



Edge I/O System

BL200 Series product

Modbus / MQTT / OPC UA / Profinet / EtherCAT/ BACnet



BL200 User Manual

Version: V1.0

Date: 2022-5-8

Shenzhen BeiLai Technology
Co., Ltd.

Website: www.bliiot.com

Foreword

Thank you for using the Edge I/O system of Shenzhen BeiLai Technology Co., Ltd. Reading this product manual will allow you to quickly understand the function and usage of this product.

Copyright

this manual is owned by Shenzhen BeiLai Technology Co., Ltd. Without the written permission of the company, any company or person individuals have no right to copy, disseminate and reproduce any part of this manual in any form, otherwise all consequences shall be borne by the violators.

Disclaimer

This product is mainly used for industrial field automation control, and the operator must be an experienced electrical expert in the field of automation or electrical. Please use it in accordance with the parameters and technical specifications provided in the manual. The company will not be responsible for property or personal injury caused by abnormal use or improper use of this product.

Revision History

| Revision Date | Version | Description | Owner |
|---------------|---------|-----------------|-------|
| Oct 12, 2021 | V1.0 | Initial Release | HYQ |
| | | | |

Content

| | |
|---|----|
| 1. Product Introduction | 5 |
| 1.1 Overview | 5 |
| 1.2 Typical application | 6 |
| 1.3 Features Highlight | 7 |
| 1.4 Technical parameter | 8 |
| 1.5 Product Models Selection Table | 9 |
| 2. Device description | 10 |
| 2.1 View | 10 |
| 2.2 Dimensions | 11 |
| 2.3 Data Contacts/Local Bus | 12 |
| 2.4 Power Jumper Contacts | 12 |
| 2.5 Terminal Point | 13 |
| 2.6 LED Indicators | 14 |
| 2.7 Ethernet interface | 15 |
| 2.8 IP address selector switch | 15 |
| 2.9 Factory reset button | 15 |
| 2.10 Electrical schematic | 16 |
| 3. Product Installation | 16 |
| 3.1 Installation sequence | 16 |
| 3.2 Install the controller | 17 |
| 3.3 Remove the controller | 17 |
| 3.4 Insert the I/O module | 18 |
| 3.5 Remove I/O module | 19 |
| 4. Device Connection | 20 |
| 4.1 Wiring | 20 |
| 4.2 Supply Power | 21 |
| 4.2.1 Power supply to Edge controller | 21 |
| 4.2.2 Power supply to I/O modules | 21 |
| 4.2.3 Earthing | 22 |
| 4.3 Connect BL200 devices each other and to network | 23 |
| 5. Web configuration | 25 |
| 5.1 Preparation before configuration | 25 |
| 5.1.1 Connect computer and controller | 25 |
| 5.1.2 Configure the computer IP address | 25 |
| 5.1.3 Configure the controller IP address | 28 |
| 5.1.4 Factory default settings | 30 |
| 5.1.5 Login configuration page | 30 |
| 5.2 Status | 32 |
| 5.3 System | 34 |
| 5.3.1 System | 34 |
| 5.3.2 Administration | 37 |

| | |
|--|----|
| 5.3.3 Backup/Flash Firmware | 38 |
| 5.3.4 Reboot | 39 |
| 5.4 Settings | 39 |
| 5.5 I/O modules | 40 |
| 5.5.1 Digital input module | 41 |
| 5.5.2 Digital output module | 42 |
| 5.5.3 Analog input module | 43 |
| 5.5.4 Analog output module | 44 |
| 5.6 Serial port RS485 module | 45 |
| 5.6.1 Serial port settings | 45 |
| 5.6.2 Modbus settings | 45 |
| 5.7 OPC UA settings | 48 |
| 5.8 Other functions | 49 |
| 6. Fieldbus Communication | 49 |
| 6.1 Modbus | 49 |
| 6.1.1 Overview | 49 |
| 6.1.2 Modbus function code description | 51 |
| 6.1.3 Modbus register mapping address | 68 |
| 6.2 OPC UA | 69 |
| 6.2.1 Overview | 69 |
| 6.2.2 Application example | 69 |
| 7. Appendix | 75 |
| 7.1 List of Figures | 75 |
| 7.2 List of Tables | 76 |

1. Product Introduction

1.1 Overview

BL200 series product is **Edge I/O System for data acquisition and industrial control**. The system consists of two parts : **Edge I/O controller and distributed I/O modules group**. Various types I/O modules are **optional**, for example digital, analog and others particular use. The Edge I/O controller supports **various Fieldbus protocols** such as Modbus, MQTT, OPC UA, Profinet, EtherCAT, BACnet etc.

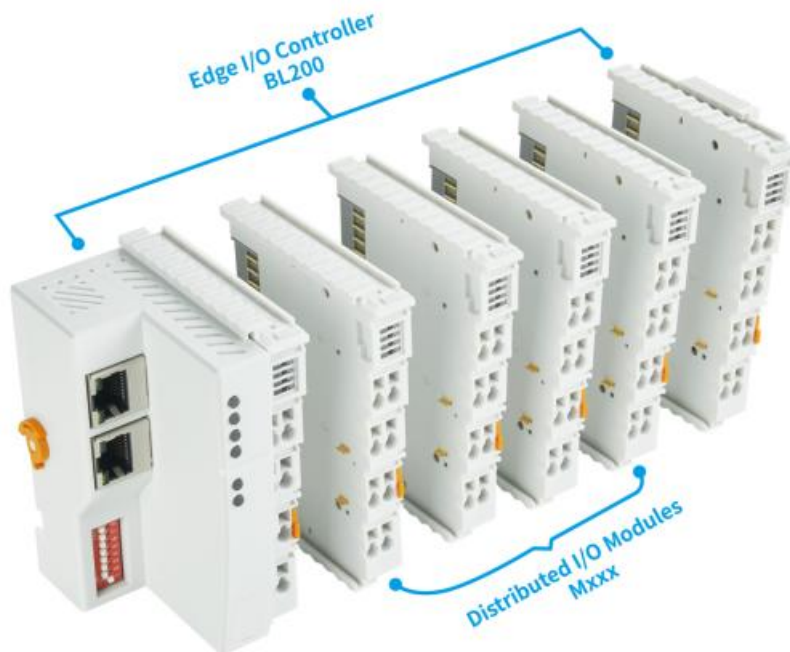


Figure 1: Edge I/O system

The Edge I/O system is design based on powerful microprocessor 32-bit ARM926EJ-S™ and uses Linux operating system. **It can quickly access to industrial field device and software (such as PLC, MES, SCADA and ERP systems), but also able connect to various cloud platforms (AWS IoT Cloud, Thingsboard, Ignition, Huawei Cloud, and Alibaba Cloud).**

It supports programmable logic control, edge computing, customization applications, very suitable for IIoT and industrial automation.

In addition, the device also provides SDK, allowing users to carry out secondary development to meet the needs of the fragmented applications in niche market.



Figure 1: Fieldbus Node --- Edge I/O system

In the industrial field communication, the edge I/O system is considered as a Fieldbus node. The communication between the node and field devices (eg PLC) is through the Ethernet port of Edge I/O controller. The communication between the edge controller and the I/O modules is via the board bus locally inside device.

The two RJ45 Ethernet ports of the BL200 have integrate the switch function inside, which can establish a linear topology without the need for additional switches or hubs.

Two sets of independent power supplies are used to supply 24V DC to the edge controller and I/O modules group respectively, therefore they are electrically isolated from each other.

When people assemble I/O modules, each I/O module can be arranged in any sequence combination. it is not required to be grouped by module type. However, a terminal module (identified as TERM) must be inserted at the end so as to ensure data transfer properly.

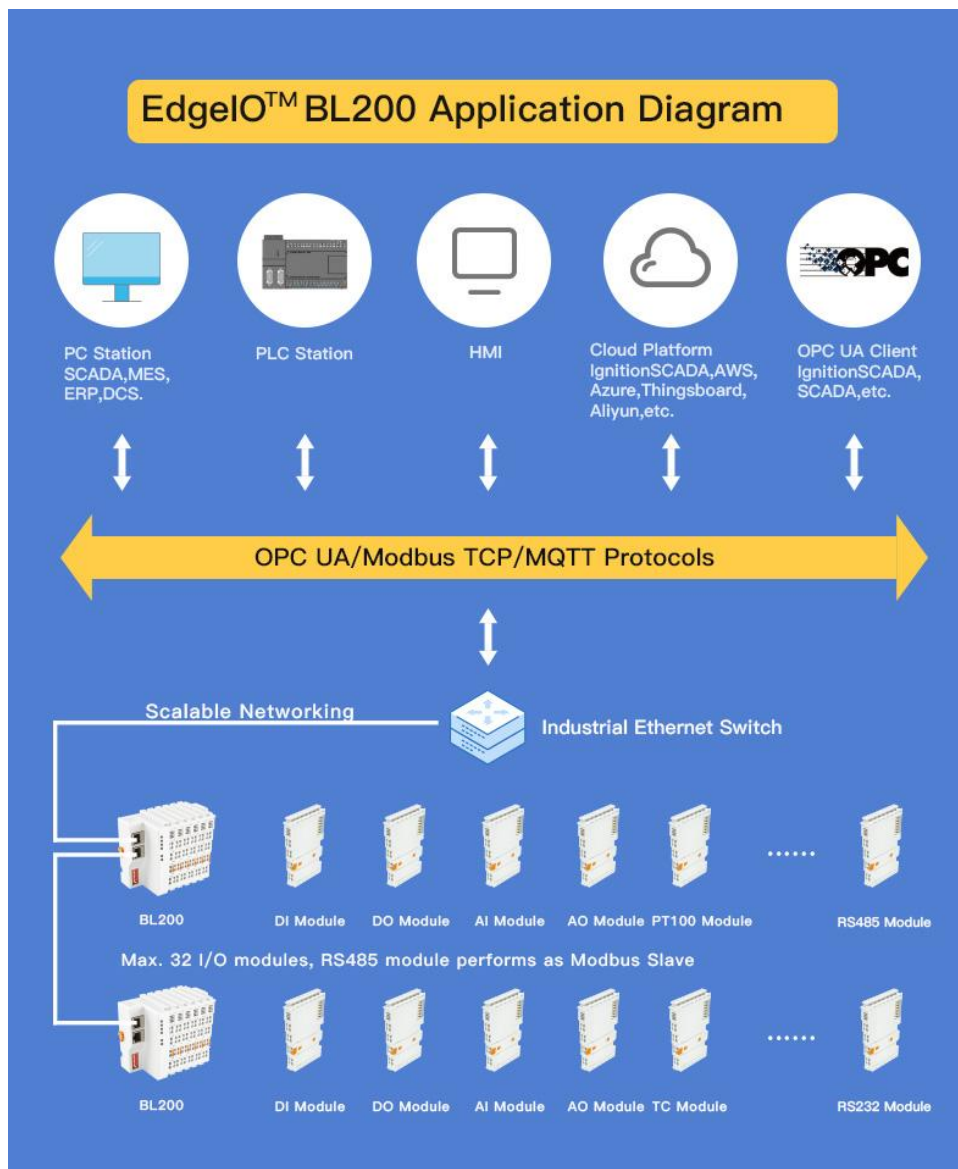
People can configure BL200 parameters through web browser for testing the device.

1.2 Typical application

BL200 is **especially suitable for distributed application scenarios**, because BL200 can be cascaded with each other through Ethernet cables. various I/O modules can be distributed on different floors of different buildings in different locations according to user needs.

The whole system has the characteristics of easy expansion, easy configuration, convenient network wiring and high reliability, can be widely used in

- industrial automation/industry 4.0/smart factories,**
- smart building HVAC/smart parking lots/smart communities,**
- sewage treatment plants,**
- new energy power generation Factory (solar photovoltaic, wind power generation)**



1.3 Features Highlight

- Up to 32pcs I/O modules can be supported, one by one to extend I/O module
- Support various Fieldbus protocols such as Modbus, MQTT, OPC UA, Profinet, EtherCAT, BACnet etc
- Build-in cloud driver, compatible with various cloud platforms (AWS IoT Core, Thingsboard, Ignition, Huawei Cloud, and Alibaba Cloud)
- Programmable logic control, Edge computing, SDK available too
- 2 x RJ45 allows cascading and bypass, save wire cost and wiring cost
- Modular, screw-free installation, easy wiring
- Cyber security + Data encryption protection
- Industrial field side, edge controller side and I/O modules side are electrically isolated from each other.

1.4 Technical parameter

Table 1: technical parameter

| name | parameter | describe |
|--------------------------------|---|---|
| Power supply to I/O controller | Input voltage(system): | 24 VDC |
| | Input current(system): | Max. 500 mA@24VDC |
| | Power Efficiency | 84% |
| | Internal bus voltage | 5VDC |
| | controller consumption current | Max. 300mA@5VDC |
| | I/O consumption current | Max. 1700mA@5VDC |
| | Isolation protection | 500 V system/supply |
| Power supply to I/O modules | Input voltage (field) | 24 VDC |
| | Power supply current across contacts (Max.) | 10 ADC |
| Ethernet | Number | 2 X RJ45 |
| | Transmission medium | Twisted Pair STP 100 Ω Cat 5 |
| | Max. cable length | 100m |
| | Baud rate | 10/100 Mbit/s |
| | Isolation protection | ESD contact: 8KV , surge: 4KV (10/1000us) |
| I/O controller System | Operating system | Linux |
| | Processor | ARM926EJ-S |
| | CPU | 300MHz |
| | RAM | 64MB |
| | Flash | 128MB |
| | Number of I/O modules | Max. 32 |
| | Process mapping data points serial module(Modbus) | <ul style="list-style-type: none"> ● Bool : 4096 ● 16 Bit : 2048 ● 32 Bit : 1024 |
| | Protocols | Modbus TCP , MQTT , OPC UA , HTTP , BootP , DHCP , DNS , SNTP , SNMP |
| | Max. number of socket links | 15 Modbus TCP |
| Field Wiring | Connection technology | CAGE CLAMP® |
| | wire cross-section | 0.08 mm ² ... 2.5 mm ² , AWG 28 ... 14 |
| | Strip length | 8 mm ... 9 mm / 0.33 in |
| Working Environment | operation | 0 ... 55 °C |
| | storage | -40 ... 70 °C |
| | Relative humidity | Max. 5%...95% without condensation |
| | Operating altitude | 0 ... 2000 m |
| | Protection type | IP20 |

| | | |
|---------------|----------------|---|
| Product Size | Width | 48mm |
| | Length | 100mm |
| | High | 69mm |
| Material | color | Light grey |
| | Shell material | Polycarbonate, Nylon 6.6 |
| | Fire load | 1.239 MJ |
| | Weight | 180 g |
| Mechanical | Mounting type | DIN-35 rail |
| Certification | 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 Product Models Selection Table

Table 2: model selection

| No. | Name | Model | Channel | Signal type |
|-----|------------------------------|----------|---------|-------------|
| 1 | Digital Input Module | M1082 | 8 | NPN |
| 2 | Digital Input Module | M1081 | 8 | PNP |
| 3 | Digital Output Module | M2082 | 8 | NPN |
| 4 | Digital Output Module | M2081 | 8 | PNP |
| 5 | Analog Input Module | M3041 | 4 | Current |
| 6 | Analog Output Module | M4043 | 4 | Voltage |
| 7 | RTD Input Module | M5041 | 4 | Resistor |
| 8 | Serial port Interface Module | M6021 | 2 | RS485 |
| 9 | 24V Power Supply Module | M701 | / | / |
| 10 | Terminal Module | M801 | / | / |
| 11 | Modbus-TCP I/O controller | BL200 | / | / |
| 12 | OPC UA I/O controller | BL200UA | / | / |
| 13 | MQTT I/O controller | BL200M | / | / |
| 14 | Multiprotocol I/O controller | BL200Pro | / | / |
| 15 | Profinet I/O controller | BL200PN | / | / |
| 16 | Ethernet/IP I/O controller | BL200EP | / | / |
| 17 | EtherCAT I/O controller | BL200EC | / | / |
| 18 | BACnet/IP I/O controller | BL200BN | / | / |

2. Device description

2.1 View

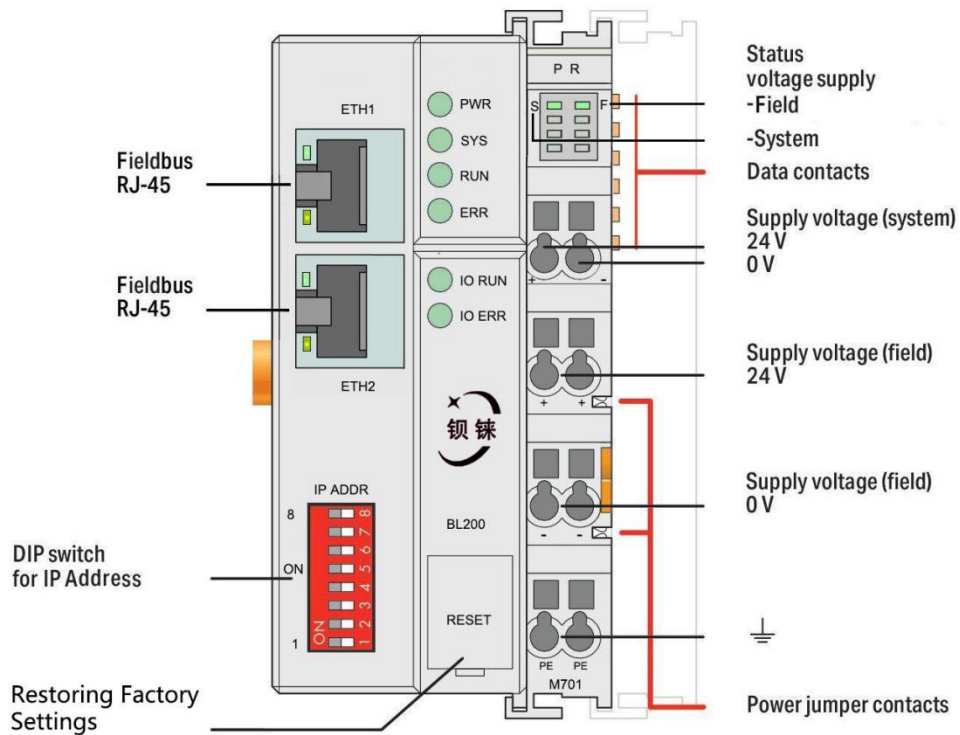


Figure 2: view

2.2 Dimensions

Unit: mm

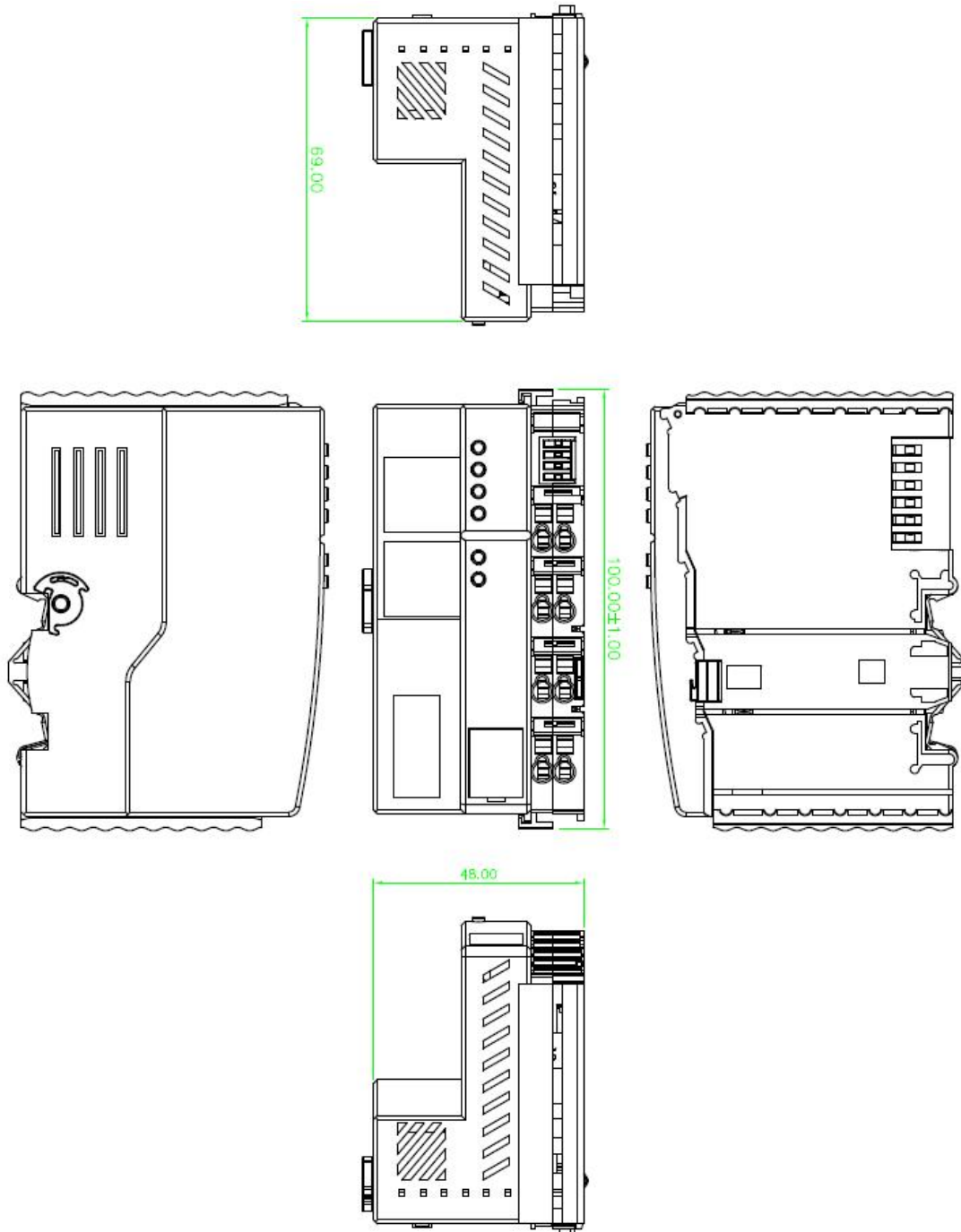


Figure 3: 2D schematic

2.3 Data Contacts/Local Bus

Communication between the fieldbus controller/controller and the I/O modules as well as the system supply of the I/O modules is carried out via the local bus. The contacting for the local bus consists of 6 data contacts, which are available as self-cleaning gold spring contacts.

