



User Manual

Instructions for Modbus TCP Slave



User Manual

Instructions for Modbus TCP Slave

V1.0

The information in this Manual must not be considered as a commitment of Agilebot and may be changed without prior notice. Agilebot assumes no responsibility for errors (if any) in this Manual.

Except as expressly specified, nothing in this Manual shall be construed as any warranty or guarantee made by Agilebot for personal loss, property damage or specific applicability.

Agilebot assumes no responsibility for any accidents or indirect injuries caused by using this Manual or the product described therein.

This Manual and any parts thereof must not be reproduced or duplicated without written permission from Agilebot.

Additional copies of this Manual may be obtained from Agilebot.

The original language of this Publication is Chinese.

International standard units are adopted in this publication. GB means Chinese national standard.

© Copyright, 2022 Agilebot Robotics Co., Ltd. All rights reserved!

Agilebot Robotics Co., Ltd.

Shanghai, China

Revision

Ver.	Date	Status
V1.0	Dec 30, 2023	Release

Table of Contents

Safety instructions	6
1 Introduction	12
2 Hardware overview	19
2.1 Communication port of controller.....	19
2.2 Cable connection	20
2.2.1 Selection of bus cables	20
3 Software overview	20
3.1 Information of slave	20
4 Configuration of slaves	21
4.1 Configuration steps	21
4.2 Configure internal slaves.	22
5 Configuration of I/O mapping	25
5.1 Configuration of I/O mapping	25
6 Configuration and monitoring of Modbus TCP Registers	29
6.1 Configuration of Modbus TCP Registers.....	29
6.2 Monitoring of Register status	29
7 System parameters	31
7.1 Industrial network	31
8 Faults	32

Safety instructions

It is necessary to read and understand the contents described in this chapter before using robots.

In this Manual, the robot system refers to an integrated system integrating the robot and its controller, teach pendant, cables, software and other accessories. So, it is required to fully consider the safety precautions of the user and the system.

Nobody is allowed to modify the robot without authorization from Agilebot Robotics Co., Ltd. Agilebot Robotics Co., Ltd. shall assume no responsibility for any damage to the robot or its components due to the use of any other components (software, tools, etc.) not provided by Agilebot.

Agilebot Robotics Co., Ltd. assumes no responsibility for any consequences caused by misuse of the robot. The misuse includes:

- Use the robot beyond the specified parameter range
- Use it as a carrier for humans or animals
- Use it as a climbing tool
- Use it in explosive environments
- Use it without safety protection

Besides safety precautions in this chapter, this Manual contains other safety instructions, which must be followed as well.

For safety issues uncovered in this Manual, please refer to the Safety Manual.

Definition of user

The operators are defined as follows:

- Operator
 - Perform power-on/off operation on the robot.
 - Start the robot program from the panel board.
- Robot engineer
 - Operate the robot.
 - Perform teaching and programming debugging of the robot within the safety fence.
- Maintenance Engineer
 - Operate the robot.
 - Perform teaching of the robot within the safety fence.
 - Carry out maintenance (repair, adjustment, replacement) operations on the robot.

The "Operator" is not allowed to enter the safety fence.

The "Robot Engineer" and "Maintenance Engineer" can carry out operations within the safety fence.

The operations within the safety fence include handling, setting, teaching, adjustment, maintenance, etc.

To carry out the operations within the safety fence, it is necessary to receive professional training on the robot.

When operating, programming and maintaining the robot, the operator, programmer and maintenance engineer must give a safety warning and wear at least the following protective articles.

- Work clothes suitable for operations
- Safety shoes
- Safety helmets

System permissions for operators

Operator

Operator's authorities include:

- 1) Turn on/off the robot.
- 2) Use the operating terminal to teach the robot; select, debug, run, start, pause and abort the programs.
- 3) Switch the currently loaded TF/UF and modify overall velocity parameters through the above status bar on the screen.
- 4) Allow operations, e.g. moving to the target point.
- 5) Review alarms and reset regular alarms.
- 6) Perform the operations of I/O status interface and register interface.

