分布式 I/O 系统

BL200 系列耦合器 Modbus/MQTT/OPC UA

钡铼

BL200 系列说明书

版本：V1.0

日期：2021-10-12

网址：深圳市钡铼技术有限公司

网址：www.bliiot.cn

前言

感谢您使用深圳市钡铼技术有限公司的分布式 I/O 系统，阅读本产品说明书能让您快速掌握本产品的功能和使用方法。

版权声明

免责声明

本产品主要用于工业现场自动化测量和控制，操作人员必须是自动化或电气领域中具有经验的电气专家。请按照说明书提供的参数和技术规格使用，本公司不承担由于不正常使用或不恰当使用本产品造成的财产或人身伤害。

修订记录

更新日期	文档版本	说明	作者
2021 年 10 月 13 日	V1.0	初版	ZLF

目录

1. 产品简介

1.1概述

BL200 系列耦合器是一个数据采集和控制系统，基于强大的 32 位 ARM926EJ-S™ 微处理器设计，采用 Linux 操作系统，支持 Modbus，MQTT，OPC UA 等多种协议，可以快速接入现场 PLC、MES 和 SCADA 以及 ERP 系统，同时也能快速连接到 AWS 云，华为云以及阿里云等众多云平台。另外该设备提供 SDK，允许用户进行二次开发，满足碎片化的应用市场需求。

BL200 系列耦合器是一种模块化的分布式 I/O 系统。该系统由 3 部分组成：现场总线耦合器和各种类型的（数字和模拟信号以及特殊功）I/O 模块以及终端模块。

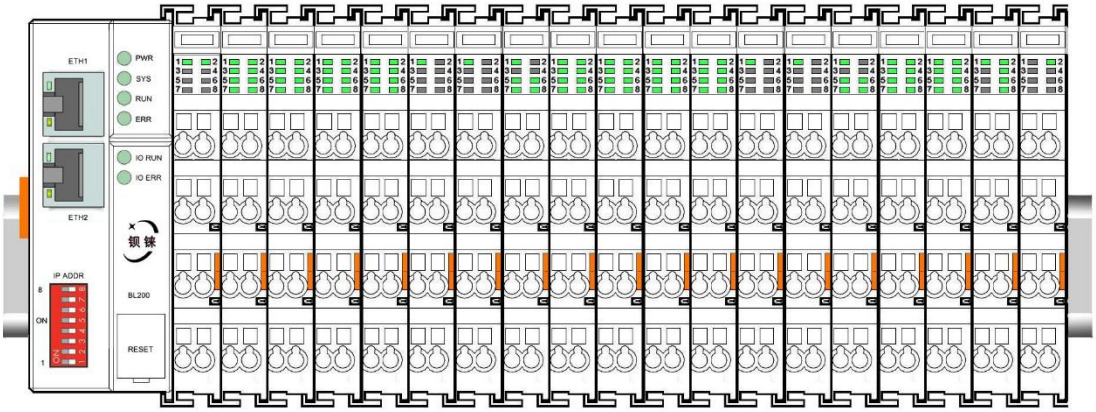


图 1：现场节点

该节点和现场设备（例如 PLC）之间的通讯通过现场总线耦合器的以太网接口实现，现场总线耦合器和 I/O 模块之间的通信通过本地总线进行。两个以太网接口内部集成交换机功能，可以建立线性拓扑，无需额外的交换机或集线器。

该系统需要使用电源模块提供 24VDC 系统电压和 24VDC 现场电压，现场总线耦合器本身包含一个电源模块。由于使用了 2 组独立的电源，现场电压和系统电压彼此电气隔离。

在装配现场总线节点模块时，各个 I/O 模块可以任意组合排列，不要求按模块类型分组。

必须始终将终端模块（例如 TERM）插入现场总线节点的末端，确保正确的数据传输。

BL200 系列耦合器可以通过内置的 web 服务器进行配置和测试。

1.2典型应用

分布式 I/O 模块系统由于具有可靠度高、容易扩展、设置容易、网络布线方便等特性，适用于分散地区的应用，广泛应用于数据收集和各种控制。产品广泛应用于物联网、智慧工厂、智慧医疗、智能家居、智能交通、机房动力环境监控、电力、石油监控等行业。

1.3功能特点

- 最大可接入 32 个 I/O 模块;
- 支持多种工业通信协议 Modbus、MQTT、OPC UA 等;
- 支持接入阿里云、华为云、AWS 云、Thingsboard 等;
- 现场侧和系统侧以及总线侧三者彼此电气隔离;
- 支持 2 X RJ45 接口，集成交换机功能，可以建立线路拓扑，节约交换机或集线器;
- 便捷的接线连接技术，免螺丝安装;

1.4技术参数

表 1：技术参数

名称	参数	描述
系统电源	供电电源(系统):	24 VDC
	输入电流(系统):	最大 500 mA@24VDC
	电源效率	84%
	内部总线电压:	5VDC
	耦合器消耗电流:	最大 300mA@5VDC
	I/O 模块消耗电流:	最大 1700mA@5VDC
	隔离保护:	500 V 系统/供电
现场电源	供电电源(现场):	24 VDC
	电源跨接触点电流(最大) :	10 ADC
以太网	数量:	2 X RJ45
	传输介质	双绞线，STP 100 Ω Cat 5
	最大线缆长度	100m
	速率	10/100 Mbit/s
	隔离保护	ESD 接触: 8KV ，浪涌: 4KV（10/1000us）
系统	操作系统	Linux
	处理器	ARM926EJ-S
	主频	300MHz
	RAM	64MB
	Flash	128MB
	I/O 模块数量	最大 32
	通过串口模块过程映射(Modbus)数据点	● Bool : 4096 ● 16 Bit : 2048 ● 32 Bit : 1024
	协议	Modbus TCP , MQTT , OPC UA , HTTP , BootP , DHCP , DNS , SNTP , SNMP
	最大连接数	15 Modbus TCP
接线方式	连接技术:	笼式弹簧连接技术
	导线直径	0.08 mm² … 2.5 mm², AWG 28 … 14

	剥线长度	8 mm ⋯ 9 mm / 0.33 in
环境	工作温度:	0 ⋯ 55 °C
	储存温度:	-40 ⋯ 70 °C
	相对湿度:	5 ⋯ 95% 无凝结
	工作海拔:	0 ⋯ 2000 m
	防护类型:	IP20
几何尺寸	宽度:	48mm
	高度:	100mm
	深度:	69mm
材料	颜色:	浅灰色
	外壳材料:	聚碳酸酯,尼龙 6.6
	火灾荷载:	1.239 MJ
	重量:	180 g
机械	安装方式	DIN-35 型导轨
认证	EMC	EN 55022: 2006/A1: 2007 (CE &RE) Class B
		IEC 61000-4-2 (ESD) Level 4
		IEC 61000-4-3 (RS) Level 4
		IEC 61000-4-4 (EFT) Level 4
		IEC 61000-4-5 (Surge)Level 3
		IEC 61000-4-6 (CS)Level 4
		IEC 61000-4-8 (M/S) Level 4

1.5设备选型

表 2：设备选型

编号	名称	型号	通道	信号类型
1	8 通道数字输入模块	M1082	8	NPN
2	8 通道数字输入模块	M1081	8	PNP
3	8 通道数字输出模块	M2082	8	NPN
4	8 通道数字输出模块	M2081	8	PNP
5	4 通道模拟输入模块	M3041	4	Current
6	4 通道模拟输出模块	M4043	4	Voltage
7	4 通道 RTD 输入模块	M5041	4	Resistor
8	2 通道串口通信模块	M6021	2	RS485
9	24V 电源模块	M701	/	/
10	终端模块	M801	/	/
11	Modbus-TCP I/O 耦合器	BL200	/	/
12	OPC UA 边缘计算 I/O 控制器	BL200UA	/	/
13	MQTT 边缘计算 I/O 控制器	BL200M	/	/
14	多协议边缘计算 I/O 控制器	BL200Pro	/	/
15	分布式 Profinet 总线耦合器	BL200PN	/	/

16	分布式 Ethernet/IP 总线耦合器	BL200EP	/	/
17	分布式 EtherCAT 总线耦合器	BL200EC	/	/
18	分布式 BACnet/IP 总线耦合器	BL200BN	/	/

2. 设备描述

2.1视图

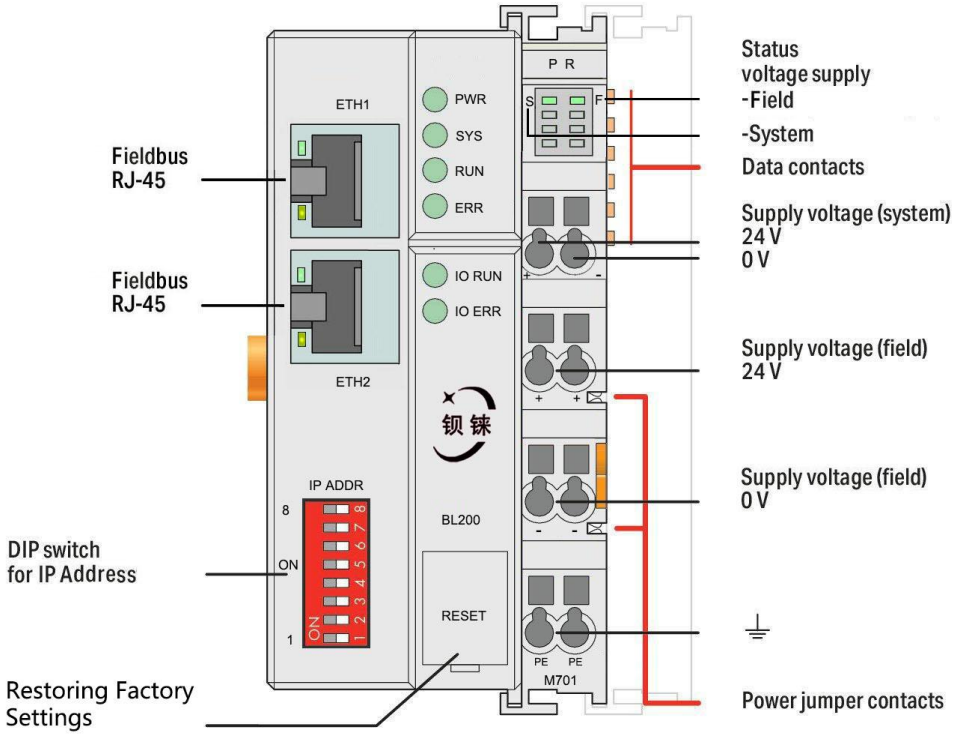


图 2：视图

2.2尺寸

单位：mm

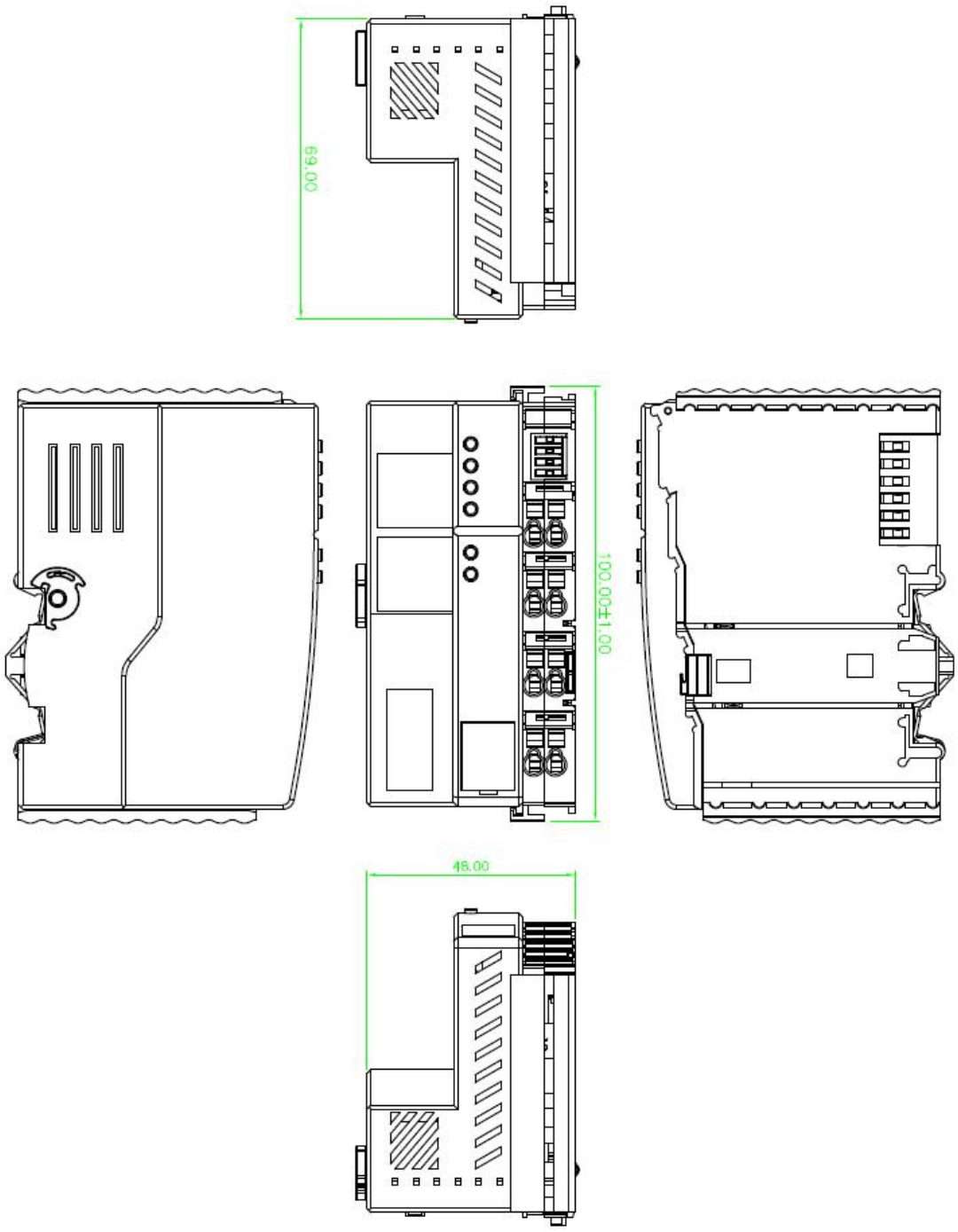


图 3：2D 结构示意图

2.3数据触点/内部总线

现场总线耦合器和 I/O 模块之间的通信，以及 I/O 模块的系统供电都是通过内部总线实现的。内部总线是由 6 个数据触点组成，这些镀金触点在连接时可进行自清洁。

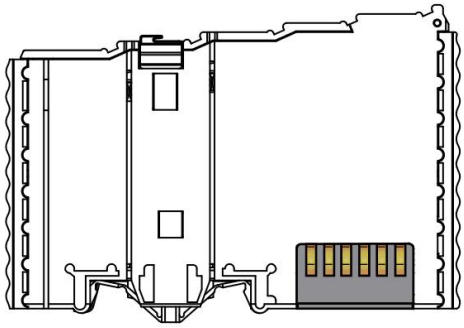


图 4：数据触点

2.4 电源跨接触点

该耦合器自带的电源模块有两个自清洁电源跨接触点，可为现场侧进行供电。该电源跨接触点的最大电流为 10A，电流超出最大值会损坏触点。在对系统进行配置时须保证不超过上述电流最大值，如超出，则需插入一个电源模块。

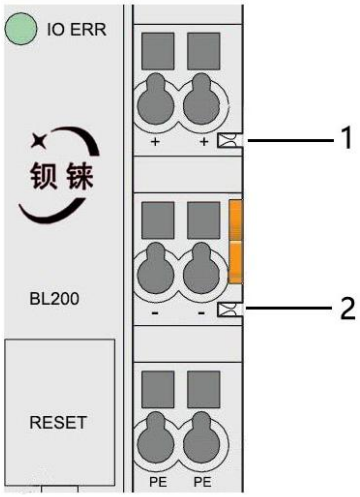


图 5：电源跨接触点

表 4："电源跨接触点"描述

编号	类型	描述
1	弹簧触点	给现场侧供电 24V
2	弹簧触点	给现场侧供电 0V

2.5接线点

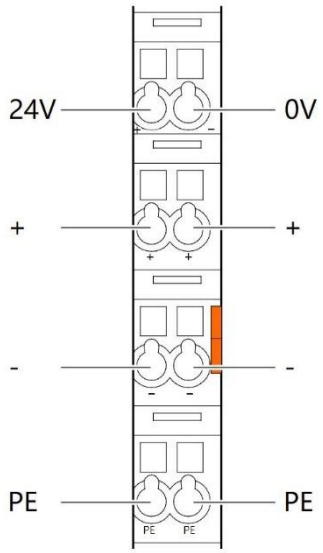


图 6：接线点

表 5：“接线点”描述

名称	描述
24V	系统电源 24VDC
0V	系统电源 0VDC
+	现场 24V 电压
+	现场 24V 电压
-	现场 24V 电压
-	现场 24V 电压
PE	接地点
PE	接地点

2.6LED 指示灯

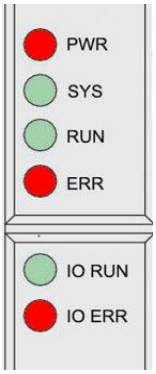


图 7：耦合器 LED 指示灯

表 6：“耦合器 LED 指示灯”描述

编号	描述	颜色	状态	功能
PWR	电源指示灯	红色	亮	连电源正常
			灭	没有电源
SYS	系统指示灯	绿色	亮	系统正常
			闪烁	系统未配置
			灭	系统故障
RUN	运行指示灯	绿色	亮	IP 正常
			闪烁	IP 冲突
			灭	未分配 IP
ERR	错误指示灯	红色	亮	至少有一个北向协议连接
			闪烁	收发数据
			灭	没有北向协议连接
I/O RUN	I/O 模块运行指示灯	绿色	亮	正常
			闪烁	节点通讯失败
			灭	没有节点
I/O ERR	I/O 模块错误指示灯	红色	亮	正常
			闪烁	节点通讯失败
			灭	没有节点

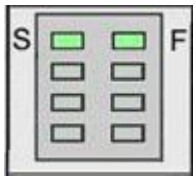


图 8：电源模块 LED 指示灯

表 7：“电源模块 LED 指示灯”描述

编号	描述	颜色	状态	功能
1	现场 24V 电源指示灯	绿色	亮	电源正常
			灭	没有电源
2	系统 24V 电源指示灯	绿色	亮	电源正常
			灭	没有电源

2.7以太网接口

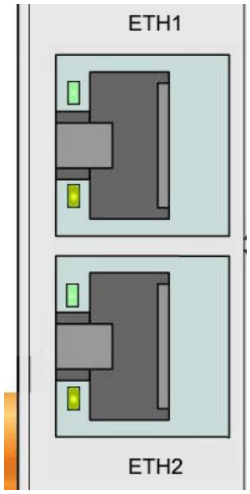


图 9：以太网接口

2.8IP 地址选择开关

8 位 DIP 开关用于设置 IP 地址。DIP 开关的编码是按位进行的，从最低有效位（2^0）的 DIP 开关 1 开始，到最高有效位（2^7）的 DIP 开关 8，对应十进制值：0-255。

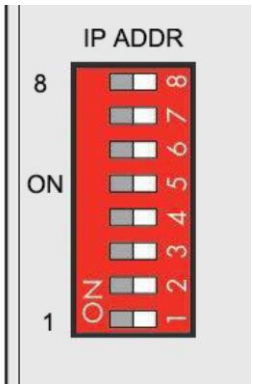


图 10：IP 地址选择开关（示例：设置“0”）

对于 DIP 开关的值为 0000 0000（十进制 0）时，IP 地址为 192.168.1.10;

对于 DIP 开关的值为 1111 1111（十进制 255）时，IP 地址根据 DHCP 协议自动分配 IP;

对于 DIP 开关的值为 0000 0001 – 1111 1110（十进制 1-254）时，确定 IP 地址的第 3 个字节，第 1、2 和 4 三个字节为固定字节，即 192.168.xxx.253。

