

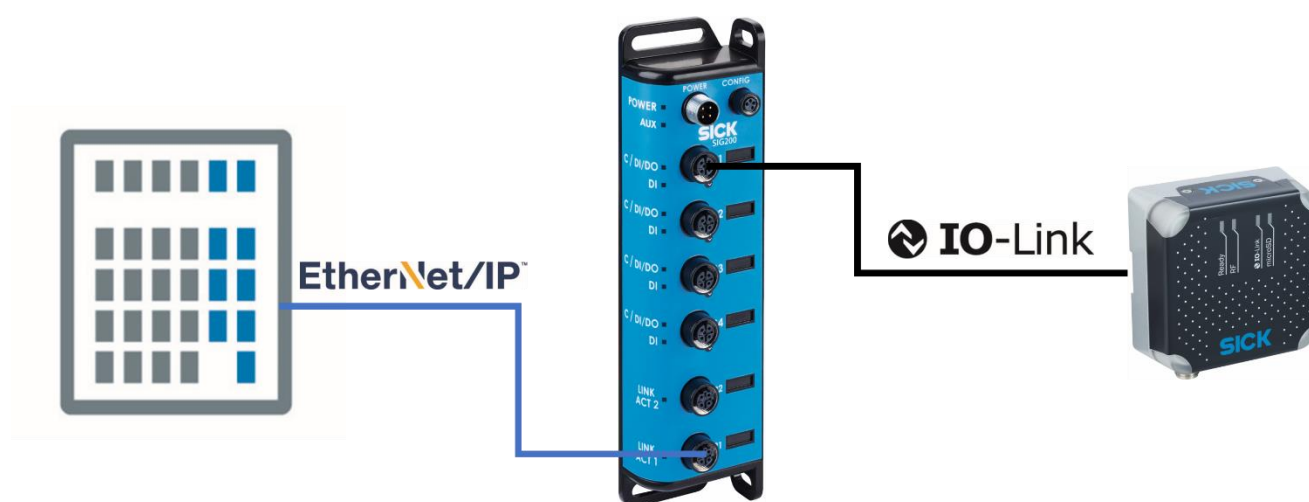
Application Note

Subject: **Connect RFU610 IOL to OMRON NX/NJ-PLC via EtherNET/IP**

SICK Product: RFU610 IO-Link; SIG200

Department: Flexible Identification

Creation Date: 2023/12/14



Version history

Version	Date	Author	Remarks
1	2023/12/14	Janis Meier	Initial Version

Table of content

1.	About this document.....	3
2.	Hardware wiring.....	4
2.1.	Needed hardware	4
2.2.	Wiring	5
3.	Needed Software	6
4.	Communication settings	7
4.1.	Overview of communication settings	7
4.2.	Set IP address of PC	7
4.3.	Set IP address of SIG200	8
4.4.	Set IP address of OMRON Controller.....	9
5.	PLC Settings.....	11
5.1.	Controller setup	11
5.2.	Network settings	18
6.	RFU610 IO-Link function block.....	28
6.1.	Integrating the function block	28
6.2.	Overview function block	29
6.3.	Operation of the function block.....	30
6.4.	Parameter of the function block.....	30

6.4.1.	Mode	30
6.4.2.	Trigger type	31
6.4.3.	WriteTag.....	31
6.4.4.	ReadTag.....	32
6.4.5.	SetTagAccessPwd.....	32
6.4.6.	SetTagKillPwd	33
6.4.7.	LockKill	33
6.4.8.	ParamAccess.....	33
6.4.9.	Inputs and Outputs of function block.....	36
7.	Use of the function block	38
7.1.	Integration in the PLC program	38
7.2.	Set parameters with function block.....	43
7.3.	Example read transponder data	45
7.4.	Example write data to transponder.....	47
8.	Overview error codes of function block	50








1. About this document

This document is a guide for commissioning an RFU610 IO-Link with a SIG200 to an NJ/NX OMRON PLC via the EtherNET/IP fieldbus. The document includes the integration of the SIG200 into the OMRON PLC and the Sysmac Studio programming environment. It also uses examples to show how transponder data can be read and written with the RFU610 IO-Link using the SICK function block.

2. Hardware wiring

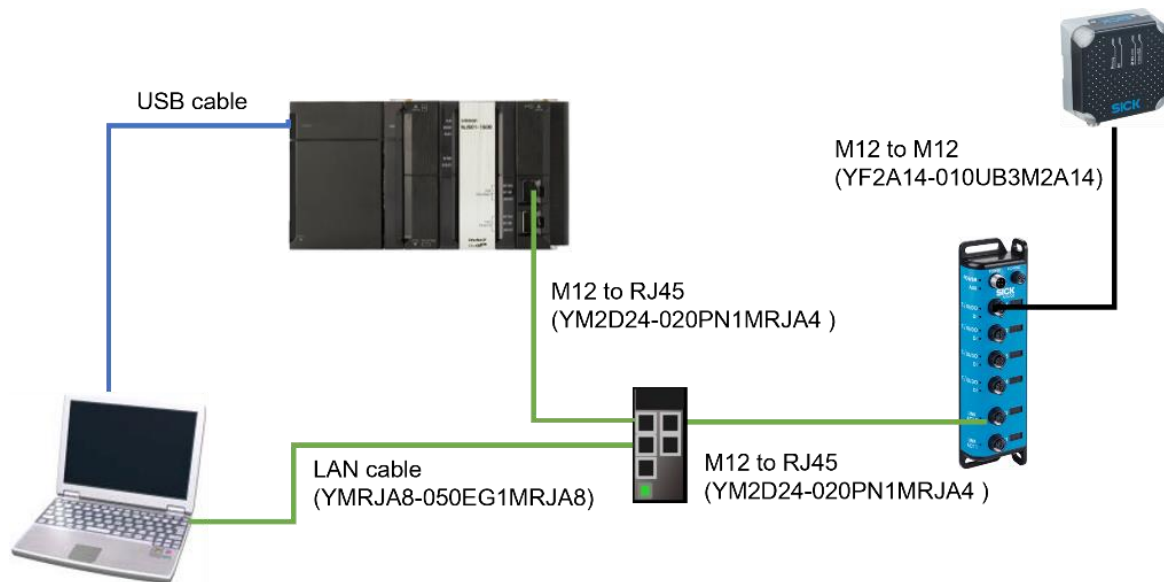
2.1. Needed hardware

List of the required hardware:

Devices	Cable
<p>RFU610 IO-Link</p> 	<p>YF2A14-010UB3M2A14 (2095997)</p> 
<p>SIG200-0A0512200 (1089796) + power cable (6075718) - optional</p> 	<p>YM2D24-020PN1MRJA4 (2106182)</p> 
<p>NJ/NX OMRON CPU-Unit</p> 	<p>YMRJA8-050EG1MRJA8 (2106264)</p> 
	<p>USB cable (USB 2.0 type B connector)</p> 

2.2. Wiring

The hardware components must be wired as follows:



3. Needed Software

The following software is required for commissioning:

1. SICK SOPAS ET - [SICK SOPAS ET Tool 64-bit](#)
2. OMRON Sysmac Studio - Programming software, available from OMRON
3. OMRON Network-Configurator - EtherNET/IP network configuration tool, available from OMRON
4. SIG 200 EDS-File - [SIG200_EDS-File](#)
5. RFU610 IOL OMRON function block - [Supportportal RFU610 IO-Link Function Blocks](#)

4. Communication settings

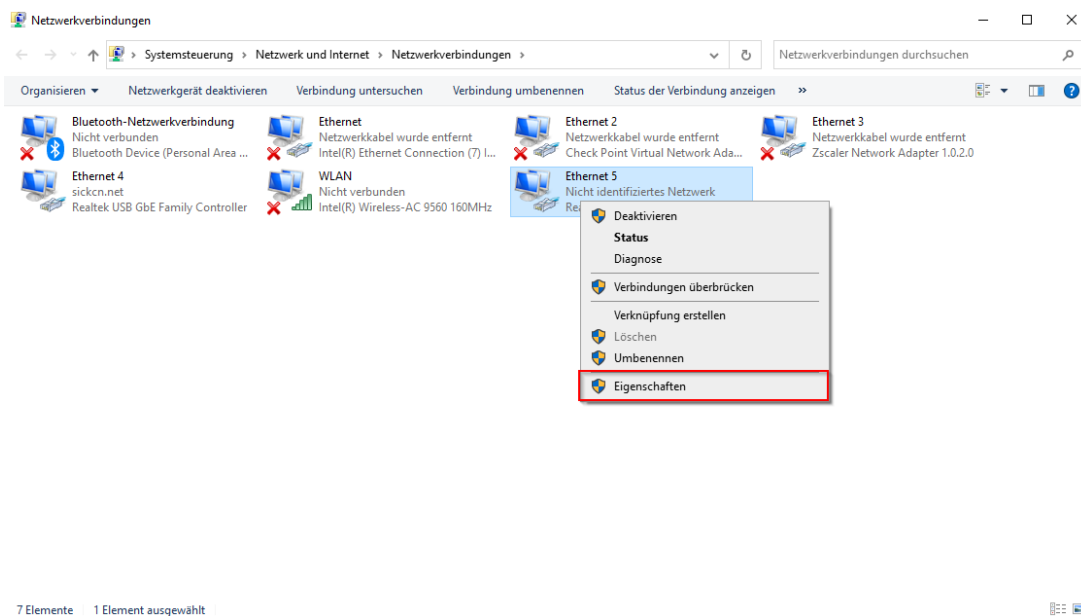
In order to establish a connection via EtherNET/IP between the IO-Link master and the OMRON CPU unit, it must be ensured that both components are in the same subnet. In this example, the subnet 192.168.250.x is used, but any other subnet can also be used.

4.1. Overview of communication settings

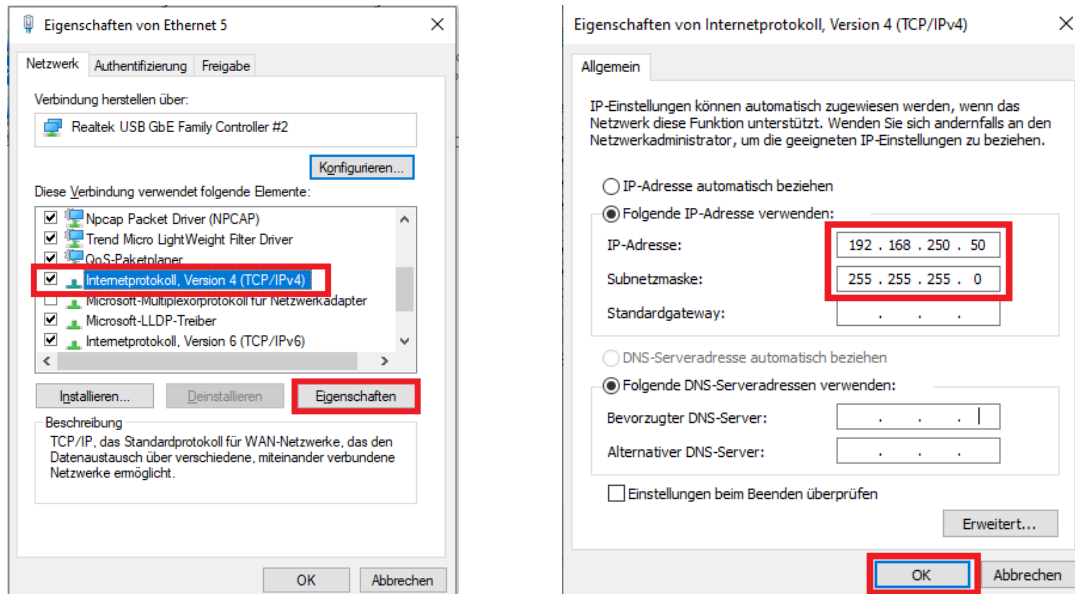
Device	Personal computer	OMRON Controller	SIG200
IP address	192.168.250.50	192.168.250.1	192.168.250.2
Subnet mask	255.255.255.0	255.255.255.0	255.255.255.0

4.2. Set IP address of PC

To change the IP address of the PC, go to Change adapter options in the Windows network settings and then open the properties of the used network.

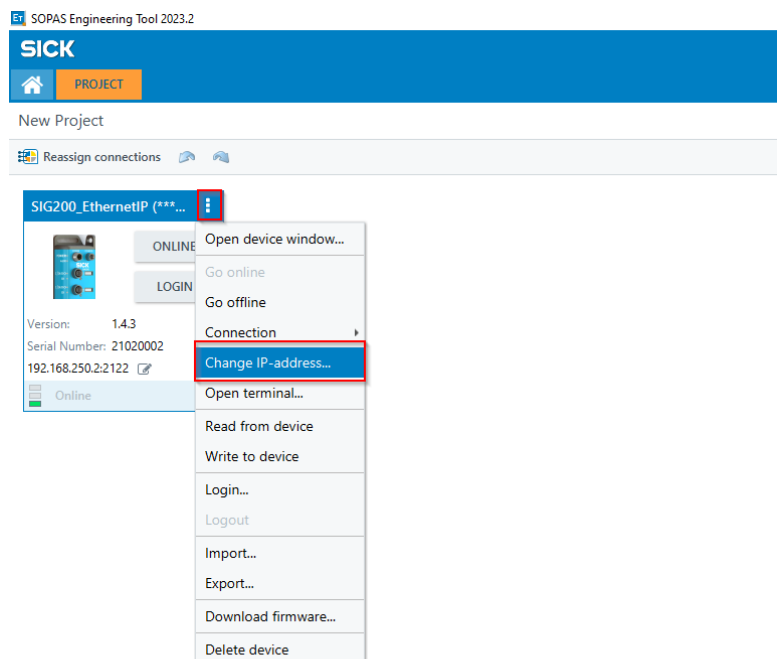


The IP address can then be changed under the properties of Internet Protocol version 4. For this example, the IP address is changed to 192.168.250.50.

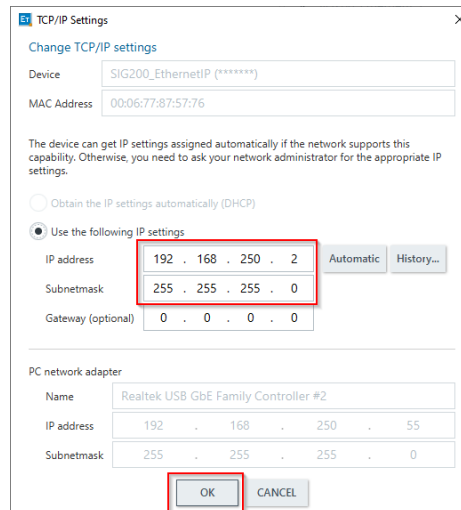


4.3. Set IP address of SIG200

The IP address of the SIG200 can be changed in SOPAS ET. To do this, the SIG200 must be searched for via SOPAS ET. Once the SIG200 has been found, the IP address can be changed using the three dots.



A window opens in which the IP address can be changed accordingly.

