

u-remote library for Rockwell Studio 5000 Logix Designer®

Abstract:

This library contains add-on instruction for easy use of Weidmüller u-remote modules in Rockwell Studio 5000 Logix Designer®

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	UR20-4AI-UI-12	1394390000	-
2	UR20-4AI-UI-16	1315620000	
3	UR20-4AO-UI-16	1315680000	
4	UR20-8DI-P-2W	1315180000	-
5	UR20-8DO-P	1315240000	-
6	UR20-4COM-IO-LINK	1315740000	FW 1.04.00 or later
7	UR20-1COM-232-485-422	1315750000	-
8	UR20-2AI-SG-DIAG	1990070000	
9	UR20-FBC-EIP	1334920000	HW2, FW 2.11
10	UR20-FBC-EIP-V2	1550550000	-
11	UR20-FBC-EIP-ECO	2799510000	-
12	Compact Logix 5370	1769-L16ER-BB1B	Rev. 35.000

Software reference

No.	Software name	Article No.	Software version
1	Studio 5000 Logix Designer	-	V35.00.00

File reference

No.	Name	Description	Version
1	libWiUr20_4AI_4AO_V1_1_0.ACD	Using EDS file	1.1.0
2	libWiUr20_8DI_8DO_V1_1_0.ACD	Using EDS file	1.1.0
3	libWiUr20_4COM_IO_LINK_V1_1_0.ACD	Using EDS file	1.1.0
4	libWiUr20_4COM_IO_LINK_CIP_V1_1_0.ACD	Using CIP Bridging	1.1.0
5	libWiUr20_2AI_SG_DIAG_V1_1_0_CIP.ACD	Using CIP Bridging	1.1.0
6	libWiUr20_1COM_232_485_422_DEPRECATED.ACD	-	1.0.0

Warning

Controls may fail in unsafe operating conditions, causing uncontrolled operation of the controlled devices. Such hazardous events can result in death and / or serious injury and / or property damage. Therefore, there must be provided safety equipment / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer

This software (programs, function blocks, functions, etc.) does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Every user is responsible for the correct operation of his control system.

By using this software prepared by Weidmüller, you accept that Weidmüller cannot be held liable for any damage to property and / or personal injury that may occur because of the use.

Note

This software does not represent customer-specific solutions, it is simply intended to help for typical tasks. The user is responsible for the proper operation of the used products. This software does not relieve you of the obligation of safe use, installation, operation and maintenance. This software is not binding and do not claim to be complete in terms of configuration as well as any contingencies.

By using this software, you acknowledge that Weidmüller cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this software at any time without notice.

We assume no liability for this software or the information contained in this document. Our liability, for whatever legal reason, for damages caused by the use of the software or instructions described in this document is excluded.

Security notes

In order to protect equipment, systems, machines and networks against cyber threats, it is necessary to implement (and maintain) a complete state-of-the-art industrial security concept. The customer is responsible for preventing unauthorized access to his equipment, systems, machines and networks. Systems, machines and components should only be connected to the corporate network or the Internet if necessary and appropriate safeguards (such as firewalls and network segmentation) have been taken.

Contact

Weidmüller Interface GmbH & Co. KG
Klingenbergstraße 26
32758 Detmold, Germany
T +49 5231 14-0
www.weidmueller.com/contact

For further support please contact your local sales representative.

Revision history

Library Version	Date	Change log	Author
1.0.0	2022-01	FB_UR20_ModbusRtuMaster and FB_UR20_1COM_232_485_422_Control (DEPRECATED! will soon be outdated and replaced)	w010485
1.1.0	2024-09	Added: AOI_UR20_4AI_ParamEiPRead AOI_UR20_4AI_ParamEiPWrite AOI_UR20_4AO_ParamEiPRead AOI_UR20_4AO_ParamEiPWrite AOI_UR20_8DI_ParamEiPRead AOI_UR20_8DI_ParamEiPWrite AOI_UR20_8DO_ParamEiPRead AOI_UR20_8DO_ParamEiPWrite AOI_UR20_4COM_IO_LINK_ParamEiPRead AOI_UR20_4COM_IO_LINK_ParamEiPWrite AOI_UR20_4COM_IO_LINK_DeviceParamRW AOI_UR20_2AI_SG_DIAG_ParamEiPRead AOI_UR20_2AI_SG_DIAG_ParamEiPWrite	wm01448 w100141 wm01070

Contents

1.	Scope	6
2.	Standardized behavior of the function blocks.....	6
2.1.	Function block variants	6
2.2.	Basic states	6
2.3.	Inputs and Outputs	7
2.4.	Simplified behavior model.....	9
2.5.	Usage of the function blocks.....	9
3.	Commissioning u-remote in Studio 5000	11
3.1.	CIP Bridging	11
3.2.	Message Configuration for reading/writing parameters	15
3.4.	Configure process data length of u-remote module	18
3.5.	Make station parameters persist.....	21
3.6.	Inhibit communication between PLC and UR20	23
4.	UR20-4AI-UI-12 / UR20-4AI-UI-16.....	25
5.	UR20-4AO-UI-16	27
6.	UR20-8DI-P-xx	30
7.	UR20-8DO-P	33
8.	UR20-4COM-IO-LINK.....	35
8.1.	General Information.....	35
8.2.	AOI_UR20_4COM_IO_LINK_ParamEiPRead	39
8.3.	AOI_UR20_4COM_IO_LINK_ParamEiPWrite	41
8.4.	AOI_UR20_4COM_IO_LINK_DeviceParamRW	43
9.	UR20-2AI-SG-DIAG.....	46
9.1.	AOI_UR20_2AI_SG_DIAG_ParamEiPRead	46
9.2.	AOI_UR20_2AI_SG_DIAG_ParamEiPWrite	49
10.	UR20-1COM-232-485-422.....	51
10.1.	AOI_UR20_1COM_232_485_422_Control (<i>Deprecated</i>)	51
10.2.	AOI_UR20_ModBusRtuMaster (<i>Deprecated</i>)	53

1. Scope

This library contains function blocks / functions for easy use of Weidmüller u-remote modules in Rockwell Studio 5000.

2. Standardized behavior of the function blocks

All Weidmüller function blocks work in a defined manner. This behavior is defined by an internal state machine, which includes a set of standardized inputs and outputs. The following part describes the basic interfaces and behavior of the function blocks, as well as procedures to activate and deactivate the function blocks and how to handle errors.

2.1. Function block variants

There are in total four different variations of the underlying behavior.

- A level controlled function block changes state by the level of the input signals (enable and execute)
- An edge controlled function block changes state by evaluating the positive edges of its control inputs (enable, disable, execute, abort)

Both variants can either be implemented as a finite (with additional output "q_xDone") or as a continuous behavior. The finite behavior is used for any kind of action that comes to a defined end, while the continuous behavior is used for any action of an infinite nature.

2.2. Basic states

There are four basic states of a function block:

- **idle**: no action is performed (initial state of each function block).
- **standby**: the function block is initialized and ready to be executed
- **active**: the function block is performing its intended task
- **error**: an error occurred and the function block had to be stopped.

The states **standby** and **active** are separated into different sub-states since the function block may take some time to be initialized (entering state **standby**) or to safely shutdown an action (state **active**).

2.3. Inputs and Outputs

Inputs for level-triggered behavior:

Input	Description
<i>i_xEnable</i>	Set <i>true</i> to enable the FB and start the initialization. The FB is initialized and ready (standby) if <i>q_xStandby</i> is <i>true</i> and <i>q_xBusy</i> is <i>false</i> . If set to <i>false</i> , all actions are stopped immediately, the FB is disabled and possible errors are reset.
<i>i_xExecute</i>	When the FB is initialized and ready (standby), setting <i>i_xExecute</i> to <i>true</i> starts the intended task and the FB becomes active. The output <i>q_xActive</i> indicates that the FB is being executed. If the FB implements a finite behavior, <i>q_xDone</i> indicates that the task is completed. If set to <i>false</i> , the active FB is stopped in a controlled way and the FB switches back to standby state. The output <i>q_xBusy</i> is <i>true</i> during shutdown.

Inputs for edge-triggered behavior:

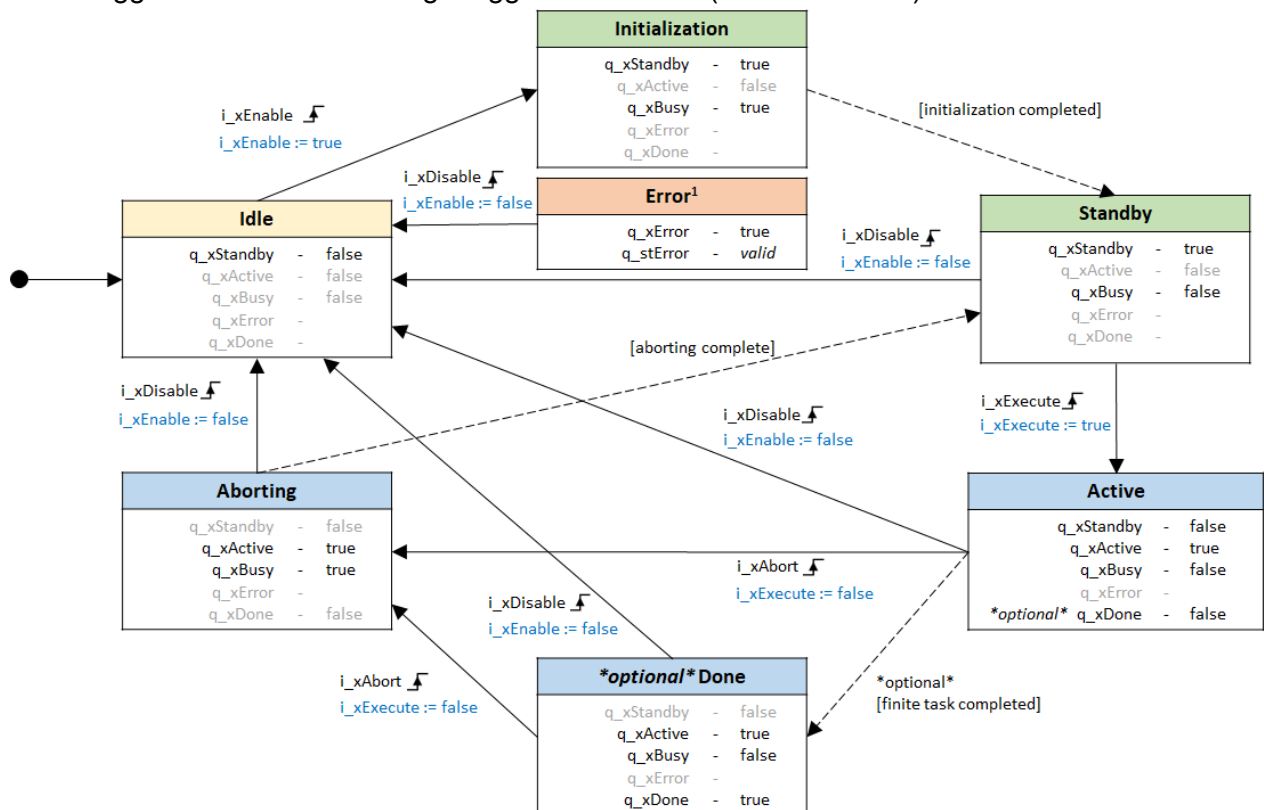
Input	Description
<i>i_xEnable</i>	A rising edge enables the FB and starts the initialization. The FB is initialized and ready (standby) if <i>q_xStandby</i> is <i>true</i> and <i>q_xBusy</i> is <i>false</i> .
<i>i_xExecute</i>	When the FB is initialized and ready (standby), a rising edge starts the intended task and the FB becomes active. The output <i>q_xActive</i> indicates that the FB is being executed. If the FB implements a finite behavior, <i>q_xDone</i> indicates that the task is completed.
<i>i_xAbort</i>	When the FB is active (<i>q_xActive</i> is <i>true</i>), a rising edge stops the FB in a controlled way. The output <i>q_xBusy</i> is <i>true</i> during shutdown. The FB switches back to standby.
<i>i_xDisable</i>	A rising edge disables the FB. All actions are stopped immediately and possible errors are reset.

Outputs:

Output	Description
<i>q_xStandby</i>	If <i>true</i> , the FB is either initializing, which may take multiple cycles (output <i>q_xBusy</i> is <i>true</i>), or already initialized and ready to be started (output <i>q_xBusy</i> is <i>false</i>)
<i>q_xActive</i>	If <i>true</i> , the FB is active. The FB is currently performing the desired task (<i>q_xDone</i> and <i>q_xBusy</i> are both <i>false</i>), or has already completed the desired, finite task (<i>q_xDone</i> is <i>true</i> and <i>q_xBusy</i> is <i>false</i>), or is currently shutting down an ongoing task in a controlled manner (<i>q_xBusy</i> is <i>true</i>)
<i>q_xBusy</i>	If <i>true</i> , the FB performs some internal tasks (shutdown or initialization) which may take multiple cycles. Wait until <i>q_xBusy</i> is false. Disabling (via inputs <i>i_xEnable</i> or <i>i_xDisable</i>) is always possible, but bypassing the usual shutdown sequence might result in undesired behaviour (depending on the specific function block).
<i>q_xError</i>	If <i>true</i> , an error has occurred and the FB is stopped.
<i>q_stError</i>	A struct that contains detailed information in case of an error (if <i>q_xError</i> is <i>true</i>). Collecting relevant error details may take some time, so the information may not be available immediately after <i>q_xError</i> indicates an error. Therefore, the output <i>q_stError.xErrorDataValid</i> is set to true as soon as all information is available in <i>q_stError</i> . Prior to that, <i>q_stError</i> might contain outdated or incorrect information.
<i>q_xDone</i>	(<i>finite behavior only</i>) If <i>true</i> , a finite task has been completed. Reset and back to Standby state (<i>q_xStandby</i>) by disabling <i>i_xExecute</i> .

2.4. Simplified behavior model

Level triggered behavior and edge triggered behavior (combined view)



Comments:

Combined view illustrating both level-triggered and edge-triggered behavior. Blue labels at transitions with inputs correspond to level-triggered behavior.