2.9恢复出厂按钮

此按钮用于将设备配置参数恢复到出厂状态。

操作方法：

1. 在设备处于正常运行状态下，打开翻盖;
2. 长按 5 秒以上，直到全部 LED 灯熄灭表明操作成功，然后设备会自动重启。

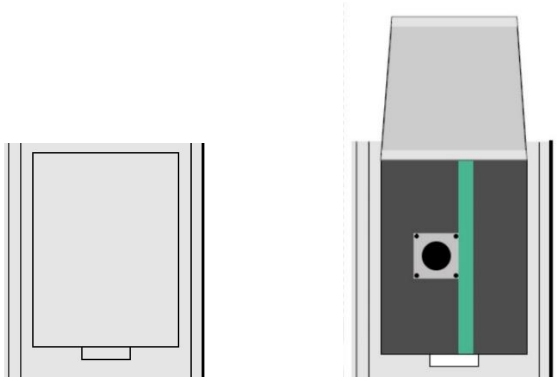


图 11：恢复出厂设置按钮（关闭和打开）

2.10 原理方框图

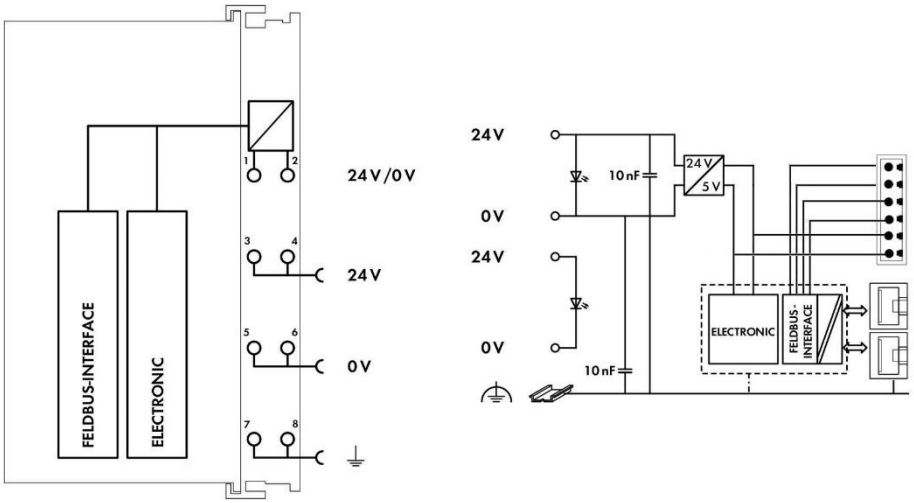


图 12：原理方框图

3. 产品安装

3.1安装顺序

钡铼技术的所有分布式现场总线耦合器和 I/O 模块必须安装在标准 DIN 35 导轨上。

从现场总线耦合器开始，从左向右依次装配总线 I/O 模块，各个模块彼此相邻安装。所有 I/O 模块的右侧都有凹槽和电源跨接触点，为了避免装配错误，必须从右侧和顶部插入 I/O 模块，以免损坏模快。

利用簧片和凹槽系统形成可靠的装配和连接。通过自动锁闭功能，在完成安装后各个组件被安全地固定在导轨上。

不要忘记安装终端模块！始终将终端模块（例如 TERM）插入现场总线节点的末端，确保正确的数据传输。

3.2安装耦合器

1．先将耦合器卡入 DIN 导轨；
2．然后使用螺丝刀等工具转动锁定凸轮，直到锁定凸轮与 DIN 导轨啮合。

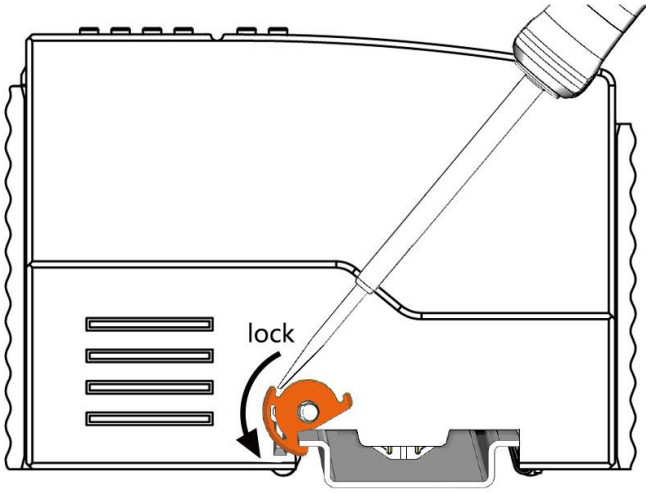


图 13：锁定耦合器

3.3移除耦合器

1．使用螺丝刀转动锁定盘凸轮，直到锁定凸轮不再与导轨啮合。

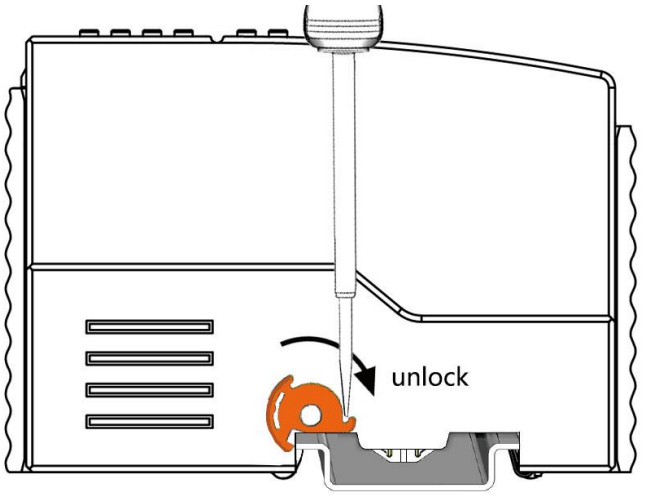


图 14：解锁耦合器

2. 拉动释放卡舌，从组件上拆下现场总线耦合器。

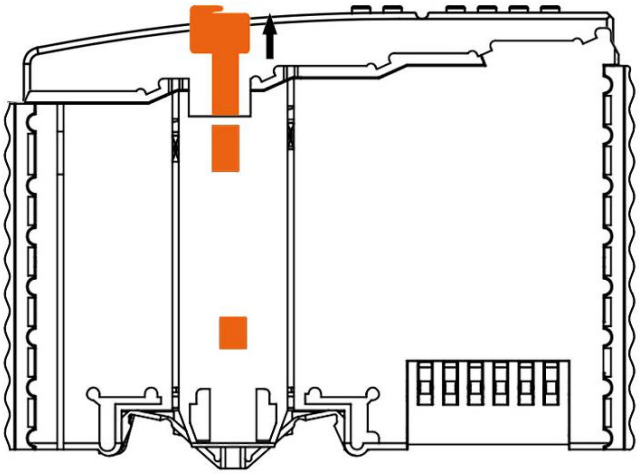


图 15：解锁耦合器

移除现场总线耦合器/控制器时，数据或电源触点与相邻 I/O 模块的电气连接断开

3.4插入模块

1．插入模块时，确认模块上的簧片对齐所连接的耦合器或者其他 I/O 模块的凹槽。

图 16：对准凹槽（示例）

2．将 I/O 模块压入装配位置，直到 I/O 模块卡入导轨。

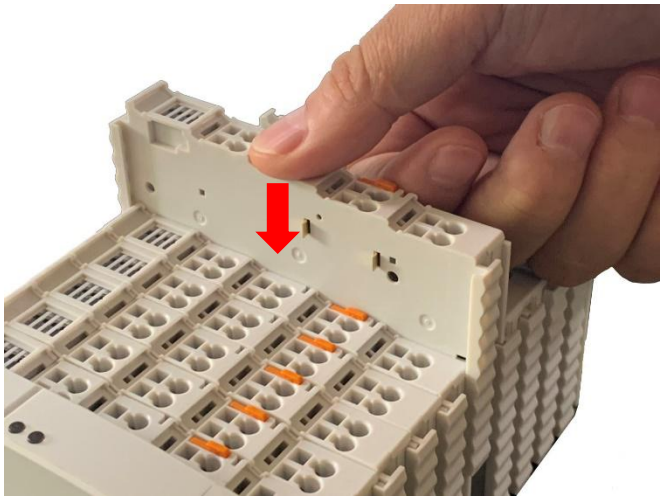


图 17：将 I/O 模块卡入到位（示例）

将 I/O 模块安装好后，就通过数据触点和电源跳线触点建立好与现场总线耦合器（或前一个 I/O 模块）和后面 I/O 模块的电气连接了。

3.5移除模块

向上提拉锁栓，从组件中移除 I/O 模块。

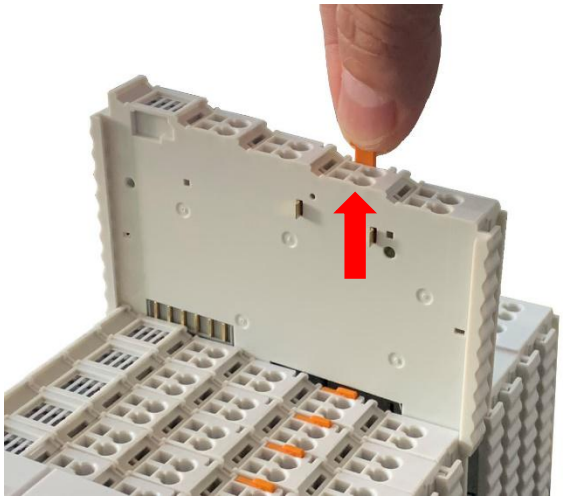


图 18：移除 I/O 模块（示例）

移除 I/O 模块后，数据或电源跳线触点的电气连接断开。

4. 连接设备

4.1连接导线

笼式弹簧连接适用于单股导线、多股导线和细多股导线。每个笼式弹簧只能连接一根导线，如果超过一根导线，则必须汇成一点后再接入。

- 1. 先将工具插入接线处上方的开口，打开笼式弹簧。
- 2. 将导线插入相应打开的连接端。
- 3. 移除工具后即可将笼式弹簧关闭，导线即被弹簧牢固地卡住。

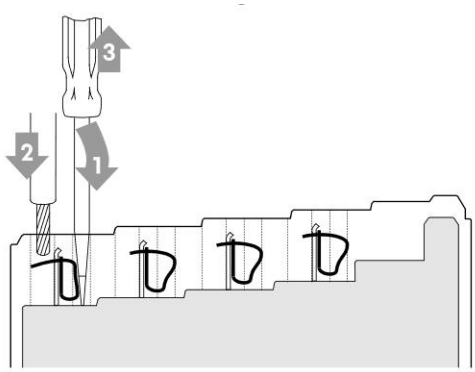


图 19：连接导线

4.2连接电源

系统和现场电压由供电模块提供。BL200 系列耦合器自带一个供电模块，可以为耦合器和 I/O 模块的内部电子设备供电。如有必要（I/O 模块比较多，电流比较大），还可通过独立的供电模块提供。

现场总线接口（以太网接口）、系统和现场三者之间彼此电位隔离。

4.2.1 系统电源

BL200 系列耦合器需要 24 V 直流系统电源，并从供电模块的接线端子接入。系统内部所需要的 5V 总线电压是通过 24V 系统电压转换而来。

供电模块仅具有适当的熔断器保护，请在外部提供适当的过流保护。

请注意匹配供电模块的输出功率和负载功率，避免负载电流过大的情况发生。

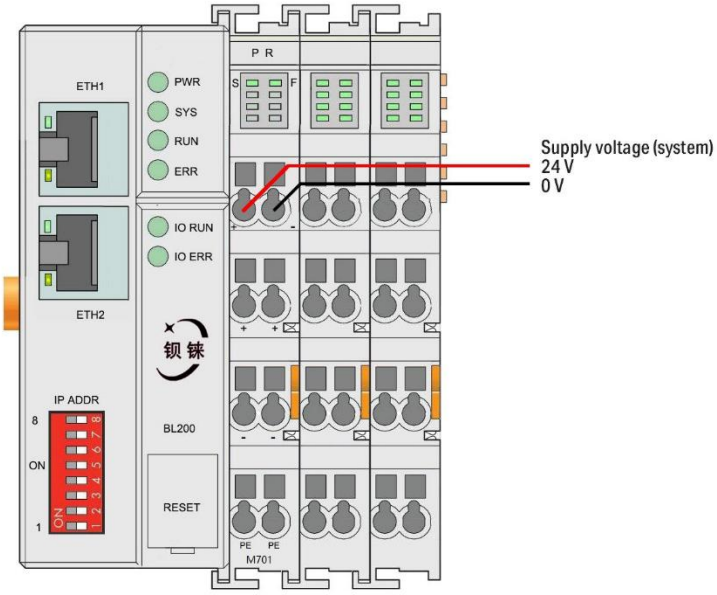


图 20：连接系统电源示意图

4.2.2 现场电源

供电模块在现场侧提供 24 VDC 电压，为传感器和执行器供电。
现场供电仅具有适当的熔断器保护。如果没有过流保护，电子设备可能会损坏。

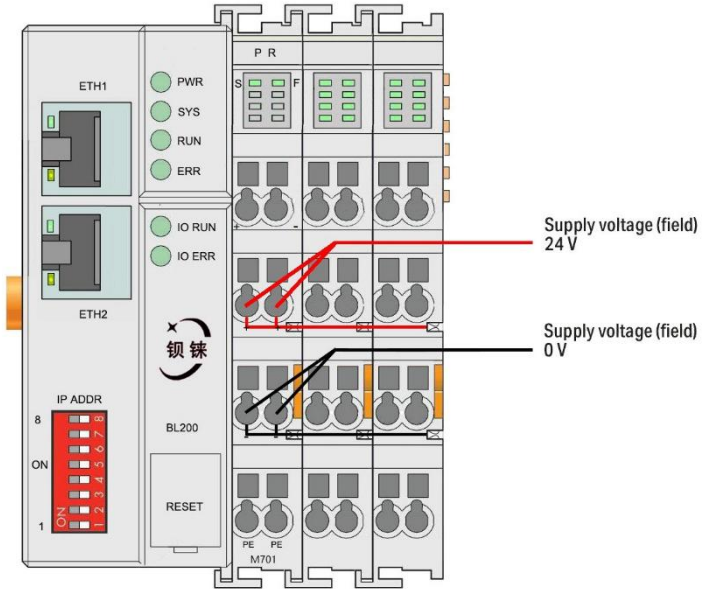


图 21：连接现场电源示意图

在连接 I/O 模块时，现场侧电源自动从电源跨接触点输出。电源跨接触点的负载持续电流不得超过 10 A。

可以通过插入额外的电源模块，解决系统侧或现场侧负载功率过大的问题。在插入额外的电源模块后，现场侧可能会出现一个新的电压电位。
在不需要电气隔离的情况下，现场电源和系统电源可以使用同一路电源。

4.2.3 接地

在安装外壳机柜时，必须将机柜接地，导轨通过螺钉和机柜建立电气连接，实现导轨良好接地。接地可以增加抗电磁干扰能力。 I/O 系统中的某些组件具有导轨触点，可将电磁干扰消耗到导轨上。

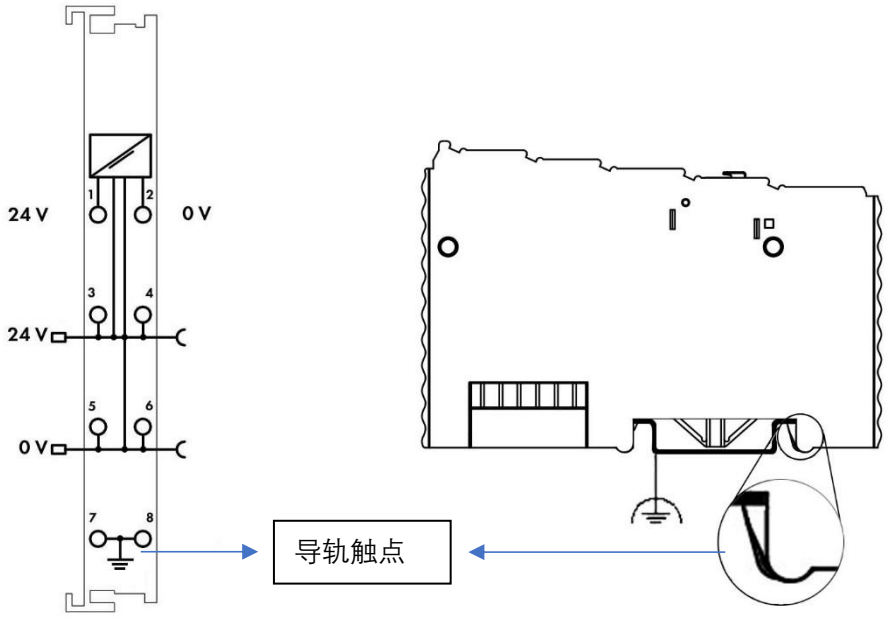


图 22：DIN 导轨触点

4.3连接总线

BL200 系列耦合器有 2 个以太网接口，内部集成交换机功能，工作在存储转发操作模式，每个端口支持 10/100 Mbit 的传输速度以及全双工和半双工传输模式。

该耦合器只能通过 ETH 2 接口连接到基于以太网的现场总线中，EHT1 用于连接其他需要接入以太网的节点，这样两个接口形成菊花链拓扑。

内部集成的交换机支持旁路模式，当耦合器系统故障时可以自动启动旁路模式，自动维持 ETH 1 和 EHT 2 的链接。

这些以太网接口的接线符合 100BaseTX 的规范，该规范规定使用 5 类双绞线电缆作为连接电缆。可以使用最大长为 100 m 的电缆类型 S/UTP（屏蔽非屏蔽双绞线）和 STP（屏蔽双绞线）。

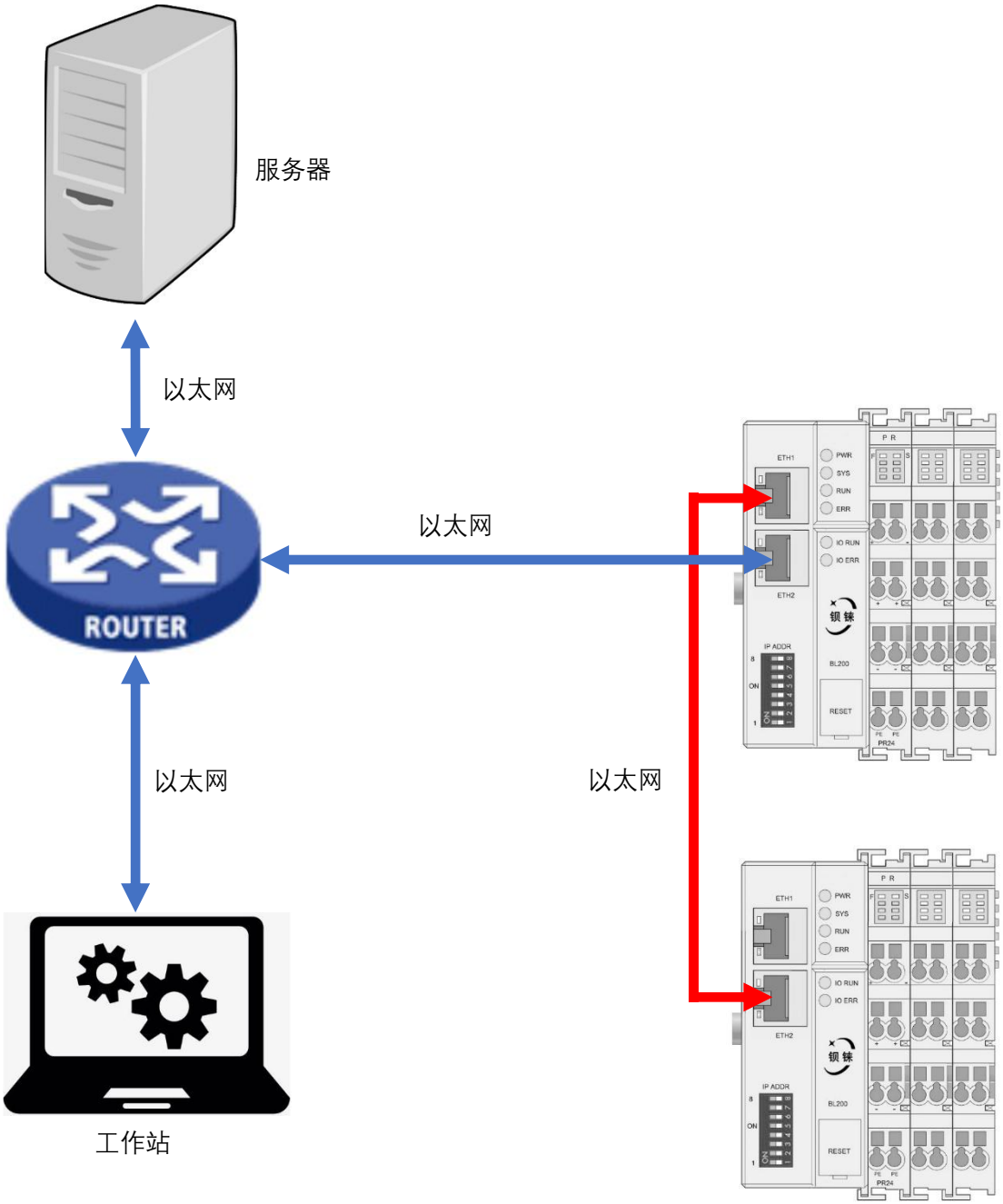


图 23：连接总线到网络

还可以通过 ETH 2 和计算机直连。

图 24：连接总线到计算机

5. 网页配置

BL200 耦合器内置 web 服务器是一个基于浏览器的配置实用程序。当节点连接到您网络时，您可以在 Web 浏览器中输入服务器的 IP 地址以访问 Web 控制台。

