GCAN-205

Industrial Modbus TCP-to-CAN Module

User Manual



Document Version: V3.33 (2024/09/03)

English Translation and Editorial Work by Shenzhen MoCAN Electric Co., Ltd.

Revision History

Version	Date	Reason
V1.00	2013/06/16	Document created
V2.01	2013/12/20	Device operating parameters corrected
V3.01	2014/10/22	Some parameters added
V3.15	2017/04/22	Description of common Modbus function codes added
V3.2	2018/07/18	Document structure adjusted
V3.3	2020/10/10	Document corrected
V3.31	2022/09/15	Document corrected
V3.32	2024/07/30	Contact information updated
V3.33	2024/09/03	LED status corrected

Contents

Revision History 2
Contents 3
1. Function Introduction 5
1.1 Overview 5
1.2 Features 5
1.2.1 Hardware Features 5
1.2.2 CAN Characteristics 5
1.2.3 Ethernet Characteristics 6
1.3 Typical Applications 6
2. Device Installation 6
2.1 Device Dimensions 6
2.2 Device Mounting 7
2.3 Interface Definition and Functions 8
3. Device Usage 8
3.1 Connection to PC 9
3.1.1 Restoring Factory Defaults 9
3.1.2 Changing the PC IP Address 9
3.1.3 Windows XP/7 Network Settings 10
Changing the Local IP Address 10
3.2 Connection to Ethernet 11
3.3 Connection to CAN-bus 11
3.4 CAN-bus Terminating Resistors 11
3.5 System Status Indicators 12
4. Gan205 EtherCanConfigv3 Software 13
4.1 Restoring Factory Defaults 13
4.2 Basic Parameter Configuration 13
4.2.1 Connecting to the GCAN-205 with a PC 13
4.2.2 CAN Baud Rate Setting 15
4.2.3 Ethernet Address Setting 15
4.3 Setting the Modbus TCP-to-CAN Data Receive Mapping Table 16
4.4 Setting the CAN Data Transmit-to-Modbus TCP Mapping Table 16
4.5 Downloading to the GCAN-205 Flash 17
4.6 Saving GCAN-205 Module Parameters 17

5. Application Examples 18
5.1 Reading Received CAN Frames 18
5.2 Writing CAN Frames to Be Sent 18
6. Technical Specifications 19
8. Frequently Asked Questions 20
9. Disclaimer 21
Copyright and Translation Notice 21
Product Information and User Responsibility 21
Interpretation 21
Appendix A: Introduction to the Modbus TCP Protocol 23
A.1 Modbus TCP Protocol Data Format 23
A.2 Common Modbus Function Codes 25
(1) Function Code: 03H 25
(2) Function Code: 10H (decimal 16) 25

1. Function Introduction

1.1 Overview

The GCAN-205 module is a Modbus TCP slave-to-CAN protocol conversion module that integrates one standard Ethernet interface and one standard CAN interface. With the GCAN-205, users can easily interconnect a CAN-bus network and an Ethernet network, further extending the range of the CAN-bus network.

The GCAN-205 bridges Ethernet networks and CAN-bus networks. Users can integrate this intelligent protocol conversion module into their own systems, giving Ethernet and CAN-bus networks — which would otherwise be unable to communicate with each other — the ability to communicate quickly, thereby saving development time, reducing development cost and capturing market opportunities rapidly.

The GCAN-205 is widely used in field-bus laboratories, industrial control networks, smart community monitoring and other network environments. Compact and plug-and-play, with DIN-rail mounting, it is especially suitable for use in industrial field installations or in cabinets together with other equipment.

The GCAN-205 integrates an electrical-isolation protection module for the CAN interface to protect the device from damage caused by transient over-current/over-voltage and enhance system reliability in harsh environments.

Users can configure the GCAN-205 through the supplied "Gan205 EtherCanConfigv3" software. Currently this configuration software supports only the conversion between the Modbus TCP and CAN protocols; other standard or custom protocols on the Ethernet or CAN side are not yet supported for user configuration. If required, our company can provide configuration services between any protocols.

1.2 Features

1.2.1 Hardware Features

- High-speed 32-bit industrial-grade controller
- Embedded hardware watchdog timer
- External power supply (24 V DC, 75 mA)
- Operating temperature range: -40°C to +85°C
- Electrostatic discharge immunity: contact discharge ±4 kV, air discharge ±8 kV
- Electrical fast transient burst immunity: ±2 kV
- Surge immunity: power interface ±1 kV, CAN-bus interface ±4 kV
- Standard DIN-rail mounting, designed for industrial use

1.2.2 CAN Characteristics

- Integrates one CAN-bus interface using terminal-block wiring
- CAN-bus lines: CAN_H, CAN_L, CAN_GND
- Supports CAN 2.0A and CAN 2.0B frame formats, compliant with ISO/DIS 11898
- CAN-bus communication baud rate from 5 kbps to 1 Mbps
- CAN-bus interface electrically isolated; isolation withstand voltage 1500 V DC

1.2.3 Ethernet Characteristics

- RJ45, supports 10/100 M auto-negotiation
- Modbus slave supported function codes: 03H, 04H, 06H, 16H
- Supports static or dynamic IP acquisition
- Automatically restores connection resources after a network outage, reliably establishing TCP connections
- Compatible with SOCKET operation; host computer communication software follows standard SOCKET rules

1.3 Typical Applications

- Interconnection of industrial Ethernet devices and CAN-network devices
- Power communication networks
- Industrial control equipment
- High-speed, high-volume communication
- Gateway/bridge between a CAN bus and a serial bus
- Industrial field network data monitoring
- CAN teaching applications and remote communication
- CAN industrial automation control systems
- Low-speed CAN network data acquisition and analysis
- Intelligent building control data broadcasting and other CAN-bus application systems
- PLC devices connecting to a CAN-bus network

2. Device Installation

2.1 Device Dimensions

The device outline dimensions are (length, including terminal block) 115 mm x (width) 23 mm x (height) 100 mm, as shown in Figure 2.1.

