

DNNF012

DNNF020

TCP/IP or UDP Integration of uniVision Products into Control Systems



Interface Protocol

Table of Contents

1. Use for Intended Purpose 3

2. Network Overview 4

3. Settings in uniVision 5

4. TIA Sample Program 11

4.1 Receiving Process Data from TCP Device 11

4.2 Receiving Process Data from UDP Device 15

4.3 Sending LIMA Commands via TCP/IP and Receiving LIMA Answers 18

5. TwinCAT3 Sample Programs 24

5.1 Receiving Process Data from TCP Device 25

5.2 Receiving Process Data from UDP Device 27

5.3 Sending LIMA Commands via TCP/IP and Receiving LIMA Answers 29

6. Rockwell Sample Programs 32

6.1 Receiving Process Data from the TCP Device 32

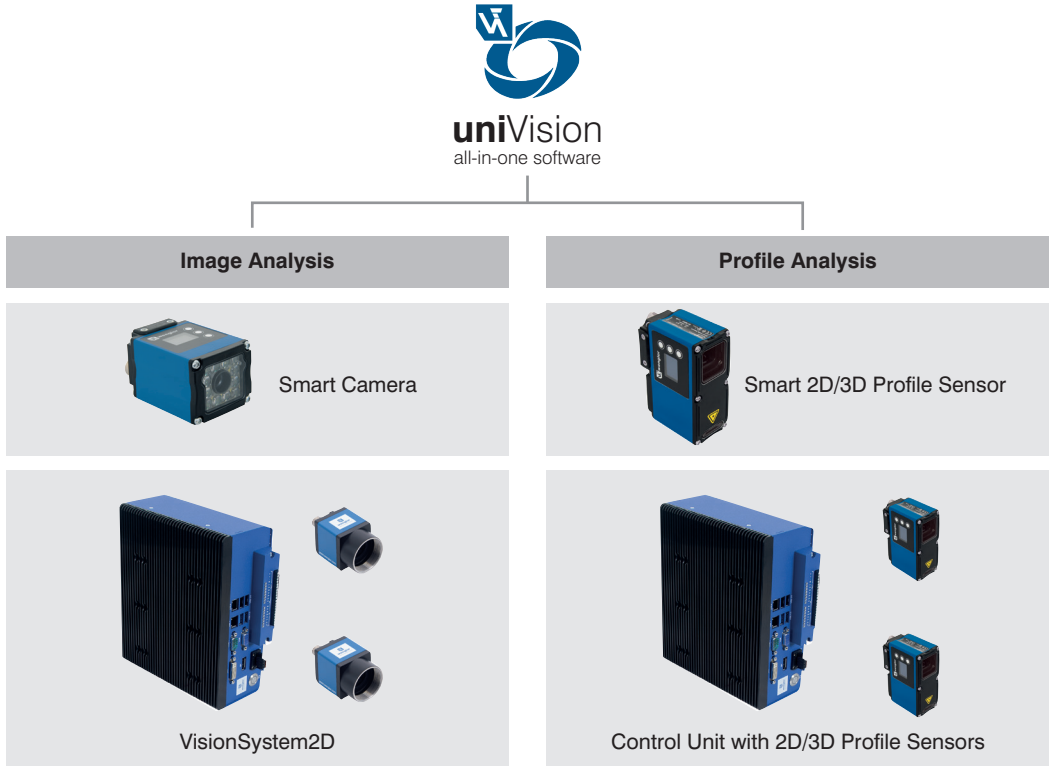
6.2 Receiving Process Data from the UDP Device 34

6.3 Sending LIMA Commands via TCP/IP and Receiving LIMA Responses 36

1. Use for Intended Purpose

The instructions show, for example, integration of uniVision products into various control environments via the TCP/IP or UDP interface. These instructions are in addition to the control sample programs and show, among other things, which changes are necessary for a different network configuration or for a different number of characters transmitted via TCP/IP or UDP.

The following uniVision products can be integrated in this way:



The sample programs are available for the following control environments:

- Siemens PLC S7-1200 with TIA Portal V15
- Beckhoff TwinCAT3
- SPS 1769-L18ERM-BB1B from Allen-Bradley with Studio 5000 Logix Designer V32

Depending on the control environment, the sample program contains a different scope of functions. In general, the following functions are possible in the control sample programs:

- Receiving process data from the TCP device
- Receiving process data from the UDP device
- Sending LIMA commands (e.g. trigger commands) via TCP/IP and receiving LIMA answers

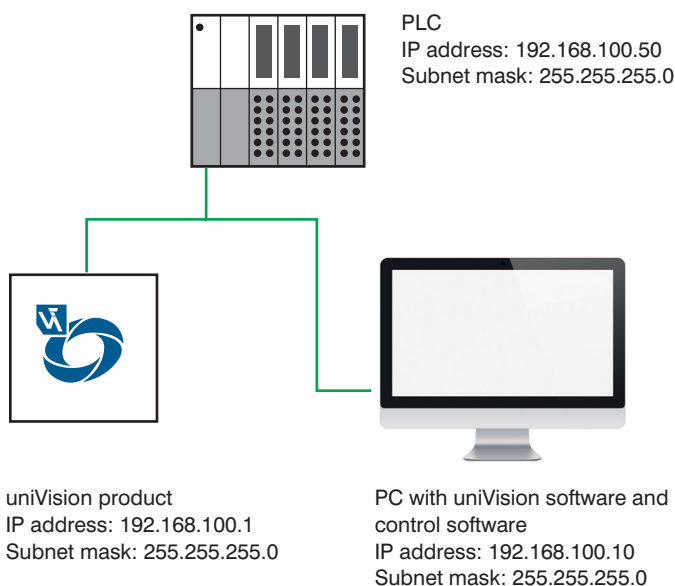


NOTE!

The control sample programs are supported starting with uniVision version 2.4.0.

2. Network Overview

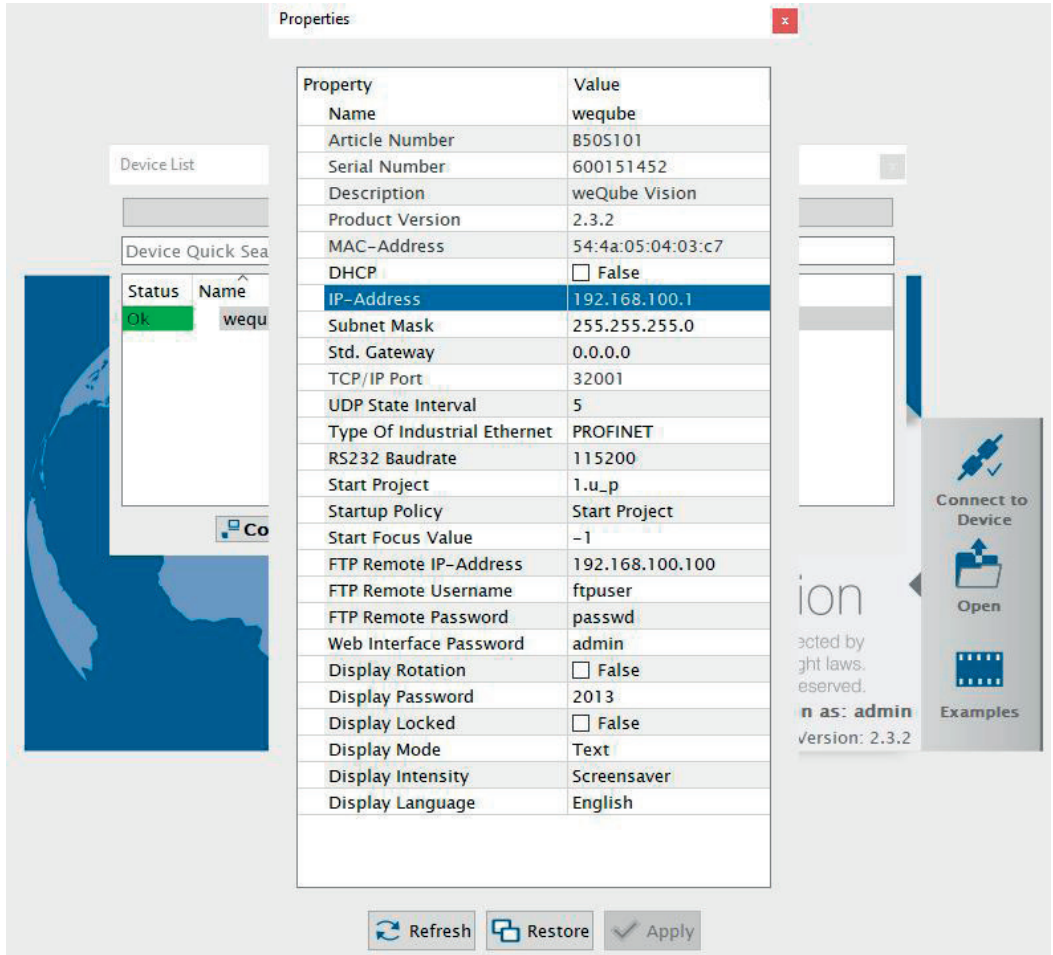
The uniVision product, the PLC and PC with uniVision software, and the control software must be on the same network. The following network settings are used in the sample program.



3. Settings in uniVision

The following steps are necessary to start the uniVision software:

1. Install and open uniVision software for Windows (article number: DNNF020)
2. Set up the network configuration of the uniVision product via the uniVision software. To do this, select the uniVision product in the device list and click Properties.



| Property Name | Value |
|-----------------------------|--------------------------------|
| Name | weqube |
| Article Number | B50S101 |
| Serial Number | 600151452 |
| Description | weQube Vision |
| Product Version | 2.3.2 |
| MAC-Address | 54:4a:05:04:03:c7 |
| DHCP | <input type="checkbox"/> False |
| IP-Address | 192.168.100.1 |
| Subnet Mask | 255.255.255.0 |
| Std. Gateway | 0.0.0.0 |
| TCP/IP Port | 32001 |
| UDP State Interval | 5 |
| Type Of Industrial Ethernet | PROFINET |
| RS232 Baudrate | 115200 |
| Start Project | 1.u_p |
| Startup Policy | Start Project |
| Start Focus Value | -1 |
| FTP Remote IP-Address | 192.168.100.100 |
| FTP Remote Username | ftpuser |
| FTP Remote Password | passwd |
| Web Interface Password | admin |
| Display Rotation | <input type="checkbox"/> False |
| Display Password | 2013 |
| Display Locked | <input type="checkbox"/> False |
| Display Mode | Text |
| Display Intensity | Screensaver |
| Display Language | English |

Buttons at the bottom: Refresh, Restore, Apply

3. Double-click to connect to the uniVision product and load a template onto the product.

4. Set the trigger mode to software or trigger in order to later use the LIMA interface via TCP/IP and send trigger commands to the uniVision device.

Navigator

Module Application

Device Camera

Module Localizer

Module Region

Module Threshold

Module Counter

Device IO Unit

Device TCP

Add Module

Start Assistant

| Property | Value | |
|------------------------|-------------------------------------|--|
| Light Internal | <input checked="" type="checkbox"/> | |
| Light External | <input type="checkbox"/> | |
| Rotate Input Image | <input type="checkbox"/> | |
| Create HSV Image | <input checked="" type="checkbox"/> | |
| Create RGB Image | <input type="checkbox"/> | |
| Create Raw Image | <input type="checkbox"/> | |
| Create BGRA Image | <input type="checkbox"/> | |
| Exposure Time [us] | 200 | |
| Focus Position [steps] | 141 | |
| Auto Focus | <input type="checkbox"/> | |
| Light Current [%] | 20 | |
| Light Mode | Flash Light | |
| Trigger Mode | Trigger | |

5. In order to send process data via TCP/IP or UDP, the TCP or UDP device must also be available in the project tree and configured accordingly.



NOTE!
The TCP device and UDP device for communicating with the control system are already preconfigured in the template. Alternatively, a new project can be created and the TCP or UDP device added manually to the project from the toolbar.

4. Any character count, preamble, separator and postamble can be configured on the TCP or UDP device. In addition, the output mode should be set to “Formatted” in order to define a fixed character count. This makes it easier to read out the process data on the control system.

Navigator

Module Application

Device Camera

Module Region

Module Threshold

Module Counter

Device IO Unit

Device TCP

Device UDP

Add Module

| Property | Value | |
|-------------------|---|--|
| Process Time [us] | 1000 | |
| Module State | 0 | |
| Output | +0027958,+0005748,+0016000,+0000000,+0035302,+0004300,+0001448,+0000000,+0006000; | |
| Preamble | | |
| Postamble | ; | |
| Delimiter | , | |
| String Count | 9 | |
| Output Mode | Formatted | |
| Error Handling | Value Substitution | |
| Connections | 5 | |
| TCP Port | 32002 | |
| Blocking Mode | <input type="checkbox"/> | |

5. If the output mode is set to “Formatted”, the character count for the various data types can be configured under “Formatting options”.



NOTE!
In the example, a total of eight characters are used for “integral numbers” and “floating point numbers” (incl. sign and comma). A character is used for bool data type results.

Navigator

Module Application

Device Camera

Module Region

Module Threshold

Module Counter

Device IO Unit

Device TCP

String Count

Error Handling

Formatting Options

Integer

Floating Point

Boolean

Device UDP

Add Module

| Property | Value |
|---------------------|-------------------------------------|
| Digits Before Comma | 4 |
| Digits After Comma | 2 |
| Print + | <input checked="" type="checkbox"/> |

- The character count should also be selected for the replacement value defined under troubleshooting. In the example, eight characters are also used for the error replacement value.

