Zebra
Aurora™ Vision

Aurora Vision Studio 5.7

Application Notes

Created: 11/10/2026

Product version: 5.7.5.80527

Table of content:

- Interfacing Zebra 4Sight computer (Profinet) to Aurora Vision Studio
- Interfacing Zebra 4Sight computer (Modbus TCP – Server) to Aurora Vision Studio
- Interfacing Zebra 4Sight computer (EtherNet/IP) to Aurora Vision Studio
- Hardware Triggering Zebra CV60 Camera using Zebra 4Sight EV7 Auxiliary Outputs
- Interacting with GigEVision cameras
- Changing parameters of GigEVision cameras
- Interfacing Photoneo to Aurora Vision Studio
- Interfacing Wenglor profile sensor to Aurora Vision Studio
- Interfacing Gocator to Aurora Vision Studio
- Acquiring data from SICK TriSpector devices
- Acquiring data from AT-C6 devices
- Testing Digital Outputs of a GigEVision Camera
- Using TCP/IP Communication
- Using Modbus TCP Communication
- Interfacing Zebra 4Sight computer (Modbus TCP – Client) to Aurora Vision Studio
- Using OPC UA Communication
- Interfacing Hilscher card (EtherCAT) to Aurora Vision Studio
- Interfacing Hilscher card (EtherNet/IP) to Aurora Vision Studio
- Interfacing Hilscher card (Profinet) to Aurora Vision Studio
- Interfacing Profinet gateway to Aurora Vision Studio
- Using Aurora Vision Studio Runtime for Linux with WebHMI

Interfacing Zebra 4Sight computer (Profinet) to Aurora Vision Studio

Purpose and equipment

This document explains how to configure Profinet PLC with Aurora Vision Studio using Zebra 4Sight EV6 computer.

Required equipment:

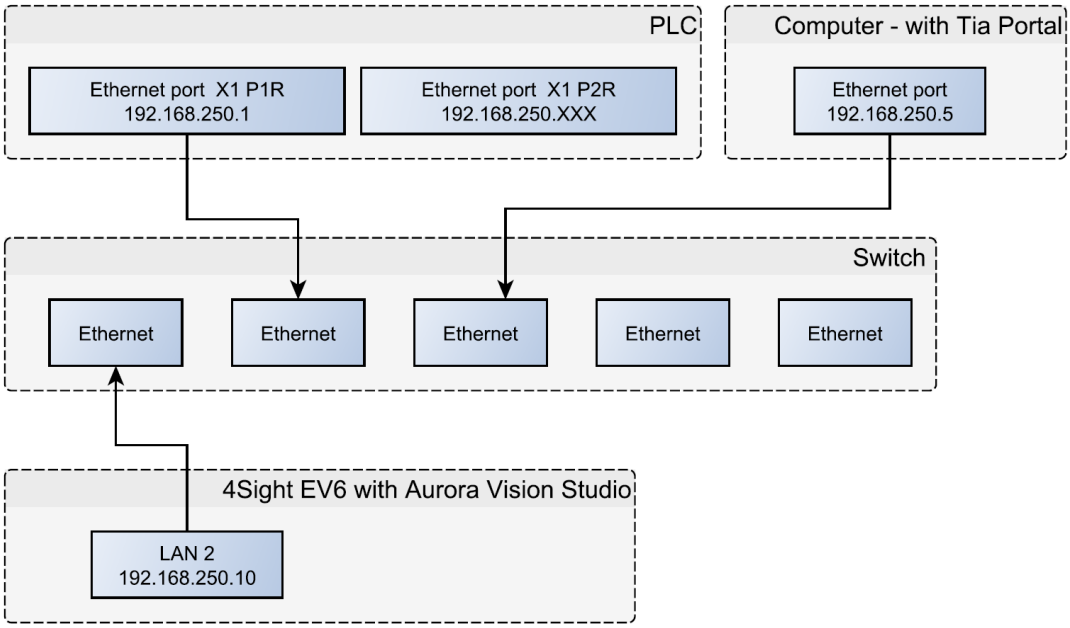
- 4Sight computer with Profinet (used 4Sight EV6);
- Siemens S7-1200 PLC;
- Ethernet switch with at least 3 In/Outs might be useful - simplifies configuration;
- TIA Portal (version v15 used);
- Aurora Vision Studio 5.3 or later (version 5.3 used).

Application notes demonstrates configuration steps for specific 4sight model, but it is not obligatory to use this one. If you have different model, please refer to manuals provided by its manufacturer.

Hardware connection

In application notes there is configured one network which uses Ethernet switch. Devices connected to this network have addresses 192.168.250.XXX, where XXX stands for number 1-254 unique for every device/port connected.

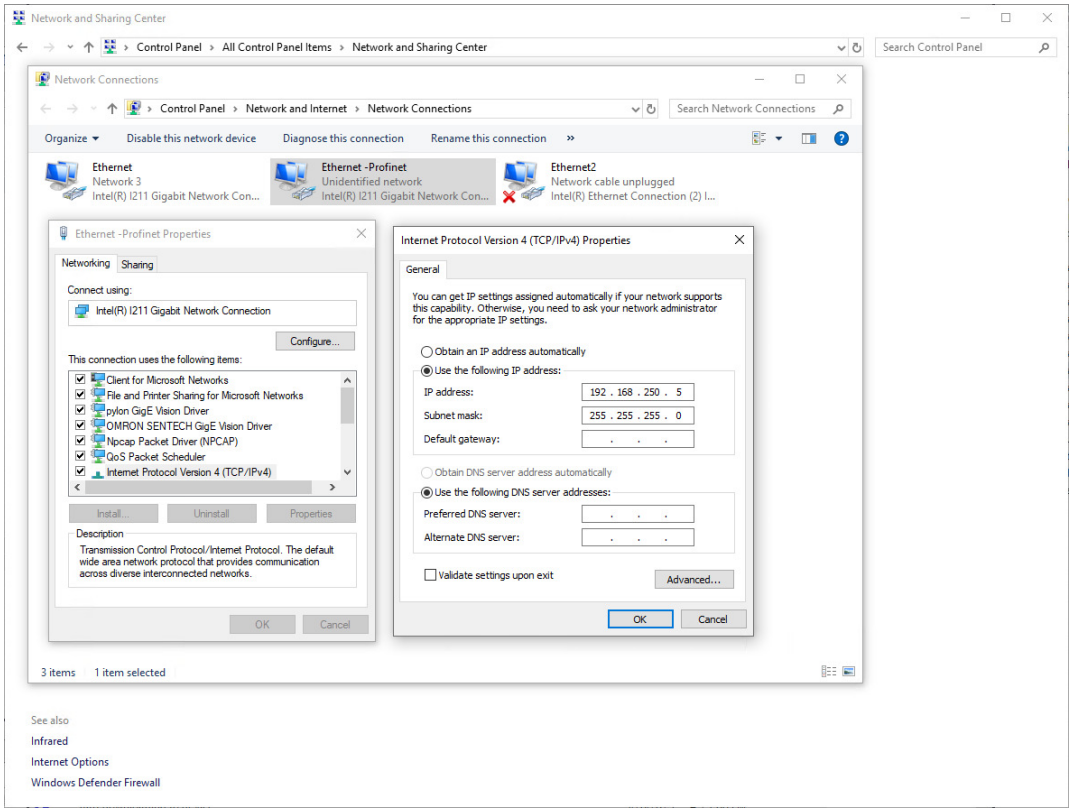
In EV6 Zebra Profinet manual it is mentioned that PLC Software should not be installed on EV6. Taking this into account, we have one PLC and two computers in our network:



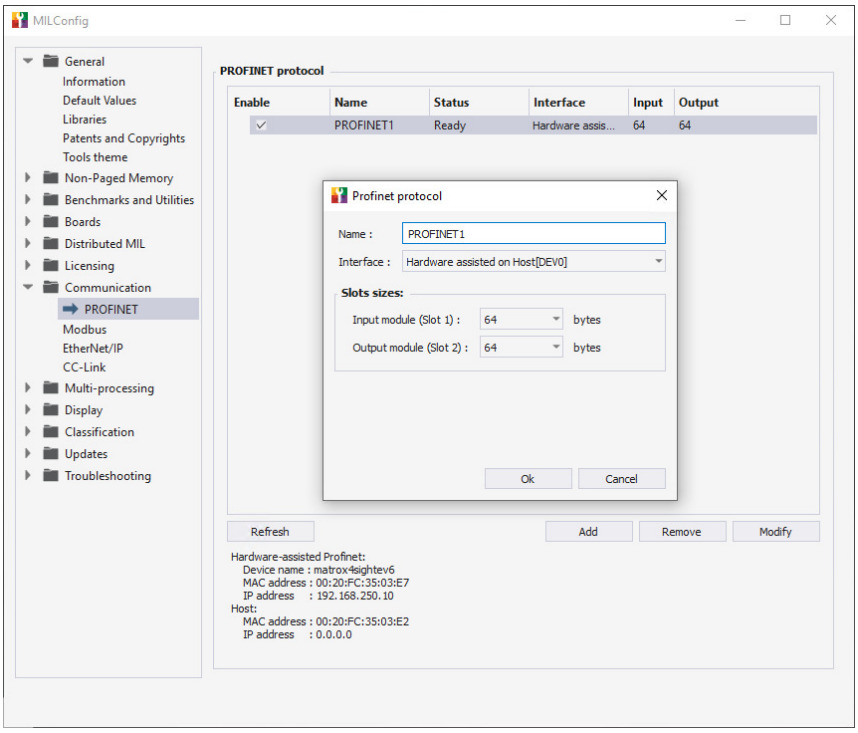
Before proceeding to the further point, make sure that all devices are powered up. **Using LAN 2 port for industrial protocols is obligatory**.

Configuring setup

1. As first configure computer with TIA Portal. Change address of the network card connected to the switch to 192.168.250.5 as shown below:



2. Enable profinet at 4Sight EV6 Computer using MILConfig tool (IP address will not be defined yet):



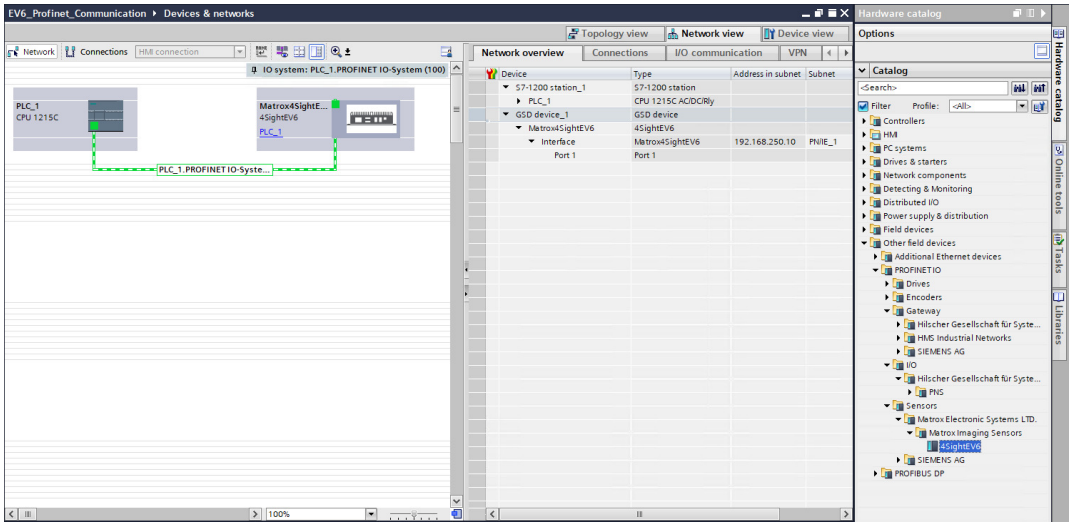
Configuring 4Sight EV6 in TIA Portal

1. Create TIA Portal project with your PLC. Find GSD file in MIL X directory, you can find it placing this in File Explorer:

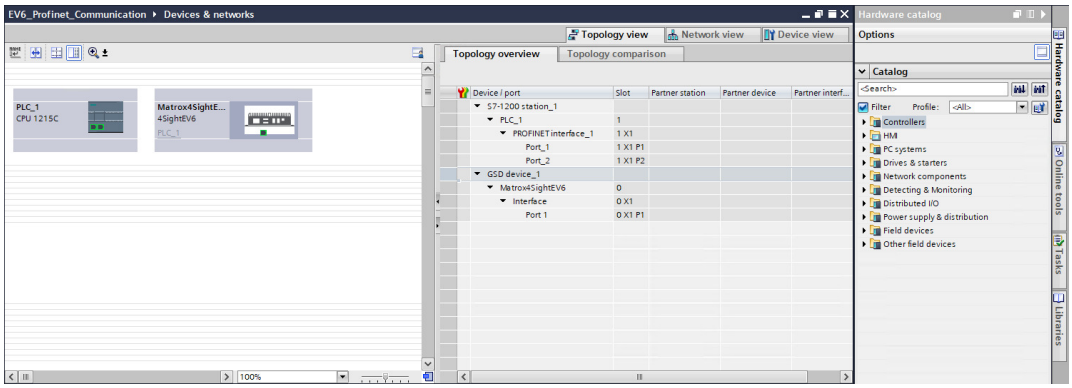
```
%MIL_Path%\..\Config\Profinet
```

In TIA Portal add GSD file from above mentioned firmware folder using tab menu on the top: "Options → Manage general station … → Select file → Install".

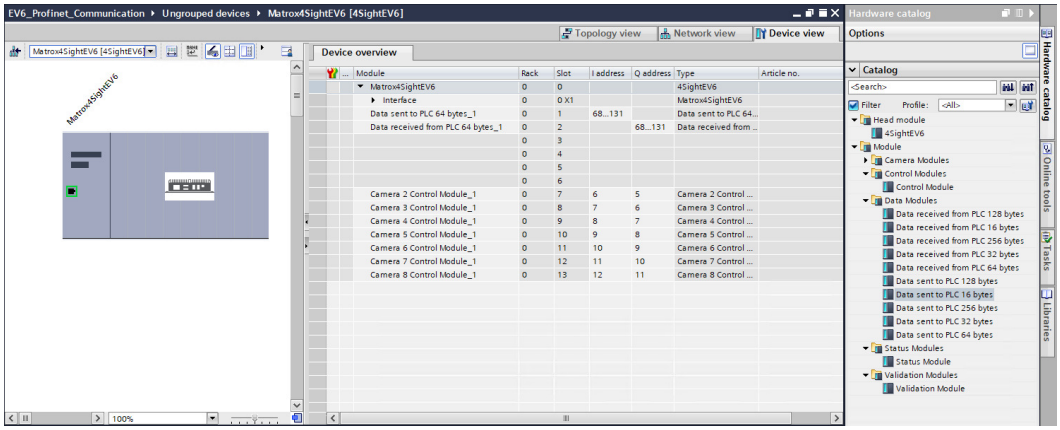
Once you upload GSD, you can find appropriate device in **Other field devices -> PROFINET IO → Sensors → Zebra Imaging Sensors**. *Drag & drop* it onto **Network view** and combine it as on the screen attached below:



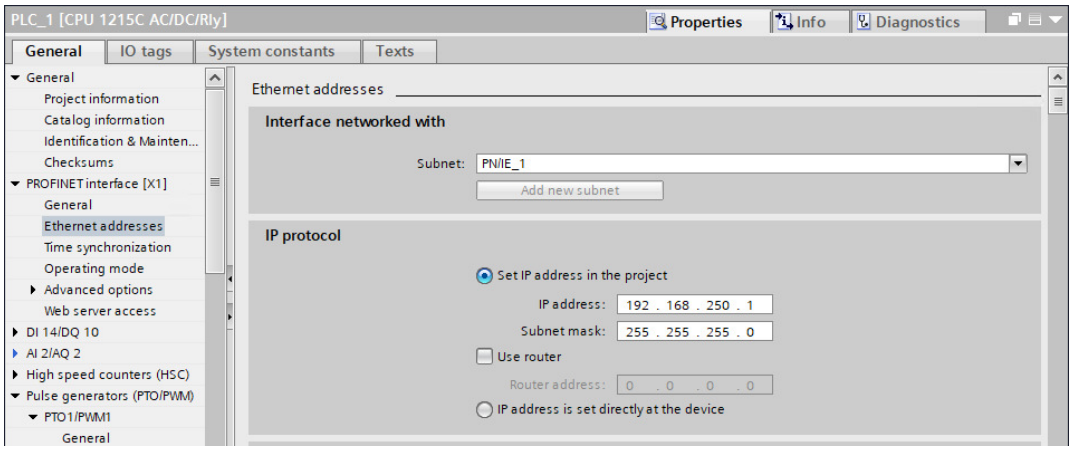
Topology view should be left unchanged:



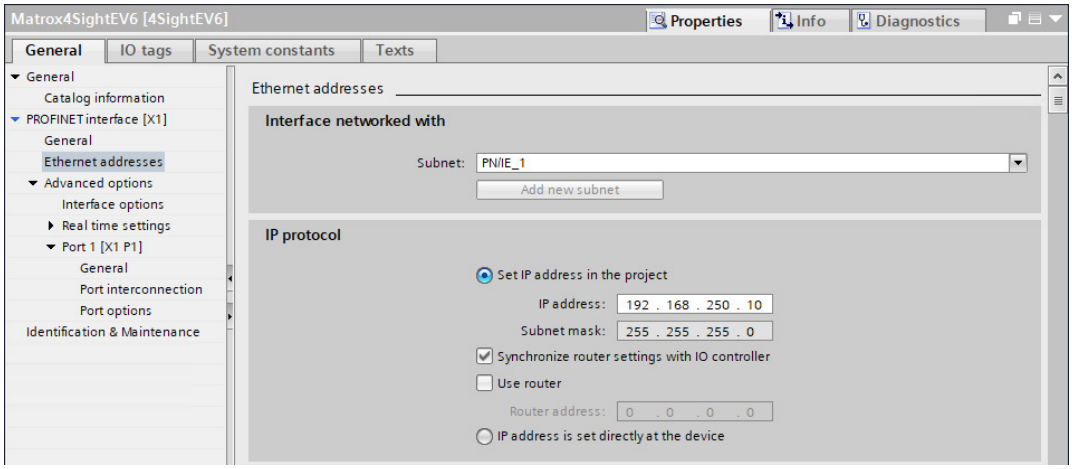
2. *Drag & drop* appropriate data modules in Device view tab (in MILConfig 64bytes modules were set):



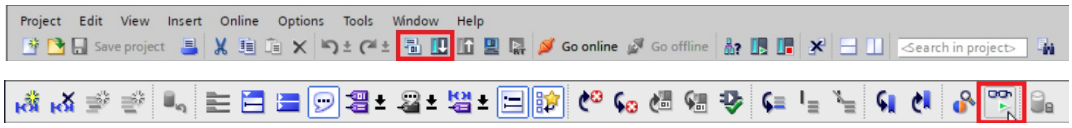
Change PLC's IP to **192.168.250.1**: 3.



4. Change 4Sight's IP to **192.168.250.10**:



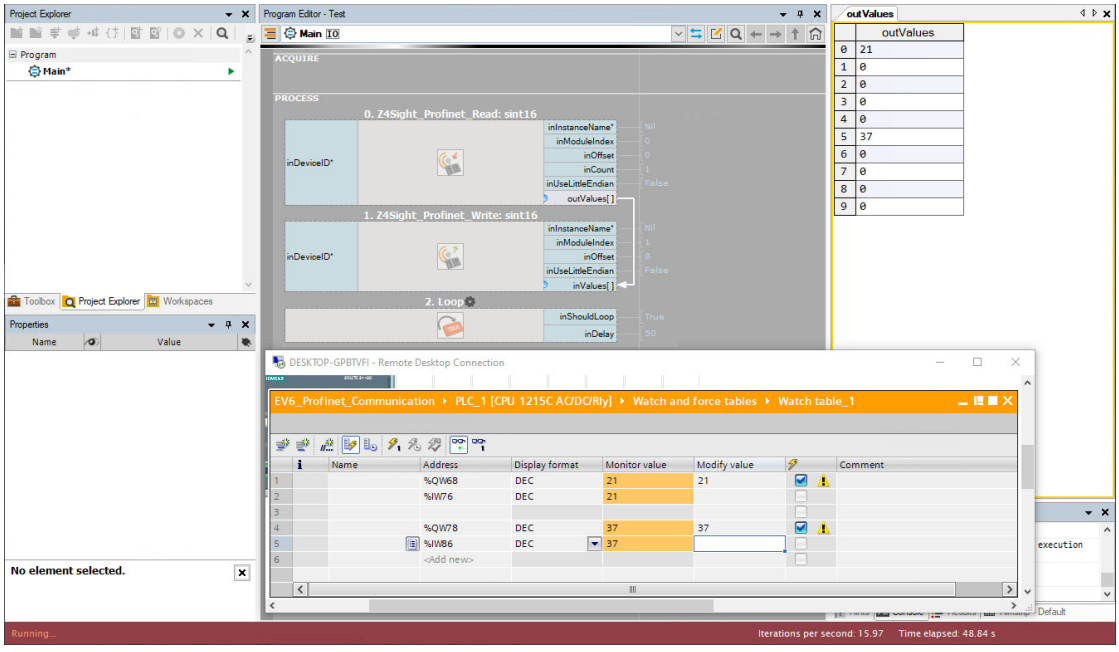
5. If all previous steps have been done correctly, your PLC program is ready. Compile the program and download it to the device. To see current states of variables, turn on Monitoring mode:



To monitor and easily modify tags please add *Watch table* in **Watch and force tables** folder.

Configuring Aurora Vision Studio application

Turn on online mode and monitoring mode in TIA Portal. You can create *watch tables* to easily modify data in TIA Portal environment:



Troubleshooting

In case of any problem with 4Sight configuration, please refer to the MIL X manuals that could be found under:

```
%MIL_Path%\..\DOC\help
```

More detailed information could be found in Zebra Design Assistant help file.

Interfacing Zebra 4Sight computer (Modbus TCP – Server) to Aurora Vision Studio

Purpose and equipment

This document explains how to configure Modbus TCP PLC with Aurora Vision Studio using Zebra 4Sight computer. In this setup, the Zebra 4Sight computer running Aurora Vision Studio acts as a Modbus Server (Slave), while the PLC acts as a Modbus Client (Master).

Required equipment:

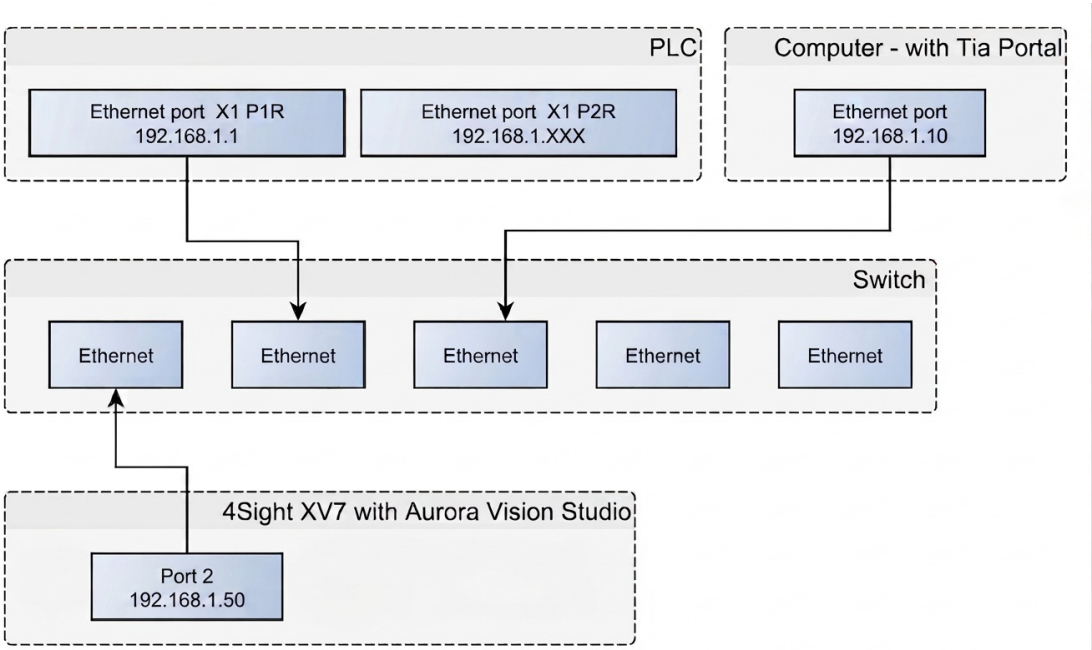
- 4Sight computer (used 4Sight XV7);
- Siemens S7-1200 PLC;
- Ethernet switch with at least 3 In/Outs might be useful – simplifies configuration;
- PC with TIA Portal V19 (for PLC configuration);
- Aurora Vision Studio 5.6 or later (version 5.6 used).

Application notes demonstrates configuration steps for specific 4sight model, but it is not obligatory to use this one. If you have different model, please refer to manuals provided by its manufacturer.

Hardware connection

In application notes there is configured one network which uses Ethernet switch. Devices connected to this network have addresses 192.168.1.XXX, where XXX stands for number 1–254 unique for every device/port connected.

In Zebra 4Sight manual it is mentioned that PLC Software should not be installed on Z4Sight PC. Taking this into account, we have one PLC and two computers in our network:

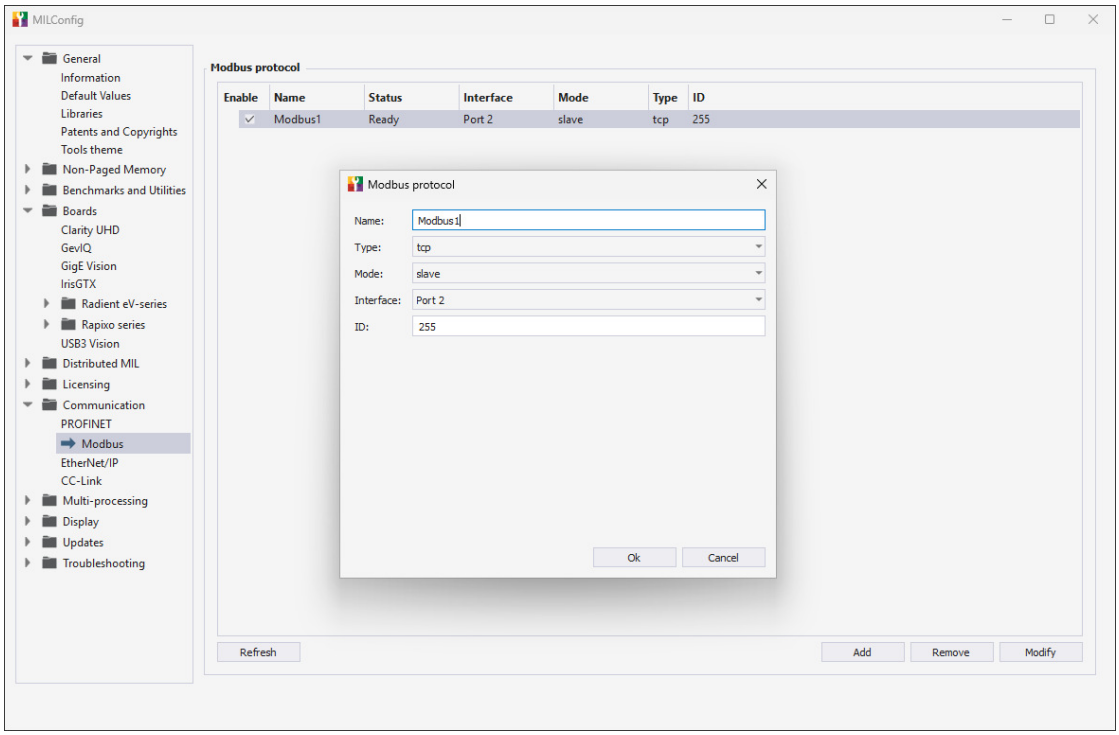


Before proceeding to the further point, make sure that all devices are powered up.

Configuring MIL on Zebra 4Sight

Before creating the application in Aurora Vision Studio, the Modbus slave service must be enabled in the MIL configuration.

1. Open the MILConfig application on the Z4Sight computer.
2. In the left pane, navigate to *Communication -> Modbus*.
3. Add a new instance. Set parameters:
    - Name: **Modbus1**
    - Type: **tcp**
    - Model: **slave**
    - Interface: **Port 2**
    - ID: **255**
4. Save the configuration

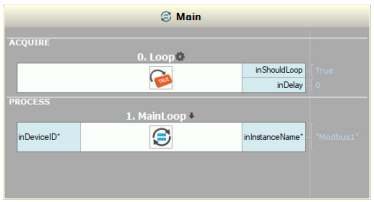


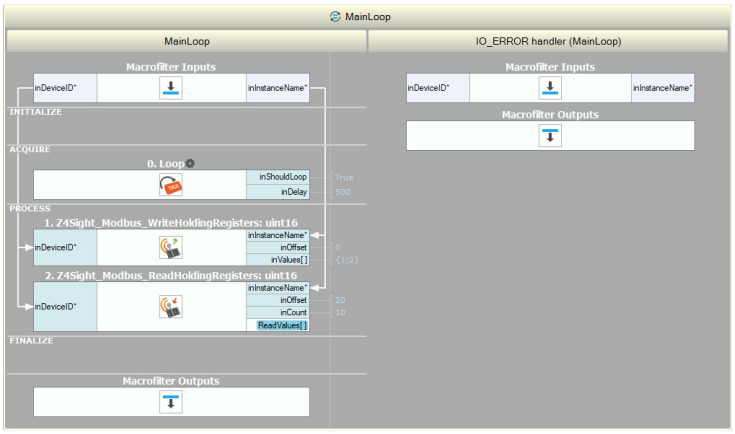
Aurora Vision Studio application

The application utilizes dedicated Aurora Imaging Library Modbus filters to read and write holding registers.

1. Inside the Main macrofilter, go to the **ACQUIRE** section and add a Loop filter to keep the task running continuously.
2. In the **PROCESS** section, create a new *Task Macrofilter* named **MainLoop**. Add two macrofilter inputs:
   - **inDeviceID** of type *AILSystemAndDeviceIdentifier**
   - **inInstanceName** of type *String**

   Set the inInstanceName value to match the MILConfig setup (*Modbus1*).
3. Open the MainLoop macrofilter. In the **ACQUIRE** section, add the Loop filter. Set its inDelay input to the desired polling rate (e.g., 500 ms) to regulate the communication frequency.
4. In the **PROCESS** section, add the AILSystem_Modbus_Server_WriteHoldingRegisters_uint16 filter. Link the inDeviceID and inInstanceName from the macrofilter inputs. Set **inOffset** to *0* and the **inValues** to *{1;2}*.
5. Add the AILSystem_Modbus_Server_ReadHoldingRegisters_uint16 filter. Link the inDeviceID and inInstanceName from the macrofilter inputs. Set **inOffset** to *20* and **inCount** to *10*. Present the **outValues** in the Preview window.
6. In the Project Explorer, right-click on the **MainLoop** macrofilter block, select **Add New Error Handler**, and choose **IO_ERROR**.





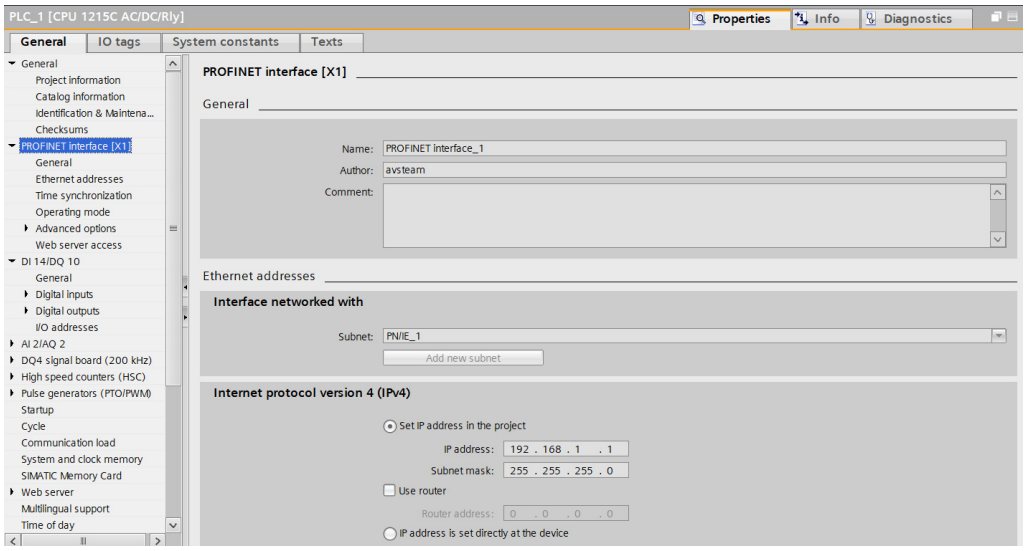
Configuring PLC in TIA Portal

Note: Aurora Vision Studio (and the Modbus protocol standard) transmits data in Big-Endian format. Siemens S7-1200 PLCs store data in Little-Endian format. A byte-swap conversion will be required on the PLC side to interpret the values correctly.

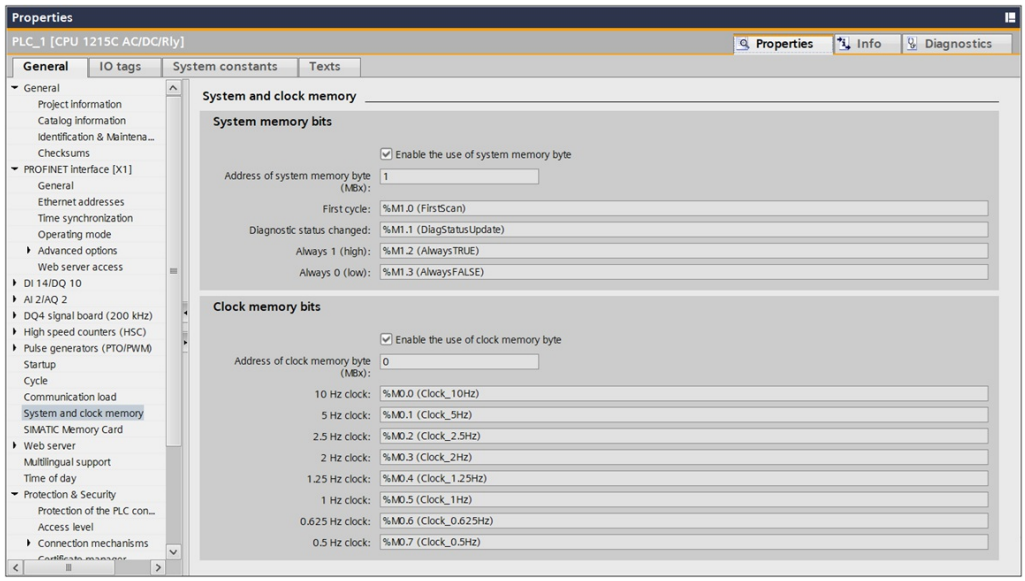
1. **Device Configuration**

   After adding the S7-1200 PLC to the project, apply the following settings in the **Properties** tab:

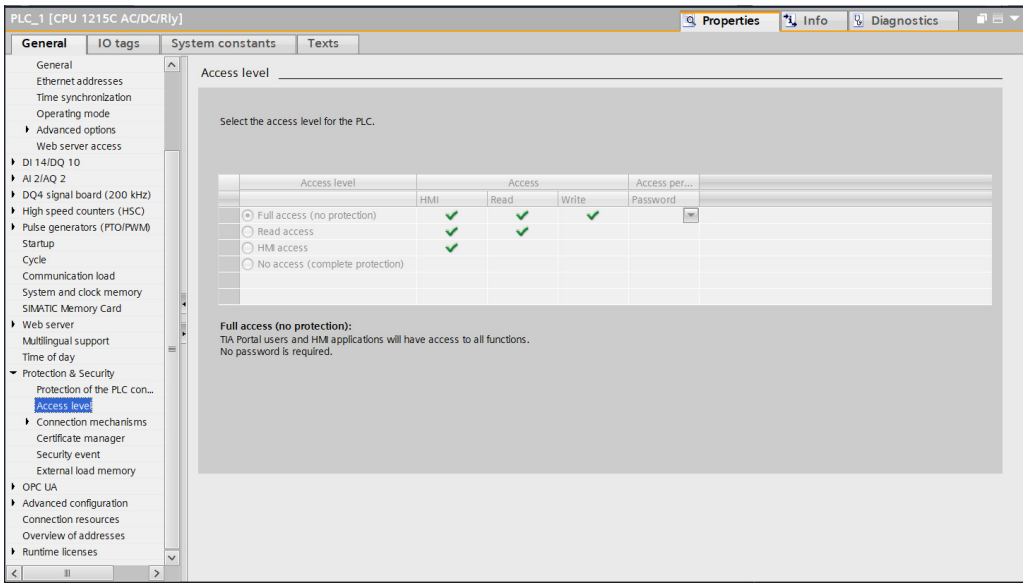
   - *PROFINET interface [X1]*: Set the appropriate IP address and add/setup the Subnet. Ensure it is in the same subnet as the Z4Sight computer.



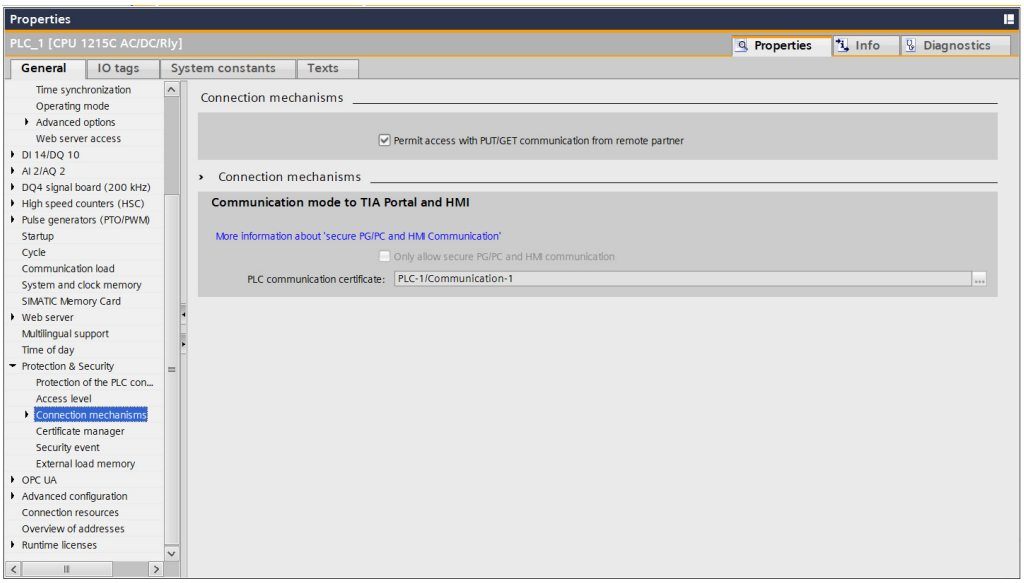
   - *System and clock memory*: Check **Enable the use of system memory byte** and **Enable the use of clock memory byte**.



   - *Access level*: Set to **Full access** (no password).



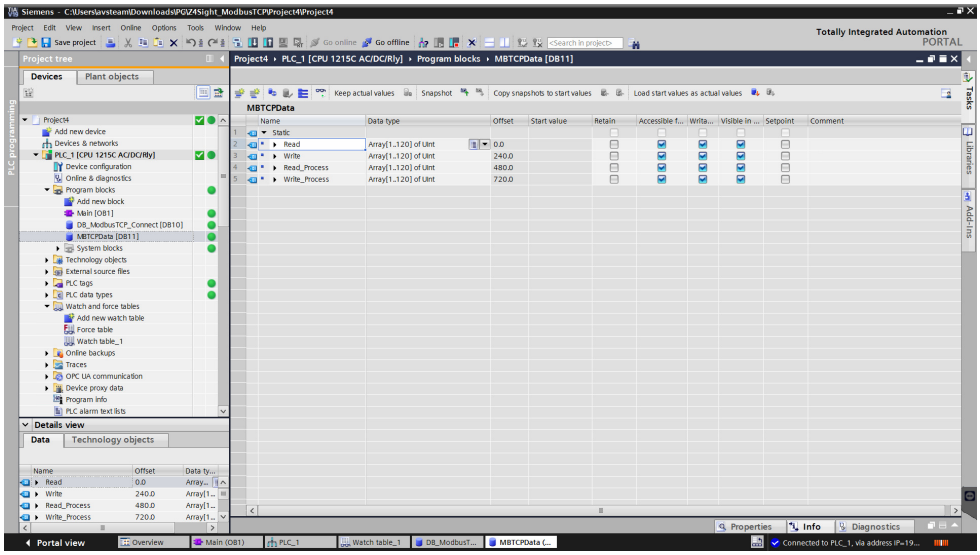
- *Connection mechanisms*: Check **Permit access with PUT/GET communication from remote partner** and uncheck **Only allow secure PG/PC and HMI communication**.



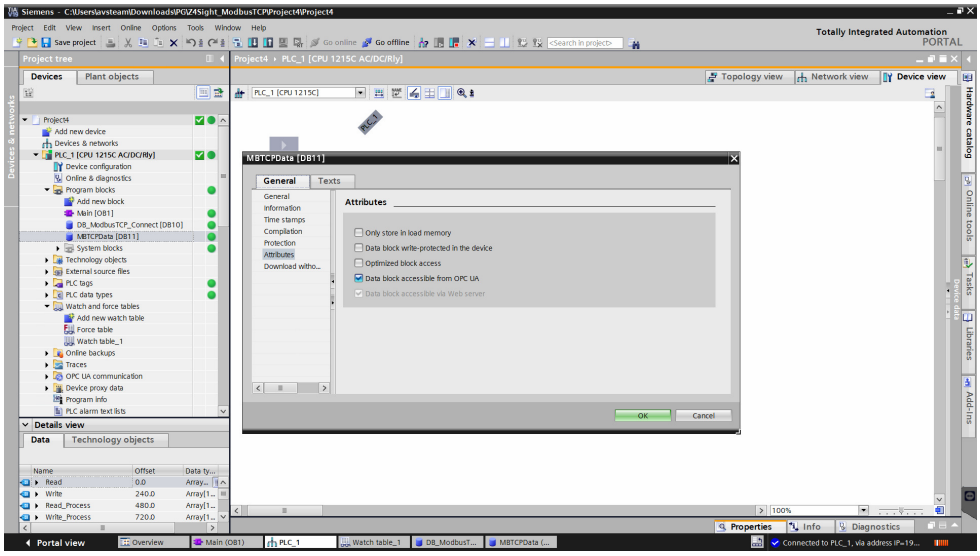
2. **Creating Data Blocks**

   Two Data Blocks are required to manage the data arrays and the connection parameters:

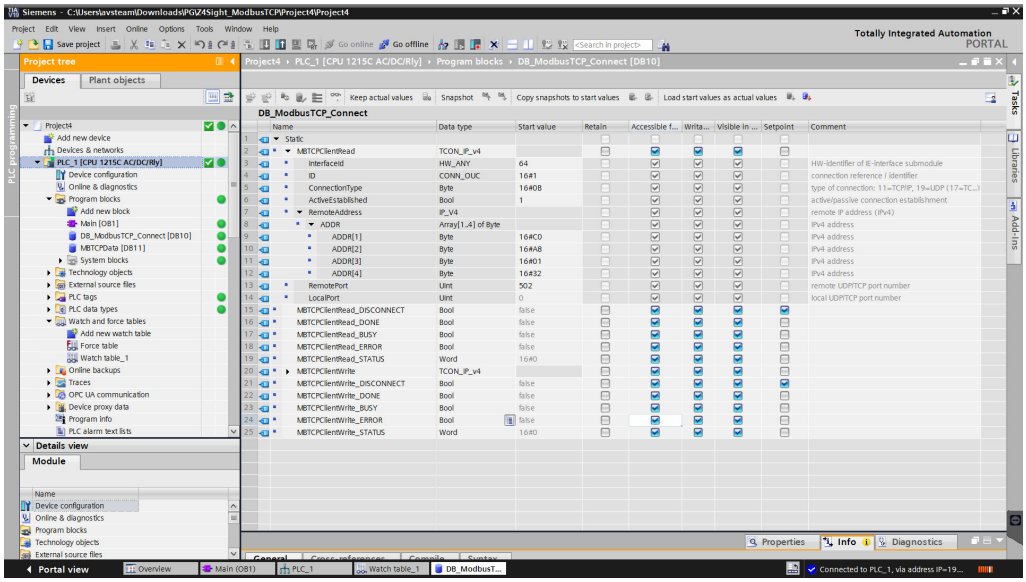
- **MBTCPData [DB11]** – it holds the raw Modbus arrays and the processed (byte-swapped) arrays.
  1. Define the following variables:
     - **Read** of type *Array[1..120] of UInt* (Raw data received from AVS)
     - **Write** of type *Array[1..120] of UInt* (Raw data to be sent to AVS)
     - **Read_Process** of type *Array[1..120] of UInt* (Byte-swapped data ready for PLC logic)
     - **Write_Process** of type *Array[1..120] of UInt* (PLC data before byte-swapping for transmission)



  2. Right-click the block, go to *Properties -> Attributes*, and uncheck the **Optimize block access** option. This is mandatory to expose absolute addressing.

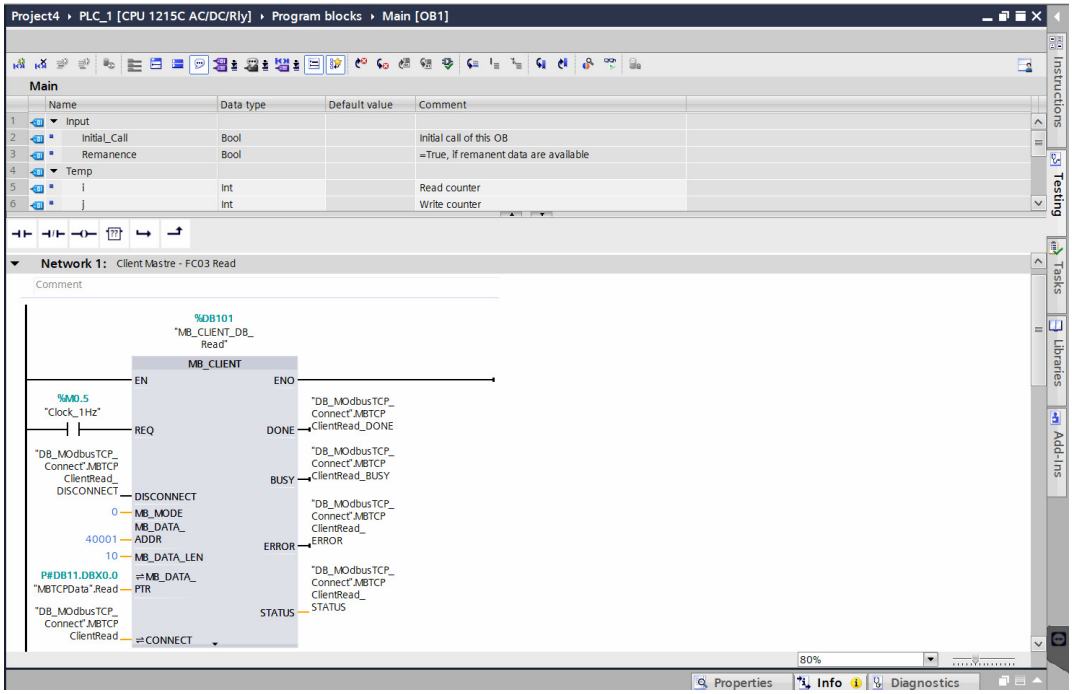


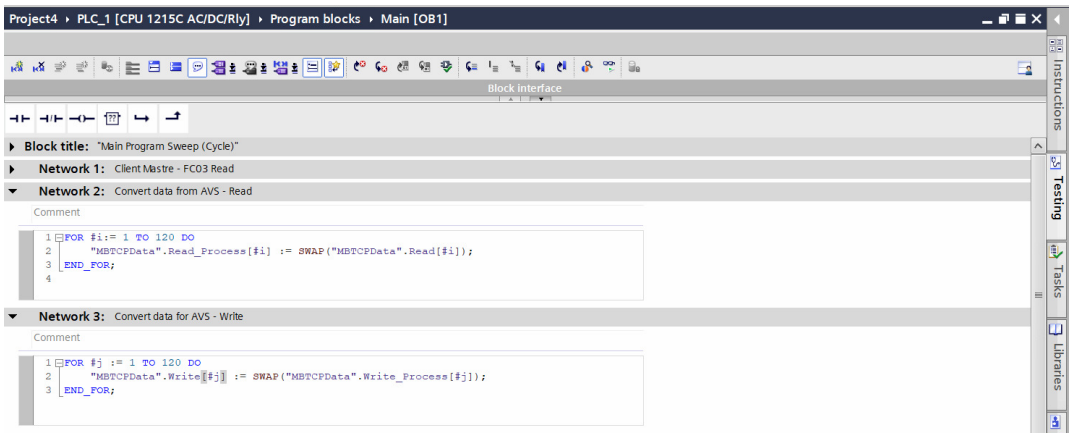
- **DB_ModbusTCP_Connect [DB10]** – it stores the connection parameters and status flags for both the read and write operations. Set their Start values as follows:
  1. **MBTCPClientRead** of type *TCON_IP_v4*:
     - *InterfaceId*: **64**
     - *ConnectionType*: **16#0B**
     - *RemoteAddress*: **192.168.1.50** (IP address of the Z4Sight computer)
     - *RemotePort*: **502**
     - *ID*: **16#1** (Must be unique and different from the ID set in MILConfig).
  2. Add *Bool* flags: **MBTCPClientRead_DISCONNECT**, **MBTCPClientRead_DONE**, **MBTCPClientRead_BUSY**, **MBTCPClientRead_ERROR**.
  3. Add **MBTCPClientRead_STATUS** of type *Word*.
  4. Create a corresponding set of variables for **MBTCPClientWrite** (using a unique ID: **16#2**).

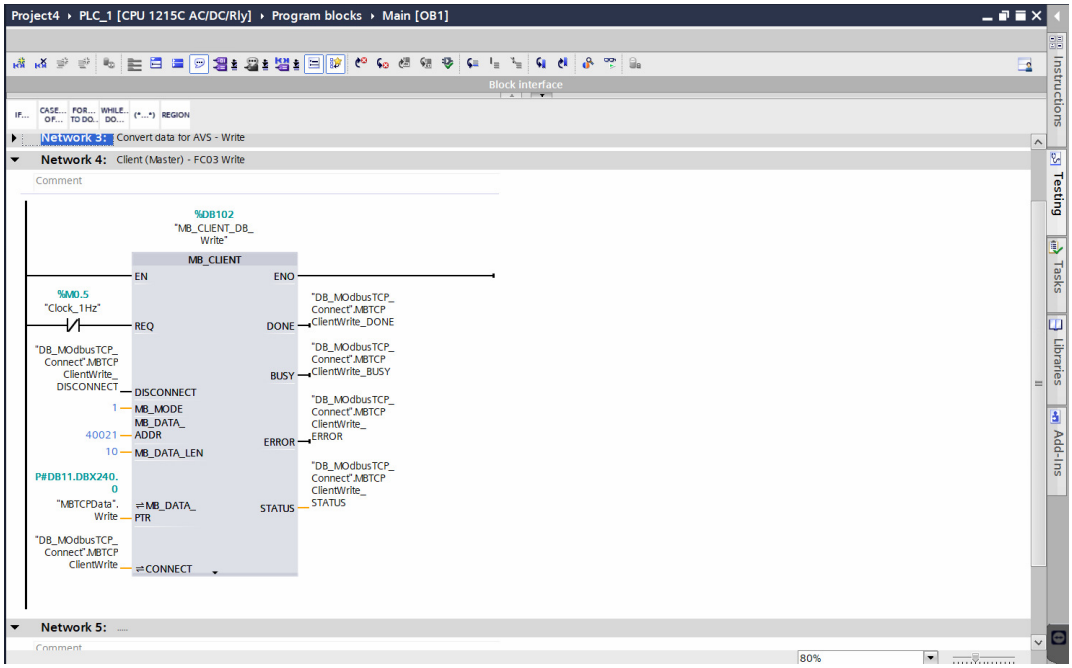


3. **Main Logic [OB1]**

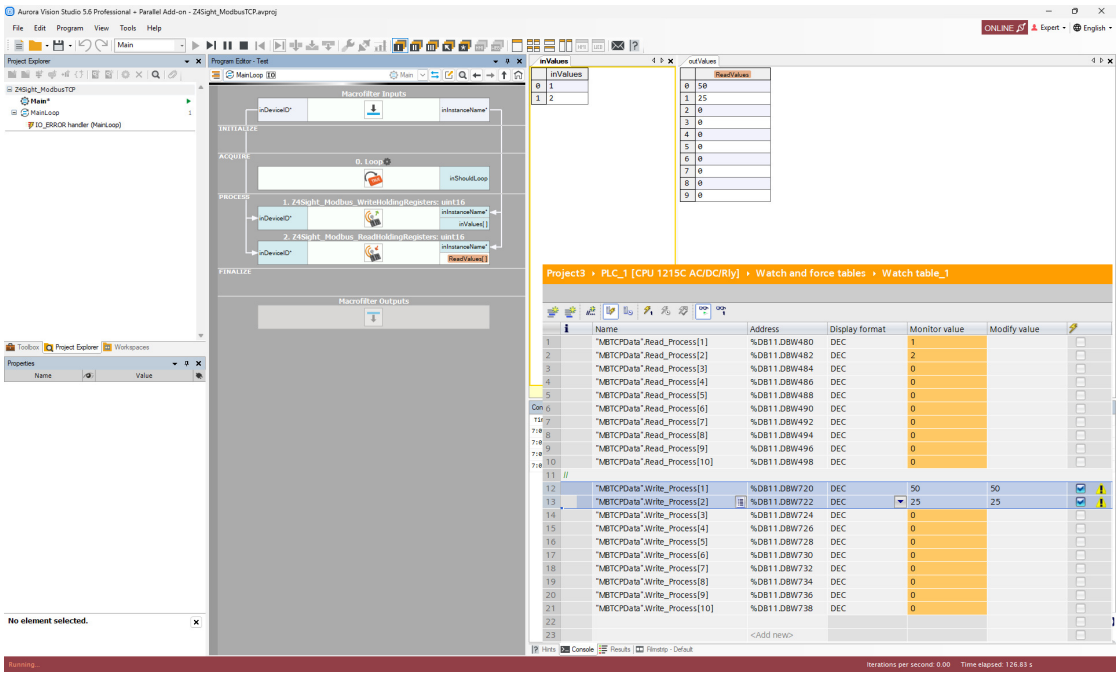
   Create structure as showed in images below:







When monitoring values in the TIA Portal **Watch Table**, always refer to the **Read_Process** and **Write_Process** arrays to see the correct decimal values.



Interfacing Zebra 4Sight computer (EtherNet/IP) to Aurora Vision Studio

Purpose and equipment

This document explains how to configure EthernetIP PLC with Aurora Vision Studio using Zebra 4Sight EV6 computer.

Required equipment:

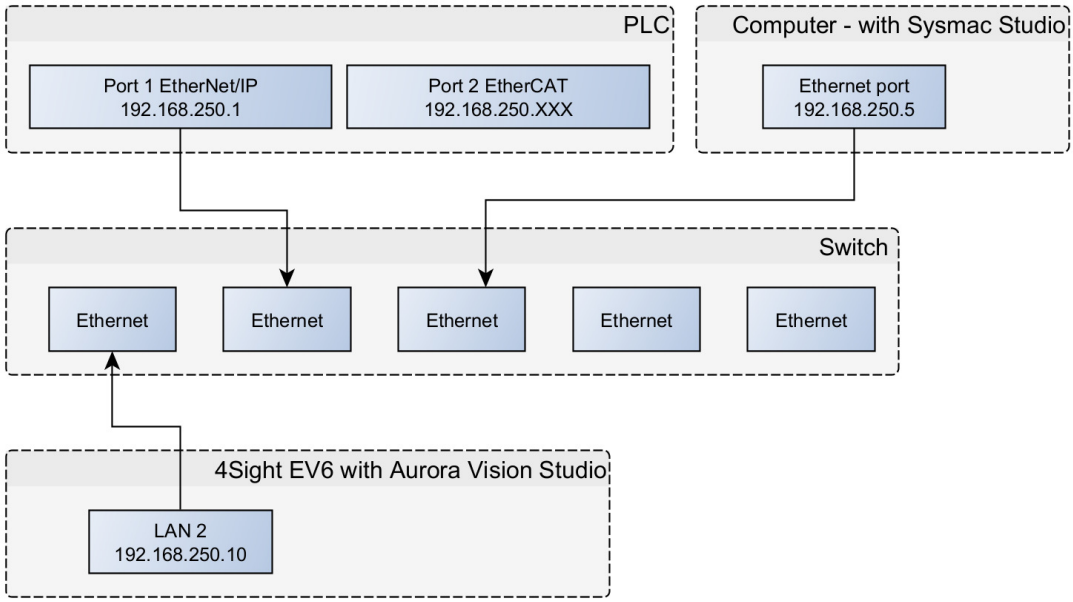
- 4Sight computer with EtherNet/IP (used 4Sight EV6);
- EtherNet/IP PLC (used NX1P2-9024DT1);
- Ethernet switch with at least 3 In/Outs might be useful - simplifies configuration;
- Sysmac Studio (software to configure PLC);
- Aurora Vision Studio 5.3 or later (version 5.3 used).

Application note demonstrates configuration steps for specific 4Sight model, but it is not obligatory to use this one. If you have different model, please refer to manuals provided by its manufacturer. Different PLC which supports EtherNet/IP can be also used.

Hardware connection

In application notes there is configured one network which uses Ethernet switch. Devices connected to this network have addresses 192.168.250.XXX, where XXX stands for number 1-254 unique for every device/port connected.

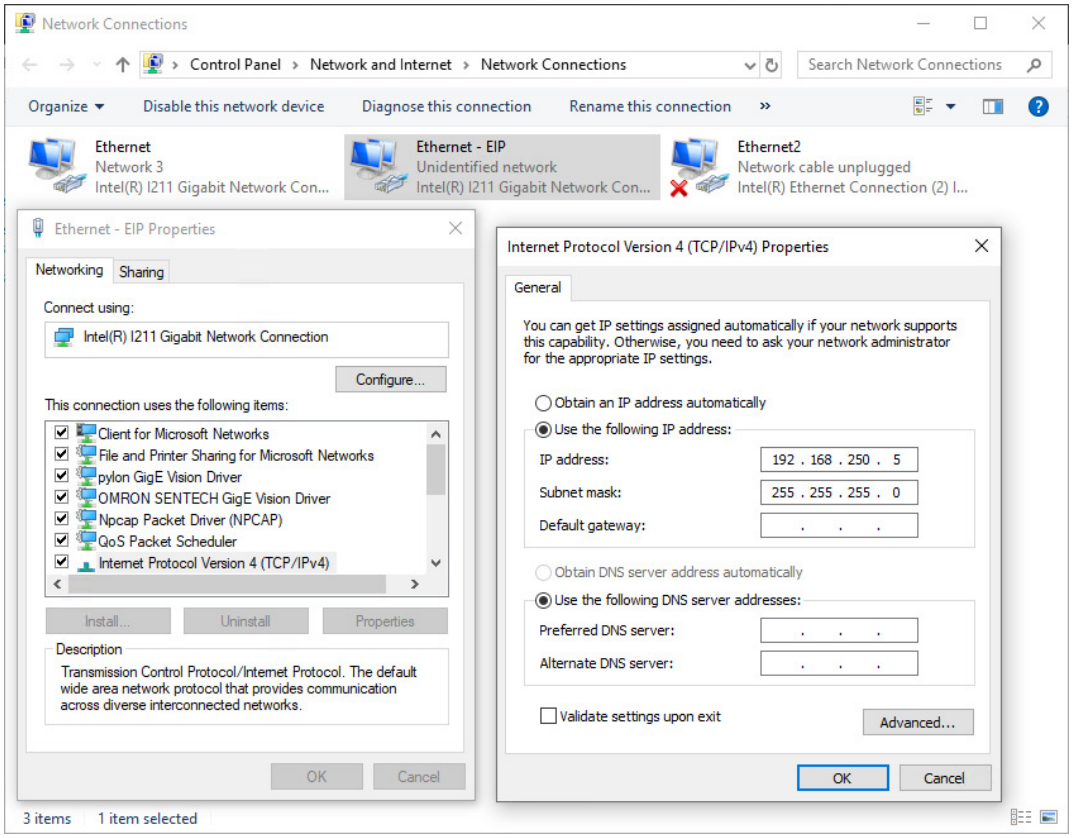
In EV6 Zebra 4Sight EtherNet/IP manual it is mentioned that PLC Software should not be installed on EV6. Taking this into account, we have one PLC and two computers in our network:



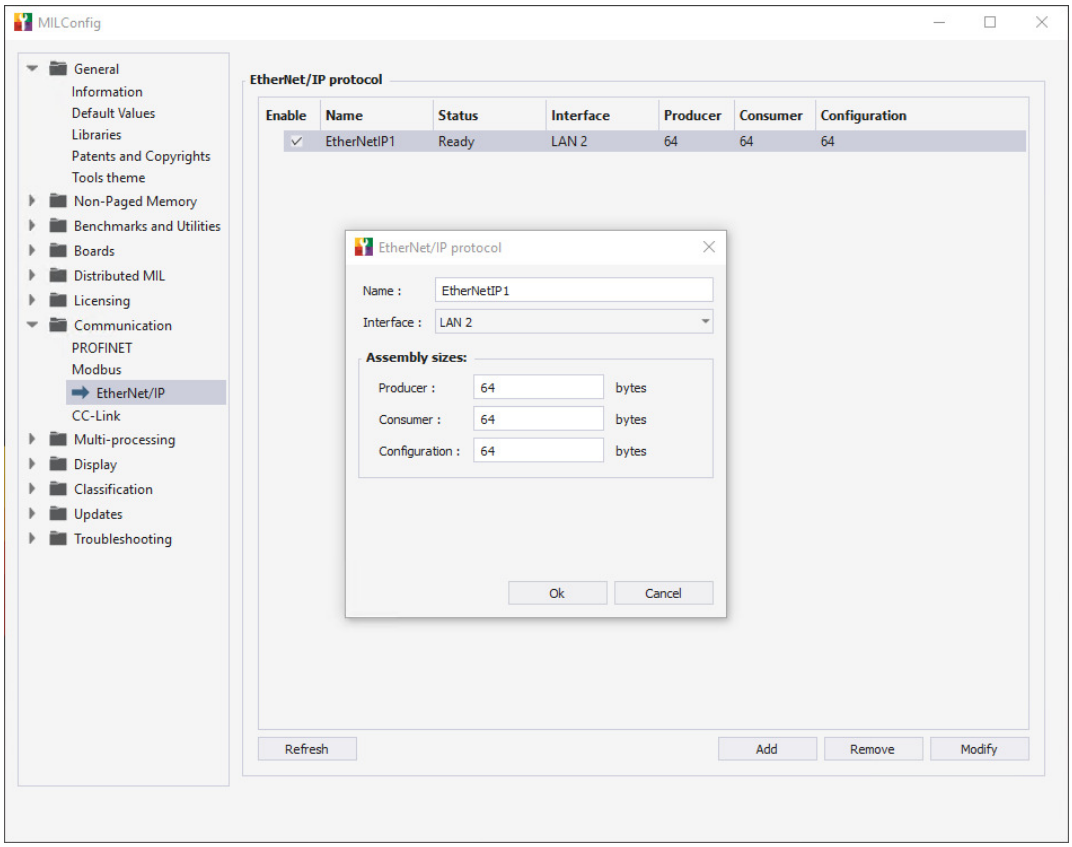
Before proceeding to the further point, make sure that all devices are powered up. **Using LAN 2 port for industrial protocols is obligatory**.

Configuring setup

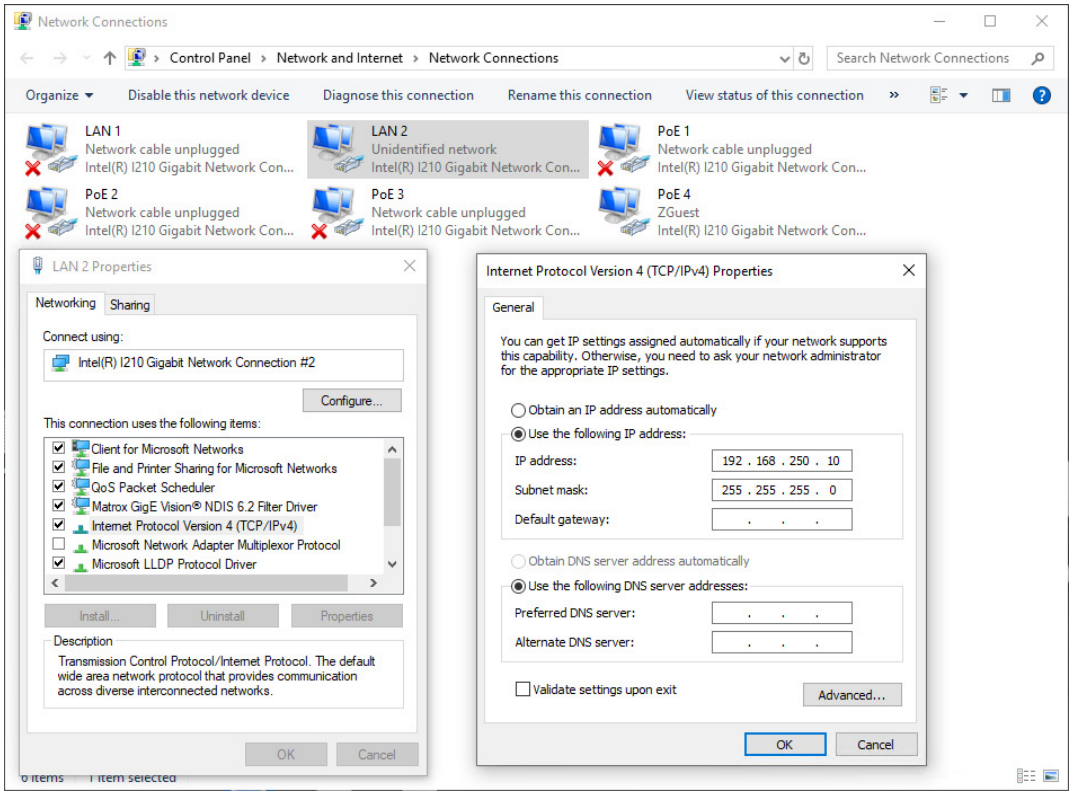
1. As first configure computer with Sysmac Studio. Change address of the network card connected to the switch to 192.168.250.5 as shown below:



2. Enable EtherNet/IP at 4Sight EV6 Computer using MILConfig tool:



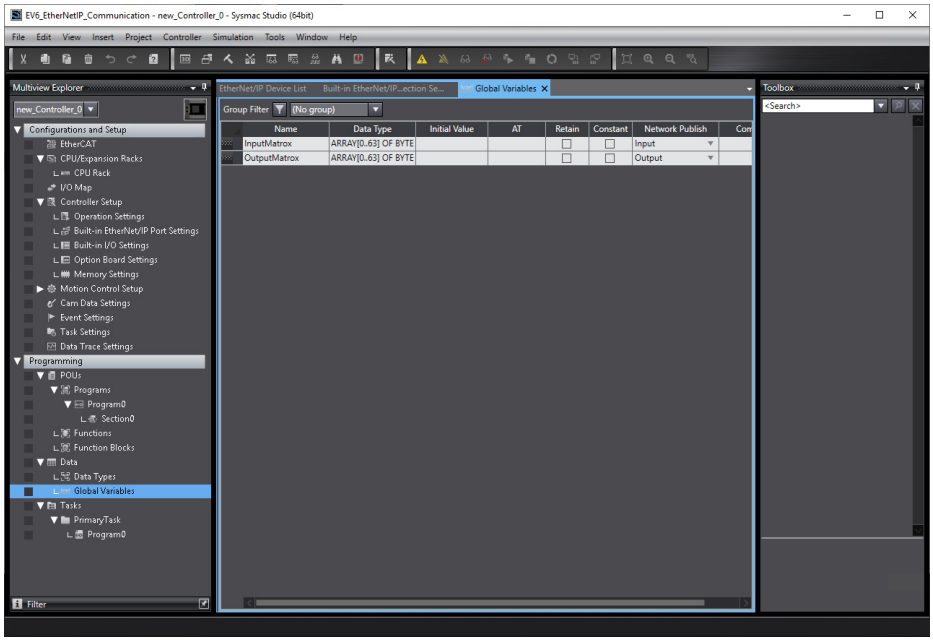
3. Change address of the network card of 4Sight EV6 computer (**LAN 2 port**) connected to the switch to 192.168.250.10 as shown below:



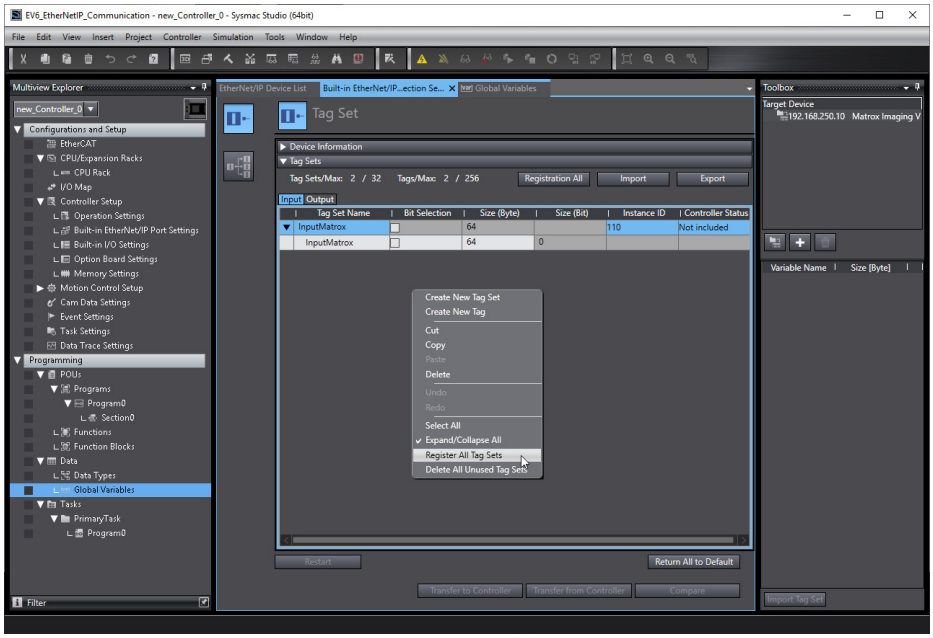
Configuring 4Sight EV6 in Sysmac Studio

Below you will find an example configuration process for EtherNet/IP PLC Omron NX1P2-9024DT in Sysmac Studio software.

There were created two global variables **InputZebra 4Sight** and **OutputZebra 4Sight** using "ARRAY [0..63] OF BYTE" data type to match configuration set in MILConfig.



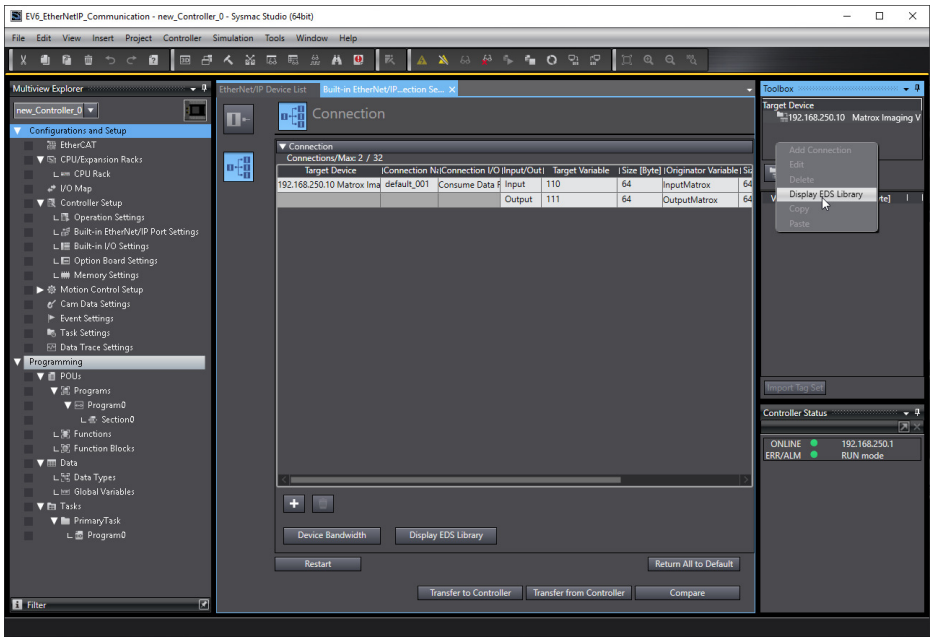
Then navigate to (Tools -> EtherNet/IP Connection Setting) and "Register All Tag Sets".



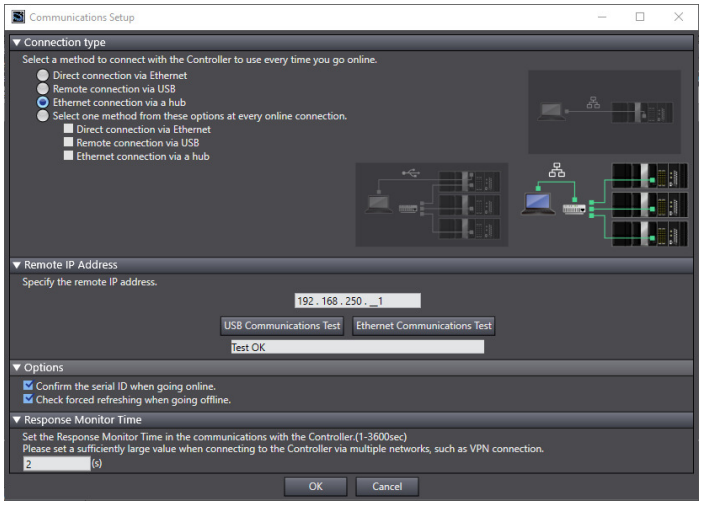
Add support for Zebra 4Sight computer using EDS files that could be found under:

```
%MIL_Path%\..\Config\EthernetIP
```

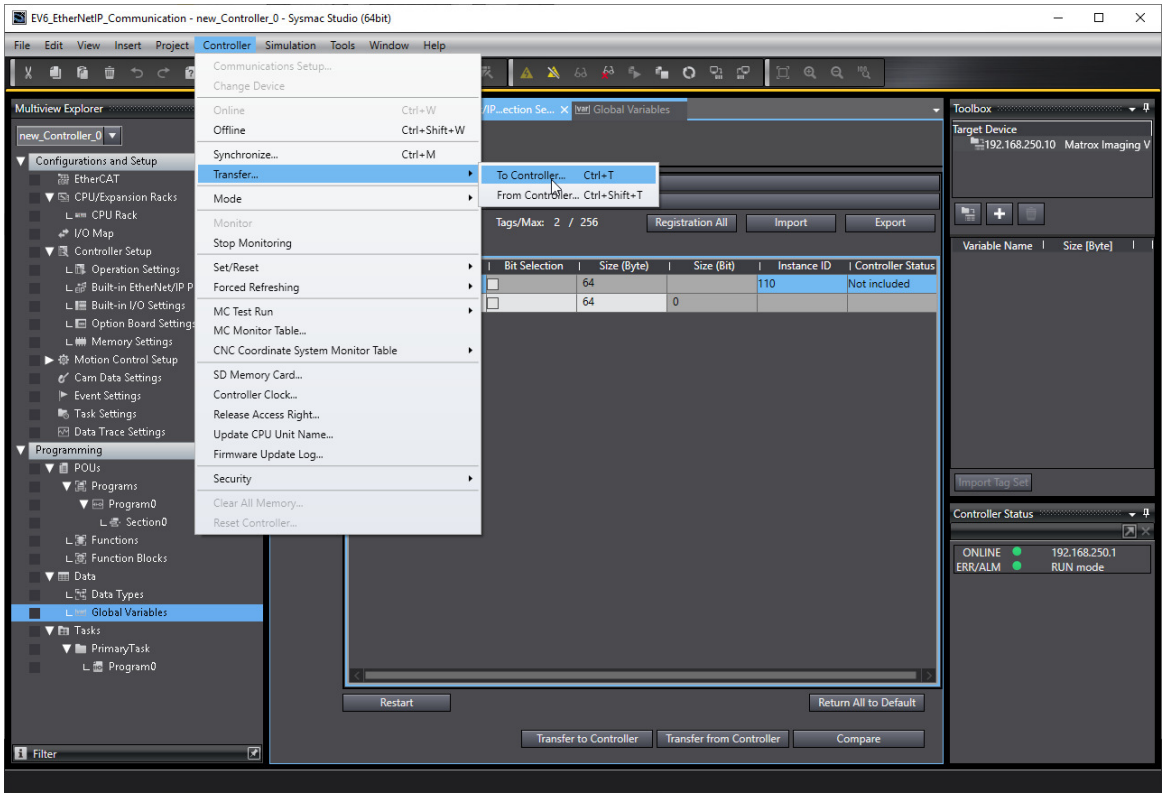
Then add "Zebra 4Sight Imaging Vision System" and configure Input and Output like below:



For this moment it is essential to connect with a PLC device, to do this in Sysmac Studio you need to navigate to "Controller -> Communication Setup...". There might be an error in Ethernet Communication Test if your network card is configured as DHCP (**Configuration of the network card** section in this tutorial). If communication with PLC is established you will receive "Test OK".

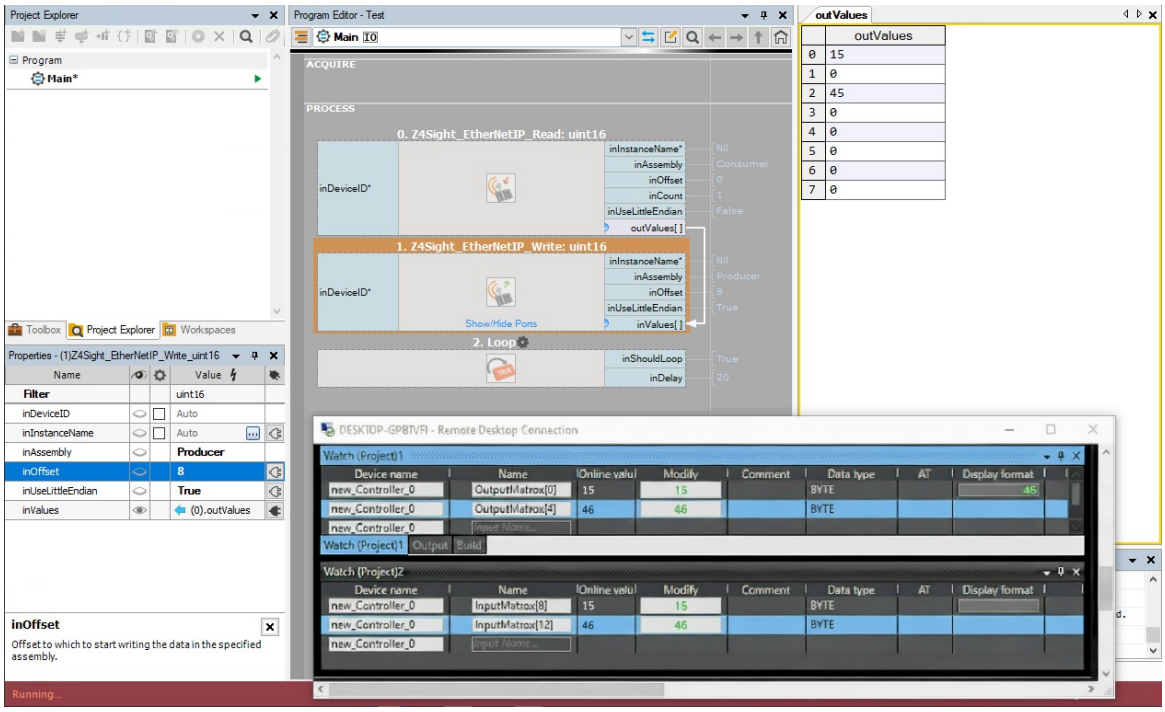


If communication with PLC is established you can download the configuration to the device.



Configuring Aurora Vision Studio application

Once communication between PLC and 4Sight EV6 is established, you can create program in Aurora Vision Studio. Add *watch tables* to easily modify data in Sysmac Studio:



Troubleshooting

In case of any problem with 4Sight configuration, please refer to the MIL X manuals that could be found under:

```
%MIL_Path%\..\DOC\help
```

More detailed information could be found in Zebra 4Sight Design Assistant help file.

Hardware Triggering Zebra CV60 Camera using Zebra 4Sight EV7 Auxiliary Outputs

1. Introduction

The aim of this application note is to demonstrate how to configure and use the auxiliary output signals of a vision controller to perform hardware triggering on an area scan camera. By the end of this guide, you will be able to set up an automated hardware trigger loop using the **Zebra 4Sight EV7** vision controller to capture images synchronously on a **Zebra CV60 Area Scan Camera**.

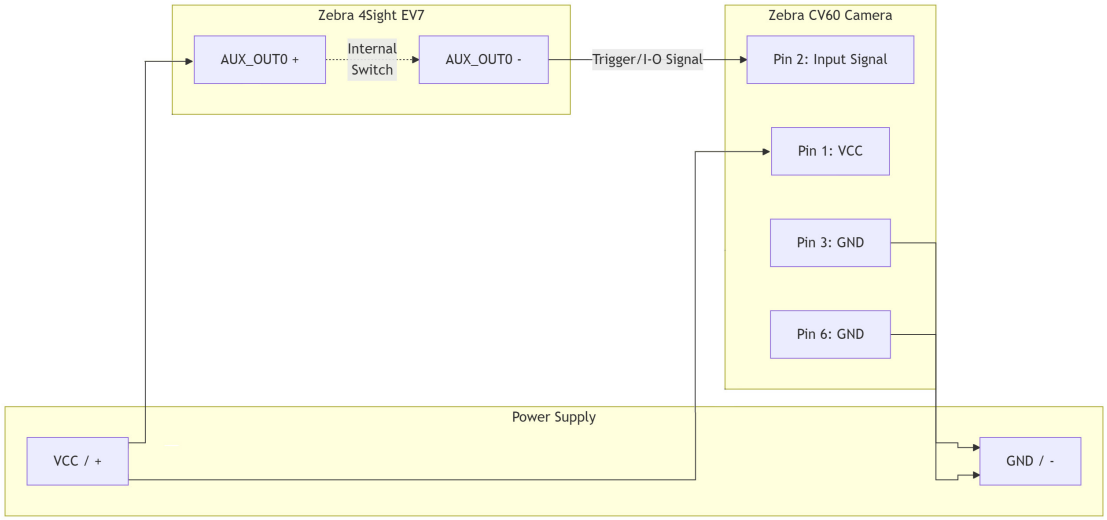
Required Hardware and Software:

- Zebra 4Sight EV7 Vision Controller
- Zebra CV60 Area Scan Camera
- 24V DC Power Supply
- Aurora Vision Studio software

2. Connecting the Devices

Before creating the software application, the hardware must be wired correctly. Based on the **Zebra 4Sight EV7 Product Reference Guide** and **CV60 Area Scan Camera** documentation, wire the devices as follows:

1. **Parallel VCC Connection:**
   - The Supply Voltage (VCC, 24V) is fed to **Pin 1 of the CV60 camera** (main power and I/O reference).
   - Simultaneously, VCC is connected to **AUX_OUT0+** on the EV7 controller.
2. **Signal Reception (CV60):**
   - The current sourced from AUX_OUT0− enters **Pin 2 of the CV60 camera**.
3. **Circuit Return (GND):**
   - **Pin 3 and Pin 6 of the CV60 camera** must be tied to the common power supply ground (GND). This completes the circuit for both the camera's main power and the opto-isolated input return path.



3. Creating the Application in Aurora Vision Studio

With the hardware wired, we will now create an application that periodically toggles the EV7 output to trigger the camera, while a separate background task handles the image acquisition.

Part A: Configuring the Output Signal (Main Task)

This section of the program will act as our trigger generator, toggling the output state.

