Weidmüller

UR20-FBC-EIP (All Variants)

Application Note for Importing module data from the u-remote web server into Studio 5000

Abstract:
The web server of all types of UR20 EIP couplers offers the option to export the module configuration as an l5x file which can be imported into Studio 5000 as a routine. All module-relevant data objects are created, tagged, and commented separately and are therefore easy to find.
This Application Note describes the import procedure of the file, the data structure of the additionally generated Controller Tags and the integration of the new routine into the main task.

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	Controller supported by Studio 5000 (e.g. Allen Bradley ControlLogix 1756-L72)	-	-
2	EtherNet/IP-compatible Ethernet interface for controller (e.g. 1756-EN2T)	-	-
3	UR20-FBC-EIP	1334920000	HW 02.xx.xx / FW 02.09.00 (or above)
4	UR20-FBC-EIP-ECO	2799510000	HW 01.xx.xx / FW 01.00.00 (or above)
5	UR20-FBC-EIP-V2	1550550000	HW 01.xx.xx / FW 01.00.01 (or above)
6	Weidmüller u-remote UR20 modules according to user requirements	-	-

Software reference

No.	Software name	Article No.	Software version
1	Rockwell Automation Studio 5000 Logix Designer	-	V26.01.00 (or above)

File reference

No.	Name	Description	Version
1	ethip-v1.6-weidmueller-ur20-fbc-EIP-HW2-1334920000.eds	Device description file for UR20-FBC-EIP	1.6 (or above)
2	ethip-v1.1-weidmueller-ur20-fbc-EIP-ECO-2799510000.eds	Device description file for UR20-FBC-EIP-ECO	1.1 (or above)
3	ethip-v1.1-weidmueller-ur20-fbc-EIP-V2-1550550000.eds	Device description file for UR20-FBC-EIP-V2	1.1 (or above)
4	USB_driver_signed.zip	Zip archive with driver files for Windows 7 or below (not needed for Windows 8 or above)	-

Contact
Weidmüller Interface GmbH & Co. KG
Klingenbergstraße 26
32758 Detmold, Germany
www.weidmueller.com

For any further support please contact your local sales representative:
https://www.weidmueller.com/countries

Content

1 Warning and Disclaimer

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 safety equipment provided / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer
This Application Note / Quick Start Guide / Example Program does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Each user is responsible for the correct operation of his control system. By using this Application Note / Quick Start Guide / Example Program 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
The given descriptions and examples do not represent any customer-specific solutions, they are simply intended to help for typical tasks. The user is responsible for the proper operation of the described products. Application notes / Quick Start Guides / Example Programs are not binding and do not claim to be complete in terms of configuration as well as any contingencies. By using this Application Note / Quick Start Guide / Example Program, you acknowledge that we cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this application note / quick start guide / example at any time without notice. In case of discrepancies between the proposals Application Notes / Quick Start Guides / Program Examples and other Weidmüller publications, like manuals, such contents have always more priority to the examples. We assume no liability for the information contained in this document. Our liability, for whatever legal reason, for damages caused using the examples, instructions, programs, project planning and performance data, etc. described in this Application Note / Quick Start Guide / Example 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.