Navigator

Module Application

Device Camera

Module Region

Module Threshold

Module Counter

Device IO Unit

Device TCP

String Count

Error Handling

Formatting Options

Integer

Floating Point

Boolean

Device UDP

Add Module

Property

Value

Substitute STRING Types by

Error###

7. The total number of characters sent via TCP or UDP can be determined under “Output” on the TCP or UDP device.

Navigator

Module Application

- Device Camera
- Module Region
- Module Threshold
- Module Counter
- IO Device IO Unit
- Device TCP
- Device UDP
- Add Module

| Property | Value |
|-------------------|---|
| Process Time [us] | 1000 |
| Module State | 0 |
| Output | +0027958,+0005748,+0016000,+0000000,+0035302,+0004300,+0001448,+0000000,+0006000; |
| Preamble | |
| Postamble | ; |
| Delimiter | , |
| String Count | 9 |
| Output Mode | Formatted |
| Error Handling | Value Substitution |
| Connections | 5 |
| TCP Port | 32002 |
| Blocking Mode | <input type="checkbox"/> |

8. Save the project on the uniVision device and store it as a starter project in the device’s properties.

4. TIA Sample Program

The TIA sample program is created with a Siemens PLC S7-1200 with TIA Portal V15. It includes the following use cases:

- Receiving process data from the TCP device
- Receiving process data from the UDP device
- Sending LIMA commands (e.g. trigger commands) via TCP/IP and receiving LIMA answers

4.1 Receiving Process Data from TCP Device

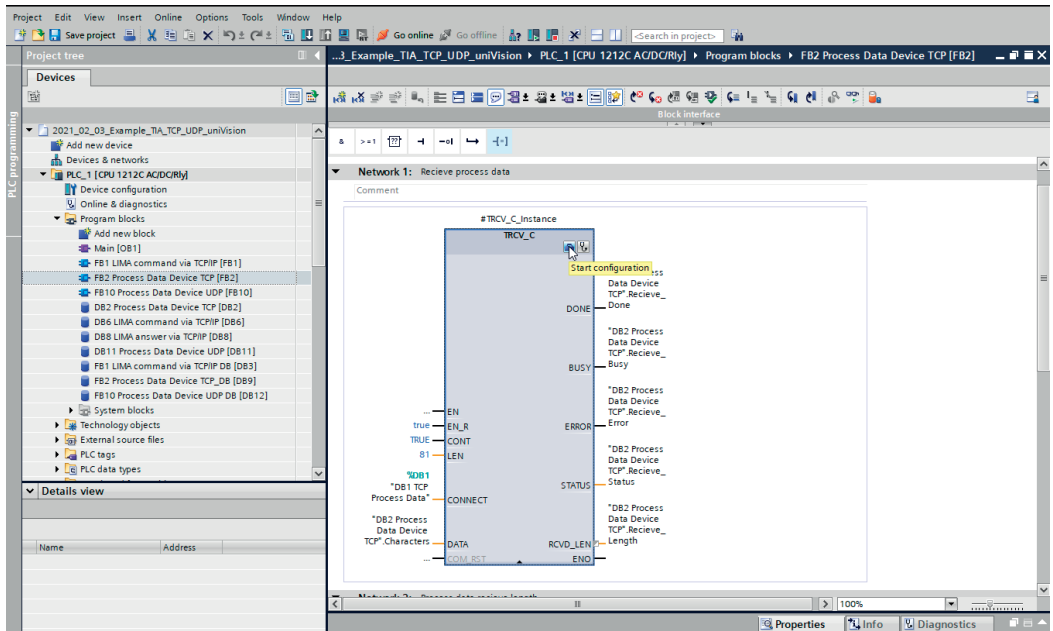
The TIA sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
- Subnet mask: 255.255.255.0

The TCP process data is sent via port 32002 by default.

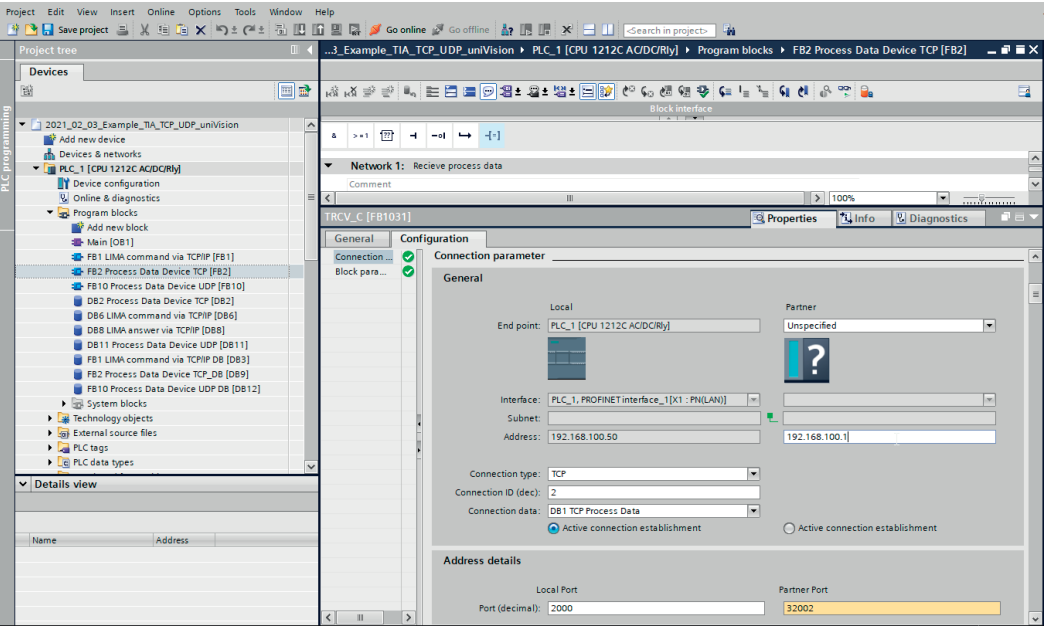
If a different network setting or another port is used on the uniVision product, the sample program must be adapted accordingly.

To do this, open the function module “FB2 Process Data Device TCP” and click on “Start Configuration” on network 1 “Receive process data”.



The screenshot displays the Siemens TIA Portal V15 interface. On the left, the 'Project tree' shows the hierarchy: '2021_02_03_Example_TIA_TCP_UDP_uniVision' > 'PLC_1 [CPU 1212C AC/DC/Rly]' > 'Program blocks' > 'FB2 Process Data Device TCP [FB2]'. The 'Details view' at the bottom left shows a table with columns 'Name' and 'Address'. The main workspace shows 'Network 1: Receive process data'. The network diagram includes a function block '#TRCV_C_Instance' of type 'TRCV_C'. It has several inputs and outputs: 'EN' (true), 'EN_R' (TRUE), 'COINT' (B1), 'LEN' (CONNECT), 'DATA' (DB2 Process Data Device TCP Characters), and 'RQVD_LEN' (ENQ). The outputs are 'DONE' (Data Device TCP Receive_Done), 'BUSY' (Busy), 'ERROR' (Error), 'STATUS' (Status), and 'LENGTH' (Length). A 'Start configuration' button is visible near the function block.

Enter the IP address and port under “Partner”.

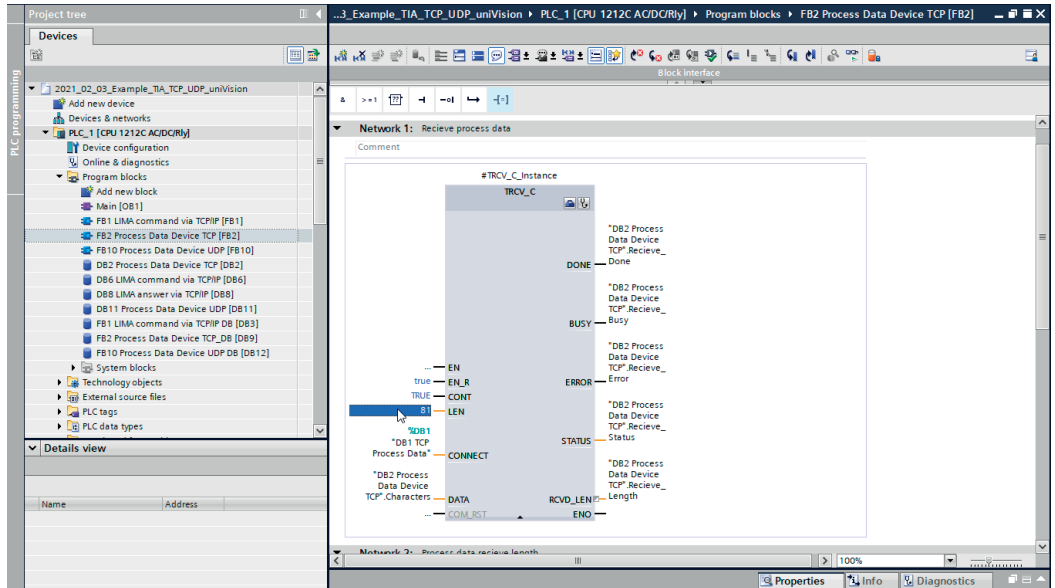


The TIA sample program is created for process data with a length of 81 characters. If a different character count is required, the sample program must be adapted accordingly.

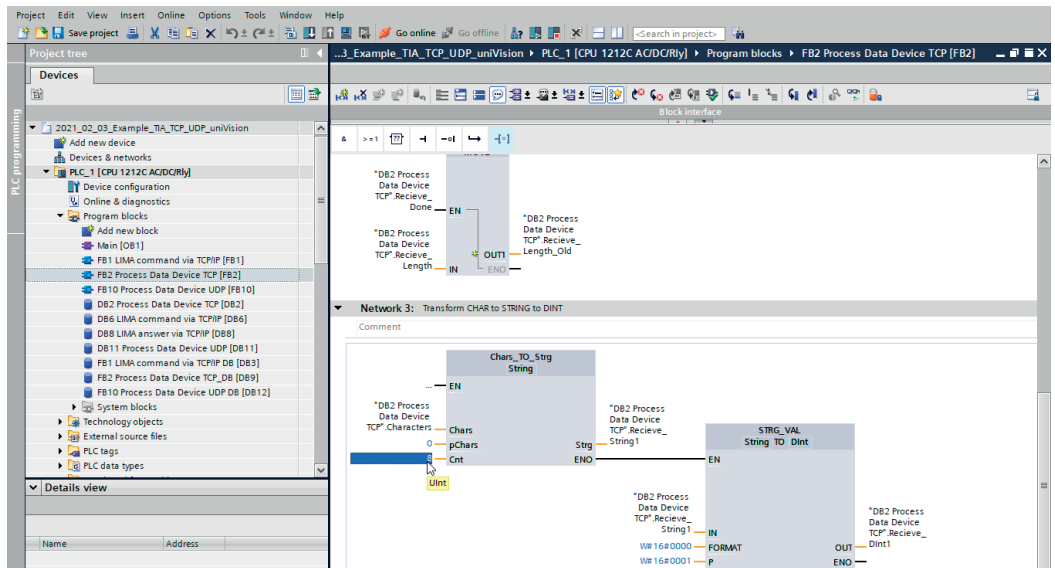


NOTE!
The total number of characters sent via TCP as process data can be determined in the uniVision software on the TCP device under “Output” (see section 3 “Settings in uniVision” on page 5). Preamble, separator and postamble as well as prefixes must be included in the character count!

To do this, adjust the character count under “LEN” on network 1 “Receive process data”.



The sample program also includes direct conversion of the characters into an integer (DINT) for the first string. The number of characters or data type for the first integer can be changed as desired.



Compile the sample program, load it onto the control system and connect it online.

The “DB2 Process Data Device TCP” data block receives the process data sent by the TCP device. The data is received as individual characters (Char).

The screenshot shows the 'DB2 Process Data Device TCP' data block configuration in the TIA Portal. The 'Characters' array is set to 'Array[0..100]' and contains 29 individual character elements, each of type 'Char'. The 'Monitor value' column shows the current value for each character, ranging from '3' to '9'.

| Name | Data type | Start value | Monitor value | Retain | Accessible f... | Write... | Visible in ... | Setpoint |
|----------------|---------------|-------------|---------------|--------|-----------------|----------|----------------|----------|
| Static | | | | | | | | |
| Characters | Array[0..100] | | | | | | | |
| Characters[0] | Char | | 3 | | | | | |
| Characters[1] | Char | | 3 | | | | | |
| Characters[2] | Char | | 5 | | | | | |
| Characters[3] | Char | | 5 | | | | | |
| Characters[4] | Char | | 5 | | | | | |
| Characters[5] | Char | | 5 | | | | | |
| Characters[6] | Char | | 0 | | | | | |
| Characters[7] | Char | | 0 | | | | | |
| Characters[8] | Char | | 0 | | | | | |
| Characters[9] | Char | | 3 | | | | | |
| Characters[10] | Char | | 3 | | | | | |
| Characters[11] | Char | | 3 | | | | | |
| Characters[12] | Char | | 5 | | | | | |
| Characters[13] | Char | | 5 | | | | | |
| Characters[14] | Char | | 5 | | | | | |
| Characters[15] | Char | | 0 | | | | | |
| Characters[16] | Char | | 0 | | | | | |
| Characters[17] | Char | | 0 | | | | | |
| Characters[18] | Char | | 3 | | | | | |
| Characters[19] | Char | | 3 | | | | | |
| Characters[20] | Char | | 5 | | | | | |
| Characters[21] | Char | | 5 | | | | | |
| Characters[22] | Char | | 5 | | | | | |
| Characters[23] | Char | | 5 | | | | | |
| Characters[24] | Char | | 0 | | | | | |
| Characters[25] | Char | | 0 | | | | | |
| Characters[26] | Char | | 0 | | | | | |
| Characters[27] | Char | | 3 | | | | | |
| Characters[28] | Char | | 3 | | | | | |

For the first string, conversion to another data type is shown on the DINT for example purposes.

