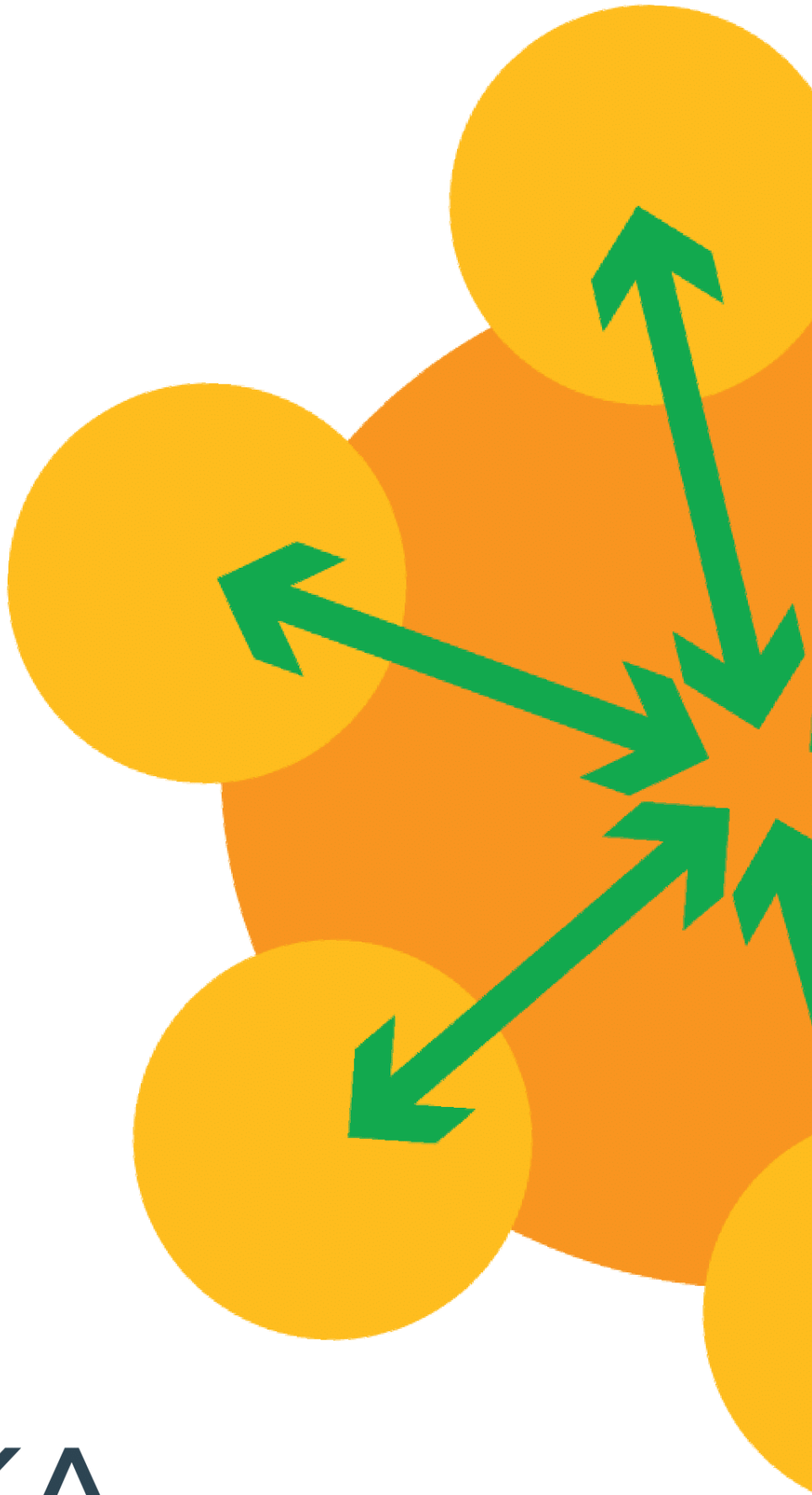




Guide

FRANKA MODBUS

© Copyright 2025
Franka Robotics GmbH
Frei-Otto-Straße 20
80797 Munich
Germany

Modbus

Release date: May 2025

Abstract

This document provides a comprehensive guide to Modbus configuration for administrators. It covers the fundamental concepts of Modbus, including its applications and benefits in industrial communication. The guide details the necessary steps for network preparation and configuration, offering a clear and structured approach to setting up Modbus TCP. It includes instructions for activating the Modbus function, saving and restoring configuration settings, and verifying the setup through testing. Additionally, the document addresses diagnostic statuses in the TCP section and troubleshooting common issues. A glossary of technical terms is included to support users in navigating the documentation effectively.

