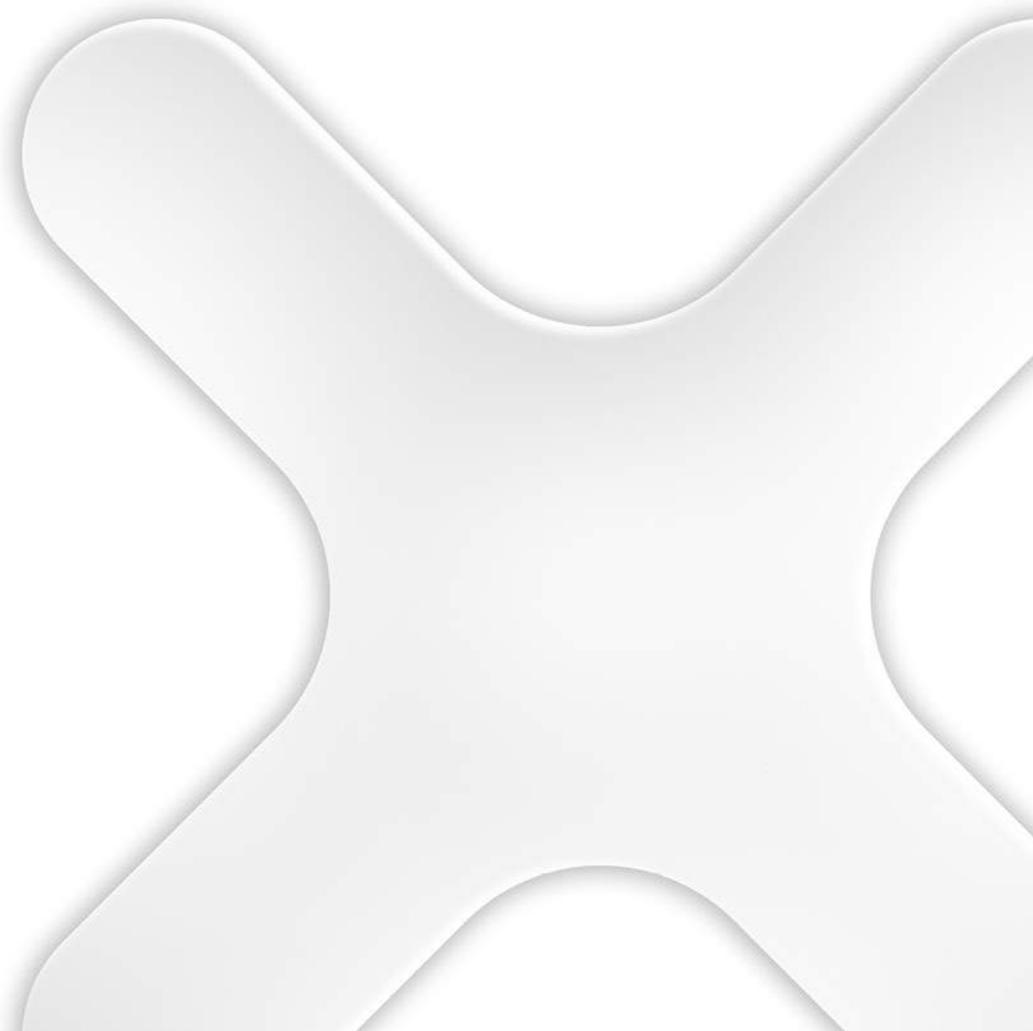
Communication Protocols

User Manual



Version

© 2010-2022 Exor International S.p.A.

Disclaimer

© 2010-2022 Exor International S.p.A.

Subject to change without notice

The information contained in this document is provided for informational purposes only. While efforts were made to verify the accuracy of the information contained in this documentation, it is provided 'as is' without warranty of any kind.

- The copyright of this manual is owned by Exor International S.p.A.
- Unauthorized reproduction of this manual is strictly prohibited.
- Windows is a registered trademark of Microsoft Corporation in the U.S. and other countries.
- Ethernet is a registered trademark of FUJIFILM Business Innovation Co., Ltd. and Xerox Corporation.
- QR Code is a registered trademark of DENSO WAVE INCORPORATED.
- Other company and product names are trademarks or registered trademarks of their respective companies.

Third-party brands and names are the property of their respective owners.

The example companies, organizations, products, domain names, e-mail addresses, logo, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred.

This products/software contains software licensed under the GNU General Public License, Version 2.0 (GPL V2.0), software licensed under the GNU LESSER General Public License, Version 2.1 (LGPL V2.1), and/or open source software other than the software licensed under the GPL V2.0 and/or LGPL V2.1. The software open source included is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

For more details or for a copy of sources where required by license , as well as the respective copyright notice, please ref to device settings menu or https://www.exorint.com or contact us at support.it@exorint.com

Contents

A-B DF1 3
A-B DH-485 15
A-B ENET 28
ABB Mint Controller HCP 40
BACnet 47
Beckhoff ADS 89
CAN Direct v2.0x 104
CANopen HMI 114
CANopen SDO 121
Client System Variables 127
CODESYS V2 ETH 129
CODESYS V2 SER 142
CODESYS V3 ETH 151
Control Techniques Modbus TCP 162
CT Modbus CMP ETH 166
Delta Modbus RTU 172
Direct Serial 182
Direct Socket 190
DMX512 Digital Multiplex 201
Eaton Suconet-K 206
Environment Variables 211
Ethernet/IP CIP 212
Fatek FACON ETH 237
Fatek FACON SER 244
GE Intelligent Platforms SNP 250
GE Intelligent Platforms SRTP 261
GE SRTP 271
Hitachi SER 282
Hitachi ETH 287
IDEC Maintenance 292
J1939 303
Jetter Ext ETH 315
Keyence KV 323
Koyo DL 333
Koyo DL ETH 339
KNX TP/IP 344
Lenze CANopen 364
Modbus RTU 368
Modbus RTU Server 383
Modbus TCP 398
Modbus TCP Server 415
Mitsubishi FX ETH 426
Mitsubishi FX SER 440
Mitsubishi iQ/Q/L ETH 449
Mitsubishi iQ/Q/L SER 462
NMEA 0183 471
NMEA 2000 495
Omron FINS ETH 509
Omron FINS SER 522
OPC UA Client 531
Panasonic FP 545
Ping 552
ProConOS ETH 557
Profibus DP 565
Profibus DP S7 569
Rexroth IndraControl 614
ROBOX BCC/31 620
ROC Plus 629
SAIA S-BUS 635
SAIA S-BUS ETH 644

Simatic S7 PPI 651
Siemens S7 Optimized 658
Simatic S7 ETH 673
Simatic S7 MPI 712
System Variables 747
Uni-Telway 749
Variables 757

A-B DF1

The A-B DF1 communication driver has been designed to connect HMI devices to a Allen-Bradley controllers through serial communication.

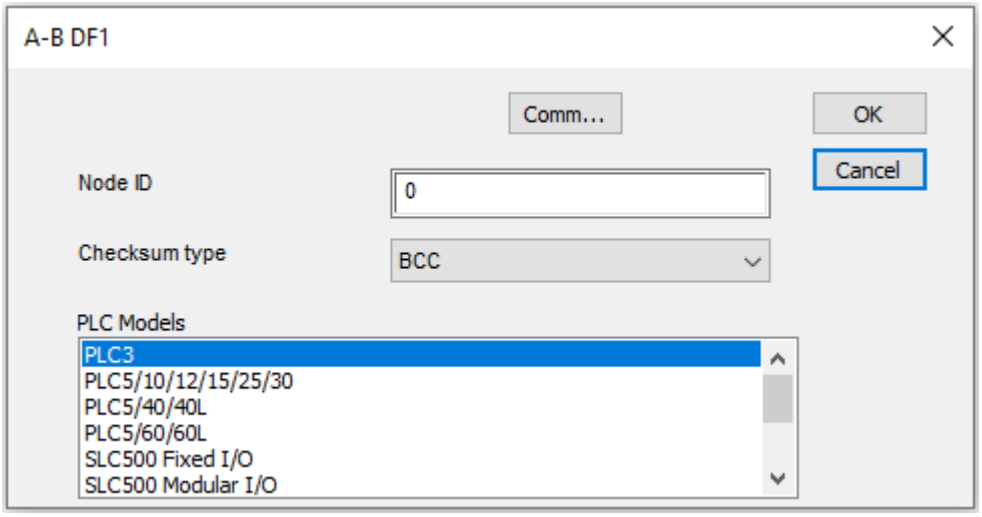
Protocol Editor Settings

Adding a protocol

To configure the protocol:

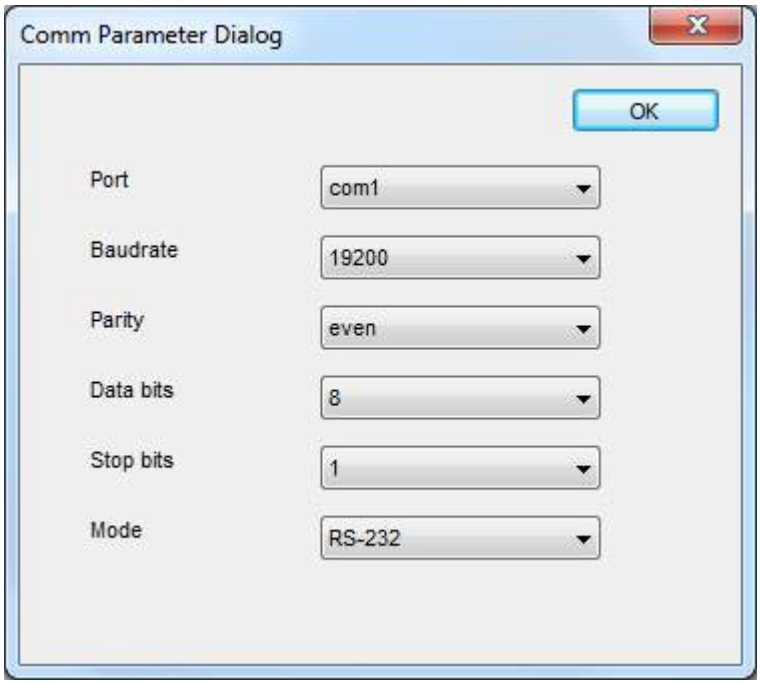
1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Node ID	Serial node associated to the PLC.
Checksum type	It can be **BCC** or **CRC**, depending on PLC settings.
PLC Models	PLC models available: • PLC3 • PLC5/10/12/15/25 • PLC5/40/40L • PLC5/60/60L • SLC500 Fixed I/O

Element	Description
	• SLC500 Modular I/O • Micrologix 1000 • Micrologix 1500 • Ultra5000
Comm...	If clicked displays the communication parameters setup dialog.



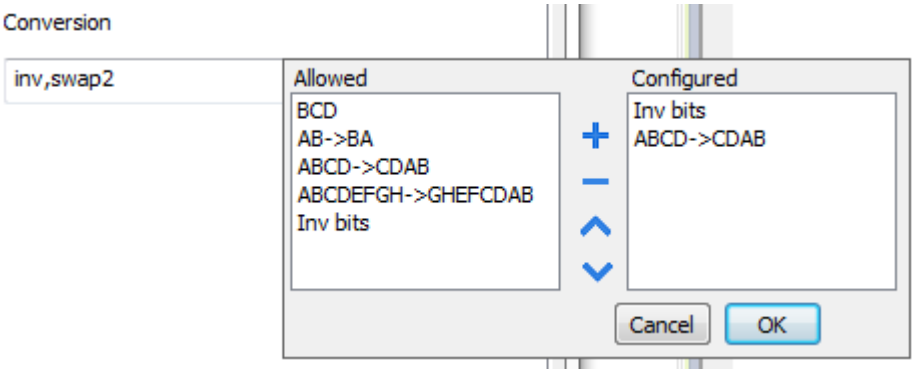
Element	Parameter
Port	Serial port selection. • **COM1**: device PLC port. • **COM2**: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board. • **COM3:** optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.
Mode	Serial port mode. Available modes: • **RS-232**. • **RS-485** (2 wires). • **RS-422** (4 wires).

Tag Editor Settings

In Tag Editor select the protocol **A-B DF1**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Disc Out</td><td>Discrete output value. O resource on PLC.</td></tr>
<tr><td>Disc In</td><td>Discrete input value. I resource on PLC.</td></tr>
<tr><td>Status</td><td>Status value. S resource on PLC.</td></tr>
<tr><td>Bit</td><td>Bit value. B resource on PLC.</td></tr>
<tr><td>Timer</td><td>Timer value. T resource on PLC.</td></tr>
<tr><td>Counter</td><td>Counter value. C resource on PLC.</td></tr>
<tr><td>Control</td><td>Control value. R resource on PLC.</td></tr>
<tr><td>Integer</td><td>Integer value. N resource on PLC.</td></tr>
<tr><td>Float</td><td>Float value. F resource on PLC.</td></tr>
</table>
</td></tr>
<tr><td>Element</td><td>Represents the line of the resource while monitoring PLC values.</td></tr>
<tr><td>Subindex</td><td>Represents the column of the resource while monitoring PLC values.</td></tr>
<tr><td>File Num</td><td>Instance of resource of the PLC.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
double
string
binary
See "Programming concepts" section in the main manual.
Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).</td></tr>
<tr><td>Arraysize</td><td>In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.</td></tr>
</table>

Element	Description
	Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Sub Element	Allows to point to specific part of a resource: • 0 (entire resource) • PRE • ACC • LEN • POS
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>Example:
9ACC → CC9A (in hexadecimal format)
39628 → 52378 (in decimal format)</td></tr>
<tr><td>ABCDEFGH -> GHEFCDAB</td><td>swap4: Swap bytes in a double word.

Example:
32FCFF54 → 54FFFC32 (in hexadecimal format)
855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000 0001
→
1 10000011100
10101010000101000101101101101100101101100001001111 1101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

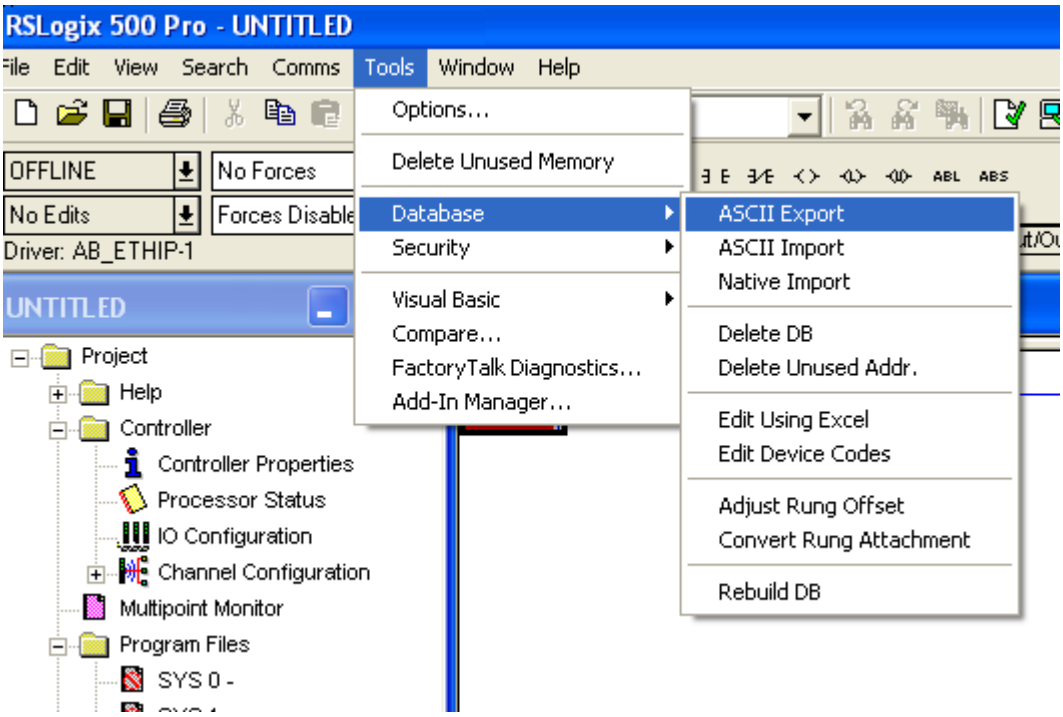
Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

Tag Import

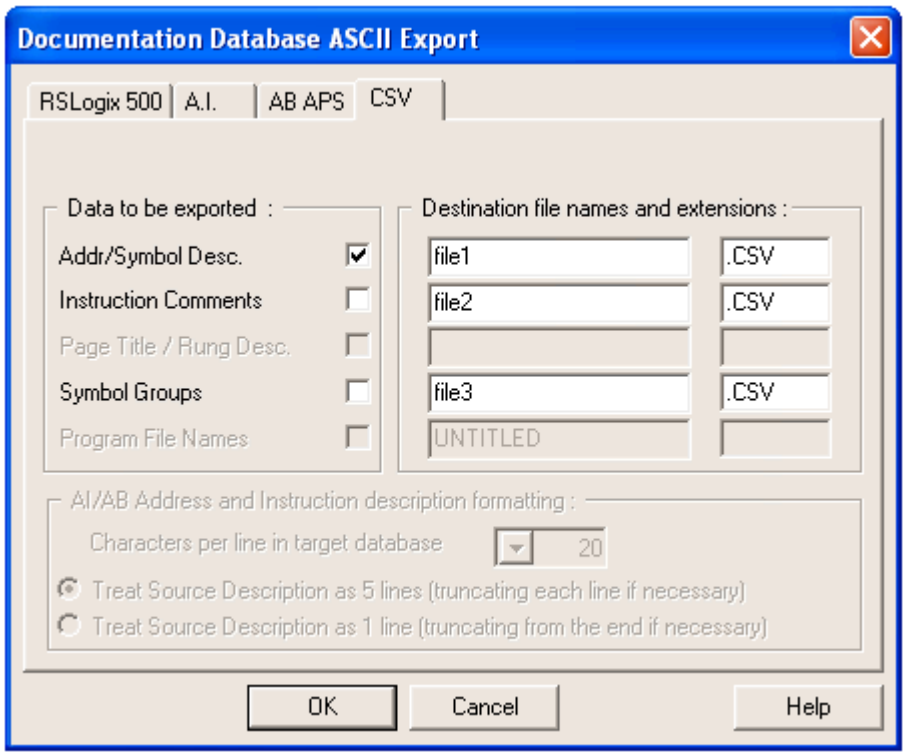
Exporting Tags from PLC

The A-B DF1 tag import filter accepts symbol files with extension “.csv” created by the Rockwell RSLogix 500.

To create the file select **Tool > Database > ASCII Export**

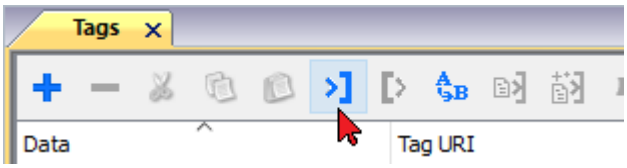


From **CSV** tab select the data to be exported and give a name to the output csv file.

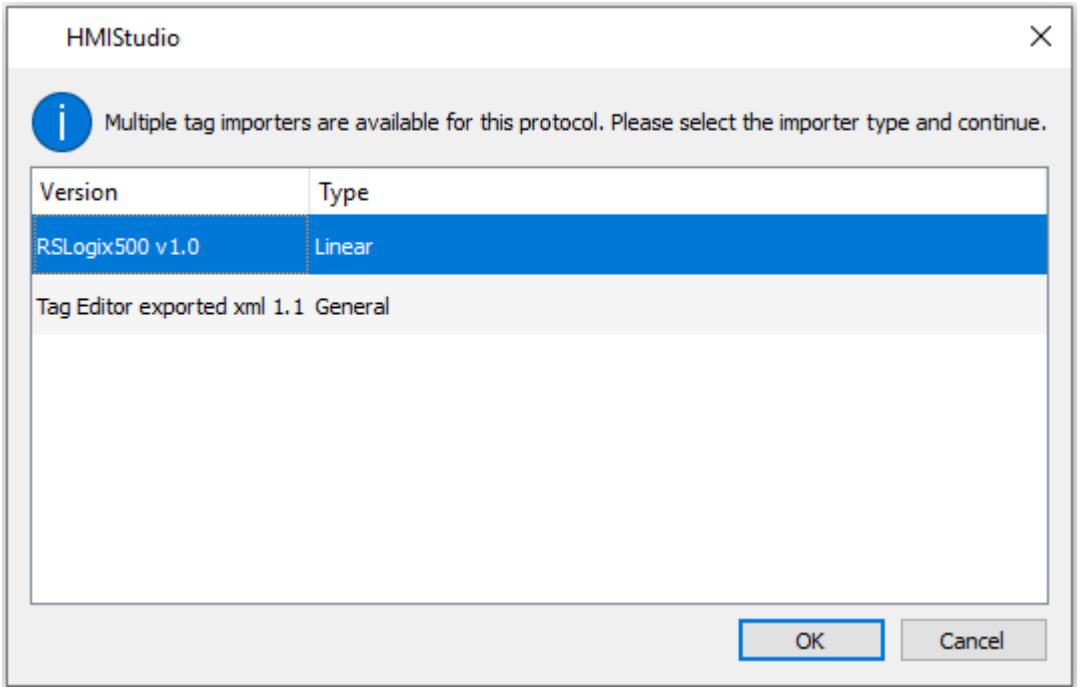


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



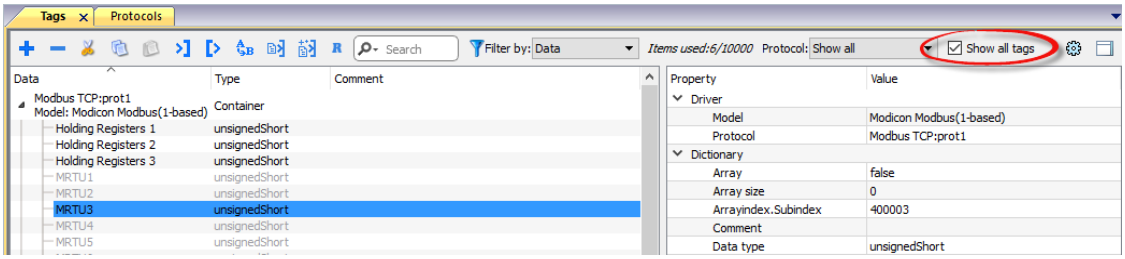
The following dialog shows which importer type can be selected.

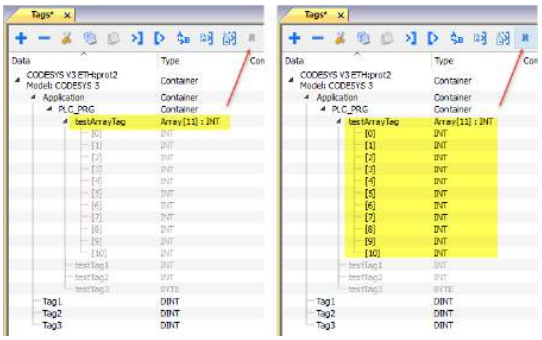
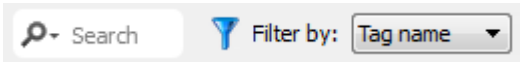


Importer	Description
RSLogix500 v1.0 Linear	Requires an **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



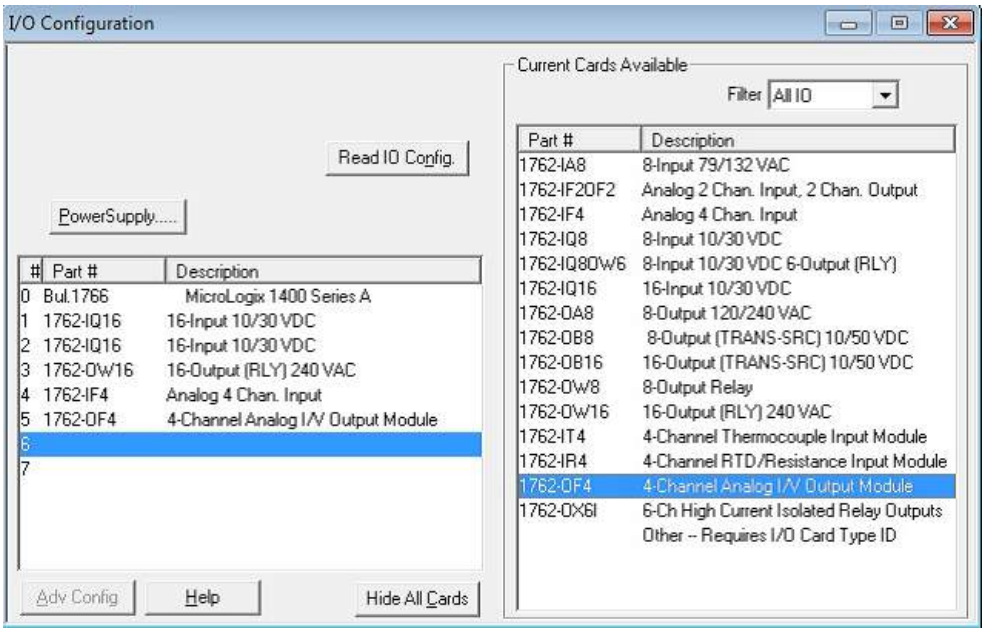
Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Logical I/O addressing

When addressing Allen Bradley I/O data, the panel uses logical addressing rather than physical addressing. While physical addressing refers to the element number as the slot number, logical addressing refers to the first element for the first I/O card of a specific file type.

Communication Protocols addressing depends on the mapping of the PLC CPU memory and not on the slot number, therefore you should be careful when changing the configuration in order to avoid remapping.

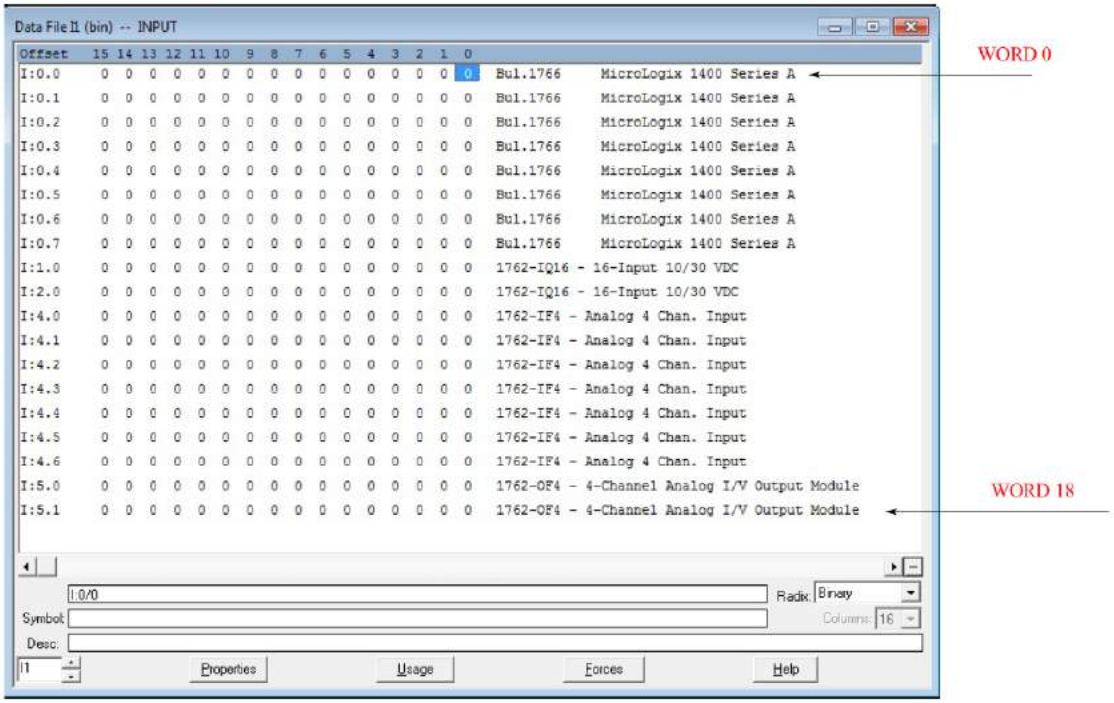
Use the RSLogix 500 I/O Configuration tool layout of the PLC I/O to configure I/O as in the example.



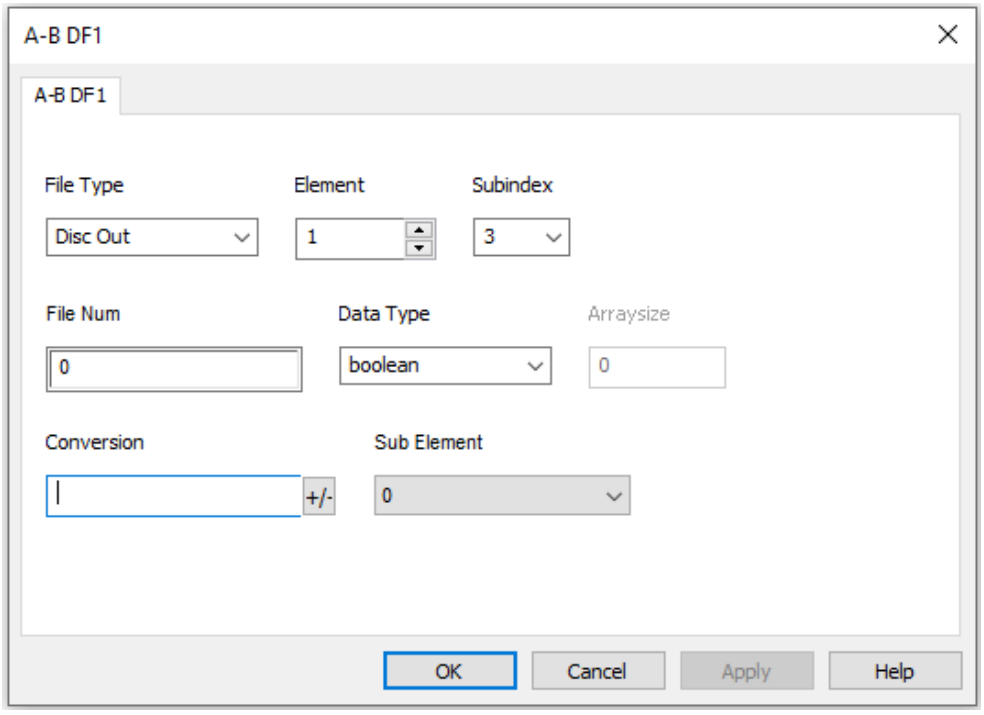
Note: When using a module with a configurable I/O size (for example, Devicenet Scanner) make sure you configure it to the largest possible size or you will have to remap it if you need to allocate more space.

Use the Data File Browser to see how the PLC allocates memory.

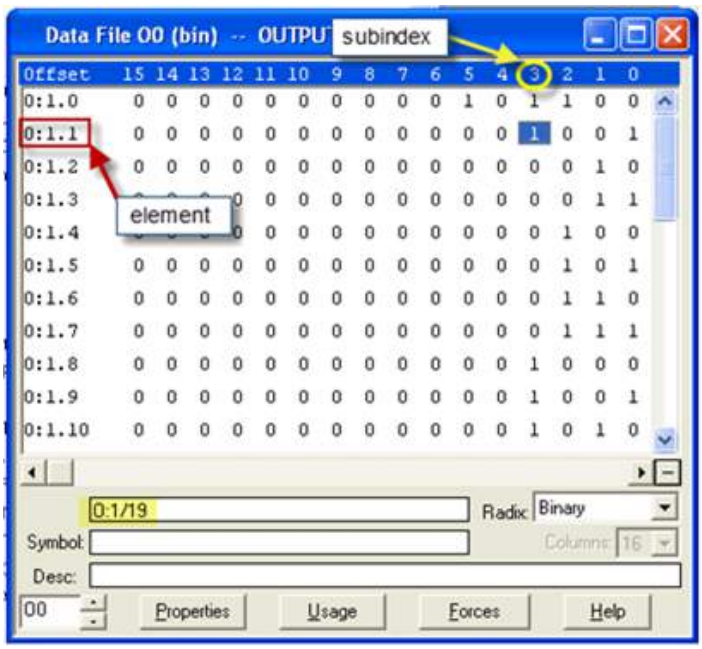
This example shows how to configure the Communication Protocols Tag for pointing to PLC resource O:1/19 (O1:1.1/3 in word terms).



The following figure shows the Communication Protocols Tag configuration.



The Communication Protocols Tag configured in the example above points on the element shown in the following figure.



Examples

I:0/19 (I1:0.1/3 in word terms) – 20th Input on CPU

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Boolean

In the Data File Browser, word 0.1 is Word 1:

Element	1
Sub Index	3

I:1/15 (I1:1.0/15 in word terms) - Last Input on Slot 1 Input Card

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Boolean

In the Data File Browser, word 1.0 is Word 8:

Element	8
Sub Index	15

I:4.0 (I1:4.0 in word terms) - First Analog Input

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Short

In the Data File Browser, word 4.0 is Word 10:

Element	10
Sub Index	-

A-B DH-485

The A-B DH-485 communication driver has been designed to connect HMI devices to a Allen-Bradley controllers through serial communication.

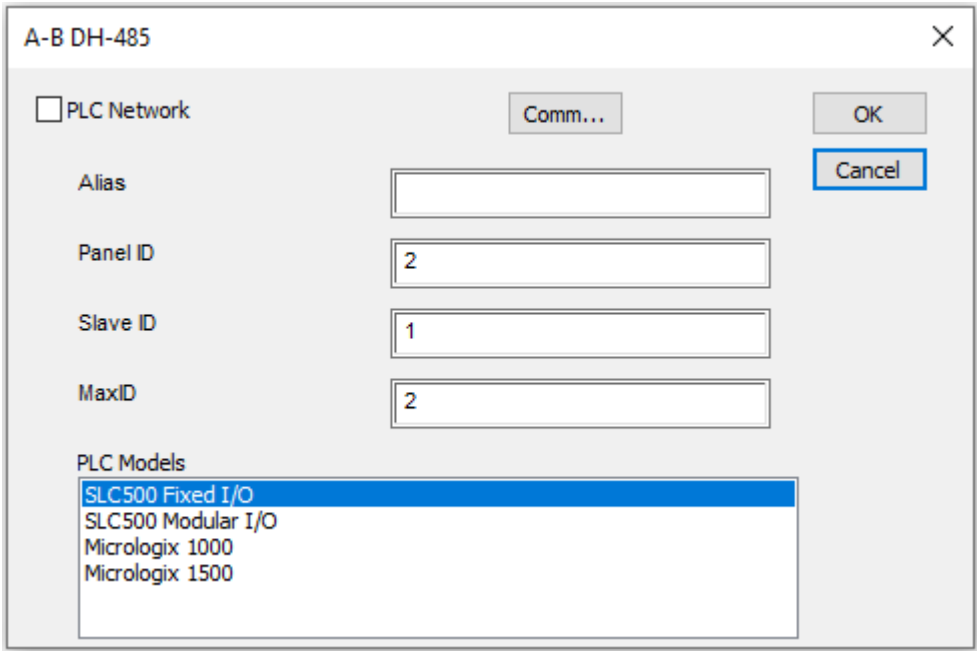
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

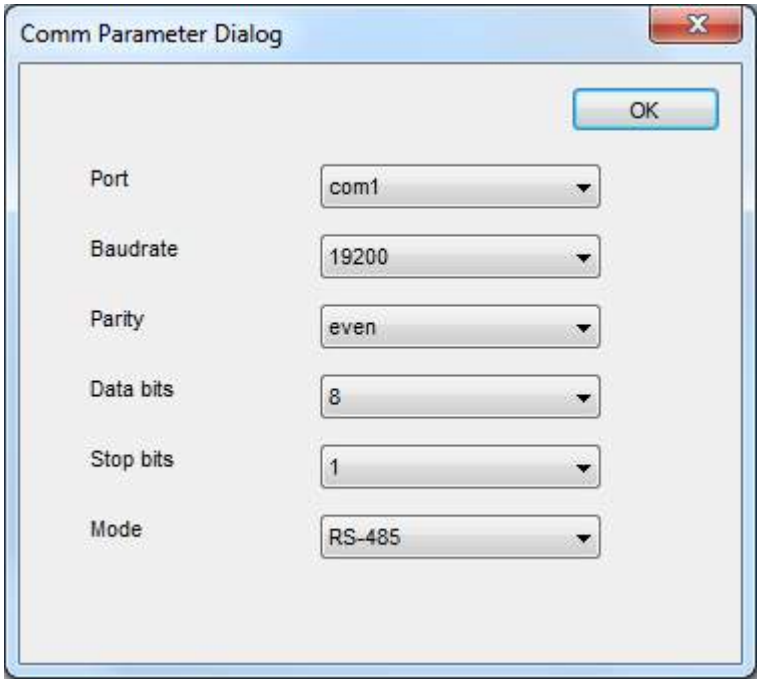
The protocol configuration dialog is displayed.



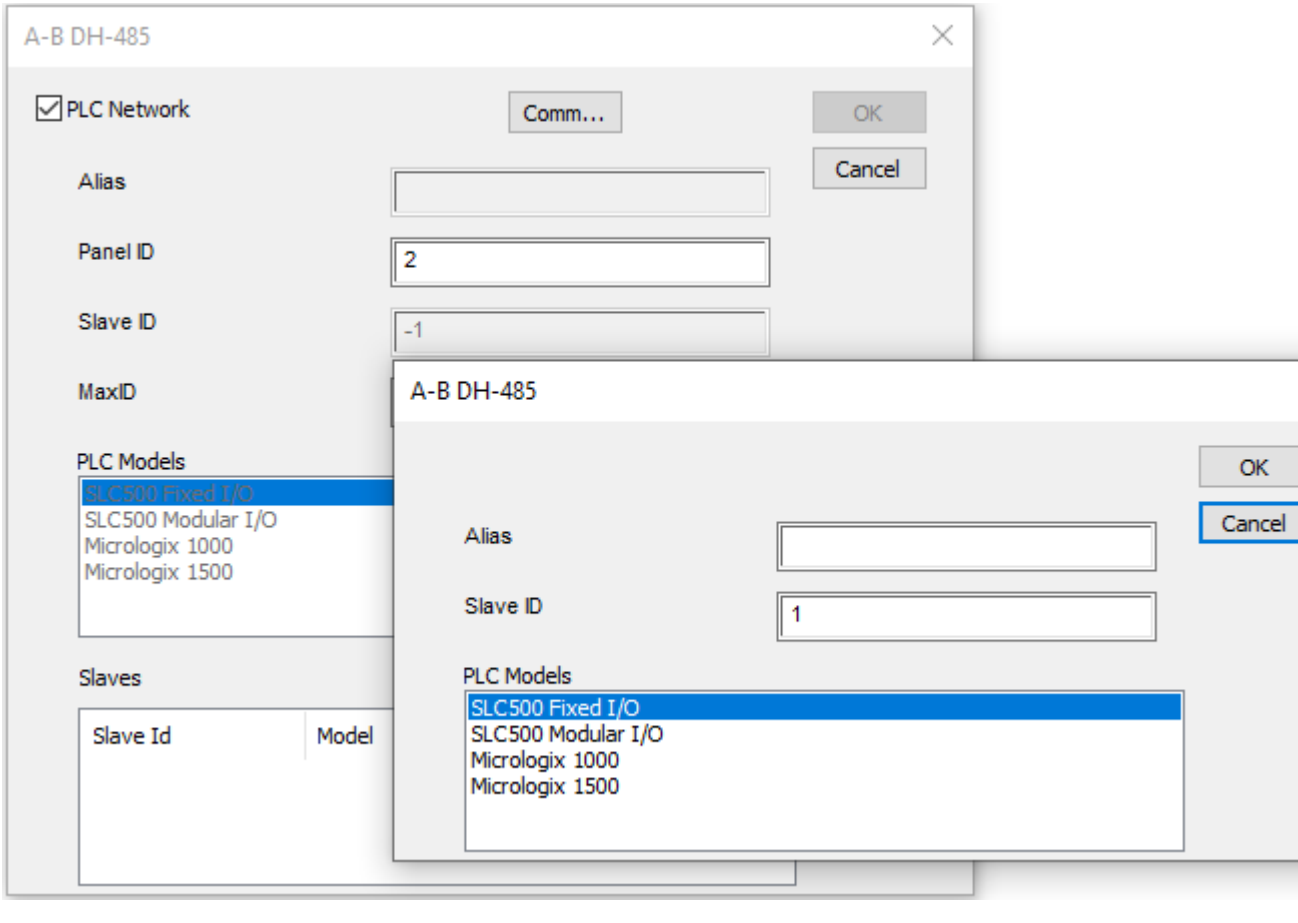
Element	Description
Panel ID	Serial node associated to the HMI.
Slave ID	Serial node associated to the PLC.
MaxID	Represent the maximum ID available in the serial network.
PLC Models	PLC models available: • SLC500 Fixed I/O

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>

SLC500 Modular I/O
Micrologix 1000
Micrologix 1500

</td></tr>
<tr><td>Comm...</td><td>If clicked displays the communication parameters setup dialog.
 

<table>
<tr><th>Element</th><th>Parameter</th></tr>
<tr><td>Port</td><td>Serial port selection.
COM1: device PLC port.
COM2: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board.
COM3: optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.</td></tr>
<tr><td>Baudrate, Parity, Data Bits, Stop bits</td><td>Serial line parameters.</td></tr>
<tr><td>Mode</td><td>Serial port mode. Available modes:
RS-232.
RS-485 (2 wires).
RS-422 (4 wires).</td></tr>
</table>
</td></tr>
<tr><td>PLC Network</td><td>IP address for all controllers in multiple connections. PLC Network must be selected to enable multiple connections.</td></tr>
</table>

Element	Description
	

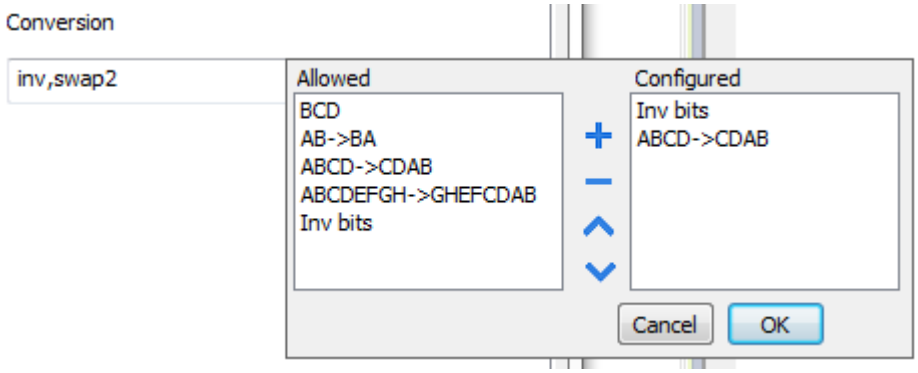
Tag Editor Settings

In Tag Editor select the protocol **A-B DH-485**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:

Element	Description
Memory Type	<table><tr><th>Memory Type</th><th>Description</th></tr><tr><td>**Disc Out**</td><td>Discrete output value. **O** resource on PLC.</td></tr><tr><td>**Disc In**</td><td>Discrete input value. **I** resource on PLC.</td></tr><tr><td>**Status**</td><td>Status value. **S** resource on PLC.</td></tr><tr><td>**Bit**</td><td>Bit value. **B** resource on PLC.</td></tr><tr><td>**Timer**</td><td>Timer value. **T** resource on PLC.</td></tr><tr><td>**Counter**</td><td>Counter value. **C** resource on PLC.</td></tr><tr><td>**Control**</td><td>Control value. **R** resource on PLC.</td></tr><tr><td>**Integer**</td><td>Integer value. **N** resource on PLC.</td></tr><tr><td>**Float**</td><td>Float value. **F** resource on PLC.</td></tr><tr><td>**String**</td><td>String value. **STR** resource on PLC.</td></tr></table>
Element	Represents the line of the resource while monitoring PLC values.
Subindex	Represents the column of the resource while monitoring PLC values.
File Num	Instance of resource of the PLC.

Element	Description
Data Type	Available data types: • **boolean** • **byte** • **short** • **int** • **unsignedByte** • **unsignedShort** • **unsignedInt** • **float** • **double** • **string** • **binary** See "Programming concepts" section in the main manual.  Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.

Element	Description
Sub Element	Allows to point to specific part of a resource: • 0 (entire resource) • PRE • ACC • LEN • POS
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH	**swap4**: Swap bytes in a double word.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>->
GHEFCDAB</td><td>Example:
32FCFF54 → 54FFFC32 (in hexadecimal format)
855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP ->
OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000 0001
→
1 10000011100
10101010000101000101101101101100101101100001001111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

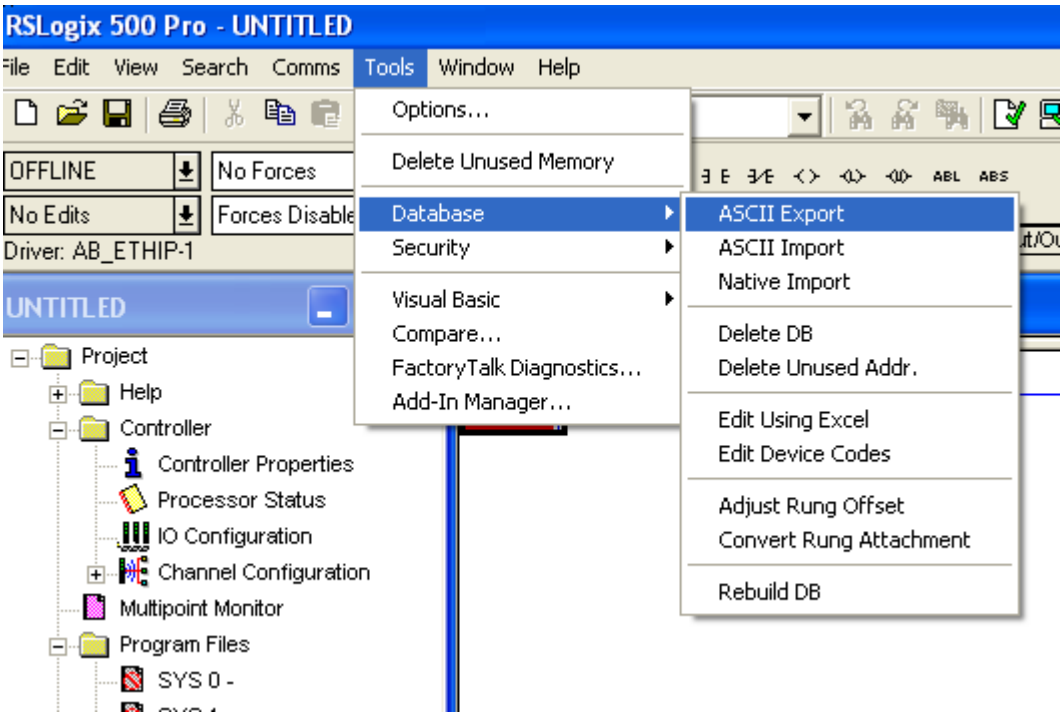
Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

Tag Import

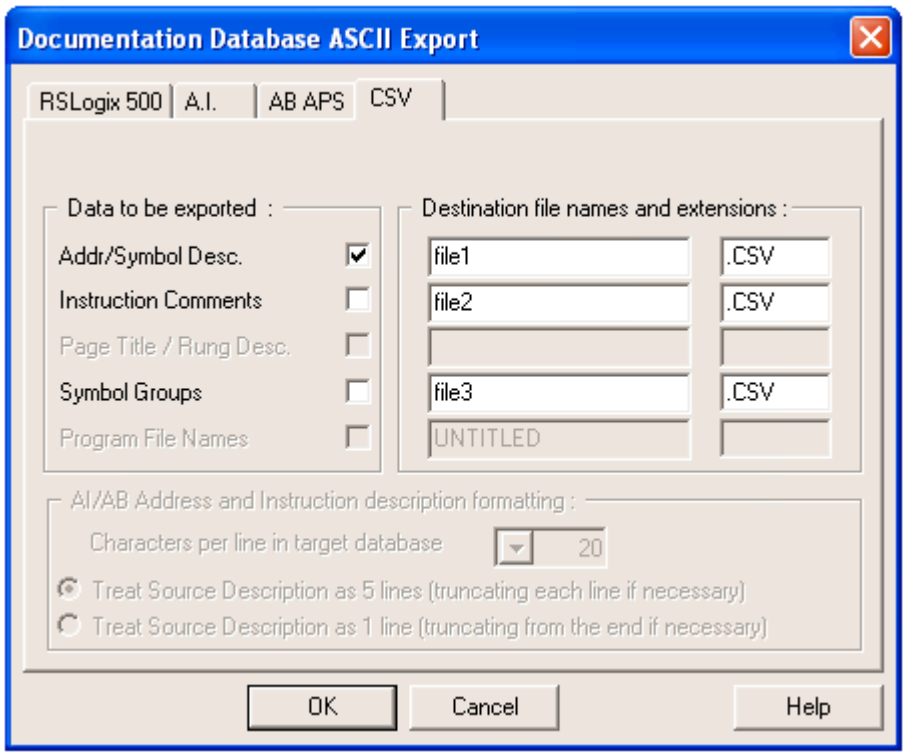
Exporting Tags from PLC

The A-B DH-485 tag import filter accepts symbol files with extension “.csv” created by the Rockwell RSLogix 500.

To create the file select **Tool > Database > ASCII Export**

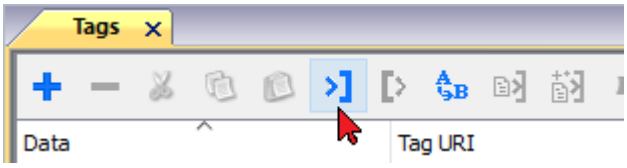


From **CSV** tab select the data to be exported and give a name to the output csv file.

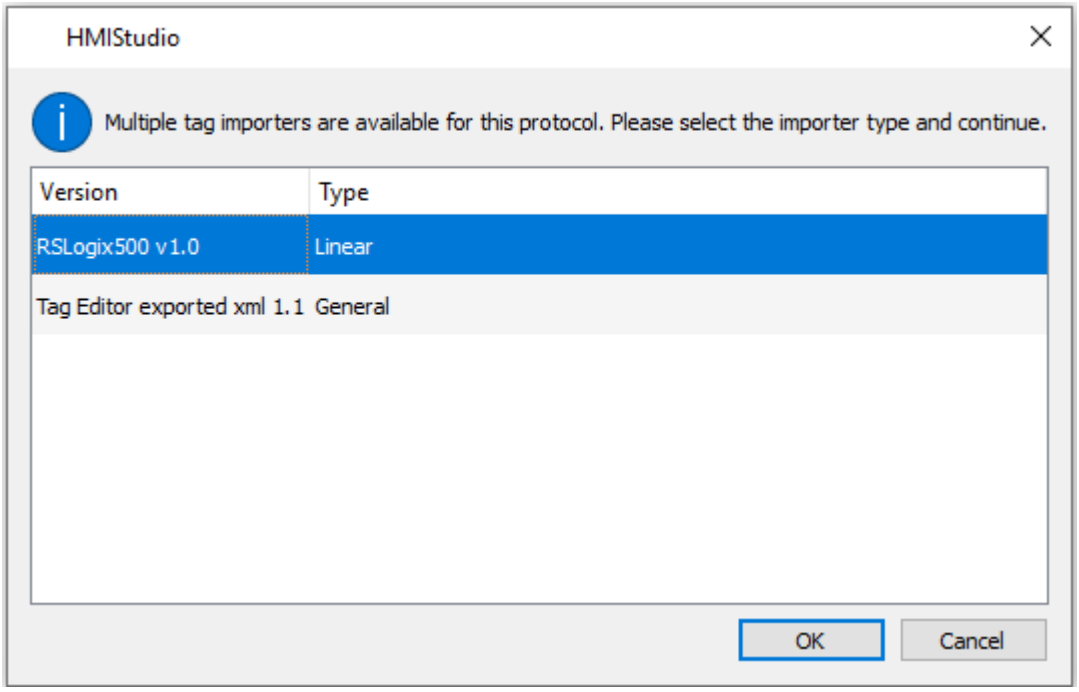


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



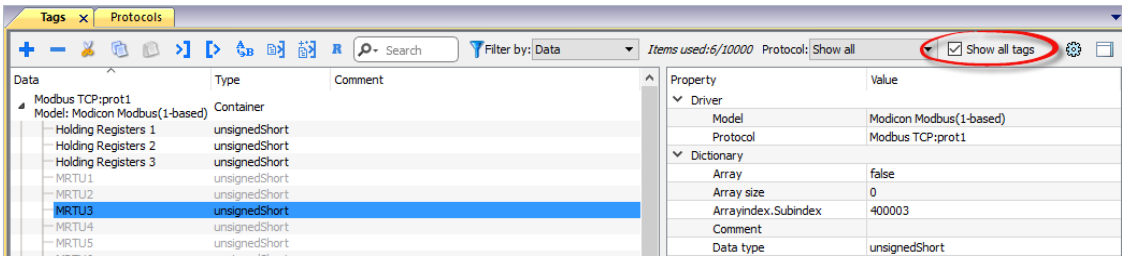
The following dialog shows which importer type can be selected.

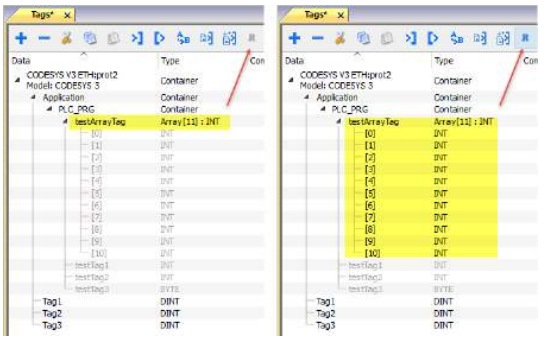
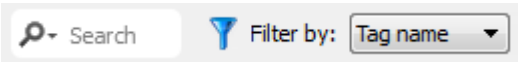


Importer	Description
RSLogix500 v1.0 Linear	Requires an **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



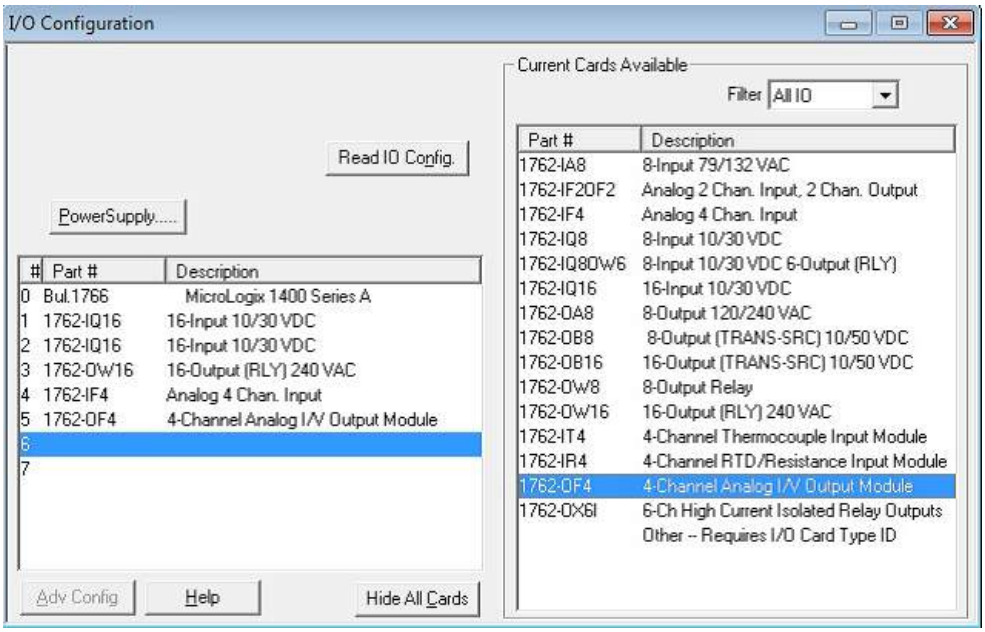
Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Logical I/O addressing

When addressing Allen Bradley I/O data, the panel uses logical addressing rather than physical addressing. While physical addressing refers to the element number as the slot number, logical addressing refers to the first element for the first I/O card of a specific file type.

Communication Protocols addressing depends on the mapping of the PLC CPU memory and not on the slot number, therefore you should be careful when changing the configuration in order to avoid remapping.

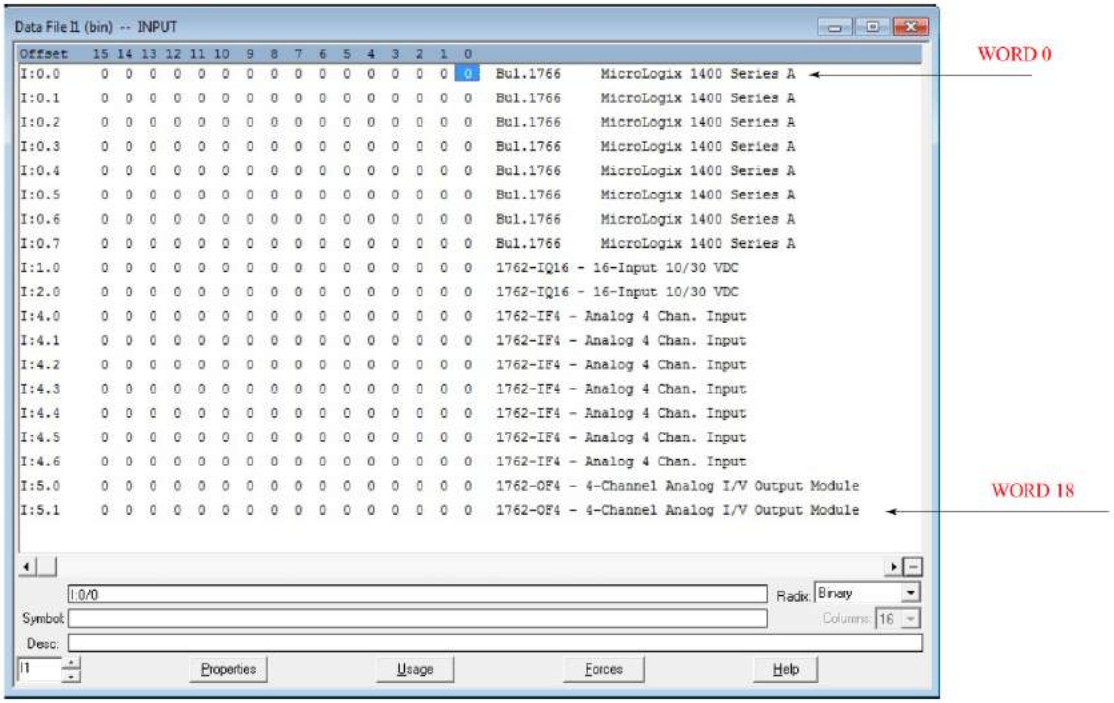
Use the RSLogix 500 I/O Configuration tool layout of the PLC I/O to configure I/O as in the example.



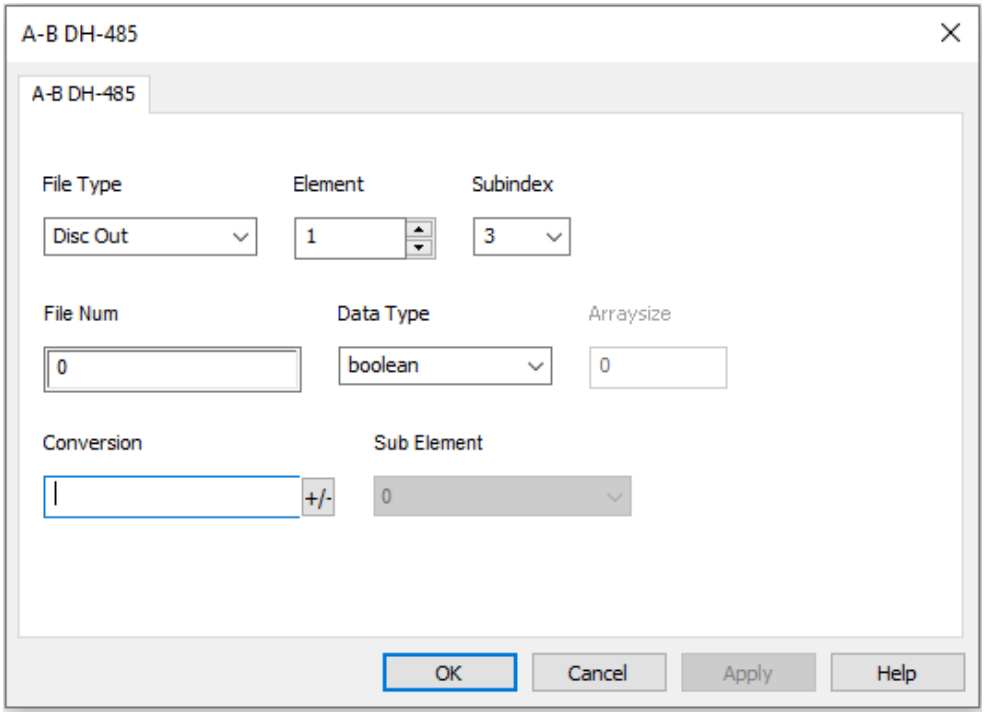
Note: When using a module with a configurable I/O size (for example, Devicenet Scanner) make sure you configure it to the largest possible size or you will have to remap it if you need to allocate more space.

Use the Data File Browser to see how the PLC allocates memory.

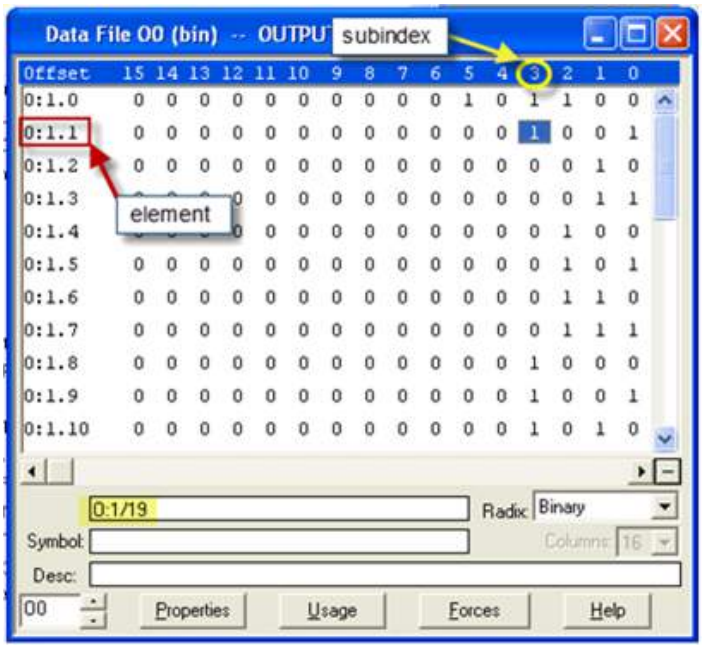
This example shows how to configure the Communication Protocols Tag for pointing to PLC resource O:1/19 (O1:1.1/3 in word terms).



The following figure shows the Communication Protocols Tag configuration.



The Communication Protocols Tag configured in the example above points on the element shown in the following figure.



Examples

I:0/19 (I1:0.1/3 in word terms) – 20th Input on CPU

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Boolean

In the Data File Browser, word 0.1 is Word 1:

Element	1
Sub Index	3

I:1/15 (I1:1.0/15 in word terms) - Last Input on Slot 1 Input Card

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Boolean

In the Data File Browser, word 1.0 is Word 8:

Element	8
Sub Index	15

I:4.0 (I1:4.0 in word terms) - First Analog Input

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Short

In the Data File Browser, word 4.0 is Word 10:

Element	10
Sub Index	-

A-B ENET

The A-B ENET communication protocol is normally used on the Allen-Bradley controllers via Ethernet communication.

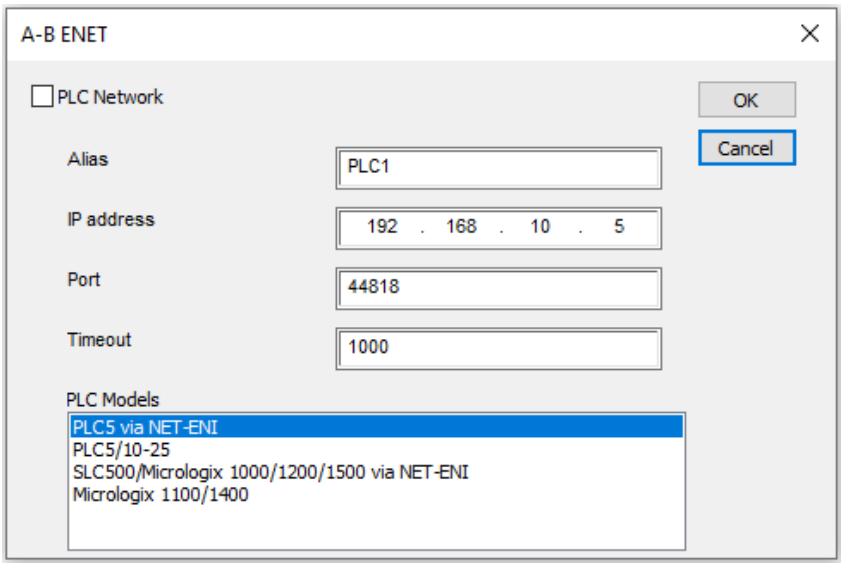
Protocol Editor Settings

Adding a protocol

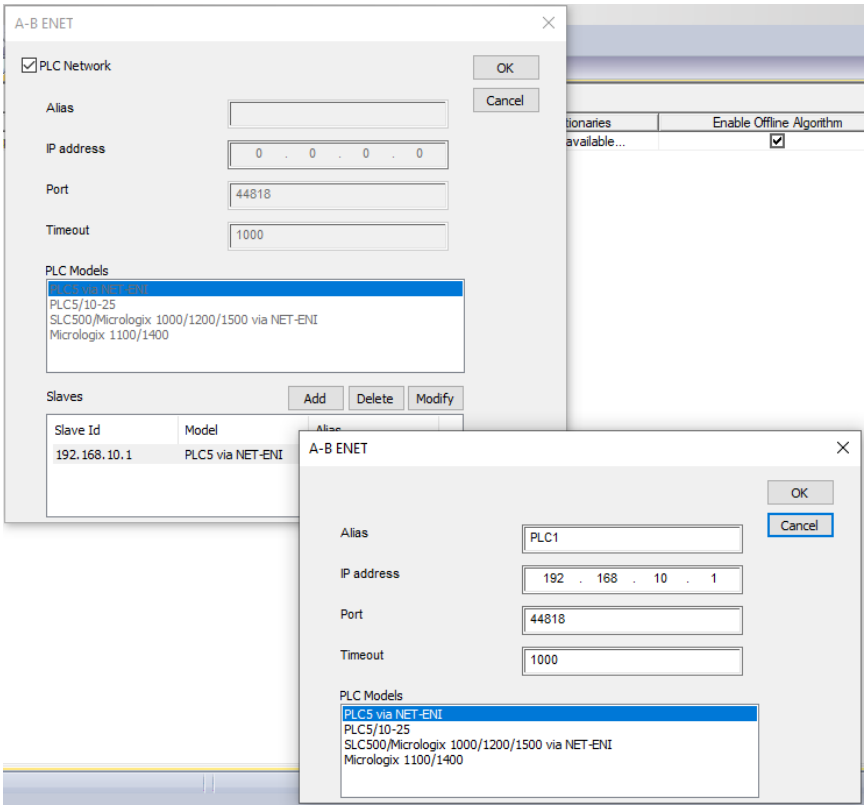
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.

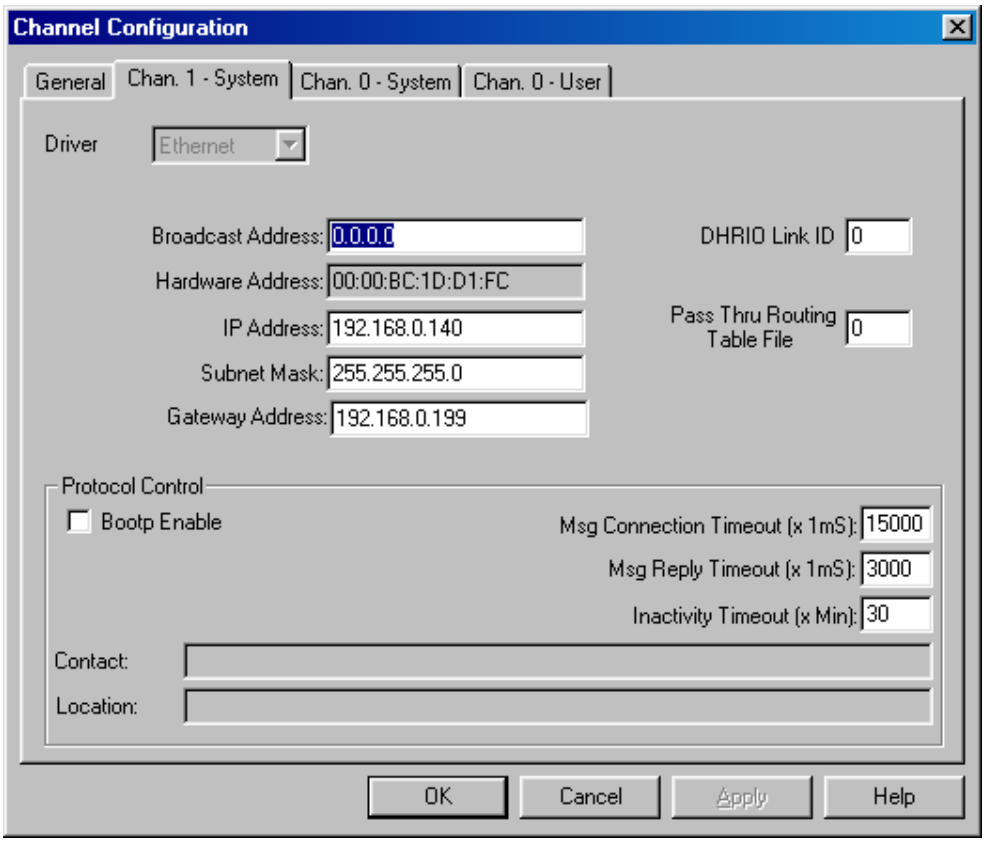


Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP Address	Ethernet IP address of the controller.
Port	Port number used by the Ethernet interface.

Element	Description
Timeout	Time delay in milliseconds between two retries in case of missing response from the controller.
PLC Network	Enable access to multiple networked controllers. For every controller (slave) set the proper option. 

Controller configuration

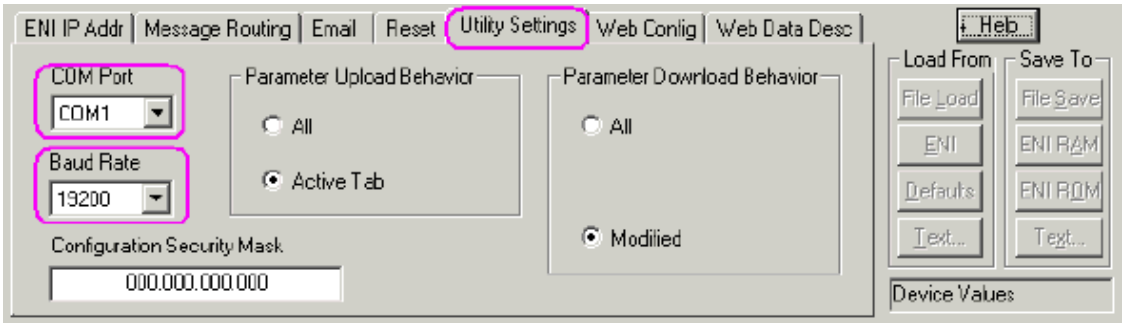
The PLC has to be correctly configured to match the IP address configured in the Protocol Editor. Normally the PLC configuration can be left as default.



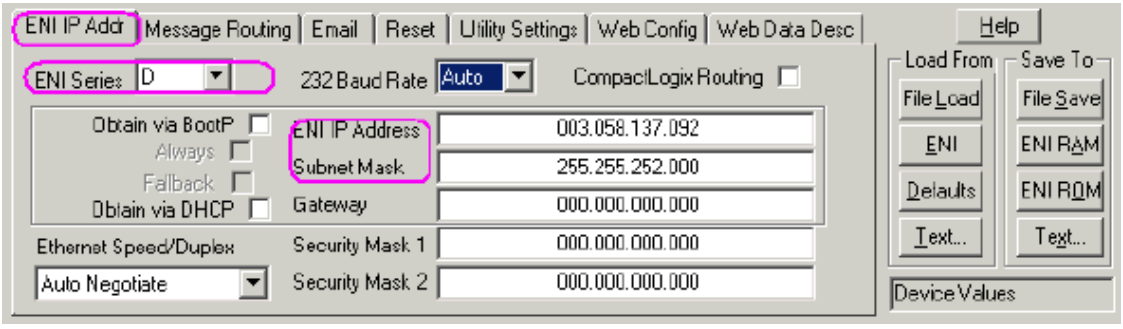
Configuring 1761-NET-ENI

Here is the procedure to configure the 1761-NET-ENI module using the Allen Bradley's ENI/ENIW Utility. The procedure requires a 1761-CBL-PM02 communication cable.

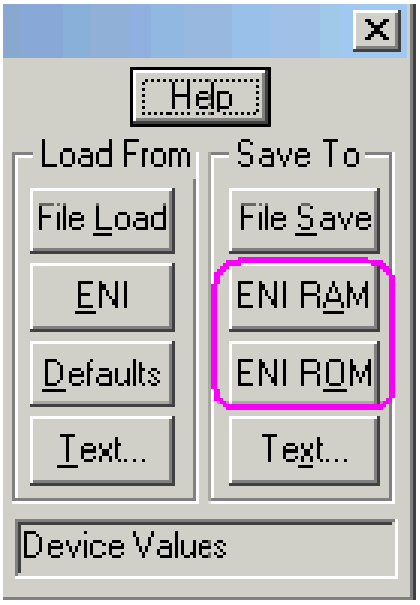
1. Connect the 8 pin din to the port 2 on the NET-ENI device and the 9 pin female D-shell to the computer COM port.
2. Connect the SLC 5/0x controller and go online.
3. In the **Utility Settings** tab, set **COM Port** and **Baud Rate**.



4. In the **ENI IP Addr** tab, select the correct **ENI Series** from the list and set **ENI IP Address, Subnet Mask** and **Baud Rate**, if needed.



5. Save the configuration to the NET-ENI device.



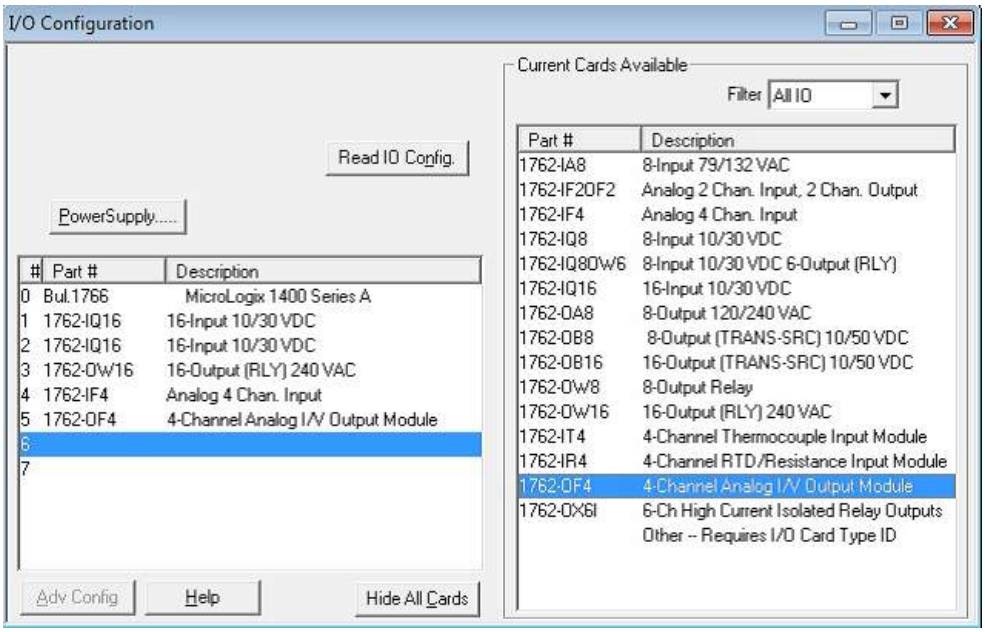
Two separate memory areas are reserved for saving the configuration : **ENI/RAM** (for temporary configurations) and **ENI/ROM** (for permanent configurations).

Logical I/O addressing

When addressing Allen Bradley I/O data, the panel uses logical addressing rather than physical addressing. While physical addressing refers to the element number as the slot number, logical addressing refers to the first element for the first I/O card of a specific file type.

Communication Protocols addressing depends on the mapping of the PLC CPU memory and not on the slot number, therefore you should be careful when changing the configuration in order to avoid remapping.

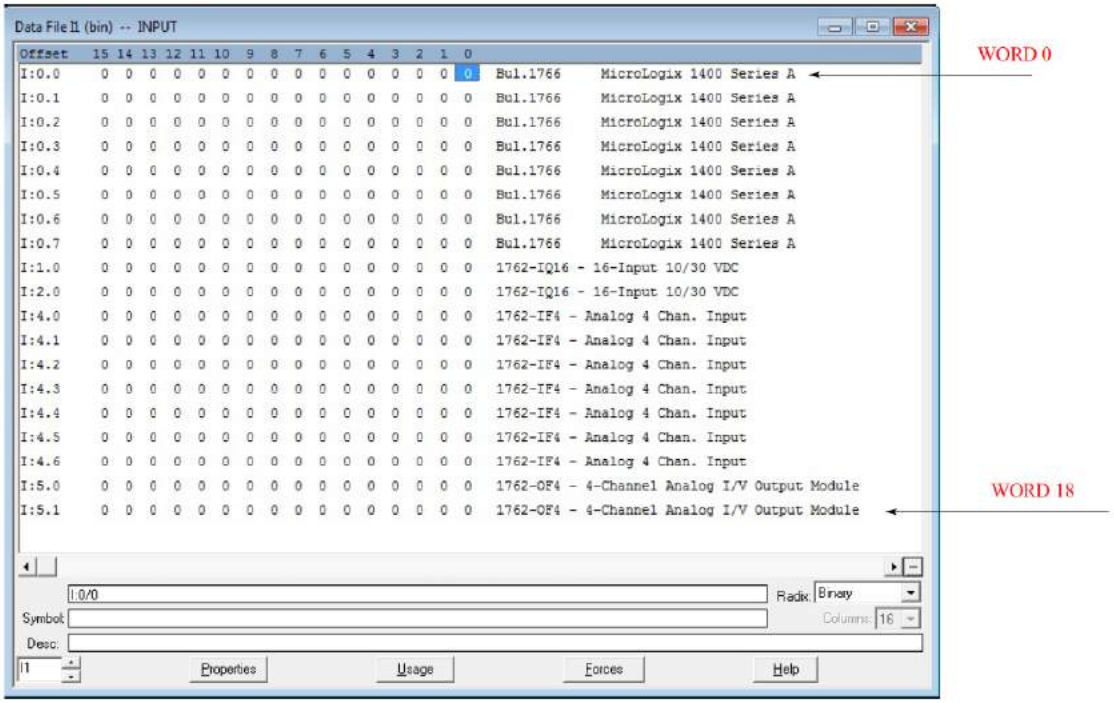
Use the RSLogix 500 I/O Configuration tool layout of the PLC I/O to configure I/O as in the example.



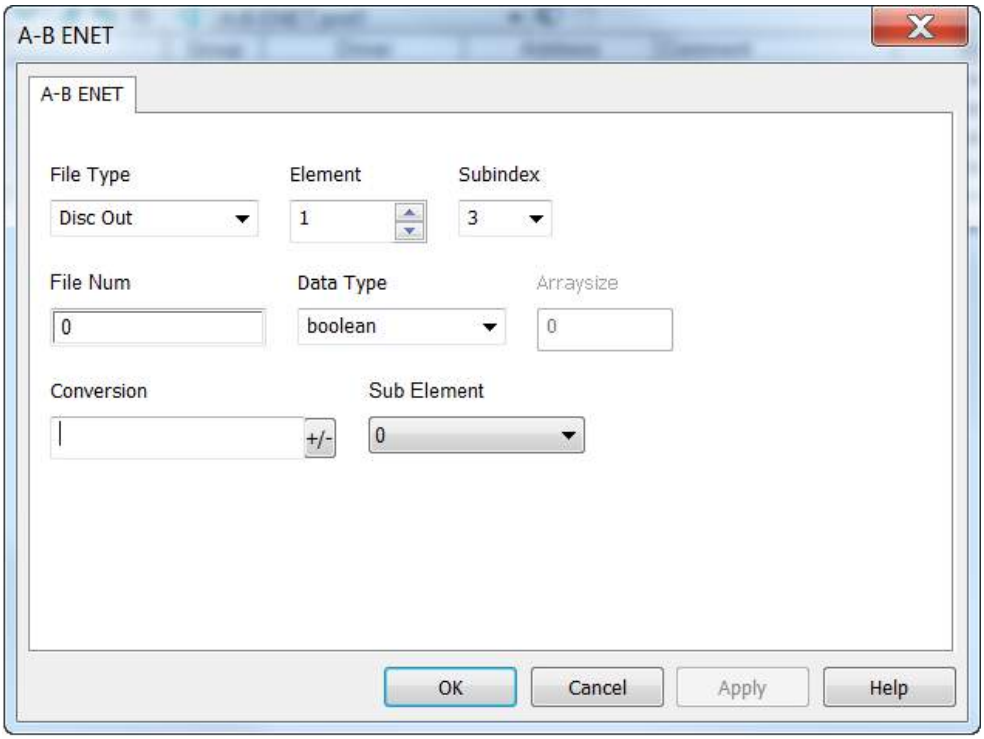
Note: When using a module with a configurable I/O size (for example, Devicenet Scanner) make sure you configure it to the largest possible size or you will have to remap it if you need to allocate more space.

Use the Data File Browser to see how the PLC allocates memory.

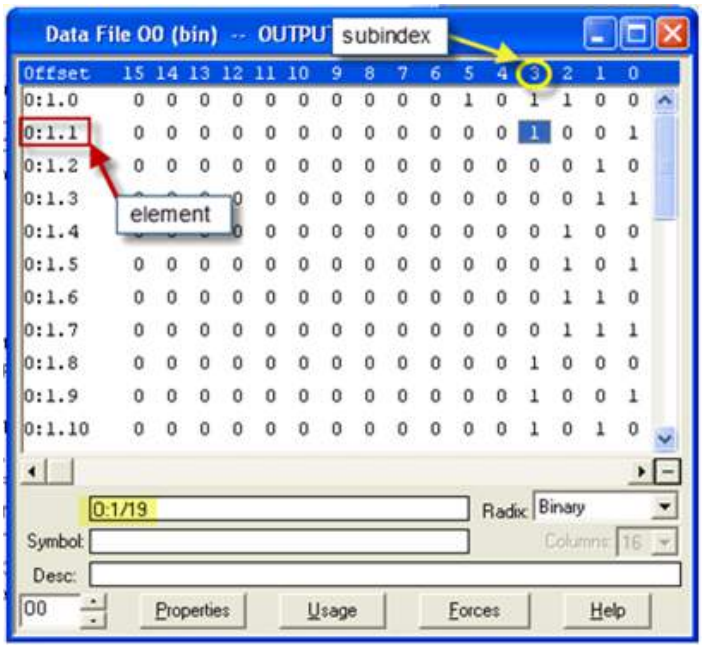
This example shows how to configure the Communication Protocols Tag for pointing to PLC resource O:1/19 (O1:1.1/3 in word terms).



The following figure shows the Communication Protocols Tag configuration.



The Communication Protocols Tag configured in the example above points on the element shown in the following figure.



Examples

I:0/19 (I1:0.1/3 in word terms) – 20th Input on CPU

Parameter	Setting
File Type	Disc In
File Num	1

Parameter	Setting
Data Type	Boolean

In the Data File Browser, word 0.1 is Word 1:

Element	1
Sub Index	3

I:1/15 (I1:1.0/15 in word terms) - Last Input on Slot 1 Input Card

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Boolean

In the Data File Browser, word 1.0 is Word 8:

Element	8
Sub Index	15

I:4.0 (I1:4.0 in word terms) - First Analog Input

Parameter	Setting
File Type	Disc In
File Num	1
Data Type	Short

In the Data File Browser, word 4.0 is Word 10:

Element	10
Sub Index	-

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

Hostname DNS or mDNS

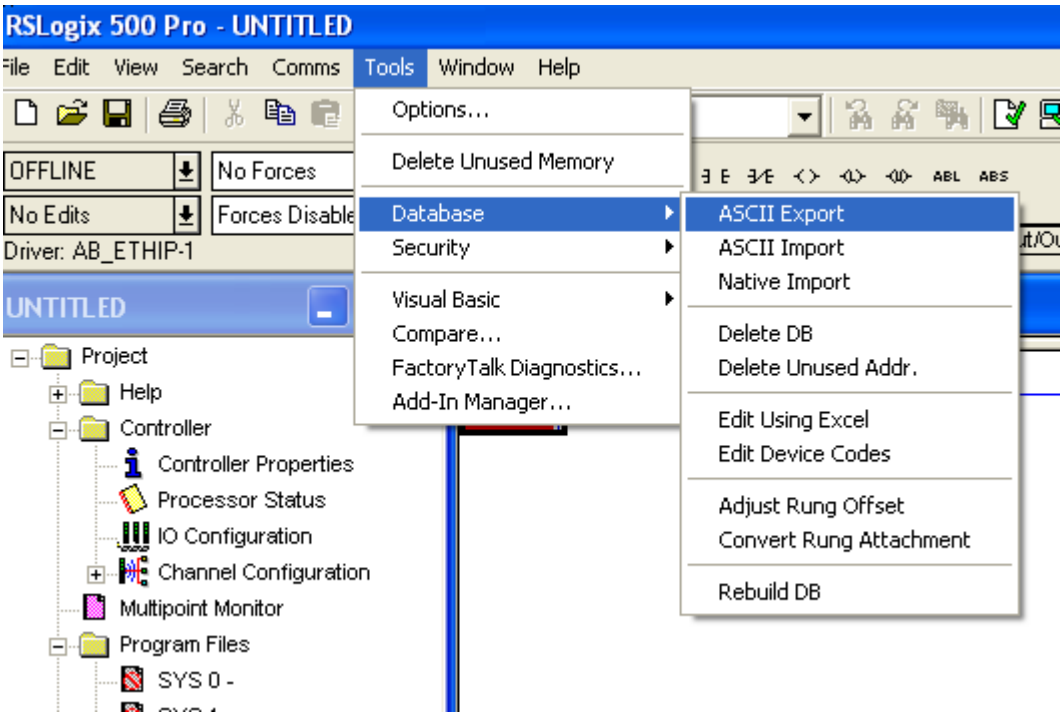
In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.

Tag Import

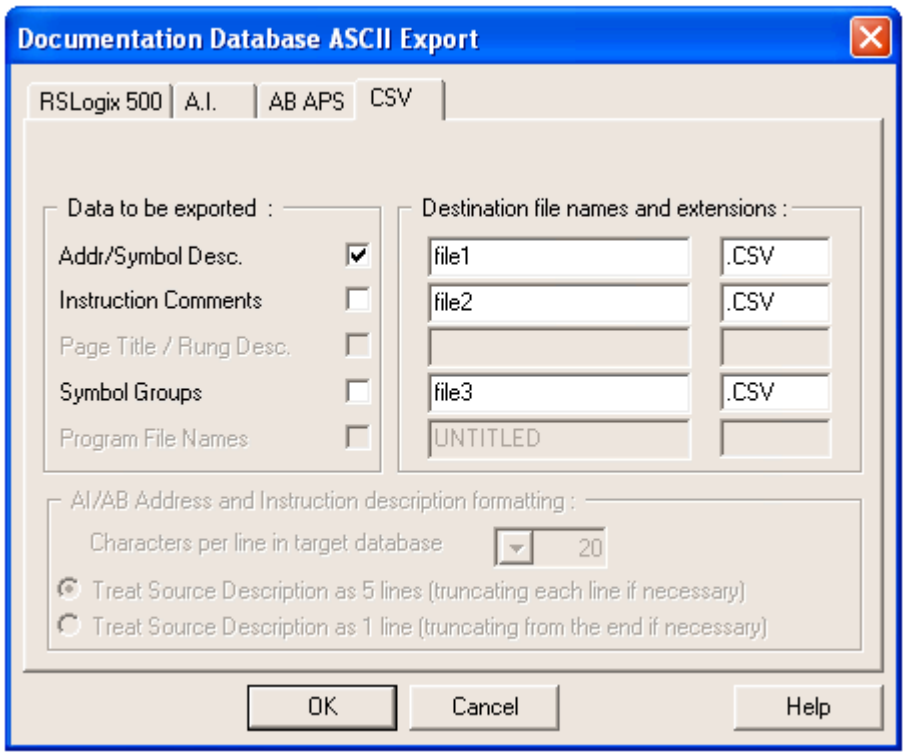
Exporting Tags from PLC

The A-B Ethernet tag import filter accepts symbol files with extension ".csv" created by the Rockwell RSLogix 500.

To create the file select **Tool > Database > ASCII Export**

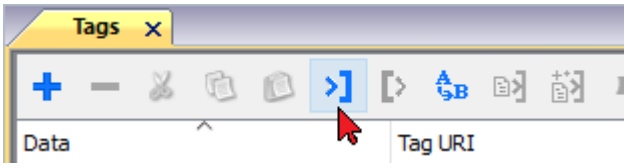


From **CSV** tab select the data to be exported and give a name to the output csv file.

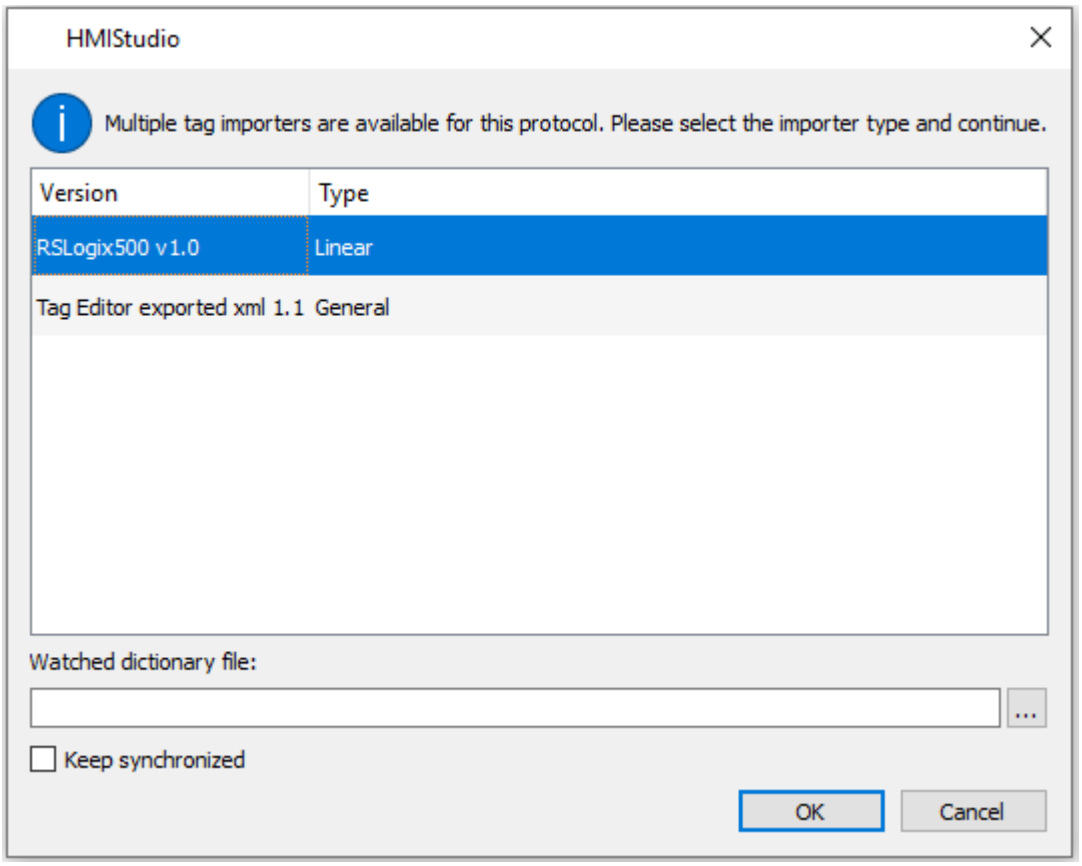


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



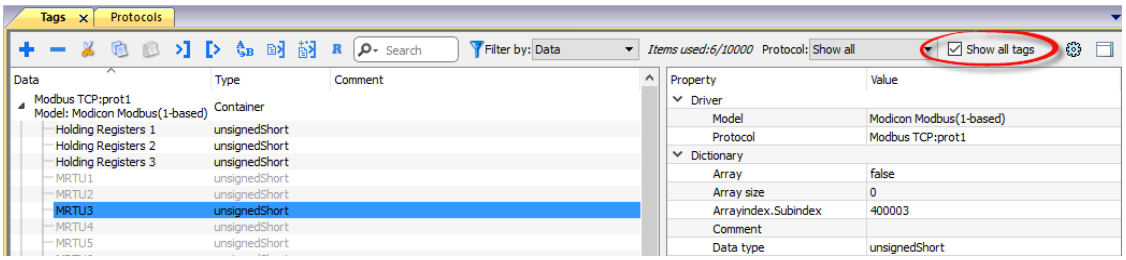
The following dialog shows which importer type can be selected.

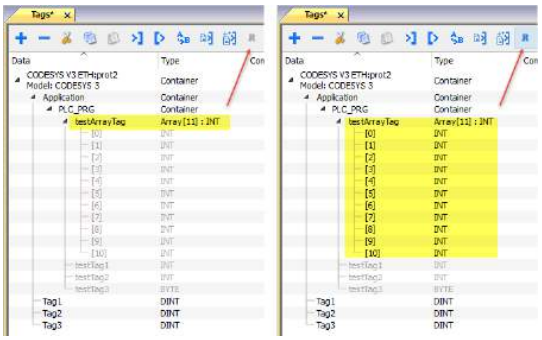
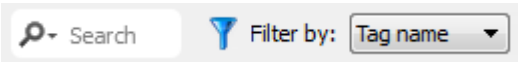


Importer	Description
RSLogix500 v1.0 Linear	Requires an **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid	The device did received a response with invalid	Check if the data programmed in the project are

Error	Cause	Action
response	format or contents from the controller.	consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

ABB Mint Controller HCP

This communication protocol allows the HMI devices to connect to the ABB motion and servo drive devices using the HCP and HCP2 communication protocols.

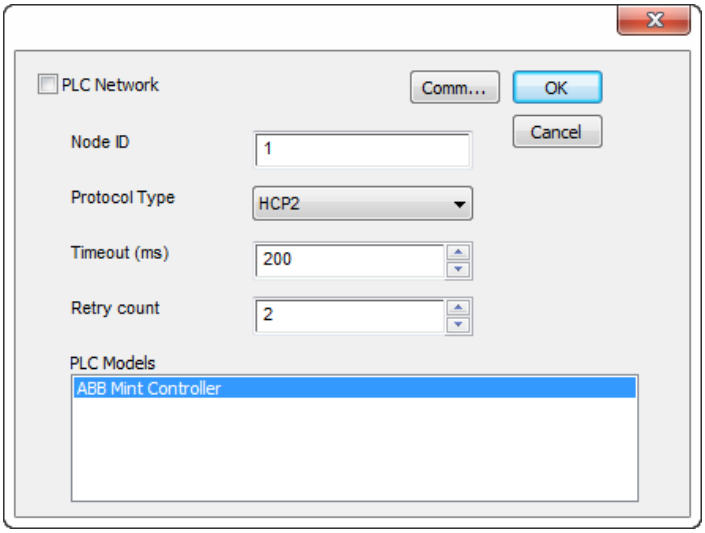
Protocol Editor Settings

Adding a protocol

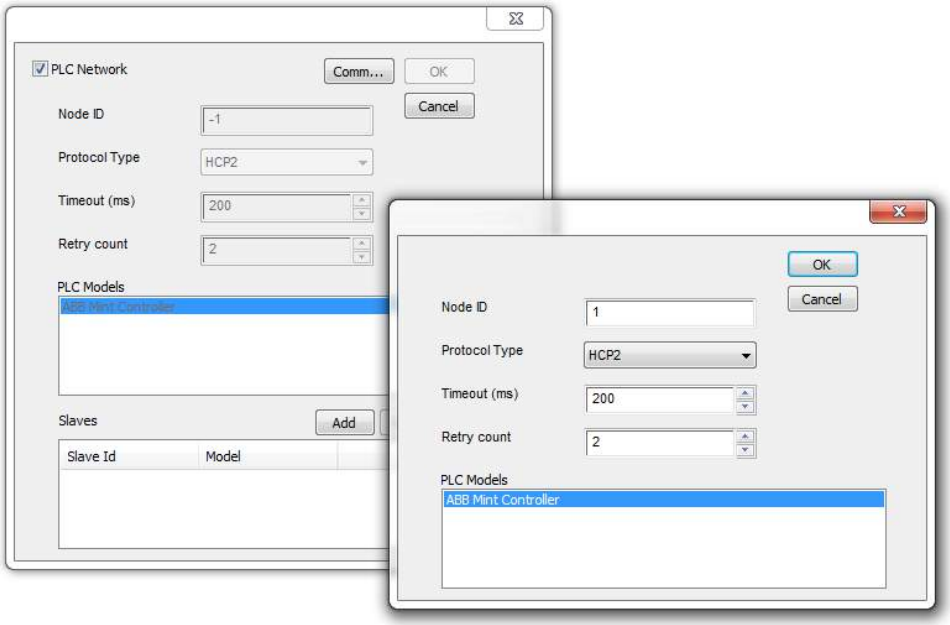
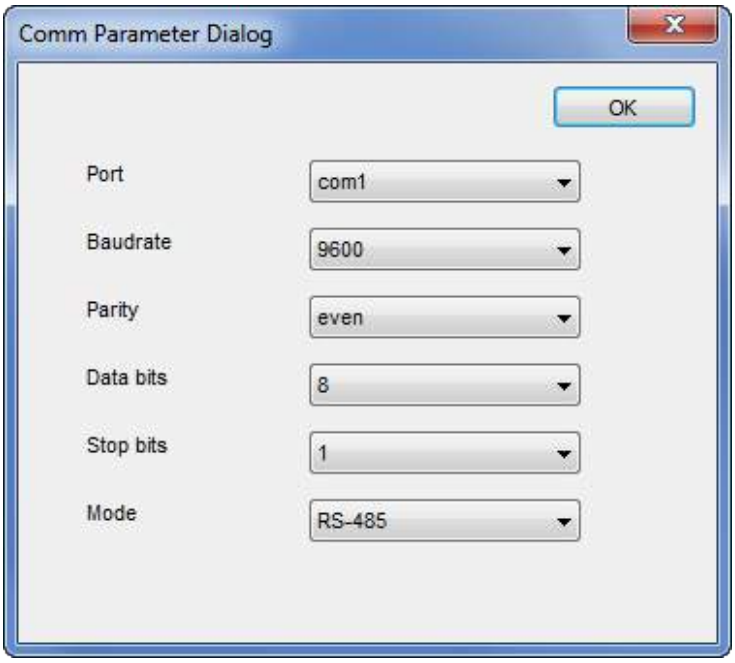
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Node ID	Node ID assigned to the controller device.
Protocol Type	Two protocols are available: • **HCP** • **HCP2**
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
Retry count	Number of times a certain message will be sent to the controller before reporting the communication error status.
PLC Models	PLC model you are going to connect to.

Element	Description
PLC Network	The protocol allows the connection of multiple controllers to one HMI device. To set-up multiple connections, check “PLC network” checkbox and enter the node ID per each slave you need to access. 
Comm...	If clicked displays the communication parameters setup dialog. 

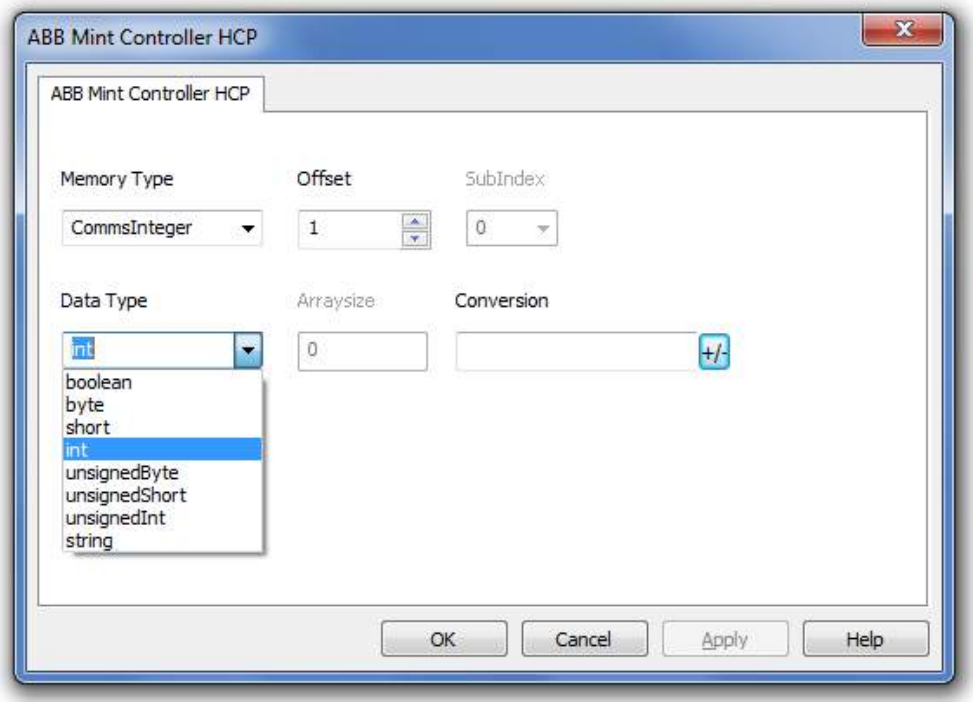
Element	Description
	<table><tr><th>Element</th><th>Description</th></tr><tr><td>Port</td><td>Serial port selection. - COM1= device PLC port. - COM2= computer/printer port. </td></tr><tr><td>Baudrate, Parity, Data Bits, Stop bits</td><td>Serial line parameters.</td></tr><tr><td>Mode</td><td>Serial port mode. Available modes: - RS-232. - RS-485 (2 wires). - RS-422 (4 wires). </td></tr></table>

Data types

The ABB Mint Controller HCP driver provides the support for two Memory Types which are referring to the same physical memory area in the Mint controller:

- **Comms**: should only be used with floating point values. The Mint program on the ABB controller should use COMMS to access this data.
- **CommsInteger**: allows a variety of integer-based data types to be selected.

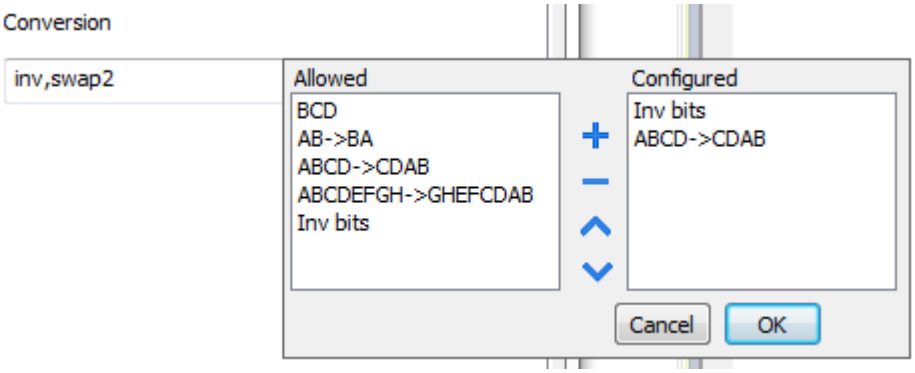
If the Mind controller program uses...	then...
COMMS keyword for a tag setup to use the Commsinteger memory type	only the bottom 23 bits will be accurate (due to floating point precision of the COMMS keyword).
COMMSINTEGER keyword for a tag setup to use the Commsinteger memory type	the value is precise for the full 32 bits.



See "Programming concepts" section in the main manual.

Tag Conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:*

Value	Description
	25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

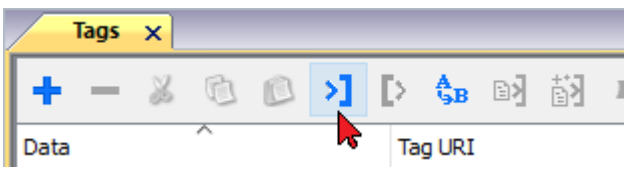
Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

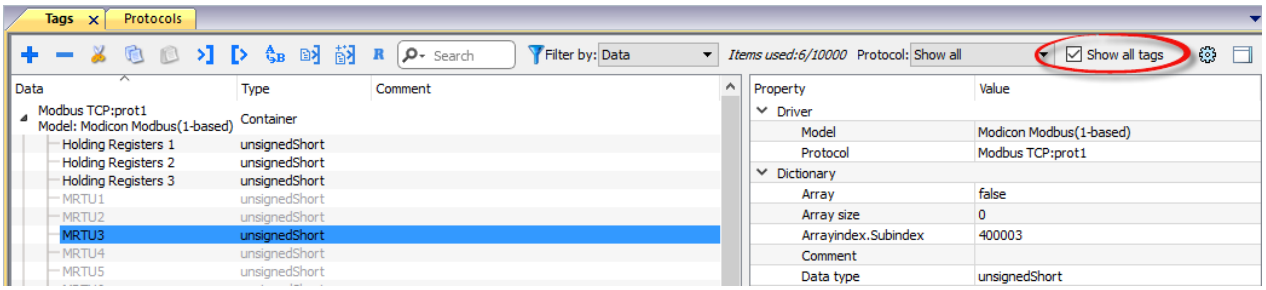
Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.

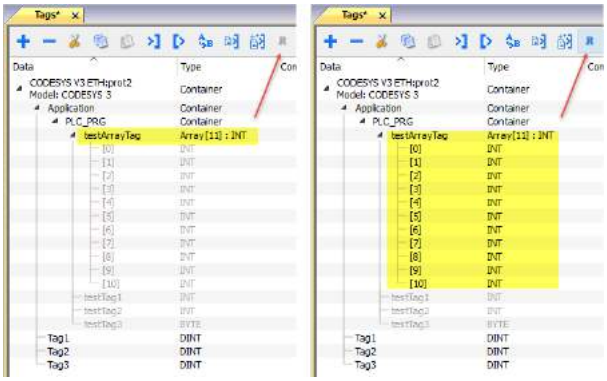
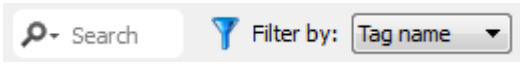


Locate the **.xml** file exported from Tag Editor and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag.

Toolbar item	Description
	Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported by this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Line Error	An error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits).	Check if the communication parameter settings of the controller is compatible with the device communication setup.
Invalid response	The device did received a response with invalid format or contents from the controller.	Ensure the data programmed in the project are consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

BACnet

The BACnet communication driver has been designed to connect HMI devices to BACnet networks and supports IP and MS/TP communication.

The HMI device operates as a BACnet device.

Implementation details

This implementation of the BACnet communication protocol allows integrating HMIs in a BACnet network and exchange data between HMI and other devices connected to the BACnet network. HMIs provide client capability for displaying properties of BACnet objects in real time using BACnet/IP or MS/TP network types.

BACnet communication protocol can be:

- Configured as BACnet IP: communication with BACnet devices is established over Ethernet using HMI Ethernet port;
- Configured as BACnet MS/SP: communication with BACnet devices is established over serial line, using HMI serial port;

Communication protocol configuration allows defining HMI BACnet ID and object name used to identify HMI in BACnet network.

BACnet object properties are reachable from HMI using explicit Tag configuration. A single Tag represents a single property for a BACnet object.

Using the property Present_Value (85) in Tag configuration, the Tag will be connected to the current value of a specific object (for example in the case of analog values, it will be the measured value).

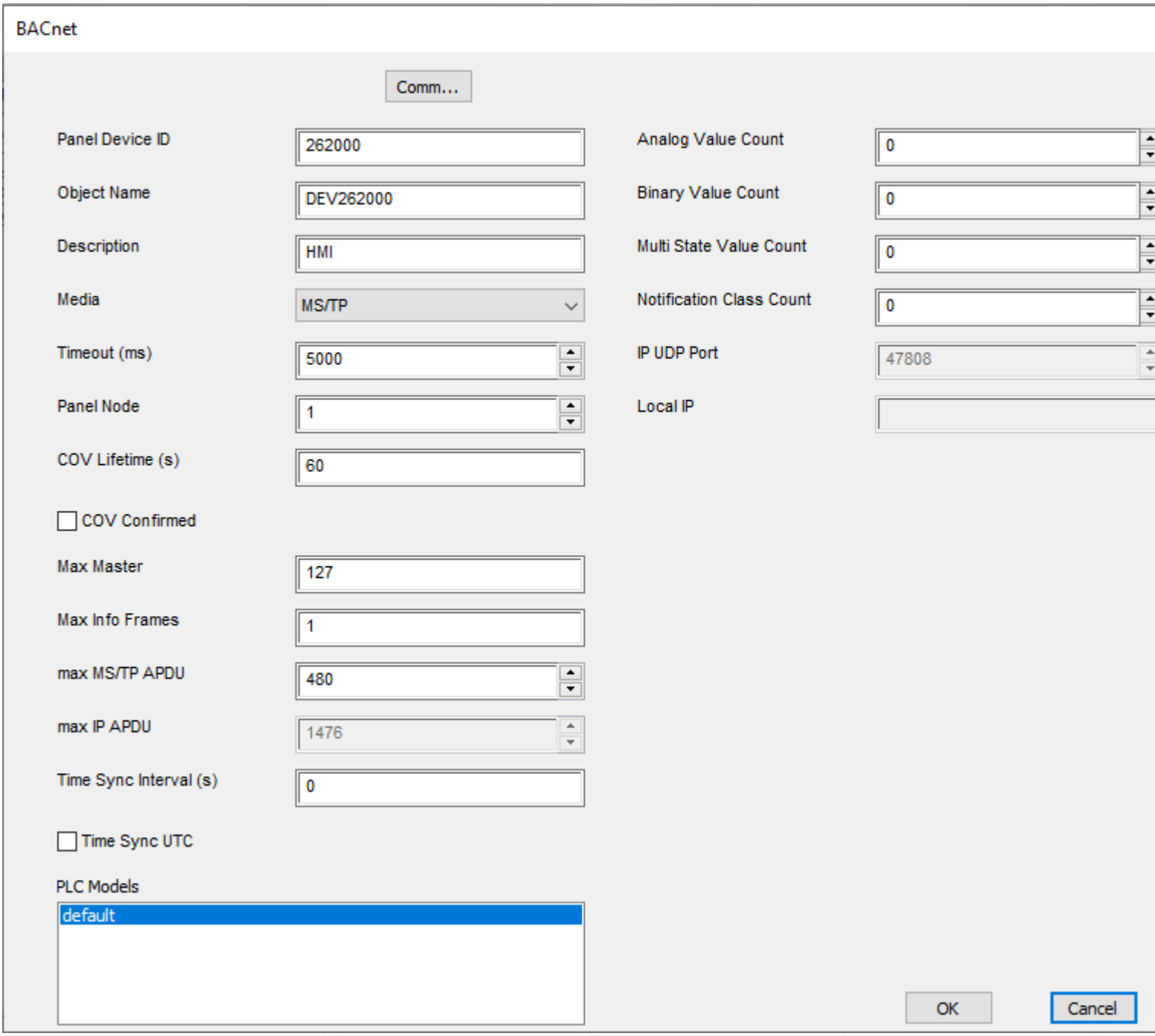
Protocol Editor Settings

Adding a protocol

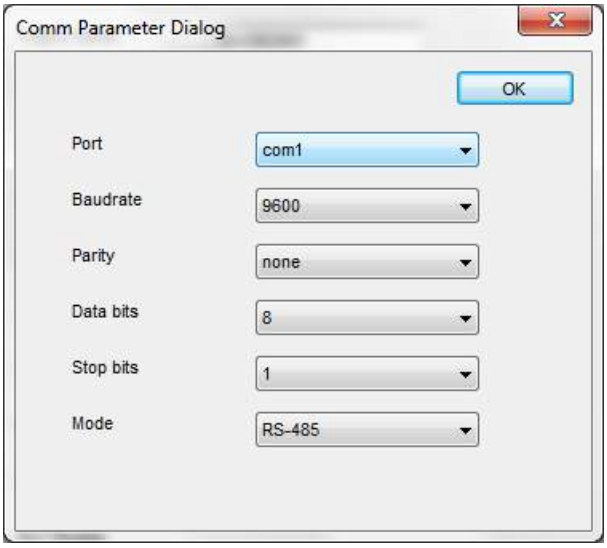
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Panel Device ID	Identifies the HMI device in the network.
Object Name	BACnet Object Name for the HMI device.
Description	HMI device description, for documentation purposes.
Media	Type of communication of the protocol. • **MS/TP**: Master-Slave/Token-Passing communication (RS-485).

Element	Description
	• **IP**: based on standard UDP/IP communication.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the BACnet device.
Panel Node *	MS/TP address. Physical device address on the link; it is not passed through routers.
COV Lifetime (s)	Desired lifetime of the subscription in seconds before the it shall be automatically cancelled.. A value of zero indicates an indefinite lifetime, without automatic cancellation.
Max Master *	Highest allowable address for master nodes. Must be less than or equal to 127.
Max Info Frames *	Maximum number of information frames the node may send before it must pass the token. Max Info Frames may have different values on different nodes and may be used to allocate more or less of the available link bandwidth to particular nodes.
Max MS/TP APDU *	Maximum length of APDU (Application Layer Protocol Data Unit), which means the actual packet length on BACnet network. This value cannot exceed 480 (default value).
Max IP APDU **	Maximum length of APDU (Application Layer Protocol Data Unit), which means the actual packet length on BACnet network. This value cannot exceed 1476 (default value).
Time Sync Interval (s)	Represent the interval between every time synchronization, in seconds. If left to 0, time synchronization is disabled.
Time Sync UTC	Option to synchronize time in UTC format. If disabled, local time format used.
PLC Models	Reserved for future use.
Comm... *	If clicked displays the communication parameters setup dialog. 

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Port</td><td>Communication port.</td></tr>
<tr><td>Baudrate, Parity, Data bits, Stop bits</td><td>Communication parameters.</td></tr>
<tr><td>Mode</td><td>Communication mode. Available modes:
RS-232
RS-485
RS-422</td></tr>
</table>
</td></tr>
<tr><td>Analog Value Count ***</td><td>Number of Analog Value objects to be instanced in BACnet Server.
Min: 0
Max: 200</td></tr>
<tr><td>Binary Value Count ***</td><td>Number of Binary Value objects to be instanced in BACnet Server.
Min: 0
Max: 200</td></tr>
<tr><td>Multi State Value Count ***</td><td>Number of Multi State Value objects to be instanced in BACnet Server.
Min: 0
Max: 200</td></tr>
<tr><td>Notification Class Count ***</td><td>Number of Notifications Class objects to be instanced in BACnet Server.
Min: 0
Max: 200</td></tr>
<tr><td>IP UDP Port **</td><td>Port number for IP communication.</td></tr>
<tr><td>Local IP **</td><td>IP Address of the network adapter to use for protocol. Not required if the device has only one Ethernet adapter.</td></tr>
</table>

Note *: Available only if media is set to **MS/TP**.

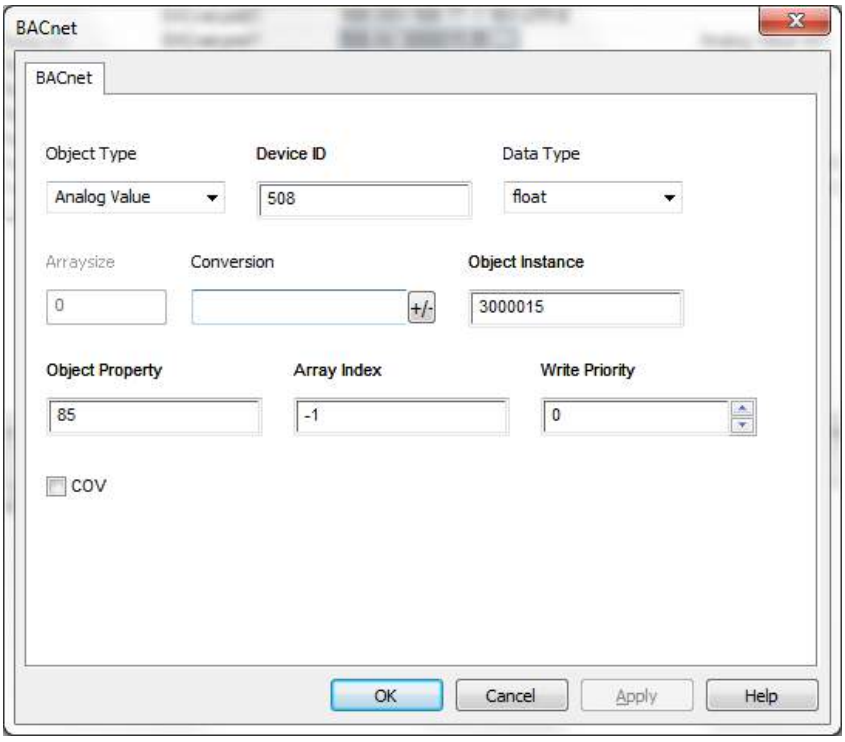
Note **: Available only if media is set to **IP**.

Note ***: Check **Using BACnet Server** chapter.

Tag Editor Settings

*Path: **ProjectView**> **Config** > double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **BACnet** from the **Driver** list: the tag definition dialog is displayed.



Element	Description
Object Type	Type of BACnet object to be referenced. Available object types: • **Device** • **Analog Input** • **Analog Output** • **Analog Value** • **Binary Input** • **Binary Output** • **Binary Value** • **Multi-state Input** • **Multi-state Output** • **Multi-state Value** • **Integer Value** • **Positive Integer Value** • **Large Analog Value**
Device ID	ID of the device containing the object.
Data Type	Data type for display presentation. Available data types: • **boolean**

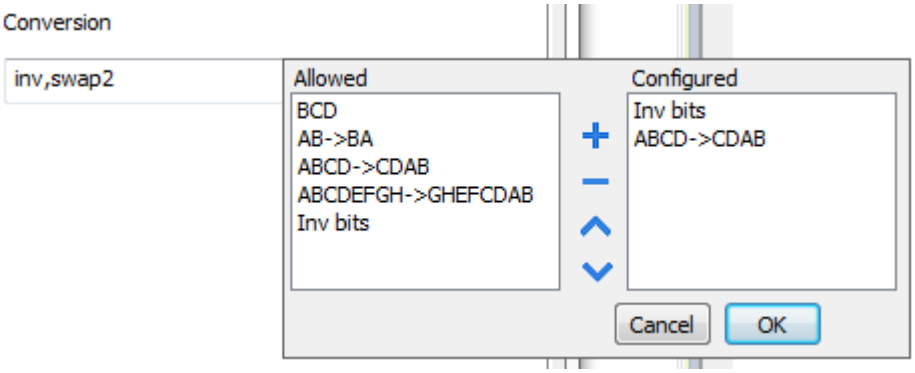
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>

int
unsignedInt
float
double
string
binary
boolean[]

These data types are data types as defined in the software.
The equivalence with BACnet data types is shown in the table:
<table>
<tr><th>BACnet data type</th><th>Software data type</th><th>Notes</th></tr>
<tr><td>BOOLEAN</td><td>Boolean</td><td>-</td></tr>
<tr><td>INTEGER</td><td>Int</td><td>-</td></tr>
<tr><td>UNSIGNED_INTEGER</td><td>unsignedInt</td><td>-</td></tr>
<tr><td>REAL</td><td>Float</td><td>-</td></tr>
<tr><td>BIT_STRING</td><td>boolean-x</td><td>x = size</td></tr>
<tr><td>CHARACTER_STRING</td><td>string-x</td><td>x = size</td></tr>
<tr><td>OCTET_STRING</td><td>binary-x</td><td>x = size</td></tr>
<tr><td>DATE</td><td>int or unsignedInt</td><td>-</td></tr>
<tr><td>TIME</td><td>int or unsignedInt</td><td>-</td></tr>
<tr><td>BACnetObjectIdentifier</td><td>int or unsignedInt</td><td>Use conversions instance and objType for proper display</td></tr>
</table>
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the tag.</td></tr>
</table>

Element	Description
	 Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>0001110010111011011001000101101000011100101011000001
→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.</td></tr>
<tr><td>Object Instance</td><td>BACnet ID of the object to be referenced.</td></tr>
<tr><td>Object Property</td><td>Numeric value of the property to be referenced (example: the value 85 means present-value for most standard objects).
The table below specifies all the BACnet Object Properties.
<table>
<tr><th>Property</th><th>Value</th><th>Property</th><th>Value</th><th>Property</th><th>Value</th><th>Property</th><th>Value</th></tr>
<tr><td>accepted-modes</td><td>175</td><td>effective-period</td><td>32</td><td>max-info-frames</td><td>63</td><td>reason-for-halt</td><td>100</td></tr>
<tr><td>acked-transitions</td><td>0</td><td>elapsed-active-time</td><td>33</td><td>max-master</td><td>64</td><td>recipient-list</td><td>102</td></tr>
<tr><td>ack-required</td><td>1</td><td>error-limit</td><td>34</td><td>max-pres-value</td><td>65</td><td>records-since-notification</td><td>140</td></tr>
<tr><td>action</td><td>2</td><td>event-</td><td>35</td><td>max-</td><td>167</td><td>record-</td><td>141</td></tr>
</table>
</td></tr>
</table>

Element	Description

Property	Value	Property	Value	Property	Value	Property	Value
		enable		segments-accepted		count	
action-text	3	event-state	36	member-of	159	reliability	103
active-text	4	event-time-stamps	130	minimum-off-time	66	relinquish-default	104
active-vt-sessions	5	event-type	37	minimum-on-time	67	required	105
active-cov-subscriptions	152	event-parameters	83	minimum-output	68	resolution	106
adjust-value	176	exception-schedule	38	minimum-value	136	scale	187
alarm-value	6	fault-values	39	minimum-value-timestamp	150	scale-factor	188
alarm-values	7	feedback-value	40	min-pres-value	69	schedule-default	174
all	8	file-access-method	41	mode	160	segmentation-supported	107
all-writes-successful	9	file-size	42	model-name	70	setpoint	108
apdu-segment-timeout	10	file-type	43	modification-date	71	setpoint-reference	109

Element	Description

Property	Value	Property	Value	Property	Value	Property	Value
apdu-timeout	11	firmware-revision	44	notification-class	17	slave-address-binding	171
application-software-version	12	high-limit	45	notification-threshold	137	setting	162
archive	13	inactive-text	46	notify-type	72	silenced	163
attempted-samples	124	in-process	47	number-of-APDU-retries	73	start-time	142
auto-slave-discovery	169	input-reference	181	number-of-states	74	state-text	110
average-value	125	instance-of	48	object-identifier	75	status-flags	111
backup-failure-timeout	153	integral-constant	49	object-list	76	stop-time	143
bias	14	integral-constant-units	50	object-name	77	stop-when-full	144
buffer-size	126	last-notify-record	173	object-property-reference	78	system-status	112
change-of-state-count	15	last-restore-time	157	object-type	79	time-delay	113
change-of-state-time	16	life-safety-alarm-values	166	operation-expected	161	time-of-active-time-reset	114

Element	Description

Property	Value
client-cov-increment	127
configuration-files	154
controlled-variable-reference	19
controlled-variable-units	20
controlled-variable-value	21
count	177
count-before-change	178
count-change-time	179
cov-increment	22
cov-period	180
cov-resubscri	128

Property	Value
limit-enable	52
limit-monitoring-interval	182
list-of-group-members	53
list-of-object-property-references	54
list-of-session-keys	55
local-date	56
local-time	57
location	58
log-buffer	131
log-device-object-property	132
log-enable	133

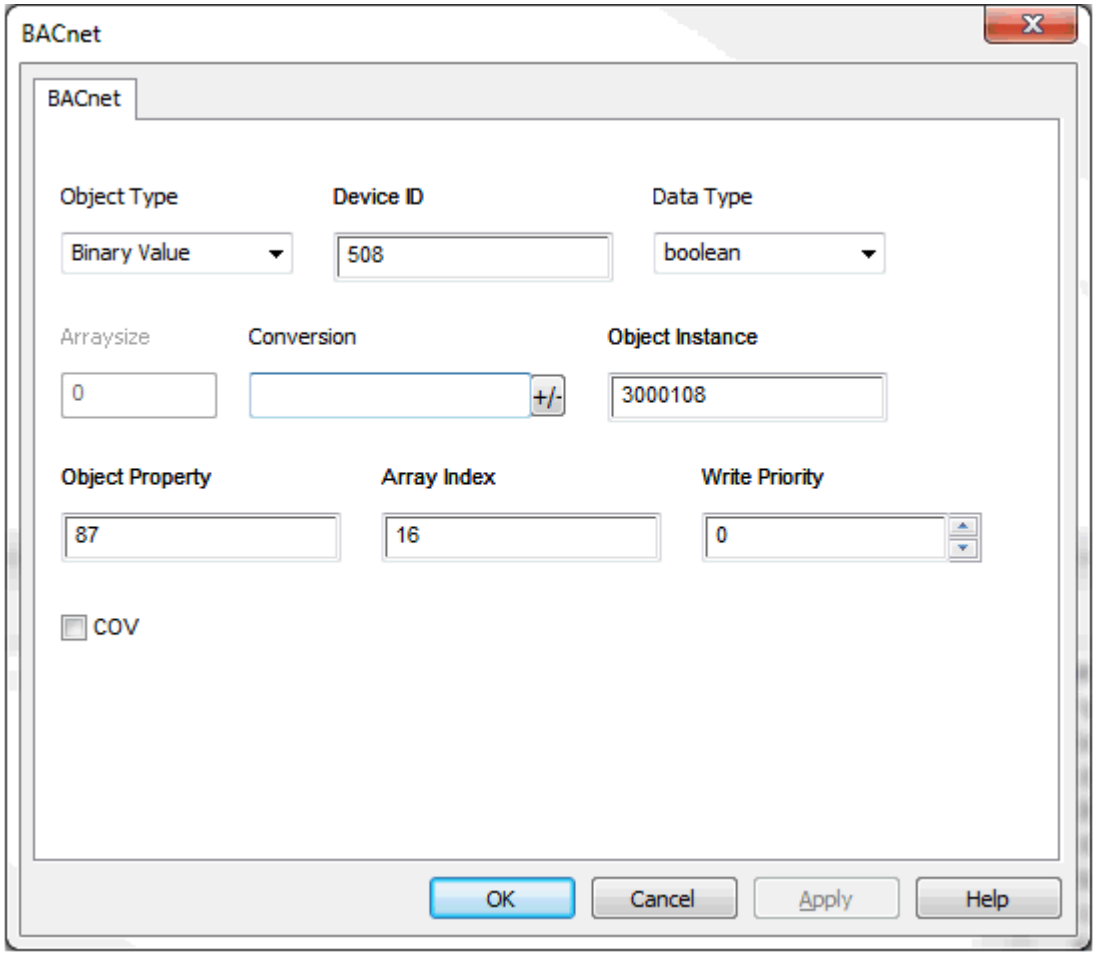
Property	Value
optional	80
out-of-service	81
output-units	82
polarity	84
prescale	185
present-value	85
priority	86
pulse-rate	186
priority-array	87
priority-for-writing	88
process-identifier	89

Property	Value
time-of-state-count-reset	115
time-synchronization-recipients	116
total-record-count	145
tracking-value	164
units	117
update-interval	118
update-time	189
utc-offset	119
valid-samples	146
value-before-change	190
value-set	191

Element	Description

Property	Value	Property	Value	Property	Value	Property	Value
ption-interval							
database-revision	155	log-interval	134	profile-name	168	value-change-time	192
date-list	23	logging-object	183	program-change	90	variance-value	151
daylight-savings-status	24	logging-record	184	program-location	91	vendor-identifier	120
deadband	25	low-limit	59	program-state	92	vendor-name	121
derivative-constant	26	maintenance-required	158	proportional-constant	93	vt-classes-supported	122
derivative-constant-units	27	manipulated-variable-reference	60	proportional-constant-units	94	weekly-schedule	123
description	28	manual-slave-address-binding	170	protocol-object-types-supported	96	window-interval	147
description-of-halt	29	maximum-output	61	protocol-revision	139	window-samples	148
device-address-binding	30	maximum-value	135	protocol-services-supported	97	zone-members	165

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Property</th><th>Value</th><th>Property</th><th>Value</th><th>Property</th><th>Value</th><th>Property</th><th>Value</th></tr>
<tr><td>device-type</td><td>31</td><td>maximum-value-timestamp</td><td>149</td><td>protocol-version</td><td>98</td><td></td><td></td></tr>
<tr><td>direct-reading</td><td>156</td><td>max-apdu-length-accepted</td><td>62</td><td>read-only</td><td>99</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Array Index</td><td>Index for subscribing elements in BACnet arrays.
-1 means read all elements
0 to n means read the specified element
Priority Array example
To read a priority array object it is necessary to set Object Property = 87 and Array Index has to refer to the priority item to be read.
The following figure shows how to read the 16th item of a priority array.</td></tr>
</table>

Element	Description
	
Write Priority	Write requests priority level. The value is in the range 1-16. 0 is interpreted as 16.
COV	Enable the Change Of Value notification.

Clear/Set Priority

The system offers actions for a more flexible handling of Write Priority.

Action	Description
BACnetClearPriority	Clears the priority array at the position associated to the BACnet tag passed as parameter. This action has immediate effect on the BACnet device.
BACnetClearAllPriorities	Clears all positions in the priority array. This action has immediate effect on the BACnet device.
BACnetSetPriority	Overrides the Write Priority value configured in the BACnet tag definition. This action has two parameters: • **TagName**: name of the BACnet tag. • **TagPriority**: new value of Write Priority for the BACnet tag passed as parameter. This action only overrides the value of Write Priority in the BACnet tag definition and does not perform any communication with the BACnet device. Any write command that will be performed to the Present Value property of the BACnet device identified by the tag, will be performed using the new Write Priority value. The priority value will be valid until: • A new call to the BACnetSetPriority action changes it. • The HMI device is restarted. The value of WritePriority defined in the project is valid in this case.

Tag Import

BACnet object information can be imported from BACnet EDE (Engineering Data Exchange) files. The EDE file must have the .csv extension.

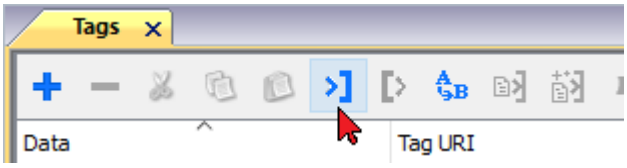
The importer uses the characters “,” and “;” as delimiters. They are considered as reserved characters and you cannot use them in file name.

Use the hierarchical importer to have a ordered list of BACnet objects and properties.

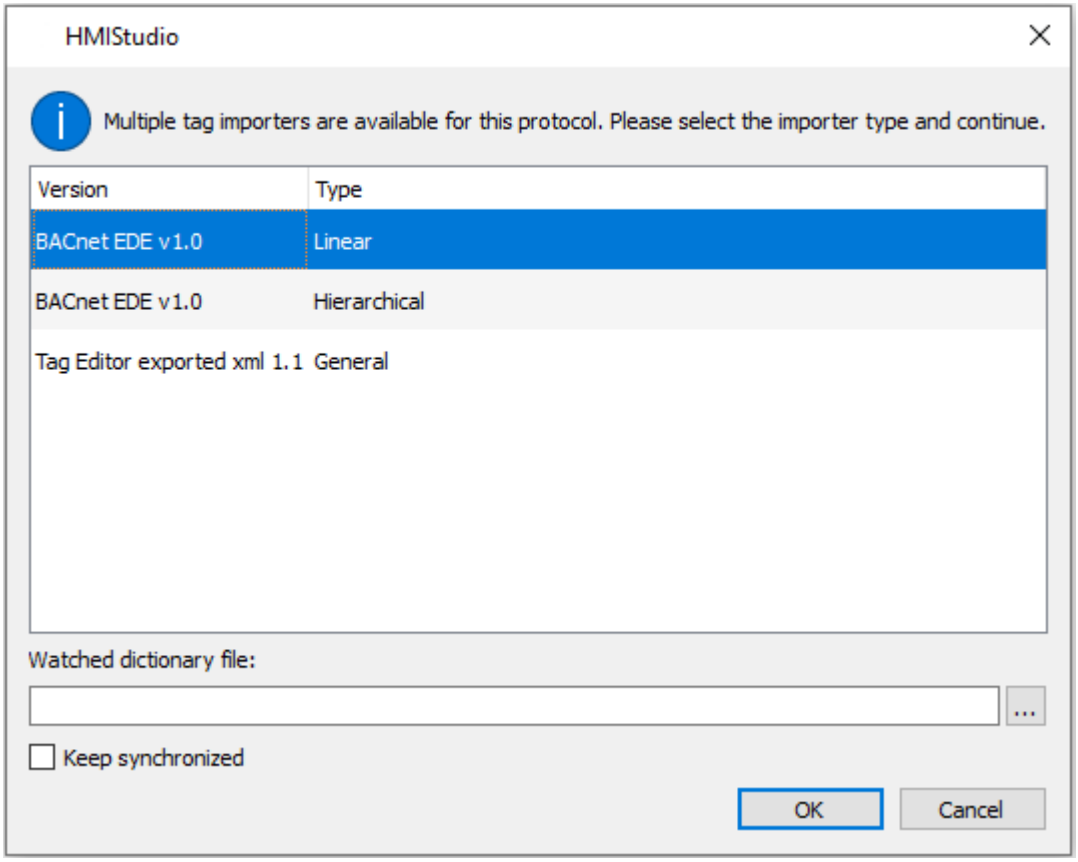
Tags will be created using the string specified in the column object-name of the EDE file. The importer will add the device ID as a prefix to avoid duplication of tag names.

Note: The importer will ask to locate the State-Texts, Unit-Texts and Object-Types files. Click Cancel to ignore.

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



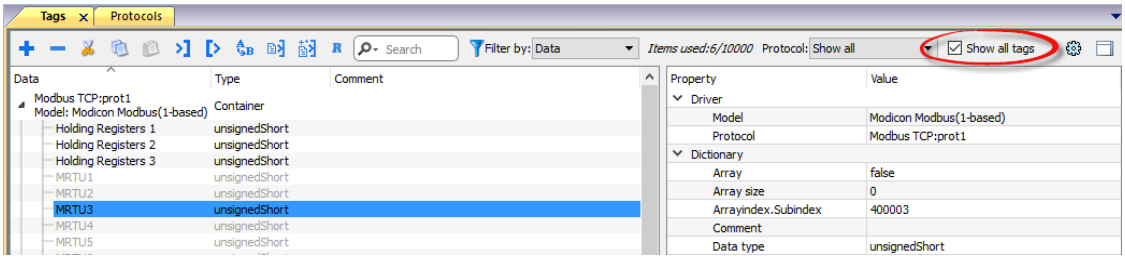
The following dialog shows which importer type can be selected.

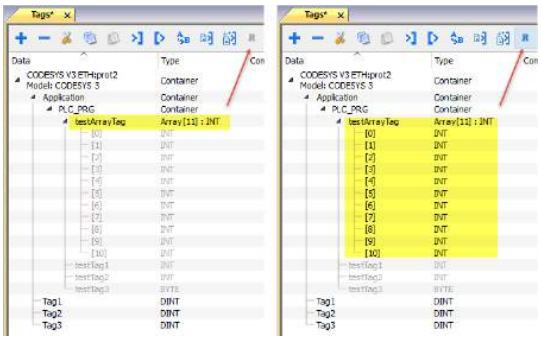
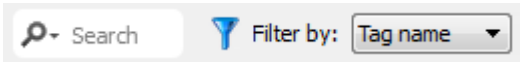


Importer	Description
BACnet EDE v1.0 Linear	Requires a **.csv** file. All variables will be displayed at the same level.
BACnet EDE v1.0 Hierarchical	Requires a **.csv** file. All variables will be displayed according to BACnet EDE Hierarchical view.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

For tags referring to BACnet objects of type Calendar or Schedule the tag refresh rate is set to “Manual”.

The following BACnet object properties are required for operation of the widgets.

Object	Tags to import
Calendar	Date_List
Schedule	Weekly_Schedule Exception_Schedule Default_Value Effective_Period

DEVICE Object Properties

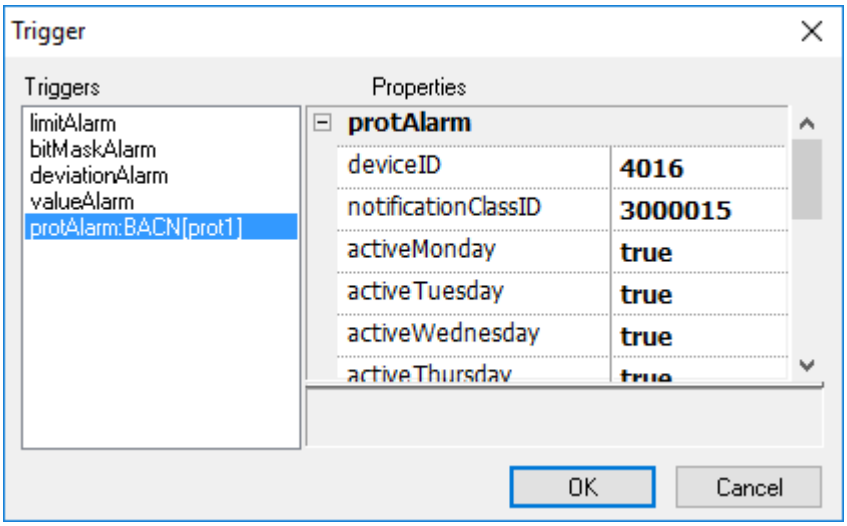
A BACnet network scanner can detect properties when exploring the network and obtaining data from HMI device.

This are the supported DEVICE object properties:

Property	Description
Object_Identifier	BACnetObjectIdentifier
Object_Name	CharacterString
Object_Type	BACnetObjectType
System_Status	BACnetDeviceStatus
Vendor_Name	CharacterString
Vendor_Identifier	Unsigned16
Model_Name	CharacterString
Firmware_Revision	CharacterString
Application_Software_Version	CharacterString
Protocol_Version	Unsigned
Protocol_Revision	Unsigned
Protocol_Services_Supported	BACnetServicesSupported
Protocol_Object_Types_Supported	BACnetObjectTypesSupported
Object_List	BACnetARRAY[N]of BACnetObjectIdentifier
Max_APDU_Length_Accepted	Unsigned
Segmentation_Supported	BACnetSegmentation
APDU_Timeout	Unsigned
Number_Of_APDU_Retries	Unsigned
Device_Address_Binding	List of BACnetAddressBinding
Database_Revision	Unsigned

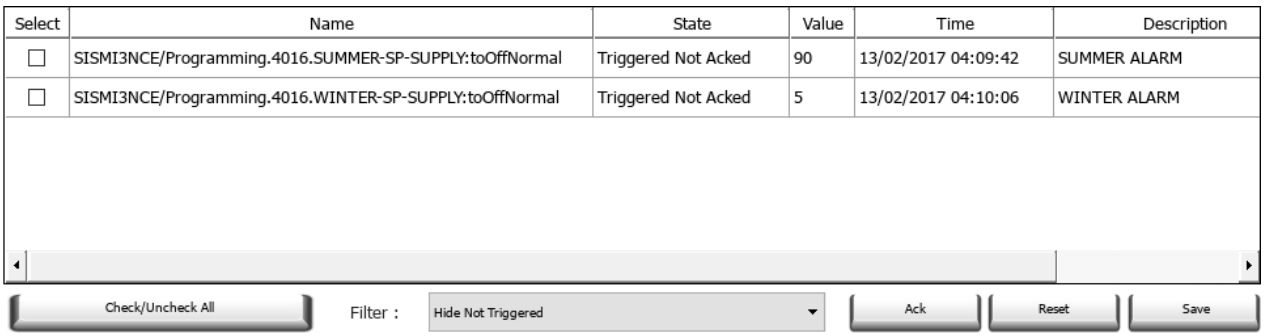
BACnet Alarm Events

The special “protAlarm:BACN” trigger mode, available from the Alarms Editor, give the possibility to receive alarm events from the BACnet native alarms module.

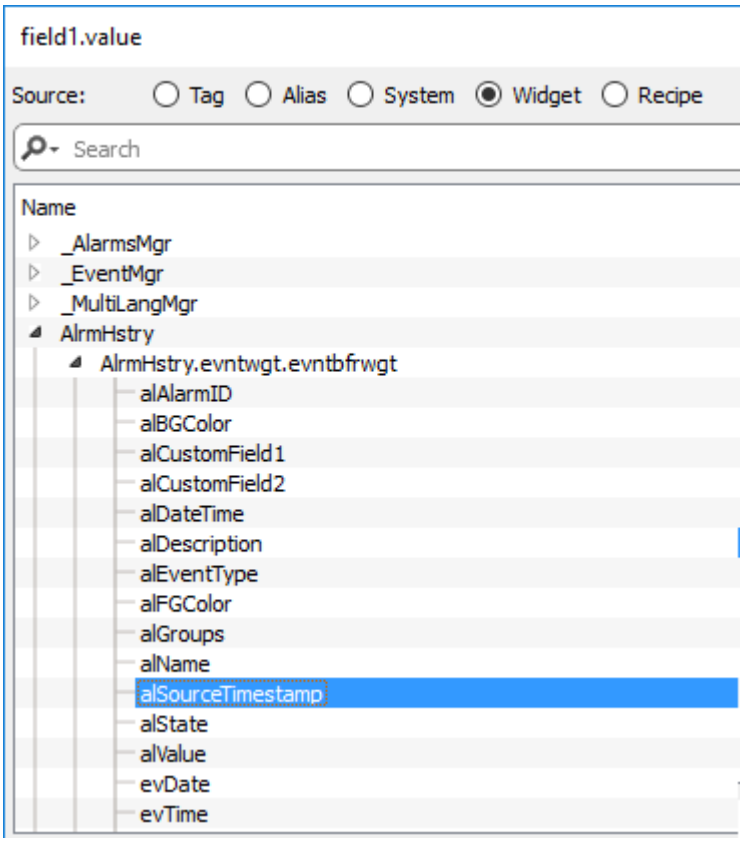


Property	Description
deviceID	Identifies the BACnet device in the network.
notificationClassID	Notification Class ID to subscribe for the alarm events retrieving
processID	Not used
activeMonday **activeTuesday** **activeWednesday** **activeThrusday** **activeFriday** **activeSaturday** **activeSunday**	Define in which days keep active the alarm events subscription • False Subscription not active • True Subscription active
startHour **startMinute** **startSecond** **endHour** **endMinute** **endSecond**	Define the time window where the alarm events subscription will be active

The alarm widgets will report the alarm information that are provided from the BACnet device.



When the special “protAlarm:BACN” trigger mode is used, the widget of the active alarms show the timestamp provided from the BACnet device while the widget of the historical alarms show the timestamp of when the alarm events are received from the HMI device. Generally, both timestamps are the same but if you need to show the timestamp from the BACnet device even inside the widget of the historical alarms you can add a new column configured to use the “alSourceTimestamp” value from the alarm history widget.

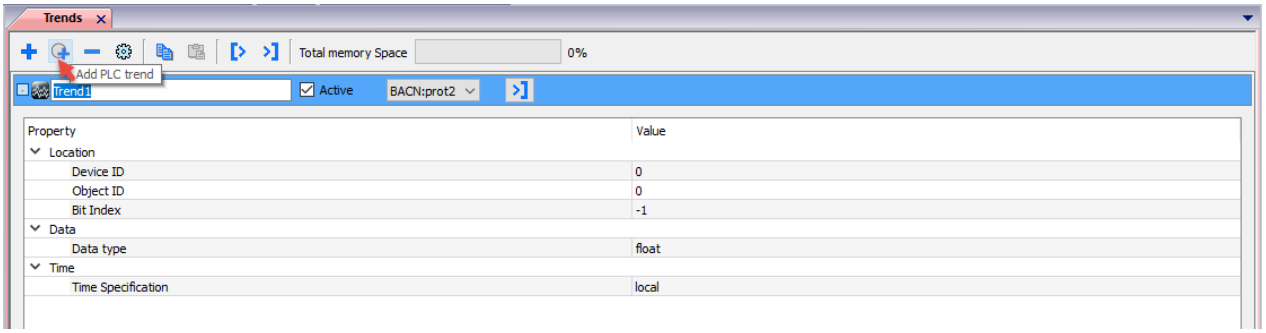


BACnet alarm is a special alarm that require a double space to be stored inside the events buffer. This means, for example, if the events buffer is configured to contain 1.000 events only the last 500 BACnet events will be stored.

BACnet Trend Buffer

To use a BACnet trend object as a trend buffer:

1. Open the Trends Editor
2. Click the "Add PLC Trend" button (This button is enabled only when at least one BACnet protocol is configured)
3. Configure the below parameters to identify the BACnet trend object to use.



Property	Description
Device ID	Identifies the BACnet device in the network.
Object ID	BACnet ID of the trend object to be referenced.
Bit Index	When the data type is boolean, it is the index to select the bit to use inside the BACnet bit_string. It is not used with the other data types.
Data type	Specify the type of data of the BACnet trend object. The supported data types are: • boolean • int • unsignedInt • float
Time Specification	Time format used inside the selected BACnet trend object • local • global (UTC)

The trend buffer thus configured can then be used inside any trend widgets.

BACnet Calendar Widget

Use Calendar widget to display content of a BACnet Calendar object.

Property	Description
Date_List	Connect to the "Date_List" tag of a BACnet calendar object in ReadOnly or Read/Write.  Note: it can be connected to an alias which indexes a list of BACnet calendar Date_List(s), in order to use one calendar widget for more than one calendar object.

Operation of Calendar Widget

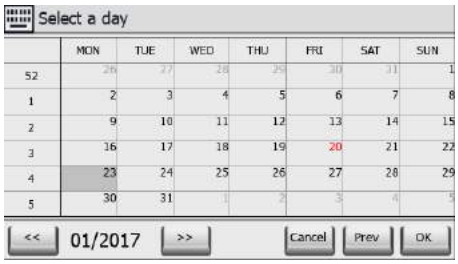
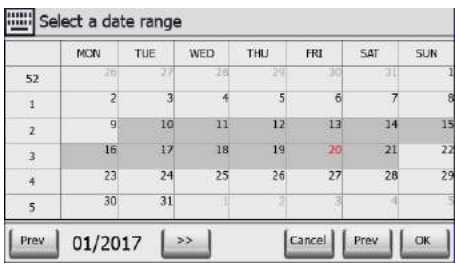
The widget shows data for one month.

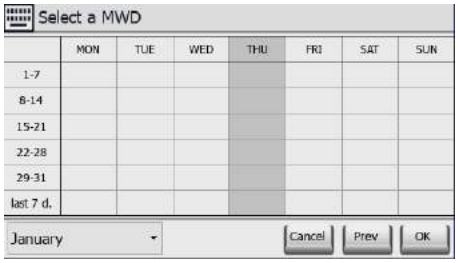
Use the < and > buttons to select the month to be displayed. The date of first day of the month is shown.

Swing gesture can be used on the widget to select the date.

New

Press the button “New” to enter a new calendar item. The button is active only if the tag associated to the calendar has been configured as Read/Write.

Calendar item	Description
Single	Click on a day to select a single day into the calendar 
Range	Click on the first day and on the last day to select a range of days into the calendar. • Single click on a day to change previous selected last day of the range. • Double click on a day to change previous selected first selected day of the range. 

Calendar item	Description
MWD	Select a Day or a Week for each year or each month. 

Clear All

Press the button “Clear All” to clear the content of the calendar object. The button is active only if the tag associated to the calendar has been configured as Read/Write. The button is configured to react to an onMouseHold event, to reduce risk of data loss.

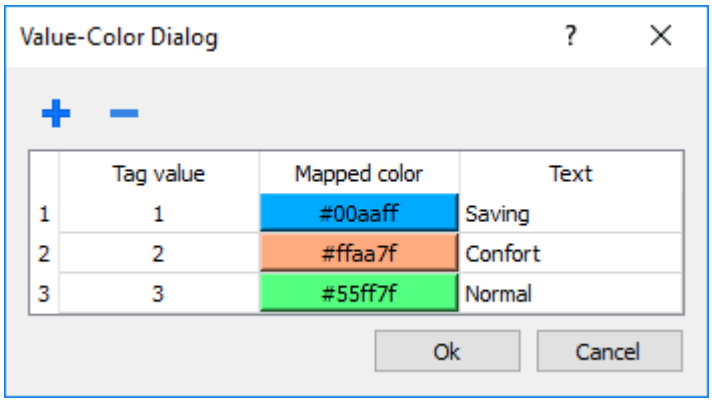
Refresh

Press the “Refresh” button to start a manual refresh of the data of the widget. Always press the Refresh button after entering data in the calendar.

BACnet Schedule Widget

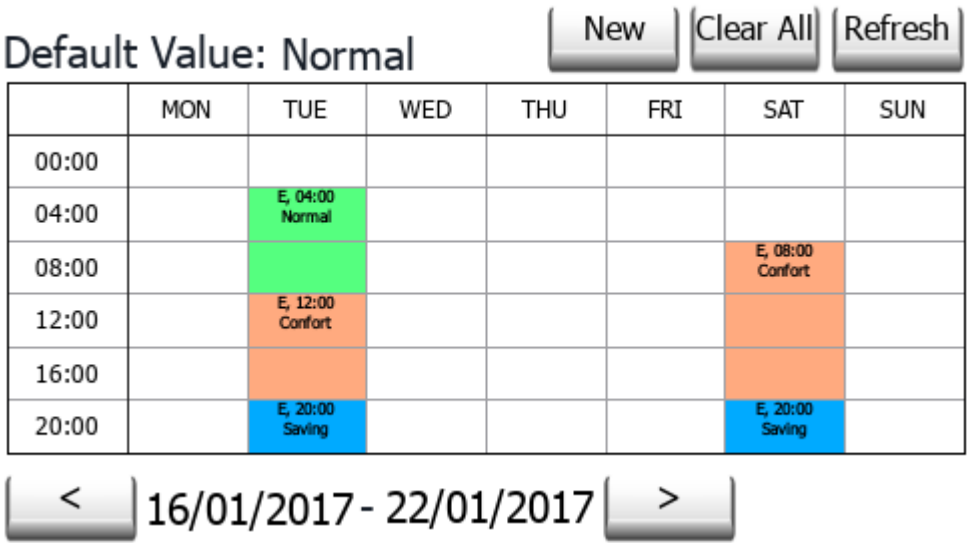
Use Schedule widget to display content of BACnet Schedule object.

Property	Description
Type	Select the type of BACnet object controlled by the schedule. Options are: • Binary • Real • Multistate
Weekly_Schedule	Attach to the Weekly_Schedule tag of the schedule object. The tag can be Read Only or Read/Write.
Exception_Schedule	Optionally attach to the Exception_Schedule tag of the schedule object. The tag can be Read Only or Read/Write. Only attach this property if exceptions are used.
Default_Value	Optionally attach to the Default_Value tag of the schedule object. The tag can be Read Only or Read/Write. Only attach this property if default values are used.
Cal. 0 (Date_List)	Optionally attach to the Date_List tag of the schedule widget in Read Only mode. Use this options to show the “calendar reference” exceptions. Note: An exception can be a single date, a date range, a mwd or a calendar reference. In this last case, exception_list does not contain

Property	Description
	the date information, but only time-value-priority and a reference to the calendar. The date_list needed to show the scheduling into the widget is stored into the relative BACNCalendar, and this is why we need this datalink. If there is no need to show calendar exceptions in the schedule, this property can be left void. Note: If it is not attached to a calendar, it is not possible to insert calendar exception. See BACNSchedKeypad for details.
Cal. 0 (Object_Name)	Optionally attach to the property of the calendar. This name is used to identify the calendar in the BACNSchedKeypad used to insert calendar exceptions. If Object_Name is not attached, the calendar is identified with its instance number. This property is used only if a Cal. 0 (Date_List) is attached to a calendar.
Cal. 1 (Date_List)	Option for a second calendar.
Cal. 1 (Object_Name)	Option for a second calendar.
Value-color-text Map	Defines the association value – Color/Text shown in the schedule. Use this option to define all possible values available in the BACNSched keypad. 

Operation of Schedule Widget

The widget shows data for one week.

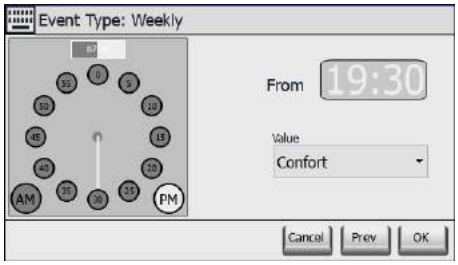
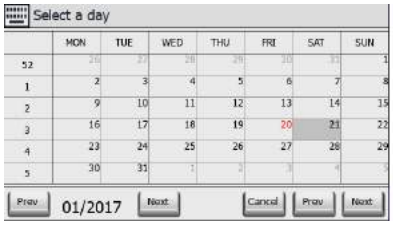
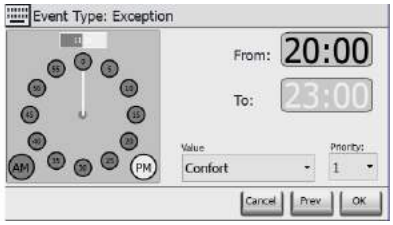


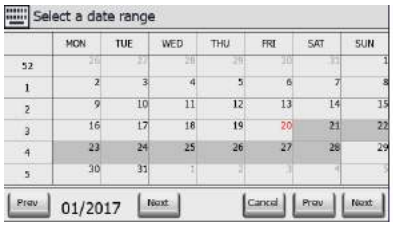
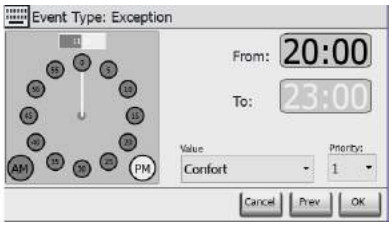
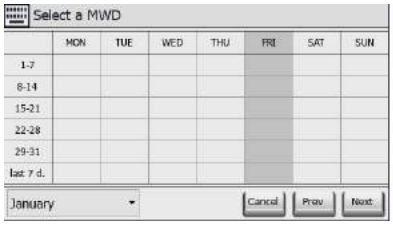
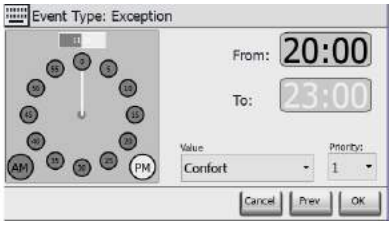
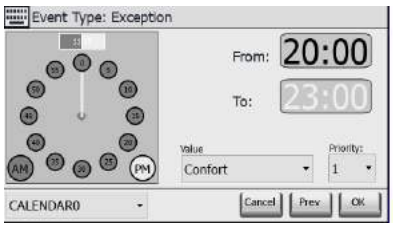
Use the < and > buttons to select the week to be displayed. The date of first day and last day of the week is shown.

Swing gesture can be used on the widget to select the date.

New

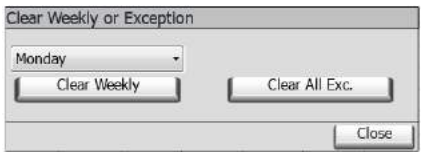
Press the button “New” to enter a new schedule item. The button is active only if the tag associated to Weekly Schedule or Exception Schedule has been configured as Read/Write.

Schedule item	Description
Weekly	Select the day and click Weekly button, the following dialog box appears. Then select the desired value and the time when it should be set. Press OK to confirm the new item. 
Exception Single	Click on a day to select a single day into the calendar. On the next dialog select the time window, the desired value and its priority.  
Exception Range	Click on the first day and on the last day to select a range of days into the calendar.

Schedule item	Description
	• Single click on a day to change previous selected last day of the range. • Double click on a day to change previous selected first selected day of the range. On the next dialog select the time window, the desired value and its priority.  
Exception MWD	Select a Day or a Week for each year or each month. On the next dialog select the time window, the desired value and its priority.  
Exception Cal Ref	This option is available only if scheduler is linked to a calendar (configured as Read/Write) Select the time window, the desired value and its priority. Value will set on all days defined from the calendar. If there are more calendars associated with Scheduler widget, select the calendar to use. 

Clear All

Press the button “Clear All” to clear the content of the schedule object. The button is active only if the tag associated to the calendar has been configured as Read/Write. The button is configured to react to onMouseClick and onMouseHold events. The onMouseHold event will clear all data in the schedule. The onMouseClick event will recall a dialog box for selection of data to clear. It is needed to choice to clear weekly data or exception data.



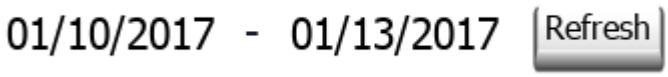
Refresh

Press the “Refresh” button to start a manual refresh of the data of the widget. Always press the Refresh button after entering data in the schedule.

BACnet Effective Period Widget

Use the Effective Period widget to feed information to the Effective_Period tag of a Schedule object, if this is requested.

Property	Description
BACnet Effective_ Period	Attach to the Effective_Period tag of the Schedule object



Operation of Effective Period Widget

The widget shows starting date and end date for the period.

Click on the area showing the dates to activate the data entry procedure showing the keypad BACNDateRange.

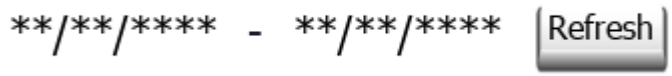
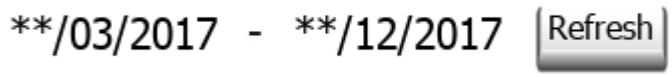
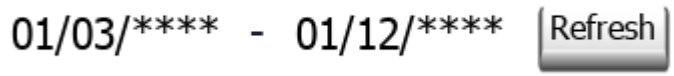
The keypad shows data for one month.

Use the < and > buttons to select the month to be displayed. The date of first day of the month is shown.

You may use the swing gesture on the widget to select the date.

Select the period clicking of first day and last day of the period. The Effective_Period is show with a different color.

The keypad offers three predefined options:

Option	Description
Always	The schedule will be always active. 
All Month	The selected period will be extended to all months. 
All Year	The selected period will be extended to all years. 

Refresh

Press the “Refresh” button to start a manual refresh of the data of the widget. Always press the Refresh button after entering data in the widget.

BACnet Keypads

BACnet widgets require dedicated keypads for data entry.

Keypad	Description
BACNCal	Keypad for BACnet Calendar.
BACNDateRange	Keypad for BACnet Effective_Period.
BACNDefVal	Keypad for default value (embedded in the BACnet Schedule).
BACNSched	Keypad for BACnet Schedule. This keypad is context sensitive. It will show different options depending on the type of schedule.

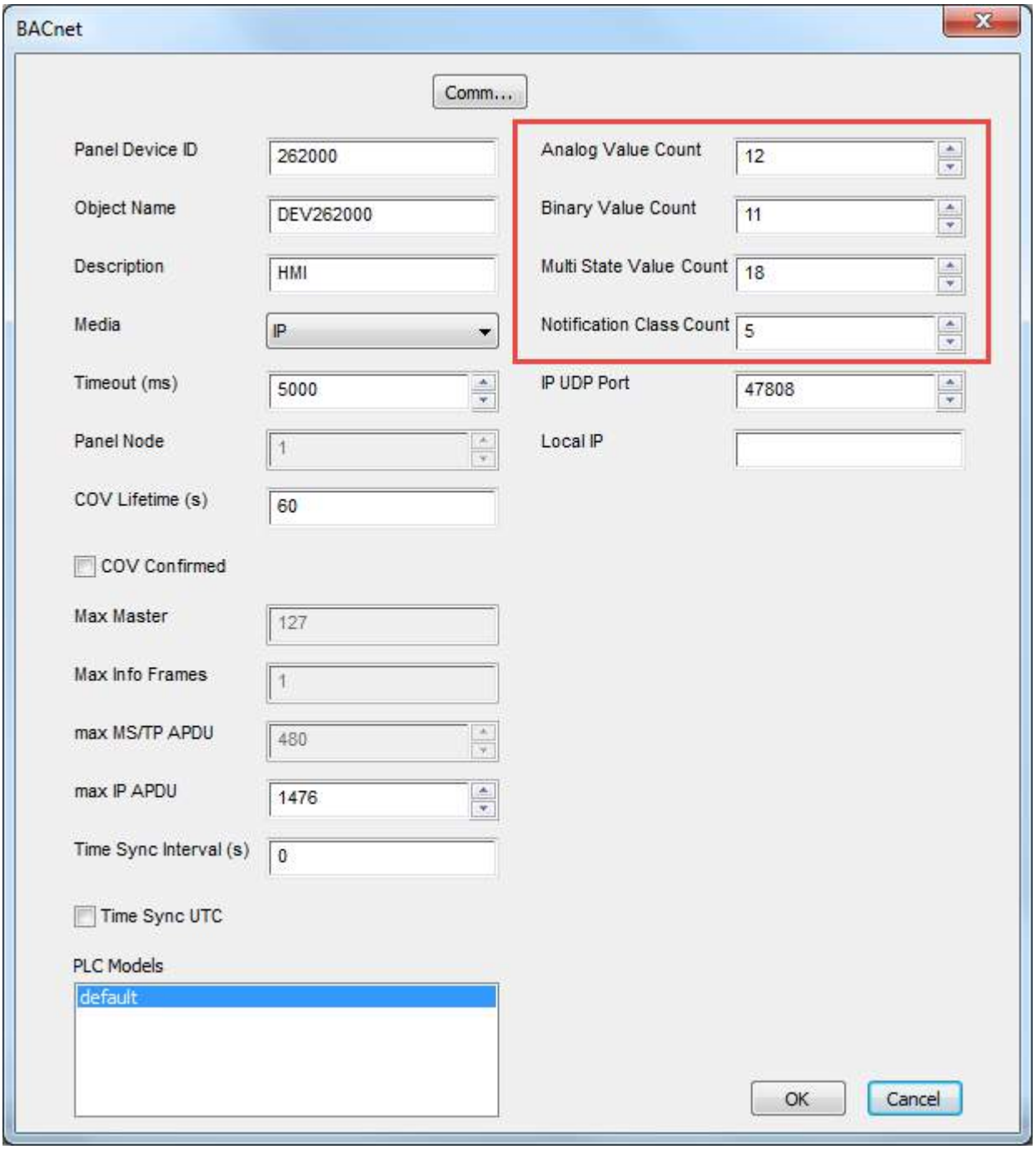
The system is configured to recall the appropriate keypad for each BACnet widget.

Using BACnet Server

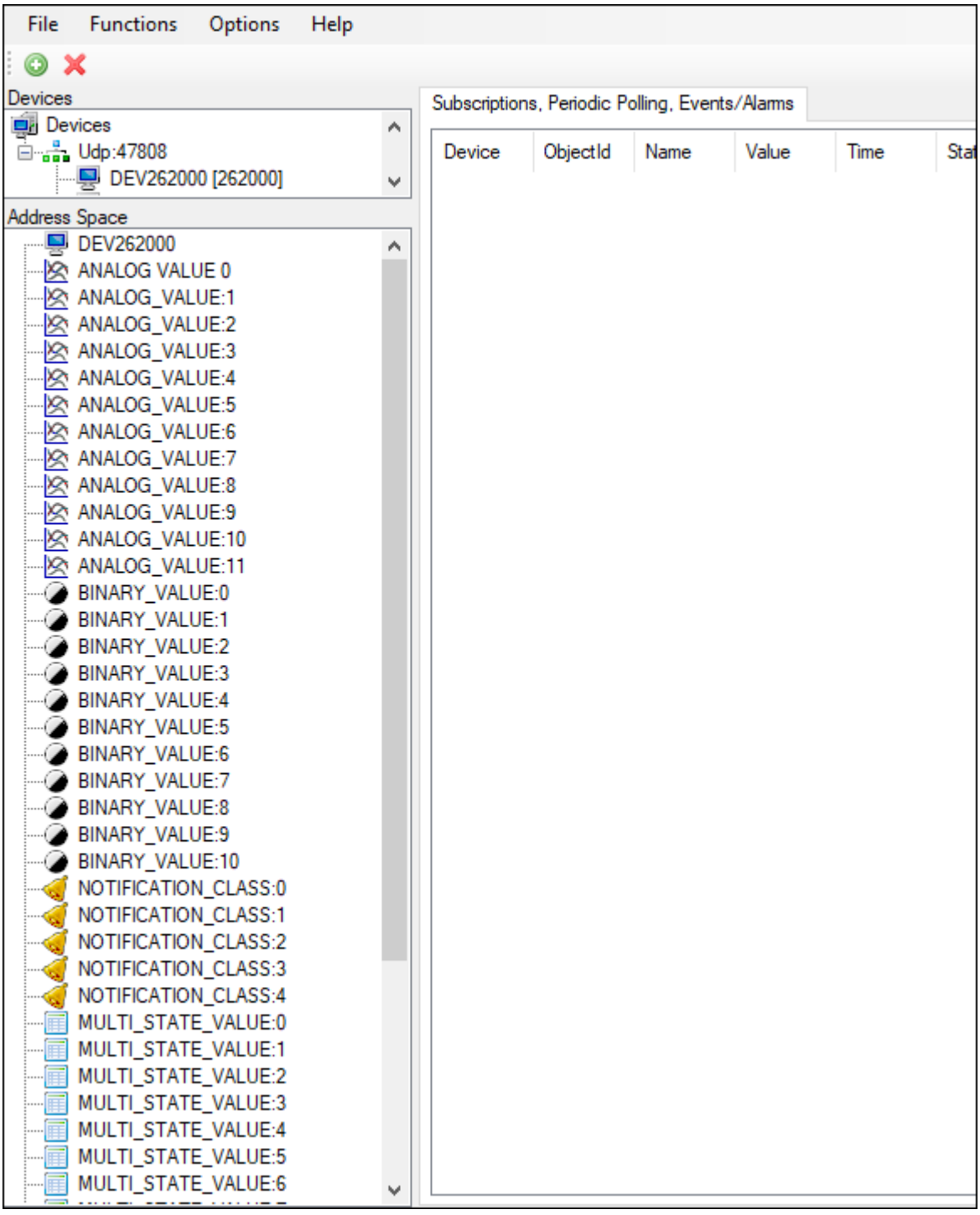
BACnet protocol is capable to act as BACnet Server, by exposing BACnet objects.

To properly setup BACnet Server, it is needed to execute the following steps:

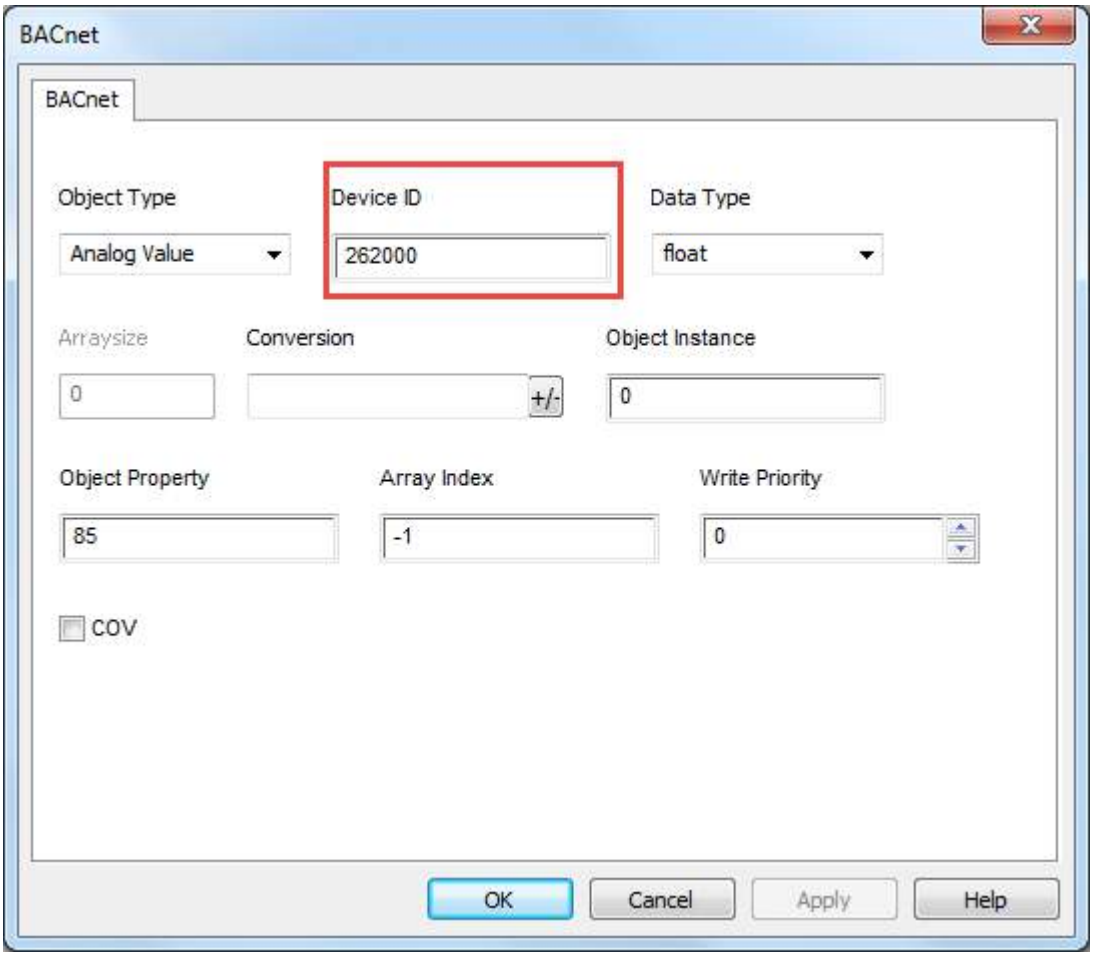
1. Configure objects to expose from **Protocol Editor Settings.**



Note: Objects configured in above image can be discovered by BACnet clients:



2. Create Tags that points to local BACnet objects, setting Device ID as the Device ID configured in Protocol Editor Settings:



Device objects description

Property Name	Code	Default value	Permanent	Note	Data Type
APDU timeout	11	Parameter	Yes		UnsignedInt
Application software version	12		Read-only		String
Database version	155		Read-only		UnsignedInt
Daylight saving status	24		Read-only		Boolean
Read-only	28	Parameter	Yes		String
Device address binding	30		Read-only		String
Firmware revision	44		Read-only		String
Local date	56		Read-only		UnsignedInt
Local time	57		Read-only		UnsignedInt

Property Name	Code	Default value	Permanent	Note	Data Type
Location	58	Parameter	Yes		String
Max APDU length accepted	62		Read-only		UnsignedInt
Max info frames	63	Parameter	Yes	Only if MSTP	String
Max master	64	Parameter	Yes	Only if MSTP	String
Model name	70		Read-only		String
Number of APDU retries	73	Parameter	Yes		UnsignedInt
Object identifier	75	Parameter	Yes		UnsignedInt + Conversion
Object list	76		Read-only		UnsignedInt + Conversion
Object name	77	Parameter	Yes		String
Object type	79		Read-only		UnsignedInt
Protocol object types supported	96		Read-only		Boolean(51)
Protocol revision	139		Read-only		UnsignedInt
Protocol services supported	97		Read-only		Boolean(40)
Protocol version	98		Read-only		UnsignedInt
Segmentation supported	107		Read-only		UnsignedInt
System status	112		Read-only		UnsignedInt
UTC offset	119		Read-only		Int
Vendor identifier	120		Read-only		UnsignedInt
Vendor name	121		Read-only		String

Analog Value objects description

Property Name	Code	Default value	Permanent	Note	Data Type
Acked transitions	0		Read-only		Boolen(3)
COV increment	22	0	Yes		Float
Deadband	25	0	Yes		Float
Description	28	"ANALOG	Yes		String

Property Name	Code	Default value	Permanent	Note	Data Type
		VALUE n”			
Event enable	35	0	Yes		Boolean(3)
Event state	36	0	Read-only		UnsignedInt
Event time stamps	130		Yes		UnsignedInt(3)
High limit	45	0	Yes		Float
Limit enable	52	0	Yes		Boolean(2)
Low limit	59	0	Yes		Float
Notification class	17	4194303	Yes		UnsignedInt
Notify type	72	0	Yes		UnsignedInt
Object identifier	75	2:n	Read-only		UnsignedInt + Conversion
Object name	77	"ANALOG VALUE n”	Yes		String
Object type	79	2	Read-only		UnsignedInt
Out of service	81	0	Yes		Boolean
Present value	85	0			Float
Priority array	87		Read-only		16 Single tag String
Reliability	103	0	Yes		UnsignedInt
Relinquish default	104	0	Yes		Float
Status flags	111		Read-only		Boolean(4)
Time delay	113	0	Yes		UnsignedInt
Units	117	98	Yes		Units

Binary Value objects description

Property Name	Code	Default value	Permanent	Note	Data Type
Acked transitions	0		Read-only		Boolean(3)
Active text	4		Yes		String
Alarm value	6	0	Yes		Boolean
Description	28	"BINARY VALUE n"	Yes		String

Property Name	Code	Default value	Permanent	Note	Data Type
Event enable	35	0	Yes		Boolean(3)
Event state	36	0	Read-only		UnsignedInt
Event time stamps	130		Yes		UnsignedInt(3)
Inactive text	46		Yes		String
Notification class	17	4194303	Yes		UnsignedInt
Notify type	72	0	Yes		UnsignedInt
Object identifier	75	5:n	Read-only		UnsignedInt + Conversion
Object name	77	"BINARY VALUE n"	Yes		String
Object type	79	5	Read-only		UnsignedInt
Out of service	81	0	Yes		Boolean
Polarity	84	0	Yes		UnsignedInt
Present value	85	0			Boolean
Priority array	87		Read-only		16 Single tag String
Reliability	103	0	Yes		UnsignedInt
Relinquish default	104	0	Yes		Boolean
Status flags	111		Read-only		Boolean(4)
Time delay	113	0	Yes		UnsignedInt

Multi State Value objects description

Property Name	Code	Default value	Permanent	Note	Data Type
Acked transitions	0		Read-only		Boolean(3)
Alarm values	7		Yes	Defines number of array elements	UnsignedInt
				Array of alarm values (0:n)	UnsignedInt(n)
Description	28	"MULTI STATE VALUE n"	Yes		String
Event enable	35	0	Yes		Boolean(3)

Property Name	Code	Default value	Permanent	Note	Data Type
Event state	36	0	Read-only		UnsignedInt
Event time stamps	130		Yes		UnsignedInt(3)
Fault values	39		Yes	Defines number of array elements	UnsignedInt
				Array of fault values (0:n)	UnsignedInt(n)
Number of states	74	1	Yes		UnsignedInt
Notification class	17	4194303	Yes		UnsignedInt
Notify type	72	0	Yes		UnsignedInt
Object identifier	75	19:n	Read-only		UnsignedInt + Conversion
Object name	77	"MULTI STATE VALUE n"	Yes		String
Object type	79	19	Read-only		UnsignedInt
Out of service	81	0	Yes		Boolean
Present value	85	0			UnsignedInt
Priority array	87		Read-only		16 Single tag String
Reliability	103	0	Yes		UnsignedInt
Relinquish default	104	0	Yes		UnsignedInt
State text	110		Yes		UnsignedInt
Status flags	111		Read-only		Boolean(4)
Time delay	113	0	Yes		UnsignedInt

Notification Class objects description

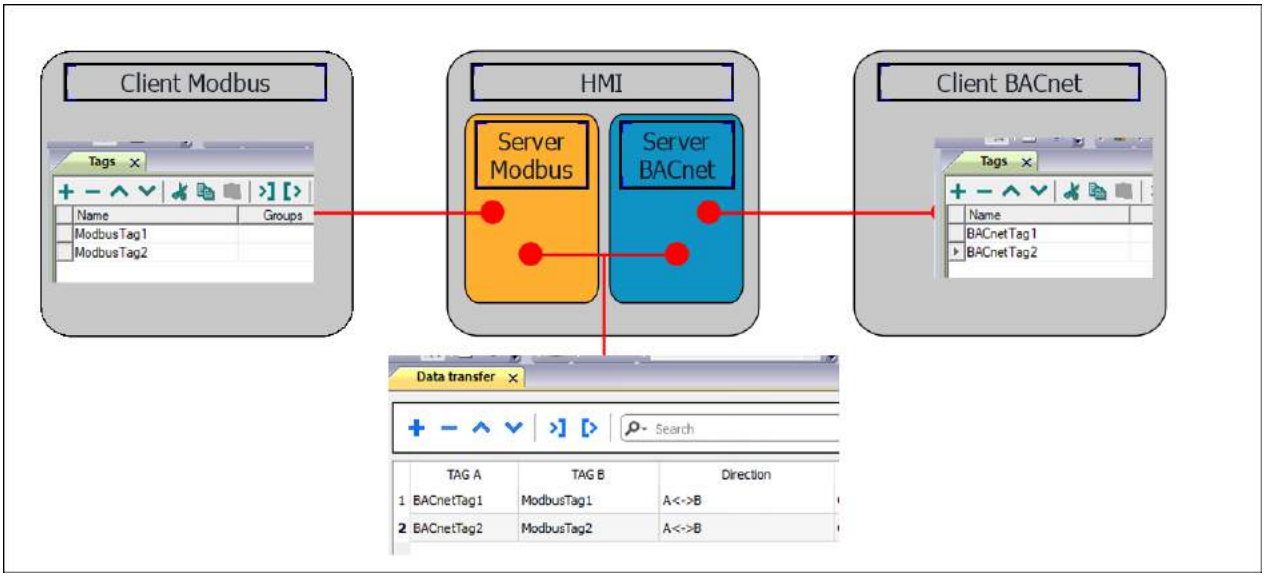
Property Name	Code	Default value	Permanent	Note	Data Type
Ack required	1	0	Yes		Boolean(3)
Description	38	"NOTIFICATION CLASS n"	Yes		String
Notification class	17	4194303	Yes		UnsignedInt
Object identifier	75	15:n	Read-only		UnsignedInt +

Property Name	Code	Default value	Permanent	Note	Data Type
					Conversion
Object name	77	"NOTIFICATION CLASS n"	Yes		String
Object type	79	15	Read-only		UnsignedInt
Priority	86	255,255,255	Yes		UnsignedInt(3)
Recipient list	102		Yes		UnsignedInt(n)

Example of usage

Once BACnet Server Tags are configured, they can be used in combination with Data Transfer feature.

