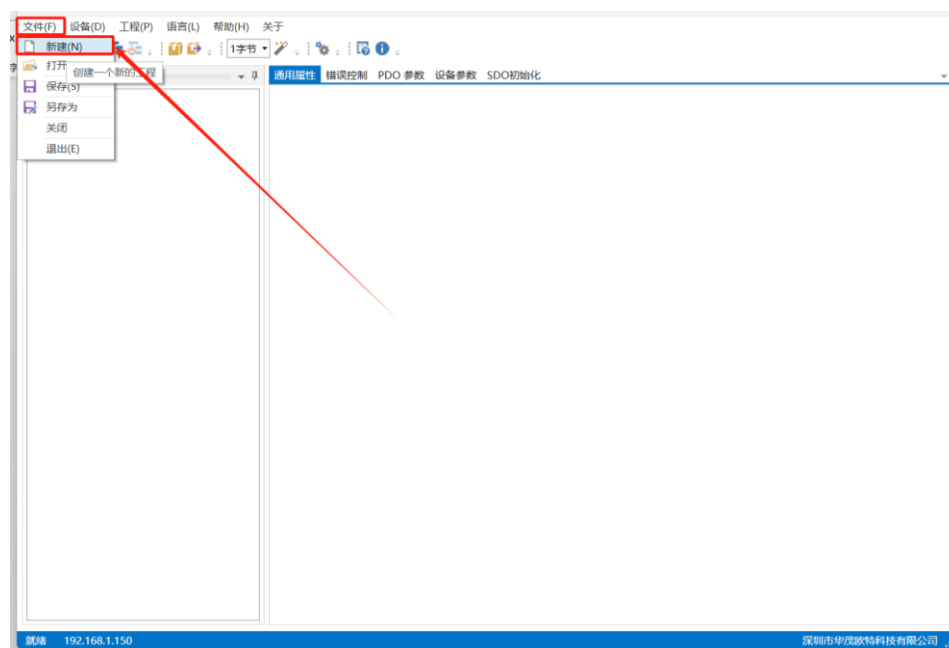



## 四、实际组态通讯案例说明

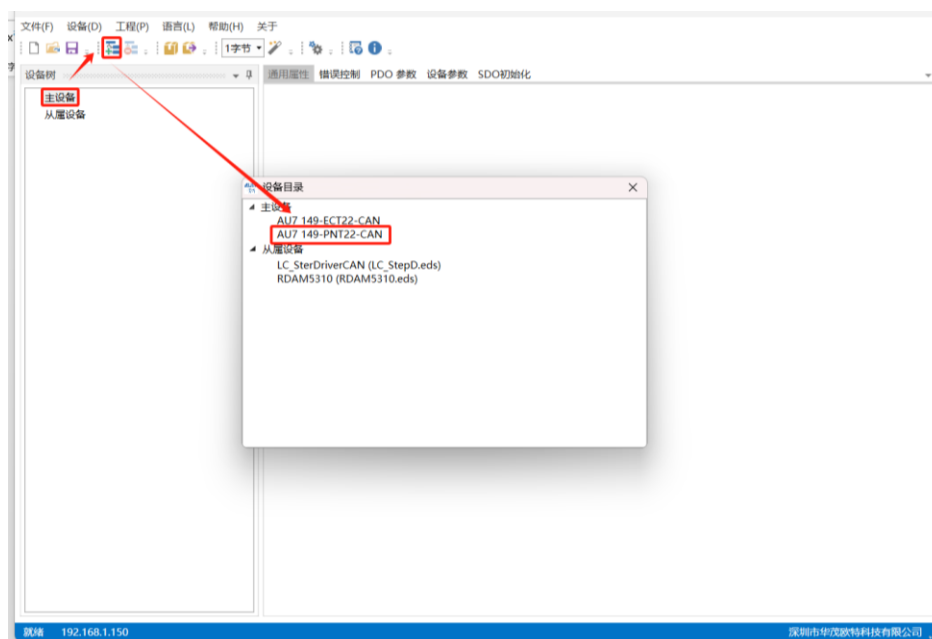
本案例是以西门子 1200PLC 作为 PROFINET 主站与 AU7 149-PNT22-CAN 网关通讯，AU7 149-PNT22-CAN 网关再做 CANopen 主站与 CANopen 驱动器和 CANopen IO 通讯，并控制 CANopen 驱动器动作。具体通讯连接示意图，如下图所示：




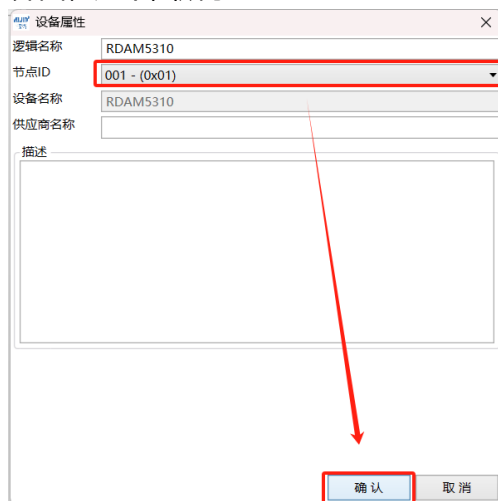
首先启动 CANopen Tool 软件，单击菜单栏“文件”--“新建”或者点击工具栏中“创建一个新的工程”，如下图所示：



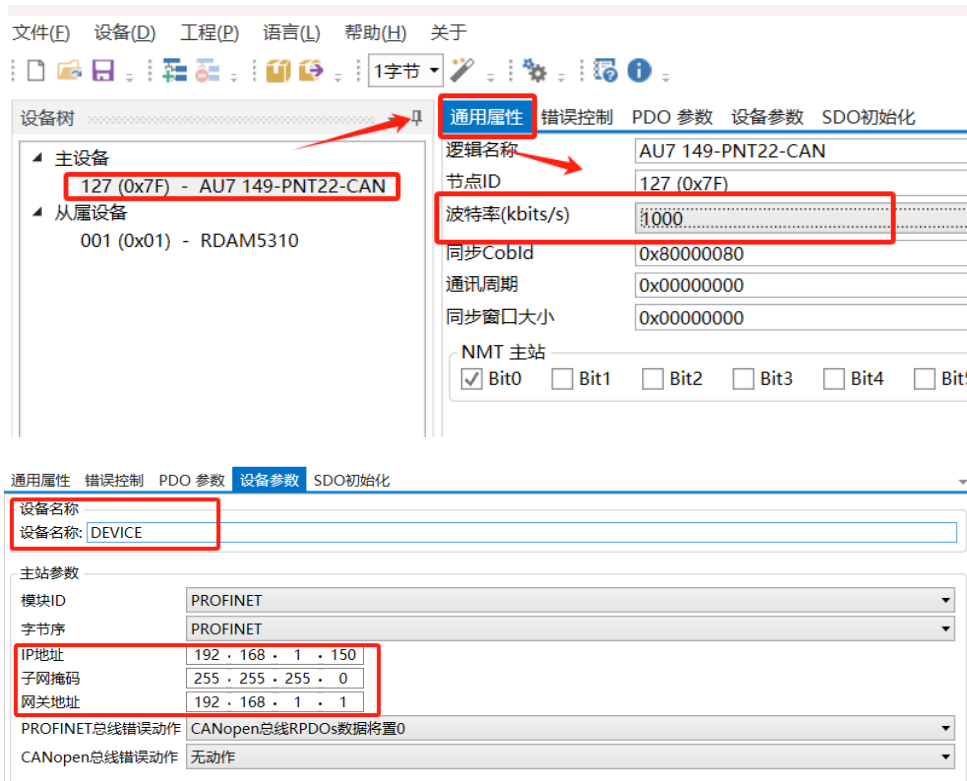
鼠标单击设备树中的“主设备”，选择点击 “”，弹出“添加设备”对话框，点击添加主设备，列表中选择“AU7 149-PNT22-CAN”（PROFINET 从站转 CANopen 主站网关）作为主设备，双击添加后返回主界面。配置如下图所示：