Table of Contents

1.	Modbus Configuration Overview	4
2.	Modbus Introduction.....	5
3.	Network Preparation and Configuration.....	6
3.1	Network Preparation	6
3.2	Network Configuration	6
4.	Configuration Overview	8
5.	Testing.....	12
6.	Troubleshooting	13
6.1	Uploading the configuration file fails.....	13
6.2	Current service values and diagnostics	14
6.2.1	No route to host (Host: [address], Port: [port]).....	14
6.2.2	Connection refused (Host: [address], Port: [port])	14
6.2.3	Illegal data address (Host: [address], Port: [port]).....	14
6.2.4	Illegal function (Host: [address], Port: [port])	14
7.	Glossary	15
8.	Table of Figures	17

1. Modbus Configuration Overview

Modbus Configuration is a feature available to administrators in the robot's Settings interface. The robot implements a Modbus Client, allowing for the setup and management of Modbus communication. This protocol is used for transmitting information over serial lines between electronic devices. The connection between the robot and the Modbus coupler is established through TCP/IP.

Key Functions:

1. Download Modbus Manual: Provides a manual with detailed instructions on Modbus configuration.
2. Activating the Modbus Function: Enables the Modbus communication feature.
3. Upload and download the Modbus Configuration: Allows you to save and restore the Modbus settings.

Steps to Save Modbus Configuration:

1. Open Settings: Access the Settings interface on your device.
2. Select the MODBUS TCP Tab: Navigate to the specific tab for Modbus TCP settings.
3. Download Configuration: Click on the "DOWNLOAD" button to save the current Modbus configuration as a file named modbus.json on your device.
4. Save the File: Store the downloaded modbus.json file in a backup directory you have created.
5. Restore Configuration: Use the "UPLOAD" button to restore a previously saved Modbus configuration file back to the system.

This process ensures that you can back up your Modbus settings and restore them when needed, maintaining the communication setup for your robot.

2. Modbus Introduction

Modbus TCP/IP is a client-server protocol used to exchange process data between different controllers over the Ethernet. It offers flexibility for decentralized control architectures. In a network, each participant can act as both a server and a client.

Communication Process

To exchange data, the client sends a data request to the server. The server responds with either a success message or the requested data. For communication to be established, the client needs to know the server's IP address and port number (typically 502).

Data Types

Modbus differentiates between the following four data types:

Type	Access	Size
Input Bits	Read	1 Bit
Bits	Read/Write	1 Bit
Input Registers	Read	16 Bits
Holding Registers	Read/Write	16 Bits

Request Elements

A server holds the data and provides information to a client upon request. The important elements within a request are:

- **Address:** The address on the server for the requested information.
- **Function:** The function code indicating the action to be performed.
- **Data:** Only necessary if a write function is called. This field contains the information to be written.

Function Codes

The function code is the identifier for the server to select the correct data and return it if a read function is called, or to write the received data to the correct range if a write function is called. Common function codes include:

Function	Code (Hex)	Action	Description
01	01 hex	Read	Read Discrete Output Coils
02	02 hex	Read	Read Discrete Input Contacts
03	03 hex	Read	Read Output Holding Registers
04	04 hex	Read	Read Input Registers
06	06 hex	Write	Write Output Holding Register
15	0F hex	Write	Write Discrete Output Coils

Example

For example, if a client wants to read the status of a coil at address 100, it will send a request with function code 01 and address 100. The server would respond with the status of that coil.

3. Network Preparation and Configuration

3.1 Network Preparation

A regular Ethernet cable can be used to enable communication between controllers. It is important to use the Ethernet port at the robot's controller, as shown in the picture below. The controllers can be connected directly or over a switch.



Figure 1: Ethernet port

3.2 Network Configuration

Once all controllers are connected, valid IP addresses need to be assigned. To configure the robot's IP address, go to Settings (found in the top left corner of Desk) and select "Network".