2 Requirements and Preparation

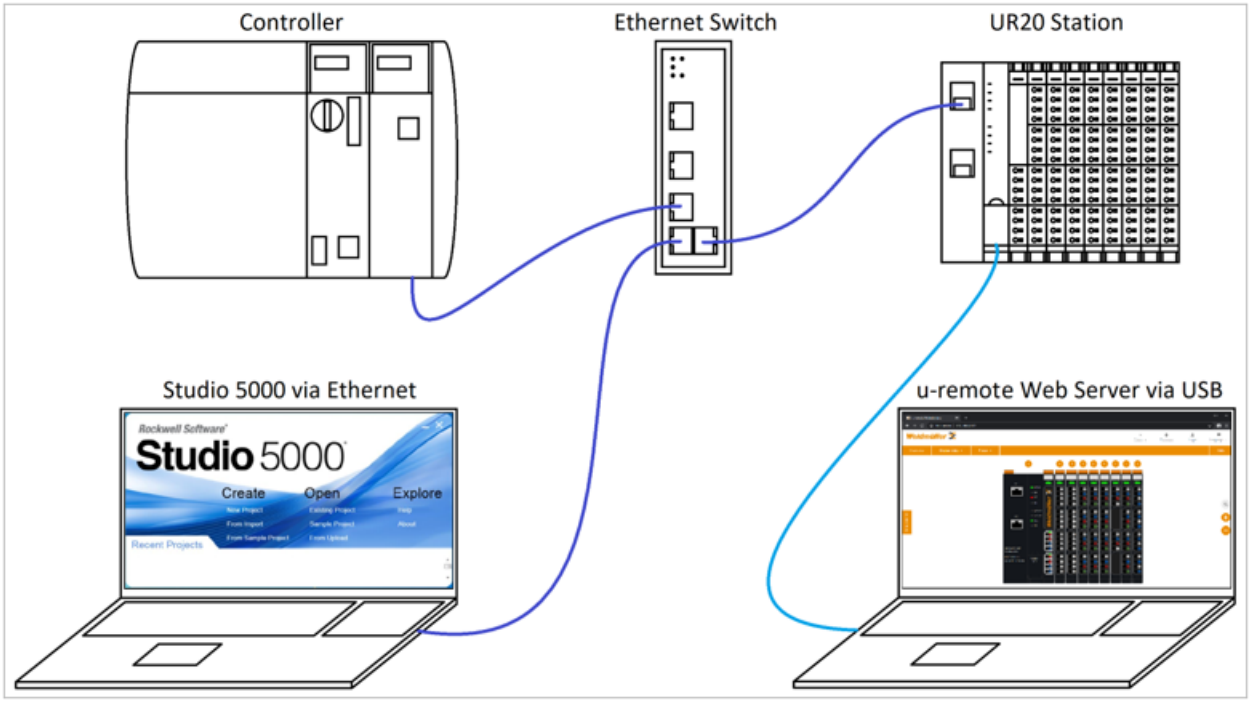


Figure 1: Application example

2.1 Related manuals

- Weidmüller 1432790000 Remote-I/O-System u-remote Manual.
- Weidmüller 2112220000 Remote-I/O-System u-remote Web server manual.
- For additional documentation of the controller and Studio 5000 please refer to https://www.rockwellautomation.com.

2.2 Preparation of Hardware and Software

The wiring of the application and the setting up of a project in Studio 5000 is not the subject of this Application Note and is assumed as given.

Caution: The entire application must be put into a safe state. Powering up the controller, UR20 coupler or any other device must not lead to any danger for life or equipment.

- All required UR20 modules must be attached to the coupler before it is supplied with power.
- A web browser needs to be connected to the web server of the UR20 EIP coupler
  - When using Ethernet, by default the coupler is in DHCP mode
  - The EIP-ECO coupler will additionally activate the IP 192.168.0.222 after 30 s if no DHCP response was received by then (this will not deactivate DHCP)

  - When using USB, default IP is: 192.168.5.202
- The following coupler parameters need to be set in the web server, if applicable (not all couplers offer all the below mentioned parameters):
  - Operation mode: Block device (default)
  - IP configuration: [as required] (recommended: Static)
  - IP address: [as required] (IP in the same subnet as the controller's IP)
  - Subnet mask: [as required] (same as the controller's subnet mask)
  - Redundancy protocol: [as required] (if not needed chose: NONE)
  - Use VLAN tagging (802.1Q): Disabled
- Controller and UR20 EIP coupler should be powered up and connected to the same network.
- Controller should be fault-free
- UR20 EIP coupler should only have the BF lit red, otherwise LEDs should be lit green only
- Studio 5000 must be connected to the controller.
- The Ethernet interface of the controller must be added to the Studio 5000 project (e.g. detected with "Discover Modules" and then added with "Create").

2.3 Adding the UR20-FBC-EIP to the Studio 5000 project

1. Download the current EDS zip file for the respective UR20 EIP coupler type in use from the Weidmüller website and unpack it on the system with the Studio 5000 installation.
2. Install the EDS file in Studio 5000 via "Tools" → "EDS Hardware Installation Tool.
3. Make sure Studio 5000 is online and the controller is in Program Mode.

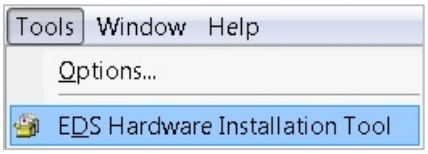


Figure 2: Studio 5000 EDS tool

4. In the Controller Organizer window right-click on the Ethernet interface and click on "Discover modules…".
5. In the new pop-up window "Select module type" the UR20 EIP coupler should be listed, select it, then click on "Create"

Figure 3: U-remote station "discovered" in the Ethernet network

Some controllers sometimes may have issues recognizing the UR20 coupler when using "Discover modules…" despite the installation of the EDS file. In that case close and re-open the project to make sure that the hardware list got loaded properly. Also, check at RockwellAutomation.com if there is a patch available for the controller in use. Alternatively, the u-remote coupler can be inserted manually.

6. In the new pop-up window "New Module", enter a device name.

For a better overview and to avoid larger renaming effort, it is advisable to enter here the same name as the l5x file will have that will be generated and imported later. The file is created according to the pattern: “[Coupler_type]_[Last octet of IP address]” (e.g.: UR20_FBC_EIP_105)

7. Click on “Change…”.

Caution: The following 3 steps use newly implemented and necessary features, which makes this implementation procedure different from the previously known one.

8. In the new pop-up window “Module Definition”, click on “Exclusive Owner” to open the drop-down menu of available connection types and select “Exclusive Owner Unpacked” (see orange marking in Figure 4).
9. Edit the Input Size according to the value given in the u-remote web server at the “Coupler” page, “General information” at **“Length of assembly input 111”** (see red markings in Figure 4).
10. Edit the Output Size according to the value given in the u-remote web server at the “Coupler” page, “General information” at **“Length of assembly output 112”** (see violet markings in Figure 4).

