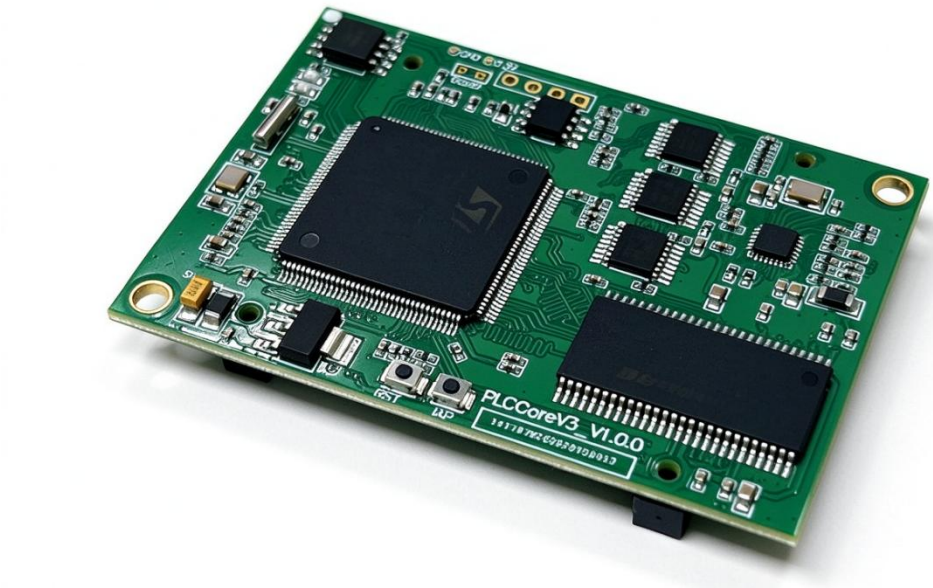
受　控

# GCAN-PLCCore-V3

PLC核心模块及开发套件

## 用户手册




文档版本：V1.00（2026 / 10 / 19）

修订历史

| 版本 | 日期 | 原因 |
| --- | --- | --- |
| V1.00 | 2026/10/19 | 创建文档 |

# 目　录

1.功能简介 ..... 1
1.1 功能概述 ..... 1
1.2 PLC 核心板资源 ..... 1
1.3 典型应用 ..... 1
2.GCAN-PLCCore-V3 引脚定义及功能说明 ..... 2
3.GCAN-PLCCore-V3 启动策略 ..... 5
4.GCAN-PLCCore-V3 核心板参数表 ..... 6
5.PLCCore-EDV-V3 开发套件 ..... 7
6.PLCCore-EDV-V3 开发套件参数 ..... 8
7.Codesys 软件使用 ..... 9
7.1 Codesys下载 ..... 9
7.2 Codesys 软件安装 ..... 9
7.3 界面介绍与语言切换 ..... 11
7.4 添加设备描述文件 ..... 12
7.5 安装库文件 ..... 13
7.6 新建工程 ..... 14
8.设备连接 ..... 16
8.1 更改电脑 IP ..... 16
8.2 Codesys 软件配置 ..... 17
8.3 下载程序 ..... 18
9. PLCCore-EDV-V3 开发套件基本例程 ..... 19
9.1 CAN数据收发实验例程 ..... 19
9.2 下载程序及重点参数 ..... 21
附录 A：Modbus 协议简介 ..... 26
10.免责声明 ..... 38
销售与服务 ..... 39

# 1.功能简介

## 1.1 功能概述

GCAN-PLCCore-V3是一种嵌入式 PLC 核心模块。该模块已经集成了 PLC控制内核，支持用户使用符合 IEC61131-3 标准的五种编程语言对其编程，包括：CFC、SFC、LD、ST、IL 并支持在线调试功能包括观看和设置变量、单周期、断点和单步执行等功能。

GCAN-PLCCore-V3 非常适合过程信号处理、集中监控、智能网络节点、分布式控制系统使用，还可以作为大系统中的特殊组件，也可以用作运动控制器的核心 PLC。

GCAN-PLCCore-V3核心模块已经包含整个 PLC 的运行环境和一部分存储外设，并且其还将芯片中的各种资源引出，提供多达 4 路 UART，3 路 CAN，1路 100M 以太网，12 路 AD，2 路 DA，52 个数字输入/输出，2 个高速计数器输入和 4路 PWM 输出。用户可以利用以上资源开发特定的程序进行采集与控制。

该模块还可按需定制。

## 1.2 PLC 核心板资源

- CPU：ARM® 32-bit Cortex®-M7 内核，处理器频率 400 MHz；
- 4 路UART、3 路CAN、1 路100M 以太网；
- 12 路AD、2 路DA；
- 52 路数字输入/输出（复用）；
- 3 路高速计数器输入（30KHz max）、 4 路PWM 输出、1组AB相计数输入；
- 一个40pin 插座、一个44pin 插座（在底部），方便接入各种底板；
- 1 个3.3V 测试点；
- 1 个电源指示灯，1 个状态指示灯；
- 1 个复位按钮，可用于复位 MCU 。

## 1.3 典型应用

- 工业级 PLC 开发；
- CAN 工业自动化控制系统开发。

# 2.GCAN-PLCCore-V3 引脚定义及功能说明

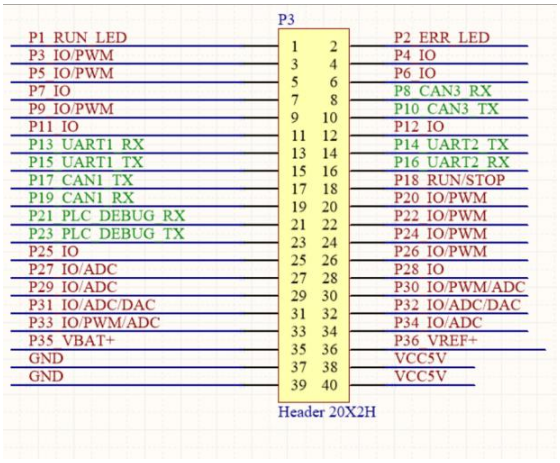
下表中对 GCAN-PLCCore-V3 的引脚定义及功能进行了详细的说明。其中，IO 表示可以做为 DI 和 DO 使用，X 表示禁止使用，引脚悬空即可。

| 引脚序号 | 引脚编号 | 功能定义 | 默认映射 | 引脚序号 | 引脚编号 | 功能定义 | 默认映射 |
|---|---|---|---|---|---|---|---|
| **P3** | | | | **P4** | | | |
| 1 | P1 | P1_RUN_LED | | 1 | P41 | P41_IO | Q14 |
| 2 | P2 | P2_ERR_LED | | 2 | P42 | P42_IO | Q15 |
| 3 | P3 | P3_IO/PWM | X01/PI-2 | 3 | P43 | P43_IO | Q16 |
| 4 | P4 | P4_IO | Q11 | 4 | P44 | P44_IO | Q17 |
| 5 | P5 | P5_IO/PWM | X03/PI-1-B | 5 | P45 | P45_IO | Q18 |
| 6 | P6 | P6_IO | Q12 | 6 | P46 | P46_IO | Q19 |
| 7 | P7 | P7_IO | Q13 | 7 | P47 | P47_IO | Q20 |
| 8 | P8 | P8_CAN3_RX | | 8 | P48 | P48_IO | Q21 |
| 9 | P9 | P9_IO/PWM | Q03/PWM-4 | 9 | P49 | P49_IO | X14 |
| 10 | P10 | P10_CAN3_TX | | 10 | P50 | P50_IO | X15 |
| 11 | P11 | P11_IO | Q06 | 11 | P51 | P51_IO | X16 |
| 12 | P12 | P12_IO | Q07 | 12 | P52 | P52_IO | X17 |
| 13 | P13 | P13_UART1_RX | | 13 | P53 | P53_IO | X18 |
| 14 | P14 | P14_UART2_TX | | 14 | P54 | P54_IO | X19 |
| 15 | P15 | P15_UART1_TX | | 15 | P55 | P55_IO | X20 |
| 16 | P16 | P16_UART2_RX | | 16 | P56 | P56_IO | X21 |
| 17 | P17 | P17_CAN1_TX | | 17 | P57 | P57_IO | X22 |
| 18 | P18 | P18_RUN/STOP | | 18 | P58 | P58_IO | X23 |
| 19 | P19 | P19_CAN1_RX | | 19 | P59 | P59_IO | X24 |
| 20 | P20 | P20_IO/PWM | X10 | 20 | P60 | P60_IO | X25 |
| 21 | P21 | P21_PLC_DEBUG_RX | | 21 | P61 | P61_IO | X26 |
| 22 | P22 | P22_IO/PWM | X11 | 22 | P62 | P62_IO | X27 |
| 23 | P23 | P23_PLC_DEBUG_TX | | 23 | P63 | P63_IO | X28 |
| 24 | P24 | P24_IO/PWM | X12 | 24 | P64 | P64_IO | X29 |
| 25 | P25 | P25_IO | Q08 | 25 | P65 | P65_ETH_TXD_P | |

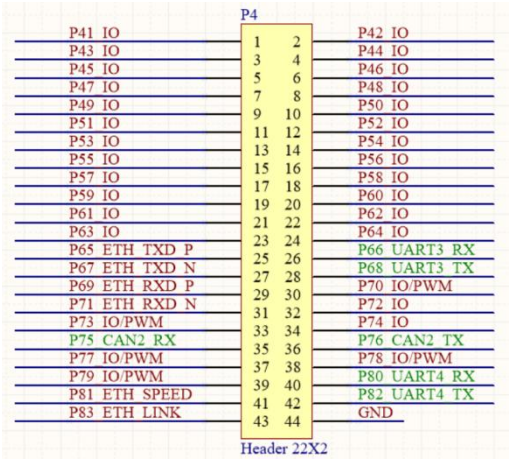
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 26 | P26 | P26_IO/PWM | X13 | 26 | P66 | P66_UART3_RX | |
| 27 | P27 | P27_IO/ADC | X04/AI_1 | 27 | P67 | P67_ETH_TXD_N | |
| 28 | P28 | P28_IO | X09 | 28 | P68 | P68_UART3_TX | |
| 29 | P29 | P29_IO/ADC | X05/AI_2 | 29 | P69 | P69_ETH_RXD_P | |
| 30 | P30 | P30_IO/PWM/ADC | X07/AI_3 | 30 | P70 | P70_IO/PWM | X00/PI-1-A |
| 31 | P31 | P31_IO/ADC/DAC | Q04/AO_1 | 31 | P71 | P71_ETH_RXD_N | |
| 32 | P32 | P32_IO/ADC/DAC | Q05/AO_2 | 32 | P72 | P72_IO | Q09 |
| 33 | P33 | P33_IO/PWM/ADC | X08/AI_4 | 33 | P73 | P73_IO/PWM | Q00/PWM-1 |
| 34 | P34 | P34_IO/ADC | X06/AI_5 | 34 | P74 | P74_IO | Q10 |
| 35 | P35 | P35_VBAT+ | | 35 | P75 | P75_CAN2_RX | |
| 36 | P36 | P36_VREF+ | | 36 | P76 | P76_CAN2_TX | |
| 37 | P37 | GND | | 37 | P77 | P77_IO/PWM | Q01/PWM-2 |
| 38 | P38 | VCC5V | | 38 | P78 | P78_IO/PWM | X02/PI-3 |
| 39 | P39 | GND | | 39 | P79 | P79_IO/PWM | Q02/PWM-3 |
| 40 | P40 | VCC5V | | 40 | P80 | P80_UART4_RX | |
| | | | | 41 | P81 | P81_ETH_SPEED | |
| | | | | 42 | P82 | P82_UART4_TX | |
| | | | | 43 | P83 | P83_ETH_LINK | |
| | | | | 44 | P84 | GND | |

GCAN-PLCCore-V3 核心板采用排针排座方式与底板连接，如用户想自己开发底板，则底板参考原理图及引脚定义如下图所示，具体连接器原理图及封装可收到货后向技术支持工程师索要。

P3：




P4：



# 3.GCAN-PLCCore-V3 启动策略

默认情况下，GCAN-PLCCore-V3 在上电或重启时会加载所有必要的固件程序，之后会开始运行 PLC 程序。因此，GCAN-PLCCore-V3 适合使用于独立型的控制系统。在电源崩溃的情况下，这样的系统能够恢复执行 PLC 程序，而无需用户干预。下图演示了系统启动的细节：

