GCAN-PLCcore-M7

PLC核心模块及开发套件

用户手册

文档版本：V1.03（2018/01/04）

修订历史

版本	日期	原因
V1.00	2017/06/16	创建文档
V1.02	2017/10/17	修正设备工作参数
V1.03	2018/01/04	新增部分例程说明

目 录

1 功能简介

1.1 功能概述

GCAN-PLCcore-M7 模块为您提供了一个全新的高性能 PLC 内核。由于其具有体积小、性能高和低功耗的特性，因此非常适合作为 PLC 核心模块，应用在工业通信领域和控制领域。请您花些时间仔细阅读本手册。本手册中包含 GCAN-PLCcore-M7 调试、配置和编程等重要信息。它将帮助你熟悉 GCAN-PLCcore-M7 的功能范围和使用方法。本文档的补充材料有 OpenPCS IEC 61131 编程系统和基于 IEC 61131-3 的 CANopen 总线扩展。

广成科技 GCAN-PLCCore-M7 核心板的板载资源十分丰富，可以满足各种应用的需求，可以独立使用。整个核心板的外形尺寸为 65mm*45mm 大小，非常小巧。除此之外，GCAN-PLCCore-M7 采用了贴片板对板连接器，使其可以方便的应用在各种项目上。

1.2 性能特点

GCAN-PLCCore-M7 核心板的板载资源如下：

- CPU：ARM® 32-bit Cortex®-M7 内核，处理器频率 216MHz；
- SDRAM：256MB；
- NAND FLASH：1GB；
- SPI FLASH：256MB；
- EEPROM：2KB；
- 双 60pin 插座（在底部），方便接入各种底板；
- 1 个 5V&3.3V 测试点，支持外接电源或输出电源给外部；
- 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 和 LCD；
- 1 个功能按钮，IAP，可用于更新固件。

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 产品描述

GCAN-PLCCore-M7 作为一个小尺寸可插入式的核心模块，显着地降低了用户开发特定控件时的工作量和成本，且可定制内核可以完成个性化的应用。GCAN-PLCCore-M7 非常适合作为过程信号处理、集中监控、智能网络节点、分布式控制使用。此外，它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-M7 的板载固件包含整个 PLC 的运行环境，其中包括 CANopen 主站的功能连接。因此，该模块能够执行控制任务，例如连接和输出或转换规则算法。数据和事件可以通过 CANopen 网络、以太网（UDP 协议）和串行接口（UART）与其他节点（例如高级主控制器、输入/输出等）进行交换。此外，通过 CANopen 设备，输入输出的数量可实现局部扩展。它还可以作为设备的即插即用核心模块，应用到用户特定的应用程序中。

GCAN-PLCCore-M7 提供多达 4 路 UART，2 路 CAN，1 路 100M 以太网，8 路 AD，2 路 DA，40 个数字输入、输出（3.3V 电平），2 个高速计数器输入和 2 路 PWM 输出（3.3 V 振幅）。用户将 PLC 程序保存在模块的闪存盘上，可以保证模块在电源故障时能够自动重启。

GCAN-PLCCore-M7 模块内部集成 PLCOpen 的实时内核，用户可以使用 OpenPCS 软件对其进行编程和开发，支持标准 IEC61131-3 编程语言，包括 5 种语言程序：FBD、SFC、LD、ST、IL。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。

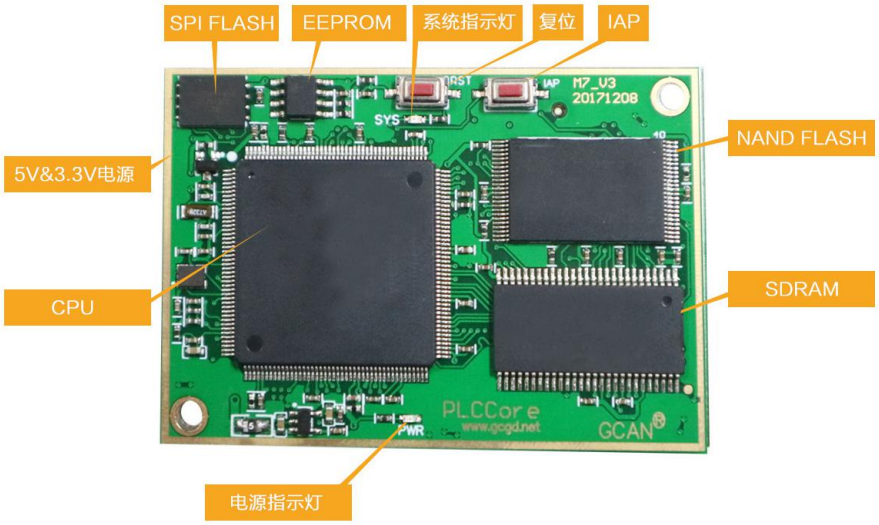


图 2.1 GCAN-PLCCore-M7 核心板资源

3 GCAN-PLCcore-M7 引脚定义及功能说明

下表中对 GCAN-PLCcore-M7 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。

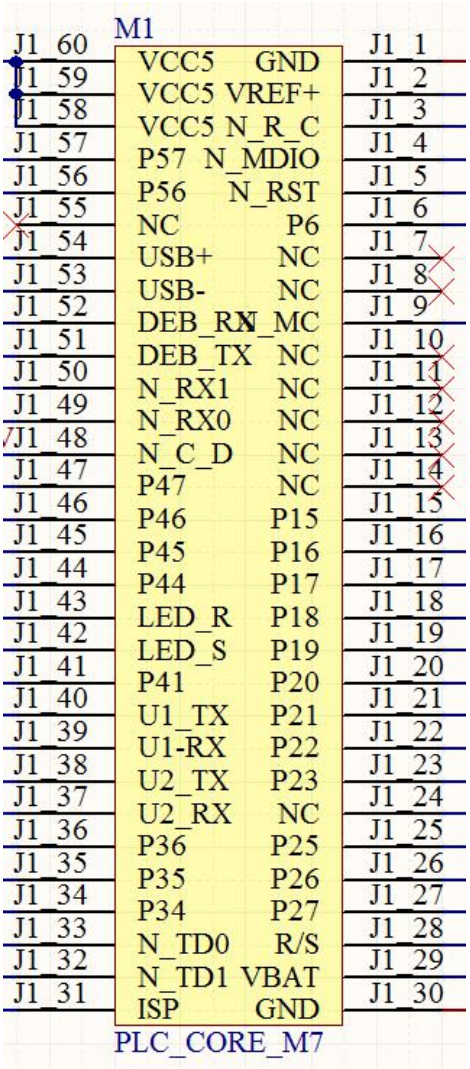
引脚序号	引脚编号	功能定义	引脚序号	引脚编号	功能定义
CON1			CON2		
1	P1	GND	1	P61	IO/ETH_RMII_TX_EN
2	P2	VREF+	2	P62	IO/PWM
3	P3	IO/RMII_REFF_CLK	3	P63	IO
4	P4	IO/ETH_MDIO	4	P64	IO
5	P5	IO/ETH_RESET	5	P65	IO
6	P6	IO	6	P66	IO/PWM
7	P7	x	7	P67	IO/PWM
8	P8	x	8	P68	IO/PWM
9	P9	IO/ETH_MDC	9	P69	IO
10	P10	RST	10	P70	IO/PWM
11	P11	IO/AD	11	P71	IO/PWM
12	P12	x	12	P72	IO/PWM
13	P13	x	13	P73	IO
14	P14	x	14	P74	IO
15	P15	IO	15	P75	IO/UART3_DE
16	P16	IO/PWM	16	P76	IO/UART3_TX
17	P17	IO/PWM	17	P77	IO/UART3_RX
18	P18	IO	18	P78	IO/PWM
19	P19	IO	19	P79	IO/PWM
20	P20	IO/UART4_DE	20	P80	IO/PWM
21	P21	IO/UART4_RX/CAN3_RX	21	P81	x
22	P22	IO/UART4_TX/CAN3_TX	22	P82	x
23	P23	IO/CAN2_RX	23	P83	x
24	P24	IO/CAN2_TX	24	P84	IO
25	P25	IO/PWM	25	P85	IO

26	P26	IO/PWM	26	P86	x
27	P27	IO/PWM	27	P87	x
28	P28	RUN/STOP	28	P88	IO
29	P29	VBAT	29	P89	IO
30	P30	GND	30	P90	IO
31	P31	x	31	P91	GND
32	P32	IO/ETH_RMII_TXD1	32	P92	IO
33	P33	IO/ETH_RMII_TXD0	33	P93	IO
34	P34	IO	34	P94	IO
35	P35	IO	35	P95	IO
36	P36	IO/UART2_DE	36	P96	IO
37	P37	IO/UART2_RXD	37	P97	IO
38	P38	IO/UART2_TXD	38	P98	IO
39	P39	IO/UART1_RXD	39	P99	IO
40	P40	IO/UART1_TXD	40	P100	IO
41	P41	IO/UART1_DE	41	P101	IO
42	P42	SYS_LED	42	P102	IO/AD
43	P43	RUN_LED	43	P103	x
44	P44	IO/AD	44	P104	x
45	P45	IO/AD/DA	45	P105	x
46	P46	IO/AD/DA	46	P106	x
47	P47	IO/AD	47	P107	x
48	P48	IO/ETH_RMII_CRS_DV	48	P108	x
49	P49	IO/ETH_RMII_RXD0	49	P109	x
50	P50	IO/ETH_RMII_RXD1	50	P110	x
51	P51	PLC_DEBUG_TX	51	P111	x
52	P52	PLC_DEBUG_RX	52	P112	x
53	P53	IO/CAN1_RX	53	P113	x
54	P54	IO/CAN2_TX	54	P114	x
55	P55	x	55	P115	x

