Hopeland

Focus on RFID core technology

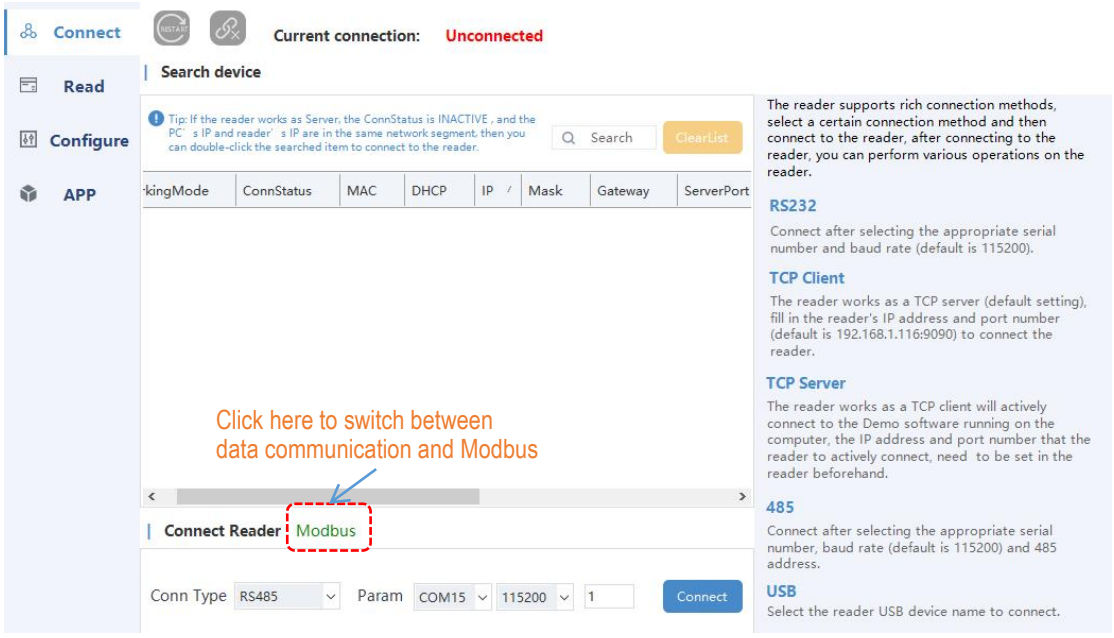
Brief Development Notes for Modbus Protocol

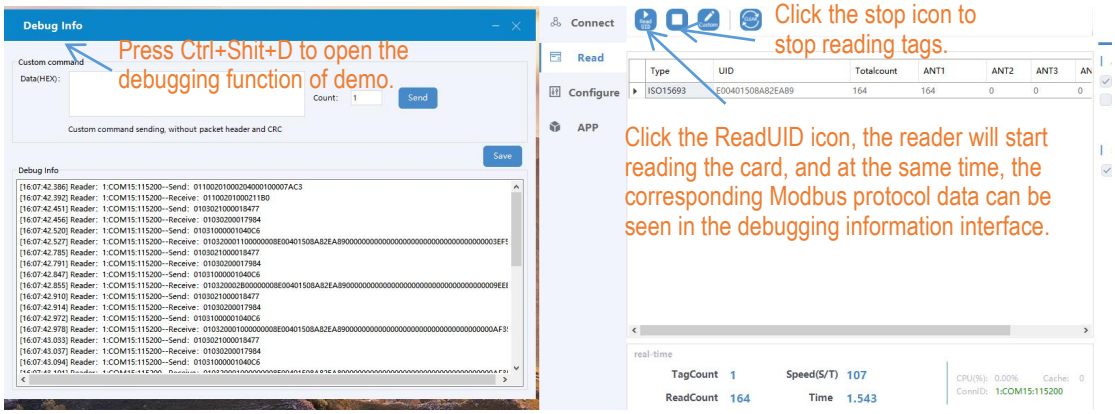
Contents

Before understanding the Modbus RTU/Modbus TCP communication protocol, it is recommended to use C # demo 4.46 (or above) to connect to the reader through RS485/TCP and test the reader's function according to the demo user manual.
After you turn on the debugging function of demo by pressing Ctrl + Shit + D, you can see the original data of the communication protocol corresponding to each step of operating demo. Combined with the communication protocol manual to see the specific commands, it will be very clear.