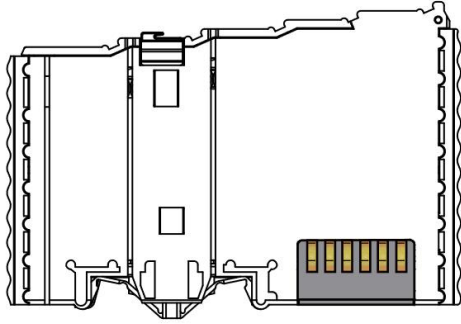


Figure 4: data contacts

2.4 Power Jumper Contacts

The power module that comes with the controller has two self-cleaning power jumper contacts to power the field side. The maximum current of this power supply across the contacts is 10A, exceeding the maximum current will damage the contacts. When configuring the system, make sure that the above current maximum value is not exceeded. If it exceeds, a power supply module must be inserted.

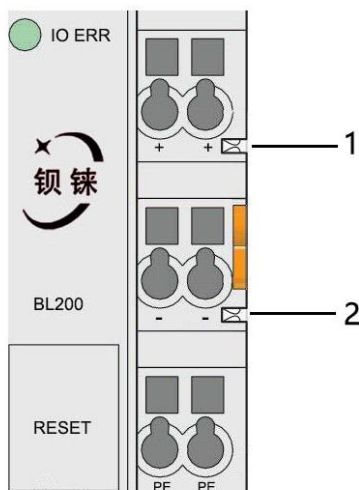


Figure 5: power jumper contacts

Tabel 3: "power jumper contacts" describe

| No. | Type | Describe |
|-----|----------------|------------------------------|
| 1 | Spring contact | Supply 24V to the field side |
| 2 | Spring contact | Supply 0V to the field side |

2.5 Terminal Point

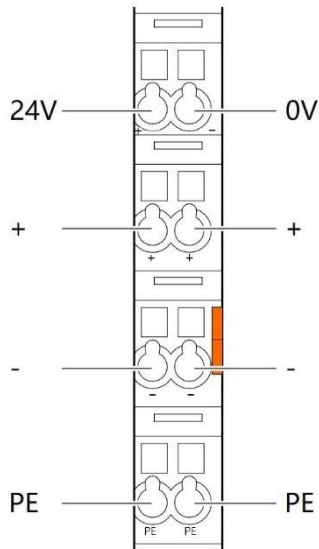


Figure 6: terminal point

Tabel 4: "terminal point" describe

| Name | Describe |
|------|---------------------------------|
| 24V | System Power 24VDC |
| 0V | System Power 0VDC |
| + | Connections Field Supply 24 VDC |
| + | Connections Field Supply 24 VDC |
| - | Connections Field Supply 0 VDC |
| - | Connections Field Supply 0VDC |
| PE | grounding |
| PE | grounding |

2.6 LED Indicators

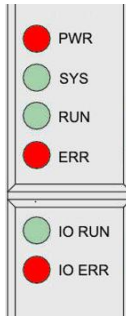


Figure 7: controller LED Indicators

Tabel 5: “controller LED Indicators” describe

| LED | Describe | Color | Status | Meaning |
|---------|-----------------------|-------|----------|--------------------------------------|
| PWR | Power indicator | red | Stay on | Normal power supply |
| | | | Off | No power |
| SYS | System indicator | green | Stay on | System is abnormal |
| | | | Off | System is running normally |
| RUN | Running indicator | green | Flashing | System is running normally |
| | | | Off | System is abnormal |
| ERR | Error indicator | red | Stay on | Northbound protocol connection error |
| | | | Off | No errors |
| I/O RUN | I/O Running indicator | green | Flashing | IO module is operating normally |
| | | | Off | No node inserted |
| I/O ERR | I/O Error indicator | red | Stay on | IO module communication error |
| | | | Off | No errors |



Figure 8: Power Module LED Indicators

Tabel 6: “Power Module LED Indicators” describe

| LED | Describe | Color | Status | Meaning |
|-----|----------------------------|-------|---------|-------------|
| S | System 24V power indicator | green | Stay on | Power is OK |
| | | | Off | No power |
| F | Field 24V power indicator | green | Stay on | Power is OK |
| | | | Off | No power |

2.7 Ethernet interface

Description: It is connected to the Ethernet-based fieldbus through the ETH2 interface, and the EHT1 is used to connect other nodes that need to be connected to the Ethernet.

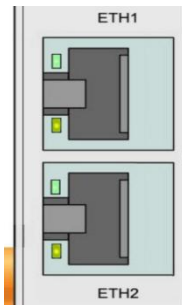


Figure 9: Ethernet interface

2.8 IP address selector switch

The 8-bit DIP switch is used to set the IP address. The encoding of DIP switches is done bit by bit, starting from DIP switch 1 with the least significant bit (2^0) to DIP switch 8 with the most significant bit (2^7), corresponding to decimal values: 0-255.

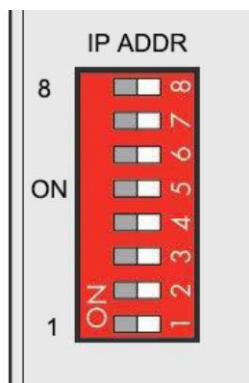


Figure 10: IP address selector switch (eg: set "0")

When the value of the DIP switch is 1111 1111 (decimal 255), the IP address is set according to the web page. The web page setting can specify the IP or set the automatic acquisition. When the web page is not set, the IP address is: 192.168.1.10.

When the value of the DIP switch is 0000 0000 – 1111 1110 (decimal 0-254), determine the 3rd byte of the IP address, and the 1st, 2nd and 4th bytes are fixed bytes, namely 192.168.xxx.253.

2.9 Factory reset button

This button is used to restore the device configuration parameters to the factory state.

How to operate:

1. When the device is in normal operation, open the flip cover;
2. Press and hold for more than 5 seconds, until all the LED lights go out, indicating that the operation is successful, and then the device will automatically restart.

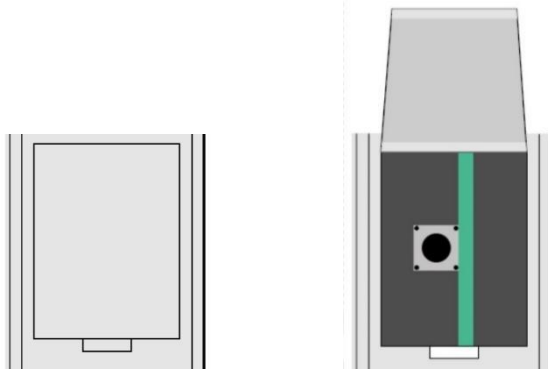


Figure 11: factory reset button

2.10 Electrical schematic

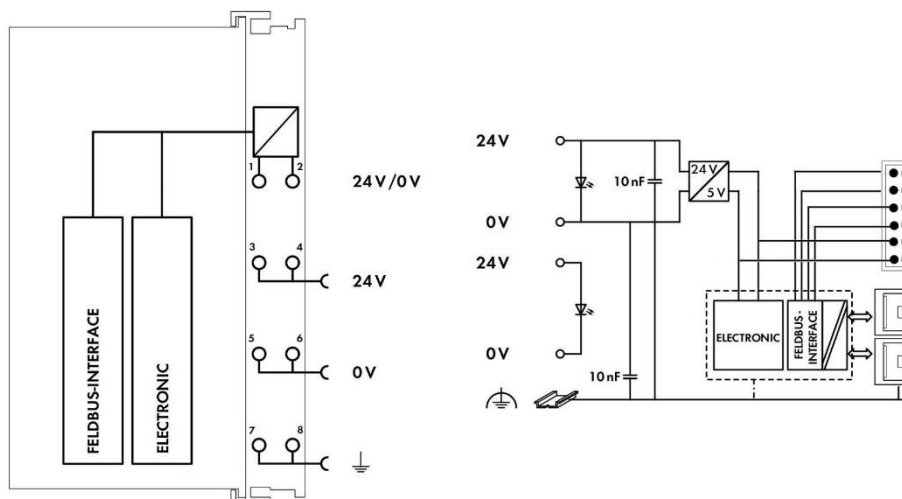


Figure 12: schematic block diagram

3. Product Installation

3.1 Installation sequence

All fieldbus controller and I/O modules must be mounted on standard DIN 35 rails.

Starting with the fieldbus controller, the bus I/O modules are assembled from left to right, with the modules mounted next to each other. All I/O modules have grooves and power jumper contacts on the right side. To avoid assembly errors, the I/O modules must be

inserted from the right side and from the top to avoid damage to the module.

Utilize a reed and groove system for reliable assembly and connection. With the automatic locking function, the individual components are securely fastened to the rail after installation.

Don't forget to install the terminal module! Always insert a terminal module (e.g. TERM) at the end of the fieldbus node to ensure correct data transmission.

3.2 Install the controller

- 1 . Snap the controller onto the DIN rail first
- 2 . Then use a tool such as a screwdriver to turn the locking cam until the locking cam engages the DIN rail.

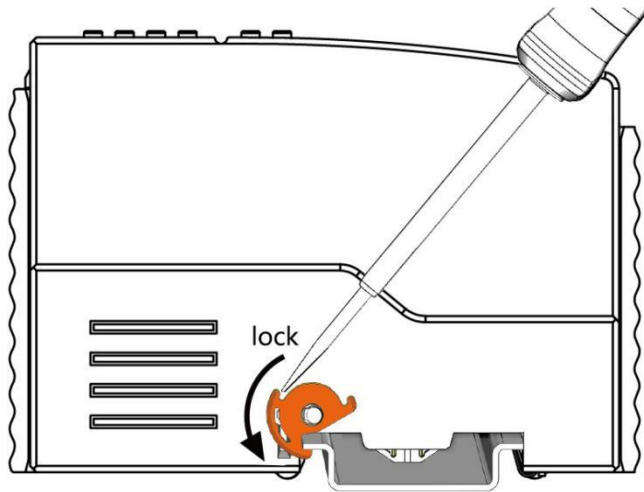


Figure 13: locking controller

3.3 Remove the controller

- 1 . Use a screwdriver to turn the locking disc cam until the locking cam no longer engages the rail.

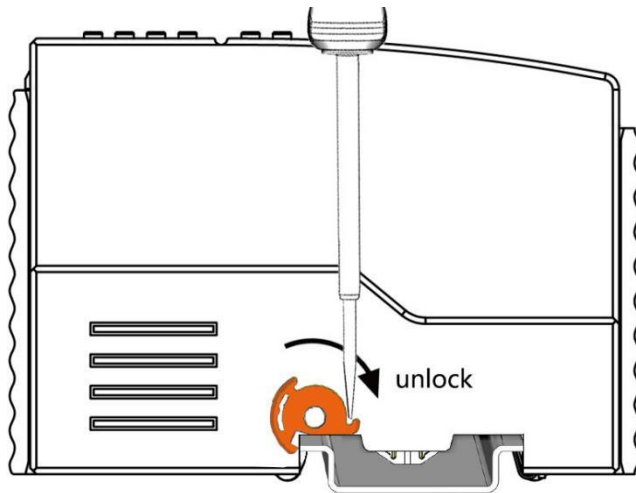


Figure 14: unlock the controller

2. Pull the release tab to remove the fieldbus controller from the assembly.

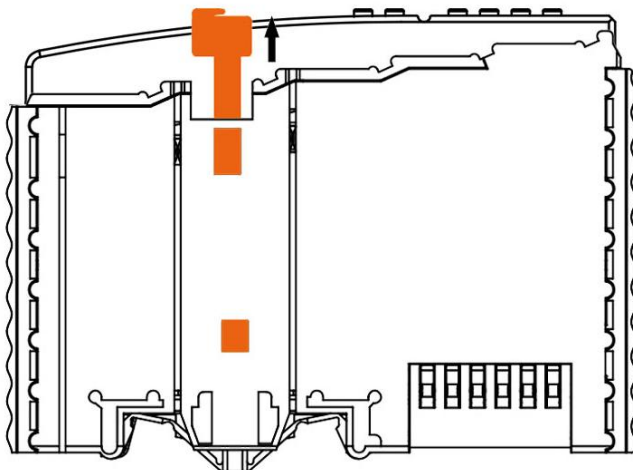


Figure 15: unlock the controller

Data or power contacts are electrically disconnected from adjacent I/O modules when the fieldbus controller/controller is removed

3.4 Insert the I/O module

- 1 . When inserting the module, make sure the spring on the module aligns with the groove on the attached controller or other I/O module



Figure 16: align the groove (example)

2 . Press the I/O module into the assembly position until the I/O module snaps into the rail.

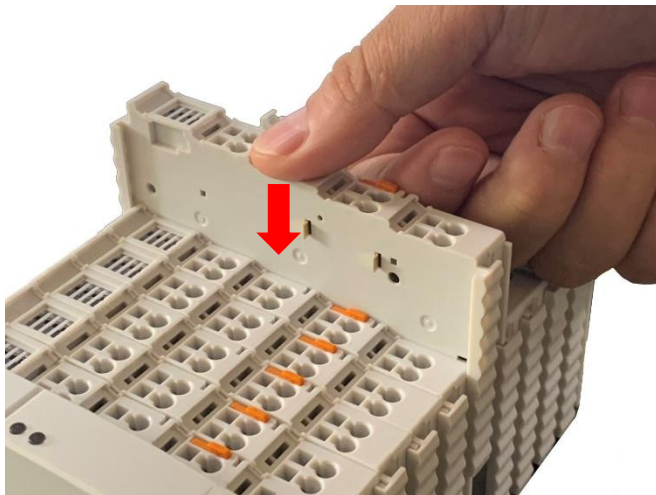


Figure 17: snap the I/O module into place (example)

Once the I/O module is installed, electrical connections to the fieldbus controller (or previous I/O module) and subsequent I/O modules are established through the data contacts and power jumper contacts.

3.5 Remove I/O module

Pull up on the latch to remove the I/O module from the assembly.

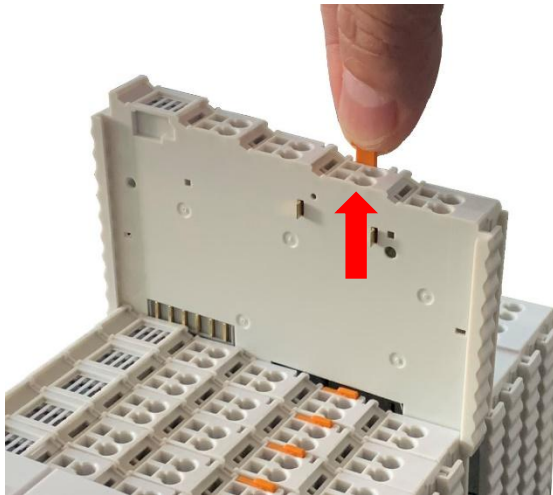


Figure 18: remove I/O module (example)

The electrical connection to the data or power jumper contacts is broken when the I/O module is removed.

4. Device Connection

4.1 Wiring

The CAGE CLAMP® connections are available for solid, stranded, and fine stranded wires. Only one wire can be connected to each CAGE CLAMP®, if there is more than one wire, it must be connected to a point.

- 1. Open the CAGE CLAMP® by first inserting the tool into the opening above the wiring.
- 2. Insert the wire into the corresponding connection opening.
- 3. For closing the CAGE CLAMP® simply remove the tool. The wire is now clamped firmly in place.

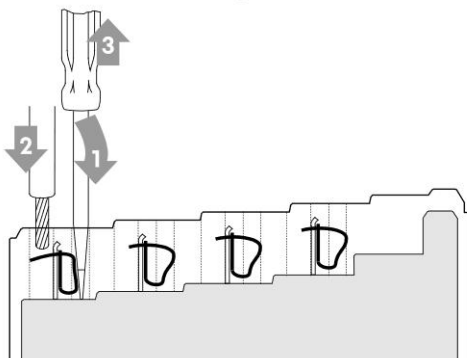


Figure 19: connecting wire

4.2 Supply Power

Both the controller and the I/O modules are powered by the power-supply module. The controller itself is fixed with a power-supply module, which supplies power to the controller and I/O modules. **When there are many I/O modules and the internal current is relatively large, it is necessary to add an independent power-supply module (model M701)**

The field bus side (Ethernet interface), the controller (system side) and the I/O modules (industrial field side) are electrically isolated from each other.

4.2.1 Power supply to Edge controller

The BL200 controller require a 24 V DC system power supply and are connected from the power-supply module's terminal block. The 5V bus voltage required by the system is converted from the 24V system voltage.

The power supply module only has proper fuse protection, please provide proper overcurrent protection externally.

Please pay attention to matching the output power and load power of the power-supply module to avoid the occurrence of excessive load current.

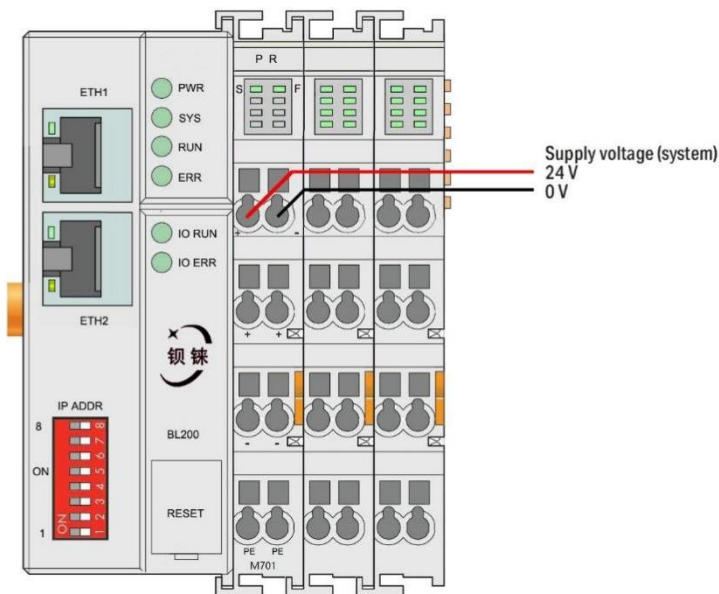


Figure 20: schematic diagram of connecting the system power supply

4.2.2 Power supply to I/O modules

The power-supply module supplies 24 VDC on the field side to power sensors and actuators.

Field power only has proper fuse protection. Without over-current protection, electronic

equipment can be damaged.

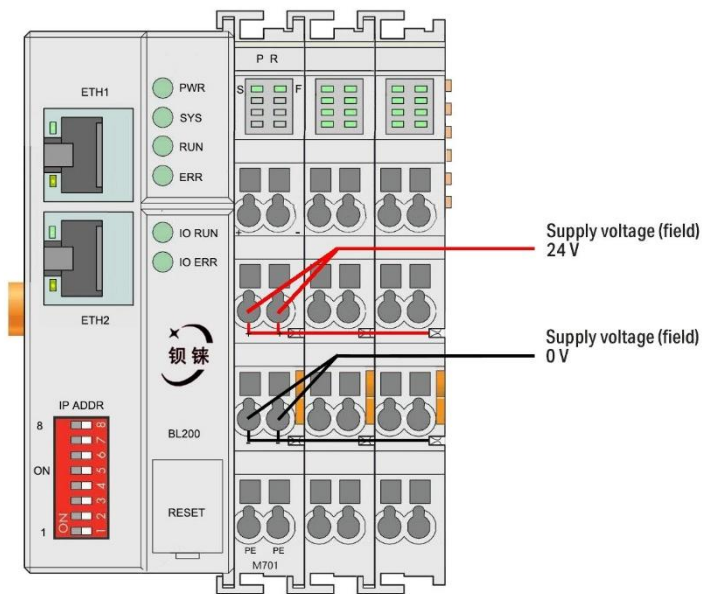


Figure 21: schematic diagram of connecting field power supply

Field power is automatically output from the power jumper contacts when the I/O module is connected. The continuous load current of the power supply across the contacts must not exceed 10 A.

The problem of excessive load power on the system or on the field can be solved by inserting additional power modules. After plugging in additional power modules, a new voltage potential may appear on the field.

In the case where electrical isolation is not required, the field power supply and system power supply can use the same power supply.

4.2.3 Earthing

When installing a cabinet with an outer shell, the cabinet must be grounded. The rail must be connected to the cabinet using screws to ensure that the rail is properly grounded. Grounding increases resistance to electromagnetic interference. Some components in an I/O system have rail contacts that consume electromagnetic interference onto the rail.