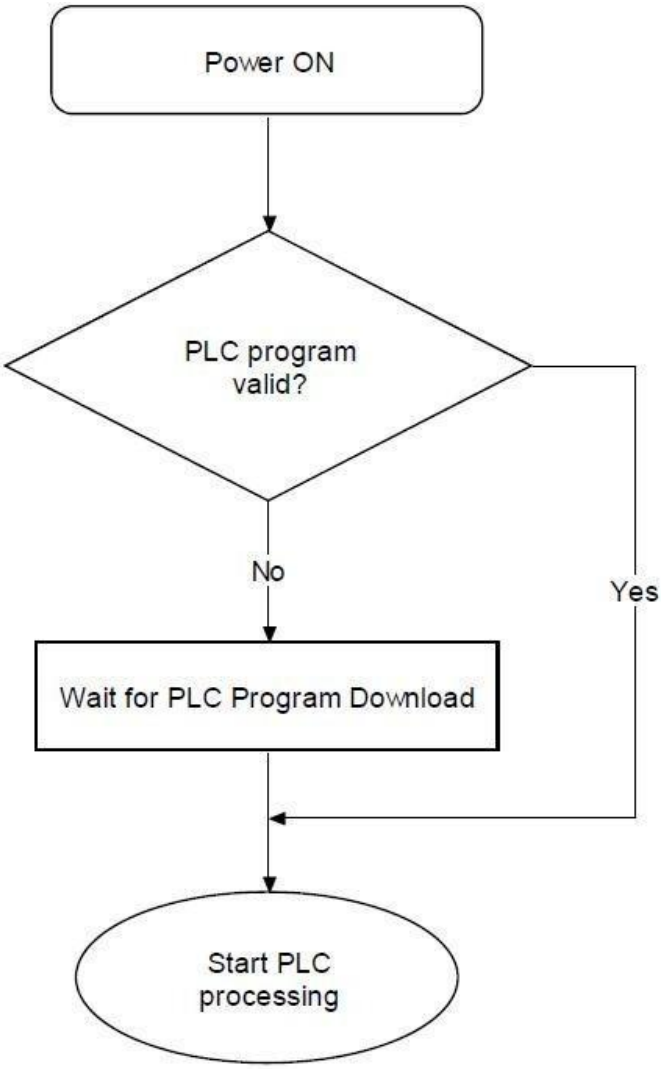



图3.1 GCAN-PLCCore-V3启动策略

如果固件在非易失性存储器中找到了有效的控制程序，则该程序将重新启动和处理。否则将激活程序，进入 GCAN-PLCCore-V3 停止模式。

# 4.GCAN-PLCCore-V3 核心板参数表

| | |
|---|---|
| 控制器特点 | |
| 控制器 | STM32H7 |
| 内核 | ARM® 32-bit Cortex®-M7 |
| 处理器频率 | 400MHz |
| 程序存储空间 | 4M 字节 |
| 数据存储器空间 | 4M 字节 |
| 掉电保存空间 | 4K 字节 |
| 接口特点 | |
| 以太网 | 1路，10/100Mbps |
| RS232/RS485 | 4路，2400bps~115200bps |
| CAN接口 | 3路，遵循ISO 11898标准，支持CAN2.0A/B |
| IO点位接口 | 52路 |
| 板载接口 | 一个40pin 插座、一个44pin 插座，2.0mm管脚间距 |
| 软件 | 内部集成IEC61131-3实时内核 |
| 编程接口 | TCP |
| RTC | 板载实时时钟 |
| 供电电源 | |
| 供电电压 | +5V DC（±5%） |
| 供电电流 | 200mA |
| 环境试验 | |
| 工作温度 | -40℃~+85℃ |
| 工作湿度 | 15%~90%RH，无凝露 |
| EMC测试 | Zone B，IEC61131-2 |
| 防护等级 | IP 20 |
| 基本信息 | |
| 外形尺寸 | 65mm *45mm |
| 重量 | 10g |

# 5.PLCCore-EDV-V3 开发套件

PLCCore-EDV-V3 开发套件是一套高性能低成本的开发套件。基于 GCAN-PLCCore-V3 核心板，开发套件 PLCCore-EDV-V3 能使用户快速完成分散、网络相兼容的自动化项目。

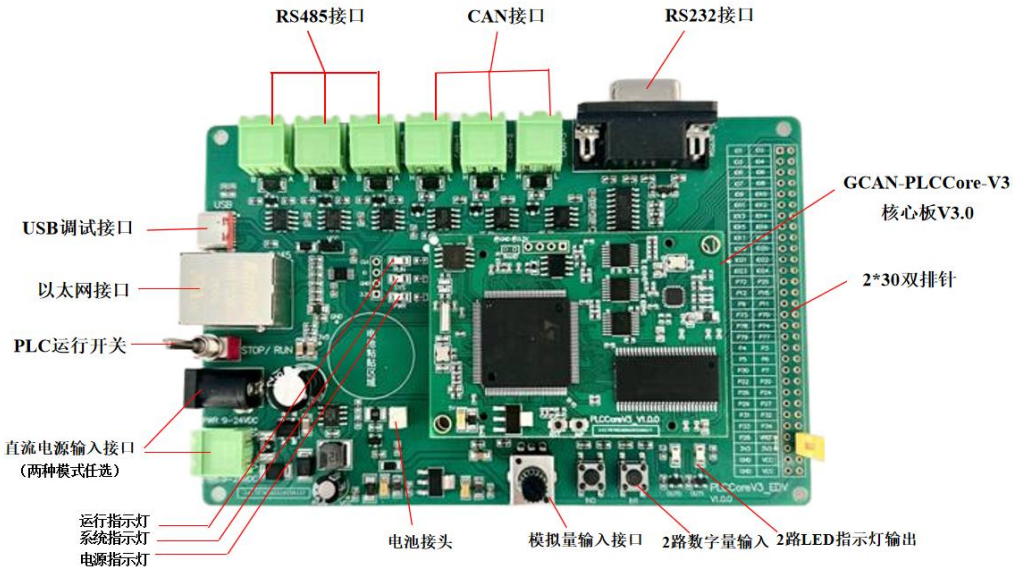



图5.1 PLCCore-EDV-V3 开发套件

GCAN-PLCCore-V3 核心板配套开发板底板的板载资源如下：

- 1 个核心板接口，支持 GCAN-PLCCore-V3 核心板；
- 1 个电源指示灯，2 个状态指示灯；
- 3 路CAN 接口，1 路以太网接口；
- 1 路RS232 接口，3 路RS485 接口；
- 52路DI/DO（可复用，其中2路DI有按键输入、2路DO有LED指示灯输出）；
- 5路模拟量输入（与DI复用），2路模拟量输出（与DO复用）；
- 1 路USB 接口，可用于 PLC 程序下载和代码调试、供电；
- 1 个直流电源输入接口（输入电压范围：DC 9-30V）；
- 1 个开关，用于 PLC 的RUN 和STOP；
- 一排针脚数量为40、一排针脚数量为44的双排针，用于引出核心板内其余的引脚；
- 引出 2x30 个接口，包含电源及 IO 接口，满足各种应用需求；
- 外形尺寸为 150mm*100mm；
- 1个电池接头可用于接电池，若要想使用掉电保持区必须接电池。

# 6.PLCCore-EDV-V3 开发套件参数

PLCCore-EDV-V3 开发套件（GCAN-PLCCore-V3 核心板配套底板）技术参数如下所示：

| | |
|---|---|
| 核心板特点 | |
| 核心板 | GCAN-PLCCore-V3 |
| 内核 | ARM® 32-bit Cortex®-M7 |
| 接口特点 | |
| 以太网 | 1路，10/100Mbps |
| RS232 | 1路，2400bps~115200bps |
| RS485 | 3路，2400bps~115200bps |
| CAN接口 | 3路，遵循ISO 11898标准，支持CAN2.0A/B |
| CAN波特率 | 1000K、500K、400K、250K、200K、125K、100K、80K、50K、40K、20K、10K |
| 数字量输入/输出 | 52路DI/DO（可复用，其中2路DI有按键输入、2路DO有LED指示灯输出） |
| 模拟量输入 | 5路（与DI复用，其中1路AI有独立输入端） |
| 模拟量输出 | 2路（与DO复用） |
| RTC | 板载实时时钟 |
| 供电电源 | |
| 供电电压 | +9-30V DC（额定电压24V DC） |
| 环境试验 | |
| 工作温度 | -40℃~+85℃ |
| 工作湿度 | 15%~90%RH，无凝露 |
| EMC测试 | Zone B，IEC61131-2 |
| 防护等级 | IP 20 |
| 基本信息 | |
| 外形尺寸 | 150mm *100mm |
| 重量 | 100g |

# 7.Codesys 软件使用

## 7.1 Codesys 下载

Codesys 编程软件是标准的 Windows 界面，支持编程、调试及配置。

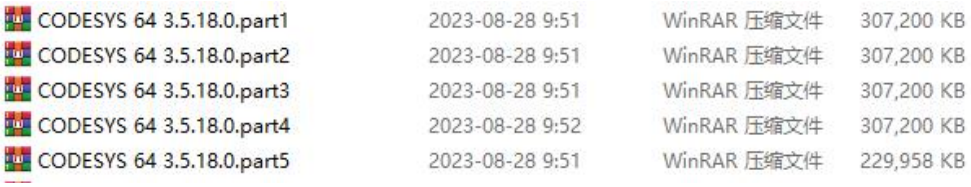
GCAN-PLC 需使用 Codesys v3.5.16.0 及以上版本。

注意：3.5.18.0 版本在 Windows7 系统上是无法运行的。Windows7 系统。建议您安装 3.5.16.0 版本。安装包可以到 Codesys 官网下载或登录 http://www.gcan.com.cn/580e/f525/e79f/3ce0 网址进行下载。

## 7.2 Codesys 软件安装

1）首先先下载 Codesys 安装包的压缩包。

2）将所有压缩包放到一个文件里面，然后解压第一个压缩包。如图所示是分包压缩，其实是一 个压缩包分成多个，便于文件传输。




注意 ：一定要把所有压缩包解压到同一个文件夹里面，否则安装包是不完整的。

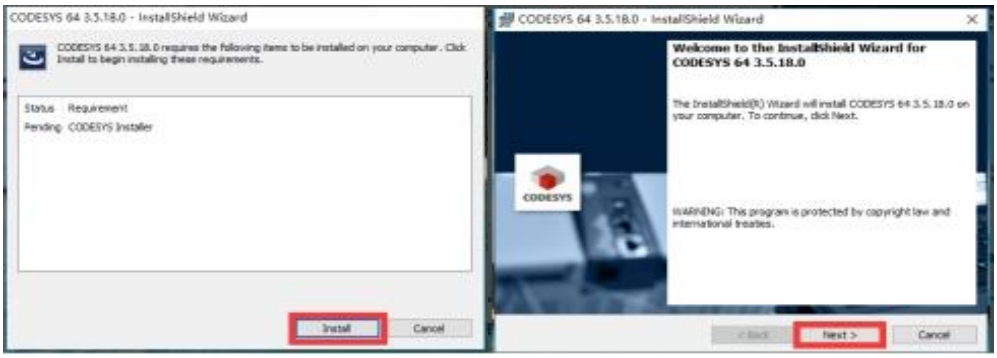
3）所有压缩包放到一个文件里面，然后解压第一个压缩包。如图所示是分包压缩，其实是一 个压缩包分成多个，便于文件传输。