The screenshot shows the 'DB2 Process Data Device TCP' data block configuration in the TIA Portal. The 'Characters' array is set to 'Array[0..100] of Char'. The 'Recieve_Dint1' field is set to 'Dint' and shows the value '5081'.

| Name | Data type | Start value | Monitor value | Retain | Accessible f... | Write... | Visible in ... | Setpoint |
|--------------------|-----------------------|-------------|---------------|--------|-----------------|----------|----------------|----------|
| Static | | | | | | | | |
| Characters | Array[0..100] of Char | | | | | | | |
| Recieve_Done | Bool | false | FALSE | | | | | |
| Recieve_Busy | Bool | false | TRUE | | | | | |
| Recieve_Error | Bool | false | FALSE | | | | | |
| Recieve_Status | Word | 16#0 | 16#7D02 | | | | | |
| Recieve_Length | Int | 0 | 0 | | | | | |
| Recieve_Length_Old | Int | 0 | 81 | | | | | |
| Recieve_String1 | String | | "-0005081" | | | | | |
| Recieve_Dint1 | Dint | 0 | 5081 | | | | | |

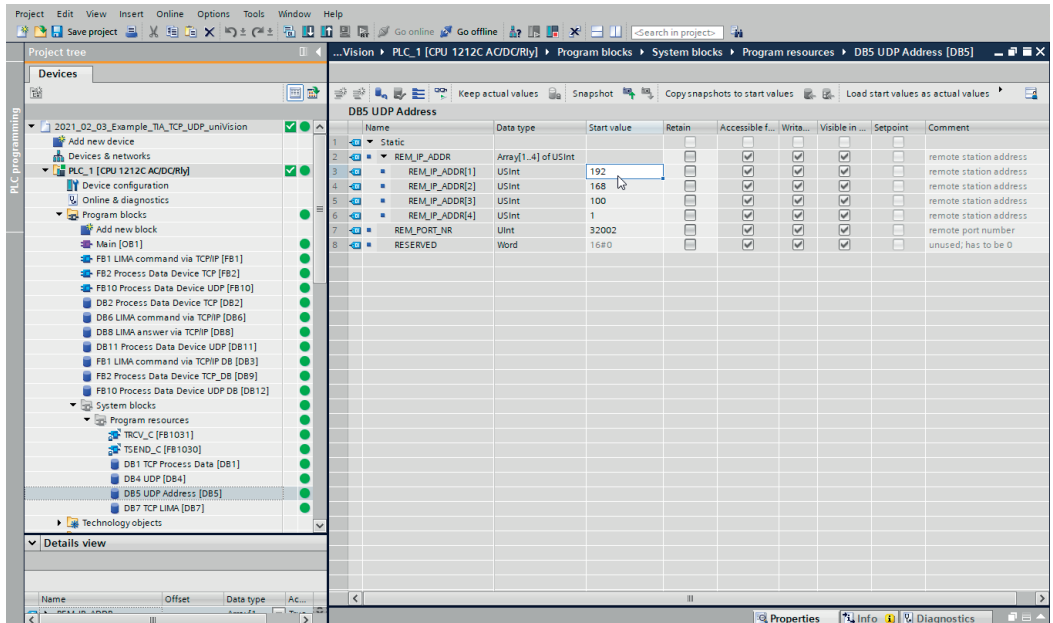
4.2 Receiving Process Data from UDP Device

The TIA sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
 - Subnet mask: 255.255.255.0
- UDP process data is sent via port 32002.

If a different network setting is used on the uniVision product, the sample program must be adapted accordingly.

To do this, open the data block “DB5 UDP Address” under “System blocks” and enter the IP address of the uniVision product under “REM_IP_ADDR”.



The screenshot shows the TIA Portal software interface. On the left, the 'Project tree' displays the hierarchy: '2021_02_03_Example_TIA_TCP_UDP_uniVision' > 'PLC_1 [CPU 1212C AC/DC/Ry]' > 'System blocks' > 'Program resources' > 'DB5 UDP Address [DB5]'. The main window displays the 'DB5 UDP Address' data block configuration table.

| Name | Data type | Start value | Retain | Accessible f... | Write... | Visible in ... | Setpoint | Comment |
|----------------|---------------------|-------------|--------|-----------------|----------|----------------|----------|------------------------|
| Static | | | | | | | | |
| REM_IP_ADDR | Array[1..4] of UInt | | | | | | | remote station address |
| REM_IP_ADDR[1] | UInt | 192 | | | | | | remote station address |
| REM_IP_ADDR[2] | UInt | 168 | | | | | | remote station address |
| REM_IP_ADDR[3] | UInt | 100 | | | | | | remote station address |
| REM_IP_ADDR[4] | UInt | 1 | | | | | | remote station address |
| REM_PORT_NR | UInt | 32002 | | | | | | remote port number |
| RESERVED | Word | 16#0 | | | | | | unused; has to be 0 |

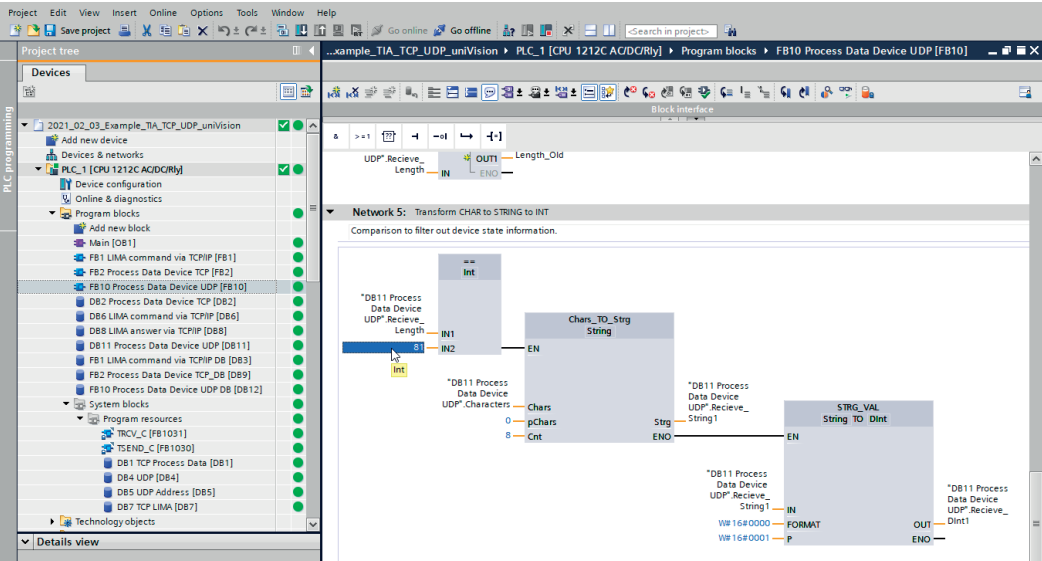
The TIA sample program is created for process data with a length of 81 characters. If a different character count is required, the sample program must be adapted accordingly.

NOTE!

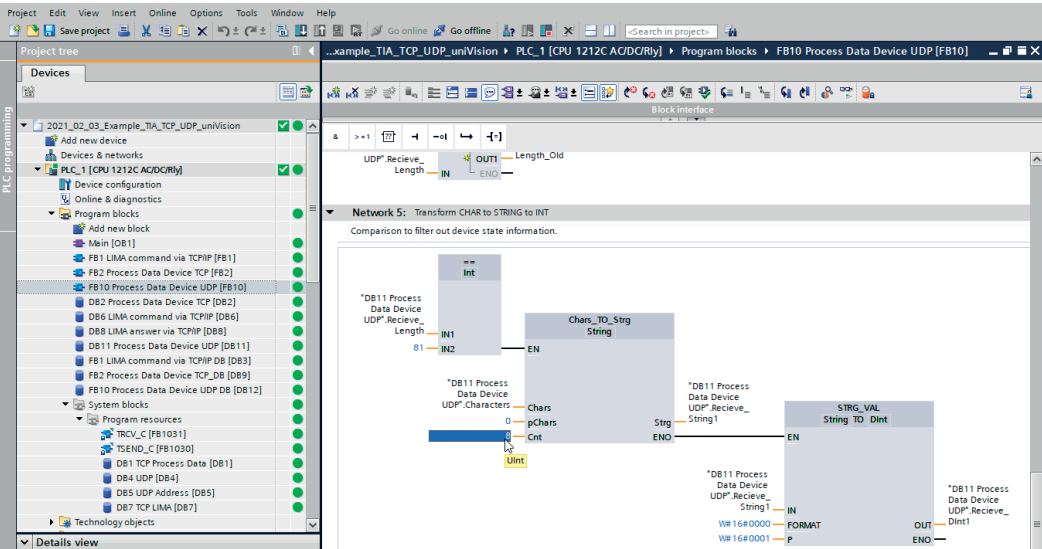


The total number of characters sent via UDP as process data can be determined in the uniVision software on the UDP device under “Output” (see section 3 “Settings in uniVision” on page 5). Preamble, separator and postamble as well as prefixes must be included in the character count!

To do this, adjust the character count under “IN2” in the function module “FB10 Process Data Device UDP” on network 5 “Transform CHAR to STRING to INT”.

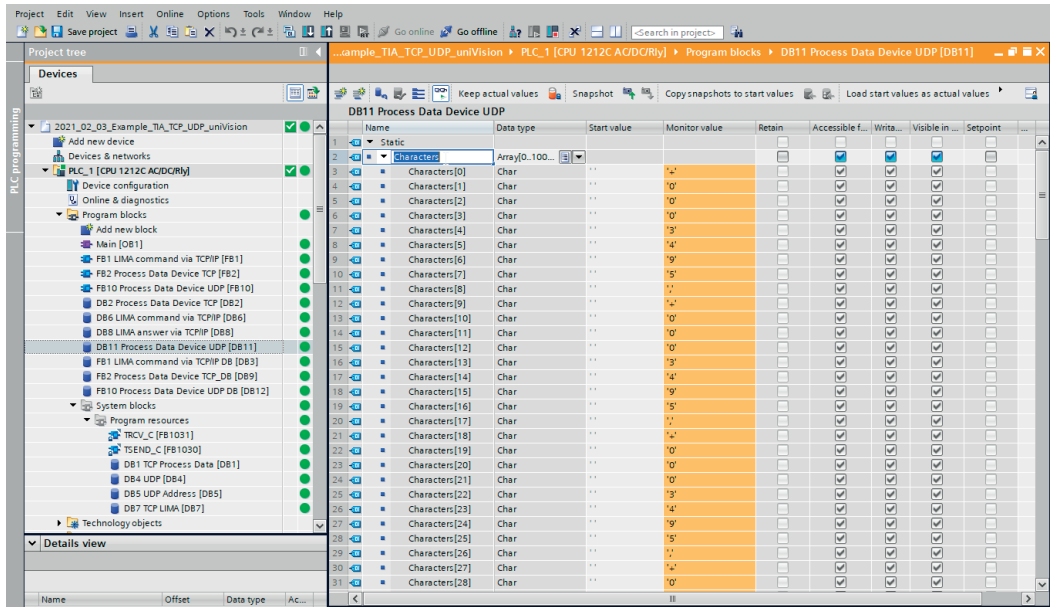


The sample program also includes direct conversion of the characters into an integer (DINT) for the first string. The number of characters or data type for the first integer can be changed as desired.



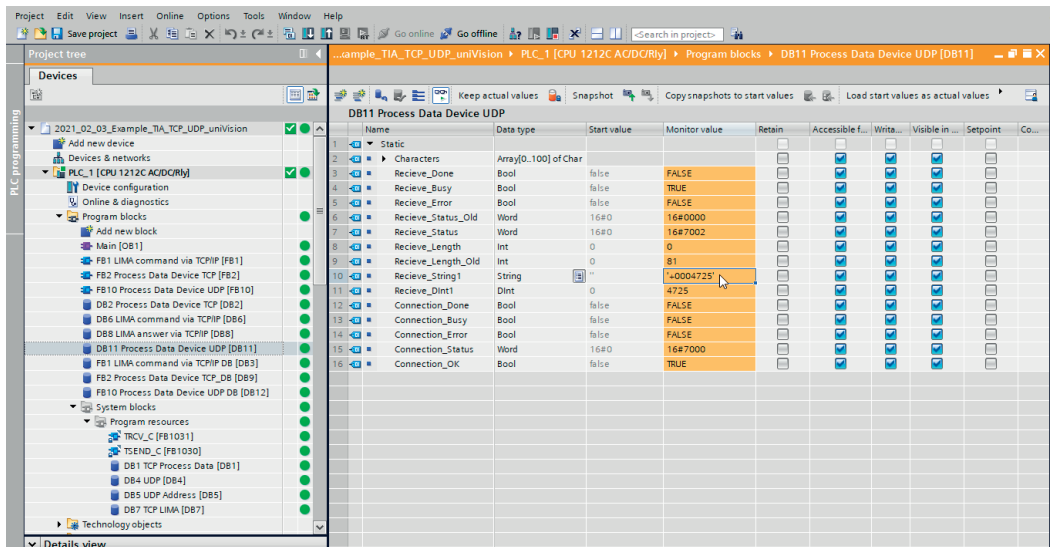
Compile the sample program, load it onto the control system and connect it online.