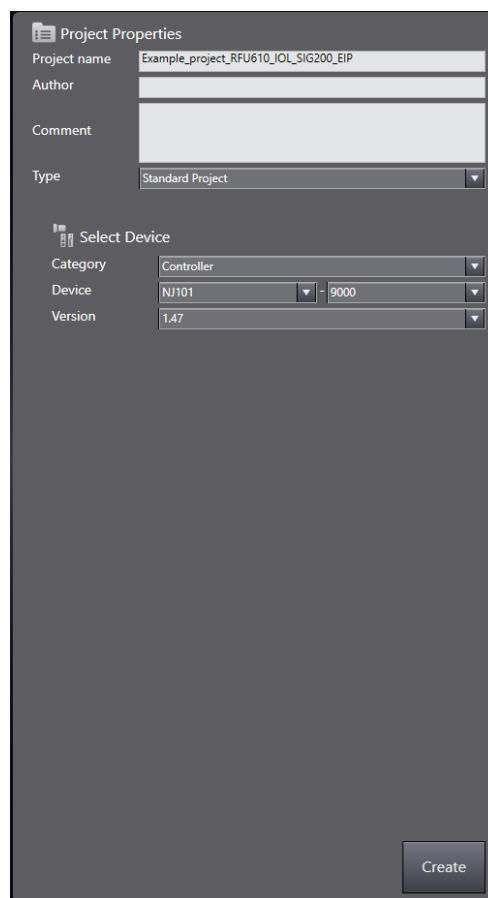


The image shows a 'TCP/IP Settings' dialog box. It has a title bar with a close button. Below the title bar is a link 'Change TCP/IP settings'. The 'Device' field contains 'SIG200_EthernetIP (*****)' and the 'MAC Address' field contains '00:06:77:87:57:76'. A note states: 'The device can get IP settings assigned automatically if the network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.' There are two radio buttons: 'Obtain the IP settings automatically (DHCP)' (unselected) and 'Use the following IP settings' (selected). The 'Use the following IP settings' section has fields for 'IP address' (192 . 168 . 250 . 2), 'Subnetmask' (255 . 255 . 255 . 0), and 'Gateway (optional)' (0 . 0 . 0 . 0). There are 'Automatic' and 'History...' buttons next to the IP address field. Below this is a 'PC network adapter' section with a table showing 'Name' (Realtek USB GbE Family Controller #2), 'IP address' (192 . 168 . 250 . 55), and 'Subnetmask' (255 . 255 . 255 . 0). At the bottom are 'OK' and 'CANCEL' buttons. A red rectangle highlights the IP address and Subnetmask fields.

PC network adapter	
Name	Realtek USB GbE Family Controller #2
IP address	192 . 168 . 250 . 55
Subnetmask	255 . 255 . 255 . 0

4.4. Set IP address of OMRON Controller

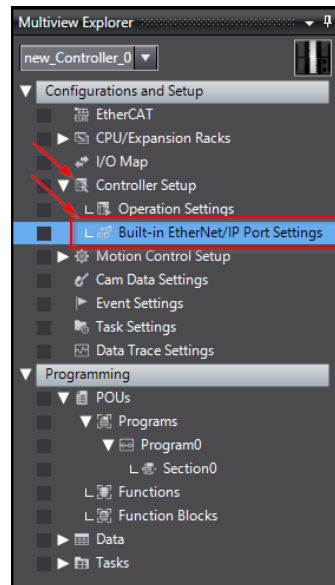
Lastly, the IP address of the OMRON controller must be set. To do this, Sysmac Studio must be started and a new project created.



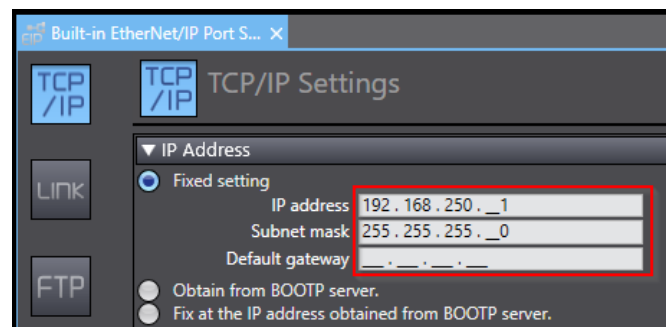
The image shows a 'Project Properties' dialog box. It has a title bar with a close button. Below the title bar is a 'Project name' field containing 'Example_project_RFU610_IOL_SIG200_EIP'. There are 'Author' and 'Comment' fields. The 'Type' dropdown is set to 'Standard Project'. Below this is a 'Select Device' section with a 'Category' dropdown set to 'Controller'. The 'Device' dropdown is set to 'NJ101' and the 'Version' dropdown is set to '1.47'. There is a 'Create' button at the bottom right.

Select Device	
Category	Controller
Device	NJ101
Version	1.47

As soon as a project has been created, the Built-in EtherNET/IP Port Settings must be opened.



Here you can enter the IP address of the EtherNET/IP port of the OMRON controller.



5. PLC Settings

5.1. Controller setup

Tags are required to transfer the cyclic process data of the SIG200 to the PLC via EtherNET/IP. For this purpose, global variables must be created in Sysmac Studio that have the appropriate size for the input and output process data of the SIG200. These global variables can then be used later as tags to ensure data exchange between the SIG200 and the OMRON PLC via EtherNET/IP.

The size of the input and output process data of the SIG200 can be found in the SIG200 operating instructions:

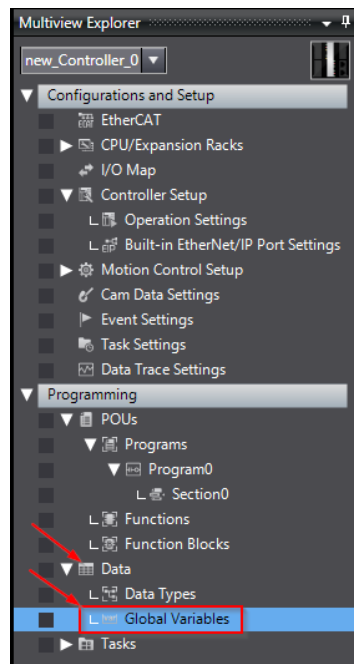
Operation via EtherNet/IP

The SIG200 can exchange process data (I/O) and (explicit) parameters via EtherNet/IP. For this purpose, the IO-Link master must be connected to a suitable programmable logic controller (PLC).

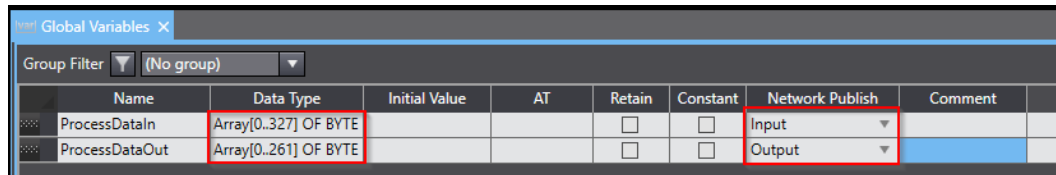
The EtherNet/IP interface of the SIG200 has the following features:

Properties	Values
Transmission rate	10 or 100 Mbit/s
Maximum distance between nodes	100 m
Process data (implicit connection)	Depending on selected assemblies Minimum cycle time: 2 ms
Max. process input data	328 byte
Max. process output data	262 byte
Asynchronous data (explicit connection)	Manufacturer-specific classes per module
Observed standard	IEEE802.3u (100Base-Tx)
Max. number of connections	8
Ethernet ports	2
CIP services	DLR, QoS
EDS file	Available at www.sick.com

The global variables can be created in sysmac studio under Programming → Data → Global Variables.

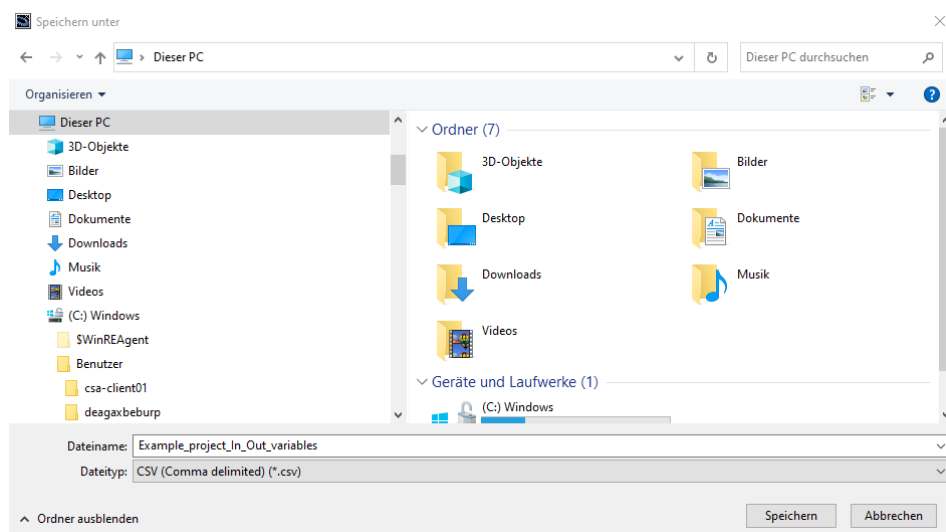
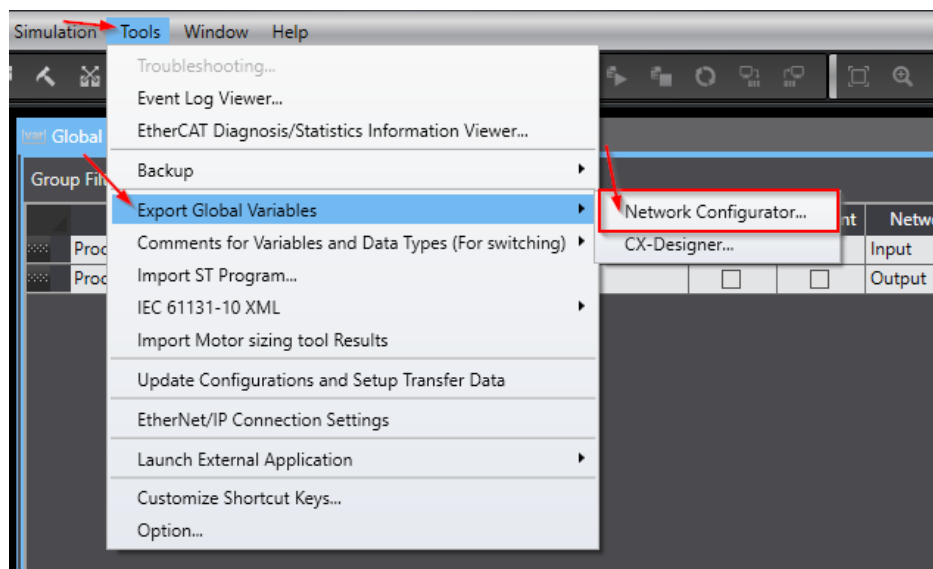


An ARRAY OF BYTES with a length of 328 bytes must be created for the input data and an ARRAY OF BYTES with a length of 262 bytes must be created for the output data. The variables can be marked accordingly as input and output data.

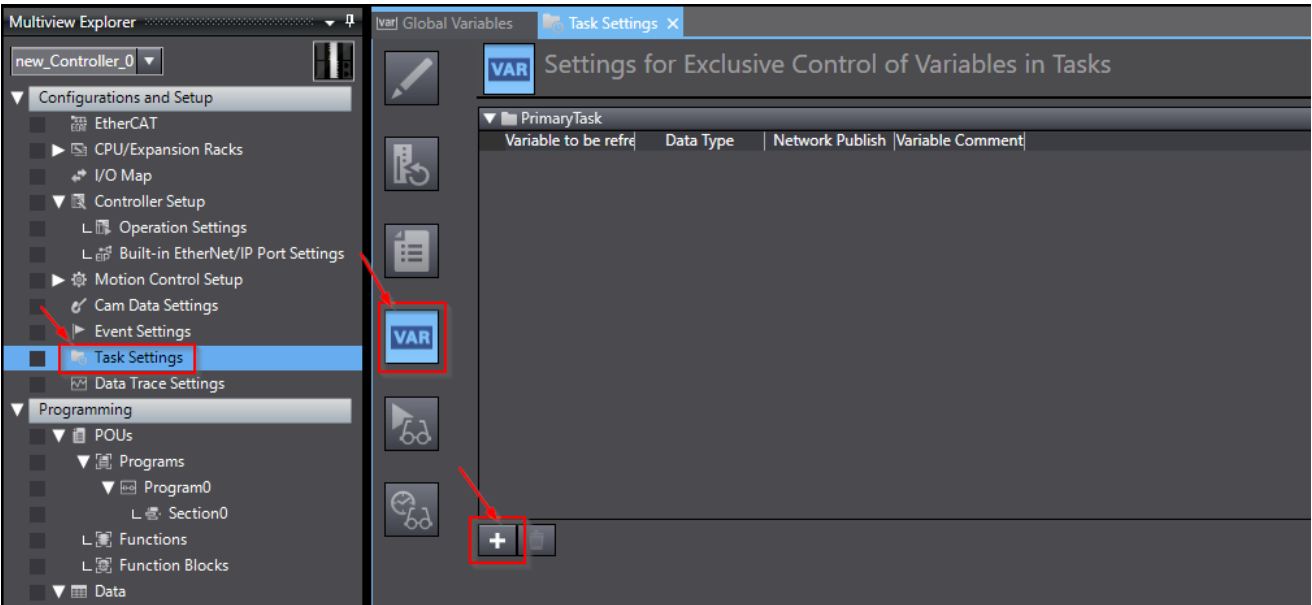


Name	Data Type	Initial Value	AT	Retain	Constant	Network Publish	Comment
ProcessDataIn	Array[0..327] OF BYTE			<input type="checkbox"/>	<input type="checkbox"/>	Input	
ProcessDataOut	Array[0..261] OF BYTE			<input type="checkbox"/>	<input type="checkbox"/>	Output	

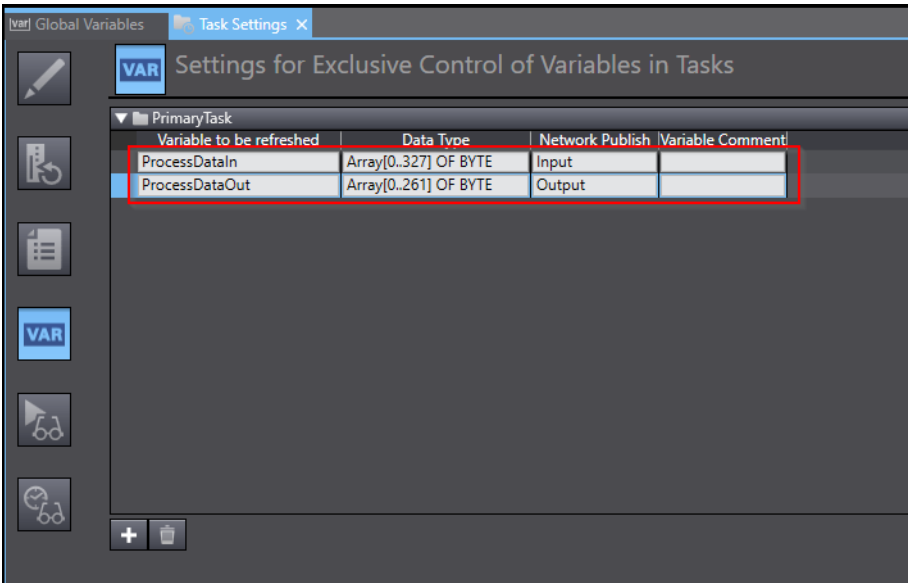
Once the global variables have been created, they can be saved as a csv file under Tools → Export Global Variables → Network Configurator.



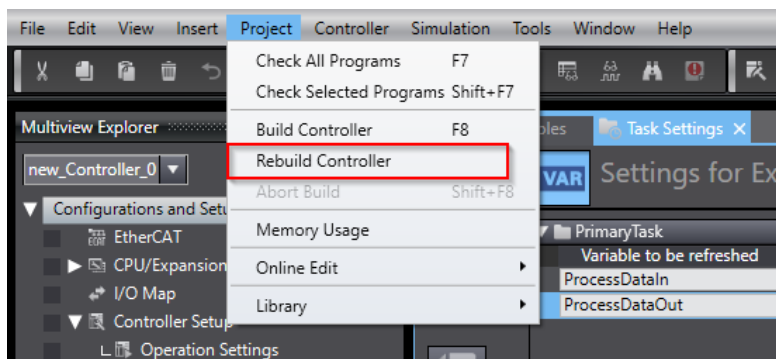
After saving the global variables, the variables must still be added under the task settings.