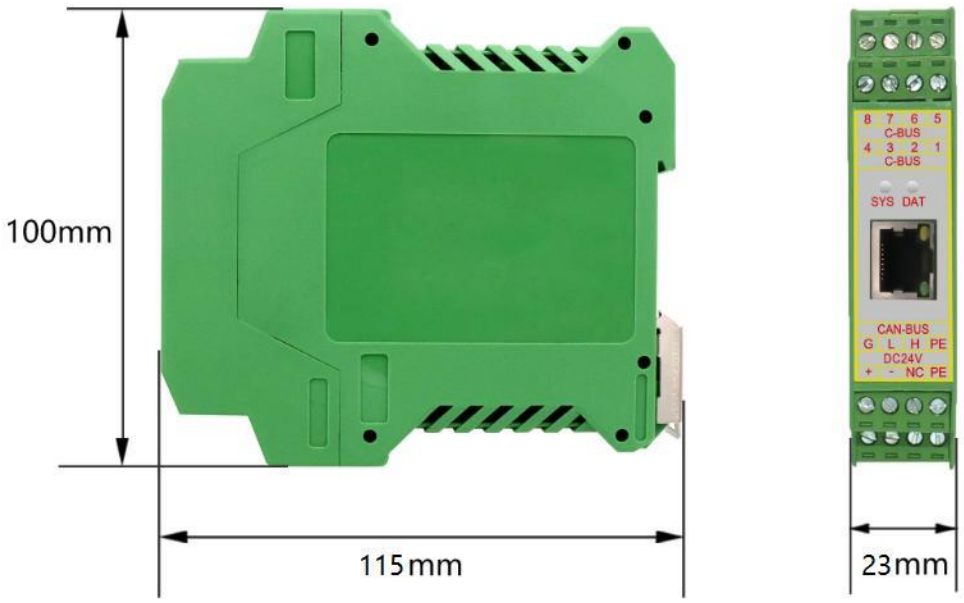

Figure 2.1 Outline dimensions of the GCAN-205 module

2.2 Device Mounting

The GCAN-205 is installed as shown in Figure 2.2; a flat-head screwdriver can be used to help mount the module onto the DIN rail.

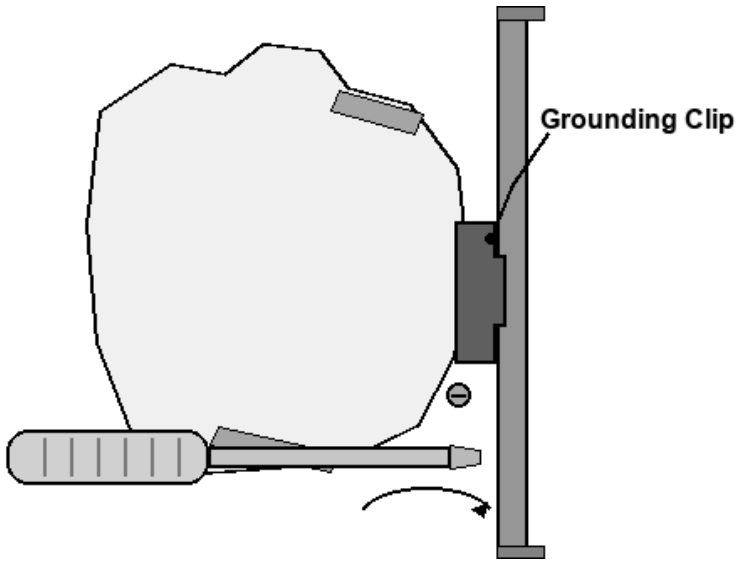

Figure 2.2 GCAN-205 module installation

Please ensure the device is well grounded, otherwise there is a risk of electric shock. Grounding should use an independent or single-point ground, not a common ground. The GCAN-205 requires an external ground wire, namely the PE pin of the terminal block.

2.3 Interface Definition and Functions

The GCAN-205 integrates one 24 V DC power interface, one standard CAN-bus interface and one standard Ethernet interface. The terminal block of the GCAN-205 is shown in Figure 2.3.



Figure 2.3 GCAN-205 terminal block

The power interface of the GCAN-205 is brought out through one 4-pin pluggable terminal block; its definition is shown in Table 2.1.

Pin (left to right)	Port name	Function
1	24 V DC +	24 V DC power input positive
2	-	24 V DC power input negative
3	NC	Not used
4	PE	Shield

Table 2.1 Power interface definition of the GCAN-205

The CAN-bus interface of the GCAN-205 is brought out through one 4-pin terminal block and can be used to connect a CAN-bus device; its definition is shown in Table 2.2.

Pin (left to right)	Port name	Function
G	CAN-G	CAN_GND
L	CAN-L	CAN_L signal line (CAN low)
H	CAN-H	CAN_H signal line (CAN high)
PE	PE	Shield
1-8	C-BUS	NC, not used

Table 2.2 CAN-bus signal assignment of the GCAN-205

3. Device Usage

The working principle of the GCAN-205 is shown in Figure 3.1.

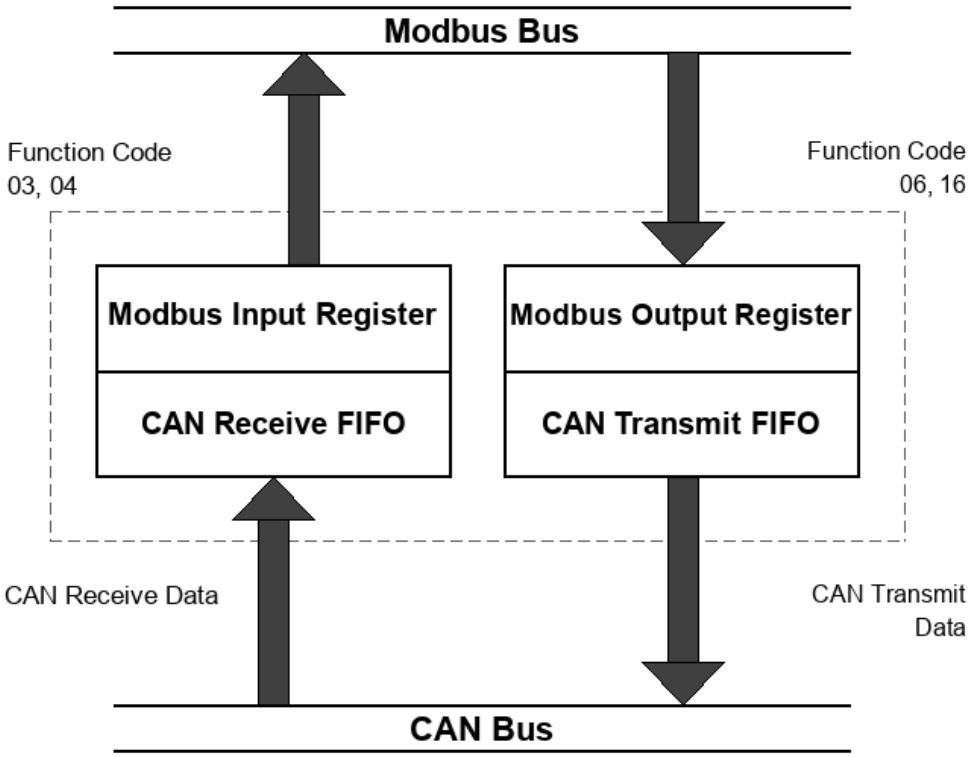

Figure 3.1 Working principle of the GCAN-205

3.1 Connection to PC

The GCAN-205 is powered by 24 V DC. Once the device is correctly powered, the "Gan205 EtherCanConfigv3" configuration software on the PC can be used to configure its working mode and basic operating parameters (see Chapter 4). Currently the GCAN-205 supports user configuration only for communication from Modbus TCP to CAN; other protocols are not yet supported for user configuration.