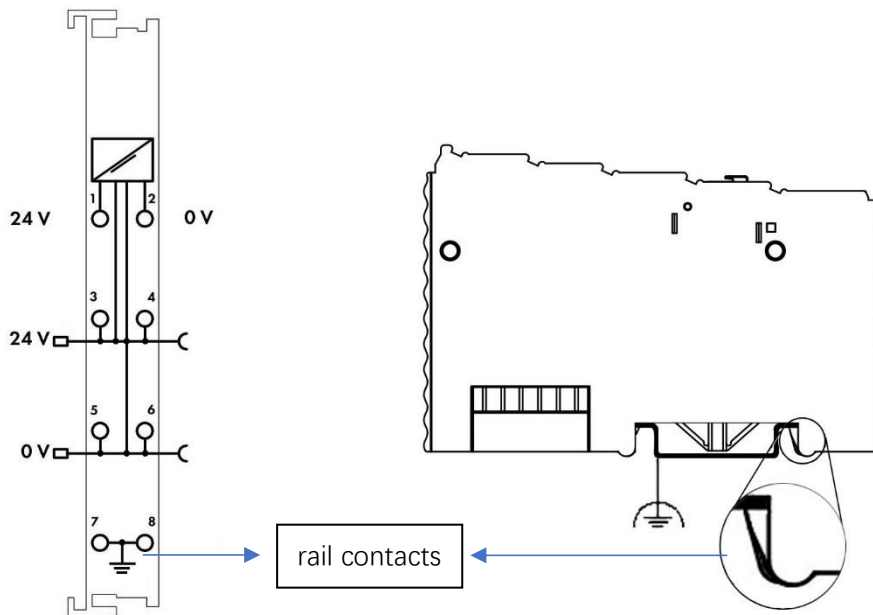


Figure 22: DIN rail contacts

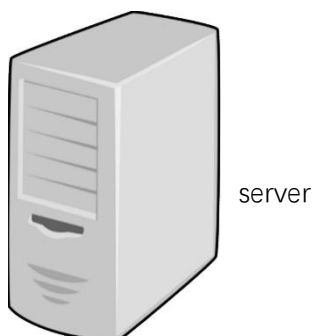
4.3 Connect BL200 devices each other and to network

The BL200 controller have 2 x RJ45 Ethernet interfaces, integrated switch function inside, work in store-and-forward operation mode, each port supports 10/100 Mbit transmission speed and full-duplex and half-duplex transmission mode.

The BL200 controller connect to the router Ethernet network via ETH2 only, while the EHT 1 is for connecting other BL200 field nodes.

The internal integrated switch supports bypass mode, which can automatically start the bypass mode when the controller system fails, and automatically maintain the link between ETH 1 and EHT 2.

The wiring of these Ethernet interfaces conforms to the 100BaseTX specification, which specifies the use of category 5 twisted pair cable as the connecting cable. Cable types S/UTP (Screened unshielded twisted pair) and STP (shielded twisted pair) can be used up to a length of 100 m.



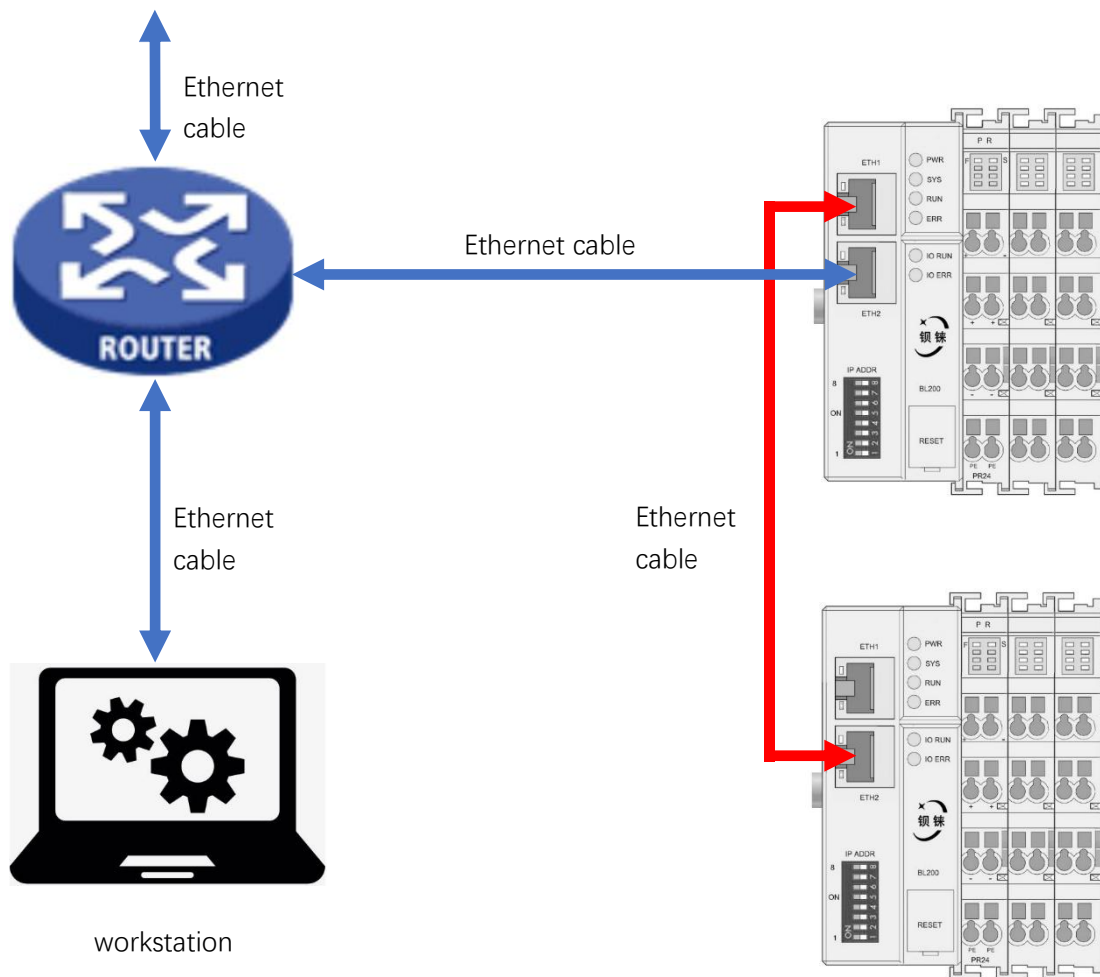


Figure 23: connect the BL200 to the network

It is also possible to connect BL200 directly to a computer via ETH2.



Figure 24: connect the BL200 to the computer

5. Web configuration

The BL200 controller's built-in web server is a browser-based configuration utility. When the node is connected to your network, you can enter the server's IP address in a web browser to access the web console.

5.1 Preparation before configuration

To successfully access the BL200 controller node, it must be properly installed and connected to the computer. In addition, configure them with correct IP addresses to keep them in the same network segment.

5.1.1 Connect computer and controller

1. Mount the fieldbus node on a DIN35 rail. Follow the installation instructions in the "Product Mounting" chapter.
2. Connect the 24 V power supply to the system power terminals.
3. The computer and the bus node can be connected in two ways, one is that the two are connected to the switch device of the local area network through the Ethernet interface; the other is that the two are directly connected point-to-point. For detailed steps, follow the instructions in the "Connection bus" chapter.
4. Turn on the power supply and start supplying power.

The controller is initialized after power-up, creates process image according to the I/O modules configuration of the fieldbus node.

5.1.2 Configure the computer IP address

On the PC, there are two ways to configure its IP address. One is to turn on the automatic IP address option on the PC's local connection to dynamically assign DHCP in the network. The other is to configure a static IP address with the controller node on the same network segment on the local connection of the PC.

The following takes Windows 7 system as an example for configuration. Windows systems are all configured similarly.

1. Click Start > Control Panel > Network and Sharing Center, and click local connection in the window that opens.

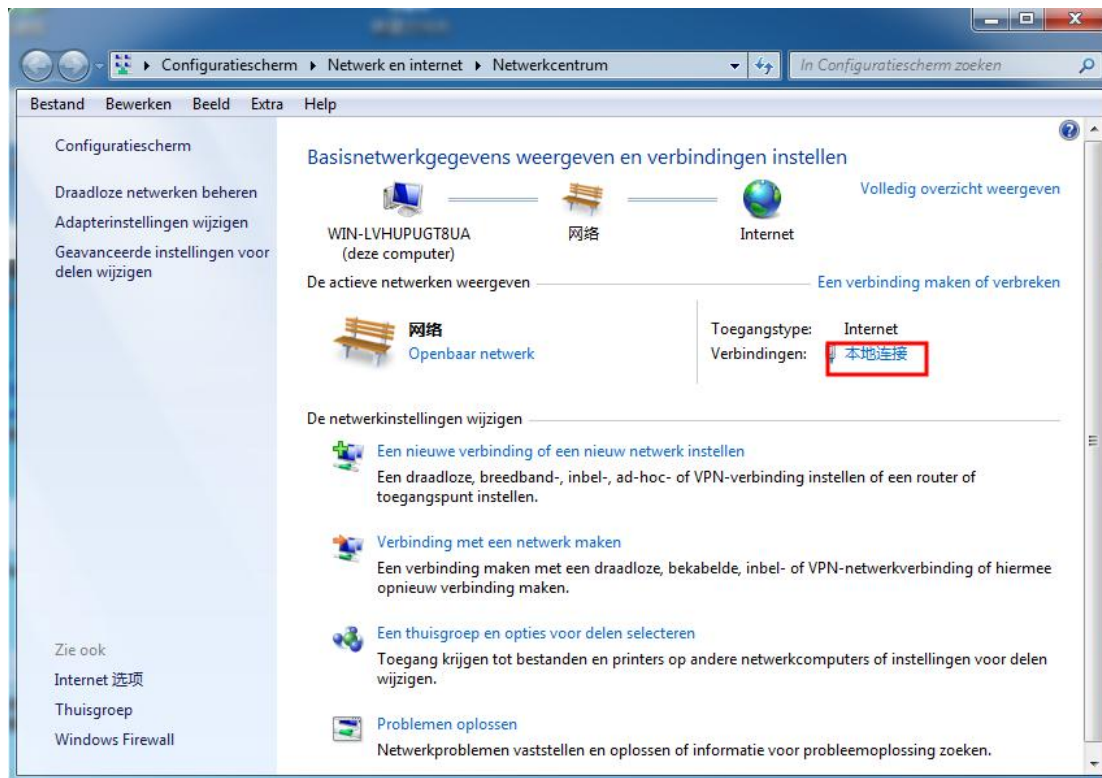


Figure 25: Network and Sharing Center

2. In the local connection status window, click Properties.

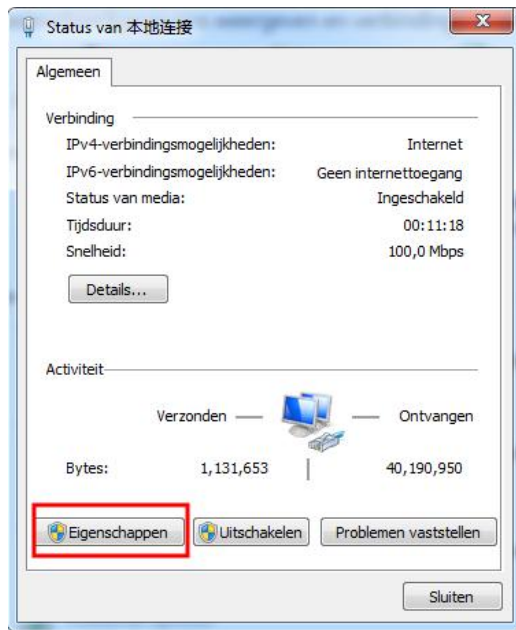


Figure 26: local connection status

3. Double-click "Internet Protocol Version 4 (TCP/IPv4)" on the local connection properties page.

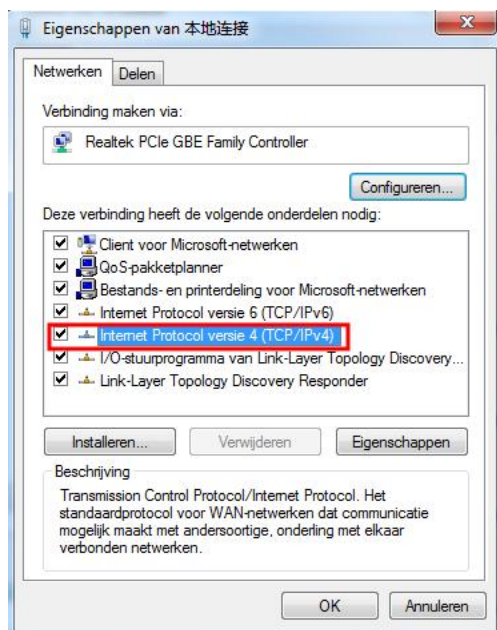


Figure 27: local connection properties

4. There are two ways to configure the IP address of the PC:
 - Obtain IP address automatically (system default mode)
 - To obtain an IP address automatically from a DHCP server, select "Obtain an IP address automatically";

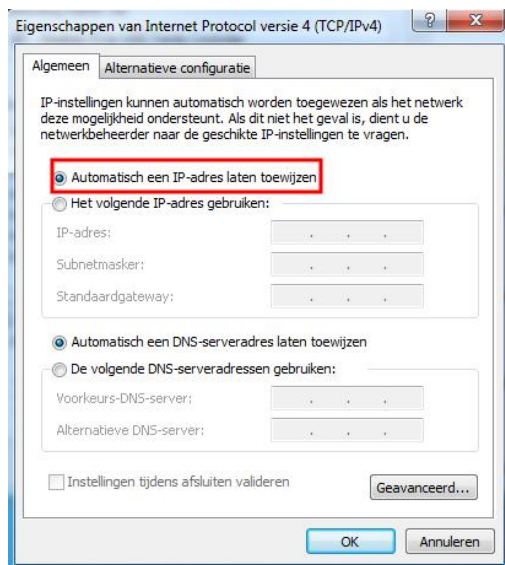


Figure 28: obtain an IP address automatically

- Set a static IP address
Select "Use the following IP address" and set the correct values for the IP address, subnet mask and default gateway.

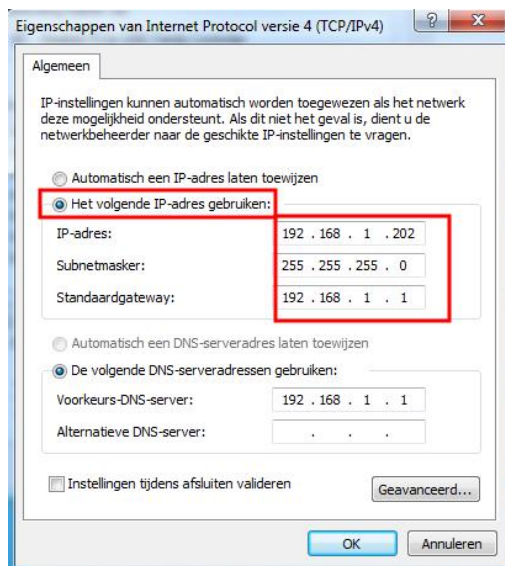


Figure 29: Set static IP

5.1.3 Configure the controller IP address

The controller has 2 ways to assign an IP address

- Assignment via built-in web page (static IP or automatic IP assignment)
- Assign via DIP switch (static IP)

DIP address selector switch definition

Table 7: DIP switch position definition

| Switch position (ON = 1) | Value | Definition |
|-----------------------------|-------|--|
| 0000 0000 --- 1111 1110 | 0-254 | Enable the DIP selector switch assignment function and determine the value of the 3rd byte. Example: 0010 0110 (22 decimal), the IP address is "192.168.22.253". |
| 1111 1111 | 255 | Enable the function of specifying IP on the web page, or select the function of DHCP automatic allocation. When the IP is not allocated through the web, the IP is 192.168.1.10. |

5.1.3.1 Configuration via web page

The fieldbus controller can be set to an IP address via the "Settings > Local Settings" page after entering the page, or it can be set to be assigned automatically. Select static address, if not set IP address, the IP is 192.168.1.10.

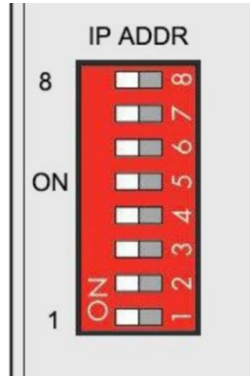


Figure 30: web page settings IP

5.1.3.2 Assign via DIP switch

Set the value of the DIP address selector switch to 0000 0000 - 1111 1110 (decimal 0 - 254), and the IP address will be assigned by the DIP switch.

The IP address consists of fixed bytes and variable bytes. The 1st, 2nd and 4th bytes are fixed bytes, the DIP selector switch determines the 3rd byte, namely: 192.168.xxx.253.

The fieldbus controller assigns an IP address via a DIP switch, and the IP address set in this way is static.

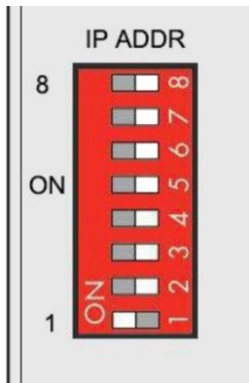


Figure 31: dip switch - assigned via dip selector switch (Example: 192.168.1.253)

5.1.4 Factory default settings

Before logging into the web configuration page, it is necessary for you to understand the following default parameters:

Modbus TCP Server Port: 502, Modbus ID: 1

IP: Determined according to the DIP switch, if the DIP switch is 1111 1111, the default IP is 192.168.1.10.

Table 8: factory default parameters

| Project | Describe |
|----------|----------|
| username | admin |
| password | 无 |

5.1.5 Login configuration page

1. Open a browser on your computer, such as IE, Chrome, Firefox, etc.
2. Enter the IP address of the controller node (192.168.1.10) in the address bar of the browser to enter the user login interface;



3. Enter "Username" and "Password" in the login interface, and then click Login.

BL200UA

Authorization Required

Please enter your username(the default is admin) and password(no password by default).

Username

Password

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

4. After successfully logging in to the web interface, the display is as follows

BL200UA Status System Settings I/O Module Serial Module OPC UA Operation&Control Logout **REFRESHING**

Status

System

| | |
|------------------|--|
| Hostname | BL200UA |
| Model | BL200UA-OPCUA IO Module |
| Firmware Version | Shenzhen Beilai Technology Co.,Ltd v1.0.11 |
| Kernel Version | 4.4.194 |
| Local Time | 2022-03-21 06:36:50 |
| Uptime | 3h 23m 36s |
| Load Average | 0.20, 0.18, 0.12 |

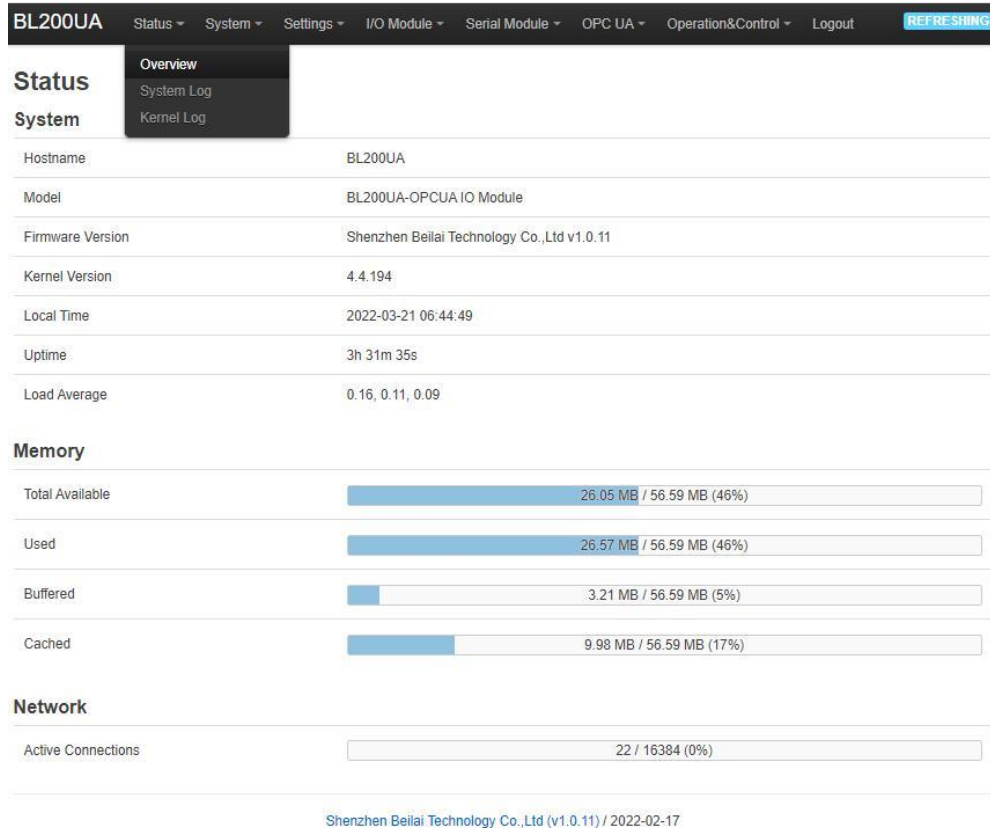
Memory

| | |
|-----------------|--|
| Total Available | <div style="width: 46%;"><div 46%;"="" style="width: 26.17 MB / 56.59 MB (46%)</div></div></td></tr><tr><td>Used</td><td><div style=" width:=""><div 5%;"="" style="width: 26.39 MB / 56.59 MB (46%)</div></div></td></tr><tr><td>Buffered</td><td><div style=" width:=""><div 17%;"="" style="width: 3.21 MB / 56.59 MB (5%)</div></div></td></tr><tr><td>Cached</td><td><div style=" width:=""><div 0%;"="" style="width: 9.78 MB / 56.59 MB (17%)</div></div></td></tr></table><h4>Network</h4><table><tr><td>Active Connections</td><td><div style=" width:=""><div 145="" 807="" 843"="" 853="" data-label="List-Group" style="width: 23 / 16384 (0%)</div></div></td></tr></table><p>Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17</p></div><div data-bbox="><ol style="list-style-type: none">5. After configuring the parameters, you need to click the "Save and Apply" button on the page to take effect.</div><div data-bbox="234 1904 756 2029" data-label="Form"><p><input type="button" value="Save & Apply"/> <input type="button" value="Save"/> <input type="button" value="Reset"/></p></div><div data-bbox="252 2060 430 2103" data-label="Page-Footer"><p>Page 31 of 78</p></div><div data-bbox="587 2060 1050 2103" data-label="Page-Footer"><p>Shenzhen Beilai Technology Co., Ltd.</p></div><div data-bbox="1259 2060 1340 2101" data-label="Page-Footer"><p>V 1.0</p></div></div></div></div></div></div> |
|-----------------|--|

5.2 Status

In the status menu, overview, system log and kernel log are provided, and you can view device parameters and device operating status.