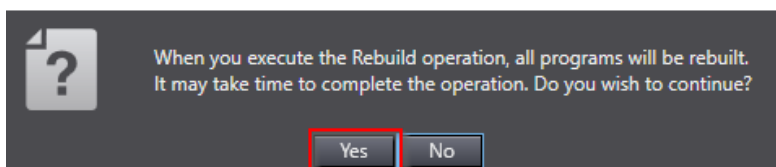
The global variables created can be selected here using the plus sign.



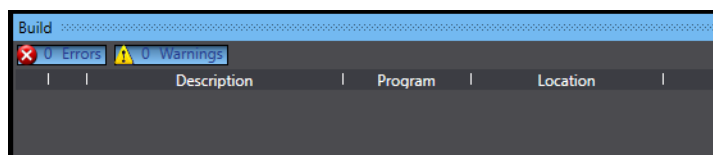
The controller can then be rebuilt to check the project for errors.



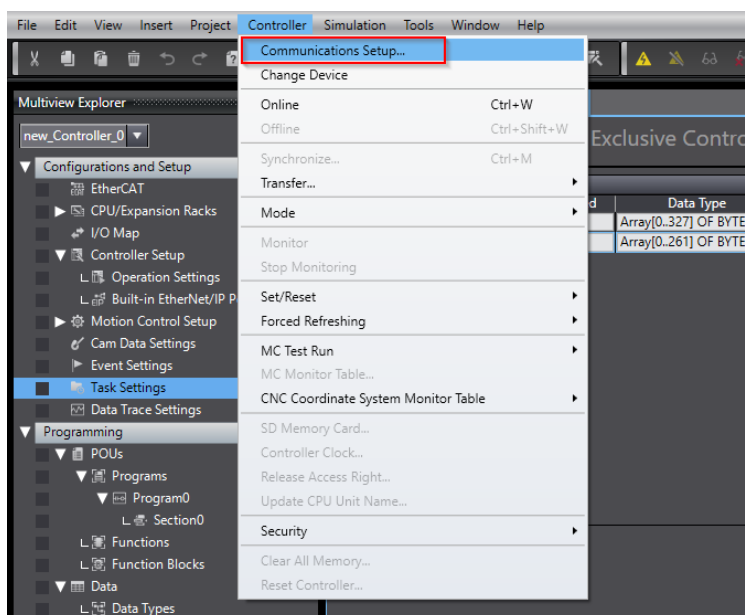
Sysmac Studio



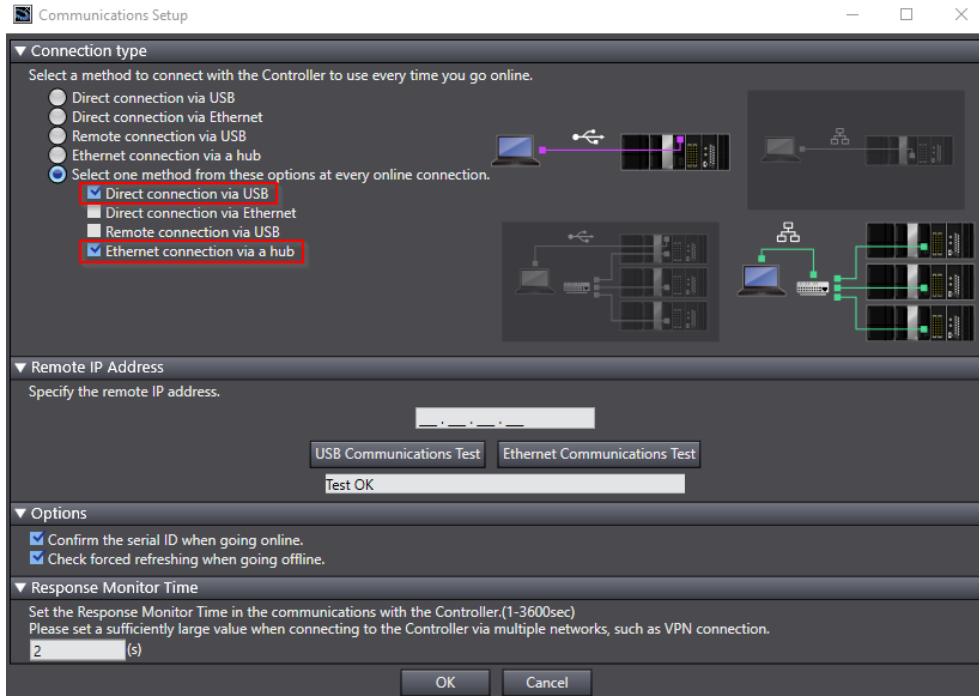
No errors or warnings should occur after building the controller.



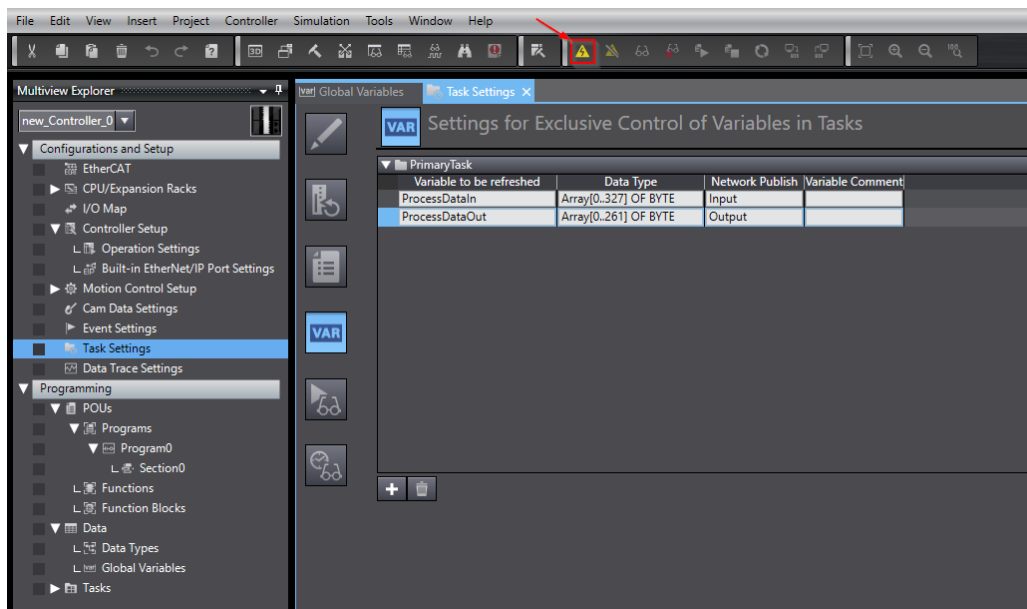
To check whether the correct communication settings are set for the connection between the PC and OMRON Controller, the Communication Setup can be opened.



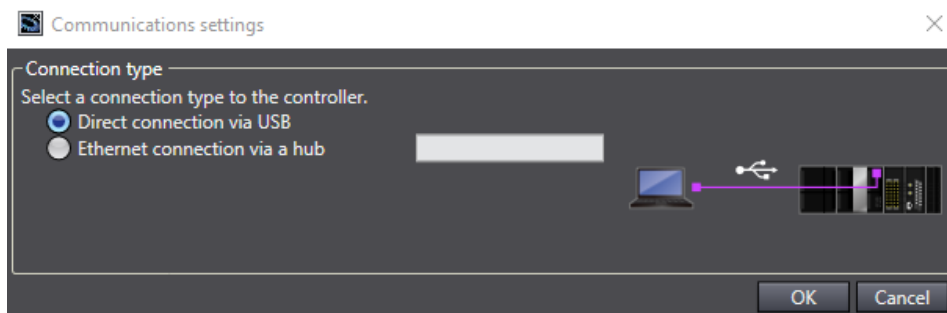
There should be a minimum connection between the PC and OMRON controller via USB or Ethernet cable. The type of connection used must be specified accordingly. In this case, a connection is possible via both USB and Ethernet.



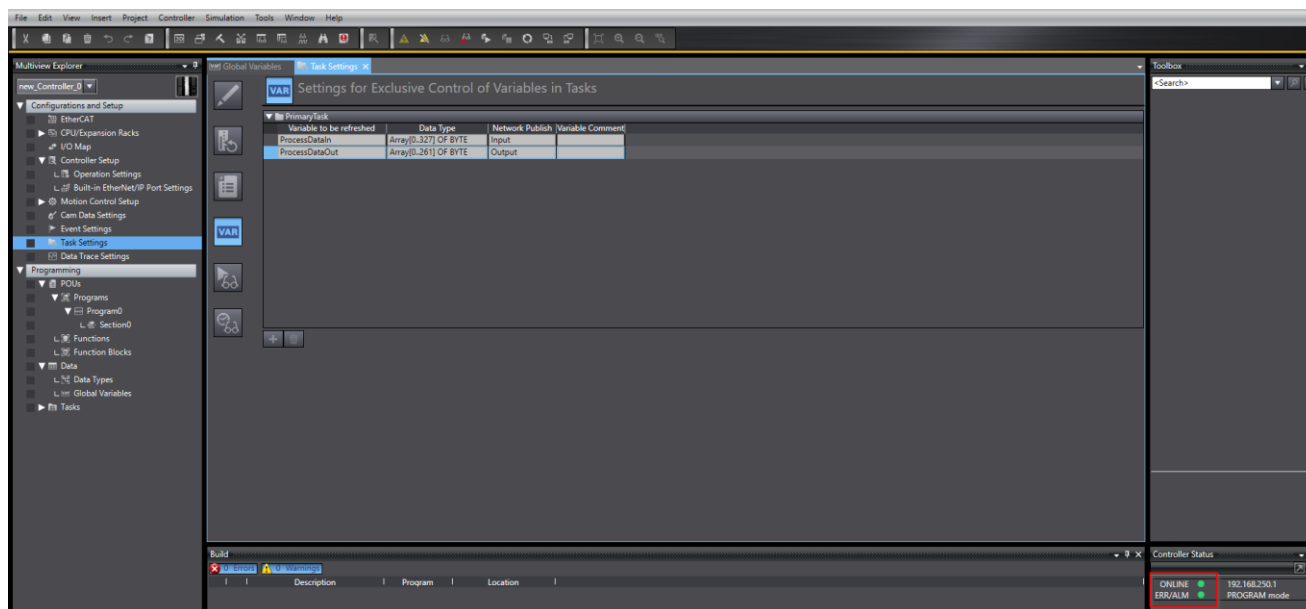
After all these steps, the OMRON controller can be switched online to establish a connection.



If both USB and Ethernet have been selected as connection options in the communication settings, you must decide which interface is to be used for the connection.



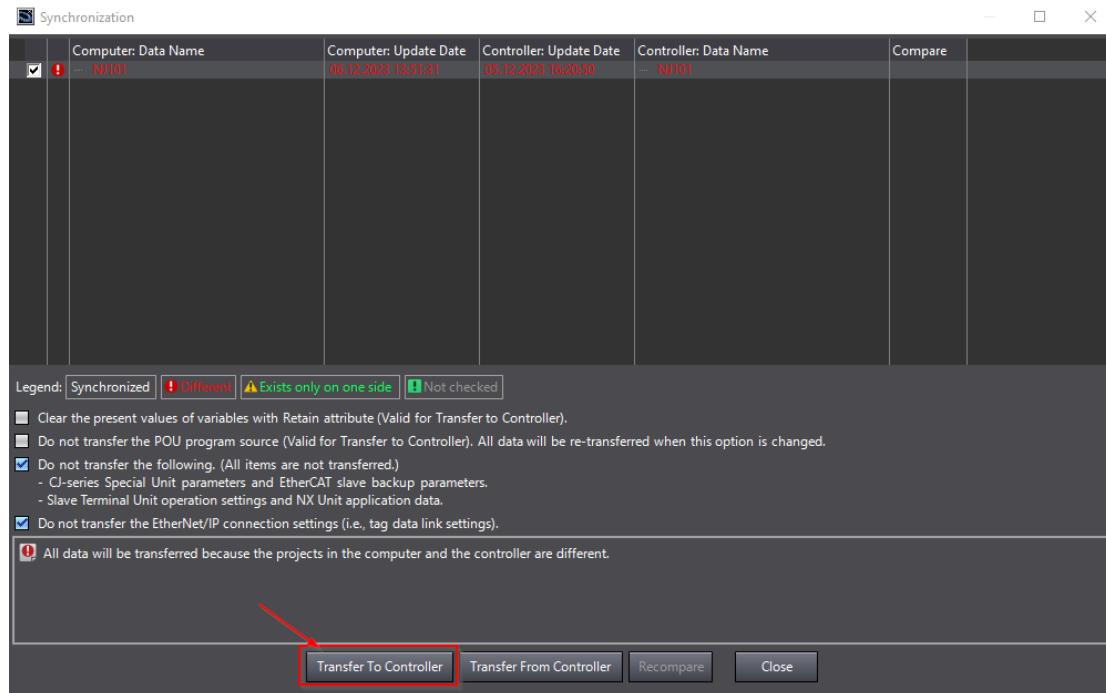
As soon as the OMRON controller is online, a yellow line is visible. The controller status should be green.



If the OMRON controller is online, the controller can be synchronized with the sysmac studio project in order to transfer the set configurations to the controller.



The transfer process is started via the Transfer to Controller button.

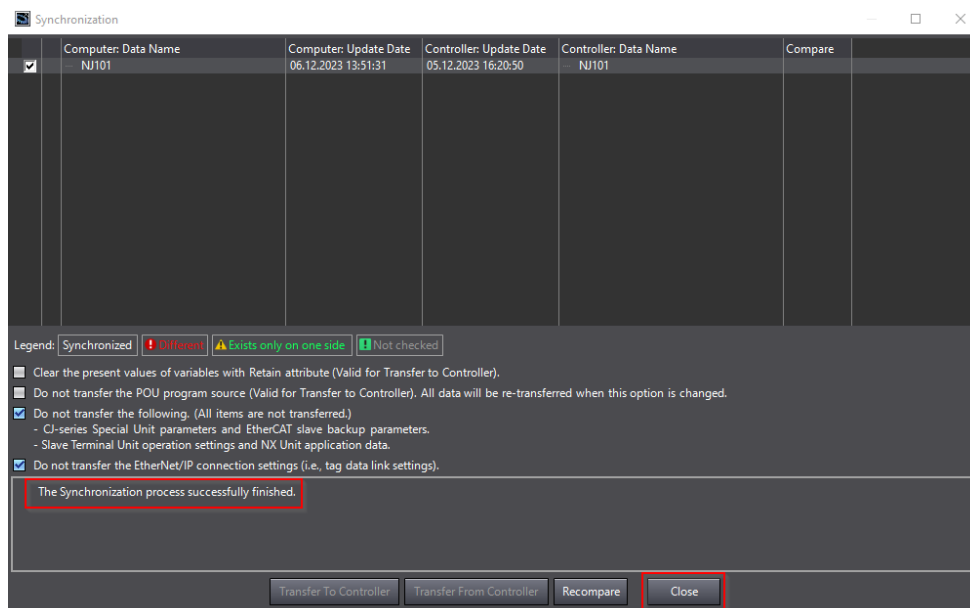


Sysmac Studio

If the Transfer to Controller operation is executed, EtherCAT slaves will be reset and forced refreshing will be cancelled.
Are you sure that you want to execute the transfer?(Y/N)

Yes No

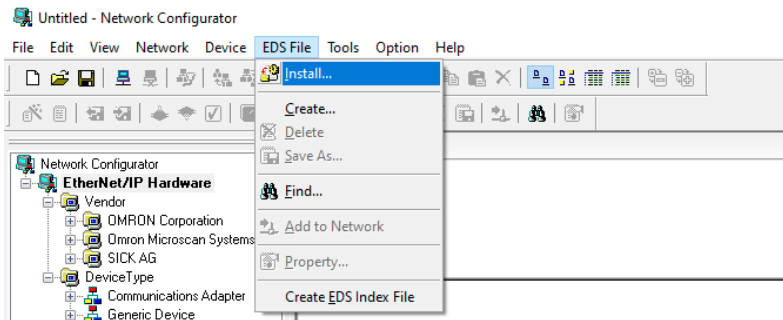
If the synchronization was successful, the window can be closed.