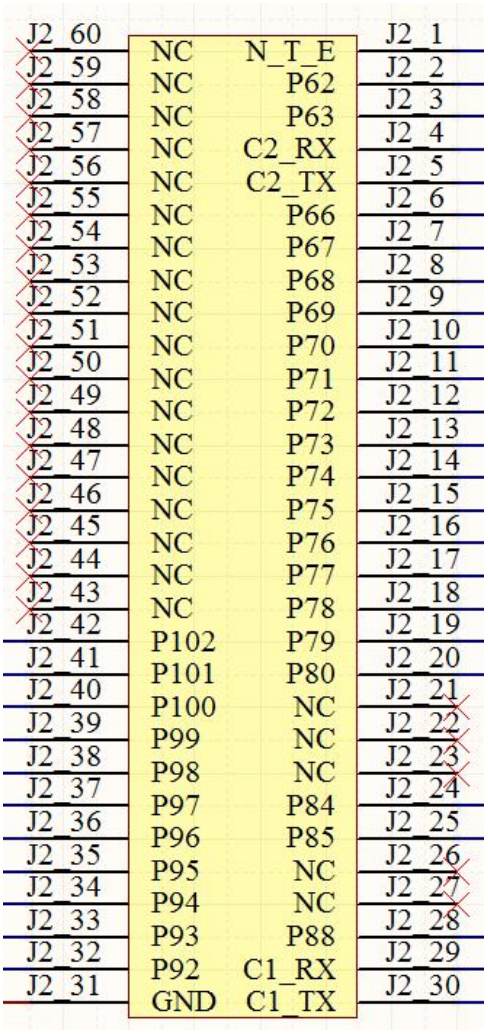
56	P56	IO/AD	56	P116	x
57	P57	IO/AD	57	P117	x
58	P58	VCC5	58	P118	x
59	P59	VCC5	59	P119	x
60	P60	VCC5	60	P120	x

GCAN-PLCcore-M7 核心板连接器的引脚定义原理图如下所示：

CON1：



CON2：



4 GCAN-PLCcore-M7 启动策略

默认情况下，GCAN-PLCcore-M7 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-M7 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：

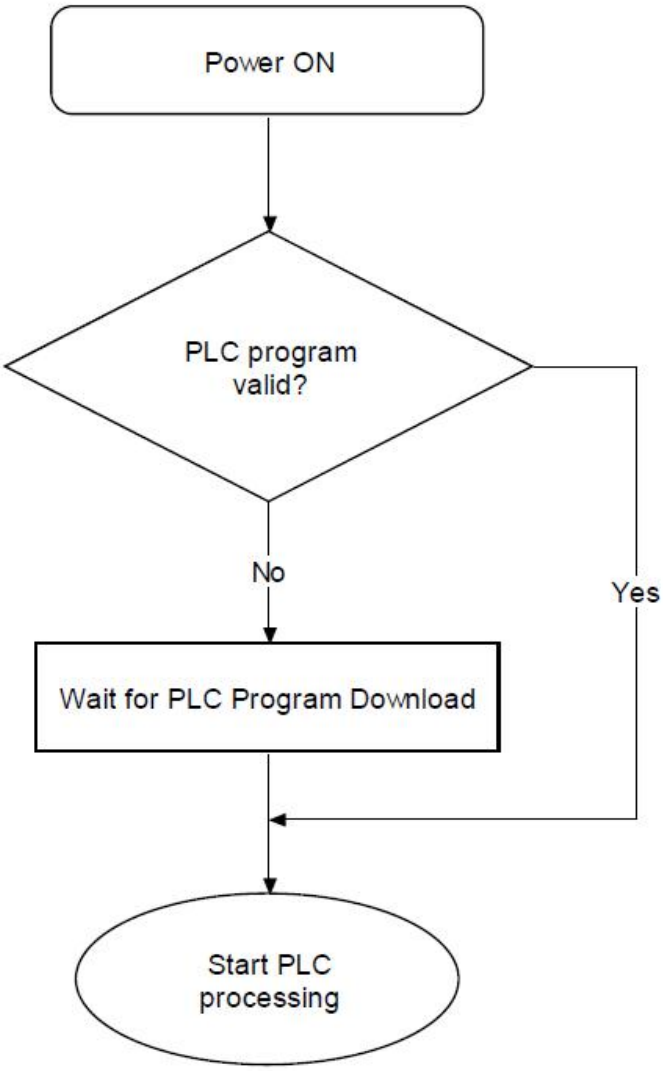


图 4.1 GCAN-PLCcore-M7 启动策略

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-M7 停止模式。

5 GCAN-PLCCore-M7 核心板参数表

控制器特点	
控制器	STM32F7
内核	ARM® 32-bit Cortex®-M7
处理器频率	216MHz
PLC程序存储空间	256MB
RAM	256MB
数据存储器空间	1GB
接口特点	
以太网	1路，10/100Mbps
RS232/RS485	4路，600bps~115200bps
CAN接口	2路，遵循ISO 11898标准，支持CAN2.0A/B
数字量输入/输出	50路DI/DO（复用）
模拟量输入	6路（复用）
模拟量输出	2路（复用）
板载接口	双60pin插座，0.8mm管脚间距
软件	内部集成IEC61131-3实时内核
编程接口	RS232、TCP（可选）、CAN（可选）
RTC	板载实时时钟，需要外接时钟电池
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	EN 55024:2011-09 EN 55022:2011-12
防护等级	IP 20
基本信息	
外形尺寸	65mm *45mm
重量	10g

6 PLCCore-EDV-M7 开发套件

PLCCore-EDV-M7 开发套件是一套高性能低成本的开发套件。基于 GCAN-PLCCore-M7 核心板，开发套件 PLCCore-EDV-M7 能使用户快速完成分散、网络相兼容的自动化项目。此外，它有助于用户了解基于 IEC 61131-3 PLC 编程相比传统编程语言的优点。

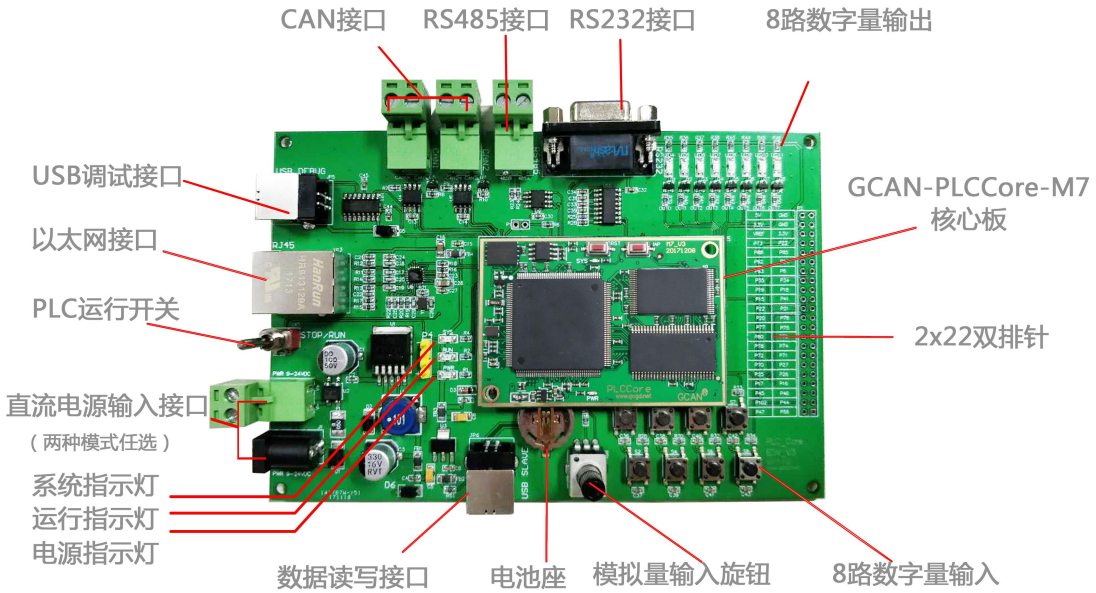


图 6.1 PLCCore-EDV-M7 开发套件

GCAN-PLCCore-M7 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-M7 核心板；
- 1 个电源指示灯，2 个状态指示灯；
- 2 路 CAN 接口，1 路以太网接口；
- 1 路 RS232 接口，1 路 RS485 接口；
- 8 路数字量输入，8 路数字量输出；
- 1 路 USB 串口，可用于 PLC 程序下载和代码调试；
- 1 路 USB Slave，可用于 PLC 内用户数据读取与写入；
- 1 个直流电源输入接口（输入电压范围：DC 9-30V）；
- 1 个 RTC 后备电池座；
- 1 个开关，用于 PLC 的 RUN 和 STOP；
- 1 组 2x22 的双排针，用于引出核心板内其余的引脚；
- 引出 2x22 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。

7 PLCCore-EDV-M7 开发套件参数

PLCCore-EDV-M7 开发套件（GCAN-PLCCore-M7 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-M7
内核	ARM® 32-bit Cortex®-M7
接口特点	
以太网	1路，10/100Mbps
RS232	1路，600bps~115200bps
RS485	1路，600bps~115200bps
CAN接口	2路，遵循ISO 11898标准，支持CAN2.0A/B
CAN波特率	1000K、500K、250K、200K、125K、100K、50K、20K
数字量输入	8路DI，按键输入
数字量输出	8路DO，LED指示灯
板载接口	双60pin插座，0.8mm管脚间距
编程接口	RS232
RTC	板载实时时钟，需要外接时钟电池
供电电源	
供电电压	+9-30V DC
环境试验	
工作温度	-40℃~+85℃
工作湿度	15%~90%RH，无凝露
EMC测试	EN 55024:2011-09 EN 55022:2011-12
防护等级	IP 20
基本信息	
外形尺寸	150mm *100mm
重量	100g

8 PLCCore-EDV-M7 开发套件基本例程

PLCCore-EDV-M7 开发套件提供了如下这些例程代码，通过这些例程用户可以快速熟悉 GCAN-PLCCore-M7 的使用及其特性，用户可以快速开发自己的项目。