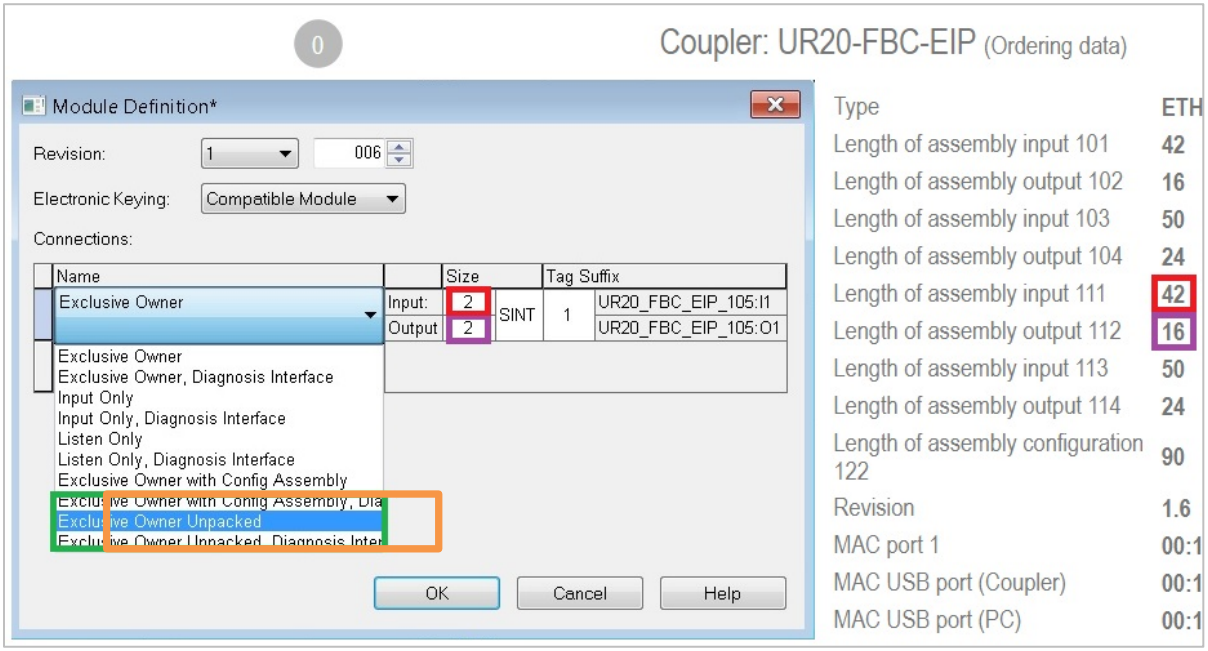


Figure 4: "Module Definition" window of Studio 5000 (still in default setting) and the required Assembly lengths shown in the coupler's “General Information” page of the u-remote web server

11. Click “OK” to close the “Module Definition” window; in case a warning appears, confirm with “Yes”.
12. Click “OK” to close the “New Module” window; in case a warning appears, confirm with “Yes”.

13. Click “Close” in the “Select Module Type” window.
14. If online, the changes should be active immediately, in this case continue with the next main bullet point of this section.
    - If not online, click on “Download”
    - In the new pop-up window “Download”, click the “Download” button
    - Wait until the “Download” window is automatically closed
15. Now the connection should be established and fault free
    - The "I/O" LED in Studio 5000 should light up permanently green and the text "I/O OK" should be displayed next to it.
    - The BF LED of the u-remote coupler should be off, and the web server should show “EIPConnected” and “EIPOperational” under “Diagnoses”.

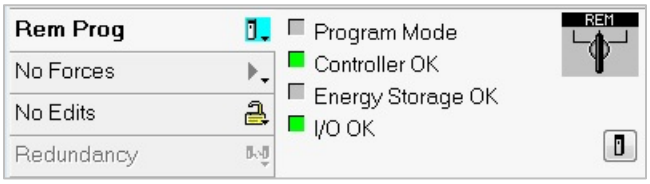


Figure 5: Virtual LEDs in Studio 5000 after the connection to the u-remote station is established

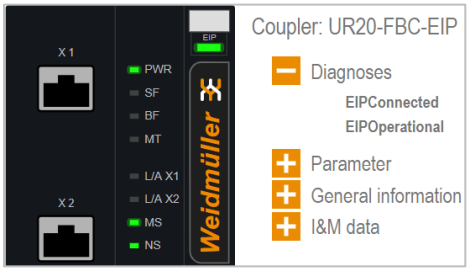


Figure 6: Virtual LEDs and Diagnoses in the u remote web server after the connection is established

3 The Hardware config file [Coupler_type]_[nnn].l5x

3.1 Saving the file in the u-remote web server

Make sure that the u-remote coupler has all the required modules attached before powering up the unit.

Make sure that the u-remote coupler has set or received the desired IP address. The file name will contain the last octet of the IP address as identifier. This identifier will also be included in the names of all objects generated from this file in Studio 5000. In this application note the identifier will be referred to as '[nnn]'.

1. Open the u-remote web server, if not already done.
2. Move the mouse arrow over "Extras" to open the drop-down menu
3. Click on "Hardware config file (.l5x)"
4. Save the file and, if necessary, copy it to the system of the Studio 5000 installation.

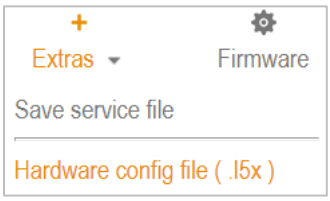


Figure 7: "Extras" menu

In case a file with the same name already exists, Windows will add a number in brackets to the file name. This will not affect the naming of generated objects by the import procedure of Studio 5000.