Status > Overview



BL200UA Status System Settings I/O Module Serial Module OPC UA Operation&Control Logout **REFRESHING**

Status Overview System Log Kernel Log

System

| | |
|------------------|--|
| Hostname | BL200UA |
| Model | BL200UA-OPCUA IO Module |
| Firmware Version | Shenzhen Beilai Technology Co.,Ltd v1.0.11 |
| Kernel Version | 4.4.194 |
| Local Time | 2022-03-21 06:44:49 |
| Uptime | 3h 31m 35s |
| Load Average | 0.16, 0.11, 0.09 |

Memory

| | |
|-----------------|--|
| Total Available | |
| Used | |
| Buffered | |
| Cached | |

Network

| | |
|--------------------|--|
| Active Connections | |
|--------------------|--|

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Status > System Log

```

BL200UA  Status - System - Settings - I/O Module - Serial Module - OPC UA - Operation&Control - Logout

System Log
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Booting Linux on physical CPU 0x0
Thu Jan 1 00:00:26 1970 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 15:23:54 2020
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=0005317f
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] CPU: VIVT data cache, VIVT instruction cache
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Machine model: Nuvoton NUC980 IOT-GateWay Version: 0.1
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Memory policy: Data cache writeback
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] On node 0 totalpages: 16384
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] free_area_init_node: node 0, pgdat c0657704, node_mem_map c3f77000
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] Normal zone: 128 pages used for memmap
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] Normal zone: 0 pages reserved
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] Normal zone: 16384 pages, LIFO batch:3
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
Thu Jan 1 00:00:26 1970 kern.debug kernel: [ 0.000000] pcpu-alloc: [0] 0
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] Kernel command line: root=/dev/mtdblock2 console=ttys0,115200n8 rdinit=/sbin/init mem=64M lpi=744448
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] PID hash table entries: 256 (order: -2, 1024 bytes)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Memory: 57756K/65536K available (4538K kernel code, 305K rwdata, 1704K rodata, 188K init, 252K bss, 7780K reserved)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] Virtual kernel memory layout:
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] vector : 0xffff0000 - 0xffff1000 ( 4 kB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] fixmap : 0xffc00000 - 0xfff00000 (3072 kB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] vmalloc : 0xc4800000 - 0xff800000 ( 944 MB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] lowmem : 0xc0000000 - 0xc4000000 ( 64 MB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] modules : 0xbf000000 - 0xc0000000 ( 16 MB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] .text : 0xc0008000 - 0xc0620f54 (6244 kB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] init : 0xc0621000 - 0xc0650000 ( 188 kB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] data : 0xc0650000 - 0xc069c784 ( 306 kB)
Thu Jan 1 00:00:26 1970 kern.notice kernel: [ 0.000000] .bss : 0xc069c784 - 0xc06db8f8 ( 253 kB)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] SLUB: HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Preemptible hierarchical RCU implementation.
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] Build-time adjustment of leaf fanout to 32.
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] NR_IRQS:545
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000000] clocksource: nuc980-timer5: mask: 0xfffff max_cycles: 0xfffff, max_idle_ns: 62215505635 ns
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000033] sched_clock: 24 bits at 120kHz, resolution 8333ns, wraps every 69905062489ns
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.000741] Console: colour dummy device 80x30
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.186616] console [ttyS0] enabled
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.190091] Calibrating delay loop (skipped) preset value.. 148.88 BogoMIPS (lpi=744448)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.198174] pid_max: default: 32768 minimum: 301
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.203133] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.209708] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.218916] CPU: Testing write buffer coherency: ok
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.224983] Setting up static identity map for 0x8400 - 0x843c
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.271558] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.282316] futex hash table entries: 256 (order: -1, 3072 bytes)
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.288874] pinctrl core: initialized pinctrl subsystem
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.296433] NET: Registered protocol family 16
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.303199] DMA: preallocated 256 KiB pool for atomic coherent allocations
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.316783] <DT> nuc980_dt_device_init +
Thu Jan 1 00:00:26 1970 kern.info kernel: [ 0.348016] <DT> nuc980_dt_device_init -

```

Status > Kernel Log

```

BL200UA  Status - System - Settings - I/O Module - Serial Module - OPC UA - Operation&Control - Logout

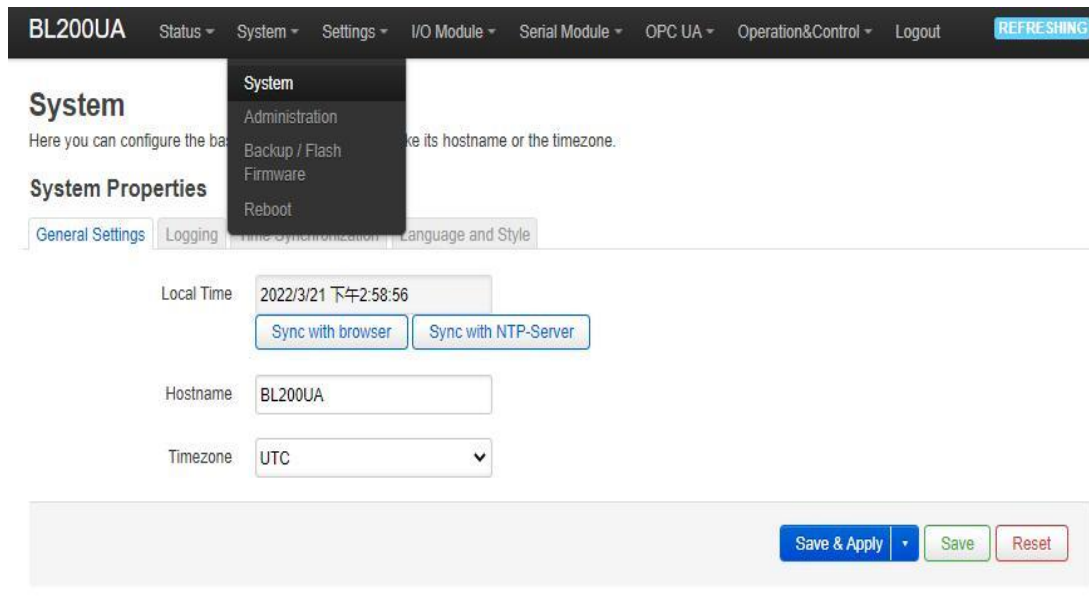
Kernel Log
[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Linux version 4.4.194 (peng@peng) (gcc version 5.4.0 (LEDE GCC 5.4.0 unknown) ) #0 PREEMPT Sat May 9 15:23:54 2020
[ 0.000000] CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=0005317f
[ 0.000000] CPU: VIVT data cache, VIVT instruction cache
[ 0.000000] Machine model: Nuvoton NUC980 IOT-GateWay Version: 0.1
[ 0.000000] Memory policy: Data cache writeback
[ 0.000000] On node 0 totalpages: 16384
[ 0.000000] free_area_init_node: node 0, pgdat c0657704, node_mem_map c3f77000
[ 0.000000] Normal zone: 128 pages used for memmap
[ 0.000000] Normal zone: 0 pages reserved
[ 0.000000] Normal zone: 16384 pages, LIFO batch:3
[ 0.000000] pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
[ 0.000000] pcpu-alloc: [0] 0
[ 0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
[ 0.000000] Kernel command line: root=/dev/mtdblock2 console=ttys0,115200n8 rdinit=/sbin/init mem=64M lpi=744448
[ 0.000000] PID hash table entries: 256 (order: -2, 1024 bytes)
[ 0.000000] Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
[ 0.000000] Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
[ 0.000000] Memory: 57756K/65536K available (4538K kernel code, 305K rwdata, 1704K rodata, 188K init, 252K bss, 7780K reserved, 0K cma-reserved)
[ 0.000000] Virtual kernel memory layout:
[ 0.000000] vector : 0xffff0000 - 0xffff1000 ( 4 kB)
[ 0.000000] fixmap : 0xffc00000 - 0xfff00000 (3072 kB)
[ 0.000000] vmalloc : 0xc4800000 - 0xff800000 ( 944 MB)
[ 0.000000] lowmem : 0xc0000000 - 0xc4000000 ( 64 MB)
[ 0.000000] modules : 0xbf000000 - 0xc0000000 ( 16 MB)
[ 0.000000] .text : 0xc0008000 - 0xc0620f54 (6244 kB)
[ 0.000000] init : 0xc0621000 - 0xc0650000 ( 188 kB)
[ 0.000000] data : 0xc0650000 - 0xc069c784 ( 306 kB)
[ 0.000000] .bss : 0xc069c784 - 0xc06db8f8 ( 253 kB)
[ 0.000000] SLUB: HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[ 0.000000] Preemptible hierarchical RCU implementation.
[ 0.000000] Build-time adjustment of leaf fanout to 32.
[ 0.000000] NR_IRQS:545
[ 0.000000] clocksource: nuc980-timer5: mask: 0xfffff max_cycles: 0xfffff, max_idle_ns: 62215505635 ns
[ 0.000033] sched_clock: 24 bits at 120kHz, resolution 8333ns, wraps every 69905062489ns
[ 0.000741] Console: colour dummy device 80x30
[ 0.186616] console [ttyS0] enabled
[ 0.190091] Calibrating delay loop (skipped) preset value.. 148.88 BogoMIPS (lpi=744448)
[ 0.198174] pid_max: default: 32768 minimum: 301
[ 0.203133] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[ 0.209708] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
[ 0.218916] CPU: Testing write buffer coherency: ok
[ 0.224983] Setting up static identity map for 0x8400 - 0x843c
[ 0.271558] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[ 0.282316] futex hash table entries: 256 (order: -1, 3072 bytes)
[ 0.288874] pinctrl core: initialized pinctrl subsystem
[ 0.296433] NET: Registered protocol family 16
[ 0.303199] DMA: preallocated 256 KiB pool for atomic coherent allocations
[ 0.316783] <DT> nuc980_dt_device_init +

```

5.3 System

5.3.1 System

System Properties > General Settings



Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Tabel 9: System > System Properties > General Settings

| Project | Describe | Default |
|------------|--|---------|
| Local time | Displays the current time of the device. You can click the "Sync browser time" or "Sync with NTP server" button to update the device time. | -- |
| Hostname | The device name can be customized to easily distinguish between multiple devices. | BL200 |
| Timezone | The time zone can be selected via the drop down menu | UTC |

System Properties > Logging

BL200UA
Status ▾
System ▾
Settings ▾
I/O Module ▾
Serial Module ▾
OPC UA ▾
Operation&Control ▾
Logout
REFRESHING

System

Here you can configure the basic aspects of your device like its hostname or the timezone.

System Properties

General Settings
Logging
Time Synchronization
Language and Style

System log buffer size:
📄 kiB

External system log server:

External system log server port:

External system log server protocol: ▾

Write system log to file:

Log output level: ▾

Cron Log Level: ▾

Save & Apply ▾
Save
Reset

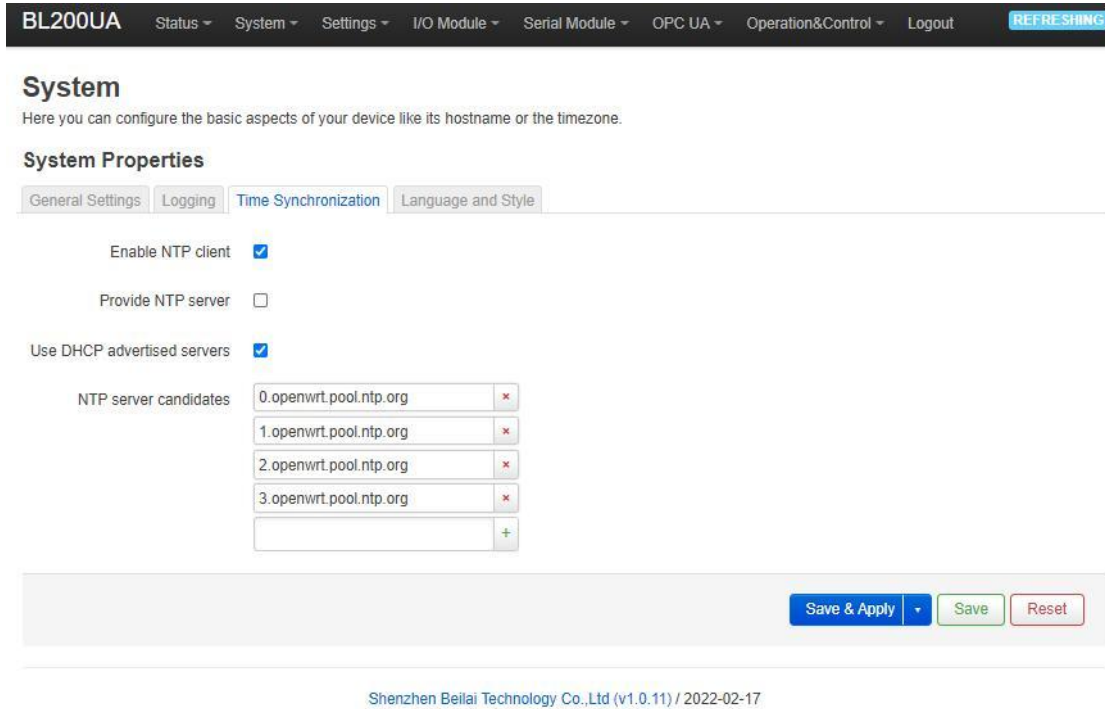
Shenzhen Beilai Technology Co., Ltd (v1.0.11) / 2022-02-17

Tabel 10: System > System Properties > Logging

| Project | Describe | Default |
|-------------------------------------|----------|---------|
| System log buffer size | | 64 |
| External system log server | | |
| External system log server port | | |
| External system log server protocol | | |
| Write system log to file | | |
| Log output level | | |
| Cron log level | | |

System Properties > Time Synchronization

An NTP server can be set to synchronize time.



System Properties > Language and Style

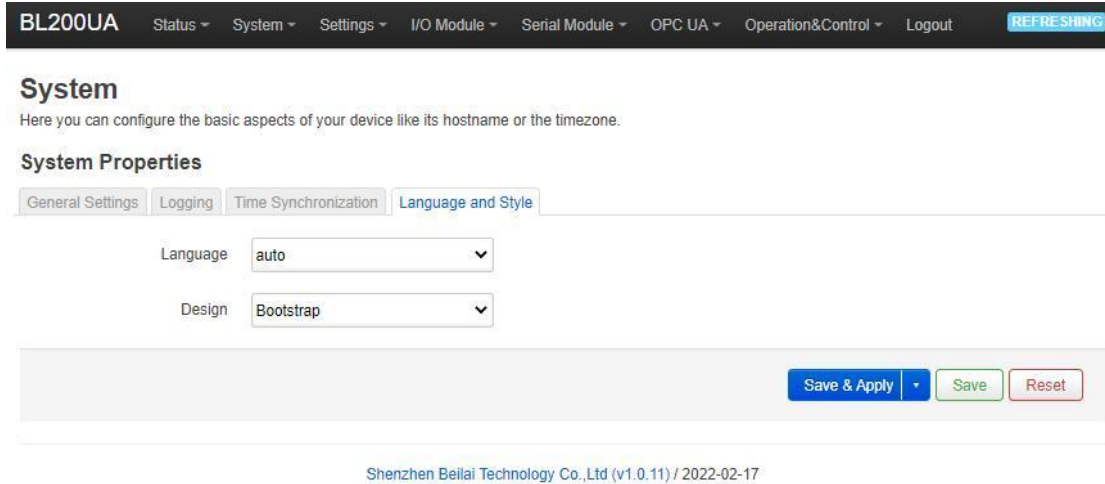


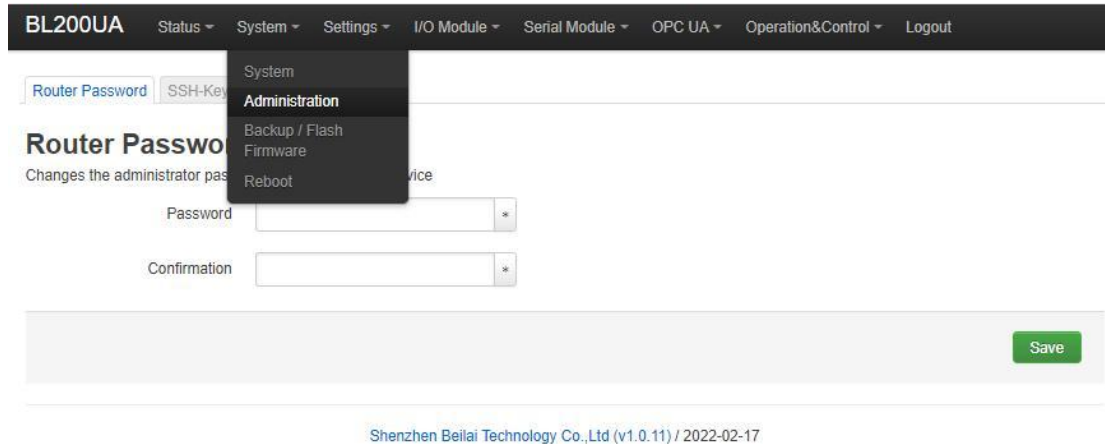
Table 11: System > System Properties > Language and Style

| Project | Describe | Default |
|----------|---|-----------|
| Language | Available in auto, English, Chinese (Chinese) | auto |
| Design | Currently only Bootstrap is supported. | Bootstrap |

5.3.2 Administration

Administration > Router Password

Change the administrator password for accessing the device.



BL200UA Status System Settings I/O Module Serial Module OPC UA Operation&Control Logout

Router Password SSH-Keys

Router Password

Changes the administrator password for accessing the device.

Password

Confirmation

Save

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Administration > SSH Keys

Public keys allow for the passwordless SSH logins with a higher security compared to the use of plain passwords. In order to upload a new key to the device, paste an OpenSSH compatible public key line or drag a `.pub` file into the input field.



BL200UA Status System Settings I/O Module Serial Module OPC UA Operation&Control Logout

Router Password SSH-Keys

SSH-Keys

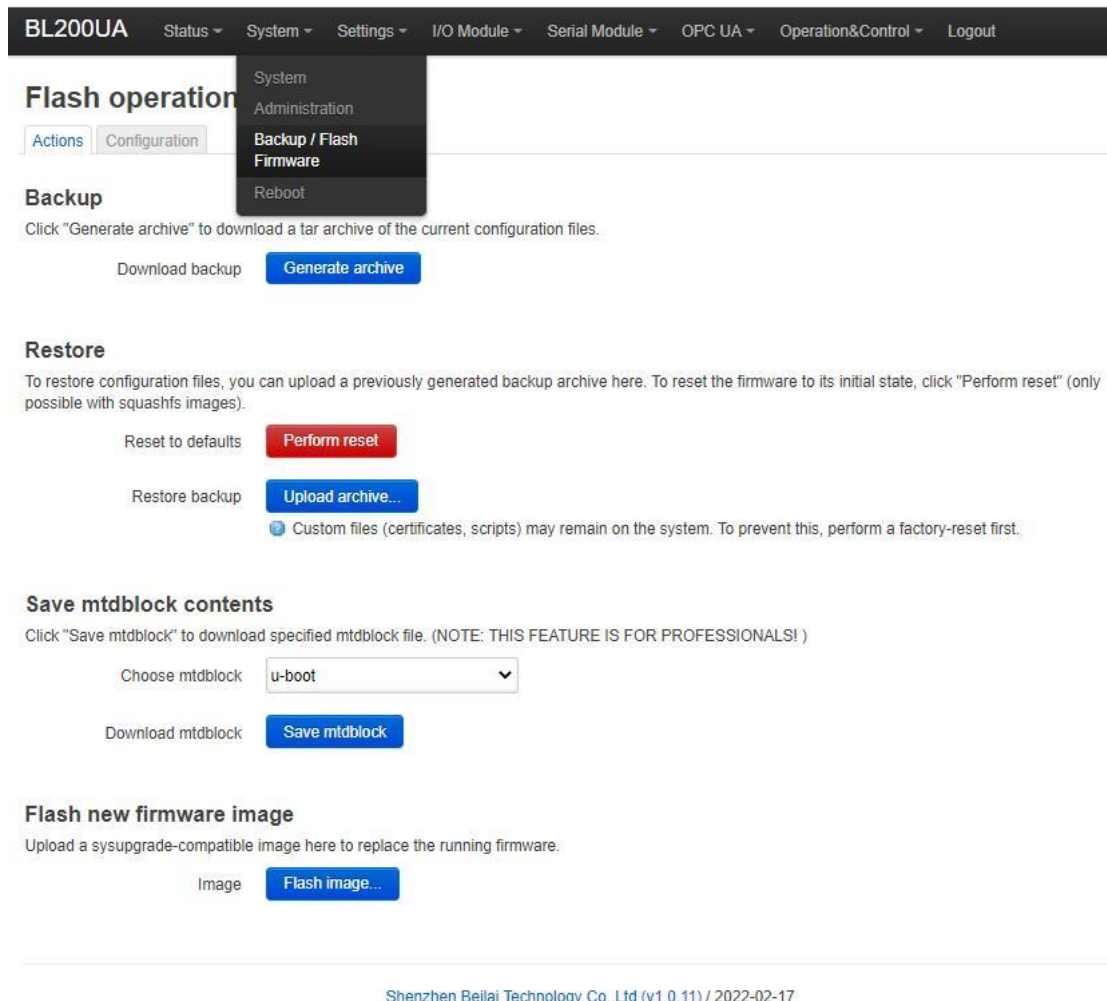
Public keys allow for the passwordless SSH logins with a higher security compared to the use of plain passwords. In order to upload a new key to the device, paste an OpenSSH compatible public key line or drag a `.pub` file into the input field.

