GCAN-204

Modbus RTU-to-CAN Converter Module

User Manual



Document Version: V3.32 (2024/07/30)

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

Revision History

Version	Date	Reason
V1.00	2013/6/16	Document created
V2.01	2013/12/20	Device operating parameters corrected
V3.01	2014/10/22	Some parameters added
V3.02	2015/08/08	Some parameters added
V3.23	2017/03/22	Description of common Modbus function codes added
V3.25	2018/07/18	Document structure adjusted
V3.30	2020/10/10	Document corrected
V3.31	2022/09/15	Document corrected
V3.32	2024/07/30	Contact information updated

Contents

Revision History 2
Contents 3
1. Function Introduction 5
1.1 Overview 5
1.2 Features 5
1.3 Typical Applications 5
2. Device Installation 6
2.1 Device Dimensions 6
2.2 Device Mounting 7
2.3 Interface Definition and Functions 7
3. Device Usage 9
3.1 Serial Connection 9
3.2 Connection to CAN-bus 9
3.3 CAN-bus Terminating Resistors 10
3.4 System Status Indicators 11
4. GCAN-204-Config-CH-3.12 Configuration Software 11
4.1 Preparation for Configuration 11
4.2 Software Connection 12
4.3 Configuring Communication Parameters 13
4.4 Configuring the Conversion Mapping Table 15
4.4.1 Setting the Modbus RTU-to-CAN Data Receive Mapping Table 15
4.4.2 Setting the CAN Data Transmit-to-Modbus RTU Mapping Table 16
4.5 Saving the Configuration 17
4.6 Saving GCAN-204 Module Parameters 17
5. Application Examples 17
5.1 Reading Received CAN Frames 17
5.2 Writing CAN Frames to Be Sent 18
6. Precautions 19
7. 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 Protocol 23
A.1 Modbus Protocol Data Format 23
(1) Address Field 23
(2) Function Field 24
(3) Data Field 24
(4) CRC Field 24
A.2 Common Modbus Function Codes 24
(1) Function Code: 03H 25
(2) Function Code: 10H (decimal 16) 25

1. Function Introduction

1.1 Overview

The GCAN-204 module is an industrial-grade Modbus RTU slave-to-CAN converter module that integrates one standard serial port and one standard CAN-bus interface. The serial-bus side of the GCAN-204 has the Modbus RTU protocol already integrated. Using the GCAN-204, devices that originally communicate over a serial bus (Modbus RTU protocol) can obtain a CAN-bus communication interface without changing their original hardware structure, thereby quickly connecting Modbus RTU devices to a CAN bus and forming nodes in a CAN-bus network.

When reading data, the GCAN-204 places data received from the CAN bus into a buffer in real time, waiting to be read by a Modbus command. After receiving a Modbus read command, the module converts the specified CAN data and sends it onto the Modbus bus; when the module receives a Modbus send command, it directly processes the Modbus data and sends it onto the CAN bus.

The GCAN-204 is a key tool for industrial-bus retrofitting and for interconnecting devices using different buses. Compact and plug-and-play, it is also the best choice for integrating existing systems. All bus interfaces of the GCAN-204 integrate isolation protection modules to protect the module from damage caused by transient over-voltage or over-current, and the DIN-rail mounting makes it easy to integrate into various control cabinets.

1.2 Features

- Modbus slave supported function codes: 03H, 04H, 06H, 16H
- Standard serial port uses a 2-wire RS485 interface
- Serial baud rate from 600 bps to 115200 bps, configurable by software
- CAN-bus 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
- Maximum conversion data throughput: 400 frames/second (measured at maximum serial baud rate)
- Powered by 9-24 V DC (24 V / 40 mA recommended)
- RS485 and CAN interfaces use terminal-block wiring
- Dedicated configuration software for setting the working mode and all parameters
- Non-volatile memory stores configuration parameters, automatically loaded at each power-on
- Can be mounted on a DIN rail
- 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

1.3 Typical Applications

- Connecting existing serial communication devices to a CAN-bus network
- Connecting PLC devices to a CAN-bus network

- Communication between Modbus RTU networks and CAN networks
- 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

2. Device Installation

The typical working environment of the GCAN-204 module is shown in Figure 2.1.

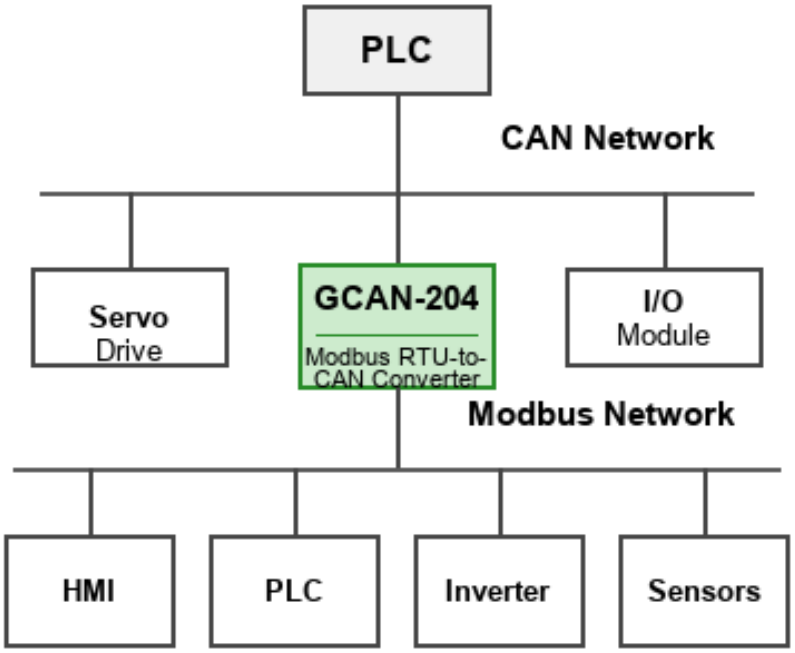

Figure 2.1 Typical working environment of the GCAN-204 module

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 2.2.