Robot engineer

The authority of the robot engineer includes:

- 1) All authorities of the operators
- 2) Setting of robot's zero point, setting of soft limit, establishment and editing of coordinate system
- 3) I/O configuration and management
- 4) Communication configuration
- 5) Creation, editing, revision, deletion and other robot program management functions
- 6) Creation and setting of various registers
- 7) Management function of robot program attributes
- 8) Setting of program launch mode
- 9) Backup and loading of files
- 10) Setting of controller IP
- 11) Setting of system time

Administrator (Admin)

The authorities of administrator include:




- 1) All authorities of the operator and robot engineer
- 2) Software installation and upgrading
- 3) Management of programmer roles, which can be added, deleted or edited

Definition of safety records

This Manual includes safety warnings to ensure personal safety of the users and avoid any damage to the machine tool and describes them with "Danger" and "Warning" in the main text based on their importance in safety.

In addition, relevant additional descriptions are described as "Caution".

Before use, the user must thoroughly read the precautions described in "Danger", "Warning" and "Caution".

Identification	Definition
 Danger	It indicates dangerous situations possibly resulting in serious injury or death to the user during incorrect operation.
 Warning	It indicates dangerous situations possibly resulting in mild or moderate personal injury or property damage during incorrect operation.
 Caution	It provides additional descriptions outside the scope of danger or warning.

Please read this Manual carefully and keep it secure for easy reference at any time.

Safety of operator

When the robot operates automatically, it is first necessary to ensure the safety of the operator. It is quite dangerous to enter the motion range of the robot during its automatic operation. Measures should be taken to prevent the operator from entering the motion range of the robot.

General precautions are listed below. Please take appropriate measures to ensure the safety of the operator.

1. All operators using the robot system should pass the training courses provided by Agilebot Robotics Co., Ltd.
2. During the operation, it is possible that the robot is waiting for a start signal and is about to start even if it appears to have stopped. Even in such cases, it should be considered that the robot is in motion.
3. Set peripheral devices outside the robot's range of motion as much as possible.
4. Arrange locks as needed to prevent personnel other than operators from turning on the power supply of the robot.
5. When conducting individual debugging of peripheral devices, it is important to disconnect the power supply of the robot in advance.
6. When handling or mounting the robot, make sure to follow the correct method shown by Agilebot Robotics Co., Ltd. The operation in the wrong way may lead to overturning of the robot, causing injury to the operator.
7. After mounting, make sure to operate the robot at a low speed for the first time. Then, gradually raise the speed and confirm if there are any abnormalities.
8. When using the robot for operation, it is important to confirm that there are no persons inside the safety fence in advance. Meanwhile, check for potential hazards and make sure to eliminate potential hazards (if any) before operation.
9. Do not operate the robot in the following situations. Otherwise, it may pose an adverse effect to the robot and also cause serious injuries to the operator.
 - 1) Flammable environments
 - 2) Explosive environments
 - 3) High-radiation environment
 - 4) Water or high humidity environment
- 5) When connecting various stop signals of peripheral devices and the robot, make sure to confirm the stop operation to avoid incorrect connections.

Safety warning label





Both the robot and the controller bear several safety and information labels, which contain important information related to the product. This information is very useful for all persons operating the robot system, e.g. during mounting, maintenance or operation.

The safety labels are only graphical and applicable to all languages.



Caution

It is required to observe the safety and health signs on the product label. In addition, it is also necessary to comply with the supplementary safety information provided by the system builder or integrator.

Sign	Description
	Warning - electric shock
	Warning - hands pinching
	Beware of burns due to high temperature.
	Grounding

1 Introduction

What's ModbusTCP?

Overview

MODBUS/TCP is a simple derivative product of the MODBUS series communication protocol used by independent manufacturers to manage and control automation equipment. Obviously, it covers the purpose of MODBUS messages in "Intranet" and "Internet" environments based on TCP/IP protocol.

Protocol communication technology

The master-slave communication technology is adopted in the Modbus protocol. Namely, the master device actively queries and operates the slave one. Generally, the protocol used by the master device is referred to as Modbus Master, while that used by the slave device is called Modbus Slave. The Modbus Master is active, while the Modbus Slave is passive.

Connection by communication protocol

The user must establish a connection with the device in order to exchange information with it.

Its communication follows the following process:

- The master device sends a request to the slave device.
- The slave device analyzes and processes the requests from the master device, and then send results to the latter.
- In case of any error, the slave device may return an abnormal function code.

