iManifold

INSTRUCTION

iManifold Robot Control System

Modbus Communication Usage Document

Copyright

Statement

Shenzhen iManifold Robot Technology Ltd. reserves the right to modify the product or its features without prior notice.

Shenzhen iManifold Robot Technology Ltd. does not assume the responsibility for direct or indirect injury or damage caused by improper use of the product.

Contact us

Shenzhen iManifold Robot Technology Ltd.

Address: 410, Building B, Phase II, Huafeng International Robot Industrial Park, Bao'an District, Shenzhen

iManifold Technology (Jiangyin) Ltd.

Address : Room 102, Building 60, Tianan Digital City, No. 55 Changshan Avenue, Jiangyin City, Wuxi City, Jiangsu Province

Contents

1 Modbus introduction to communication protocol

This function provides a modbus_tcp communication method for industrial robots, providing interfaces for system configuration parameters, variable resources, motion control instructions, and robot work status.

Through this function, modbus_tcp communication can be performed on the robot system, thereby achieving real-time monitoring and collection of the robot's operating status, multi device interconnection and collaborative operation, remote operation control, and maintenance needs.

Partial features:

1. Effective in remote mode
2. Update cycle:
    1. The input variable is updated every 100ms
    2. The variable part in hold is updated 100 times every 100ms, in a loop (all variables are updated once every 1000ms)
    3. The coil variable is updated every 100ms
    4. The discrete variable is updated every 100ms
    5. The user trigger command to update automatically through the signal slot
3. Please use the address calculation tool for the detailed address

2 Register address table

Modbus supports four types of registers

Register type	Object type	Access type	Contents
Input_Register Status register	1-Bit	Read only	The I/O system provides this type of data
Coli Coil digital input	1-Bit	Reading and writing	Changing this type of data through an application
Holding_Registe Command register	16-Bit	Reading and writing	Changing this type of data through an application
Discrete_Input Digital input	16-Bit	Read only	The I/O system provides this type of data

2.1 Input_Register

Status Register: Read Only

Name	Data type	Address	Length	Notes
System status	16-bit integer	0x0000	1	0x0000: Standby mode; 0x0001: Enable; 0x0002: In progress; 0x0003: Pause; 0x0A0: Warning; 0x0A01: Servo warning 0x0A02:Running warning 0x0A03: Warning during pause 0x0A04: Alarm;

Servo status	16-bit integer	0x0001	1	0: Servo off; 1: Servo open; 2: Upper servo alarm; 3: Driver error;
Working mode	16-bit integer	0x0002	1	0: Self teaching mode; 1: Playback mode; 2: Remote mode; 3: Remote teaching mode; 4: Remote playback mode;
Posture number	16-bit integer	0x0003	1	Current posture number
External axis number	16-bit integer	0x0004	1	
Playback speed	16-bit integer	0x0005	1	Percentage[1-100]
Teaching speed	16-bit integer	0x0006	1	Percentage[1-100]
Joint position：axis 1	32-bit float	0x0010	2	
Joint position：axis 2	32-bit float	0x0012	2	
Joint position：axis 3	32-bit float	0x0014	2	
Joint position：axis 4	32-bit float	0x0016	2	
Joint position：axis 5	32-bit float	0x0018	2	
Joint position：axis	32-bit float	0x001A	2	

6				
Joint position：axis 7	32-bit float	0x001C	2	
Joint position：axis 8	32-bit float	0x001E	2	
External axle position：axis 1	32-bit float	0x0020	2	
External axle position：axis 2	32-bit float	0x0022	2	
External axle position：axis 3	32-bit float	0x0024	2	
External axle position：axis 4	32-bit float	0x0026	2	
External axle position：axis 5	32-bit float	0x0028	2	
External axle position：axis 6	32-bit float	0x002A	2	
External axle position：axis 7	32-bit float	0x002C	2	
External axle position：axis 8	32-bit float	0x002E	2	
The position of the tool in the robot coordinate system: x	32-bit float	0x0030	2	
The position of the tool in the robot coordinate system: y	32-bit float	0x0032	2	
The position of the tool in the robot	32-bit float	0x0034	2	

coordinate system: z				
The position of the tool in the robot coordinate system: r	32-bit float	0x0036	2	
The position of the tool in the robot coordinate system: p	32-bit float	0x0038	2	
The position of the tool in the robot coordinate system: y	32-bit float	0x003A	2	
Position of the tool in the workpiece coordinate system: x	32-bit float	0x003C	2	
Position of the tool in the workpiece coordinate system: y	32-bit float	0x003E	2	
Position of the tool in the workpiece coordinate system: z	32-bit float	0x0040	2	
Position of the tool in the workpiece coordinate system: r	32-bit float	0x0042	2	
Position of the tool in the workpiece coordinate system: p	32-bit float	0x0044	2	
Position of the tool in the workpiece coordinate system: y	32-bit float	0x0046	2	
Alarm message	128-byte string	0x0050	64	Save alarm information in byte stream
tcs_id	16-bit integer	0x0090	1	
wcs_id	16-bit	0x0091	1	

	integer			
pcs1_id	16-bit integer	0x0092	1	
pcs2_id	16-bit integer	0x0093	1	
Current coordinate system	16-bit integer	0x0094	1	0: Joint coordinate system; 1: Cartesian coordinate system; 2: Workpiece coordinate system 1; 3: Workpiece coordinate system 2; 4: World coordinate system; 5: Tool coordinate system; 6: Expand joint coordinate system; 7: Undefined
Current project name	32-byte string	0x00A0	16	Save alarm information in byte stream
Current program name	32-byte string	0x00C0	16	Save alarm information in byte stream
Current number of running program lines	32-bit integer	0x00E0	2	
Current program execution mode	16-bit integer	0x00E2	1	0: Once; 1: Step forward; 2: Cycle;
AI1	32-bit float	0x00F0	2	Offset Address: 0x0002 * n+0x00EE

……	……	……		
AI32	32-bit float	0x012E	2	

2.2 Holding_Register

Command register (bold table of command words, the rest are parameters)

Attention: Parameters need to be modified first

Name	Data type	Address	Length	Notes
Enable on servo	16-bit integer	0x0000	1	The rising edge is effective, and it automatically resets after taking effect; If it is not currently enabled, the servo will be enabled after the signal is triggered
Servo removal enable	16-bit integer	0x0001	1	The rising edge is effective, and it automatically resets after taking effect; If it is currently enabled, the signal will trigger the servo to enable it
Clear alarm	16-bit integer	0x0002	1	The rising edge is effective, and it automatically resets after taking effect; Attempt to clear the alarm message after the signal is triggered

Playback speed	16-bit integer	0x0003	1	Percentage(1-100),value changes take effect immediately
Teaching speed	16-bit integer	0x0004	1	Percentage(1-100),value changes take effect immediately
Working mode	16-bit integer	0x0005	1	3: Remote jog; 4: Remote playback Value changes take effect immediately
Execution project name	32-byte string	0x0010	16	Save alarm information in byte stream
Execution program name	32-byte string	0x0020	16	Save alarm information in byte stream
Number of executed program lines	16-bit integer	0x0030	1	
Execute program mode	16-bit integer	0x0031	1	0: Once; 1: Step forward; 2: Cycle;
Execution program	16-bit integer	0x0032	1	The rising edge is effective, and it automatically resets after taking effect; Execute the program after the signal is triggered
Pause program	16-bit integer	0x0033	1	The rising edge is effective, and it automatically resets after taking effect; Pause the program after the signal is triggered