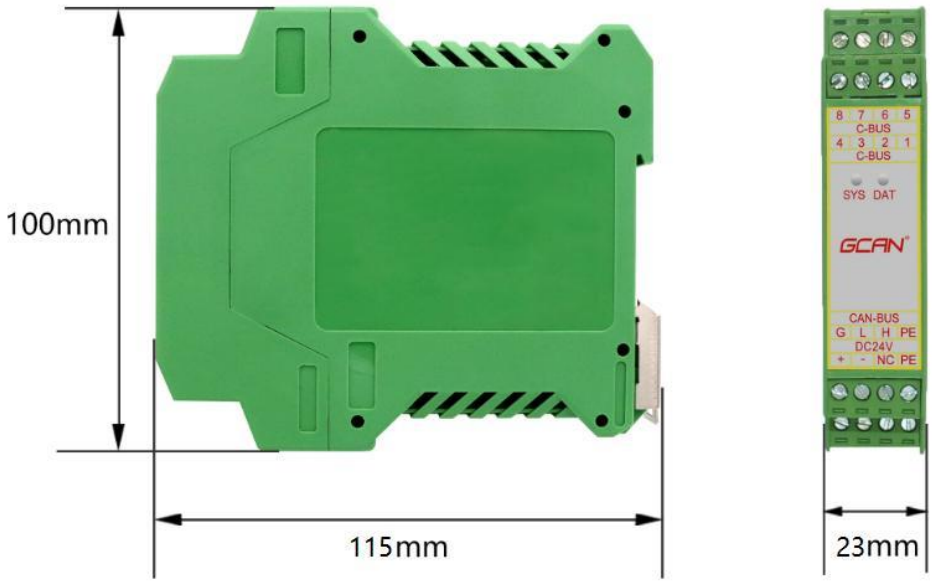

Figure 2.2 Outline dimensions of the GCAN-204 module

2.2 Device Mounting

The GCAN-204 is installed as shown in Figure 2.3; a flat-head screwdriver can be used to help mount the module onto the DIN rail.

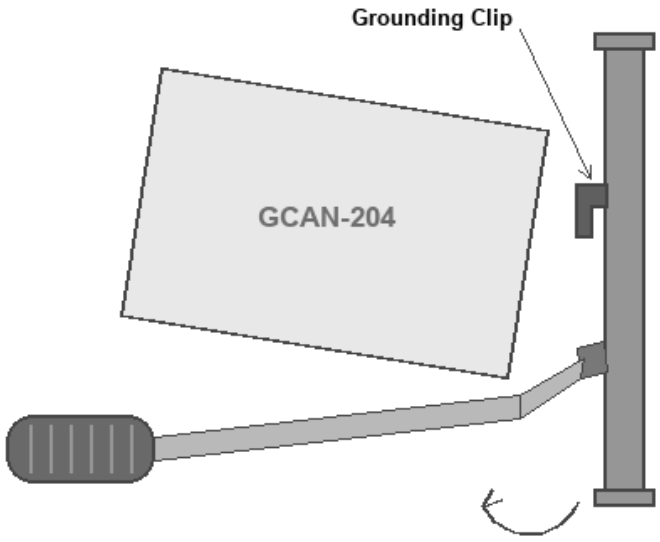

Figure 2.3 GCAN-204 module installation

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-204 requires an external ground wire, namely the PE pin of the terminal block.

2.3 Interface Definition and Functions

The GCAN-204 integrates one power interface, one standard CAN-bus interface and one standard RS485 interface. The terminal block of the GCAN-204 is shown in Figure 2.4.



Figure 2.4 GCAN-204 terminal block

The power interface of the GCAN-204 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	Power +	9-24 V DC power input positive
2	-	9-24 V DC power input negative
3	NC	Not used
4	PE	Shield

Table 2.1 Power interface definition of the GCAN-204

The CAN-bus interface of the GCAN-204 is brought out through one 4-pin terminal block and can be used to connect one CAN-bus network or a CAN-bus device; its definition is shown in Table 2.2.

Pin (left to right)	Port name	Function
G	CAN-G	CAN_GND (ground)
L	CAN-L	CAN_L signal line (CAN low)
H	CAN-H	CAN_H signal line (CAN high)
PE	PE	Shield

Table 2.2 CAN-bus signal assignment of the GCAN-204

The RS485 interface of the GCAN-204 is brought out through one 4-pin terminal block and is used to connect a Modbus RTU network or a Modbus RTU master device; its definition is shown in Table 2.3.

Pin (left to right)	Port name	Function
C-BUS 4	A+	485 A (+) signal line
C-BUS 1	B-	485 B (-) signal line
Others	NC	Not used

Table 2.3 RS485 interface definition of the GCAN-204

3. Device Usage

The working principle of the GCAN-204 is shown in Figure 3.1.