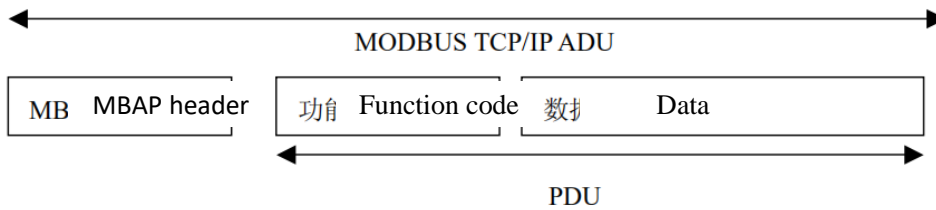
Data transmission mode

The data transmission of modbus is defined as reading/writing to the following four storage blocks:

- The operation unit of coils is a 1-bit switch value, which is the output bit of the PLC and can be read and written in Modbus.
- The operation unit of discrete inputs is a 1-bit switch value, which is the input bit of the PLC and can be only read in Modbus.
- The operation unit of input registers is a 16-bit digit (two bytes), They are registers, which can only be changed from analog inputs in PLC and are read-only in Modbus.
- The operation unit of holding registers is a 16-bit digit (two bytes), They are registers used to output analog signals in PLC and are read and written in Modbus.

Structure of Modbus TCP message

The message of Modbus TCP consists of MBAP + PDU.



MBAP header

The MBAP header is 7 bytes long and consists of:

Domain	Length	
Transaction identifier	2 bytes	The identifier for MODBUS request/response transaction is mainly used to enable the master device to understand the request for the response received.
Protocol Identifier	2 bytes	For the MODBUS protocol, it is always 0 here.
Length	2 bytes	Number of bytes below, namely number of bytes in the complete message minus 6
Unit identifier	1 bytes	The identification code of the remote slave connected on a serial link or other bus, namely identifier of the slave to be accessed. Since there is only one byte, a master device can only access up to 256 slave devices.

Modbus function code

PDU consists of function code (one byte) and data (n bytes).

The function code contains one byte. The function codes defined by modbus include:

- 01 Read the status of single or multiple coils.
- 02 Read the status of single or multiple discrete inputs.
- 03 Read single or multiple holding registers.
- 04 Read single or multiple input registers.
- 05 Write the status of single coil.
- 06 Write single holding register.
- 15 Write multiple coils.
- 16 Write multiple holding registers.

Modbus abnormal code

When the highest bit of the function code is 1 in the response message, it indicates an abnormal response. At this time, the data is a one-byte abnormal code. The specific definitions of the abnormal codes are as follows:

- 01 The function code cannot be recognized by the slave.
- 02 The unit identifier of the slave is incorrect.
- 03 The value is not accepted by the slave.
- 04 An unrecoverable error occurs when the slave tries to perform the requested operation.
- 05 The slave has accepted the request and is processing it, but it will take a long time. This response is returned to avoid timeout errors in the host. The master can send a complete message in the next polling program to determine if the processing is complete.
- 06 The slave is processing a long command. Master should try again later.
- 07 The slave cannot execute program functions. The master should request diagnostic or error information from the slave.
- 08 The slave finds a parity error in memory. The master can retry the request, but the slave may require service.
- 10 It is specially used for Modbus gateways. It indicates a gateway with incorrect configuration.
- 11 It is dedicated to Modbus gateway responses. It is sent when the slave is unable to respond.

PDU text body

1. Read coils.

Request message:

Function code: 01, one byte	Offset (start position for read data), two bytes	Read quantity, two bytes
-----------------------------	--	--------------------------

Normal response message:

Function code: 01, one byte	Data length (in bytes), one byte	Coil status data, n bytes (Since the data transmitted through the network is in whole bytes, the received data may have more bits to read than the number of bits in the request. In this case, the data is converted into a switch value bit by bit. So, it is only required to just parse the number of bits in the read quantity field in the request.)
-----------------------------	----------------------------------	--

Abnormal response message:

Function code: 129(0x81), one byte	Abnormal code, one byte
------------------------------------	-------------------------

2. Read discrete inputs.

Request message:

Function code: 02, one byte	Offset (start position for read data), two bytes	Read quantity, two bytes
-----------------------------	--	--------------------------

Normal response message:

Function code: 02, one byte	Data length (in bytes), one byte	Discrete input status data, n bytes (Since the data transmitted through the network is in whole bytes, the received data may have more bits to read than the number of bits in the request. In this case, the data is converted into a switch value bit by bit. So, it is only required to just parse the number of bits in the read quantity field in the request.)
-----------------------------	----------------------------------	--

Abnormal response message:

Function code: 130(0x82), one byte	Abnormal code, one byte
------------------------------------	-------------------------

3. Read holding registers.

Request message:

Function code: 03, one byte	Offset (start position for read data), two bytes	Read quantity, two bytes
-----------------------------	--	--------------------------

Normal response message:

Function code: 03, one byte	Data length (in bytes, twice the number of reads in the request message), one byte	Data in the holding register, n bytes (twice the number of reads in the request message)
-----------------------------	--	--

Abnormal response message:

Function code: 131(0x83), one byte	Abnormal code, one byte
------------------------------------	-------------------------

4. Read input registers.

Request message:

Function code: 04, one byte	Offset (start position for read data), two bytes	Read quantity, two bytes
-----------------------------	--	--------------------------

Normal response message:

Function code: 04, one byte	Data length (in bytes, twice the number of reads in the request message), one byte	Data in the input register, n bytes (twice the number of reads in the request message)
-----------------------------	--	--

Abnormal response message:

Function code: 132(0x84), one byte	Abnormal code, one byte
------------------------------------	-------------------------

5. Read single coil.

Request message:

Function code: 05, one byte	Offset (start position for written data), two bytes	Coil status value to be written (only focusing on 0 and non-0), two bytes
-----------------------------	---	---

Normal response message:

Function code: 05, one byte	Offset (start position for written data), two bytes	Coil status value to be written (only focusing on 0 and non-0), two bytes
-----------------------------	---	---

Abnormal response message:

Function code: 133(0x85), one byte	Abnormal code, one byte
------------------------------------	-------------------------

6. Write single holding register.

Request message:

Function code: 06, one byte	Offset (start position for written data), two bytes	Holding register data to be written, two bytes
-----------------------------	---	--

Normal response message:

Function code: 06, one byte	Offset (start position for written data), two bytes	Holding register data to be written, two bytes
-----------------------------	---	--

Abnormal response message:

Function code: 134(0x86), one byte	Abnormal code, one byte
------------------------------------	-------------------------

7. Write multiple coils.

Request message:

Function code: 15, one byte	Offset (start position for written data), two bytes	Quantity to be written, two bytes	Data length (in bytes), one byte	Coil status data, n bytes
-----------------------------	---	-----------------------------------	----------------------------------	---------------------------

Normal response message:

Function code: 15, one byte	Offset (start position for written data), two bytes	Quantity to be written, two bytes
-----------------------------	---	-----------------------------------

Abnormal response message:

Function code: 143(0x8f), one byte	Abnormal code, one byte
------------------------------------	-------------------------

8. Write multiple holding registers.

Request message:

Function code: 16, one byte	Offset (start position for written data), two bytes	Quantity to be written, two bytes	Data length (in bytes), one byte	Holding register data, n bytes
-----------------------------	---	-----------------------------------	----------------------------------	--------------------------------

Normal response message:

Function code: 16, one byte	Offset (start position for written data), two bytes	Quantity to be written, two bytes
-----------------------------	---	-----------------------------------

Abnormal response message:

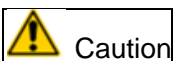
Function code: 144(0x90), one byte	Abnormal code, one byte
------------------------------------	-------------------------

ModbusTCP for IRC controller

Description

The IRC controller is configured with the Modbus TCP Slave bus protocol.

Modbus TCP communication can be realized from multiple I/O devices and gateways.



Caution

The IRC controller can only serve as an Modbus TCP slave.