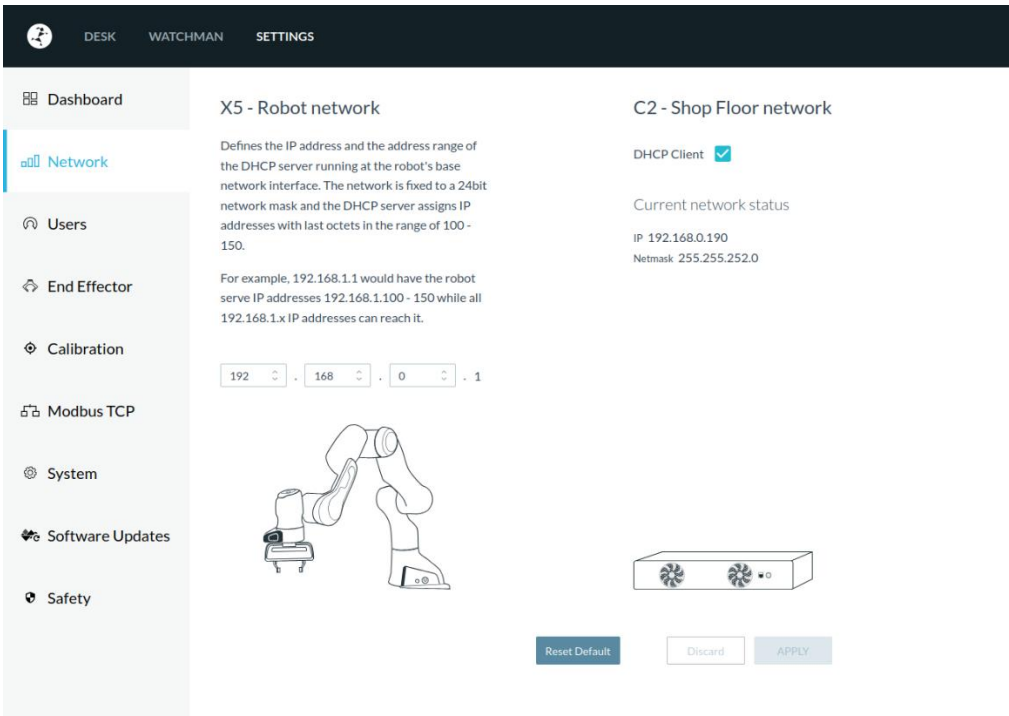


Figure 2: Network

You will not need to edit the settings at “Robot network” as the communication should be established over the Shop Floor network. Please enter a valid IP address and apply your settings. The robot control and your module must be configured to appear on the same network. The simplest way to achieve this is to use static IP addresses. If this is not a valid option for you, please contact your administrator. An example of a valid network configuration would be:

Control	Module
Address Robot	192.168.0.190
Address Modbus	192.168.0.147
Netmask Robot	255.255.252.0
Netmask Modbus	255.255.252.0

The information on how to configure the IP address of the Modbus TCP Server device / coupler in the network should be available in their respective manuals.

4. Configuration Overview

The service on the robot acts as a client and needs to know the configuration of the server with which it should communicate. The configuration file is in JSON format. If you encounter any difficulties with this format, further information can be found here. <http://www.json.org/>.

Before diving into the configuration syntax in detail, let's look at the service itself.

4.1 Service Description

The service (acting as a client) can communicate with multiple servers in the network, referred to as "Modules" in the configuration. For each server, multiple input and output settings can be defined, known as "Cards". Once the configuration file is uploaded, it will be validated, and the service will attempt to connect to the server and start polling information from each server.

4.2 Configuration Settings

The following table provides an overview of each setting in the configuration file:

Setting Name	Description
moduleName	The name of the server module.
inputCards	A list of input settings for the server.
outputCards	A list of output settings for the server.
pollingInterval	The interval at which the service polls the server for information.
connectionTimeout	The timeout duration for establishing a connection with the server.

An example configuration could look like the following snippet:

Code Block

```
{
  "frequency": 10,
  "modules": [
    {
      "ip": "192.168.0.147",
      "port": "502",
      "inputs": [
        {
          "read_address": 8000,
          "count": 16,
          "type": "register"
        }
      ]
    },
    {
      "read_address": 9001,
      "write_address": 9001,
      "count": 16,
      "type": "register"
    }
  ]
}
```

The configuration snippet shown above is designed for a Beckhoff EK9000 at IP address 192.168.0.147, with 16 inputs (DI EL1809) and 16 outputs (DO EL2809) connected to it. In this configuration, modules, inputs, and outputs are defined as arrays, allowing for the definition of multiple objects.