Continue to execute	16-bit integer	0x0034	1	The rising edge is effective, and it automatically resets after taking effect; Stop the program after the signal is triggered
Stop the program	16-bit integer	0x0035	1	The rising edge is effective, and it automatically resets after taking effect; Stop the program after the signal is triggered
Axis 1 jog	16-bit integer	0x0040	1	Teach in the current coordinate system 3: Stop moving; 1: Positive movement; 2: Reverse motion Generate effects along the signal, automatically reset for 500ms after activation, then automatically stop moving, and trigger the refresh timer again during this period
……	……	……	……	……
Axis 8 jog	16-bit integer	0x0047	1	Teach in the current coordinate system 3: Stop moving; 1: Positive movement; 2: Reverse motion Generate effects along the signal,

				automatically reset for 500ms after activation, then automatically stop moving, and trigger the refresh timer again during this period
External axis 1 jog	16-bit integer	0x0050	1	Teach in the current coordinate system 3: Stop moving; 1: Positive movement; 2: Reverse motion Generate effects along the signal, automatically reset for 500ms after activation, then automatically stop moving, and trigger the refresh timer again during this period
……	……	……	……	……
External axis 8 jog	16-bit integer	0x0057	1	Teach in the current coordinate system 3: Stop moving; 1: Positive movement; 2: Reverse motion Generate effects along the signal, automatically reset for 500ms after activation, then automatically stop moving, and trigger the refresh timer again during this period
tcs_id	16-bit integer	0x0058	1	Value changes take effect immediately

wcs_id	16-bit integer	0x0059	1	Value changes take effect immediately
pcs1_id	16-bit integer	0x005A	1	Value changes take effect immediately
pcs2_id	16-bit integer	0x005B	1	Value changes take effect immediately
Current coordinate system	16-bit integer	0x005C	1	0: Joint coordinate system; 1: Cartesian coordinate system; 2: Workpiece coordinate system 1; 3: Workpiece coordinate system 2; 4: World coordinate system; 5: Tool coordinate system; 6: Expand joint coordinate system; 7: Undefined Value changes take effect immediately
AO1	32-bit float	0x0060	1	Value changes take effect immediately
……	……	……	……	……
AO32	32-bit float	0x0080	1	Value changes take effect immediately
Global position variable 0001: Coordinate System type	16-bit integer	0x0100	1	The value change takes effect immediately, offset address: 0x0020 * n + 0x00E0

				Refer to the coordinate system corresponding to the current coordinate system values above
Global positional variable 0001: Coordinate system number	16-bit integer	0x0101	1	The value change takes effect immediately, offset address: 0x0020 * n + 0x00E1
Global positional variable 0001: Joint 1	32-bit float	0x0102	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00E2
Global positional variable 0001: Joint 2	32-bit float	0x0104	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00E4
Global positional variable 0001: Joint 3	32-bit float	0x0106	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00E6
Global positional variable 0001: Joint 4	32-bit float	0x0108	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00E8
Global positional variable 0001: Joint 5	32-bit float	0x010A	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00EA
Global positional variable 0001: Joint 6	32-bit float	0x010C	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00EC
Global positional variable 0001:	32-bit float	0x010E	2	The value change takes effect immediately,

Joint 7				offset address: 0x0020 * n + 0x00EE
Global positional variable 0001: Joint 8	32-bit float	0x0110	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00F0
Global positional variable 0001: Cartesian X	32-bit float	0x0112	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00F2
Global positional variable 0001: Cartesian Y	32-bit float	0x0114	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00F4
Global positional variable 0001: Cartesian Z	32-bit float	0x0116	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00F6
Global positional variable 0001: Cartesian R	32-bit float	0x0118	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00F8
Global positional variable 0001: Cartesian P	32-bit float	0x011A	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00FA
Global positional variable 0001: Cartesian Y	32-bit float	0x011C	2	The value change takes effect immediately, offset address: 0x0020 * n + 0x00FC
……	……	……	……	……
Global positional variable 1000: Coordinate system	16-bit integer	0x7DE0	1	The value change takes effect immediately

type				
Global positional variable 1000: Coordinate system number	16-bit integer	0x7DE1	1	The value change takes effect immediately
Global positional variable 1000: Joint 1	32-bit float	0x7DE2	2	The value change takes effect immediately
Global positional variable 1000: Joint 2	32-bit float	0x7DE4	2	The value change takes effect immediately
Global positional variable 1000: Joint 3	32-bit float	0x7DE6	2	The value change takes effect immediately
Global positional variable 1000: Joint 4	32-bit float	0x7DE8	2	The value change takes effect immediately
Global positional variable 1000: Joint 5	32-bit float	0x7DEA	2	The value change takes effect immediately
Global positional variable 1000: Joint 6	32-bit float	0x7DEC	2	The value change takes effect immediately
Global positional variable 1000: Joint 7	32-bit float	0x7DEE	2	The value change takes effect immediately
Global positional variable 1000: Joint 8	32-bit float	0x7DF0	2	The value change takes effect immediately
Global positional variable 1000: Cartesian X	32-bit float	0x7DF2	2	The value change takes effect immediately
Global positional variable 1000: Cartesian Y	32-bit float	0x7DF4	2	The value change takes effect immediately

Global positional variable 1000: Cartesian Z	32-bit float	0x7DF6	2	The value change takes effect immediately
Global positional variable 1000: Cartesian R	32-bit float	0x7DF8	2	The value change takes effect immediately
Global positional variable 1000: Cartesian P	32-bit float	0x7DFA	2	The value change takes effect immediately
Global positional variable 1000: Cartesian Y	32-bit float	0x7DFC	2	The value change takes effect immediately
Global numerical variable 0001	32-bit float	0x7E10	2	The value change takes effect immediately, offset address: 0x0002 * n + 0x7E0E
……	……	……	……	……
Global numerical variable 0300	32-bit float	0x8066	2	The value change takes effect immediately

2.3 Coil

Digital output: 4 groups, each with 16 pieces, totaling 48 pieces

Name	Data type	Address	Notes
DO1.1~DO1.16	1-bit	0x0000~0x000F	The value change takes effect immediately
DO2.1~DO2.16	1-bit	0x0010~0x001F	The value change takes effect immediately
DO3.1~DO3.16	1-bit	0x0020~0x002F	The value change takes effect

			immediately
DO4.1~DO4.16	1-bit	0x0030~0x003F	The value change takes effect immediately
DO5.1~DO5.16	1-bit	0x0040~0x004F	The value change takes effect immediately
DO6.1~DO6.16	1-bit	0x0050~0x005F	The value change takes effect immediately
DO7.1~DO7.16	1-bit	0x0060~0x006F	The value change takes effect immediately
DO8.1~DO8.16	1-bit	0x0070~0x007F	The value change takes effect immediately