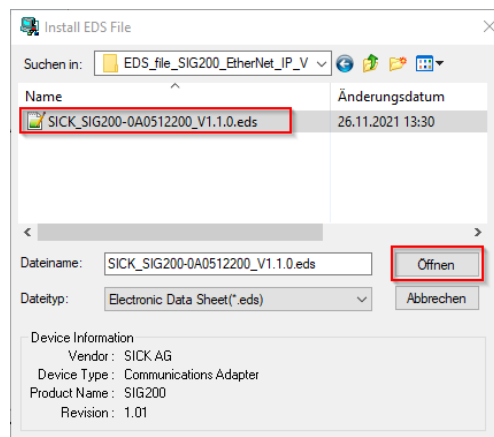
5.2. Network settings

To be able to set the EtherNET/IP tag data links, the Network Configurator must be started as an administrator. The Network Configurator is a tool that is included with the installation of Sysmac Studio.

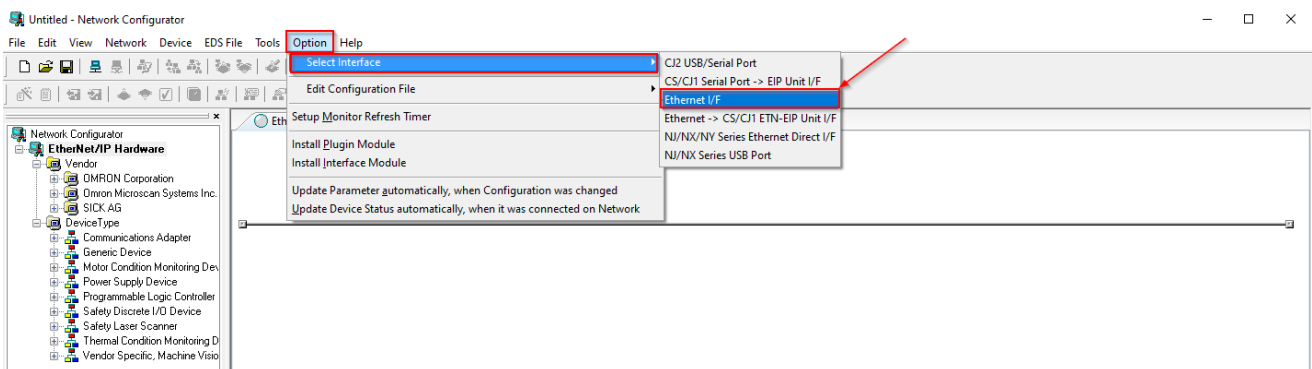
The first step is to integrate the EDS file of the SIG200 by clicking on install under the EDS file tab.



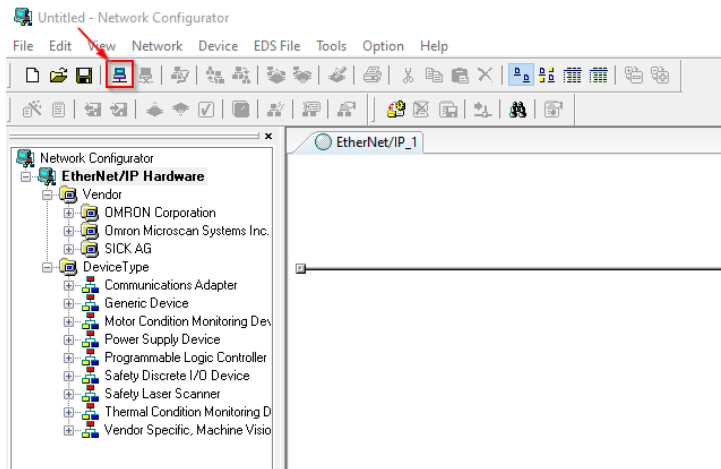
The corresponding EDS file from the SIG200 can then be selected.



The next step is to select the correct interface to go online with the network. To do this, Ethernet I/F must be selected as the interface.



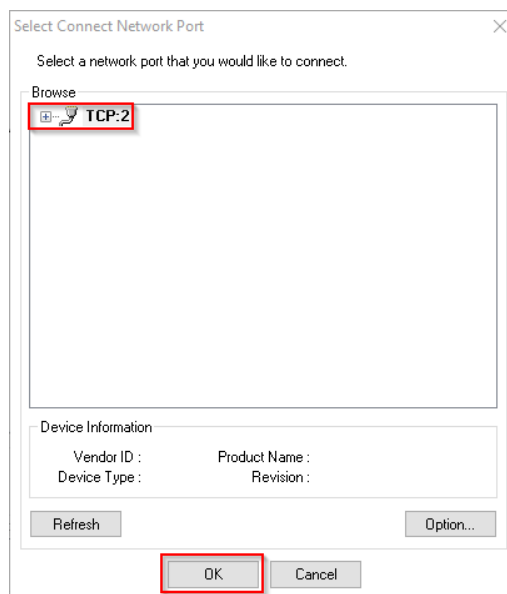
The network can then be switched online.



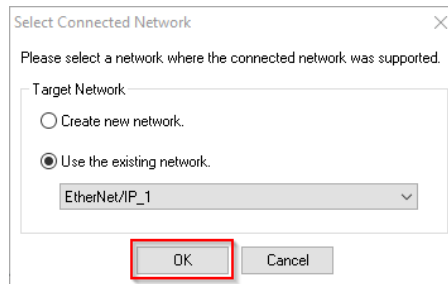
The appropriate adapter must be selected. In this example, the adapter with the IP address 192.168.250.55 is required.



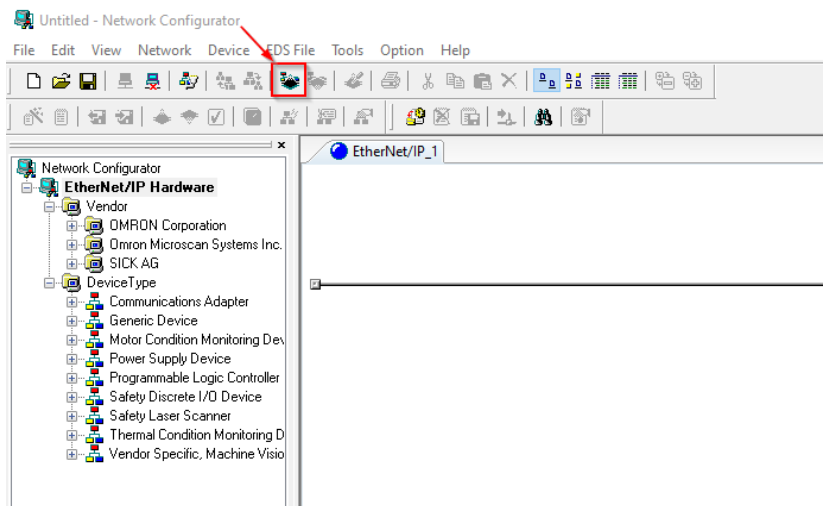
TCP:2 must be selected as the Connect Network Port.



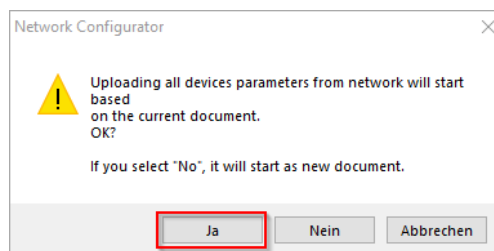
Next, the network must be selected; an existing network can be selected here if available.



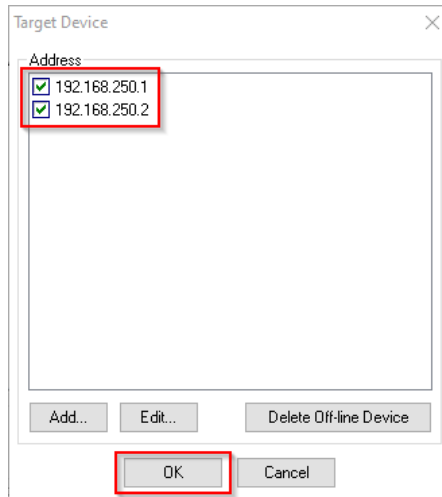
When the network is online, the color in front of the network icon should be blue. After going online, an upload from the network can be carried out to load the device parameters from the network.



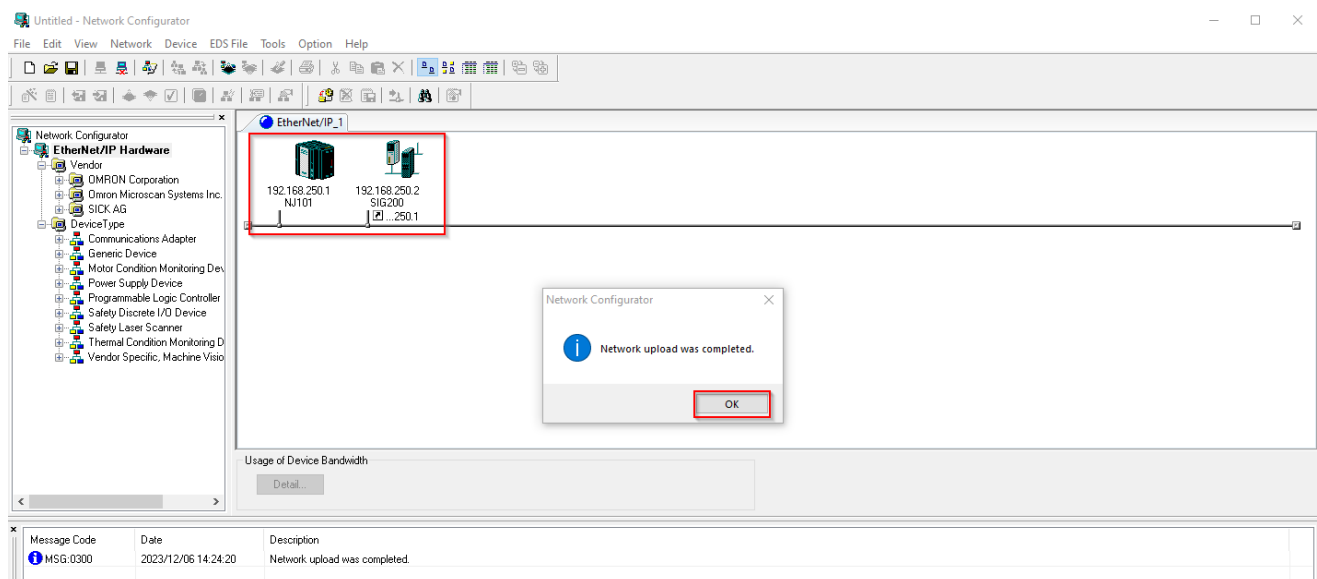
Confirm the dialog box that appears with Yes.



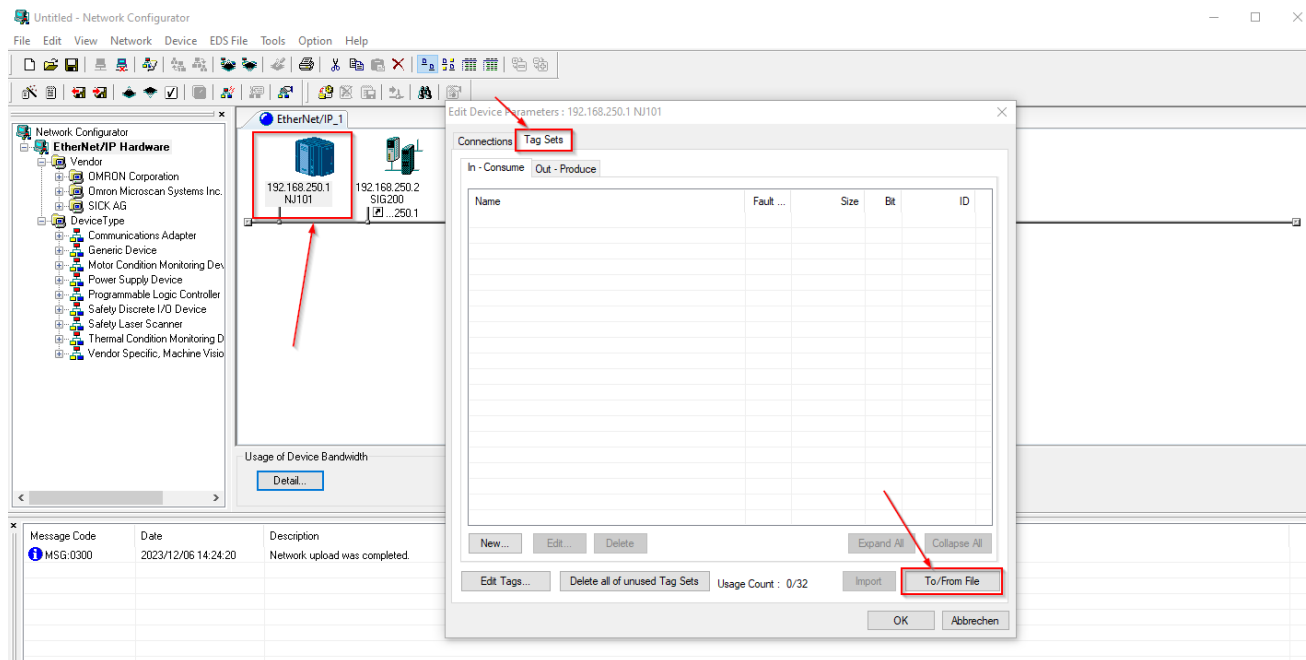
In the upload procedure, the corresponding IP addresses of the target devices must be selected. In this example, the OMRON controller (192.168.250.1) and the SIG200 (192.168.250.2). If the IP addresses are not suggested, the IP addresses can be added using the Add button.



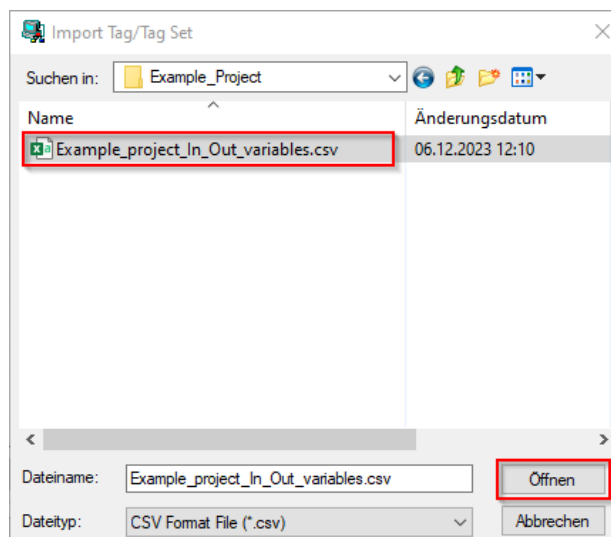
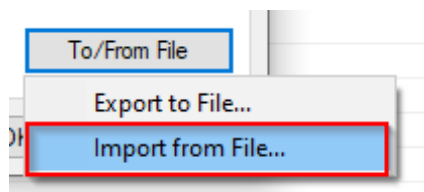
If the upload was successful, both the OMRON controller and the SIG200 should be displayed in the network.



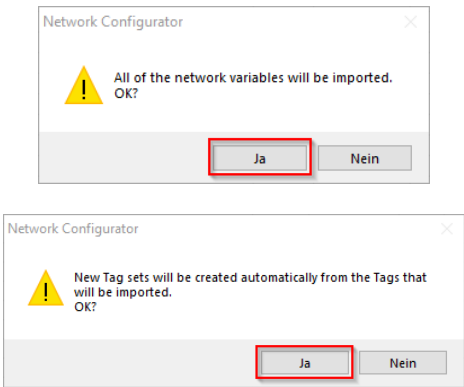
To create tags of the in Sysmac Studio generated global variables, right-click on the OMRON controller go to parameter → Edit. The device parameter window then opens, in which the global variables can be imported via the Tag Sets tab.