注意：一定要把所有压缩包解压到同一个文件夹里面，否则安装包是不完整的。

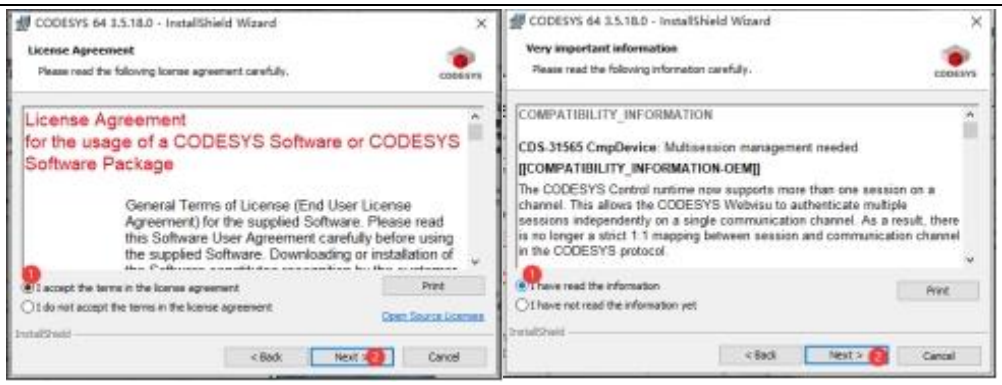
4）解压完成后，会出现对应版本的安装程序，双击安装程序启动安装。



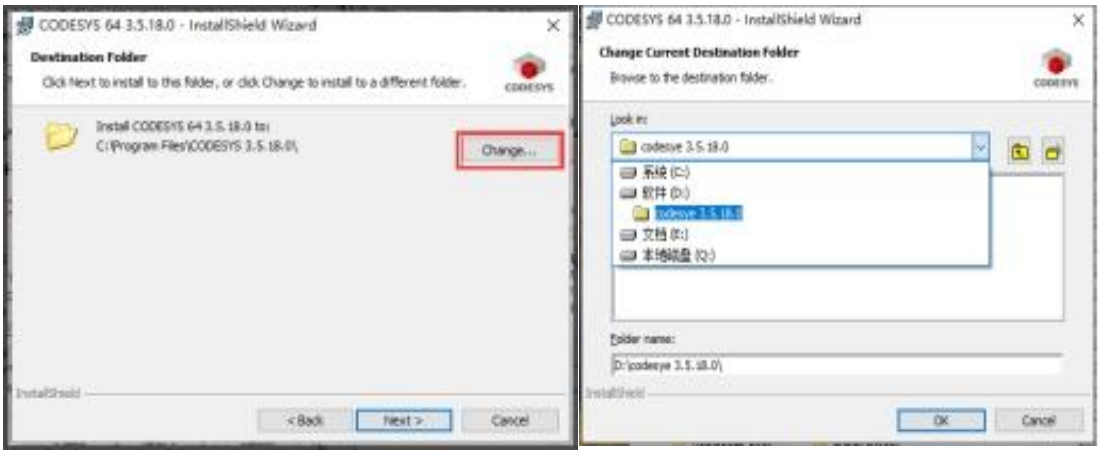

5）点击 Install ，然后点击 Next>



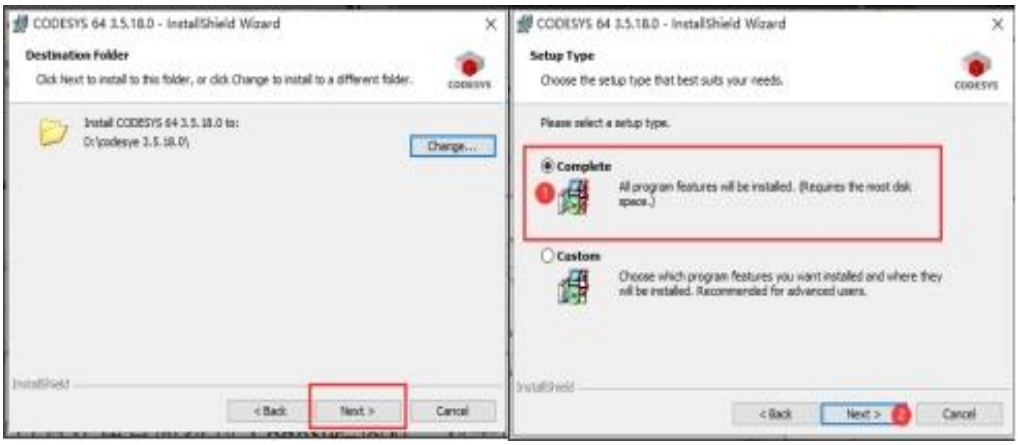

6） 选择 I accept the terms in the license agreement ， 然后点击 Next>接下来选择 I have read the information ， 然后点击 Next。




7）点击 Change...来选择您想把软件安装到的位置，我这里选择安装到 D 盘 Codesys3.5.18.0 这个文件夹中，点击 OK。

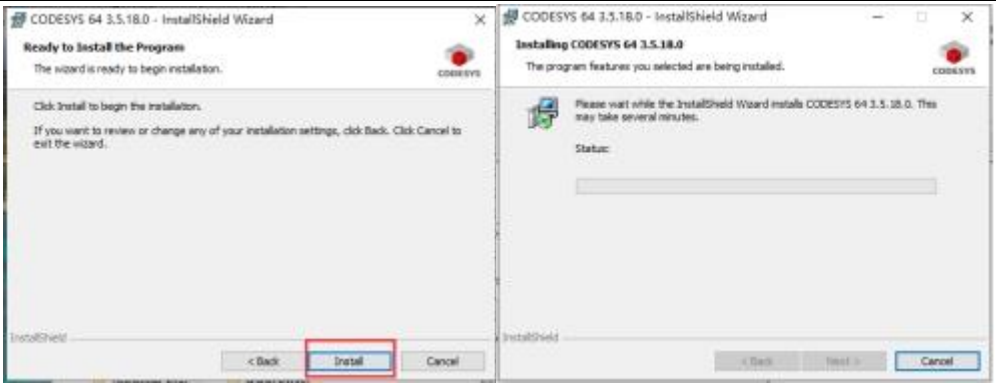



8） 点击 Next>，然后选择 Complete，再点击 Next>。




9）再点击 Install，然后会进入读条安装的过程。

注：安装时间因电脑性能不同而长短不一，如果进度缓慢请耐心等待。




注：如提示以下弹窗，请单独安装 WebView

下载地址：http://www.gcan.com.cn/580e/f525/e79f/d13e

10）安装完成后，可以直接在桌面找到 Codesys 图标，双击运行打开会显示软件版本。




## 7.3 界面介绍与语言切换

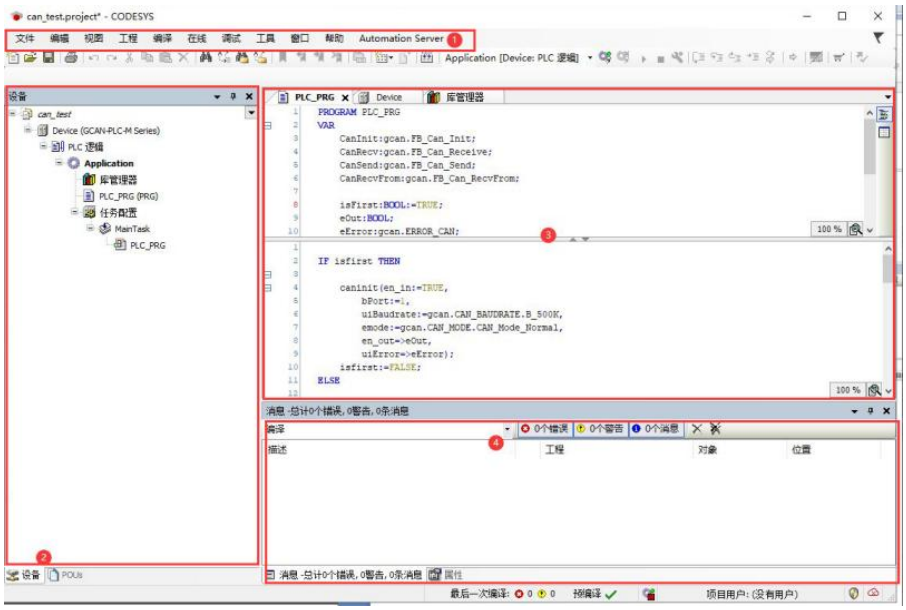
（1） 编程界面介绍

1. 菜单工具栏

2. 设备/ POU 窗口

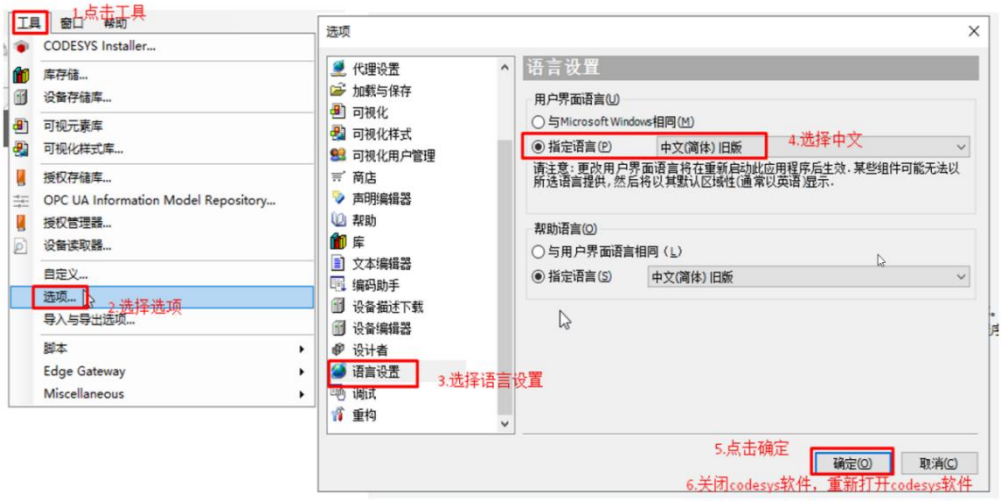
3. 编程窗口

4.消息窗口



（2） 切换中文环境

点击工具(TOOLS)=>选项(options)=>语言设置(International Setting)=>指定语言(specific language)=>中文=>重启 Codesys 软件




## 7.4 添加设备描述文件

相关资料下载地址：http://www.gcan.com.cn/580e/95a1/c063/045e



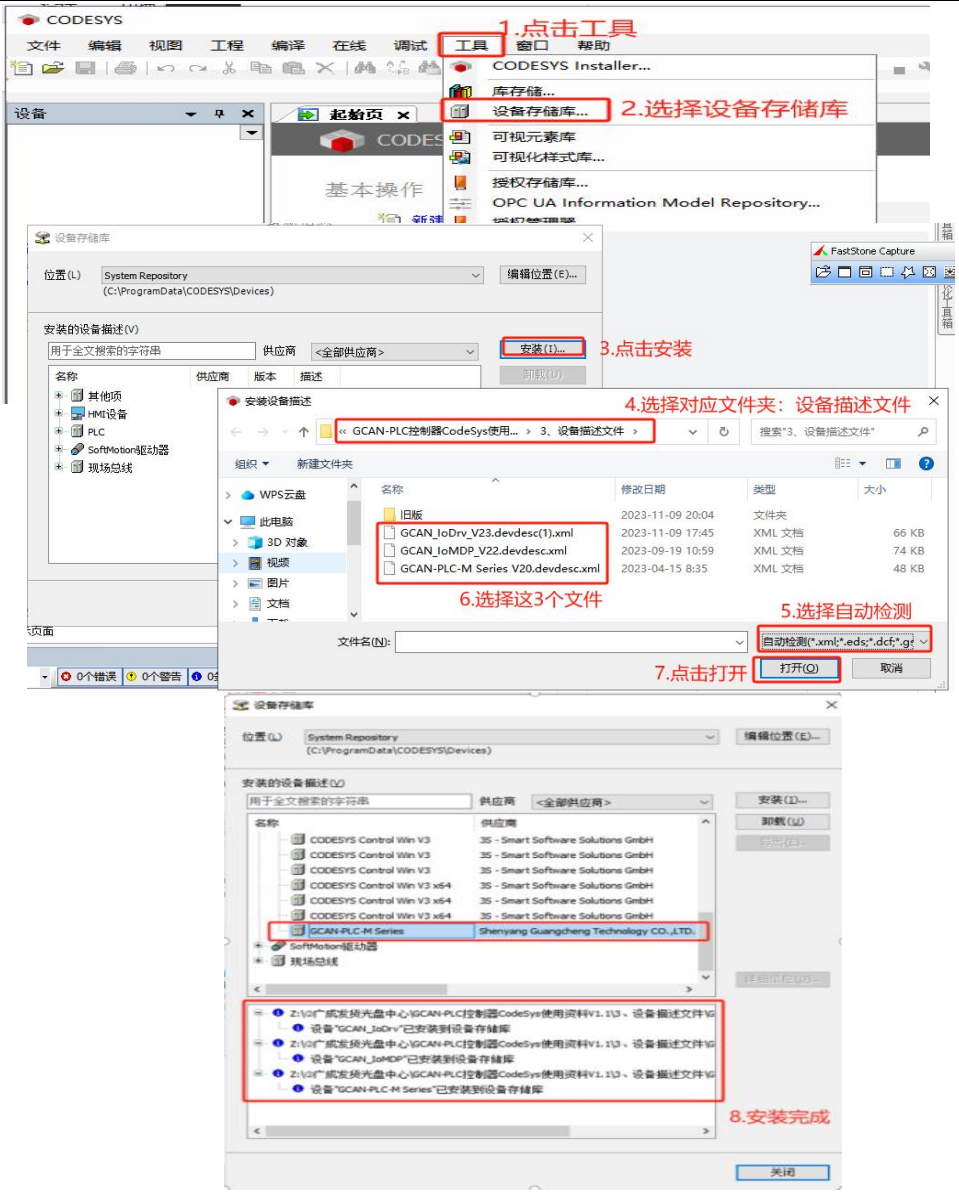

1）软件双击桌面 Codesys 标志，打开 Codesys




2）安装设备描述文件：选择工具=>设备存储库，在打开的界面中进行设备描述文件的安装。

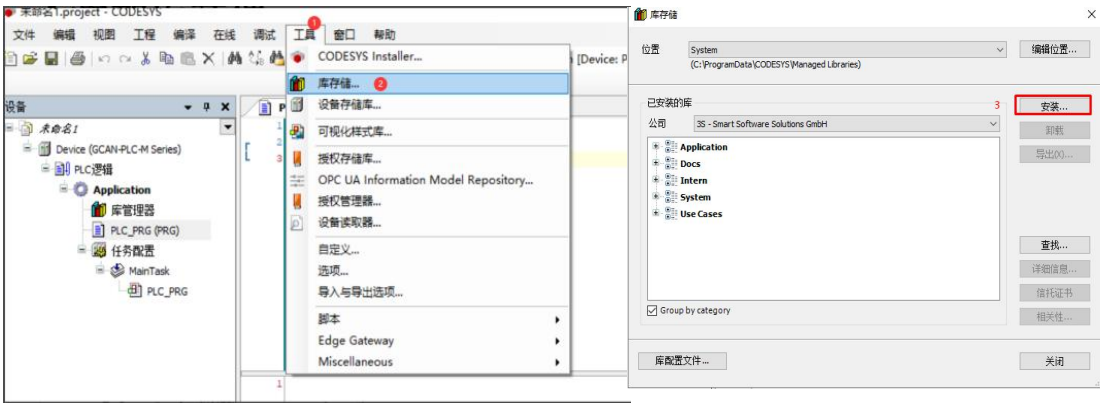
GCAN-PLC-530/GCAN-PLC-521 使用：GCAN-PLC-A Series

