Application note			
Subject	MicroHAWK V430-F PROFINET Setup with Siemens PLCs using TIA Portal 15.1	**Date**	25-May-2021
Version	1.4	**Author**	Eldad Ben Shalom, FAE EMEA eldad.ben.shalom@omron.com

Subject

This document describes the setup configurations required for adding a V430-F Code Reader to a Siemens PLC project using TIA portal version 15.1. This is a walkthrough document with working examples of how to read / write data, trigger, and send match data to the reader.

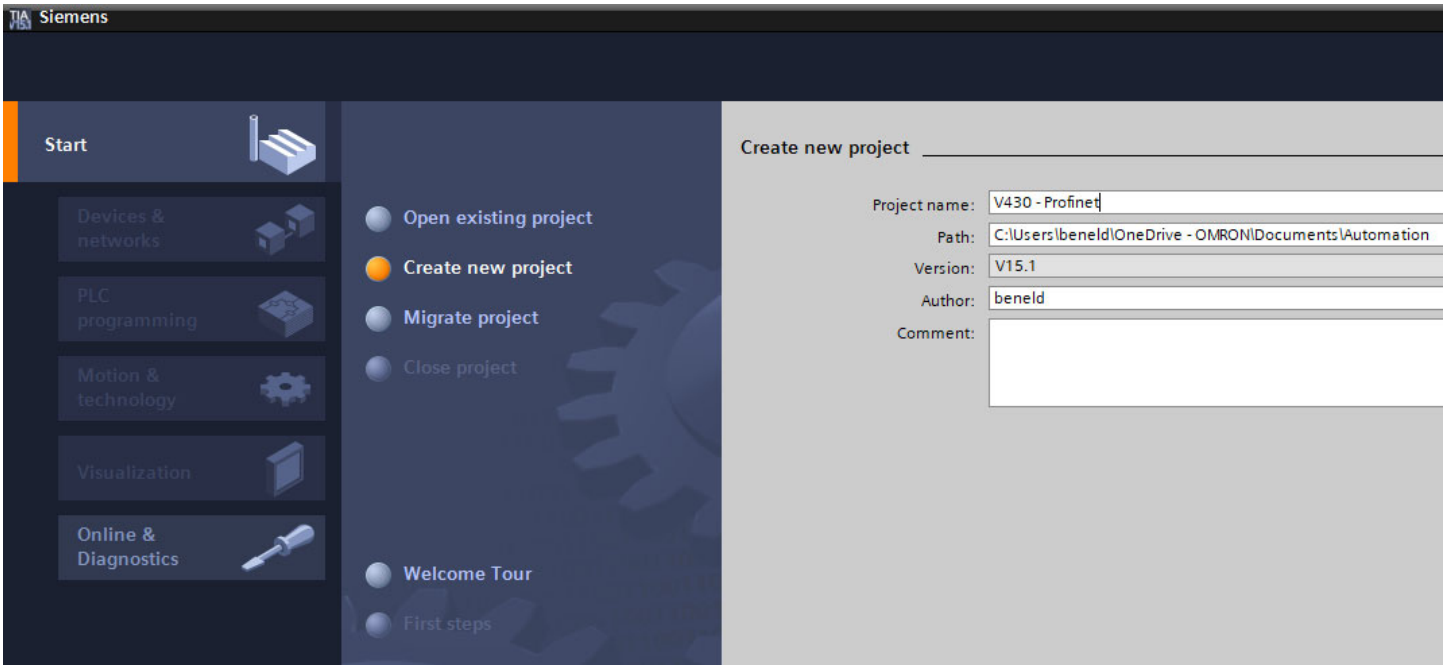
Required Hardware

This tutorial uses Siemens PLC 1200 (1214C), TIA software 15.1, and V430-F with firmware version 2.1.

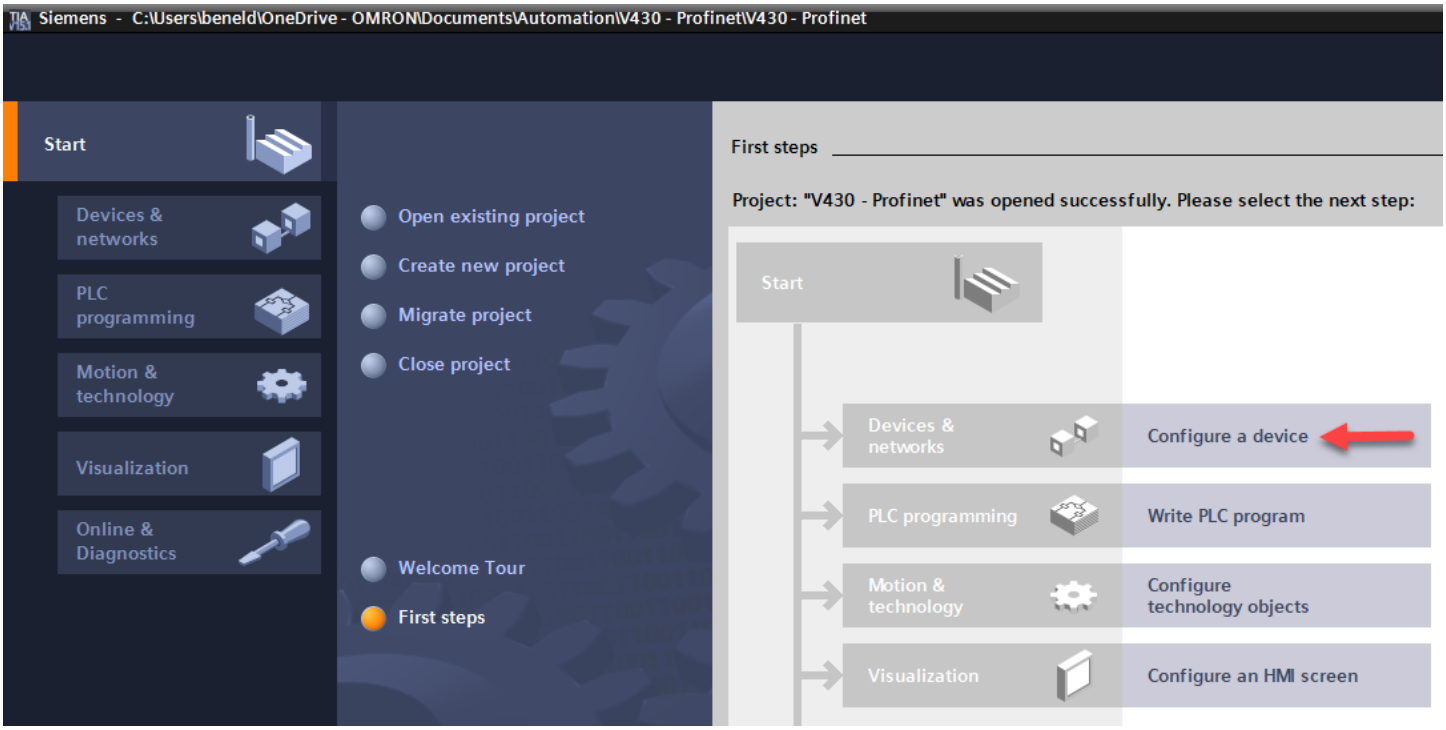
Table of Contents

Create new project

1. Start-up TIA software and create a new project, providing a new name and path.

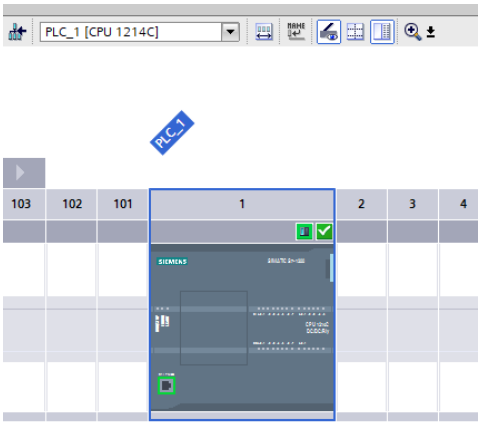


2. In the next step, select ‘Configure a device’.


Siemens - C:\Users\beneld\OneDrive - OMRON\Documents\Automation\V430 - Profinet\V430 - Profinet
Start
Devices & networks
PLC programming
Motion & technology
Visualization
Online & Diagnostics
Open existing project
Create new project
Migrate project
Close project
Welcome Tour
First steps
First steps
Project: "V430 - Profinet" was opened successfully. Please select the next step:
Start
Devices & networks
Configure a device
PLC programming
Write PLC program
Motion & technology
Configure technology objects
Visualization
Configure an HMI screen

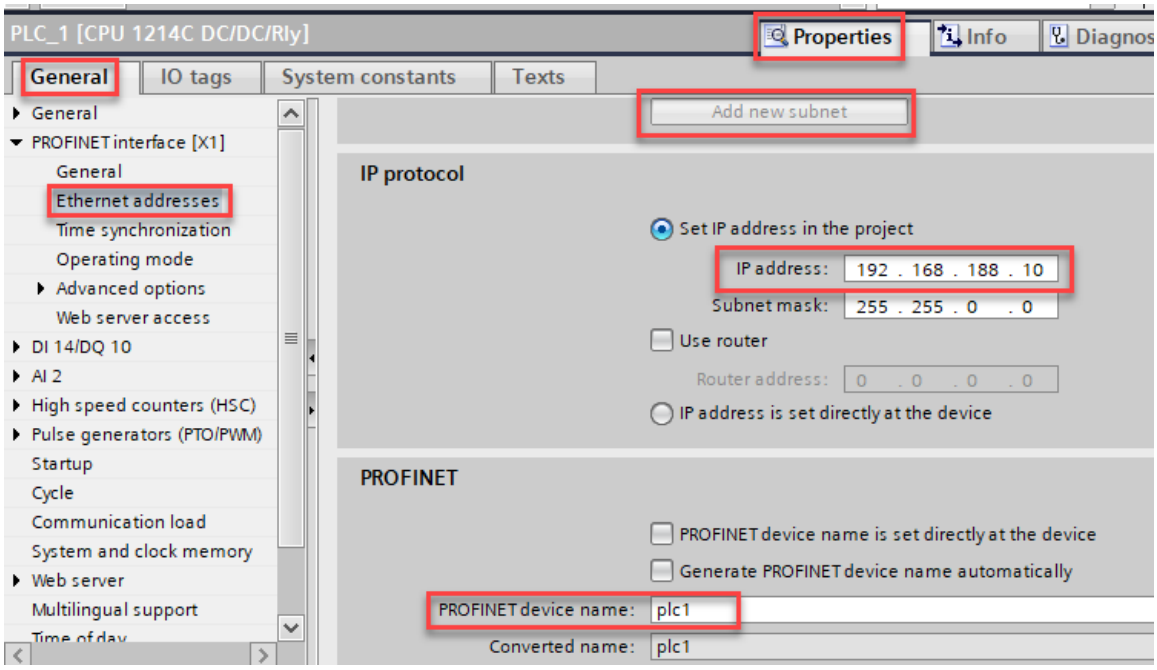
Adding the PLC controller to the project:

3. Adding a PLC controller: Select **Add new device** -> **Controllers**, then select the correct PLC type for your HW.

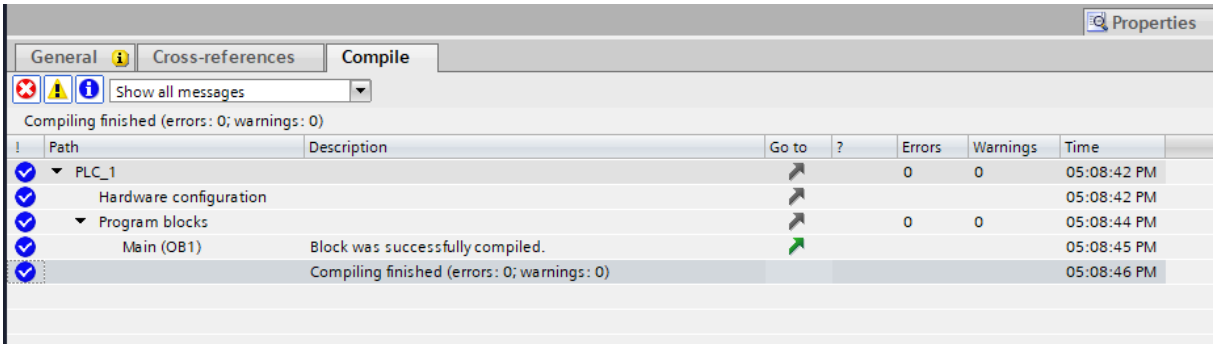


- Add any other HW module that you may have in your setup (I/O, communication), to avoid possible errors later related to those modules. After adding HW verify no errors on PLC.

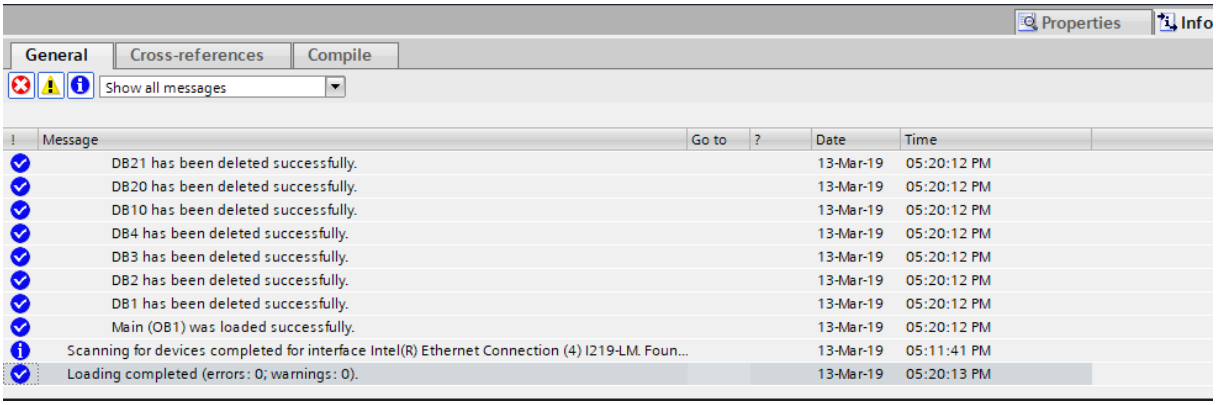
4. Double click on 'Device configuration' to reach this screen below. Click on PLC->Properties->General tab. Then select Profinet interface->Ethernet addresses, to set the desired IP address of the PLC. (Verify that IP addresses and subnet mask of PLC, Camera and PC are on the same network). After typing the Network settings for the PLC, click on button 'Add new subnet'.



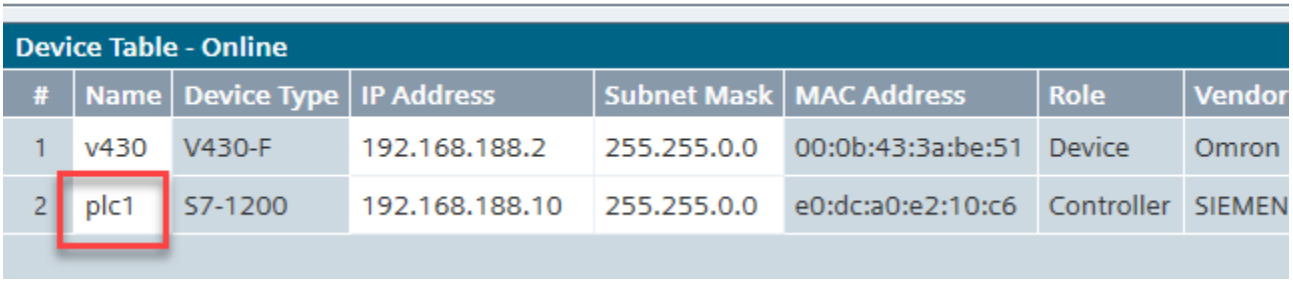
5. Save project and compile. Verify successful compilation with no errors:



6. Connect to PLC and download project. In load preview window, click Load, then Finish. Verify that project download is completed without errors:



7. For network settings, read/write IP addresses and Profinet device name, use Proneta Utility. Verify that the PLC has the correct Profinet name as used in the project settings.