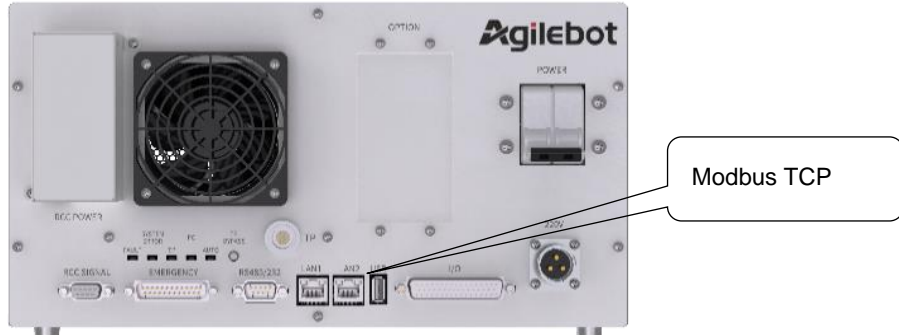
Specification overview

Type	Specification
Industrial network type	Modbus TCP
Port	RJ45
Baud rate	10/100Mbit/s
Master~Slave connection	Slave

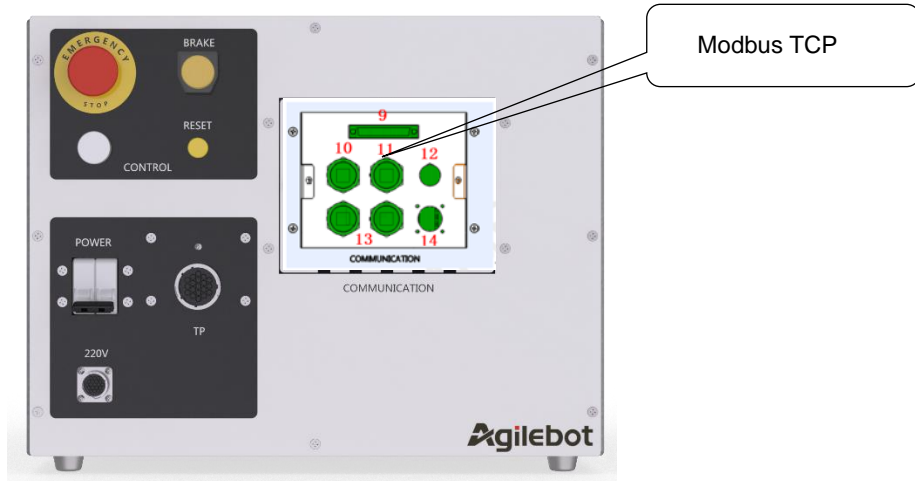
2 Hardware overview

2.1 Communication port of controller

Communication port



Modbus TCP port for IRC-14A-C controllers



Modbus TCP port for IRC-18A-S controllers

Both LAN1 and LAN2 can serve as Modbus TCP ports.

2.2 Cable connection

2.2.1 Selection of bus cables

ModbusTCP is communicating via standard Ethernet cables with a shielding layer above CAT5.

Bus network equipment should be reliably grounded to reduce interference and improve stability.

3 Software overview

3.1 Information of slave

Overview

Generally, Modbus TCP slave can be used to:

- Connect the IRC controller to PLC.
- Connect the IRC controller to a module with Modbus TCP master function.

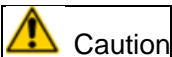
The Modbus TCP slave shares IP and port number with the controller.

Predefined networks

The IRC controller has been provided with and pre-configured with Modbus TCP slave functions

I/O connection

The slave supports polling connections.



Caution

When a polling connection is adopted in the Modbus TCP slave device, the device's signal is directly updated.

Restriction

Modbus TCP internal slave devices have the following restrictions:

- The predefined address range supported by internal slaves "Coils" and "Discrete Inputs" is: 1-1024.
- The predefined address range supported by internal slaves "Holding Registers" and "Input Registers" is: 1-120.
- Input and output mappings can be set independently.

4 Configuration of slaves

4.1 Configuration steps

Overview

Internal slave information has been pre-configured during system startup. The numbers of signal inputs and outputs cannot be changed.

When the controller is connected to an external master, it serves as a slave in the Modbus TCP network.



Caution

The controller can only be used as a slave.

Basic procedures

Adopt this step to configure Modbus TCP slave.

Action	Further information
Configure internal slave. Use the teach pendant of Compass or IRC controller.	Configure the internal slave in Section 4.2 "Configuring Internal Slaves".

4.2 Configure internal slaves.

Configuration of internal slave

Adopt this process to configure internal slaves in the IRC controller and use TP or Compass.

1. Start the controller, log in to your account and choose the manual mode.
2. After entering the system, click on the Menu - System - Other Settings. Then, click on "Modify Controller IP" to change it to the desired IP.