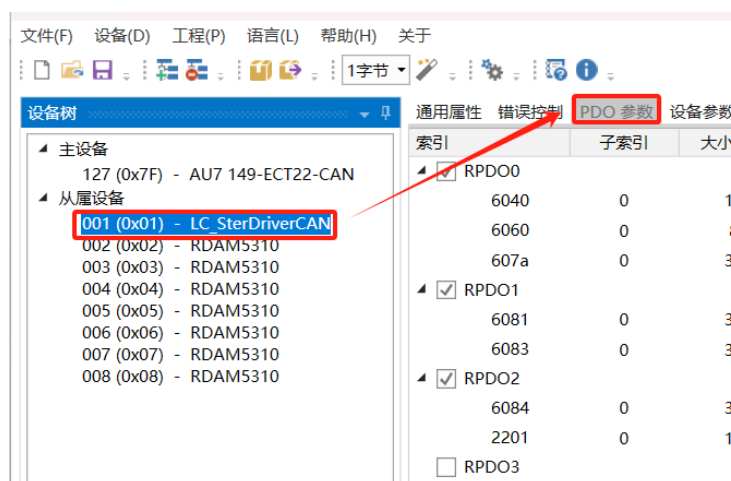
添加完主设备后，用同样的方法添加从属设备。“主设备”，点击 “” 弹出“添加设备”对话框，点击添加从属设备，添加完后会弹出设备属性弹窗，选择好站好后点击确定返回主界面，如下图所示：



添加完主设备以及从属设备后，点击“AU7 149-PNT22-CAN” ---通用属性---波特率，将波特率设置成与 Canopen 从站一致。配置完波特率后点击设备参数再修改设备名称以及所需要的 IP 地址（注：此处的设备名称和 IP 对应后期博图组态）如下图所示



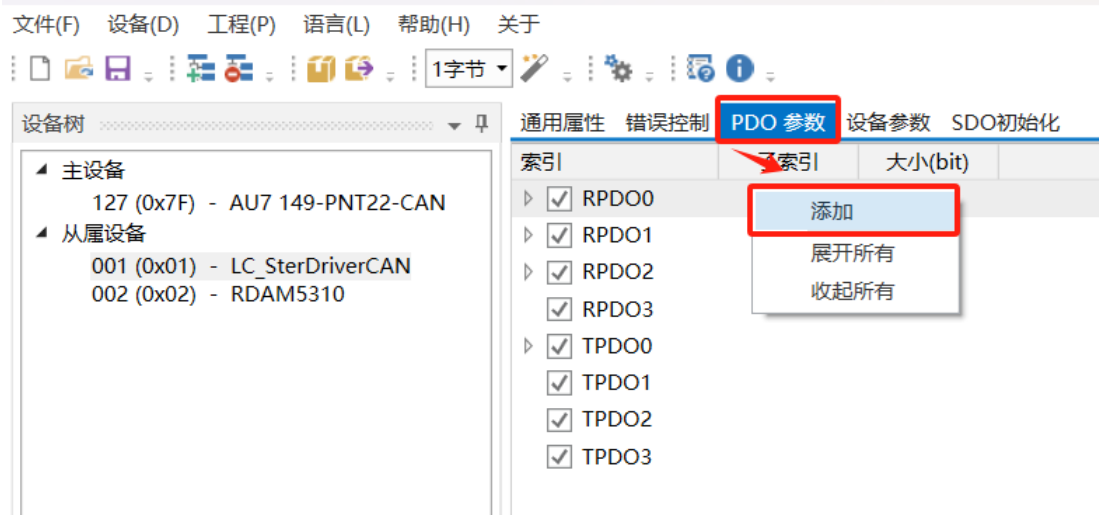
选择一个从属设备，在“配置区”切换到“PDO 参数”页面，如下图所示：



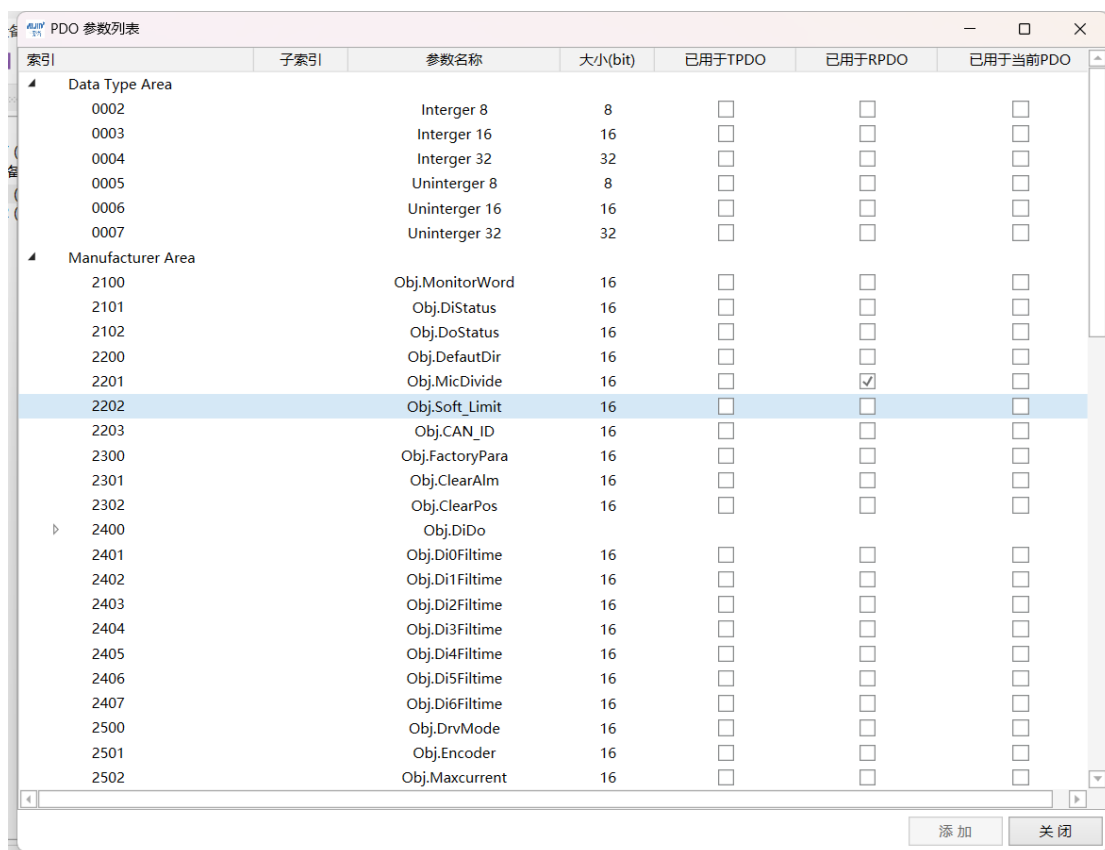
在该界面中，分为上下两部分，上面部分列出了设备所支持的 PDO 对象。根据设备的不同，会列出不同个数的 RPDO 与 TPDO。