¹The state "Error" is reached from any other state in case of a general error. For reasons of simplicity, the corresponding transitions are not shown in the diagrams.

2.5. Usage of the function blocks

Activation of the function block

The activation of an FB is controlled by the two boolean inputs of the function block, *i_xEnable* and *i_xExecute*. The activation process is split into two procedures and performed in the same way for every type of behavior:

- Initialize the function block: switch from state **idle** to **standby**. By setting *i_xEnable*, parameters are validated and the block is initialized. The parameter validation takes one single plc cycle and results either into a start-up routine or an error. If needed, the start-up routine may perform some initial actions, which may take multiple plc cycles and may also include interaction with devices, parameterization, communication, etc. After the input *i_xEnable* has been set, the output *q_xStandby* changes from *false* to *true*. As long as the function block performs its initialization routine, the output *q_xBusy* stays *true*. After the initialization is finished, *q_xBusy* changes back to *false*, which indicates the function block can be executed now.

- Execute the function block and perform its intended task: switch from state **standby** to state **active**. Once the output *q_xStandby* indicates *true* and *q_xBusy* has been reset to *false*, the main operation can be started. This is always done by setting the input *i_xExecute* to *true*. The next step (internally performed) is to check whether the current process values are ok. If they are not ok, the activation results in an error state. If they are ok, the function block starts with its intended action. While this action is performed, the output *q_xActive* is *true*. After a function block that implements a finite behavior has performed its action completely, the output *q_xDone* is set to *true*. A continuous type function block stays in state **active** as long as no disable command is given.

Deactivation of the function block

Once the intended action of the function block has been done (or an ongoing action needs to be aborted), the function block needs to be deactivated. The procedure differs with regard to the implemented behaviour of the function block:

- for a level controlled function block, the input *i_xExecute* needs to be set to *false*.
- for an edge controlled function block, a positive edge on the input *i_xAbort* is required.

Once the deactivation command has been received, the function block will change to state **standby**. In some cases, a shutdown routine is implemented and while this routine is executed, the output *q_xBusy* becomes *true* to indicate that the FB is shutting down. After the shutdown procedure has been finished, the outputs *q_xActive* and *q_xBusy* (and possibly *q_xDone*) will change to *false*, and *q_xStandby* will become *true* to indicate that the function block is now again in state **standby**.

A deactivation can also be achieved by

- setting the *i_xEnable* to false (in case of a level controlled function block), or
- providing a rising edge on the input *i_xDisable* (in case of an edge controlled function block).

This way of deactivating the function block sets it back to its idle state. All outputs are set to *false* and possible errors are reset. This bypasses the usual shutdown sequence and might result in undesired behaviour (depending on the specific function block). For reactivation, the complete activation procedure is required.

Detect and reset an error

In some cases, errors might occur and the function block will switch into an error state. During this state, the error information is generated and provided via the *q_stError* output. The process of collecting error information is finished once *q_stError.xErrorDataValid* becomes *true*. While this value is *false*, any information contained in *q_stError* is not valid and should not be processed, even if *q_xError* already indicates that an error has occurred. To reset the function block after an error, it needs to be disabled (*i_xEnable* = *false* or positive edge on *i_xDisable*).

3. Commissioning u-remote in Studio 5000

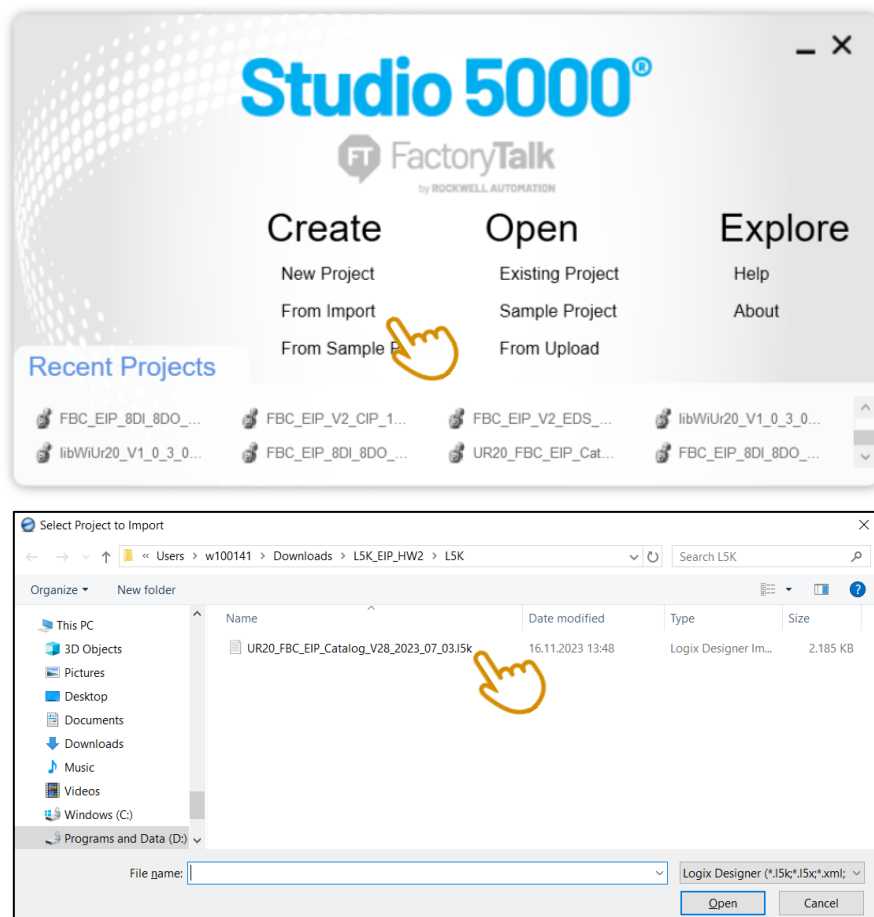
In general, commissioning of a u-remote station as a block device in Studio 5000 is described in the u-remote manual. The UR20-4COM-IO-LINK module is somewhat special in that it has a configurable process data length. Commissioning of the module in Studio 5000 is explained in the module specific manual, chp. 6.7

This chapter will focus on how to commission the UR20-4COM-IO-LINK module using CIP bridging. Explaining the pros and cons of using CIP bridging would be out of scope for this document. In the context of a third-party modular I/O system like u-remote, one of the core benefits is that you can setup your I/O configuration with all the process data as controller tags comfortably. The alternative would be to import the EDS file for the u-remote FBC, physically assemble the u-remote station and to generate the controller tags with the u-remote webserver to then import them into Studio 5000. Keep in mind that this is not a limitation specific to u-remote, but a consequence of the limited support for modular third-party I/O in Studio 5000.

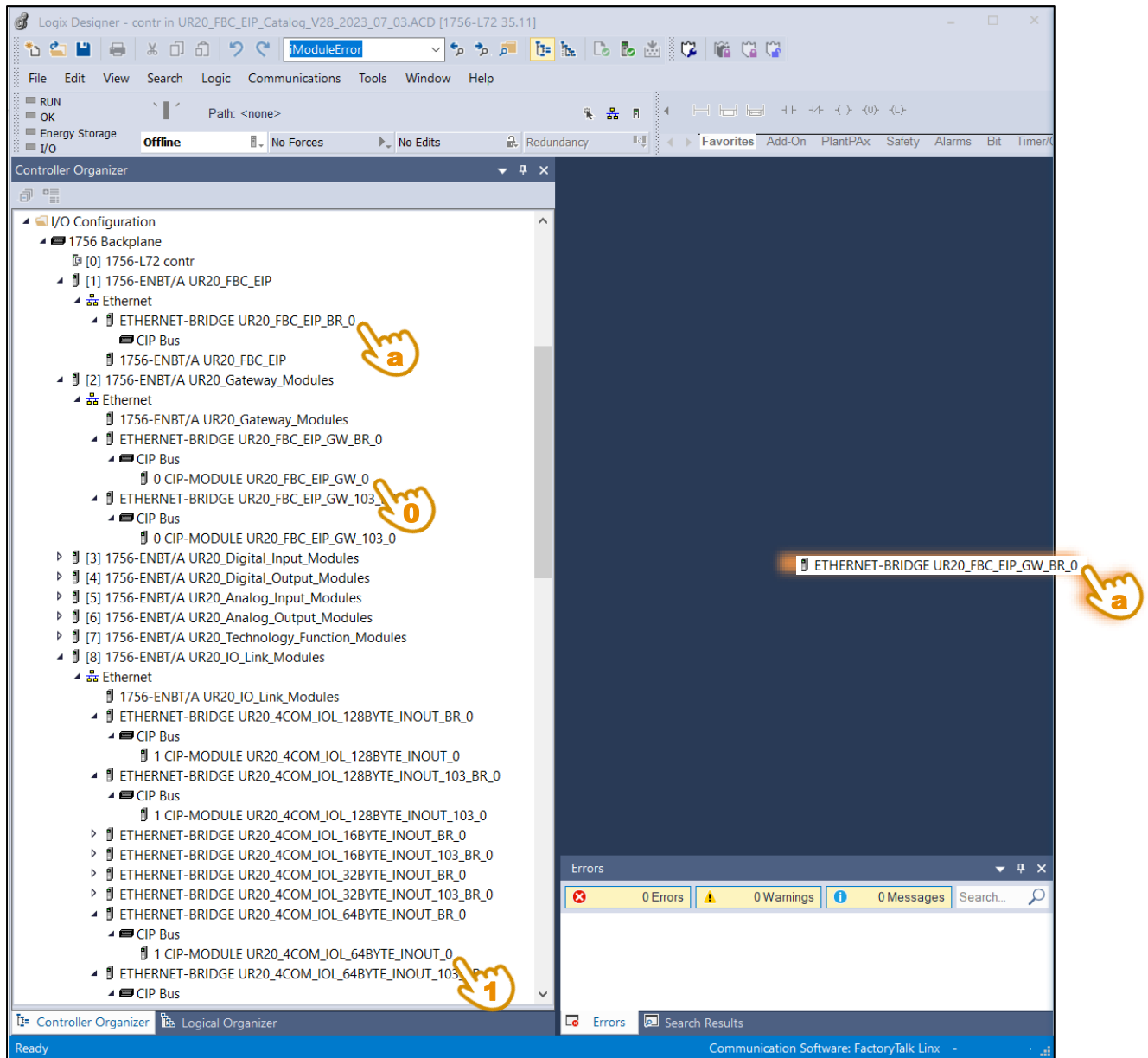
3.1. CIP Bridging

Download the CIP bridging catalog project for the u-remote FBC you're using from the [Support Center \(weidmueller.com\)](https://support.weidmueller.com).

Create a new Studio 5000 project by importing the .l5k file you downloaded:



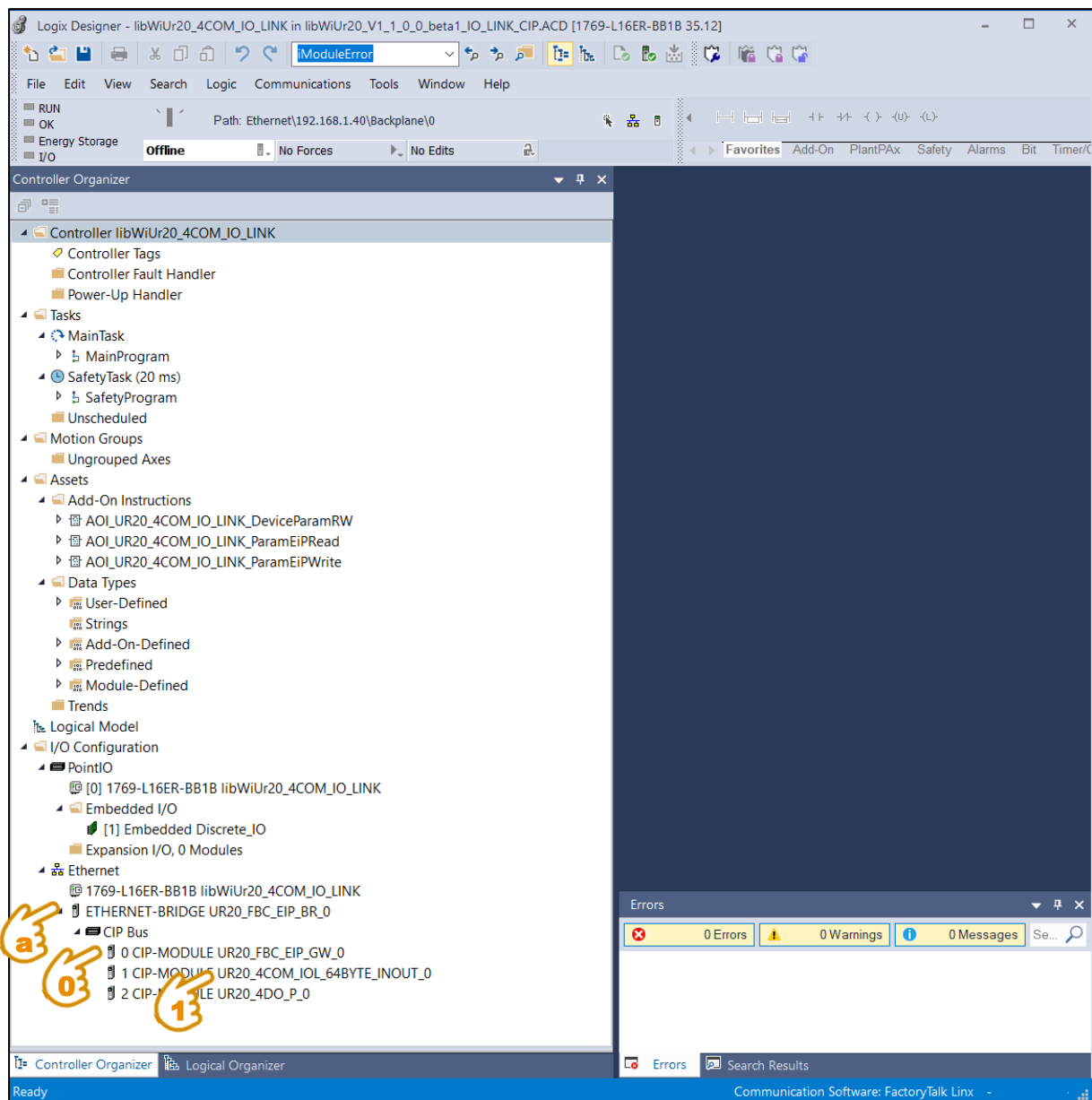
u-remote library for Rockwell Studio 5000 Logix Designer®



Drag & drop or copy-paste the components you need from the catalog project into your own project. First start with the ETHERNET-BRIDGE (a), then the CIP-MODULE for the Gateway / FBC (0), and then all the modules in the same order as on the physical station. In our example the UR20-4COM-IO-LINK is the first module (1), a UR20-4DO-P is the second module (2).