5.1配置前准备

想要顺利的访问 BL200 耦合器节点，必须要将它正确的安装并且和计算机建立连接。除此之外还要给它们配置正确的 IP 地址，使他们保持在同一个网段内。

5.1.1 连接计算机和耦合器

1. 将现场总线节点安装到 DIN35 导轨上。按照“产品安装”章节中的安装说明进行操作。
2. 将 24 V 电源连接到系统电源端子。
3. 计算机和总线节点可以通过 2 种方式建立连接，一种是两者通过以太网接口接入本地局域网的交换机设备；另一种是两者点对点直接相连。详细步骤请按照“连接总线”章节中的说明进行操作。
4. 将电源开机，开始供电。

耦合器上电工作后被初始化，同时根据现场总线节点的 I/O 模块配置并创建过程映像。

5.1.2 配置计算机 IP 地址

在 PC 端，有两种方法配置其 IP 地址；一种是在 PC 的本地连接上开启自动获取 IP 地址选项，通过网络中 DHCP 动态分配；另一种是在 PC 的本地连接上配置一个跟耦合器节点处于同一个网段的静态 IP 地址。

下面以 Windows 7 系统为例进行配置。Windows 系统的配置均相似。

1. 单击“开始 > 控制面板 > 网络和共享中心”，在打开的窗口中单击“本地连接”。

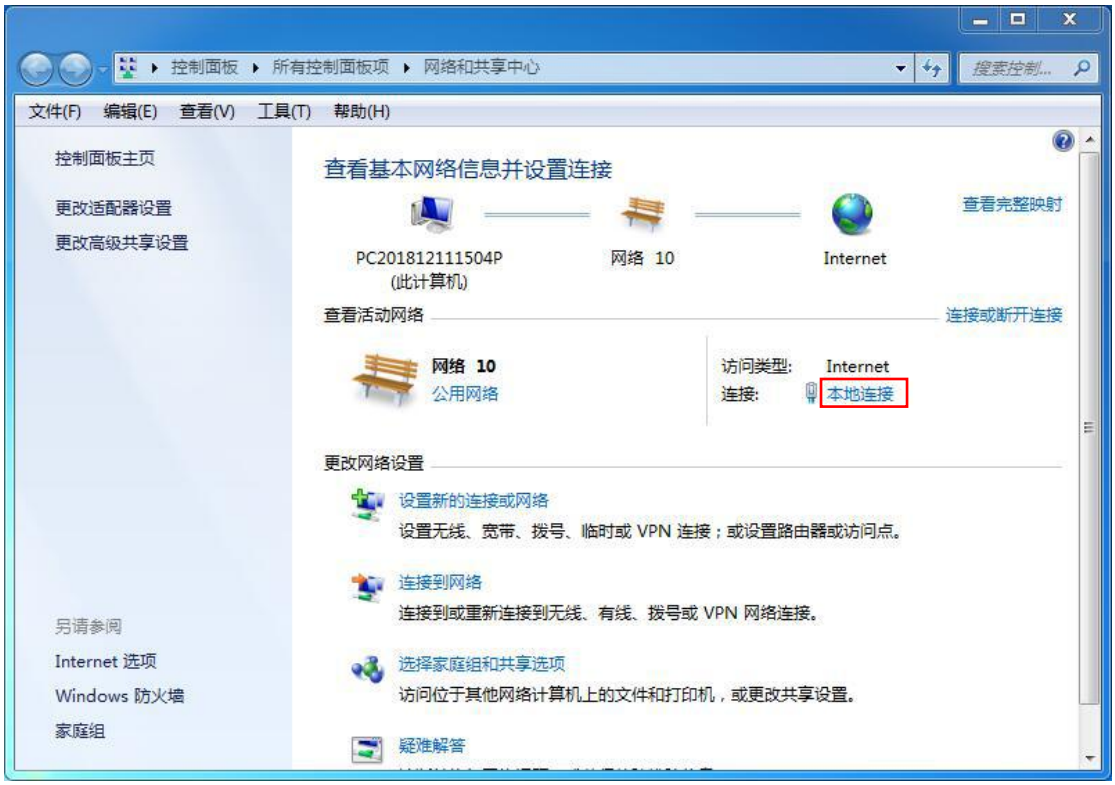


图 25：网络和共享中心

2. 在“本地连接状态”窗口中，单击“属性”。

图 26：本地连接状态

3. 在本地连接属性页面双击“Internet 协议版本 4（TCP/IPv4）”

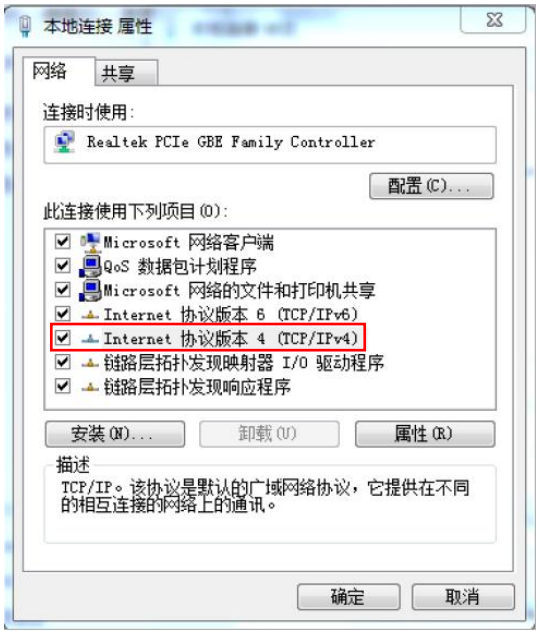


图 27：本地连接属性

4. 两种方法配置 PC 的 IP 地址：

■ 自动获得 IP 地址（系统默认模式）

自动从 DHCP 服务器获取 IP 地址，选中“自动获得 IP 地址”;

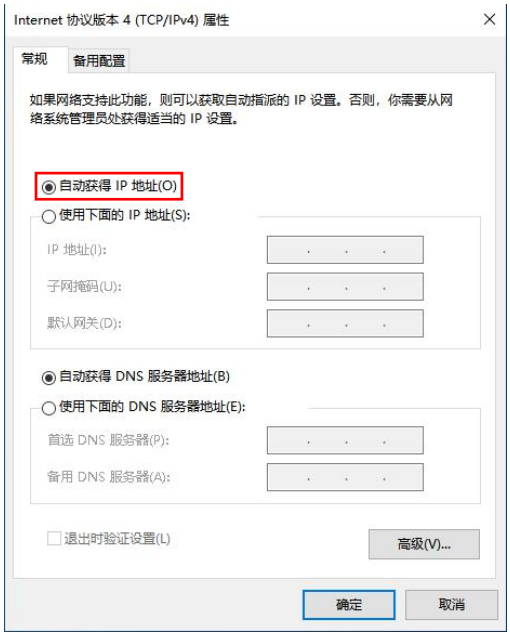


图 28：自动获得 IP 地址

■ 设置静态 IP 地址

选中 “使用下面的 IP 地址”，在 IP 地址，子网掩码和默认网关设置正确的数值。

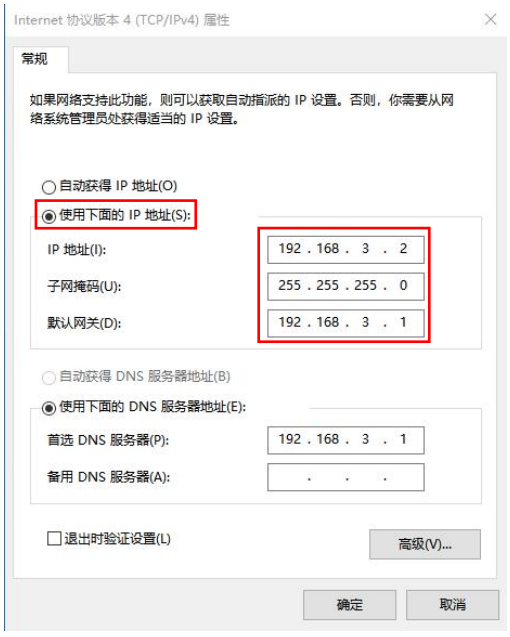


图 29：设置静态 IP

5.1.3 配置节点 IP 地址

该节点有 4 种方式分配 IP 地址:

- 通过 DHCP 自动分配 IP（动态 IP）
- 通过启用内置固定 IP（静态 IP）
- 通过内置的 web 页面分配（静态 IP）
- 通过拨码选择开关分配（静态 IP）

设备出厂默认启用“通过启用内置固定 IP（静态 IP）”，IP 为 192.168.1.10。重新分配 IP 地址后需要重启设备生效。

各种方式的开启和关闭通过 DIP 地址选择开关定义。

表 8：DIP 开关位置定义

开关位置（ON = 1）	值	功能
0000 0000	0	启用内置固定 IP：192.168.1.10，关闭其他分配 IP 地址功能。
0000 0001 --- 1111 1110	1-254	当通过 web 分配 IP 时候，启用 web 页面分配功能，手动设置的 IP 生效； 当没有通过 web 分配 IP 时候，启用拨码选择开关分配功能，并确定第 3 个字节的值。 示例：0010 0110(十进制 22)，IP 地址为“192.168.22.253”
1111 1111	255	启用 DHCP 自动分配功能，关闭其他分配 IP 地址功能。

5.1.3.1 通过启用内置固定 IP

将 DIP 地址选择开关的值设置为 0000 0000（十进制 0），启用内置固定 IP 功能。此时 IP 地址为 192.168.1.10，通过这种方式设置的 IP 地址是静态的。

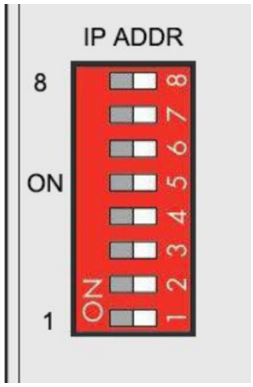


图 30：拨码开关-启用内置固定 IP

5.1.3.2 通过 DHCP 自动分配

将 DIP 地址选择开关的值设置为 1111 1111（十进制 255），开启 DHCP 功能。此时 IP 地址由网络中 DHCP 服务器动态分配，通过这种方式设置的 IP 地址是动态的。

请注意，通过 DHCP 分配的 IP 地址是有时间限制的，当网络中 DHCP 不可用时，IP 地址会变为空闲，无法访问现场总线节点。为了永久使用 IP 地址，请将其更改为“静态”。

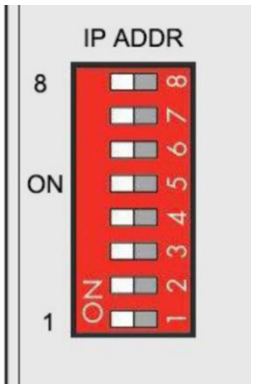


图 31：拨码开关-通过 DHCP 自动分配

5.1.3.3 通过 web 页面配置

现场总线耦合器还可以在进入页面后通过“设置 > 本机设置“ 页面设置 IP 地址，通过这种方式设置的 IP 地址是静态的。

5.1.3.4 通过拨码选择开关分配

将 DIP 地址选择开关的值设置为 0000 0001 - 1111 1110（十进制 1 - 254），将通过拨码开关分配 IP 地址。

该 IP 地址由固定字节和可变字节两部分组成。第 1、2 和 4 三个字节为固定字节，拨码选择开关确定第 3 个字节，即：192.168.xxx.253。

现场总线耦合器通过拨码选择开关分配 IP 地址，通过这种方式设置的 IP 地址是静态的。

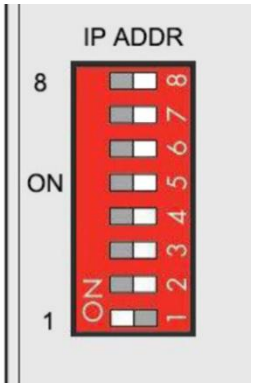


图 32：拨码开关-通过拨码选择开关分配（示例：192.168.1.253）

5.1.4 出厂默认设置

登录 Web 配置页面前，您有必要了解以下的默认参数。

表 9：IP 出厂默认参数

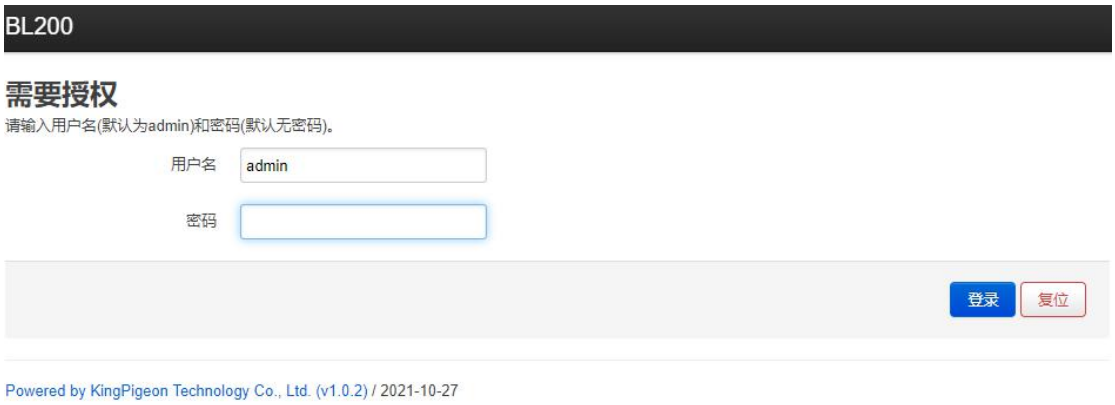
项目	描述
登录 IP 地址（出厂默认）	192.168.1.10
用户名	admin
密码	无

5.1.5 登录配置页面

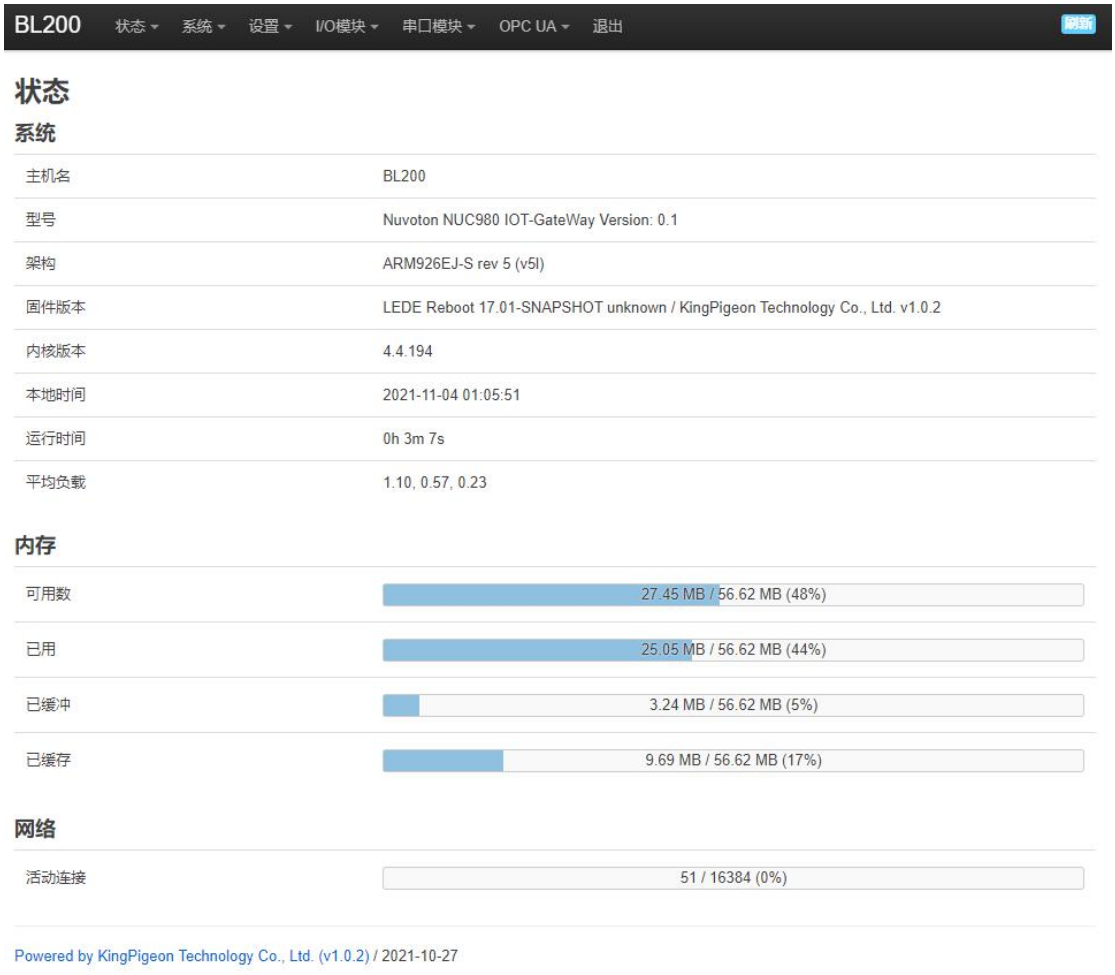
1. 在计算机上打开浏览器，如 IE、 Chrome、Firefox 等。

2. 在浏览器的地址栏输入耦合器节点的 IP 地址(默认 192.168.1.10)进入用户登录界面;