3.2 Import procedure in Studio 5000

This procedure can be done offline or online. If online, make sure the controller is in Program Mode and "I/O OK" is displayed.

1. In the menu bar click on "File".
2. Move the mouse arrow over "Import Component".
3. In the new drop-down menu click on "Routine".
4. In the new pop-up window "Import Routine", browse to the just saved/copied l5x file.
5. Double-click on the l5x file or select it and click on "Open".
6. Wait until the pop-up "Preparing import information" is automatically closed.
7. In the new pop-up "Import Configuration…", check for red flag symbols (errors) in the "Import Content" list and fix them if necessary.

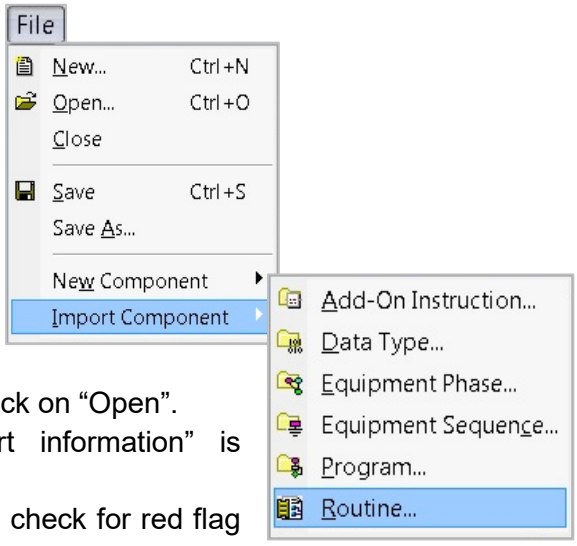


Figure 8: "File" menu

Red flags left of the “Tags” and the “Other Components” branches indicate that the names of l5x file and UR20 instance in the project do not match. In this case open the branches, find the flagged objects, and set their reference to the respective object in the project manually. This should eliminate the red flags. The execution of the import procedure will be blocked until all red flags in these branches are cleared.

A red flag left of the “Data Types” branch indicates that a l5x file from an UR20 coupler with the same IP address has been imported before and some data objects with the same name already exist. If this was not intended, abort. Otherwise open the branch, find the flagged objects, and change the “Operation” selection from “Use existing” to “Overwrite”. This will not clear the red flag, but it will not block the import procedure.

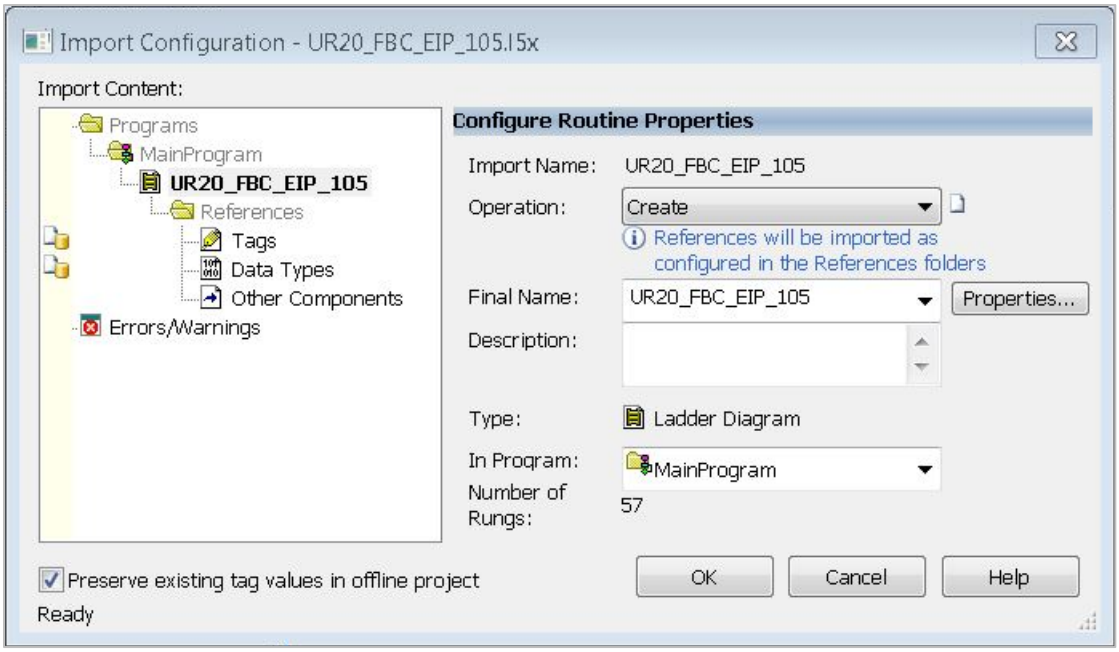


Figure 9: “Import Configuration" window without errors (no red flags) and ready for execution

8. Click on “OK”.
9. If online, in the new pop-up “Online Options”, click on option “Finalize All Edits In Program”, then click on “OK”.
10. Wait until the pop-up window “Performing Import” is automatically closed.
11. Make sure no errors have occurred during the import sequence
12. In the Controller Organizer open the branch “Tasks” / “MainTask” / “MainProgram”.
13. A new routine with the same name as the l5x file ([Coupler_type]_[nnn]) should be listed.