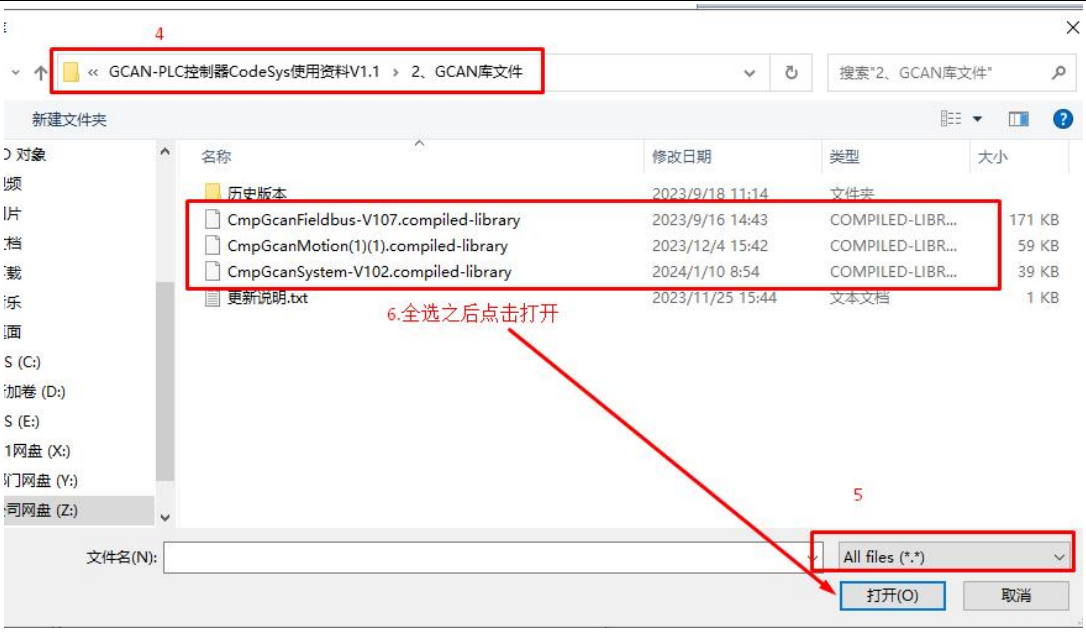
其他型号：GCAN-PLC-M Series



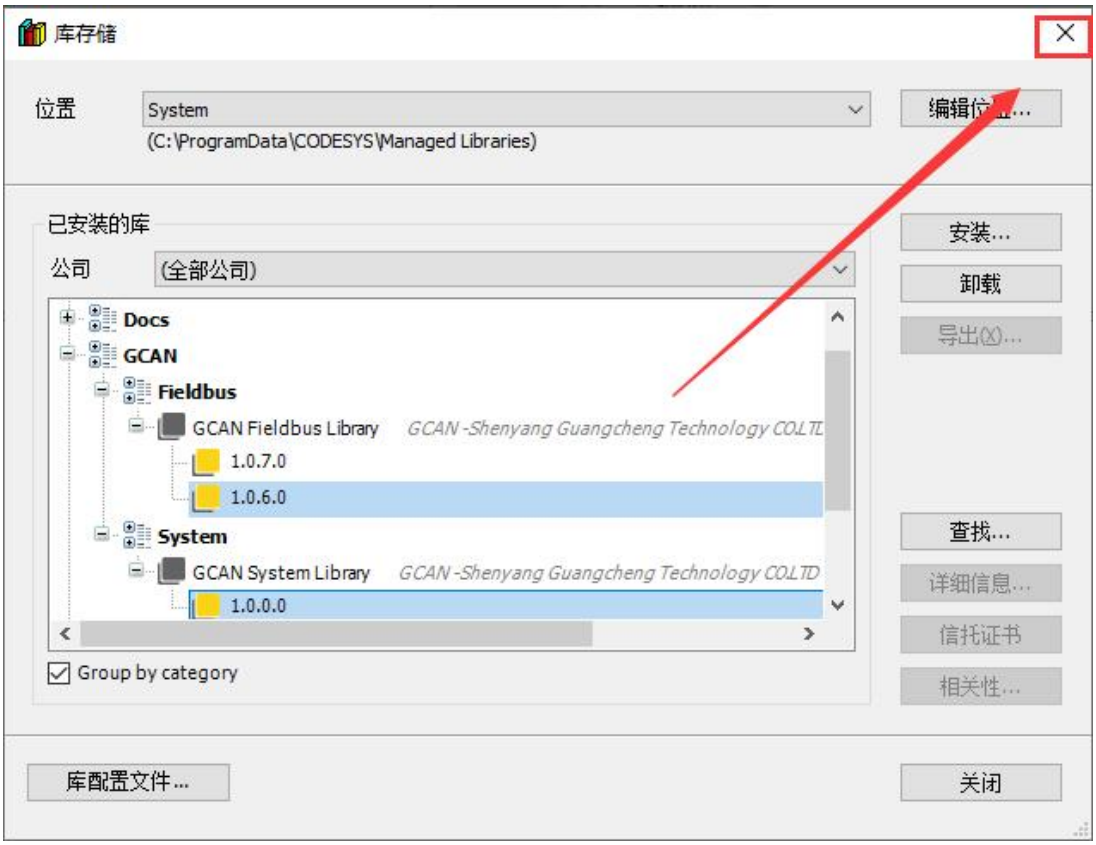

## 7.5 安装库文件

1）在 Codesys 编程界面的上方找到工具点开，选择库存储...，点击右侧安装...，选择想要安装的库文件然后点打开。



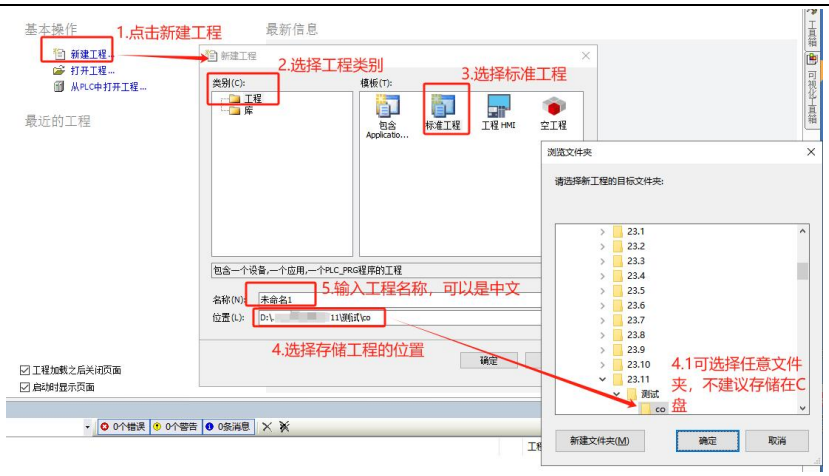


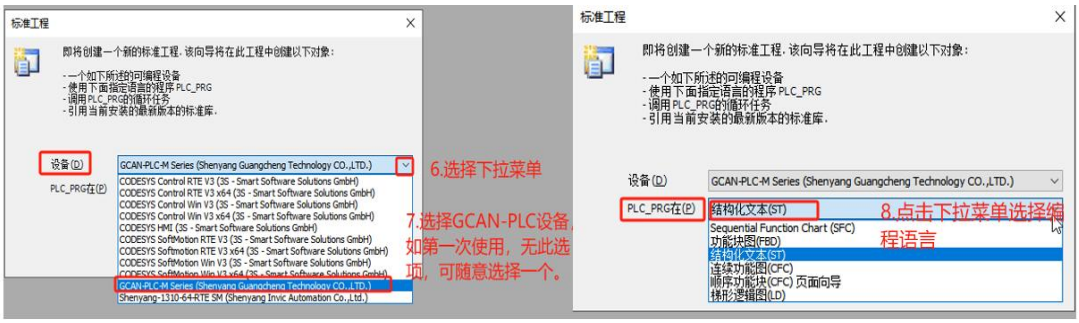

2）添加完毕后点击右上角关闭即可。




## 7.6 新建工程

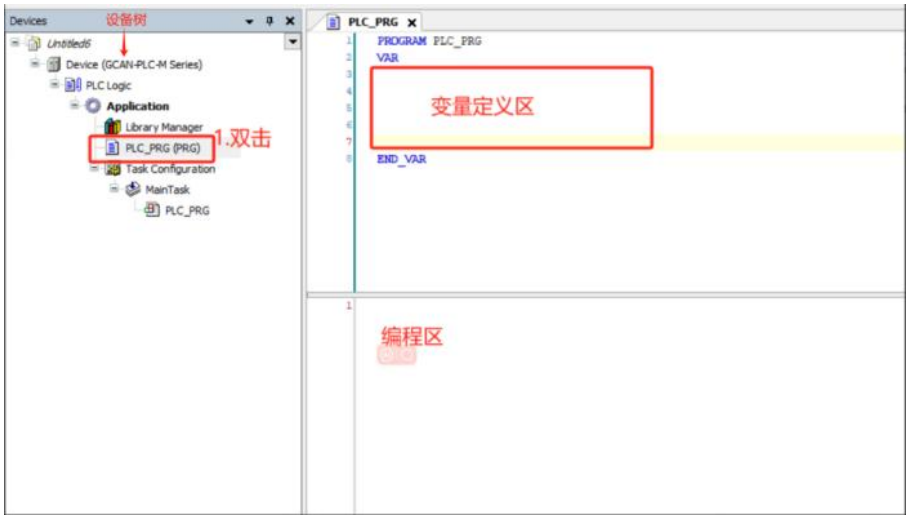
点击新建工程...，如果您的手上有程序想要打开，请点击打开工程...，如下图


基本操作
1.点击新建工程
最新信息
新建工程
打开工程...
从PLC中打开工程...
最近的工程
2.选择工程类别
3.选择标准工程
类别(C):
模板(T):
标准工程
工程 HMI
空工程
5.输入工程名称，可以是中文
4.选择存储工程的位置
4.1可选择任意文件夹，不建议存储在C盘
新建文件夹(M)
确定
取消


标准工程
即将创建一个新的标准工程. 该向导将在此工程中创建以下对象:
- 一个如下所述的可编程设备
- 使用下面指定语言的程序 PLC_PRG
- 调用 PLC_PRG的循环任务
- 引用当前安装的最新版本的标准库.
设备(D)
PLC_PRG在(P)
GCAN-PLC-M Series (Shenyang Guangcheng Technology CO.,LTD.)
6.选择下拉菜单
7.选择GCAN-PLC设备，如第一次使用，无此选项，可随意选择一个。
结构化文本(ST)
8.点击下拉菜单选择编程语言
Sequential Function Chart (SFC)
功能块图(FBD)
连续功能图(CFC)
顺序功能块(CFC) 页面向导
梯形逻辑图(LD)


标准工程
即将创建一个新的标准工程. 该向导将在此工程中创建以下对象:
- 一个如下所述的可编程设备
- 使用下面指定语言的程序 PLC_PRG
- 调用 PLC_PRG的循环任务
- 引用当前安装的最新版本的标准库.
设备(D)
GCAN-PLC-M Series (Shenyang Guangcheng Technology CO.,LTD.)
PLC_PRG在(P)
结构化文本(ST)
9.点击确定
确定
取消