Note that there are two versions of each component, one ending in a _0 and one ending in a _103. The latter is to be used if you are using the diagnostic data of the u-remote station. Make sure to use one or the other for all the modules, do not mix _0 and _103 in the same project!

u-remote library for Rockwell Studio 5000 Logix Designer®



Also note that there are a few versions of the UR20-4COM-IO-LINK module for different process data lengths. Choose a process data length that is bigger than the sum of all process input / output data of the IO-link devices connected to the module!

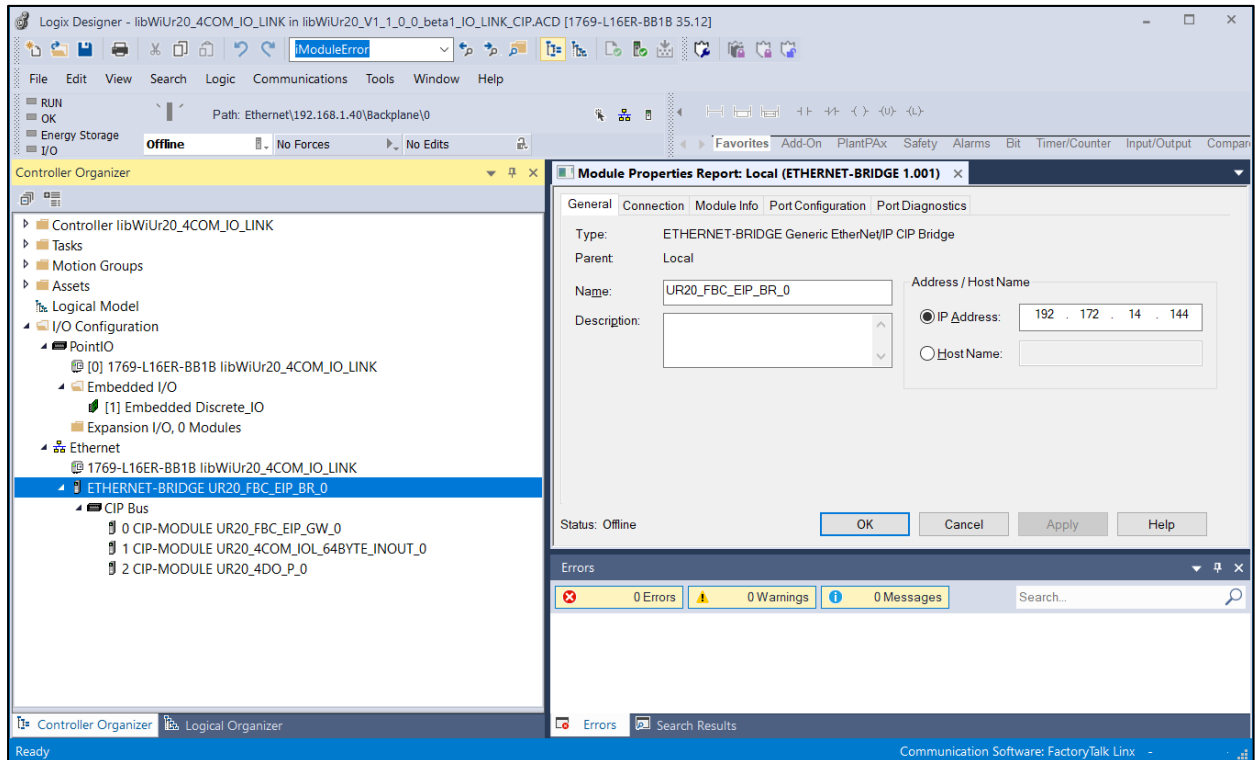
Example:

Two combined flow & temperature sensors with 16 byte input process data each, one pressure sensor with 10 byte input process data and one pneumatic manifold with 2 byte input process data and 2 byte output process data:
44 byte input process data and 2 byte output process data total.

➔ Choose the **_64BYTE_INOUT** module. **_0** without diag, or **_103** with diag.

u-remote library for Rockwell Studio 5000 Logix Designer®

Once you have copied all required modules, configure the IP address of the ethernet bridge:



Go online and download the project to the PLC. Change into Remote Run and verify that the communication between the PLC and the u-remote FBC is working.

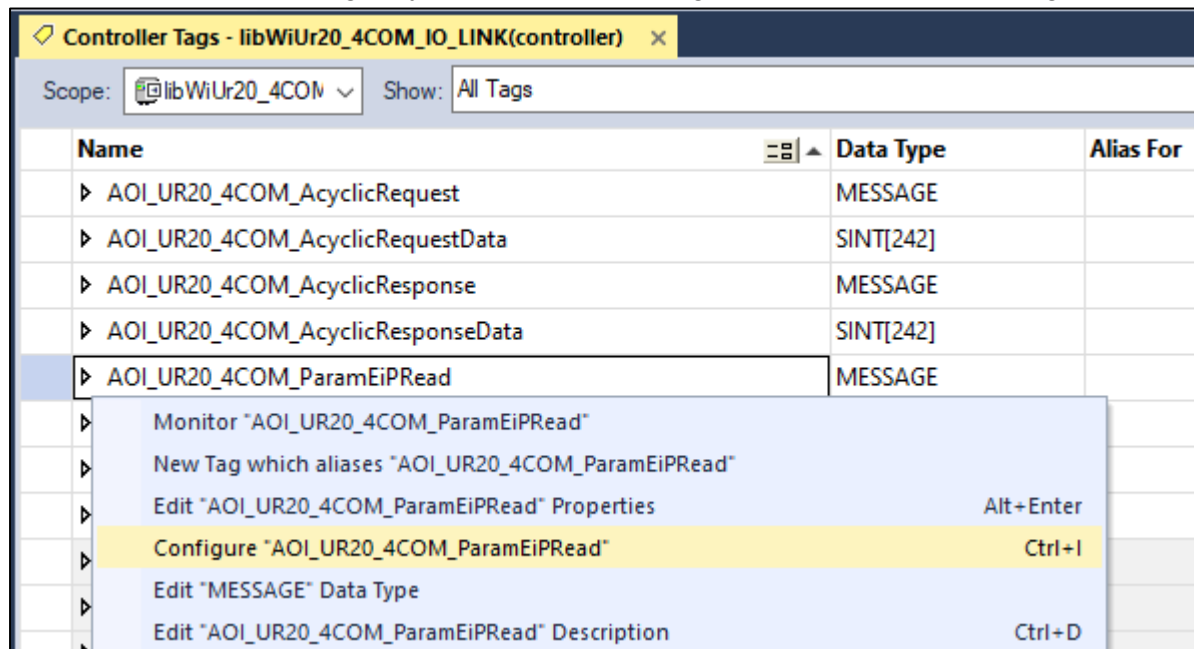
3.2. Message Configuration for reading/writing parameters

Multiple Add-On Instructions in this library read or write module or device parameters using a message instruction passed as a parameter. The message instruction must be created as a controller tag. The following example explains how to setup this instruction.

Let's assume we want to setup a message instruction with the following properties:

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single
Class	0x67
Instance	1
Attribute	0x86
Destination Element	First element of the array mapped to iq_arsiParamRead

Create a new controller tag of type MESSAGE and right click on it. Select "Configure":



Configure the message according to the table above.

Message Configuration - AOI_UR20_4COM_ParamEIPRead

Configuration Communication Tag

Message Type: CIP Generic

Service Type: Get Attribute Single

Service Code: e (Hex) Class: 67 (Hex) Instance: 1 Attribute: 86 (Hex)

Source Element: Source Length: 0 (Bytes) Destination Element: nRead_ParamData[0]

New Tag...

Enable Enable Waiting Start Done Done Length: 1

Error Code: Extended Error Code: Timed Out

Error Path: UR20_FBC_EIP_144

Error Text:

OK Cancel Apply Help

1

Message Configuration - AOI_UR20_4COM_ParamEIPRead

Configuration Communication Tag

Path: UR20_FBC_EIP_144

Browse...

Broadcast:

Communication Method

CIP DH+ Channel: A Destination Link: 0

CIP With Source ID Source Link: 0 Destination Node: 0 (Octal)

Connected Cache Connections Large Connection

Enable Enable Waiting Start Done Done Length: 1

Error Code: Extended Error Code: Timed Out

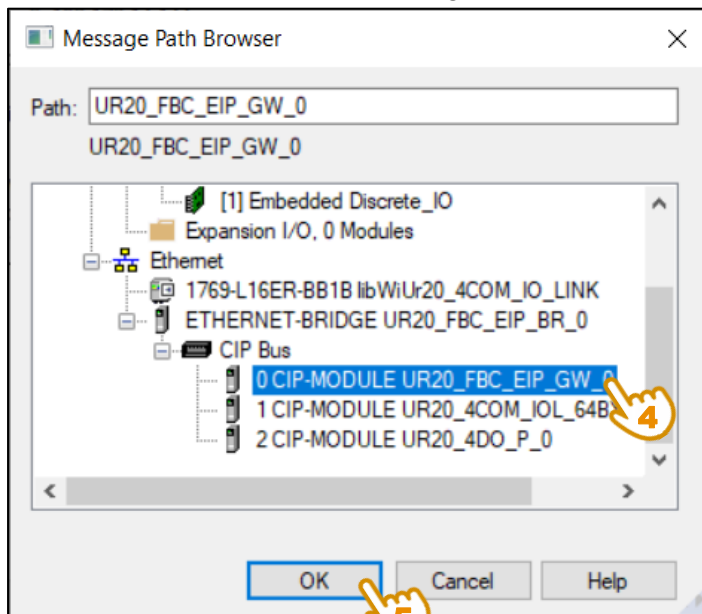
Error Path: UR20_FBC_EIP_144

Error Text:

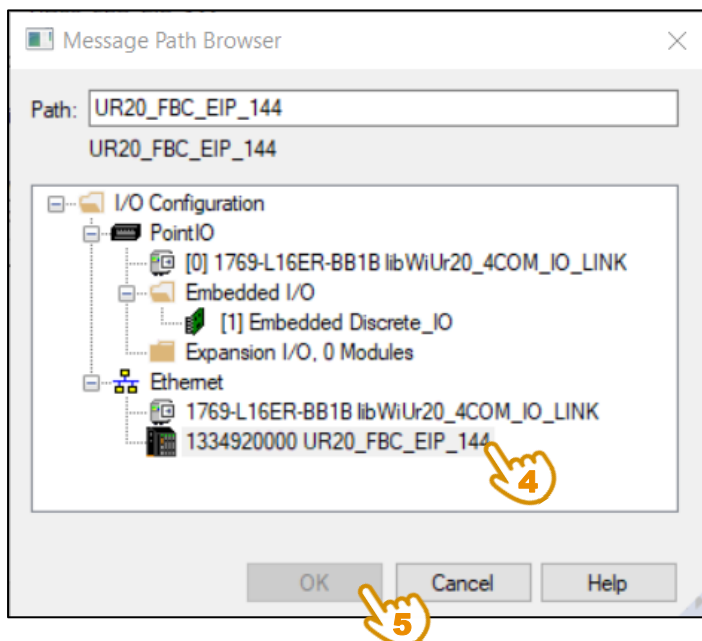
OK Cancel Apply Help

2 3 6

If you are using **CIP Bridging**, select Element 0 (the Gateway / FBC) on the CIP Bus as the destination element for the message:

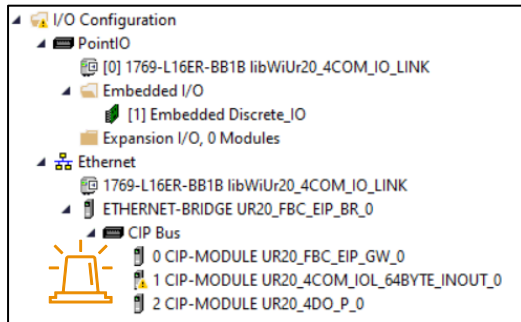


If you are **not** using **CIP Bridging** but the EDS file approach, select the FBC under the Ethernet interface as the destination element for the message:



3.4. Configure process data length of u-remote module

Certain u-remote modules like the UR20-4COM-IO-LINK feature a variable process data length for their input- and output-data. If these do not match the process data length configured in the Studio 5000 project, you might notice that the module goes into an error state when switching the PLC application to RUN:

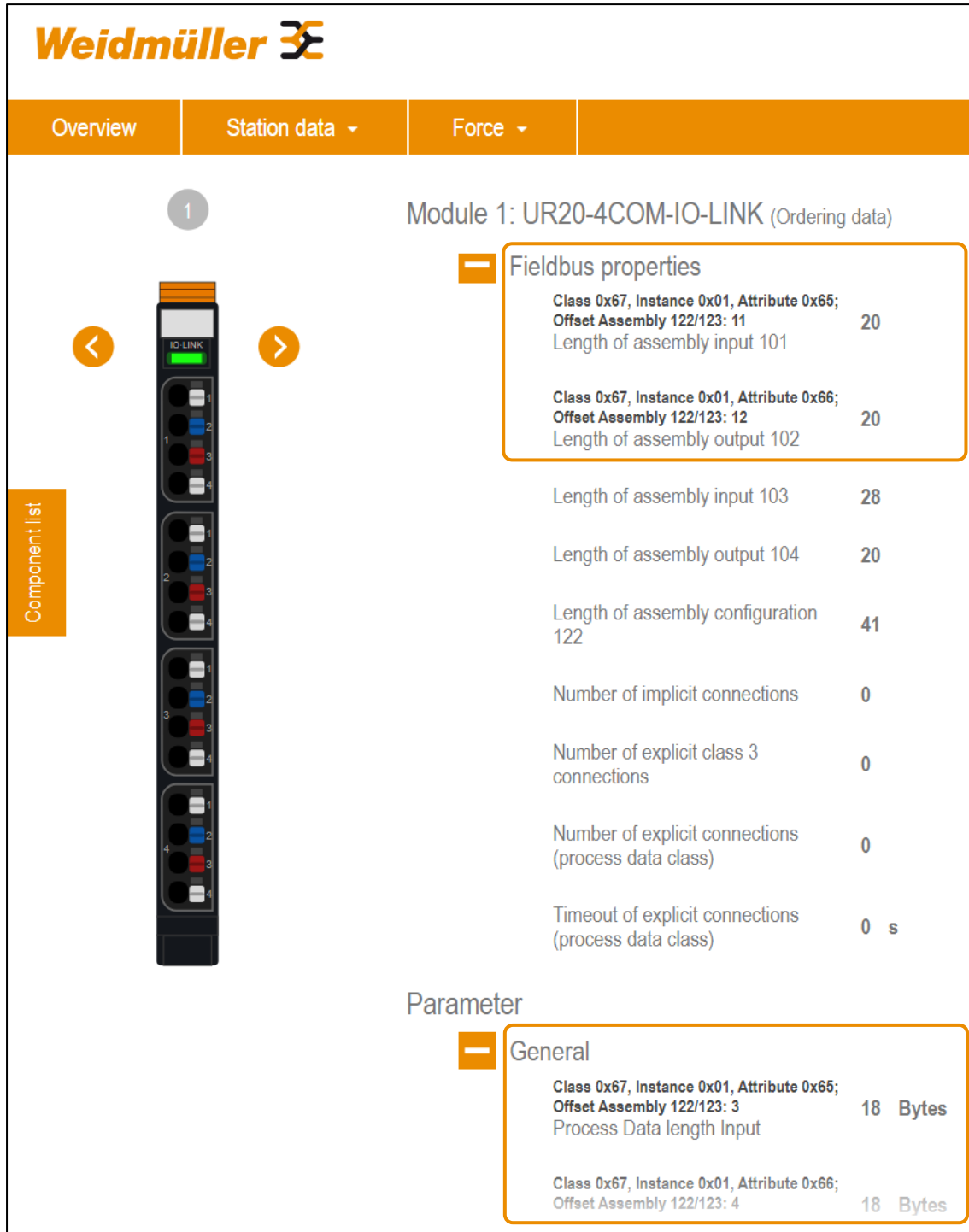


Check the status of the module in Studio 5000, or alternatively, in the u-remote webserver. Notice the EIPModMismatch under the Diagnoses menu:



While this error could also indicate an actual mismatch between the physical station and the I/O configuration in Studio 5000, in this particular case it most likely is caused by a mismatch between the process data length of the IO-link module on the station and in the project.

Click on the UR20-4COM-IO-LINK module and inspect its fieldbus properties:



Weidmüller

Overview Station data Force

1

Module 1: UR20-4COM-IO-LINK (Ordering data)

Fieldbus properties

Class 0x67, Instance 0x01, Attribute 0x65; Offset Assembly 122/123: 11 Length of assembly input 101	20
Class 0x67, Instance 0x01, Attribute 0x66; Offset Assembly 122/123: 12 Length of assembly output 102	20
Length of assembly input 103	28
Length of assembly output 104	20
Length of assembly configuration 122	41
Number of implicit connections	0
Number of explicit class 3 connections	0
Number of explicit connections (process data class)	0
Timeout of explicit connections (process data class)	0 s

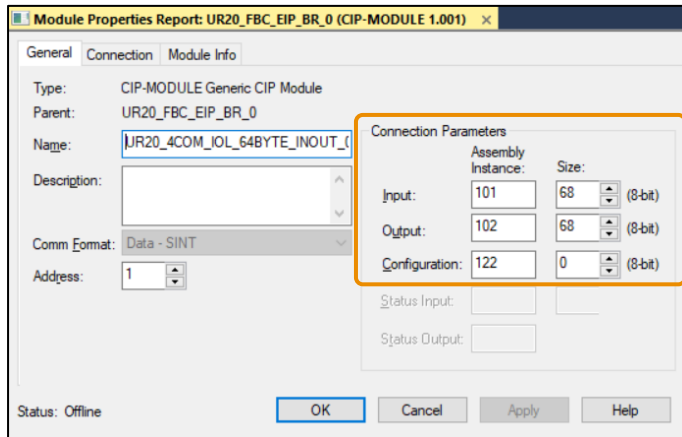
Parameter

General

Class 0x67, Instance 0x01, Attribute 0x65; Offset Assembly 122/123: 3 Process Data length Input	18 Bytes
Class 0x67, Instance 0x01, Attribute 0x66; Offset Assembly 122/123: 4	18 Bytes

Note that the module is configured for a process data length of 18 bytes for input and output. This is the default setting.

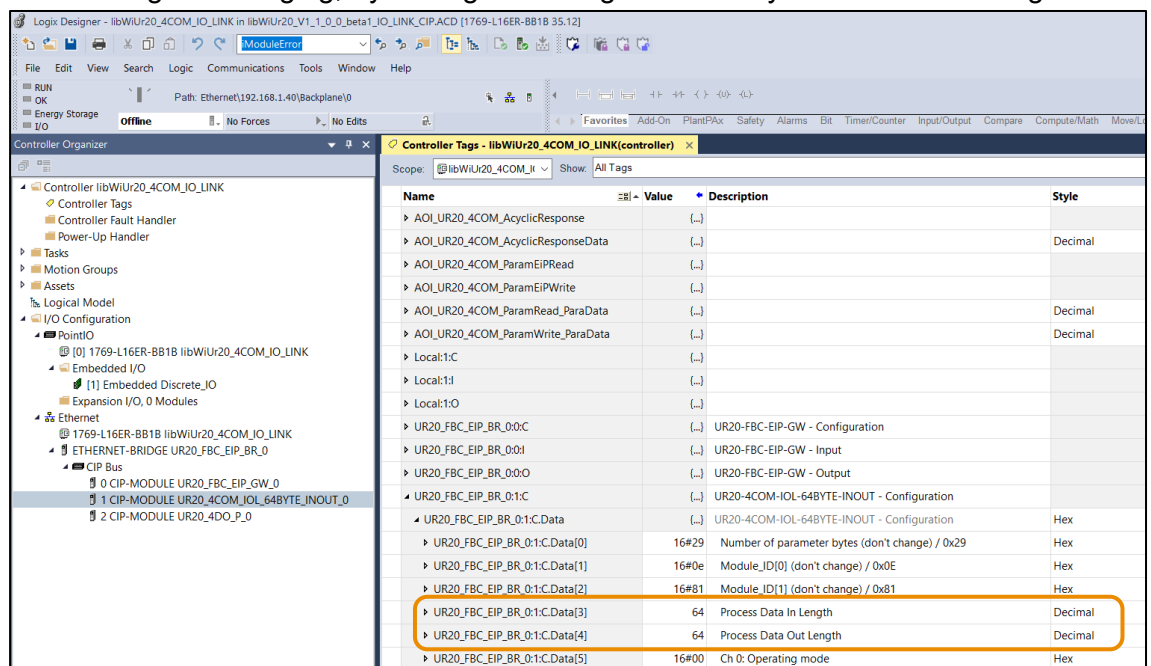
Compare this to the connection parameters of the module in the Studio 5000 project:



To fix this discrepancy, the module needs to be configured to the correct process data length.

This can be done by

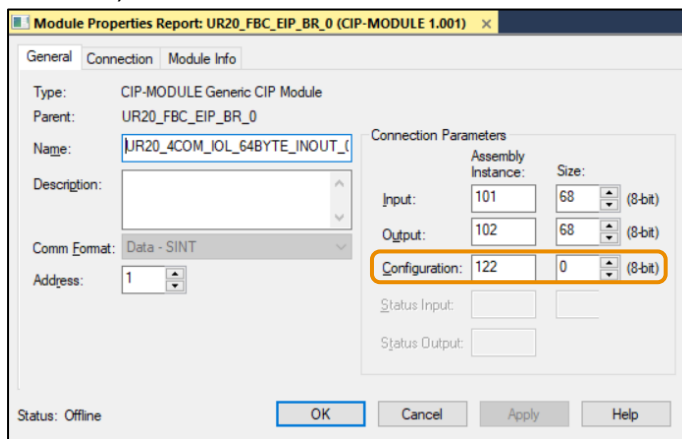
1. Sending a “Set Attribute Single” message with the correct value (i.e. 64) to Class 0x67, Instance 0x01 and Attribute 0x65 (process data length input) and Attribute 0x66 (process data length output) each.
2. Setting the attribute with the Class Instance Editor of RSNetworkx as described in chp. 6.7 of the UR20-4COM-IO-LINK manual
3. Setting the module parameters via the AOI described in chp. 8.3 of this document
4. When using CIP bridging, by editing the configuration array in the controller tags:



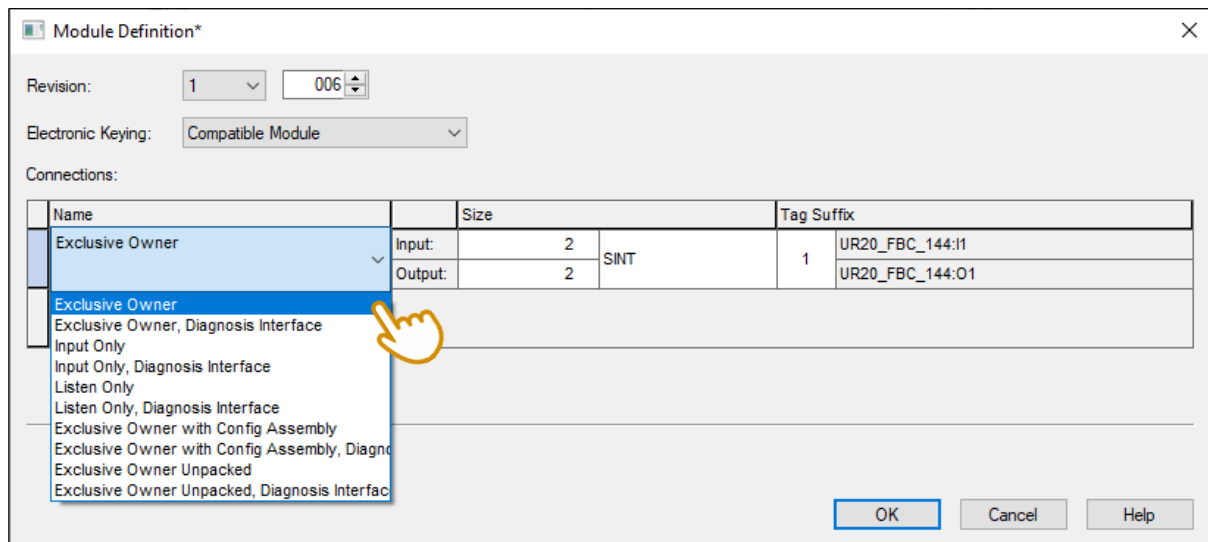
3.5. Make station parameters persist

Note that any changes made to the u-remote stations parameters via methods 1. - 3. described above are not permanent. **When the station is power cycled the parameters will be set to default values.** Method 4. does not apply the parameters permanently to the station either but will apply them automatically each time a connection between the station and the PLC is established.

It is important to note that this behavior is the default. If you do not want the stations parameters to be overwritten by whatever is defined in the controller tags as described in 4. above, set the length of the configuration assembly to 0 for the respective module (or all modules):

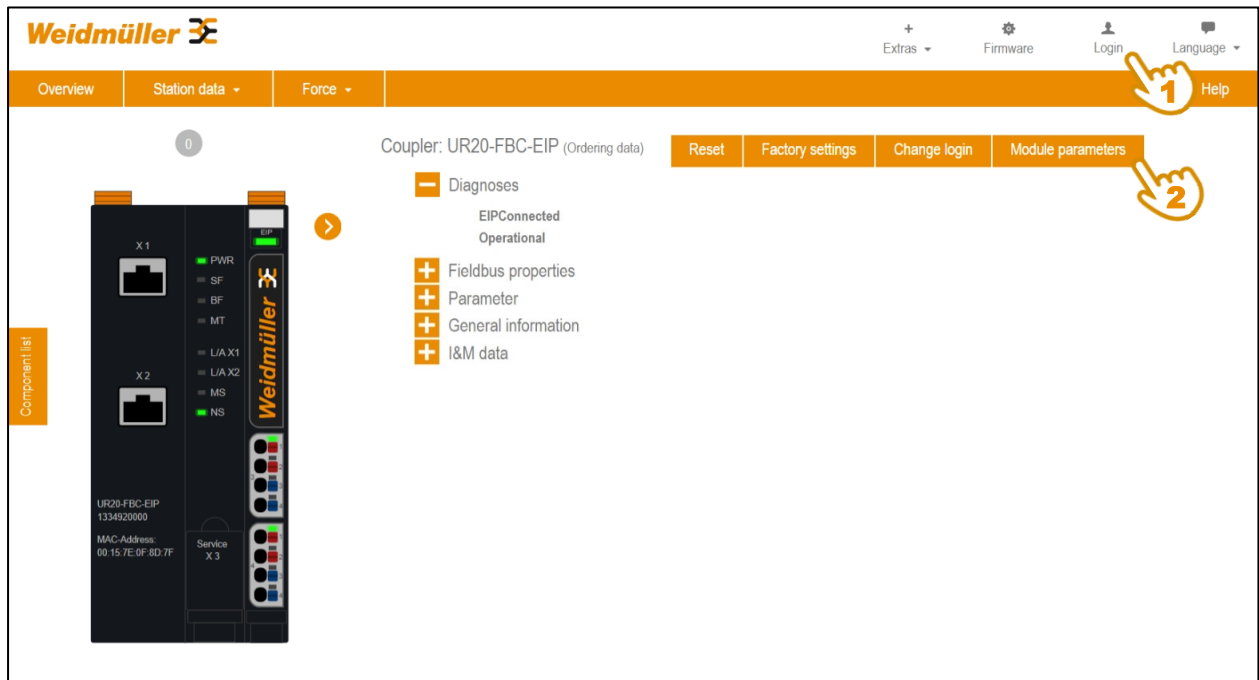


When using the u-remote coupler as a block device (not CIP bridging), select a connection mode without config assembly to prevent the PLC from overwriting the couplers configuration:

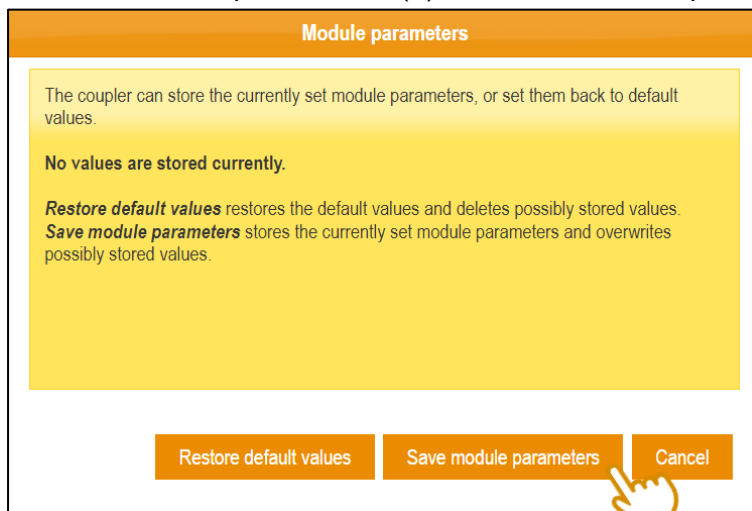


u-remote library for Rockwell Studio 5000 Logix Designer®

The parameters of all modules can be stored permanently in the coupler via the u-remote webserver. First hit login and enter your credentials (default: user “admin”, password “Detmold”).