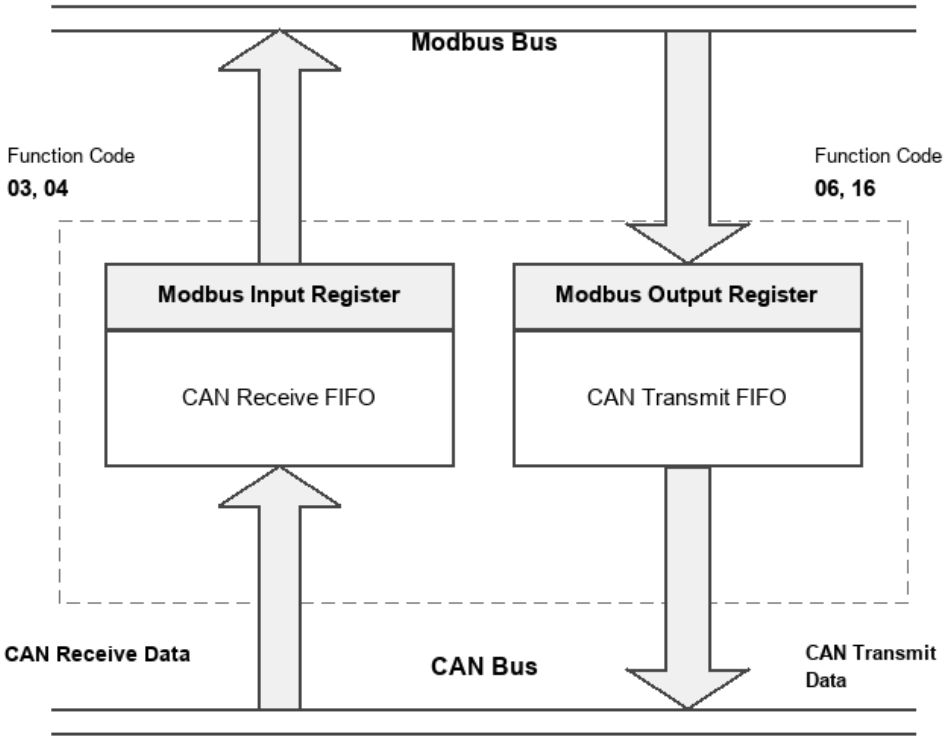

Figure 3.1 Working principle of the GCAN-204

3.1 Serial Connection

The GCAN-204 uses standard serial levels (RS485: -7 to +12 V), so the module can be connected directly to devices with an RS485 interface.

3.2 Connection to CAN-bus

To connect the GCAN-204 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 3.2.

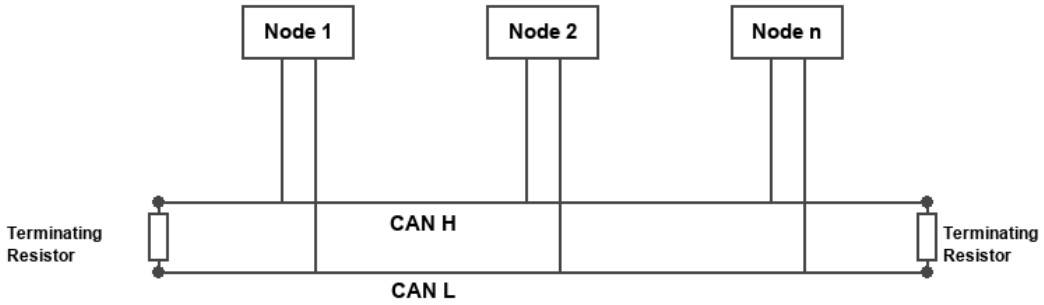

Figure 3.2 Topology of a CAN-bus network

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

3.3 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 3.3. 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. If other nodes on the network use different transceivers, the terminating resistor must be calculated separately.

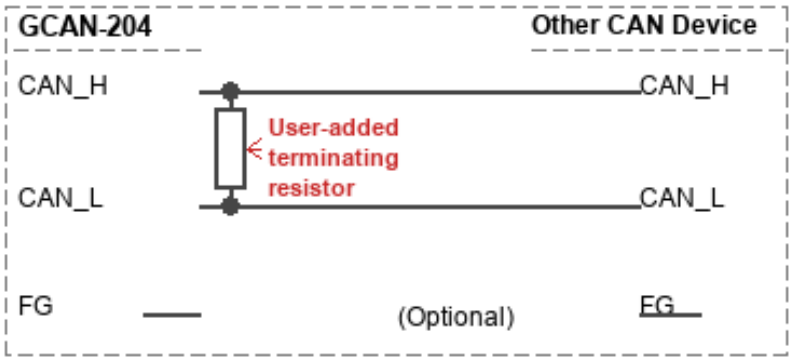

Figure 3.3 GCAN-204 connected to other CAN node devices

Please note: The GCAN-204 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.

3.4 System Status Indicators

The GCAN-204 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-204 indicators

After the GCAN-204 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 Modbus RTU 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	Flashing	Data transmission between buses
	Off or steady on	No data transmission between buses

Table 3.3 GCAN-204 indicator states

4. GCAN-204-Config-CH-3.12 Configuration Software

4.1 Preparation for Configuration

1) Disconnect the power from the GCAN-204, then gently press the upper and lower ends of the module's green housing with a flat-head screwdriver to open it.