1. In the Main macrofilter, add a Loop filter to the **ACQUIRE** section. Set the inDelay input to 1000 (milliseconds).
2. Add a AILSystem_SetOutputLevel filter in the **PROCESS** section.
   - Set the inDeviceID input to "Host" and "Default".
   - Set the inForceUserOutputSource input to True.
3. In the Project Explorer window, right-click on the Main macrofilter and choose the **"Add macrofilter register"** option. Name the new register "state" and set its type to Bool. Set its initial value to True.
4. Connect prev state from the Macrofilter Register Inputs block to the inLevel input of the **AILSystem_SetOutputLevel** filter.
5. Add a Not filter to the **PROCESS** section.
   - Connect its input to prev state from the Register Inputs.
   - Connect its output to next state on the Register Outputs block.

Part B: Configuring the Camera Acquisition (Worker Task)

Image grabbing must run independently of our 1-second trigger loop, so we will place it in a Worker Task.

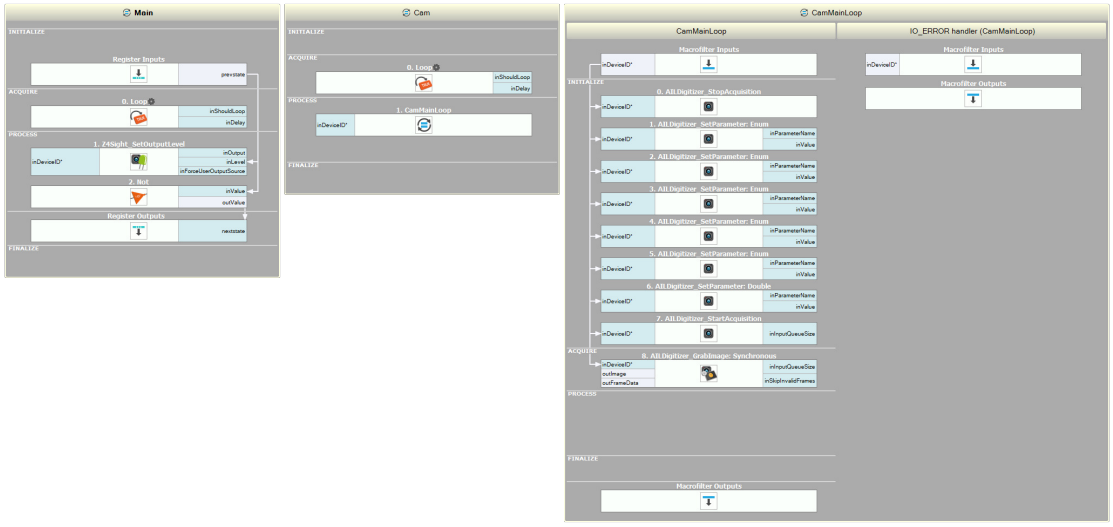
6. Create a new Worker Task macrofilter. Name it "Cam".
7. Inside the "Cam" worker task, add a Loop filter to the **ACQUIRE** section. Set the inDelay input to 0.
8. Create a new standard Step macrofilter and name it "CamMainLoop". Drag and drop this new task into the "Cam" worker task (inside the loop).
9. Inside the "CamMainLoop" task, add the AILDigitizer_GrabImage filter to the **ACQUIRE** section. Drag and drop its inDeviceID input to the Macrofilter Inputs block to expose it.
10. Go back to the "Cam" worker task. Select the "CamMainLoop" block and set the inDeviceID parameter values in the Properties window:
    - System: GigEVision
    - Device: DEV0
    - Digitizer: DEV0

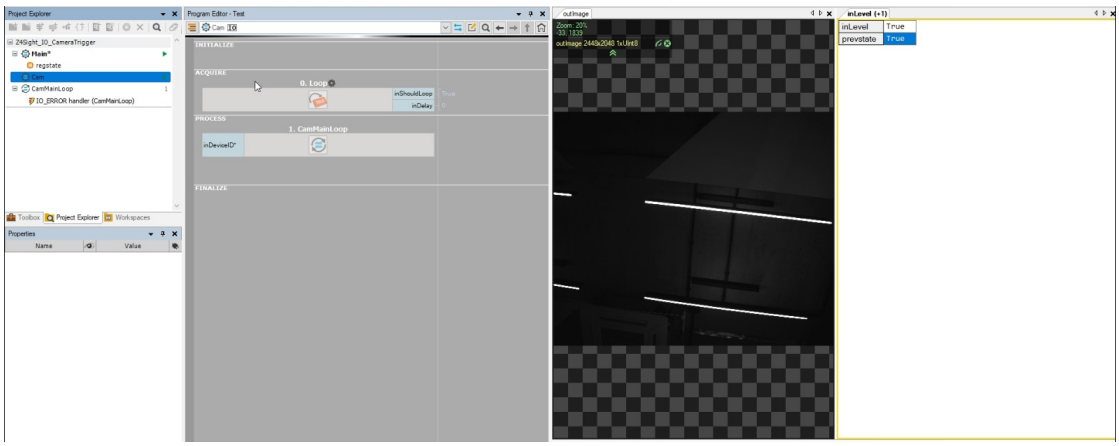
Part C: Initializing Camera Parameters

We must configure the camera's GenICam features to listen for the hardware trigger on the correct line.

11. Go back inside the "CamMainLoop" task. In the **INITIALIZE** section, add the following filters in order, and connect the inDeviceID input of each tool to the inDeviceID from the Macrofilter Inputs block:
    - 1x AILDigitizer_StopAcquisition
    - 7 instances of AILDigitizer_SetEnumParameter
    - 1x AILDigitizer_SetDoubleParameter
    - 1x AILDigitizer_StartAcquisition
12. Configure the seven AILDigitizer_SetEnumParameter filters as follows:
    - **First:** Parameter name: AcquisitionMode | Value: Continuous
    - **Second:** Parameter name: TriggerSelector | Value: FrameStart
    - **Third:** Parameter name: TriggerMode | Value: On
    - **Fourth:** Parameter name: TriggerSource | Value: Line5 *(Note: maps to physical Pin 2 on the CV60)*
    - **Fifth:** Parameter name: TriggerActivation | Value: RisingEdge
    - **Sixth:** Parameter name: ExposureMode | Value: Timed
    - **Seventh:** Parameter name: ExposureAuto | Value: Off
13. Configure the AILDigitizer_SetDoubleParameter filter:
    - Parameter name: ExposureTime | Value: 1000.0 (microseconds)

Now you can run the application. The system will alternate the EV7's auxiliary output state every second, generating a rising edge that hardware-triggers the CV60 camera to acquire a frame.



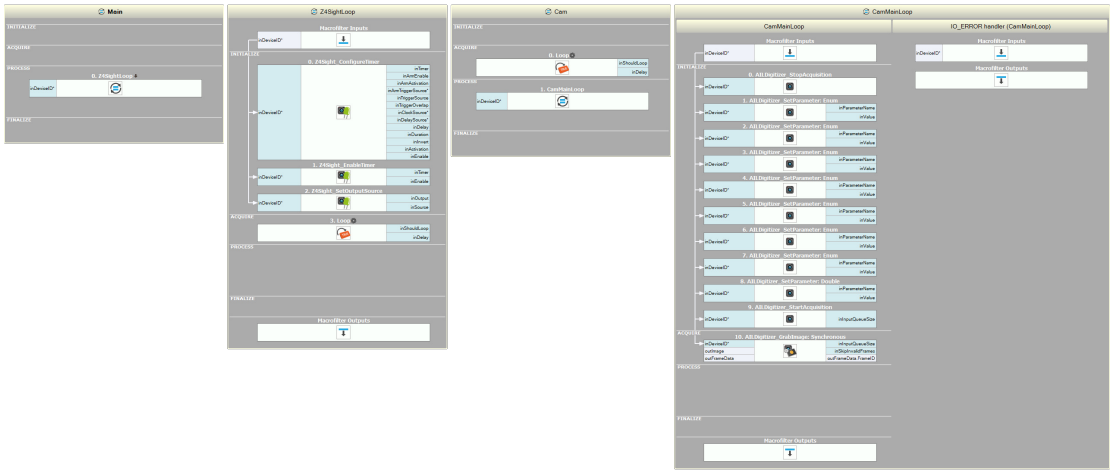


Part D: Using the EV7 Timer Mechanism (Alternative Method)

Instead of changing the output value by software as shown in Part A, you can use the **EV7 Timer mechanism**. To do that:

14. Delete all the filters inside the Main macrofilter.
15. Create a new task macrofilter and name it "Z4SightLoop". Add it to the **PROCESS** section of the Main macrofilter.
16. Inside the Z4SightLoop, add these filters to the **INITIALIZE** section: AILSystem_ConfigureTimer, AILSystem_EnableTimer, AILSystem_SetOutputSource.
17. Drag and drop the inDeviceID input of one of these tools to the Macrofilter Inputs block to create a new task input. Connect the new input to the rest of the Aurora Imaging Library filters.
18. Go back to the Main macrofilter. Set the inDeviceID input values of the Z4SightLoop:
    - System: Host
    - Device: Default
19. Go back to the Z4SightLoop.
20. For the AILSystem_ConfigureTimer filter, set the inDelay parameter to 5000000000.
21. For the AILSystem_SetOutputSource filter, set the inSource input to Timer1.
22. Add the Loop filter to the **ACQUIRE** section.

Now you can run the application.



4. References & Media

Hardware Documentation:

- Zebra CV60 Connector Diagrams
- Zebra 4Sight EV7 Product Reference Guide

Interacting with GigEVision cameras

Introduction

The following guide provides information on how to utilize a GigE-compliant camera in Aurora Vision Studio.

While the guide focuses on the GigE protocol most of the content can also apply with slight changes to different camera protocols, especially GenICam.

Basic Image Acquisition

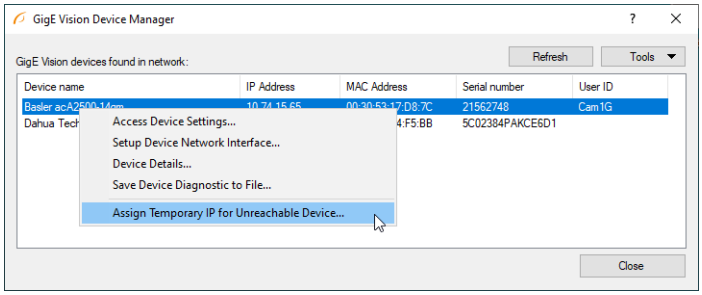
The simplest scenario, in which cameras are utilized in Aurora Vision Studio, is continuous synchronous acquisition.

Continuous means that the camera acquires new images by itself - it does not need to be triggered to do so.

Synchronous means that the program will wait for a new image until it is acquired - the iteration will not be finished without a new image.

The steps to perform such acquisition are as follows:

1. Connect the camera to the computer or network switch. If the camera is to be powered through Power-Over-Ethernet (PoE), make sure that it is connected to a device capable of supply power that way.
2. In a new program in Aurora Vision Studio add the filter **GigEVision_GrabImage**.
3. Open the editor of the inAddress input and set the camera address.
    a. If the camera is displayed with a warning sign, it means it is not in the computer's subnet. This can be solved by assigning a temporary IP address the camera.

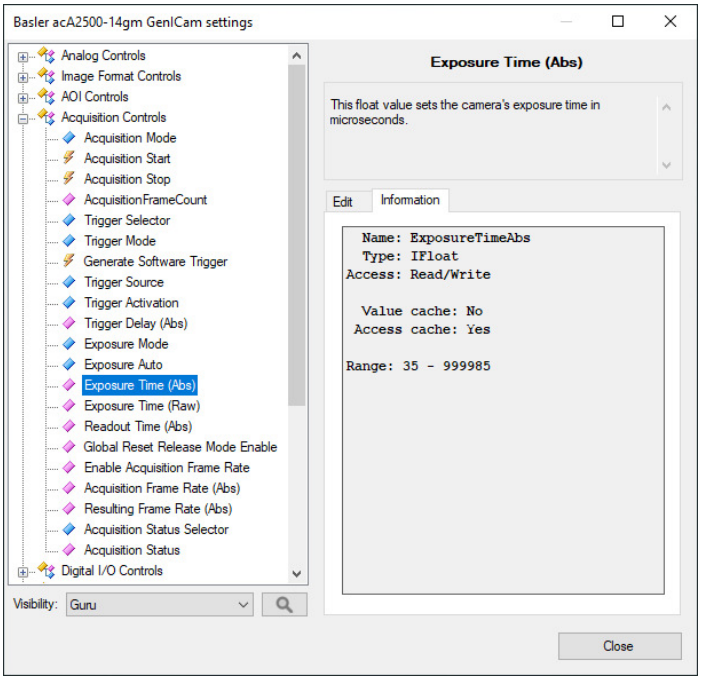


    b. When the camera is in the same subnet, it is possible to change its settings (for example changing its static IP).
    c. If the camera is connected but not detected you may open the window (from point a), manually type the camera MAC address and assign it another IP address. The menu can be opened from the Tools menu.
4. Now you can run the program. Previewing the outImage will show you the view from the camera.

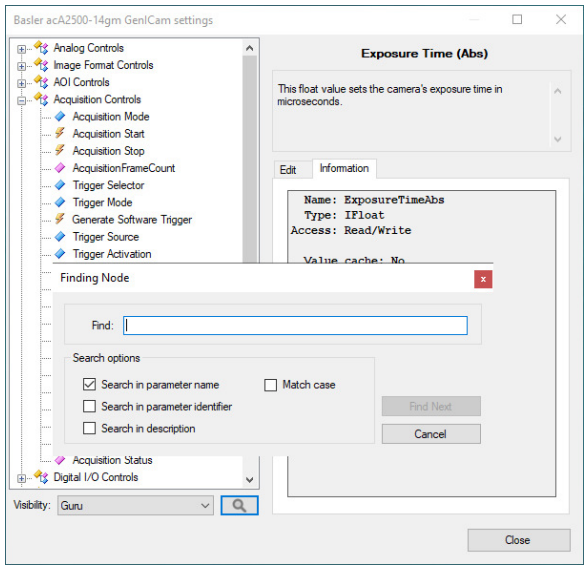
Changing camera parameters

You can change camera parameters programmatically using SetParameters filters. To demonstrate this, we will expand the previous program.

1. Add a **GigEVision_SetRealParameter** filter and set its **inAddress** to the same address as **GigEVision_GrabImage**.
2. Specify the parameter name. If you are not sure about the name you can select through the GigEVision device tree. To do that click on the "..." button near the **inParameterName**.
3. Here you can see all available camera parameters with a short description. Select the parameter with name Exposure Time (Abs) (or similar if not present). As you can see it controls the camera exposure time in microseconds and its type is IFloat.



   a. If you cannot find the parameter try using the search function (the magnifying glass icon)



   b. The type of the filter needs to match the type of the parameter. The exception is that IFloat is represented as Real in Aurora Vision Studio.
   c. Some more advanced parameters may not be visible unless Visibility is set to the correct level.
4. Make sure that the parameter ExposureAuto is set to Off. Otherwise, it will not be possible to manually change the exposure value.
5. Before closing the window note the minimum and maximum values of exposure time. After selecting the parameter, set the **inValue** to the lowest acceptable value.
6. Run the program. While running, steadily increase the **inValue** input. You will notice that the camera image gets brighter and brighter.
   a. Try not to go outside the acceptable range. It will result in an error if the input **inVerify** is set to True.
7. It is also possible to check a parameter's value by using **GigEVision_GetParameter** filters. Try adding one in the Float variant and specifying the same name as in the previous filter.
   a. You many notice that not always the read value is equal to **inValue** of SetParameter. It is because the camera modifies it.

With **GigEVision_GetParameter** filters it is not only possible to check editable parameters but also to check read-only ones. For example, if the camera was set to automatically adjust exposure, the current exposure time might be useful to know. Another example would be to read parameters holding device-specific information, such as the maximum image size.

For some quick testing you may also want to set parameters directly in the GigEVision tree.

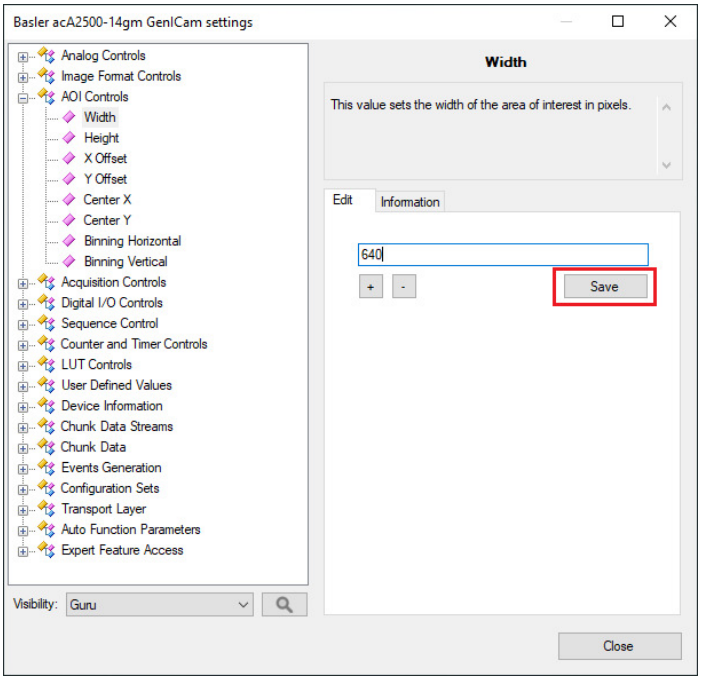
It is important to note that while most parameters are stored in volatile memory and as such will reset after unplugging the camera from power. There are some parameters that are stored in non-volatile memory, such as Device User ID, but they usually do not affect the acquisition directly.

Because of this, it is recommended to design the program in such a way that every parameter with a value different from the default is set programmatically.

Starting and stopping acquisition

It is important to note that not all parameters can be changed while the acquisition is running. Some require stopping the acquisition and restarting it.

1. Run the program and open the device tree (Tools -> Manage GigE Vision Devices...). Find the Width parameter. You will notice that the save button is grayed out due to running acquisition.



2. Add another **GigEVision_SetIntegerParameter** filter and select the Width parameter. You may notice that it is greyed out.
    a. While selecting the parameter check the information tab. There may be additional information about possible value, e.g. increment. Some parameters can only take value that are multiples of certain numbers, like 2 or 4.
3. Enter 240 as inValue and run the program. You will encounter an error saying that the parameter is not writeable.
4. Now add a **GigEVision_StopAcquisition** before the previous Set filter and rerun the program.
5. The parameter is now set without errors and the output image is smaller.

As you can see the Width parameter was not writable while the acquisition was running. Stopping the acquisition before changing the parameter made it writable.

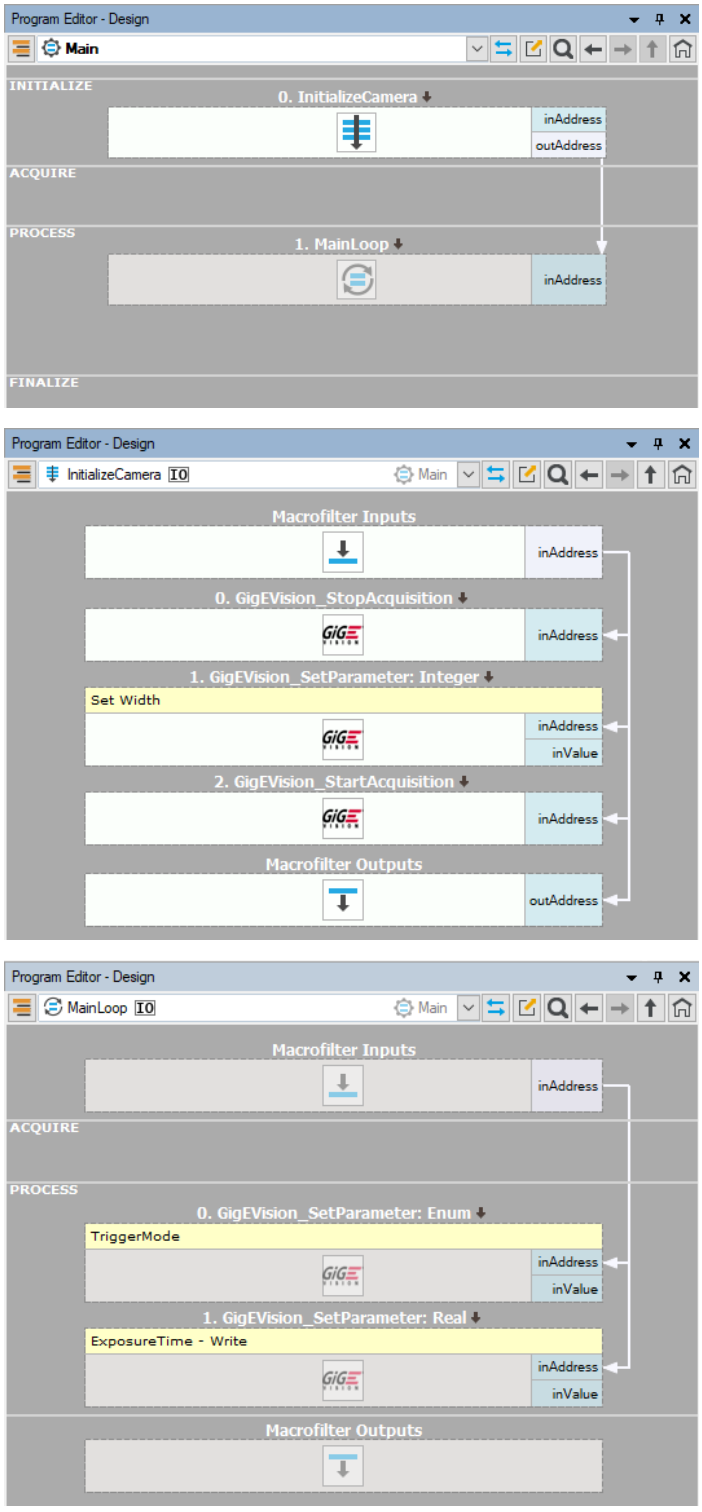
You may also notice that we have a filter to explicitly stop the acquisition but there is not filter to explicitly start it. **GigEVision_GrabImage** will attempt use an ongoing acquisition but if none is present it will start a new one.

While it is acceptable for simple programs, it should be avoided when the status of the acquisition is program-controlled.

Now we will make a proper program out of this.

1. Start with removing the filters added in this chapter.
2. Now select the rest and extract a task macrofilter (called MainLoop) from them. This will be the main acquisition loop.
3. Create a new step macrofilter called InitializeCamera and add it before MainLoop in INITIALIZE section.
4. In InitializeCamera add the three following filters **GigEVision_StopAcquisition**, **GigEVision_SetIntegerParameter** (select Width parameter) and **GigEVision_StartAcquisition**.
    a. Set the width to the maximum possible value.
5. Drag **inAddress** of any of the filter to the top bar to create a new input. Then add an output of the same type and connect it to the input.
6. Go up one level and drag the **outAddress** output to the MainLoop filter creating a new input.

7. Inside that filter connect the newly created input to all the GigE filters
    a. Now every GigE filter in the whole program shares the same camera address.
    b. The final program should look like this:



Now the program is divided into two parts. The first part is executed once, after that the program enters MainLoop which run continuously until the program is stopped.

The parameters that can be changed during acquisition are set in the InitializeCamera filter where the acquisition is guaranteed to be stopped.

All the other parameters are set in MainLoop.

In real applications not all parameters need to be changed during runtime. Even then they probably will only be changed from time to time, not in every iteration.

For instance, specifying if the camera will run continuously in trigger mode will likely be done only once (even though it can be set during acquisition). Exposure time might be changed multiple times.

It is good practice to have all the parameters that are set only once in one macrofilter (like InitializeCamera) regardless of if they required acquisition to be stopped.

The parameters changed from time to time should be set in a variant macrofilter.

The reason of that is time-saving. Setting the parameter to the same value does not take much time (Aurora Vision Studio caches previous values and avoids resending the same ones), but if we have a lot of parameters it adds up. It is also more intuitive if one-off parameters are in a dedicated filter.

It is important to note how the cameras work in Aurora Vision Studio. When the acquisition is started Aurora Vision Studio creates a background thread which buffers incoming frames in memory. This thread persists until the acquisition is stopped or the application exits the task that started the acquisition.

For example, if InitializeCamera was a task and not a step macrofilter the acquisition would stop when exiting it and it would have to be restarted in MainLoop.

Also, instead of passing the camera address as a parameter it is possible to put it into a global parameter.

Changing trigger mode – basic

While the default mode of most cameras is continuous acquisition very often it is more desirable to only acquire new images after a certain signal, i.e. trigger.

Not only you have more control over the acquisition time of each frame, less frequent acquisition requires smaller bandwidth.

To demonstrate triggered acquisition, we will use the current program.

Run the program, ensure that you have displayed the camera image in the preview.

1. Now open the GigeVision device tree and find the parameter Trigger Mode. Set it to On. Also make sure that Trigger Selector is set to Frame Start.
2. You will notice that no new images are grabbed by the camera.
3. Find the Trigger Source parameter and set it to Software.
4. Find the Generate Software Trigger command (commands feature the lightning bolt icon) and execute it.
5. A new image will be grabbed by the camera.

You may also want to have an external trigger (for example, an output from a PLC). Then you must select the appropriate Trigger Source, e.g. Line1.

You may also have to ensure that the signal matches what the camera expects. The Trigger Activation parameter specifies what value will the camera look for.

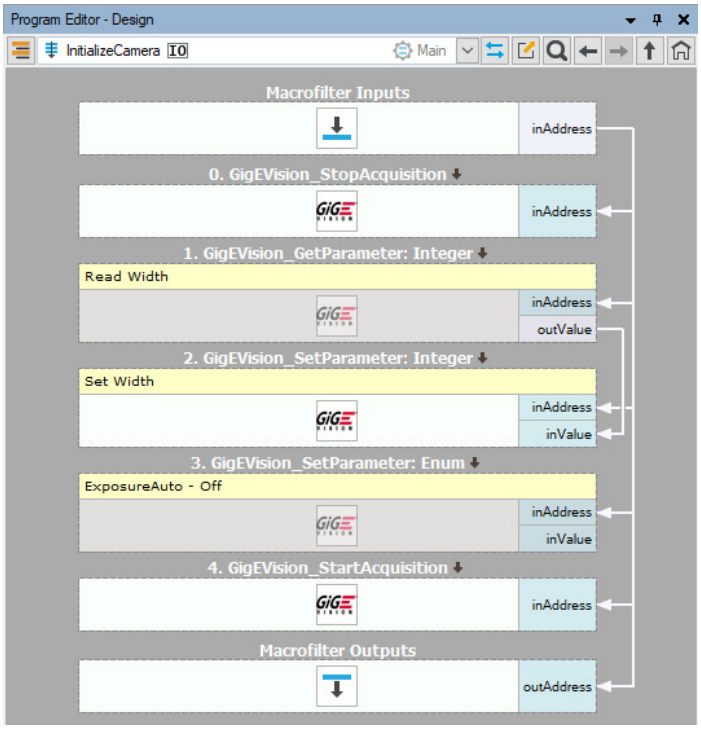
Changing trigger mode – advanced

Very often we want the trigger properties to be controlled directly by the program. For example, the camera may be triggered in one part of the program but also acquire images continuously in another part. Or maybe the camera is to be triggered by different sources depending on the program state.

Trigger parameters can be changed during acquisition. Therefore, it is possible to change them on-the-fly.

Now we will prepare a more advanced program which initializes the camera and allows the user to change how the camera is triggered and the exposure time from the HMI while the program is running.

1. To ensure that the camera will be set to use Software as the source of trigger, add a **GigEVision_SetEnumParameter** filter in the InitializeCamera (before **GigEVision_StopAcquisition**). Through the Device Manager choose the TriggerSource parameter and set **inValue** to Software.
2. Now repeat the previous step, changing the parameter to ExposureAuto and set its value to Off.
3. Since different cameras may have different acceptable maximum width you may want to add a **GigEVision_GetIntegerParameter** filter that would read the parameter WidthMax in device information. Connect its output to the filter that changes the Width.

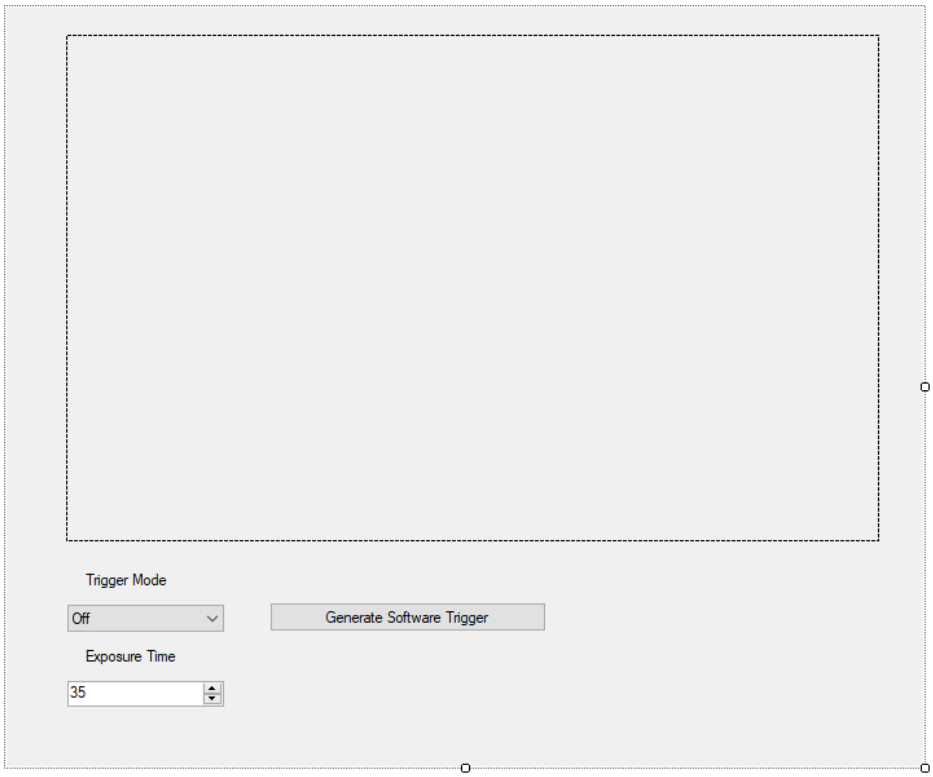


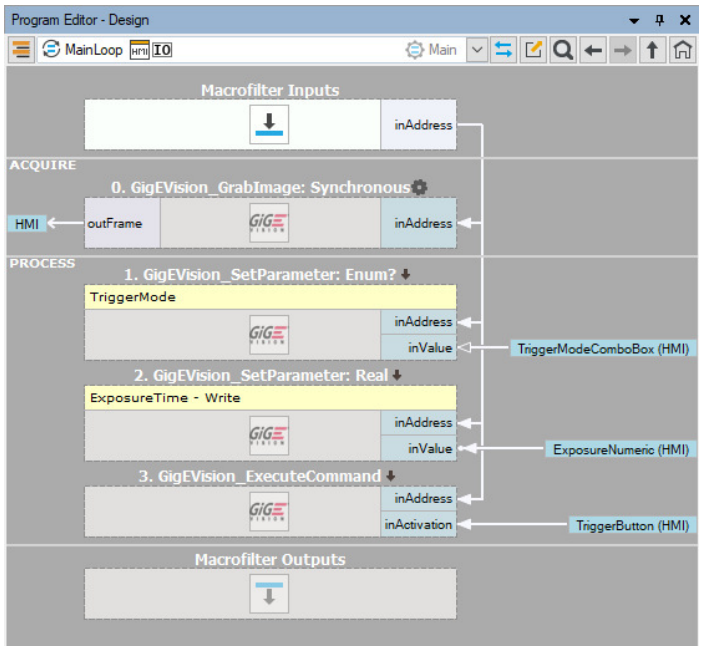
4. Now we will modify the contents of MainLoop worker task. We want to be able to change trigger mode and to execute a software trigger.
    a. Add an instance of **GigEVision_SetEnumParameter**. Connect it to the camera address and select Trigger Mode as its parameter
    b. Add a **GigEVision_ExecuteCommand** filter, connect them to the camera address. Through Device Manager select parameter TriggerSoftware.
    c. Move all those filters to PROCESS section and add **GigEVision_GrabImage** to ACQUIRE section.
5. Let's design an HMI. In this case it will feature:
    a. View2DBox – which will display the camera image;
    b. ComboBox – for selecting trigger mode;
    c. NumericUpDown – to control exposure time;
    d. ImpulseButton – to generate software trigger;
    e. Labels – to label other controls.
6. Now we will configure the controls:
    a. View2DBox – change its InitialSizeMode to FitToWindow and connect to GrabImage's outImage;
    b. ComboBox (for trigger mode)
        I. Connect **outText** to **inValue** of Set Parameter for TriggerMode;
        II. Expand List in Data category in Properties and add the following item: On, Off;
        III. Set Selection to 0
    c. ImpulseButton – connect its **outValue** to ExecuteCommand's **inValue**;

d. NumericUpDown:

   I. Connect **outValue** to **inValue** of SetParameter for ExposureTime;

   II. Set Minimum and Maximum to the values specified in the Device Tree for that parameter (for the camera used while writing this note the values were 35 and 999985.





7. The program should display the camera image on the HMI. You should be able to change the exposure time. However, when you change the ComboBox's value to On, the program will freeze. Clicking the trigger button will not do anything.

   a. This is caused by the fact that GrabImage is in its synchronous variant. The program cannot get past GrabImage and execute a trigger, so it waits for an image indefinitely.

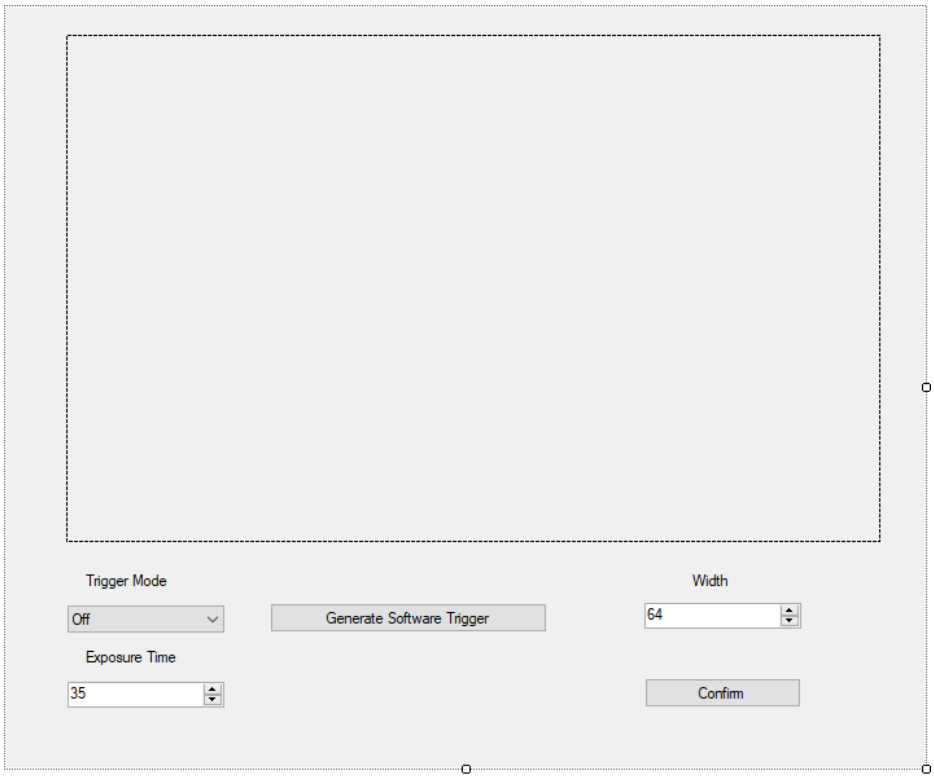
   b. To solve that we can change **GigEVision_GrabImage** variant to **GigEVision_GrabImage_WithTimeout**. Now the program will only attempt grabbing for a specified amount of time. Let's set **inTimeout** to 100ms and rerun the program.

8. Now the program does not freeze when setting triggered mode on. If no image is grabbed in 100ms GrabImage returns Nil and the program proceeds to the next iteration, where the camera may be triggered.

   a. Generally, if the camera is meant to acquire images only from time to time, it is a good idea to use the WithTimeout variant of GrabImage. It prevents program hang-ups in case of camera problems and allows you to inform the user that the camera may not be working correctly. However, the rest of the program must be designed in a way that handles Nil images properly.

We can now expand the program to enable the user to change other parameters, including those which require acquisition to be off.

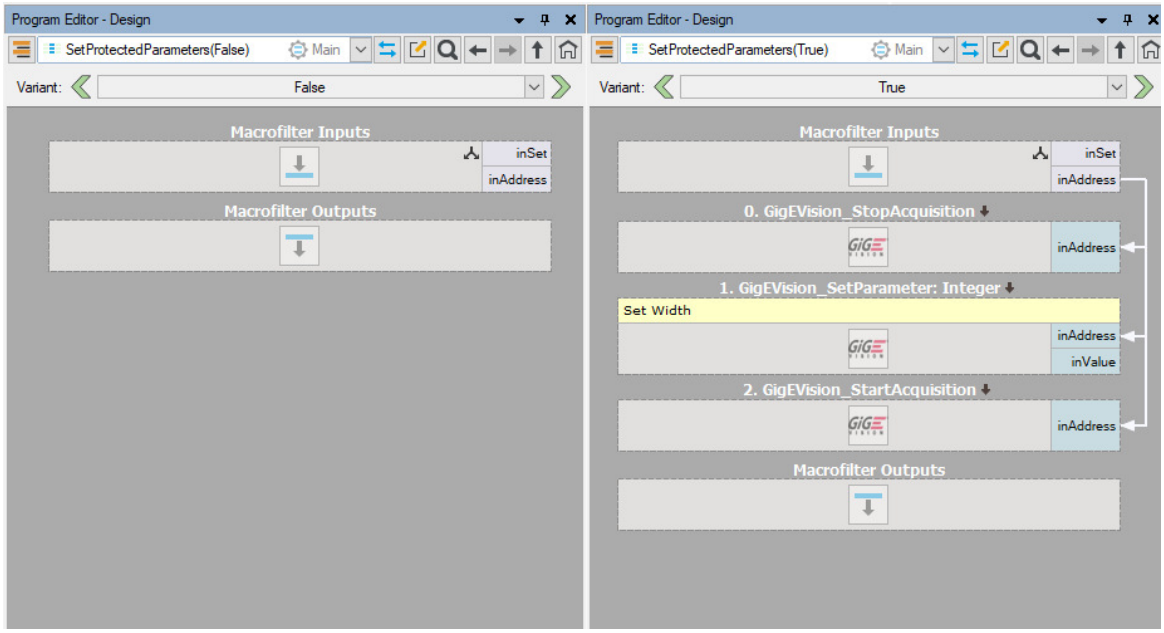
1. Create an empty variant macrofilter inside MainLoop and choose Bool as the fork type. Drag the camera address to the filter to create an input.
2. Enter the filter and choose the variant True. Add a StopAcquisition filter, followed by a SetParameter filter in variant Integer and select Width parameter. After them add a StartAcquisition filter. Connect all filters to the camera address.
3. In the HMI add two new controls:
    a. NumericUpDown – to control width's value; set its Minimum and Maximum to the respective limits of the Width parameter in the camera (for the camera used while writing this note the values were 35 and 999985); set Increment to the respective value as well.
    b. ImpulseButton – to confirm new value.

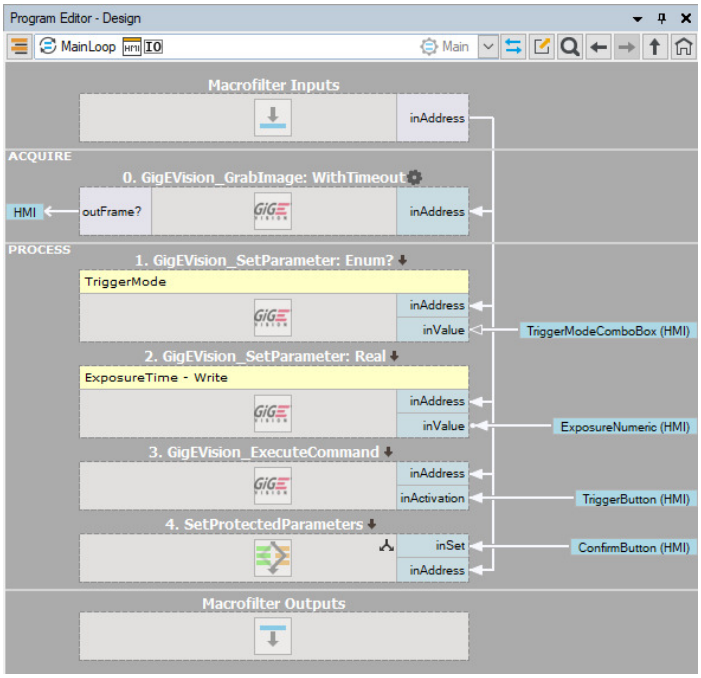


4. Connect NumericUpDown's outValue to the new SetParameter's inValue.
5. Go back to MainLoop and connect the new ImpulseButton's outValue to the condition port of the variant macrofilter.

Now when the user clicks the confirm button the program will enter the variant macro, stop the acquisition, change the width and restart the acquisition. If the user changes the value by using NumericUpDown buttons the value will be properly aligned.

However, if the user enters a value by typing, it may not be guaranteed to work with the camera. It is possible to design a macrofilter which will make sure the value is accepted by the camera, but this is not the topic of this application note.





Additional remarks

Camera filters cannot be executed in array mode (for example when an array of camera addresses is connected).

Since a camera is an external hardware, they may be causing errors (like connection errors) unrelated to the Aurora Vision application. To properly handle those errors the application should feature error handling.

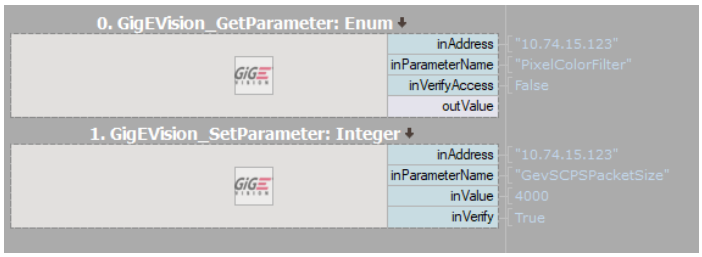
Changing parameters of GigEVision cameras

Introduction

The following guide provides information on how to programmatically choose and connect to a camera and modify its parameters. The modification of the acquisition parameters may involve restarting the acquisition. By implementing an interface to trigger the acquisition by software the program also shows how to execute camera commands. All the functionalities are available to the user through an HMI.

How to interact with camera

In Aurora Vision Studio users can interact with a GigE camera through various filters. Those allow the user to change some of the camera's parameters or access acquisition data. Filters **GigEVision_SetParameter** and **GigEVision_GetParameter** can be used to read and write parameter's values. Their common inputs include camera address and name of the parameter. The last common input – **inVerify** is used to determine if the program should check the validity of parameter before write or read.



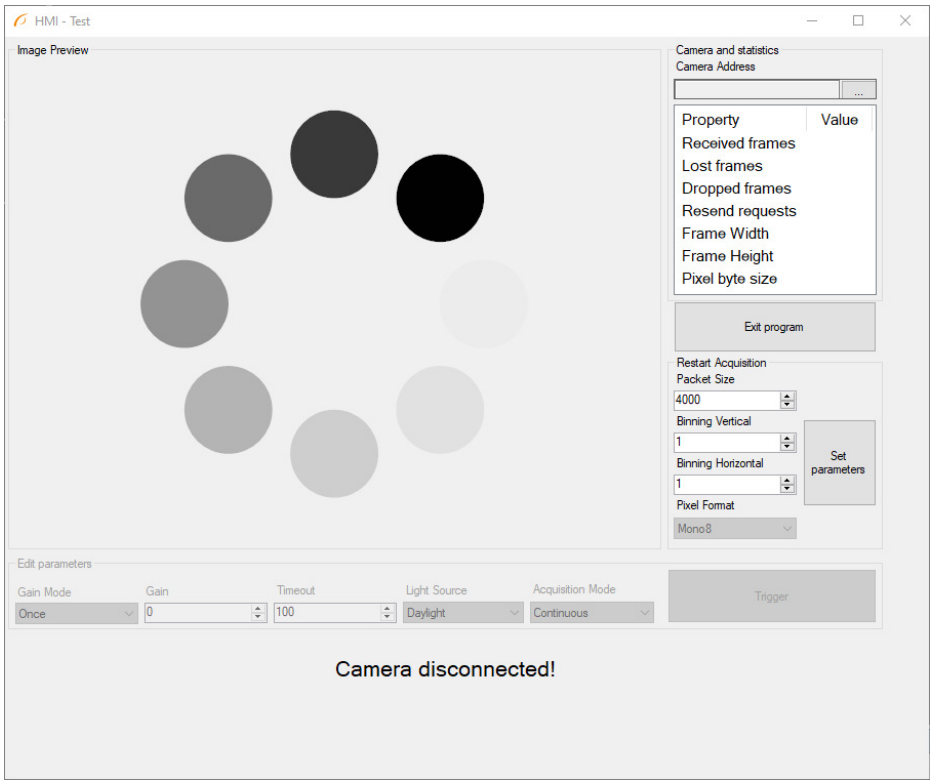
GigEVision_SetParameter has one more input, where the user specifies the value to be written. In **GigEVision_GetParameter** there is an output with the value of the parameter.

Both filters have variants depending on the type of the parameter being accessed – Real, Integer, Bool, Enum and String.

To see what the possible values for a given parameter are, the user may just click on **inParameterName** which shows a window with available parameters for the currently connected camera.

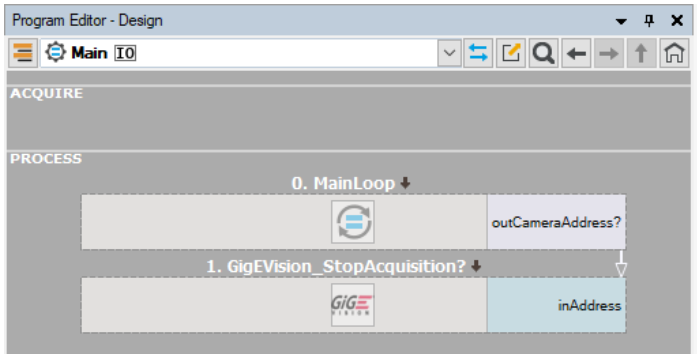
Program overview

The designed program allows the user to select a camera from a list of connected devices. After selecting one, the acquisition starts. The program displays a preview of the camera images. The user may modify selected camera parameters like gain, light source correction, and acquisition mode. If the camera is set to triggered acquisition mode, the user can release a software trigger. While running the program displays information about the current acquisition.



Under the preview there are two groups of controls. The controls on the left allow changing the parameters that require a restart of acquisition. Because of that there is also a button which lets the user restart the acquisition with new parameters.

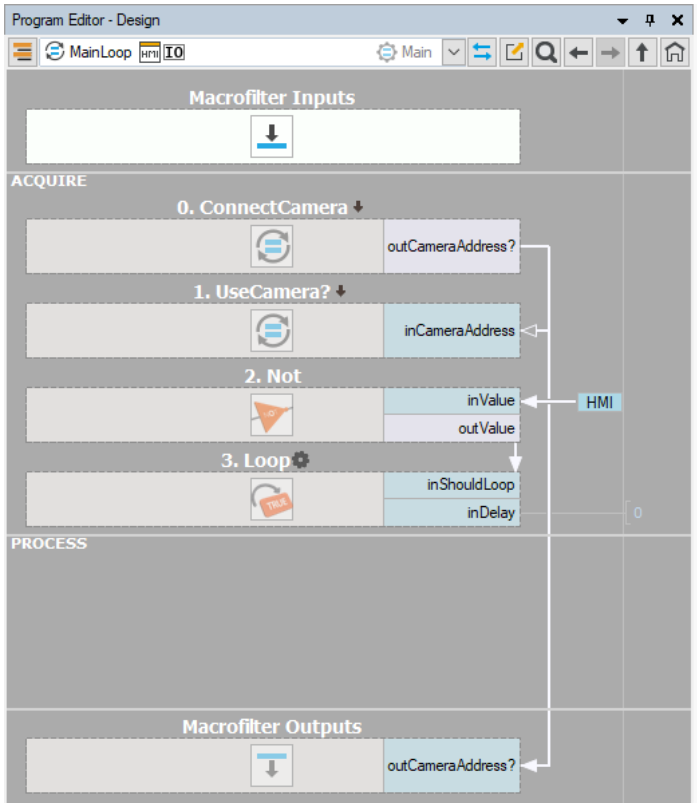
The other group has controls for the parameters that can be changed during the acquisition. The button labeled "Trigger" is used to start the acquisition using a software trigger. It is disabled when in continuous mode.



The *MainLoop* filter of the program can be divided into 3 parts. The first part is connecting the camera – here it is done by the *ConnectCamera* task macrofilter.

The second and the third part are done by the *UseCamera* task macrofilter. First the program starts the acquisition with set parameters. Then the program continuously acquires images while being able to change some of the camera parameters.

The **Not** and **Loop** filters control whether *MainLoop* will continue. The Not filter is connected to the "Exit program" HMI button. If the button has been pressed, the loop will not continue – the program goes back to *Main* and stops the acquisition (if a camera had been connected), which finishes the program.



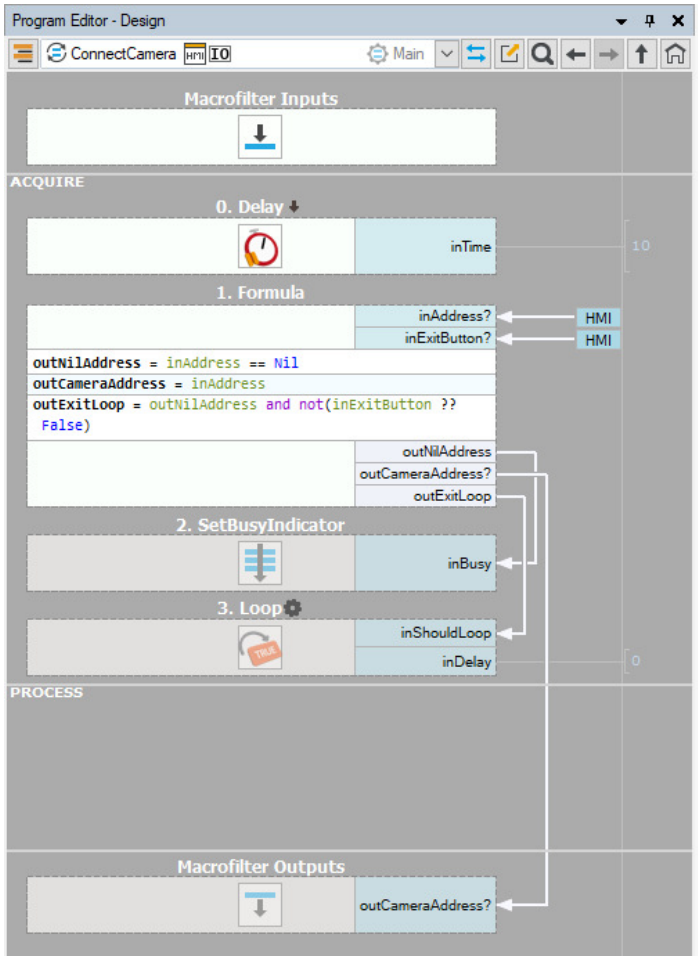
Connecting the camera

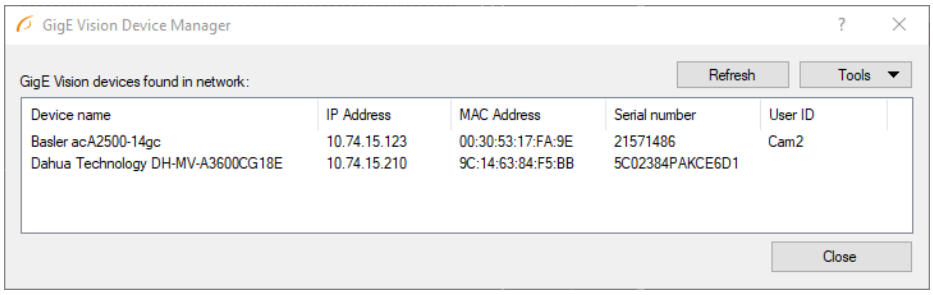
The first part of the program allows the user to connect a camera. The delay in the filter is used to limit the number of iterations per second. The **inAddress** input of the formula is connected to the HMI control labeled *Camera Address*.

After clicking the 3-dot button, a window opens letting the user select the camera. The selected camera's address is then sent to the formula block.

The animated waiting indicator is visible only if no camera has been chosen (so the address is Nil). The state of the indicator is controlled by the **outNilAddress** output of the formula.

If the user chooses a camera and its address is no longer Nil this exits the loop of this macrofilter and the camera address is outputted.





Setting the parameters

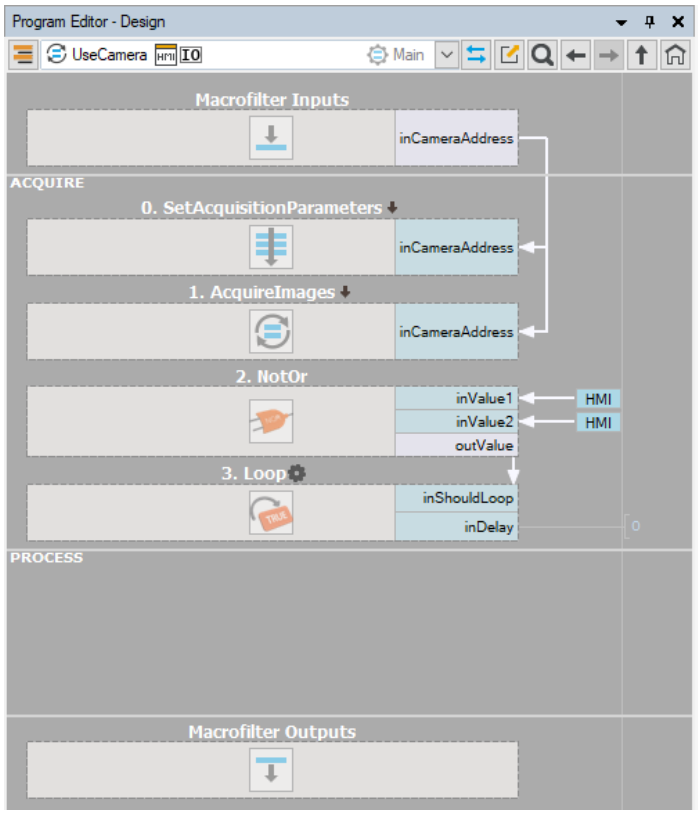
After the camera has been connected the next step is to start the acquisition. However, some acquisition parameters can only be set while not acquiring images. Such parameters include binning, packet size and pixel format. They are set in the *SetAcquisitionParameters* step macrofilter

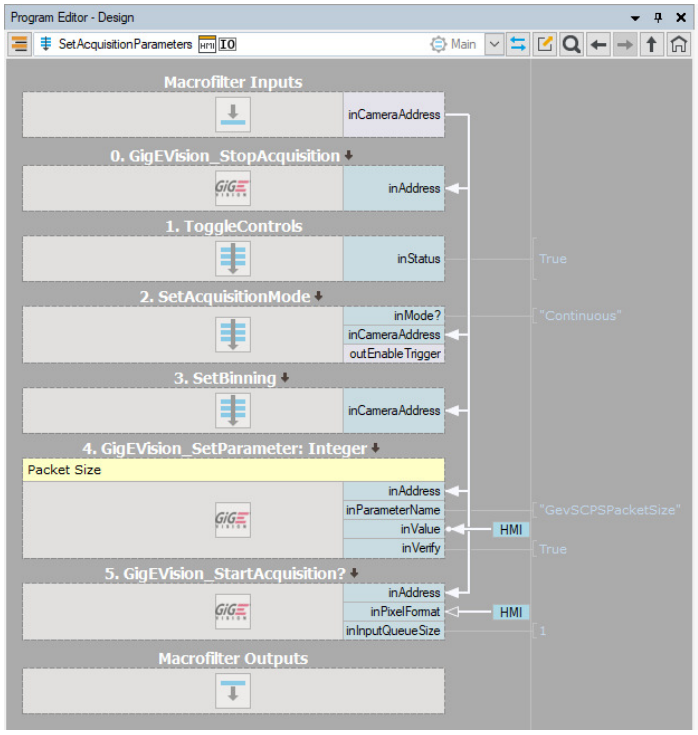
The **NotOr** filter negates the sum of two boolean values related to HMI controls. The first being **outAddressChanged** of the camera address picker and the second – whether the "Exit program" button has been pressed. If both are false the value of **NotOr** is true, which enables the loop to continue. If at least one of them is true – the program exits the *UseCamera* loop and goes back to *MainLoop*.

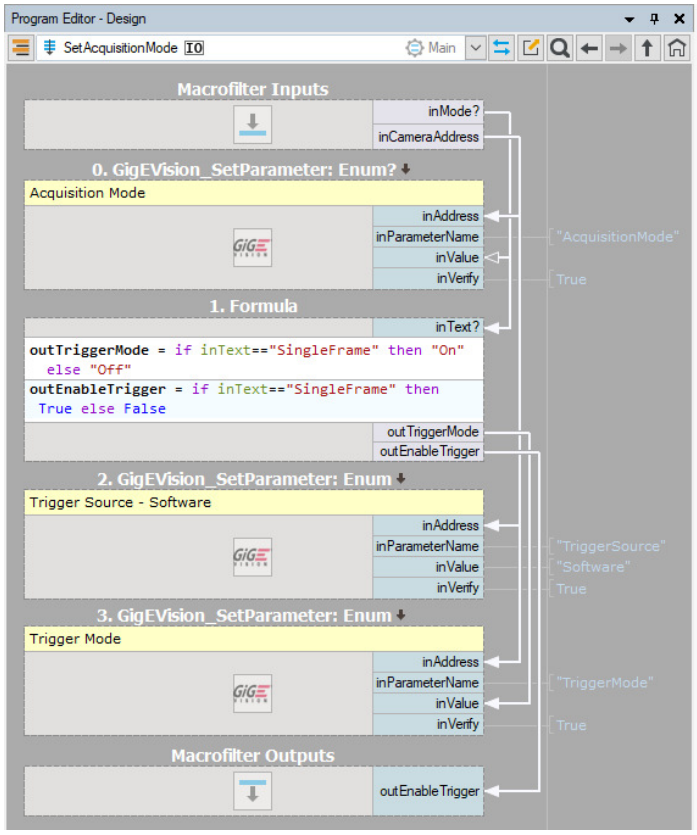
First any undergoing acquisition is stopped to ensure the parameters can be set. After that the mode of acquisition is set to "Continuous". This allows the camera to grab images to show in the preview of the program. The formula sets **outTriggerMode** to "Off" if the **inMode** parameter is anything other than "Single Frame".

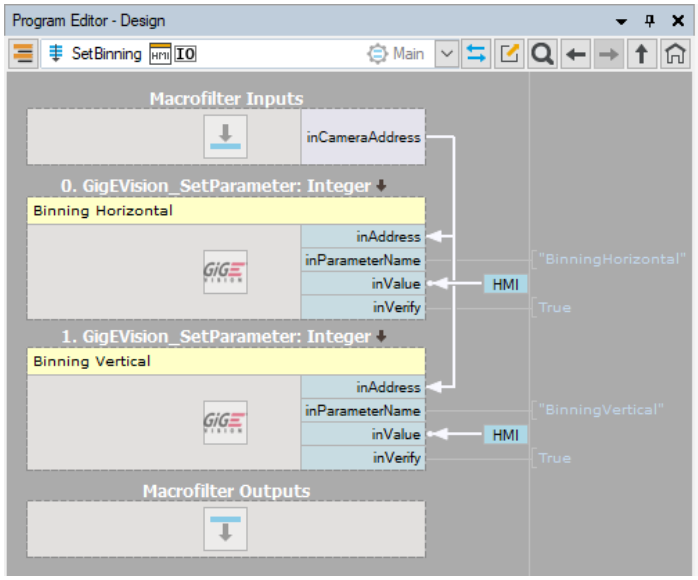
The next steps *SetAcquisitionParameters* set the binning of the camera (in both the horizontal and vertical axes) as well as the packet size. The possible values of the HMI controls used should be limited to be compatible with the camera value range. For example, the packet size here ranges from 220 to 16404 in increments of 4.

Finally, the program starts the acquisition with the pixel format chosen by the user.


Program Editor - Design
UseCamera
Main
Macrofilter Inputs
inCameraAddress
ACQUIRE
0. SetAcquisitionParameters
inCameraAddress
1. AcquireImages
inCameraAddress
2. NotOr
inValue1
HMI
inValue2
HMI
outValue
3. Loop
inShouldLoop
inDelay
0
PROCESS
Macrofilter Outputs

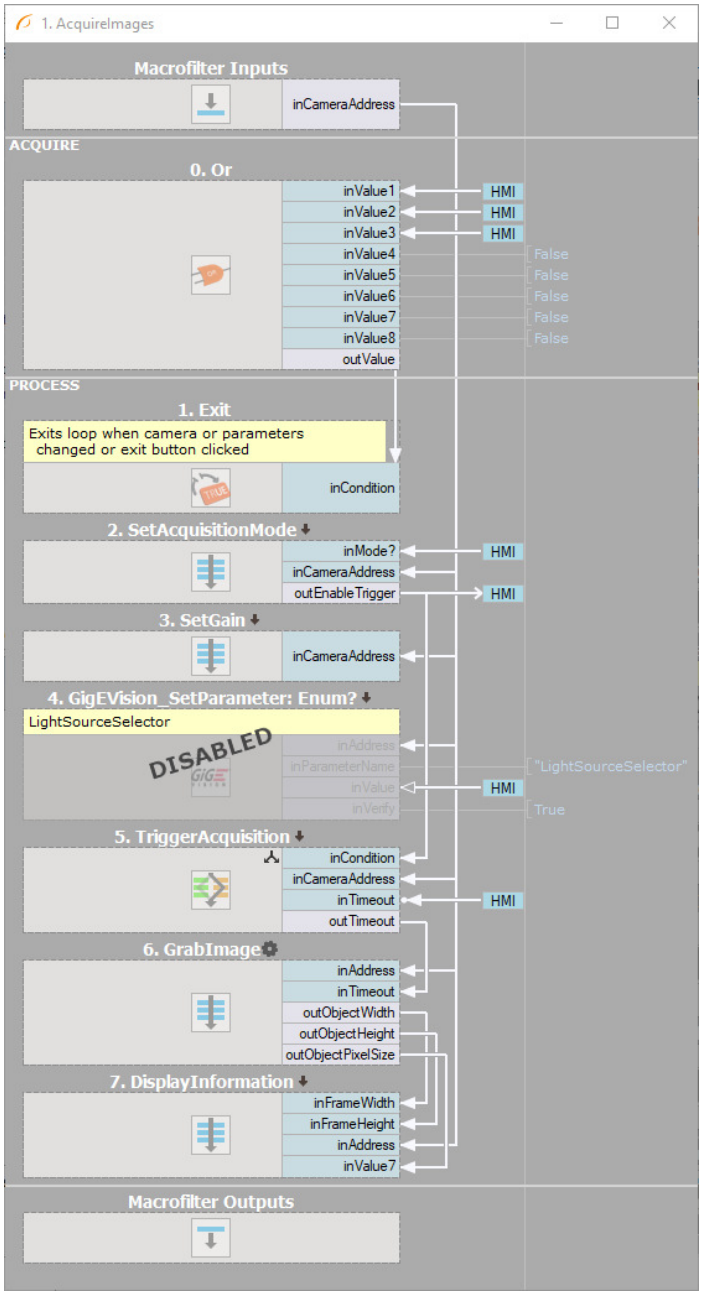

Program Editor - Design
SetAcquisitionParameters
Main
Macrofilter Inputs
inCameraAddress
0. GigEVision_StopAcquisition
inAddress
1. ToggleControls
inStatus
True
2. SetAcquisitionMode
inMode?
"Continuous"
inCameraAddress
outEnableTrigger
3. SetBinning
inCameraAddress
4. GigEVision_SetParameter: Integer
Packet Size
inAddress
inParameterName
"GevSCPSPacketSize"
inValue
HMI
inVerify
True
5. GigEVision_StartAcquisition?
inAddress
inPixelFormat
HMI
inInputQueueSize
1
Macrofilter Outputs





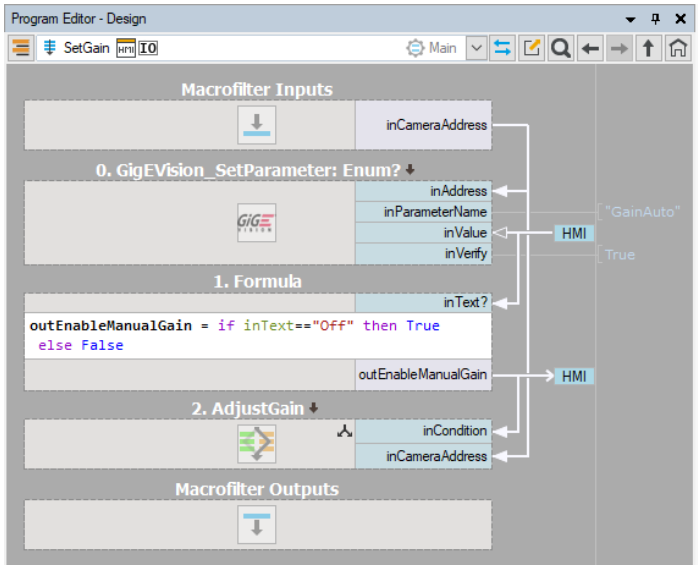
Acquiring images

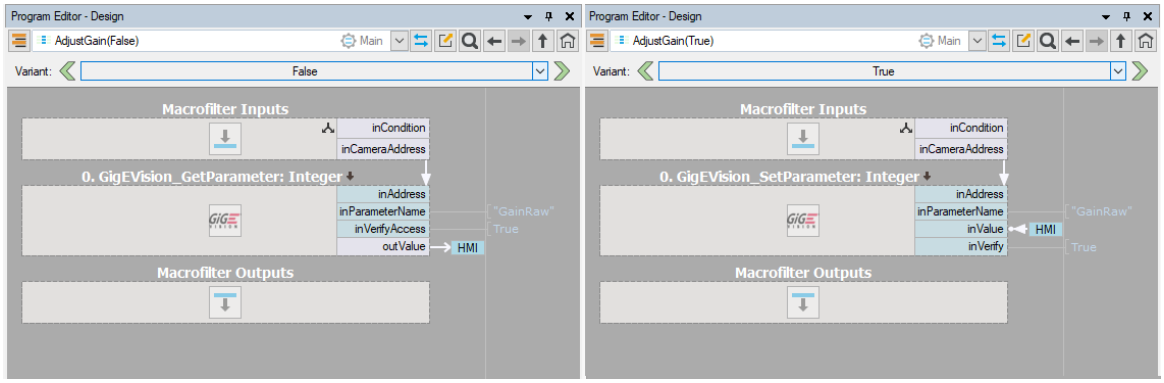
Acquiring images takes place in a loop in the filter *AcquireImages*. It begins with an **Or** filter that sums the states of three HMI controls: "Exit program" button, **outAddressChanged** of the camera address picker and "Set parameters" button. If any of them is true, the program exits the loop and goes back to *UseCamera*.



After that there is another instance of *SetAcquisitionMode*. This time the **inMode** parameter is connected to the appropriate control allowing the user to switch between continuous and triggered acquisition.

SetGain allows the user to switch between different modes of gain adjustment as well as set its value manually. First the program sets whether the gain will be adjusted manually or not based on the value from the control. If it is, the formula below sets the state for the next filter as well as enables (or disables) the gain value control in the HMI.



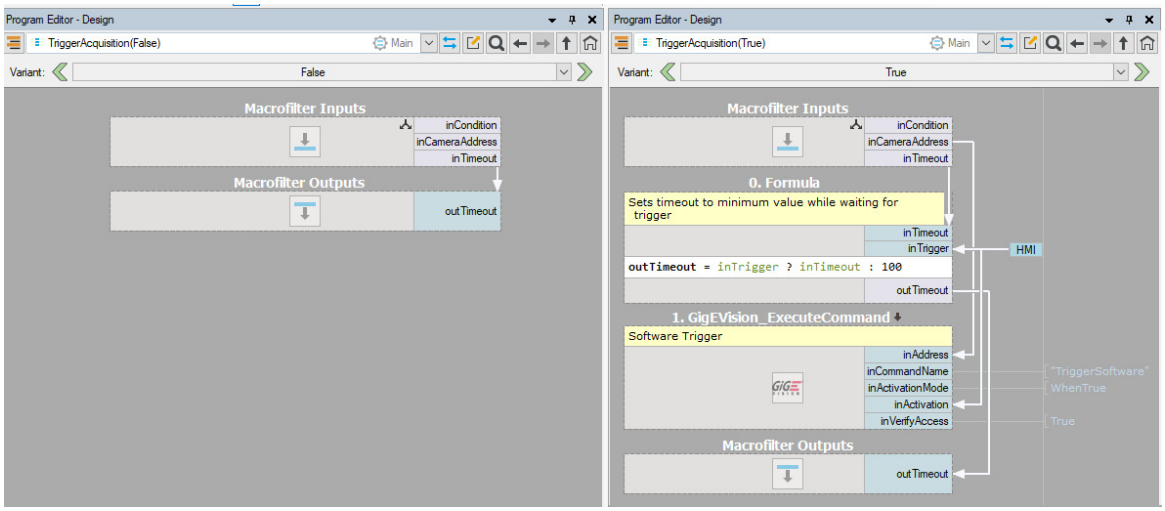


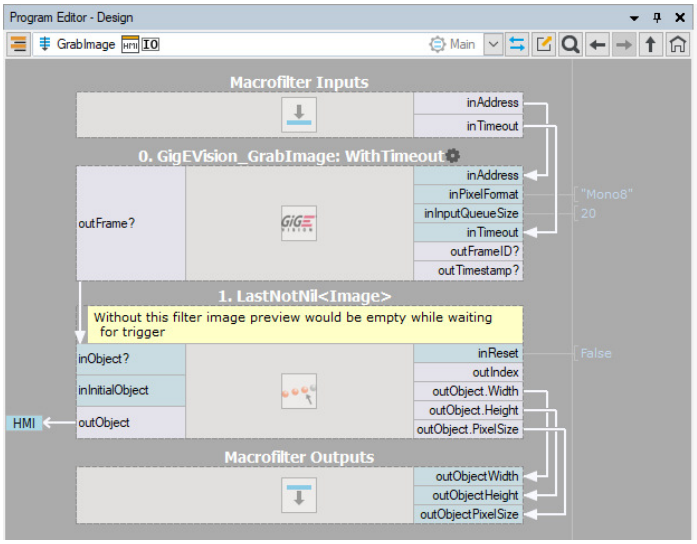
AdjustGain is a variant macrofilter. If gain is to be adjusted manually it sets the parameter to the given value. If gain is controlled automatically, it instead reads the value. This enables the user to see the automatically adjusted gain value in real time.

Next the light source is set. It allows the user to select the preset best suited to the current lighting conditions and by extension – to make colors look more natural.

TriggerAcquisition is another variant macrofilter. The **inTimeout** parameter sets how long should the program wait for an image from the camera. The variant for continuous acquisition is almost empty – it only passes **inTimeout** to **outTimeout**.

If the acquisition is triggered the program first check the state of the "Trigger" button. If it had been pressed the program executes the trigger command and passes **inTimeout** to **outTimeout**. If the button had not been pressed no command is executed and timeout is set to its lowest possible value – 100. Low timeout makes program more responsive when the user does not trigger the acquisition.

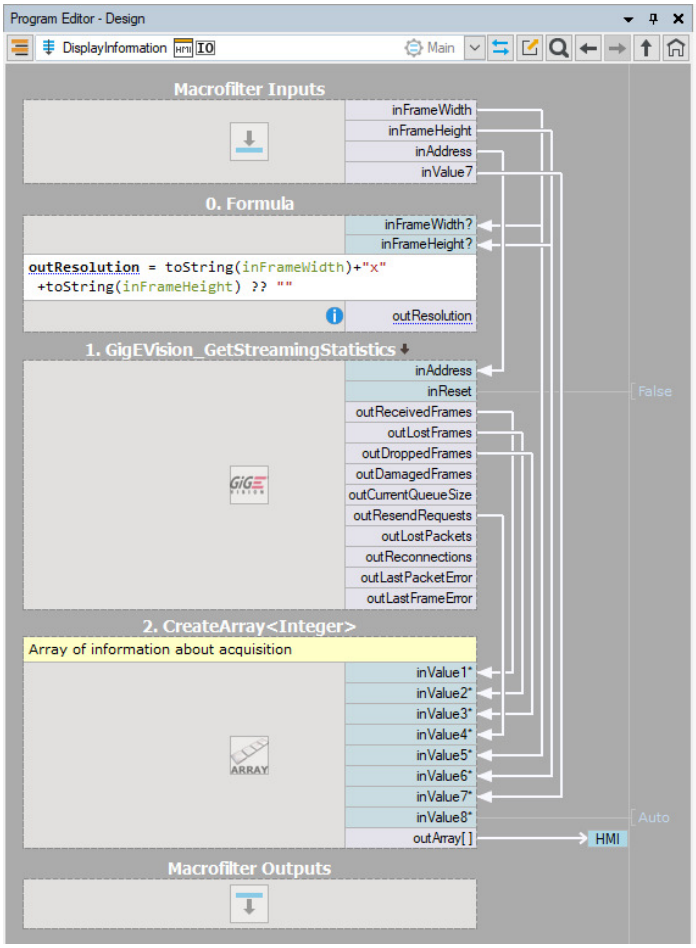




After that, in *GrabImage* macrofilter, the program grabs and displays the image from the camera. Since the **inPixelFormat** has been set when starting the acquisition in the previous part of the program it is not necessary to change the value of this parameter here.

LastNotNil filter keeps the image in the preview in case the user is in triggered acquisition mode. Without it all images from the camera would only be displayed for one iteration of *AcquireImages*.

The last macrofilter in *AcquireImages* gathers data about the current acquisition and displays it on the HMI.



Notes

- The program may not work on every camera. This is dependent on the available parameters as well as their values.
- It is possible to modify more parameters in the program.
- To better understand how some of the parameters work reading the camera's documentation is recommended.
- If functionalities of some of the filters used in this example are unclear, check their documentation.

Interfacing Photoneo to Aurora Vision Studio

Purpose and requirement

This document explains how to interface a PhoXi 3D Scanner to Aurora Vision Studio.

A PhoXi 3D Scanner is an advanced sensor created by Photoneo. It is purposed for 3D machine vision and processing point clouds. The PhoXi 3D Scanner has many functionalities including:

- Scanning objects and representing them as a point cloud or intensity images,
- Various representations of scans.

Required equipment:

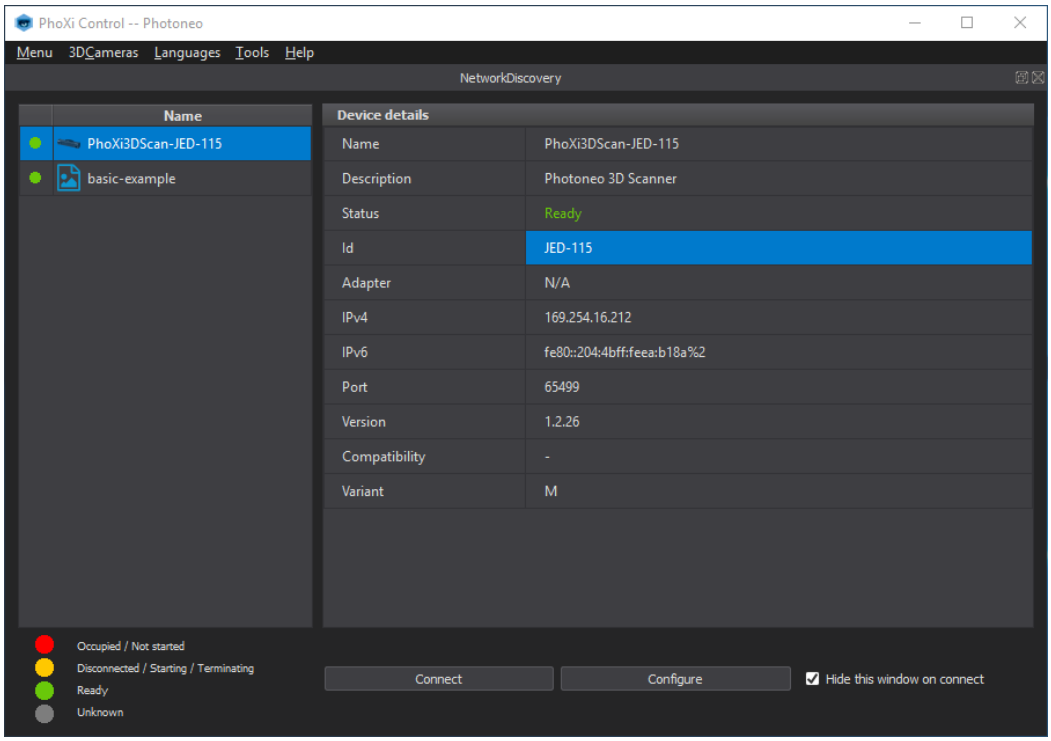
- PhoXi Control v1.2.35 or later
- Aurora Vision Studio 4.11 Professional or later

Getting started with a PhoXi 3D Scanner in Aurora Vision Studio

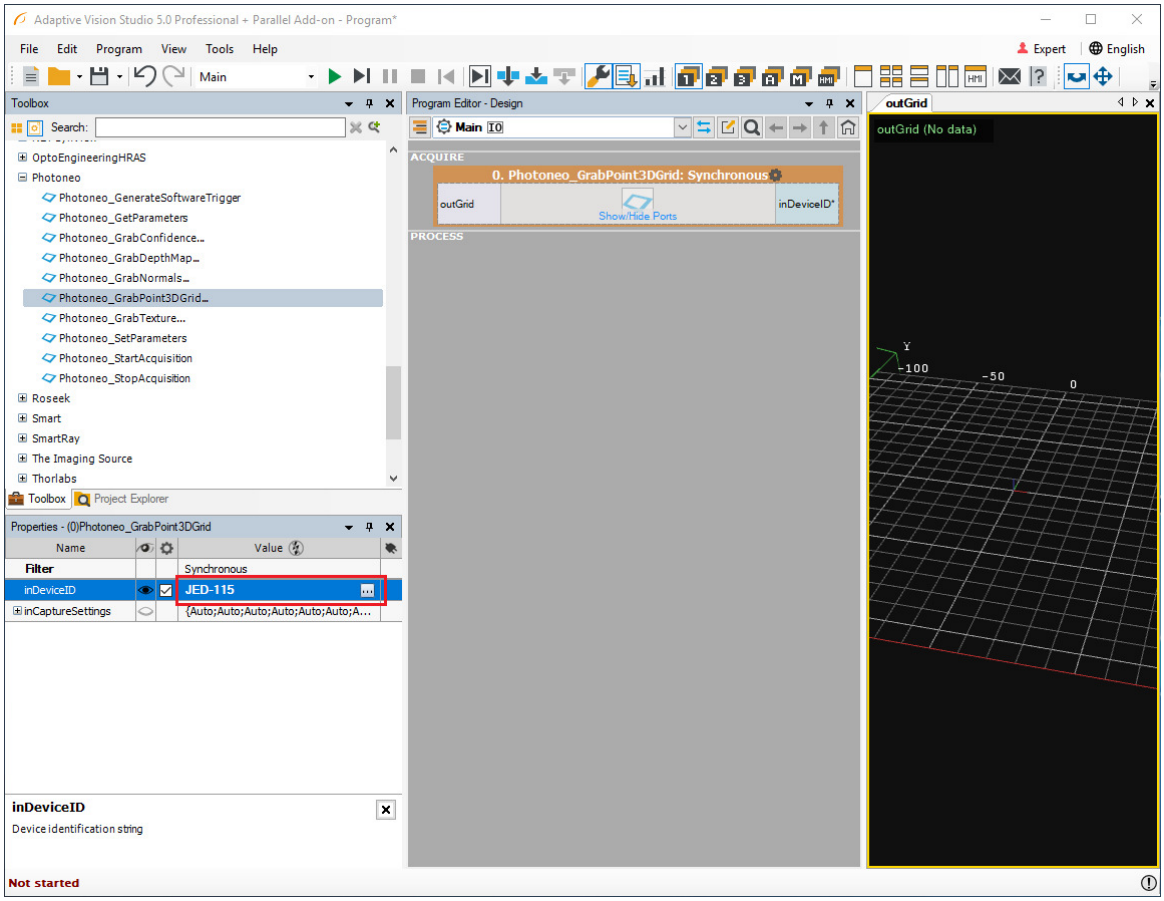
At the beginning download proper version of PhoXi Control from producer's website and install it.

Follow these steps:

1. Connect a PhoXi 3D Scanner to the PC running Aurora Vision Studio Professional.
2. Power up the PhoXi 3D Scanner, connect it to the PC via Ethernet interface.
3. Open PhoXi Control application.
4. Copy ID of the scanner you would like to use in Aurora Vision Studio.



5. To use PhoXi 3D Scanner its Status should be "Ready"
6. Full list of filters intended to work with PhoXi 3D Scanner you can find under this **link**. In this particular example **Photoneo_GrabPoint3DGrid** filter will be used.
7. At the beginning create simple program with **Photoneo_GrabPoint3DGrid** filter to test the connection. Find **Photoneo_GrabPoint3DGrid** filter in the Filter Catalog or in the Toolbox (category Image Acquisition (Third Party)), then drag it and drop it to the main program.
8. Paste ID (from step 4) into **inDevice** input. If you do not have access to any Photoneo scanner, but you would like to test the connection between PhoXi Control and Aurora Vision Studio, please enter "InstalledExamples-basic-example" into **inDevice** input.

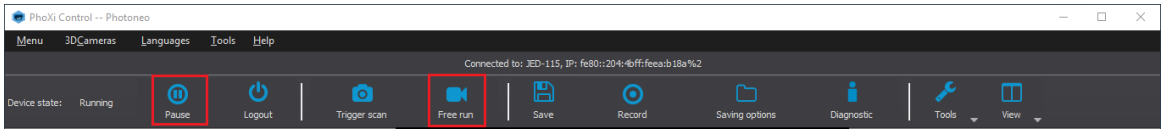


9. Connect **outGrid** output to preview window and run program. If everything is done correctly, you should get Point3DGrid with scanned object.

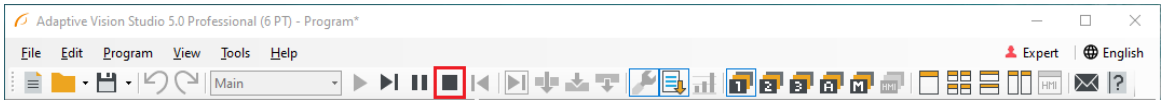
Troubleshooting

If you had any problems during the connection process, you can find some solutions here.

- Make sure you can grab point cloud in PhoXi Control application. If you cannot, please check connection and change the properties of acquisition.
- Make sure you are using proper version of Aurora Vision Studio and PhoXi Control application (from chapter Purpose and requirement)
- Make sure that the device is in running mode. If necessary, unpause the device in PhoXi Control and use Free run, then you should be able to grab Point3DGrid in Aurora Vision Studio.



- You will not be able to change parameters in the PhoXi Control application while a program in Aurora Vision Studio is running and acquiring data from a scanner. You can stop the program by clicking the Stop button or using the Shift + F5 key combination.