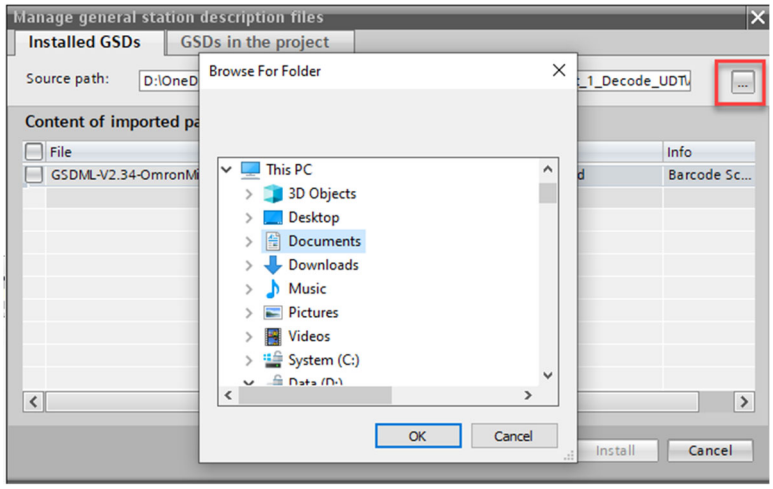


Loading the GSD configuration file of the V430 reader

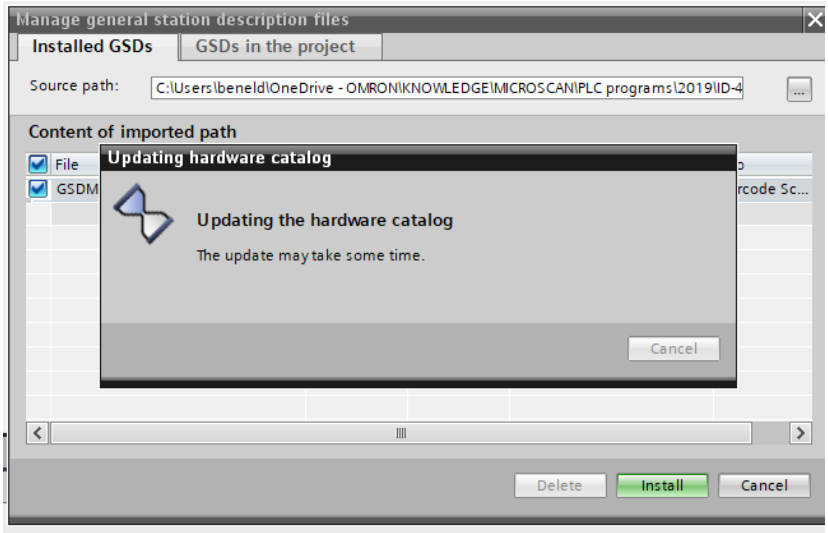
1. Download the GSD file provided by OMRON for the V430 reader.
Available for download from GTKS Omron documents website, or Product page on OMRON website:
https://automation.omron.com/en/us/products/family/V430

- Verify that the GSD file you are using corresponds to the FW version on the reader.

2. Once the correct GSD file is saved locally on the PC, Select Options->Manage GSD.
Browse to the folder containing the GSD file and load it. Then select the file and click 'install'.



- When the file status shows **'Already Installed'**, it means that the file is installed successfully.
After first time GSD file installation, the HW catalog in TIA will be updated immediately with V430 data:

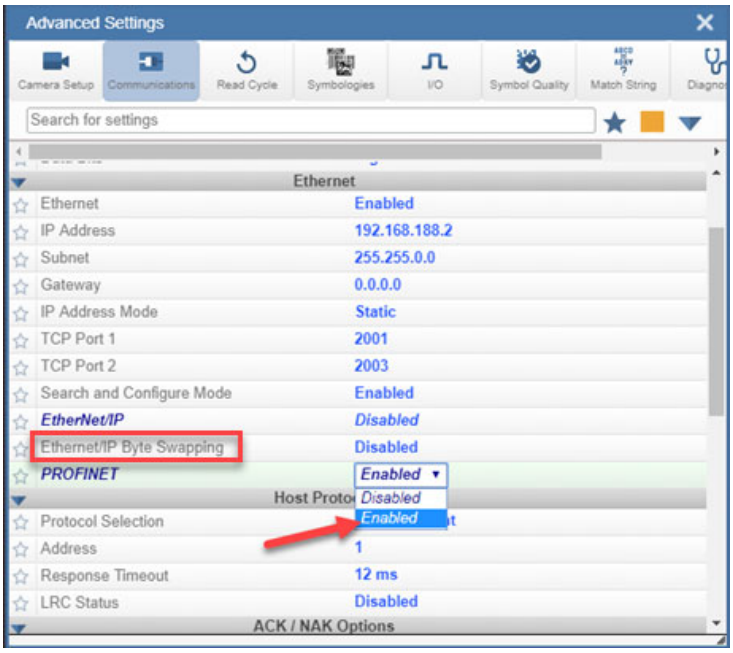


Omron Europe B.V.
Wegalaan 67-69
2132 JD Hoofddorp

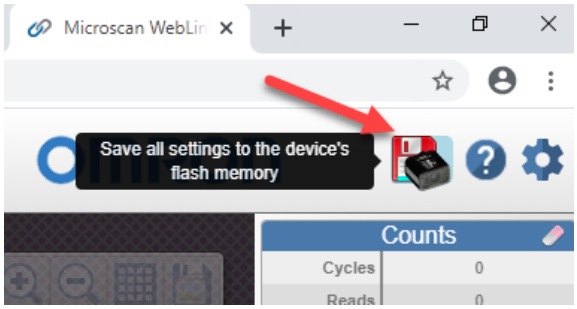
Enable Profinet protocol in the reader using Weblink:

1. Enable Profinet communication in the reader, using Weblink. To access Weblink, Open browser and type the IP address of the reader. (Default: 192.168.188.2)

- Enable Byte Swap as well, to be able to read data correctly on PLC side.

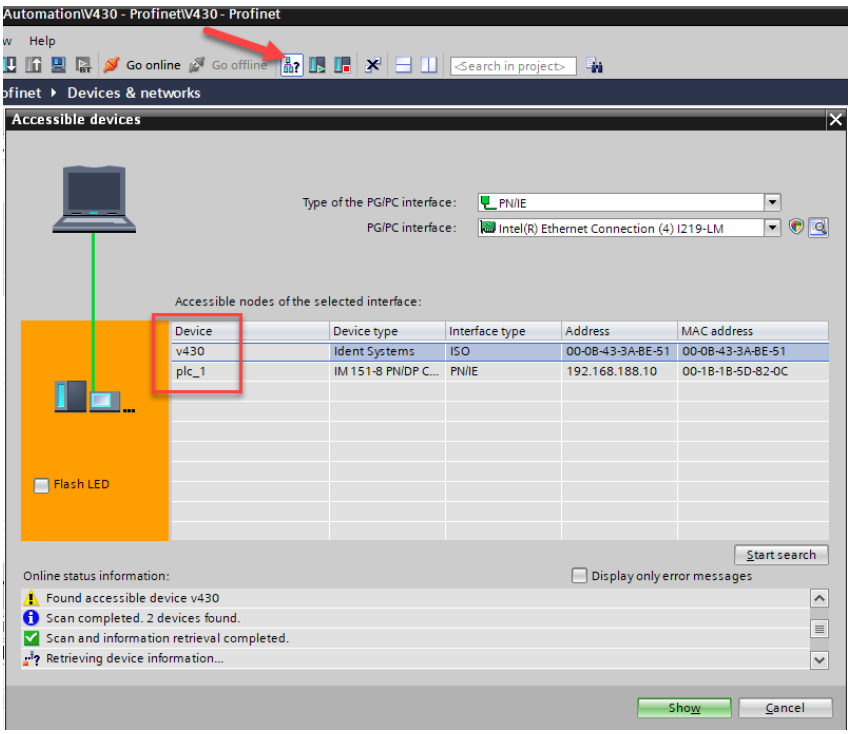


2. Save settings to flash, then reboot the reader, so the changes will take effect:

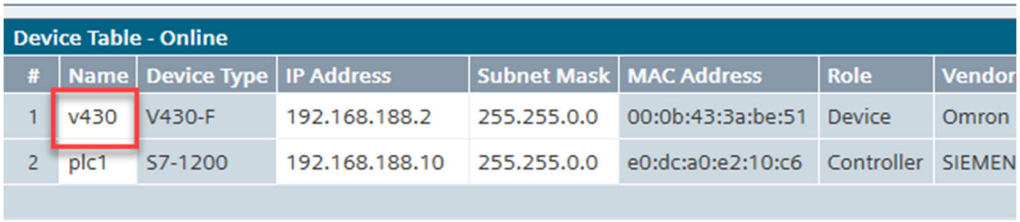


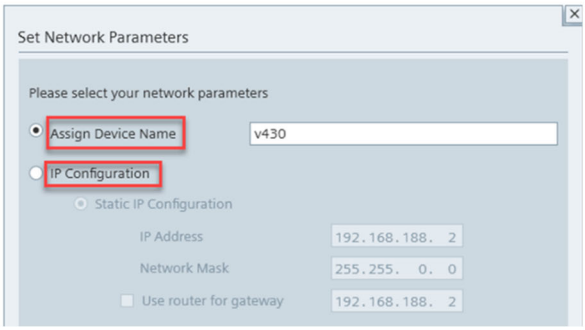
Adding the V430 reader to the project:

1. First check the existing devices on the network by pressing the button ‘accessible devices’. Verify that the PLC has a **Profinet device name**.

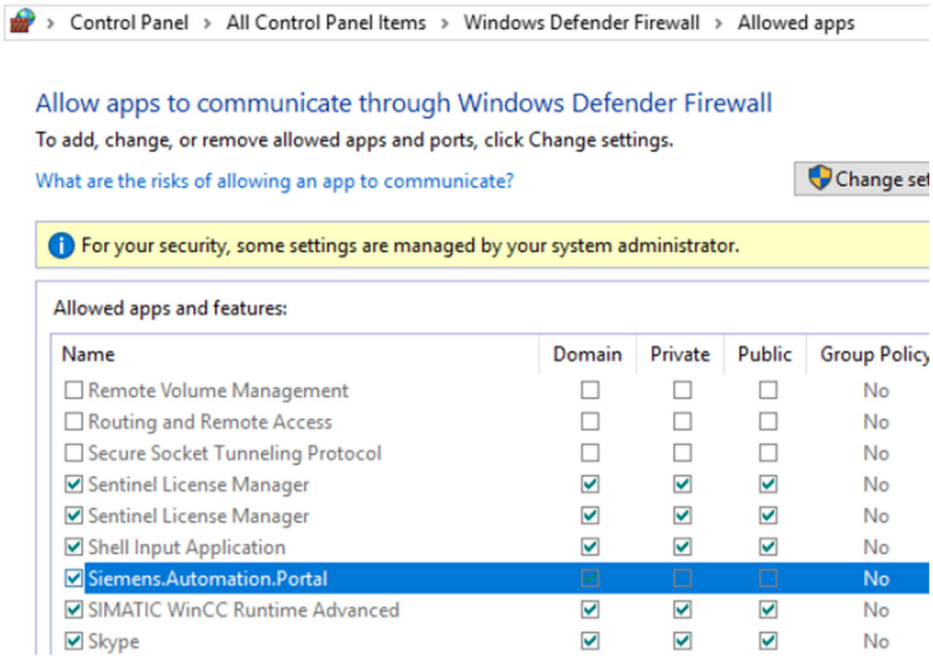


- When the above window cannot detect the reader or PLC, use **Siemens Proneta Utility** to read IP settings and existing Profinet name. When required, these settings can be changed in Proneta. The Profinet device name appears in Proneta must be identical to the Profinet device name used in the TIA project (see following steps). Otherwise, Profinet connection cannot be established.

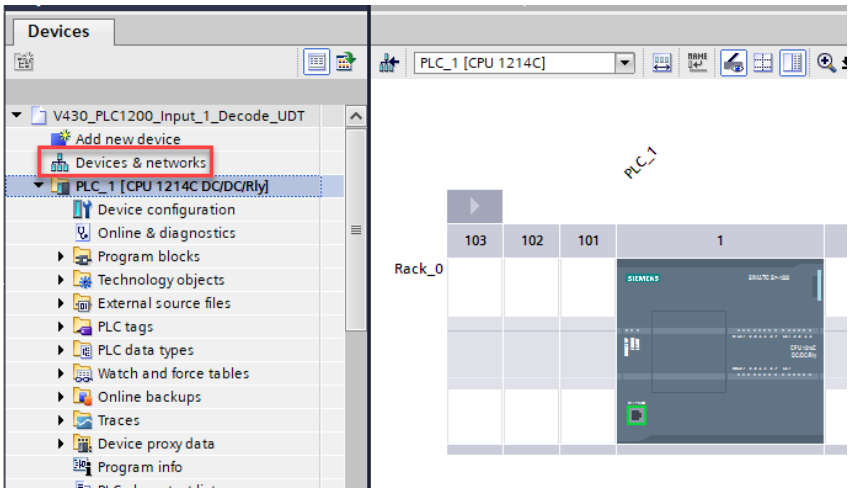




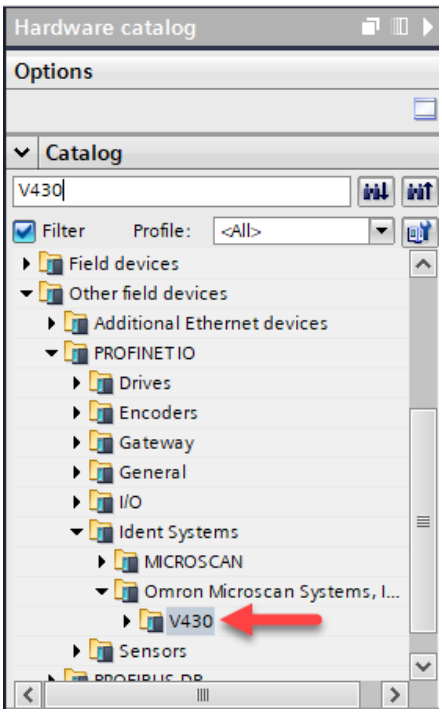
- If the reader is not discovered by TIA, it is possible that UDP packets are blocked by the firewall. To solve, verify that the TIA portal app is added to the allowed app list in your firewall settings. If not, add it manually.