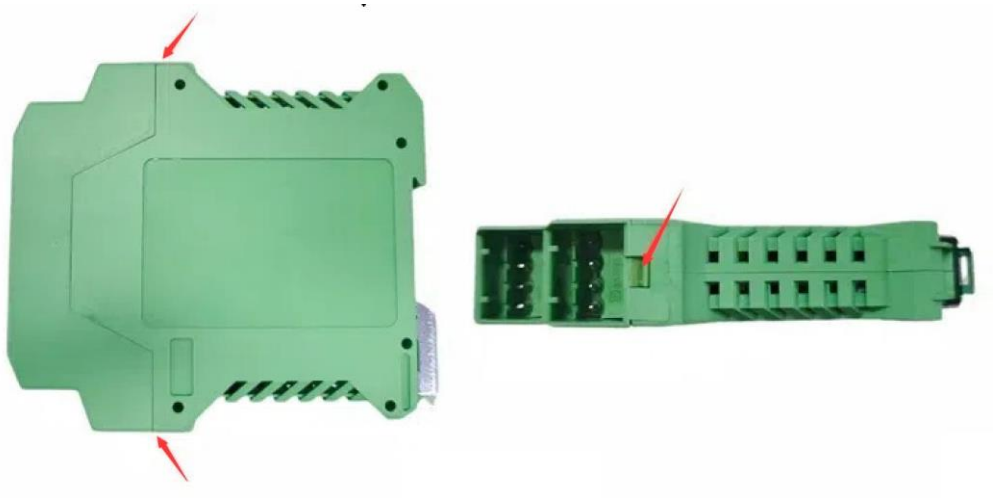

Figure 4.1 GCAN-204 housing position

2) Locate the DIP switch shown in Figure 4.2, set switch No. 2 to the ON position, then power the module on again to enter configuration mode.

Please note: The GCAN-204 is configured through the RS485 interface. The Mini USB port next to the DIP switch is reserved by the manufacturer — please do not connect it. After configuration is complete, set DIP switch No. 2 back to the OFF position.

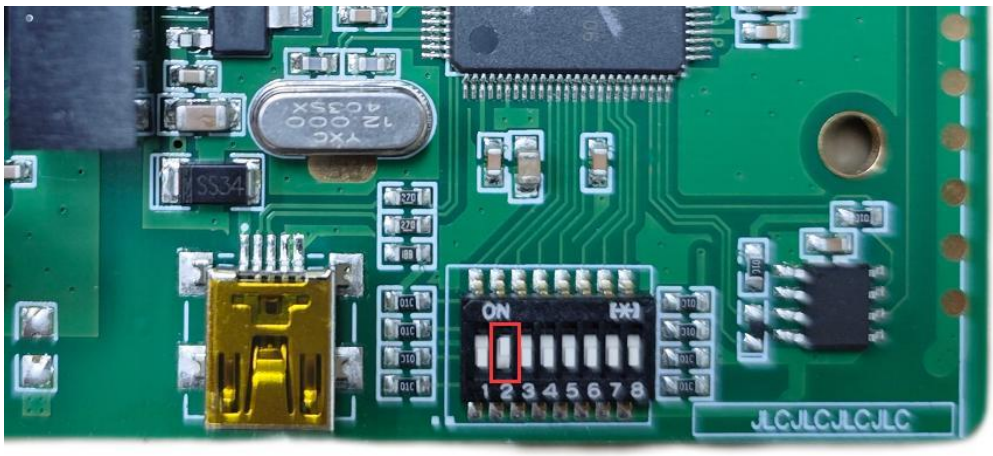

Figure 4.2 GCAN-204 DIP switch position

4.2 Software Connection

When the GCAN-204 is in configuration mode and correctly connected to the PC through the RS485 interface, open the "GCAN-204-Config" software to configure the module. The software interface is shown in Figure 4.3.

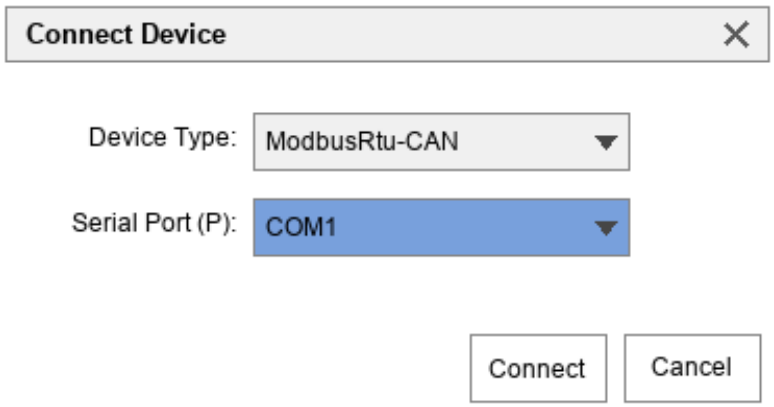

Figure 4.3 Initial interface of the GCAN-204 configuration software

After opening the software, select the serial port number connected to the GCAN-204 and click "Connect" to establish the connection. If you do not know the serial port number, you can check it in the PC's Device Manager; the software displays serial port numbers 1-7. If the number is outside this range, please change the serial port number yourself.

After the connection is established, click "UpLoad" to read the module's current configuration, as shown in Figure 4.4.

Please note: At this point the device is already connected to the PC; please do not click "Connect" again.