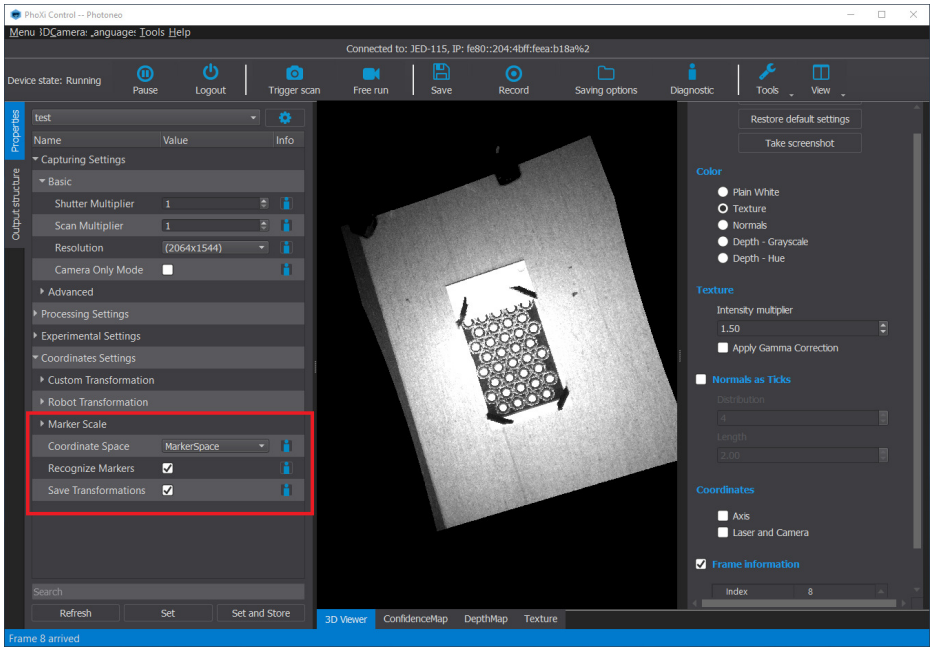
If you encounter a problem that has not been mentioned above, please do not hesitate to contact us, so that we could investigate it and add the description of its solution to this section.

Calibration

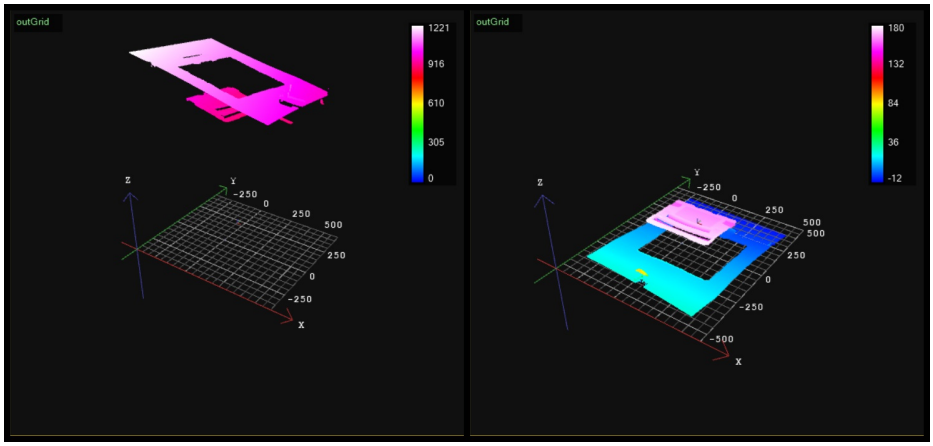
Point's coordinate system is defined by scanner's camera position and its specific angle depending on PhoXi 3D Scanner type. You can read more about Coordinate spaces of Photoneo scanners in this document

You can align coordinate system position by following this instruction. After completing steps outlined by the manufacturer, the coordinate system will be changed also in case of scans acquired with Aurora Vision Studio.

To make an calibration of PhoXi 3D Scanner you have to print calibration board. Then navigate to Properties -> Coordinates Settings -> Marker Scale in PhoXi Control and change parameters to these like on the picture below. Acquire scan of points cloud using Trigger scan and then click Set and Store.

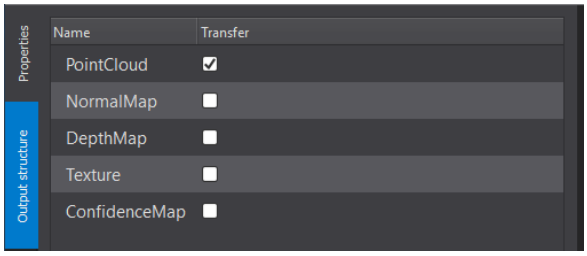


Below picture represents points cloud before and after calibration.



Setting parameters and maps reading

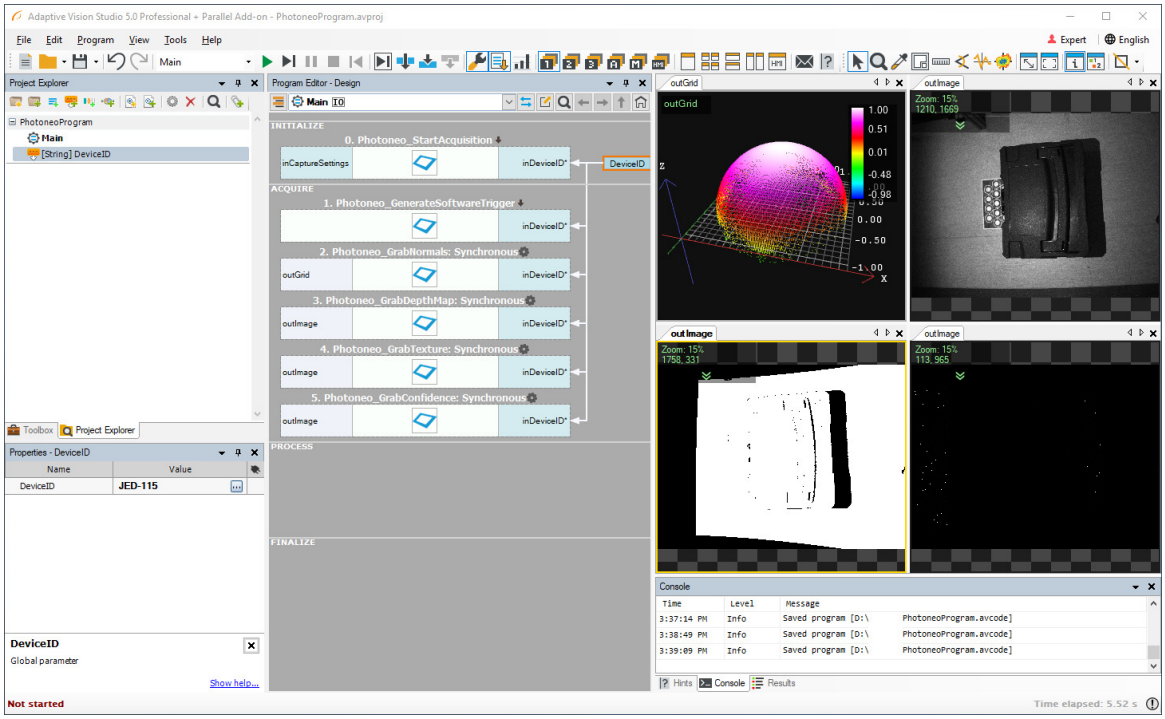
Default parameters of PhoXi 3D Scanner are shown below.



These settings does not allow to send Normal maps, Depth maps, Confidence maps and Textures to Aurora Vision Studio application. You can tick checkboxes in PhoXi Control software or you can set these parameters in Aurora Vision Studio.

To make it possible in Aurora Vision Studio, you need to change the parameters using **Photoneo_SetParameters** filter, which should be executed at the very beginning of program or between **Photoneo_StopAcquisition** and **Photoneo_StartAcquisition** filters.

If parameters, responsible for maps reading, will be set as True, you can use proper filters (**Photoneo_GrabNormals**, **Photoneo_GrabTexture**, **Photoneo_GrabDepthMap**, **Photoneo_GrabConfidence**) to acquire respective maps to Aurora Vision Studio application.


Adaptive Vision Studio 5.0 Professional + Parallel Add-on - PhotoneoProgram.avproj
File Edit Program View Tools Help
Expert English
Main
Project Explorer
PhotoneoProgram
Main
[String] DeviceID
Program Editor - Design
Main
INITIALIZE
0. Photoneo_StartAcquisition
inCaptureSettings
inDeviceID*
DeviceID
ACQUIRE
1. Photoneo_GenerateSoftwareTrigger
inDeviceID*
2. Photoneo_GrabNormals: Synchronous
outGrid
inDeviceID*
3. Photoneo_GrabDepthMap: Synchronous
outImage
inDeviceID*
4. Photoneo_GrabTexture: Synchronous
outImage
inDeviceID*
5. Photoneo_GrabConfidence: Synchronous
outImage
inDeviceID*
PROCESS
FINALIZE
Toolbox
Project Explorer
Properties - DeviceID
Name
Value
DeviceID
JED-115
DeviceID
Global parameter
Show help...
outGrid
outGrid
1.00
0.51
0.01
-0.48
0.00
-0.50
-1.00
outImage
Zoom: 15%
1210, 1669
outImage
Zoom: 15%
1758, 331
outImage
Zoom: 15%
113, 965
Console
Time
Level
Message
3:37:14 PM Info Saved program [D:\ PhotoneoProgram.avcode]
3:38:49 PM Info Saved program [D:\ PhotoneoProgram.avcode]
3:39:09 PM Info Saved program [D:\ PhotoneoProgram.avcode]
Hints
Console
Results
Not started
Time elapsed: 5.52 s

Interfacing Wenglor profile sensor to Aurora Vision Studio

Purpose and requirement

This document explains how to interface a Wenglor profile sensor to Aurora Vision Studio.

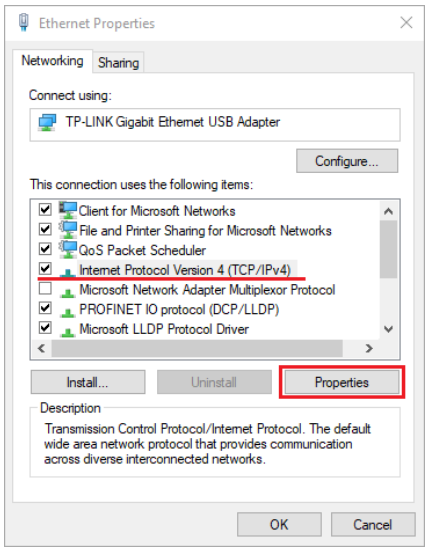
Required equipment:

- Wenglor weCat3D Server 2.0.0 or later. You can download it under **GigE Vision Interface** section (version 2.2.1 used);
- Aurora Vision Studio 5.0 Professional or later (version 5.2 used);
- Wenglor weCat3D sensor (MLSL123 used).

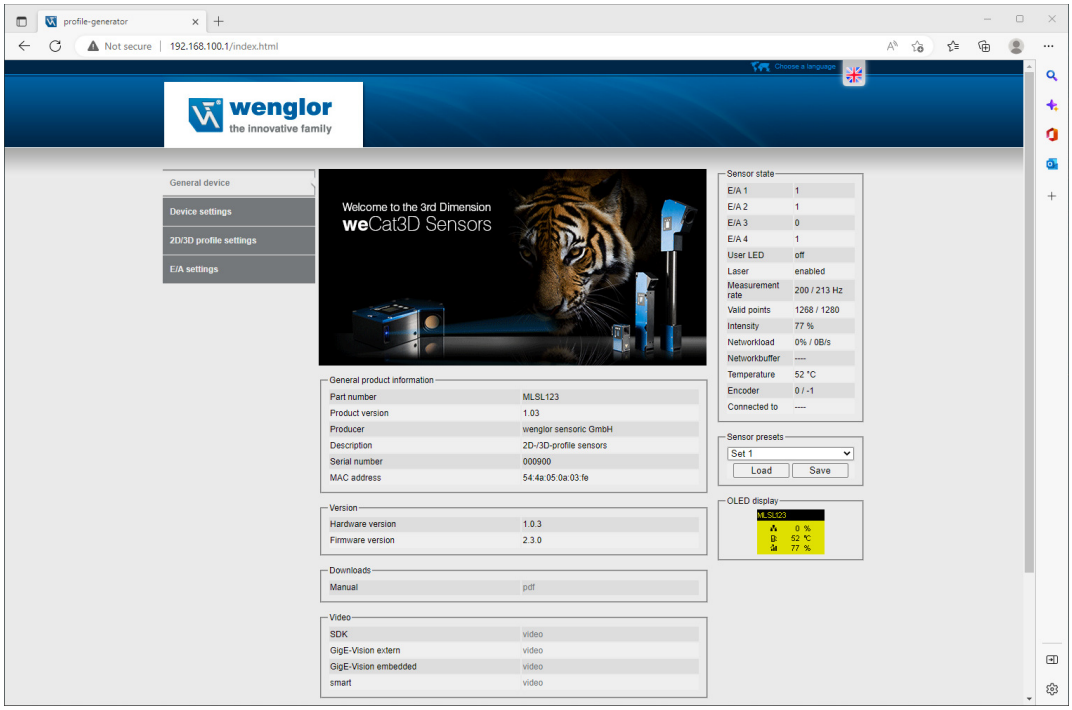
Setting up Wenglor weCat3D sensor

Each sensor is shipped with the default IP address of 192.168.100.1. So before working with the Wenglor you must make sure that the IP address has been set correctly and is unique for each device in the network. You can verify that with the steps described below:

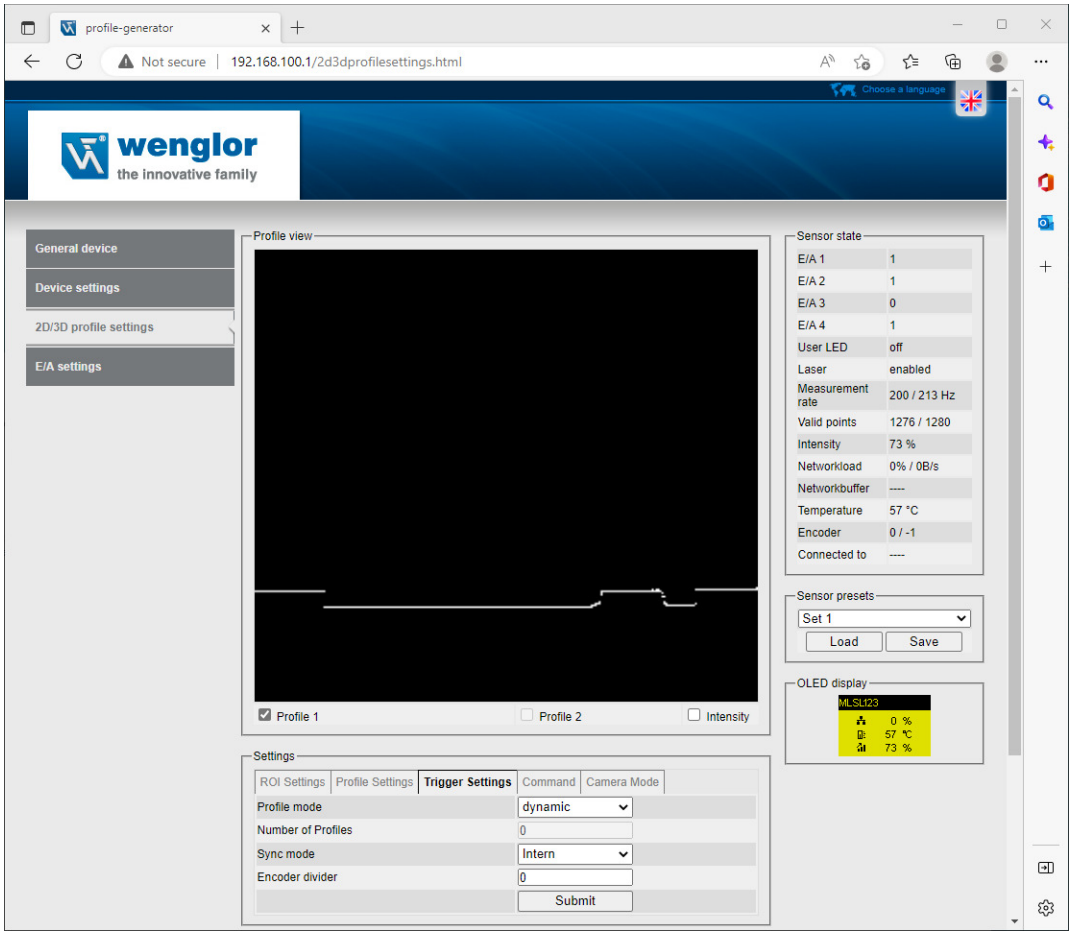
1. Connect the Wenglor to the PC running Aurora Vision Studio Professional.
2. Power up Wenglor, connect it to the PC via Ethernet interface.
3. Open Control Panel on your PC.
4. Find Network and Sharing Center.
5. Choose Change adapter settings.
6. Right-click on unidentified network connection, choose Properties.
7. In Ethernet Properties find Internet Protocol Version 4 (TCP/IPv4) and click Properties.



8. In General tab, choose Use the following IP address and set:
    a. IP address to (for example) 192.168.100.181
    b. Subnet mask to 255.255.255.0
9. Click OK and Close. Now you should be able to connect with the Wenglor scanner. You can verify this using your web browser.
10. In a search tab, type 192.168.100.1 and click Enter.
11. You should see this user interface:

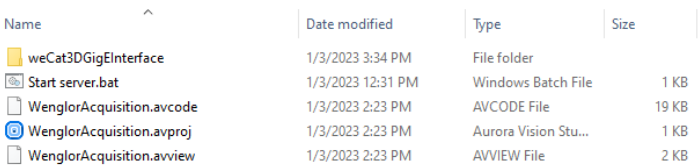


12. The provided browser-based user interface allows you to configure the parameters of the Wenglor sensor. You can navigate to the *2D/3D profile settings tab* to check if profile is acquired with *Intern* Sync mode:

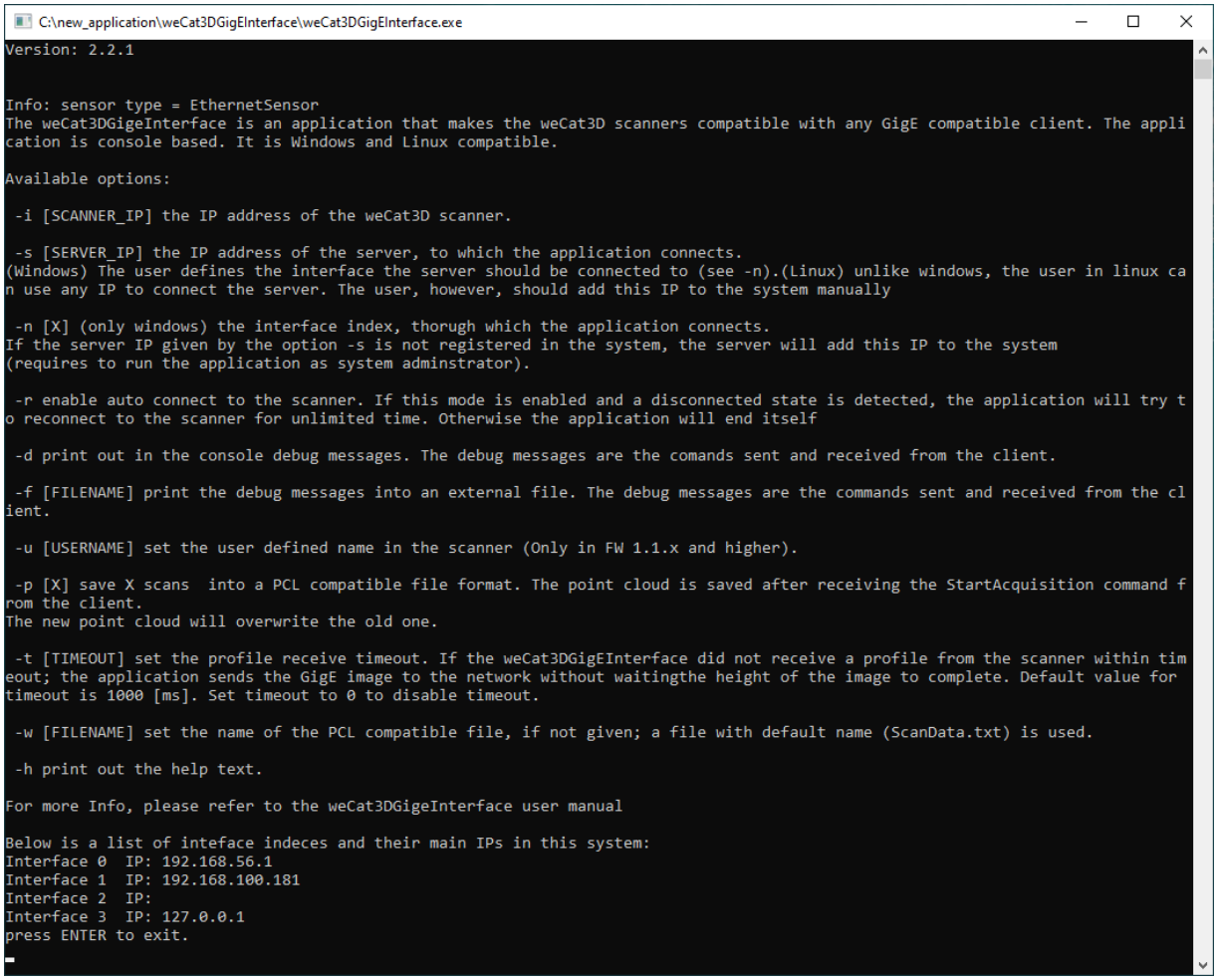


Setting up Wenglor weCat3D server

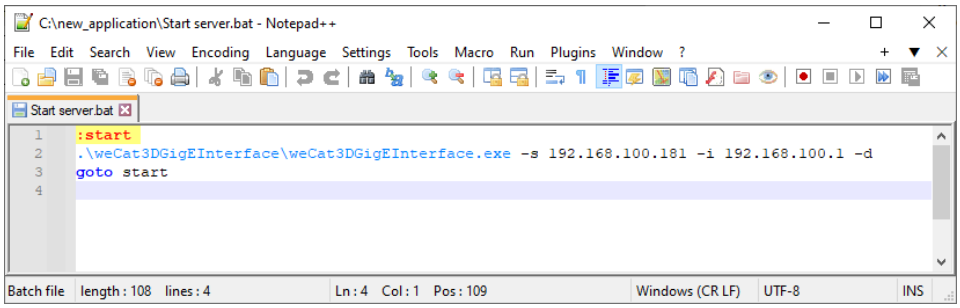
If you download Wenglor server archive, there should be a folder named "weCat3DGigEInterface". Copy it to the folder with Aurora Vision Studio project:



If you run Wenglor server executable, console application will be shown:



This console application should be run with appropriate arguments described in Wenglor server documentation. You can create file with *bat* extension with the following syntax:

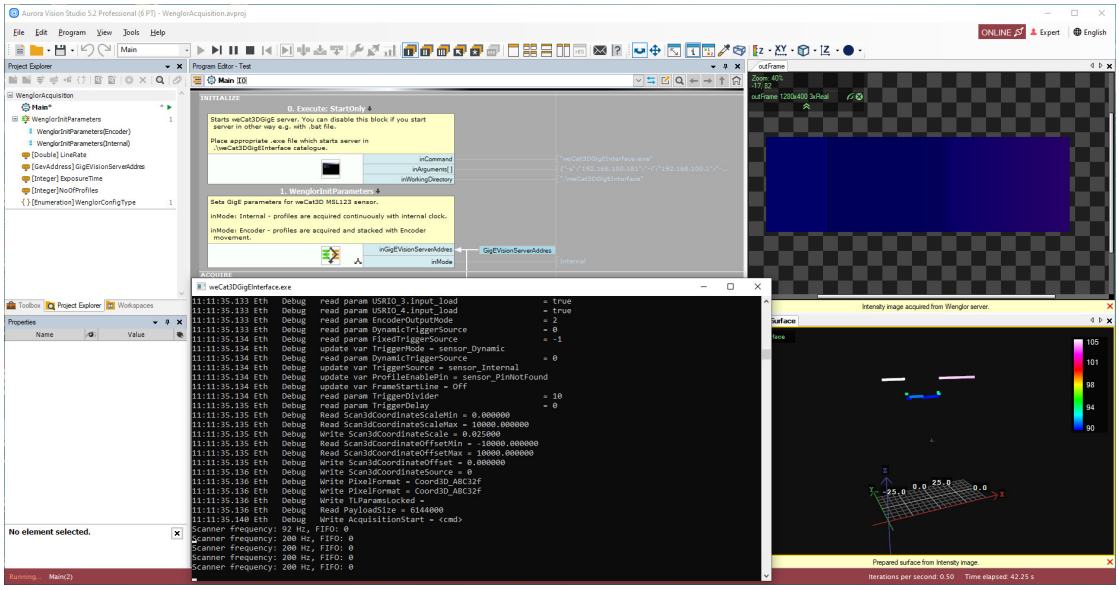


Using *bat* file extension is not obligatory and you can start server with appropriate arguments by *Command prompt*. You can start server with Execute_StartOnly filter – it is described in the next chapter.

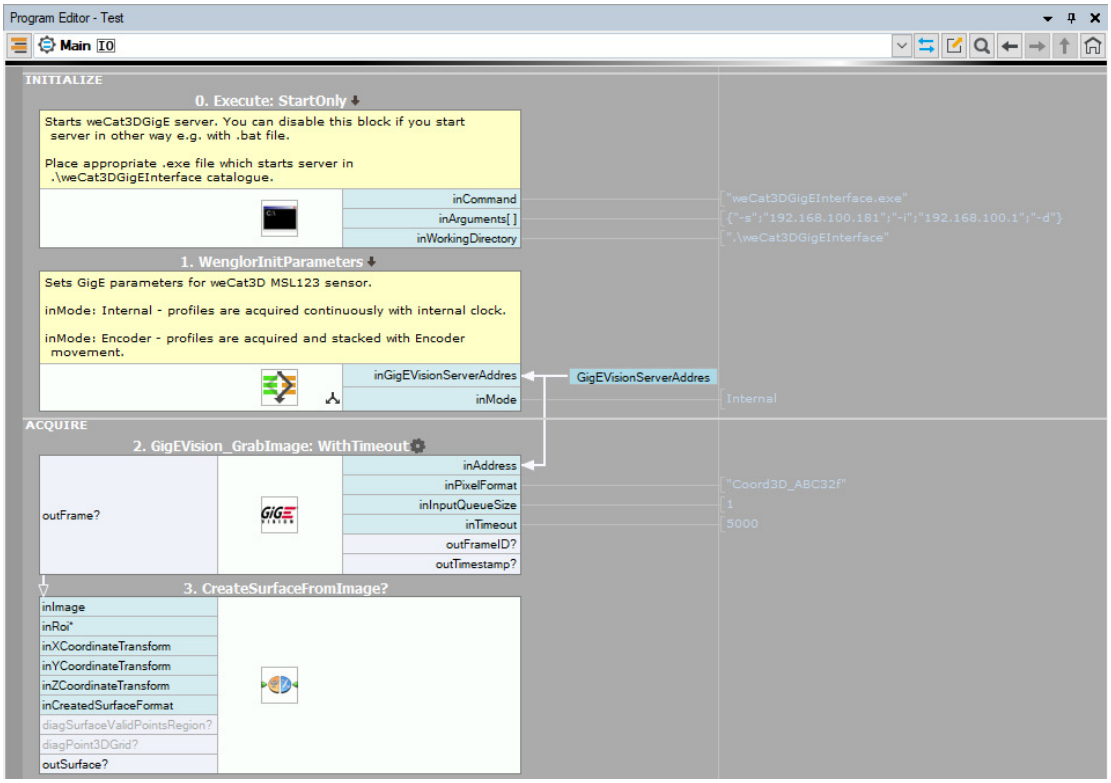
Setting up Wenglor weCat3D scanner with Aurora Vision Studio

In order to combine Wenglor server with Aurora Vision Studio, please refer to **Wenglor – GigE Server 2.0 Connection** official example.

Before you start this application, it is necessary to place "weCat3DGigEInterface" folder inside application folder. If everything is set up correctly, you should be able to acquire profiles with *Internal* mode:

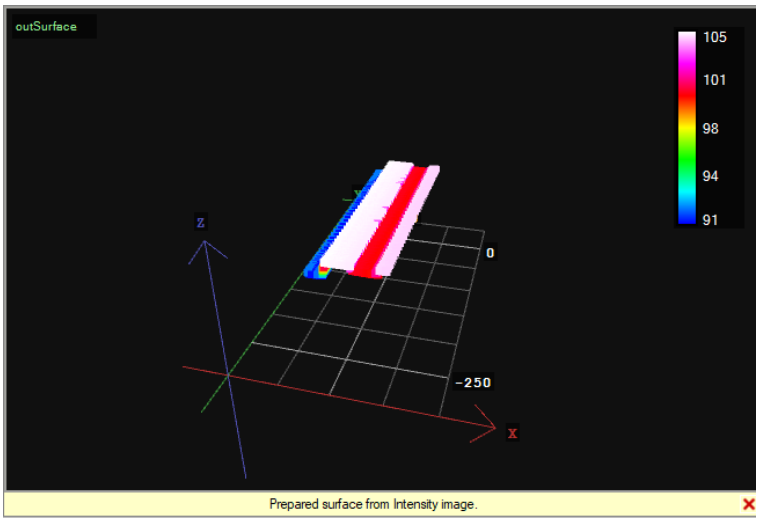


In INITIALIZE section Execute_StartOnly filter starts Wenglor WeCat3D server. Then **WenglorInitParameters** macrofilter sets appropriate parameters with **GigEVision_SetParameter** filters:



For more details about parameters initiation with Wenglor, please refer to the official example inside Aurora Vision Studio and to the manual attached to the downloaded WeCat3D Server.

If you change **inMode** from Internal to Encoder, you will be able to acquire whole surface from profiles acquired with encoder trigger source:


outSurface
105
101
98
94
91
Z
0
-250
X
Prepared surface from Intensity image.

Interfacing Gocator to Aurora Vision Studio

Purpose and requirement

This document explains how to interface a Gocator sensor to Aurora Vision Studio.

A Gocator is an advanced sensor created by LMI Technologies. It is purposed for 3D machine vision and processing point clouds. The Gocator has many functionalities including:

- Scanning objects and representing them as a point cloud or intensity images,
- Various representations of scans,
- Basic tools for 3D processing like: filtering, thresholding etc.,
- Tools for performing measurements.

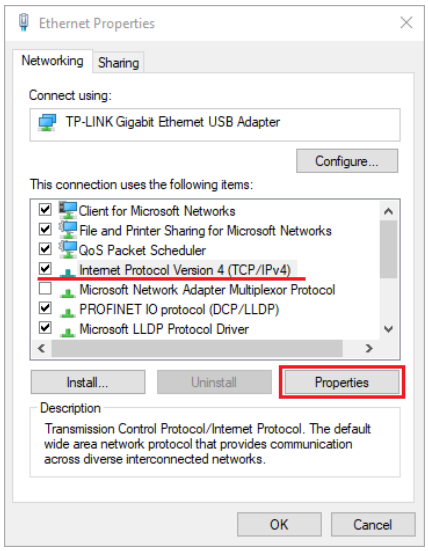
Required equipment:

- Gocator Firmware Release 3.2 or later
- Aurora Vision Studio 4.10 Professional or later

Setting up Aurora Vision Studio with a Gocator for the first time

Each sensor is shipped with a default IP address of 192.168.1.10. So before working with the Gocator you must make sure that the IP address has been set correctly and is unique for each device in the network. You can verify that with the steps described below:

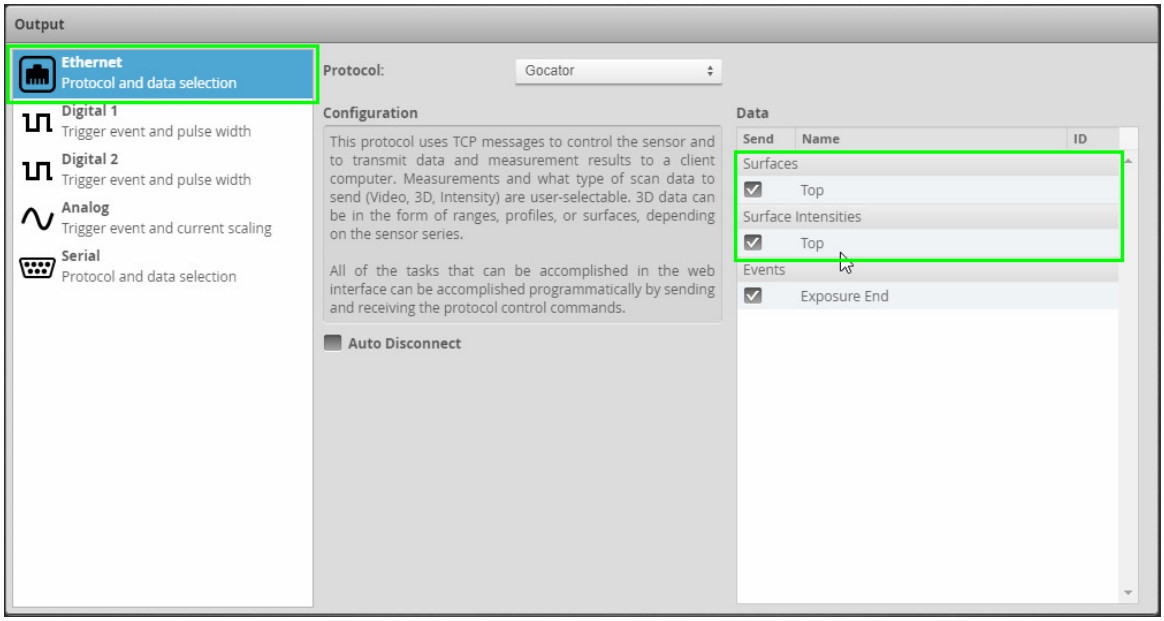
1. Connect a Gocator to the PC running Aurora Vision Studio Professional.
2. Power up the Gocator, connect it to the PC via Ethernet interface.
3. Open Control Panel on your PC.
4. Find Network and Sharing Center.
5. Choose Change adapter settings.
6. Right-click on unidentified network connection, choose Properties.
7. In Ethernet Properties find Internet Protocol Version 4 (TCP/IPv4) and click Properties.



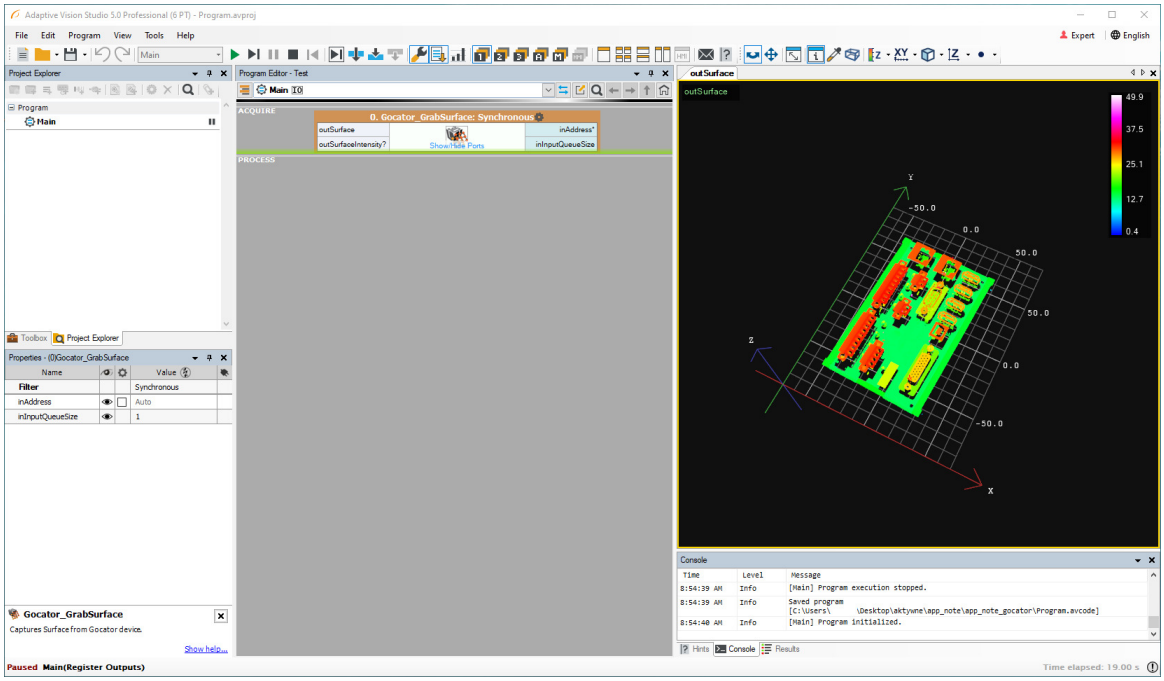
8. In General tab, choose Use the following IP address and set:
    a. IP address to (for example) 192.168.1.5
    b. Subnet mask to 255.255.255.0
9. Click OK and Close. Now you should be able to connect with the Gocator. You can verify this using your web browser.
10. In a search tab, type 192.168.1.10 and click Enter.
11. You should see this user interface:



12. The provided browser-based user interface allows you to configure the parameters of the Gocator sensor. Before working with Gocator in Aurora Vision Studio it is required to set parameters in the browser interface because from the Studio side you are unable to change them. In Aurora Vision you will see data the same as in the browser. To send specific data outputs to Aurora Vision Studio click on the "Output" tab and there select Ethernet output. Then you must choose the data to send. Note that marking all outputs is optional so it is possible to mark only these which are interesting for you. For example, with the setup shown below you will be able to acquire Surface and Surface Intensity in Aurora Vision Studio. For the outputs in the Measurement section there are also specified ID's. The same ID should be later set on the Aurora Vision side.



13. Now go back to Aurora Vision Studio. In Image Acquisition (Third Party) section find LMI and double-click. Choose **Gocator_GrabSurface** to grab a surface or **Gocator_GrabProfile** to grab a profile. Run or iterate the program. It is recommended to use the other option for testing. If everything works fine you will receive the point cloud or the profile of the object from your Gocator Note that this filter has an input inAddress, but it is not necessary to use it if there is only one Gocator connected with your PC. You must set this input if there are more sensors.



14. If no problems occurred so far, it means you have successfully connected Aurora Vision Studio to the Gocator.

Troubleshooting

If you had any problems during the connection process, you can find some solutions here.

- If you cannot connect with a Gocator, please verify your Ethernet connection. Unplug your Internet cable and plug the cable from the Gocator.
- Verify whether you have correctly set IP address.
- If the device cannot connect, use the Security button on your Master device.
- Make sure your browser uses the newest version of Flash.
- In the case that other page is displayed on the 192.168.1.10 address press Ctrl + F5 to force a cache refresh.

If you meet the problem we have not mentioned above, please let us know, so that we could investigate it and add it to this section.

Gocator filters in Aurora Vision Studio

You can find the full list of filters intended to work with Gocator devices under this **link**.

Working with Emulator

Gocator Emulator is a software that allows for running a virtual sensor instead of using the real one. Emulator's interface is identical to the one in a web browser. Thanks to that you can get acquainted with all functionalities that the Gocator provides. To start, download the proper tools package from: https://lmi3d.com/support. Then, to run the application, first unzip the package, then find the GoEmulator file and run it. The emulator lets you change the language and select a version. After launching the application, start from step 12 according to the walkthrough in chapter 3. After starting GoEmulator, you must choose a scenario to run. It is possible to display data from scenarios in Aurora Vision Studio. The connection between Aurora Vision Studio and GoEmulator should be set automatically – you do not have to set a specific IP.

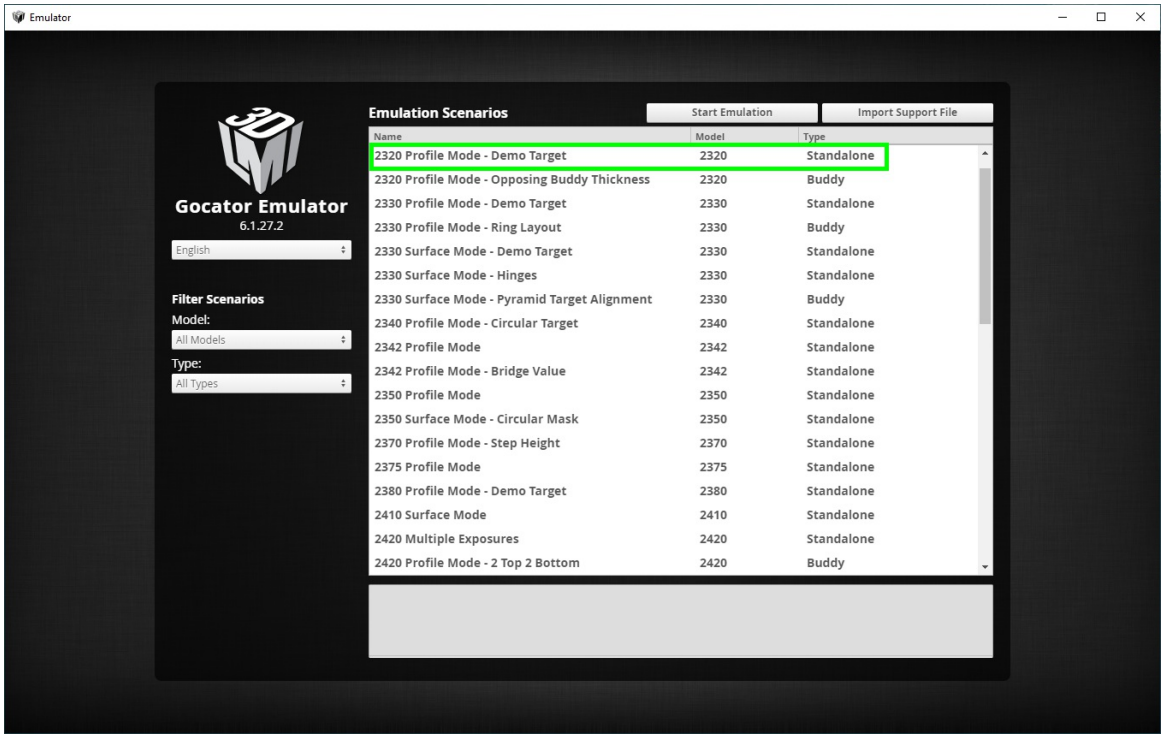
Examples

In this chapter several examples are presented to help you understand how to use functionalities described above and to help you learn how to work with the Gocator's user interface and filters in Aurora Vision Studio.

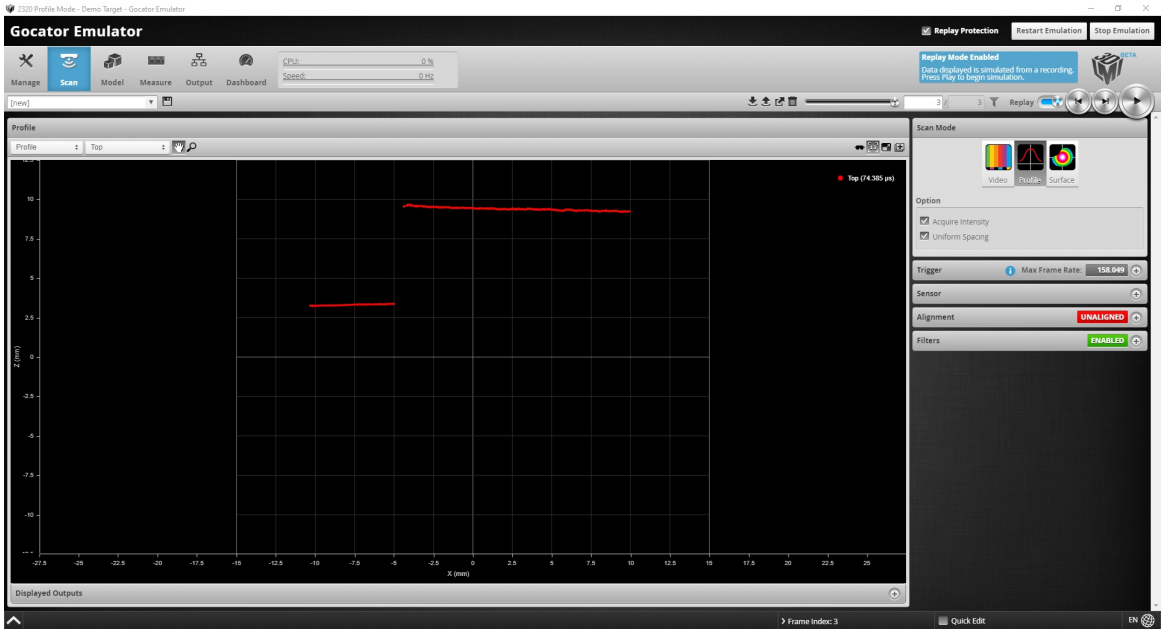
Example 1 – getting profile

To grab profile from a Gocator (in this case Emulator) follow the steps below:

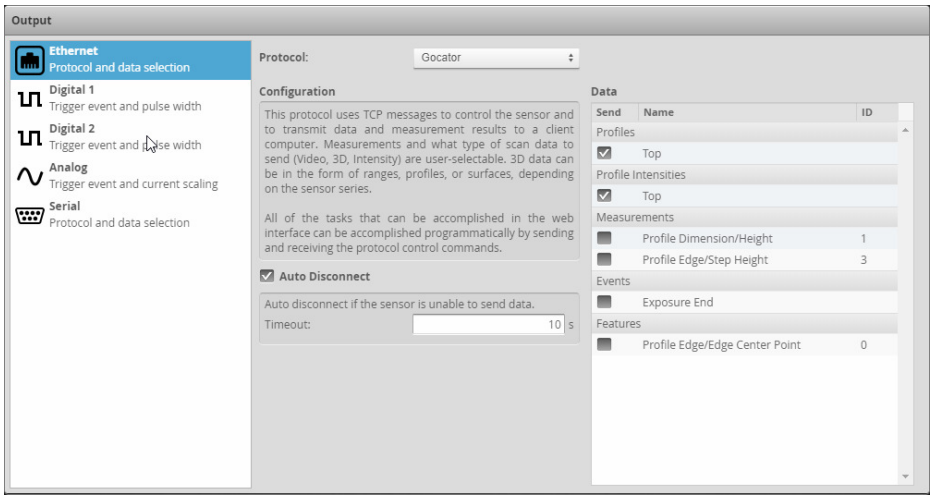
- Choose scenario:



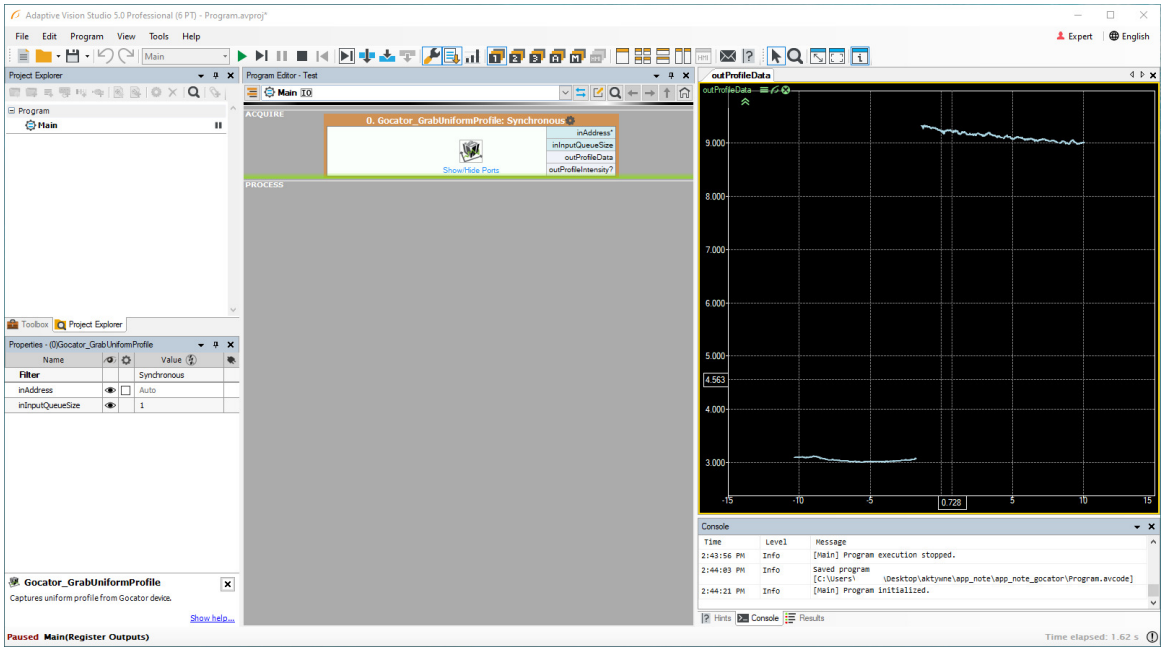
- In the "Scan" tab run the scenario:



- In the "Output" tab make sure that the profile top output is marked:



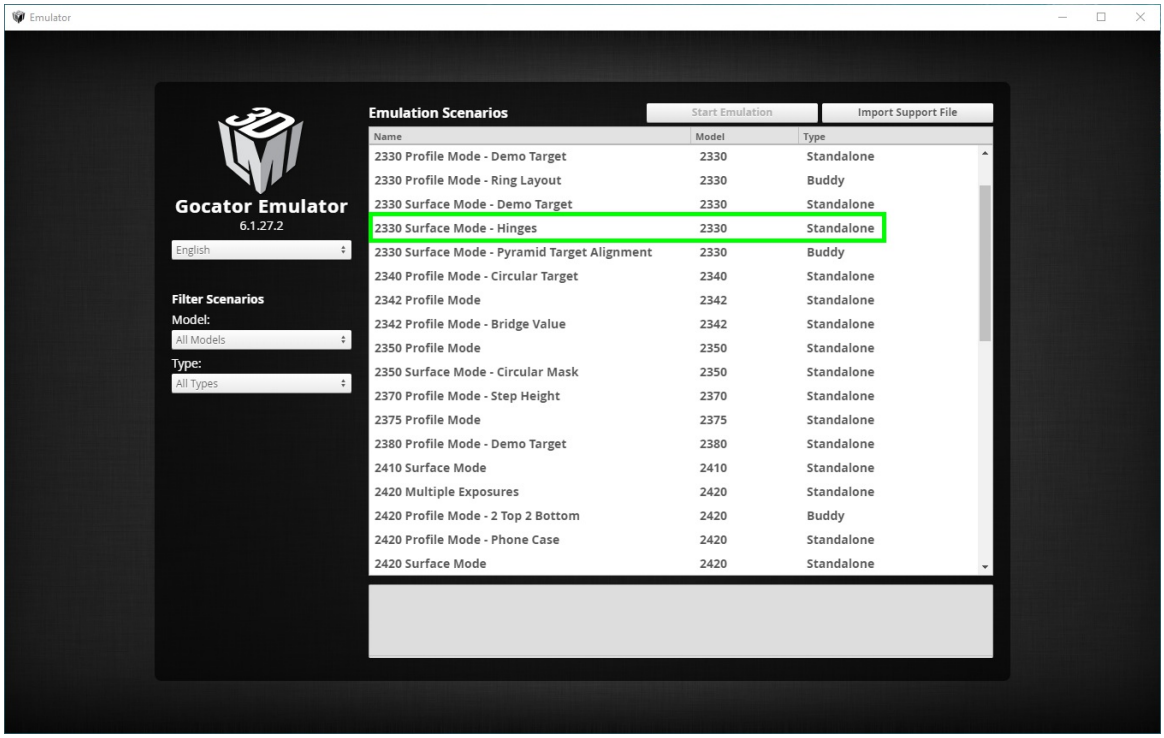
- In Aurora Vision Studio choose the **Gocator_GrabUniformProfile** filter and add the output **outProfileData** to a preview window:



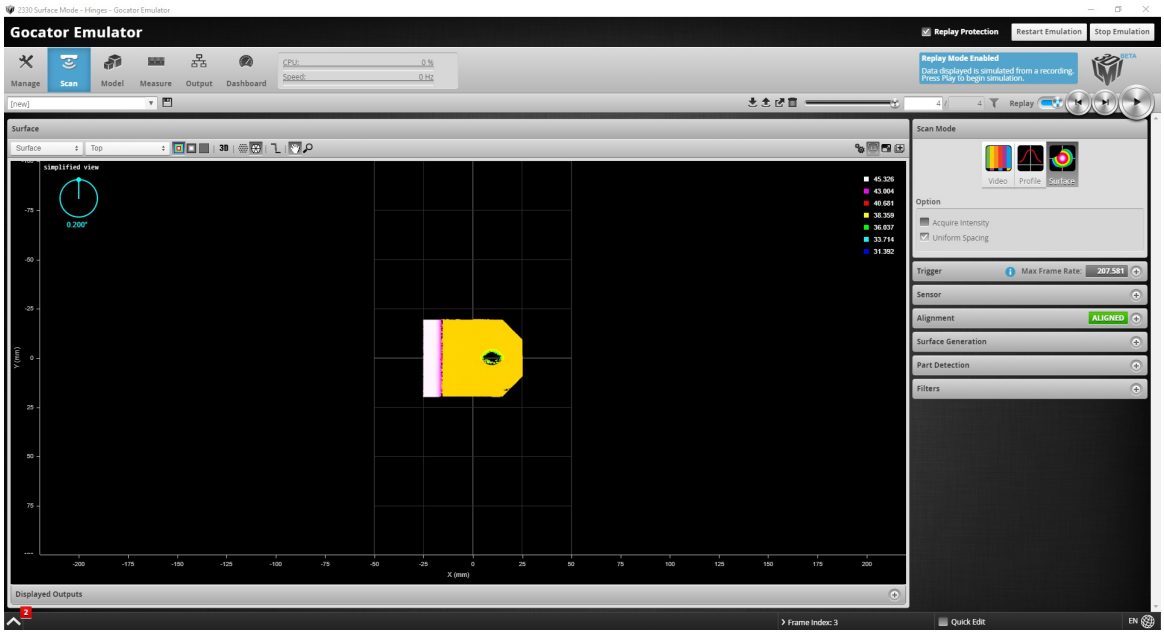
Example 2 – getting surface

To grab surface from a Gocator (in this case Emulator) follow the steps below:

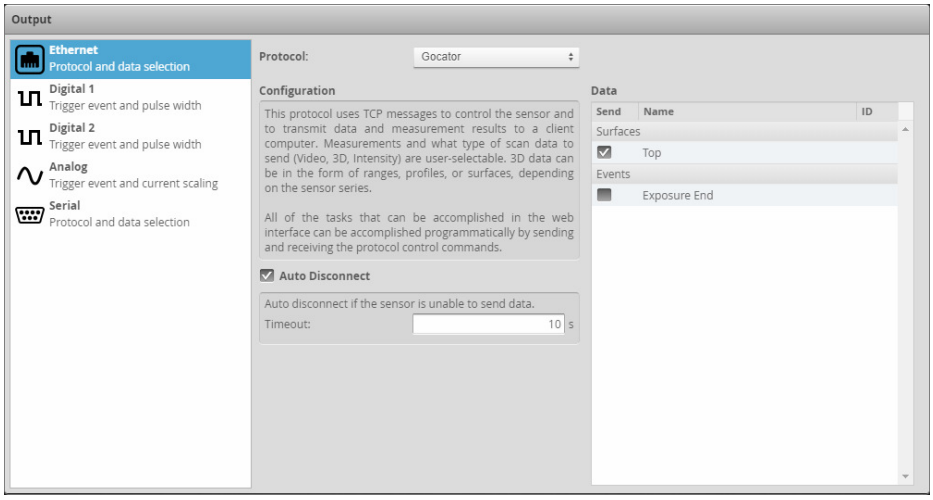
- Choose scenario:



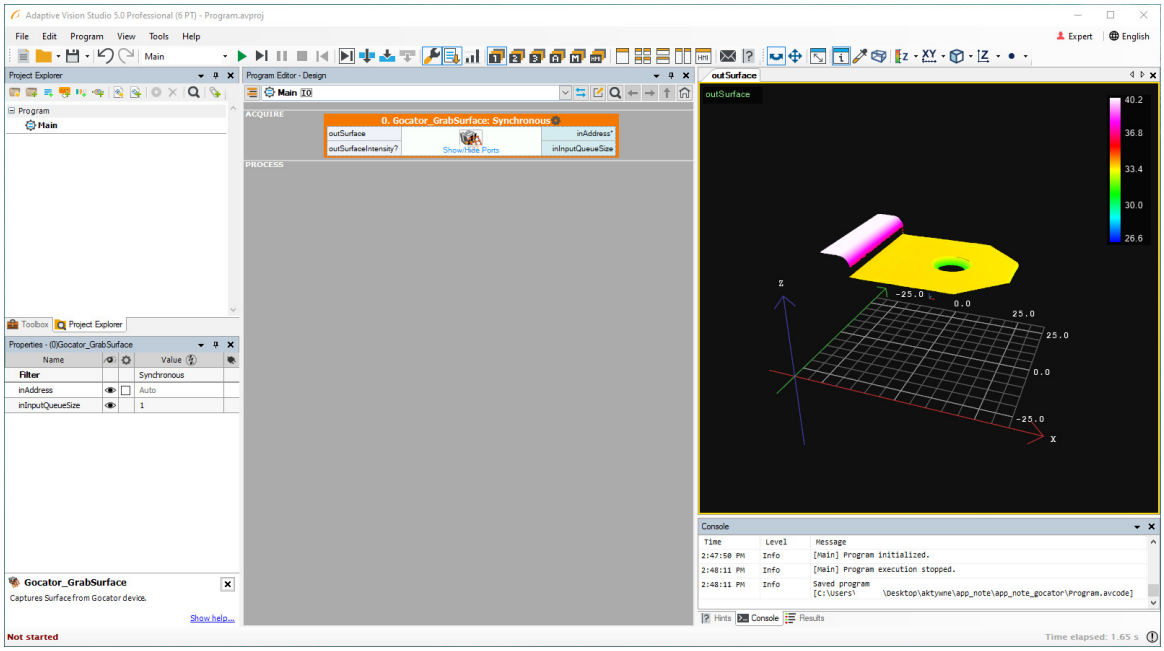
- In the "Scan" tab run the scenario:



- In the "Output" tab make sure that the profile top output is marked:



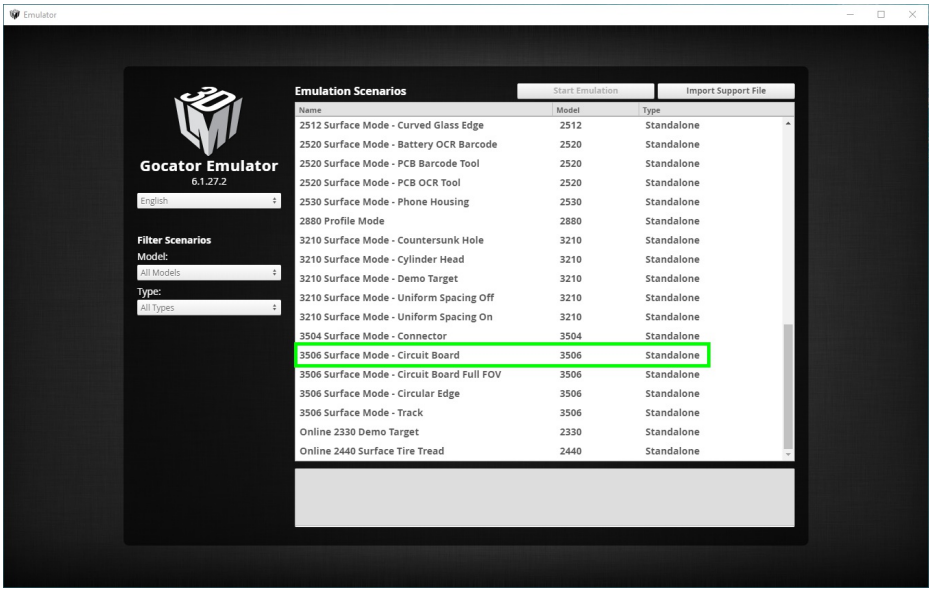
- In Aurora Vision Studio choose the **Gocator_GrabSurface** filter and add the output **outSurface** to a preview window:



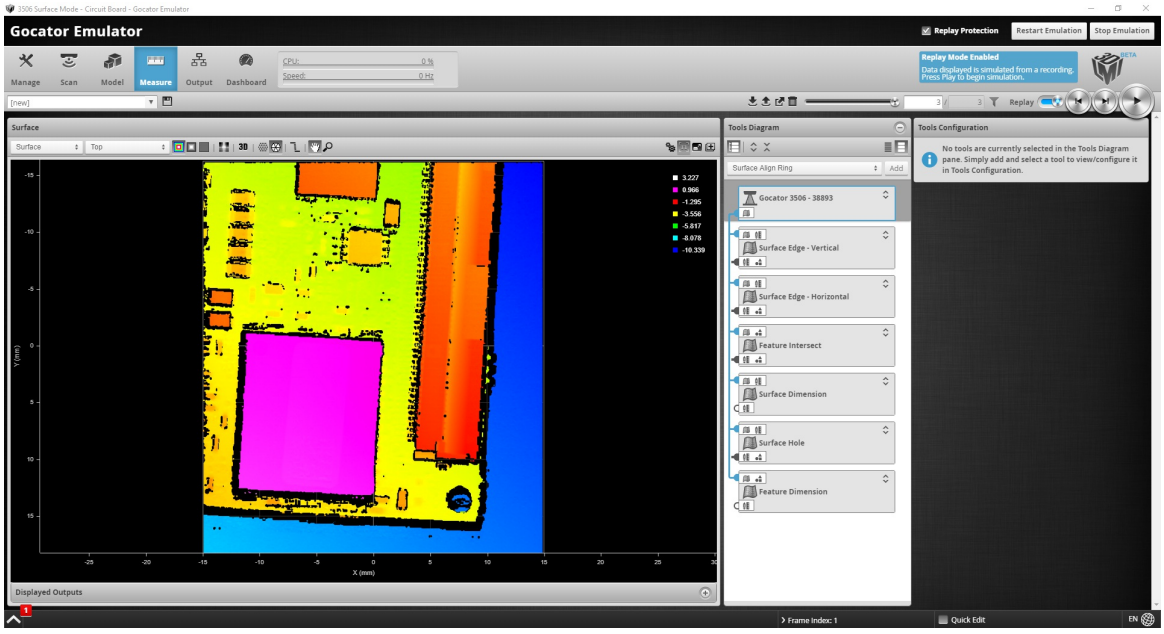
Example 3 – getting measurements

To grab measurement from a Gocator (in this case Emulator) follow the steps below:

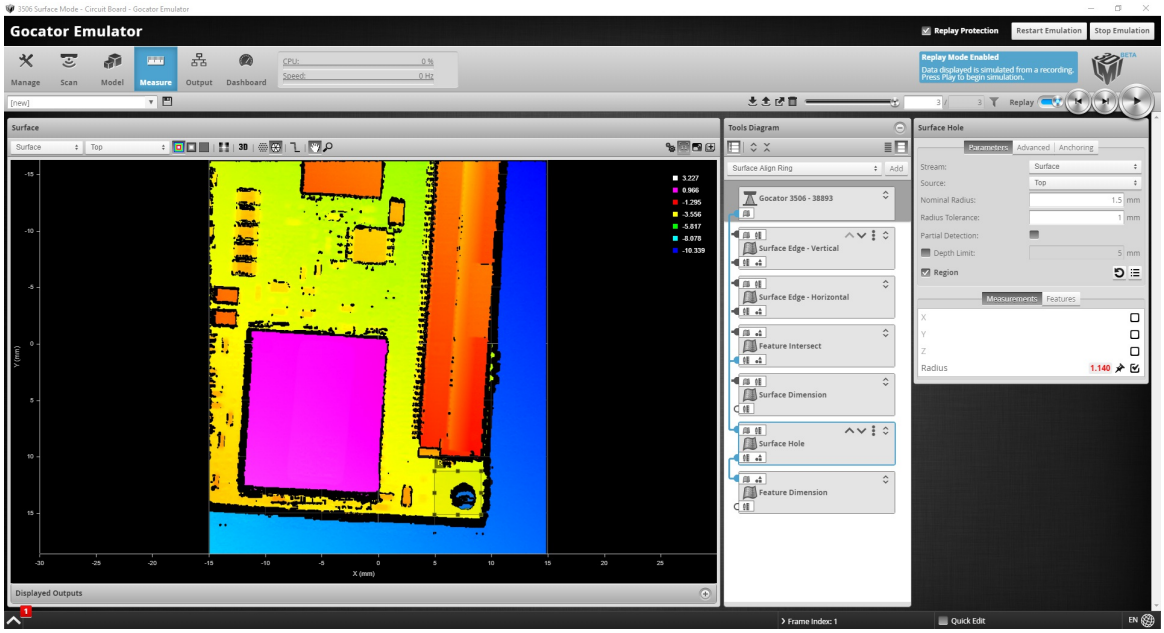
- Choose scenario:



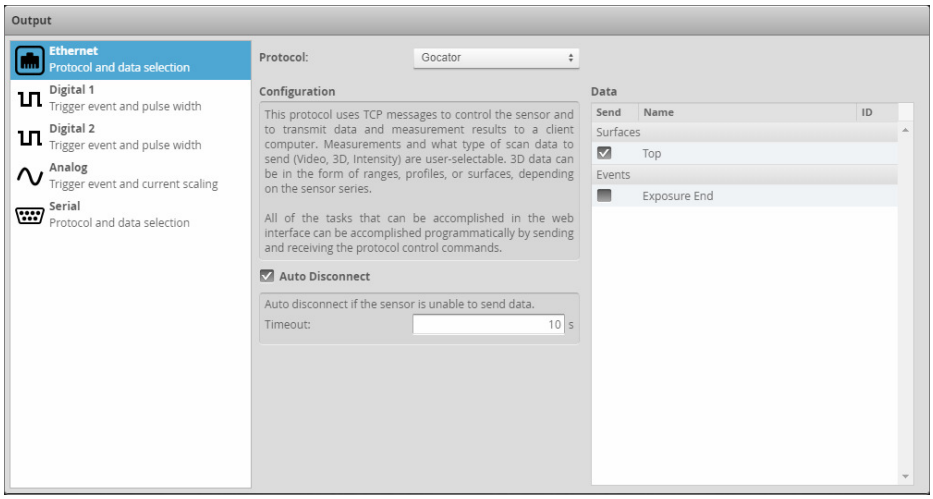
- In the "Scan" tab run the scenario:



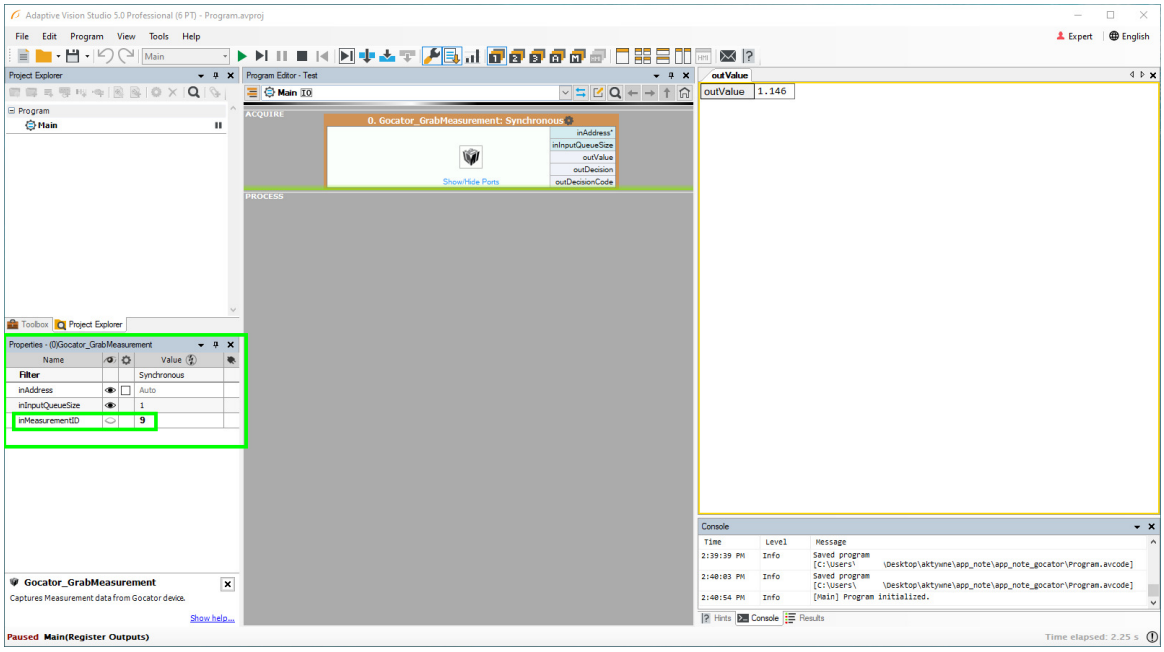
- In the "Measure" tab choose required measurements:



- In the "Output" tab make sure that the output data you would like to send is marked:



- In Aurora Vision Studio choose the **Gocator_GrabMeasurement** filter and add the output **outValue** to a preview window. Please note that the data output from a previous paragraph has a specific ID, so you must enter this value in Aurora Vision Studio – in Property window there is the parameter named **inMeasurementID** that should be set to 9, because our goal is to display the value of hole's radius:



Acquiring data from SICK TriSpector devices

Purpose and requirement

The following instruction provides information on how to connect and use the TriSpector camera to receive a 3D image in the Aurora Vision Studio. For additional information refer to the device documentation. While working with the device keep in mind work safety and general safety regulations.

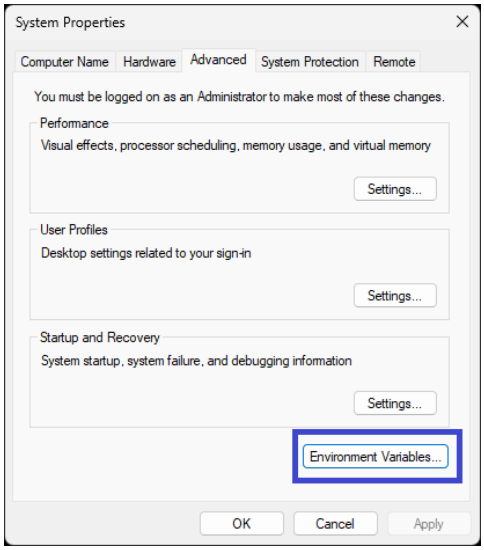
Required equipment:

- SOPAS Engineering Tool.
- Aurora Vision Studio 4.10 Professional or later

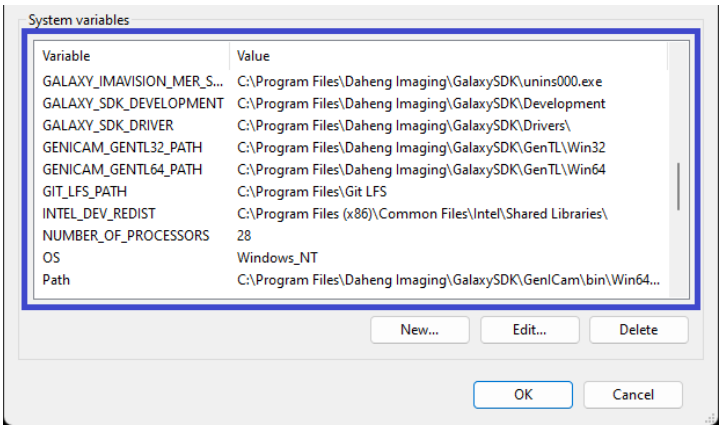
Checking environmental variable

You should have received a .zip folder named **SICKTrispectorTLxx_V2.4.0_1r6cwIe** or similar. After unzipping it you will find **.cti** files.

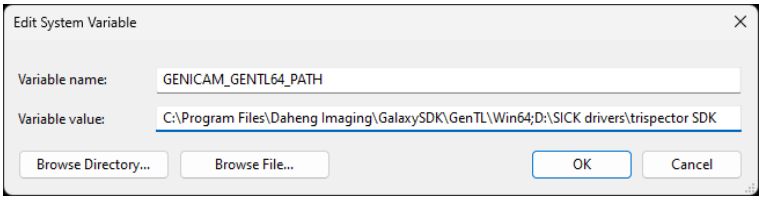
1. In Advanced System Settings, in the Advanced tab press **Environmental Variables**.



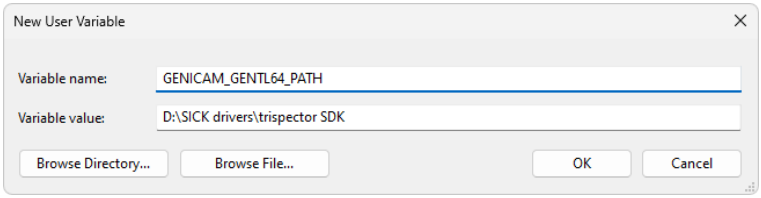
2. Check if variables GENICAM_GENTL64_PATH and GENICAM_GENTL32_PATH exist. To do it browse the selected window:



   - If the variables exist, use **Edit** option for each one and add the .cti file location path to the existing one (use ; as a separator)



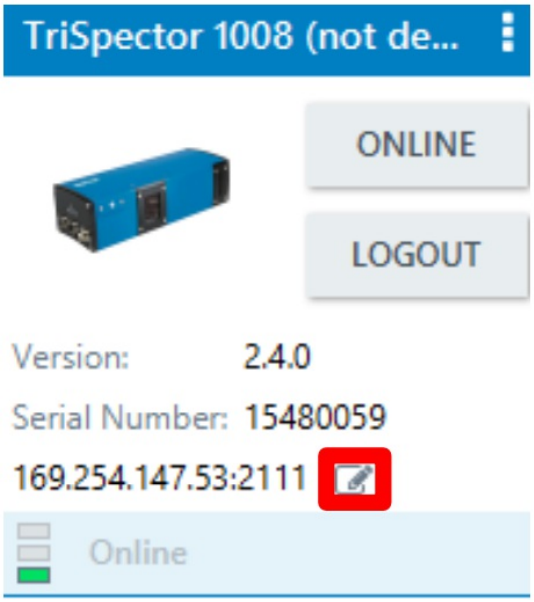
   - If there is no variable like that you need to create them using the appropriate name (GENICAM_GENTL64_PATH, GENICAM_GENTL32_PATH) and the .cti file location path:



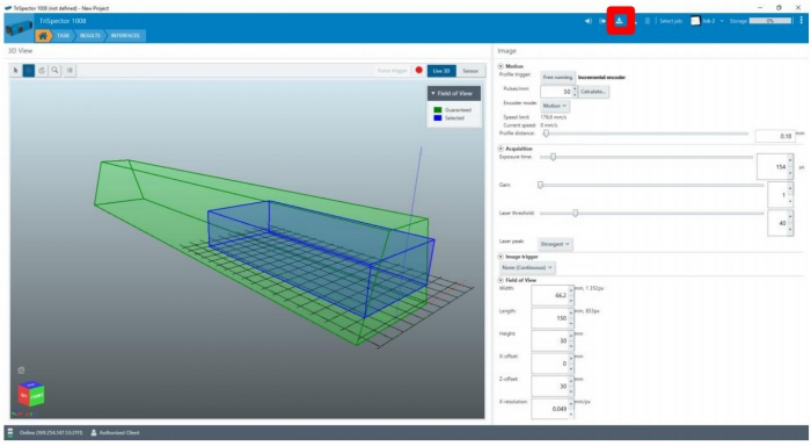
3. In both cases confirm by pressing OK.

SOPAS Engineering Tool

It is necessary to detect the camera in Aurora Vision Studio later on and configure all of the parameters used by it (SICK does not allow to modify majority of the parameters by third party software). The next step is to install the correct drivers as they are required for the device to work correctly. In the window seen below you can adjust the IP address of the camera (red box) or enter the editor (double click).

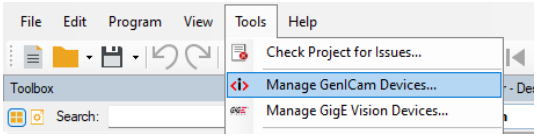


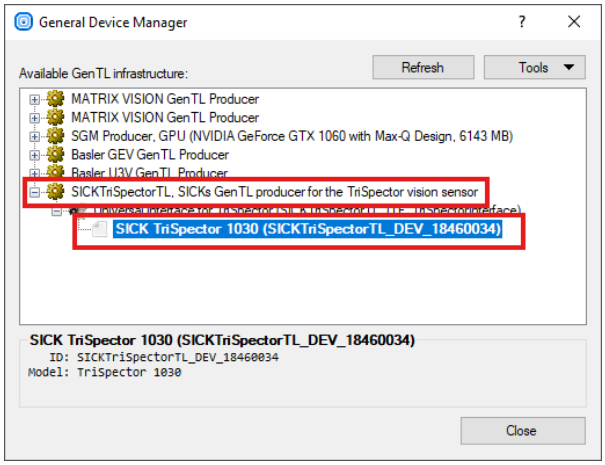
In the menu below you can adjust the settings regarding increment, image acquisition, field of vision and many more. The description of usage of that program is presented in the video tutorials provided by the producer. After that you can load the settings to the device and turn off the program.



TriSpector in Aurora Vision Studio

After all the previous steps restart or turn on Aurora Vision Studio. Check if TriSpector is connected in the **GenICam Device Manager**:



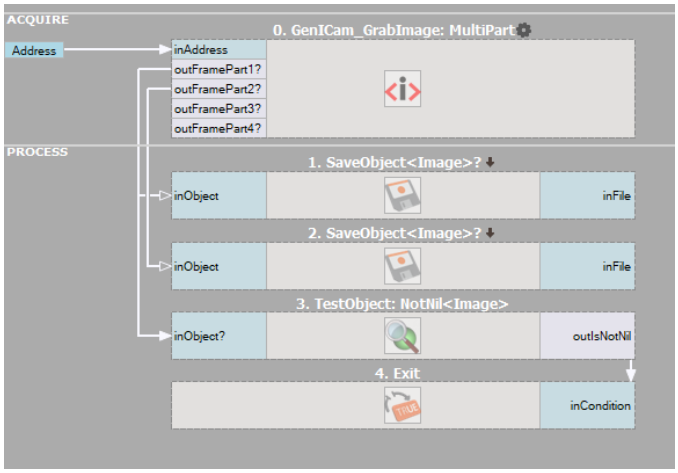


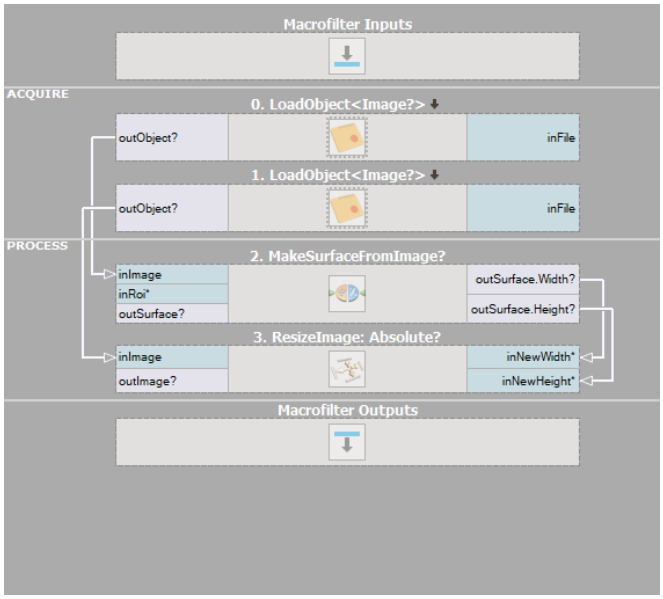
You can acquire data from the camera using GenICam_GrabMultiPartImage. By doing so you acquire two objects: a point grid and a 2D picture.

A simple program that stops when the full image form the camera is acquired is presented on the first image below. The second image is the program that loads the acquired objects, resizes them properly and allows further analysis of the image.

Parameters used in MakeSurfaceFromImage filter are as follows:

- **inXScale** – parameter X-resolution used during TriSpector configuration
- **inYScale** – parameter Profile Distance used during TriSpector configuration
- **inZScale** – number form 0,0025 to 0,005 depending on the distance to the study object





Acquiring data from AT-C6 devices

Purpose and requirement

The following instruction provides information on how to interface an AT-Sensors device with Aurora Vision Studio.

Required equipment:

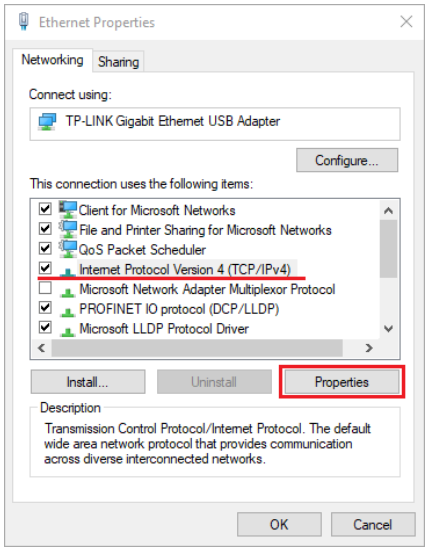
- A sensor from the AT-C6 series (C6-1280, C6-2040, C6-3070, C6-3070W, C6-4090).
- AT SolutionPackage 2023.12.
- Aurora Vision Studio 5.4 Professional or later
- cxExplorer (for AT-C6 device configuration).

PC Network settings

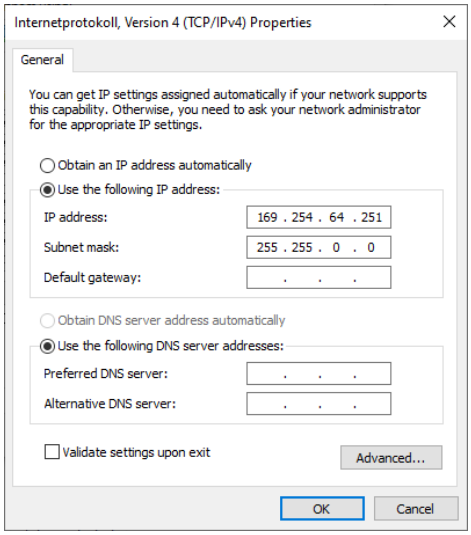
In the delivery state, the DHCP of the device is activated and the camera obtains the IP automatically. In general, the usage of static IP addresses is recommended. For this the network adapter and the device must be set to the same IP address range.

In Windows OS, the network adapter settings can be accessed by following these steps:

1. Open Control Panel on your PC.
2. Find Network and Sharing Center.
3. Choose Change adapter settings.
4. Right-click on unidentified network connection, choose Properties.
5. In Ethernet Properties find Internet Protocol Version 4 (TCP/IPv4) and click Properties.



6. In General tab, choose Use the following IP address and set:
    - IP address to (for example) 169.254.64.251
    - Subnet mask to 255.255.0.0

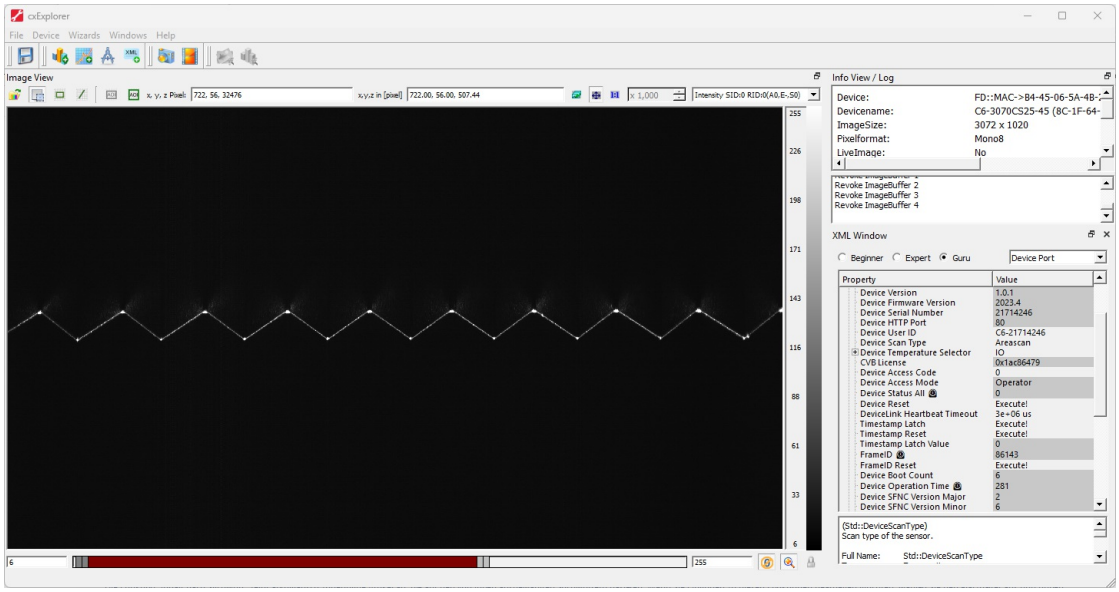


7. Click OK and Close.

AT-C6 Configuration

Configuration of an AT-C6 device can be easily done with cxExplorer, which is a graphical user interface provided by AT Sensors. With the help of the cxExplorer a sensor can be simply adjusted to the required settings.