3. 在登录界面输入“用户名“ 和 ”密码“，然后点击登录。



4. 成功登录 web 界面显示如下

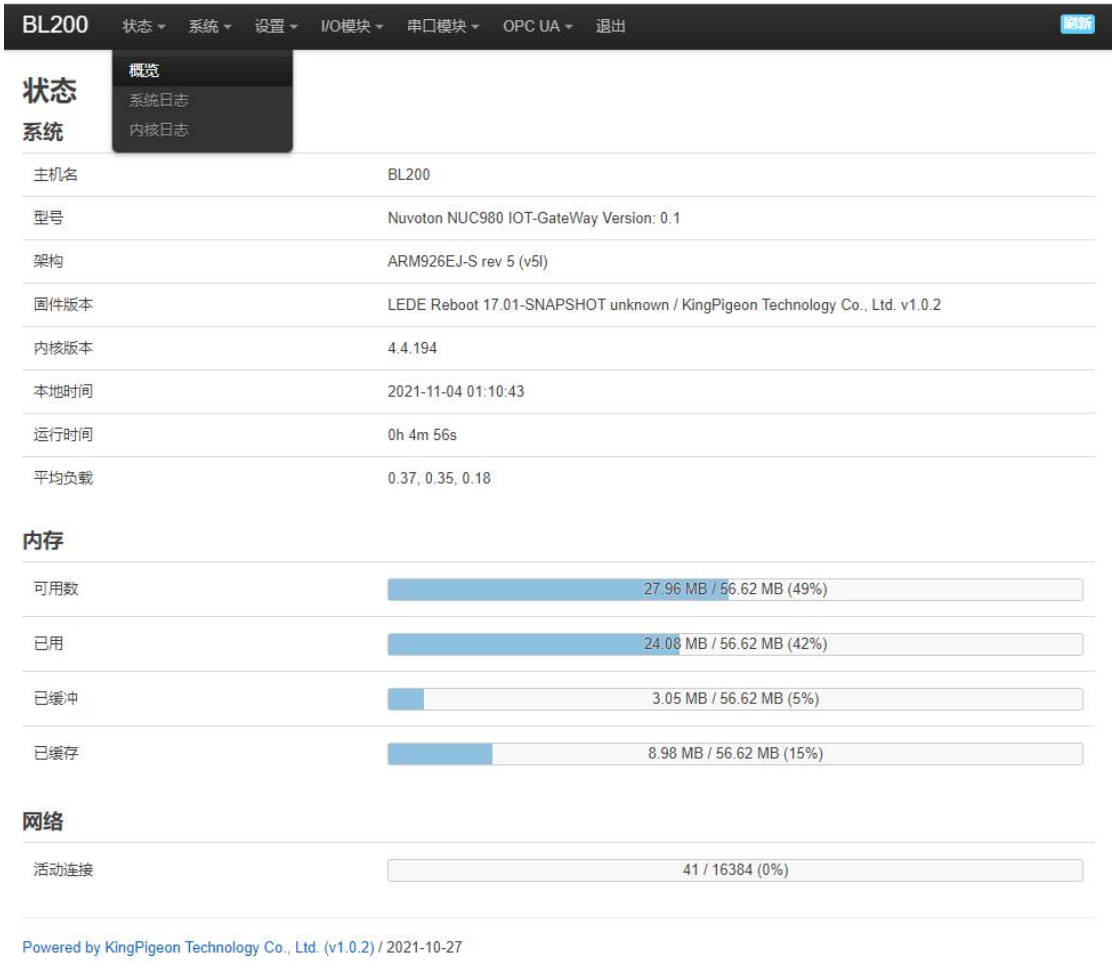


5. 配置完参数后，需要点击页面上的“保存并应用”按钮才能生效。

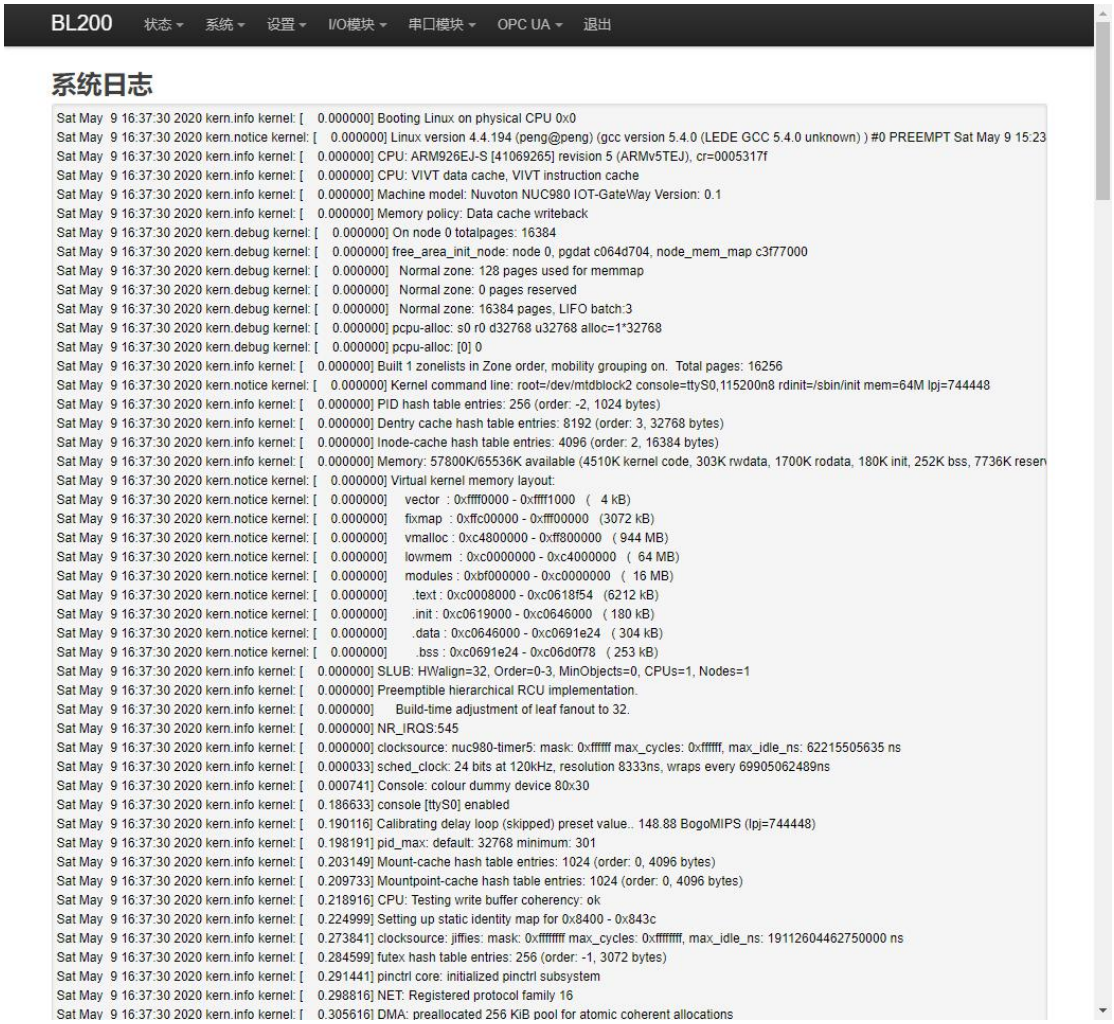
5.2状态

在状态菜单中，提供了概览、系统日志和内核日志，可以查看设备参数和设备运行状态。

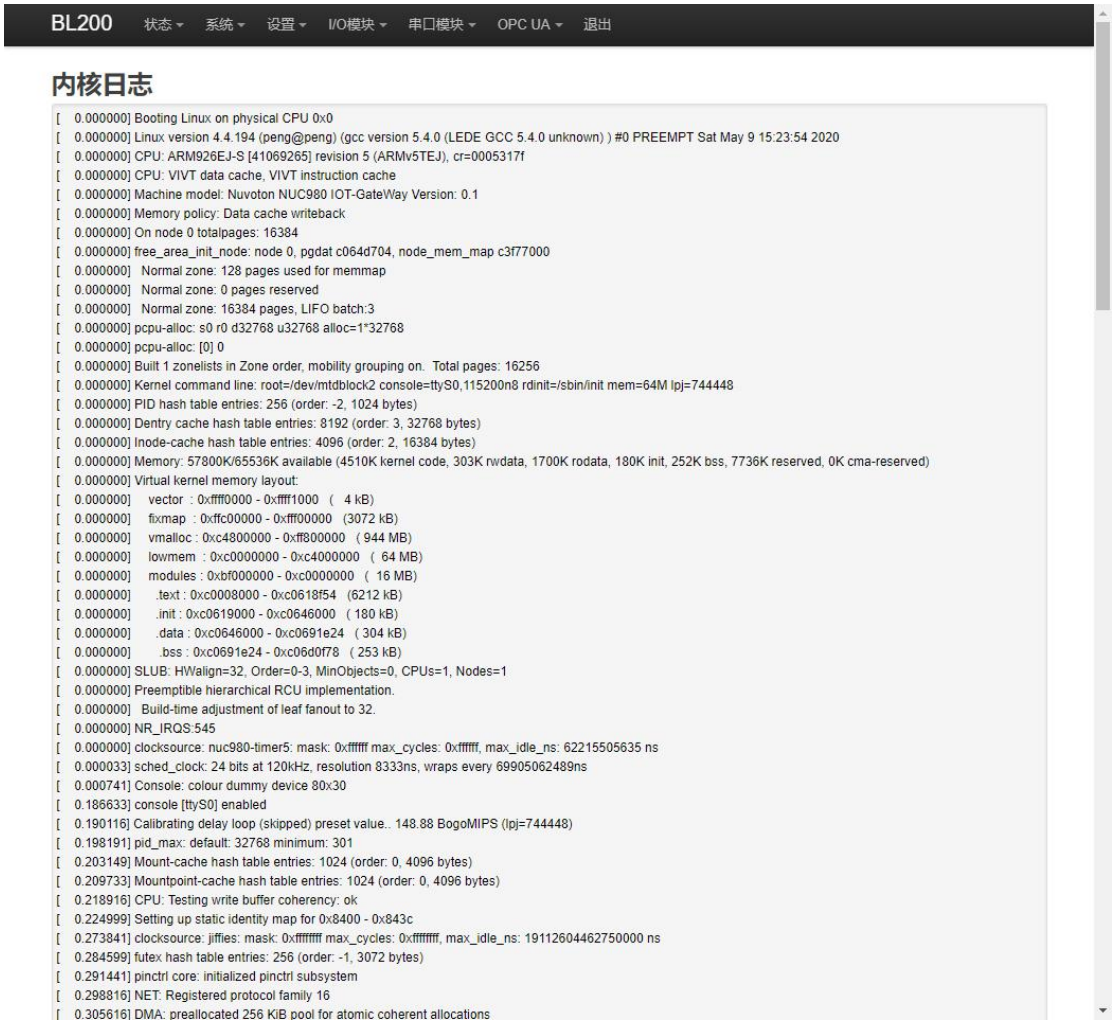
状态 > 概览



状态 > 系统日志


BL200
状态
系统
设置
I/O模块
串口模块
OPC UA
退出
系统日志
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Booting Linux on physical CPU 0x0
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] Linux version 4.4.194 (peng@peng) (gcc version 5.4.0 (LEDE GCC 5.4.0 unknown)) #0 PREEMPT Sat May 9 15:23
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=0005317f
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] CPU: VIVT data cache, VIVT instruction cache
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Machine model: Nuvoton NUC980 IOT-GateWay Version: 0.1
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Memory policy: Data cache writeback
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] On node 0 totalpages: 16384
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] free_area_init_node: node 0, pgdat c064d704, node_mem_map c3f77000
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] Normal zone: 128 pages used for memmap
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] Normal zone: 0 pages reserved
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] Normal zone: 16384 pages, LIFO batch:3
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
Sat May 9 16:37:30 2020 kern.debug kernel: [0.000000] pcpu-alloc: [0] 0
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] Kernel command line: root=/dev/mtdblock2 console=ttyS0,115200n8 rdinit=/sbin/init mem=64M lpj=744448
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] PID hash table entries: 256 (order: -2, 1024 bytes)
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Memory: 57800K/65536K available (4510K kernel code, 303K rwdata, 1700K rodata, 180K init, 252K bss, 7736K reser
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] Virtual kernel memory layout:
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] vector : 0xffff0000 - 0xffff1000 (4 kB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] fixmap : 0xffc00000 - 0xfff00000 (3072 kB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] vmalloc : 0xc4800000 - 0xff800000 (944 MB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] lowmem : 0xc0000000 - 0xc4000000 (64 MB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] modules : 0xbf000000 - 0xc0000000 (16 MB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] .text : 0xc0008000 - 0xc0618f54 (6212 kB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] .init : 0xc0619000 - 0xc0646000 (180 kB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] .data : 0xc0646000 - 0xc0691e24 (304 kB)
Sat May 9 16:37:30 2020 kern.notice kernel: [0.000000] .bss : 0xc0691e24 - 0xc06d0f78 (253 kB)
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] SLUB: HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Preemptible hierarchical RCU implementation.
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] Build-time adjustment of leaf fanout to 32.
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] NR_IRQS:545
Sat May 9 16:37:30 2020 kern.info kernel: [0.000000] clocksource: nuc980-timer5: mask: 0xffffff max_cycles: 0xffffff, max_idle_ns: 62215505635 ns
Sat May 9 16:37:30 2020 kern.info kernel: [0.000033] sched_clock: 24 bits at 120kHz, resolution 8333ns, wraps every 69905062489ns
Sat May 9 16:37:30 2020 kern.info kernel: [0.000741] Console: colour dummy device 80x30
Sat May 9 16:37:30 2020 kern.info kernel: [0.186633] console [ttyS0] enabled
Sat May 9 16:37:30 2020 kern.info kernel: [0.190116] Calibrating delay loop (skipped) preset value.. 148.88 BogoMIPS (lpj=744448)
Sat May 9 16:37:30 2020 kern.info kernel: [0.198191] pid_max: default: 32768 minimum: 301
Sat May 9 16:37:30 2020 kern.info kernel: [0.203149] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
Sat May 9 16:37:30 2020 kern.info kernel: [0.209733] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
Sat May 9 16:37:30 2020 kern.info kernel: [0.218916] CPU: Testing write buffer coherency: ok
Sat May 9 16:37:30 2020 kern.info kernel: [0.224999] Setting up static identity map for 0x8400 - 0x843c
Sat May 9 16:37:30 2020 kern.info kernel: [0.273841] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
Sat May 9 16:37:30 2020 kern.info kernel: [0.284599] futex hash table entries: 256 (order: -1, 3072 bytes)
Sat May 9 16:37:30 2020 kern.info kernel: [0.291441] pinctrl core: initialized pinctrl subsystem
Sat May 9 16:37:30 2020 kern.info kernel: [0.298816] NET: Registered protocol family 16
Sat May 9 16:37:30 2020 kern.info kernel: [0.305616] DMA: preallocated 256 KiB pool for atomic coherent allocations

系统 > 内核日志



5.3系统

5.3.1 系统

系统属性 > 常规设置

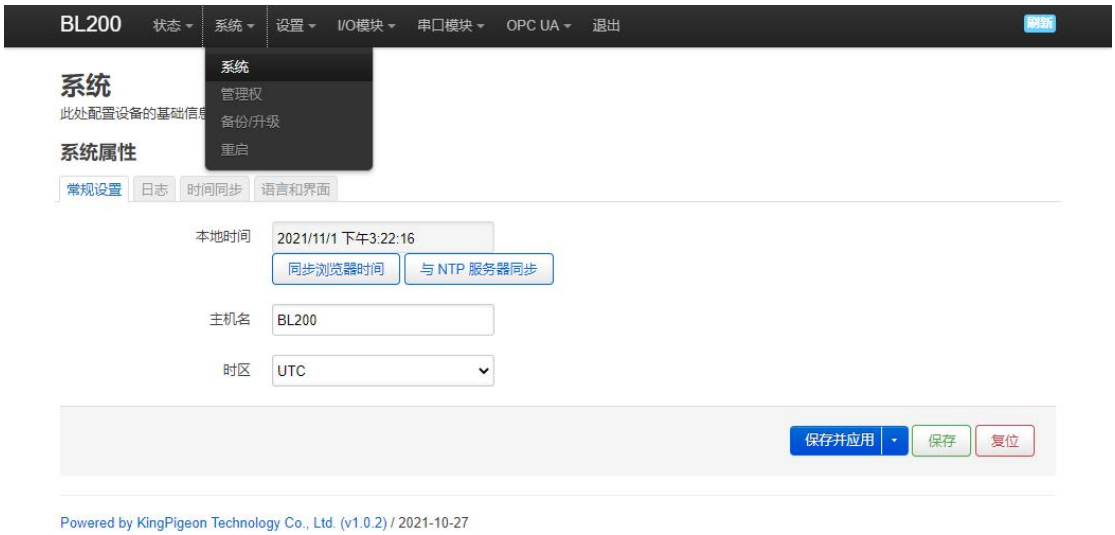


表 10：系统 > 系统属性 > 常规设置

项目	描述	默认
本地时间	显示设备当前时间。可以点击“同步浏览器时间”或者“与 NTP 服务器同步”按钮更新设备时间。	--
主机名	可以自定义设备名称，便于区分多个设备。	BL200
时区	可以通过下拉菜单选择时区	UTC

系统属性 > 日志

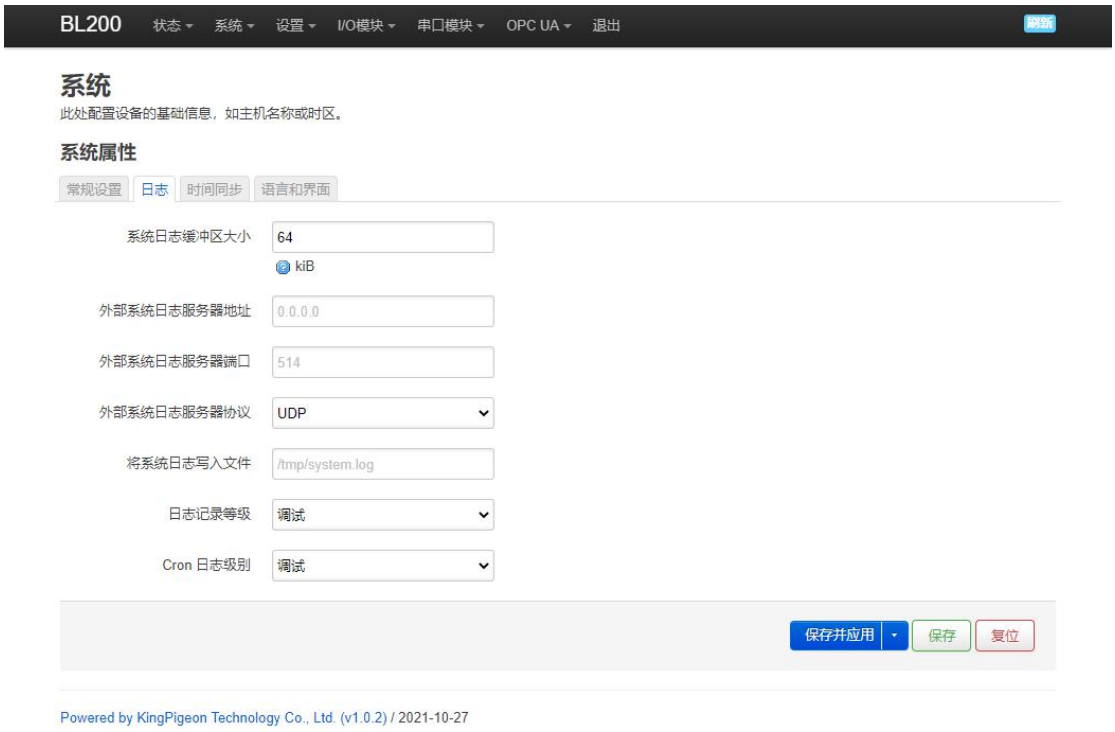
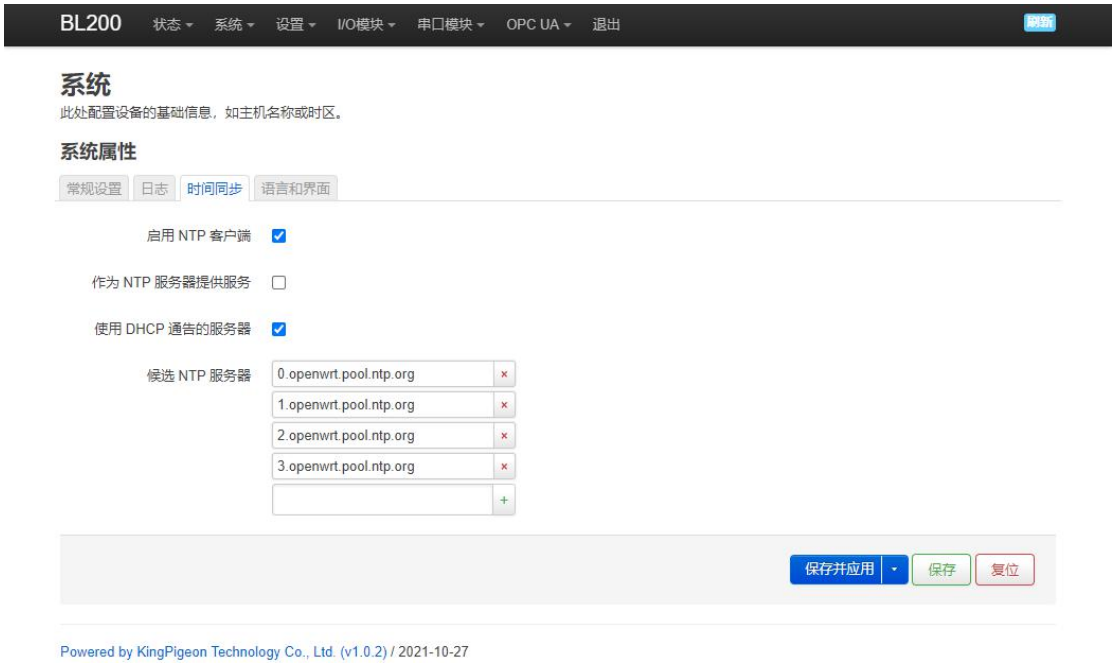


表 11：系统 > 系统属性 > 日志

项目	描述	默认
系统日志缓冲区大小		64
外部系统日志服务器地址		
外部系统日志服务器端口		
外部系统日志服务器协议		
将系统日志写入文件		
日志记录等级		
Cron 日志级别		

系统属性 > 时间同步

可以设置 NTP 服务器，用来同步时间。



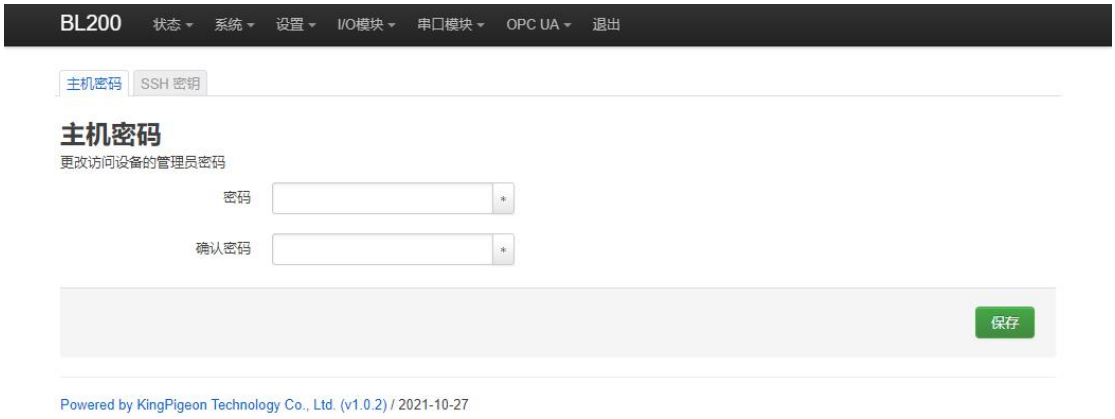
系统属性 > 语言和界面

表 12：系统 > 系统属性 > 语言和界面

项目	描述	默认
语言	可以在 auto、 English、中文（Chinese）种选	auto
主题	目前只支持 Bootstrap。	Bootstrap

5.3.2 管理权

管理权 > 主机密码
更改访问设备的管理员密码



管理权 > SSH 密钥
与使用普通密码相比，公钥允许无密码 SSH 登录具有更高的安全性。要将新密钥上传到设备，请粘贴 OpenSSH 兼容的公钥行或将 `.pub` 文件拖到输入字段中。