当选择一个 PDO 对象时，会在下面显示出该 PDO 对象可配置的参数，包括 COB-ID、传输类型、抑制时间、事件计时器。

在 PDO 对象上右键，例如 RPDO0，选择“添加对象”，如下图所示：



在弹出的 PDO 参数列表对话框中，根据需要选择对象，点击“添加”按钮即可将其添加到所选择的 PDO 对象下面，如下图所示：



这里添加索引<6040>下的子索引对象<0>。然后单击“添加”按钮，回到主界面，同理在 TPDO0 添加索引<6041>下的子索引对象<0>，此时 PDO 添加完成，如下图所示：



其中的通道偏移量，用户可根据需要配置 PDO 后点击工具栏 “” 或者自定义偏移量即可。如果不需要子对象，选中单击右键，选择“移除”即可；同时支持子对象“上移”和“下移”操作，如下图所示：



在从属性设备 PDO 参数下面有 PDO 通讯参数设置窗口，每个 PDO 在对象字典中用到 2 个对象描述：

**PDO 映射参数：**包含一个对象字典中对象的列表，这些对象映射到 PDO 里，包括数据长度，生产者和消费者必须知道这个映射，用来解释 PDO 内容；

**PDO 通讯参数：**包含哪个 COB-ID 将被 PDO 使用，传输类型、传输速率、抑制时间和事件计时器；

**PDO 通信参数**

使能	<input checked="" type="checkbox"/>
COB-ID	0x0000201
传输类型	Synchronous (cyclic)
传输速率	0x01
抑制时间(100us)	0x0000
事件计时器(ms)	0x0000

**使能：**勾选代表启用该 PDO；

**COB-ID：**Communication Object Identifier ,CAN ID;

**传输类型：**

同步：通过接收 SYNC 对象实现同步。

非周期：由远程帧预触发传送，或者由设备子协议中规定的对象特定事件预触发

传送。

周期：传送在每 1 到 240 个 SYNC 消息触发。

异步：由远程帧触发传送 或 由设备子协议中规定的对象特定事件触发传送。

PDO 传输类型与 PDO 触发模式对应表如下所示：

传输类型	触发 PDO 的条件 B= both needed O=one or both			PDO 传输
	SYNC	RTR	Event	
Synchronous (acyclic)	B		B	同步，非周期
Synchronous (cyclic)	O			同步，循环周期
RTR-only (synchronous)	B	B		同步，在 RTR 之后
RTR-only (event-driven)		O		异步，在 RTR 之后
Event-driven (profile)		O	O	异步，设备子协议特定事件
Event-driven (manufacturer)		O	O	异步，制造商特定事件
SYNC：接收到 SYNC – object(同步对象) RTR：接收到远程帧 Event：数值改变或者定时器中断 B 代表两个触发条件均满足时触发 PDO 传输， O 代表一个或者两个触发条件满足时均可触发 PDO 传输				


传输速率：1-240，该数字代表两个 PDO 之间的 SYNC 对象的数目，

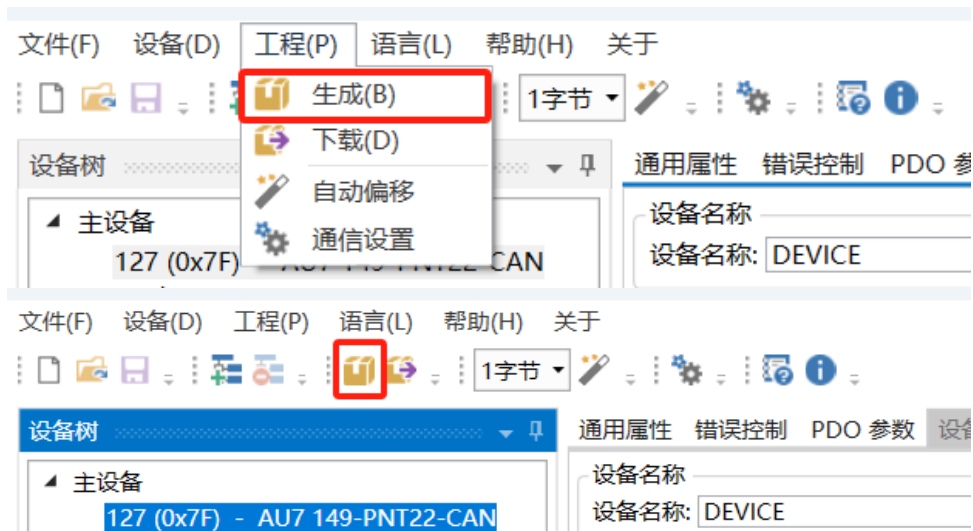
注意：该模式与主设备“通用属性”的通讯周期有关，如果启用同步周期模式，通讯周期时间大于 0；

抑制时间：一个 PDO 可以指定一个禁止时间，即定义两个连续 PDO 传输的最小间隔时间，避免由于高优先级信息的数量太大，始终占据总线，而使其它优先级较低的数据无力竞争总线的问题，单位 100 微秒；

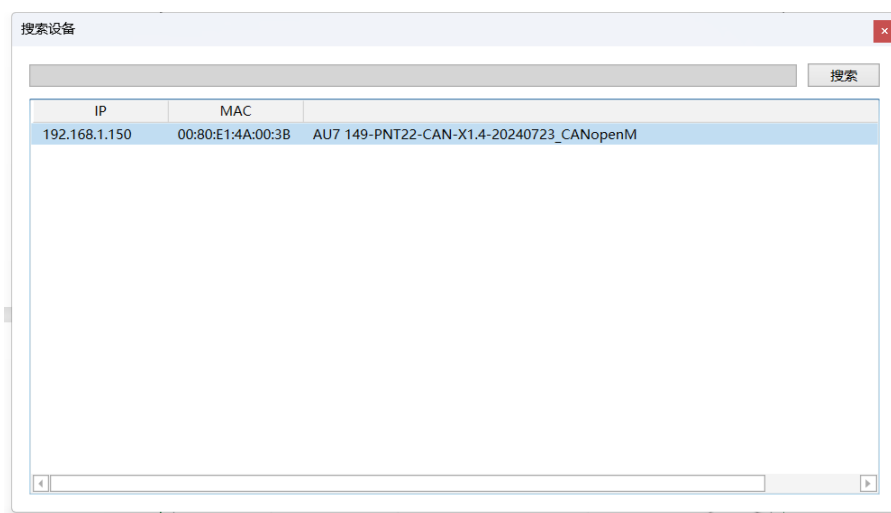
事件计时器：一个 PDO 可以指定一个事件定时周期，当超过定时时间后（PDO 发送最大时间间隔），一个 PDO 传输可以被触发，单位为 1ms；