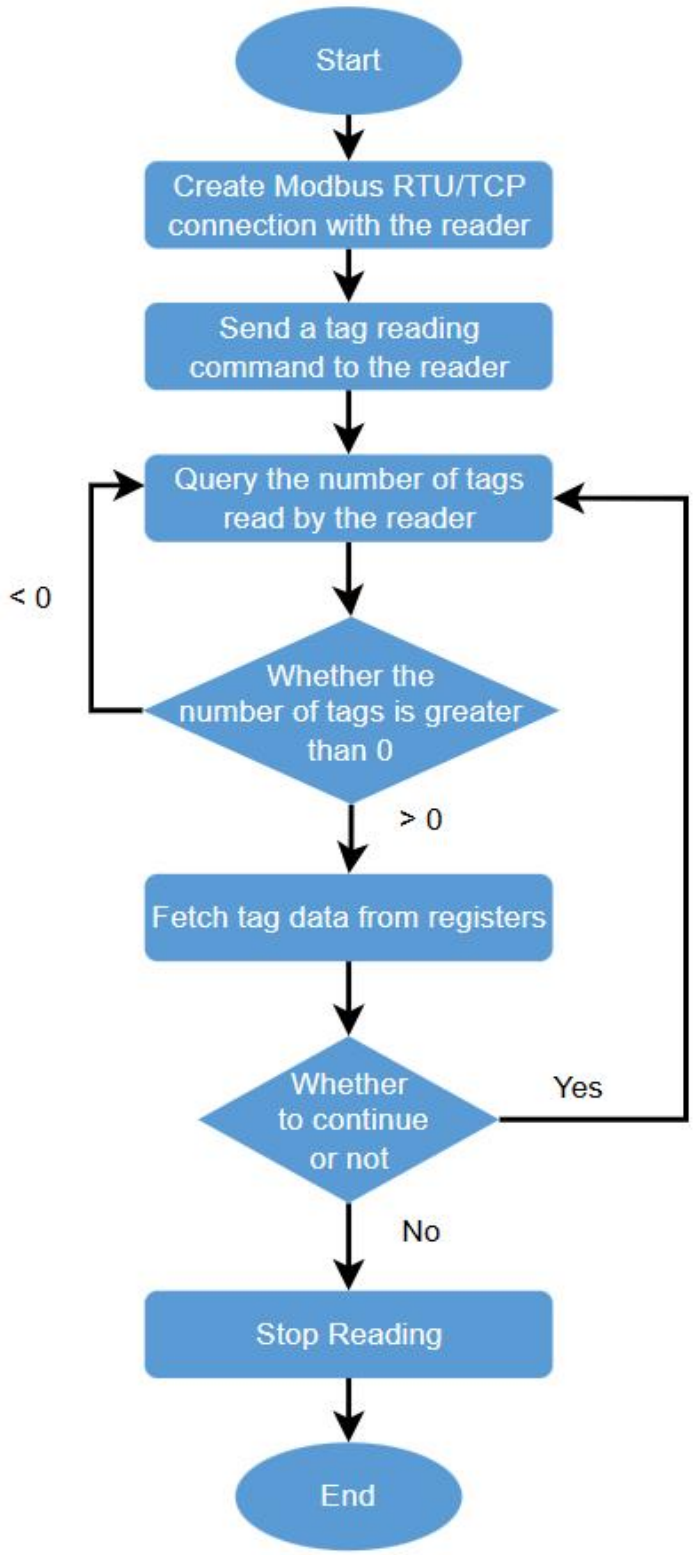
Demo connects to the reader through Modbus

RS485 default baud rate: 115200, RS485 address: 1.
TCP default IP:192.168.1.116:9090, RS485 address: 1.