实验 1 跑马灯实验
实验 2 输入输出实验
实验 3 串口实验
实验 4 CAN 通信实验
实验 5 TCP Server 实验

8.1 实验 6 Modbus TCP 实验

本实验用于实现 GCAN-PLCCore 的 Modbus TCP 从站功能，涉及的功能码主要包含 02 03 04 05。Modbus TCP 的数据格式详见附录“表 B.2 Modbus TCP 响应数据格式”。

A.下载程序

请参照“**9.3.4 设置调试连接**”及“**9.3.5 下载程序并调试**”完成程序的下载。下载完程序之后，您需要点击一下 GCAN-PLCCore 上面的 RESET 键或重新给 GCAN-PLCCore 开发板上电，使新的程序生效。

B.主要功能块说明

本实验中使用的功能块有：LAN_INIT、LAN_ASCII_TO_INET、Modbus_TCP_SLAVE_INIT、Modbus_SLAVE_CTRL 四个功能块。它们的作用如下表所示。功能块原型、操作数定义及描述等信息请参照“GCAN-PLC 扩展功能块说明书”。

功能块	作用
LAN_INIT	用于初始化以太网接口。
LAN_ASCII_TO_INET	将 IP 地址的 ASCII 码形式转换成 IP 地址的整数形式。
Modbus_TCP_SLAVE_INIT	用于初始化 Modbus TCP 从站接口。
Modbus_SLAVE_CTRL	用于执行 Modbus TCP 从站接口的控制指令。

表 8.1 Modbus TCP 实验中使用的功能块

C.程序段说明

打开 TEST.ST 文件，可查看到如下代码：

```
if lanInit=false then

mIP := LAN_ASCII_TO_INET('192.168.1.31');
mNetMask := LAN_ASCII_TO_INET('255.255.0.0');
mGateWay := LAN_ASCII_TO_INET('192.168.1.1');
    inst0_LAN_INIT(
            ENABLE :=true ,
            HOSTNAME := 'PLCCore',
            IP := mIP,
            NETMASK :=mNetMask,
            GATEWAY :=mGateWay,
```

```
                NETNUMBER:=1|mConfirm:= CONFIRM,
                mError:= ERROR,
                mErrorinfo:= ERRORINFO
                 );
        lanInit:=true;
        else
            if mModbusInitOK=false then

                inst0_MODBUS_TCP_SLAVE_INIT(
                ENABLE := true,
                MODE := 1,
                ADDRESS :=1 ,
                NETNUMBER :=1   |   mConfirm:= CONFIRM,
                mError:= ERROR,
                mErrorinfo:= ERRORINFO
                );
                mModbusInitOK:=true;
            end_if;
        end_if;
```

这段代码用于初始化以太网接口和 Modbus TCP 从站接口。用户可通过设置 IP、NETMASK、GATEWAY 这三个参数，建立以太网连接。可通过将 MODE 设置为 1 将 Port 定义为 Modbus 协议并使能该协议，通过将 MODE 设置为 0 将 Port 定义为 PPI 并禁止 Modbus 协议；通过 ADDRESS 参数设定 TCP 端的本站地址。

```
VAR
lanInit:bool:=false;
inst0_LAN_INIT:LAN_INIT;
mConfirm:bool;
mError:usint;
mErrorinfo:usint;
mIP:udint;
mNetMask:udint;
mGateWay:udint;
mSocket:INT;
inst0_MODBUS_TCP_SLAVE_INIT:MODBUS_TCP_SLAVE_INIT;
mModbusInitOK:bool:=false;
modbusDOBuf : ARRAY[0..5] OF byte;
DO_Ptr : POINTER;
modbusDIBuf : ARRAY[0..5] OF byte;
DI_Ptr : POINTER
modbusAIBuf : ARRAY[0..10] OF int;
AI_Ptr : POINTER;
modbusRegBuf : ARRAY[0..127] OF int;
mPtr : POINTER;
xDONE:BOOL;
```

```
inst4_MODBUS_TCP_SLAVE_CTRL:MODBUS_TCP_SLAVE_CTRL;
xError:usint;
mDI at %I0.0:byte;
mAI at %I1.0:int;
mDO      at %Q0.0:byte;
END_VAR

mDO:=modbusDOBuf[0];
modbusDIBuf[0]:=mDI;
modbusAIBuf[0]:=mAI;
modbusRegBuf[0]:=byte_to_int(mDI);   (*此处定义 modbus 数字量输入值存放在*)
                                      (*Modbus 保持寄存器中的地址*)
modbusRegBuf[1]:=mAI;                 (*模拟量输入值*)
modbusRegBuf[2]:=byte_to_int(mDO);    (*数字量输出值*)

mPtr:=&modbusRegBuf;
DO_Ptr:=&modbusDOBuf;
DI_Ptr:=&modbusDIBuf;
AI_Ptr:=&modbusAIBuf;

if mModbusInitOK=true then
    inst4_MODBUS_TCP_SLAVE_CTRL(
        NETNUMBER :=1,
        DO_ENABLE :=1 ,
        DO_PTR :=DO_Ptr ,
        DO_LENGTH :=5 ,                (*请注意其长度限制*)
        DI_ENABLE :=1 ,
        DI_PTR :=DI_Ptr ,
        DI_LENGTH :=5 ,
        AI_ENABLE :=1 ,
        AI_PTR :=AI_Ptr ,
        AI_LENGTH :=10 ,
        REG_ENABLE :=1 ,
        REG_PTR :=mPtr ,
        REG_LENGTH :=127
        | xDONE := DONE,
        xError := ERROR);
end_if;
```

这段代码用于执行 Modbus TCP 从站接口的控制指令。modbusDIBuf 为输入寄存器（数字量），modbusAIBuf 为输入寄存器（模拟量），modbusRegBuf 为保持寄存器。modbusRegBuf 共包含 127 个地址，您可以通过赋值语句将不同的输入输出值存放到 Modbus 保持寄存器中的地址。如“modbusRegBuf[3]:=byte_to_int(mDO); ”，将数字量输出值存放到 Modbus 寄存器中的 3 号地址。

D.实际测试

您可以通过网络调试助手给 GCAN-PLCCore-M7 发送控制指令。

控制指令含义	控制指令及返回指令	功能码及含义
点亮 LED0	发送：00 00 00 00 00 06 01 **05** 00 00 FF 00 返回：00 00 00 00 00 06 01 **05** 00 00 FF 00	05 强置单线圈
点亮 LED1	发送：00 00 00 00 00 06 01 **05** 00 01 FF 00 返回：00 00 00 00 00 06 01 **05** 00 01 FF 00	05 强置单线圈
熄灭 LED0	发送：00 00 00 00 00 06 01 **05** 00 00 00 00 返回：00 00 00 00 00 06 01 **05** 00 00 00 00	05 强置单线圈
读取 DI 的值	发送：00 00 00 00 00 06 01 **02** 00 00 00 08 返回：00 00 00 00 00 04 01 **02** 01 0C	02 读取输入状态
读取 AI 的值	发送：00 00 00 00 00 06 01 **04** 00 00 00 01 返回：00 00 00 00 00 05 01 **04** 02 0F FB	04 读取输入寄存器
读取两字节保持寄存器	发送：00 00 00 00 00 06 01 **03** 00 00 00 02 返回：00 00 00 00 00 07 01 **03** 04 00 00 00 0C	03 读取保持寄存器

表 8.2 Modbus TCP 实验测试指令

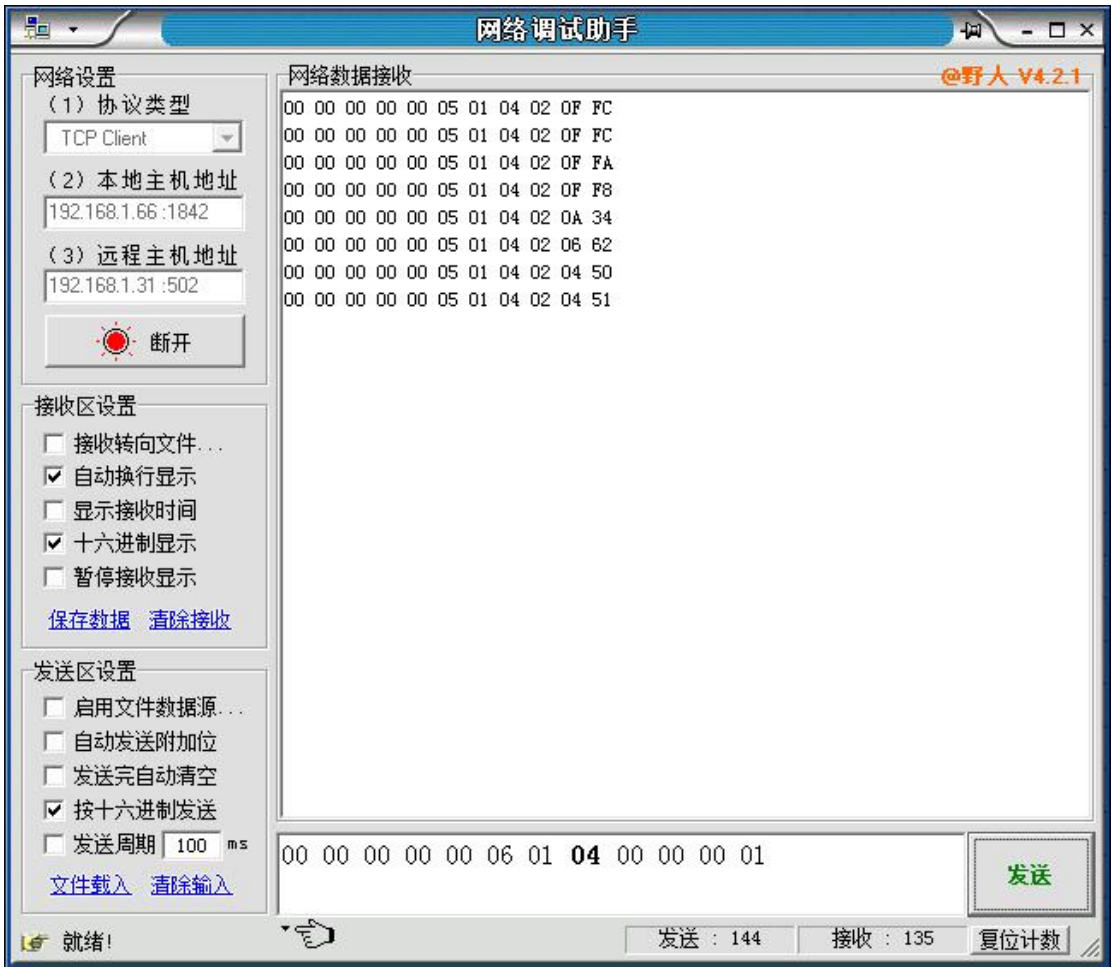


图 8.1 Modbus TCP 实验数据发送界面

我们以“读取 AI 的值”为例，说明一下 Modbus TCP 发送数据与接收数据的含义。通过发送如表 8.3 显示的指令，主机将读取 PLC（Server）中的输入寄存器，