No public keys present yet.

Paste or drag SSH key file... Add key

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

5.3.3 Backup/Flash Firmware



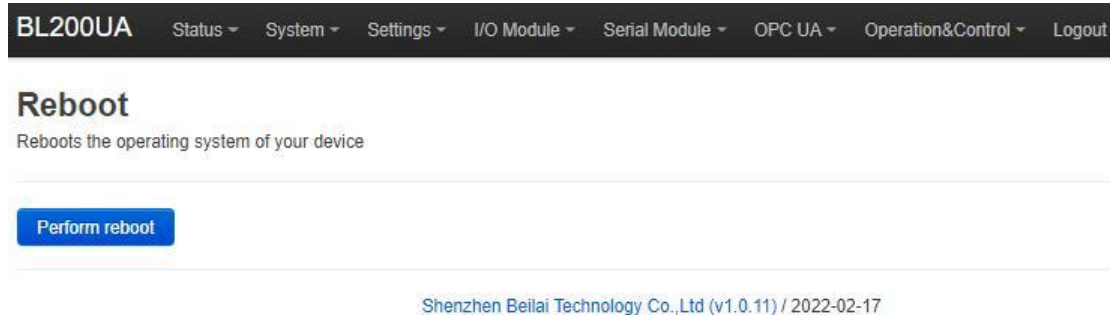
The screenshot shows the BL200UA web interface. At the top, there is a navigation bar with 'BL200UA' and several menu items: Status, System, Settings, I/O Module, Serial Module, OPC UA, Operation&Control, and Logout. A dropdown menu is open under 'System', showing options: System, Administration, **Backup / Flash Firmware**, and Reboot. Below the navigation, the 'Flash operation' section is active, with 'Actions' and 'Configuration' tabs. The 'Backup' section contains a 'Generate archive' button. The 'Restore' section contains 'Perform reset' and 'Upload archive...' buttons. The 'Save mtddblock contents' section has a dropdown menu set to 'u-boot' and a 'Save mtddblock' button. The 'Flash new firmware image' section has a 'Flash image...' button. At the bottom of the page, it says 'Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17'.

Tabel 12: System > Backup/ Flash Firmware > Actions

| Project | Describe | Default |
|--------------------------|--|---------|
| Backup | Click "Generate archive" to download a tar archive of the current configuration files. | -- |
| Restore | To restore configuration files, you can upload a previously generated backup archive here. To reset the firmware to its initial state, click "Perform reset" (only possible with squashfs images). | -- |
| Save mtddblock contents | Click "Save mtddblock" to download specified mtddblock file. (NOTE: THIS FEATURE IS FOR PROFESSIONALS!) | -- |
| Flash new firmware image | Upload a sysupgrade-compatible image here to replace the running firmware. | -- |

5.3.4 Reboot

Clicking on "Perform reboot" will reboot your device.



5.4 Settings

Device settings

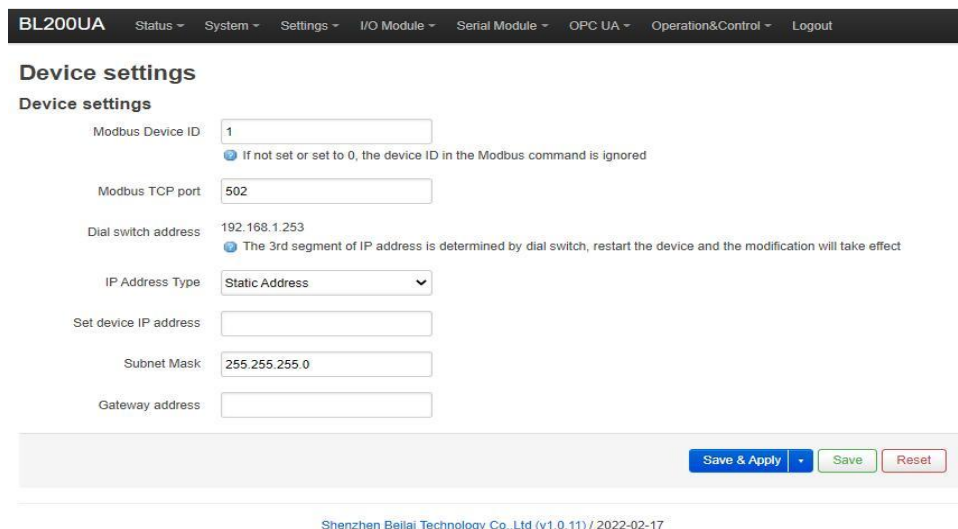


Table 13: Settings > Device settings

| Project | Describe | Default |
|-----------------------|---|---------|
| Modbus Device ID | Modbus device ID range is 1~247. | 1 |
| Modbus TCP port | Modbus TCP protocol port number, which can be customized. | 502 |
| Dial switch address | Displays the IP address set by the Dial switch. | |
| IP address type | Select from "Static Address", "Dynamic Address(DHCP)". | |
| Set device IP address | The IP address of the device can be set by yourself, and it needs to be restarted to take effect after setting. | -- |
| Subnet mask | Set IP subnet mask | |

| | | |
|-----------------|------------------------|--|
| Gateway address | Set IP gateway address | |
|-----------------|------------------------|--|

5.5 I/O modules

After power-up, the controller automatically recognizes all I/O modules connected to it and creates an internal local process image based on the module type, data width and the module's position in the node.

If I/O modules are added, changed or removed, a new process image is created and the process data addresses change. When adding an I/O module, the process data of all previous I/O modules must be considered.

The controller can connect up to 32 I/O modules, including digital input and output, analog input and output and special function modules.

| BL200UA | | | | | | | | |
|---|-------------|-------------|----------------|----------------|-------------------|--------------|-----------|--------------------------------|
| Status - System - Settings - I/O Module - Serial Module - OPC UA - Operation&Control - Logout | | | | | | | | |
| IO status | | | | | | | | |
| IO Slot | Module Name | Module Type | Channel Number | Modbus Address | 24V Address-State | Soft Version | IO Status | Channel Status |
| 1 | M1081 | DI | 8 | 2000-2007 | 9001-Power On | 5 | Normal | Channel Status |
| 2 | M2082 | DO | 8 | 1000-1007 | 9002-Power On | 5 | Normal | Channel Status |
| 3 | M3041 | AI | 4 | 3000-3006 | 9003-Power On | 5 | Normal | Channel Status |
| 4 | M4044 | AO | 4 | 4000-4006 | 9004-Power On | 5 | Normal | Channel Status |
| 5 | M6021 | COM | 2 | 0-0 | 9005-Power On | 5 | Normal | Channel Status |

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Tabel 14: I/O Module > I/O Status

| Project | Describe |
|-------------------|--|
| IO slot | The order of IO modules in the card slot, the first module card position close to the controller is 1, and the following ones are 2 3 4... |
| Module name | Detailed model of IO module |
| Module type | IO module function type |
| Channel Number | Data width of IO module |
| Modbus Address | Process map address of the IO module inside the controller |
| 24V Address State | Power supply status on the field side of the IO module, digital, 1 bit |
| Software version | IO module internal firmware version |
| IO status | IO module and controller communication status |
| Channel status | Click to view and set the parameters of different types of IO modules |

5.5.1 Digital input module

The digital input module can provide two types of data, one is the current input state value, Boolean type; the other is the counter value, 32-bit numerical type, which supports the clear function.

BL200UA
Status ▾
System ▾
Settings ▾
I/O Module ▾
Serial Module ▾
OPC UA ▾
Operation&Control ▾
Logout

IO status

IO Slot:1,Module Type:DI,Module Name:M1081

| Channels | Modbus Address | Value |
|----------|----------------|-------|
| 1 | 2000 | Open |
| 2 | 2001 | Open |
| 3 | 2002 | Open |
| 4 | 2003 | Open |
| 5 | 2004 | Open |
| 6 | 2005 | Open |
| 7 | 2006 | Open |
| 8 | 2007 | Open |

DI Count

| Channel | Modbus Address | Value | Clear |
|---------|----------------|-------|--------------------------------------|
| 1 | 5000 | 0 | <input type="button" value="Clear"/> |
| 2 | 5002 | 0 | <input type="button" value="Clear"/> |
| 3 | 5004 | 0 | <input type="button" value="Clear"/> |
| 4 | 5006 | 0 | <input type="button" value="Clear"/> |
| 5 | 5008 | 0 | <input type="button" value="Clear"/> |
| 6 | 5010 | 0 | <input type="button" value="Clear"/> |
| 7 | 5012 | 0 | <input type="button" value="Clear"/> |
| 8 | 5014 | 0 | <input type="button" value="Clear"/> |

Tabel 15: Digital Input Modules > IO Status

| Project | Describe |
|----------------|---|
| Channels | Channel number of the digital input module |
| Modbus Address | Process map address of Boolean status data inside the |

| | |
|-------|--|
| | controller |
| Value | Display the current input state, open: logic 0, close: logic 1 |

Tabel 16: Digital Input Modules > DI Count

| Project | Describe |
|----------------|--|
| Channels | Channel number of the digital input module |
| Modbus Address | Process map address of the count value inside the controller |
| Value | Display the current input count value, 32-bit unsigned integer |
| Clear | Can clear the current channel counter value |

5.5.2 Digital output module

BL200UA Status ▾ System ▾ Settings ▾ I/O Module ▾ Serial Module ▾ OPC UA ▾ Operation&Control ▾ Logout

IO status

IO Slot:2,Module Type:DO,Module Name:M2082

| Channels | Modbus Address | Value | PowerOn Status | Open/Close |
|----------|----------------|-------|----------------|------------|
| 1 | 1000 | Open | Open ▾ | Open/Close |
| 2 | 1001 | Open | Open ▾ | Open/Close |
| 3 | 1002 | Open | Open ▾ | Open/Close |
| 4 | 1003 | Open | Open ▾ | Open/Close |
| 5 | 1004 | Open | Open ▾ | Open/Close |
| 6 | 1005 | Open | Open ▾ | Open/Close |
| 7 | 1006 | Open | Open ▾ | Open/Close |
| 8 | 1007 | Open | Open ▾ | Open/Close |

Back to Overview
Save & Apply ▾
Save
Reset

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

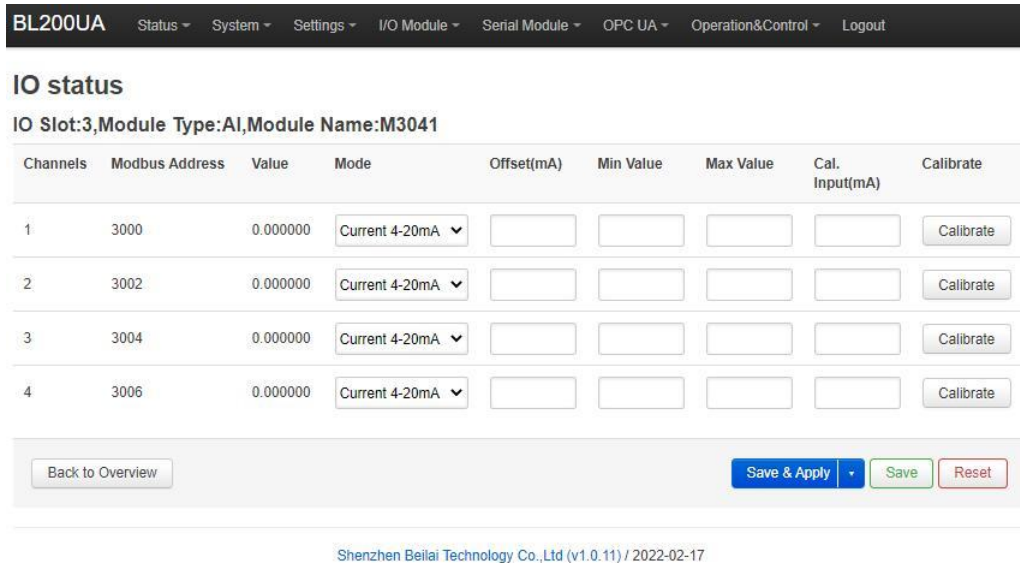
Tabel 17: Digital output module

| Project | Describe |
|-----------------|--|
| Channels | Channel number of the digital output module |
| Modbus address | Process map address of the digital output boolean data inside the controller |
| Value | Display the current output state, open: 0, close: 1 |
| power-on status | Set the state of DO after power-on, select from "open", "close", "last" |

| | |
|------------|--|
| open/close | Can control the current channel output state |
|------------|--|

5.5.3 Analog input module

The analog input (AI) type module supports setting parameters through the controller web page, so that the data conversion is automatically realized inside the module, and the actual engineering value corresponding to the sensor can be directly output.



Shenzhen Beilai Technology Co., Ltd (v1.0.11) / 2022-02-17

Tabel 18: Analog input module

| Project | Describe |
|---------------------|---|
| Channels | Channel number of the analog input module |
| Modbus Address | Process map address of the analog input module inside the controller |
| Value | Display the actual project value input by the current channel, 32-bit single-precision floating-point type |
| Mode | Different models of analog input modules have different options, please refer to the specific analog input I/O module manual for details. |
| Offset(mA) | The offset can be used to adjust the acquisition and actual error. |
| Min Value | Sensor range minimum |
| Max Value | sensor range maximum |
| Calibrate Input(mA) | To calibrate the AI, enter the actual current of the AI. |
| Calibrate | Click "Calibrate" to confirm the calibration AI. |

There is a linear relationship between the electrical signal value of the analog input module (usually a sensor) and the actual engineering value. Their formulas are as follows (take 4-20mA as an example):

Actual engineering value = (current value - 4) * ((maximum - minimum) / (20 - 4)) + minimum

Take the 4-20mA type water level sensor to measure the depth of the water tower as an example:

The known water level sensor range is 0-100m, the current data is 5.6mA, and the depth of the water tower is calculated:

Into the formula:

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

the depth of the water tower is 10m

5.5.4 Analog output module

BL200UA Status ▾ System ▾ Settings ▾ I/O Module ▾ Serial Module ▾ OPC UA ▾ Operation&Control ▾ Logout

IO status

IO Slot:4,Module Type:AO,Module Name:M4044

| Channels | Modbus Address | Value | Mode | Offset(V) | Min Value | Max Value | Set Value |
|----------|----------------|----------|-----------------|----------------------|----------------------|----------------------|----------------------|
| 1 | 4000 | 0.000000 | Voltage 0-10V ▾ | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |
| 2 | 4002 | 0.000000 | Voltage 0-10V ▾ | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |
| 3 | 4004 | 0.000000 | Voltage 0-10V ▾ | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |
| 4 | 4006 | 0.000000 | Voltage 0-10V ▾ | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |

Back to Overview
Save & Apply ▾
Save
Reset

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Tabel 19: Analog output module

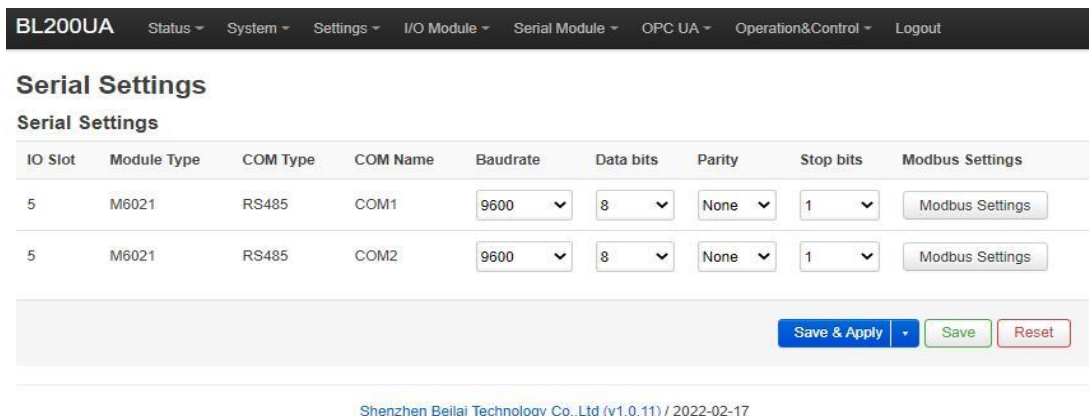
| Project | Describe |
|----------------|---|
| Channels | Channel number of the analog output module |
| Modbus Address | Process map address of the analog output module inside the controller |
| Value | Display the actual project value output by the current channel, 32-bit single-precision floating-point type |
| Mode | Different models of analog output modules have different options, please refer to the specific analog output I/O module manual for details. |

| | |
|-----------|--|
| Offset | Adjust the setting and the actual error |
| Min value | Actual engineering value minimum value |
| Max value | Actual engineering value maximum value |
| Set value | You can set the actual project value required for the output |

5.6 Serial port RS485 module

Various sensors, meters and other devices that support Modbus RTU protocol can be connected to the edge controller through the serial port module. It allows process mapping between external sensor data and the controller via the local bus.

5.6.1 Serial port settings



BL200UA Status System Settings I/O Module Serial Module OPC UA Operation&Control Logout

Serial Settings

Serial Settings

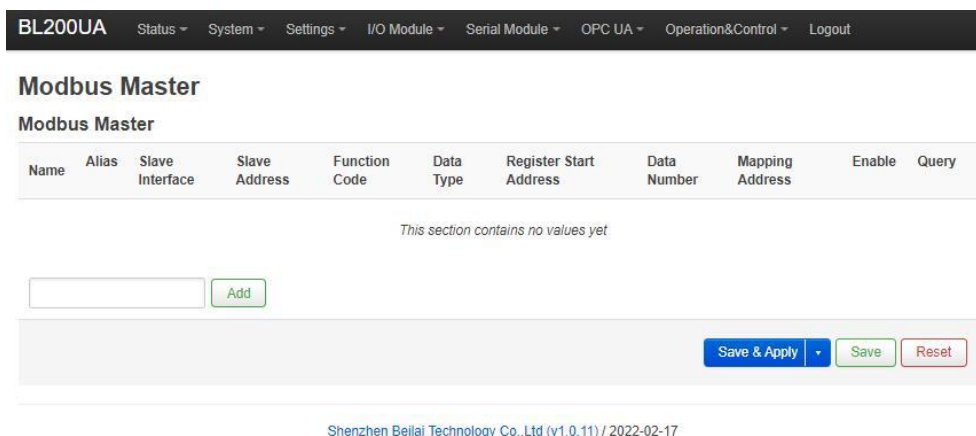
| IO Slot | Module Type | COM Type | COM Name | Baudrate | Data bits | Parity | Stop bits | Modbus Settings |
|---------|-------------|----------|----------|----------|-----------|--------|-----------|-----------------|
| 5 | M6021 | RS485 | COM1 | 9600 | 8 | None | 1 | Modbus Settings |
| 5 | M6021 | RS485 | COM2 | 9600 | 8 | None | 1 | Modbus Settings |

Save & Apply Save Reset

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

5.6.2 Modbus settings

Modbus settings are used to add Modbus RTU devices to the serial communication I/O module.