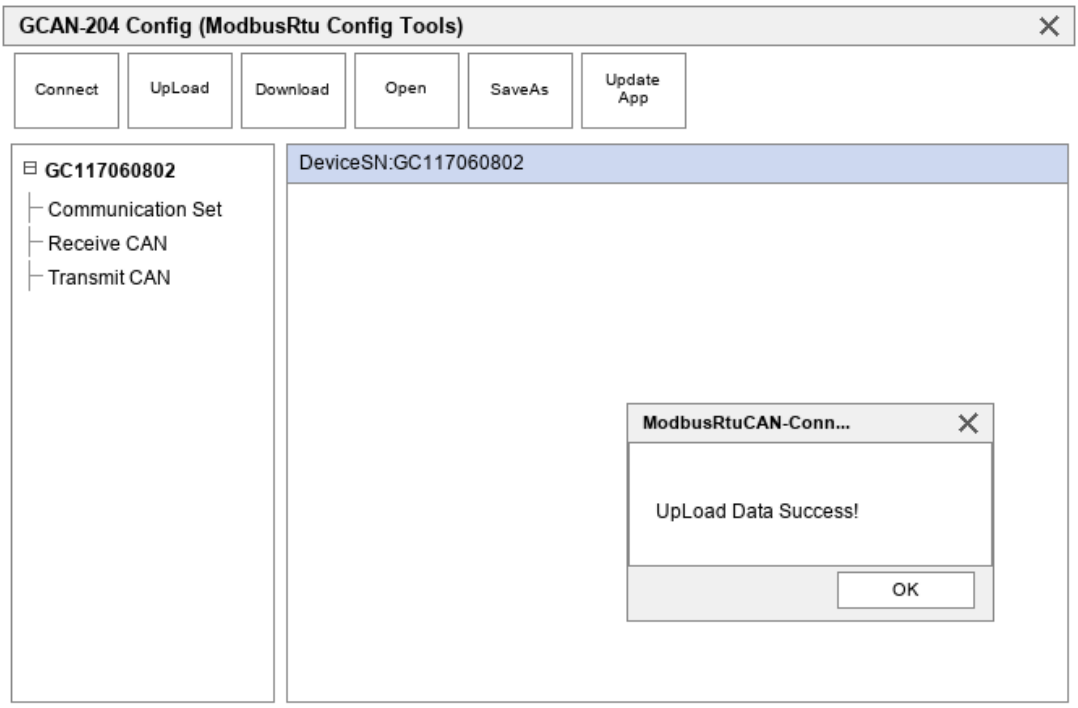

Figure 4.4 Main interface of the GCAN-204 configuration software

4.3 Configuring Communication Parameters

Click "Communication Set". In this interface you can set the Modbus node ID, serial parity, byte size and stop bits, the Modbus RTU-side baud rate (RS485 Baud) and the CAN-side baud rate (CAN Baudrate). The factory defaults are 57600 bps for the Modbus RTU side and

250 kbps for the CAN bus. The "Communication Setting" basic parameters are shown in Figure 4.5.

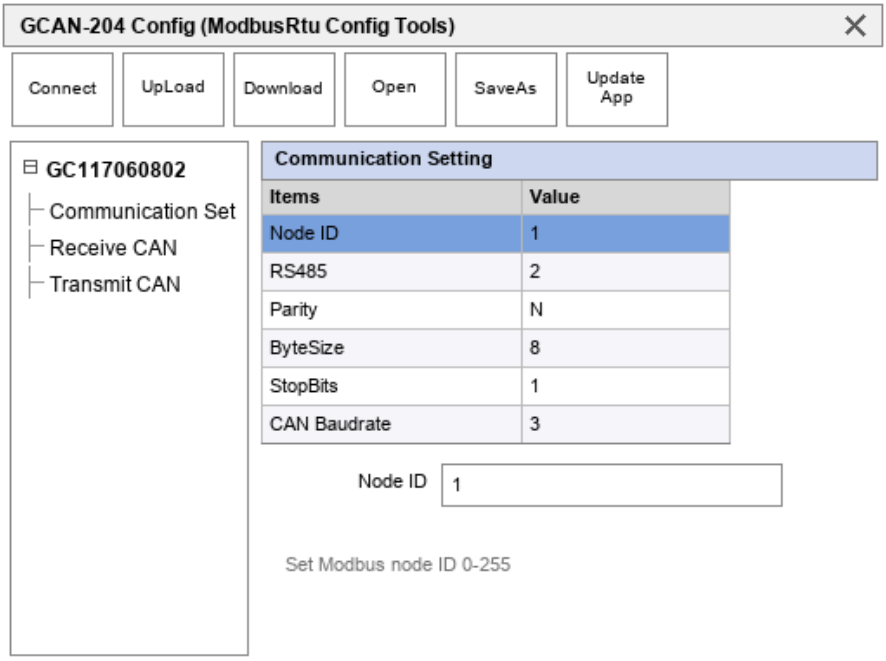

Figure 4.5 GCAN-204 parameter setting interface

"Node ID" — sets the Modbus node ID.

"RS485 Baud" — sets the RS485 bus communication baud rate; details are shown in Table 4.1.

"Parity" — sets the parity mode.

"ByteSize" — sets the data length; the default is 0.

"StopBits" — sets the number of stop bits; the default is 1.

"CAN Baudrate" — sets the CAN-side baud rate; details are shown in Table 4.2.

Baud rate	Corresponding number
0	0
115200	1
57600	2
56000	3
43000	4
38400	5
19200	6
9600	7
4800	8
2400	9
1200	10
600	11

Table 4.1 RS485 baud rate and corresponding number

Baud rate	Corresponding number
1000k	1
500k	2
250k	3
200k	4
125k	5
100k	6
50k	7
10k	8
5k	9

Table 4.2 CAN baud rate and corresponding number

Note: If you want to use a custom baud rate, please contact Shenyang Guangcheng Technology Co., Ltd.