Tag reading process



High frequency reader Modbus RTU reads UID of the 15693 tag

A 15693 tag, UID is E00401508A82EA89

Reader:1:COM15:115200--Send:01100201000204000100007AC3 //Read the UID of 15693 tags continuously

Command Analysis:

01 //485 address

10 //The function code 0x10 stands for writing multiple registers

0201 //The register start address of the read command, write only

0002 //Number of registers

04 //Data length

0001 //0x0000 represents a single tag read, 0x0001 represents continuous tag reading, after starting continuous tag reading, the reader will continue to read tags until it receives a stop command. At this point, just periodically fetch the tag data from the corresponding registers.

0000 //AFI in 15693 Protocol Mode, 0x00 stands for Select All Tags

7AC3 //Checksum

Reader:1:COM15:115200--Receive:01100201000211B0 //Reader response

Reader:1:COM15:115200--Send:01030210000018477 //Query the number of tags read

Command Analysis:

01 //485 address

03 //Function code

0210 //Query the starting address of the register with the number of tags read. Read only.

0001 //Number of registers

8477 //Checksum

Reader:1:COM15:115200--Receive:010302000179 84 //The reader responds and reads 1 piece of tag data. If the queried tag data is 0, you can continue to query the number of tags read by the reader, and then fetch the tag data from the corresponding registers until there are tags be read.

Reader:1:COM15:115200--Send:01031000001040C6 //Fetch tag data from registers

Command Analysis:

01 //485 address

03 //Function code

1000 //Start address of the register where the tag is stored

0010 //Number of registers, number of 0x0010 registers, that is, 16 registers

40C6 //Checksum

Reader:1:COM15:115200--Receive:0103200013000000008E00401508A82EA8900000000000000000000000000000000000001F34 //Tag data

Data Analysis:

01 //485 address

03 //Function code

20 //Data length

0013 //Number of reads

0000 //RSSI value, high-frequency tags are useless, only applicable to UHF tags

0008 //Tag data length,in bytes

E00401508A82EA89 //UID, which is the tag serial number

00000000000000000000000000000000 //Fill 0 data, useless

1F34 //Checksum

Reader:1:COM15:115200--Send:0103021000018477 //The reader is in the state of continuous tag reading, and continues to query the number of tags read.

Reader:1:COM15:115200--Receive:010302000179 84 //The reader responds and reads 1 piece of tag data.

Reader:1:COM15:115200--Send:01031000001040C6 //Fetch tag data from registers

Reader:1:COM15:115200--Receive:0103200010000000 8E00401508A82EA8900000000000000000000000000000000AF35 //Tag data

Reader:1:COM15:115200--Send:0106020000008872 //Stop reading

Command Analysis:

01 //485 address

06 //Function code 0x06 represents writing a single register.

0200 //Register start address

0000 //Write 0x0000 to stop reading tags.

8872 //Checksum

Reader:1:COM15:115200--Receive:0106020000008872 //Reader response

High frequency reader Modbus TCP reads UID of the 15693 tag

Reader:1:192.168.1.116:9090--Send:000C0000000B011002010002040001000 00 //Read the UID of 15693 tags continuously

Command Analysis:

000C //The sequence number of the message, 1 should be added after each communication to distinguish different communication data messages

0000 //Protocol identifier, 00 00 for the ModbusTCP protocol

000B //Indicates the length of the following data, in bytes

01 //485 address

10 //The function code 0x10 stands for writing multiple registers

0201 //The register start address of the read command, write only

0002 //Number of registers

04 //Data length

0001 //0x0000 represents a single tag read, 0x0001 represents continuous tag reading, after starting continuous tag reading, the reader will continue to read tags until it receives a stop command. At this point, just periodically fetch the tag data from the corresponding registers.