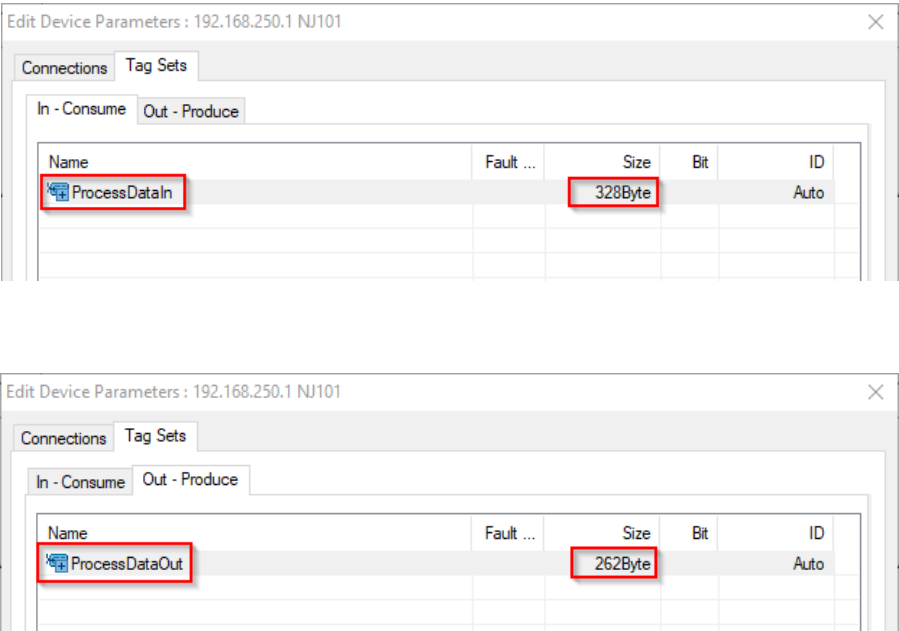
Click on Import File and then select the appropriate csv file.



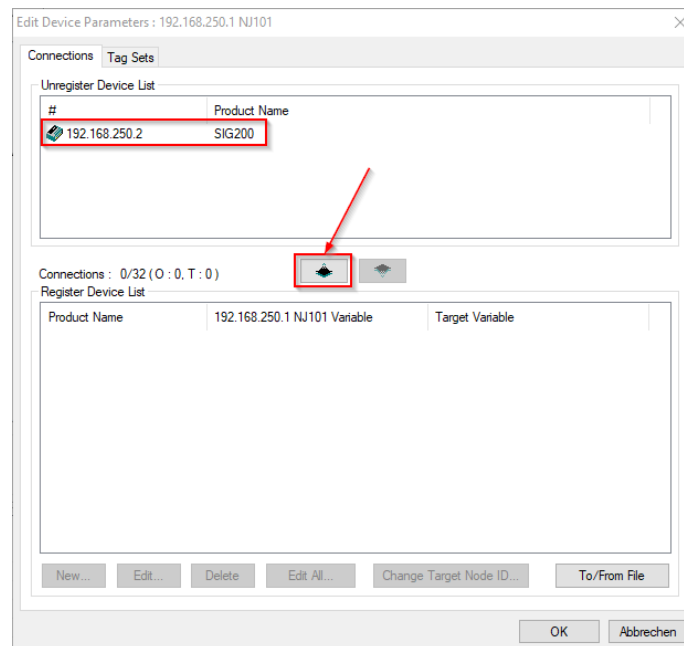
The next two dialog boxes can both be confirmed with Yes to import the created variables.



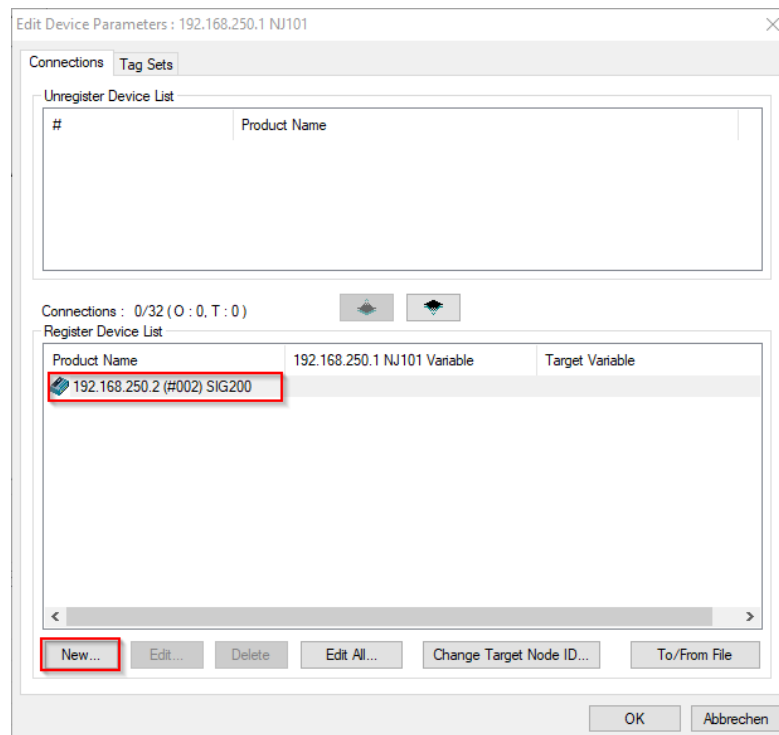
If the import was successful, the variables previously created in Sysmac Studio should be visible as tags. You should check whether the variables have the correct sizes.



Once the tags have been defined, the tags from the SIG200 can be linked to the tags from the OMRON controller. To do this, click on the SIG200 and press the download button.



The button “New” can then be used to define the connection settings for the tags.



Connection I/O Type must be set to Exclusive Owner in order to link both the input and output tags. The corresponding tags created must be selected in the Input Tag Set and Output Tag Set fields (ProcessDataIn and ProcessDataOut). A point-to-point connection must be selected as the Connection Type.

Important: Under Detail Parameter, the RPI (Requested Packet Interval) must be set to a value of less than 10 ms. **An RPI of 4 ms is recommended!**

192.168.250.2 SIG200 Edit Connection

It will add a connection configuration to originator device.
Please configure the Tag Set each of originator device and target device.

Connection I/O Type: **Exclusive Owner**

Originator Device

Node Address: 192.168.250.1

Comment: NJ101

Input Tag Set: **ProcessDataIn - [328Byte]**

Connection Type: **Point to Point connection**

Output Tag Set: **ProcessDataOut - [262Byte]**

Connection Type: **Point to Point connection**

Target Device

Node Address: 192.168.250.2

Comment: SIG200

Output Tag Set: **Input_100 - [328Byte]**

Input Tag Set: **Output_101 - [262Byte]**

Hide Detail

Detail Parameter

Packet Interval (RPI): **4** ms (1.0 - 3200.0 ms)

Timeout Value: Packet Interval (RPI) x 4

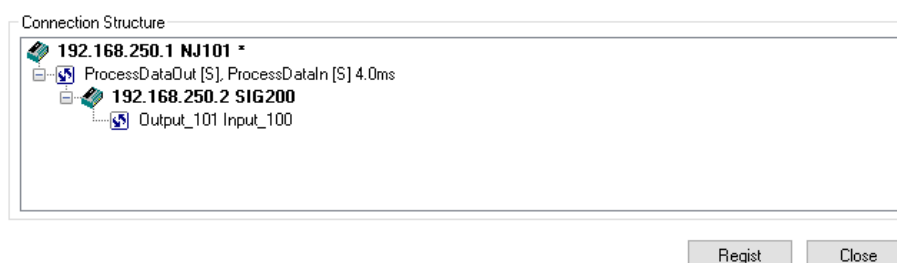
Connection Name: (Possible to omit)

Connection Structure

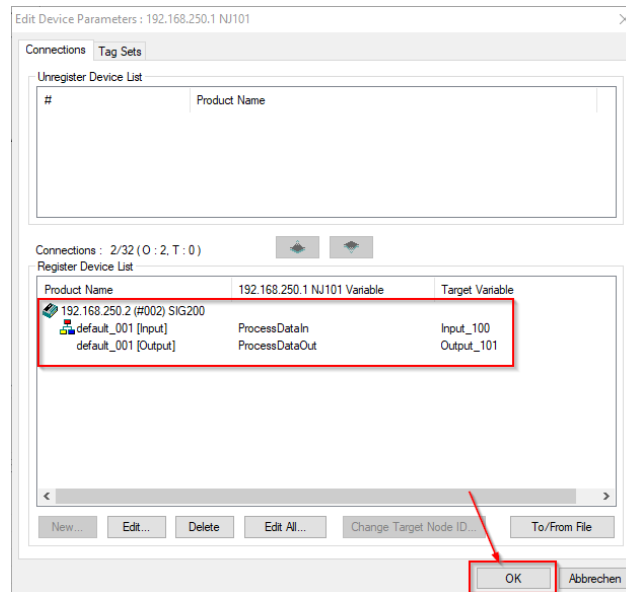
192.168.250.1 NJ101 *

Register Close

Once the values shown above have been entered, you can click on Register. The connection structure should then look like this:

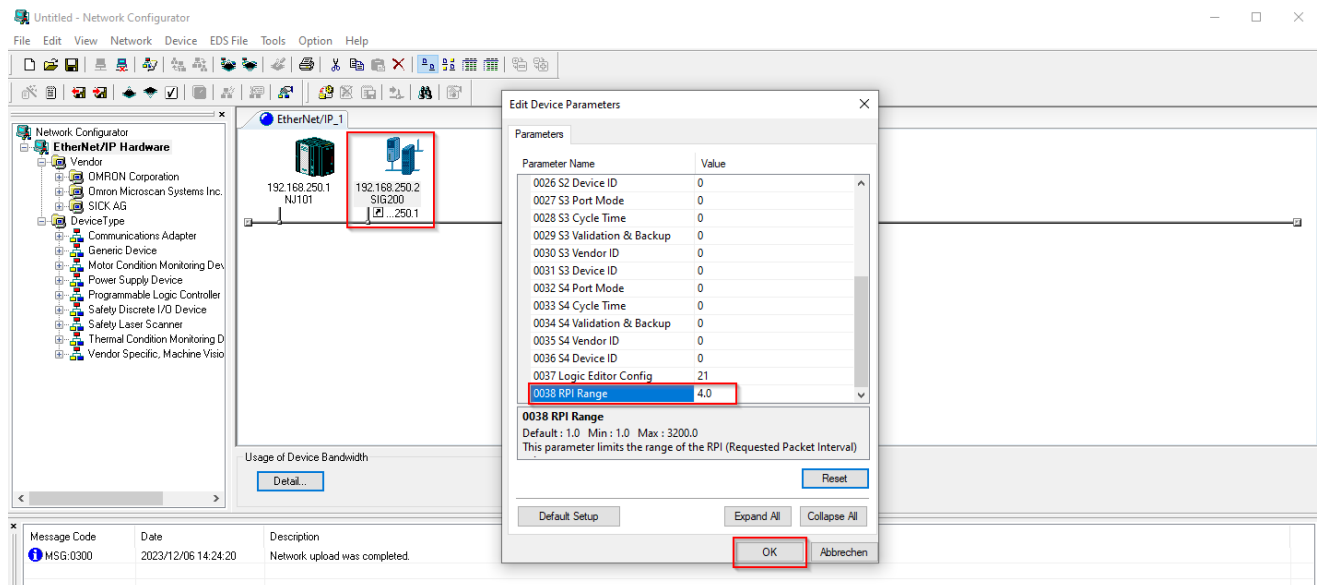


It should look like this in the Device List tab:

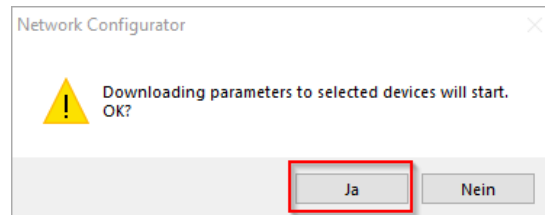
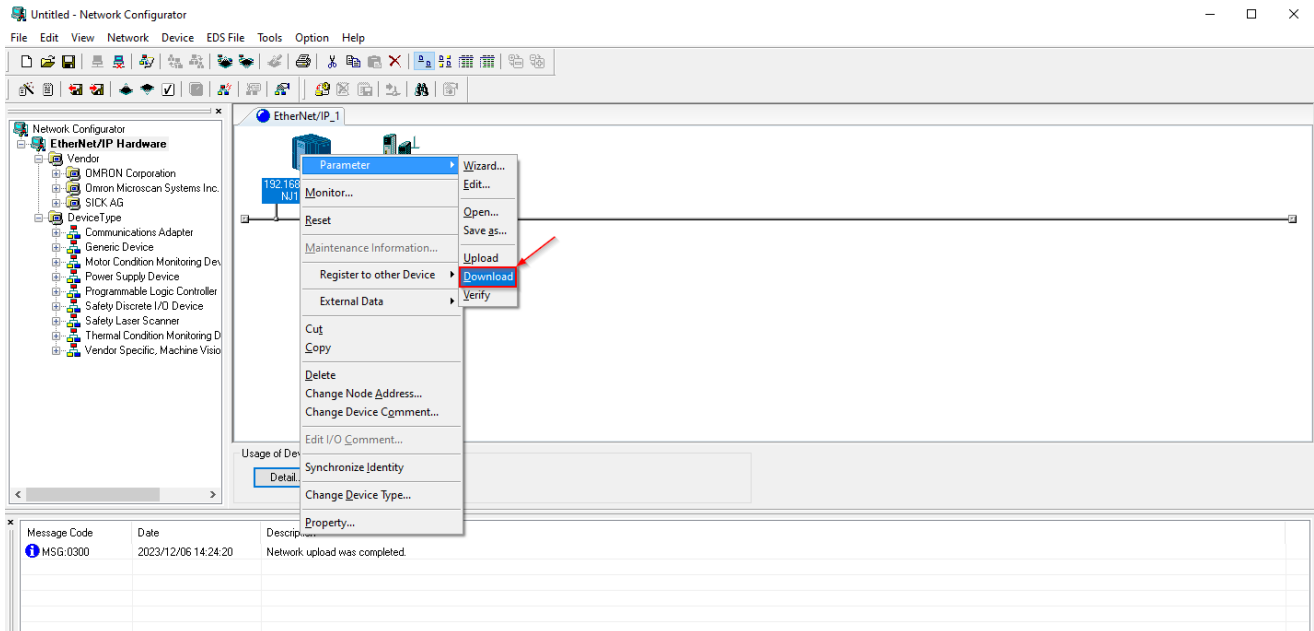


After setting the OMRON controller parameters, the RPI value for the SIG200 must be set to 4 ms. To do this, right-click on the SIG200 and go to parameters → Edit.

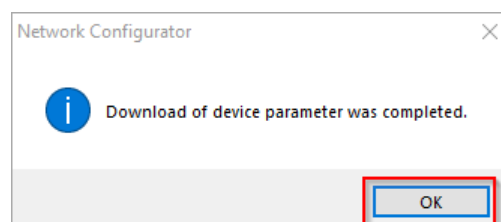
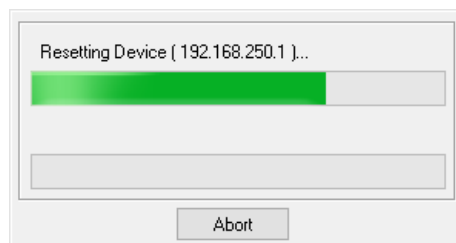
Important: The RPI (Requested Packet Interval) must be set to a value of less than 10 ms. **An RPI of 4 ms is recommended!**



Once the parameters have been set on the SIG200 and the OMRON controller, the parameters still need to be loaded onto the OMRON controller. This can be done by right-clicking on the OMRON controller and then parameter → Download.