The “DB11 Process Data Device UDP” data block receives the process data sent by the UDP device. The data is received as individual characters (Char).



| Name | Data type | Start value | Monitor value | Retain | Accessible f... | Write... | Visible in ... | Setpoint |
|----------------|---------------|-------------|---------------|--------|-----------------|----------|----------------|----------|
| Static | Array[0..100] | | | | | | | |
| Characters[0] | Char | '' | 'A' | | | | | |
| Characters[1] | Char | '' | '0' | | | | | |
| Characters[2] | Char | '' | '0' | | | | | |
| Characters[3] | Char | '' | '0' | | | | | |
| Characters[4] | Char | '' | '3' | | | | | |
| Characters[5] | Char | '' | '4' | | | | | |
| Characters[6] | Char | '' | '9' | | | | | |
| Characters[7] | Char | '' | '5' | | | | | |
| Characters[8] | Char | '' | '7' | | | | | |
| Characters[9] | Char | '' | '2' | | | | | |
| Characters[10] | Char | '' | '2' | | | | | |
| Characters[11] | Char | '' | '0' | | | | | |
| Characters[12] | Char | '' | '0' | | | | | |
| Characters[13] | Char | '' | '3' | | | | | |
| Characters[14] | Char | '' | '4' | | | | | |
| Characters[15] | Char | '' | '9' | | | | | |
| Characters[16] | Char | '' | '5' | | | | | |
| Characters[17] | Char | '' | '' | | | | | |
| Characters[18] | Char | '' | 'A' | | | | | |
| Characters[19] | Char | '' | '0' | | | | | |
| Characters[20] | Char | '' | '0' | | | | | |
| Characters[21] | Char | '' | '0' | | | | | |
| Characters[22] | Char | '' | '3' | | | | | |
| Characters[23] | Char | '' | '2' | | | | | |
| Characters[24] | Char | '' | '9' | | | | | |
| Characters[25] | Char | '' | '5' | | | | | |
| Characters[26] | Char | '' | '' | | | | | |
| Characters[27] | Char | '' | 'A' | | | | | |
| Characters[28] | Char | '' | '0' | | | | | |

For the first string, conversion to another data type is shown on the DINT for example purposes.



| Name | Data type | Start value | Monitor value | Retain | Accessible f... | Write... | Visible in ... | Setpoint | Co... |
|--------------------|-----------------------|-------------|---------------|--------|-----------------|----------|----------------|----------|-------|
| Static | Array[0..100] of Char | | | | | | | | |
| Receive_Done | Bool | false | FALSE | | | | | | |
| Receive_Busy | Bool | false | TRUE | | | | | | |
| Receive_Error | Bool | false | FALSE | | | | | | |
| Receive_Status_Old | Word | 16#0 | 16#0000 | | | | | | |
| Receive_Status | Word | 16#0 | 16#7002 | | | | | | |
| Receive_Length | Int | 0 | 0 | | | | | | |
| Receive_Length_Old | Int | 0 | 81 | | | | | | |
| Receive_String1 | String | '' | '*0004725' | | | | | | |
| Receive_Dint1 | Dint | 0 | 4725 | | | | | | |
| Connection_Done | Bool | false | FALSE | | | | | | |
| Connection_Busy | Bool | false | FALSE | | | | | | |
| Connection_Error | Bool | false | FALSE | | | | | | |
| Connection_Status | Word | 16#0 | 16#7000 | | | | | | |
| Connection_OK | Bool | false | TRUE | | | | | | |

4.3 Sending LIMA Commands via TCP/IP and Receiving LIMA Answers

LIMA commands can be sent via the TCP/IP interface. In the sample program, a trigger command is sent to the uniVision product, which triggers an image or profile recording. Details on the commands available can be found in the LIMA interface protocol. It is available in the download area of the uniVision product detail page (<https://www.wenglor.com/product/DNNF020>).

The LIMA command must be entered with individual characters under “DB6 LIMA command via TCP/IP”.

<T/> must be sent for the trigger command.

The screenshot shows the Siemens TIA Portal software interface. On the left, the 'Project tree' displays the project structure for '2021_02_03_Example_TIA_TCP_uniVision'. The 'Program blocks' folder is expanded, showing various LIMA-related blocks. The 'DB6 LIMA command via TCP/IP' block is selected. The main window displays the 'DB6 LIMA command via TCP/IP' table, which is a table with 31 rows and 10 columns. The columns are: Name, Data type, Start value, Retain, Accessible from, Write, Visible in, Setpoint, and Comment. The table lists 31 characters (Characters[0] to Characters[30]) with their respective data types and settings.

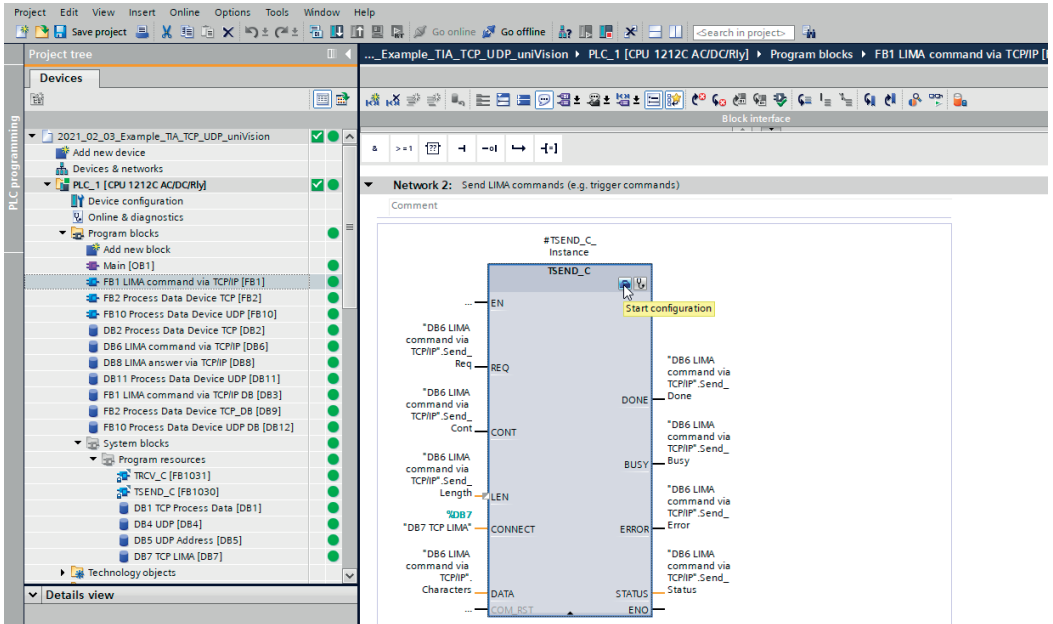
| Name | Data type | Start value | Retain | Accessible from | Write | Visible in | Setpoint | Comment |
|----------------|-----------------------|-------------|--------|-----------------|-------|------------|----------|---------|
| Static | | | | | | | | |
| Characters | Array[0..100] of Char | | | | | | | |
| Characters[0] | Char | | | | | | | |
| Characters[1] | Char | | | | | | | |
| Characters[2] | Char | | | | | | | |
| Characters[3] | Char | | | | | | | |
| Characters[4] | Char | | | | | | | |
| Characters[5] | Char | | | | | | | |
| Characters[6] | Char | | | | | | | |
| Characters[7] | Char | | | | | | | |
| Characters[8] | Char | | | | | | | |
| Characters[9] | Char | | | | | | | |
| Characters[10] | Char | | | | | | | |
| Characters[11] | Char | | | | | | | |
| Characters[12] | Char | | | | | | | |
| Characters[13] | Char | | | | | | | |
| Characters[14] | Char | | | | | | | |
| Characters[15] | Char | | | | | | | |
| Characters[16] | Char | | | | | | | |
| Characters[17] | Char | | | | | | | |
| Characters[18] | Char | | | | | | | |
| Characters[19] | Char | | | | | | | |
| Characters[20] | Char | | | | | | | |
| Characters[21] | Char | | | | | | | |
| Characters[22] | Char | | | | | | | |
| Characters[23] | Char | | | | | | | |
| Characters[24] | Char | | | | | | | |
| Characters[25] | Char | | | | | | | |
| Characters[26] | Char | | | | | | | |
| Characters[27] | Char | | | | | | | |
| Characters[28] | Char | | | | | | | |

The TIA sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
 - Subnet mask: 255.255.255.0
- LIMA commands are sent via port 32001.

If a different network setting is used on the uniVision product, the sample program must be adapted accordingly.

To do this, open the function module “FB1 LIMA command via TCP/IP” and click on “Start Configuration” on network 2 “Send LIMA commands (e.g. trigger commands)”.

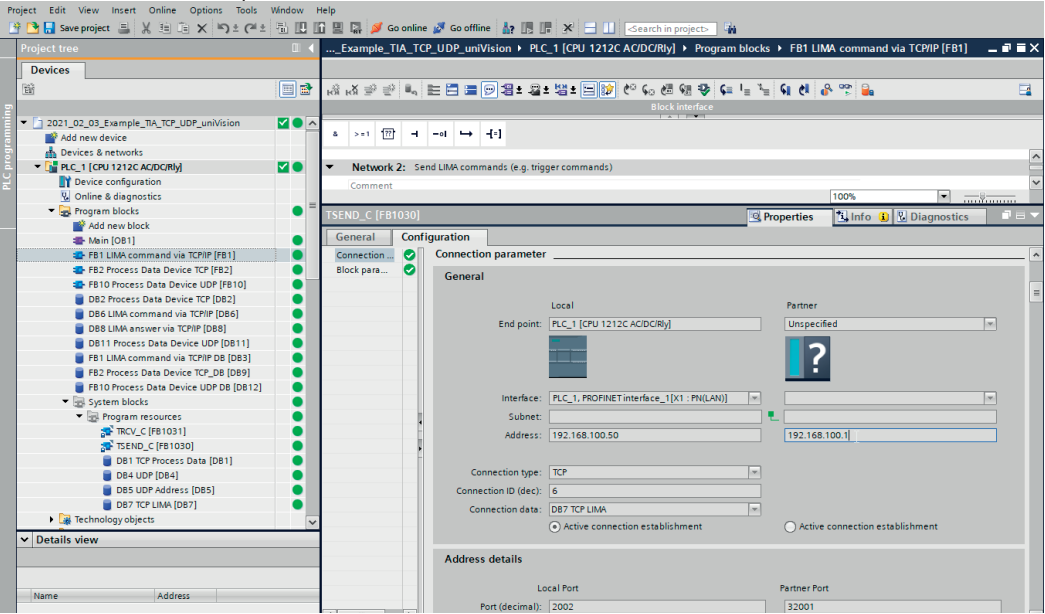


The screenshot displays the Siemens STEP 7 LAD editor interface. On the left, the 'Project tree' shows the project structure, including '2021_02_03_Example_TIA_TCP_UDP_uniVision' and 'PLC_1 [CPU 1212C AC/DC/Rly]'. The 'Program blocks' section is expanded, showing various function blocks, with 'FB1 LIMA command via TCP/IP [FB1]' selected. The main workspace shows 'Network 2: Send LIMA commands (e.g. trigger commands)'. The function block 'TSEND_C' is configured with the following parameters:

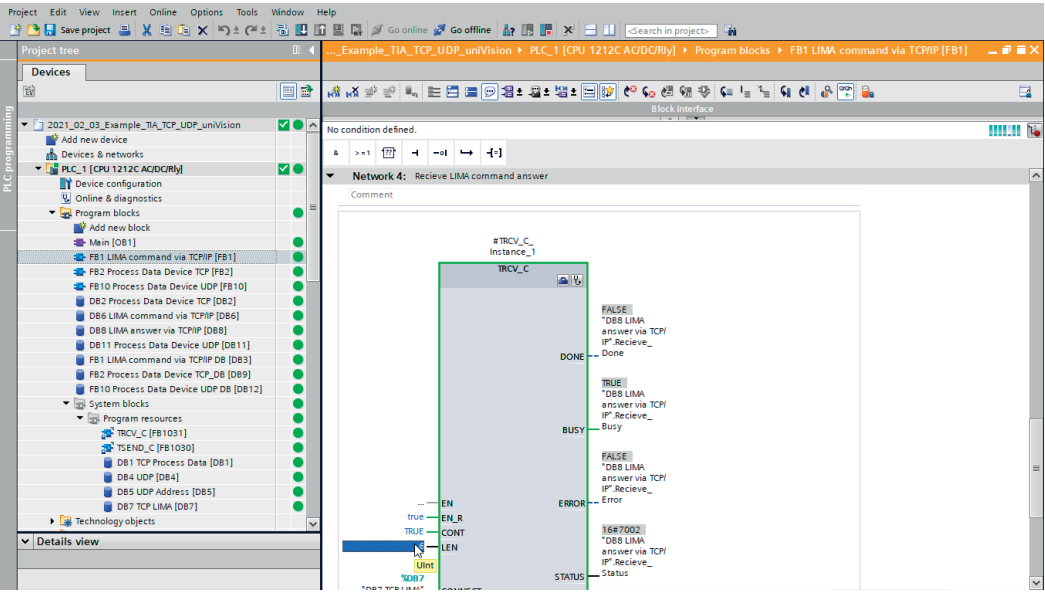
- EN**: EN (Enable)
- REQ**: "DB6 LIMA command via TCP/IP".Send_Req
- CONT**: "DB6 LIMA command via TCP/IP".Send_Cont
- LEN**: "DB7 TCP LIMA".Length
- CONNECT**: "DB7 TCP LIMA".CONNECT
- DATA**: "DB6 LIMA command via TCP/IP".Characters
- STATUS**: "DB6 LIMA command via TCP/IP".Status
- ERROR**: "DB6 LIMA command via TCP/IP".Error
- DONE**: "DB6 LIMA command via TCP/IP".Send_Done
- BUSY**: "DB6 LIMA command via TCP/IP".Send_Busy
- ERROR**: "DB6 LIMA command via TCP/IP".Error
- STATUS**: "DB6 LIMA command via TCP/IP".Status

A yellow box labeled 'Start configuration' is positioned near the top right of the function block, indicating the next step in the configuration process.