Caution

The set IP takes effect after the controller is restarted.

Modify Controller IP ✕

! The following configuration will take effect after the controller reboots

* New IP address: • • •

* Subnet mask: • • •

Default gateway: • • •

Cancel
Confirm

3. Click the Menu - Communication - Bus Configuration, then click the defined name after configuration, set the response delay of the slave and save the data.
 - (1) IP: IP of current controller, namely IP of the robot as a Modbus TCP slave, read-only.
 - (2) Port number: The slave's port range is 6000-6099, or 502.
 - (3) Name: Name of slave, modifiable.
 - (4) Response delay: After receiving the request, the slave delays for a certain amount of time before responding. The default delay is 0ms, namely no active delay.
 - (5) Disable/Enable: "Enable" indicates that the slave is in an active state, joining the Modbus network. "Disable" means that the slave is disabled and removed from the Modbus network.

	admin	No Program Running	UF:0	Group:1	Joint Coordinate	10% No Limit
	2024-01-18 14:04:30	Loading...	TF:0	SERVO_OFF	Continue	

ModBus-TCP

Robot as : **Slave**

[Config](#) [Function](#)

IP 172.16.1.202

Port

* Name

* Response Delay ms Disable Enable

- After configuration, click on the function options to configure the addresses and numbers of Coil Status, Discrete Inputs, Holding Registers and Input Registers.

	admin	No Program Running	UF:0	Group:1	Joint Coordinate	10% No Limit
	2024-01-18 14:05:06	Loading...	TF:0	SERVO_OFF	Continue	

ModBus-TCP

Robot as : **Slave**

[Config](#) [Function](#)

[Coil Status](#)

[Discrete inputs](#)

[Holding Registers](#)

[Input Registers](#)

Address

Quantity

- Return to the "Configuration" option and click "Enable" to enable the Modbus TCP function.

	admin	No Program Running	UF:0	Group:1	Joint Coordinate	10% No Limit
	2024-01-18 14:05:26	Loading...	TF:0	SERVO_OFF	Continue	

ModBus-TCP

Robot as : **Slave**

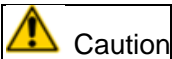
[Config](#) Function

IP **172.16.1.202**

Port **502**

* Name **ModbusServer**

* Response Delay **0** ms Disable Enable



Caution

The address range supported by Coils and Discrete Inputs is: 1-1024.

The address range supported by Holding Registers and Input Registers is: 1-120.

For the registers of the same type, the addresses are consecutive.

5 Configuration of I/O mapping

5.1 Configuration of I/O mapping

Overview

Map Coils and Discrete Inputs in Modbus TCP to I/O of the robot one by one.

Characteristics of Modbus TCP data

The communication data of Modbus TCP has the following characteristics.

Data	Type	Read and write	Robot mapping
Coils	Bit	Read and write	DI
Discrete Inputs	Bit	Read only	DO
Holding Registers	16 Bit	Read and write	MH register
Input Registers	16 Bit	Read only	MI register

The following mode is recommended for communication between TCP master (client) and slave (server):