The tag data link parameters are downloaded from the Network Configurator to the OMRON controller. The connection to the controller must not be interrupted during this process.

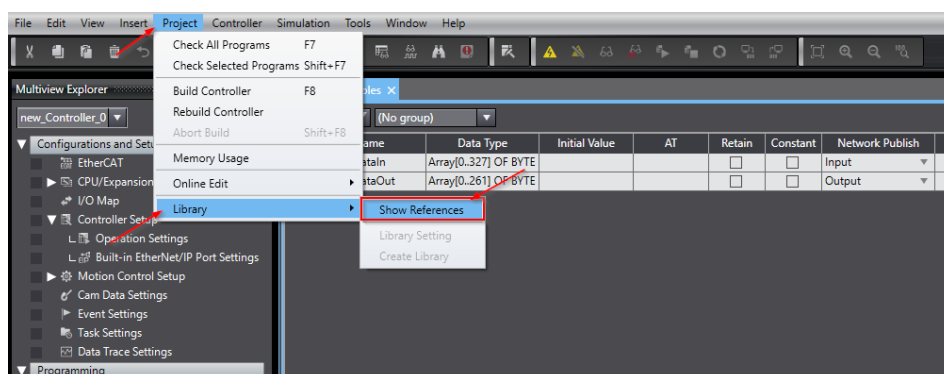


6. RFU610 IO-Link function block

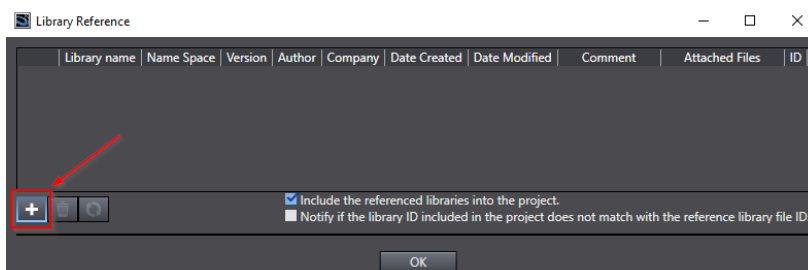
6.1. Integrating the function block

The RFU610 IO-Link function block for the OMRON PLC can be downloaded here: [Supportportal RFU610 IO-Link Function Blocks](#)

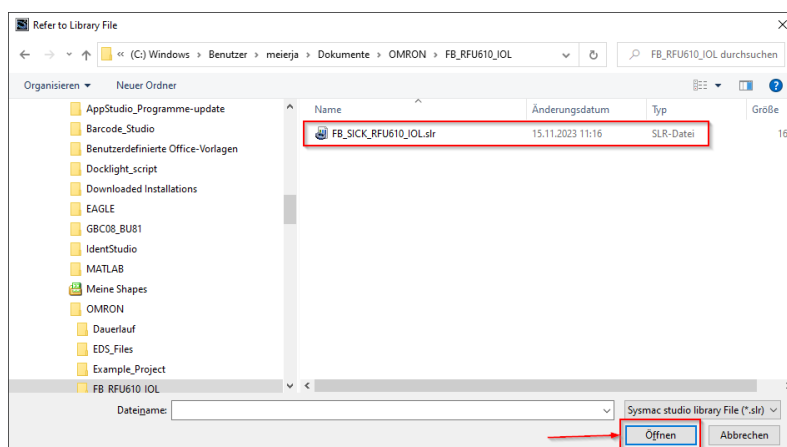
To integrate the SICK function block, open the Sysmac Studio project. Make sure that the OMRON controller is switched offline (no yellow line). Libraries can be integrated into the project via the Project tab → Library → Show References.



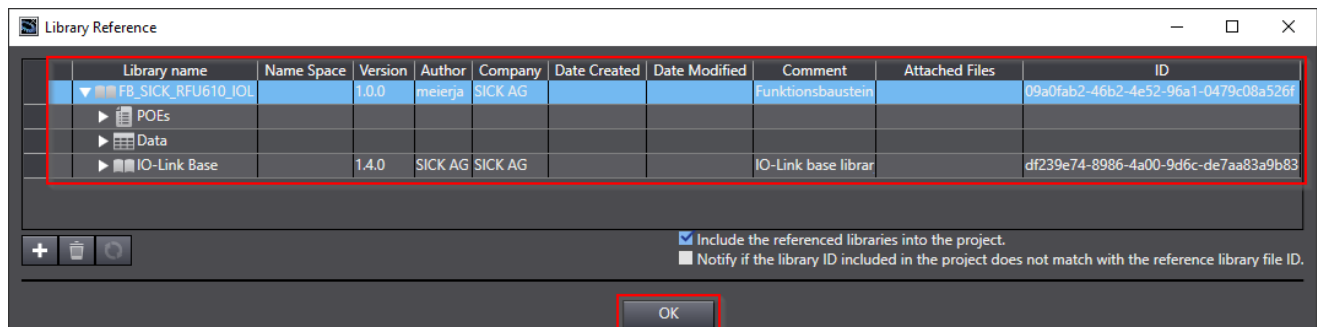
The FB_SICK_RFU610_IOL library can be integrated into the project via the plus sign.



The library file can be selected in the storage location in which you saved the library or function block after downloading.



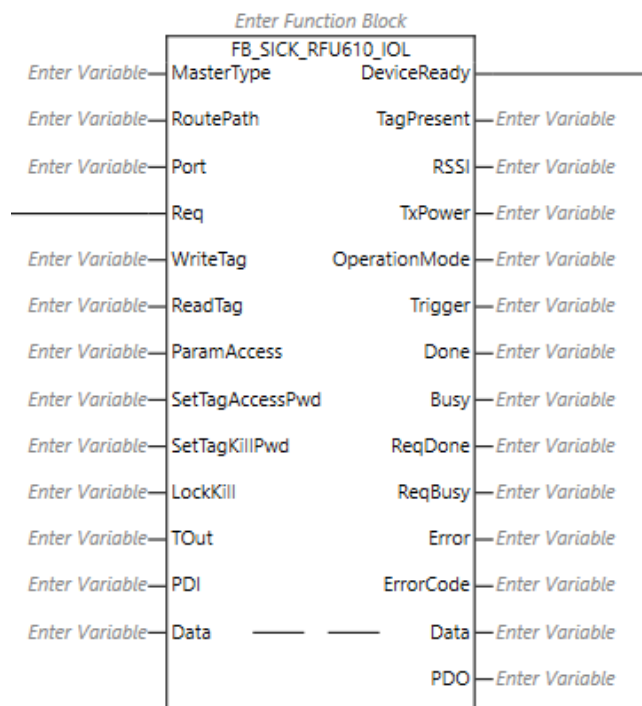
The FB_SICK_RFU610_IOL library should be visible, as well as the IO-Link Base Library in the library references.



6.2. Overview function block

The FB uses only the IO-Link process data communication channel to interact with the device. For device parameter access, the device uses the acyclic IO-Link service data communication channel. The function block works asynchronously, which means the processing requires several call ups. Therefore, it is necessary that the function block is called cyclically in the user program.

The function block with all inputs and outputs is shown here:



6.3. Operation of the function block

Each block action (#ReadTag, #WriteTag etc.) can be parameterized via the data type "ST_SICK_RFU610" (Data). To execute a function block action, the desired action must be selected first. It is not possible to select more than one action at time. To execute the selected action, the parameter #Req or an external connected digital signal must be triggered with a positive edge (signal change from a logical zero to one). The #Busy signal is set to TRUE as long as no valid device answer has been received. As soon as a falling edge is detected on the #Req input, the active command (e.g., #ReadTag) is cancelled. This behavior prevents ISDU Parameter access during an active read or write access.

If the function block signalizes #Done = TRUE at the output parameter, the action has been done successfully. If, for this action (e.g. "ReadTag") data has been requested from the device, it will be copied into the respective data area (#Data).

6.4. Parameter of the function block

The functionality of the individual inputs and outputs and the parameters of the function block are described below.

6.4.1. Mode

The RFU610 IO-Link can work in two operating modes.

Value	Mode	Description
0	ReadUII	Cyclic readout of the UII without trigger.
1	ReadWrite	Allows reading / writing of transponder contents of the different memory banks.

Please note:

If the ReadUII Mode (#OperationMode = 0) is active, the inputs #WriteTag or #ReadTag will be functionless. If the trigger shall be set manually, set the OperationMode to 1 (ReadWrite).

6.4.2. Trigger type

Depending on the trigger mode, the trigger window is opened/closed via the function block (Software-Trigger) or via a signal connected directly to the RFU (Hardware-Trigger). The trigger type can be changed via an ISDU (index 310) or via SOPAS-ET. The detected mode is shown on the function block output #TriggerType.

Value	Type	Description
0	Software-Trigger	The trigger gate is opened on a read/write request (#Req) and is closed according to the stop condition (index 312). If the stop condition is set to "Trigger input", the trigger gate is closed by resetting the #Req flag.
1	Hardware-Trigger	In this case the #Req flag has no effect. The selected function is executed as soon as the external digital signal (e.g. light switch) opens / closes a trigger window.

6.4.3. WriteTag

WriteTag function is used to write values to a defined area of a TAG.

Parameter	Declaration	Data type	Description
WriteTag. Bank	Input	USINT	Memory bank to be written. 0= Reserved 1= UII/EPC 2=TID 3=UMEM Valid range: [0..3]
WriteTag. StartWord	Input	UINT	First word (16 bit) to be written beginning with 0.
WriteTag. WordCount	Input	UINT	Number of words to be written. Valid range: [1..14]
WriteTag. Data	Input	ARRAY [0..27] OF BYTE	Data to be written to the selected tag area. The length is defined by the WordCount parameter. A maximum of 14 words (28 byte) can be written.

6.4.4. ReadTag

The ReadTag function is used to read a defined area of a TAG.

Parameter	Declaration	Data type	Description
ReadTag. Bank	Input	USINT	Memory bank to be read. 0= Reserved 1= UII/EPC 2=TID 3=UMEM Valid range: [0..3]
ReadTag. StartWord	Input	UINT	First word (16 bit) to be read beginning with 0.
ReadTag. WordCount	Input	UINT	Number of words to be read out. Valid range: [1..13]
ReadTag. DataLengthReceived	Output	USINT	Received data length in byte
Data	Output	ARRAY [0..25] OF BYTE	Data that were read out

6.4.5. SetTagAccessPwd

This function is used to set a password for TAG-Access on the TAG. Only if the password on the RFU and the TAG match, a TAG access is possible.

Parameter	Declaration	Data type	Description
TagPasswords. TagAccessPwd	Input	ARRAY [0..3] OF BYTE	Tag access password to be written to the tag. Each octet stands for one character. Example: Password = 1234ABCD [0]= 0x12 [1]= 0x34 [2]= 0xAB [3]= 0xCD

6.4.6. SetTagKillPwd

This function is used to set a password for transponder kill on the TAG. Only if the password on the RFU and the TAG match, a TAG kill is possible.

Parameter	Declaration	Data type	Description
TagPasswords. KillTagPwd	Input	ARRAY [0..3] OF BYTE	Kill password to be written to the tag. Each octet stands for a one character Example: Password = 1234ABCD [0]= 0x12 [1]= 0x34 [2]= 0xAB [3]= 0xCD

6.4.7. LockKill

This function is used to lock or kill a tag. Whether a tag will be locked or killed depends on the value in the ISDU on Index 0x1A8 (Subindex 1) "Command". If the ISDU is '0', the tag will be locked, if the ISDU is '1', the tag will be killed. If a tag is password protected, the password needs to be defined on Index 0x1A6 when locking or Index 0x1A7 when killing a tag.

6.4.8. ParamAccess

This function is used to set device parameters also known as ISDUs (Indexed **S**ervice **D**ata **U**nits). There are several parameters, that may be written to. The parameters which shall be written to the device, must be selected via the #Data.ParamAccess.Selection structure. The value of the selected parameters must be defined via the #Data.ParamAccess.Values structure. There may be more than one parameter selected per command request, as selected parameters are queued and processed in serial. The #Data.ParamAccess.RW bit is used to decide whether a value should be read or written to.

Parameter (ISDU)	Data type	Description
Mode	UINT	Operation Mode Select [0..1] 0: ReadUII 1: ReadWrite Index: 0x78
ReadPower	UDINT	Read Power [0..140] e-1 dBm Index: 0x190
RSSIFilter	Struct	RSSI Filter

			Index: 0x19A
→	"On/Off"	Bool	Absolute RSSI Filter – RSSI Filter On/Off Subindex: 1
→	LowerThreshold	INT	Absolute RSSI Filter – RSSI Filter Threshold [-100..0] Subindex: 2
	TriggerType	USINT	Trigger Source [0..1] 0: Software-Trigger 1: Hardware-Trigger Index: 0x136
	TriggerStop	USINT	Trigger Stop Condition [0..1] 0: Timer 1: Trigger Input Index: 0x138
	ReadingGateLength	UINT	Reading Gate Length [0..50000] ms Index: 0x13A
	TriggerDelay	UINT	Start of object trigger delay [0..10000] ms Index: 0x137
	DeviceAccessPassword	ARRAY [1..8] OF CHAR	Device access password UTF8 formatted Index: 0x1A6
	DeviceKillPassword	ARRAY [1..8] OF CHAR	Device kill password UTF8 formatted Index: 0x1A7
	LockKillSelector	Struct	Lock/Kill Selector Index: 0x1A8
→	Command	UINT	Defines the command [0..1] 0: Lock 1: Kill Subindex: 1
→	BankUserMemory	UINT	Defines the Bank User Memory [0..4] 0: Preserve 1: Unlock 2: Lock 3: Unlock permanently 4: Lock permanently

			Subindex: 2
→	BankEPC	UINT	Defines the Bank User Memory [0..4] 0: Preserve 1: Unlock 2: Lock 3: Unlock permanently 4: Lock permanently Subindex: 3
→	BankTID	UINT	Defines the Bank User Memory [0..4] 0: Preserve 1: Unlock 2: Lock 3: Unlock permanently 4: Lock permanently Subindex: 4
→	BankTagAccess	UINT	Defines the Bank User Memory [0..4] 0: Preserve 1: Unlock 2: Lock 3: Unlock permanently 4: Lock permanently Subindex: 5
→	BankTagKill	UINT	Defines the Bank User Memory [0..4] 0: Preserve 1: Unlock 2: Lock 3: Unlock permanently 4: Lock permanently Subindex: 6

6.4.9. Inputs and Outputs of function block

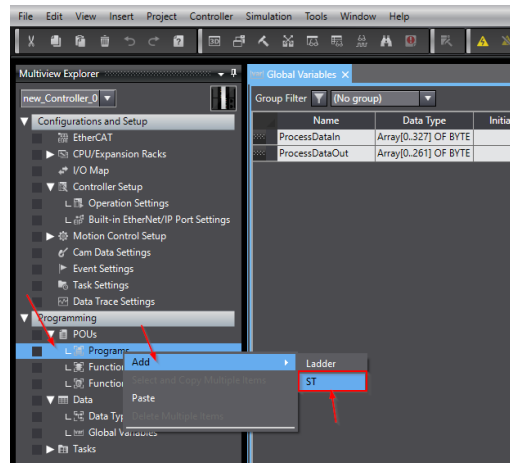
Parameter	Declaration	Data type	Description
MasterType	Input	IOLBase\E_IOL_MasterType_EIP	Type of IO-Link-Master: - eCOMTROL - eSIG200 - eSIG350
RoutePath	Input	String [64]	Route path of IO-Link Master
Port	Input	USINT	IO-Link Master port number
Req	Input	BOOL	A rising edge executes the selected actions.
WriteTag	Input	BOOL	Write tag data. Data to be written needs to be defined in the corresponding structure (#Data).
ReadTag	Input	BOOL	Read tag data. Data to be read needs to be defined in the corresponding structure (#Data).
ParamAccess	Input	BOOL	Access device Parameters. Allows read and write access to selected ISDUs. Parameter to be read or written needs to be defined in the corresponding structure (#Data).
SetTag AccessPwd	Input	BOOL	Write tag access password to the transponder.
SetTag KillPwd	Input	BOOL	Write kill tag password to the transponder.
LockKill	Input	BOOL	Locks / kills the tag area defined via an ISDU parameter.
TOut	Input	TIME	Timeout until the function block interrupts the execution
PDI	Input	ARRAY [0..31] OF BYTE	32 bytes of cyclic input process data from the RFU610 IO-Link.
Data	In/Out	ST_SICK_RFU610_Data	Contains input and output parameters for all supported function block actions.
DeviceReady	Output	BOOL	Indicates if the RFU is ready. This flag is updated cyclically.
TagPresent	Output	BOOL	Indicates if there is a tag in the RF field of the RFU610.
RSSI	Output	SINT	RSSI signal level coming from the transponder.
TxPower	Output	USINT	Used transmission power [dBm]
OperationMode	Output	UINT	Indicates the current operation mode (0 = ReadEPC, 1 = Read/Write)

TriggerType	Output	UINT	Indicates the current trigger type (0 = Software-trigger, 1 = Hardware-trigger)
Done	Output	BOOL	Indicates that the selected function block action has been performed without errors.
Busy	Output	BOOL	Request in process FALSE: Request is terminated TRUE: Request is being processed
ReqDone	Output	BOOL	Indicates that the selected acyclic request has been performed without errors. (#SetParameters)
ReqBusy	Output	BOOL	Indicates that an acyclic request is being process.
Error	Output	BOOL	An error occurred. FALSE: No error TRUE: Error detected
Errorcode	Output	ST_SICK_RFU610_Error	Error information
PDO	Output	ARRAY [0..31] OF BYTE	32 bytes of cyclic output process data to the RFU610 IO-Link.