Example: Modbus TCP/RTU Tags can be transferred to BACnet Tags (with same data type). In this way, all BACnet clients can reach BACnet Server and see actual value of Modbus Tags, using BACnet Tags as interface.



JavaScript Interface

Beside Tag interface the user can access the protocol via JavaScript.

Although defined Tags can be accesses by JavaScript too, JavaScript can access directly to a Command interface implemented in protocol. This interface does not require the definition of Tags and is direct to protocol resulting in more efficiency.

The following commands are supported:

Command	Description
scan (minID, maxID, <timeout>)	Executes a scan for devices in the given range.
scan_status	Get the scanning result.
devices	Get the list of devices.
objectCount (deviceID, objectType)	Get the object count of given object types in given device.
objectNames (start, count)	Get the part of object names asked by previous objectCount.
properties (deviceID, objectType, objectInstance)	Get the properties of given device/object.

- **scan**
  Scan the bus to find all present devices having ID in the range minID – maxID.
  To scan the whole network use 0 and 999999 ad minID and maxID.
  The optional timeout can be indicated in milliseconds. Default value is 2000 ms.
  The function starts the scan operation; the function scan_status can be used to know the status of the operation.
  The result of the operation is "**scanning**".
- **scan_status**
  Get the status of last started scan operation. It returns "**scanning**" or "**finished**".
  Scan operation finishes when the timeout time is expired
- **devices**
  Get the list of devices found by latest scan operation. The result is a JSON string containing of each device:
  - device name
  - model name
  - vendor name
  - vendor ID

  Example:

```
{"minID":0,"maxID":999999,"devices":[262000,1101],"deviceNames":
["DEV262000","S01101"],"modelNames":["HMI model","EY-AS525F001"],"vendorNames":
["Company Name","SAUTER"],"vendorIDs":[262,80]}
```

- **objects**
  Get the list of all objects from the devices having the given ID.
  The list is returned as a JSON string containing for each object
  - type
  - instance number

  type can be:

  OBJECT_ANALOG_INPUT = 0,

  OBJECT_ANALOG_OUTPUT = 1,

  OBJECT_ANALOG_VALUE = 2,

OBJECT_BINARY_INPUT = 3,

OBJECT_BINARY_OUTPUT = 4,

OBJECT_BINARY_VALUE = 5,

OBJECT_CALENDAR = 6,

OBJECT_COMMAND = 7,

OBJECT_DEVICE = 8,

OBJECT_EVENT_ENROLLMENT = 9,

OBJECT_FILE = 10,

OBJECT_GROUP = 11,

OBJECT_LOOP = 12,

OBJECT_MULTI_STATE_INPUT = 13,

OBJECT_MULTI_STATE_OUTPUT = 14,

OBJECT_NOTIFICATION_CLASS = 15,

OBJECT_PROGRAM = 16,

OBJECT_SCHEDULE = 17,

OBJECT_AVERAGING = 18,

OBJECT_MULTI_STATE_VALUE = 19,

OBJECT_TRENDLOG = 20,

OBJECT_LIFE_SAFETY_POINT = 21,

OBJECT_LIFE_SAFETY_ZONE = 22,

OBJECT_ACCUMULATOR = 23,

OBJECT_PULSE_CONVERTER = 24,

OBJECT_EVENT_LOG = 25,

OBJECT_GLOBAL_GROUP = 26,

OBJECT_TREND_LOG_MULTIPLE = 27,

OBJECT_LOAD_CONTROL = 28,

OBJECT_STRUCTURED_VIEW = 29,

OBJECT_ACCESS_DOOR = 30,

OBJECT_TIMER = 31,

OBJECT_ACCESS_CREDENTIAL = 32,

OBJECT_ACCESS_POINT = 33,

OBJECT_ACCESS_RIGHTS = 34,

OBJECT_ACCESS_USER = 35,

OBJECT_ACCESS_ZONE = 36,

OBJECT_CREDENTIAL_DATA_INPUT = 37,

OBJECT_NETWORK_SECURITY = 38,

OBJECT_BITSTRING_VALUE = 39,

OBJECT_CHARACTERSTRING_VALUE = 40,

OBJECT_DATE_PATTERN_VALUE = 41,

OBJECT_DATE_VALUE = 42,

OBJECT_DATETIME_PATTERN_VALUE = 43,

OBJECT_DATETIME_VALUE = 44,

OBJECT_INTEGER_VALUE = 45,

OBJECT_LARGE_ANALOG_VALUE = 46,

OBJECT_OCTETSTRING_VALUE = 47,

OBJECT_POSITIVE_INTEGER_VALUE = 48,

OBJECT_TIME_PATTERN_VALUE = 49,

OBJECT_TIME_VALUE = 50,

OBJECT_NOTIFICATION_FORWARDER = 51,

OBJECT_ALERT_ENROLLMENT = 52,

OBJECT_CHANNEL = 53,

OBJECT_LIGHTING_OUTPUT = 54,

OBJECT_BINARY_LIGHTING_OUTPUT = 55,

OBJECT_NETWORK_PORT = 56,

Other types are manufacturer specific.

- **objectCount**
  Returns the number of objects of a defined type in the device having the indicated ID.
  If specified type is -1 the command will return the number of all objects.

  Example:
  objectCount 1101 -1
  77

  objectCount 1101 0
  1

  objectCount 1101 1
  1

  objectCount 1101 3
  2

objectCount 1101 29
16

- **objectNames**
  Returns a part of the objects listed by a previous **objectCount** command, from start index. The list contains only counted objects according to filter previously used
  The list is returned as a JSON string containing for each object
  - type
  - instance number
  - name

  Example:


```
{"deviceID":1101,"objects":[{"type":29,"instance":0,"name":"0x7400000"},
{"type":29,"instance":16,"name":"0x7400010"},
{"type":29,"instance":18,"name":"0x7400012"},
{"type":29,"instance":19,"name":"0x7400013"},
{"type":29,"instance":20,"name":"0x7400014"},
{"type":29,"instance":21,"name":"0x7400015"},
{"type":29,"instance":22,"name":"0x7400016"},
{"type":29,"instance":23,"name":"0x7400017"},
{"type":29,"instance":24,"name":"0x7400018"},
{"type":29,"instance":25,"name":"0x7400019"},
{"type":29,"instance":26,"name":"0x740001a"},
{"type":29,"instance":27,"name":"0x740001b"},
{"type":29,"instance":28,"name":"0x740001c"},
{"type":29,"instance":29,"name":"0x740001d"},
{"type":29,"instance":30,"name":"0x740001e"},
{"type":29,"instance":31,"name":"0x740001f"}]}
```


- **properties**
  Returns the list of properties available for object with given type and instance number in device having the given ID.
  The list is returned as a JSON string containing for each object
  - deviceID
  - object type
  - object instance
  - list of available properties

  Example:


```
{"deviceID":1101,"objectType":2,"objectInstance":1,
"properties":
[22,28,36,65,69,75,77,79,81,85,87,103,104,111,117,168,8309,8314,8332,8333]}
```


Example of usage:

```
var tagMgr = project.getWidget("_TagMgr");
var protID = "prot2"; // to be set according to protocol numbering
var params = String(fromId) + " " + String(toId) + " " + String
(timeout); // fromID and toID are min and max IDs

var json_str = tagMgr.invokeProtocolCommand(protID , "scan", params, state); //json_
str containts JSON string with scanned devices.
```

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported by this communication driver:

Error	Cause
Cannot bind to the device_id	Cannot establish communication with the Device ID provided for this tag.
Cannot read the property data type	The type of the property to write cannot be determined.
write conversion error	A conversion associated to this tag has failed.
Cannot write ICOM type …. BACnet type ….	A datatype selected for this tag is not compatible with the BACnet property to set.
Timeout on COV subscription	A request for COV subscription for this tag has timed out.
Timeout on waiting COV update	A COV notification has not been received for this tag whithin timeout.
Can't get COV for this property	The selected property for COV notification is unsupported.
datagramItem conversion error	A conversion associated to a tag that is part of a datagram has failed.
Timeout waiting on response	No response for a request of read or write property within timeout.
datagram element …., no data available	No data available for a tag that is part of datagram.
datagram element …., Unsupported BACnet data type	Read datagram element is of unsupported BACnet type.
datagram element …., can't convert BACnet type to ….	A Data Type selected for a tag which is part of a datagram is not compatible with the BACnet property to read.

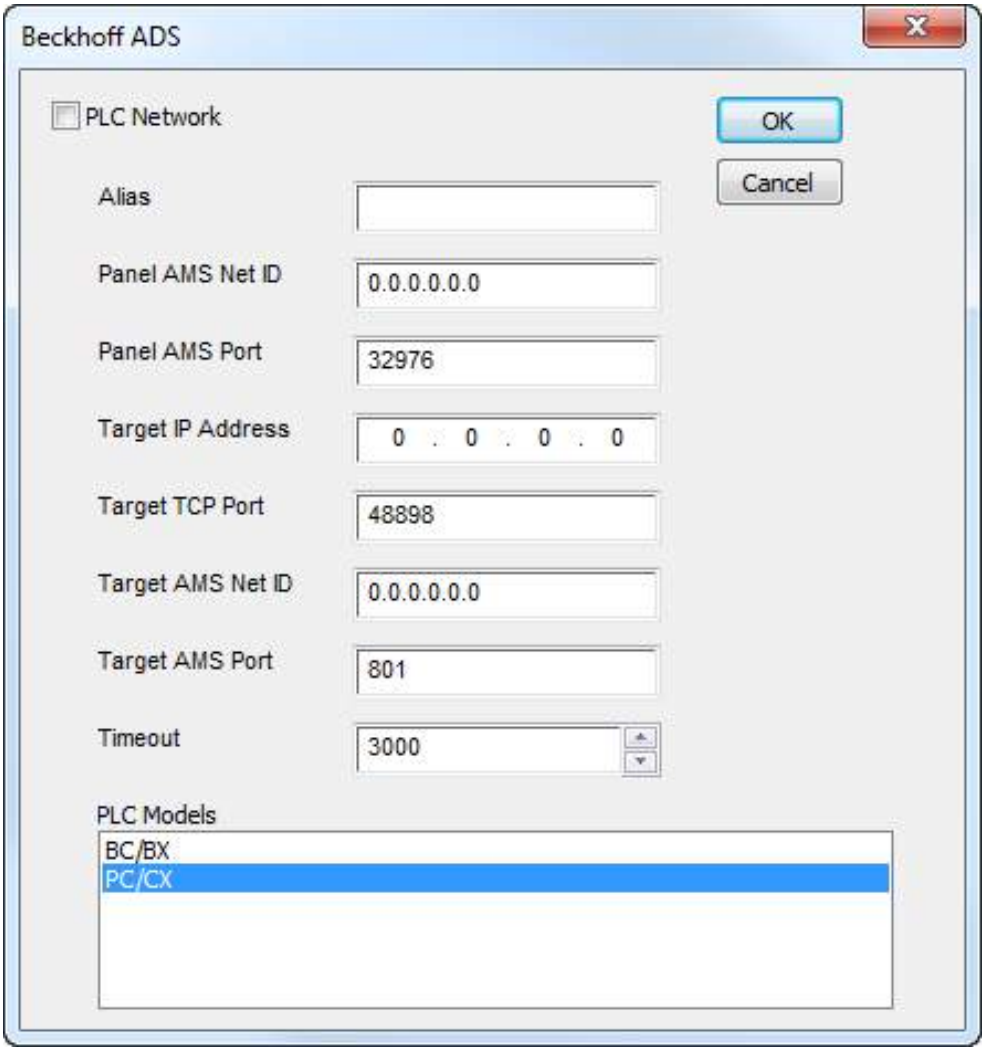
Error	Cause
No data in response	No data available for a tag.
Datagram element 'element_URI' error: 'error_class': error_code	The reading of indicated datagram element 'element_URI' was reported as error. The error descriptions **error_class** and **error_code** are included in the message.
datagram object does not match	The object of the received datagram item does not match the asked object.
datagram property does not match	The property of the received datagram item does not match the asked property.
BACnet abort: reason_of abort	BACnet abort message was received. The reason of abort is given.
BACnet reject: reason_of_rejection	BACnet reject message was received. The reason of rejection is given.
BACnet error: error_class: error_code	BACnet error message was received. The error description is given as combination of **error_class** and **error_code**.
parameter 'parameter_name' out of range	The protocol parameter **parameter_name** value is out of range.

Beckhoff ADS

Beckhoff ADS protocol driver is used for communication with Beckhoff controllers through Ethernet connection. This implementation of Beckhoff ADS protocol driver is based on the information published by Beckhoff.

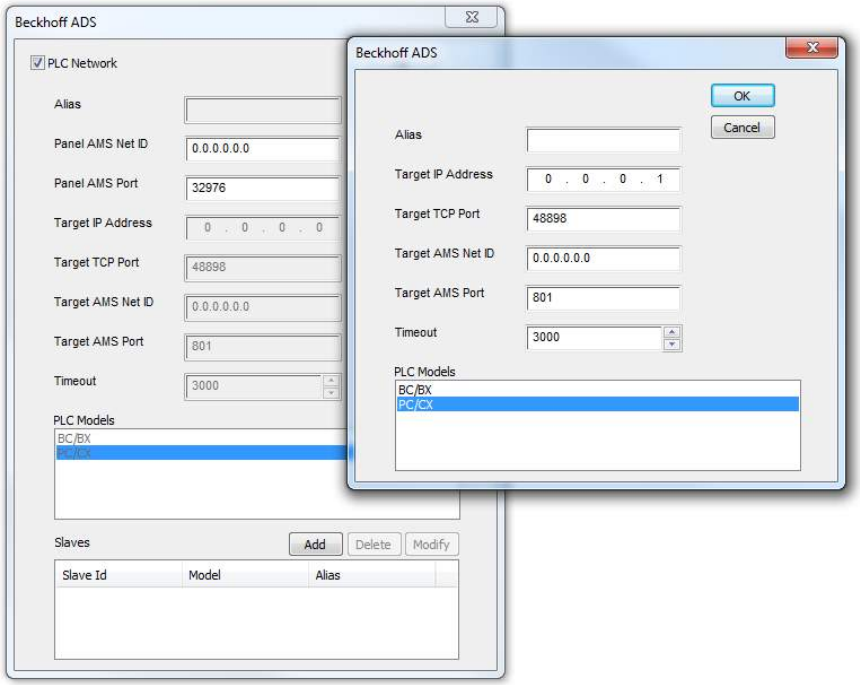
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol “Beckhoff ADS” from the list of available protocols.



Element	Description
Alias	Name to be used to identify nodes in the plc network configuration. The name will be added as a prefix to each tag name imported for each network node.
Panel AMS Net ID	Specifies the AMS net ID of the panel; the first 4 bytes must match the panel IP address assigned to the HMI device. If panel has IP address 192.168.10.100 then AMS Net ID could be 192.168.10.100.1.1
Panel	Specifies the panel AMS port number to be used on panel.

Element	Description
AMS Port	Using TwinCAT2, default Panel AMS Port is 32976. Using TwinCAT3, default Panel AMS Port is 32844.
Target IP Address	Specifies the IP address of the target controller.
Target AMS Net ID	Specifies the Target AMS net ID of the target controller.
Target AMS Port	Specifies the port number dedicated to the communication on target device. Using TwinCAT2, default Target AMS Port is 801. Using TwinCAT3, default Target AMS Port is 851.
Timeout	The number of milliseconds between retries when communication fails.
PLC models	Select the model which corresponds to the device to be connected. Model selection is very important to be set properly.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and enter the Target Controller settings for every node.

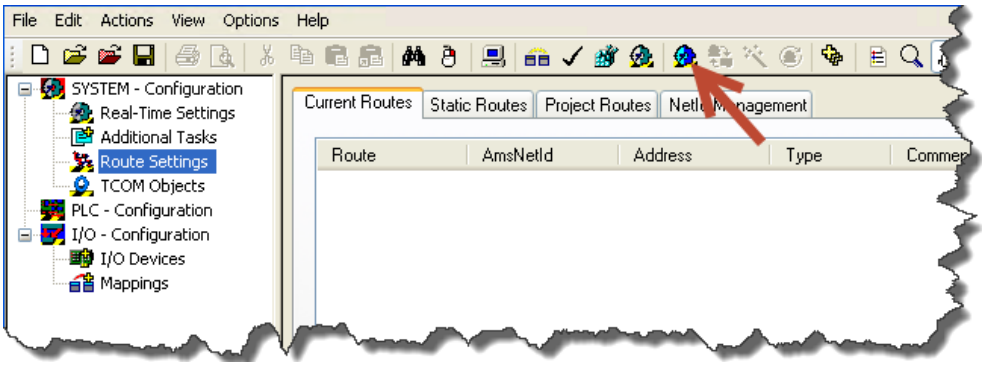


TwinCAT2 Route Settings

Beckhoff controllers require some specific settings to allow connection from HMI devices.

In TwinCAT2 System Manager you need to configure Static Route.

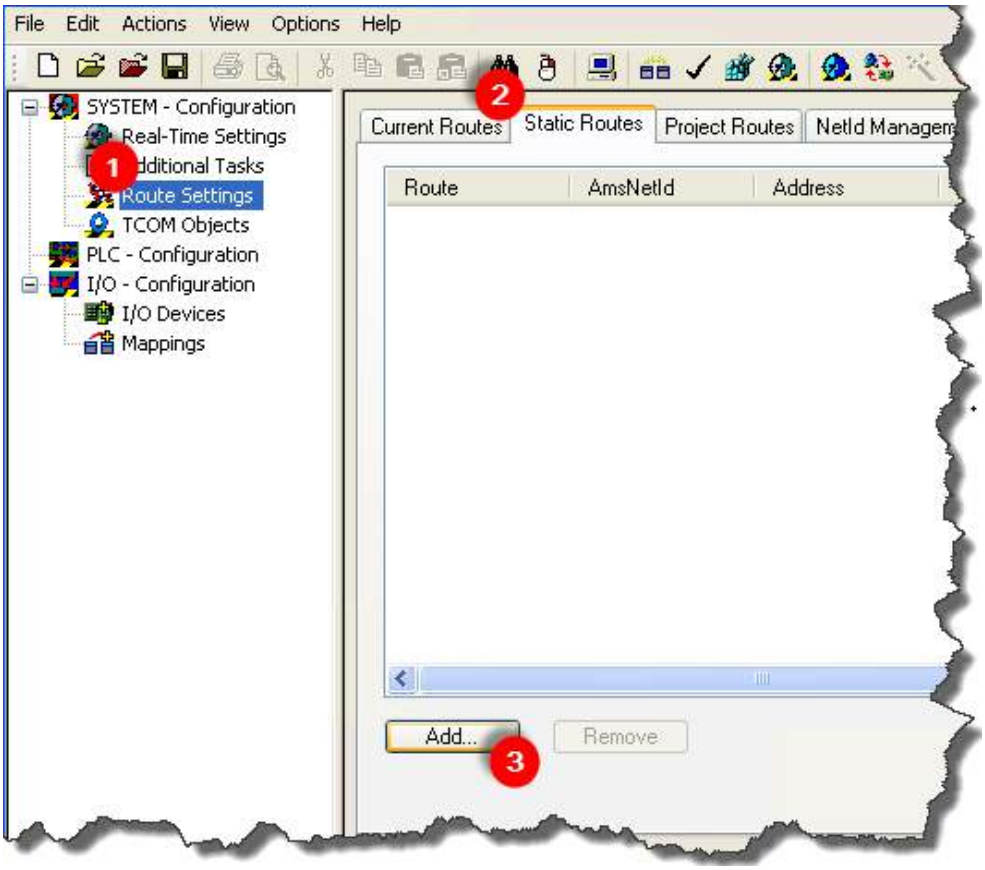
First of all the system must be reset in Configuration Mode using the toolbar button as showed in the following figure.



Then confirm to Restart TwinCAT2 System in Config Mode as in the figure below.

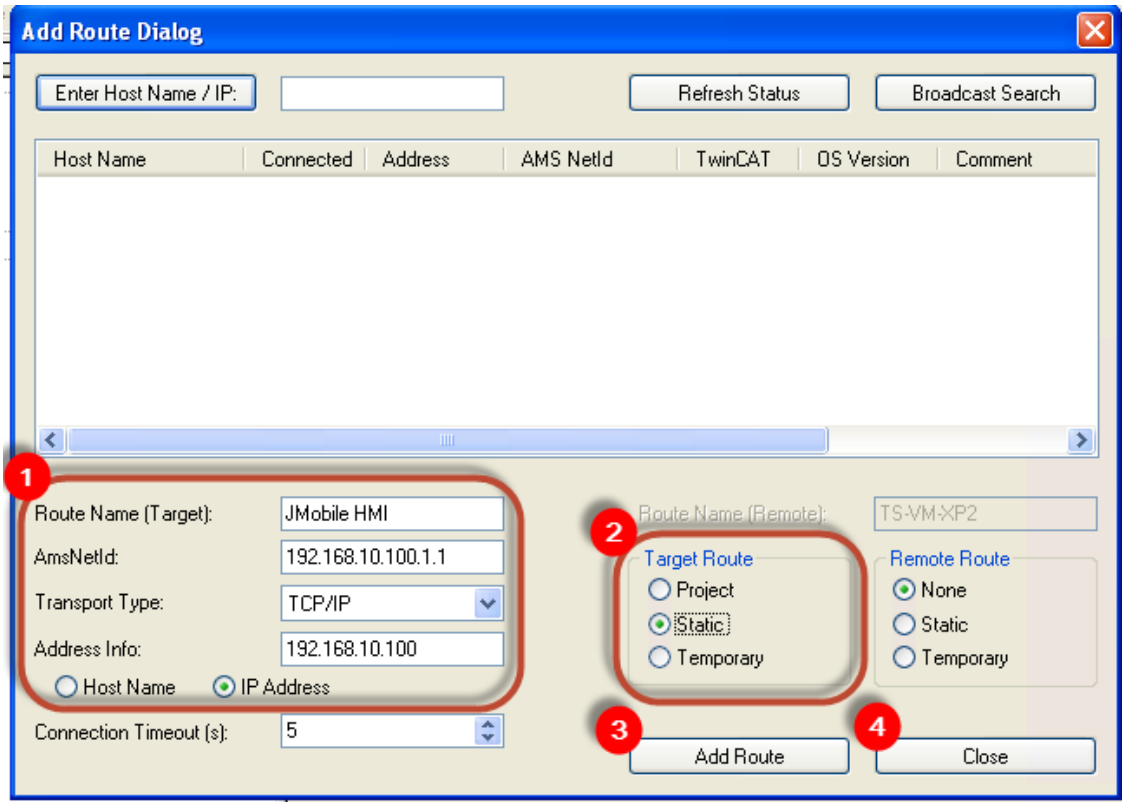
Once restarted, as in the next figure, follow these steps to add a new Route:

1. Open Route Settings.
2. Select Static Routes tab.
3. Click on [Add] button.



Into Add Route Dialog user must set:

1. Route Name: a name useful to indentify the Route i.e. “HMI”, AmsNetId: The Panel AMS Net ID as configured into Beckhoff ADS protocol, Transport Type: TCP/IP.
   Address Info: Type in the Panel IP Address with “IP Address” option selected.
2. Target Route: Static.
3. Click on [Add Route] button. Note: no warning or message will be shown.
4. Click on [Close] button.

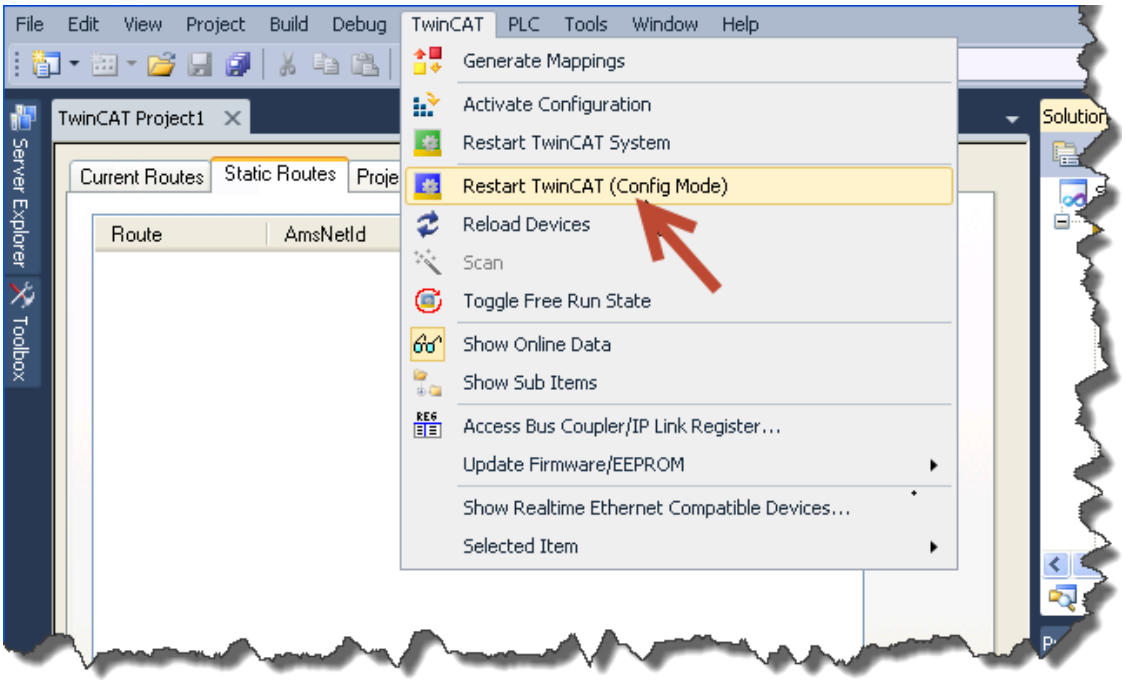


Then the route will appear under Static Routes list.

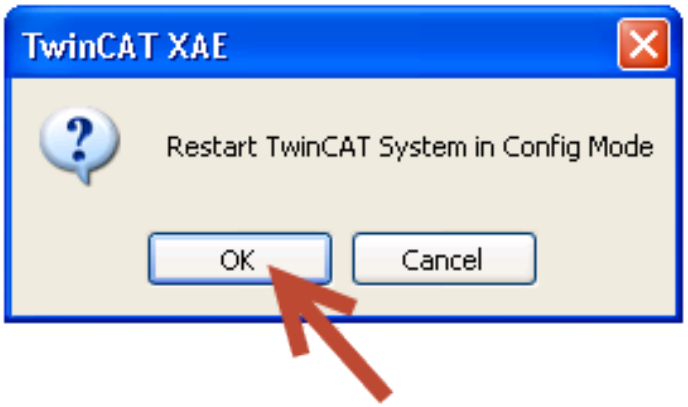
TwinCAT3 Route Settings

Beckhoff controllers require some specific settings to allow connection from HMI devices. In TwinCAT3 XAE you need to configure a Static Route.

First of all TwinCAT3 system must be reset in Configuration Mode using the toolbar button as showed in the following figure.

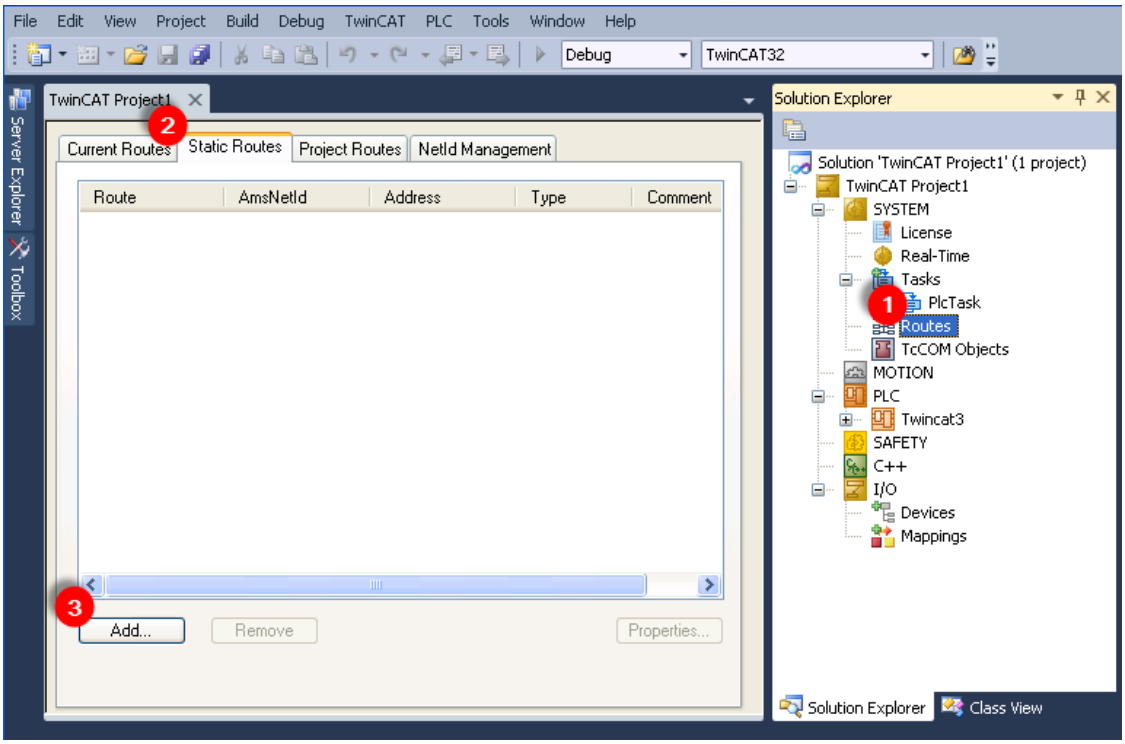


Then confirm to Restart TwinCAT3 System in Config Mode.



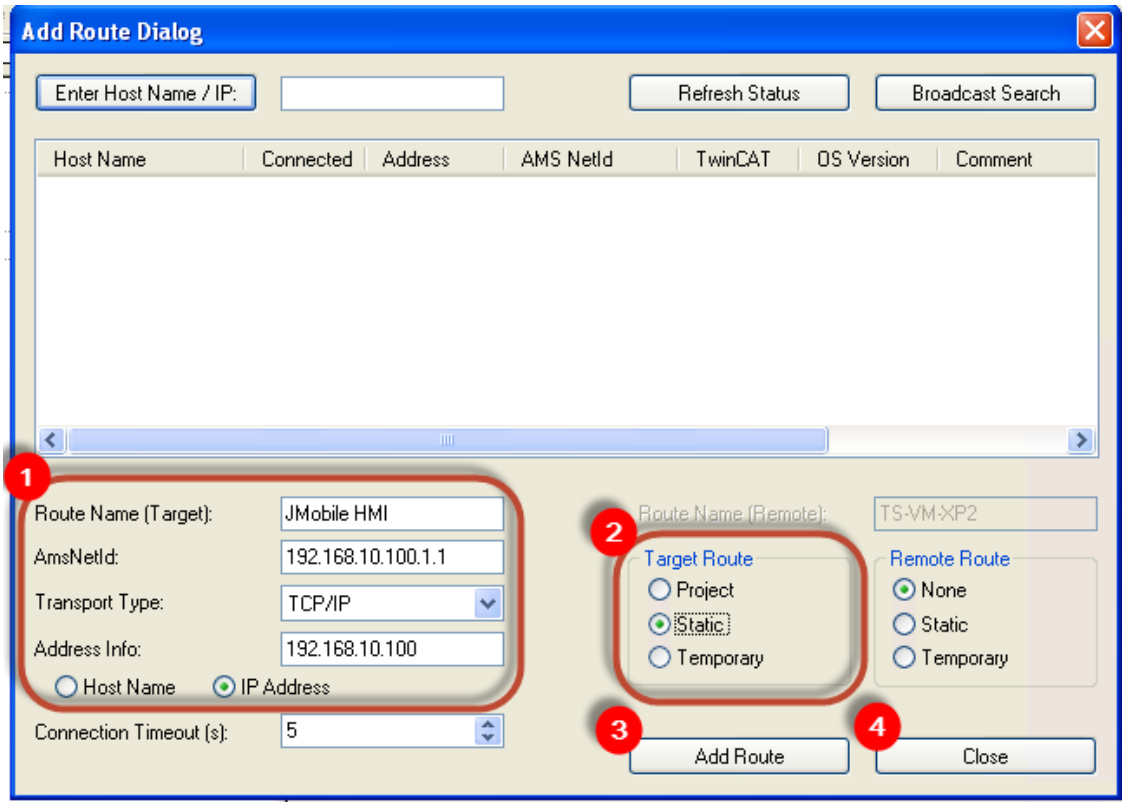
Once restarted, as in the next figure, follow these steps to add a new Route:

1. Open Routes.
2. Select Static Routes tab.
3. Click on [Add] button.



Into Add Route Dialog user must set:

1. Route Name: a name useful to indentify the Route i.e. “HMI”, AmsNetId: The Panel AMS Net ID as configured into Beckhoff ADS protocol, Transport Type: TCP/IP.
   Address Info: Type in the Panel IP Address with “IP Address” option selected.
2. Target Route: Static.
3. Click on [Add Route] button. Note: no warning or message will be shown.
4. Click on [Close] button.



Then the route will appear under Static Routes list.

Tag Import

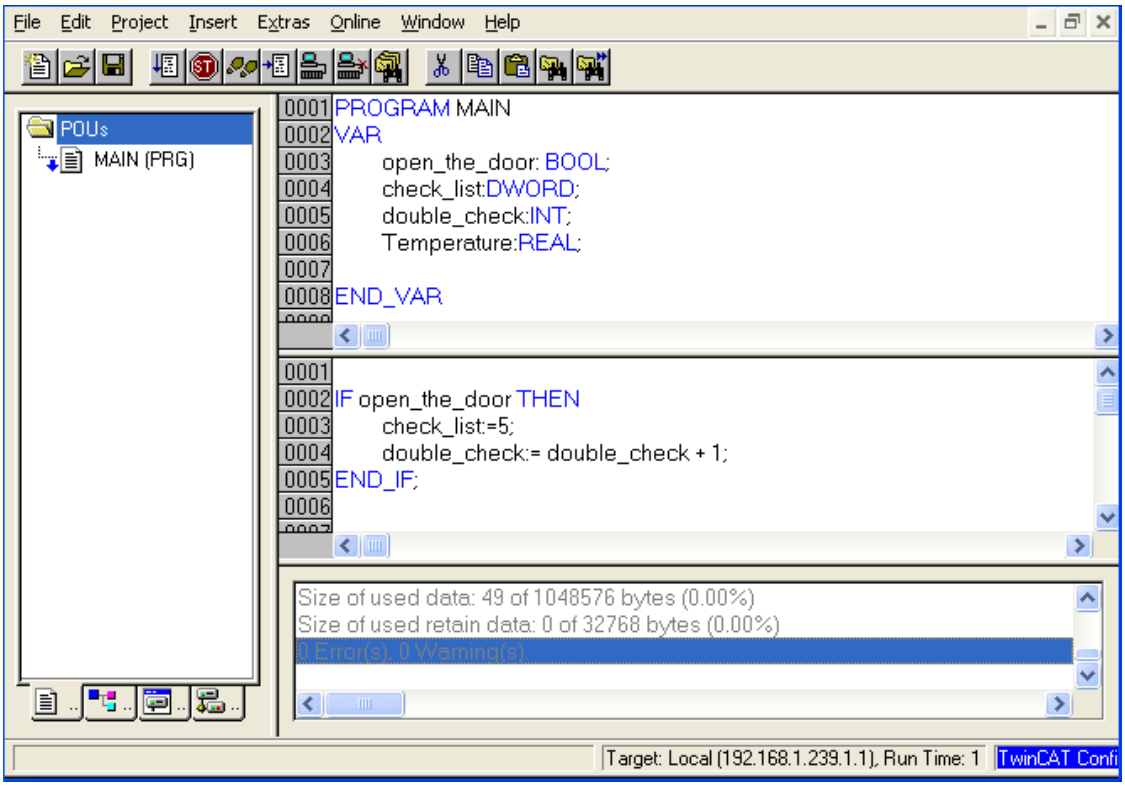
Exporting Tags from PLC

The data in the Beckhoff system is based on tags.

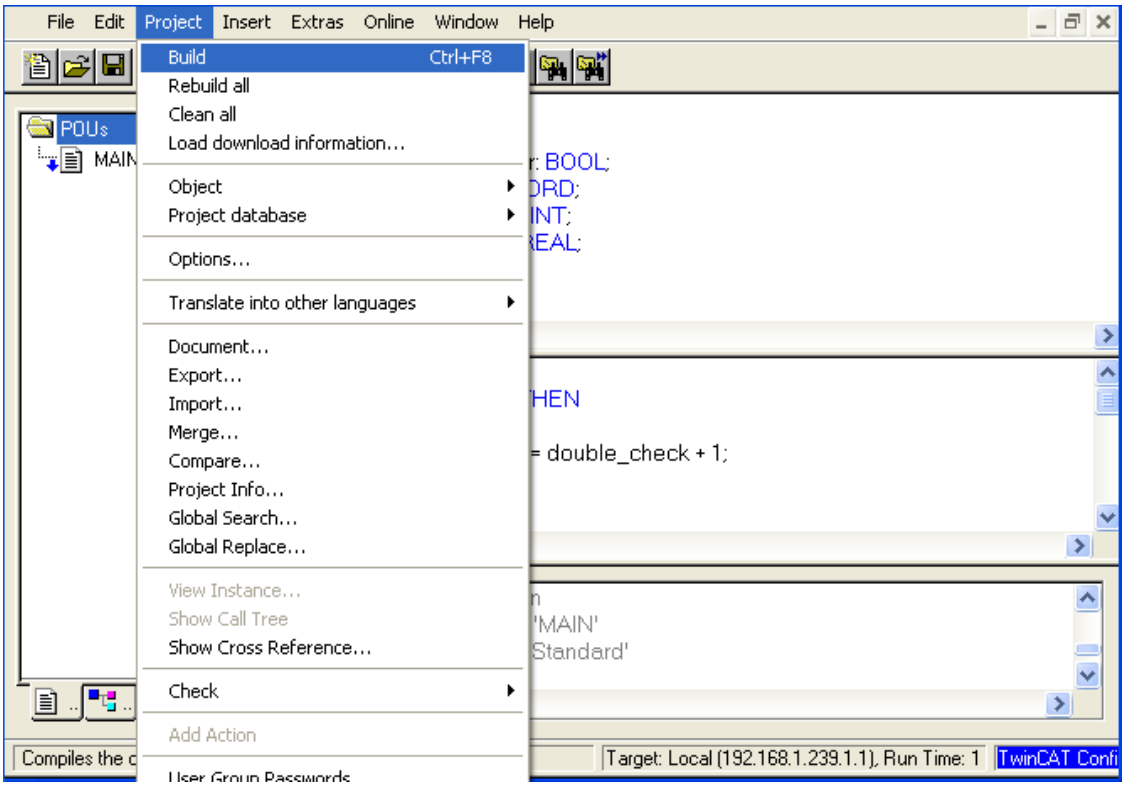
The organization of the internal memory of the controller is not fixed but it is configured by the user at development time. Each data item can be identified by a string called “tag”.

The TwinCAT development environment generates the list of tags created for each controller in the configuration of the application.

The project in the panel must refer to the tag names assigned in the TwinCAT PLC Control programming software at development time. The Designer Tag Editor supports direct import of the tag file generated by the Beckhoff software.



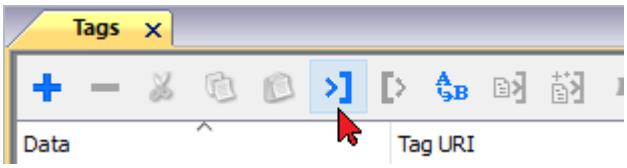
To export tags defined for the selected controller, click on Project > Build as shown.



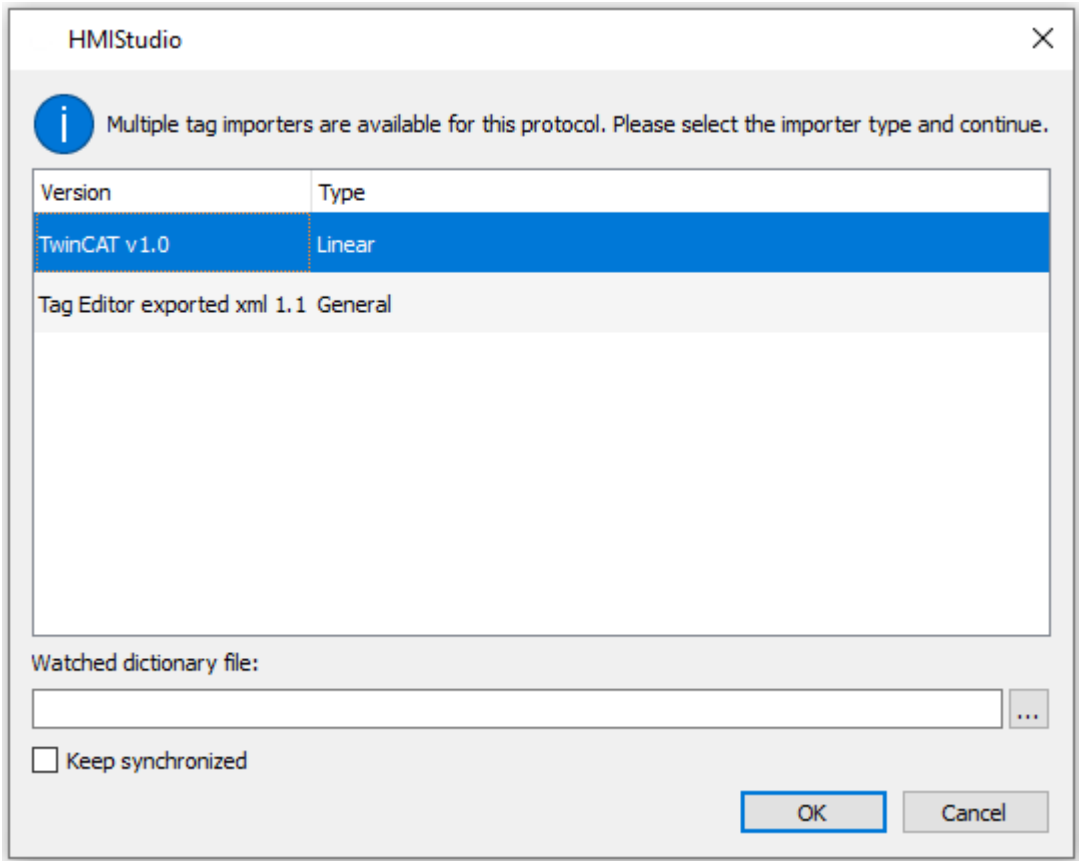
The TwinCAT PLC Control software will create a file with extension TPY.

Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



The following dialog shows which importer type can be selected.

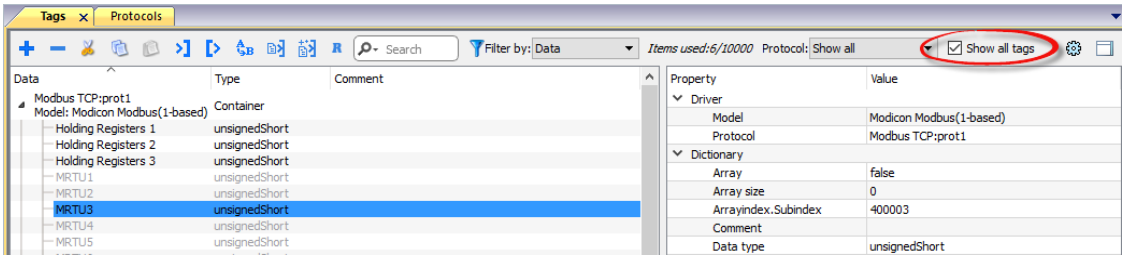


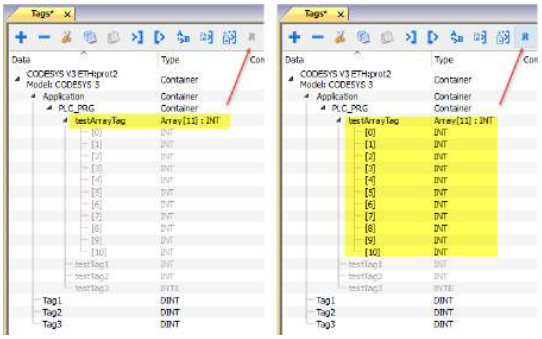
Importer	Description
TwinCAT v1.0 Linear	Requires a **.tpy** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Note: the Beckhoff driver supports direct access to the PLC tags using the handles; this means that if no tags are added to the PLC and the PLC program is just re-compiled, you do not need to re-import tags as the access to them does not depend from the offset, but only from name.

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

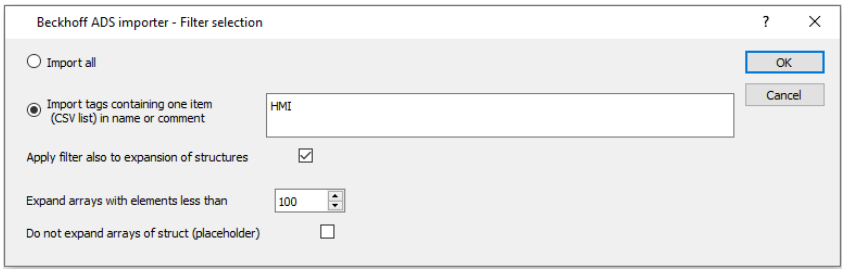
Using TwinCAT v1.0 Import Filter

When importing tags, the user can decide to import all the tags from the **.tpy** file or apply a filter importing only a subset of them.

The figure below shows how to specify the filter. The filter consist in a string (no wildcards are supported). The import filter will import only the tags having the specified string in the description.

If the description is applied to an “instance declaration” of a Function Block, all the tags within the block will be imported.

If the string is contained only as comment of some variables inside the Function Block, only that variables will be imported.



As an example for the use of the import filter, please see the following case.

```
FUNCTION_BLOCK FB_Motor
VAR_INPUT
     bStartMotor: BOOL;
     bReset: BOOL;
END_VAR
VAR_OUTPUT
     bMotorOn: BOOL;
     bAlarm: BOOL;(* HMI Thermal alarm *)
END_VAR
VAR
     sData: STRING;
     bResetStatistics: BOOL; (* HMI Reset statistics *)
END_VAR
VAR PERSISTENT
     stStat: ST_MotorStats; (* HMI Motor statistics *)
END_VAR

Function block instances declaration:
VAR
     fbMotor1: FB_Motor;
     fbMotor2: FB_Motor; (* HMI only show Motor 2!! *)
END_VAR
```

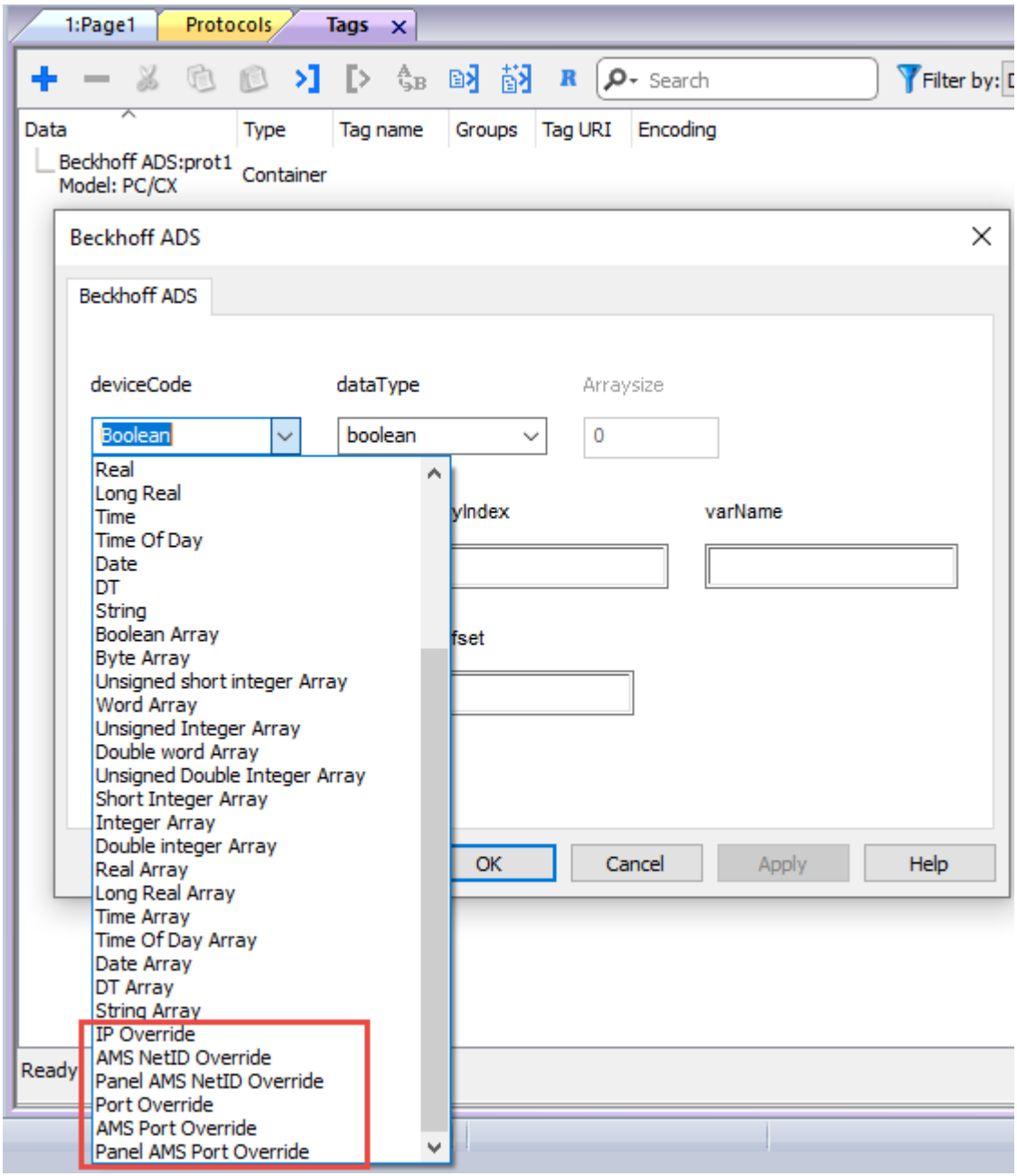
The following tags will be imported:

- MAIN/fbMotor2/bAlarm
- MAIN/fbMotor2/bResetStatistics
- MAIN/fbMotor2/ST_MotorStats

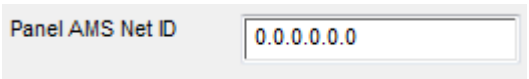
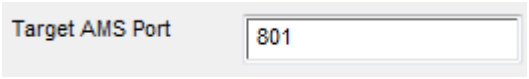
Override Data Types

The protocol provides special data types which allow you to change the protocol configuration at runtime.

If added in the project, these variables are initialized with the value specified in the project at programming time.



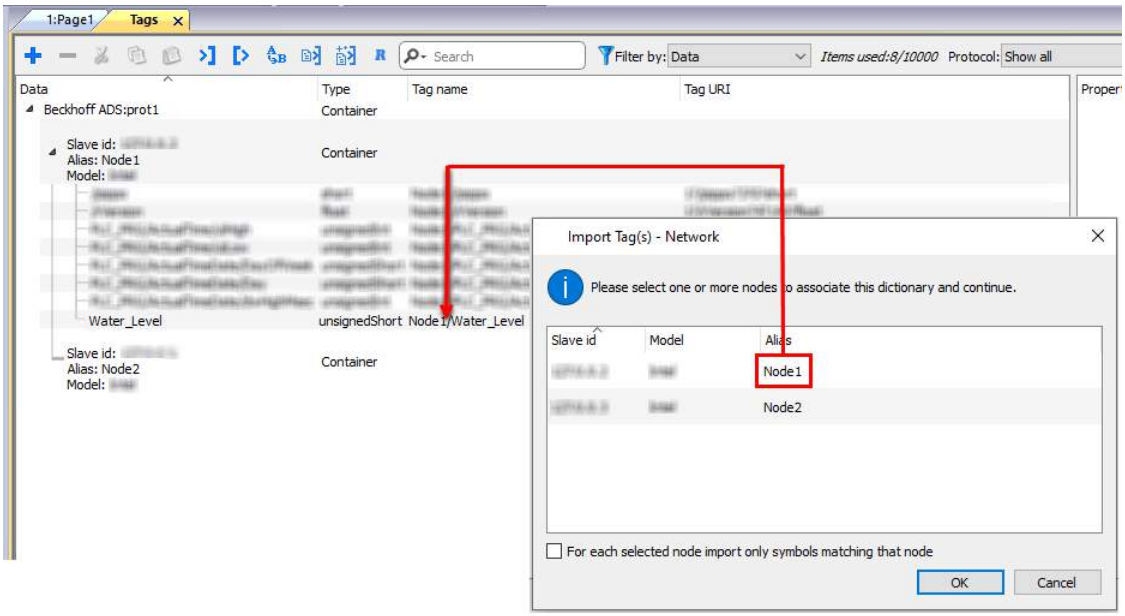
The table below shows which data type to use for any protocol parameter to override at runtime.

Override Data Type	Protocol Parameter	Description
IP Override		Overrides the PLC IP address. It is an **unsignedByte** array of 4 elements, one per each byte of IP address.
AMS NetID Override		Overrides the PLC AMS NetID. It is an **unsignedByte** array of 6 elements, one per each byte of AMS NetID.
Panel AMS NetID Override		Overrides the PLC AMS NetID. It is an **unsignedByte** array of 6 elements, one per each byte of AMS NetID.
Port Override		Overrides the PLC TCP port. It is an **unsignedShort**.
AMS Port Override		Overrides the PLC AMS port. It is an **unsignedShort**.
Panel AMS Port Override		Overrides the Panel AMS port. It is an **unsignedShort**.

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Note: Aliasing tag names is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
General Error	Error cannot be identified; should never be reported; contact technical support

CAN Direct v2.0x

CAN Direct communication driver allows to communicate with CAN devices over CAN ports of HMI.

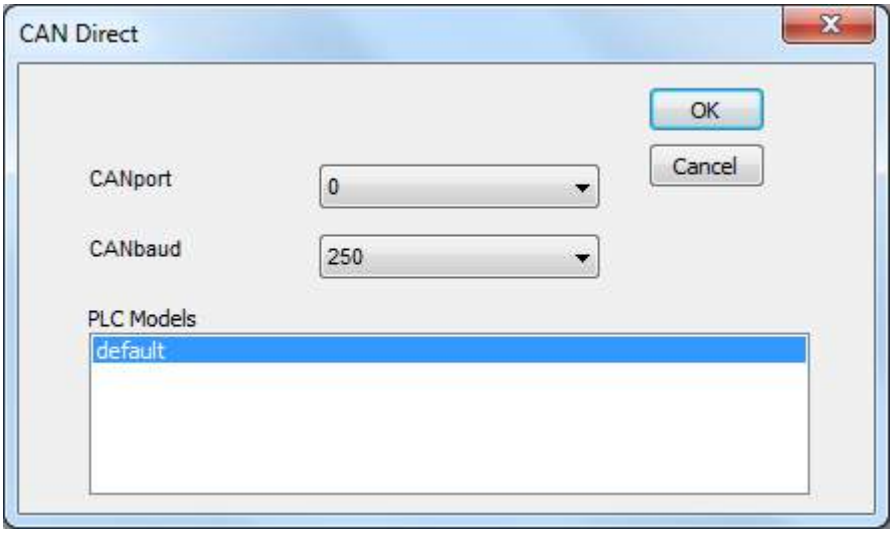
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.

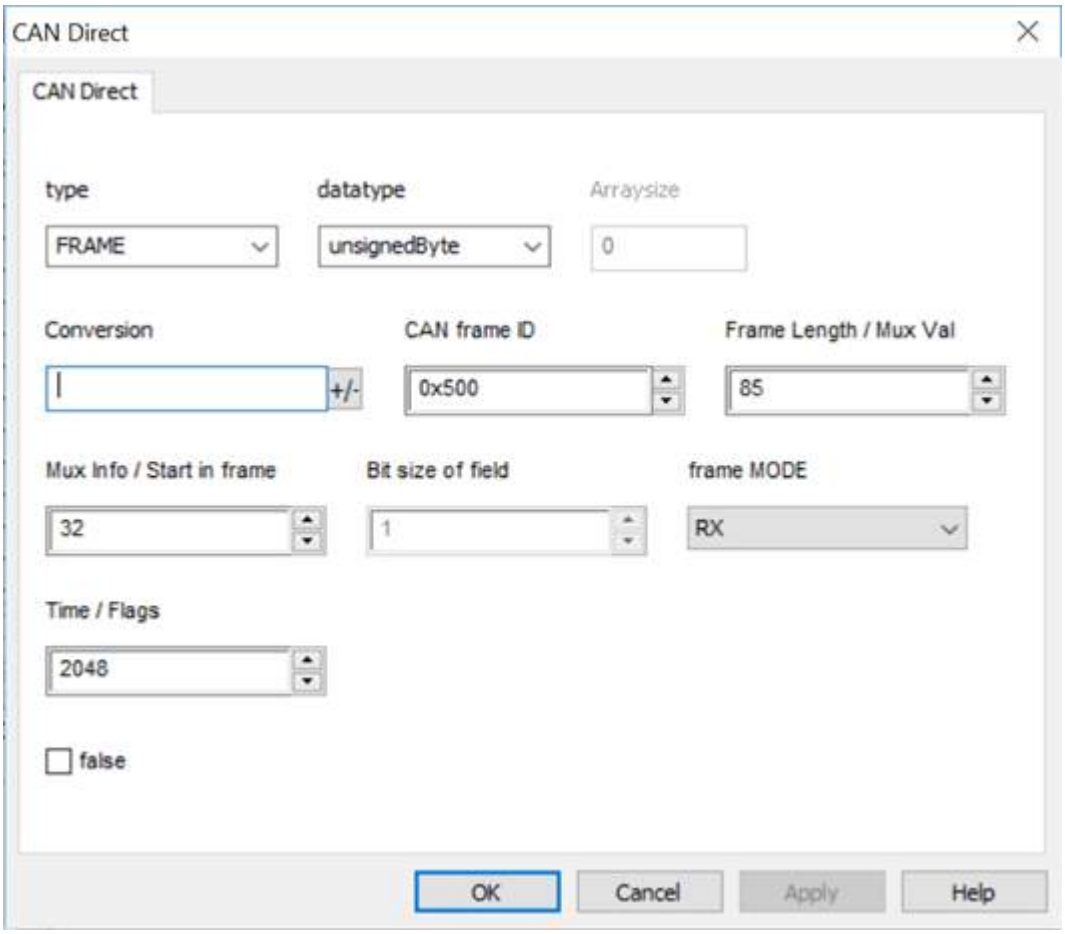


Element	Description
CANport	Indicates the CAN port used. Allowed values are 0, 1, 2, 3 according to hardware platform.
CANbaud	Indicates the baudrate. Allowed values are 100, 125, 250, 500, 800, 1000.
PLC Models	Fixed to default.

Tag Editor Settings

*Path: **ProjectView*> *Config* > *double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **CAN Direct** from the protocol list: tag definition dialog is displayed.



There are two Tags type:

- FRAME
- VAR

FRAME definition

Element	Description
Type	This table refers to **FRAME** type
Datatype	For **FRAME** type, there are two data type available:

Data Type	Memory Space	Limits
unsignedByte **frame Mode**	8-bit data	0 ... 255
unsignedInt[3] **idx 0 frame Mode** **idx 1 frame Length** **idx 2 frame Time**	Array of 32 bit integers	

Element	Description
Conversion	No conversion allowed for **FRAME** type
CAN frame ID	11 or 29 bit CAN frame identifier
Extended Frame	**checked**: Frame is 29 bit type **not checked**: Frame is 11 bit type
Frame Length	1..1024: Length of data, expressed in bytes. Default is 8
Mux Info	Multiplexer information. If the frame is multiplexed it contains the following info: Bit 0 to 15 position of multiplexer in frame (MSbit if multiplexer is big-endian) Bit 16 to 21 size if multiplexer in bit (1 to 63) Bit 22 set if multiplexer is big-endian If frame is not multiplexed Mux Info contains 0
Bit Size of field	(Grayed out) not available for frames
Frame MODE	Indicates direction and transmission mode: **RX**: received frame **TXall**: frame is sent when all variables in frames are written **TXany**: frame is sent when any variable in frame is written **TXperiod**: frame is sent every TIME milliseconds
Validity Time or Period	**For RX frames**: indicates the validity time of variables (0=forever), expressed in milliseconds **For TX frames**: indicates the sending period, expressed in milliseconds

Writing to elementsof the frame tag it is possible to control the behavior

idx 0 frame Mode it is possible to redefine the mode from RX to TX---- etc.

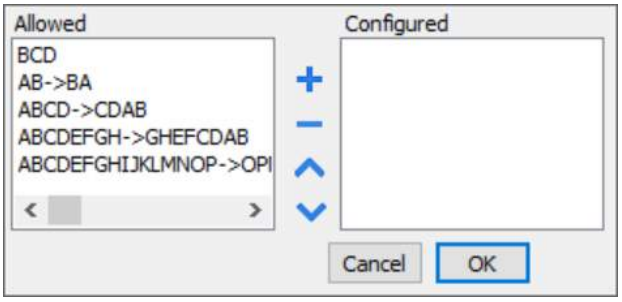
idx 1 frame Length it is possible to adapt frame length on the fly before sending the frame

idx 2 frame Time it is possible to redefine the sending time period for TXperiod mode or the timeout for Rx mode

in case of Mode = TXperiod and Time = 0 every write acceszs to the frame TAG will trigger a transmission of the frame

VAR definition

Element	Description
Type	This table refers to **VAR** type
Datatype	For **VAR** type, available data types are:

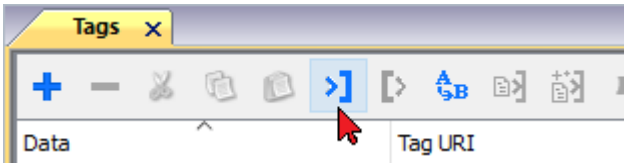
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
</td></tr>
<tr><td>Conversion</td><td> 

BCD Swap Swap2 Swap4 Swap8 and Negate conversions are possible according to DataType</td></tr>
<tr><td>CAN frame ID</td><td>11 or 29 bit CAN frame identifier. It must match the predefined FRAME Tag.</td></tr>
<tr><td>Mux Val</td><td>If the frame is multiplexed each frame can contain different information according to the value of multiplexed field. This filed define for which value of multiplexer this VAR must be updated. In this case bit 0 of flags must be set. Otherwise tha VAR will be updated every time the frame is received</td></tr>
<tr><td>Start bit in</td><td>Indicates the starting position int bits (0-63) of the data</td></tr>
</table>

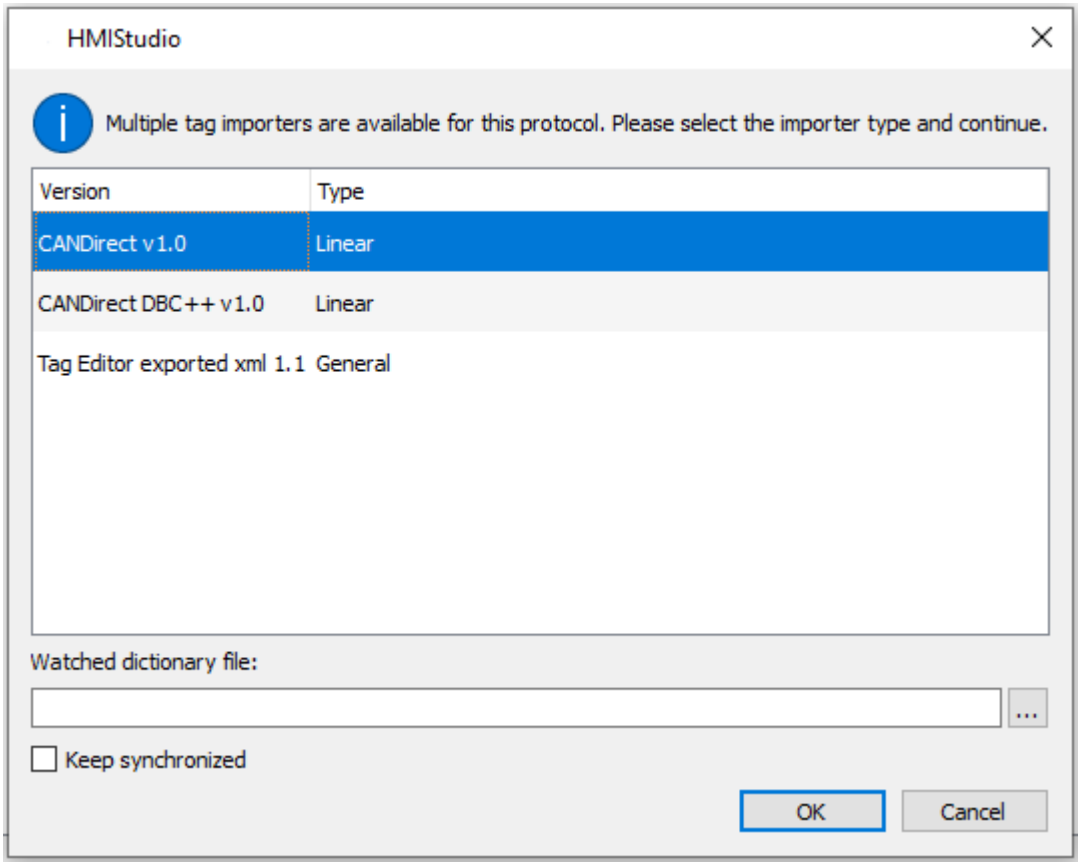
Element	Description
frame	
Var size in bits	Indicates the number of bits of data
Frame Mode	(grayed out) not available for VAR
Flags	Bit 0 indicates that the VAR is multiplexed Bit1 indicates that VAR value is big-endian

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



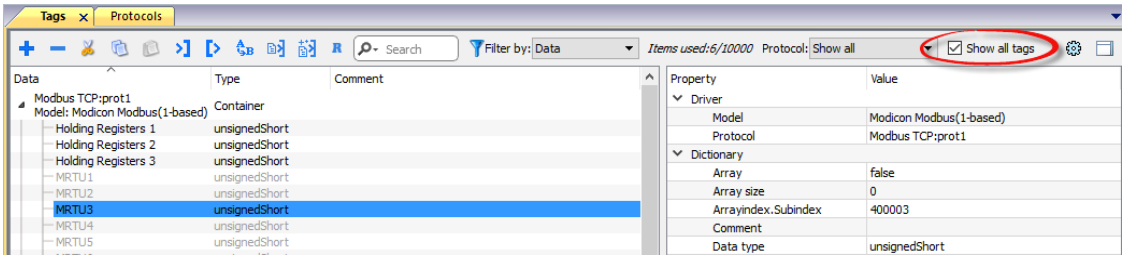
The following dialog shows which importer type can be selected



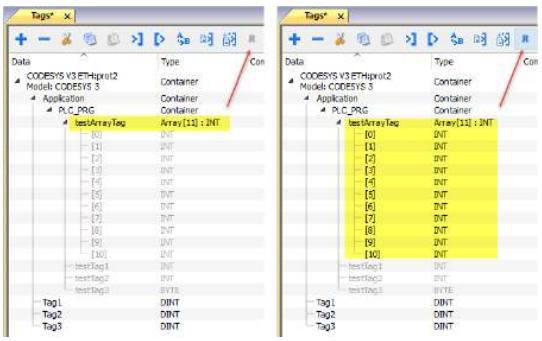
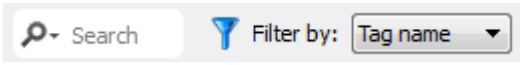
Type	Description
CANDirect v1.0 Linear	Requires a **.sym** file generated by WE ICCS SDK PLUS specifying inversion of direction and type 1, or by other Symbol editors like PCAN Symbol Editor All variables will be displayed at the same level.
Vector DBC+ v1.0 Linear	Requires a .dbc file generated by Vector CANdb++ Editor All the frames will be generated with type = Rx, so frames created for transmission must be reedit after importation
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	Import Tag(s). Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	Update Tag(s). Click on this icon to update the tags in the project, due a new dictionary import.
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

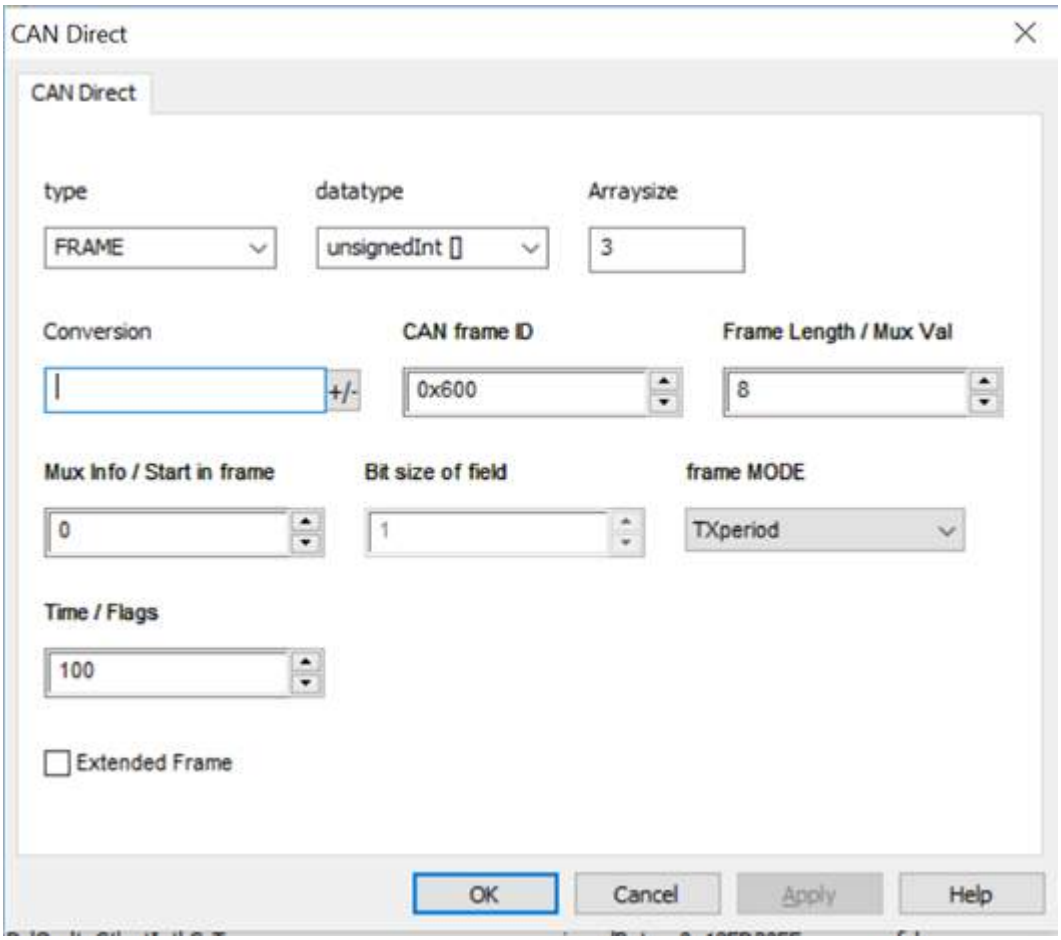
Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

Example of usage

Define TX Frame

In this example, Tag1 is declared as FRAME tag.

It sends a message to the ID 600 (258HEX) every 100 miliseconds.



Define RX Frame

In data definition on CODESYS V2:

1. In sendpack object, define the frame ID (yellow)
2. In sendpack object, define the frame length
3. In "Data" array write the value that must be written

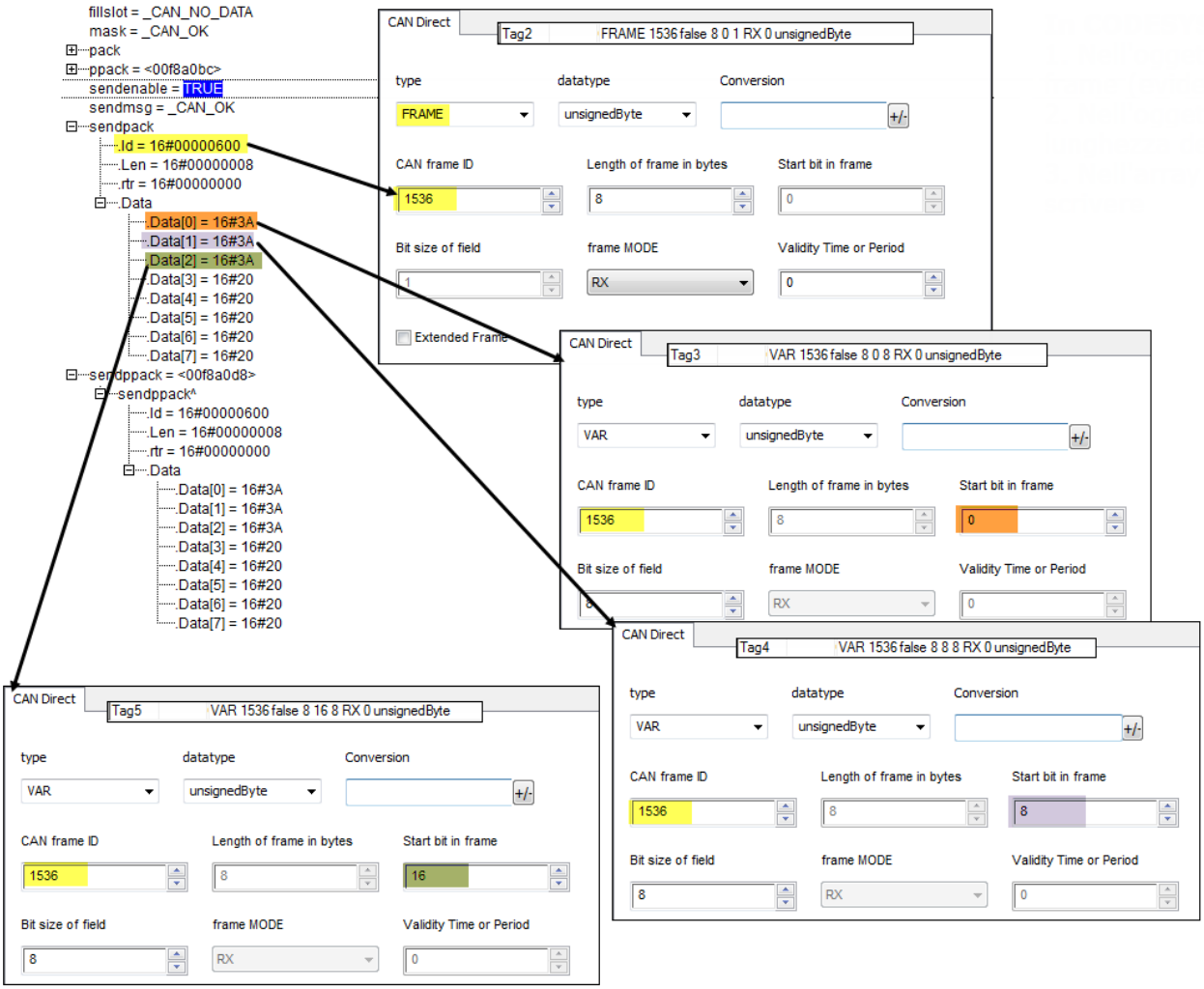
Declare Tag2 as FRAME type, with frame MODE set to RX. CAN frame ID must be the same as ID set in data definition on CODESYS side (yellow).

Note: On protocol side, CAN frame ID is expressed in decimal format.

Declare Tag3, Tag4 and Tag5 as VAR type. CAN frame ID must be the same as ID set in data definition on CODESYS side (yellow).

For each VAR tag, set the correct Start bit in frame property.

Refer to following image, which summerize the above example.



JavaScript Interface

Beside Tag interface the user can access the protocol via JavaScript.

Although defined Tags can be accesses by JavaScript too, JavaScript can access directly to a Command interface implemented in protocol. This interface does not require the definition of Tags and is direct to protocol resulting in more efficiency.

This interface provides the access to token queue and sending function. The following commands are supported:

Command	Description
put	Put the token to send contained in string parameter.
get	Get the received token.
get_token_length	Get the length of received token.
tokens_available	Get number of tokens received.
token_ack	Acknowledge reading token.

Using the command interface the following JS code should receive data:

```
var tagMgr = project.getWidget("_TagMgr");
var protID = "prot2"; // to be set according to protocol numbering

var avail = tagMgr.invokeProtocolCommand(protID, "tokens_available", "");
while (parseInt(avail) > 0)
{
      var str = tagMgr.invokeProtocolCommand(protID, "get", ""); // get the next
token
      var status = tagMgr.invokeProtocolCommand(protID, "token_ack",""); //
acknowledge current token
      avail = tagMgr.invokeProtocolCommand(protID, "tokens_available",""); // get
number of available tokens in queue
}
```

CANopen HMI

The CANopen HMI communication driver has been designed to connect HMI products to a CANopen network. A new device communication profile has been developed for the HMI. This profile takes advantage from the advanced user interface features of the products, while retaining the simple networking concept supported by the CANopen network.

The basic idea is create a client/server communication structure where the HMI is the client and the CANopen controller is the server.

Connection to CANopen network requires the optional CANopen communication module. Verify the suitable version for your HMI model.

Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Please ensure that the latest driver is used in the application.

CANopen HMI Profile

In this communication model the HMI initiates the communication sessions, acting as a source of messages.

The basic messages are PDO messages with the standard size of 8 bytes.

The COB-ID of the messages is defined in a way that makes clear, from the well-known CANopen rules, what is the target of the PDO message.

The format of the PDO message has been defined according to a custom application layer protocol. This application layer protocol defines a device-independent communication profile optimized for HMI applications.

When the CANopen master controller receives the PDO message, it will interpret its contents and produce a PDO message with the response addressed to the HMI device.

The definition of this client/server relationship is independent of the CANopen Master in the sense that it can easily be supported in any particular CANopen master system. The resulting solution is easily portable to any CANopen master.

The software IDE offers a user interface that adapts itself to show the typical addressing model of CANopen master controller where the panel is going to be connected.

Adapting to different masters is possible using a profile customization file that may contain data definitions for different controller types.

Profile Details

This chapter provides the specification of the HMI profile and describes the subset of the request/response formats used by this implementation of the protocol.

The communication driver in the HMI generates PDO messages initiating communication request sessions as soon as the HMI runtime requires data from the protocol.

The panel is using the first transmit PDO identified by the COB-ID 0x180 combined with the Node Number assigned to the panel.

The communication profile uses only one transmit PDO and one receive PDO; the limited number of bytes available in standard PDO message maybe limiting, in some cases, the driver capabilities especially in terms of performance.

Request Format: HMI to Controller (Transmit PDO)

The PDO message transmitted by the HMI is formatted according to the following table.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Offset Low	Offset High	Data 0	Data 1	Data 2	Data 3	Data Length and Job Number	Operation Type and Controller ID

The request frame includes the following elements:

Offset Low	Low byte of the offset (16 bits address) for the requested block of data
Offset High	High byte of the offset (16 bits address) for the requested block of data
Data 0 … Data 3	Data for Write Operations; not used in Read Operations
Data Length and Job Number	Contains: • number of requested bytes • job Number indicator;
Operation Type and Controller ID	Contains: • type of operation requested • the Controller ID that identifies the target of the message;

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data Length [1]	Data Length [0]	Job Number [5]	Job Number [4]	Job Number [3]	Job Number [2]	Job Number [1]	Job Number [0]

The “Data Length” parameter is coded in 2 bits and takes values between 1 and 4 according to the following rules:

00	1 bytes
01	2 bytes
10	3 bytes
11	4 bytes

Note that the elementary size of each data item depends on the Controller memory organization.

The “Job Number” occupies 6 bits and can have values between 0 and 63; the “Job number” parameter is placed as last element in the PDO to ensure data consistency; the PLC program running the controller should constantly monitor the value of the “Job Number” parameter and consider the received message as valid only when detecting a change in the value of the “Job Number” field. “Job Number” is automatically increased at each new communication session (new request frame).

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Operation Type	Controller ID [6]	Controller ID [5]	Controller ID [4]	Controller ID [3]	Controller ID [2]	Controller ID [1]	Controller ID [0]

The “Operation Type” uses one bit with the following definition:

0	Read	data is transferred from controller
1	Write	data is transferred to controller

The “Controller ID” uses 6 bits; it represents the Node Number in the CANopen network of the master controller addressed by the current request.

This parameter is required in case the CAN network has more than one master controller; the CANopen standard defines in fact the COB-ID of the messages in a way that all the partners of the bus known the originator. In case more than one master device is present in the same network, the “Controller ID” field will specify the target of each individual request message. Only the master controller that recognizes in this field its own Node ID will consider the message and process the PDO contents.

Response Format: Controller to Panel (Receive PDO)

The PDO message returned by the controller must be formatted as defined in the following table.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Status Flag / Error Code	Dummy – Always 0	Data 0	Data 1	Data 2	Data 3	Data Length and Job Number	Operation Type and Controller ID

The request frame consists of the following elements:

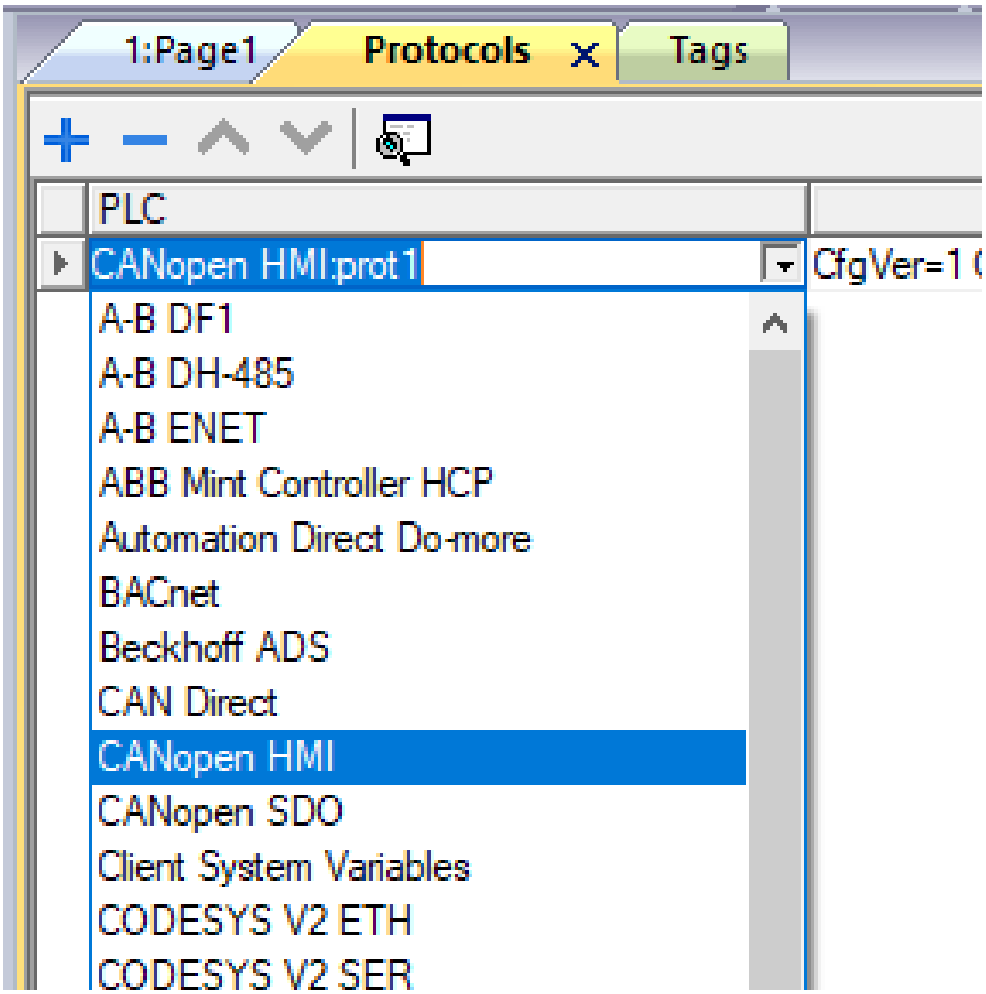
Status Flag / Error Code	Contains the information related to the execution of the operation type of the request; the next table shows the coding information
Data 0 … Data 3	Contain the data information returned to the panel in response to a Read request
Data Length and Job Number	It is the copy of the corresponding field of the request frame
Operation Type and Controller ID	It is the copy of the corresponding field of the request frame

	Status Flag / Error Code	
Operation Type in the Request Frame	**No Errors**	**Error**
Read	0x01	0x81
Write	0x02	0x82

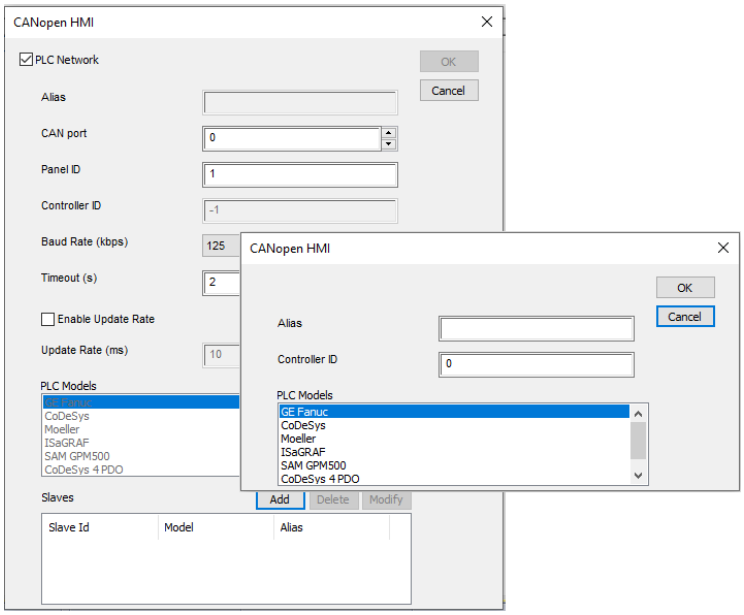
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “CANopen HMI” from the list of available protocols.

The driver configuration dialog is shown in figure.



Element	Description
Panel ID	CANopen node ID assigned to the HMI
Controller ID	CANopen Node ID assigned to the CAN controller device
Baud Rate (kbps)	Speed of the CANopen network
Timeout (s)	Maximum allowed time the driver will wait for a response from the PLC before reporting a communication error
Enable Update Rate	Use this option to enable a wait time between two communication requests
Update Rate (ms)	Minimum interval time between two requests; it can be useful when the bus load needs to be properly controller and limited

Element	Description
PLC Models	The list allows selecting the controller model you are going to connect to. The selection will influence the data range offset per each data type according to the specific controller memory resources
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and enter the node ID per each slave you need to access. 

Connecting the HMI to CODESYS V2 Controllers

This chapter describes all the steps you have to follow in order to establish a successful connection between the HMI and CODESYS CANopen master controller.

The PLC support program has been developed with CODESYS programming software version 2.

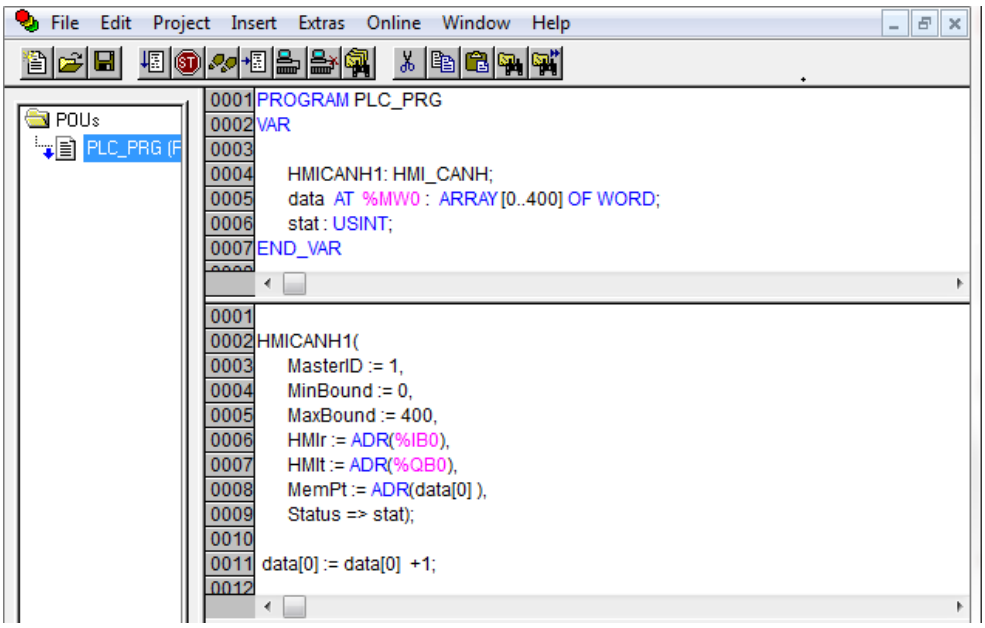
PLC Library Call

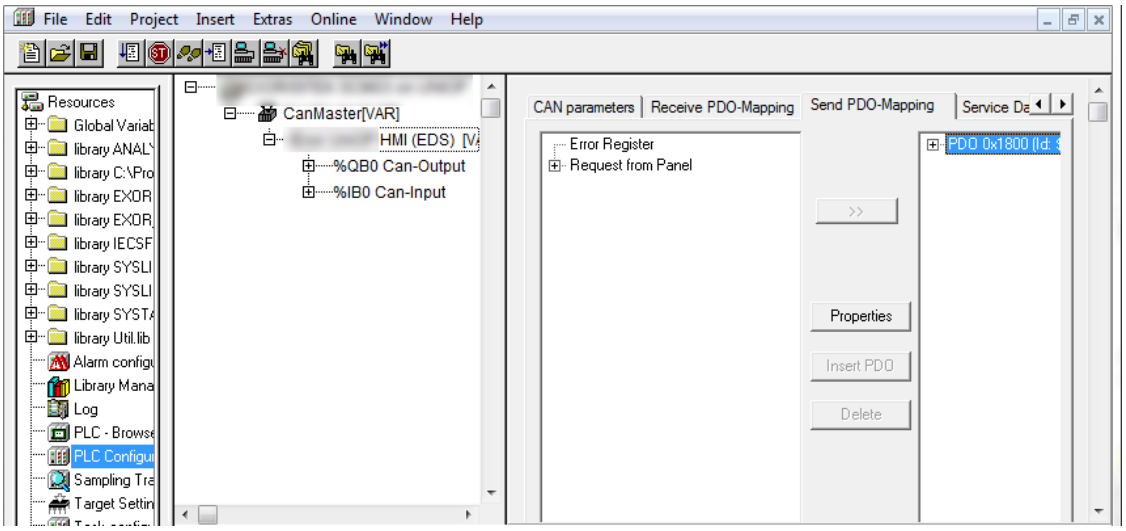
The server function running in the PLC program has been designed in the form of Library called “HMI_Canh”, written using the “ST” programming language. Proper working example is available on demand.

The Function Block parameters are the following:

MasterID	CANopen Master Node number;
MinBound	Lower limit of the PLC memory addressable (visible) by the HMI
MaxBound	Upper limit of the PLC memory addressable (visible) by the HMI
HHIr	Offset in the PLC memory where the PDO message received from the panel is mapped
HMIt	Offset in the PLC memory where the PDO message to be sent to the panel is mapped
MemPt	Offset in the PLC memory where the data is received
Status	Status

The PLC Function block support the use of more than one panel simply repeating the call of the same function for all the additional units specifying before each call the proper calling parameters.

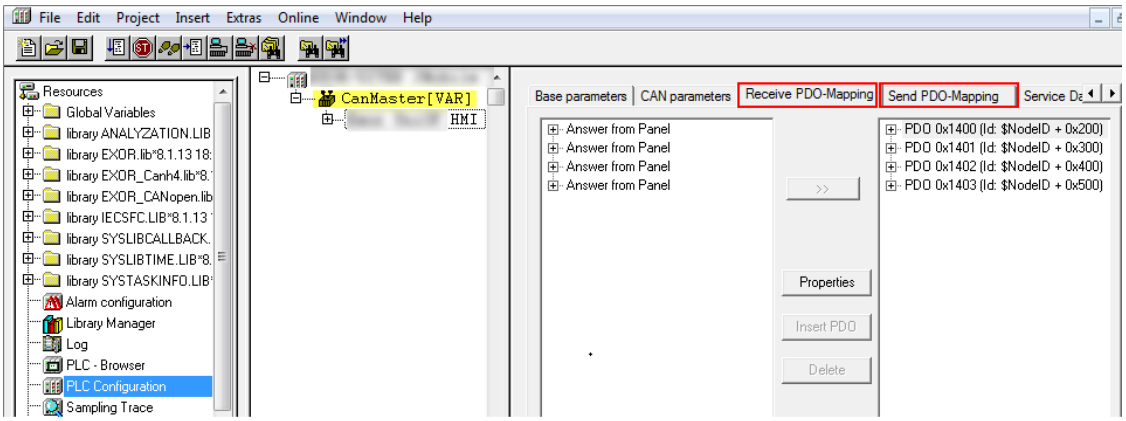




CODESYS V2 4PDO

In some cases it is useful to choose the model “CODESYS 4 PDO” where 4 PDO objects are used for transmission and 4 for reception. This solution may provide higher communication speed between the two devices.

To operate with 4 PDO the correct model should be set in HMI project and the PDOs for receive and transmit slots.



Note: CANopen Master PLC Configuration must be configured properly. In case of “CODESYS 4 PDO”.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Line Error	Returned when an error on the communication parameter setup is detected (baud rate); ensure the communication parameter settings of the controller is compatible with panel communication setup
Invalid response	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
CAN port not found	Make sure option module is correctly plugged
CAN port in use	Make sure option module is not already in use
General error	Error cannot be identified; should never be reported; contact technical support

CANopen SDO

CANopen SDO communication driver allows to communicate with CANopen Master devices over CAN ports of HMI.

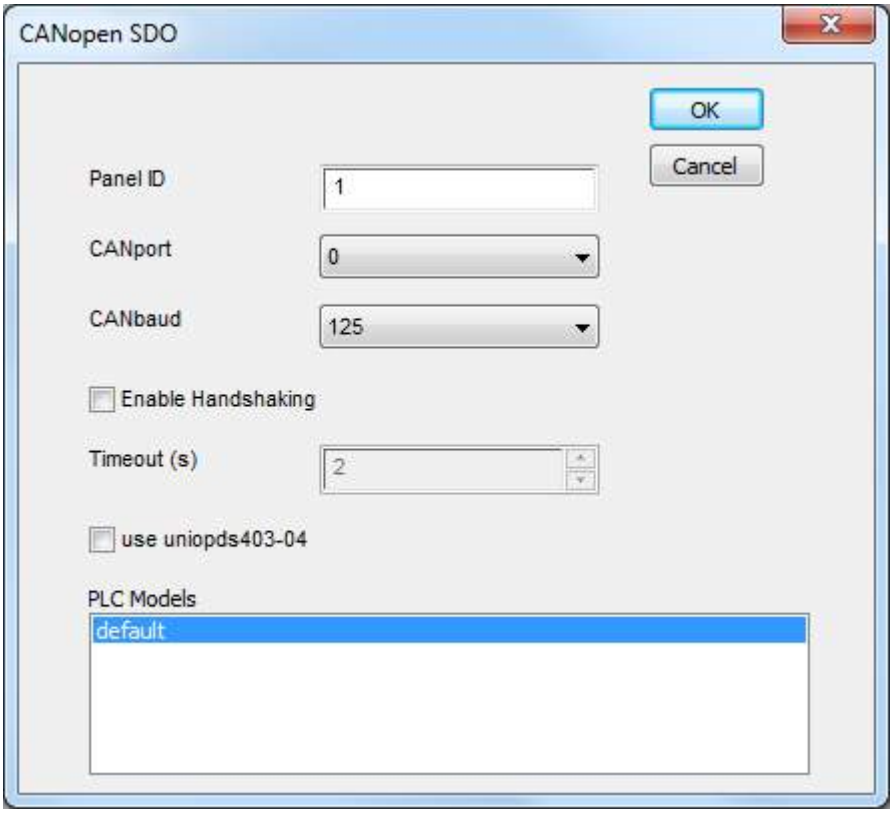
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



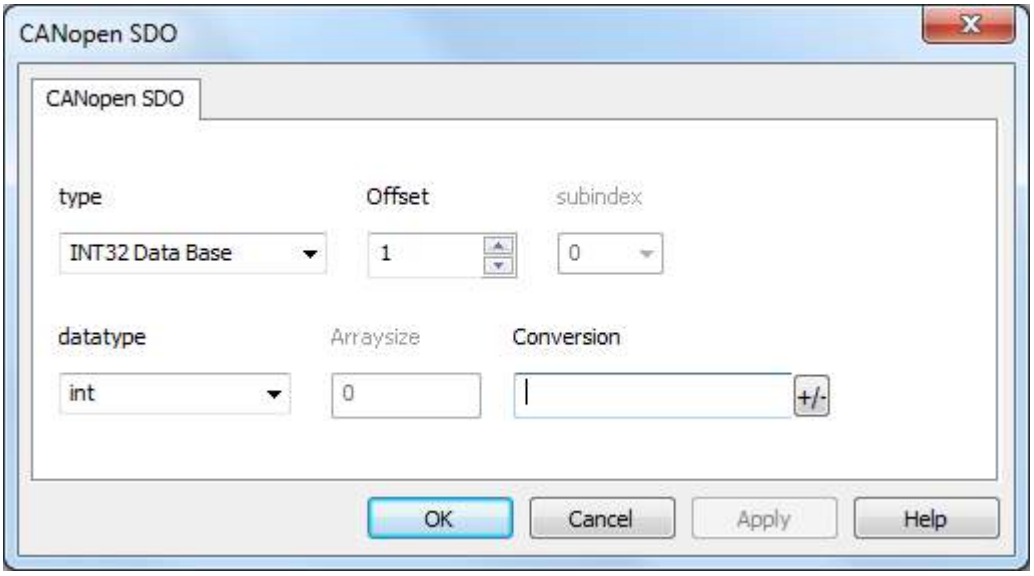
Element	Description
PanelID	Indicates the ID of the panel as a slave in the CANopen network. Allowed values are 1 to 127
CANport	Indicates the CAN port used. Allowed values are 0, 1, 2, 3 according to hardware platform.

Element	Description
CANbaud	Indicates the baudrate. Allowed values are 100, 125, 250, 500, 800, 1000.
Enable Handshaking	Allow handshaking with the CANopen Master during single or multiple write to PLC operations. If selected the NewData Ready flag (obj 0x9000 sub1) will remain at 1 until Master will read the indicated object, until a timeout. Default is FALSE.
Timeout (s)	Timeout for waiting for Master reading new data when Handshaking is enabled (in seconds) . Default is 2 seconds.
Use uniopDS403-04	Select the compatibility with uniopDS403-04.EDS file. If selected the object 0x9000 will be compatible with older versions. Default is FALSE (use uniopDS403-04B).
PLC Models	Fixed to default.

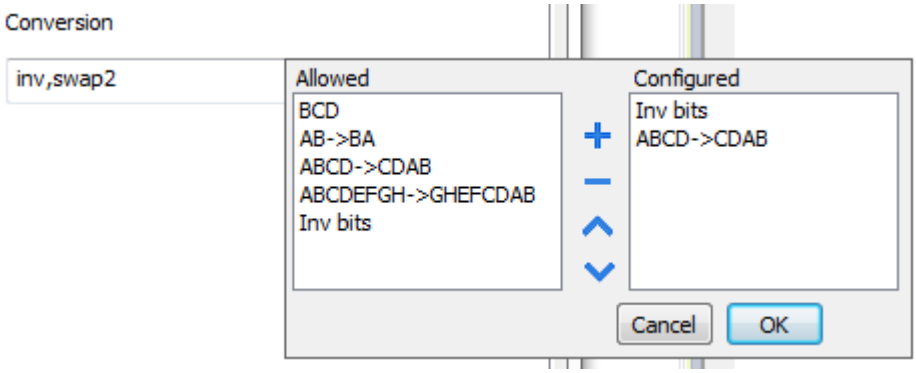
Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **CANopen SDO** from the **Driver** list: tag definition dialog is displayed.



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Type</td><td>Selects the internal data base in which the tag will be defined Possible values are:
INT232 Data Base (any combination of binary and decimal values)
FLOAT Data Base (single IEEE754 floating point values)</td></tr>
<tr><td>Datatype</td><td>For INT32 Data Base type, the available types are:
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
For FLOAT Data Base type the possible value is:
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
</table></td></tr>
</table>

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH ->	**swap4**: Swap bytes in a double word.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>GHEFCDAB</td><td>Example:
32FCFF54 → 54FFFC32 (in hexadecimal format)
855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000001
→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

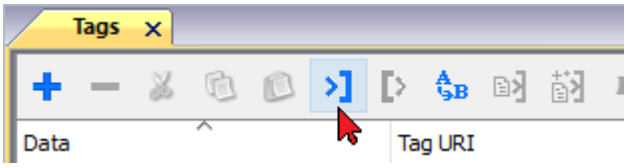
Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.

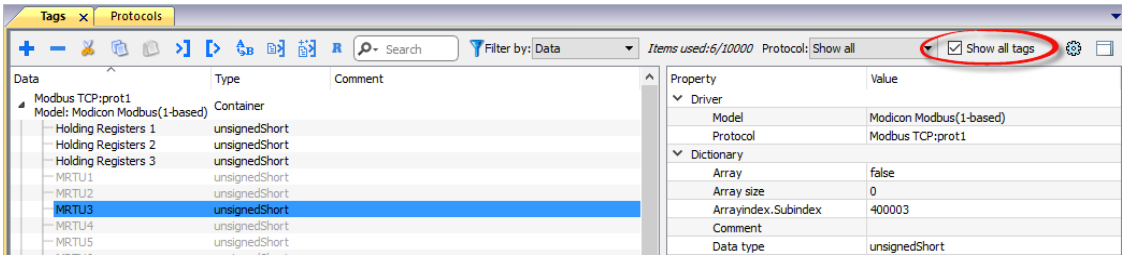


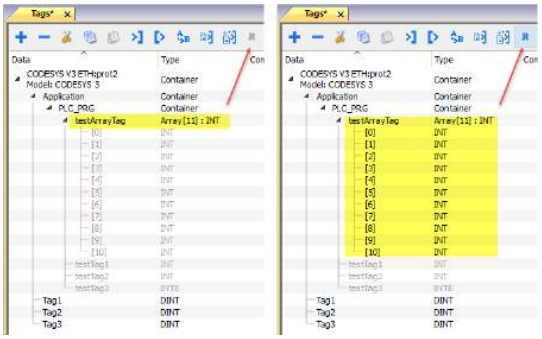
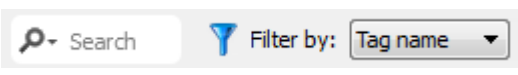
It is possible to import a Tag Editor exported xml

Type	Description
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Client System Variables

Client System Variables communication driver allows to create Tags that point to system information.

Refer to *Client System Variables > Protocol* chapter of User's Manual.

Protocol Editor Settings

Client System Variables communication driver allows to create Tags that point to system information.

Refer to *Client System Variables > Protocol* chapter of User's Manual.

CODESYS V2 ETH

CODESYS V2 ETH communication driver for supports communication through Ethernet connection with controllers based on the CODESYS V2.3 version.

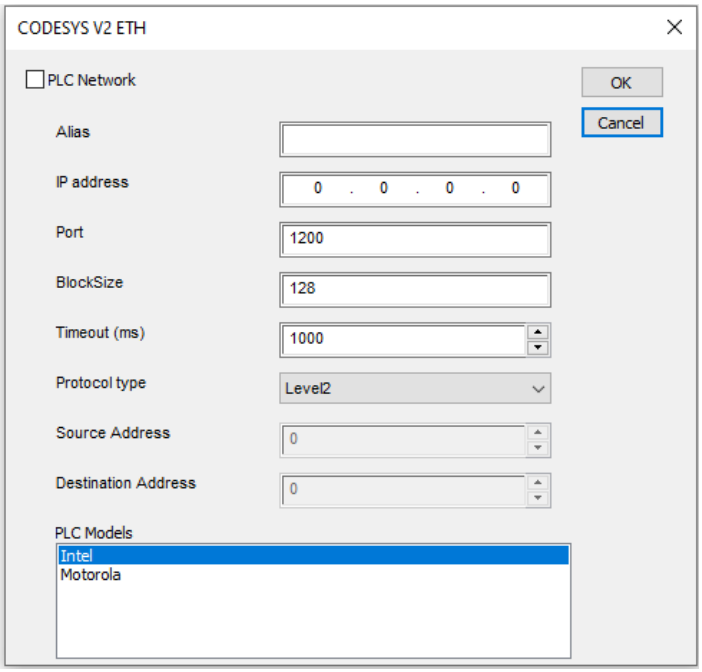
Protocol Editor settings

Adding a protocol

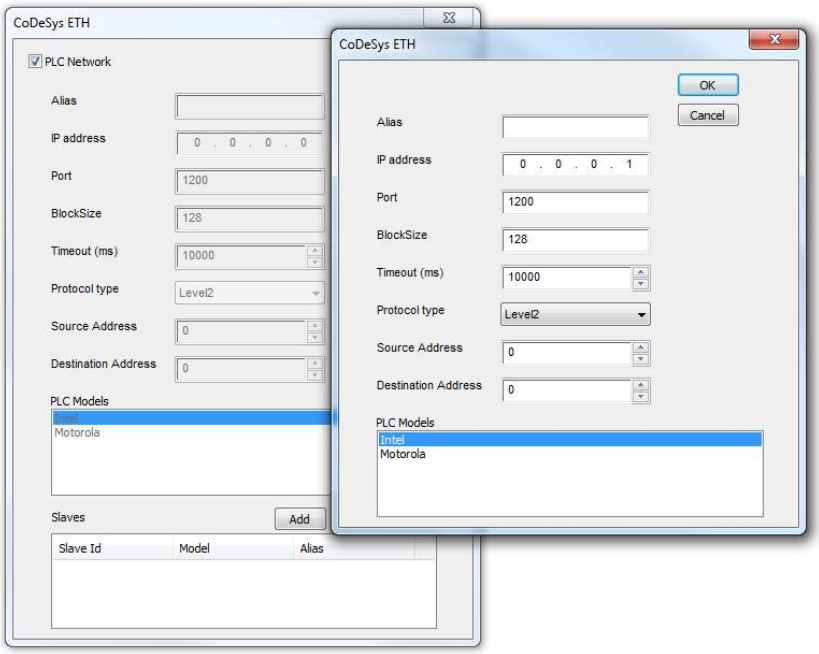
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	Port number used by the CODESYS V2 Ethernet driver. The default value is set to **1200**, which is also the default setting of CODESYS-based controllers.
Block Size	Maximum block size supported by your controller (limit is 1024 KB).
Timeout (ms)	Time delay in milliseconds between two retries of the same message when communication fails.

Element	Description
Protocol type	Protocol variant to be used. Please make sure you check which protocol variant is supported by the CODESYS runtime you want to connect.
Source Address, Destination Address	Available only when **TCP/IP Level 2 Route** is selected in **Protocol Type**. The Destination is the node of the PLC and allows the protocol to read variables in a sub-network. The address is used to read variables when multiple PLCs are connected in a sub-network (serial network) but only one have the Ethernet interface.
PLC Models	Two PLC models are available. • **Intel** • **Motorola**
PLC Network	IP address for all controllers in multiple connections. **PLC network** check box must be selected to enable multiple connections. 

CODESYS V2 Ethernet driver supports connection to multiple controllers starting from version V1.60.

Note: CODESYS V2 Ethernet driver is recommended when creating projects for the internal controller iPLC CODESYS. To use the CODESYS V2 Ethernet driver with iPLC, configure the IP address of the PLC as localhost (127.0.0.1).

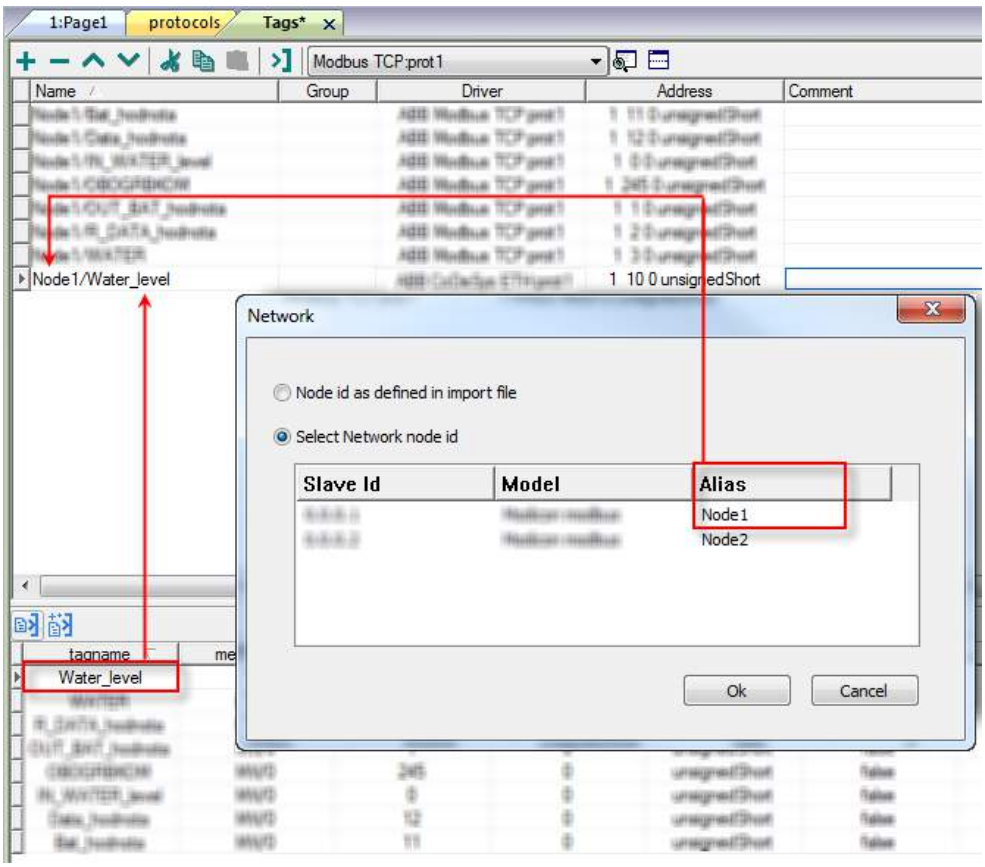
iPLC CODESYS supports communication with CODESYS V2 Ethernet driver with symbol based support starting from V1.55 and above.

Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

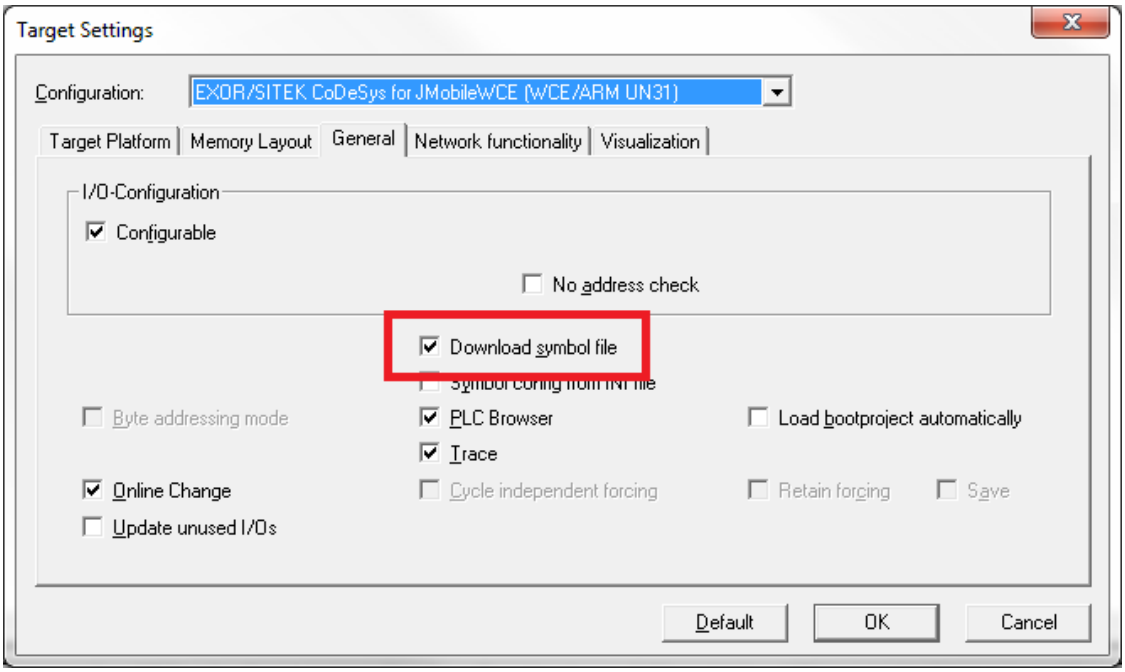
In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



Note: Aliasing tag names is only available for imported tags. Tags added manually in the Tag Editor cannot have the Alias prefix in the tag name.
The Alias string is attached at the time of tag import. If you modify the Alias string after the tag import has been completed, there will be no effect on names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

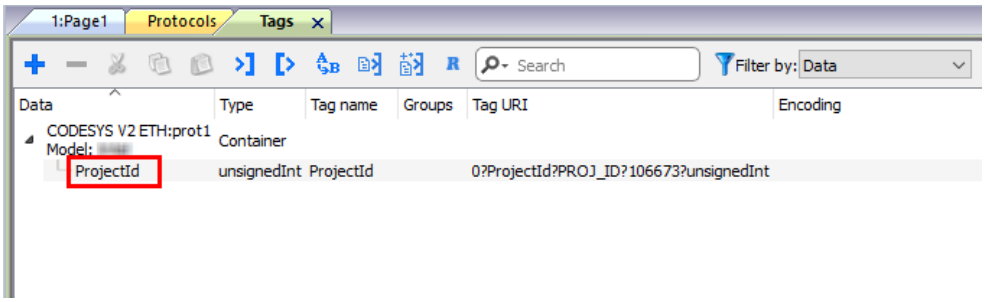
CODESYS software settings

When creating the project in CODESYS, select **Download symbol file**.



Note: CODESYS V2 Ethernet communication driver supports the automatic symbol file (SDB) upload from the PLC; any change in the tag offset due to new compilation of the PLC program does not require a symbol file re-import. Tag file has to be re-imported only in case of tag rename or definition of new tags.

When the option **Download symbol file** is not available or cleared, the protocol can work only if the **ProjectId** tag is imported. If the tag offset changes because of a new compilation of the PLC program, the symbol file must be re-imported.



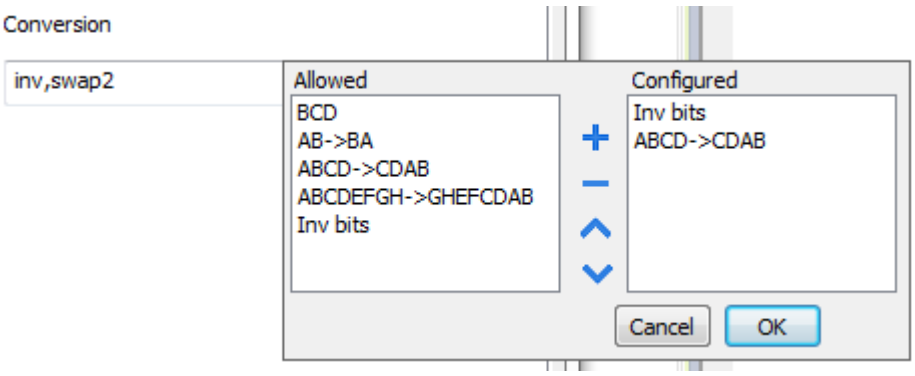
Data types

The import module supports variables of standard data types and user defined data types.

Supported data types	• BOOL • WORD • DWORD • INT • UINT • UDINT • DINT • STRING * • REAL • TIME • DATE & TIME and 1-dimensional ARRAY of the types above. See "Programming concepts" section in the main manual. Note *: String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.
Unsupported data types	• LWORD • LINT • LREAL

Tag conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

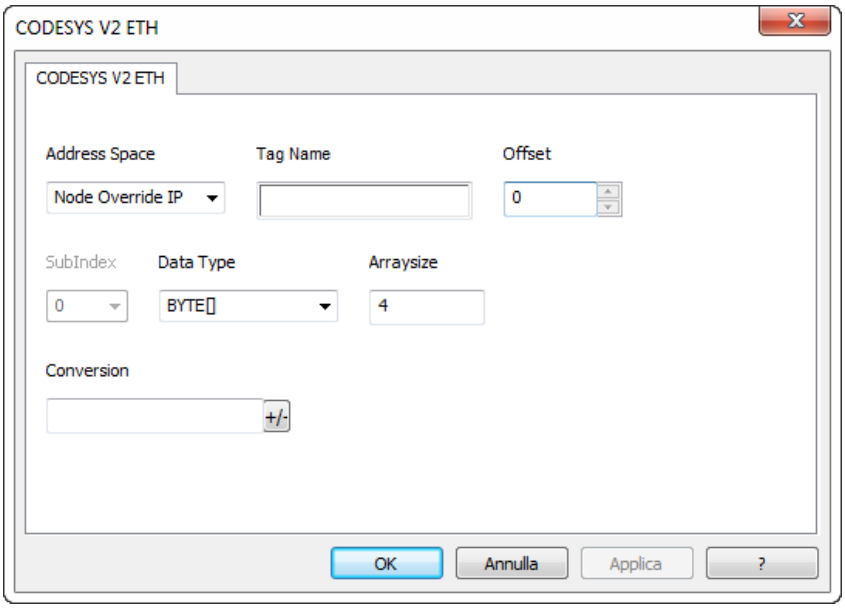
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.



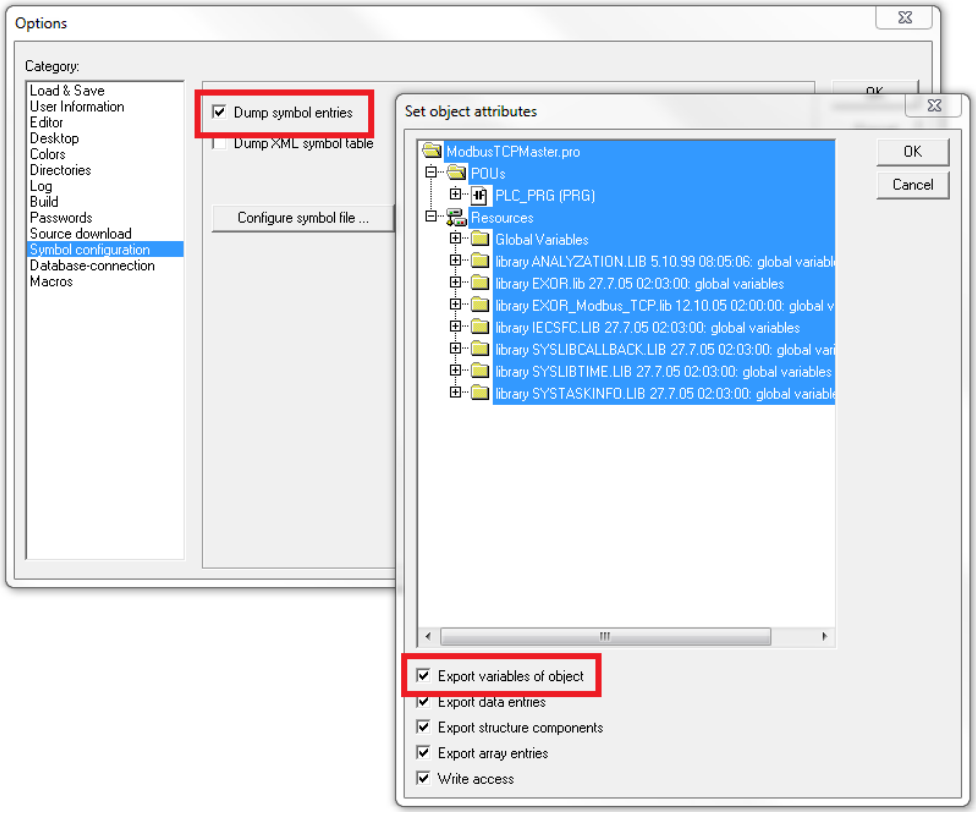
Tag Import

Exporting Tags from PLC

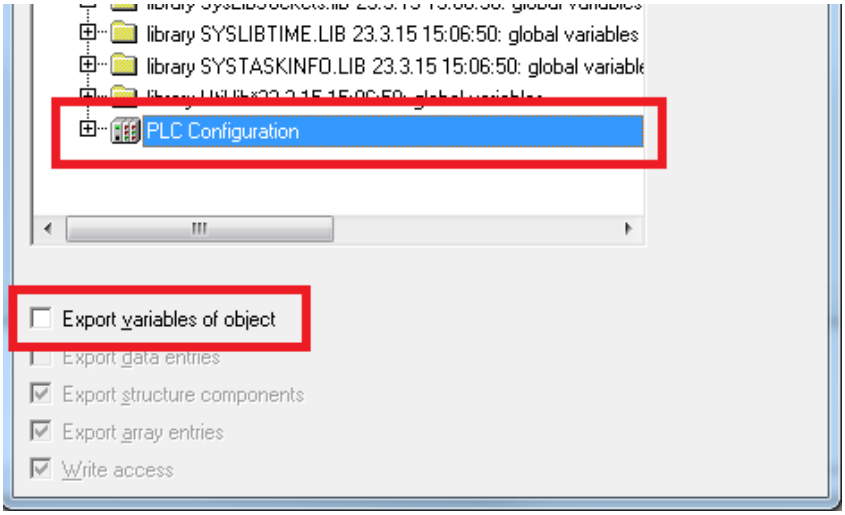
When configuring PLC using the manufacturer's configuration software, enable Symbol file (.sym extension) creation under the CODESYS programming software:

1. In the **Project** menu, click **Options**.
2. Click **Symbol configuration**.
3. Select **Dump symbol entries**.
4. Click **OK**.

Note: Click then **Configure symbol file…** and select **Export variables of object**. We recommend to clear the check box and re-select to be sure about the proper settings.

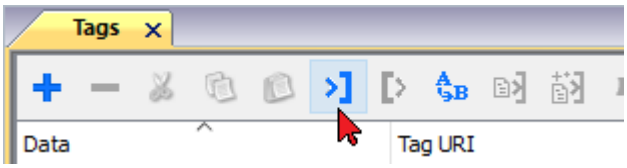


In some cases, duplication of symbols for variables associated to integrated I/O modules in the ".sym" file may be experienced. To remove the duplication selected the "PLC Configuration" voice from the objects list and uncheck the option "Export variables of object".

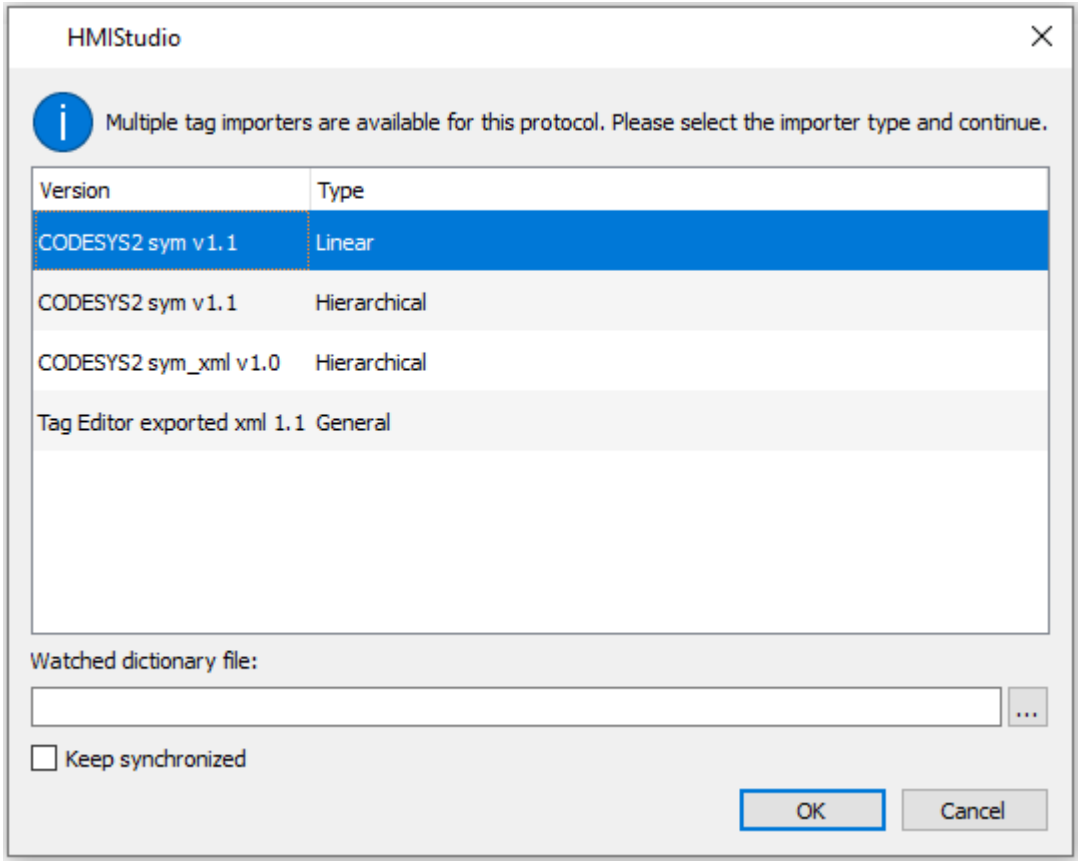


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



The following dialog shows which importer type can be selected.

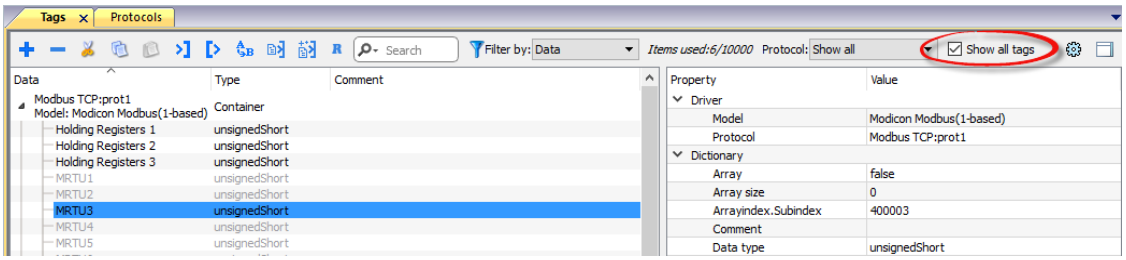


Importer	Description
CODESYS2 sym v1.1 Linear	Requires a **.sym** file. All variables will be displayed at the same level.
CODESYS2 sym v1.1 Hierarchical	Requires a **.sym** file. All variables will be displayed according to CODESYS V2 Hierarchical view.

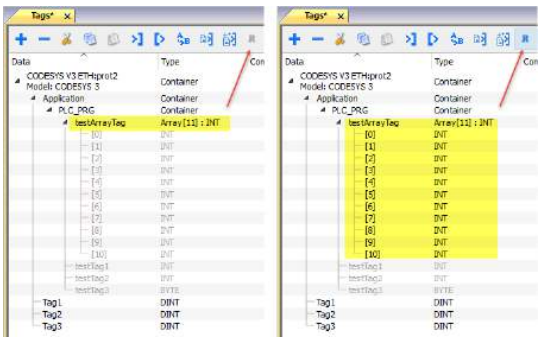
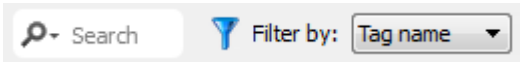
Importer	Description
CODESYS2 sym_xml v1.0 Hierarchical	Requires a **.sym_xml** file. All variables will be displayed according to CODESYS V2 Hierarchical view.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.

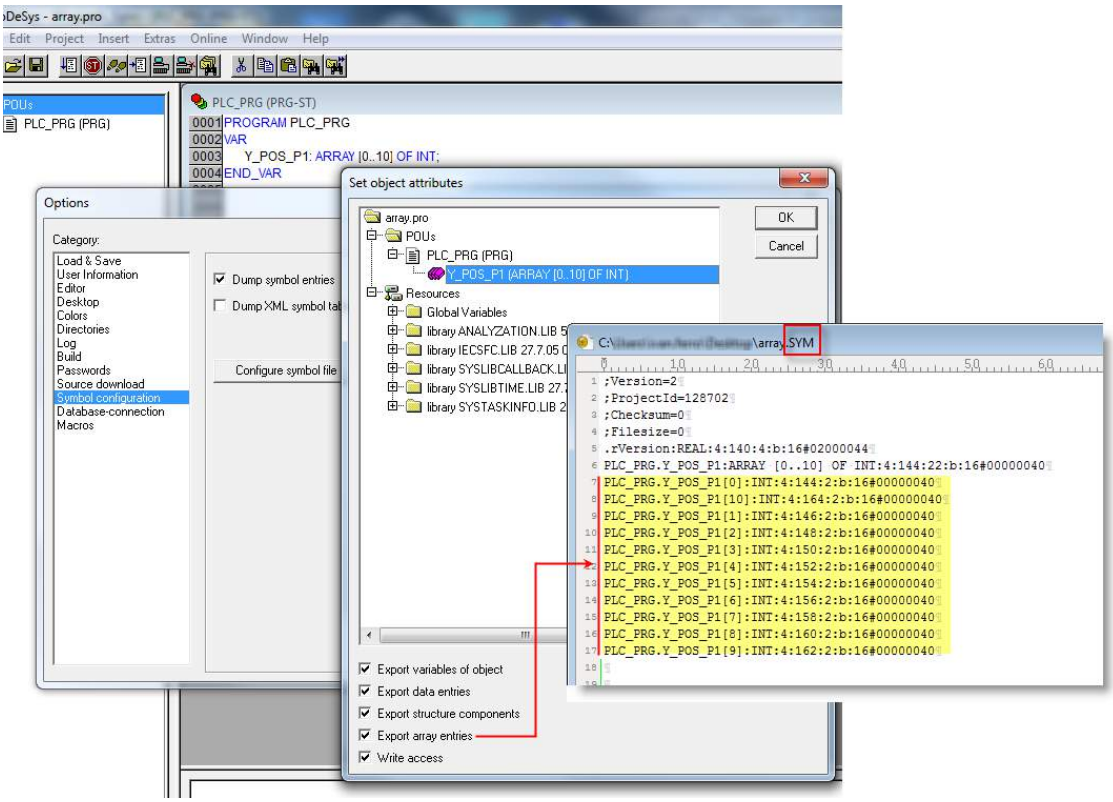


Toolbar item	Description
	Import Tag(s). Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	Update Tag(s). Click on this icon to update the tags in the project, due a new dictionary import.
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

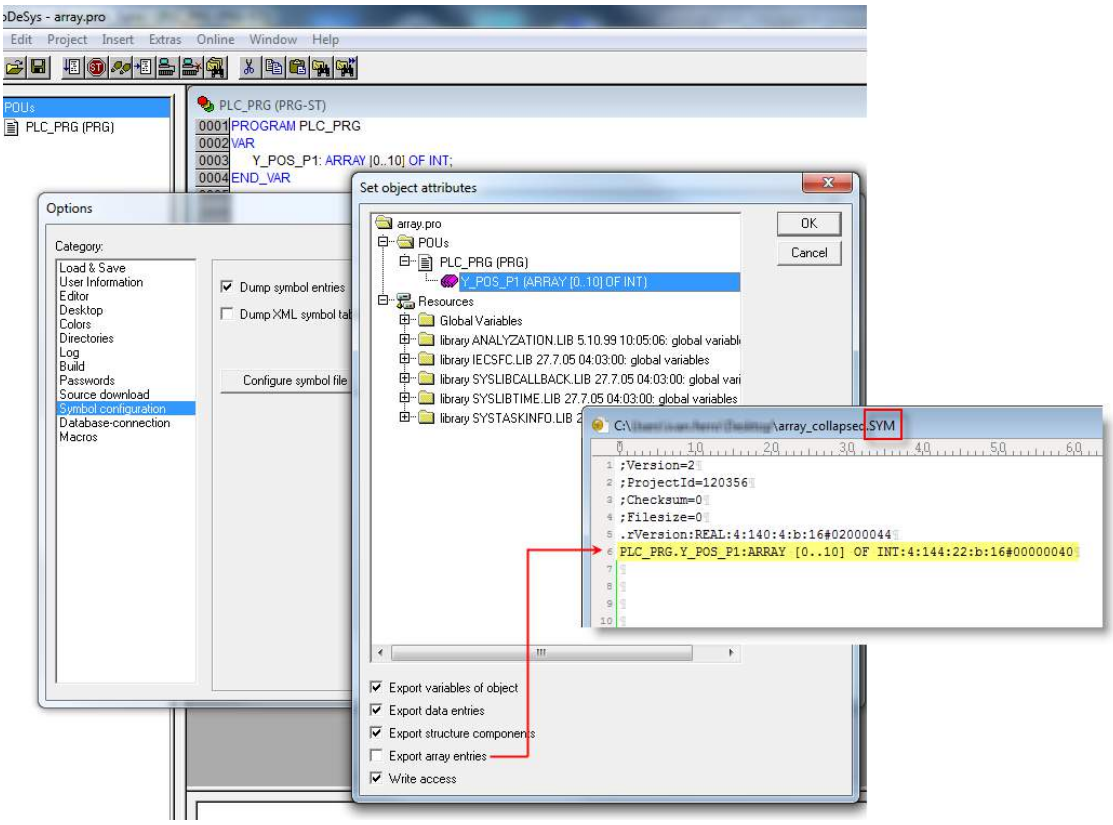
Exporting tag arrays

In CODESYS V2 program tag arrays are split into individual elements and one tag for each element is created. In the following example one array with 10 elements.

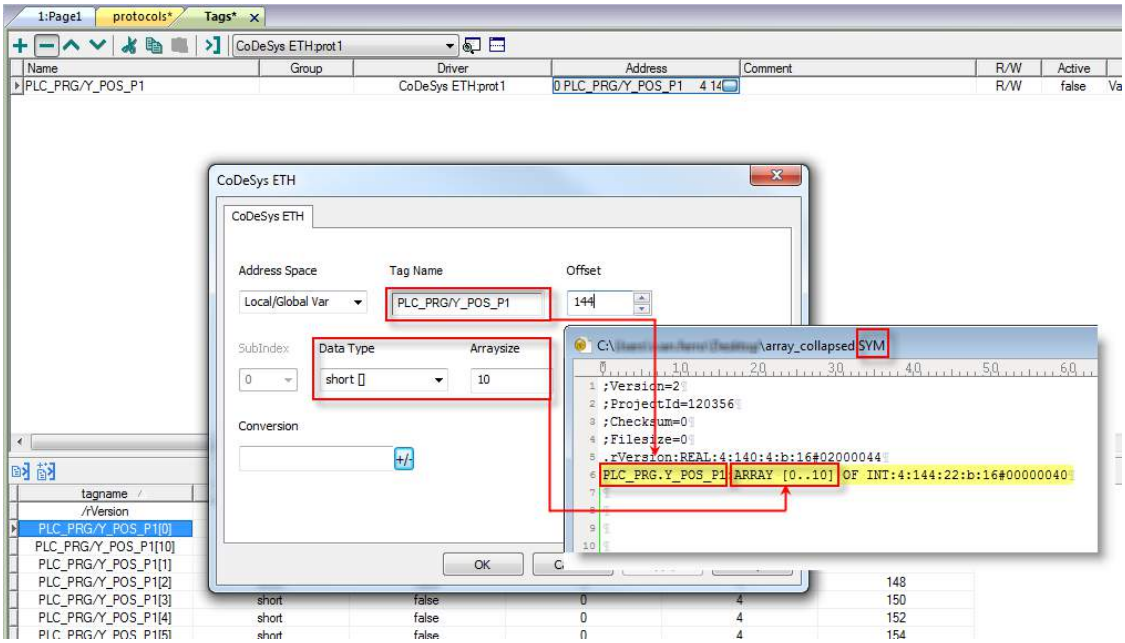


Note: If **Export array entries** is selected, a tag for each element will be created and exported into the .sym file. The entire tag list will be automatically imported into the Tag editor.

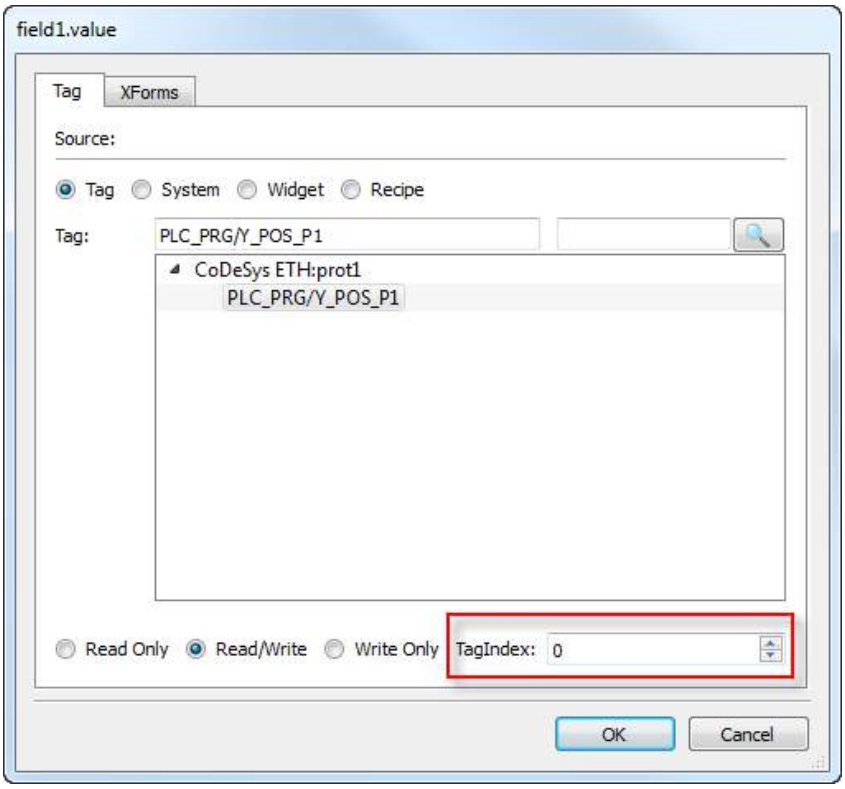
By clearing **Export array entries** only one tag for each one array can be created.



Note: When **Export array entries** has been cleared, only one tag is created and exported into the .sym file. The array is not automatically imported in the Tag editor and tags need to be manually configured in Tag editor.



All tag elements can be referenced in the editor using **TagIndex** in the **Attach to Tag** dialog.



Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported by this communication driver:

Error	Cause and action
Symbols file not present	Check Symbol file and download again the PLC program.
“tag” not present in Symbols files	Check if the Tag is present into the PLC project.
Time out on Acknoledge	Controller didn’t send acknowledge.
Time out on last Acknoledge	Controller didn’t sent last ack.
Time out on data reciving	Controller does not reply with data.
Connection timeout	Device not connected.

CODESYS V2 SER

The CODESYS V2 SER communication driver has been designed for serial communication with controllers based on CODESYS V2.3.

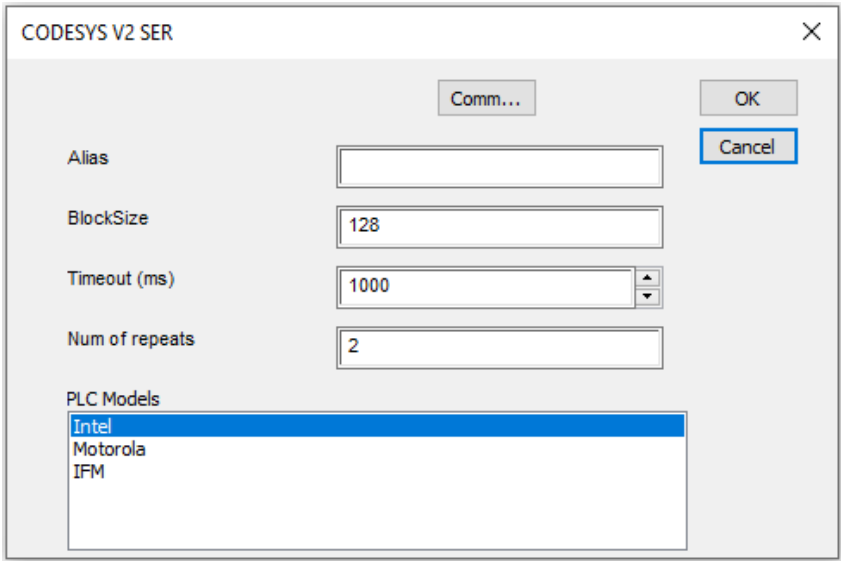
Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Accordingly, always ensure that the latest driver is used in the application.

Limitations

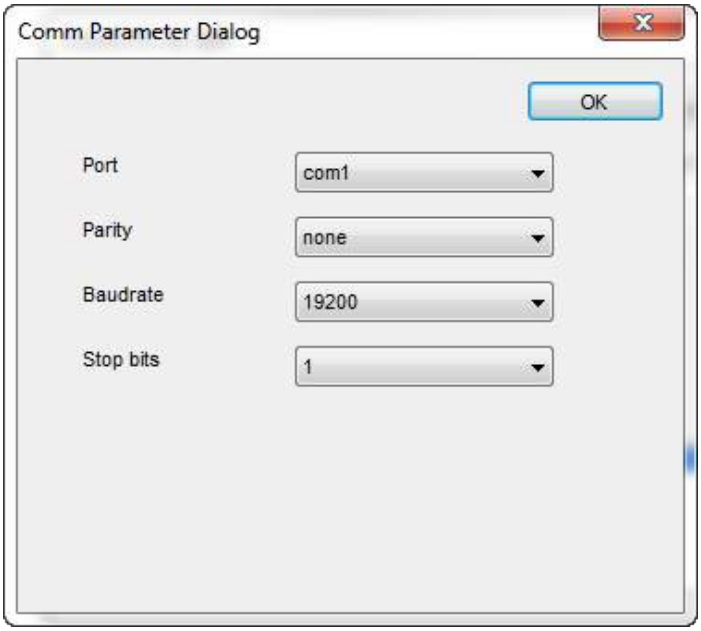
Max block size is 1024 byte.

Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “CODESYS Serial” from the list of available protocols.

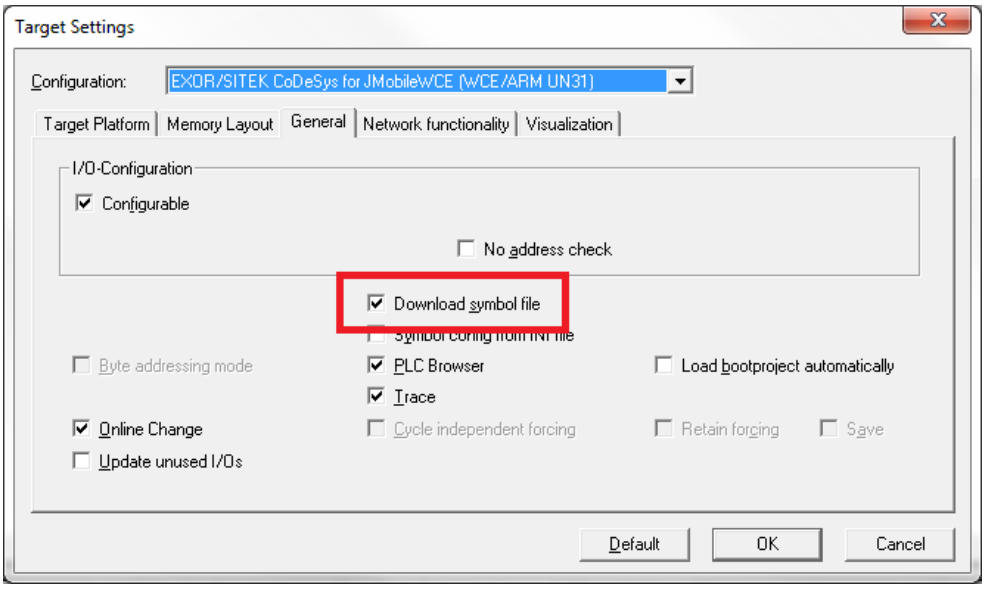


Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
Block Size	Enter the max block size supported by your controller (limit is 1024)
Timeout	The number of milliseconds between retries when communication fails
Num of repeats	This parameter defines the number of times a certain message will be sent to the controller before reporting the communication error status. A value of 1 for the parameter “No of repeats” means that the panel will eventually report the communication error status if the response to the first request packet is not correct.
PLC Model	Defines the byte order that will be used by the communication driver when sending

Element	Description
	communication frames to the PLC 
Port	Serial port selection. • COM1 is the PLC port. • COM2 is PC/Printer port on panels with 2 serial ports or refers to the optional plug-in module plugged in Slot 1/2 for panels with 1 serial port on-board. • COM3 refers to the optional plug-in module plugged in Slot 3/4 for panels with 1 serial port on-board.
Baudrate, Parity, Data bits, Stop bits	Communication parameters for the serial line.

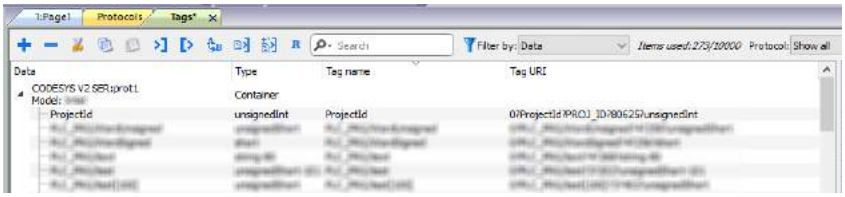
CODESYS Software Settings

When creating the project in CODESYS, the option Download Symbol File (in Target Settings/General) must be checked.



Note: CODESYS Serial communication driver supports the automatic symbol file (SDB) upload from the PLC; any change in the tag offset due to new compilation of the PLC program does not require a symbol file re-import. Tag file has to be re-imported only in case of tag rename or definition of new tags.

When the option Download symbol file is not available or not checked, the protocol can work only if the ProjectId tag is imported. Any change in the tag offset due to new compilation of the PLC program requires that symbol file is imported again.



Standard Data Types

The following data types in the CODESYS programming tool are considered standard data types by the import module:

BOOL

WORD

DWORD

INT

UINT

UDINT

DINT

STRING

REAL

TIME

DATE & TIME

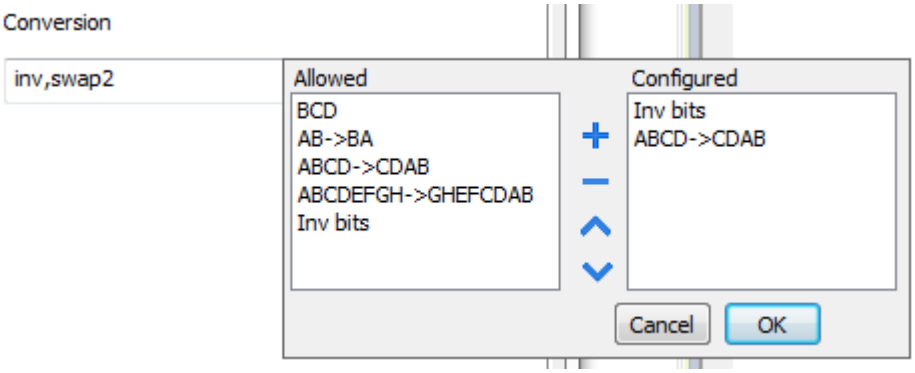
and 1-dimensional ARRAY of the types above.

The 64-bit data types LWORD, LINT and LREAL are not supported.

String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35)) or default size (str: STRING) which is 80 characters.

Tag Conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)

Value	Description
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

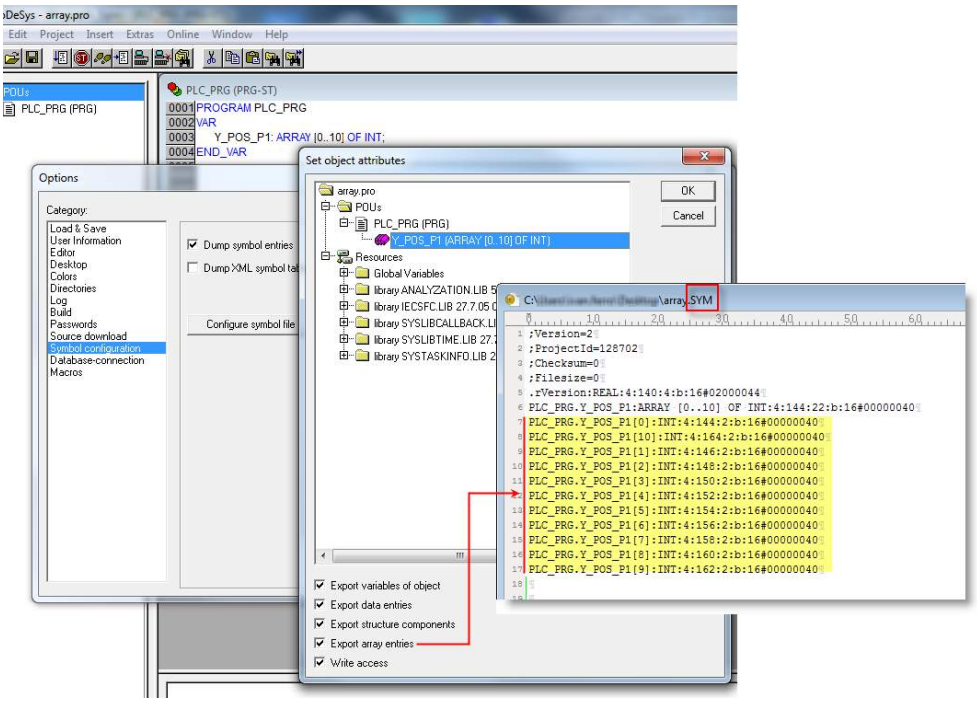
Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

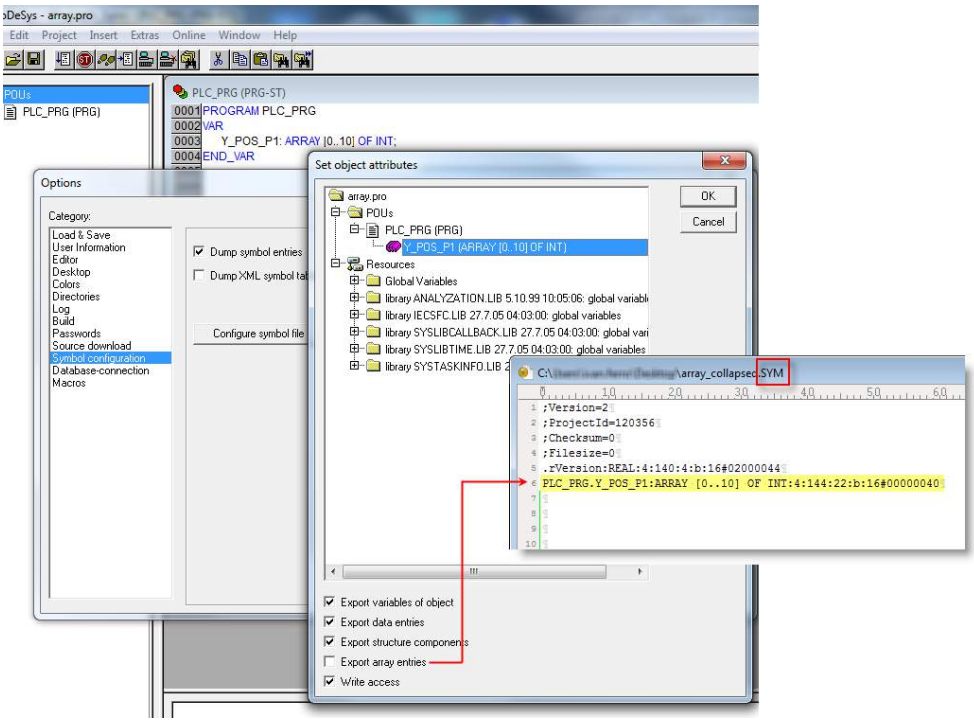
Tag Array

Tag Arrays are split into individual elements and one Tag for each element is created. The figure below shows an example of one Array with 10 elements

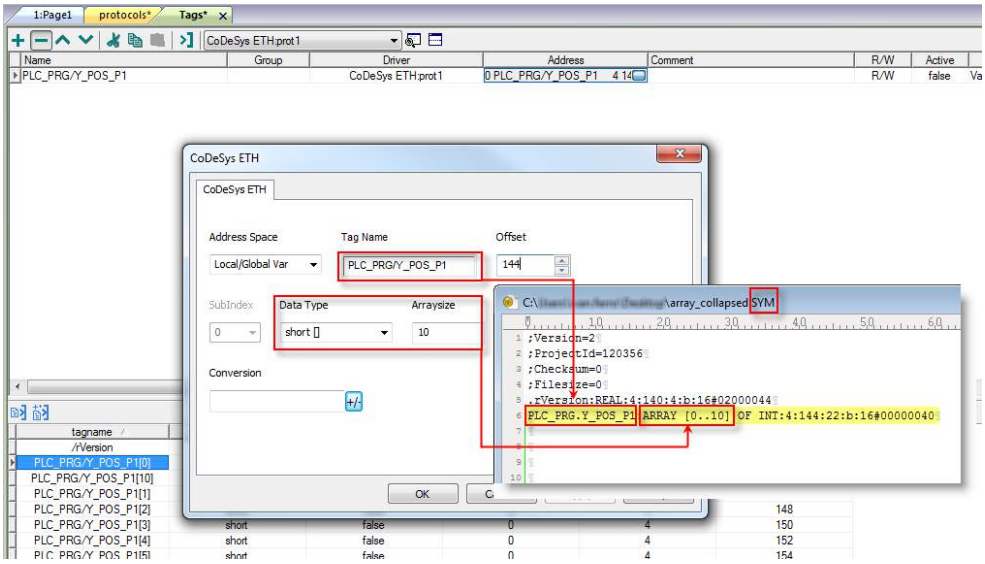


Note: When “Export array entries” is set, a tag for each element is created and exported into the SYM file. The entire tag list is automatically imported into Tag Editor.

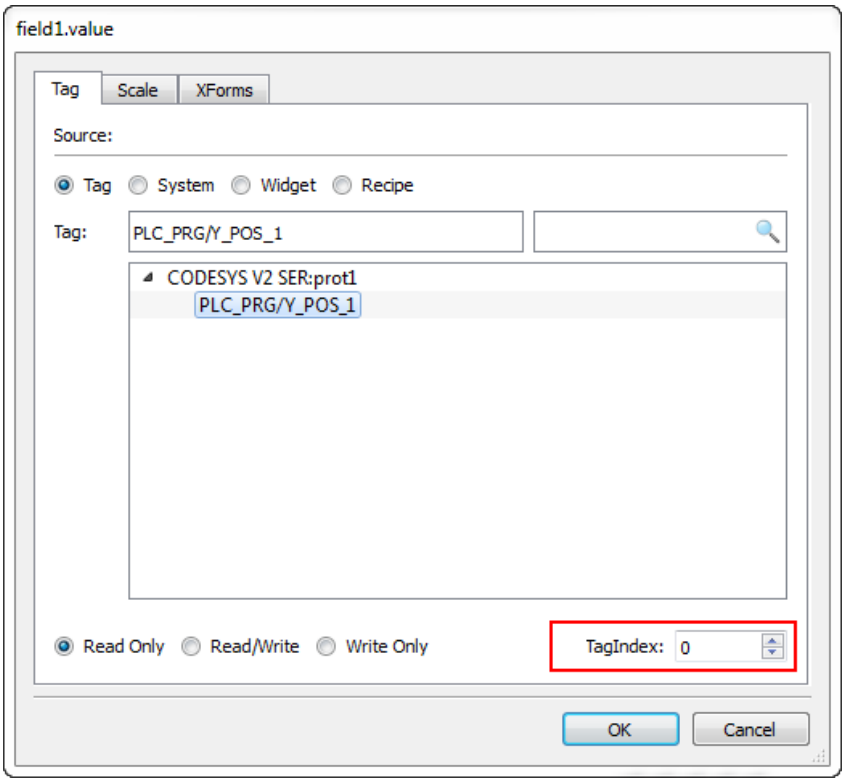
The amount of tags can be reduced and only one Tag for each one array can be created by removing the checkbox “Export array entries”, see figure below.



Note: When “Export array entries” is not set, only one tag is created and exported into the SYM file. The Array will not be automatically imported in Tag Editor and Tags need to be manually configured in Tag Editor



All Tag elements can be referenced in the editor using “TagIndex” in the “Attach to Tag” dialog



Aliasing Tag Names in Network Configurations

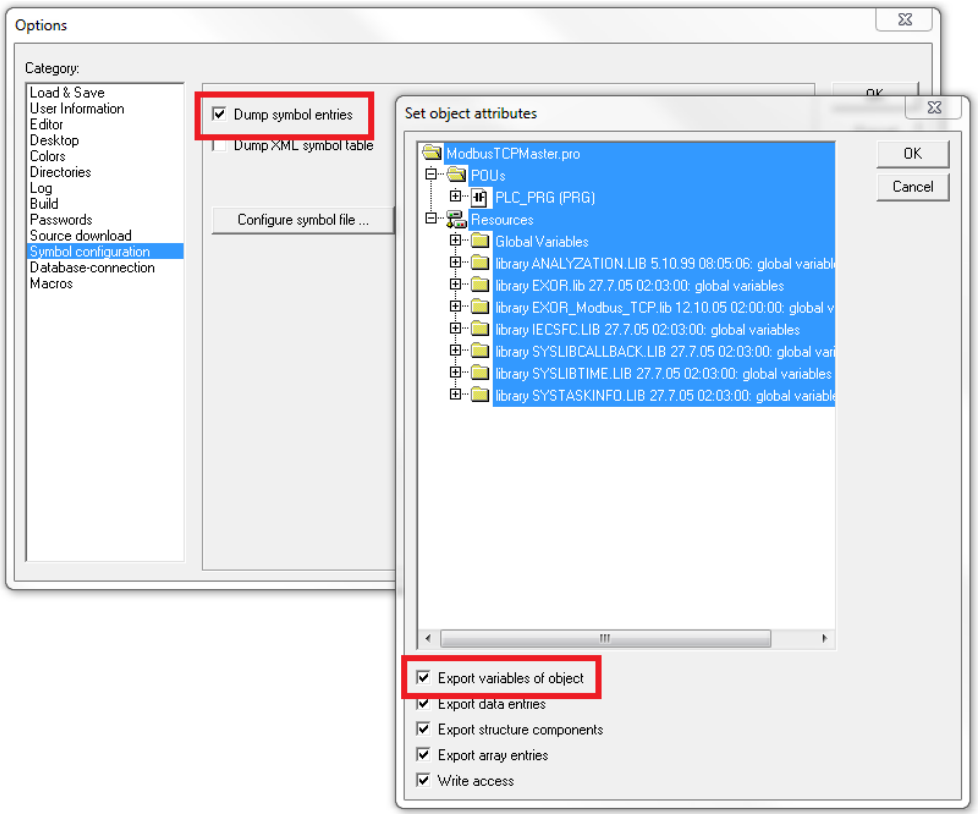
Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”.

Note: An Aliasing tag name is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

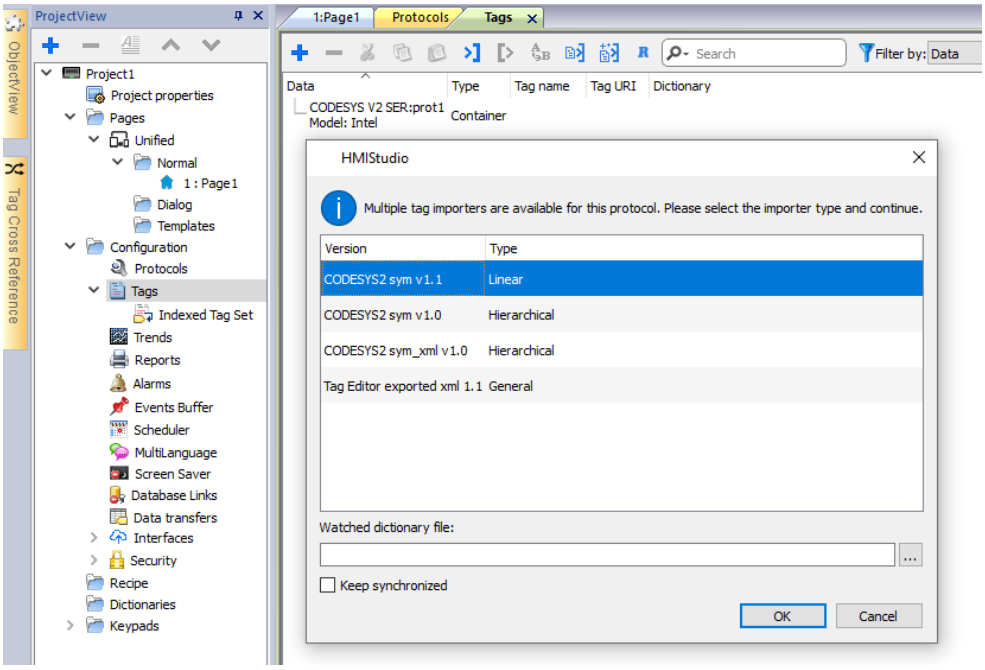
Tag Import

When configuring PLC using the manufacturer’s configuration software, make sure to enable Symbol file creation (file with .SYM extension). It can be done under the CODESYS programming software, by selecting “Project\Option\Symbol configuration” and mark the check box “Dump symbol entries” as shown in the picture below.



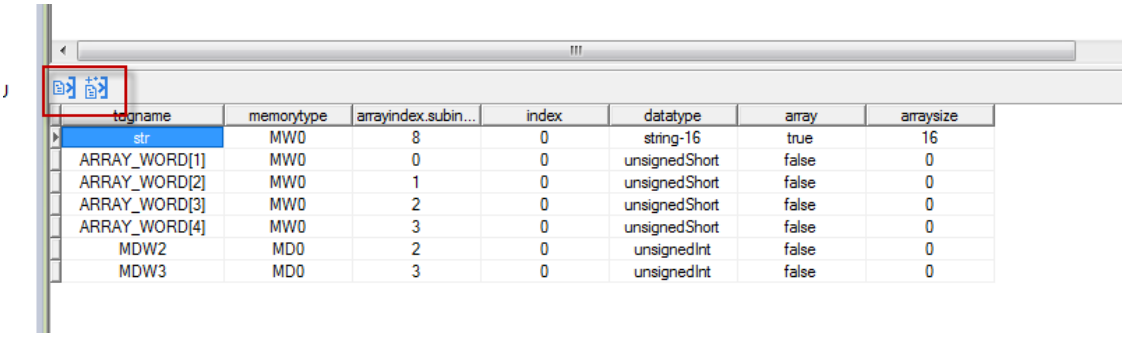
Note: Click then on the “Configure symbol file…” button and make sure the “Export variables of object” check box is marked as shown in the following picture. We recommend to un-check the check box and mark it again to be sure about the proper settings.

Select the driver in the Studio tag editor and click on the “Import tag” button to start the importer.



Once the importer has been selected, locate the symbol file and click Open.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in the following figure.



tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
Symbol file not present	Check Symbol file and download again the PLC program
“tag” not present in Symbol file	Check if the Tag is present in the PLC project
Time out on Acknowledge	Controller didn’t send acknowledge
Time out on last Acknoledge	Controller didn’t send last acknowledge
Time out on data receiving	Controlled does not reply with data
Connection timeout	Device not connected

CODESYS V3 ETH

The CODESYS V3 ETH communication driver supports communication thought Ethernet connection with controllers based on the CODESYS V3 PLC software by the company 3S.

Note: To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Make sure the latest driver is used in the application.

Note: Changes in the controller protocol or hardware may have occurred since this documentation was created. This may interfere with the functionality of this driver. Therefore, always test and verify the functionality of the application.

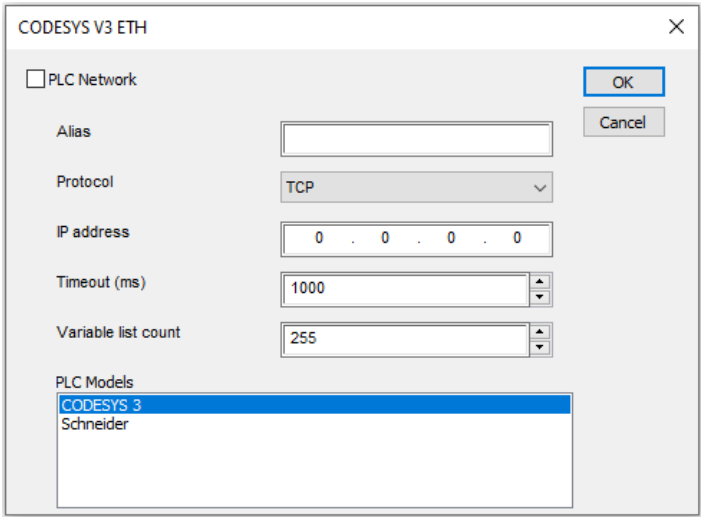
Protocol Editor Settings

Adding a protocol

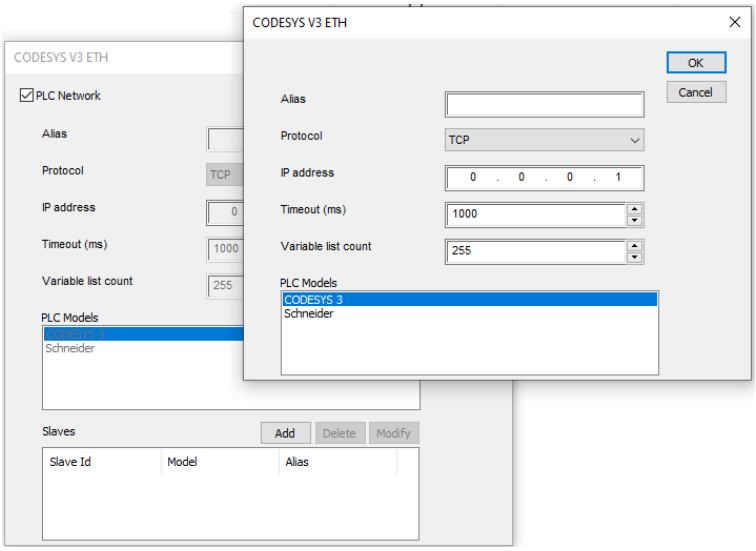
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Protocol	Select between TCP and UDP protocol type.
IP address	Ethernet IP address of the controller
Variable	Variable List is the best method to achieve higher performance in the CODESYS V3

Element	Description
list count	communication protocol, as it allows requesting multiple data items in a single protocol session. Since some implementations of CODESYS V3 at runtime have a limited number of Variable Lists that can be allocated, this parameter allows you to set the maximum number of Variable Lists the communication driver tries to create in the PLC.
PLC Model	Byte order that will be used by the communication driver when sending communication frames to the PLC.
Timeout	Number of milliseconds between retries when communication fails.
PLC Network	Enable access to multiple networked controllers. For every controller (slave) set the proper option. 

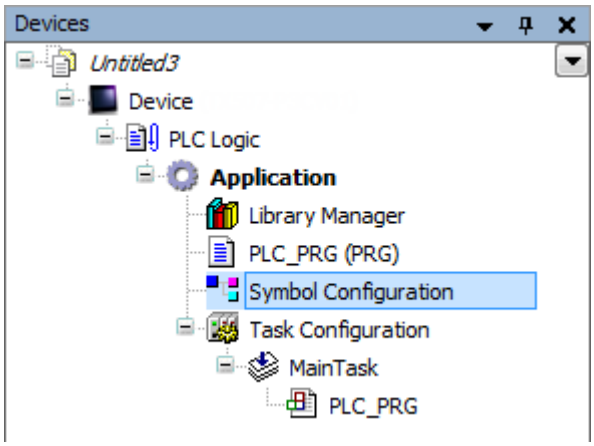
Note: Refer to the controller documentation to verify required values for the parameters **Full node address** or **Variable list count**.

Tag Import

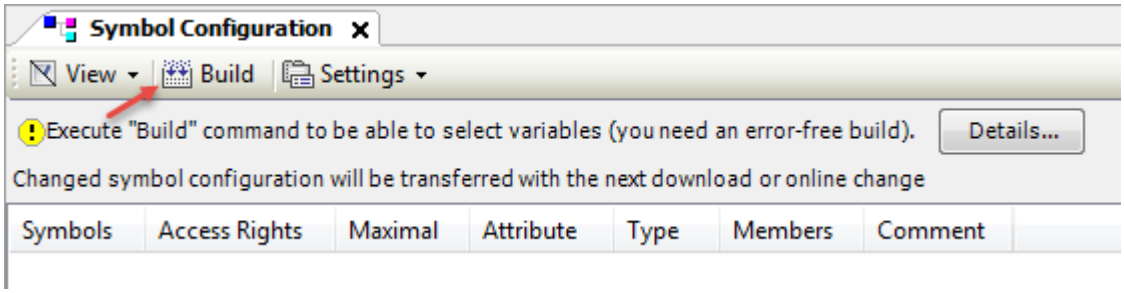
Exporting Tags from PLC

When creating the project using CODESYS V3, properly configure the symbol file to contain the required variables.

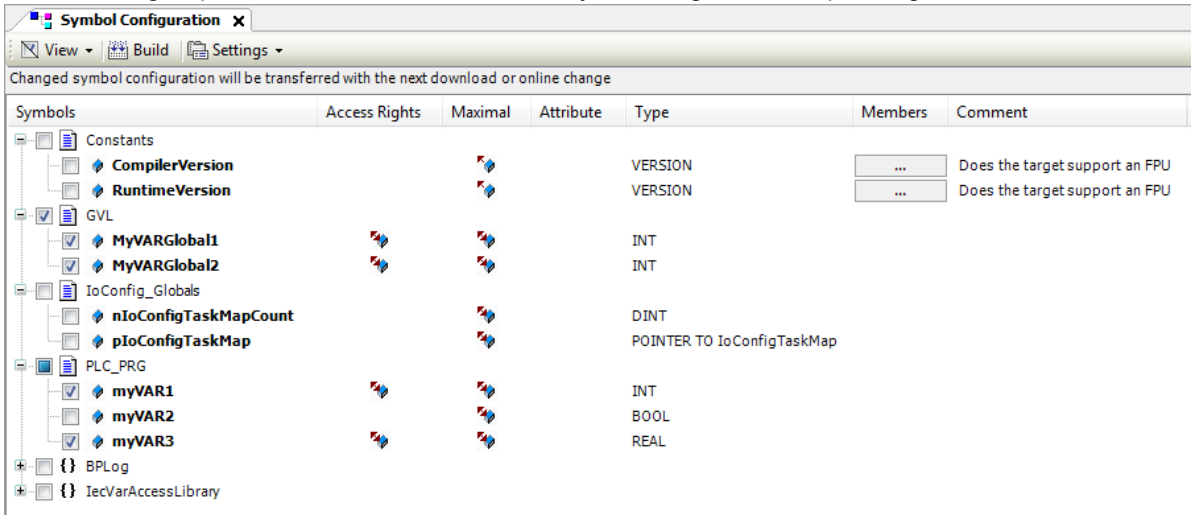
1. To add the Symbol configuration in CODESYS V3 project, right click on the Application item from the project tree, then into the context menu select Add Object > Symbol configuration. The symbol configuration item will be added to the project tree.