0000 //AFI in 15693 Protocol Mode, 0x00 stands for Select All Tags

Reader:1:192.168.1.116:9090--Receive:000C00000006011002010002 //Reader response

Reader:1:192.168.1.116:9090--Send:000D00000006010302100001 //Query the number of tags read

Command Analysis:

000D //The sequence number of the message, 1 should be added after each communication to distinguish different communication data messages

0000 //Protocol identifier, 00 00 for the ModbusTCP protocol

0006 //Indicates the length of the following data, in bytes

01 //485 address

03 //Function code

0210 //Query the starting address of the register with the number of tags read. Read only.

0001 //Number of registers

Reader:1:192.168.1.116:9090--Receive:000D00000005010302<u>0001</u> //The reader responds and reads 1 piece of tag data. If the queried tag data is 0, you can continue to query the number of tags read by the reader, and then fetch the tag data from the corresponding registers until there are tags be read.

Reader:1:192.168.1.116:9090--Send:000E00000006010310000010 //Fetch tag data from registers

Command Analysis:

000E //The sequence number of the message, 1 should be added after each communication to distinguish different communication data messages

0000 //Protocol identifier, 00 00 for the ModbusTCP protocol

0006 //Indicates the length of the following data, in bytes

01 //485 address

03 //Function code

1000 //Start address of the register where the tag is stored

0010 //Number of registers, number of 0x0010 registers, that is, 16 registers

Reader:1:192.168.1.116:9090--Receive:000E00000023010320001700000008

E00401508A82EA8900000000000000000000000000000000 //Tag data

Data Analysis:

000E //The sequence number of the message, 1 should be added after each communication to distinguish different communication data messages

0000 //Protocol identifier, 00 00 for the ModbusTCP protocol

0023 //Indicates the length of the following data, in bytes

01 //485 address

03 //Function code

20 //Data length

0017 //Number of reads

0000 //RSSI value, high-frequency tags are useless, only applicable to UHF tags

0008 //Tag data length

E00401508A82EA89 //UID, which is the tag serial number

00000000000000000000000000000000 //Fill 0 data, useless

Reader:1:192.168.1.116:9090--Send:000F00000006010302100001 //The reader is in the state of continuous tag reading, and continues to query the number of tags read.

Reader:1:192.168.1.116:9090--Receive:000F000000501030<u>20001</u> //The reader responds and reads 1 piece of tag data.

Reader:1:192.168.1.116:9090--Send:001000000006010310000010 //Fetch tag data from registers

Reader:1:192.168.1.116:9090--Receive:00100000002301032000120000000 8E00401508A82EA8900000000000000000000000000000000 //Tag data

Reader:1:192.168.1.116:9090--Send:001100000006010602000000 //Stop reading

Command Analysis:

0011 //The sequence number of the message, 1 should be added after each communication to distinguish different communication data messages

0000 //Protocol identifier, 00 00 for the ModbusTCP protocol

0006 //Indicates the length of the following data, in bytes

01 //485 address

06 //Function code 0x06 represents writing a single register.

0200 //Register start address

0000 //Write 0x0000 to stop reading tags.

Reader:1:192.168.1.116:9090--Receive:001100000006010602000000 //Reader response

UHF reader Modbus RTU reads 6C tag EPC

Send：01060201<u></u>00011872 //Read the EPC of 6C tags continuously

Command Analysis:

01 //485 地址 address

06 //Function code 0x06 represents writing a single register.

0201 //The register start address of the read command, write only

0001 //0x0000 represents a single tag read, 0x0001 represents continuous tag reading, after starting continuous tag reading, the reader will continue to read tags until it receives a stop command. At this point, just periodically fetch the tag data from the corresponding registers.

1872 //Checksum

Receive：0106020100011872 //Reader response

Send：01030210000184 77 //Query the number of tags read

Command Analysis:

01 //485 address

03 //Function code

0210 //Query the starting address of the register with the number of tags read. Read only.