BL200UA Status System Settings I/O Module Serial Module OPC UA Operation&Control Logout

Modbus Master

Modbus Master

| Name | Alias | Slave Interface | Slave Address | Function Code | Data Type | Register Start Address | Data Number | Mapping Address | Enable | Query |
|-------------------------------------|-------|-----------------|---------------|---------------|-----------|------------------------|-------------|-----------------|--------|-------|
| This section contains no values yet | | | | | | | | | | |

Add

Save & Apply Save Reset

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Enter the custom data name in the input box and click Add

BL200UA Status ▾ System ▾ Settings ▾ I/O Module ▾ Serial Module ▾ OPC UA ▾ Operation&Control ▾ Logout

Modbus Master

Modbus Master

| Name | Alias | Slave Interface | Slave Address | Function Code | Data Type | Register Start Address | Data Number | Mapping Address | Enable | Query |
|---|-------|------------------------------------|---------------|---------------|-----------|------------------------|-------------|-----------------|--------|--|
| This section contains no values yet | | | | | | | | | | |
| <input style="border: 2px solid red;" type="text"/> | | <input type="button" value="Add"/> | | | | | | | | |
| | | | | | | | | | | <input type="button" value="Save & Apply"/> ▾ <input type="button" value="Save"/> <input type="button" value="Reset"/> |

Shenzhen Beilai Technology Co., Ltd (v1.0.11) / 2022-02-17

The configuration box pops up to configure

Modbus Master - 1

Alias:

Slave Interface: ▾

Slave Address:

Function Code: ▾

Register Start Address:

Data Number:

Mapping address alloc: ▾

Polling period(s):
ⓘ If not set, the default is 0.2s

Response timeout(s):
ⓘ If not set, the default is 0.5s

Tabel 20: Modbus master

| Project | Describe |
|------------------------|--|
| Alias | Device nickname can be used to distinguish data |
| Slave Interface | Select serial channel |
| Slave address | Slave device address, range 1-247 |
| Function code | Select according to the slave data type, including: "01", "02", "03", "04" |
| Register start address | Register start address of slave data |
| Data number | Number of slave data |
| Mapping address | Support distribution method: |

| | |
|----------------------|--|
| alloc | <p>auto</p> <p>According to different data types, the system automatically allocates down the starting address of the mapping, and the addresses are continuous.</p> <p>manual</p> <p>Manual allocation allows mapping addresses to be allocated across segments</p> |
| Polling period (s) | The interval between two adjacent polling commands |
| Response timeout (s) | After sending the command to the slave station, wait for the maximum time for the slave station to return data. If the time exceeds this time, the slave station will be considered to have no response. |

For the built slave, you can modify, delete, and view data, or you can disable collection.

BL200UA
Status ▾
System ▾
Settings ▾
I/O Module ▾
Serial Module ▾
OPC UA ▾
Operation&Control ▾
Logout

Modbus Master

Modbus Master

| Name | Alias | Slave Interface | Slave Address | Function Code | Data Type | Register Start Address | Data Number | Mapping Address | Enable | Query |
|------|-------|-----------------|---------------|---------------|-----------|------------------------|-------------|-----------------|-------------------------------------|--|
| 1 | 1 | COM1 | 1 | 1 | Bool | 0 | 1 | 10000-10000 | <input checked="" type="checkbox"/> | <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;"> Q u e r y Edit Delete </div> |

Add

Save & Apply ▾
Save
Reset

5.7 OPC UA settings

BL200UA [Status](#) [System](#) [Settings](#) [I/O Module](#) [Serial Module](#) [OPC UA](#) [Operation&Control](#) [Logout](#)

OPC UA settings

OPC UA settings

OPC UA Name

Port

Security Policy

Message Security Mode

Certificate

Private key

Allow Anonymous

Username

Password

Data select

Model File(.xml)

Dependent model files

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Tabel 21: OPC UA settings

| Project | Describe | Default |
|-----------------------|---|---------|
| OPC UA name | OPC UA server name | |
| Port | OPC UA service port number | 4840 |
| Security policy | None basic128rsa15 basic256 basic256sha256 aes128sha256rsaoaep All security policies | None |
| Message security mode | sign Sign and encrypt | |
| Certificate | OPC UA certificate, click the uploaded certificate to load the configuration page. | |
| Private key | OPC UA private key, click on the uploaded | |

| | | |
|-----------------------|--|----------|
| | certificate to load it into the configuration page. | |
| Allow anonymous | Whether to enable user name and password login | |
| Username | Fill in the username | |
| Password | Fill in password | |
| Data select | all data Select data point information model | all data |
| Select data point | You can select the data points you want to read. "Data selection" option to select "select data point" to have this option | |
| Model file (.xml) | Upload the information model (.xml) file, select "Information Model" in the "Data Selection" item to have this option | |
| Dependent model files | Select the number of information models to reference, up to 5 can be selected. | |
| Dependent Models 1-5 | Upload the information model (.xml) file to be referenced | |

Note: For a customized information model, the data point description item must be in the format of REG + Modbus address during modeling. For example, DO1 point description item fills in REG1000, and other items are customized.

5.8 Other functions

- 1、 Logic operation and control function
 - 2、 OpenVPN function
 - 3、 Uplink protocols: Huawei Cloud IoT, Ali Cloud IoT, AWS IoT, MQTT, thingsboard, sparkplugB, Kingpigeon Cloud and other protocols.
- These functions are introduced in the subsequent manual.

6. Fieldbus Communication

6.1 Modbus

6.1.1 Overview

Modbus is an open, manufacturer-independent fieldbus standard protocol for a variety of applications in manufacturing and process automation.

MODBUS is an application layer messaging protocol at layer 7 of the OSI model that

enables client/server communication between devices connected on different types of buses or networks.

Several commonly used networks are as follows:

- TCP/IP over Ethernet。
- Asynchronous serial transmission of multiple media (wired: EIA/TIA-232-E, EIA-422, EIA/TIA-485-A; optical fiber, radio, etc.).
- MODBUS PLUS, high-speed token.

MODBUS is a request/response protocol that provides services specified by function codes.

The MODBUS protocol allows easy communication within all types of network architectures.

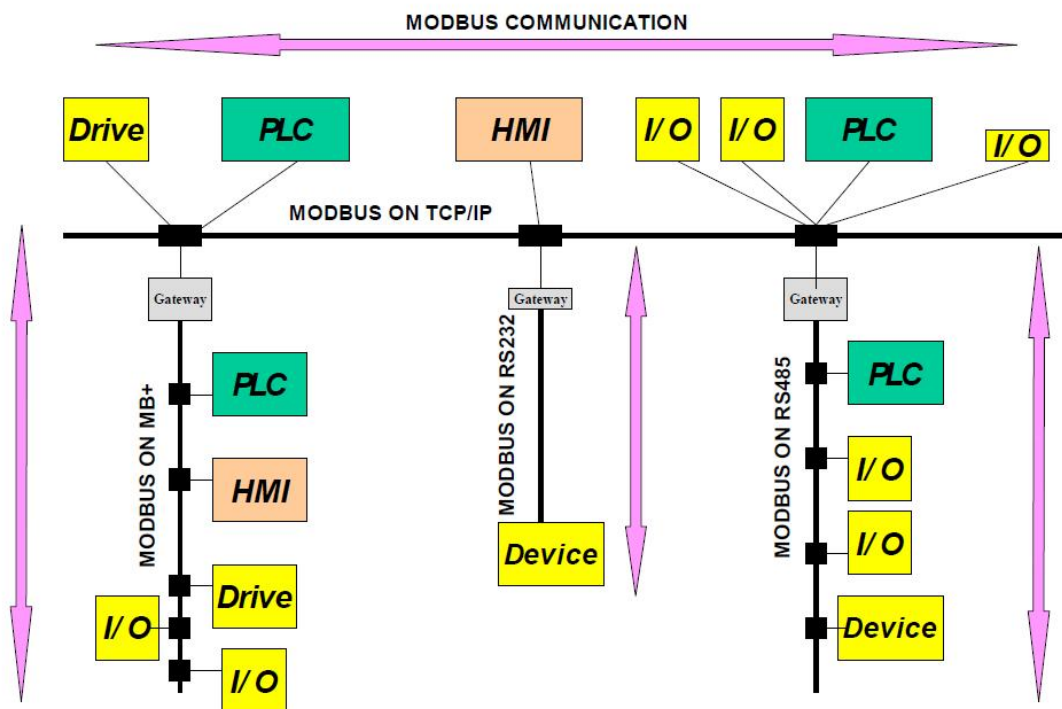


Figure 32: Modbus Network Architecture

The MODBUS protocol defines a simple protocol data unit (PDU) independent of the underlying communication layer. The mapping of the MODBUS protocol on a specific bus or network can introduce some additional fields on the Application Data Unit (ADU).

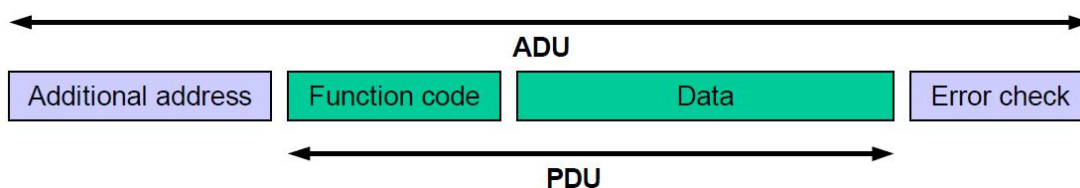


Figure 33: Modbus data frame

More details on the Modbus open protocol specification can be found on the website www.modbus.org.

6.1.1 Modbus TCP

The Modbus TCP protocol is a variant of the Modbus protocol that is optimized for communication over a TCP/IP connection. The protocol is designed for data exchange at the field level (ie for I/O data exchange in the process image). On the server side, all packets are sent over a TCP connection with port number 502.

The general Modbus TCP message is as follows:

| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 - n |
|------------|------------------------|---|---------------------|---|--------------|---|---------------|----------------------|-------|
| definition | transaction identifier | | protocol identifier | | field length | | slave address | Modbus function code | data |

6.1.2 Modbus data encoding

Modbus uses "big endian" representation for addresses and data items. This means that when transferring numbers larger than a single byte, the most significant byte is sent first.

6.1.3 Modbus data type

The modbus protocol is based on the following basic data types:

Tabel 22: Modbus basic data type

| type of data | object type | access type | describe |
|------------------|---------------|-------------|----------------|
| digital input | 1 bit | read | digital input |
| coil | 1 bit | read/write | digital output |
| input register | 16 bit (word) | read | analog input |
| holding register | 16 bit (word) | read/write | analog output |

For each basic data type, one or more function codes are defined. These function codes allow digital or analog input and output data, as well as internal variables to be set or read directly from the fieldbus node.

6.1.2 Modbus function code description

The function codes supported by the BL200 fieldbus node are shown in the table below. To perform the required functions, please specify the respective function codes and the

address of the selected input or output channel or register.

Tabel 23: Modbus function code list

| Modbus function code | function | access type | describe |
|----------------------|--------------------------|-------------|------------------|
| 0x02 | read digital input | read | Access by 1 bit |
| 0x01 | read coil | read/write | |
| 0x05 | write a single coil | read/write | |
| 0x0F | write multiple coils | read/write | |
| 0x04 | read input register | read | Access by 16 Bit |
| 0x03 | read multiple registers | read/write | |
| 0x06 | write a single register | read/write | |
| 0x10 | write multiple registers | read/write | |

The MODBUS function is performed as follows:

1. The MODBUS TCP master station (such as PC) sends a request to the BL200 fieldbus node using a specific function code;
2. The BL200 fieldbus node receives the data message, and then responds to the master with correct data according to the master's request.

If a fieldbus node receives an incorrect request, it sends an error data telegram (exception) to the master.

The meaning of the exception code contained in the exception is as follows:

Tabel 24: Modbus exception code

| Exception code | Describe |
|----------------|----------------------|
| 0x01 | illegal function |
| 0x02 | illegal data address |
| 0x03 | illegal data value |
| 0x04 | slave device failure |

6.1.2.1 Function code 0x02 (read digital input)

This function code is used to read the continuous state of single or multiple digital inputs.

1. request

The request specifies the starting address and the quantity to be read.

Tabel 25: Function code 0x02 - request message

| Field Name | Number of bytes | Example | Describe |
|-------------|-----------------|---------|--------------------------|
| Transaction | 2 Byte | 0x00 01 | Identification of Modbus |

| | | | |
|---------------------|--------|---------|--|
| identifier | | | request/response transactions |
| Protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| Message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| Function code | 1 Byte | 0x02 | Read digital input, use function code 0x02 |
| Start address | 2 Byte | 0x07 D0 | The address is detailed in the "Modbus Register Mapping" chapter |
| Enter quantity | 2 Byte | 0x08 | Read 8 digital inputs |

2. response

The data field indicates the value of the input state. A binary 1 corresponds to the on state and a 0 corresponds to the off state. The least significant bit (LSB) of the first data byte contains the first bit of the request, the others are in ascending order. If the response data is not a multiple of 8, the remaining bits of the last data byte will be padded with zeros (towards the upper bits of the byte).

Tabel 26: Function code 0x02-response message

| Field Name | Number of bytes | Example | Describe |
|------------------------|-----------------|---------|--|
| Transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| Protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| Message length | 2 Byte | 0x00 04 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| Function code | 1 Byte | 0x02 | Read digital input, use function code 0x02 |
| Data bytes | 1 Byte | 0x01 | number of bytes of data |
| Data | 1 Byte | 0x89 | response data |

3. abnormal

Tabel 27: Function code 0x02 - abnormal response

| Field Name | Number of bytes | Example | Describe |
|---------------|-----------------|---------|-----------------------------|
| ... | | | |
| Function code | 1 Byte | 0x82 | Modbus function code + 0x80 |

| | | | |
|---------------|--------|------|--------------|
| abnormal code | 1 Byte | 0x01 | 0x01 or 0x02 |
|---------------|--------|------|--------------|

4. Example

Read the value of 8 digital inputs from address 2000 to 2007.

request

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

Tabel 28: Function Code 0x02-Request Message-Example

| 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 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | device address | function code | stat address | | number of coils | |

response

0x00 01 00 00 00 04 01 02 01 89

Tabel 29: Function Code 0x02-Response Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|------------|------------------------|---|---------------------|---|----------------|---|----------------|---------------|------------|------|
| Data | 00 01 | | 00 00 | | 00 04 | | 01 | 01 | 01 | 89 |
| illustrate | transaction identifier | | protocol identifier | | message length | | device address | function code | data bytes | data |

Status from 2007 to 2000 is displayed as byte value 0x89 or binary 1000 1001. Address 2007 is the most significant bit MSB of the byte, 2000 is the least significant bit LSB, the distribution from high to low is as follows:

Tabel 30: digital input data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|-------|------|------|------|-------|------|------|-------|
| address | 2007 | 2006 | 2005 | 2004 | 2003 | 2002 | 2001 | 2000 |
| status | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| illustrate | close | open | open | open | close | open | open | close |

6.1.2.2 Function code 0x01 (read coil)

This function code is used to read the continuous status of single or multiple coils in the remote device.

1. request

The request specifies the starting address, which specifies the address of the first coil, and the number of coils.

Tabel 31: Function code 0x01 - request message

| Field Name | Number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| Transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| Protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| Message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| Function code | 1 Byte | 0x01 | Read coil, use function code 0x01 |
| Start address | 2 Byte | 0x03 E8 | The address is detailed in the "Modbus Register Mapping" chapter |
| Number of coils | 2 Byte | 0x00 08 | Read 8 coil states |

2. response

The data field indicates the value of the input state. A binary 1 corresponds to the on state and a 0 corresponds to the off state. The least significant bit (LSB) of the first data byte contains the first bit of the request, the others are in ascending order. If the response data is not a multiple of 8, the remaining bits of the last data byte will be padded with zeros (towards the upper bits of the byte).

Table 32: Function code 0x01-response message

| Field Name | Number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| Transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| Protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| Message length | 2 Byte | 0x00 04 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| Function code | 1 Byte | 0x01 | Read coil, use function code 0x01 |
| Data bytes | 1 Byte | 0x01 | number of bytes of data |
| Data | 1 Byte | 0x89 | response data |

3. abnormal

Table 33: Function code 0x01-abnormal

| Field Name | Number of bytes | Example | illustrate |
|------------|-----------------|---------|------------|
| ... | | | |

| | | | |
|---------------|--------|------|-----------------------------|
| function code | 1 Byte | 0x81 | Modbus function code + 0x80 |
| abnormal code | 1 Byte | 0x01 | 0x01 or 0x02 |

4. example

Read the status values of 8 coils from addresses 1000 to 1007.

request

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

Tabel 34: Function Code 0x01-Request Message-Example

| 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 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | initial address | | Number of coils | |

response

0x00 01 00 00 00 04 01 01 01 89

Tabel 35: Function code 0x01-response message

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|------------|------------------------|---|---------------------|---|----------------|---|----------------|---------------|------------|------|
| Data | 00 01 | | 00 00 | | 00 04 | | 01 | 01 | 01 | 89 |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | data bytes | data |

Status from 1007 to 1000 is displayed as byte value 0x89 or binary 1000 1001. Address 1007 is the most significant bit MSB of the byte, 1000 is the least significant bit LSB, the distribution from high to low is as follows:

Tabel 36: Coil data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|-------|------|------|------|-------|------|------|-------|
| address | 1007 | 1006 | 1005 | 1004 | 1003 | 1002 | 1001 | 1000 |
| status | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| illustrate | close | open | open | open | close | open | open | close |

6.1.2.3 Function code 0x05 (write a single coil)

This function will write a single coil status to the slave device.

1. request

Tabel 37: Function code 0x05 - request message

| Field Name | Number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x05 | To write a single coil, use function code 0x05 |
| register address | 2 Byte | 0x03 E8 | The address is detailed in the "Modbus Register Mapping" chapter |
| data input | 2 Byte | 0xFF 00 | This value is: 0xFF 00 or 0x00 00. 0xFF 00 means write 1, 0x00 00 means write 0. |

2. response

Tabel 38: Function code 0x05-response message

| Field Name | Number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x05 | To write a single coil, use function code 0x05 |
| data bytes | 2 Byte | 0x03 E8 | Write the register address of the coil |
| data input | 2 Byte | 0xFF 00 | This value is: 0xFF 00 or 0x00 00. 0xFF 00 means write 1, 0x00 00 means write 0. |

3. abnormal

Tabel 39: Function code 0x05-abnormal

| Field Name | Number of bytes | Example | illustrate |
|------------|-----------------|---------|------------|
| ... | | | |

| | | | |
|---------------|--------|------|-----------------------------|
| function code | 1 Byte | 0x85 | Modbus function code + 0x80 |
| abnormal code | 1 Byte | 0x81 | 0x01 or 0x02 |

4. example

Write the state value of the coil at address 1000 as 1, that is, the closed state.

request

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

Tabel 40: Function Code 0x05-Request Message-Example

| 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 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | Coil address | | write "1" | |

response

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

Tabel 41: Function Code 0x05-Response Message-Example

| 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 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | Coil address | | write "1" | |

6.1.2.4 Function code 0x0F (write multiple coils)

This function code is used to set multiple consecutive coils to open or close. The on/off state of the request is specified by the content of the request data field. A logical "1" requests the corresponding output to close, and a logical "0" requests it to open. The normal response returns the function code, the starting address and the number of coils executed.

1. request

Tabel 42: Function code 0x0f - request message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 08 | The number of bytes of the following data |

| | | | |
|-----------------|--------|---------|--|
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x0F | Write multiple coils, use function code 0x0F |
| start address | 2 Byte | 0x03 E8 | The address is detailed in the "Modbus Register Mapping" chapter |
| Number of coils | 2 Byte | 0x00 08 | |
| data bytes | 1 Byte | 0x01 | |
| data | 1 Byte | 0xFF | |

2. response

Tabel 43: Function code 0x0f - response message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 00 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x0F | Write multiple coils, use function code 0x0F |
| start address | 2 Byte | 0x03 E8 | |
| Number of coils | 2 Byte | 0x00 08 | |

3. abnormal

Tabel 44: Function code 0x0f-abnormal

| Field Name | number of bytes | Example | illustrate |
|---------------|-----------------|---------|-----------------------------|
| ... | | | |
| function code | 1 Byte | 0x8F | Modbus function code + 0x80 |
| abnormal code | 1 Byte | | 0x01 or 0x02 |

4. example

Starting from address 1000, close all 8 coils, that is, write the value of 8 coils as 0xFF.
request

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

Tabel 45: Function code 0x0f-request message-example

| 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 |

| | | | | | | | | | |
|------------|------------------------|---------------------|----------------|----------------|---------------|---------------|-----------------|------------|------|
| illustrate | transaction identifier | protocol identifier | message length | Device address | function code | start address | Number of coils | data bytes | data |
|------------|------------------------|---------------------|----------------|----------------|---------------|---------------|-----------------|------------|------|

response

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

Tabel 46: Function code 0x0f-response message-example

| 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 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | start address | | Number of coils | |

6.1.2.5 Function code 0x04 (read input register)

This function code is used to read consecutive input registers in multiple remote devices. The request PDU specifies the address of the starting register and the number of registers. The register data in the response message is packed into two bytes per register, and the binary content within each byte is right-aligned.