2. Double click on Symbol configuration item, then click on "Build" button.

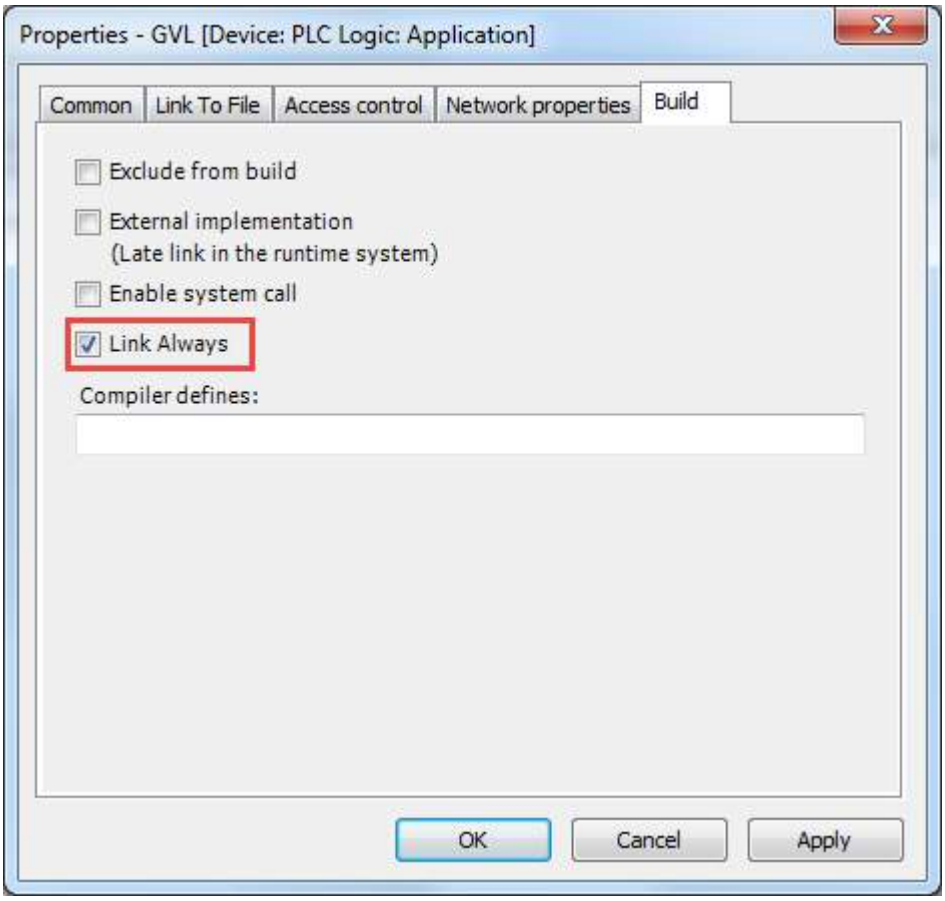


3. Symbol configuration item contains a list of all the variables available into the CODESYS V3 project, single variables or groups of variables can be selected by checking the corresponding item in the list.



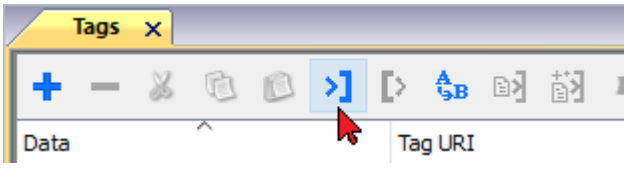
4. After the symbols have been configured, download the project or use the **Generate code** function (Build > Generate code) to create an .xml file containing all the variables read to be imported in the Tag Editor.

Note: GVL global variables are listed in Symbols Configuration only if they are used in PLC program.
To always list global variables right click on GVL and select "Properties". From "Build" tab check "Link Always" option.

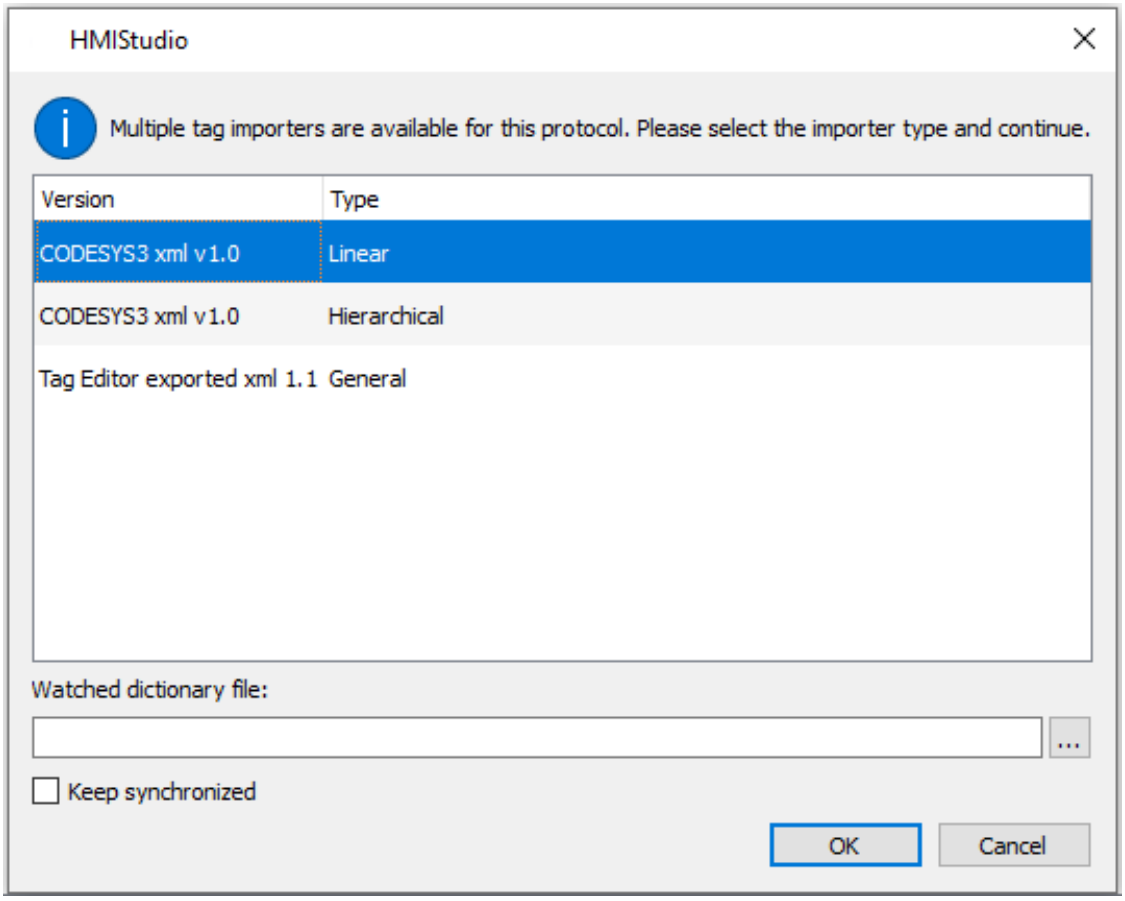


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



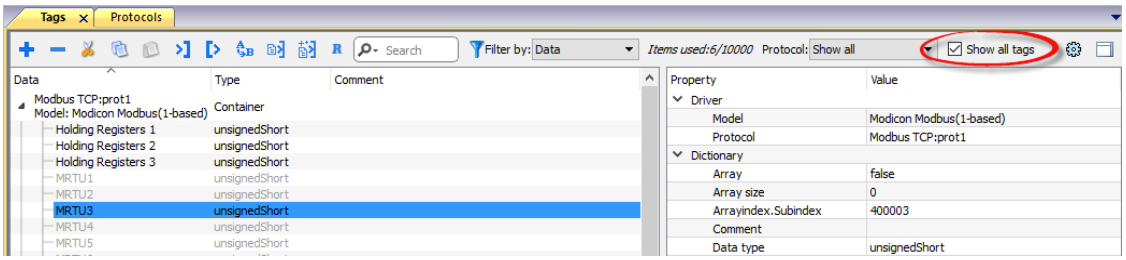
The following dialog shows which importer type can be selected.

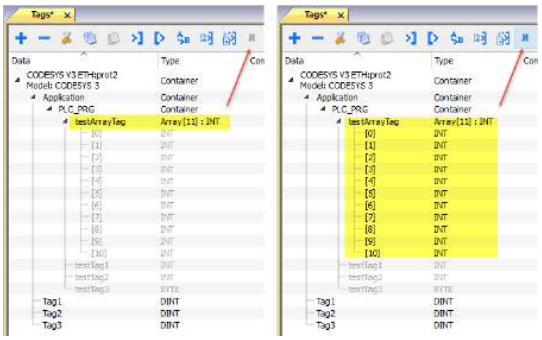
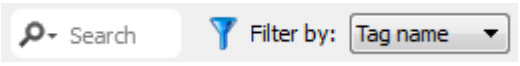


Importer	Description
CODESYS3 xml v1.0 Linear	Requires an **.xml** file. All variables will be displayed at the same level.
CODESYS3 xml v1.0 Hierarchical	Requires an **.xml** file. All variables will be displayed according to CODESYS V3 Hierarchical view.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



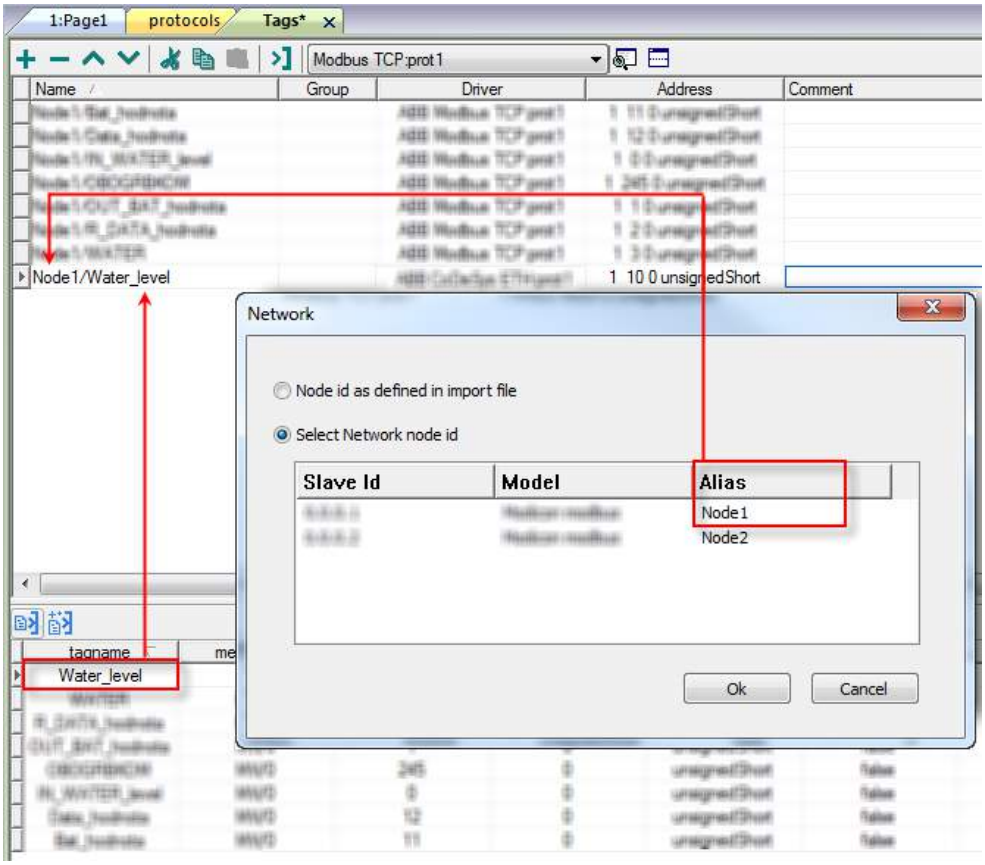
Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



Note: Aliasing tag names is only available for imported tags. Tags added manually in the Tag Editor cannot have the Alias prefix in the tag name.
The Alias string is attached at the time of tag import. If you modify the Alias string after the tag import has been completed, there will be no effect on names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

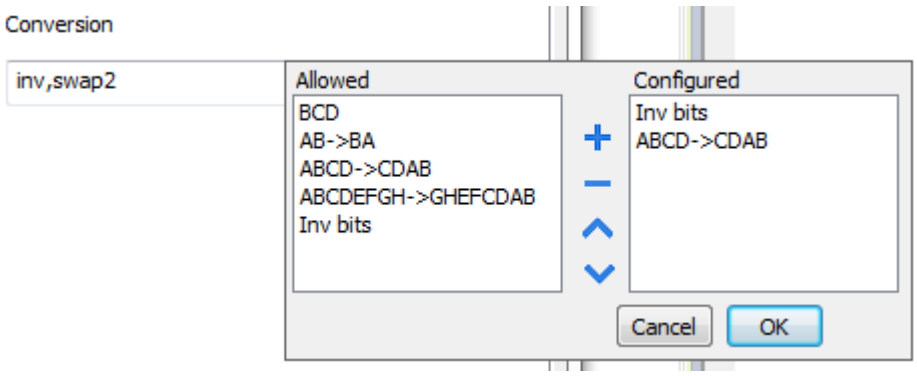
Data Types

The import module supports variables of standard data types and user defined data types.

Supported data types	• BOOL • INT • SINT • UINT • UDINT • DINT • STRING* • REAL • LREAL • BYTE • ULINT • LINT and 1-dimensional ARRAY of the types above. See "Programming concepts" section in the main manual. Note *: String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.
Unsupported data types	• LWORD • LINT

Tag conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

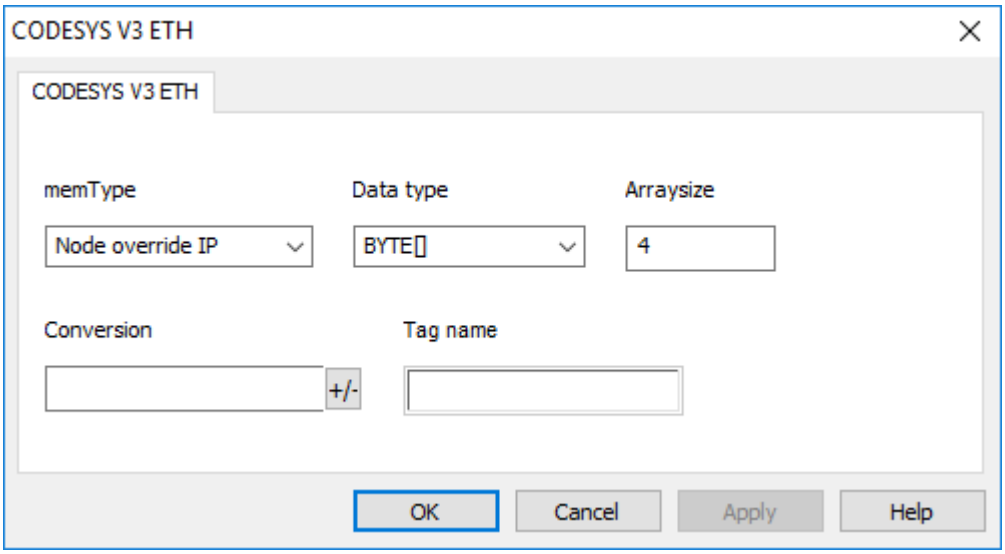
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.



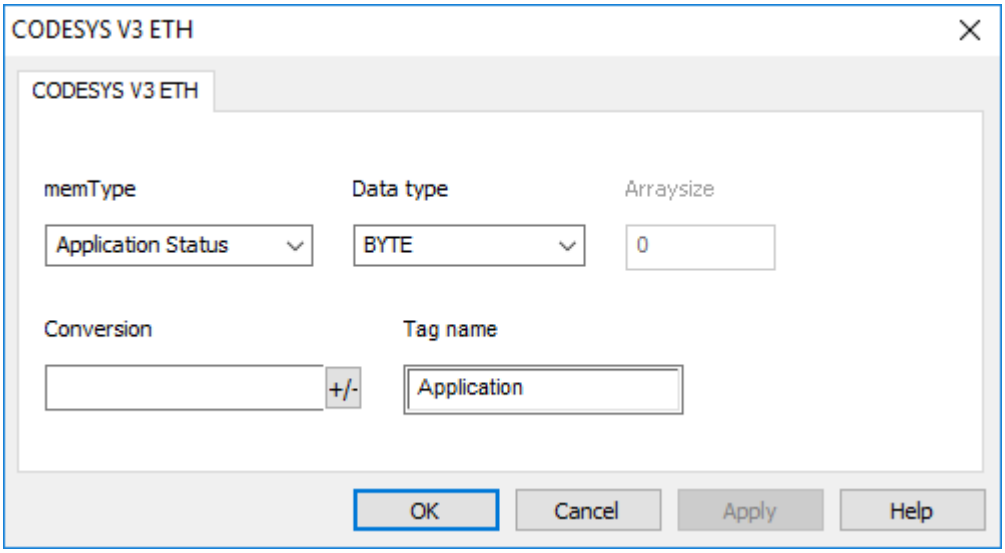
Application Status

The protocol provides the special data type Application Status which allows you to check or change the applications status.

Functionality available only if supported by the CODESYS device

The tags pointing to Application Status must contains into field "**Tag name**" the name of the PLC application (frequently the default name is "Application")

If the HMI device is connected to a network with more than one controller node, each node has its own Application Status variable.



Application Status	Description
0	RUNNING
1	STOPPED
2	HALTED ON BreakPoint It is not possible to write 2 as new status
251	Reboot CODESYS device
252	Shutdown CODESYS
253	Reset ORIGIN
254	Reset COLD
255	Reset WARM

Communication Status

Current communication status can be displayed using System Variables. See "System Variables" section in the main manual.

Control Techniques Modbus TCP

Control Techniques Unidrive M Series are using Modbus TCP protocol where the device id should be always set to 0 or 255. This communication protocol is known as Control Techniques Modbus TCP. The HMI protocol identifies Control Techniques Modbus TCP devices using their IP addresses

You should take note of these addresses as you assign them because you will need them later in the set-up phase of the user interface application. The HMI protocol can be set to access to a different menu range

Different physical media, gateways, routers and hubs can be used in the communication network. Also, other devices can independently make simultaneous use of the network. However, it is important to ensure that the traffic generated by these devices does not degrade the communication speed (round-trip time) to an unacceptable level.

The implementation of the protocol operates as a Modbus TCP client only.

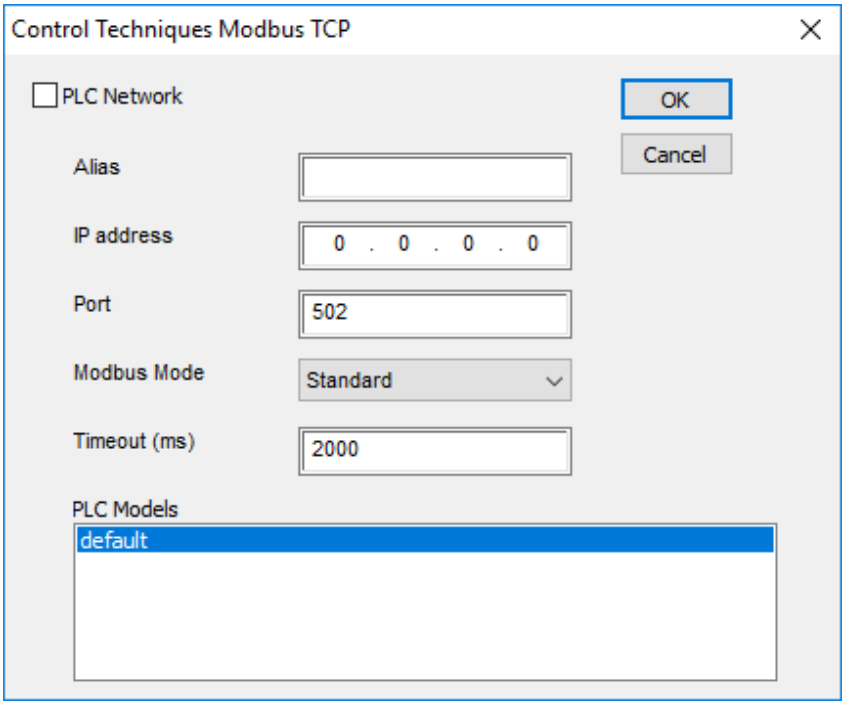
The HMI Control Techniques Modbus TCP protocol uses the standard port number 502 as the destination port.

The HMI Control Techniques Modbus TCP protocol supports the standard commonly referred as “Ethernet II”.

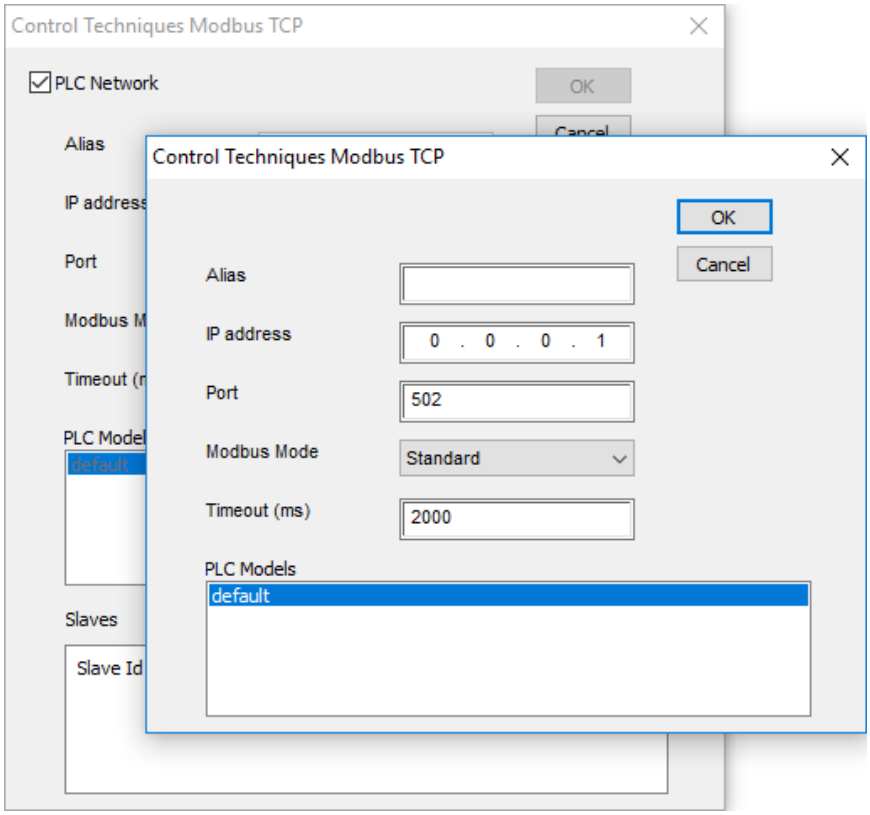
Protocol Editor Settings

Add (+) a new driver in the Protocol editor and select the protocol called “Control Techniques Modbus TCP” from the list of available protocols.

The driver configuration dialog is shown in figure.



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Alias</td><td>Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node</td></tr>
<tr><td>IP address</td><td>Ethernet IP address of the controller</td></tr>
<tr><td>Port</td><td>Port number used by the Modbus TCP driver; the default value can be changed when the communication goes through routers or Internet gateways where the default port number is already in use</td></tr>
<tr><td>Modbus Mode</td><td>This parameter define the communication protocol used and needs to be set in according with the setting made on the drive (parameter S.15.013). Modified mode is provided to allow register numbers up to 255 to be addressed. If any menus with numbers above 63 should contain more than 99 parameters, then these parameters cannot be accessed via Modbus.
<table>
<tr><th>Protocol</th><th>Register address</th></tr>
<tr><td>Standard</td><td>(menu number * 100) + parameter number - 1
where menu number ≤ 162 and parameter number ≤ 99</td></tr>
<tr><td>Modified</td><td>(menu number * 256) + parameter number – 1
where menu number ≤ 63 and parameter number ≤ 255</td></tr>
</table></td></tr>
<tr><td>Timeout (ms)</td><td>Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. Value is expressed in milliseconds.</td></tr>
</table>

Element	Description
PLC Models	Selection of device models that may affect operation of the protocol. Currently only one model is available
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and enter IP Address for all controllers. 

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The error codes supported by this communication driver are:

Error	Notes
No response	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Incorrect node address in response	The panel did receive from the controller a response with invalid node address

Error	Notes
The received message too short	The panel did receive from the controller a response with invalid format
Incorrect writing data acknowledge	Controller did not accept write request; ensure the data programmed in the project are consistent with the controller resources

CT Modbus CMP ETH

The CT Modbus CMP ETH communication protocol is known also as “Modbus over CTNet”

CMP stands for CTNet Message Protocol; it is a messaging system designed to implement distributed control applications. The protocol permits exchange of parameters between Control Techniques drives and HMI devices, SCADA systems or other computer applications.

CMP is normally encapsulated in an existing network protocol. CMP has been successfully encapsulated also into the Modbus network. The communication protocol support implements the Modbus encapsulation of CMP.

Unidrive SP drives support the CTNet network using optional communication units called “SM Applications” modules.

Please note that changes in the communication protocol specifications or PLC hardware may have occurred since this documentation was created. Some changes may eventually affect the functionality of this communication driver. Always test and verify the functionality of your application. To fully support changes in PLC hardware and communication protocols, communication drivers are continuously updated. Always ensure that the latest version of communication driver is used in your application.

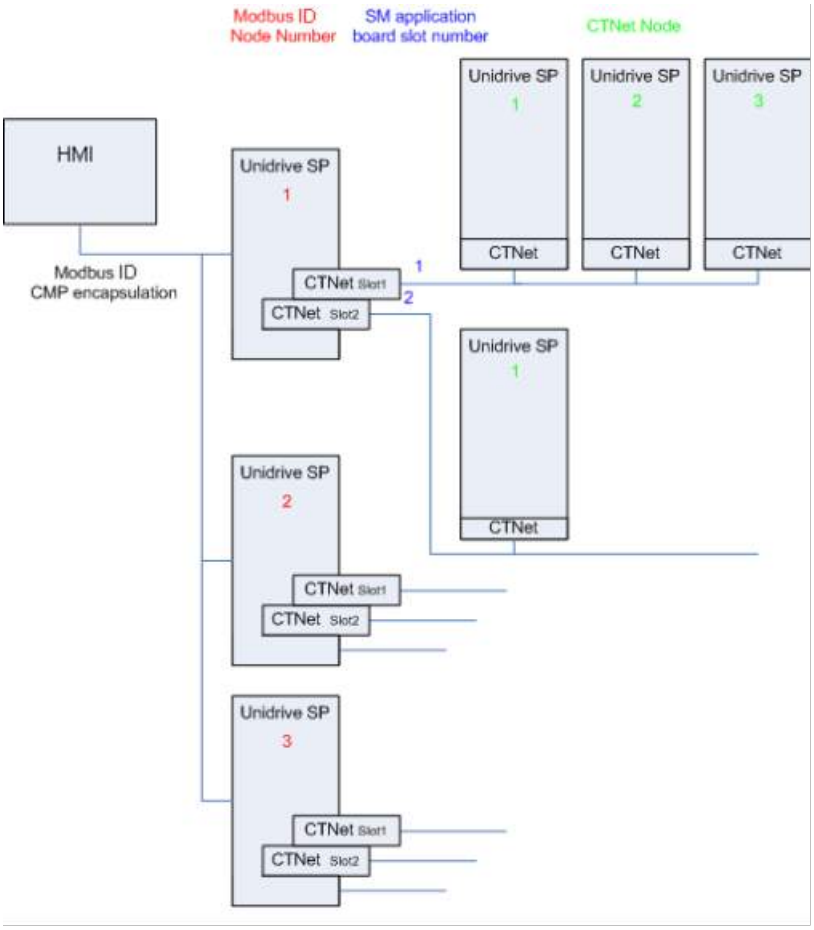
Concept of Operation

The network topology supported by the HMI communication protocol is shown in the figure below.

The HMI panel will communicate with a set of drives over the network; drives are addressed using their Modbus ID node number.

Each drive can host up to three SM application boards; they may be used for CTNet communication.

The addressing model is based on a three level space; from the HMI point of view, each drive is identified with a unique ID composed of a maximum of three numbers; the ID can be calculated looking to the network topology.



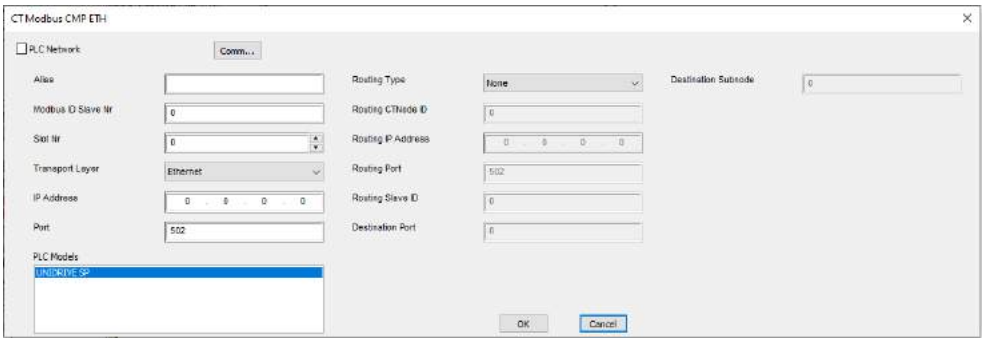
Protocol Editor Settings

Add (+) a driver in the Protocol Editor and select the protocol called “CT Modbus CMP ETH” from the list of available protocols.

The CT Modbus CMP ETH driver supports three different protocol types:

- none
- CTNet
- Ethernet

The protocol type can be selected from the “Routing Type” combo box in the dialog.

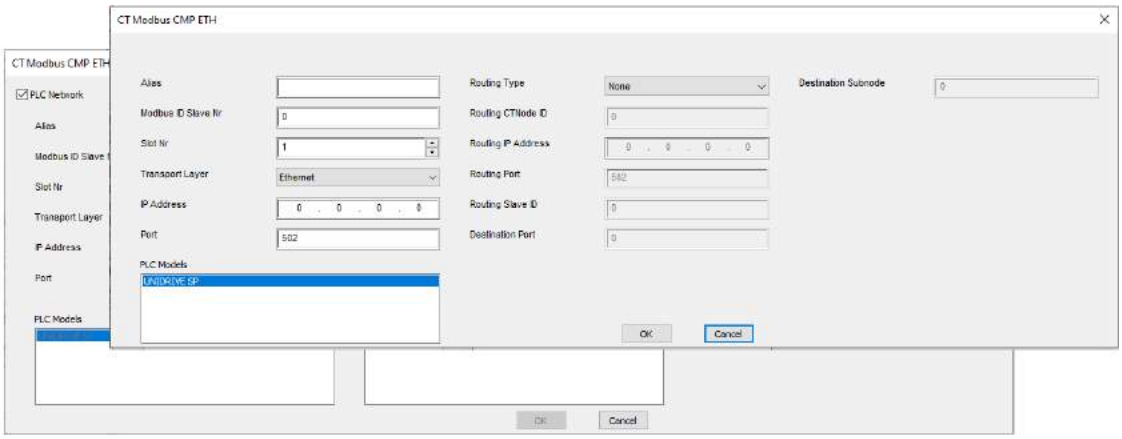


Some of the parameters of the dialog are common to all the protocol types, some others are specific.

The parameters common to all protocol types are the following:

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Modbus ID Slave Nr</td><td>Valid slave node addresses are 1 through 247. In the request from the protocol master this value indicates the target slave node.</td></tr>
<tr><td>Slot Nr</td><td>This value gives the option slot to which the message is directed. The port/slot mapping is as follows:<table><tr><th>Port</th><th>Message addressed to</th></tr><tr><td>1</td><td>Option in slot 1</td></tr><tr><td>2</td><td>Option in slot 2</td></tr><tr><td>3</td><td>Option in slot 3</td></tr></table></td></tr>
<tr><td>IP address</td><td>Ethernet IP address of the controller</td></tr>
<tr><td>Port</td><td>Allows to change the default port number used by the Modbus TCP driver; it could be useful whenever the communication goes through Routers or Internet gateways where the default port number is already in use. Default value for this parameter is 502.</td></tr>
<tr><td>Routing type</td><td>The FC64 encapsulated protocol includes extra destination fields to be used for message routing between nodes on different networks. The combination of CMP destination port and CMP destination subnode address or subnode addressing scheme fields, allows a Modbus TCP server to decide whether to process a received message or retransmit the message through another port onto a different communications network.
User can select one of the following routing methods:
None: means that the communication will be established directly to the drive. The message is directed to a drive or to an option in the drive, and there is no routing onto a subnetwork to be performed.
CTNet: users can enter CTNet node number which represents the drive in the subnetwork.
Ethernet: SM-Ethernet modules will provide the capability to reroute messages on Ethernet.</td></tr>
<tr><td>Routing CTNode ID
Routing IP Address
Routing Port
Routing Slave ID
Destination Port
Destination Subnode</td><td>When Ethernet routing method has been selected, you have to enter Routing IP Address, Routing port, Routing Slave ID, Destination Port and Destination Subnode of the drive you want to connect. For more information on routing, please check the drive user's manual or CT Modbus specification.</td></tr>
</table>

Element	Description
PLC Network	The protocol allows the connection of multiple drives to one operator panel. To set-up multiple connections, check “PLC Network” checkbox and enter parameters for each drive you want to connect.
PLC Model	Selection of device models that may affect operation of the protocol. Currently only one model is available



Configuring the Drives

This protocol only supports Ethernet connection.

The Unidrive SP does not have a built-in Ethernet interface. So SM-Ethernet modules are required. The Modbus ID must be set in each drive (parameter 00.37 or 11.23)

The “Reduce SP serial interface priority” parameter in the SM-Ethernet module should be set to “True” (parameter 15.37, 16.37 or 17.37 depending on which slot the SM-Ethernet module is found).

Addressing the Drives

The HMI will address the drives in different ways, depending on their position in the network.

In case the drive to be addressed is attached to the Modbus network and is the master of a CTNet network, it is sufficient to specify its Modbus address.

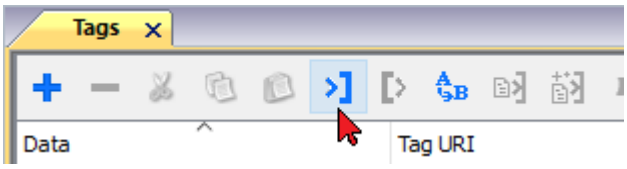
In case the drive is a CTNet slave, it will require an address depending on its logical position in the network. The 3-digit identifier is composed of the following elements:

- the first number is the Modbus Node ID of the drive master of the CTNet network
- the second number is the slot where the SM application card is plugged-in
- the third number is the CTNet node number of the drive.

When the drive master of the CTNet network has only one SM application unit, the slot information specified into the HMI project is not relevant. In fact, the communication protocol supports an automatic recognition of the slot number; this makes possible to move the SM application board to another slot, maintaining the same configuration at HMI project level.

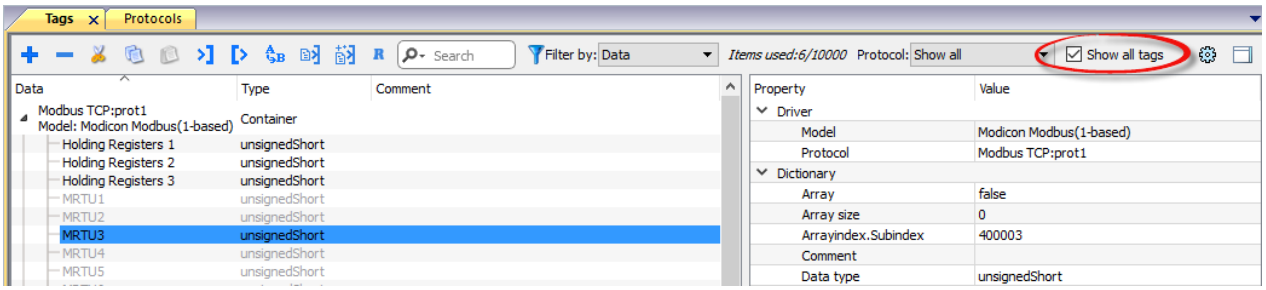
Tag Import

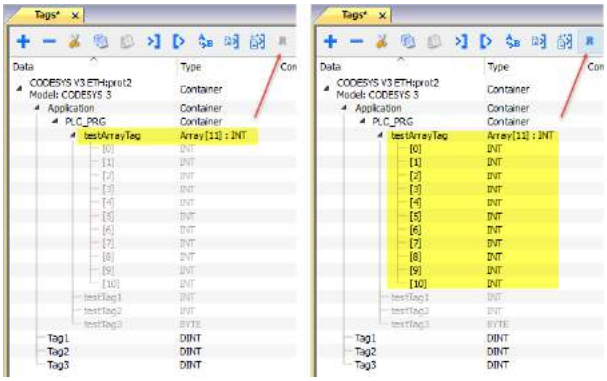
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



Locate the **.xml** file exported from Tag Editor and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the chapter “system variables” about available types and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Invalid response	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support

Delta Modbus RTU

Delta Modbus RTU communication driver has been designed to connect HMI devices to Delta PLC through Serial connection.

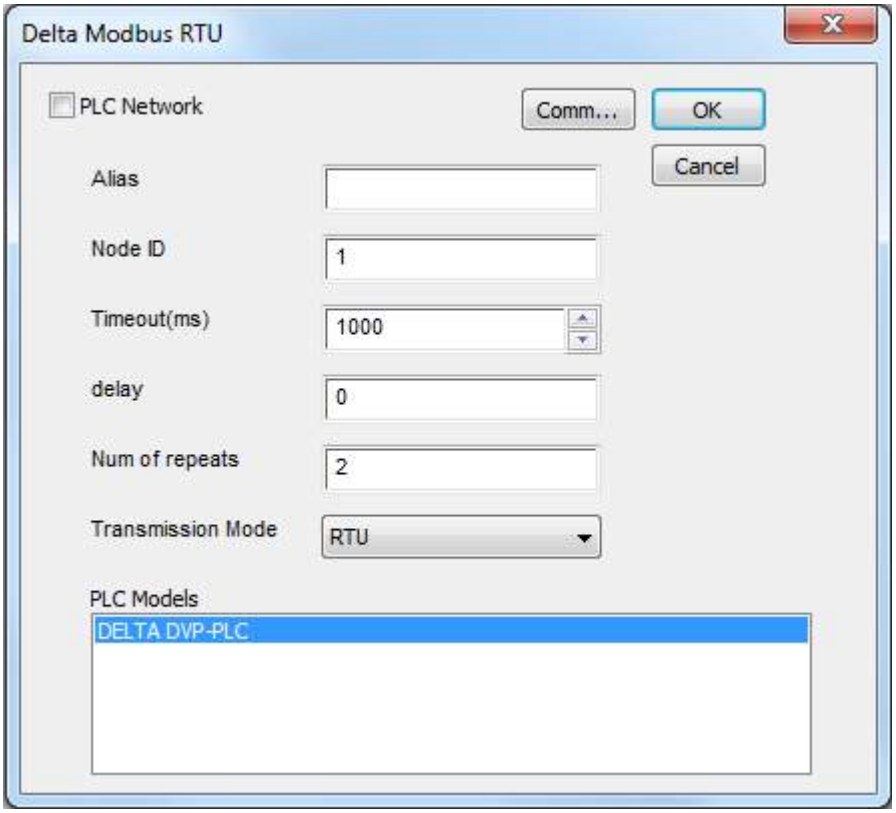
Protocol Editor Settings

Adding a protocol

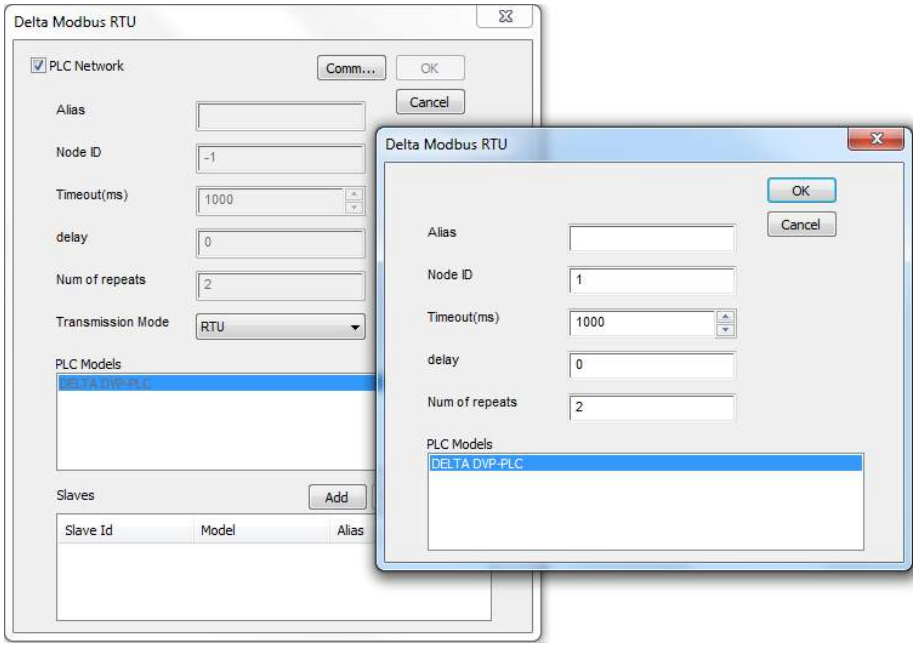
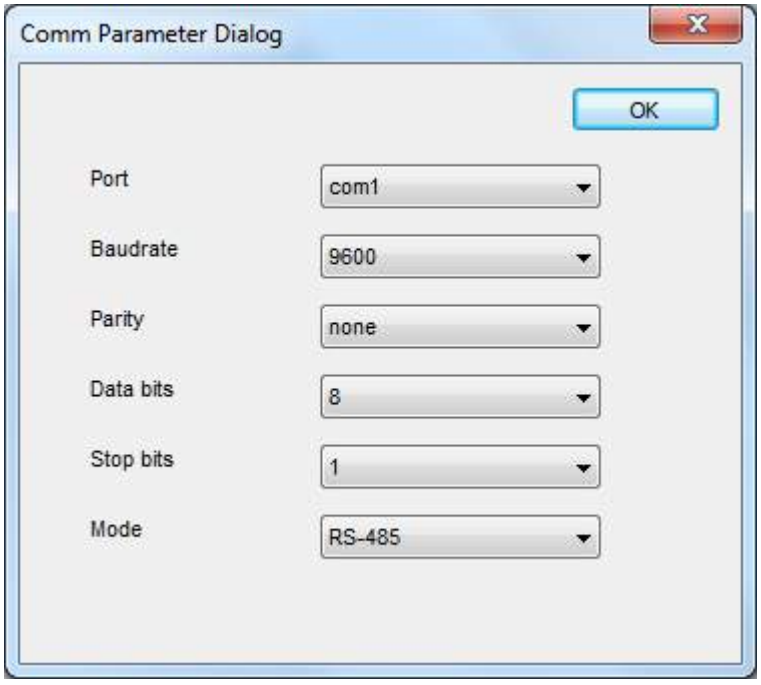
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Node ID	Serial node associated to the PLC.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
delay	Time delay in milliseconds between the end of the last received frame and the starting of a new request. If set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
Num of repeats	Number of times a certain message will be sent to the controller before reporting the communication error status. When set to 1 the panel will report the communication error if the response to the first request packet is not correct.
Transmission Mode	• **RTU**: use RTU mode • **ASCII**: use ASCII mode  Note: When PLC network is active, all nodes will be configured with the same Transmission Mode.
PLC Models	PLC model available: • DELTA DVP-PLC

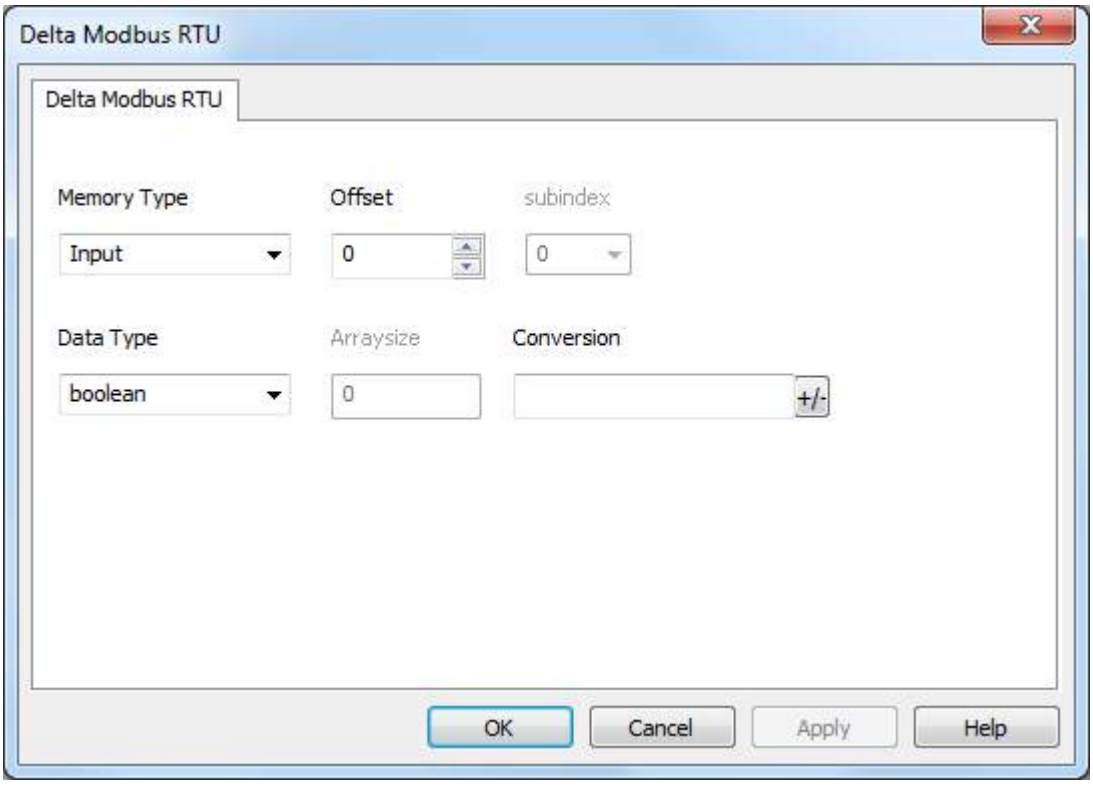
Element	Description
PLC Network	IP address for all controllers in multiple connections. **PLC Network** must be selected to enable multiple connections. 
Comm...	If clicked displays the communication parameters setup dialog. 

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Element</th><th>Parameter</th></tr>
<tr><td>Port</td><td>Serial port selection.
COM1: On-board port
COM2: Optional Plug-in module plugged on slot#1 or slot#2
COM3: Optional Plug-in module plugged on slot#3 or slot#4</td></tr>
<tr><td>Baudrate, Parity, Data Bits, Stop bits</td><td>Serial line parameters.</td></tr>
<tr><td>Mode</td><td>Serial port mode. Available modes:
RS-232.
RS-485 (2 wires).
RS-422 (4 wires).</td></tr>
</table>
</td></tr>
</table>

Tag Editor Settings

In Tag Editor select **Delta Modbus RTU** protocol.

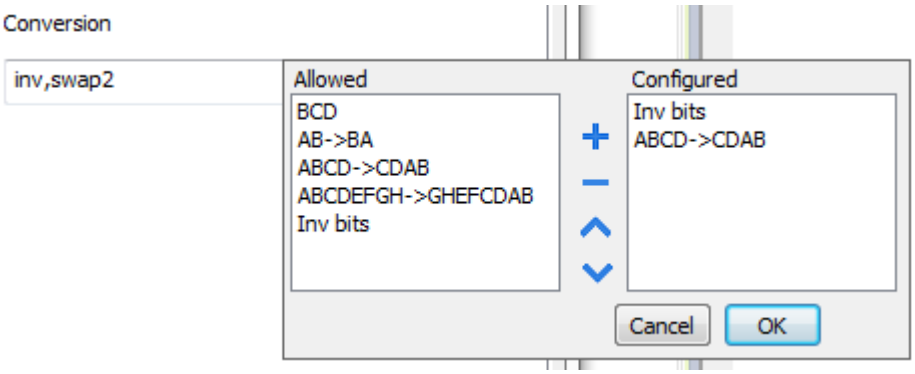
Add a tag using [+] button. Tag setting can be defined using the following dialog:



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Input</td><td>X resources. Corresponding to internal digital Input point.</td></tr>
<tr><td>Output</td><td>Y resources. Corresponding to internal digital Output point.</td></tr>
<tr><td>Auxiliary Relay</td><td>M resources. Corresponding to PLC internal memory.</td></tr>
<tr><td>Step Relay</td><td>S resources.</td></tr>
<tr><td>Timer Contact</td><td>T resources.</td></tr>
<tr><td>Counter Contact</td><td>C resources.</td></tr>
<tr><td>Timer Value</td><td>TV resources.</td></tr>
<tr><td>Counter Value</td><td>CV resources.</td></tr>
<tr><td>Counter 32bit Value</td><td>CV32 resources.</td></tr>
<tr><td>Data Register</td><td>D resources.</td></tr>
<tr><td>Node Override ID</td><td>see Special Data Types for mode details</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>Starting address for the Tag. The possible range depend on PLC model selected.</td></tr>
<tr><td>Subindex</td><td>This allows resource offset selection depending on the selected data type.</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating</td><td>1.17e-38 ...</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td></td><td>point type</td><td>3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>
Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Inv bits</td><td>inv: Invert all the bits of the tag.

Example:
1001 → 0110 (in binary format)
9 → 6 (in decimal format)</td></tr>
<tr><td>Negate</td><td>neg: Set the opposite of tag value.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>Example:
25.36 → -25.36</td></tr>
<tr><td>AB -> BA</td><td>swapnibbles: Swap nibbles in a byte.

Example:
15D4 → 514D (in hexadecimal format)
5588 → 20813 (in decimal format)</td></tr>
<tr><td>ABCD -> CDAB</td><td>swap2: Swap bytes in a word.

Example:
9ACC → CC9A (in hexadecimal format)
39628 → 52378 (in decimal format)</td></tr>
<tr><td>ABCDEFGH -> GHEFCDAB</td><td>swap4: Swap bytes in a double word.

Example:
32FCFF54 → 54FFFC32 (in hexadecimal format)
855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
0001110010111011011001000101101000011100101011000001
→
1 10000011100
1010101000010100010110110110110010110110000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

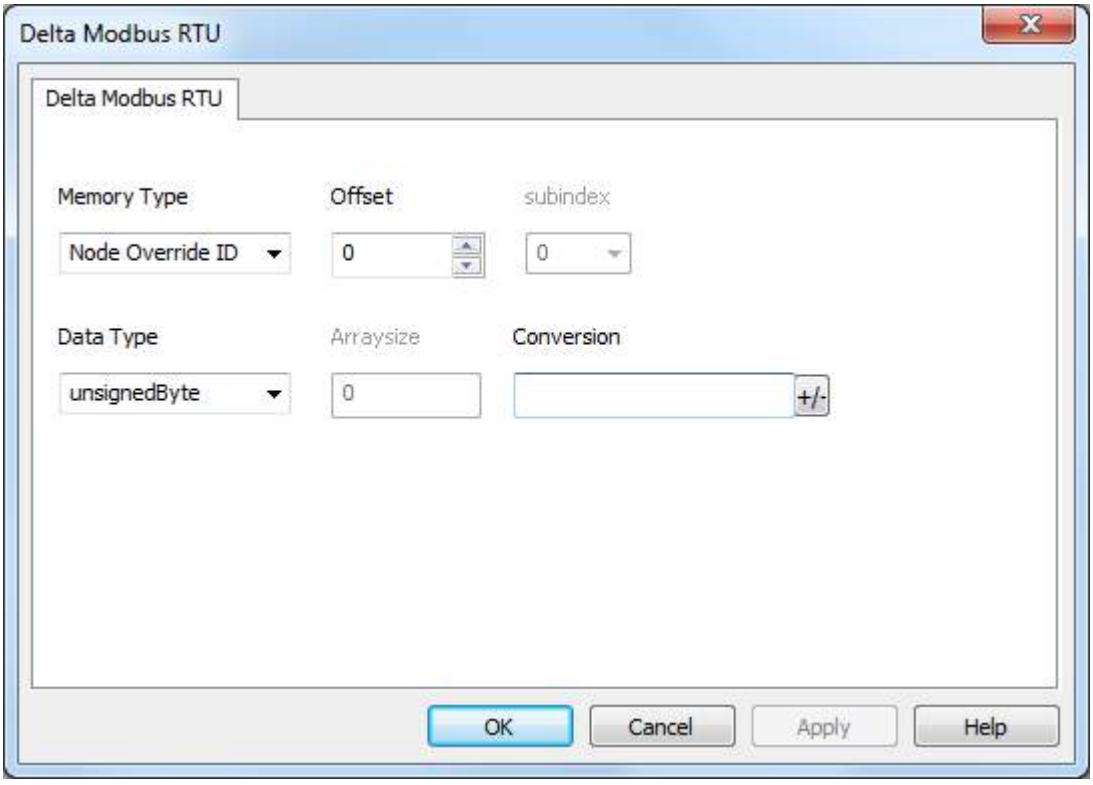
Node Override ID

The protocol provides the special data type Node Override ID which allows you to change the node ID of the slave at runtime. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

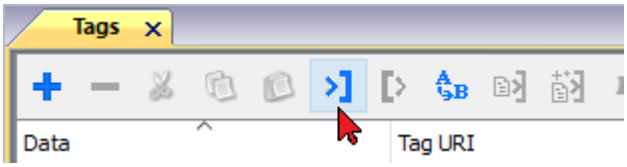
Node Override ID	Modbus operation
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.

Note: Node Override ID value assigned at runtime is retained through power cycles.

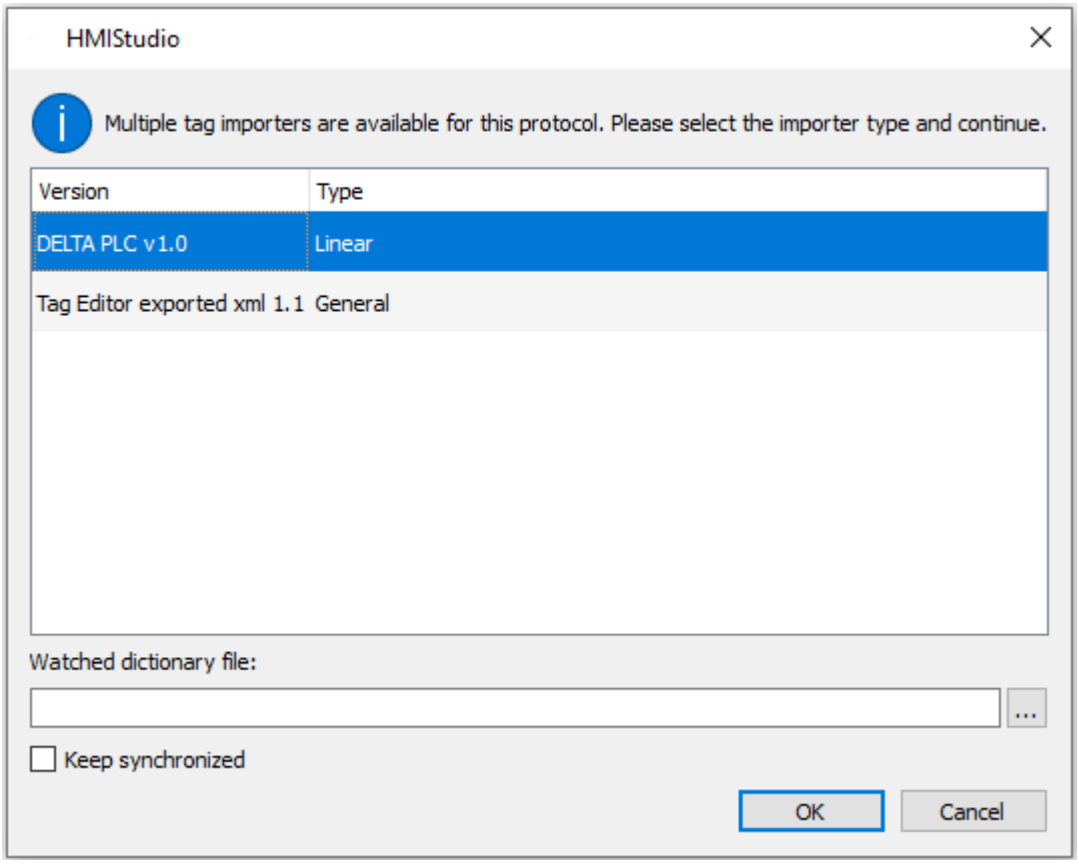


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



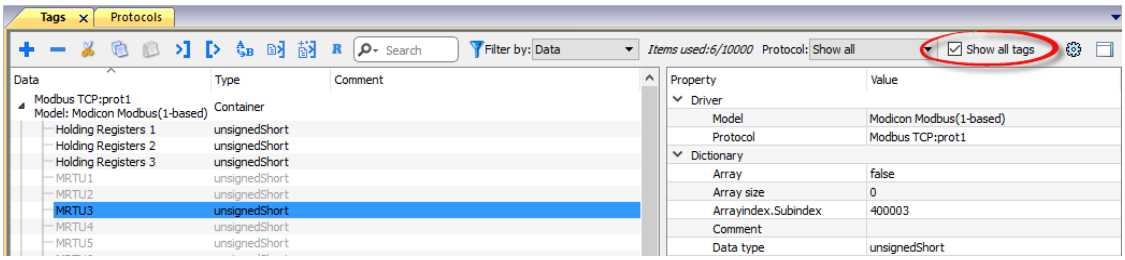
The following dialog shows which importer type can be selected.

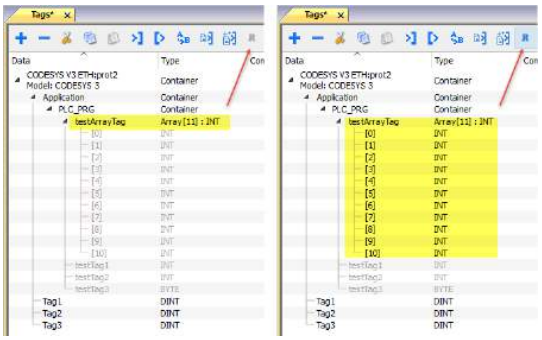
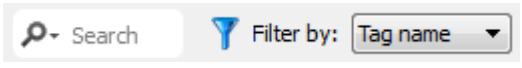


Type	Description
DELTA PLC v1.0 Linear	Requires a **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Direct Serial

Direct Serial communication driver is a generic protocol that allows low level access to serial functions.

Using this protocol the application itself can realize some serial based protocol (RS-232/485/422) without requirement for a development of a dedicated protocol.

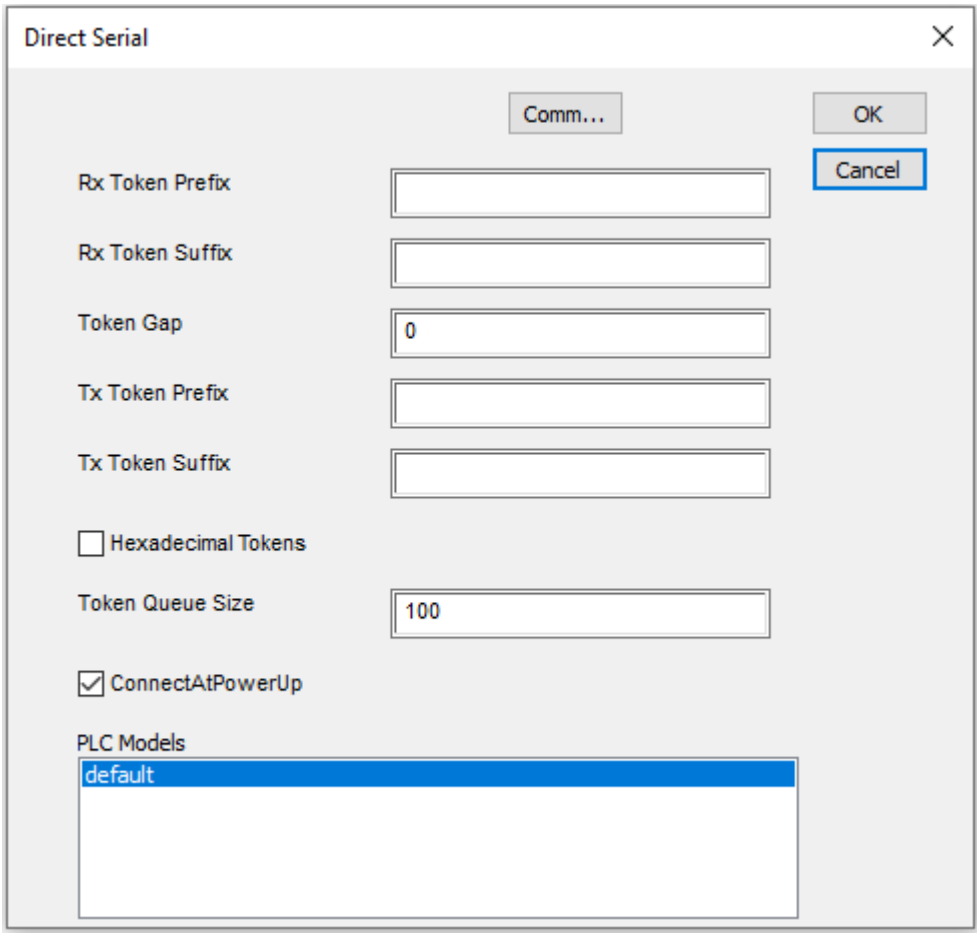
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Rx Token Prefix	Indicates the prefix for read token, as string specified by hexadecimal characters.
Rx Token Suffix	Indicates the suffix for read token, as string specified by hexadecimal characters.
Token Gap	Indicates the period between tokens, in milliseconds.
Tx Token Prefix	Indicates the prefix for sent token, as string specified by hexadecimal characters.
Tx Token Suffix	Indicates the suffix for sent token, as string specified by hexadecimal characters.
Hexadecimal Tokens	checked = tokens are in hexadecimal not checked = tokens are not in hexadecimal
Token Queue Size	Indicates the number of tokens in the queue, as an integer value from 1 to 10000 (default: 100)

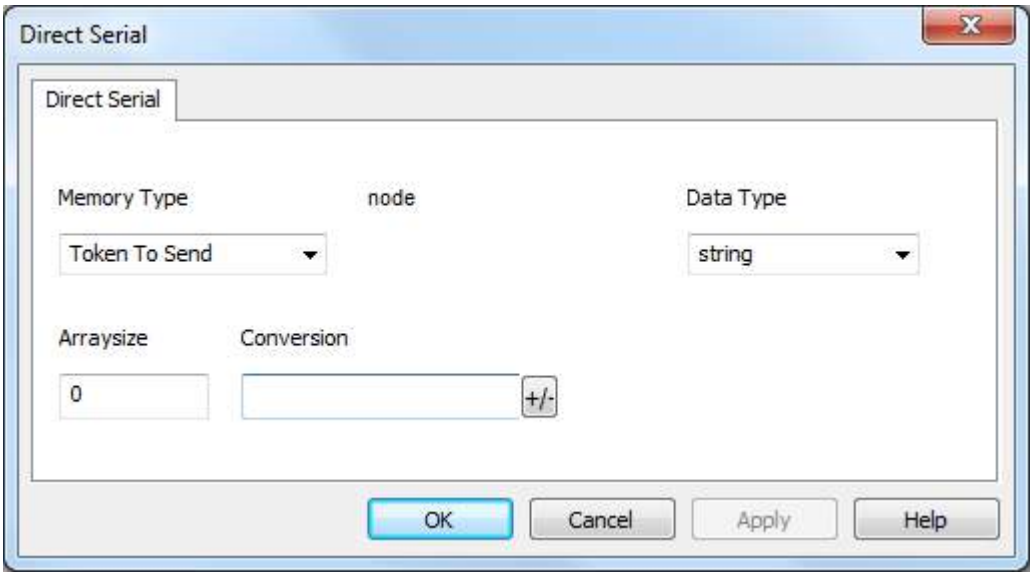
These parameters are determining the behavior of the driver during RX and TX operations, as defined in next paragraphs. In addition the standard communication parameters are available.

All protocols parameters can be overwritten at runtime using the appropriate memory types, so the complete setup can be achieved during runtime using Tags. Settings using memory types are saved to permanent storage using standard procedures. The “Serial Done” memory type is used in order that all set parameters are transferred to usage at once. If any of the serial parameter is changed the serial driver is re-programmed.

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Direct Serial** from the protocol list: tag definition dialog is displayed.

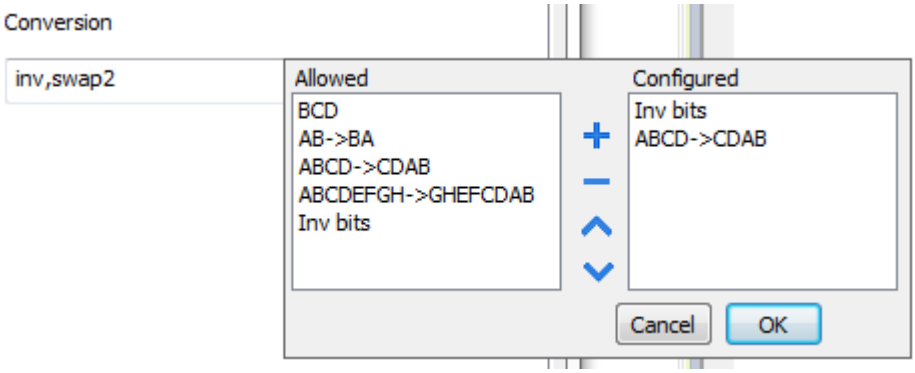


<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Name</th><th>Datatype</th><th>Description</th></tr>
<tr><td>Token To Send</td><td>string</td><td>Write only. Writing on this memory type sends the given string to communication.</td></tr>
<tr><td>Token Received</td><td>string</td><td>Read only. Reading from this memory type gets the front token from the receiving queue.</td></tr>
<tr><td>Length of Token Received</td><td>unsignedInt</td><td>Read only. Returns the length in bytes of the front token from the receiving queue.</td></tr>
<tr><td>Tokens Available</td><td>unsignedInt</td><td>Read only. Gives the number of tokens in the receiving queue.</td></tr>
<tr><td>Token Acknowledge</td><td>boolean</td><td>Write only. Writing to this memory type removes the front token from the receiving queue.</td></tr>
<tr><td>Serial Baudrate</td><td>unsignedInt</td><td>Overrides serial baudrate parameter.</td></tr>
<tr><td>Serial Bits</td><td>unsignedByte</td><td>Overrides serial bits parameter.</td></tr>
<tr><td>Serial Stop Bits</td><td>unsignedByte</td><td>Overrides serial stop bit parameter.</td></tr>
<tr><td>Serial Parity</td><td>unsignedByte</td><td>Overrides serial parity parameter.</td></tr>
<tr><td>Serial Mode</td><td>unsignedByte</td><td>Overrides serial mode parameter.</td></tr>
<tr><td>Rx Token Prefix</td><td>string</td><td rowspan="9">Overrides protocol parameters. Check "Protocol Editor Settings" from details.</td></tr>
<tr><td>Rx Token Suffix</td><td>string</td></tr>
<tr><td>Token Gap</td><td>unsignedInt</td></tr>
<tr><td>Tx Token Prefix</td><td>string</td></tr>
<tr><td>Tx Token Suffix</td><td>string</td></tr>
<tr><td>Hexadecimal Tokens</td><td>boolean</td></tr>
<tr><td>Token Queue Size</td><td>unsignedInt</td></tr>
<tr><td>Serial Done</td><td>boolean</td><td>Writing to this memory type transfers all new values written in the other tags to protocol parameters, and to permanent storage.</td></tr>
</table>
</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte []”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Inv bits</td><td>inv: Invert all the bits of the tag.

Example:
1001 → 0110 (in binary format)
9 → 6 (in decimal format)</td></tr>
<tr><td>Negate</td><td>neg: Set the opposite of tag value.

Example:
25.36 → -25.36</td></tr>
<tr><td>AB -> BA</td><td>swapnibbles: Swap nibbles in a byte.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>Example:
15D4 → 514D (in hexadecimal format)
5588 → 20813 (in decimal format)</td></tr>
<tr><td>ABCD -> CDAB</td><td>swap2: Swap bytes in a word.

Example:
9ACC → CC9A (in hexadecimal format)
39628 → 52378 (in decimal format)</td></tr>
<tr><td>ABCDEFGH -> GHEFCDAB</td><td>swap4: Swap bytes in a double word.

Example:
32FCFF54 → 54FFFC32 (in hexadecimal format)
855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000001
→
1 10000011100
1010101000010100010110110110110010110110000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

Implementation Details

Receiving algorithm

The protocol applies a separate thread that receives the characters from specified serial port.

When tokens (substrings) are identified they are put into the receiving queue (as strings).

Both ASCII and binary mode are available. When binary data can be present into receiving stream the **Hexadecimal Tokens** parameter can be set. In this case tokens are stored in queue using hex string coding (each byte is stored using two chars representing the hex value 0 to F). When defining the tags used to read tokens the appropriate string length should be computed considering the binary mode.

The **Token Queue Size** parameter specifies the maximum number of tokens saved into the queue. When the queue becomes full the oldest token is discarded.

The token identification is as follows:

- if the parameters specify a rx-prefix all characters before detecting the prefix are ignored
- if protocol specifies a rx-suffix it is used to detect the token end
- if rx-suffix is specified the parameter 'gap' specifies the timeout after which the token receiving is restarted
- if rx-suffix is not specified the parameter gap specifies the timeout that terminates the token (anything received up to this interval). If within this time the rx-prefix is detected again the token is ended and stored and reception of a new token is started

In summary we can have four combinations:

a. No rx-prefix and rx-suffix: the incoming stream is divided in tokens according to gap detection

b. Rx-prefix specified but no suffix: all the received chars before prefix are ignored. All the chars after prefix are stored in a token till the gap detection

c. Rx-prefix and Rx-suffix specified: all the chars between prefix and suffix are stored in a token. All the chars received before prefix or after suffix till the gap detection or till a new prefix are ignored

d. Rx-suffix specified but not RX-prefix: all the chars received till suffix are stored in a token. All the chars received after suffix till the gap detection are ignored

The rx-prefix and rx-suffix parameters are specified as hex strings, so any characters can be specified (like DLE STX CR LF etc...). i.e. to define the string “STR” as prefix the string “535452” must be used.

Before putting string to the receiving queue the prefix and suffix are removed (only 'payload' saved).

Transmission algorithm

The strings to be transmitted are prepared adding the “Tx-prefix” in front and the “Tx-suffix” in the end, if defined. Then the whole string is transmitted immediately.

Interface to user project

Reading a tag defined as **Token Received** gets the front string from the queue. If there are no new tokens an empty string is returned.

Reading a tag defined as **Length of Token Received** gets the length in bytes of the token.

Reading a tag defined as **Tokens Available** gets the number of tokens currently stored in the queue.

Writing to a tag defined as **Token Acknowledge** removes the token from queue and makes available the next token if present.

Writing to a tag defined as **Token To Send** means immediate sending, without any queue used.

JavaScript Interface

Beside Tag interface the user can access the protocol via JavaScript.

Although defined Tags can be accesses by JavaScript too, JavaScript can access directly to a Command interface implemented in protocol. This interface does not require the definition of Tags and is direct to protocol resulting in more efficiency.

This interface provides the access to token queue and sending function. The following commands are supported:

Command	Description
put	Put the token to send contained in string parameter.
get	Get the received token.
get_token_length	Get the length of received token.
tokens_available	Get number of tokens received.
token_ack	Acknowledge reading token.

Using the command interface the following JS code should receive data:

```
var tagMgr = project.getWidget("_TagMgr");
var protID = "prot2"; // to be set according to protocol numbering

var avail = tagMgr.invokeProtocolCommand(protID, "tokens_available", "");
while (parseInt(avail) > 0)
{
      var str = tagMgr.invokeProtocolCommand(protID, "get", ""); // get the next
token
      var status = tagMgr.invokeProtocolCommand(protID, "token_ack",""); //
acknowledge current token
      avail = tagMgr.invokeProtocolCommand(protID, "tokens_available",""); // get
number of available tokens in queue
}
```

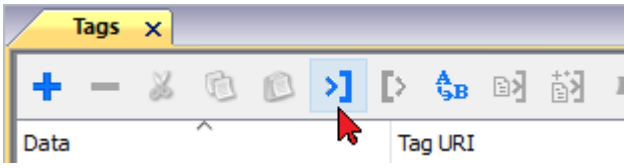
VCS access

The protocol supports the remote (virtual com port) access in exclusive mode.

When VCS is enabled the serial line usage is suspended and serial line becomes available for remote user. At the end the protocol is restarted. The content of the token queue is lost.

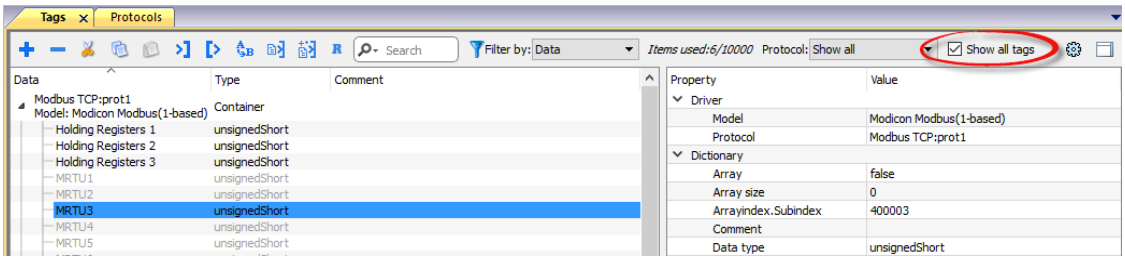
Tag Import

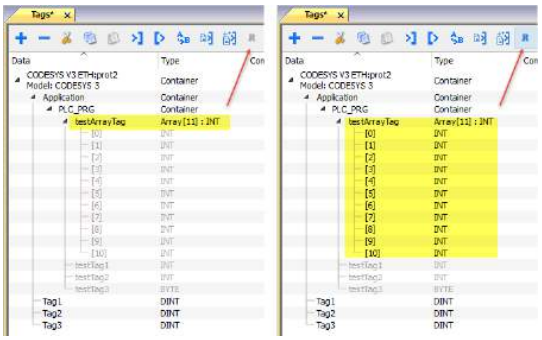
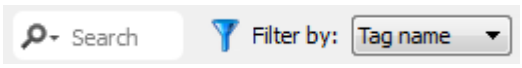
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



Locate the Tag Editor Exported symbol file and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Direct Socket

Direct Socket protocol is a generic protocol that allows low level access to socket functions.

Using this protocol the application itself can realize some IP based protocol without requirement for a development of a dedicated protocol.

Direct Socket protocol can be used as a standard (tag interface) protocol but also there is the appropriate implementation of DoCommand interface to enable using protocol from JavaScript.

The protocol can be used only with client socket type.

The protocol supports just one client socket. In case that application requires many sockets there could be many protocols installed, as the protocol supports multi-instance.

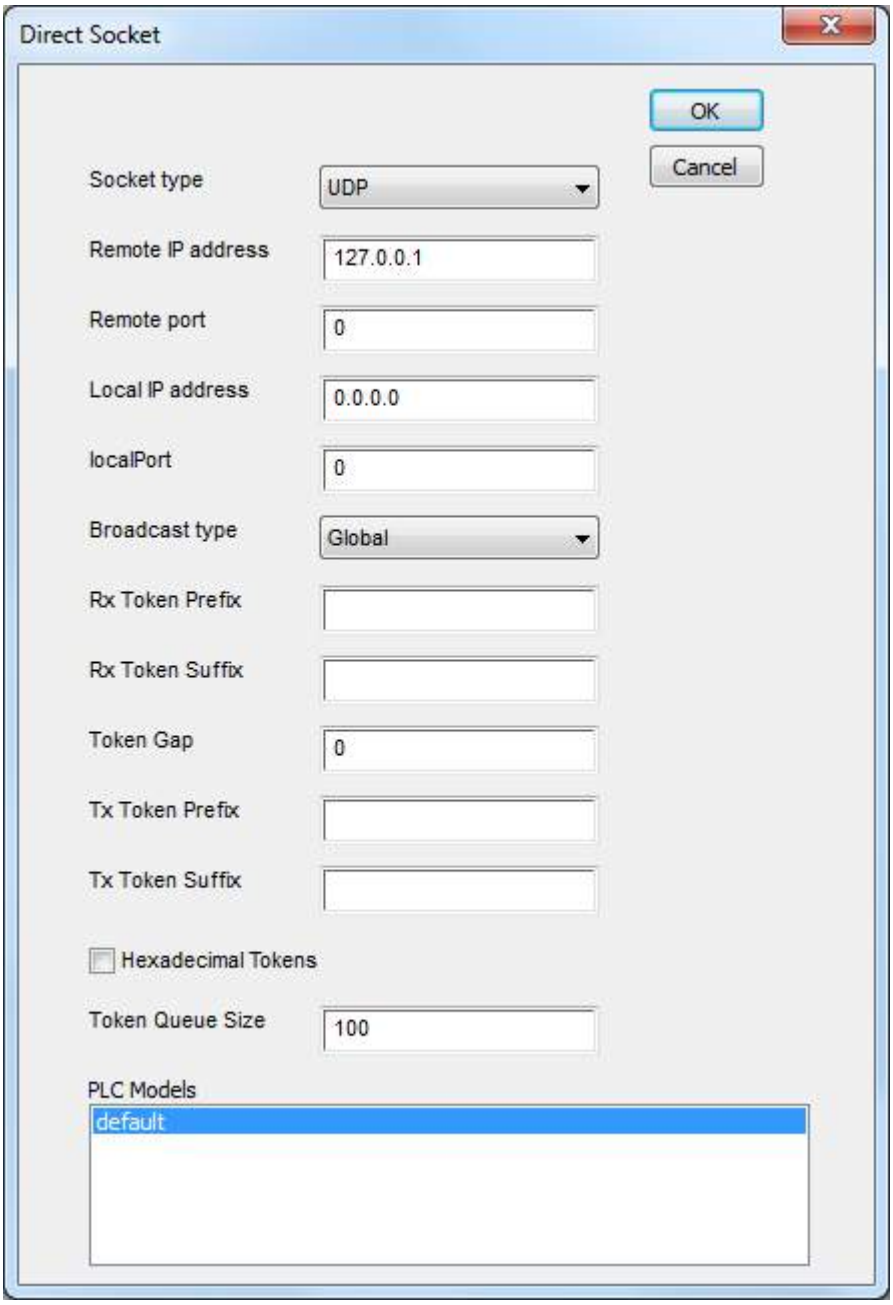
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Protocol parameters define a way how the connection is set and how the tokens are exchanged. The parameters are generally defined by the project. Many parameters can be accessed also as variables, allowing the runtime changes.

Element	Description
Socket type	Type of socket used for communication. Possible choices are UDP or TCP.
Remote IP Address	String. Indicates the IP address of remote device.
Remote Port	Integer. Indicates the port used by remote device.
Local IP Address	String. Indicates the IP address of local device. Mandatory for UDP usage.

Element	Description
Local Port	Integer. Indicates the port used by local device. Mandatory for UDP usage.
Broadcast Type	Type of broadcast used. Possible choices are Global or Local.

The following parameters are determining the behavior of the driver during RX and TX operations, as defined *Implementation Details* chapter.

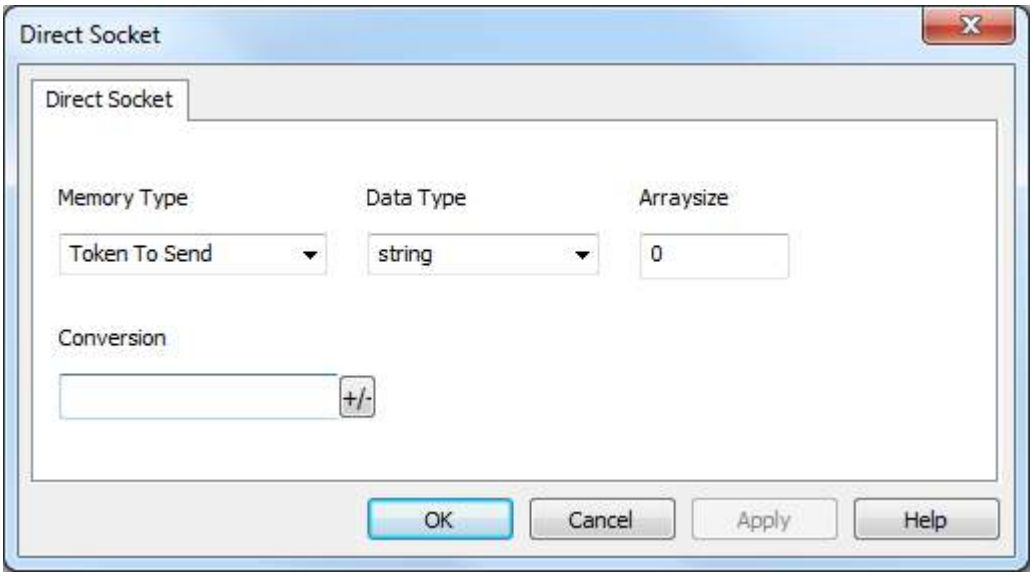
Element	Description
Rx Token Prefix	Indicates the prefix for read token, as string specified by hexadecimal characters.
Rx Token Suffix	Indicates the suffix for read token, as string specified by hexadecimal characters.
Token Gap	Indicates the period between tokens, in milliseconds.
Tx Token Prefix	Indicates the prefix for sent token, as string specified by hexadecimal characters.
Tx Token Suffix	Indicates the suffix for sent token, as string specified by hexadecimal characters.
Hexadecimal Tokens	checked = tokens are in hexadecimal not checked = tokens are not in hexadecimal
Token Queue Size	Indicates the number of tokens in the queue, as an integer value from 1 to 10000 (default: 100)

All protocols parameters can be overwritten at runtime using the appropriate memory types, so the complete setup can be achieved during runtime using Tags. Settings using memory types are saved to permanent storage using standard procedures. The “Done” memory type is used in order that all set parameters are transferred to usage at once. If any parameter is changed the driver is re-programmed.

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

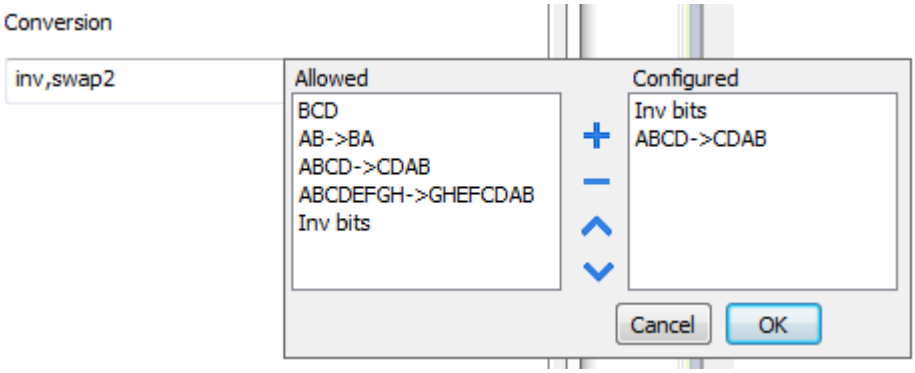
1. To add a tag, click **+**: a new line is added.
2. Select **Direct Socket** from the protocol list: tag definition dialog is displayed.


Direct Socket
Direct Socket
Memory Type
Data Type
Arraysize
Token To Send
string
0
Conversion
+/-
OK
Cancel
Apply
Help

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Name</th><th>Datatype</th><th>Description</th></tr>
<tr><td>Token To Send</td><td>string</td><td>Write only. Writing on this memory type sends the given string to communication.</td></tr>
<tr><td>Token Received</td><td>string</td><td>Read only. Reading from this memory type gets the front token from the receiving queue.</td></tr>
<tr><td>Length of Token Received</td><td>unsignedInt</td><td>Read only. Returns the length in bytes of the front token from the receiving queue.</td></tr>
<tr><td>Tokens Available</td><td>unsignedInt</td><td>Read only. Gives the number of tokens in the receiving queue.</td></tr>
<tr><td>Token Acknowledge</td><td>boolean</td><td>Write only. Writing to this memory type removes the front token from the receiving queue.</td></tr>
<tr><td>Connect</td><td>boolean</td><td>Write only. Writing 1 to this variable enables the connection.</td></tr>
<tr><td>Connection Status</td><td>boolean</td><td>Read only. Gives the status of the connection In TCP mode it reflects effective connection with the peer. In UDP mode it is TRUE as soon as Connect is TRUE</td></tr>
<tr><td>Socket type</td><td>string</td><td rowspan="14">Overrides protocol parameters. Check "Protocol Editor Settings" from details.</td></tr>
<tr><td>Remote IP Address</td><td>string</td></tr>
<tr><td>Remote Port</td><td>unsignedShort</td></tr>
<tr><td>Local IP Address</td><td>strgin</td></tr>
<tr><td>Local Port</td><td>unsignedShort</td></tr>
<tr><td>Broadcast Type</td><td>string</td></tr>
<tr><td>Rx Token Prefix</td><td>string</td></tr>
<tr><td>Rx Token Suffix</td><td>string</td></tr>
<tr><td>Token Gap</td><td>unsignedInt</td></tr>
<tr><td>Tx Token Prefix</td><td>string</td></tr>
<tr><td>Tx Token Suffix</td><td>string</td></tr>
<tr><td>Hexadecimal Tokens</td><td>boolean</td></tr>
<tr><td>Token Queue Size</td><td>unsignedInt</td></tr>
<tr><td>Done</td><td>boolean</td><td>Writing to a tag of this memory type transfers all new values written in the other tags to protocol parameters, and to permanent storage.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
Note: to define arrays. select one of Data Type format followed by square brackets like “byte []”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

Element	Description
	If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Implementation Details

Principle of operation

Protocol is parameterized by number of protocols parameters. The parameters define which socket type is used and the host address.

The data access is based on ‘tokens’. Token is data string that can be surrounded by prefix and suffix.

The protocol receiving process reads data from the specified IP/port and identifies tokens. Identified tokens are put to the queue from where they can be read by application. In the sending direction the application writes the token to protocol.

Protocol adds the defined tx_prefix/tx_suffix and sends data to the defined host.

Token extraction

The token extraction is slightly different for UDP and TCP sockets.

UDP protocols starts searching for tokens at the start of the received datagram. The search ends at the datagram end. If no rx_prefix is specified the token starts at datagram start. If no rx_suffix is specified the token ends on the datagram end. By specifying neither prefix nor suffix the whole datagram is delivered as a token. When both prefix and suffix are specified there can be many tokens extracted from a single datagram.

TCP protocol starts searching for tokens immediately after the previous rx_prefix. The search ends either when suffix is found or if the time gap without data is detected. If neither prefix nor suffix is specified the tokens will be all received data separated by time gaps.

The tokens can be plain ASCII strings, or hexadecimal strings. This is defined by the parameter ‘hex_tokens’.

The prefix/suffix strings must always be in hexadecimal format.

Common behavior

Both ASCII and binary mode are available. When binary data can be present into receiving stream the **Hexadecimal Tokens** parameter can be set. In this case tokens are stored in queue using hex string coding (each byte is stored using two chars representing the hex value 0 to F). When defining the tags used to read tokens the appropriate string length should be computed considering the binary mode.

The **Token Queue Size** parameter specifies the maximum number of tokens saved into the queue. When the queue becomes full the oldest token is discarded.

The token identification is as follows:

- if the parameters specify a rx-prefix all characters before detecting the prefix are ignored
- if protocol specifies a rx-suffix it is used to detect the token end
- if rx-suffix is specified the parameter 'gap' specifies the timeout after which the token receiving is restarted

- if rx-suffix is not specified the parameter gap specifies the timeout that terminates the token (anything received up to this interval). If within this time the rx-prefix is detected again the token is ended and stored and reception of a new token is started

In summary we can have four combinations:

a. No rx-prefix and rx-suffix: the incoming stream is divided in tokens according to gap detection
b. Rx-prefix specified but no suffix: all the received chars before prefix are ignored. All the chars after prefix are stored in a token till the gap detection
c. Rx-prefix and Rx-suffix specified: all the chars between prefix and suffix are stored in a token. All the chars received before prefix or after suffix till the gap detection or till a new prefix are ignored
d. Rx-suffix specified but not RX-prefix: all the chars received till suffix are stored in a token. All the chars received after suffix till the gap detection are ignored

The rx-prefix and rx-suffix parameters are specified as hex strings, so any characters can be specified (like DLE STX CR LF etc...). i.e. to define the string “STR” as prefix the string “535452” must be used

Before putting string to the receiving queue the prefix and suffix are removed (only 'payload' saved).

Interface to user project

Reading a tag defined as **Token Received** gets the front string from the queue. If there are no new tokens an empty string is returned.

Reading a tag defined as **Length of Token Received** gets the length in bytes of the token.

Reading a tag defined as **Tokens Available** gets the number of tokens currently stored in the queue.

Writing to a tag defined as **Token Acknowledge** removes the token from queue and makes available the next token if present.

Writing to a tag defined as **Token To Send** means immediate sending, without any queue used.

Data traffic control

The TCP sockets can be controlled by variables “Connect” and “Connection Status”. If the bool variable “Connect” is set the protocol will permanently try to make the connection to the specified host. If the TCP connection breaks it will be re-established automatically. If the variable “Connect” is false the protocol will wait. The state of connection can be read by variable Connection Status”.

For UDP there is no connection control. The socket is always connected and sends/receives data.

JavaScript Interface

Beside Tag interface the user can access the protocol via JavaScript.

Although defined Tags can be accesses by JavaScript too, JavaScript can access directly to a Command interface implemented in protocol. This interface does not require the definition of Tags and is direct to protocol resulting in more efficiency.

This interface provides the access to token queue and sending function. The following commands are supported:

Command	Description

set_ip_port <IPaddress> <port>	Specify the remote IP/port couple to use for connection.
	If protocol is already connected it is disconnected from current peer and re-connected to new one.
	Example of usage in JavaScript:
	`var tagMgr = project.getWidget("_TagMgr");`
	`var protID = "prot2"; // to be set according to protocol numbering`
	`tagMgr.invokeProtocolCommand(ProtID,"set_ip_port","127.0.0.1 502");`

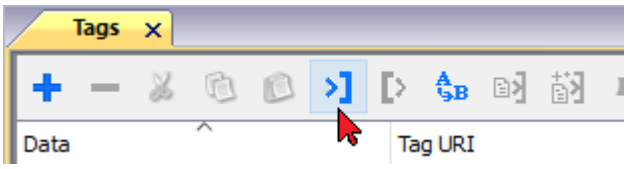
connect <ON|OFF>	Enables/disables the connection.
get_stat	Status of connection <connected|disconnected>.
put <string>	Put the token to send contained in string parameter.
get	Get the received token.
get_token_length	Get the length of received token.
tokens_available	Get number of tokens received.
token_ack	Acknowledge reading token.

Using the command interface the following JS code should receive data:

```
var tagMgr = project.getWidget("_TagMgr");
var protID = "prot2"; // to be set according to protocol numbering

var avail = tagMgr.invokeProtocolCommand(protID, "tokens_available", "");
while (parseInt(avail) > 0)
{
      var str = tagMgr.invokeProtocolCommand(protID, "get", ""); // get the next token
      var status = tagMgr.invokeProtocolCommand(protID, "token_ack",""); // acknowledge current token
      avail = tagMgr.invokeProtocolCommand(protID, "tokens_available",""); // get number of available tokens in queue
}
```

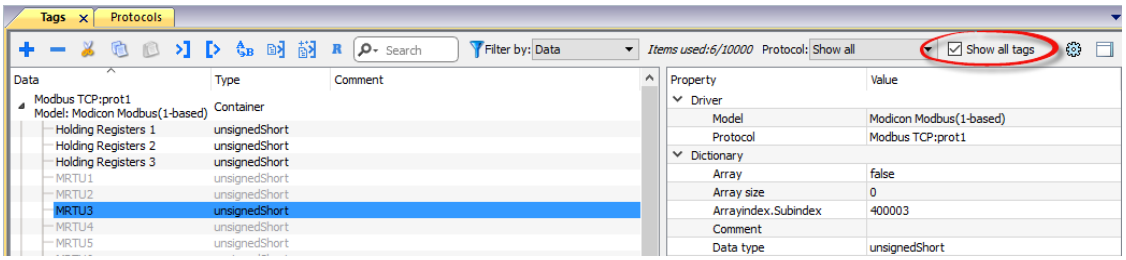
Tag Import

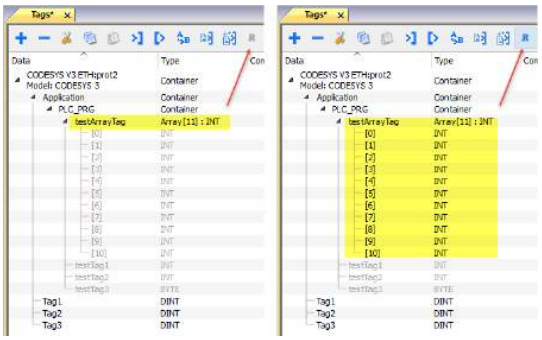
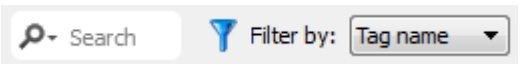
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



Locate the Tag Editor Exported symbol file and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

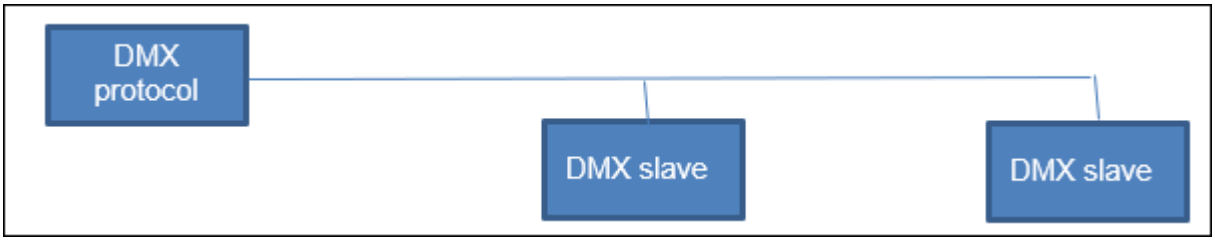
DMX512 Digital Multiplex

This document describes and specifies the implementation of **DMX512 Digital Multiplex** communication driver.

Purpose of implementation is to allow driving up to 512 channels connected to a RS485 serial line, or to merge additional channels, or to overwrite existing channels to an existing DMX controller.

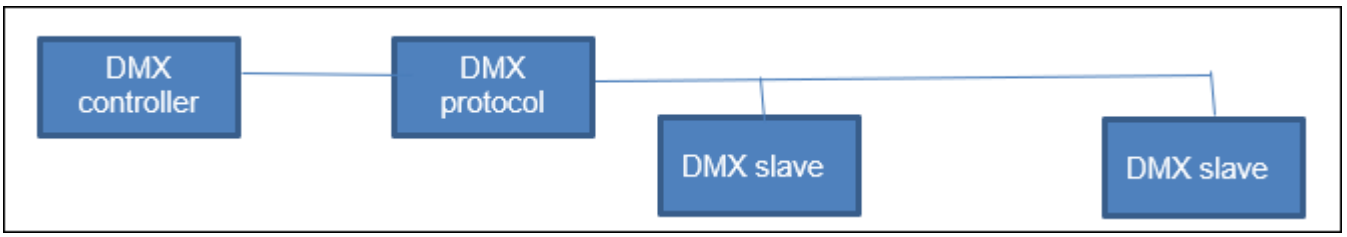
Possible topologies

Normal mode



In normal mode only Tx signal of the serial line is connected.

Merge mode



In merge mode the existing serial line must be opened and the origin line must be connected to Rx input.

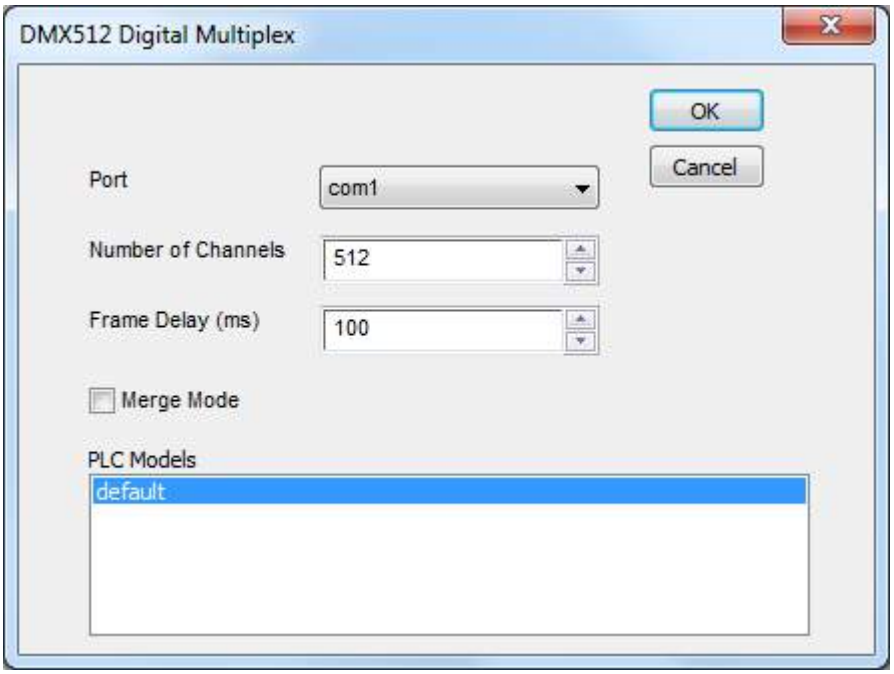
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.

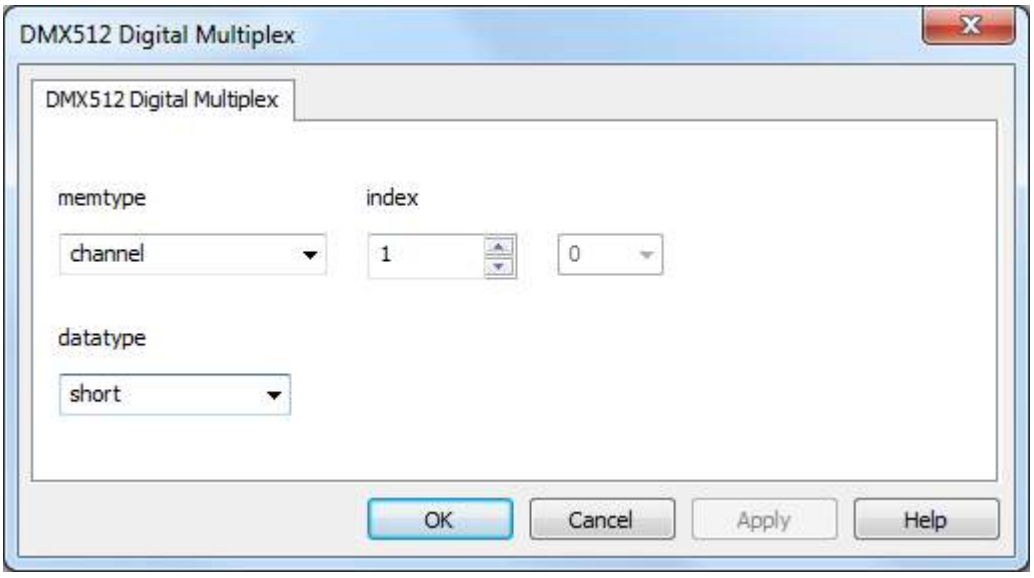


Element	Description
Port	COM port to be used. Serial line parameters are fixed.
Number of Channels	1 - 512. Defines the number of channels transmitted in the multiplex flame.
Frame Delay (ms)	10 - 1000. Defines inter-frame delay to adapt to specifications of slaves. Delay is applied at the end of frame so the real frame rate is determined by formula: (approx) *Time (microsec) = 120 + 20 + 40 x (nr of channels) + Frame Delay * 1000*
Merge Mode	Selects the Merge Mode in which the unit receives a frame from an external controller and substitutes the values of some of the channels or add other channels in the end of the frame
PLC Models	Only "default" is available.

Tag Editor Settings

In Tag Editor select **DMX512 Digital Multiplex** protocol.

Add a tag using [+] button. Tag setting can be defined using the following dialog:



Each channel can be assigned to a Tag.

Element	Description		
memtype	**Memory Type**	**Description**	
	channel	Only available memory type.	
index	Refer to channel number to point to.		
datatype	**Data Type**	**Memory Space**	**Limits**
	short	16-bit data	-32768 ... 32767

Channel behavior

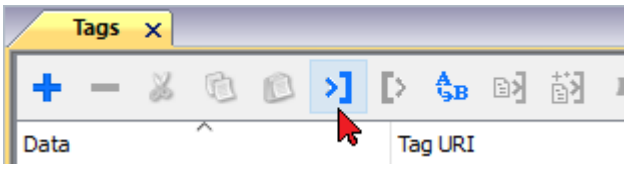
Only available DataType is short (signed 16-bit data) so a Tag can assume values from -32768 to 32767. Anyway the protocol uses only values from 0 to 255.

Other values are used in Merge Mode: when the channel overwrites an existing channel the negative values are used to disable overwriting.

Value	Normal Mode	Merge Mode
0 to 255	0 to 255	0 to 255
> 255	255	255
< 0	0	original value of channel in the incoming frame

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.

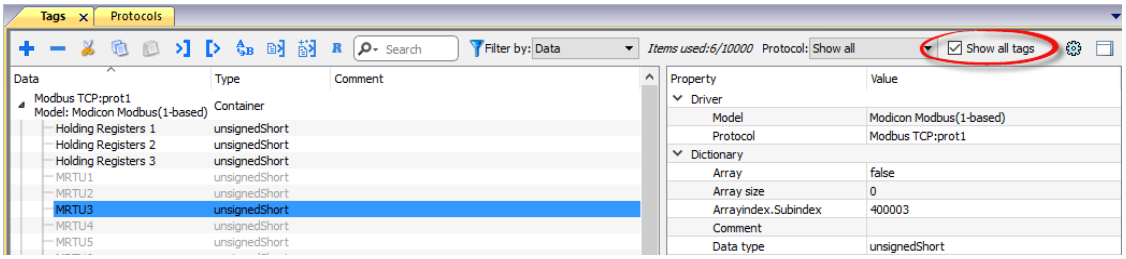


It is possible to import a Tag Editor exported xml

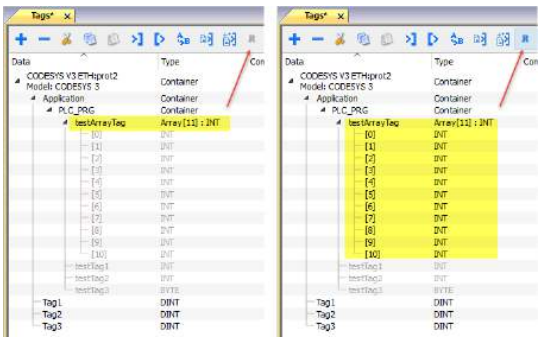
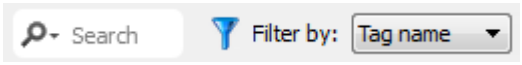
Type	Description
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



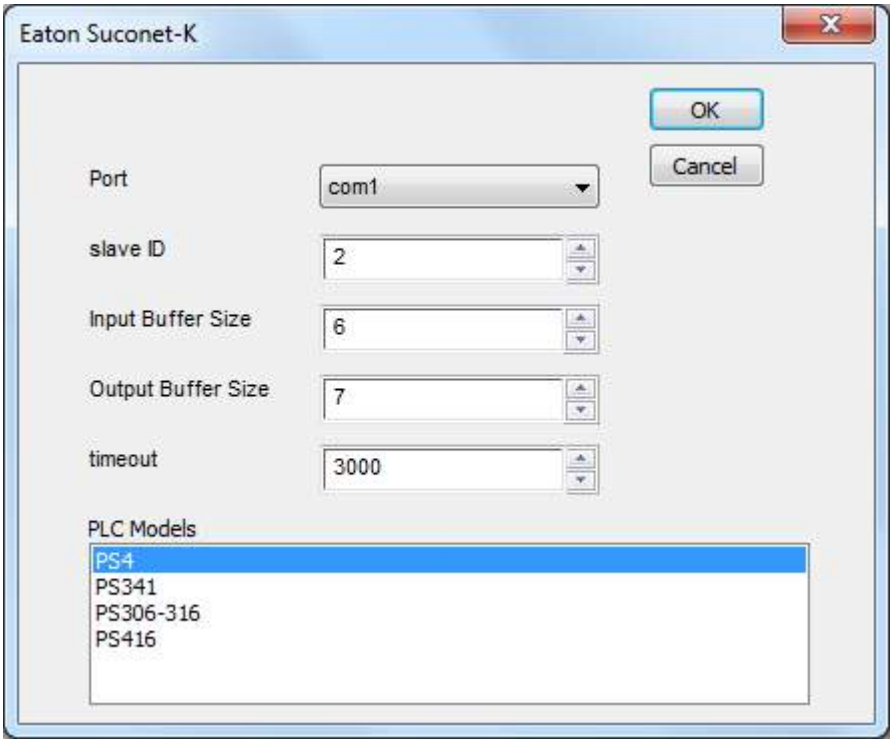
Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

Eaton Suconet-K

The Eaton Suconet-K communication driver has been designed to connect HMI devices to a Suconet-K network with a Möeller PLC.

Protocol Editor Settings



Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

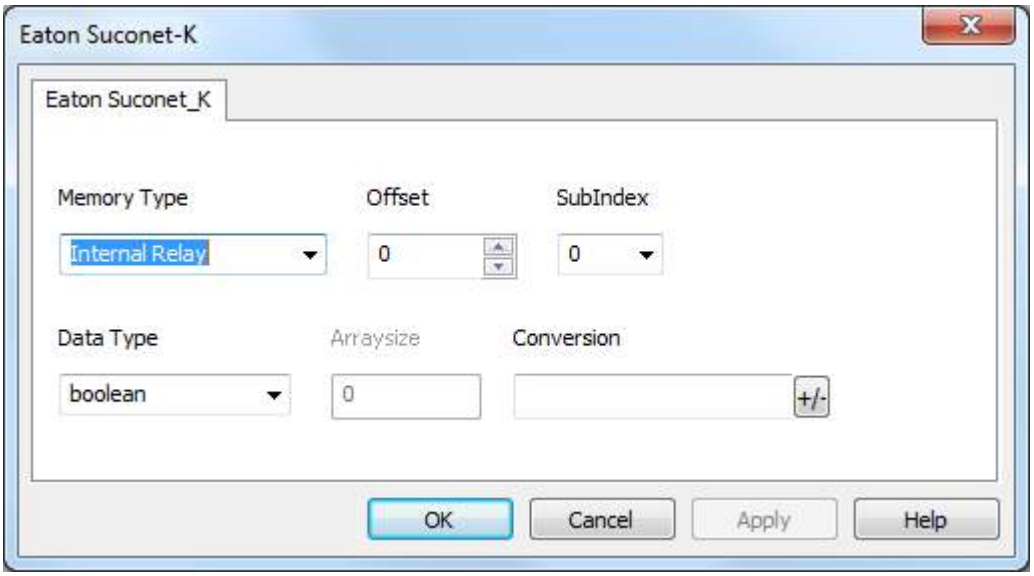
The protocol configuration dialog is displayed.

Element	Description
Port	Serial port selection. • **COM1**: device PLC port. • **COM2**: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board. • **COM3:** optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.
slave ID	node of the slave device.
Input Buffer Size	Size of Input Buffer. Input data length must be exactly the same as in PLC configuration.
Output Buffer Size	Size of Output Buffer. Output data length must be exactly the same as in PLC configuration.
timeout	Time delay in milliseconds between two retries in case of missing response from the server device.
PLC Models	Two PLC models are available: • PS4 • PS341 • PS306-316 • PS416

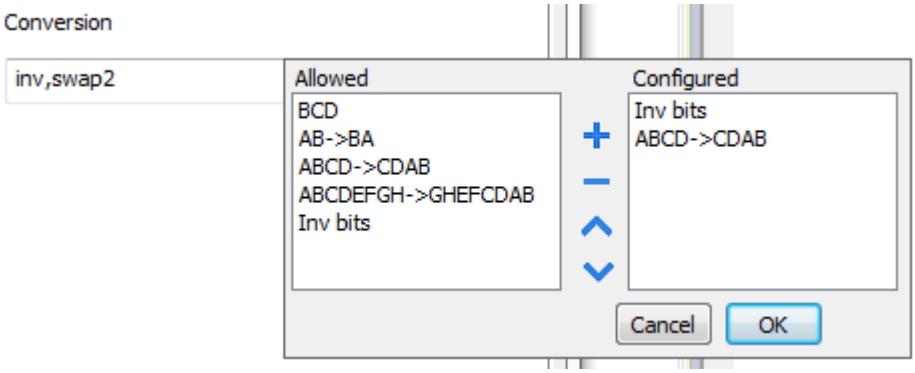
Tag Editor Settings

In Tag Editor select the protocol **Eaton Suconet-K**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Internal relay</td><td>Internal memory of PLC. It can be addressed using Offset and Data Type.</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>Starting address for the Tag. The possible range depend on PLC model selected.</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection depending on the selected data type.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
double
string
binary
See "Programming concepts" section in the main manual.
Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).</td></tr>
</table>

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH	**swap4**: Swap bytes in a double word.

Element	Description

Value	Description
-> GHEFCDAB	*Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 1 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Environment Variables

This protocol gives the possibility to copy the environment variables of the hosting Operative System inside tags. All variables will be read only, namely, is not possible to modify them.

Environment Variables communication driver is not counted as physical protocol.
Refer to **Table of functions and limits** from main manual in "Number of physical protocols" line.

Protocol Editor Settings

Adding a protocol

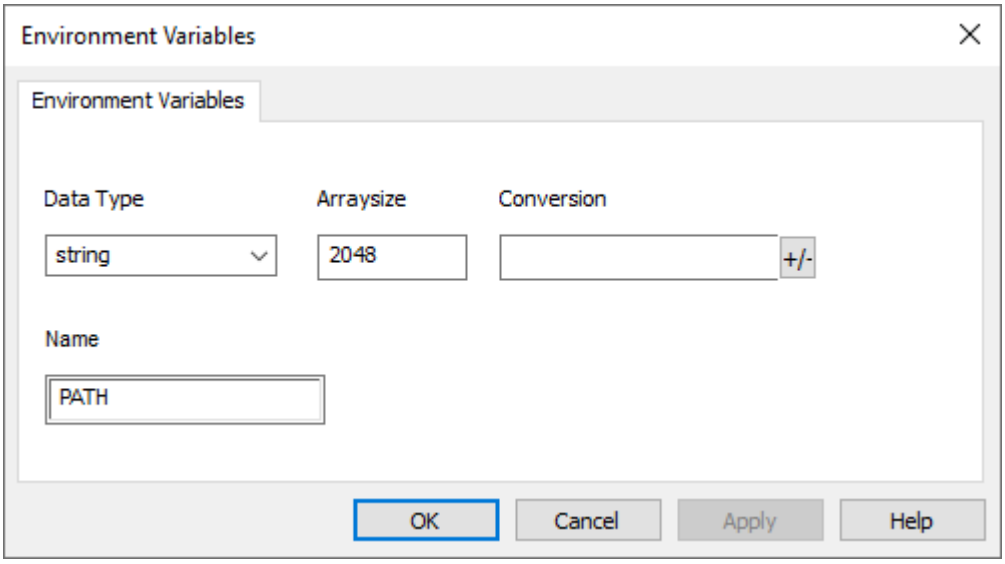
To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the **Environment Variables** protocol from the **PLC** list.

Tag Editor Settings

*Path: **ProjectView**> **Config** > double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Environment Variables** from the protocol list: tag definition dialog is displayed.



Element	Description
Name	Name of the environment variable that you want to read.
Data Type	System variables are of type string, but if a different type is chosen, e.g. int, casting to the chosen type will be made.
Arraysize	This property represents the maximum number of bytes available in the string or in the array Tag.

Ethernet/IP CIP

The protocol has been implemented according to the published Ethernet/IP specifications (available from www.odva.org).

The Ethernet/IP CIP driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. Although the Ethernet/IP CIP driver is fast, we suggest to use short Tag names. Tags are read from and written to the device by specifying their symbolic name in the communications request, therefore the longer the tag name is, the larger the request will be.

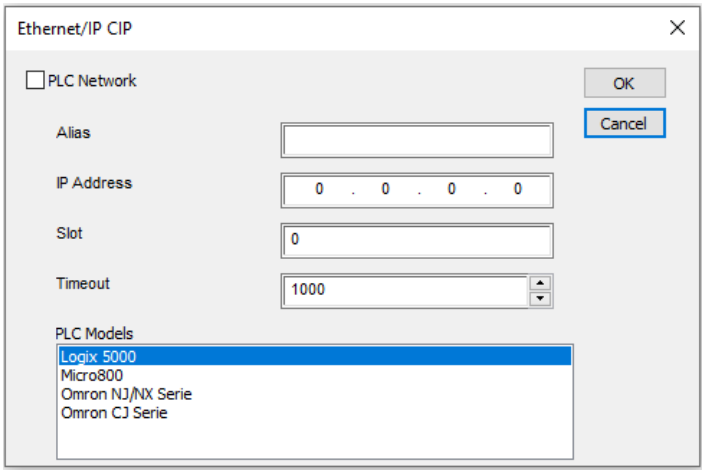
Protocol Editor Settings

Adding a protocol

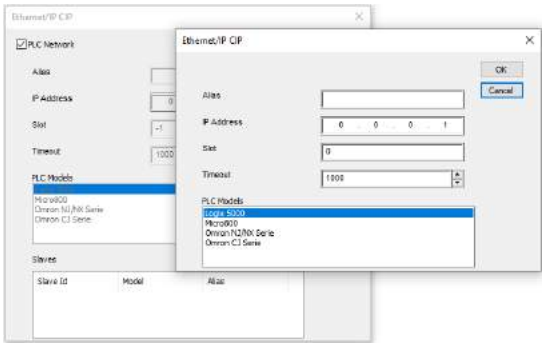
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Field	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP Address	Ethernet IP address of the controller.
Slot	CPU slot number for Logix 5000 models (typically 0). Refer to the controller documentation for further details.

Field	Description
PLC Models	PLC model used to import tags file.
PLC Network	Enable access to multiple networked controllers. For every controller (slave) set the proper option. 

Controller Model Logix 5000

The Ethernet/IP CIP driver allows to connect Allen-Bradley ControlLogix and CompactLogix Ethernet controllers.

Communication with ControlLogix® 5500 controllers can be accomplished through an Ethernet/IP communication module for Ethernet such as the 1756-EN2T or 1756-ENET.

Ethernet communication with CompactLogix™ 5300 controllers requires a processor with a built-in Ethernet/IP port such as the 1769-L32E.

All trademarks are the property of their respective owners.

The internal memory organization of the Logix CPUs is not fixed but configured by the user at development time. Each data item can be identified by a string called “Tag”. The RSLogix 5000 software can then export to the application the list of Tags created for each controller.

The project loaded on the HMI device must refer to Tag names assigned in RSLogix 5000 software at development time. The Tag Editor supports direct import of the Tag file generated by RSLogix 5000 software in .CSV format.

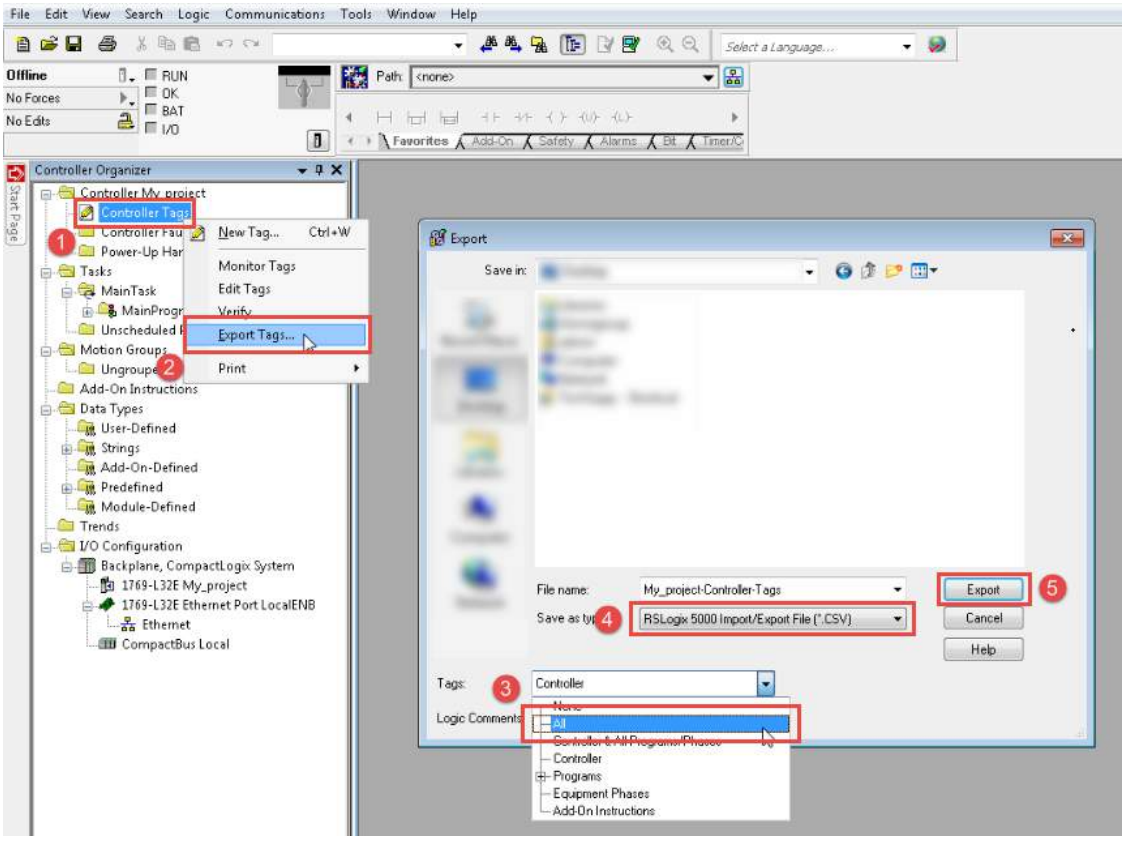
The implementation of the Ethernet/IP driver also supports access to structured data types which can be imported from .L5X files.

The driver supports access to both Controller and Program Tags.

Export CSV and L5X files using RSLogix5000

To export the **.CSV** Tag file:

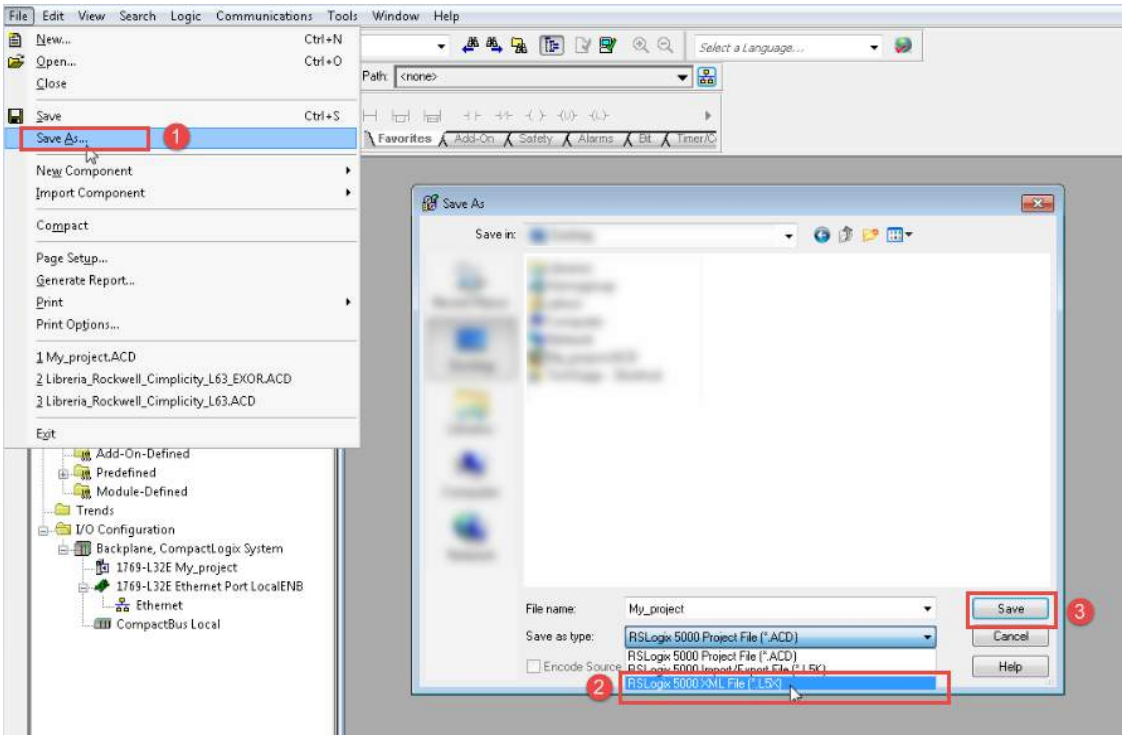
1. From the **Controller Organizer** pane, right-click on **Controller Tags**.
2. Select **Export Tags**: the **Export** dialog is displayed.



3. Choose **All** from the **Tags** list to export all Tags.
4. Select the **Save as type** option to **.CSV**.
5. Click **Export**: all the Tags are exported to an **.CSV** file.

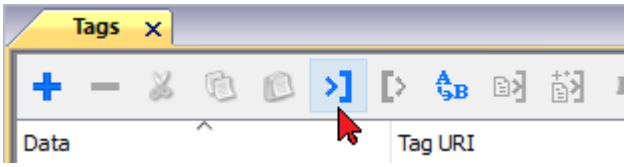
To export the **.L5X** data type file:

1. Choose **File > Save As**.
2. Select the **Save as type** option to **.L5X**.
3. Click **Save**: all the Tags are exported to an **.L5X** file.

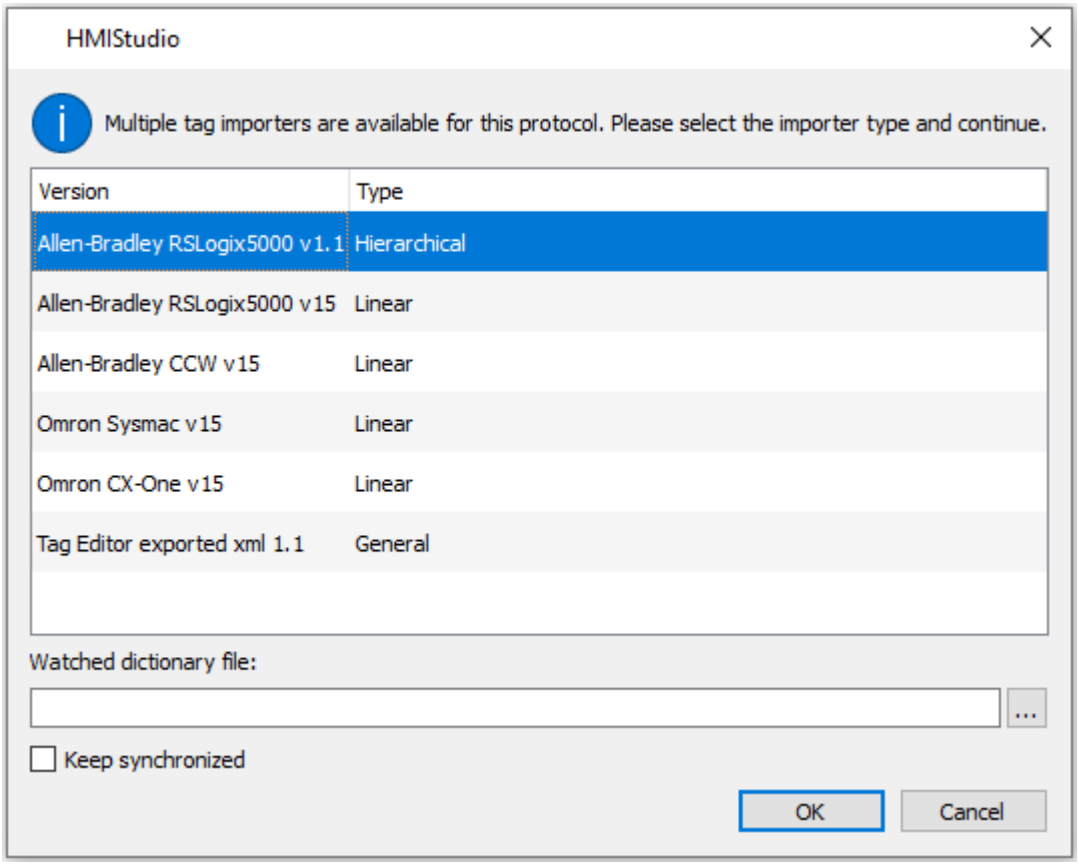


Import Files in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



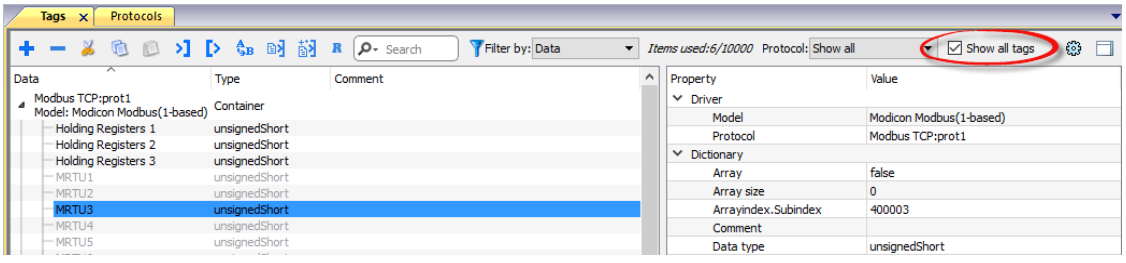
The following dialog shows which importer type can be selected.



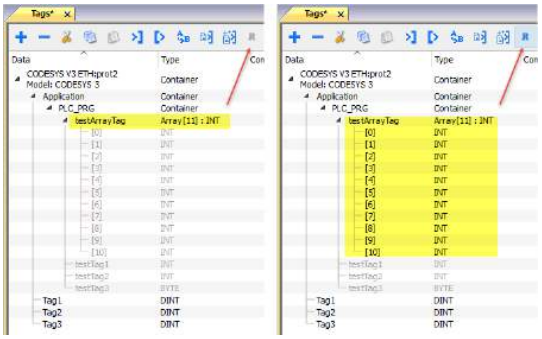
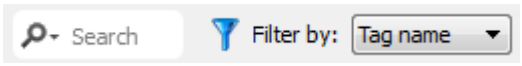
Select **Allen-Bradley RSLogix5000 v15** option.

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.

Toolbar item	Description
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Note: When importing the array data types, the importer is expanding them creating individual Tags per each array element; this is valid for all the data types, except for arrays of boolean. In this case they are imported as “boolean-32” and the single array element can be addressed using “Tag Index” parameter from “Attach to…” dialog.

Module-Defined and User-Defined data types

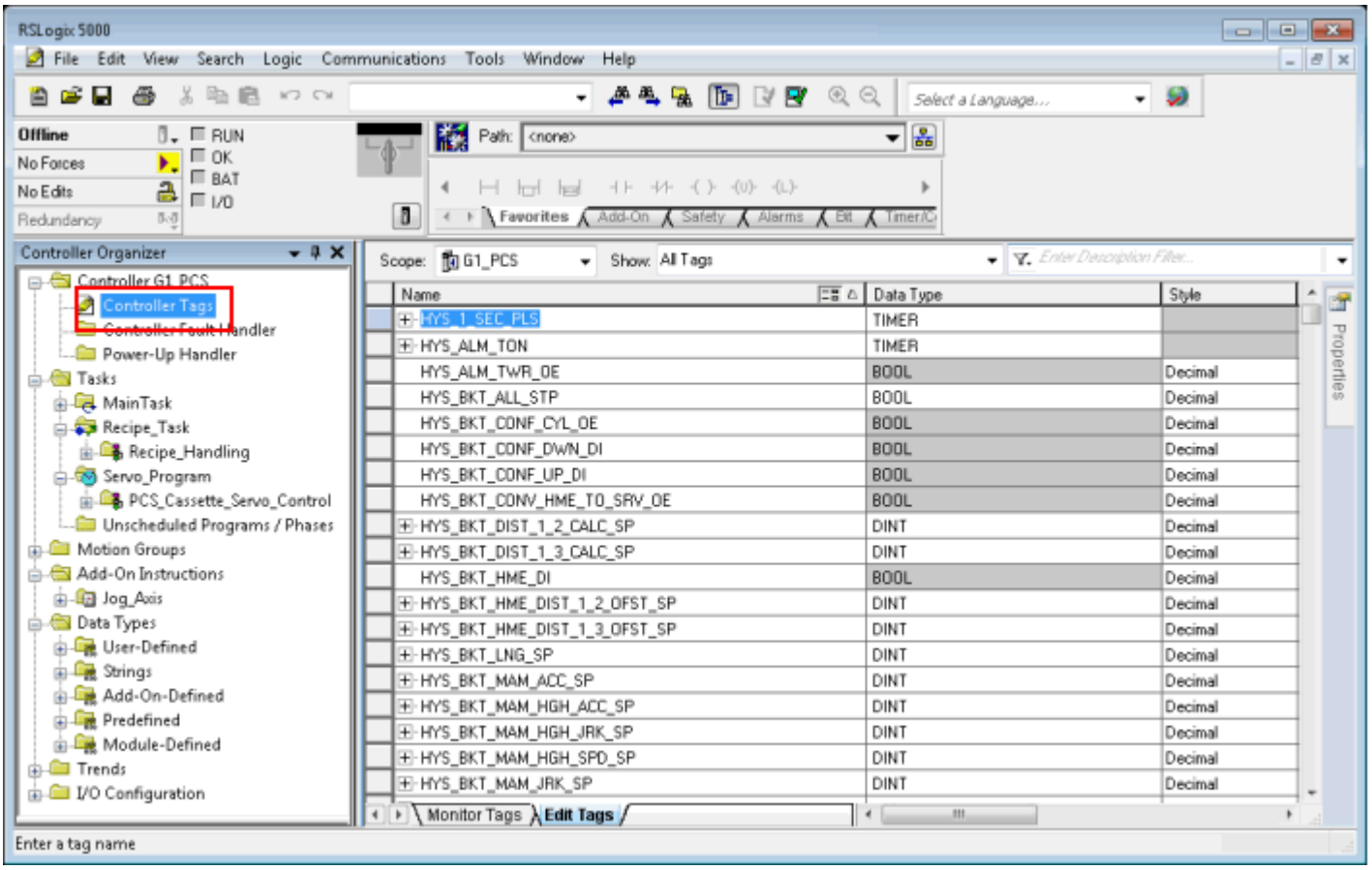
RSLogix 5000 allows you to define Tags with several data types.

Data type group	Description
Predefined	Standard data types such as BOOL, DINT, SINT, INT and other less common data types such as PID, COUNTER, TIMER.
Module-Defined	Data type associated with I/O optional modules usually referenced by aliases.
User-Defined	Custom data type defined by user

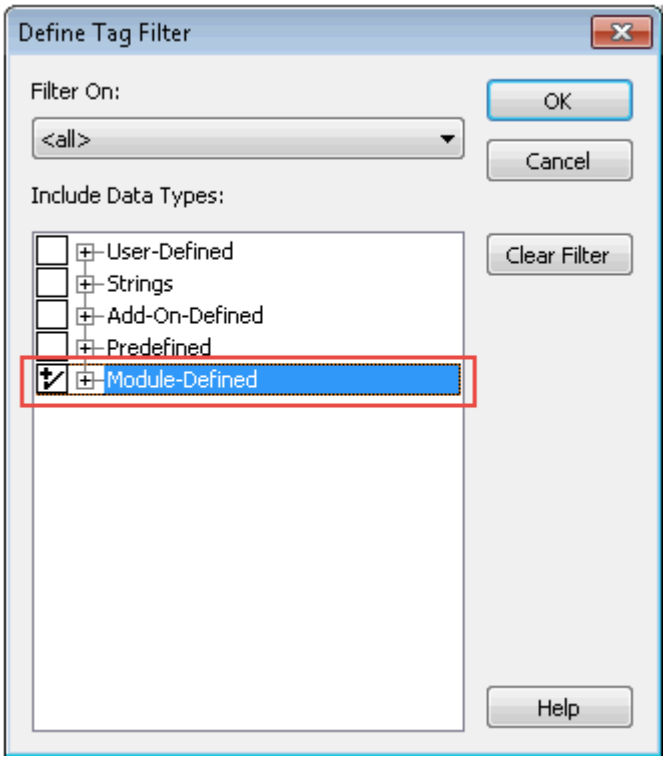
In order to import Predefined (with the exception of standard data types which are always imported) and Module-Defined data type you need to edit the ETIPSpecialDataTypes.xml file located under *languages\shared\studio\tagimport* or *studio\tagimport* depending on installed version.

In RSLogix5000 software:

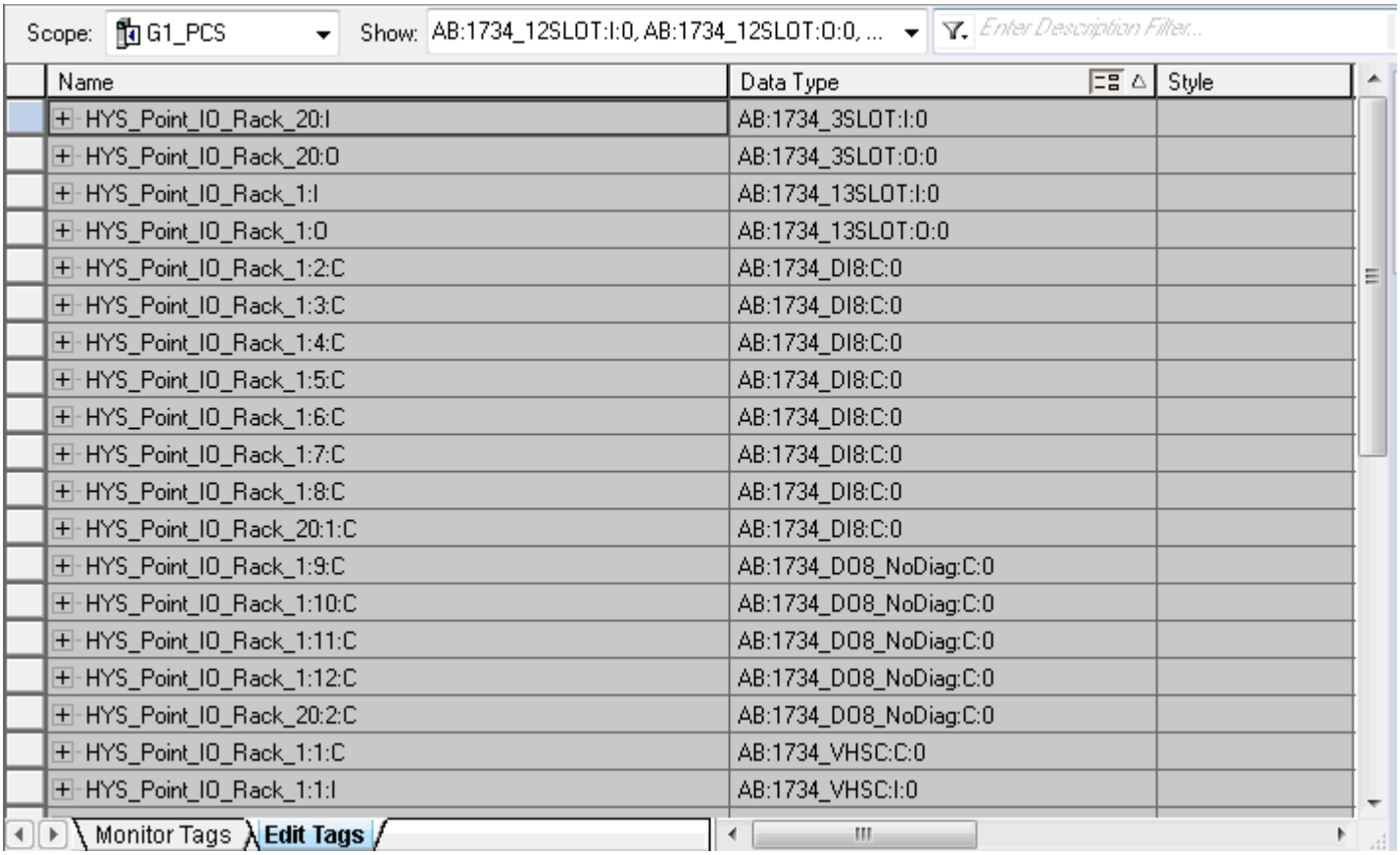
1. From the **Controller Organizer** pane, select **Controller Tags**.



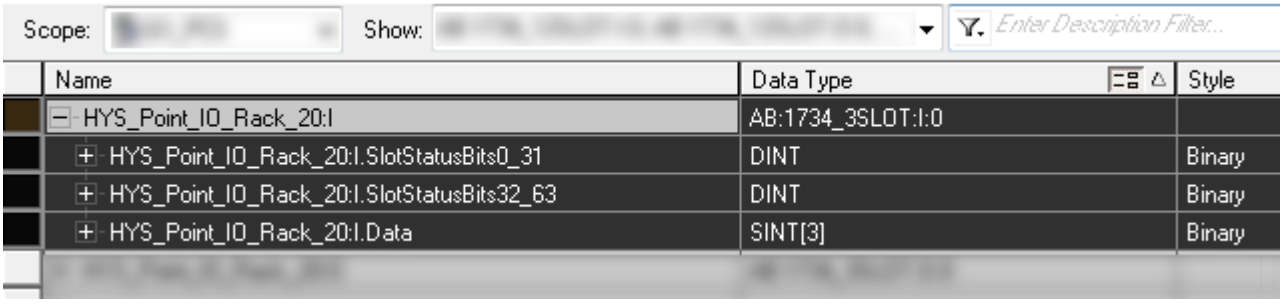
2. Filter tags to display only **Module-Defined** Tags.



Only tags (alias) with data type belonging to optional I/O Modules will be displayed.



In this example alias HYS_Point_IO_Rack_20:I refers to data type AB:1734_3SLOT:I:0. Expand this tag to see how this data type is structured:



To make sure that HYS_Point_IO_Rack_20:I, and all his sub-tags, will be imported into the project, open the ETIPSpecialDataTypes.xml file in any text editor and check if the AB:1734_3SLOT:I:0 data type is included. If so you can proceed with the following data type. If not, you need to add it manually.

The structure is as in this example:

```
<DataType Name="aaa">
<Members>
<Member Name="bbb" DataType="ccc" Dimension="ddd" Radix="eee"/>
</Members>
</DataType>
```

where:

- `aaa` = Alias/Tag data type
- `bbb` = Sub-tag Name (it's sub-tag name part after dot)
- `ccc` = Sub-tag data type
- `ddd` = Array dimension (0 if it is not an array)
- `eee` = Style

In the example above:

ETIPSpecialDataTypes.xml

```
238
239  <DataType Name="AB:1734_3SLOT:I:0">
240  <Members>
241   <Member Name="SlotStatusBit0_31" DataType="DINT" Dimension="0" Radix="Binary"/>
242   <Member Name="SlotStatusBit32_63" DataType="DINT" Dimension="0" Radix="Binary"/>
243   <Member Name="Data" DataType="SINT" Dimension="3" Radix="Binary"/>
244  </Members>
245  </DataType>
```

3. Repeat step 2 for all Module-Defined data types.
4. Repeat the procedure from step 2, filtering Tags to display only **Predefined** Tags.

Controller Model Omron Sysmac

Data in NJ and CJ controllers can be accessed via CIP protocol.

Each data item can be identified by a string called “Tag”. Use appropriate programming tools for controller to export the list of Tags.

NJ series controller are programmed using Sysmac Studio:

- NJ301-xxxx
- NJ501-xxxx

CJ series controller are programmed using CX-One:

- CJ2M CPU-3x
- CJ2H CPU 6x-EIP
- Any CPU with a CJ1W-EIP21 attached.

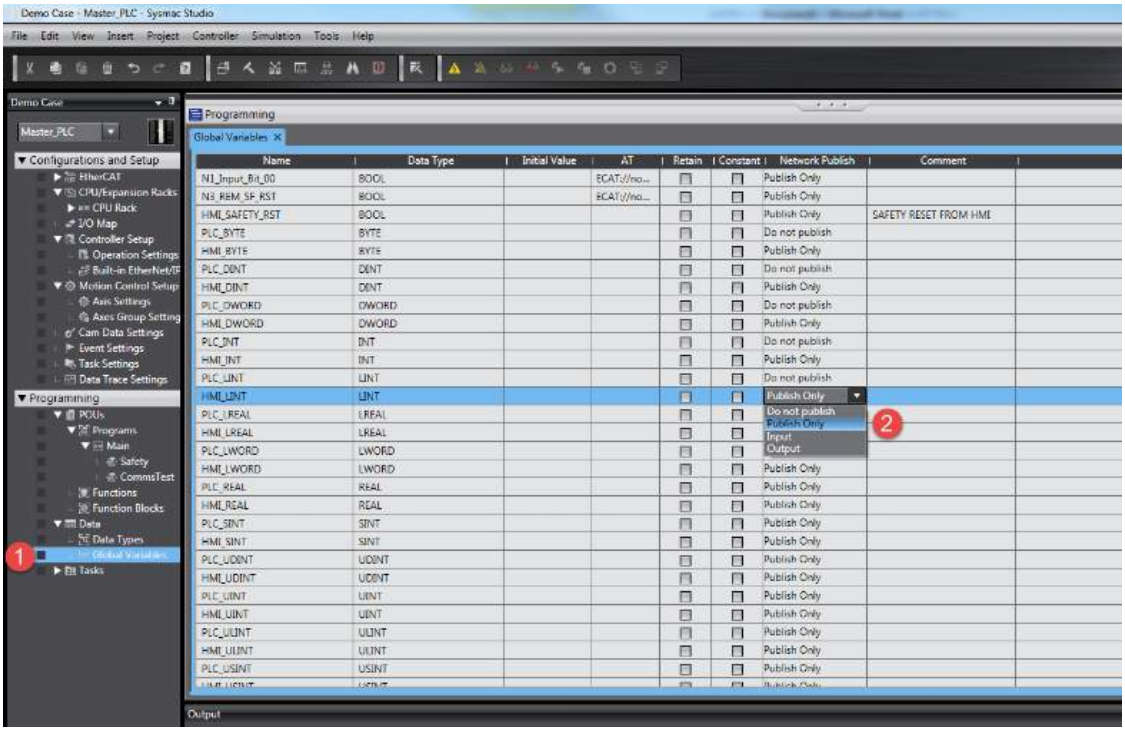
The project loaded on the HMI device must refer to the Tag names assigned in the programming software at development time. The Tag Editor supports direct import of the Tag file generated by Sysmac Studio software in .NJF format or generated by CX-One in the .CJF format.

All Tags to be accessed by the HMI device must be declared as Global Variables.

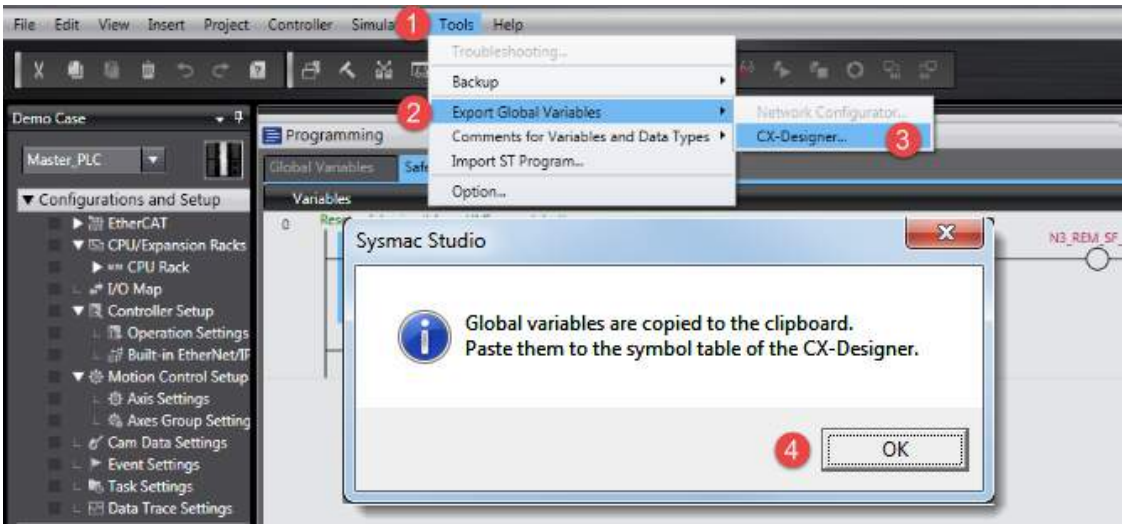
Export NJF files using Sysmac Studio

To export the **.NJF** Tag file:

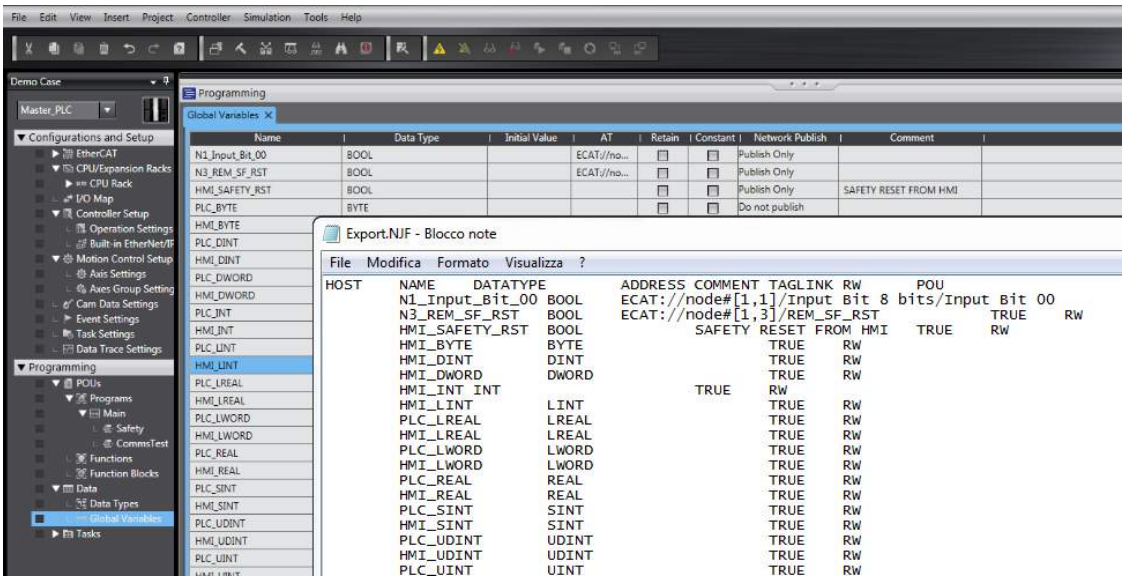
1. In Sysmac Studio declare Tags as **Global Variables**.
2. Set the **Network Publish** attribute to **Publish Only**.



2. From the **Tools** menu, choose **Export Global Variables > CX-Designer**.



3. Click **OK** to confirm.
4. Cut and paste the content of the clipboard in any text editor.



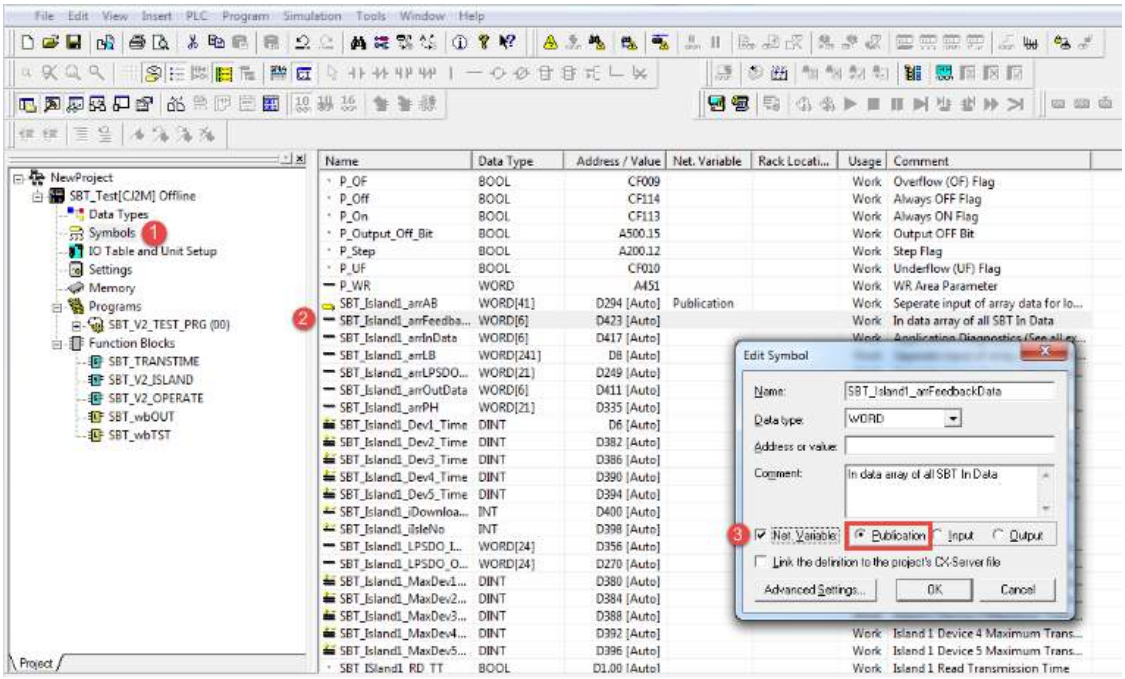
4. Save the file as **.NJF**.

Note: Using Notepad as text editor, make sure to save the text file with **.NJF** extension by selecting "Save as type" as "All Files" although the file will be named *.njf.txt and it will not be visible from importer.

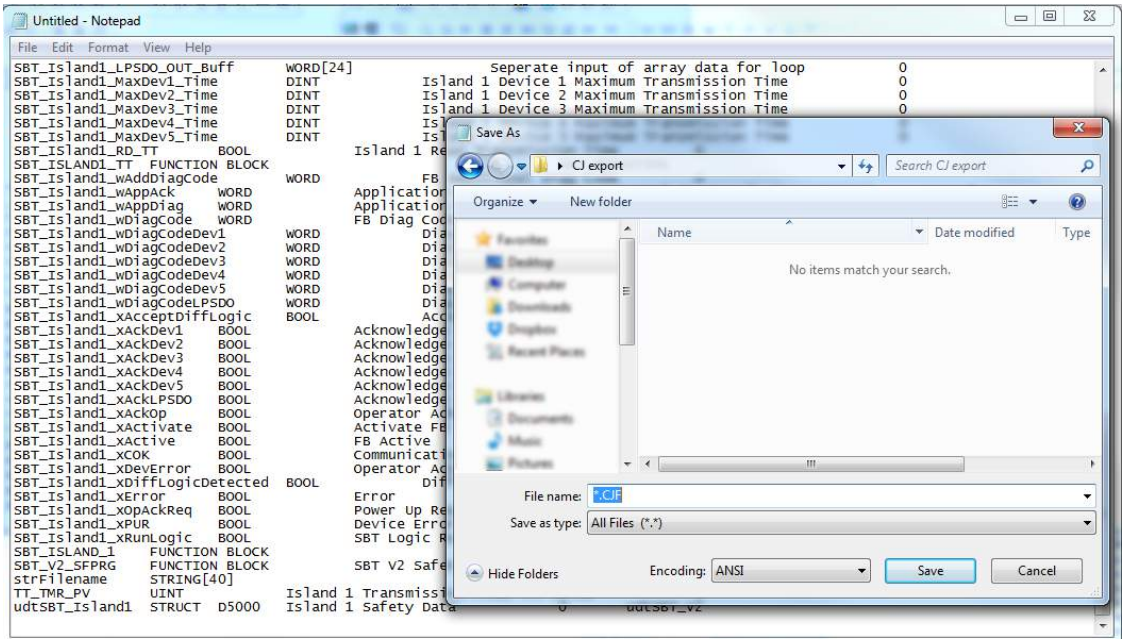
Export CJF file using CX-One

To export the **.CJF** Tag file:

1. In CX-One open the Symbols file in the project.
2. In the **Edit Symbol** dialog set the **Net. Variables** attribute to **Publication**.



3. Copy and paste all the Tags in any text editor.



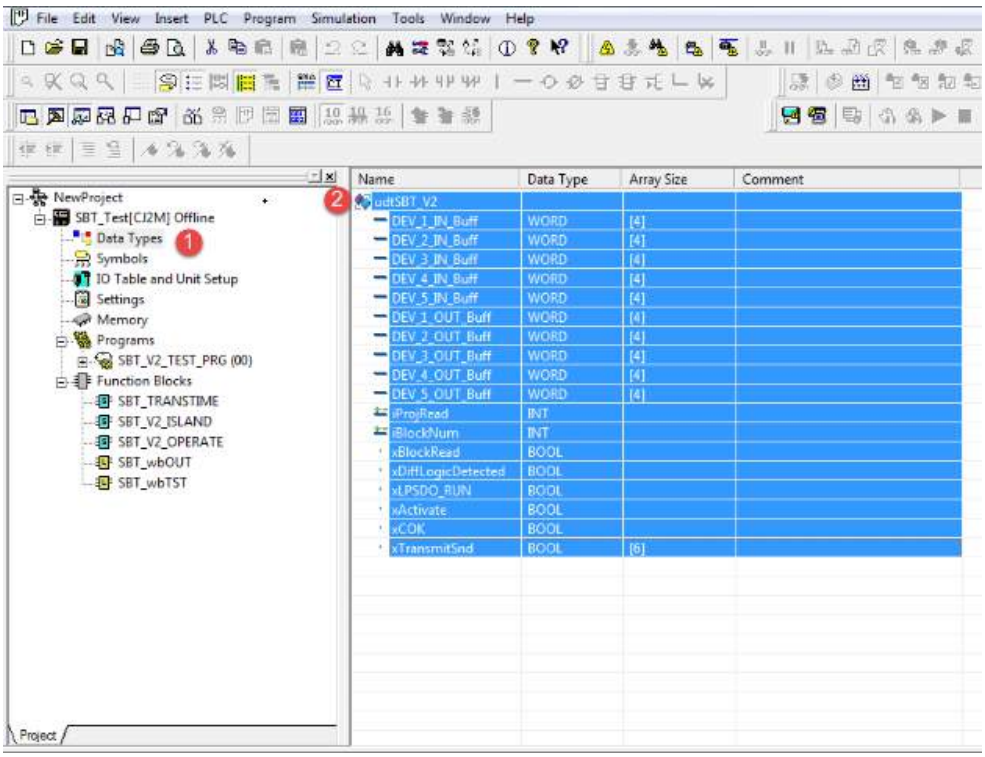
4. Save the file as **.CJF**.

Note: Using Notepad as text editor, make sure to save the text file with **.CJF** extension by selecting "Save as type" as "All Files" although the file will be named *.cjf.txt and it will not be visible from importer.

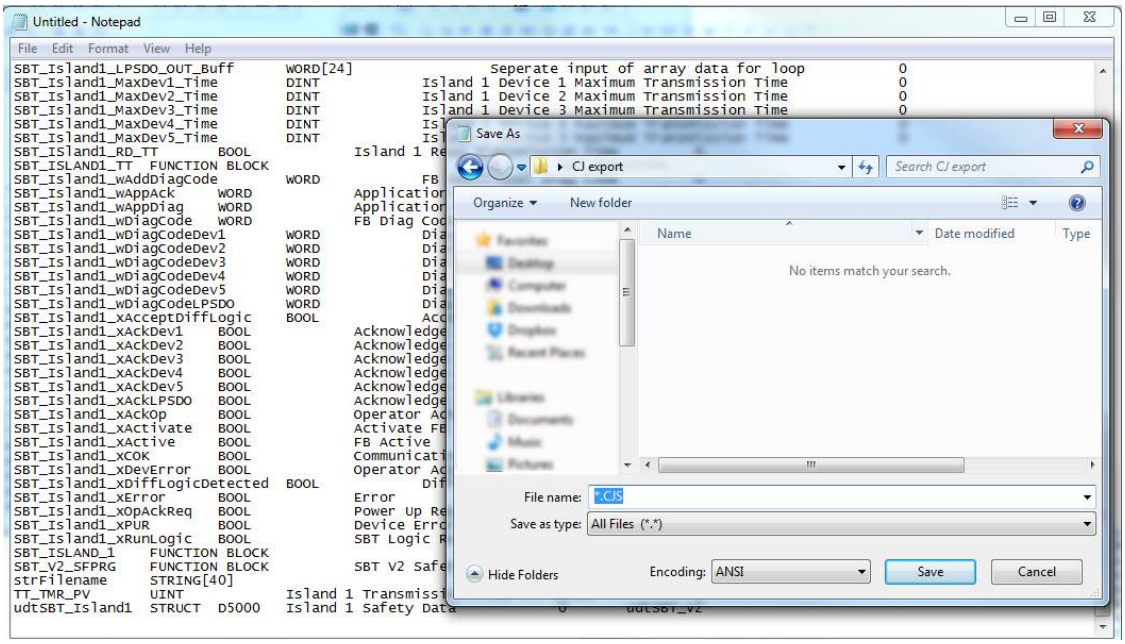
Export User Defined structures

To export the **.CJS** Tag file:

1. In CX-One open the Data Types file in the project.



2. Copy and paste all the Tags in any text editor.

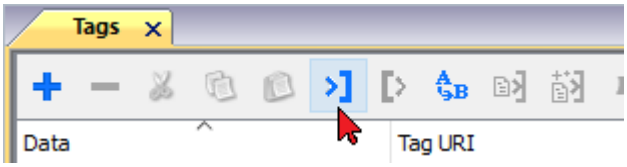


3. Save the file as **.CJS**.

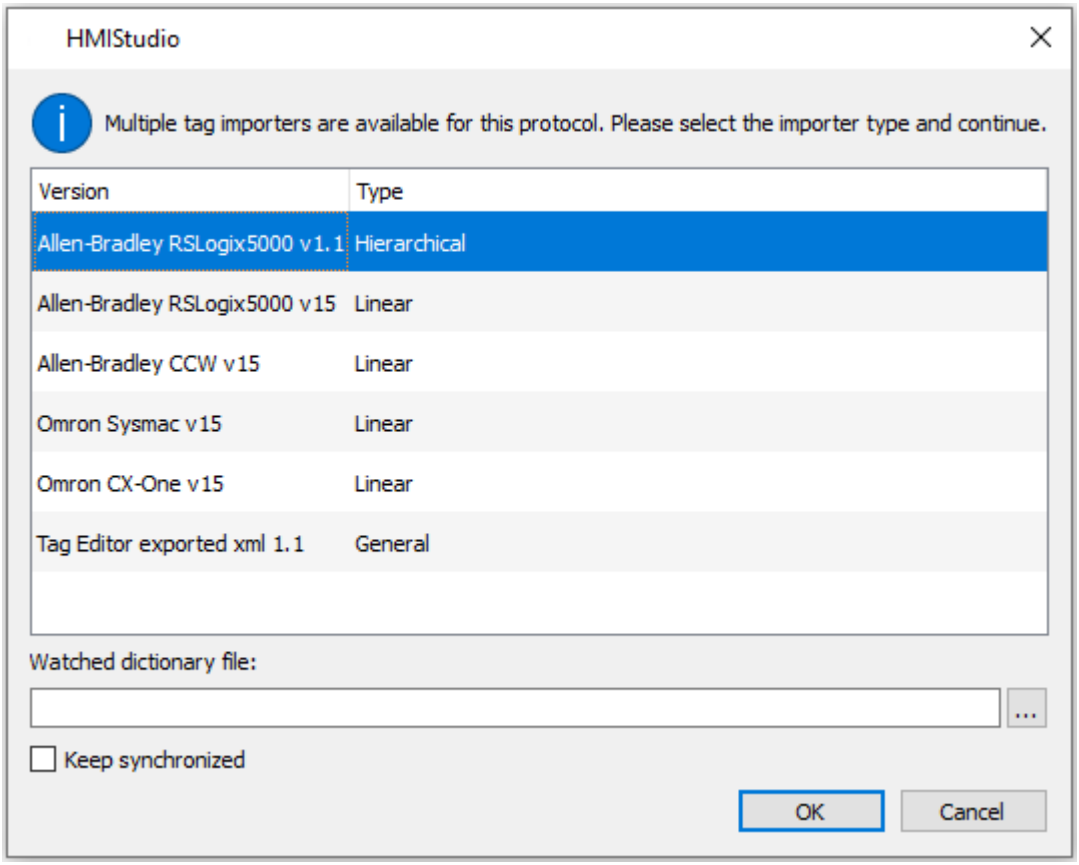
Note: Using Notepad as text editor, make sure to save the text file with **.CJS** extension by selecting "Save as type" as "All Files" although the file will be named *.cjs.txt and it will not be visible from importer.

Import Files in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



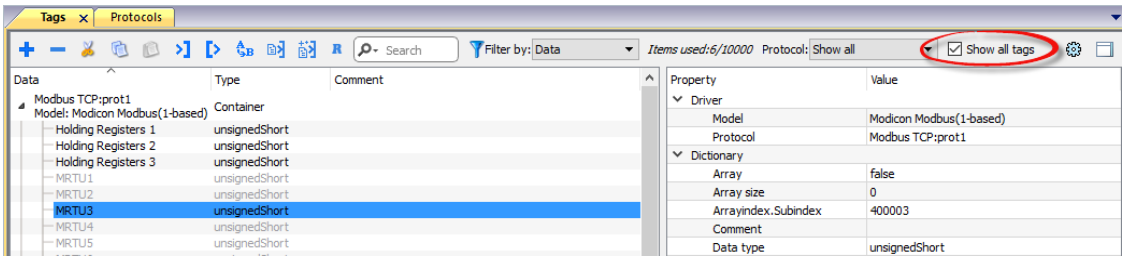
The following dialog shows which importer type can be selected.



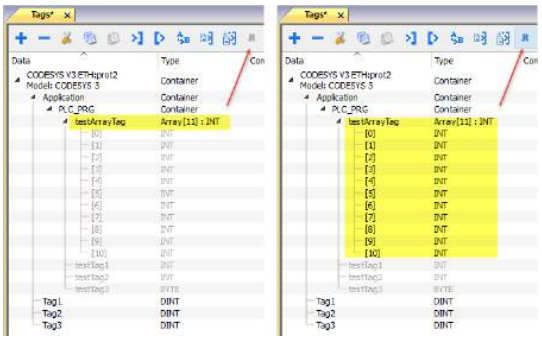
Select **Omron Sysmac** to import a **.NJF** Tags file or **Omron CX-One** to import a **.CJF** Tags file.

Once the importer has been selected, locate the Tags file and click **Open**. The system will ask for User Defined structures **.CJS** file. If not required, skip the dialog by clicking on Cancel button.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a

Toolbar item	Description
	new dictionary import.
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
Search Filter by: Tag name	Searches tags in the dictionary basing on filter combo-box item selected.

Note: When importing the array data types, the importer is expanding them creating individual Tags per each array element; this is valid for all the data types, except for arrays of boolean. In this case they are imported as “boolean-32” and the single array element can be addressed using “Tag Index” parameter from “Attach to…” dialog.

Controller Model Micro800

The Ethernet/IP CIP driver provides an easy and reliable way to connect to Allen-Bradley Micro800 controllers.

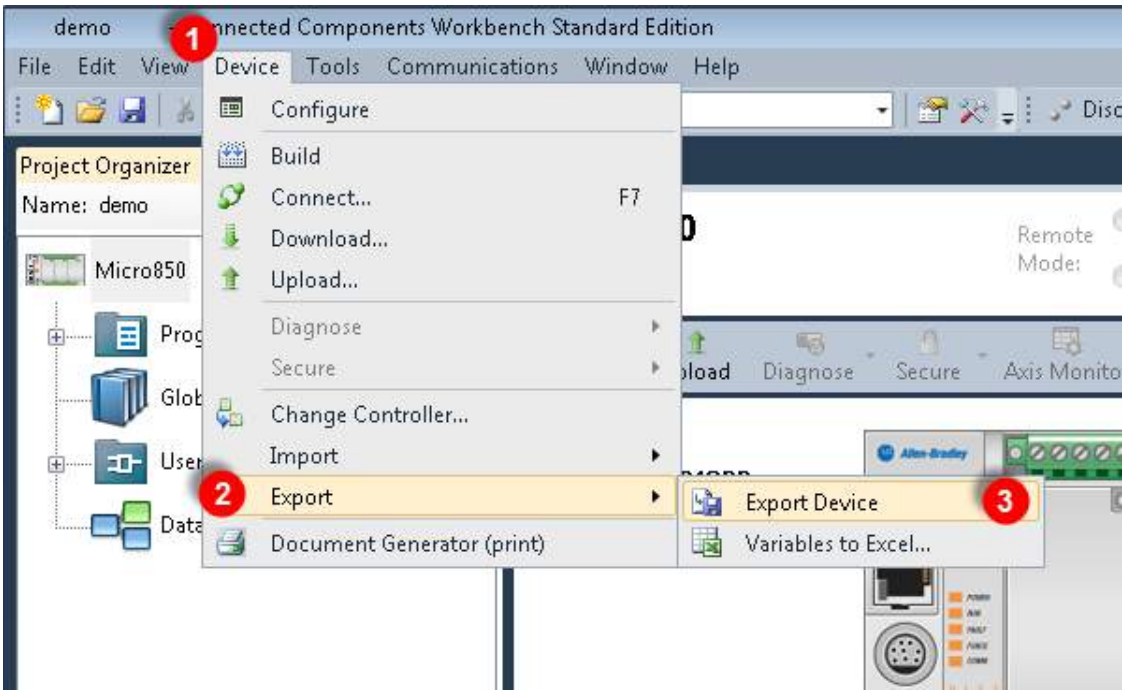
The scope of variables into a Micro800 controller can be local to a program or global:

Scope	Description
Local Variables	Program-scoped Tags. Tags are assigned to a specific program in the project and available only to that program. These Tags are **not supported** within this driver.
Global Variables	Controller-scoped Tags. Tags belong to the controller in the project and are available to any program in the project. These Tags are **supported** within this driver.

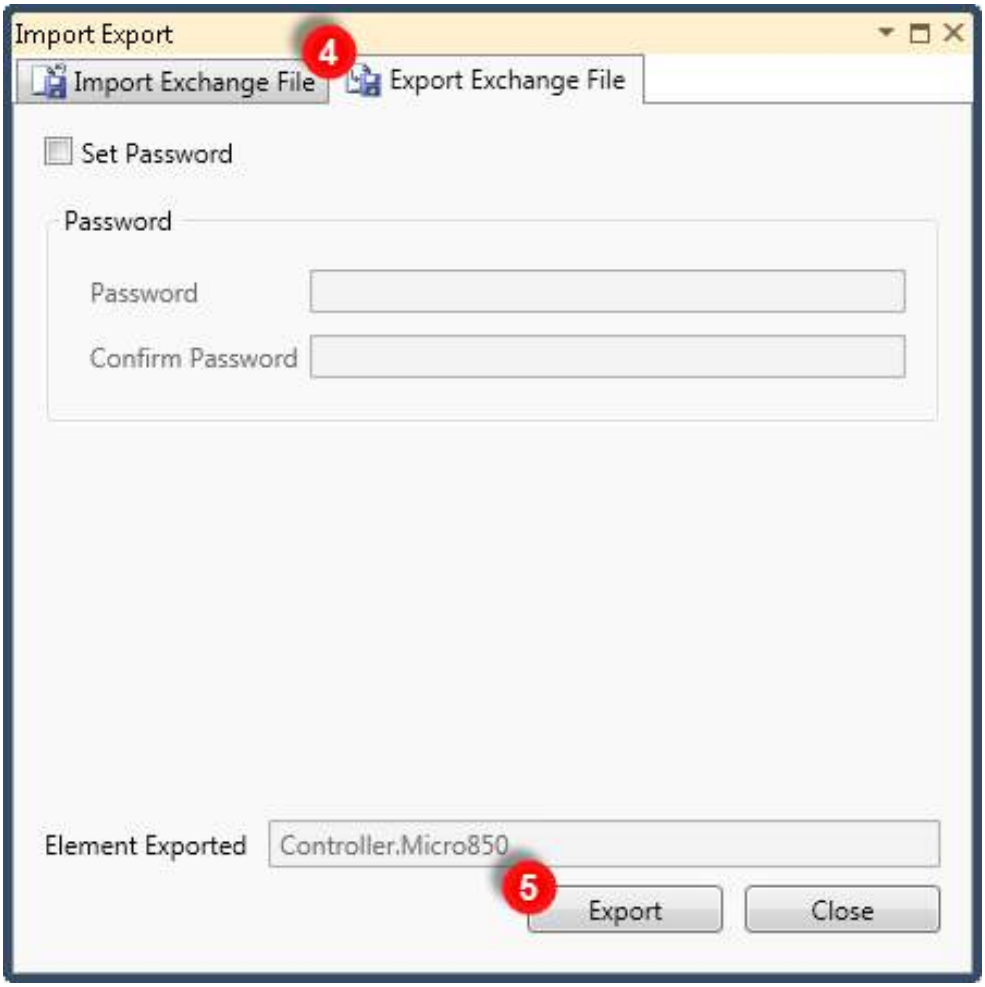
Export ISAXML file using Connected Component Workbench

To export **.ISAXML** global variables including I/O tags:

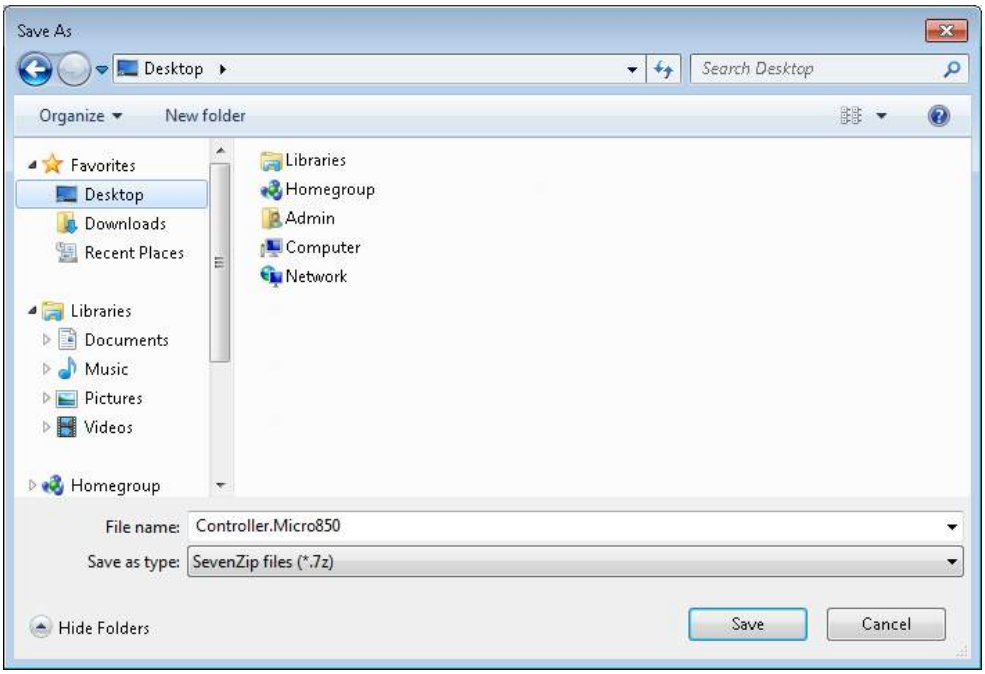
1. Select **Device** tab.
2. Expand **Export** item.
3. Select **Export Device**.



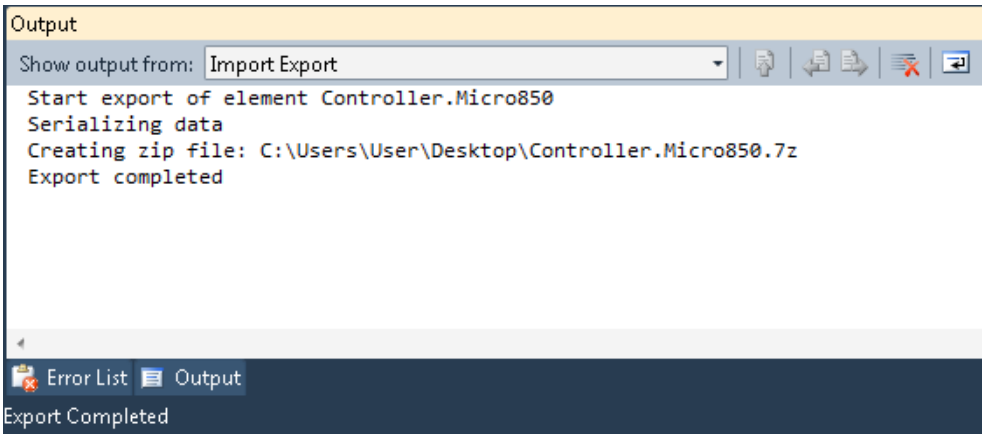
4. Click on**Export Exchange File** tab.
5. Click **Export** button.