不使能的 PDO 选项去掉勾选使能即可；

所有关于 canopen 从站的参数都配置好后点击 “” 生成，或者点击上方任务栏的“工程”再点击生成。如下图所示



生成完后将电脑用网线与 AU7 149-PNT22-CAN 上的任意一个网口连接起来，然后程序里点击“工程”再点击“通讯设置”，此处会弹出一个通讯设置的信息框，如果知道模块的 IP，在“TCP”处输入 IP 点确认即可（网关默认的 IP 为 192.168.0.8），如果不知道设备的 IP，在“TCP”的右侧点击“搜索”即可搜索模块并查询 IP，如下图所示

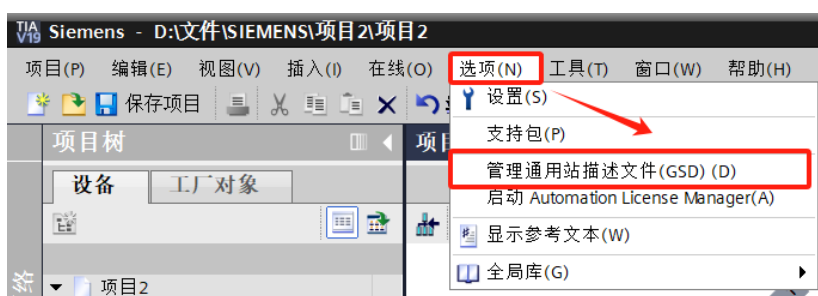


填写好 IP 后，点击上方任务栏的 “” 即可下载，或者点击“工程”再点击“下载”即可将工程下载至模块，如下图所示





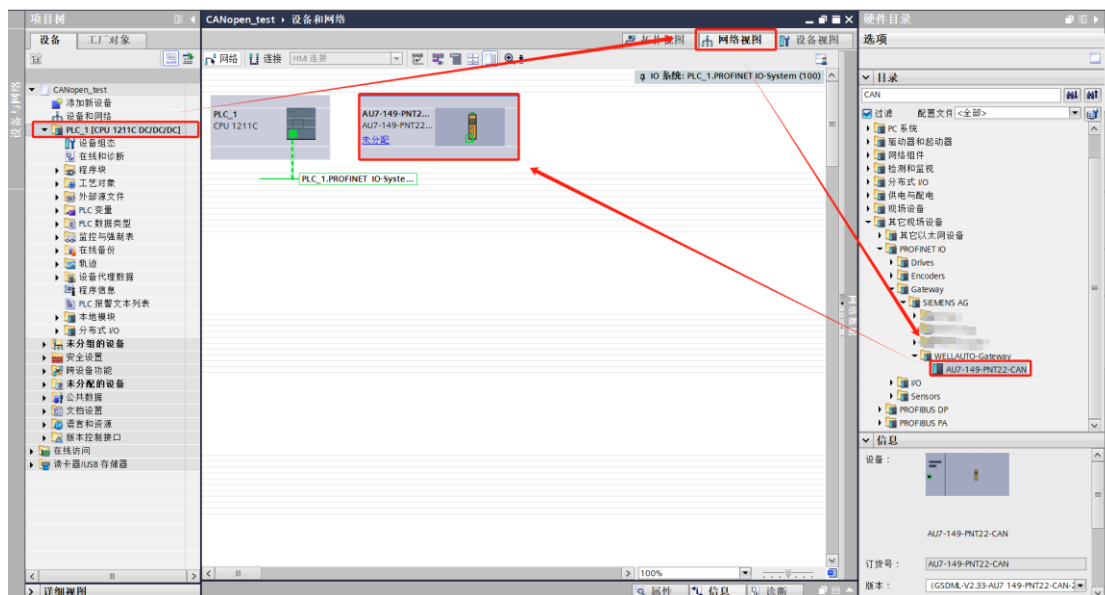
CANopen Tool 软件配置完后，打开博图新建工程，选择好对应的 PLC 型号，填写工程名称创建好新工程。创建好工程后，在上方任务栏点击“选项”---“管理通用站描述文件”导入 AU7 149-PNT22-CAN 的 GSD 文件。如下图所示



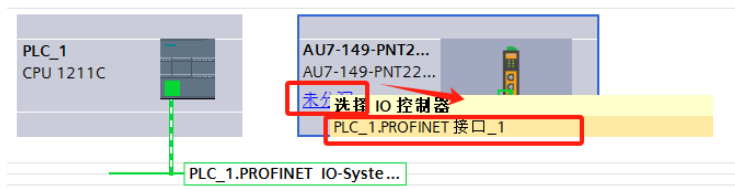
点击“管理通用站描述文件”完后会弹出一个窗口，在右侧点击“...”，选择 AU7 149-PNT22-CAN 的 GSD 文件，勾选并安装。如下图所示。



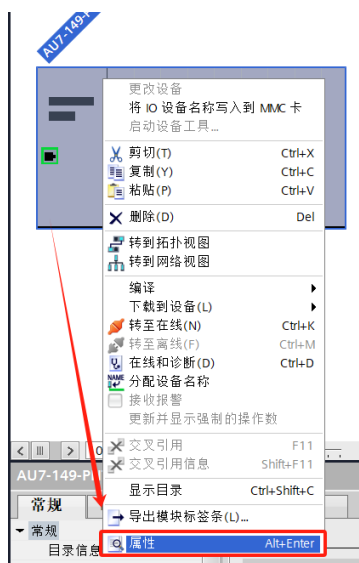
添加完 GSD 文件后，点击左侧任务栏的“设备组态”，再点击“网络试图”，然后在右侧硬件目录的任务栏里面找到“AU7 149-PNT22-CAN”，找到后将其拖入中间的网络视图。如下图所示



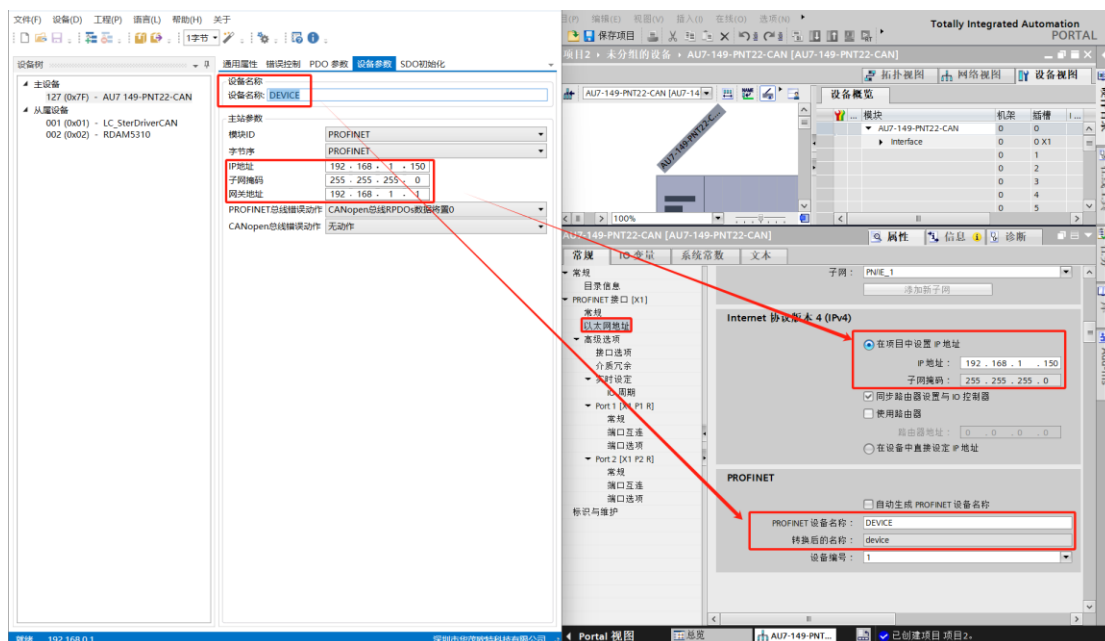
添加完模块后，点击“AU7 149-PNT22-CAN”上的“未分配”，将其分配给 PLC。  
如下图所示