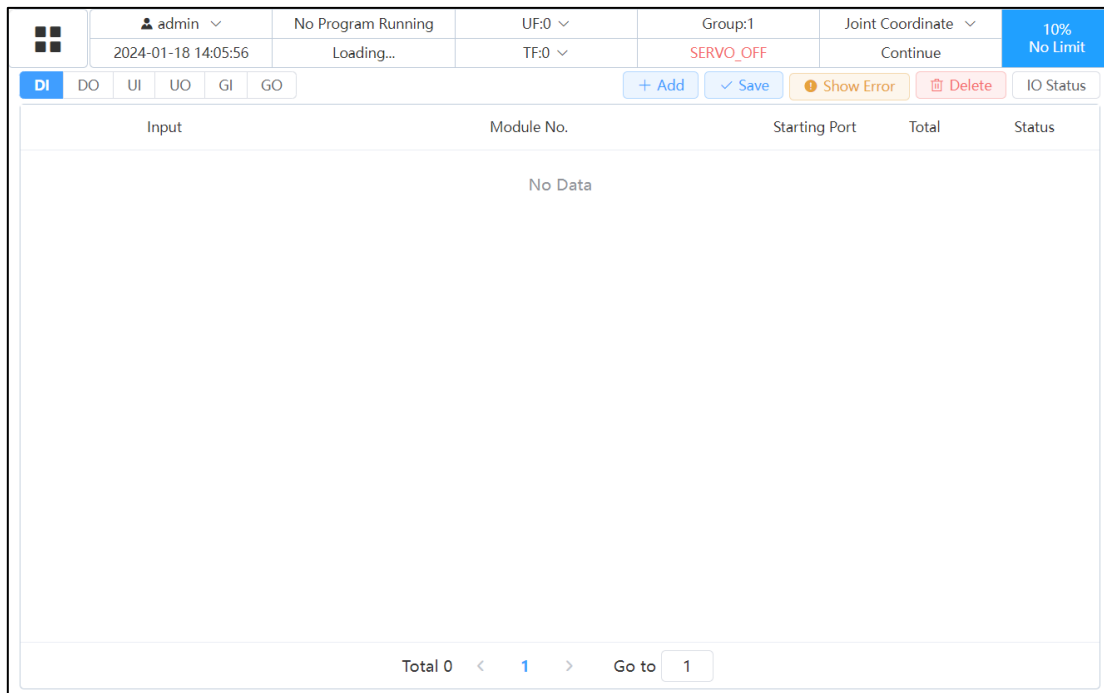
Coils and Holding Registers are used when the master writes data to the slave.

Discrete Inputs and Holding Registers are used when the slave writes data to the master.

Configuration of I/O mapping

Use TP or Compass to configure internal I/O mapping in the IRC controller.

1. Enable Modbus TCP function according to Section 4.2 "Configuring Internal Slaves".
2. Click Menu - Communication- I/O Mapping to enter the screen of I/O mapping configuration.



- Set DI/DO start and end addresses, select the ModbusServers/TCP module number, set the start port number, and click Save. Then, the status is displayed as Active.

The screenshot shows the configuration interface for the Modbus TCP Slave. The top status bar indicates 'No Program Running', 'UF:0', 'TF:0', 'SERVO_OFF', and '10% No Limit'. The 'DI' tab is selected. The configuration table below shows one entry:

Input	Module No.	Starting Port	Total	Status
DI 1 ~ 5	ModbusServer/TCP/128/20in:20out	1	5	active

At the bottom, the summary shows 'Total 1' and 'Go to 1'.

- Set UI/UO start and end addresses, select the ModbusServer/TCP module number, set the start port number, and click Save. Then, the status is displayed as Active.

The screenshot shows the configuration interface for the Modbus TCP Slave. The top status bar indicates 'No Program Running', 'UF:0', 'TF:0', 'SERVO_OFF', and '10% No Limit'. The 'UI' tab is selected. The configuration table below shows three entries:

Input	Module No.	Starting Port	Total	Status
UI 1 ~ 5	ModbusServer/TCP/128/20in:20out	6	5	active
UI 6 ~ 10	ModbusServer/TCP/128/20in:20out	11	5	active
UI 11 ~ 13	ModbusServer/TCP/128/20in:20out	16	3	active

At the bottom, the summary shows 'Total 3' and 'Go to 1'.

- Set GI/GO start and end addresses, select the ModbusServer/TCP module number, set the start port number, and click Save. Then, the status is displayed as Active.

Serial Number	Module No.	Port	Total	Status
GI[1]	ModbusServer/TCP/128/20in:20out	1-5	5	active

Total 1 < 1 > Go to 1

I/O status query and modification

After I/O mapping, the I/O status can be queried through "Menu - Communication - I/O Status".

View the states of UI/UO and GI/GO by clicking the dropdown triangle next to DI/DO.

Rename the port by changing the port number. After "OFF", click the drop-down triangle to modify the switch status of corresponding I/O port.

Please refer to the *Operation Manual for Agilebot Robot System* for details.