4.4 Configuring the Conversion Mapping Table

The factory default number of mapping groups is 1 for both Receive and Transmit. Users can click Add in the interface to add mapping data, up to 32 groups (data shorter than 8 bytes can save register usage). The Modbus addresses of Receive CAN range from 0x00 to 0x7F, and those of Transmit CAN from 0x100 to 0x17F. This interface can set parameters such as CAN frame ID, frame format, frame type, corresponding Modbus address and CAN frame data length.

4.4.1 Setting the Modbus RTU-to-CAN Data Receive Mapping Table

The GCAN-204 can buffer received CAN data with specified frame IDs in the Modbus RTU address table according to the set correspondence, waiting to be read by a Modbus command. The Modbus register addresses for received data range from 0x00 to 0x7F, 128 addresses in total, each being an integer variable. Click "Receive CAN" in the left column of the interface to enter the CAN data receive mapping interface, as shown in Figure 4.6. 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.

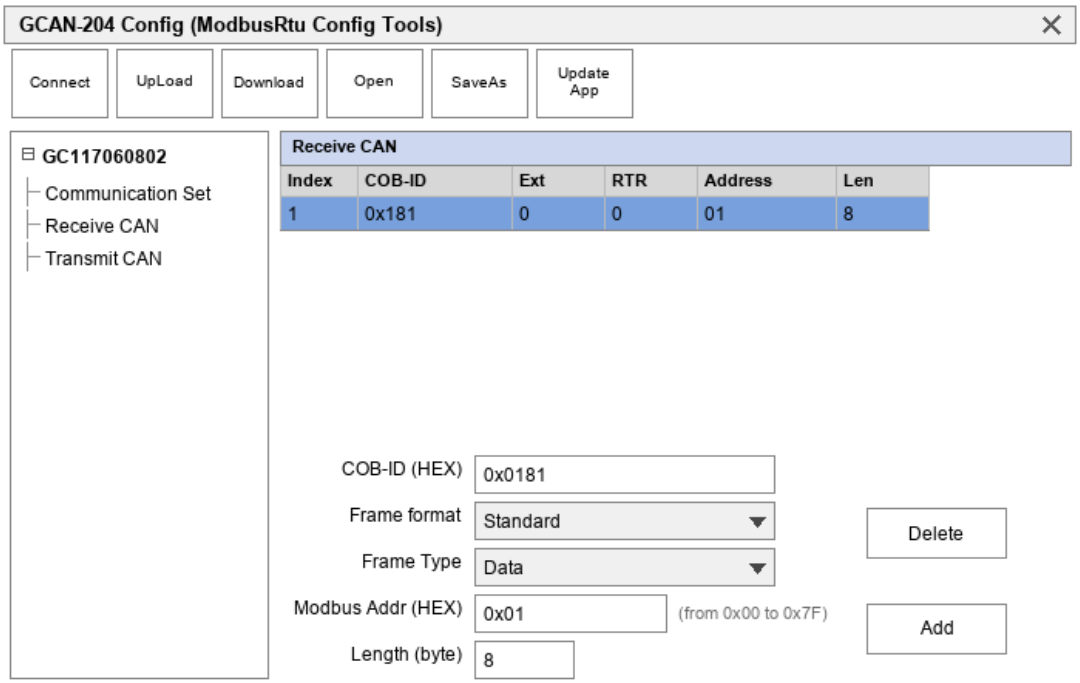

Figure 4.6 Receive CAN mapping settings

4.4.2 Setting the CAN Data Transmit-to-Modbus RTU Mapping Table

Select "Transmit CAN" to add data mapping groups, up to 32 groups (data shorter than 8 bytes can save register usage). Users can use function codes 06 and 16 to forward data from Modbus registers onto the CAN bus. The Modbus mapping addresses for CAN transmit data range from 0x100 to 0x17F, each address being an integer (2 bytes). Users can set the data trigger mode or cyclic send mode (Trace/Timer). As shown in Figure 4.7, 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.