For each module, an IP address and a port must be set. Different input and output settings can be defined. Each input or output must include several inputs or outputs and their types. Types can be either "coil" or "register." Input configurations must contain a read address. Output configurations must contain a write address and can optionally include a read address if it differs from the write address.

Once your configuration file is ready, please go to Settings and select "Modbus TCP." From there, you will be able to upload the configuration file.

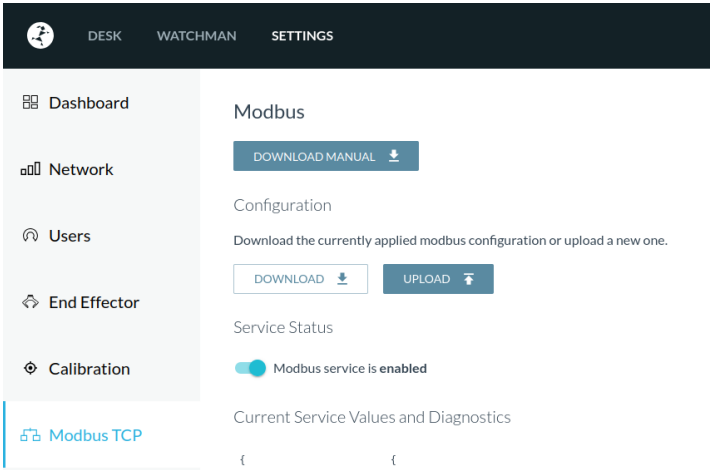


Figure 3: TCP

Depending on the configuration different functions are used by the client on the robot controller to read/write data to/from the module:

Modbus Function Codes

Operation	Type	0x04	0x02	0x03	0x01	0x06	0x0F
Read Input	Coil		x				
	Register	x					
Read Output	Coil			x			
	Register				x		
Write Output	Coil					x	
	Register						x

In the upper example the function 0x04 will be used to read the inputs at server address 8000. Function 0x03 will be used to read the outputs at server address 9001 and function 0x06 will be used to write the output registers at address 9001.

If desired, it is possible to name each I/O-pin, by using the key “pin_names” in the JSON -file. Additionally, you can set up pin configurations by using the key “pin_config”. This is particularly helpful if you want to use a set of pins several times. The pin names and the pin configurations will be visible as part of the configuration panels of the Apps, in Desk.

4.3 Example: Naming I/O-pins

Add the key “pin names” with the pin numbers and the corresponding names for the pins to the JSON-file. An example configuration could look like the following code block:

Code Block

```
{
  "frequency": 10,
  "modules": [
    {
      "inputs": [
        {
          "count": 16,
          "read_address": 0,
          "type": "register"
        }
      ],
      "ip": "192.168.0.147",
      "outputs": [
        {
          "count": 16,
          "read_address": 2048,
          "type": "register",
          "write_address": 2048,
          "pin_names": {
            "1": "conveyor_belt",
            "2": "solenoid",
            "3": "any name"
          }
        }
      ],
      "port": "502"
    }
  ]
}
```

The result will be visible in the Apps' configuration panel, as shown in the following screenshot:

Select a module and a card, and set its pins to your desired values.

Module 1 / Card 1

conveyor belt 1

Off

On

solenoid 2

Off

On

any name 3

Off

On

unnamed pin 4

Off

On

unnamed pin 5

Off

On

unnamed pin 6

Off

On

unnamed pin 7

Off

On

unnamed pin 8

Off

On

unnamed pin 9

Off

On

unnamed pin 10

Off

On

unnamed pin 11

Off

On

unnamed pin 12

Off

On

unnamed pin 13

Off

On

unnamed pin 14

Off

On

unnamed pin 15

Off

On

unnamed pin 16

Off

On

Off all

Ignore all

On all

Figure 4: Configuration panel

To write or set the Modbus signals, you can utilize the "Modbus Out" app. For reading the Modbus signals, you can use the "Branch", "Catch Event", "If", or "Repeat" apps. These logical apps enable users to teach robots based on the current Modbus signal configuration.