2.4 Discrete_Input read only

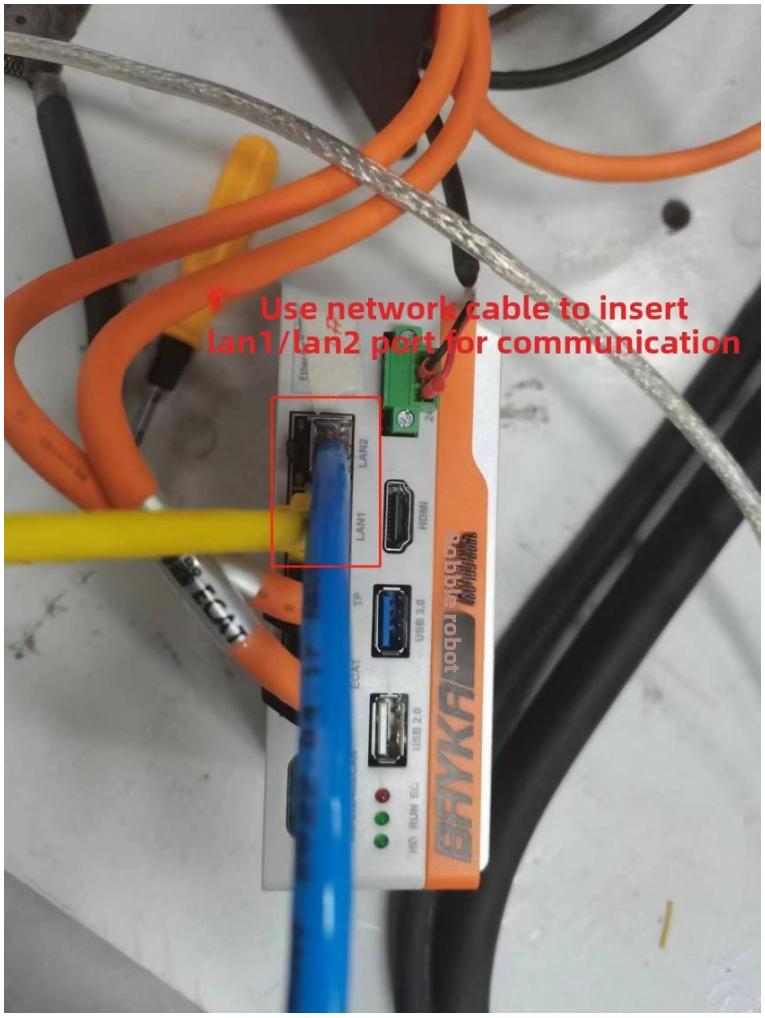
Digital output: 4 groups, each with 16 pieces, totaling 48 pieces

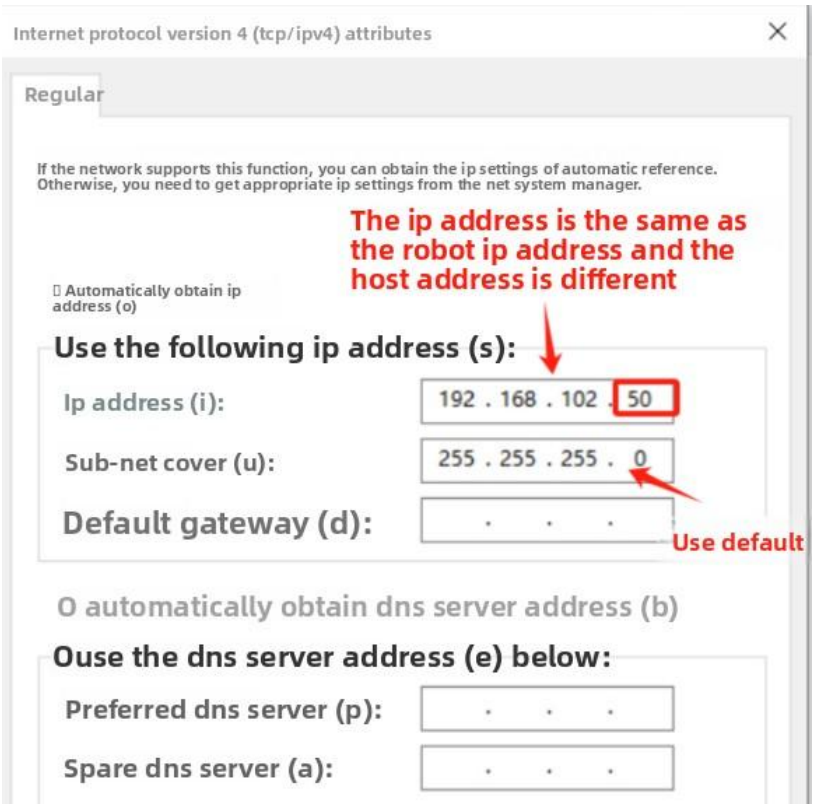
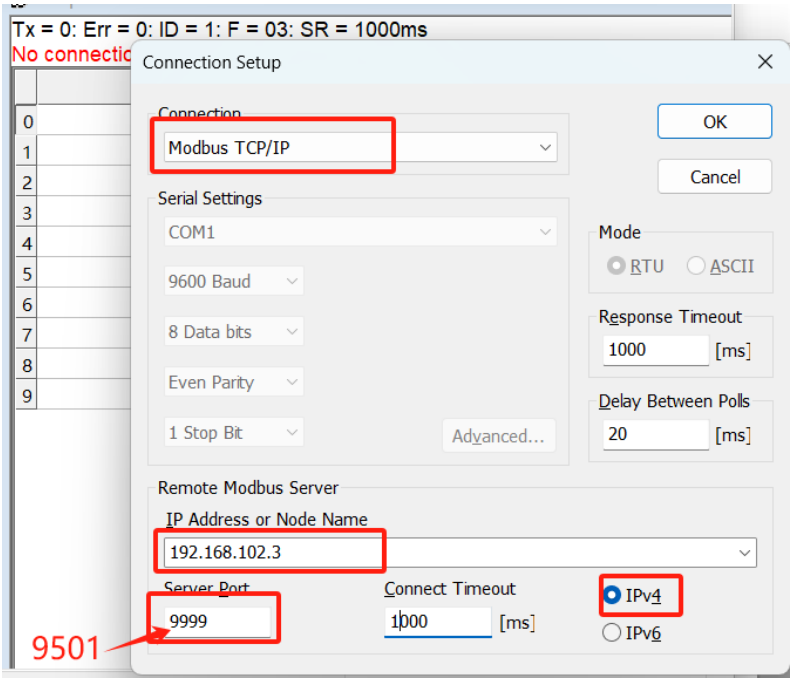
Name	Data type	Address	Notes
DI1.1~DI1.16	1-bit	0x0000~0x000F	
DI2.1~DI2.16	1-bit	0x0010~0x001F	
DI3.1~DI3.16	1-bit	0x0020~0x002F	
DI4.1~DI4.16	1-bit	0x0030~0x003F	
DI5.1~DI5.16	1-bit	0x0040~0x004F	
DI6.1~DI6.16	1-bit	0x0050~0x005F	
DI7.1~DI7.16	1-bit	0x0060~0x006F	
DI8.1~DI8.16	1-bit	0x0070~0x007F	