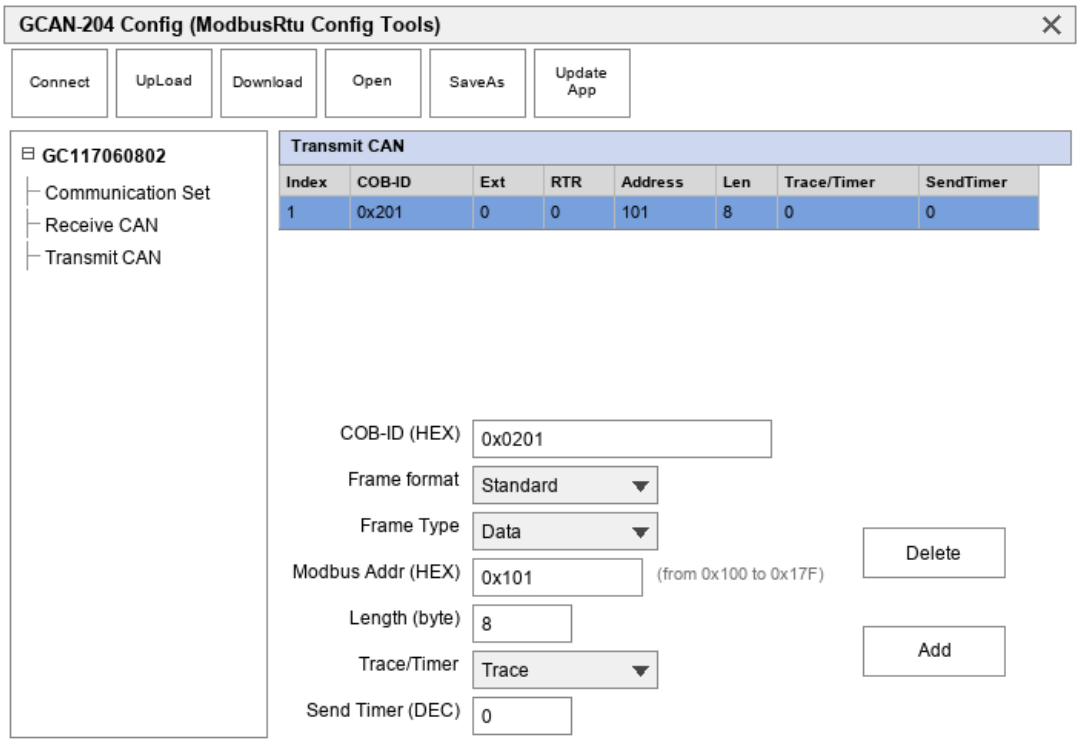

Figure 4.7 Transmit CAN mapping settings

4.5 Saving the Configuration

After configuration is complete, click "Download" in the toolbar to write the configuration data into the device's FLASH. A prompt appears after a successful write; if the write fails, please write again.

Please note: After the data is written successfully, set DIP switch No. 2 to OFF and power the device on again to enable the new configuration.

4.6 Saving GCAN-204 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.

5. Application Examples

Users can send Modbus commands for debugging using the serial-port debug assistant supplied with the device.

Please note: When using the serial-port 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: 01 03 00 01 00 04 15 C9

The GCAN-204 response frame: 01 03 08 02 01 04 03 06 05 08 07 17 C1

At this point, the Modbus side of the GCAN-204 has received a data frame with frame ID 0x181 sent from the CAN side of another device.

Master sends	Bytes	Example (Hex)
Slave address	1 byte	01
Function code	1 byte	03
Starting register address	2 bytes	00 01
Register quantity	2 bytes	00 04
CRC	2 bytes	15 C9

Table 5.1 Request frame from the master

Slave returns	Bytes	Example (Hex)
Slave address	1 byte	01
Function code	1 byte	03
Data byte count	1 byte	08
Data 1	2 bytes	02 01
Data 2	2 bytes	04 03
Data 3	2 bytes	06 05
Data 4	2 bytes	08 07
CRC	2 bytes	17 C1

Table 5.2 Response frame from the slave

5.2 Writing CAN Frames to Be Sent

For example: configure the Modbus slave address as 1, the Modbus starting register address as 0x101 (257 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: 01 10 01 01 00 04 08 02 01 04 03 06 05 08 07 CB 2E

The GCAN-204 response frame: 01 10 01 01 00 04 91 F6

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-204.

Master sends	Bytes	Example (Hex)
Slave address	1 byte	01
Function code	1 byte	10
Starting register address	2 bytes	01 01
Register quantity	2 bytes	00 04
Data byte count	1 byte	08
Data 1	2 bytes	02 01
Data 2	2 bytes	04 03

Data 3	2 bytes	06 05
Data 4	2 bytes	08 07
CRC	2 bytes	CB 2E

Table 5.3 Request frame from the master

Slave returns	Bytes	Example (Hex)
Slave address	1 byte	01
Function code	1 byte	10
Starting register address	2 bytes	01 01
Register quantity	2 bytes	00 04
CRC	2 bytes	91 F6

Table 5.4 Response frame from the slave

6. Precautions

- If the module receives two data frames with the same ID from the CAN bus, the later received data automatically overwrites the data previously stored at the Modbus address. If the CAN-bus data volume is large, the Modbus master must speed up its reading.
- After switching between "Configuration Mode" and "Normal Operation Mode", the device must be powered on again; otherwise the original working mode is still executed and the switch cannot be completed successfully.

7. Technical Specifications

Category	Item	Specification
Connection	Serial interface	Terminal block
	CAN interface	Terminal block
Interface features	Serial interface	Standard RS485 level interface
	Serial baud rate	600 bps to 115200 bps
	CAN interface	Compliant with ISO 11898, supports CAN 2.0A/B
	CAN baud rate	1000k, 500k, 250k, 200k, 125k, 100k, 50k, 10k, 5k
	Electrical isolation	1500 V DC
	CAN terminating resistor	Not integrated
Power supply	Supply voltage	9-24 V DC
	Supply current	40 mA max (24 V DC)
Environmental test	Operating temperature	-40°C to +85°C
	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 7.1 Technical specifications

8. Frequently Asked Questions

1. Is a 120-ohm terminating resistor always necessary on the CAN bus?

The 120-ohm terminating resistor is used 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. What is the maximum data conversion rate of the GCAN-204?

A single CAN channel of the GCAN-204 supports a maximum CAN-bus data conversion rate of 400 fps; the maximum conversion rate is measured at a serial baud rate of 115200 bps.

3. How is the communication baud rate of the GCAN-204 set?

The GCAN-204 provides a set of commonly used baud-rate values. To use other baud rates, please contact Shenyang Guangcheng Technology Co., Ltd.

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

Yes. All processing will stop, which may 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-204, 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 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.