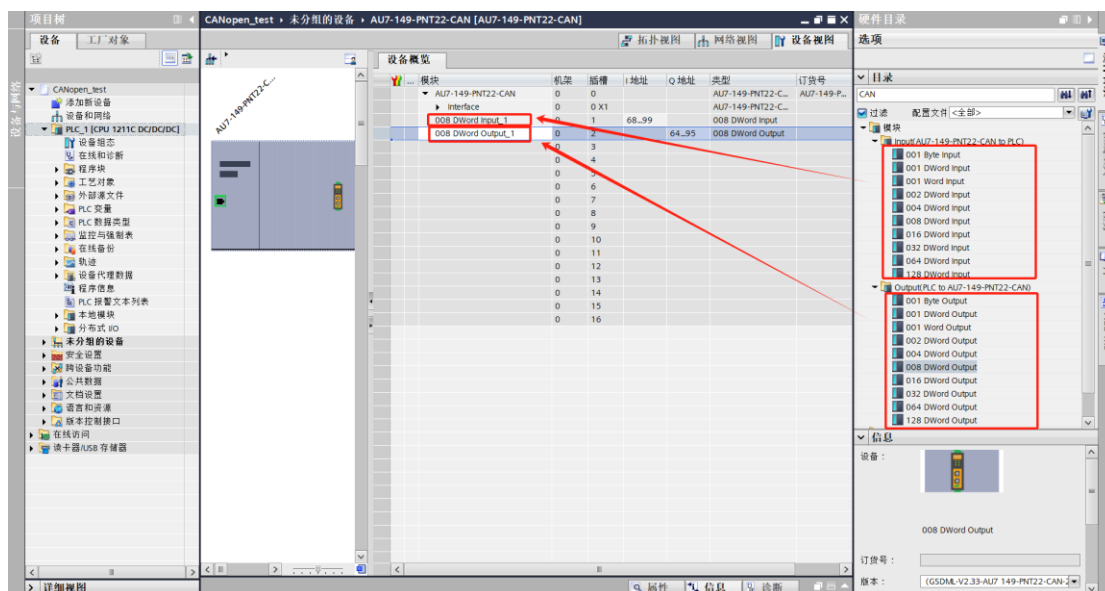
分配好 PLC 后，双击“AU7 149-PNT22-CAN”进入模块的内部组态页面，在模块处右键点击属性。如下图所示



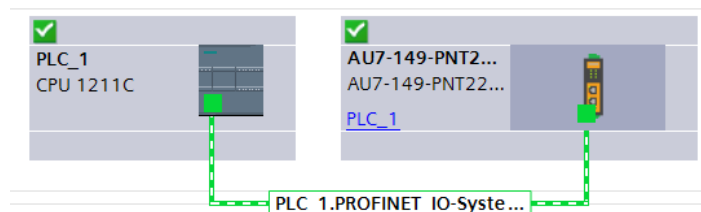
打开属性后点击“以太网地址”，此处的 IP 以及 PROFINET 设备名称需要与 CANopen Manager 软件主设备里面的“设备参数一致”否则与 AU7 149-PNT22-CAN 网关通讯不上，如下图所示



设置好 IP 以及 PROFINET 设备名称后，点击“设备试图”根据 CANopen Manager 软件里面配置的 can 从站 PDO 所需的数据大小进行配置，如下图所示



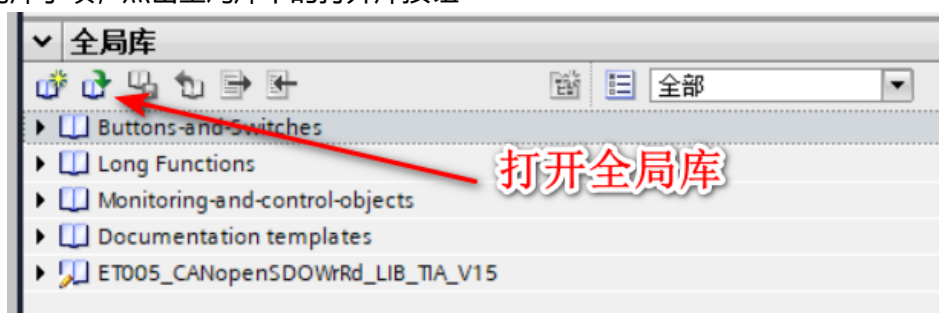
配置完后，将工程下载至 PLC 并将 PLC 与 AU7 149-PNT22-CAN 用网线连起来，在线监控显示无错误即可，如下图所示



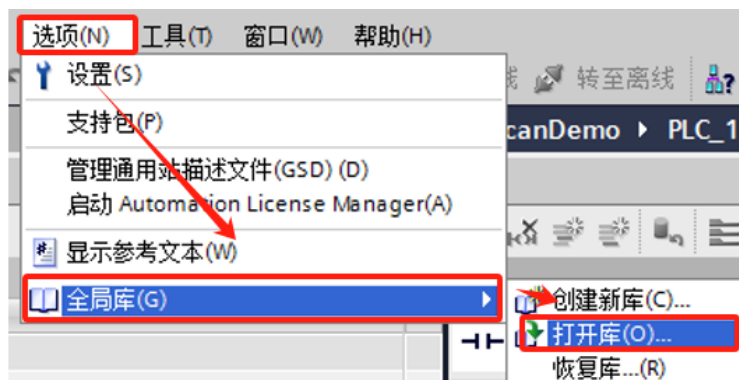
注：博图地址对应 CANopen Manager 软件里面 PDO 参数的偏移量，例如下图的 PDO 参数里面的“StatusWord”对应到博图就是 IW68 地址，后面的则往后偏移。如下图所示

The screenshot shows the TIA Portal software interface. On the left, the 'PDO Manager' window displays a list of PDO parameters for the device 'AU7-149-PNT22-CAN'. The parameters are organized into three sections: 01 (0x01) - LC\_SterDriverCAN, 02 (0x02) - RDAMS310, and 03 (0x03) - RDAMS310. The 'StatusWord' parameter is highlighted in red, showing its address as 0x01 and its size as 16 bits. On the right, the 'Device View' window shows the mapping of these parameters to TIA Portal addresses. The 'StatusWord' parameter is mapped to address 0x01, which corresponds to the IW68 address in the TIA Portal.