即 modbusAIBuf，从“00 00”起始地址开始，读取一个寄存器。PLC（Server）回送数据如表 8.4 所示。您也可以通过发送“00 00 00 00 00 06 01 **03** 00 02 00 01”，通过读取保持寄存器，即 modbusRegBuf，从“00 02”起始地址开始，读取一个寄存器，同样可以返回当前 AI 的值。

主机（Client）发送	字节	数据（Hex）
传输标识码	Byte0、Byte1	00 00
协议标识符	Byte2、Byte3	00 00
数据长度	Byte4、Byte5	00 06
设备地址	Byte6	01
功能码	Byte7	**04**
起始地址	Byte8、Byte9	00 00
读取寄存器数量	Byte10、Byte11	00 01

表 8.3 “读取 AI 的值”发送指令的含义

从机（Server）回送	字节	数据
传输标识码	Byte0、Byte1	00 00
协议标识符	Byte2、Byte3	00 00
数据长度	Byte4、Byte5	00 05
设备地址	Byte6	01
功能码	Byte7	04
数据字节数	Byte8	02
寄存器值	Byte9、Byte10	0F 0C

表 8.4 “读取 AI 的值”回送指令的含义

9 OpenPCS PLC 软件使用

9.1 软件安装

OpenPCS 2008 编程软件（随货光盘中包含本软件，也可网上下载安装软件）

9.2 PLC 编程界面简介

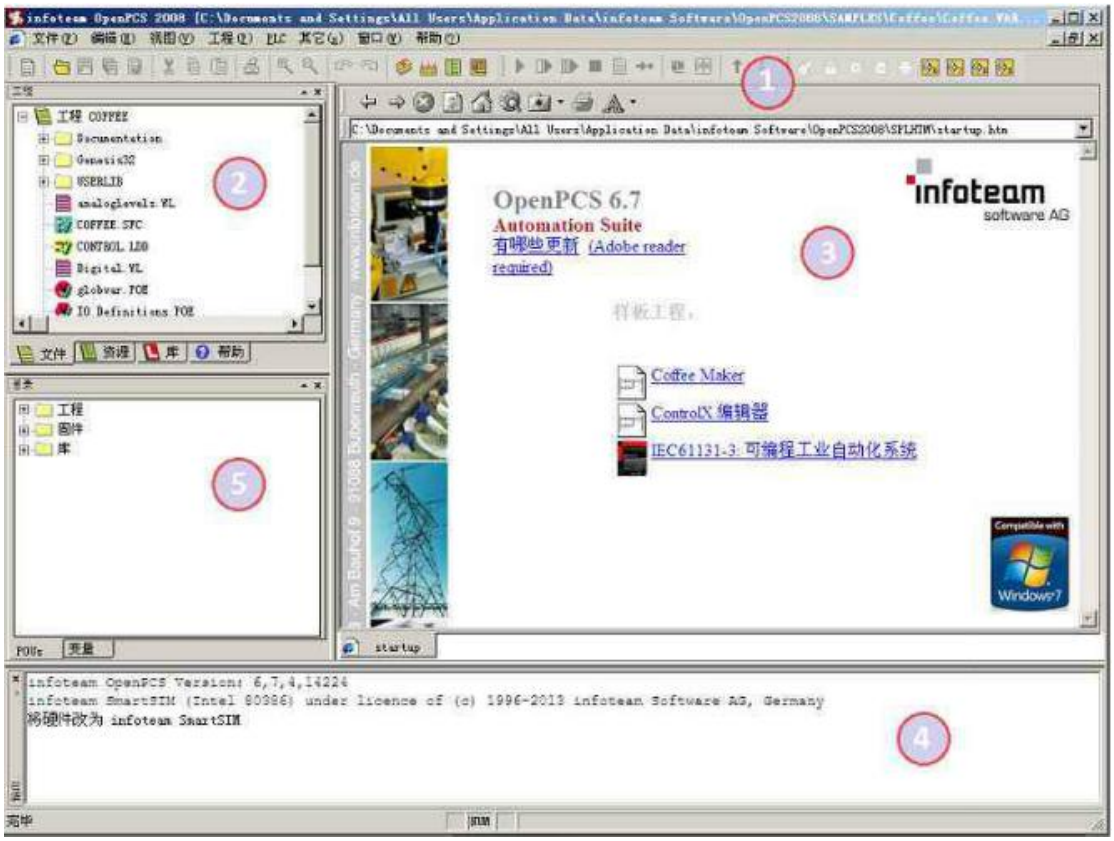


图 9.1 OpenPCS 编程界面

OpenPCS 编程界面中主要包含：

1） 菜单工具栏
2） 工程浏览器
3） 编辑窗口
4） 输出窗口
5） 目录窗口

9.3 创建项目

9.3.1 工程创建

点击文件->新建，创建新项目，如下图 9.2 所示。

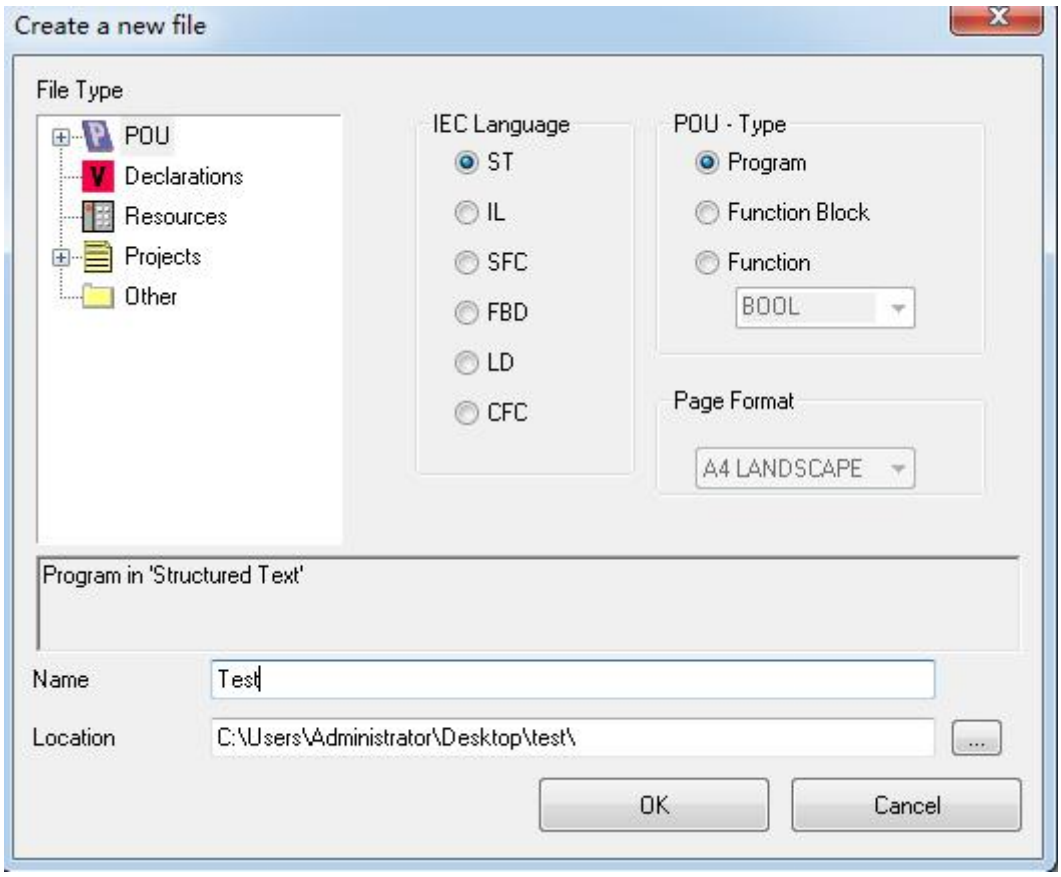


图 9.2 创建项目

9.3.2 添加文件

为项目添加文件(例如：添加功能块-Fuction Block，Sample FB)，如图 9.3 所示。**请注意，Name（名称）一栏中填入的字符串不能以数字为开头。**