0001 //Number of registers

8477 //Checksum

Receive：010302000239 85 //The reader responds and reads 2 pieces of tag data. If the queried tag data is 0, you can continue to query the number of tags read by the reader, and then fetch the tag data from the corresponding registers until there are tags be read.

Send：0103100000204 0D2 //Fetch tag data from registers

Command Analysis:

01 //485 address

03 //Function code

1000 //Start address of the register where the tag is stored

0020 //Number of registers, 0x10 registers store one piece of tag data. Two tags were detected, so 0x0020 registers were taken.

40D2 //Checksum

Receive：
0103400002006600 0CE123123456000000000000100000000000000000000000000000000200500006111111111234000000000000000000000000000000000000000000001FB //Tag data

Data Analysis:

01 //485 address

03 //Function code

40 //Indicates the length of the following data, in bytes

0002 0066 000C E123 1234 5600 0000 0000 0001 0000 0000 0000 0000 0000 0000 0000 //The 1st tag data. 16 registers store one piece of data, with a maximum of 8 pieces of data.

//0002 --Number of reads

//0066 --RSSI value

//000C --Tag data length,in bytes

//E123 1234 5600 0000 0000 0001 --EPC value

//0000 0000 0000 0000 0000 0000 0000 --Fill 0 data, useless

0002 0050 0006 1111 1111 1234 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 //The 2nd tag

data.

//0002 --Number of reads

//0050 --RSSI value

//0006 --Tag data length,in bytes

//1111 1111 1234 --EPC value

//0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 --Fill 0 data, useless

01FB //Checksum

Send：0103021000018477 //Query the number of tags read

Receive：0103020003F845 //The reader responds and reads 3 pieces of tag data

Send：0103100000030411E //Fetch tag data from registers, 0x10 registers store one piece of tag data. 3 tags were detected, so 0x0030 registers were taken.

Receive：

0103600001004C000C300833B2DDD9014000000000000000000000000000000000000000000000030066000CE12312345600000000000001000000000000000000000000000001004E000611111111234000000000000000000000000000000000000000003E4 //Tag data

Send：0106020000008872 //Stop reading

Receive：0106020000008872 //Reader response

MODBUS Data Format

MODUBUS function codes only support 0x03, 0x06, 0x10, other function codes return error codes.

The function code is defined as follows:

- 0x03 stands for read register
- 0x06 means to write a single register
- 0x10 means to write multiple registers

RTU Frame structure

Host request command (function code 03):

Address	Function code	Register start address	Number of registers	CRC
1 byte	0x03	2 bytes	2bytes	2bytes

The slave responds to the command:

Address	Function code	Data length	Data content	CRC
1 byte	0x03	1 byte	n bytes	2 bytes

Host request command (function code 06):

Address	Function code	Register address	Data content	CRC
1 byte	0x06	2 bytes	2bytes	2bytes

The slave responds to the command:

Address	Function code	Register address	Data content	CRC
1 byte	0x06	2 bytes	2bytes	2bytes

Host request command (function code 16, 0x10):

Address	Functionc ode	Register start address	Number of r egisters	Data length	Data content	CRC
1byte	0x10	2bytes	2bytes	1byte	n bytes	2bytes

The slave responds to the command:

Address	Function code	Register start address	Number of registers	CRC
1byte	0x10	2bytes	2bytes	2bytes

- Address 0 is the broadcast address, and any device can receive the data.
- The default factory device address is 0x01

TCP Frame structure

In Modbus TCP mode, the 2-byte CRC check digit is deleted and a 6-byte message header is added, as follows.

Transaction ID	Protocol ID	Length
2bytes	2bytes	2bytes

Content	Description
Transaction ID	It can be understood as the serial number of the message. Generally, 1 is added after each communication to distinguish different communication data messages.
Protocol ID	00 00 means Modbus TCP protocol.
Length	Indicates the length of the following data, in bytes.