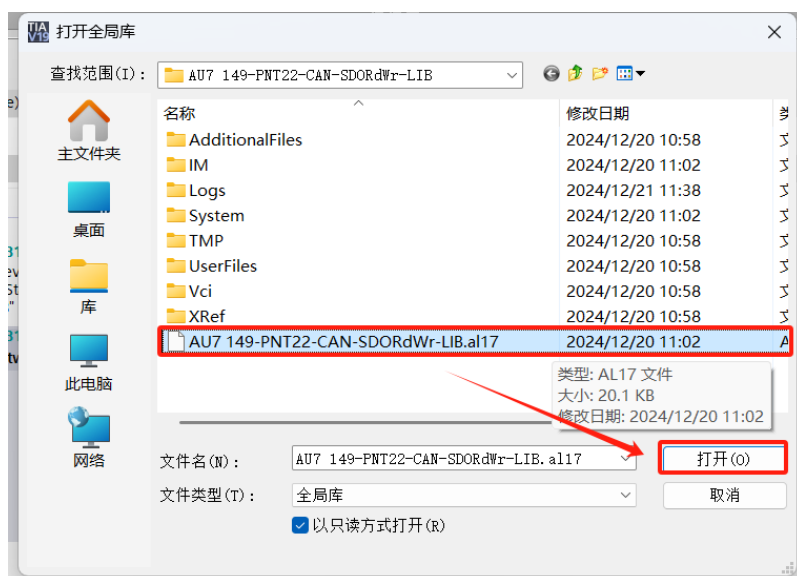
此外我们还做读取状态的功能块，打开博途软件选择相应的CPU模块后，在右侧栏选择全局库子项，点击全局库下的打开库按钮



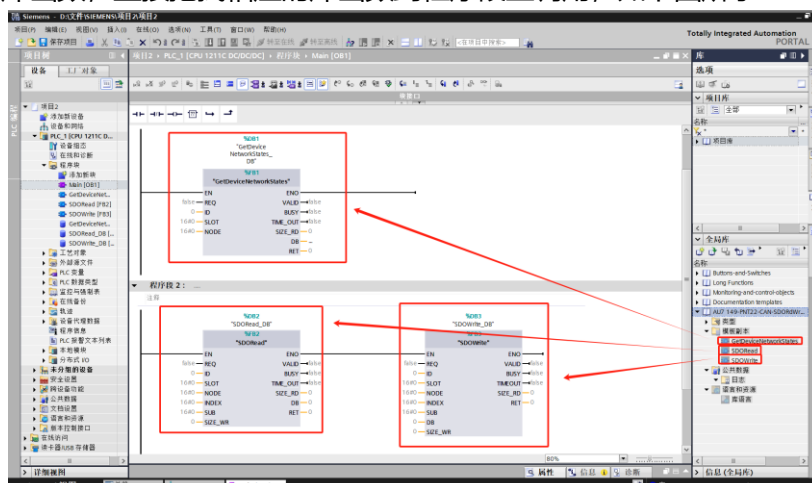
(或者在标题栏 - 选项 - 全局库 - 打开库)



弹出打开全局库对话框，选择AU7 149-PNT22-CAN-SDORdWr-LIB.al17路径下文件，点击打开。



“全局库”出现AU7 149-PNT22-CAN-SDORdWr-LIB\_VXX库文件，在主模板下出现新添加的库函数，直接拖拽相应的库函数到程序段里调用，如下图所示



拖拽至相应的程序段后，按照引脚定义创建变量并连接即可，对应的针脚类型请看如下

## SDO读写功能块

输入参数:

引脚名	数据类型	描述
REQ	BOOL	启动请求, REQ = TRUE 时, 执行功能块
ID	HW_IO	模块硬件地址
SLOT	BYTE	
NODE	BYTE	CANopen 站地址
INDEX	WORD	操作字典索引
SUB	BYTE	操作字典子索引
DB	DINT	数据块的序号, 仅 SDO_WR 有效, SDO_RD 作为数据输出端
SIZE_WR	UINT	操作数据长度

输出参数:

引脚名	数据类型	描述
VALID	BOOL	功能块操作完成且有效
BUSY	BOOL	直到操作结束, BUSY 一直为 TRUE
TIME_OUT	BOOL	功能块超时
SIZE_RD	UINT	读到的数据长度
DB	DINT	读到的数据, 仅 SDO_RD 有效, SDO_WR 无此项
RET	UDINT	Error Code, BUSY 为 FALSE 的时候可以 获得, 直到下一个 REQ 变为 TRUE

## GetDeviceNetworkStates库函数引脚说明

获取CAN网络状态

输入参数:

引脚名	数据类型	描述
REQ	BOOL	启动请求, REQ = TRUE 时, 执行功能块
ID	HW_IO	模块硬件地址
SLOT	BYTE	
NODE	BYTE	CANOpen 站地址
SIZE_WR	UINT	操作数据长度

输出引脚:

引脚	数据类型	描述
----	------	----

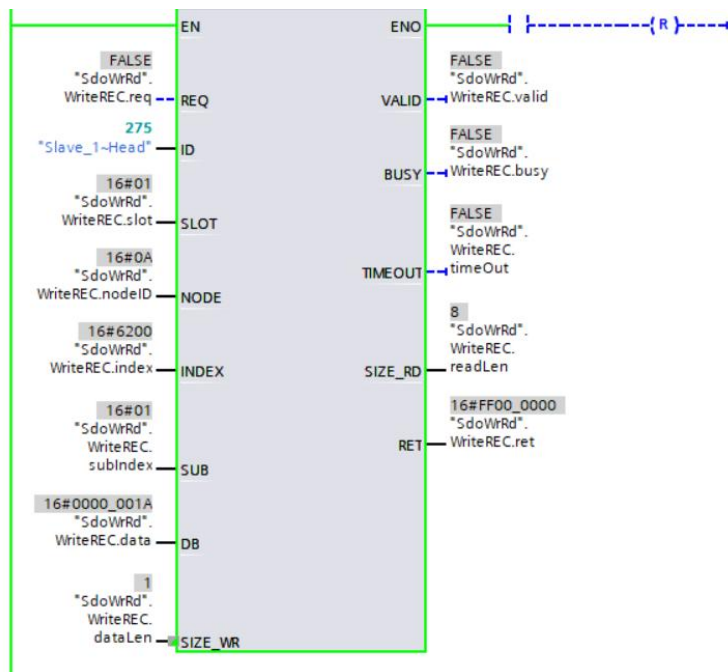
VALID	BOOL	功能块操作完成且有效
BUSY	BOOL	直到操作结束，BUSY 一直为 TRUE
TIME_OUT	BOOL	功能块超时
SIZE_RD	UINT	读到的数据长度
DB	ARRAY [0...15] OF BYTE	当输入参数的站地址为 0x7F 的时候，回复 16 字节，当为 0-0x7E 时回复 2 字节
RET	UDINT	Error Code，BUSY 为 FALSE 的时候可以获取，直到下一个 REQ 变为 TRUE

写SDO操作：

在全局库中找到SDOWrite函数，按住鼠标左键拖拽至程序段中。新建一个DB块，填写需要的参数并运用在SDOWrite功能块的针脚上。如下图所示

SdoWrRd			
	名称	数据类型	起始值
1	Static		
2	WriteREC	Struct	
3	req	Bool	false
4	slot	Byte	16#01
5	nodeID	Byte	16#0A
6	index	Word	16#6200
7	subIndex	Byte	16#01
8	data	Dint	16#1A
9	dataLen	UDInt	1
10	valid	Bool	false
11	busy	Bool	false
12	timeOut	Bool	false
13	readLen	UInt	0
14	ret	UDInt	0