A.1 Modbus Protocol Data Format

The Modbus protocol offers two data transmission modes for the user to choose from: ASCII (American Standard Code for Information Interchange) and RTU (Remote Terminal Unit). However, all devices on a Modbus network must select the same transmission mode and serial parameters. In RTU mode, each 8-bit byte in a message frame contains two 4-bit hexadecimal characters. Compared with ASCII mode, expressing the same information in RTU mode requires fewer bits and gives a higher data throughput at the same rate. Therefore, RTU mode is more commonly used. The GCAN-204 also uses RTU mode.

An RTU message begins and ends with at least a 3.5-character interval (T1-T2-T3-T4 in Table A.1). A frame consists of an address field, a function field, a data field and a CRC check field; all characters are hex digits 0-9 and A-F. The entire message frame must be transmitted as a continuous stream. If a pause of more than 1.5 character times occurs before the frame is complete, the receiving device discards the incomplete message and assumes the next byte is the address field of a new message. Likewise, if a new message begins less than 3.5 character times after the previous one, the receiving device treats it as a continuation of the previous message. This leads to an error because the value of the final CRC field cannot be correct.

Start Bit	Device Address	Function Code	Data	CRC Check	End Bit
T1-T2-T3-T4	8 Bit	8 Bit	N x 8 Bit	16 Bit	T1-T2-T3-T4

Table A.1 RTU message frame format

(1) Address Field

Specifies the destination address of the message, 8 bits. The address range of an individual device is 1 to 247. The master selects the slave by placing the slave's address in the address

field of the message. When the slave sends a response, it places its own address in the response address field so the master knows which device responded. Address 0 is used as a broadcast address so that all slaves can recognize it.

(2) Function Field

When a message is sent from the master to a slave, the function-code field tells the slave what actions to perform — for example, reading the status of inputs, reading the data contents of a group of registers, reading the diagnostic status of a slave, or allowing program download, recording and verification in the slave. When the slave responds, it uses the function-code field to indicate whether the response is normal (no error) or whether some error occurred (called an exception response). For a normal response, the slave simply echoes the corresponding function code. After the master application receives an exception response, the typical action is to resend the message, or diagnose the message sent to the slave and report it to the operator.

(3) Data Field

The data field is composed of two sets of hexadecimal digits, ranging from 00 to FF. The data field of a message from the master to the slave contains the parameters the slave needs to execute the master's function code, such as the address of the object register, the number of items to process and the actual number of data bytes in the field. For example, if the master needs to read a group of holding registers from a slave (function code 03), the data field specifies the starting register and the number of registers to read. If the master writes a group of registers in a slave (function code 16, i.e. 10H), the data field specifies the starting register to write, the number of registers, the number of data bytes and the data to be written. If no error occurs, the data field returned by the slave contains the requested data; if an error occurs, this field contains an exception code that the master application can use to determine the next action.

In some messages the data field may be absent (zero length). For example, if the master asks the slave to return a communication event record (function code 0BH), no additional information is needed from the slave.

When transmitting a 2-byte piece of data, the high byte (MSB) is transmitted first, followed by the low byte (LSB). This is the opposite of the DeviceNet transmission method.

(4) CRC Field

The CRC field checks the entire message content and consists of two bytes containing a 16-bit binary value. It is calculated by the transmitting device and added to the message. The receiving device recalculates the CRC of the received message and compares it with the value in the received CRC field; if the two values differ, an error has occurred. When the CRC is added to the message, the low byte is added first, followed by the high byte.

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 the master/slave transmit and receive data packet formats for two Modbus commands; 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.

Master sends	Bytes	Example (Hex)	Comment
Slave address	1 byte	01	Communicate with slave 01
Function code	1 byte	03	Read register data
Starting register address	2 bytes	00 01	Starting address is 0001H
Register quantity	2 bytes	00 03	Read 3 consecutive registers
CRC	2 bytes	54 0B	CRC of the first 6 bytes

Table A.2 Query for function code 03H

Slave returns	Bytes	Example (Hex)	Comment
Slave address	1 byte	01	Communicate with slave 01
Function code	1 byte	03	Read register data
Data byte count	1 byte	06	3 registers occupy 6 bytes
Data 1	2 bytes	02 0B	Data in register 0001H
Data 2	2 bytes	00 00	Data in register 0002H
Data 3	2 bytes	00 64	Data in register 0003H
CRC	2 bytes	84 BD	CRC of the first 9 bytes

Table A.3 Response for function code 03H

(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.

Master sends	Bytes	Example (Hex)	Comment
Slave address	1 byte	01	Communicate with slave 01
Function code	1 byte	10	Preset multiple registers
Starting register address	2 bytes	10 20	Starting address is 1020H
Register quantity	2 bytes	00 03	3 consecutive registers
Byte count	1 byte	06	3 registers occupy 6 bytes
Data 1	2 bytes	02 01	Data in register 1020H
Data 2	2 bytes	04 03	Data in register 1021H
Data 3	2 bytes	06 05	Data in register 1022H
CRC	2 bytes	BD 9B	CRC of the first 13 bytes

Table A.4 Query for function code 10H

Slave returns	Bytes	Example (Hex)	Comment
Slave address	1 byte	01	Communicate with slave 01
Function code	1 byte	10	Write register data
Starting register address	2 bytes	10 20	Starting address is 1020H
Register quantity	2 bytes	00 03	3 consecutive registers
CRC	2 bytes	85 02	CRC of the first 6 bytes

Table A.5 Response for function code 10H

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 current binary values in one or more holding registers
04	Read Input Registers	Read the current 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.6 Common Modbus function codes