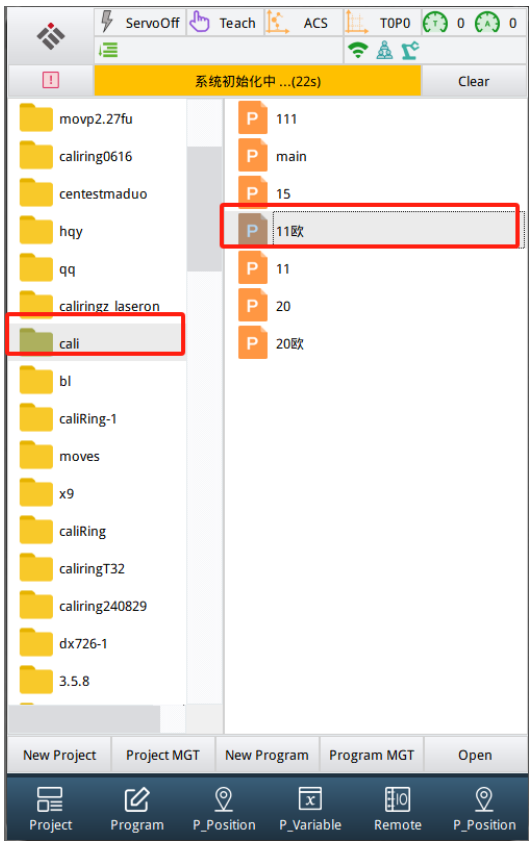
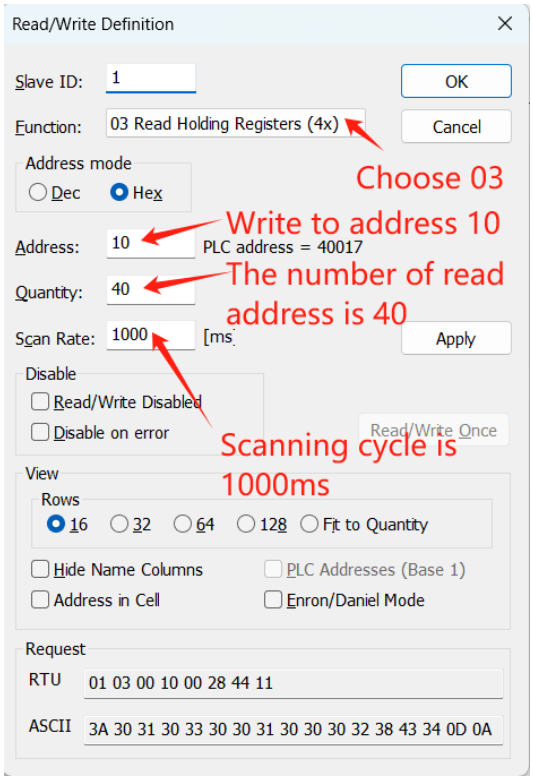
3 Communication instance

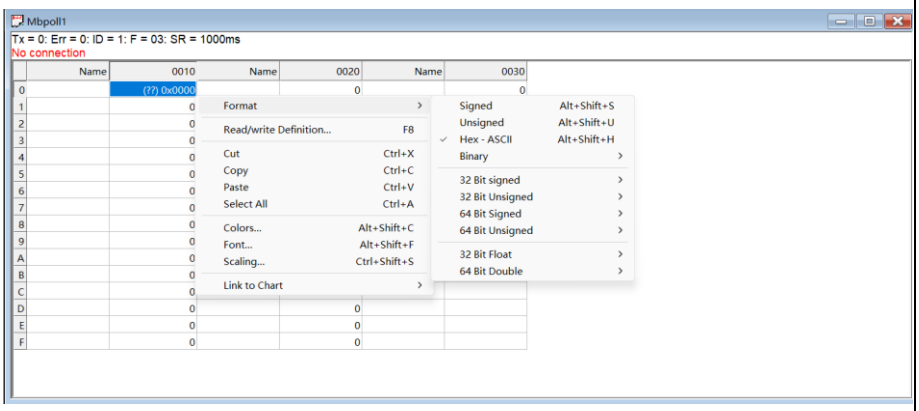
Modbus has numerous clients, and in this document, the Modbus poll tool is used to demonstrate how to communicate with robot systems through Moubus. Modbus communication only occurs when the robot is in remote mode.

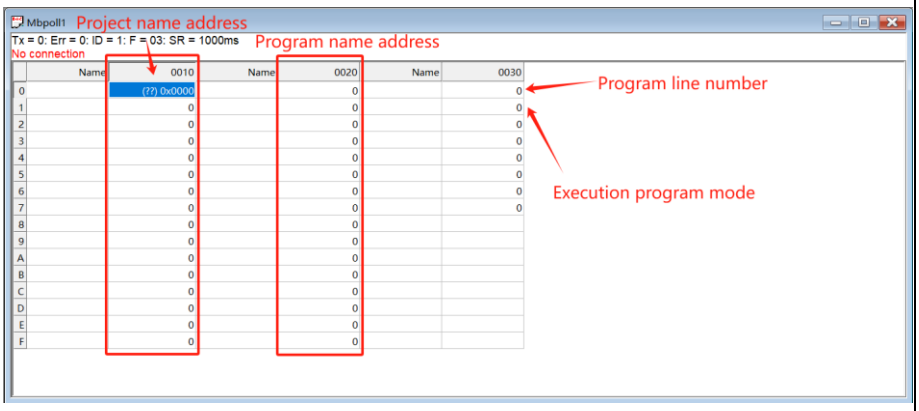
3.1 Operation instance

Project	Operation steps
Connection controller	 Connect Lan1 or Lan2 to the computer via Ethernet cable

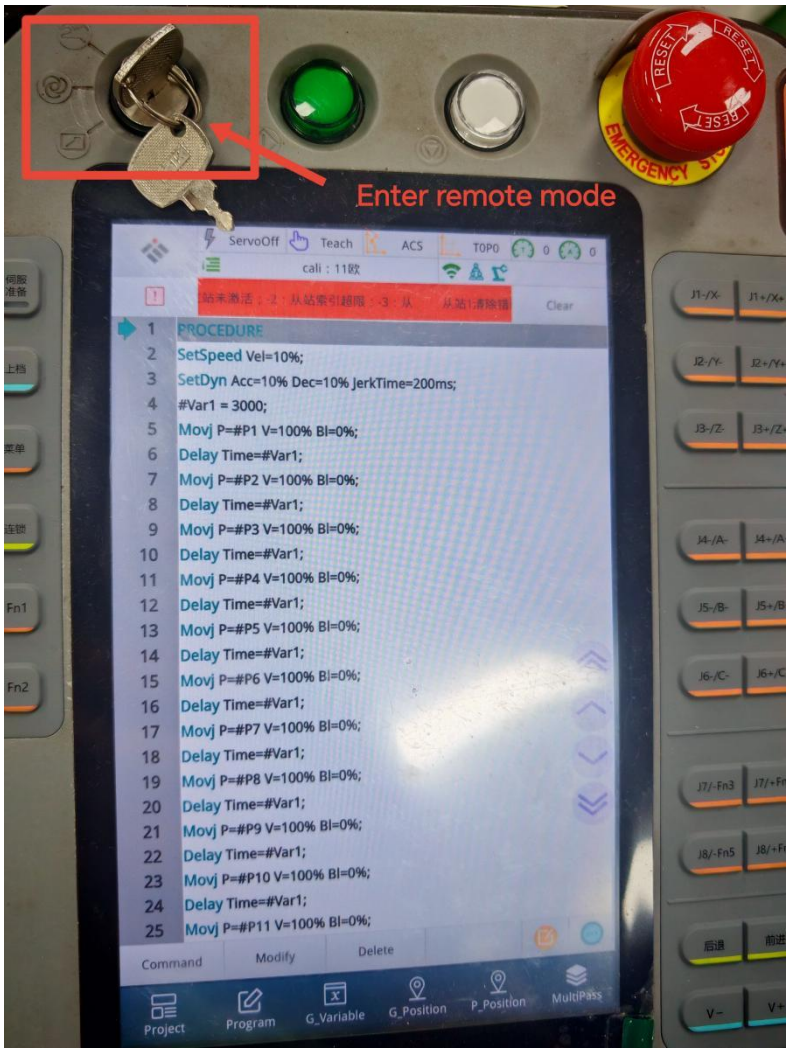
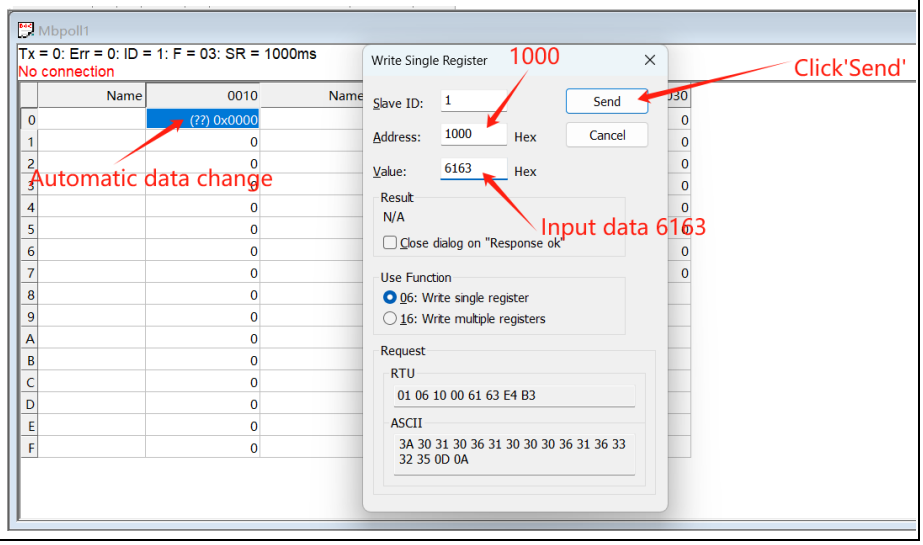
Setting network	 Need to keep the computer IP consistent with the controller IP
Connections setting up	 After opening Modbus Poll, open the connection interface using Modbus TCP/IP protocol; The IP address should be the same as the robot's IP address; The server port is 9501; The protocol version is IPv4; Other defaults.