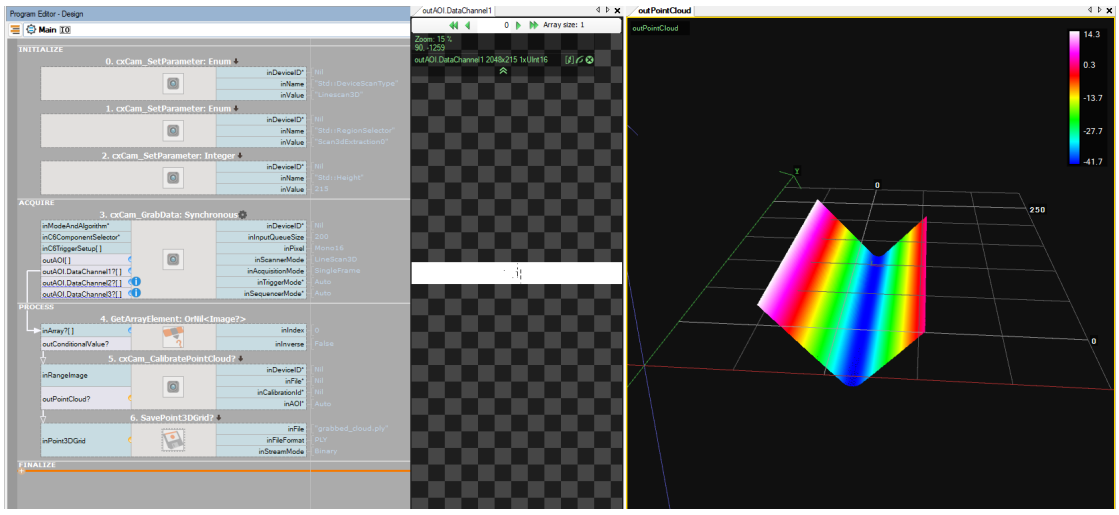
After configuring the device with cxExplorer, you can start with the Aurora Vision Studio software.



Using AT-C6 with Aurora Vision Studio

After all the previous steps restart or turn on Aurora Vision Studio. You can acquire data from the device and set its parameters using dedicated tools: cxCam.

A simple program that grabs the point cloud from the device is presented on the image below.



Testing Digital Outputs of a GigEVision Camera

Purpose and equipment

The aim of this document is to demonstrate how to control external lighting using the digital output of a GigE Vision camera in Aurora Vision Studio. The application features a WebHMI interface with a switch that allows the user to manually change the state of the camera's output pin, which in turn turns the connected light on or off.

Required equipment:

- Zebra CV60 Area Scan Camera
- EFFI-FLEX-15 LED Barlight with M12 Power – 4 pins connector
- 24V DC Power Supply
- PC with Aurora Vision Studio (version 5.7 used)

Hardware connection

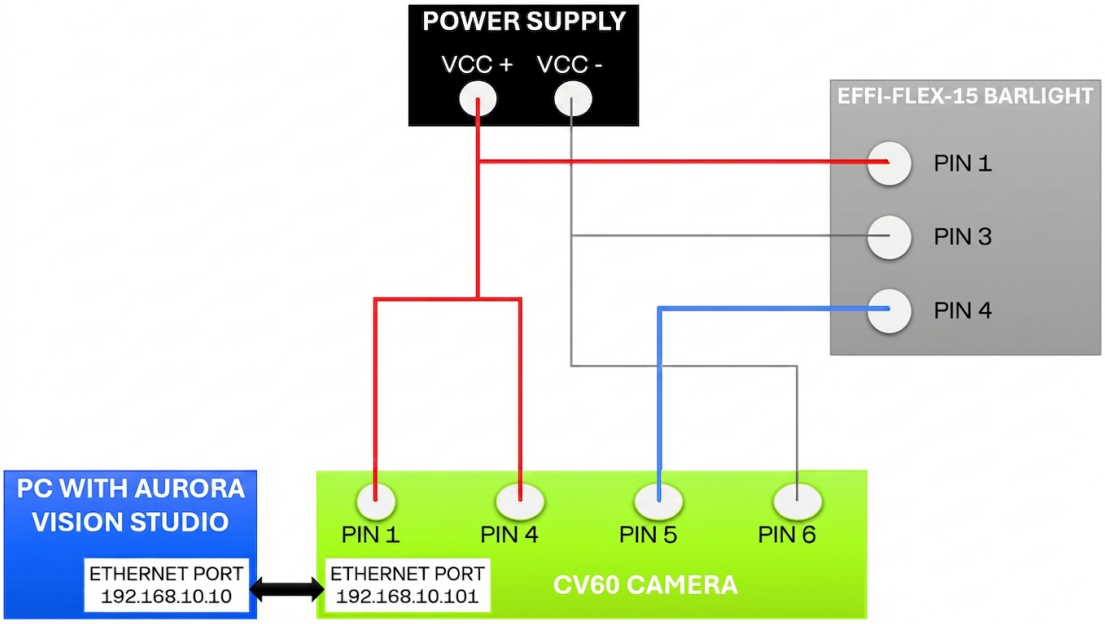
PC and camera – Ethernet connection

- Ensure that both the PC network interface card (NIC) and the GigE Vision camera are configured to operate within the same subnet. In this application note, a single network is configured. Devices connected to this network have addresses 192.168.10.XXX, where XXX stands for a number from 1 to 254, unique for every device/port connected.
- If the PC is equipped with PoE ports, the camera can be connected directly to one of them. If the PC does not support PoE, the Ethernet connection from the camera to the network card port on the PC must pass through an external PoE Injector.

Electrical circuit

To allow the camera to trigger the external lighting successfully, the electrical wiring must be configured as follows:

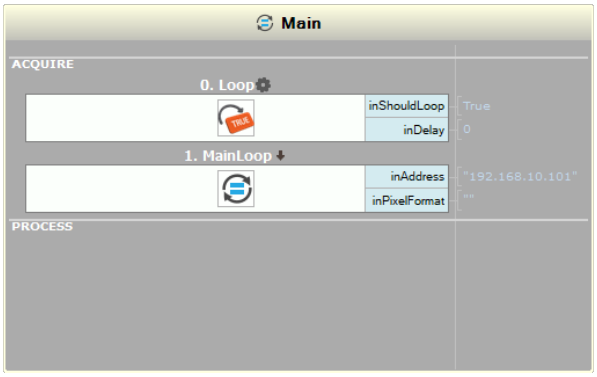
1. **The Supply Voltage (VCC, 24V)** – connect in parallel to:
    - Pin 1 (Main Power and I/O Reference) and Pin 4 (Opto Out+) of the CV60 camera
    - Pin 1 of the EFFI-FLEX-15 LED Barlight (Power)
2. **Signal Connection** – connect Pin 5 of the camera (Opto Out–) directly to Pin 4 of the barlight (Pin) to completes the switching circuit.
3. **Power Supply Ground (VCC–)** – connect to:
    - Pin 6 of the camera (Ground)
    - Pin 3 of the barlight (Ground)

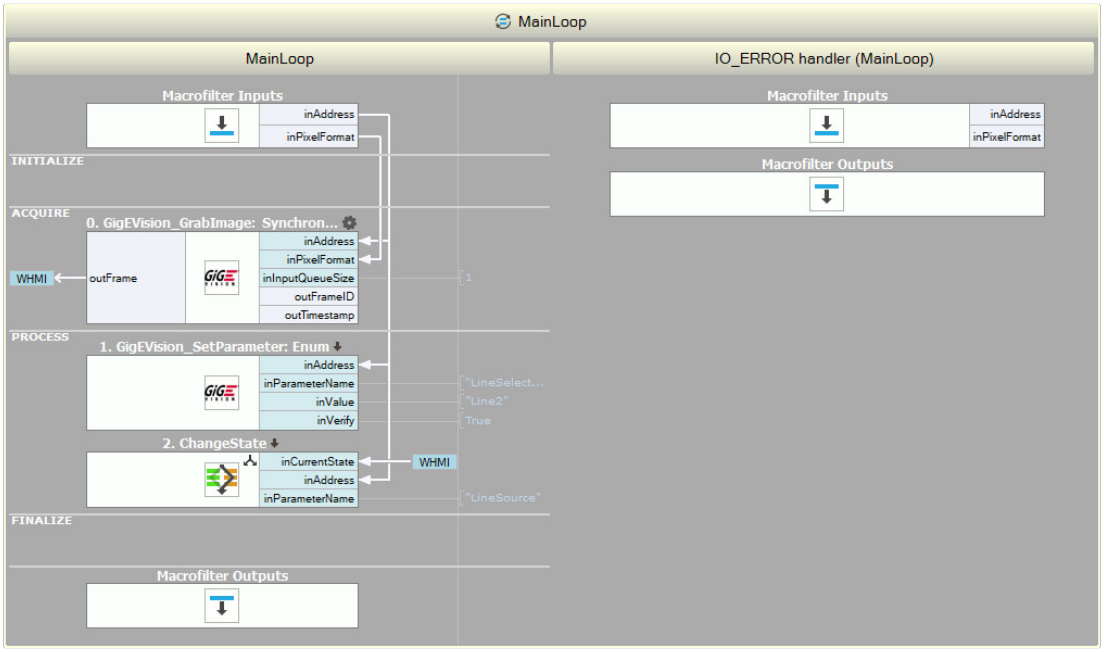


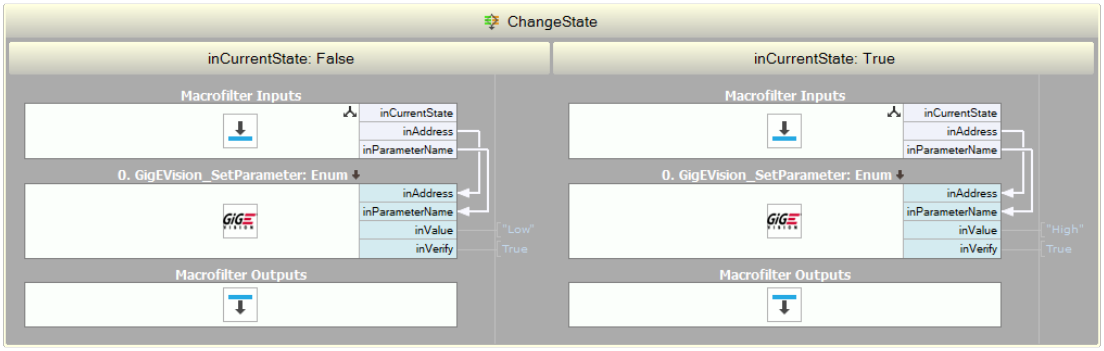
Configuring Aurora Vision Studio application

1. Inside the *ACQUIRE* section of the **Main** macrofilter, add the Loop filter with default settings.
2. Create a new *task* named **MainLoop** and put it in the *PROCESS* section of the **Main** macrofilter.
3. Inside the **MainLoop**, add the GigEVision_GrabImage filter to the *ACQUIRE* section.
4. Drag and drop the **inAddress** and **inPixelFormat** inputs from the added filter to the **Macrofilter Inputs** block.
5. Go back to the **Main** macrofilter.
6. Set the **inAddress** value of the **MainLoop** macrofilter to *192.168.10.101*.
7. Go back to the **MainLoop** macrofilter.
8. Inside the *PROCESS* section, add the GigEVision_SetEnumParameter filter. Set the parameters as below:
    - inParameterName: **LineSelector**
    - inValue: **Line2**
9. Create a new *variant* macrofilter named **ChangeState** with the fork port **inCurrentState** of type *bool*.
10. Drag and drop the **inAddress** output from the **Macrofilter Inputs** block to the **ChangeState** macrofilter.
11. Inside the **ChangeState** variant macrofilter:
    - **False**: add the GigEVision_SetEnumParameter filter. Drag and drop its **inParameterName** input to the **Macrofilter Inputs** block. Set the **inValue** parameter to **Low**.
    - **True**: add the GigEVision_SetEnumParameter filter. Connect its **inParameterName** input to the **inParameterName** input from the **Macrofilter Inputs** block. Set the **inValue** parameter to **High**.
12. Go back to the **MainLoop** macrofilter.
13. Create a WebHMI panel that contains the PictureBox and the Switch controls. You can use one of the WebHMI official templates.
14. Connect the PictureBox input to the **outFrame** output of the GigEVision_GrabImage filter.
15. Connect the Switch output to the **inCurrentState** input of the **ChangeState** macrofilter.
16. Add the IO_ERROR handler to the **MainLoop** macrofilter.

The final program should look like this:

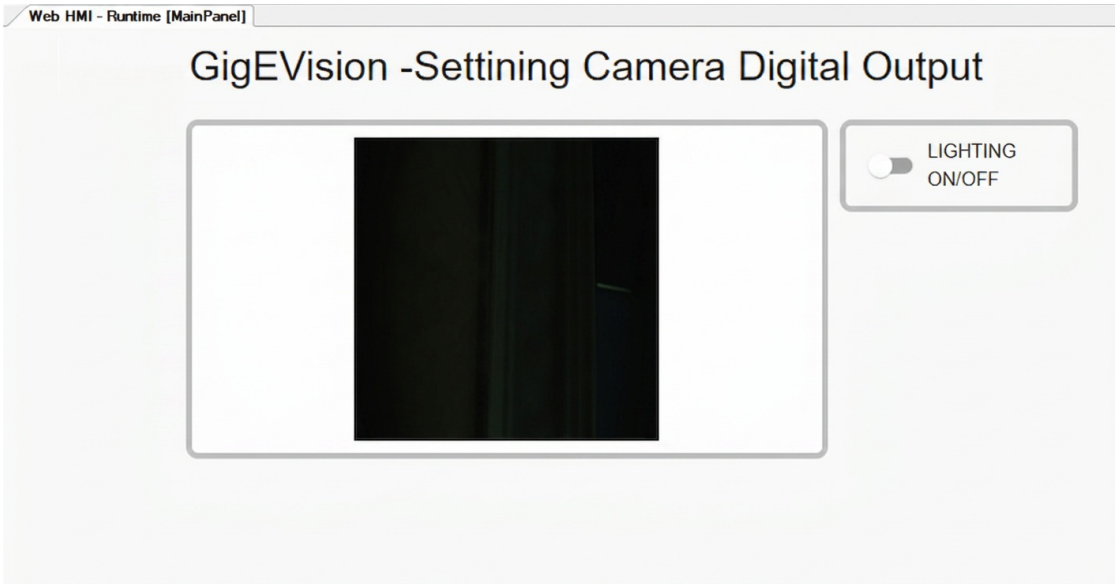




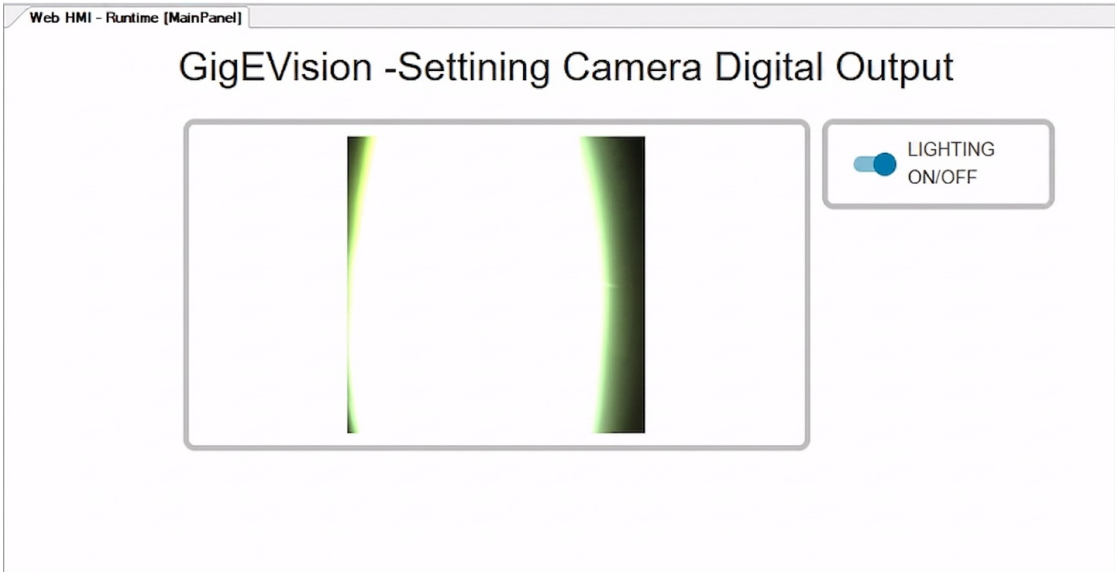


Results

The lighting is off:



The lighting is on:



References

- Zebra CV60 Connector Diagrams
- EFFI-FLEX-15 LED Barlight Connector Diagrams

Using TCP/IP Communication

Purpose and equipment

This document describes how to a make practical example of TCP/IP communication between Aurora Vision Studio and a PLC device. For this case Aurora Vision Studio 4.11 and Simatic S7- 1200 CPU 1212C AC/DC/RLY with TIA Portal V15 have been used. For more theory you can check out TCP/IP Communication article.

Required equipment:

- SIMATIC S7-1200 controller or other from SIMATIC S7 family
- TIA Portal V15
- Aurora Vision Studio 4.11 Professional or later

Overview and first steps

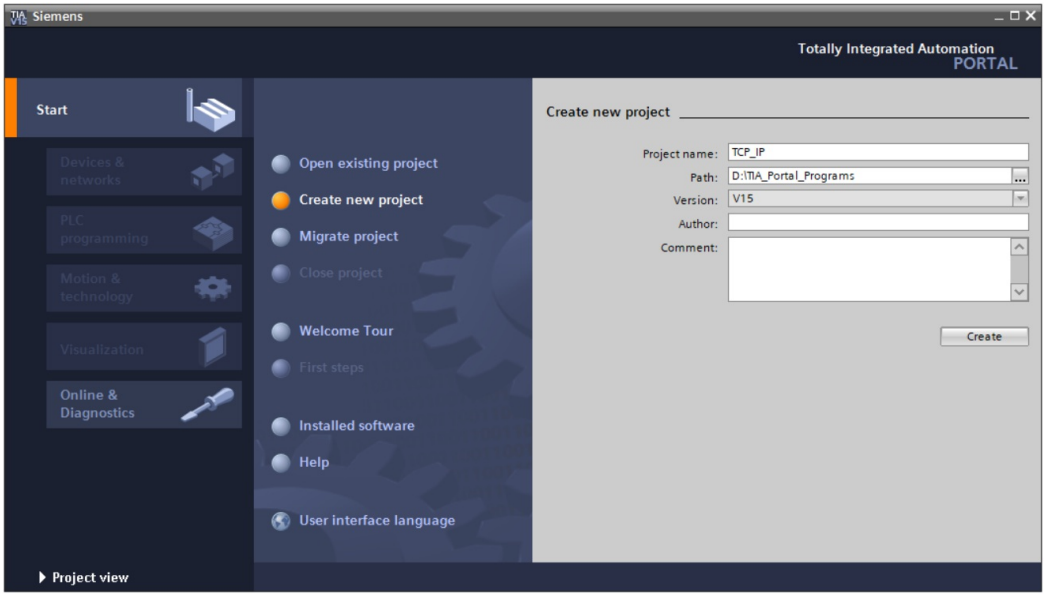
In this example, you will create an application which sends a number from Aurora Vision Studio to a PLC and receives the square of this value calculated by the PLC. The application will be configured as a master and the PLC as a slave.

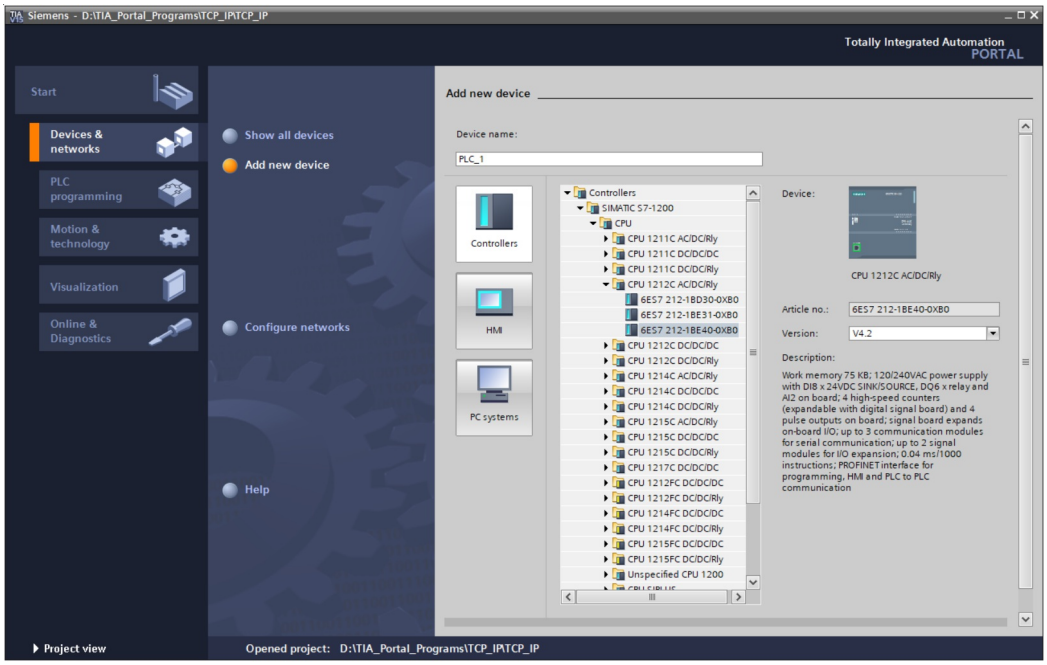
First, you need to make a hardware connection between the PLC and the PC. Network configuration will be described in the next chapter.

Network configuration

Main network configuration

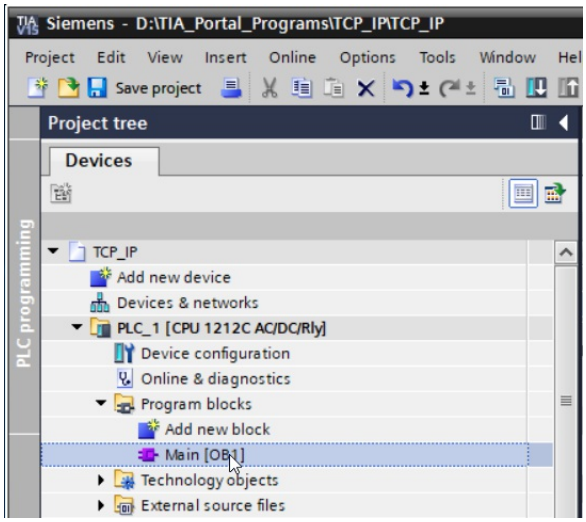
Create a new project in the TIA Portal environment and add your device by double-clicking on *Add new device*. Select your controller model, its CPU etc. in the list of controllers available in a dialog box. Click on *OK* button if you are ready.

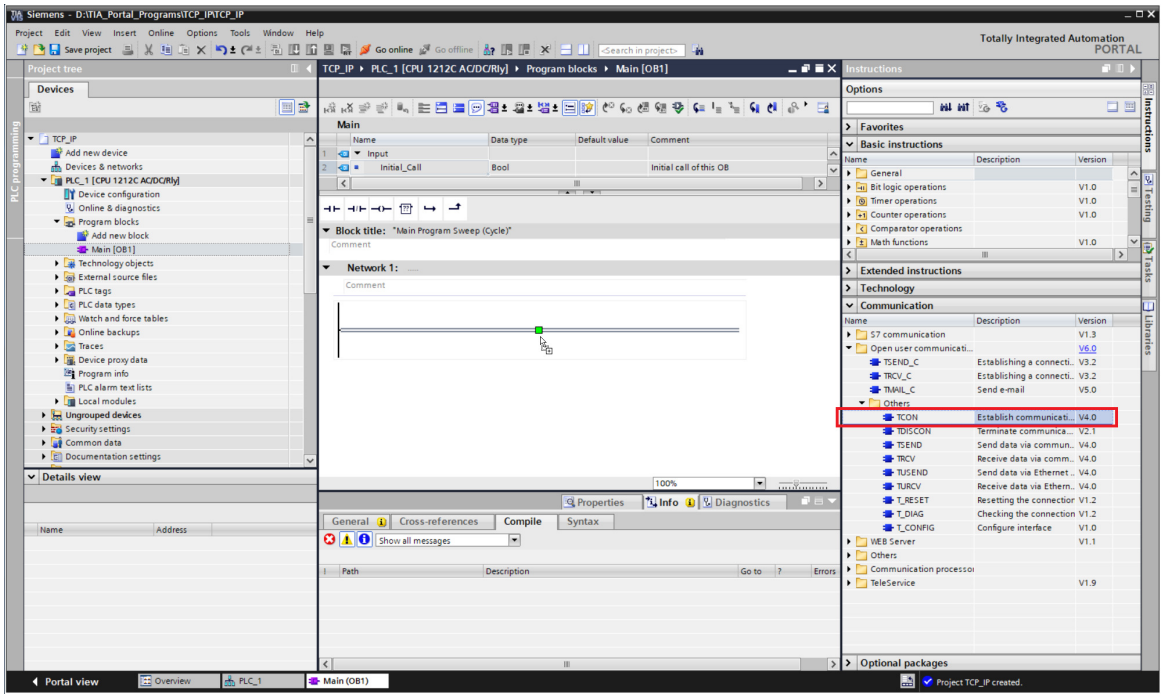




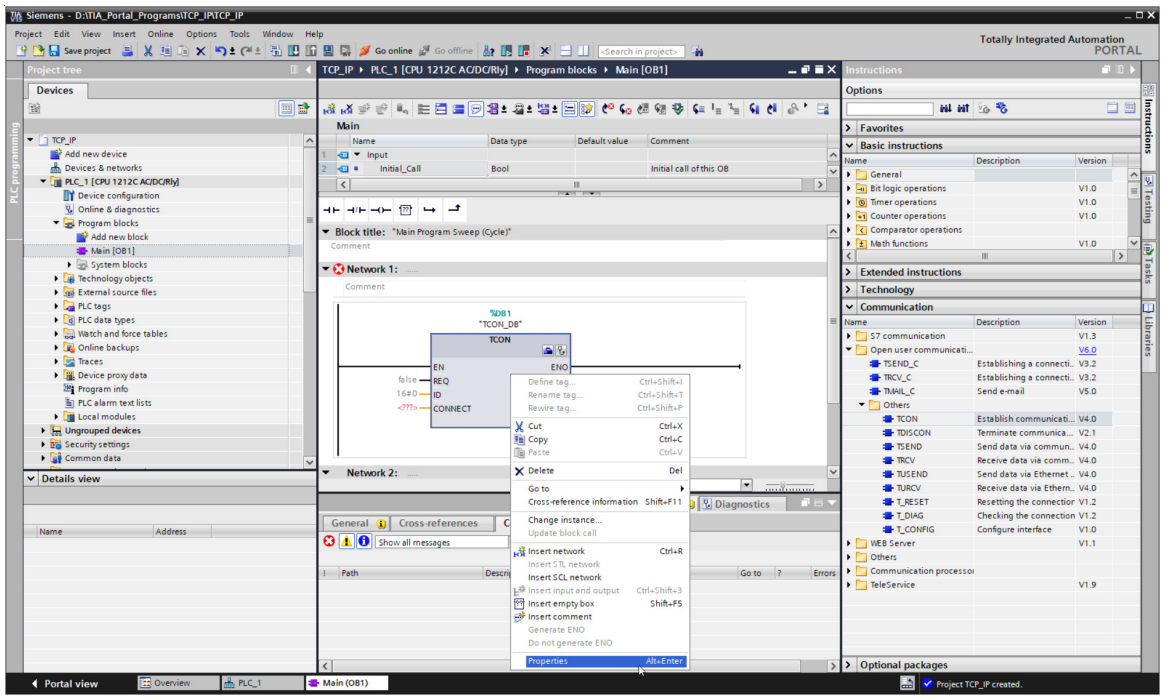
Expand a *Program blocks* list and double-click on the *Main* icon. Now you should see a network view. Find the Communication tab on the right side and double-click on the TCON icon available in

Communication -> Open user communication -> Others.



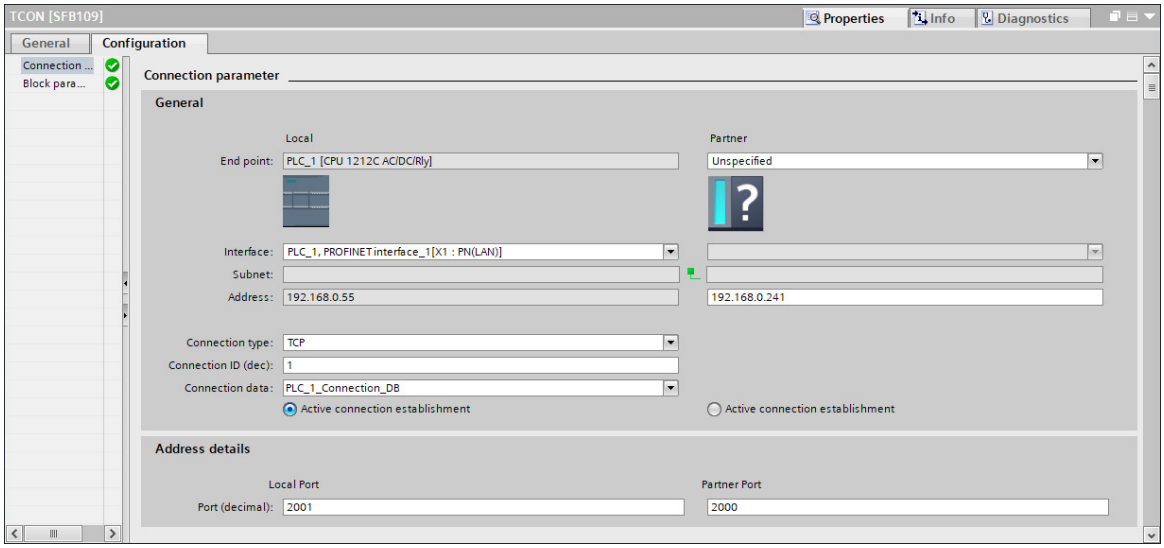


The TCON instruction is used for establishing the TCP communication. It is now visible in the *Network View*. Right-click on it and select *Properties*.



In the *Configuration* tab, you need to set the IP address of your PC. To check the IP address, you can

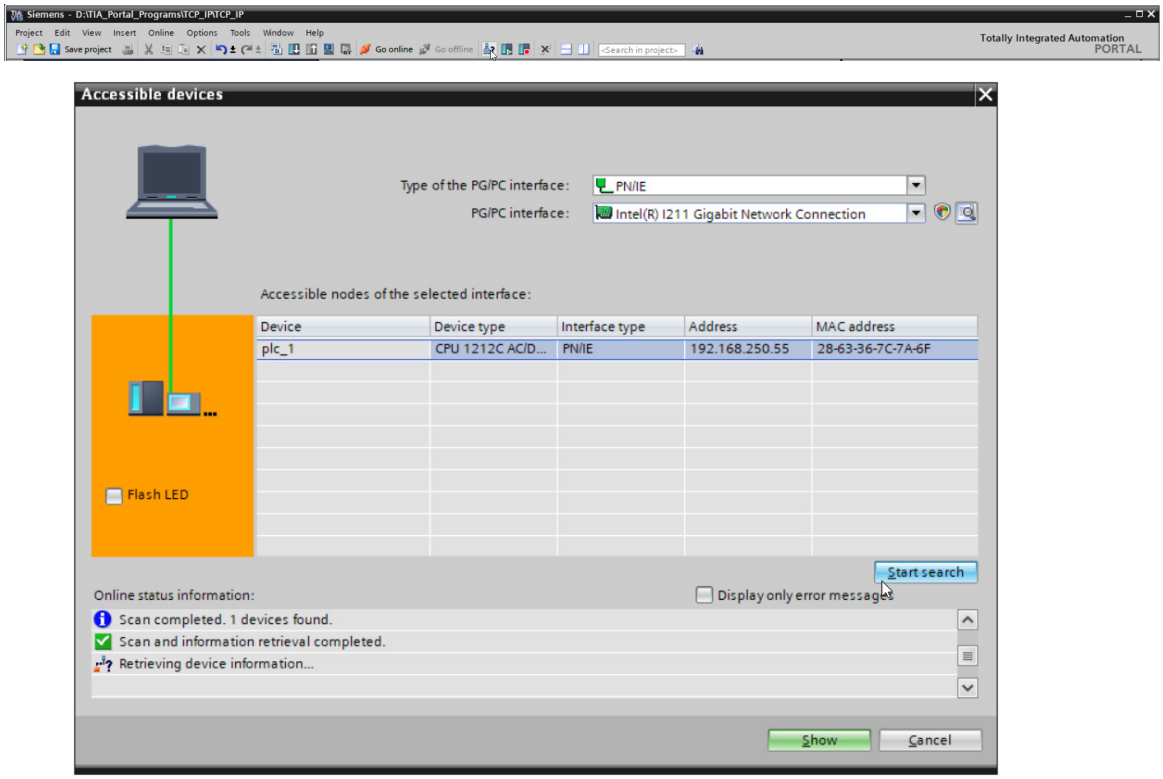
use the Command Prompt and the **ipconfig** command. Other parameters should be set like in the next figure. If you use these settings, the IP of the PLC device should be set automatically. If you cannot establish the connection, please follow steps described in next chapter about troubleshooting.



Troubleshooting IP addressing

If you have not been able to properly set the IP address of the PC, as described in the previous chapter, you should set a static IP following steps described below or otherwise, feel free to skip this step.

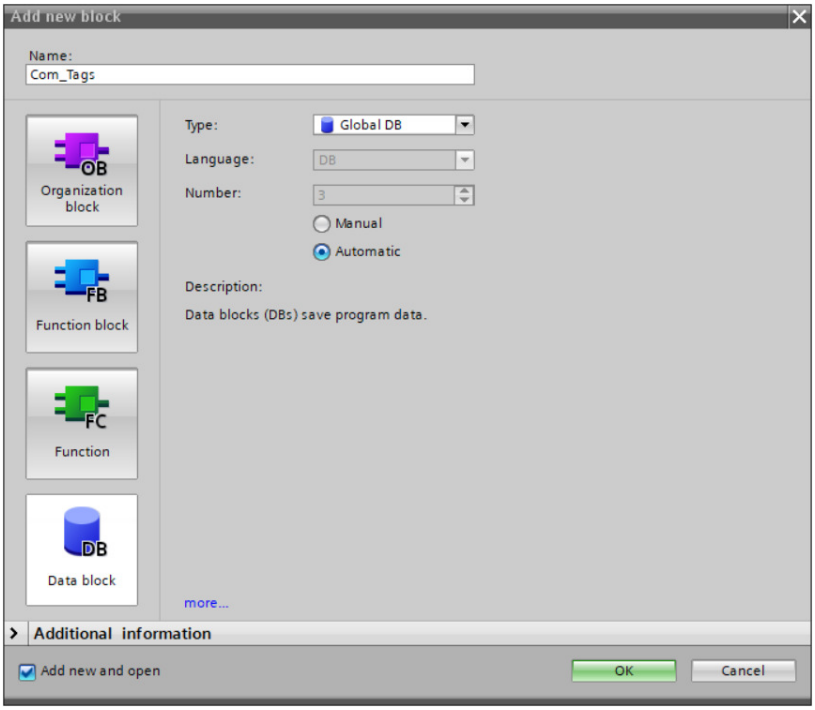
Choose *Accessible devices* from a toolbar to open the configuration window. Select *PN/IE* type of the *PG/PC* interface and select the Ethernet card connected to the PLC. Click the *Start search* button, choose your PLC from the list of accessible nodes and click on the *Show* button. You should get a new static IP which you can use in steps from the previous chapter.



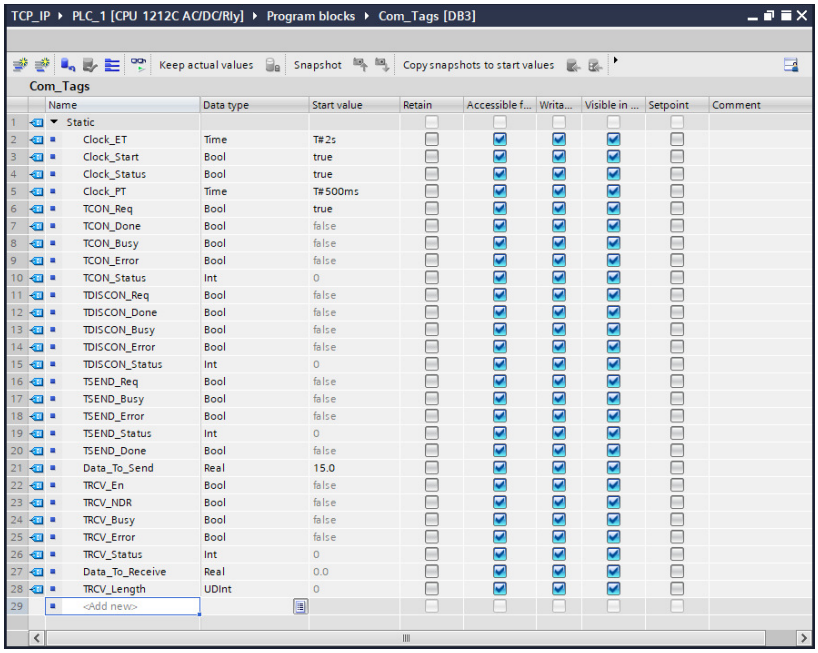
Creating a PLC program

This chapter describes how to create a PLC program establishing TCP/IP communication.

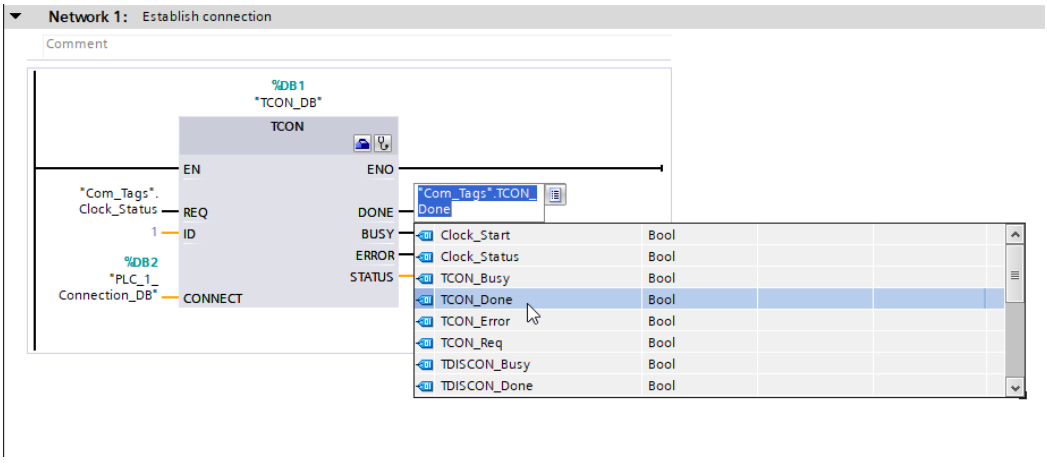
You should already have the TCON function block in the *Network view*. Now it is time to create a *Variable Table* by double-clicking on the *Add new block* in the *Project Tree* like it is shown here. Select the Data block (DB) and name the block as shown in the image below.



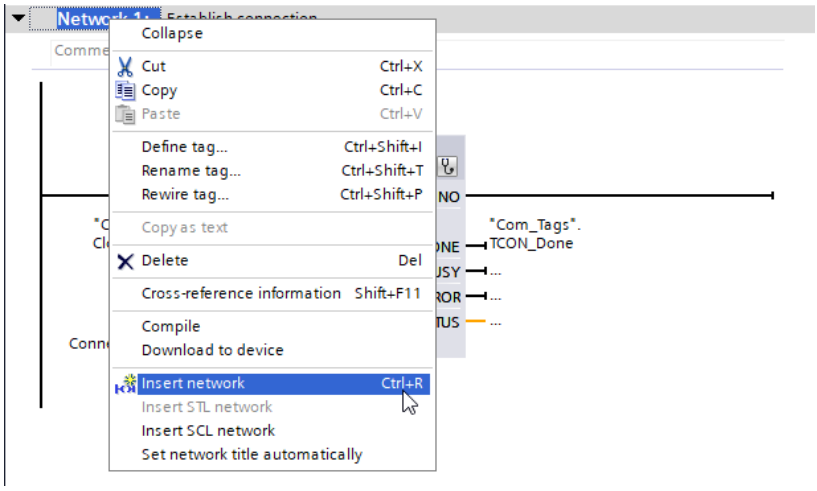
Now define variables in the *Data block*. Use the same names and data types as shown in the image below. If you want to create a new row, just right click on any row and select *Insert row*. Please note that setting the right values of *Start Values* is essential in this step.



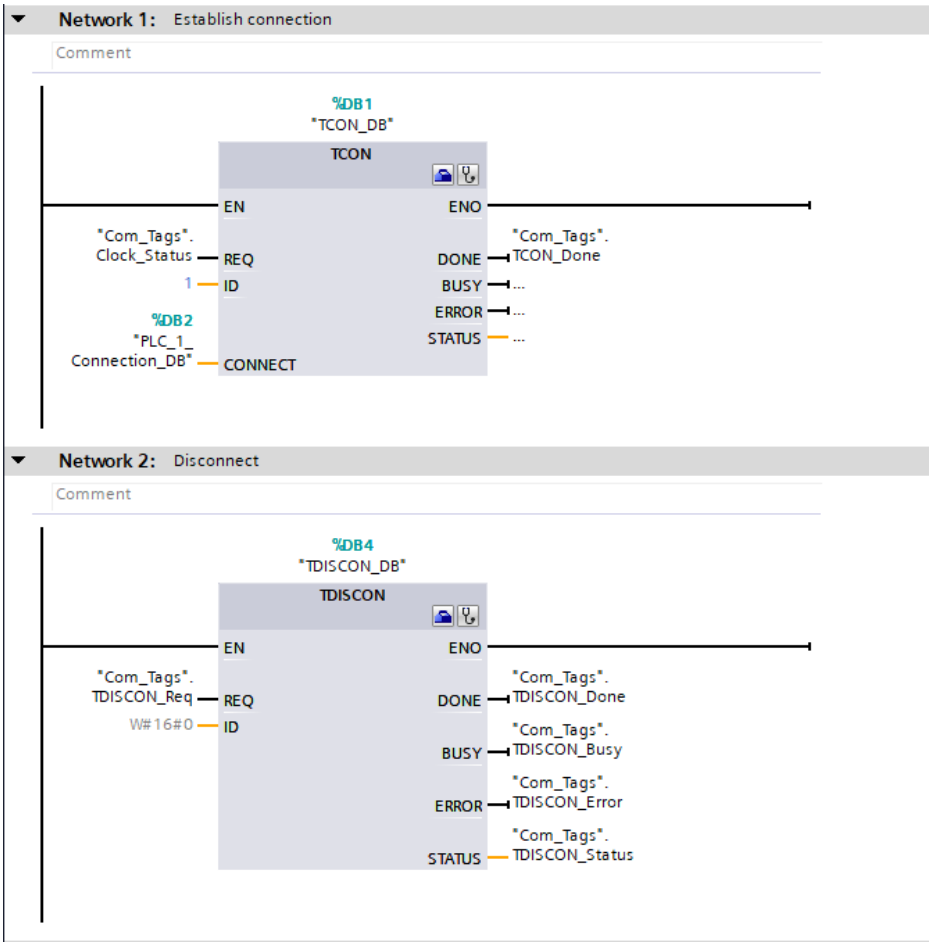
Label all the inputs and outputs of the TCON block in the network view. To label connection drag and drop variables from Data block to the Program block (for example TYCON) or double click on a connection and select displayed icon like in the image shown below.



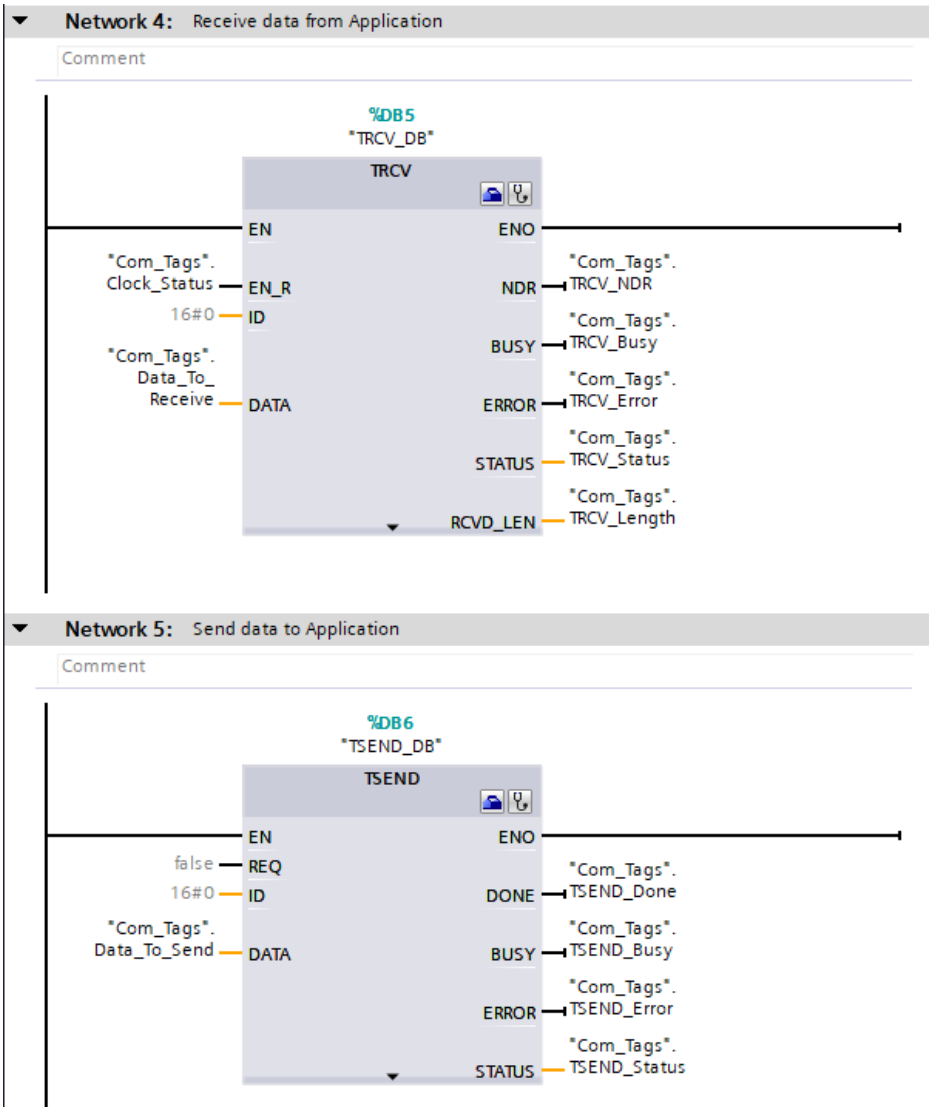
Add 5 networks to the Main program. In order to do that right-click on the existing network and select Insert network as shown in the image below:



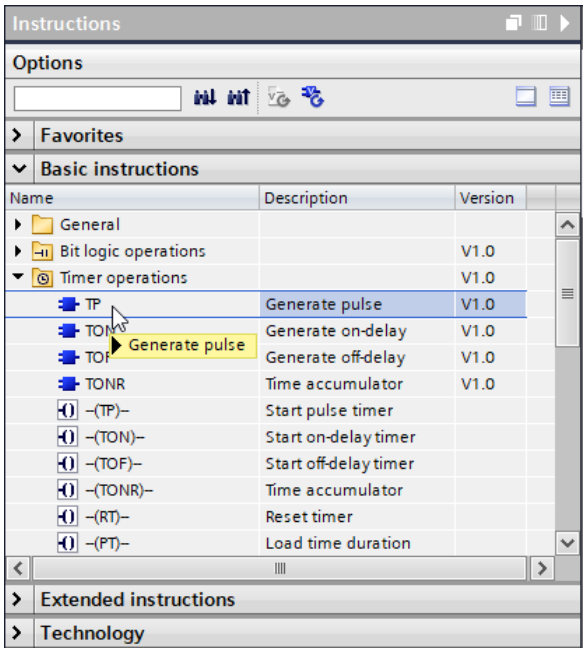
Insert additional communication block, TDISCON, from the Communication tab like on this previous picture. Label all connections as shown in the image below. TCON block will be used to establish TCP/IP connection while TDISCON will be used to close the connection.



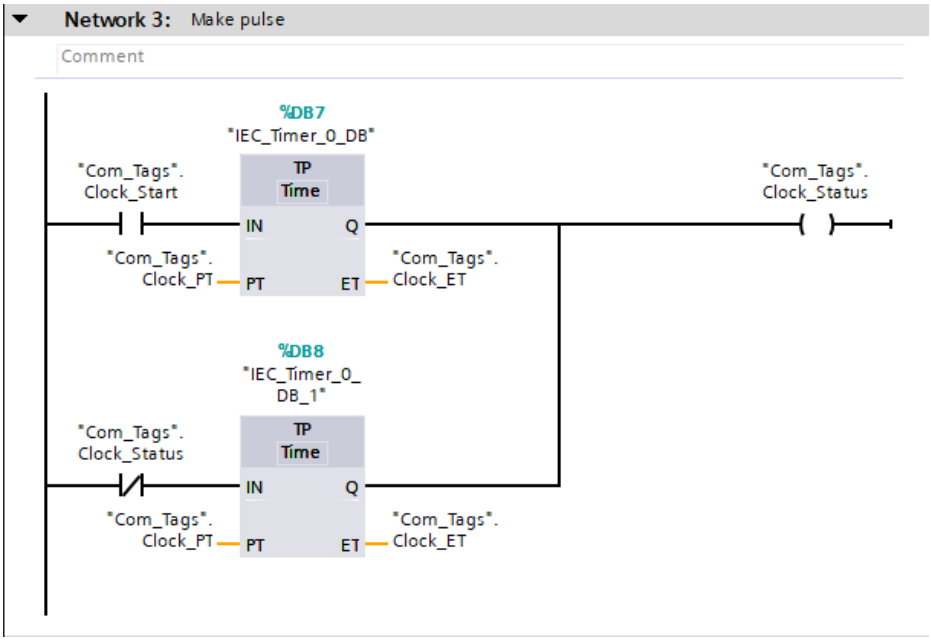
The next step is to allow exchanging data between Aurora Vision Studio and PLC. Use TRCV block for receiving messages and TSEND block for sending messages to Aurora Vision Studio. Label added blocks as shown in the image below:



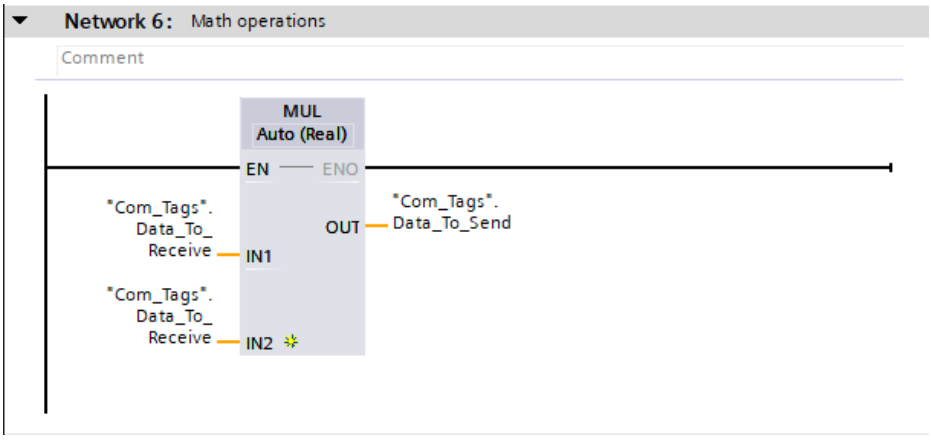
To start connection over TCP/IP using TCON you need to use rising edge signal on REQ input. Same applies to other function blocks (in TRCV a rising edge signal should be set on EN_R input). You can switch these values manually by right-clicking on the connection and selecting *Modify*. In this sample application, an automatic pulse generator will be used in order to avoid switching the values manually. Insert a new network and add a TP block located in the *Basic instructions* tab inside the *Timer operation* folder (picture below).



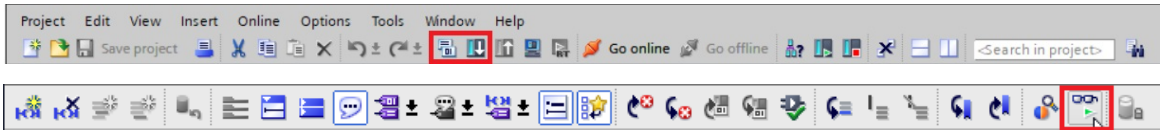
Use configuration from next picture to create a network which will be generating a proper signal.



In the last step please add a sample math function e.g. multiplying. In this example a number received from Aurora Vision Studio will be squared and the result calculated on the PLC's side will be sent back to Aurora Vision Studio.



If all previous steps have been done correctly, your PLC program is ready. Compile the program and download it to the device. To see current states of variables, turn on *Monitoring mode*.



Creating application in Aurora Vision Studio

In this chapter, the individual steps how to create an Aurora Vision Studio application to communicate with a PLC over TCP/IP are described. Before proceeding to the next steps, make sure that all previous steps have been done correctly.

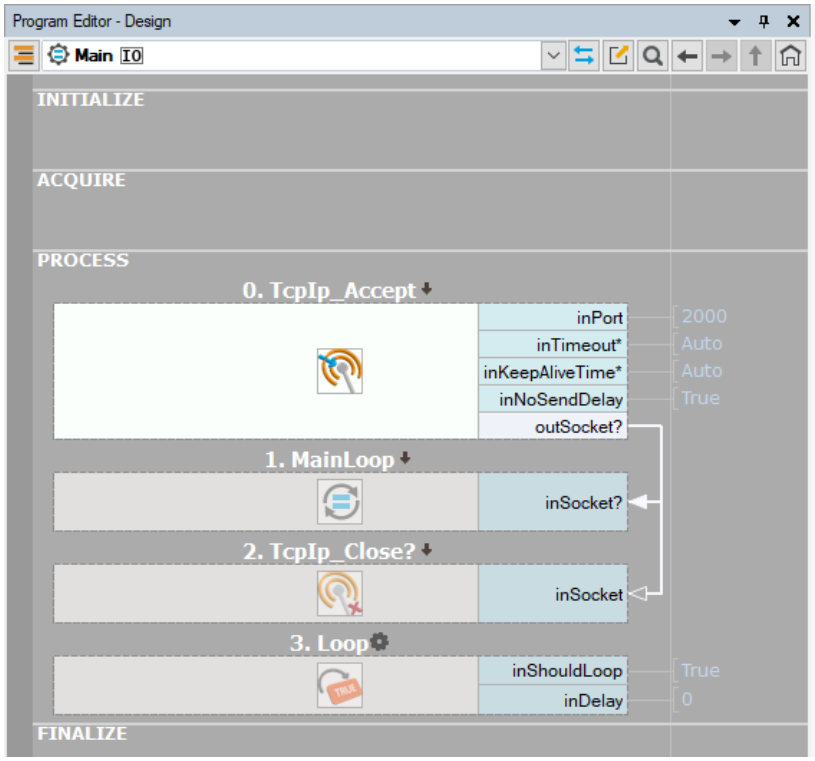
Aurora Vision Studio provides a set of ready-to-use filters for communication over the TCP protocol. A full list of the filters is available here. To get more detailed information on how to work with TCP filters, please refer to our official example *IO Simple TCPIP Communication* and the article Using TCP/IP Communication in the Aurora Vision Studio documentation.

First of all, create a new project and add a TcpIp_Accept filter. This filter, as well as other TCP/IP filters, are available in the Program I/O category of a Toolbox. Please note that TcpIp_Accept filter accepts a connection from a remote client, so in this case the Aurora Vision Studio program will be working as a server. If you want the program to work as a Client, the TcpIp_Connect filter should be used.

Set inPort value to 2000 to match configuration from PLC. Insert a TcpIp_Close filter below the TcpIp_Accept and connect the **outSocket** output with the **inSocket** input as shown in the image below. If the connection has been configured properly, you should be able to step over all steps in program.

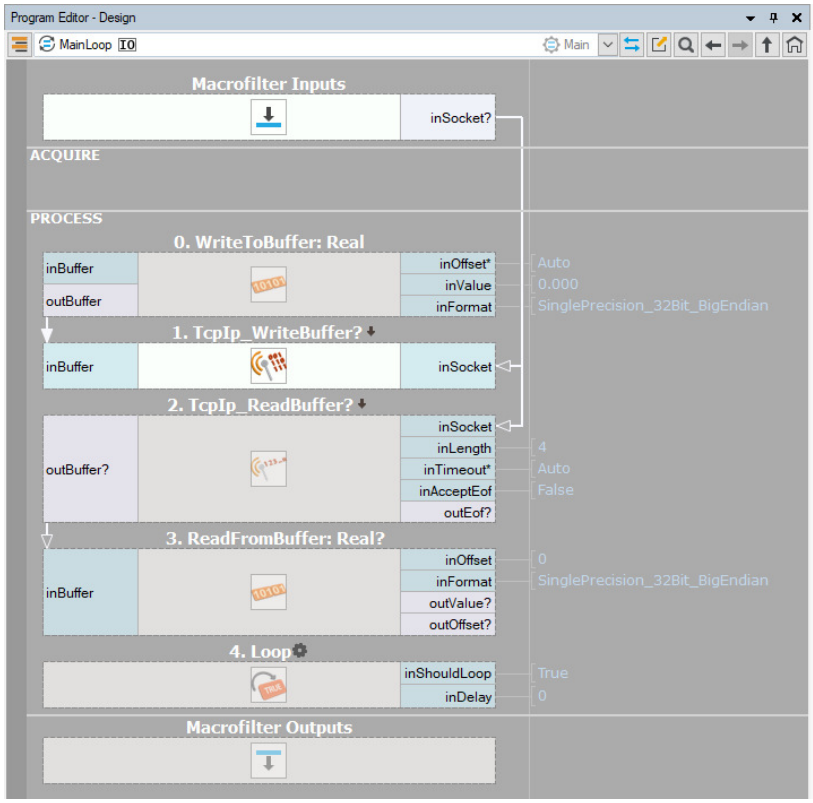


Create a new Task Macrofilter and create an **inSocket** input (you can do this by dragging the outSocket and dropping it on the macrofilter). Connect **outSocket** from TcpIp_Accept filter to the **inSocket** input of *Task Macrofilter*. In this example Aurora Vision Studio will connect over TCP/IP only once and the connection will be held. The data exchange will be executed inside the Task Macrofilter.

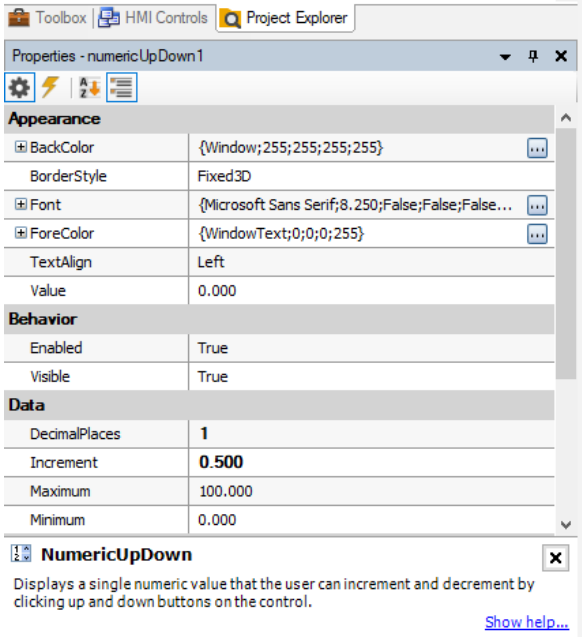


Inside the *MainLoop* macrofilter you need to add the TcpIp_WriteBuffer filter to send messages and TcpIp_ReadBuffer to receive messages over TCP/IP. PLC program works on *Buffer* data type, in our example user will be specifying decimal numbers, therefore conversion from *Buffer* to Real value is needed. To do so, use the WriteRealToBuffer and ReadRealFromBuffer filters which automatically convert decimal value into specified binary representation and write it to/read it from a buffer.

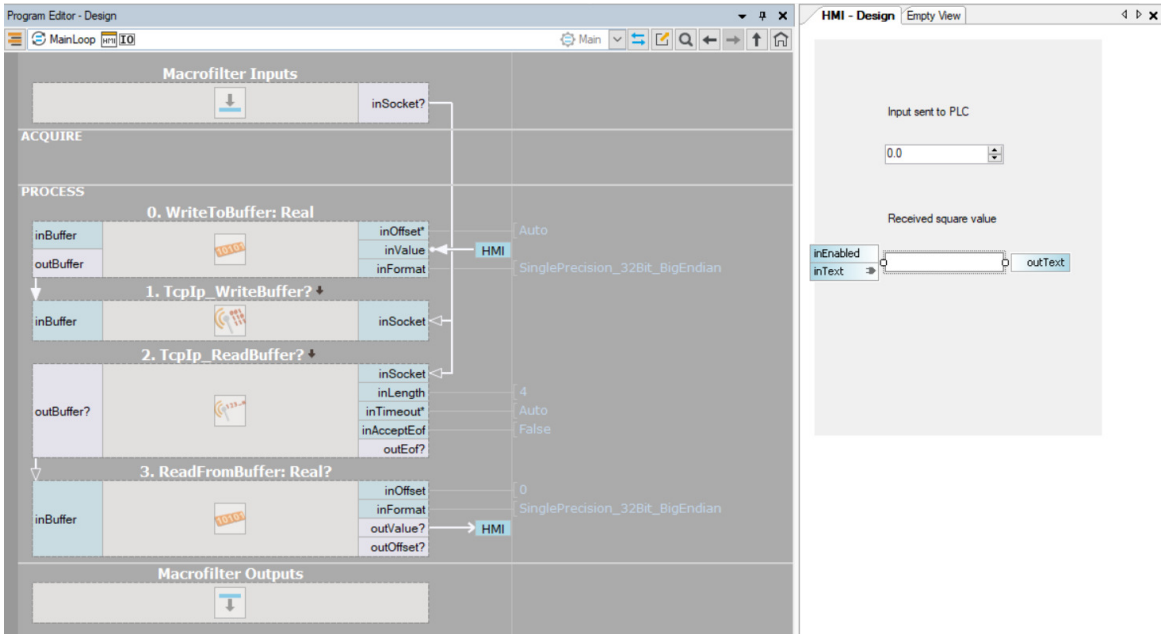
Add the Loop filter at the end of the algorithm inside *MainLoop* macrofilter. Make sure to set proper format – *SinglePrecision_32Bit_BigEndian*, otherwise program will not work properly. The algorithm should look like the one shown in the next picture.



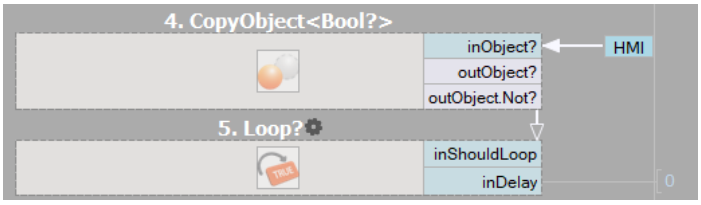
In the next step simple HMI will be created. Add a NumericUpDown and TextBox controls from the *Controls* tab. Connect NumericUpDown **outValue** output to the WriteRealToBuffer **inValue** input (previous picture) and the output **outValue** of ReadRealFromBuffer filter to the **inText** of the TextBox control. Set the properties of the NumericUpDown HMI control as shown in the next picture.



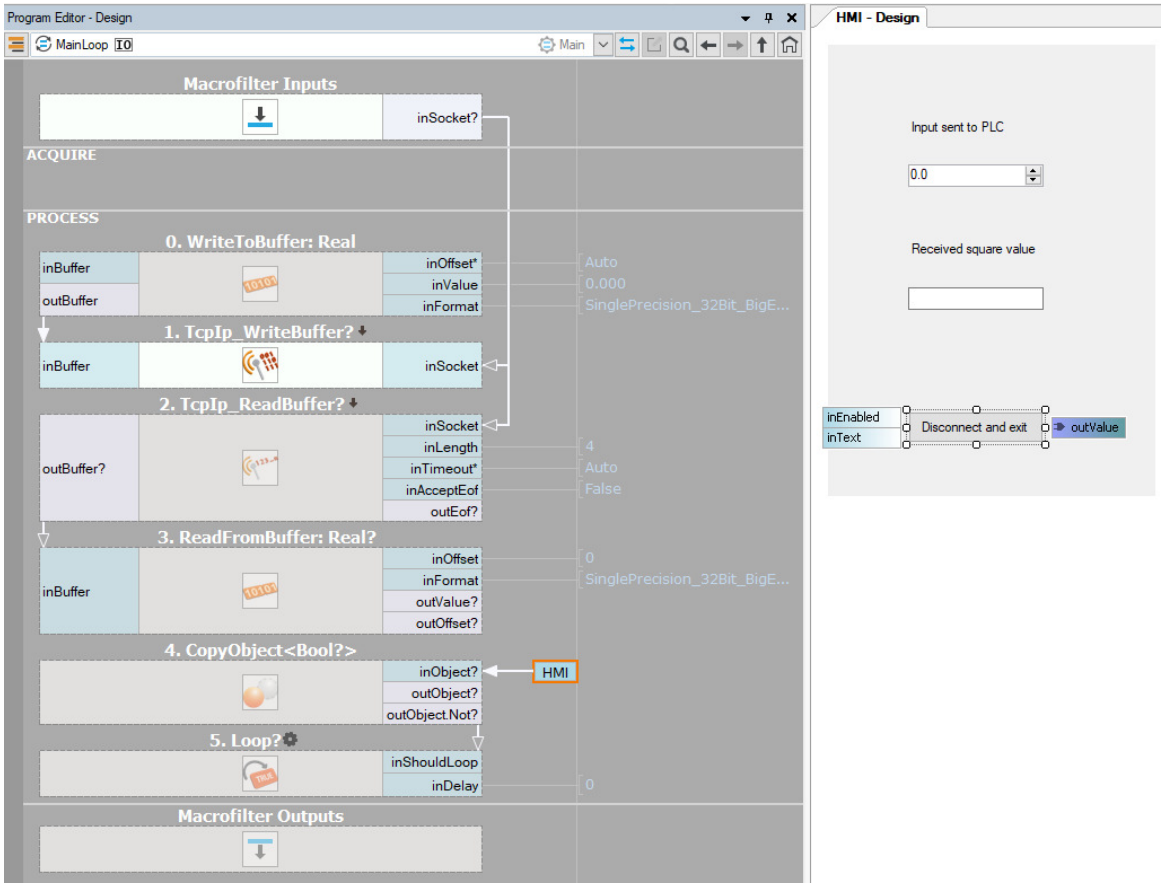
In the last step use Label control to describe the previously added controls. The current algorithm should look like the one shown in the next picture. The program can work in current state; however, it will be improved in next steps.



Main Loop is running continuously due to the **inShouldLoop** input of Loop, which is always true in the current structure. In order to change that, add ImpulseButton and name it "Disconnect and exit". Loop filter generates loop when true value is passed to the **inShouldLoop** input. Default state of ImpulseButton **outValue** output is false, so this value should be negated before connecting it to the Loop filter. Insert the CopyObject filter inside *MainLoop* macrofilter and set the Bool? data type. Connect ImpulseButton **outValue** to the **inObject** input of the CopyObject filter. Right click on the **outObject** output, select *Property Output* and *Not* variant. Now connect **outObject.Not** output to the **inShouldLoop** input as shown in the next picture.

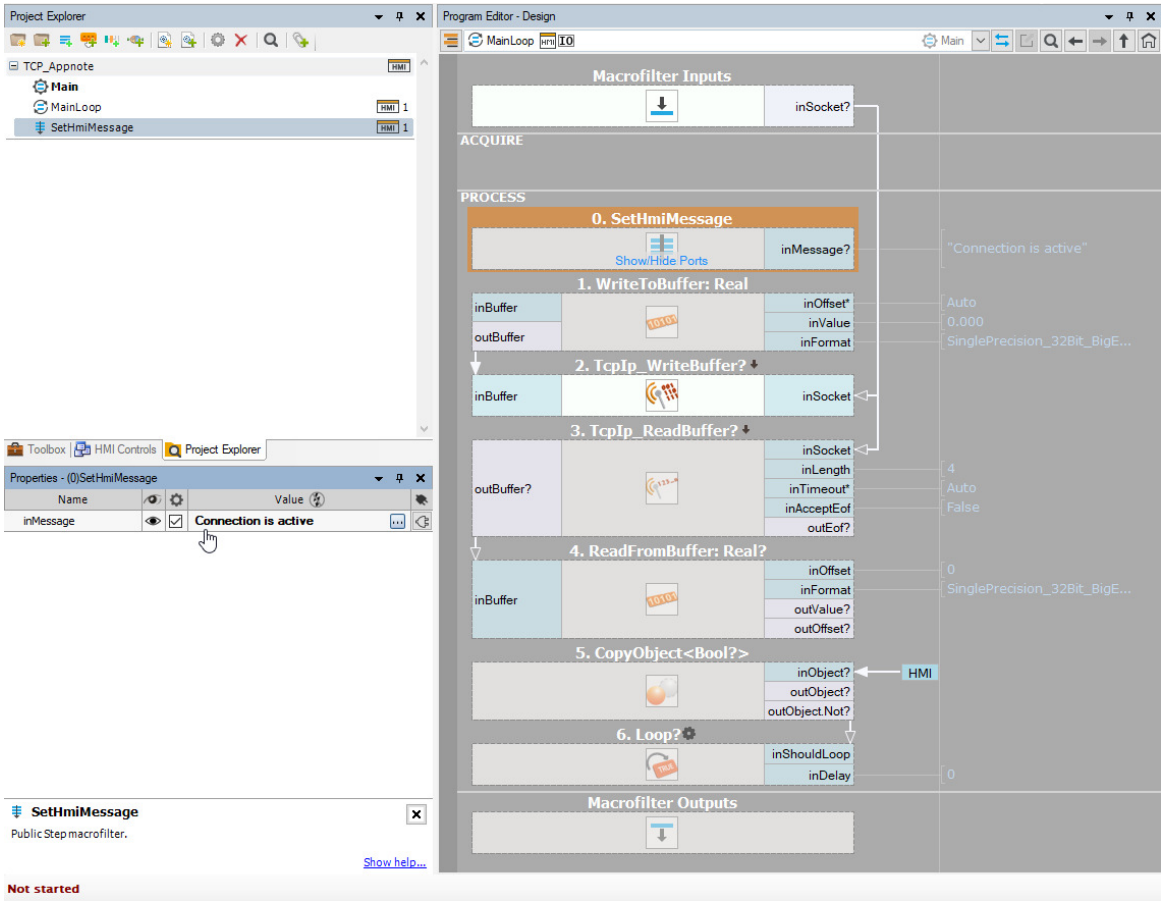


The same *ImpulseButton* control will be used to Close TCP/IP connection in Main task. Use the structure known from description above and Figure 25 to control Loop filter in Main task as shown in the image below:

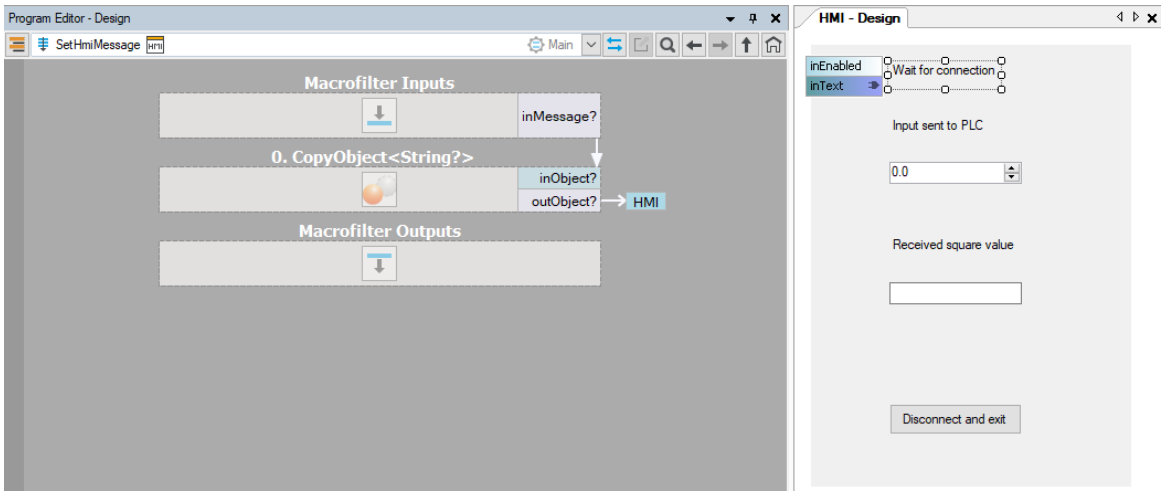


The application is almost ready, but every good and stable application should have Error handling. Next modification will allow you to see the current connection state, for example: wait for

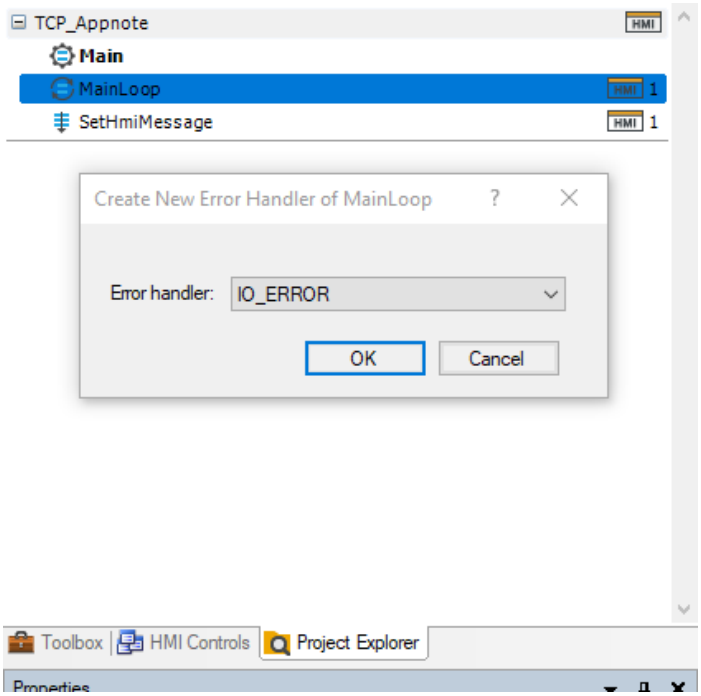
connection, connection is active, and lost connection. In order to do that, go inside *MainLoop* macrofilter, right click on the program editor window and insert a step macrofilter. Name new created macrofilter as "SetHmiMessage" and create new String? type input. Name the newly created input as **inMessage**. This macrofilter will be used in a few places is this application, in case you use it inside *MainLoop* macrofilter, insert command "Connection is active" in macrofilter properties as shown in image below:



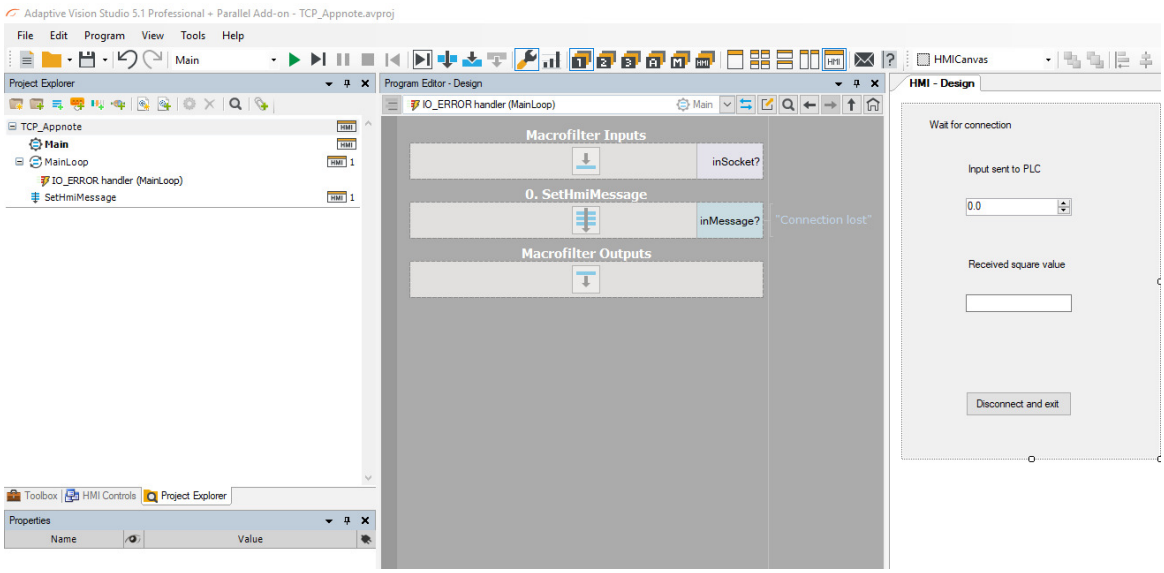
Enter *SetHmiMessage* macrofilter and add the CopyObject filter with String? type. Add a new Label control to the HMI window. As default text set "Wait for connection". Connect the **outObjects** output from CopyObject filter to the **inText** input of Label control. The result should look like in the image below:



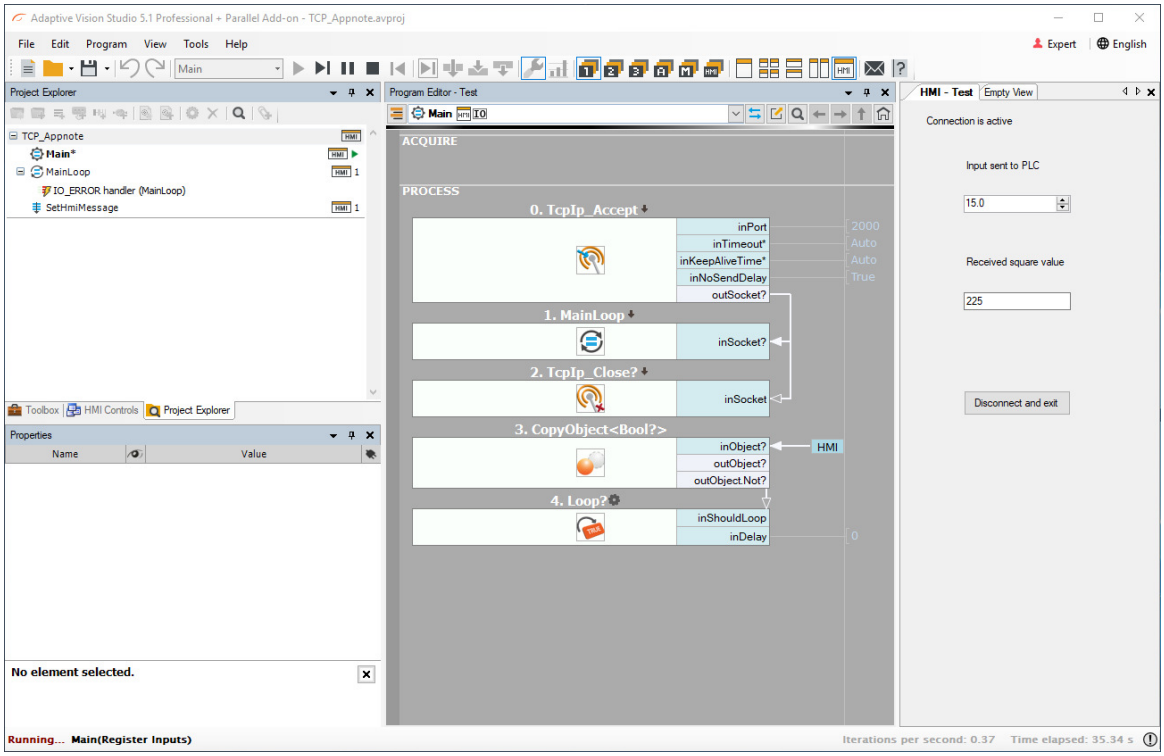
Currently, connection state message has two different variants: Wait for connection and Connection is active. Now, we are going to add message which will be prompted when connection is broken. In order to do that, you need to use Error handlers for *MainLoop* macrofilter. Right click on the MainLoop macrofilter, visible in the Project Explorer window, and select *Add New Error Handler...* Choose IO ERROR from the list, as shown in the next figure.



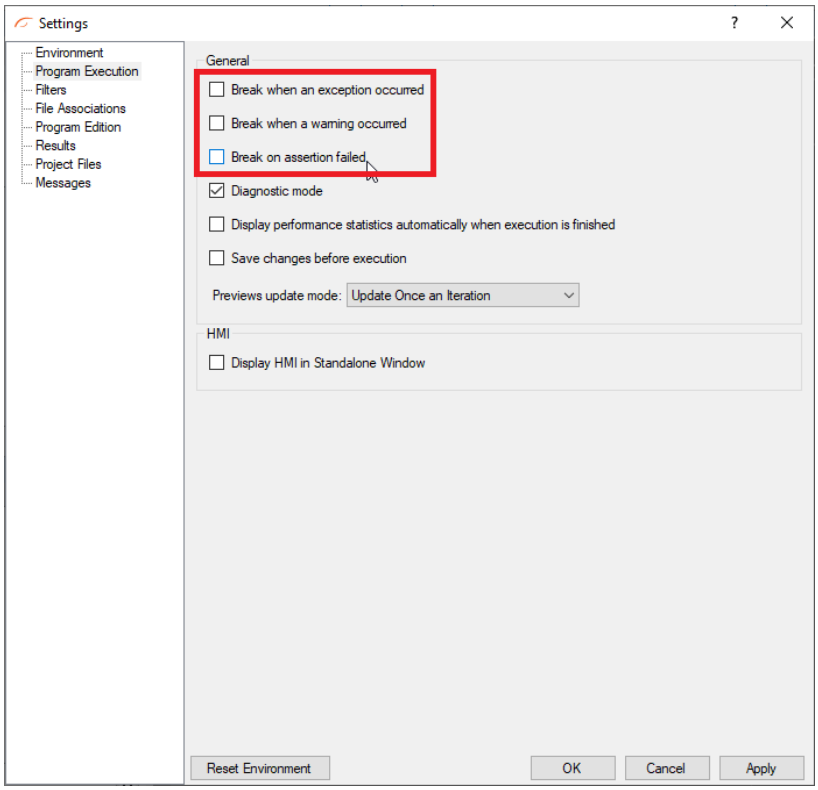
Enter newly created error handler and add *SetHmiMessage* macrofilter. Set **inMessage?** input to "Connection lost" as shown in the figure below:



The result should look like in the following picture. If all previous steps have been done correctly, program should work without any problems.



If you want to test program with occurring errors, you should unlock breaks in settings, as shown in the image below. With these settings, a pop-up window with error messages will not appear.



Using Modbus TCP Communication

Purpose and equipment

This document describes how to make a practical example of establishing Modbus TCP communication between Aurora Vision Studio and a PLC device or a Modbus TCP simulator. Aurora Vision Studio 5.2 and the Simatic S7- 1200 CPU 1212C AC/DC/RLY controller with TIA Portal V15 were used to accomplish this task. For more theoretical background refer to the Working with Modbus TCP Devices entry.