Enter the IP address and port 32001 under “Partner”.



Similarly, click on “Start Configuration” on network 4 “Receive LIMA command answer” and enter the IP address and port 32001 again. In addition, the character count of the LIMA answers must be entered on network 4 under “LEN”. The trigger command answer contains 6 characters (<TOK/>).



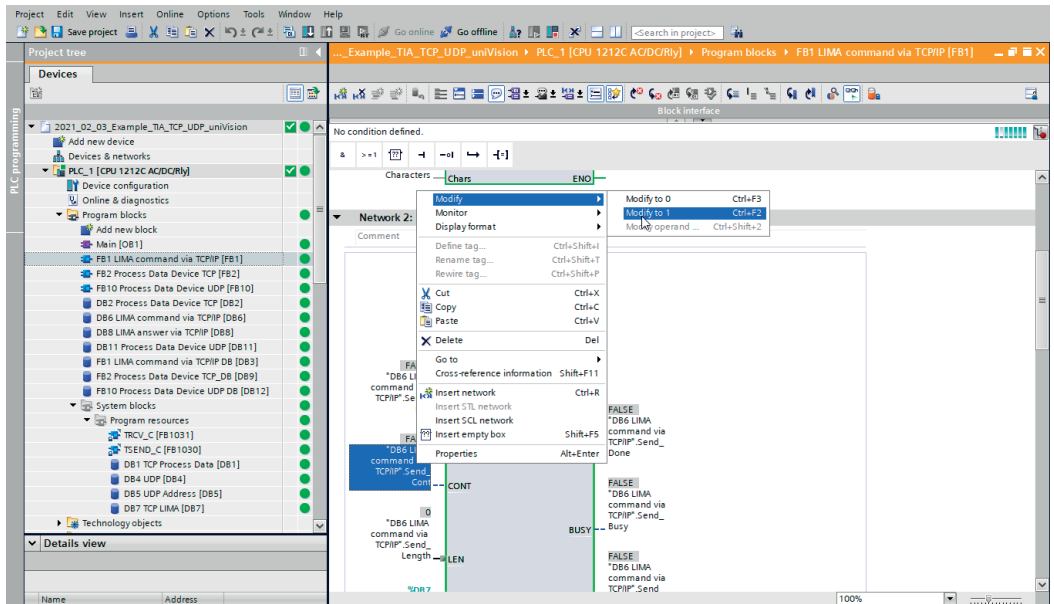
Compile the sample program, load it onto the control system and connect it online.

To send the LIMA command, first establish the connection to the uniVision product. To do this, open the function module “FB1 LIMA command via TCP/IP” and set “Send LIMA commands (e.g. trigger commands)” CONT to 1 on network 2.

NOTE!



The connection can only be established if port 32001 is available for the control system. Depending on the product or operating mode of the uniVision software, port 32001 is also required by the uniVision software (e.g. in editing mode). In this case, the uniVision software must disconnect so that the connection can be established via the control system.



The LIMA command is sent to the uniVision device by setting REQ to 1.

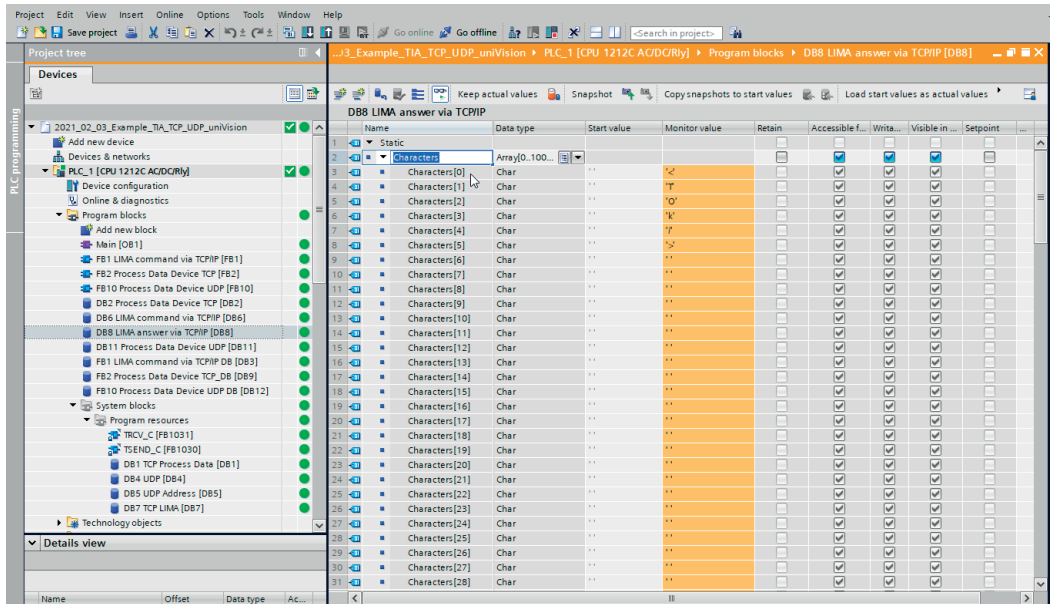
The screenshot displays the Siemens TIA Portal interface. On the left, the 'Project tree' shows a project named '2021_02_03_Example_TIA_TCP_udp_uniVision'. Under 'PLC programming', the 'Program blocks' section is expanded, showing a list of blocks including 'FB1 LIMA command via TCP/IP [FB1]'. The main workspace shows a ladder logic network for 'Network 2'. A context menu is open over the 'REQ' input of the 'FB1 LIMA' block, with the 'Apply to 1' option selected. The network diagram shows the 'REQ' input connected to a 'TRUE' constant, and the 'DONE' output connected to a 'FALSE' constant. The 'BUSY' output is also connected to a 'FALSE' constant. The 'LEN' input is connected to a '0' constant.

NOTE!



The LIMA command is reset in the sample program immediately after sending so that only one image or profile is taken from the uniVision product. The associated results for the trigger can be received via the process data. For example, the execution counter can be used to check when the results are available.

The LIMA answer can be received in the data block “DB8 LIMA answer via TCP/IP”. For the trigger command, <Tok/> is sent by the uniVision product in response to a successful execution of the trigger command.



The screenshot displays the Siemens SIMATIC Manager interface. The left pane shows the project tree for '2021_02_03_Example_TIA_UDP_uniVision'. The right pane shows the configuration for the data block 'DB8 LIMA answer via TCP/IP'. The configuration table lists 28 characters, each with a name, data type, start value, monitor value, and various control flags.

| Name | Data type | Start value | Monitor value | Retain | Accessible f... | Write... | Visible in ... | Setpoint |
|----------------|-----------------|-------------|---------------|--------|-----------------|----------|----------------|----------|
| Static | | | | | | | | |
| Characters | Array[0..100..] | | | | | | | |
| Characters[0] | Char | " | " | | | | | |
| Characters[1] | Char | " | " | | | | | |
| Characters[2] | Char | " | " | | | | | |
| Characters[3] | Char | " | " | | | | | |
| Characters[4] | Char | " | " | | | | | |
| Characters[5] | Char | " | " | | | | | |
| Characters[6] | Char | " | " | | | | | |
| Characters[7] | Char | " | " | | | | | |
| Characters[8] | Char | " | " | | | | | |
| Characters[9] | Char | " | " | | | | | |
| Characters[10] | Char | " | " | | | | | |
| Characters[11] | Char | " | " | | | | | |
| Characters[12] | Char | " | " | | | | | |
| Characters[13] | Char | " | " | | | | | |
| Characters[14] | Char | " | " | | | | | |
| Characters[15] | Char | " | " | | | | | |
| Characters[16] | Char | " | " | | | | | |
| Characters[17] | Char | " | " | | | | | |
| Characters[18] | Char | " | " | | | | | |
| Characters[19] | Char | " | " | | | | | |
| Characters[20] | Char | " | " | | | | | |
| Characters[21] | Char | " | " | | | | | |
| Characters[22] | Char | " | " | | | | | |
| Characters[23] | Char | " | " | | | | | |
| Characters[24] | Char | " | " | | | | | |
| Characters[25] | Char | " | " | | | | | |
| Characters[26] | Char | " | " | | | | | |
| Characters[27] | Char | " | " | | | | | |
| Characters[28] | Char | " | " | | | | | |

5. TwinCAT3 Sample Programs

The TwinCAT3 sample programs for UDP and TCP include the following use cases:

- Receiving process data from the TCP device (in the TCP sample program)
- Receiving process data from the UDP device (in the UDP sample program)
- Sending LIMA commands (e.g. trigger commands) via TCP/IP and receiving LIMA response (in the TCP sample program)

In the example, the following network configuration is used:

- PC with TwinCAT3:
 - IP address: 192.168.100.181
 - Subnet mask: 255.255.255.0
- uniVision product:
 - IP address: 192.168.100.1
 - Subnet mask: 255.255.255.0



NOTE!

To do this, the latest TwinCAT3 version must be installed, including the TF6310 TC3 TCP/IP module. For details, please contact Beckhoff support.

5.1 Receiving Process Data from TCP Device

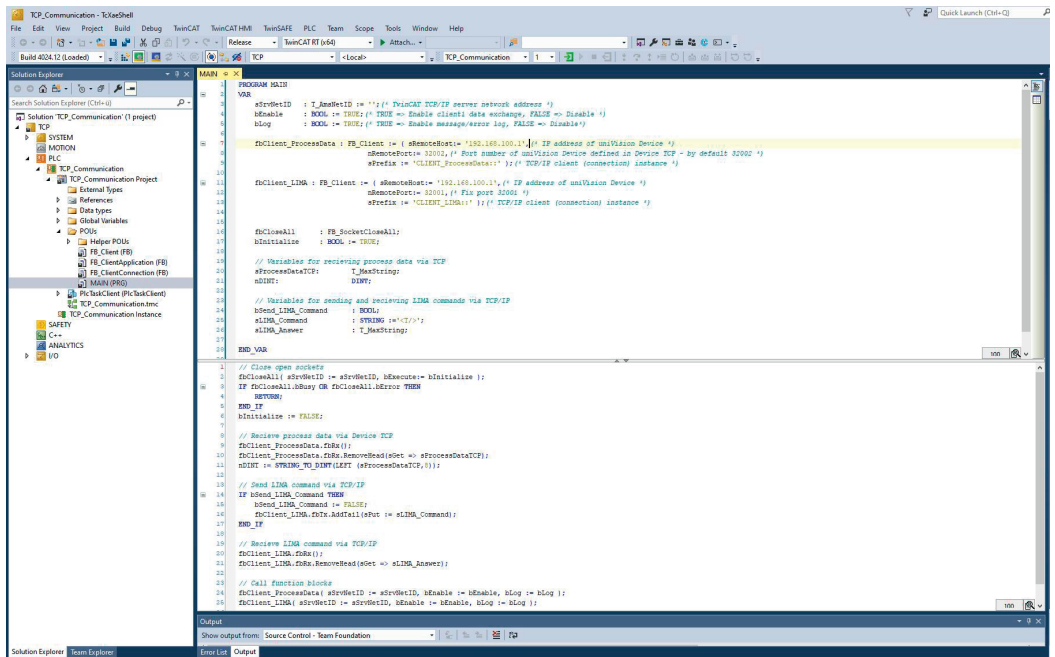
The sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
- Subnet mask: 255.255.255.0

The TCP process data is sent via port 32002 by default.

If a different network setting or another port is used on the uniVision product, the sample program must be adapted accordingly.