6. Choose a location where to save the export file and click **Save**.



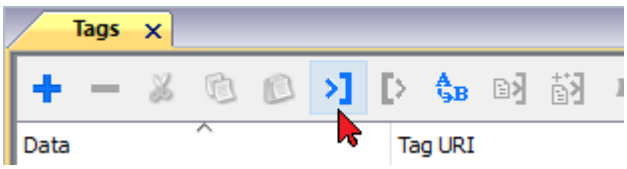
7. When the export is completed successfully the output information is displayed:



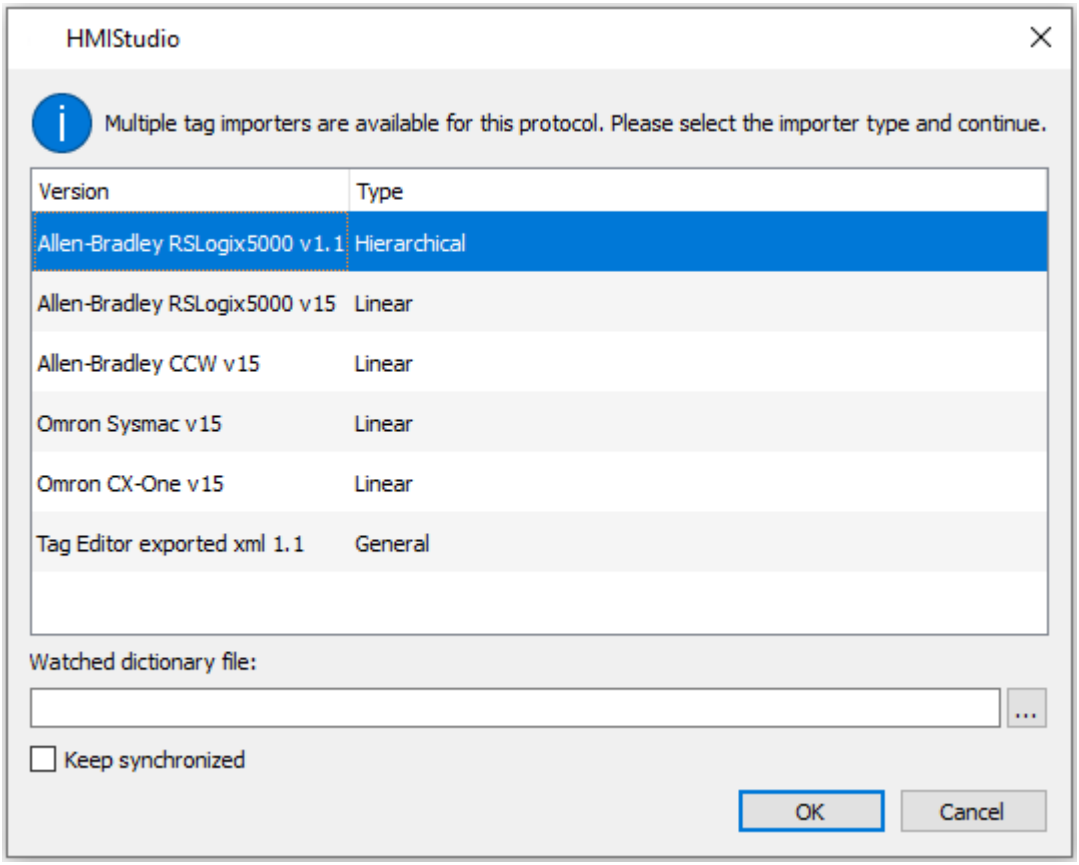
Note: CCW export file is a 7-zip compressed archive. Use a suitable zip utility to extract archive content into a local folder.

Import Files in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



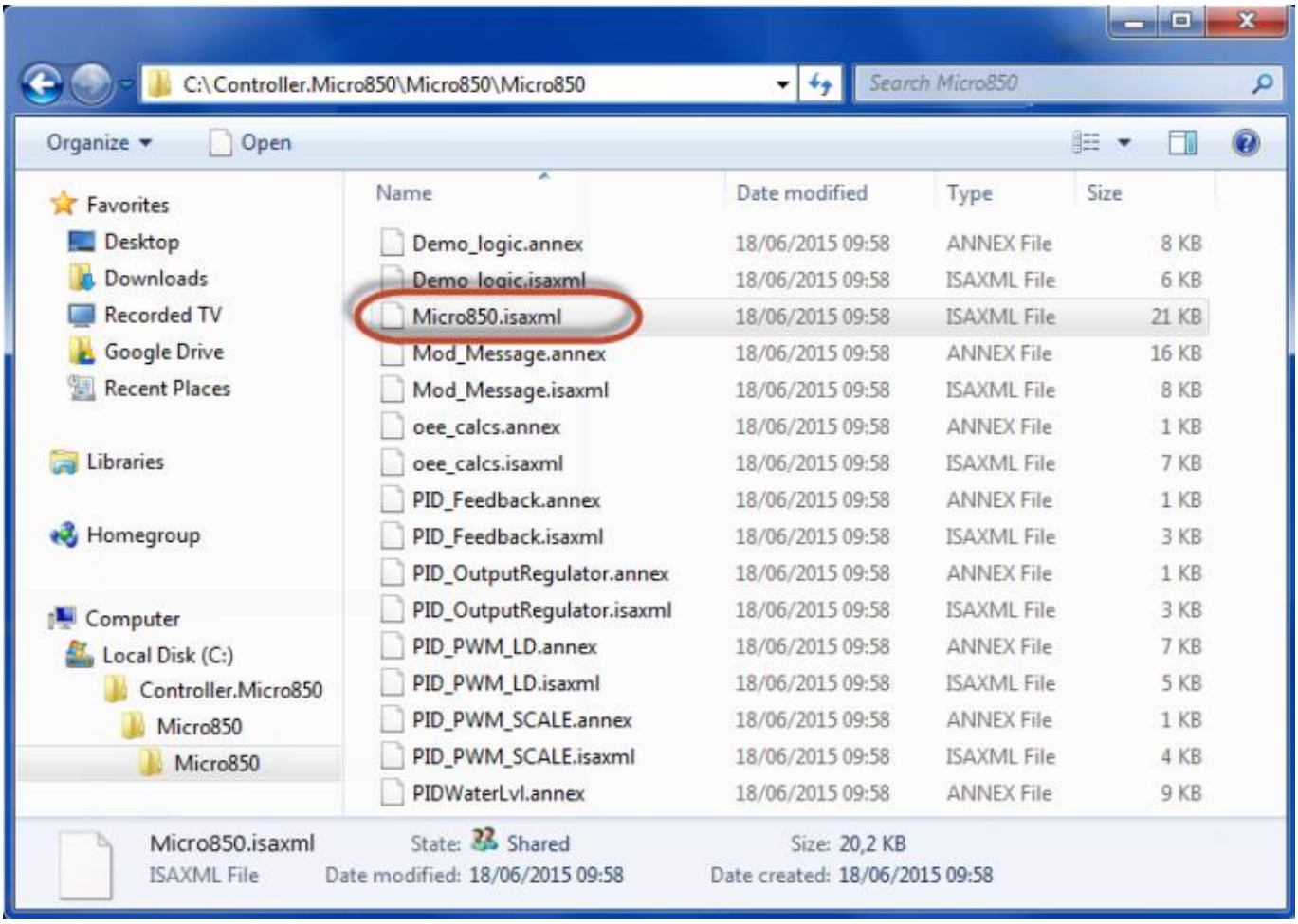
The following dialog shows which importer type can be selected.



Select **Allen-Bradely CCW v15** option.

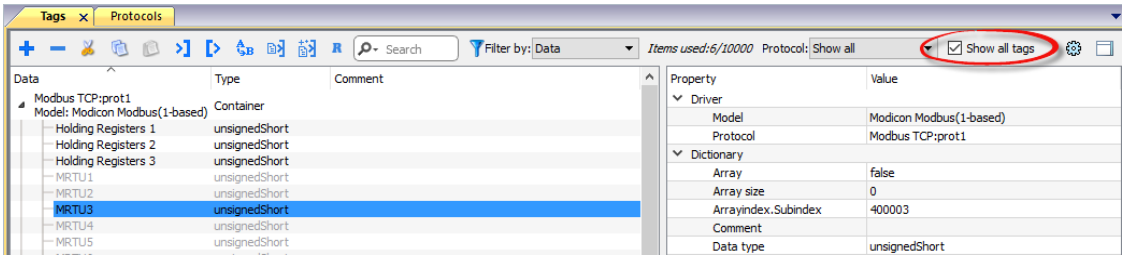
Directory structure extracted from 7z file is something like: "..\<folder_name>\Micro8xx\Micro8xx\"

Inside this last folder, select the Micro8xx.isaxml file as shown below:

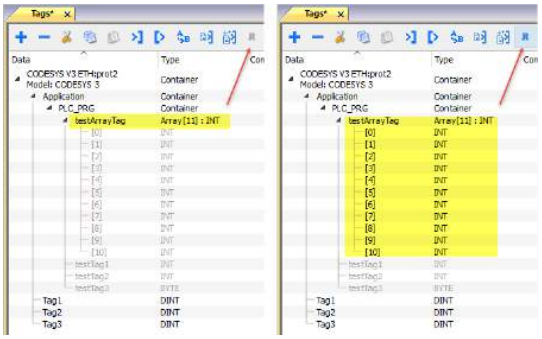
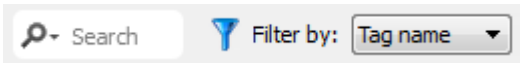


Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**.

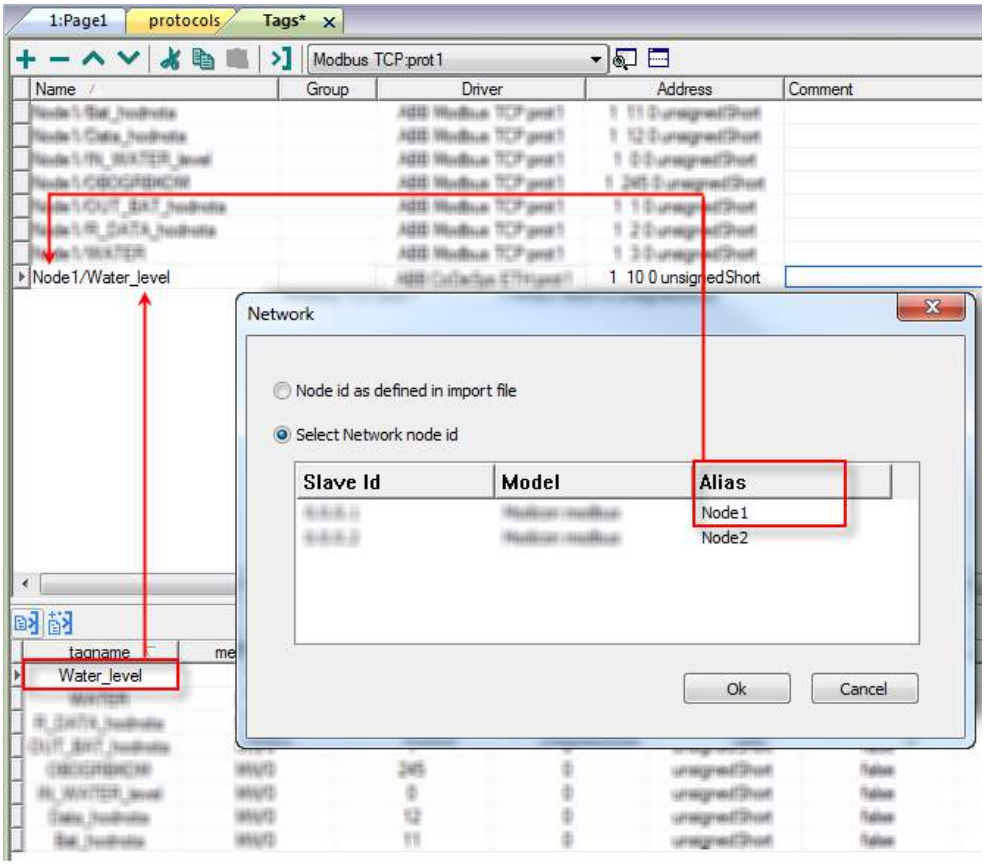
Toolbar item	Description
	Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



Note: Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

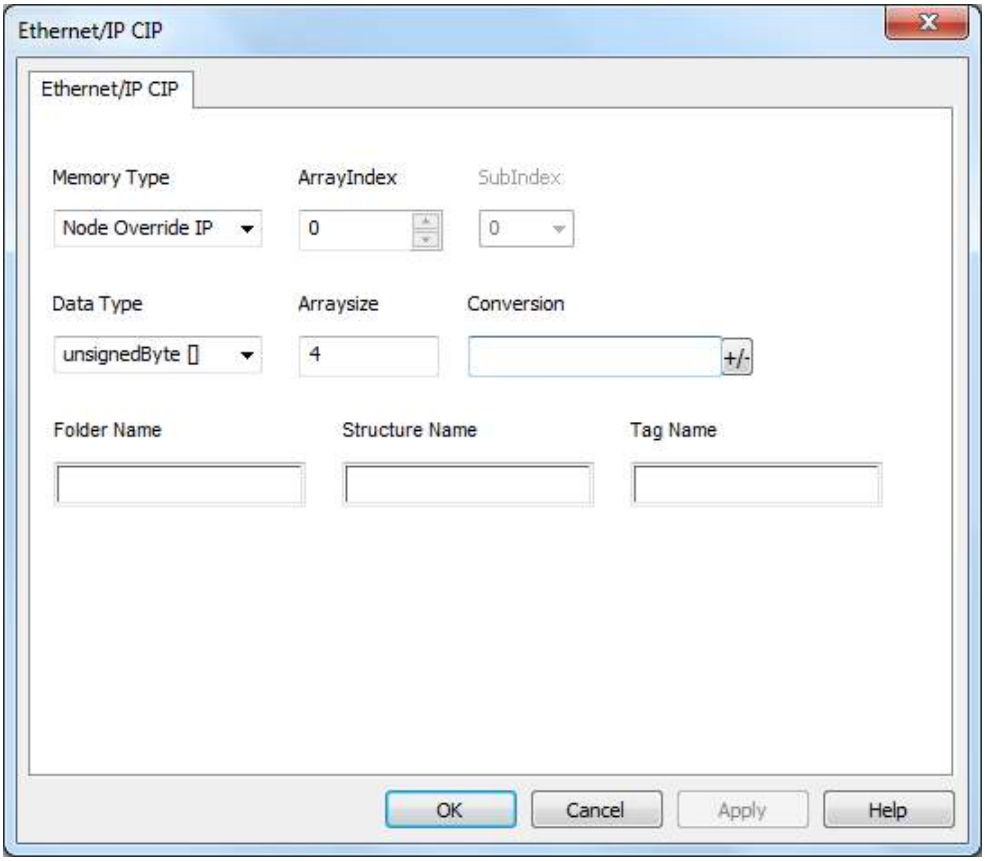
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

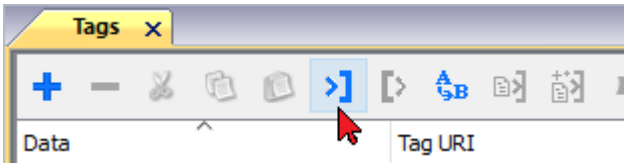
Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.

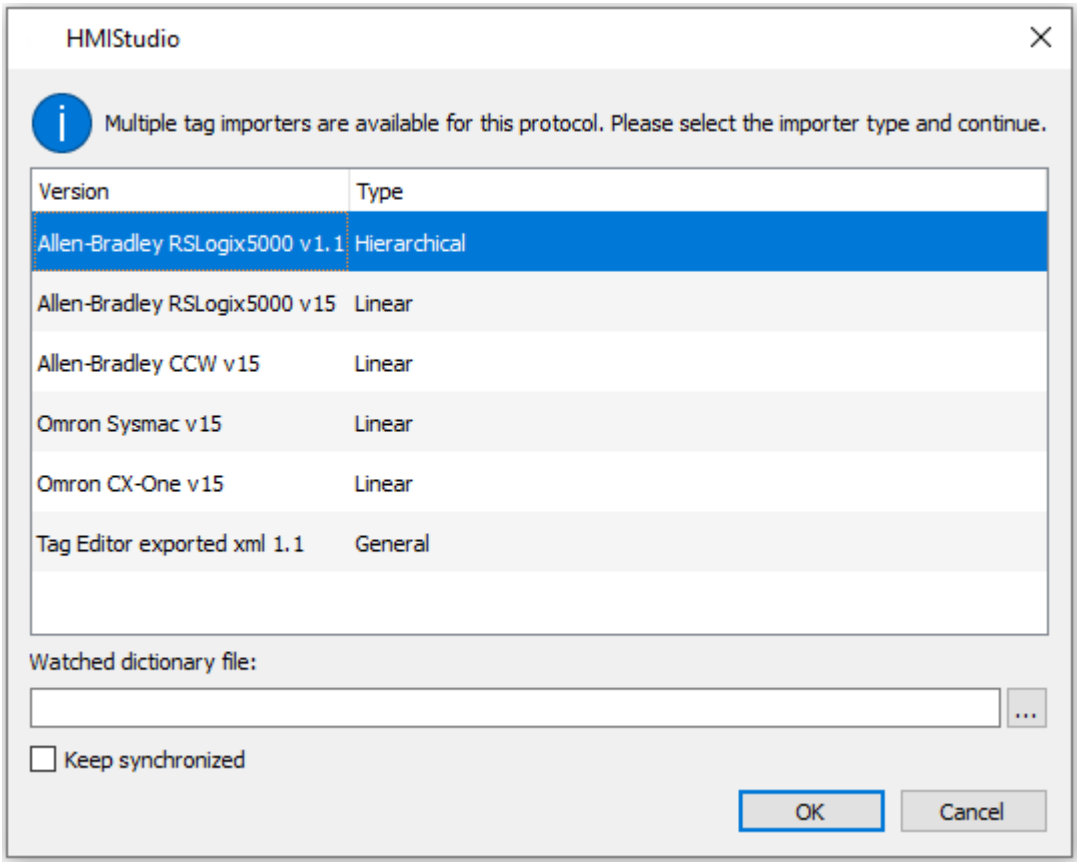


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



The following dialog shows which importer type can be selected.

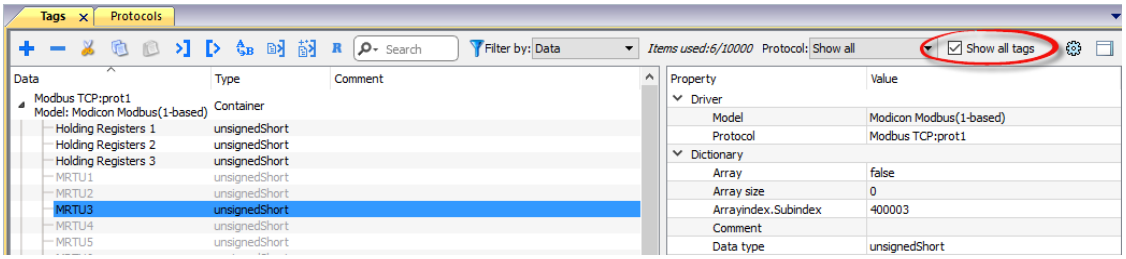


Importer	Description
Allen-Bradley L5X v1.1 Hierarchical	Requires a .L5X file. Check **Controller Model Logix 5000** for more details. All variables will be displayed according to RSLogix5000 Hierarchical view.
Allen-Bradley RSLogix5000 v15 Linear	Requires a **.CSV**and **.L5X** (optional) files. Check **Controller Model Logix 5000** for more details. All variables will be displayed at the same level.
Allen-Bradley CCW v15 Linear	Requires a **.ISAXML** file. Check **Controller Model Micro800** for more details. All variables will be displayed at the same level.
Omron Sysmac v15 Linear	Requires a **.NJF** file. Check **Controller Model Omron Sysmac** for more details. All variables will be displayed at the same level.
Omron CX-One v15 Linear	Requires a **.CJF**and **.CJS** (optional) files. Check **Controller Model Omron Sysmac** for more details.

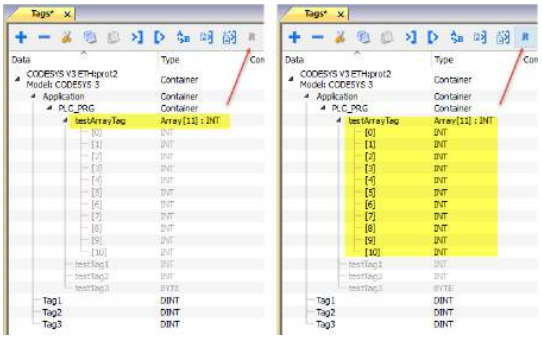
Importer	Description
	All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.

Toolbar item	Description
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
Search Filter by: Tag name	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

Current communication status can be displayed using System Variables. See "System Variables" section in the main manual.

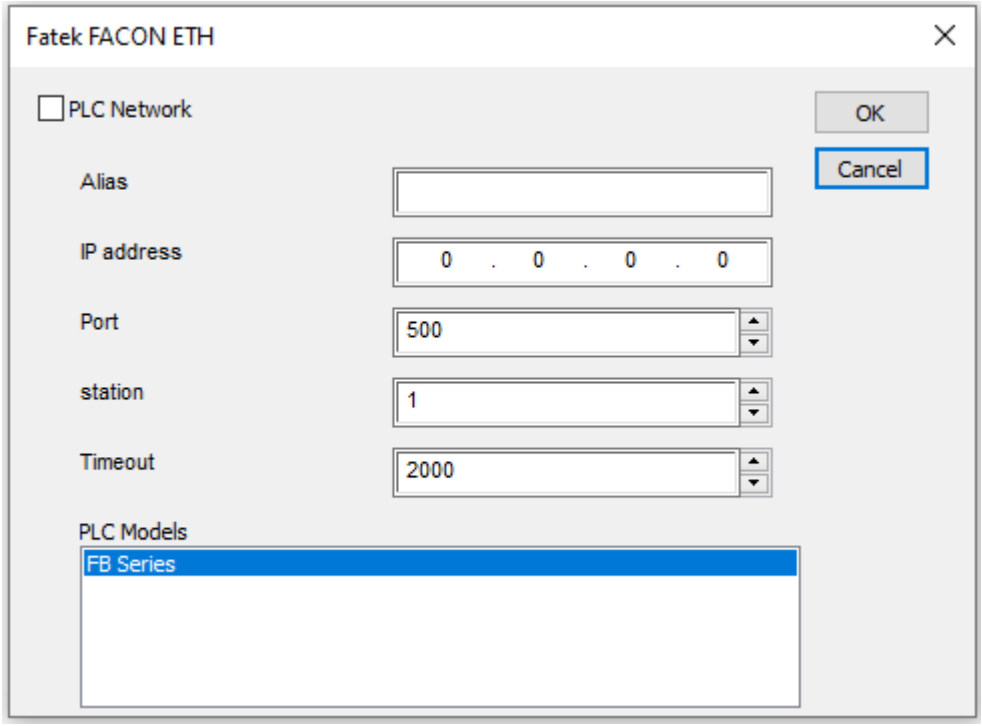
Codes supported for this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid response	The device did received a response with invalid format or contents from the controller .	Ensure the data programmed in the project are consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

Fatek FACON ETH

The Fatek FACON ETH communication driver has been designed to connect HMI devices to a Fatek FACON PLC through Ethernet connection.

Protocol Editor Settings



Adding a protocol

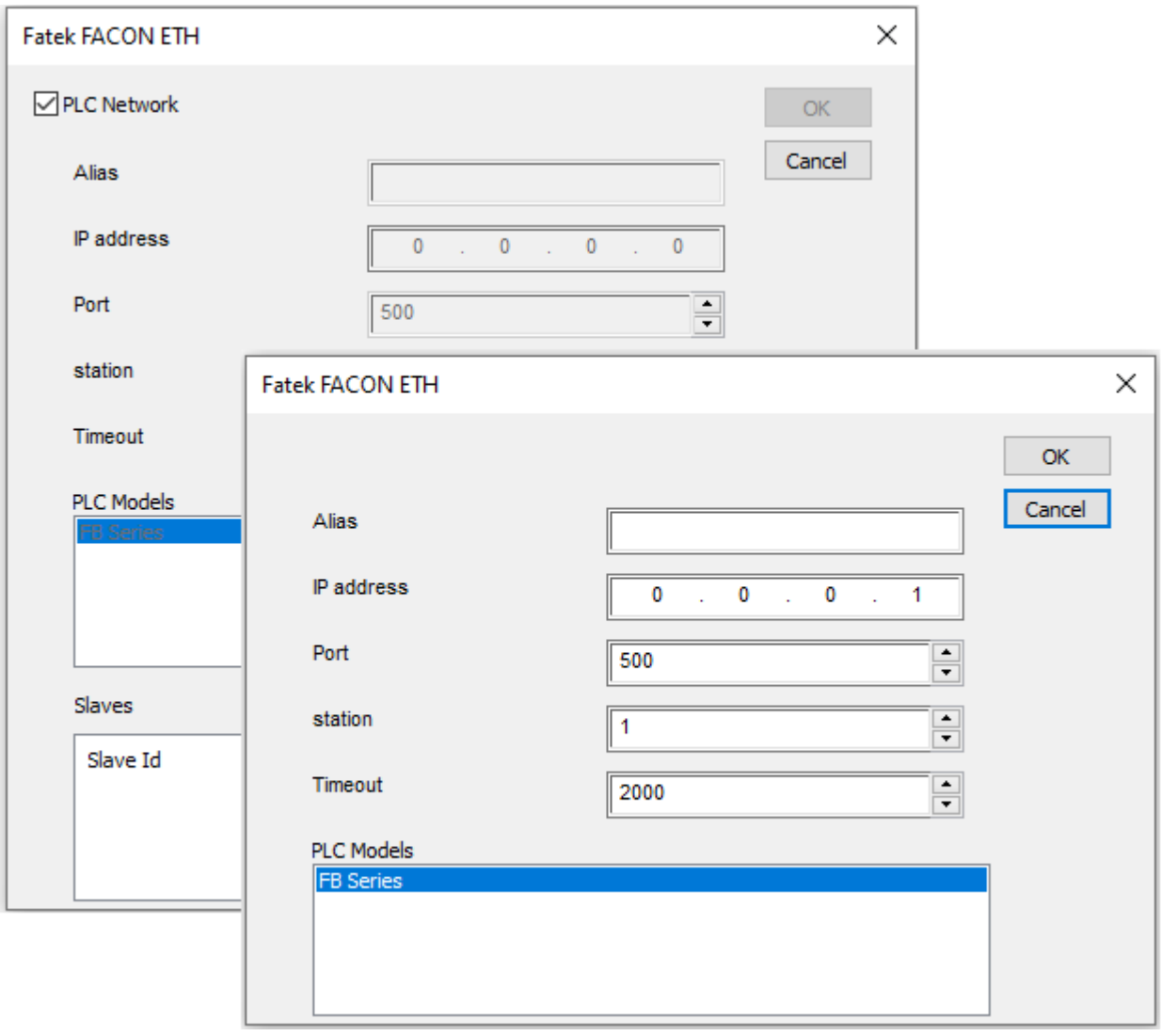
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.

Element	Description
IP Address	Ethernet IP address of the PLC.
Port	Port number used to communicate with PLC.
station	station number according to PLC configuration.

Element	Description
Timeout	Time delay in milliseconds between two retries in case of missing response from the PLC.
PLC Models	PLC model available: • FB Series
PLC Network	IP address for all controllers in multiple connections. **PLC Network** must be selected to enable multiple connections.



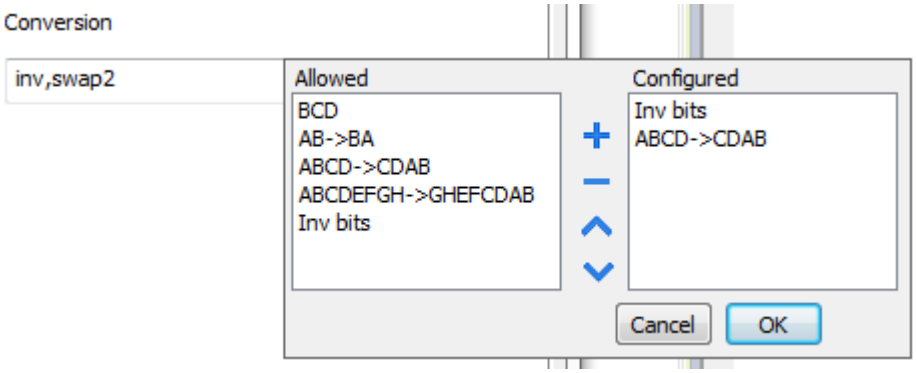
Tag Editor Settings

In Tag Editor select the protocol **Fatek FACON ETH**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:


Fatek FACON ETH
Fatek FACON ETH
Memory Type
Offset
SubIndex
Input Discrete
0
0
Data Type
Arraysize
Conversion
boolean
0
+/-
OK
Cancel
Apply
Help

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Input Discrete</td><td>X resources. Corresponding to External Digital Input Point.</td></tr>
<tr><td>Output Relay</td><td>Y resources. Corresponding to External Digital Output Point.</td></tr>
<tr><td>Internal Relay</td><td>M resources. Corresponding to PLC internal memory.</td></tr>
<tr><td>Step Relay</td><td>S resources.</td></tr>
<tr><td>Timer Discrete</td><td>T resources.</td></tr>
<tr><td>Counter Discrete</td><td>C resources.</td></tr>
<tr><td>Timer Register</td><td>Current Time Value Register.</td></tr>
<tr><td>Counter Register</td><td>Current Counter Value Register.</td></tr>
<tr><td>Data Register - HR</td><td>R resources.</td></tr>
<tr><td>Data Register - DR</td><td>D resources.</td></tr>
<tr><td>Run</td><td>Boolean value. Corresponding to PLC status.</td></tr>
<tr><td>Node Override IP</td><td>See Special Data Types for specifications.</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>Starting address for the Tag. The possible range depend on PLC model selected.</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection depending on the selected data type.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
double
string
binary
See "Programming concepts" section in the main manual.
Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).</td></tr>
</table>

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH	**swap4**: Swap bytes in a double word.

Element	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>-> GHEFCDAB</td><td>Example: 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)</td></tr><tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100001001111101 (in binary format)</td></tr><tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9) Example: 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)</td></tr></table> Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the PLC at runtime.

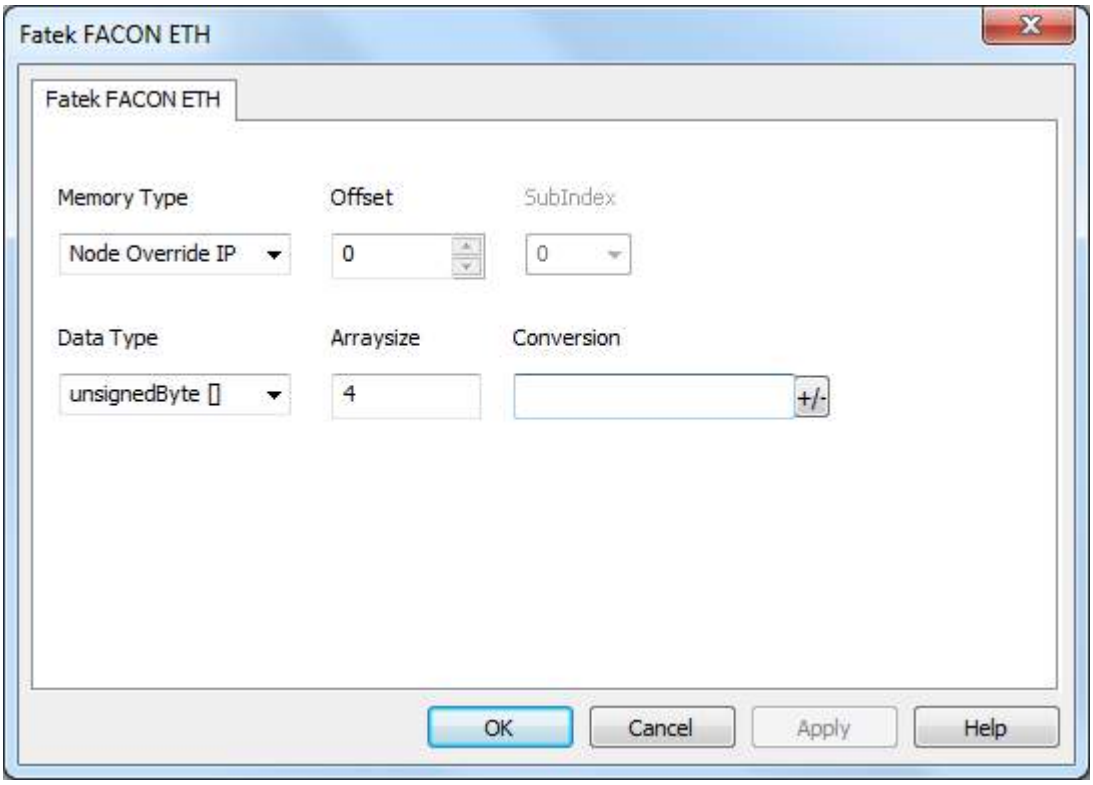
This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the PLC IP specified in the project at programming time.

Node Override IP	Modbus operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one PLC node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.



Fatek FACON SER

The Fatek FACON SER communication driver has been designed to connect HMI devices to a Fatek FACON PLC through Serial connection.

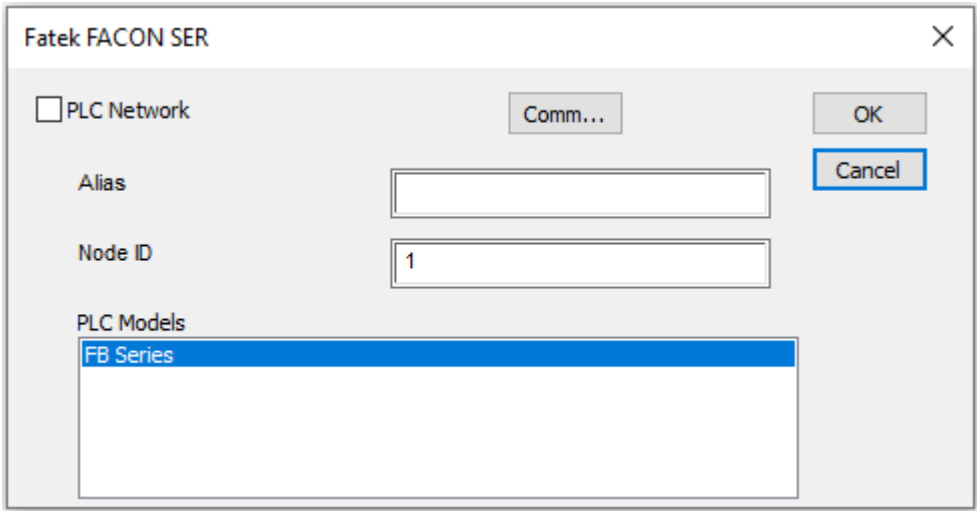
Protocol Editor Settings

Adding a protocol

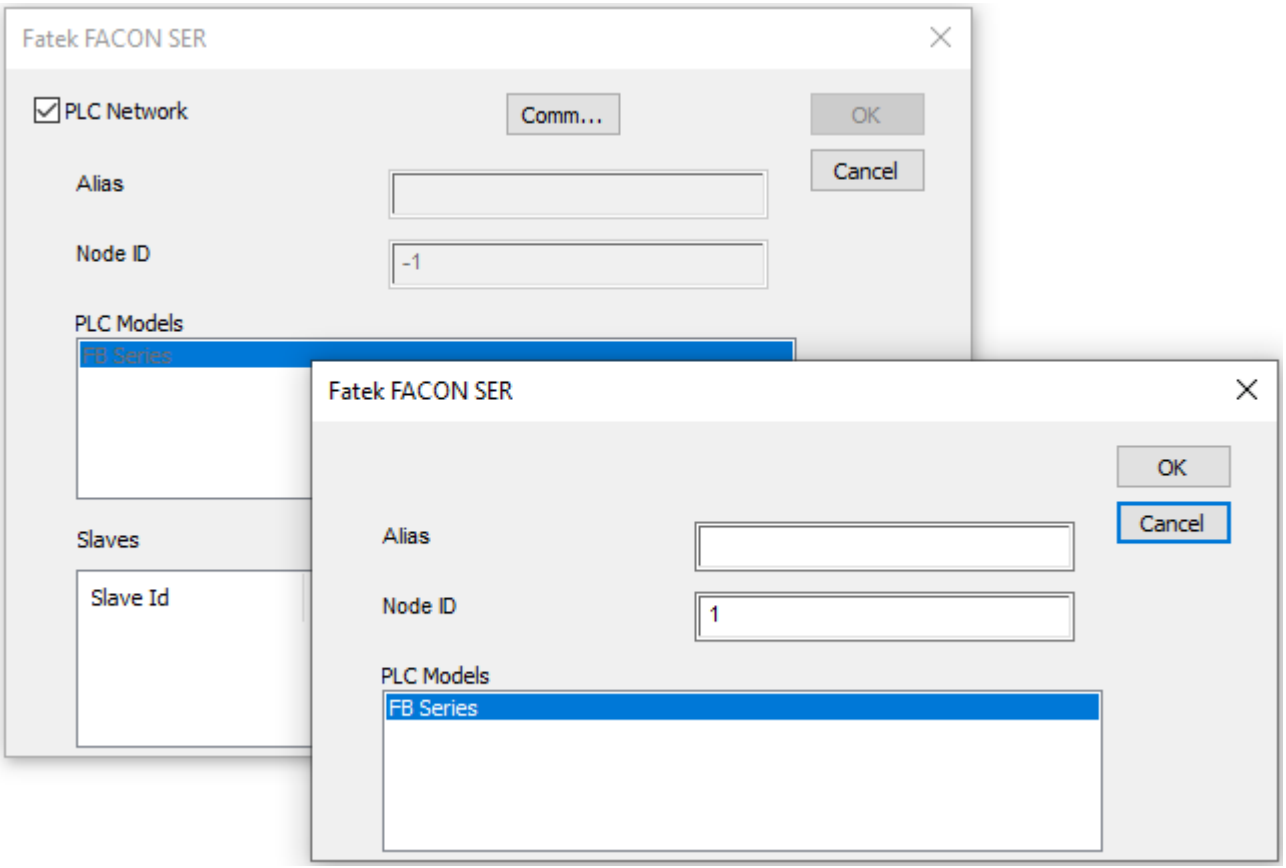
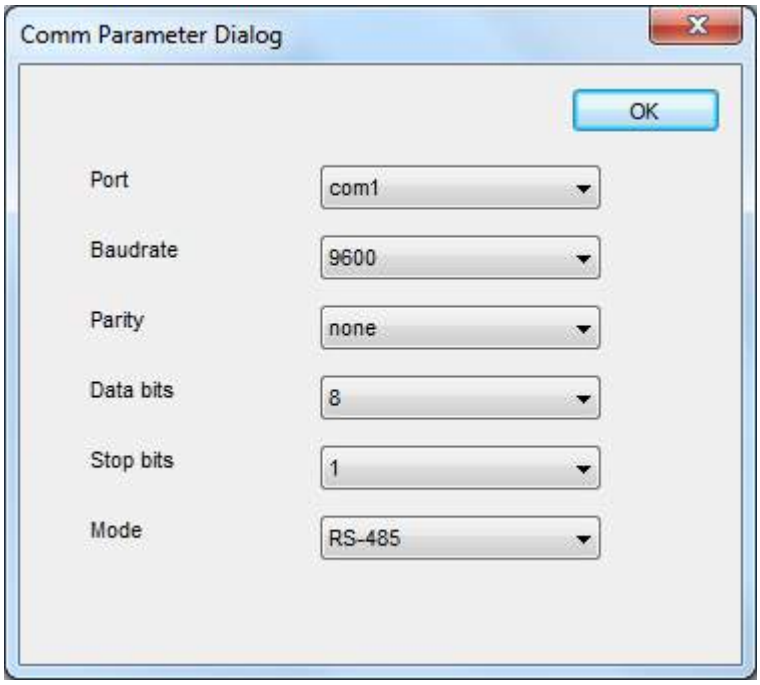
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Node ID	Serial node associated to the PLC.
PLC Models	PLC model available: • FB Series
PLC Network	IP address for all controllers in multiple connections. **PLC Network** must be selected to enable multiple connections.

Element	Description
	
Comm...	If clicked displays the communication parameters setup dialog. 

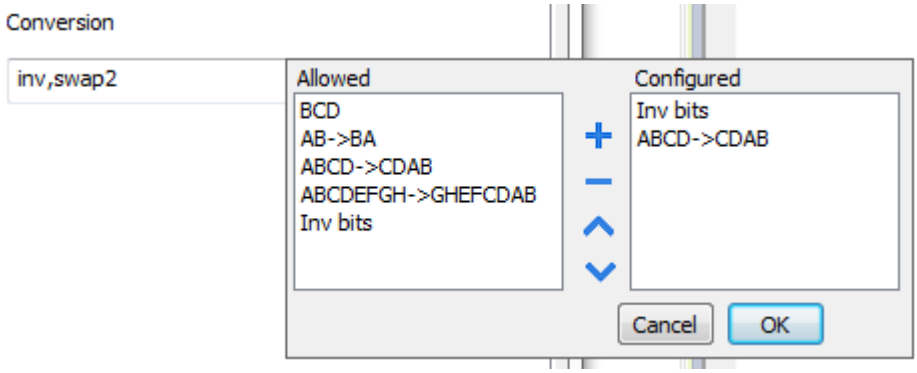
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Element</th><th>Parameter</th></tr>
<tr><td>Port</td><td>Serial port selection.
COM1: device PLC port.
COM2: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board.
COM3: optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.</td></tr>
<tr><td>Baudrate, Parity, Data Bits, Stop bits</td><td>Serial line parameters.</td></tr>
<tr><td>Mode</td><td>Serial port mode. Available modes:
RS-232.
RS-485 (2 wires).
RS-422 (4 wires).</td></tr>
</table>
</td></tr>
</table>

Tag Editor Settings

In Tag Editor select the protocol **Fatek FACON SER**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Input Discrete</td><td>X resources. Corresponding to External Digital Input Point.</td></tr>
<tr><td>Output Relay</td><td>Y resources. Corresponding to External Digital Output Point.</td></tr>
<tr><td>Internal Relay</td><td>M resources. Corresponding to PLC internal memory.</td></tr>
<tr><td>Step Relay</td><td>S resources.</td></tr>
<tr><td>Timer Discrete</td><td>T resources.</td></tr>
<tr><td>Counter Discrete</td><td>C resources.</td></tr>
<tr><td>Timer Register</td><td>Current Time Value Register.</td></tr>
<tr><td>Counter Register</td><td>Current Counter Value Register.</td></tr>
<tr><td>Data Register - HR</td><td>R resources.</td></tr>
<tr><td>Data Register - DR</td><td>D resources.</td></tr>
<tr><td>Run</td><td>Boolean value. Corresponding to PLC status.</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>Starting address for the Tag. The possible range depend on PLC model selected.</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection depending on the selected data type.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
double
string
binary
See "Programming concepts" section in the main manual.
  Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).</td></tr>
</table>

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH	**swap4**: Swap bytes in a double word.

Element	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>-> GHEFCDAB</td><td>Example: 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)</td></tr><tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 0001 → 1 10000011100 10101010000101000101101101101100101101100001001111101 (in binary format)</td></tr><tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9) Example: 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)</td></tr></table> Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

GE Intelligent Platforms SNP

The GE Intelligent Platforms SNP driver can be used to connect the HMI device to the GE controllers through serial connection using the native and proprietary SNP communication protocol.

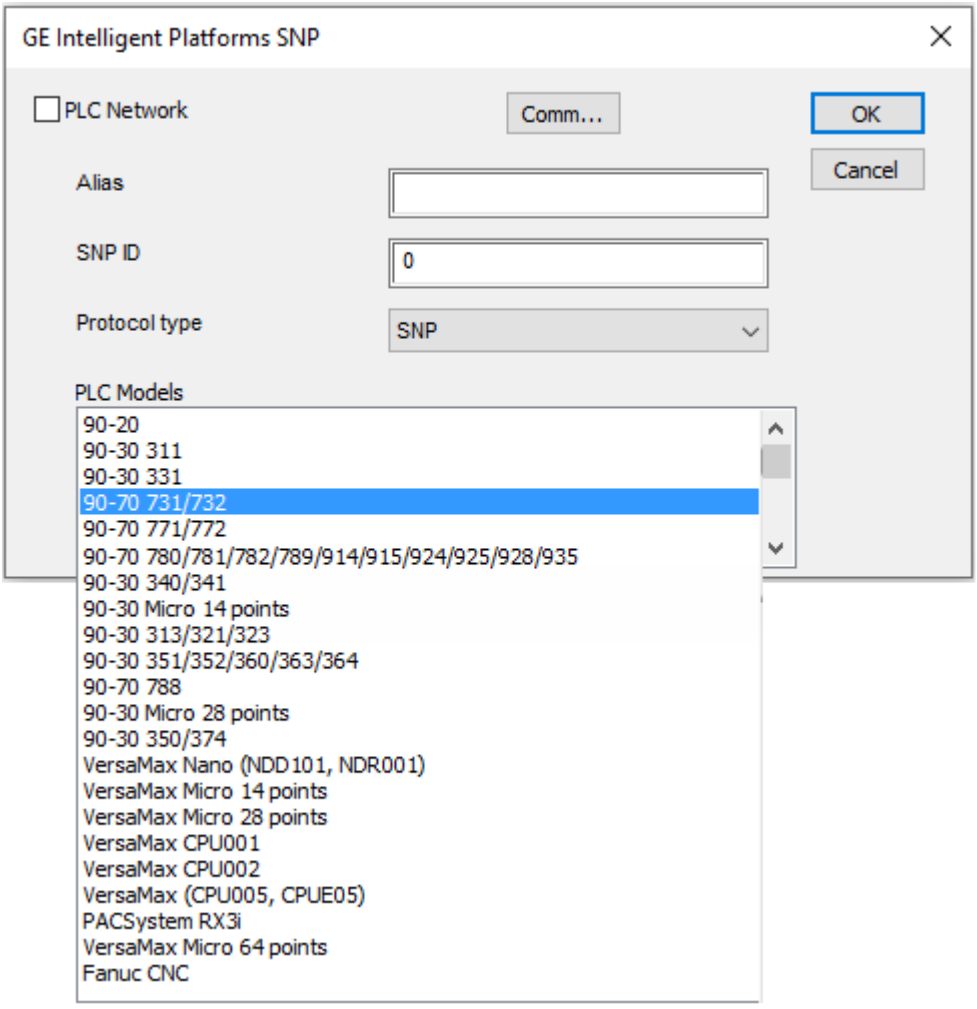
Protocol Editor Settings

Adding a protocol

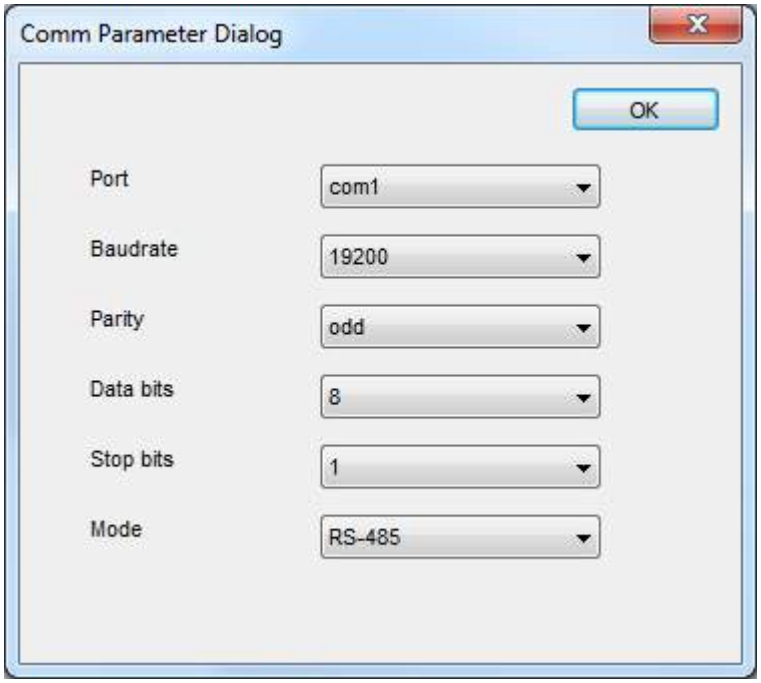
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

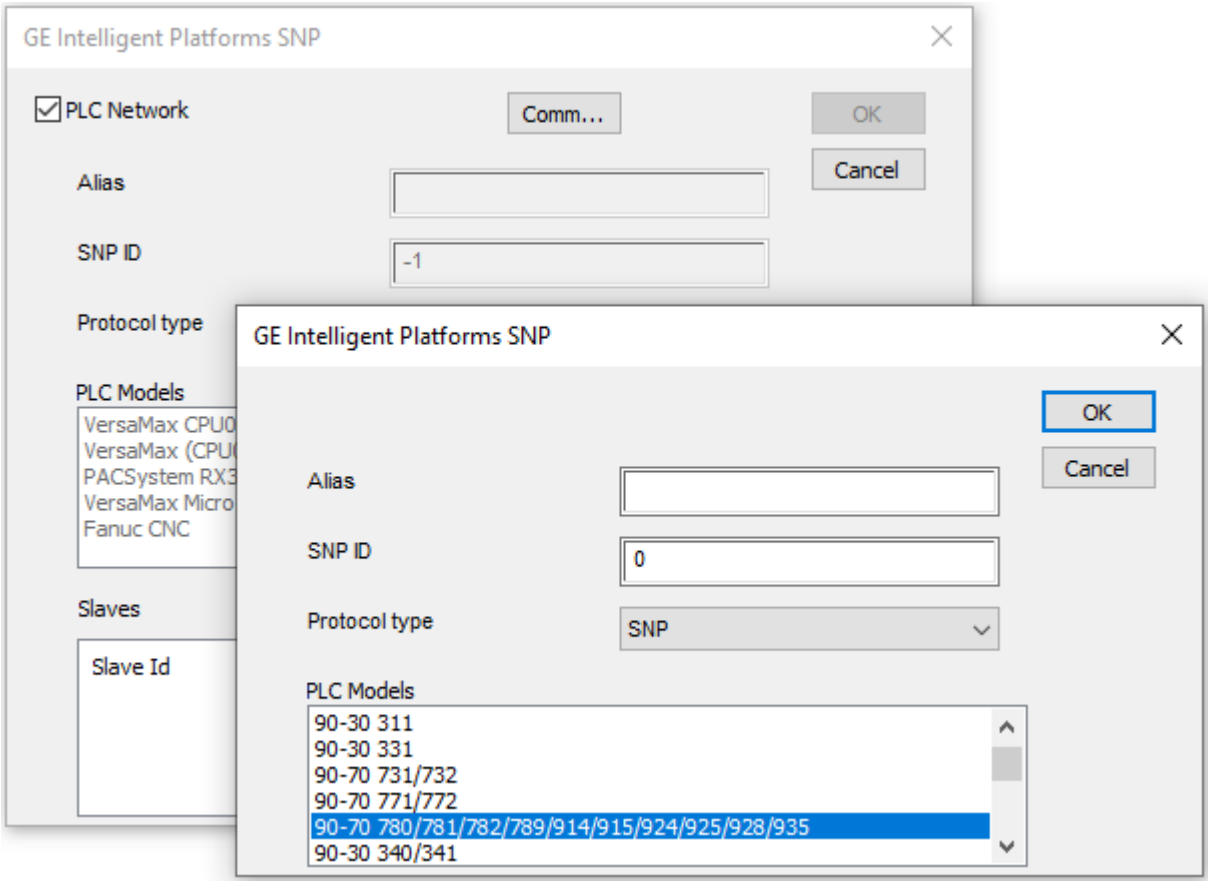
The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
PLC Models	PLC models available.

Element	Description
Protocol type	Allows to select between SNP and SNP-X protocol.
Comm...	If clicked displays the communication parameters setup dialog. 

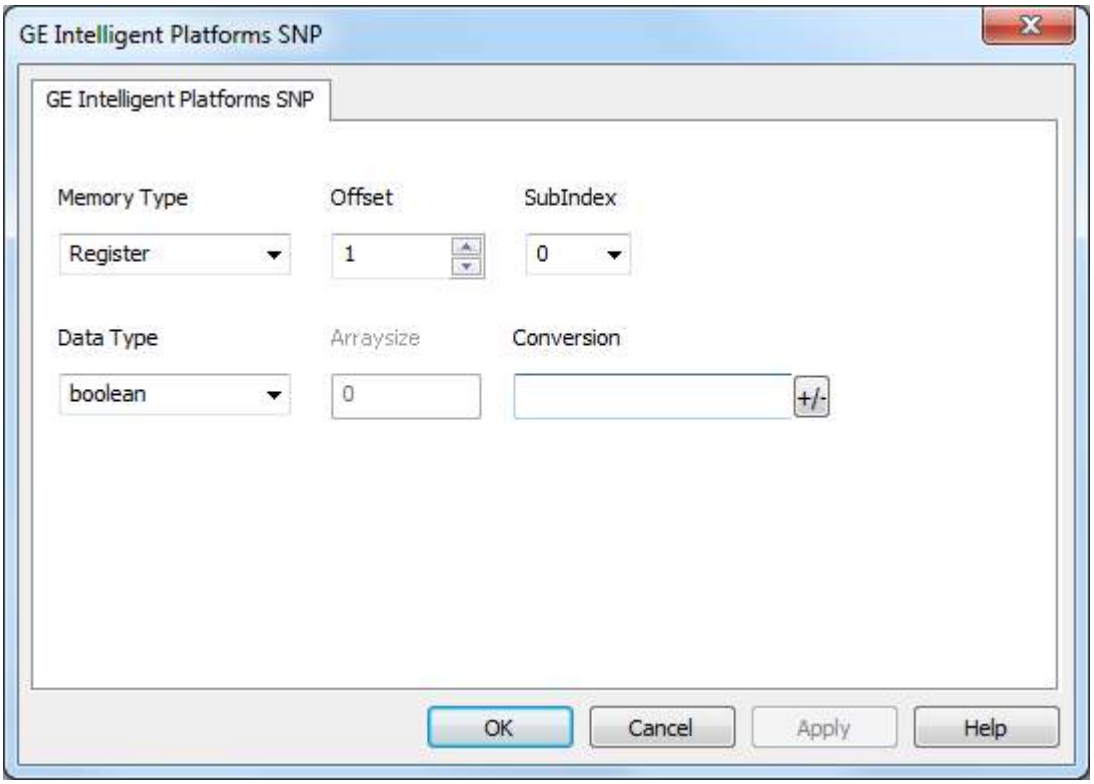
Element	Parameter
Port	Serial port selection. • **COM1**: On-board port • **COM2**: Optional Plug-in module plugged on slot#1 or slot#2 • **COM3**: Optional Plug-in module plugged on slot#3 or slot#4
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Element</th><th>Parameter</th></tr>
<tr><td>Mode</td><td>Serial port mode. Available modes:
RS-232.
RS-485 (2 wires).
RS-422 (4 wires).</td></tr>
<tr><td>PLC Network</td><td>Multiple controllers can be connected to one HMI device. To set-up multiple connections, select PLC networkand click Add to configure each slave
 
</td></tr>
</table>
</td></tr>
</table>

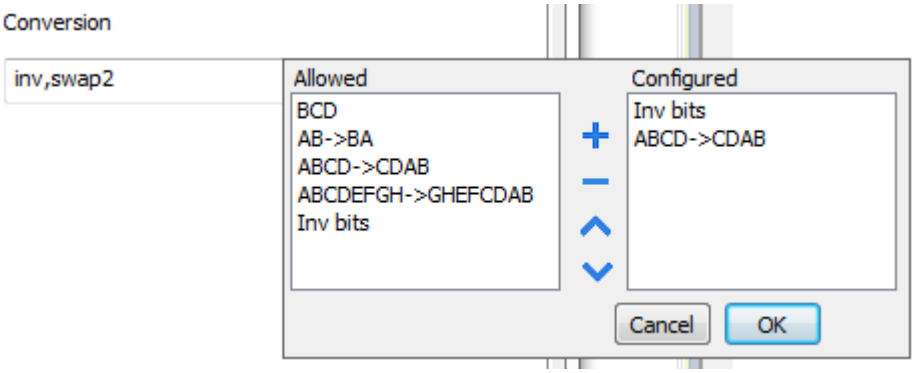
Tag Editor Settings

In Tag Editor select the protocol **GE Intelligent Platforms SNP**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:



Element	Description	
Memory Type	**Memory Type**	**Description**
	Register	**R** resource on PLC.
	Discrete Input	**I** resource on PLC.
	Discrete Output	**Q** resource on PLC.
	Discrete Global	**G** resource on PLC.
	Internal Coil	**M** resource on PLC.
	Temporary Coil	**T** resource on PLC.
	System Status	**S** resource on PLC.
	Analog Input	**AI** resource on PLC.
	Analog Output	**AQ** resource on PLC.
	Clear I/O Fault	**IOF** resource on PLC.
	Clear PLC Fault	**PLF** resource on PLC.
Offset	Offset address where tag is located. Offset range depends on specific memory type and PLC model selected.	

Element	Description
Data Type	Available data types: • **boolean** • **byte** • **short** • **int** • **unsignedByte** • **unsignedShort** • **unsignedInt** • **float** • **double** • **string** • **binary** See "Programming concepts" section in the main manual. Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 0001 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

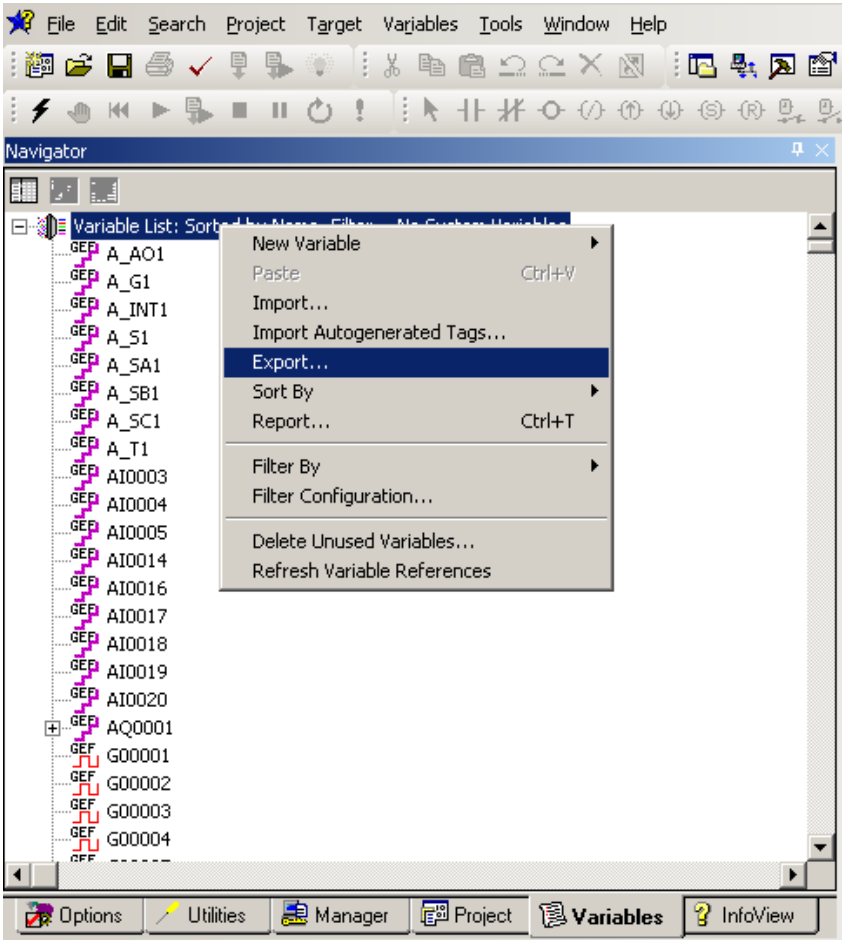
Tag Import

Exporting Tags from PLC

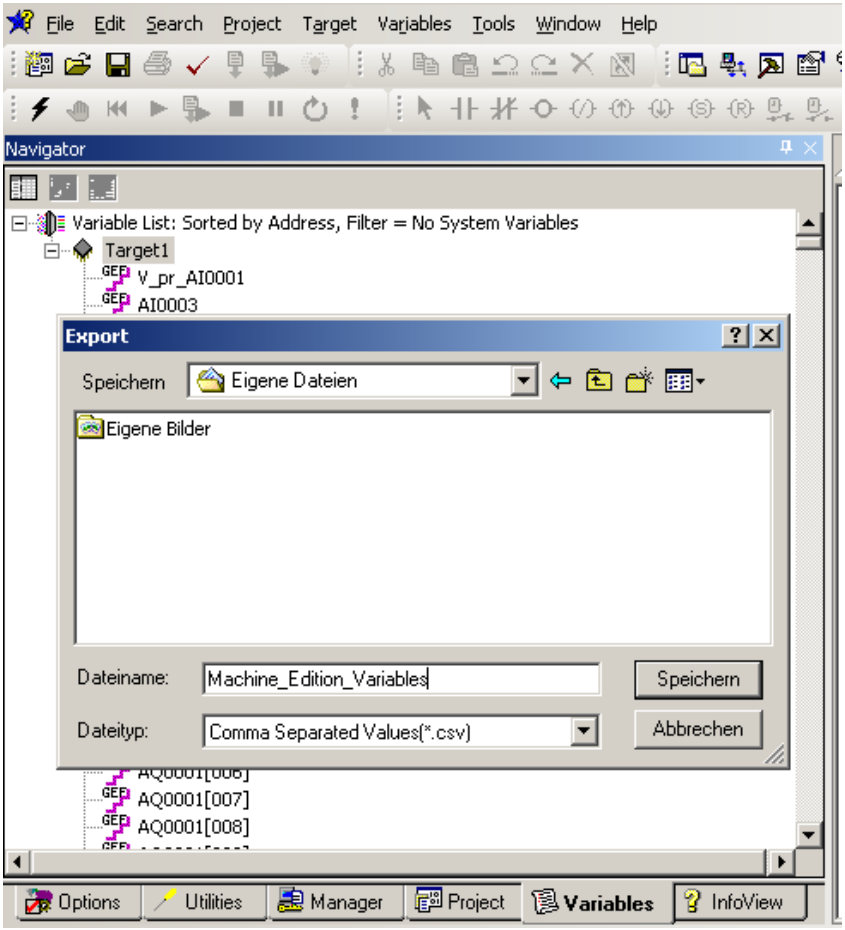
The GE Intelligent Platforms SRTP Ethernet driver support the Tag Import facility.

Variables can be exported by the controller programming software Proficy Machine Edition,

selecting "Variables" tab, then right mouse click and from context menu select the Export option as shown in following figure.

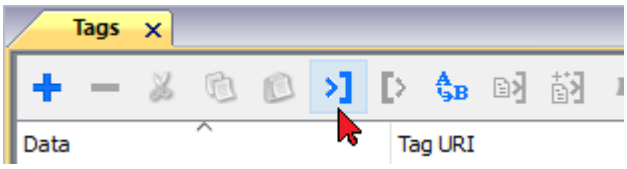


In the following dialog select then the file name and the file location on the computer.

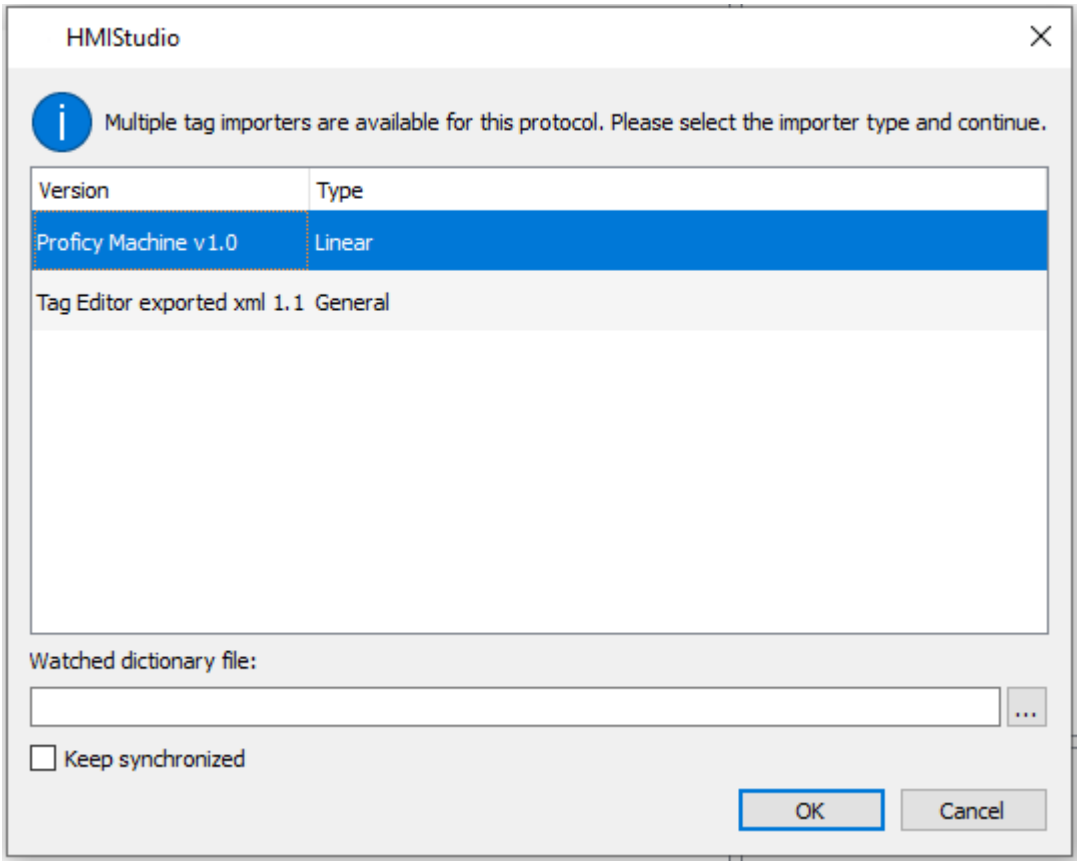


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



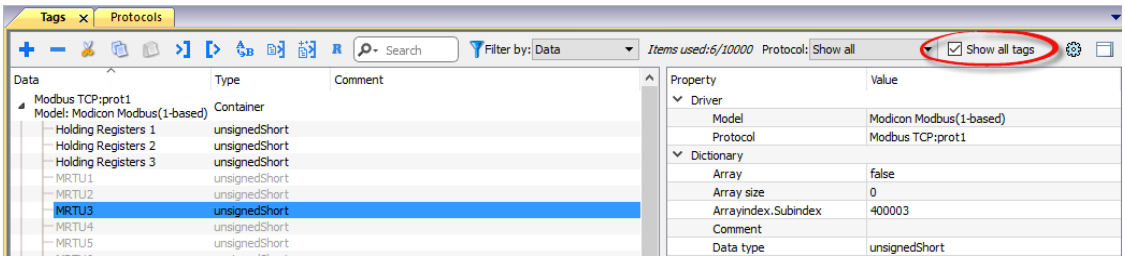
The following dialog shows which importer type can be selected.

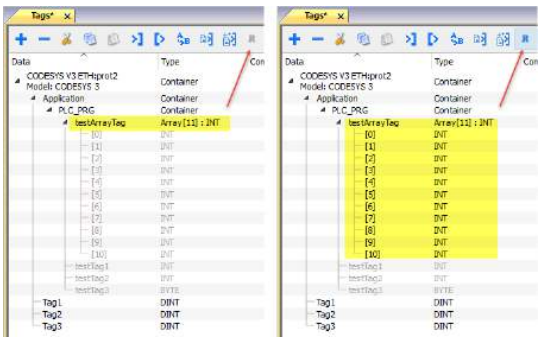
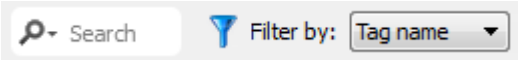


Importer	Description
Proficy Machine v1.0 Linear	Requires an **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



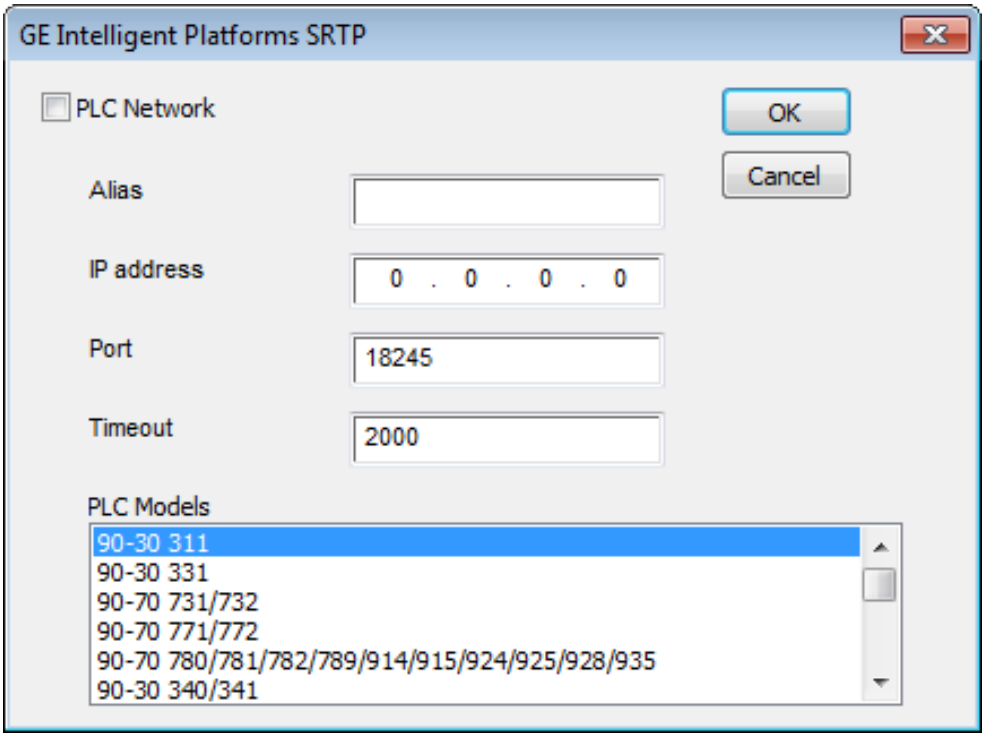
Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

GE Intelligent Platforms SRTP

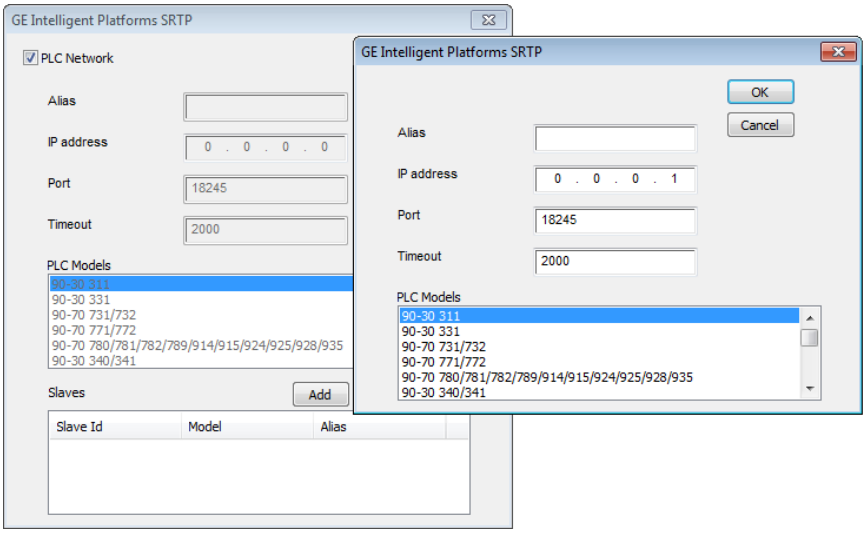
The GE Intelligent Platforms SRTP driver can be used to connect the HMI device to the GE controllers through Ethernet connection using the native and proprietary SRTP communication protocol.

Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “GE Intelligent Platforms SRTP” from the list of available protocols.



Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
IP Address	The IP address of the Ethernet interface of the controller
Port	Communication Port number for the Ethernet interface
Timeout	The time the protocol waits the answer from the controller before issuing a new retry.

Element	Description
PLC Models	List of compatible controller models. Make sure to select the right model in this list when configuring the protocol.
PLC Network	The protocol supports connection to multiple controllers. To enable this, check the "PLC Network" check box and provide the configuration per each node. 

Data Types

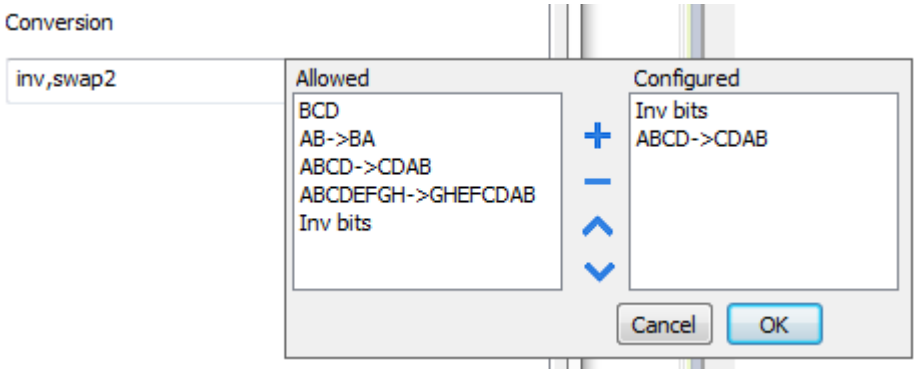
The import module supports variables of standard data types as per the following list.

- BOOL
- BYTE (8-bits unsigned integers)
- DINT (32-bits signed integers)
- DWORD (32-bit bit strings, displayed as unsigned integers)
- INT (16-bit signed integers)
- REAL (32-bit floating point data)
- STRING (character string)
- UINT (16-bit unsigned integers)
- WORD (16-bit bit strings, displayed as unsigned integers)

Note: User defined structure and predefined structures are not supported. 64-bit data are also not supported

Tag Conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)

Value	Description
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Special Data Types

The GE Intelligent Platforms SRTP driver provides one special data type called "Node Override IP".

The Node Override IP allows changing at runtime the IP address of the target controller you want to connect. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

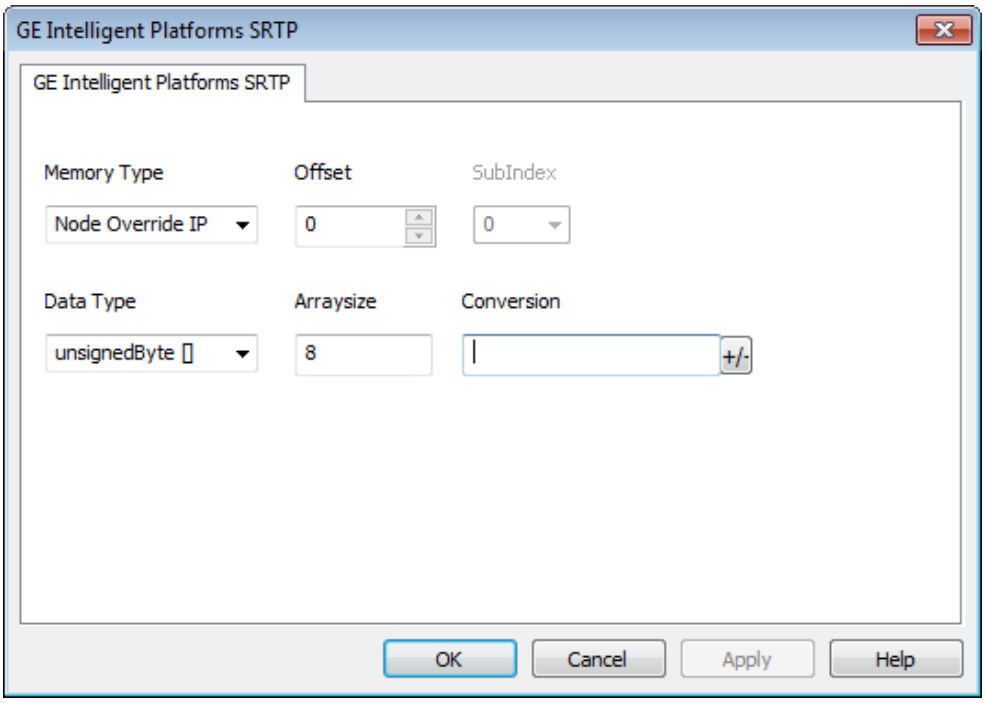
The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

If the IP Override is set to 0.0.0.0, all the communication with the node is stopped, no request frames are generated anymore.

If the IP Override has a value different from 0.0.0.0, it is interpreted as node IP override and the target IP address is replaced at runtime with the new value.

In case the panel has been configured to access to a network of controllers, each node has its own Override variable.

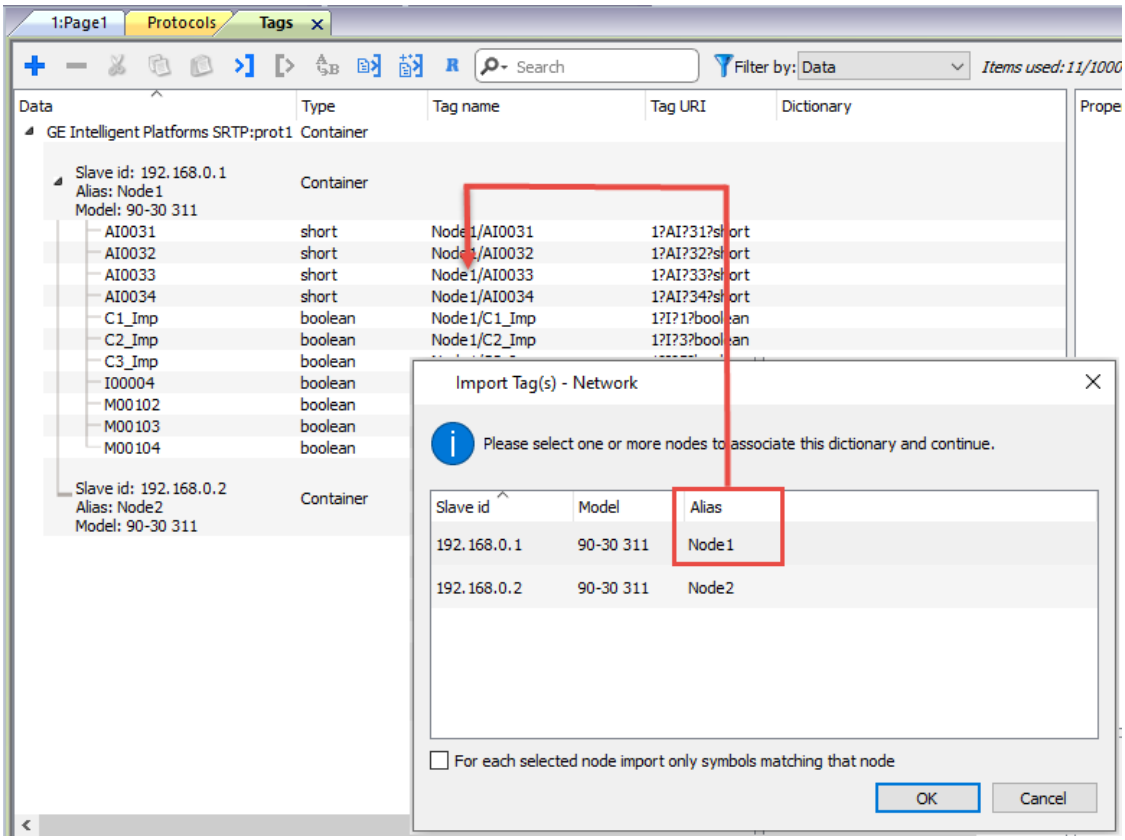
Note: the IP Override values assigned at runtime are retained through power cycles.



Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Note: Aliasing tag names is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name. The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

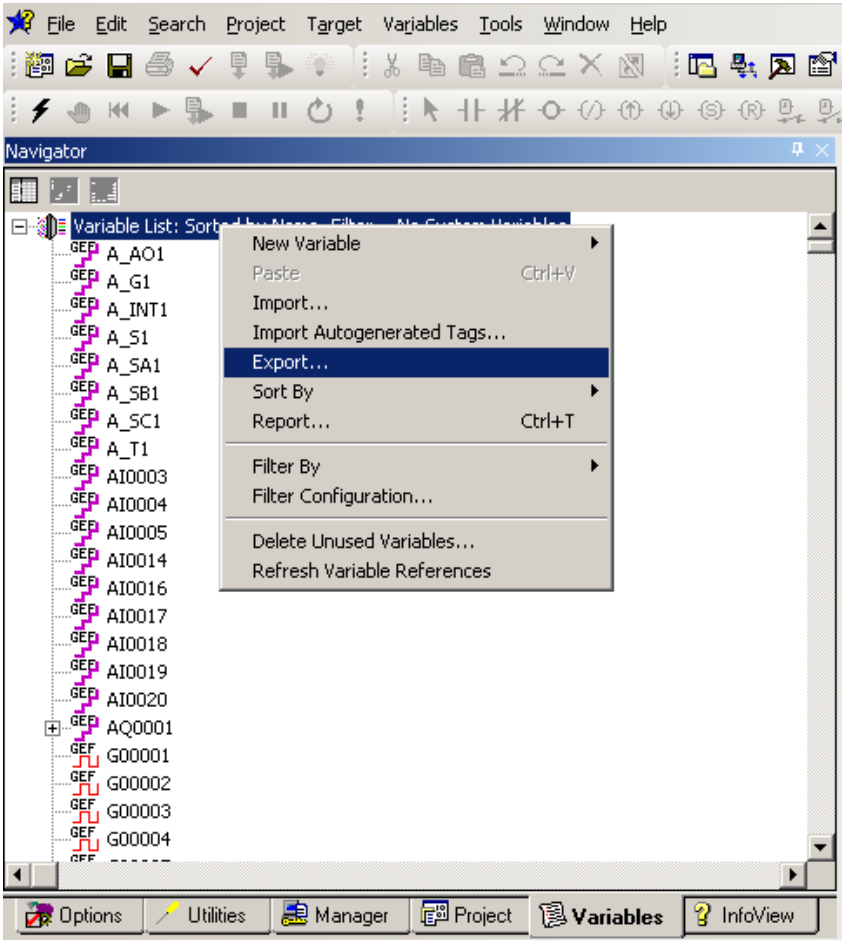
Tag Import

Exporting Tags from PLC

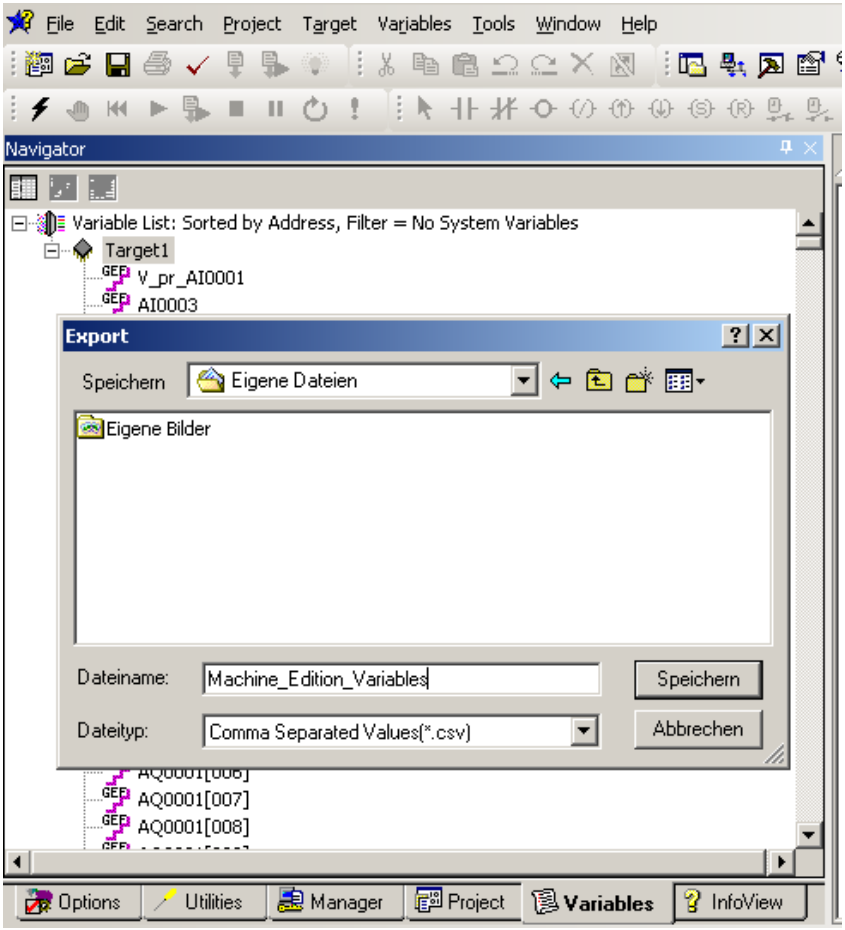
The GE Intelligent Platforms SRTP Ethernet driver support the Tag Import facility.

Variables can be exported by the controller programming software Proficy Machine Edition,

selecting “Variables” tab, then right mouse click and from context menu select the Export option as shown in following figure.

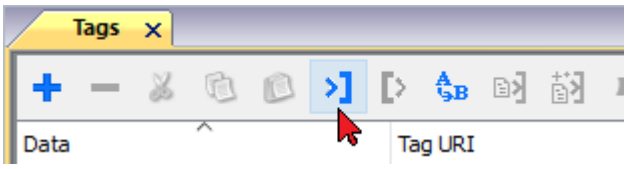


In the following dialog select then the file name and the file location on the computer.

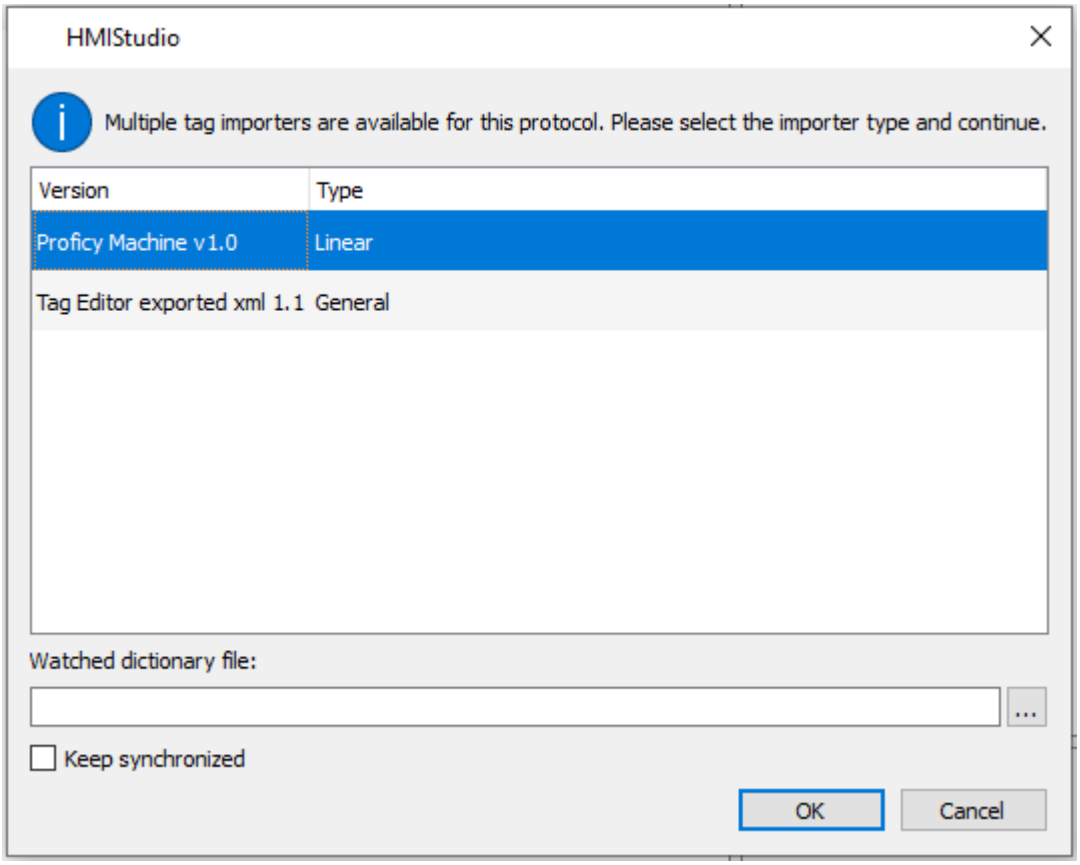


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



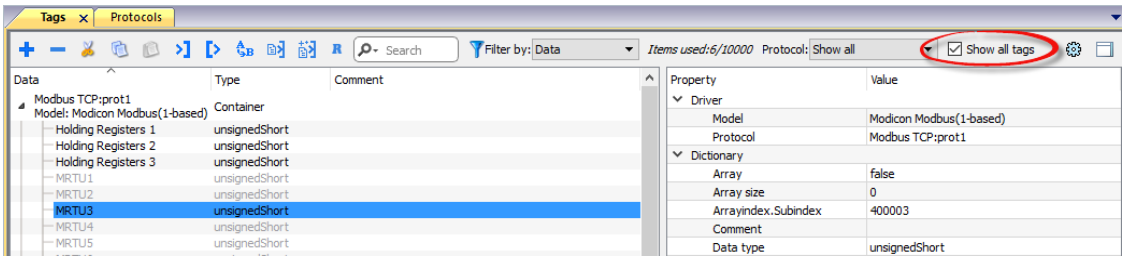
The following dialog shows which importer type can be selected.

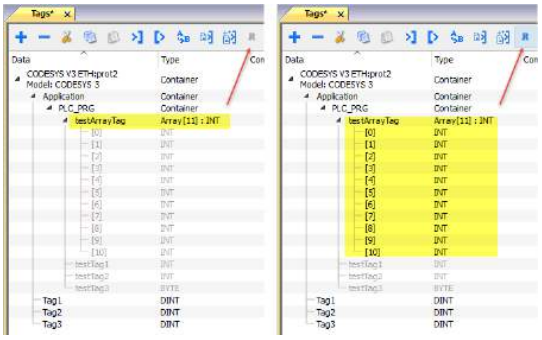
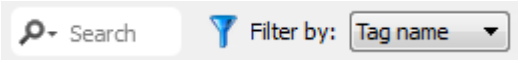


Importer	Description
Proficy Machine v1.0 Linear	Requires an **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The status codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Invalid response	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support

GE SRTP

The GE SRTP communication driver has been designed to connect HMI devices to GE PLCs.

The driver allows symbolic communication with GE PLC model PacSystemRx3i.

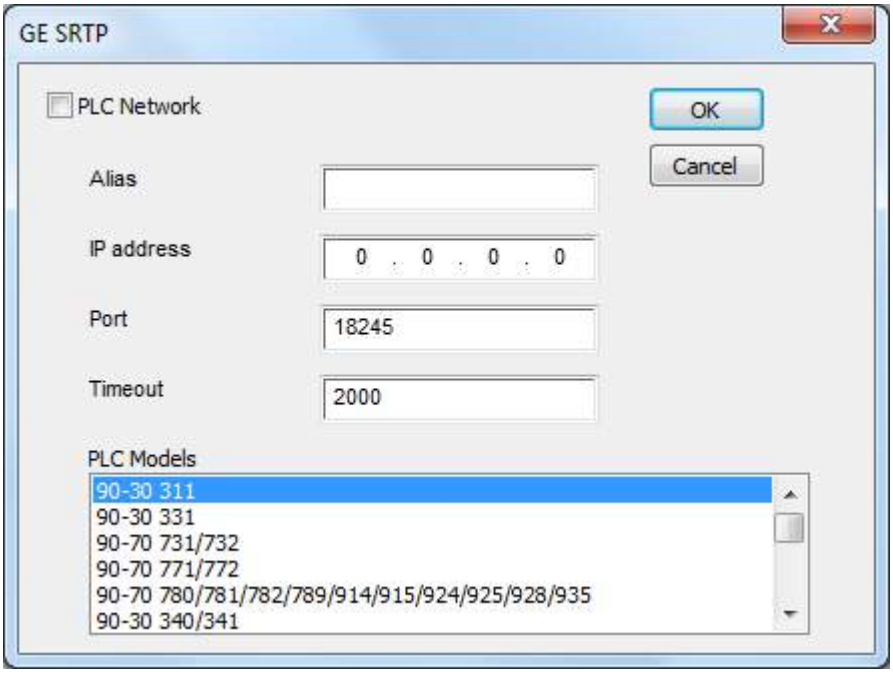
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	Port number used by the driver. The default value is **18245**.
Timeout	Time delay in milliseconds between two retries in case of missing response from the server device.
PLC Models	SAIA PLC models available:

Element	Description
	• **90-30 311** • **90-30 331** • **90-70 731/732** • **90-70 771/772** • **90-70 780/781/782/789/914/915/924/925/928/935** • **90-30 340/341** • **90-30 313** • **90-30 351/352/360/363/364** • **90-70 788** • **90-30 350/374** • **VersaMax CPU001** • **VersaMax CPU002** • **VersaMax (CPU005, CPUE05)** • **PACSystem RX3i**
PLC Network	Multiple controllers can be connected to one HMI device. To set-up multiple connections, select **PLC network**and click **Add** to configure each node

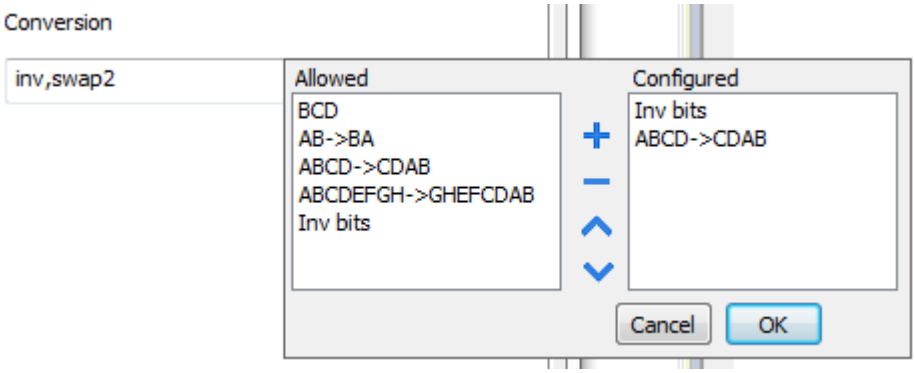
Tag Editor Settings

*Path: **ProjectView*** > ***Config*** > *double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **GE SRTP** from the **Driver** list: tag definition dialog is displayed.


GE SRTP
GE SRTP
Memory Type
Offset
SubIndex
Register
1
0
Symbol Name
Data Type
Arraysize
unsignedShort
0
Conversion
+/-
OK
Cancel
Apply
Help

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Register</td><td>unsigned 16 bit data register (default)</td></tr>
<tr><td>Discrete Input</td><td>1 bit data input (default)</td></tr>
<tr><td>Discrete Output</td><td>1 bit data output (default)</td></tr>
<tr><td>Discrete Global</td><td>1 bit data global (default)</td></tr>
<tr><td>Internal Coil</td><td>1 bit data coil (default)</td></tr>
<tr><td>Temporary Coil</td><td>1 bit data coil (default)</td></tr>
<tr><td>System Status</td><td>1 bit data status</td></tr>
<tr><td>System Status A</td><td>1 bit data status</td></tr>
<tr><td>System Status B</td><td>1 bit data status</td></tr>
<tr><td>System Status C</td><td>1 bit data status</td></tr>
<tr><td>Analog Input</td><td>unsigned 16 bit data input (default)</td></tr>
<tr><td>Analog Output</td><td>unsigned 16 bit data output (default)</td></tr>
<tr><td>SYMBOL</td><td>1 bit data symbol (default)</td></tr>
<tr><td>Node Override IP</td><td>unsigned 8 bit array (see Special Data Types for mode details)</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>This parameter is the address on the physical memory of the controller. The range for any memory type depends on the PLC model.</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
double
string
See "Programming concepts" section in the main manual.
Note: To define arrays, select one of Data Type format followed by square brackets.</td></tr>
</table>

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH ->	**swap4**: Swap bytes in a double word.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>GHEFCDAB</td><td>Example:
32FCFF54 → 54FFFC32 (in hexadecimal format)
855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000001
→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

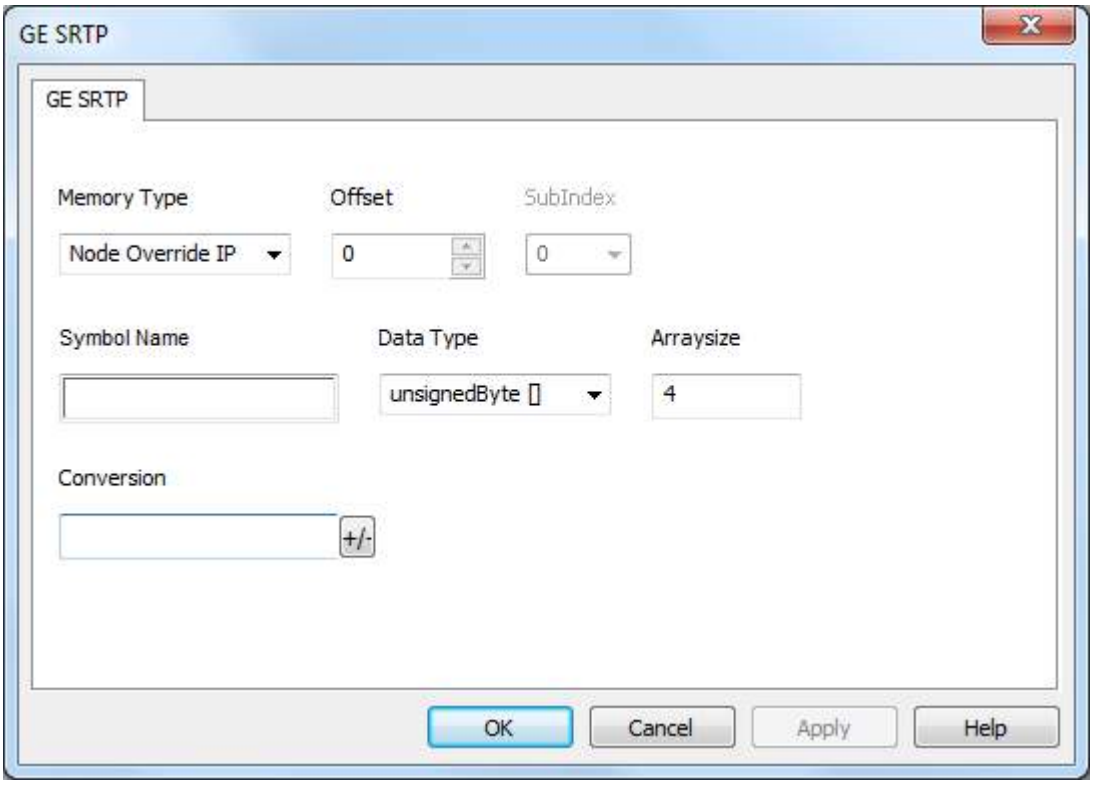
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

Hostname DNS or mDNS

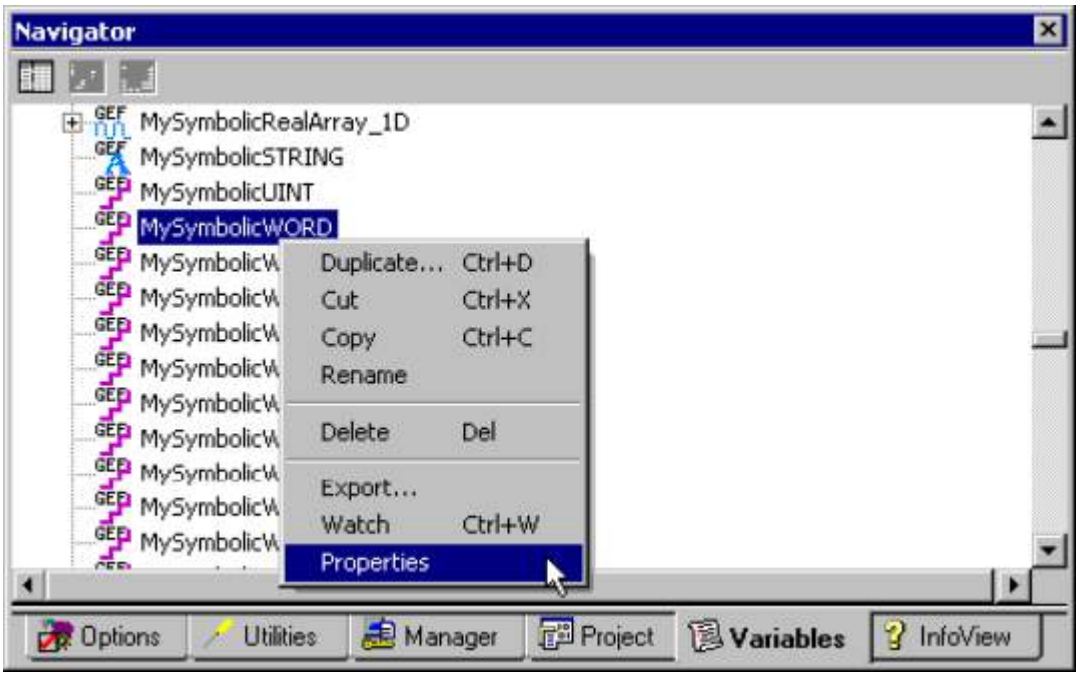
In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.



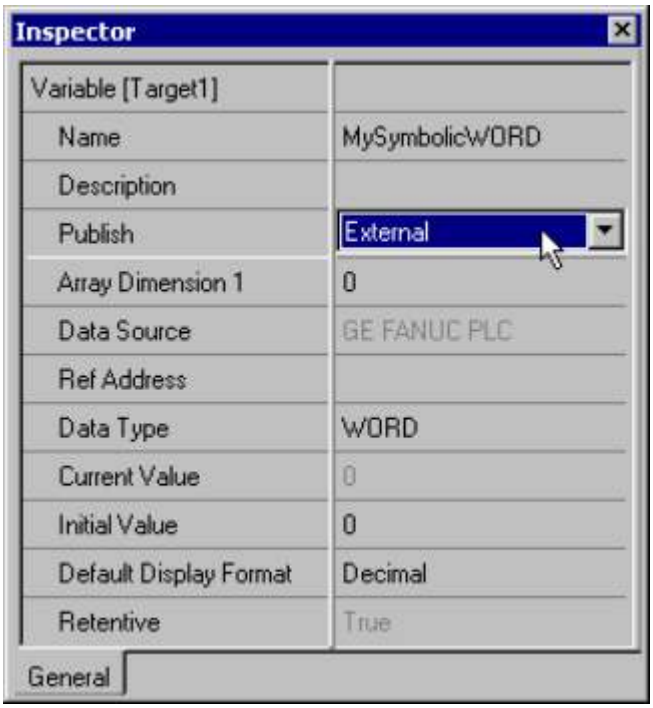
Tag Import

For GE PLC model PacSystemRx3i it is possible to create symbolic variables.

To create a new variable, right-click on the **Variables View** and select **New Variable**. To edit an existing variable, right-click on it and then select **Properties**.

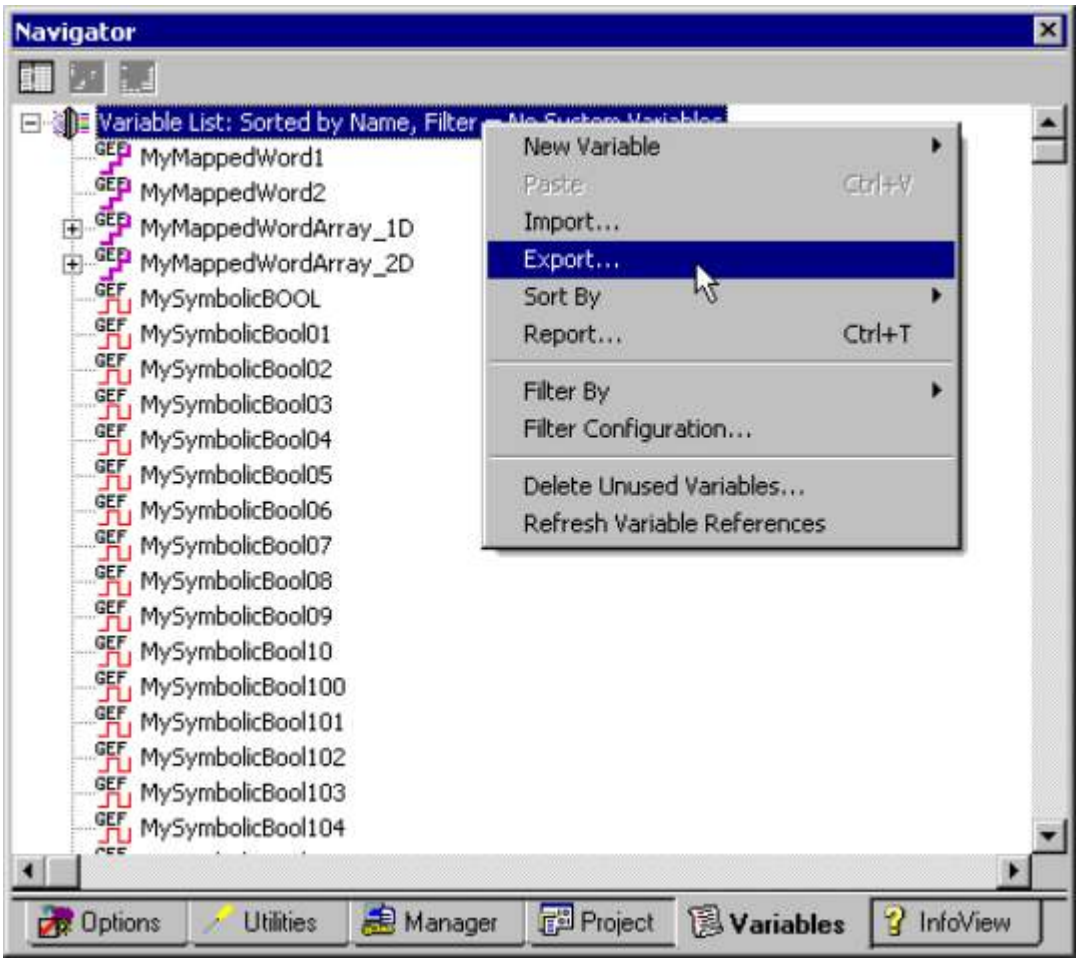


In both cases, the variable's **Properties Inspector** dialog will appear as shown below.

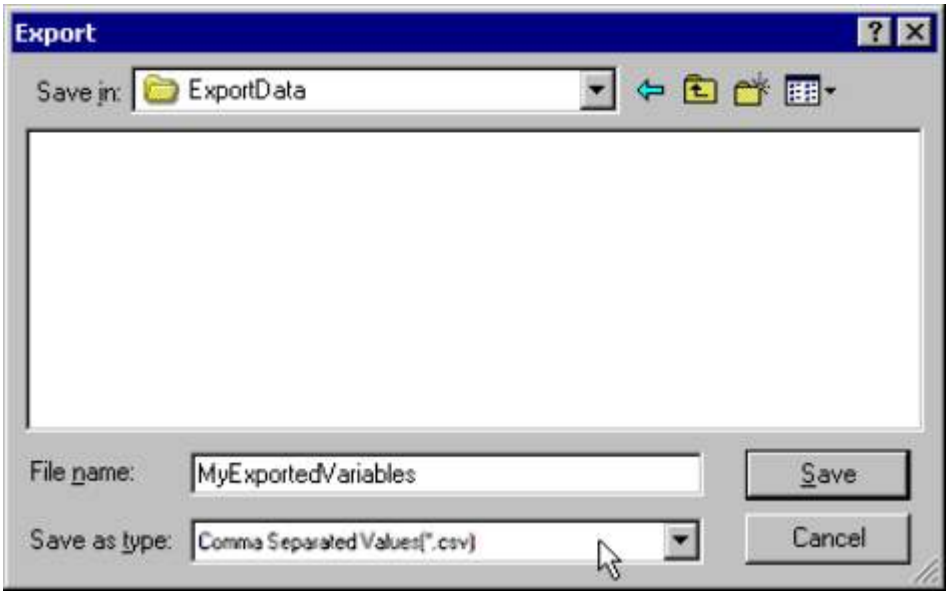


Important: In order for a symbolic variable to be visible to this driver, **Publish** must be set to **External**. The access must be set to **Read/Write**.

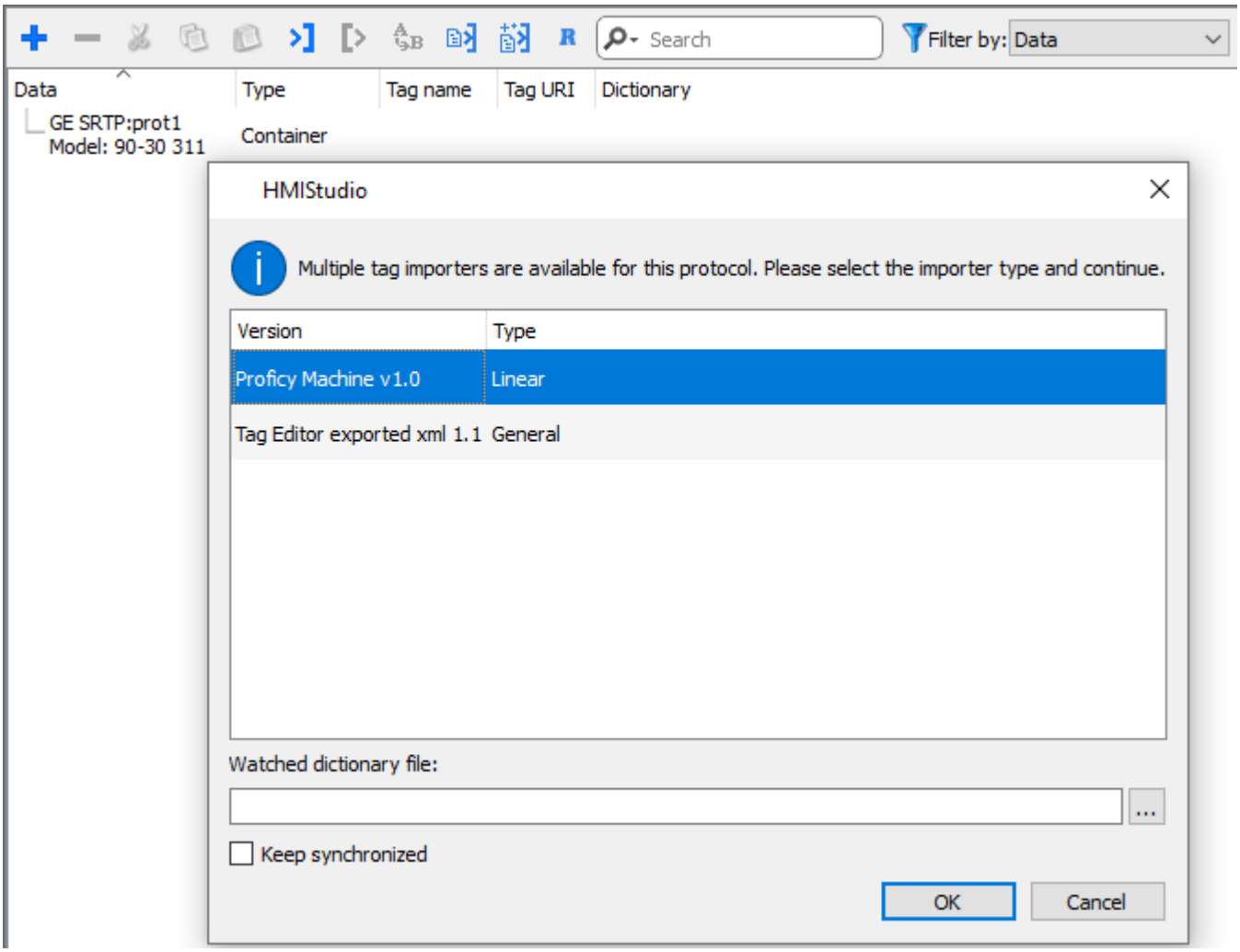
To export these variables from **PACSystem** programming software, right click on **Variable list** (or on selected variables) and click **Export**.



In the **Save as Type** drop-down list, select **Comma Separated Variable (*.csv)** as the export file type. The dialogs should appear as shown below.

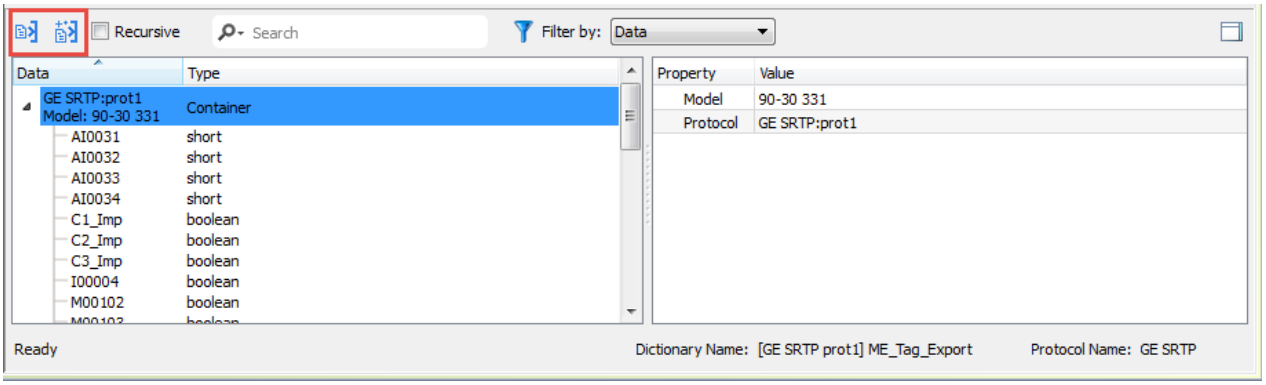


Select the driver in the Studio tag editor and click on the “Import tag” button to start the importer.



Select **Linear** and locate the .csv file, then confirm.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in the following figure.



In case of **Online Changes** performed on PLC side, the tag database must be updated manually to correctly **Read** from PLC.
Write operations do not need a database update.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid response	The device did received a response with invalid format or contents from the controller.	Check if the data programmed in the project are consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

Hitachi SER

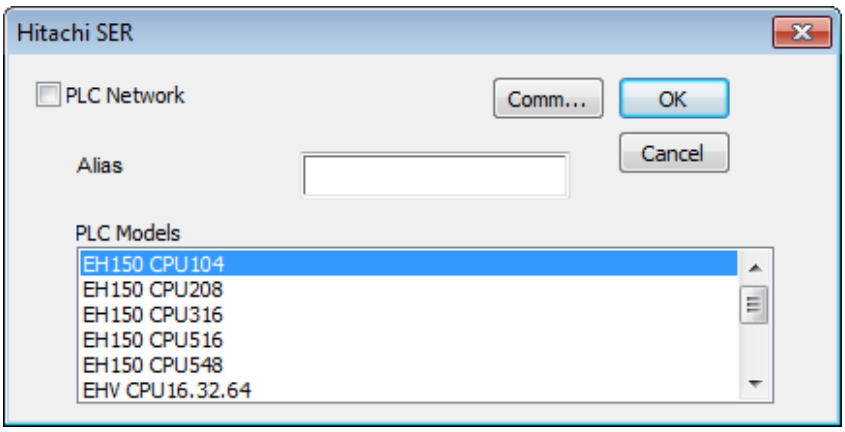
HMI devices can be connected to a Hitachi EH/EHV PLC as the network master using this communication driver.

This driver has been designed for serial connection to the programming port of the PLC.

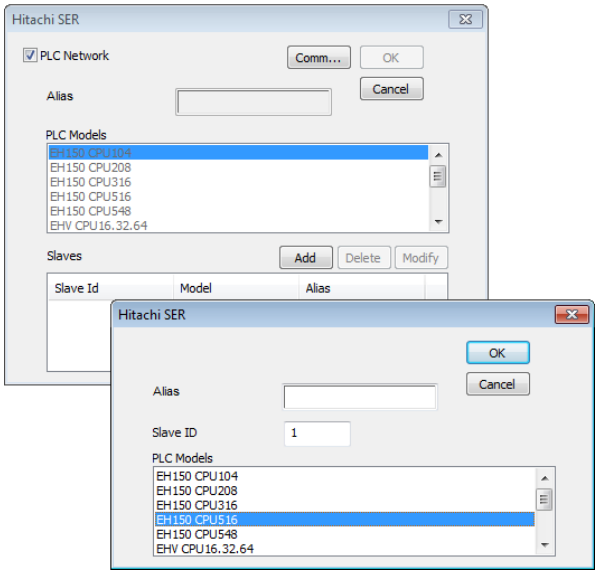
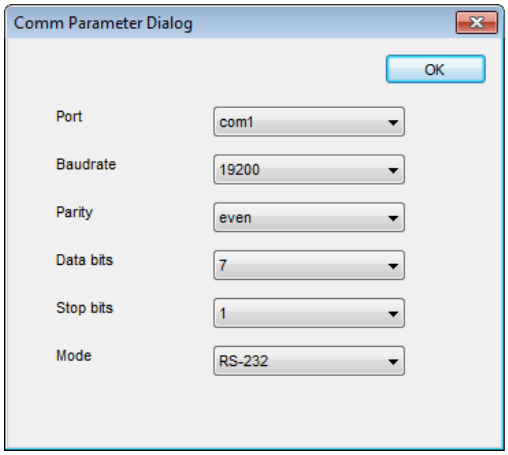
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “Hitachi SER” from the list of available protocols.

The driver configuration dialog box is shown in figure.



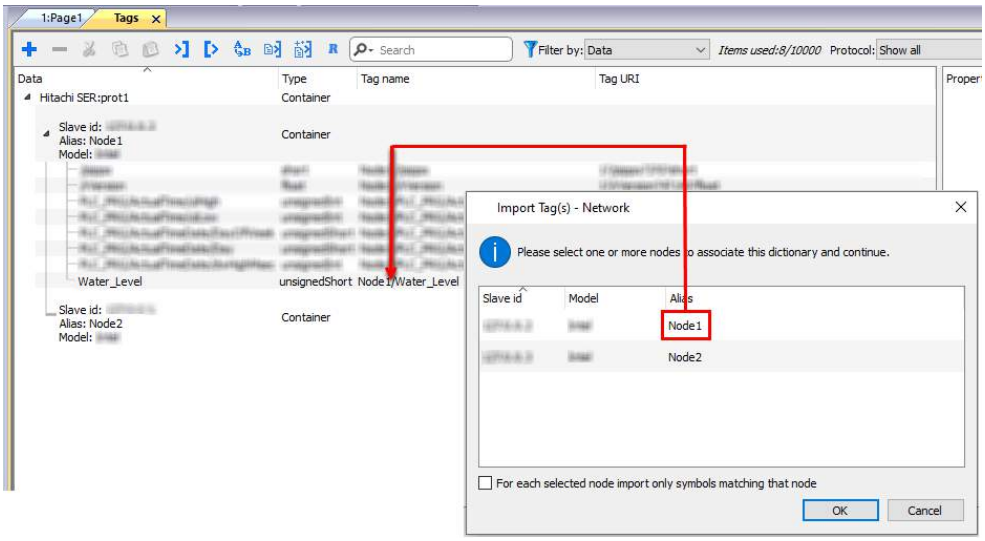
Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
PLC Models	Select from the list the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
PLC Network	The protocol allows the connection of multiple controllers to one HMI. To set-up multiple connections, check “PLC network” checkbox and create the list of controllers pressing the “Add” button. You must specify the node ID for each device you want to connect.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td> 
</td></tr>
<tr><td>Comms.</td><td>Opens the serial port configuration parameters as shown in figure.
 
</td></tr>
<tr><td>Port</td><td>Serial port selection
<table>
<tr><th></th><th>Series 400</th><th>Series 500</th></tr>
<tr><td>com1</td><td>PLC Port</td><td>Serial Port</td></tr>
<tr><td>com2</td><td>PC/Printer Port</td><td>Option Module</td></tr>
</table></td></tr>
<tr><td>Baud rate, Parity, Data bits, Stop bits</td><td>Communication parameters for serial communication</td></tr>
<tr><td>Mode</td><td>Serial port mode; available options:
• RS-232,
• RS-485 (2 wires)
• RS-422 (4 wires)</td></tr>
</table>

Tag Name Aliasing in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.

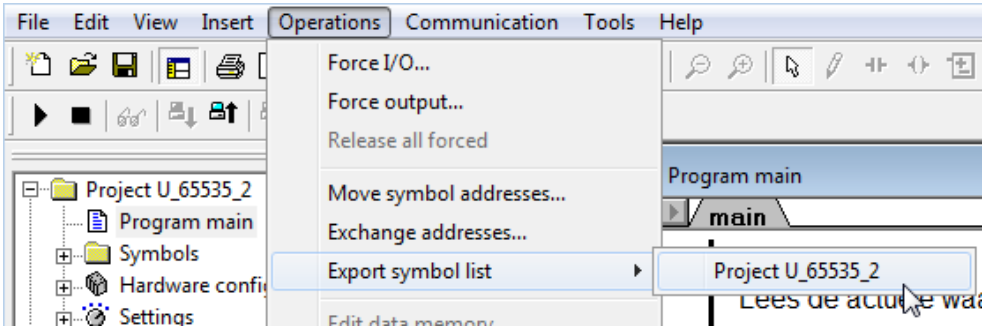


Note: Tag name aliasing is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

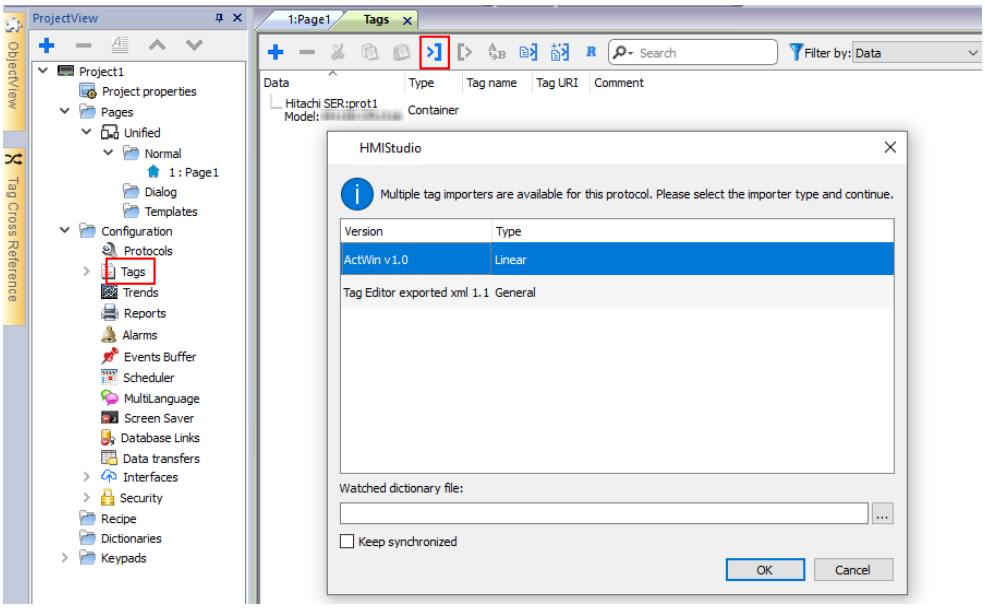
Tag Import

The Hitachi SER communication driver supports importing tags from the PLC programming software. The tag import filter accepts symbol files with extension “.txt” created by the Actwin-H programming tool.

In the Actwin-H Software, click on the menu “Operations” then “Export symbol list” and then select the project which should be exported as shown in figure.

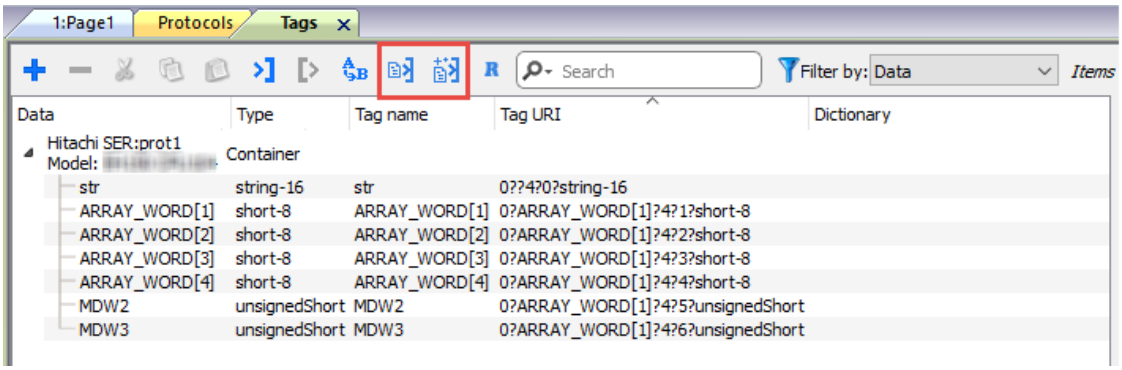


In the Tag Editor select the driver and click on the “Import tag” button to start the importer



Once the importer has been selected, locate the symbol file and click Open.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in figure.



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured for communication

Error	Notes
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits); ensure the communication parameter settings of the controller is compatible with panel communication setup
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources

Hitachi ETH

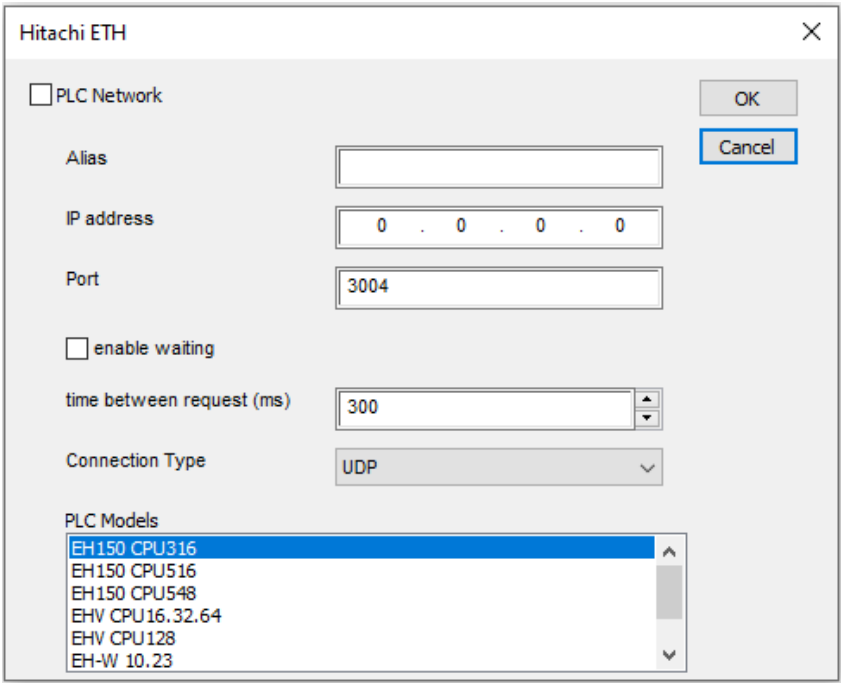
This communication driver has been designed to support communication to Hitachi controllers with Ethernet connection. Hitachi controllers must either have an on-board Ethernet port (EHV CPU) or be equipped with an appropriate Ethernet interface (EH-ETH, ET-ETH2 or OB- ETH).

The communication driver supports both TCP/IP and UDP/IP communication protocols.

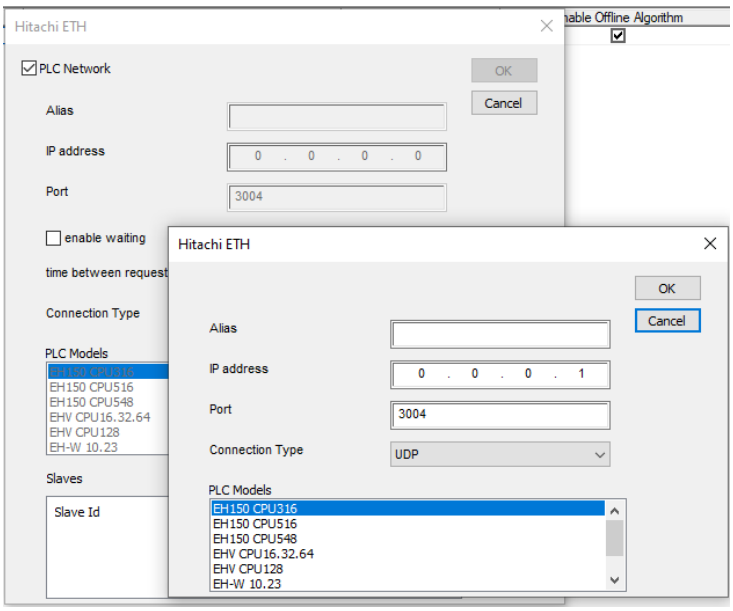
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “Hitachi ETH” from the list of available protocols.

The driver configuration dialog is shown in figure.



Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
IP address	Ethernet IP address of the controller
Port	Port number used for the communication. Default value 3004 and it corresponds to the default setting of Hitachi controllers.
Enable waiting	Introduces a wait time between two communication requests
Time between request	Wait time between two requests if enable waiting option has been activated

Element	Description
Connection type	UDP: use communication based on UDP/IP protocol TCP: use communication based on TCP/IP protocol
PLC Models	Select from the list the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
PLC Network	To set-up multiple connections, check “PLC network” checkbox and create the list of controllers pressing the “Add” button. The IP address for each device you want to connect must be specified. 

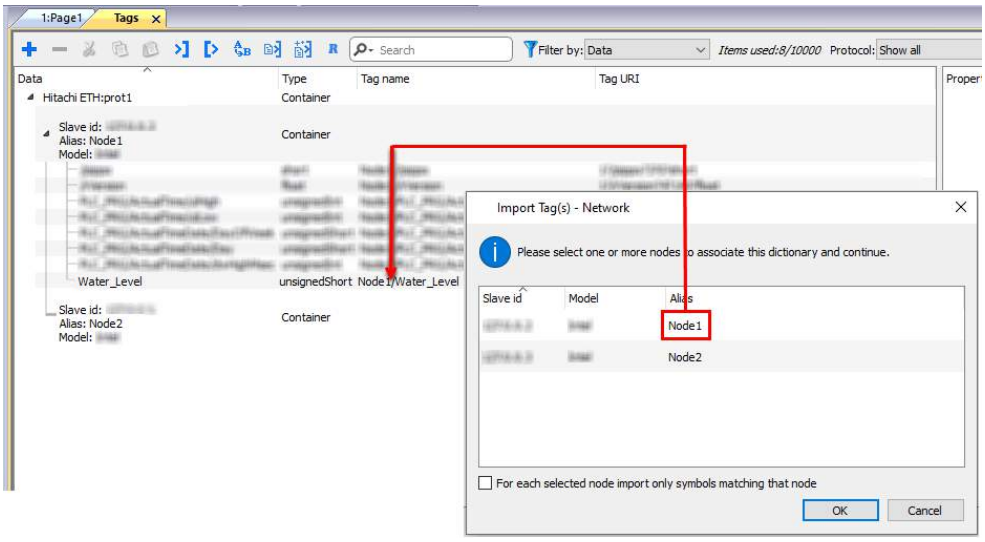
Controller Configuration

The PLC must to be properly configured to support either UPD/IP or TCP/IP communication using port numbers 3004, 3005, 3006 or 3007.

Tag Name Aliasing in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.

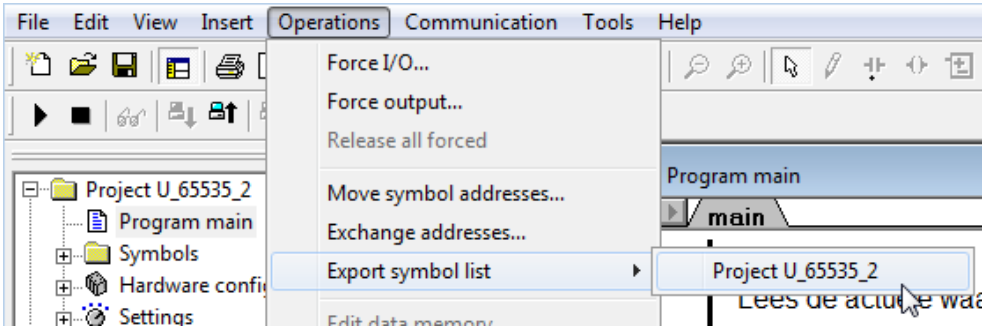


Note: Tag name aliasing is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

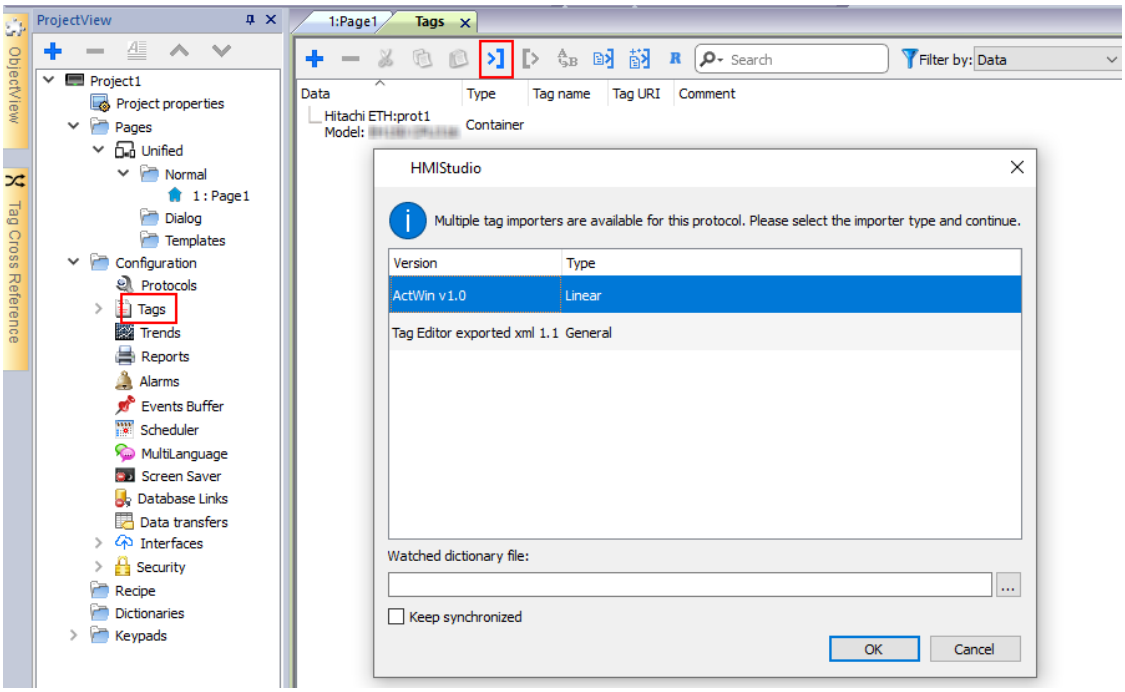
Tag Import

The Hitachi ETH communication driver supports importing tags from the PLC programming software. The tag import filter accepts symbol files with extension “.txt” created by the Actwin-H programming tool.

In the Actwin-H Software, click on the menu “Operations” then “Export symbol list” and then select the project which should be exported as shown in figure.

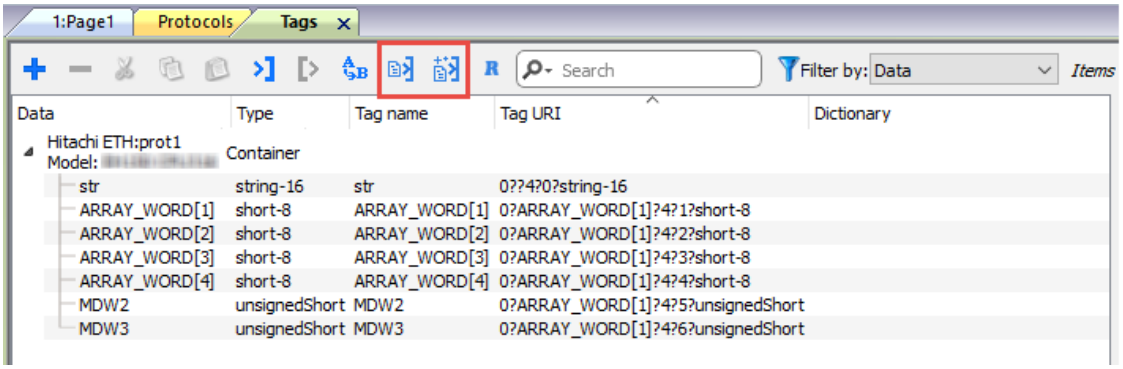


In the tag editor select the driver and click on the “Import tag” button to start the importer



Once the importer has been selected, locate the symbol file and click Open.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in figure.



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured for communication
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller

Error	Notes
	resources
General Error	Error cannot be identified; should never be reported; contact technical support

IDEC Maintenance

IDEC Maintenance communication driver has been designed to connect HMI devices to IDEC PLC through Serial or Ethernet connection.

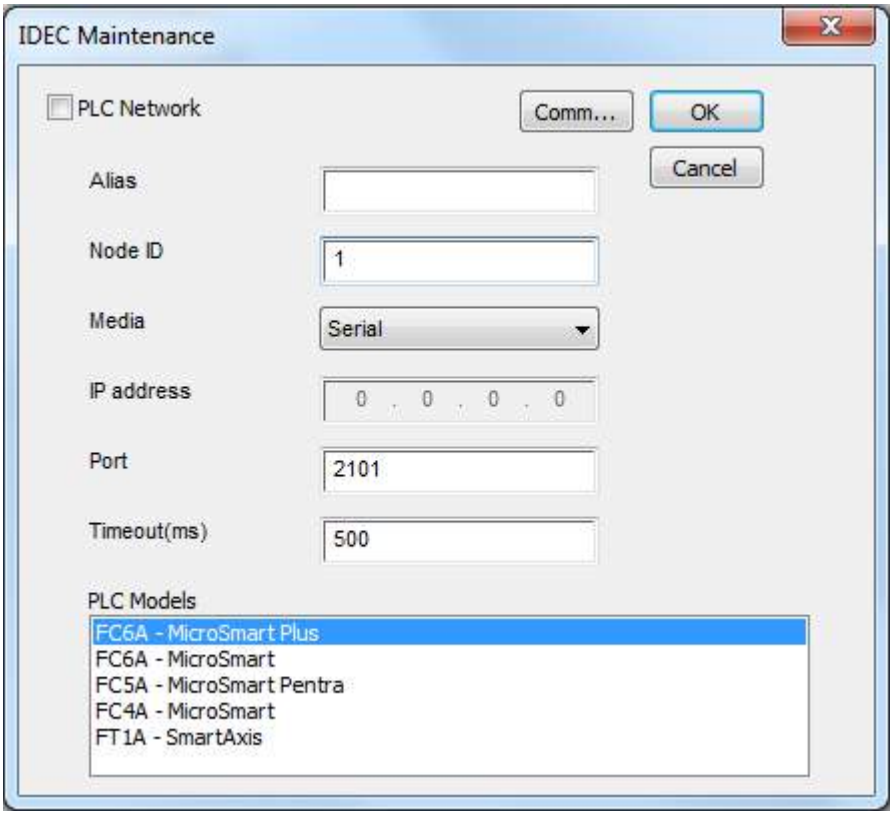
Protocol Editor Settings

Adding a protocol

To configure the protocol:

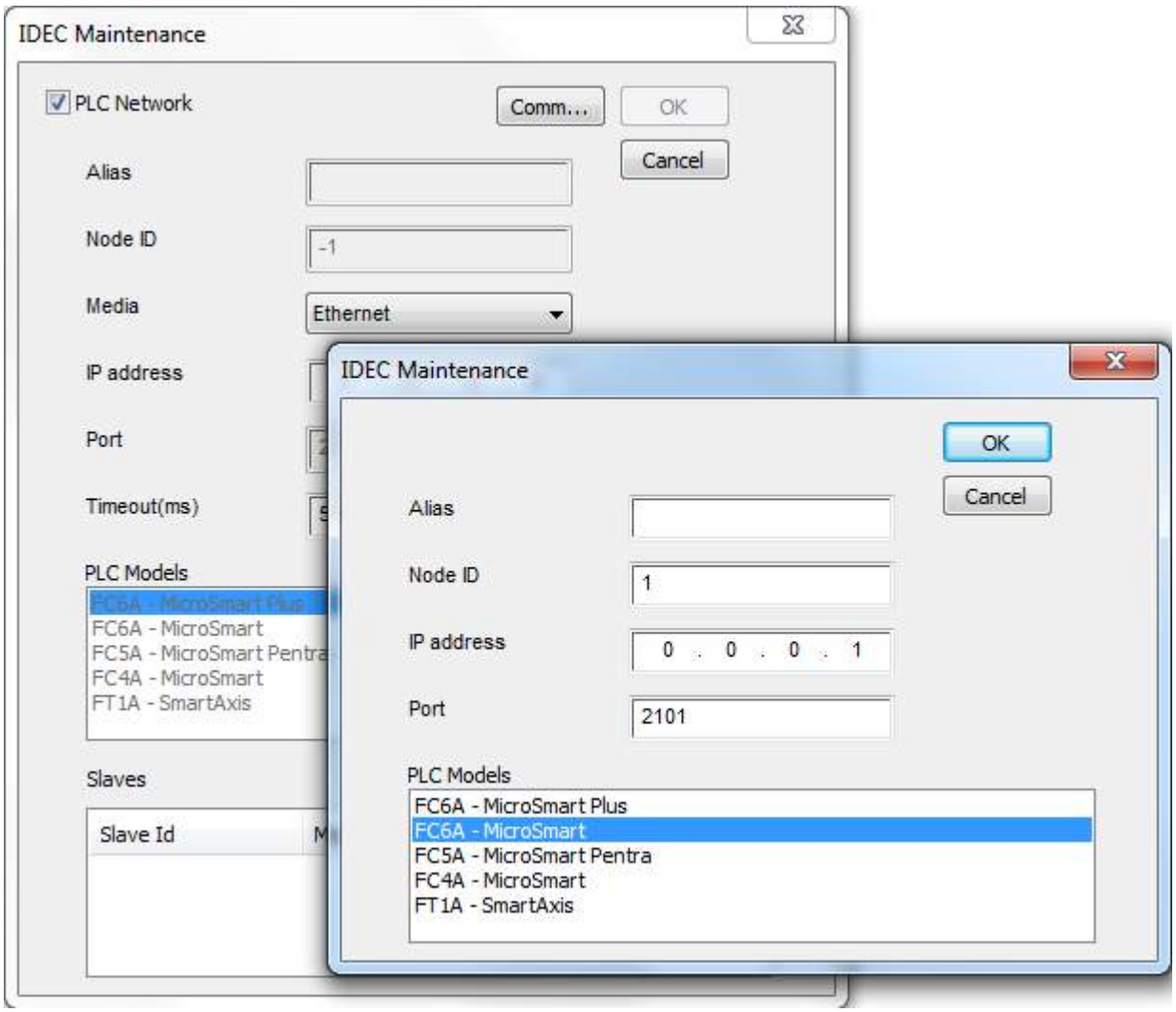
1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

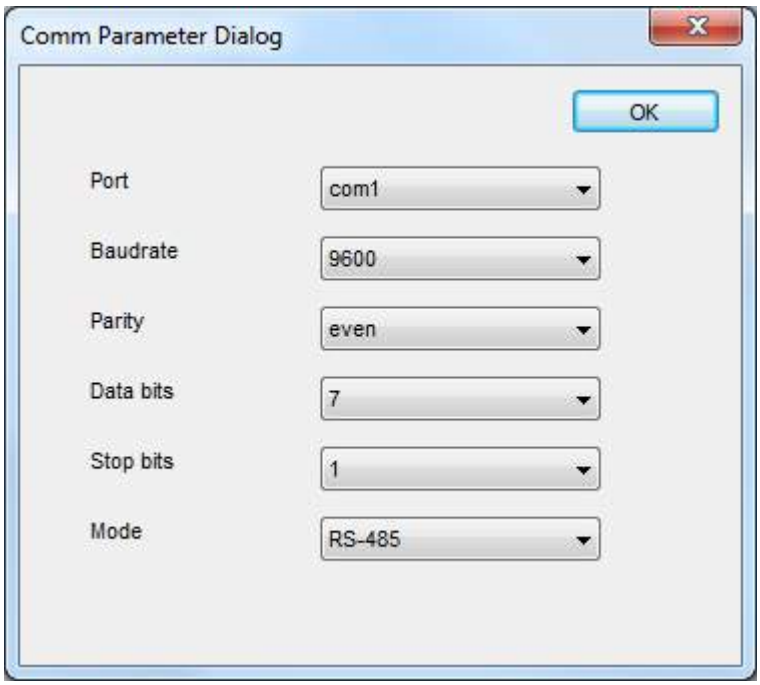
The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Node ID	Serial node associated to PLC.
Media	Allows the selection of transport Media.

Element	Description
	• select **Serial** to connect via serial line • select **Ethernet** to connect via TCP/IP
IP address	IP address of PLC (only available if Ethernet media is selected)
Port	Port number of PLC
Timeout (ms)	Time delay in milliseconds between retries in case of missing response
PLC Models	PLC model available: • FC6A - MicroSmart Plus • FC6A - MicroSmart • FC5A - MicroSmart Pentra • FC4A - MicroSmart • FT1A - SmartAxis
PLC Network	Enable configuration of multiple connections.

Element	Description
	
Comm...	If clicked displays the communication parameters setup dialog (only available if Serial media is selected)

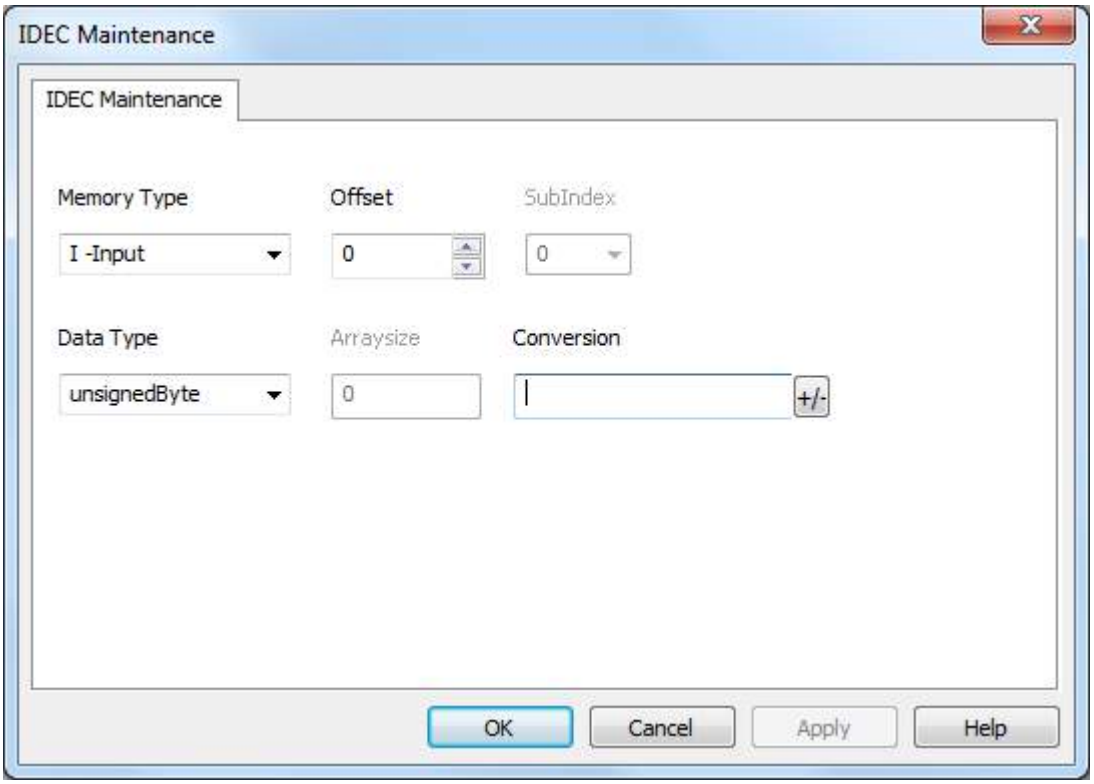
Element	Description
	

Element	Parameter
Port	Serial port selection. • **COM1**: On-board port • **COM2**: Optional Plug-in module plugged on slot#1 or slot#2 • **COM3**: Optional Plug-in module plugged on slot#3 or slot#4
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.
Mode	Serial port mode. Available modes: • **RS-232**. • **RS-485** (2 wires). • **RS-422** (4 wires).

Tag Editor Settings

In Tag Editor select **IDEC Maintenance** protocol.

Add a tag using [+] button. Tag setting can be defined using the following dialog:

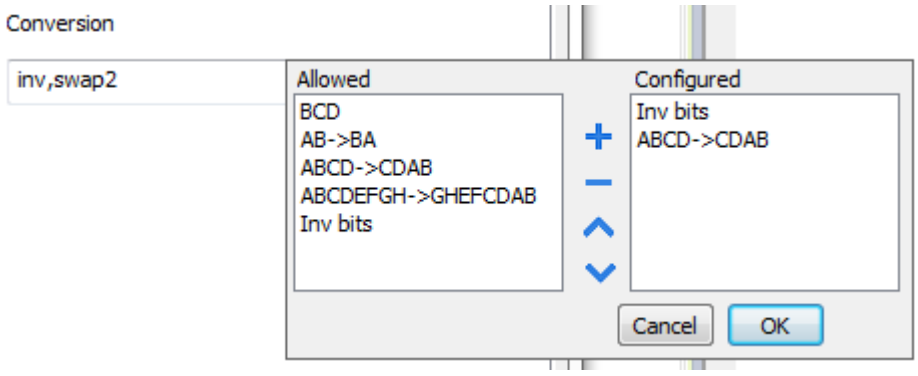

IDEC Maintenance
IDEC Maintenance
Memory Type
Offset
SubIndex
I -Input
0
0
Data Type
Arraysize
Conversion
unsignedByte
0
+/-
OK
Cancel
Apply
Help

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>I - Input</td><td>I resources. Corresponding to internal digital Input point.</td></tr>
<tr><td>Q - Output</td><td>Q resources. Corresponding to internal digital Output point.</td></tr>
<tr><td>M - Internal Relay</td><td>M resources. Corresponding to PLC internal memory.</td></tr>
<tr><td>R - Shift Register</td><td>S resources. Corresponding to PLC shift registers.</td></tr>
<tr><td>T - Timer</td><td>T resources. Corresponding to PLC timers.</td></tr>
<tr><td>TC - Timer Current Value</td><td>TC resources. Corresponding to PLC timer current values.</td></tr>
<tr><td>TP - Timer Preset Value</td><td>TP resources. Corresponding to PLC timer preset values.</td></tr>
<tr><td>C - Counter</td><td>C resources. Corresponding to PLC counters.</td></tr>
<tr><td>CC - Counter Current Value</td><td>CC resources. Corresponding to PLC counter current values.</td></tr>
<tr><td>CP - Counter Preset Value</td><td>CP resources. Corresponding to PLC counter preset values.</td></tr>
<tr><td>D - Data register</td><td>D resources. Corresponding to PLC data registers.</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>Starting address for the Tag. The possible range depend on PLC model selected.</td></tr>
<tr><td>Subindex</td><td>This allows resource offset selection depending on the selected data type.</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

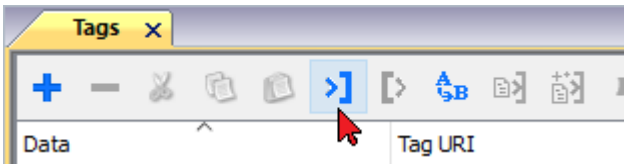
Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001 → 1 10000011100 1010101000010100010111011011011001011011000010011101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

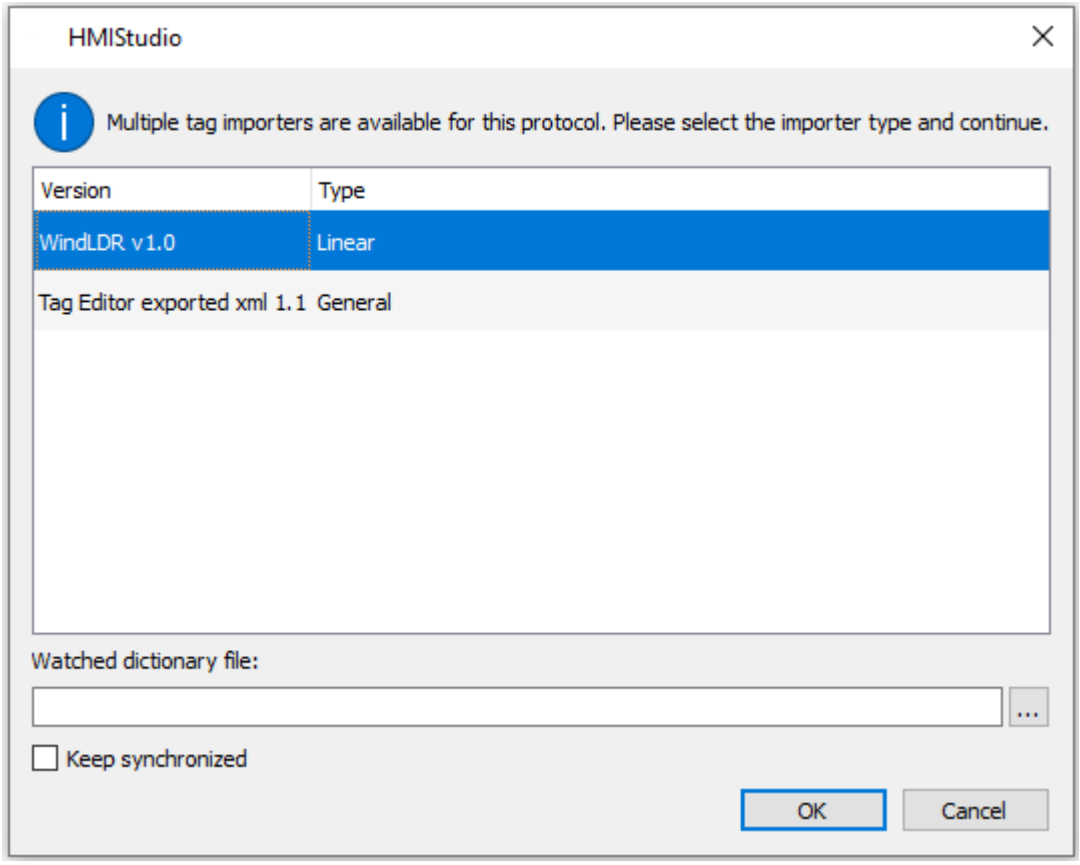
Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



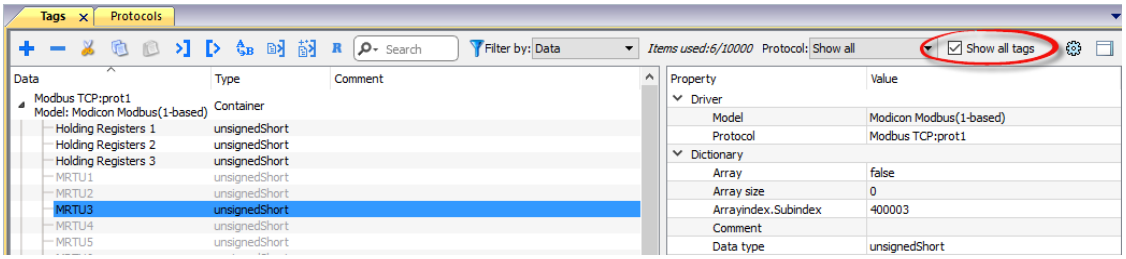
The following dialog shows which importer type can be selected.



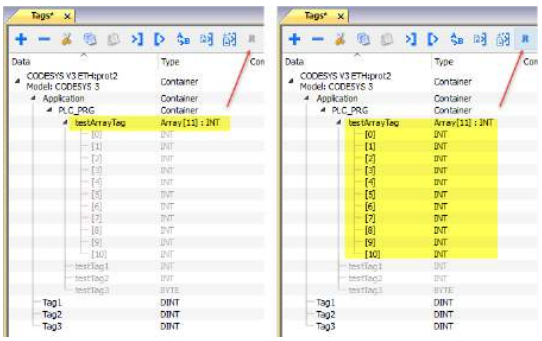
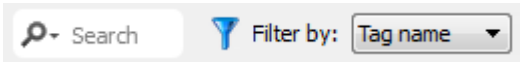
Type	Description
WindLDR v1.0 Linear	Requires a **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

J1939

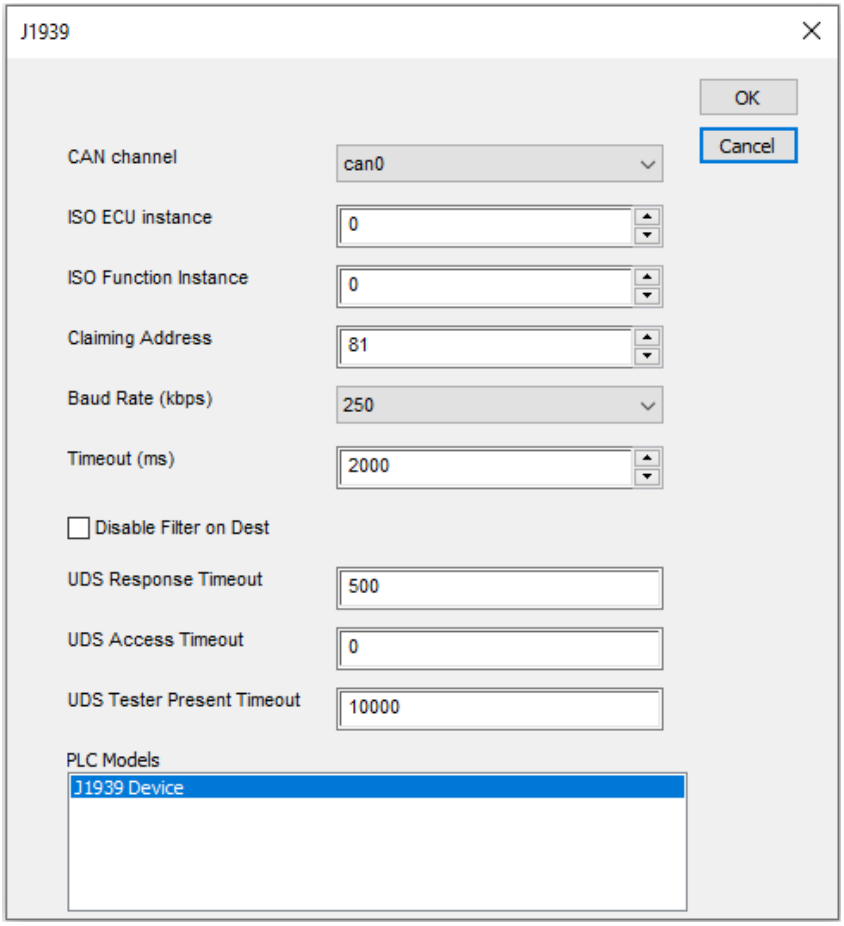
Use this communication driver to connect HMI devices to CAN networks including devices communicating with SAE J1939.

Please note that changes in the communication protocol specifications or J1939 hardware may have occurred since this documentation was created. Some changes may eventually affect the functionality of this communication driver. Always test and verify the functionality of your application. To fully support changes in J1939 hardware and communication protocols, communication drivers are continuously updated. Always ensure that the latest version of communication driver is used in your application.

Protocol Editor Settings

Select Add [+] in Protocol Editor and select J1939.

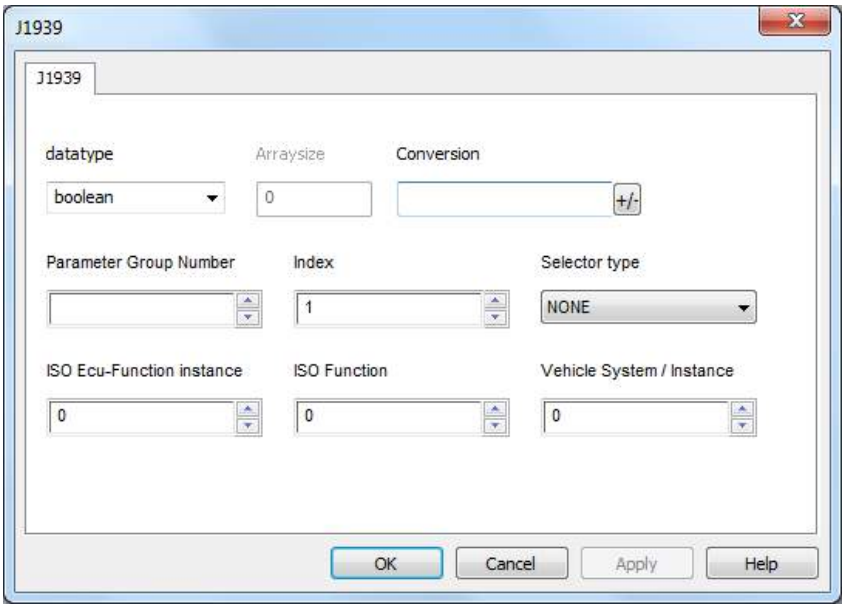
The driver configuration dialog is shown in figure.



Element	Description
CAN Channel	Configure the CAN Channel. CAN interface is available only with a proper option module. UN31 platforms allow only one module, select Can0. UN30 platforms allow up to two modules, select Can0 or Can1.
ISO ECU Instance	Identifier of the equipment in the J1939 network (in case several HMI are coexisting in the network)
ISO Function Instance	Identifier of the function in the network (in case more than one device is providing the same functionality)
Claiming Address	Default value of the address of the equipment used as starting value for the Address Claim algorithm
Baud Rate (kbps)	Baud rate of the CAN bus (typical is 250)
Timeout (ms)	Timeout for the validity of received values. After the time indicated since last reception any value is declared “old” and its quality changed to “bad”. The value 0 disables the timeout check

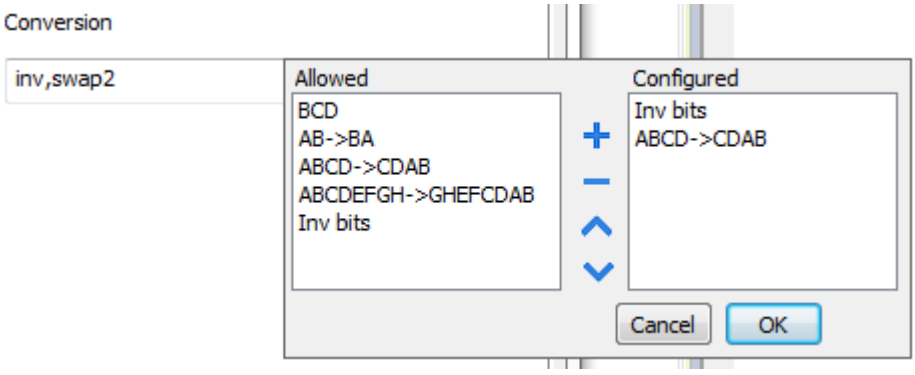
Tag Editor Settings

In Tag Editor select the protocol “J1939” from the list of defined protocols and add a tag using [+] button. Tag settings can be defined using the following dialog:



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>Inv bits</td><td>inv: Invert all the bits of the tag. Example: 1001 → 0110 (in binary format) 9 → 6 (in decimal format)</td></tr><tr><td>Negate</td><td>neg: Set the opposite of tag value. Example: 25.36 → -25.36</td></tr><tr><td>AB -> BA</td><td>swapnibbles: Swap nibbles in a byte. Example: 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)</td></tr><tr><td>ABCD -> CDAB</td><td>swap2: Swap bytes in a word. Example: 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)</td></tr><tr><td>ABCDEFGH -> GHEFCDAB</td><td>swap4: Swap bytes in a double word. Example: 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)</td></tr><tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001 → 1 10000011100 1010101000010100010110110110110010110110000100111101 (in binary format)</td></tr><tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9) Example: 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)</td></tr></table>

Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.
Parameter Group Number	Parameter Group Number value
Index	Index value
Selector Type	When adding tags it can be necessary to duplicate them to read data coming from several devices generating same physical quantity. In this case the Address of the tag must be edited. The Tag Editor dialog is shown in figure: In case of duplication of the tag, the selection of incoming data can be done using one of following methods: **NONE** Selector Type not selected **INSTANCE** uses a defined bitfield value in data of PGN to distinguish between the possible sources. The value of received bitfield is compared with parameter “Vehicle System / Instance” for matching **DEVICE** uses the source address to find out the device sending the PGN based on Address Claim algorithm. The devices are selected based on parameter “ISO function” **ADDRESS** uses directly the source address as it is to select the source. The received source address is compared with parameter “ISO Ecu – Function Instance”
ISO Ecu-Function Instance	Instance of ISO Ecu-Function checked with Selector Type “DEVICE”
ISO Function	ISO Function parameter
Vehicle System/Instance	Vehicle System / Instance parameter used with Selector Type “INSTANCE”

J1939 PGN Definition File

J1939 can connect hundreds of different devices offering access to thousands of different physical values. The standard defines several hundred PGNs for various applications. However, many devices use manufacturer-specific PGN definitions.

In order to manage this complex application scenario, the J1939 driver loads the PGN definition table at startup from a configuration file. The file with the PGN definition table is “J1939_pgnTable.csv” located in the folder “*target\protocols*\”; it is loaded automatically from disk when downloading the project.

The file containing the PGN defined by the standard protocol specification is placed in the proper folder when the driver is installed. It can be edited adding or removing PGN definitions. The user must respect the following rules:

- the file contains most of the PGN defined by the standard. Custom PGN and SPN can be added assigning free indexes.
- description of a PGN is composed by a PGN declaration line followed by a list of Field description lines

PGN declaration line

```
PGN: Name, PGN number, DefaultPriority, DefaultRate, InstanceIndex, Direction [, PGN request rate]
```

Name	Name of the PGN
PGN number	Number code of PGN
DefaultPriority	Transmission priority (output PGN)
DefaultRate	Transmission rate (output PGN)
Instance Index	Index of instance (output PGN)
Direction	INPUT/OUTPUT
PGN request rate	Optional parameter. Time in milliseconds. If PGN not received in the meanwhile, it is requested

Example of PGN declaration:

PGN: Torque/Speed Control 1, 0, 3, 100, 0, INPUT

// Torque/Speed Control 1 id PGN nr.0, its default priority is 3 and default transmission rate is 100 ms. Instance Index is 0 and direction is INPUT

Field declaration line

```
FieldIndex, FieldName, FieldPosition, FieldBitSize, SPN Conversion, AccessType, FieldDataType
```

FieldIndex	Index of field in the PGN
FieldName	Name of the field
FieldPosition	N (1 to 8) byte position
	N.M (1.1 to 8.8) bit position
	N-M (N from 1 to 7, M from 2 to 8) byte range
FieldBitSize	1-64
	number of bits of the field
SPN Conversion	SPN conversion is indicated by "SPN"index es. SPN79
	SPN0 indicates a raw copy of data
AccessType	Defines usage of field in combination with PGN direction.

If PGN direction is declared as OUTPUT, the fields can be only used for write operations.

If PGN direction is declared as INPUT the fields can always be read. In case they are written the behavior is described below.

PGN Direction	Access Type	Behavior
OUTPUT	WRITE	the PGN is sent immediately with current value of the fields
	READ_ ONLY REPLY	the PGN is sent as soon as all the fields are written with a fresh value
INPUT	READ_ ONLY	Error
	REPLY	the PGN is sent only if it was received almost once, with update value of the written field
	WRITE	the PGN is sent immediately with current value of the fields

FieldDataType

Boolean

boolean-nn

byte

unsignedByte

short

unsignedShort

int

unsignedInt

float

double

string-nn

Example of Field declaration:

1, Engine Override Control Mode, 1.1, 2, SPN0, READ_ONLY, unsignedByte

SPN declaration line

```
SPN: index, constK, constL, type [,bigEndian]
```

index	index of SPN
constK	SPN conversion parameters
constL	the conversion applied when reading is:
	var(type) = raw value * constK + constL
	the conversion applied when writing is:
	raw value = (var(type) - constL) / constK
type	bits
	char
	uchar
	short
	ushort
	int
	uint
	float
	double
	longlong
	ulonglong
	float80
bigEndian	Optional parameter. Defines if endianity conversion is needed on raw data before applying the SPN conversion.
	0 default endianity, do not change
	1 apply endianity transformation

Example of SPN declaration:

SPN:, 79, 0.03125, -273, short, 1

Send DM1 alarms

Diagnostic Trouble Codes (DTC) are part of SAE J1939 standard which is widely used as a communication and diagnostic protocol for the physical CAN layer in automotive applications. The SAE J1939 standard provides 12 Diagnostic Messages (DMs).

The HMI device can send the DM1 message through the JavaScript using the *invokeProtocolCommand()* interface.

invokeProtocolCommand(protID, command, params)

Example of usage:

```
var tagMgr = project.getWidget("_TagMgr");
var protID = "prot1";            // to be set according to protocol numbering
```

```
var params = "B1 B1 SPN FMI CM"; // to be filled with real data

tagMgr.invokeProtocolCommand(protID, "SetAlarm", params);
```

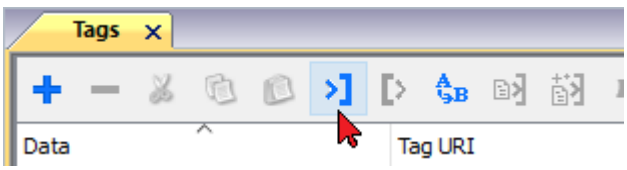
Parameter	Description
protID	Protocol id, available from the protocol editor (e.g. "prot2")
command	Command to execute, it can be: • "SetAlarm" • "ResetAlarm"
params	It is a string that contains five numbers separated by a spaces (e.g.: "4 4 520803 31 53") **Byte** / **Bits** / **Description** B1 / 8-7 / Malfunction Indicator Lamp Status 6-5 / Red Stop Lamp Status 4-3 / Amber Warning Lamp Status 2-1 / Protect Lamp Status B2 / 8-7 / Flash Malfunction Indicator Lamp 6-5 / Flash Red Stop Lamp 4-3 / Flash Amber Warning Lamp 2-1 / Flash Protect Lamp SPN / - / Suspect Parameter Number (as per J1939 specifications) FMI / - / Failure mode identifier (as per J1939 specifications) CM / - / Conversion Method (as per J1939 specifications)

Tag Import

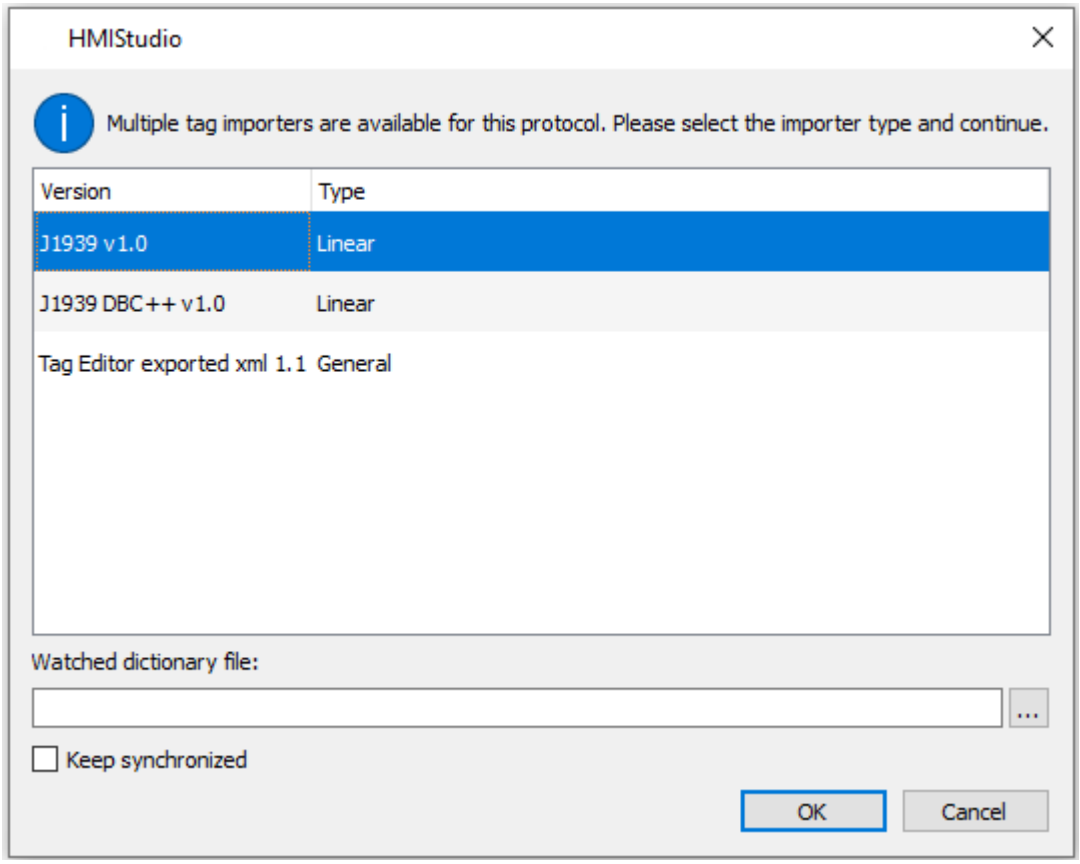
The J1939 driver can import tag information from any CSV file, following same rules of PGN definition file and maintain several dictionaries for different scenarios.

The user can also import the whole “J1939_pgnTable.csv” and use only one large dictionary.

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



The following dialog shows which importer type can be selected



Type	Description
J1939 v1.0 Linear	Requires a **.csv** file. All variables will be displayed at the same level.
J1939 DBC+ v1.0 Linear	Requires a .dbc file generated by Vector CANdb++ Editor All the frames will be generated with type = Rx, so frames created for transmission must be reedit after importation
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

The tags resulting from the import process may be used as they are if there is only one source for such value in the network. When several sources are supplying the same value the associated tags must be duplicated and named using one of the addressing methods shown in the Tag Editor chapter.