Required equipment:

- Modbus TCP simulator, e.g. EasyModbusTCP
- SIMATIC S7-1200 or other controller from the SIMATIC S7 family
- TIA Portal V15
- Aurora Vision Studio 5.0 Professional or later (version 5.2 was used)

Overview and first steps

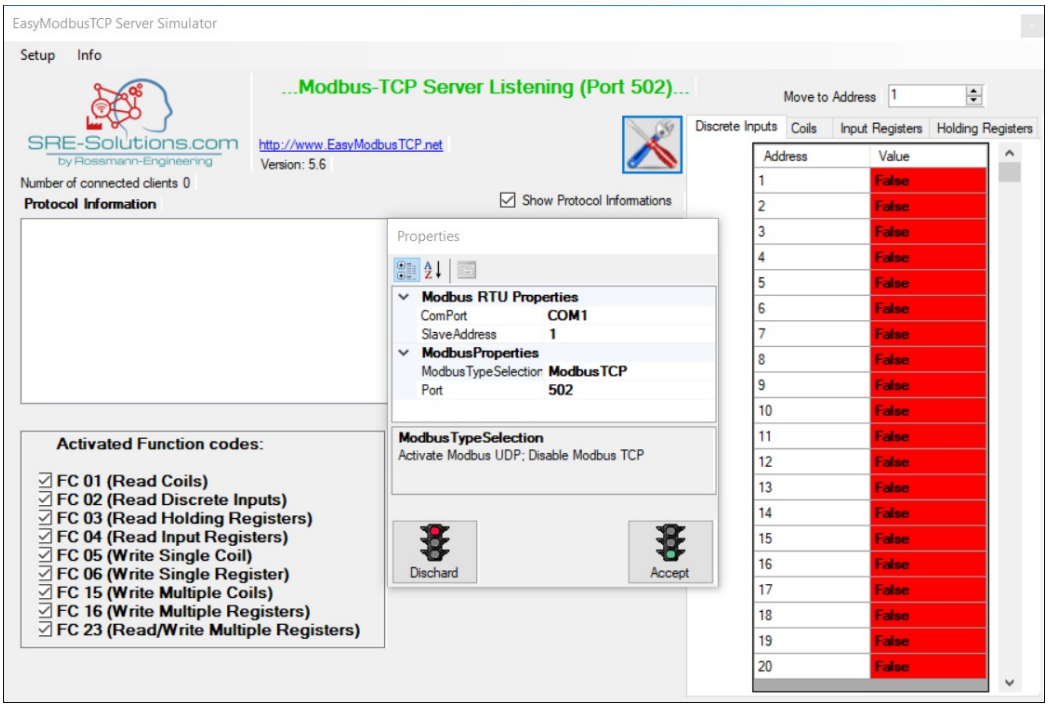
In this example, you will create two versions of the application, basic and advanced:

- the ***basic*** version of the application does not require a PLC; a Modbus TCP simulator can be used instead. This program enables exchanging basic data types between the Aurora Vision Studio client and the Modbus TCP server (PLC or a simulator);
- the ***advanced*** version of the application requires additional logic to be implemented in the PLC program. It simulates a dynamic mathematical model which represents a heater: a user can change parameters of the model parameters so the object can react slower or faster. A user can manually modify the model input parameters which represent the voltage fed to the actuator device. The PLC sends feedback about the temperature level which is then shown on the HMI;

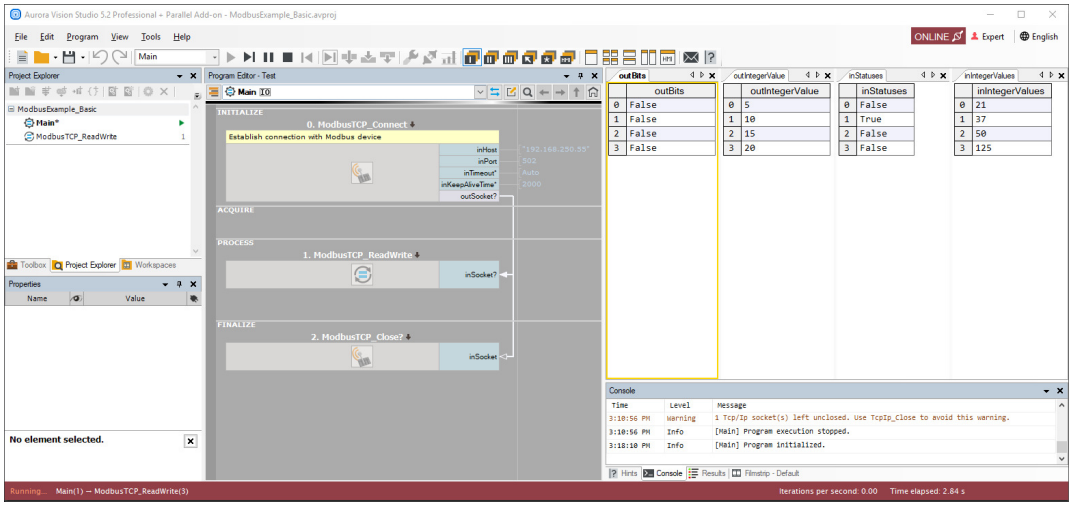
First, you need to establish a hardware connection between the PLC or the PLC simulator and the PC. The network configuration will be described later in this entry.

Basic program with Modbus TCP simulator

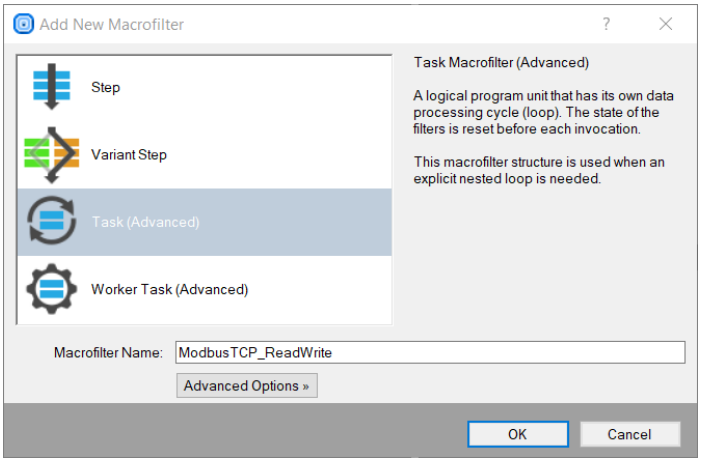
Open the EasyModbusTCP simulator and press the Setup button in the lower left corner of the application window. A *Properties* window will pop up: follow the image below to set up the configuration:



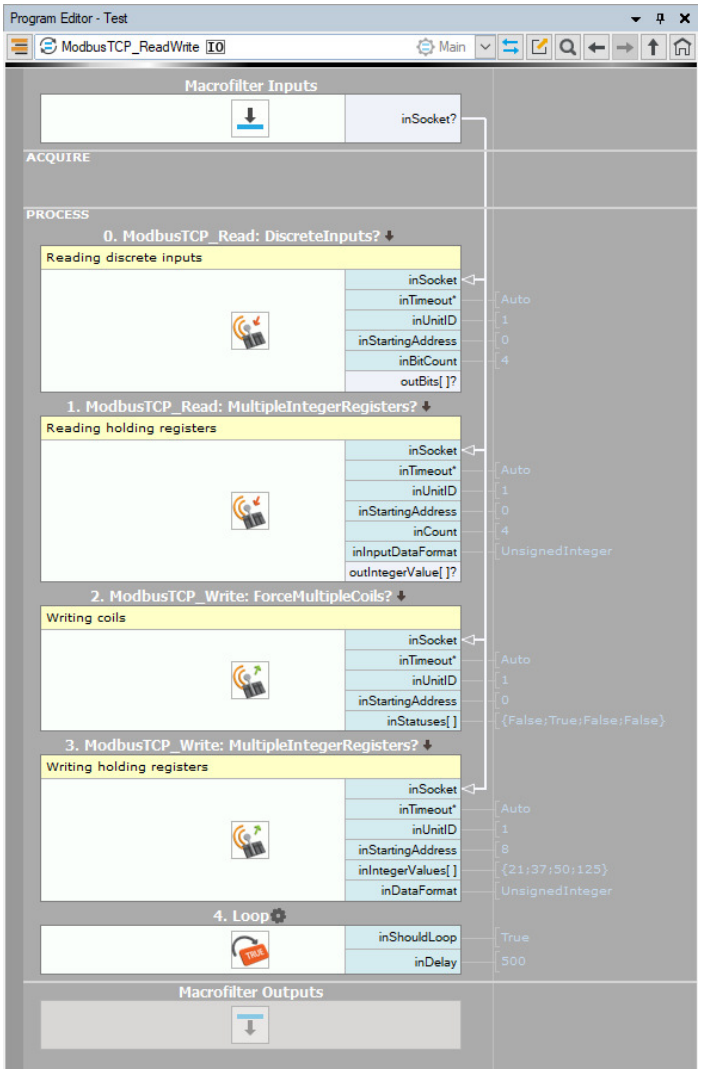
Then open a new Aurora Vision Studio project, add the ModbusTCP_Connect filter to the INITIALIZE section and the ModbusTCP_Close filter to the FINALIZE section. Aurora Vision Studio provides a set of ready-to-use filters for communication over the ModbusTCP protocol. A full list of the relevant filters is available here.



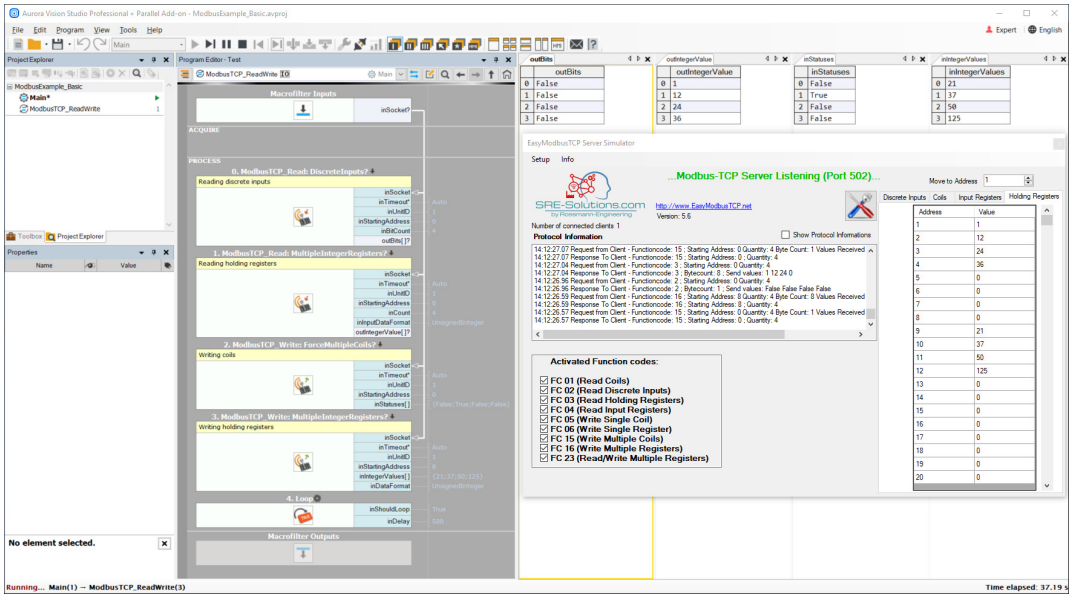
Add a Task Macrofilter to the PROCESS section.



Open the newly created Task Macrofilter and add the ModbusTCP_ReadDiscreteInputs, ModbusTCP_ReadMultipleIntegerRegisters, ModbusTCP_ForceMultipleCoils and ModbusTCP_WriteMultipleIntegerRegisters filters in the arrangement simulating that in the image below:



Below you can see working communication between the ModbusTCP client within the Aurora Vision Studio application and the simulated server.

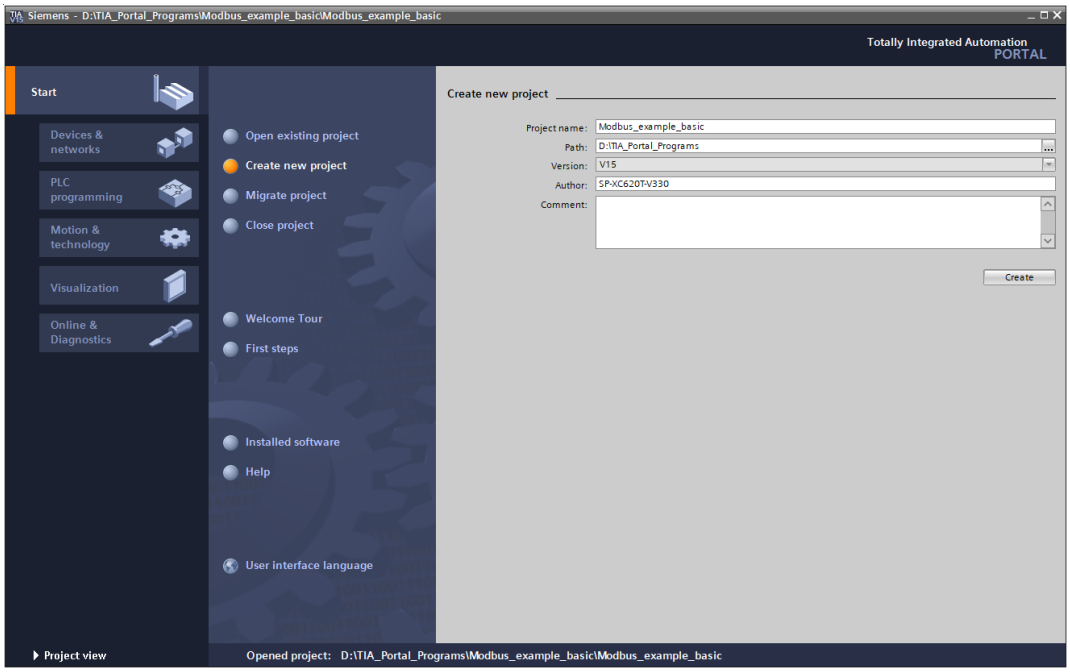


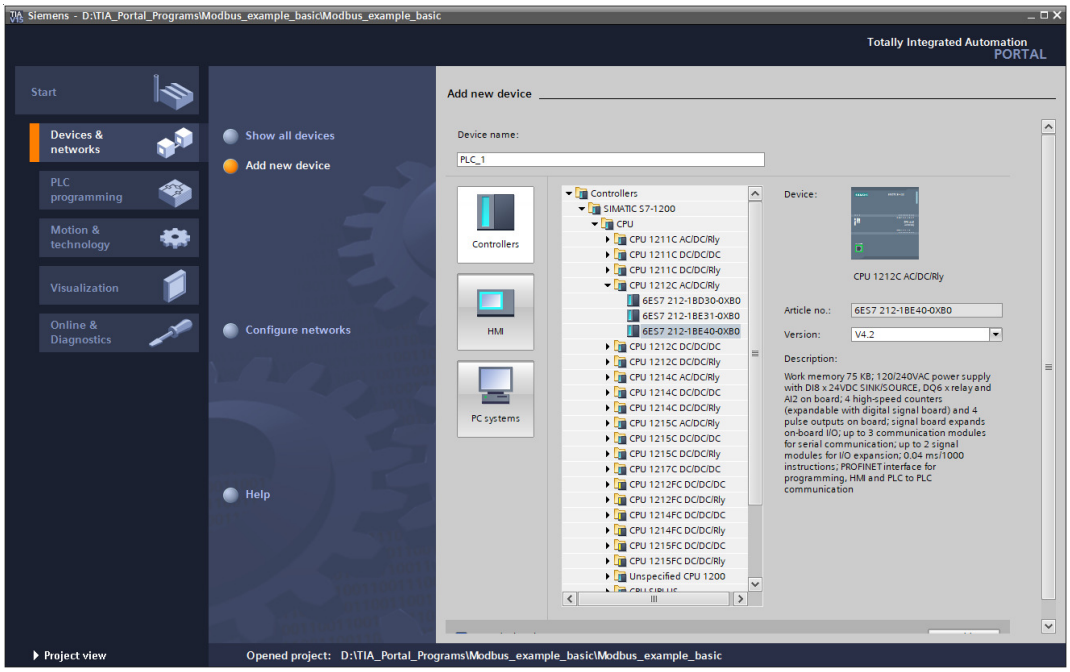
Basic program with a PLC

Network configuration

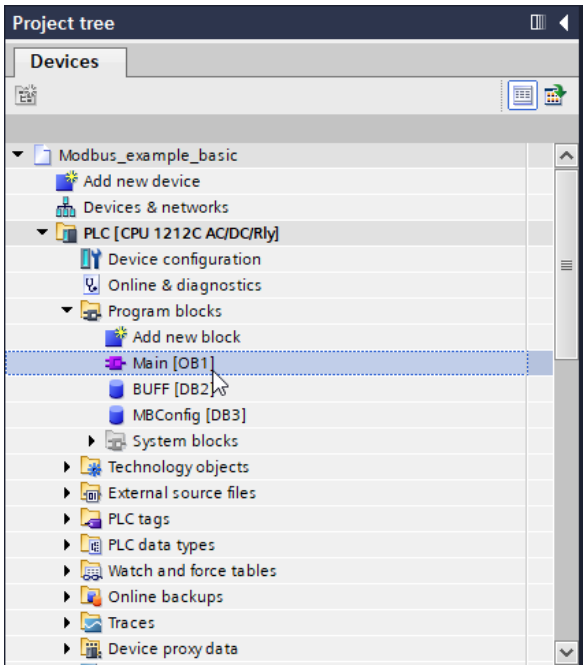
Main network configuration

Create a new project in the TIA Portal environment and add your device by double-clicking on the *Add new device* button. Select your controller model, set its CPU and other available components listed in the drop-down menu. Once you are ready, click the *OK* button.

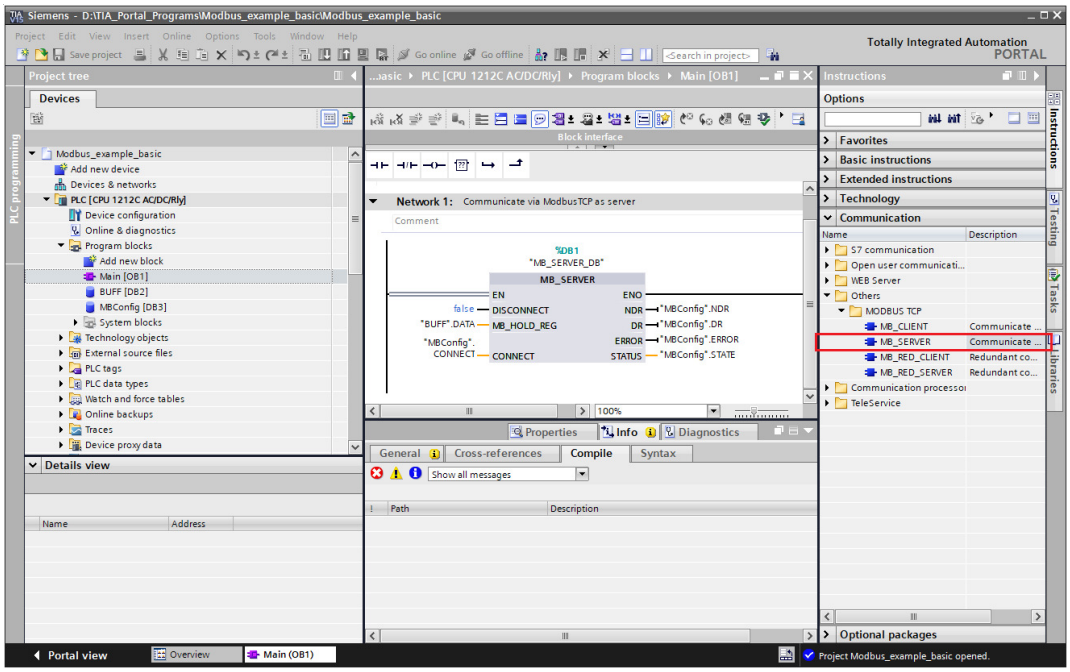




Expand the *Program blocks* list and double-click the *Main* icon. A network view will appear. Expand the Communication tab on the right-hand side (within the Instructions section) and double-click on the MB_SERVER icon available at the *Communication -> Others -> MODBUS TCP* location.

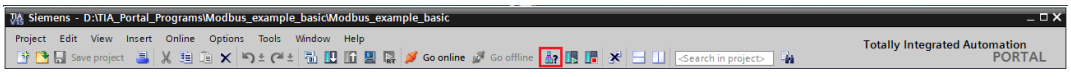


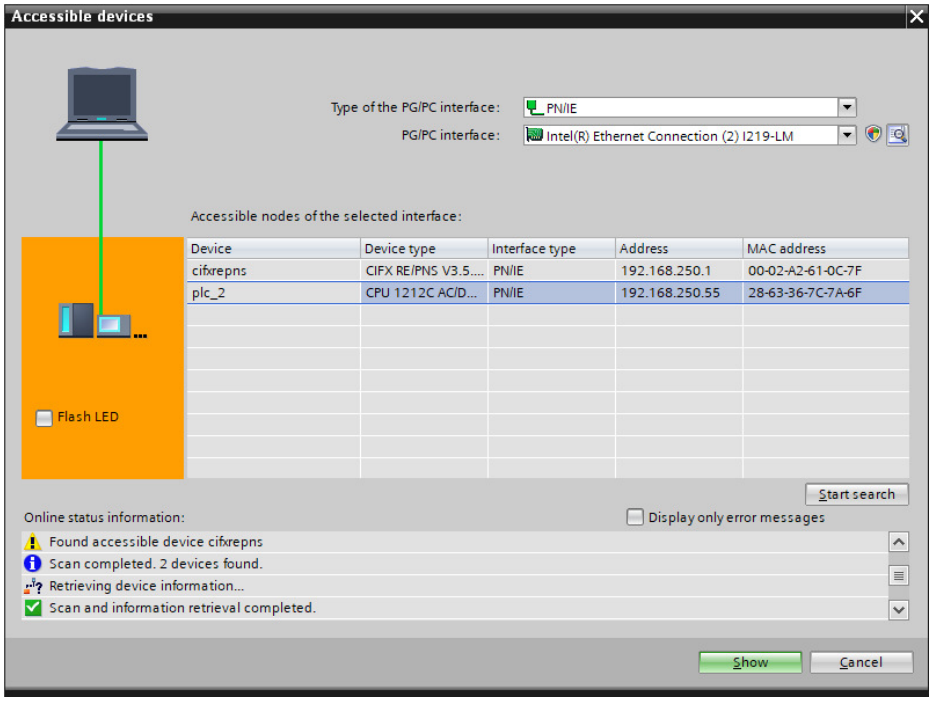
Drag the MB_SERVER block from the *Communication -> Others -> MODBUS TCP* location and drop it on to Network 1 within the application Basic.



Troubleshooting IP addressing

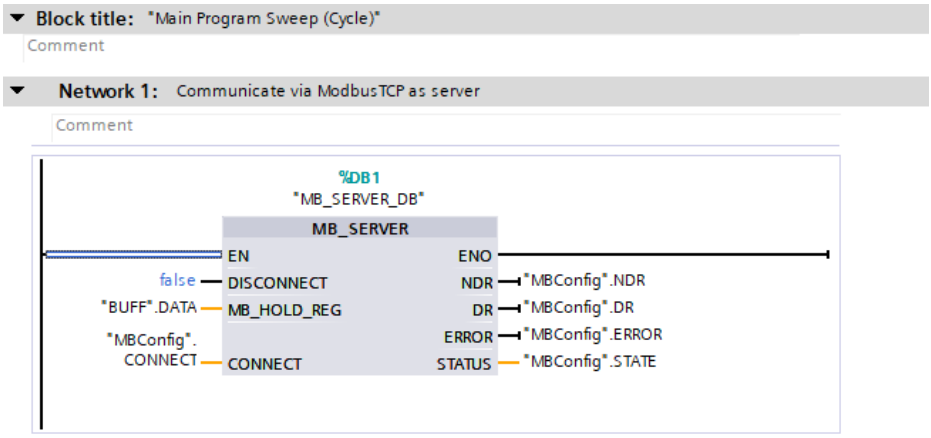
Select the *Accessible devices* icon from the toolbar to open the configuration window. Select the *PN/IE* type of the *PG/PC* interface and then the Ethernet card connected to the PLC. Click the *Start search* button, choose your PLC from the list of accessible nodes and click the *Show* button.





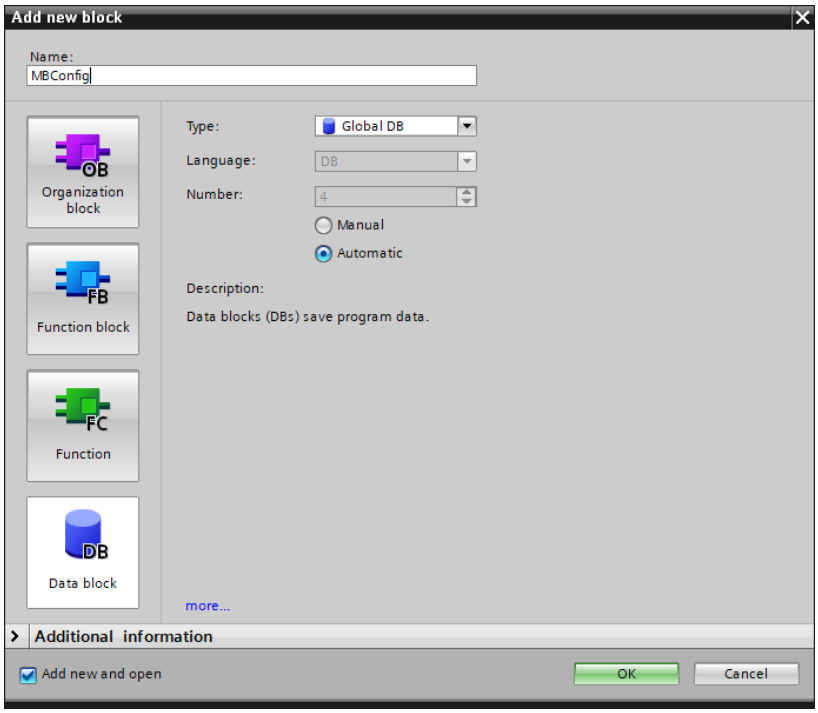
Creating a PLC program

This chapter describes how to create a PLC program to establish communication. The basic application that we start out with has only one function block; it is the same one that was created in the previous section:

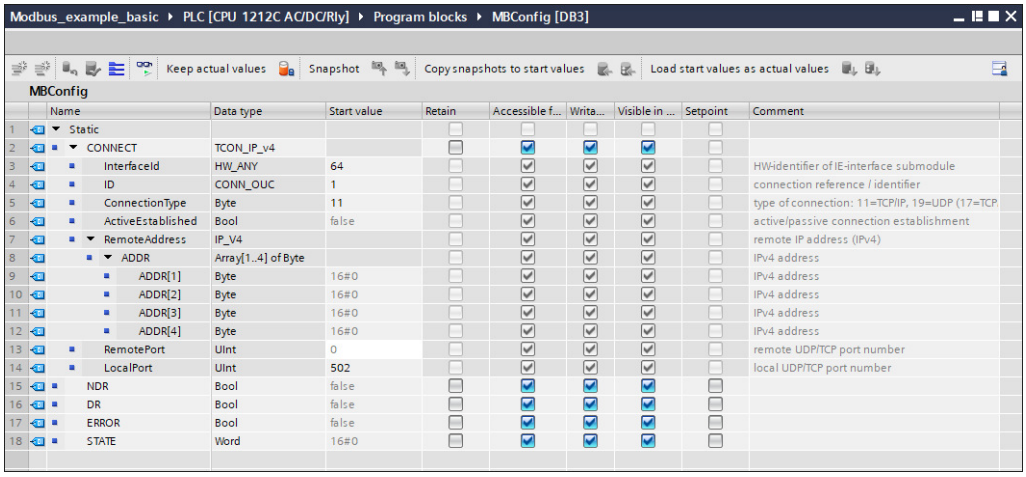


The MB_SERVER instruction is used to establish the Modbus TCP communication. In the image above, you can see some DB blocks connected to several inputs and outputs of the MB_Server block. In order to establish communication between a client and the server, you will need to set them up according to the way presented below:

- MBConfig – DB block which defines the connection type, the address of the ModbusTCP server as well as the used Port (by default 502)

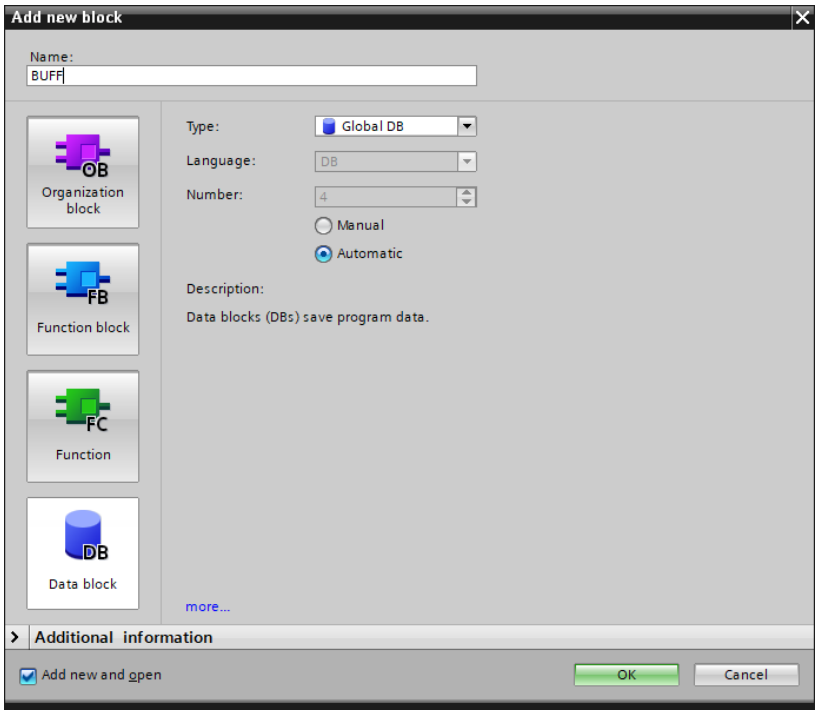


Now, define variables in the Data block. Use the same names and data types as in the image below. If you want to create a new row, just right-click on any row and select *Insert row*. Note that setting the right values of the *Start values* is essential in this step.

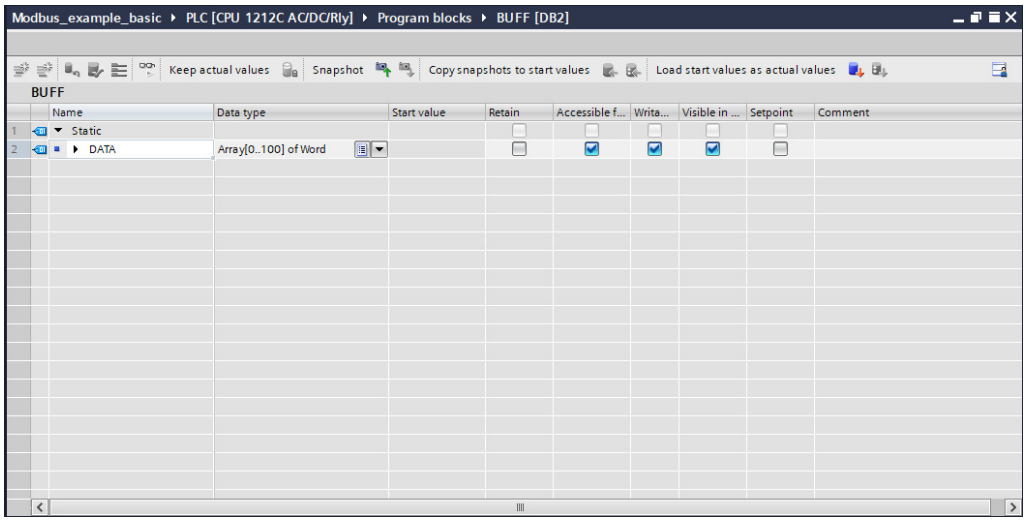


	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static								
2	CONNECT	TCON_IP_v4							
3	InterfaceId	HW_ANY	64						HW-identifier of IE-interface submodule
4	ID	CONN_OUC	1						connection reference / identifier
5	ConnectionType	Byte	11						type of connection: 11=TCP/IP, 19=UDP (17=TCP
6	ActiveEstablished	Bool	false						active/passive connection establishment
7	RemoteAddress	IP_V4							remote IP address (IPv4)
8	ADDR	Array[1..4] of Byte							IPv4 address
9	ADDR[1]	Byte	16#0						IPv4 address
10	ADDR[2]	Byte	16#0						IPv4 address
11	ADDR[3]	Byte	16#0						IPv4 address
12	ADDR[4]	Byte	16#0						IPv4 address
13	RemotePort	UInt	0						remote UDP/TCP port number
14	LocalPort	UInt	502						local UDP/TCP port number
15	NDR	Bool	false						
16	DR	Bool	false						
17	ERROR	Bool	false						
18	STATE	Word	16#0						

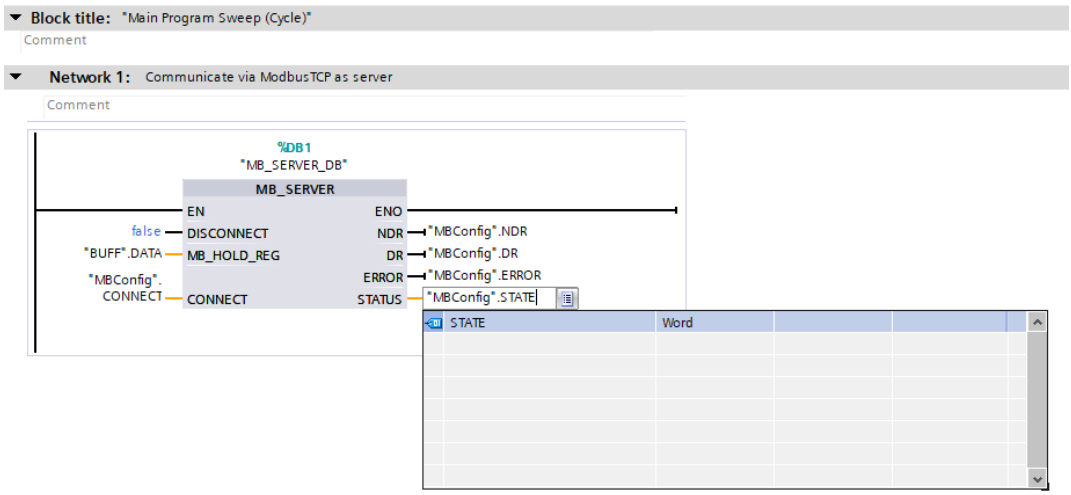
- BUFF – DB block serving as the communication buffer for the data received or sent between the PLC and the Aurora Vision Studio application



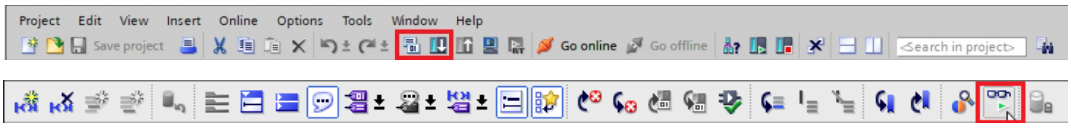
Create a word array in the BUFF data block, it should have the same size as it does in the image below:



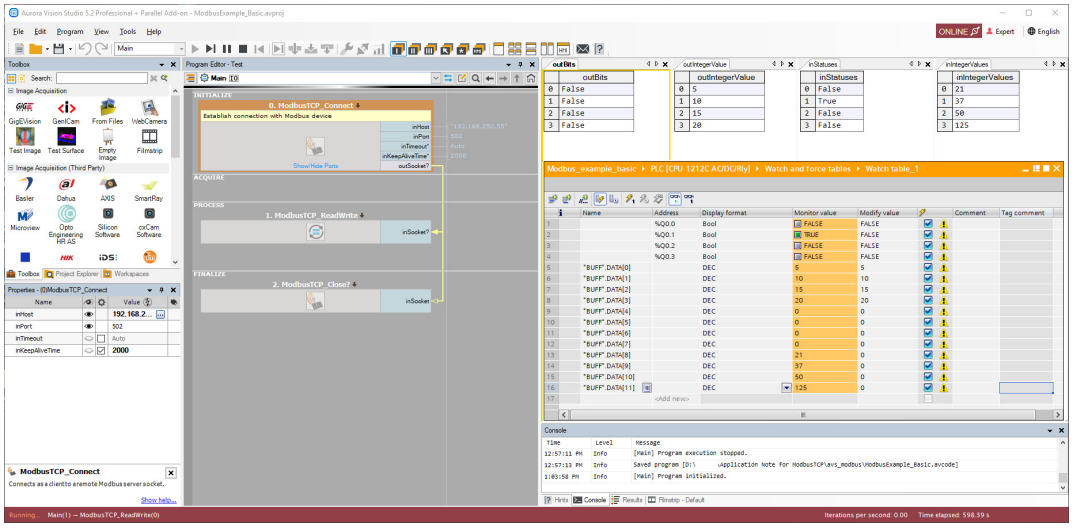
Attach the data block variables to the inputs and outputs of the MB_SERVER block in the way presented below:



If all previous steps have been performed correctly, your PLC program is ready. Compile the application and download it to the device. To see the current states of variables, turn on the Monitoring mode:



Use the application created in Aurora Vision Studio in the section where the Modbus TCP simulator was used. To test discrete inputs, coils and registers in TIA Portal, create a Watch table and a Force table, like in the image below:

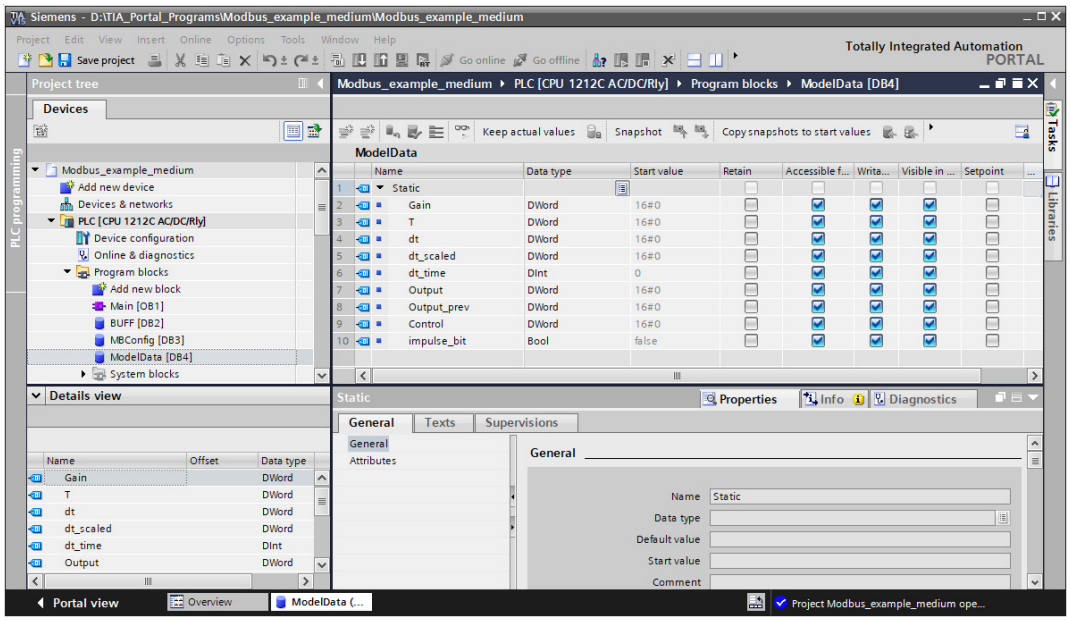


Advanced program with a PLC

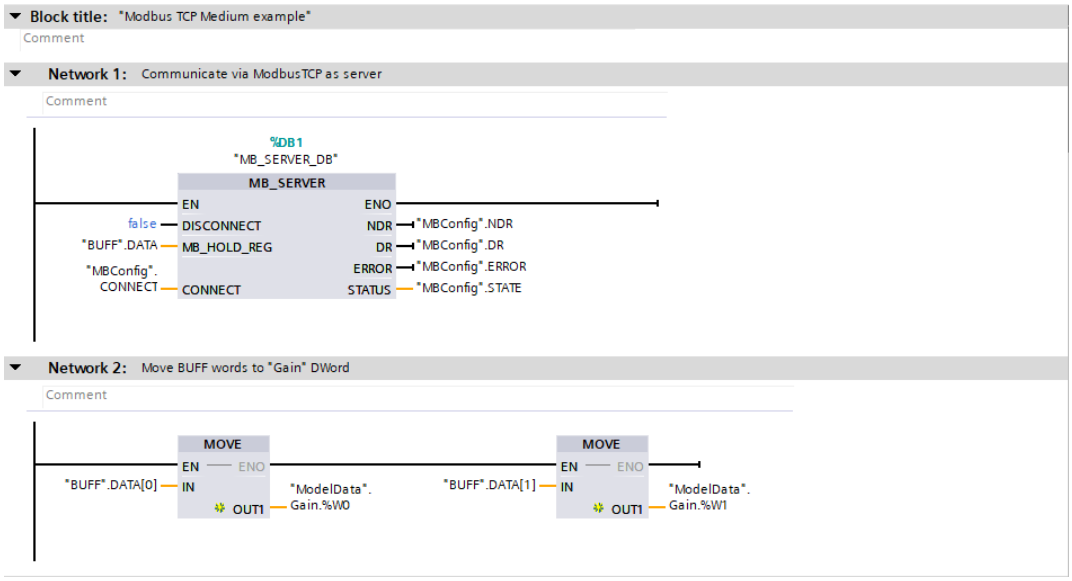
Creating a PLC program

This chapter describes how to modify a basic PLC program to meet requirements of advanced usage as mentioned at the beginning of this article. If you have created a Basic PLC program described in the previous chapters, save this project as a copy with a new name. Otherwise, create BUFF and MBConfig data blocks as described in this section.

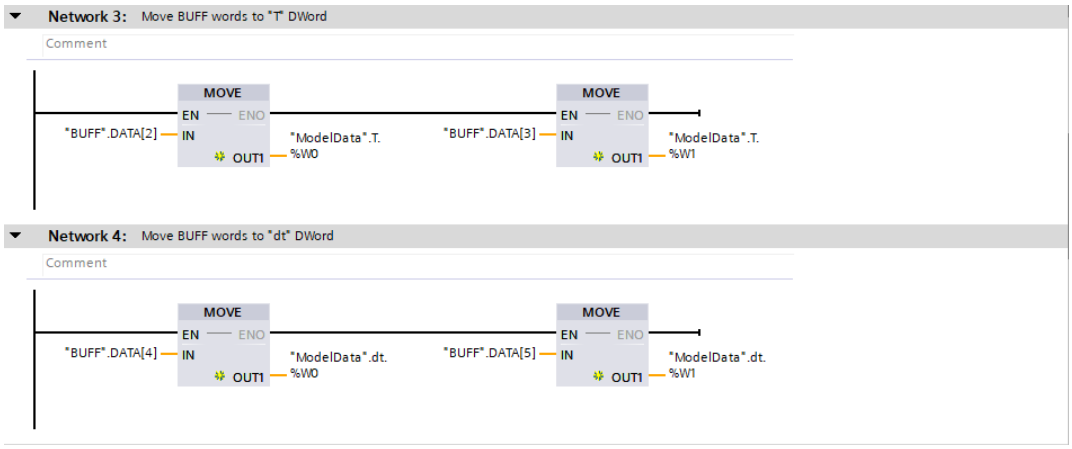
Create a new Data block called ModelData with the variables set like in the image below:



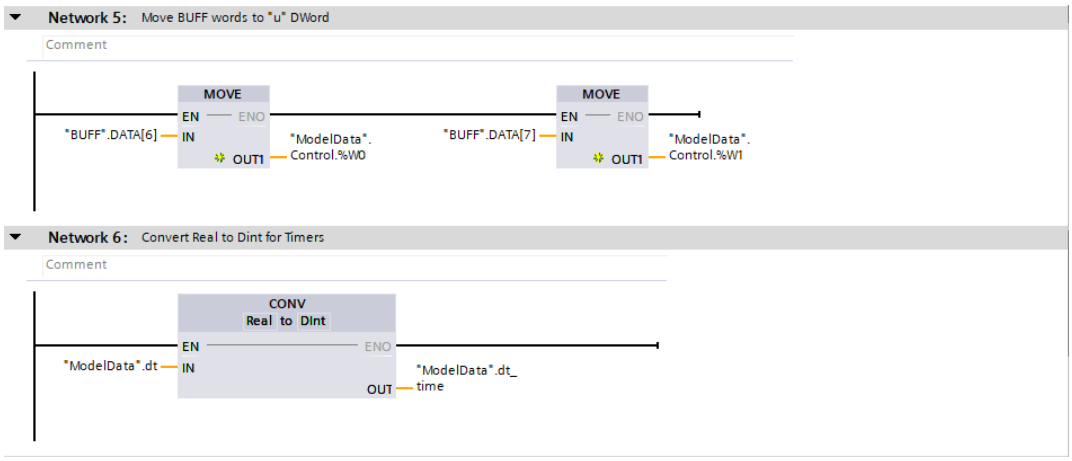
Nothing has changed for Network 1 of the Advanced program; MB_SERVER block is used to establish the Modbus TCP connection. For Network 2 use MOVE blocks to move appropriate BUFF data words to the ModelData.Gain variable.



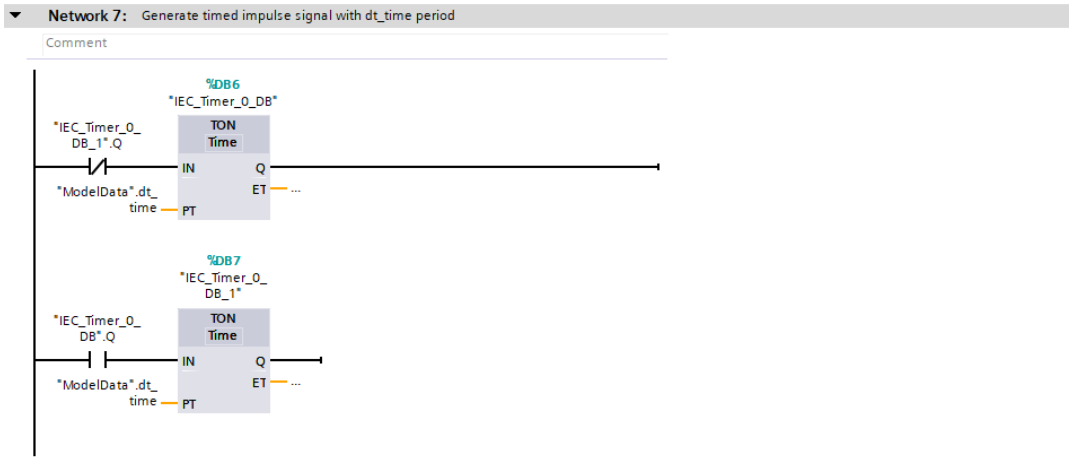
For Networks 3 and 4, use MOVE blocks to move BUFF words to ModelData.T and ModelData.dt.



For Network 5, use MOVE blocks to move BUFF words to ModelData.Control. For Network 6, use a CONV block to convert Real values to Dint values for further use in timers blocks.

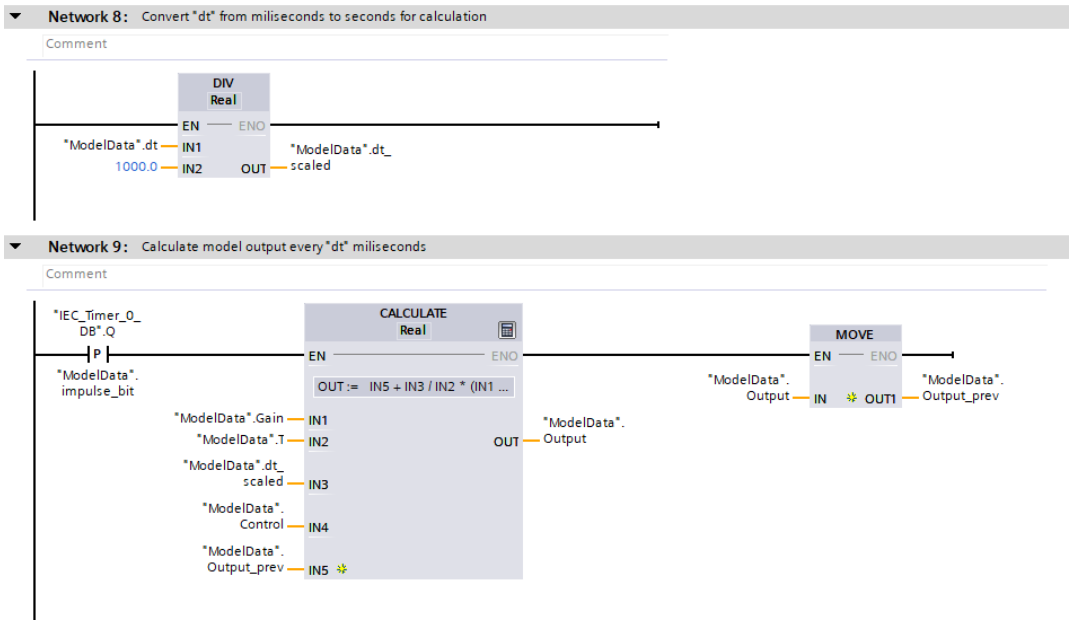


Network 7 uses TON Timer blocks to generate a timed impulse signal for further calculations. Create a new branch in this network, put TON blocks in the network and create Data blocks appropriate for them.

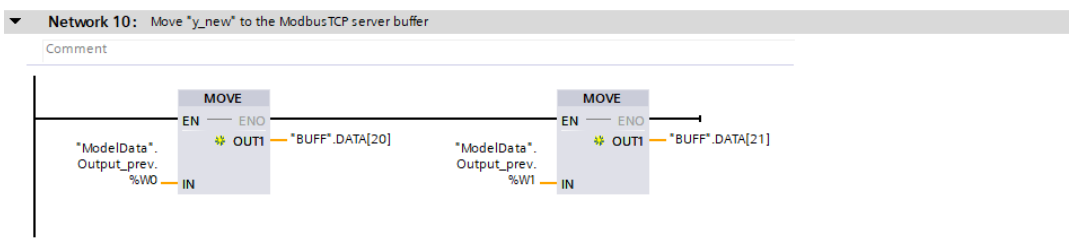


For Network 8, use a DIV block to scale ModelData.dt from milliseconds to seconds for further calculations. For Network 9, use the logic presented in the image below. The formula for the CALCULATE block should be as follows:

```
OUT:= IN5 + IN3 / IN2 * (IN1 * IN4 - IN5)
```

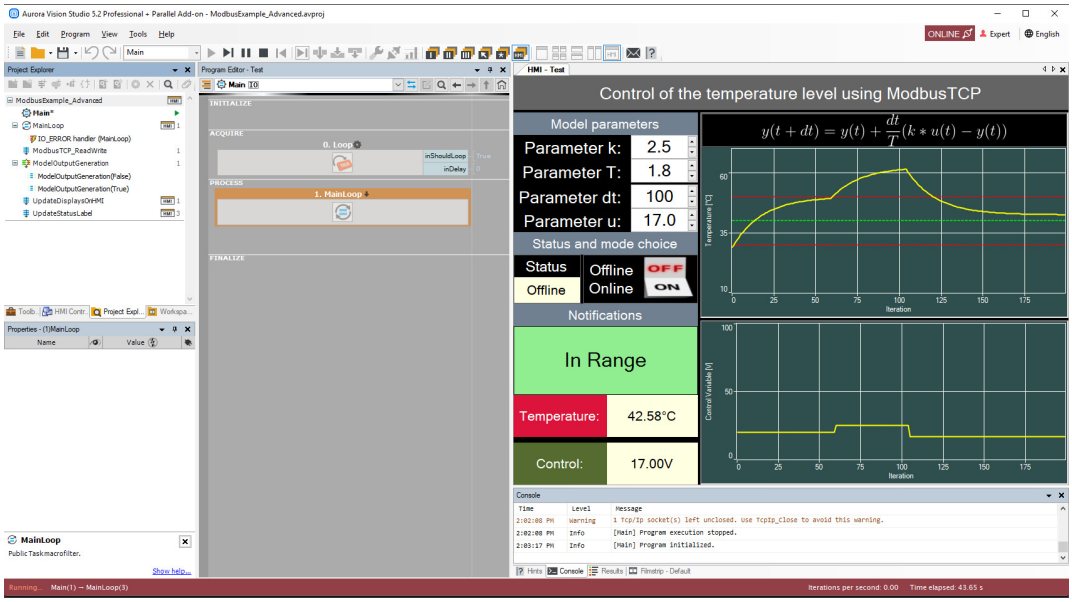


For Network 10, use MOVE blocks to move ModelData.Output_prev words to BUFF. This variable will be read in Aurora Vision Studio application.

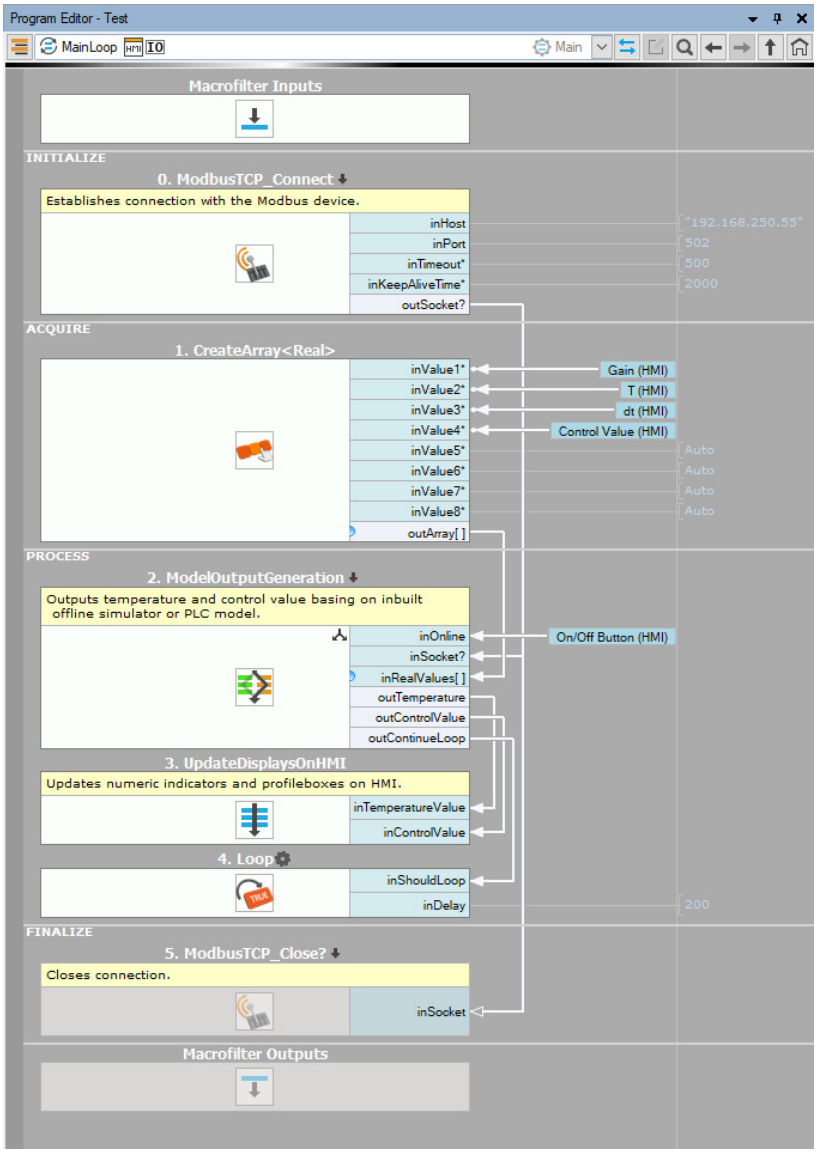


Creating Aurora Vision Studio program

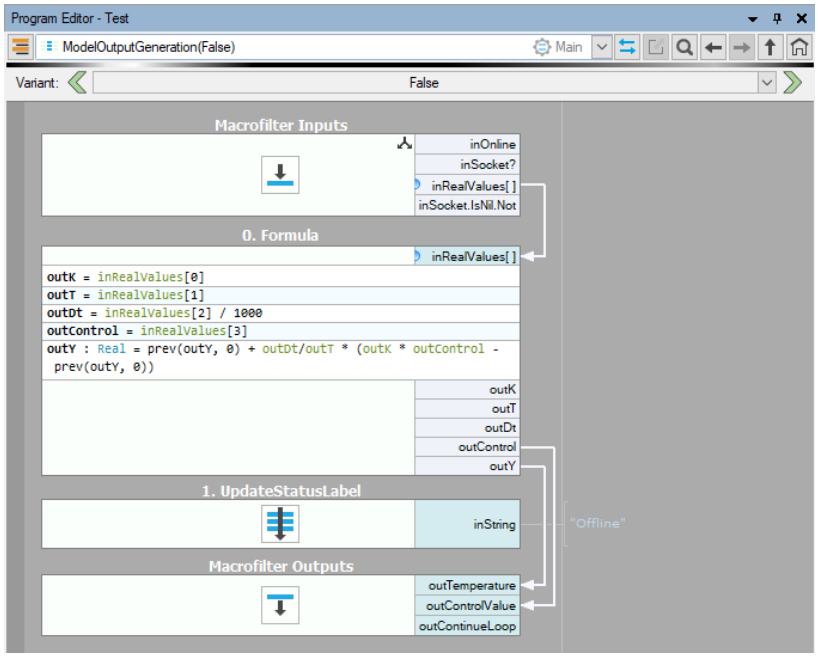
This Application Note does not explain how to create an HMI for an advanced program example. Take a look at the official Aurora Vision Studio example ModbusExample_Advanced.



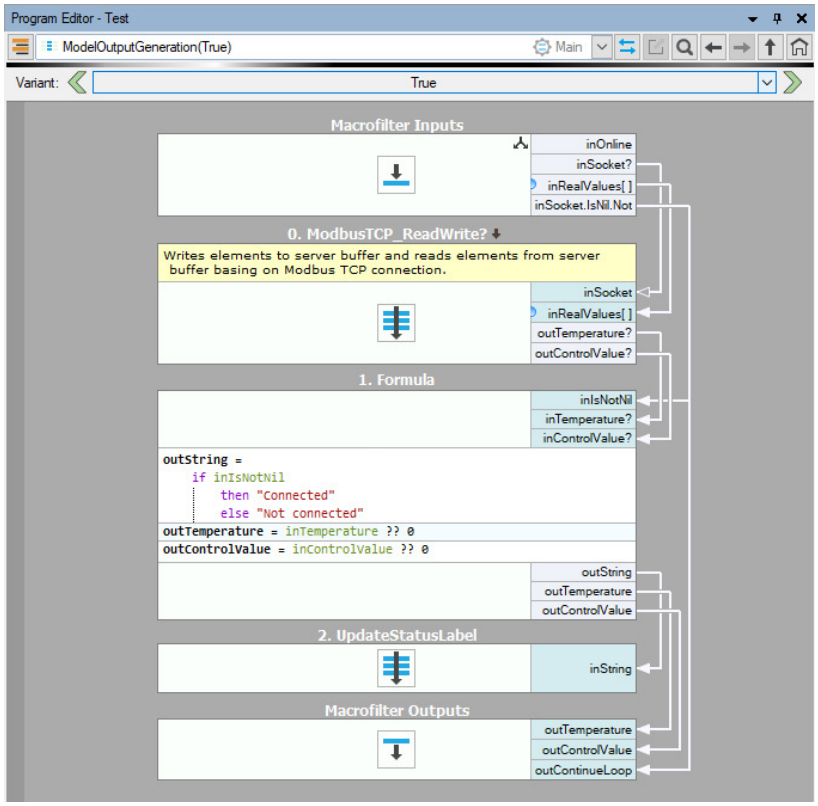
The *MainLoop* macrofilter body which is iterated every 200 ms:



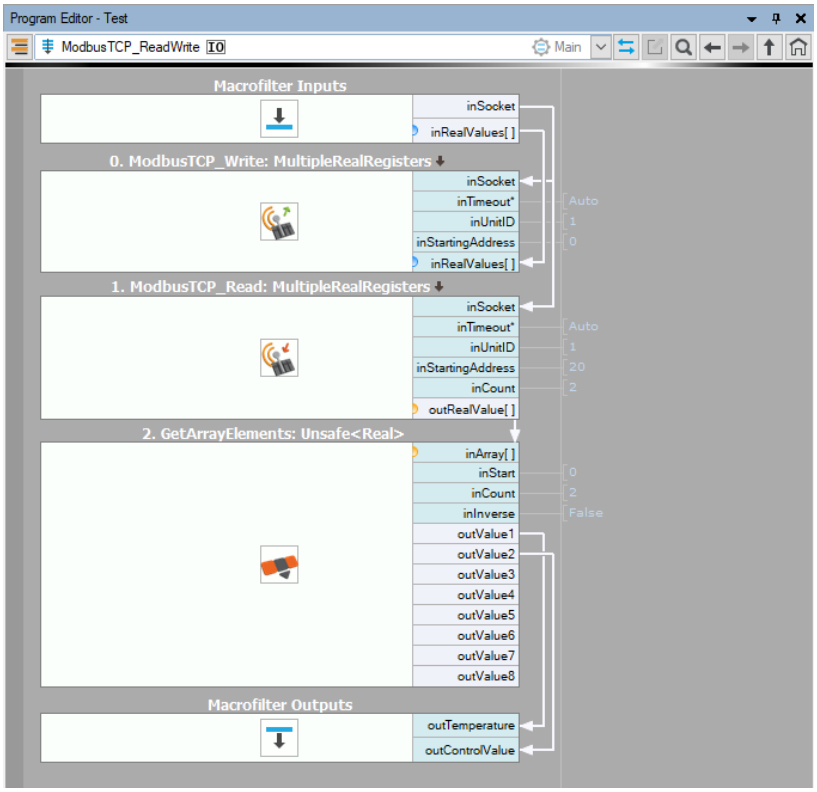
The False branch of the *ModelOutputGeneration* variant macrofilter:



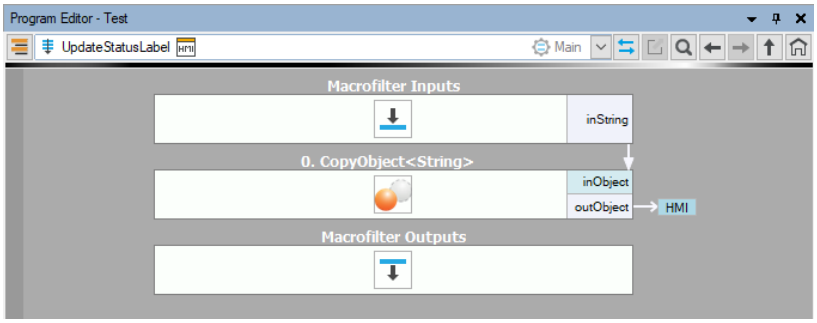
The True branch of the *ModelOutputGeneration* variant macrofilter:



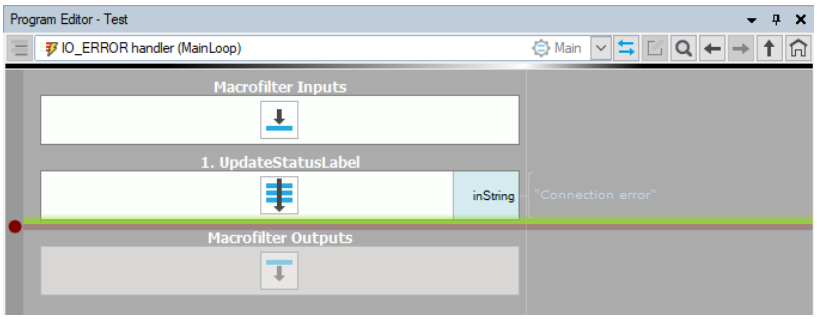
The *ModbusTCP_ReadWrite* macrofilter which is invoked in the online mode:



The *UpdateStatusLabel* macrofilter which is invoked in a few parts of the program; for instance, to notify the user about successful execution or an upcoming error:



An IO_ERROR error handler which is executed when connection in the online mode is interrupted:



Interfacing Zebra 4Sight computer (Modbus TCP – Client) to Aurora Vision Studio

Purpose and equipment

This document explains how to configure Modbus TCP PLC with Aurora Vision Studio using Zebra 4Sight computer. In this setup, the Zebra 4Sight computer running Aurora Vision Studio acts as a Modbus Client (Master), while the PLC acts as a Modbus Server (Slave).

Required equipment:

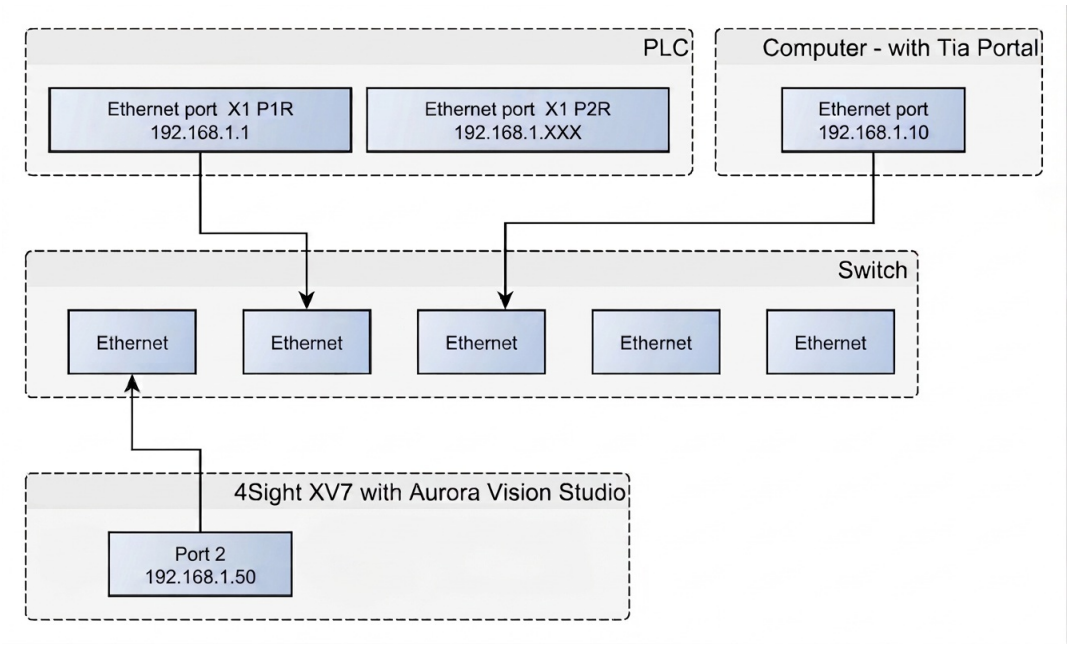
- 4Sight computer (used 4Sight XV7);
- Siemens S7-1200 PLC;
- Ethernet switch with at least 3 In/Outs might be useful – simplifies configuration;
- PC with TIA Portal V19 (for PLC configuration);
- Aurora Vision Studio 5.7 or later (version 5.7 used).

Application notes demonstrates configuration steps for specific 4sight model, but it is not obligatory to use this one. If you have different model, please refer to manuals provided by its manufacturer.

Hardware connection

In application notes there is configured one network which uses Ethernet switch. Devices connected to this network have addresses 192.168.1.XXX, where XXX stands for number 1-254 unique for every device/port connected.

In Zebra 4Sight manual it is mentioned that PLC Software should not be installed on Z4Sight PC. Taking this into account, we have one PLC and two computers in our network:

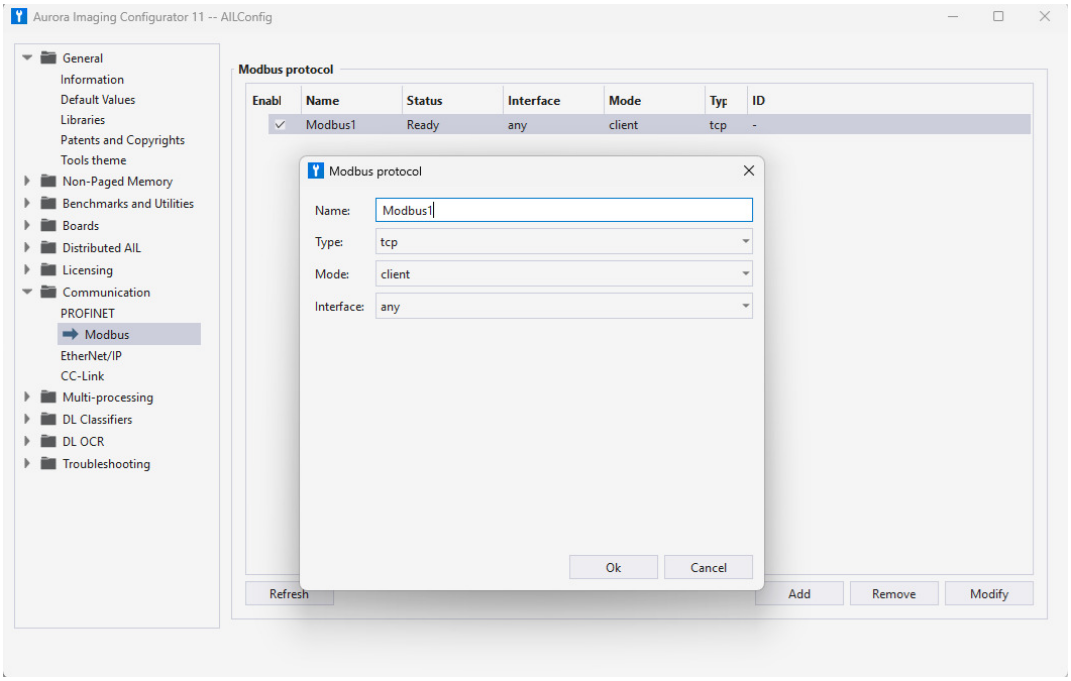


Before proceeding to the further point, make sure that all devices are powered up.

Configuring AIL on Zebra 4Sight

Before creating the application in Aurora Vision Studio, the Modbus Client service must be enabled in the AIL configuration.

1. Open the AILConfig application on the Z4Sight computer.
2. In the left pane, navigate to *Communication -> Modbus*.
3. Add a new instance. Set parameters:
    - Name: **Modbus1**
    - Type: **tcp**
    - Mode: **client**
    - Interface: **any**
4. Save the configuration

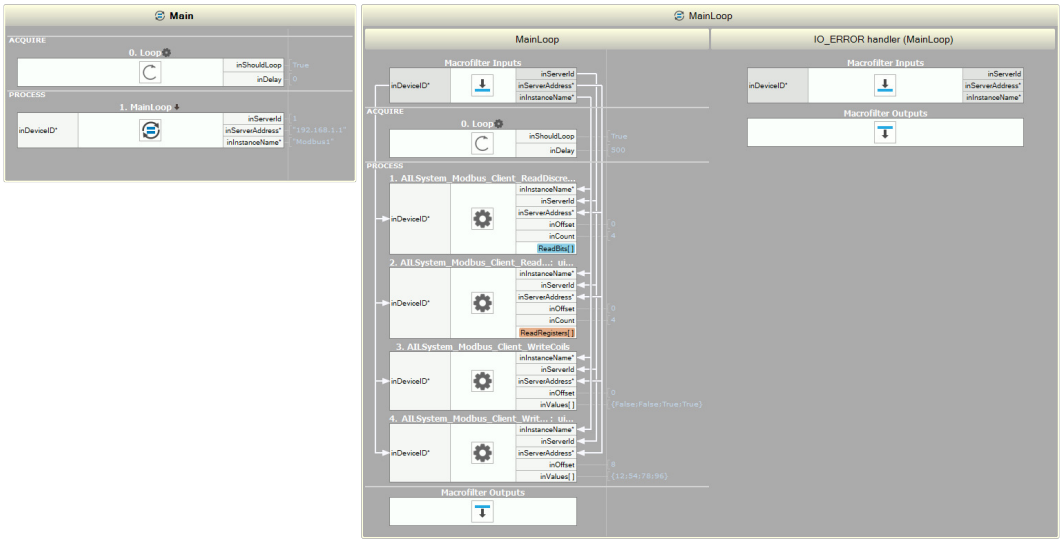


Aurora Vision Studio application

The application utilizes dedicated Aurora Imaging Library Modbus filters to read and write holding registers and coils.

1. Inside the Main macrofilter, go to the **ACQUIRE** section and add a Loop filter to keep the task running continuously.
2. In the **PROCESS** section, create a new *Task Macrofilter* named **MainLoop**. Add four macrofilter inputs:
   - **inDeviceID** of type *AILSystemAndDeviceIdentifier**
   - **inInstanceName** of type *String**
   - **inSeverID** of type *Integer**
   - **inServerAddress** of type *String**

   Set the inInstanceName value to match the AILConfig setup (*Modbus1*) and the inServerAddress to the address of the PLC (*192.168.1.1*).
3. Open the MainLoop macrofilter. In the **ACQUIRE** section, add the Loop filter. Set its inDelay input to the desired polling rate (e.g., 500 ms) to regulate the communication frequency.
4. In the **PROCESS** section, add the AILSystem_Modbus_Client_ReadDiscreteInputs filter. Link the inDeviceID, inInstanceName, inSeverID and inServerAddress from the macrofilter inputs. Set **inOffset** to *0* and the **inCount** to *4*. Present the **outValues** in the Preview window.
5. Add the AILSystem_Modbus_Client_ReadHoldingRegisters_uint16 filter. Link the inDeviceID, inInstanceName, inSeverID and inServerAddress from the macrofilter inputs. Set **inOffset** to *0* and **inCount** to *4*. Present the **outValues** in the Preview window.
6. Add the AILSystem_Modbus_Client_WriteCoils filter. Link the inDeviceID, inInstanceName, inSeverID and inServerAddress from the macrofilter inputs. Set **inOffset** to *0* and the **inValues** to *{False;False;True;True}*.
7. Add the AILSystem_Modbus_Client_WriteHoldingRegisters_uint16 filter. Link the inDeviceID, inInstanceName, inSeverID and inServerAddress from the macrofilter inputs. Set **inOffset** to *8* and the **inValues** to *{12;54;78;96}*.
8. In the Project Explorer, right-click on the **MainLoop** macrofilter block, select **Add New Error Handler**, and choose **IO_ERROR**.



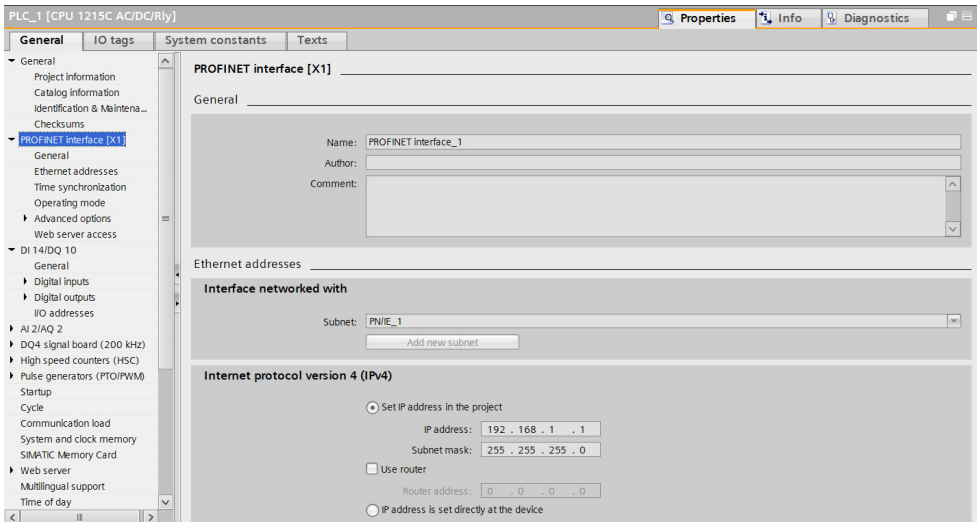
Configuring PLC in TIA Portal

Note: Aurora Vision Studio (and the Modbus protocol standard) transmits data in Big-Endian format. Siemens S7-1200 PLCs store data in Little-Endian format. A byte-swap conversion will be required on the PLC side to interpret the values correctly.

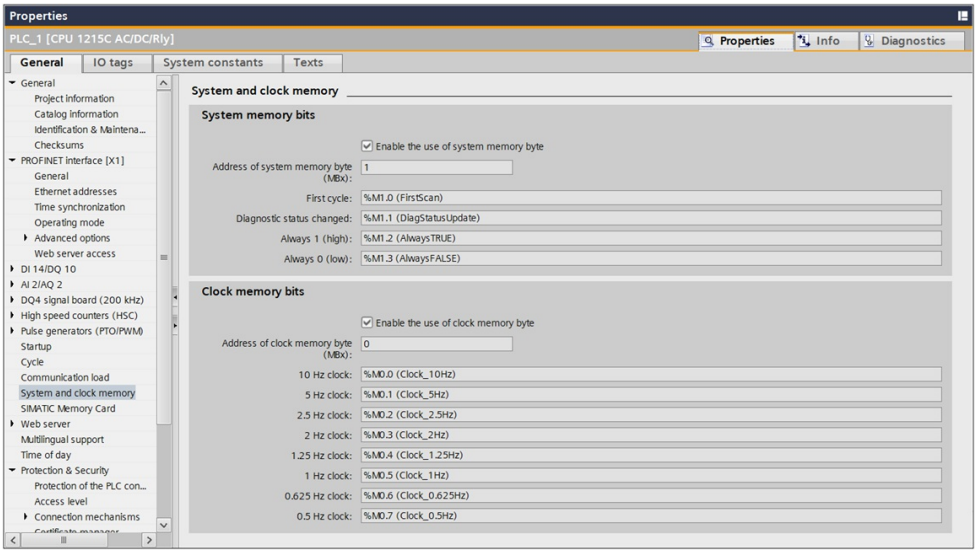
1. **Device Configuration**

   After adding the S7-1200 PLC to the project, apply the following settings in the **Properties** tab:

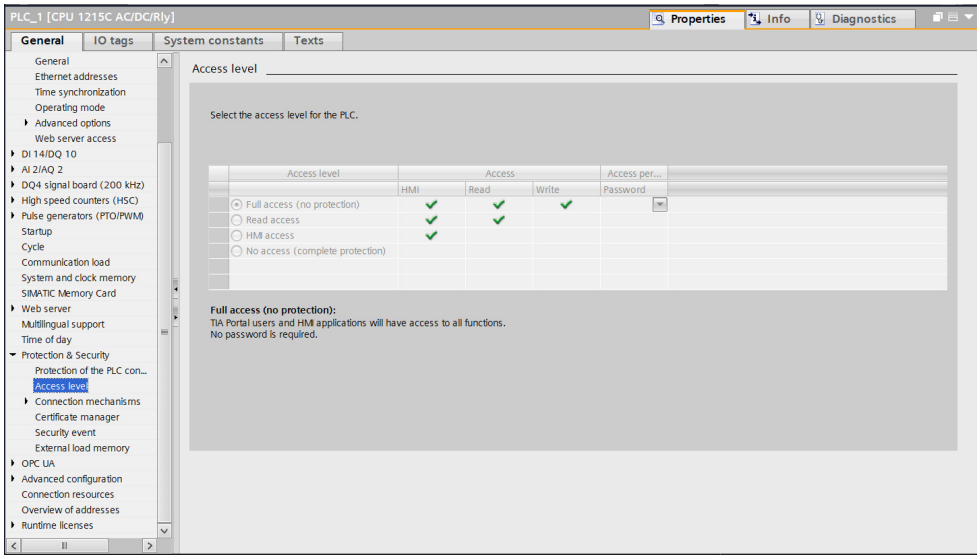
   - *PROFINET interface [X1]*: Set the appropriate IP address and add/setup the Subnet. Ensure it is in the same subnet as the Z4Sight computer.



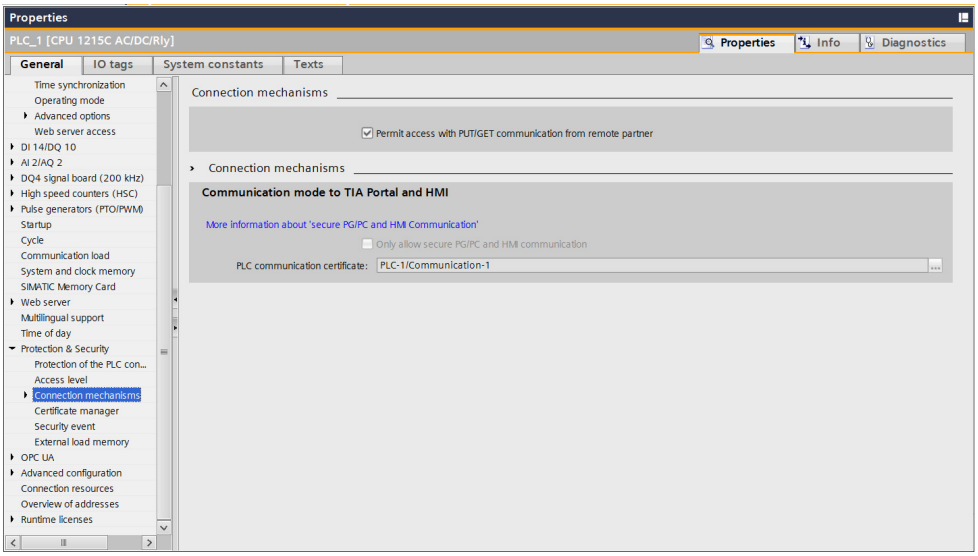
   - *System and clock memory*: Check **Enable the use of system memory byte** and **Enable the use of clock memory byte**.



- *Access level*: Set to **Full access** (no password).



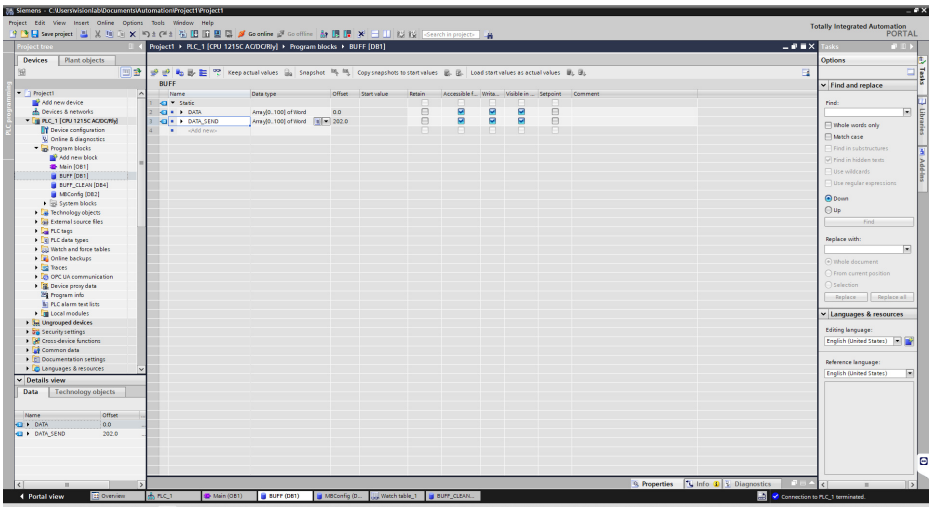
- *Connection mechanisms*: Check **Permit access with PUT/GET communication from remote partner** and uncheck **Only allow secure PG/PC and HMI communication**.