编程界面如图所示：

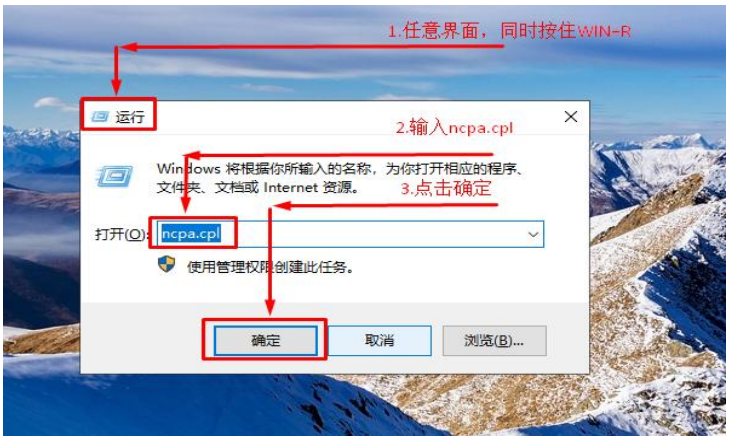

Devices
设备树
Device (GCAN-PLC-M Series)
PLC Logic
Application
Library Manager
PLC_PRG (PRG)
1.双击
Task Configuration
MainTask
PLC_PRG
PLC_PRG
PROGRAM PLC_PRG
VAR
变量定义区
END_VAR
编程区

# 8.设备连接

进行设备连接前，请保证 PLC 电源正确输入，用网口进行连接。

## 8.1 更改电脑 IP

打开网络适配器界面：任意界面，同时按住键盘 WIN+R 打开运行界面=>输入 ncpa.cpl=>点击确定

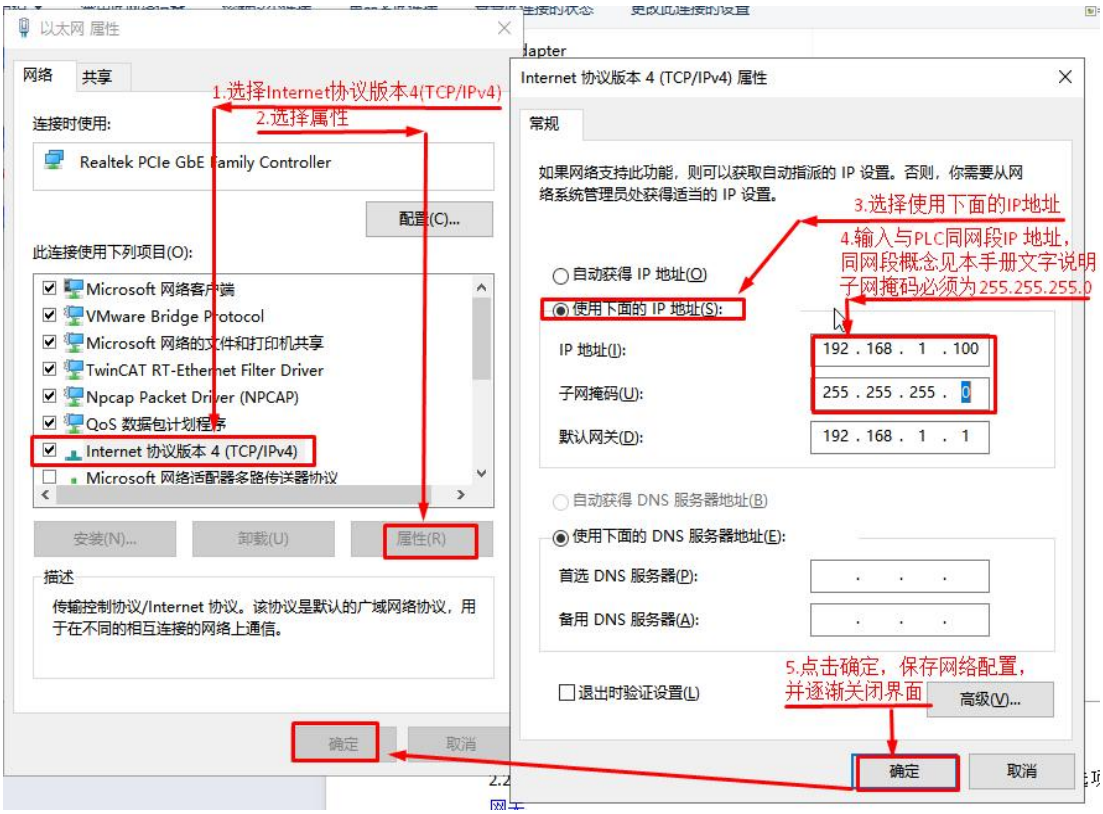



设置电脑 IP：选择对应网口=>右键选择属性=>IPv4=>属性=>使用下面的 IP 地址=>输入同网段IP=>输入子网掩码=>确定=>逐级关闭界面即可

同网段：如与 GCAN-PLC 保持同网段，GCAN-PLC 默认 IP 为 192.168. 1.30，则同网段设备应设置IP 为 192.168. 1.n

n 应为 1-254 之间的正整数，但不能是 192.168. 1.30 、192.168. 1.10 ，n=10,30 会造成地址冲突。地址冲突：同一网段的设备有两个相同的IP 地址。






## 8.2 Codesys 软件配置

1）连接双击 Device（GCAN-PLC-M Series），点击网关，在弹出的下拉选项中选择配置本地网关...




2）点击添加，点击添加接口...，下拉选项中选择 UDP 接口，点击确定。



3）点击扫描网络，在 Gateway- 1 下方会出现对应的网口连接的设备。




4）如下图所示，即已完成网络连接，然后设备就通过网口连接到 Codesys 软件上了。




## 8.3 下载程序

连接设备以后，选择在线=>登录=> 。会弹出是否创建此应用并下载的提示，选择是。



# 9. PLCCore-EDV-V3 开发套件基本例程

## 9.1 CAN数据收发实验例程

功能：1.收到一帧CAN数据后，ID+1，数据不变发出。如果接收功能块没有接收到数据，每间隔1000ms往外发送一条数据；

2.收到制定帧ID，ID+2,数据不变发出去；

应用场景：广泛应用于CAN2.0协议通讯；

例程如下：1.变量定义区：

```
PROGRAM PLC_PRG
VAR
    CanInit:gcan.FB_Can_Init;
    CanRecv:gcan.FB_Can_Receive;
    CanSend:gcan.FB_Can_Send;
    CanRecvFrom:gcan.FB_Can_RecvFrom;
    isFirst:BOOL:=TRUE;
    eOut:BOOL;
    eError:gcan.ERROR_CAN;
    msg:gcan.CAN_MSG_STRUCT;
    newData:BOOL:=FALSE;
    fromNewData:BOOL:=FALSE;
    bport:BYTE:=0;
    tt: INT;
END_VAR
```

2. 程序实现区：

```
bport:=0;
IF isfirst THEN
    caninit(en_in:=TRUE,
    bPort:=bport,
    uiBaudrate:=gcan.CAN_BAUDRATE.B_500K,
    emode:=gcan.CAN_MODE.CAN_Mode_Normal,
    en_out=>eOut,
    uiError=>eError);
    isfirst:=FALSE;
ELSE
```

```
    CanRecv(en_in:=TRUE,
    bport:=bport,
    p_canmsg:=ADR(msg),
    en_out=>eout,
    bresult=>newData,
    uierror=>eError);
    tt:=tt+1;
IF newData THEN
    msg.bID:=msg.bID+1;
    CanSend(en_in:=TRUE,
    bport:=bport,
    p_canmsg:=ADR(msg),
    en_out=>eout,
    uierror=>eError);
    tt:=0;
ELSIF newData=0 AND tt=1000 THEN
    msg.bID:=16#000;
    msg.bDLC:=8;
    msg.b_Data[0]:=16#11;
    msg.b_Data[1]:=16#11;
    msg.b_Data[2]:=16#11;
    msg.b_Data[3]:=16#11;
    msg.b_Data[4]:=16#11;
    msg.b_Data[5]:=16#11;
    msg.b_Data[6]:=16#11;
    msg.b_Data[7]:=16#11;
    CanSend(en_in:=TRUE,
    bport:=bport,
    p_canmsg:=ADR(msg),
    en_out=>eout,
    uierror=>eError);
    tt:=0;
END_IF;
    CanRecvFrom(en_in:=TRUE,
```

```
            bPort:=bport,
            bMaskid:=1,
            bMaskExtended:=0,
            bMaskRTR:=0,
            p_CanMsg:=ADR(msg),
            EN_OUT=>eout,
            bResult=>fromNewData,
            uiError=>eError);
    IF fromNewData THEN
            msg.bID:=msg.bID+2;
            CanSend(en_in:=TRUE,
            bport:=bport,
            p_canmsg:=ADR(msg),
            en_out=>eout,
            uierror=>eError);
    END_IF
    END_IF
```

## 9.2 下载程序及重点参数

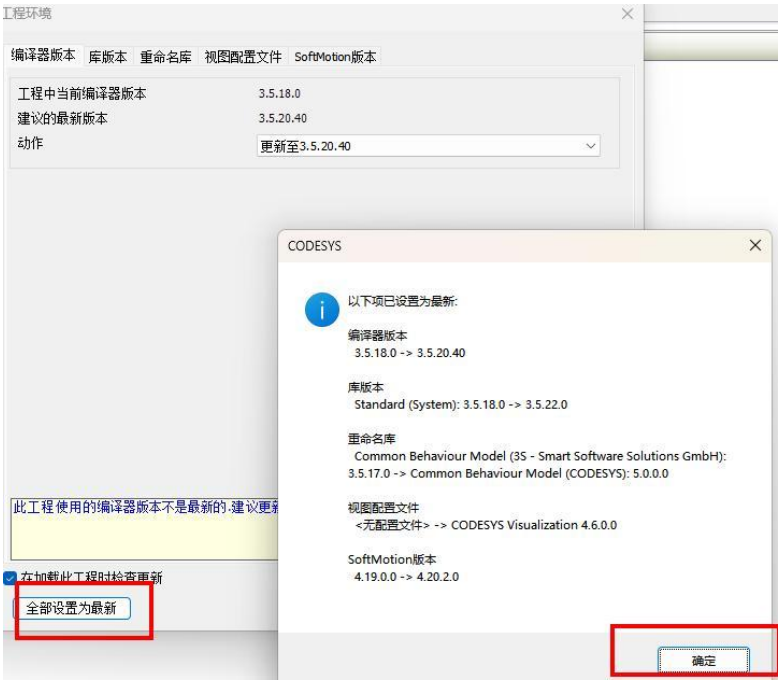
（1）下载好CAN数据收发实验ST源码，解压并打开程序；



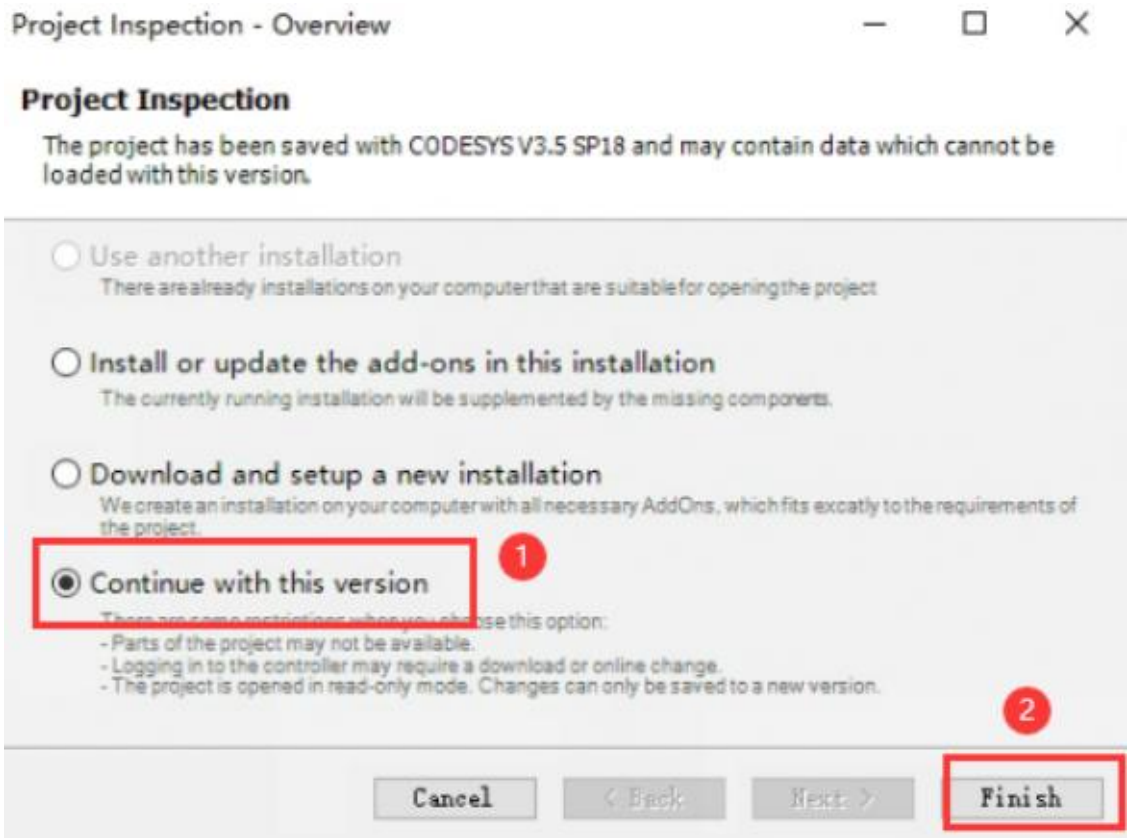

**备注：**打开程序后如出现如下图的弹窗：“解压到以下文件夹”→通过“...”选择解压程序所在位置→勾选②框内的所有复选框→点击确定



如出现下图的弹窗：单击“全部设置为最新”后，单击“确定”然后再单击“确定”