To do this, enter the IP address of the uniVision product under “sRemoteHost” and the port under “nRemotePort” in the MAIN of TCP_Communication under fbClient_ProcessData.



```

PROGRAM MAIN
VAR
  sRemoteID : T_AnnetID := ''; (* TwinCAT TCP/IP server network address *)
  bEnable   : BOOL := TRUE; (* TRUE => Enable client's data exchange, FALSE => Disable *)
  bLog      : BOOL := TRUE; (* TRUE => Enable message/error log, FALSE => Disable *)

  fbClient_ProcessData : FB_Client := ( sRemoteHost := '192.168.100.1', (* IP address of uniVision Device *)
                                         sRemotePort := 32002, (* Port number of uniVision Device defined in Device TCP - by default 32002 *)
                                         sPrefix := 'CLIENT_ProcessData:' ); (* TCP/IP client (connection) instance *)

  fbClient_LINQ : FB_Client := ( sRemoteHost := '192.168.100.1', (* IP address of uniVision Device *)
                                sRemotePort := 32001, (* File port 32001 *)
                                sPrefix := 'CLIENT_LINQ:' ); (* TCP/IP client (connection) instance *)

  fbCloseAll : FB_SocketCloseAll;
  Initialize : BOOL := TRUE;

  // Variables for receiving process data via TCP
  sProcessDataTCP : T_MessString;
  sMSG :
  sMSG_Command :
  sMSG_Answer : T_MessString;

  // Variables for sending and receiving LINQ commands via TCP/IP
  sMSG_Command :
  sMSG_Answer : T_MessString;

END VAR

// Close open sockets
fbCloseAll( sRemoteID := sRemoteID, bResource := Initialize );
IF fbCloseAll.bBusy OR fbCloseAll.bError THEN
  RETURN;
END IF
Initialize := FALSE;

// Receive process data via Device TCP
fbClient_ProcessData.sData.ReceiveHead(sData := sProcessDataTCP);
nMSG := sMSGING_WU_SIZE(LEFT( sProcessDataTCP,1));

// Send LINQ command via TCP/IP
IF sMSG_Command THEN
  fbClient_LINQ.sData.ReceiveHead(sData := sMSG_Answer);
END IF

// Receive LINQ command via TCP/IP
fbClient_LINQ.sData.ReceiveHead(sData := sMSG_Answer);

// Call function blocks
fbClient_ProcessData( sRemoteID := sRemoteID, bEnable := bEnable, bLog := bLog );
fbClient_LINQ( sRemoteID := sRemoteID, bEnable := bEnable, bLog := bLog );
  
```

The sample program also includes direct conversion of the first eight characters into an integer (DINT) for the first string. The number of characters or data type can be changed as desired.

Enable the sample program, log in and start it. The process data sent by the TCP device appears under the variables "sProcessDataTCP". The data for the first DINT appears under "nDINT".

| Expression | Type | Value | Prepared value | Address | Comment |
|---------------------|----------------|---|----------------|---------|---|
| sServID | T_AnsNetID | - | | | TwinCAT TCP/IP server network address |
| bEnable | BOOL | TRUE | | | TRUE => Enable client, data exchange, FALSE... TRUE => Enable mess. error log, FALSE =>... |
| bClient_ProcessData | PLC_Client | | | | TCP/IP client (connection) instance |
| bClient_LINA | PLC_Client | | | | TCP/IP client (connection) instance |
| bClientLose | PLC_ClientLose | | | | |
| bInitialize | BOOL | FALSE | | | |
| sProcessDataTCP | T_MasterString | +0594672,+051437L,+0594672,+0594672,+0594672,+0594672,+0594672,+0594672 | | | Variables for receiving process data via TCP |
| nDINT | DINT | 0x0000 | | | |
| bSend_LINA_Command | BOOL | FALSE | | | Variables for sending ... receiving LINA com... |
| sLINA_Command | STRING | <TCP> | | | |
| sLINA_Answer | T_MasterString | - | | | |

```

1 // Close open sockets
2 if bCloseAll() sServID := sServID; bExecute := bInitialize;
3 if bCloseAll() bEnable := FALSE; bCloseAll() bError := TRUE;
4 RETURN;
5 END_IF
6 bInitialize := FALSE;
7
8 // Receive process data via Device TCP
9 bClient_ProcessData.RemoveHead(serv := sProcessDataTCP(0594672));
10 bClient_ProcessData.RemoveHead(serv := sProcessDataTCP(0594672));
11 bDINT := bDINT := sDINT(0594672, 0594672, 0594672, 0594672, 0594672, 0594672, 0594672, 0594672);
12
13 // Send LINA command via TCP/IP
14 if bSend_LINA_Command = TRUE THEN
15     bSend_LINA_Command := FALSE;
16     bClient_LINA.RemoveHead(serv := sLINA_Command(0594672));
17 END_IF
18
19 // Receive LINA command via TCP/IP
20 bClient_LINA.RemoveHead(serv := sLINA_Answer(0594672));
21 bClient_LINA.RemoveHead(serv := sLINA_Answer(0594672));
22
23 // Call function blocks
24 bClient_ProcessData( sServID := sServID, sMaster := sMaster, sData := sData, sData := sData, sData := sData );
25 bClient_LINA( sServID := sServID, sMaster := sMaster, sData := sData, sData := sData, sData := sData );
26 RETURN;
    
```

5.2 Receiving Process Data from UDP Device

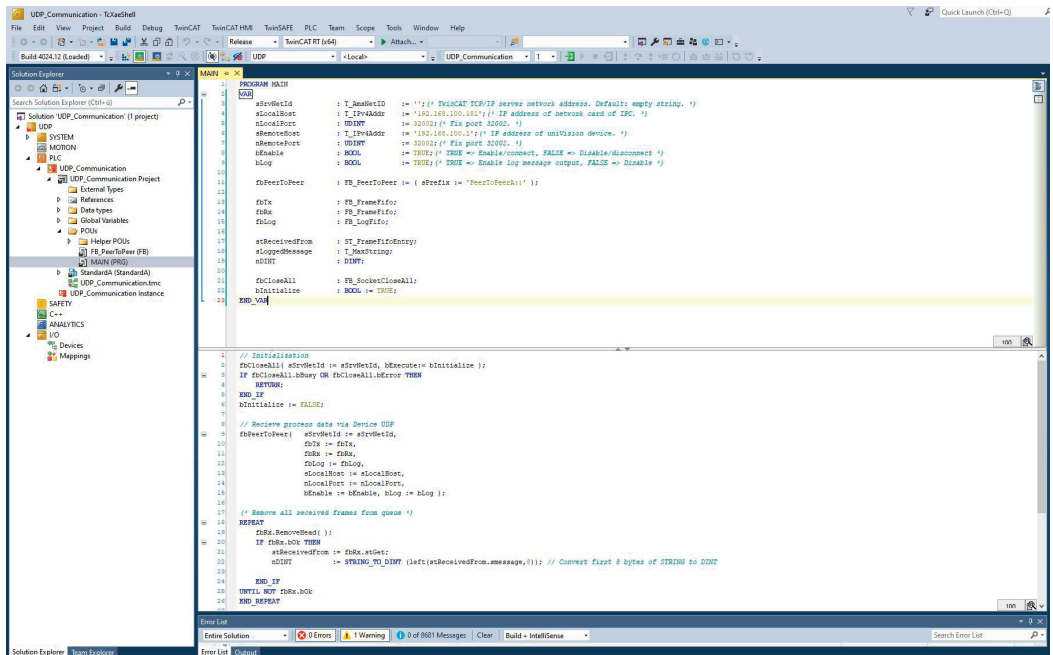
The sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
- Subnet mask: 255.255.255.0

UDP process data is sent via port 32002.

If a different network setting is used on the uniVision product, the sample program must be adapted accordingly.

To do this, enter the IP address of the uniVision product in the MAIN of UDP_Communication under the variables "sRemoteHost".



```

PROGRAM MAIN
END

: T_AnalogID := '' (* TrinCAT TCP/IP server network address. Default: empty string. *)
: T_IPV4Addr := '192.168.100.11' (* IP address of network card of PLC. *)
: T_PORT := 32002 (* Port 32002. *)
: sRemoteHost := '192.168.100.1' (* IP address of uniVision device. *)
: sRemotePort := 32002 (* File port 32002. *)
: bConnect := TRUE (* TRUE => Enable connect, FALSE => Disable/disconnect *)
: bLog := TRUE (* TRUE => Enable log message output, FALSE => Disable *)

fbPeerToPeer := FB_PeerToPeer := ( sPrefix := 'PeerToPeer!' );

fbTx := FB_FrameFifo;
fbRx := FB_FrameFifo;
fbLog := FB_LogFifo;

sReceivedFrom := ST_FrameFifoEntry;
sLogMsg := T_String;
sDINT := DINT;

fbCloseAll := FB_SocketCloseAll;
bInitialize := bLog := TRUE;

END_MAIN

// Initialization
fbCloseAll sReceivedFrom := sReceivedFrom, bConnect := bInitialize;
IF fbCloseAll.bBusy ON fbCloseAll.bError THEN
  RETURN;
END_IF
bInitialize := FALSE;

// Receive process data via Device UDP
fbPeerToPeer sReceivedFrom := sReceivedFrom,
fbTx := fbTx,
fbRx := fbRx,
fbLog := fbLog,
sLocalHost := sLocalHost,
sLocalPort := sLocalPort,
bEnable := bEnable, bLog := bLog;

(* Show all received frames from queue *)
REPEAT
  fbRx.RemoveRead();
  IF fbRx.bOk THEN
    sReceivedFrom := fbRx.sGet;
    sDINT := STRING_TO_DINT (left(sReceivedFrom, message, 1)); // Convert first 8 bytes of STRING to DINT
  END_IF
UNTIL NOT fbRx.bOk
END_REPEAT
  
```

The sample program also includes direct conversion of the first eight characters into an integer (DINT) for the first string. The number of characters or data type can be changed as desired.

Enable the sample program, log in and start it. The process data sent by the UDP device appears under the variables “stReceivedFrom” -> “sMessage”. The data for the first DINT appears under “nDINT”.

UDP_Communication - TcKadShell

File

Edit

View

Project

Build

Debug

TwinCAT

TwinCAT HW

TwinSAFE

PLC

Team

Scope

Tools

Window

Help

Build 4024.12 (Lower6)

UDP_Communication

Attach...

Solution Explorer (Ctrl+U)

Solution UDP_Communication (1 project)

SYSTEM

MOTION

PLC

UDP_Communication

UDP_Communication Project

External Types

References

Data types

Global Variables

POUs

Helper POU:

FB_PeerToPeer (FB)

MAIN (FNC)

Standard (Standard)

UDP_CommunicationIntc

UDP_Communication Instance

SAFETY

C++

ANALYTICS

I/O

Devices

Mappings

MAIN (FNC)

UDP_Communication MAIN

| Expression | Type | Value | Prepared value | Address | Comment |
|----------------|-------------------|--|----------------|---------|---|
| stServId | T_AnsiStrID | - | | | TwinCAT TCP/IP serve. network address, Def... |
| sLocalHost | T_IPv4Addr | 192.168.100.187 | | | IP address of network card of PC. |
| sLocalPort | UDINT | 32002 | | | Fix port 32002 |
| sRemoteHost | T_IPv4Addr | 192.168.100.1 | | | IP address of un/vision device. |
| sRemotePort | UDINT | 32002 | | | Fix port 32002. |
| sEnable | BOOL | TRUE | | | TRUE => Enable listen. FALSE => Disable... |
| sLog | BOOL | TRUE | | | TRUE => Enable log. age output, FALSE ... |
| fbPeerToPeer | FB_PeerToPeer | | | | |
| fbTx | FB_Transmit | | | | |
| fbRx | FB_Transmit | | | | |
| fbLog | FB_LogInfo | | | | |
| stReceivedFrom | ST_FrameHeader | | | | Remote address, 80m. obtaining an (IPv4) L... |
| sRemoteAddr | ST_SockAddr | | | | |
| sMessage | T_MexStrng | "0256095,0256095,0256095,0256095,0256095,0256095,0256095,0256095," | | | |
| sLoggedMessage | T_MexStrng | PeerToPeer::Connected (UDP) socket created: 192.168.100.187:32002 | | | UDP packet data |
| nDINT | DINT | 256095 | | | |
| fbCloseAll | FB_SocketCloseAll | | | | |
| bInitialize | BOOL | FALSE | | | |

```

1 // Initialization
2 fbCloseAll stServId := stServId; bExecute := bInitialize := FALSE;
3 IF fbCloseAll.bError THEN
4     RETURN;
5 END_IF
6 bInitialize := FALSE;
7
8 // Receive process data via Device UDP
9 fbPeerToPeer( stServId := stServId,
10    stData := stData,
11    stData := stData,
12    stLog := stLog,
13    sLocalHost := "192.168.100.1",
14    sLocalPort := 32002,
15    sRemoteHost := "192.168.100.1",
16    sRemotePort := 32002 );
17
18 // Receive all received frames from queue
19 REPEAT
20     fbRx.RemoveHead();
21     IF fbRx.bError THEN
22         stReceivedFrom := stData.stData;
23         nDINT := 256095;
24         nDINT := nDINT * 10;
25     END_IF
26 UNTIL NOT fbRx.bError
27 END_REPEAT

```

Output

Show output from: Source Control - Team Foundation

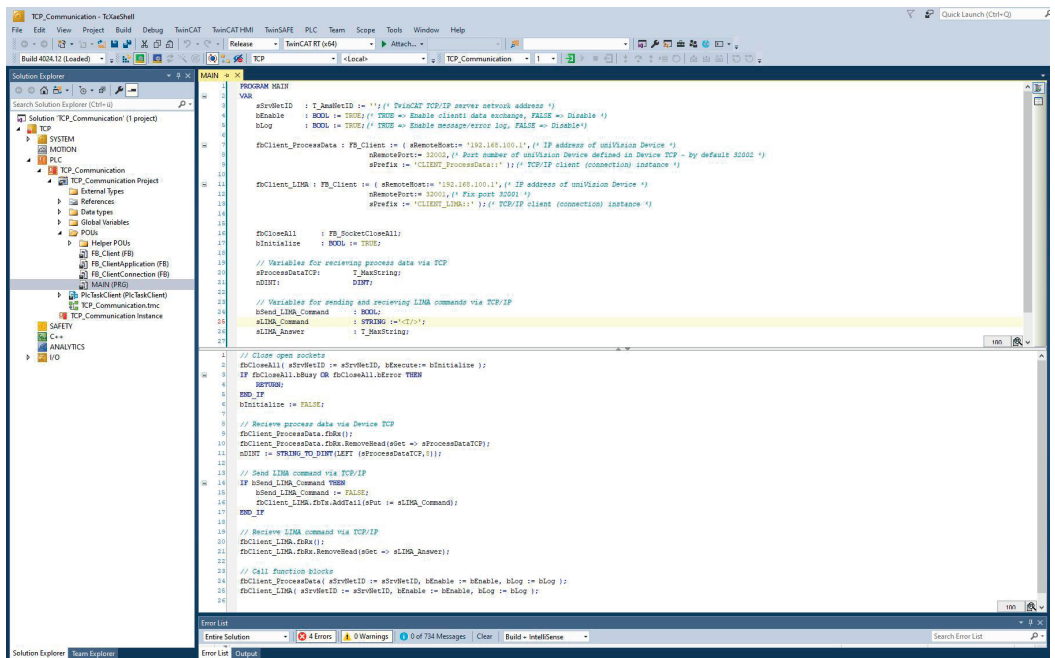
Error List

Output

5.3 Sending LIMA Commands via TCP/IP and Receiving LIMA Answers

LIMA commands can be sent via the TCP/IP interface. In the sample program, a trigger command is sent to the uniVision product, which triggers an image or profile recording. Details on the commands available can be found in the LIMA interface protocol. It is available in the download area of the uniVision product detail page (<https://www.wenglor.com/product/DNNF020>).