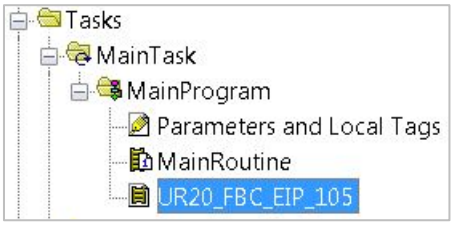


Figure 10: The new routine

3.3 Invoke the imported subroutine

1. Double-click on “MainRoutine”.
2. If required, create an empty rung.
3. Right-click on an empty rung to open the Context Menu.
4. Click on “Add Ladder Element”.
5. In the new pop-up “Add Ladder Element”, type in “JSR”.
6. Make sure “Jump To Subroutine” is selected, then click “OK”.
7. A JSR instruction now should be added to the selected rung.
8. In the instruction double-click on “Routine name”.
9. Click on the newly appeared triangle.
10. In the drop-down menu, click on the name of the just imported routine, then press the Return key.
11. Delete all other Input and Return Parameters of the instruction, e.g. by right-click on parameter and select “Remove Instruction Parameter.

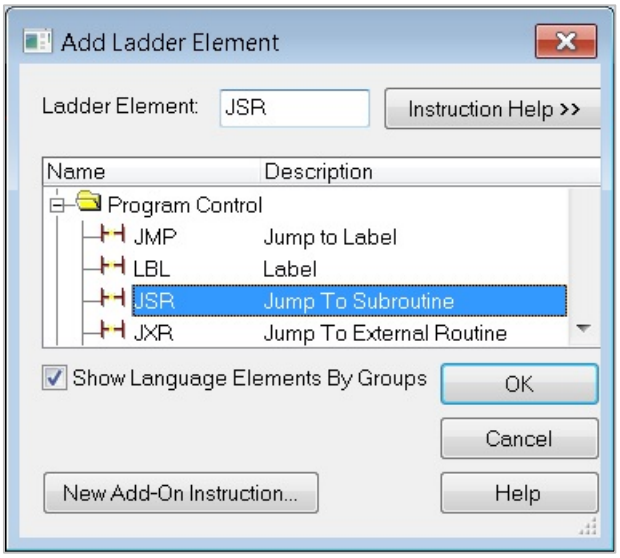


Figure 11: “Add Ladder Element” window

Figure 12: “Routine Name” parameter of JSR instruction

The JSR instruction should be executed without conditions.

12. **If offline**:
    a. Click on “Verify Routine”.
    b. Wait for the message to appear at the bottom right of the main window: “Verify complete with no errors or warnings.”
    c. Download the project to the controller.

    If online:
    a. Click on “Accept Pending Program Edits”
13. Now the new subroutine with the u-remote module data should be activated.

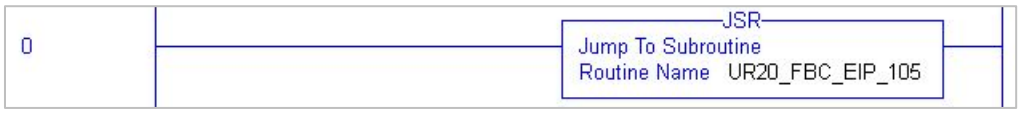


Figure 13: Verified rung with JSR instruction

4 Controller Tags of the UR20 station

After successfully importing the l5x file, new tags have been generated in addition to the 2 tags already created by the system when implementing the u-remote station into the project.

To allow the implementation of multiple u-remote stations without name duplication, the number of the last octet of the IP address of the coupler is appended to all tags as an identifier. In this application note the identifier will be referred to as '[nnn]'.

4.1 ID[nnn] Tags - do not edit!

The ID[nnn] Tags have been generated by the import procedure. They contain all Message instructions required for acyclic access to parameter and diagnostic data objects. The instructions are called via enable bits in the separately created tags for the modules under the tag [Coupler_type]_[nnn] (see chapter 5).

Name	Description	Data Type
ID105_InputDataLength	Message Command Input Data Data Length	MESSAGE
ID105_LengthParameterdata	Message Command Parameter Data Length	MESSAGE
ID105_MSG_Diagnostic	Message Get Diagnosis Module	MESSAGE
ID105_MSG_ProcessAlarm	Message Get Get Process Alram Module	MESSAGE
ID105_OutputDataLength	Message Command Output Data Data Length	MESSAGE
ID105_ReceiveMSG1	Message Command Parameter Receive Data	MESSAGE
ID105_ReceiveMSG2	Message Command Parameter Receive Data	MESSAGE
ID105_ReceiveMSG3	Message Command Parameter Receive Data	MESSAGE
ID105_SendMSG1	Message Command Parameter Send Data	MESSAGE
ID105_SendMSG2	Message Command Parameter Send Data	MESSAGE
ID105_SendMSG3	Message Command Parameter Send Data	MESSAGE

Figure 14: ID[nnn] Tags

Caution: Changing values of these tags could lead to unexpected behavior of the UR20 station. Therefore, they must not be edited.