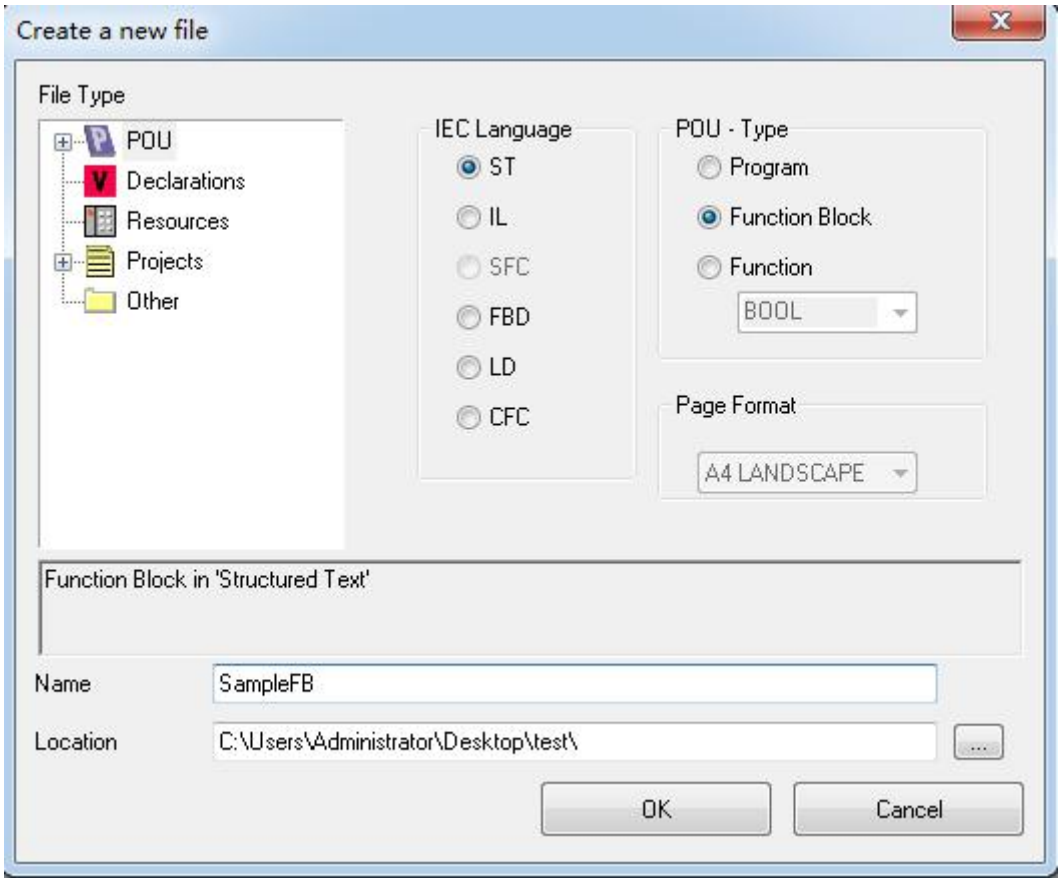


图 9.3 创建功能块

9.3.3 程序编写

首先需要在变量区定义变量（VAR 到 END_VAR）。

```
VAR                             (*内部变量段开始                                    *)
    v1:INT:=0;                  (*:为变量/类型分隔符，:=为初始化操作符              *)
    v2:INT:=0;
    oled at%Q0.0:Byte;          (*   %Q0.0表示输出0单元第0位，:为变量/类型分割符    *)
                                (*符号变量地址声明。分配Q0.0到字节 OLED             *)
                                (* 如果对变量声明不理解，可参考电子书第49页，变量声明的示例*)
END_VAR
```

完成变量定义后便可在下方的编程界面开始编程了，下面为用 ST 编写的简单例程语句：

LED 跑马灯例程：

```
IF v1<100 THEN                  (*v1自加到100时，v1归零。v1越大，闪灯的变化频率越慢   *)
    v1:=v1+1;                   (* := 可表示初始化、输入连接或者赋值                 *)
                                (* 如果对st语言的各种符号不理解可参考电子书第25页，分界符 *)
ELSE
    v1:=0;
    v2:=v2+1;
    if v2>=255 then             (*v2自加到255时，v2归零                             *)
        v2:=0;
    end_if;
    oled:=int_to_byte(v2);      (* int_to_byte 整型转字节。类型转换类函数，电子书57页 *)
                                (* :=这里表示赋值                                    *)
end_if;
```

9.3.4 设置调试连接

1、点击 PLC->Connections...（连接...）。

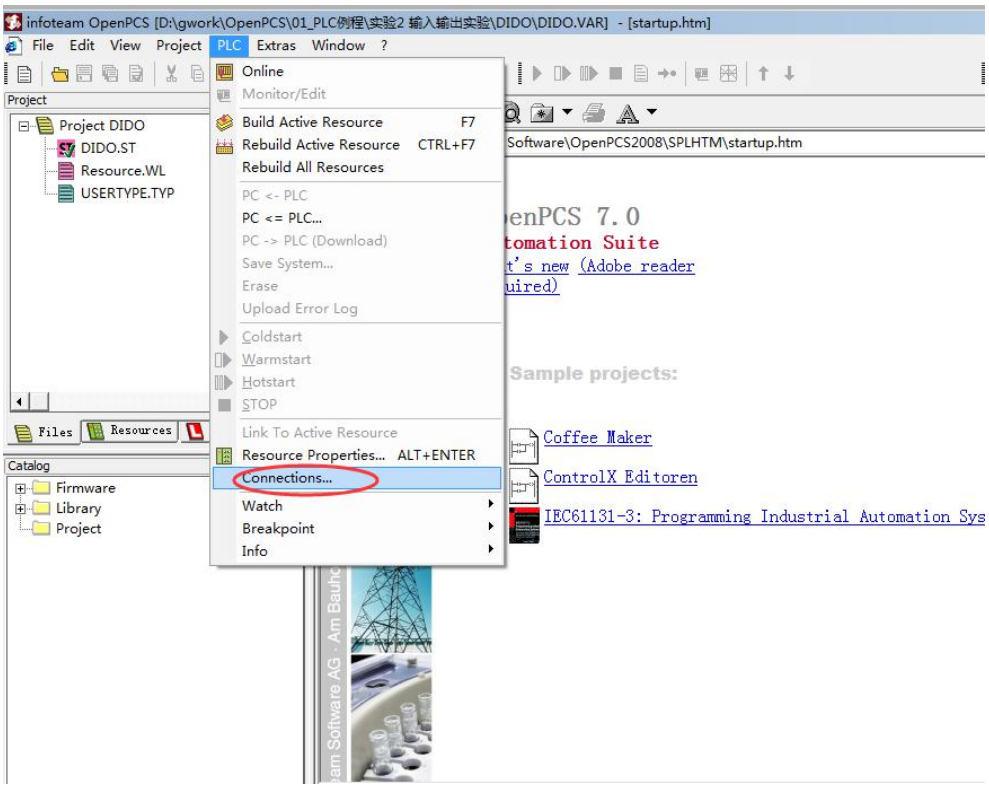


图 9.4 调试连接

2、在 Connection Setup（连接设置）窗口新建连接，设置参数。点击“New”按钮。

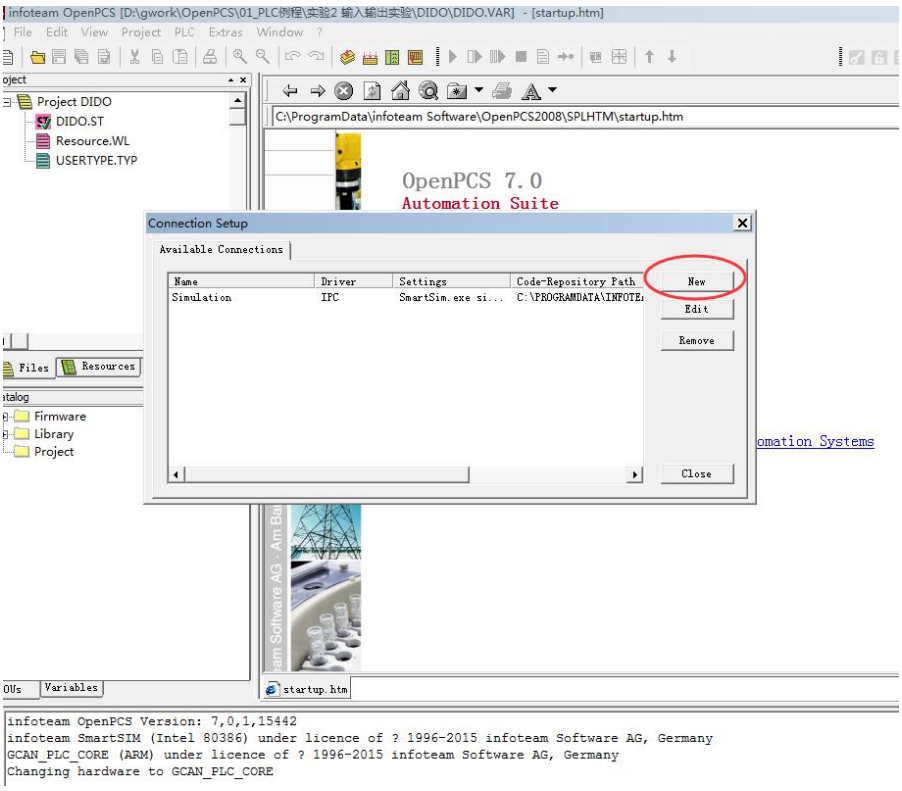


图 9.5 点击“New”