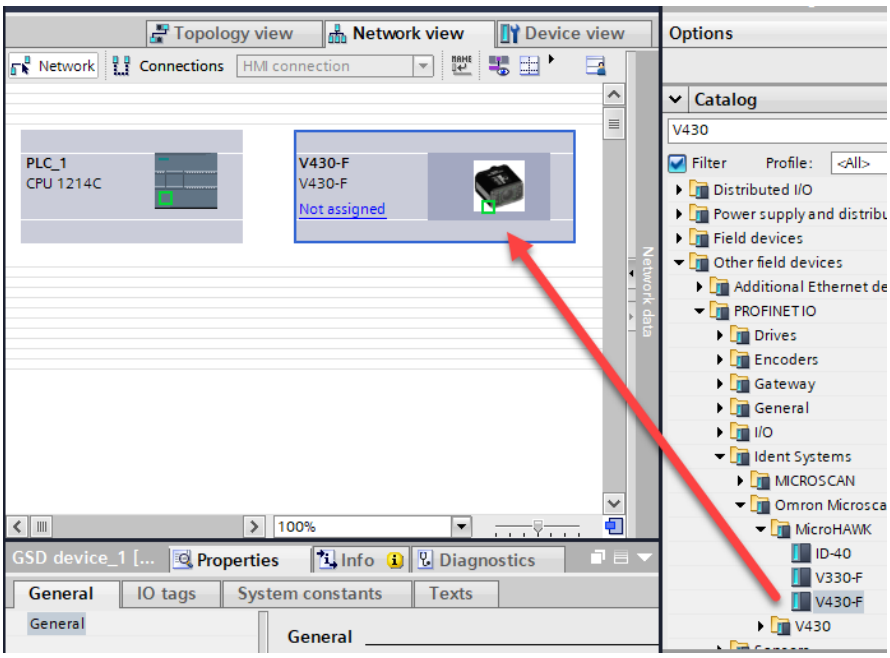
2. When the reader is ready to be added to the project, double click ‘Devices & Networks’ in on ‘Devices’ panel to view all existing devices in the project:



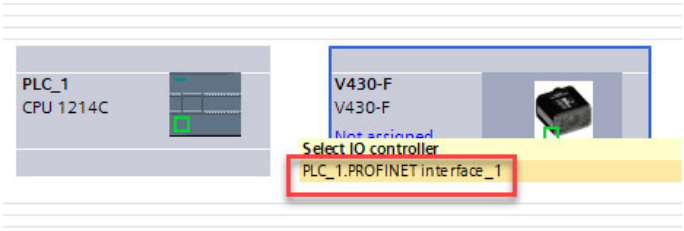
3. Select the HW catalog tab and browse to the V430 reader (or use the search box):



4. Drag the reader to the network view panel, so the reader will be added to the network view:

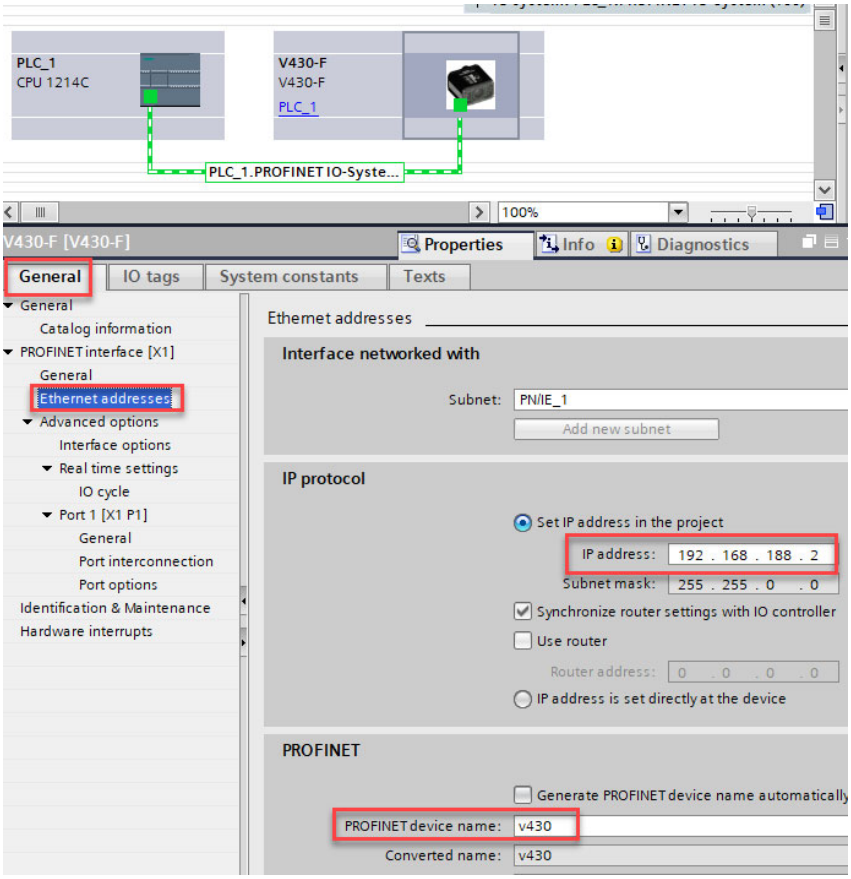


5. Click on **'not assigned'**, then select the Profinet interface, so the PLC and reader will be connected over the same Profiinet interface:

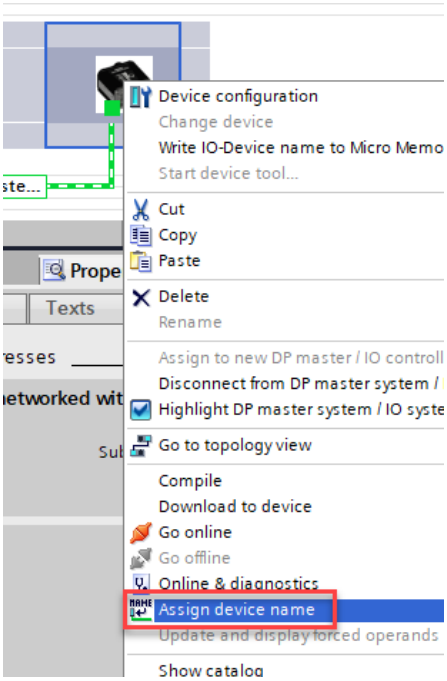


6. Right click the reader icon, then select properties. Select **General tab->Profinet interface**, and set Profinet name for the reader:

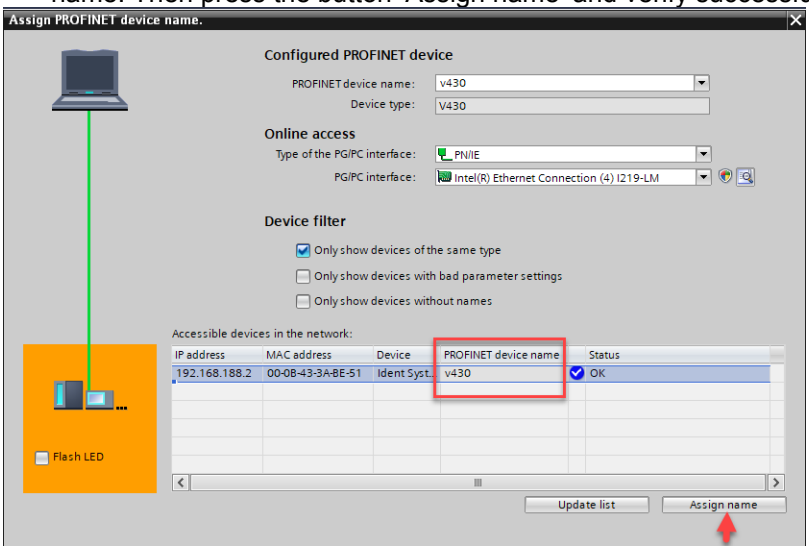
   - Note that Profinet name must be properly assigned and identical to the Profinet name on the device which can be read using Proneta utility. Otherwise, Profinet connection is not possible.
   - Notice that <u>this is not the Device name parameter</u> which is saved by the factory on the flash memory of the reader:



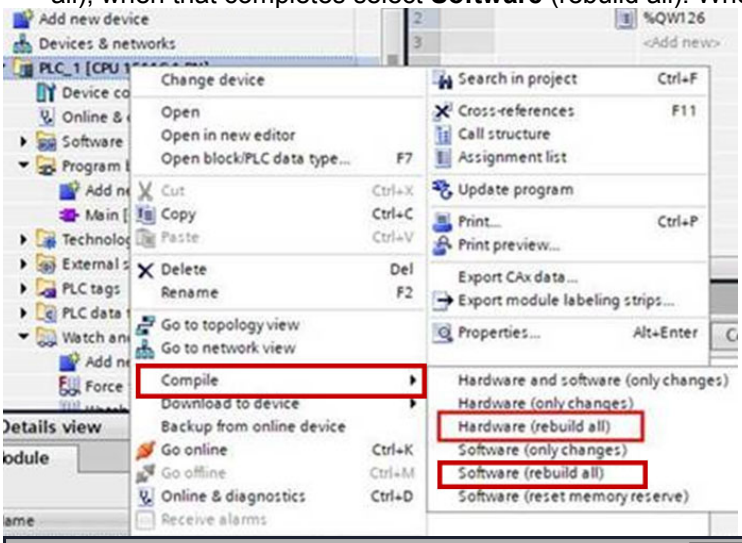
7. To assign the Profinet name of the device in the TIA project, right click on the reader icon, then select ‘Assign device name’:

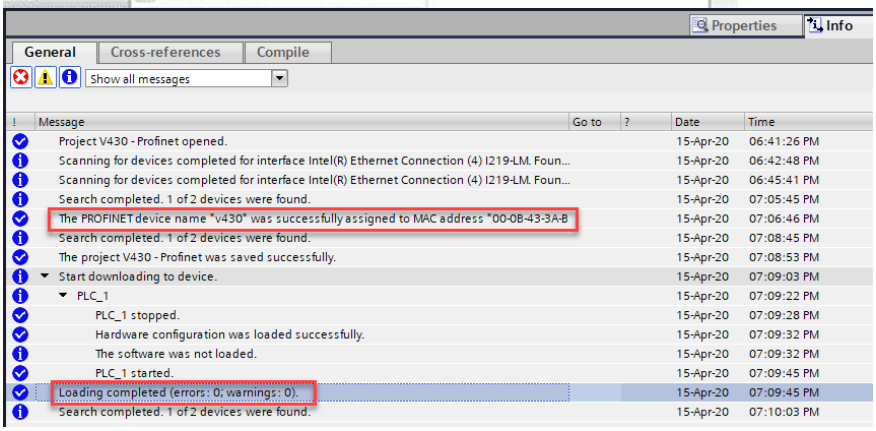


In the Assign window which will pop up, click ‘update list’ to see the reader, then select the reader and press ‘assign name’ to give the reader the new Profinet name we typed in the previous step. **This step is mandatory for communication over Profinet to work.** After ‘Update list’ button is pressed, the reader appears without an assigned name. Then press the button ‘Assign name’ and verify successful result:



8. Once the reader’s name is assigned successfully, Compile the project: Right click on PLC, select **Hardware** (rebuild all), when that completes select **Software** (rebuild all). When done, download to PLC and verify no errors on PLC.



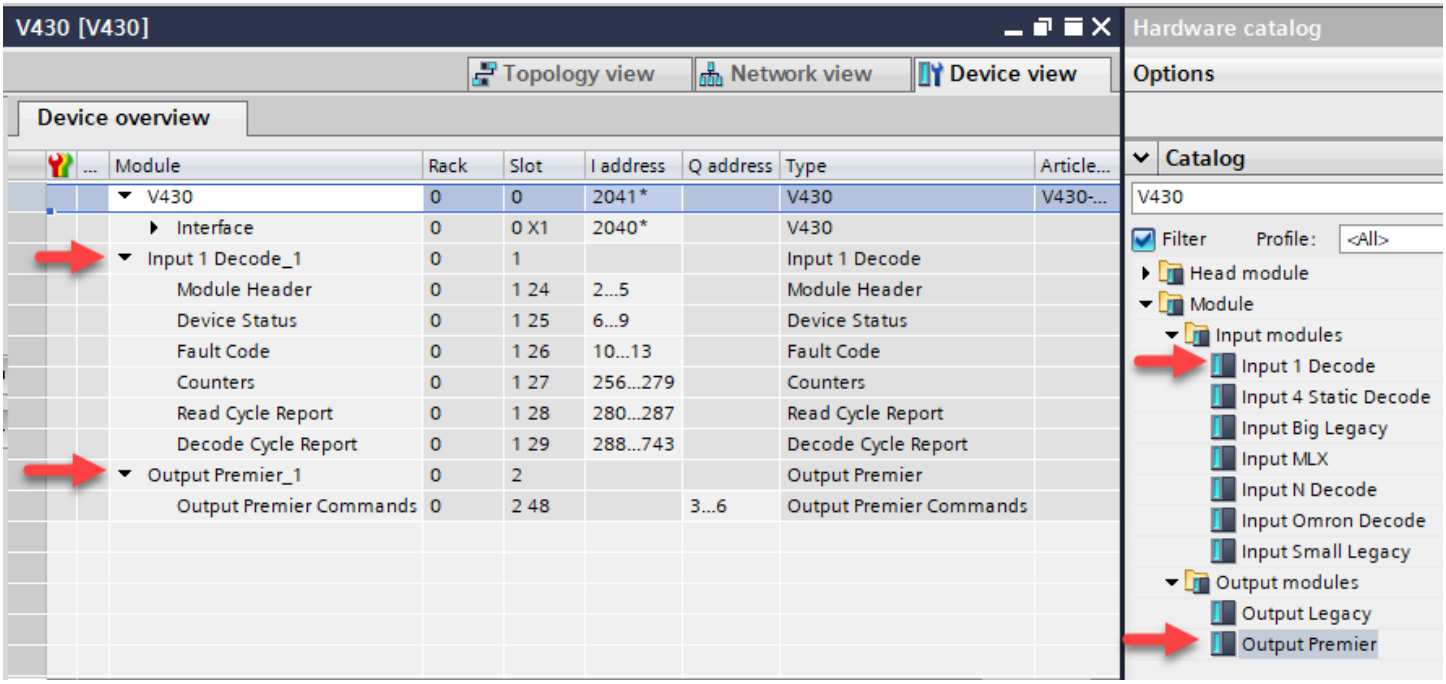


Adding input / output memory modules

To read or write data from the reader, first the input / output modules should be added to the project. We will do this in 'Devices & Networks' view. Then double click on the reader icon to enter the device view.

1. Go to Hardware catalog window and double click the desired modules to add them to the project. In this example we will add the following assemblies:

=>Input 1 Decode (To read results or status data sent from the reader to PLC)
=>Output Premier (To write data from the PLC to the reader. For example, trigger input)



- The two modules above must be selected together as a pair. Only one module can be selected from input, and one form output. See table below for all the valid pairs of input/output assemblies.
- Note that selecting a wrong combination of input/output assemblies might lead to malfunction in the communication.