Communication Diagnostic

The error types supported for this communication driver are:

Error Class	Error
Configuration Errors	invalid CAN channel
	cannot read MACID
	Unable to access the PGN Table
	Unable to get the PGN file path
	SPN conversion not supported
	Sending PGN with dynamic field length not supported
	Preparing PGN field for sending failed
	Writing a read-only tag
	The output PGN can't be read
	invalid offset in PGN
	Not byte boundary on dynamic field
	Something wrong with the PGN data block size
	Too many bits to use
	Not byte boundary on dynamic field
	SPN conversion not supported
Runtime Errors	Communication Failure > Can't send the APL PGN message
	Not Connected > The PGN for the command reply has not been received yet
	Not Connected > PGN block not registered
	Not Connected > the value never received
	Timeout Error > timeout on the value refresh

Error Class	Error
Tag Definition Errors	there must be 7 tag specification fields
	PGN field missing
	SPN definition not found in the table
	index field missing
	ecuFunctionInstance field missing
	function field missing
	classOrInstance field missing
	icomType field missing
	Can't access protocol common parameters
	Can't access protocol node parameters
	Can't access model
	Can't access memory type
	strError.c_str()
	not allowed icom type
	invalid natural data type for this memory type
	invalid field 'selector type'
	PGN definition not found in the table
	The field not found in this PGN

Jetter Ext ETH

The Jetter Ext ETH driver has been developed to communicate with Jetter devices using the PCOM7 protocol.

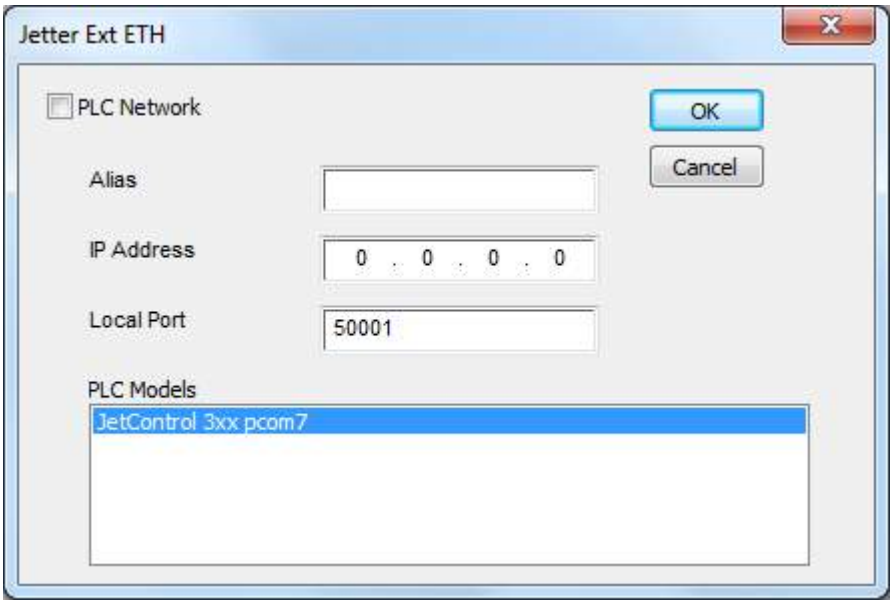
The HMI protocol identifies Jetter devices using their IP addresses. You should take note of these addresses as you assign them because you will need them later in the set-up phase of the user interface application.

Different physical media, gateways, routers and hubs can be used in the communication network. Also, other devices can independently make simultaneous use of the network. However, it is important to ensure that the traffic generated by these devices does not degrade the communication speed (round-trip time) to an unacceptable level. Too slow communication between the device and the Jetter device may result in low display update rate.

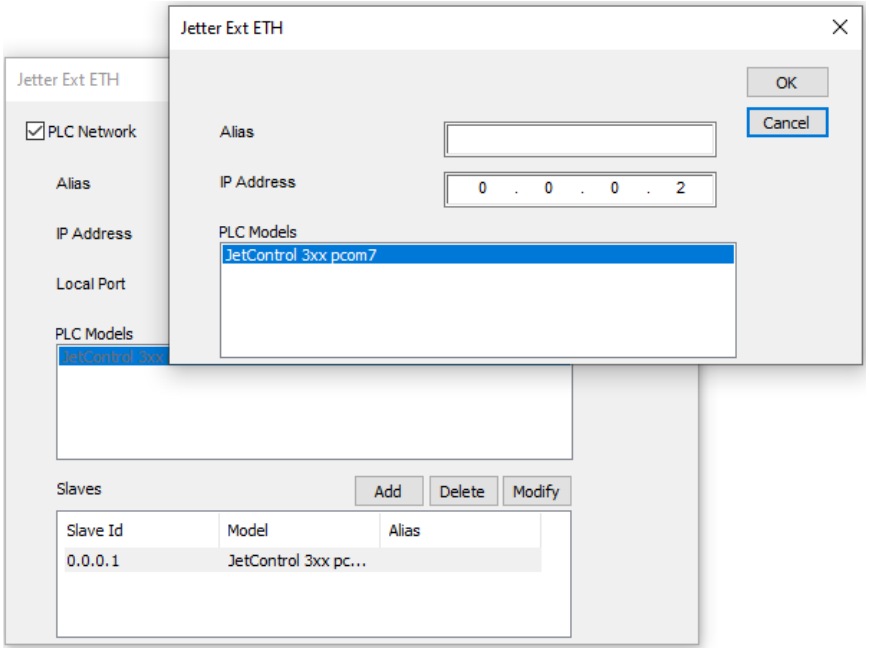
Protocol Editor Settings

Add (+) a new driver in the Protocol editor and select the protocol called “Jetter Ext ETH” from the list of available protocols.

The driver configuration dialog is shown in the following figure.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the PLC.
Local Port	Allows to specify the source Port used from the HMI to communicate with PLC.

Element	Description
PLC Models	An unique PLC model is available: JetControl 3xx pcom7.
PLC Network	The protocol allows the connection of multiple controllers to one HMI device. To set-up multiple connections, check “PLC network” checkbox and enter IP Address for all PLCs. 

Tag Editor Settings

Into Tag editor select the protocol “Jetter Ext ETH” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:

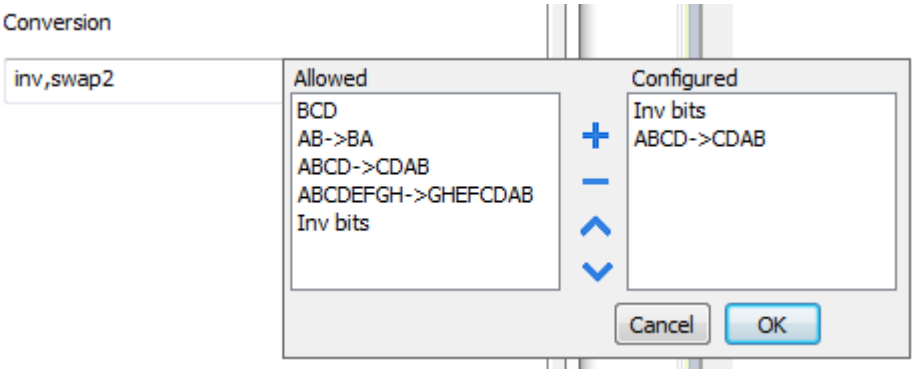
Element	Description
Memory Type	Area of PLC where tag is located.
Offset	Offset address where tag is located.
SubIndex	This allows resource offset selection within the register.
Type	

Data Type	Memory Space	Limits
boolean	1 bit data	0 ... 1
byte	8-bit data	-128 ... 127
short	16-bit data	-32768 ... 32767
int	32-bit data	-2.1e9 ... 2.1e9
unsignedByte	8-bit data	0 ... 255

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
<tr><td>string</td><td colspan="2">Refer to “String data type chapter”</td></tr>
</table>

Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the Allowed list shows one or more conversions, listed below.
</td></tr>
</table>

Element	Description

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	Separate the byte in two nibbles, and reads them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)
New Format	Jetter “string” data format

Element	Description
	Select the conversion and click on plus button. The selected item will be added on **Configured** list. If more conversions are configured, they will be applied in order (from top to bottom of **Configured** list). Use the arrow buttons to order the configured conversions.

Special data types

The Jetter Ext ETH driver provides one special data type called "Node Override IP".

The Node override IP allows changing at runtime the IP address of the controller. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

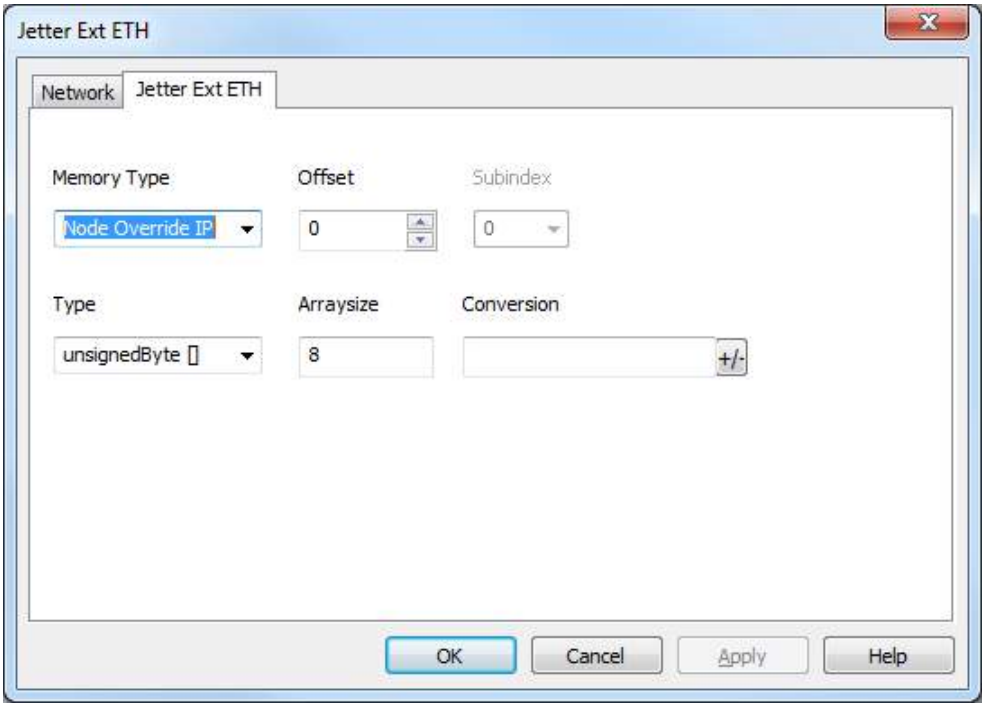
The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

If the Node Override IP is set to 0.0.0.0, all the communication with the slave is stopped, no request frames are generated anymore.

If the Node Override IP has a value different from 0.0.0.0, it is interpreted as node IP override and the controller IP address is replaced runtime with the new value.

In case the device has been configured to access to a network of controllers, each node has its own Node Override IP variable.

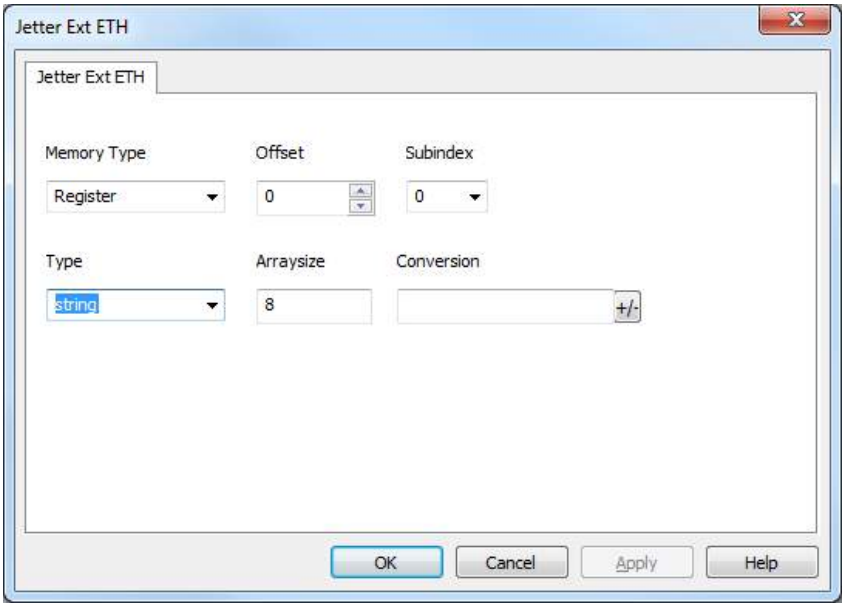
Note:the Node Override IP values assigned at runtime are retained through power cycles



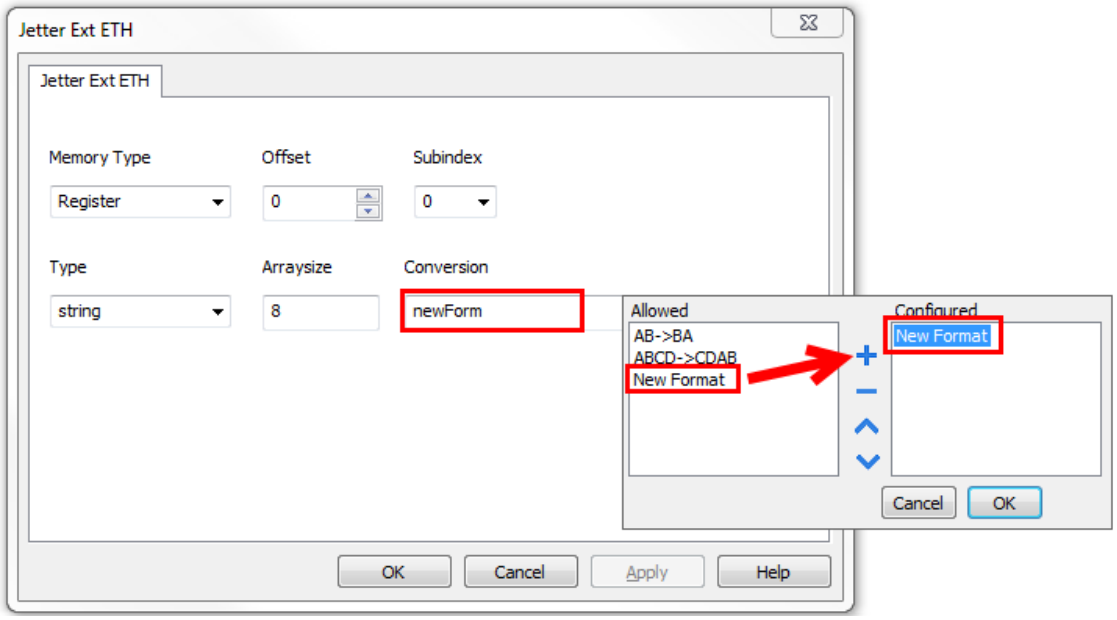
String data type

The Jetter devices allow to define within the programming software two different type of string variables: “Regstring” is the old format while “string” is the new format, both these formats are supported by the Jetter Ext ETH driver.

When “Regstring” format is used the corresponding Tag must be configured simply selecting string as data type as shown in the following figure, no further steps are required.



When “string” format is used once selected the string data type in the Tag definition dialog it is necessary, as shown in the following figure, to add a New Format conversion.



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
No response	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Incorrect node address in response	The device did receive from the controller a response with invalid node address
The received message too short	The device did receive from the controller a response with invalid format
Incorrect writing data acknowledge	Controller did not accept write request; ensure the data programmed in the project are consistent with the controller resources

Keyence KV

Keyence KV communication driver has been designed to connect HMI devices to KEYENCE PLCs through Serial or Ethernet connection.

Please note that changes in the communication protocol specifications or PLC hardware may have occurred since this documentation was created. Some changes may eventually affect the functionality of this communication driver. Always test and verify the functionality of your application. To fully support changes in PLC hardware and communication protocols, communication drivers are continuously updated. Always ensure that the latest version of communication driver is used in your application.

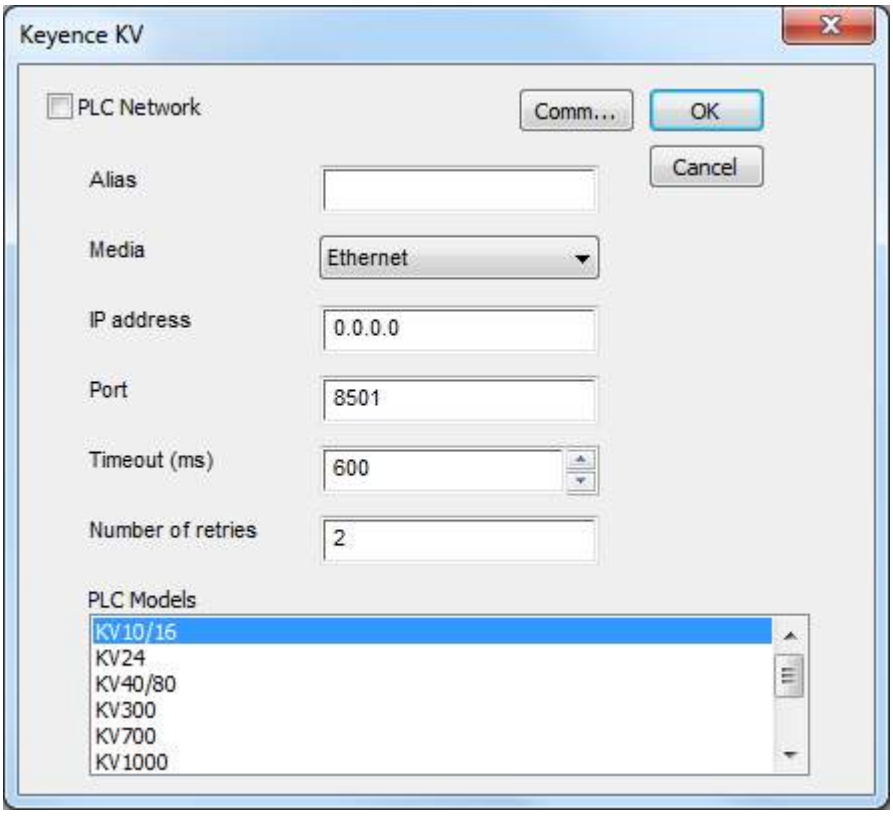
Setting-up the PLC for Communication

Keyence KV PLC's do not require any particular setup-up for communication at the programming port.

Protocol Editor Settings

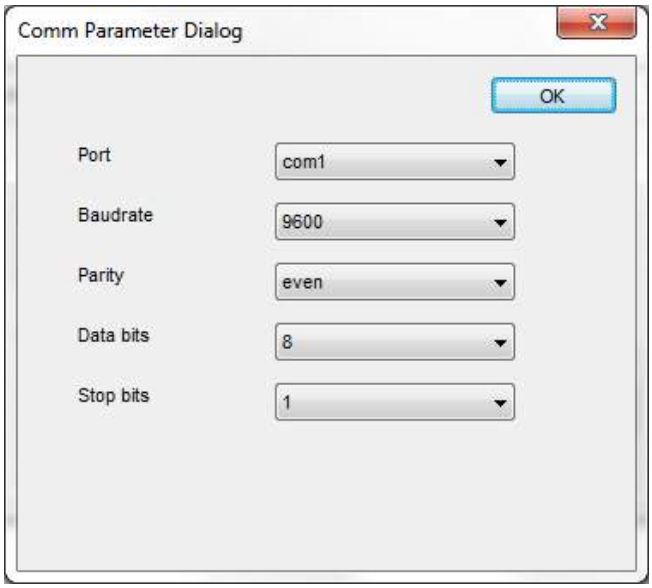
Add (+) a driver in the Protocol Editor and select the protocol called "Keyence KV" from the list of available protocols.

The driver configuration dialog is shown in figure.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Media	Allows the selection of transport Media.

Element	Description
	• select **Serial** to connect via serial line • select **Ethernet** to connect via TCP/IP
IP address	IP Address of the controller. Only available for **Ethernet** Media.
Port	Port number used by PLC. The default value is **8501**. Only available for **Ethernet** Media.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from PLC.
Number of retries	Number of times a communication session is repeated before declaring reporting communication error.

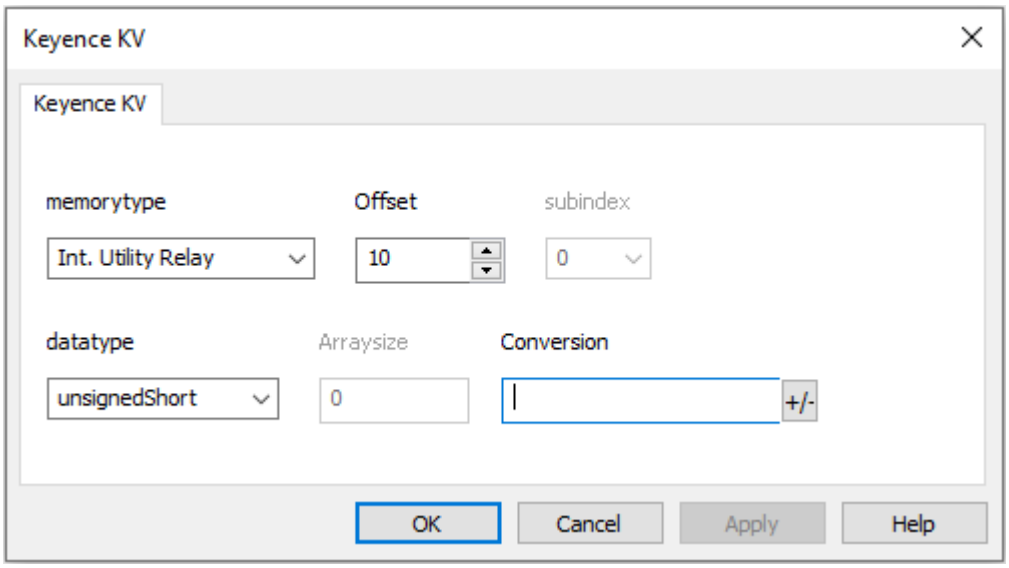
Element	Description
PLC Models	The list allows selecting the PLC model. The selection will influence the data range offset per each data type according to the specific PLC memory resources. Available models: • KV10/16 • KV24 • KV40/80 • KV300 • KV700 • KV1000 • KV3000/5000/5500 • KV7300/7500 • KV8000
Comm…	Opens the serial port configuration dialog box. Only available for **Serial** Media. 

Element	Parameter
Port	Serial port selection. • **COM1**: On-board port • **COM2**: Optional Plug-in module plugged on slot#1 or slot#2 • **COM3**: Optional Plug-in module plugged on slot#3 or slot#4
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Keyence KV** from the **Driver** list: tag definition dialog is displayed.

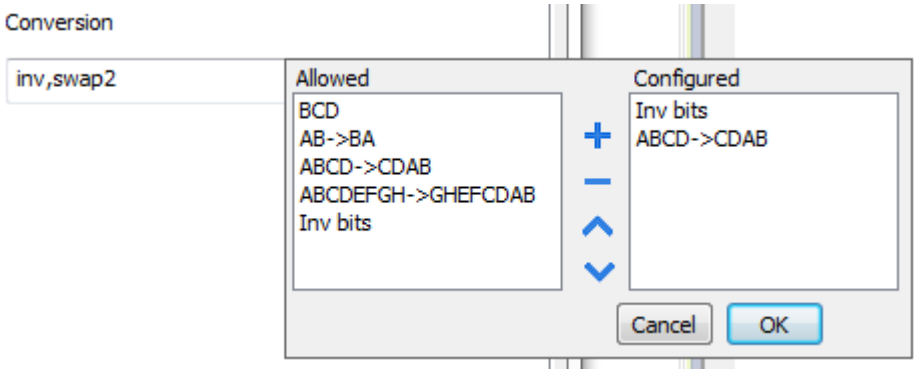


<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Resource where tag is located on PLC.
Available resources are:
Int. Utility Relay
Data Memory
Timer Contact
Timer Current
Timer Preset
Counter Contact
Counter Current
Counter Preset
Digital Trimmer
Control Memory
Temporary Data Memory
Control Relay
Link Relay
Int. Aux. Relay
Latch Relay
Virtual Relay
Ext. Data Memory
Curr. File Register
Dial File Register
Virtual Memory
Index Register
Link Register</td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.</td></tr>
<tr><td>subIndex</td><td>This allows resource offset selection within the register.</td></tr>
<tr><td>datatype</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description

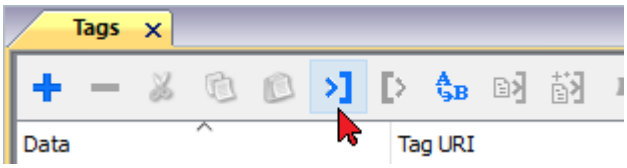
Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

Element	Description
	If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

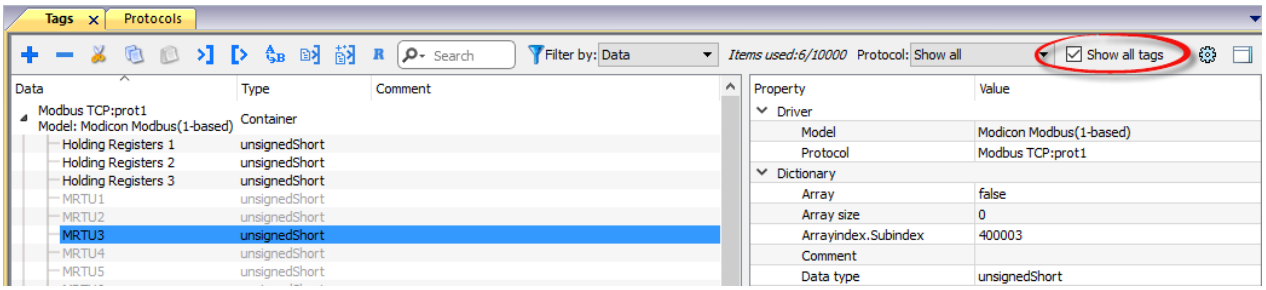
Tag Import

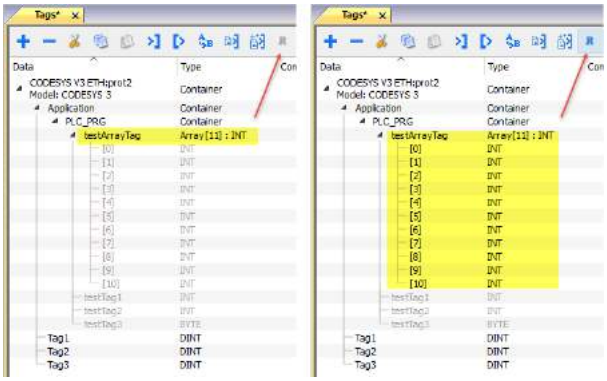
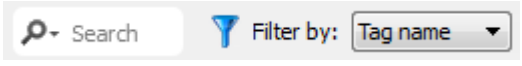
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



Locate the **.xml** file exported from Tag Editor and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Description
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Timeout receiving response characters	Returned when a request is not replied within the specified timeout period between chars in frame, should never be reported; contact technical support
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits); ensure the communication parameter settings of the controller is compatible with panel communication setup
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources

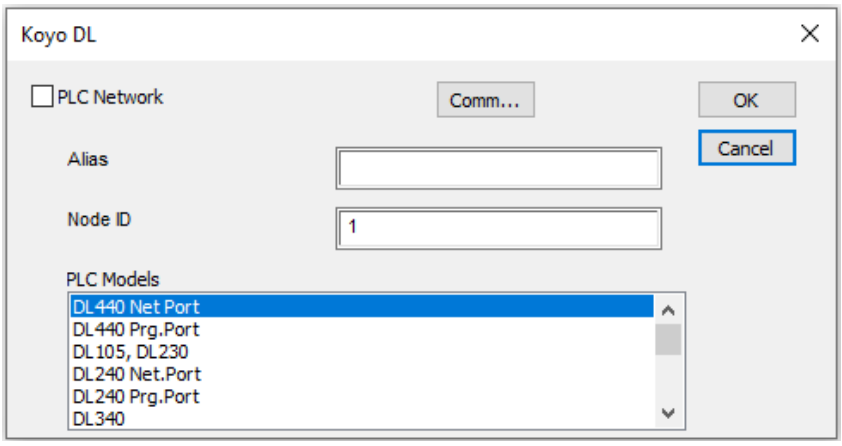
Koyo DL

The Koyo DL driver has been developed for the communication with Koyo DL series controllers trough serial connection.

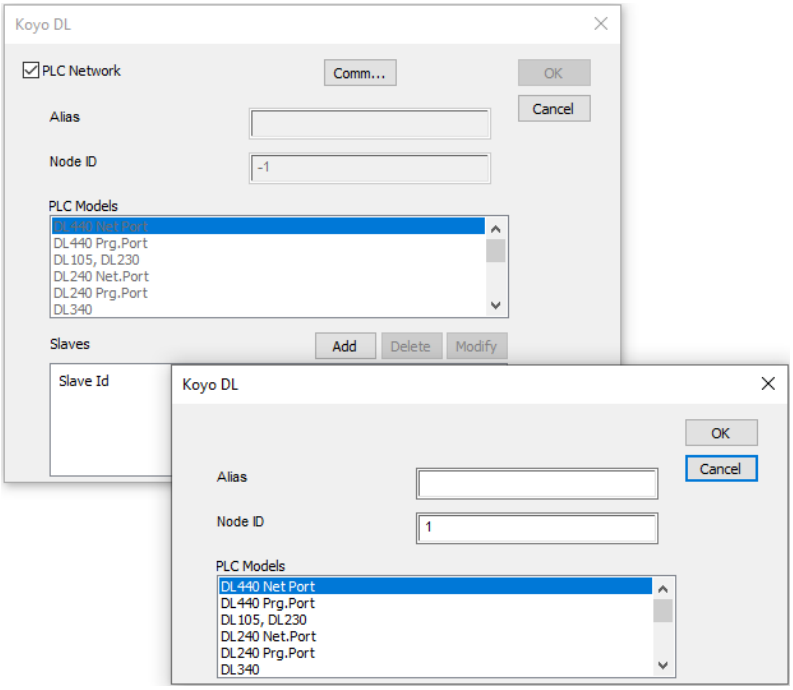
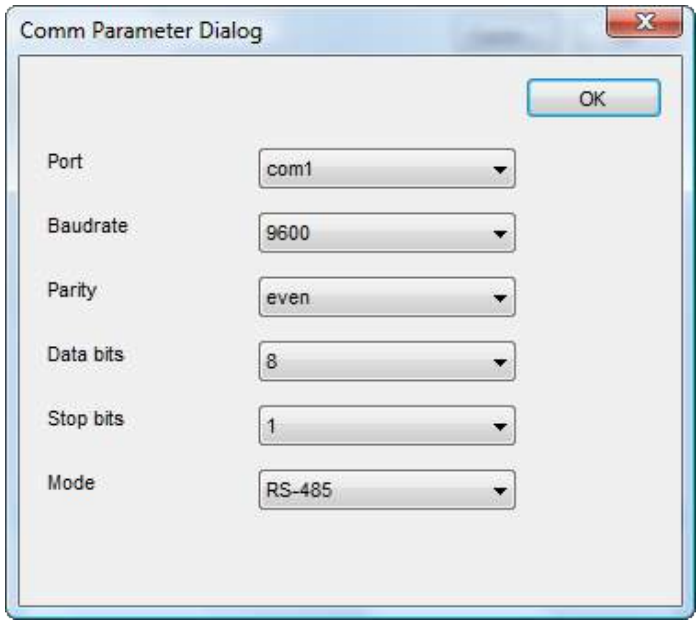
Protocol Editor Settings

Add (+) a new driver in the Protocol editor and select the protocol called “Koyo DL” from the list of available protocols.

The driver configuration dialog is shown in the following figure:



Element	Description
Node ID	Controller Node ID
PLC Models	The driver supports communication with different DL controllers. Please check directly in the programming IDE software for a complete list of supported controllers.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and configure all controllers.

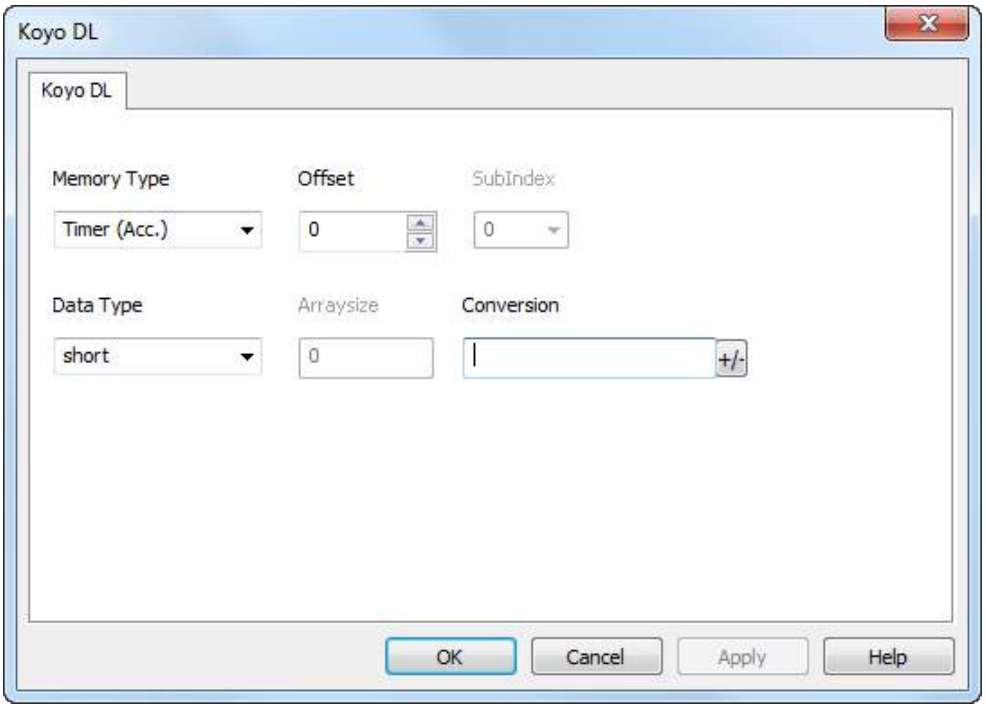
Element	Description
	
Comm…	Gives access to the serial port configuration parameters as shown in the figure below. 
Port	Serial port selection

Element	Description
Baud rate, Parity, Data bits, Stop bits	Communication parameters for serial communication
Mode	Serial port mode; available options: • RS-232, • RS-485 (2 wires) • RS-422 (4 wires)

Tag Editor Settings

Into Tag editor select the protocol “Koyo DL” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:



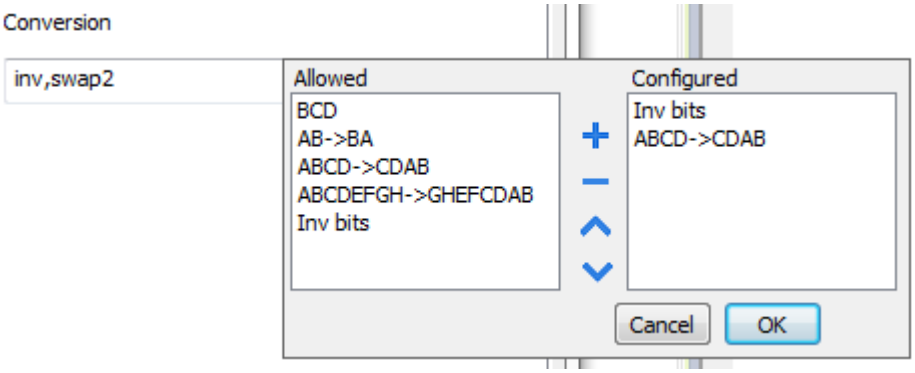
Element	Description
Memory Type	Memory resource where tag is located.
Offset	Offset address where tag is located.
SubIndex	This allows resource offset selection within the register.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding.</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the tag.</td></tr>
</table>

Element	Description
	 Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011 00

Element	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>0001 → 1 10000011100 10101010000101000101101101101100101101100001001111101 (in binary format)</td></tr><tr><td>**BCD**</td><td>**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)</td></tr></table> Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

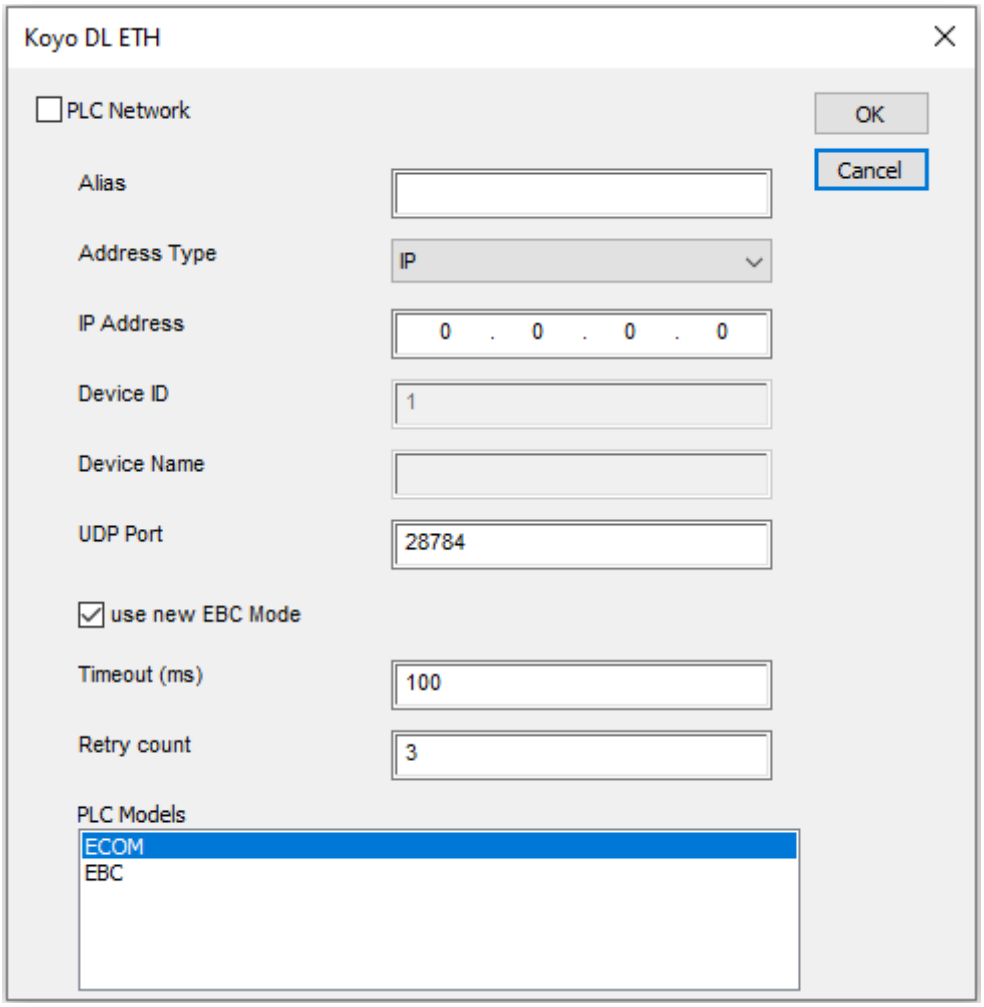
Koyo DL ETH

The Koyo DL ETH driver has been developed for the connection of Koyo DL series controllers trough Ethernet.

Protocol Editor Settings

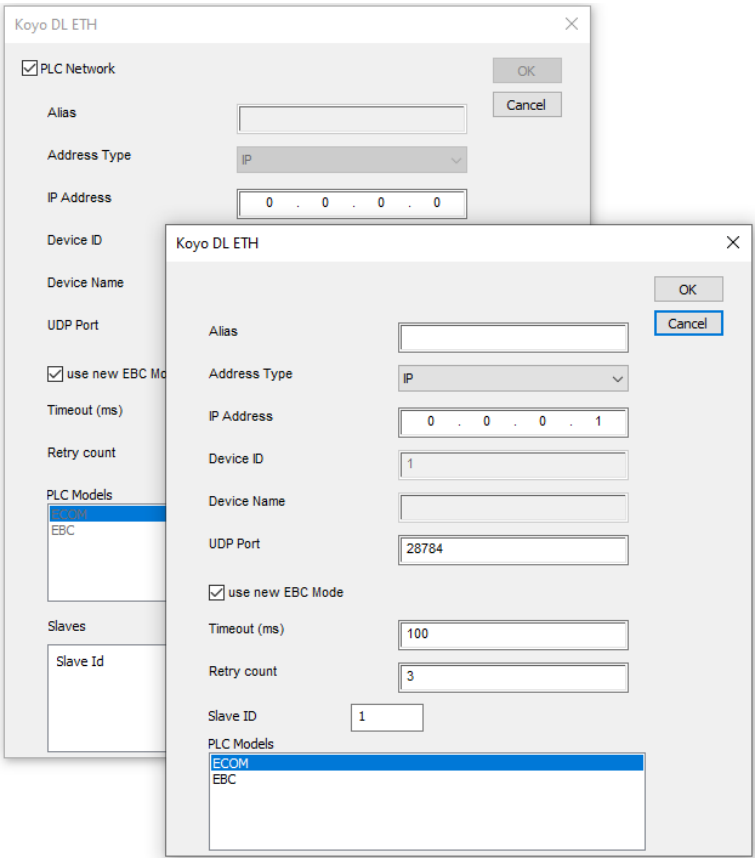
Add (+) a new driver in the Protocol editor and select the protocol called “Koyo DL ETH” from the list of available protocols.

The driver configuration dialog is shown in the following figure:



Element	Description
Address Type	Allow to select which address type to use
IP Address	When Address Type is “IP”, define the controller IP Address
Device ID	When Address Type is “ID”, define the controller Device ID

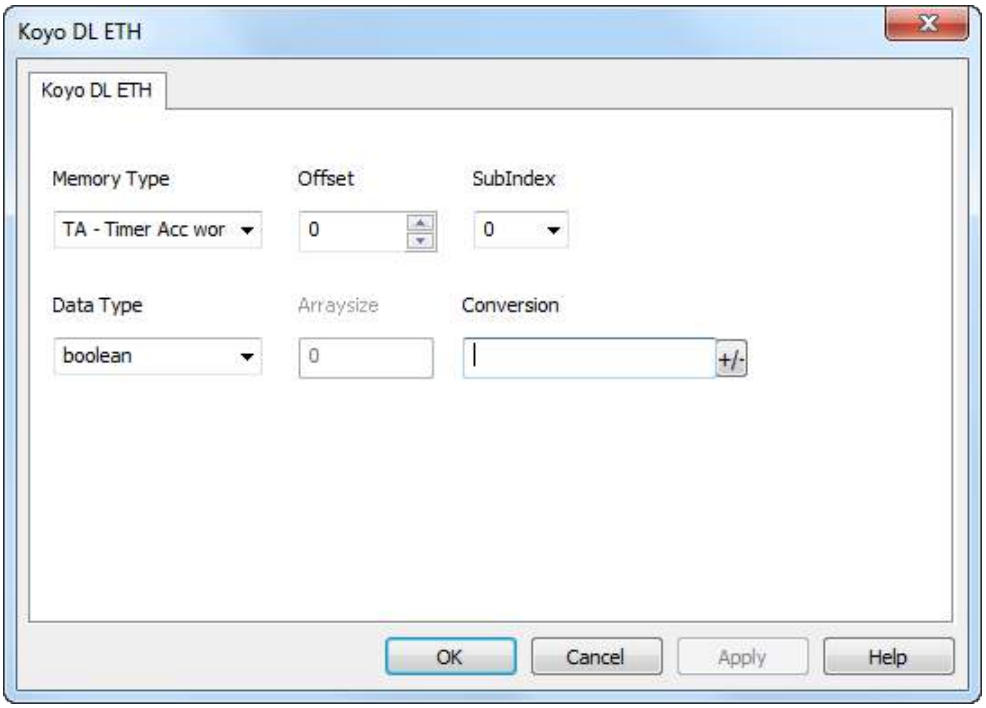
Element	Description
Device Name	When Address Type is “Name”, define the controller name
UDP Port	UDP port of controller
use new EBC Mode	If PLC Model is “EBC” allow to use the new EBC Mode
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. Value is expressed in milliseconds.
Retry count	Defines the number of times a certain message will be sent to the controller before reporting the communication error status. A value of 1 for this parameter means the HMI will eventually report the communication error status if the response to the first request packet is not correct.
PLC Models	The driver supports communication with different DL controllers. Please check directly in the programming IDE software for a complete list of supported controllers.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and configure all controllers.



Tag Editor Settings

Into Tag editor select the protocol “Koyo DL ETH” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:

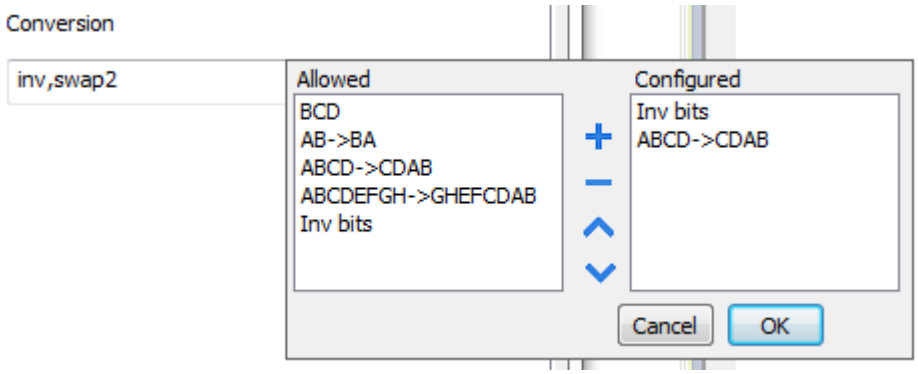


Element	Description
Memory Type	Memory resource where tag is located.
Offset	Offset address where tag is located.
SubIndex	This allows resource offset selection within the register.
Data Type	<table><tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr><tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr><tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr><tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr><tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr><tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr><tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr><tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr><tr><td>float</td><td>IEEE single-precision</td><td>1.17e-38 ... 3.40e38</td></tr></table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td></td><td>32-bit floating point type</td><td></td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding.</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

NOTE: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Inv bits</td><td>inv: Invert all the bits of the tag.

Example:
1001 → 0110 (in binary format)
9 → 6 (in decimal format)</td></tr>
<tr><td>Negate</td><td>neg: Set the opposite of tag value.</td></tr>
</table>
</td></tr>
</table>

Element	Description

Value	Description
	Example: 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 0001 → 1 10000011100 10101010000101000101101101101100101101100001001111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

KNX TP/IP

KNX is the association that promotes the KNX communication standard, designed for applications in home and building automation systems.

The KNX standard, approved as European Standard EN 50090, EN 13321-1, is based on the communication stack of EIB with some extensions. EIB is the acronym for European Installation Bus.

Additional information and further details can be found in the KNX web site www.knx.org.

The network communication media supported by the HMI panels are:

- TP-1: twisted pair, type 1, which corresponds to a bus line operating at 9600 bit/s.
- IP: network connection via TCP/IP over Ethernet network.

Note: Connection to KNX systems in TP Mode requires the optional KNX communication module PLCM02. Verify the suitable version of communication module for your HMI model.

The EIB is an event-driven decentralized automation system.

The information to be transmitted over the bus is organized in “telegrams” sent by a source to one or more destination devices.

The bus line of EIB systems carries both data and power for the devices. The data is modulated over the DC voltage of the power supply.

HMI panels are not powered from the network and they still need the usual power supply.

The planning, design and commissioning of KNX installations are normally done using the ETS configuration software. This software tool is supplied by the KNX organization. ETS is a registered trademark of KNX.

This document contains the information required to use ETS in combination with the HMI panels.

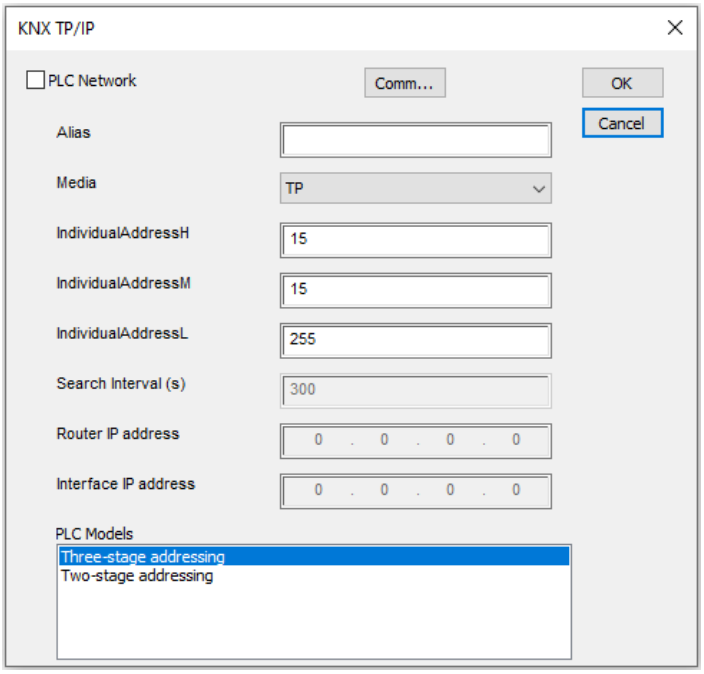
All KNX compliant devices come with a device descriptor delivered as a file to be imported in the configuration tool.

The model adopted by HMIs corresponds to a KNX device with no objects. For what concerns the ETS, the only function supported by the HMI panels is the device physical address assignment.

Protocol Editor Settings

Add (+) a driver in the Protocol Editor and select the protocol called “KNX TP/IP” from the list of available protocols.

The protocol parameters can be selected from the dedicated dialog box:

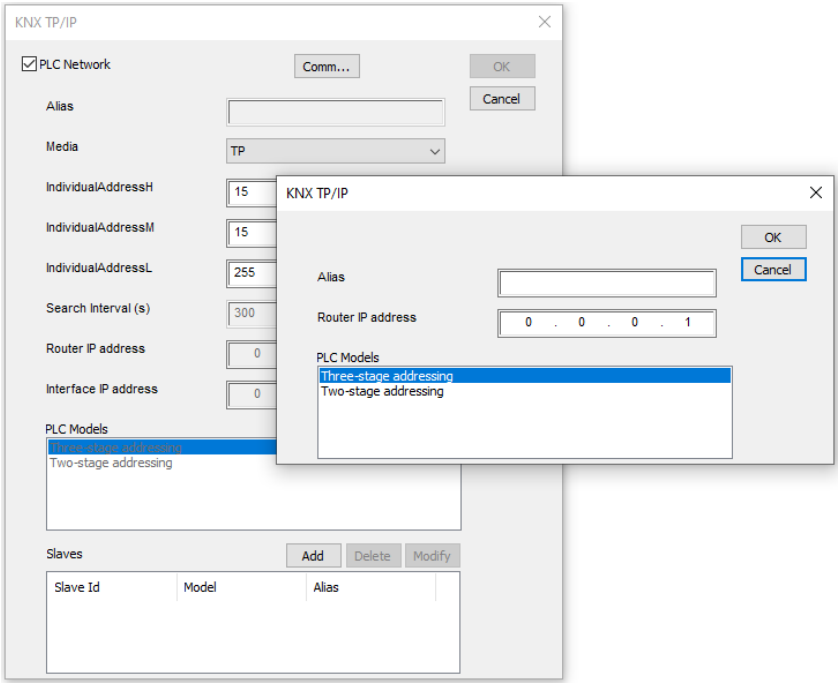
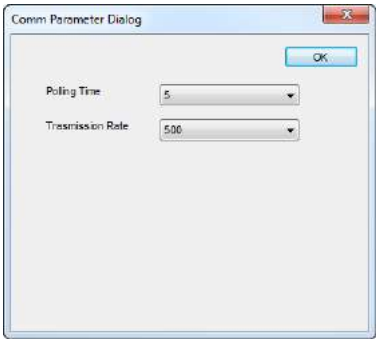


The Individual Phisical Address can be assigned on the HMI screen at the first download of the project configured for the KNX protocol. This is the Phisical Address that identifies the panel in the KNX network. The default address value is: 15.15.255.

Note: As any other KNX device, also the HMI device must have unique Individual Address in the KNX network and it must correspond to the real point in the network where the HMI device is connected.

Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Media	Allows the selection of the transport Media. • select **TP** to connect to the KNX network using the optional KNX communication module PLCM02 • select **IP** to connect to the KNX network via TCP/IP
IndividualAddressH	Physical Address High Part (Area)
IndividualAddressM	Physical Address Medium Part (Line)
IndividualAddressL	Physical Address Low Part (Device)
PLC Models	Allows to choose if KNX telegrams have two or three stage addressing. This selection have to be made basing on KNX device used. • **Two-stage addressing** = KNX telegrams are composed by GoupAddressH / GroupAddressL • **Three-stage addressing** = KNX telegrams are composed by

Element	Description
	GoupAddressH / GroupAddressM / GroupAddressL
Search Interval (s)	Available only when Media property is set to **IP**. The KNX driver will re-evaluate the network with period “Search Interval” (default: 300 seconds). On searching the network, the KNX driver will discover the tunneling endpoints that are available at that time. Endpoints will therefore be registered as possible sources / destinations for group address operations. Depending on endpoints settings or endpoints temporary unavailability the available sources / destinations for group address operations may vary. Thus the capability for the KNX driver to re-evaluate periodically its knowledge about the network.
Router IP address	Available only when Media property is set to **IP**. This option allows to define the KNX router IP address. If this property is left "0.0.0.0", a multicast request is sent (with timing specified in Search Interval property) via TCP/IP to find a valid KNX TCP interface.

Element	Description
PLC Network	This option allows to define a network of devices, by specifying Alias, Router IP address and PLC Model for each node. 
Comm...	If clicked displays the communication parameters setup dialog. 

Element	Description
Polling Time	Defines how often the tags with Polling attribute enabled are requested to the network (seconds).
Transmission Rate	Defines the interval of time between two consecutive write operations performed by the operator panel (milliseconds).

Tag Editor Settings

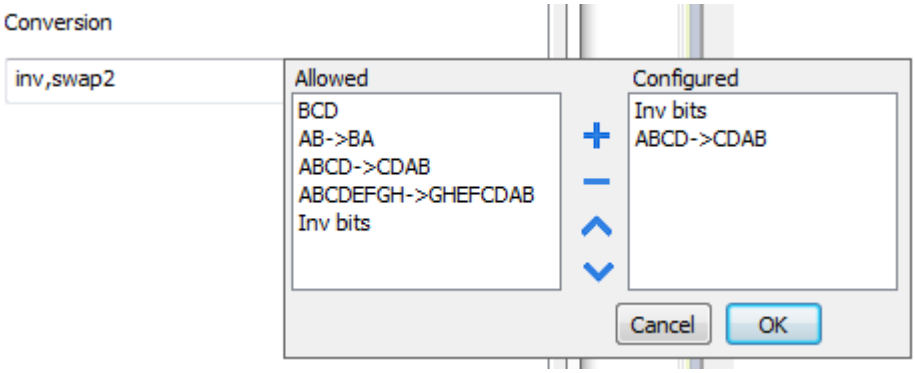
Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **KNX TP/IP** from the **Driver** list: tag definition dialog is displayed.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>KNX resource where tag is located.
<table>
<tr><th>Memory Type</th><th>KNX Data Type</th><th>KNX Datapoint Type</th></tr>
<tr><td>BIT</td><td>Bit</td><td>1.0xx</td></tr>
<tr><td>1BIT</td><td>1 Bit Controlled</td><td>2.0xx</td></tr>
<tr><td>3BIT</td><td>3 Bits Controlled</td><td>3.007</td></tr>
<tr><td>CS</td><td>Character Set</td><td>4.00x</td></tr>
<tr><td>OU</td><td>Octet, Unsigned</td><td>5.00x
17.001
18.001</td></tr>
<tr><td>OS</td><td>Octet, Signed</td><td>6.001
6.010</td></tr>
<tr><td>2OU</td><td>2 Octets, Unsigned</td><td>7.0xx</td></tr>
<tr><td>2OS</td><td>2 Octets, Signed</td><td>8.0xx</td></tr>
<tr><td>2OF</td><td>2 Octets, Float</td><td>9.0xx</td></tr>
<tr><td>TIM</td><td>Time</td><td>10.001</td></tr>
<tr><td>DAT</td><td>Date</td><td>11.001</td></tr>
<tr><td>STR</td><td>String</td><td>16.000
16.001</td></tr>
<tr><td>4OU</td><td>4 Octets, Unsigned</td><td>12.001</td></tr>
<tr><td>4OS</td><td>4 Octets, Signed</td><td>13.0xx</td></tr>
<tr><td>4OF</td><td>4 Octets, Float</td><td>14.0xx</td></tr>
<tr><td>ACC</td><td>Access</td><td>15.000</td></tr>
<tr><td>U1</td><td>Uncertain (1 byte)</td><td>Uncertain</td></tr>
<tr><td>U2</td><td>Uncertain (2 Bytes)</td><td>Uncertain</td></tr>
<tr><td>U3</td><td>Uncertain (3 Bytes)</td><td>Uncertain</td></tr>
<tr><td>U4</td><td>Uncertain (4 Bytes)</td><td>Uncertain</td></tr>
<tr><td>Programming Mode</td><td colspan="2" rowspan="2">Check "Special Data Types" chapter for details</td></tr>
<tr><td>Individual Address</td></tr>
</table>

For some KNX Datapoint Types it may be needed to apply the “Scaling” functionality, available from Tag editor. The next figure shows an example of scaling conversion for Percent values of dimmer actuators (Datapoint Type 5.001 DPT_Scaling). Applying this</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>Scaling conversion, the “0/0/1 % Value” tag manage values in range 0÷100 instead of standard range 0÷255 of Unsigned Octet.
 
</td></tr>
<tr><td>GroupAddress H</td><td>High Group Address of KNX resource.
Range: 0 - 31</td></tr>
<tr><td>GroupAddress M</td><td>Middle Group Address of KNX resource.
Range: 0 - 2047
Available only if PLC Model property is set to Three-stage addressing.
Check "Protocol Editor Settings" chapter for details.</td></tr>
<tr><td>GroupAddress L</td><td>Low Group Address of KNX rerource.
Range: 0 - 255</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…</td></tr>
<tr><td>Arraysize</td><td>• In case of array tag, this property represents the number of array elements.
• In case of string tag, this property represents the maximum number of bytes available in the string tag.
Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.</td></tr>
</table>

Element	Description
Conversion	Conversion to be applied to the Tag.  Depending on data type selected, the **Allowed** list shows one or more conversions, listed below.

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format)

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>0 10000000110
00011100101110110110010001011010000111001010110000001
→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>Separate the byte in two nibbles, and reads them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
<tr><td>KNX_DATE</td><td rowspan="3">Check "Special Data Types" chapter for details</td></tr>
<tr><td>KNX_TIME</td></tr>
<tr><td>KNX_DayOfWeek</td></tr>
</table>

Select the conversion and click on plus button. The selected item will be added on Configured list.

If more conversions are configured, they will be applied in order (from top to bottom of Configured list).

Use the arrow buttons to order the configured conversions.
</td></tr>
<tr><td>Polling</td><td>If checked, this option allows to force continuous read requests from the HMI to the Tag. The timing of polling requests is defined from "Polling Time" option available in "Comm..." window.
Check "Protocol Editor Settings" chapter for details.</td></tr>
</table>

Special Data Types

Programming Mode

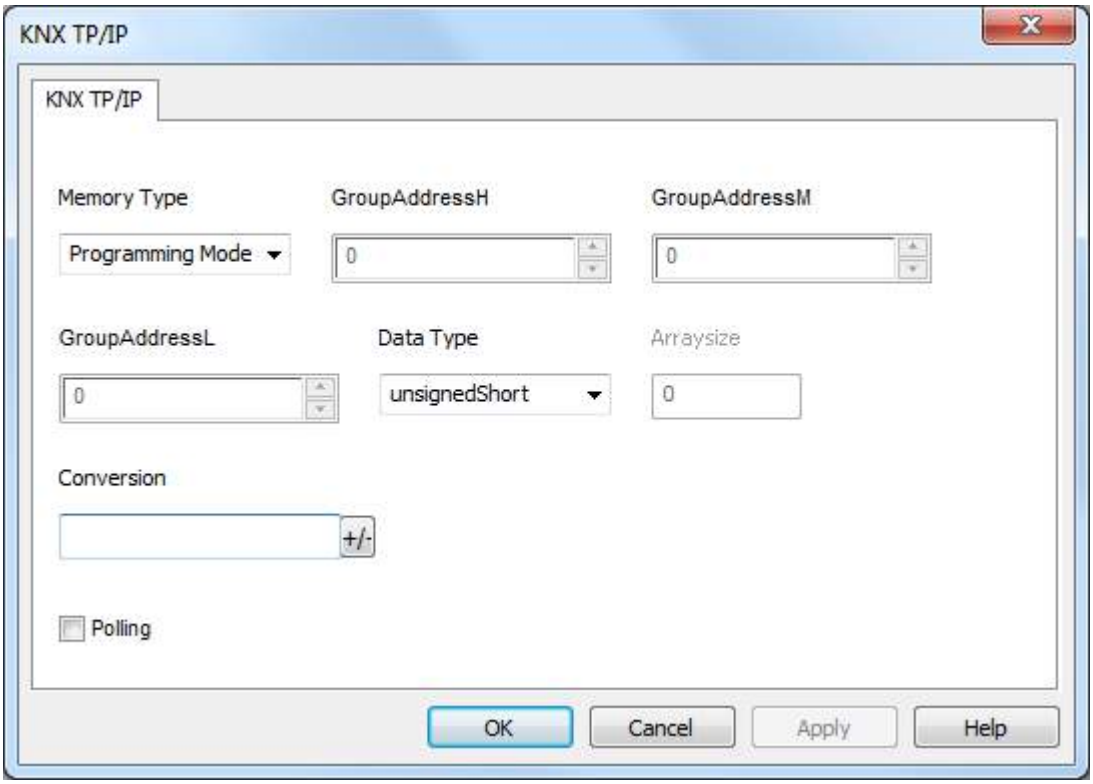
Programming Mode is a special device operating mode that allows changing some system parameters. It is common to most KNX TP devices.

Programming Mode for Individual Address programming via ETS can be set directly in the HMI device.

The first time a HMI project made for the KNX TP communication driver is downloaded to an HMI panel, the unit is assigned the specified Physical Address.

Programming Mode for the HMI panel can be enabled by placing on the screen a widget assigned to the Programming Mode internal variable.

At present there are no database files that can be imported in ETS, so the HMI device can’t be programmed using ETS software. The Programming Mode is available only for future functions.



The “Programming Mode” value can be 0 or 1.

Individual Address

The Individual Address can be displayed placing on the HMI screen an object for “Individual Address” data type.

The Individual Address can be alternatively assigned directly on HMI screen with a write operation to the internal variable.

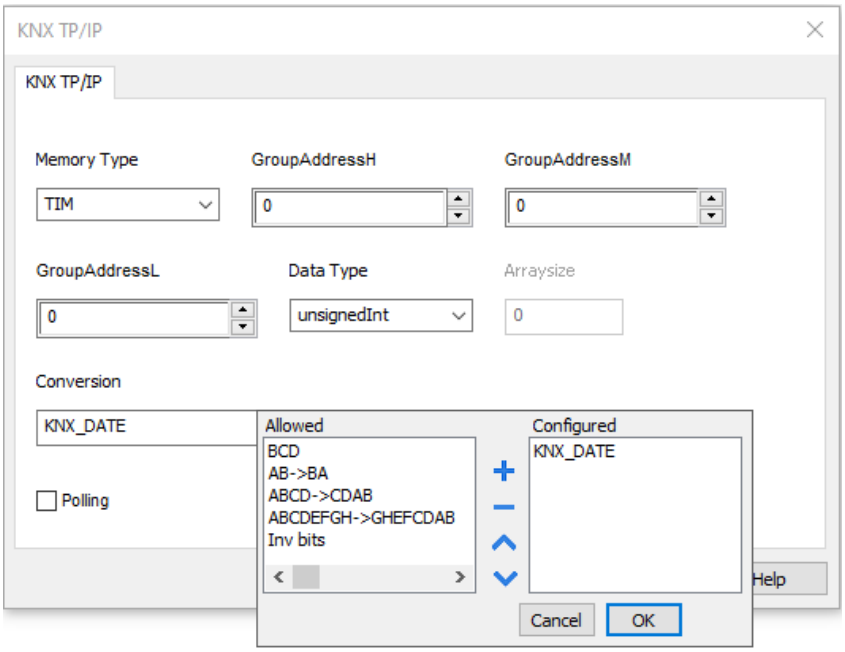
Please note that, as any other KNX device, also the HMI device must have unique Individual Address in a KNX network.

In the following figure an example of how the individual address in hex format has to be interpreted.

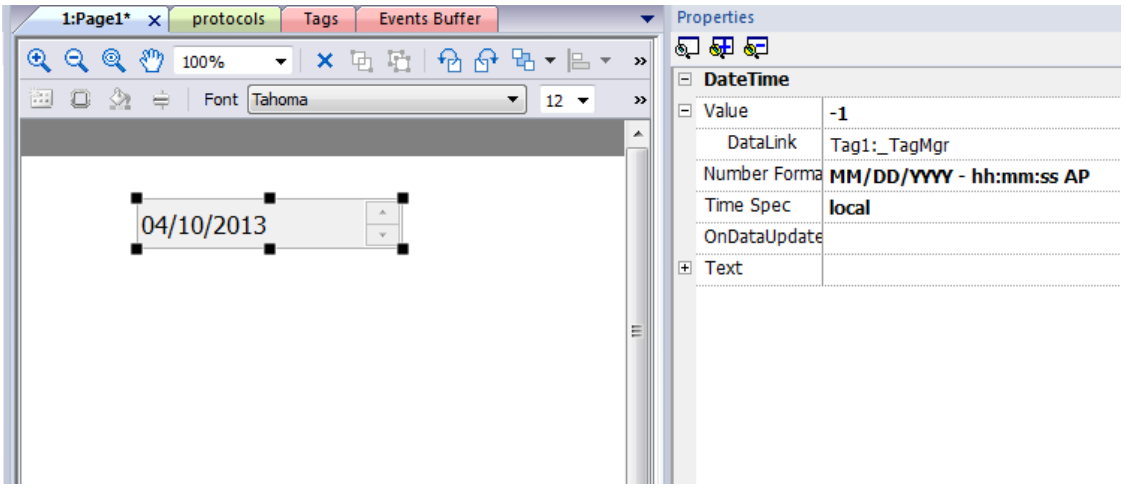
Note: The max value for Individual address is 15.15.255

Date

The Date data type requires a special data conversion.

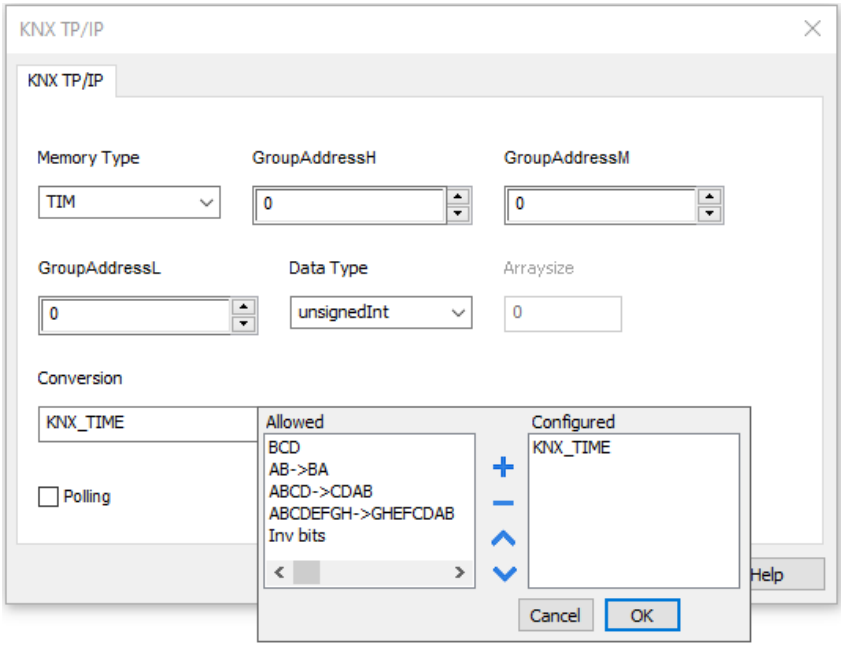


The correct visualization of the date information from this tag can be achieved using the widget dedicated to handle “DateTime” data source.

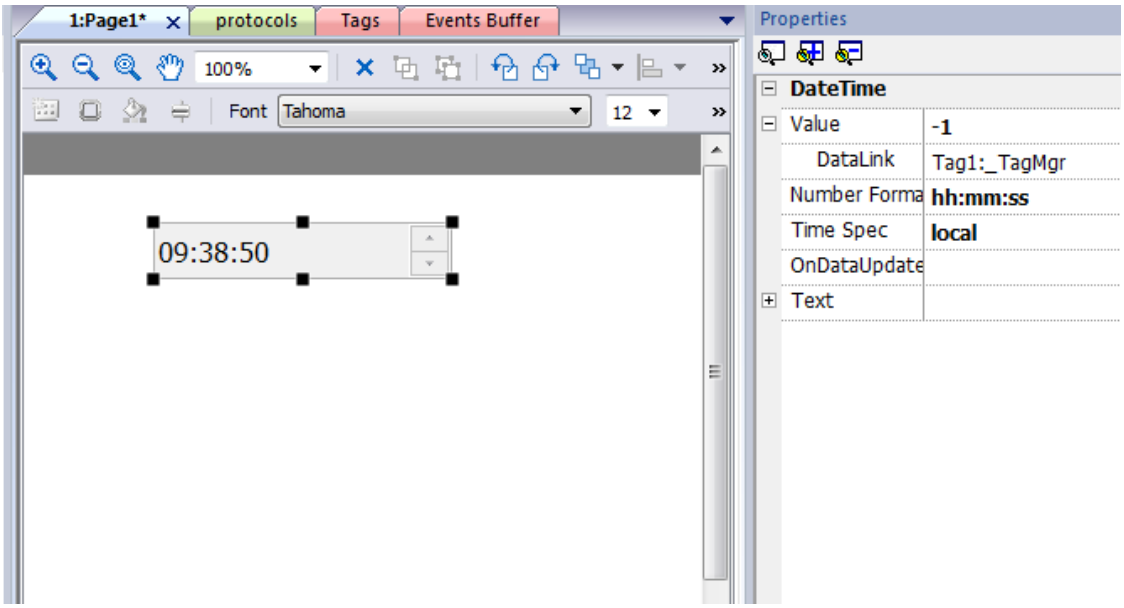


Time

The Time data type requires a special data conversion.



The correct visualization of the time information from this tag can be achieved using the widget dedicated to handle “Time” data source.

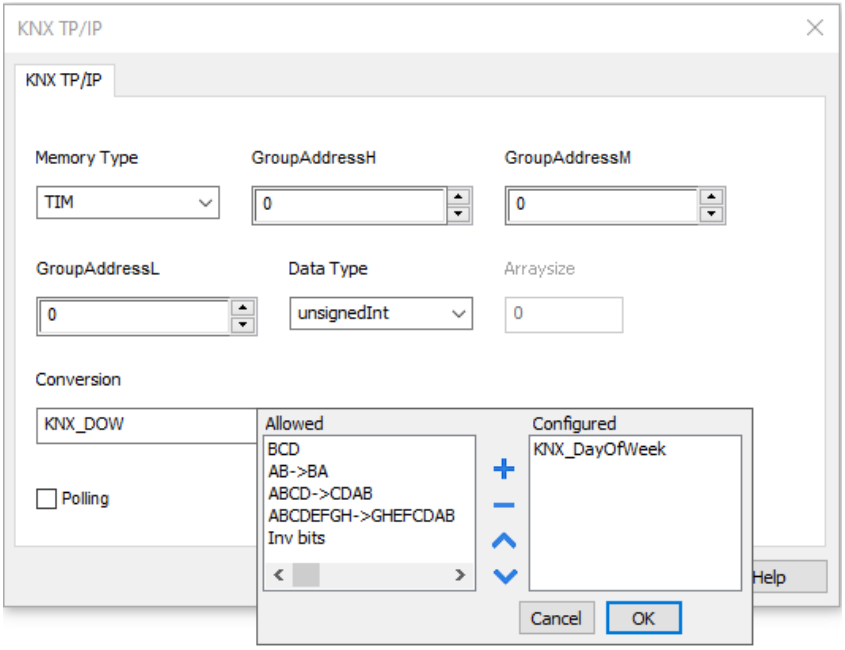


Note: In the “DateTime” widget it is important to set properly the “Time Spec” property in order to avoid the influence on the visualization of the HMI clock timezone and DST settings; Select Number format properly.

Note: Write operation from HMI to KNX network will be executed only with “No Day” information.

Day of Week

The Day of Week data type is part of Time telegram and requires a special data conversion.

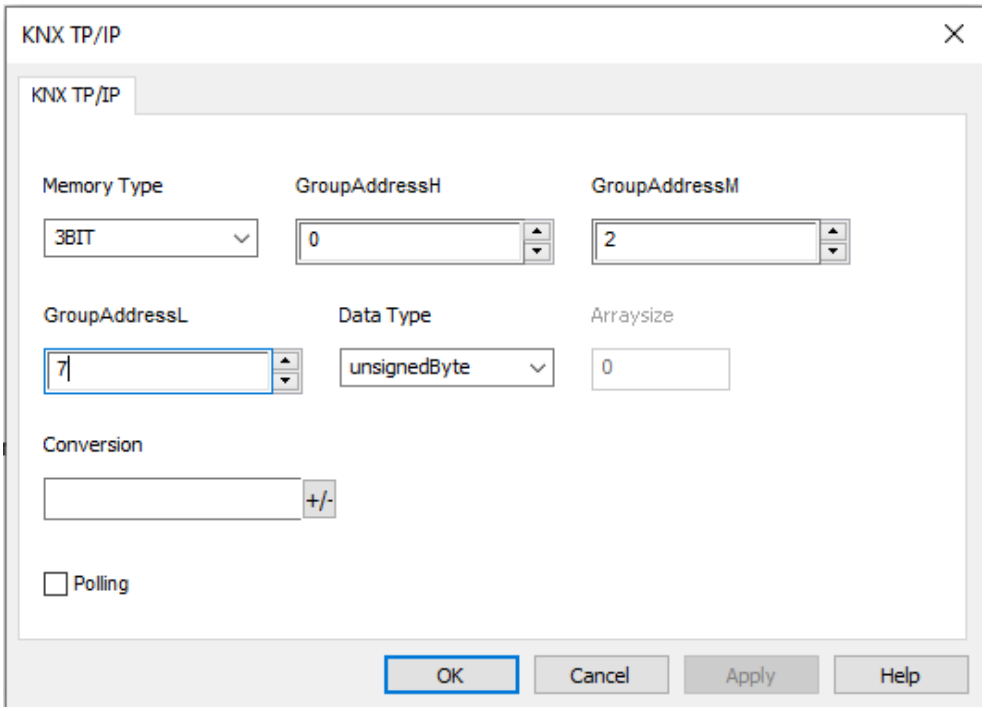


Note: This object is in read-only mode

Dimming function

3 Bits Controlled data type has to be used to operate a dimming function.
This is a 4 bit data where the 1st bit is used to determine if increment or decrement the value and the remaining 3 bits determines the percentage of dimming applied.

The Tag will represent a fixed percentage value (from 0% to 100%) of increasing or decreasing of a particular device value.

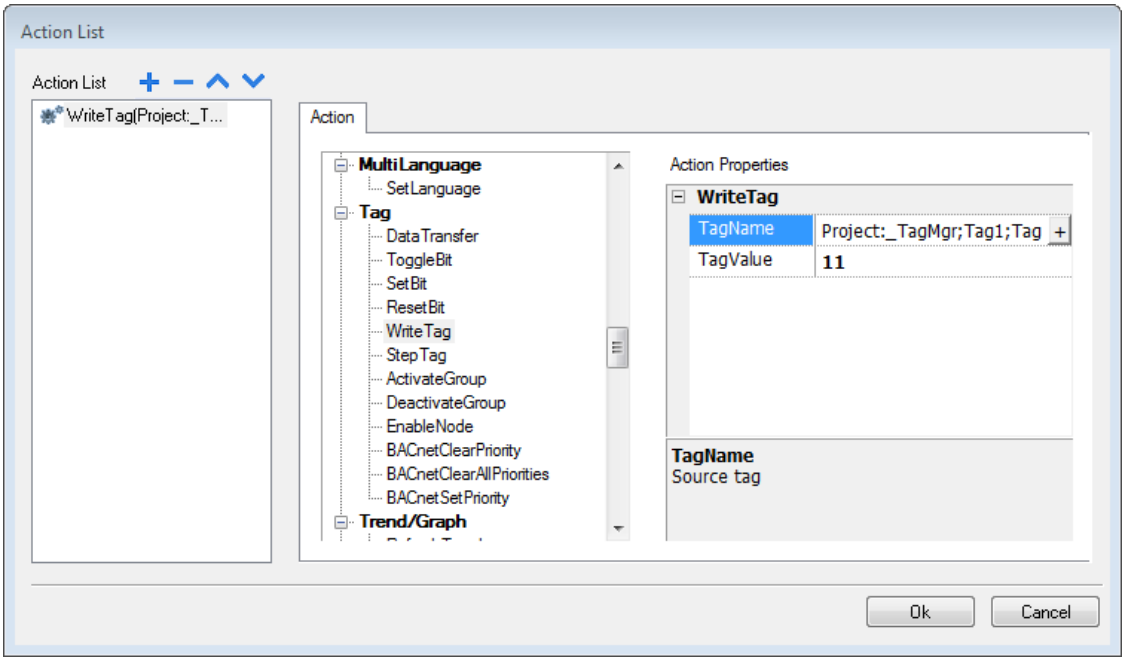


The table below reports the action performed for each value assumed by the Tag.
For example, to increase the dimmed value of 25% it is necessary to write into the Tag that manages the dimming the binary value “1011”, which in decimal code, corresponds to “11”.

Direction	Data	Action
0	001	Down 100%
0	010	Down 50%
0	011	Down 25%
0	100	Down 12%
0	101	Down 6%
0	110	Down 3%
0	111	Down 1%
1	001	Up 100%
1	010	Up 50%
1	011	Up 25%

Direction	Data	Action
1	100	Up 12%
1	101	Up 6%
1	110	Up 3%
1	111	Up 1%

As mentioned before to increase the dimmed value by 25% it is necessary to write 11 in the corresponding Tag.
To do this a Write Tag action programmed as shown in the next figure must be created.



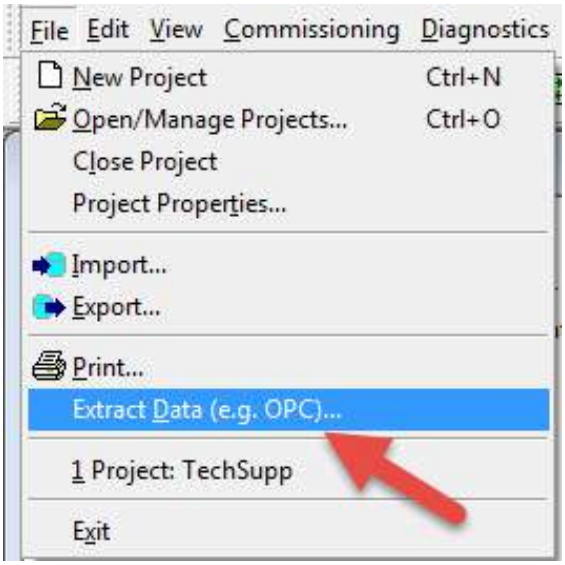
Tag Import

Exporting Tags from PLC

The KNX TP/IP driver supports the Tag import facility. The import filter accepts symbol files with extension “.esf” created by the ETS programming tools.

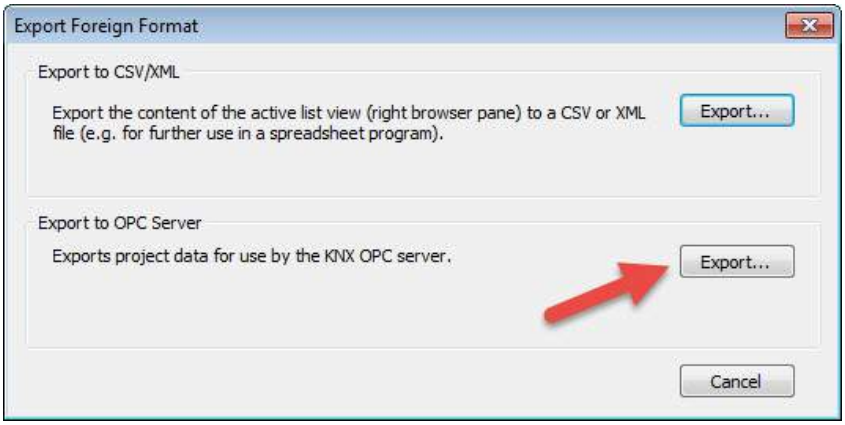
The ETS configuration software can export the database information related to group addresses.

To export database information select “Extract data” from the File menu of ETS software.



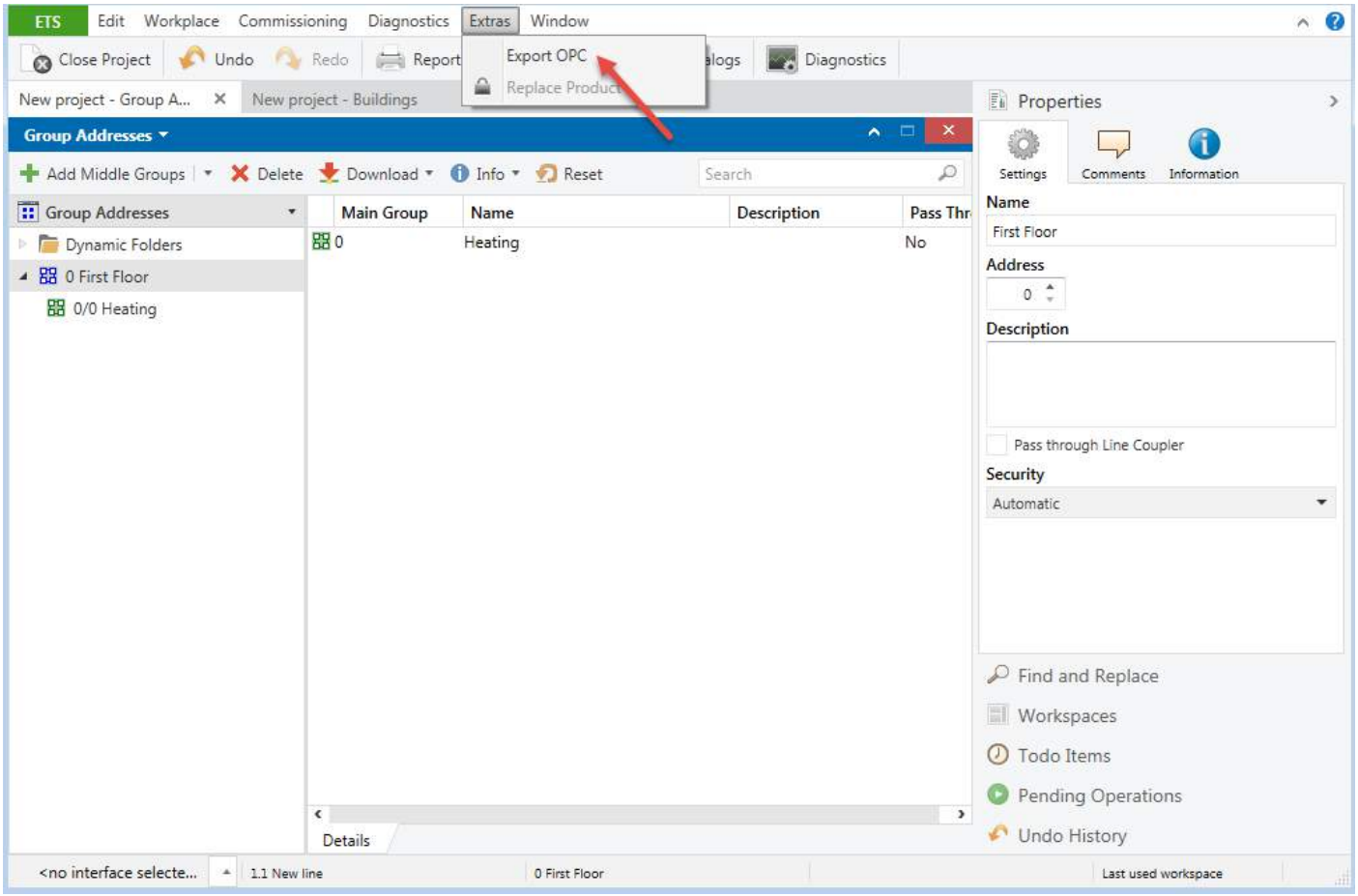
Select the option “Export to OPC Server” to export data in “.esf” format.

Clicking on “Export...” creates the “.esf” file to be imported in the Tag Editor.

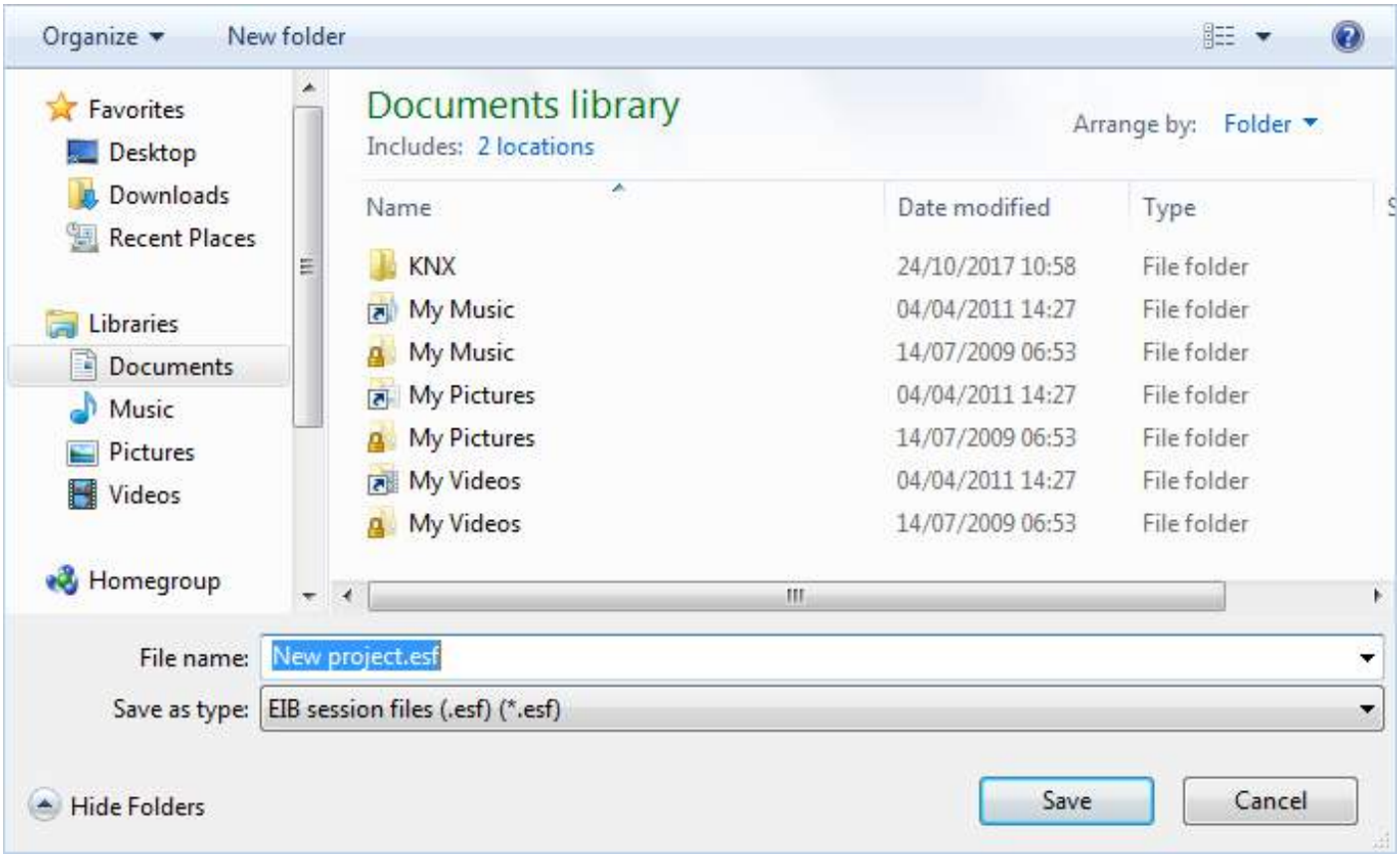


Exporting Tags from PLC using ETS5

From ETS5 programming software click on *Extras* > *Export OPC*

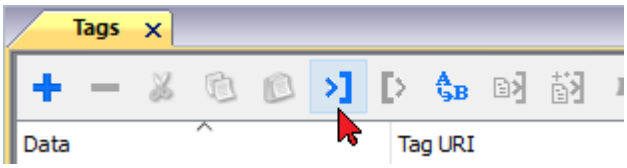


In next step select location and file name for .esf file.

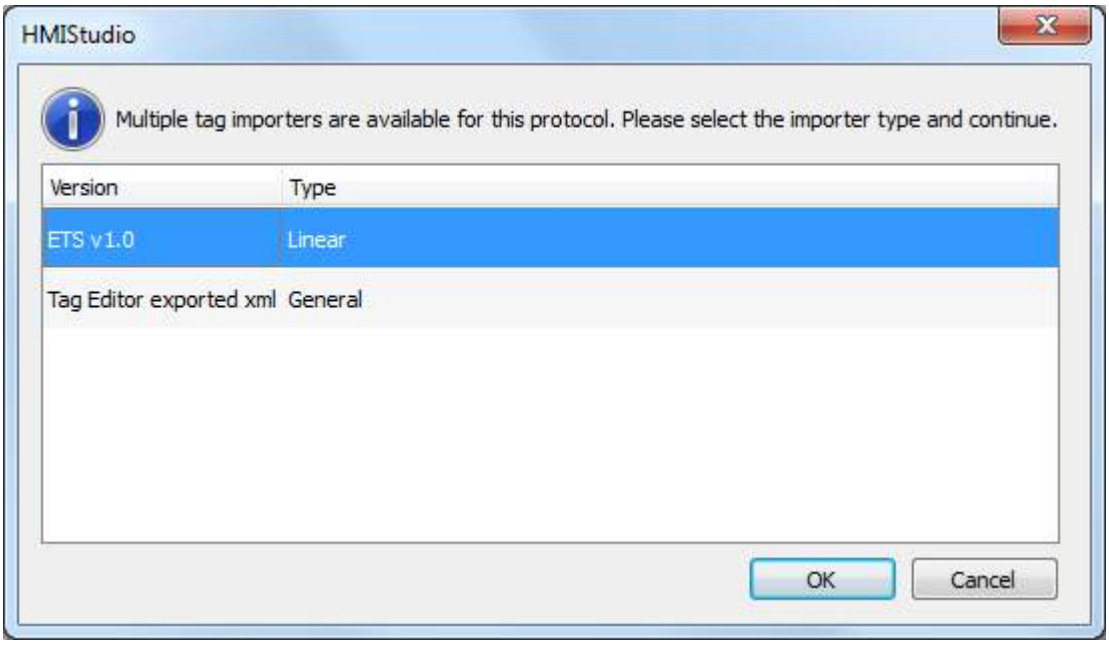


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



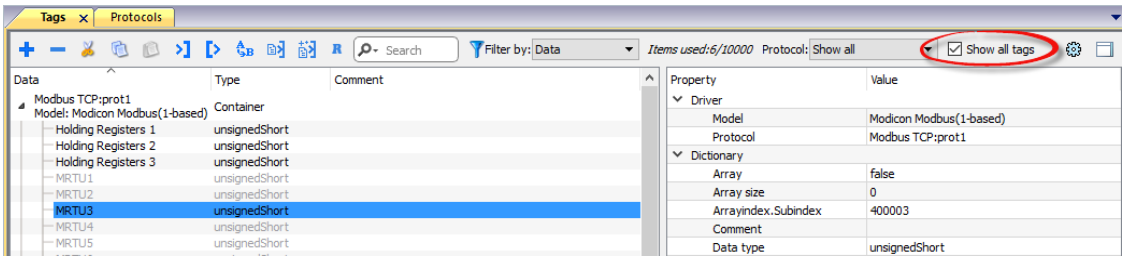
The following dialog shows which importer type can be selected.

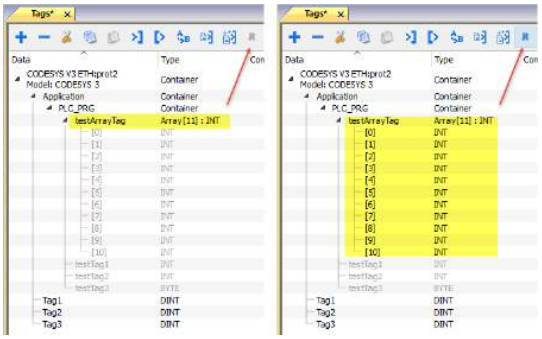
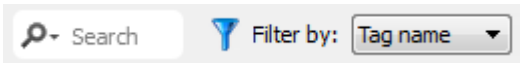


Importer	Description
ETS v1.0 Linear	Requires a **.esf** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The status codes supported for this communication driver are:

Error	Notes
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Response error	The tag requested by the panel may be not available in the system or communication session completed with errors
General Error	Error cannot be identified; should never be reported; contact technical support
Internal software error	Unrecognized error

Lenze CANopen

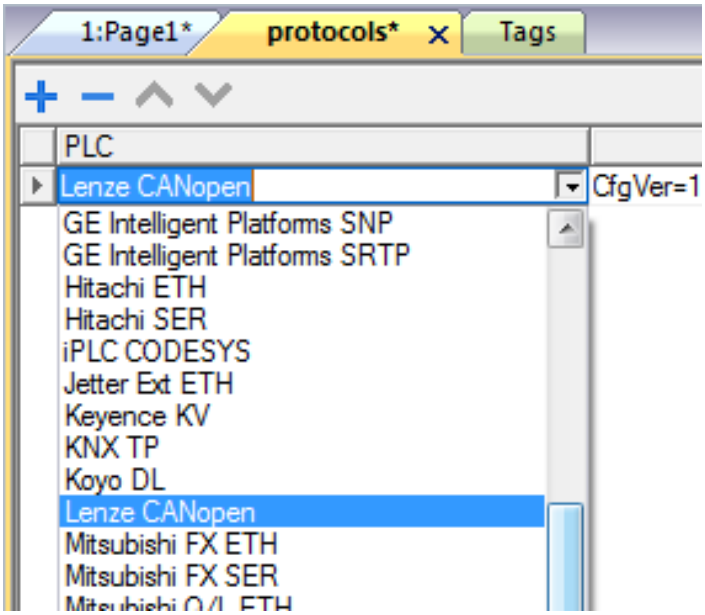
The Lenze CANopen communication driver has been designed to connect HMI products to Lenze controllers using the CANopen network. A new device communication profile has been developed to takes advantage from the advanced user interface features of the software, while retaining the simple networking concept supported by the CANopen network.

Connection to CANopen networks requires the optional CANopen communication module.

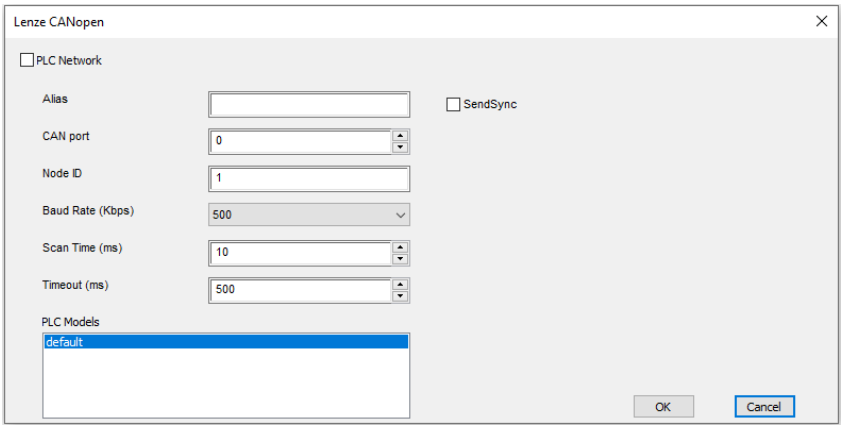
Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Please ensure that the latest driver is used in the application.

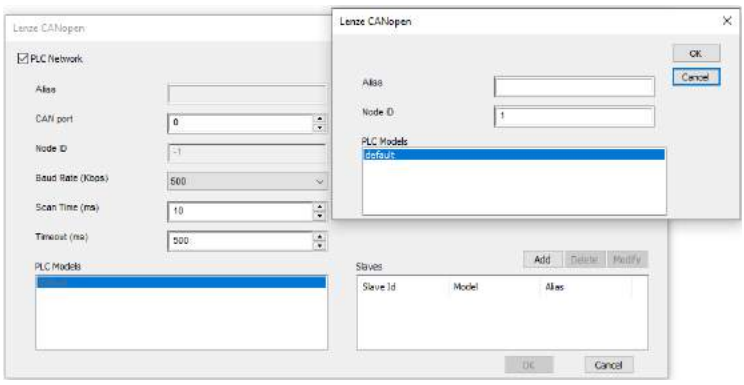
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “Lenze CANopen” from the list of available protocols.



The protocol configuration dialog is shown in figure.

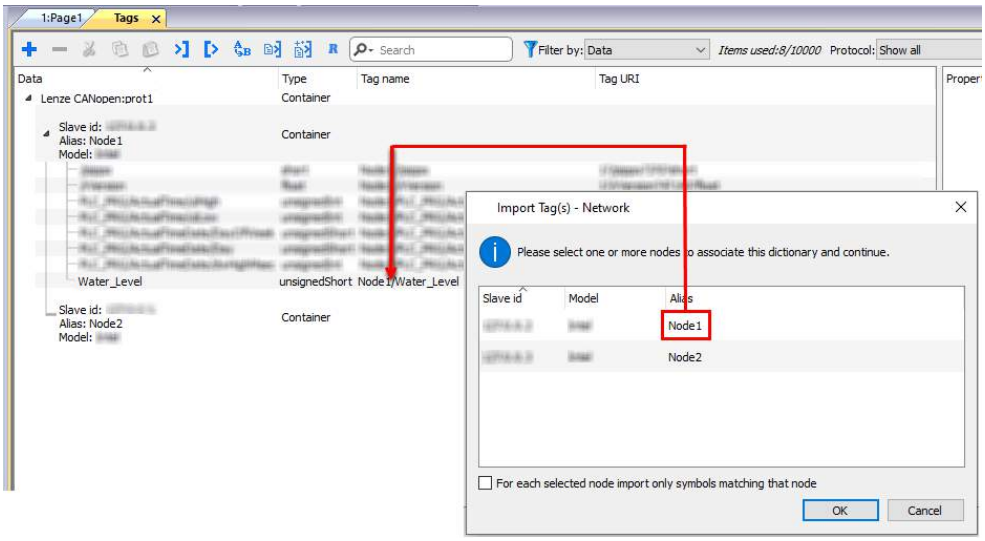


Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
Node ID	CANopen Node ID assigned to the slave device
Baud Rate (kbps)	Speed of the CANopen network
Scan Time (ms)	Scan time is dependent upon your specific process or application requirements and the capabilities of your controller.
Timeout (ms)	Maximum allowed time the driver will wait for a response from the device before reporting a communication error
SendSync	The Sync-Producer provides the synchronization-signal for the Sync-Consumer. When the Sync-Consumers receive the signal they start carrying out their synchronous tasks.
PLC Models	This version supports only one device model.
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and enter the node ID per each slave you need to access. 

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.

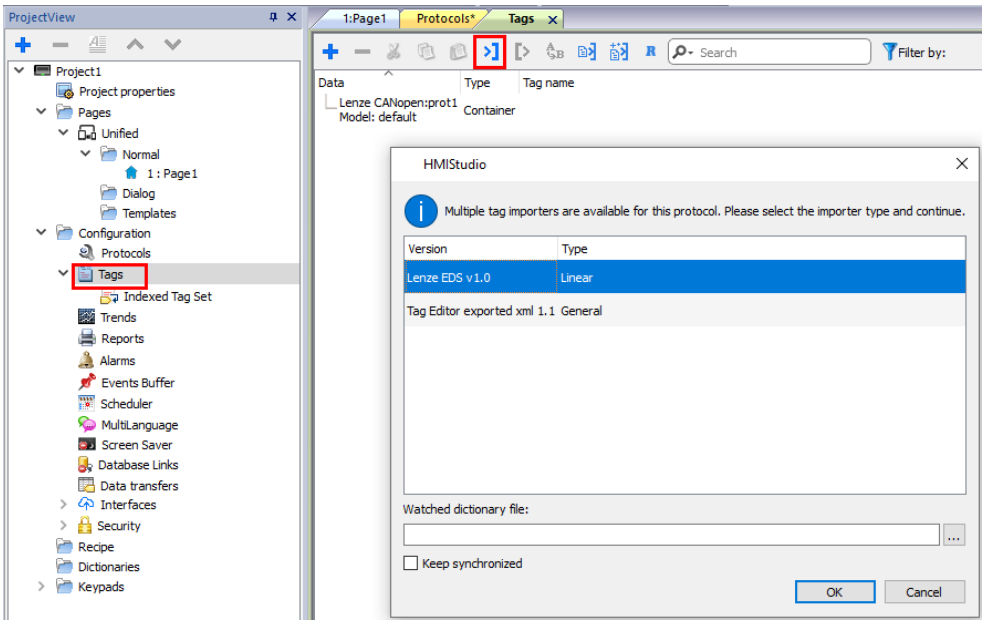


Note: Aliasing tag names is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Tag Import

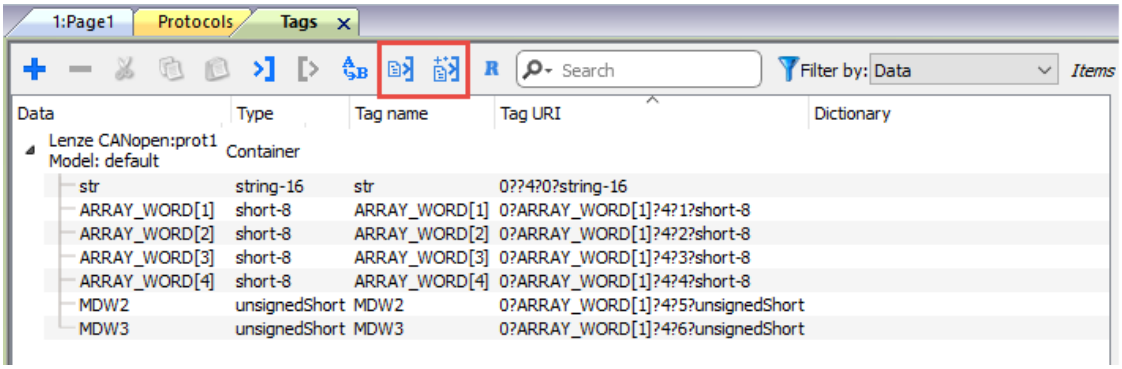
The Lenze CANopen driver supports the Tag import facility. The import filter accepts symbol files with extension ".eds" provided by Lenze, the device manufacturer.

In Tag Editor select the communication driver and click on the "Import tag" button to start the importer.



Once the importer has been selected, locate the symbol file and click Open.

The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the "add tags" button as shown in figure.



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
No response	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Invalid access to var	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Wrong answer frame from server	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
Var is too long	The panel did receive from the controller a response, but its length exceeded the max length admitted ensure the data programmed in the project are consistent with the controller resources.

Modbus RTU

The operator panels can be connected to a Modbus network as the network master using this communication driver.

Implementation details

The Modbus RTU implementation supports only a subset of the Modbus standard RTU function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the device Coil area
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
03	Read Holding Registers	Read multiple Registers
04	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
05	Force Single Coil	Forces a single Coil to either ON or OFF
06	Preset Single Register	Presets a value in a Register
16	Preset Multiple Registers	Presets value in multiple Registers

Note: Communication speed with controllers is supported up to 115200 baud.

Note: Floating point data format is IEEE standard compliant.

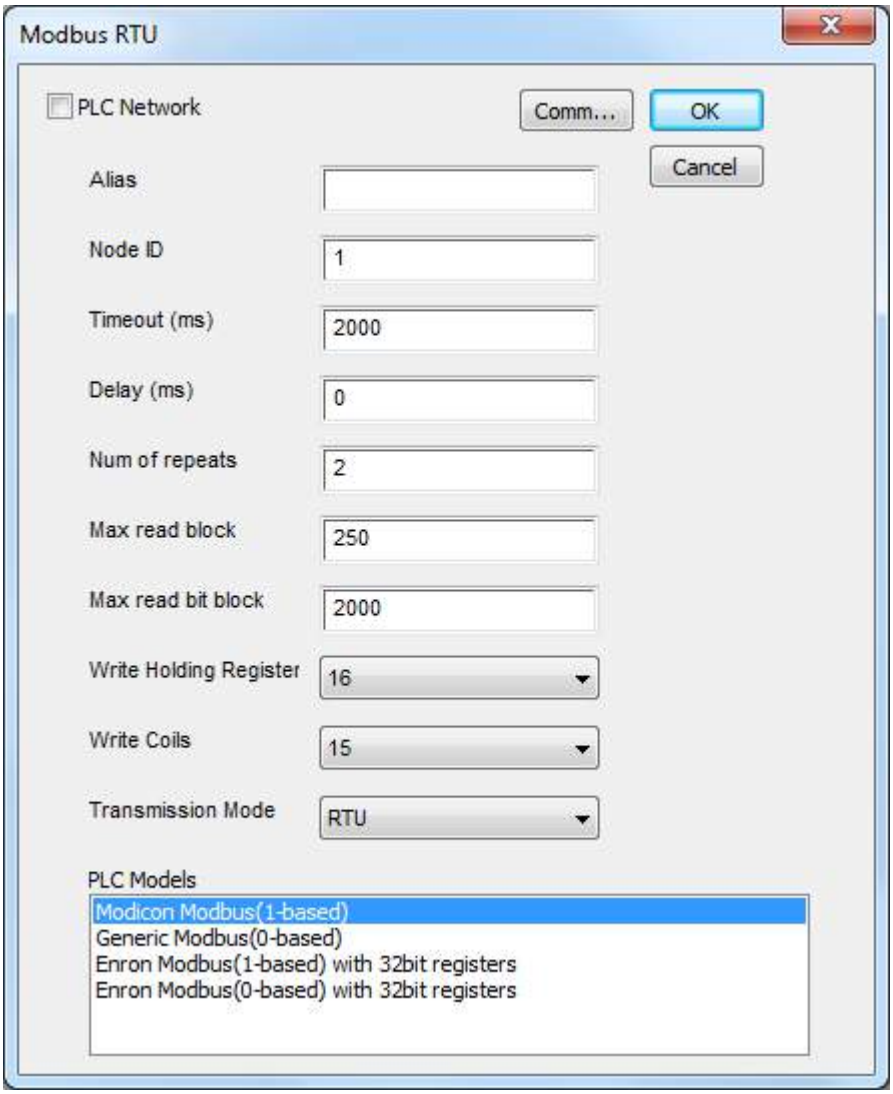
Protocol Editor Settings

Adding a protocol

To configure the protocol:

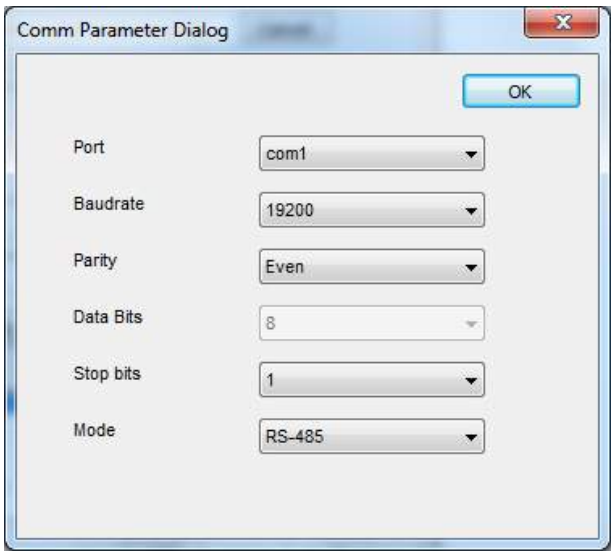
1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Node ID	Modbus node of the slave device.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
Delay (ms)	Time delay in milliseconds between the end of the last received frame and the starting of a new request. If set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
Num of repeats	Number of times a certain message will be sent to the controller before reporting the communication error status. When set to 1 the panel will report the communication error if the response to the first request packet is not correct.

Element	Description
Max read block	Maximum length in bytes of a data block request. It applies only to read access of Holding Registers.
Max read bit block	Maximum length in bits of a block request. It applies only to read access of Input Bits and Output Coils.
Write Holding Register	Modbus function for write operations to Holding Registers. Select between the function **06** (preset single register) and function **16** (preset multiple registers). If function **06** is selected, the protocol will always use function **06** for writing to the controller, even when writing to multiple consecutive registers. If function **16** is selected, the protocol will always use function **16** to write to the controller, even for a single register write request and the **Max read block size** parameter of the query is set to **2**. The use of function **16** may result in higher communication performance.
Write Coils	Modbus function for write operations to Output Coils. Select between the function **05** (write single coil) and function **15** (write multiple coils). If Modbus function **05** is selected, the protocol will always use function **05** for writing to the controller, even when writing to multiple consecutive coils. If Modbus function **15** is selected, the protocol will always use function **15** to write to the controller, even for a single coil write request. The use of function **15** may result in higher communication performance.
Transmission Mode	• **RTU**: use RTU mode • **ASCII**: use ASCII mode Note: When PLC network is active, all nodes will be configured with the same Transmission Mode.
PLC Models	Allows to select between different PLC models: • **Modicon Modbus (1-based)**: Modbus implementation where all resources starts with offset 1. • **Generic Modbus (0-based)**: Modbus implementation where all resources starts with offset 0. • **Enron Modbus (1-based)**: Extends Modicon Mobdus implementation with 32 bit registers memory area. • **Enron Modbus (0-base)**: Extends Generic Modbus implementation with 32 bit registers memory area. Note: The address range used in the Modbus frames is always between 0 and 65535 for the Holding Registers and between 0 and 65535 for Coils.
Comm...	If clicked displays the communication parameters setup dialog.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>


<table>
<tr><th>Element</th><th>Parameter</th></tr>
<tr><td>Port</td><td>Serial port selection.
COM1: On-board port
COM2: Optional Plug-in module plugged on slot#1 or slot#2
COM3: Optional Plug-in module plugged on slot#3 or slot#4</td></tr>
<tr><td>Baudrate, Parity, Data Bits, Stop bits</td><td>Serial line parameters.</td></tr>
<tr><td>Mode</td><td>Serial port mode. Available modes:
RS-232.
RS-485 (2 wires).
RS-422 (4 wires).</td></tr>
</table>
</td></tr>
<tr><td>PLC Network</td><td>Multiple controllers can be connected to one HMI device. To set-up multiple connections, select PLC networkand click Add to configure each slave</td></tr>
</table>

Tag Editor Settings

*Path: **ProjectView**> **Config** > double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Modbus RTU** from the protocol list: tag definition dialog is displayed.


Modbus RTU
Modbus RTU
Memory Type
Offset
SubIndex
Input bits
100001
0
Data Type
Arraysize
Conversion
boolean
0
+/-
OK
Cancel
Apply
Help

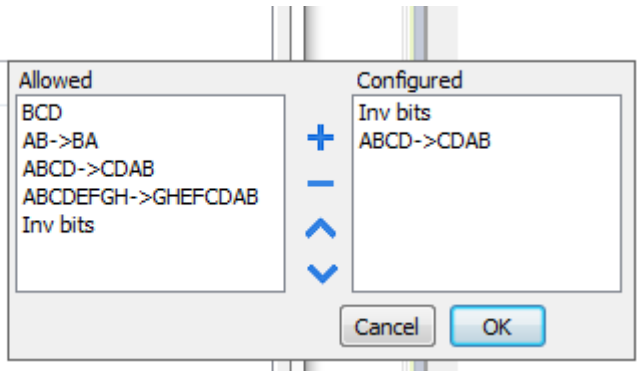
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Modbus resource where tag is located.
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>Coil Status</td><td>Coils</td></tr>
<tr><td>Input Status</td><td>Discrete Input</td></tr>
<tr><td>Input Registers</td><td>Input Registers</td></tr>
<tr><td>Holding Registers</td><td>Holding Registers</td></tr>
<tr><td>32 bit Registers</td><td>32 bit registers memory area.
Available only for Enron Modbus PLC Models</td></tr>
<tr><td>Node Override ID</td><td rowspan="8">protocol parameter (see Special Data Types for mode details)</td></tr>
<tr><td>Modicon Mode</td></tr>
<tr><td>Serial Baudrate</td></tr>
<tr><td>Serial Parity</td></tr>
<tr><td>Serial Stop Bits</td></tr>
<tr><td>Serial Mode</td></tr>
<tr><td>Serial Done</td></tr>
</table></td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.
Offset addresses are six digits composed by one digit data type prefix + five digits resource address.
<table>
<tr><th>Memory Type</th><th>Studio
Offset range</th><th>Modicon
Offset range</th><th>Generic Modbus
Offset range</th></tr>
<tr><td>Coil Status</td><td>0 – 65535</td><td rowspan="5">1 – 65536</td><td rowspan="5">0 – 65535</td></tr>
<tr><td>Input Status</td><td>100000 – 165535</td></tr>
<tr><td>Input Registers</td><td>300000 – 365535</td></tr>
<tr><td>Holding Registers</td><td>400000 – 465535</td></tr>
<tr><td>32 bit Registers</td><td>0 – 65535</td></tr>
</table></td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the tag.</td></tr>
</table>

Element	Description
	Conversion inv,swap2  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000

Element	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>001 → 1 10000011100 10101010000101000101101101101100101101100001001111 101 (in binary format)</td></tr><tr><td>BCD</td><td>**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)</td></tr></table> Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

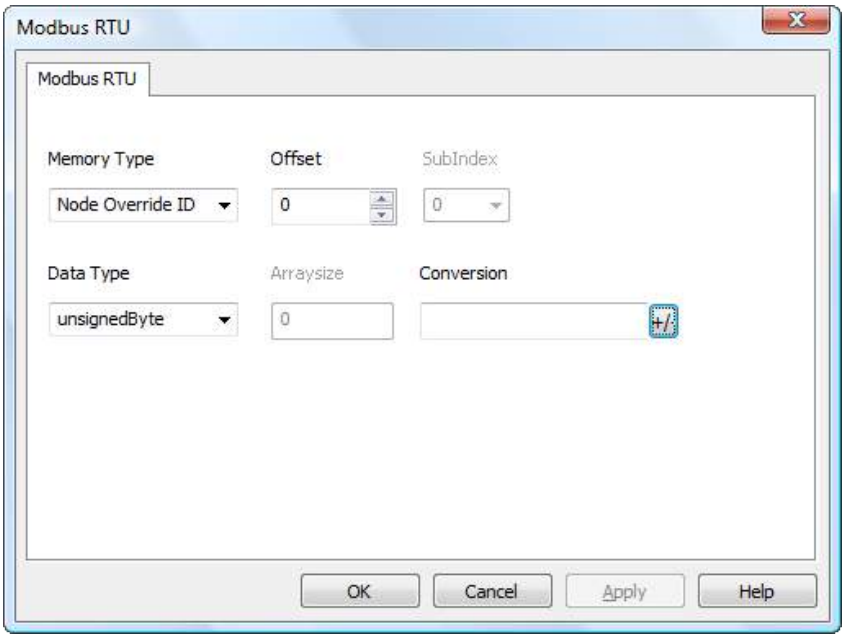
Node Override ID

The protocol provides the special data type Node Override ID which allows you to change the node ID of the slave at runtime. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Override ID	Modbus operation
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.

Note: Node Override ID value assigned at runtime is retained through power cycles.



Modicon Mode

The protocol provide a special data type that can be used to override the Modicon Mode parameter at runtime.

Modicon Mode	Description
0	Generic Modbus (0-based). Register indexes start from 0.
1	Modicon Modbus (1-based). Register indexes start from 1.

Note: Modicon Mode parameter value assigned at runtime is retained through power cycles.

Serial Parameters Override

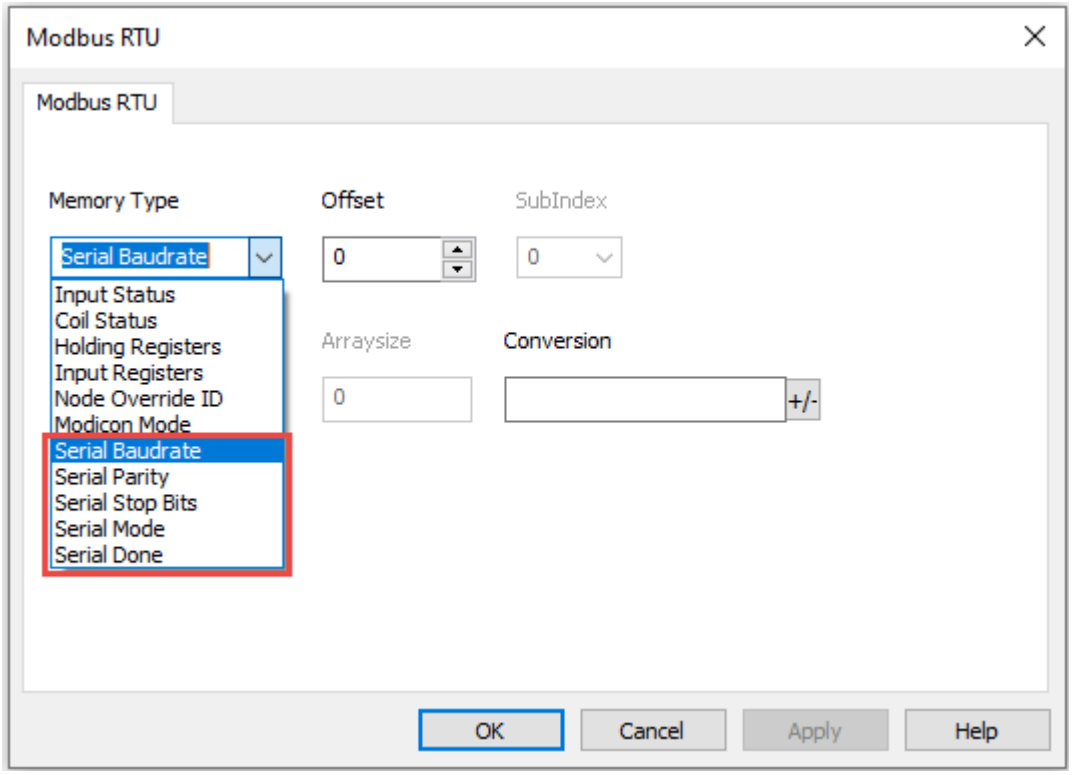
The protocol provide special data types that can be used to override the serial parameters at runtime.

Parameter	Description
Serial Baudrate	unsigned 32 bit value for baudrate overriding. Possible values are 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
Serial Parity	unsigned 8 bit value for parity overriding. Possible values are described in the following list.

Value	Description
0	none parity
1	even parity
2	odd parity

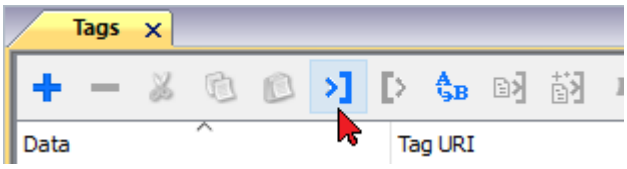
Serial Stop Bits	unsigned 8 bit value for stop bits overriding. Possible values are 1, 2.
Serial Mode	unsigned 8 bit value for serial mode overriding. Possible values are described in the following list.

Parameter	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>**0**</td><td>RS-232 mode</td></tr><tr><td>**1**</td><td>RS-485 mode</td></tr><tr><td>**2**</td><td>RS-422 mode</td></tr></table>
Serial Done	Set to 1 to overwrite the communication line parameters. The parameters are processed all together only when this variable is set to value 1

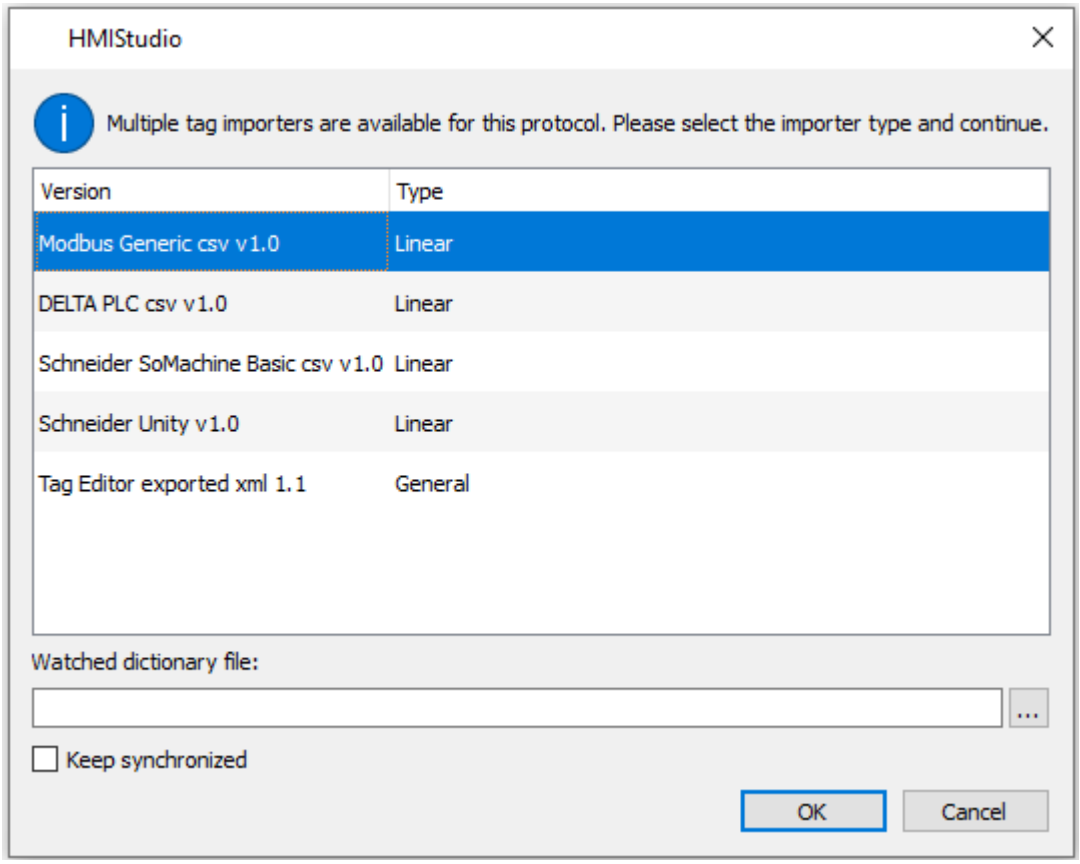


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



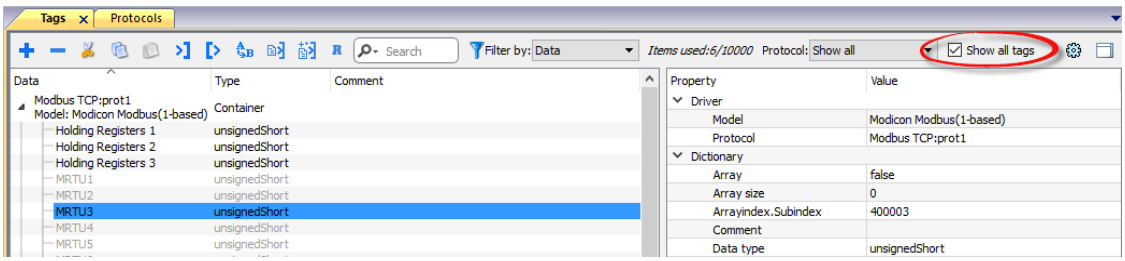
The following dialog shows which importer type can be selected.

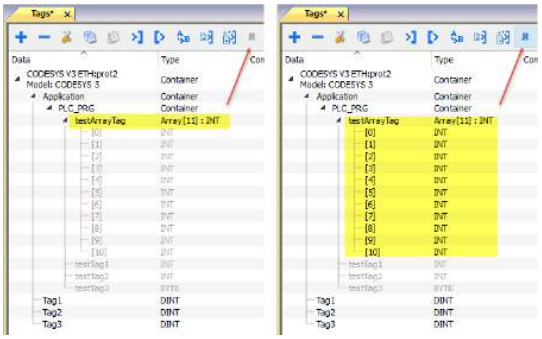
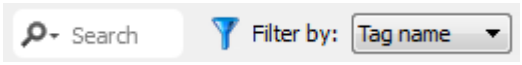


Type	Description
Modbus Generic csv v1.0 **Linear**	Requires a **.csv** file. All variables will be displayed at the same level.
DELTA PLC csv v1.0	Requires a **.csv** file. All variables will be displayed at the same level.
Schneider Unity v1.0 **Linear**	Requires a **.uny** file. The file containing symbols must be exported in **.txt** format and later renamed as **.uny**. The importer considers only variables located at fixed address and disregards arrays of strings. All other arrays, except for boolean type, are expanded.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Modbus Generic csv file structure

This protocol supports the import of tag information when provided in **.csv** format according to the following format:

```
NodeID, TagName, MemoryType, Address, DataFormat,...,[Comment]
```

Note: Fields in brackets are optional as well as fields between Data Format and Comment.

Field	Description
NodeID	Node the tag belongs to
TagName	Tag description
MemoryType	• OUTP • INP • IREG • HREG
Address	Offset compatible with Modbus notation
DataFormat	Data type in internal notation. See "Programming concepts" section in the main manual.
Comment	Optional additional description.

Tag file example

Example of .csv line:

```
2,Holding Register 1, HREG, 400001, unsignedShort,
```

Note: This line has no comment. When the Comment is missing, the comma as a terminator character is mandatory.

Communication status

Current communication status can be displayed using System Variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
No response	No reply within the specified timeout.	Check if the controller is connected and properly configured to get network access.
Incorrect node address in response	The device received a response with an invalid node address from the controller .	-
The received message too short	The device received a response with an invalid format from the controller .	-
Incorrect writing data acknowledge	The controller did not accept a write request.	Check if project data is consistent with the controller resources.

Modbus RTU Server

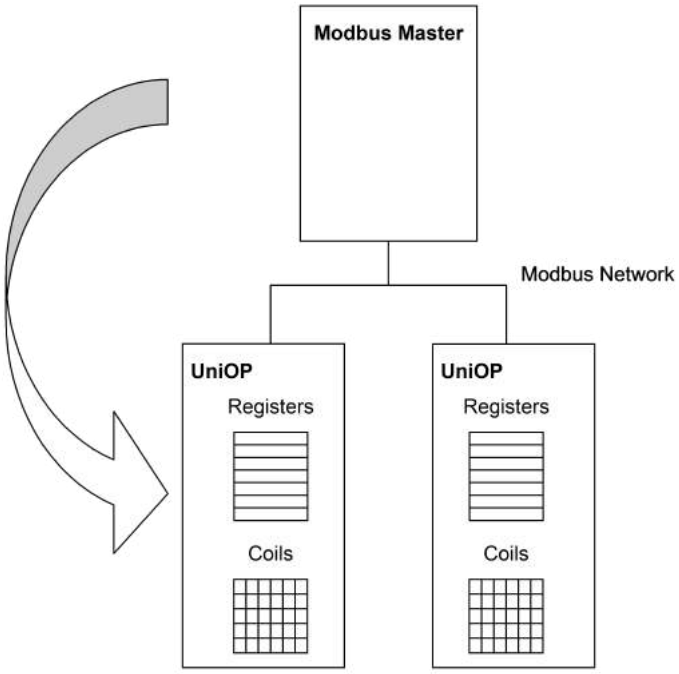
Modbus RTU Server communication driver allows connecting the HMI device as a slave in a Modbus RTU network. Standard Modbus messages are used for information exchange.

This approach allows connecting HMI devices to SCADA systems through the universally supported Modbus RTU communication protocol.

Principle of operation

This communication driver implements a Modbus RTU slave unit in the HMI device. A subset of the complete range of Modbus function codes is supported. The available function codes allow data transfer between the master and the slave.

The following diagram shows the system architecture.



The HMI device is actually simulating the communication interface of a PLC: Coils and Registers are respectively boolean and 16 bit integers.

The device always access data in its internal memory. Data can be transferred to and from the Modbus Master only on initiative of the Master itself.

Implementation details

This Modbus RTU slave implementation supports only a subset of the standard Modbus function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the device Coil area.
03	Read Holding Registers	Read multiple device Registers.

Code	Function	Description
05	Force Single Coil	Forces a single device Coil to either ON or OFF.
06	Preset Single Register	Presets a value in a device Register.
08	Loopback Diagnostic Test	Only sub function 00 (Return Query Data) is supported.
15	Force Multiple Coils	Forces multiple device Coils to either ON or OFF.
16	Preset Multiple Registers	Presets value in multiple device Registers.
17	Report Slave ID	Returns diagnostic information of the controller present at the slave address.
23	Read Write Multiple Registers	Read & presets values in multiple device Registers

Exception Codes

Code	Description
01	**Illegal Function**. the function code received in the query is not supported
02	**Illegal Data Address**. Data Address received in the query exceeds the predefined data range (see **Tag Definition** for detailed ranges of all types).
03	**Illegal Data Value**. A sub function other than 00 is specified in Loopback Diagnostic Test (Code 08).

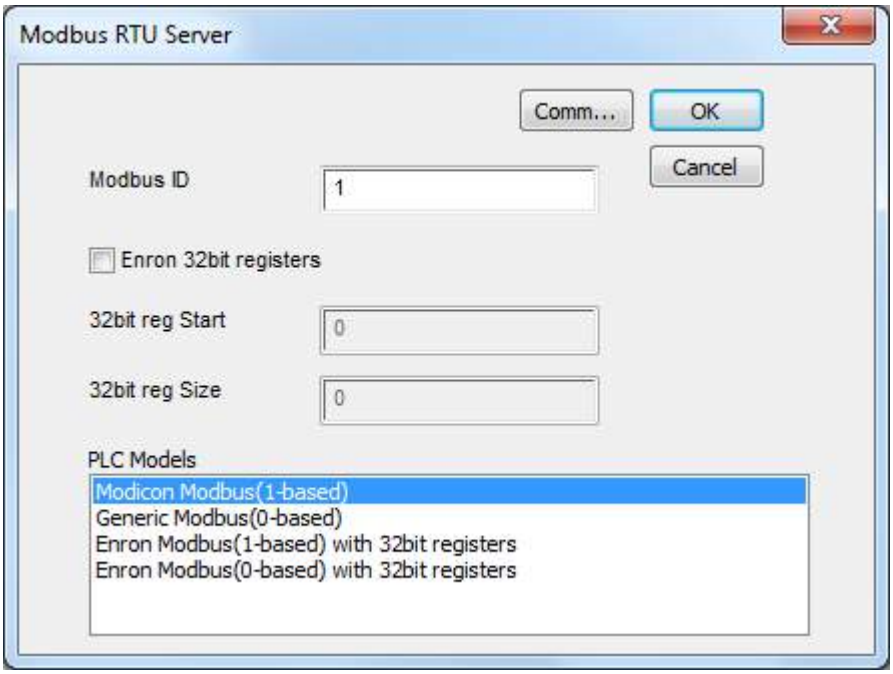
Protocol Editor Settings

Adding a protocol

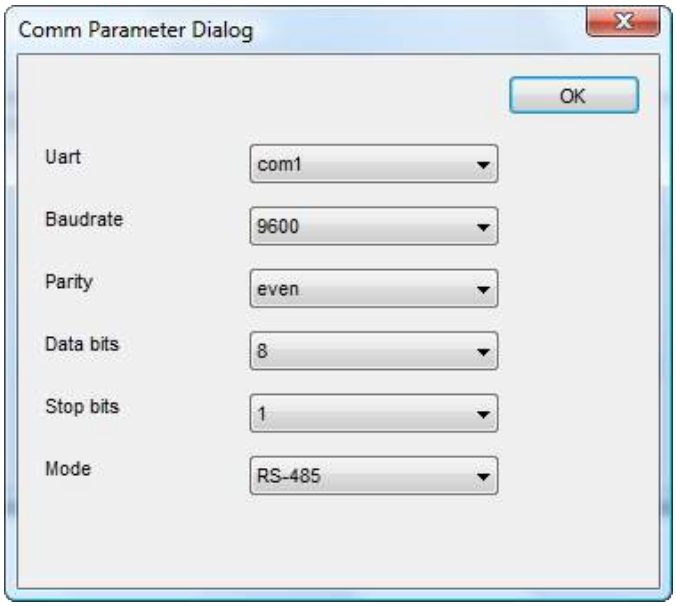
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Modbus ID	Modbus node ID. Every Modbus server device in the network must have its own Modbus ID.
Enron 32bit registers	If selected, allows to define the first register address and the number of registers for 32 bit registers memory area. Note: 32 bit registers are available only for **Enron Modbus** PLC Models.
32bit reg Start **32bit reg Size**	32 bit registries memory area definition. **Start** value represents the first register address. **Size** value represents the number of registries. Note: A request to one of the registries inside this area gives a 4 byte answer.
PLC Models	Allows to select between different PLC models: • **Modicon Modbus (1-based)**: Modbus implementation where all resources starts with offset 1. • **Generic Modbus (0-based)**: Modbus implementation where all resources starts with offset 0. • **Enron Modbus (1-based)**: Extends Modicon Mobdus implementation with 32 bit registers memory area. • **Enron Modbus (0-base)**: Extends Generic Modbus implementation with 32 bit registers memory area.

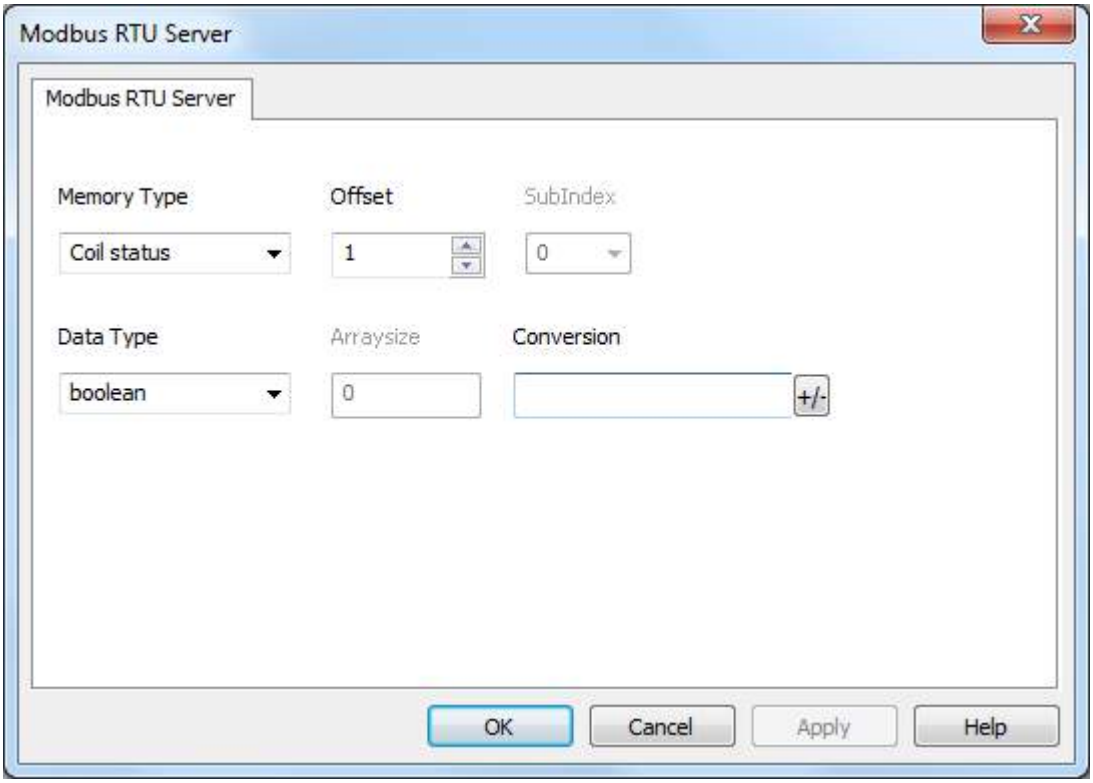
Element	Description
	Note: The address range used in the Modbus frames is always between 0 and 65535 for the Holding Registers and between 0 and 65535 for Coils.
Comm...	If clicked, displays the communication parameters setup dialog. You have to set parameters according to the values programmed in Modbus Master. 

Element	Description
Uart	Serial port selection. • **COM1**: On-board port • **COM2**: Optional Plug-in module plugged on slot#1 or slot#2 • **COM3**: Optional Plug-in module plugged on slot#3 or slot#4
Baudrate, Parity, Data bits, Stop bits	Serial line parameters.
Mode	Serial port mode. Available options: • **RS-232** • **RS-485** (2 wires) • **RS-422** (4 wires)

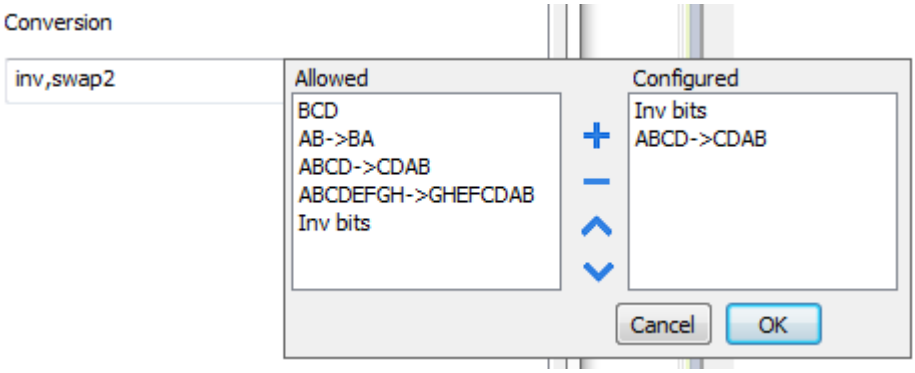
Tag Editor Settings

*Path: **ProjectView**> **Config** > double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Modbus RTU Server** from the protocol list: tag definition dialog is displayed.



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Modbus resource where tag is located.
<table>
<tr><th>Memory Type</th><th>Modbus Resource</th></tr>
<tr><td>Coil Status</td><td>Coils</td></tr>
<tr><td>Input Status</td><td>Discrete Input</td></tr>
<tr><td>Input Registers</td><td>Input Registers</td></tr>
<tr><td>Holding Registers</td><td>Holding Registers</td></tr>
<tr><td>32 bit Registers</td><td>32 bit registers memory area.
Available only for Enron Modbus PLC Models</td></tr>
<tr><td>Node Override ID</td><td rowspan="8">protocol parameter (see Special Data Types for mode details)</td></tr>
<tr><td>Modicon Mode</td></tr>
<tr><td>Serial Baudrate</td></tr>
<tr><td>Serial Parity</td></tr>
<tr><td>Serial Stop Bits</td></tr>
<tr><td>Serial Mode</td></tr>
<tr><td>Serial Done</td></tr>
</table></td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.
Offset addresses are six digits composed by one digit data type prefix + five digits resource address.
<table>
<tr><th>Memory Type</th><th>Studio
Offset range</th><th>Modicon
Offset range</th><th>Generic Modbus
Offset range</th></tr>
<tr><td>Coil Status</td><td>0 – 65535</td><td rowspan="5">1 – 65536</td><td rowspan="5">0 – 65535</td></tr>
<tr><td>Input Status</td><td>100000 – 165535</td></tr>
<tr><td>Input Registers</td><td>300000 – 365535</td></tr>
<tr><td>Holding Registers</td><td>400000 – 465535</td></tr>
<tr><td>32 bit Registers</td><td>0 – 65535</td></tr>
</table></td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>
Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>When configuring array or string tags, this option define the amount of array elements or characters of the string.</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the tag.
 
</td></tr>
</table>

Element	Description
	Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000
001
→
1 10000011100
10101010000101000101101101101100101101100001001111
101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

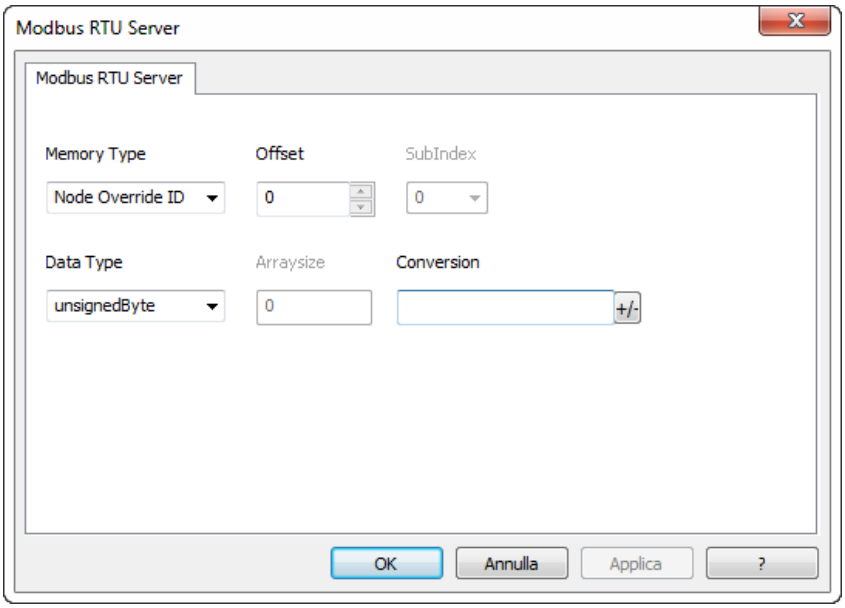
Node Override ID

The protocol provides the special data type Node Override ID which allows you to change the node ID of the slave at runtime. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Override ID	Modbus operation
0	Communication with the slave is stopped. In case of write operation, the device will not respond to request frames.
1 to 255	It is interpreted as the value of the new node ID and is replaced for runtime operation.

Note: Node Override ID value assigned at runtime is retained through power cycles.

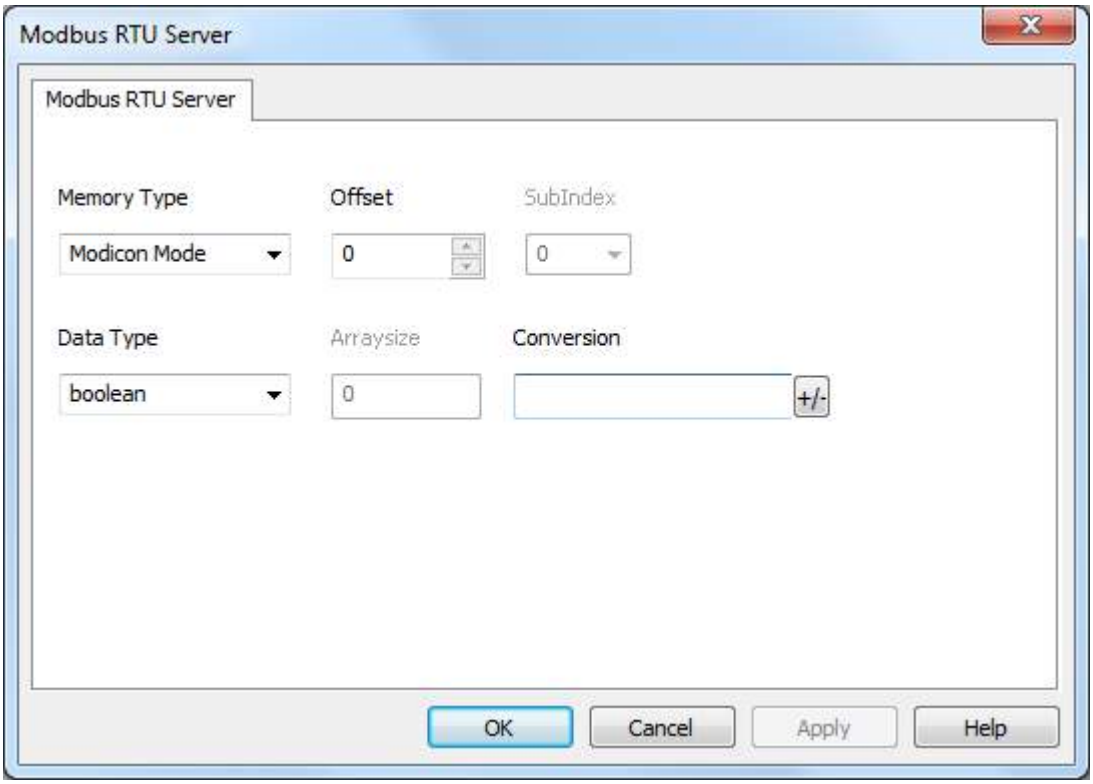


Modicon Mode

The protocol provide a special data type that can be used to override the Modicon Mode parameter at runtime.

Modicon Mode	Description
0	Generic Modbus (0-based). Register indexes start from 0.
1	Modicon Modbus (1-based). Register indexes start from 1.

Note: Modicon Mode parameter value assigned at runtime is retained through power cycles.



Serial Parameters Override

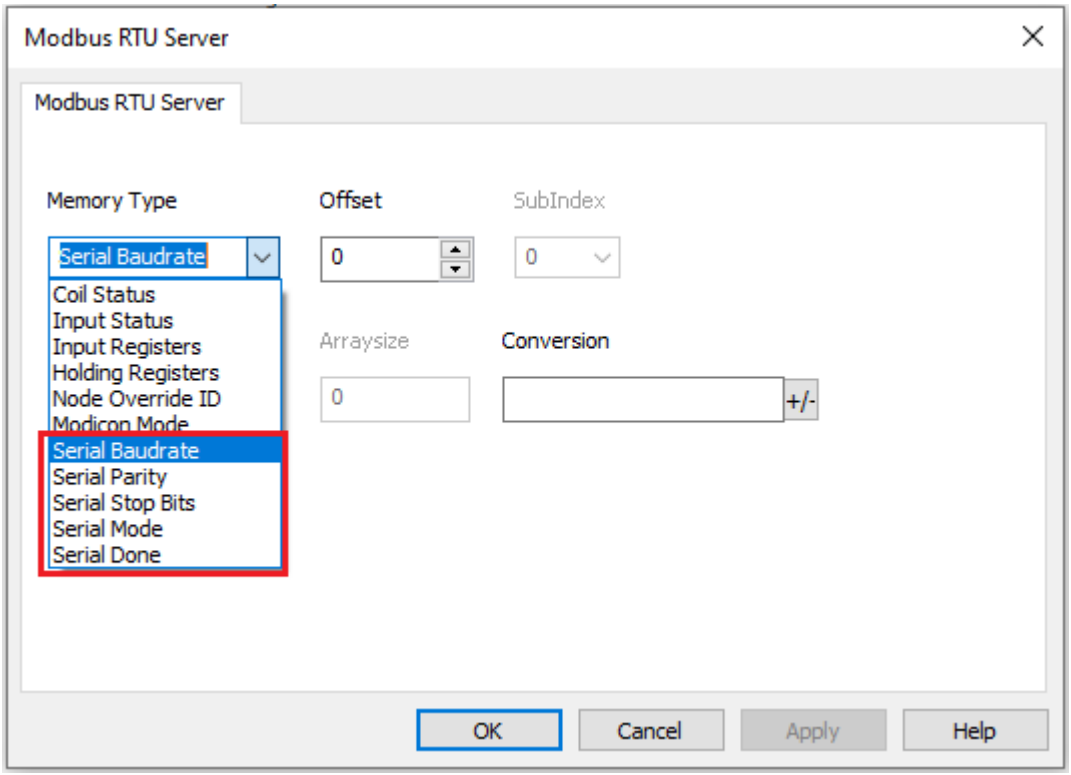
The protocol provide special data types that can be used to override the serial parameters at runtime.

Parameter	Description
Serial Baudrate	unsigned 32 bit value for baudrate overriding. Possible values are 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
Serial Parity	unsigned 8 bit value for parity overriding. Possible values are described in the following list.

Value	Description
0	none parity
1	even parity
2	odd parity

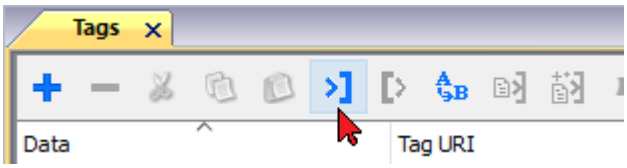
Serial Stop Bits	unsigned 8 bit value for stop bits overriding. Possible values are 1, 2.
Serial Mode	unsigned 8 bit value for serial mode overriding. Possible values are described in the following list.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>RS-232 mode</td></tr>
<tr><td>1</td><td>RS-485 mode</td></tr>
<tr><td>2</td><td>RS-422 mode</td></tr>
</table>
</td></tr>
<tr><td>Serial Done</td><td>Set to 1 to overwrite the communication line parameters. The parameters are processed all together only when this variable is set to value 1</td></tr>
</table>

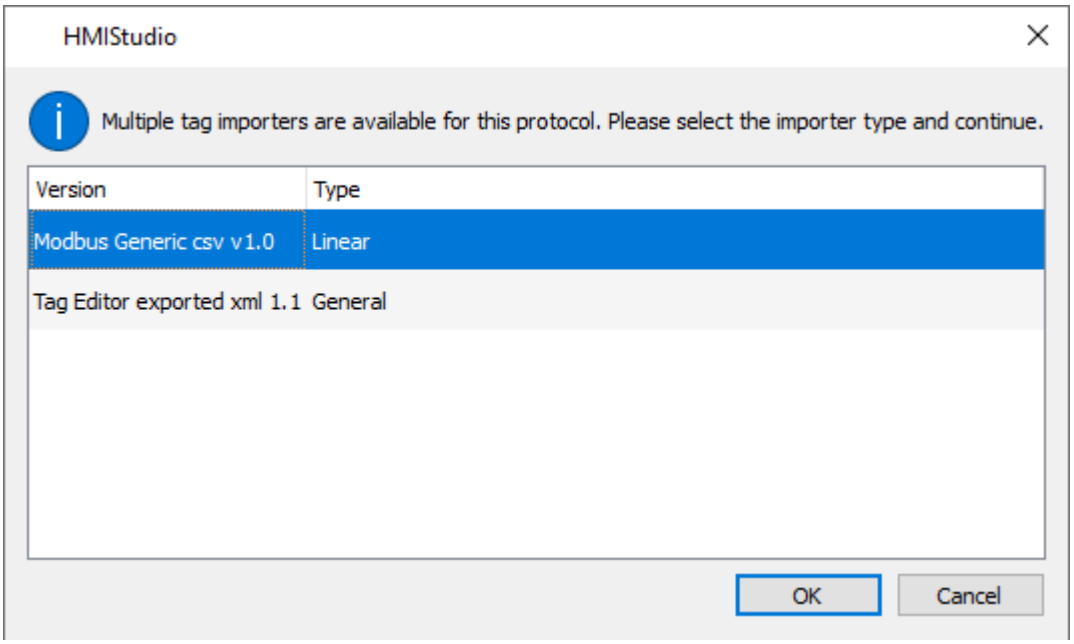


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



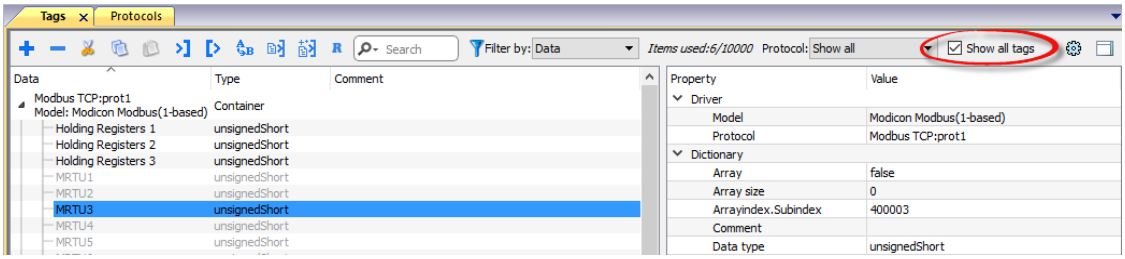
The following dialog shows which importer type can be selected.

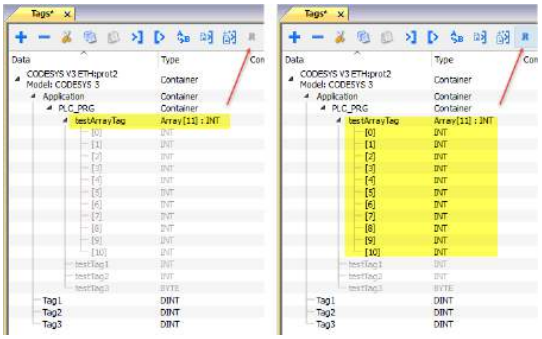
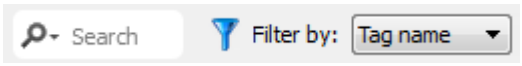


Type	Description
Modbus Generic csv v1.0 **Linear**	Requires a **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Modbus Generic csv file structure

This protocol supports the import of tag information when provided in **.csv** format according to the following format:

```
NodeID, TagName, MemoryType, Address, DataFormat,...,[Comment]
```

Note: Fields in brackets are optional as well as fields between Data Format and Comment.

Field	Description
NodeID	Node the tag belongs to
TagName	Tag description
MemoryType	• OUTP • INP • IREG • HREG
Address	Offset compatible with Modbus notation

Field	Description
DataFormat	Data type in internal notation. See "Programming concepts" section in the main manual.
Comment	Optional additional description.

Tag file example

Example of .csv line:

```
2,Holding Register 1, HREG, 400001, unsignedShort,
```

Note: This line has no comment. When the Comment is missing, the comma as a terminator character is mandatory.

Communication status

Current communication status can be displayed using system variables. This communication protocol acts as server and doesn't return any specific Protocol Error Message.

See "System Variables" section in the main manual.

Modbus TCP

Various Modbus TCP-capable devices can be connected to HMI devices. To set-up your Modbus TCP device, please refer to the documentation you have received with the device.

The implementation of the protocol operates as a Modbus TCP client only.

Implementation details

This Modbus TCP implementation supports only a subset of the Modbus TCP standard function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the HMI device Coil area.
02	Read Input Status	Reads the ON/OFF status of the discrete inputs (1x reference) in the slave.
03	Read Holding Registers	Reads multiple registers.
04	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave.
05	Force Single Coil	Forces a single coil to either ON or OFF.
06	Preset Single Register	Writes a value to one register.
15	Write Multiple Coils	Writes each coil in a sequence of coils to either ON or OFF.
16	Preset Multiple Registers	Writes values to a block of registers in sequence.

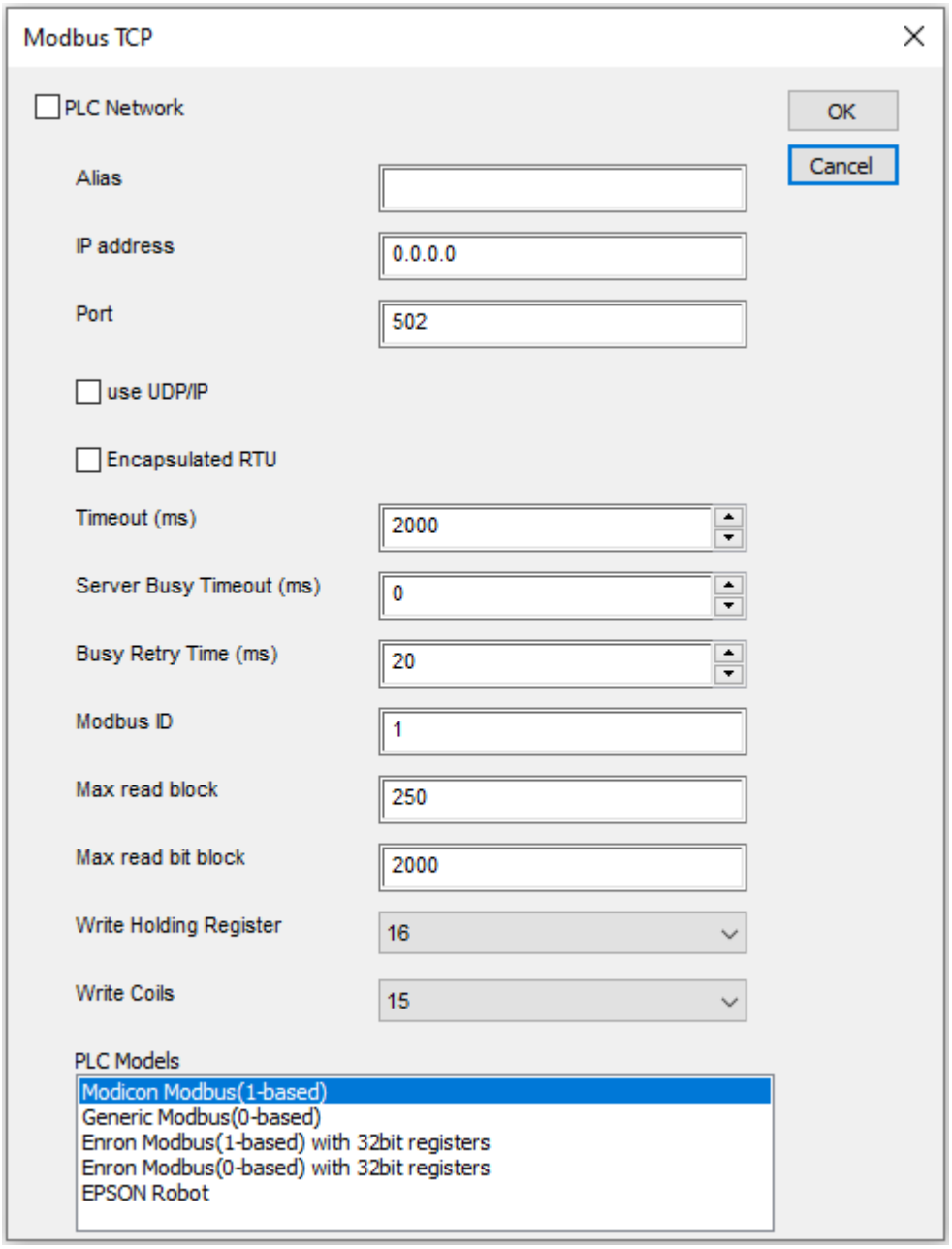
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

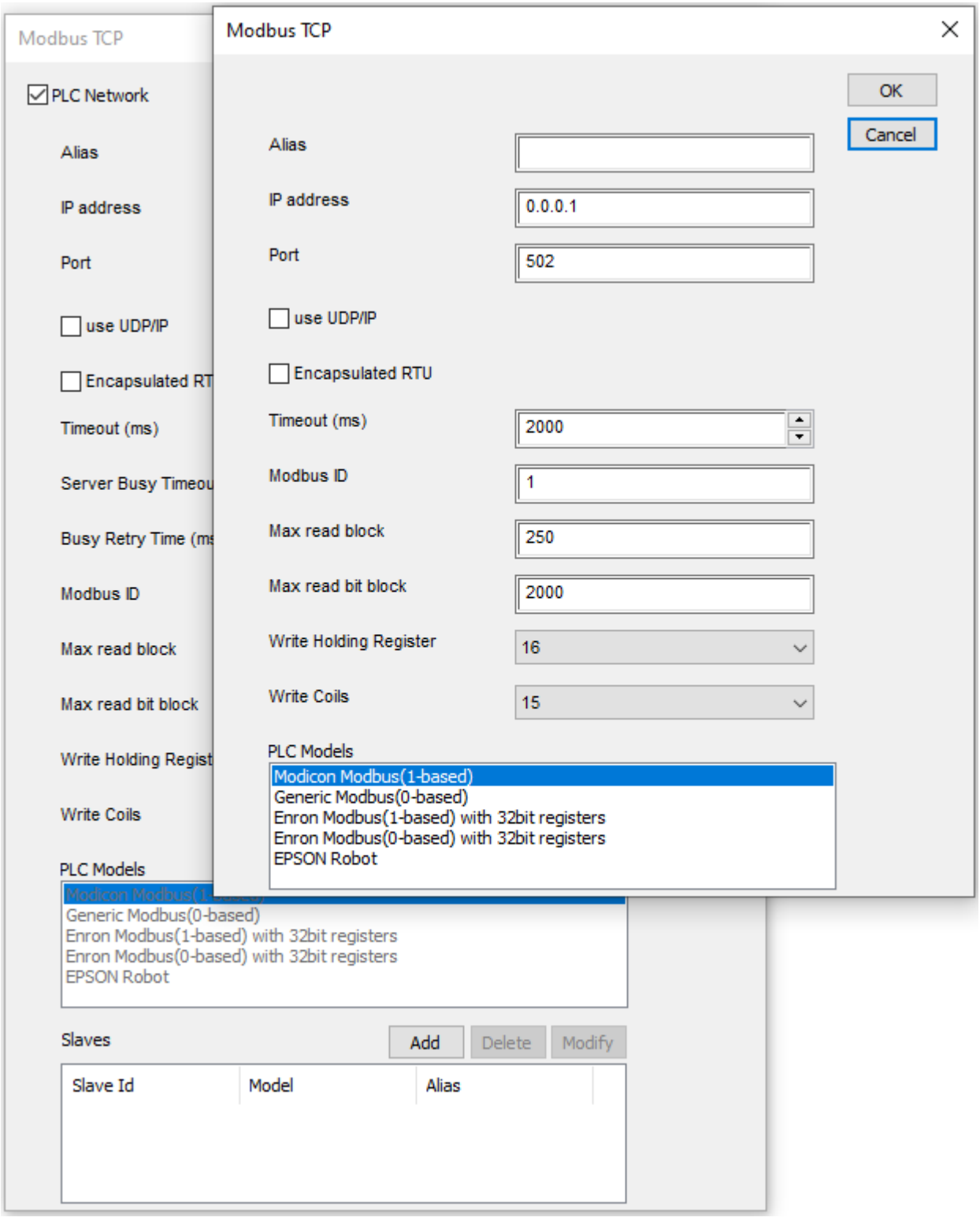
The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Address of the controller.
Port	Port number used by the Modbus TCP driver. The default value is **502** and can be changed when the communication goes through routers or Internet gateways where the default port number is already in use.
use UDP/IP	If selected, the protocol will use connectionless UDP datagrams.
Encapsulate	If selected, the protocol will use serial RTU protocol over Ethernet instead of Modbus TCP

Element	Description
d RTU	protocol, independently from TCP or UDP usage.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
Modbus ID	Usually used when communicating over Ethernet-to-serial gateways and then interpreted as the Slave ID. This value is simply copied into the Unit Identifier field of the Modbus TCP communication frame. This must correspond to server configuration. In most cases, server answers to Modbus ID 1, so this parameter can be left 1.
Max read block	Maximum length in bytes of a data block request. It applies only to read access of Holding Registers.
Max read bit block	Maximum length in bits of a block request. It applies only to read access of Input Bits and Output Coils.
Write Holding Register	Modbus function for write operations to Holding Registers. Select between the function **06** (preset single register) and function **16** (preset multiple registers). If **06** is selected, the protocol will always use function **06** for writing to the controller, even when writing to multiple consecutive registers. If **16** is selected, the protocol will always use function **16** to write to the controller, even for a single register write request and the **Max read block size** parameter of the query is set to **2**. The use of function **16** may result in higher communication performance. If **Auto** is selected, the protocol will use both function **06** or function **16** depending on number of registries to be written.
Write Coils	Modbus function for write operations to Output Coils. Select between the function **05** (write single coil) and function **15** (write multiple coils). If Modbus function **05** is selected, the protocol will always use function **05** for writing to the controller, even when writing to multiple consecutive coils. If Modbus function **15** is selected, the protocol will always use function **15** to write to the controller, even for a single coil write request. The use of function **15** may result in higher communication performance.

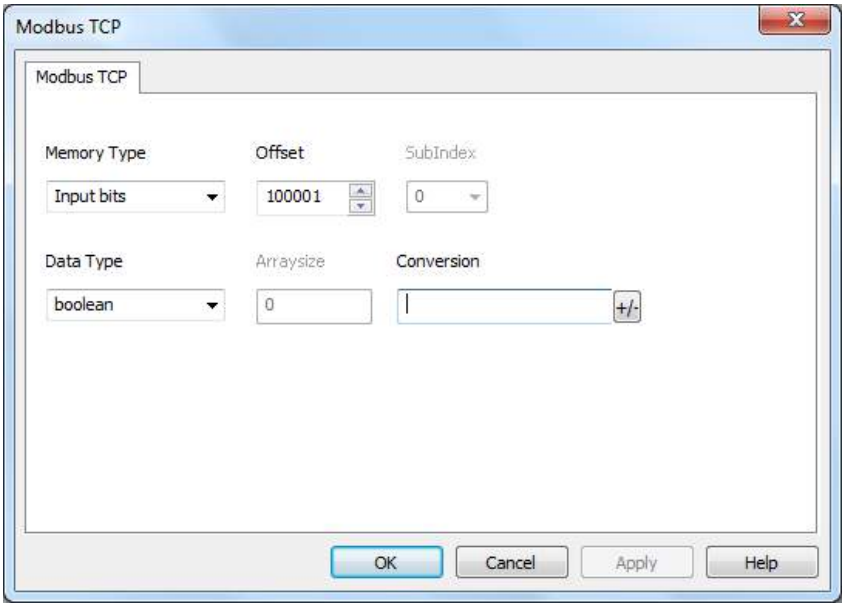
Element	Description
PLC Models	Allows to select between different PLC models: • **Modicon Modbus (1-based)**: Modbus implementation where all resources starts with offset 1. • **Generic Modbus (0-based)**: Modbus implementation where all resources starts with offset 0. • **Enron Modbus (1-based)**: Extends Modicon Mobdus implementation with 32 bit registers memory area. • **Enron Modbus (0-base)**: Extends Generic Modbus implementation with 32 bit registers memory area.  Note: The address range used in the Modbus frames is always between 0 and 65535 for the Holding Registers and between 0 and 65535 for Coils.
PLC Network	IP address for all controllers in multiple connections. **PLC Network** must be selected to enable multiple connections.

Element	Description
	

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

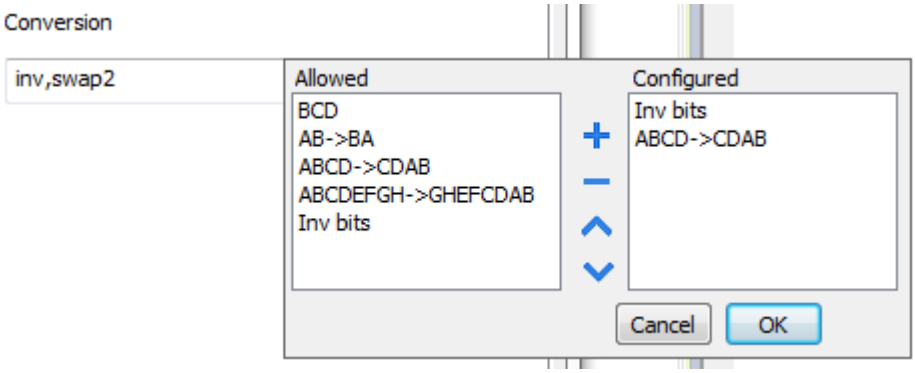
1. To add a tag, click **+**: a new line is added.
2. Select **Modbus TCP** from the **Driver** list: tag definition dialog is displayed.



Element	Description
Memory Type	Modbus resource where tag is located. **Memory Type** / **Modbus Resource** **Coil Status** – Coils **Input Status** – Discrete Input **Input Registers** – Input Registers **Holding registers** – Holding Registers **32 bit Registers** – 32 bit registers memory area. Available only for **Enron Modbus** PLC Models **Node Override IP**, **Node Override Port**, **Node Override ID**, **Modicon Mode** – protocol parameter (see **Special Data Types** for mode details)
Offset	Offset address where tag is located. Offset addresses are six digits composed by one digit data type prefix + five digits resource address.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Memory Type</th><th>Studio
Offset range</th><th>Modicon
Offset range</th><th>Generic
Modbus
Offset range</th></tr>
<tr><td>Coil Status</td><td>0 – 65535</td><td rowspan="5">1 – 65536</td><td rowspan="5">0 – 65535</td></tr>
<tr><td>Input Status</td><td>100000 – 165535</td></tr>
<tr><td>Input Registers</td><td>300000 – 365535</td></tr>
<tr><td>Holding Registers</td><td>400000 – 465535</td></tr>
<tr><td>32 bit Registers</td><td>0 – 65535</td></tr>
</table>
</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte []”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>
• In case of array tag, this property represents the number of array elements.
• In case of string tag, this property represents the maximum number of bytes available in the
</td></tr>
</table>

Element	Description
	string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format)

Element	Description

Value	Description
	855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

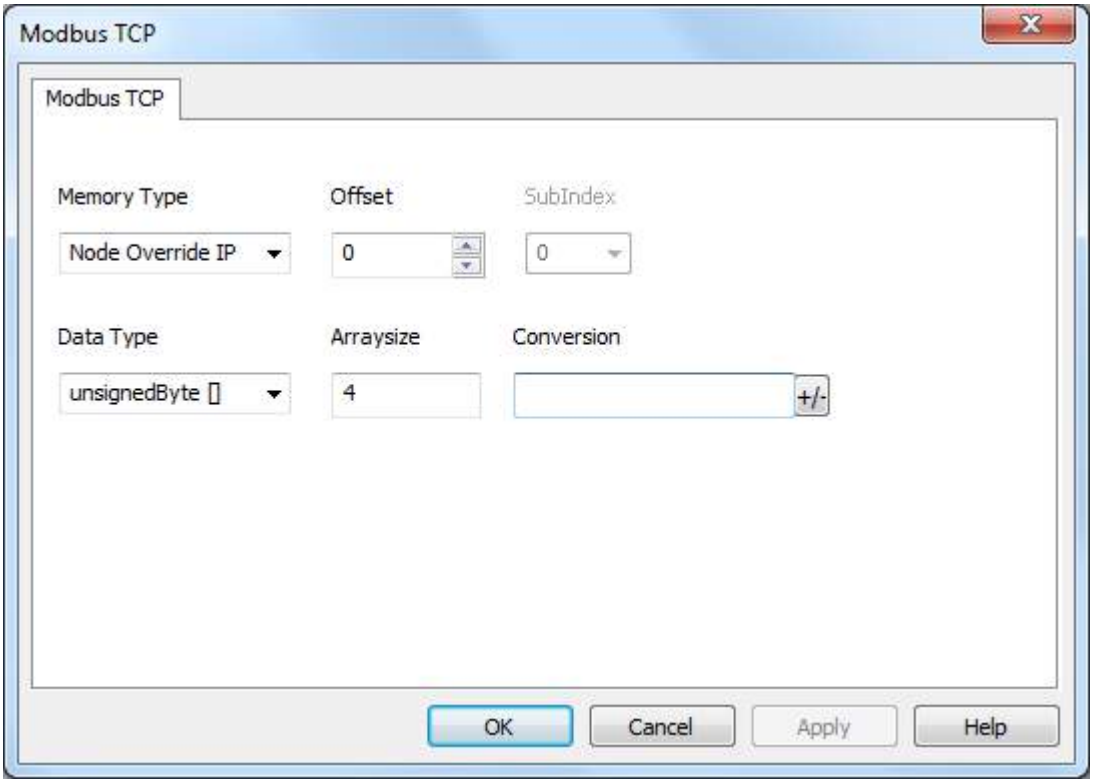
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.



Node Override Port

The protocol provides the special data type Node Override Port which allows you to change the network Port of the target controller at runtime.

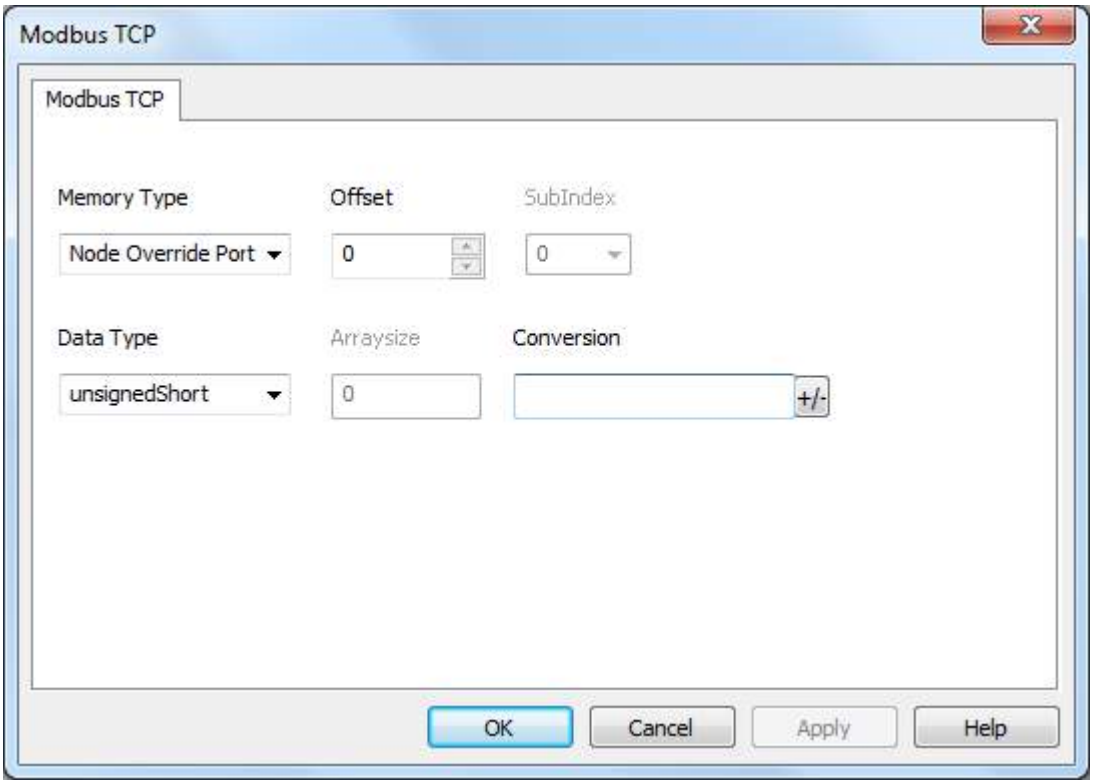
This memory type is unsigned short.

Node Override Port is initialized with the value of the controller Port specified in the project at programming time.

Node Override Port	Modbus operation
0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0	It is interpreted as the value of the new port and is replaced for runtime operation.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override Port variable.

Note: Node Override Port values assigned at runtime are retained through power cycles.



Node Override ID

The protocol provides the special data type Node Override ID which allows you to change the node ID of the slave at runtime. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Override ID	Modbus operation
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.

Note: Node Override ID value assigned at runtime is retained through power cycles.

Modicon Mode

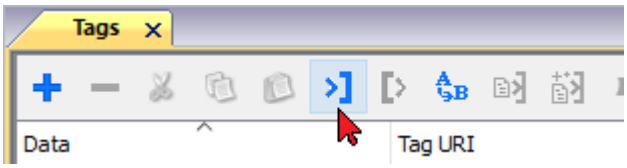
The protocol provide a special data type that can be used to override the Modicon Mode parameter at runtime.