5.3.3 备份/升级

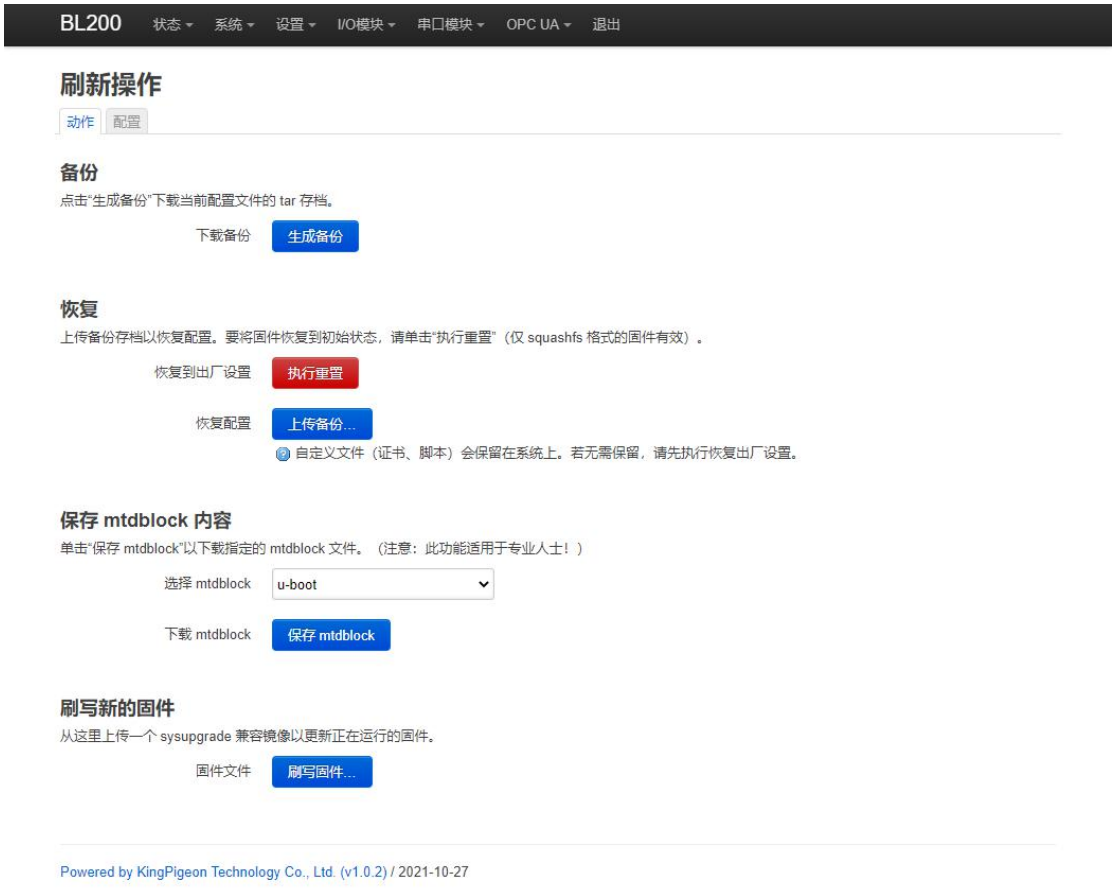


表 13：系统 > 备份/升级 > 动作

项目	描述	默认
备份	点击"生成备份"下载当前配置文件的 tar 存档。	--
恢复	上传备份存档以恢复配置。要将固件恢复到初始状态，请单击"执行重置"（仅 squashfs 格式的固件有效）。	--
保存 mtdblcok 内容	单击"保存 mtdblock"以下载指定的 mtdblock 文件。（注意：此功能适用于专业人士！）	--
刷写新的固件	从这里上传一个 sysupgrade 兼容镜像以更新正在运行的固件。	--

5.3.4 重启

点击"执行重启"将重启您的设备

5.4设置

本机设置

表 14：设置 > 本机设置

项目	描述	默认
Modbus 设备 ID	Modbus 设备 ID 范围 1~247	1
拨码开关地址	显示拨码开关设置的 IP 地址	192.168.1.10
设置设备 IP 地址	可以自定设置设备 IP 地址，设置后需要重启生效。	--
Modbus TCP 端口	Modbus TCP 协议端口号，可以自定义设置。	502

5.5I/O 模块

上电后，耦合器自动识别所有与之相连的 I/O 模块，并根据模块的类型、数据宽度和模块在节点中的位置创建内部本地过程映像。

如果添加、更改或移 除 I/O 模块，会建立新的过程映像，过程数据地址会改变。在添加 I/O 模块时，则必须考虑所有先前 I/O 模块的过程数据。

控制器最多可连接 32 个 I/O 模块，包括数字输入输出，模拟输入输出和特殊功能模块。

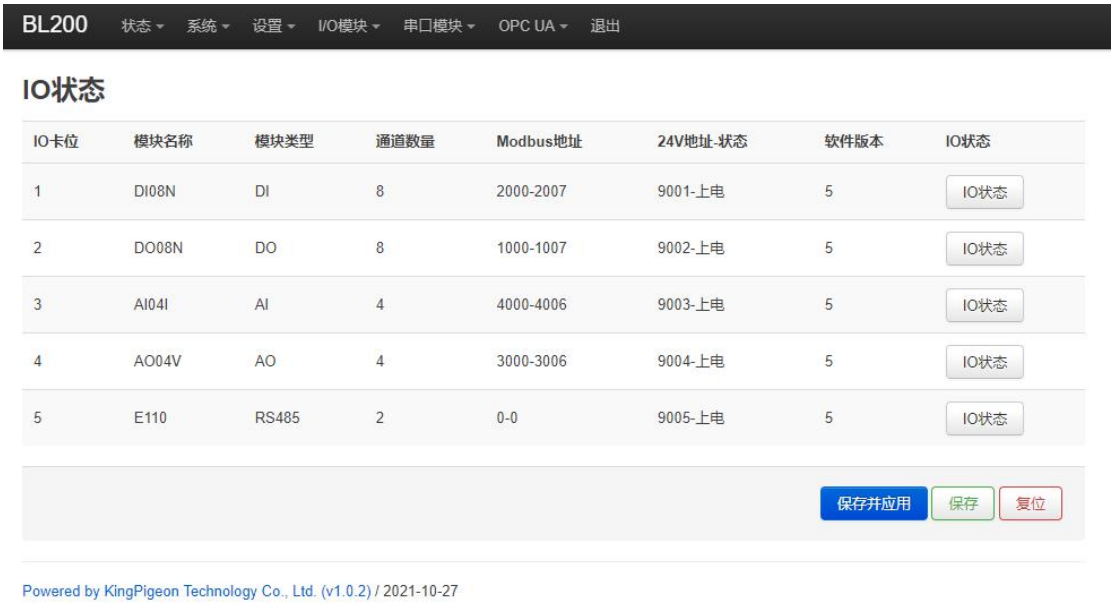


表 15：I/O 模块 > I/O 状态

项目	描述
IO 卡位	IO 模块在卡槽中的排序，第一个靠近耦合器的模块卡位为 1，后面的依次为 2 3 4… …
模块名称	IO 模块的详细型号
模块类型	IO 模块功能类型
通道数量	IO 模块的数据宽度
Modbus 地址	IO 模块在耦合器内部的过程映射地址
24V 地址-状态	IO 模块现场侧供电状态，数字量，占 1 个比特位
软件版本	IO 模块内部固件版本
IO 状态	点击后可以查看和设置不同类型 IO 模块的参数

5.5.1 数字输入模块

数字输入模块可以提供两种类型的数据，一种是当前输入的状态值，布尔类型；另一种是计数器数值，32 位数值型，支持清除功能。

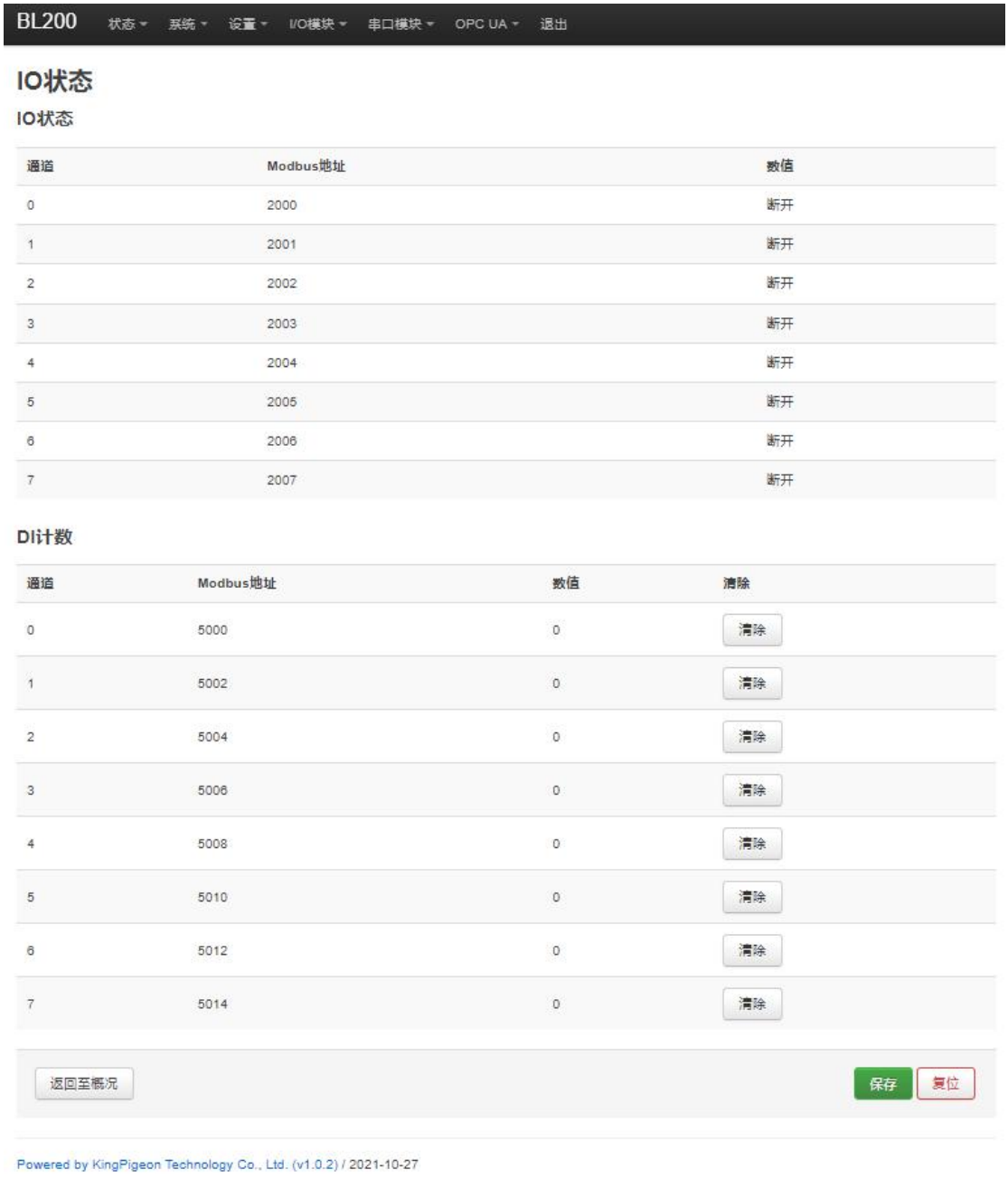


表 16：数字输入模块 > IO 状态

项目	描述
通道	数字输入模块的通道编号
Modbus 地址	布尔状态数据在耦合器内部的过程映射地址
数值	显示当前的输入状态，断开：逻辑 0，闭合：逻辑 1

表 17：数字输入模块 > DI 计数

项目	描述
通道	数字输入模块的通道编号
Modbus 地址	计数值在耦合器内部的过程映射地址
数值	显示当前的输入计数值，32 位无符号整型
清除	可以清除当前通道计数器值

5.5.2 数字输出模块

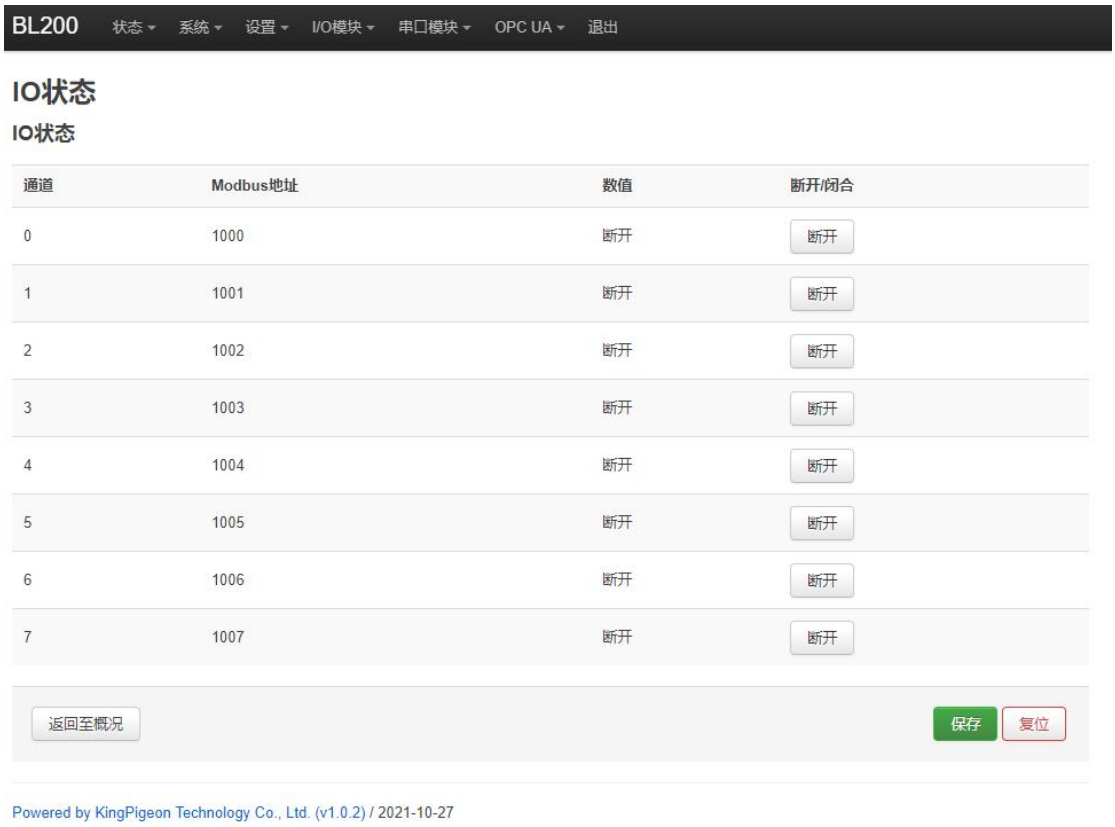


表 18：数字输出模块

项目	描述
通道	数字输出模块的通道编号
Modbus 地址	数字输出布尔数据在耦合器内部的过程映射地址
数值	显示当前的输出状态，断开：0，闭合：1
断开/闭合	可以控制当前通道输出状态

5.5.3 模拟输入模块

模拟输入（AI）类型模块支持通过耦合器 web 页面设置参数，这样模块内部自动实现数据换算，可以直接输出与传感器对应的实际工程数值。

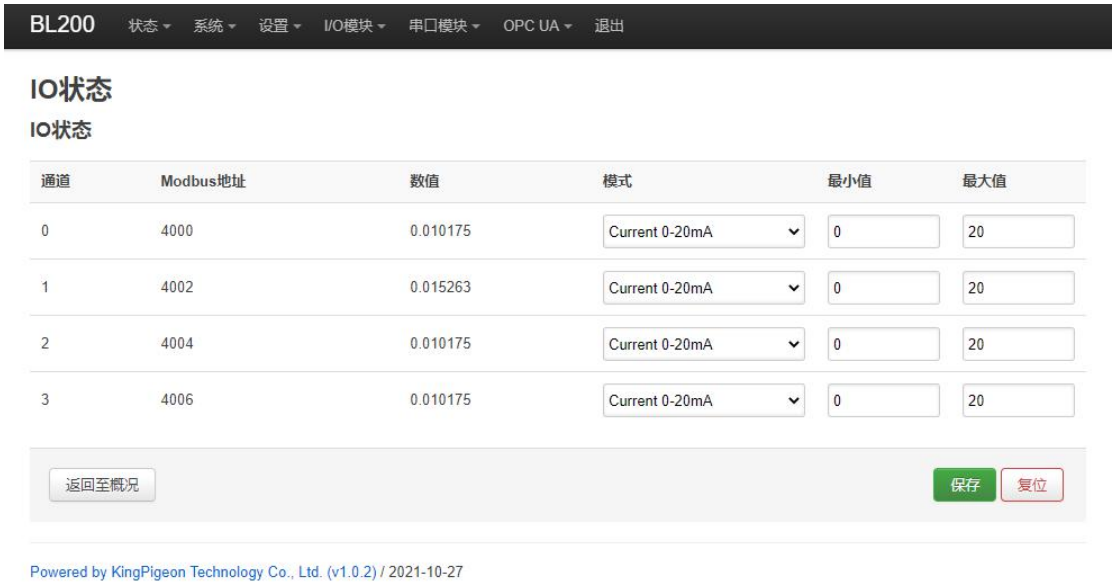


表 19：模拟输入模块

项目	描述
通道	模拟输入模块的通道编号
Modbus 地址	模拟输入模块在耦合器内部的过程映射地址
数值	显示当前通道输入的实际工程数值，32 位单精度浮点型
模式	不同型号的模拟输入模块有不同的选项，详细参照具体模拟输入 I/O 模块手册。
最小值	传感器量程最小值
最大值	传感器量程最大值