- request

Tabel 47: Function code 0x04 - request message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x04 | Read input register, use function code 0x04 |
| start address | 2 Byte | 0x0B B8 | The address is detailed in the "Modbus Register Mapping" chapter |
| Number of registers | 2 Byte | 0x00 08 | |

2. response

Tabel 48: Function code 0x04-response message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|--|--|
| transaction identifier | 2 Byte | 0x00 00 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 13 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x04 | Read input register, use function code 0x04 |
| data bytes | 1 Byte | 0x10 | |
| data | 16 Byte | 0x 3F 8E 38 86 40 0E 38 86 40 55 54 CA 40 8E 35 3F | |

5. abnormal

Tabel 49: Function code 0x04-abnormal

| Field Name | number of bytes | Example | illustrate |
|---------------|-----------------|---------|-----------------------------|
| ... | | | |
| function code | 1 Byte | 0x84 | Modbus function code + 0x80 |
| abnormal code | 1 Byte | 0x01 | 0x01 or 0x02 |

6. example

Starting at address 3000, read the values of the 4 analog inputs. Since the BL200 controller node register map data type is 32Bit Float, that is, 1 analog input data = 2 registers = 4 bytes, 8 input registers need to be read.

request

0x00 01 00 00 00 06 01 04 0B B8 00 08

Tabel 50: Function Code 0x04-Request Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|------|-------------|----------|---------|--------|----------|-------|-----------|---|---|----|----|----|
| Data | 00 01 | 00 00 | 00 06 | 01 | 04 | 0B B8 | 00 08 | | | | | |
| illu | transaction | protocol | message | Device | function | start | Number of | | | | | |

| | | | | | | | |
|------------|------------|------------|--------|---------|------|---------|-----------|
| stra te | identifier | identifier | length | address | code | address | registers |
|------------|------------|------------|--------|---------|------|---------|-----------|

response

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

Table 51: Function Code 0x04-Response Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10...25 |
|--------------------|---------------------------|---|------------------------|---|-------------------|---|-------------------|------------------|------------|---------|
| Data | 00 01 | | 00 00 | | 00 13 | | 01 | 04 | 10 | xxx |
| illu stra te | transaction identifier | | protocol identifier | | message length | | Device address | function code | data bytes | data |

The data part has a total of 16 bytes, which are converted into decimal as follows

Table 52: read input register - convert data to decimal

| 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 | | | |
| decimal | 1.23 | | | | 2.34 | | | | 3.45 | | | | 4.56 | | | |
| illustrate | first data | | | | second data | | | | third data | | | | fourth data | | | |

6.1.2.6 Function code 0x03 (read holding register)

This function code is used to read continuous holding registers in multiple remote devices. The request PDU specifies the address of the starting register and the number of registers. The register data in the response message is packed into two bytes per register, and the binary content within each byte is right-aligned.

1. request

Table 53: Function code 0x03 - request message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x03 | Read holding register, use function |

| | | | |
|---------------------|--------|---------|--|
| | | | code 0x03 |
| start address | 2 Byte | 0x0F A0 | The address is detailed in the "Modbus Register Mapping" chapter |
| Number of registers | 2 Byte | 0x00 08 | Number of holding registers to read |

2. response

Tabel 54: Function code 0x03-response message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|--|--|
| transaction identifier | 2 Byte | 0x00 00 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 13 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x03 | Read holding register, use function code 0x03 |
| data bytes | 1 Byte | 0x10 | data bytes |
| data | 16 Byte | 0x 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85 | response data |

3. abnormal

Tabel 55: Function code 0x03-abnormal

| Field Name | number of bytes | Example | illustrate |
|---------------|-----------------|---------|-----------------------------|
| ... | | | |
| function code | 1 Byte | 0x83 | Modbus function code + 0x80 |
| abnormal code | 1 Byte | 0x01 | 0x01 or 0x02 |

4. example

Starting at address 4000, read the values of the 4 analog outputs (belonging to the holding registers). Since the analog output I/O module register map data type is 32Bit Float, that is, 1 analog output data = 2 registers = 4 bytes, it is necessary to read 8 holding registers.

request

0x00 01 00 00 00 06 01 03 0F A0 00 08

Table 56: Function Code 0x03-Request Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|------------|------------------------|---|---------------------|---|----------------|---|----------------|---------------|---------------|----|---------------------|----|
| Data | 00 01 | | 00 00 | | 00 06 | | 01 | 03 | 0F A0 | | 00 08 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | start address | | Number of registers | |

response

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

Table 57: Function Code 0x03-Response Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10...25 |
|------------|------------------------|---|---------------------|---|----------------|---|----------------|---------------|------------|---------|
| Data | 00 01 | | 00 00 | | 00 13 | | 01 | 03 | 10 | xxx |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | data bytes | data |

The data part has a total of 16 bytes, and the conversion to decimal is as follows:

Table 58: Read Holding Registers - Convert Data Decimal

| 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 | | | |
| decimal | 1.23 | | | | 2.34 | | | | 3.45 | | | | 4.56 | | | |
| illustrate | first data | | | | second data | | | | third data | | | | fourth data | | | |

6.1.2.7 Function code 0x06 (write a single register)

This function code is used to write to holding registers in a single remote device. The request PDU specifies the address of the starting register and the number of registers. The register data in the response message is packed into two bytes per register, and the binary content within each byte is right-aligned.

This function code is only suitable for reading the serial port I/O module register mapping data, the address range: 40000 ... 49999. The data type of the analog input/output I/O module is 32Bit Float format, the complete data cannot be read, and this function cannot be used.

1. request

Table 59: Function code 0x06 - request message

| Field Name | number of | Example | illustrate |
|------------|-----------|---------|------------|
|------------|-----------|---------|------------|

| | bytes | | |
|------------------------|--------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x06 | Write a single holding register, use function code 0x06 |
| register address | 2 Byte | 0x9C 40 | The address is detailed in the "Modbus Register Mapping" chapter |
| data | 2 Byte | 0x04 D2 | |

2. response

Tabel 60: Function code 0x06-response message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|---|
| transaction identifier | 2 Byte | 0x00 00 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 06 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x06 | Write a single holding register, use function code 0x06 |
| Register address | 2 Byte | 0x75 30 | |
| data | 2 Byte | 0x04 D2 | |

3. abnormal

Tabel 61: Function code 0x06-abnormal

| Field Name | number of bytes | Example | illustrate |
|---------------|-----------------|---------|-----------------------------|
| ... | | | |
| function code | 1 Byte | 0x86 | Modbus function code + 0x80 |
| abnormal code | 1 Byte | 0x01 | 0x01 or 0x02 |

4. example

Write the value of register address 40000 to 1234 (0x04 D2).

request

0x00 01 00 00 00 06 01 06 9C 40 04 D2

Tabel 62: Function Code 0x06-Request Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|------------|------------------------|---|---------------------|---|----------------|---|----------------|---------------|------------------|----|-------|----|
| Data | 00 01 | | 00 00 | | 00 06 | | 01 | 06 | 9C 40 | | 04 D2 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | register address | | Data | |

response

0x00 01 00 00 00 06 01 06 9C 40 04 D2

Tabel 63: Function Code 0x06-Response Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|------------|------------------------|---|---------------------|---|----------------|---|----------------|---------------|------------------|----|-------|----|
| Data | 00 01 | | 00 00 | | 00 06 | | 01 | 0F | 9C 40 | | 04 D2 | |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | register address | | Data | |

6.1.2.8 Function code 0x10 (write multiple registers)

This function code is used to write to consecutive holding registers in multiple remote devices. The request PDU specifies the address of the starting register and the number of registers. The register data in the response message is packed into two bytes per register, and the binary content within each byte is right-aligned.

1. request

Tabel 64: Function code 0x10 - request message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 01 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 17 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x10 | Write multiple holding registers, use function code 0x10 |
| start address | 2 Byte | 0x0F A0 | The address is detailed in the "Modbus Register Mapping" chapter |

| | | | |
|---------------------|---------|--|--|
| Number of registers | 2 Byte | 0x00 08 | |
| data bytes | 1 Byte | 0x10 | |
| data | 16 Byte | 0x 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85 | |

2. response

Tabel 65: Function code 0x10-response message

| Field Name | number of bytes | Example | illustrate |
|------------------------|-----------------|---------|--|
| transaction identifier | 2 Byte | 0x00 00 | Identification of Modbus request/response transactions |
| protocol identifier | 2 Byte | 0x00 00 | 0x00 00: Modbus protocol |
| message length | 2 Byte | 0x00 13 | The number of bytes of the following data |
| Device address | 1 Byte | 0x01 | Slave address identification |
| function code | 1 Byte | 0x10 | Write multiple holding registers, use function code 0x10 |
| start address | 2 Byte | 0x0F A0 | |
| Number of registers | 2 Byte | 0x00 08 | |

3. abnormal

Tabel 66: Function code 0x10-abnormal

| Field Name | number of bytes | Example | illustrate |
|---------------|-----------------|---------|-----------------------------|
| ... | | | |
| function code | 1 Byte | 0x90 | Modbus function code + 0x80 |
| abnormal code | 1 Byte | 0x01 | 0x01 or 0x02 |

4. example

Starting at address 4000, write the values of the 4 analog outputs. Since the BL200 controller node register map data type is 32Bit Float, that is, 1 analog output data = 2 holding registers = 4 bytes, 8 holding registers need to be written.

request

0x00 01 00 00 00 17 01 10 0F A0 00 08 10 3F 9D 70 A4 40 15 C2 8F 40 5C CC CD 40 91 EB 85

Table 67: Function code 0x10-request message-example

| 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 | 0F | A0 | 00 | 08 | 10 | xxx |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | start address | | Number of registers | | data bytes | Data |

The data part has a total of 16 bytes, and the conversion to decimal is as follows:

Table 68: Write Holding Registers - Convert Data Decimal

| Byte | 14 | | | | | | | | | | | | | | | |
|------------|-------------|--|--|--|-------------|--|--|--|-------------|--|--|--|-------------|--|--|--|
| Data | 3F 9D 70 A4 | | | | 40 15 C2 8F | | | | 40 5C CC CD | | | | 40 91 EB 85 | | | |
| decimal | 1.23 | | | | 2.34 | | | | 3.45 | | | | 4.56 | | | |
| illustrate | first data | | | | second data | | | | third data | | | | fourth data | | | |

response

0x00 01 00 00 00 06 01 10 0F A0 00 08

Table 69: Function Code 0x10-Response Message-Example

| Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|------------|------------------------|----|---------------------|----|----------------|----|----------------|---------------|---------------|----|---------------------|----|
| Data | 00 | 01 | 00 | 00 | 00 | 06 | 01 | 10 | 0F | A0 | 00 | 08 |
| illustrate | transaction identifier | | protocol identifier | | message length | | Device address | function code | start address | | Number of registers | |

6.1.3 Modbus register mapping address

The internal register map of BL200 field controller node consists of 2 parts, one part is the data map of digital input and output and analog input and output module, the address range is 1000...9999; the other part is the serial port module, the address range is 10000...49999

The state of digital and analog I/O modules can be determined or changed through the register map (addresses 1000 ... 9999).

Table 70: Modbus Register Mapping address - I/O Modules

| Modbus address | | type of data | access type | function code | describe |
|----------------|-------------------|--------------|-------------|---------------|-------------------|
| decimal | hex | | | | |
| 1000...1999 | 0x03 E8...0x07 CF | 1 Bit | read/write | 0x01/05/0F | Digital output DO |

| | | | | | |
|-------------|-------------------|--------------|------------|------------|------------------------|
| 2000...2999 | 0x07 D0...0x0B B7 | 1 Bit | read | 0x02 | Digital input DI |
| 3000...3999 | 0x0B B8...0x0F 9F | 32 Bit Float | read | 0x04 | Analog input AI |
| 4000...4999 | 0x0F A0...0X13 87 | 32 Bit Float | read/write | 0x03/06/10 | Analog output AO |
| 5000...8999 | 0x13 88...0x23 27 | 32 Bit Unint | read/write | 0x03/04/10 | DI count value |
| 9000...9999 | 0x23 28...0x27 0F | 1 Bit | read | 0x02 | Module power-on status |

And through addresses 10000 ... 49999 it is possible to determine or change the state of the data mapped from the serial I/O module.

Tabel 71: Modbus Register Mapping address - Serial Port Module

| Modbus address | | type of data | access type | function code | describe |
|----------------|-------------------|--------------|-------------|---------------|-------------------|
| decimal | hex | | | | |
| 10000...19999 | 0x27 10...0x4E 1F | 1 Bit | read/write | 0x01/05/0F | Digital output DO |
| 20000...29999 | 0x4E 20...0x75 2F | 1 Bit | read | 0x02 | Digital input DI |
| 30000...39999 | 0x75 30...0x9C 3F | 16 Bit | read | 0x04 | Analog input AI |
| 40000...49999 | 0x9C 40...0XC3 4F | 16 Bit | read/write | 0x03/06/10 | Analog output AO |

6.2 OPC UA

6.2.1 Overview

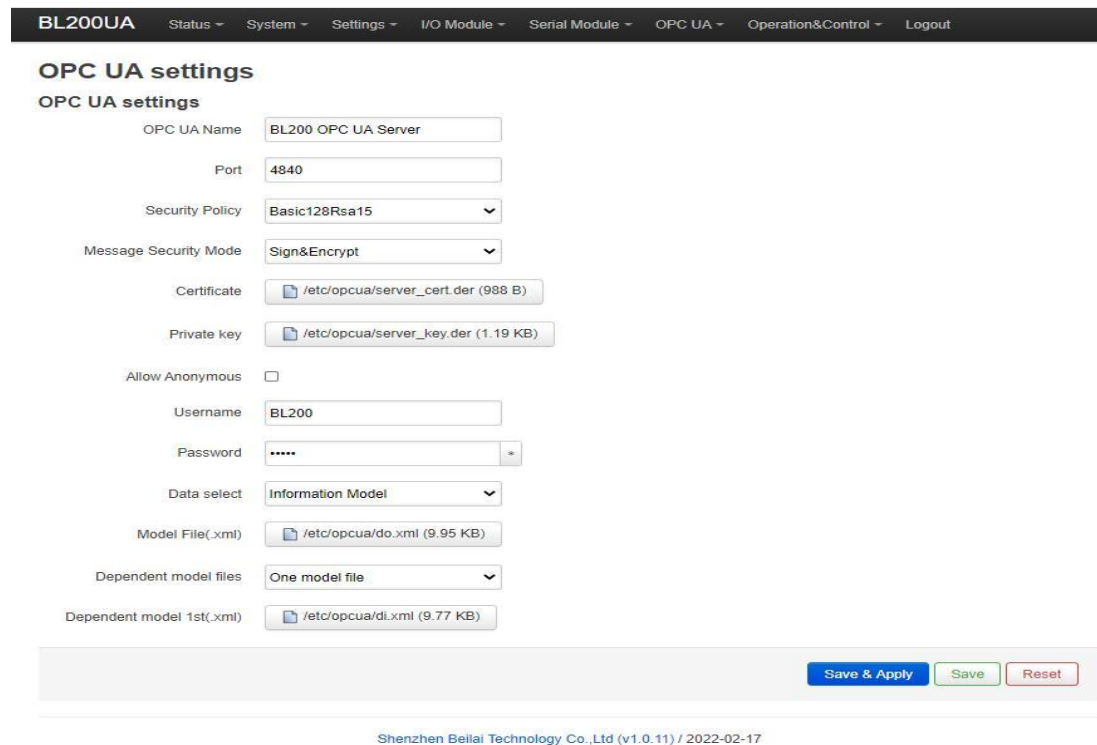
BL200 series distributed I/O system supports OPC UA Server function and provides external data in the form of server. Compliant with IEC 62541 industrial automation unified architecture communication standard, data can be transmitted by encryption (X.509 certificate) and authentication. The security policy supports basic128rsa15, basic256, basic256sha256, aes128sha256rsaoaep, optional signature or signature and encryption. Support custom information model function, you can fill in up to 5 reference models.

6.2.2 Application example

Take the collection of DI, DO, and AI modules, select basic128rsa15 for the security policy, select the signature and encryption method, and use the custom information model for the data format. Refer to an information model as an example. Data can also be

uploaded directly in the company's format. For the definition of each configuration, please refer to the introduction in chapter 5.7 [OPC UA](#).

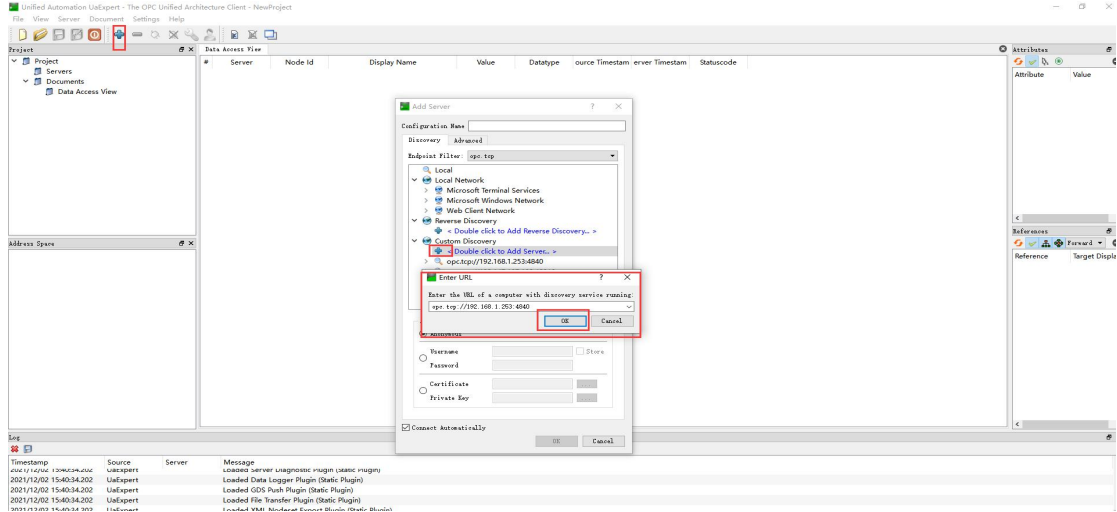
6.2.2.1 OPC UA web configuration



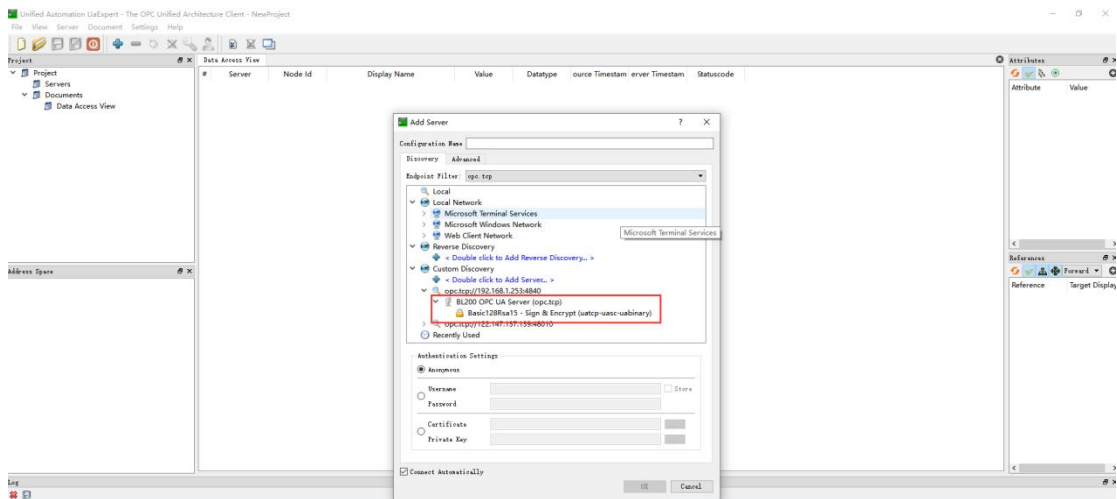
Steps: (1) Fill in the OPC UA name, which can be customized to facilitate the OPC UA client to search and distinguish different OPC UA servers. For example: fill in "BL200 OPC UA Server". (2) The port number of the OPC UA server, default: 4840. (3) Security policy selection. For example, choose basic128rsa15. (4) Message security mode selection. For example, choose Signing and Encryption. (5) Upload the certificate and key, click "Select File" > click "Upload File" > select your certificate or key file, click Open > After it is displayed in the file name box, click Upload file > After uploading the file successfully The file you uploaded will be displayed in the box, click the certificate or key file you uploaded > then your certificate or key file will be displayed in the certificate or key item. (6) Whether to allow anonymity, because of the use of signature and encryption methods, allow anonymity is not checked. (7) Fill in the username and password. The client needs to fill in the username and password when connecting. (8) Select the data, because the user-defined information model is used, so choose the "information model". (9) Upload the information model file. The upload method is the same as uploading the certificate or key file. The uploaded file is an xml file. (10) Depends on the model file, whether there is a reference model, and how many references are there. (11) Dependent model: Upload the model you refer to. The upload method is the same as uploading the certificate or key file. The upload is an xml file. (12) Click "Save and Apply".

6.2.2.2 Send and receive data using UaExpert client connection

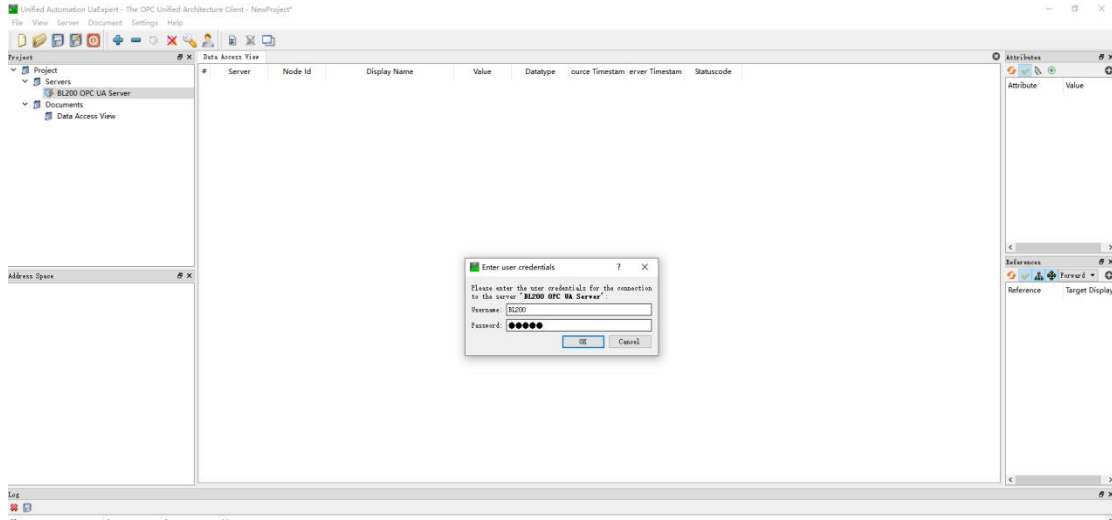
Open UaExpert (OPC UA client) and enter the OPC UA server IP and port.