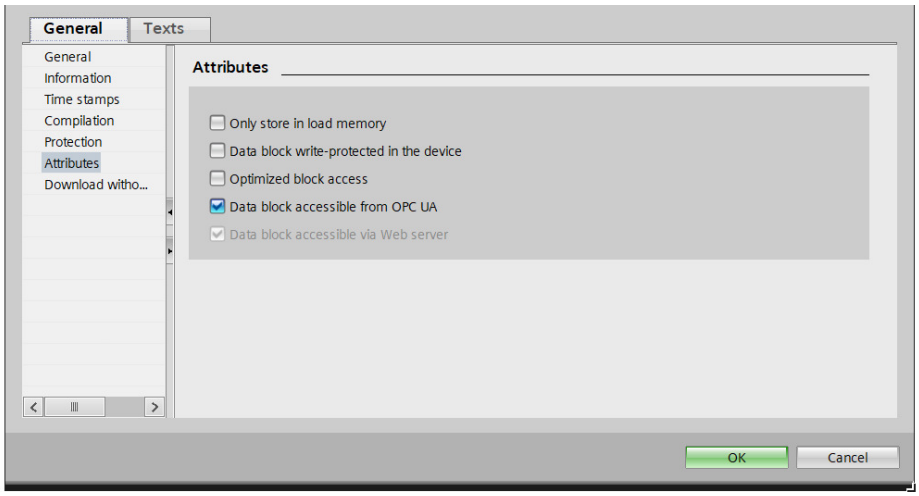
2. **Creating Data Blocks**

   Two Data Blocks are required to manage the data arrays and the connection parameters:

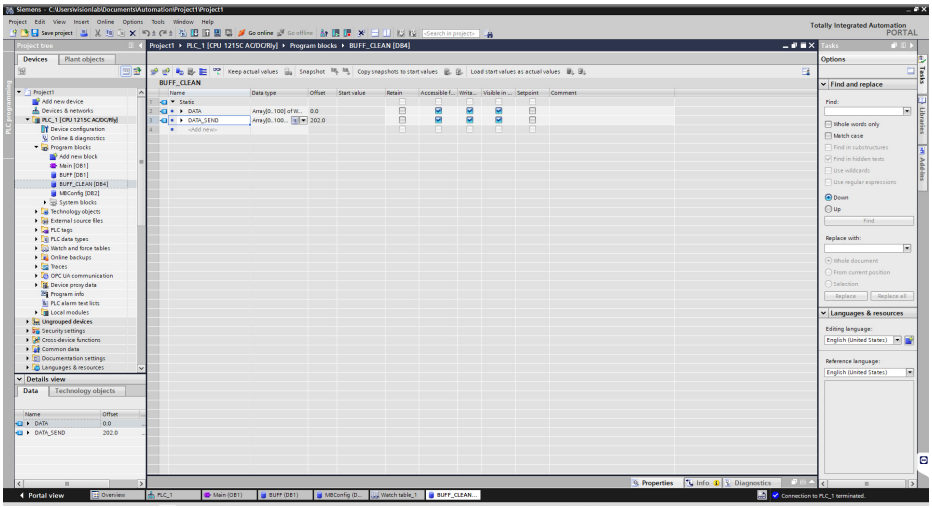
- **BUFF [DB1]** – it holds the raw Modbus arrays and the processed (byte-swapped) arrays.
  1. Define the following variables:
     - **DATA** of type *Array[1..100] of Word* (Raw data received from AVS)
     - **DATA_SEND** of type *Array[1..100] of Word* (Raw data to be sent to AVS)



  2. Right-click the block, go to *Properties -> Attributes*, and uncheck the **Optimize block access** option. This is mandatory to expose absolute addressing.

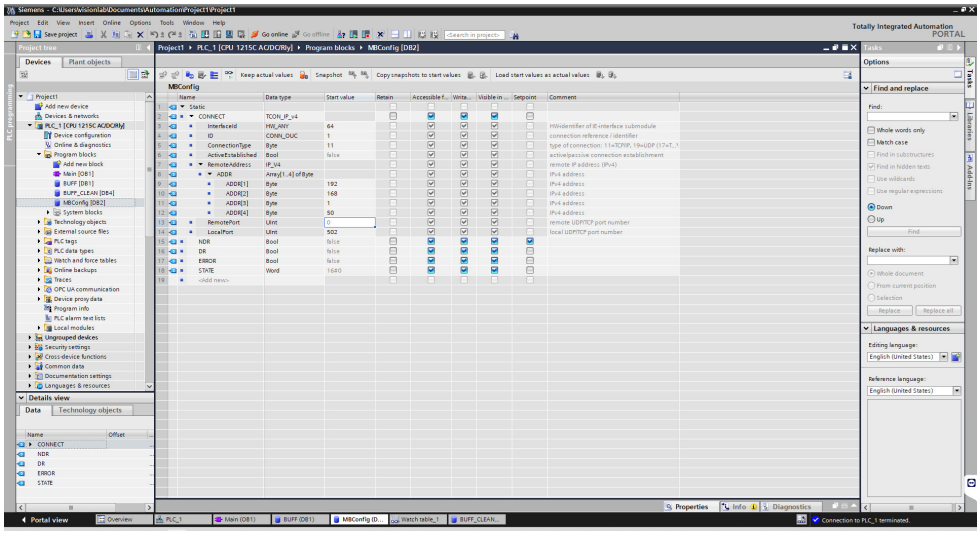


- **BUFF_CLEAN [DB4]** – it holds the processed (byte-swapped) arrays.
  1. Define the following variables:
     - **DATA** of type *Array[1..100] of Word* (Raw data received from AVS)
     - **DATA_SEND** of type *Array[1..100] of Word* (Raw data to be sent to AVS)



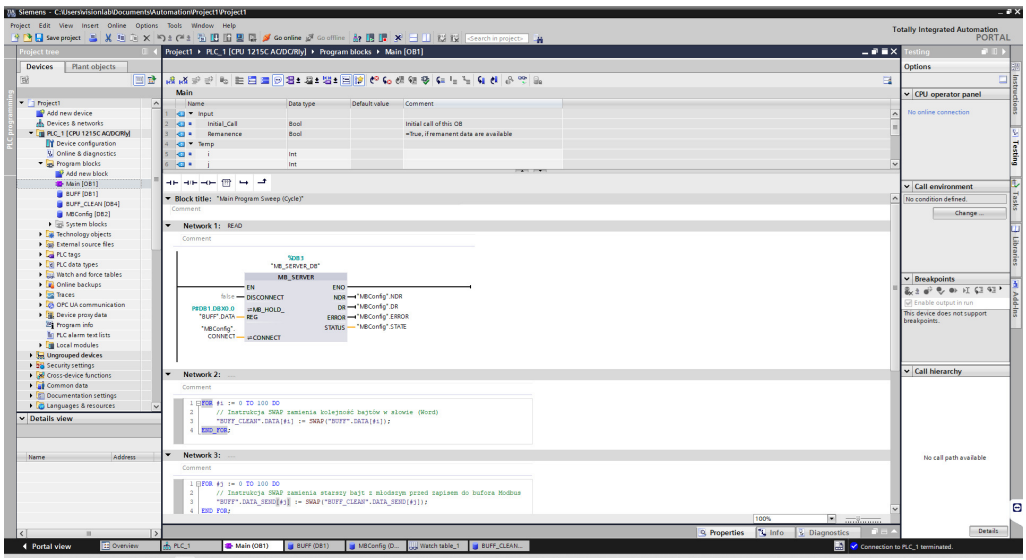
  2. Right-click the block, go to *Properties -> Attributes*, and uncheck the **Optimize block access** option. This is mandatory to expose absolute addressing.

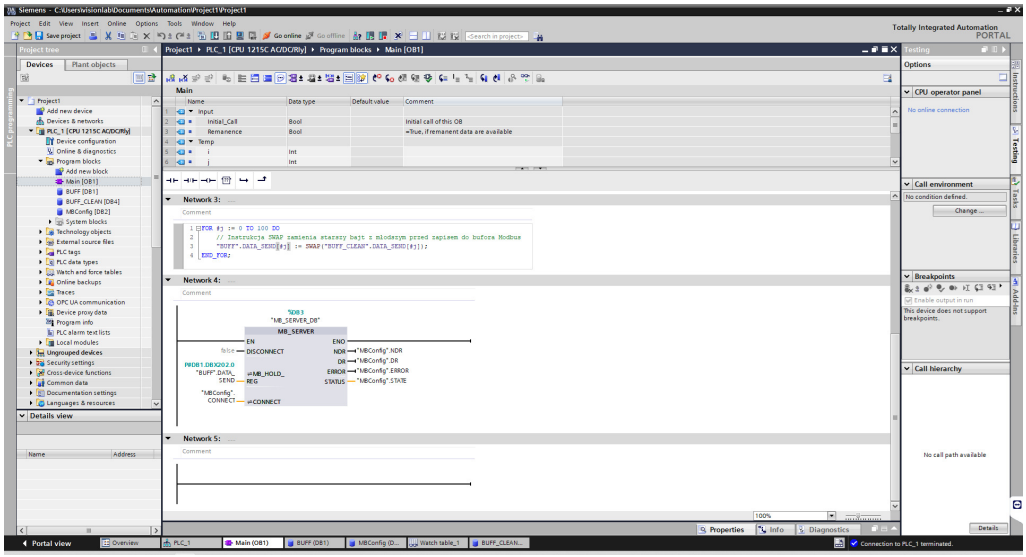
- **MBConfig [DB2]** – it stores the connection parameters and status flags for both the read and write operations. Set their Start values as follows:
  1. **CONNECT** of type *TCON_IP_v4*:
     - *InterfaceId*: **64**
     - *ConnectionType*: **11**
     - *RemoteAddress*: **192.168.1.50** (IP address of the Z4Sight computer)
     - *LocalPort*: **502**
     - *ID*: **1**.
  2. Add *Bool* flags: **NDR**, **DR**, **ERROR**.
  3. Add **STATE** of type *Word*.



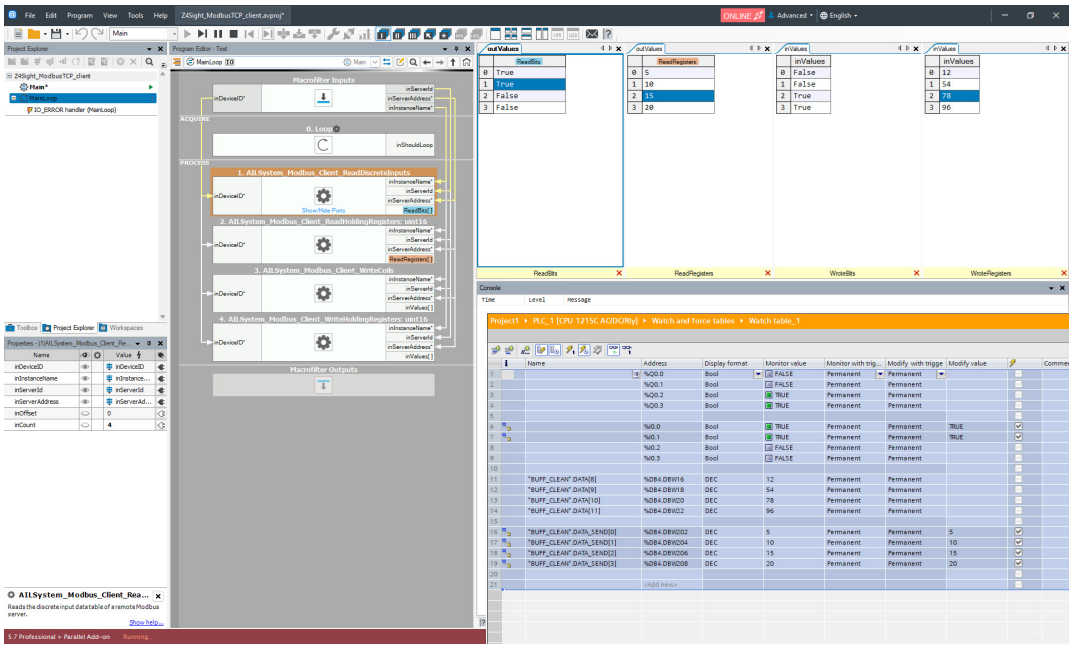
3. **Main Logic [OB1]**

Create structure as showed in images below:





When monitoring values in the TIA Portal **Watch Table**, always refer to the **BUFF_CLEAN** arrays to see the correct decimal values.



Using OPC UA Communication

Purpose and equipment

This document describes how to make a practical example of establishing OPC UA communication between the Aurora Vision Studio and an OPC UA server. For more information about possible exchangeable arguments please refer to the Communicating over OPC UA technical issue. List of available filters can be found here: OPC UA filters.

Required equipment:

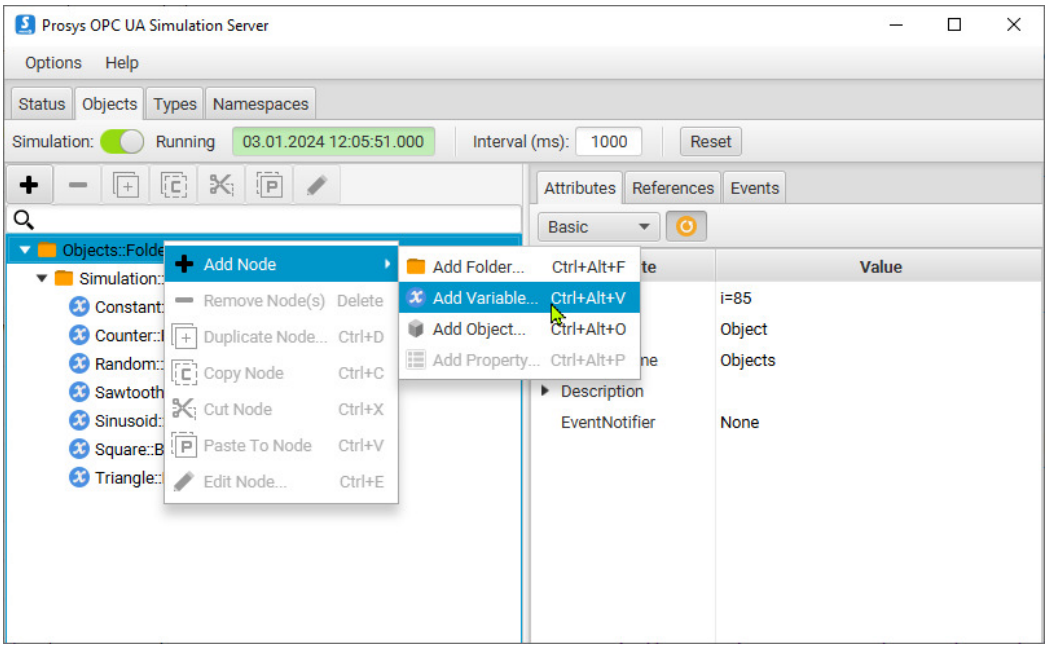
- OPC UA server simulator, e.g. OPC UA Simulation Server
- Aurora Vision Studio 5.4 Professional or later (version 5.4 was used)

Basic program with OPC UA server simulator

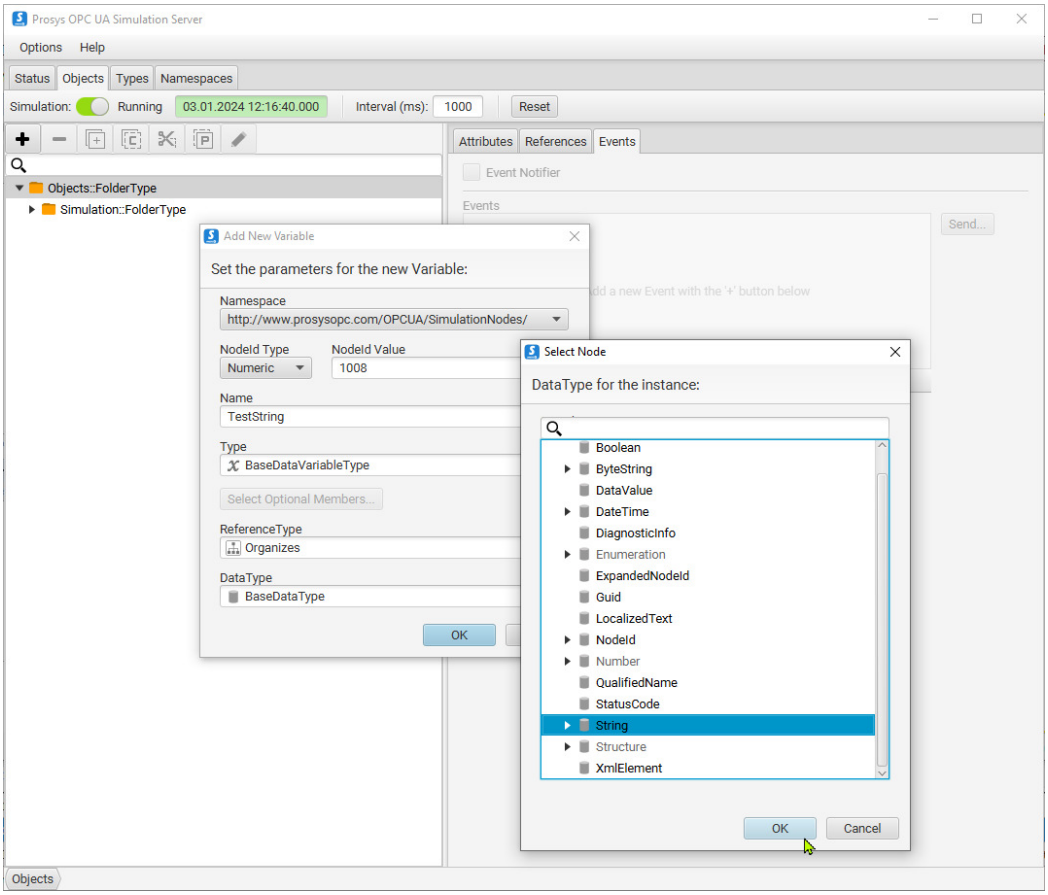
Open the OPC UA Simulation Server and navigate to the *Status* tab. The **Connection Address (UA TCP)** will be further needed for the Aurora Vision Studio client:



Please navigate to the *Objects* tab and with the right click expand *Add Node -> Add Variable...* and click on it:

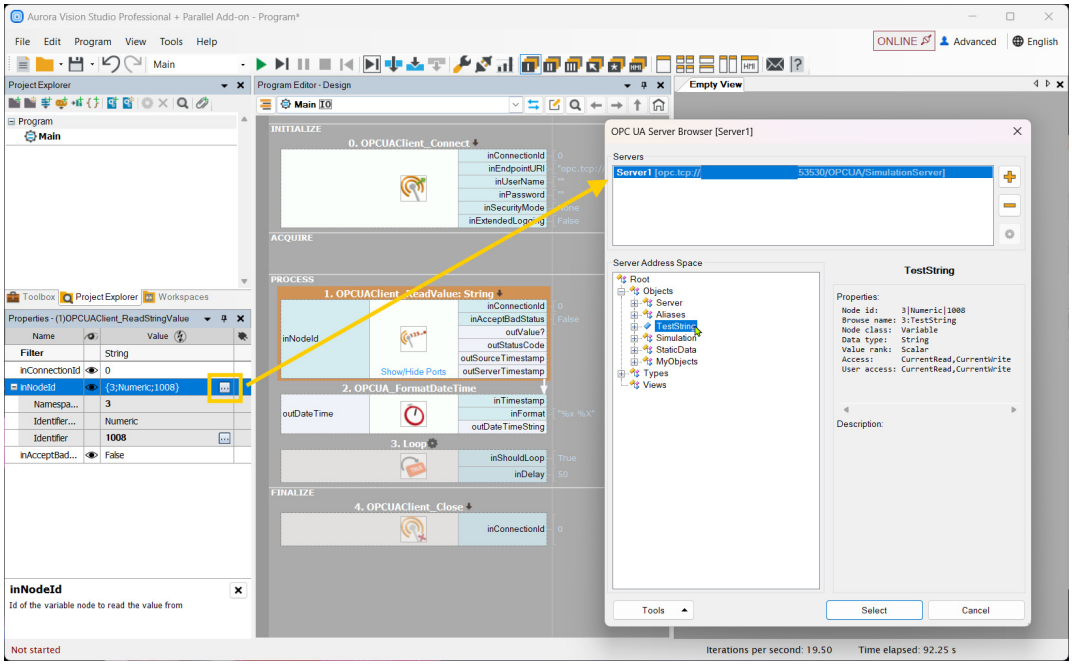


The *Add New Variable* should pop out. Change the *Name* to *TestString* and go to the *DataType* record to change from *BaseDataType* to *String*. Click *OK* to save:

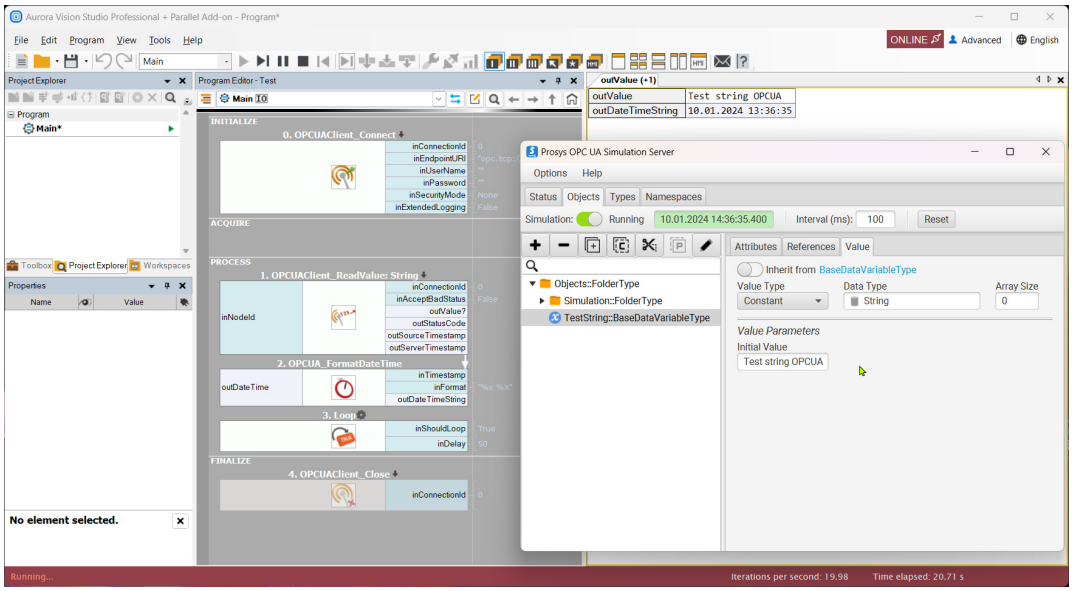


Create a solution in Aurora Vision Studio and place the OPCUAClient_Connect in the **INITIALIZE** section. Copy the **Connection Address (UA TCP)** from the server to the **inEndpointUrl** input of the filter. Place the OPCUAClient_ReadStringValue in the **PROCESS** section.

There is need to adjust the **inNodeId** input. You can do it by copying the **Node ID** from the simulator and pasting it to the **inNodeId.Identifier** input in Aurora Vision Studio (by expanding **inNodeId** input). It is also achievable with the click of the "3 dots" icon to open the OPC UA Server Browser:



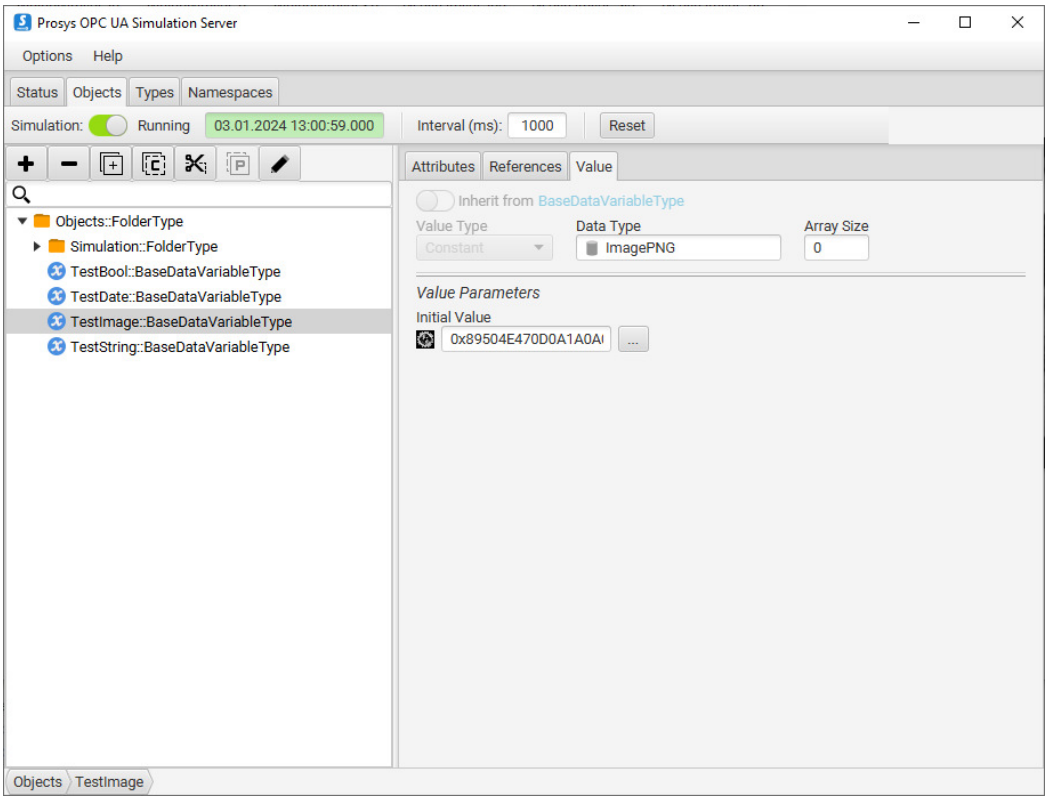
Add the **OPCUA_FormatDateTime** and connect the **outServerTimestamp** with the **inTimestamp** input as on the screen below:



Advanced program with OPC UA server simulator

This paragraph refers to the "OPCUA_ClientWithServerSimulator" Aurora Vision Studio example which can be found in the examples tree under *OPCUA Client To Server Communication* directory.

Add to the simulated server 4 nodes of type: **Boolean**, **Date**, **ImagePNG** and **String**. Change the initial value of the **ImagePNG** variable to some **.png** type image under the *Value* tab for this variable:



Open the "OPCUA Client To Server Communication" example and adjust nodes (if needed) like in the previous paragraph for the basic application. The application uses 5 nodes: 4 created above and one **double** (the simulated sine which can be found in the **Simulation::FolderType** in the server simulator).

To modify values on the server side navigate to the *Value* tab. You can see changes in the OPCUA simulated server under the *Attributes* tab (on the screen attached below inspecting a boolean value):

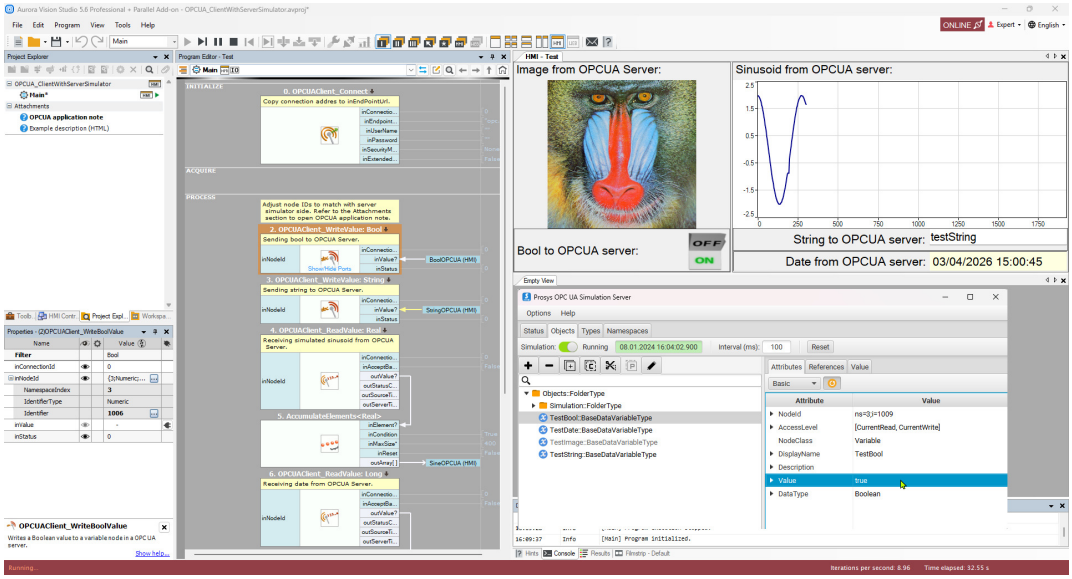


Image from OPCUA Server:
Sinusoid from OPCUA server:
Bool to OPCUA server:
OFF
ON
String to OPCUA server: testString
Date from OPCUA server: 03/04/2026 15:00:45

Interfacing Hilscher card (EtherCAT) to Aurora Vision Studio

Purpose and requirement

This document explains how to configure EtherCAT PLC with Aurora Vision Studio using Hilscher EtherCAT card.

Required equipment:

- Supported Hilscher EtherCAT card
- Aurora Vision Studio 5.1 or later

Hardware connection

PLC and PC must work in a common LAN network, hereinafter called the shared network. In most cases they are connected to each other through a network switch. For the EtherCAT interface, the Hilscher communication card must be configured as a slave device. In this tutorial, Port 1 of the Hilscher card has to be connected to EtherCAT Port 2 of the PLC.

The devices shall be connected as follows:

- Master device e.g. PLC to the shared network
- Hilscher card to PC
- Hilscher card to the EtherCAT port of PLC
- PC's network card to shared network

Before proceeding to the further point, make sure that all devices are powered up.

Configuring Hilscher Devices

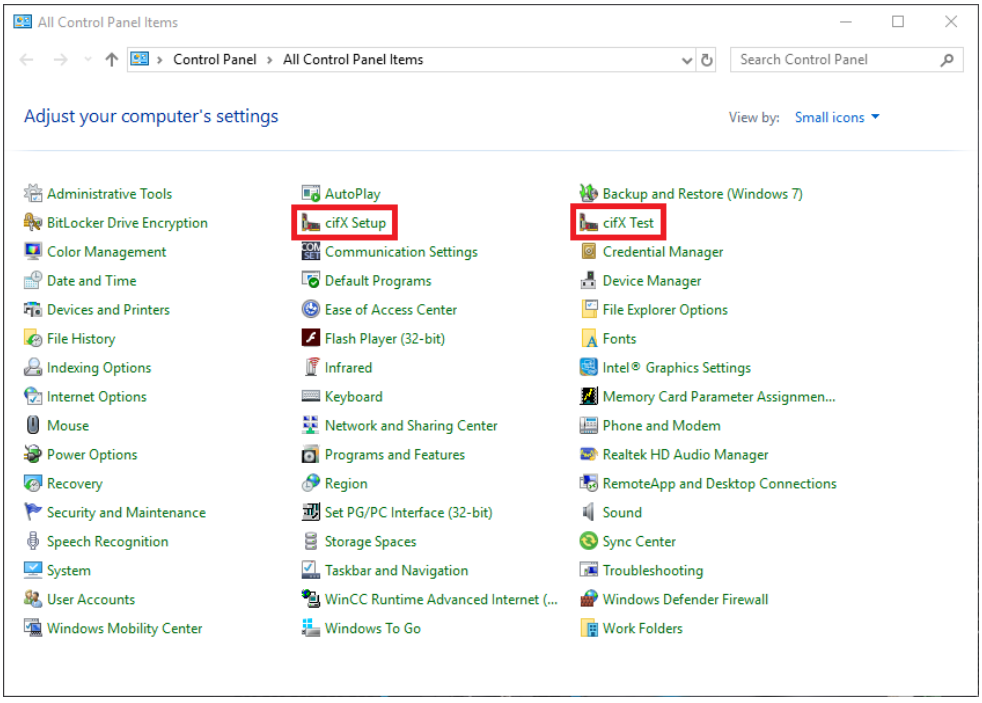
Be sure to have the newest version of the following drivers:

- List of the EtherCAT firmware (Slave V4 recommended): download link
- cifX Driver Setup software (1.5.0.0 version recommended): download link
- SYCON.net to configure Hilscher card and generate configuration files: download link

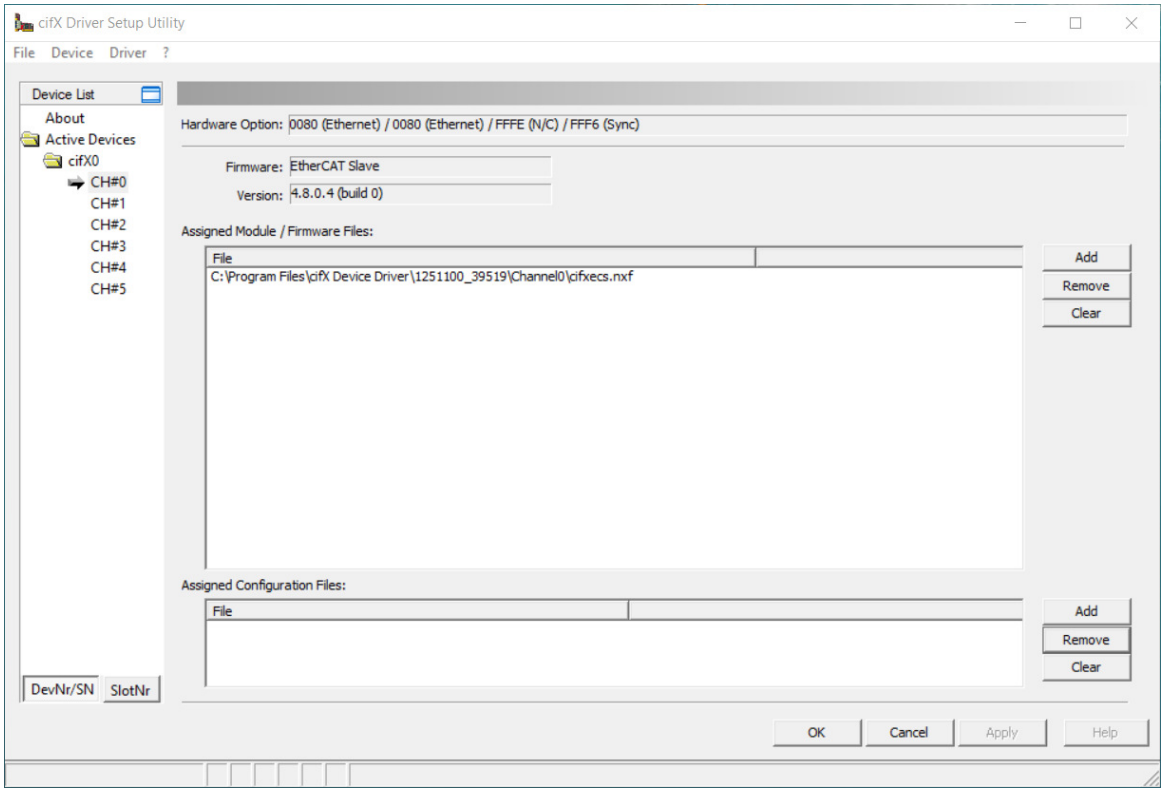
Make sure you have the recommended driver installed, the newest SYCON.net and firmware prepared.

1) Configuration using cifX Driver Setup Software

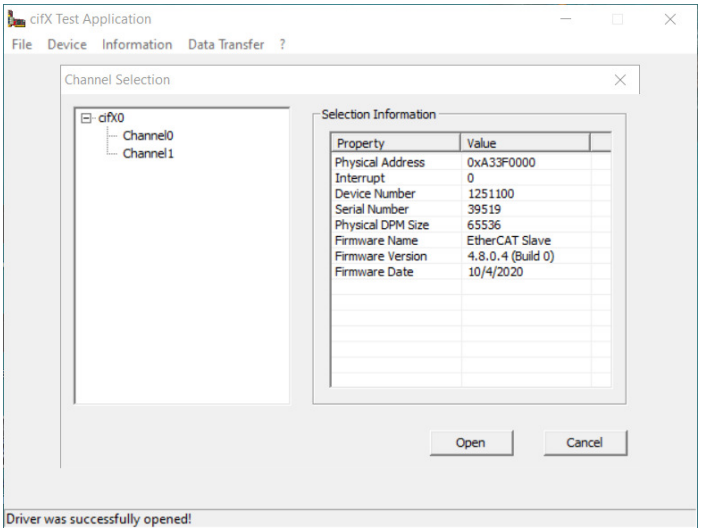
The easiest way to install the firmware to the cifX Driver is using the cifX Driver Setup Software. You can find tools for this software for instance using Control Panel menu as in the picture below.



Using cifX Setup application select the channel you would like to configure (usually channel – CH#0) and remove preexisting firmware by clicking "Clear". Then click "Add" to add a new firmware file and navigate to the downloaded EtherCAT Slave firmware (path \COMSOL-ECS V4.8.0.4\Firmware\cifX). Then click "Apply" and finish configuration with "OK".

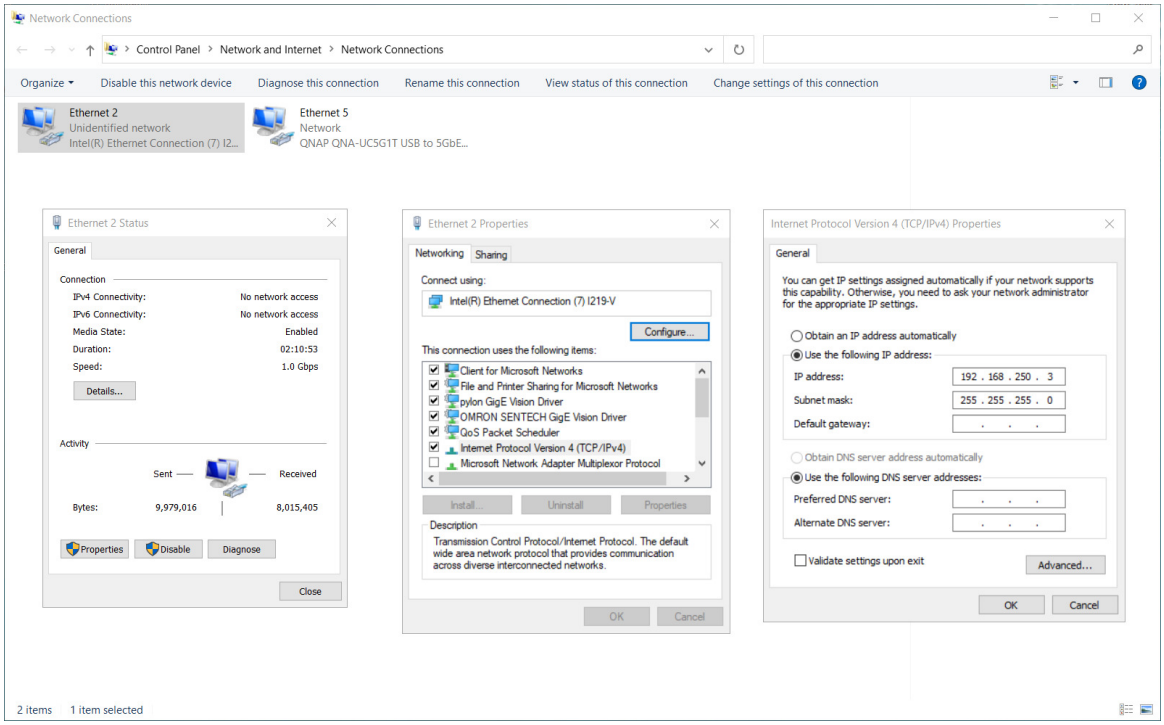


To make sure that firmware is installed properly, you can open cifX Test software and then navigate to Device -> Open to achieve the result shown below.



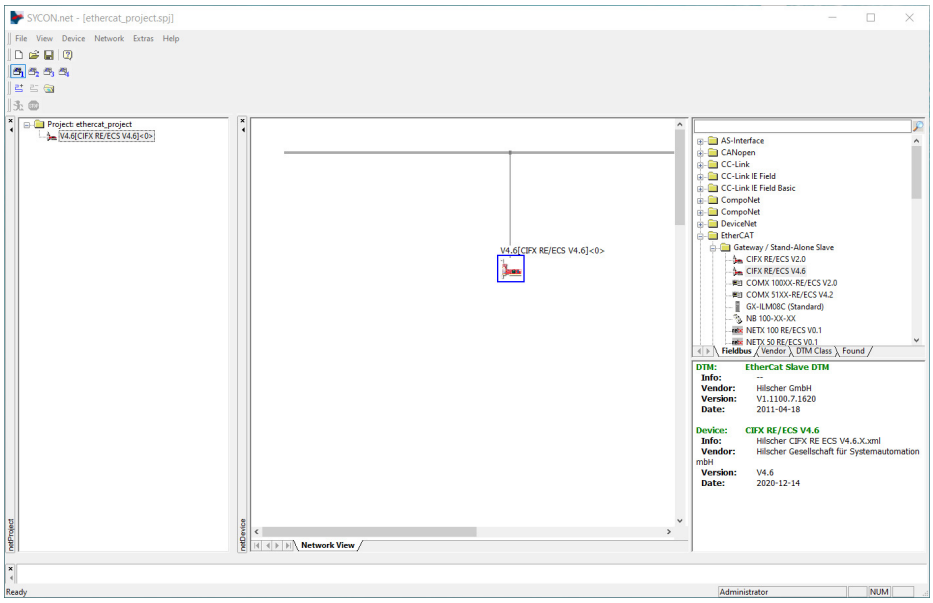
2) Configuration of the network card

It might be necessary to change the IP of the network card from dynamic to static. In this tutorial the following settings are used.

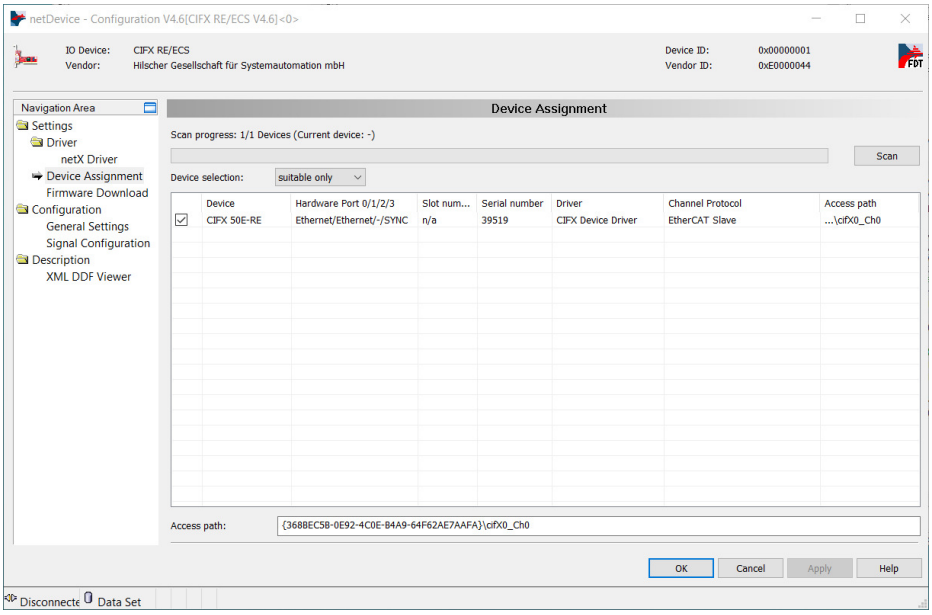


3) Configuration using SYCON.net

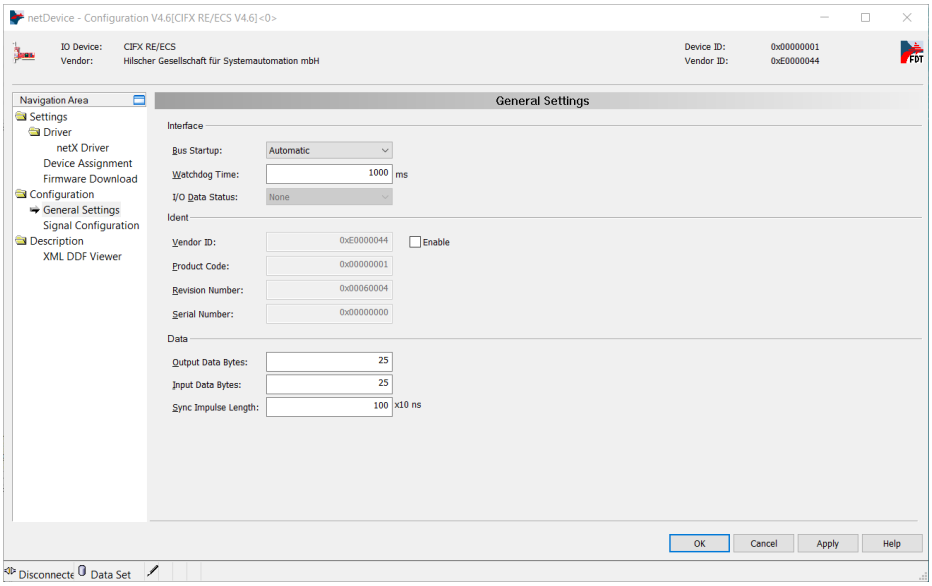
Configuration files are generated in SYCON.net software. Firstly, pick a proper device and drag it to the gray bus on the "Network View". This device you can find navigating to "EtherCAT -> Gateway / Stand - Alone Slave -> CIFX RE/ECS V4.6".



To open the configuration card double click on dropped icon and in the popped-up menu navigate to "Settings/Device Assignment". Then scan for devices by clicking "Scan", mark found one it with a tick like on the image below and then click "Apply".

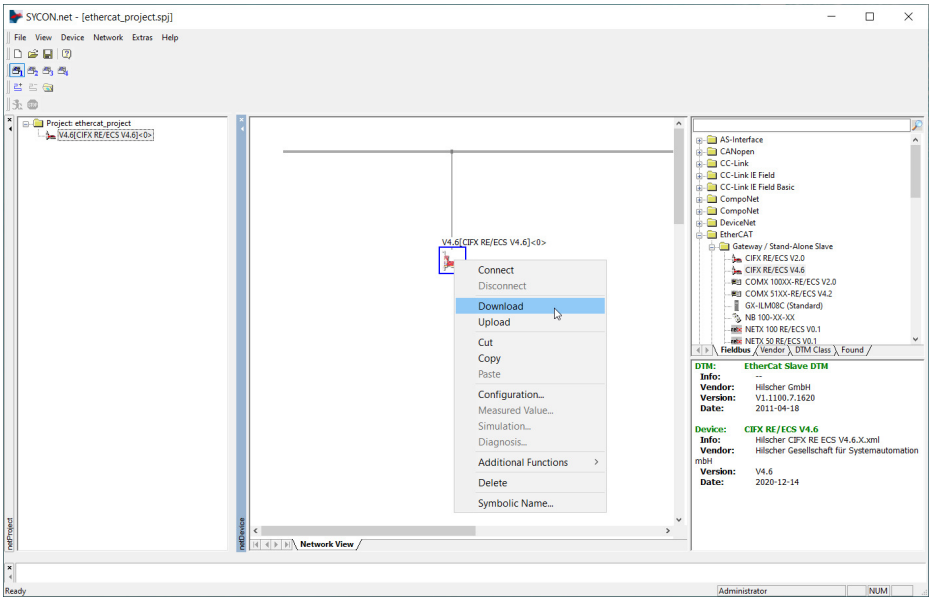


In "Configuration/General Settings" you can change the configuration of memory blocks for IN/OUT operations. By default, the Data length is set to 200 bytes. All subsequent PLC data blocks mentioned in this document are configured for 25 bytes. In case you need memory blocks of a different size, you can change the Data length from 0 up to 256 bytes.



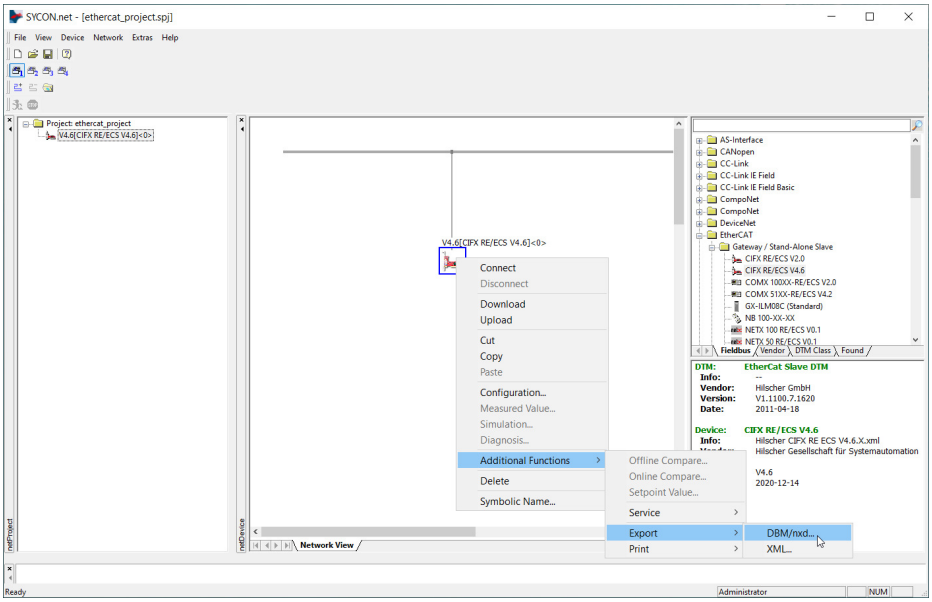
Confirm all changes by clicking "Apply" and then click "OK".

Then you have to download the configuration to the device like on the screen below.



The final step is to generate configuration files for Aurora Vision Studio. You can do this by right-clicking on the device icon, then navigating to "Additional Functions -> Export -> DBM/nxd..", entering your configuration name and clicking "Save". You can now close SYCON.net for this example,

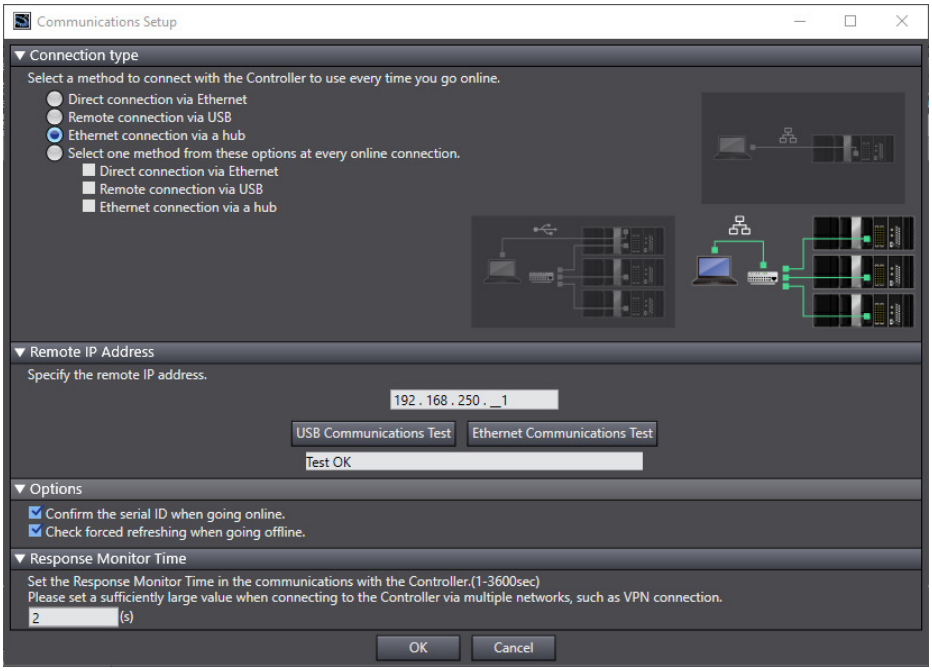
remember to save your project before.



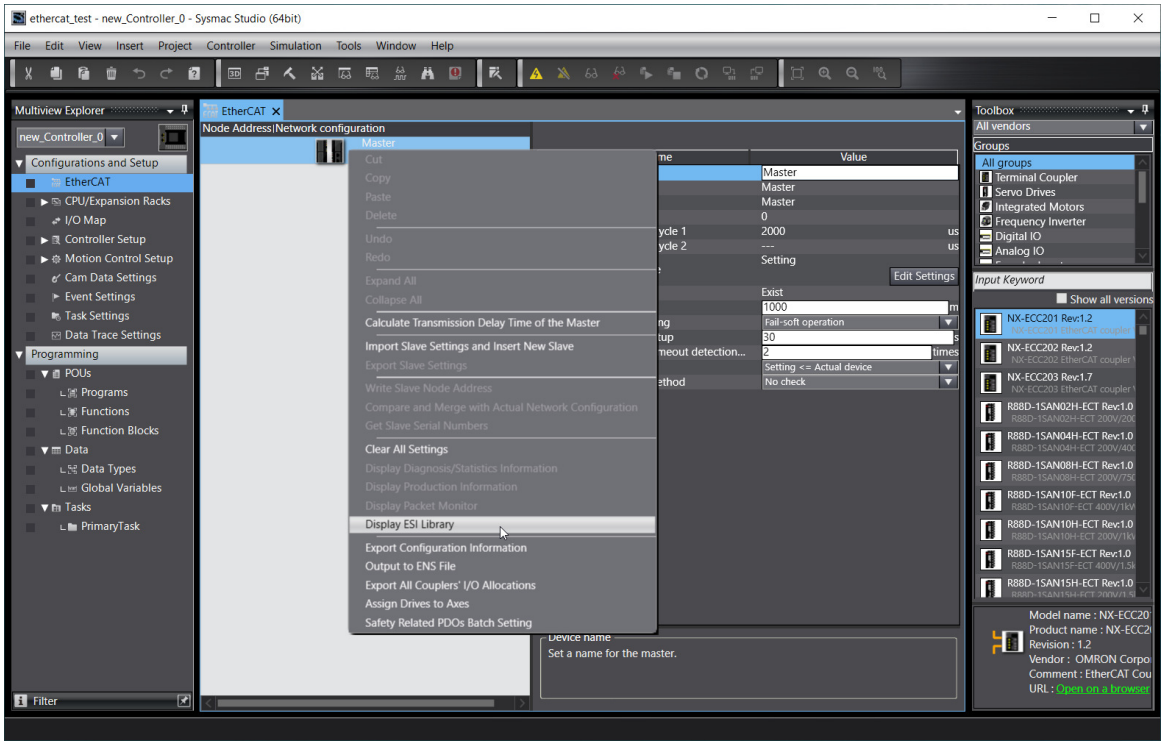
4) Example configuration of EtherNet/IP PLC

Below you will find an example configuration process for EtherCAT PLC Omron NXJP2-9024DT in Sysmac Studio software.

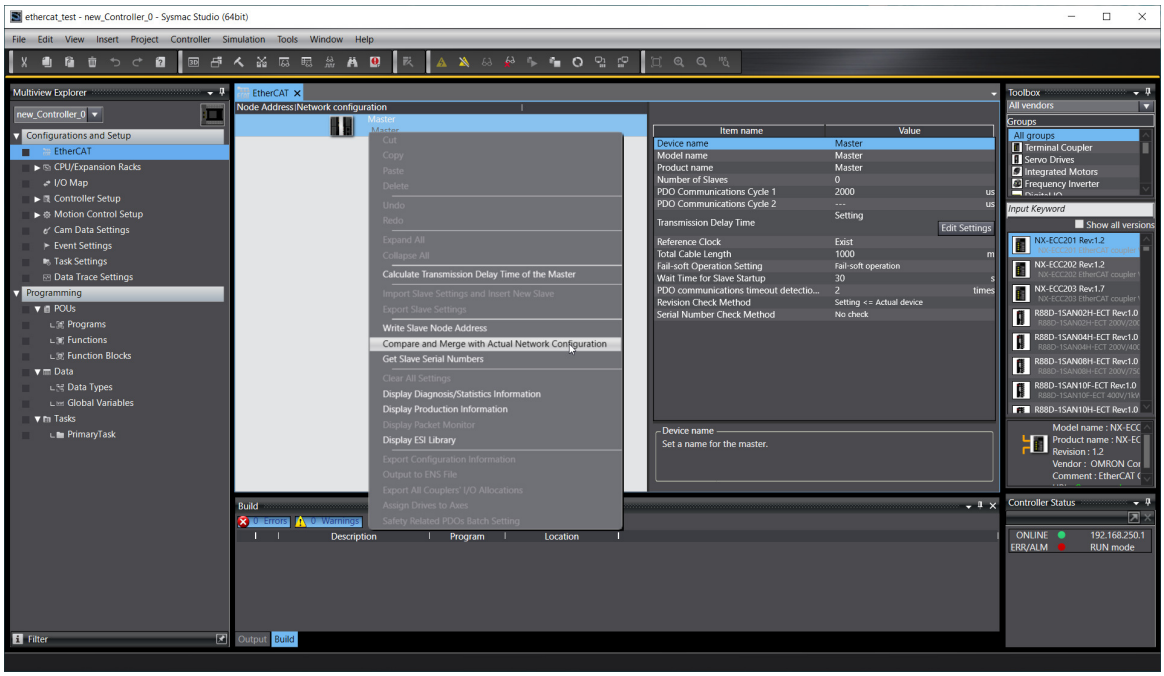
For this moment it is essential to connect with a PLC device, to do this in Sysmac Studio you need to navigate to "Controller -> Communication Setup...". There might be an error in Ethernet Communication Test if your network card is configured as DHCP (**Configuration of the network card** section in this tutorial). If communication with PLC is established you will receive "Test OK".



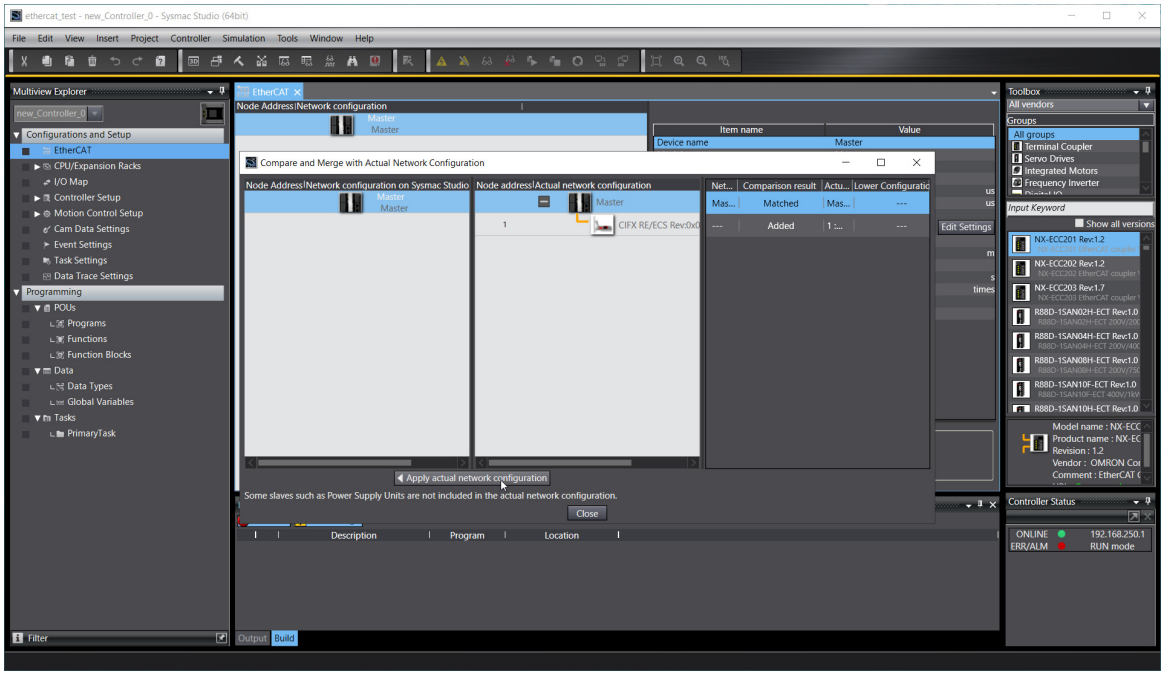
Add support for Hilscher card using ESI Library configuration (path \COMSOL-ECS V4.8.0.4\EDS). To do this in Sysmac Studio you need to navigate to EtherCAT tab, right-click the Master icon and choose "Display ESI Library".



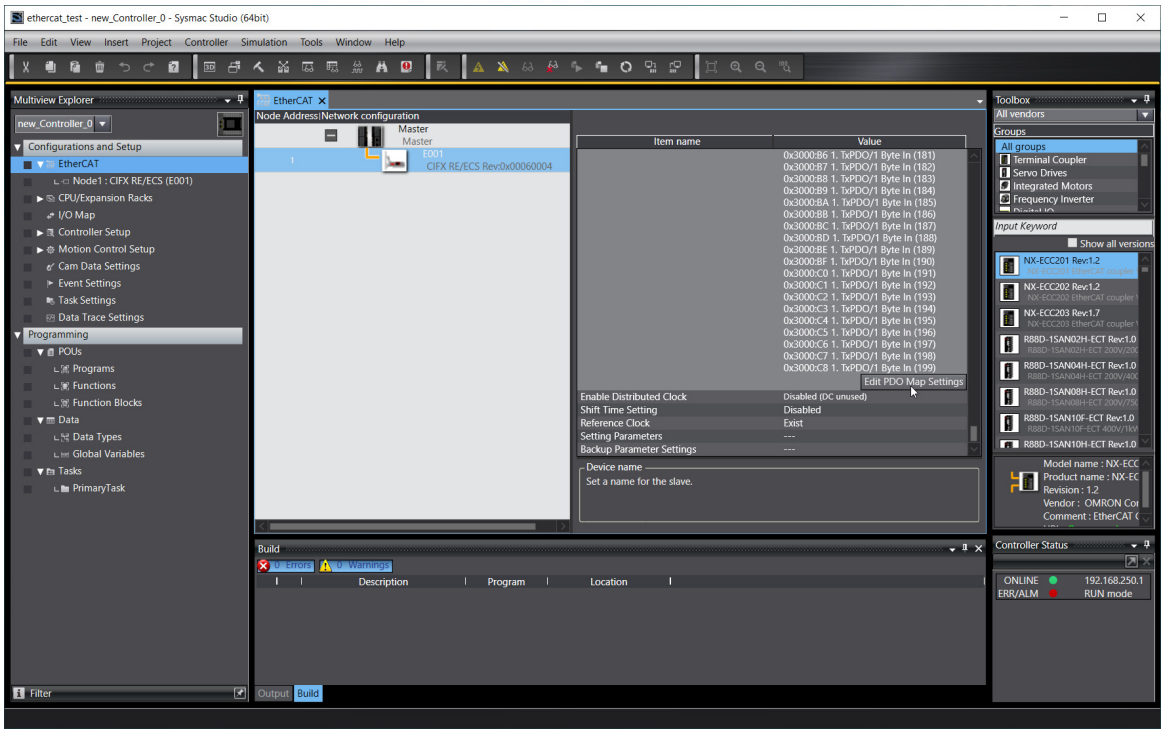
If communication with PLC is established and Hilscher Card is connected to the EtherCAT port of PLC it is possible to load EtherCAT configuration to the PLC. To accomplish that right-click Master icon and choose "Compare and Merge with Actual Network Configuration".



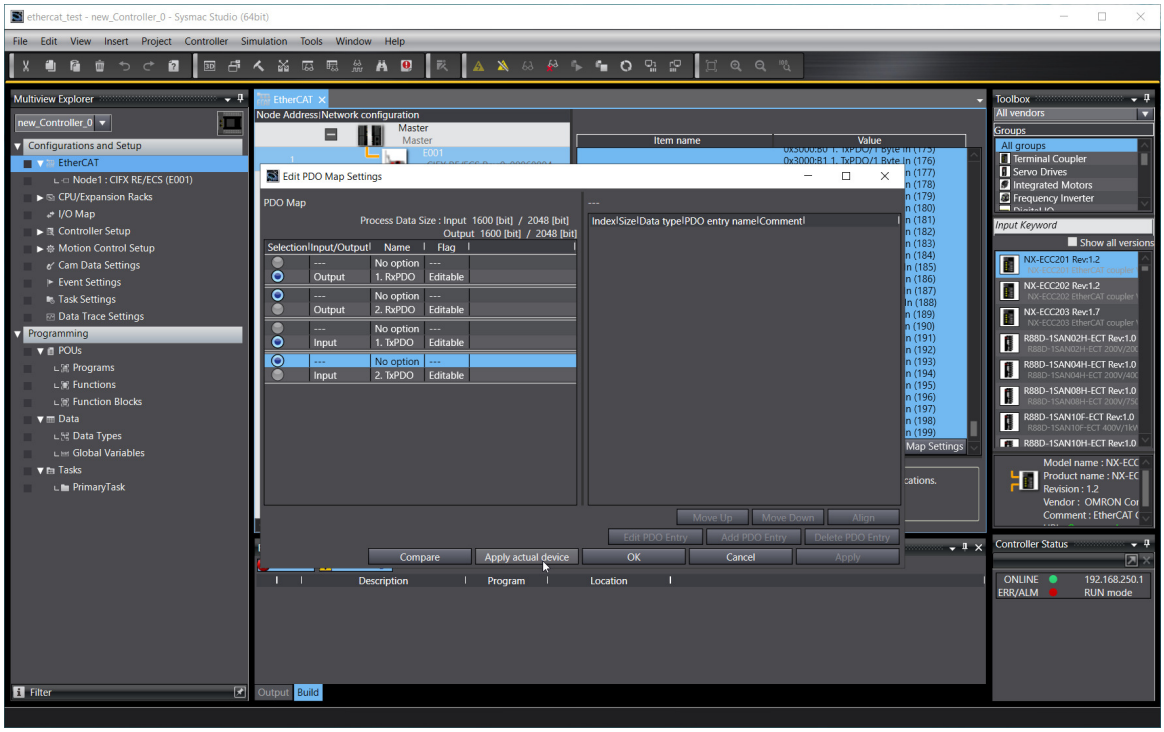
If everything is connected correctly Compare and Merge with Actual Network Configuration window should pop-up. To match configuration in software to configuration in hardware click "Apply actual network configuration".



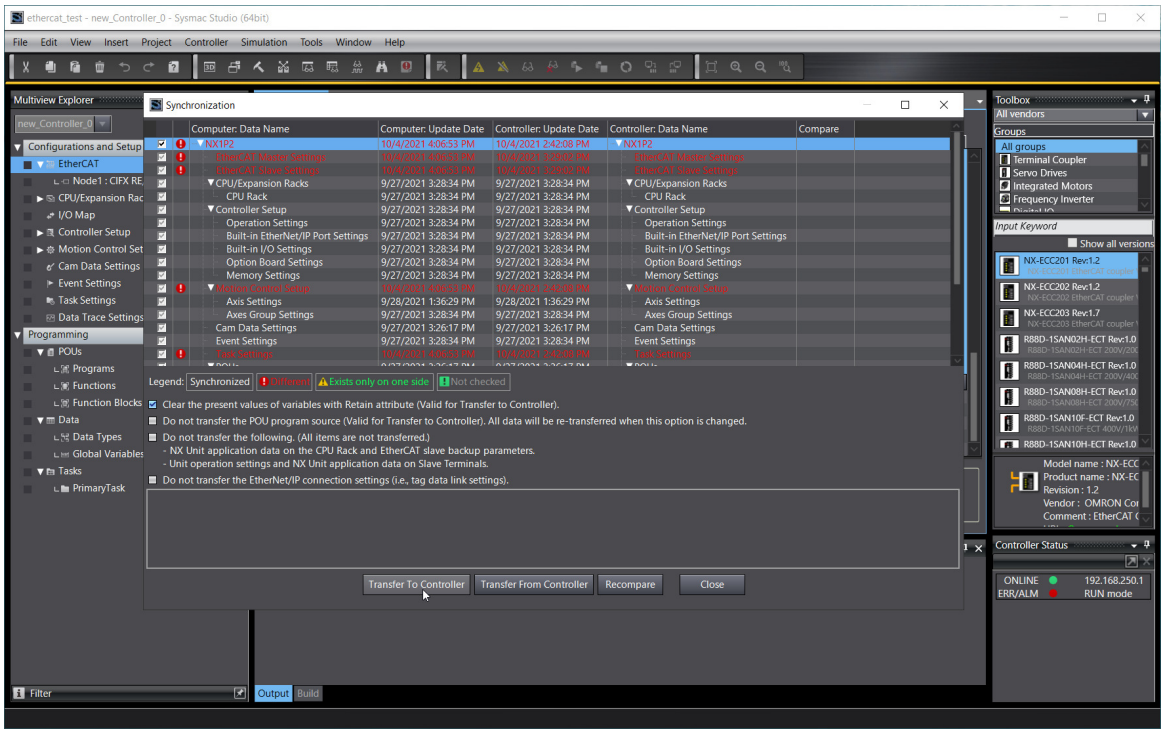
Choose previously added slave device and navigate to the "Edit PDO Map Settings" button and click it.
This step is essential to match the configuration from SYCON.net.



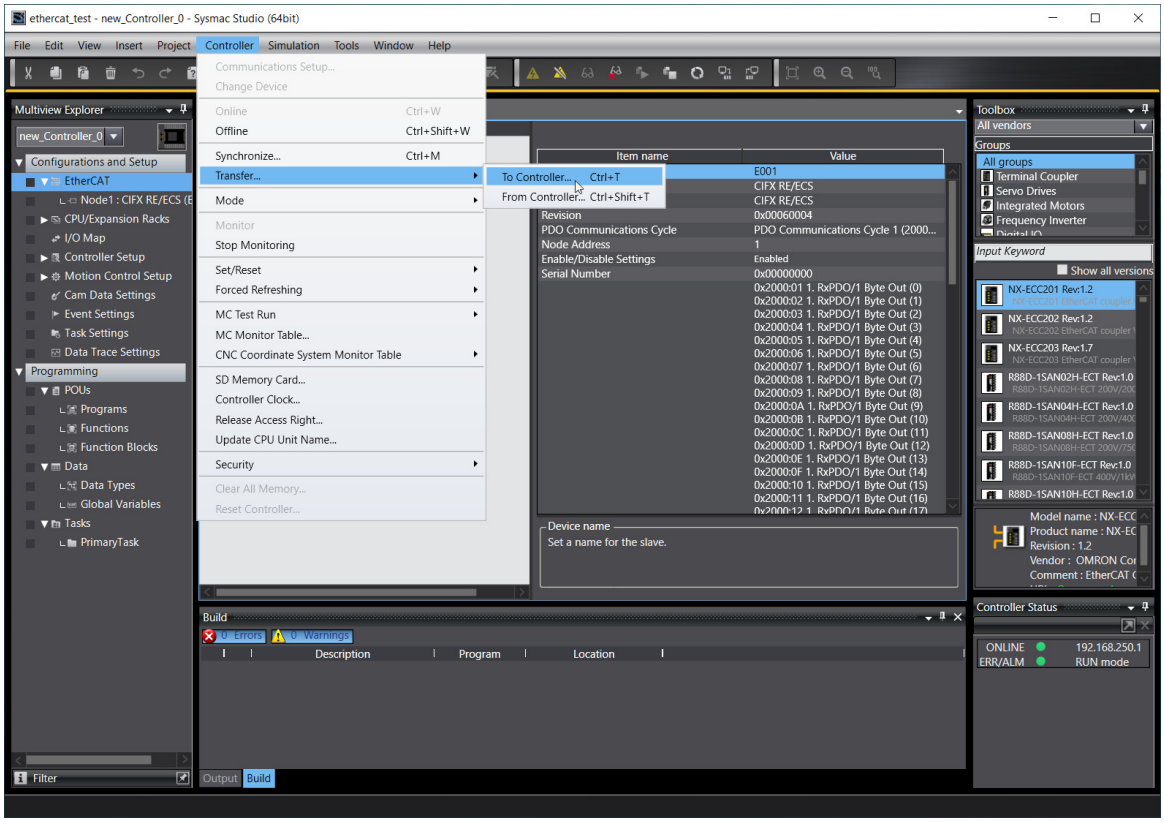
In Edit PDO Map Settings window click "Apply actual device". Then click "Apply" and "OK".



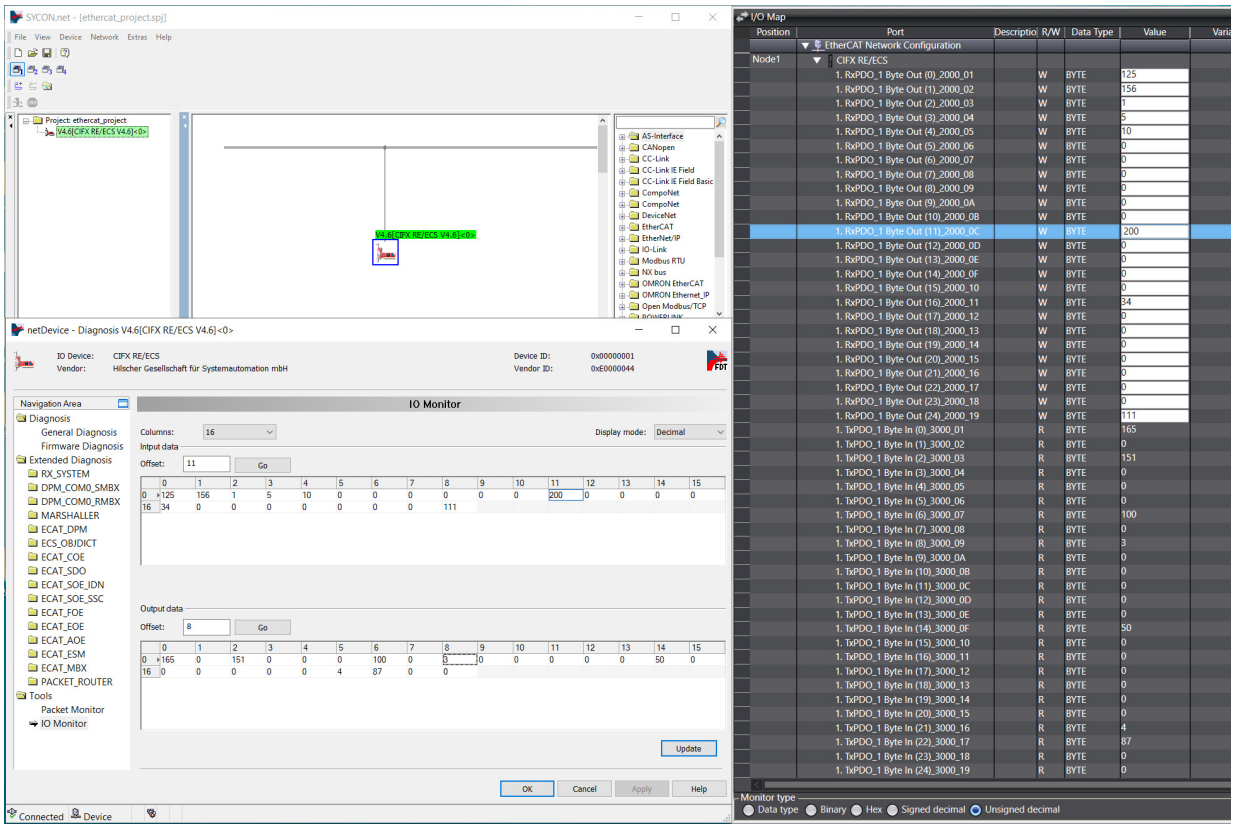
In the online mode of the PLC controller navigate to "Controller -> Synchronize...". Mark with ticks all the changes and click "Transfer To Controller".



In the online mode of the PLC controller navigate to "Controller -> Transfer... -> Transfer To Controller" and download changes.

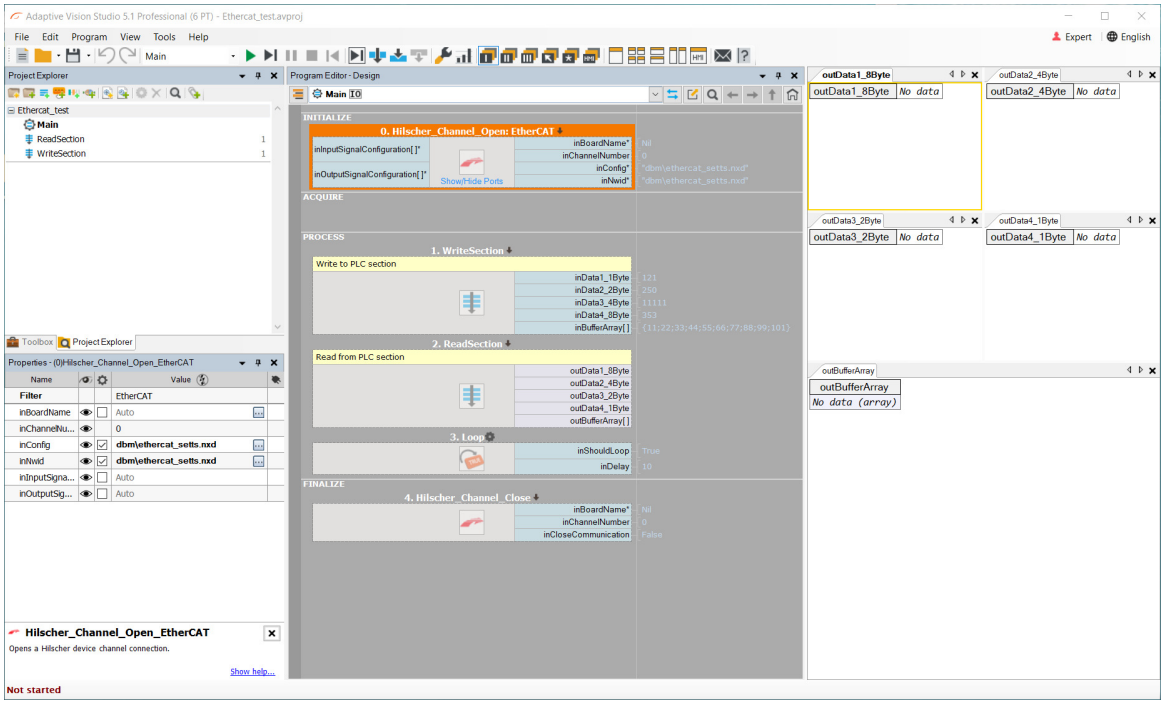


Below you can find a test of input and output memory using a watchtable in Sysmac Studio and IO Monitor in SYCON.net. To turn on IO Monitor you have to right-click on the icon of the card in SYCON.net project and click "Connect". Then double click this icon and navigate to "Tools -> IO Monitor". By clicking "Update" you can send a changed frame to the PLC.

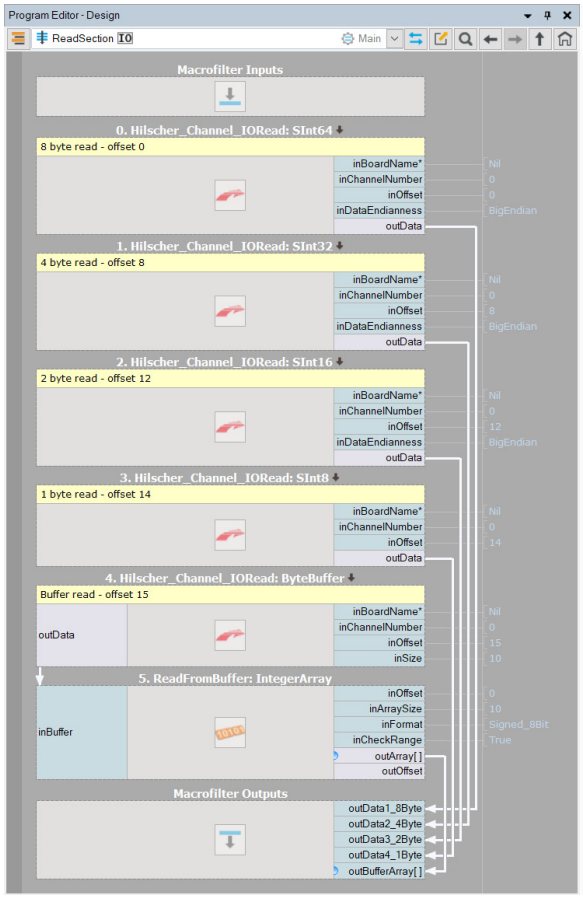


5) Example configuration in Aurora Vision Studio

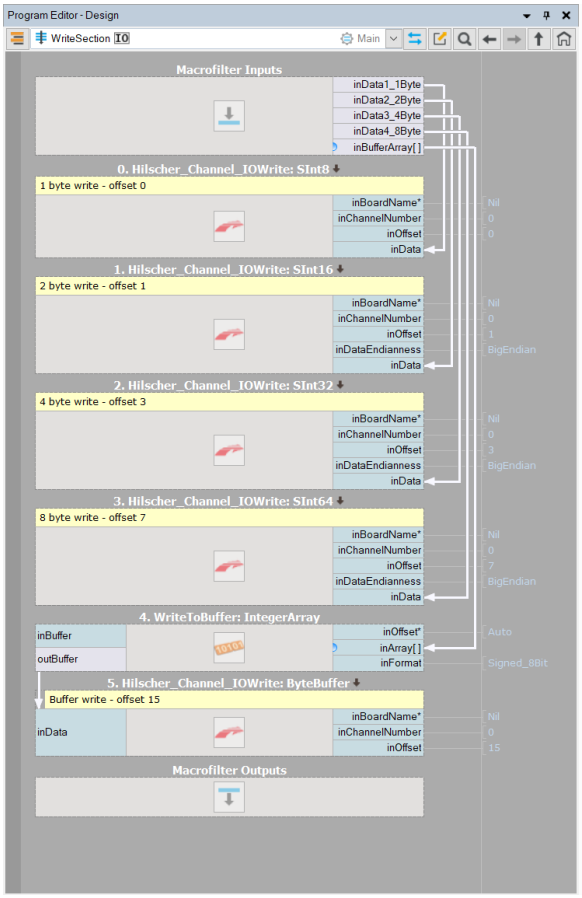
To use EtherCAT filters in Aurora Vision Studio first you need to attach configuration files from SYCON.net to Hilscher_Channel_Open_EtherCAT filter in INITIALIZE section. The configuration files generated in the previous step are now required in inConfig (xxx.nxd) and inNwid (xxx_nwid.nxd) properties of that filter. Below you can find two Step macrofilters responsible for writing data to PLC and receiving data from it. To have cyclic communication you have to place Loop macrofilter at the end of PROCESS section. In FINALIZE section place Hilscher_Channel_Close macrofilter.



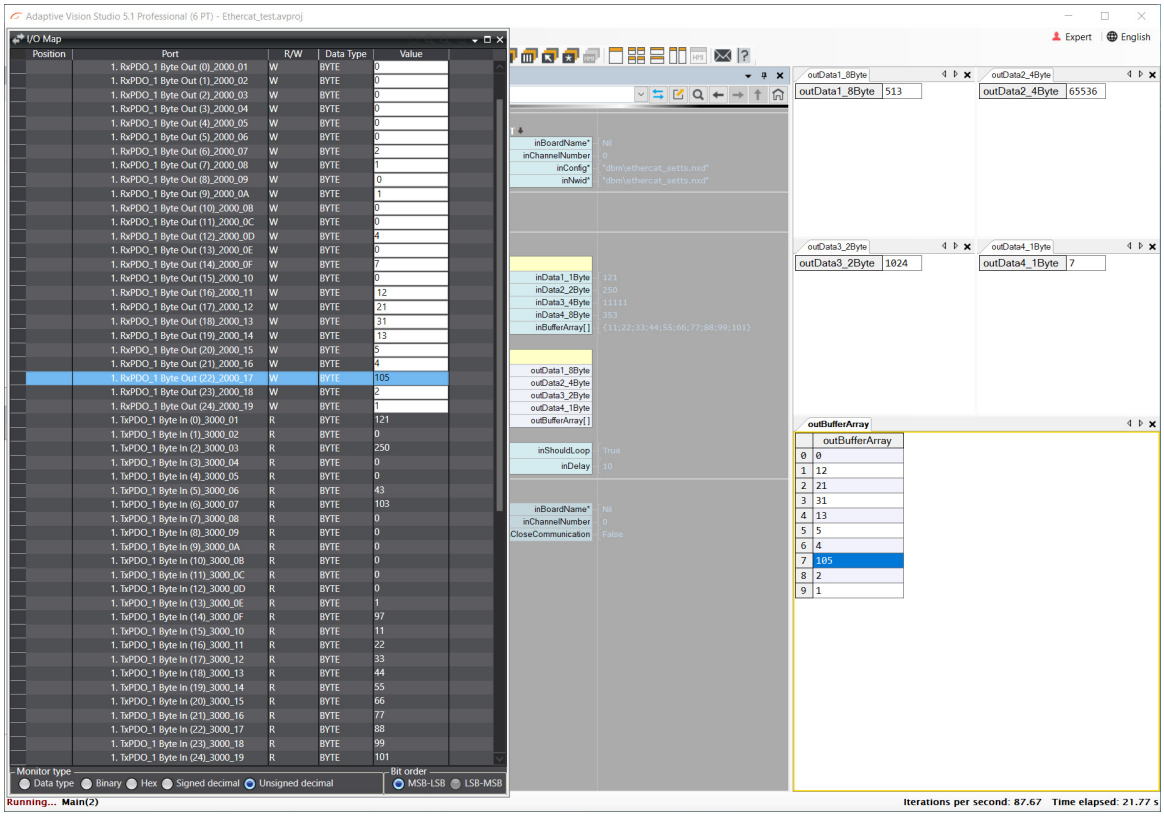
In step macrofilter ReadSection you can find for example Hilscher_Channel_IORead_SInt8 filter which writes 8 bytes of signed data to the predefined memory area. Using different offsets for each macrofilter enables access to different variables created in PLC (**Example configuration of EtherCAT PLC** section of this tutorial).