Modicon Mode	Description
0	Generic Modbus (0-based). Register indexes start from 0.
1	Modicon Modbus (1-based). Register indexes start from 1.

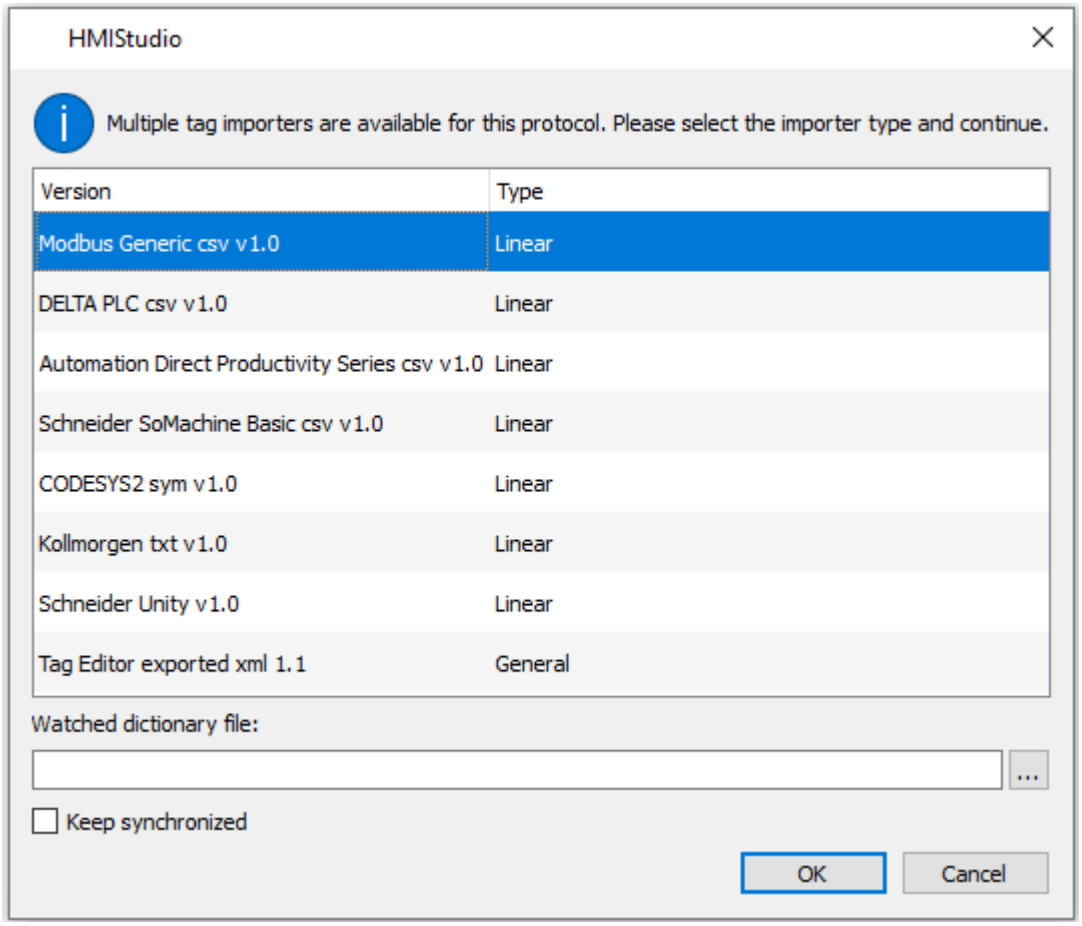
Note: Modicon Mode parameter value assigned at runtime is retained through power cycles.

Tag Import

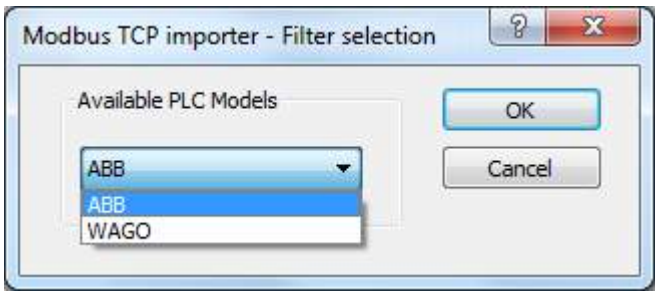
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



The following dialog shows which importer type can be selected.

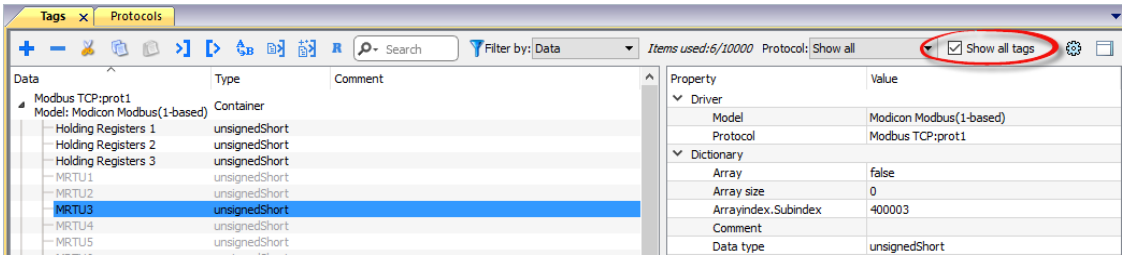


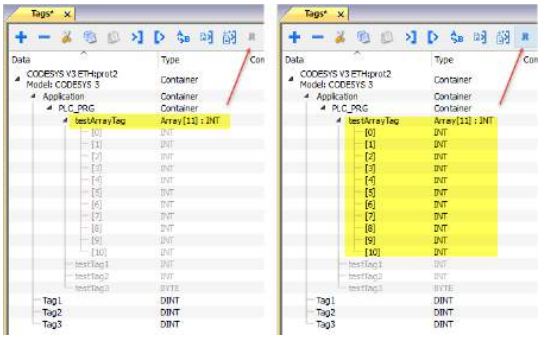
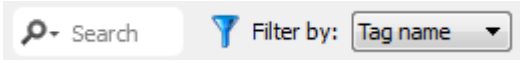
Type	Description
Modbus Generic csv v1.0 **Linear**	Requires a **.csv** file. All variables will be displayed at the same level.
DELTA PLC csv v1.0	Requires a **.csv** file. All variables will be displayed at the same level.
CODESYS2 sym v1.0 **Linear**	Requires a **.sym** file. All variables will be displayed at the same level. After selecting the **.sym** file, the following dialog will appear for PLC model selection.

Type	Description
	
Kollmorgen txt v1.0 Linear	Requires a **.txt** file. All variables will be displayed at the same level.
Schneider Unity v1.0 Linear	Requires a **.uny** file. The file containing symbols must be exported in **.txt** format and later renamed as **.uny**. The importer considers only variables located at fixed address and disregards arrays of strings. All other arrays, except for boolean type, are expanded.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Modbus Generic csv file structure

This protocol supports the import of tag information when provided in **.csv** format according to the following format:

```
NodeID, TagName, MemoryType, Address, DataFormat,...,[Comment]
```

Note: Fields in brackets are optional as well as fields between Data Format and Comment.

Field	Description
NodeID	Node the tag belongs to
TagName	Tag description
MemoryType	• OUTP • INP • IREG • HREG
Address	Offset compatible with Modbus notation

Field	Description
DataFormat	Data type in internal notation. See "Programming concepts" section in the main manual.
Comment	Optional additional description.

Tag file example

Example of .csv line:

```
2,Holding Register 1, HREG, 400001, unsignedShort,
```

Note: This line has no comment. When the Comment is missing, the comma as a terminator character is mandatory.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
No response	No reply within the specified timeout.	Check if the controller is connected and properly configured to get network access.
Incorrect node address in response	The device received a response with an invalid node address from the controller .	-
The received message too short	The device received a response with an invalid format from the controller .	-
Incorrect writing data acknowledge	The controller did not accept a write request.	Check if project data is consistent with the controller resources.

Modbus TCP Server

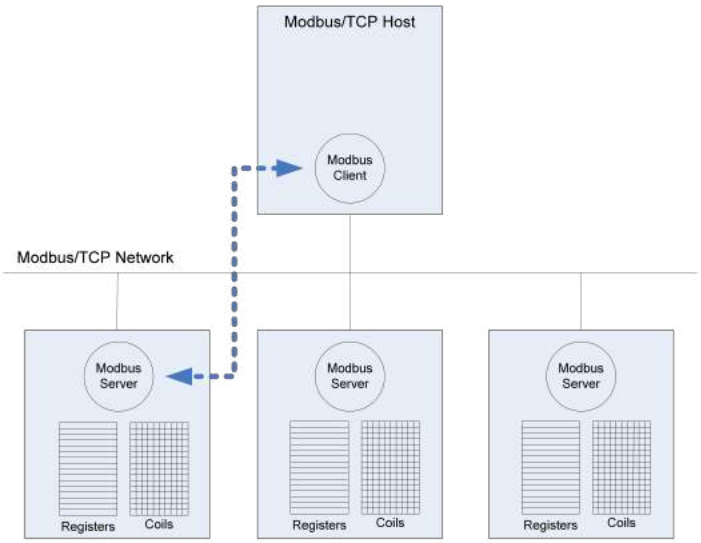
Modbus TCP Server communication driver allows connecting the HMI device as a server in a Modbus TCP network. It is possible for Modbus TCP clients to connect then to multiple HMI panels acting as servers. Standard Modbus TCP messages are used for information exchange.

This approach allows connecting HMI devices to SCADA systems through the universally supported Modbus TCP communication protocol.

Principle of operation

This communication driver implements a Modbus TCP Server unit in HMI device. A subset of the complete range of Modbus function codes is supported. The available function codes allow data transfer between clients on the TCP network and the server. The HMI device acts as a server in the network. It can exchange data with up to 32 clients. This means that up to 32 clients can be connected to the HMI device at the same time. If all the 32 available connections are in use, any further attempt to connect by a client will be refused by the system.

The following diagram shows the system architecture.



The device simulates the communication interface of a PLC: Coils and Registers data types are respectively boolean and 16 bit integers.

The device always access data in its internal memory. Data can be transferred to and from the Modbus Client only on the initiative of the client itself.

Implementation details

This Modbus TCP Server implementation supports only a subset of the Modbus standard function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the device Coil area.
02	Read Input Status	Reads multiple bits in the device Coil area.
03	Read Holding Registers	Read multiple device Registers.

Code	Function	Description
04	Read Input Registers	Read multiple device Registers.
05	Force Single Coil	Forces a single device Coil to either ON or OFF.
06	Preset Single Register	Presets a value in a device Register.
15	Force Multiple Coils	Forces multiple device Coils to either ON or OFF.
16	Preset Multiple Registers	Presets value in multiple device Registers.
23	Read Write Multiple Registers	Read & presets values in multiple device Registers

Note: For both PLC models the Read Coil Status and Read Input Status function codes both access the same Coil memory area in the HMI device memory. The Read Holding Registers and Read Input Registers function codes both access the same Register area in the HMI device memory.

Exception Codes

Code	Description
01	**Illegal Function**. the function code received in the query is not supported
02	**Illegal Data Address**. Data Address received in the query exceeds the predefined data range (see **Tag Editor Settings** for detailed ranges of all types).
03	**Illegal Data Value**. A sub function other than 00 is specified in Loopback Diagnostic Test (Code 08).

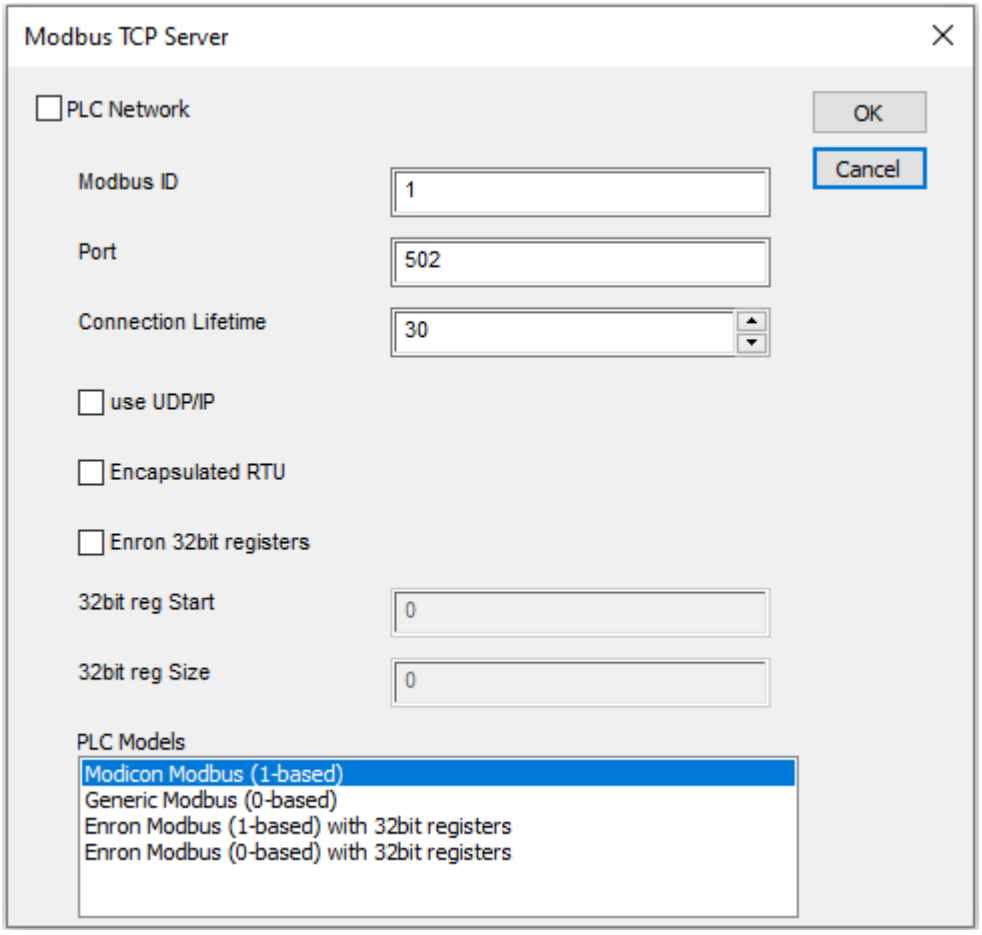
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



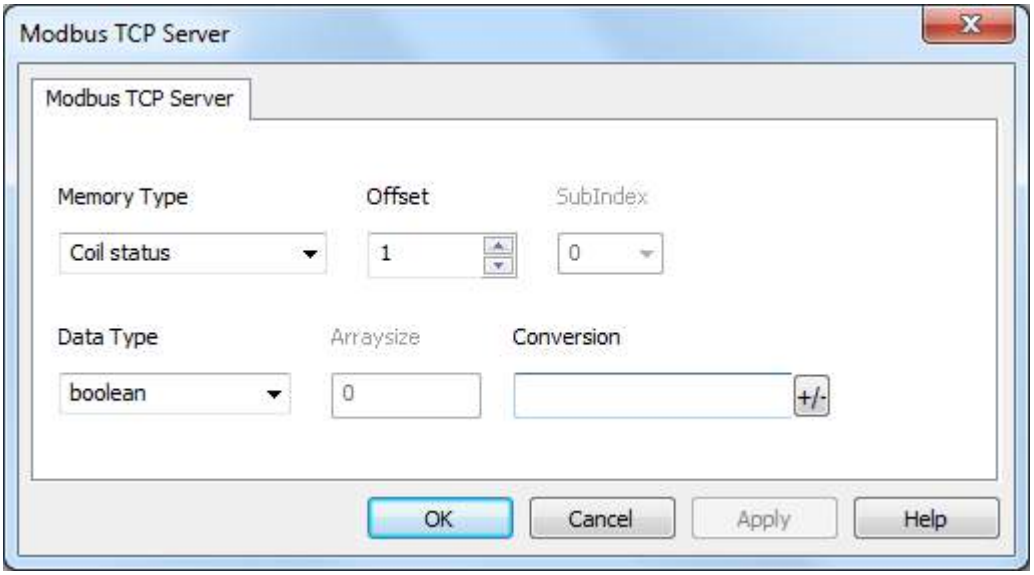
Element	Description
Modbus ID	Modbus node ID of the HMI device. Every Modbus server device in the network must have its own Modbus ID.
Port	Port number used by the Modbus TCP protocol. Default value is **502**. Set the value accordingly to the port number used by your Modbus TCP Network.
use UDP/IP	If selected, the protocol will use connectionless UDP datagrams.
Encapsulated RTU	If selected, the protocol will use serial RTU protocol over Ethernet instead of Modbus TCP protocol, independently from TCP or UDP usage.
Enron 32bit registers	If selected, allows to define the first register address and the number of registers for 32 bit registers memory area. Note: 32 bit registers are available only for **Enron Modbus** PLC Models.
32bit reg Start **32bit reg Size**	32 bit registries memory area definition. **Start** value represents the first register address. **Size** value represents the number of registries.

Element	Description
	Note: A request to one of the registries inside this area gives a 4 byte answer.
PLC Models	Allows to select between different PLC models: • **Modicon Modbus (1-based)**: Modbus implementation where all resources starts with offset 1. • **Generic Modbus (0-based)**: Modbus implementation where all resources starts with offset 0. • **Enron Modbus (1-based)**: Extends Modicon Mobdus implementation with 32 bit registers memory area. • **Enron Modbus (0-base)**: Extends Generic Modbus implementation with 32 bit registers memory area. Note: The address range used in the Modbus frames is always between 0 and 65535 for the Holding Registers and between 0 and 65535 for Coils.

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

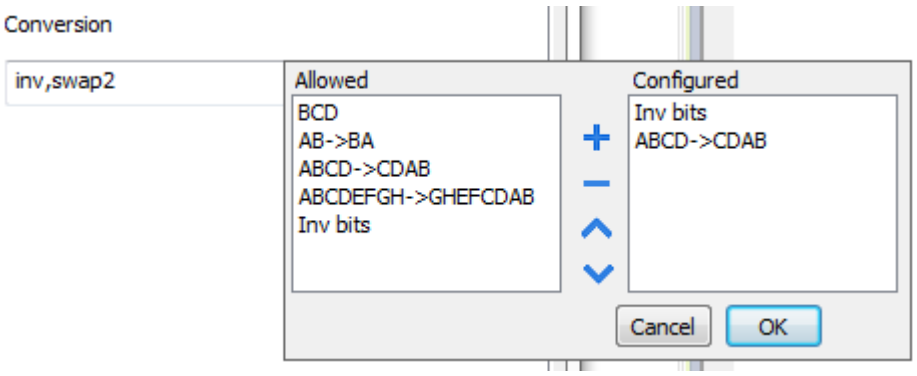
1. To add a tag, click **+**: a new line is added.
2. Select **Modbus TCP Server** from the protocol list: tag definition dialog is displayed.



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Modbus resource where tag is located.
<table>
<tr><th>Memory Type</th><th>Modbus Resource</th></tr>
<tr><td>Coil Status</td><td>Coils</td></tr>
<tr><td>Input Status</td><td>Discrete Input</td></tr>
<tr><td>Input Registers</td><td>Input Registers</td></tr>
<tr><td>Holding Registers</td><td>Holding Registers</td></tr>
<tr><td>32 bit Registers</td><td>32 bit registers memory area.
Available only for Enron Modbus PLC Models.</td></tr>
<tr><td>Modicon Mode</td><td>protocol parameter (see Special Data Types for mode details)</td></tr>
</table></td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.
Offset addresses are six digits composed by one digit data type prefix + five digits resource address.
<table>
<tr><th>Memory Type</th><th>Studio
Offset range</th><th>Modicon
Offset range</th><th>Generic Modbus
Offset range</th></tr>
<tr><td>Coil Status</td><td>0 – 65535</td><td rowspan="5">1 – 65536</td><td rowspan="5">0 – 65535</td></tr>
<tr><td>Input Status</td><td>100000 – 165535</td></tr>
<tr><td>Input Registers</td><td>300000 – 365535</td></tr>
<tr><td>Holding Registers</td><td>400000 – 465535</td></tr>
<tr><td>32 bit Registers</td><td>0 – 65535</td></tr>
</table></td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
<tr><td>Data type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>
Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001 → 1 10000011100 1010101000010100010110110110110010110110000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

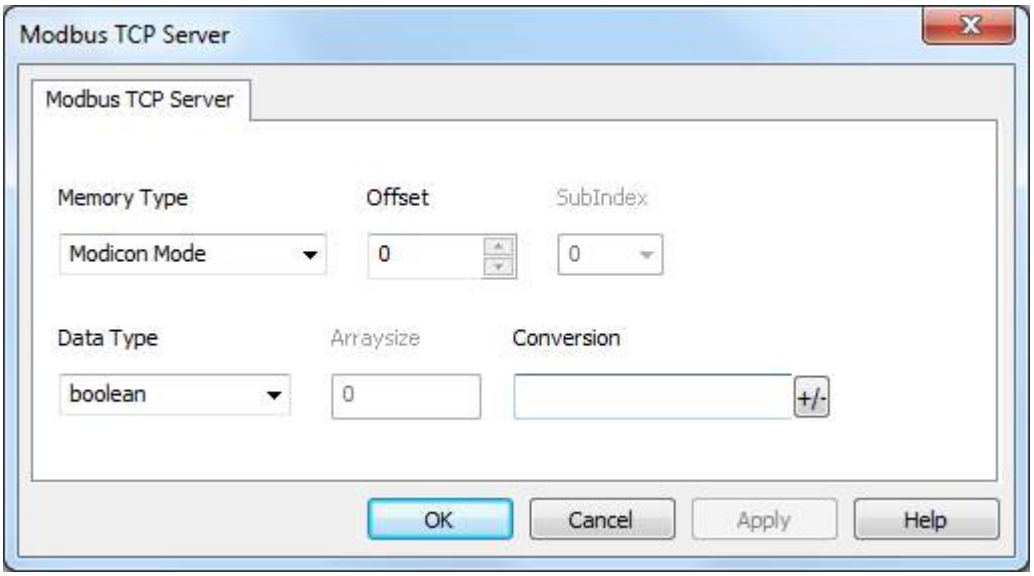
Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Modicon Mode

The protocol provide a special data type that can be used to override the Modicon Mode parameter at runtime.

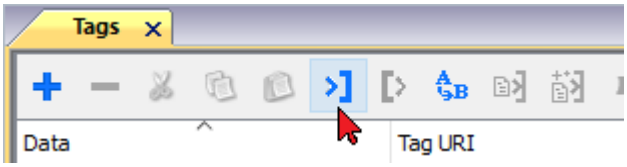
Modicon Mode	Description
0	Generic Modbus (0-based). Register indexes start from 0.
1	Modicon Modbus (1-based). Register indexes start from 1.

Note: Modicon Mode parameter value assigned at runtime is retained through power cycles.

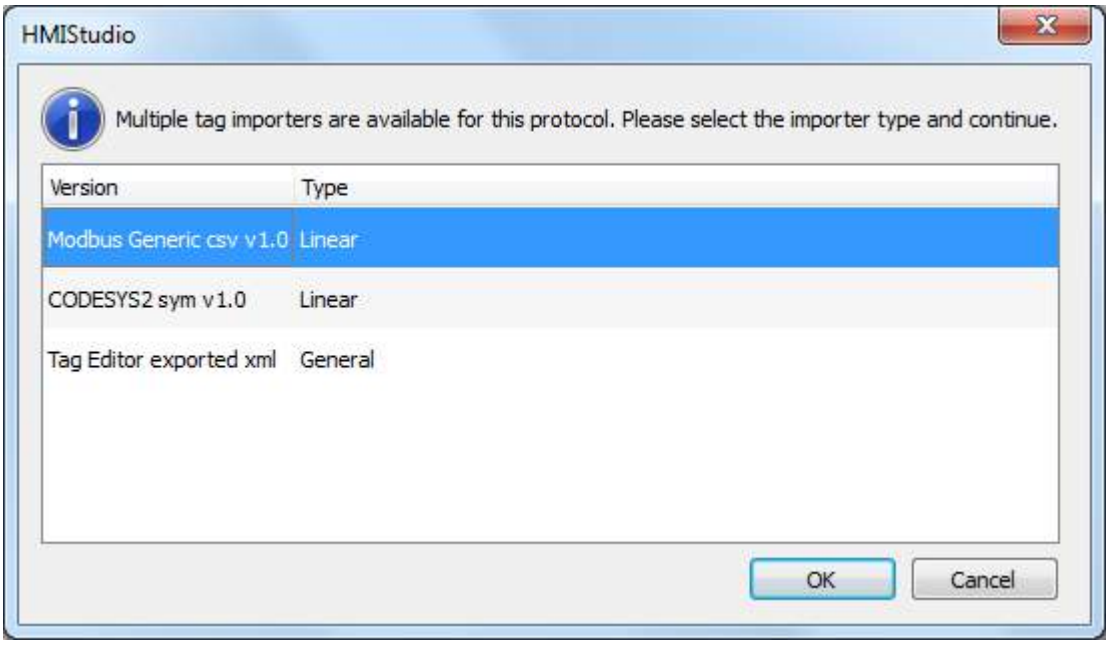


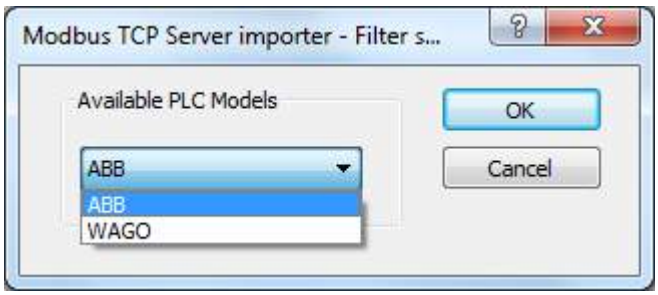
Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



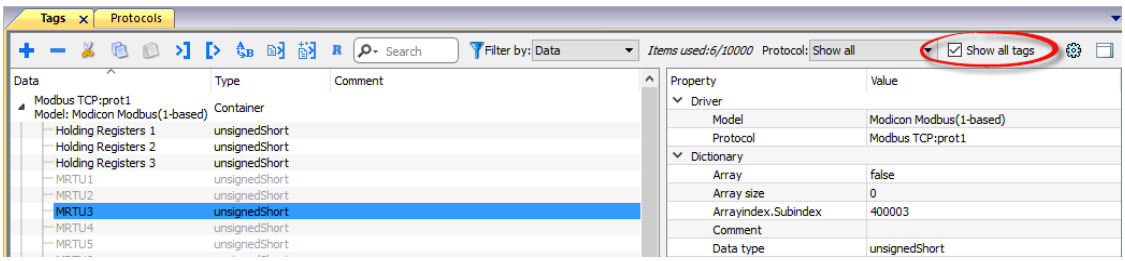
The following dialog shows which importer type can be selected.

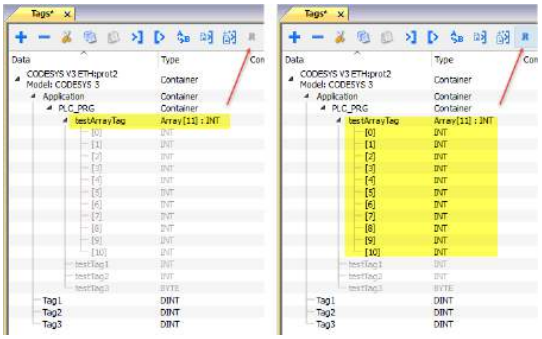
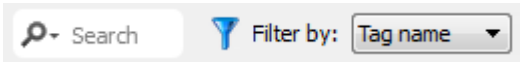


Importer	Description
Modbus Generic csv v1.0 **Linear**	Requires a **.csv** file. All variables will be displayed at the same level.
CODESYS2 sym v1.0 **Linear**	Requires a **.sym** file. All variables will be displayed at the same level. After selecting the **.sym** file, the following dialog will appear for PLC model selection. 
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Modbus Generic csv file structure

This protocol supports the import of tag information when provided in **.csv** format according to the following format:

```
NodeID, TagName, MemoryType, Address, DataFormat,...,[Comment]
```

Note: Fields in brackets are optional as well as fields between Data Format and Comment.

Field	Description
NodeID	Node the tag belongs to
TagName	Tag description
MemoryType	• OUTP • INP • IREG • HREG
Address	Offset compatible with Modbus notation
DataFormat	Data type in internal notation. See "Programming concepts" section in the main manual.
Comment	Optional additional description.

Tag file example

Example of .csv line:

```
2,Holding Register 1, HREG, 400001, unsignedShort,
```

Note: This line has no comment. When the Comment is missing, the comma as a terminator character is mandatory.

Communication status

The HMI device is a server station in the Modbus TCP network. The current implementation of the protocol doesn't report any communication error code apart from standard communication error codes related to the proper driver loading.

See "System Variables" section in the main manual.

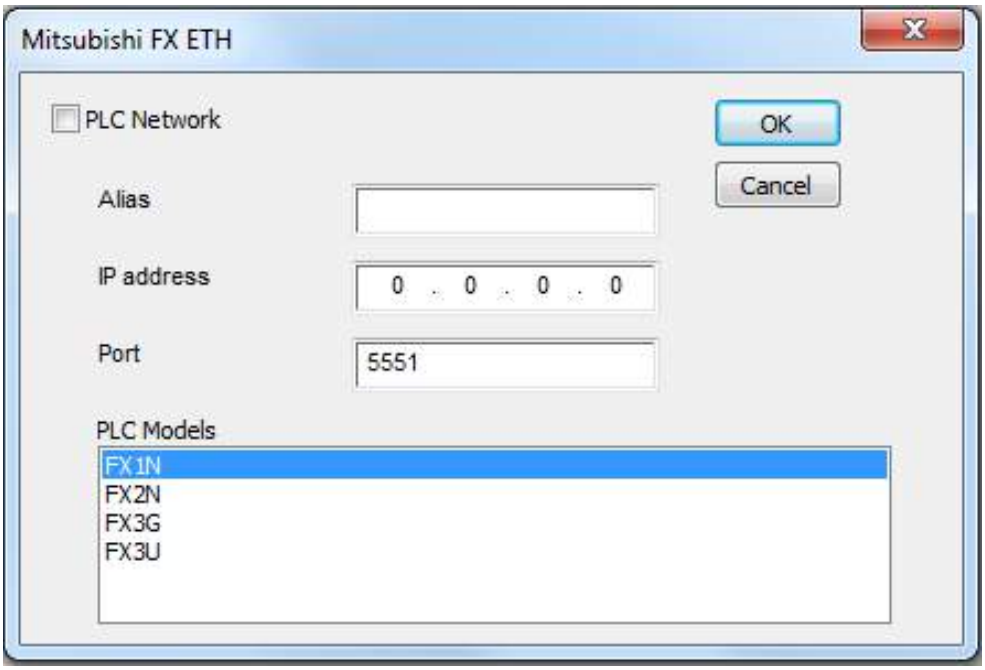
Mitsubishi FX ETH

Mitsubishi FX ETH implements the MELSEC-F (or MC) communication protocol that can be used with FX CPUs as described in the Mitsubishi document “FX3U-ENET USER’S MANUAL”, chapter 8 “Communication using MC protocol”.

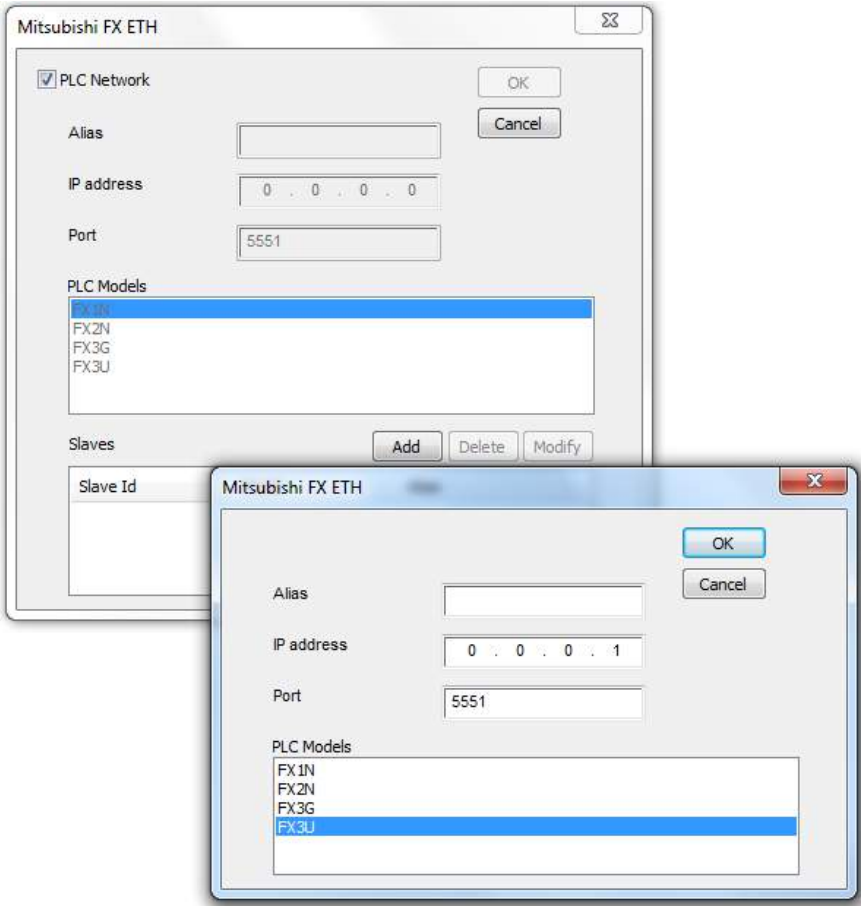
Note: Mitsubishi FX3U controller must be equipped with the appropriate Ethernet module: FX3U-ENET

Protocol Editor Settings

Add [+] a driver in the Protocol editor and select the protocol called “Mitsubishi FX ETH” from the list of available protocols.

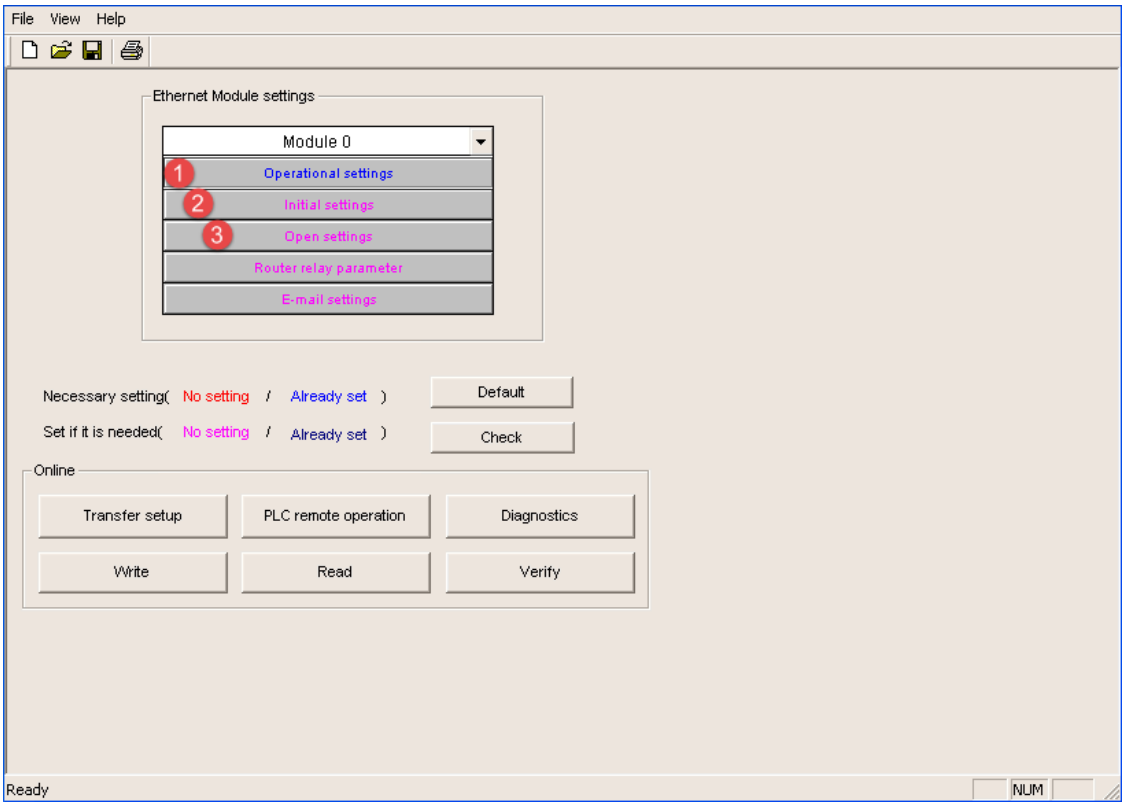


Element	Description
IP address	Ethernet IP address of the controller
Port	Specifies the port number (decimal) used in the communication with the PLC.

Element	Description
PLC Model	Defines the PLC model connected
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and enter IP Address for all controllers. 

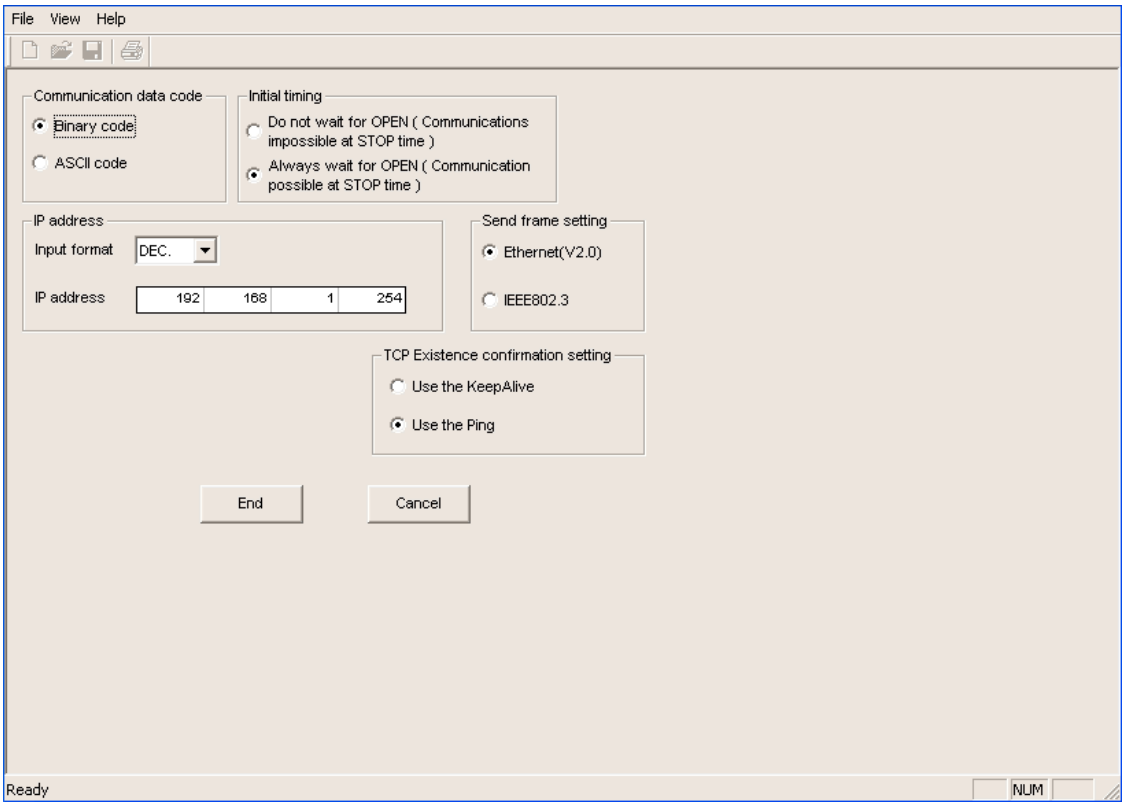
Controller Settings with GX Developer

The Mitsubishi FX system must be properly configured for Ethernet communication using the Mitsubishi FX Configurator. Click on “Operational settings” as shown at point (1) in the following figure:

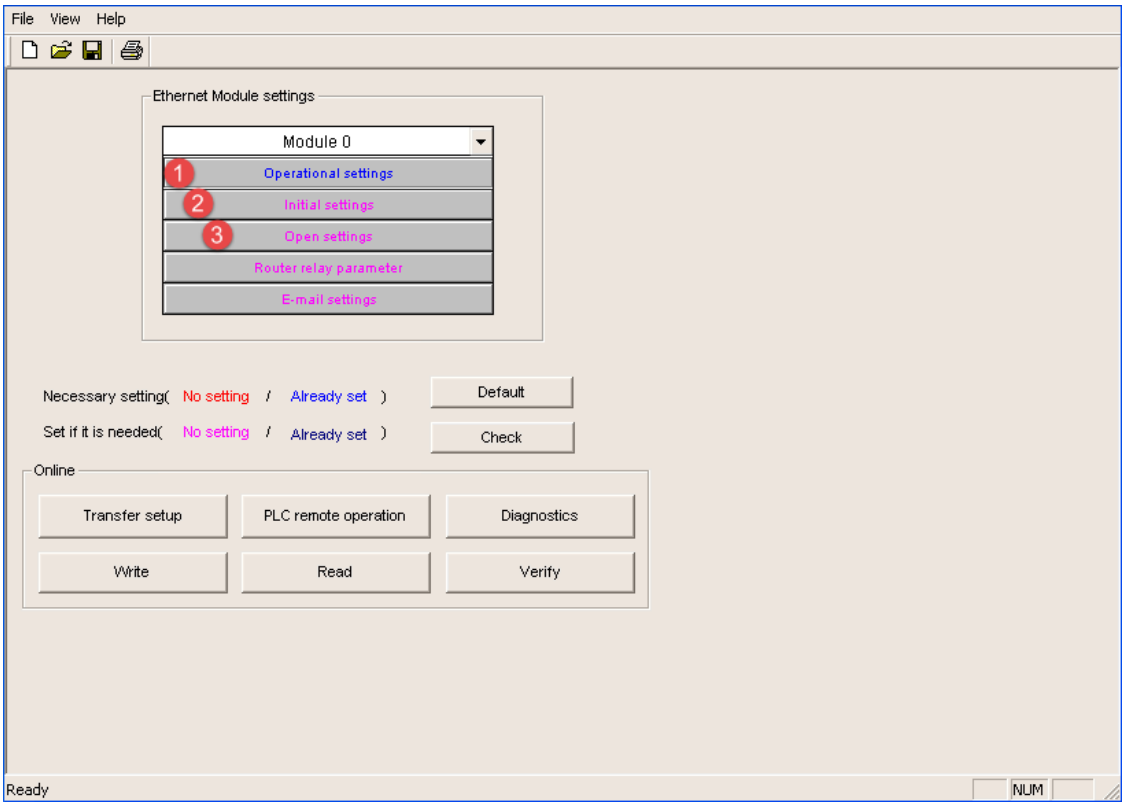


Into Operational Settings dialog, verify the “Communication data code” is set to “Binary code”,

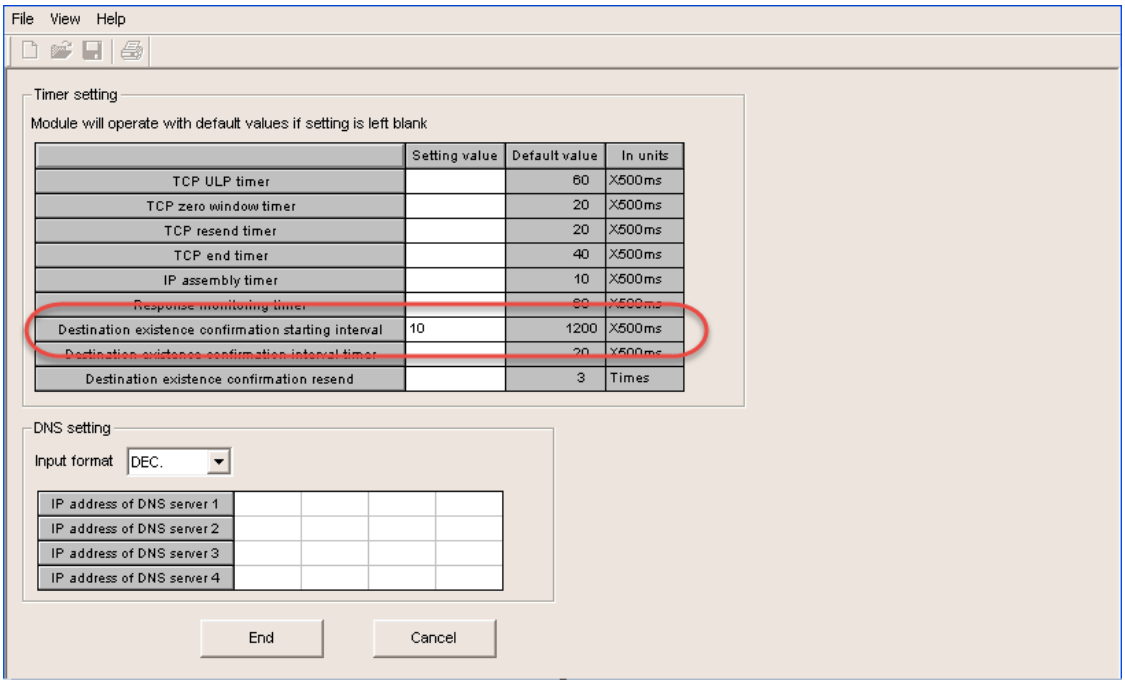
Then type-in the Controller IP Address and confirm with [End] button.



Click now on “Initial settings” as shown at point (2) of Figure below:

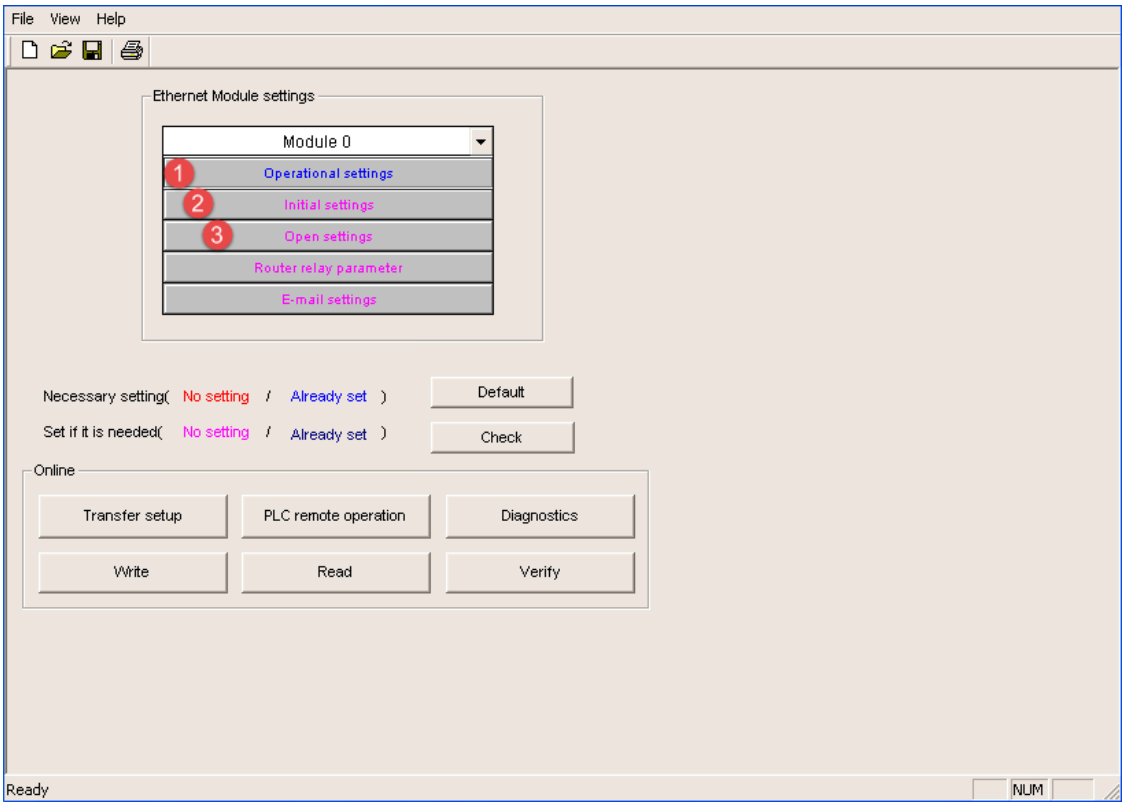


For proper communication between HMI and controller it is required to change “Destination existence confirmation starting interval” from the default value of 1200 to 10ms.



In case of communication error, this avoid controller keeps alive the connection for a too long time before to allow a new connection from the HMI.

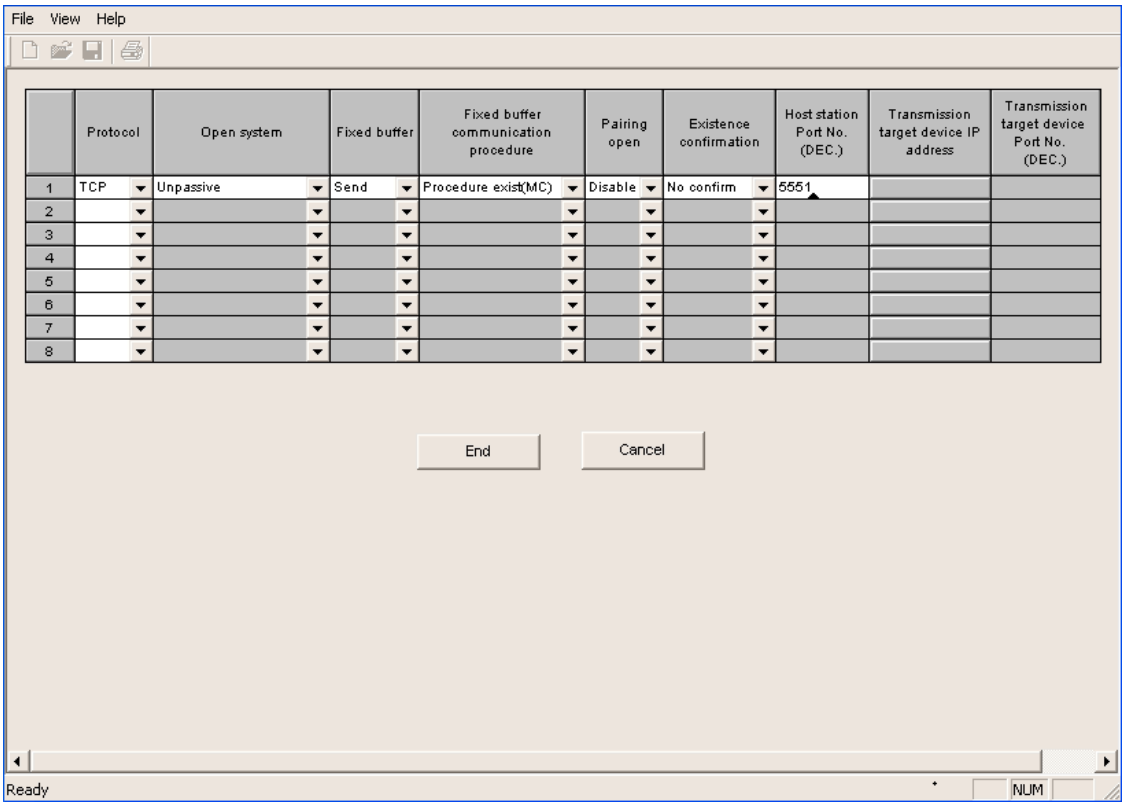
Click now on “Open settings” as shown at point (3) of Figure below



The next figure shows the “Ethernet open settings” configuration.

The detailed explanation of the meaning of each setting is available in Chapter 5.5 of the Mitsubishi “FX3U-ENET USER’S MANUAL”.

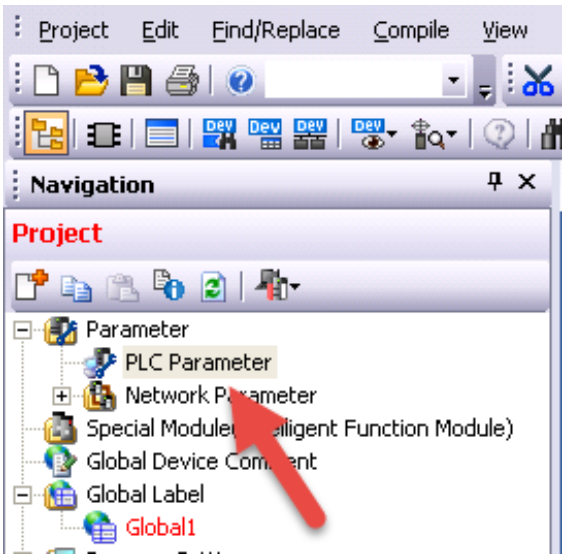
“Host station Port No.” defined here is the same must be used into Protocol Editor Settings chapter.



Note: the usage of more than one panel communicating with the same controller requires to define proper settings in the “Open settings” configuration dialog: one connection per each panel must be configured with proper properties

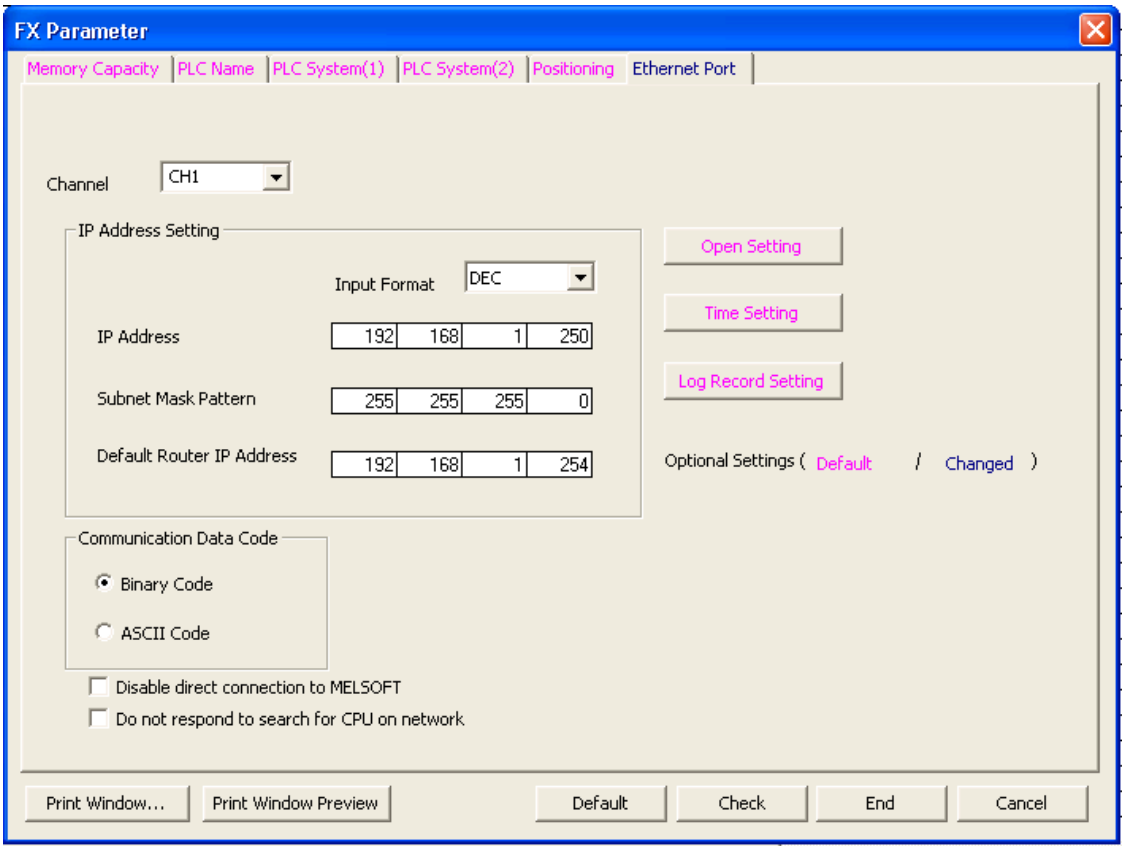
Controller Settings with GX Works2

The Mitsubishi FX system must be properly configured for Ethernet communication inside GX Works2 programming suite. FX Parameter dialog can be recalled with double-click on PLC Parameter:

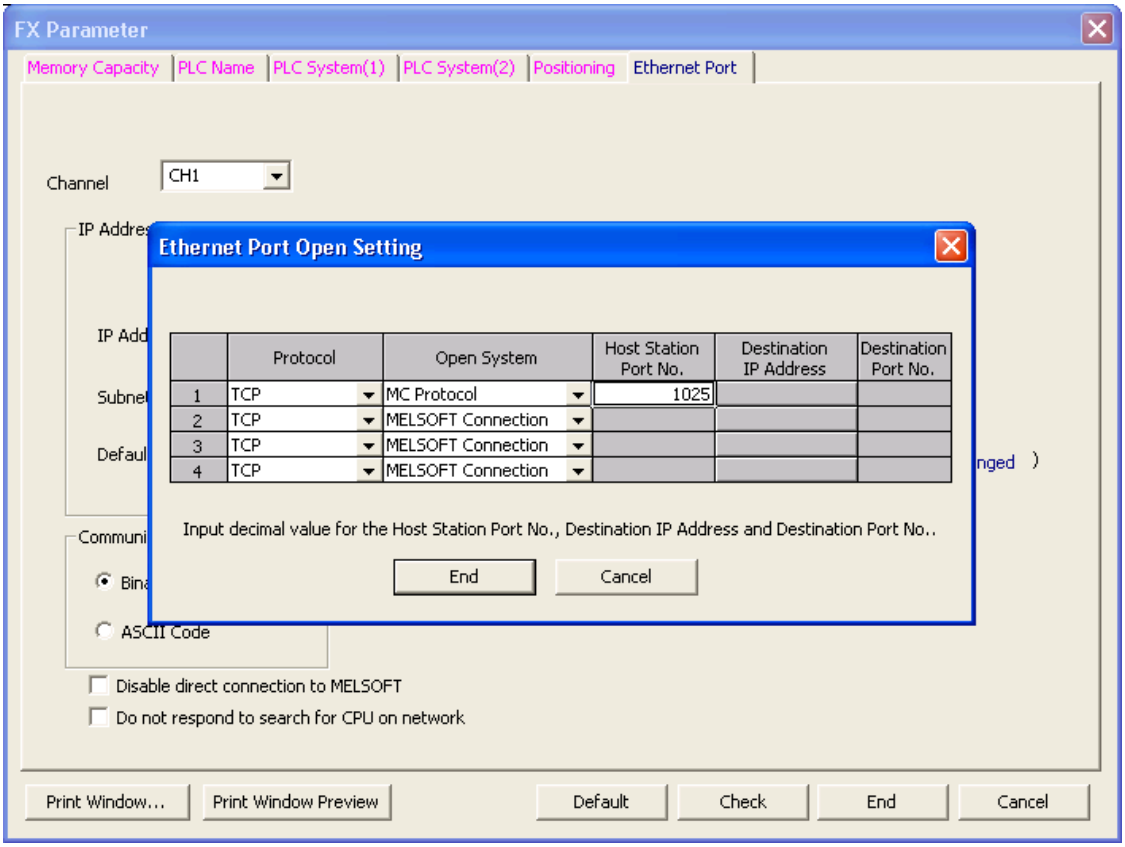


Then select “Ethernet Port” tab where is possible to configure IP Address.

Verify the “Communication data code” is set to “Binary code” as shown below:



Then click on “Open Settings” button to recall the “Ethernet Port Open Setting” dialog.



"Host station Port No." defined here is the same must be used into Protocol Editor Settings chapter.

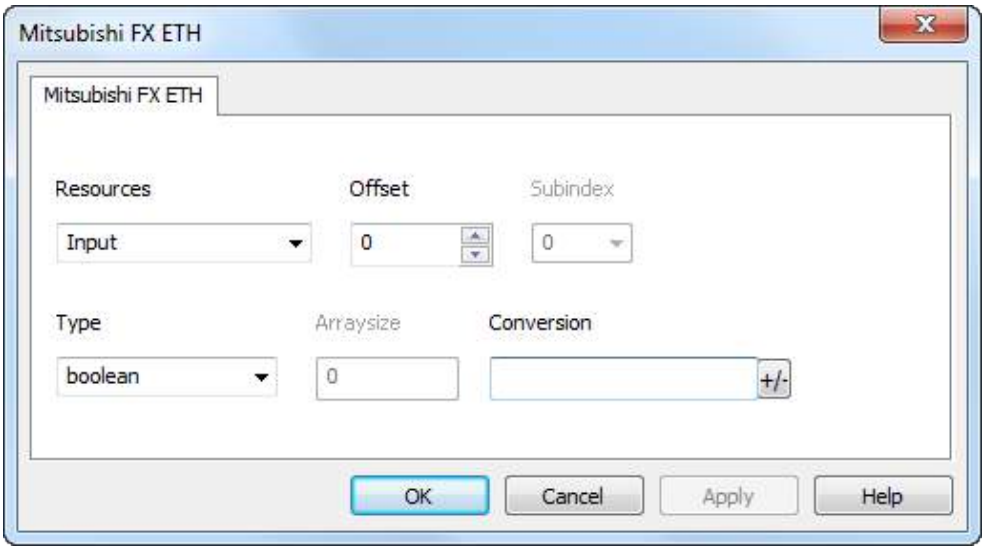
Note: For FX3GE Controller, the Open System must be set as "Data Monitor" and Port set to 1025.

Note: the usage of more than one panel communicating with the same controller requires to define proper settings in the "Open settings" configuration dialog: one connection per each panel must be configured with proper properties.

Tag Editor Settings

Into Tag editor select the protocol "Mitsubishi FX ETH" from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:



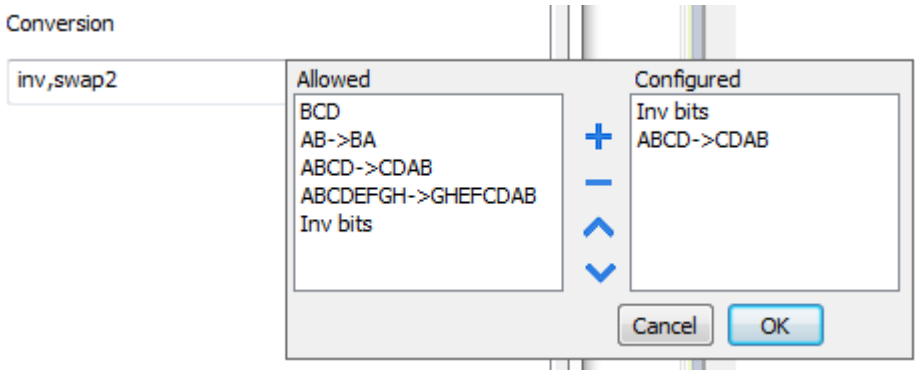
Element	Description
Resources	Area of PLC where tag is located
Offset	Offset address where tag is located.
SubIndex	This allows resource offset selection within the register.
Type	

Data Type	Memory Space	Limits
boolean	1 bit data	0 ... 1
byte	8-bit data	-128 ... 127
short	16-bit data	-32768 ... 32767
int	32-bit data	-2.1e9 ... 2.1e9
unsignedByte	8-bit data	0 ... 255
unsignedShort	16-bit data	0 ... 65535

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
<tr><td>string</td><td colspan="2">Refer to “String data type chapter”</td></tr>
</table>

Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Inv bits</td><td>inv: Invert all the bits of the tag.

Example:
1001 → 0110 (in binary format)
9 → 6 (in decimal format)</td></tr>
<tr><td>Negate</td><td>neg: Set the opposite of tag value.

Example:</td></tr>
</table>
</td></tr>
</table>

Element	Description

Value	Description
	25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 01 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

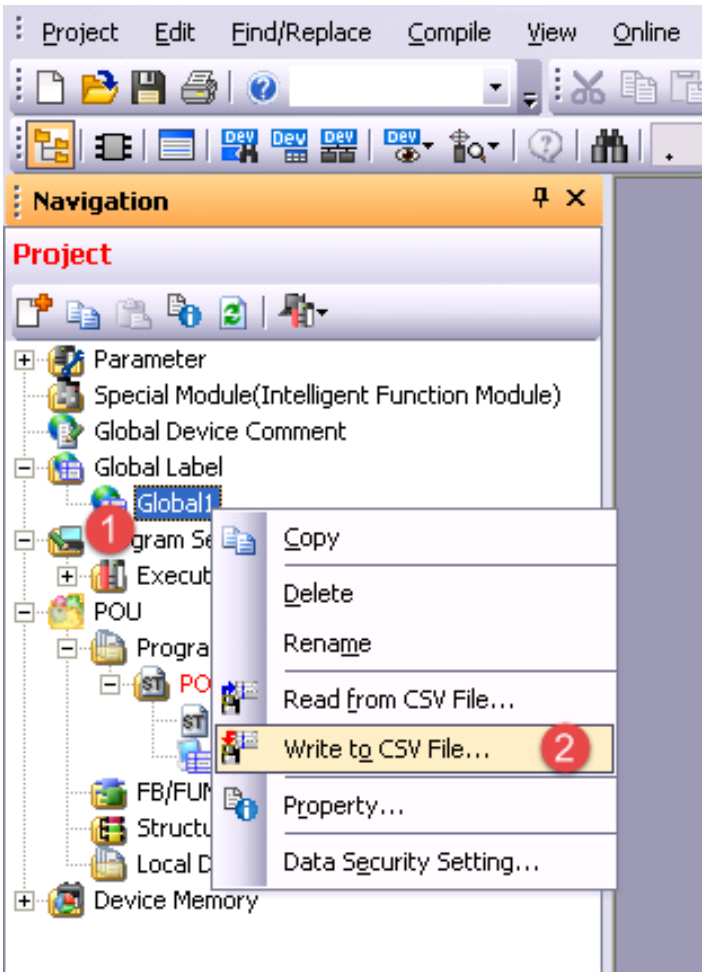
Tag Import

Exporting Tags from PLC

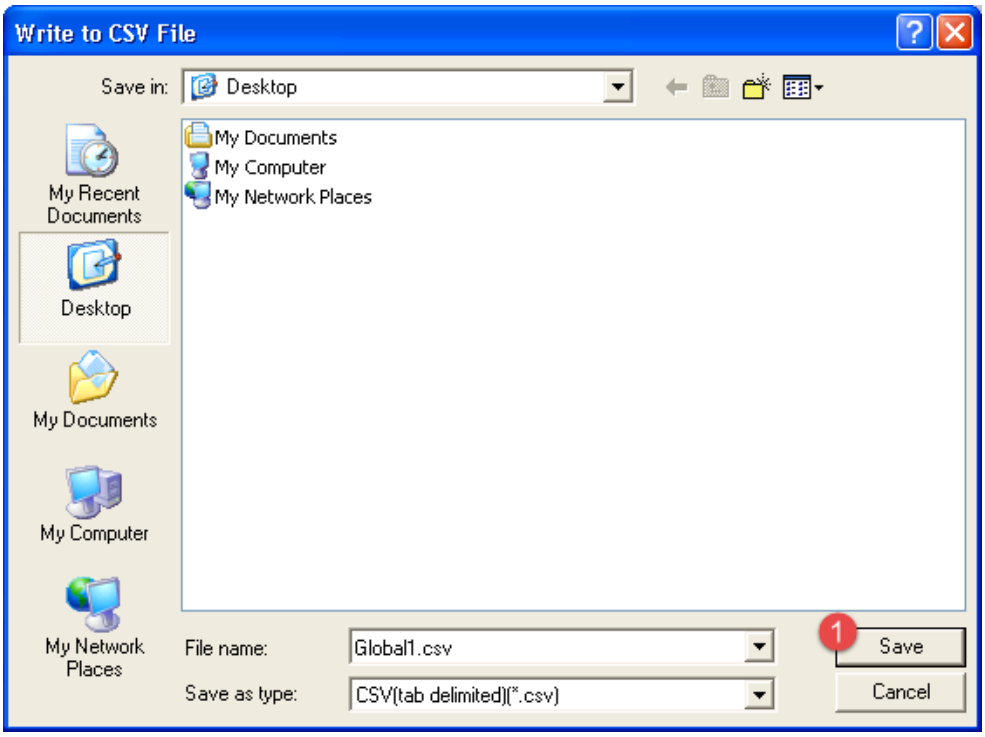
The Mitsubishi FX Ethernet tag import accepts symbol files with extension “csv” created by the Mitsubishi GX Works2 (Not from GX Developer).

The “.csv” file can be exported from the Project tree, as shown in the following figure.

1. Right-click on the Global variable list that need to be exported,
2. Select “Write to CSV File…”

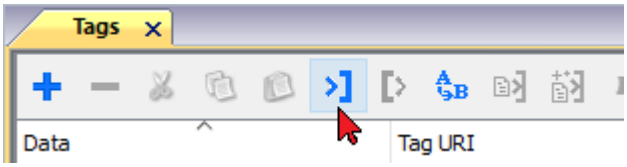


Into following dialog select the file name and location:

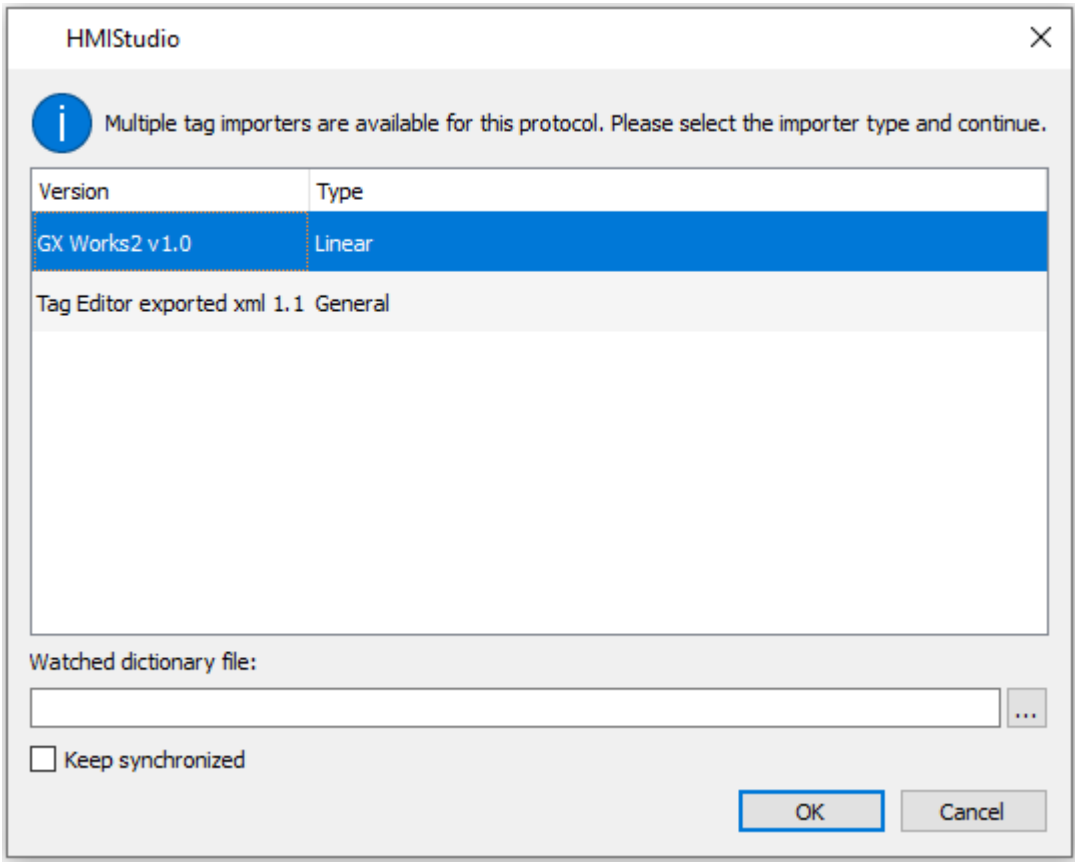


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



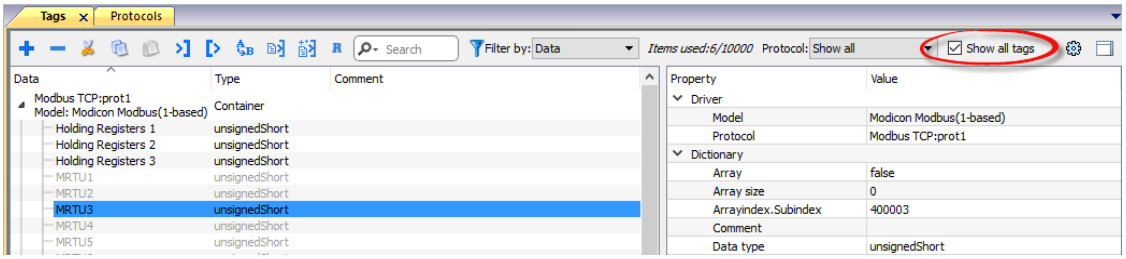
The following dialog shows which importer type can be selected.

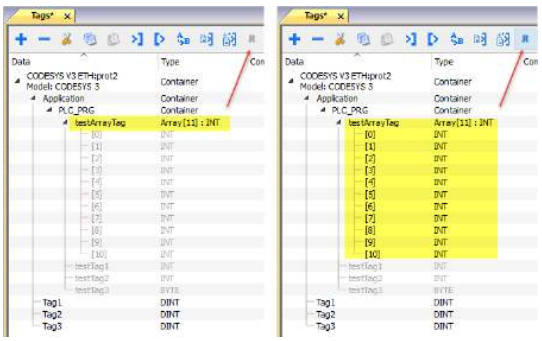
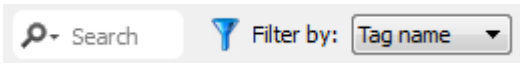


Importer	Description
GX Works2 v1.0 Linear	Requires a **.csv**file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
General Error	Error cannot be identified; should never be reported; contact technical support

Mitsubishi FX SER

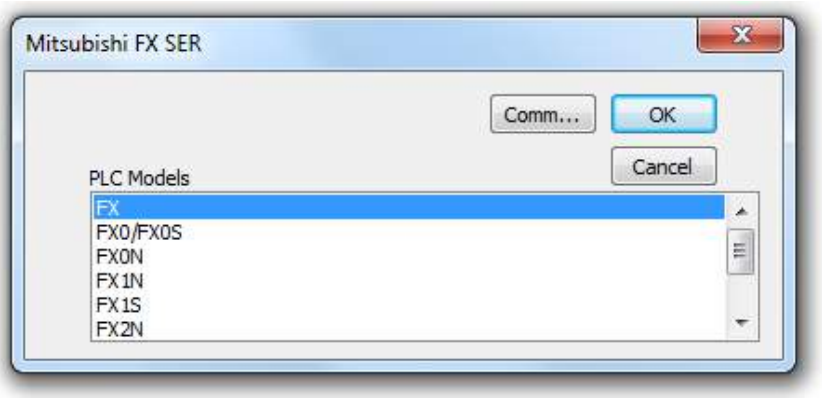
The HMI operator panels can be connected to Mitsubishi FX PLC as the network master using this communication driver.

The protocol has been designed to connect to the programming port of the PLC.

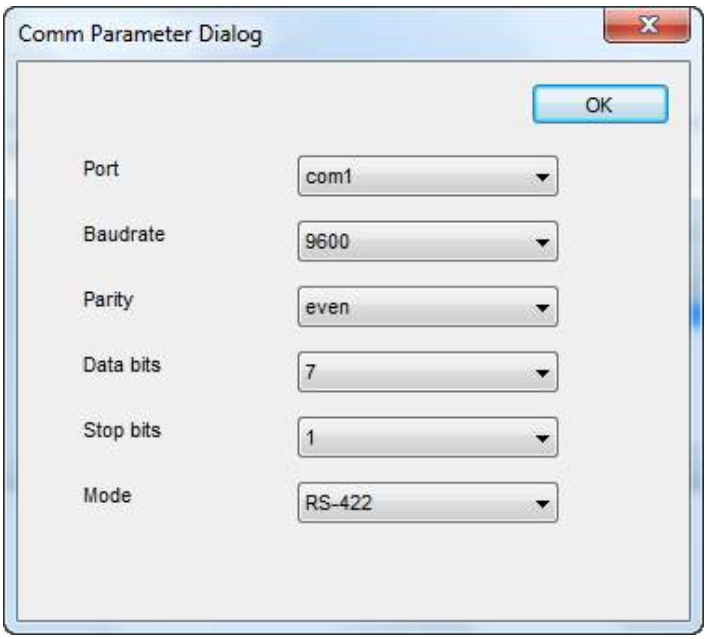
Please note that changes in the communication protocol specifications or PLC hardware may have occurred since this documentation was created. Some changes may eventually affect the functionality of this communication driver. Always test and verify the functionality of your application. To fully support changes in PLC hardware and communication protocols, communication drivers are continuously updated. Always ensure that the latest version of communication driver is used in your application.

Protocol Editor Settings

Add [+] a driver in the Protocol editor and select the protocol called “Mitsubishi FX SER” from the list of available protocols.



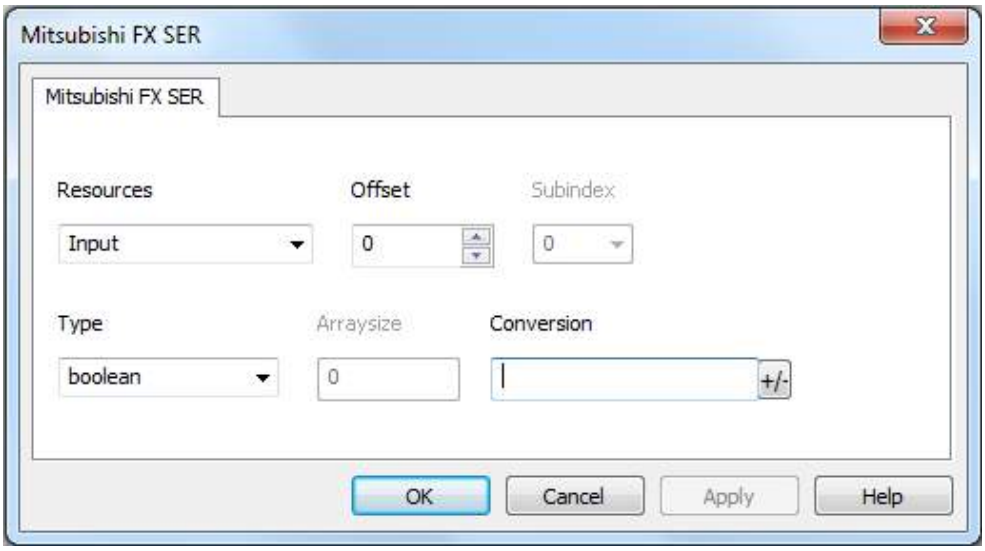
Element	Description
PLC Models	The list allows selecting the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
Comm…	Gives access to the serial port configuration parameters as shown in the figure below.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td> 
</td></tr>
<tr><td>Port</td><td>Serial port selection:
<table>
<tr><th>Port</th><th>Series 400</th><th>Series 500/600</th></tr>
<tr><td>com1</td><td>PLC Port</td><td>Onboard Serial Port</td></tr>
<tr><td>com2</td><td>PC/Printer Port</td><td>Optional Module on slot #1 or #2</td></tr>
<tr><td>com3</td><td>Not available</td><td>Optional Module on slot #3 or #4</td></tr>
</table></td></tr>
<tr><td>Baud rate, Parity, Data bits, Stop bits</td><td>Communication parameters for serial communication</td></tr>
<tr><td>Mode</td><td>Serial port mode; available options:
RS-232,
RS-485 (2 wires)
RS-422 (4 wires)</td></tr>
</table>

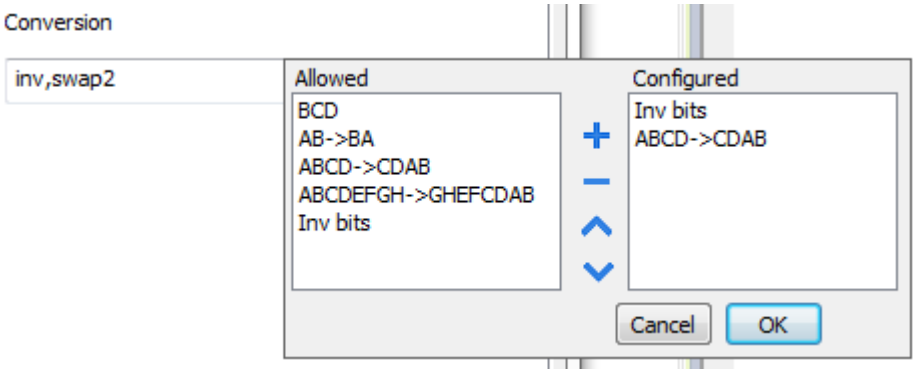
Tag Editor Settings

Into Tag editor select the protocol “Mitsubishi FX SER” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:



Element	Description
Resources	Area of PLC where tag is located
Offset	Offset address where tag is located.
SubIndex	This allows resource offset selection within the register.
Type	<table><tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr><tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr><tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr><tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr><tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr><tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr><tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr><tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr><tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr><tr><td>string</td><td colspan="2">Refer to “String data type chapter”</td></tr></table> Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag.

Element	Description
	Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)

Element	Description

Value	Description
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001 → 1 10000011100 1010101000010100010110110110110010110110000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

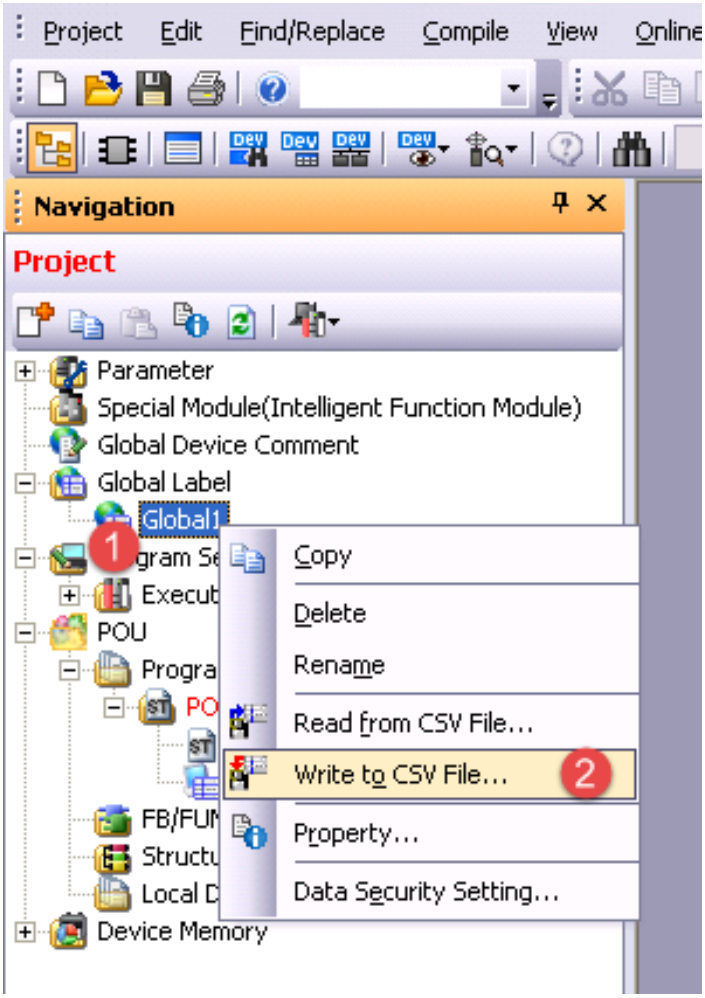
Tag Import

Exporting Tags from PLC

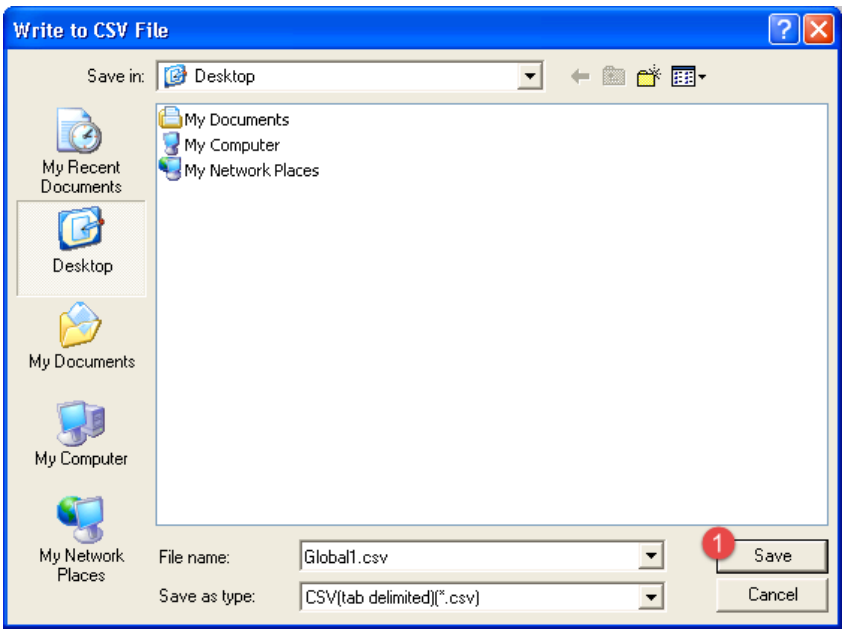
The Mitsubishi FX Serial tag import accepts symbol files with extension “csv” created by the Mitsubishi GX Works2 (Not from GX Developer).

The “.csv” file can be exported from the Project tree, as shown in the following figure.

1. Right-click on the Global variable list that need to be exported,
2. Select “Write to CSV File…”

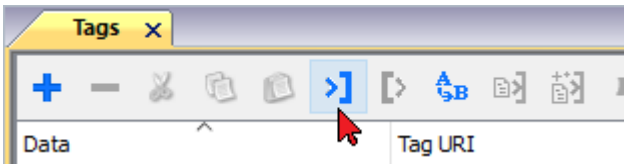


Into following dialog select the file name and location:

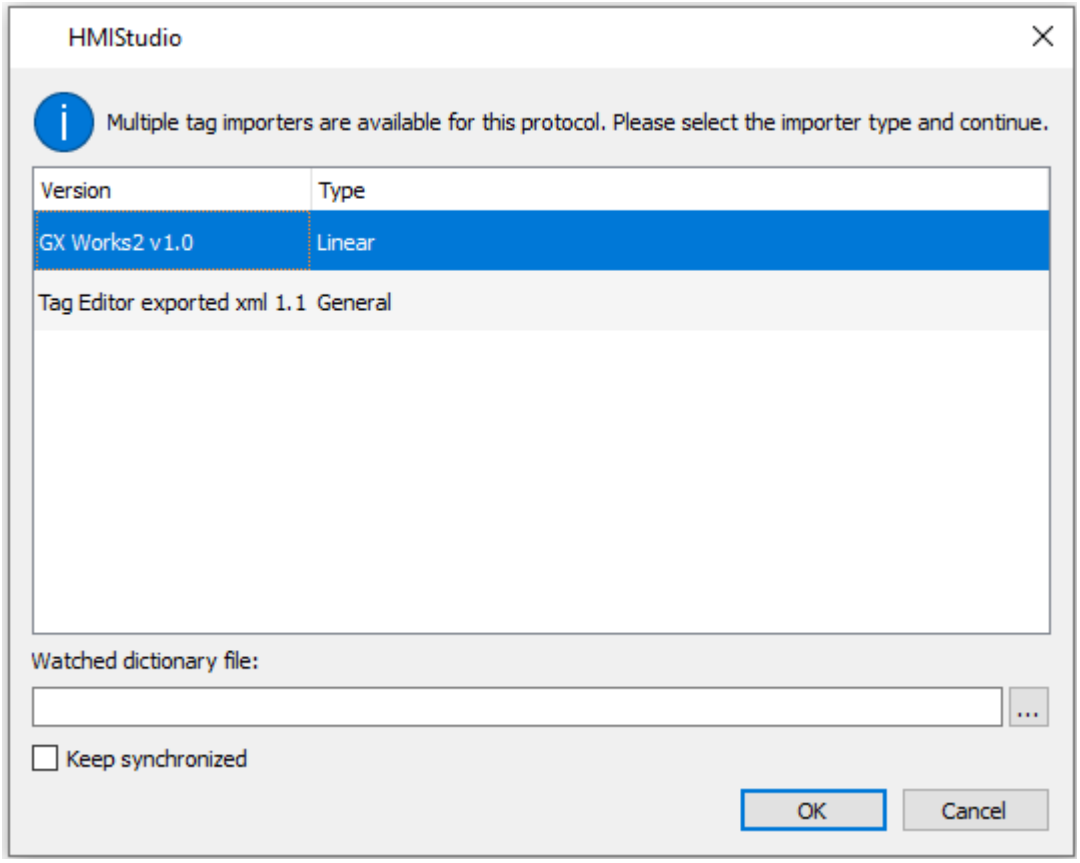


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



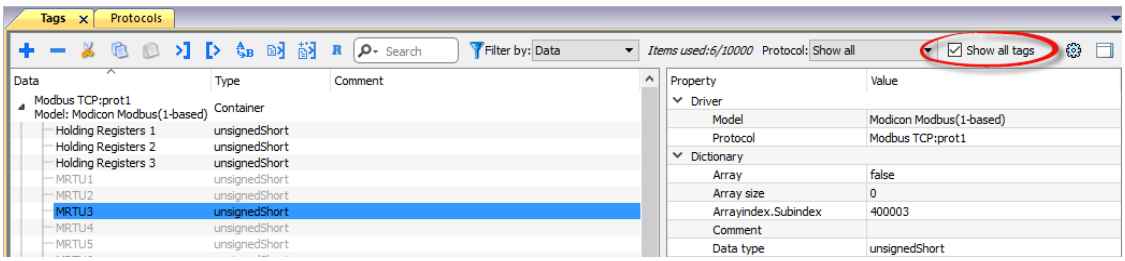
The following dialog shows which importer type can be selected.

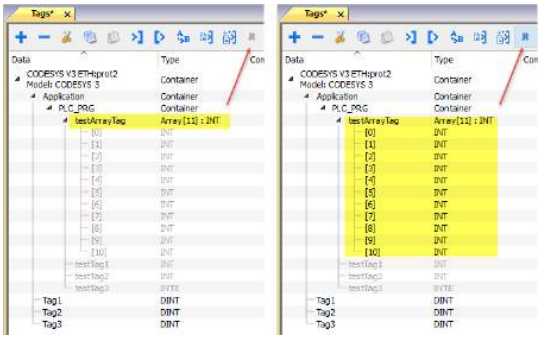
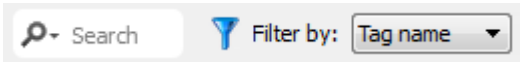


Importer	Description
GX Works2 v1.0 Linear	Requires a **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the

Error	Notes
	controller is connected and properly configured to get network access
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits); ensure the communication parameter settings of the controller is compatible with panel communication setup
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
General Error	Error cannot be identified; should never be reported; contact technical support

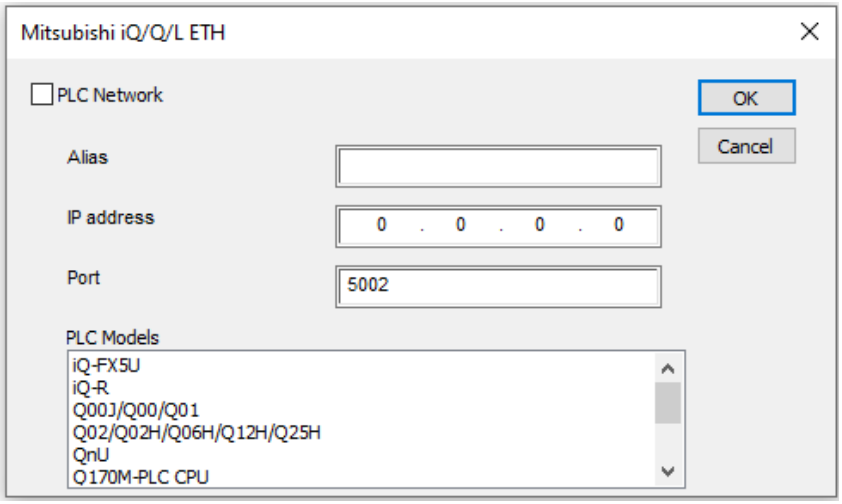
Mitsubishi iQ/Q/L ETH

The Mitsubishi iQ/Q/L ETH driver supports communication with Mitsubishi controllers with integrated Ethernet port and with external Ethernet card (QJ71E71-100).

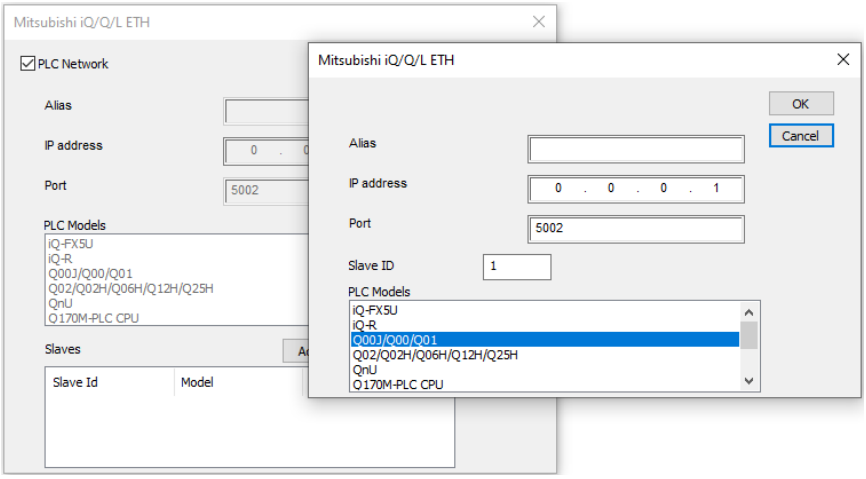
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “Mitsubishi iQ/Q/L ETH” from the list of available protocols.

The driver configuration dialog is shown as in the following figure:



Element	Description
IP address	Ethernet IP address of the controller
Port	Specifies the port number (decimal) used in the communication with the PLC.

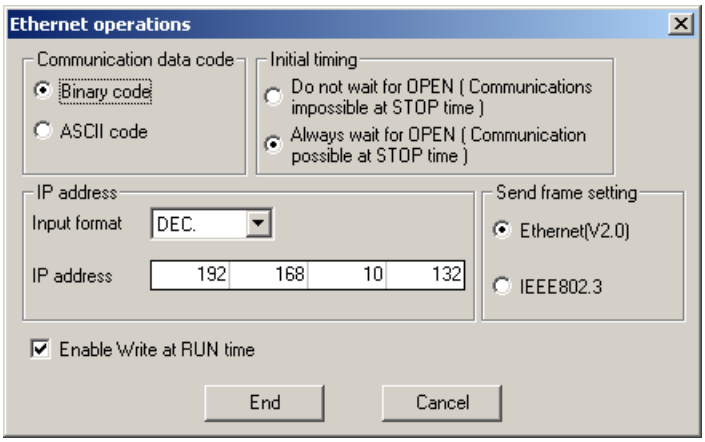
Element	Description
PLC Model	The driver supports communication with different Mitsubishi iQ, Q and L controllers. Note: PLC Model selection has only effect on range values of variables. If a particular model is not present in the list, try selecting a similar one. If range values of variables are the same, the communication will be correctly established.
PLC Network	The protocol allows the connection of multiple controllers to one HMI device. To set-up multiple connections, check “PLC network” checkbox and create your network using the command “Add” per each slave device you need to include in the network. 

Controller Settings

GX Works2

The Mitsubishi Q system must be properly configured for Ethernet communication using the Mitsubishi GX Developer software version 7 or higher, from GX Works2 software.

The Figure below shows an example of network configuration for Ethernet communication.

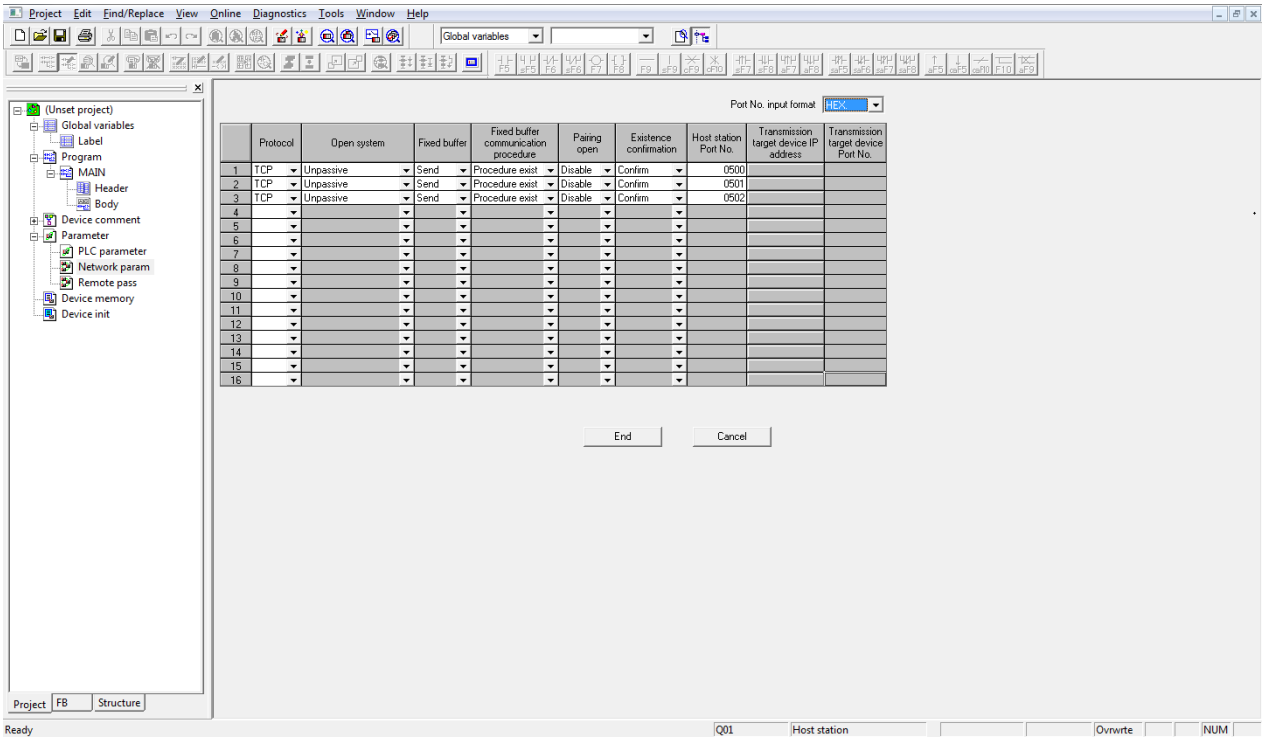


Please note that the communication protocol supports only Binary code communication.

The PLC system must be configured to accept incoming data from the external device.

In the GX Developer Software open “Parameters”, “Network Param” and select Ethernet/ CC IE/ MELSECNET”. Add the number of connections of the operator panels you want to configure in the network.

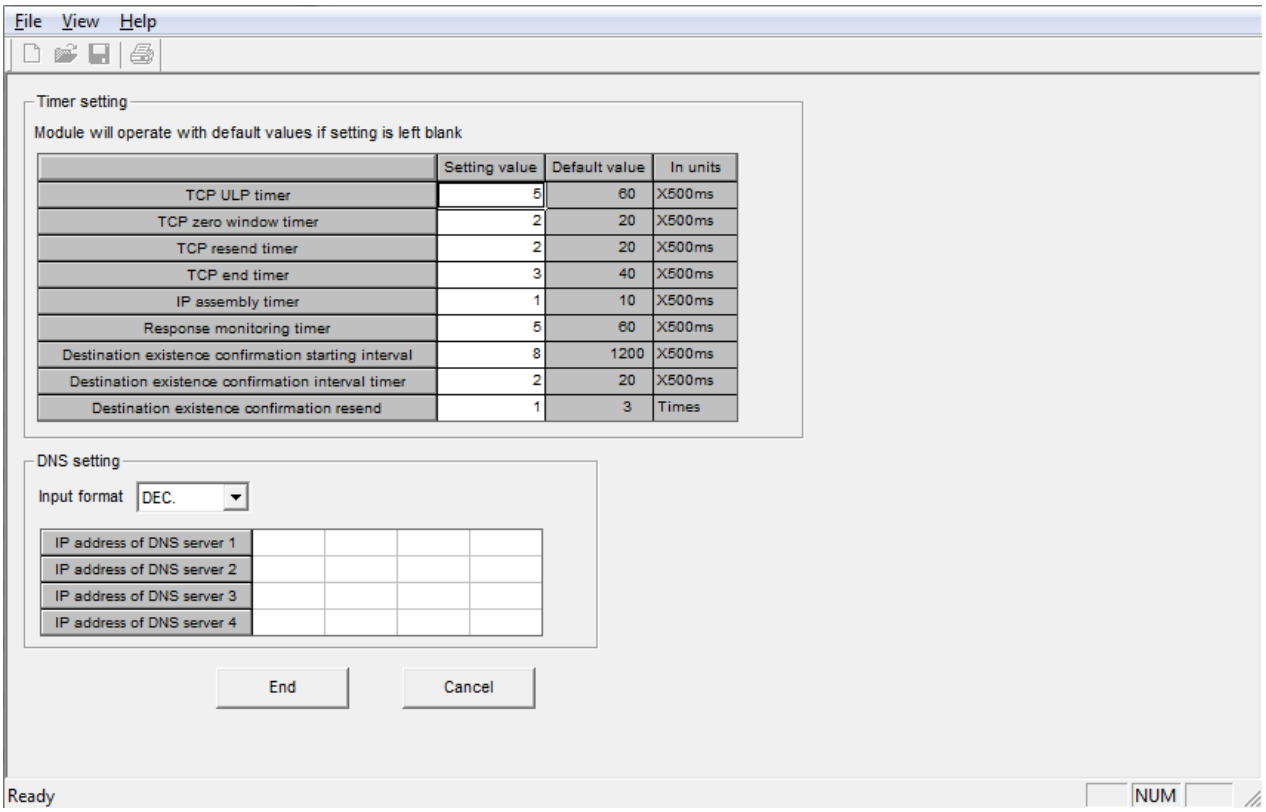
When using the Mitsubishi CPU with external Ethernet card (QJ71E71-100) the connections have to be configured according to the following figure as "Unpassive":

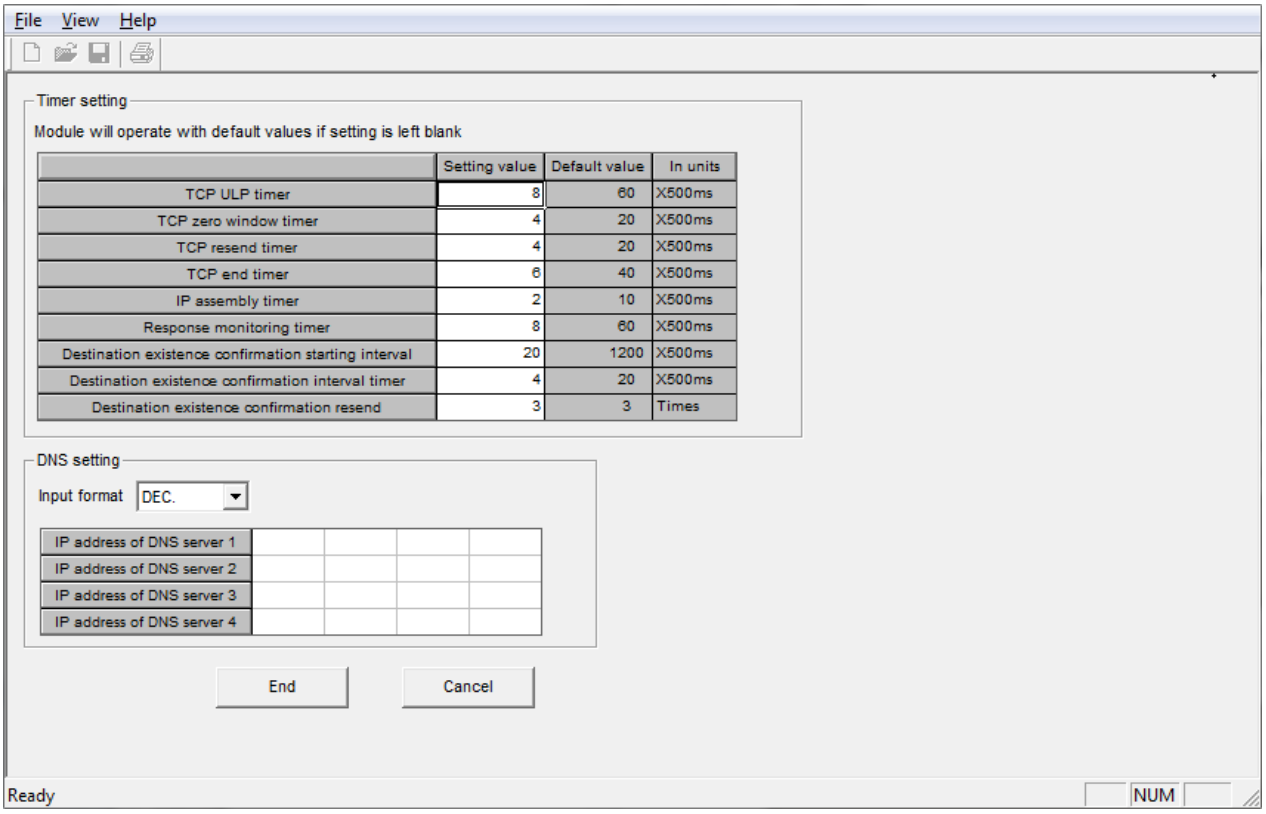


When the “Existence confirmation” setting has been set to Confirm, the TCP connection will be closed when it is not used (connection lost); by default the TCP port remains open and it is not possible to reconnect.

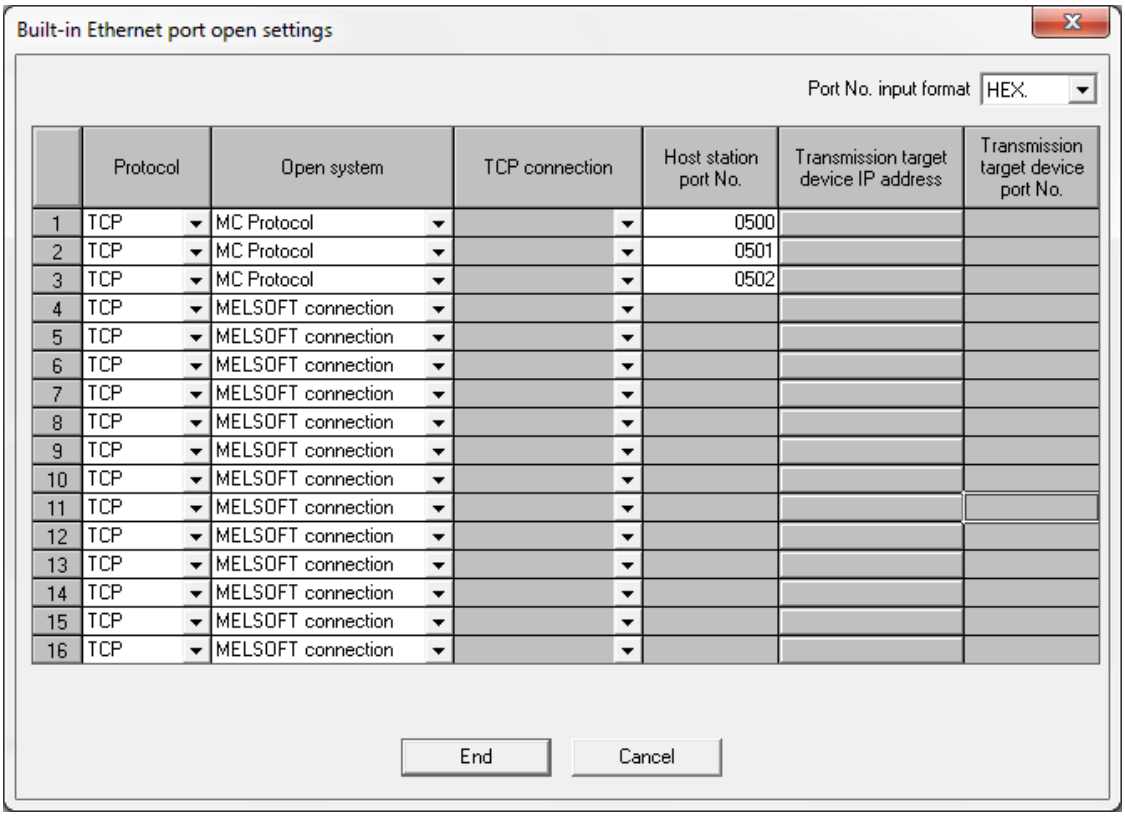
Note: The GX Developer software allows entering the conventional representation settings (decimal or hexadecimal) for the port number; in the above figure it is in hexadecimal.

In the next figures there are 2 examples about how to set “Initial settings” for 5 and 15 seconds timeout.





When using Mitsubishi CPU with integrated Ethernet port the "Open System" settings should be changed to "MC connection"



Note: The number format for Host Station Port No. is hexadecimal, not decimal.

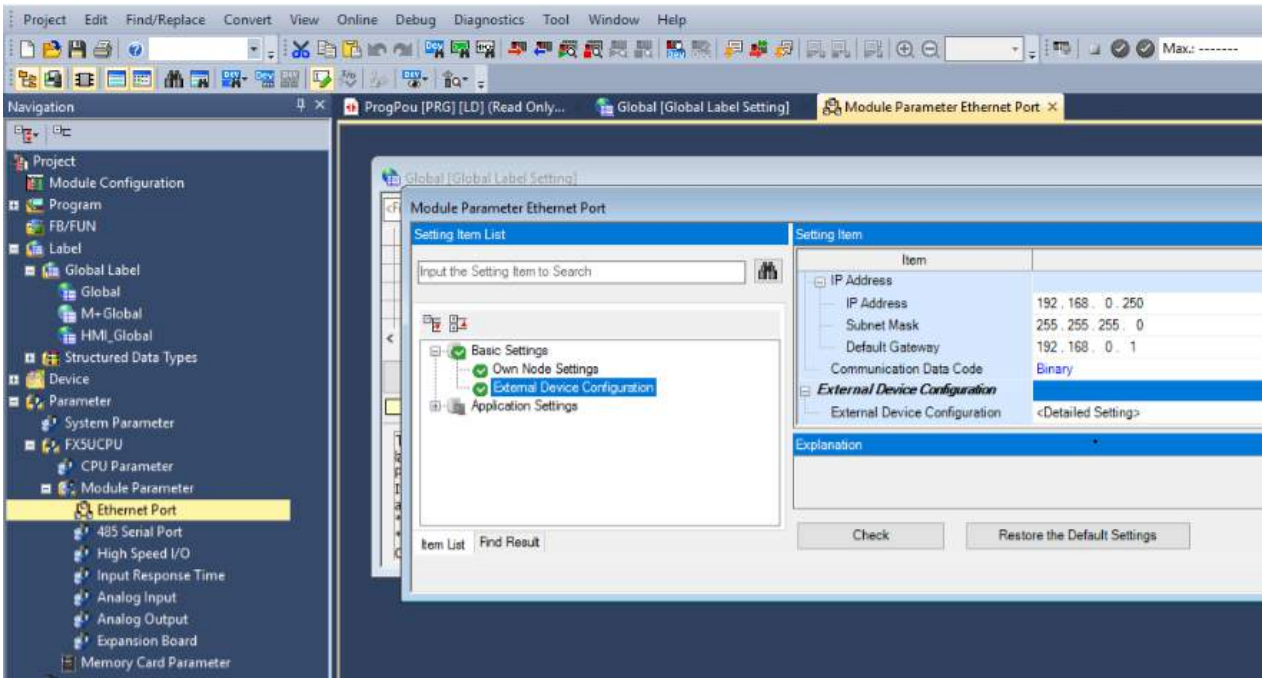
GX Works3

The Mitsubishi Q system must be properly configured for Ethernet communication using GX Works3 software.

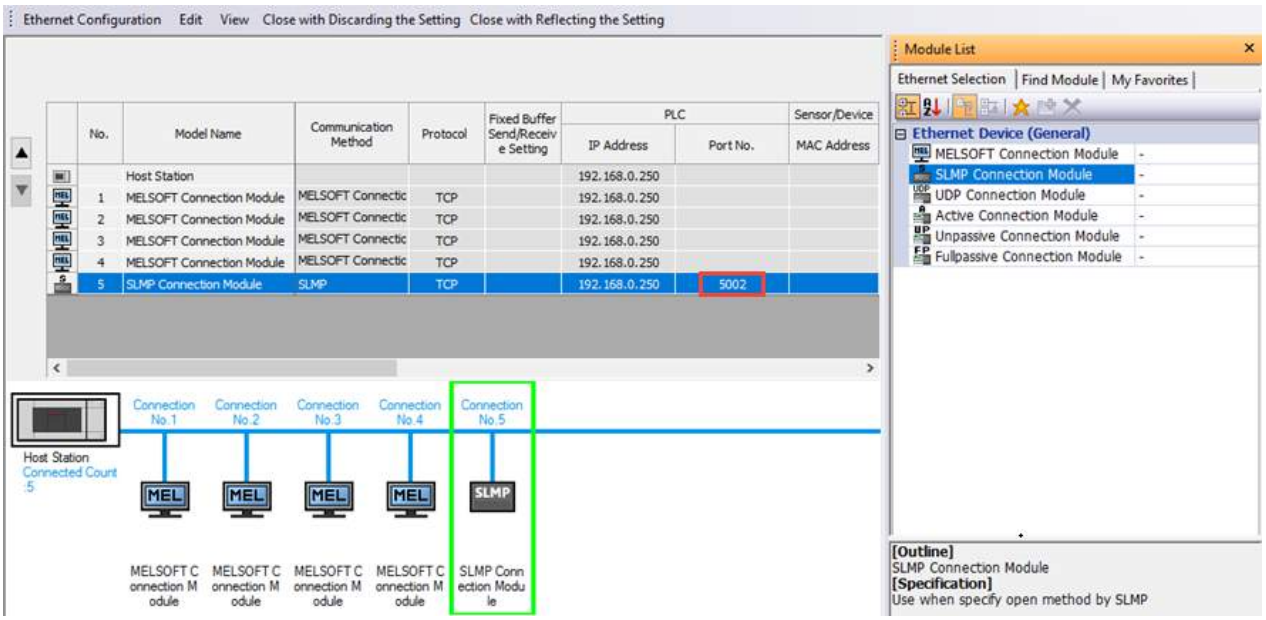
The communication driver is based on SLMP function.

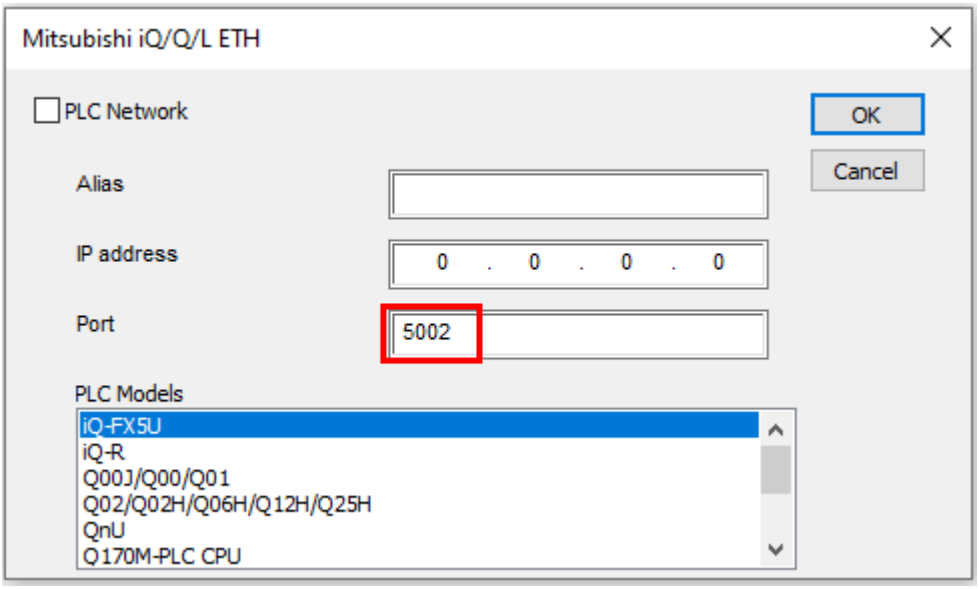
SLMP (Seamless Message Protocol) is a protocol for accessing SLMP-compatible devices from an external device (such as HMI) using TCP or UDP through Ethernet.

From GX Works3 software, Ethernet port parameters must be set from **Module parameter > Ethernet Port > Basic Settings > Own Node Settings**.



SLMP Connection Module must be added in **Module parameter > Ethernet Port > Basic Settings > External Device Configuration > Detailed Settings > Ethernet Configuration (Built-in Ethernet Port)**. **Port No.** parameter must be the same as per **Port** parameter from Protocol Editor Settings (see images below).



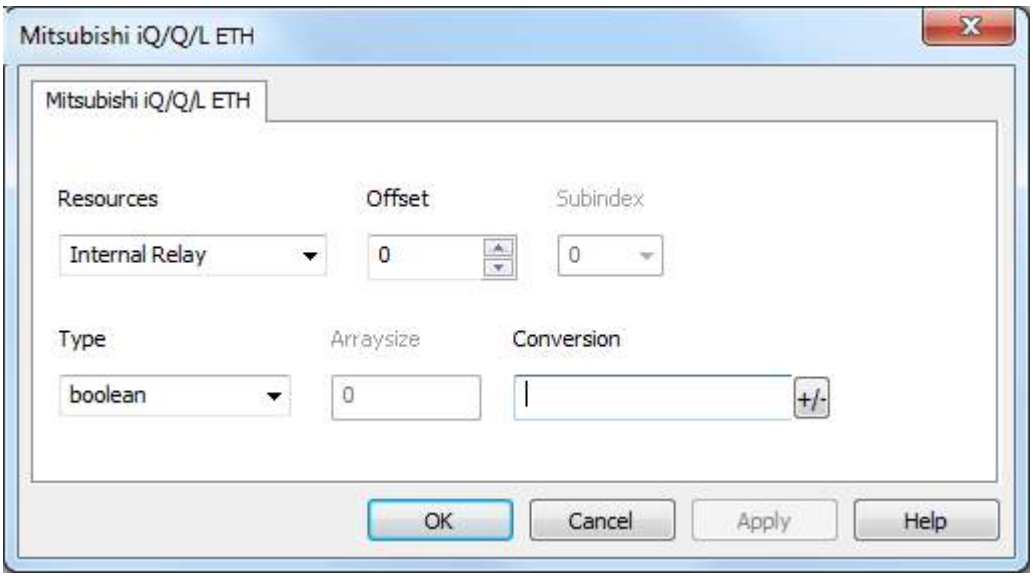


Note: To actually get communication with HMI it is necessary to initialize the PLC after the above settings have been applied.
To initialize the PLC it possibile to use the Run/Stop/Reset switch or by simply rebooting the PLC.

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Mitsubishi iQ/Q/L ETH** from the protocol list: tag definition dialog is displayed.

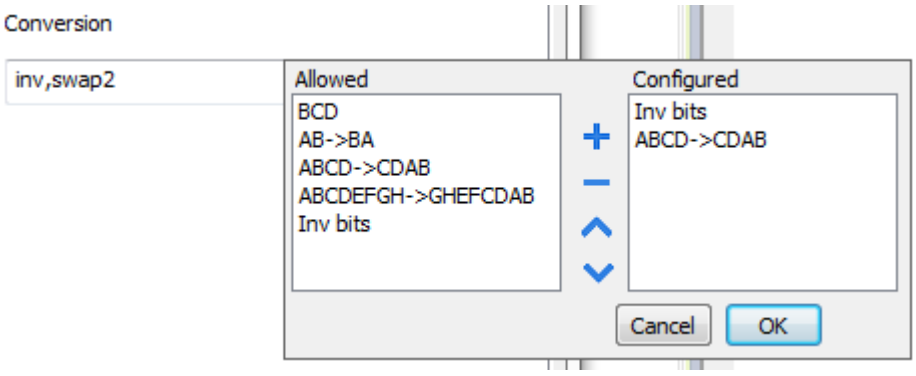


<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Resources</td><td>PLC resources. Available resources are:
Internal Relay
Error Relay
Input (hex)
Output (hex)
Latch Relay
Link Relay
Data Register
Link Register
Timer (Current)
Counter (Current)
Timer (Switch)
Timer (Coil)
Counter (Switch)
Counter (Coil)
Special Relay
Special Register
File Register
Input (oct)
Output (oct)</td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.</td></tr>
<tr><td>SubIndex</td><td>Allows resource offset selection.</td></tr>
<tr><td>Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

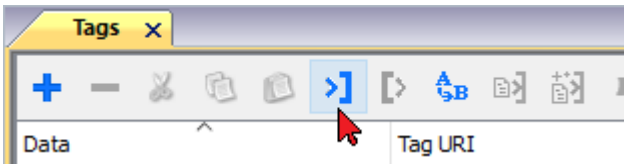
Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 1 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

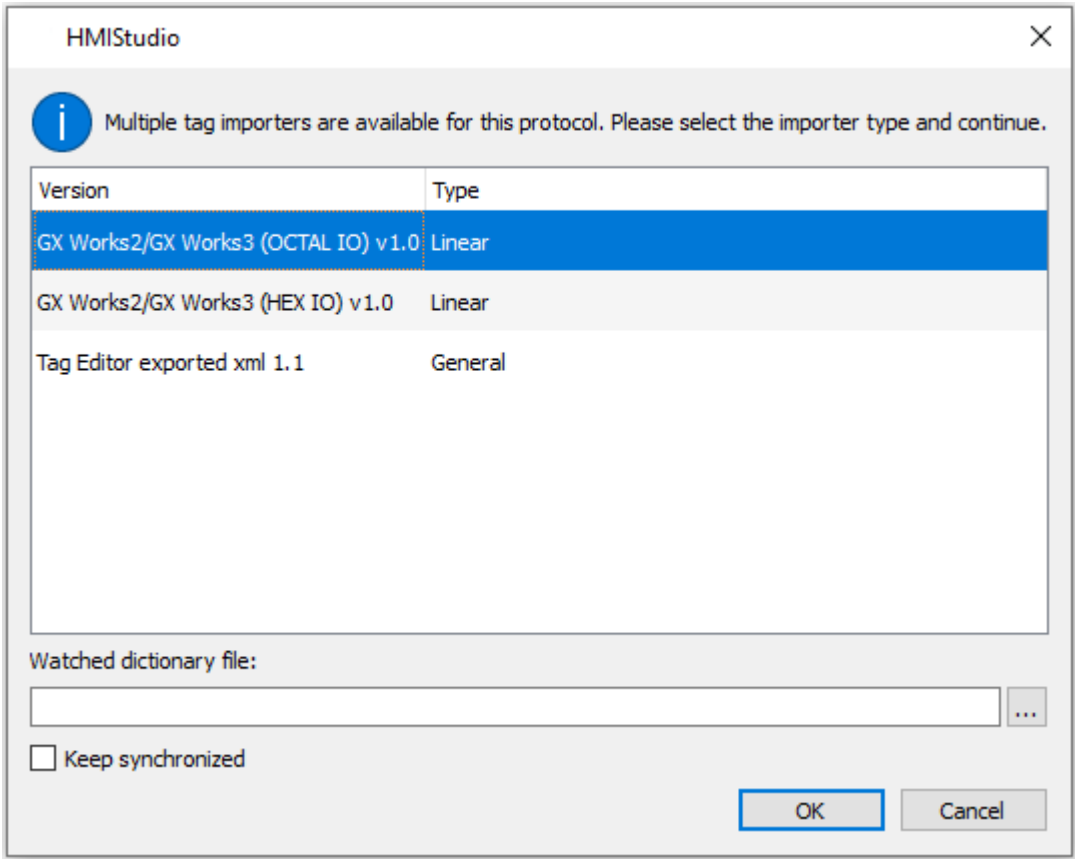
Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



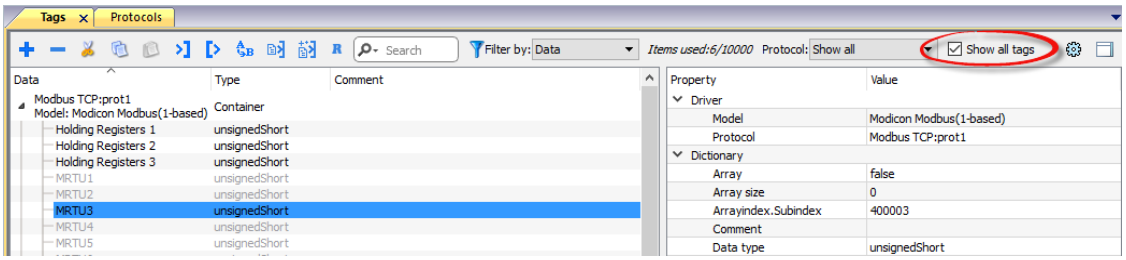
The following dialog shows which importer type can be selected.



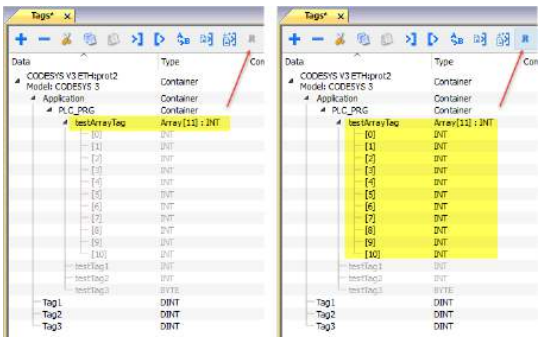
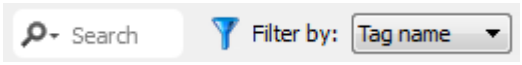
Importer	Description
GX Works2/GX Works3 v1.0 Linear	Requires a **.csv**file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	Import Tag(s). Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	Update Tag(s). Click on this icon to update the tags in the project, due a new dictionary import.
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
General Error	Error cannot be identified; should never be reported; contact technical support

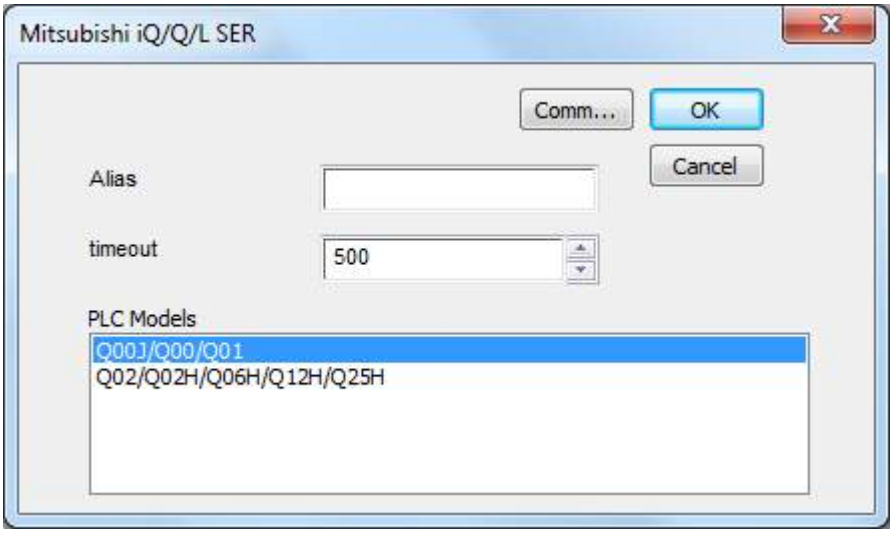
Mitsubishi iQ/Q/L SER

The Mitsubishi iQ/Q/L SER driver supports communication with Mitsubishi controllers with integrated serial port.

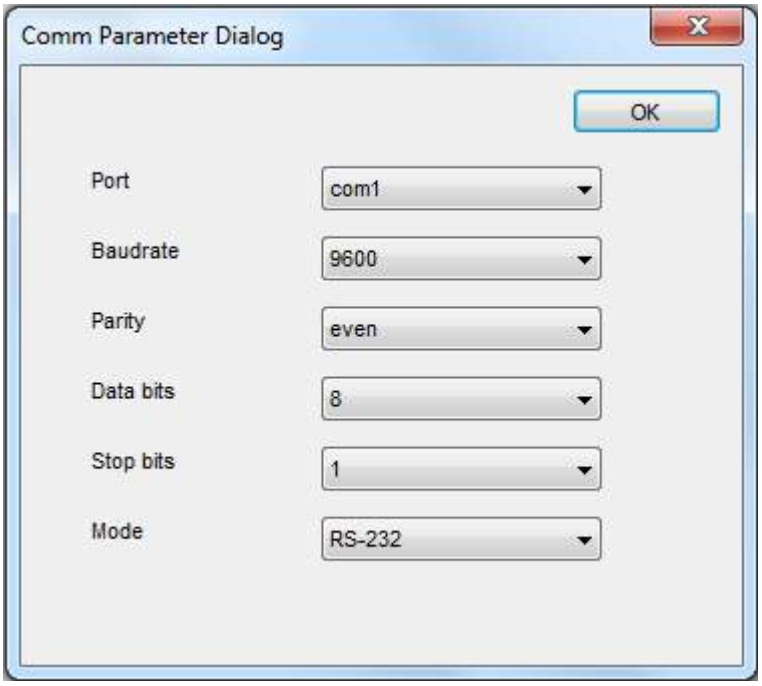
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “Mitsubishi iQ/Q/L SER” from the list of available protocols.

The driver configuration dialog is shown as in the following figure:



Element	Description
Alias	Name identifying PLC. The name will be added as a prefix to each tag name.
timeout	Time delay in milliseconds between two retries in case of missing response from the device.

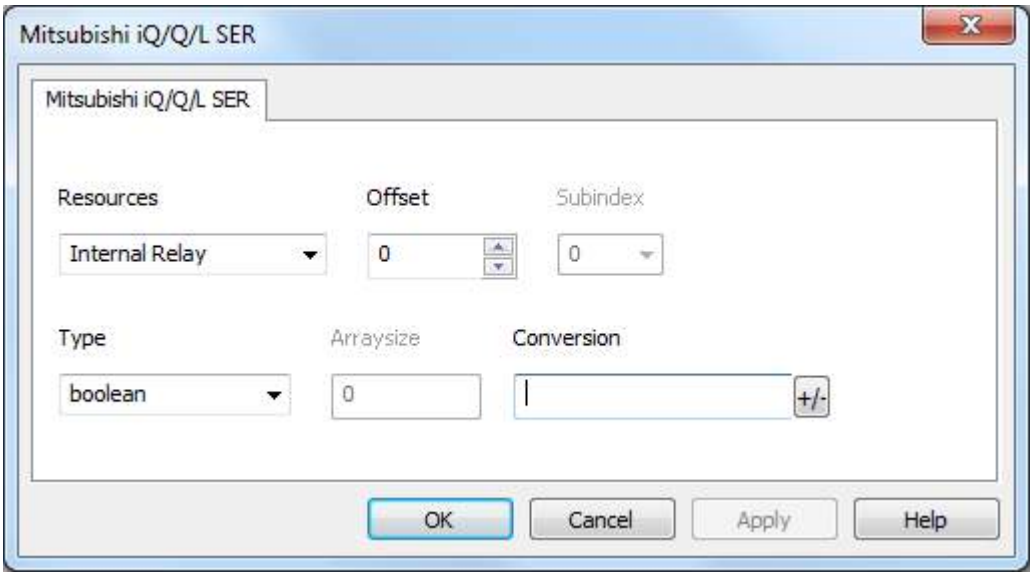
Element	Description
PLC Model	The driver supports communication with different Mitsubishi iQ, Q and L controllers. Note: PLC Model selection has only effect on range values of variables. If a particular model is not present in the list, try selecting a similar one. If range values of variables are the same, the communication will be correctly established.
Comm	If clicked displays the communication parameters setup dialog. 

Element	Parameter
Port	Serial port selection. • **COM1**: On-board port • **COM2**: Optional Plug-in module plugged on slot#1 or slot#2 • **COM3**: Optional Plug-in module plugged on slot#3 or slot#4
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.
Mode	Serial port mode. Available modes: • **RS-232**. • **RS-485** (2 wires). • **RS-422** (4 wires).

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Mitsubishi iQ/Q/L SER** from the protocol list: tag definition dialog is displayed.

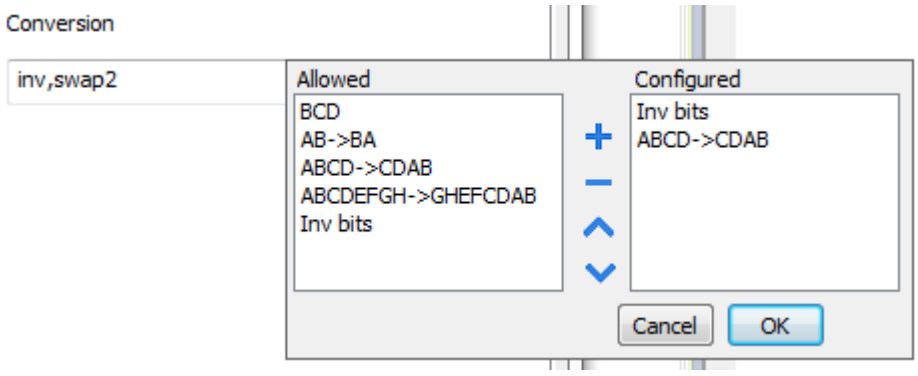


<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Resources</td><td>PLC resources. Available resources are:
Internal Relay
Error Relay
Input (hex)
Output (hex)
Latch Relay
Link Relay
Data Register
Link Register
Timer (Current)
Counter (Current)
Timer (Switch)
Timer (Coil)
Counter (Switch)
Counter (Coil)
Special Relay
Special Register
File Register
Input (oct)
Output (oct)</td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.</td></tr>
<tr><td>SubIndex</td><td>Allows resource offset selection.</td></tr>
<tr><td>Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

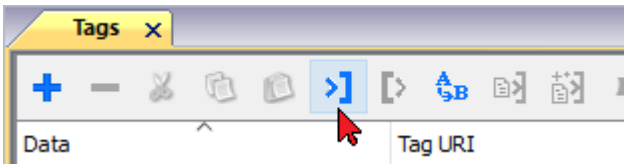
Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001 → 1 10000011100 1010101000010100010111011011011001011011000010011101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

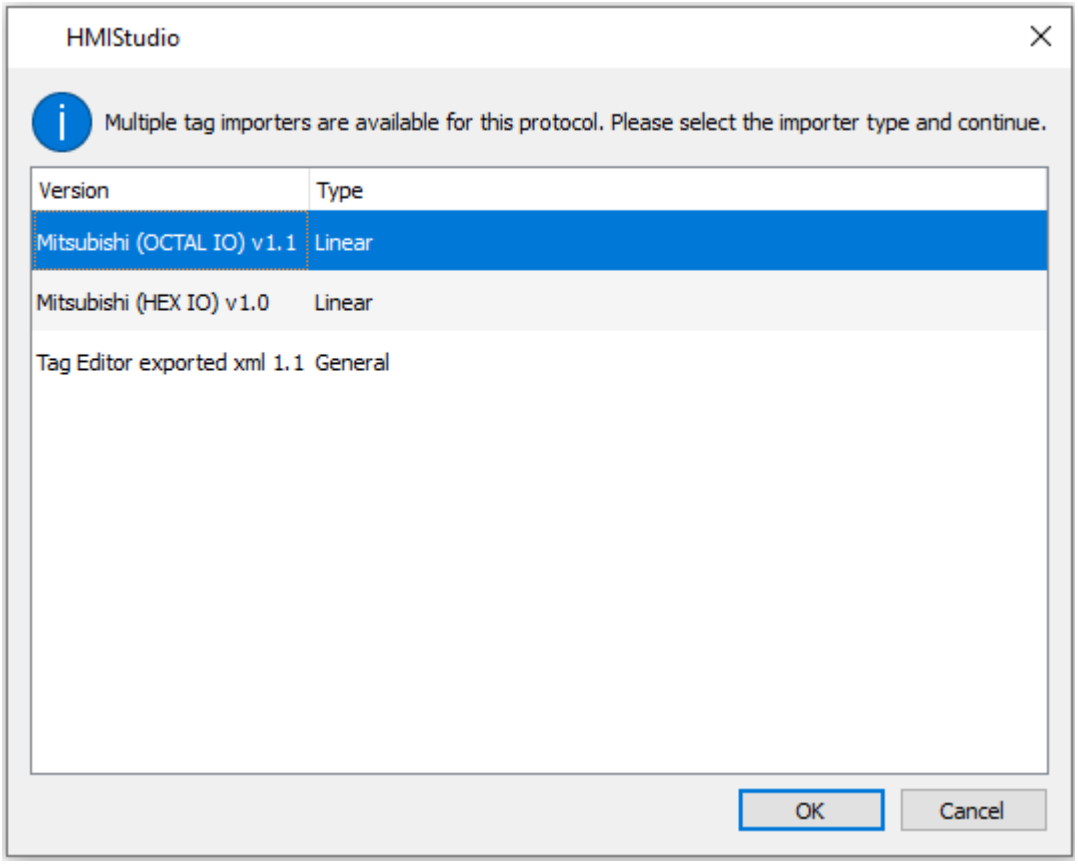
Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



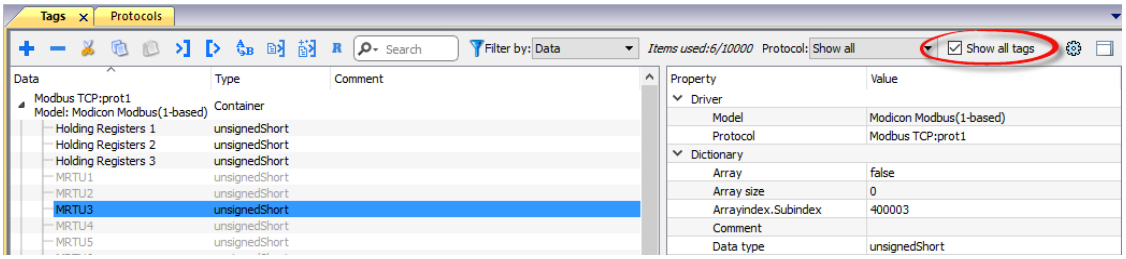
The following dialog shows which importer type can be selected.



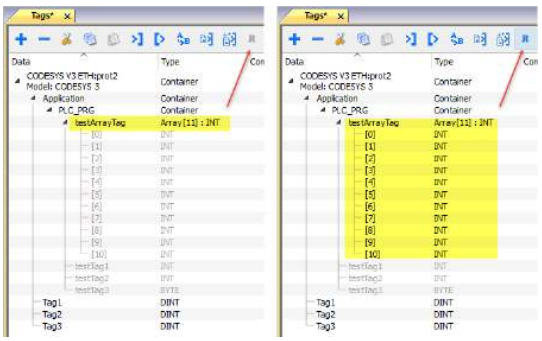
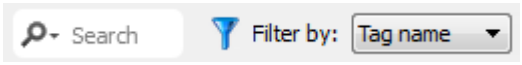
Importer	Description
Mitsubishi v1.1 Linear	Requires a **.csv**file generated by GX Works2/GX Works3 software. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	Import Tag(s). Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	Update Tag(s). Click on this icon to update the tags in the project, due a new dictionary import.
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
General Error	Error cannot be identified; should never be reported; contact technical support

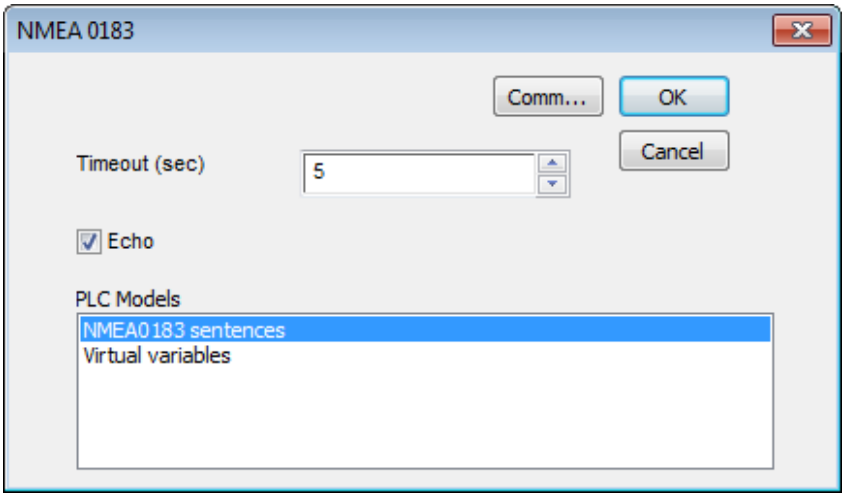
NMEA 0183

The NMEA 0183 driver has been developed to communicate with NMEA 0183 compatible devices trough the operator panel serial ports.

Protocol Editor Settings

Add (+) a new driver in the Protocol editor and select the protocol called “NMEA 0183” from the list of available protocols.

The driver configuration dialog is shown in the following figure.

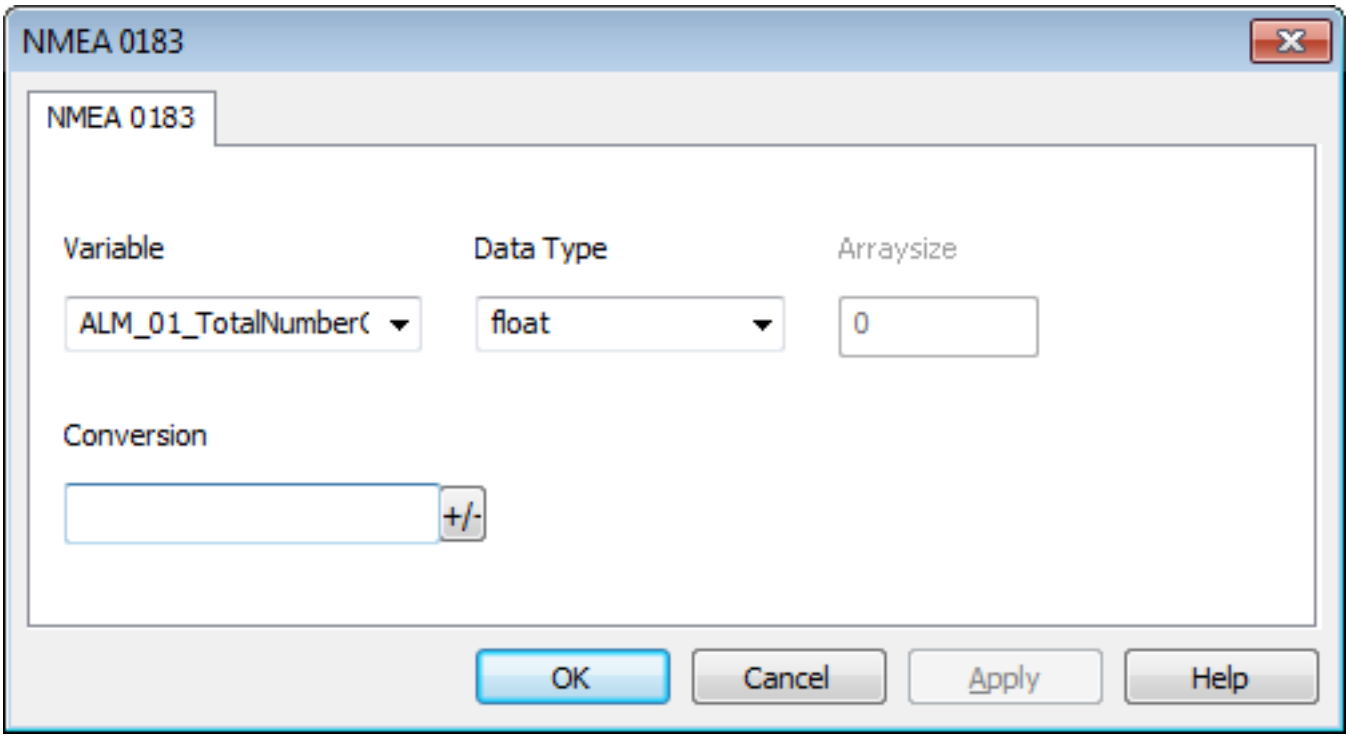


Element	Description
Timeout (sec)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. It is expressed in seconds.
Echo	If selected the NMEA messages received on the RX channel of serial port are sent out from the TX channel. This allows to continue the NMEA network downstream of the operator panel whether required.
PLC Models	Two PLC models are available: NMEA 0183 Sentences: when selected the Tags will point univocally to the specified NMEA sentence. Virtual variables: when selected the Tag will show the value coming from any NMEA sentence of the specified type, for example any NMEA sentence of Latitude type.

Tag Editor Settings

Into Tag editor select the protocol “NMEA 0183” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:



Element	Description
Variable	The NMEA Sentence or Virtual variable
Data Type	See table below
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.

Data Type	Memory Space	Limits
boolean	1 bit data	0 ... 1
byte	8-bit data	-128 ... 127
short	16-bit data	-32768 ... 32767
int	32-bit data	-2.1e9 ... 2.1e9
unsignedByte	8-bit data	0 ... 255
unsignedShort	16-bit data	0 ... 65535
unsignedInt	32-bit data	0 ... 4.2e9
float	IEEE single-precision 32-bit floating point type	1.17e-38 ... 3.40e38
string	String data	

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the tag.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Degrees</td><td>Shows Degrees data only from coordinates sentence</td></tr>
<tr><td>Minutes</td><td>Shows Minutes data only from coordinates sentence</td></tr>
<tr><td>Seconds</td><td>Shows Seconds data only from coordinates sentence</td></tr>
</table>
</td></tr>
</table>

List of supported NMEA 0183 commands

The NMEA 0183 commands supported from the communication protocol are the following:

AAM_01_StatusArrivalCircle

AAM_02_StatusPerpendicular

AAM_03_ArrivalCircleRadius

AAM_04_UnitsOfRadius

AAM_05_WaypointID

ACK_01_LocalAlarmNumber

ALM_01_TotalNumberOfMessages

ALM_02_MessageNumber

ALM_03_SatelliteNumber

ALM_04_WeekNumber

ALM_05_SVhealth

ALM_06_Eccentricity

ALM_07_AlmanacReferenceTime

ALM_08_InclinacionAngle

ALM_09_RateOfRightAscension

ALM_10_RootOfSemimajorAxis

ALM_11_ArgumentOfPerigee

ALM_12_LongitudeOfAscesionNode

ALM_13_MeanAnomaly

ALM_14_ClockParameter0

ALM_15_ClockParameter1

ALR_01_TimeOfAlarmConditionChange

ALR_02_LocalAlarmNumber

ALR_03_AlarmCondition

ALR_04_AlarmAcknowledgeState

ALR_05_AlarmDescriptionState

APB_01_StatusSNR

APB_02_StatusLock

APB_03_MagnitudeOfXTE

APB_04_DirectionToStear

APB_05_UnitsXTE

APB_06_StatusArrivalCircle

APB_07_StatusPerpendicular

APB_08_BearingOriginToDestination

APB_09_MagneticOrTrue

APB_10_DestinatonWaypointID

APB_11_Bearing

APB_12_BearingMagneticOrTrue

APB_13_HeadingToSteer

APB_14_HeadingMagneticOrTrue

APB_15_ModeIndicator

BEC_01_ObservationUTC

BEC_02_WaypointLatitude

BEC_03_WaypointLatitudeInd

BEC_04_WaypointLongitude

BEC_05_WaypointLongitudeInd

BEC_06_BearingTrue

BEC_07_BearingTrueInd

BEC_08_BearingMagnetic

BEC_09_BearingMagneticInd)

BEC_10_Distance

BEC_11_DistanceUnits

BEC_12_WaypointID

BOD_01_BearingTrue

BOD_02_BearingTrueInd

BOD_03_BearingMagnetic

BOD_04_BearingMagneticInd

BOD_05_DestinationWaypointID

BOD_06_OriginWaypointID

BWC_01_ObservationUTC

BWC_02_WaypointLatitude

BWC_03_WaypointLatitudeInd

BWC_04_WaypointLongitude

BWC_05_WaypointLongitudeInd)

BWC_06_BearingTrue

BWC_07_BearingTrueInd

BWC_08_BearingMagnetic

BWC_09_BearingMagneticInd

BWC_10_Distance

BWC_11_DistanceUnits

BWC_12_WaypointID

BWC_13_ModeIndicator

BWR_01_ObservationUTC

BWR_02_WaypointLatitude

BWR_03_WaypointLatitudeInd

BWR_04_WaypointLongitude

BWR_05_WaypointLongitudeInd

BWR_06_BearingTrue

BWR_07_BearingTrueInd

BWR_08_BearingMagnetic

BWR_09_BearingMagneticInd

BWR_10_Distance

BWR_11_DistanceInd

BWR_12_WaypointID

BWR_13_ModeIndicator

BWW_01_BearingTrue

BWW_02_BearingTrueInd

BWW_03_BearingMagnetic

BWW_04_BearingMagneticInd

BWW_05_ToWaypointID

BWW_06_FromWaypointID

DBT_01_WaterDepthFeet

DBT_02_WaterDepthFeetInd

DBT_03_WaterDepthMeters

DBT_04_WaterDepthMetersInd

DBT_05_WaterDepthFathoms

DBT_06_WaterDepthFathomsInd

DCN_01_DeccaChainIdentifier

DCN_02_RedZoneIdentifier

DCN_03_RedLineOfPosition

DCN_04_StatusRedMasterLine

DCN_05_GreenZoneIdentifier

DCN_06_GreenLineOfPosition

DCN_07_StatusGreenMasterLine

DCN_08_PurpleZoneIdentifier

DCN_09_PurpleLineOfPosition

DCN_10_StatusPurpleMasterLine

DCN_11_RedLineNavigationUse, A=Valid

DCN_12_GreenLineNavigationUse, A=Valid

DCN_13_PurpleLineNavigationUse, A=Valid

DCN_14_PositionUncertainty

DCN_15_PositionUncertaintyInd

DCN_16_FixDataBasis

DPT_01_WaterDepth

DPT_02_OffsetFromTransducer

DPT_03_MaximumRangeScale

DSC_01_FormatSpecifier

DSC_02_Address

DSC_03_Cattegory

DSC_04_NatureOfDistress

DSC_05_TypeOfCommunication

DSC_06_PositionOrChannel

DSC_07_TimeOrTelNo

DSC_08_ShipMMSI

DSC_09_NatureOfDistress

DSC_10_Acknowledgment

DSC_11_ExpansionIndicator

DSE_01_TotalNumberOfMessages

DSE_02_MessageNumber

DSE_03_Query_ReplyFlag

DSE_04_Vessel_MMSI

DSE_05_DataSet1Code

DSE_06_Dataset1Data

DSE_07_Dataset2Code

DSE_08_Dataset2Data

DSE_09_Dataset3Code

DSE_10_Dataset3Data

DSE_11_Dataset4Code

DSE_12_Dataset4Data

DSE_13_Dataset5Code

DSE_14_Dataset5Data

DSE_15_Dataset6Code

DSE_16_Dataset6Data

DSE_17_Dataset7Code

DSE_18_Dataset7Data

DSE_19_Dataset8Code

DSE_20_Dataset8Data

DSE_21_Dataset9Code

DSE_22_Dataset9Data

DSE_23_Dataset10Code

DSE_24_Dataset10Data

DSI_01_TotalNumberOfMessages

DSI_02_MessageNumber

DSI_03_Vessel_MMSI

DSI_04_VesselCourse

DSI_05_VesselType

DSI_06_GeographicArea

DSI_07_Commandset1Code

DSI_08_Commandset1Data

DSI_09_Commandset2Code

DSI_10_Commandset2Data

DSI_11_Commandset3Code

DSI_12_Commandset3Data

DSI_13_ExpansionIndicator

DSR_01_TotalNumberOfMessages

DSR_02_MessageNumber

DSR_03_Vessel_MMSI

DSR_04_Dataset1Code

DSR_05_Dataset1Data

DSR_06_Dataset2Code

DSR_07_Dataset2Data

DSR_08_Dataset3Code

DSR_09_Dataset3Data

DSR_10_ExpansionIndicator

DTM_01_LocalDatumCode

DTM_02_LocalDatumSubdivisioncode

DTM_03_LatOffset

DTM_04_LatOffsetInd

DTM_05_LonOffset

DTM_06_LonOffsetInd

DTM_07_AltitudeOffset

DTM_08_ReferenceDatumCode

FSI_01_TransmitingFrequency

FSI_02_ReceivingFrequency

FSI_03_ModeOfOperation

FSI_04_PowerLevel

GBS_01_UTC

GBS_02_ExpectedLatitudeError

GBS_03_ExpectedLongitudeError

GBS_04_ExpectedAltitudeError

GBS_05_FailedSatelliteID

GBS_06_ProbabilityOfMissedDetection

GBS_07_EstimateOfBiasMeters

GBS_08_StandardDeviationOfBiasEstimate

GGA_01_UTC

GGA_02_Latitude

GGA_03_LatitudeInd

GGA_04_Longitude

GGA_05_LongitudeInd

GGA_06_QualityIndicator

GGA_07_NumberOfSatellitesInUse

GGA_08_HorizontalDilutionOfPrecision

GGA_09_Altitude

GGA_10_AltitudeInd

GGA_11_GeoidalSeparation

GGA_12_GeoidalSeparationInd

GGA_13_AgeOfDifferentialData

GGA_14_DifferentialReferenceID

GLC_01_GRI

GLC_02_MasterTOA

GLC_03_SignalStatus1

GLC_04_TD1

GLC_05_SignalStatus2

GLC_06_TD2

GLC_07_SignalStatus3

GLC_08_TD3

GLC_09_SignalStatus4

GLC_10_TD4

GLC_11_SignalStatus5

GLC_12_TD5

GLC_13_SignalStatus6

GLL_01_Latitude

GLL_02_LatitudeInd

GLL_03_Longitude

GLL_04_LongitudeInd

GLL_05_UTC

GLL_06_Status

GLL_07_ModeIndicator

GNS_01_UTC

GNS_02_Latitude

GNS_03_LatitudeInd

GNS_04_Longitude

GNS_05_LongitudeInd

GNS_06_ModeIndicator

GNS_07_NumberOfSatellitesInUse

GNS_08_HDOP

GNS_09_AntennaAltitude

GNS_10_GeoidalSeparation

GNS_11_AgeOfDifferentialData

GNS_12_DifferentialStationID

GRS_01_UTC

GRS_02_Mode

GRS_03_RangeResidual

GRS_04_RangeResidual

GRS_05_RangeResidual

GRS_06_RangeResidual

GRS_07_RangeResidual

GRS_08_RangeResidual

GRS_09_RangeResidual

GRS_10_RangeResidual

GRS_11_RangeResidual

GRS_12_RangeResidual

GRS_13_RangeResidual

GRS_14_RangeResidual

GSA_01_Mode

GSA_02_Mode

GSA_03_ID

GSA_04_ID

GSA_05_ID

GSA_06_ID

GSA_07_ID

GSA_08_ID

GSA_09_ID

GSA_10_ID

GSA_11_ID

GSA_12_ID

GSA_13_ID

GSA_14_ID

GSA_15_PDOP

GSA_16_HDOP

GSA_17_VDOP

GST_01_UTC

GST_02_RMSvalueOfStandardDeviation

GST_03_StandardDeviationOfSemiMajorAxis

GST_04_StandardDeviationOfSemiMinorAxis

GST_05_OrientationOfSemiMajorAxis

GST_06_StandardDeviationOfLatitude

GST_07_StandardDeviationOfLongitude

GST_08_StandardDeviationOfAltitude

GSV_01_NumberOfMessages

GSV_02_MessageNumber

GSV_03_NumberOfSatellitesInView

GSV_04_SET1_SatelliteID

GSV_05_SET1_Elevation

GSV_06_SET1_Azimuth

GSV_07_SET1_SNR

GSV_08_SET2_SatelliteID

GSV_09_SET2_Elevation

GSV_10_SET2_Azimuth

GSV_11_SET2_SNR

GSV_12_SET3_SatelliteID

GSV_13_SET3_Elevation

GSV_14_SET3_Azimuth

GSV_15_SET3_SNR

GSV_16_SET4_SatelliteID

GSV_17_SET4_Elevation

GSV_18_SET4_Azimuth

GSV_19_SET4_SNR

HDG_01_MagneticHeading

HDG_02_MagneticDeviation

HDG_03_MagneticDeviationInd

HDG_04_MagneticVariation

HDG_05_MagneticVariation

HDM_01_MagneticHeading

HDM_02_MagneticHeadingInd

HDT_01_Heading

HDT_02_HeadingInd

HMR_01_HeadingSensor1ID

HMR_02_HeadingSensor2ID

HMR_03_DifferenceLimit

HMR_04_HeadingSensorDifference

HMR_05_WarningFlag

HMR_06_HeadingReadingSensor1

HMR_07_StatusSensor1

HMR_08_TypeSensor1

HMR_09_DeviationSensor1

HMR_10_DeviationSensor1Ind)

HMR_11_HeadingReadingSensor

HMR_12_StatusSensor2

HMR_13_TypeSensor2

HMR_14_DeviationSensor2

HMR_15_DeviationSensor2Ind)

HMR_16_Variation

HMR_17_VariationInd)

HMS_01_HeadingSensor1ID

HMS_02_HeadingSensor2ID

HMS_03_MaximumDifference

HSC_01_CommandedHeading

HSC_02_CommandedHeadingInd

HSC_03_CommandedHeadingMagnetic

HSC_04_CommandedHeadingMagneticInd

HTC_01_Override

HTC_02_CommandedRudderAngle

HTC_03_CommandedRudderDirection

HTC_04_SelectedSteeringMmode

HTC_05_TurnMode

HTC_06_CommandedRudderLimit

HTC_07_CommandedOffHeadingLimit

HTC_08_CommandedRadiusOfTurn

HTC_09_CommandedRateOfTurn

HTC_10_CommandedHeadingToSteer

HTC_11_CommandedOffTrackLimit

HTC_12_CommandedTrack

HTC_13_HeadingReferenceInUse

HTD_01_Override

HTD_02_CommandedRudderAngle

HTD_03_CommandedRudderDirection

HTD_04_SelectedSteeringMode

HTD_05_TurnMode

HTD_06_CommandedRudderLimit

HTD_07_CommandedOffHeadingLimit

HTD_08_CommandedRadiusOfTurn

HTD_09_CommandedRateOfTurn

HTD_10_CommandedHeadingToSteer

HTD_11_CommandedOffTrackLimit

HTD_12_CommandedTrack

HTD_13_HeadingReferenceInUse

HTD_14_RudderStatus

HTD_15_OffHeadingStatus

HTD_16_OffTrackstatus

HTD_17_VesselHeading

LCD_01_GRI

LCD_02_MasterSNR

LCD_03_MasterECD

LCD_04_Secondary1_SNR

LCD_05_Secondary1_ECD

LCD_06_Secondary2_SNR

LCD_07_Secondary2_ECD

LCD_08_Secondary3_SNR

LCD_09_Secondary3_ECD

LCD_10_Secondary4_SNR

LCD_11_Secondary4_ECD

LCD_12_Secondary5_SNR

LCD_13_Secondary5_ECD

MDA_01_BarometricPressureInchesOfMercury

MDA_02_BarometricPressureInchesOfMercuryInd

MDA_03_Barometric pressureBars

MDA_04_Barometric pressureBarsInd

MDA_05_AirTemperature

MDA_06_AirTemperatureInd

MDA_07_WaterTemperature

MDA_08_WaterTemperatureInd

MDA_09_RelativeHumidity

MDA_10_AbsoluteHumidity

MDA_11_DewPoint

MDA_12_DewPointInd

MDA_13_WindDirectionTrue

MDA_14_WindDirectionTrueInd

MDA_15_WindDirectionMagnetic

MDA_16_WindDirectionMagneticInd

MDA_17_WindSpeedKnots

MDA_18_WindSpeedKnotsInd

MDA_19_WindSpeedMs

MDA_20_WindSpeedMsInd

MLA_01_TotalNumberOfMessages

MLA_02_MessageNumber

MLA_03_SatelliteID

MLA_04_CalendarDay

MLA_05_GeneralizedHealth

MLA_06_Eccentricity

MLA_07_DOT

MLA_08_ArgumentOfPerigee

MLA_09_SystemTimeScaleCorrectionMSB

MLA_10_CorrectionOfAverageValueDraconitic

MLA_11_TimeOfAscensionNode

MLA_12_GreenwichLongitude

MLA_13_CorrectionToAverageValueInclination

MLA_14_SystemTimeScaleCorrectionLSB

MLA_15_CourseValueOfTimeScaleShift

MSK_01_BeaconFrequency

MSK_02_Auto_Manual_Frequency

MSK_03_BeaconBitRate

MSK_04_Auto_Manual_BitRate

MSK_05_IntervalForSending

MSK_06_ChannelNumber

MSS_01_SignalStrength

MSS_02_SNR

MSS_03_BeaconFrequency

MSS_04_BeaconBitRate

MSS_05_ChannelNumber

MTW_01_Temperature

MTW_02_TemperatureInd

MWD_01_WindDirection

MWD_02_WindDirectionInd

MWD_03_WindDirectionMagnetic

MWD_04_WindDirectionMagneticInd

MWD_05_WindSpeedKnots

MWD_06_WindSpeedKnotsInd

MWD_07_WindSpeedMs

MWD_08_WindSpeedMsInd

MWV_01_WindAngle

MWV_02_Reference

MWV_03_WindSpeed

MWV_04_WindSpeedInd

MWV_05_Status

NMEA_Altitude

NMEA_Course

NMEA_Latitude

NMEA_LatitudeInd

NMEA_Longitude

NMEA_LongitudeInd

NMEA_SpeedKnots

NMEA_UTC

OSD_01_Heading

OSD_02_HeadingStatus

OSD_03_VesselCourse

OSD_04_CourseReference

OSD_05_VesselSpeed

OSD_06_SpeedReference

OSD_07_VesselSet

OSD_08_VesselDrift

OSD_09_SpeedUnits

RMA_01_Status

RMA_02_Latitude

RMA_03_LatitudeInd

RMA_04_Longitude

RMA_05_LongitudeInd

RMA_06_TimeDifferenceA

RMA_07_TimeDifferenceB

RMA_08_SpeedOverGroundKnots

RMA_09_CourseOverGround

RMA_10_MagneticVariation

RMA_11_MagneticVariationInd

RMA_12_ModeIndicator

RMB_01_DataStatus

RMB_02_CrossTrackError

RMB_03_DirectionToSteer

RMB_04_OriginWaypointID

RMB_05_DestinationwaypointID

RMB_06_DestinationwaypointLat

RMB_07_DestinationwaypointLatInd

RMB_08_DestinationWaypointLongitude

RMB_09_DestinationWaypointLongitudeInd

RMB_10_RangeToDestination

RMB_11_BearingToDestination

RMB_12_DestinationClosingVelocity

RMB_13_ArrivalStatus

RMB_14_ModeIndicator

RMC_01_UTC

RMC_02_Status

RMC_03_Latitude

RMC_04_LatitudeInd

RMC_05_Longitude

RMC_06_LongitudeInd

RMC_07_SpeedOverGround

RMC_08_CourseOverGround

RMC_09_Date

RMC_10_MagneticVariation

RMC_11_MagneticVariationInd

RMC_12_ModeIndicator

ROT_01_RateOfTurn

ROT_02_Status

RPM_01_SourceShaftEngine

RPM_02_EngineOfShaftNumber

RPM_03_Speed

RPM_04_PropellerPitch

RPM_05_Status

RSA_01_StarboardRudderSensor

RSA_02_StatusRudderSensor)

RSA_03_PortRudderSensor

RSA_04_StatusPortRudderSensor)

RSD_01_Origin1Range

RSD_02_Origin1Bearing

RSD_03_VariableRangeMarker1

RSD_04_BearingLine1

RSD_05_Origin2Range

RSD_06_Origin2Bearing

RSD_07_VRM2

RSD_08_EBL2

RSD_09_CursorRange

RSD_10_CursorBearing

RSD_11_RangeScale

RSD_12_RangeScaleUnits

RSD_13_DisplayRotation

RTE_01_TotalNumberOfMessages

RTE_02_MessageNumber

RTE_03_MessageMode

RTE_04_RouteIdentifier

RTE_05_WaypointIdentifier1

RTE_06_WaypointIdentifier2

RTE_07_WaypointIdentifier3

RTE_08_WaypointIdentifier4

RTE_09_WaypointIdentifier5

RTE_10_WaypointIdentifier6

RTE_11_WaypointIdentifier7

RTE_12_WaypointIdentifier8

RTE_13_WaypointIdentifier9

RTE_14_WaypointIdentifier10

SFI_01_TotalNumberOfMessages

SFI_02_MessageNumber

SFI_03_1stFrequency

SFI_04_1stMode

SFI_05_2ndFrequency

SFI_06_2ndMode

SFI_07_3rdFrequency

SFI_08_3rdMode

SFI_09_4thFrequency

SFI_10_4thMode

SFI_11_5thFrequency

SFI_12_5thMode

SFI_13_6thFrequency

SFI_14_6thMode

STN_01_TalkerID

TLB_01_TargetNumber

TLB_02_LabelAssigned

TLB_03_TargetNumber1

TLB_04_LabelAssigned1

TLB_05_TargetNumber2

TLB_06_LabelAssigned2

TLB_07_TargetNumber3

TLB_08_LabelAssigned3

TLB_09_TargetNumber4

TLB_10_LabelAssigned4

TLB_11_TargetNumber5

TLB_12_Labelassigned5

TLB_13_TargetNumber6

TLB_14_LabelAssigned6

TLB_15_TargetNumber7

TLB_16_LabelAassigned7

TLB_17_TargetNumber8

TLB_18_LabelAssigned8

TLB_19_TargetNumberReported

TLB_20_TargetLabelAssigned

TLL_01_TargetNumber

TLL_02_TargetLatitude

TLL_03_TargetLatitudeInd

TLL_04_TargetLongitude

TLL_05_TargetLongitudeInd

TLL_06_TargetName

TLL_07_UTC

TLL_08_TargetStatus

TLL_09_ReferenceTarget

TTM_01_TargetNumber

TTM_02_TargetDistance

TTM_03_Bearing

TTM_04_BearingInd

TTM_05_TargetSpeed

TTM_06_TargetCourse

TTM_07_TargetCourseInd

TTM_08_DistanceOfClosestPoint

TTM_09_TimeToCPA

TTM_10_SpeedAndDistanceUnits

TTM_11_TargetName

TTM_12_TargetStatus

TTM_13_ReferenceTarget

TTM_14_UTC

TTM_15_TypeOfAcquisition

TXT_01_TotalNumberOfMessages

TXT_02_MessageNumber

TXT_03_TextIdentifier

TXT_04_TextMessage

VBW_01_LongitudinalWaterSpeed

VBW_02_TransverseWaterSpeed

VBW_03_StatusWaterSpeed

VBW_04_LongitudinalGroundSpeed

VBW_05_TransverseGroundSpeed

VBW_06_StatusGroundSpeed

VBW_07_SternTransverseWaterSpeed

VBW_08_StatusSternWaterSpeed

VBW_09_SternTransverseGroundSpeed

VBW_10_StatusSternGroundSpeed

VDR_01_Direction

VDR_02_DirectionInd

VDR_03_DirectionMagnetic

VDR_04_DirectionMagneticInd

VDR_05_CurrentSpeed

VDR_06_CurrentspeedInd

VHW_01_Heading

VHW_02_HeadingInd

VHW_03_HeadingMagnetic

VHW_04_HeadingMagneticInd

VHW_05_SpeedKnots

VHW_06_SpeedKnotsInd

VHW_07_SpeedKmh

VHW_08_SpeedKmhInd

VLW_01_TotalCumulativeDistance

VLW_02_TotalCumulativeDistanceInd

VLW_03_DistanceSinceReset

VLW_04_DistanceSinceResetInd

VPW_01_SpeedKnots

VPW_02_SpeedKnotsInd)

VPW_03_SpeedMs

VPW_04_SpeedMsInd

VTG_01_CourseOverGround

VTG_02_CourseOverGroundInd

VTG_03_CourseOverGroundMagnetic

VTG_04_CourseOverGroundMagneticInd

VTG_05_SpeedOverGroundKnots

VTG_06_SpeedOverGroundKnotsInd

VTG_07_SpeedOverGroundKmh

VTG_08_SpeedOverGroundKmhInd

VTG_09_ModeIndicator

VWR_01_MeasuredWindAngle

VWR_02_VesselHeading

VWR_03_MeasuredWindSpeed

VWR_04_MeasuredWindSpeedInd

VWR_05_WindSpeedMeters

VWR_06_WindSpeedMetersInd

VWR_07_WindSpeedKmh

VWR_08_WindSpeedKmhInd

VWT_01_CalculatedWindAngle

VWT_02_VesselHeading

VWT_03_CalculatedWindSpeed

VWT_04_CalculatedWindSpeedInd

VWT_05_WindSpeedMeters

VWT_06_WindSpeedMetersInd

VWT_07_WindSpeedKmh

VWT_08_WindSpeedKmhInd

WCV_01_VelocityComponent

WCV_02_VelocityComponentInd

WCV_03_WaypointIdentifier

WCV_04_ModeIndicator

WNC_01_DistanceMiles

WNC_02_DistanceMilesInd

WNC_03_DistanceKm

WNC_04_DisttanceKmInd

WNC_05_WaypointIdentifierFrom

WNC_06_WaypointIdentifierTo

WPL_01_WaypointLatitude

WPL_02_WaypointLatitudeInd

WPL_03_WaypointLongitude

WPL_04_WaypointLongitudeInd

WPL_05_WaypointIdentifier

XDR_01_Transducer1Type

XDR_02_Measurmnt1Data

XDR_03_UnitsOfMeasure1

XDR_04_Transducer1

XDR_05_Transducer2Type

XDR_06_Measurment2Data

XDR_07_UnitsOfMeasure2

XDR_08_Transducer2

XDR_09_Transducer3Type

XDR_10_Measurment3Data

XDR_11_UnitsOfMeasure3

XDR_12_Transducer3

XDR_13_Transducer4Type

XDR_14_Measurment4Data

XDR_15_UnitsOfMeasure4

XDR_16_Transducer4

XDR_17_Transducer5Type

XDR_18_Measurment5Data

XDR_19_UnitsOfMeasure5

XDR_20_Transducer5

XDR_21_Transducer6Type

XDR_22_Measurment6Data

XDR_23_UnitsOfMeasure6

XDR_24_Transducer6

XDR_25_Transducer7Type

XDR_26_Measurment7Data

XDR_27_UnitsOfMeasure7

XDR_28_Transducer7

XDR_29_Transducer8Type

XDR_30_Measurment8Data

XDR_31_UnitsOfMeasure8

XDR_32_Transducer8

XTE_01_Status1

XTE_02_Status2

XTE_03_MagnitudeOfCrossTrackError

XTE_04_DirectionToSteer

XTE_05_Units

XTE_06_ModeIndicator

XTR_01_MagnitudeOfCrossTrackError

XTR_02_DirectionToSteer

XTR_03_Units

ZDA_01_UTC

ZDA_02_Day

ZDA_03_Month

ZDA_04_Year

ZDA_05_LocalZoneHours

ZDA_06_LocalZoneMinutes

ZDL_01_TimeToPoint

ZDL_02_DistanceToPoint

ZDL_03_TypeOfPoint

ZFO_01_UTC

ZFO_02_ElapsedTime

ZFO_03_OriginWaypointID

ZTG_01_UTC

ZTG_02_TimeToGo

ZTG_03_DestinationWaypointID

NMEA 2000

NMEA 2000 (IEC 61162-3) describes a low cost, moderate capacity, bi-directional multi-transmitter/multi-receiver instrument network to interconnect marine electronic devices.

This implementation of the protocol allows connection of multiple NMEA 2000-capable devices. To setup your NMEA 2000 device and to prepare the information needed for the communication driver, please refer to the documentation you have received with the device.

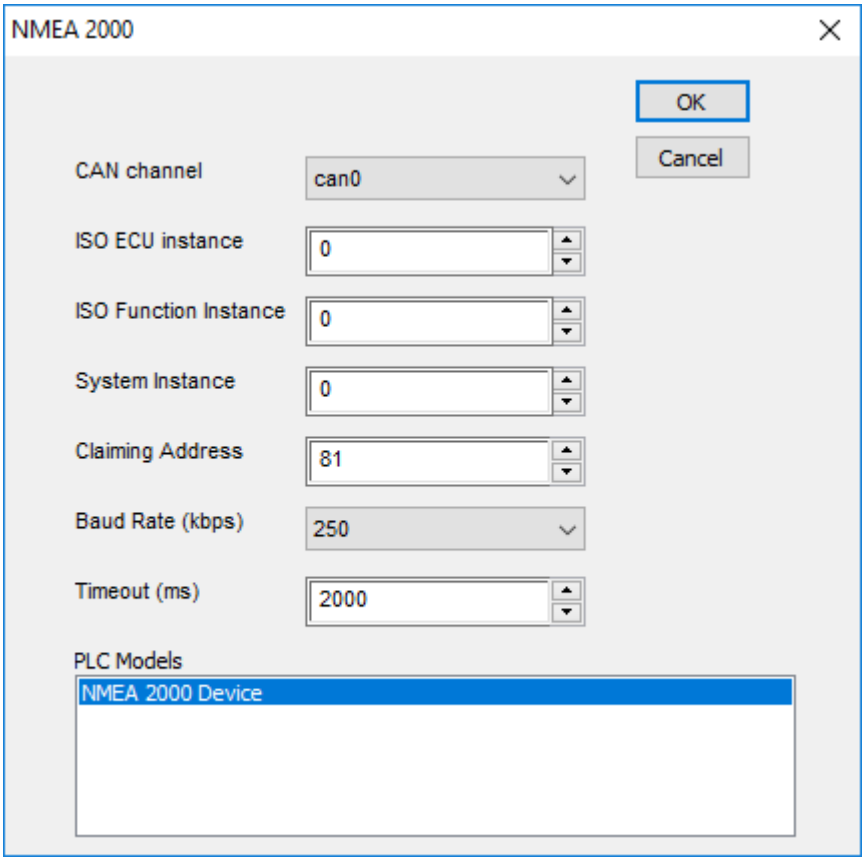
This communication driver identifies directly NMEA 2000 messages on the bus using the PGN number. You should keep track of the PGN generated by the equipment you want to communicate because you will need them later in the set-up phase of the user interface application.

NMEA 2000 is based on CAN bus; CAN interface is required for operation.

Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol “NMEA 2000” from the list of available protocols.

The driver configuration dialog is shown in figure.