3.1.1 Restoring Factory Defaults

The factory default IP address of the GCAN-205 is 192.168.1.10. If the user has changed the IP address and forgotten it, the parameters can be reset; see section 4.1 for details.

3.1.2 Changing the PC IP Address

Before communicating with the GCAN-205, make sure the PC has an Ethernet adapter and that the PC and the GCAN-205 are on the same network segment. The GCAN-205 is assigned a default IP address (192.168.1.10) and subnet mask (255.255.255.0) at the factory. The user can check whether the device is on the same network segment as the PC using the flow shown in Figure 3.2.

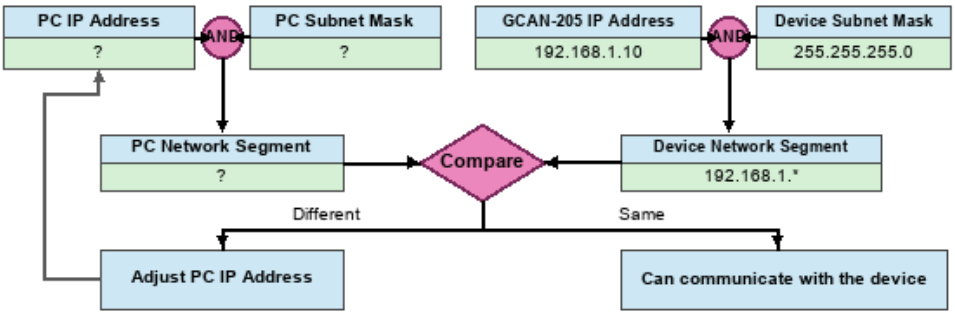

Figure 3.2 Flow for checking whether the PC and GCAN-205 are on the same network segment

Please note: Only when they are on the same network segment can you configure the GCAN-205 from the PC. If the network segments differ, make the following settings on the PC.

3.1.3 Windows XP/7 Network Settings

If the operating system is Windows XP/7, 8 or 10, you can change the local IP address and network segment. For Windows 8 and Windows 10, follow the Windows 7 procedure.

Changing the Local IP Address

After entering the operating system, open the Control Panel, then "Network Connections" (WinXP) or "Network and Sharing Center" (Win7/8/10), then the "Local Area Connection" properties, then the properties of "Internet Protocol (TCP/IP)" (WinXP) or "Internet Protocol Version 4 (TCP/IPv4)" (Win7); the dialog shown in Figure 3.3 appears.

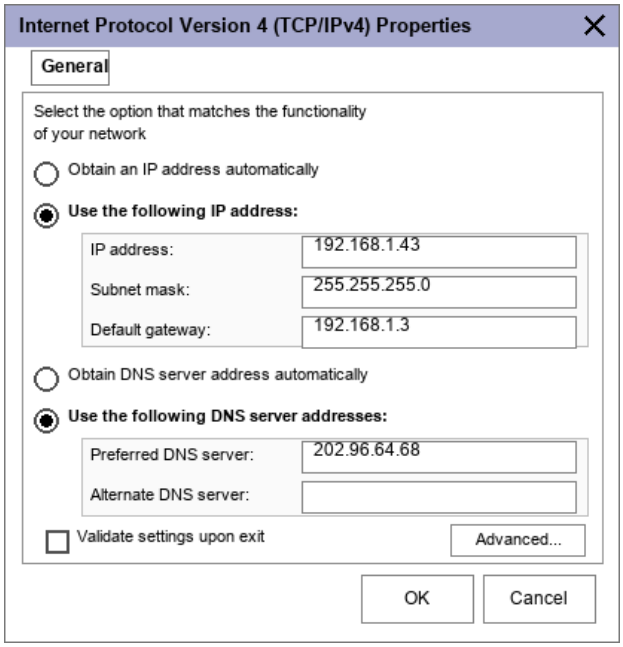

Figure 3.3 General TCP/IP settings

In the "IP address" field, click to modify and enter an IP address on the same network segment as the GCAN-205, as shown in Figure 3.3. If the PC is set to obtain an IP address automatically, select "Use the following IP address" in the dialog of Figure 3.3 and enter an IP address on the same network segment as the GCAN-205.

3.2 Connection to Ethernet

The Ethernet interface of the GCAN-205 integrates a 10/100 M auto-negotiation Ethernet chip, compliant with standard Ethernet protocols and supporting plug-and-play. Users can connect industrial Ethernet to the GCAN-205 using a Category-5 or better network cable.

3.3 Connection to CAN-bus

To connect the GCAN-205 to a CAN bus, simply connect CAN_H to CAN_H and CAN_L to CAN_L to establish communication. The CAN-bus network uses a linear topology; the two most distant ends of the bus need 120-ohm terminating resistors. If there are more than two nodes, the intermediate nodes do not need 120-ohm terminating resistors. For branch connections, the length should not exceed 3 meters. The CAN-bus connection is shown in Figure 3.4.

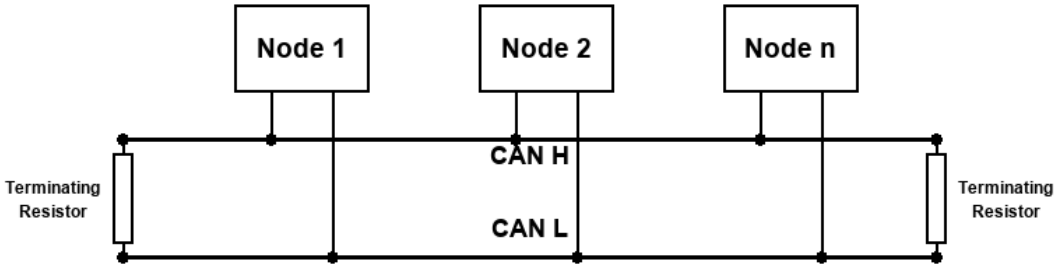

Figure 3.4 Topology of a CAN-bus network

Note: The CAN-bus cable can use ordinary twisted pair or shielded twisted pair. The theoretical maximum communication distance depends mainly on the bus baud rate; the relationship between maximum bus length and baud rate is shown in Table 3.1. If the communication distance exceeds 1 km, ensure the wire cross-section is greater than 1.0 mm²; the specific specification depends on the distance and is generally increased appropriately as the distance grows.

Baud rate	Bus length
1 Mbit/s	25 m
500 kbit/s	100 m
250 kbit/s	250 m
125 kbit/s	500 m
50 kbit/s	1 km
20 kbit/s	2.5 km
10 kbit/s	5 km
5 kbit/s	13 km

Table 3.1 Baud rate vs. maximum bus length

3.4 CAN-bus Terminating Resistors

To enhance the reliability of CAN communication and eliminate signal-reflection interference at the bus ends, terminating matching resistors are usually added at the two most distant ends of the CAN-bus network, as shown in Figure 3.5. The value of the terminating resistor is

determined by the characteristic impedance of the transmission cable. For example, if the characteristic impedance of the twisted pair is 120 Ω, then a 120-ohm terminating resistor should be integrated at both ends of the bus.