当功能块的REQ针脚 = TRUE，SDORead功能块触发执行SDO读操作，读操作成功后程序复位REQ使能端。



其中:

ID为PROFINET/PROFIBUS DP从站的硬件标识符;

SLOT:暂时缺省;

NODE = 16#0A, 写入CANopen网络里的ID为10号从站设备;

INDEX = 16#6200, 写入10号从站索引地址为16#6200参数;

SUB = 16#01, 写入10号从站子索引地址为16#01参数;

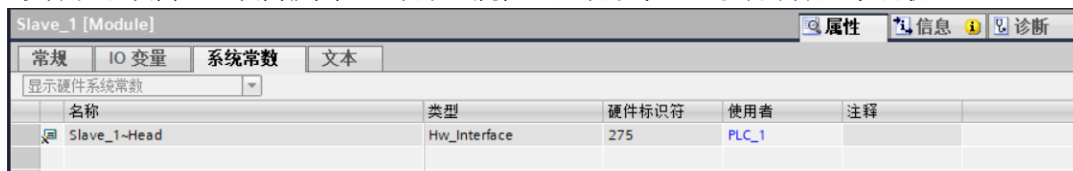
DB = 16#0000\_001A, 写入数据值, 该数据大小与SIZE\_WR参数关联 (0x12345678, 其中0x78为低地址值, 0x12为高地址值)

SIZE\_WR = 16#01, 写入的DB数据为1字节, 例如16#0A; 同理写入4字节则SIZE\_WR = 4;

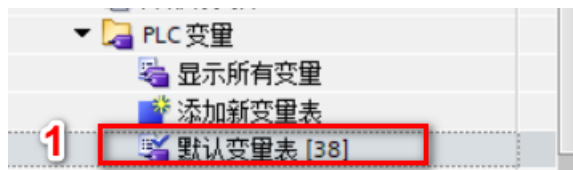
SIZE\_RD=8 为SDO\_WR操作完成后正确操作;

RET值参考 "RET值"

ID参数在该设备的 "设备视图" 选项下 "属性" 选项卡中的 "系统常数" 中获取



或者在默认变量表中 "系统常量" 列表里, 如下图





项目2 ▸ PLC\_1 [CPU 1215C AC/DC/Rly] ▸ PLC 变量 ▸ 默认变量表 [38]

变量 用户常量2 系统常量

**默认变量表**

	名称	数据类型	值	注释
25	Local-Pulse_1	Hw_Pwm	265	
26	Local-Pulse_2	Hw_Pwm	266	
27	Local-Pulse_3	Hw_Pwm	267	
28	Local-Pulse_4	Hw_Pwm	268	
29	OB_Main	OB_PCYLE	1	
30	Local-CM_1243-5~DP_接口	Hw_Interface	269	
31	Local-CM_1243-5	Hw_SubModule	271	
32	Local-DP-Mastersystem	Hw_IoSystem	272	
33	Slave_1~Head	Hw_Interface	275	
34	Slave_1~DPSlave	Hw_DpSlave	273	
35	Slave_1~Input_64_bytes_1	Hw_SubModule	276	
36	Slave_1~Output_64_bytes_1	Hw_SubModule	277	

执行上述操作后，CANopen网络通讯数据报文如下：

1314	接收	10:40:54.065	0x0000048A	数据帧	标准帧	0x08	00 00 00 00 00 00 00 00
1315	接收	10:40:54.111	0x0000060A	数据帧	标准帧	0x08	2F 00 62 01 1A 00 00 00
1316	接收	10:40:54.113	0x0000058A	数据帧	标准帧	0x08	60 00 62 01 00 00 00 00
1317	接收	10:40:54.127	0x00000080	数据帧	标准帧	0x00	

如果WRREC侧的网络为PROFIBUS，数据报文如下：

时间戳	源地址	目标地址	数据类型	值	注释
392088602.8s	SD2	2 → 3	SRD_HGHI	Data Exchange	Req
392088947.8s	SD2	2 → 3	OK	Data Exchange	Res
392090263.8s	SD2	2 → 3	SRD_LDW	DPV1_Write_Req	Req
392090497.8s	ACK		SRD_ACK	Short Acknowledged	
392090546.8s	SD1	2 → 43	FDL_Status	Req FDL_Status	Req
392090521.8s	SD4	2 → 2	Token Pass	Pass Token	
392090592.8s	SD2	2 → 3	SRD_HGHI	Data Exchange	Req
392091807.8s	SD2	2 → 3	OK	Data Exchange	Res
392092553.8s	SD2	2 → 3	SRD_LDW	DPV1_Pull	Req
392092788.8s	SD2	2 → 3	OK	DPV1_Write_Res	Res
392092991.8s	SD1	2 → 44	FDL_Status	Req FDL_Status	Req
392093366.8s	SD4	2 → 2	Token Pass	Pass Token	
392093437.8s	SD2	2 → 3	SRD_HGHI	Data Exchange	Req
392094253.8s	SD2	2 → 3	OK	Data Exchange	Res
392095093.8s	SD1	2 → 45	FDL_Status	Req FDL_Status	Req