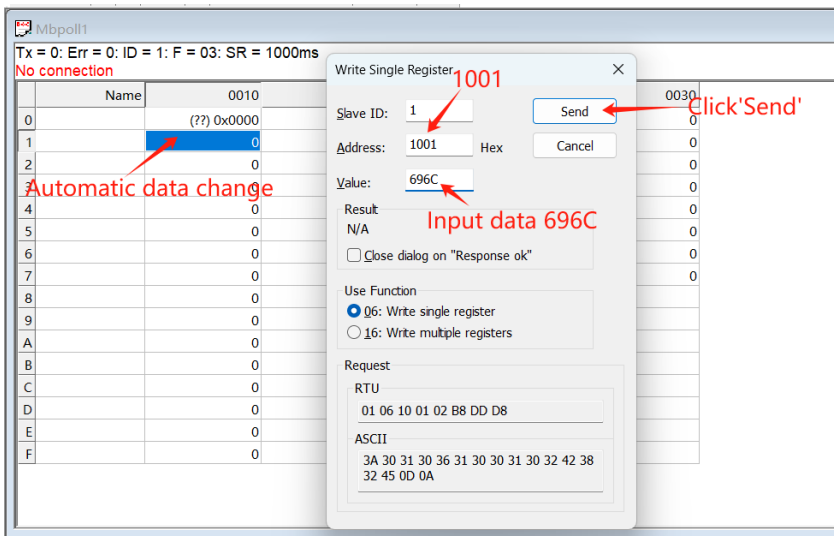
Project instance	 Demo: Start a project named Cali and a program named 11 ohms using Modbus poll. The execution mode of the program is multiple loops, starting from line 11 of the program
Set read/write definition	

<table>
<tr><td></td><td>According to the different function codes used, it is necessary to find the corresponding function address table for correspondence; The default address format is decimal, which must be changed to hexadecimal, otherwise the data address will be incorrect;</td></tr>
<tr><td>Menu address switching</td><td>
<table>
<tr><td>Execution engineering name</td><td>128-byte string</td><td>0x1000~0x103F</td><td>Save the alarm information by the bytes</td></tr>
<tr><td>Execution program name</td><td>128-byte string</td><td>0x1040~0x107F</td><td>Save the alarm information by the bytes</td></tr>
<tr><td>Number of implementation procedures</td><td>32-bit integer</td><td>0x1080~0x1081</td><td></td></tr>
<tr><td>Execution program mode</td><td>16-bit integer</td><td>0x1082</td><td>0: once;
1: step;
2: circulation;</td></tr>
</table>
According to the functional address table, find the execution project name, execution program name, number of execution program lines, and execution mode address. Different addresses require different data types to be input, and data types need to be converted. Need to use a conversion tool to convert to hexadecimal.
Conversion tool（http://www.kjson.com/transform/ox2str/）</td></tr>
<tr><td>Format selection</td><td> 

Select addresses 1000 and 1040, and change the address format to "hexadecimal ASCII"</td></tr>
<tr><td>Data address</td><td> 

According to the address table, find the 0x0030 execution program line</td></tr>
</table>

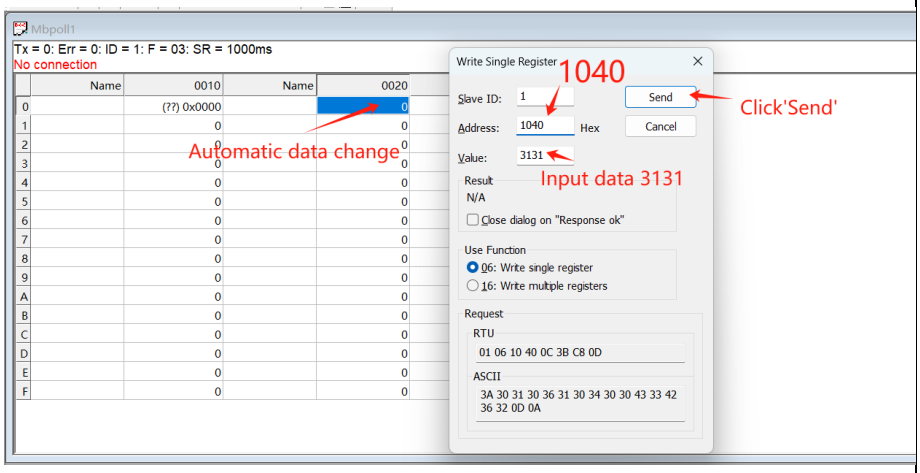
	number and 0x0031 execution program mode address to set
Switch mode	
Write project name	