The LIMA command must be entered in the MAIN of TCP_Communication under "sLIMA_Command". <T/> must be sent for the trigger command.



```

// PROLOGUE
VAR
  sDeviceID : STRING := ''; (* PrintCAT TCP/IP server network address *)
  bEnable : BOOL := TRUE; (* TRUE => Enable client's data exchange, FALSE => Disable *)
  bLog : BOOL := TRUE; (* TRUE => Enable message/error log, FALSE => Disable *)

  sClient_ProcessData : FB_Client := ( sRemoteHost: '192.168.100.1', (* IP address of uniVision Device *)
    sRemotePort: 32002, (* Port number of uniVision Device defined in Device TCP - by default 32002 *)
    sPrefix := "CLIENT_ProcessData:" ); (* TCP/IP client (connection) instance *)

  sClient_LIMA : FB_Client := ( sRemoteHost: '192.168.100.1', (* IP address of uniVision Device *)
    sRemotePort: 32001, (* Port 32001 *)
    sPrefix := "CLIENT_LIMA:" ); (* TCP/IP client (connection) instance *)

  sCloseAll : FB_SocketCloseAll;
  bInitialize : BOOL := TRUE;

  // Variables for receiving process data via TCP
  sProcessDataTCP : T_MacString;
  sDINT : DINT;

  // Variables for sending and receiving LIMA commands via TCP/IP
  sSend_LIMA_Command : BOOL;
  sLIMA_Command : STRING := '<T/>';
  sLIMA_Answer : T_MacString;

  // Close open sockets
  sCloseAll( sDeviceID := sDeviceID, bEnable := bInitialize );
  IF sCloseAll.bError THEN
    bERROR;
  END_IF
  bInitialize := FALSE;

  // Receive process data via Device TCP
  sClient_ProcessData.sData.RemoveHead(sData := sProcessDataTCP);
  sDINT := STRING_TO_DINT(LEFT(sProcessDataTCP,1));

  // Send LIMA command via TCP/IP
  IF sSend_LIMA_Command THEN
    sSend_LIMA_Command := FALSE;
    sClient_LIMA.sData.AddTail(sPut := sLIMA_Command);
  END_IF

  // Receive LIMA command via TCP/IP
  sClient_LIMA.sData();
  sClient_LIMA.sData.RemoveHead(sData := sLIMA_Answer);

  // Call function blocks
  sClient_ProcessData( sDeviceID := sDeviceID, bEnable := bEnable, bLog := bLog );
  sClient_LIMA( sDeviceID := sDeviceID, bEnable := bEnable, bLog := bLog );
  
```

The sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
- Subnet mask: 255.255.255.0

LIMA commands are sent via port 32001.

If a different network setting is used on the uniVision product, the sample program must be adapted accordingly.

To do this, enter the IP address of the uniVision product under “sRemoteHost” in fbClient_LIMA.

Enable the sample program, log in and start it.

NOTE!



Connection from the control unit to the uniVision product can only be established if port 32001 is available for the control system. Depending on the product or operating mode of the uniVision software, port 32001 is also required by the uniVision software (e.g. in editing mode). In this case, the uniVision software must disconnect so that the connection can be established via the control system.

The LIMA command is sent to the uniVision product by setting “bSend_LIMA_Command” to TRUE. The command may only be sent once, not sent constantly, so that only one image or profile is recorded. A new command must not be sent until the LIMA answer to the previous command has been received.

The LIMA answer is contained in “sLIMA Answer”. For the trigger command, <Tok/> is sent by the uniVision product in response to a successful execution of the trigger command. In addition, after data recording and evaluation, the new process data is also available via TCP under “sProcessDataTCP”. The execution counter can be used, for example, to check when new results are available.

[illegible]

6. Rockwell Sample Programs

The Rockwell sample programs for process data and LIMA include the following application cases:

- Receiving process data from the TCP device (in the sample program Example_Rockwell_ProcessData.ACD)
- Receiving process data from the UDP device (in the sample program Example_Rockwell_ProcessData.ACD)
- Sending LIMA commands (e.g., trigger commands) via TCP/IP and receiving the LIMA response (in the sample program Example_Rockwell_LIMA.ACD)

In the example, the following network configuration is used:

- PLC:
 - IP address: 192.168.100.70
 - Subnet mask: 255.255.255.0
- uniVision product:
 - IP address: 192.168.100.1
 - Subnet mask: 255.255.255.0



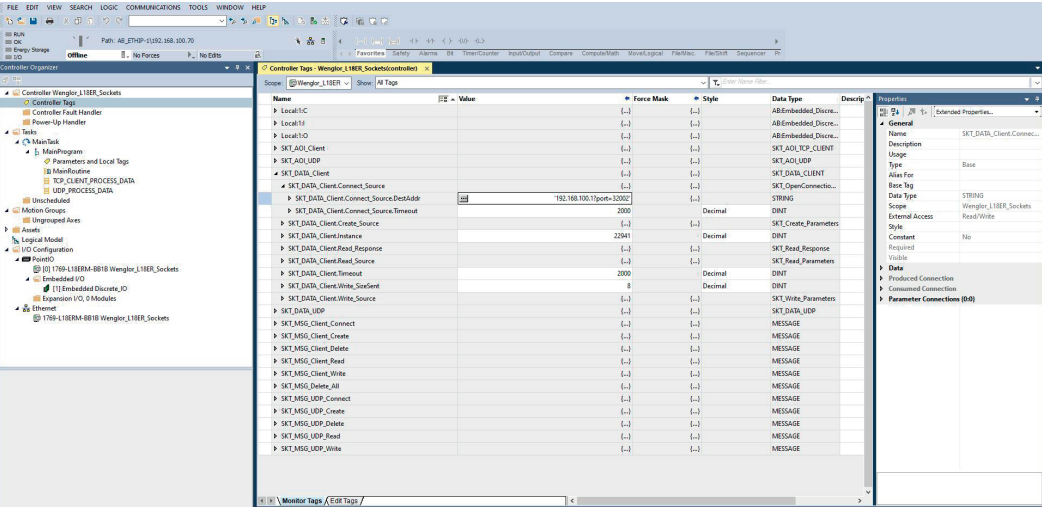
NOTE!
The sample program is created with an Allen-Bradley 1769-L18ERM-BB1B PLC using Studio 5000 Logix Designer V32.

6.1 Receiving Process Data from the TCP Device

The sample program Example_Rockwell_ProcessData.ACD is created with the following client network setting for the uniVision product:

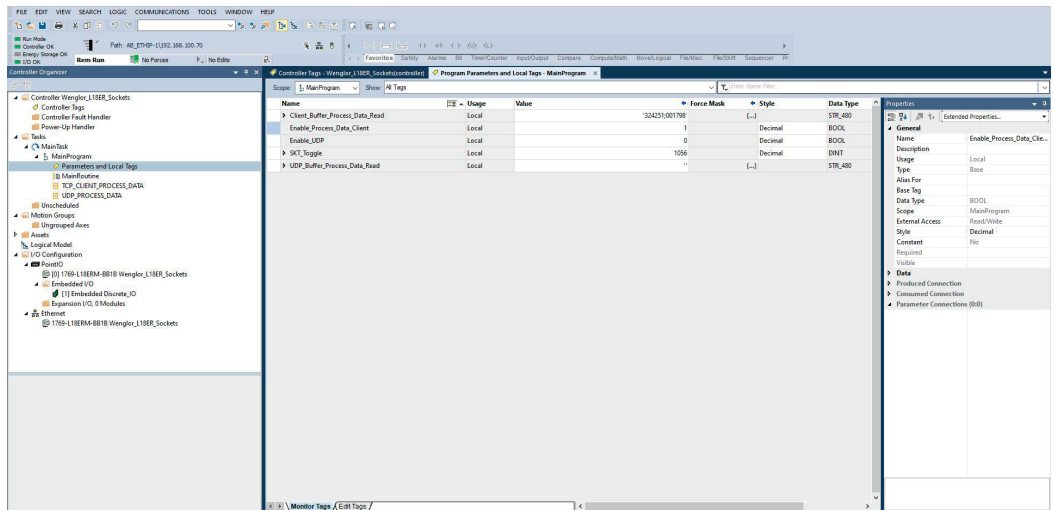
- IP address: 192.168.100.1
- Subnet mask: 255.255.255.0

The TCP process data are sent via port 32002 by default.



Transfer the sample program to the controller and go online.

The TCP connection is established by activating the value Enable_Process_Data_Client under Parameters and Local Tags. The process data sent by the TCP device appear under the Client_Buffer_Process_Data_Read.



| Name | Usage | Value | Force Mask | Style | Data Type |
|---------------------------------|-------|--------------|------------|---------|-----------|
| Client_Buffer_Process_Data_Read | Local | 134251001708 | [...] | | STR_480 |
| Enable_Process_Data_Client | Local | 1 | | Decimal | BOOL |
| Enable_UDP | Local | 0 | | Decimal | BOOL |
| SAT_Toggle | Local | 1056 | | Decimal | DINT |
| UDP_Buffer_Process_Data_Read | Local | -1 | | | STR_480 |

Properties

General

Name: Enable_Process_Data_Client

Description: Local

Usage: Base

Type: Local

Alias For: Base

Base Tag: Base

Data Type: BOOL

Scope: MainProgram

External Access: Read/Write

Style: Decimal

Constant: No

Inquired: No

Visible: No

Data

Produced Connection: Connected Connection

Consumed Connection: Connected Connection

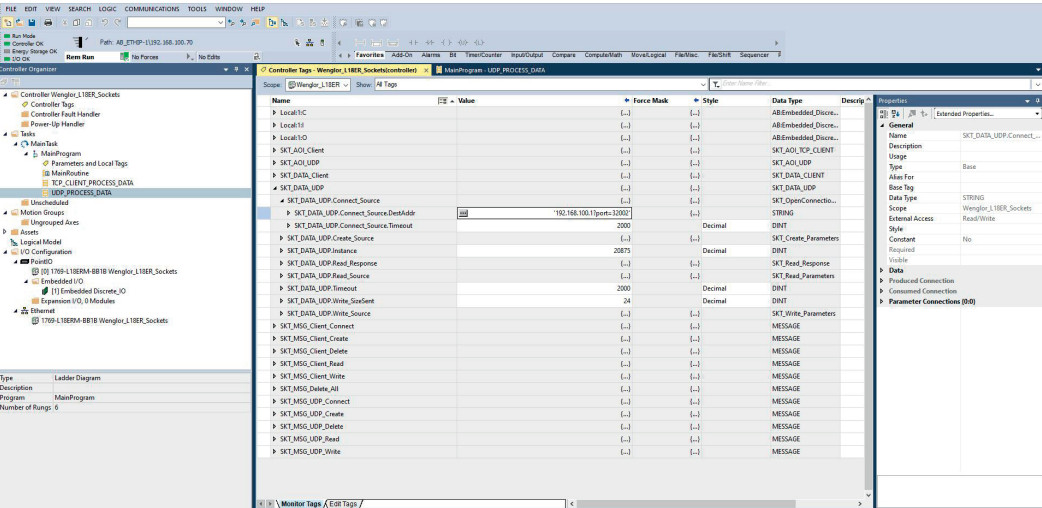
Parameter Connections (0/0)

6.2 Receiving Process Data from the UDP Device

The sample program Example_Rockwell_ProcessData.ACD is created with the following network setting for the uniVision product:

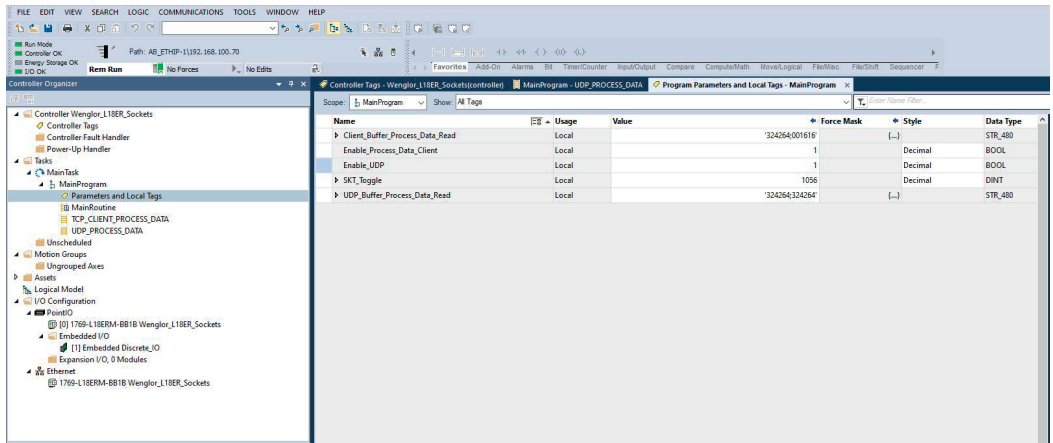
- IP address: 192.168.100.1
- Subnet mask: 255.255.255.0

The UDP process data are sent via port 32002.
If a different network setting is used on the uniVision product, the sample program must be adapted accordingly. To do so, open the controller tags and enter the IP address under SKT_DATA_UDP.Connect_Source.DestAddr.