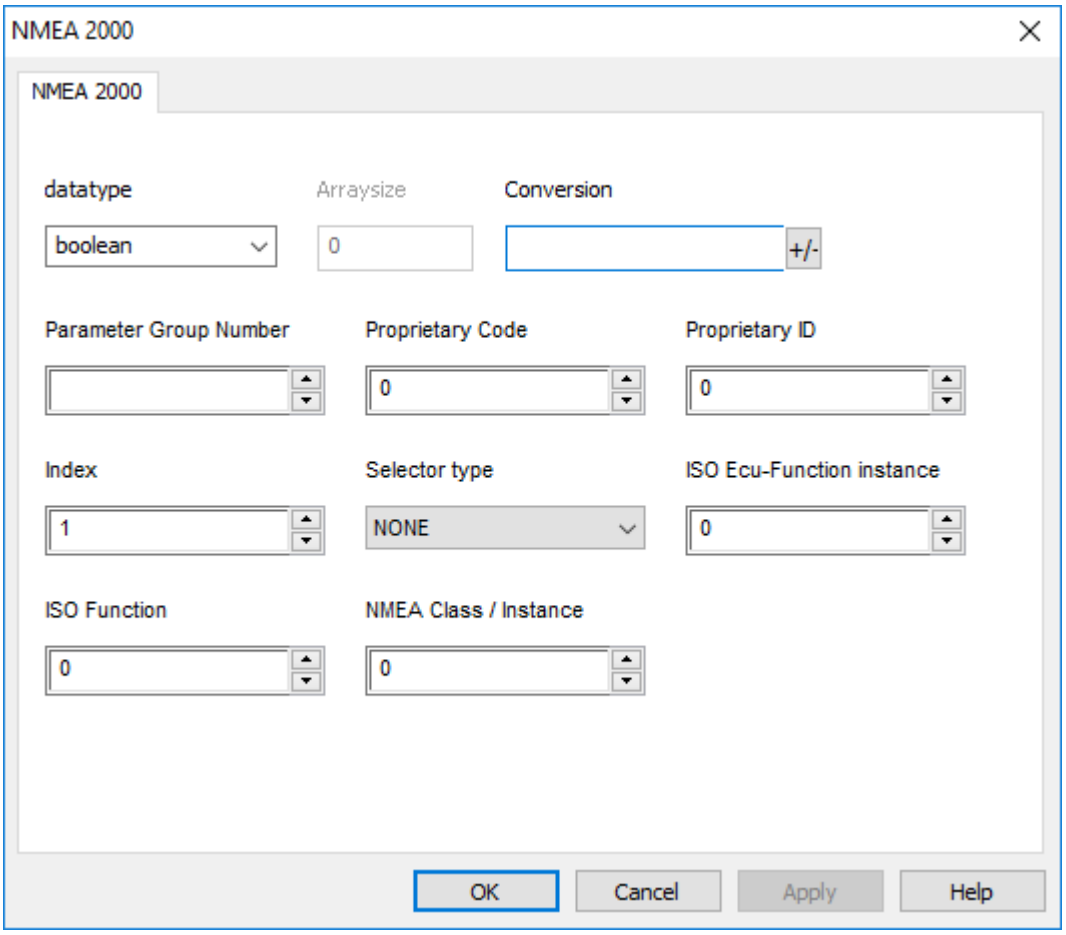
Element	Description
CAN channel	Select CAN port for use with this protocol.
ISO ECU	According to NMEA 2000 standard this number (0 to 7) specifies which of a group of

Element	Description
instance	ECUs associated with a given function is referenced.
ISO Function instance	According to NMEA 2000 standard this number (0 to 31) specifies the occurrence of a function on a particular device system of a network.
System Instance	According to NMEA 2000 standard this number is a 4-bit field with a valid range from 0 to 15 that indicates the occurrence of devices in additional network segments, redundant or parallel networks, or sub networks.
Claiming Address	Specifies the starting value for Address Claiming algorithm.
Baud Rate (kbps)	Even if NMEA 2000 define 250kpbs as standard, it is possible to modify the CAN bus baud rate.
Timeout (ms)	NMEA 2000 reading timeout expressed in milliseconds.  Note: if the sensor sends data every 2.5 seconds (2500), this parameter must be set to a value greater than 2500 (example: 3000).
PLC Models	Available models are: • NMEA 2000 Device

The parameters ISO ECU instance, ISO Function Instance and System Instance are also programmable on the field, according to NMEA2000 Standard V2.1, using PGN 126208. The values eventually modified on the field are saved locally by the driver and reloaded at the next start-up.

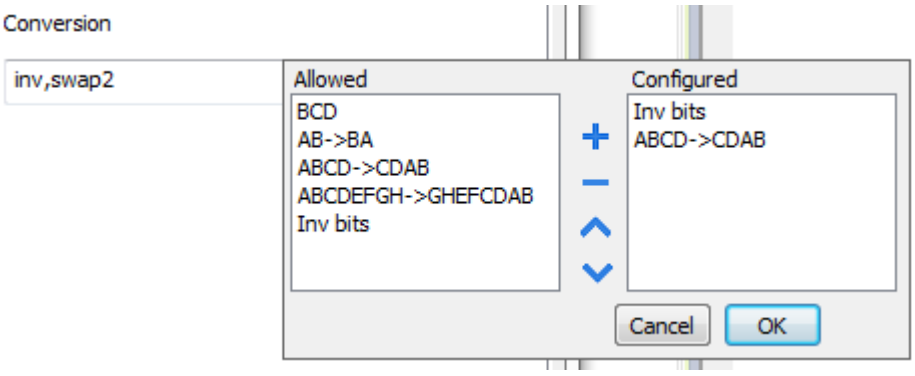
Tag Editor Settings

In Tag Editor select the protocol “NMEA 2000” from the list of protocols and add a new tag using [+] button.


NMEA 2000
NMEA 2000
datatype
Arraysize
Conversion
boolean
0
+/-
Parameter Group Number
Proprietary Code
Proprietary ID
0
0
Index
Selector type
ISO Ecu-Function instance
1
NONE
0
ISO Function
NMEA Class / Instance
0
0
OK
Cancel
Apply
Help

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 01 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.
Parameter Group Number	NMEA2000 Parameter Group Number code. Refer to standard NMEA tables.
Proprietary Code	Proprietary Code value. Refer to standard NMEA tables.
Proprietary ID	Proprietary ID value. Refer to standard NMEA tables.
Index	Indicates the index of the field in PGN data to point to.
Selector Type	When adding tags it can be necessary to duplicate them to read data coming from several devices generating same physical quantity or to select several instances of same PGN. In this case the Selector of the tag must be edited. In case of duplication of the tag, the selection of incoming data can be done using one of following methods: **NONE** Selector Type not selected **INSTANCE** Uses a defined bitfield value in data of PGN to distinguish between the possible sources. The value of received bitfield is compared with **NMEA Class / Instance** parameter for matching. **DEVICE** Uses a combination of parameters to select one of the possible devices sending the PGN. Each device is identified using its ECU instance number its function its Function Instance number and its NNEA class number. **DEV&INST** It is a combination of both DEVICE and INSTANCE selectors. It will allow to distinguish between several PGN containing different instance numbers and sent by several devices. Uses a combination of parameters to select one of the possible devices sending the PGN.
ISO Ecu-Function Instance	ISO Ecu-Function instance parameter used with Selector Type **DEVICE** or **DEV&INST**. It must contain (Function instance * 8 + ECU instance) of the selected device.

Element	Description
ISO Function	ISO Function parameter used with Selector Type **DEVICE** or **DEV&INST**. It must contain the function of the selected device.
NMEA Class / Instance	NMEA Class / Instance parameter. With selector **INSTANCE** It must contain the instance of the PGN. With selector **DEVICE** it must contain the NMEA class of the device. With selector **DEV&INST** it must contain (instance * 256 + NMEA class).

Definition of List of PGN Available in the System

The system can operate on a limited subset of PGNs defined by an additional file named “pgnTable.csv”.

Note: the file pgnTable.csv is mandatory and used by the protocol and it will be automatically downloaded to the runtime. The file must be located in C:\.....\target\protocols\.

The file can be created or edited using any text editor and it contains a list of standard or proprietary PGNs that the system can manage and the specification of each field in the PGN.

Each element is composed by a line starting with “PGN:” the pgnTable is composed by two parts,

List of elements and list of fields

The meaning of each parameter is described below.

Parameter	Description
PGN Name	name assigned to the PGN by NMEA 2000 standard or by the manufacturer
PGN number	the PGN number (decimal) you can find in standard
ISO/NMEA type	indicates if PGN is following NMEA 2000 standard or other ISO standards such as J1939
SINGLE/MULTIPLE	indicated if the PGN is SINGLE frame or MULTIPLE frame type
DefaultPriority	indicates the default priority in case of transmission (not used for receive only PGN)
DefaultRate	indicates the default transmission rate in case of transmission (not used for receive only PGN)
ProprietaryCode	in case of proprietary PGN indicates the proprietary code identifying the manufacturer
Proprietary ID Field	in case of proprietary PGN it indicates the field used to select the meaning of the PGN
Proprietary ID Value	in case proprietary PGN it indicates the value of the ID identifying the PGN

Parameter	Description
InstanceIndex	Specify which of the fields in the PGN is containing the NMEA 2000 instance. Value = 0 means no instance
INPUT/OUTPUT	Indicates the direction of the PGN. For PGNs defined for reading and reply use INPUT. For PGN defined for generation use OUTPUT

Example:

```
PGN:, Wind Data, 130306, NMEA, SINGLE, 2, 100, 0, 0, 0, 0, INPUT
```

The meaning of each field parameter, including reserved bit fields, is described below.

Field	Description
FieldName	standard name assigned to the field by NMEA 2000 standard or by the manufacturer
FieldBitSize	dimension of data in bits
Field DataFormat	numerical conversion following NMEA 2000 standard (DFxx) This data is always available in PGN description or must be specified by manufacturer in case of proprietary PGNs
READ_ ONLY/REPLY	Indicates if the field can be written by the application. In such case the PGN (if received almost once) will be retransmitted embedding the new data

Examples:

```
PGN:, Wind Data, 130306, NMEA, SINGLE, 2, 100, 0, 0, 0, 0, INPUT
```

1, SequenceID, 8, DF53, READ_ONLY

2, Wind Speed, 16, DF35, READ_ONLY

3, Wind Direction, 16, DF02, READ_ONLY

4, Wind Reference, 3, DF52, READ_ONLY

5, reserved, 21, DF52, READ_ONLY

```
PGN:, Rudder, 127245, NMEA, SINGLE, 2, 100, 0, 0, 0, 1, INPUT
```

1, Rudder Instance, 8, DF52, READ_ONLY

2, Direction Order,3, DF52, REPLY

3, reserved,5, DF52, READ_ONLY

4, Angle Order,16, DF04, REPLY

5, Position,16, DF04, READ_ONLY

6, reserved,16, DF52, READ_ONLY

```
PGN:, Switch Bank Control, 127502, NMEA,  SINGLE, 3, 1000 , 0, 0, 0, 1, OUTPUT
```

1, Switch Bank Instance,8, DF53, READ_ONLY

2, Switch 01,2, DF52,READ_ONLY

3, Switch 02,2, DF52,READ_ONLY

4, Switch 03,2, DF52,READ_ONLY

5, Switch 04,2, DF52,READ_ONLY

6, Switch 05,2, DF52,READ_ONLY

7, Switch 06, 2, DF52,READ_ONLY

……

29,Switch 28,2, DF52,READ_ONLY

Definition of Configuration Information (PGN 126998)

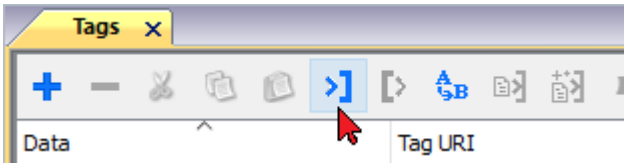
By default the driver answer to a request of PGN 126998 with a set of 3 empty strings. It is possible to customize the answer introducing the 3 strings (up to 70 chars each) containing:

- Installation description field1
- Installation description field2
- Manufacturer Information

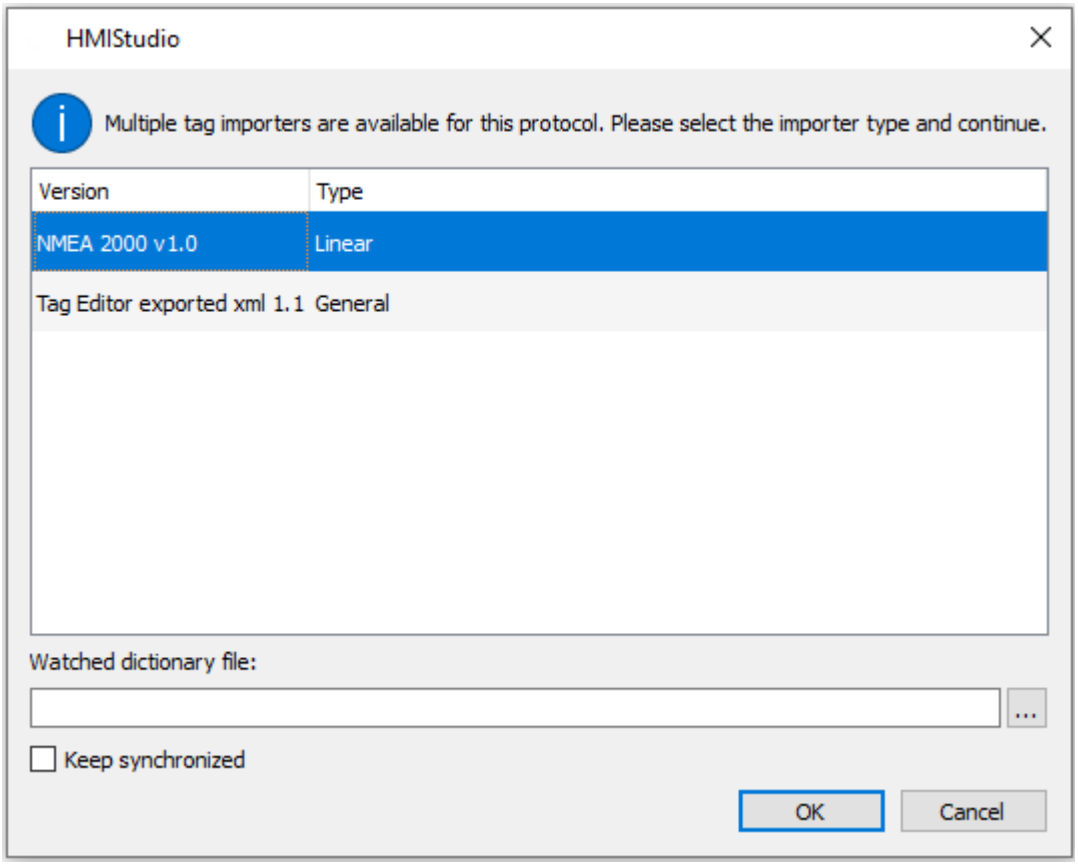
The three strings must be prepares into a text file named “confInfo.txt” (case sensitive) that can be manually deployed into unit in folder <runtime>/protocols.

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



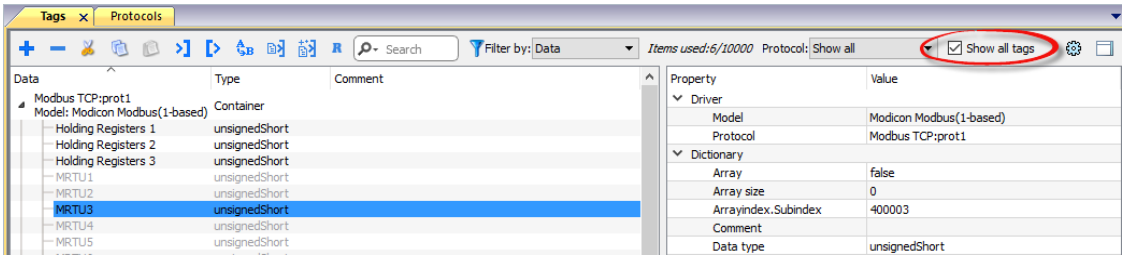
The following dialog shows which importer type can be selected.

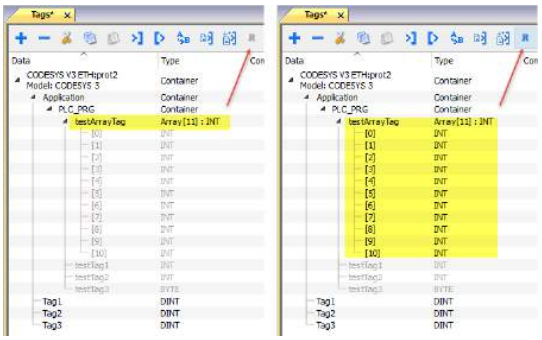
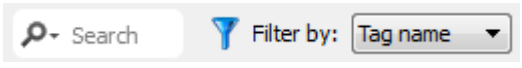


Importer	Description
NMEA 2000 v1.1 Linear	Requires a **.csv** file. All variables will be displayed at the same level. A NMEA2000 PGN Table is included into Studio distribution. It can be found in *<installation_folder>*/target/protocols/pgnTable.csv
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

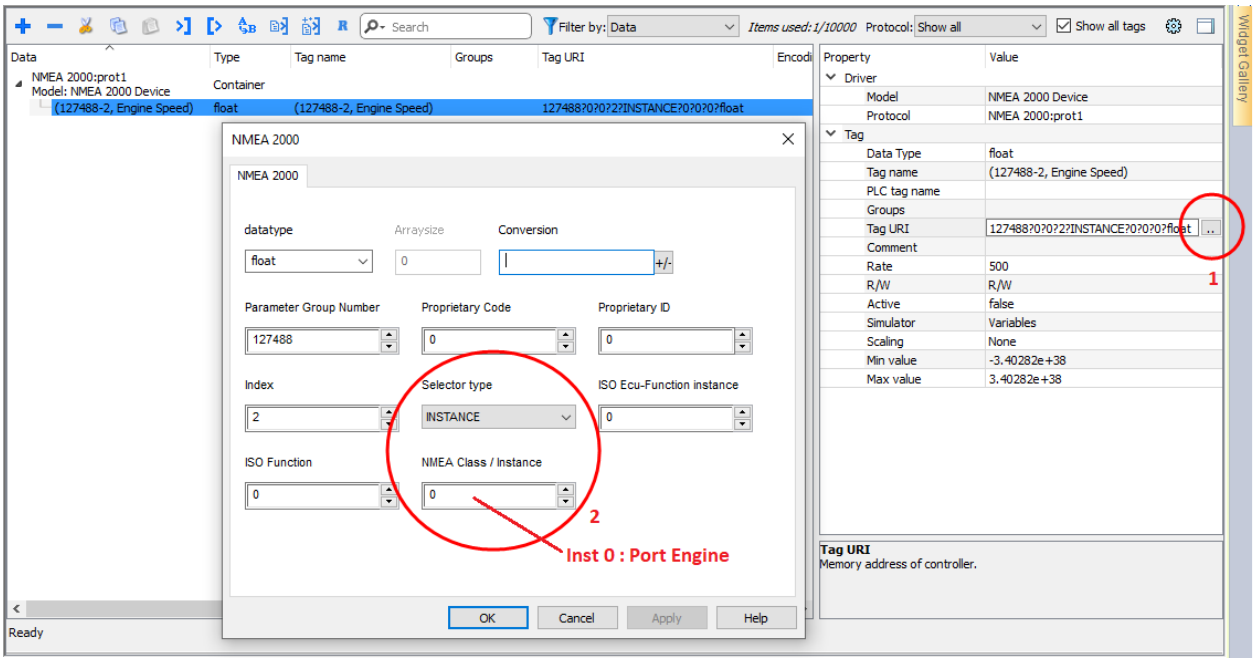
Instance Definition

If the selected PGN “supports” instances, you must modify the “**instance**” and **“selector type**” field in tag configuration.

For this Example we have defined 2 instances of “Engine Speed” for **Port Engine (Instance 0) /Starboard Engine (Instance 1)** as follow:

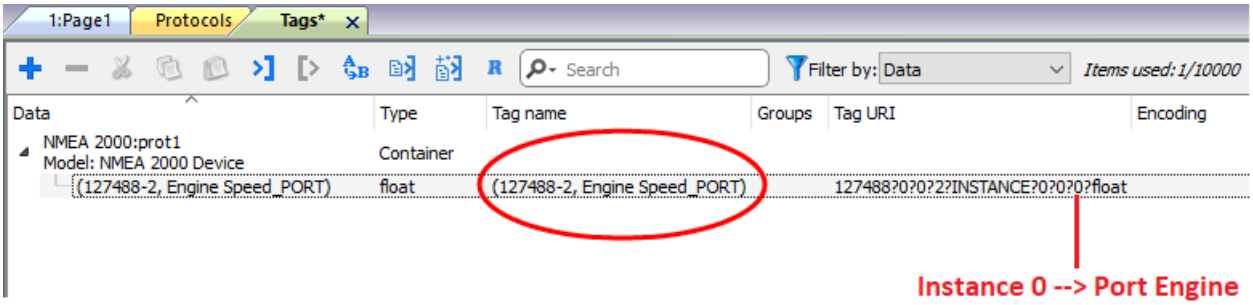
1) Open **Tags tab**

2) Open the editor for the already created Engine Speed Tag , then configure the tag propriety like is shown in the figure below

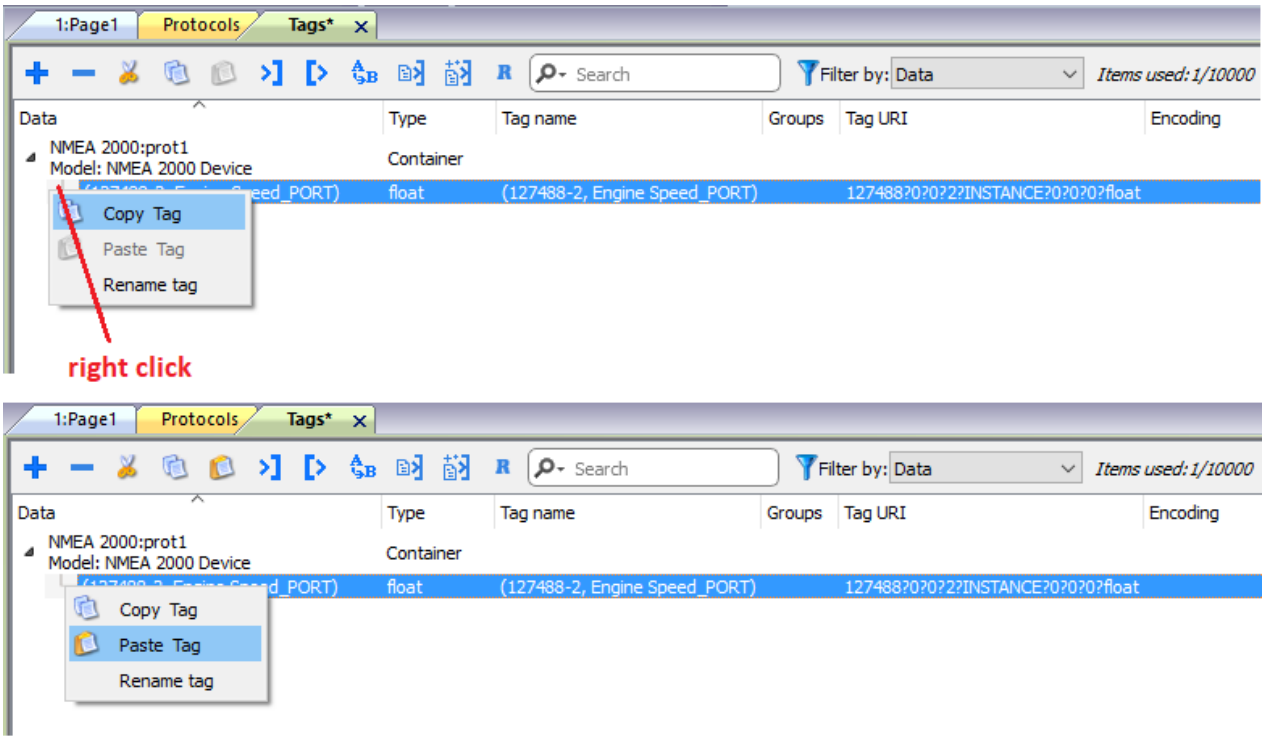


Selector Type: INSTANCE (this tag works with instances)
NMEA Class / Instance: Indicates the instance of this tag (for this example, 0 -> Port Engine)

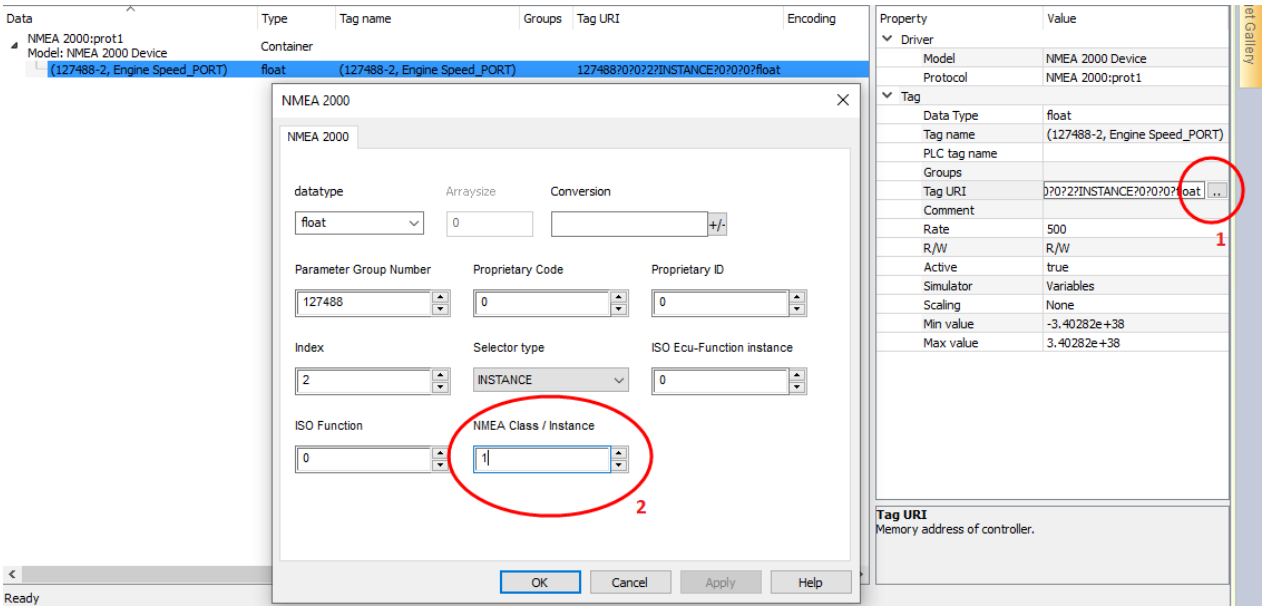
3) Press OK then rename the tag to remind which engine it refers:



4) Now you have to define the 2nd Engine Speed Instance for STARBOARD Engine (Instance 1).
Copy and paste the Port tag as shown in the figure below:

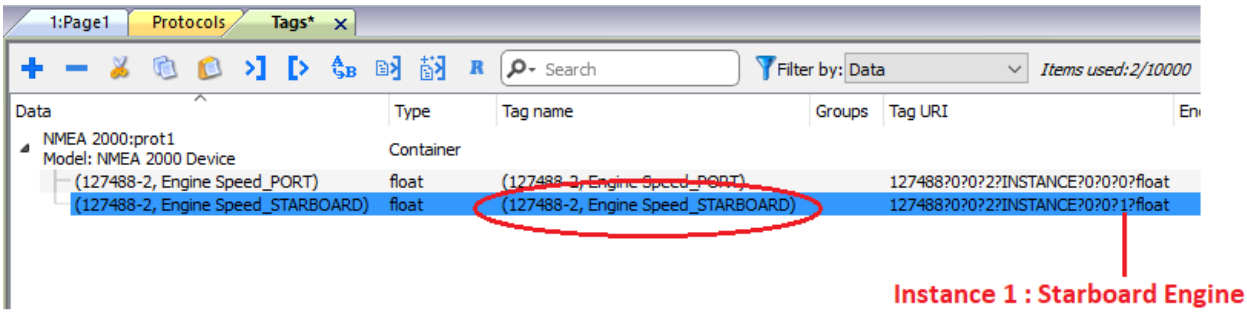


5) Open the editor for the copied Engine Speed Tag , then configure the tag propriety like is shown in the figure below:



NMEA Class / Instance: Indicates the instance of this tag (for this example, 1 -> Starboard Engine)

6) Press OK then rename the tag to remind which engine it refers:



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

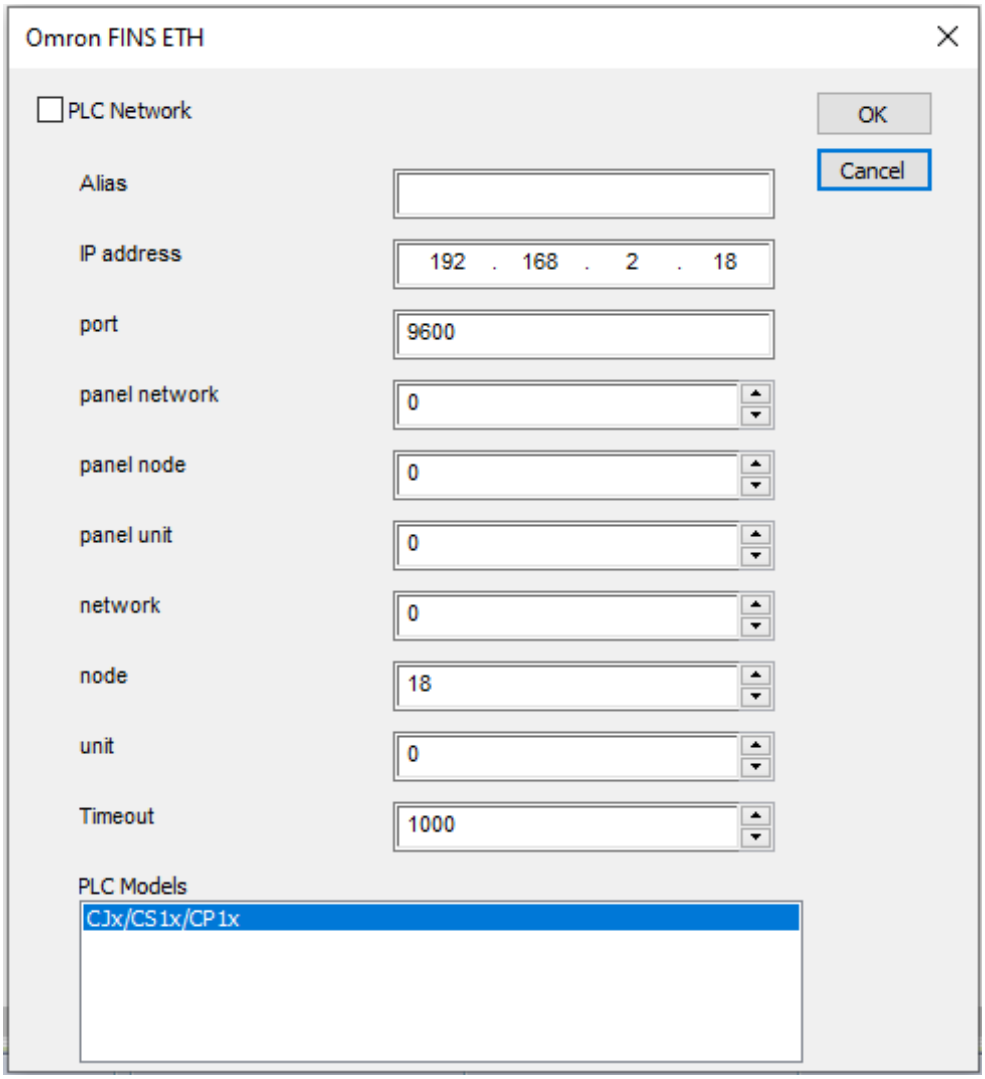
The codes supported for this communication driver are:

Error	Notes
the value never received	PGN of the Tag was never received
timeout on the value refresh	Tag data is too old (see timeout value in driver configuration)
Not byte boundary on dynamic field **Something wrong with the PGN** **Too many bits to use**	error in PGn definition

Omron FINS ETH

This driver supports the FINS protocol via Ethernet connection. For a list of models that support the FINS Communications Service, refer to the manufacturer's website.

Protocol Editor Settings

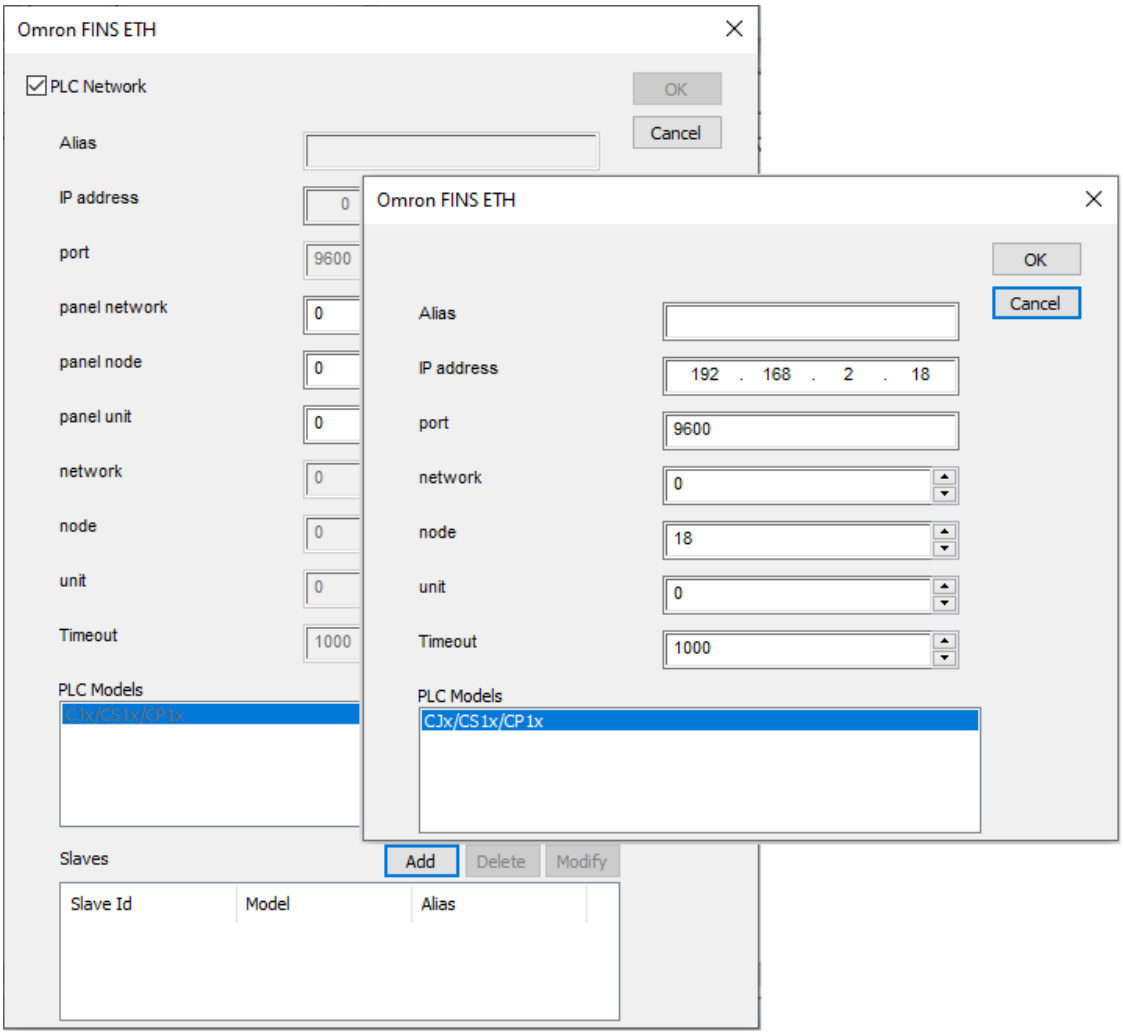


Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
IP address	The Ethernet IP address of the controller connected to the operator panel
Port	Defines the port number used in the communication with the PLC. The UDP Port number must match the value specified in the PLC configuration; the default value is 9600. Most

Element	Description
	applications will use the default value.
Network Node Unit	Parameters that define the FINS address of the device. There is a conversion rule to determine the IP address of a device starting from the FINS address in the Omron network. When using the FINS communication service, it is necessary to specify the node addressing according to the FINS addressing scheme. Even in this case, data must be sent and received on the Ethernet network using IP addresses. Therefore, IP addresses are converted from FINS addresses. There are three ways to convert the FINS addresses into the corresponding IP address; they are: • Automatic generation (default) • IP address table • Combined method (uses Automatic and IP address table) The Omron documentation contains all the details related to determine the IP address of the controller depending on the FINS address assigned to it. The next chapter shows an example of controller configuration based on IP address table.
Panel Network **Panel Node** **Panel Unit**	The Panel Network/Node/Unit parameters assigned to HMI should be compatible with the ones assigned in the Omron network to the PLC: • Network Number must match the one specified for the PLC • Node Number should match the last number of the IP address of the HMI; in the figure above the panel has been configured with IP address 192.168.2.15. • Unit represent the possible different network cards over the same node; for the HMI should be always set to zero since there is always only one communication unit.

The protocol supports the connections to multiple controllers.

To enable this, check the "PLC Network" check box and provide the configuration per each node.



Controller Settings

PLC must be properly configured to handle the communication with HMI.

Below an example of configuration based on a real scenario.

Configuration windows in this chapter are depending on PLC model.
Following lines must be used as guidelines for any specific configuration.

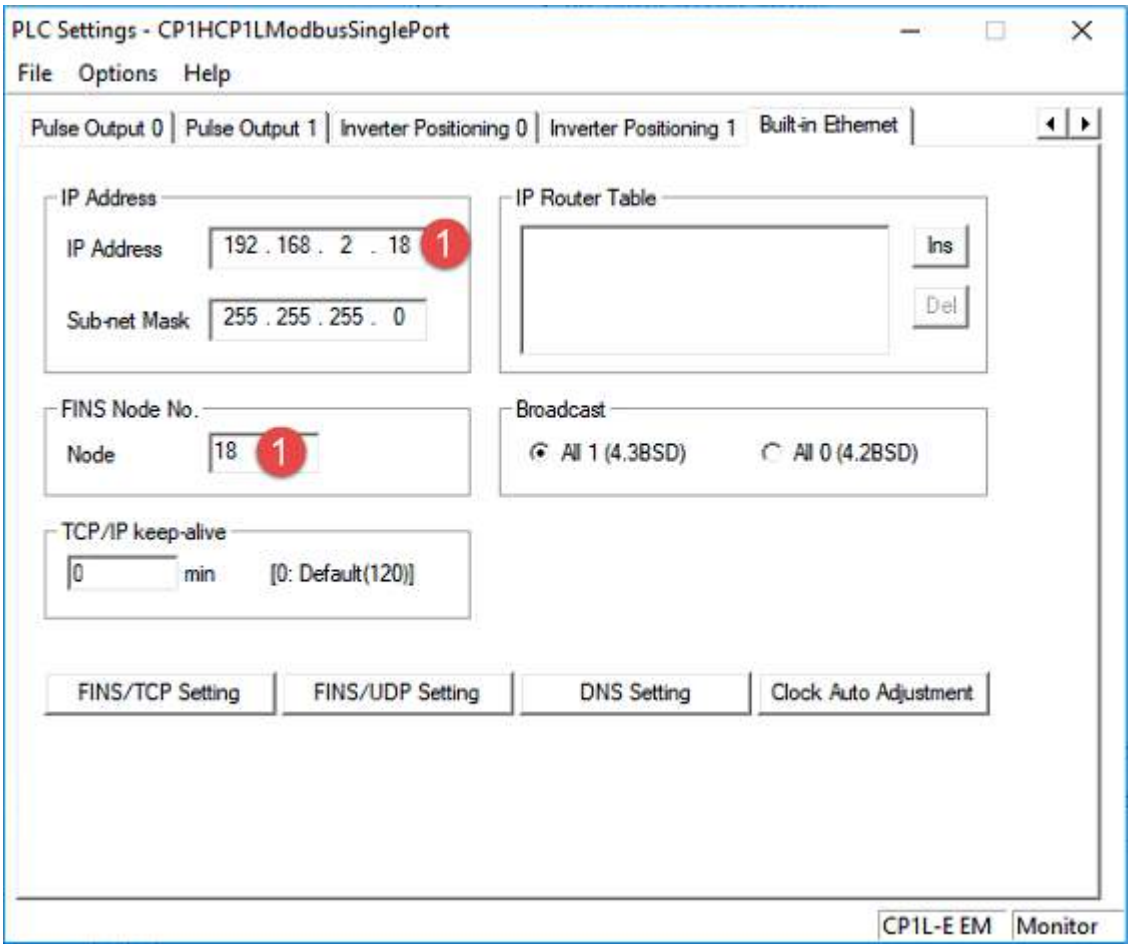
Example Setup

HMI IP address = 192.168.2.16

PLC IP address = 192.168.2.18

In Ethernet configuration Tab:

1. Make sure that last number of IP address is the same of FINS Node No.

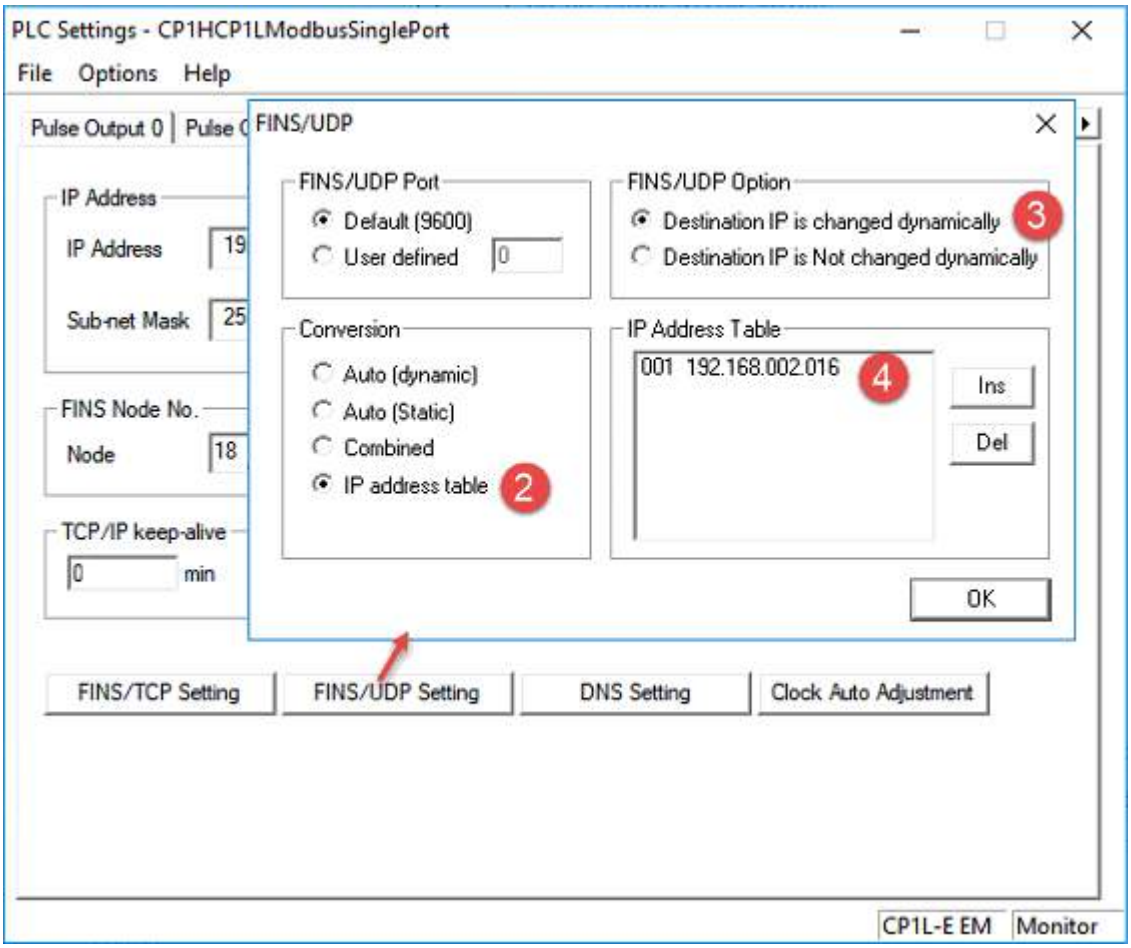


In FINS/UDP Setting

2. Set Conversion to "IP address table"

3. Set FINS/UDP Options to "Destination IP is changed dynamically"

4. Insert HMI IP address



IP Address Table can contain more than one address.
In these cases make sure that index of IP addresses is consecutive:
001 192.168.002.016
002 192.168.002.017
003 192.168.002.033

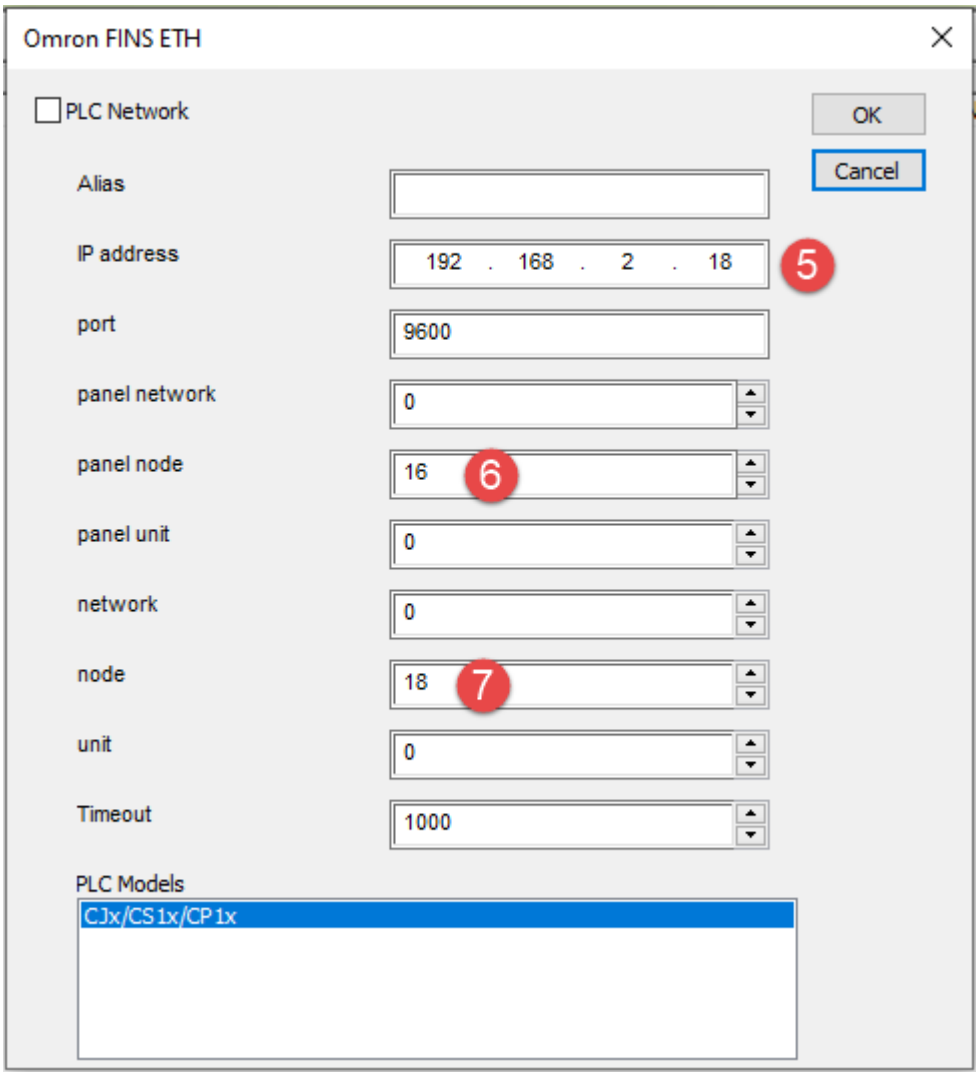
Add PC IP address in IP Address Table described above to allow communication between PLC and online Simulation.

In protocol editor

5. Set the IP address of PLC

6. Insert last number of HMI IP address in panel node parameter

7. Insert last number of PLC IP address in node parameter



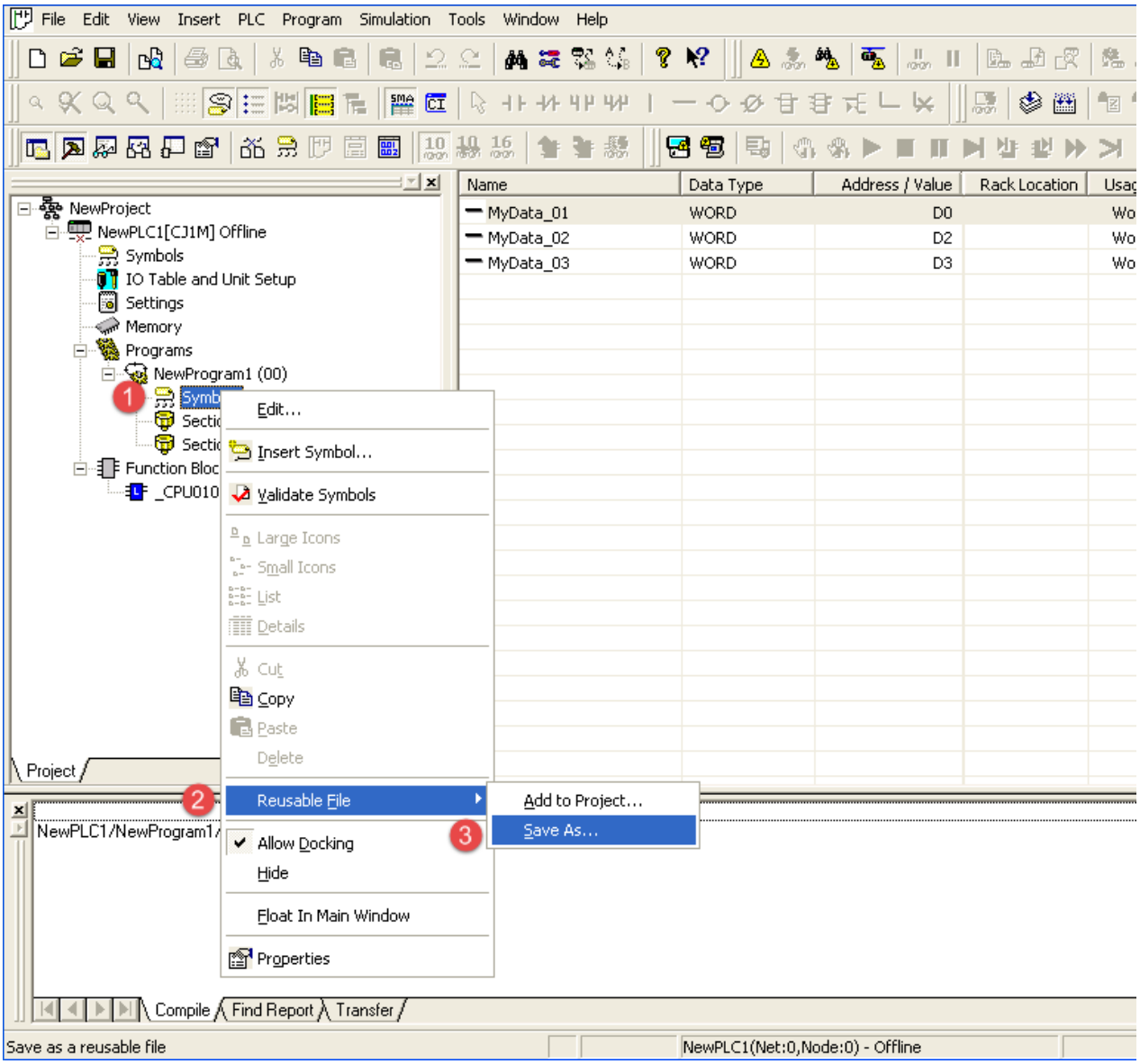
Tag Import

Exporting Tags from PLC

The Omron FINS Ethernet driver can import tag information from CX-Programmer PLC programming software. The tag import filter accepts symbol files with extension ".cxr" created by the Omron programming tool.

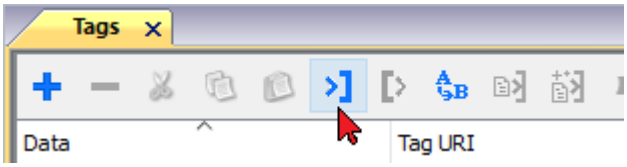
The ".cxr" files can be exported from the symbol table utility.

See in figure how to access the Symbol Table (if configured) from the Omron programming software.

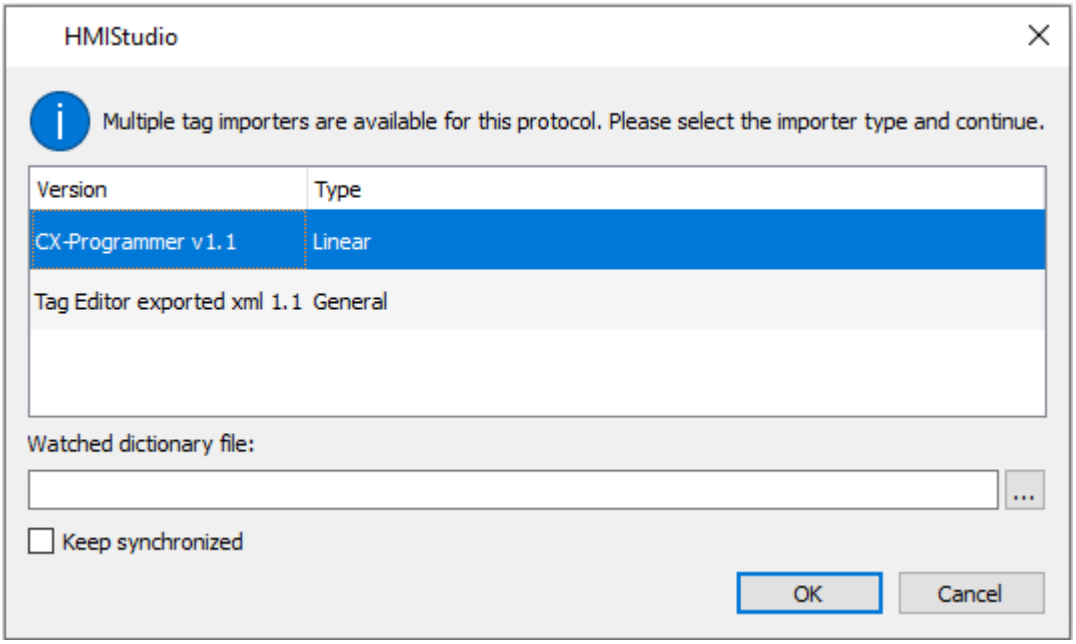


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



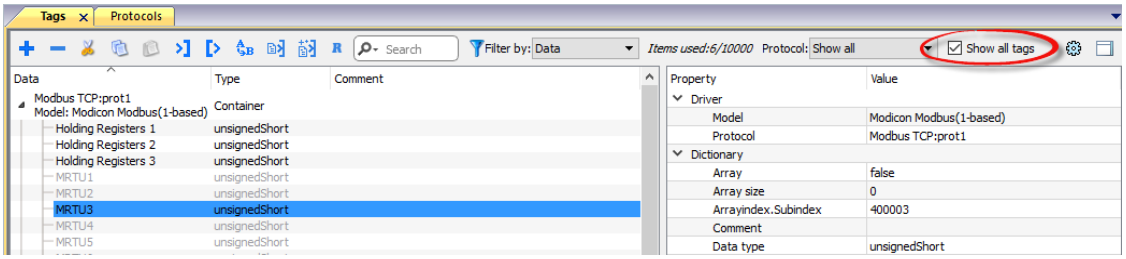
The following dialog shows which importer type can be selected.

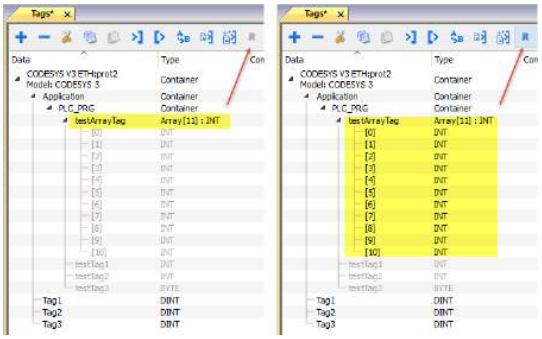
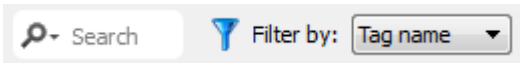


Importer	Description
CX-Programmer v1.1 Linear	Requires a **.cxr**file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

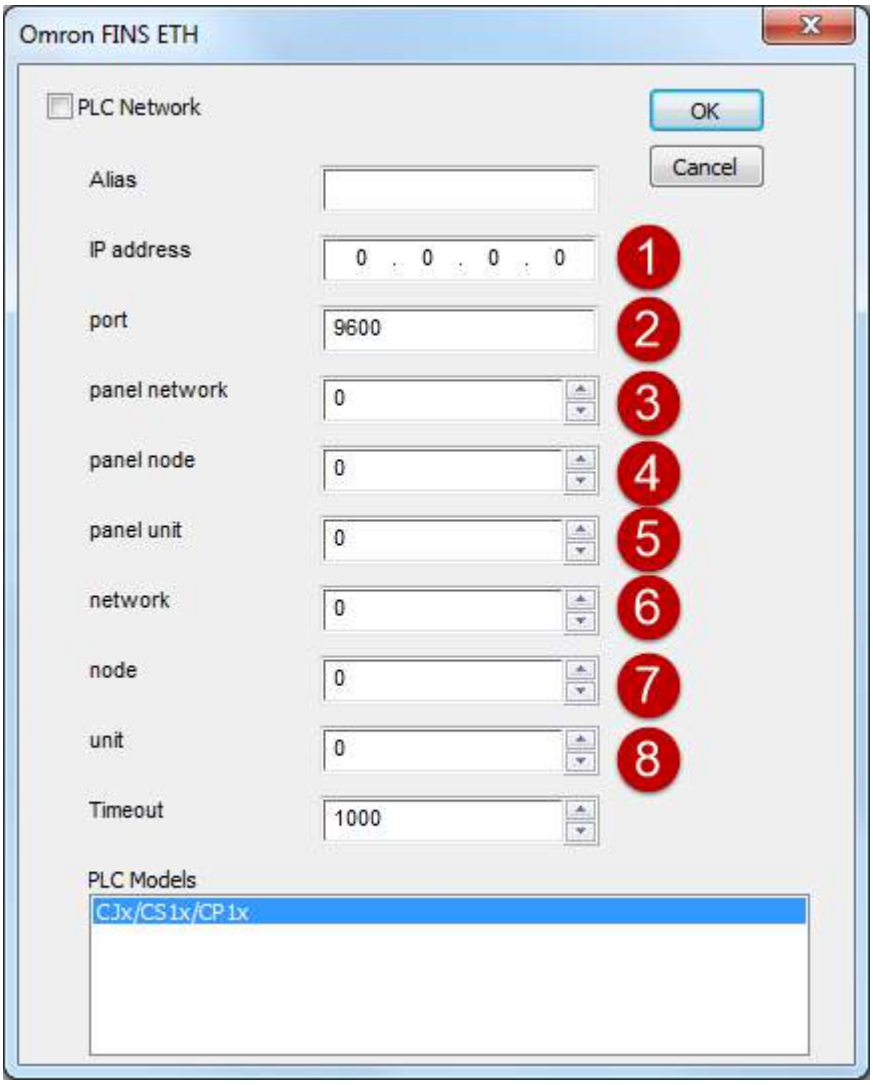
The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



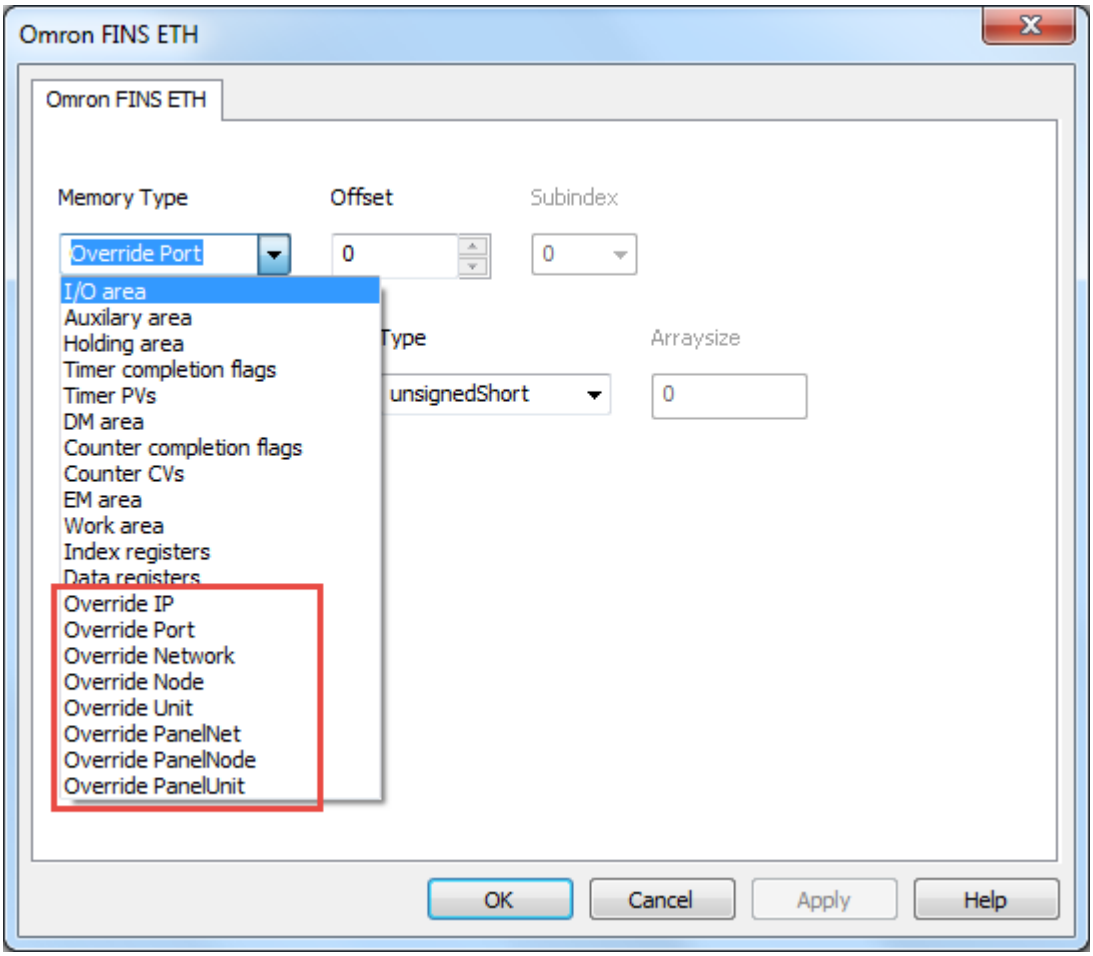
Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Override variables

The protocol provides the special data types to override the following protocol settings:



Tags can be created by manually add them from Tag Editor



Tag Name	Description
Override IP	Permits to override "IP address" property (1) in runtime. **Data type:** array unsigned bytes. Notes: - when address is set as "0.0.0.0" communication with the controller is stopped, no request frames are generated anymore. - when address is different than "0.0.0.0" it is interpreted as a real IP address to override and target PLC IP address is replaced in runtime with the new value.
Override Port	Permits to override "port" property (2) in runtime. **Data type:** unsignedShort.
Override Network	Permits to override "network" property (6) in runtime. **Data type:** unsignedByte.
Override Node	Permits to override "node" property (7) in runtime. **Data type:** unsignedByte.

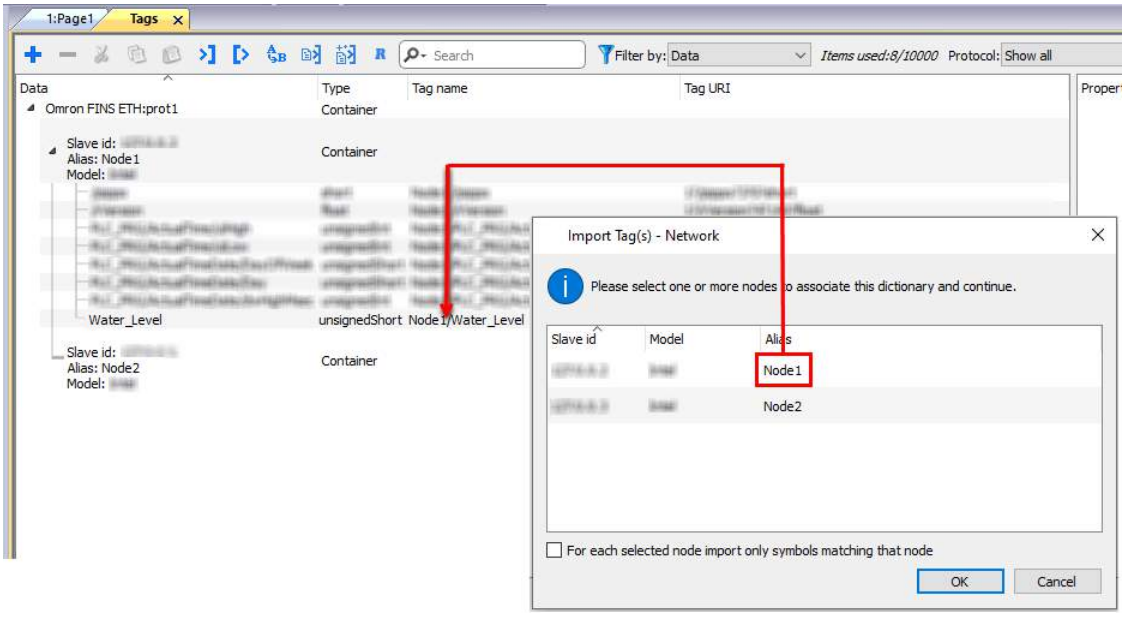
Tag Name	Description
Override Unit	Permits to override "unit" property (8) in runtime. **Data type:** unsignedByte.
Override PanelNet	Permits to override "panel network" property (3) in runtime. **Data type:** unsignedByte.
Override PanelNode	Permits to override "panel network" property (4) in runtime. **Data type:** unsignedByte.
Override PanelUnit	Permits to override "panel unit" property (5) in runtime. **Data type:** unsignedByte.

Note: Override Tags are initialized with the value of properties specified in Protocol Editor. Override values assigned at runtime are retained through power cycles.

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Note: aliasing tag names is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you

modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge; can be returned also in case the network/node/unit parameters contained in the PLC response are not matching with panel configuration
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources. The same error can be returned also in case the PLC could not complete the processing of the panel request and sent back to the panel and invalid/not completed response.
Cnt error	Returned when a specific control character in the protocol frame received does not match with the corresponding one in the request; verify the proper settings of the controller network configuration
General Error	Error cannot be identified; should never be reported; contact technical support

Omron FINS SER

This driver supports the FINS protocol via serial connetcion. For a list of models that support the FINS Communications Service, refer to the manufacturer's website.

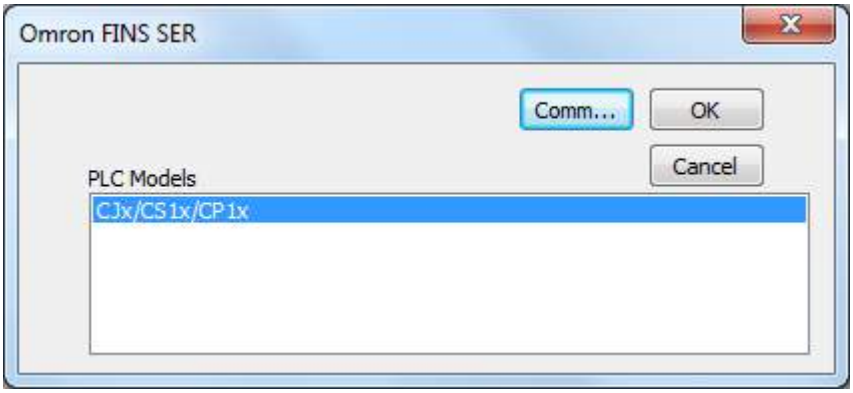
Protocol Editor Settings

Adding a protocol

To configure the protocol:

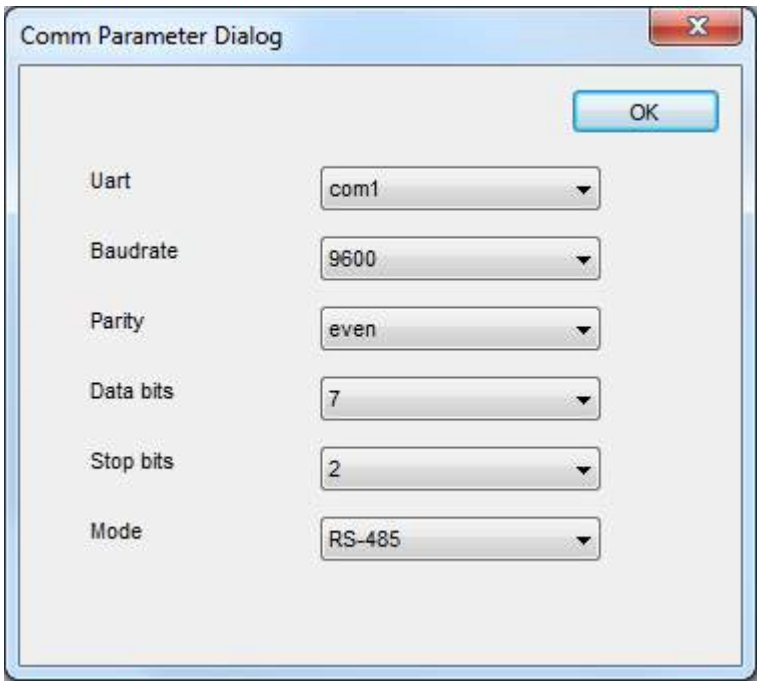
1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
PLC Models	PLC models available: • CJx/CSx/CP1x
Comm...	If clicked displays the communication parameters setup dialog.

Element	Description

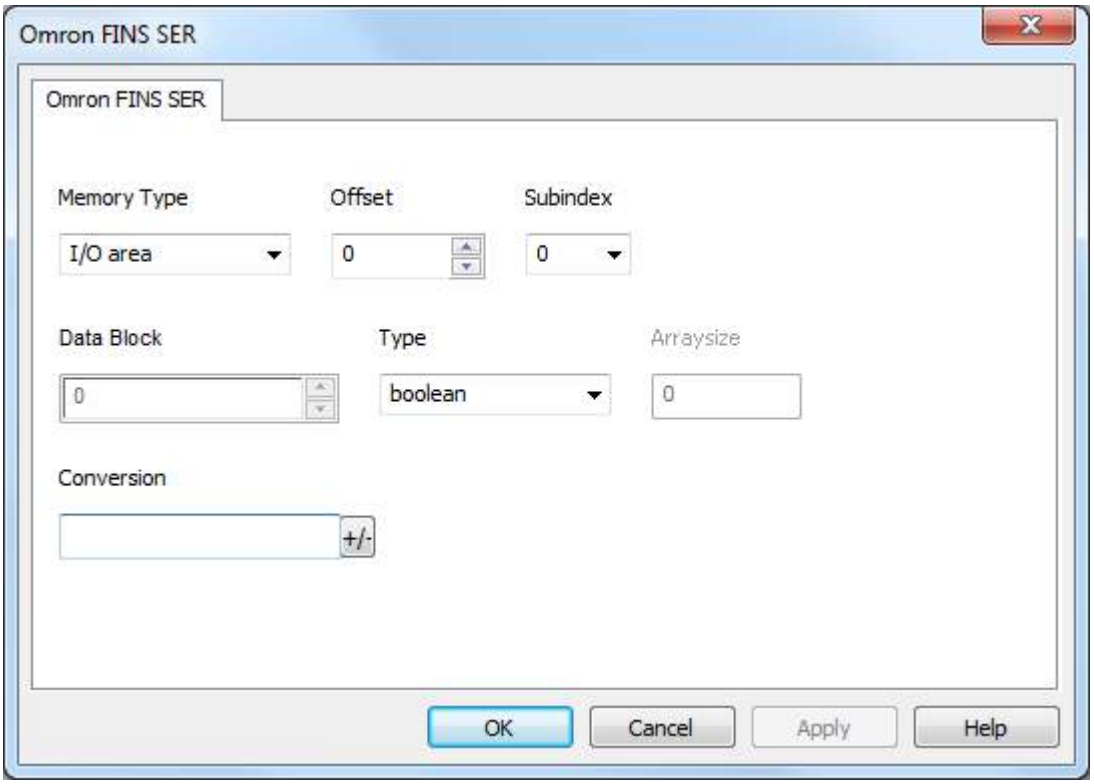


Element	Parameter
Port	Serial port selection. • **COM1**: device PLC port. • **COM2**: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board. • **COM3:** optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.
Mode	Serial port mode. Available modes: • **RS-232**. • **RS-485** (2 wires). • **RS-422** (4 wires).

Tag Editor Settings

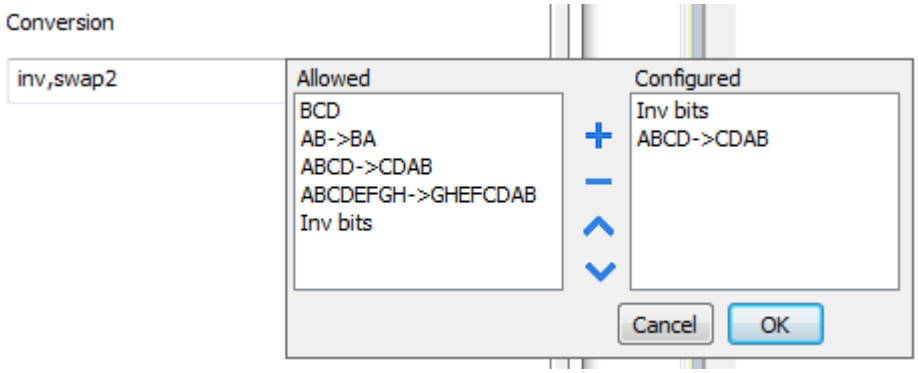
In Tag Editor select the protocol **Omron FINS SER**.

Add a tag using [+] button. Tag setting can be defined using the following dialog:



Element	Description
Memory Type	<table><tr><th>Memory Type</th><th>Description</th></tr><tr><td>**I/O area**</td><td>Corresponds to **CIO** resource on PLC</td></tr><tr><td>**Auxiliary area**</td><td>Corresponds to **A** resource on PLC</td></tr><tr><td>**Holding area**</td><td>Corresponds to **H** resource on PLC</td></tr><tr><td>**Timer completion flags**</td><td>Corresponds to **T** resource on PLC</td></tr><tr><td>**Timer PVs**</td><td>Corresponds to **TPV** resource on PLC</td></tr><tr><td>**DM area**</td><td>Corresponds to **D** resource on PLC</td></tr><tr><td>**Counter completion area**</td><td>Corresponds to **C** resource on PLC</td></tr><tr><td>**Counter CVs**</td><td>Corresponds to **CVS** resource on PLC</td></tr><tr><td>**EM area**</td><td>Corresponds to **E** resource on PLC</td></tr><tr><td>**Work area**</td><td>Corresponds to **W** resource on PLC</td></tr><tr><td>**Index registers**</td><td>Corresponds to **IR** resource on PLC</td></tr><tr><td>**Data registers**</td><td>Corresponds to **DR** resource on PLC</td></tr></table>
Offset	Starting address for the Tag. The possible range depend on memory type selected.

Element	Description
Subindex	This parameter allow to select a single part of the resource if the selected data type is shorter than the resource data type
Data block	Instance of resource of the PLC.
Data Type	Available data types: • **boolean** • **byte** • **short** • **int** • **unsignedByte** • **unsignedShort** • **unsignedInt** • **float** • **double** • **string** • **binary** See "Programming concepts" section in the main manual.  Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]…).

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH	**swap4**: Swap bytes in a double word.

Element	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>-> GHEFCDAB</td><td>Example: 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)</td></tr><tr><td>ABC...NOP -> OPM...DAB</td><td>swap8: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 01 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)</td></tr><tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9) Example: 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)</td></tr></table> Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

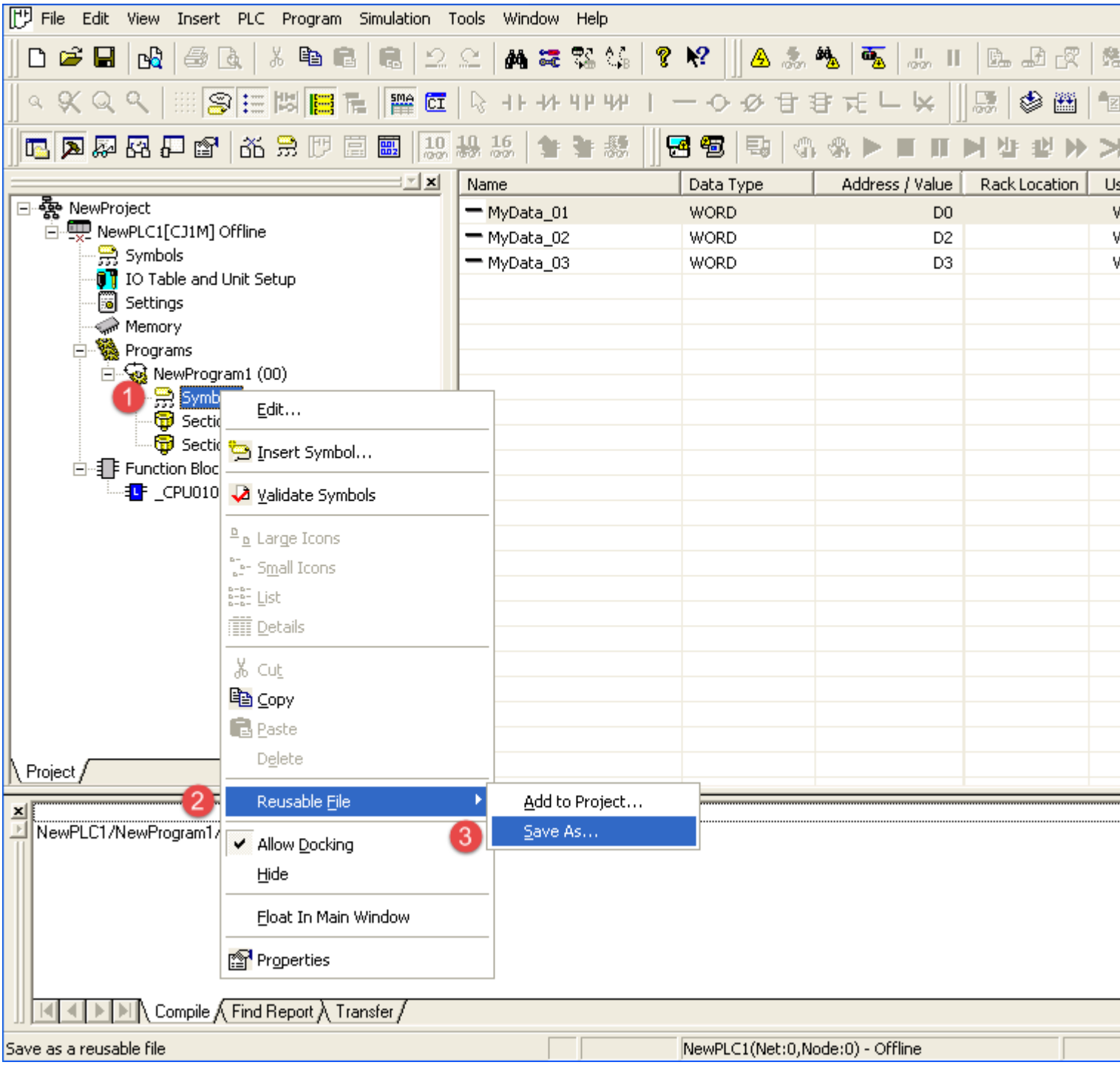
Tag Import

Exporting Tags from PLC

The Omron FINS SER driver can import tag information from CX-Programmer PLC programming software. The tag import filter accepts symbol files with extension ".cxr" created by the Omron programming tool.

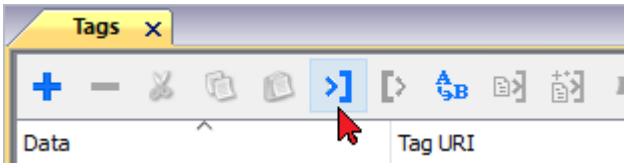
The ".cxr" files can be exported from the symbol table utility.

See in figure how to access the Symbol Table (if configured) from the Omron programming software.

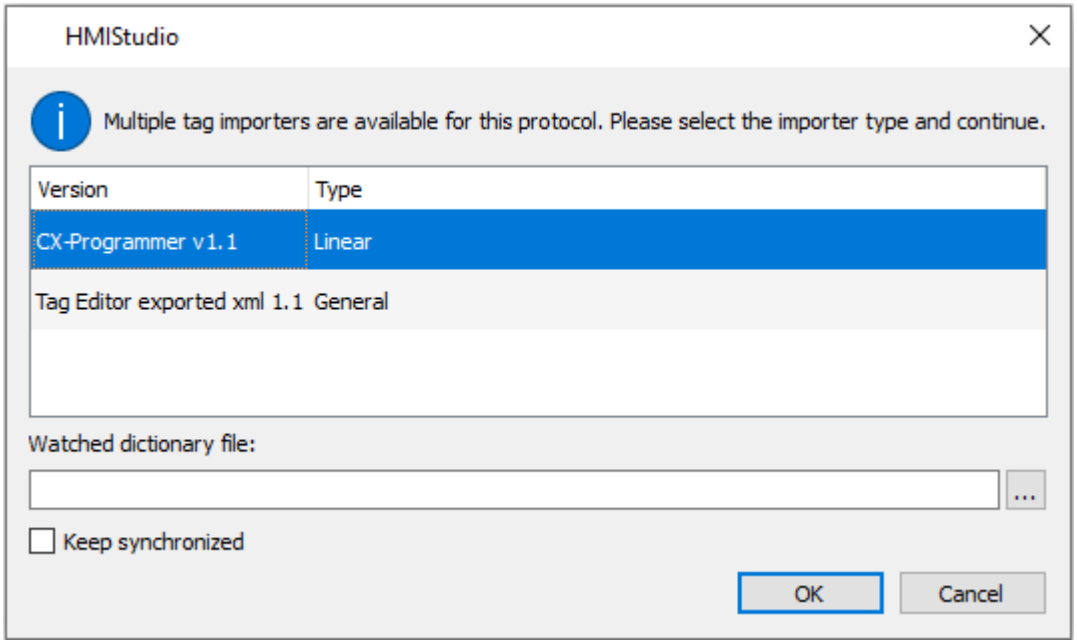


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



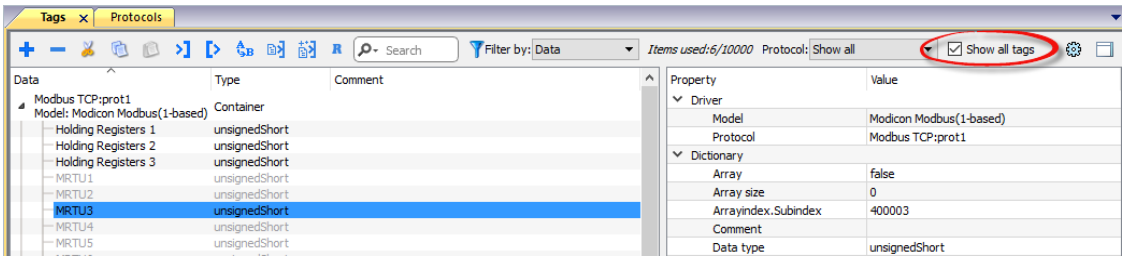
The following dialog shows which importer type can be selected.

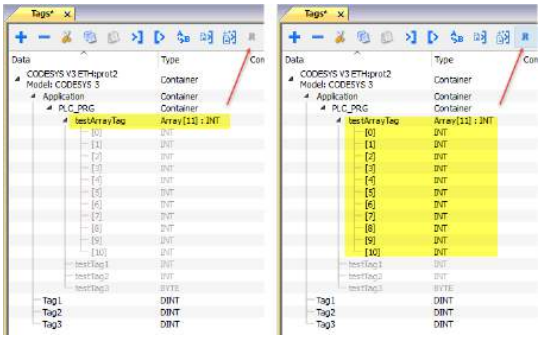
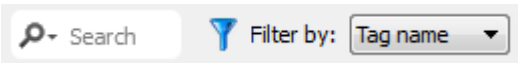


Importer	Description
CX-Programmer v1.1 Linear	Requires a **.cxr** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

OPC UA Client

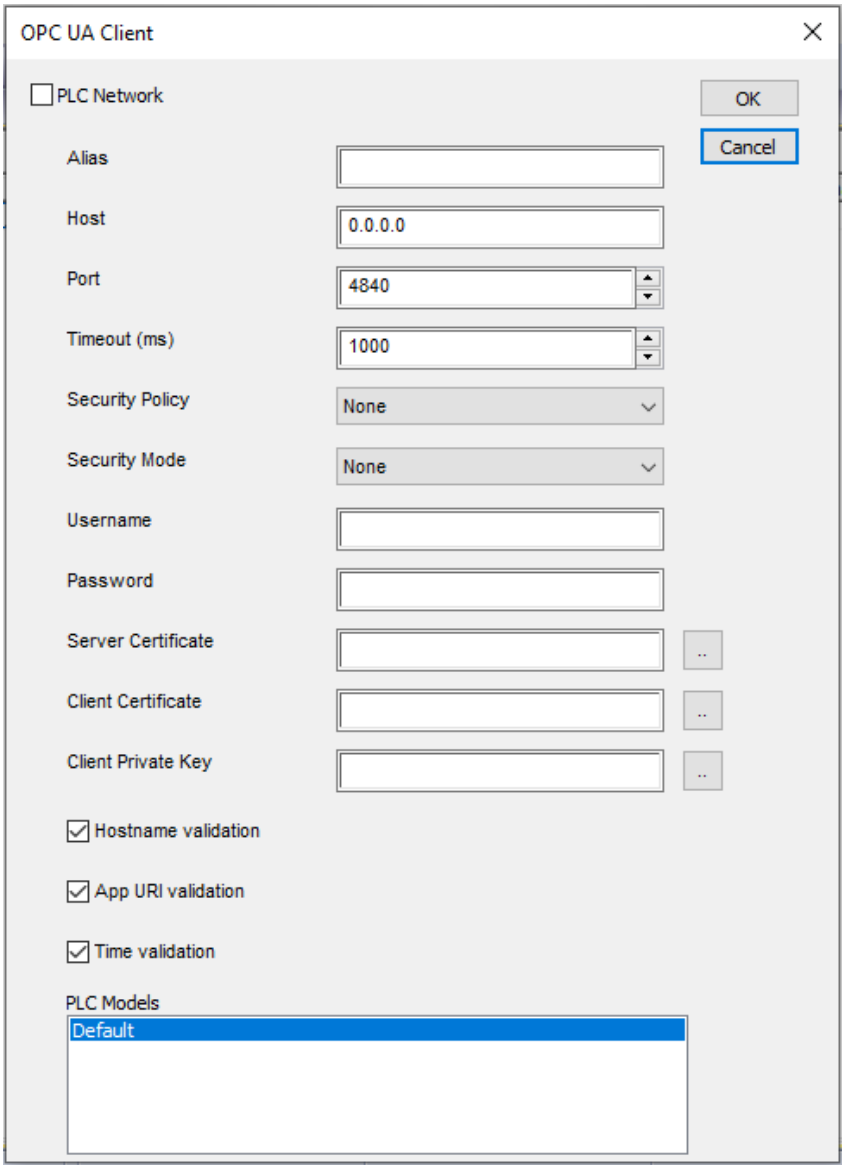
The OPC UA Client communication driver has been designed to connect HMI devices to OPC UA servers.

This implementation of the protocol operates as a client only.

Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “OPC UA Client” from the list of available protocols.

The driver configuration dialog is shown as in the following figure:



Element	Description
PLC Network	Enable access to multiple networked controllers. For every controller set proper options.
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP Address	IP address of the server.
Port	Port number where the server is listening.
Timeout (ms)	Time delay in milliseconds between two retries in case of no response from the server device.
Security Mode	Type of authentication: • **None**: Certificates are not used • **Sign**: Certificates only used for authentication with server. • **SignAndEncrypt**: Certificates used for authentication with server and data encryption.
Security Policy	Encryption level to use (used only when Security Mode is active). • Basic256 • Basic256Sha256
Username Password	Authentication with user name and password
Server Certificate	Certificate for OPC UA Server. Server certificate can be downloaded using tag importer. See "Remote OPC UA Server certificate" on page 542
Client Certificate	Certificate used by OPC UA client. If blank, a certificate is automatically generated.
Client Private Key	Key used by OPC UA client. If blank, a key is automatically generated.
PLC Models	No options available.

Notes:

- Before choosing security options, be aware that not all security modes might be supported by the OPC UA server. Make sure to use security mode that is supported.
- When working within a private network you do not need to provide devices' certificates because you trust used devices. On a public network, instead, the certificate will give you a guarantee of the identity of devices.

External Certificate

ASCII version of the certificate (usually a file with .pem extension) is required.

Edit the certificate files and then copy and paste the full text of your certificate to the certificate fields.

Step 1: Remove header and footer lines

```
-----BEGIN CERTIFICATE-----
MIIDNjCCAh4CCQCJtJgjqDDUqjANBgkqhkiG9w0BAQsFADBdMQswCQYDVQQGEwJJ
VDEPMA0GA1UEBwwGVmVyb25hMRQwEgYDVQQKDAtDb21wYW55TmFtZTERMA8GA1UE
CwwIUiZEIFRlYW0xFDASBgNVBAMMC0hNSURldmljZUlEMB4XDTE4MDMyNjA5MTAz
OFoXDTI4MDMyMzA5MTAzOFowXTELMAkGA1UEBhMCSVQxDzANBgNVBAcMBlZlcm9u
YTEUMBIGA1UECgwLQ29tcGFueU5hbWUxETAPBgNVBAsMCFImRCBUZWFtMRQwEgYD
VQQDDAtITUlEZXZpY2VJRDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
ALONtzGwlrGv6cXH8i7sNWbwmx9Xo4tp20khnt/VJnDLoYHv7ZvV1vQYHom3/HiC
IaWV/uUvYnXaNBlxHnPsQPV0bEEg26Np01ne8jXEHY6bcMVK3XBV3eno3adOwHA5
vio0MmF6fPQVWTfyVb4/MrcfqUkelgWk3sFlFxEtxXlRLOwNK1+G7Wbnb3Oj4oPL
Ev60VN3DwisDzvivpW7Nv4RPjNK9XJ2DVI+/+KDCNNLlP8GpD0xB1iIpj1S8BwqZ
om1+SUs10IM1cfv/AfArZj9QaIo3c2uPwkLncqQxfDvmlC1fCfsRVxm5N3bmimwC
2F6hbkZksLp7ovCx/haKhfkCAwEAATANBgkqhkiG9w0BAQsFAAOCAQEALVjkNEa/
4OJnMZIVkSZZWGylHHGZ8rphcUPH4olbq7MkaHk7mKacYKqI/qorrIPhmKf7Y2x5
UcTN4Uff6NT0xjrMUg2Q6Lp+a/fBqOUvEebrtmd8NYbhjTs4iVYg3R/NBlgrfx9N
6IppO6OJoOhYXjwDZU0HADnSXVABeBxzAESvLVK7mxgXypdB1D+kgcC6hL9Xv4u5
melNI24LNkRiBT35Exlo2YTu4I9YHFelc5iILvC6DpUYHeSlIEKiNmccL2DDGEBZ
TscRZykvWRi1Xpm2WMzjbf9HE0XNRM8DTCkOscxcrYZrcTVpm0a0WH5OD2531LnF
XsH5sLPyOxtKFw==
-----END CERTIFICATE-----
```

Step 2: Remove all Newline characters

```
MIIDNjCCAh4CCQCJtJgjqDDUqjANBgkqhkiG9w0BAQsFADBdMQswCQYDVQQGEwJJVDEPMA0GA1.....
```

Step 3: Copy and paste the single text line of the certificate to the protocol dialog

Script to generate a Certificate

If you want to use your own certificate, note that the certificate must include the “Subject Alternative Name (SAN)” parameters as required by the OPC UA standard.

Here is an example of how to generate certificate files using a public OpenSSL-Win32 library (Reference: https://www.openssl.org/)

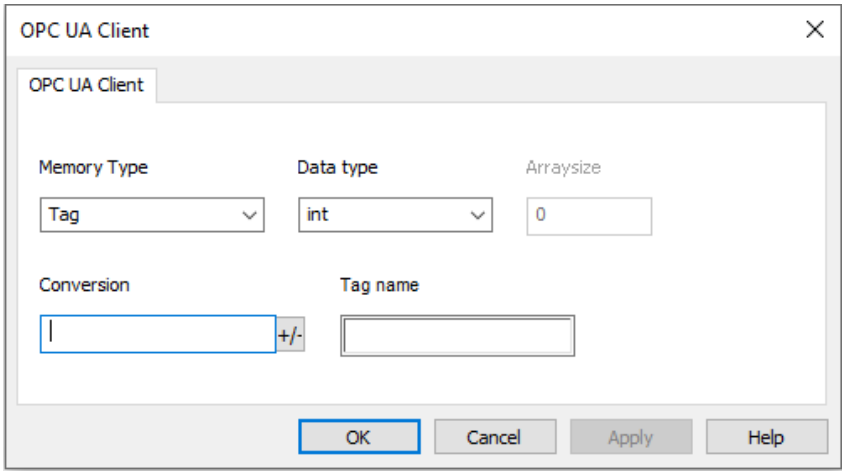
```
@echo off
set OpenSSL="C:\Program Files (x86)\OpenSSL-Win32\bin\openssl.exe"
set NodeName=HMI-Client

rem Generate an RSA key
    %OpenSSL% genrsa -out client-key.pem 2048

rem Creating Certificate Signing Requests
    %OpenSSL% req -new -key client-key.pem -out client.csr -subj "/ST=NY/C=US/L=New
York/O=CompanyName/OU=R&D Team/CN=OPCUAClient@%NodeName%"

rem Creating Certificate (.pem)
    echo subjectAltName=URI:urn:%NodeName%:CompanyName:OPCUAClient > san.txt
    echo
keyUsage=digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment,keyCertSign
>> san.txt
    echo extendedKeyUsage=critical,serverAuth,clientAuth >> san.txt
    echo authorityKeyIdentifier=keyid,issuer >> san.txt
    echo basicConstraints=CA:TRUE >> san.txt
```

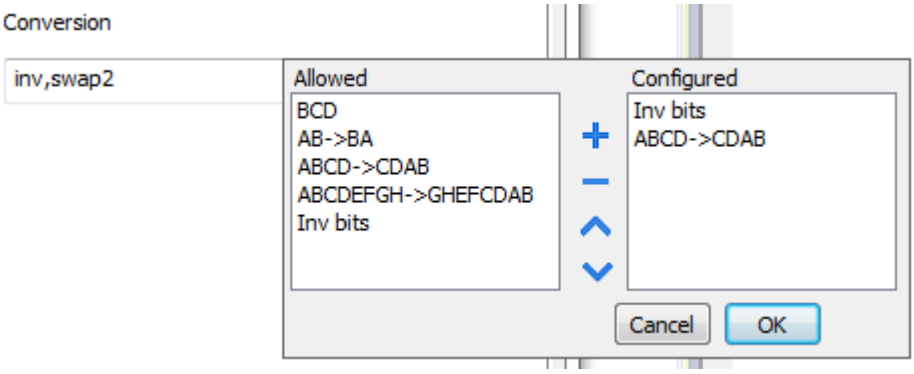
```
    %OpenSSL% x509 -req -days 3650 -in client.csr -signkey client-key.pem -out
client.crt -extfile san.txt

rem    Convert Certificate (.der)
    %OpenSSL% x509 -in client.crt -outform der -out client.der

rem Not necessary files
    del san.txt

pause
```

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. Select **OPC UA Client** from the protocol list.
2. To add a tag, click **+**: tag definition dialog is displayed.



Element	Description
Data Type	Available data types: • **boolean** • **byte** • **short** • **int** • **unsignedByte** • **unsignedShort** • **unsignedInt** • **float** • **double** • **time** • **uint64** • **int64** • **string** • **binary** See "Programming concepts" section in the main manual. Note: To define arrays, select one of Data Type format followed by square brackets.
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000 1 → 1 10000011100 10101010000101000101101101101100101101100001001111 01 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.
Tag name	Name of tag to be used in communication.

Note: Tag properties result from import process. In most cases manual creation of new tags is not necessary.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

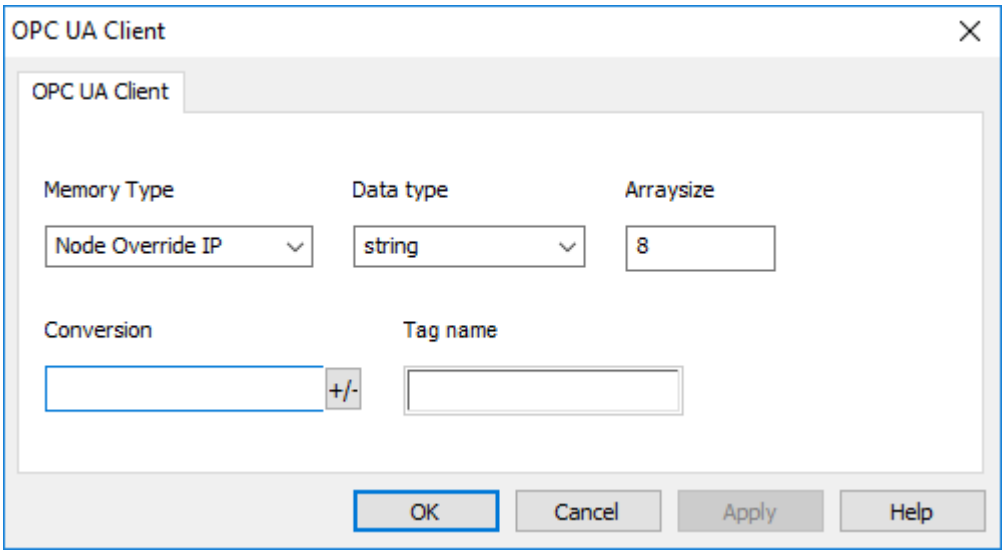
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.



Node Override Port

The protocol provides the special data type Node Override Port which allows you to change the network Port of the target controller at runtime.

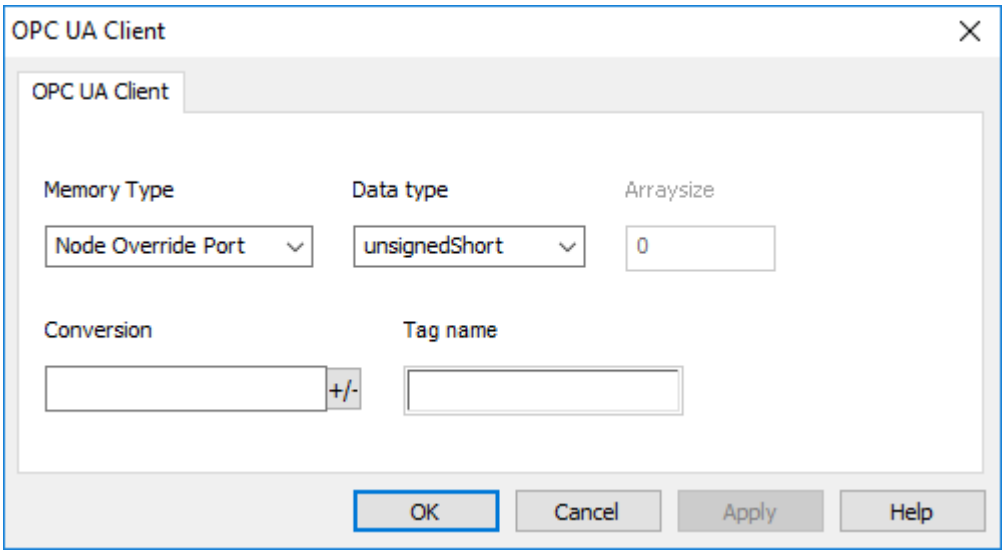
This memory type is unsigned short.

Node Override Port is initialized with the value of the controller Port specified in the project at programming time.

Node Override Port	Modbus operation
0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0	It is interpreted as the value of the new port and is replaced for runtime operation.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override Port variable.

Note: Node Override Port values assigned at runtime are retained through power cycles.

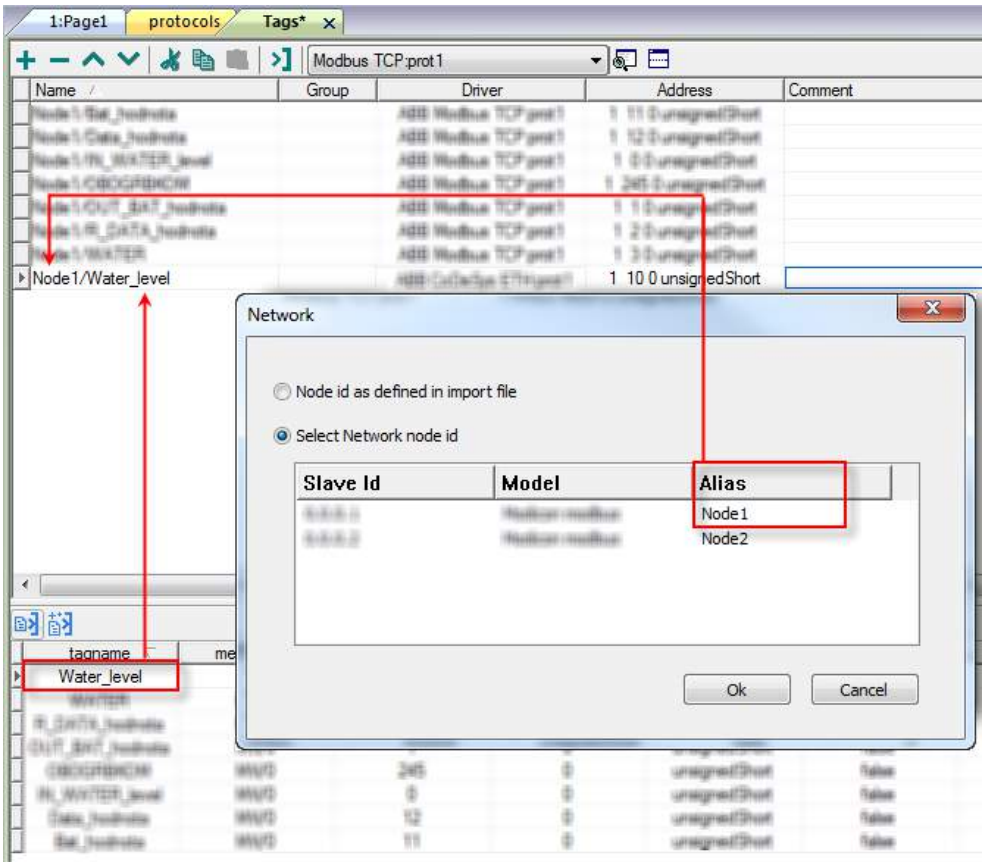


Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



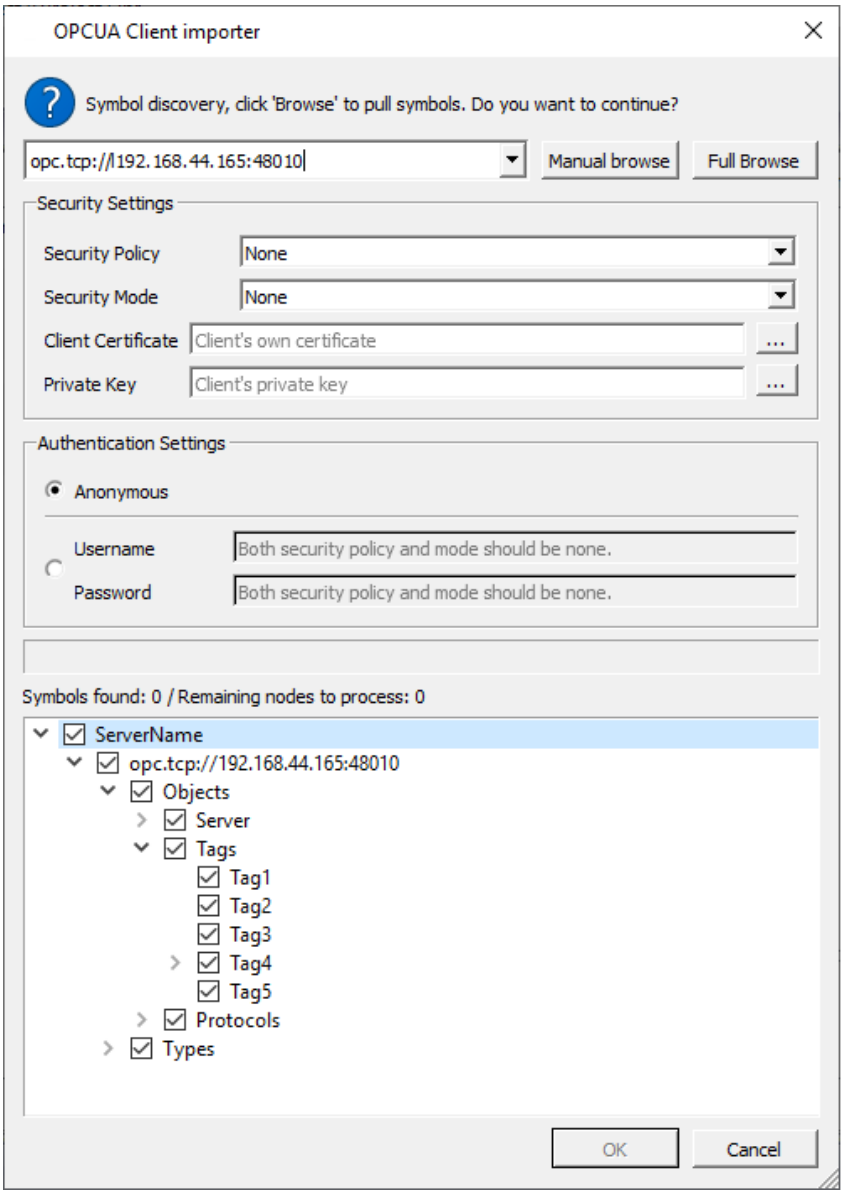
Note: Aliasing tag names is only available for imported tags. Tags added manually in the Tag Editor cannot have the Alias prefix in the tag name.
The Alias string is attached at the time of tag import. If you modify the Alias string after the tag import has been completed, there will be no effect on names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

Importing tags

Tags for OPC UA Client protocol must be imported from OPC UA servers.

*Path: **ProjectView*> ***Config*** > *double-click **Tags***

1. Select **OPC UA Client** in the list of available protocols.
2. Click **Import Tags.**
3. Select **Hierarchical importer.**
4. Enter address of the server.
5. Choose Security and Authentication mode.
6. Click **Browse** to connect and retrieve tag dictionary from the OPC UA server.
7. The OPC UA Server will provide its own certificate. You have to accept the certificate to continue and retrieve data.
8. When the discovery process has been completed, click **OK** to create the dictionary with the tags.

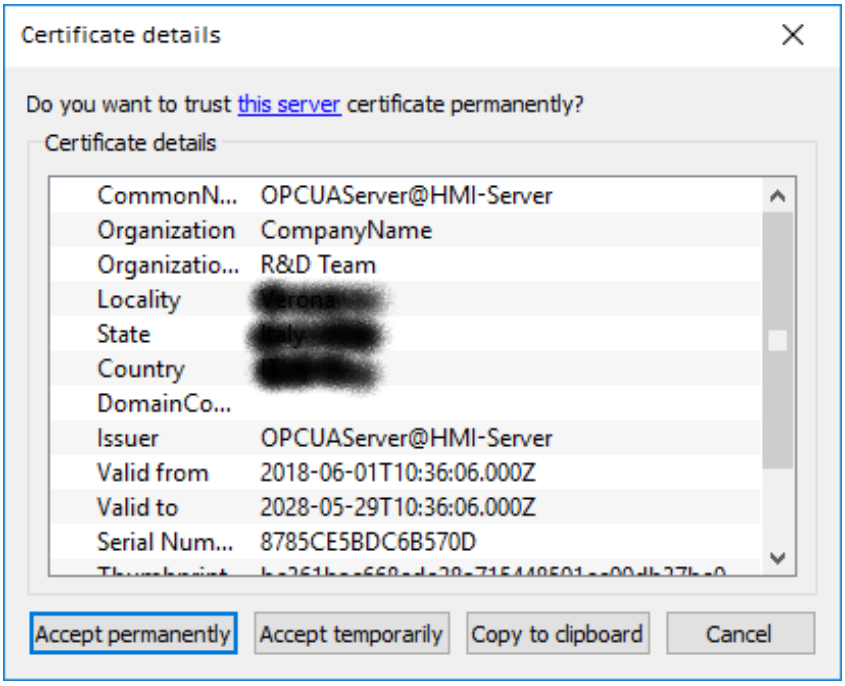


Element	Description
Remote URI	Address of OPC UA Server in the form: *opc:tcp:<IPAddress>:<Port>* Example: • opc.tcp://192.168.44.165:4840
Security Mode	Type of authentication: • **None**: No authentication with server and no data encryption. • **Sign**: Certificates only used for authentication with server. • **SignAndEncrypt**: Certificates used for authentication with server and data encryption.
Security Policy	Encryption level to use (used only when Security Mode is active).

Element	Description
	• Basic128Rsa15 • Basic256 • Basic256Sha256
Username Password	Authentication with user name and password
Client Certificate	Certificate used by OPC UA client. If blank, a certificate is automatically generated. The certificate is used by the importer only if requested by the server
Client Private Key	Key used by OPC UA client. If blank, a key is automatically generated.

To be allowed to retrieve data from the OPC UA Server you must provide the required security parameters. Dialog will be filled automatically with the parameters provided by protocol editor settings (you can simply accept the proposed values)

Remote OPC UA Server certificate



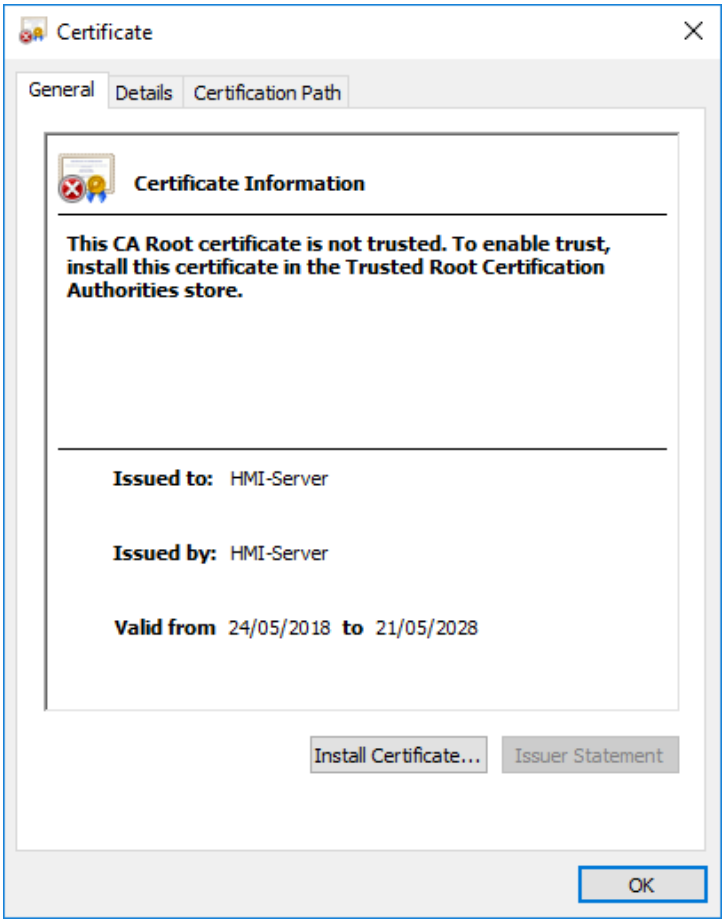
When OPC UA Server provides its own certificate, you have the option to:

- **Accept temporarily**
  Certificate is accepted for current working session only.
- **Accept permanently**
  Certificate is accepted and copied to computer. Any future import request for the same OPC UA Server will be accepted automatically without asking confirmation.

The certificate file will be copied inside the folder:
%AppData%\Roaming\...\studio\OPCUA\pki\trusted\certs

- **Copy to clipboard**
  ASCII format of the certificate is copied to the clipboard to allow you to verify its authenticity, save and insert it into protocol configuration (if required).

  To verify a certificate, use a text editor to paste it from the clipboard to a text file with the extension .crt. You can then double-click the .crt file to allow Windows to view the properties of certificate.



- **Cancel**
  Cancel the import operation

Communication status

Current communication status can be displayed using System Variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Description
Connecting <Error description>	Error during connection
Connection while reading: <Error description>	Error encountered when connecting for read operation
Bad status while reading: <Error description>	Error in read operation

Error	Description
Connection while writing: <Error description>	Error encountered when connecting for write operation
Bad status writing: <Error description>	Error in write operation
OPC UA client for given node ID not found	Wrong node ID information

<Error description> can be one of the following:

Error	Notes
BadTimeout	Timeout error. No answer from server.
BadSecurityChecksFailed	Error during exchange of certificates. Typically occurs when the server does not accept the client certificate as trusted.
BadCertificatexxxInvalid	Error in client or server certificate.
BadNodeUnknown	The tag (node) does not exist.
BadAttributeNotFound	Attempt to access an invalid attribute.
BadNotWritable	Attempt to write to a read-only attribute.

Panasonic FP

The operator panels can be connected to a Panasonic FP PLC as the network master using this communication driver.

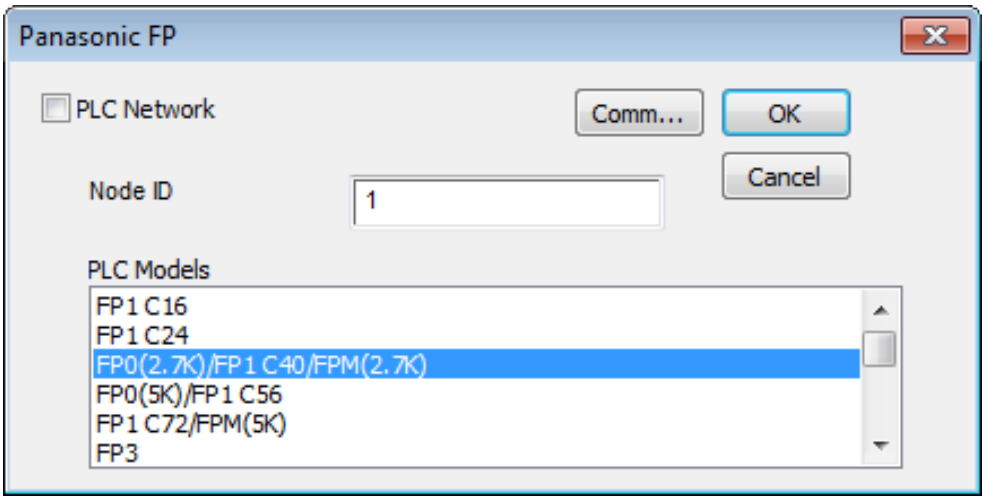
This driver has been designed for connection to the programming port of the PLC.

Please note that changes in the communication protocol specifications or PLC hardware may have occurred since this documentation was created. Some changes may eventually affect the functionality of this communication driver. Always test and verify the functionality of your application. To fully support changes in PLC hardware and communication protocols, communication drivers are continuously updated. Always ensure that the latest version of communication driver is used in your application.

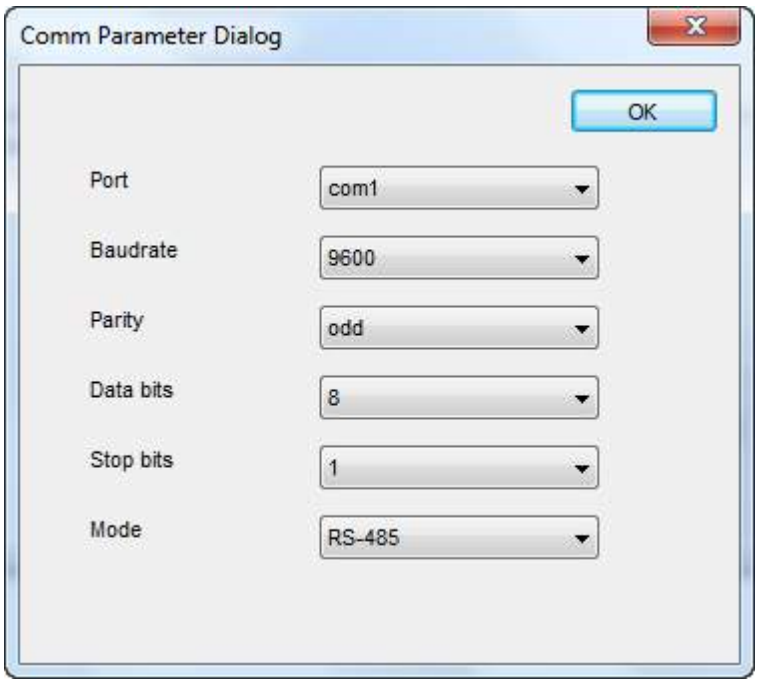
Protocol Editor Settings

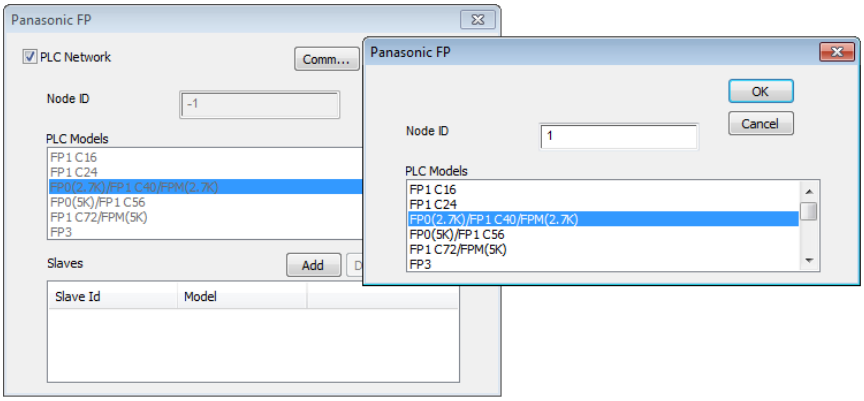
Add (+) a driver in the Protocol editor and select the protocol called “Panasonic FP” from the list of available protocols.

The driver configuration dialog is shown in the following figure.



Element	Description
Node ID	Node number of the slave device
PLC Models	The list allows selecting the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.
Comm…	Recalls the serial port configuration parameters as shown in the figure below.

Element	Description
	
Port	Serial port selection. COM1 is the PLC port. COM2 is PC/Printer port on panels with two serial ports or refers to the optional Plug-In module plugged on Slot 1 (or 2) for panels with one serial port on-board. COM3 refers to the optional Plug-In module plugged on Slot 3 (or 4) for panels with one serial port on-board.
Baud rate **Parity** **Data bits** **Stop bits**	Communication parameters for serial communication

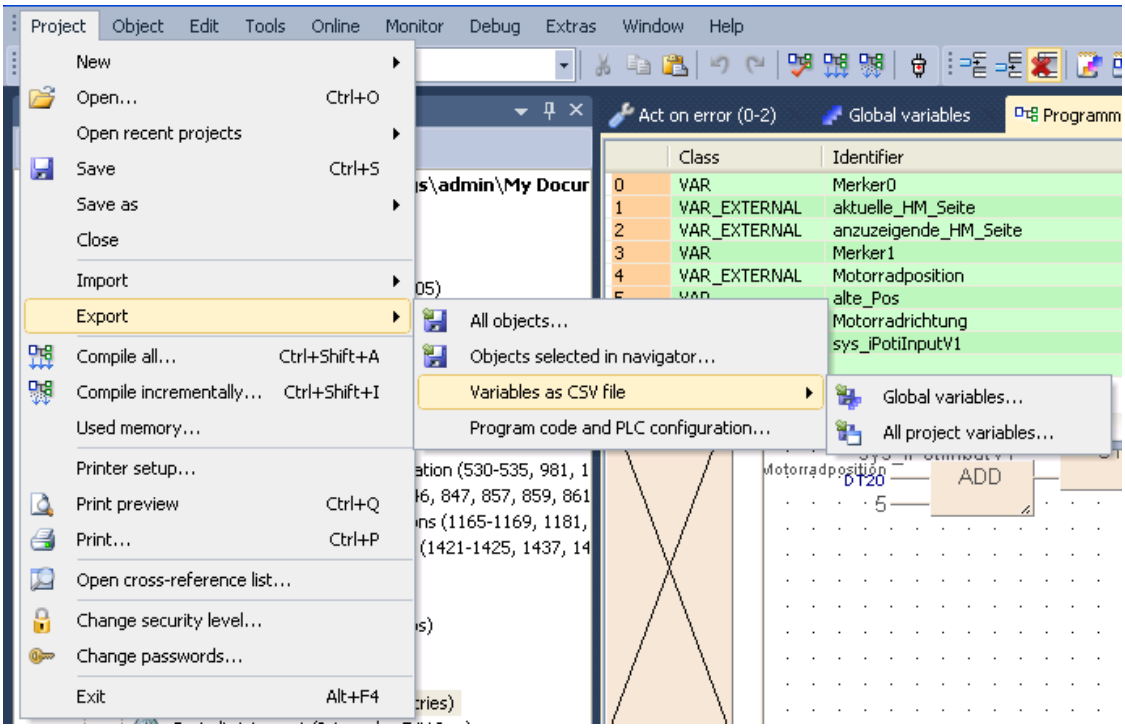
Element	Description
Mode	Serial port mode. Available options: RS-232, RS-485 (2 wires) RS-422 (4 wires)
PLC Network	The protocol supports connection to multiple controllers. To enable this, check the "PLC Network" check box and provide the configuration per each node. 

Tag Import

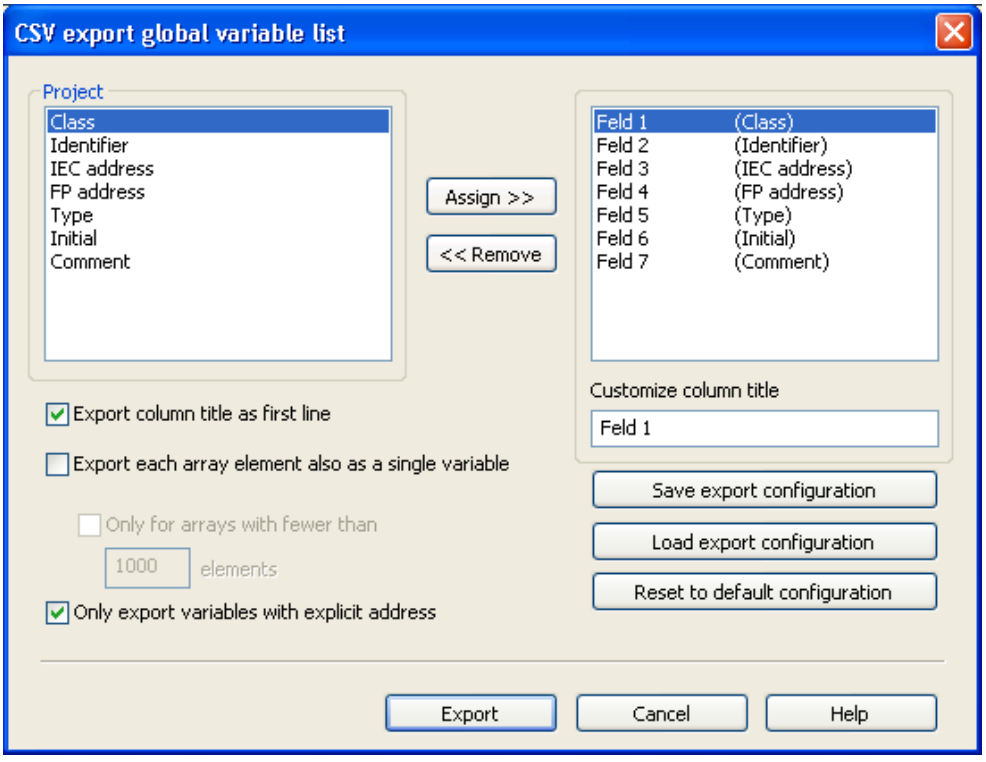
Exporting Tags from PLC

The Panasonic FP driver supports the Tag Import facility. The symbol file can be exported by the controller programming software FPWIN.

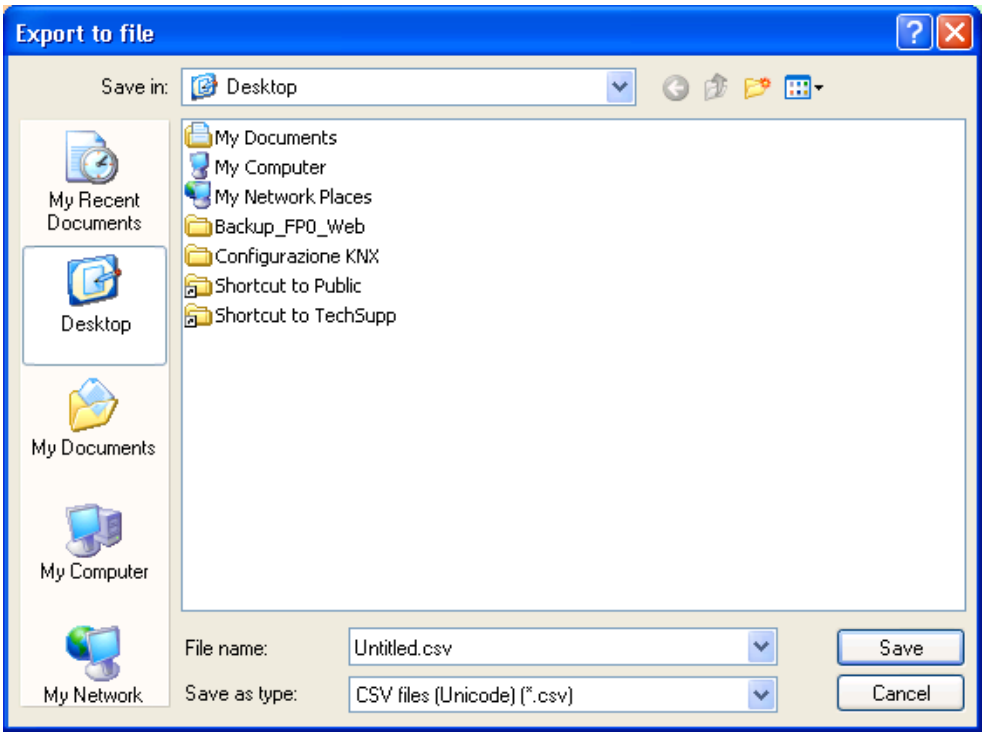
In FPWIN menu, click on “Project > Export > Variables as CSV file”, then you can choose if you want to export only the Global variables or All project variables.



If you choose to export only the Global variables, FPWIN will show the window of the following picture that allow to customize the elements of the exported csv file.

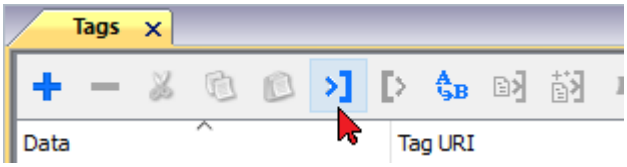


Then, in the “Export to file” window, choose the “CSV file (Unicode)” format.

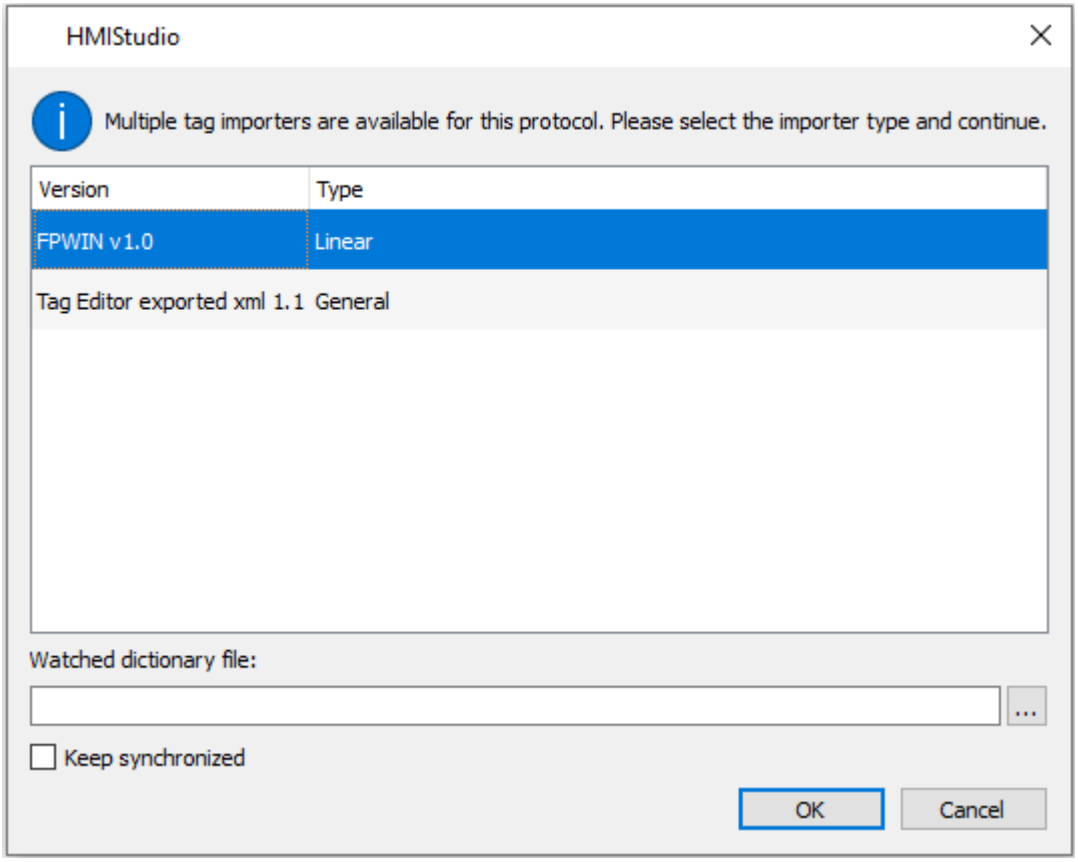


Importing Tags in Tag Editor

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



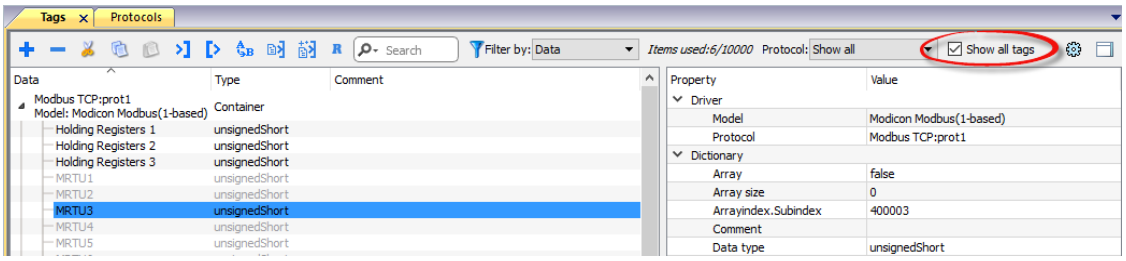
The following dialog shows which importer type can be selected.

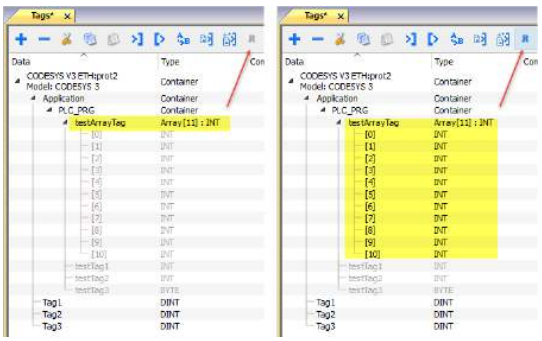
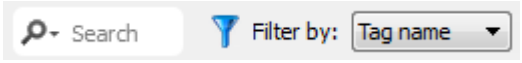


Importer	Description
FPWIN v1.0 Linear	Requires a **.csv** file. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits); ensure the communication parameter settings of the controller is compatible with panel communication setup
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources
General Error	Error cannot be identified; should never be reported; contact technical support

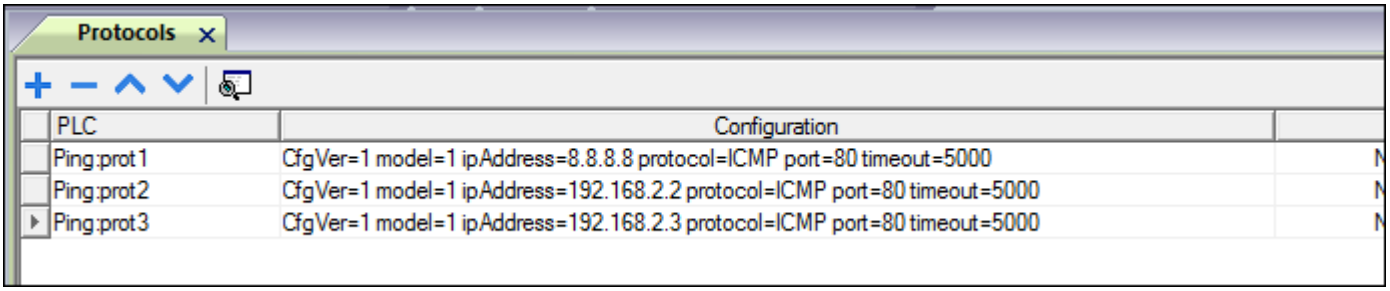
Ping

Ping communication driver allows to send ping commands to a specific IP address.

The purpose of this communication driver are:

- test a connection between the HMI and another device in the same network
- check internet connectivity by executing ping commands to a public IP address (example 8.8.8.8)

In case it is needed to send ping commands to many IP addresses at the same time, it is possible to create many instances of Ping protocol:



Ping communication driver is not counted as physical protocol.
Refer to **Table of functions and limits** from main manual in "Number of physical protocols" line.

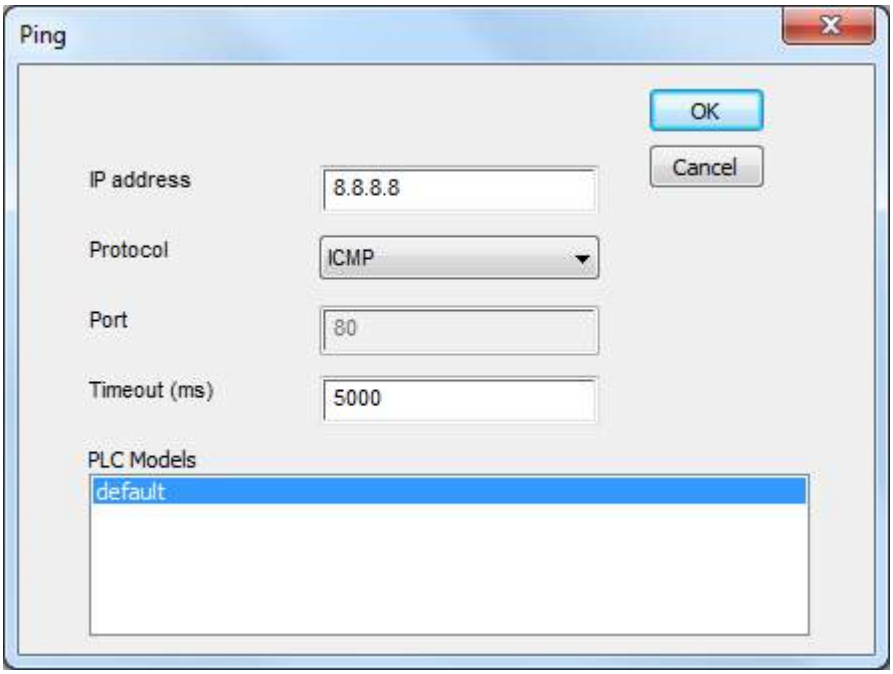
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.

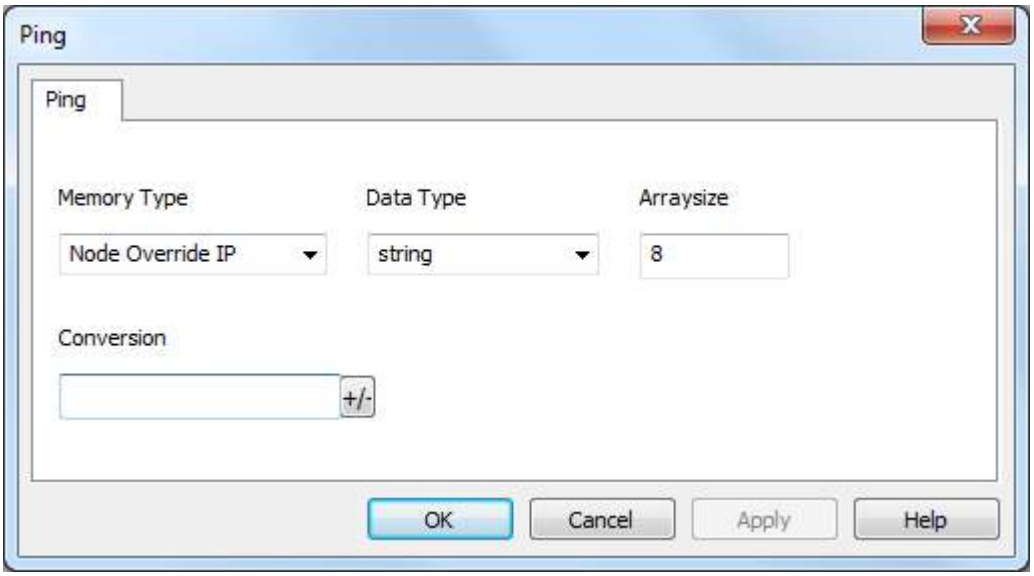


Element	Description
IP address	Destination IP address to which ping commands are sent.
Protocol	Network protocol used to send ping commands (default is ICMP).
Port	Network port used for sending ping commands (fixed to 53 for ICMP Protocol).
Timeout (ms)	Polling time between each ping command sent.
PLC Models	Fixed to default.

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Ping** from the protocol list: tag definition dialog is displayed.

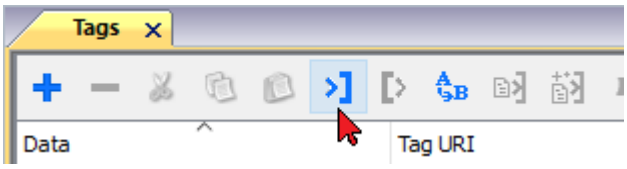


<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Node Override IP</td><td>If defined, this Tag allows to change the destination IP address to which ping commands are sent, at runtime.</td></tr>
<tr><td>Status</td><td>Represents the result of last ping command:
0 = last ping command failed
1 = last ping command got response</td></tr>
<tr><td>Last ping time</td><td>Represents the result of last ping time, expressed in milliseconds.</td></tr>
</table>
</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>unsignedByte[]</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>string</td><td colspan="2">Express the number of characters used to specify the destination IP address
Example: string[15] --> xxx.xxx.xxx.xxx</td></tr>
</table>
</td></tr>
<tr><td>Arraysize</td><td>This property represents the maximum number of bytes available in the string or in the array Tag.

Note: number of bytes corresponds to number of string chars if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one char requires 2 bytes.</td></tr>
</table>

Tag Import

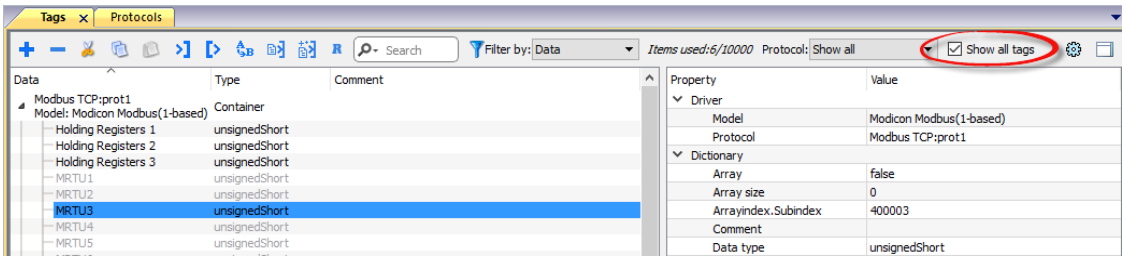
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



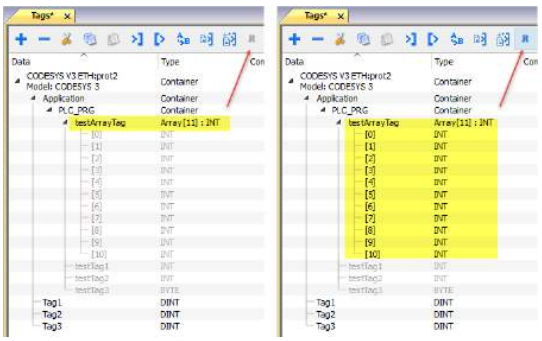
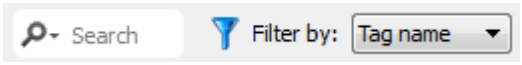
The system will require a generic XML file exported from Tag Editor by appropriate button.

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	Import Tag(s). Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	Update Tag(s). Click on this icon to update the tags in the project, due a new dictionary import.
R	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

ProConOS ETH

The ProConOS ETH driver has been developed for the connection to ProConOS compatible controllers via Ethernet.

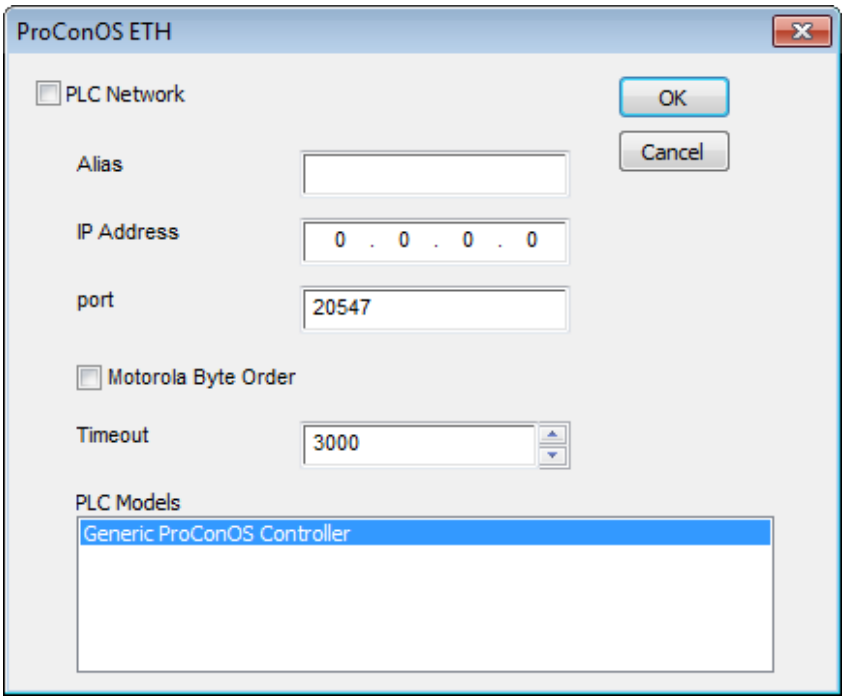
Yaskawa MPiec controllers that can communiucate using ProConOSdriver are:

- MP2300Siec
- MP2310iec

For such models it is possible to export variables to be imported in Tag Editor (see **Tag Import** chapter).

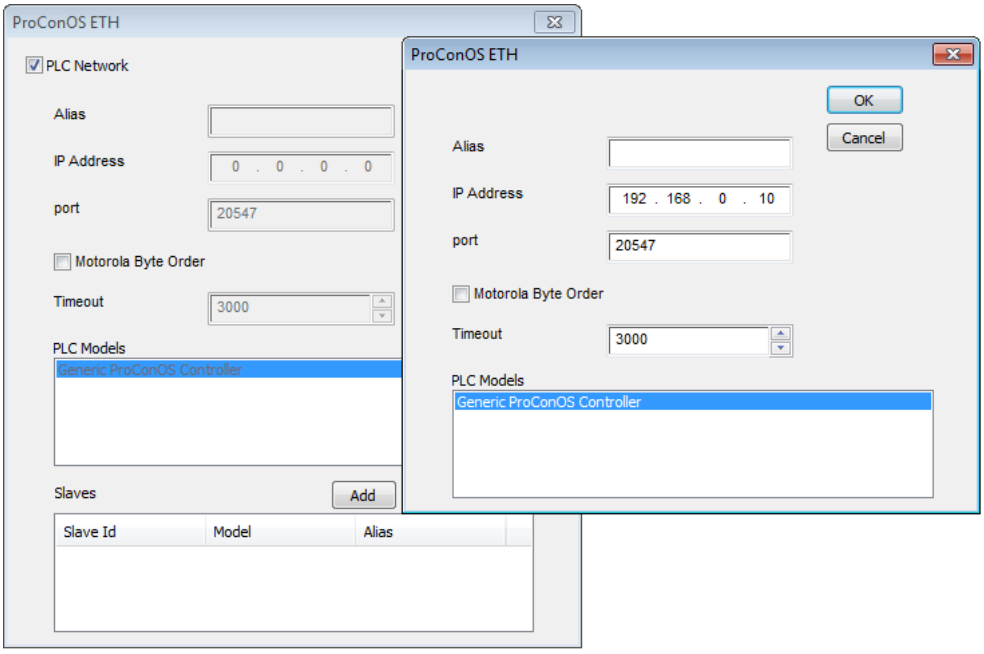
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “ProCoNos ETH” from the list of available protocols.



Element	Description
Alias	Name to be used to identify nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node
IP Address	Controller IP address
Port	Controller port number for Ethernet interface
Motorola Byte Order	This option is used to identify if the PLC you’re working with is a Big Endian type (default, option checked), or Little Endian (option unchecked).
Timeout	The time the protocol waits the answer from the controller before issuing a new retry.

Element	Description
PLC Models	List of compatible controller models. Make sure to select the right model in this list when configuring the protocol.
PLC Network	The protocol supports connection to multiple controllers. To enable this, check the "PLC Network" check box and provide the configuration per each node.



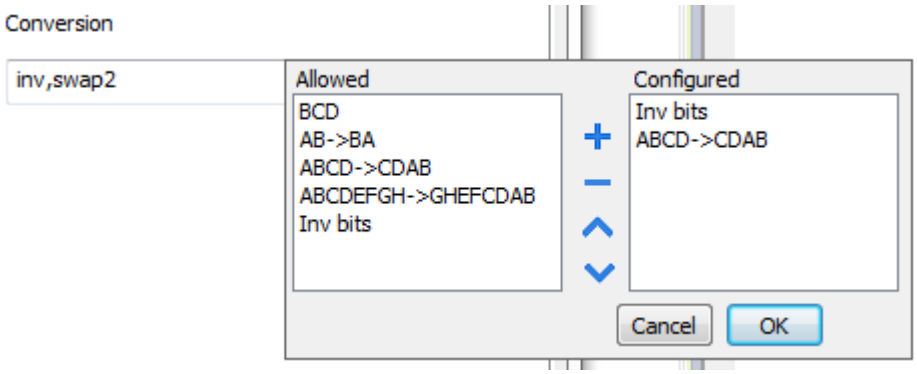
Data Types

The import module supports variables of standard data types as per the following list.

- BOOL
- SINT (8-bits signed integers)
- INT (16-bit signed integers)
- DINT (32-bits signed integers)
- USINT (8-bits unsigned integers)
- BYTE (8-bits unsigned integers)
- UINT (16-bit unsigned integers)
- WORD (16-bit bit strings, displayed as unsigned integers)
- UDINT (32-bits unsigned integers)
- DWORD (32-bit bit strings, displayed as unsigned integers)
- REAL (32-bit floating point data)
- LREAL (64-bit floating point data)
- TIME
- STRING (character string)

Tag Conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001

Value	Description
	→ 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Special Data Types

The ProCoNos Ethernet driver provides one special data type called "Node Override IP".

The Node Override IP allows changing at runtime the IP address of the target controller you want to connect. This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

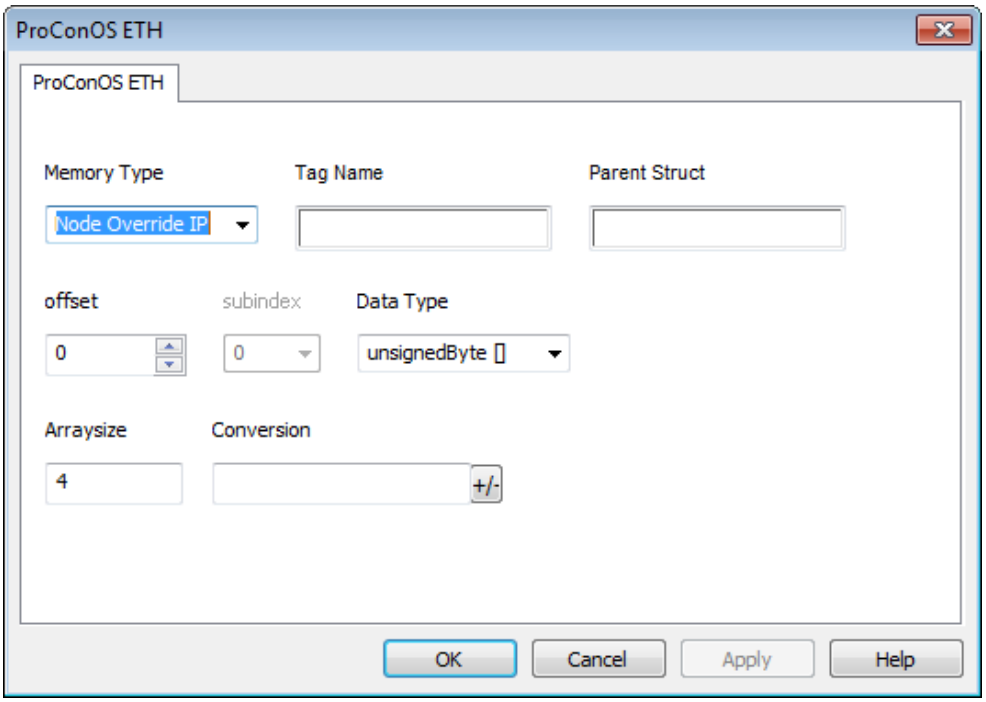
The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

If the IP Override is set to 0.0.0.0, all the communication with the node is stopped, no request frames are generated anymore.

If the IP Override has a value different from 0.0.0.0, it is interpreted as node IP override and the target IP address is replaced at runtime with the new value.

In case the panel has been contabld to access to a network of controllers, each node has its own Override variable.

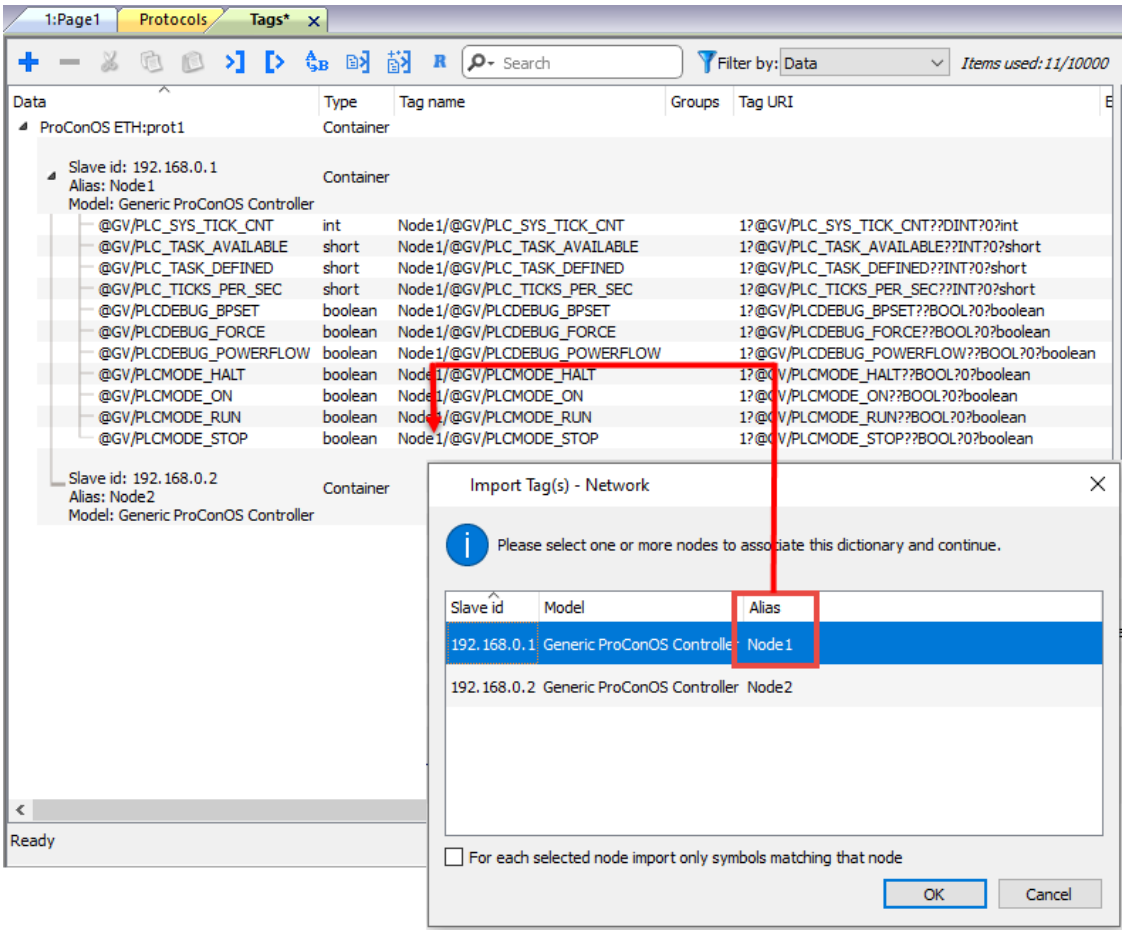
Note: the IP Override values assigned at runtime are retained through power cycles.



Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names are to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



Note: Aliasing tag names is only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name. The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

Tag Import

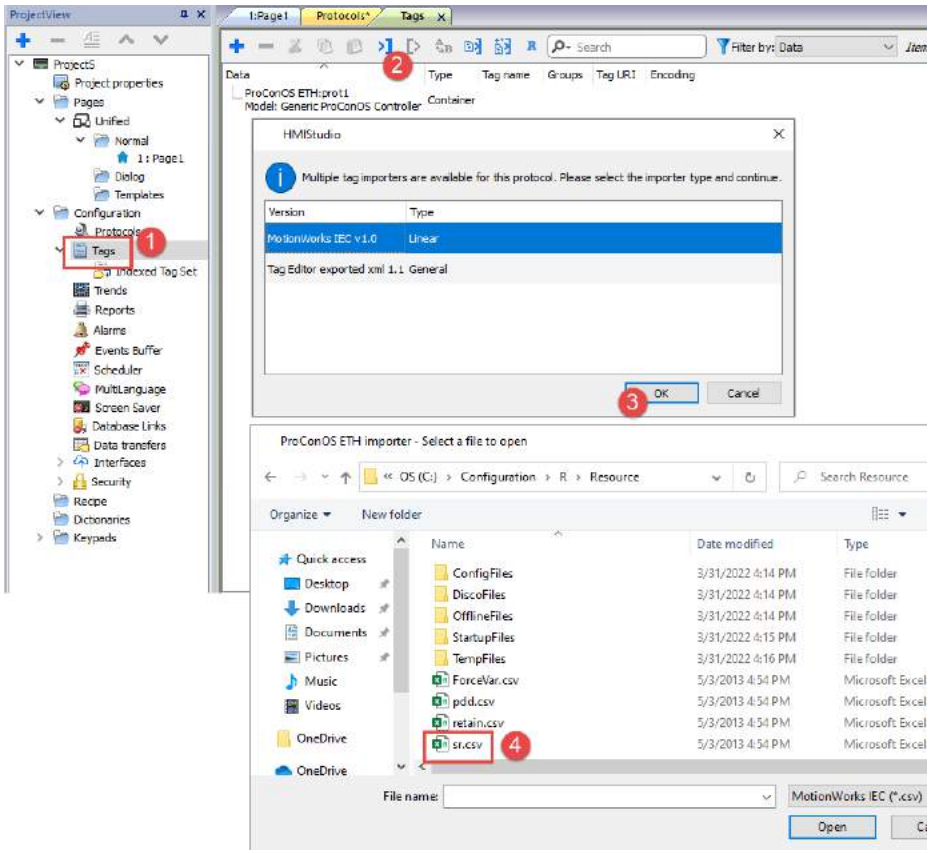
The ProCoNos Ethernet driver support the Tag Import facility.

The symbol file can be exported by the controller programming software.

To import the tags from IEC project:

1. Select the Tags tab from ProjectView
2. Click the "Import tag" button
3. In the Tag Import window click the "OK" button to select the .csv file
4. Point to the "sr.csv" file from the IEC project

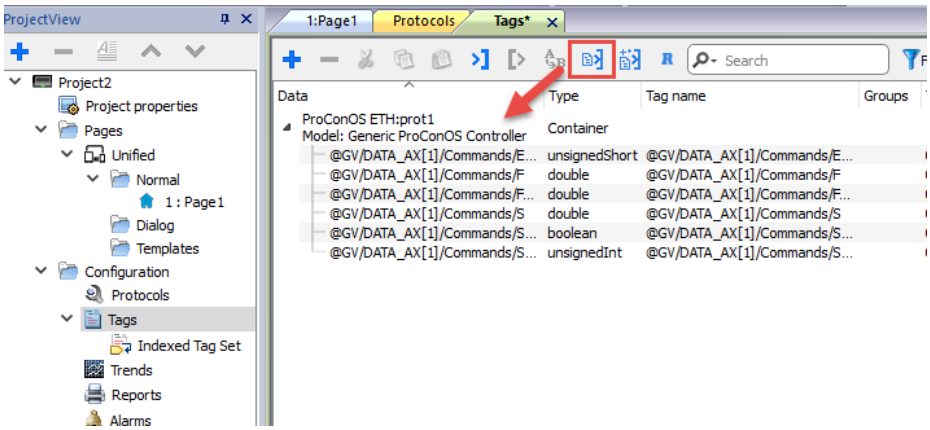
   The Path is “ProjectFolder > C > Configuration > R > Resource”



5. After “sr.csv” file import, select the "DT" Directory for Data Types.

   If the IEC project contains custom data types you have to select the "DT" folder from IEC Project to correctly import all the Tags.

6. Now all the variables are available as Dictionary in project. Select the desired variables and add to the tag list as shown in the figure below.



Communication Status

The communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The status codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Invalid response	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support

Profibus DP

The Profibus DP communication driver has been designed to connect HMI products to a Profibus DP network as slave nodes. With the Profibus DP driver, the HMI simply exchanges Input and Output data with the Master. It is up to the Master to make sense of this data.

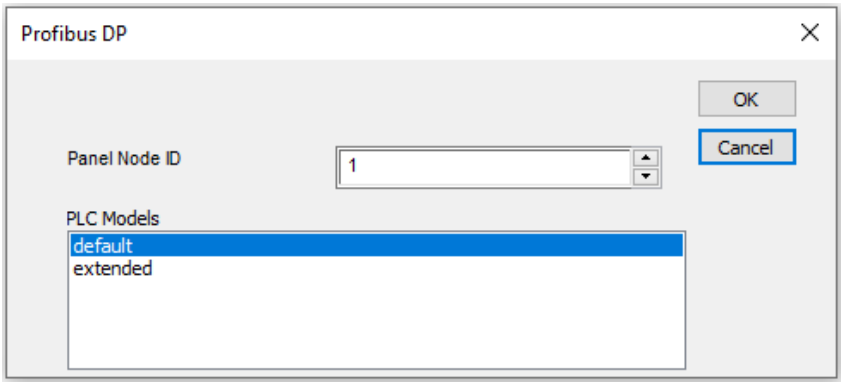
Connection to Profibus DP network requires the optional Profibus DP communication module. Verify the suitable version for your HMI model.

Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Please ensure that the latest driver is used in the application.

Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “Profibus DP” from the list of available protocols.

The driver configuration dialog is shown in figure.



Element	Description
Panel Node ID	The Profibus node ID assigned to the HMI

Configuring the HMI as a Slave Node

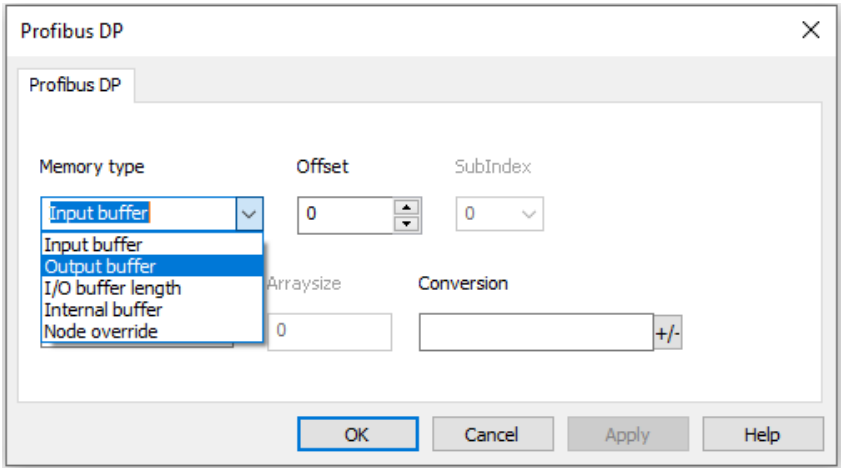
The Profibus DP master must be configured to communicate with the slaves devices present in the network. To configure the Master System you will generally need a software package available from the manufacturer of the Master System. Before the master configuration software can recognize the the HMI device as slave, it must be included in the catalog of devices. For this purpose it is available a device description file in the standard GSD format. The device description file is EX9649AX.GSD. It must be installed following the instructions of the network configuration software you are using.

One of the fundamental steps of the configuration of a slave station in a Profibus DP system is the mapping of the slave’s I/O buffers in the memory of the master.

The HMI panels support Input / Output buffer sizes of 8, 16 or 32 bytes and they expect that both the Input and the Output areas are configured to the same size, i.e. both 8 bytes, either 16 bytes or both 32 bytes. The HMI panels will automatically detect the buffer size used by the master.

The feature generally referred to as Response Monitoring should always be disabled in the master for the HMI panel slaves.

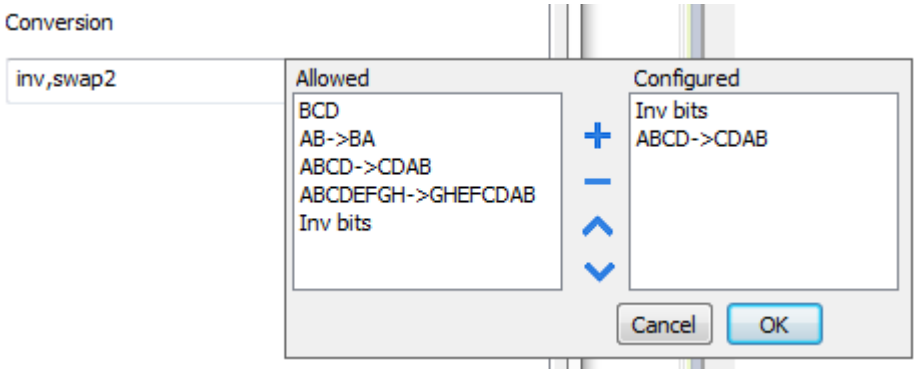
Tag Editor Settings



1. Studio allows you to access the HMI panel “Output Buffer”, the area containing data sent from the PLC, as well as the HMI panel “Input Buffer”, the area containing data to be sent to the PLC. The data in the Output Buffer is read only, while the data in the Input Buffer is read write. The Address Offset range (in bytes) for these 2 types is from 0 - 31. It should be borne in mind, however, that that Input / Output buffer range configured in the PLC for the panel can be either in the range 0 - 7, 0 - 15 or 0 - 31.
2. In addition to the Input Buffer and the Output Buffer Designer also allows you to access the “Internal Work Buffer” data type. This buffer is purely an internal buffer in the panel. The panel sets aside 256 bytes for this buffer. The data in this buffer is neither read from nor written to the PLC. It is purely a work area.

Tag Conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Special Data Types

The Profibus DP communication driver provides one special data type called "Node Override".

The Node Override ID allows changing at runtime the value of Panel Node ID. This memory type is an unsigned byte.

The Node Override ID is initialized to the value defined as Panel Node ID in the project at programming time.

The communication with the master is described in the table.

Node Override ID value	Behavior
0	The communication with the master is stopped
1 to 255	If Node Override ID has a value different from 0, it is interpreted as the new node ID for the slave device.

Note: the Node Override values assigned at runtime are retained through power cycles

Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Invalid response	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General error	Error cannot be identified; should never be reported; contact technical support

Profibus DP S7

The Profibus DP S7 communication driver has been designed to connect the HMI products to a Profibus DP network as slave nodes. This communication driver has been specially created to offer optimal data exchange features for Profibus DP networks where the bus master is a Siemens Simatic S7 PLC.

This Technical Note gives the technical details for a successful connection.

A Profibus DP network can contain multiple nodes. A node in a Profibus DP network can be either a Master or a Slave. The Masters in the network have a group of Slaves assigned to them. A Master is able to exchange data with the Slaves that are under its control.

The HMI panel is always a Slave device in a Profibus DP network and it is only able to exchange data with a single Master PLC. The HMI has a complex communication profile, as it needs to access data in the Master PLC memory. This communication profile is not something normally available for Profibus DP Slave devices. To enable the HMI to communicate under this profile, a set of special function blocks must be added to the PLC program in the Master PLC. These special function blocks are required by the PLC to process the requests from the HMI. These special function blocks use a Data Block, called the Comm DB, within the Master PLC to store configuration information. This approach has the advantage that it offers to the HMI slave device full access to the data in the PLC, as if the HMI device was directly connected to the programming port of the PLC.

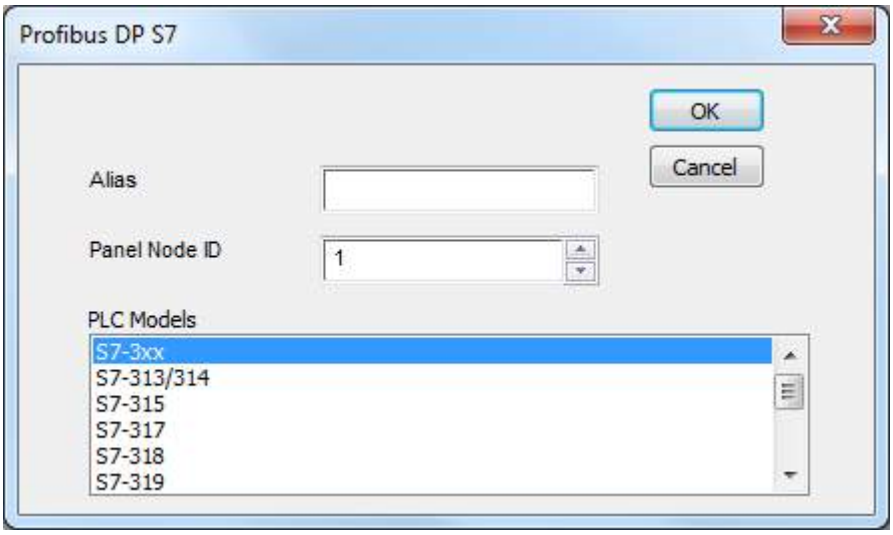
Note: Connection to Profibus DP network requires the optional Profibus communication module. Verify the suitable version for your HMI model.

Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Please ensure that the latest communication driver is used in the application.

Protocol Editor Settings

Add [+] a driver in the Protocol Editor and select the protocol called “Profibus DP S7” from the list of available protocols.

The driver configuration dialog is shown in figure.



Element	Description
Panel Node ID	The Profibus node ID assigned to the HMI.
PLC Models	List of compatible controller models. Make sure to select the correct PLC model in this list when configuring the protocol.

Configuring HMI as a slave station with STEP7

The Master PLC must be configured to communicate with Profibus DP slaves. You can do this with the STEP 7 programming software. This package configures the Profibus DP network attached to the Master PLC (or to the CP communication processor) so that it exchanges data with the specified Slaves. With this package you can select different types of Slaves such as HMI, distributed I/O, drives, etc.

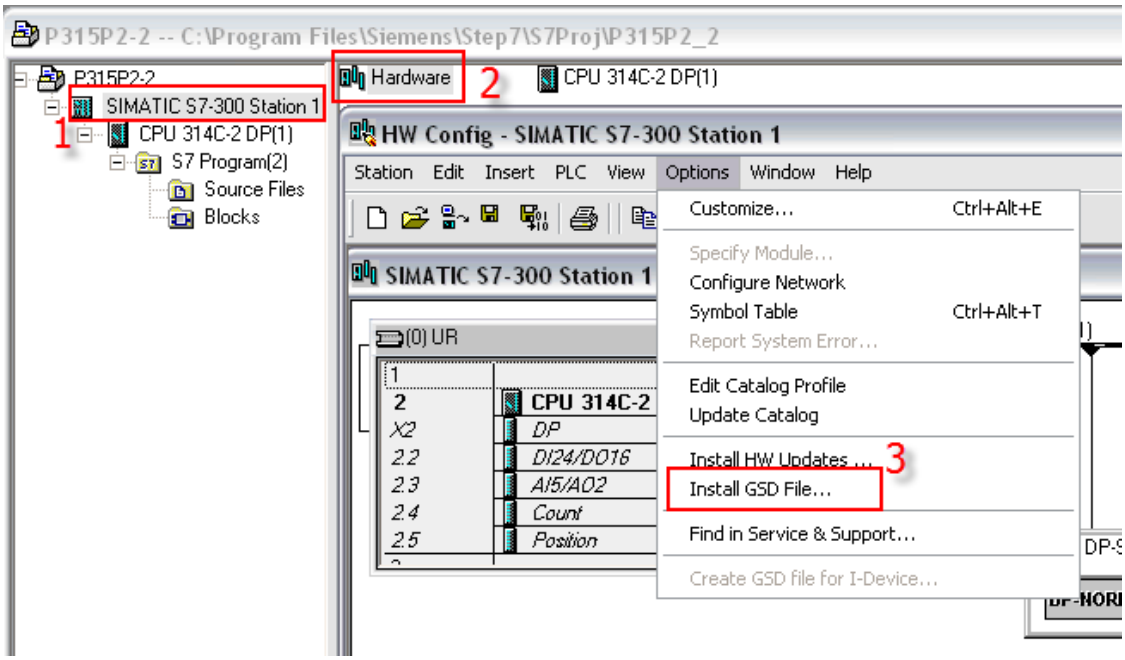
Note: Step7 versions 5.5 SP1 has been used to create the examples included in this Tech Note. Using another version of the Step7 software package may require changes to the procedures described in this document.

Adding the DDB file to your system

A Profibus DP Slave type file (GSD) is available for the Profibus DP configuration. The filename is EX9649AX.GSD; this file contains the description of the HMI devices as Profibus DP Slaves.

To include the file in the system, follow the procedure:

1. Select your station
2. Double click on “Hardware” to open “HW Config” editor
3. From menu Options select “Install GSD File…” and follow the wizard



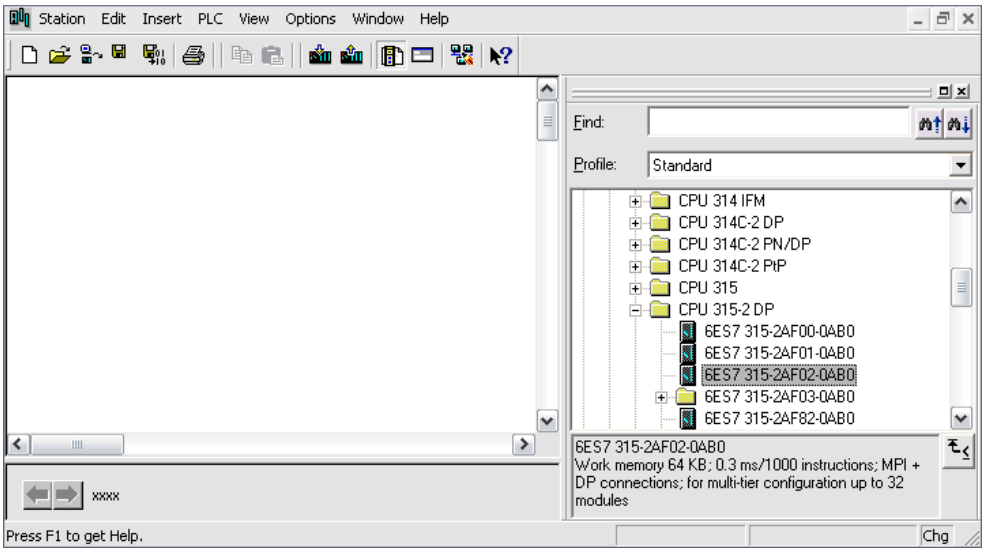
This will enable STEP 7 to recognize the HMI panels as an element of the class ‘Additional Field Devices’.