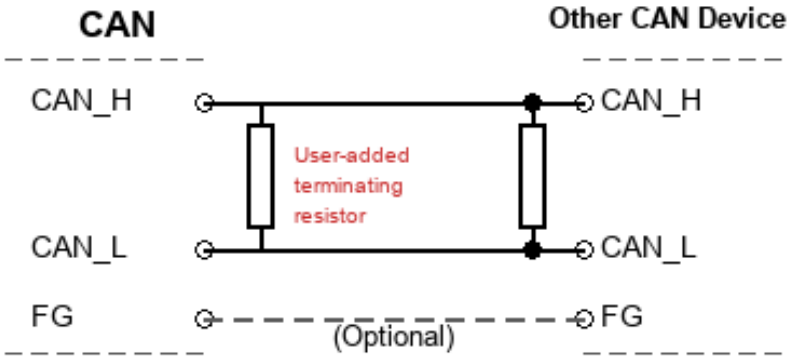

Figure 3.5 GCAN-205 connected to other CAN node devices

Please note: The GCAN-205 does not have a 120-ohm terminating resistor integrated internally; instead, external resistor terminals are provided. When a terminating resistor is needed, connect the two ends of the resistor to CAN_L and CAN_H respectively.

3.5 System Status Indicators

The GCAN-205 has one SYS indicator to show the operating status of the device and one DAT indicator to show data transmission. The functions of these two indicators are listed in Table 3.2, and the CAN-bus status corresponding to the various indicator states is shown in Table 3.3.

Indicator	Color	Indication
SYS	Green	System running indication
DAT	Green	Data conversion and transmission indication

Table 3.2 GCAN-205 indicators

After the GCAN-205 is powered on, the system-initialization indicator SYS lights up, indicating that the device is powered and the system is initializing; otherwise, there is a power fault or a serious error in the system. After both the Ethernet side and the CAN side are connected correctly, the data-signal indicator DAT flashes when data is being transferred between the buses.

Indicator	State	Indication
SYS	Flashing	Device initialization passed, standby state
	Off	Device initialization failed
DAT	Off or steady on	No data transmission between buses
	Flashing	Data transmission between buses

Table 3.3 GCAN-205 indicator states

4. Gan205 EtherCanConfigv3 Software

4.1 Restoring Factory Defaults

The factory default IP address of the GCAN-205 is 192.168.1.10. If the user has changed the IP address and forgotten it, the parameters can be reset through the DIP switch on the device. The default CAN baud rate is 1M, and the default Modbus mapping tables are all 0.

The procedure is: first do not power the module; use a flat-head screwdriver to pry open the clips at the top and bottom of the module and open the housing; locate the DIP switch shown in Figure 3.2, set switch No. 2 to the ON position, then power the system on and wait about 3 seconds until the SYS indicator on the front panel flashes; then turn off the power and set switch No. 2 back to OFF. At this point the device has been restored to its factory defaults, with the default IP 192.168.1.10.

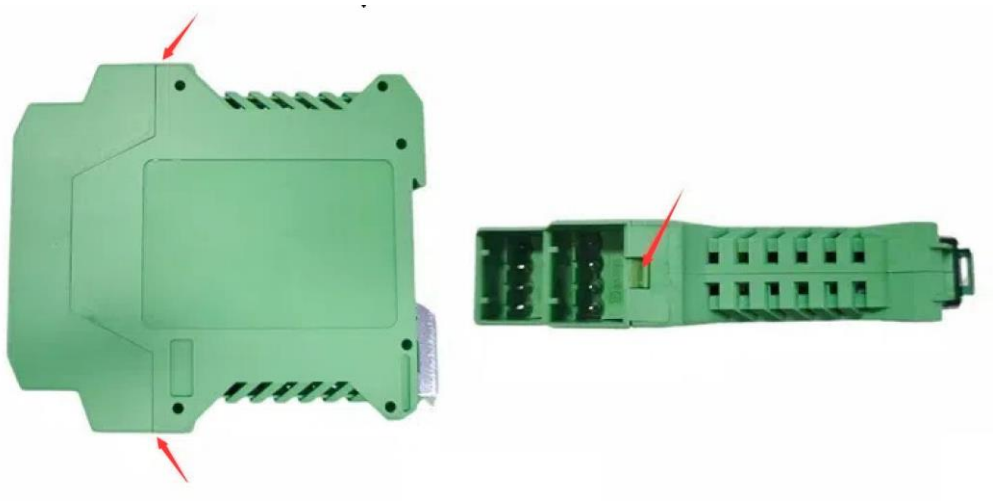

Figure 4.1 Restoring factory defaults on the GCAN-205

Please note: After restoring factory defaults, all parameter settings and mapping-table settings are cleared; please operate with care.

4.2 Basic Parameter Configuration

The GCAN-205 can be configured with the "Gan205 EtherCanConfigv3" software, including working mode, working port, destination port, destination IP, CAN working mode and CAN baud rate.

4.2.1 Connecting to the GCAN-205 with a PC

1. First power on the GCAN-205 and connect it to the PC with a network cable. When the device's SYS indicator flashes, the GCAN-205 has finished initialization and is waiting for connection.

2. Open the "Gan205 EtherCanConfigv3" software from the CD, enter the IP address of the GCAN-205 and click "Connect" to establish the connection, as shown in Figure 4.2.



Figure 4.2 Initial interface of the configuration software

After clicking "Connect", the software main interface shown in Figure 4.3 appears.

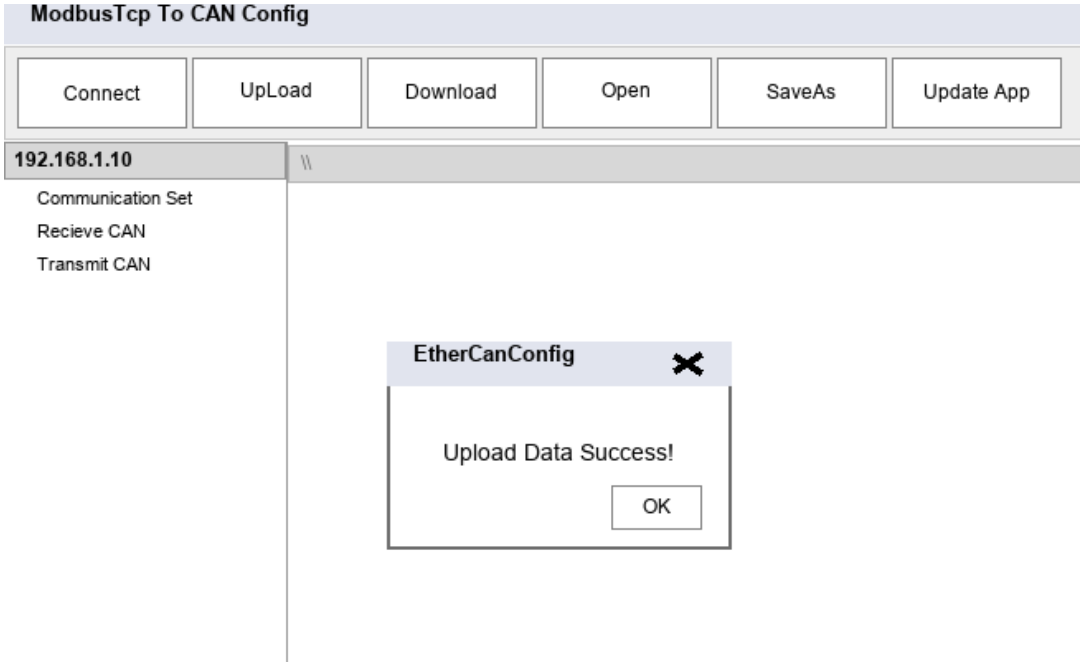

Figure 4.3 Main interface of the configuration software

There are six tool buttons at the top of the software:

"Connect" — connect to the device;

"Upload" — read the configuration information from the device;

"Download" — download the configuration information to the device's Flash;

"Open" — open and read a configuration file on the PC;

"SaveAs" — save the configuration file to the PC;

"Update App" — reserved by the manufacturer.

3. After connecting, the left column of the software shows the setting list of the connected device. You can click "Upload" to upload the device parameters to the PC. After a successful upload, the information shown in Figure 4.4 is displayed.

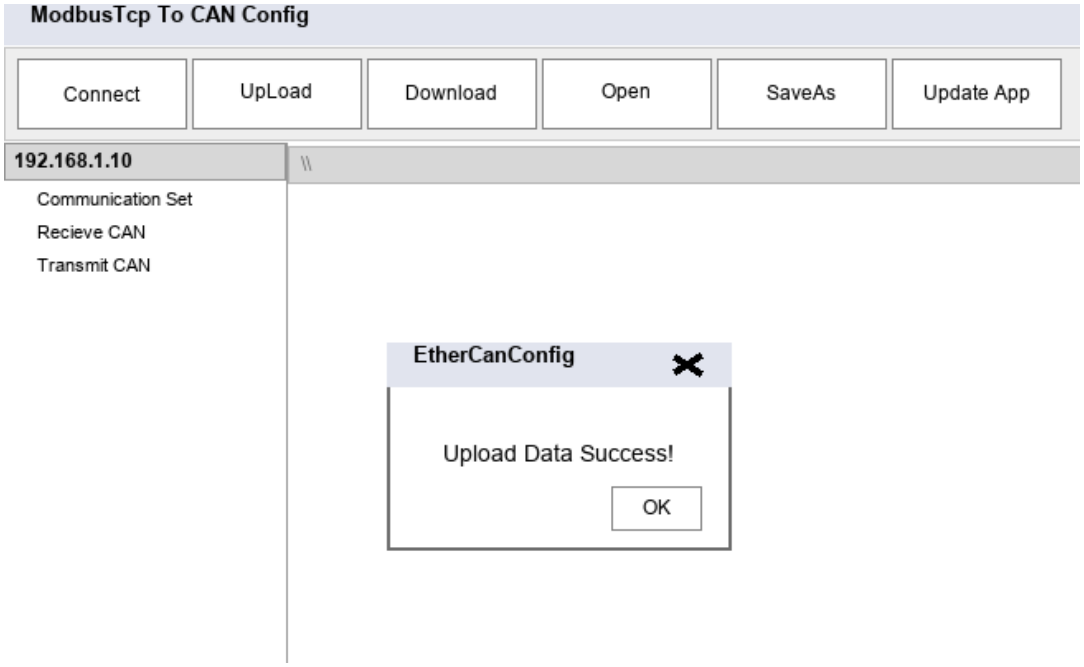

Figure 4.4 Upload successful

4.2.2 CAN Baud Rate Setting

"CAN Baudrate" sets the CAN-bus baud rate; enter it according to Table 4.1.

Parameter	Corresponding baud rate
0	1000k
1	800k
2	666k
3	500k
4	400k
5	250k
6	200k
7	125k
8	100k
9	80k
10	50k
11	40k
12	20k
13	10k
14	5k

Table 4.1 Baud rate parameter correspondence

4.2.3 Ethernet Address Setting

"IP Address" — sets the IP address of the device;

"Subnet Mask" — sets the subnet mask;

"Gateway" — sets the communication gateway.

After a parameter is modified, the corresponding field is marked in red.

4.3 Setting the Modbus TCP-to-CAN Data Receive Mapping Table

The GCAN-205 buffers received CAN-frame data in the Modbus TCP address table according to the set correspondence, waiting to be read by a Modbus command. Click "Receive CAN" in the left column of the interface, as shown in Figure 4.5.

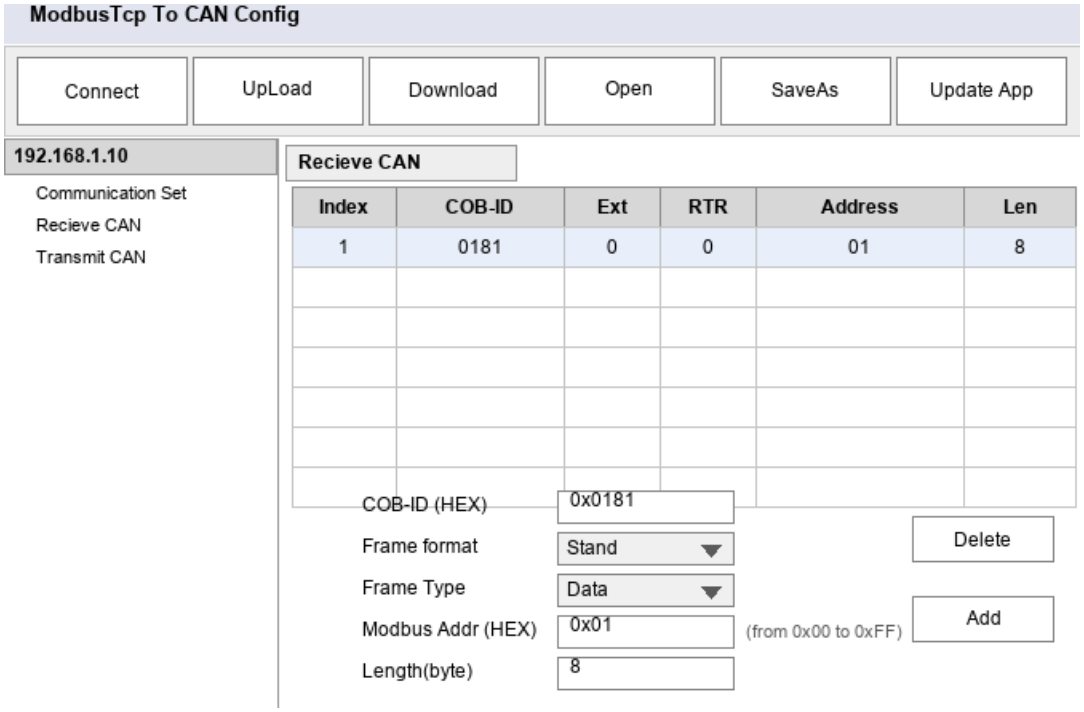

Figure 4.5 Receive CAN setting interface

In the lower-right column of the interface, click "Add" to add mapping data, up to 32 groups (data shorter than 8 bytes can save register usage); the Modbus addresses for received data range from 0x00 to 0x7F, 127 addresses in total, each being an integer variable. COB-ID is the CAN-bus frame ID setting; Frame Format is the frame format setting (Standard/Extended); Frame Type is the frame type setting (Data/RTR); Modbus Address is the starting register address on the Modbus side; Length is the CAN frame data length, with a maximum of 8. Note: this length is in bytes, while a Modbus address is an integer (2 bytes); therefore, if the CAN data length is 8 bytes, the corresponding Modbus address mapping is 4 registers.