WriteSection step macrofilter has Hilscher_Channel_IOWrite filters (for instance Hilscher_Channel_IOWrite_SInt8) filters with adequate offsets and data types used to match PLC data variables configuration.

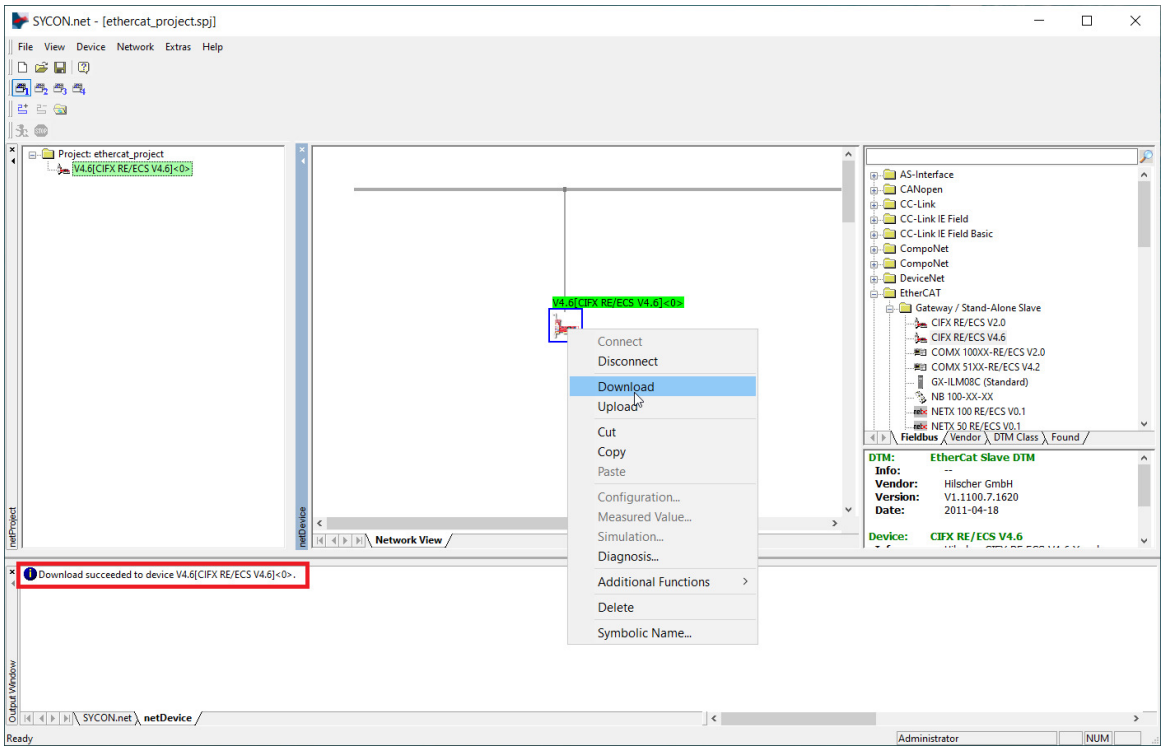


Below is presented reading and writing to PLC with Aurora Vision Studio and Sysmac Studio I/O Map. Decimal values of variables are dependent on the data type used.



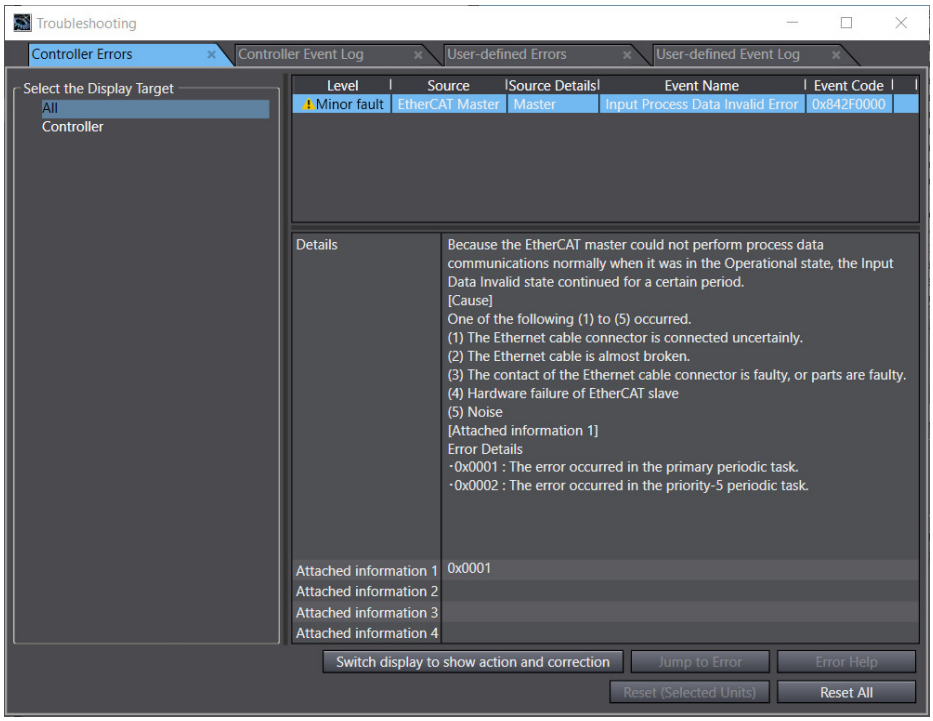
Troubleshooting

1. Make sure that the current program setting is loaded to the Hilscher card. If not please use SYCON.net application to connect and download settings to the devices as shown in the picture below.

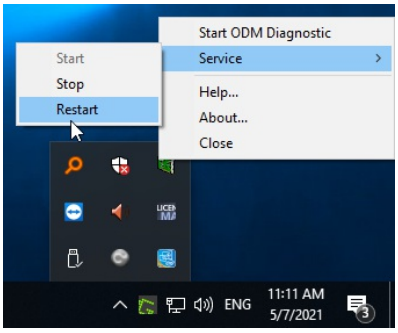


2. Check in your device in Device Assignment. This step was described in **Configuration using SYCON.net** section of this tutorial.

3. Using Sysmac Studio you can encounter Minor fault Controller Error like below which typically does not affect data exchange. Replacing Ethernet cables or hardware could possibly make this warning disappear.



4. After changes it may be necessary to restart ODMV3 service.



5. If none of the above advice has helped, please restart your computer.

Interfacing Hilscher card (EtherNet/IP) to Aurora Vision Studio

Purpose and requirement

This document explains how to configure EtherNet/IP PLC with Aurora Vision Studio using Hilscher EtherNet/IP card.

Required equipment:

- Supported Hilscher EtherNet/IP card
- Aurora Vision Studio 5.1 or later

Hardware connection

All devices must work in a common LAN network, hereinafter called the shared network. In most cases they are connected to each other through a network switch.

The devices shall be connected as follows:

- Master device (e.g. PLC) to the shared network
- Hilscher card to PC
- Hilscher card to the shared network
- PC's network card to shared network

Before proceeding to the further point, make sure that all devices are powered up.

Configuring Hilscher Devices

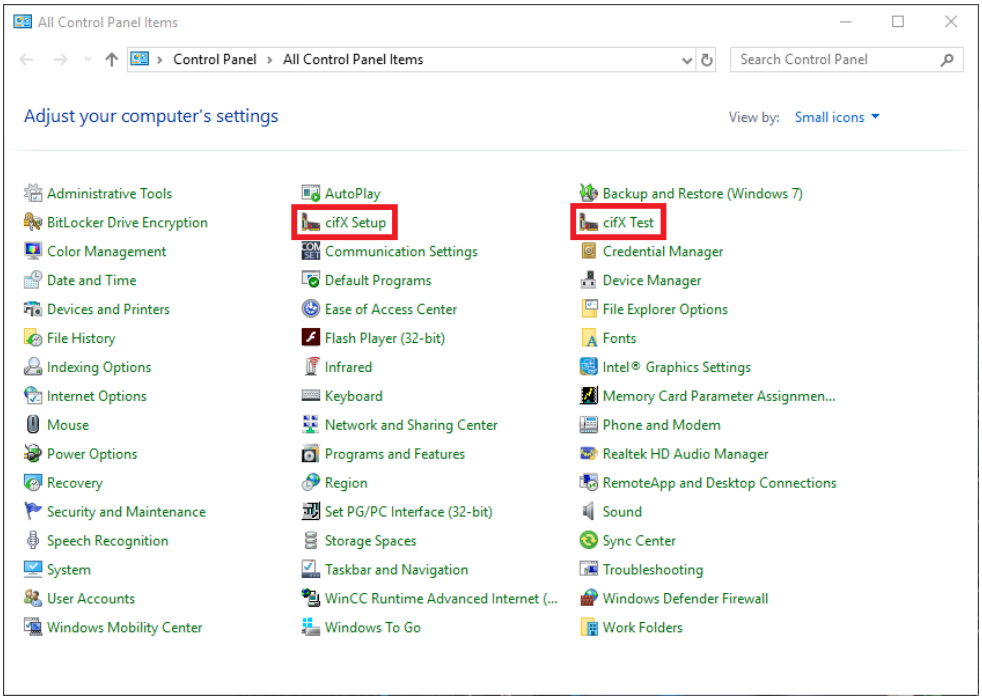
Be sure to have the newest version of the following drivers:

- List of the EtherNet/IP firmware: download link (for CIFX 50E-RE download V3.x version)
- cifX Driver Setup software (1.5.0.0 version recommended): download link
- SYCON.net to configure Hilscher card and generate configuration files: download link

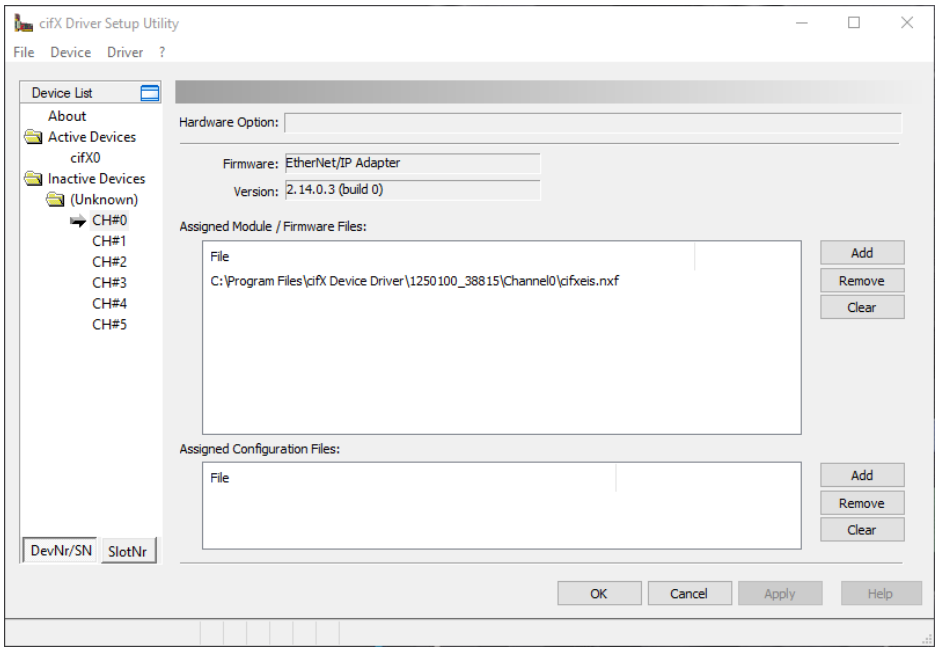
Make sure you have the recommended driver installed, the newest SYCON.net and firmware prepared.

1) Configuration using cifX Driver Setup Software

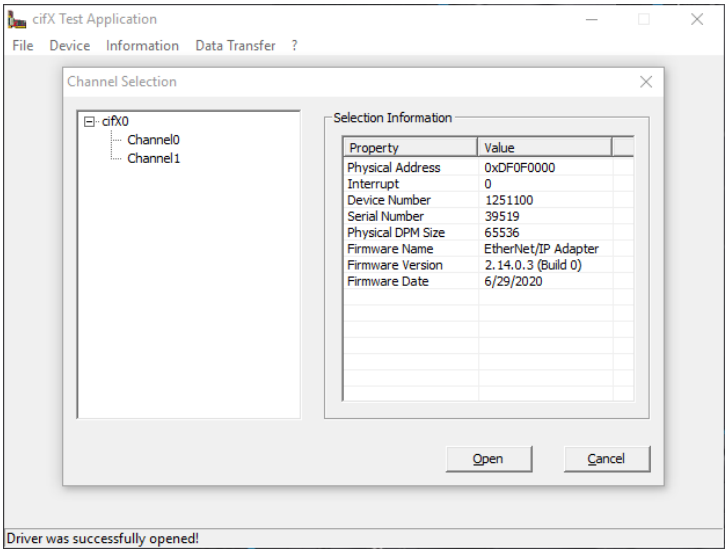
The easiest way to install the firmware to the cifX Driver is using the cifX Driver Setup Software. You can find tools for this software for instance using Control Panel menu as in the picture below.



Using cifX Setup application select the channel you would like to configure (usually channel - CH#0) and remove preexisting firmware by clicking "Clear". Then click "Add" to add a new firmware file and navigate to the downloaded Ethernet/IP Adapter firmware (path \COMSOL-EIS V2.15.0.1\Firmware\cifX). Then click "Apply" and finish configuration with "OK".

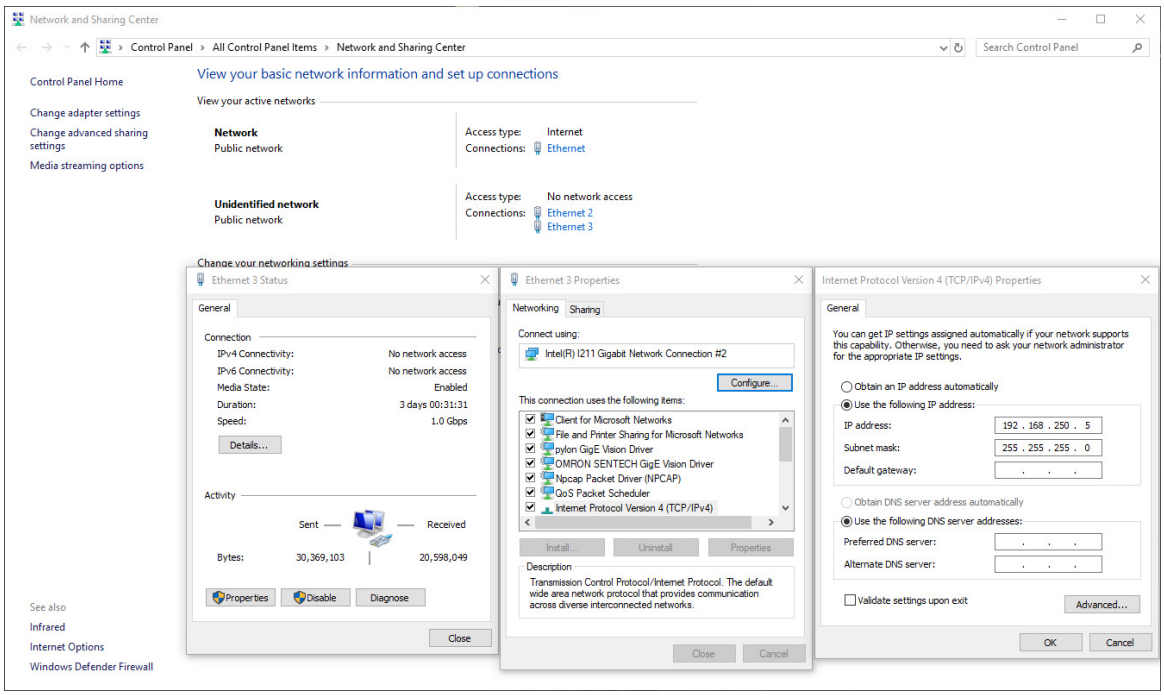


To be sure that firmware is installed properly you can open cifX Test software and then navigate to Device -> Open to achieve the result like below.



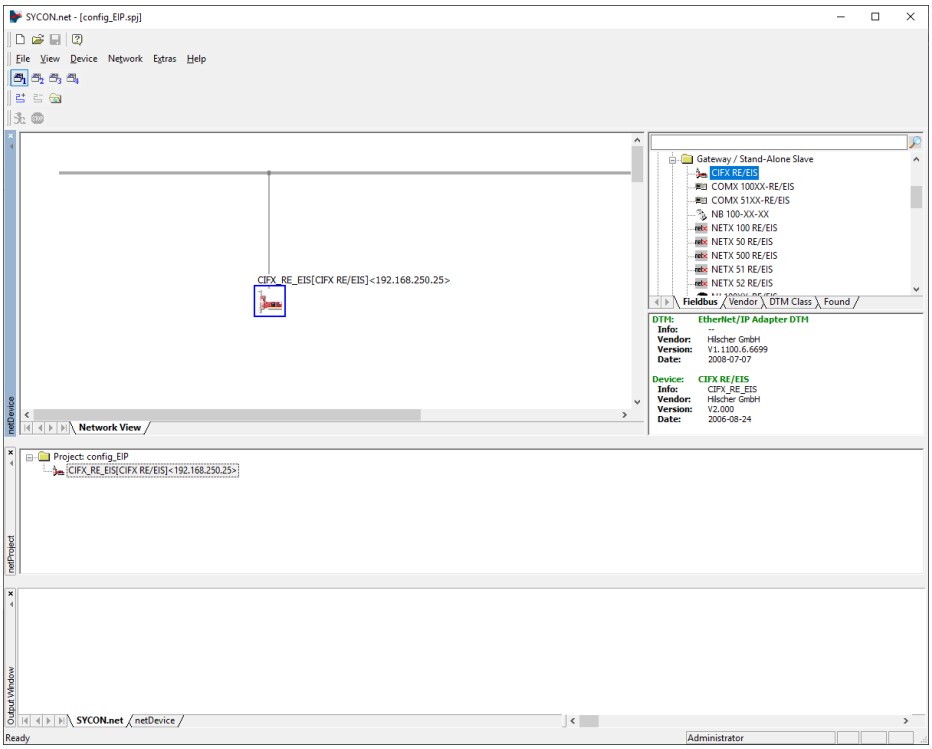
2) Configuration of the network card

It might be necessary to change the IP of the network card from dynamic to static. In this tutorial the following settings are used.

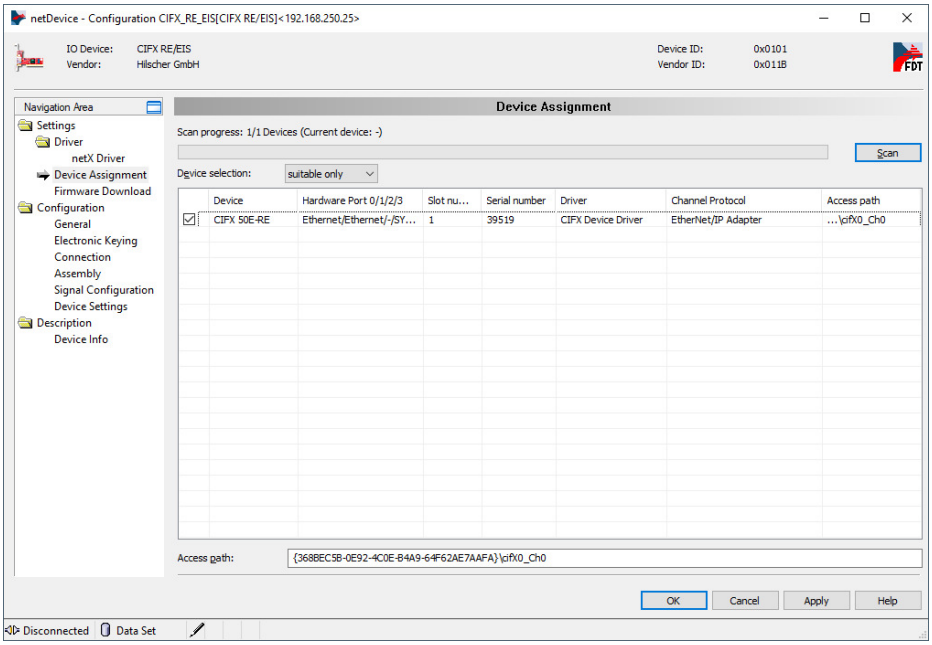


3) Configuration using SYCON.net

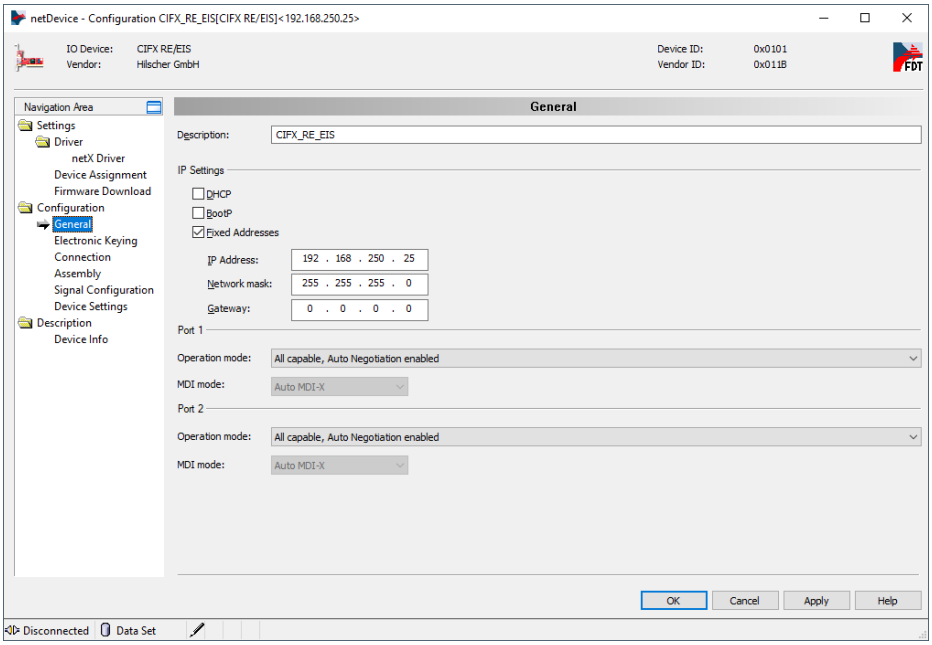
Configuration files are generated in SYCON.net software. Firstly, pick a proper device and drag it to the gray bus on the "Network View". This device you can find navigating to "EtherNet/Ip -> Gateway / Stand – Alone Slave -> CIFX RE/EIS".



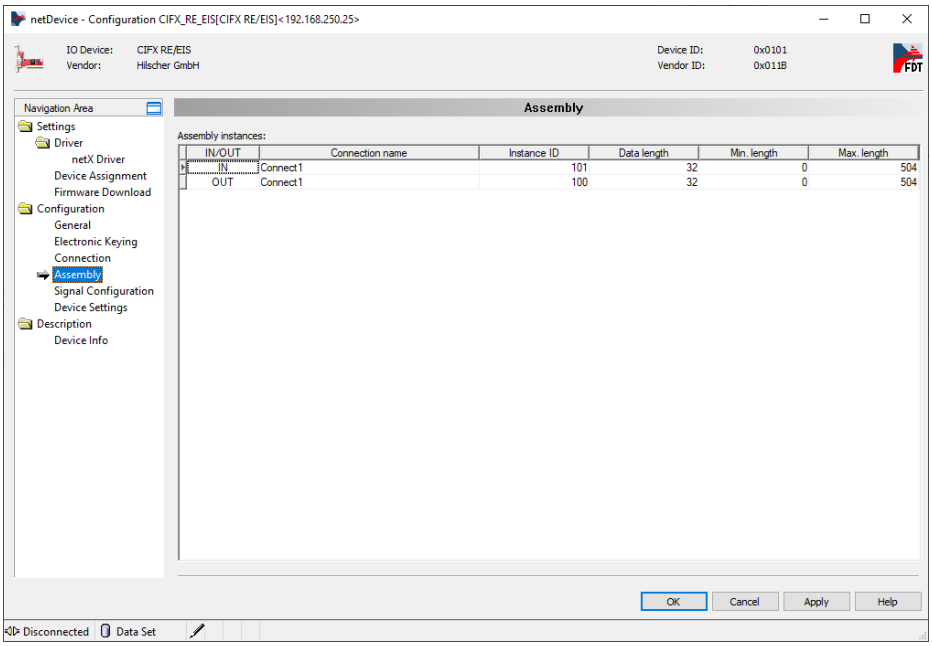
To open the configuration card double click on dropped icon and in the popped-up menu navigate to "Settings/Device Assignment". Then scan for devices by clicking "Scan", mark found one it with a tick like on the image below and then click "Apply".



In "Configuration/General" you can change the IP of the device from DHCP to Static like on the below screen.

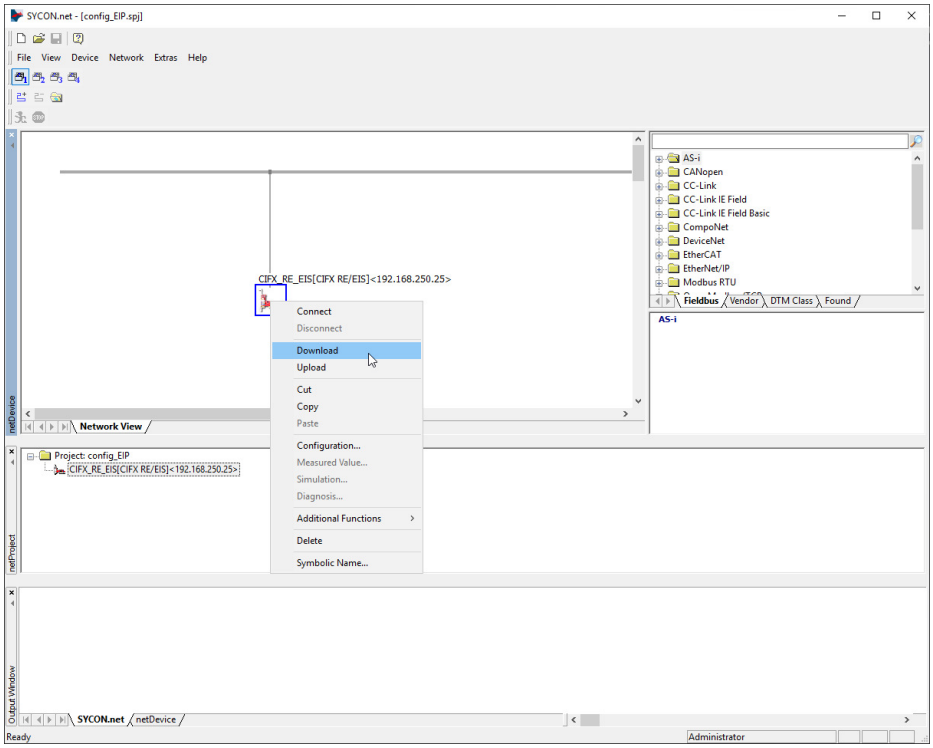


In "Configuration/Assembly" you can change the configuration of memory blocks for IN/OUT operations. By default, the Data length is set to 32 bytes. All subsequent PLC data blocks mentioned in this document are configured for 32 bytes. In case you need memory blocks of a different size, you can change the Data length from 0 up to 504 bytes.

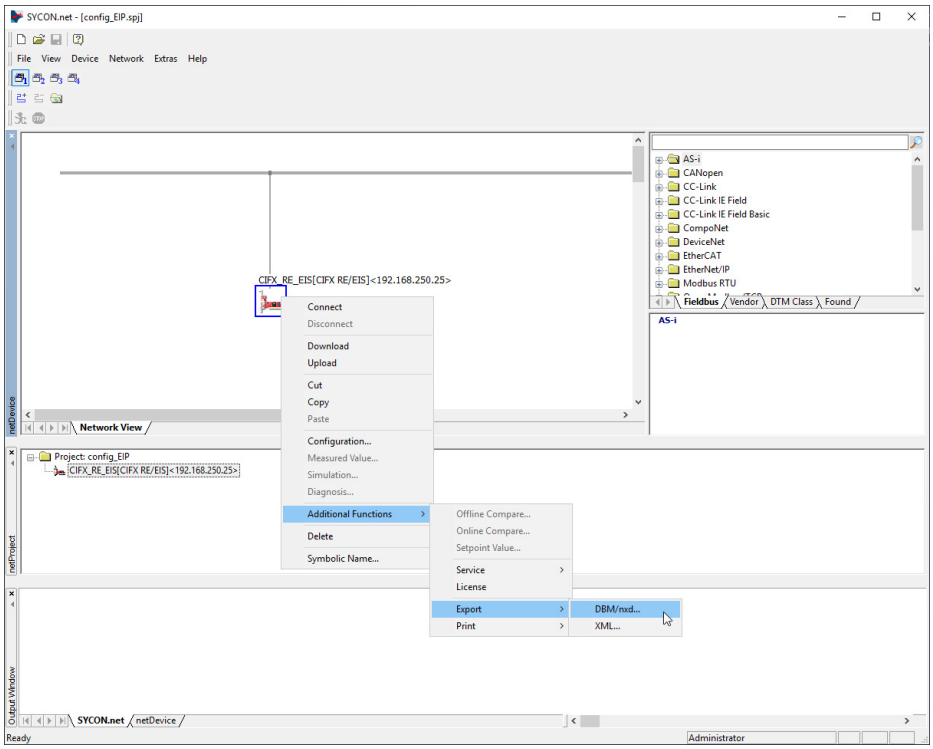


Confirm all changes by clicking "Apply" and then click "OK".

Then you have to download the configuration to the device like on the screen below.



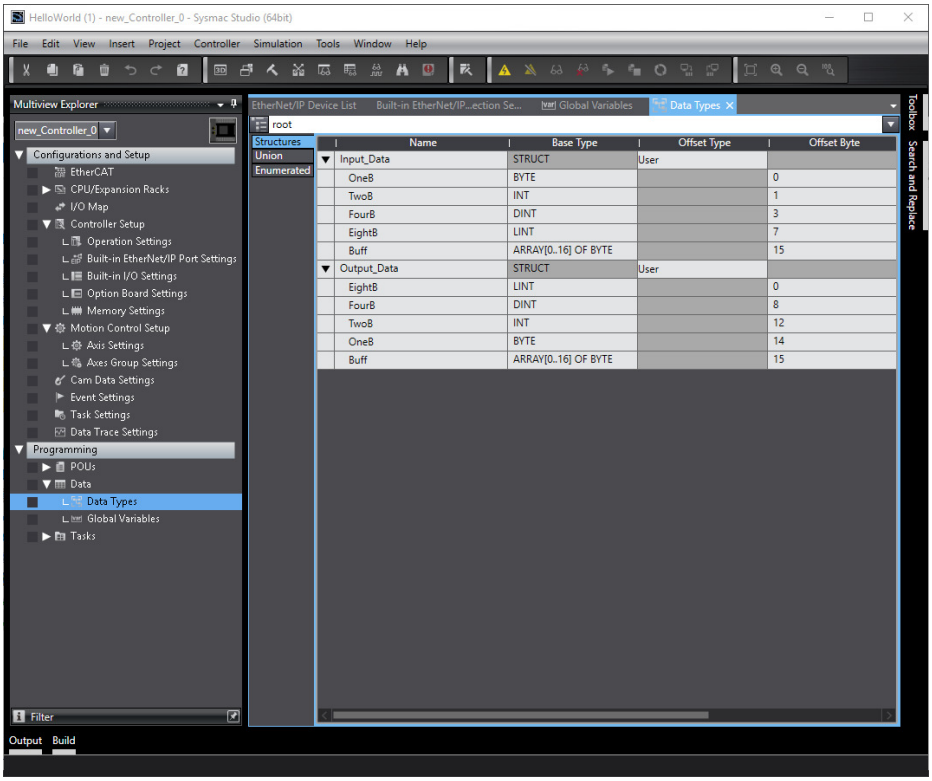
The final step is to generate configuration files for Aurora Vision Studio. You can do this by right-clicking on the device icon, then navigating to "Additional Functions -> Export -> DBM/nxd..", entering your configuration name and clicking "Save". You can now close SYCON.net for this example, **remember to save your project before.**



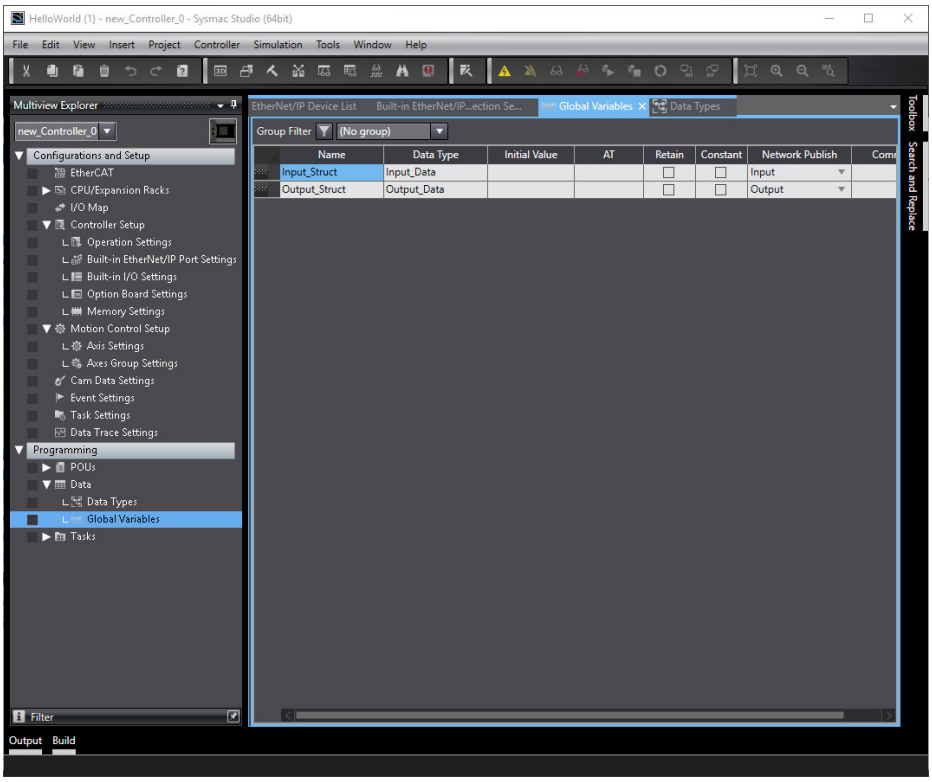
4) Example configuration of EtherNet/IP PLC

Below you will find an example configuration process for EtherNet/IP PLC Omron NXJP2-9024DT in Sysmac Studio software.

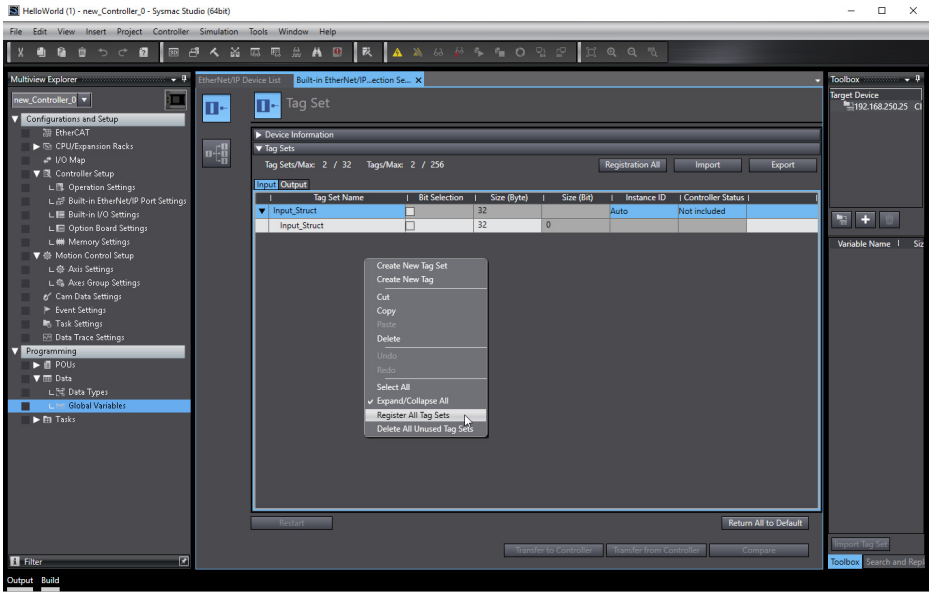
First of all, there were defined two structures Data Types: **Input_Data** and **Output_Data** for further tests of Aurora Vision Studio macrofilters. Information about offsets will be further used in Aurora Vision Studio.



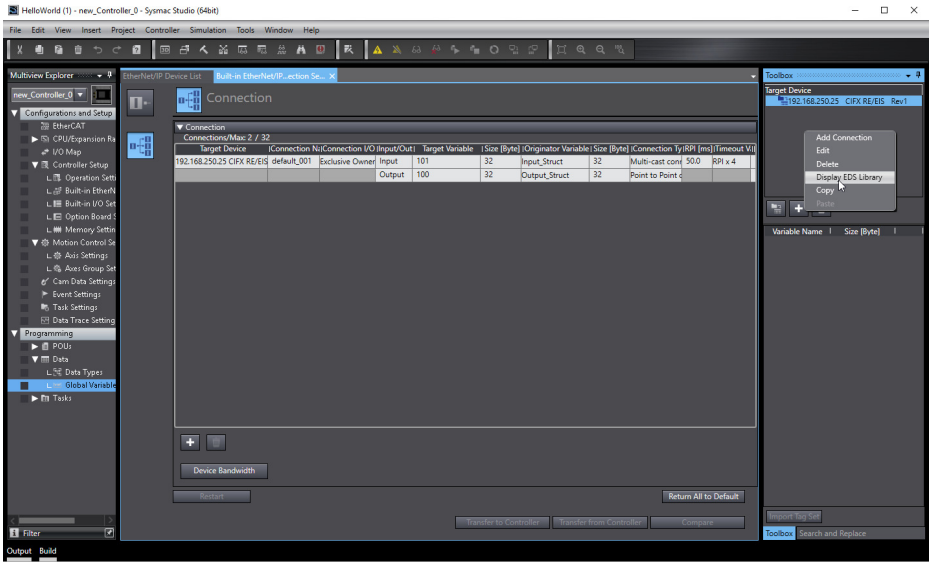
There were created two global variables **Input_Struct** and **Output_Struct** using previously created structures to handle communication with Hilscher card.



Then navigate to (Tools -> EtherNet/IP Connection Setting) and "Register All Tag Sets".

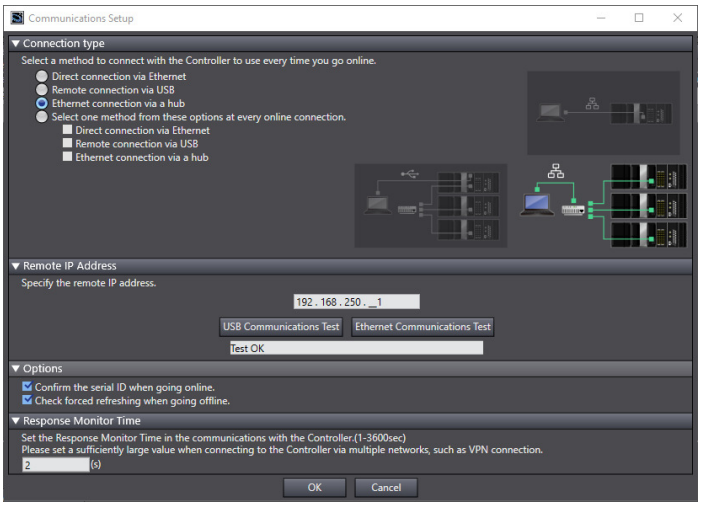


Add support for Hilscher card using EDS Library configuration (path \COMSOL-EIS V2.15.0.1\EDS). Then add "CIFX RE/EIS" and configure Input and Output like below. **It is necessary to match the configuration from SYCON.net.**

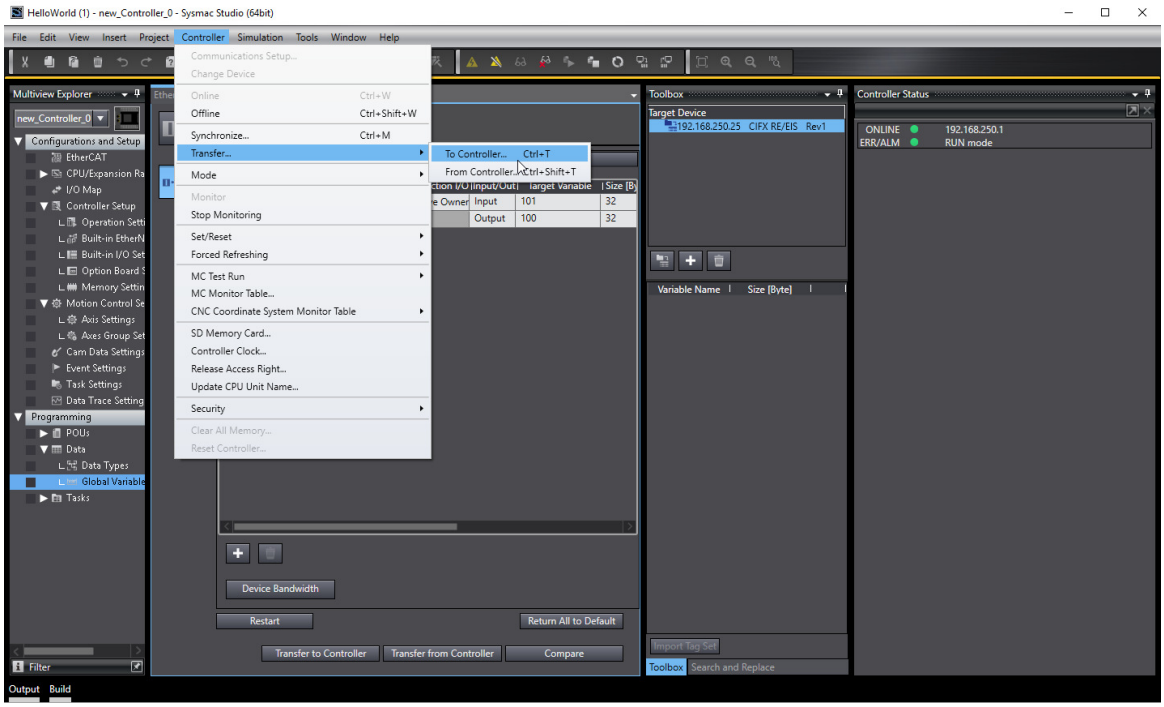


.

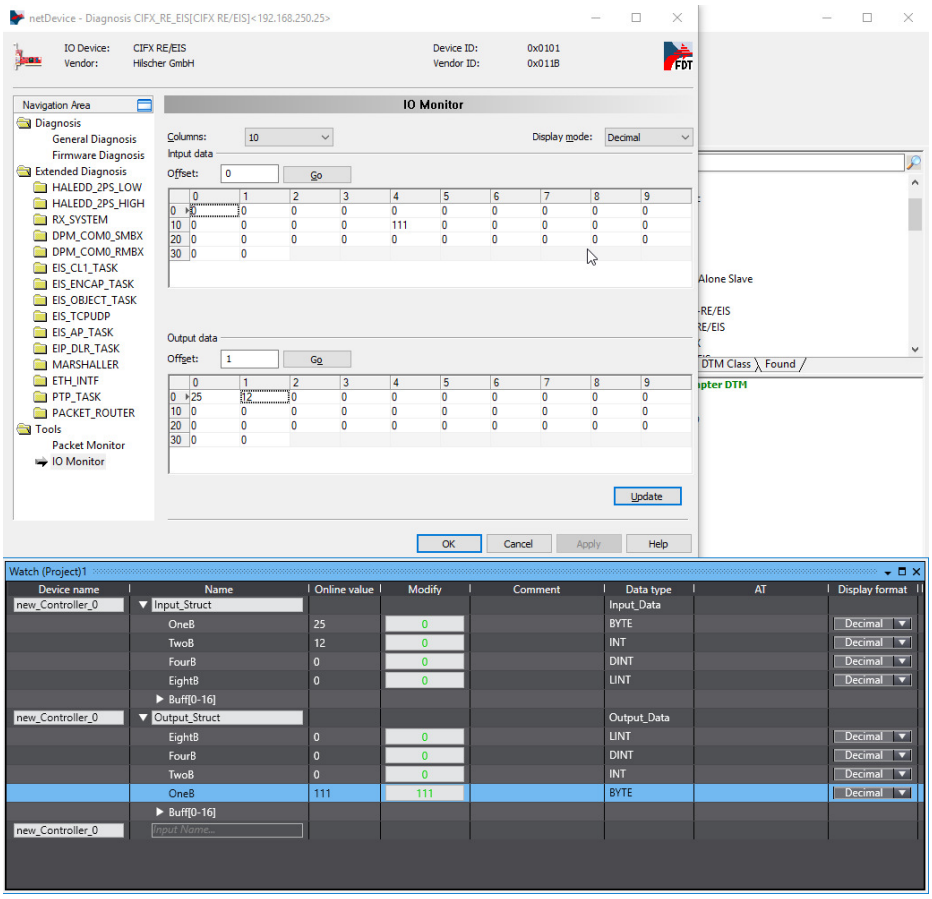
For this moment it is essential to connect with a PLC device, to do this in Sysmac Studio you need to navigate to "Controller -> Communication Setup...". There might be an error in Ethernet Communication Test if your network card is configured as DHCP (**Configuration of the network card** section in this tutorial). If communication with PLC is established you will receive "Test OK".



If communication with PLC is established you can download the configuration to the device.

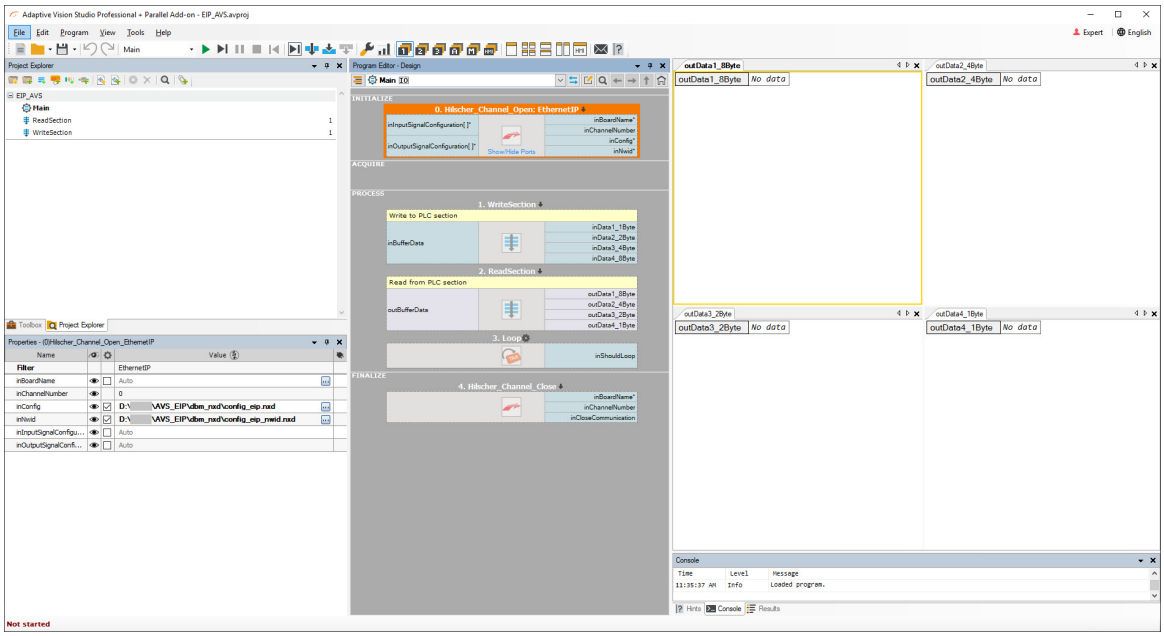


Below you can find a test of input and output memory using a watchtable in Sysmac Studio and IO Monitor in SYCON.net. To turn on IO Monitor you have to right-click on the icon of the card in SYCON.net project and click "Connect". Then double click this icon and navigate to "Tools -> IO Monitor". By clicking "Update" you can send a changed frame to the PLC.

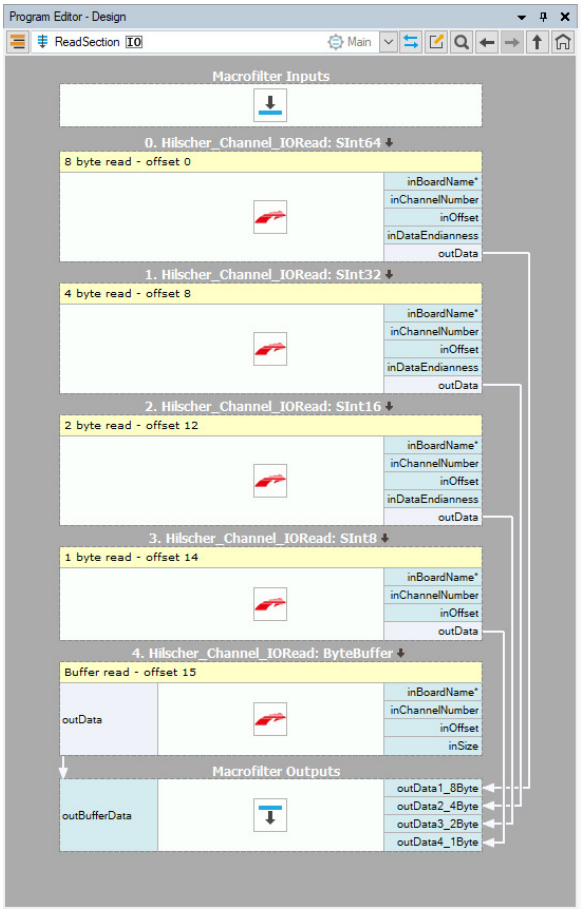


5) Example configuration in Aurora Vision Studio

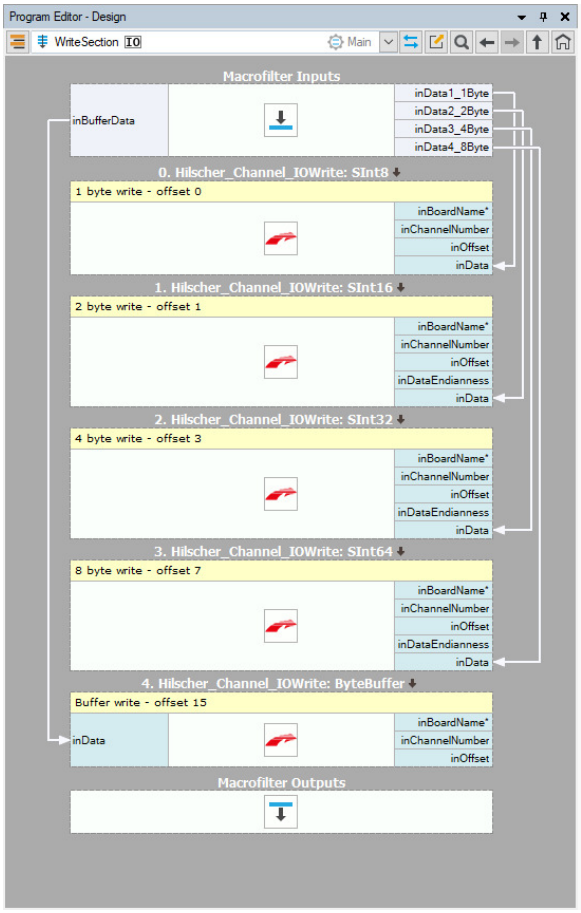
To use EtherNetIP filters in Aurora Vision Studio first you need to attach configuration files from SYCON.net to Hilscher_Channel_Open_EthernetIP filter in INITIALIZE section. The configuration files generated in the previous step are now required in inConfig (xxx.nxd) and inNwid (xxx_nwid.nxd) properties of that filter. Below you can find two Step macrofilters responsible for writing data to PLC and receiving data from it. To have cyclic communication you have to place Loop macrofilter at the end of PROCESS section. In FINALIZE section place Hilscher_Channel_Close macrofilter.



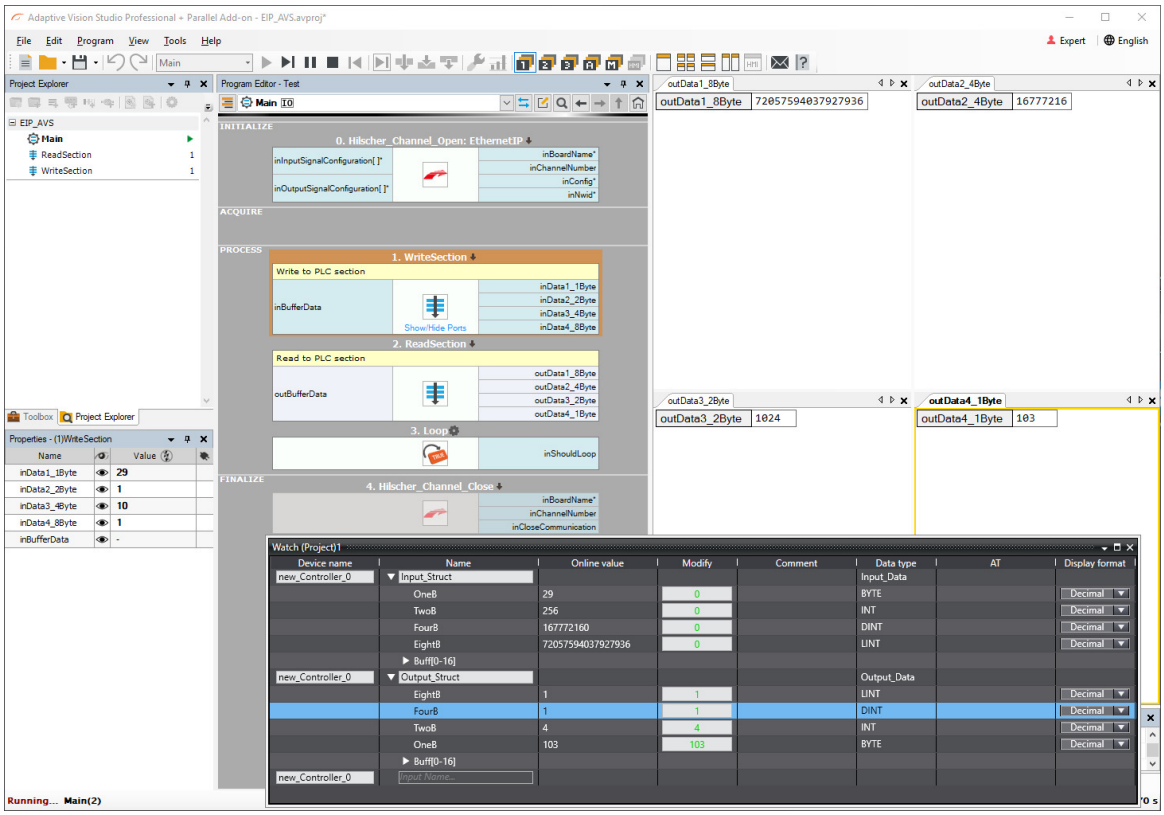
In step macrofilter ReadSection you can find for example Hilscher_Channel_IORead_SInt8 filter which writes 8 bytes of signed data to the predefined memory area. Using different offsets for each macrofilter enables access to different variables created in PLC (**Example configuration of Ethernet/IP PLC** section of this tutorial).



WriteSection step macrofilter has Hilscher_Channel_IOWrite filters (for instance Hilscher_Channel_IOWrite_SInt8) filters with adequate offsets and data types used to match PLC data variables configuration.



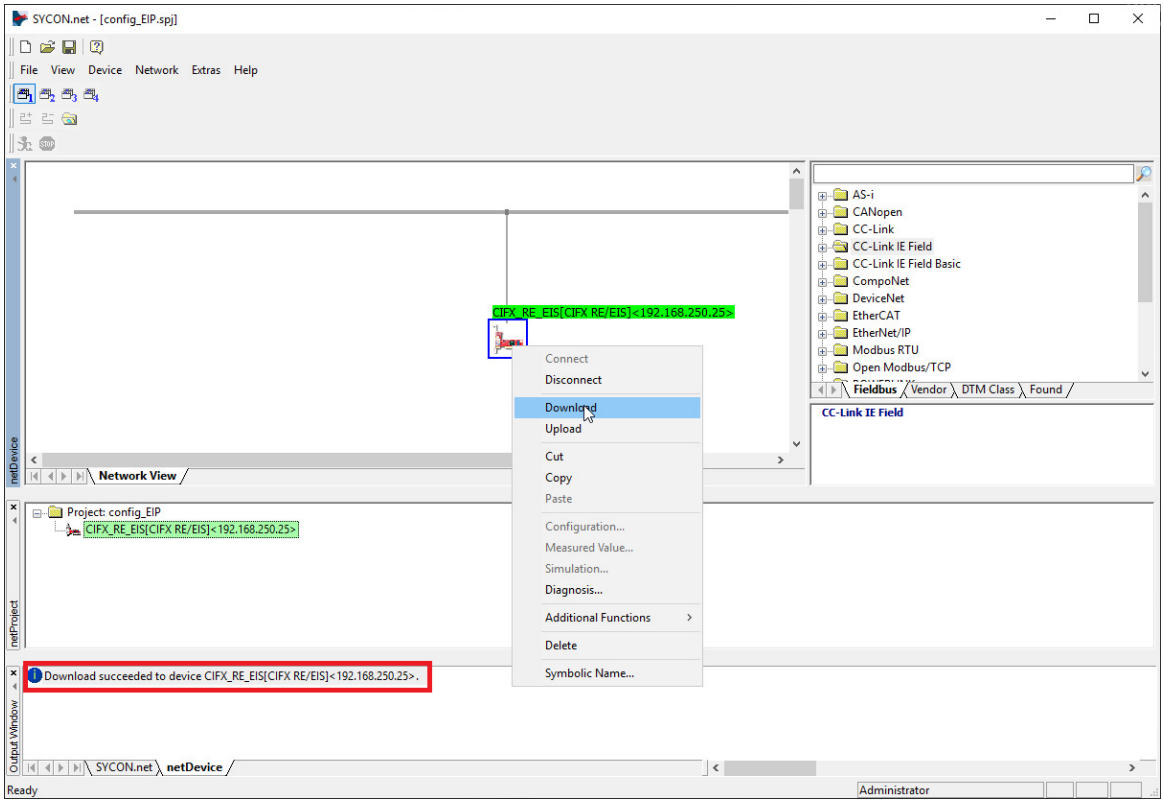
Below is presented reading and writing to PLC with Aurora Vision Studio and Sysmac Studio watchtable. Decimal values of variables are dependent on the data type used.



Troubleshooting

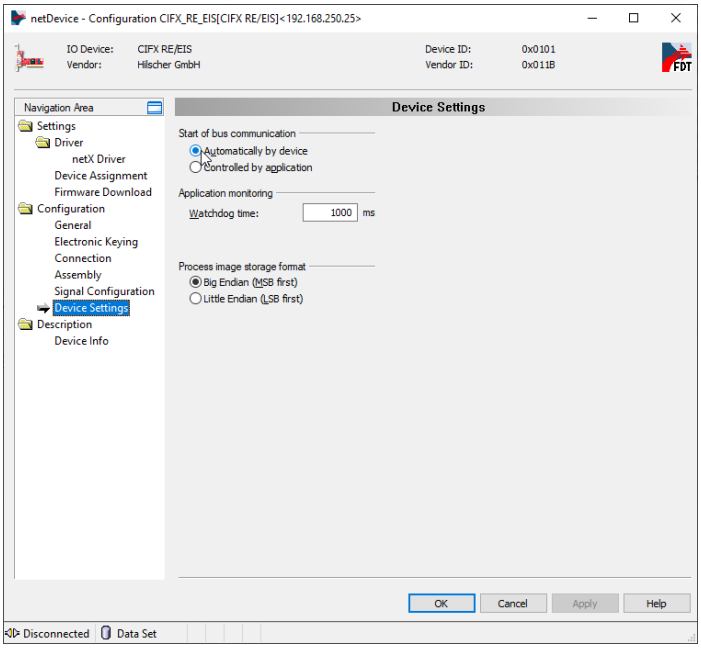
1. Set static IP to Hilscher card. Go to the section **Configuration using SYCON.net** of this tutorial.

2. Make sure that the current program setting is loaded to the Hilscher card. If not please use SYCON.net application to connect and download settings to the devices as shown in the picture below.

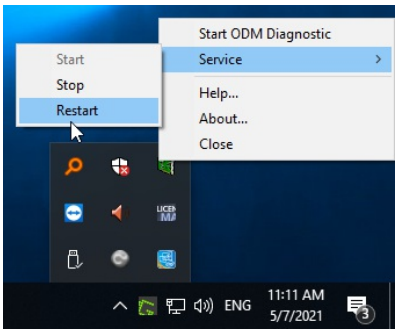


3. Check in your device in Device Assignment. This step was described in **Configuration using SYCON.net** section of this tutorial.

4. If your master device has a problem with connection with the Hilscher card use the following settings in the SYCON.net application. In order to do this right-click on the card icon and select Configuration...



5. After changes it may be necessary to restart ODMV3 service.



6. If none of the above advice has helped, please restart your computer.

Interfacing Hilscher card (Profinet) to Aurora Vision Studio

Purpose and equipment

This document explains how to configure Profinet PLC with Aurora Vision Studio using Hilscher Profinet card.

Required equipment:

- Supported Hilscher Profinet PC card
- Aurora Vision Studio 4.12 or later

Hardware connection

All devices must work in a common LAN network, hereinafter called the shared network. In most cases they are connected to each other through a network switch.

The devices shall be connected as follows:

- Master device (e.g. PLC) to the shared network
- Hilscher card to PC
- Hilscher card to the shared network
- PC's network card to shared network

Before proceeding to the further point, make sure that all devices are powered up.

Configuring Hilscher Devices

Be sure to have the newest version of the following drivers:

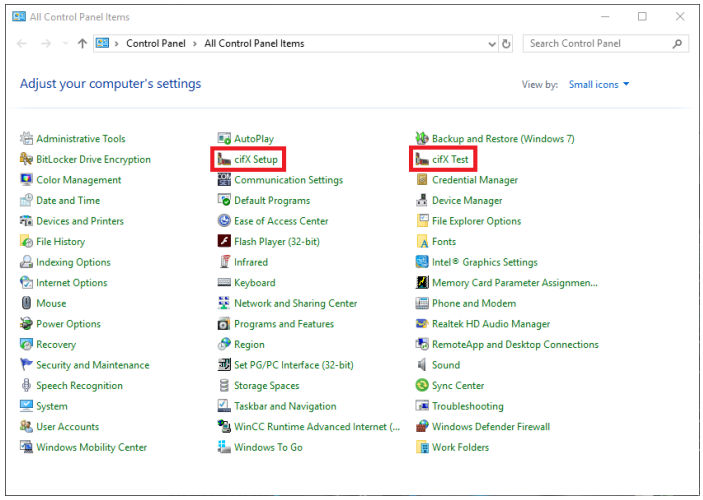
- Windows driver (recommended 1.5.0.0): download link (for CIFX 50E-RE download V2.x version)
- SYCON.net for configuring slots, generating configuration files: download link
- List of the Profinet firmware (recommended 4.x version, tested 4.7.0.1) download link

A computer restart is highly recommended after driver installations.

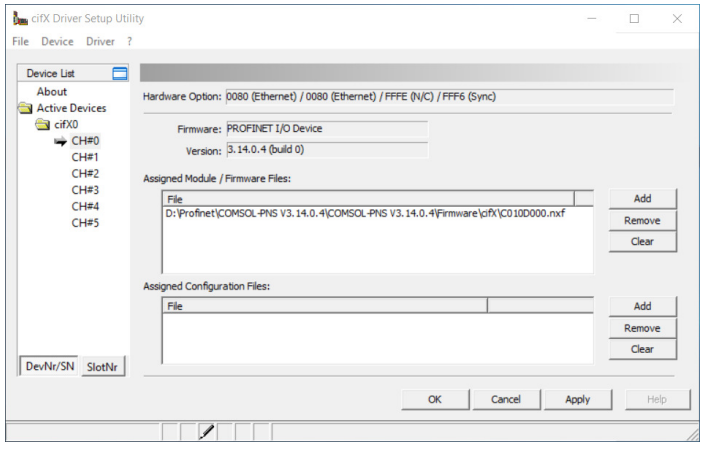
Make sure you have the recommended driver installed, newest SYCON.net, and firmware prepared. Profinet requirements are written down in Hilscher_Channel_Open_Profinet documentation.

1) Configuration using cifX Driver Setup Software

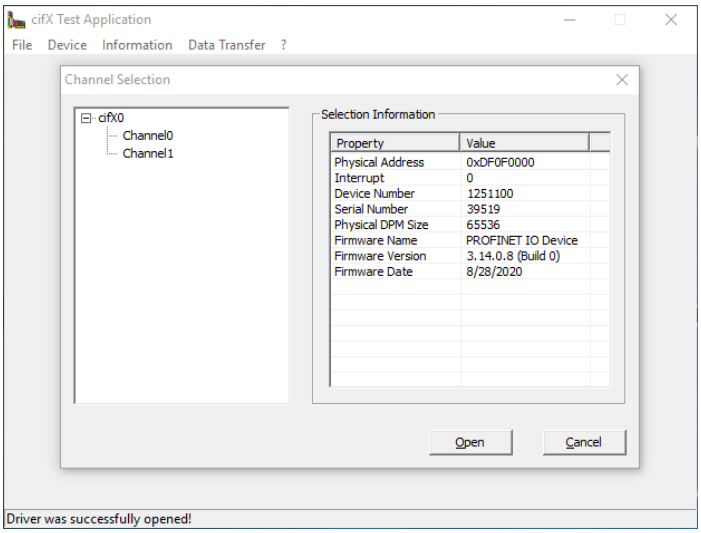
The easiest way to install the firmware to the cifX Driver is using the cifX Driver Setup Software. You can find tools for this software for instance using Control Panel menu as in the picture below.



Using cifX Setup application select the channel you would like to configure (usually channel - CH#0) and remove preexisting firmware by clicking "Clear". Then click "Add" to add a new firmware file and navigate to the downloaded Profinet Adapter firmware (path \\COMSOL-PNS V3.14.0.4\\Firmware\\cifX). Then click "Apply" and finish configuration with "OK".

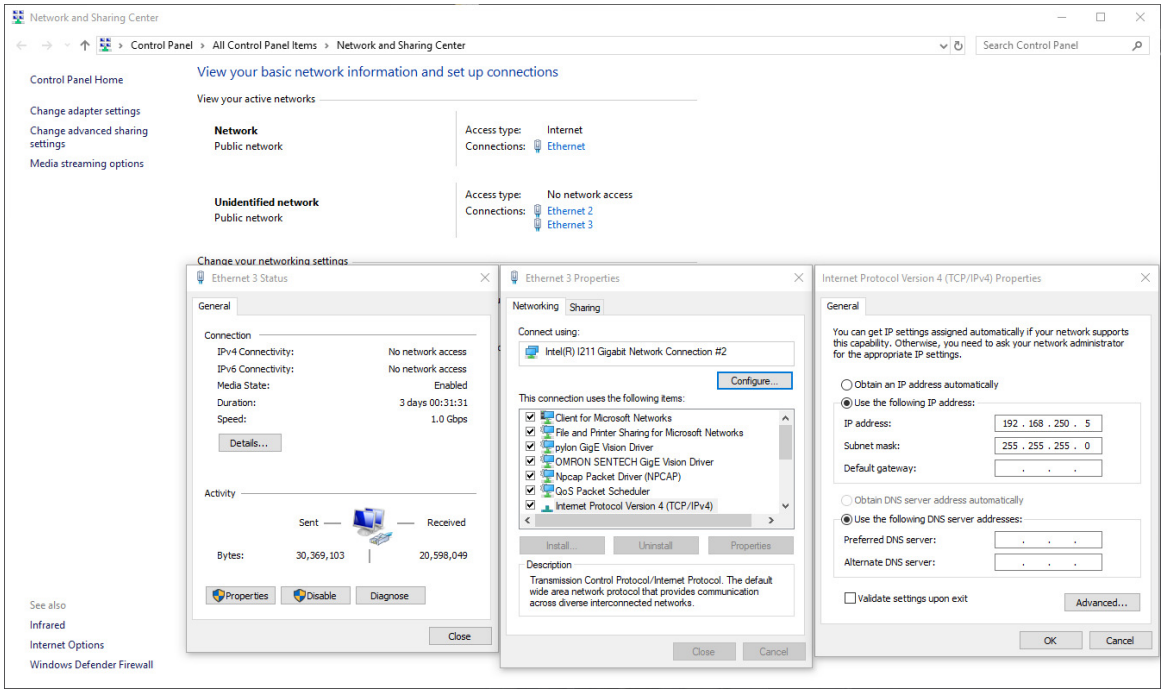


To be sure that firmware is installed properly you can open cifX Test software and then navigate to Device -> Open to achieve the result like below.



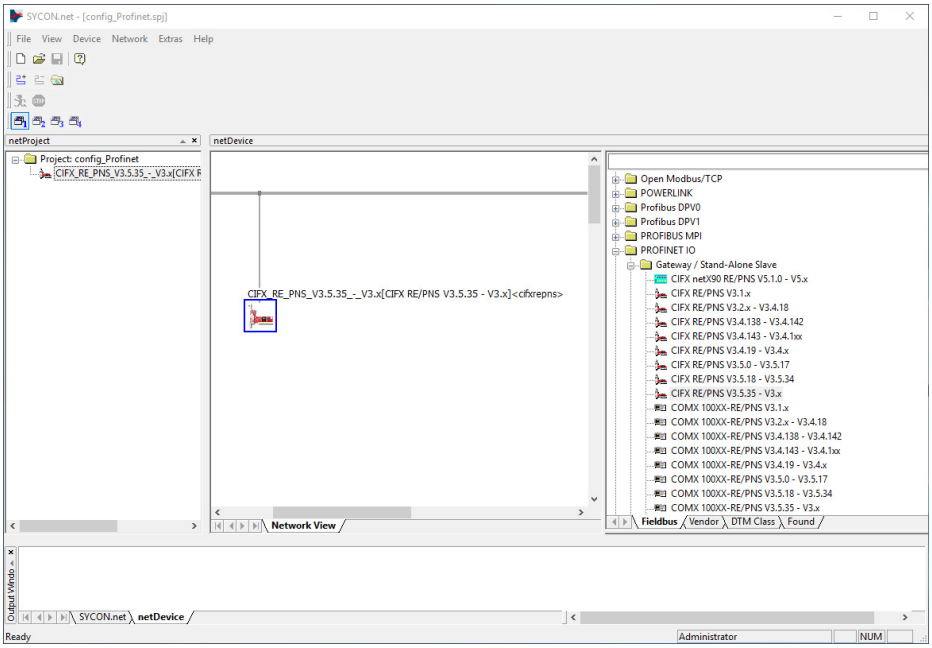
2) Configuration of the network card

It might be necessary to change the IP of the network card from dynamic to static. In this tutorial the following settings are used.

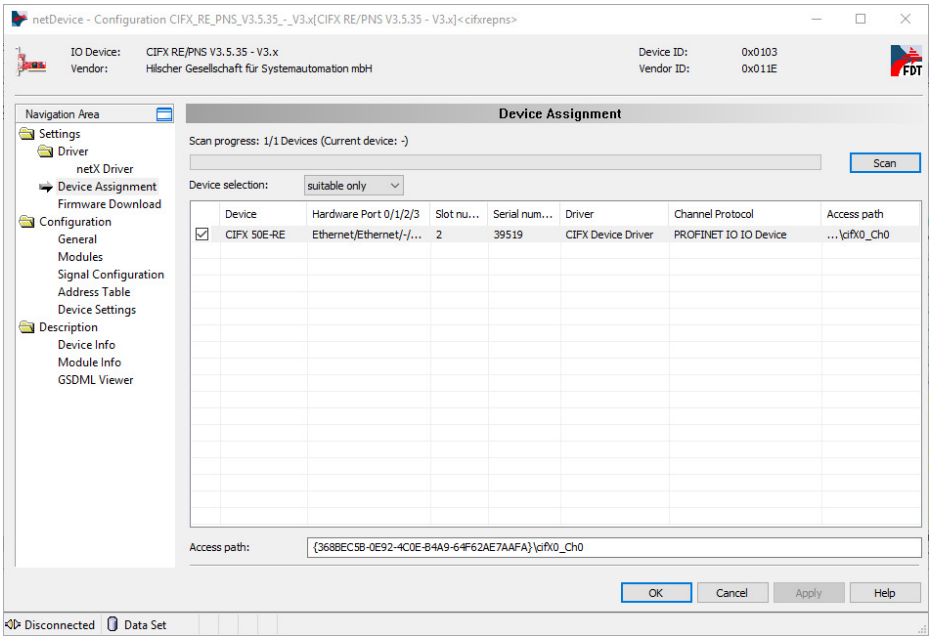


3) Configuration using SYCON.net

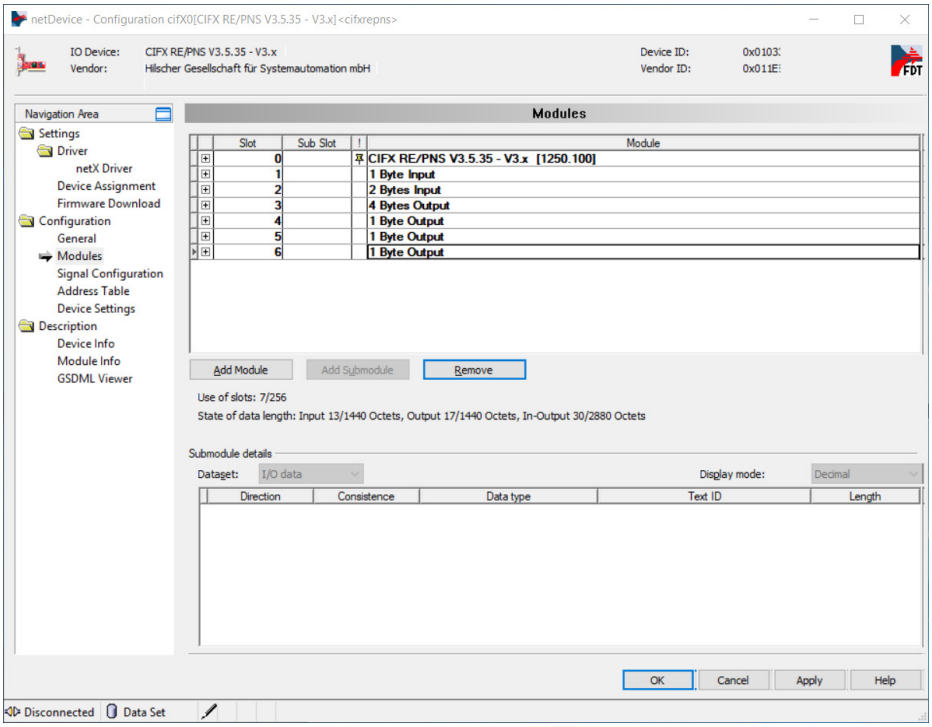
Configuration files are generated in SYCON.net software. Firstly, pick a proper device and drag it to the gray bus on the "Network View". This device you can find navigating to "PROFINET IO -> Gateway / Stand – Alone Slave -> CIFX RE/PNS V3.5.35. – V3.x".



To open the configuration card double click on dropped icon and in the popped-up menu navigate to "Settings/Device Assignment". Then scan for devices by clicking "Scan", mark found one it with a tick like on the image below and then click "Apply".

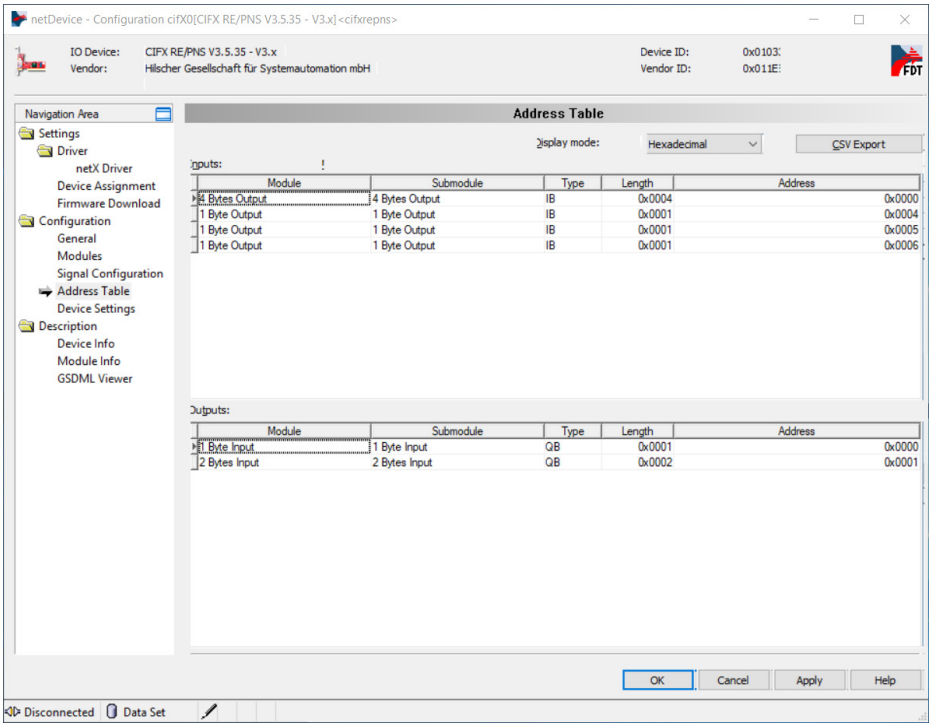


To open card configuration, double click on its icon and select the "Configuration/Modules" category in the Navigation Area on the left side of the dialog. In the main window you can add individual modules. Remember that in Profinet, **Slot configuration must match the configuration from your master device, otherwise the connection will not work**. For boolean indicators we recommend "1 Byte Output" or "1 Byte Input".

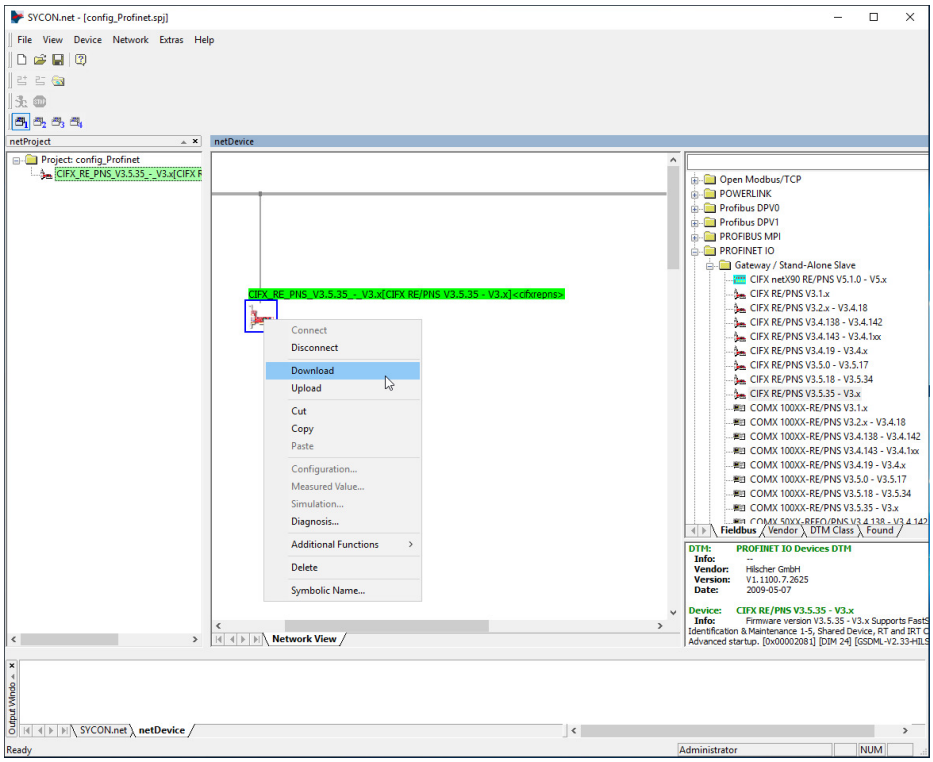


Confirm all changes by clicking "Apply" and then click "OK".

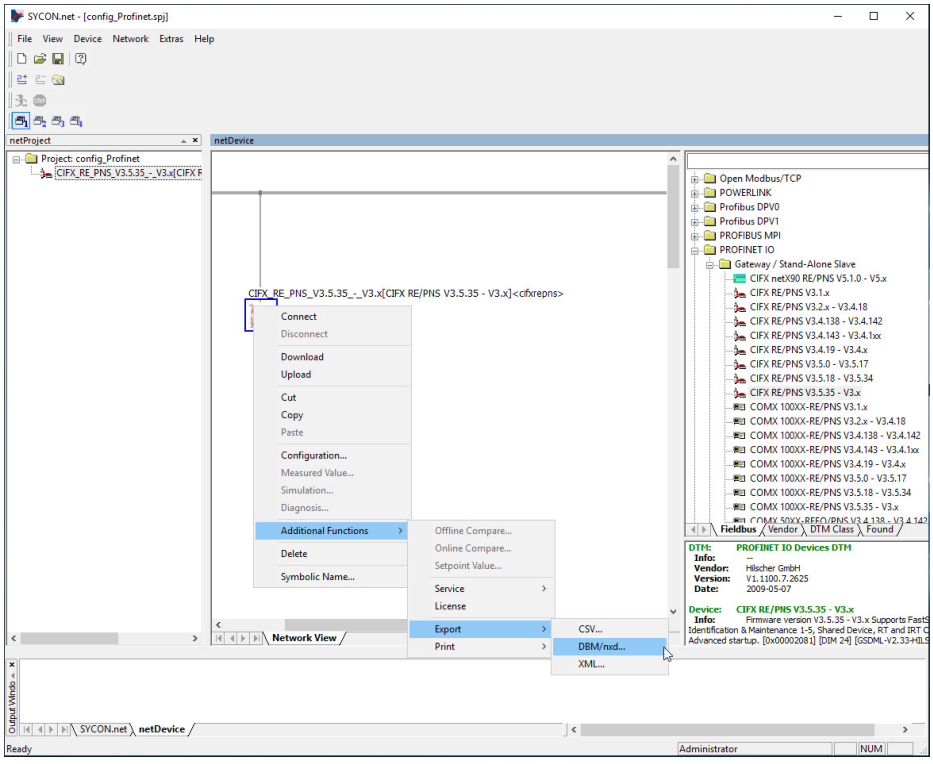
You can see the final address table (addresses on the Hilscher device where input or output data will be available) in the "Configuration/Address table" category. If you plan to use *IORead* and *IOWrite* filters, for example Hilscher_Channel_IORead_SInt8, note the addresses. You can switch the display mode to decimal, as Aurora Vision Studio accepts decimal addressing, not hexadecimal. Aurora Vision Studio implementation of Profinet checks whether the address and data size match. In the sample configuration below, writing a byte to address 0x002 would not work, because that module address starts at 0x001 and spans 2 bytes. Moreover, Aurora Vision Studio prohibits writing with a *SlotWrite* filter to input areas and reading with a *SlotRead* filter from output areas. Click "OK" when you have finished the proper configuration.



Then you have to download the configuration to the device like on the screen below.