Caution

During configuration, it should be noted that an In/Out port can only be assigned to one DI/UI or DO/UO.

	admin ▼ 2024-01-18 14:08:33	No Program Running Loading...	UF:0 ▼ TF:0 ▼	Group:1 SERVO_OFF	Joint Coordinate ▼ Continue	10% No Limit
DI/DO ^				⊞ Cancel All Simulations IO Mapping		
DI/DO	Name	Simulation	Value	Port	Name	Value
UI/UO	<input type="text"/>	UnSim Sim	OFF	DO[1]	<input type="text"/>	OFF ▼
RI/RO	<input type="text"/>	UnSim Sim	OFF	DO[2]	<input type="text"/>	OFF ▼
GI/GO	<input type="text"/>	UnSim Sim	OFF	DO[3]	<input type="text"/>	OFF ▼
DI[3]	<input type="text"/>	UnSim Sim	OFF	DO[4]	<input type="text"/>	OFF ▼
DI[4]	<input type="text"/>	UnSim Sim	OFF	DO[5]	<input type="text"/>	OFF ▼
DI[5]	<input type="text"/>	UnSim Sim	OFF	DO[6]	<input type="text"/>	OFF ▼
DI[6]	<input type="text"/>	UnSim Sim	UNKNOWN	DO[7]	<input type="text"/>	OFF ▼
DI[7]	<input type="text"/>	UnSim Sim	UNKNOWN	DO[8]	<input type="text"/>	OFF ▼
DI[8]	<input type="text"/>	UnSim Sim	UNKNOWN	DO[9]	<input type="text"/>	OFF ▼
DI[9]	<input type="text"/>	UnSim Sim	UNKNOWN	DO[10]	<input type="text"/>	OFF ▼
DI[10]	<input type="text"/>	UnSim Sim	UNKNOWN			
Total 1024 < > Go to <input style="width: 30px;" type="text" value="1"/>				Total 1024 < > Go to <input style="width: 30px;" type="text" value="1"/>		

6 Configuration and monitoring of Modbus TCP Registers

Overview

The robot provides a Modbus register monitoring screen, which is used to read or write Holding Registers and Input Registers configured in the Modbus TCP slave.

6.1 Configuration of Modbus TCP Registers

Overview

For Holding Registers and Input Registers, it is only required to set the address and number in the function options when Modbus TCP is enabled. Mapping configuration is not required.

Address range

The address range of Register is 1-120.

Range of Register value

The range of values for Input Register and Holding Register is an integer between [0 and 65535].

6.2 Monitoring of Register status

Definition in the system

In the robot system, the Holding Register is represented by the MH register, while the Input Register is represented by the MI register.

Status monitoring

1. Configure the slave and enable the Modbus TCP function according to Section 4.2
2. Click Menu - Data - Modbus Register to perform monitoring.
3. Click the dropdown triangle to select Holding Registers and Input Registers.
4. Register values can be manually modified.

	admin	No Program Running	UF:0	Group:1	Joint Coordinate	10% No Limit
	2024-01-18 14:08:57	Loading...	TF:0	SERVO_OFF	Continue	

Holding Registers
MH[1]
MH[2]
MH[3]
MH[4]
MH[5]
MH[6]
MH[7]
MH[8]
MH[9]
MH[10]

< > Go to

Address 1

Register Value 0

7 System parameters

7.1 Industrial network

The ModbusTCP industrial network has a mandatory requirement for addresses. Master and internal slave devices of ModbusTCP should communicate and connect with other devices in the ModbusTCP network via clear addresses.

The IP address should not be the same as another I/O device in the network.

The IP address is the default address of the robot controller and the port number is 502 by default.

The IP address and Modbus TCP master should be in the same network segment.

8 Faults

Description

Possible causes for Modbus TCP communication failure in IRC controller:

- Slave station is not activated.
- The Internet cable is not connected correctly.
- IP is set wrongly.
- The master is set wrongly.

Possible accidents

The controller cannot detect communication failure (if any) of Modbus TCP. It is required to avoid accidents caused by Modbus TCP communication failure during use.

Polling scan

When starting working, Modbus TCP master accesses slaves in a polling manner. If a slave does not respond to the master's request, it means that the communication of the slave fails. Reset the master and slave after the fault is repaired.

Contact us

Agilebot Robotics Co., Ltd. (Shanghai Headquarters):

Floor 8, Tower 6, Zhongjian Jinxiu Plaza, No. 50, Lane 308, Xumin Road, Qingpu District, Shanghai

Agilebot Operation and Technical Service Center:

Building 1, No. 338 Jiuye Road, Qingpu District, Shanghai

Service hotline: +86-21-5986 0805

Website: www.sh-agilebot.com