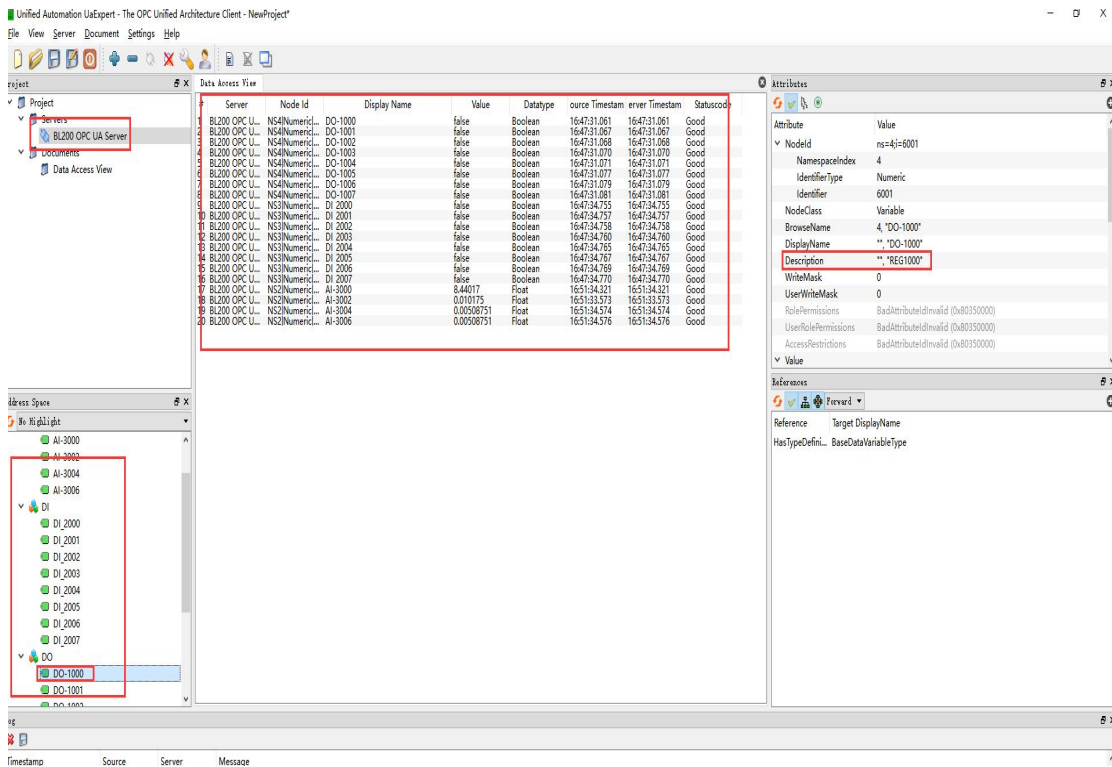
Click Search, click the searched OPC UA server, and click basic128rsa15 for Signature and Encryption.



Enter the set username and password



The collected data is as follows:



The description item of the custom information model data point must be REG+Modbus address, as shown in the description of the DO-1000 point in the figure above.

OPC UA client data delivery

Take the following data point DO-1000 as an example

IO status

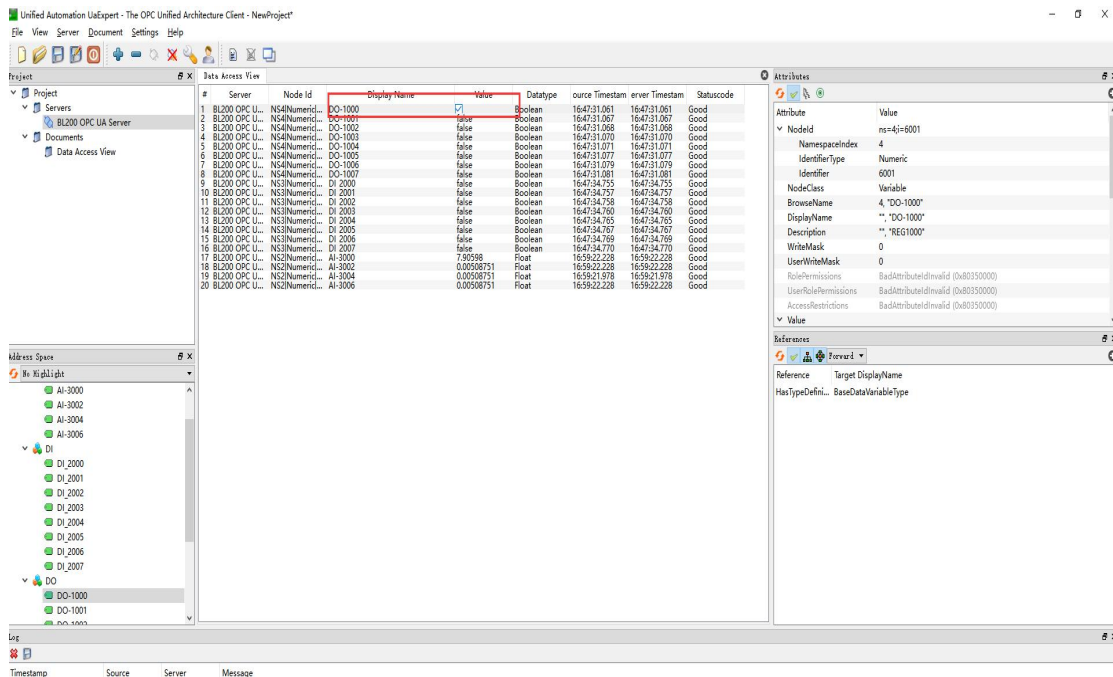
IO Slot:2,Module Type:DO,Module Name:M2082

| Channels | Modbus Address | Value | PowerOn Status | Open/Close |
|----------|----------------|-------|----------------|------------|
| 1 | 1000 | Open | Open | Open/Close |
| 2 | 1001 | Open | Open | Open/Close |
| 3 | 1002 | Open | Open | Open/Close |
| 4 | 1003 | Open | Open | Open/Close |
| 5 | 1004 | Open | Open | Open/Close |
| 6 | 1005 | Open | Open | Open/Close |
| 7 | 1006 | Open | Open | Open/Close |
| 8 | 1007 | Open | Open | Open/Close |

Back to Overview Save & Apply Save Reset

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

Click the value of the DO-1000 data point, it turned out to be false, there is no √ in the square, click once to put √, click the left mouse button in the blank space or press the [Enter] key on the keyboard



The OPC UA client will send a message successfully. Because the server responds quickly, you can see that the value has changed to "true".

The screenshot shows the UaExpert interface with a table of data points and a log window. The table has columns for #, Server, Node Id, Display Name, Value, Datatype, Source Timestamp, Server Timestamp, and Statuscode. Row 1 shows Node Id NS4:NumericId:DO-1000 with a Value of true. The log window shows a message: "2021/12/02 16:59:23.065 DA Plugin BL200 OPC U... Write to node 'NS4:NumericId:6001' succeeded (ret = Good (0x00000001))".

| # | Server | Node Id | Display Name | Value | Datatype | Source Timestamp | Server Timestamp | Statuscode |
|----|----------------|-----------------------|--------------|----------|----------|------------------|------------------|------------|
| 1 | BL200 OPC U... | NS4:NumericId:DO-1000 | DO-1000 | true | Boolean | 16:59:22.729 | 16:59:22.729 | Good |
| 2 | BL200 OPC U... | NS4:NumericId:DO-1001 | DO-1001 | false | Boolean | 16:47:31.067 | 16:47:31.067 | Good |
| 3 | BL200 OPC U... | NS4:NumericId:DO-1002 | DO-1002 | false | Boolean | 16:47:31.068 | 16:47:31.068 | Good |
| 4 | BL200 OPC U... | NS4:NumericId:DO-1003 | DO-1003 | false | Boolean | 16:47:31.070 | 16:47:31.070 | Good |
| 5 | BL200 OPC U... | NS4:NumericId:DO-1004 | DO-1004 | false | Boolean | 16:47:31.071 | 16:47:31.071 | Good |
| 6 | BL200 OPC U... | NS4:NumericId:DO-1005 | DO-1005 | false | Boolean | 16:47:31.077 | 16:47:31.077 | Good |
| 7 | BL200 OPC U... | NS4:NumericId:DO-1006 | DO-1006 | false | Boolean | 16:47:31.079 | 16:47:31.079 | Good |
| 8 | BL200 OPC U... | NS4:NumericId:DO-1007 | DO-1007 | false | Boolean | 16:47:31.081 | 16:47:31.081 | Good |
| 9 | BL200 OPC U... | NS3:NumericId:DI-2000 | DI-2000 | false | Boolean | 16:47:34.755 | 16:47:34.755 | Good |
| 10 | BL200 OPC U... | NS3:NumericId:DI-2001 | DI-2001 | false | Boolean | 16:47:34.757 | 16:47:34.757 | Good |
| 11 | BL200 OPC U... | NS3:NumericId:DI-2002 | DI-2002 | false | Boolean | 16:47:34.758 | 16:47:34.758 | Good |
| 12 | BL200 OPC U... | NS3:NumericId:DI-2003 | DI-2003 | false | Boolean | 16:47:34.760 | 16:47:34.760 | Good |
| 13 | BL200 OPC U... | NS3:NumericId:DI-2004 | DI-2004 | false | Boolean | 16:47:34.763 | 16:47:34.763 | Good |
| 14 | BL200 OPC U... | NS3:NumericId:DI-2005 | DI-2005 | false | Boolean | 16:47:34.767 | 16:47:34.767 | Good |
| 15 | BL200 OPC U... | NS3:NumericId:DI-2006 | DI-2006 | false | Boolean | 16:47:34.769 | 16:47:34.769 | Good |
| 16 | BL200 OPC U... | NS3:NumericId:DI-2007 | DI-2007 | false | Boolean | 16:47:34.770 | 16:47:34.770 | Good |
| 17 | BL200 OPC U... | NS2:NumericId:AI-3000 | AI-3000 | 7.89072 | Float | 17:00:05.231 | 17:00:05.231 | Good |
| 18 | BL200 OPC U... | NS2:NumericId:AI-3002 | AI-3002 | 0.010175 | Float | 17:00:04.931 | 17:00:04.931 | Good |
| 19 | BL200 OPC U... | NS2:NumericId:AI-3004 | AI-3004 | 0.010175 | Float | 17:00:04.981 | 17:00:04.981 | Good |
| 20 | BL200 OPC U... | NS2:NumericId:AI-3006 | AI-3006 | 0.010175 | Float | 17:00:05.231 | 17:00:05.231 | Good |

Check the DO status in the web configuration of BL200. DO1 is also changed from the original open to close.

The screenshot shows the BL200UA web configuration interface. The top navigation bar includes: BL200UA, Status, System, Settings, I/O Module, Serial Module, OPC UA, Operation&Control, and Logout. The main content area is titled "IO status" and contains a table with columns: IO Slot, Module Name, Module Type, Channel Number, Modbus Address, 24V Address-State, Soft Version, IO Status, and Channel Status. Row 2 is highlighted with a red box, showing IO Slot 2, Module Name M2082, Module Type DO, Channel Number 8, Modbus Address 1000-1007, 24V Address-State 9002-Power Off, Soft Version 5, IO Status Normal, and Channel Status.

| IO Slot | Module Name | Module Type | Channel Number | Modbus Address | 24V Address-State | Soft Version | IO Status | Channel Status |
|---------|-------------|-------------|----------------|----------------|-------------------|--------------|-----------|----------------|
| 1 | M1081 | DI | 8 | 2000-2007 | 9001-Power Off | 5 | Normal | Channel Status |
| 2 | M2082 | DO | 8 | 1000-1007 | 9002-Power Off | 5 | Normal | Channel Status |
| 3 | M3041 | AI | 4 | 3000-3006 | 9003-Power Off | 5 | Normal | Channel Status |
| 4 | M4044 | AO | 4 | 4000-4006 | 9004-Power Off | 5 | Normal | Channel Status |
| 5 | M6021 | COM | 2 | 0-0 | 9005-Power Off | 5 | Normal | Channel Status |

BL200UA
Status ▾
System ▾
Settings ▾
I/O Module ▾
Serial Module ▾
OPC UA ▾
Operation&Control ▾
Logout

IO status

IO Slot:2,Module Type:DO,Module Name:M2082

| Channels | Modbus Address | Value | PowerOn Status | Open/Close |
|----------|----------------|-------|----------------|------------|
| 1 | 1000 | Close | Open ▾ | Open/Close |
| 2 | 1001 | Open | Open ▾ | Open/Close |
| 3 | 1002 | Open | Open ▾ | Open/Close |
| 4 | 1003 | Open | Open ▾ | Open/Close |
| 5 | 1004 | Open | Open ▾ | Open/Close |
| 6 | 1005 | Open | Open ▾ | Open/Close |
| 7 | 1006 | Open | Open ▾ | Open/Close |
| 8 | 1007 | Open | Open ▾ | Open/Close |

Back to Overview
Save & Apply ▾
Save
Reset

Shenzhen Beilai Technology Co.,Ltd (v1.0.11) / 2022-02-17

7. Appendix

7.1 List of Figures

| | |
|---|----|
| <i>Figure 1: Fieldbus Node</i> | 6 |
| <i>Figure 2: view</i> | 10 |
| <i>Figure 3: 2D schematic</i> | 11 |
| <i>Figure 4: data contacts</i> | 12 |
| <i>Figure 5: power jumper contacts</i> | 12 |
| <i>Figure 6: terminal point</i> | 13 |
| <i>Figure 7: controller LED Indicators</i> | 14 |
| <i>Figure 8: Power Module LED Indicators</i> | 14 |
| <i>Figure 9: Ethernet interface</i> | 15 |
| <i>Figure 10: IP address selector switch (eg: set"0")</i> | 15 |
| <i>Figure 11: factory reset button</i> | 16 |
| <i>Figure 12: schematic block diagram</i> | 16 |
| <i>Figure 13: locking controller</i> | 17 |

| | |
|--|----|
| Figure 14: unlock the controller..... | 18 |
| Figure 15: unlock the controller..... | 18 |
| Figure 16: align the groove (example)..... | 19 |
| Figure 17: snap the I/O module into place (example)..... | 19 |
| Figure 18: remove I/O module (example)..... | 20 |
| Figure 19: connecting wire..... | 20 |
| Figure 20: schematic diagram of connecting the system power supply..... | 21 |
| Figure 21: schematic diagram of connecting field power supply..... | 22 |
| Figure 22: DIN rail contacts..... | 23 |
| Figure 23: connect the bus to the network..... | 24 |
| Figure 24: connect the bus to the computer..... | 24 |
| Figure 25: Network and Sharing Center..... | 26 |
| Figure 26: local connection status..... | 27 |
| Figure 27: local connection properties..... | 27 |
| Figure 28: obtain an IP address automatically..... | 28 |
| Figure 29: Set static IP..... | 28 |
| Figure 30: web page settings IP..... | 29 |
| Figure 31: dip switch - assigned via dip selector switch (Example: 192.168.1.253)..... | 30 |
| Figure 32: Modbus Network Architecture..... | 50 |
| Figure 33: Modbus data frame..... | 50 |

7.2 List of Tables

| | |
|--|----|
| Table 1: technical parameter..... | 8 |
| Table 2: model selection..... | 9 |
| Tabel 3: "power jumper contacts" describe..... | 13 |
| Tabel 4: "terminal point" describe..... | 13 |
| Tabel 5: "controller LED Indicators" describe..... | 14 |
| Tabel 6: "Power Module LED Indicators" describe..... | 14 |
| Tabel 7: DIP switch position definition..... | 29 |
| Tabel 8: factory default parameters..... | 30 |
| Tabel 9: System > System Properties > General Settings..... | 34 |
| Tabel 10: System > System Properties > Logging..... | 35 |
| Tabel 11: System > System Properties > Language and Style..... | 36 |
| Tabel 12: System > Backup/ Flash Firmware > Actions..... | 38 |
| Tabel 13: Settings > Device settings..... | 39 |
| Tabel 14: I/O Module > I/O Status..... | 40 |
| Tabel 15: Digital Input Modules > IO Status..... | 41 |
| Tabel 16: Digital Input Modules > DI Count..... | 42 |
| Tabel 17: Digital output module..... | 42 |
| Tabel 18: Analog input module..... | 43 |
| Tabel 19: Analog output module..... | 44 |

| | |
|--|----|
| <i>Tabel 20: Modbus master</i> | 46 |
| <i>Tabel 21: OPC UA settings</i> | 48 |
| <i>Tabel 22: Modbus basic data type</i> | 51 |
| <i>Tabel 23: Modbus function code list</i> | 52 |
| <i>Tabel 24: Modbus exception code</i> | 52 |
| <i>Tabel 25: Function code 0x02 - request message</i> | 52 |
| <i>Tabel 26: Function code 0x02-response message</i> | 53 |
| <i>Tabel 27: Function code 0x02 - abnormal response</i> | 53 |
| <i>Tabel 28: Function Code 0x02-Request Message-Example</i> | 54 |
| <i>Tabel 29: Function Code 0x02-Response Message-Example</i> | 54 |
| <i>Tabel 30: digital input data</i> | 54 |
| <i>Tabel 31: Function code 0x01 - request message</i> | 54 |
| <i>Tabel 32: Function code 0x01-response message</i> | 55 |
| <i>Tabel 33: Function code 0x01-abnormal</i> | 55 |
| <i>Tabel 34: Function Code 0x01-Request Message-Example</i> | 56 |
| <i>Tabel 35: Function code 0x01-response message</i> | 56 |
| <i>Tabel 36: Coil data</i> | 56 |
| <i>Tabel 37: Function code 0x05 - request message</i> | 57 |
| <i>Tabel 38: Function code 0x05-response message</i> | 57 |
| <i>Tabel 39: Function code 0x05-abnormal</i> | 57 |
| <i>Tabel 40: Function Code 0x05-Request Message-Example</i> | 58 |
| <i>Tabel 41: Function Code 0x05-Response Message-Example</i> | 58 |
| <i>Tabel 42: Function code 0x0f - request message</i> | 58 |
| <i>Tabel 43: Function code 0x0f - response message</i> | 59 |
| <i>Tabel 44: Function code 0x0f-abnormal</i> | 59 |
| <i>Tabel 45: Function code 0x0f-request message-example</i> | 59 |
| <i>Tabel 46: Function code 0x0f-response message-example</i> | 60 |
| <i>Tabel 47: Function code 0x04 - request message</i> | 60 |
| <i>Tabel 48: Function code 0x04-response message</i> | 61 |
| <i>Tabel 49: Function code 0x04-abnormal</i> | 61 |
| <i>Tabel 50: Function Code 0x04-Request Message-Example</i> | 61 |
| <i>Tabel 51: Function Code 0x04-Response Message-Example</i> | 62 |
| <i>Tabel 52: read input register - convert data to decimal</i> | 62 |
| <i>Tabel 53: Function code 0x03 - request message</i> | 62 |
| <i>Tabel 54: Function code 0x03-response message</i> | 63 |
| <i>Tabel 55: Function code 0x03-abnormal</i> | 63 |
| <i>Tabel 56: Function Code 0x03-Request Message-Example</i> | 64 |
| <i>Tabel 57: Function Code 0x03-Response Message-Example</i> | 64 |
| <i>Tabel 58: Read Holding Registers - Convert Data Decimal</i> | 64 |
| <i>Tabel 59: Function code 0x06 - request message</i> | 64 |
| <i>Tabel 60: Function code 0x06-response message</i> | 65 |
| <i>Tabel 61: Function code 0x06-abnormal</i> | 65 |
| <i>Tabel 62: Function Code 0x06-Request Message-Example</i> | 66 |

| | |
|---|----|
| <i>Tabel 63: Function Code 0x06-Response Message-Example</i> | 66 |
| <i>Tabel 64: Function code 0x10 - request message</i> | 66 |
| <i>Tabel 65: Function code 0x10-response message</i> | 67 |
| <i>Tabel 66: Function code 0x10-abnormal</i> | 67 |
| <i>Tabel 67: Function code 0x10-request message-example</i> | 68 |
| <i>Tabel 68: Write Holding Registers - Convert Data Decimal</i> | 68 |
| <i>Tabel 69: Function Code 0x10-Response Message-Example</i> | 68 |
| <i>Tabel 70: Modbus Register Mapping address - I/O Modules</i> | 68 |
| <i>Tabel 71: Modbus Register Mapping address - Serial Port Module</i> | 69 |