3、在 Name 中输入 RS232，点击 Select 按钮。

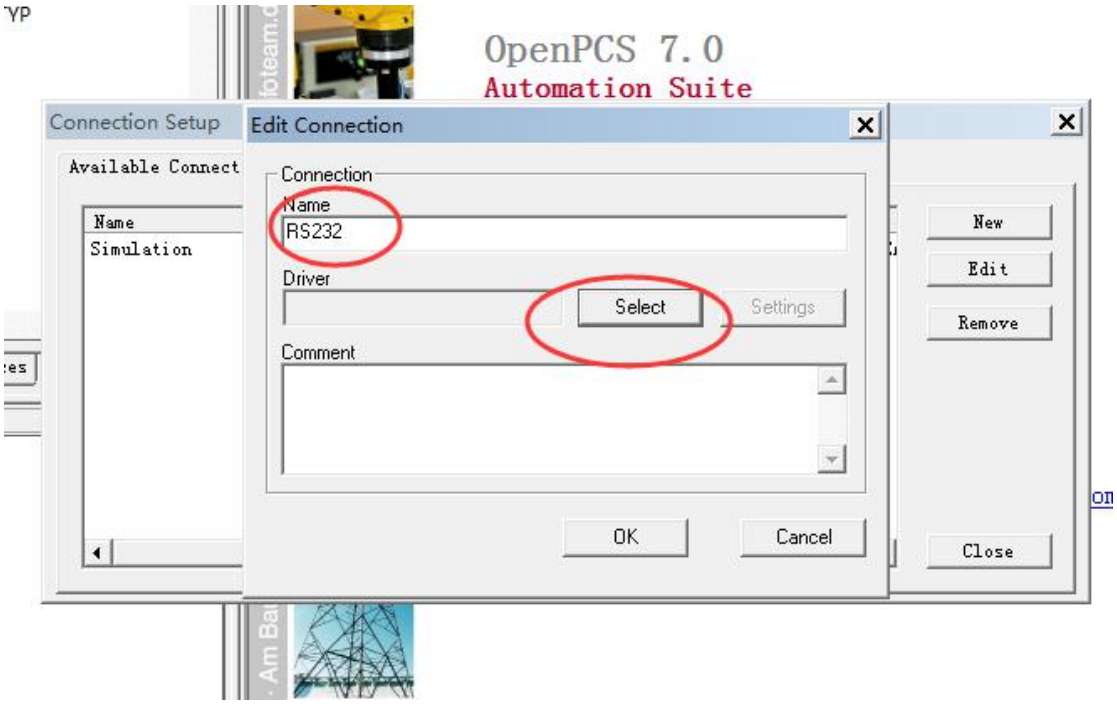


图 9.6 点击“Select”按钮

4、点击 RS232 图标 ，之后点击 OK。

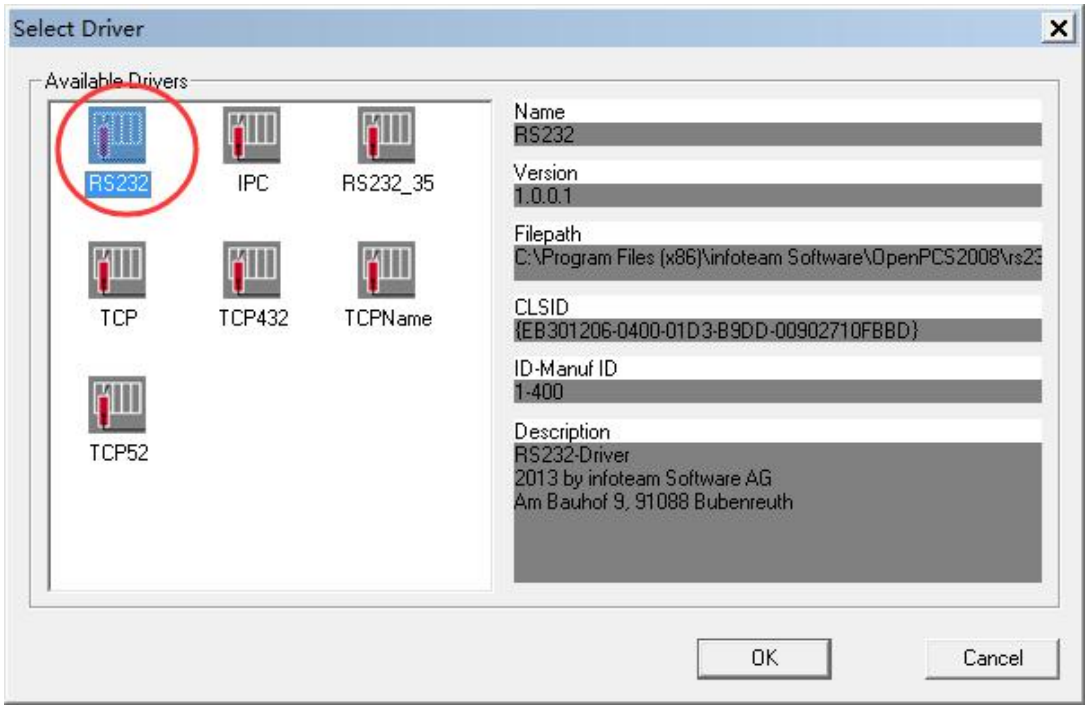


图 9.7 选择 RS232

5、Driver 中会显示“RS232”字样，点击“Settings（设置）”按钮。

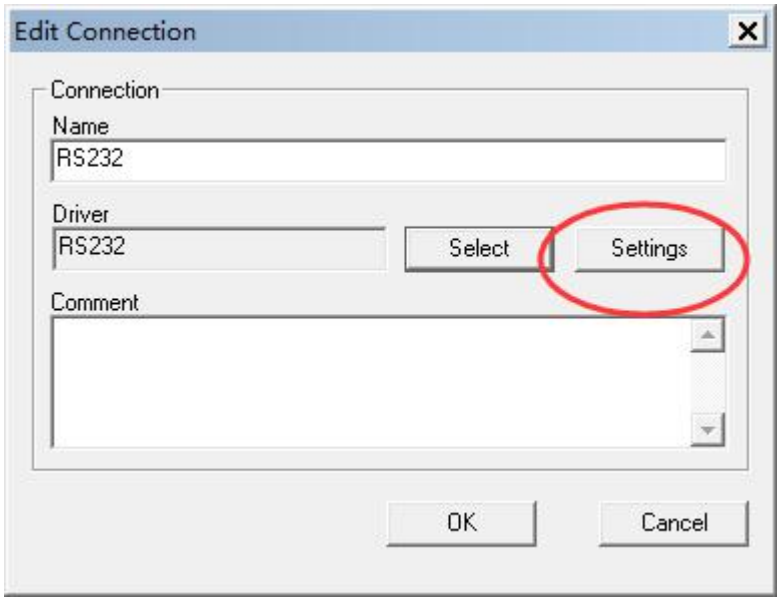


图 9.8 点击“Settings”按钮

6、Port（端口）请选择我司设备与 PC 机连接时的串口号，如不确定请在设备管理器中进行查询。Baud rate 选择 19200，Parity（奇偶性）选择 None，Stop bits（停止位）选择 1，Protocol（协议）选择 None。设置好后点击 OK。

请注意：在日常使用时，串口号（Port）可能会发生变动，会产生联机失败的现象。此时需要您及时修改串口号。

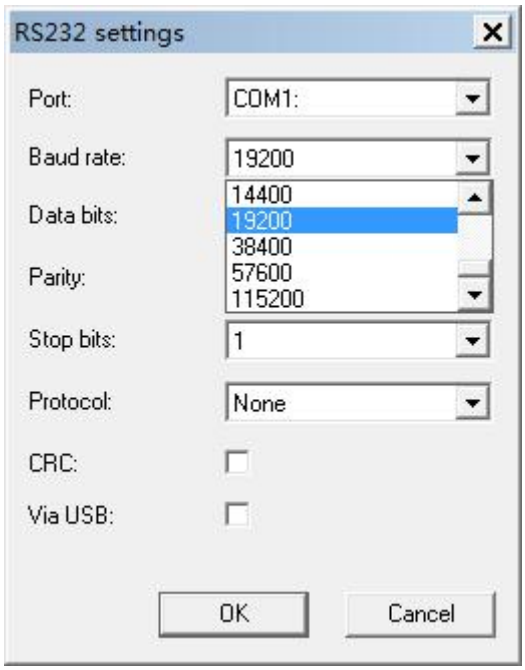


图 9.9 RS232 波特率设置

7、设置好后，返回 Connection Setup（连接设置）界面，点击“Close（关闭）”。

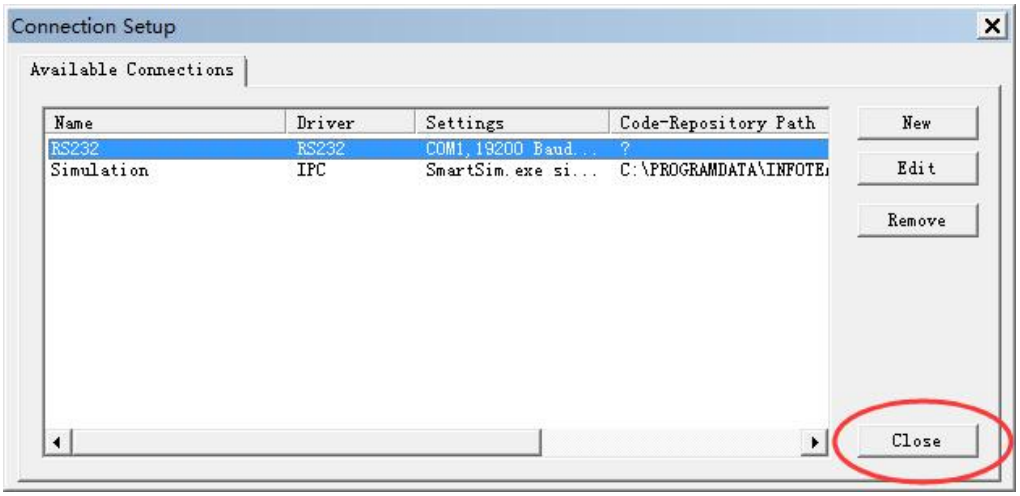


图 9.10 点击“Close”