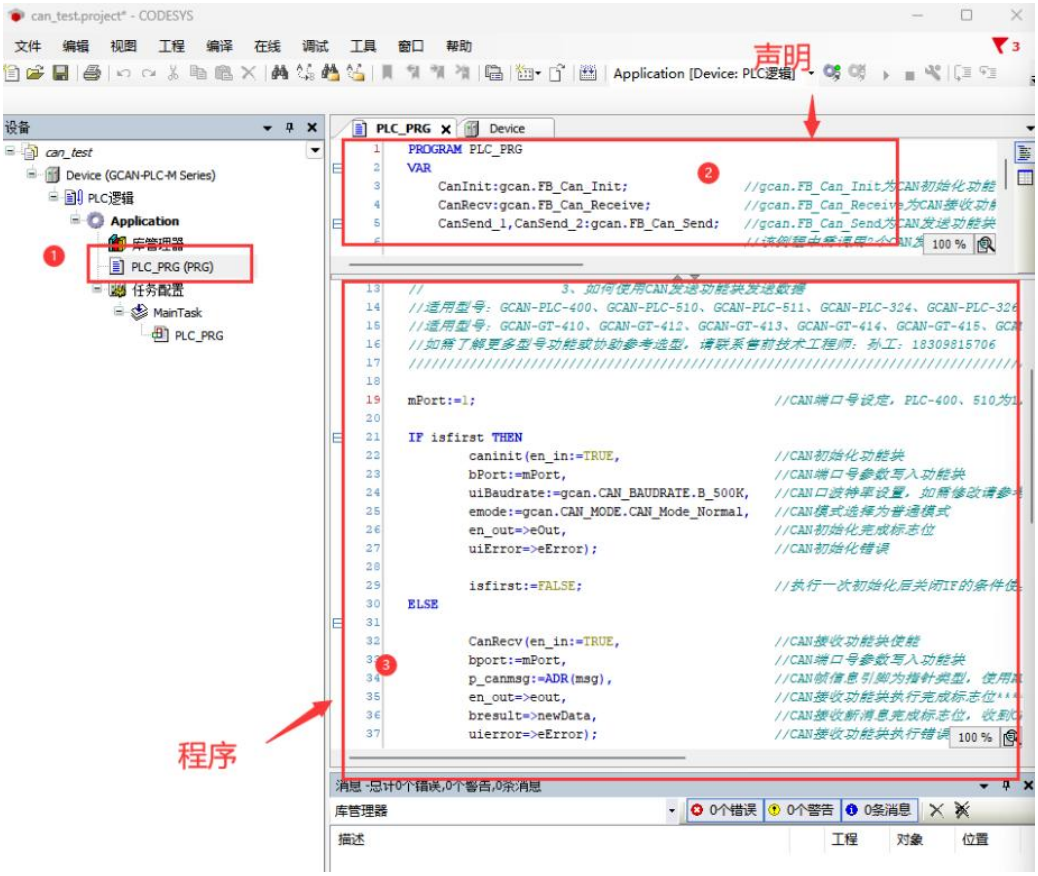



如出现下图弹窗：continue with this version→finish




如程序出现报错是库文件安装的问题，请安装GCAN的库文件

（2）点开PLC_PRG(PRG)，上半部分为程序声明，下半部分为具体程序。




（3） mPort为CAN的端口号，uiBaudrate为波特率，此程序中为500k，emode为模式，一般选普通模式。注：PLCCore-EDV-V3有三个CAN口，CAN1的端口号为0，CAN2的端口号为1，CAN3的端口号为2。




（4） 接收can数据后，数据不变，帧ID+1发送出去。如果接收功能块没有接收到数据，每间隔1000ms往外发送一条数据；

```
CanRecv(en_in:=TRUE,            //CAN接收功能块使能
    bport:=bport,               //CAN端口号参数写入功能块
    p_canmsg:=ADR(msg),         //CAN帧信息引脚为指针类型，使用ADR()指向gcan.CAN_MSG_STRUCT类型的结构体变量msg，结构体中包含的内容详见GCAN库
    en_out=>eout,               //CAN接收功能块执行完成标志位'''''''功能块执行完成后输出一个扫描周期的TRUE'''''''
    bresult=>newData,           //CAN接收新消息完成标志位，收到CAN数据的当周期置1，下周期再无信息自动清0
    uierror=>eError);           //CAN接收功能块执行错误
tt:=tt+1;
IF newData THEN         //如果收到新消息的标志位为true，则调用CAN发送功能块，将CAN ID+1后发出去
    msg.bID:=msg.bID+1;         //ID加1''''''''''变量+ ，可查看CAN_MSG_STRUCT结构体中的可用内容''''''''''
    CanSend(en_in:=TRUE,        //CAN发送功能块使能，发送机制：当使能为TRUE时，PLC程序每周期运行到此处就执行一次发送
    bport:=bport,               //CAN端口号参数写入功能块
    p_canmsg:=ADR(msg),         //CAN帧信息引脚为指针类型，使用ADR()指向gcan.CAN_MSG_STRUCT类型的结构体变量msg，结构体中包含的内容详见GCAN库
    en_out=>eout,               //CAN发送功能块执行完成标志位
    uierror=>eError);           //CAN发送功能块执行错误
    tt:=0;                      //发送成功以后时间标志位置为0
ELSIF newData=0 AND tt=1000 THEN            //如果没有收到新消息且间隔时间到达1000ms，则调用CAN发送功能块，
    msg.bID:=16#000;            //can的ID为0x000
    msg.bDLC:=8;                //can的数据长度为8
    msg.b_Data[0]:=16#11;       //can数据全为16#11
    msg.b_Data[1]:=16#11;
    msg.b_Data[2]:=16#11;
    msg.b_Data[3]:=16#11;
    msg.b_Data[4]:=16#11;
    msg.b_Data[5]:=16#11;
    msg.b_Data[6]:=16#11;
    msg.b_Data[7]:=16#11;
    CanSend(en_in:=TRUE,        //CAN发送功能块使能，发送机制：当使能为TRUE时，PLC程序每周期运行到此处就执行一次发送
    bport:=bport,               //CAN端口号参数写入功能块
    p_canmsg:=ADR(msg),         //CAN帧信息引脚为指针类型，使用ADR()指向gcan.CAN_MSG_STRUCT类型的结构体变量msg，结构体中包含的内容详见GCAN库
    en_out=>eout,               //CAN发送功能块执行完成标志位
    uierror=>eError);           //CAN发送功能块执行错误
    tt:=0;                      //发送成功以后时间标志位置为0
```

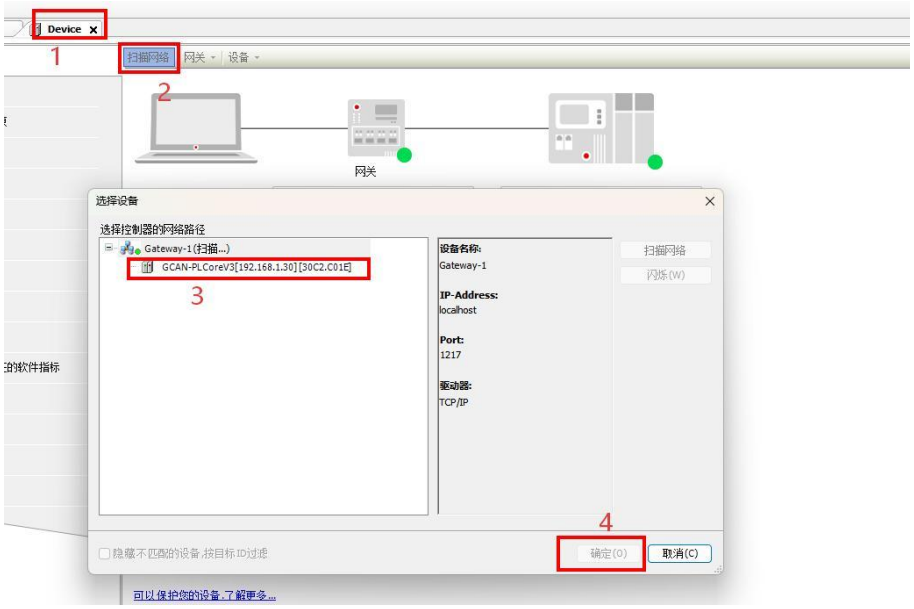


（5）定向接收帧ID为0X001，帧ID+2，数据不变发出去

```
46          uierror=>eError);                   //CAN发送功能块执行错误
47
48      END_IF;
49
50      CanRecvFrom(en_in:=TRUE,                //CAN定向接收功能块使能
51      bPort:=mPort,                           //CAN端口号参数写入功能块
52      bMaskid:=16#001,                        //设定定向接收的ID
53      bMaskExtended:=0,                       //设置0为接收标准帧，设置1为接
54      bMaskRTR:=0,                            //设置0为接收数据帧，设置1为接
55      p_CanMsg:=ADR(msg),                     //CAN帧信息引脚为指针类型，使
56      EN_OUT=>eout,                           //CAN定向接收功能块执行完成标
57      bResult=>fromNewData,                   //CAN接收到指定新消息完成标志
58      uiError=>eError);                       //CAN定向接收功能块执行错误
59
60      IF fromNewData THEN                     //如果收到新的定向消息的标志位
61
62          msg.bID:=msg.bID+2;
63          CanSend_2(en_in:=TRUE,
64          bport:=mPort,
65          p_canmsg:=ADR(msg),
66          en_out=>eout,
67          uierror=>eError);
68
69      END_IF
70
```



（6） 点击Device→扫描网络→选择对应设备→确定




（7） 下载程序，点击图标



（8） 会出现如下弹窗，点击“是”




（9） 点击运行，图标如下



# 附录 A：Modbus 协议简介

Modbus通信协议是由Modicon公司开发的应用在PLC或其他工业控制器上的一种通用语言。通过此协议，各控制器之间可以实现串行通信，Modbus通信协议定义了一个控制器能识别使用的消息结构，描述了主控制器访问从站设备的过程，例如规定从站怎样做出应答响应，检查和报告传输错误等。Modbus协议的通信方式为主从方式。主站首先向从站设备发送通信请求指令，从节点根据请求指令中的功能码向主站发回回答数据。网络中的每个从站设备都必须分配给一个唯一的地址，最多可达31个从站设备。通过多达24种总线命令实现主控制器与从站设备之间的信息交换。从站设备只执行发给自己的指令，对于其它从站地址开头的报文不作应答。这种一问一答的通信模式，大大提高了通信的正确率。因其具有操作简

单、高效、通信可靠等优点，Modbus协议已成为一个国际通信标准，得到了国际上大多数工控产品生产厂家的支持。该通信协议已广泛应用于机械、水利、电力、环保等行业设备中。

Modbus TCP通信协议可供自动化设备的监控使用。常见的应用是开发基于该协议的网关，通过网关可以将PLC、I/O模块和其它总线连到以太网上。Modbus TCP是在不改变原有的Modbus协议基础上，只是将其作为应用层协议简单的移植到TCP/IP协议上。Modbus TCP协议每一个呼叫都要求一个应答。利用TCP/IP协议，通过网页的形式可以使用户界面更加友好。利用网络浏览器就可以查看企业网内部的设备运行情况。Schneider公司已经为Modbus注册了502端口，这样就可以将实时数据嵌入到网页中，通过在设备中嵌入Web服务器，就可以将Web浏览器作为设备的操作终端。但是Modbus协议本身存在一些缺陷，它不支持诸如基于对象的通信模型等一些正在被广泛采用的网络新技术，用户在使用的时候，不得不手工配置一些参数，比如信息数据类型、寄存器号等等。