7. Use of the function block

7.1. Integration in the PLC program

The first step is to create a program by right-clicking on Programs. You will then be asked for the desired programming language. In this example, the programming language ST (structured text) is used.



To be able to call and use the function block in the PLC program, the function block can be dragged from the right-hand side of the ToolBox into the program. Two additional lines of code must be added one before and one after the function block.

```
MemCopy(ProcessDataIn[8],ProcessDataIn_SIG200[0],32);
MemCopy(ProcessDataOut_SIG200[0],ProcessDataOut[6],32);
```

The lines of code ensure that the 328 bytes of input process data and 262 bytes of output process data from the SIG200 are written to a 32-byte array so that only the process data from the RFU610 IO-Link is used. The process data from the RFU610 IO-Link begins (when the RFU is connected to port 1) from the 8th byte for the input process data from the SIG200 and from the 6th byte for the output process data (see figures). For this reason, the input process data from byte 8 to byte 39 is copied from the global variable (ProcessDataIn / ProcessDataOut) into a local array (ProcessDataIn_SIG200 / ProcessDataOut_SIG200) and the same for the output process data from byte 6 to byte 37.

I/O assemblies with process data

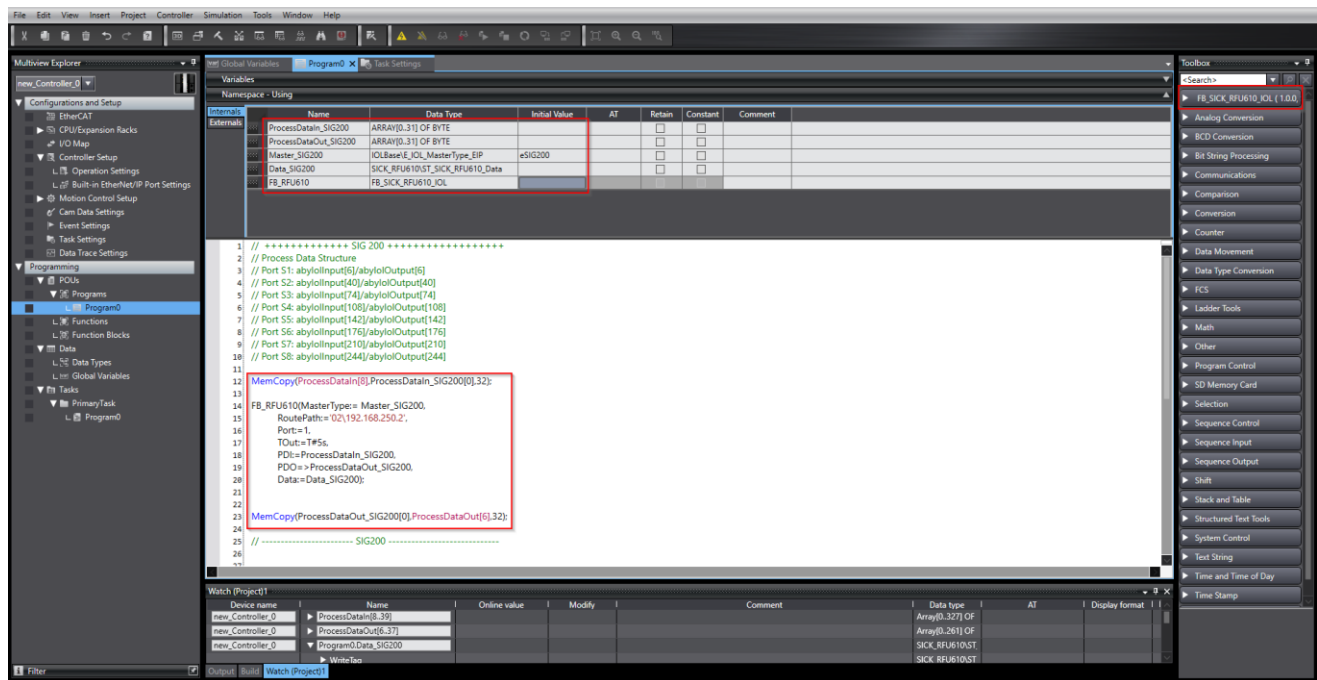
Table 19: Assembly input - instance 100

Byte	Designation		Data length	Data Type	Description
0	Inputs		1 byte	UINT8	See bit description for input data
1...7	Reserved		7 byte	ARRAY	-
8...39	Port S1	IOLink input data	32 byte	ARRAY	See documentation for connected device
40		IOLink Status	1 byte	UINT8	See bit description for IOL status
41		IOLink Error	1 byte	UINT8	See bit description for IOL error

Table 20: Assembly output - instance 101

Byte	Designation		Data length	Data Type	Description
0...5	Reserved		6 byte	ARRAY	-
6	Port S1	IO-Link output data/DO pin	1 byte	ARRAY	See documentation of the connected device/bit 0, sets the output.
7...37		IO-Link output data	31 byte	ARRAY	See documentation of the connected device
38	Port S2	IO-Link output data/DO pin	1 byte	ARRAY	See documentation of the connected device/bit 0, sets the output.

Regarding to the function block the following input parameters must then be given: **MasterType** (SIG200 must be defined as the master; a variable of the data type E_IOL_MasterType_EIP can be created for which the initial value eSIG200 is selected), **RoutePath** ("02\1" must be entered followed by the IP address of the SIG200), **Port** (The port number to which the RFU is connected), **TOut** (The timeout time of the function block), **PDI/PDO** (The array of bytes of the input and output process data of the RFU must be entered here) and **Data** (variable of data type ST_SICK_RFU610_Data which contains the input and output parameters).



The code required for this example then looks like this:

```
MemCopy(ProcessDataIn[8],ProcessDataIn_SIG200[0],32);
```

```
FB_RFU610(MasterType:= Master_SIG200,
RoutePath:='02\192.168.250.2',
Port:=1,
TOut:=T#5s,
PDI:=ProcessDataIn_SIG200,
PDO=>ProcessDataOut_SIG200,
Data:=Data_SIG200);
```

```
MemCopy(ProcessDataOut_SIG200[0],ProcessDataOut[6],32);
```

In order to be able to control the function block via the watch table in this example, the #Req input must be reset after calling the function block when the function block has finished an operation and the #Done output is set to TRUE.

```

13
14 FB_RFU610(MasterType:= Master_SIG200,
15   RoutePath:='02\192.168.250.2',
16   Port:=1,
17   TOut:=T#5s,
18   PDI:=ProcessDataIn_SIG200,
19   PDO=>ProcessDataOut_SIG200,
20   Data:=Data_SIG200);
21
22
23 MemCopy(ProcessDataOut_SIG200[0],ProcessDataOut[6],32);
24
25 // ----- SIG200 -----
26
27 // Reset Req
28 IF FB_RFU610.Done THEN
29   FB_RFU610.Req := FALSE;
30 END_IF;
31
32

```

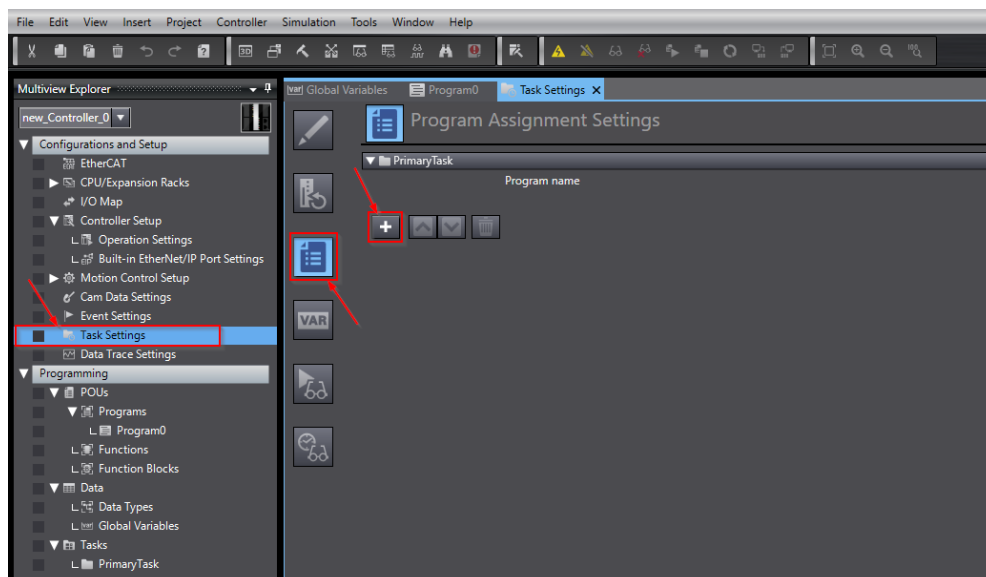
The needed code looks like this:

```

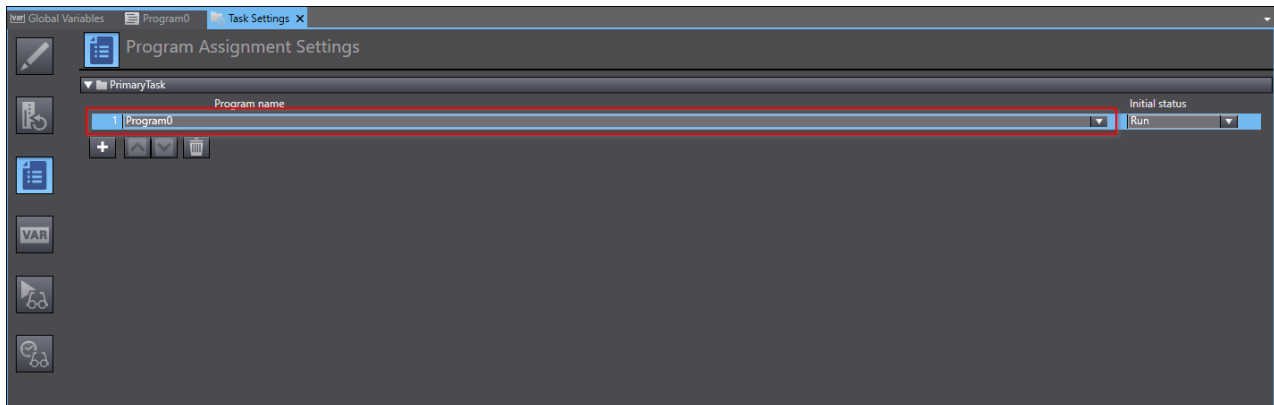
// Reset Req
IF FB_RFU610.Done THEN
    FB_RFU610.Req := FALSE;
END_IF;

```

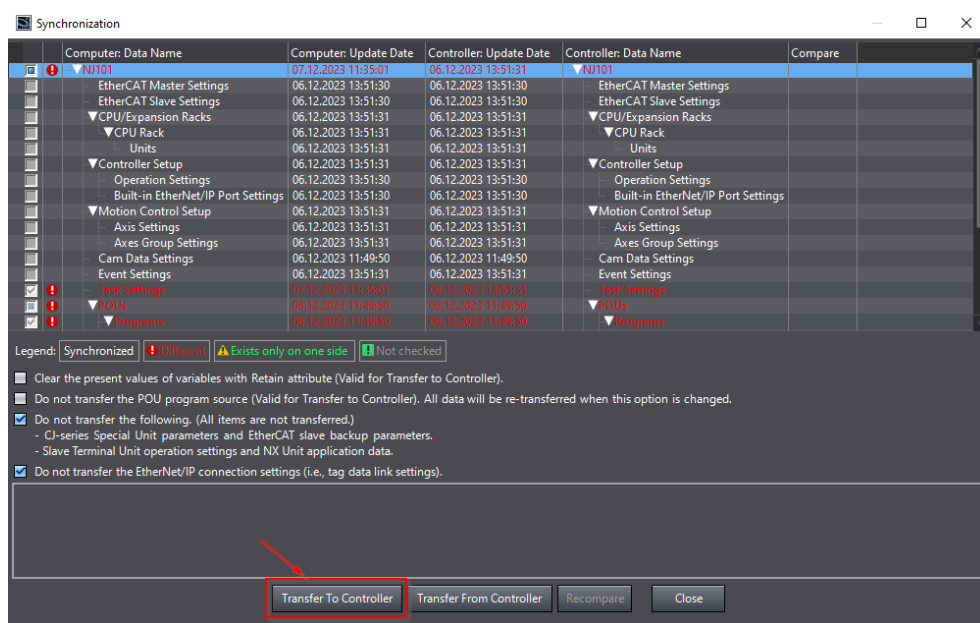
Once the corresponding lines of code and the function block have been inserted into the program, the PLC program still needs to be added to the PrimaryTask. This ensures that the program is executed immediately when the OMRON controller goes into RUN mode.



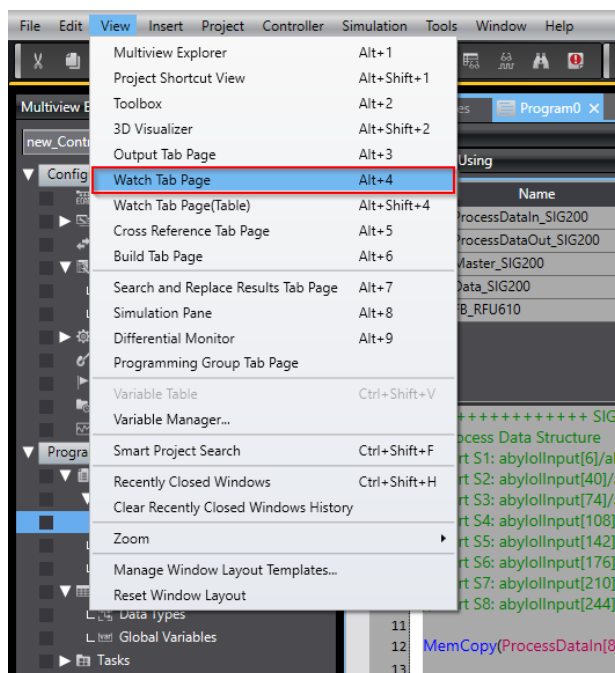
The program (in this example Program0) must be added using the plus sign.