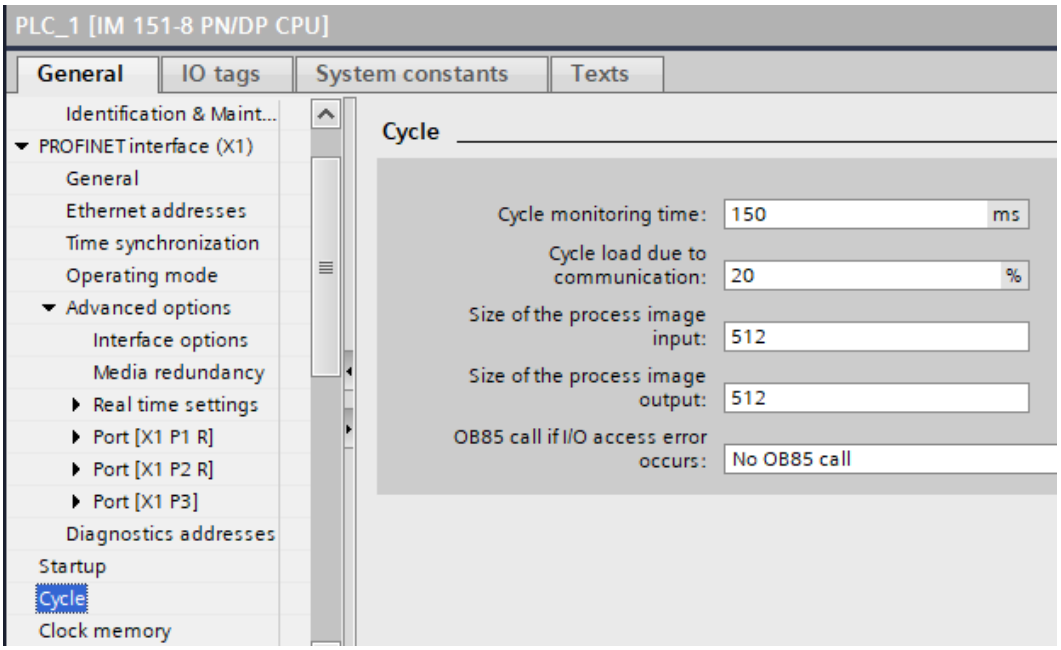
Possible combinations	Input Assembly name	ID Number	Output Assembly name	ID Number
1	Small Input	100	Output legacy	198
2	Big Input	101	Output legacy	198
3	MXL/SLC	102	Output premier	197
4	Input 1 decode	103	Output premier	197
5	Input 4 decode	104	Output premier	197
6	Input N decode	105	Output premier	197

With the input assembly selected above (Input_1_Decode), the following info is available to be read from the reader: (See documentation for more details).

Member Name	Size (Bytes)
INFO BITS	1
RESERVED	1
RESERVED	1
RESERVED	1
DEVICE STATUS	4
FAULT CODE	4
COUNTERS	24
READ CYCLE REPORT	8
DECODE CYCLE REPORT	8
CODE TYPE	4
PIXELS PER ELEMENT	4
DECODE DATA LENGTH	4
DECODE DATA STRING	436

Total Size: 500 Bytes

- In case there are problems with adding assemblies or with read/write data, increase the process image size in the properties of the PLC to 512 bytes instead of 128 bytes. The existing default of 128 is too small to include all the data when using large assemblies:



Read status info from the reader

3.5.4.2.2.1 Device Status Bit Field

DEVICE STATUS

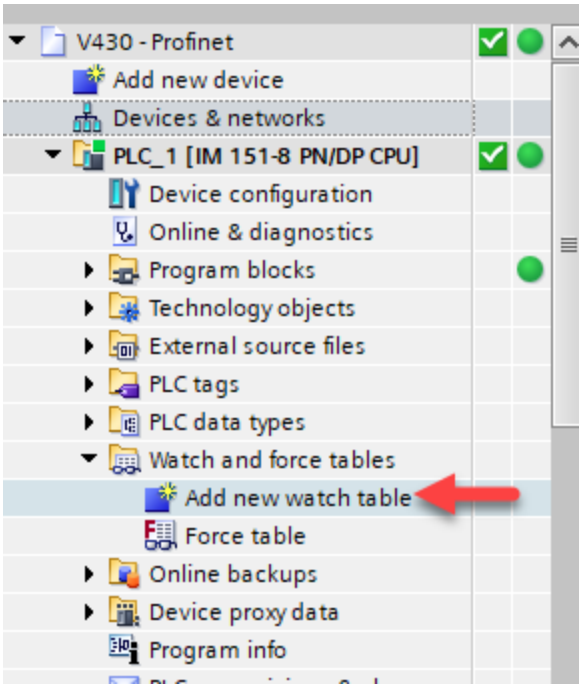
BIT FIELD	Status
0	Online
1	Trigger Acknowledge
2	Exposure Done
3	Decoding
4	Data Is Ready
5	Read Cycle Pass
6	Read Cycle Fail
7	General Fault
8	New match code acknowledged
9	Match Code Enabled
10	Image Sensor Calibrating
11	Image Sensor Calibration Complete
12	Training
13	Training Complete
14	Optimizing
15	Optimization Complete
16	AutoImage Photometry Enabled
17	AutoImage Photometry Complete
18	Output1 Status
19	Output2 Status
20	Output3 Status
21	Buffer Overflow
22-31	Reserved

When looking at the Input addresses table in our project in TIA portal, we see that this register has byte address 6 to 9:

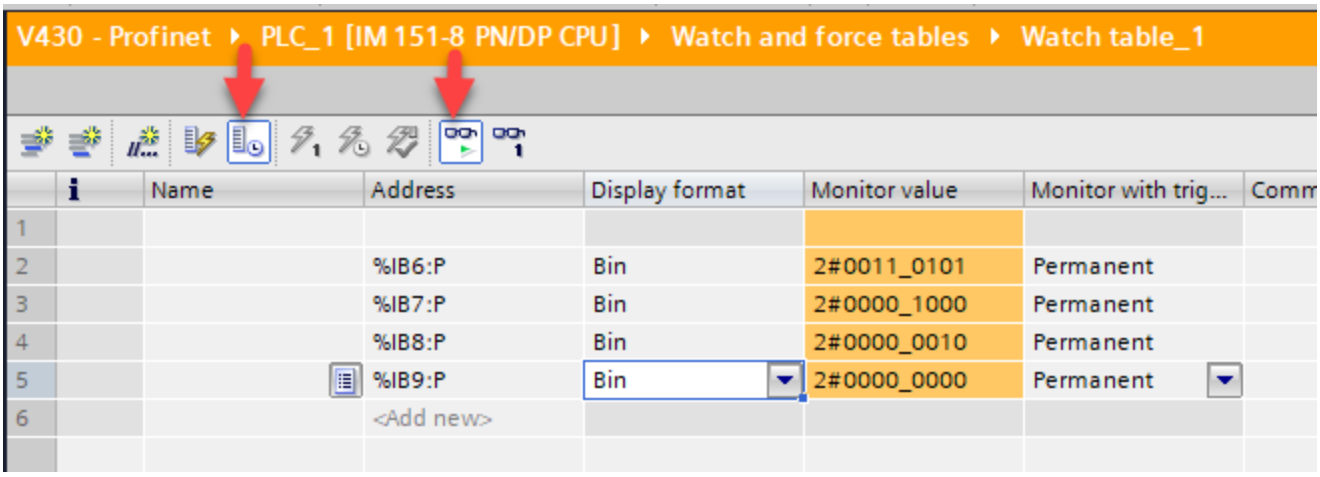
Device overview

Module	Rack	Slot	I address	Q address	Type	Article...
V430	0	0	2041*		V430	V430-...
Interface	0	0 X1	2040*		V430	
Input 1 Decode_1	0	1			Input 1 Decode	
Module Header	0	1 24	2...5		Module Header	
Device Status	0	1 25	6...9		Device Status	
Fault Code	0	1 26	10...13		Fault Code	
Counters	0	1 27	256...279		Counters	
Read Cycle Report	0	1 28	280...287		Read Cycle Report	
Decode Cycle Report	0	1 29	288...743		Decode Cycle Report	
Output Premier_1	0	2			Output Premier	
Output Premier Commands	0	2 48		3...6	Output Premier Commands	

1. To read the status of the reader, create a new watch list
   (select watch and force tables->add new watch list)



2. Define watch items of the 4 bytes of status register. Use the correct syntax as shown in the example below, to watch the data as byte type. (You may also define each bit in separate as bool type).
   When done adding watch items, go online, verify that PLC is running, and press on 'Monitor All' in the watch window. Verify that expanded mode view is enabled:



When the reader is triggered while watching the monitor window above, verify that the correct bits change state – as expected. See the documentation for the list of available status bits (see table later in this document as well).
Trigger once and watch these quick examples:

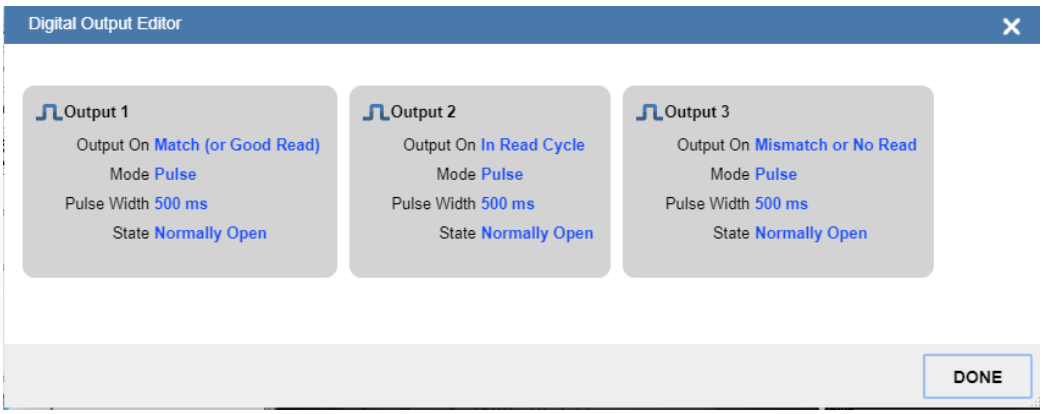
Online status bit

=>Online bit (bit 0) should be always TRUE when the reader is online.
=>Inspection Pass and Inspection fail bits (bit 5 and 6) should be corresponding to the decode result (Pass / Fail).

Descrete output status bits

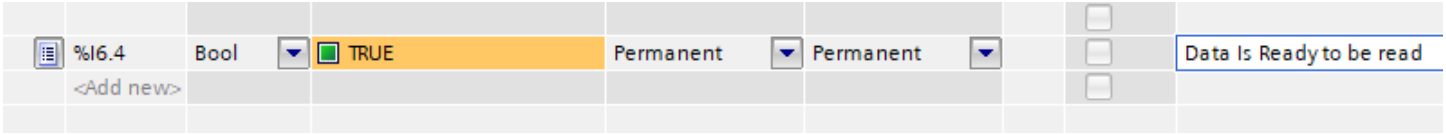
=>Discrete outputs 1,2,3 (bit 18, 19, 20) should be corresponding to the chosen configuration in the reader which can done via Weblink.

See below the configuration window for discrete outputs in Weblink. Configure the reader with the same settings, then trigger once and watch the relevant status bits. Verify that the values are as expected.



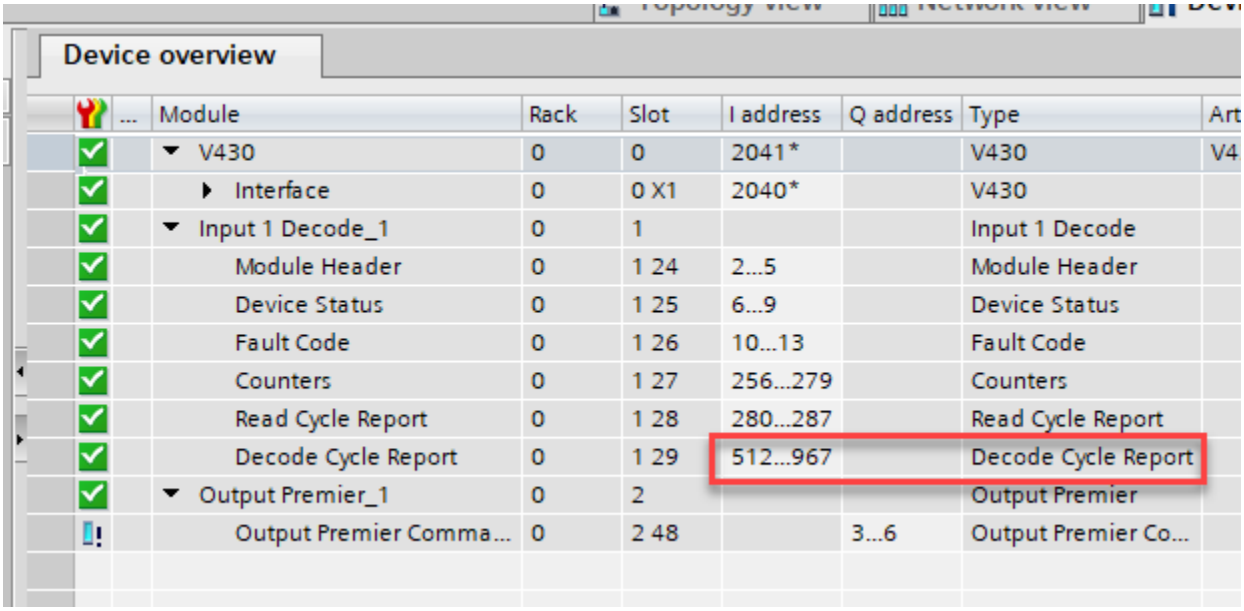
Data Ready status bit

It is very important to Sync between PLC and reader for reading output results in the correct timing using a simple handshake between them.
We can use the bit ‘Data Is Ready’ for this purpose. This is bit #4 in the first byte of status register.
This bit is important and useful for properly sync between PLC and reader when reading results. This bit will be FALSE during read cycle, then TRUE when the output results from the reader are valid and ready to be read by the PLC.



Read decode results

The decode results together with other interesting results info can be found within the Cycle Report. When looking at our memory map in the device overview in TIA, we see that the Decode Cycle Report info is starting at byte #512.



Device overview

	Module	Rack	Slot	I address	Q address	Type	Art
✓	V430	0	0	2041*		V430	V4
✓	Interface	0	0 X1	2040*		V430	
✓	Input 1 Decode_1	0	1			Input 1 Decode	
✓	Module Header	0	1 24	2...5		Module Header	
✓	Device Status	0	1 25	6...9		Device Status	
✓	Fault Code	0	1 26	10...13		Fault Code	
✓	Counters	0	1 27	256...279		Counters	
✓	Read Cycle Report	0	1 28	280...287		Read Cycle Report	
✓	Decode Cycle Report	0	1 29	512...967		Decode Cycle Report	
✓	Output Premier_1	0	2			Output Premier	
!	Output Premier Comma...	0	2 48		3...6	Output Premier Co...	

The Decode Cycle report starts at byte #512. To find the decoded text string we need to skip the first 20 bytes which contains other info (see details in documentation). It means that the string is starting in the address of byte #532, while each byte represents a single character.

DECODE CYCLE REPORT	8
CODE TYPE	4
PIXELS PER ELEMENT	4
DECODE DATA LENGTH	4
DECODE DATA STRING	436

The decoded text is the characters string: '**3752214'**. See below reading result in Weblink:

%IB532:P	Character	'3'	Permanent
%IB533:P	Character	'7'	Permanent
%IB534:P	Character	'5'	Permanent
%IB535:P	Character	'2'	Permanent
%IB536:P	Character	'2'	Permanent
%IB537:P	Character	'1'	Permanent
%IB538:P	Character	'4'	Permanent
%IB539:P	Character	'$00'	Permanent
%IB540:P	Character	'$00'	Permanent

Write control data to reader

The Command Bit Field gives several useful bit commands for controlling the reader operation and status from the PLC. See some useful commands below:

1. Bit 0 - Online / Offline command. Use it to put the reader to online mode (default) or offline mode (ignoring triggers).
2. Bit 1 - Trigger bit – use this bit to trigger the reader
3. Bit 11, 11, 12 – Use these bits to set / reset the status of discrete outputs 1, 2, 3.