## A.1 Modbus RTU 协议数据格式

Modbus 协议有ASCII(美国标准信息交换代码)和 RTU(远程终端单元)两种数据传输方式可由用户选择，但在一个 Modbus 网络上的所有设备都必须选择相同的传输模式和串口参数。其中 RTU 模式信息帧中的 8 位数据包括两个 4 位 16 进制字符，相对于 ASCII 模式表达相同的信息只需较少的位数，在相同的速率下较 ASCII 模式具有更大的数据流量。因此，在通常情况下较多使用 RTU 模式。GCAN-204 设备也采用 RTU 模式。

RTU 模式消息发送至少以 3.5 个字符间隔时间(如表A.1 的T1-T2-T3-T4)标志开始和结束，信息帧由地址域、功能域、数据域和 CRC 校验域构成，所有字符位由 16 进制 0-9、A-F 组成。整个消息帧必须作为一连续的流传输。如果在帧完成之前有超过 1.5 个字符时间的停顿时间，接受设备将刷新不完整的消息并假定

下一个字节是一个新消息的地址域。同样的，如果一个新消息在小于 3.5 个字符时间内接着前个消息开始，接收的设备将认为它是前一消息的延续。这将导致一个错误，因为在最后的 CRC 域的值不可能是正确的。

| 起始位 | 设备地址 | 功能代码 | 数据 | CRC 校验 | 结束符 |
|---|---|---|---|---|---|
| T1-T2-T3-T4 | 8Bit | 8Bit | N 个8Bit | 16Bit | T1-T2-T3-T4 |

表A.1 RTU 消息帧格式

（1）地址域

指定报文的目的地址，包括 8bit。单个设备的地址范围是 1~247。主设备通过将要联络的从设备的地址放入消息中的地址域来选通从设备。当从设备发送回应消息时，它把自己的地址放入回应的地址域中，以便主设备知道是哪一个设备作出回应。地址 0 用作广播地址，以使所有的从设备都能认识。

（2）功能域

当消息从主设备发往从设备时，功能代码域将告之从设备需要执行哪些行为。例如去读取输入的开关状态，读一组寄存器的数据内容，读从设备的诊断状态，允许调入、记录、校验在从设备中的程序等。当从设备回应时，它使用功能代码域来指示是正常回应(无误)还是有某种错误发生(称作异议回应)。对正常回应，从设备仅回应相应的功能代码。主设备应用程序得到异议的回应后，典型的处理过程是重发消息，或者诊断发给从设备的消息并报告给操作员。

（3）数据域

数据域是由两个十六进制数集合构成的，范围 00~FF。从主设备发给从设备消息的数据域包含从机执行主机功能代码中所需的参数，如处理对象的寄存器地址，要处理项的数目，域中实际数据字节数。举例说明，如果主设备需要从设备读取一组保持寄存器（功能代码 03），数据域指定了起始寄存器以及要读的寄存器数量。如果主设备写一组从设备的寄存器(功能代码 16，即 10H)，数据域则指明了要写的起始寄存器以及要写的寄存器数量，数据域的数据字节数，要写入寄存器的数据。如果没有错误发生，从设备返回的数据域包含请求的数据。如果有错误发生，此域包含一异议代码，主设备应用程序可以用来判断采取下一步行动。在某种消息中数据域可以是不存在的(0 长度)。例如，主设备要求从设备回应通信事件记录(功能代码 0B H)，从设备不需任何附加的信息。

当传送一个 **2** 个字节的数据时，高字节**(MSB)**将被首先传送，然后传送低字节**(LSB)**。这与 **DeviceNet** 的传送方式刚好相反。

（4）CRC 校验域

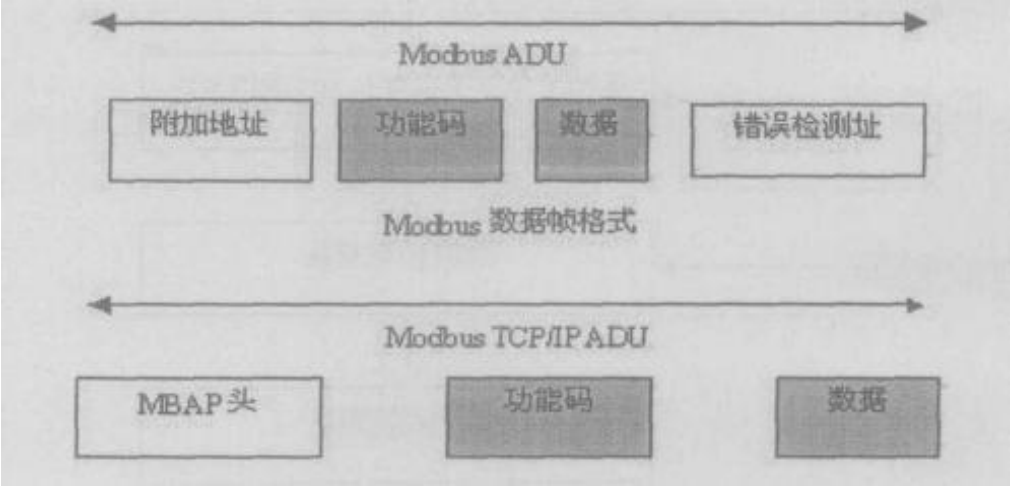
CRC 域检测整个消息的内容，包括两个字节，包含一个 16 位的二进制值。它由传输设备计算后加入到消息中。接收设备将重新计算收到消息的 CRC，并与接收到的 CRC 域中的值进行比较。如果两值不同，则有误。CRC 添加到消息中时，低字节先加入，然后是高字节。

## A.2 Modbus TCP 协议数据格式

TCP/IP 协议和以太网的链路层校验机制已可保证数据包传递的正确性，因此 Modbus TCP 报文中不再存在 CRC-16 或 LRC 校验域，但需要添加一个 Modbus 应用帧头(MBAP)。它可对 Modbus 的参数及功能进行解释。每个 TCP/IP 报文仅可含有一个 Modbus 帧。

在 Modbus TCP ADU 中，MBAP 头部占 7 个字节（含 4 个子域），及交易标识符 TI(Transaction Identifier)、协议标识符 PI(Protocol Identifier)，长度标识符 L(Length)(占用 2 字节，指明 Protocol Identifier 和 Data 域的总长度)和单元标识符 UI(Unit Identifier)组成。TI 占用 2 字节，用来标识 Modbus 帧的次序，PI 占用 2 字节，用于确认应用层协议。UI 占 1 字节，用于标识 Modbus 设备单元。功能码占1 字节，可分为位操作和 16 位字操作两类。功能码指出要进行的操作，如功

能码 15 代表写多个位寄存器，功能码 06 表示对独立的 16 位字寄存器进行写操作。数据域最多可达 248 字节，其具体格式与功能码相关。当客户机发送请求数据时，数据域给出要操作的寄存器的起始地址（2 字节）和个数（1 字节）；当服务器发送应答数据时，数据域给出被操作的寄存器个数（1 字节）及各寄存器状态值。图 A.1 给出了 Modbus 与 Modbus TCP 数据帧格式比较。




图A.1 Modbus 与Modbus TCP/IP 帧格式

Modbus TCP 的ADU 数据单元规范如表 A.2 所示。

| | 描述 | 所占字节 |
|---|---|---|
| **MBAP 头** | 传输标识码高位 Hi | 1 |
| | 传输标识码低位 Lo | 1 |
| | 协议标识符 | 2 |
| | 长度标识符 | 2 |
| | 单元标识符 | 1 |
| **Modbus 请求** | 功能码 | 1 |
| | 开始地址 | 2 |
| | 寄存器数目 | 2 |

表A.2 Modbus TCP 的ADU 数据单元规范

在通过 Modbus TCP 传送数据之前，需要在客户机和服务器之间建立一个 TCP/IP 连接。服务器使用端口 502 作为 Modbus TCP 的连接端口。Modbus TCP 连接的建立通常由 TCP/IP Socket 接口的软件协议自动实现，因此对应用完全透明。一旦客户端和服务器之间的 TCP/IP 连接建立，同样的连接可以根据要求的方向用来传输任意数量的用户数据。客户端和服务器还可以同时建立多个 TCP/IP 连接，最大的连接数量取决于 TCP/IP 接口的规范。

当某一设备发出请求，则其相应的设备要做出响应。响应的数据格式如表 A.3 所示。

| 字节 | 响应数据 |
|---|---|
| Byte0、Byte1 | 传输标识码=0（响应时拷贝该数据） |
| Byte2、Byte3 | 协议标识符 |
| Byte4 | 长度标识符高字节=0 |
| Byte5 | 长度标识符低字节（标识其后有多少个字节） |
| Byte6 | 单元标识符（从设备地址） |
| Byte7 | Modbus 功能码 |
| Byte8 | 数据 |

表A.3 Modbus TCP 响应数据格式

## A.3 Modbus 常用功能码

在 Modbus 消息帧的功能码中较常使用的是 01、02、03、04、05、06 和 16 功能码，使用它们即可实现对从机的数字量和模拟量的读写操作。

Modbus 标准地址与各个功能码的对应关系如下所示。

| **Modbus** 标准地址 | 数据 | 功能码 |
|---|---|---|
| 00001-0xxxx | DO | 01、05、15 |
| 10001-1xxxx | DI | 02 |
| 30001-3xxxx | AI | 04 |
| 40001-4xxxx | 保持寄存器 | 03、06、16 |

| 功能码 | 名称 | 功能说明 |
|---|---|---|
| 01 | 读取线圈状态 | 取得一组线圈的当前状态(ON/OFF) |
| 02 | 读取输入状态 | 取得一组开关输入的当前状态(ON/OFF) |
| 03 | 读取保持寄存器 | 在一个或多个保持寄存器中取得当前的二进制值 |
| 04 | 读取输入寄存器 | 在一个或多个输入寄存器中取得当前的二进制值 |
| 05 | 强置单线圈 | 强置一个逻辑线圈的通断状态 |
| 06 | 预置单寄存器 | 把具体二进制值装入一个保持寄存器 |
| 07 | 读取异常状态 | 取得 8 个内部线圈的通断状态 |
| 08 | 回送诊断校验 | 把诊断校验报文送从机，通信诊断 |
| 16 | 预置多寄存器 | 把具体二进制值装入一串连续的保持寄存器 |
| 128~255 | 保留 | 用于异常应答 |

下面以在 RTU 传输模式下通讯为例，对这些功能码进行详细介绍。

下面是 7 个 Modbus RTU 命令的主从机收发的数据包格式，其余的命令可参照其格式。

（1）功能码：01H

代码功能：读取线圈状态（DO）

说明：读取从机 DO 的 ON/OFF 状态，不支持广播。

查询：查询信息规定了要读的起始线圈地址和线圈量，线圈的起始地址为 0000H，1-16 个线圈的寻址地址分为 0000H-0015H。

| 主机发送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 01 | 读取线圈状态 |
| 线圈首地址 | 2 字节 | 00 00 | 线圈首址为 0000H |
| 线圈数量 | 2 字节 | 00 08 | 连续读 8 个线圈 |
| CRC | 2 字节 | 3D CC | 前6 个字节的 CRC 校验码 |

响应：响应信息中的各线圈的状态与数据区的每一位的值相对应，即每个 DO 占用一位（1 = ON, 0 = OFF ）。数据区从高位到低位依次为 DO7、DO6.....DO0。

| 从机回送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 01 | 读取线圈状态 |
| 数据字节数 | 1 字节 | 01 | 1 个字节 |
| 数据 | 1 字节 | 02 | 二进制为 0000 0010，DO1 为ON |
| CRC | 2 字节 | D0 49 | 前4 个字节的 CRC 校验码 |

（2）功能码：02H

代码功能：读取输入状态（DI）

说明：读取从机 DI 的 ON/OFF 状态，不支持广播。

查询：查询信息规定了要读的输入起始地址及输入信号的数量，输入寻址起始地址为 0000H，输入 1-16 所对应的地址分别为 0-15。

| 主机发送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 02 | 读取输入状态 |
| 输入首地址 | 2 字节 | 00 00 | 输入首址为 0000H |
| 寄存器数量 | 2 字节 | 00 08 | 连续读 8 个输入口 |
| CRC | 2 字节 | 79 CC | 前6 个字节的 CRC 校验码 |

响应：响应信息中的各输入口的状态与数据区的每一位的值相对应，即每个 DI 占用一位（1 = ON, 0 = OFF ）。数据区从高位到低位依次为DI7、DI6.....DI0。

| 从机回送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 02 | 读取输入状态 |
| 数据字节数 | 1 字节 | 01 | 1 个字节 |
| 数据 | 1 字节 | 81 | 二进制为 1000 0001，DI7 与 DI0 为 ON |
| CRC | 2 字节 | 61 E8 | 前4 个字节的 CRC 校验码 |