4.4 Setting the CAN Data Transmit-to-Modbus TCP Mapping Table

Select "Transmit CAN" and add data mapping groups, up to 32 groups (data shorter than 8 bytes can save register usage), as shown in Figure 4.6.

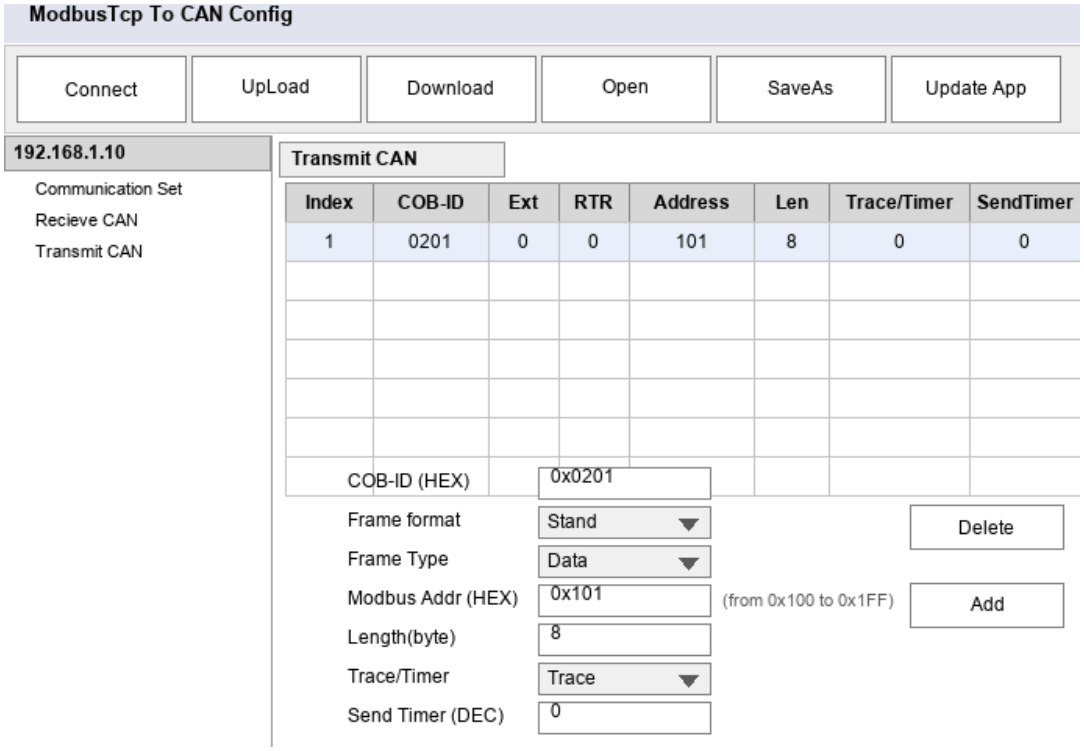

Figure 4.6 Transmit CAN setting interface

The Modbus mapping addresses for CAN transmit data range from 0x100 to 0x17F, each address being an integer (2 bytes). COB-ID is the frame ID setting for data forwarded to the CAN bus; Frame Format is the frame format setting (Standard/Extended); Frame Type is the frame type setting (Data/RTR); Modbus Address is the starting register address on the Modbus side; Length is the CAN frame data length, with a maximum of 8.

You can set the data trigger mode or the cyclic send mode (Trace/Timer). Data trigger mode (Trace): when a Modbus master writes data to the corresponding Modbus address with function code 06 and the data changes, the corresponding CAN frame data is triggered to be sent. Cyclic send mode (Timer): the corresponding CAN frame data is sent cyclically at a set interval; the interval is set in Send Timer as a decimal number in milliseconds — for example, entering 1000 means sending once every 1000 ms.

4.5 Downloading to the GCAN-205 Flash

After configuration is complete, click "Download" in the toolbar to write the configuration data into the device's FLASH. After the data is written successfully, power the device on again to enable the new settings.

4.6 Saving GCAN-205 Module Parameters

Both the configuration parameters you set and those uploaded from the device can be saved to the PC. Click "SaveAs" in the toolbar and follow the prompts to set a file name, then the configuration parameters are saved to the PC. The configuration file can be opened and used again.

Please note: After the data is written successfully, power the device on again to enable the new configuration.

5. Application Examples

Users can send Modbus commands for debugging using the network debug assistant supplied with the device. When using it, select TCP Client and set the remote host address to 192.168.1.10:502 (the IP address is a factory default and can be modified; the port number cannot be modified).

Please note: When using the network debug assistant, please select "Hex Display" and "Hex Send".

5.1 Reading Received CAN Frames

For example: configure the Modbus slave address as 1, the Modbus starting register address as 0x01 (factory default), function code 03, CAN standard frame with frame ID 0x181, and CAN frame data 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08.

The Modbus master sends the request frame: 00 00 00 00 00 06 01 03 00 01 00 04

The GCAN-205 response frame: 00 00 00 00 00 0B 01 03 08 02 01 04 03 06 05 08 07

At this point, the Modbus side of the GCAN-205 has received a data frame with frame ID 0x181 sent from the CAN side of another device.

Master sends	Bytes	Example (Hex)
Transaction identifier	2 bytes	00 00
Protocol identifier	2 bytes	00 00
Data length	2 bytes	00 06
Device address	1 byte	01
Function code	1 byte	03
Starting address	2 bytes	00 01
Register quantity	2 bytes	00 04

Table 5.1 Request frame from the master

Slave returns	Bytes	Example (Hex)
Transaction identifier	2 bytes	00 00
Protocol identifier	2 bytes	00 00
Data length	2 bytes	00 0B
Device address	1 byte	01
Function code	1 byte	03
Response byte count	1 byte	08
Register values	8 bytes	02 01 04 03 06 05 08 07

Table 5.2 Response frame from the slave

5.2 Writing CAN Frames to Be Sent

For example: the Modbus starting register address is 0x100 (256 in decimal, the factory default), function code 16 (10H), CAN standard frame with frame ID 0x201, and CAN frame data 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08.