Then hit “Module parameters” (2) and “Save module parameters” in the popup dialog.



If logging in fails due to an active fieldbus connection, either disconnect the PLC from the FBC, or use the inhibit feature of Studio 5000 as described below.

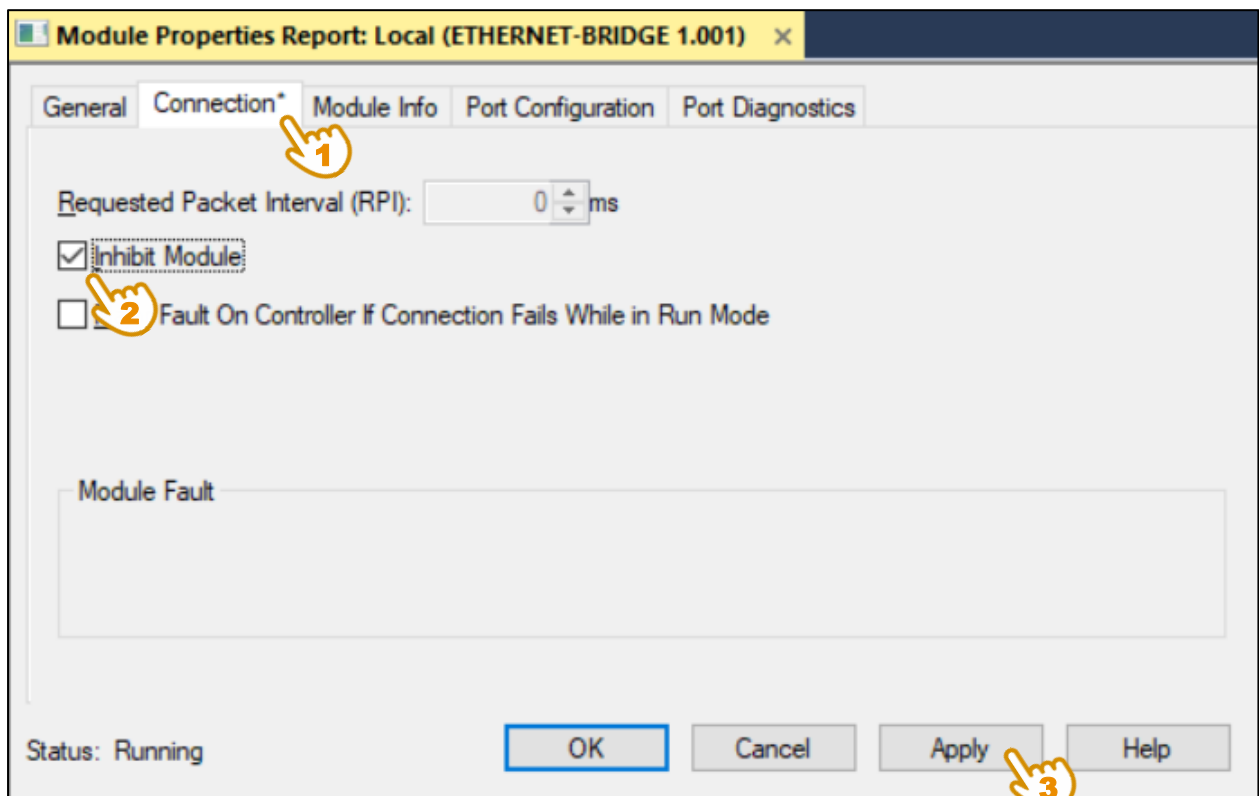
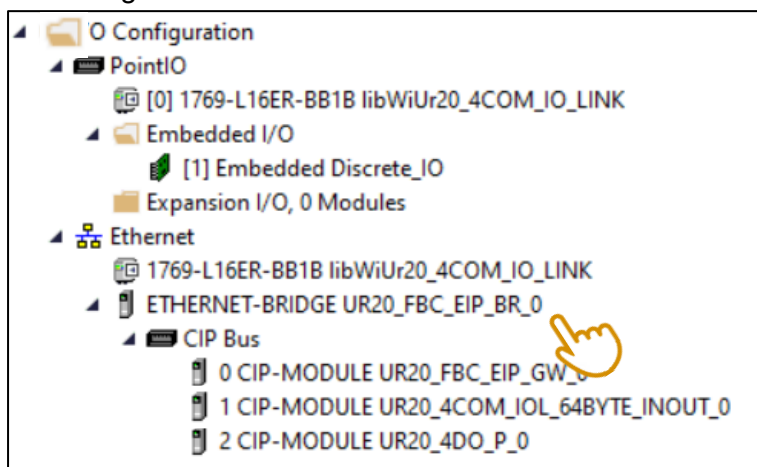
3.6. Inhibit communication between PLC and UR20

When first commissioning or while troubleshooting a u-remote station, it can be necessary to change some parameters using the u-remote webserver or the u-mation configurator. Both tools require you to login to the u-remote station.

Logging in will fail if the u-remote station is connected to the fieldbus, to prevent changes being made during operation of a machine or plant.

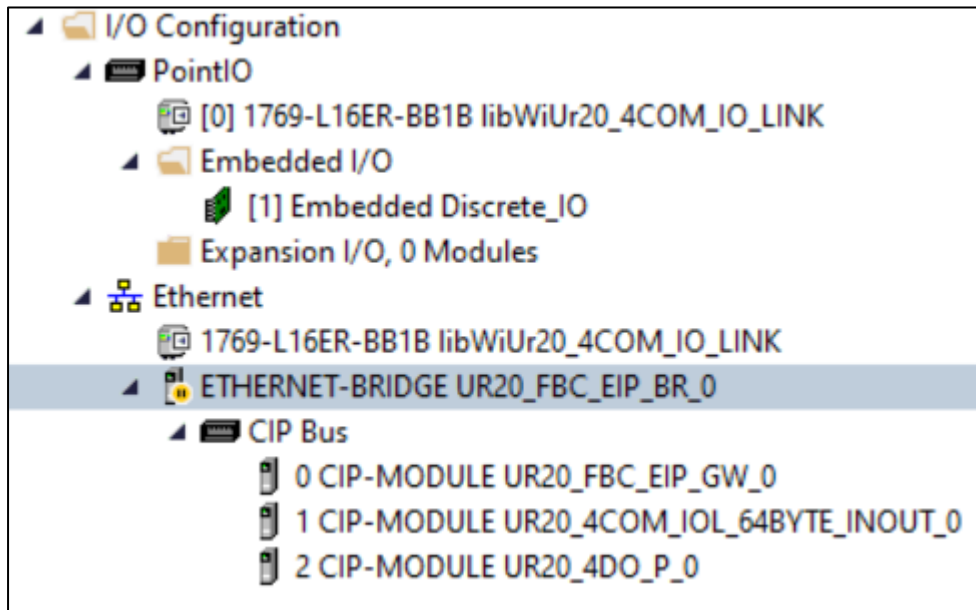
To avoid unplugging the PLC or the u-remote from the network, you can utilize Studio 5000's inhibit feature. This will stop the communication between the PLC and the u-remote station, without any changes to the network.

If you are using CIP Bridging, navigate to the ETHERNET-BRIDGE UR20_FBC_EIP... in your I/O Configuration in Studio 5000:



u-remote library for Rockwell Studio 5000 Logix Designer®

Notice that the Ethernet Bridge now has a pause icon next to it in the I/O configuration:



Now you can login using the u-remote webserver or the u-mation configurator.

4. UR20-4AI-UI-12 / UR20-4AI-UI-16

The library contains two AOIs for the four-channel analog input modules:

- AOI_UR20_4AI_ParamEiPRead
- AOI_UR20_4AI_ParamEiPWrite

Functional Description

The addon instruction allows to read / write the parameter from / to the module UR20-4AI-UI-12 or UR20-4AI-UI-16. For this the addon instruction must be imported, as well as the corresponding UDT and the message instruction. The corresponding message instruction must be set to the correct EiP-Endpoint. After enabling the function block, it is ready for its action. Once the execute input is set to true, the request to the module is send and the function block waits on the module response. Once the response has been received, the function block indicates done. If there is no response or invalid data from the module, the function block changes to error state. **Once input *i_xEnable* is set to false, the output arrays are reset to false or zero (read-AOI).** To avoid data loss the array data needs to be saved or processed before. The set values for the parameter data (write-AOI) can be changed during standby state before sending the next message. The communication settings can be changed during idle state before sending the next parameter to another module.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable function block for initialization
i_xExecute	BOOL	execute action
i_iModuleIndex.	SINT	Input parameter containing the module index value
i_stDeviceParam (ParamEiPWrite)	UDT	Struct contains parameter values of module
General_frequency_suppresion	SINT	general setting frequency suppression [0 – none, 1 – 50Hz, 2 – 60Hz, 3 – avarege over 16 values]
Data_format_Channel_0	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_0	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Data_format_Channel_1	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_1	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Data_format_Channel_2	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_2	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Data_format_Channel_3	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_3	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	function block is activated
q_xBusy	BOOL	function block is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	function block is in error state
q_stDeviceParam (ParamEIPRead):	UDT	Struct contains parameter values of module
General_frequency_suppresion	SINT	general setting frequency suppression [0 – none, 1 – 50Hz, 2 – 60Hz, 3 – avarege over 16 values]
Data_format_Channel_0	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_0	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Data_format_Channel_1	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_1	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Data_format_Channel_2	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_2	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Data_format_Channel_3	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_3	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]

InOut

Name	Type	Comment
iq_arParMessage	SINT[0..9]	Parameter buffer for received values
iq_fbParMessage	MESSAGE	Message Configuration specified to the AOI and the communication endpoint

Possible Errors

Reason	Description
watchdog for read sequence expired	Parameter values cannot be read or written. Check Message Instruction settings and module index
module slot address not valid	The set slot address for the module is not in range of 1...16

Message Configuration

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single for reading / Set Attribute Single for writing
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamWrite
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

5. UR20-4AO-UI-16

The library contains two AOIs for the four-channel analog output module:

- AOI_UR20_4AO_ParamEiPRead
- AOI_UR20_4AO_ParamEiPWrite

Functional Description

The addon Instruction allows to read / write the parameter from / to the module UR20-4AO-UI-16. For this the addon instruction must be imported, as well as the corresponding UDT and the message instruction. The corresponding message instruction must be set to the correct EiP-Endpoint. After enabling the function block, it is ready for its action. Once the execute input is set to true, the request to the module is send and the function block waits on the module response. Once the response has been received, the function block indicates done. If there is no response or invalid data from the module, the function block changes to error state. **Once input *i_xEnable* is set to false, the output arrays are reset to false or zero (read-AOI).** To avoid data loss the array data needs to be saved or processed before. The set values for the parameter data (write-AOI) can be changed during standby state before sending the next message. The communication settings can be changed during idle state before sending the next parameter to another module.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable function block for initialization
i_xExecute	BOOL	execute action
i_iModuleIndex.	SINT	Input parameter containing the module index value
i_stDeviceParam (ParamEiPWrite)	UDT	Struct contains parameter values of module
Data_format_Channel_0	SINT	data format [0 – S5 data format, 1 – S7 data format]
Output_range_Channel_0	SINT	Output range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 – -10-10V, 4 – 0-5V, 5 – -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch0	SINT	Value of substitute value [0 - off, x - depending on the channels data format (S5/S7)]
Data_format_Channel_1	SINT	data format [0 – S5 data format, 1 – S7 data format]

Measurement_range_Channel_1	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch1	SINT	Value of substitute value [0 - off, x - depending on the channels data format (S5/S7)]
Data_format_Channel_2	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_2	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch2	SINT	Value of substitute value [0 - off, x - depending on the channels data format (S5/S7)]
Data_format_Channel_3	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_3	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch4	SINT	Value of substitute value [0 - off, x - depending on the channels data format (S5/S7)]

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	function block is activated
q_xBusy	BOOL	function block is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	function block is in error state
q_stDeviceParam (ParamEiPRead):	UDT	Struct contains parameter values of module
Data_format_Channel_0	SINT	data format [0 – S5 data format, 1 – S7 data format]
Output_range_Channel_0	SINT	Output range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch0	SINT	Value of substitute value [0 - off, x - depending on the channels data format (S5/S7)]
Data_format_Channel_1	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_1	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 - -10-10V, 4 – 0-5V, 5 - -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch1	SINT	Value of substitute value [0 - off, x -

		depending on the channels data format (S5/S7)]
Data_format_Channel_2	SINT	data format [0 – S5 data format, 1 – S7 data format]
Measurement_range_Channel_2	SINT	measurement range [0 – 0-20mA, 1 – 4-20mA, 2 – 0-10V, 3 – -10-10V, 4 – 0-5V, 5 – -5-5V, 6 – 1-5V, 7 – 2-10V, 8 – deactivated]
Substitute_value_Ch2	SINT	Value of substitute value [0 - off, x - depending on the channels data format (S5/S7)]

InOut

Name	Type	Comment
iq_arParMessage	SINT[0..9]	Parameter buffer for received values
iq_fbParMessage	MESSAGE	Message Configuration specified to the AOI and the communication endpoint

Possible Errors

Reason	Description
watchdog for read sequence expired	Parameter values cannot be read or written. Check Message Instruction settings and module index
module slot address not valid	The set slot address for the module is not in range of 1...16

Message Configuration

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single for reading / Set Attribute Single for writing
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamWrite
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

6. UR20-8DI-P-xx

The library contains two AOIs for the eight-channel digital input module:

- AOI_UR20_8DI_ParamEiPRead
- AOI_UR20_8DI_ParamEiPWrite

Functional Description

The addon Instruction allows to read / write the Parameter from / to the module UR20-8DI-P-xx. For this the addon instruction must be imported, as well as the corresponding UDT and the message instruction. The corresponding message instruction must be set to the correct EiP-Endpoint. After enabling the function block, it is ready for its action. Once the execute input is set to true, the request to the module is send and the function block waits on the module response. Once the response has been received, the function block indicates done. If there is no response or invalid data from the module, the function block changes to error state.