5. Testing

Once a correct configuration file is uploaded, some lines will appear in the section “ModBus TCP” of Settings, under “Current Services Values and Diagnostics”.

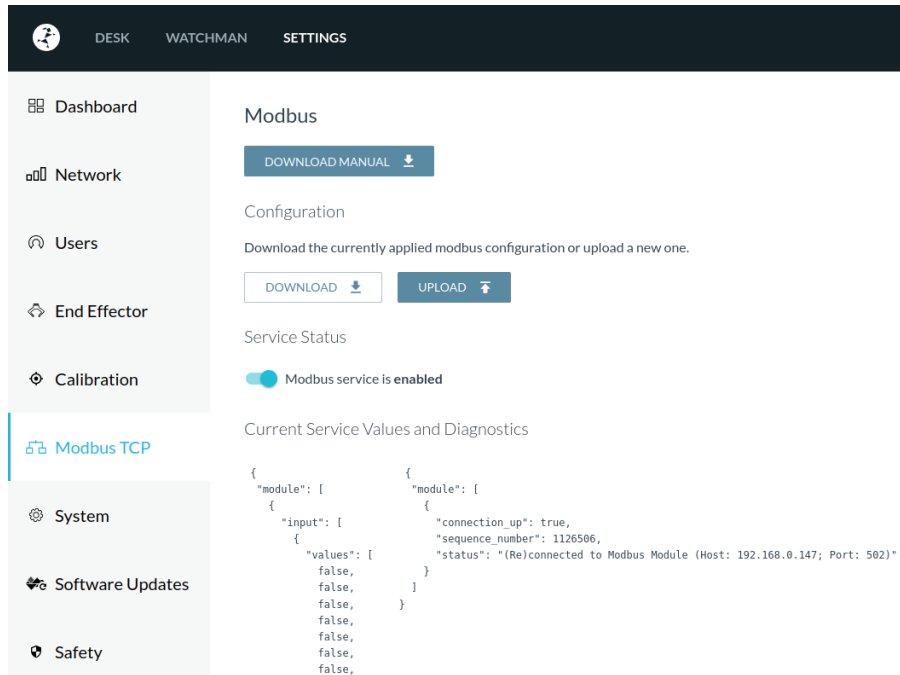


Figure 5: TCP Settings

To test if the configuration file enables successful communication, follow these steps:

1. **Create a Task:**
 - Go back to Desk and create a task, for example, named "Output Test".
2. **Add the App:**
 - Drag the "ModBus Out" App into the timeline.
3. **Select Module and Card:**
 - Select the module and card of the connected server. In this configuration, the pins can be set to "ON".
4. **Execute the Task:**
 - After executing the task, the output signal of the pin on your connected server should be switched on. You will also be able to see the pin flashing on the Modbus coupler hardware.

Select a module and a card, and set its pins to your desired values.

Module 1 / Card 1

conveyor_belt 1

Off

On

solenoid 2

Off

On

any name 3

Off

On

unnamed pin 4

Off

On

unnamed pin 5

Off

On

unnamed pin 6

Off

On

unnamed pin 7

Off

On

unnamed pin 8

Off

On

unnamed pin 9

Off

On

unnamed pin 10

Off

On

unnamed pin 11

Off

On

unnamed pin 12

Off

On

unnamed pin 13

Off

On

unnamed pin 14

Off

On

unnamed pin 15

Off

On

unnamed pin 16

Off

On

Off all

Ignore all

On all

Figure 6: Output signal

6. Troubleshooting

6.1 Uploading the configuration file fails

If there is a syntax error in the JSON configuration file, the upload will fail with the following message:

Uploading configuration file failed



Expected ',' or ']' after array element in JSON at position 697 (line 29 column 6)

Dismiss

Figure 7: Fail message.

In such case, please check the syntax for errors and fix them. The message also gives you a hint at which character position in the configuration file the parsing of the JSON file failed, including a parsing error message.

6.2 Current service values and diagnostics

```
{
  "connection_up": false,
  "sequence_number": 13791,
  "status": "Operation now in progress (Host: 192.168.0.147; Port: 502)"
}
```

Figure 8: Status

Each configured module publishes its status in the route in the Modbus TCP section.

6.2.1 No route to host (Host: [address], Port: [port])

There is no ethernet connection to the server which was defined in the configuration. Please check if the network wiring is ok and that all configured IP-Addresses are in the same range.