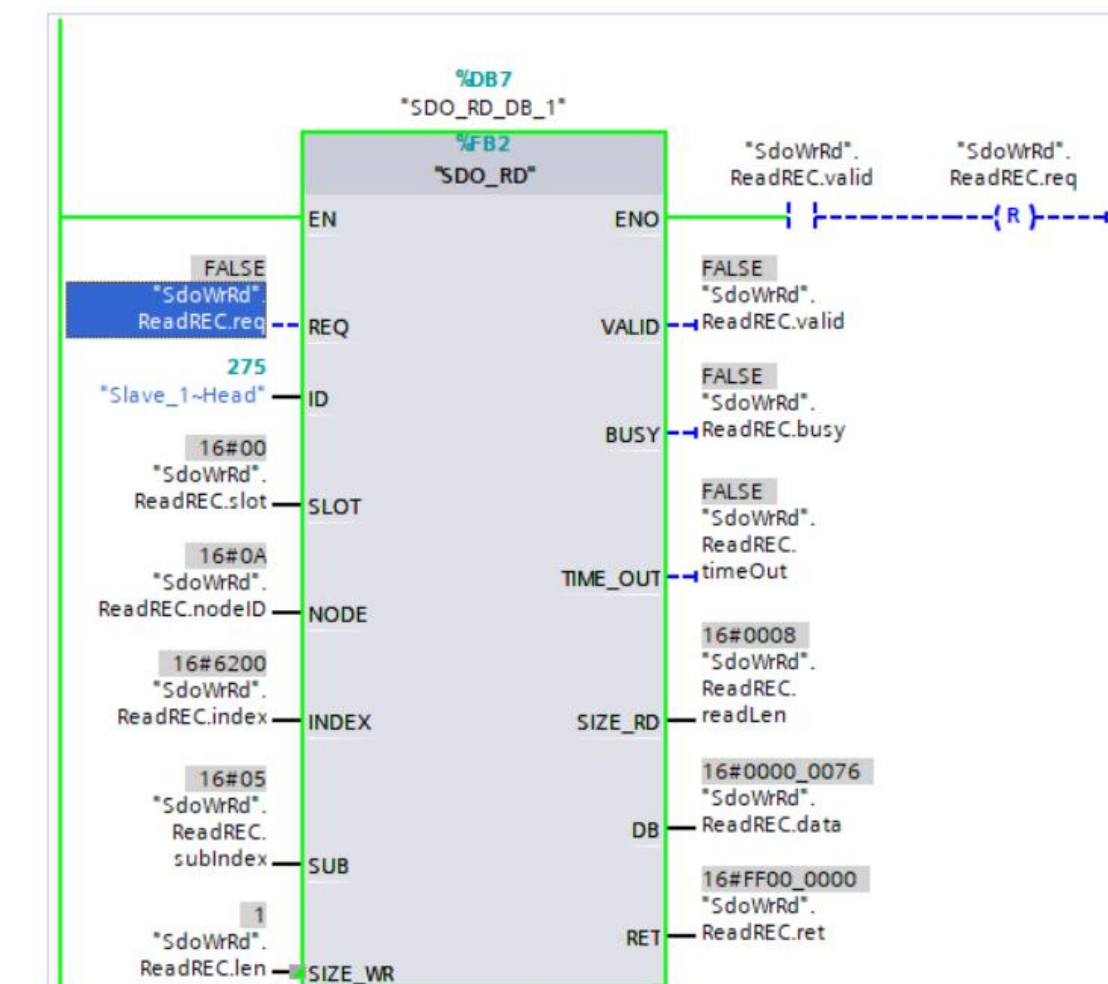
读 SDO 操作：

在全局库中找到SDORead函数，按住鼠标左键拖拽至程序段中。新建一个DB块，填写需要的参数并运用在SDORead功能块的针脚上。如下图所示

**SdoWrRd**

名称	数据类型	起始值
Static		
WriteREC	Struct	
ReadREC	Struct	
req	Bool	false
slot	Byte	16#0
nodeID	Byte	16#0A
index	Word	16#6200
subIndex	Byte	16#05
len	UDInt	16#01
valid	Bool	false
busy	Bool	false
timeOut	Bool	false
readLen	UInt	0
data	DInt	0
ret	UDInt	0

当功能块的REQ针脚= TRUE，SDORead功能块触发执行SDO读操作，读操作成功后程序复位REQ使能端。



SUB = 5，读取索引 16#6200，子索引 = 05 的参数值；

SIZE\_WR = 1，读取长度为 1 字节；

DB 为读取索引 16#6200，子索引 = 05 返回的数据值；

执行上述操作后，CANopen网络通讯数据报文如下：

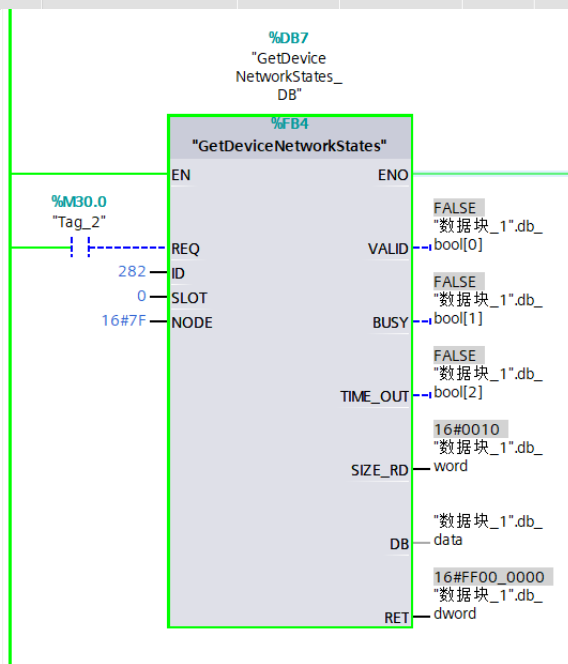
940	接收	11:25:09.424	0x0000050A	数据帧	标准帧	0x08	00 00 00 00 00 00 00 00
941	接收	11:25:09.428	0x0000060A	数据帧	标准帧	0x08	40 00 62 05 00 00 00 00
942	接收	11:25:09.428	0x0000058A	数据帧	标准帧	0x08	4F 00 62 05 76 00 00 00
943	接收	11:25:09.487	0x00000080	数据帧	标准帧	0x00	
944	接收	11:25:09.487	0x0000020A	数据帧	标准帧	0x08	00 00 00 00 76 77 00 00
945	接收	11:25:09.487	0x0000018A	数据帧	标准帧	0x08	00 00 00 00 00 00 00 00

如果WRREC侧的网络为PROFIBUS，数据报文如下：

[illegible]

在全局库中找到GetDeviceNetworkStates函数，按住鼠标左键拖拽至程序段中。新建一个DB块，填写需要的参数并运用在GetDeviceNetworkStates功能块的针脚上。如下图所示

数据块_1										
	名称	数据类型	起始值	监视值	保持	从 HMI/OP...	从 H...	在 HMI ...	设定值	
1	Static				<input type="checkbox"/>				<input type="checkbox"/>	
2	db_data	Array[0..15] of Byte			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	db_data[0]	Byte	16#0	16#02	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4	db_data[1]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
5	db_data[2]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
6	db_data[3]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
7	db_data[4]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
8	db_data[5]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
9	db_data[6]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
10	db_data[7]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
11	db_data[8]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
12	db_data[9]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
13	db_data[10]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	db_data[11]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
15	db_data[12]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
16	db_data[13]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
17	db_data[14]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
18	db_data[15]	Byte	16#0	16#00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
19	db_bool	Array[0..5] of Bool			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
20	db_bool[0]	Bool	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
21	db_bool[1]	Bool	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
22	db_bool[2]	Bool	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
23	db_bool[3]	Bool	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
24	db_bool[4]	Bool	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
25	db_bool[5]	Bool	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
26	db_dword	DWord	16#0	16#FF00_0000	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
27	db_word	Word	16#0	16#0010	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	



对DB引脚描述如下:

Node参数	0x7F	0x00-0x7E
描述	<p>代表获取整个CAN网络的从站是否在线, 回复的SIZE_RD=16#0010(16字节)</p> <p>每一个bit代表一个从站的状态; 其中第一个字节的bit0 – bit7代表站地址0-7的设备;</p> <p>例如Data.NET[0].DATA[0] = 16#FE(2#1111_1110), 表示站地址1-站地址7的设备在线;</p> <p>Data.NET[0].DATA[1] = 16#01(2#0000_0001), 其中bit8 = 1表示站地址8的设备在线, bit9-bit15均是0, 表示站地址9-站地址15的设备不在线; 其它依此类推总共表示0-127个从站设备。</p>	<p>获取站地址的详细状态, 回复的SIZE_RD=16#0002(2字节)</p> <p>其中:</p> <p>0x0000: Initial状态,</p> <p>0x0001: Disconnect状态,</p> <p>0x0004: Stop状态</p> <p>0x0005: Operational状态</p> <p>0x000F: Pre Operational状态</p> <p>其他值保留。</p>

RET状态值描述

RET 值	描述
0xFF00 0000	操作成功
0xFF00 0001	PROFINET 超时
0xFF00 0002	Slot 错误
0xFF00 0003	操作数据超长
0xFF00 0004	参数错误