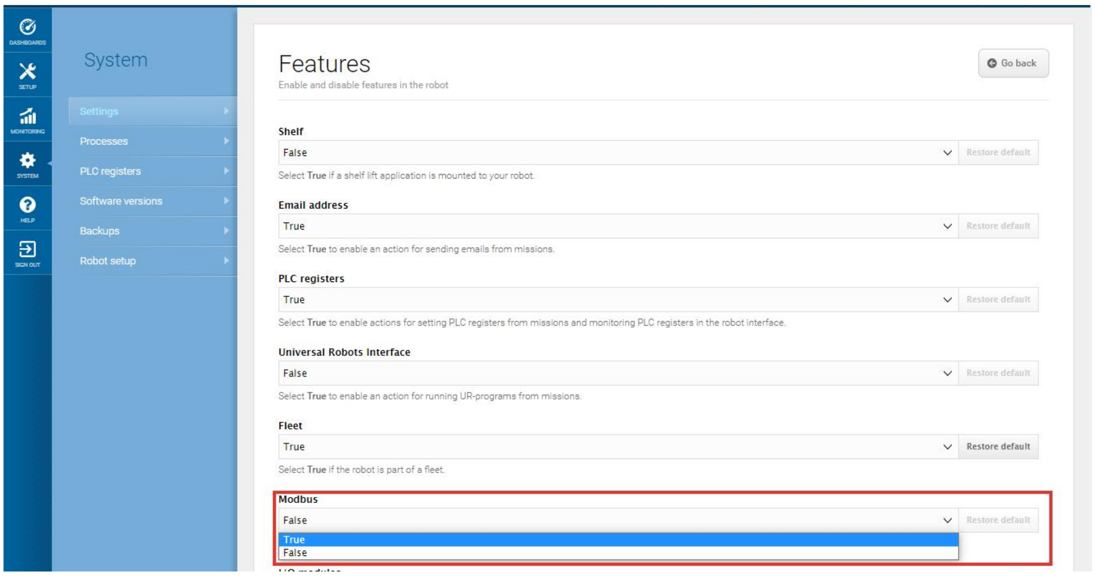
How to use Modbus with MiR robots

Date: 2021-05-13
Document version: 1.3
Valid for: All MiR robots
Valid for software version: N/A
Valid for hardware version: N/A

Modbus is a software communication tool that is used to communicate between MiR robots and other devices. You can enable or disable it as needed.

This article explains what Modbus is, how it can be utilized, and how to set up Modbus on your MiR robots.

To enable the Modbus feature on the robot, go to **System > Settings > Features** and enable **Modbus**. Select **Save changes** at the bottom of the page.



Features
Enable and disable features in the robot

Shelf
False
Select **True** if a shelf lift application is mounted to your robot.

Email address
True
Select **True** to enable an action for sending emails from missions.

PLC registers
True
Select **True** to enable actions for setting PLC registers from missions and monitoring PLC registers in the robot interface.

Universal Robots Interface
False
Select **True** to enable an action for running UR-programs from missions.

Fleet
True
Select **True** if the robot is part of a fleet.

Modbus
False
True
False

Purpose of Modbus

Modbus TCP can be used to communicate between MiR robots and other devices to achieve the following purposes:

- To control the status of the robot, such as Pause/Continue, Cancel current mission, Clear mission queue, and Clear error message.
- To get basic information from the robot, such as its status, power percentage, and error messages.
- To read and write registers in the robot.
- To trigger a mission.

If a PLC is connected to the internal network of the robot, it is not possible to access the external network through the robot.

How Modbus works on MiR robots

The Modbus slave/server is running on the robot and broadcasting on both local network IP (192.168.12.20) and an external WiFi IP address. It is listening on the standard port 502 for commands from outside devices. Usually, the device would be a PLC acting as the master/client device, and the MiR robot would be the slave/server device. The master/client device is the one in charge of polling the data.

In a typical case, the PLC (master/client) establishes a connection with the robot (slave/server). The robot waits for an incoming connection from the PLC. Once a connection is established, the robot responds to the queries from the PLC until the PLC closes the connection.

Several masters/clients can connect to the robot at the same time. Each connection is threaded.

The data is cached to avoid a high-frequency load of requests on the API while providing a response to any request.