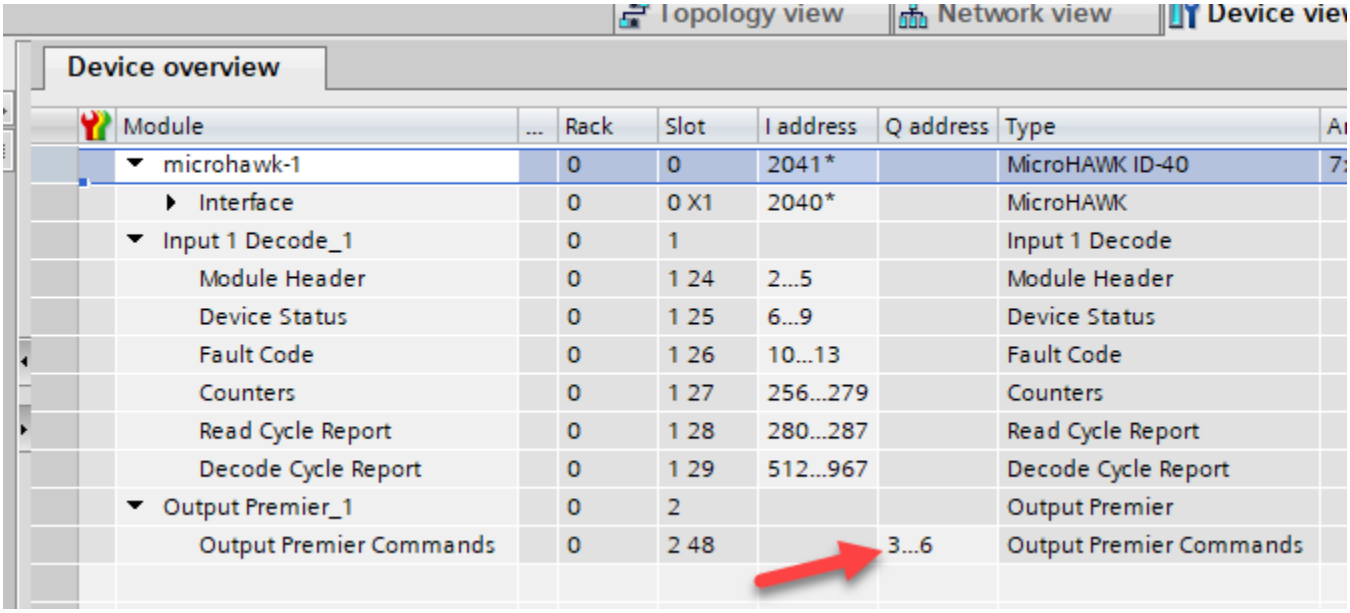
See below the full list of command bits. More details can be found in the documentation.

3.6.2.2.1.1 Command Bit Field

BIT FIELD	COMMAND
0	Run Mode
1	Trigger
2	Enable MatchCode
3	Reset General Fault
4	Clear No Read ReadCycle Count
5	Clear MisMatch ReadCycle Count
6	Clear No Read Count
7	Clear Trigger Count
8	Clear Matchcode Count
9	Clear Mismatch Count
10	Output_1
11	Output_2
12	Output_3
13-31	Reserved for future use

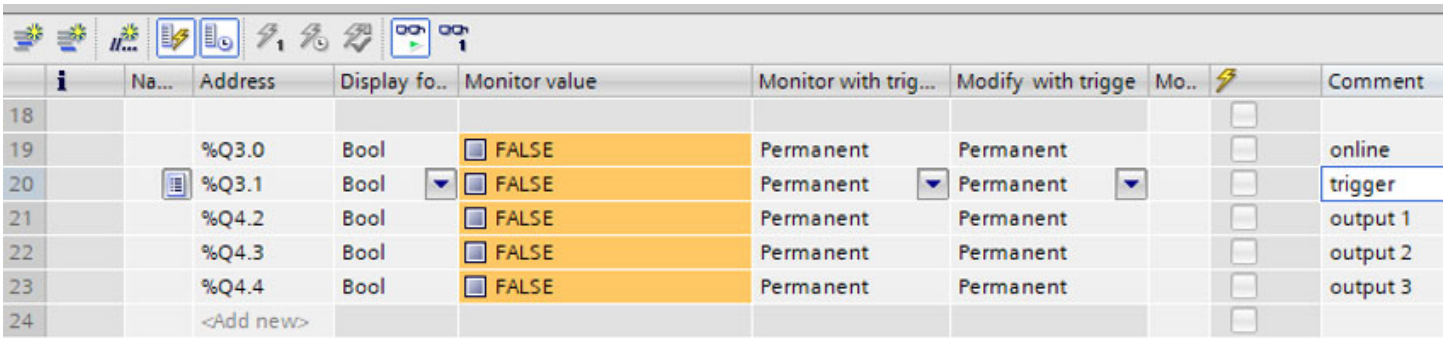
Triggering the device from the PLC over Profinet:

To find the correct address of the trigger bit command, we look in the memory map for 'output premier commands':



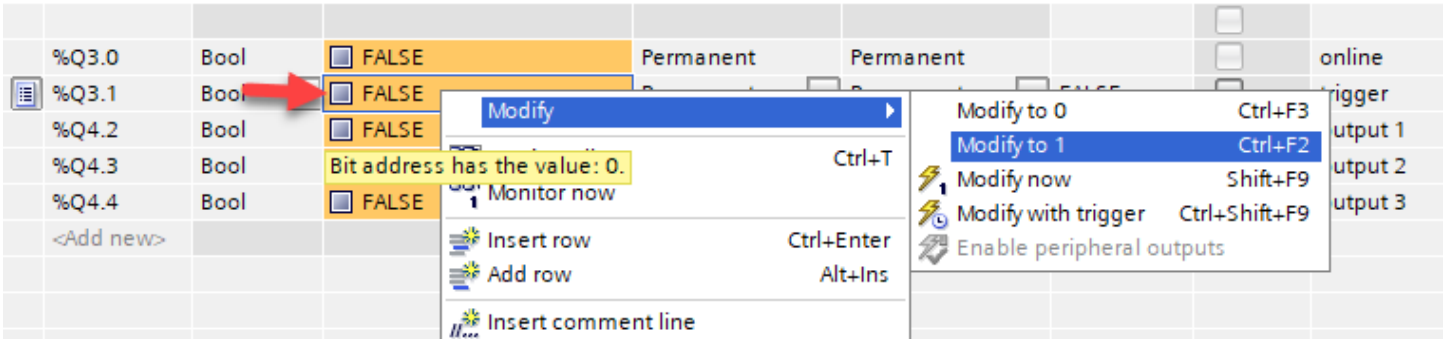
Module	...	Rack	Slot	I address	Q address	Type	A
microhawk-1		0	0	2041*		MicroHAWK ID-40	7
Interface		0	0 X1	2040*		MicroHAWK	
Input 1 Decode_1		0	1			Input 1 Decode	
Module Header		0	1 24	2...5		Module Header	
Device Status		0	1 25	6...9		Device Status	
Fault Code		0	1 26	10...13		Fault Code	
Counters		0	1 27	256...279		Counters	
Read Cycle Report		0	1 28	280...287		Read Cycle Report	
Decode Cycle Report		0	1 29	512...967		Decode Cycle Report	
Output Premier_1		0	2			Output Premier	
Output Premier Commands		0	2 48		3...6	Output Premier Commands	

We see from the memory map table that the relevant memory area for control commands is from byte 3 to byte 6. See below example definition of a few useful bit commands:

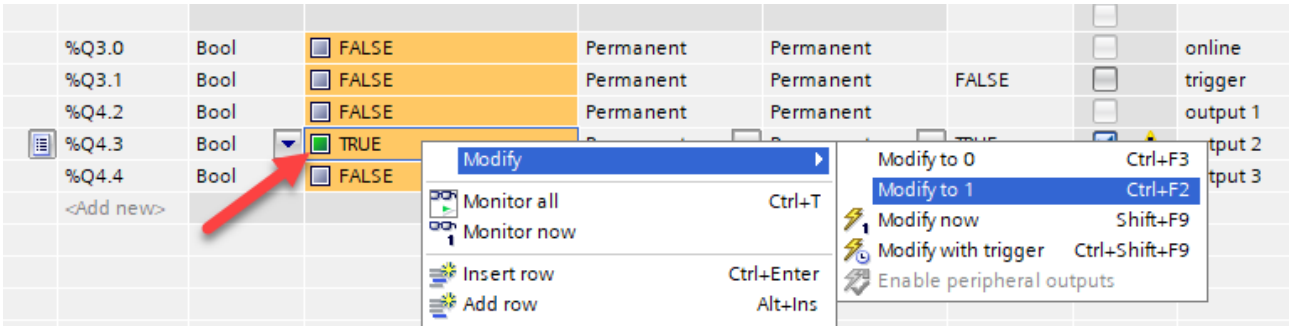


	Na...	Address	Display fo..	Monitor value	Monitor with trig...	Modify with trigge	Mo..	Comment
18								
19		%Q3.0	Bool	FALSE	Permanent	Permanent		online
20		%Q3.1	Bool	FALSE	Permanent	Permanent		trigger
21		%Q4.2	Bool	FALSE	Permanent	Permanent		output 1
22		%Q4.3	Bool	FALSE	Permanent	Permanent		output 2
23		%Q4.4	Bool	FALSE	Permanent	Permanent		output 3
24		<Add new>						

To test the trigger command from the PLC, force on the trigger bit command TRUE, then back to FALSE. See that the reader is triggered, view read results and status bits to verify that the values are as expected.



Set status of reader's discrete outputs via Profinet:



When looking at a light tower connected to output 2, we will see that this reader output is energized.
(This is not such a common feature, but brought here as example of sending control data to reader over Profinet).

Send match data from PLC to reader via TCP:

Sending new matchcode from PLC to reader cannot be done via Explicit messages as done in Ethernet/IP communication, as Profinet is not supporting this feature. Therefore, we will show in this section how to do it over TCP from a Siemens PLC using TIA15.1.

K command for sending new matchcode to the reader

- Sending new matchcode to the reader is done using the following K command:

<K231, 1, 12345678>
This command sends the following new matchcode: 12345678

- Requesting the current matchcode from the reader is done using the following K command:

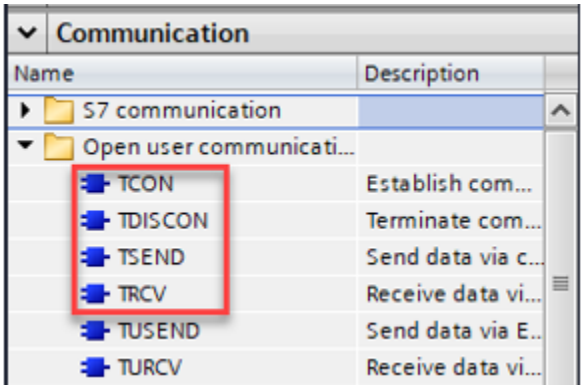
<K231?>
Then the reader will reply with the following data:
<K231, 1, 12345678>
Which means that the current matchcode saved on the reader is: 12345678.

Sending a new matchcode is recommended to do in the following way, combining these two K commands together in one string: **<K231, 1, 12345678><K231?>**
This way we can force the reader to send immediate reply which can help us to verify two things at once:
1. That the command was accepted OK
2. That the change is done.

Creating TCP client in TIA

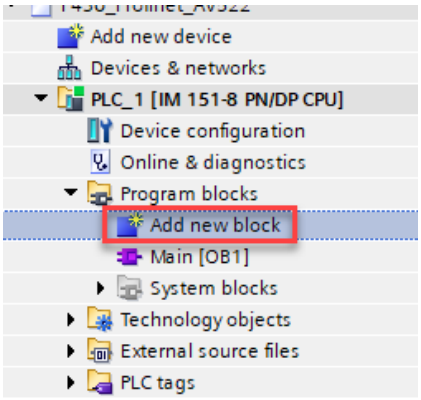
For TCP communication in TIA, there are dedicated FBs provided by default:

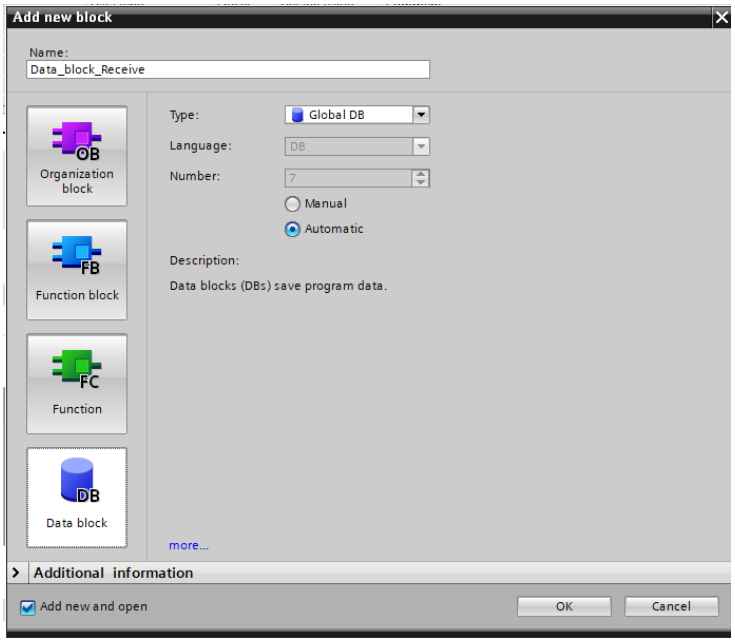
FB name	FB functionality
TCON	Connect
TDISCON	Disconnect
TSEND	Send data
TRCV	Receive data



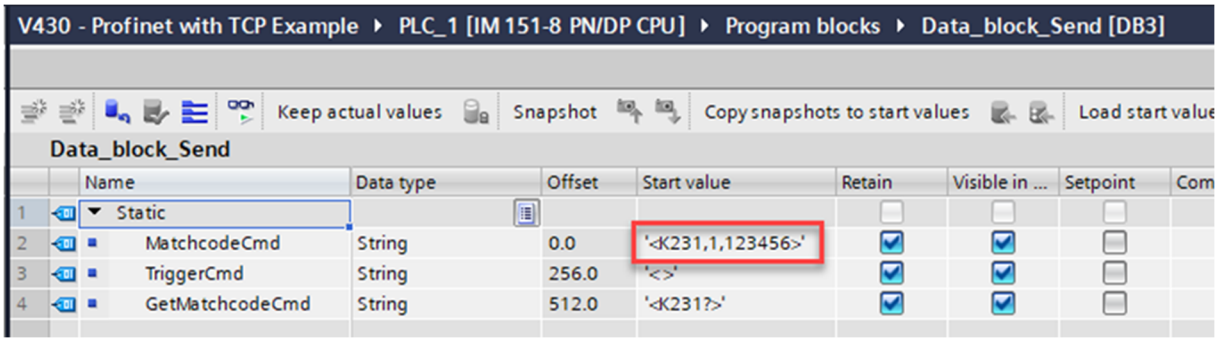
Please follow the steps below to create TCP client inside your TIA project:

1. To keep the string of the K command that we want to send, add **new data block**:

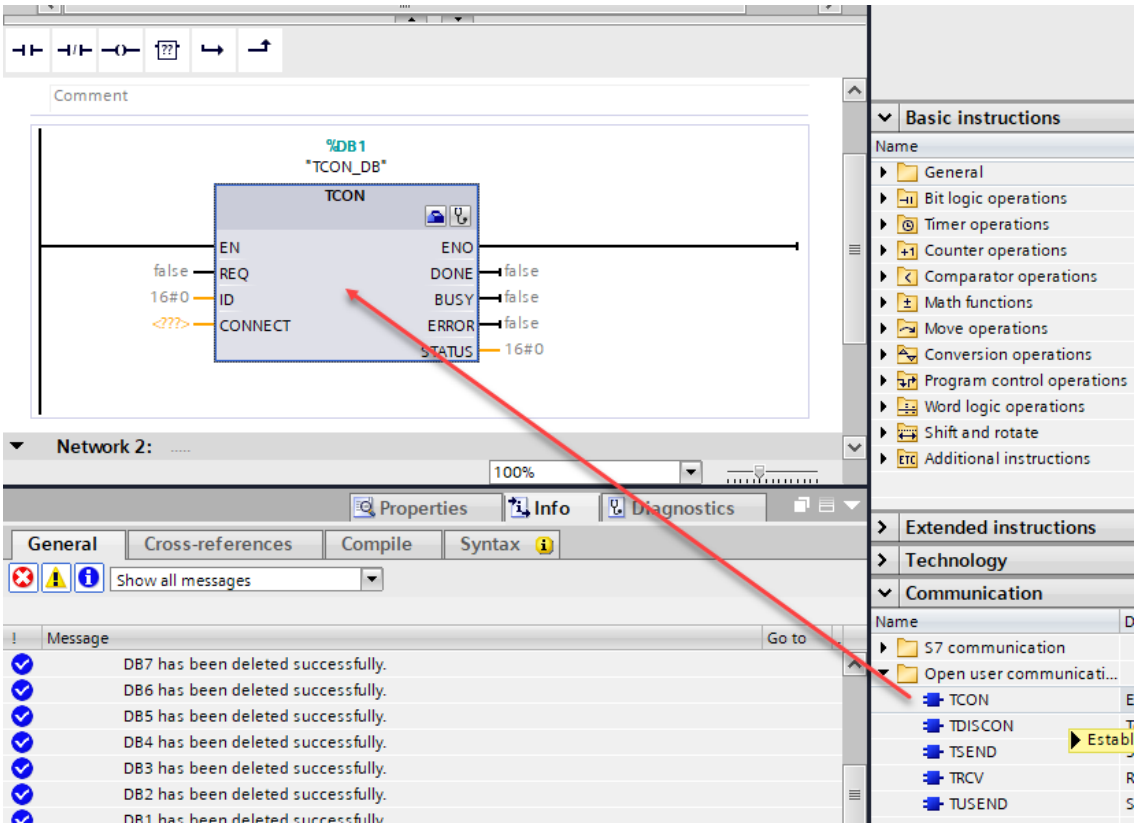




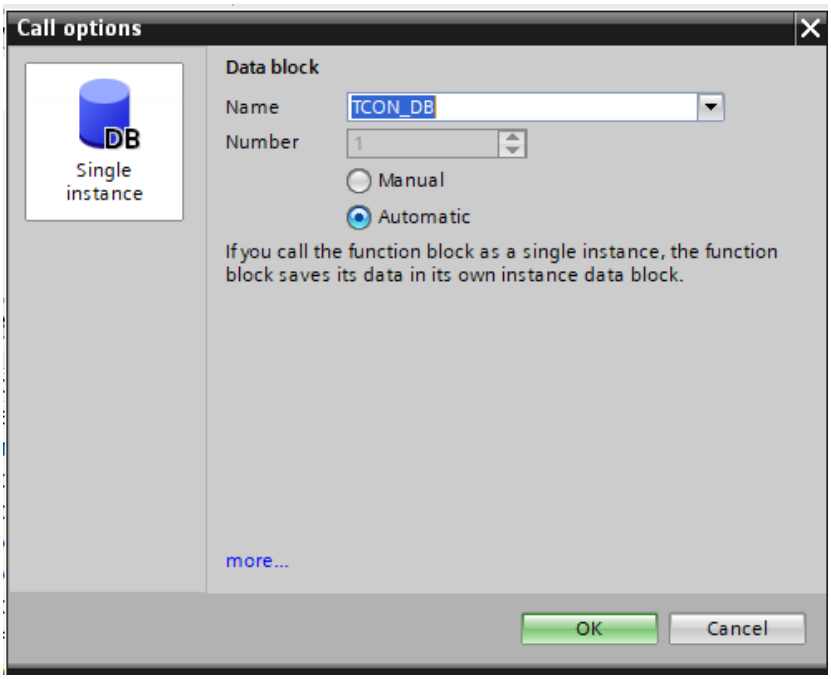
2. Add the relevant K commands inside the data block as string parameters:



3. Add TCON FB into Network 1 for creating the TCP connection.
Drag and drop from the FB catalog, then click OK:

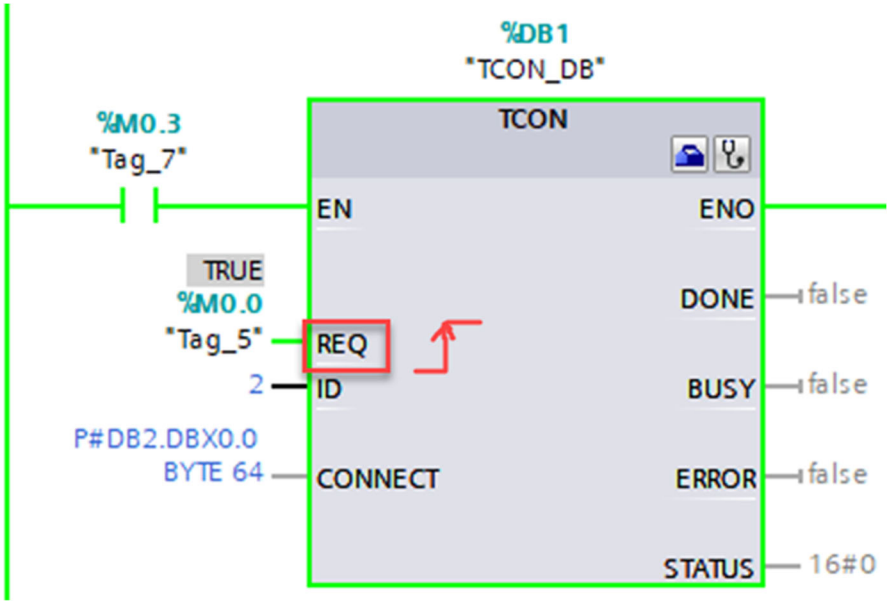


TCON DB will be created and connected automatically to the FB. Type a name and click OK:



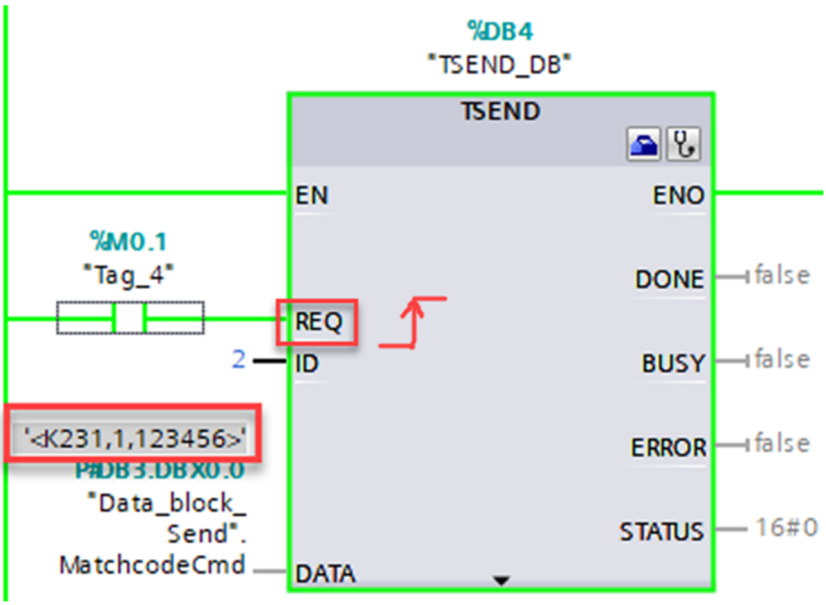
TCP client connect

4. Now when the **TCON** FB is added and setup, in order to connect to the reader, enable the FB and set **REQ** input from FALSE to TRUE:



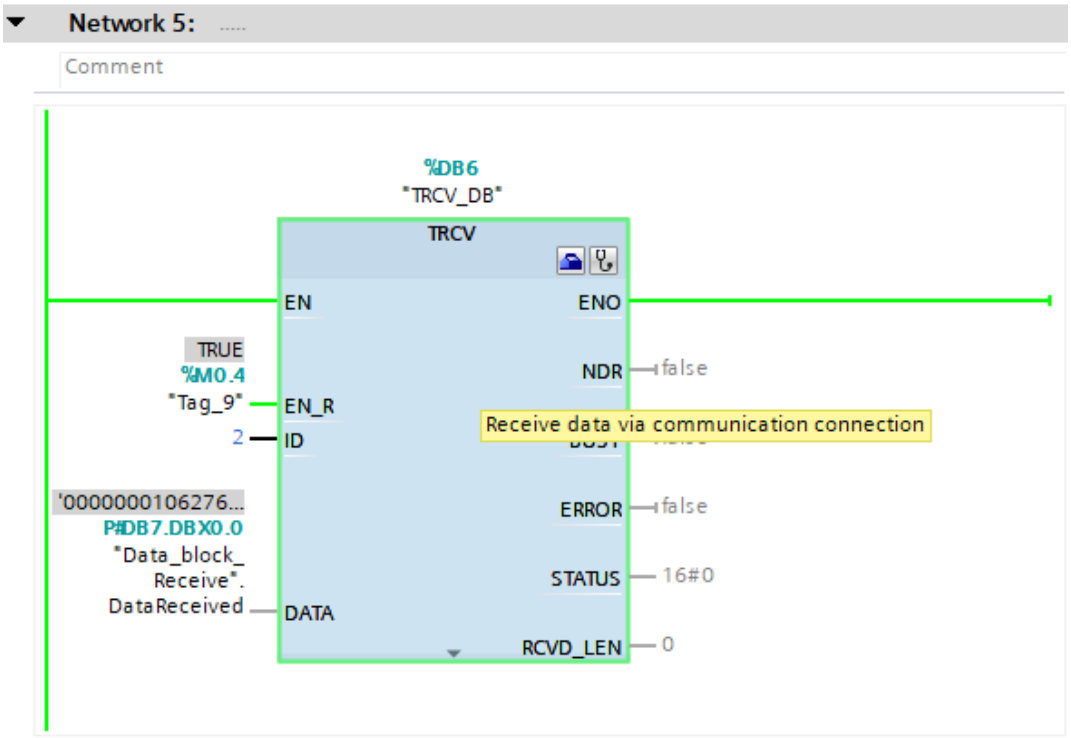
TCP send matchcode

To send the new matchcode K command, use **TSEND** FB. After adding to your program, connect the data block we created in the first step (specify the string parameter containing the K command) to the input **DATA**, enable the FB, then set **REQ** input from FALSE to TRUE.

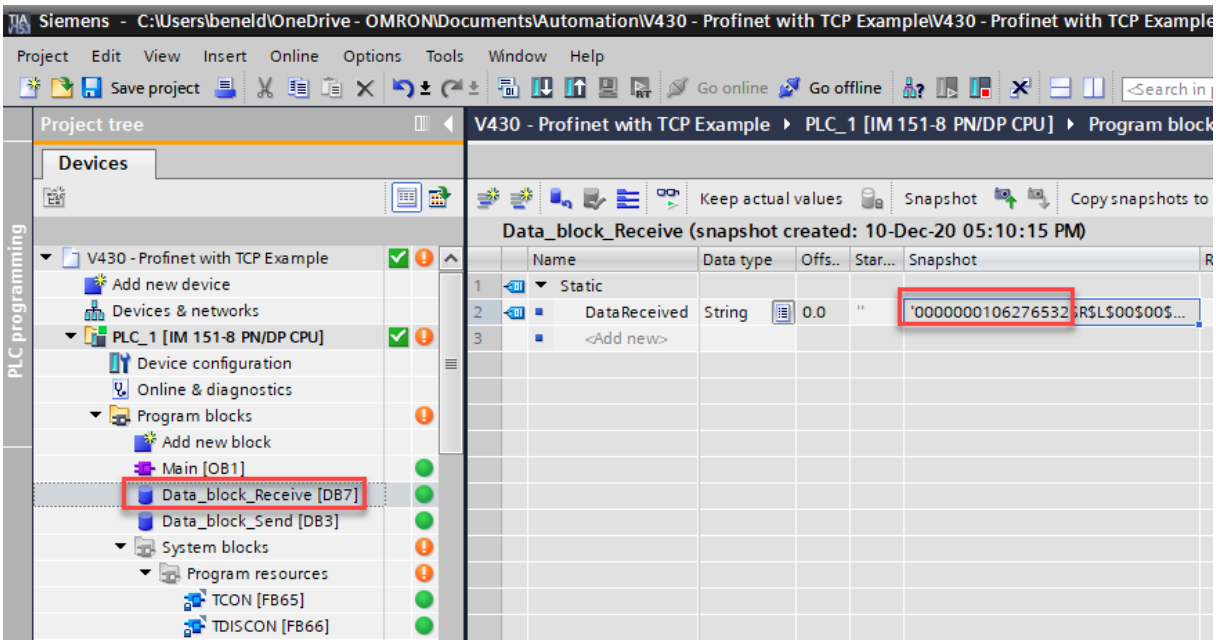


TCP receive data

Receiving data might be needed for receiving the reply for matchcode request <K231?> when sent, or for reading decode results. To receive data from the reader, Add **TRCV** FB, connect it to a data block with a string parameter inside for the received data, then set the input EN_R from FALS to TRUE to read the data which is sent from the reader.

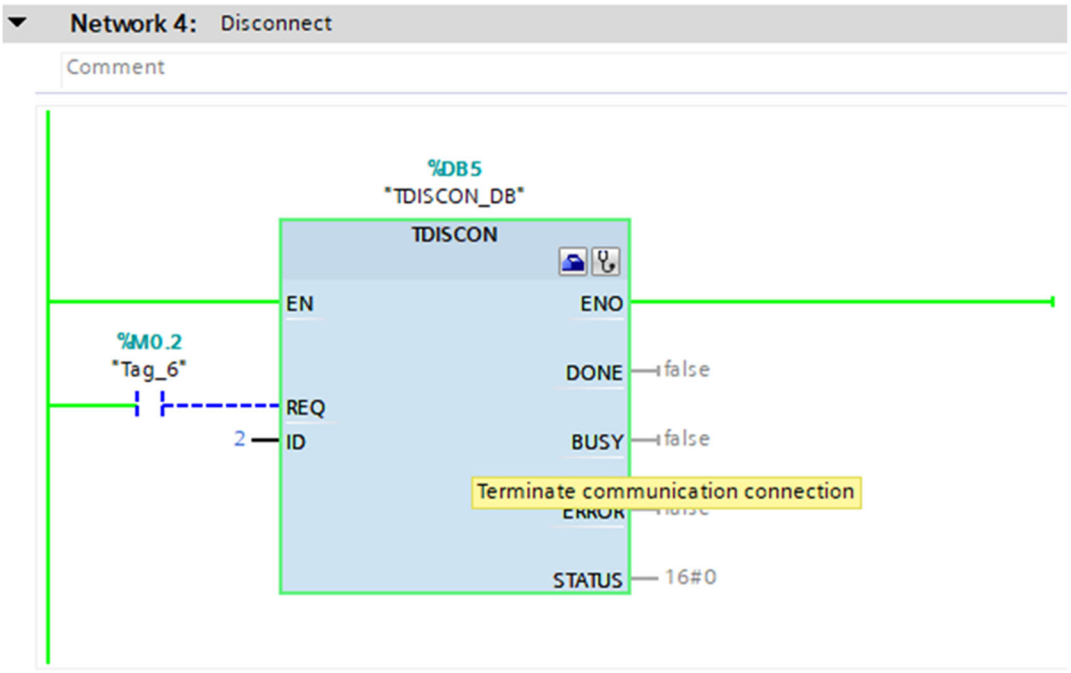


The received string can be viewed inside the receive data block:



TCP client disconnect

To disconnect the TCP client from the reader, use **TDISCON** FB. Add it to the ladder program, enable the FB and set **REQ** input from FALSE to TRUE to disconnect:



User data types (UDT files)

What are UDT files

UDT files are pre-defined data structures which are relevant for a specific Profinet device. When connected over Profinet, the PLC can read and write data according to specific structure which must be defined in the PLC project. Once the data type files are added properly to the project, the PLC programmer can then use those pre-defined data types in the programming.
It is not mandatory to use UDT files for communication with V430 from the PLC. The UDT files provide the PLC designer with tools to create efficient and readable code especially in cases of many readers connected to a single PLC. The UDT files helps to create a more organized and readable program, saving design time and also modifications and debug time later.