6.2.2 Connection refused (Host: [address], Port: [port])

The Modbus TCP Server (the externally connected Modbus module) refused the connection of the Modbus Client in Control. Please check if your connected module accepts requests on the configured port.

6.2.3 Illegal data address (Host: [address], Port: [port])

The client requested data from an illegal data address. Please check the manual of your Modbus TCP server for the correct addresses.

6.2.4 Illegal function (Host: [address], Port: [port])

The function sent by the client is not supported by your server. In the section “configuration” of this guide, an overview was included to illustrate which function is used given the configuration.

7. Glossary

Term	Definition
Modbus Configuration	A feature available to administrators in the robot's Settings interface, allowing for the setup and management of Modbus communication.
Modbus Client	The implementation within the robot that facilitates Modbus communication.
Modbus Protocol	A protocol used for transmitting information over serial lines between electronic devices.
TCP/IP	A suite of communication protocols used to interconnect network devices on the internet, including the connection between the robot and the Modbus coupler.
Modbus Manual	A document providing detailed instructions on Modbus configuration.
Modbus Function	The feature that enables Modbus communication.
Modbus Configuration File (modbus.json)	A file used to save and restore Modbus settings.
Client-Server Protocol	A communication model where the client sends requests to the server, which responds with the requested data or a successful message.
Ethernet	A system for connecting computers within a local area network (LAN).
Input Bits	Data type in Modbus, accessible for reading, consisting of 1 bit.
Bits	Data type in Modbus, accessible for reading and writing, consisting of 1 bit.
Input Registers	Data type in Modbus, accessible for reading, consisting of 16 bits.
Holding Registers	Data type in Modbus, accessible for reading and writing, consisting of 16 bits.
Function Code	An identifier used by the server to select the correct data for reading or writing.
Discrete Output Coils	A type of data in Modbus, read using function code 01.
Discrete Input Contacts	A type of data in Modbus, read using function code 02.
Output Holding Registers	A type of data in Modbus, read using function code 03.
Input Registers	A type of data in Modbus, read using function code 04.
Output Holding Register	A type of data in Modbus, written using function code 06.
Discrete Output Coils	A type of data in Modbus, written using function code 15.
Network Preparation	The process of setting up the network for communication between controllers, typically using Ethernet cables.
Network Configuration	The process of assigning valid IP addresses to controllers and configuring the robot's IP address in the Settings interface.

Static IP Address	A fixed IP address assigned to a device, ensuring it remains the same over time.
Netmask	A mask used to determine what subnet an IP address belongs to.
Modbus TCP Server	A device or coupler in the network that communicates using the Modbus TCP protocol.
Service Description	Details about the service acting as a client, communicating with multiple servers in the network.
Configuration File (JSON)	A file in JSON format used to define the settings for Modbus communication.
Polling Interval	The interval at which the service polls the server for information.
Connection Timeout	The duration for establishing a connection with the server before timing out.
Modbus Module	A server in the network with which the client communicates.
Card	Input and output settings defined for each server module.
Pin Names	Names assigned to each I/O pin in the configuration file.
Pin Config	Settings for configuring pins, useful for repeated use of a set of pins.
Testing	The process of verifying if the configuration file enables successful communication.
Task	A Task refers to an automation process composed of individual Apps. Examples of Apps are a Move, Wait or Modbus Out App.
Modbus Out App	An application used in the timeline to manage Modbus outputs in Desk.
Troubleshooting	The process of diagnosing and resolving issues with Modbus configuration and communication.
Diagnostic Status	The status information published by each configured module in the TCP section.
No Route to Host	An error indicating no Ethernet connection to the server defined in the configuration.
Connection Refused	An error indicating that the server refused the connection from the client.
Illegal Data Address	An error indicating the client requested data from an illegal address.
Illegal Function	An error indicating the function sent by the client is not supported by the server.

8. Table of Figures

Figure 1: Ethernet port	6
Figure 2: Network	7
Figure 3: TCP	10
Figure 4: Configuration panel	11
Figure 5: TCP Settings	12
Figure 6: Output signal	13
Figure 7: Fail message	13
Figure 8: Status	14

Franka Robotics GmbH
Frei-Otto-Straße 20
80797 Munich
Germany