4.2 [Coupler_type]_[nnn] Tag – access to all relevant data objects

Under this parent tag is the work area for processing the module data of the u-remote station. It is described in chapter 5.

Name	Description	Data Type
UR20_FBC_EIP_105	1st Station UR20_FBC_EIP	UDT_ID105

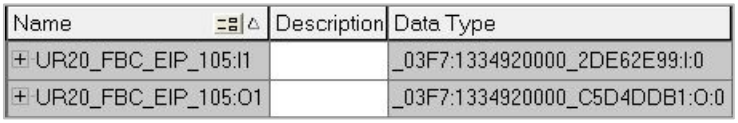
Figure 15: [Coupler_type]_[nnn] Tag

4.3 System generated Input and Output Tags – do not edit!

If the UR20 instance implemented in the project got the same name as the l5x file (see chapter 2.3), the system generated the following input and output tags.

- [Coupler_type]_[nnn]:I1
- [Coupler_type]_[nnn]:O1

These tags are now read and written by the newly created subroutine.



Name	Description	Data Type
UR20_FBC_EIP_105:I1		_03F7:1334920000_2DE62E99:I:0
UR20_FBC_EIP_105:O1		_03F7:1334920000_C5D4DDB1:O:0

Figure 16: System generated Input and Output Tags

Caution: Changing values of the Output Tags could lead to unexpected behavior of the UR20 station. Therefore, they must not be edited.

5 Structure of the data objects of the u-remote station

⊟ UR20_FBC_EIP_105	1st Station UR20_FBC_EIP	UDT_ID105
⊞ UR20_FBC_EIP_105.IN	1st Station UR20_FBC_EIP IN Data	UDT_ID105_I
⊞ UR20_FBC_EIP_105.OUT	1st Station UR20_FBC_EIP OUT Data	UDT_ID105_O
⊞ UR20_FBC_EIP_105.Status	1st Station UR20_FBC_EIP Statusword	UDT_ID105_Statusword
⊞ UR20_FBC_EIP_105.MISC	1st Station UR20_FBC_EIP MISC	UDT_ID105_MISC
UR20_FBC_EIP_105.WriteParameter	1st Station UR20_FBC_EIP Write all Parameter values for all UR20_FBC_EIP Modules	BOOL
UR20_FBC_EIP_105.ReadParameter	1st Station UR20_FBC_EIP Read all Parameter values for all UR20_FBC_EIP Modules	BOOL
UR20_FBC_EIP_105.EditParameter	1st Station UR20_FBC_EIP Allowed Edit all Parameter values for all UR20_FBC_EIP Modules	BOOL
UR20_FBC_EIP_105.SetDefaultParameter	1st Station UR20_FBC_EIP Reset all module parameters to their default value	BOOL

Figure 17: Expanded [Coupler_type]_[nnn] parent tag

5.1 Module Input data [Coupler_type]_[nnn].IN

For each module a sub tag was created here with the slot number and the module's name.

⊟ UR20_FBC_EIP_105.IN	1st Station UR20_FBC_EIP IN Data	UDT_ID105_I
⊞ UR20_FBC_EIP_105.IN.M01_UR20_16DI_P	1st Station UR20_FBC_EIP UDT for UR20_16DI_P- Module	UDT_015_I
⊞ UR20_FBC_EIP_105.IN.M02_UR20_16DI_P	1st Station UR20_FBC_EIP UDT for UR20_16DI_P- Module	UDT_015_I
⊞ UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16- Module	UDT_045_I
⊞ UR20_FBC_EIP_105.IN.M04_UR20_4AI_UI_16_DIAG	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16_DIAG- Module	UDT_046_I
⊞ UR20_FBC_EIP_105.IN.M05_UR20_4AI_UI_16_DIAG	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16_DIAG- Module	UDT_046_I
⊞ UR20_FBC_EIP_105.IN.M06_UR20_2CNT_100	1st Station UR20_FBC_EIP UDT for UR20_2CNT_100- Module	UDT_032_I
⊞ UR20_FBC_EIP_105.IN.M07_UR20_4DO_P_2A	1st Station UR20_FBC_EIP UDT for UR20_4DO_P_2A- Module	UDT_077_I
⊞ UR20_FBC_EIP_105.IN.M08_UR20_8DO_P	1st Station UR20_FBC_EIP UDT for UR20_8DO_P- Module	UDT_103_I

Figure 18: Expanded *.IN sub tag of an UR20 example station

Under the module sub tag the different types of input data are further subdivided:

- ***.Data** is only generated if the module supports input process data. Data type and size are variable and are created module-specific.
- ***.Diagnostic** lists the 47 Byte diagnostic data of the module
- ***.Process Alarm** is a 4 Byte array object that is generated for each module. However, it only contains relevant data if the module supports this function and it is activated in the module and coupler parameters.
- ***.StatusDiagAlarm** is a flag that shows if the module has an unacknowledged diagnose pending
- ***.StatusProcessAlarm** is a flag that shows if the module has an unacknowledged process alarm pending

All input data is updated with each fieldbus cycle (RPI).

UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16- Module	UDT_045_I
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.Data	1st Station UR20_FBC_EIP UR20_4AI_UI_16 generic Module Data Data	UDT_045_Data_I
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.Data.Ch_00	1st Station UR20_FBC_EIP UR20_4AI_UI_16 Process Data Ch_00	INT
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.Data.Ch_01	1st Station UR20_FBC_EIP UR20_4AI_UI_16 Process Data Ch_01	INT
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.Data.Ch_02	1st Station UR20_FBC_EIP UR20_4AI_UI_16 Process Data Ch_02	INT
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.Data.Ch_03	1st Station UR20_FBC_EIP UR20_4AI_UI_16 Process Data Ch_03	INT
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.Diagnostic	1st Station UR20_FBC_EIP UR20_4AI_UI_16 generic Module Data Diagnostic	UDT_Diagnostic
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.ProcessAlarm	1st Station UR20_FBC_EIP UR20_4AI_UI_16 generic Module Data ProcessAlarm	UDT_ProcessAlarm
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.StatusDiagAlarm	1st Station UR20_FBC_EIP UR20_4AI_UI_16 generic Module Data StatusDiagAlarm	BOOL
UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16.StatusProcessAlarm	1st Station UR20_FBC_EIP UR20_4AI_UI_16 generic Module Data StatusProcessAlarm	BOOL

Figure 19: Expanded *.IN.* module sub tag of an UR20-4AI-UI-16 in slot 3

5.2 Module Output data [Coupler_type]_[nnn].OUT

For each module that has output process data and/or parameter data, a sub tag was created here with the slot number and the module's name.

UR20_FBC_EIP_105.OUT	1st Station UR20_FBC_EIP OUT Data	UDT_ID105_O
UR20_FBC_EIP_105.OUT.M03_UR20_4AI_UI_16	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16- Module	UDT_045_O
UR20_FBC_EIP_105.OUT.M04_UR20_4AI_UI_16_DIAG	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16_DIAG- Module	UDT_046_O
UR20_FBC_EIP_105.OUT.M05_UR20_4AI_UI_16_DIAG	1st Station UR20_FBC_EIP UDT for UR20_4AI_UI_16_DIAG- Module	UDT_046_O
UR20_FBC_EIP_105.OUT.M06_UR20_2CNT_100	1st Station UR20_FBC_EIP UDT for UR20_2CNT_100- Module	UDT_032_O
UR20_FBC_EIP_105.OUT.M07_UR20_4DO_P_2A	1st Station UR20_FBC_EIP UDT for UR20_4DO_P_2A- Module	UDT_077_O
UR20_FBC_EIP_105.OUT.M08_UR20_8DO_P	1st Station UR20_FBC_EIP UDT for UR20_8DO_P- Module	UDT_103_O

Figure 20: Expanded *.Out sub tag of an UR20 example station

Under the module sub tag the different types of output data are further subdivided:

- ***.Data** is only generated if the module supports output process data. Data type and size are variable and are created module-specific. Output process data is transmitted with each fieldbus cycle.
- *.Parameter is only generated if the module has editable parameters. Each parameter is represented by a separate sub tag with consisting of the channel number, if applicable, and the name of the parameter. Data type and size are variable and created parameter-specific. Available parameter options and the respective **values representing these options are listed in the "Description" column** of each parameter. The transmission of module parameters is controlled with the parameter handling bits (see chapter 5).

The parameter option values to be used here are raw data that do not correspond to the easily calculated option values of Class 103 "Module Parameters".

All available options of each module parameter can be displayed in a tooltip by hovering the mouse arrow over the respective cell in the "Description" column.

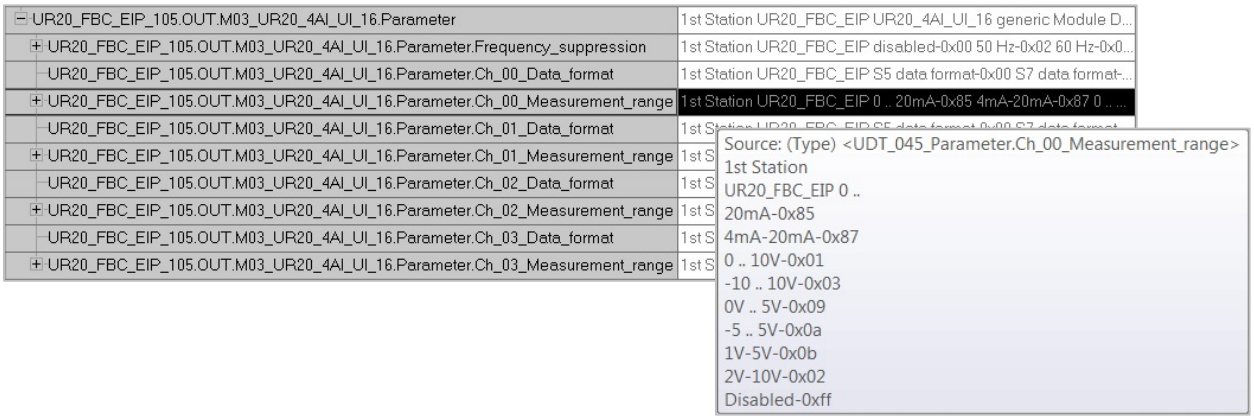


Figure 21: Expanded *.Parameter tag of an UR20-4AI-UI-16 module with the tooltip with all available options and the respective option value in hexadecimal for the parameter "Channel 0 Measurement range"

5.3 Coupler Status Word [Coupler_type]_[nnn].Status

This controller tag contains the 2 Byte header of the Input Assembly which represents the Coupler Status Word.