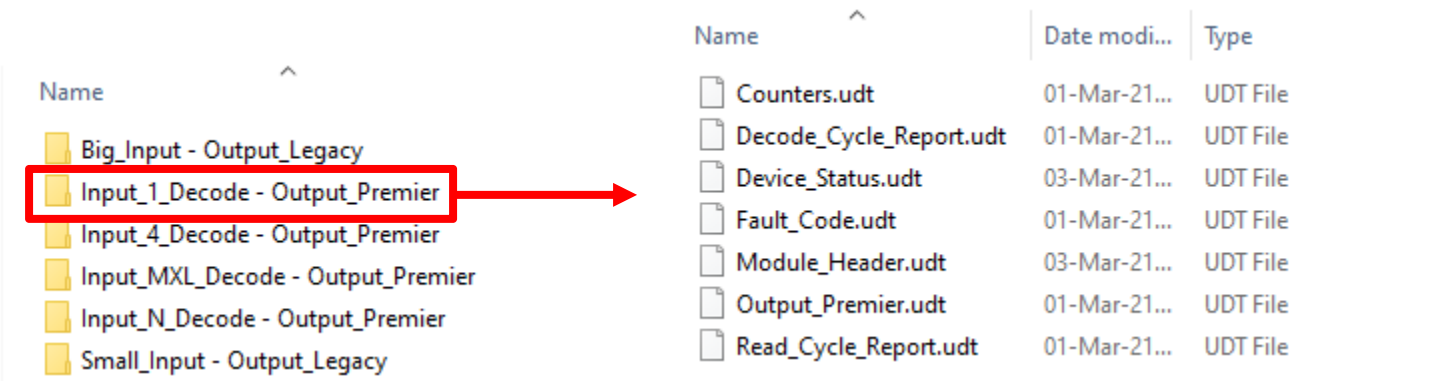
UDT files for V430

Like many other providers of Profinet I/O sensors, OMRON delivers a set of UDT files together with the V430 code reader which are developed with TIA 15.1 and S7-1200 PLC. For every component in the Input or Output assembly, there is a UDT file which can be added and used inside the PLC project. Each one of the provided UDT files corresponds to the module definitions of the GSD file. This division is done for giving the user freedom with memory addressing.

In this chapter we will show step by step tutorial on how to import the UDT files to a TIA project and then show simple examples of how to use them within the project (Defining new PLC Tags and read/write data). For this chapter S7-1200 PLC was used with V430 FW 2.1.

How to add UDT file to TIA project

1. Download and save the provided UDT files on your local PC.
The provided package looks like this:



There are 6 separate sets of UDT file, which corresponds to 6 valid combinations of Input⇔Output assembly pairs:

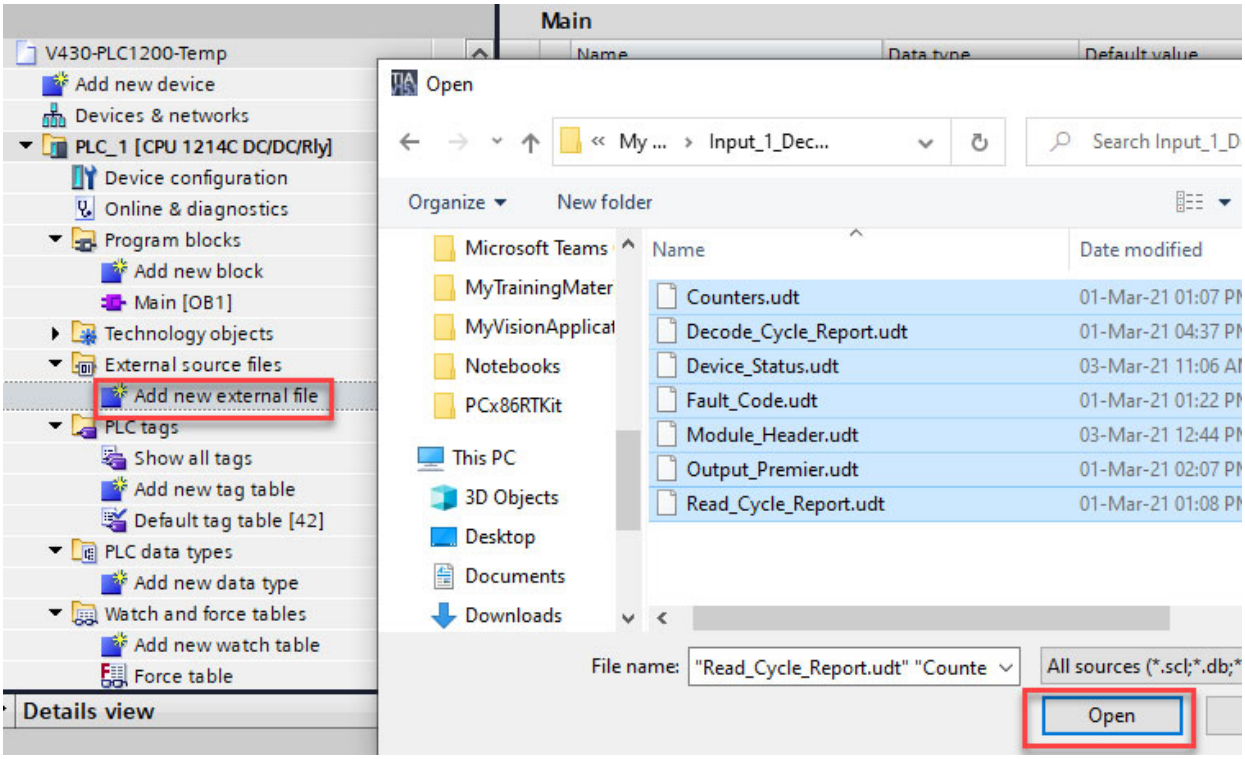
Possible combinations	Input Assembly name	ID Number	Output Assembly name	ID Number
1	Small Input	100	Output legacy	198
2	Big Input	101	Output legacy	198
3	MXL/SLC	102	Output premier	197
4	Input 1 decode	103	Output premier	197
5	Input 4 decode	104	Output premier	197
6	Input N decode	105	Output premier	197

For any V430 which is added to a PLC project, we can use only one of the 6 pairs above. Then we need to use only one of the folders with the corresponding UDT files.

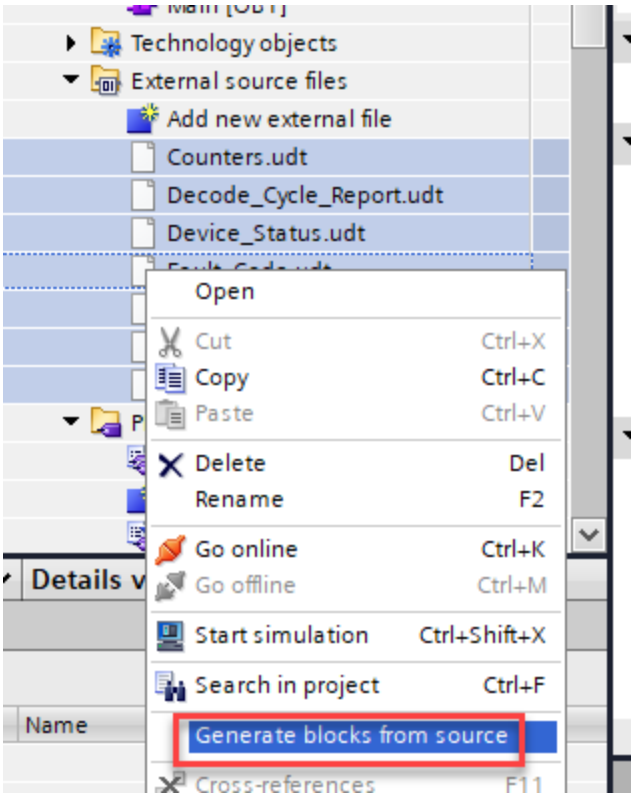
Example - Import UDT files to TIA project

In this example we will use the UDT files for this pair: **Input_1_decode Assembly and Output_Premier assembly**.

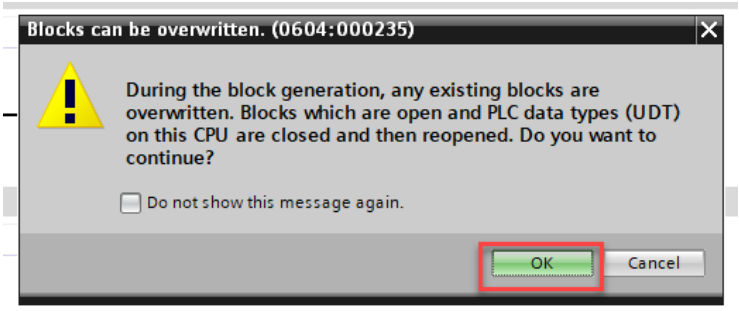
1. In TIA, import the relevant UDT files to the project as external source files: double click ‘add external files’, then browse to the downloaded UDT files which are saved on the PC:



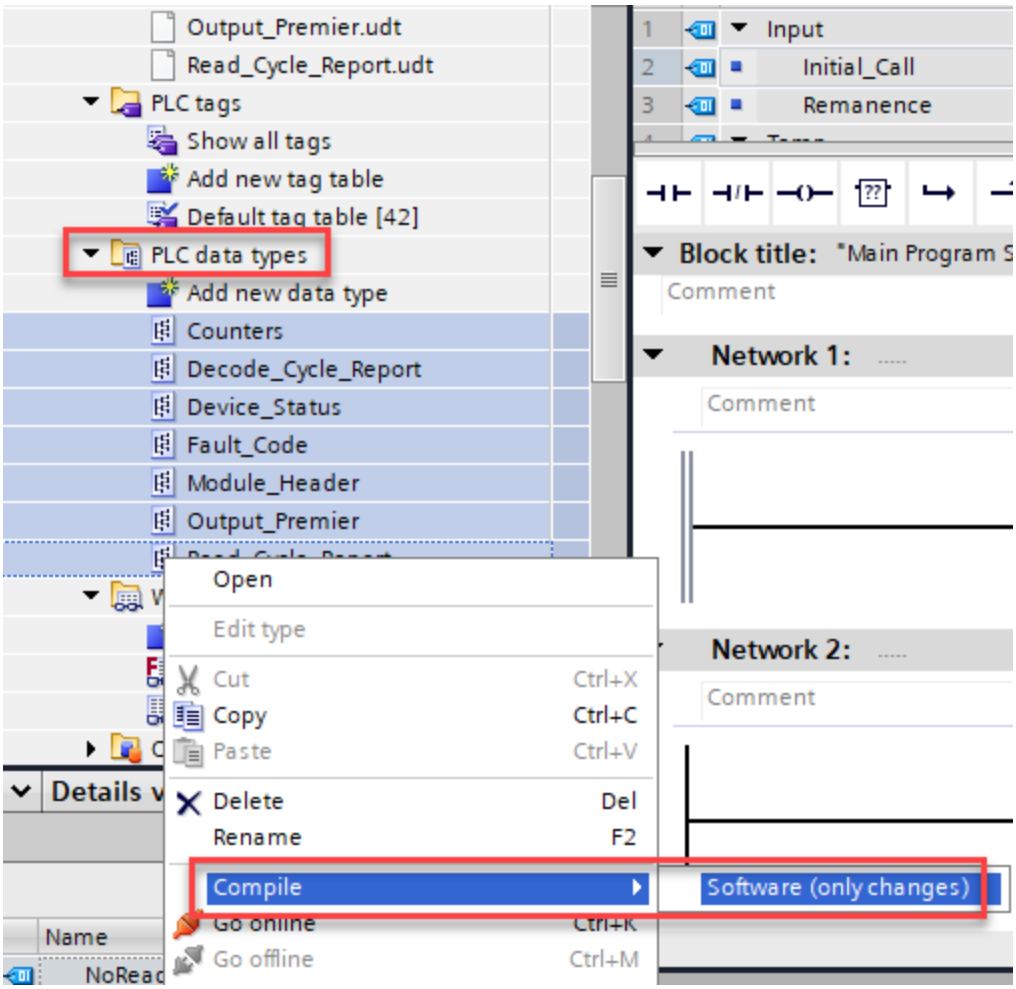
2. To generate new data types from the imported files, select the external files which are just added to TIA, then right mouse -> 'Generate blocks from source'.



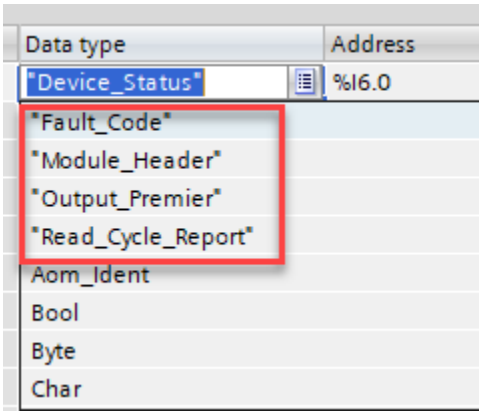
If you get this warning, click OK:



3. In this stage, a new set of data types will be added under 'PLC data types'. In order to be able to use them in the project, we need first to compile them. Select the new data types, then right mouse -> Compile:



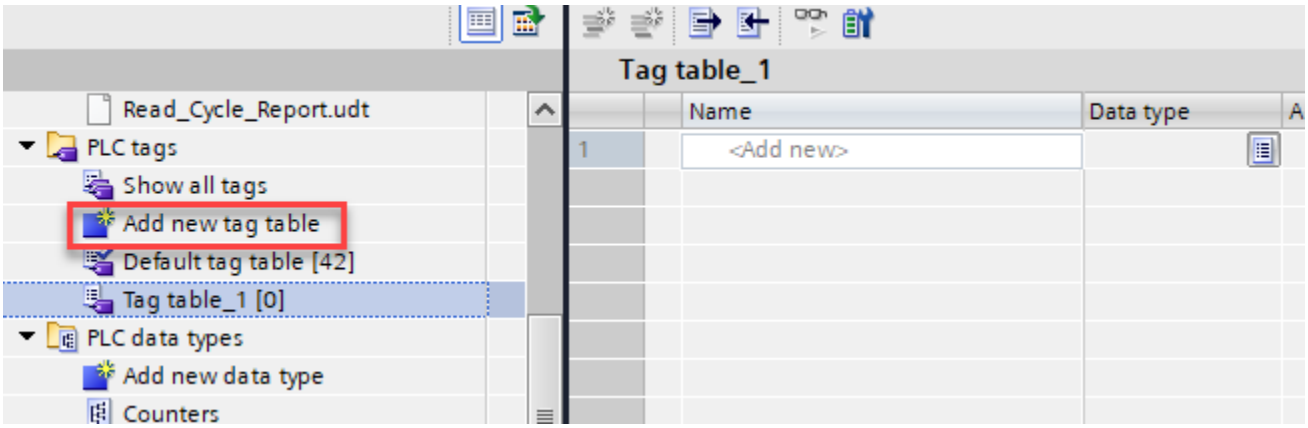
Now anywhere in the project where we want to define a parameter type, we will have the new data types as selection option, together with all other default data types like: bool, char…



Example – Define PLC tags using the new data types

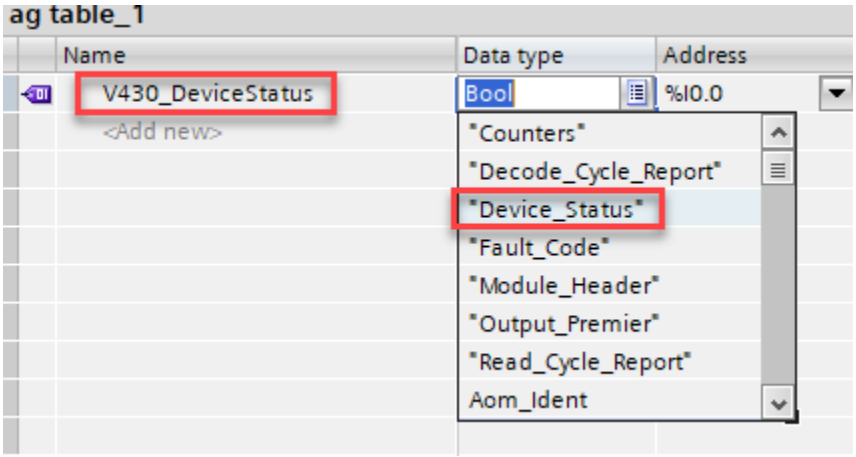
There are many ways to use imported data types in a project. A simple way is to define new PLC tags using the existing data types. This way we will have a quick access to any data item we want inside the V430 input and output assembly.