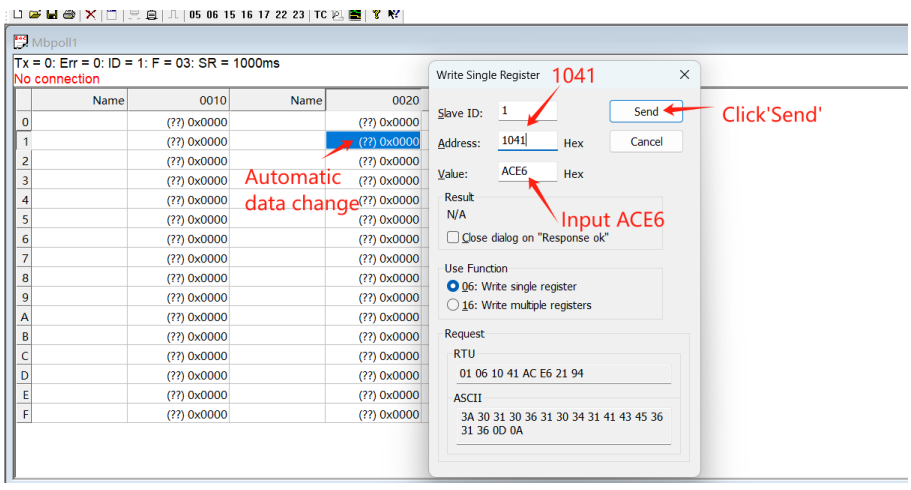


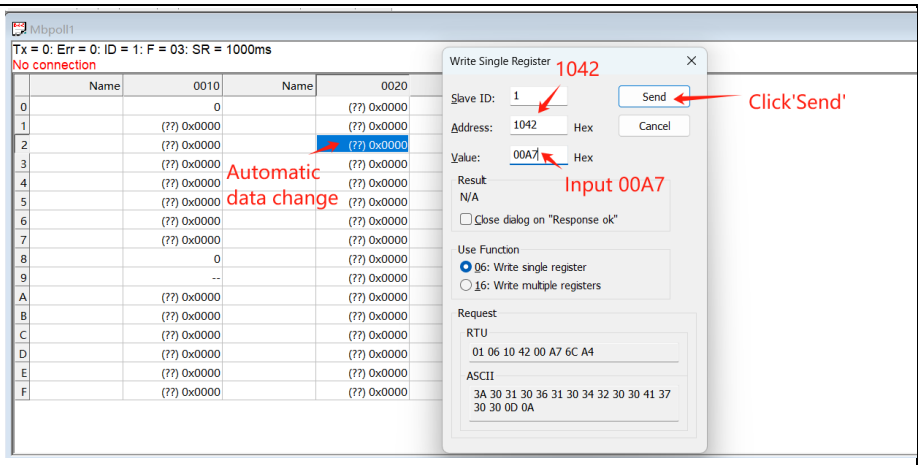
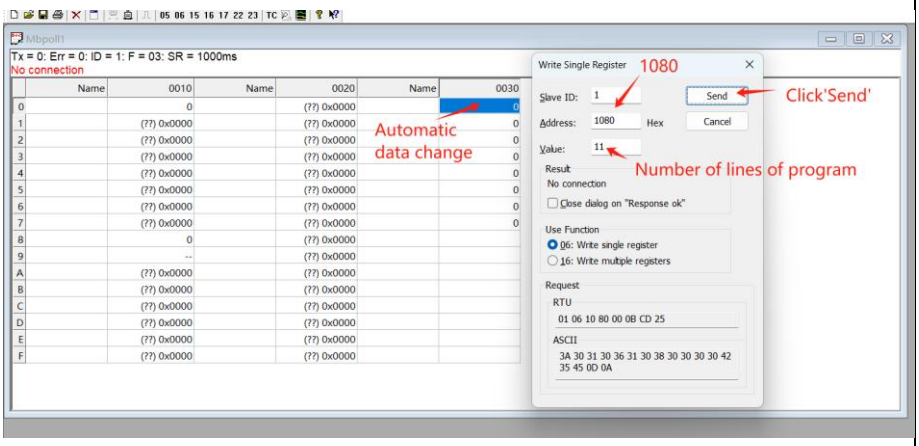
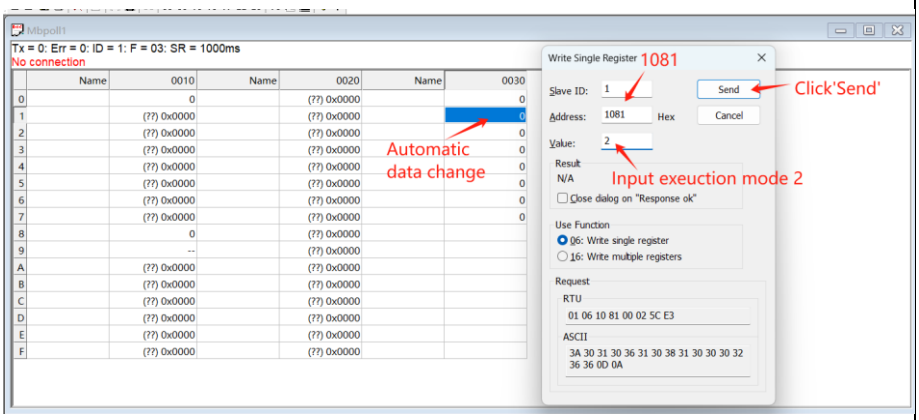
The data transmission format for this feature uses little_dedian_stwap, and attention should be paid to byte order. Convert the project name 'cali' to hexadecimal code using a conversion tool.

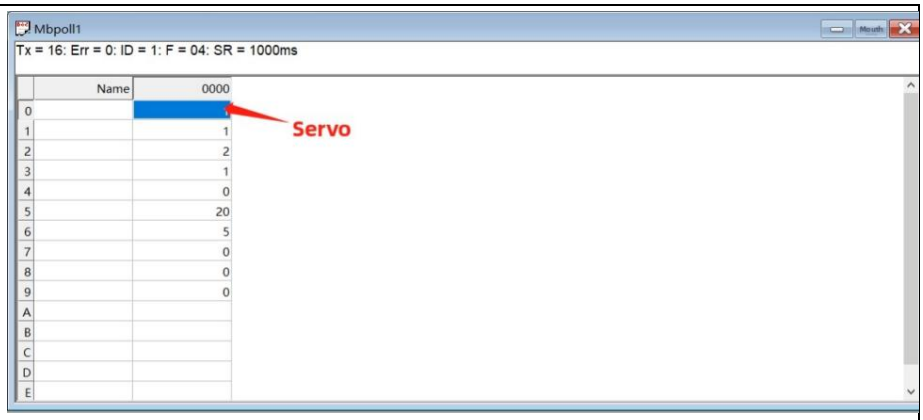
Conversion tool（http://www.kjson.com/transform/ox2str/）

Write program name





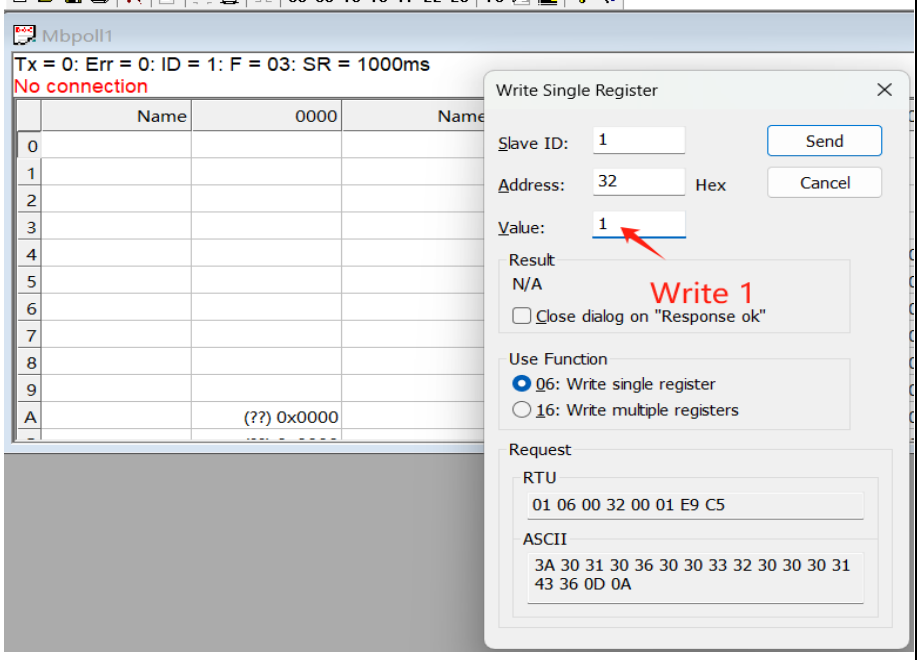
<table>
<tr><td></td><td> 
</td></tr>
<tr><td>Write the number of lines to execute the program</td><td> 
</td></tr>
<tr><td>Write and execute program mode</td><td> 
</td></tr>
<tr><td>Execute servo</td><td><table>
<tr><td>Servo enable</td><td>16-bit integer</td><td>0x0000</td><td>1</td><td>The upper rise edge is effective and the slot will automatically recover after effect; if the current failure is not available, the signal will trigger the rear servo enable</td></tr>
<tr><td>Servo to enable</td><td>16-bit integer</td><td>0x0001</td><td>1</td><td>The upper rise edge is effective and the slot will automatically recover after effect; if the current energy has been used, the signal will be connected and the servo can be used after being sent</td></tr>
</table></td></tr>
</table>