8、设置 Resource Properties（资源属性），如下图所示。

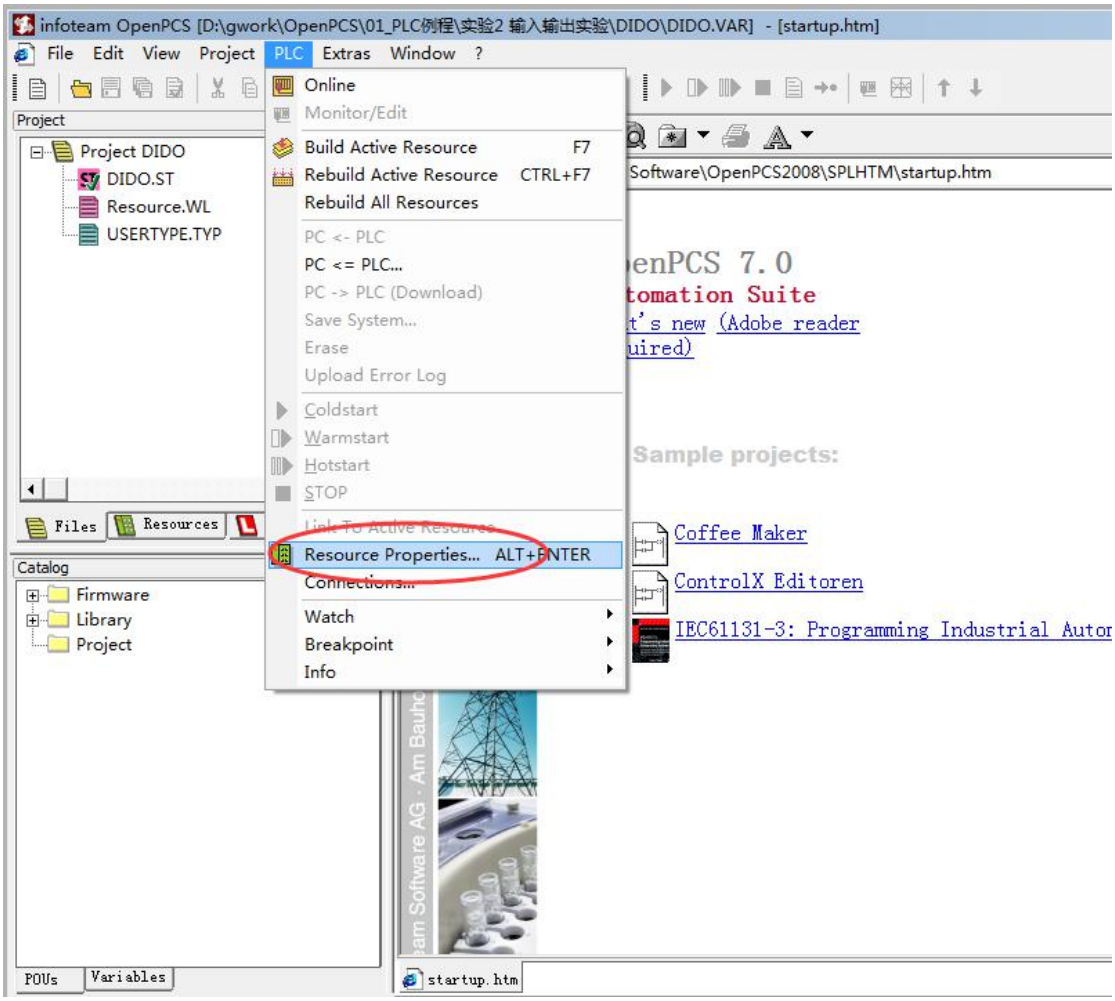


图 9.11 设置资源属性

9、选择 GCAN_PLC 和 RS232。

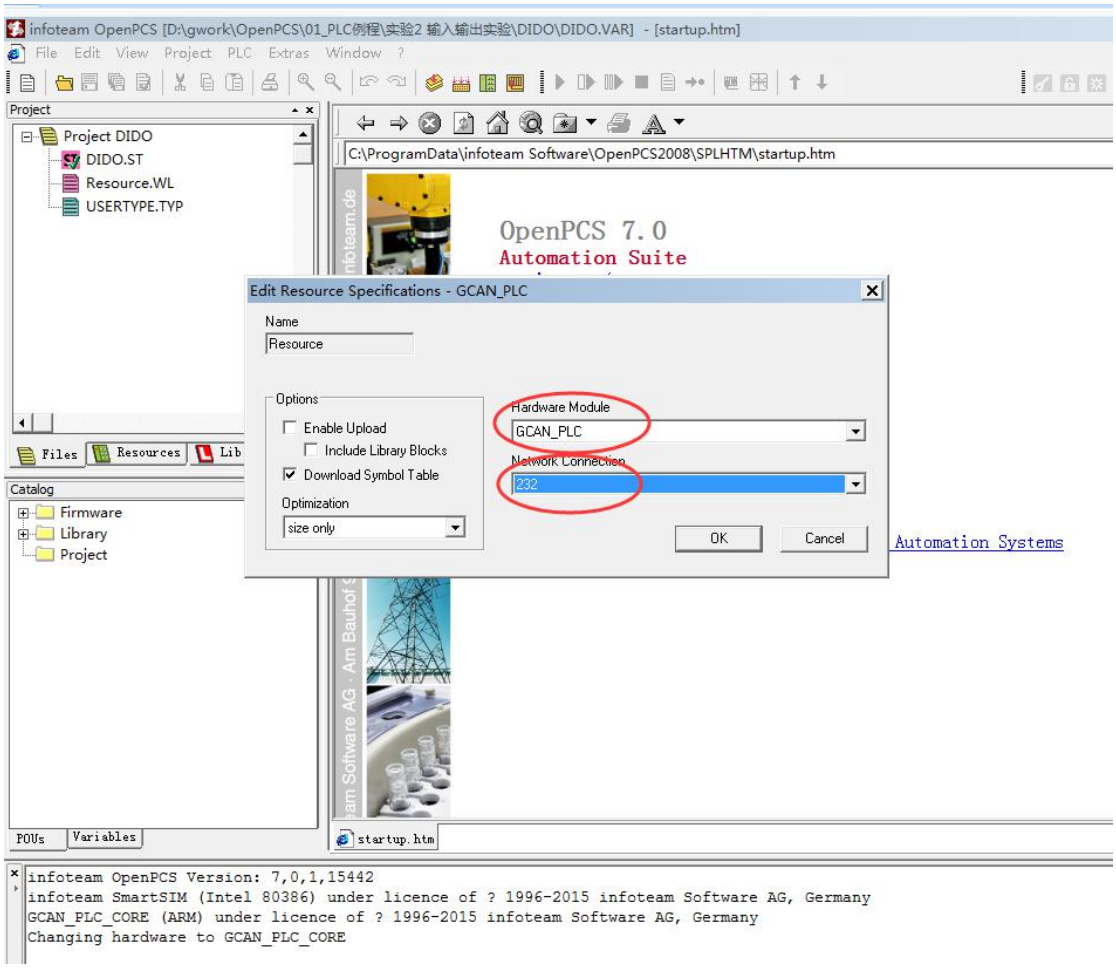


图 9.12 选择 GCAN_PLC 和 RS232

9.3.5 下载程序并调试

1、完成程序编写后需点击 Build Active Resource（生成当前资源）按钮，如图 9.13 所示。

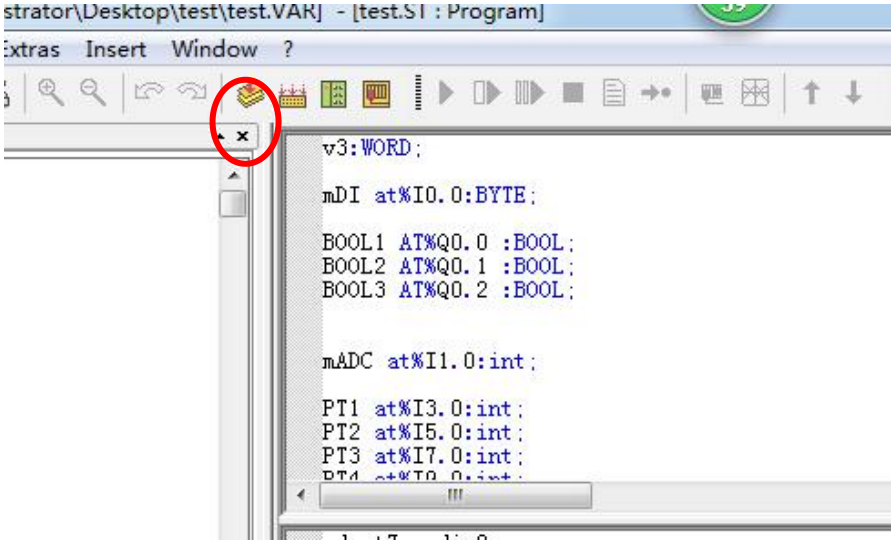


图 9.13 点击 Build Active Resource 按钮

2、编译完成后，提示没有错误。如下图所示。

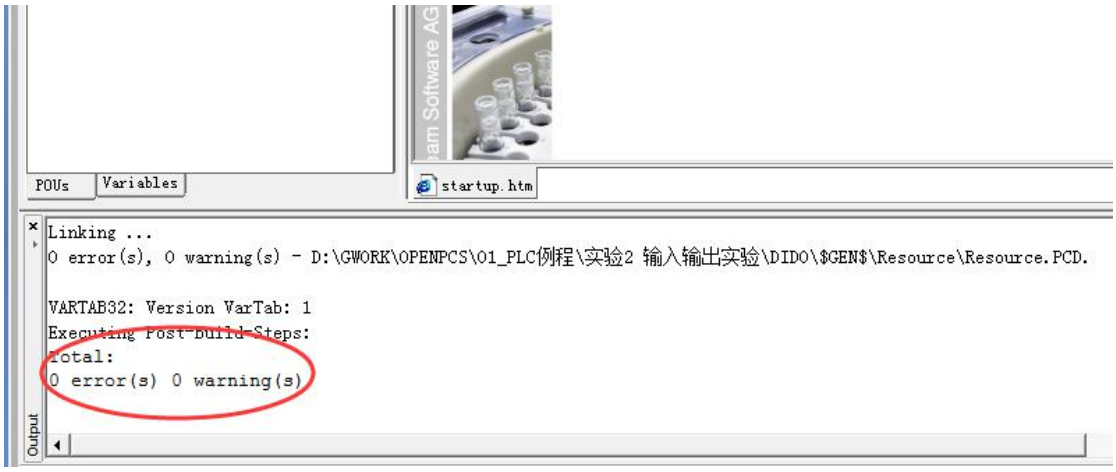


图 9.14 编译完成

3、点击 Online（联机）按钮。

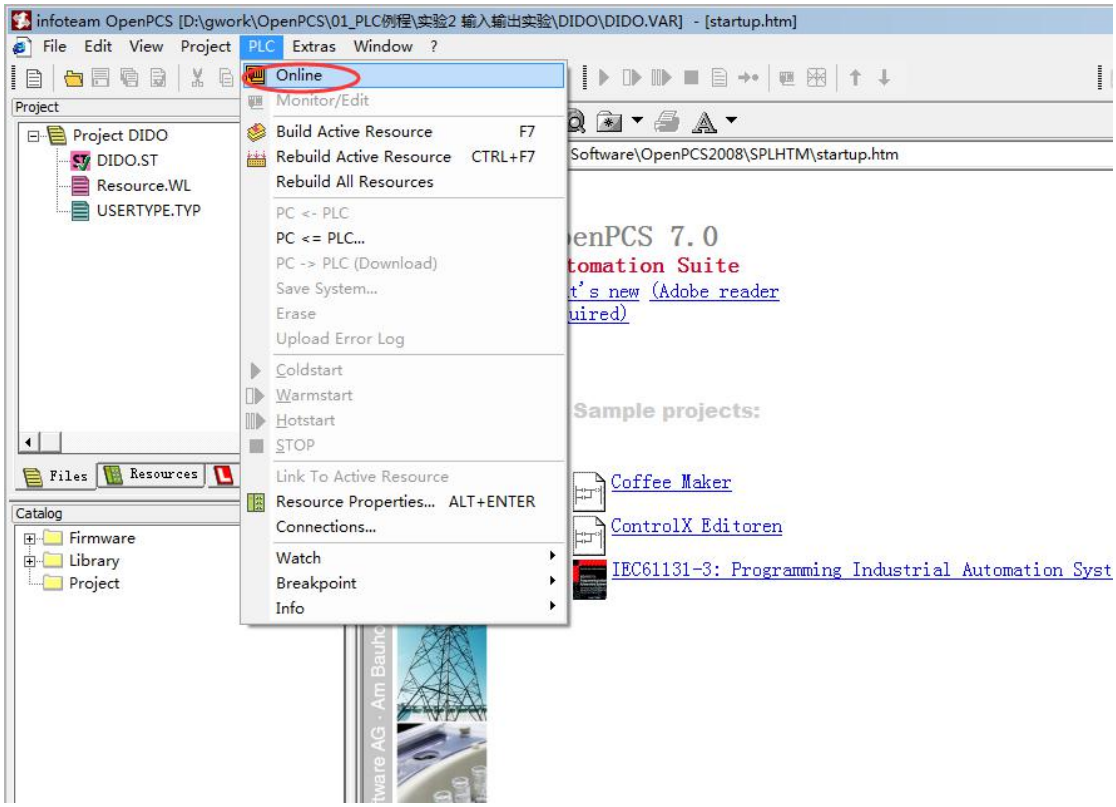


图 9.15 点击 Online 按钮

4、在下拉菜单中点击 PC->PLC(Download)下载程序。

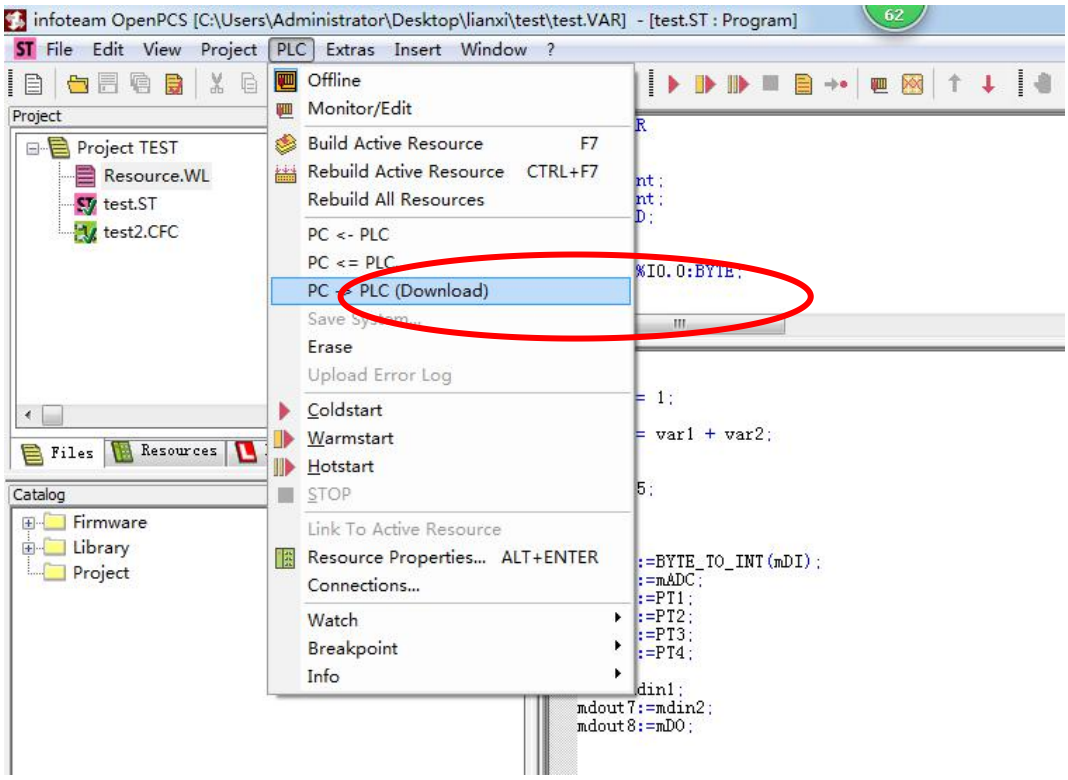
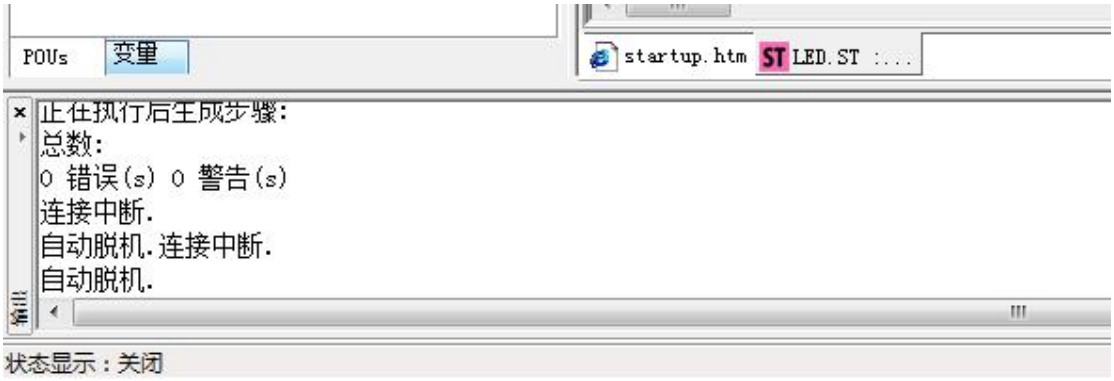

infoteam OpenPCS [C:\Users\Administrator\Desktop\lianxi\test\test.VAR] - [test.ST : Program]
File Edit View Project PLC Extras Insert Window ?
Project
Project TEST
Resource.WL
test.ST
test2.CFC
Files Resources
Catalog
Firmware
Library
Project
Offline
Monitor/Edit
Build Active Resource F7
Rebuild Active Resource CTRL+F7
Rebuild All Resources
PC <- PLC
PC <= PLC
PC -> PLC (Download)
Erase
Upload Error Log
Coldstart
Warmstart
Hotstart
STOP
Link To Active Resource
Resource Properties... ALT+ENTER
Connections...
Watch
Breakpoint
Info
%I0.0:BYTE;
= 1;
= var1 + var2;
:=BYTE_TO_INT(mDI);
:=mADC;
:=PT1;
:=PT2;
:=PT3;
:=PT4;
mdout7:=mdin2;