The Modbus master sends the request frame: 00 00 00 00 00 0F 01 10 01 01 00 04 08 02 01 04 03 06 05 08 07

The GCAN-205 response frame: 00 00 00 00 00 06 01 10 01 01 00 04

At this point, the CAN side of the other device has received a data frame with frame ID 0x201 sent from the Modbus side of the GCAN-205.

Master sends	Bytes	Example (Hex)
Transaction identifier	2 bytes	00 00
Protocol identifier	2 bytes	00 00
Data length	2 bytes	00 0F
Device address	1 byte	01
Function code	1 byte	10
Starting address	2 bytes	01 01
Register quantity	2 bytes	00 04
Response byte count	1 byte	08
Register values	8 bytes	02 01 04 03 06 05 08 07

Table 5.3 Request frame from the master

Slave returns	Bytes	Example (Hex)
Transaction identifier	2 bytes	00 00
Protocol identifier	2 bytes	00 00
Data length	2 bytes	00 06
Device address	1 byte	01
Function code	1 byte	10
Starting address	2 bytes	01 01
Register quantity	2 bytes	00 04

Table 5.4 Response frame from the slave

6. Technical Specifications

Category	Item	Specification
Connection	Ethernet	RJ45
	CAN	OPEN4 terminal block
Interface features	Ethernet interface	10/100 M auto-negotiation
	CAN interface	Compliant with ISO 11898, supports CAN 2.0A/B
	CAN baud rate	5 kbit/s to 1 Mbit/s
	Electrical isolation	1500 V DC
	CAN terminating resistor	Not integrated
Power supply	Supply voltage	9-24 V DC
	Supply current	75 mA max (24 V DC)
Environmental	Operating	-40°C to +85°C

test	temperature	
	Operating humidity	15% to 90% RH, non-condensing
EMC test	Standard	EN 55024:2011-09; EN 55022:2011-12
Protection	Ingress rating	IP 20
General	Dimensions	115 mm x 100 mm x 23 mm
	Weight	120 g

Table 6.1 Technical specifications

8. Frequently Asked Questions

1. Is a 120-ohm terminating resistor always necessary?

A 120-ohm terminating resistor is recommended to absorb end reflections and provide a stable physical link. A complete CAN bus needs one and only two 120-ohm terminating resistors, connected at the two most distant nodes of the bus.

2. Can multiple GCAN-205 modules be installed on one CAN bus?

Because of the structure of a CAN bus, multiple CAN nodes can be connected to one bus; therefore, as long as the bus is not affected, multiple GCAN-205 gateways can be installed on one CAN bus. The actual number depends on the maximum number of nodes supported by the CAN bus.

3. What is the maximum data conversion rate of the GCAN-205?

A single CAN channel of the GCAN-205 supports a maximum CAN-bus data conversion rate of 8000 fps. The frames mentioned here are standard data frames with 8 data bytes; frames with fewer than 8 data bytes or remote frames may be faster.

4. Why is the DAT indicator not lit?

The DAT indicator lights up only when data is being transmitted on the CAN or Ethernet side and the module is converting data.

5. Why does the system report an illegal operation when an interface function is called?

First, when using the interface functions, carefully read the function descriptions to ensure the input parameters are legal, paying special attention to the passing of pointers (addresses), or refer to the provided example programs. If the problem is still not solved, please contact our technical support.

6. How is the communication baud rate of the GCAN-205 set?

The Gan205 EtherCanConfigv3 software provides a set of commonly used baud-rate values. To use other baud rates, please contact Guangcheng Technology.

7. Does the system entering standby or sleep mode affect reception?

Yes. All processing will stop, which may at worst cause a hardware receive-buffer overflow error. If a program has opened the device, it will try to prevent the system from entering standby or sleep mode to ensure normal operation. When using the GCAN-205, please disable the system's standby and sleep functions.

9. Disclaimer

Thank you for purchasing GCAN series hardware and software products.

1. Copyright and Translation Notice

The original Chinese-language version of this document was prepared by Shenyang Guangcheng Technology Co., Ltd., which retains the copyright to the original document and its contents.

This English version has been independently translated and prepared by Shenzhen MoCAN Electric Co., Ltd. based on its understanding of the original Chinese documentation. It is intended solely for overseas customer reference and technical support purposes. It has not been officially reviewed, verified or approved by Shenyang Guangcheng Technology Co., Ltd., and should not be regarded as an official or authoritative English version of the original documentation.

The English translation, editorial work and original layout elements created by Shenzhen MoCAN Electric Co., Ltd. are protected by applicable copyright laws. No part of this English version may be reproduced, redistributed, republished or used for commercial purposes without prior written permission from Shenzhen MoCAN Electric Co., Ltd., except as permitted by applicable law.

GCAN is a registered trademark of Shenyang Guangcheng Technology Co., Ltd.

2. Product Information and User Responsibility

Product specifications, technical parameters, functions and other information provided in this document are for reference purposes and may be subject to change without prior notice.

Users are responsible for evaluating the suitability of the products for their specific applications and ensuring proper installation, configuration and operation in accordance with the relevant technical documentation and applicable safety requirements.

For product selection, technical clarification or application-specific requirements, users are advised to consult the manufacturer or an authorized supplier before making purchasing or implementation decisions.

3. Interpretation

This English document is provided as a reference translation of the original Chinese-language documentation. While reasonable efforts have been made to ensure the accuracy and consistency of the translation, differences in technical terminology, interpretation or expression may occur.

In the event of any discrepancy, ambiguity or inconsistency between this English version and the original Chinese-language documentation, the original Chinese-language version shall prevail.

For technical questions or clarification regarding the product specifications and intended functions, users are encouraged to contact Shenyang Guangcheng Technology Co., Ltd. or their authorized supplier.

Appendix A: Introduction to the Modbus TCP Protocol

The Modbus communication protocol is a universal language developed by Modicon for use on PLCs or other industrial controllers. Through this protocol, serial communication between controllers can be realized. The Modbus protocol defines the message structure that a controller can recognize and use, and describes the process by which a master controller accesses slave devices — for example, how a slave responds and how transmission errors are checked and reported. Modbus uses a master-slave communication method. The master first sends a communication request to a slave device, and the slave returns response data to the master according to the function code in the request. Each slave device on the network must be assigned a unique address; up to 31 slave devices are supported. Information exchange between the master controller and slave devices is achieved through up to 24 bus commands. A slave executes only commands addressed to itself and does not respond to messages beginning with other slave addresses. This request-response communication mode greatly improves the accuracy of communication. Because of its simple operation, high efficiency and reliable communication, the Modbus protocol has become an international communication standard supported by most industrial control product manufacturers worldwide. It is widely used in machinery, water conservancy, electric power, environmental protection and other industrial equipment.