5.4 Auxiliary Tags [Coupler_type]_[nnn].MISC – do not edit!

All the tags that are required for the background processes of the subroutine are stored here.

Caution: Changing values of these tags could lead to unexpected behavior of the UR20 station. Therefore, they must not be edited.

5.5 Parameter control bits

These bits enable read and write access to module parameters.

- ***.WriteParameter** activates the transmission of all values of the *.parameter tags to the coupler, self-resetting, works only when the ***.EditParameter** Bit is set.
- ***.ReadParameter** activates the copying of the current parameter setup from the coupler into the *.parameter tags, self-resetting, also resets ***.EditParameter** Bit.
- ***.EditParameter** enables write access to module parameters.
- ***.SetDefaultParameter** copies module default parameters in the *.parameter tags and resets the ***.EditParameter Bit**.

Caution: Module parameters are written in bulk. To avoid unintentional changes, it is advisable to first read the current parameter setting from the coupler before each write access. Then this parameter image can be edited and afterwards written. That way it can be ensured that all unedited parameter values match the ones on the coupler.

To edit module parameters the following procedure is recommended:

1. Read the current parameter status by setting the Read Bit.
2. Enable parameter write access by setting the Edit Bit.
3. Enter the desired parameter data in the *.Out area.
4. Transfer the changed parameter image by setting the Write Bit.
5. Optional: Reset the Edit Bit.

UR20_FBC_EIP_105.WriteParameter	1st Station UR20_FBC_EIP Write all Parameter values for all UR20_FBC_EIP Modules	BOOL
UR20_FBC_EIP_105.ReadParameter	1st Station UR20_FBC_EIP Read all Parameter values for all UR20_FBC_EIP Modules	BOOL
UR20_FBC_EIP_105.EditParameter	1st Station UR20_FBC_EIP Allowed Edit all Parameter values for all UR20_FBC_EIP Modules	BOOL
UR20_FBC_EIP_105.SetDefaultParameter	1st Station UR20_FBC_EIP Reset all module parameters to their default value	BOOL

Figure 22: Parameter control bits

6 Examples

1. An UR20-FBC-EIP with the IP **192.168.100.105** is connected to the controller.
2. An UR20-FBC-EIP instance was implemented in a Studio 5000 project with the name UR20_FBC_EIP_105.
3. A hardware config file with the name UR20_FBC_EIP_105.l5x was imported as routine.
4. The third module behind the coupler is an UR20-4AI-UI-16-DIAG.

6.1 Parameterizing the measurement range of an analogue input

Setting an UR20-4AI-UI-16-DIAG, Channel 0 Measurement range to “4 – 20mA”:

1. Read the current parameter image of the coupler into the .OUT. tag area:
   UR20_FBC_EIP_105.ReadParameter := True;

2. Enable parameter write access:
   UR20_FBC_EIP_105.EditParameter := True;

3. Find the required value representing “4 – 20mA” in the description column of the parameter tag, then set the parameter:
   UR20_FBC_EIP_105.OUT.M03_UR20_4AI_UI_16_DIA.Parameter
   .Ch_00_Measurement_Range := 16#87;

4. Transfer the edited parameter image to the coupler:
   UR20_FBC_EIP_105.WriteParameter := True;

5. Optional: Disable parameter write access:
   UR20_FBC_EIP_105.EditParameter := False;

6.2 Reading the analogue input channel

`myINTvariable` := UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16_DIAG.Data.Ch_00;

6.3 Reading first Byte of Module Diagnoses on a Diagnose Event

- Coupler Parameter “Diagnostic alarm” must be enabled (not supported by the routine).
- Module Parameter “Channel 0, Channel diagnosis” must be enabled.

```
myBOOLvariable := UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16_DIAG.StatusDiagAlarm;
IF myBOOLvariable THEN
        mySINTvariable :=
                UR20_FBC_EIP_105.IN.M03_UR20_4AI_UI_16_DIAG.Diagnostic.Err_A;
END_IF;
```

7 FAQ

1. **Where are the coupler parameters?**
   So far, the l5x file creation routine in the web server takes its data solely from the communication between coupler and modules. Coupler parameters don't get send to the modules, therefore in this version no coupler parameters are available. However, it is planned to add them in a future release.

2. **Can the process data length of an UR20-4COM-IO-LINK be edited in an existing routine?**
   Changing the process data length of the IO-LINK module changes the data stream between coupler and modules from which the imported routine was derived. Therefore, a new l5x file needs to be created and imported to replace the previous routine. Also, the Input and/or Output length in the Studio 5000 Module Definitions of the UR20 EIP coupler needs to be adapted before the controller can reconnect to the coupler.

3. **Can new modules be added to an existing imported routine?**
   Again, additional modules change the data stream between coupler and modules from which the imported routine was derived. A new routine needs to be imported to replace the old one and the I/O length in the Module Definitions needs to be adapted.

4. **Can a l5x file be created without having the u-remote station, yet?**
   Sadly not. There are possible future solutions under evaluation.