Once input *i_xEnable* is set to false, the output arrays are reset to false or zero (read-AOI). To avoid data loss the array data needs to be saved or processed before. The set values for the parameter data (write-AOI) can be changed during standby state before sending the next message. The communication settings can be changed during idle state before sending the next parameter to another module.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable function block for initialization
i_xExecute	BOOL	execute action
i_iModuleIndex.	SINT	Input parameter containing the module index value
i_stDeviceParam (ParamEiPWrite)	UDT	Struct contains parameter values of module
Switch_on_delay_Ch0	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch1	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch2	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch3	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch4	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch5	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch6	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch7	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	function block is activated
q_xBusy	BOOL	function block is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	function block is in error state
q_stDeviceParam (ParamEIPRead):	UDT	Struct contains parameter values of module
Switch_on_delay_Ch0	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch1	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch2	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch3	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch4	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch5	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]
Switch_on_delay_Ch6	SINT	Value of switch on delay [0 - none, 1 - 0,3ms, 2 - 3ms, 3 - 10ms, 4 - 20ms, 5 - 40ms]

InOut

Name	Type	Comment
iq_arParMessage	SINT[0..8]	Parameter buffer for received values
iq_fbParMessage	MESSAGE	Message Configuration specified to the AOI and the communication endpoint

Possible Errors

Reason	Description
watchdog for read sequence expired	Parameter values cannot be read or written. Check Message Instruction settings and module index
module slot address not valid	The set slot address for the module is not in range of 1...16

Message Configuration

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single for reading / Set Attribute Single for writing
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamWrite
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

7. UR20-8DO-P

The library contains two AOIs for the eight-channel digital output module:

- AOI_UR20_8DO_ParamEiPRead
- AOI_UR20_8DO_ParamEiPWrite

Functional Description

The addon Instruction allows to read / write the Parameter from / to the module UR20-8DO-P. For this the addon instruction must be imported, as well as the corresponding UDT and the message instruction. The corresponding message instruction must be set to the correct EiP-Endpoint. After enabling the function block, it is ready for its action. Once the execute input is set to true, the request to the module is send and the function block waits on the module response. Once the response has been received, the function block indicates done. If there is no response or invalid data from the module, the function block changes to error state. **Once input *i_xEnable* is set to false, the output arrays are reset to false or zero (read-AOI).** To avoid data loss the array data needs to be saved or processed before. The set values for the parameter data (write-AOI) can be changed during standby state before sending the next message. The communication settings can be changed during idle state before sending the next parameter to another module.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable function block for initialization
i_xExecute	BOOL	execute action
i_iModuleIndex.	SINT	Input parameter containing the module index value
i_stDeviceParam (ParamEiPWrite)	UDT	Struct contains parameter values of module
Substitute_value_Ch0	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch1	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch2	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch3	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch4	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch5	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch6	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch7	SINT	Value of substitute value [0 - off, 1 - on]

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	function block is activated
q_xBusy	BOOL	function block is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	function block is in error state
q_stDeviceParam (ParamEiPRead):	UDT	Struct contains parameter values of module

Substitute_value_Ch0	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch1	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch2	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch3	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch4	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch5	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch6	SINT	Value of substitute value [0 - off, 1 - on]
Substitute_value_Ch7	SINT	Value of substitute value [0 - off, 1 - on]

InOut

Name	Type	Comment
iq_arParMessage	SINT[0..8]	Parameter buffer for received values
iq_fbParMessage	MESSAGE	Message Configuration specified to the AOI and the communication endpoint

Possible Errors

Reason	Description
watchdog for read sequence expired	Parameter values cannot be read or written. Check Message Instruction settings and module index
module slot address not valid	The set slot address for the module is not in range of 1...16

Message Configuration

The Add-On Instruction reads / writes the module parameters using a message instruction passed to it as the parameter iq_fbParMessage. The message instruction must be created as a controller tag and setup as follows:

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single for reading / Set Attribute Single for writing
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamWrite
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

8. UR20-4COM-IO-LINK

The library contains three AOIs for the four-channel io-link master module. As this is a rather complex function module, please take care to first read the following information before using the AOIs.

8.1. General Information

Definitions

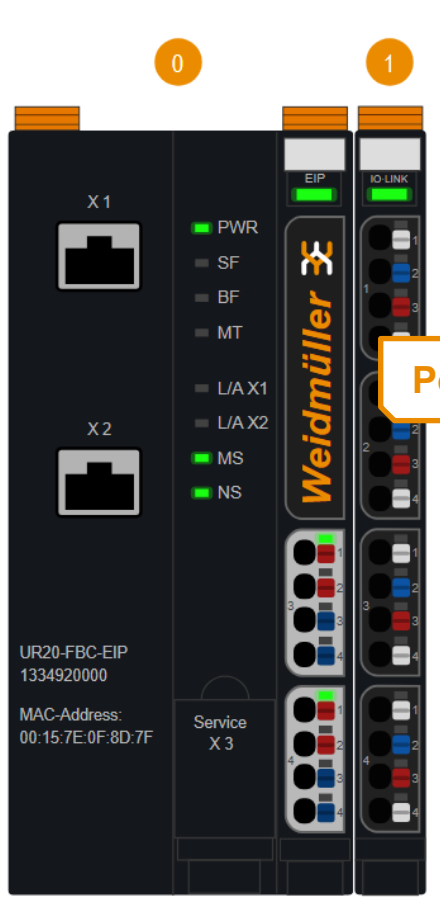
IO-link specific

Parameters of an IO-link device are addressed by a combination of index and subindex, usually denoted as index:subindex, i.e. 235:0 or 0xEB:0x0.

These addresses are part of the IODD file of each device. Most manufacturers also publish them in their device's manual or datasheet.

u-remote specific

Module Index or Slot: The position of the module. 0 is the FBC, 1 the leftmost module
Port Index or Channel: Port 1 is the top port, shown in the webserver as channel 0.



Module 1: UR20-4COM-IO-LINK

- + General information

Parameter

- General
 - Class 0x67, Instance 0x01, Attribute 0x65; Offset Assembly 122/123: 11
Process Data length Input
 - Class 0x67, Instance 0x01, Attribute 0x66; Offset Assembly 122/123: 12
Process Data length Output
- Channel 0
 - Class 0x67, Instance 0x01, Attribute 0x67; Offset Assembly 122/123: 13.0
Operating mode
 - Class 0x67, Instance 0x01, Attribute 0x68; Offset Assembly 122/123: 13.4
Port cycle
 - Class 0x67, Instance 0x01, Attribute 0x69; Offset Assembly 122/123: 17
Port cycle time
 - Class 0x67, Instance 0x01, Attribute 0x6A; Offset Assembly 122/123: 13.6
IO-Link device check

Port 1 = Channel 0

Introduction

The UR20-4COM-IO-LINK is a communications module for u-remote IP20 and u-control. It can exchange cyclic process data as well as acyclic parameter data with IO-link devices like temperature, pressure and flow sensors, pneumatic valve manifolds or Weidmüller's PRO-COM IO-link module for the topGUARD range of intelligent DC load monitoring devices or PROtop 24V power supplies.

Just like most other UR20 modules, the UR20-4COM-IO-LINK has module parameters which can be edited using the u-remote webserver. Refer to the u-remote webserver manual for instructions on how to edit these using a web browser. Refer to the module specific manual for a complete list of parameters, general electrical characteristics, and relevant information for commissioning of the module, including the use of the u-mation configurator. The u-mation configurator (not to be confused with the Weidmüller configurator for rail assemblies) is an easy to use tool to parametrize IO-link devices connected to a UR20-4COM-IO-LINK module.

Please find these documents in the [Support Center \(support.weidmueller.com\)](http://support.weidmueller.com)

This library documentation describes how to use AOIs to configure the module as well as IO-link devices connected to it.

The library contains three AOIs for the UR20-4COM-IO-LINK module:

1. AOI_UR20_4COM_IO_LINK_ParamEiPRead
2. AOI_UR20_4COM_IO_LINK_ParamEiPWrite
3. AOI_UR20_4COM_IO_LINK_DeviceParamRW

No. 1 is used to read the parameters of the module, while no. 2 is used to write a set of parameters to the module. Both operate on the whole set of module parameters; they cannot read or write individual parameters.

No. 3 can read or write individual parameters of an IO-link device connected to the module. Parameter data is passed in raw format as an array of SINT.

Module Parameters

The parameter set is defined as a user defined datatype (UDT). It consists of the length for the input and output process data of the module and an array of UDT for the parameters of all four channels:

UDT_UR20_4COM_IO_LINK_ModuleParameter			
–	ProcessDataLengthInput	INT	[2..130 byte, default 18]
–	ProcessDataLengthOutput	INT	[2..130 byte, default 18]
–	Channel	UDT_UR20_4COM_IO_LINK_ChannelParameter[4]	

Each channel has eight parameters:

UDT_UR20_4COM_IO_LINK_ChannelParameter			
– OperatingMode	SINT	[0: disabled, 1: DO, 2: DI, 3: IO-Link]	
– PortCycle	SINT	[0: free running, 1: fixed cycle, 2: message sync]	
– PortCycleTime	INT	[4..1326]	
– IO_LinkDeviceCheck	SINT	[0: disabled, 1: type compare, 2: identical]	
– DS_ActivationState	SINT	[0: disabled, 1: enabled, 2: clear]	
– ChannelDiagnostics	SINT	[0: disabled, 1: enabled]	
– ProcessDataLengthInput	SINT	[0..32 byte, 33: auto]	
– ProcessDataLengthOutput	SINT	[0..32 byte, 33: auto]	

Please refer to the UR20-4COM-IO-LINK manual for a full explanation of these parameters.

Device Parameters

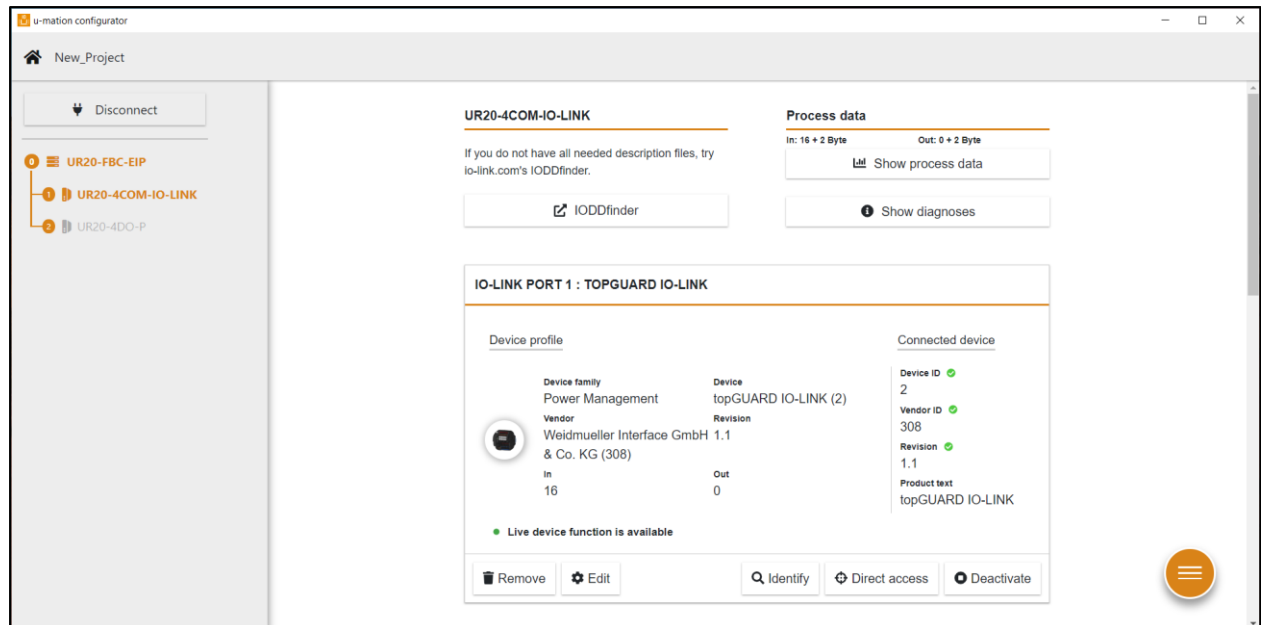
As the conversion required to derive physical values from the raw parameter data and vice versa can be different for every device and parameter, this must be implemented by the user. As of now the AOI Weidmüller provides can only read or write a single parameter during each execution.

In case the device manufacturer does not provide documentation on how to convert raw parameter data to physical values, examine the device's IODD file in an XML text editor and use the u-mation configurator as for reference when implementing your own conversions.