The final step is to generate configuration files for Aurora Vision Studio. You can do this by right-clicking on the device icon, then navigating to "Additional Functions -> Export -> DBM/nxd..", entering your configuration name and clicking "Save". You can now close SYCON.net for this example, **remember to save your project before, so that it is easier to add new slots.**

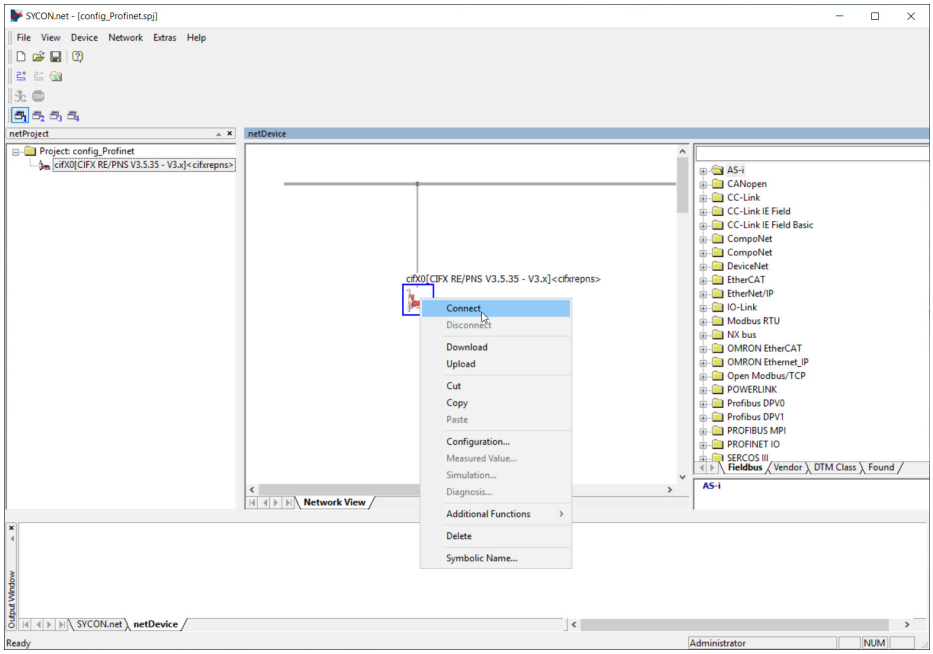


4) Working with Profinet in Aurora Vision Studio

The full list of filters for communication over Hilscher devices is available under this **link**

.

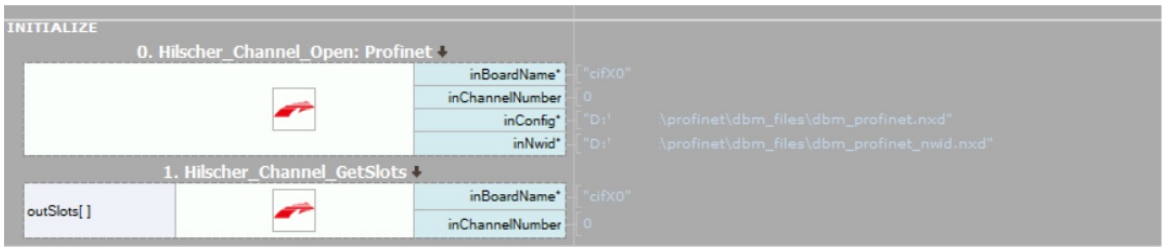
Before proceeding to the further steps, please check Hilscher and AVS connection using Hilscher_Driver_GetBoardInformation filter. If all previous steps are done correctly you will be able to get device board information including BoardName and ID. Please note that in this step the connection with the Hilscher card via Sycon.net is required, so you need to run the software, right-click on the card icon and click *Connect*.



As described in previous chapter the Sycon.net application is not required for a proper operation. On the contrary, it is highly recommended to keep it closed and use the generated configuration files to guarantee high stability of the connection.

After following the instruction from chapter 3, you have a xxx.nxd and xxx_nwid.nxd files. To send data over Profinet, follow the below steps:

1. Configure connection in AVS with Hilscher_Channel_Open_Profinet filter. Drag it from Program I/O category in the Toolbox and drop it in the Program Editor.
2. In **inBoardName** input enter value acquired from Hilscher_Driver_GetBoardInformation filter. In **inConfig** and **inNwid** inputs enter paths to generated files from Sycon.net.

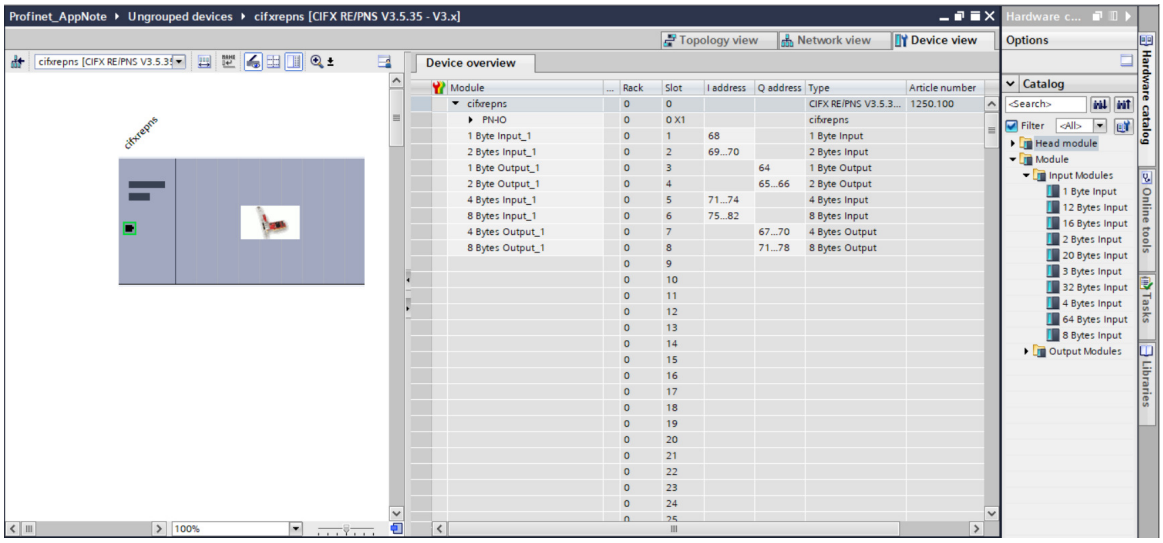


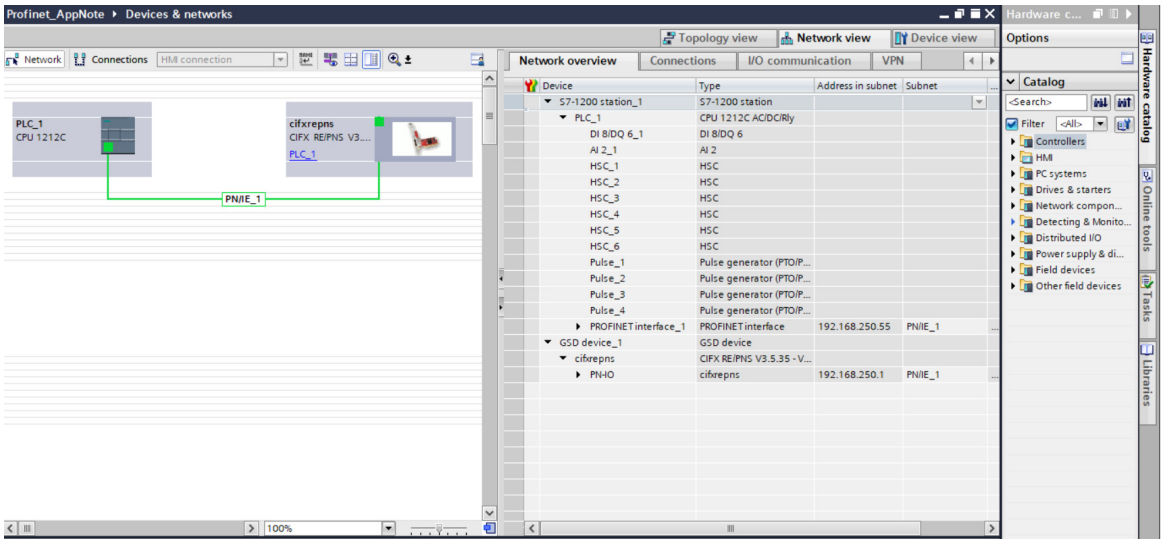
3. To prevent from I/O errors make sure that you open connection in the initialization step with Hilscher_Channel_Open_Profinet and close it at the end of the program using Hilscher_Channel_Close filter.
4. For IO, we recommend *SlotRead* and *SlotWrite* filters, as they are more convenient. For example, the Hilscher_Channel_SlotWrite_SInt8 filter writes 8 bytes of signed data to the selected slot. Slot numbers match those in the "Configuration/Modules" category of card configuration in SYCON.net program.

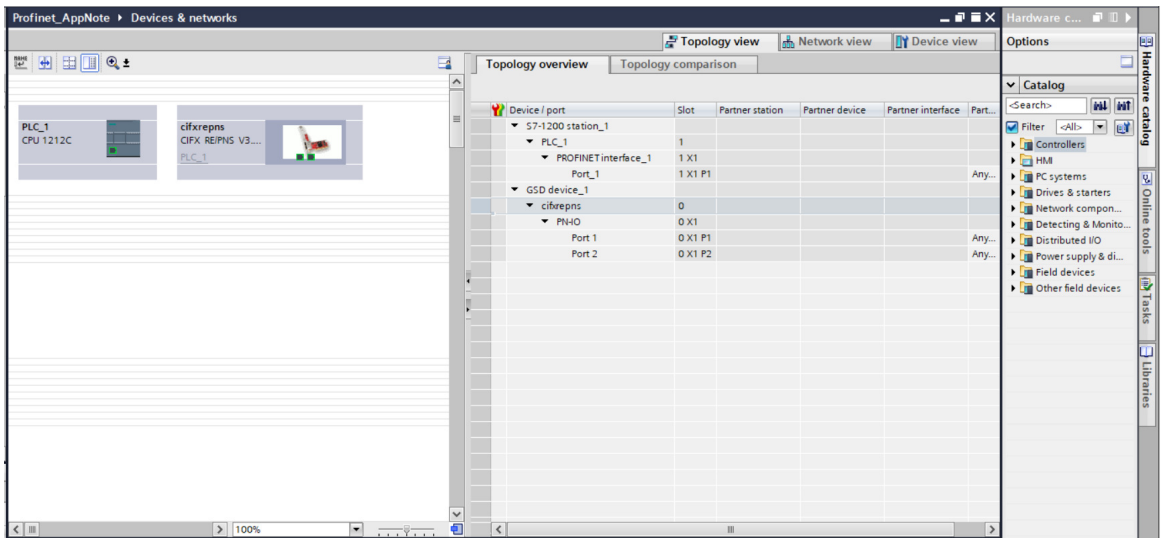
5) Example Configuration

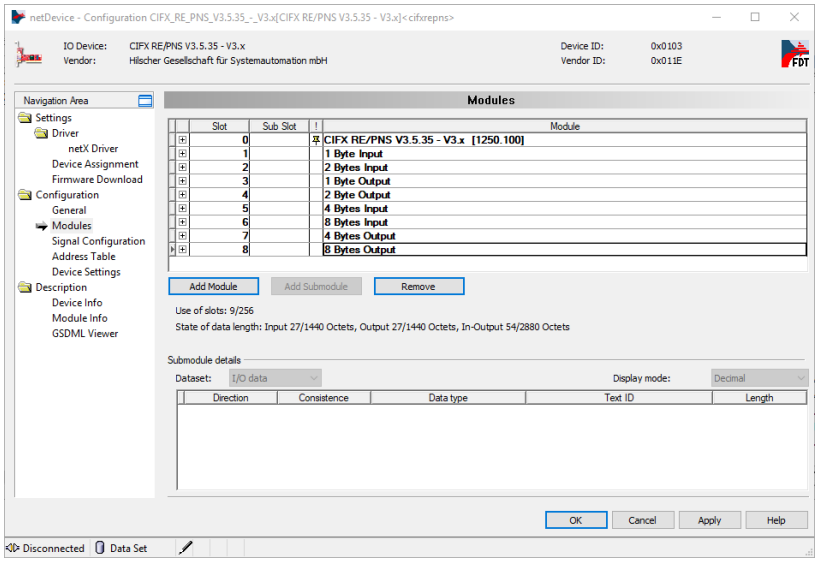
Below you will find a sample configuration of Siemens TIA Portal and Siemens S7-1200 PLC, Hilscher Sycon and Aurora Vision Studio.

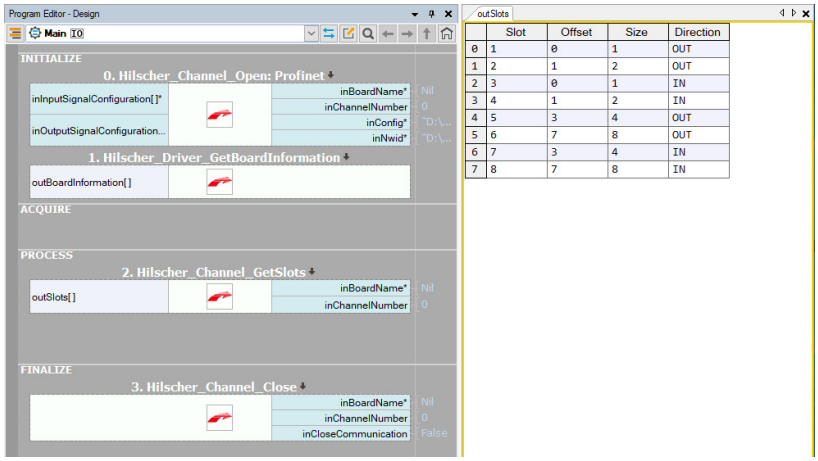
Before establishing communication between a Siemens S7-1200 PLC and the Hilscher card, you must first upload the device's GSDML file into TIA Portal. You can usually get the GSDML file via SYCON.net, or if the full driver setup package is installed, the XML-based GSDML file (typically named in a format like GSDML-V2.x-HILSCHER-CIFX RE PNS-XXXXXXXX.xml) is located in the driver installation directory under the Firmware_&_EDS or DeviceDescription folders.


Profinet_AppNote ▸ Ungrouped devices ▸ cifxrepns [CIFX RE/PNS V3.5.35 - V3.x]
Topology view
Network view
Device view
Device overview
Module
Rack
Slot
I address
Q address
Type
Article number
cifxrepns
PN-IO
1 Byte Input_1
2 Bytes Input_1
1 Byte Output_1
2 Byte Output_1
4 Bytes Input_1
8 Bytes Input_1
4 Bytes Output_1
8 Bytes Output_1
Hardware c...
Options
Catalog
Head module
Module
Input Modules
Output Modules
Hardware catalog
Online tools
Tasks
Libraries


Profinet_AppNote ▸ Devices & networks
Topology view
Network view
Device view
Network
Connections
HMI connection
Network overview
Connections
I/O communication
VPN
PLC_1
CPU 1212C
cifxrepns
CIFX RE/PNS V3....
PLC_1
PN/IE_1
Device
Type
Address in subnet
Subnet
S7-1200 station_1
PLC_1
PROFINET interface_1
GSD device_1
cifxrepns
PN-IO
192.168.250.55
192.168.250.1
Hardware c...
Options
Catalog
Controllers
HMI
PC systems
Drives & starters
Network compon...
Detecting & Monito...
Distributed I/O
Power supply & di...
Field devices
Other field devices
Hardware catalog
Online tools
Tasks
Libraries


Profinet_AppNote ▸ Devices & networks
Topology view
Network view
Device view
Topology overview
Topology comparison
PLC_1
CPU 1212C
cifxrepns
CIFX RE/PNS V3....
PLC_1
Device / port
Slot
Partner station
Partner device
Partner interface
S7-1200 station_1
PLC_1
PROFINET interface_1
Port_1
GSD device_1
cifxrepns
PN-IO
Port 1
Port 2
Hardware c...
Options
Catalog
Controllers
HMI
PC systems
Drives & starters
Network compon...
Detecting & Monito...
Distributed I/O
Power supply & di...
Field devices
Other field devices
Hardware catalog
Online tools
Tasks
Libraries



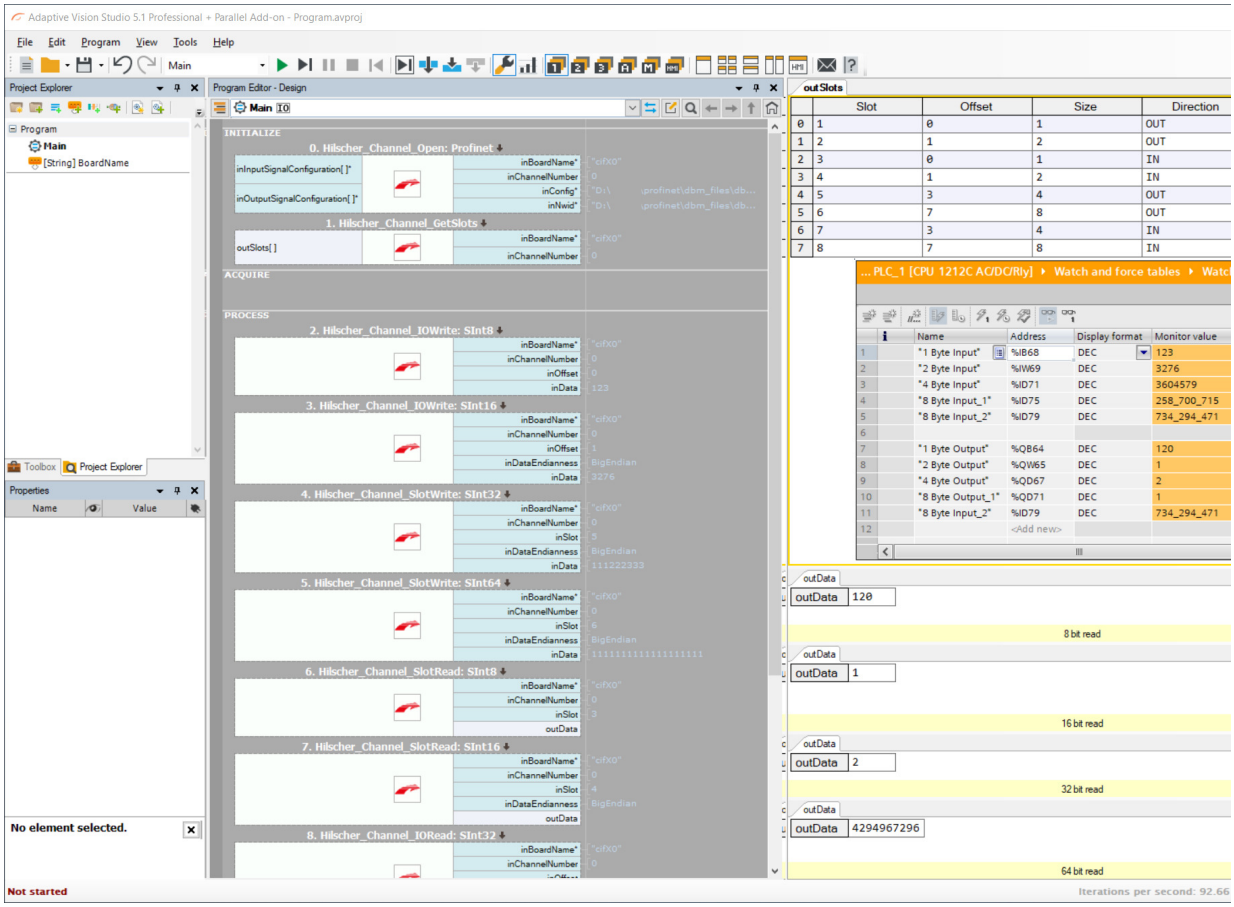


As you can see on the screenshots above the slots are common for each software. The only difference is that Siemens and Hilscher inputs corresponds to outputs from Aurora Vision Studio and vice-versa – Siemens output is Aurora Vision Studio input. It should be considered in the following manner: data outgoing from PLC (output) is incoming to Aurora Vision Studio (input) while data incoming to PLC (input) is outgoing from Aurora Vision Studio (output).

The summarization of data type and directions you will find in the table below:

TIA Portal		Sycon		Aurora Vision Studio	
Size	Type	Size	Type	Size	Type
1 Byte	Input	1 Byte	Input	SInt8	Output
2 Bytes	Input	2 Bytes	Input	SInt16	Output
4 Bytes	Input	4 Bytes	Input	SInt32	Output
8 Bytes	Input	8 Bytes	Input	SInt64	Output
1 Byte	Output	1 Byte	Output	SInt8	Input
2 Bytes	Output	2 Bytes	Output	SInt16	Input
4 Bytes	Output	4 Bytes	Output	SInt32	Input
8 Bytes	Output	8 Bytes	Output	SInt64	Input

For the slot configuration described in this paragraph the Aurora Vision Studio program will look as follows:

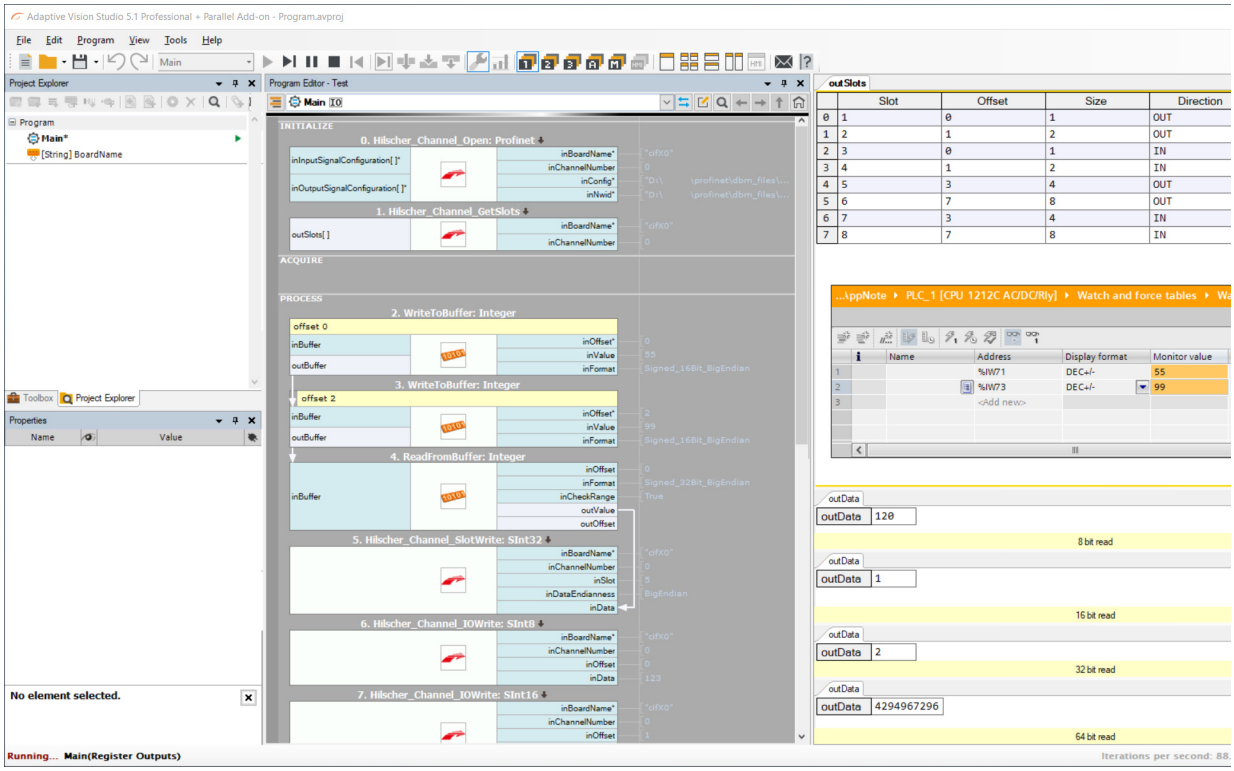


You can use both Hilscher_Channel_IOWrite / Hilscher_Channel_IORead or Hilscher_Channel_SlotWrite / Hilscher_Channel_SlotRead to send and receive data. The only difference is in addressing the accessible data:

- *IORead* and *IOWrite* filters use the **offset** value
- *SlotRead* and *SlotWrite* filters use the **slot** numbers

Both of the information you will find in output **outSlots** from Hilscher_Channel_GetSlots filter.

Sometimes it is necessary to combine few integer values into one slot of data. In order to accomplish this task you can use binary buffers in Aurora Vision Studio. Let's assume that someone would like to write two 2-bytes integers into one 4 bytes slot (*SInt32* in Aurora Vision Studio). In this situation the program should look as follows:



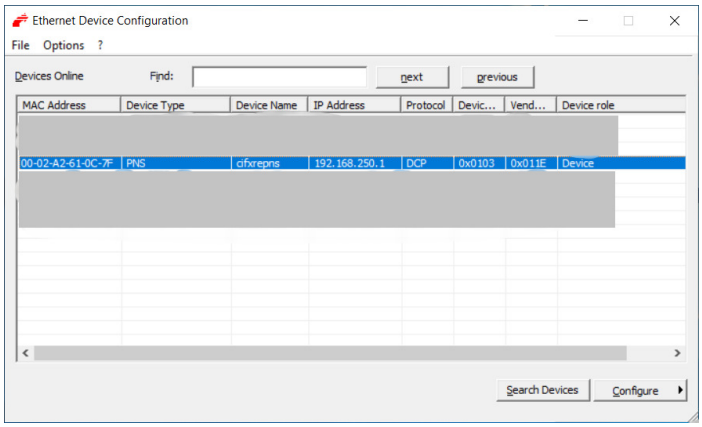
First we use two WriteIntegerToBuffer filters with specified format: **Signed_16Bit_BigEndian** connected in cascade to write two different values into one buffer (in this case 55 and 99). Also we specify the offset where the second value should be written. Later we extract the combined value as one Integer with filter ReadIntegerFromBuffer. For 8 bytes integers should be used ReadLongFromBuffer. Here we

also specify the format of data (**Signed_32Bit_BigEndian**). Later we send the obtained number with Hilscher_Channel_SlotWrite_SInt32. Between *ReadFromBuffer* and *SlotWrite* the data is meaningless but later in TIA Portal when we access it as two separate 2-bytes values we get the send values.

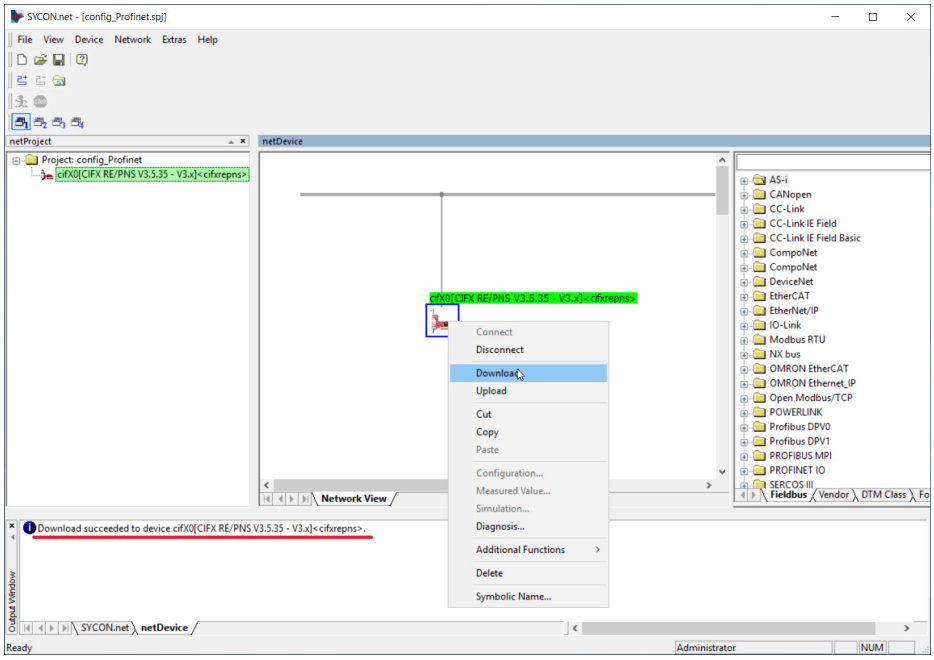
Troubleshooting

If it is still not possible to use AVS with Hilscher card first, make sure that you done correctly all previous steps and then use following advices.

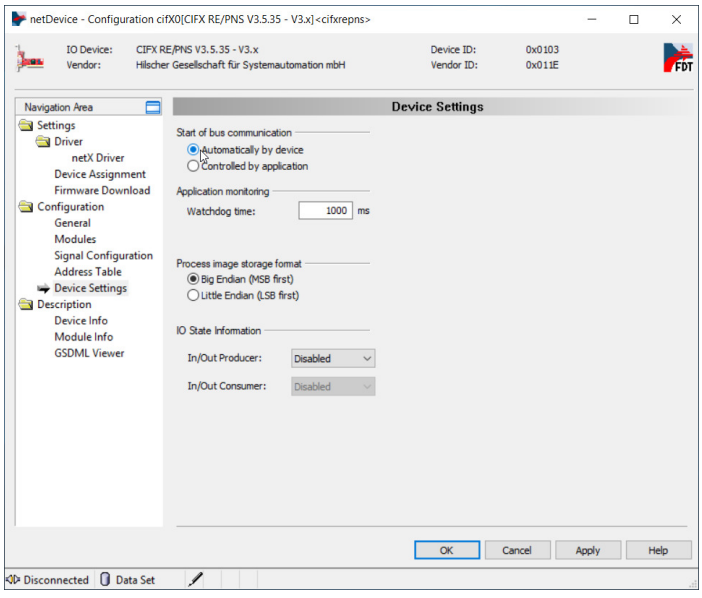
1. Use Ethernet Device Configuration software to set static IP address to Hilscher card.



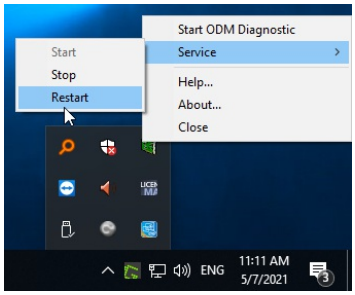
2. Make sure that the current program setting is loaded to the Hilscher card. If not, please use the SYCON.net application to connect and download settings to the devices, as shown in the picture below.



3. If your master device has problem with connection with Hilscher card use following settings using Sycon.net application. In order to do this right-click on the card icon and select *Configuration...*



4. After changes it may be necessary to restart ODMV3 service.



5. If none of the above advice has helped, please restart your computer.

Interfacing Profinet gateway to Aurora Vision Studio

Purpose and equipment

This document explains how to configure Profinet PLC with Aurora Vision Studio using Modbus-TCP – PROFINET gateway.

Required equipment:

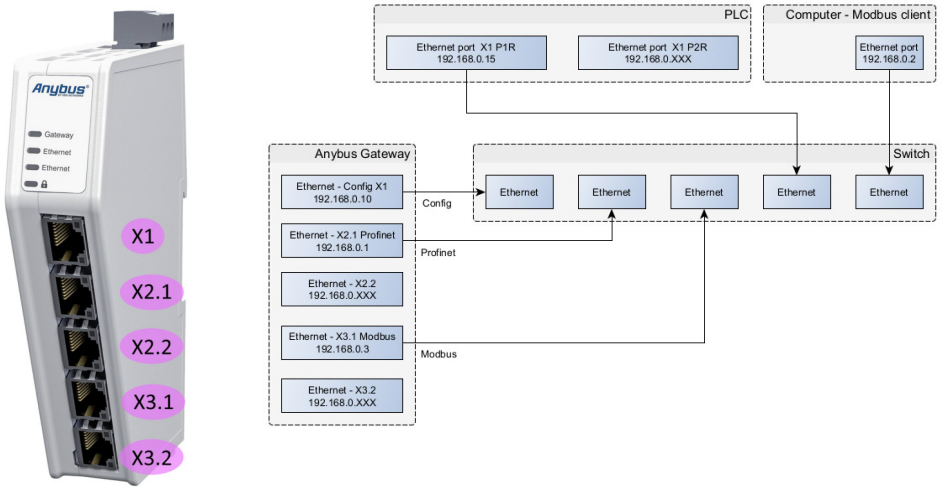
- Supported Modbus-TCP – PROFINET gateway (used Anybus ABC4090);
- Siemens S7-1200 PLC;
- Ethernet switch with at least 5 In/Outs might be useful – simplifies configuration;
- TIA Portal (version v15 used);
- Aurora Vision Studio 5.0 or later (version 5.2 used).

This application note demonstrates configuration steps for a specific gateway model, but it is not obligatory to use this one. If you have a different gateway than the one described in this application note, please refer to manuals provided by its manufacturer.

Hardware connection

In application notes there is configured one network which uses Ethernet switch. Devices connected to this network have addresses 192.168.0.XXX, where XXX stands for number 1-254 unique for every device/port connected.

If you use Anybus ABC4090 gateway, it has 5 configurable In/Outs. Please find below description of these ports and connecting scheme:



- X1 – Configuration port. By default this address is set to 192.168.0.10 and this address is used in this application notes. If you want to change it, use HMS IPconfig tool in the way described in ABC4090 configuration manual. Connected to switch.
- X2.1 – configurable port from network 1. Configured as Profinet. Connected to switch.
- X2.2 – configurable port from network 1. Configured as Profinet.
- X3.1 – configurable port from network 2. Configured as Modbus-TCP. Connected to switch.
- X3.2 – configurable port from network 2. Configured as Modbus-TCP.

Before proceeding to the further point, make sure that all devices are powered up.

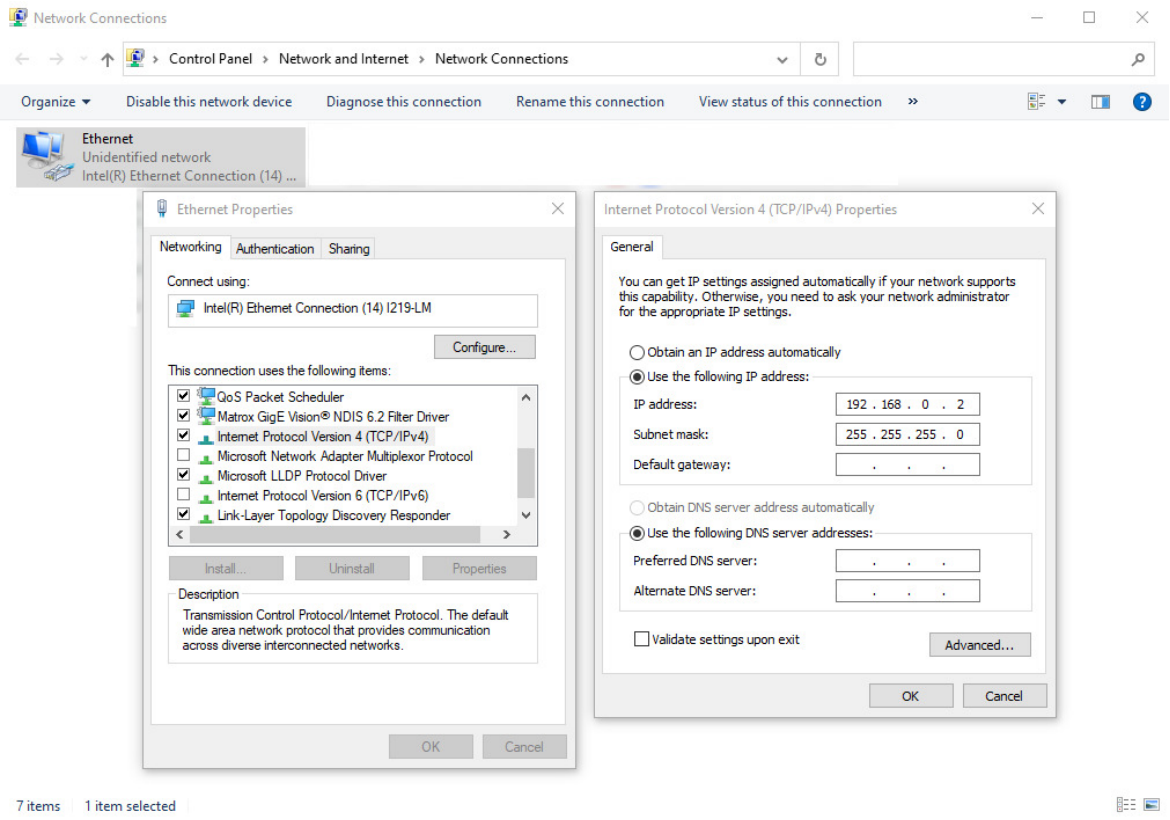
Configuring Anybus gateway in web configurator

1. By default Anybus ABC4090 is configured as EtherNet/IP – PROFINET gateway. To have Modbus-TCP – PROFINET gateway there is need to upload new firmware to the device.

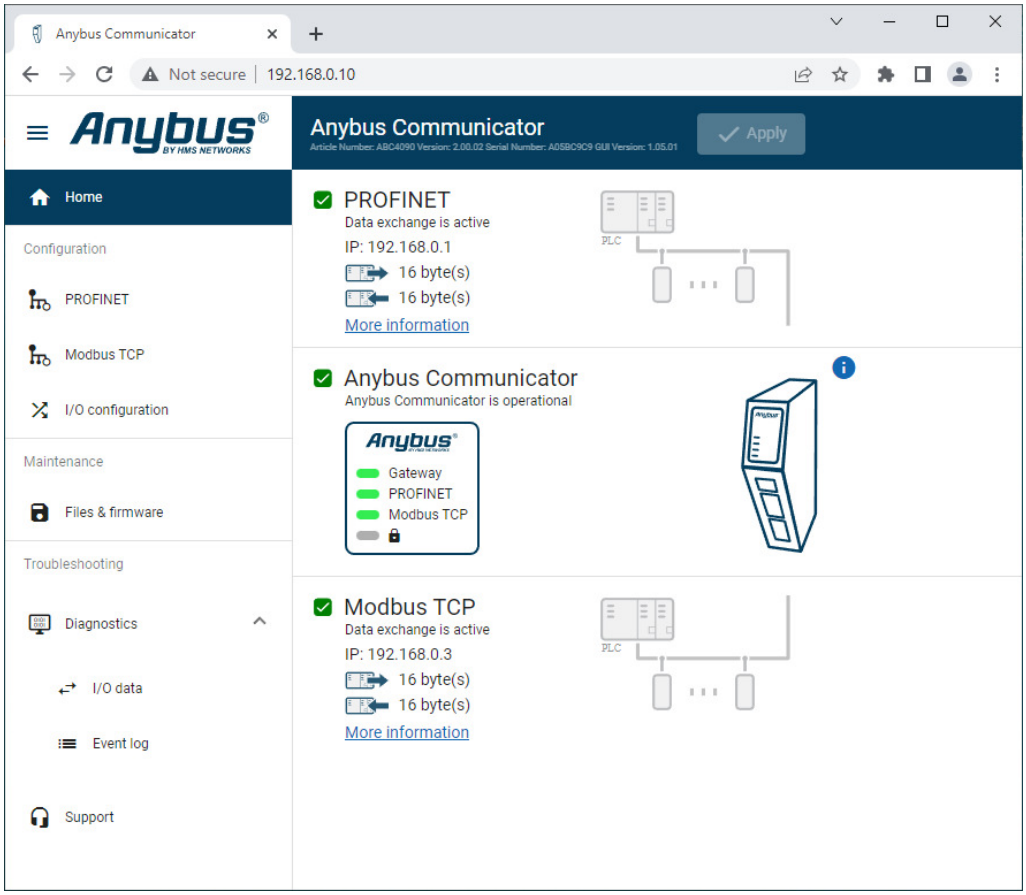
   Please download appropriate drivers before further reading from **firmware download** section:

File	Version	Size	Download
FIRMWARE			
Firmware Update EtherCAT - EtherNet/IP	1.00.01	3.88 MB	Download
Firmware Update EtherCAT - EtherNet/IP	1.01.01	3.90 MB	Download
Firmware Update EtherCAT - Modbus-TCP	1.00.01	3.84 MB	Download
Firmware Update EtherCAT - Modbus-TCP	1.01.01	3.86 MB	Download
Firmware Update EtherCAT - PROFINET	1.00.01	3.96 MB	Download
Firmware Update EtherCAT - PROFINET	1.01.01	4.04 MB	Download
Firmware Update EtherNet/IP - PROFINET	2.00.01	4.02 MB	Download
Firmware Update Modbus-TCP - PROFINET	2.00.02	3.99 MB	Download
Firmware Update Modbus-TCP - EtherNet/IP	2.01.01	3.90 MB	Download

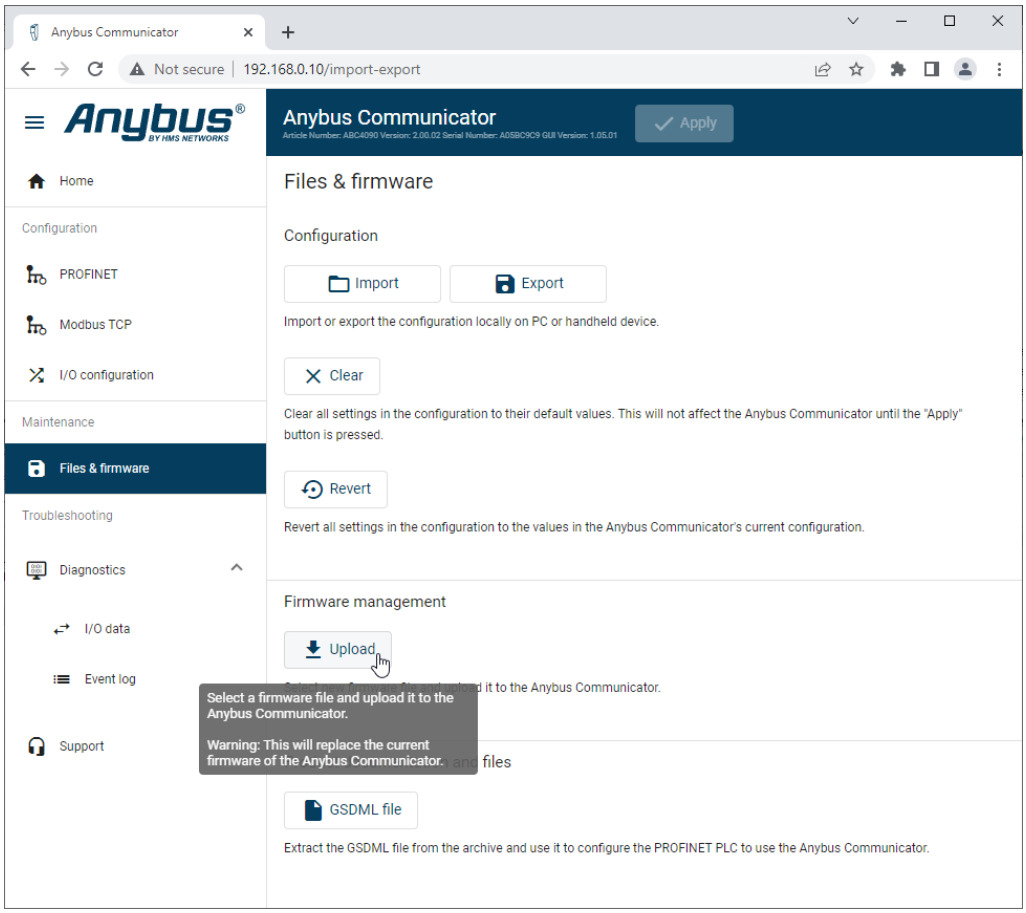
2. Default configuration IP address should be set to 192.168.0.10. The PC must work in the 192.168.0.0 network, so the Ethernet-AnybusConfig IP was changed accordingly:



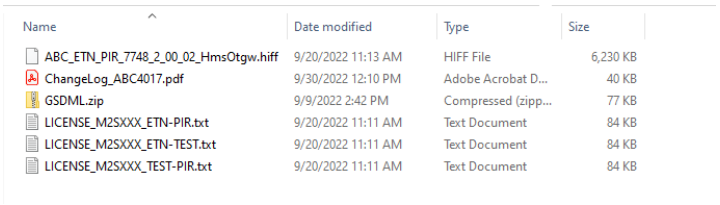
If you change the network card's IP as shown above and connect it to the CONFIG port of the gateway, you should be able to access the configuration menu by typing the 192.168.0.10 IP address in a web browser:



3. In order to change gateway's firmware navigate to **Maintenance -> Files & firmware** tab:

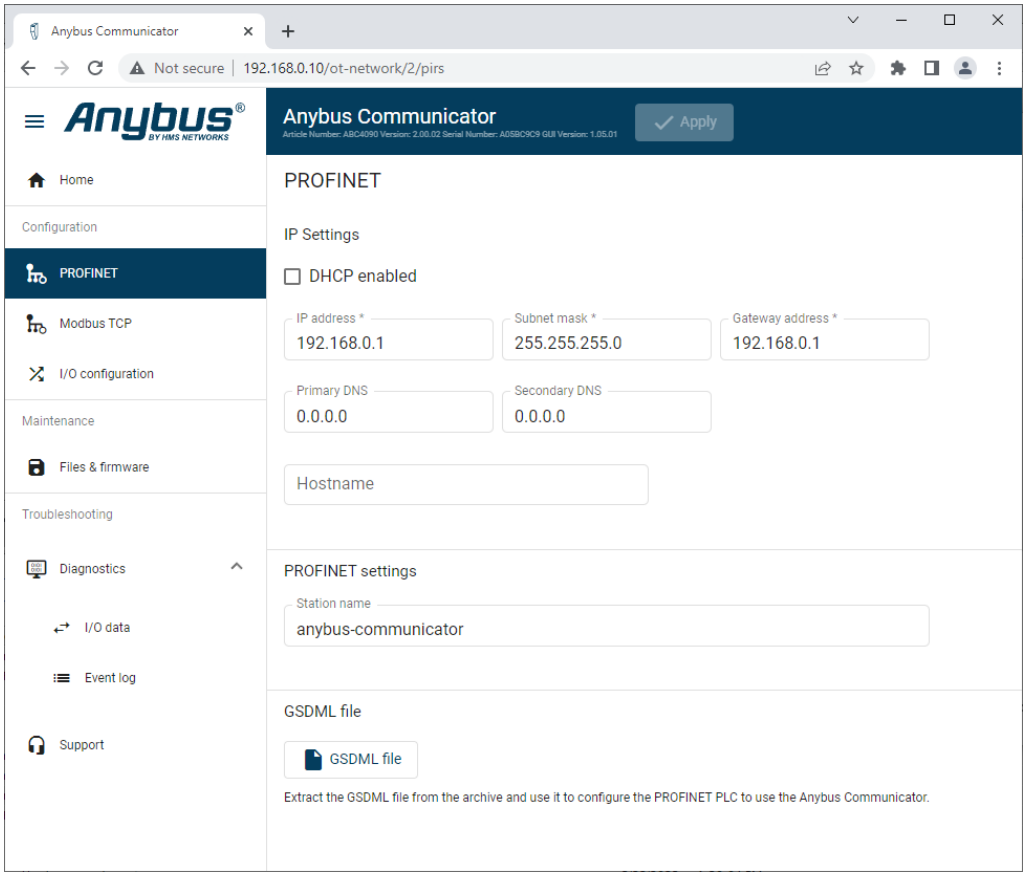


Click **Upload** and navigate to download firmware directory. It consists of GSDML file for later (hardware definition for Profinet PLC) and firmware file:

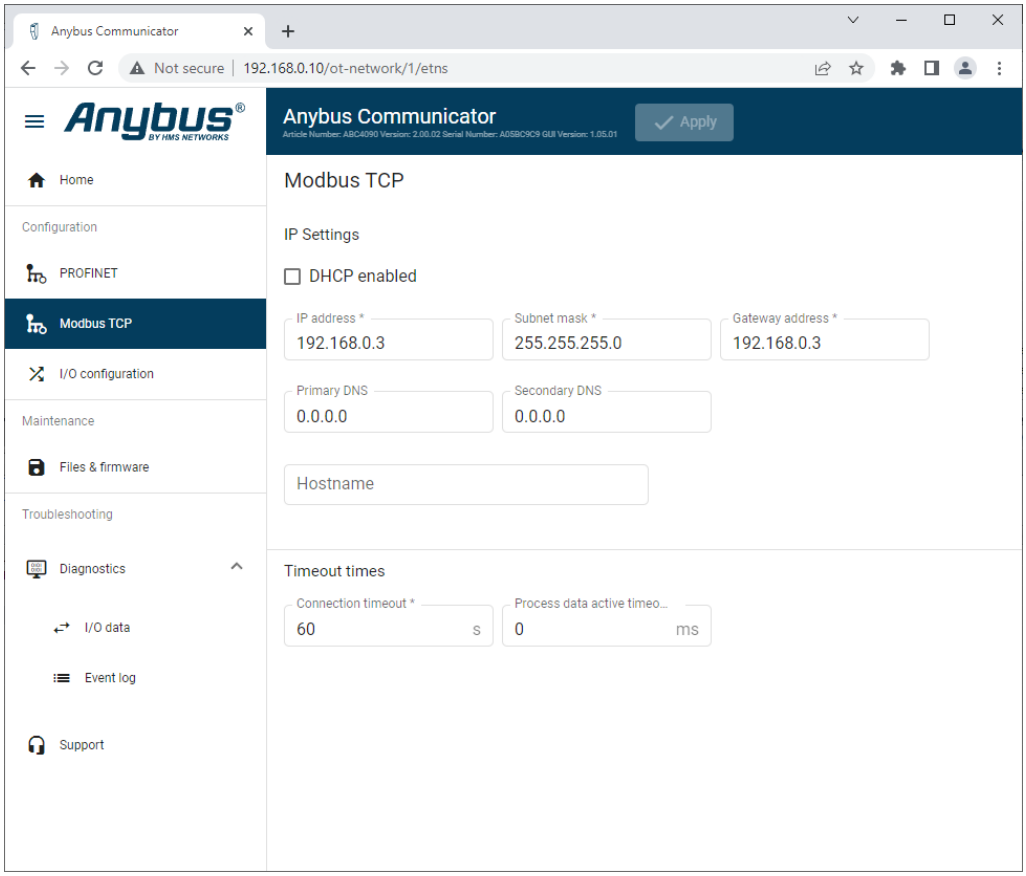


Name	Date modified	Type	Size
ABC_ETN_PIR_7748_2_00_02_HmsOtgw.hiff	9/20/2022 11:13 AM	HIFF File	6,230 KB
ChangeLog_ABC4017.pdf	9/30/2022 12:10 PM	Adobe Acrobat D...	40 KB
GSDML.zip	9/9/2022 2:42 PM	Compressed (zipp...	77 KB
LICENSE_M2SXXX_ETN-PIR.txt	9/20/2022 11:11 AM	Text Document	84 KB
LICENSE_M2SXXX_ETN-TEST.txt	9/20/2022 11:11 AM	Text Document	84 KB
LICENSE_M2SXXX_TEST-PIR.txt	9/20/2022 11:11 AM	Text Document	84 KB

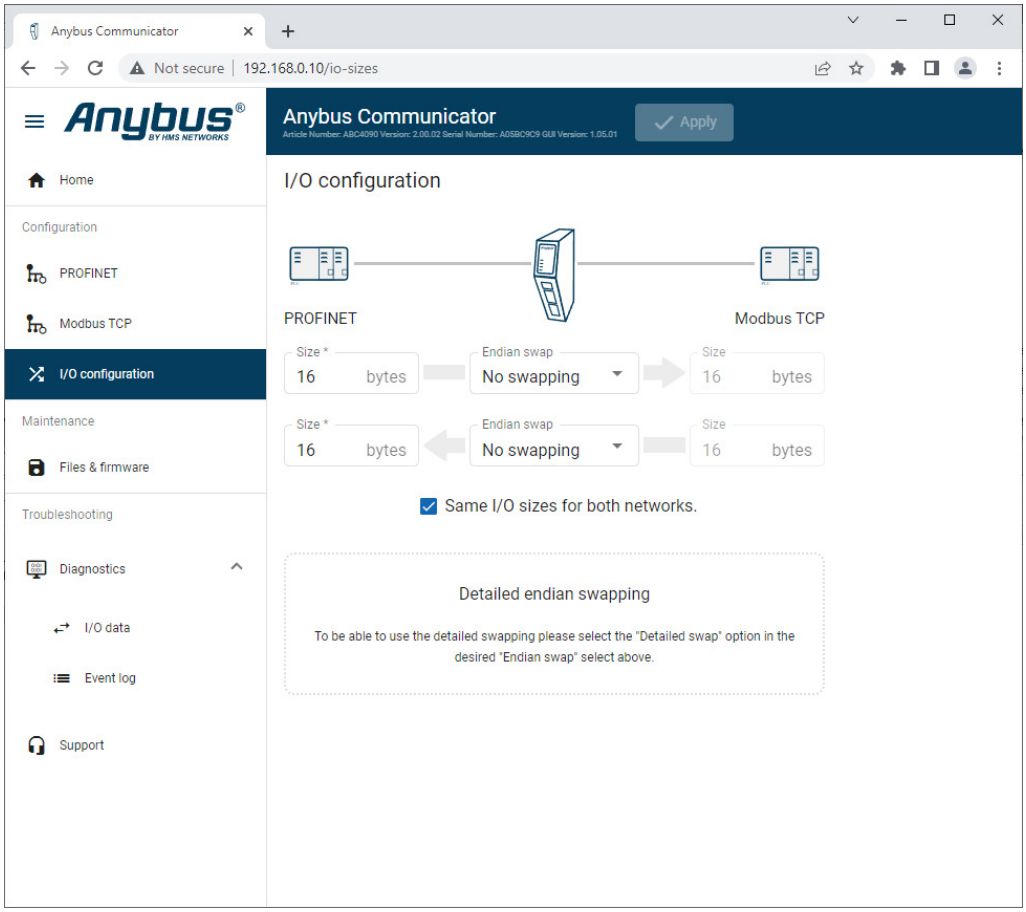
4. Devices linked should have an IP address which belongs to 192.168.0.0 network. Go to **Configuration -> PROFINET** tab, change settings like on the screen attached below and click **Apply**:



5. Devices linked should have an IP address which belongs to 192.168.0.0 network. Go to **Configuration -> Modbus TCP** tab, change settings like on the screen attached below and click **Apply**:



6. Navigate to **I/O configuration** tab, change settings like on the screen attached below and click **Apply**:



If your application requires more or less bytes to be exchanged feel free to change its values. **It is important to have the same configuration in the gateway and in the PLC.**

Configuring Anybus gateway in TIA Portal

This chapter describes how to configure PLC and gateway in TIA Portal software.

Reminding network connections, our PC with TIA Portal has to communicate with PLC controller in order to modify PLC program. Ethernet-Profinet connection is used and devices have an IP from 192.168.250.0 network:

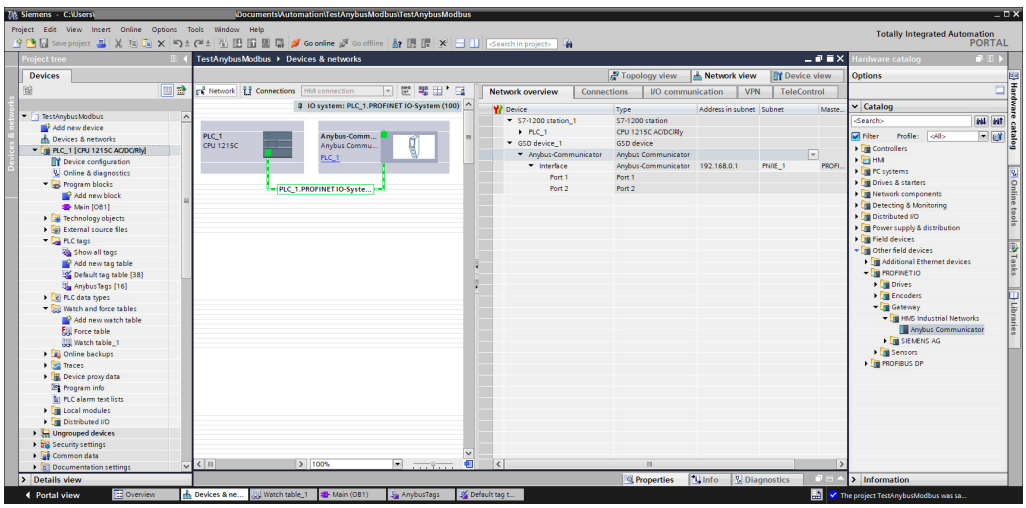
List of IP addresses in network:

- PC – 192.168.0.2;
- PLC – 192.168.0.15;
- Gateway – 192.168.0.1

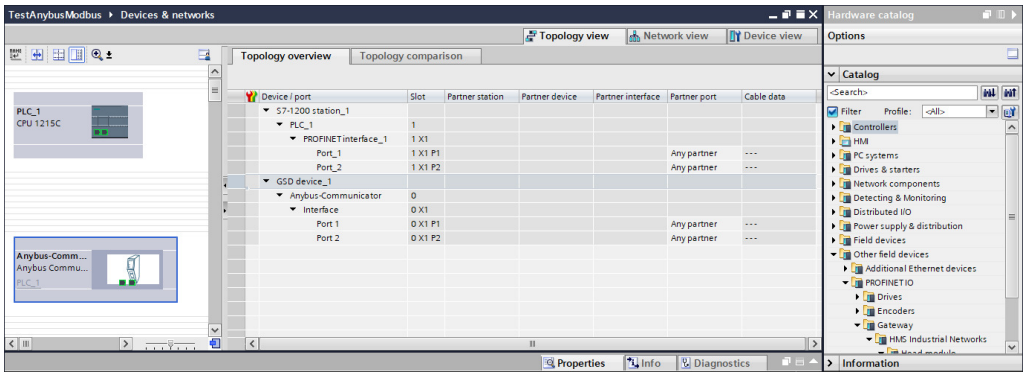
Please find attached below screens of hardware and software configuration from TIA Portal.

1. Create TIA Portal project with your PLC. Add GSD file from downloaded firmware folder in TIA Portal using tab menu on the top:
   "Options → Manage general station … → Select file → Install".

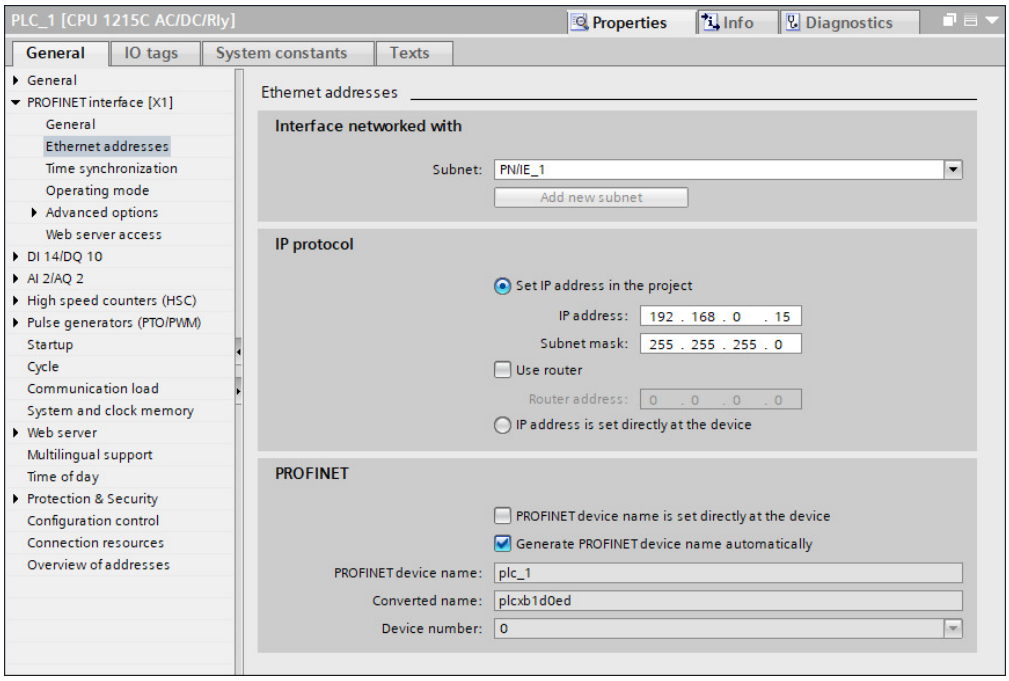
   Once you upload GSD, you can find appropriate device in **Other field devices –> PROFINET IO → Gateway → HMS Industrial Networks**. *Drag & drop* it onto **Network view** and combine it as on the screen attached below:



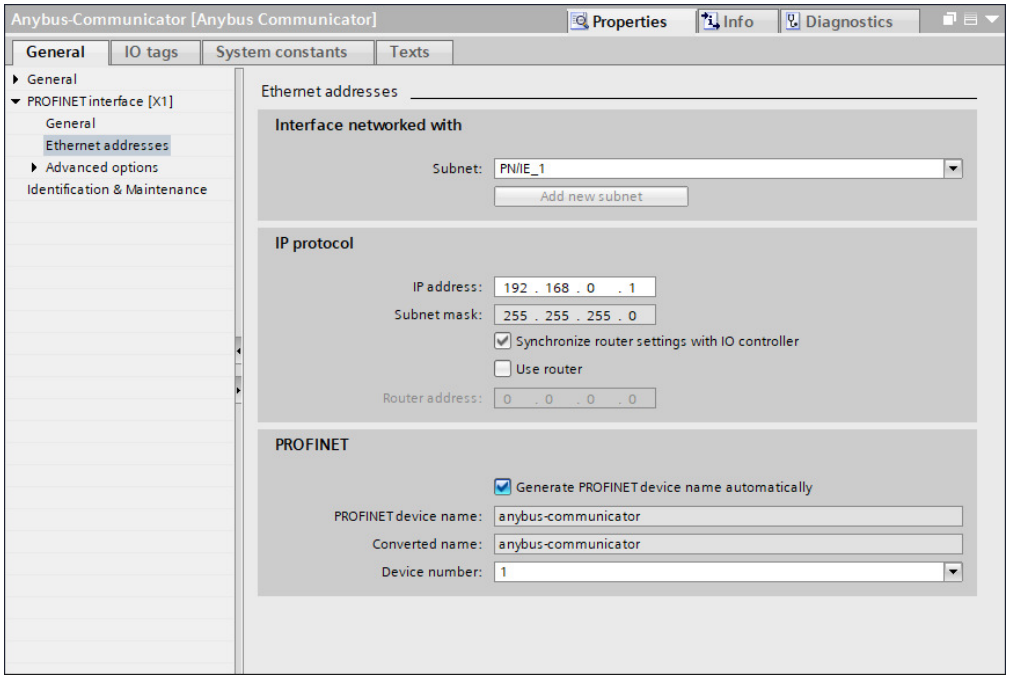
2. **Topology view** is set up this way:



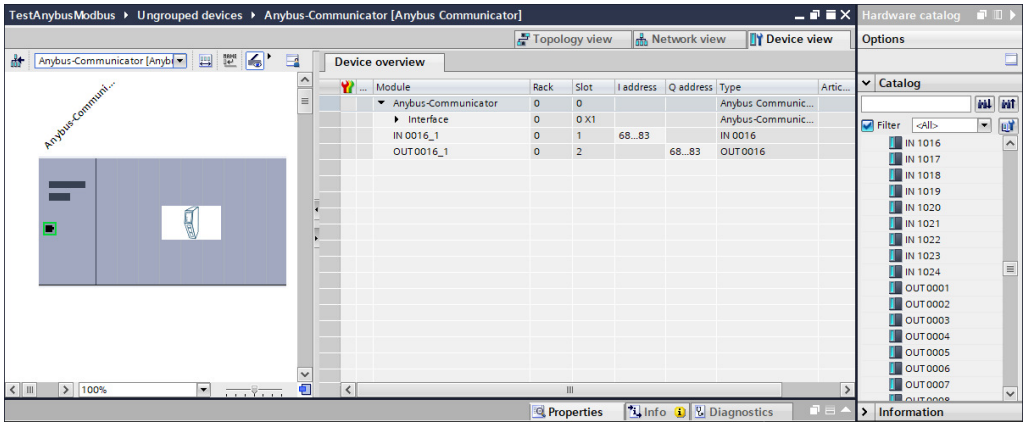
3. Change PLC's IP to **192.168.0.15**:



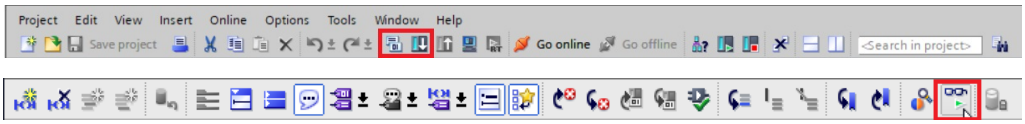
4. Change gateway's IP to **192.168.0.1**:



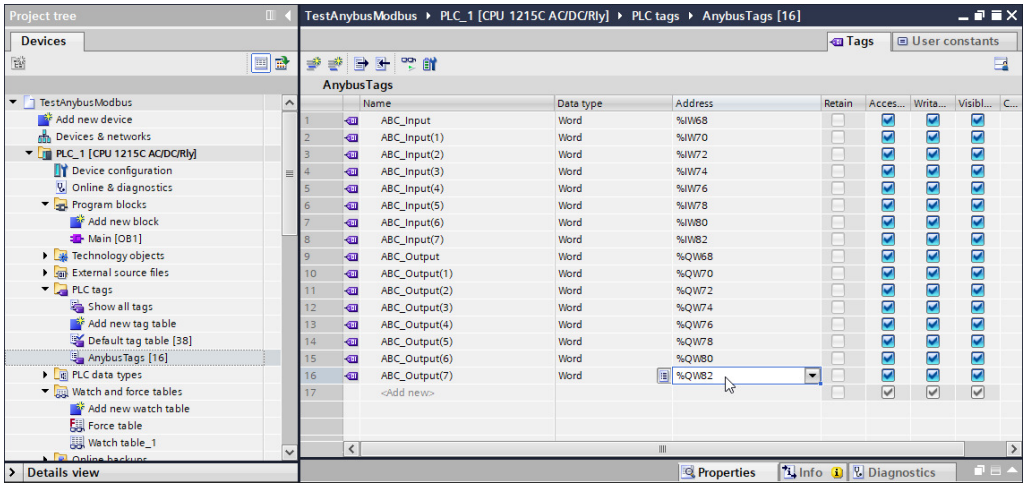
5. Add two modules to the project which match these configured in Anybus web configurator. *Drag & drop* **IN 0016** and **OUT 0016** module onto **Device overview**:



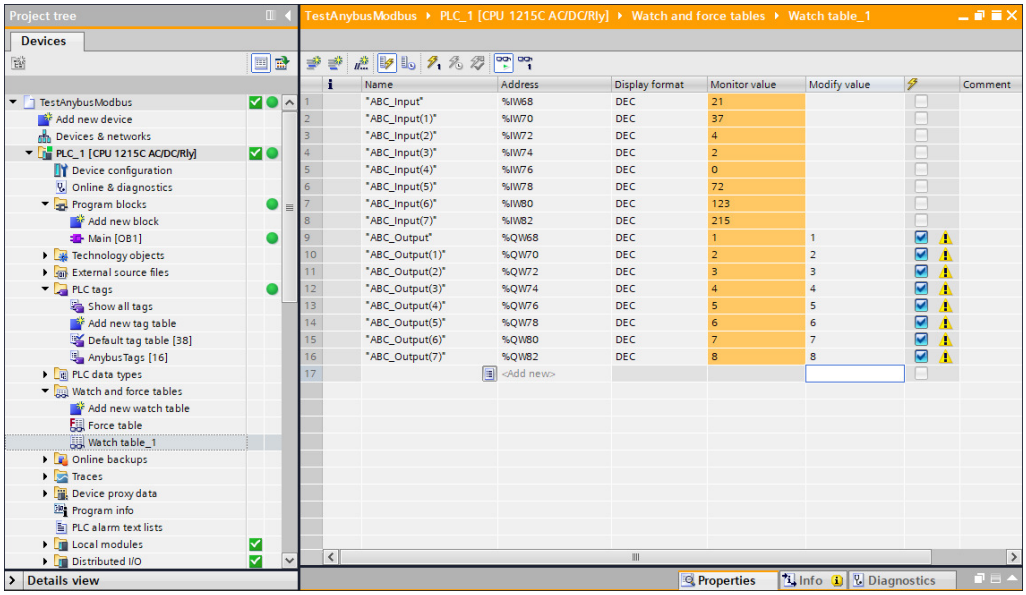
6. If all previous steps have been done correctly, your PLC program is ready. Compile the program and download it to the device. To see current states of variables, turn on Monitoring mode:



7. It is good manner to use gateways In/Outs by PLC tags – using raw addresses you can mistakenly modify/read wrong bytes. Create "AnybusTags" table in **PLC tags** folder:

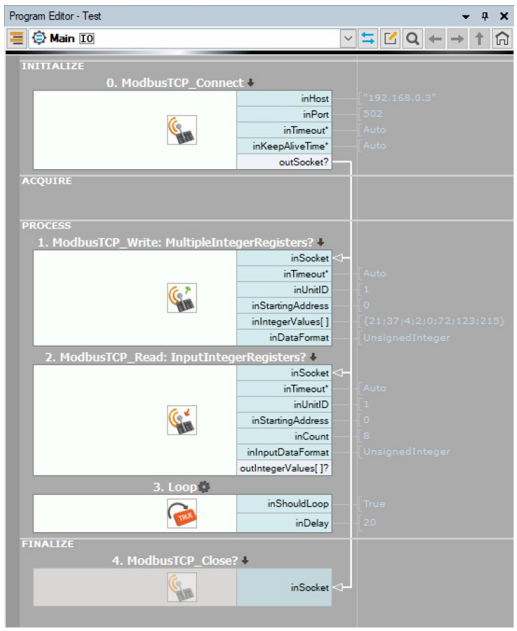


To monitor and easily modify tags please add *Watch table* in **Watch and force tables** folder.:

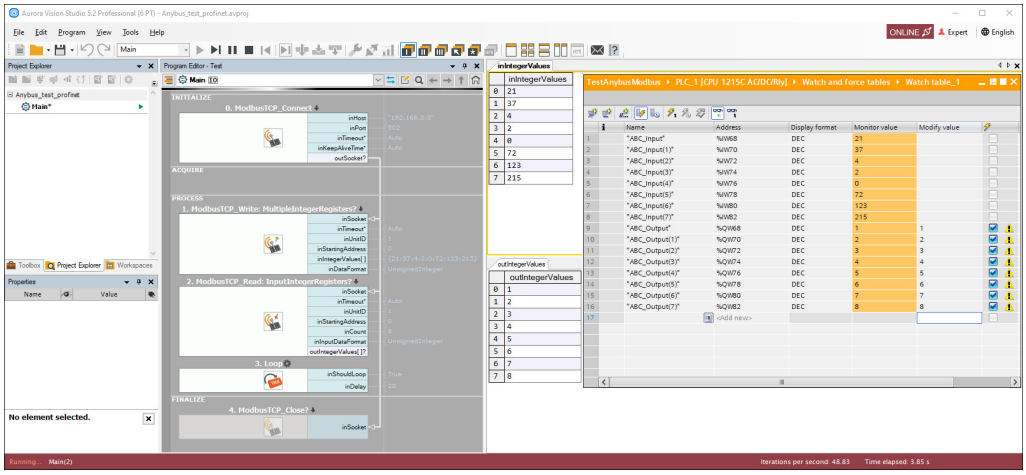


Configuring Aurora Vision Studio application

1. Aurora Vision Studio application should have ModbusTCP_Connect and I/O Read/Write filters as on the screen attached below:



2. Turn on online mode and monitoring mode in TIA Portal. You can create *watch tables* to easily modify data in TIA Portal environment:



Troubleshooting

In case of any problem with gateway configuration, please try manuals and tutorial videos from Anybus support page.

Using Aurora Vision Studio Runtime for Linux with WebHMI

Purpose and requirements

This document describes how to run an application with a WebHMI interface on a Linux based systems using Aurora Vision Studio Runtime for Linux. To connect the host with the target system the SSH connection is used. It allows for data exchange between systems and run commands on the target system remotely (especially useful when the target system does not provide a graphical user interface).

Check if the target system meets requirements listed in Using Library on Linux article.

In this guide iam NET GmbH ARM smart camera is used, but presented steps apply to any linux based device that meets requirements.

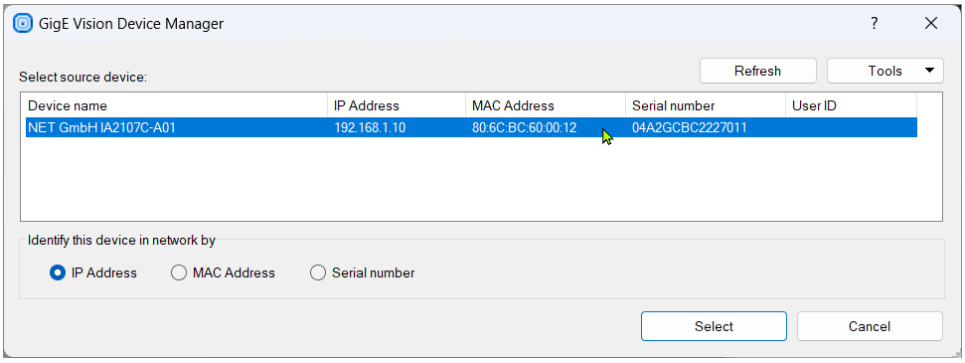
Required equipment:

- Aurora Vision Studio 5.4 Professional or later (version 5.4 was used);
- Linux based target device;
- PuTTY executes the SSH connection;
- Xming might be useful to open graphical interface of Linux applications on Windows;
- WinSCP is responsible for data exchange between host and target systems;
- Aurora Vision Studio 5.4 (or later) Runtime for Linux (version 5.4 was used).

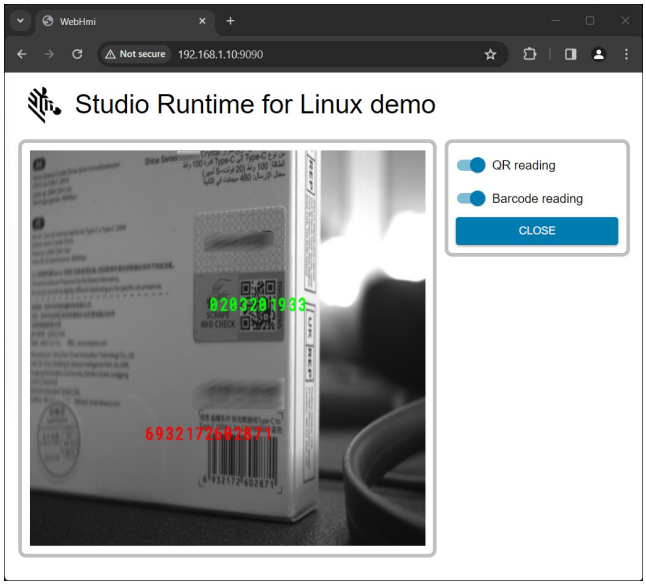
Basic program with the WebHMI interface

This paragraph refers to the *WebHmi - Runtime for Linux Demo* Aurora Vision Studio official example.

Please determine if the smart camera is visible in the GigE Vision Device Manager:



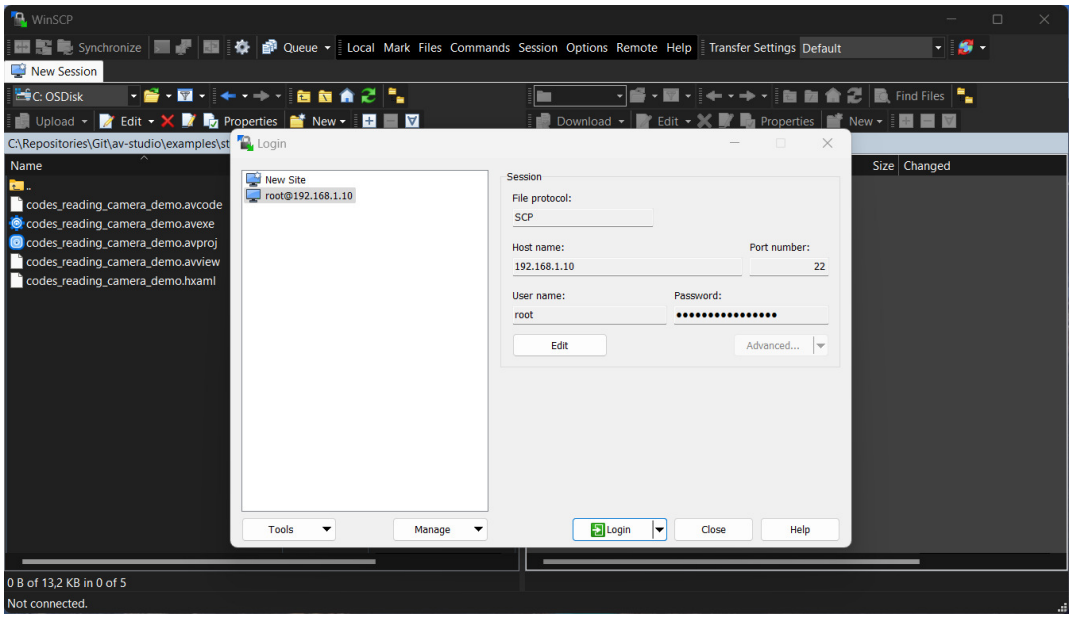
You can run this example on host windows PC and display webhmi in the browser typing **localhost:9090** (default port in the settings is 9090). Our goal is to run example on target linux device:



Aurora Vision Studio Runtime for Linux installation

To manage and exchange files on the target system SSH connection is established. For convenience WinSCP and PuTTY are configured following the Getting started with the Windows Host System guide.

Connect to the iam smart camera using WinSCP:



Default credentials:

```
User: root  Password: root
```

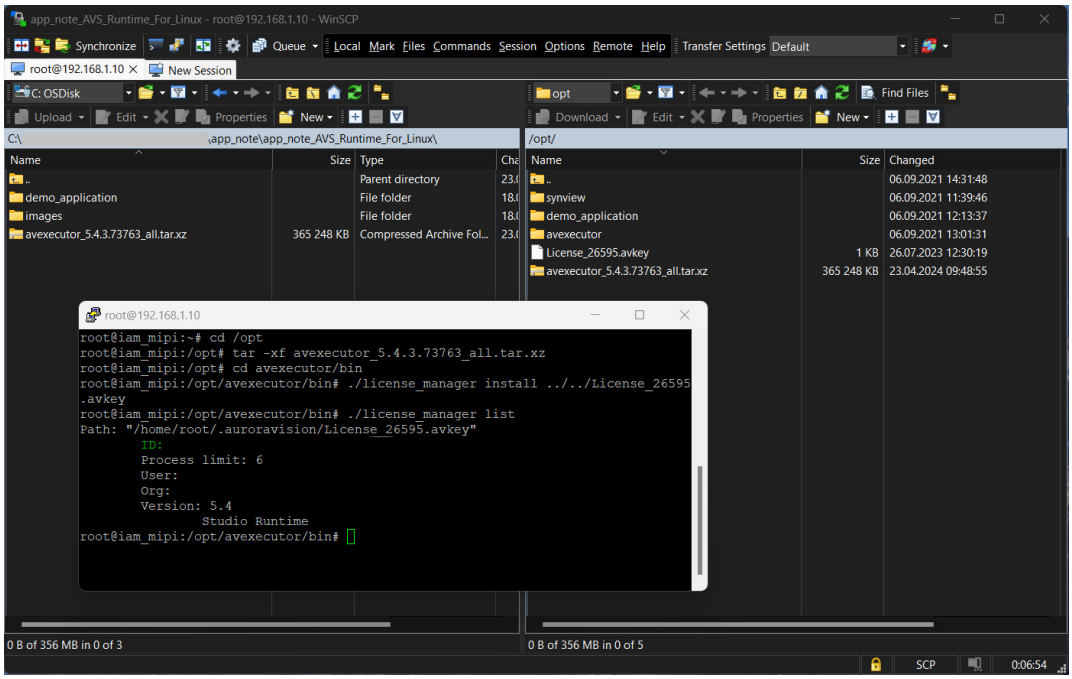
In WinSCP window on the left there is the Windows host file tree, on the right the Linux target tree. In this tutorial we copied executor installer and license to the **/opt** folder. You can open session with PuTTY using *Ctrl + P* shortcut or clicking an icon with the bolt on the top toolbar in WinSCP (login with the default credentials). If you begin in the *home* folder, navigate to the **/opt** executing:

```
cd /opt
```

Then execute following commands:

```
tar -xf [avexecutor_archive.tar.xz]
cd avexecutor/bin
./license_manager install ../../[License_file.avkey]
./license_manager list
```

It will install the executor, load the license key and list attached licenses at the end:

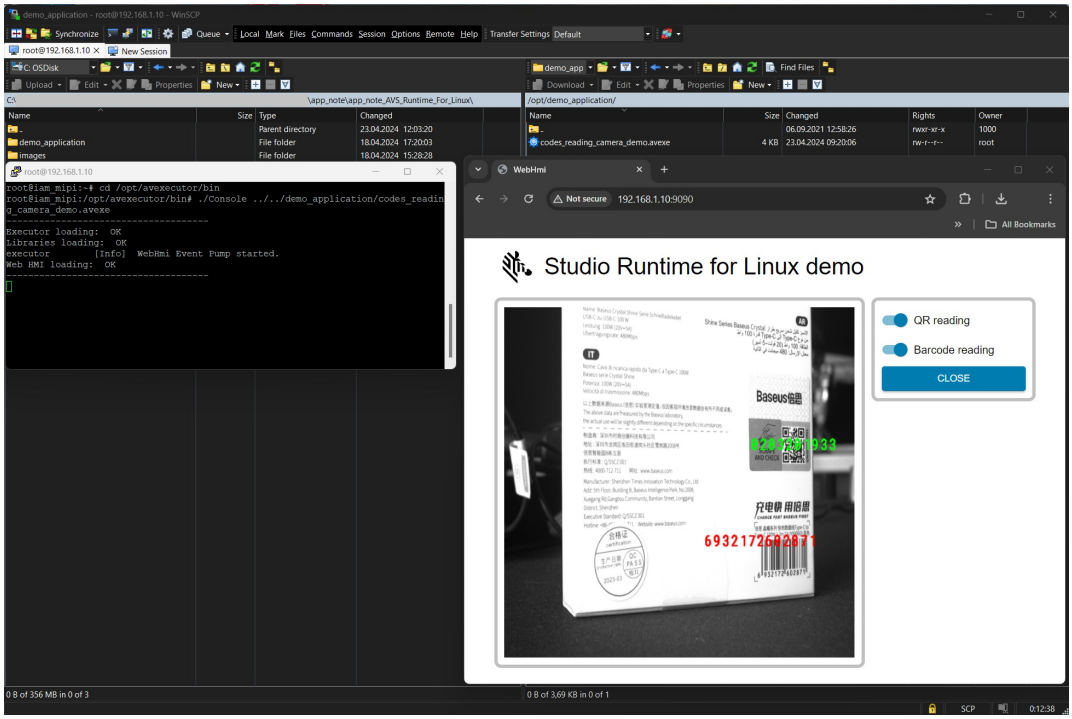


Aurora Vision Studio Runtime for Linux solution deployment

Generate runtime solution in Aurora Vision Studio navigating to *File -> Export to Runtime Executable...*. In this tutorial .avexe solution is copied to **opt/demo_application** folder. To deploy solution execute following commands:

```
cd /opt/avexecutor/bin
./Console ../../demo_application/[Solution.avexe]
```

To display the WebHMI on the host system, type **192.168.1.10:9090**:



Console application allows users to specify parameters others than just the path to the .avexe file. When working on application with WebHMI, it might be especially useful to modify WebHMI port value. This can be done with **-webhmi-port<N>** parameter where N indicates requested port value (default one is 9090). To list all parameters user just need to run Console without any of them **(./Console)**.

Zebra
Aurora™ Vision

This article is valid for version 5.7.5

Legal Terms of Use Privacy Policy

ZEBRA and the stylized Zebra head are trademarks of Zebra Technologies Corp., registered in many jurisdictions worldwide. All other trademarks are the property of their respective owners. ©2026 Zebra Technologies Corp. and/or its affiliates.