According to the address table of the Holding Register command register function, find the upper servo address for remote upper servo

Execute program

Execution program mode	16-bit integer	0x1082	0: once; 1: step; 2: circulation;
Execution program	16-bit integer	0x1083	It is effective and automatically resumes the slot after effect; the execution process after this signal is triggered
Pause program	16-bit integer	0x1084	The upper rise edge is effective and the slot will automatically recover after effect; the pause program after this signal is triggered
Continue to execute	16-bit integer	0x1085	The upper rise edge is effective and the slot will automatically recover after effect; the program will be stopped after this signal is triggered
Stop program	16-bit integer	0x1086	The upper rise edge is effective and the slot will automatically recover after effect; the program will be stopped after this signal is triggered



According to the address table, find the address 0x0032, write the value 1, and the program will execute

4 Common problems

4.1 Conversion of data types

Modbus poll does not support strings, ACSII code needs to be converted to hexadecimal.

Chinese characters and Chinese symbols require the use of string to hexadecimal conversion tools.

Example: "Circle" → "e59c88"

Conversion tool（http://www.kjson.com/transform/ox2str/）

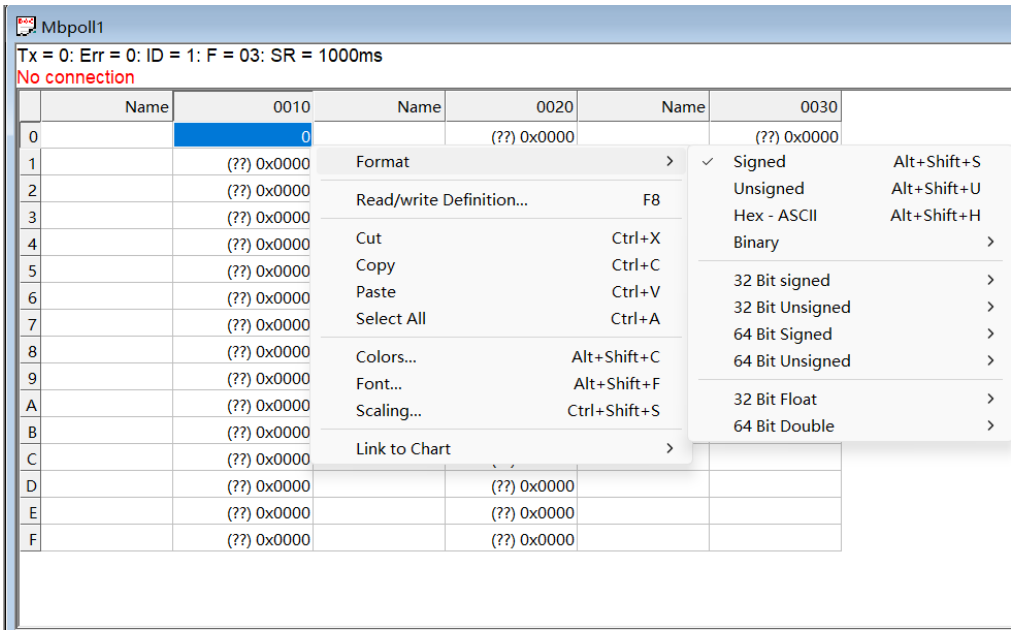
ACSII code requires the use of ASCII to hexadecimal conversion tools.

Example: "Cali" → "63616C69"

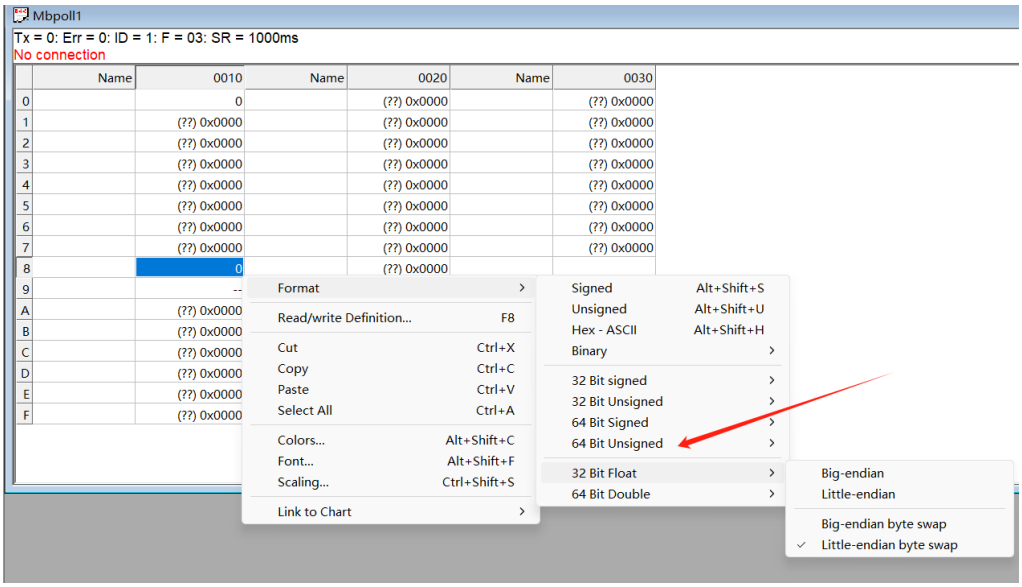
4.2 Data type

The data transmission format for this feature uses little_dedian_stwap. The data types of the menu are 16 bit integer, 32-bit float, 32-bit string, and 128 byte string. When viewing data, it is important to pay attention to the data types.

Modbus Poll defaults to "16 bit integer (signed)".



If the data type is "32-bit float", the format needs to be set to "32-bit floating point number" → "small byte swap" to view the data.



4.3 Numerical input

The address value input needs to be checked in the remarks column of the function code, and the valid command word for the rising edge can only be entered as 1.

The format of string data input to the address needs to be changed to "hexadecimal ASCII", and then double-click the value that needs to be changed to modify it (using the "write single register" function defaults to decimal input and cannot write hexadecimal data).

When entering hexadecimal code, note that 4 words form 1 group and pay attention to byte order. When the data exceeds 4 bits, multiple addresses need to be entered.