Function code

The server can respond to function codes described in ***Table 1.1***.

Table 1.1.
Modbus function codes for MiR robots

Function	Code	Definition
01	Read	Read coils
02	Read	Read discrete inputs
03	Read	Read holding registers
04	Read	Read input register
05	Write	Write single coil
06	Write	Write single register
16	Write	Write multiple registers

In the MiR robot, the content of coils and discrete inputs are matched with each other, as are the input registers and holding registers. So you can get the same data from them, but you can only send information to coils and holding registers.

Error codes

Modbus can respond with the error codes (called exception codes) described in ***Table 1.2*** if the user sends a wrong signal to Modbus.

Table 1.2.
Modbus exception codes

Exception code (hex code)	Name	Definition
1 (01 hex)	Illegal Function	The function code received in the query is not an allowable action for the slave. It could also indicate that the slave is in the wrong state to process a request of this type, for example, because it is unconfigured and is being asked to return register values. If a Poll Program Complete command was issued, this code indicates that no program function preceded it.

Exception code (hex code)	Name	Definition
2 (02 hex)	Illegal Data Address	The data address received in the query is not an allowable address for the slave. Specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 will generate exception 02.
3 (03 hex)	Illegal Data Value	A value contained in the query data field is not an allowable value for the slave. This indicates a fault in the structure of remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
4 (04 hex)	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.
5 (05 hex)	Acknowledge	Specialized use in conjunction with programming commands. The slave has accepted the request and is processing it but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed.
6 (06 hex)	Slave Device Busy	Specialized use in conjunction with programming commands. The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free.
7 (07 hex)	Negative Acknowledge	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14

Exception code (hex code)	Name	Definition
		decimal. The master should request diagnostic or error information from the slave.
8 (08 hex)	Memory Parity Error	Specialized use in conjunction with function codes 20 and 21 and reference type 6 to indicate that the extended file area failed to pass a consistency check. The slave attempted to read extended memory or record a file but detected a parity error in memory. The master can retry the request, but service may be required on the slave device.
10 (0A hex)	Gateway Path Unavailable	Specialized use in conjunction with gateways. Indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means the gateway is misconfigured or overloaded.
11 (0B hex)	Gateway Target Device Failed to Respond	Specialized use in conjunction with gateways. Indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

Data model

The data model is described with the following points:

- Addressing
  - The Modbus address that the robot uses can be found in the tables under **Modbus data addresses reference below**.
  - The Modbus server has 0-based addressing. Be aware that some other devices are 1-based, for example, Anybus X-gateways. When using a 1-based device, add one to the address on that device, for example, address 3 on the robot will be address 4 on the Anybus X-gateway.
- Read/write scheme
  - Status messages: Read only.
  - Registers: Read and write.
  - Mission triggers: Write available. Reading always return 0 if triggers have been set.
  - Action commands: Write available. Reading always returns 0.
- Integer register
  - MiR robots save the integer in 32 bits, but the Modbus communications are 16 bits encoded. To accomodate this, In the Modbus server, two addresses are combined to represent one integer value in the robot. MiR Modbus uses big endian encoding as default configuration. This method is applied to all the integer registers and integer values of the robot's information.
- Floating register
  - MiR Modbus usees IEEE 754 ISO standard to encode floating point values.

1.1 Modbus data addresses reference

The following tables describe which Modbus addresses can be used on the robot, including their function, permission, and data type.

Action commands

Table 1.3.
Action commands you can use with MiR robots

Title	Description	Permission	Data type	Address
Pause robot	Address of the coil used to trigger the Pause action on the robot	W	Bool	[00002]
Cancel current mission	Address of the coil used to cancel the ongoing mission, if any	W	Bool	[00003]
Clear mission queue	Address of the coil used to clear the entire mission queue	W	Bool	[00004]
Clear error	Address of the coil used to clear the errors on the robot	W	Bool	[00005]
Continue robot	Address of the coil used to trigger the Continue action on the robot	W	Bool	[00006]

Mission triggers

Table 1.4.
Mission triggers you can use with MiR robots.

Title	Description	Permission	Data type	Address
Triggers	Trigger the linked mission in system settings.	W	Bool	[01001]... [02000]

Status messages

Table 1.5.
Statuses you can read from MiR robots.