模拟输入模块的电信号数值（一般是传感器）和实际工程数值存在线性关系，它们的公式如下（以 4-20mA 为例）：

实际工程数值 =（电流值 - 4）*（(最大值 – 最小值）/（20 - 4））+最小值

以 4-20mA 类型的水位传感器测量水塔深度为例：

已知水位传感器量程是 0-100m，电流数据 5.6mA，计算水塔深度：
代入公式：

(5.6 - 4) * ((100 - 0) / (20 - 4)) + 0 = 10

水塔深度是 10m；

5.5.4 模拟输出模块

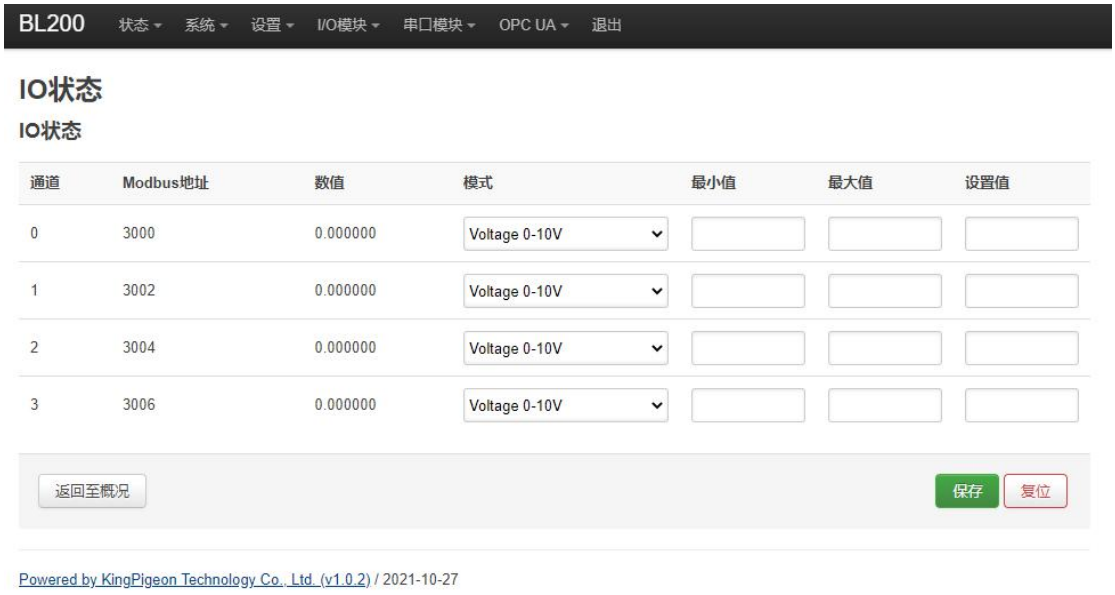


表 20：*模拟输出模块*

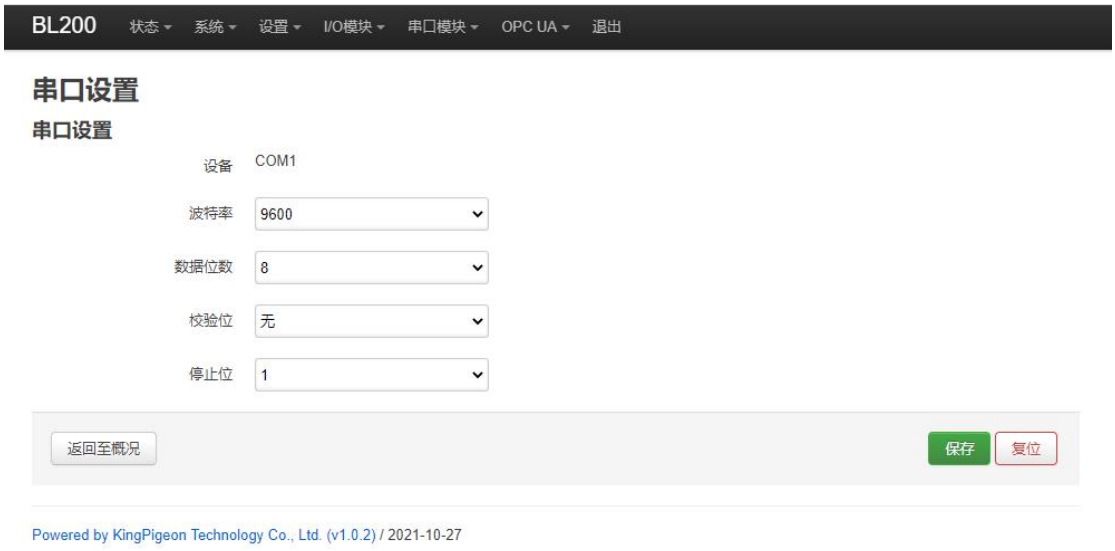
项目	描述
通道	模拟输出模块的通道编号
Modbus 地址	模拟输出模块在耦合器内部的过程映射地址
数值	显示当前通道输出的实际工程数值，32 位单精度浮点型
模式	不同型号的模拟输出模块有不同的选项，详细参照具体模拟输出 I/O 模块手册。
最小值	实际工程数值最小值
最大值	实际工程数值最大值
设置值	可以设置输出所需要的实际工程数值

5.6串口模块

可以通过串口模块接入各种支持 Modbus RTU 协议的传感器，终端等设备。串口模块内置 Modbus RTU (Master)协议，把外部传感器数据通过本地总线和耦合器建立过程映射关系。

5.6.1 串口设置

点击右侧的设置按钮进入设置页面



5.6.2 Modbus 设置

Modbus 设置用于串口通信 I/O 模块添加 Modbus RTU 设备。

在输入框输入自定义数据名称，点击添加

在右侧点击编辑

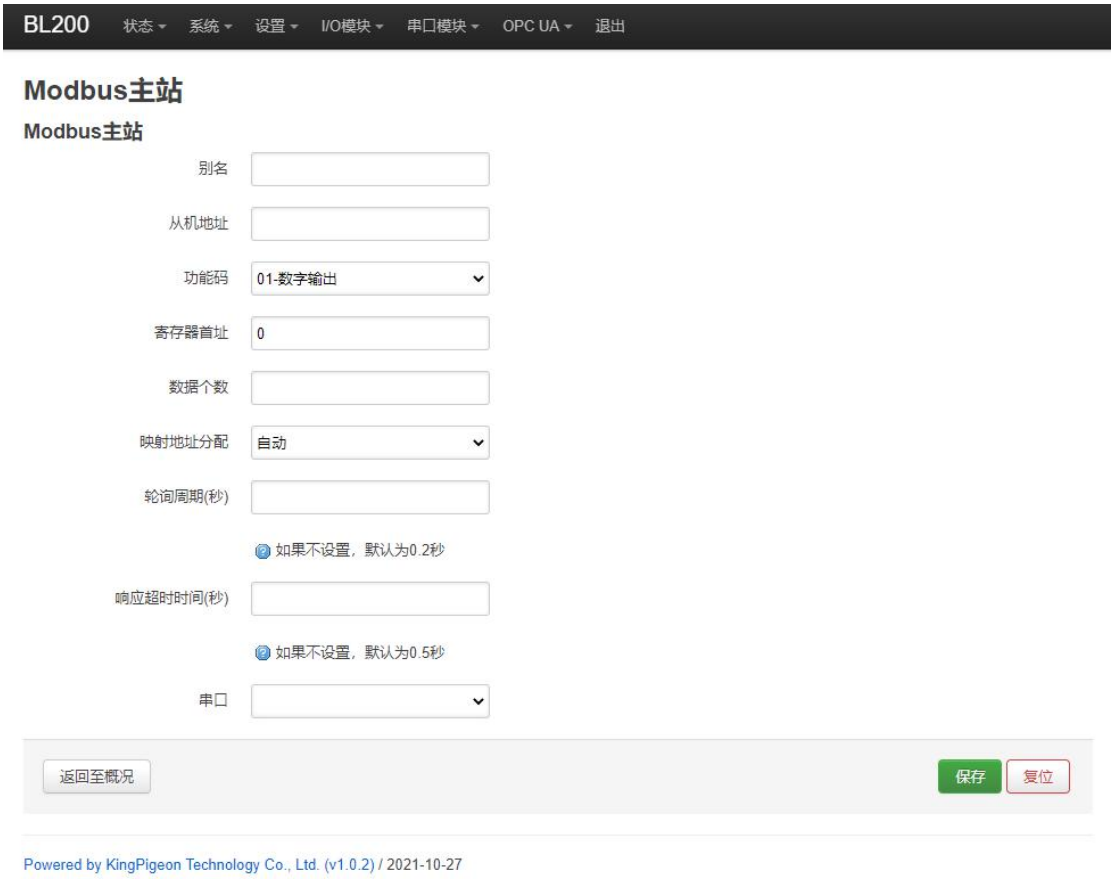


表 21：Modbus 设置

项目	描述
别名	设备昵称可以用与区分数据
从机地址	从站设备地址，范围 0-247
功能码	根据从站数据类型选择，包括：“01”、“02”、“03”、“04”
寄存器首址	从站数据的寄存器起始地址
数据个数	从机数据的个数
映射地址分配	支持分配方式 ● 自动 系统按照不同的数据类型，按照映射起始地址自动依次往下分配，地址是连续的。 ● 手动 手动分配的方式可以让映射地址跨段分配
轮询周期(秒)	相邻两条轮询指令之间的间隔时间
响应超时时间(秒)	发送指令给从站后，等待从站返回数据的最长时间，超过该时间，将会认为从站无应答。
串口	选择串口通道

5.7OPC UA

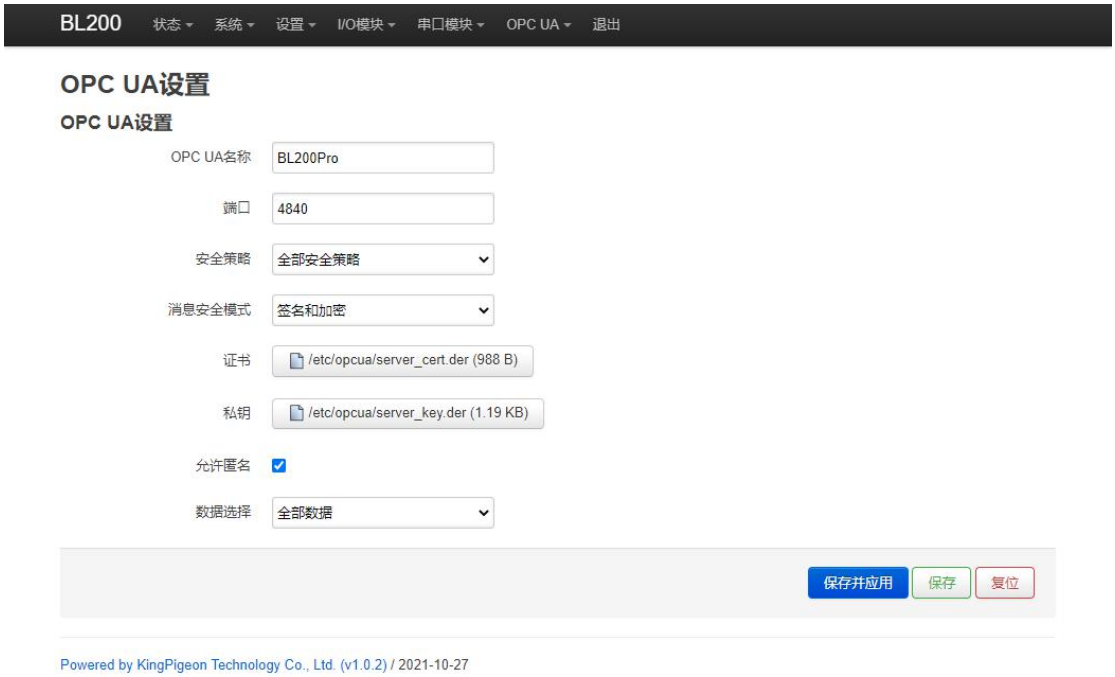


表 22：*OPC UA* ***设置***

项目	描述	默认
OPC UA 名称	OPC UA 服务器名称	
端口	OPC UA 服务端口号	4840
安全策略	无、 basic128rsa15 basic256 basic256sha256 aes128sha256rsaoaep 全部安全策略	无
消息安全模式	签名 签名和加密	
证书	OPC UA 证书,点击上传的证书,加载到配置页面。	
私钥	OPC UA 密钥,点击上传的证书,加载到配置页面。	
允许匿名	是否开启用户名和密码登录	
用户名	填写用户名	
密码	填写用户名密码	
数据选择	全部数据点 选择数据点 信息模型	全部数据
选择数据点	可以选择你要读的数据点。“数据选择”项选择“选择数据点”才有这项	
模型文件（.xml）	上传信息模型(.xml)文件，“数据选择”项选择“信息	

	模型"才有这项	
依赖模型文件	选择要参照的信息模型数量,最多可以选择 5 个.	
依赖的模型 1-5	上传要参照的信息模型(.xml)文件	

说明：自定义的信息模型，建模时数据点描述项一定要是 REG +Modbus 地址的格式，如 DO1 点描述项填写 REG1000，其他项自定义。

6. 现场总线通信

6.1 Modbus

6.1.1 概述

Modbus 是一种独立于制造商的开放式现场总线标准协议，适用于制造和过程自动化中的各种应用。

MODBUS 是一种应用层消息传递协议，位于 OSI 模型的第 7 层，可以在不同类型的总线或网络上连接的设备之间进行客户端/服务器通信。

几种常用的网络如下：

- TCP/IP over Ethernet。
- 多种媒体异步串行传输（有线：EIA/TIA-232-E、EIA-422、EIA/TIA-485-A；光纤、无线电等）。
- MODBUS PLUS，高速令牌。

MODBUS 是一种请求/应答协议，提供由功能代码指定的服务。

MODBUS 协议允许在所有类型的网络架构内轻松通信。

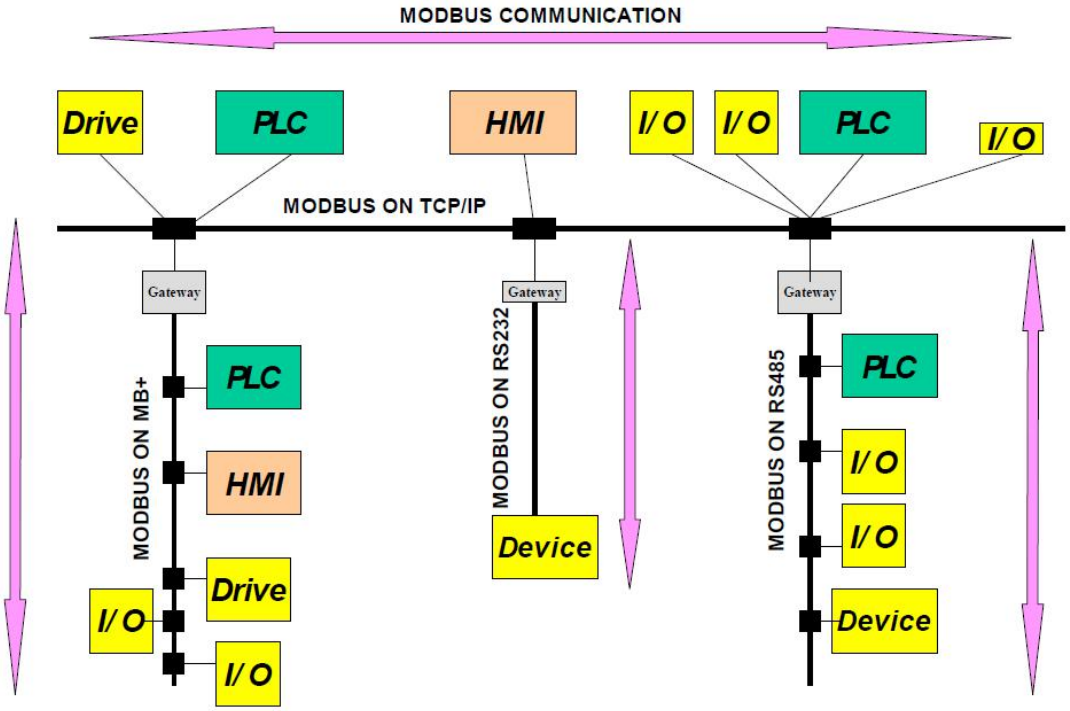


图 33：Modbus 网络架构

MODBUS 协议定义了一个独立于底层通信层的简单协议数据单元 (PDU)。 MODBUS 协议在特定总线或网络上的映射可以在应用数据单元（ADU）上引入一些附加字段。

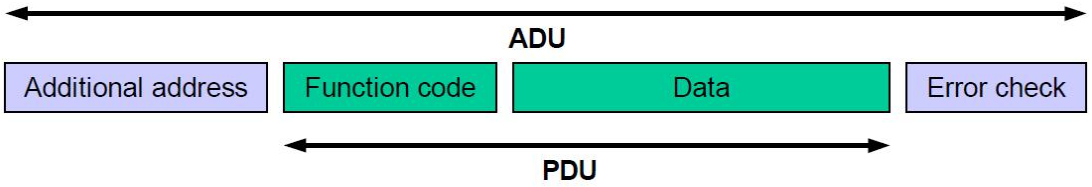


图 34：Modbus 数据帧

更多关于 Modbus 开放协议规范的细节可以在网站 www.modbus.org 查看。

6.1.1 Modbus TCP

Modbus TCP 协议是 Modbus 协议的一种变体，该协议经过优化，可通过 TCP/IP 连接进行通信。该协议设计用于现场级的数据交换（即用于过程映像中的 I/O 数据交换）。在服务端，所有数据包都通过端口号为 502 的 TCP 连接发送。

一般 Modbus TCP 报文如下:

字节	0	1	2	3	4	5	6	7	8 - n
定义	事务标识符		协议标识符 (始终为 00)		字段长度		从机地址	Modbus 功能码	数据

6.1.2 Modbus 数据编码

MODBUS 对地址和数据项使用“大端”表示。这意味着当传输大于单个字节的数字量时，首先发送最高有效字节。

6.1.3 Modbus 数据类型

modbus 协议基于以下基础数据类型：

表 23：Modbus 基础数据类型

数据类型	对象类型	访问类型	描述
数字输入	1 bit	只读	数字输入
线圈	1 bit	读/写	数字输出
输入寄存器	16 bit (word)	只读	模拟输入
保持寄存器	16 bit (word)	读/写	模拟输出

对于每个基础数据类型，都定义了一个或多个功能码。这些功能码允许数字或模拟的输入和输出数据，以及内部变量被设置或直接从现场总线节点中读取。

6.1.2 Modbus 功能码描述

BL200 现场总线节点支持的功能码如下表所示，要执行所需的功能，请指定各自的功能码和所选的输入或输出通道或寄存器的地址。

表 24：Modbus 功能码列表

功能码	功能	访问类型	描述
0x02	读数字输入	只读	按 1 bit 访问
0x01	读线圈	读/写	
0x05	写单个线圈	读/写	
0x0F	写多个线圈	读/写	
0x04	读输入寄存器	只读	按 16 Bit 访问
0x03	读多个寄存器	读/写	
0x06	写单个寄存器	读/写	
0x10	写多个寄存器	读/写	

MODBUS 功能执行如下：

1. MODBUS TCP 主站（例如 PC）使用特定功能码向 BL200 现场总线节点发出请求；

2. BL200 现场总线节点接收数据报文，然后根据主站的请求，以正确的数据响应主站。

如果现场总线节点收到不正确的请求，它会向主站发送错误数据报文（异常）。
异常中包含的异常代码含义如下：

表 25：Modbus 异常代码

异常代码	描述
0x01	非法功能
0x02	非法数据地址
0x03	非法数据值
0x04	从机设备故障

6.1.2.1 功能码 0x02（读数字输入）

此功能码用于读取单个或多个数字输入连续状态。

1. 请求

该请求指定了起始地址和要读取的数量。

表 26：功能码 0x02-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x02	读数字输入，用功能码0x02
起始地址	2 Byte	0x07 D0	地址详见“Modbus 寄存器映射”章节
输入数量	2 Byte	0x08	读 8 个数字输入

2. 响应

数据字段表明了输入状态的值。二进制 1 对应开启状态，0 对应关闭状态。第一个数据字节的最低有效位（LSB）包含请求的第一位，其他的则按升序排列。如果响应数据不是 8 的倍数，则最后一个数据字节的其余位将填充零（朝向字节的高位）。

表 27：功能码 0x02-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 04	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x02	读数字输入，用功能码0x02
数据字节数	1 Byte	0x01	数据的字节数
数据	1 Byte	0x89	响应的数据

3. 异常

表 28：功能码 0x02-响应异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x82	Modbus功能码 + 0x80
异常编码	1 Byte	0x01	0x01 或 0x02

4. 示例

从地址 2000 到 2007 读 8 个数字输入的值。

请求

0x00 01 00 00 00 06 01 02 07 D0 00 08

表 29：功能码 0x02-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	01	07 D0		00 08	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		线圈数量	

响应

0x00 01 00 00 00 04 01 02 01 89

表 30：功能码 0x02-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10
Data	00 01		00 00		00 04		01	01	01	89
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	数据字节数	数据

从 2007 到 2000 的状态显示为字节值 0x89 或二进制 1000 1001。地址 2007 是该字节最高有效位 MSB，2000 是最低有效位 LSB，从高位到低位的分布如下：

表 31：数字输入数据

Bit	7	6	5	4	3	2	1	0
地址	2007	2006	2005	2004	2003	2002	2001	2000

状态	1	0	0	0	1	0	0	1
说明	闭合	断开	断开	断开	闭合	断开	断开	闭合

6.1.2.2 功能码 0x01（读线圈）

此功能码用于读取远程设备中单个或多个线圈的连续状态。

1. 请求

该请求指定了起始地址，即指定第一个线圈的地址，以及线圈的数量。

表 32：功能码 0x01-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x01	读线圈，用功能码0x01
起始地址	2 Byte	0x03 E8	地址详见“Modbus 寄存器映射”章节
线圈数量	2 Byte	0x00 08	读 8 个线圈状态

2. 响应

数据字段表明了输入状态的值。二进制 1 对应开启状态，0 对应关闭状态。第一个数据字节的最低有效位（LSB）包含请求的第一位，其他的则按升序排列。如果响应数据不是 8 的倍数，则最后一个数据字节的其余位将填充零（朝向字节的高位）。

表 33：功能码 0x01-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 04	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x01	读线圈，用功能码0x01
数据字节数	1 Byte	0x01	数据的字节数
数据	1 Byte	0x89	响应的数据

3. 异常

表 34：功能码 0x01-异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x81	Modbus功能码 + 0x80

异常编码	1 Byte	0x01	0x01 或 0x02

4. 示例

从地址 1000 到 1007 读 8 个线圈的状态值。

请求
0x00 01 00 00 00 06 01 01 03 E8 00 08

表 35：功能码 0x01-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	01	03 E8		00 08	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		线圈数量	