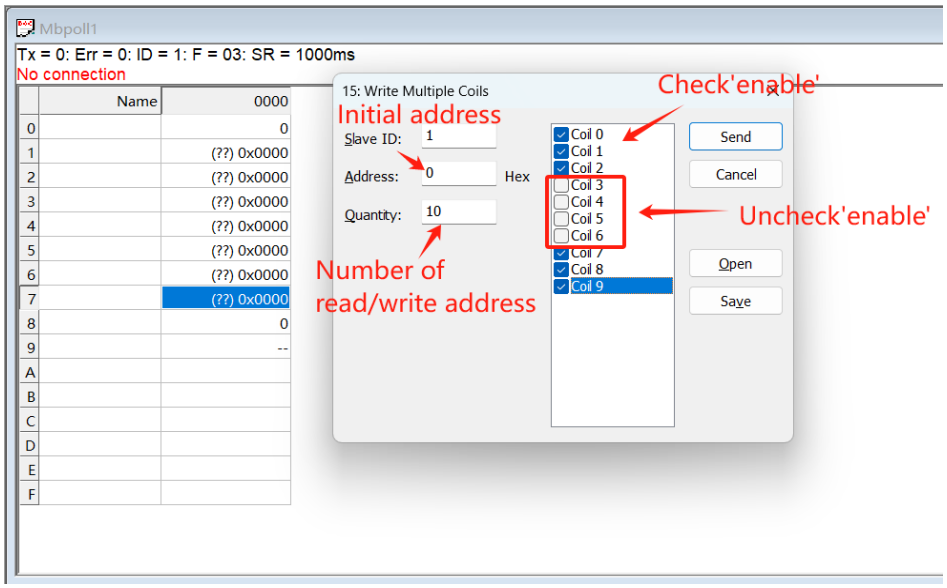
Example: "Circle" → ". e59c88" When entering an address, write "0x9ce5" or "0x0088"

4.4 Definition of Read/Write

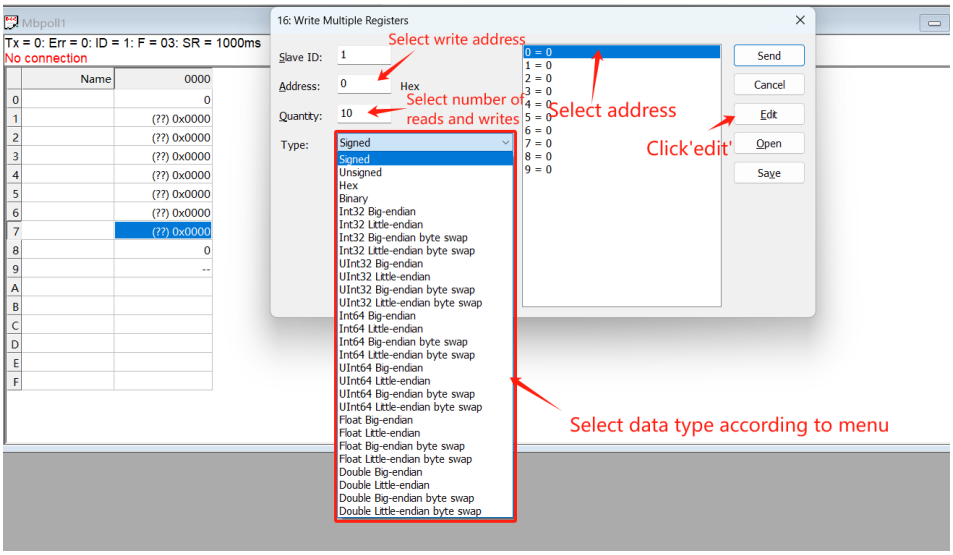
In the definition of read/write, the slave station is fixed at 1; There are a total of 4 function codes: 01, 02, 03, 04; The default address format is decimal, which needs to be changed to hexadecimal, otherwise the data address will be incorrect; Default scanning rate is 1000ms

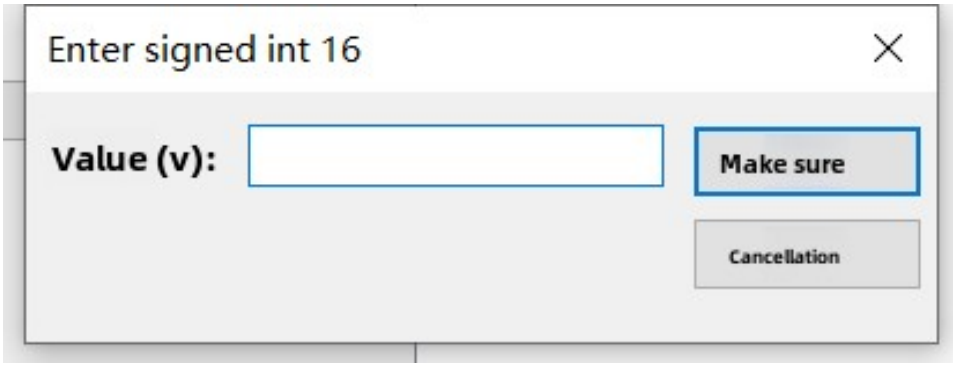
4.5 Writing to multiple coils/registers

Modbus supports multiple coil/register inputs, for example: enabling multiple coils



Example: Writing multiple registers





4.6 Modbus poll error

No connection	The normal unconnected state can be eliminated by connecting when the normal COM port is not occupied.
Timeout	All commands issued by software without a response from the slave device will display Timeout. There are many possibilities for the slave device not to reply, such as: 1. Connection configuration error, the host's baud rate, Slave ID and other information do not correspond to the slave device, and the slave will not respond. 2. The line is abnormal, and there is an abnormality in the communication line before the computer follows the device, and it is also unable to receive a reply normally. 3. The slave device does not respond to abnormal parsing, which can be found in the detailed explanation of the Modbus protocol.
Write error/Read error	Using a USB to 485 conversion tool, swapping the A and B wires of 485 may result in this situation. If a USB to 232 or TTL converter is used, short circuiting Tx and Rx will result in this situation. When connecting with a network cable, disconnecting the cable can cause this situation. During the sending process, similar errors may occur when data conflicts are

	received on the bus.
Checksum Error	The CRC check returned by the slave device is incorrect. There is interference on the communication bus. The checksum, data bits, and stop bits in the connection configuration are configured incorrectly.
Insufficient bytes received	If the number of received bytes is incomplete, it may be due to the circuit or some reason that the length of the returned instruction does not match the theoretical length of the returned instruction, which will result in some errors.
Illegal Function	Function code exception, usually when the accessed slave device does not have an operable function code, it will return the 01 exception code that does not exist for this function code. When the software receives this instruction, it will report this error.
Illegal Data Address	Address exception, usually when the accessed slave device does not have the register/coil address to be read, it will return an 02 exception code indicating that this address does not exist. When the software receives this instruction, it will report this error.
Illegal Data Value	Data exception, usually refers to the current data to be read/written. If the slave device does not allow the operation of data at this address, it will return a 03 exception code indicating that the data cannot be operated. When

	the software receives this instruction, it will report this error.
Slave Device Busy	The slave is busy, usually when the slave device replies with an exception code of 06, the software will report this error when receiving this command.
Response Error	Response exception is usually reported when the slave device replies with some unrecognized commands. Another issue is whether it is a bug in the software. Generally, the broadcast command (0 address) cannot read data and can only broadcast write. However, this software can be set to read with broadcast commands. When this is set, this "Response Error" exception will appear.