（3）功能码：03H

代码功能：读取保持寄存器

说明：读从机保持寄存器的二进制数据，不支持广播。

查询：查询信息规定了要读的寄存器起始地址及寄存器的数量，寄存器寻址起始地址为 0000H，寄存器 1-16 所对应的地址分别为 0-15。

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 03 | 读取保持寄存器数据 |
| 寄存器首地址 | 2 字节 | 00 01 | 寄存器首址为 0001H |
| 寄存器数量 | 2 字节 | 00 03 | 连续读 3 个寄存器 |
| CRC | 2 字节 | 54 0B | 前6 个字节的 CRC 校验码 |

响应：响应信息中的寄存器数据为二进制数据，每个寄存器分别对应 2 个字节，第一个字节为高位值数据，第二个字节为低位数据。

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 03 | 读取保持寄存器数据 |
| 数据字节数 | 1 字节 | 06 | 3 个寄存器占 6 个字节 |
| 数据 1 | 2 字节 | 02 0B | 0001H 寄存器中的数据 |
| 数据 2 | 2 字节 | 00 00 | 0002H 寄存器中的数据 |
| 数据 3 | 2 字节 | 00 64 | 0003H 寄存器中的数据 |
| CRC | 2 字节 | 84 BD | 前9 个字节的 CRC 校验码 |

（4）功能码：04H

代码功能：读取输入寄存器（AI）

说明：读取从机输入寄存器(3X 类型)中的二进制数据，不支持广播。

查询：查询信息规定了要读的寄存器起始地址及寄存器的数量，寄存器寻址起始地址为 0000H，寄存器 1-16 所对应的地址分别为 0-15。

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 04 | 读取输入寄存器数据 |
| 寄存器首地址 | 2 字节 | 00 00 | 寄存器首址为 0000H |
| 寄存器数量 | 2 字节 | 00 01 | 连续读 1 个寄存器 |
| CRC | 2 字节 | 31 CA | 前6 个字节的 CRC 校验码 |

响应：响应信息中的寄存器数据为二进制数据，每个寄存器分别对应 2 个字节，第一个字节为高位值数据，第二个字节为低位数据。

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 04 | 读取输入寄存器数据 |
| 数据字节数 | 1 字节 | 02 | 1 个寄存器占 2 个字节 |
| 数据 1 | 2 字节 | 0F FB | 0000H 寄存器中的数据 |
| CRC | 2 字节 | FD 43 | 前5 个字节的 CRC 校验码 |

（5）功能码：05H

代码功能：强置单线圈（DO）

说明：强制单个线圈(DO，0X 类型)为 ON 或 OFF 状态，广播时，该功能可

强制所有从机中同一类型的线圈均为 ON 或 OFF 状态。

查询：查询信息规定了需要强制线圈的地址及状态，线圈的起始地址为 0000H，寄存器 1-16 所对应的地址分别为 0-15。查询时，由查询数据区中的一个常量，规定被请求线圈的 ON/OFF 状态，FF00H 值请求线圈处于 ON 状态，0000H 值请求线圈处于 OFF 状态，其它值对线圈无效，不起作用。

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 05 | 强置单线圈 |
| 线圈地址 | 2 字节 | 00 01 | 线圈地址为 0001H |
| 线圈状态值 | 2 字节 | FF 00 | ON 状态 |
| CRC | 2 字节 | DD FA | 前6 个字节的 CRC 校验码 |

响应：对这个命令请求的正常响应是在 DO 状态改变以后，原样传送接收到的数据。

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 05 | 强置单线圈 |
| 线圈地址 | 2 字节 | 00 01 | 线圈地址为 0001H |
| 线圈状态值 | 2 字节 | FF 00 | ON 状态 |
| CRC | 2 字节 | DD FA | 前6 个字节的 CRC 校验码 |

（6）功能码：06H

代码功能：预置单寄存器

说明：把一个值预置到一个保持寄存器（4X 类型）中，广播时，该功能把值预置到所有从机相同类型的寄存器中。该功能可越过控制器的内存保护。使寄存器中的预置值保持有效。只能由控制器的下一个逻辑信号来处理该预置值。若控制逻辑中无寄存器程序时，则寄存器中的值保持不变。

查询：查询信息规定了要预置寄存器的类型，寄存器寻址起始地址为 0000H，寄存器 1-16 所对应的地址分别为 0-15。

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 06 | 读寄存器数据 |
| 寄存器地址 | 2 字节 | 00 03 | 预置寄存器地址为 0003H |
| 寄存器的值 | 2 字节 | AB CD | 将该值预置到寄存器中 |
| CRC | 2 字节 | C7 6F | 前6 个字节的 CRC 校验码 |

响应：对这个命令请求的正常响应是在寄存器值状态改变以后，原样传送接收到的数据。

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 06 | 读寄存器数据 |
| 寄存器地址 | 2 字节 | 00 03 | 预置寄存器地址为 0003H |
| 寄存器的值 | 2 字节 | AB CD | 将该值预置到寄存器中 |
| CRC | 2 字节 | C7 6F | 前6 个字节的 CRC 校验码 |

（7）功能码：**10H**（十进制为 **16**）

代码功能：预置多个寄存器

说明：把数据按顺序预置到各(4x 类型)寄存器中，广播时该功能代码可把数据预置到全部从机中的相同类型的寄存器中。需要注意的是该功能代码可越过控制器的内存保护，在寄存器中的预置值一直保持有效，只能由控制器的下一个逻辑来处理寄存器的内容，控制逻辑中无该寄存器程序时，则寄存器中的值保持不变。

查询：信息中规定了要预置的寄存器类型，寄存器寻址的起始地址为 0。查询数据区中指定了寄存器的预置值，M 84 和 484 型控制器使用 10 位二进制数据，2 个字节，剩余的高 6 位置 0。而其他类型的控制器使用一个 16 位二进制数据，每个寄存器 2 个字节。

| 主机发送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 10 | 预置多个寄存器 |
| 寄存器首地址 | 2 字节 | 10 20 | 写入寄存器首址为 1020H |
| 寄存器数量 | 2 字节 | 00 03 | 连续 3 个寄存器 |
| 字节数 | 1 字节 | 06 | 3 个寄存器占 6 个字节 |
| 数据 1 | 2 字节 | 02 01 | 寄存器 1020H 中的数据 |
| 数据 2 | 2 字节 | 04 03 | 寄存器 1021H 中的数据 |
| 数据 3 | 2 字节 | 06 05 | 寄存器 1022H 中的数据 |
| CRC | 2 字节 | BD 9B | 前13 个字节的 CRC 校验码 |

响应：正常响应返回从机地址、功能代码、起始地址和预置寄存器数。

| 从机回送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 10 | 写寄存器数据 |
| 寄存器首地址 | 2 字节 | 10 20 | 写入寄存器首址为 1020H |
| 寄存器数量 | 2 字节 | 00 03 | 连续 3 个寄存器 |
| CRC | 2 字节 | 85 02 | 前6 个字节的 CRC 校验码 |

下面是 7 个 **Modbus TCP** 命令的主从机收发的数据包格式，其余的命令可参照其格式。本部分略去代码功能及说明，相关内容请参考 Modbus RTU 部分。

（1） 功能码：**01H**

| 主机发送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | 其后有 6 个字节 |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 01 | 读取线圈状态 |
| 线圈首地址 | 2 字节 | 00 00 | 线圈首址为 0000H |
| 线圈数量 | 2 字节 | 00 08 | 连续读 8 个线圈 |

| 从机回送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 04 | 其后有 4 个字节 |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 01 | 读取线圈状态 |
| 数据字节数 | 1 字节 | 01 | 1 个字节 |
| 数据 | 1 字节 | 02 | 二进制为 0000 0010，DO1 为ON |

（2） 功能码：**02H**

| 主机发送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 02 | 读取输入状态 |
| 输入首地址 | 2 字节 | 00 00 | 输入首址为 0000H |
| 寄存器数量 | 2 字节 | 00 08 | 连续读 8 个输入口 |

| 从机回送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 04 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 02 | 读取输入状态 |
| 数据字节数 | 1 字节 | 01 | 1 个字节 |
| 数据 | 1 字节 | 81 | 二进制为 1000 0001，DI7 与 DI0 为 ON |

（3） 功能码：**03H**

| 主机发送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 03 | 读取保持寄存器数据 |
| 寄存器首地址 | 2 字节 | 00 01 | 寄存器首址为 0001H |
| 寄存器数量 | 2 字节 | 00 03 | 连续读 3 个寄存器 |

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 09 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 03 | 读取保持寄存器数据 |
| 数据字节数 | 1 字节 | 06 | 3 个寄存器占 6 个字节 |
| 数据1 | 2 字节 | 02 0B | 0001H 寄存器中的数据 |
| 数据2 | 2 字节 | 00 00 | 0002H 寄存器中的数据 |
| 数据3 | 2 字节 | 00 64 | 0003H 寄存器中的数据 |

（4） 功能码：**04H**

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 04 | 读取输入寄存器数据 |
| 寄存器首地址 | 2 字节 | 00 00 | 寄存器首址为 0000H |
| 寄存器数量 | 2 字节 | 00 01 | 连续读 1 个寄存器 |

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 05 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 04 | 读取输入寄存器数据 |
| 数据字节数 | 1 字节 | 02 | 1 个寄存器占 2 个字节 |
| 数据1 | 2 字节 | 0F FB | 0000H 寄存器中的数据 |

（5） 功能码：**05H**

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 05 | 强置单线圈 |
| 线圈地址 | 2 字节 | 00 01 | 线圈地址为 0001H |
| 线圈状态值 | 2 字节 | FF 00 | ON 状态 |

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 05 | 强置单线圈 |
| 线圈地址 | 2 字节 | 00 01 | 线圈地址为 0001H |
| 线圈状态值 | 2 字节 | FF 00 | ON 状态 |

（6） 功能码：**06H**

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 06 | 读寄存器数据 |
| 寄存器地址 | 2 字节 | 00 03 | 预置寄存器地址为 0003H |
| 寄存器的值 | 2 字节 | AB CD | 将该值预置到寄存器中 |

| 从机回送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 06 | 读寄存器数据 |
| 寄存器地址 | 2 字节 | 00 03 | 预置寄存器地址为 0003H |
| 寄存器的值 | 2 字节 | AB CD | 将该值预置到寄存器中 |

（7） 功能码：**10H**（十进制为 **16**）

| 主机发送 | 字节数 | 例（Hex） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 0D | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 10 | 预置多个寄存器 |
| 寄存器首地址 | 2 字节 | 10 20 | 写入寄存器首址为 1020H |
| 寄存器数量 | 2 字节 | 00 03 | 连续 3 个寄存器 |
| 字节数 | 1 字节 | 06 | 3 个寄存器占 6 个字节 |
| 数据 1 | 2 字节 | 02 01 | 寄存器 1020H 中的数据 |
| 数据 2 | 2 字节 | 04 03 | 寄存器 1021H 中的数据 |
| 数据 3 | 2 字节 | 06 05 | 寄存器 1022H 中的数据 |

| 从机回送 | 字节数 | 例（**Hex**） | 注释 |
|---|---|---|---|
| 传输标识 | 2 字节 | 00 00 | |
| 协议标识 | 2 字节 | 00 00 | |
| 数据长度 | 2 字节 | 00 06 | |
| 从机地址 | 1 字节 | 01 | 与01 号从机通信 |
| 功能码 | 1 字节 | 10 | 写寄存器数据 |
| 寄存器首地址 | 2 字节 | 10 20 | 写入寄存器首址为 1020H |
| 寄存器数量 | 2 字节 | 00 03 | 连续 3 个寄存器 |

# 10.免责声明

感谢您购买广成科技的 GCAN 系列软硬件产品。GCAN 是沈阳广成科技有限公司的注册商标。本产品及手册为广成科技版权所有。未经许可，不得以任何形式复制翻印。在使用之前，请仔细阅读本声明，一旦使用，即被视为对本声明全部内容的认可和接受。请严格遵守手册、产品说明和相关的法律法规、政策、准则安装和使用该产品。在使用产品过程中，用户承诺对自己的行为及因此而产生的所有后果负责。因用户不当使用、安装、改装造成的任何损失，广成科技将不承担法律责任。

关于免责声明的最终解释权归广成科技所有。

# 销售与服务

沈阳广成科技有限公司

地址：辽宁省沈阳市浑南区长青南街 135-21 号5 楼

邮编：110000

网址：www.gcgd.net

淘宝官方店：https://shop72369840.taobao.com/

天猫官方店：https://gcan.tmall.com/

京东官方店：https://mall.jd.com/index-684755.html

官方服务热线：13019325660

售前服务电话与微信号：15712411229

售前服务电话与微信号：18309815706

售前服务电话与微信号：18940207426

售后服务电话与微信号：15566084645

售后服务电话与微信号：18609810321

售后服务电话与微信号：17602468871