响应
0x00 01 00 00 00 04 01 01 01 89

表 36：功能码 0x01-响应报文

Byte	1	2	3	4	5	6	7	8	9	10
Data	00 01		00 00		00 04		01	01	01	89
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	数据字节数	数据

从 1007 到 1000 的状态显示为字节值 0x89 或二进制 1000 1001。地址 1007 是该字节最高有效位 MSB，1000 是最低有效位 LSB，从高位到低位的分布如下：

表 37：线圈数据

Bit	7	6	5	4	3	2	1	0
地址	1007	1006	1005	1004	1003	1002	1001	1000
状态	1	0	0	0	1	0	0	1
说明	闭合	断开	断开	断开	闭合	断开	断开	1000

6.1.2.3 功能码 0x05（写单个线圈）

此功能将向从站设备写单个线圈状态。

1. 请求

表 38：功能码 0x05-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数

设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x05	写单个线圈，用功能码0x05
寄存器地址	2 Byte	0x03 E8	地址详见“Modbus 寄存器映射”章节
写入数据	2 Byte	0xFF 00	此值为：0xFF 00 或者0x00 00。0xFF 00 表示写入1，0x00 00 表示写入0。

2. 响应

表 39：功能码 0x05-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x05	写单个线圈，用功能码0x05
数据字节数	2 Byte	0x03 E8	写入线圈的寄存器地址
写入数据	2 Byte	0xFF 00	此值为：0xFF 00 或者 0x00 00。0xFF 00 表示写入 1，0x00 00 表示写入 0。

3. 异常

表 40：功能码 0x05-异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x85	Modbus功能码 + 0x80
异常编码	1 Byte	0x81	0x01 或 0x02

4. 示例

把地址 1000 的线圈的状态值写为 1，即闭合状态。

请求

0x00 01 00 00 00 06 01 05 03 E8 FF 00

表 41：功能码 0x05-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	05	03 E8		FF 00	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	线圈地址		写“1”	

响应

0x00 01 00 00 00 06 01 05 03 E8 FF 00

表 42：功能码 0x05-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	05	03 E8		FF 00	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	线圈地址		写“1”	

6.1.2.4 功能码 0x0F（写多个线圈）

此功能码用于把连续的多个线圈设置为断开或闭合。请求的开/关状态由请求数据字段的内容指定。逻辑“1”请求闭合相应的输出，逻辑上的“0”请求其断开。正常响应返回功能码、起始地址和执行的线圈数量。

1. 请求

表 43：功能码 0x0f-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 08	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x0F	写多个线圈，用功能码0x0F
起始地址	2 Byte	0x03 E8	地址详见“Modbus 寄存器映射”章节
线圈数量	2 Byte	0x00 08	
数据字节数	1 Byte	0x01	
数据	1 Byte	0xFF	

2. 响应

表 44：功能码 0x0f-响应报文

字段名称	字节（**Byte**）	示例	说明
事务标识符	2 Byte	0x00 00	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x0F	写多个线圈，用功能码0x0F
起始地址	2 Byte	0x03 E8	
线圈数量	2 Byte	0x00 08	

3. 异常

表 45：功能码 0x0f-异常

字段名称	字节（**Byte**）	示例	说明
…			

功能码	1 Byte	0x8F	Modbus功能码 + 0x80
异常编码	1 Byte		0x01 或 0x02

4. 示例

从地址 1000 开始，将 8 个线圈全部闭合，即将 8 个线圈的值写为 0xFF。

请求

0x00 01 00 00 00 08 01 0F 03 E8 00 08 01 FF

表 46：功能码 0x0f-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Data	00 01		00 00		00 08		01	0F	03 E8		00 08		01	FF
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		线圈数量		字节数	数据

响应

0x00 01 00 00 00 06 01 0F 03 E8 00 08

表 47：功能码 0x0f-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	0F	03 E8		00 08	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		线圈数量	

6.1.2.5 功能码 0x04（读输入寄存器）

此功能码用于读取多个远程设备中的连续输入寄存器。请求 PDU 指定起始寄存器的地址和寄存器的数量。响应消息中的寄存器数据被打包为每个寄存器两个字节，每个字节内的二进制内容靠右对齐。

1. 请求

表 48：功能码 0x04-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x04	读输入寄存器，用功能码0x04
起始地址	2 Byte	0x0F A0	地址详见“Modbus 寄存器映射”章节
寄存器数量	2 Byte	0x00 08	

2. 响应

表 49：功能码 0x04-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 00	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 13	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x04	读输入寄存器，用功能码0x04
字节数	1 Byte	0x10	
数据	16 Byte	0x 3F 8E 38 86 40 0E 38 86 40 55 54 CA 40 8E 35 3F	

5. 异常

表 50：功能码 0x04-异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x84	Modbus功能码 + 0x80
异常编码	1 Byte	0x01	0x01 或 0x02

6. 示例

从地址 4000 开始，读 4 个模拟输入的值。由于 BL200 耦合器节点寄存器映射数据类型是 32Bit Float，即 1 个模拟输入数据 = 2 个寄存器 = 4 个字节 ，因此需要读 8 个输入寄存器。

请求

0x00 01 00 00 00 06 01 04 0F A0 00 08

表 51：功能码 0x04-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	04	0F A0		00 08	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		寄存器数量	

响应

0x00 01 00 00 00 13 01 04 10 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85

表 52：功能码 0x04-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10…25
Data	00 01		00 00		00 13		01	04	10	xxx

说明	事务标识符	协议标识符	报文长度	设备地址	功能码	字节数	数据

其中数据部分共 16 个字节，转换成十进制如下：

表 53：读取输入寄存器 - 转换数据十进制

Byte	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Data	3F 9D 70 A4				40 15 C2 8F				40 5C CC CD				40 91 EB 85			
十进制	1.23				2.34				3.45				4.56			
说明	第一个数据				第二个数据				第三个数据				第四个数据			

6.1.2.6 功能码 0x03（读保持寄存器）

此功能码用于读取多个远程设备中的连续保持寄存器。请求 PDU 指定起始寄存器的地址和寄存器的数量。响应消息中的寄存器数据被打包为每个寄存器两个字节，每个字节内的二进制内容靠右对齐。

1. 请求

表 54：功能码 0x03-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x03	读保持寄存器，用功能码0x03
起始地址	2 Byte	0x0B B8	地址详见“Modbus 寄存器映射”章节
寄存器数量	2 Byte	0x00 08	需要读取的保持寄存器数量

2. 响应

表 55：功能码 0x03-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 00	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 13	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x03	读保持寄存器，用功能码0x03
字节数	1 Byte	0x10	数据字节数
数据	16 Byte	0x 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD	响应的数据

		40 91 EB 85	

3. 异常

表 56：功能码 0x03-异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x83	Modbus功能码 + 0x80
异常编码	1 Byte	0x01	0x01 或 0x02

4. 示例

从地址 3000 开始，读 4 个模拟输出（属于保持寄存器）的值。由于模拟输出 I/O 模块寄存器映射数据类型是 32Bit Float，即 1 个模拟输出数据 = 2 个寄存器 = 4 个字节 ，所以需要读取 8 个保持寄存器。

请求

0x00 01 00 00 00 06 01 03 0B B8 00 08

表 57：功能码 0x03-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	03	0B B8		00 08	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		寄存器数量	

响应

0x00 01 00 00 00 13 01 03 10 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85

表 58：功能码 0x03-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10…25
Data	00 01		00 00		00 13		01	03	10	xxx
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	字节数	数据

其中数据部分共 16 个字节，转换成十进制如下：

表 59：读取保持寄存器 - 转换数据十进制

Byte	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Data	3F 9D 70 A4				40 15 C2 8F				40 5C CC CD				40 91 EB 85			
十进制	1.23				2.34				3.45				4.56			
说明	第一个数据				第二个数据				第三个数据				第四个数据			

6.1.2.7 功能码 0x06（写单个寄存器）

此功能码用于写入单个远程设备中的保持寄存器。请求 PDU 指定起始寄存器的地址和

寄存器的数量。响应消息中的寄存器数据被打包为每个寄存器两个字节，每个字节内的二进制内容靠右对齐。

该功能码只适用读取串口 I/O 模块寄存器映射数据，地址范围： 30000 ... 39999 。模拟输入输出 I/O 模块的数据类型是 32Bit Float 格式，无法读取完整数据，不可以使用该功能。

1. 请求

表 60：功能码 0x06-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x06	写单个保持寄存器，用功能码0x06
寄存器地址	2 Byte	0x75 30	地址详见“Modbus 寄存器映射”章节
数据	2 Byte	0x04 D2	

2. 响应

表 61：功能码 0x06-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 00	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 06	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x06	写单个保持寄存器，用功能码0x06
寄存器地址	2 Byte	0x75 30	
数据	2 Byte	0x04 D2	

3. 异常

表 62：功能码 0x06-异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x86	Modbus功能码 + 0x80
异常编码	1 Byte	0x01	0x01 或 0x02

4. 示例

将寄存器地址 30000 的值写入 1234 (0x04 D2).

请求

0x00 01 00 00 00 06 01 06 75 30 04 D2

表 63：功能码 0x06-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	06	75 30		04 D2	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	寄存器地址		数据	

响应

0x00 01 00 00 00 06 01 06 75 30 04 D2

表 64：功能码 0x06-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	0F	75 30		04 D2	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	寄存器地址		数据	

6.1.2.8 功能码 0x10（写多个寄存器）

此功能码用于写入多个远程设备中的连续保持寄存器。请求 PDU 指定起始寄存器的地址和寄存器的数量。响应消息中的寄存器数据被打包为每个寄存器两个字节，每个字节内的二进制内容靠右对齐。

1. 请求

表 65：功能码 0x10-请求报文

字段名称	字节数	示例	说明
事务标识符	2 Byte	0x00 01	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ： Modbus协议
报文长度	2 Byte	0x00 17	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x10	写多个保持寄存器，用功能码0x10
起始地址	2 Byte	0x0B B8	地址详见“Modbus 寄存器映射”章节
寄存器数量	2 Byte	0x00 08	
数据字节数	1 Byte	0x10	
数据	16 Byte	0x 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85	

2. 响应

表 66：功能码 0x10-响应报文

字段名称	字节（Byte）	示例	说明
事务标识符	2 Byte	0x00 00	Modbus请求/响应事务处理的识别
协议标识符	2 Byte	0x00 00	0x00 00 ：Modbus协议
报文长度	2 Byte	0x00 13	后面数据的字节数
设备地址	1 Byte	0x01	从站的地址识别
功能码	1 Byte	0x10	写多个保持寄存器，用功能码0x10
起始地址	2 Byte	0x0B B8	
寄存器数量	2 Byte	0x00 08	

3. 异常

表 67：功能码 0x10-异常

字段名称	字节（Byte）	示例	说明
…			
功能码	1 Byte	0x90	Modbus功能码 + 0x80
异常编码	1 Byte	0x01	0x01 或 0x02

4. 示例

从地址 3000 开始，写 4 个模拟输出的值。由于 BL200 耦合器节点寄存器映射数据类型是 32Bit Float，即 1 个模拟输出数据 = 2 个保持寄存器 = 4 个字节 ，所以需要写入 8 个保持寄存器。

请求

0x00 01 00 00 00 17 01 10 0B B8 00 08 10 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85

表 68：功能码 0x10-请求报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14…23
Data	00 01		00 00		00 17		01	10	0B B8		00 08		10	xxx
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		寄存器数量		字节数	数据

其中数据部分共 16 个字节，转换成十进制如下：

表 69：写入保持寄存器 - 转换数据十进制

Byte	14															
Data	3F 9D 70 A4				40 15 C2 8F				40 5C CC CD				40 91 EB 85			
十进制	1.23				2.34				3.45				4.56			
说明	第一个数据				第二个数据				第三个数据				第四个数据			

响应

0x00 01 00 00 00 06 01 10 0B B8 00 08

表 70：功能码 0x10-响应报文-示例

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Data	00 01		00 00		00 06		01	10	0B B8		00 08	
说明	事务标识符		协议标识符		报文长度		设备地址	功能码	起始地址		寄存器数量	

6.1.3 Modbus 寄存器映射

BL200 现场耦合器节点内部寄存器映射由 2 部分组成，一部分是数字输入输出和模拟输入输出模块的数据映射组成，地址范围是 1000 ... 9999；另外一部分是串口模块，地址范围是 10000 ... 49999

通过寄存器映射（地址 1000 ... 9999），可以确定或更改数字和模拟 I/O 模块的状态。

表 71：Modbus 寄存器映射-I/O 模块

Modbus 地址		数据类型	访问	功能码	描述
10 进制	**16 进制**				
1000…1999	0x03 E8…0x07 CF	1 Bit	读/写	0x01/05/0F	数字输入 DI
2000…2999	0x07 D0…0x0B B7	1 Bit	只读	0x02	数字输出 DO
3000…3999	0x0B B8…0x0F 9F	32 Bit Float	读/写	0x03/10	模拟输出 AO
4000…4999	0x0F A0…0X13 87	32 Bit Float	只读	0x04	模拟输入 AI
5000…8999	0x13 88…0x	32 Bit Unint	读/写	0x03/04/10	DI 计数值
9000…9999	0x23 28…0x	1 Bit	只读	0x02	模块上电状态

而通过地址 10000 ... 49999 可以确定或更改从串口 I/O 模块映射的数据状态。

表 72：Modbus 寄存器映射-串口模块

Modbus 地址		数据类型	访问	功能码	描述
10 进制	**16 进制**				
10000…19999	0x27 10…0x4E 1F	1 Bit	读/写	0x01/05/0F	数字输入 DI
20000…29999	0x4E 20…0x75 2F	1 Bit	只读	0x02	数字输出 DO
30000…39999	0x75 30…0x9C 3F	16 Bit	读/写	0x03/06/10	模拟输出 AO
40000…49999	0x9C 40…0XC3 4F	16 Bit	只读	0x04	模拟输入 AI

6.2OPC UA

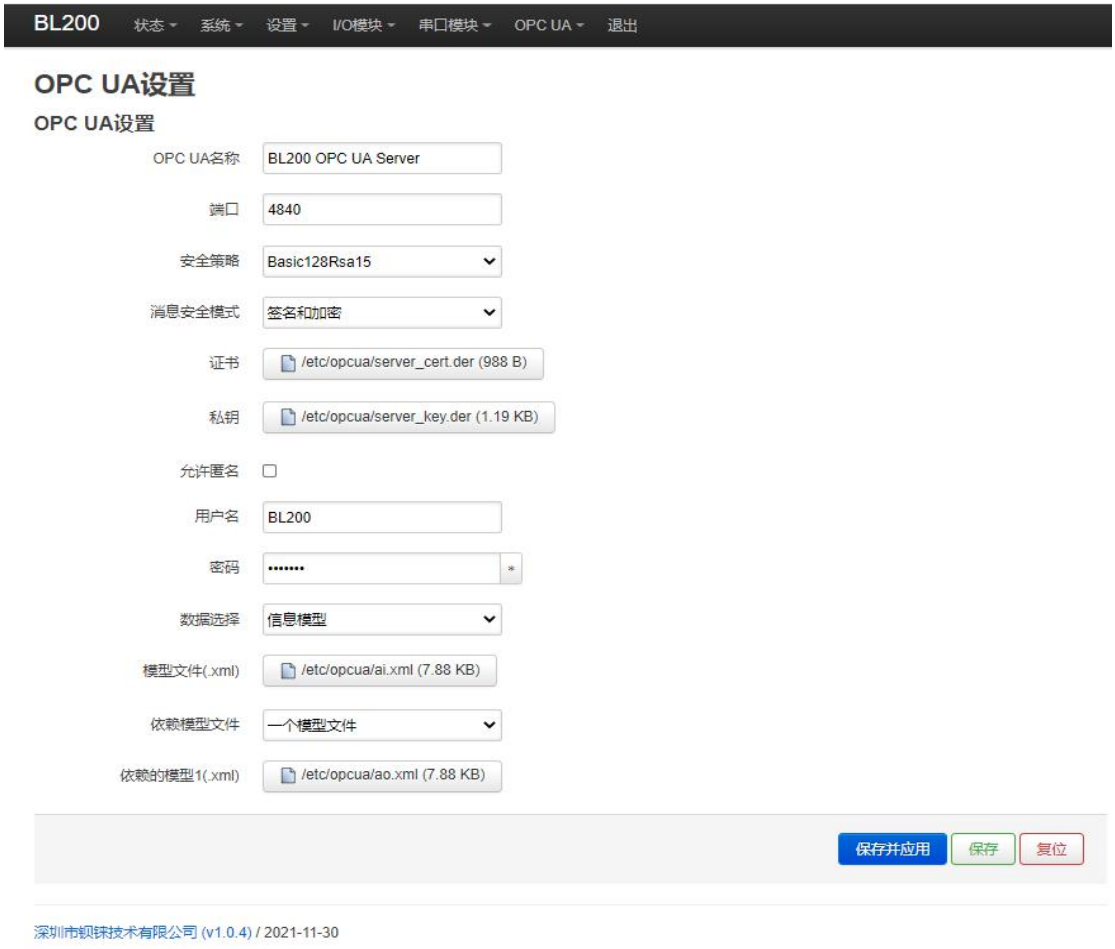
6.2.1 概述

BL200 系列分布式 I/O 系统支持 OPC UA Server 功能，以服务器形式对外提供数据。符合 IEC 62541 工业自动化统一架构通讯标准，数据可以选择加密（X.509 证书）、身份验证方式传送。安全策略支持 basic128rsa15、basic256、basic256sha256、aes128sha256rsaoaep，可选签名或签名和加密方式。支持自定义的信息模型功能，最多可以填写 5 个参考模型。

6.2.2 应用示例

以采集 DI、DO、AI 模块，安全策略选择 basic128rsa15,选择签名和加密方式，数据格式按自定义信息模型方式，参考一个信息模型为例。数据也可以按本公司的格式直接上传，每项配置的定义请参考 5.7 OPC UA 章节介绍。

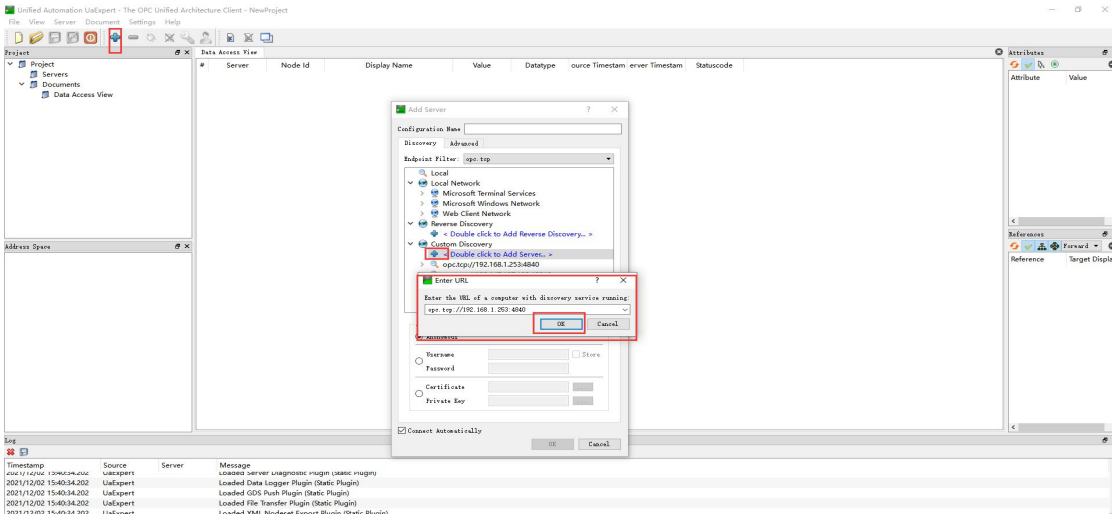
6.2.2.1 OPC UA 网页配置



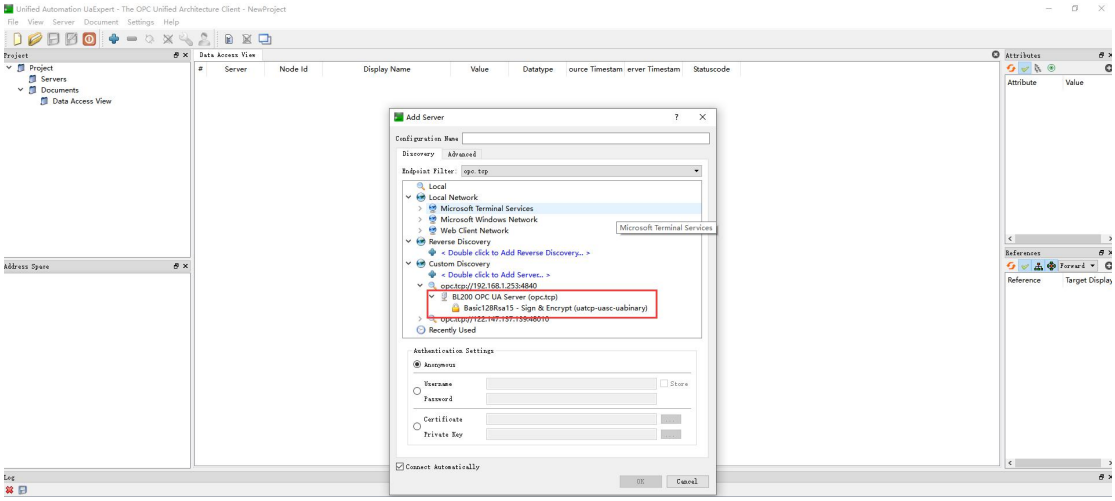
步骤：（1）填写 OPC UA 名称，可以自定义，方便 OPC UA 客户端搜索区分不同的 OPC UA 服务器。如：填写“BL200 OPC UA Server”。（2）OPC UA 服务端的端口号，默认：4840。（3）安全策略选择。如选择 basic128rsa15。（4）消息安全模式选择。如选择签名和加密。（5）上传证书和密钥，点击“选择文件”>点击“上传文件”>选择你的证书或密钥文件，点击打开>在文件名方框内显示后，点击上传文件>上传文件成功后会在方框显示你上传的文件，点击你上传的证书或密钥文件>这时就会在证书或密钥项显示你的证书或密钥文件。（6）是否允许匿名，因使用签名和加密的方式，则允许匿名不打钩。（7）填写用户名和密码，客户端连接时需要填写这个用户名和密码。（8）选择数据，因用自定义的信息模型，所以要选择“信息模型”。（9）信息模型文件上传，上传方法和上传证书或密钥文件一样，上传的是 xml 文件。（10）依赖模型文件，是否有参考模型，有参考多少个。（11）依赖的模型：上传你参考的模型，上传方法和上传证书或密钥文件一样，上传的是 xml 文件。（12）点击“保存并应用”。