When all of these adjustments have been made, the controller must be switched online and the changes must be transferred to the controller.



In order to be able to create a watch table to set the individual parameters of the function block, the watch tab page must be opened via the View tab.



The watch table opens, in which variables from the program can be added. The following variables are required to control the function block:

ProcessDataIn [8..39]

ProcessDataOut [6..37]

Programm0.Data_SIG200

Programm0.FBRFU610

Watch (Project)1							
Device name	Name	Online value	Modify	Comment	Data type	AT	Display format
new_Controller_0	ProcessDataIn[8..39]				Array[0..327] OF		
new_Controller_0	ProcessDataOut[6..37]				Array[0..261] OF		
new_Controller_0	Programm0.Data_SIG200				SICK_RFU610.ST		
new_Controller_0	Programm0.FBRFU610				FB_SICK_RFU610		
	Input Name...						

You can then switch to RUN mode.



To check whether data is arriving from the SIG200 to the OMRON controller, you can look in the watch table under ProcessDataIn[8..39] to see whether data is contained in the first byte (byte 8). If data is contained in ProcessDataIn[8..39], the connection between SIG200 and OMRON Controller is successfully established.

7.2. Set parameters with function block

To be able to read or write data with the RFU610 IO-Link, it must first be ensured that the RFU610 IO-Link is in the correct operating mode and that the correct trigger mode is set. In order to read or write a transponder triggered via the function block, it is necessary that the operating mode is set to ReadWrite (#Mode = 1) and the trigger mode is set to software trigger (#TriggerType = 0).

To check whether the correct operating mode and the correct trigger type are set, the value for the #Mode and #TriggerType parameters can be checked in the watch table under Program0.Data_SIG200 → ParamAccess → Values.

Device name	Name	Online value
	▼ ParamAccess	
	▼ Values	
	Mode	1
	ReadPower	140
	► RSSIFilter	
	TriggerType	0
	TriggerStop	0
	ReadingGateLength	1000
	TriggerDelay	0

If the operating mode or trigger mode needs to be changed, the values can be changed via the watch table. To do this, the #Mode and #TriggerType variables must first be set to TRUE via Program0.Data_SIG200 → ParamAccess → Selection.

Device name	Name	Online value	Modify		Comment
	► DeviceKillPassword[1-8]				
	► LockKillSelector				Lock/Kill Selector, Index 161A8, 10424
	▼ Selection				RFU Parameter Selection
	Mode	True	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	ReadPower	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	RSSIFilter	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	TriggerType	True	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	TriggerStop	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	ReadingGateLength	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	TriggerDelay	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	DeviceAccessPassword	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	DeviceKillPassword	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request

The variable #RW must then be set to TRUE under Program0.Data_SIG200 → ParamAccess to enable the writing of parameters.

Device name	Name	Online value	Modify		Comment
	ReadPower	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	RSSIFilter	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	TriggerType	True	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	TriggerStop	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	ReadingGateLength	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	TriggerDelay	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	DeviceAccessPassword	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	DeviceKillPassword	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	LockKillSelector	False	TRUE	FALSE	If true, parameter will be written to RFU on the next ParamAccess Request
	RW	True	TRUE	FALSE	RFU Read or Write Parameter, FALSE = READ, TRUE = WRITE

Once these parameters have been set, parameter writing process can be started via the function block. To do this, the #ParamAccess input must first be set to TRUE under Program0.FB_RFU610 and then the parameter access command can be started via the #Req input.

Device name	Name	Online value	Modify		Comment
	Port	1			IOL-M Port number
	Req	False	TRUE	FALSE	Starts a request on a rising edge
	WriteTag	False	TRUE	FALSE	Command Selector: WriteTag
	ReadTag	False	TRUE	FALSE	Command Selector: ReadTag
	ParamAccess	True	TRUE	FALSE	
	SetTagAccessPwd	False	TRUE	FALSE	Command Selector: SetTagAccessPwd
	SetTagKillPwd	False	TRUE	FALSE	Command Selector: SetTagKillPwd
	LockKill	False	TRUE	FALSE	Command Selector: LockKill
	TOut	5s0.000ms			Timeout until the FB interrupts the execution
	► PDI[0-31]				
	DeviceReady	True	TRUE	FALSE	Device ready
	TagPresent	True	TRUE	FALSE	Indicates if a tag is present [FALSE=not present, TRUE=present]

7.3. Example read transponder data

Reading transponder data is shown here using the following example:

- Reading 4 bytes (2 words) from the UII memory area without offset

Preconditions:

- Mode = 1 (ReadWrite)
- TriggerType = 0 (software trigger)

In order to be able to execute a read command, the ReadTag parameters must be set first.

Parameter	Declaration	Data type	Description
ReadTag. Bank	Input	USINT	Memory bank to be read. 0= Reserved 1= UII/EPC 2=TID 3=UMEM Valid range: [0..3]
ReadTag. StartWord	Input	UINT	First word (16 bit) to be read beginning with 0.
ReadTag. WordCount	Input	UINT	Number of words to be read out. Valid range: [1..13]
ReadTag. DataLengthReceived	Output	USINT	Received data length in byte
Data	Output	ARRAY [0..25] OF BYTE	Data that were read out

For the example above, the input parameters must be set as follows:

- ReadTag.Bank = 1 (UII/EPC)
- ReadTag.StartWord = 0 (No Offset)
- ReadTag.WordCount = 2 (2 words → 4 bytes)

The parameters can be set via the watch table under Program0.Data_SIG200 → ReadTag:

Device name	Name	Online value	Modify
new_Controller_0	▶ ProcessDataIn[8..39]		
new_Controller_0	▶ ProcessDataOut[6..37]		
new_Controller_0	▼ Program0.Data_SIG200		
	▶ WriteTag		
	▼ ReadTag		
	Bank	1	1
	StartWord	0	
	WordCount	2	2
	DataLengthReceived	0	
	▶ Data[0..25]		
	▶ TagPasswords		
	▶ ParamAccess		

The read command can then be executed via the function block by first setting the #ReadTag input to TRUE and then triggering the read command via the #Req input.

Device name	Name	Online value	Modify		Comment
	MasterType	eSIG200			Type of IOL-Master
	RoutePath	02\192.168.250.2			Route path of IOL Master
	Port	1			IOL-M Port number
	Req	False	TRUE	FALSE	Starts a request on a rising edge
	WriteTag	False	TRUE	FALSE	Command Selector: WriteTag
	ReadTag	True	TRUE	FALSE	Command Selector: ReadTag
	ParamAccess	False	TRUE	FALSE	
	SetTagAccessPwd	False	TRUE	FALSE	Command Selector: SetTagAccessPwd
	SetTagKillPwd	False	TRUE	FALSE	Command Selector: SetTagKillPwd
	LockKill	False	TRUE	FALSE	Command Selector: LockKill
	TOut	5s0.000ms			Timeout until the FB interrupts the execution

The reading result can then be found under Program0.FB_RFU610 → Data → ReadTag → Data:

Device name	Name	Online value	Modify	
	► PDO[0-31]			
	▼ Data			
	► WriteTag			
	▼ ReadTag			
	Bank	1		
	StartWord	0		
	WordCount	2		
	DataLengthReceived	4		
	▼ Data[0-25]			
	Data[0]	9B		
	Data[1]	6C		
	Data[2]	7D		
	Data[3]	A1		
	Data[4]	00		
	Data[5]	00		
	Data[6]	00		
	Data[7]	00		
	Data[8]	00		
	Data[9]	00		

4 Byte Ull

7.4. Example write data to transponder

Writing data to a transponder is shown here using the following example:

- Write 8 bytes (4 words: 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88) in UMEM memory area without offset

Preconditions:

- Mode = 1 (ReadWrite)
- TriggerType = 0 (software trigger)

In order to be able to write transponder data, WriteTag parameters must first be set:

Parameter	Declaration	Data type	Description
WriteTag. Bank	Input	USINT	Memory bank to be written. 0= Reserved 1= UII/EPC 2=TID 3=UMEM Valid range: [0..3]
WriteTag. StartWord	Input	UINT	First word (16 bit) to be written beginning with 0.
WriteTag. WordCount	Input	UINT	Number of words to be written. Valid range: [1..14]
WriteTag. Data	Input	ARRAY [0..27] OF BYTE	Data to be written to the selected tag area. The length is defined by the WordCount parameter. A maximum of 14 words (28 byte) can be written.

For this example the following WriteTag parameters must be set:

- WriteTag.Bank = 3 (UMEM)
- WriteTag.StartWord = 0
- WriteTag.WordCount = 4 (4 words → 8 bytes)
- WriteTag.Data = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88

The parameters can be set via the watch table under Program0.Data_SIG200 → WriteTag:

Device name	Name	Online value	Modify
new_Controller_0	▶ ProcessDataIn[8..39]		
new_Controller_0	▶ ProcessDataOut[6..37]		
new_Controller_0	▼ Program0.Data_SIG200		
	▼ WriteTag		
	Bank	3	3
	StartWord	0	
	WordCount	4	4
	▼ Data[0-27]		
	Data[0]	11	11
	Data[1]	22	22
	Data[2]	33	33
	Data[3]	44	44
	Data[4]	55	55
	Data[5]	66	66
	Data[6]	77	77
	Data[7]	88	88
	Data[8]	00	
	Data[9]	00	
	Data[10]	00	

The write command can be executed by setting the #WriteTag input to TRUE in the function block and then starting the writing using the #Req input.

Device name	Name	Online value	Modify	Comment
	P_First_Run	False	TRUE FALSE	
	MasterType	eSIG200		Type of IOL-Master
	RoutePath	02\192.168.250.2		Routepath of IOL Master
	Port	1		IOL-M Port number
	Req	False	TRUE FALSE	Starts a request on a rising edge
	WriteTag	True	TRUE FALSE	Command Selector: WriteTag
	ReadTag	False	TRUE FALSE	Command Selector: ReadTag
	ParamAccess	False	TRUE FALSE	
	SetTagAccessPwd	False	TRUE FALSE	Command Selector: SetTagAccessPwd
	SetTagKillPwd	False	TRUE FALSE	Command Selector: SetTagKillPwd
	LockKill	False	TRUE FALSE	Command Selector: LockKill

In order to check whether the data was successfully written to the transponder, the UMEM memory area can be read out. To do this, the following ReadTag parameters must be set:

- ReadTag.Bank = 3 (UMEM)
- ReadTag.StartWord = 0 (No Offset)
- ReadTag.WordCount = 4 (4 words → 8 bytes)

The WriteTag parameters can be set via the watch table under Program0.Data_SIG200 → ReadTag:

Device name	Name	Online value	Modify
new_Controller_0	▶ ProcessDataOut[6..37]		
new_Controller_0	▼ Program0.Data_SIG200		
	▶ WriteTag		
	▼ ReadTag		
	Bank	3	3
	StartWord	0	
	WordCount	4	4
	DataLengthReceived	8	
	▶ Data[0-25]		

To execute the read command, the #ReadTag input must first be set to TRUE in the function block and then reading can be started using the #Req input.

Device name	Name	Online value	Modify		Comment
	RoutePath	02\192.168.250.2			Route path of IOL Master
	Port	1			IOL-M Port number
	Req	False	TRUE	FALSE	Starts a request on a rising edge
	WriteTag	False	TRUE	FALSE	Command Selector: WriteTag
	ReadTag	True	TRUE	FALSE	Command Selector: ReadTag
	ParamAccess	False	TRUE	FALSE	
	SetTagAccessPwd	False	TRUE	FALSE	Command Selector: SetTagAccessPwd
	SetTagKillPwd	False	TRUE	FALSE	Command Selector: SetTagKillPwd
	LockKill	False	TRUE	FALSE	Command Selector: LockKill
	TOut	5s0.000ms			Timeout until the FB interrupts the execution

The reading result can be found under Program0.FB_RFU610 → Data → ReadTag → Data:

Device name	Name	Online value	Modify	
	▼ Data			
	► WriteTag			
	▼ ReadTag			
	Bank	3		
	StartWord	0		
	WordCount	4		
	DataLengthReceived	8		
	▼ Data[0-25]			
	Data[0]	11		
	Data[1]	22		
	Data[2]	33		
	Data[3]	44		
	Data[4]	55		
	Data[5]	66		
	Data[6]	77		
	Data[7]	88		
	Data[8]	00		
	Data[9]	00		

8. Overview error codes of function block

The parameter "Errorcode" contains the following error information:

- Block specific error code
- Device error code
- Extended error code

Block Errorcode	Description
16#0000	No error
16#0001	Timeout error occurs. The processing of the actions takes longer than the time set at the "TOut" parameter. The timeout is only active when "Software-Trigger" is used.
16#0002	A request was executed without selecting a parameter, or with more than one parameter selected.
16#0003	Unsupported memory bank (>3)
16#0004	Reserved
16#0005	No transponder present in the RF-Field of RFU
16#0006	Write tag: Wordcount > 14 not usable in mode 1
16#0007	Read tag: Wordcount > 13 not usable in mode 1
16#0008	Unsupported device mode (>1)
16#0009	Unsupported trigger type (>1)
16#000A	The variable, which is connected to the "DeviceDataOut" parameter, is not from the type "Array of Byte". Please assign a valid variable for the outgoing process data.
16#000B	The variable, which is connected to the "DeviceDataIn" parameter, is not from the type "Array of Byte". Please assign a valid variable for the incoming process data.
16#000C	The assigned array with the output process data (DeviceDataOut) must be at least 32 bytes long.
16#000D	The assigned array with the input process data (DeviceDataIn) must be at least 32 bytes long.
16#000E	Error when copying the output process image.
16#000F	Error when copying the input process image.

16#0010	Device error detected	
	The variable "DeviceErrorcode" contains the device error code.	
	Errorcode	Description
	0x0001	Device not ready
	0x0002	Command execution failed
	0x0003	Invalid number of requested data
	0x0004	Invalid telegram length
	0x0005	More than one tag detected
	0x0007	Received EPC is too long
	0x0008	Requested write operation failed
	0x0009	Timeout occurred during confirmed messaging protocol was processing
	0x000A	Wrong coding standard or inconsistent data
	0x000B	Lock command failed
	0x000C	Kill command failed
	0x000D	Wrong trigger input active
	0x000E	Logging active, but no SD card inserted
	0x000F	Logging active, but SD card failed
	0x0010	Odd length of offset or length
	0x0011	Command execution interrupted
	0x0012	Additional unexpected PDO(s) while last requested operation still in progress
	0x200A	MAC: Reverse power too high
	0x200F	MAC: Timeout
	0x2010	MAC commands not allowed at the time
	0x2011	Antenna frequencies could not be set
	0x2013	There is no active antenna configured
	0x2014	Frequency standard is not supported by the current firmware version
	0x2015	MAC: Power amplifier temperature threshold exceeded

Block Errorcode	Description								
	<table><tr><th>Errorcode</th><th>Description</th></tr><tr><td>0x2017</td><td>Wrong version of blocks.xml exist on the device</td></tr><tr><td>0x2018</td><td>MAC: Communication Error</td></tr><tr><td>0x20FF</td><td>MAC: Generic error</td></tr></table>	Errorcode	Description	0x2017	Wrong version of blocks.xml exist on the device	0x2018	MAC: Communication Error	0x20FF	MAC: Generic error
Errorcode	Description								
0x2017	Wrong version of blocks.xml exist on the device								
0x2018	MAC: Communication Error								
0x20FF	MAC: Generic error								
16#0011	Selected command is not supported in ReadEPC Mode (Mode = 0)								
16#0200	There are no parameters selected in #Data.ParamAccess.Selection. Select at least on parameter and re-trigger the command.								
16#0210	One or more values that shall be written to the parameters are not supported. Make sure the values are in the supported range and re-trigger the command.								