Transfer the sample program to the controller and go online.

To receive the UDP process data, activate the value Enable_UDP under Parameters and Local Tags. The process data sent by the UDP device appear under UDP_Buffer_Process_Data_Read.



The screenshot shows the Wenglor software interface. On the left is the 'Controller Organizer' tree, and on the right is the 'MainProgram - UDP_PROCESS_DATA' window displaying a table of parameters and local tags.

Controller Organizer Tree:

- Controller Wenglor_L1BER_Sockets
 - Controller Tags
 - Controller Fault Handler
 - Power-Up Handler
 - Tasks
 - MainTask
 - MainProgram
 - Parameters and Local Tags
 - MainRoutine
 - TCP_CLIENT_PROCESS_DATA
 - UDP_PROCESS_DATA
 - Unscheduled
 - Motion Groups
 - Ungrouped Axes
 - Assets
 - Logical Model
 - I/O Configuration
 - PrintIO
 - (0) 1769-L1BERM-8B1B Wenglor_L1BER_Sockets
 - Embedded I/O
 - (1) Embedded Discrete_IO
 - Expansion I/O, 0 Modules
 - Ethernet
 - 1769-L1BERM-8B1B Wenglor_L1BER_Sockets

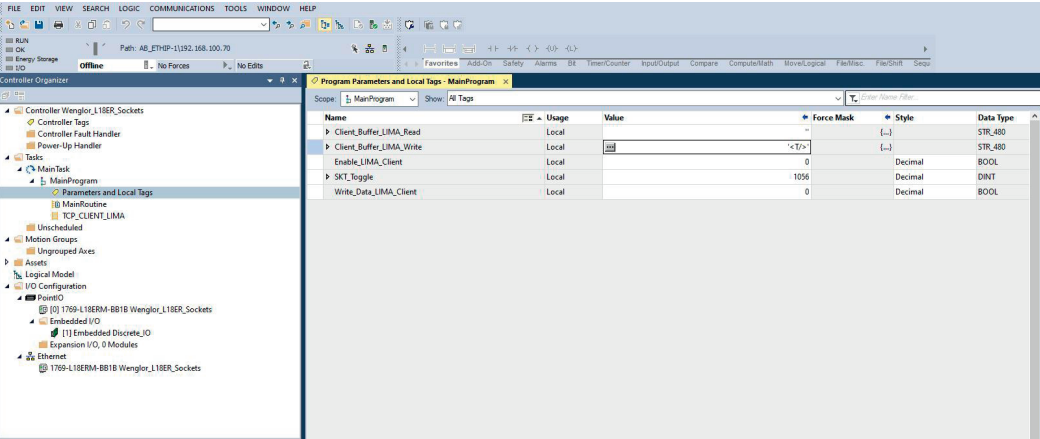
MainProgram - UDP_PROCESS_DATA Table:

| Name | Usage | Value | Force Mask | Style | Data Type |
|---------------------------------|-------|-----------------|------------|---------|-----------|
| Client_Buffer_Process_Data_Read | Local | '324264:001616' | | (...) | STR_480 |
| Enable_Process_Data_Client | Local | | 1 | Decimal | BOOL |
| Enable_UDP | Local | | 1 | Decimal | BOOL |
| SKT_Toggle | Local | 1056 | | Decimal | DINT |
| UDP_Buffer_Process_Data_Read | Local | '324264:324264' | | (...) | STR_480 |

6.3 Sending LIMA Commands via TCP/IP and Receiving LIMA Responses

LIMA commands can be sent via the TCP/IP interface. In the sample program Example_Rockwell_LIMA.ACD, a trigger command is sent to the uniVision product, which triggers an image or profile recording. Details on the commands available can be found in the LIMA interface protocol. It is available in the download area of the uniVision product detail page (<https://www.wenglor.com/product/DNNF020>).

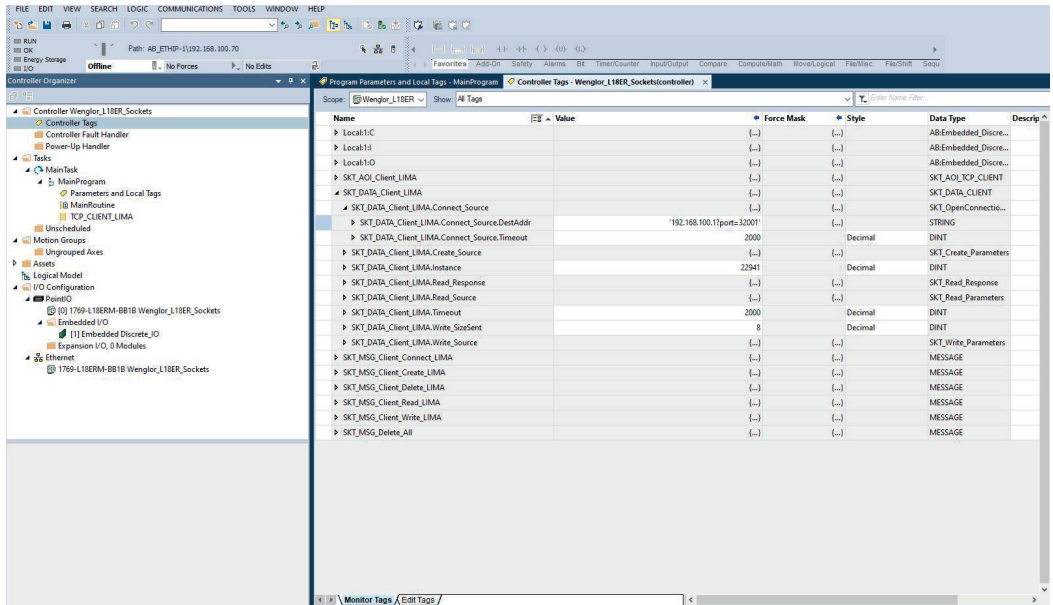
The LIMA command must be entered under Client_Buffer_LIMA_Write under Parameters and Local Tags. <T/> must be sent for the trigger command.



The sample program is created with the following network setting for the uniVision product:

- IP address: 192.168.100.1
 - Subnet mask: 255.255.255.0
- LIMA commands are sent via port 32001.

If a different network setting is used on the uniVision product, the sample program must be adapted accordingly. To do so, open the controller tags and enter the IP address under SKT_DATA_Client_LIMA.Connect_Source.DestAddr.



| Name | Force Mask | Style | Data Type | Description |
|--|------------|-------|----------------------|-------------|
| Local:I-C | (--) | (--) | ABEmbedded_Disc... | |
| Local:I-I | (--) | (--) | ABEmbedded_Disc... | |
| Local:I-O | (--) | (--) | ABEmbedded_Disc... | |
| SKT_AOI_Client_LIMA | (--) | (--) | SKT_AOI_TCP_CLIENT | |
| SKT_DATA_Client_LIMA | (--) | (--) | SKT_DATA_CLIENT | |
| SKT_DATA_Client_LIMA.Connect_Source | (--) | (--) | SKT_OpenConnectio... | |
| SKT_DATA_Client_LIMA.Connect_Source.DestAddr | (--) | (--) | STRING | |
| SKT_DATA_Client_LIMA.Connect_Source.Timeout | 2000 | (--) | DINT | |
| SKT_DATA_Client_LIMA.Create_Source | (--) | (--) | Decimal | |
| SKT_DATA_Client_LIMA.Instance | 23941 | (--) | DINT | |
| SKT_DATA_Client_LIMA.Read_Response | (--) | (--) | SKT_Read_Response | |
| SKT_DATA_Client_LIMA.Read_Source | (--) | (--) | SKT_Read_Parameters | |
| SKT_DATA_Client_LIMA.Timeout | 2000 | (--) | DINT | |
| SKT_DATA_Client_LIMA.Write_SizeSent | 8 | (--) | DINT | |
| SKT_DATA_Client_LIMA.Write_Source | (--) | (--) | SKT_Write_Parameters | |
| SKT_MSG_Client_Connect_LIMA | (--) | (--) | MESSAGE | |
| SKT_MSG_Client_Create_LIMA | (--) | (--) | MESSAGE | |
| SKT_MSG_Client_Delete_LIMA | (--) | (--) | MESSAGE | |
| SKT_MSG_Client_Read_LIMA | (--) | (--) | MESSAGE | |
| SKT_MSG_Client_Write_LIMA | (--) | (--) | MESSAGE | |
| SKT_MSG_Delete_All | (--) | (--) | MESSAGE | |

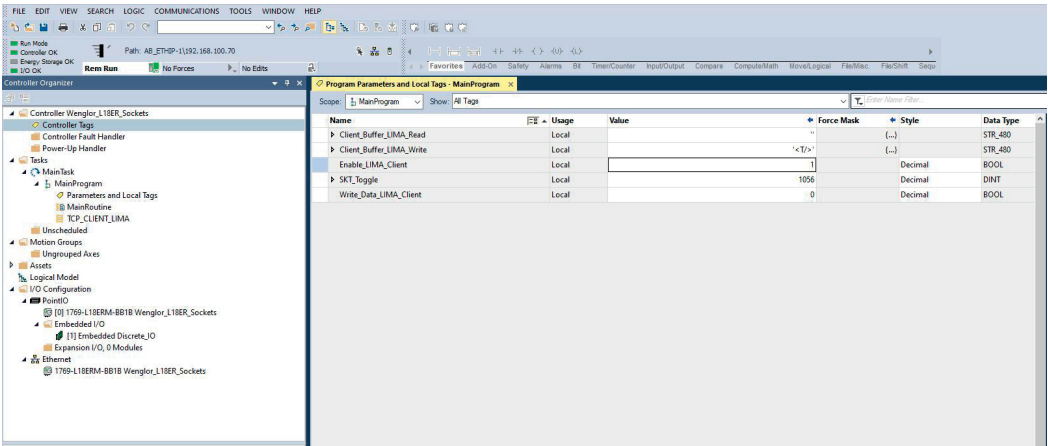
Transfer the sample program to the controller and go online.

NOTE!



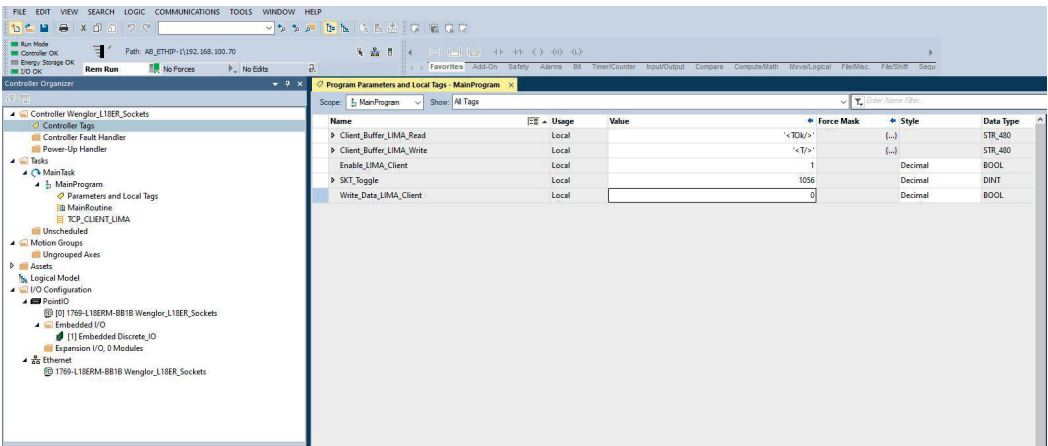
A connection from the controller to the uniVision product can be established only if port 32001 is available for the controller. Depending on the product or the mode of operation of the uniVision software, port 32001 may also be required by the uniVision software (e.g., in editing mode). In this case, the connection via the uniVision software may have to be disconnected so that the connection can be established via the controller.

The TCP connection is established by activating the value Enable_LIMA_Client under Parameters and Local Tags.



The LIMA command is sent to the uniVision product by activating Write_Data_LIMA_Client. The command may be sent once only and must not be permanently set so that only one image or profile is recorded. A new command must not be sent until the LIMA response to the previous command has been received.

The LIMA response is contained under Client_Buffer_LIMA_Read. For the trigger command, <TOK/> is sent by the uniVision product in response to a successful execution of the trigger command.



In addition, after data recording and evaluation, the new process data are also available via TCP (see the sample program Example_Rockwell_ProcessData.ACD). The run counter can be used, for example, to check if new results are available.