Title	Description	Permission	Data type	Address
Software version	Robot software version. Each address represents one digit of the software version	R	int16	[40001, 40002, 40003]
Mode	The current mode of the robot. Refer to Code Definition for more information	R	int16	[40004]
State	The current state of the robot. Refer to Code Definition for more information	R	int16	[40005]
Error code	The last error registered on the robot. 0 if no errors were detected	R	int16	[40006, 40007]
Battery level	Remaining charge in percent [%]	R	int16	[40008]
Uptime	The robots uptime	R	int32	[40009, 40010]
Distance run	The total distance run by the robot	R	float32	[40011, 40012]
Position X	Position X in global coordinates	R	float32	[40013, 40014]
Position Y	Position Y in global coordinates	R	float32	[40015, 40016]
Position Orientation	Orientation of the robot in global coordinates [in degrees]	R	float32	[40017, 40018]
Length of mission queue	Number of missions pending or executing	R	int16	[40019]

Registers

Table 1.6.
Registers you can get/set on MiR robots.

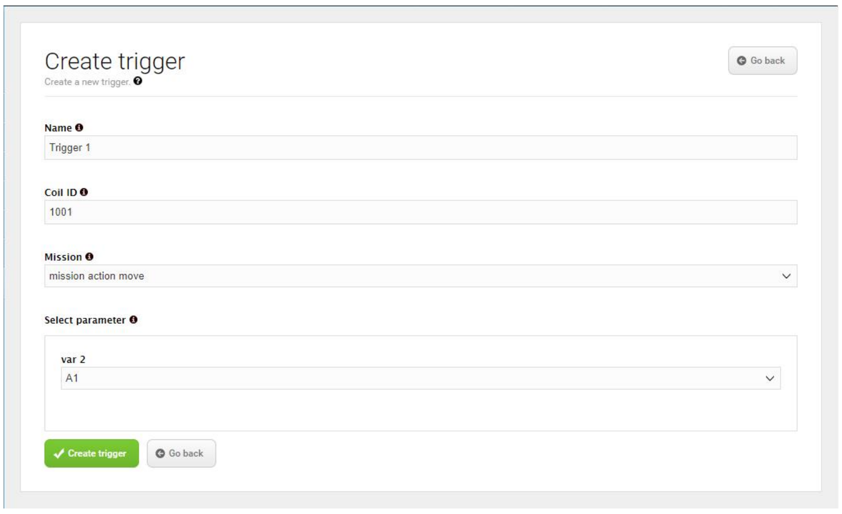
Title	Description	Permission	Data type	Address
Integer Registers	Value of integer registers	R/W	int32	[41001,41002]...[41199, 41200]
Float Registers	Value of float registers	R/W	float32	[42001,42002]...[42199, 42200]

1.2 Setting up Modbus

By default you will be able to set up a Modbus master on your PLC to get status messages, get/set registers, and set action commands from the robot. If you need to trigger missions on the robot, mission triggers need to be set up as described below.

After you have turned on the Modbus feature and created a mission, go to **System > Triggers > Create trigger** to create a new trigger. The coil ID can be assigned to any integer between 1001 and 2000. After the trigger is created, the robot will trigger the mission once you set the coil to True via Modbus.

If the mission uses variables for any parameters, you must set the variable value when you create the trigger.


Create trigger

Create a new trigger.

Name

Trigger 1

Coil ID

1001

Mission

mission action move

Select parameter

var 2

A1

Code Definition

The following tables describe the ID values for the robot's modes and states.

Robot mode

Mode ID	Description
0	Start mode
3	Robot is mapping
4	Robot is finalizing the map
7	Robot is executing
255	Robot is transitioning to a map

Robot state

State ID	Description
0	None
1	Starting
2	Shutting down
3	Ready
4	Pause
5	Executing
6	Aborted
7	Completed
8	Docked
9	Docking
10	Emergency/Protective stop
11	Manual control
12	Error

Document history

Version	Date	Description
1.0	2019-12-03	First edition.
1.1	2020-09-15	Added robot states 10-12. Small corrections throughout the guide.
1.2	2020-09-30	Added table with Modbus exception codes.
1.3	2020-10-28	Added note regarding PLC connection.