6.2.2.2 用 UaExpert 客户端连接收发数据

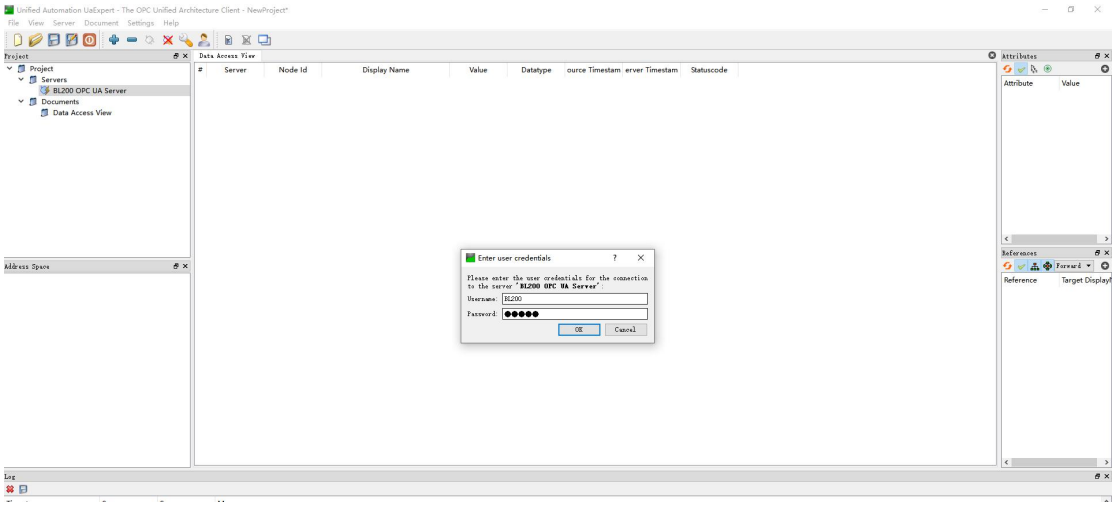
打开 UaExpert（OPC UA 客户端），输入 OPC UA 服务器 IP 和端口。



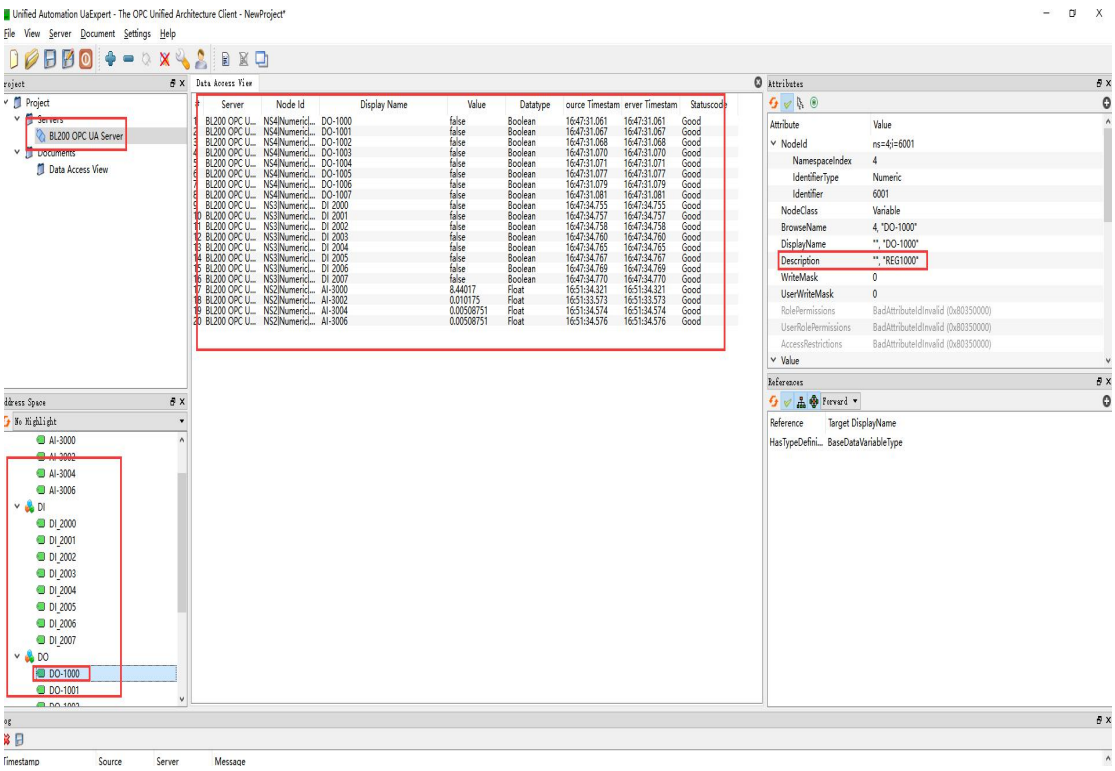
点击搜索，点击搜索到的 OPC UA 服务器，点击签名和加密方式的 basic128rsa15。



输入设置的用户名和密码



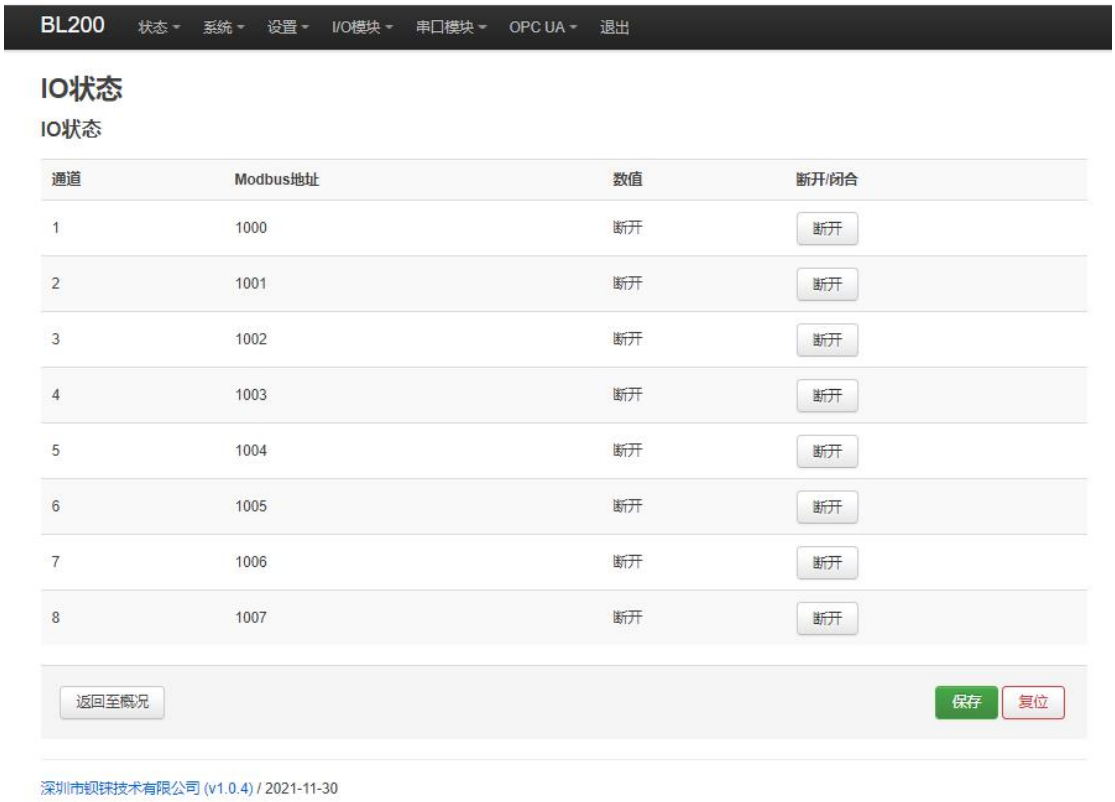
采集到的数据如下



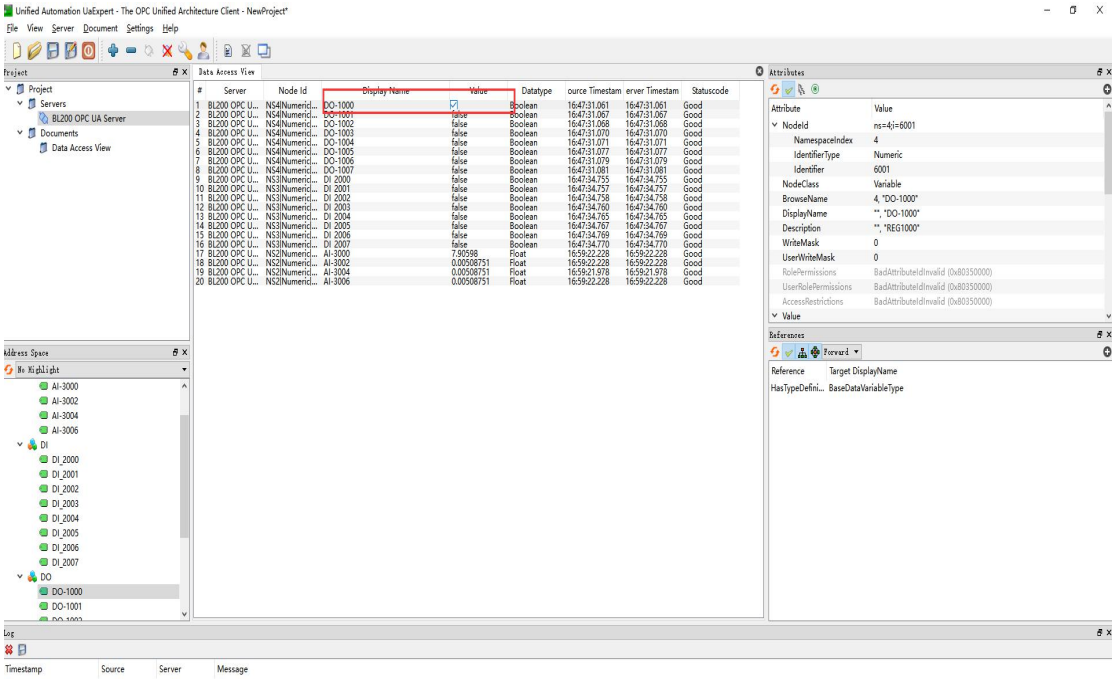
自定义的信息模型数据点的描述项一定要是 REG+Modbus 地址,如上图的 DO-100 点的描述。

OPC UA 客服端数据下发

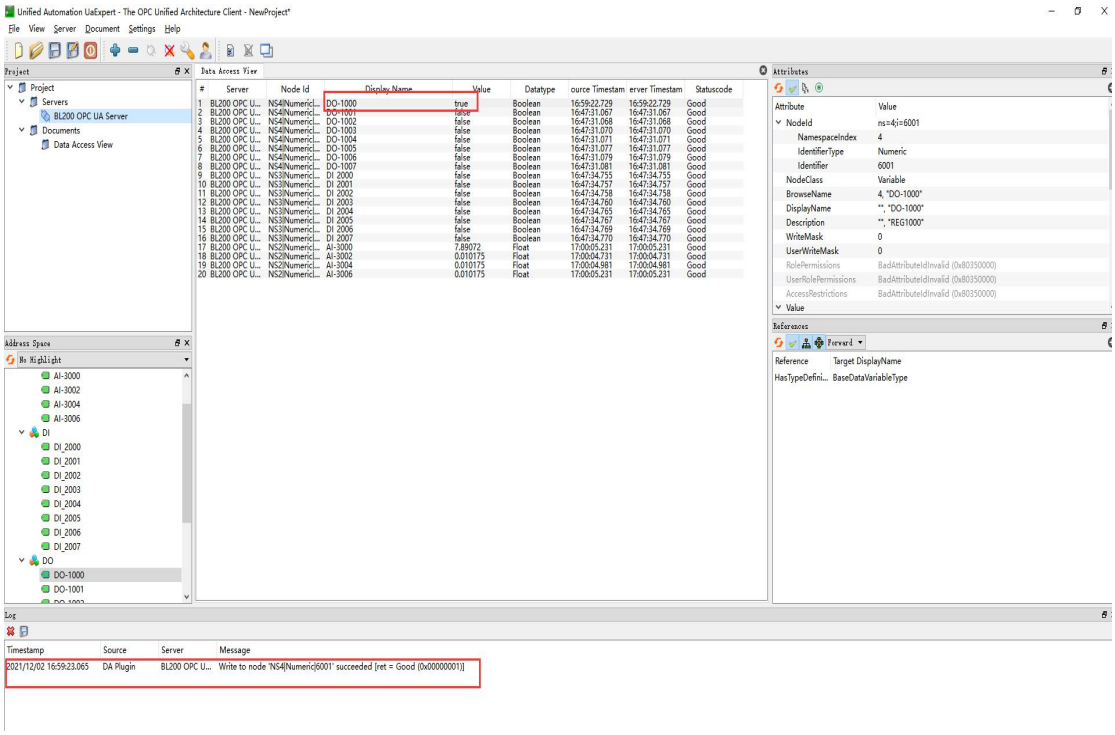
以下发 DO-1000 这个数据点为例



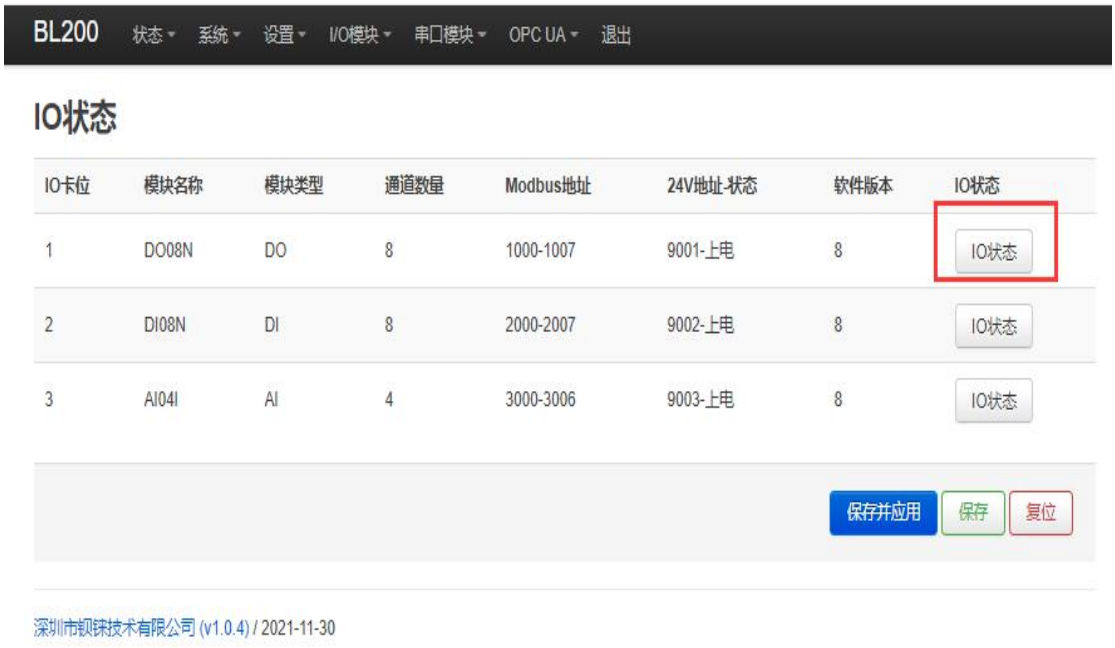
点击 DO-1000 数据点的值，原来是 false，方格内没有 √，点击一下打上 √，在空白的地方点击鼠标左键或按键盘上的【Enter】键。

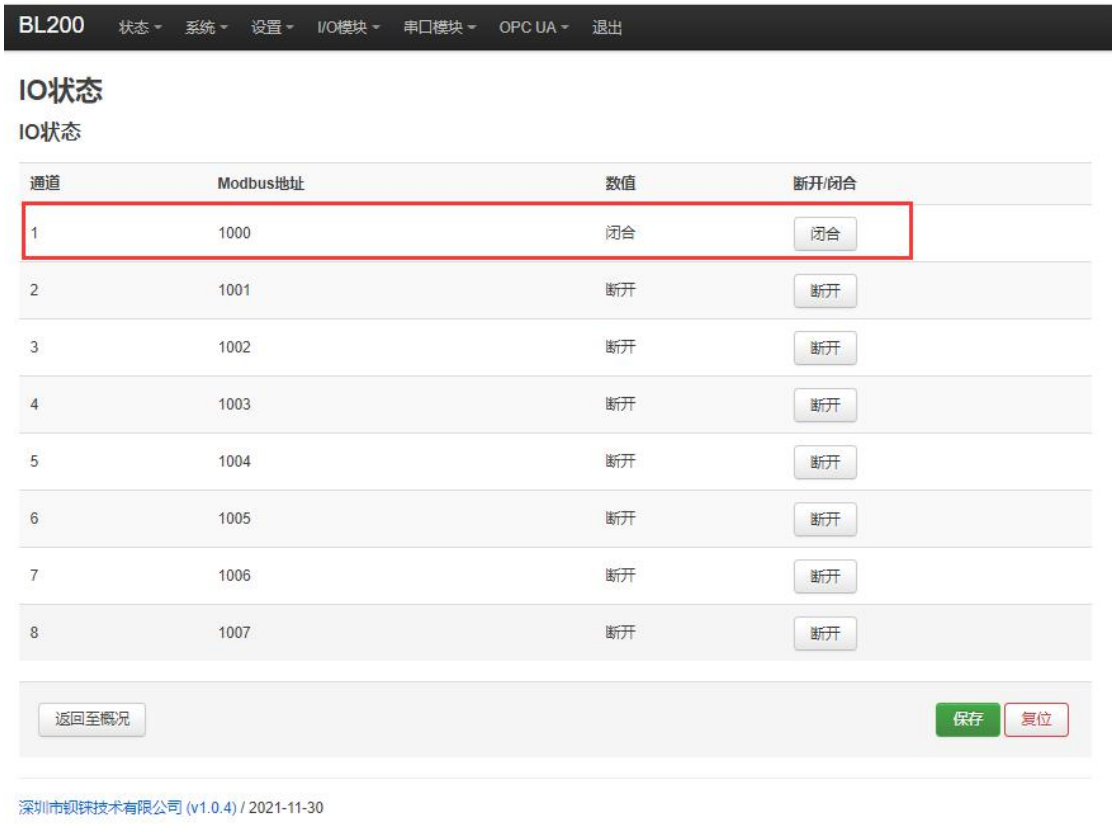


OPC UA 客户端下发成功会有报文，因服务器响应也很快，可以在上面看到值已经变为“true”。



在 BL200 的网页配置查看 DO 的状态。DO1 也是由原来的断开变成闭口。





7. 附录

7.1插图列表

7.2表格列表