mdout8:=mDO;

图 9.16 下载程序

附录 A：常见问题

1. Q: 点击“联机”按钮后，状态栏中显示自动脱机。无法下载程序。



A: 请确定您已按照“**9.3.4 设置调试连接**”设置好了串口端的通信参数，设置的串口号与电脑中设备管理器显示的 CH340 串口号一致。如 CH340 驱动显示异常，请尝试重新安装驱动。如上述情况都已考虑但仍显示自动脱机，请重新启动电脑。

附录 B：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。

B.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 B.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 B.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。

B.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（I 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 B.1 给出了 Modbus 与 Modbus TCP 数据帧格式比较。

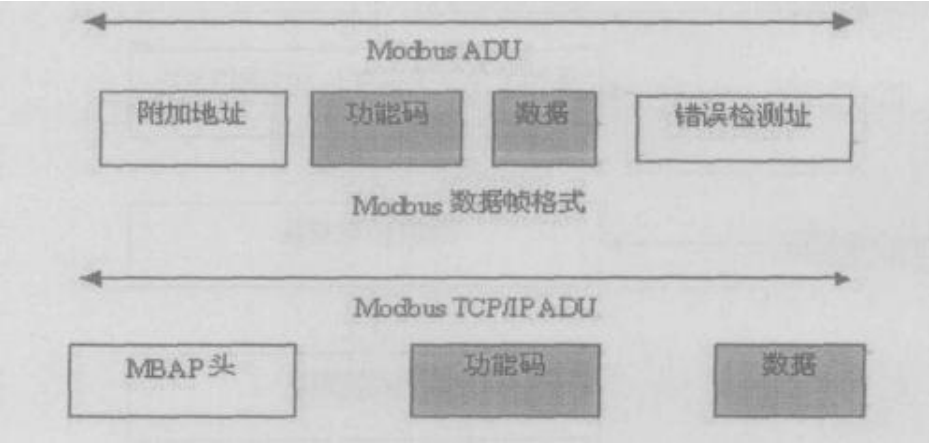


图 B.1 Modbus 与 Modbus TCP/IP 帧格式

Modbus TCP 的 ADU 数据单元规范如表 B.1 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 B.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 B.2 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 B.3 Modbus TCP 响应数据格式

B.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

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。

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，M84 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

销售与服务

沈阳广成科技有限公司

地址：辽宁省沈阳市皇姑区崇山中路 42 号工业设计中心

邮编：110000

电话：024-31230060

全国服务电话：400-6655-220

网址：www.gcgd.net

全国销售与服务电话：400-6655-220

售后服务电话与微信号：13840170070