The Modbus TCP communication protocol can be used for monitoring automation equipment. A common application is to develop gateways based on this protocol, through which PLCs, I/O modules and other buses can be connected to Ethernet. Modbus TCP does not change the original Modbus protocol; it simply migrates it as an application-layer protocol onto TCP/IP. Every Modbus TCP call requires a response. Using TCP/IP, a web-based interface can make the user interface friendlier; a web browser can be used to view the operation of devices on an enterprise network. Schneider has registered port 502 for Modbus, so real-time data can be embedded in web pages; by embedding a web server in a device, a web browser can serve as the device's operating terminal. However, the Modbus protocol itself has some shortcomings: it does not support newer widely adopted network technologies such as object-based communication models, so users must manually configure some parameters such as information data types and register numbers.

A.1 Modbus TCP Protocol Data Format

The TCP/IP protocol and the Ethernet link-layer checksum mechanism already ensure the correctness of data packet delivery, so the CRC-16 or LRC check field is no longer present in a Modbus TCP message; instead, a Modbus Application Protocol header (MBAP) is added. It interprets the Modbus parameters and functions. Each TCP/IP message can contain only one Modbus frame.

In a Modbus TCP ADU, the MBAP header occupies 7 bytes (containing 4 sub-fields): the Transaction Identifier TI, the Protocol Identifier PI, the Length field L (2 bytes, indicating the total length of the Protocol Identifier and the Data field) and the Unit Identifier UI. TI occupies 2 bytes to identify the sequence of Modbus frames; PI occupies 2 bytes to identify the application-layer protocol; UI occupies 1 byte to identify the Modbus device unit. The function code occupies 1 byte and can be divided into bit operations and 16-bit word operations. The function code indicates the operation to perform — for example, function code 15 means write multiple bit registers, and function code 06 means write to an individual 16-bit word register. The data field can be up to 248 bytes, with a specific format depending on the

function code. When a client sends a request, the data field gives the starting address (2 bytes) and the number (1 byte) of registers to operate on; when a server sends a response, the data field gives the number of operated registers (1 byte) and the status values of each register. Figure A.1 compares the Modbus and Modbus TCP frame formats.

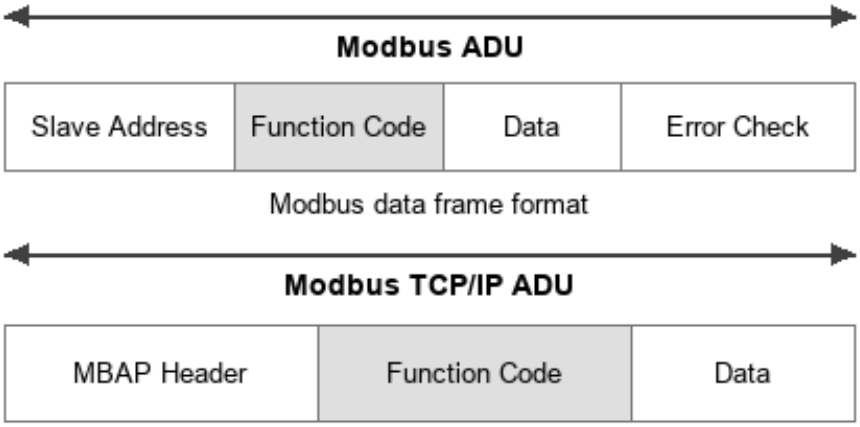

Figure A.1 Modbus vs. Modbus TCP/IP frame format

The ADU data unit specification of Modbus TCP is shown in Table A.1.

Description	Bytes
MBAP header - Transaction ID high byte (Hi)	1
MBAP header - Transaction ID low byte (Lo)	1
MBAP header - Protocol identifier	2
MBAP header - Length identifier	2
MBAP header - Unit identifier	1
Modbus request - Function code	1
Modbus request - Starting address	2
Modbus request - Number of registers	2

Table A.1 ADU data unit specification of Modbus TCP

Before transferring data over Modbus TCP, a TCP/IP connection must be established between the client and the server. The server uses port 502 as the Modbus TCP connection port. Establishment of a Modbus TCP connection is usually implemented automatically by the TCP/IP socket interface software protocol, so it is completely transparent to the application. Once the TCP/IP connection between client and server is established, the same connection can be used to transfer any amount of user data in the required direction. The client and server can also establish multiple TCP/IP connections simultaneously; the maximum number of connections depends on the TCP/IP interface specification.

When a device sends a request, its corresponding device must respond. The response data format is shown in Table A.2.

Byte	Response data
Byte0, Byte1	Transaction identifier = 0 (copied from the request in the response)

Byte2, Byte3	Protocol identifier
Byte4	Length identifier high byte = 0
Byte5	Length identifier low byte (indicates how many bytes follow)
Byte6	Unit identifier (slave device address)
Byte7	Modbus function code
Byte8	Data

Table A.2 Modbus TCP response data format

A.2 Common Modbus Function Codes

Among the function codes in a Modbus message frame, the most commonly used are 01, 02, 03, 04, 06 and 16, which can implement read and write operations on digital and analog quantities in slaves. The following describes these function codes in detail, using communication in RTU transmission mode as an example.

Below are two Modbus commands with their master/slave transmit and receive data packet formats; other commands can follow the same format.

(1) Function Code: 03H

Function: Read holding registers.

Description: Reads the binary data in the slave's holding registers; broadcasting is not supported.

Query: The query message specifies the starting address and the number of registers to read. Register addressing starts at 0000; registers 1-16 correspond to addresses 0-15.

Response: The register data in the response is binary data; each register corresponds to 2 bytes, the first byte being the high-order value and the second byte the low-order value.

(2) Function Code: 10H (decimal 16)

Function: Preset multiple registers.

Description: Presets data into registers (type 4x) in sequence. In broadcast mode this function code can preset data into the same type of registers in all slaves. Note that this function code can override the controller's memory protection; the preset values in registers remain valid until processed by the next control logic of the controller. If no program for the register exists in the control logic, the value in the register remains unchanged.

Query: The message specifies the type of registers to preset; register addressing starts at 0. The query data field specifies the preset values. M84 and 484 controllers use 10-bit binary data (2 bytes) with the remaining high 6 bits set to 0, while other controller types use one 16-bit binary value (2 bytes) per register.

Response: A normal response returns the slave address, function code, starting address and number of preset registers.

Function Code	Name	Function Description
01	Read Coils	Read the current status of a group of coils (ON/OFF)
02	Read Discrete Inputs	Read the current status of a group of switch inputs (ON/OFF)
03	Read Holding Registers	Read the binary values in one or more holding registers
04	Read Input Registers	Read the binary values in one or more input registers
05	Write Single Coil	Force the ON/OFF state of a single logic coil
06	Write Single Register	Load a specific binary value into a holding register
07	Read Exception Status	Get the ON/OFF status of 8 internal coils
08	Diagnostics	Send a diagnostic check message to the slave for communication diagnosis
16	Write Multiple Registers	Load specific binary values into a series of consecutive holding registers
128~255	Reserved	Used for exception responses

Table A.3 Common Modbus function codes