1. Create a new PLC tags table by double click on ‘Add new tag table’:

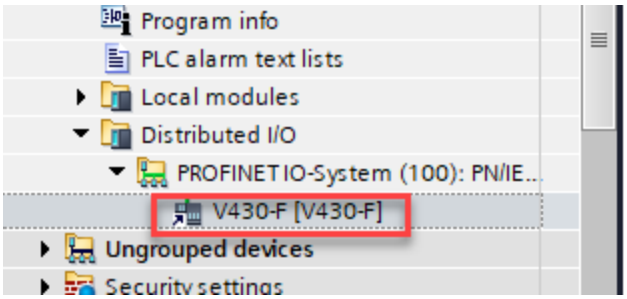


Now we are ready to define the relevant PLC tags for the V430 modules which we want to use in the project.

2. In our case we will want to read the status of the V430, so we need to add a new tag name ‘V430_DeviceStatus’ with data type ‘Device_Status’ which we already have after importing the UDT files:



3. Now we need to provide the correct start address for this module. This info we can find in the memory map. To view the memory map, double click on the V430 under Profinet I/O devices:

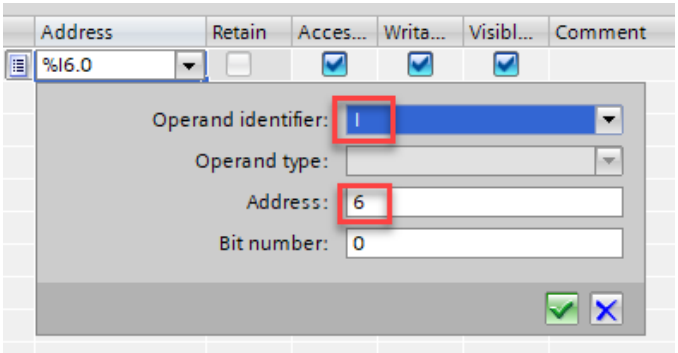


See our memory map in the screenshot below. It provides the memory addresses of all modules which we chose to use in this project. The 'I' column shows the Input modules and 'Q' column shows the Output modules. For every module there are always 2 given numbers: Start address and End address. We need to use the start address in our definition of new tag in TIA.

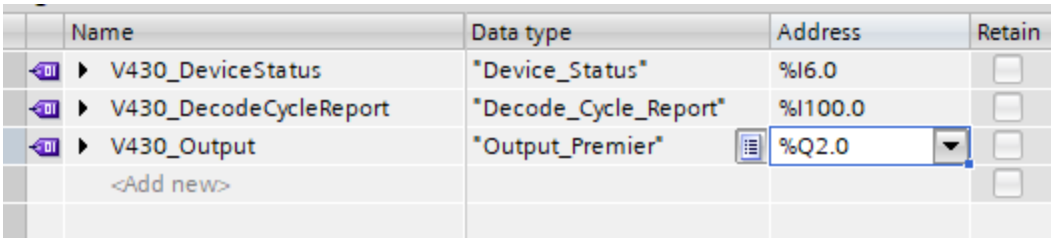
Device overview

...	Module	Rack	Slot	I address	Q address	Type	A.
	V430-F	0	0			V430-F	V.
	Interface	0	0 X1			V430-F	
	Input 1 Decode_1	0	1			Input 1 Decode	
	Module Header	0	1 24	2...5		Module Header	
	Device Status	0	1 25	6...9		Device Status	
	Fault Code	0	1 26	10...13		Fault Code	
	Counters	0	1 27	68...91		Counters	
	Read Cycle Report	0	1 28	92...99		Read Cycle Report	
	Decode Cycle Report	0	1 29	100...555		Decode Cycle Report	
	Output Premier_1	0	2			Output Premier	
	Output Premier Comma...	0	2 48		2...5	Output Premier Commands	

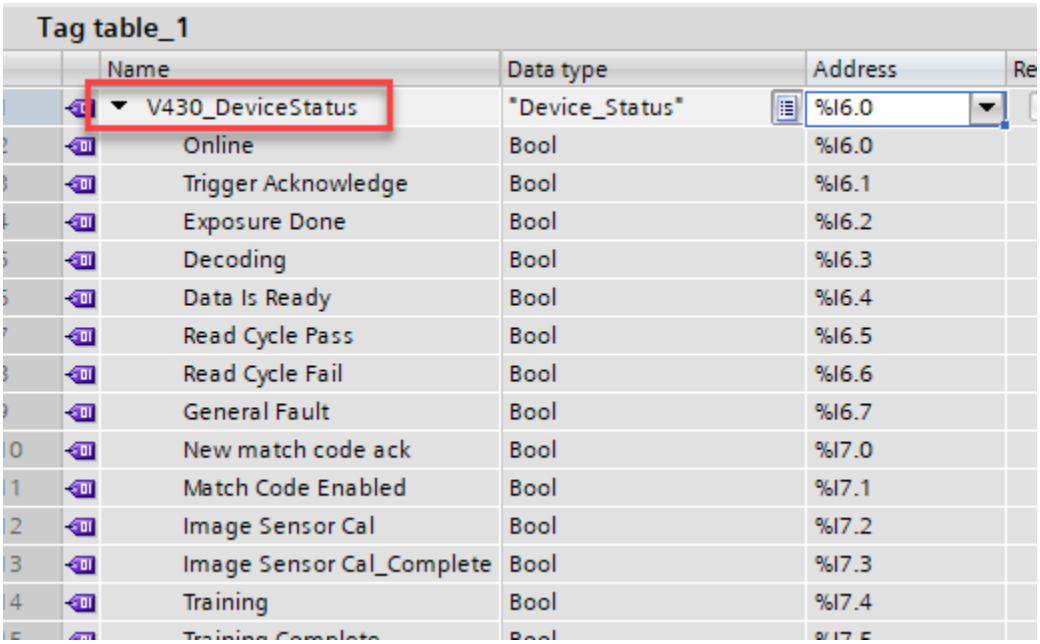
For example, for completing the definition of our new tag for DeviceStatus Input module, we need to provide the correct start address for this tag, which is 6 according to the memory map. As identifier we select I, as this is input data coming from the V430. Then we click OK:



In the same way we define the rest of the PLC tags we need in our project:

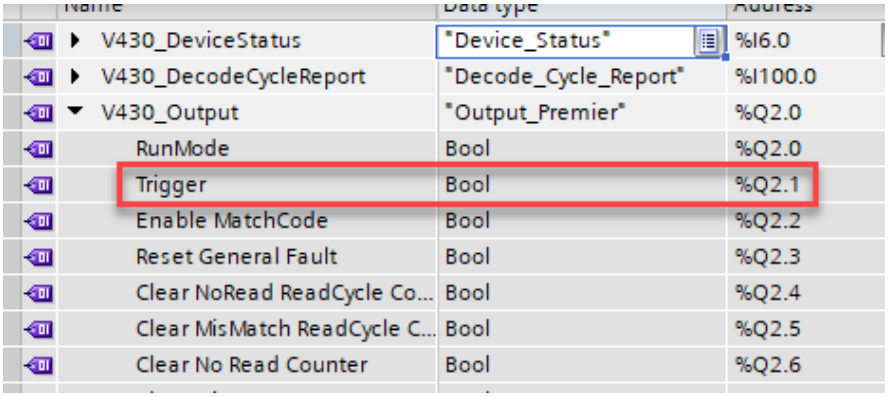


4. After compiling everything and downloading to PLC, we can expand these tags and see all bit fields pre-defined with the correct addresses for us to use in our program:



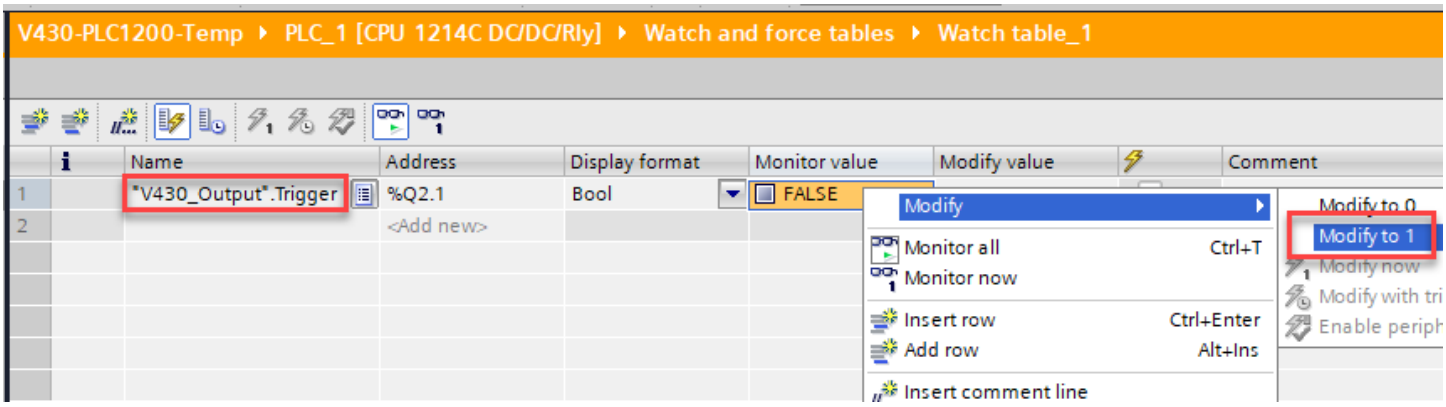
Example – Trigger the reader using our new output PLC tags

Once we define the output module as a PLC tag following the instructions in the previous section, we will have easy access to all the bit fields inside the 4 bytes of bit commands. To trigger the reader, all we need is to provide a Pulse on 'Trigger' bit, which is already defined for us with the correct address:

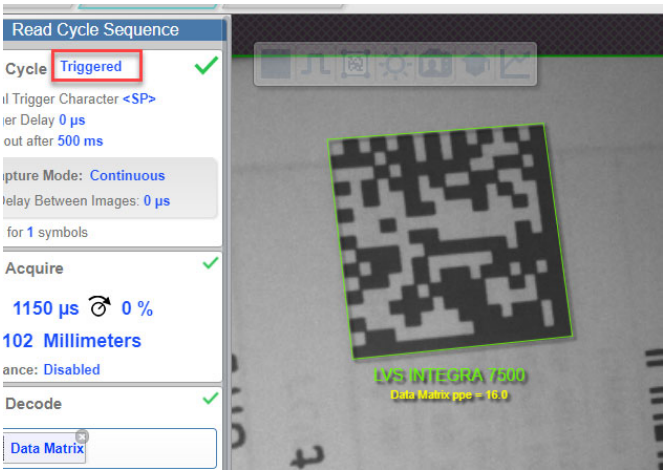


Name	Data type	Address
V430_DeviceStatus	"Device_Status"	%I6.0
V430_DecodeCycleReport	"Decode_Cycle_Report"	%I100.0
V430_Output	"Output_Premier"	%Q2.0
RunMode	Bool	%Q2.0
Trigger	Bool	%Q2.1
Enable MatchCode	Bool	%Q2.2
Reset General Fault	Bool	%Q2.3
Clear NoRead ReadCycle Co...	Bool	%Q2.4
Clear MisMatch ReadCycle C...	Bool	%Q2.5
Clear No Read Counter	Bool	%Q2.6

To test this, we can use watch window, selecting this bit as a view item, then forcing to HIGH. Doing this should trigger the reader immediately:



*Reader must be running, online and in triggered mode:



Omron Europe B.V.
Wegalaan 67-69
2132 JD Hoofddorp

Example – read the decoded text using our new PLC tags

To view the decoded text string, we need to define a PLC tag for the input module 'DecodeCycleReport'. We did this in the previous sections:

Name	Data type	Address	Retain
V430_DeviceStatus	"Device_Status"	%I6.0	
V430_DecodeCycleReport	"Decode_Cycle_Report"	%I100.0	
V430_Output	"Output_Premier"	%Q2.0	
<Add new>			

So now we can simply view this tag content in the tags window or watch window. To read the decoded text string, we need go online, then expand the tag structure to reach the Decode_String char array:

Tag table_1

Name	Data type	Address	Re..	Acces...	Writa...	Visibl...	Monitor value
V430_DeviceStatus	"Device_Status"	%I6.0		☑	☑	☑	
V430_DecodeCycleReport	"Decode_Cycle_Report"	%I100.0		☑	☑	☑	
Decode_Bounding_Box_Top	Int	%IW100		☑	☑	☑	20224
Decode_Bounding_Box_Left	Int	%IW102		☑	☑	☑	11778
Decode_Bounding_Box_Hei...	Int	%IW104		☑	☑	☑	5377
Decode_Bounding_Box_Wi...	Int	%IW106		☑	☑	☑	8193
Code_Type	Struct	%I108.0		☑	☑	☑	
Pixels_Per_Element	Real	%ID112		☑	☑	☑	16#0000_8041
Decode_Data_Length	DInt	%ID116		☑	☑	☑	268_435_456
Decode_String	Array[0..435] of Char	%I120.0		☑	☑	☑	
Decode_String[0]	Char	%IB120		☑	☑	☑	'L'
Decode_String[1]	Char	%IB121		☑	☑	☑	'V'
Decode_String[2]	Char	%IB122		☑	☑	☑	'S'
Decode_String[3]	Char	%IB123		☑	☑	☑	' '
Decode_String[4]	Char	%IB124		☑	☑	☑	'I'
Decode_String[5]	Char	%IB125		☑	☑	☑	'N'
Decode_String[6]	Char	%IB126		☑	☑	☑	'T'
Decode_String[7]	Char	%IB127		☑	☑	☑	'E'
Decode_String[8]	Char	%IB128		☑	☑	☑	'G'
Decode_String[9]	Char	%IB129		☑	☑	☑	'R'
Decode_String[10]	Char	%IB130		☑	☑	☑	'A'
Decode_String[11]	Char	%IB131		☑	☑	☑	' '

*Note that in order to read properly the counter values above, as well as other numeric values, **Byte Swap** feature must be enabled in reader's settings.