Edit IO-link device parameters with u-mation configurator

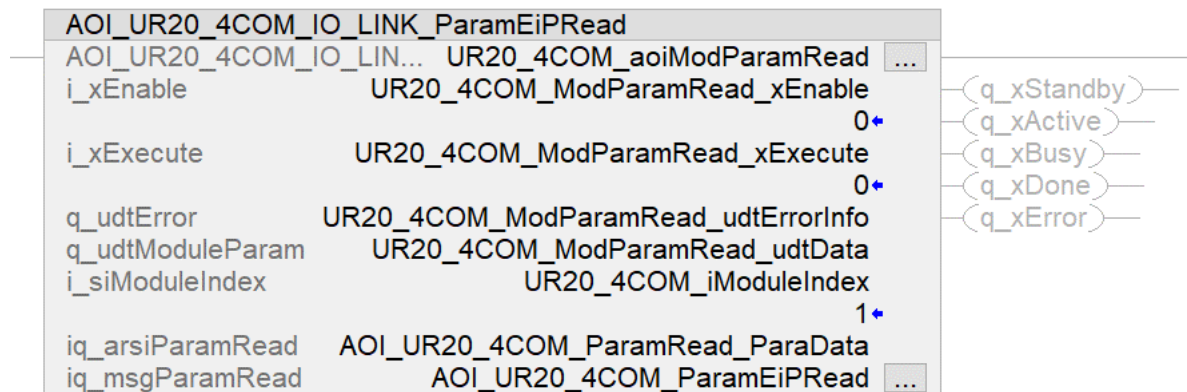
The u-mation configurator is a great tool to commission or to troubleshoot IO-link devices connected to a UR20-4COM-IO-LINK module, or to be used as a reference to compare values read via the DeviceParamRW AOI described in chp. 8.4. It can be used to read or write the parameters of any IO-link device, regardless of manufacturer. It only requires the IODD file for the device.

u-remote library for Rockwell Studio 5000 Logix Designer®



It also supports the UR20-1COM-CANOPEN module and can be used on any UR20-FBC fieldbus coupler, with the exception of the UR20-FBC-EC-ECO. The u-mation configurator is available free of charge from the [Support Center \(weidmueller.com\)](https://supportcenter.weidmueller.com).

8.2. AOI_UR20_4COM_IO_LINK_ParamEIPRead



Functional description

The Add-On Instruction allows to read the parameters of a UR20-4COM-IO-LINK module. After enabling the Add-On Instruction, it is ready for its action. Before reading the module parameters, the Module Index or slot must be selected. This can be done after the Add-On Instruction has been enabled or during idle. Once the execute input is set to true, a get attributes message is sent to the UR20 FBC and the Add-On Instruction waits on its response. Once the response has been received, the Add-On Instruction indicates done. If no response or invalid data is received, the Add-On Instruction changes to error state.

The received parameters persist when input *i_xExecute* is set to false. They are overwritten on the next execution of the Add-On Instruction.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable Add-On Instruction for initialization
i_xExecute	BOOL	execute action
i_siModuleIndex	SINT	the position of the module (slot 1..64)

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	Add-On Instruction is activated
q_xBusy	BOOL	Add-On Instruction is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	Add-On Instruction is in error state

InOut

Name	Type	Comment
q_udtModuleParam	UDT	parameters read from the module, see chp. 0
iq_arsiParamRead	SINT[64]	array with received parameter raw data
iq_msgParamRead	MSG	get attribute message
q_udtError	Struct	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
srtModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arstrParameter	STRING [0 .. 2]	additional error information

Possible Errors

Reason	Description
module slot address not valid	the i_siModuleIndex parameter is not valid, i.e. > 64
parameter read fault	the message could not be sent or produced an error
watchdog for read sequence expired	the module did not respond within a specific time during a transmission

Message Configuration

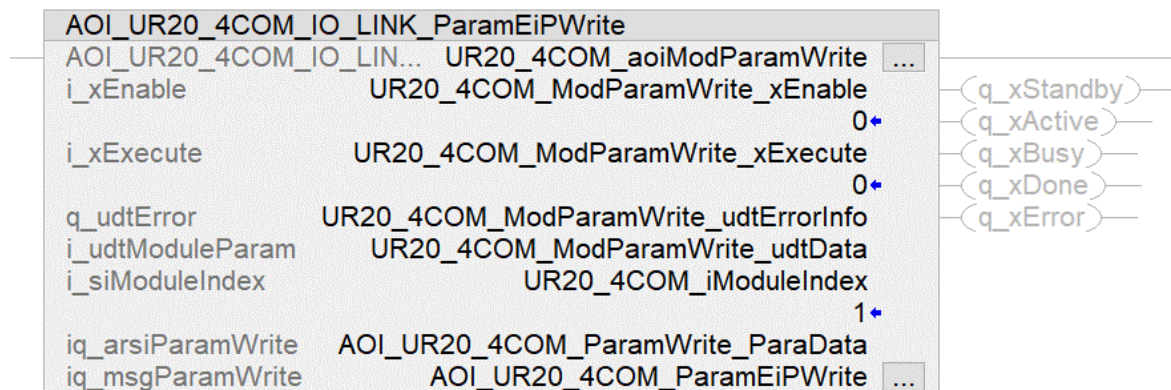
The Add-On Instruction reads the module parameters using a message instruction passed to it as the parameter iq_msgParamRead. The message instruction must be created as a controller tag and setup as follows:

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamRead
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

8.3. AOI_UR20_4COM_IO_LINK_ParamEiPWrite

This AOI allows to write parameters to a UR20-4COM-IO-LINK module.



Functional description

The Add-On Instruction allows to write parameters of a UR20-4COM-IO-LINK module. After enabling the Add-On Instruction, it is ready for its action. Before writing the module parameters, the Module Index or slot must be selected, and a valid set of parameters must be given as i_udtModuleParam. This can be done after the Add-On Instruction has been enabled or during idle. Once the execute input is set to true, a get attributes message is sent to the UR20 FBC and the Add-On Instruction waits on its response. Once the response has been received, the Add-On Instruction indicates done. If no response or an error is received, the Add-On Instruction changes to error state.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable Add-On Instruction for initialization
i_xExecute	BOOL	execute action
i_siModuleIndex	SINT	the position of the module (slot 1..64)

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	Add-On Instruction is activated
q_xBusy	BOOL	Add-On Instruction is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	Add-On Instruction is in error state

InOut

Name	Type	Comment
i_udtModuleParam	UDT	parameters to write to the module, see chp. 0
iq_arsiParamWrite	SINT[64]	array with parameter raw data to write
iq_msgParamWrite	MSG	set attribute message
q_udtError	Struct	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
srtModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arstrParameter	STRING [0 .. 2]	additional error information

Possible Errors

Reason	Description
module slot address not valid	the i_siModuleIndex parameter is not valid, i.e. > 64
parameter write fault	the message could not be sent or produced an error
invalid parameter	one or more parameter values are invalid
watchdog for write sequence expired	the module did not respond within a specific time during a transmission

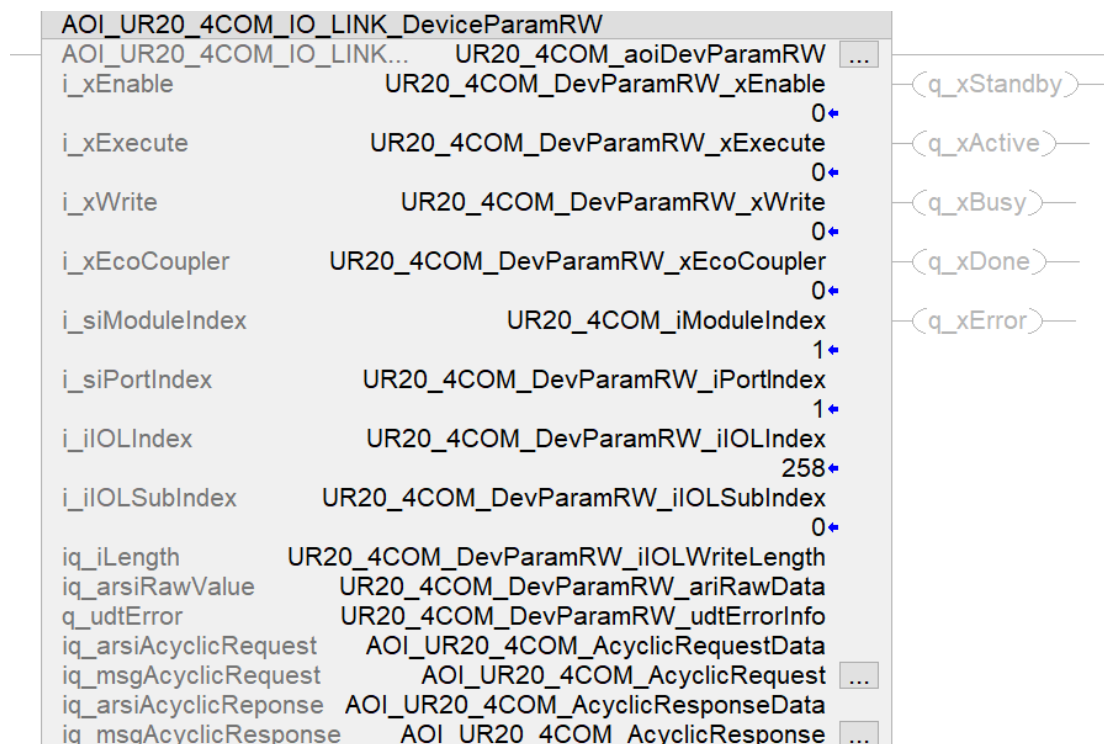
Message Configuration

The Add-On Instruction writes the module parameters using a message instruction passed to it as the parameter iq_msgParamWrite. The message instruction must be created as a controller tag and setup as follows:

Property	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamWrite
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

8.4. AOI_UR20_4COM_IO_LINK_DeviceParamRW



Functional Description

The Add-On Instruction allows to read or write parameters of an IO-link device connected to a UR20-4COM-IO-LINK module. After enabling the Add-On Instruction, it is ready for its action. Before reading or writing the device parameter, all inputs must be set. When writing a parameter, the byte array iq_arsiRawValue must be set to the desired parameter raw value, see chp. 0. This can be done after the Add-On Instruction has been enabled or during idle. Once the execute input is set to true, an acyclic request is sent as a set attribute message to the UR20 FBC and the Add-On Instruction waits on its response. Once the response has been received, the Add-On Instruction cyclically polls the acyclic response attribute using a get attribute message. When it receives a valid response, it indicates done. If no response or an error is received, the Add-On Instruction changes to error state.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable Add-On Instruction for initialization
i_xExecute	BOOL	execute action
i_xWrite	BOOL	set to true when writing a parameter
i_xEcoCoupler	BOOL	set to true if an ECO coupler is used
i_siModuleIndex	SINT	the position of the module (slot 1..64)
i_siPortIndex	SINT	the port the IO-link device is connected to
i_iOLIndex	INT	the index of the parameter
i_iOLSubIndex	INT	the subindex of the parameter

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	Add-On Instruction is activated
q_xBusy	BOOL	Add-On Instruction is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	Add-On Instruction is in error state

InOut

Name	Type	Comment
iq_iLength	INT (0..232)	length of the parameter in byte
iq_arsiRawValue	SINT[232]	parameter read from / write to the device, see chp. 0
iq_arsiAcyclicRequest	SINT[242]	array with acyclic request data
iq_msgAcyclicRequest	MSG	acyclic request message
iq_arsiAcyclicResponse	SINT[242]	array with acyclic response data
iq_msgAcyclicResponse	MSG	acyclic response message
q_udtError	Struct	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
srtModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arstrParameter	STRING [0 .. 2]	additional error information

Possible Errors

Reason	Description
module slot address not valid	the i_siModuleIndex parameter is not valid, i.e. > 64
request message produced an error	the message could not be sent or produced an error
response message content invalid	a response was received, but its contents are invalid
response message produced an error	no response was received or receiving it produced an error
response message received an error from the module. double-check your request!	the module responded with an error. double check the parameter index and sub-index as well as the raw value!
watchdog for read sequence expired	the module did not respond within a specific time during a transmission

Message Configuration

The Add-On Instruction writes the device parameters using a set of two message instructions passed to it as the parameters `iq_msgAcyclicRequest` and `iq_msgAcyclicResponse`. The message instructions must be created as a controller tag and setup as follows:

Acyclic Request

Property	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
Class	0x64
Instance	1
Attribute	0x78
Source Element	First element of the array mapped to <code>iq_arsiAcyclicRequest</code>
Source Length	10
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Message Configuration - AOI_UR20_4COM_AcyclicRequest

Configuration Communication Tag

Message Type: CIP Generic

Service Type: Set Attribute Single Source Element: .AcyclicRequestData

Source Length: 10 (Bytes)

Service Code: 10 (Hex) Class: 64 (Hex) Destination Element:

Instance: 1 Attribute: 78 (Hex)

New Tag...

Acyclic Response

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single
Class	0x64
Instance	1
Attribute	0x78
Destination Element	First element of the array mapped to <code>iq_arsiAcyclicResponse</code>
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Message Configuration - AOI_UR20_4COM_AcyclicResponse

Configuration Communication Tag

Message Type: CIP Generic

Service Type: Get Attribute Single Source Element:

Source Length: 0 (Bytes)

Service Code: e (Hex) Class: 64 (Hex) Destination Element: AOI_UR20_4COM_Ai

Instance: 1 Attribute: 78 (Hex)

New Tag...

Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

9. UR20-2AI-SG-DIAG

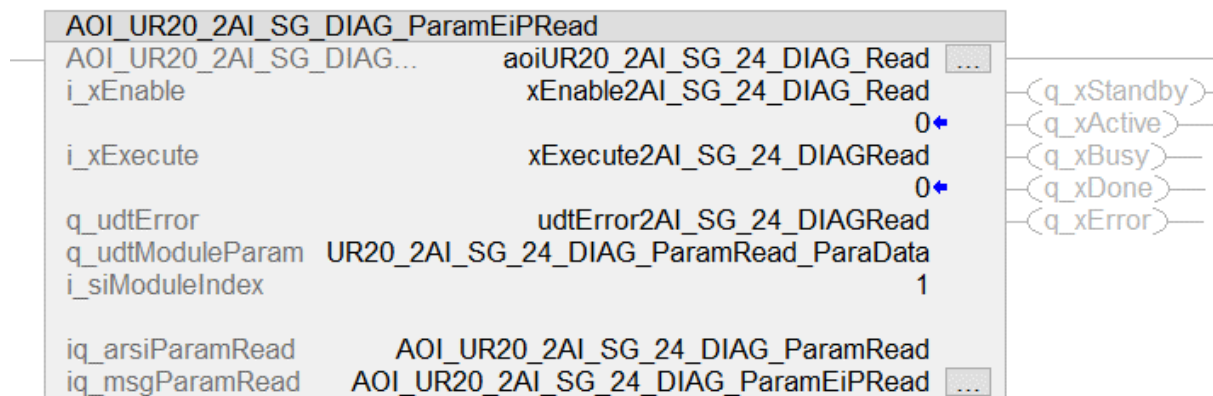
The library contains two AOIs for the two-channel strain gauge input module:

- AOI_UR20_2AI_SG_DIAG_ParamEiPRead
- AOI_UR20_2AI_SG_DIAG_ParamEiPWrite

In addition, the Studio 5000 project contains a basic example in Ladder Logic on how to calibrate the module.

Please note that there is a known issue with the UR20-FBC-EIP (HW2), FW 2.14. Use FW 2.11 or a more recent version if you notice parameter write errors.

9.1. AOI_UR20_2AI_SG_DIAG_ParamEiPRead



Functional description

The Add-On Instruction allows to read parameters of a UR20-2AI-SG-DIAG module. After enabling the Add-On Instruction, it is ready for its action. Before reading the module parameters, the Module Index or slot must be selected. This can be done after the Add-On Instruction has been enabled or during idle. Once the execute input is set to true, a get attributes message is sent to the UR20 FBC and the Add-On Instruction waits on its response. Once the response has been received, the Add-On Instruction indicates done. If no response or invalid data is received, the Add-On Instruction changes to error state.

The received parameters persist when input *i_xExecute* is set to false. They are overwritten on the next execution of the Add-On Instruction.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable Add-On Instruction for initialization
i_xExecute	BOOL	execute action
i_siModuleIndex	SINT	the position of the module (slot 1..64)

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	Add-On Instruction is activated

u-remote library for Rockwell Studio 5000 Logix Designer®

q_xBusy	BOOL	Add-On Instruction is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	Add-On Instruction is in error state

InOut

Name	Type	Comment
i_udtModuleParam	UDT	parameters to write to the module
Channel	UDT[2]	channel parameters (2x)
ConnectionType	SINT	[0: 4-wire, 1: 6-wire]
ConversionTime	SINT	[0: 800ms, 1: 400ms, 2: 240ms, 3: 160ms, 4: 80ms, 5: 20ms, 6: 10ms, 7: 5ms]
ChannelDiagnosis	SINT	[0: disabled, 1: enabled]
TareFunction	SINT	[0: disabled, 1: via DI, 2: via PLC, 3: via DI *and* PLC, 4 via DI *or* PLC]
SensorSensitivity	DINT	[500.000 .. 30.000.000]
FullScaleValue	DINT	[-2.147.483.648 .. 2.147.483.647]
Offset	DINT	[-2.147.483.648 .. 2.147.483.647]
iq_arsiParamWrite	SINT[64]	array with parameter raw data to write
iq_msgParamWrite	MSG	set attribute message
q_udtError	Struct	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
srtModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arstrParameter	STRING [0 .. 2]	additional error information

Possible Errors

Reason	Description
module slot address not valid	the i_siModuleIndex parameter is not valid, i.e. > 64
parameter write fault	the message could not be sent or produced an error
invalid parameter	one or more parameter values are invalid
watchdog for write sequence expired	the module did not respond within a specific time during a transmission

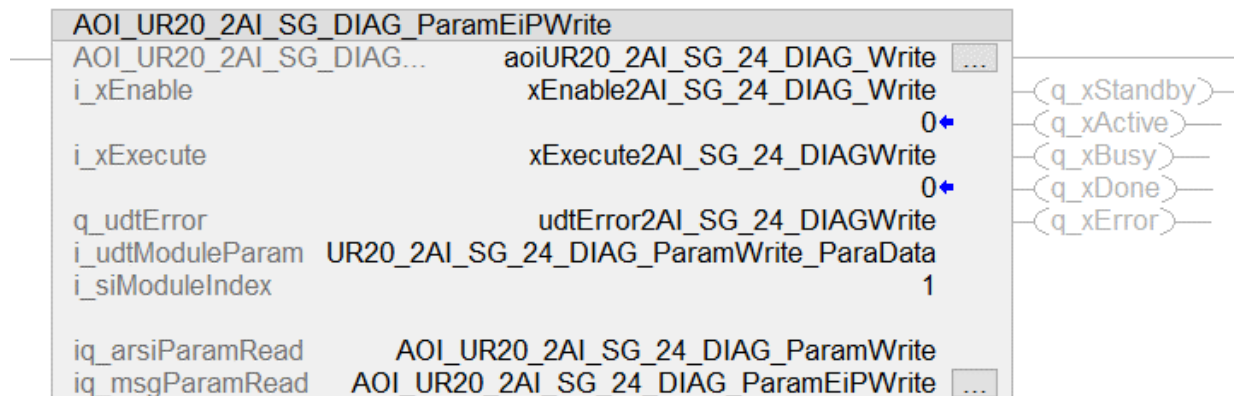
Message Configuration

The Add-On Instruction reads the module parameters using a message instruction passed to it as the parameter iq_msgParamWrite. The message instruction must be created as a controller tag and setup as follows:

Property	Value
Message Type	CIP Generic
Service Type	Get Attribute Single
<i>Class</i>	0x67
<i>Instance</i>	1
<i>Attribute</i>	0x65
Source Element	First element of the array mapped to iq_arsiParamRead
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

9.2. AOI_UR20_2AI_SG_DIAG_ParamEiPWrite



Functional description

The Add-On Instruction allows to write parameters of a UR20-2AI-SG-DIAG module. After enabling the Add-On Instruction, it is ready for its action. Before writing the module parameters, the Module Index or slot must be selected, and a valid set of parameters must be given as i_udtModuleParam. This can be done after the Add-On Instruction has been enabled or during idle. Once the execute input is set to true, a get attributes message is sent to the UR20 FBC and the Add-On Instruction waits on its response. Once the response has been received, the Add-On Instruction indicates done. If no response or an error is received, the Add-On Instruction changes to error state.

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable Add-On Instruction for initialization
i_xExecute	BOOL	execute action
i_siModuleIndex	SINT	the position of the module (slot 1..64)

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	Add-On Instruction is activated
q_xBusy	BOOL	Add-On Instruction is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	Add-On Instruction is in error state

InOut

Name	Type	Comment
i_udtModuleParam	UDT	parameters to write to the module
Channel	UDT[2]	channel parameters (2x)
ConnectionType	SINT	[0: 4-wire, 1: 6-wire]
ConversionTime	SINT	[0: 800ms, 1: 400ms, 2: 240ms, 3: 160ms, 4: 80ms, 5: 20ms, 6: 10ms, 7: 5ms]
ChannelDiagnosis	SINT	[0: disabled, 1: enabled]
TareFunction	SINT	[0: disabled, 1: via DI, 2: via PLC, 3: via DI *and* PLC, 4 via DI *or* PLC]
SensorSensitivity	DINT	[500.000 .. 30.000.000]
FullScaleValue	DINT	[-2.147.483.648 .. 2.147.483.647]
Offset	DINT	[-2.147.483.648 .. 2.147.483.647]
iq_arsiParamWrite	SINT[64]	array with parameter raw data to write
iq_msgParamWrite	MSG	set attribute message
q_udtError	Struct	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
srtModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arstrParameter	STRING [0 .. 2]	additional error information

Possible Errors

Reason	Description
module slot address not valid	the i_siModuleIndex parameter is not valid, i.e. > 64
parameter write fault	the message could not be sent or produced an error
invalid parameter	one or more parameter values are invalid
watchdog for write sequence expired	the module did not respond within a specific time during a transmission

Message Configuration

The Add-On Instruction writes the module parameters using a message instruction passed to it as the parameter iq_msgParamWrite. The message instruction must be created as a controller tag and setup as follows:

Property	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
Class	0x67
Instance	1
Attribute	0x65
Source Element	First element of the array mapped to iq_arsiParamWrite
Communication Path	Element 0 of the CIP Bus / EIP FBC in the I/O config

Properties in *cursive* are optional, as these will be overwritten by the AOI.
Refer to chp. 3.2 for step-by-step instructions on how to configure the message.

10. UR20-1COM-232-485-422

10.1. AOI_UR20_1COM_232_485_422_Control (*Deprecated*)

Functional Description

The UR20-1COM-232-485-422 control function block. The function block provides the possibility to send and receive data either on RS232, RS485 or RS422. The parametrization of the module must be done via the u-remote web interface.

The function block can be used in combination with the following u-remote modules:
UR20-1COM-232-485-422

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable function block for initialization
i_xExecute	BOOL	execute action
i_xReadWriteMode	BOOL	defines the type of operation 0 = write 1 = read
i_iTransactionSize	INT	number of bytes to be send or read

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	function block is activated
q_xBusy	BOOL	function block is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	function block is in error state
q_xReceivedNewMessage	BOOL	message from module new data present
q_xReceivedBufferNearlyFull	BOOL	message from module only 10 bytes left in buffer

InOut

Name	Type	Comment
i_udtControlParameter	UDT	contains parameter for operation
xTxBufferBehavior	BOOL	defines the behavior of the TX buffer 0 = direct sending 1 = triggered sending
xCreateModbusCRC	BOOL	activates and deactivates the local CRC check for Modbus RTU
diTransmissionWatchdogTime	DINT	time to stop ongoing transmission if no response has been received
xTransmissionWatchdogActive	BOOL	transmission watchdog 0 = inactive 1 = active
diMemoryFlushTimer	DINT	waiting time for flushing the RX and TX buffer
diMemoryFlushTimeOut	DINT	timeout flushing memory

xMemoryFlushTimerActive	BOOL	memory flush timer 0 = inactive 1 = active
iq_arsiReceiveData	SINT[1]	array receive data
iq_arsiTransmitData	SINT[1]	array transmit data
q_udtError	Struct	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
srtModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arsrtParameter	STRING [0 .. 2]	additional error information
i_arsiRxData	SINT[16]	hardware receive data
q_arsiTxData	SINT[16]	hardware send data

Possible Errors

Reason	Description
communication fault	communication error detected during operation
frame to long during read	a frame that was received by the module is too long for being read into the output array of the function block
frame to long during write	a frame that shall be send is too long for the input buffer of the module
transmission watchdog exceeded	the module did not response within a specific time during a transmission
invalid parameter	watchdog time is too short – select watchdog time > 0
watchdog for buffer flush expired	time to flush the buffer was too long

10.2. AOI_UR20_ModBusRtuMaster (Deprecated)

Functional Description

The function block allows to use the 1-COM Serial Module as a Modbus RTU master. As it uses the FB_UR20_1COM_232_485_422 function block internally, any of its faults may appear. To get some deeper information regarding this kind of fault, the user needs to refer to section 0 of this document. After enabling the function block, it is ready for its action. To send a message, the user needs to set the communication settings like slave address or function code. This can be done after the function block has been enabled or during idle. Depending on the function code, the block uses the coil array (FC 1, 2, 5, 15) or register array (FC 4, 3, 6, 16) to read data to be send or store data that was received from the slave. Once the execute input is set to true, the request to the slave node is send and the function block waits on its response. Once the response has been received, the function block indicates done. If there is no response or invalid data from the slave, the function block changes to error state. If the master requested data from the slave, the data is stored into the coil or register array. **Once input *i_xExecute* is set to false, these arrays are reset to false or zero.** To avoid data loss the array data needs to be saved or processed before. The communication settings can be changed during standby state before sending the next message.

The function block can be used in combination with the following u-remote modules:
UR20-1COM-232-485-422

Inputs

Name	Type	Comment
i_xEnable	BOOL	enable function block for initialization
i_xExecute	BOOL	execute action
i_iSlaveAdr.	INT	modbus slave address 1-247
i_siFcCode	SINT	modbus function code
i_iStartAdr	INT	UR20-1COM-232-485-422
i_iQuantityData	INT	quantity of data 1-2000 coils or 1-125 registers

Outputs

Name	Type	Comment
q_xStandby	BOOL	waiting for activation
q_xActive	BOOL	function block is activated
q_xBusy	BOOL	function block is activated and doing its supposed task
q_xDone	BOOL	execution has come to its supposed end
q_xError	BOOL	function block is in error state

InOut

Name	Type	Comment
ip_udtParameterInputs	UDT	struct contains control parameter
diReqTimeout	DINT	modbus communication timeout after sending a request
diHwMemFlushTimer	DINT	waiting period after flushing the TX/RX buffer inside the UR20-1Com Modul
diWaitAfterSend	DINT	waiting period after sending the request message
q_udtError	UDT	detailed error information
xErrorDataValid	BOOL	detailed error information is complete and ready to be read
strModule	STRING	name of the module / fb
strSource	STRING	source of the module / fb / sub-fb
strReason	STRING	description of the error
arsrtParameter	STRING [0 .. 2]	additional error information
iq_arxCoilData	BOOL [0 ..2016]	read and send buffer for bit-oriented Modbus coils and discrete inputs
iq_ariRegisterData	INT [0..125] of Int	read and send buffer for word-oriented Modbus registers
i_arsiHwProcessInputData	SINT[16]	array of process inputs from UR20-1Com Module
i_arsiHwProcessOutputData	SINT[16]	array of process outputs to UR20-1Com Module

Possible Errors

Reason	Description
invalid process value	one of the module parameters is wrong, refer to <i>q_stError.arstrParameter[1]</i> , which indicates the faulted parameter input
invalid parameter	the communication settings are wrong, refer to <i>q_stError.arstrParameter[1]</i> , which indicates the faulted parameter input
Timeout 1Com-Driver	sub-FB doesn't return an answer, please check communication settings with modbus-Slave
Invalid Response	modbus slave returns an invalid value
Slave Error Response	modbus slave returns an errorcode