Network configuration

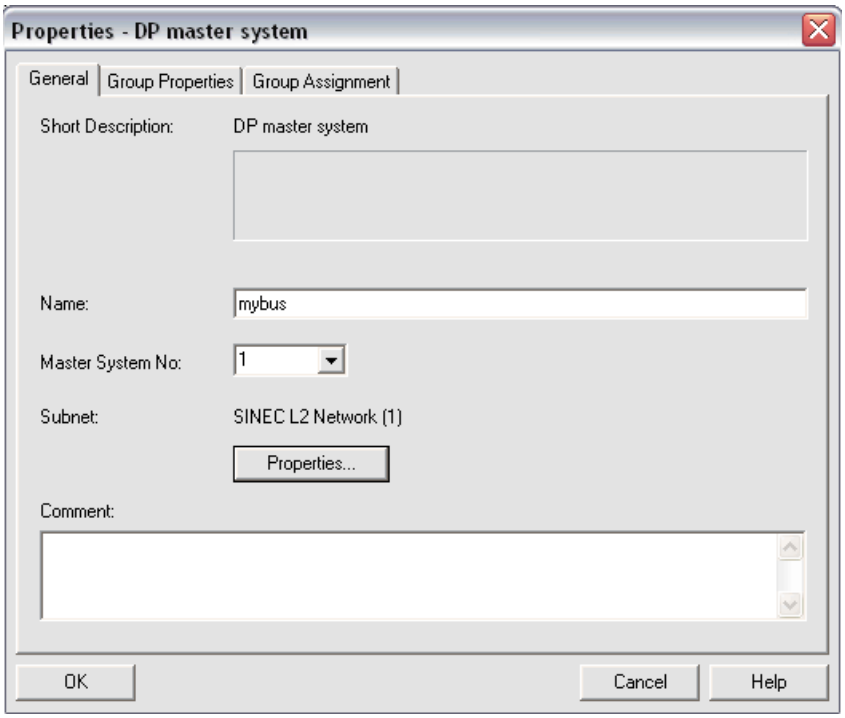
The basic steps of the Profibus DP configuration are described below.

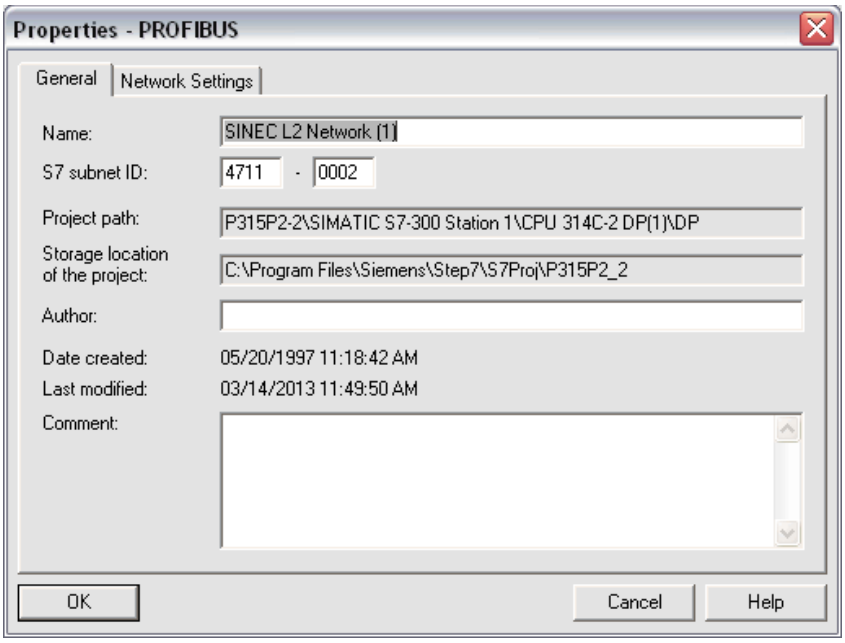
Create a new Step7 project or open an existing project.

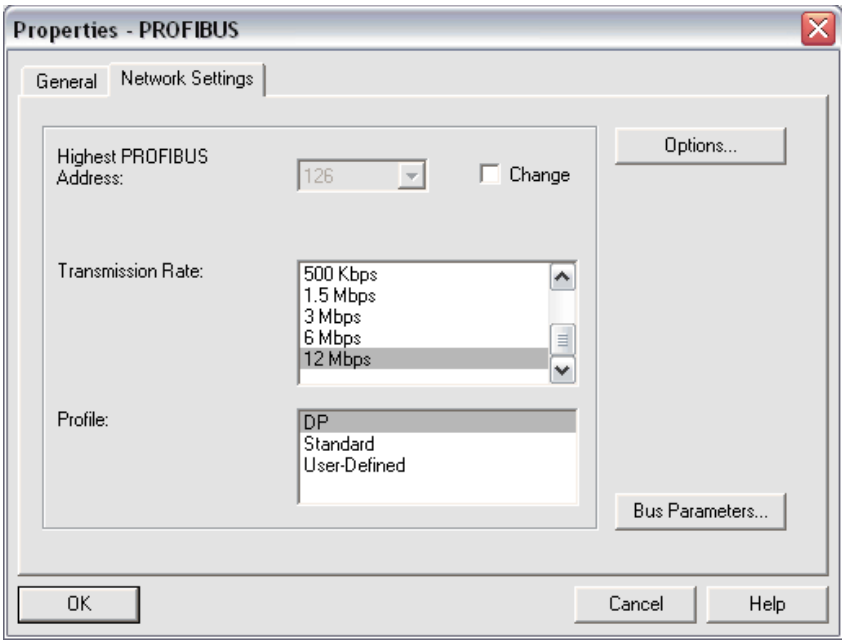
Configure the system (Hardware Configuration) using components from the Hardware Catalog

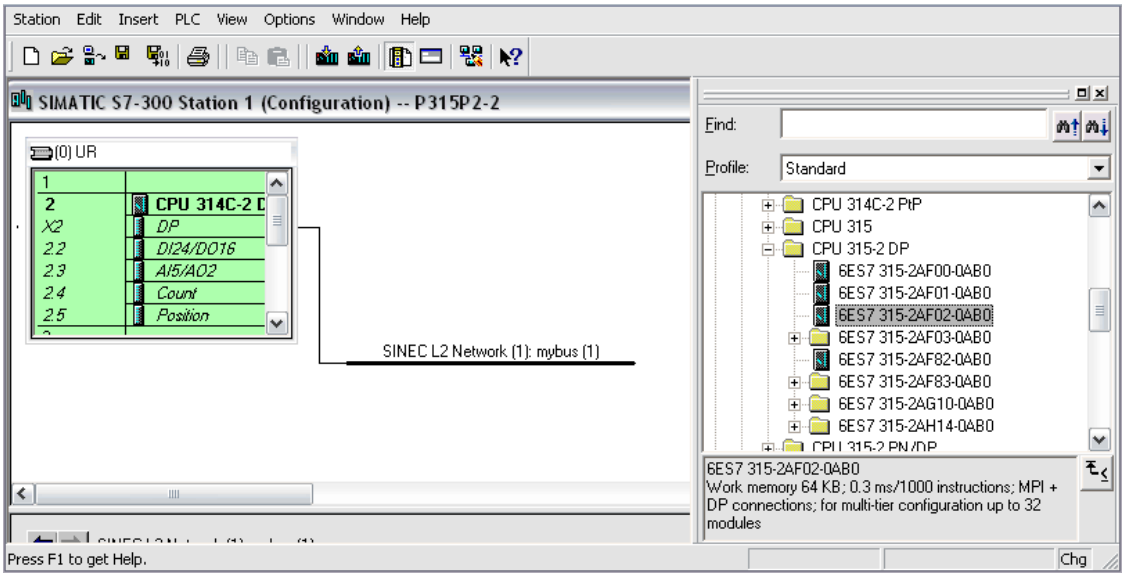


Create and configure the Profibus DP network.

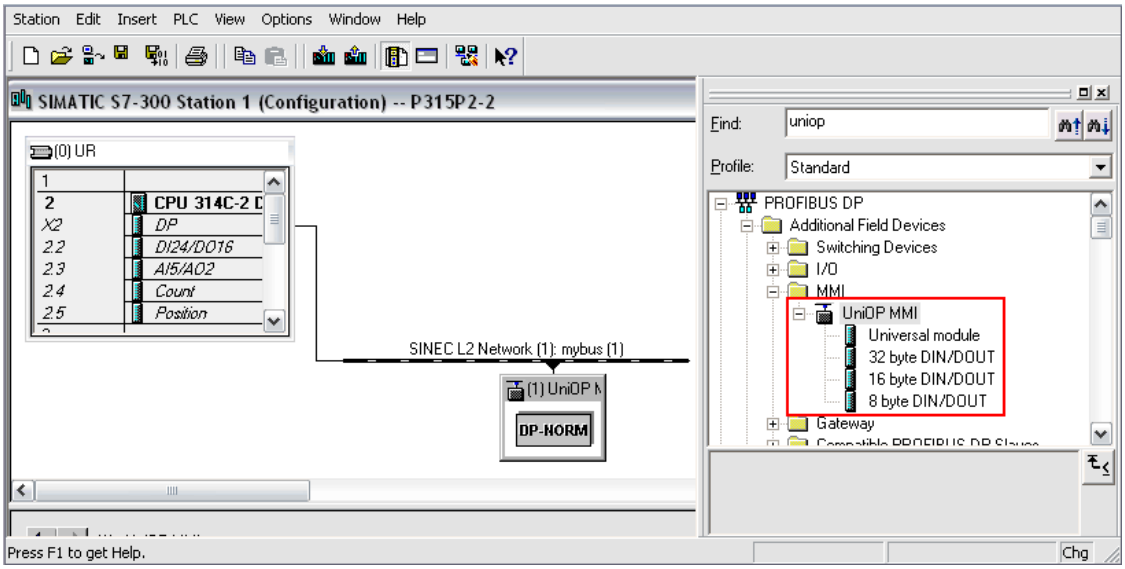



Properties - PROFIBUS
General
Network Settings
Name:
SINEC L2 Network (1)
S7 subnet ID:
4711
0002
Project path:
P315P2-2\SIMATIC S7-300 Station 1\CPU 314C-2 DP(1)\DP
Storage location of the project:
C:\Program Files\Siemens\Step7\S7Proj\P315P2_2
Author:
Date created:
05/20/1997 11:18:42 AM
Last modified:
03/14/2013 11:49:50 AM
Comment:
OK
Cancel
Help

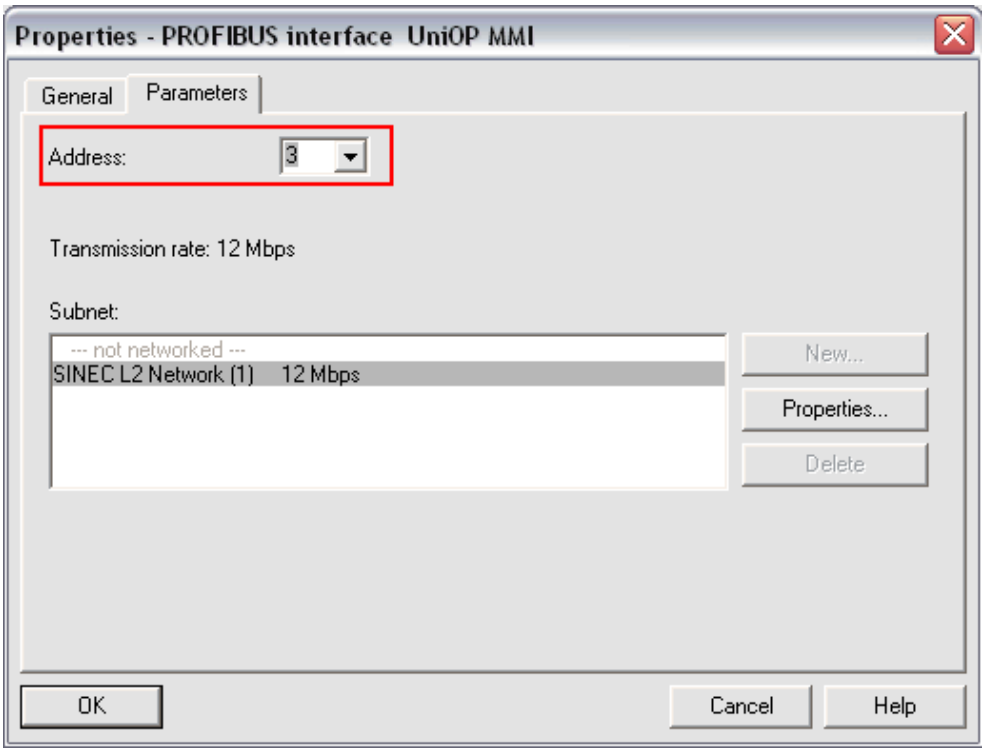

Properties - PROFIBUS
General
Network Settings
Highest PROFIBUS Address:
126
Change
Options...
Transmission Rate:
500 Kbps
1.5 Mbps
3 Mbps
6 Mbps
12 Mbps
Profile:
DP
Standard
User-Defined
Bus Parameters...
OK
Cancel
Help



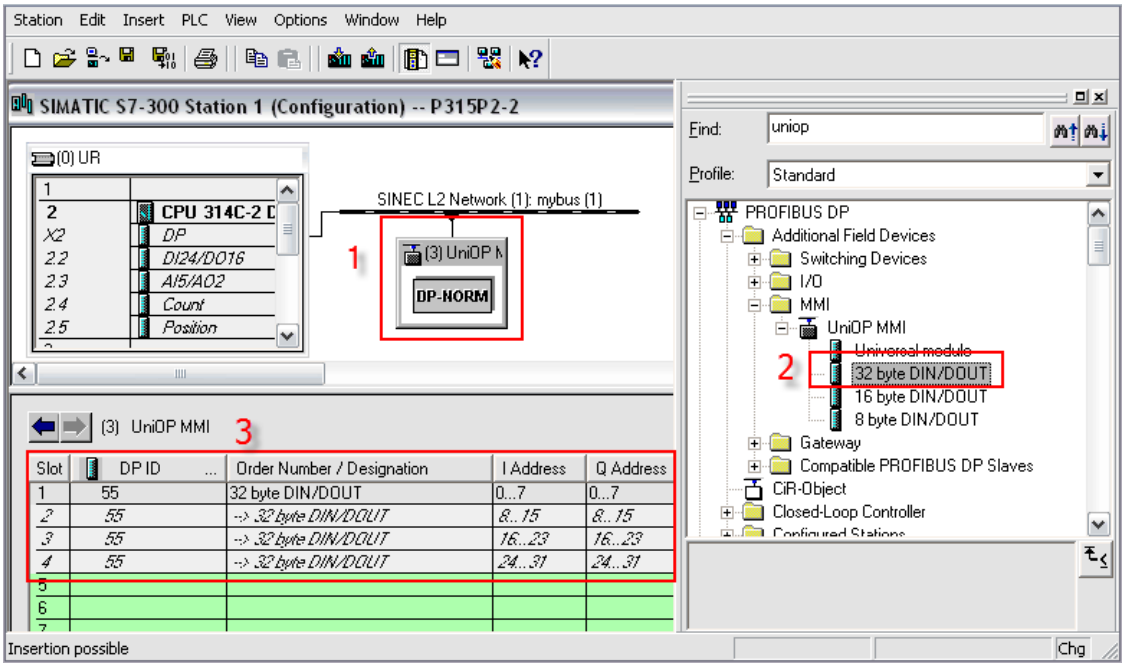
The Operator Panels will then be available for selection in the Hardware Catalog as shown in the figure below. Note that the DDB Files must have been updated as described in chapter 1.1.



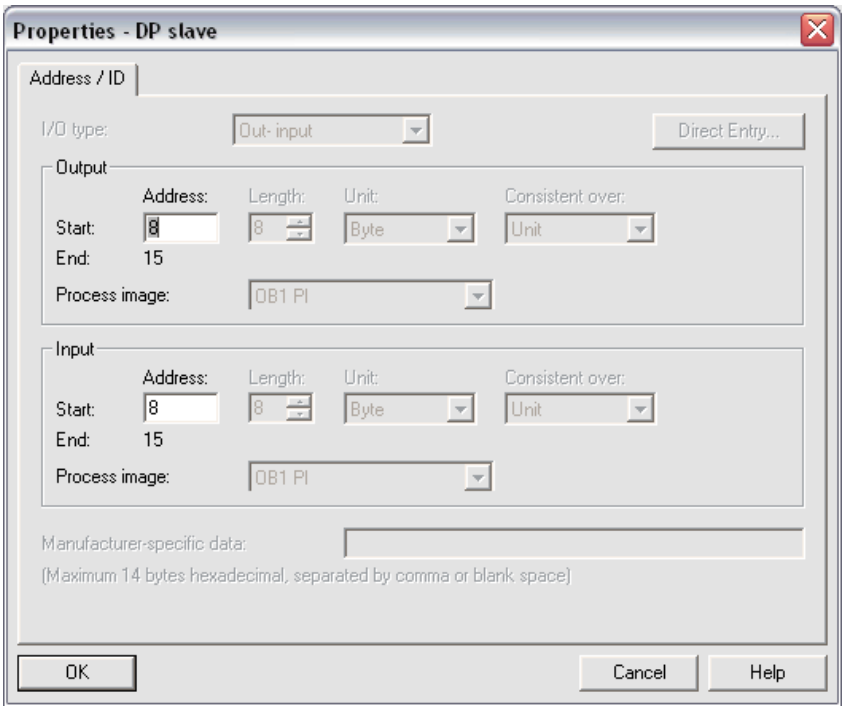
Select the Device from Hardware catalog and Drag & Drop it to the Bus line, once added assign the Address properly

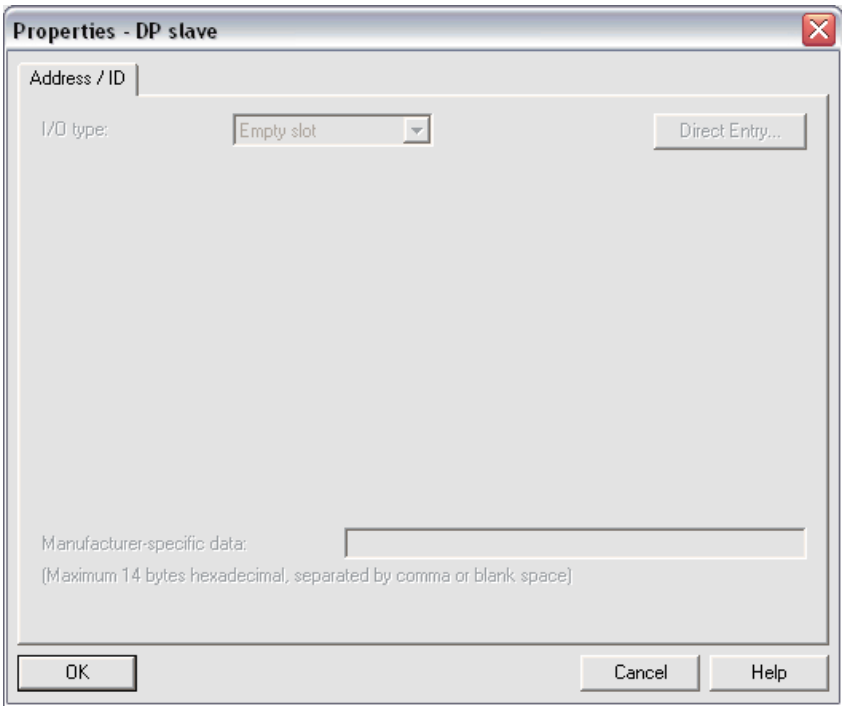


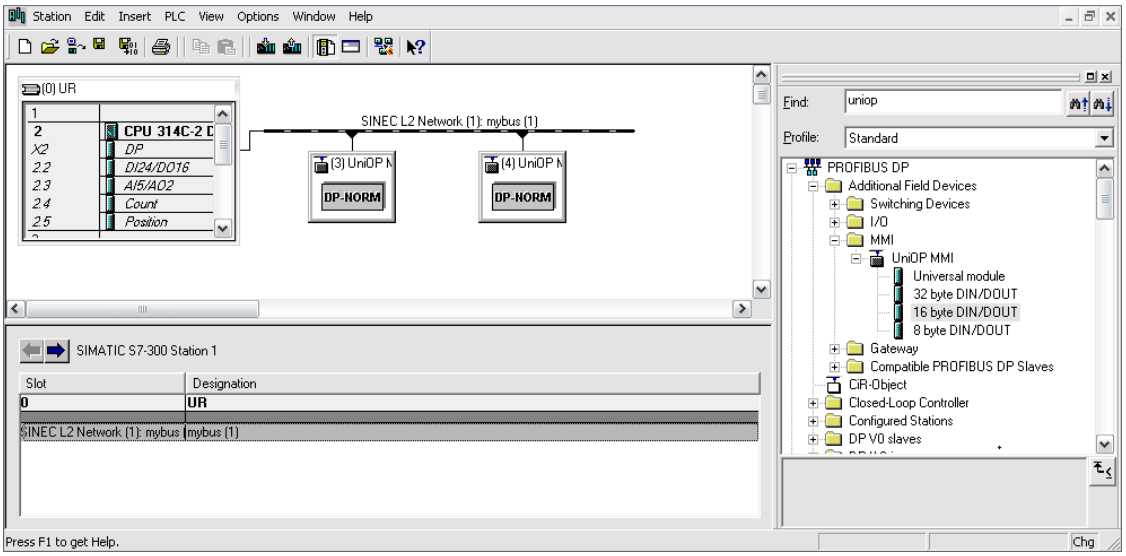
Once the HMI devices have been included in the Profibus DP network configuration, you will have to open the slave configuration and enter the required parameters. 2 or 4 blocks must be configured in the DP image area for the device depending on the size of the buffer (16 or 32 bytes) which has been selected in the previous step. The HMI panels can work with a DP image size of 16 bytes or 32 bytes. Using 32 bytes will offer improved communication performance at the expense of an increased memory usage in the process image area



Configure the blocks in the DP image area. If buffer size of 16 bytes is selected, unused blocks are automatically set to ‘Empty slot’.


Properties - DP slave
Address / ID
I/O type:
Out- input
Direct Entry...
Output
Address:
Length:
Unit:
Consistent over:
Start:
8
8
Byte
Unit
End:
15
Process image:
OB1 PI
Input
Address:
Length:
Unit:
Consistent over:
Start:
8
8
Byte
Unit
End:
15
Process image:
OB1 PI
Manufacturer-specific data:
(Maximum 14 bytes hexadecimal, separated by comma or blank space)
OK
Cancel
Help


Properties - DP slave
Address / ID
I/O type:
Empty slot
Direct Entry...
Manufacturer-specific data:
(Maximum 14 bytes hexadecimal, separated by comma or blank space)
OK
Cancel
Help



The configuration procedure must be repeated for all the HMI devices to be included as slaves in the Profibus DP network. Finally the network configuration will have to be transferred to the master PLC.

Using the function blocks in the master PLC

To make possible for the HMI device to access all the data in the Master PLC, some support from the PLC program is required. It is accomplished by adding to the user PLC application some special program modules. Samples of these program modules required to support Profibus DP communication are available.

The core functionality is provided by one special function block must be added to the user's program. The complete support includes also 2 Data Blocks.

The Function Block and the other blocks are available in the form of ready-to-run sample projects. Function and Data Blocks may be extracted from the sample projects for integration into the user's project.

Apart from adding the special blocks to the PLC program you also need to cyclically call FB1. You can do this by adding a call to FB1 in OB1.

It is important that FB1 is called cyclically; you should not call it only one time, as the function block only processes the requests from the slave devices when it is called.

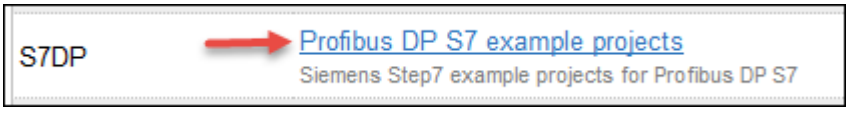
The HMI devices will not be able to communicate with the Master PLC if it is in STOP mode as the special function block will not be called.

Note: if you have multiple HMI devices connected to the Master PLC you do NOT need to call FB1 once for each panel. One call to the FB1 for every cycle of the PLC program is sufficient to process all the HMI slave devices in the Profibus DP network attached to the Master PLC.

Sample PLC programs

Sample PLC programs are available on our website in Software section.

Click on **Profibus DP S7 example projects** to start the download, as shown in picture below.



Creating the comm data block

The Comm DB (Communication Data Block) is used to provide the program modules supporting Profibus DP communication with information on:

- the number of HMI devices configured as Profibus DP slaves and
- the addresses for the Input and Output data of the slave devices in the Master PLCs memory.

The Comm DB has 2 distinct parts; the first part contains information about the configuration of the Profibus DP network of the PLC while the second part contains information about the various HMI devices that are connected to this port. Basically this information is a duplication of the data that you enter in the Profibus Master with Step7.

The Profibus DP Port part is placed in the first 14 bytes of the Comm DB and has the following format:

DBB0	Number of Panels
DBB1	Frame Length
DBB2	Data Type for Input Buffer
DBB3	Data Type for Output Buffer
DBW4	DB Number (Input Buffer)
DBW6	Input Area Base (Input Buffer)
DBW8	DB Number (Output Buffer)
DBW10	Output Area Base (Output Buffer)
DBB12	Sequence Type
DBB13	Reserved for Internal Use

Number of Panels	total number of HMI panels that have to communicate with the Master PLC.
Frame Length	size of the Profibus buffers used to communicate with the Master. Two buffer sizes are supported: 16 bytes and 32 bytes. Enter the appropriate number in this location. Input and Output buffers always have the same size
Data Type for Input Buffer	type of PLC data where the Profibus DP input buffer for the panels is located. **Value** / **Data Type** 0 – DB 4 – I The Input buffer contains the information received by the Master from the slave.
Data Type for Output Buffer	type of PLC data where the Profibus DP output buffer for the panels is located.

Value	Data Type
0	DB
5	Q

The Output buffer contains the information written by the Master to be sent to a Slave.

DB Number (Input Buffer)	if the location specified for the Input Buffer is a DB, enter here the DB number
Input Area Base (Input Buffer)	offset in the Input Buffer where the data for the panels starts.
DB Number (Output Buffer)	if the location specified for the Output Buffer is a DB, enter here the DB number.
Output Area Base (Output Buffer)	offset in the Output Buffer where the data for the panels starts.
Sequence Type	specifies how you want to handle the case of having Number of Panels set to greater than 1. If you set this item to 0 then the function block will process the requests from all the HMI panels before returning. If you set this item to 1 then the function block will process the request from only a single panel before returning, it will then process the request for the next panel on the subsequent call. This means that if Sequence Type is set to 0 the requests from the HMI panels will be processed faster but the execution time of the PLC program will be longer. If the increased execution time of the PLC program causes problems for your application you can set Sequence Type to 1.
Reserved For Internal Use	is actually used to keep track of which panel was processed last. This is used if Sequence Type is set to 1

Note: in this chapter the terms ‘Input’ and ‘Output’ are referred to the Master PLC and not to the slaves. The information entered in this section must be the same entered in the Profibus DP network configuration.

Following on from the header data comes the HMI panel data. The number of HMI panels connected to this port is specified by ‘Number of Panels’. Each HMI panel is assigned 8 bytes in the Comm DB.

Note: each panel included in the Profibus DP network must have its descriptor in the Comm Data Block. All descriptors have to be placed in consecutive memory locations.

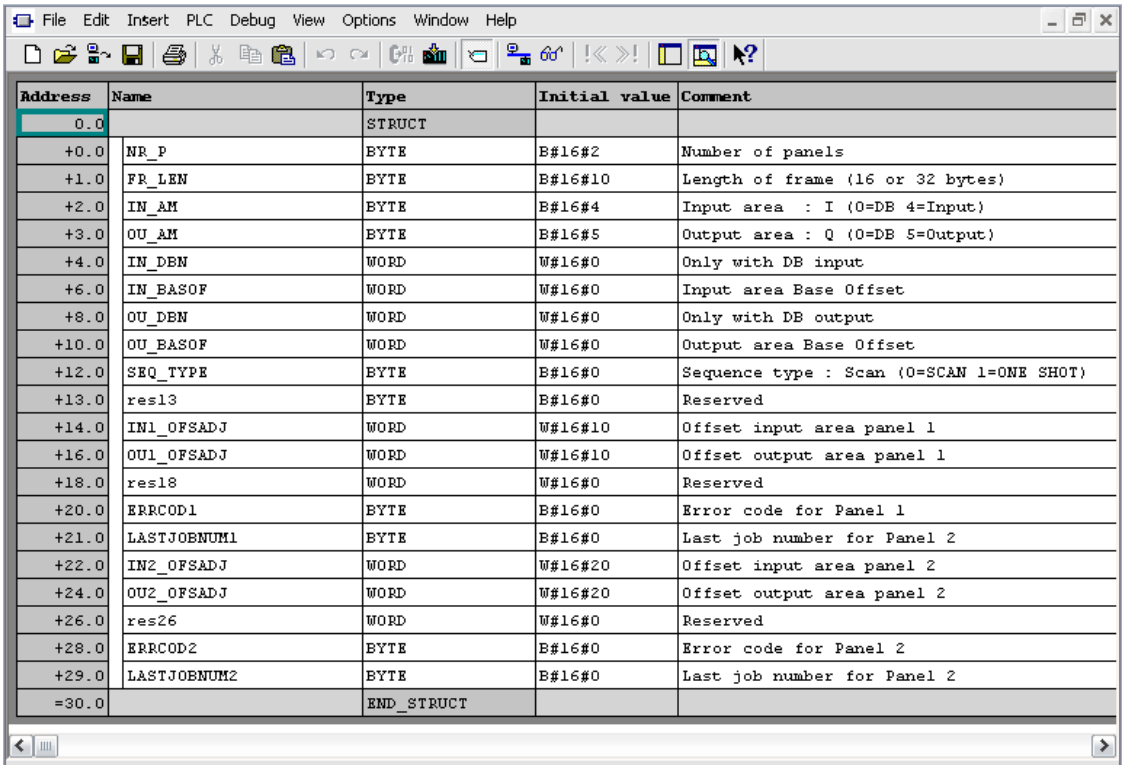
The format of the data block for the individual slave devices has the following format:

DBW14	Input Area Offset
DBW16	Output Area Offset
DBB18	Reserved
DBB19	Reserved
DBB20	Error Code for Last Request
DBB21	Last Job Number

Input Area Offset	this number is added to the Input Area Base (in the header) to obtain the address where this input data for this panel starts
Output Area Offset	this number is added to the Output Area Base (in the header) to obtain the address where this output data for this panel starts
Error Code for Last Request	Error Code for the last communication request for this panel. The error codes have the following meaning: **Error Code** / **Meaning** **0** – No Request Received **1** – Request Processed OK **2** – Request Rejected You do not need to set this element. The function blocks will actually write to this element to give you an indication of the error status of the communication. This field is therefore just for information.
Last Job Number	Job Number for the last communication request for this panel. Every time the HMI panel makes a request it includes the Job Number in the request to the PLC. This Job Number is incremented for every new request. You do not need to set this element. This field is therefore just for information.

Example

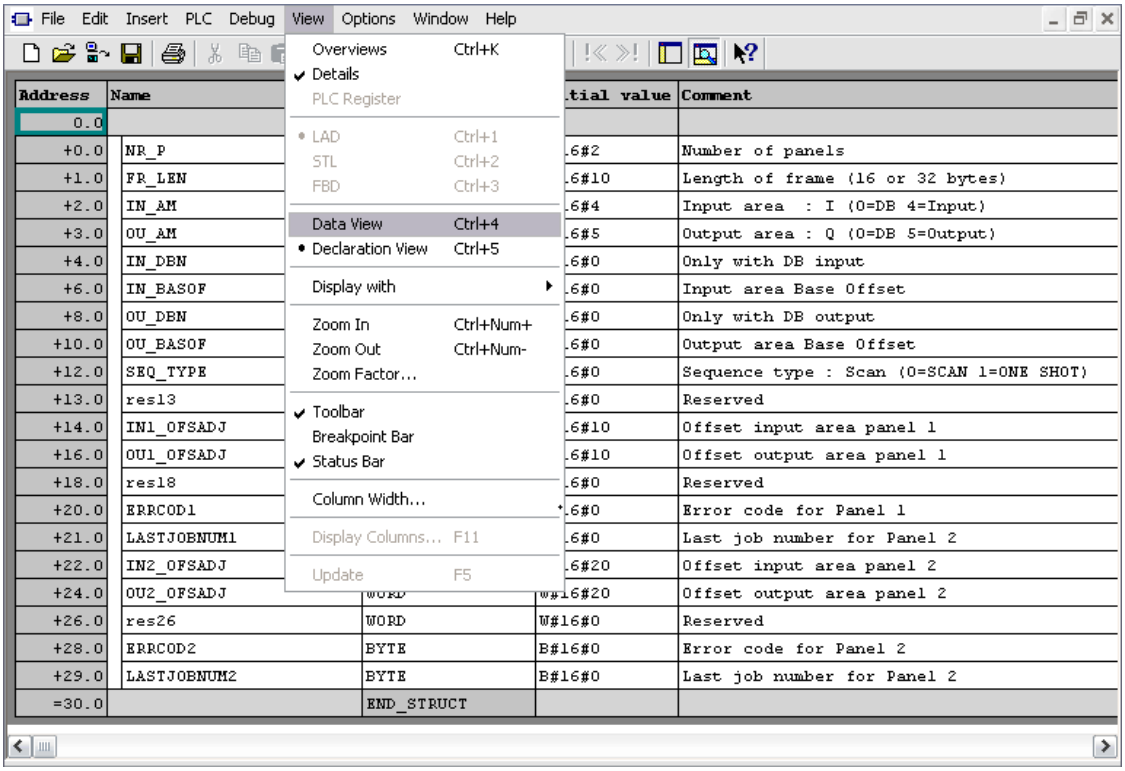
As an example, imagine we have 2 HMI devices attached to a Master PLC that uses I/O addressing and 16 bytes Frame Length. The Input address for the first panel is set to IB16 and the Output address to QB16. The Input address for the second panel is set to IB32 and the Output address to QB32. The Comm DB would take the following form:

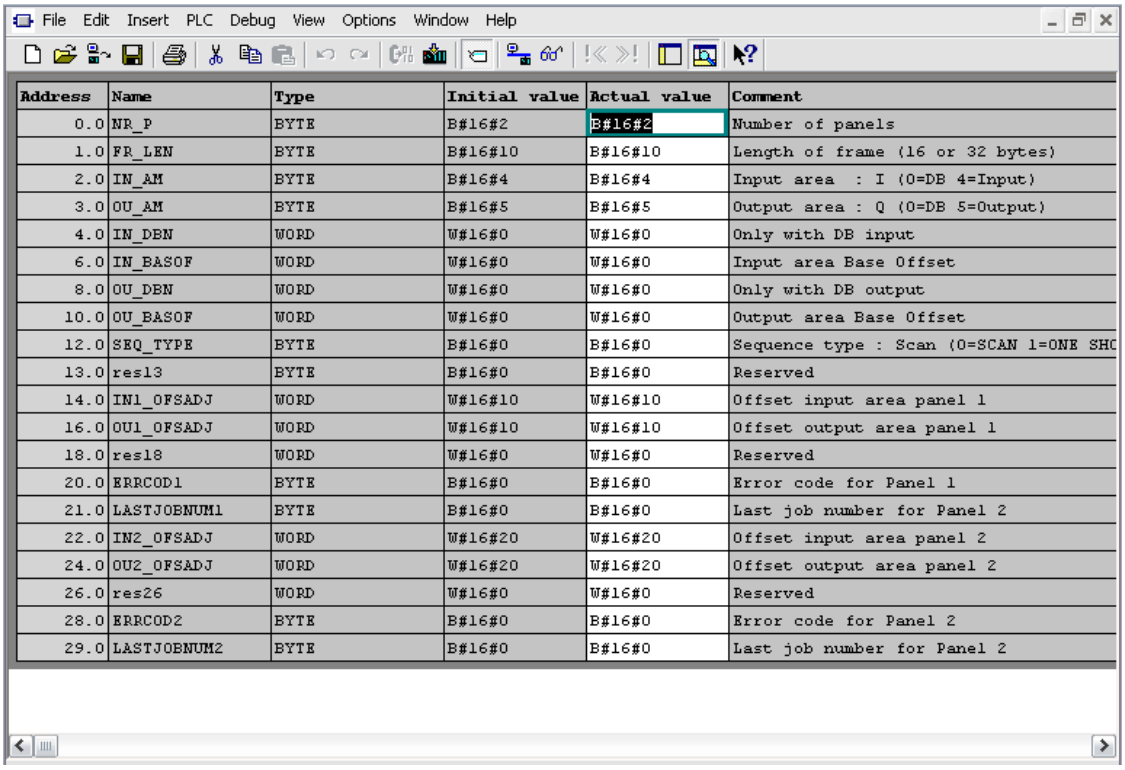


Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	NR_P	BYTE	B#16#2	Number of panels
+1.0	FR_LEN	BYTE	B#16#10	Length of frame (16 or 32 bytes)
+2.0	IN_AM	BYTE	B#16#4	Input area : I (0=DB 4=Input)
+3.0	OU_AM	BYTE	B#16#5	Output area : Q (0=DB 5=Output)
+4.0	IN_DBN	WORD	W#16#0	Only with DB input
+6.0	IN_BASOF	WORD	W#16#0	Input area Base Offset
+8.0	OU_DBN	WORD	W#16#0	Only with DB output
+10.0	OU_BASOF	WORD	W#16#0	Output area Base Offset
+12.0	SEQ_TYPE	BYTE	B#16#0	Sequence type : Scan (0=SCAN 1=ONE SHOT)
+13.0	res13	BYTE	B#16#0	Reserved
+14.0	IN1_OFSADJ	WORD	W#16#10	Offset input area panel 1
+16.0	OU1_OFSADJ	WORD	W#16#10	Offset output area panel 1
+18.0	res18	WORD	W#16#0	Reserved
+20.0	ERRCOD1	BYTE	B#16#0	Error code for Panel 1
+21.0	LASTJOBNUM1	BYTE	B#16#0	Last job number for Panel 2
+22.0	IN2_OFSADJ	WORD	W#16#20	Offset input area panel 2
+24.0	OU2_OFSADJ	WORD	W#16#20	Offset output area panel 2
+26.0	res26	WORD	W#16#0	Reserved
+28.0	ERRCOD2	BYTE	B#16#0	Error code for Panel 2
+29.0	LASTJOBNUM2	BYTE	B#16#0	Last job number for Panel 2
=30.0		END_STRUCT		

When you download a data block to the PLC or when you modify any values in the data block, you will have to make sure that the modified values are also the current values into the PLC memory. To do this you should change the viewing mode of the data block in the Step7 software from “Declaration View” to “Data View” as shown in the next figure.

When you are in Data View mode, the values in the column ‘Actual Value’ must match the values on the column ‘Initial Value’. If there are some differences you have to correct the wrong value on the ‘Actual Value’ column and download again the Data Block to the PLC. The ‘Actual Value’ column displays at any time the actual PLC data values.





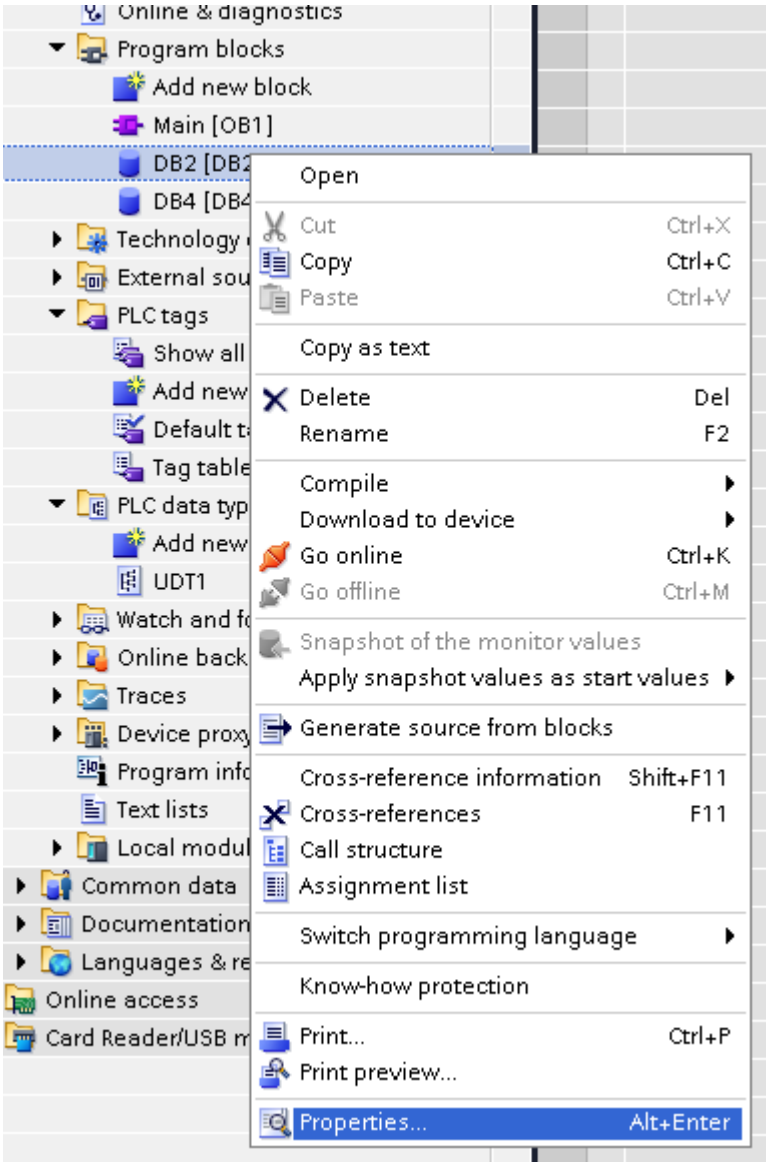
Address	Name	Type	Initial value	Actual value	Comment
0.0	NR_P	BYTE	B#16#2	B#16#2	Number of panels
1.0	FR_LEN	BYTE	B#16#10	B#16#10	Length of frame (16 or 32 bytes)
2.0	IN_AM	BYTE	B#16#4	B#16#4	Input area : I (0=DB 4=Input)
3.0	OU_AM	BYTE	B#16#5	B#16#5	Output area : Q (0=DB 5=Output)
4.0	IN_DBN	WORD	W#16#0	W#16#0	Only with DB input
6.0	IN_BASOF	WORD	W#16#0	W#16#0	Input area Base Offset
8.0	OU_DBN	WORD	W#16#0	W#16#0	Only with DB output
10.0	OU_BASOF	WORD	W#16#0	W#16#0	Output area Base Offset
12.0	SEQ_TYPE	BYTE	B#16#0	B#16#0	Sequence type : Scan (0=SCAN 1=ONE SHO
13.0	res13	BYTE	B#16#0	B#16#0	Reserved
14.0	IN1_OFSADJ	WORD	W#16#10	W#16#10	Offset input area panel 1
16.0	OU1_OFSADJ	WORD	W#16#10	W#16#10	Offset output area panel 1
18.0	res18	WORD	W#16#0	W#16#0	Reserved
20.0	ERRCOD1	BYTE	B#16#0	B#16#0	Error code for Panel 1
21.0	LASTJOBNUM1	BYTE	B#16#0	B#16#0	Last job number for Panel 2
22.0	IN2_OFSADJ	WORD	W#16#20	W#16#20	Offset input area panel 2
24.0	OU2_OFSADJ	WORD	W#16#20	W#16#20	Offset output area panel 2
26.0	res26	WORD	W#16#0	W#16#0	Reserved
28.0	ERRCOD2	BYTE	B#16#0	B#16#0	Error code for Panel 2
29.0	LASTJOBNUM2	BYTE	B#16#0	B#16#0	Last job number for Panel 2

Direct Import of TIA Portal project

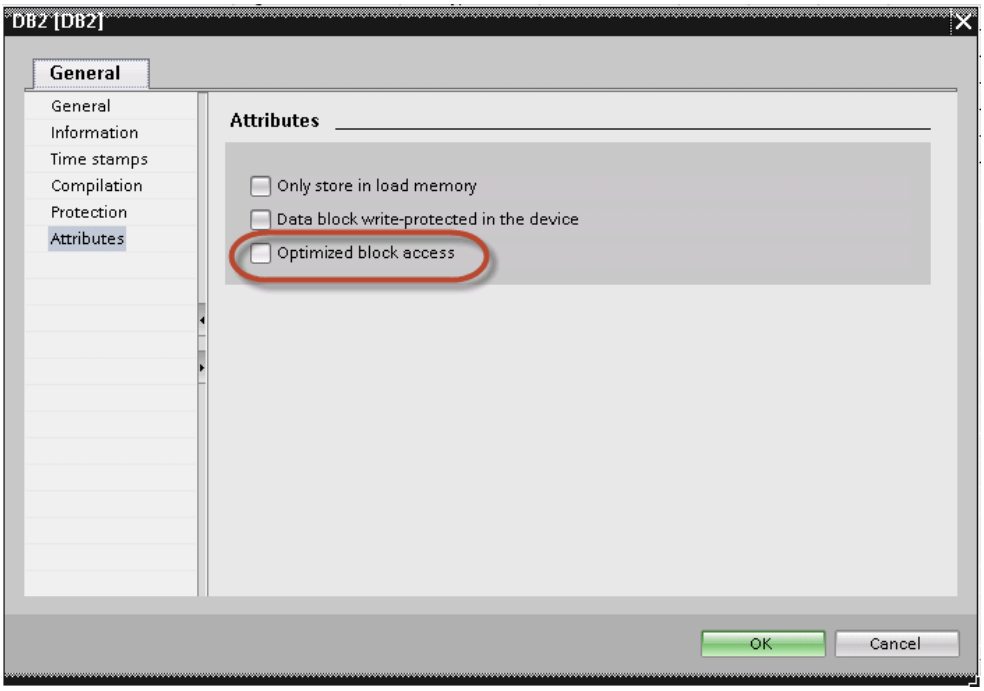
It is possible to import TIA Portal variables directly from TIA Portal project, by selecting "TIA Portal Project v12 or newer" from import selection (refer to "Tag Import" chapter).

Data Blocks must be set as Not optimized:

1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:



3. In the **General** tab select **Attributes** and unselect **Optimized block access**.



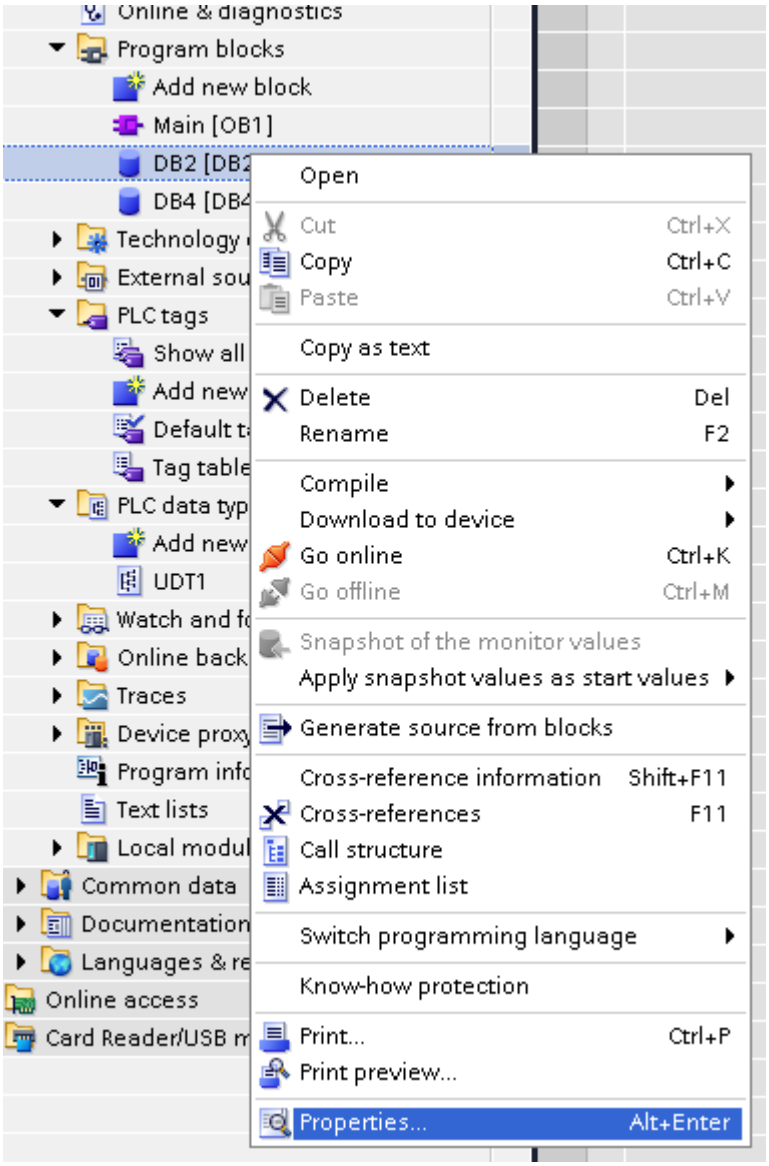
Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

Export using TIA Portal v13, v14 or newer

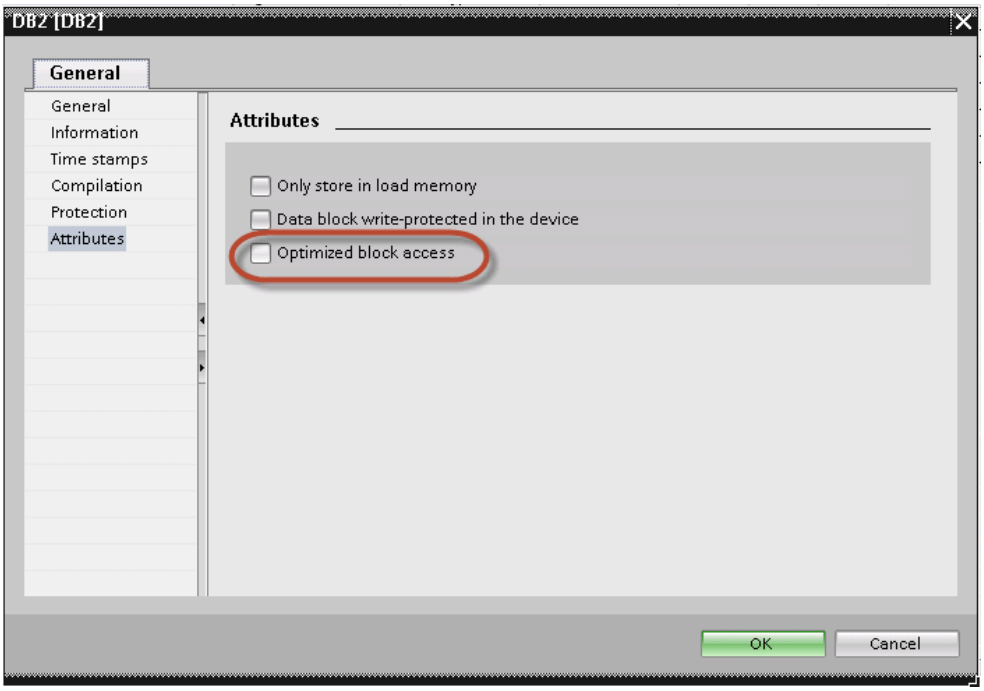
Exporting Program blocks

These files refer to DB tags defined in **Program blocks**.

1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:

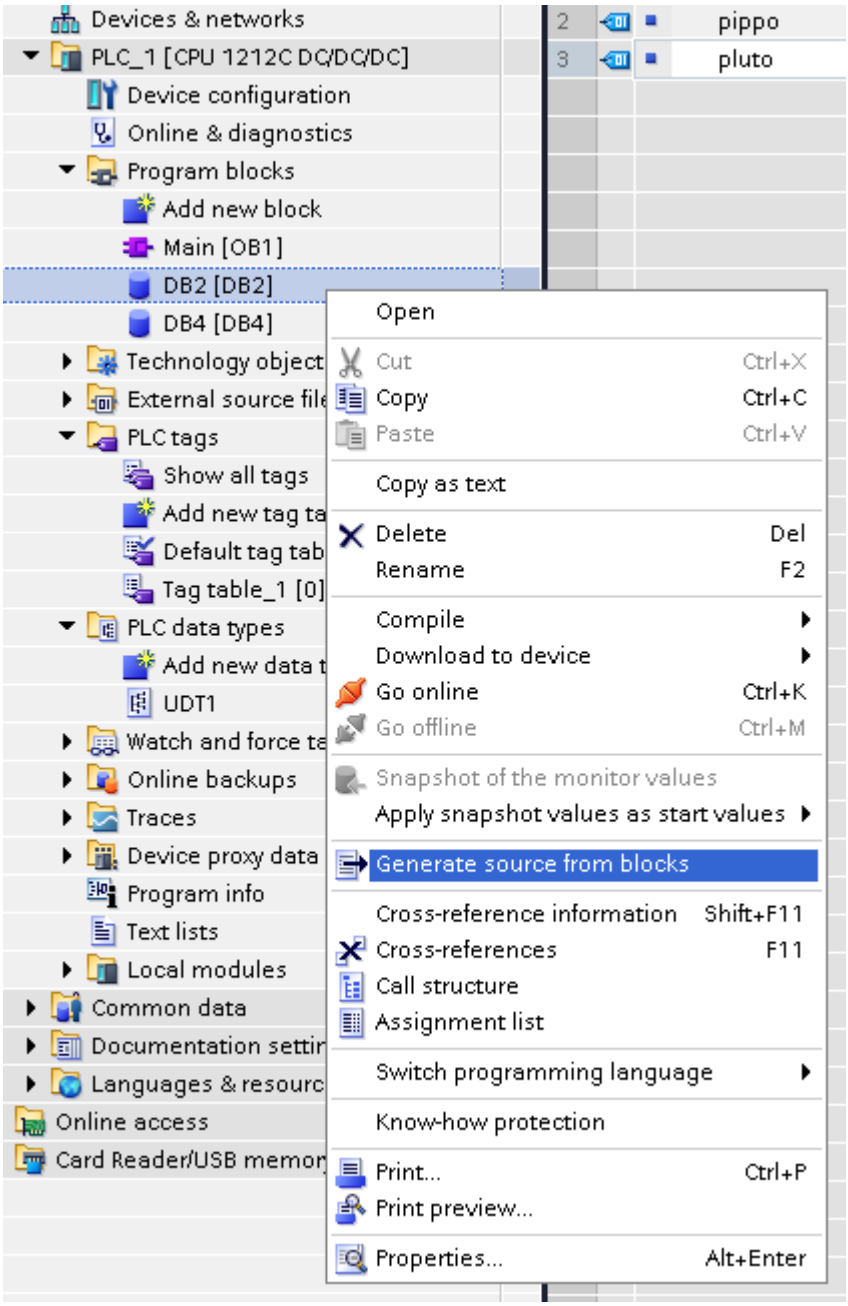


3. In the **General** tab select **Attributes** and unselect **Optimized block access**.

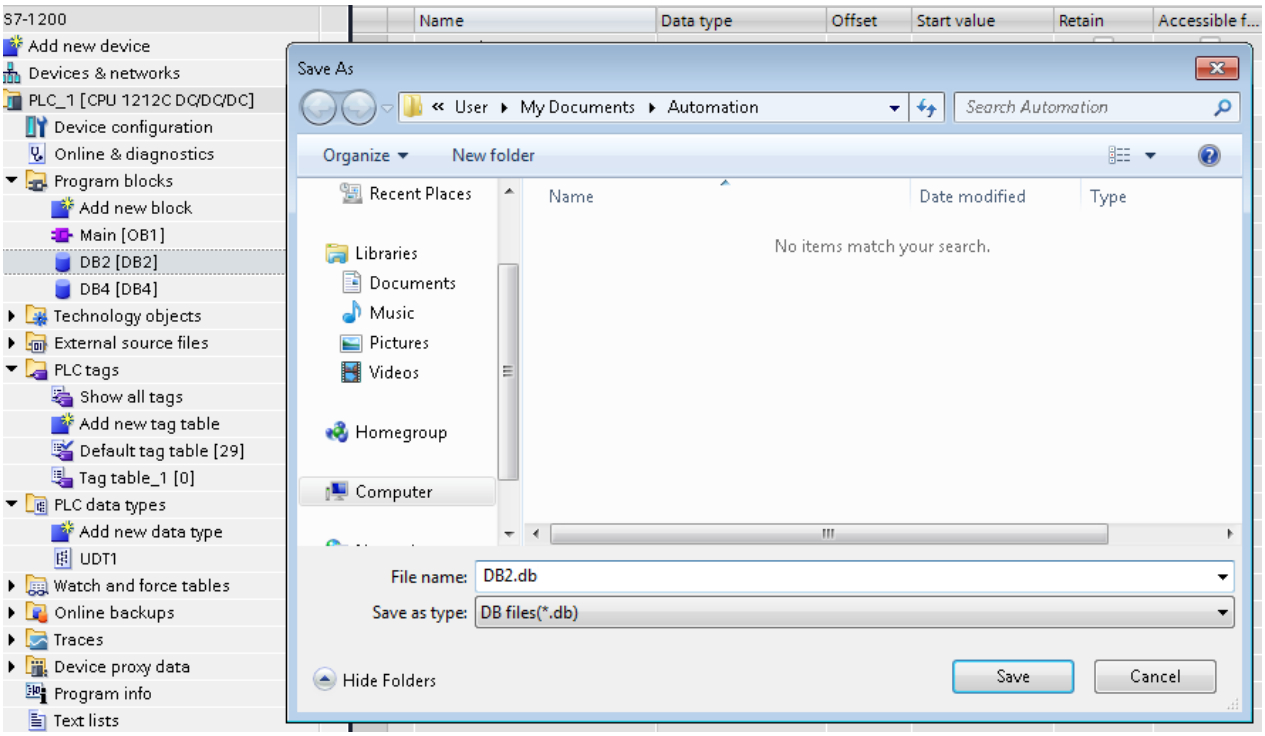


Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

4. Right-click on the Data Block and choose **Generate source from blocks**:



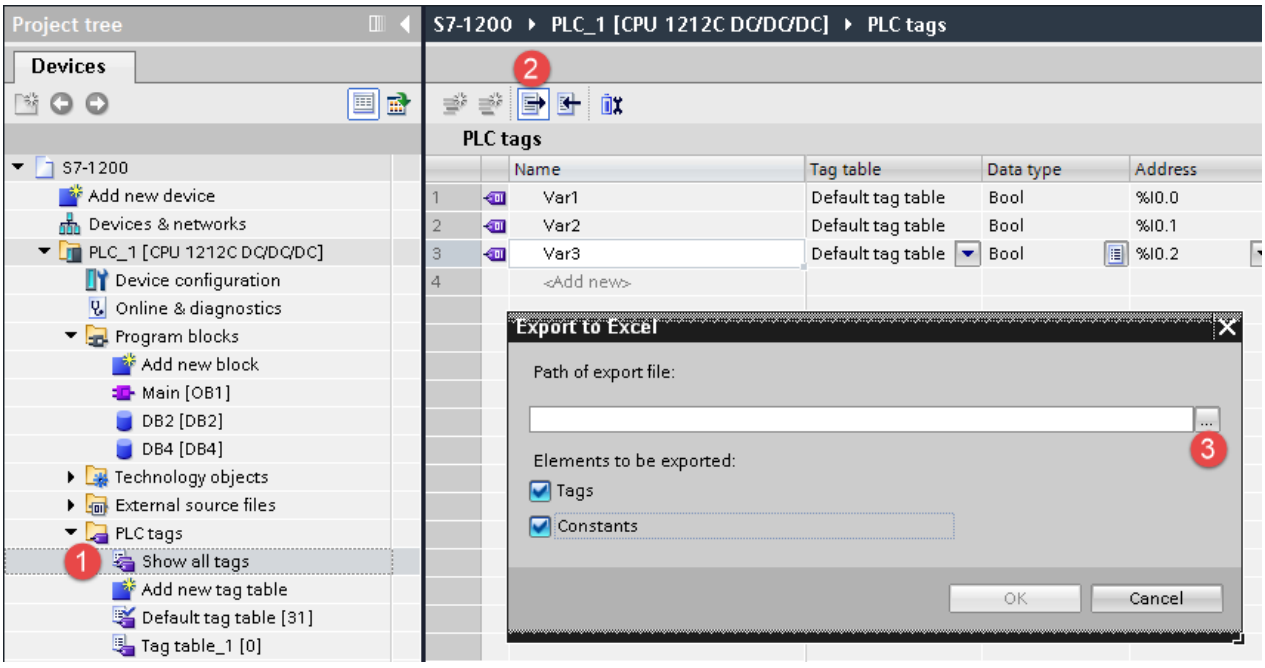
5. Save the file as DBxxx.db, where xxx=number of DB.



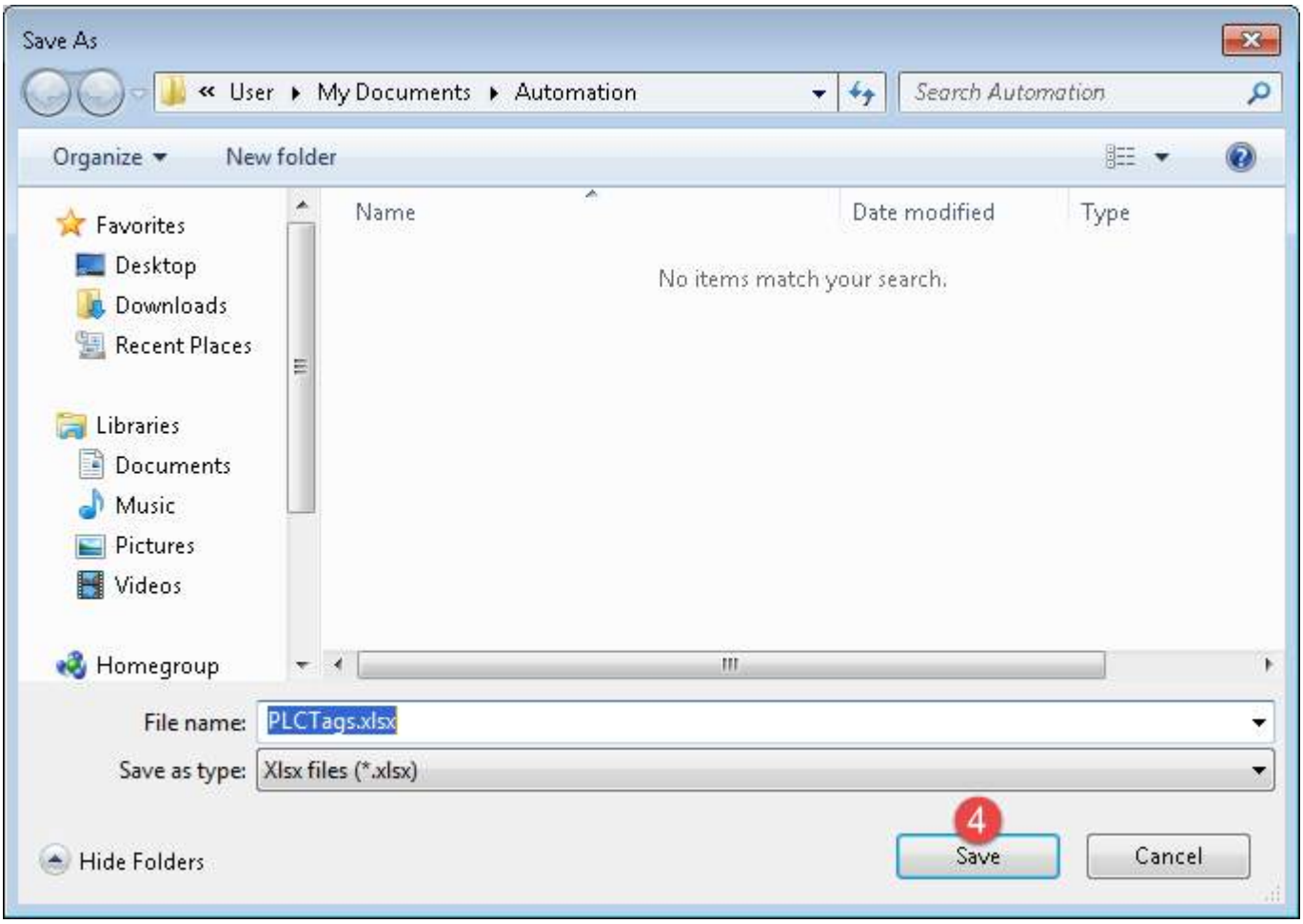
Exporting PLC tags

An Excel file refers to PLC tags.

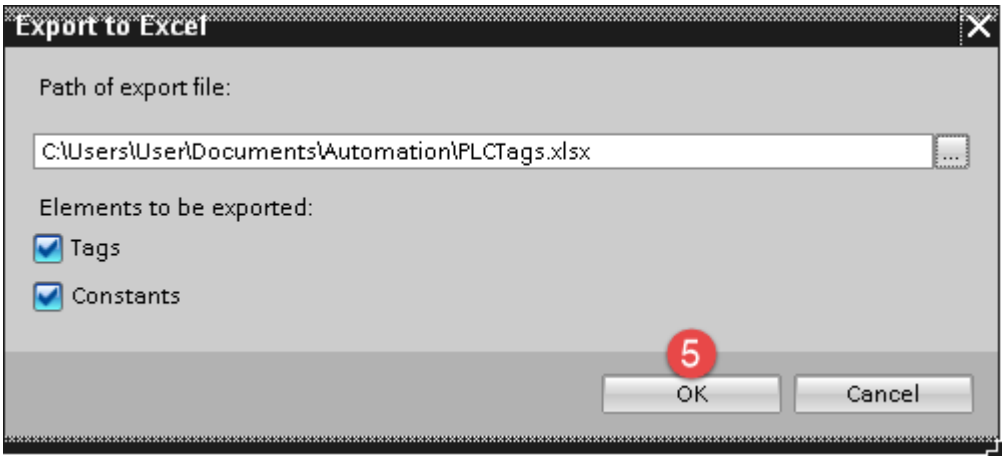
1. Double-click **Show all tags**: the tag table is displayed.
2. Click the **Export** button and browse for path file.
3. Define file name.



4. Click **Save** to confirm.

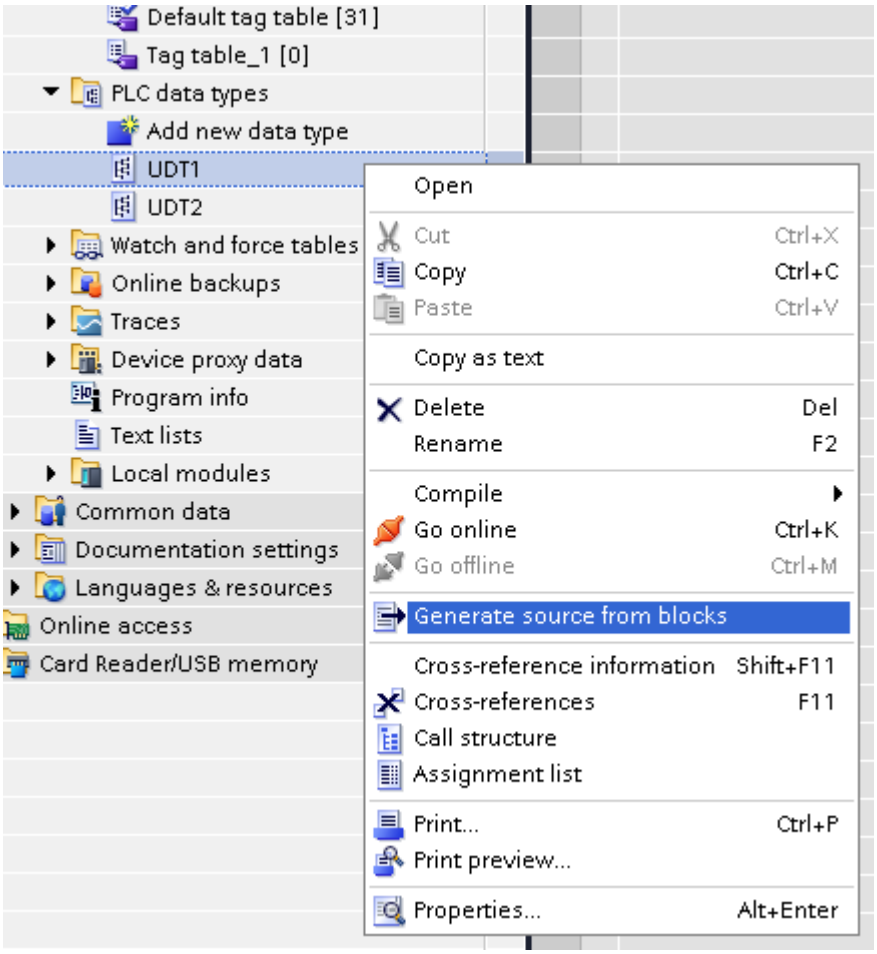


5. Click **OK** to export.

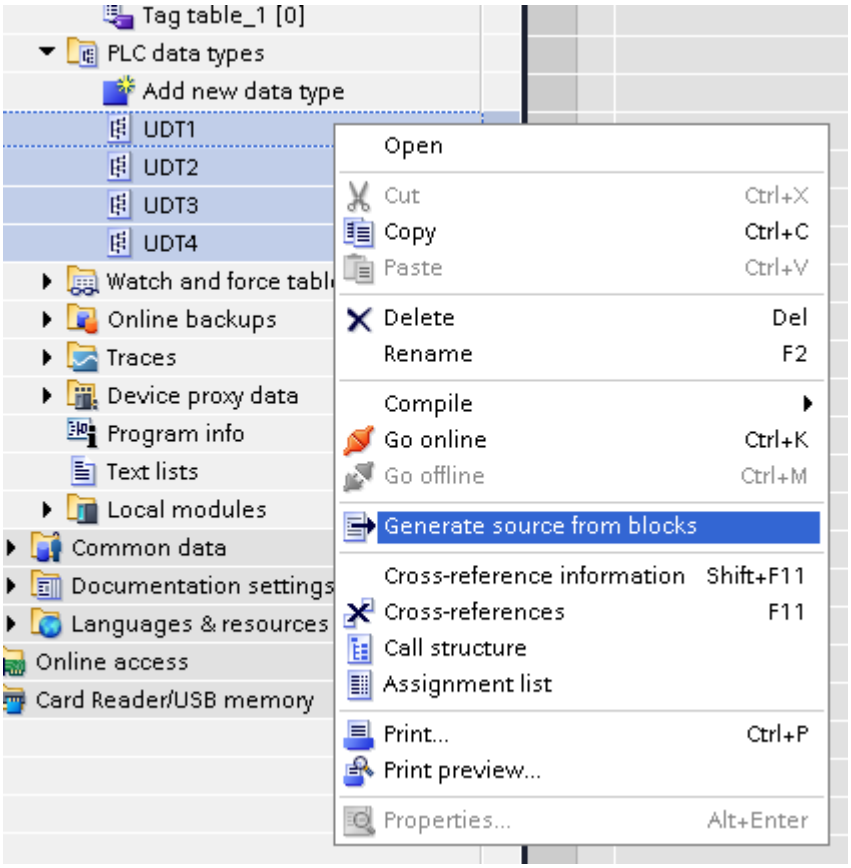


Exporting PLC data types

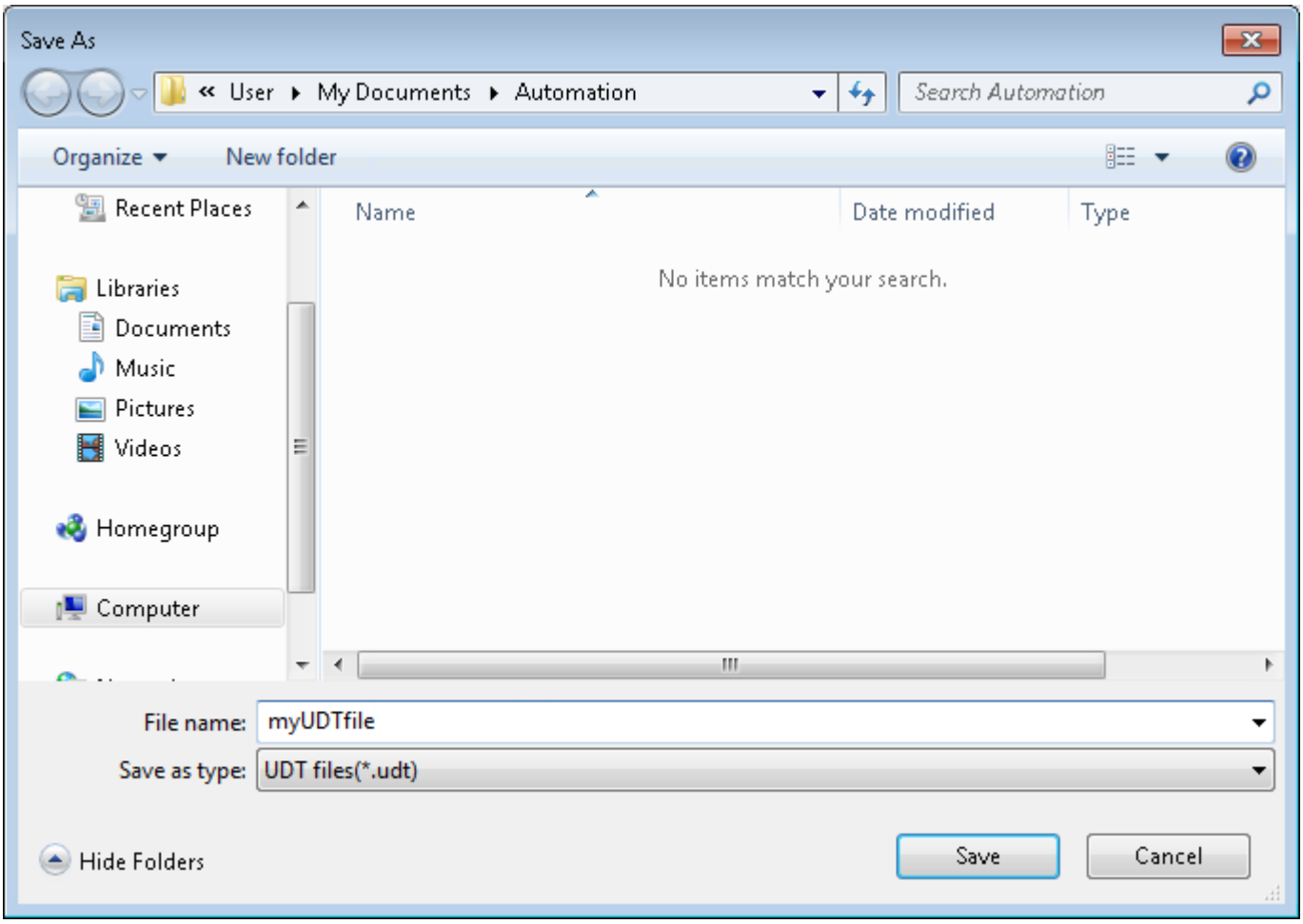
To create the file, expand **PLC data types** item from TIA Portal project tree and right click on the user defined structure. Then click on **Generate source from blocks**.



In case of multiple PLC data types in PLC project, it is necessary to select them all from **PLC data types** list, right click and select **Generate source from blocks** to create the .UDT file that contains all the PLC data types defined.



In the next step, give a name to the .UDT file and choose the path to where to save the file.



This file will content all the PLC data types and it can be used for importing tags in Tag Editor.

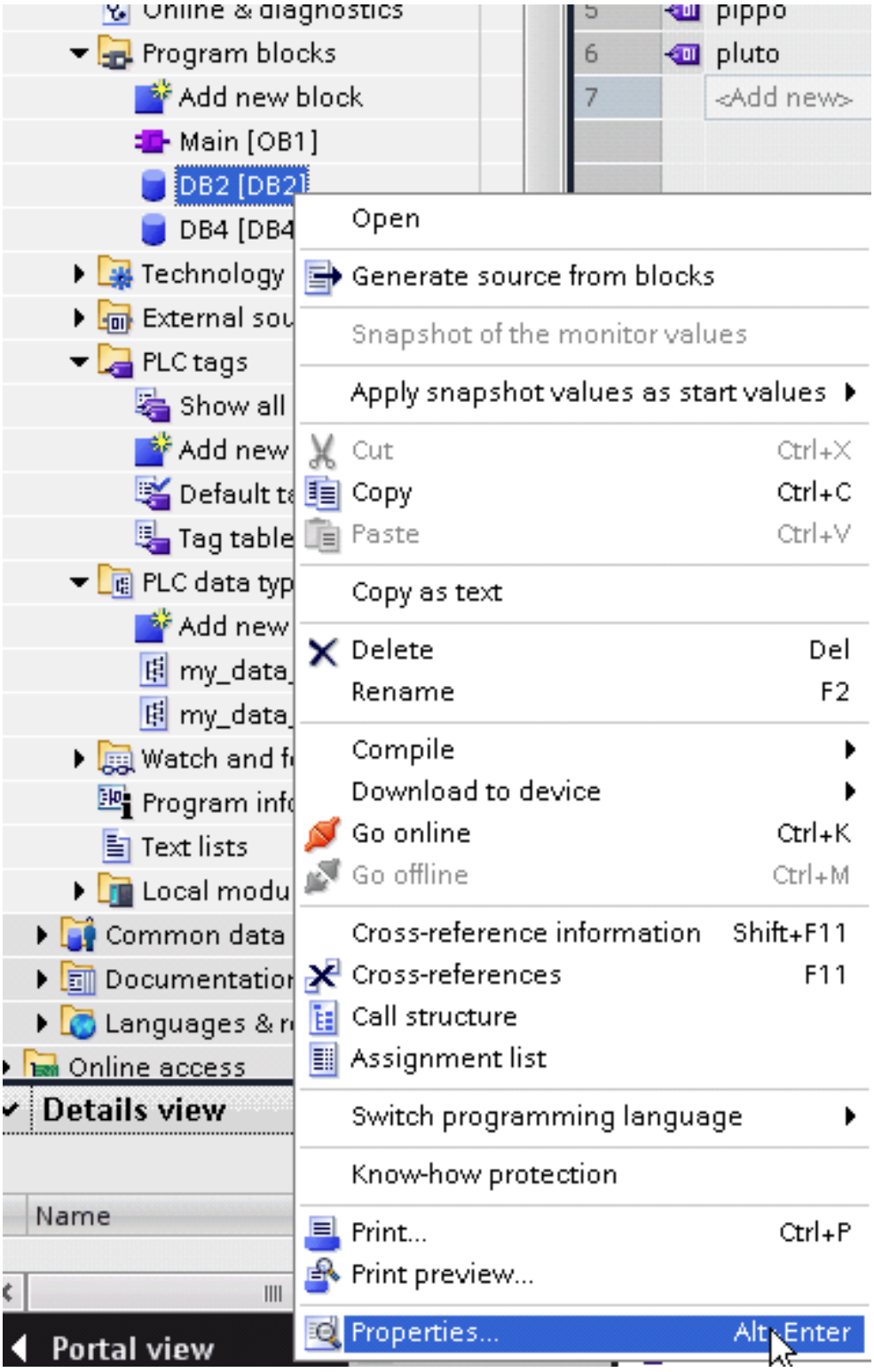
Check **Tag Import** chapter for more details.

Export using TIA Portal v10, v11, v12

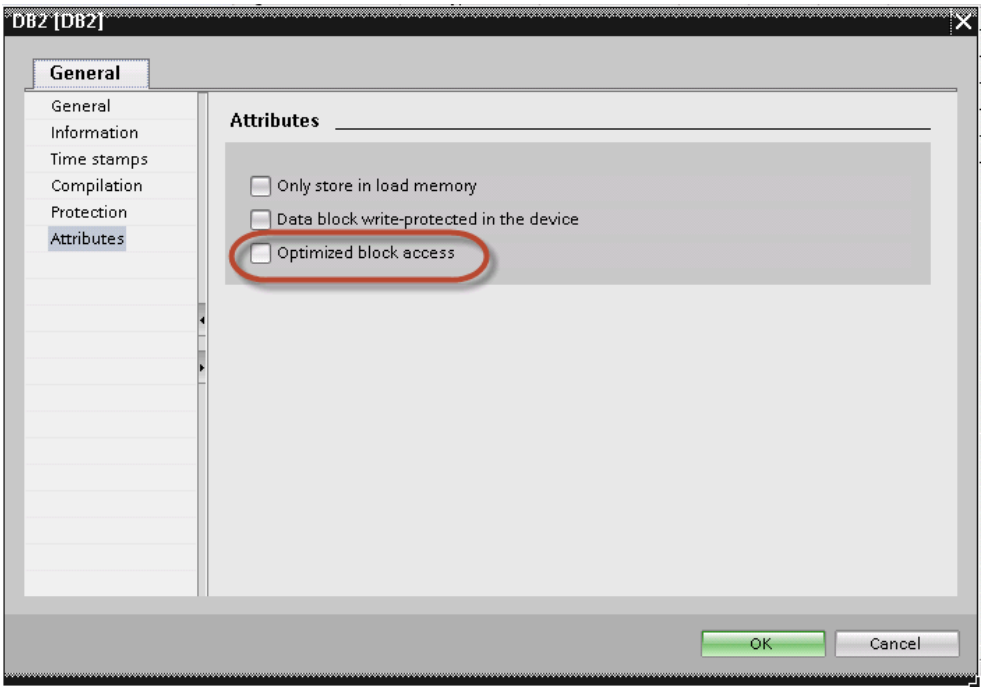
Exporting Program blocks

These files refer to DB tags defined in **Program blocks**.

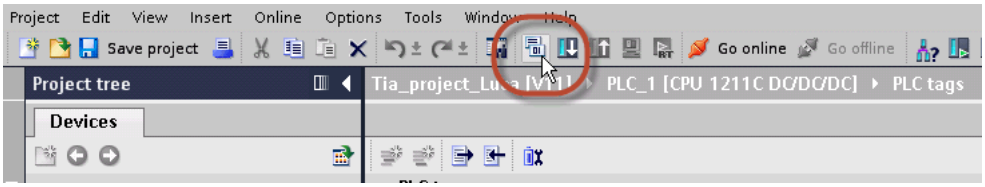
1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:



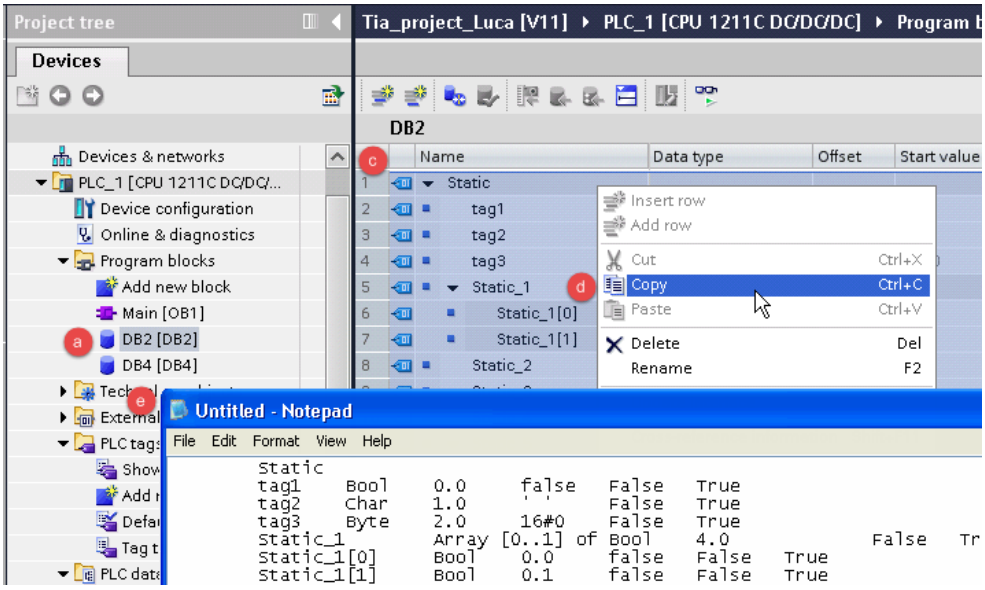
3. In the **General** tab select **Attributes** and unselect **Optimized block access**.



Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".



4. Build the project to make sure TIA Portal calculates the tags offset.

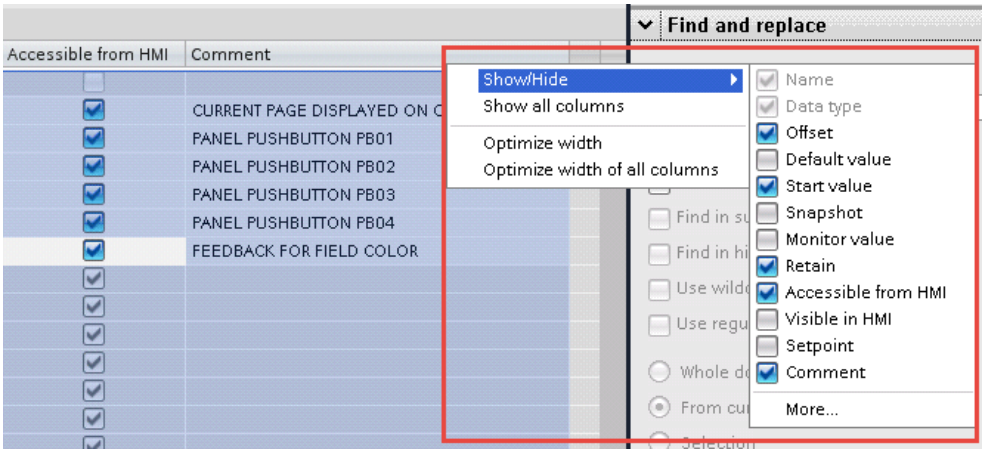


5. Double-click on a DB name.
6. Expand the view of program block selected.
7. Select all rows.
8. Copy and paste into any text editor.
9. Save the file as DBxxx.tia, where xxx=number of DB.

Note: Make sure you use the **Save As** function or the file will be named DB2.tia.txt and will not be visible from the importer.

10. Repeat from step 5 for all program blocks.

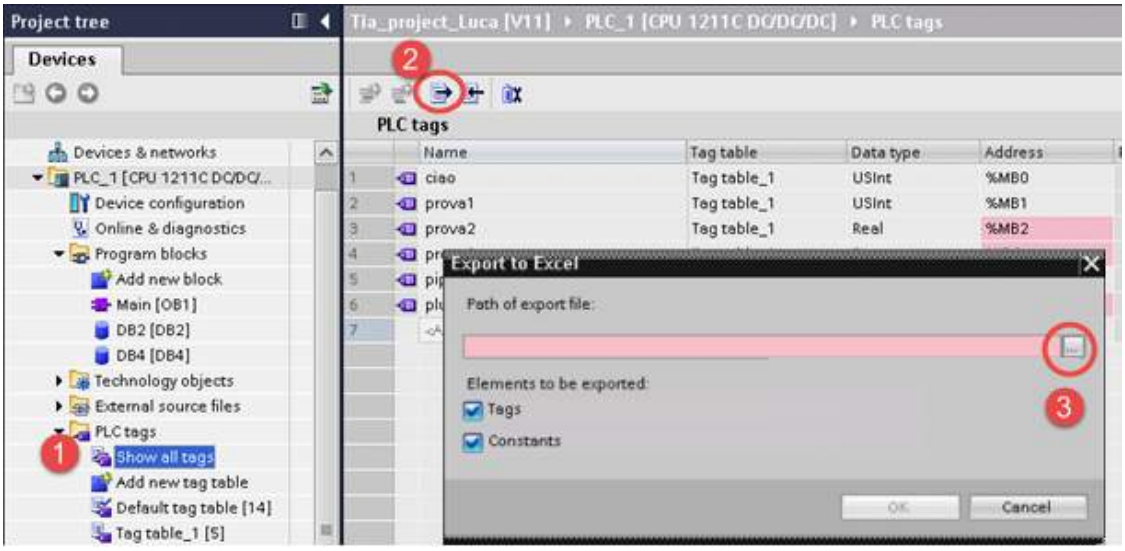
Note: Make sure that only the following columns are shown in DB editor before copying all data in the txt file



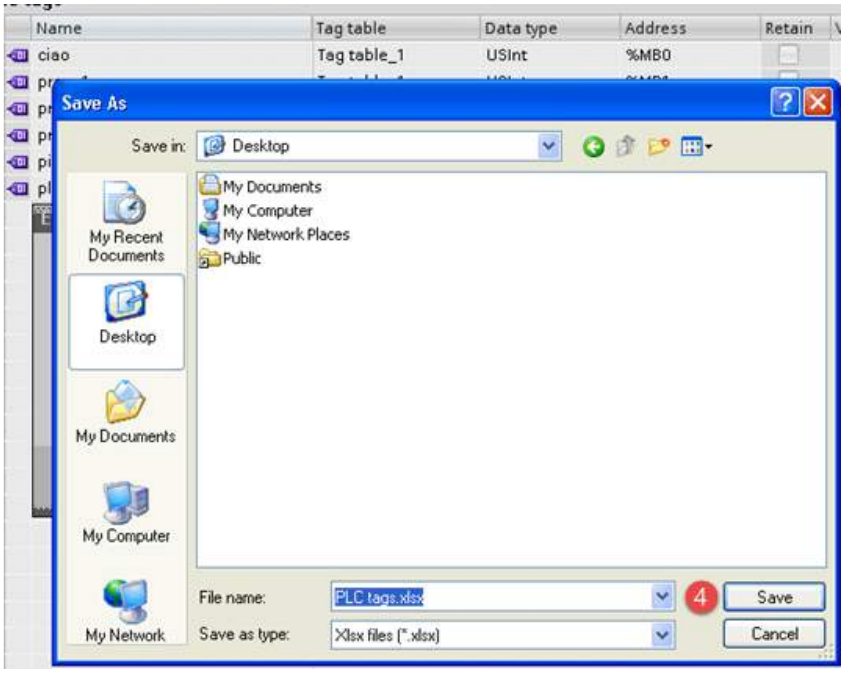
Exporting PLC tags

An Excel file refers to PLC tags.

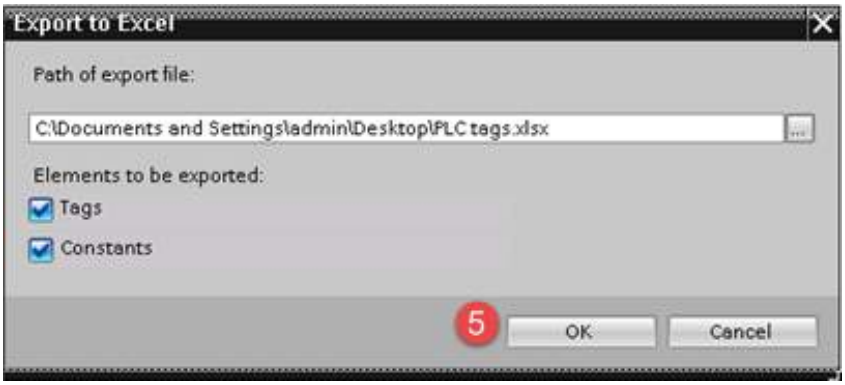
1. Double-click **Show all tags**: the tag table is displayed.



2. Click the **Export** button and browse for path file.
3. Define file name.
4. Click **Save** to confirm.

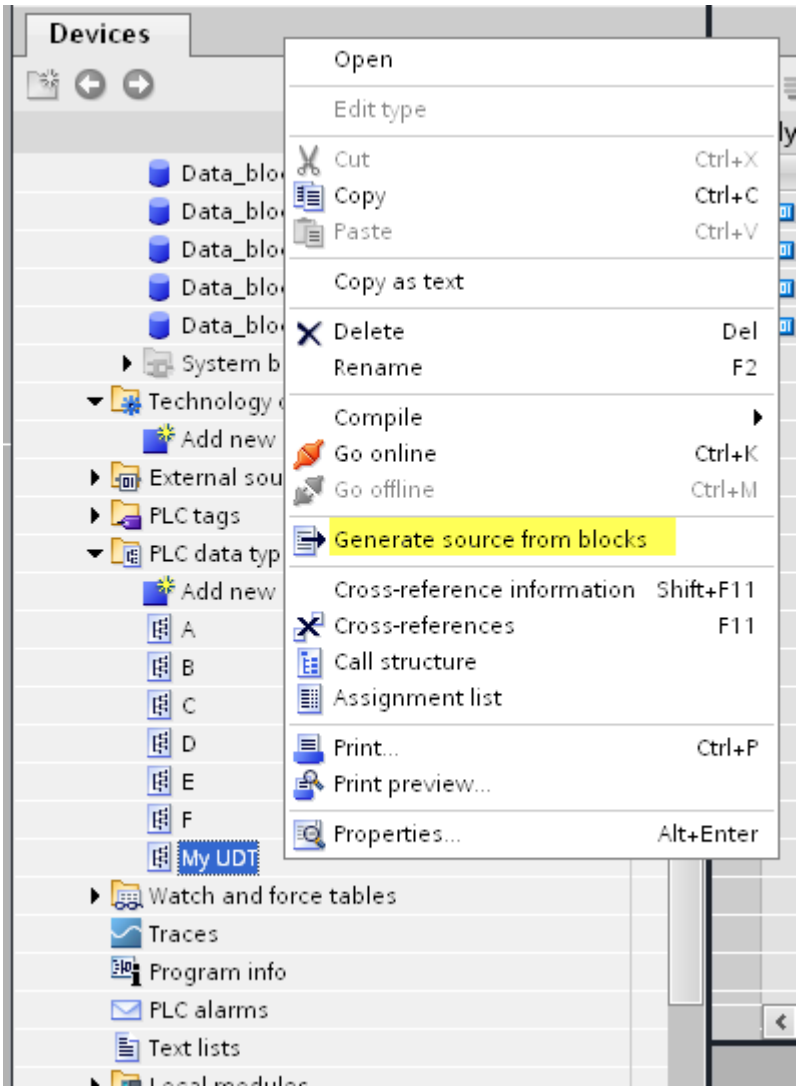


5. Click **OK** to export.

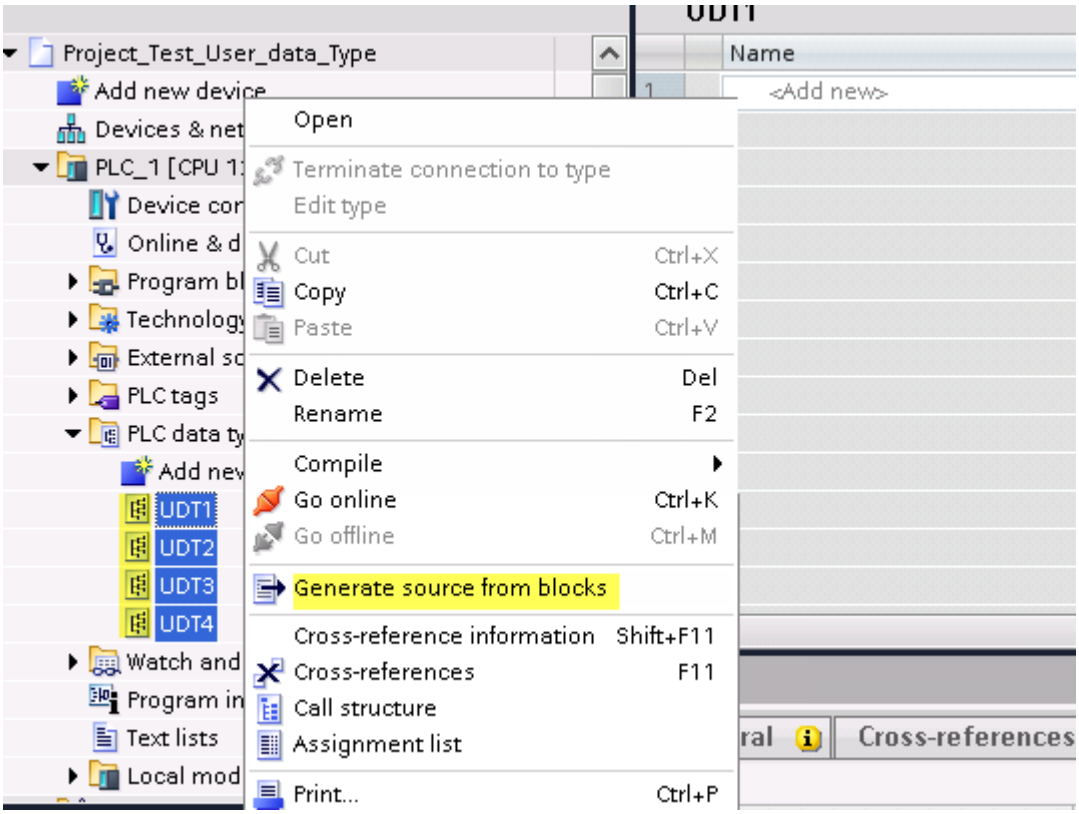


Exporting PLC data types

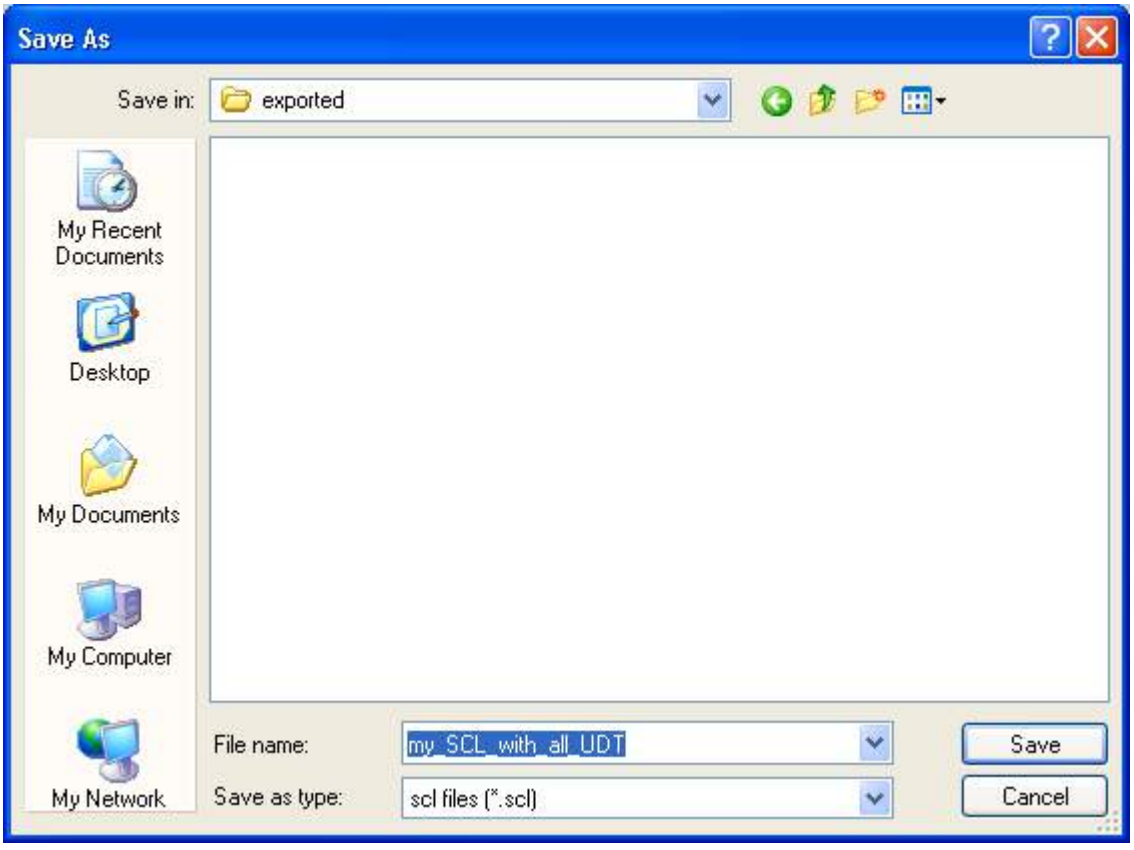
To create the file, expand **PLC data types** item from TIA Portal project tree and right click on the user defined structure. Then click on **Generate source from blocks**.



In case of multiple PLC data types in PLC project, it is necessary to select them all from **PLC data types** list, right click and select **Generate source from blocks** to create the .SCL file that contains all the PLC data types defined.



In the next step, give a name to the .SCL file and choose the path to where to save the file.



This file will content all the PLC data types and it can be used for importing tags in Tag Editor.

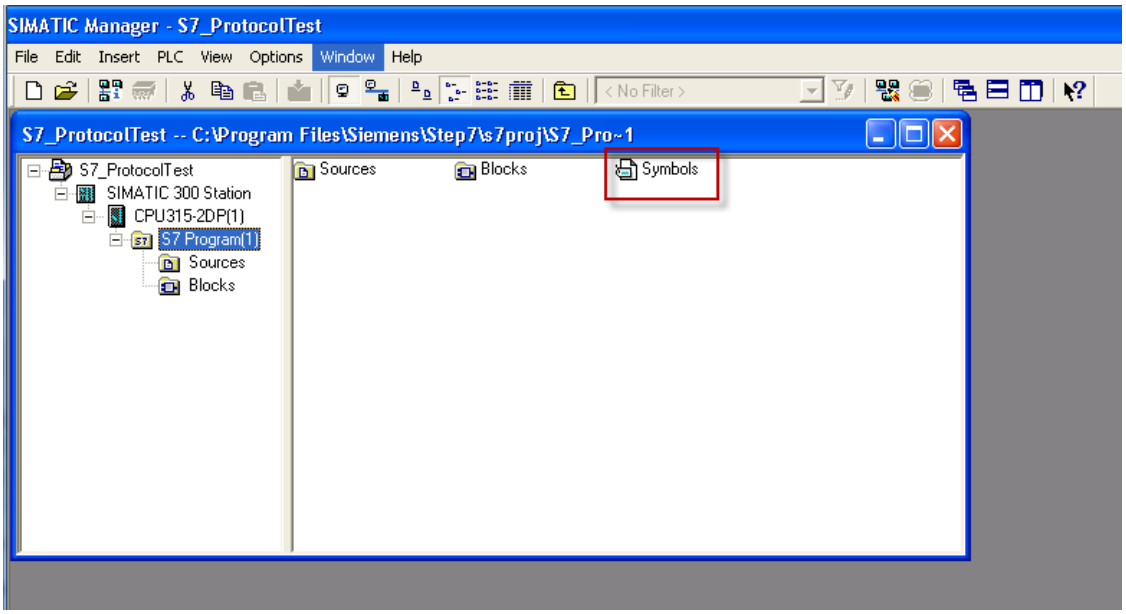
Check **Tag Import** chapter for more details.

Export using STEP7

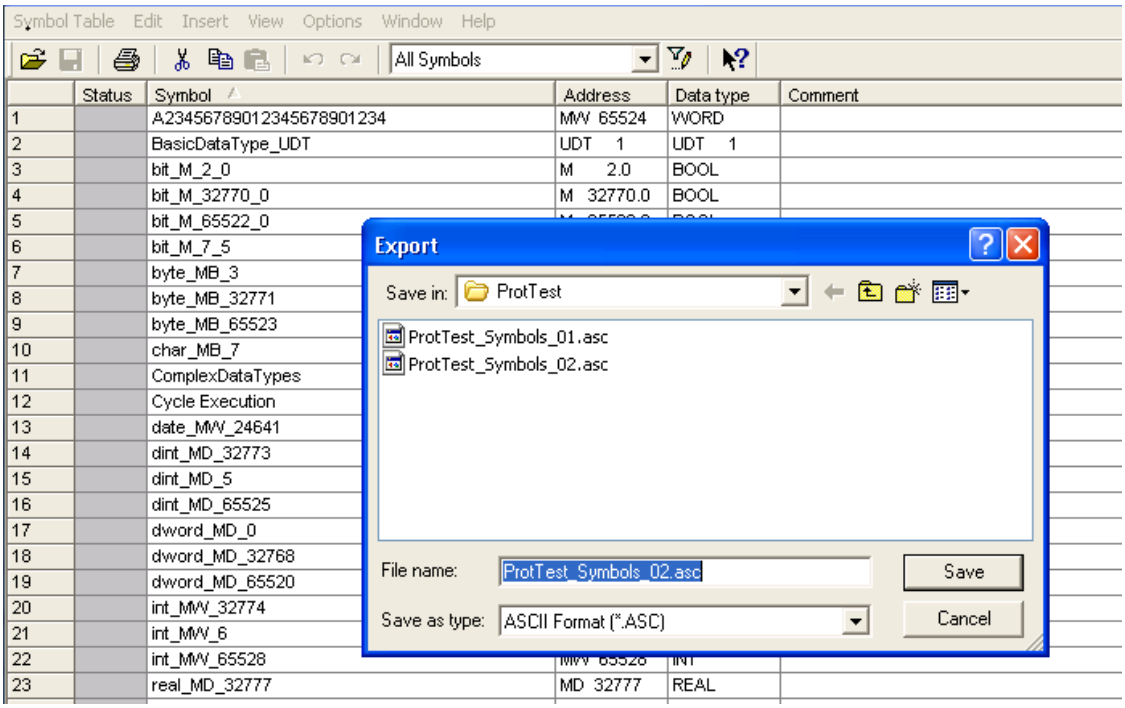
The Simatic S7 ETH Tag importer accepts symbol files (ASCII format .asc) and source files (.awl extension) created by the Simatic Step7. The symbol file can be previously exported using the Step7 symbol table utility.

Exporting Symbols table

Symbol files (.asc) can be exported from the symbol table utility.



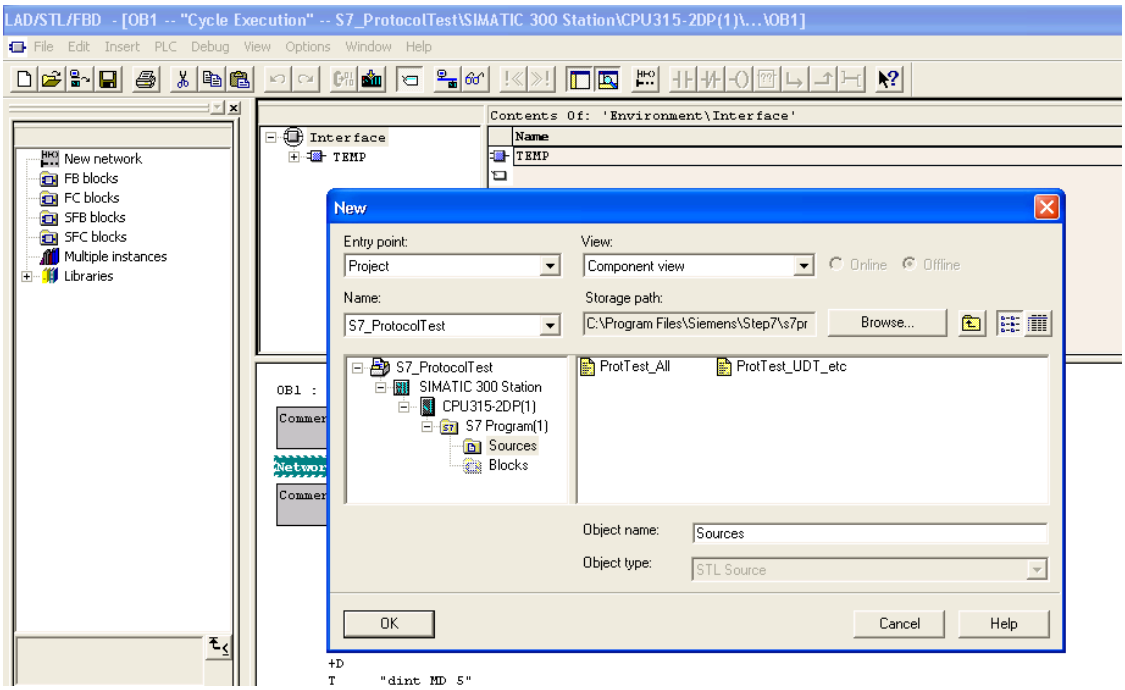
1. From the **Symbol Table** menu in the Symbol Editor choose **Export**.
2. Assign a name and save the symbol table as ASCII file.



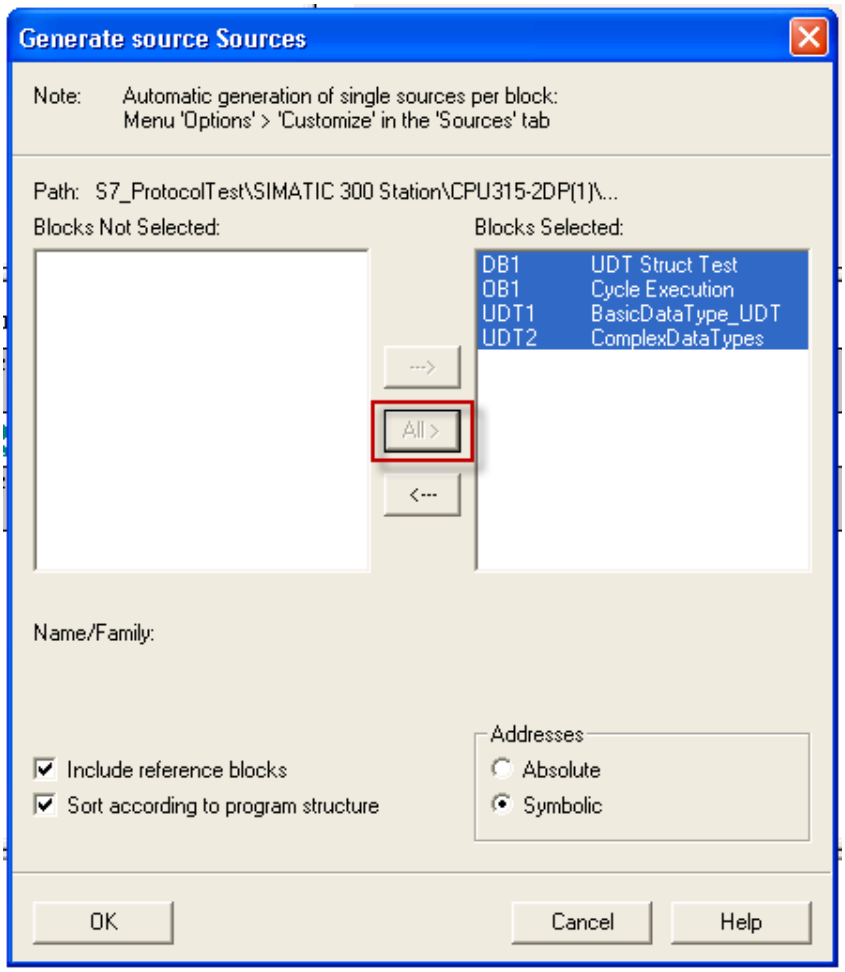
Exporting Sources

These files are created exporting source code.

1. Open any program block in the editor, "OB1" in this example.
2. From the **File** menu choose **Generate Source**: the following dialog is displayed:



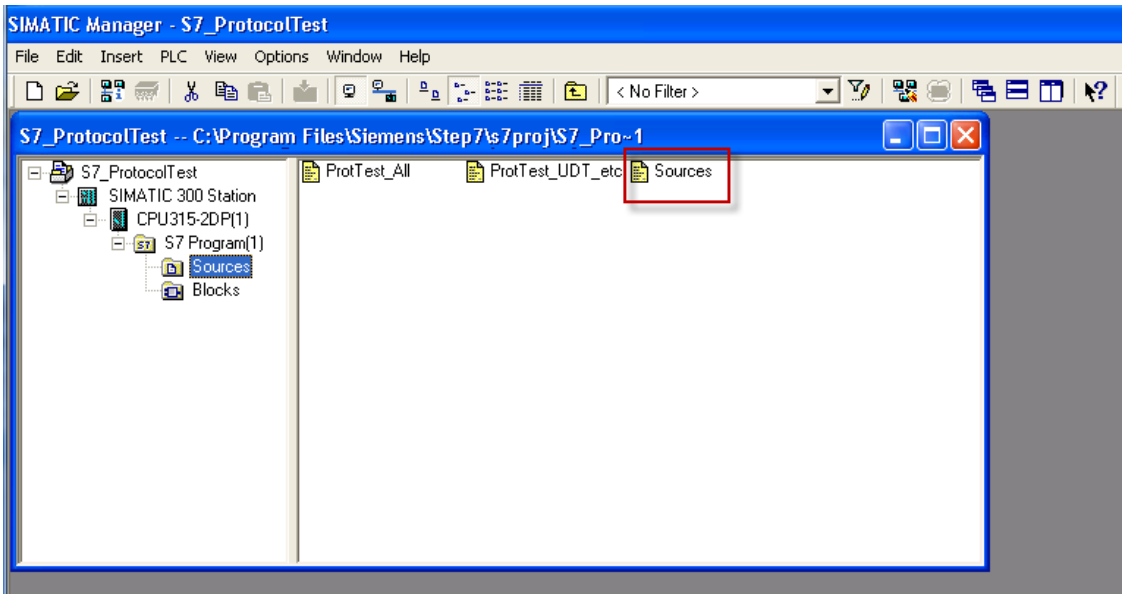
1. Assign a name, "Sources" in the example, and click **OK**: the **Generate source Sources** dialog is displayed.



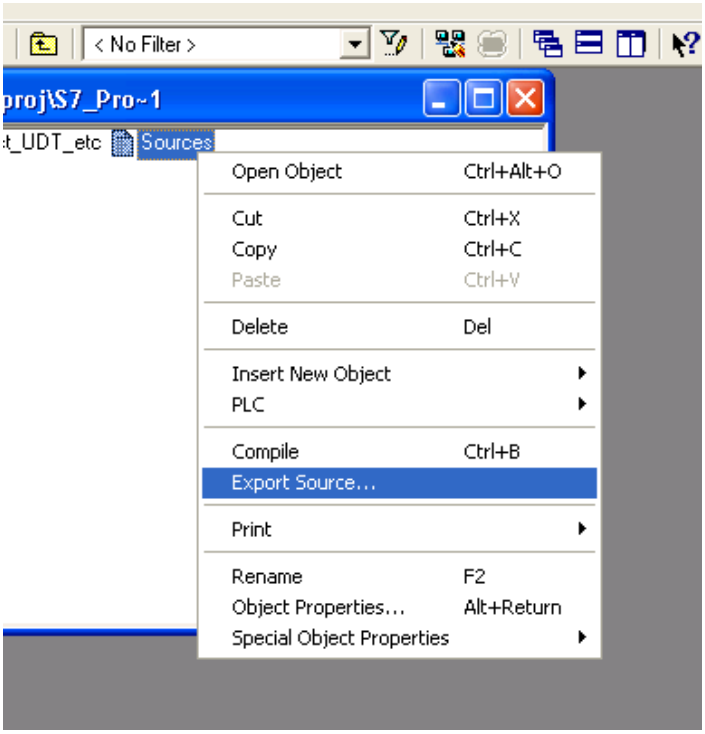
2. Click **All >** to generate source for all blocks.
3. Select the following options:

- **Include reference blocks**
- **Sort according to program structure**
- **Symbolic** address

4. Click **OK** to confirm: the "Sources" object is generated in the Step7 project as in the example.



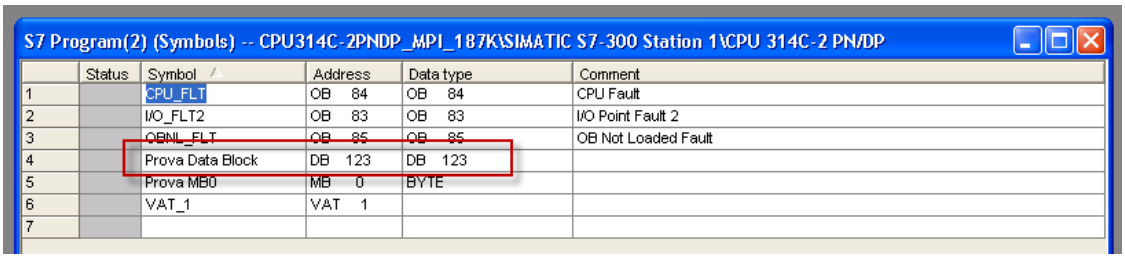
5. Right click on the object and select **Export Sources**.



The generated .awl file can be imported in the Tag Editor.

Note: The .awl file contains additional information not included in the .asc file exported from the symbol table.

Make sure that reference to all data blocks is inserted in the symbol table. The tags from a data block are imported only if the symbol table contains a line with the data block name and related comment.

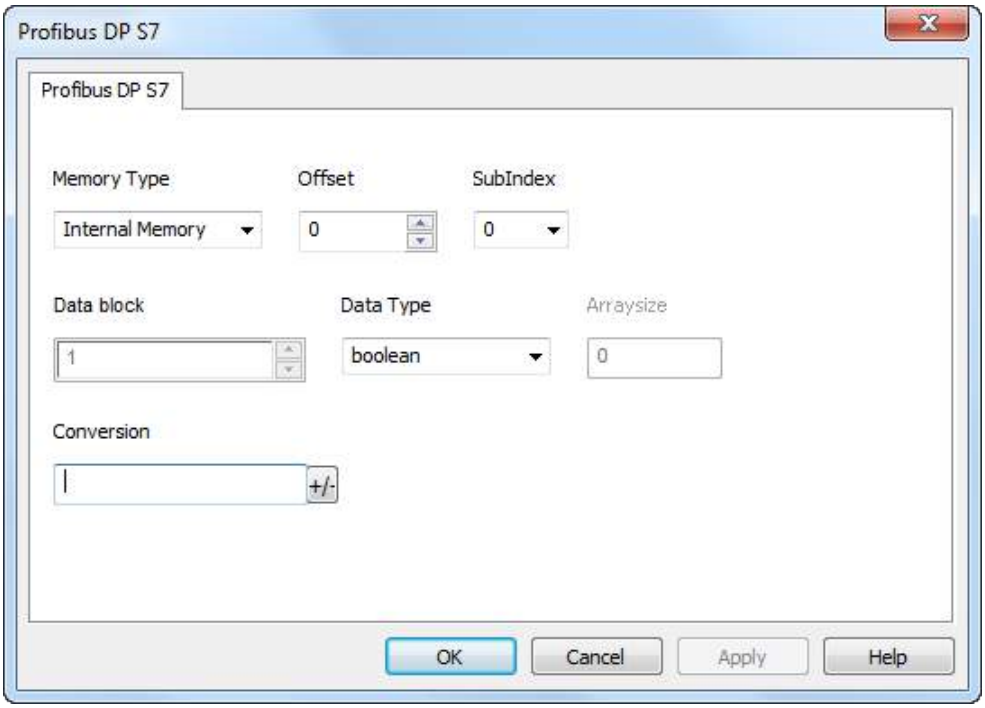


Each entry enables the import filter to import the tags related to the specified data block.

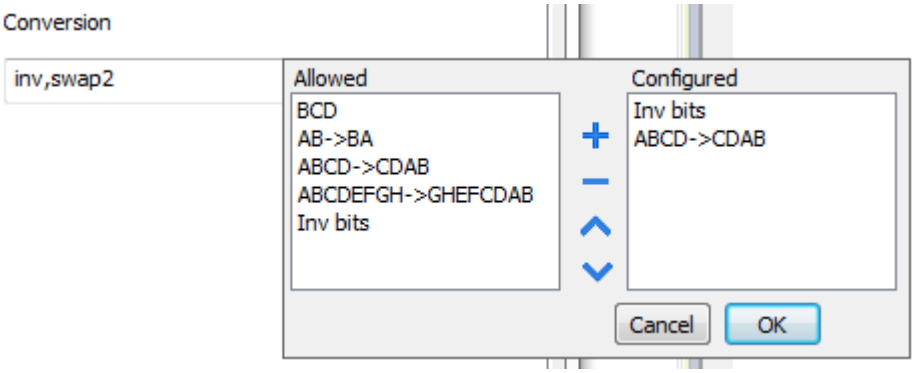
Tag Editor Settings

Into Tag editor select the protocol “Profibus DP S7” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Area of PLC where tag is located
<table><tr><th>Data Type</th><th>Simatic Type</th></tr><tr><td>Internal Memory</td><td>M</td></tr><tr><td>Data Block</td><td>DB</td></tr><tr><td>Input</td><td>I (E)</td></tr><tr><td>Output</td><td>O (A)</td></tr><tr><td>Timer value</td><td>T</td></tr><tr><td>Counter value</td><td>C</td></tr></table></td></tr>
<tr><td>Offset</td><td>Offset address where tag is located</td></tr>
<tr><td>SubIndex</td><td>In case of Boolean data type, this is the offset of single bit</td></tr>
<tr><td>Data Block</td><td>If Memory Type is “Data Block”, this will identify the DB number</td></tr>
<tr><td>Data Type</td><td><table><tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr><tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr><tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr><tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr><tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr><tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr><tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr><tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr><tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr><tr><td>string</td><td colspan="2">Refer to “String data type channel”</td></tr></table>
Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…</td></tr>
<tr><td>Arraysize</td><td>• In case of array tag, this property represents the number of array elements.
• In case of string tag, this property represents the maximum number of bytes available in the string tag.</td></tr>
</table>

Element	Description
	Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the **Allowed** list shows one or more conversions, listed below.

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format)

Element	Description

Value	Description
	855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	Separate the byte in two nibbles, and reads them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)
S5timer(BCD)	Used to support S5timer. Check **Simatic S5timer special data type** for more details.
S5timer(BIN)	Legacy transformation for S5timer in binary format.

Select the conversion and click on plus button. The selected item will be added on **Configured** list.

If more conversions are configured, they will be applied in order (from top to bottom of **Configured** list).

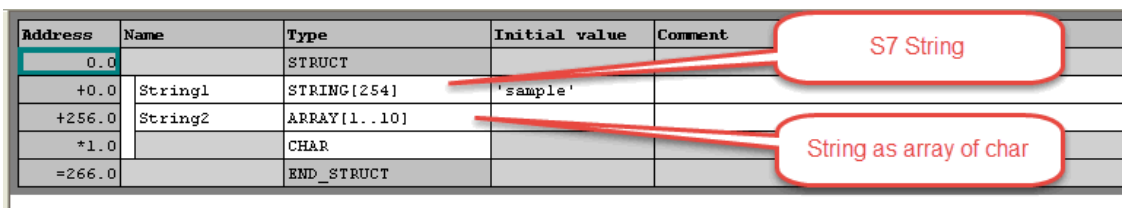
Use the arrow buttons to order the configured conversions.

String data type

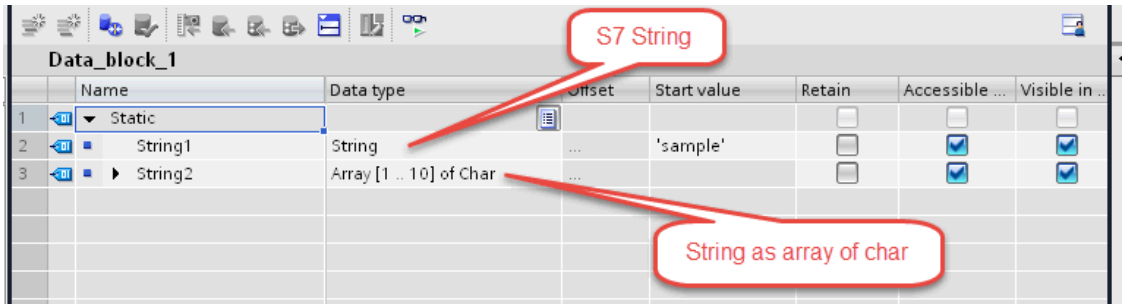
In ProfibusDP S7 PLC it's possible to define two different types of tags to manage string variables.

- as Array [1..xx] of Chars,
- as String[xx].

Step7 string declaration is showed in the following figure:



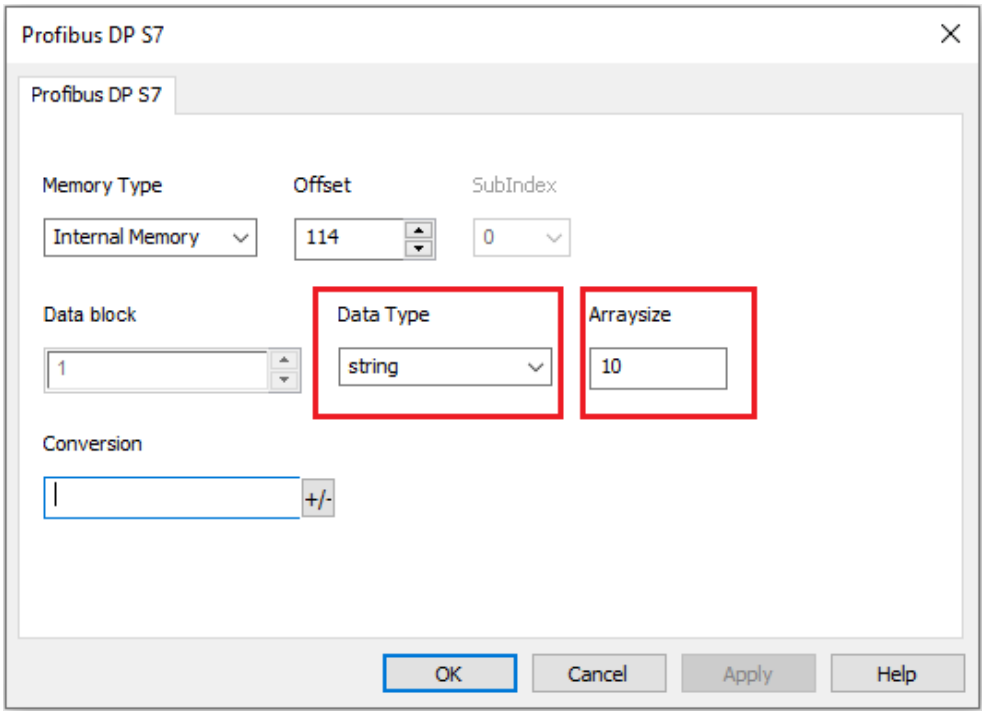
TIA Portal string declaration is showed in the following figure:

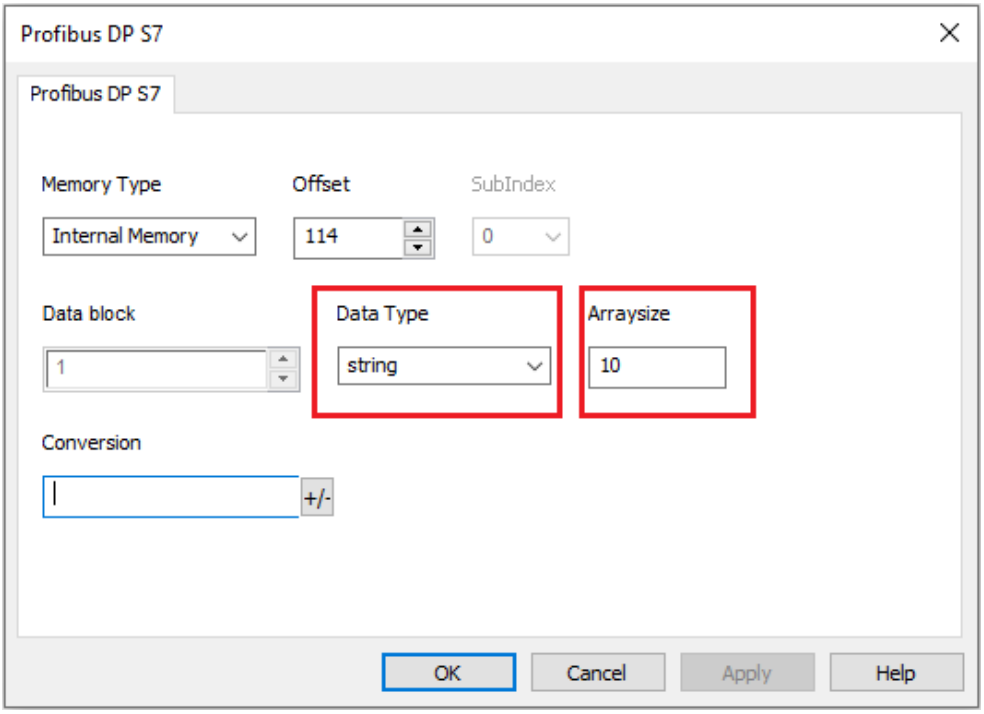


Note: Usage of String[xx] data type is allowed but a specific Conversion must be applied to the tag. Anyway using tag importer to import tag dictionary from TIA Portal or Step7 string tags are automatically configured and no changes/conversion are needed.

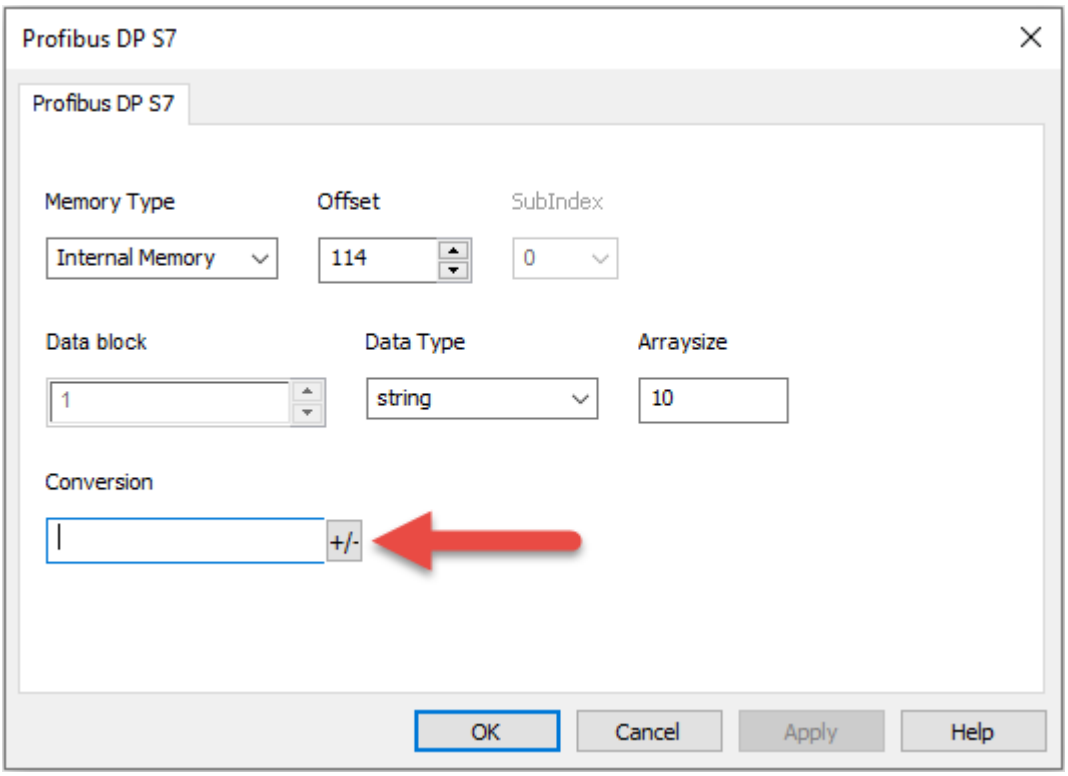
To manually add an "Array [1..xx] of Chars" data type tag, press the [+] button in the Tag Editor,

then select "string" as Data Type of the Tag and type the string length in the "Arraysize" field:



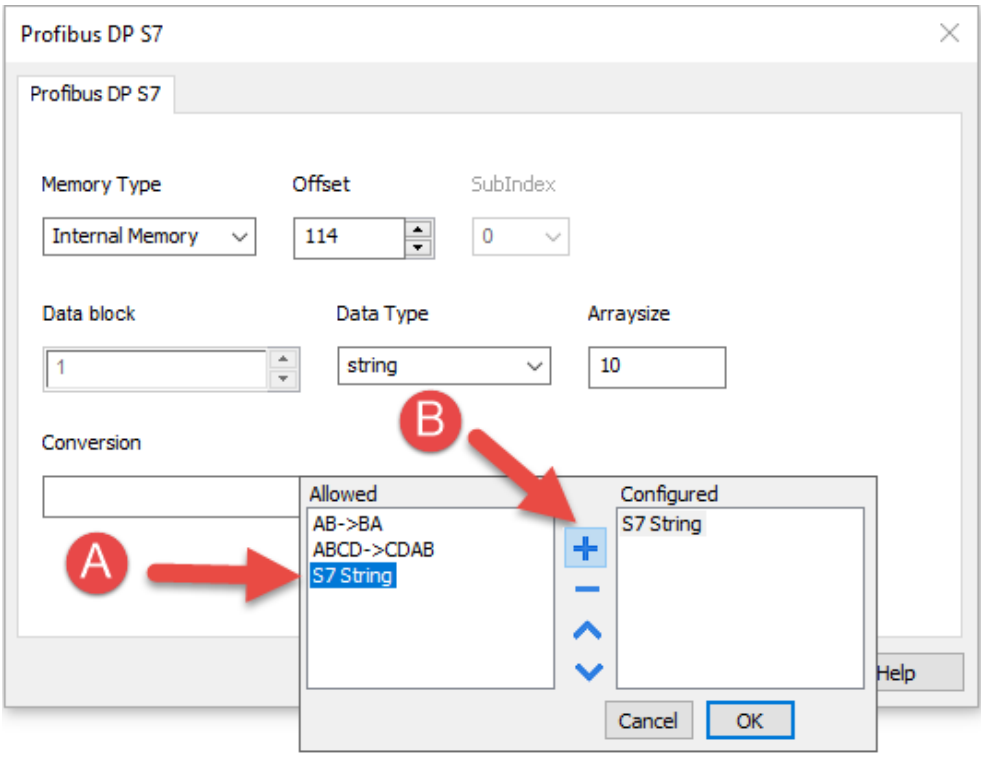


then click on [+/-] button to open the Conversion dialog.



Into conversion dialog:

- select the "S7 String" conversion type
- click on [+] button to add the conversion.



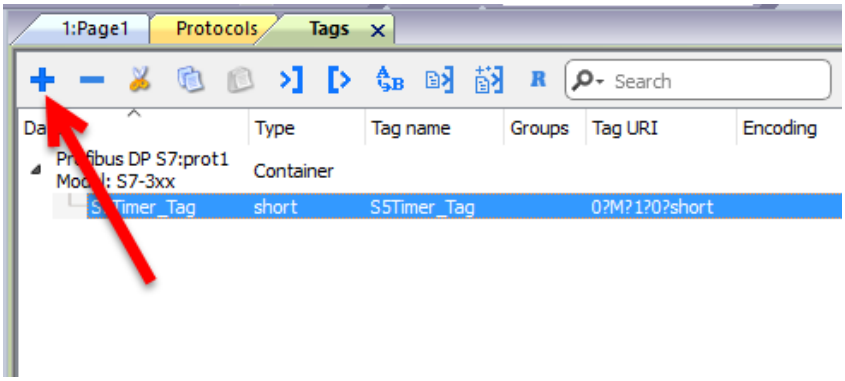
The conversion will be listed into the Configured window on the right.

Confirm with OK button.

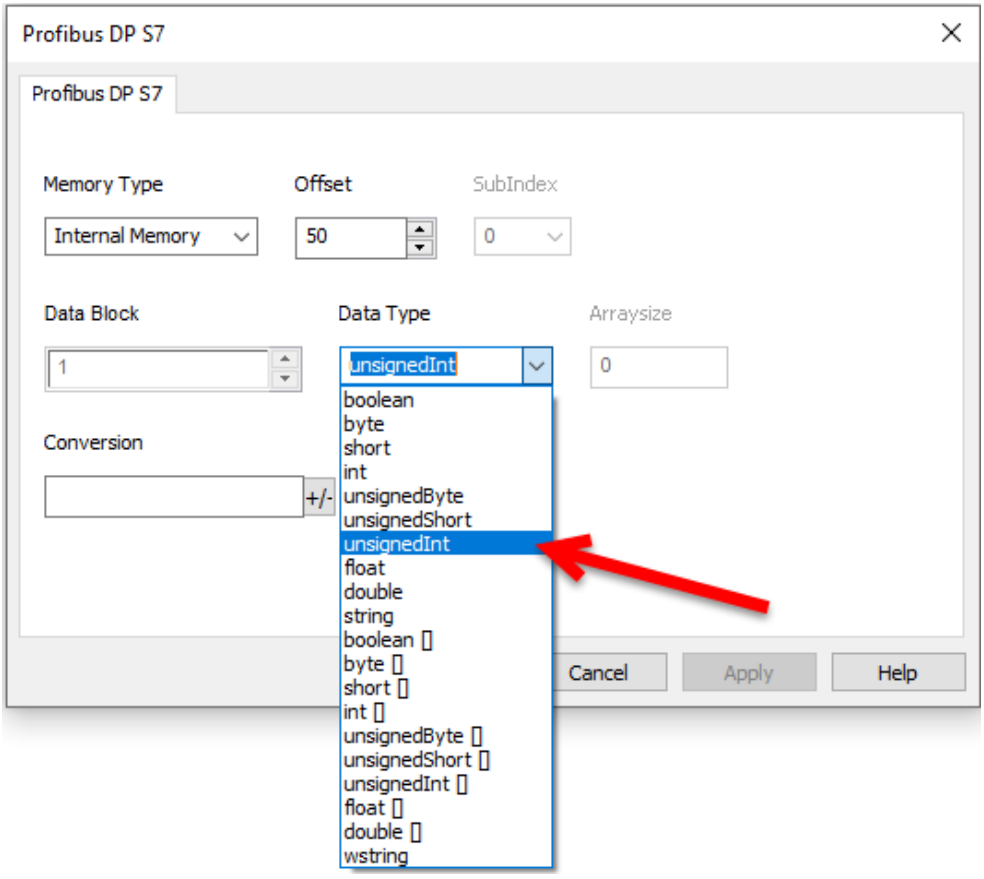
Simatic S5timer data type

Simatic drivers support a special data type, called S5Timer.
The tag must be configured with a specific data type and a conversion must be applied to the Tag to correctly read/write a Simatic S5Timer Variable.

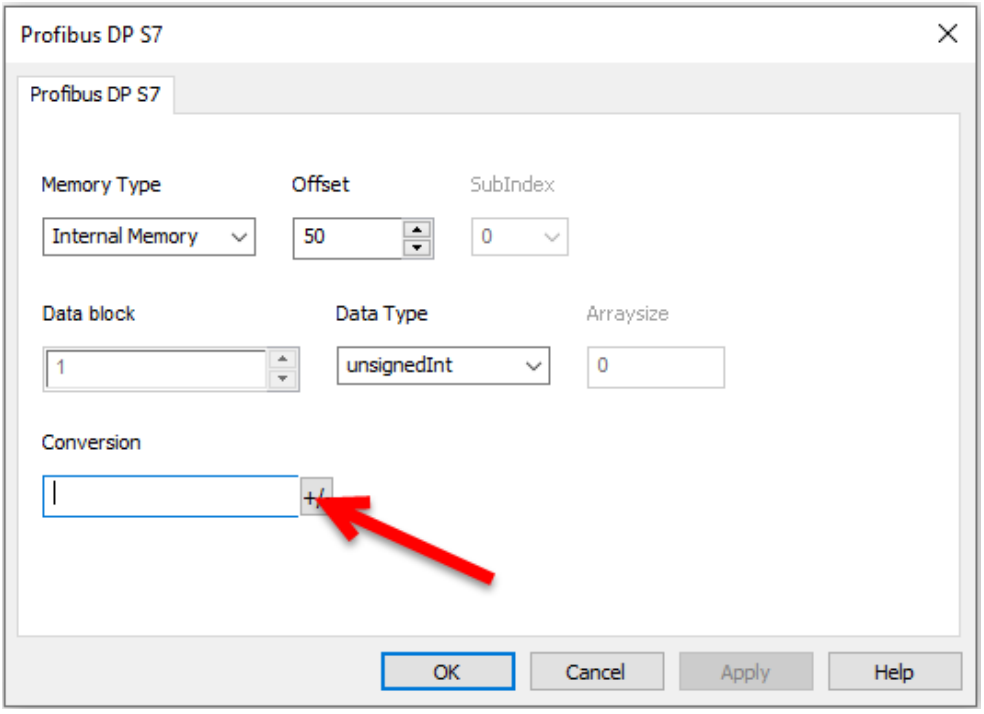
Open the Tag Editor and add a Tag pressing the Plus button.



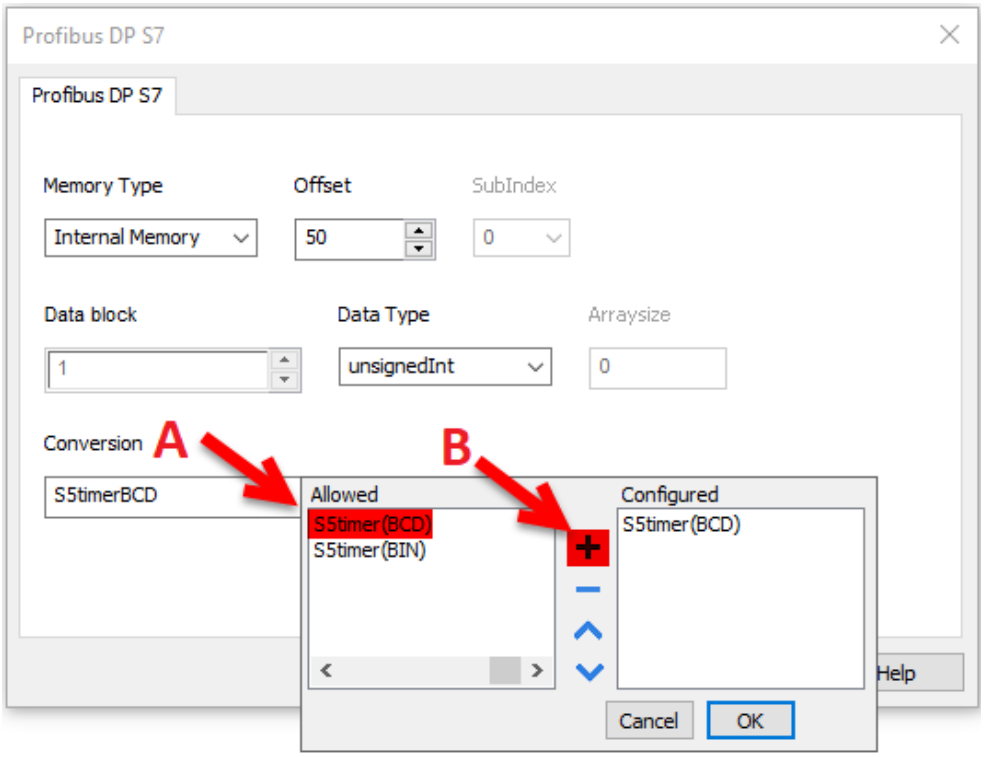
Select “unsignedInt” as Data Type of the Tag.



Click on +/- button to open the Conversion dialog.

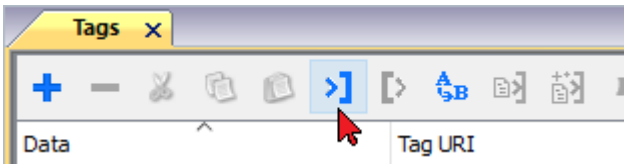


In the Conversion dialog select the S5timer(BCD) conversion type [A] then click on Plus button [B] to add the conversion, the configured conversion will be listed into the Configured window on the right. Then confirm with OK.

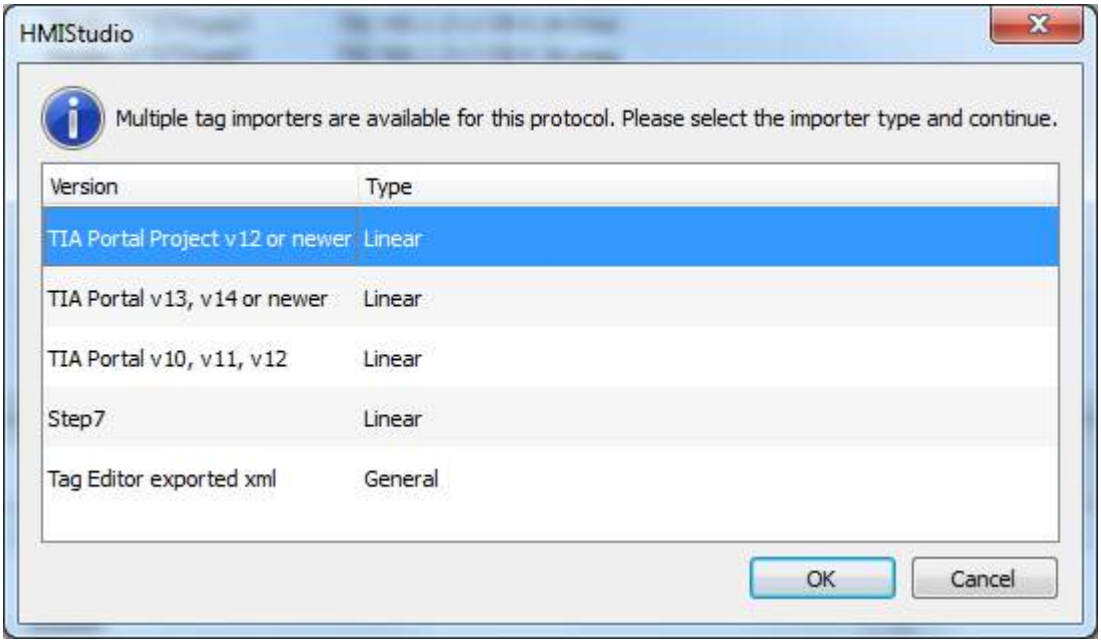


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



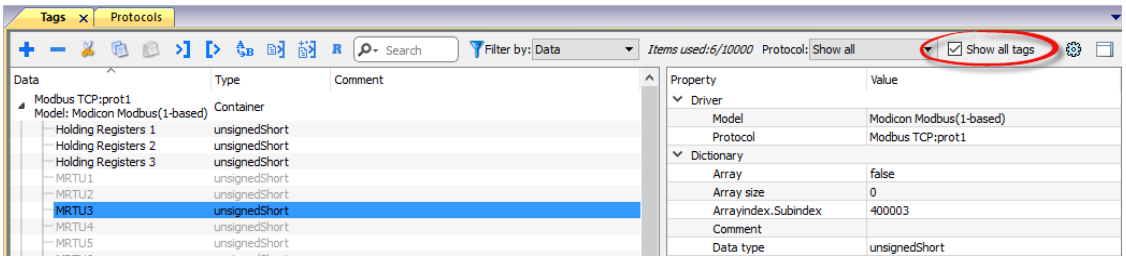
The following dialog shows which importer type can be selected.

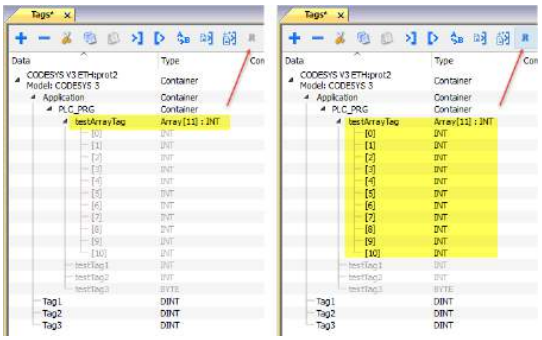
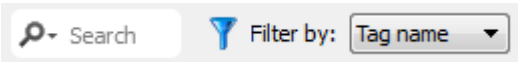


Importer	Description
TIA Portal Project v12 or newer **Linear**	Allows to import the whole TIA Portal project file using **.apxx** file (where "xx" is the TIA Portal version, example: for TIA Portal 13 , file name is "project.ap13"). All variables will be displayed at the same level.
TIA Portal v13, v14 or newer **Linear**	Allows to import: • Program blocks using **.db** file • PLC tags using **.xlsx** file • PLC data types using **.udt** file Check **Export using TIA Portal v13, v14 or newer** for more details. All variables will be displayed at the same level.
TIA Portal v10, v11, v12 **Linear**	Allows to import: • Program blocks using **.tia** file • PLC tags using **.xlsx** file • PLC data types using **.scl** file Check **Export using TIA Portal v10, v11, v12** for more details. All variables will be displayed at the same level.
Step7 **Linear**	Allows to import: • Symbols table **.asc** file • Sources using **.awl** file Check **Export using STEP7** for more details. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Invalid	The panel did receive from the controller a response, but its format or its contents or its

Error	Notes
response	length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General error	Error cannot be identified; should never be reported; contact technical support

Rexroth IndraControl

The Rextoth IndraControl communication driver has been designed to connect HMI devices to Bosch Rexroth PLC trough ethernet connection.

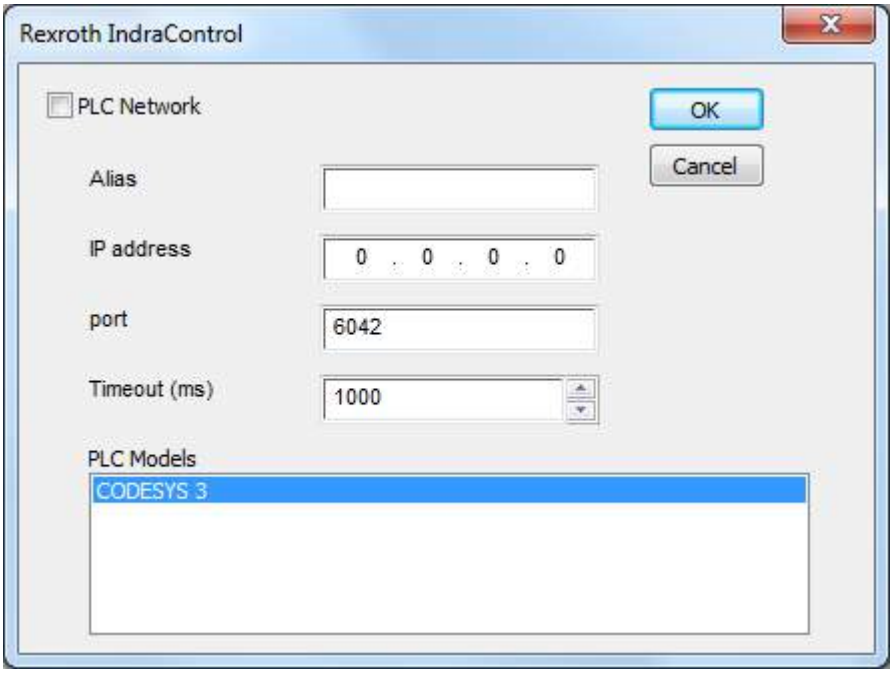
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	Port number used by the driver. The default value is **6042**.
Timeout	Time delay in milliseconds between two retries in case of missing response from the server device.
PLC Models	PLC models available:

Element	Description
	• **CODESYS 3**
PLC Network	Multiple controllers can be connected to one HMI device. To set-up multiple connections, select **PLC network**and click **Add** to configure each node

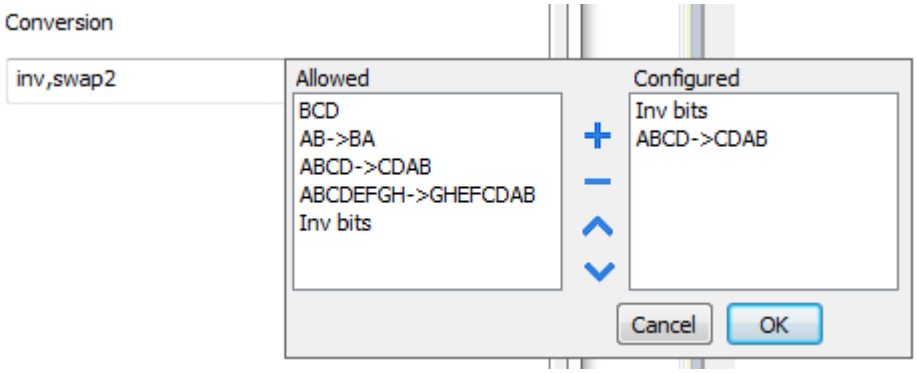
Data Types

The import module supports variables of standard data types and user defined data types.

Supported data types	• BOOL • INT • SINT • UINT • UDINT • DINT • STRING* • REAL • LREAL • BYTE • ULINT • LINT and 1-dimensional ARRAY of the types above. See "Programming concepts" section in the main manual. Note *: String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.
Unsupported data types	• LWORD • LINT

Tag Conversion

Conversion to be applied to the tag.



Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

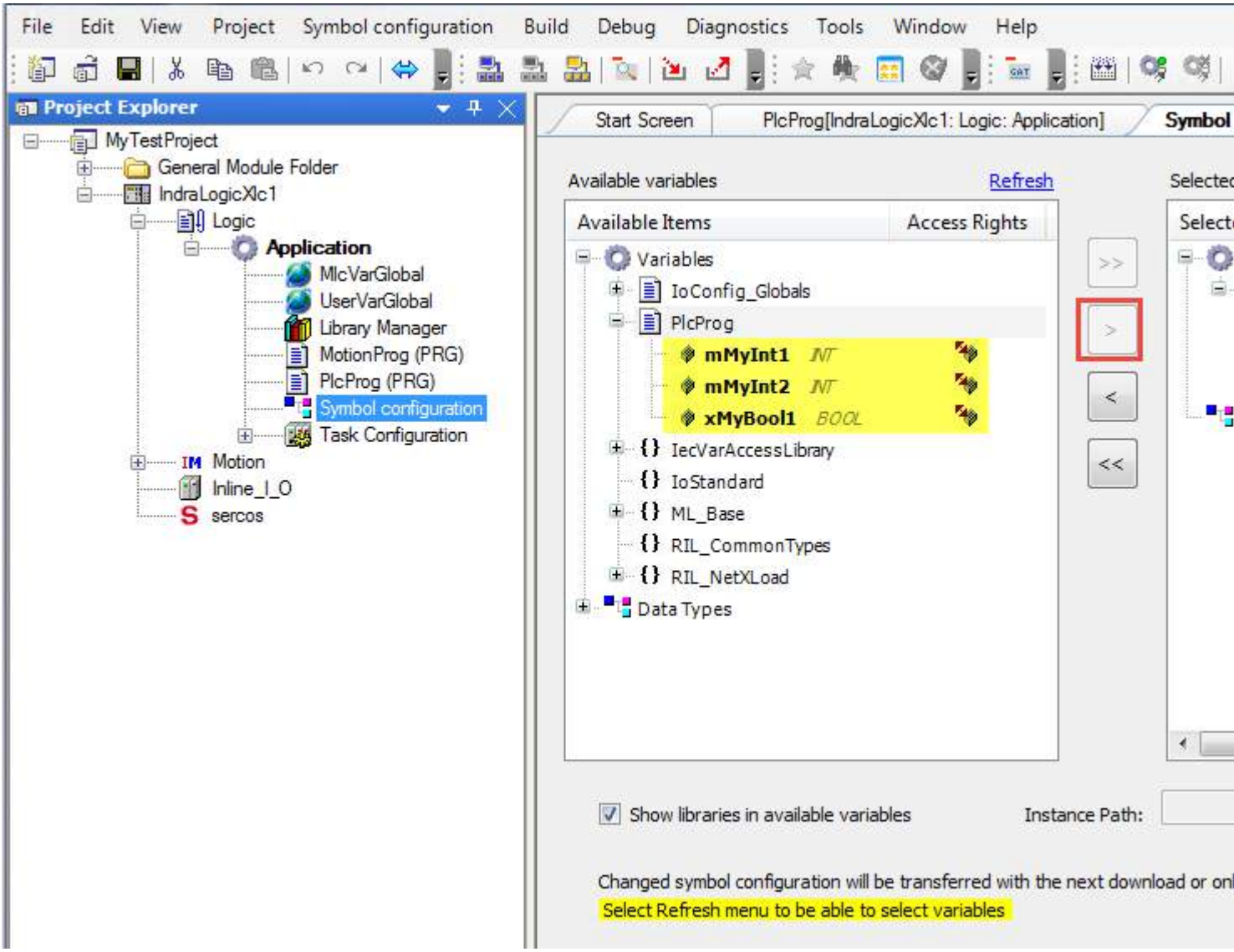
If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Tag import

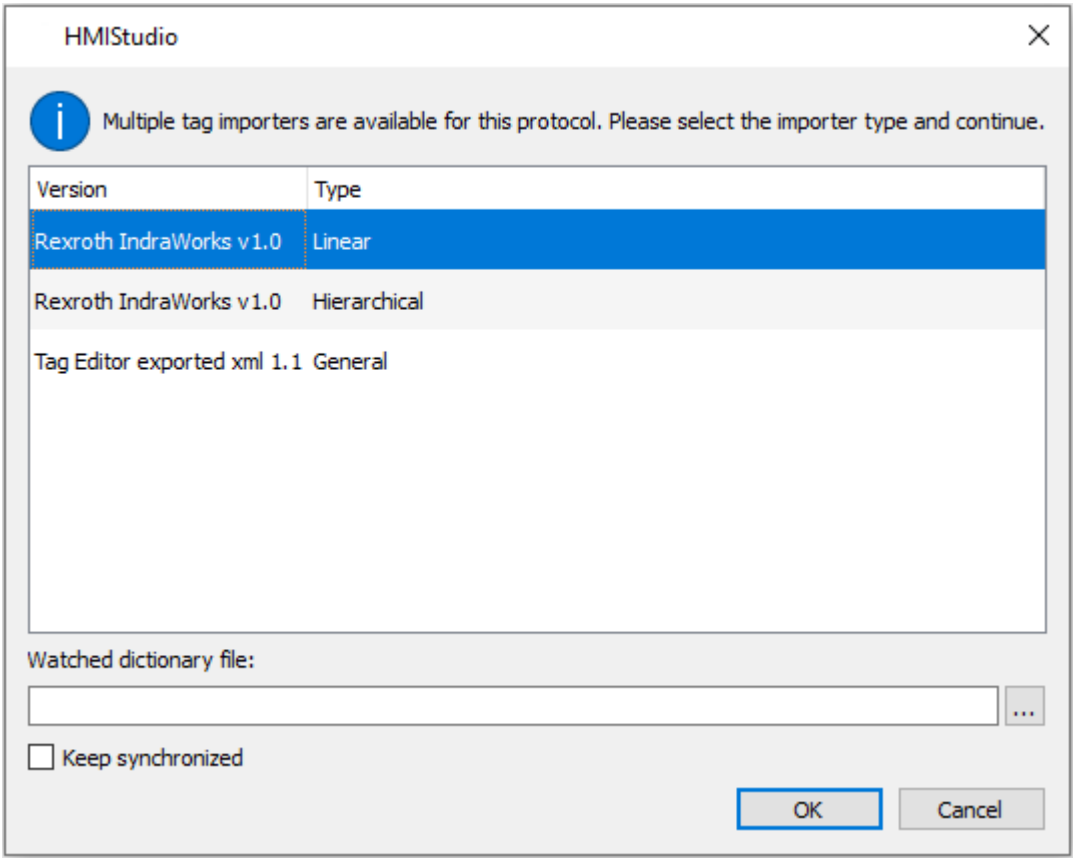
When creating the project using IndraWorks programming software, properly configure the symbol file to contain the required variables.

Symbol configuration item contains a list of all the variables available into the IndraWorks project, single variables or groups of variables can be selected and moved to **Selected variables** column.



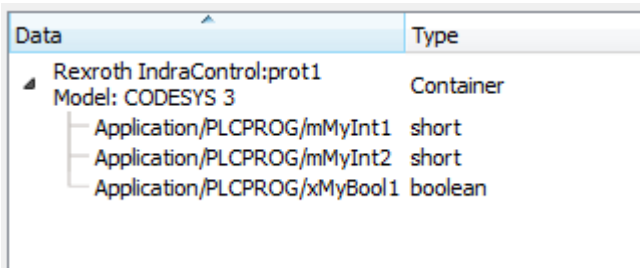
1. After the symbols have been configured, download the project or use the **Generate code** function (Build > Generate code) to create an .xml file containing all the variables read to be imported in the Tag Editor. The .xml file is created in "*C:\ProgramData\IW-Projects\0\Project\IndraLogic*" by default.

2. Select the driver in the Tag Editor.
3. Click the **Import Tags** button to start the import process.

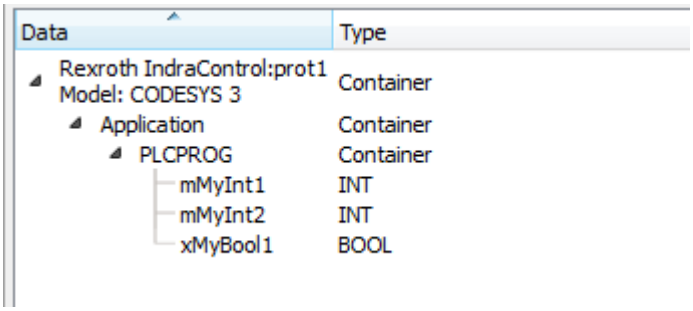


Select the importer by choosing from the list above.

Linear All variables will be displayed at the same level.

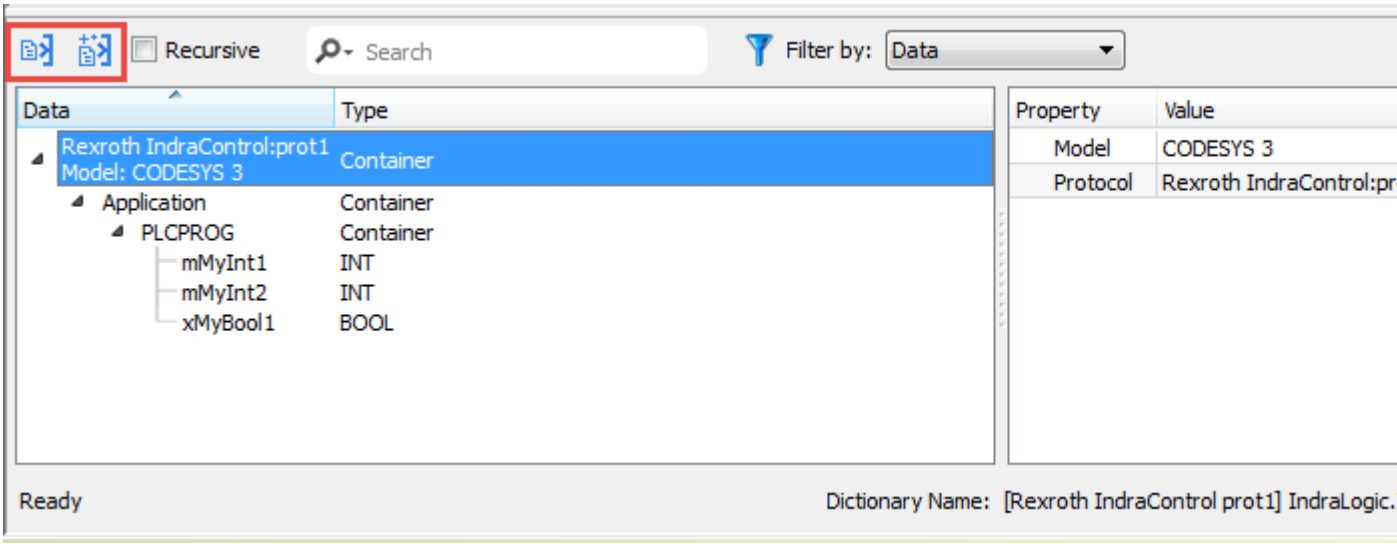


Hierarchical All variables will be displayed according to CODESYS V3 Hierarchical view



General Select this importer to read a general XML file exported from the Tag editor

4. Locate the .xml file and click **OK**: the tags included in the created document are listed in the tag dictionary.
5. To add the tags to the project click the **Add tags** button.



See "My first project" section in the main manual.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid response	The device did received a response with invalid format or contents from the controller.	Check if the data programmed in the project are consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

ROBOX BCC/31

ROBOX BCC/31 communication driver has been designed to connect HMI devices to ROBOX BCC/31 PLC through Ethernet connection.

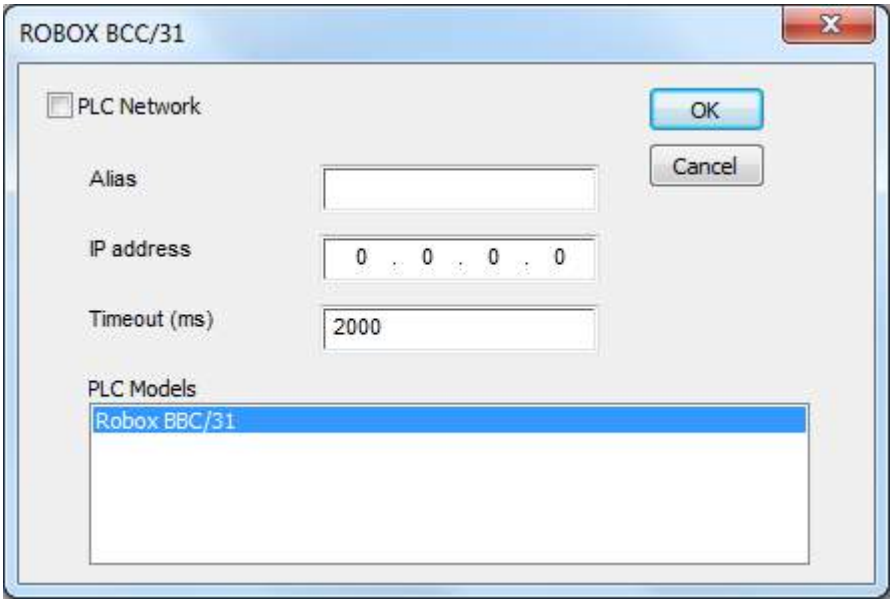
Protocol Editor Settings

Adding a protocol

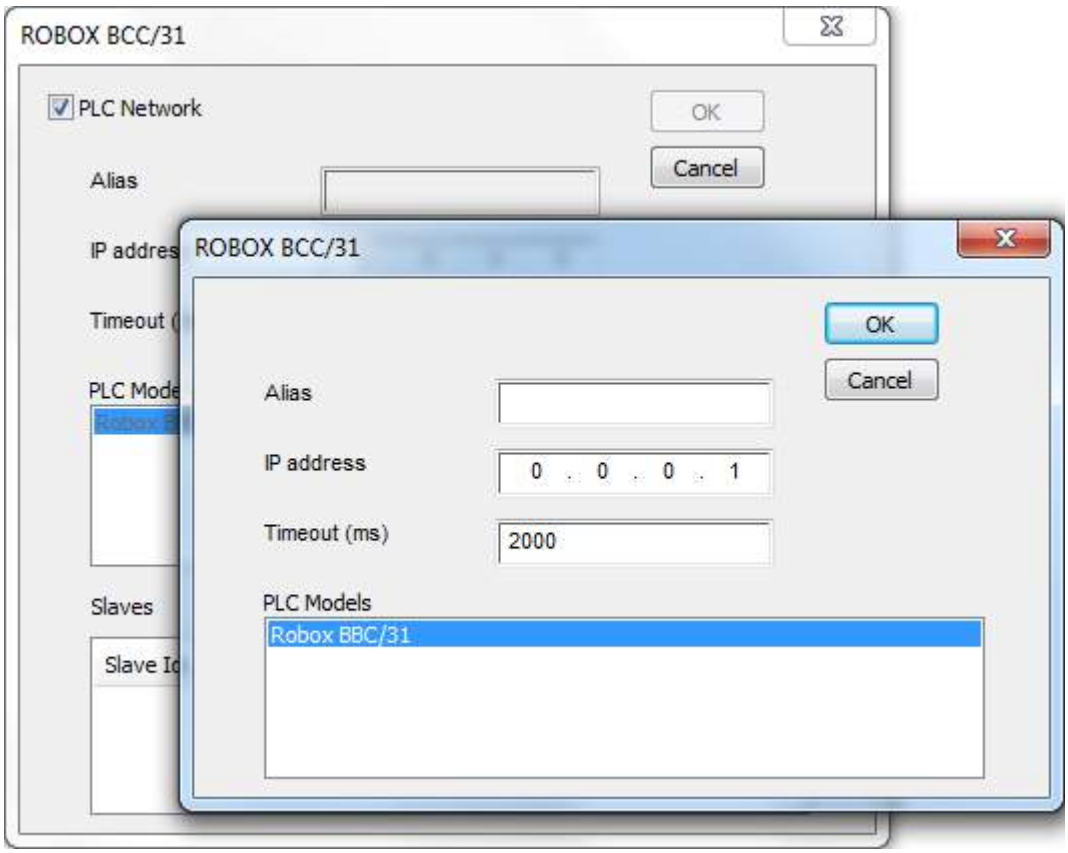
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Address of PLC.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
PLC Models	Allows to select between different PLC models:

Element	Description
	• **Robox BBC/31**
PLC Network	IP address for all PLCs in multiple connections. **PLC Network** must be selected to enable multiple connections. 

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **ROBOX BCC/31** from the **Driver** list: tag definition dialog is displayed.

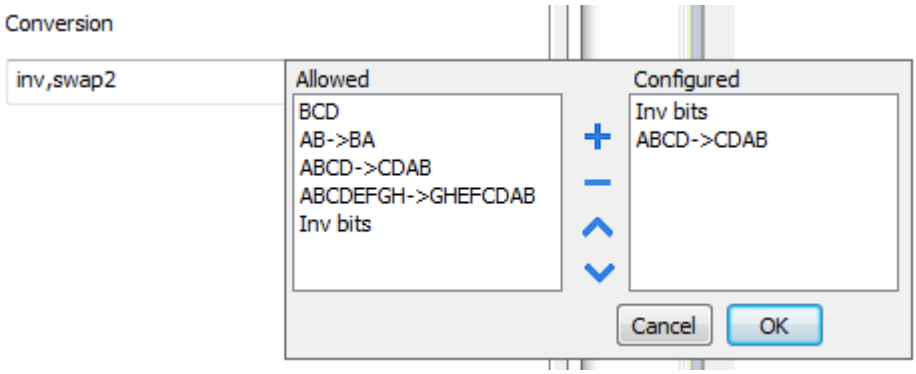

ROBOX BCC/31
ROBOX BCC/31
Memory Type
Offset
SubIndex
Logic Input Bit
1
0
Axis Index
Data Type
Arraysize
1
boolean
0
Conversion
+/-
OK
Cancel
Apply
Help

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Resource where tag is located on PLC.
Available resources are:
Logic Input Bit
Logic Input Word
Logic Output Bit
Logic Output Word
Phis Input Bit
Phys Input Word
Phys Output Bit
Phys Output Word
Non Volatile I32
Non Volatile Double
Non Volatile string
Volatile I32
Volatile Double
Volatile string
Parameter I32
Parameter Double
Axis Parameter I32
Axis Parameter Double
Alarm Mask
Alarm Code
Alarm string</td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.
Offset addresses are six digits composed by one digit data type prefix + five digits resource address.</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the selected memory type.</td></tr>
<tr><td>Axis Index</td><td>Allows to select Axis index. Available only for Axis memory types.</td></tr>
<tr><td>Data Type</td><td><table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description

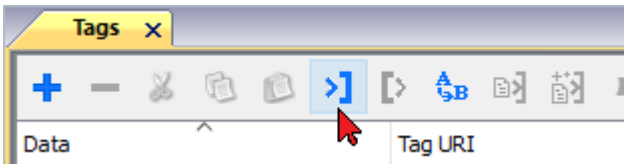
Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

Element	Description
	If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.

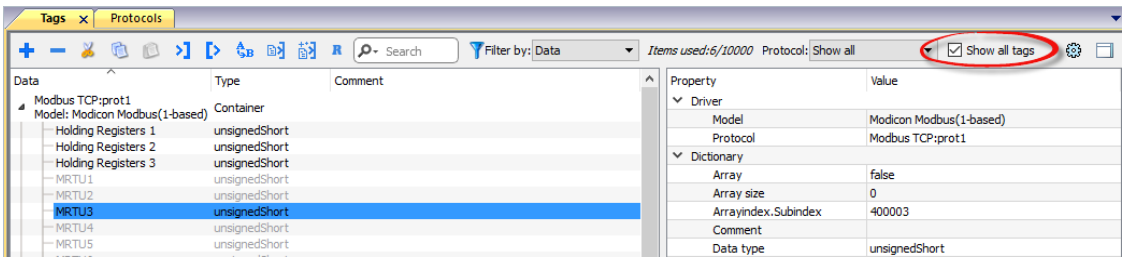


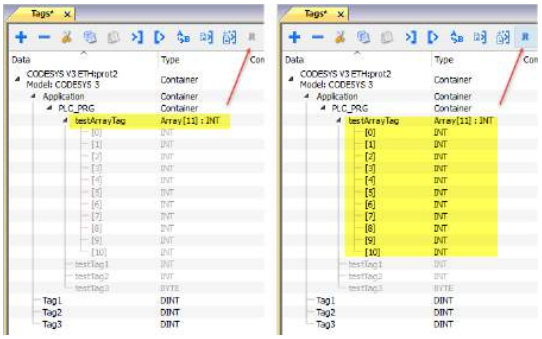
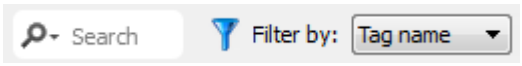
It is possible to import a Tag Editor exported xml

Type	Description
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

JavaScript Interface

Beside Tag interface the user can access the protocol via JavaScript.

Although defined Tags can be accesses by JavaScript too, JavaScript can access directly to a Command interface implemented in protocol. This interface does not require the definition of Tags and is direct to protocol resulting in more efficiency.

The following commands are supported:

Command	Description
dir (node,path)	Get directory of node starting from path.
readFile (node,deviceFilePath,localFilePath)	Get file from node.
writeFile (node,deviceFilePath,localFilePath)	Write file to node.
deleteFile	Delete file into node.

Example of usage:

```
var tagMgr = project.getWidget("_TagMgr");
var protID = "prot2"; // to be set according to protocol numbering

var params = "0 /F@/file.ext /mnt/usbmemory/file.ext";
tagMgr.invokeProtocolCommand(protID, "writeFile", params, state);
```

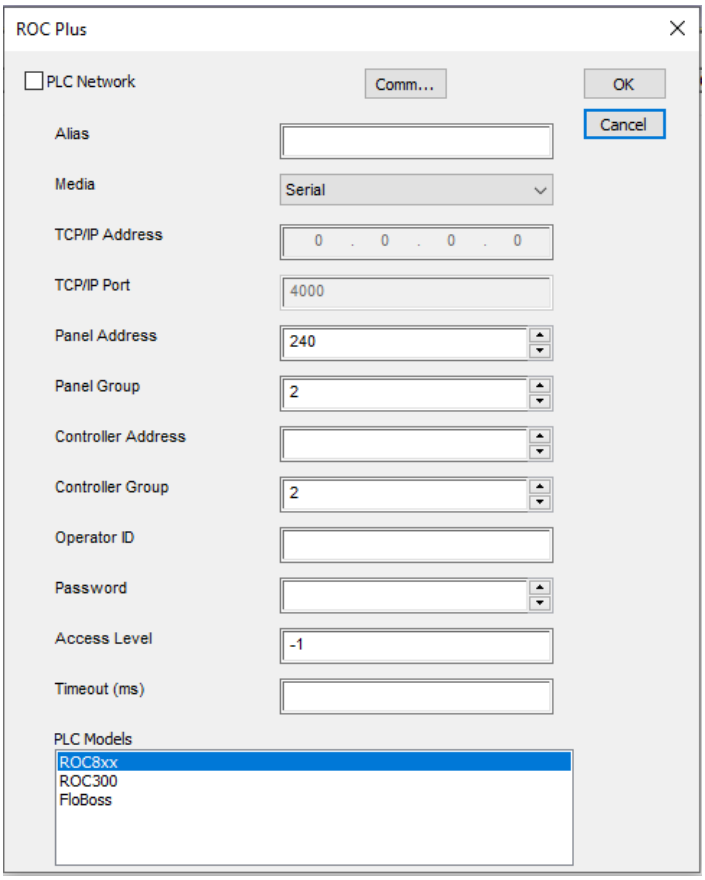
ROC Plus

The HMI device can be connected to a ROC Plus network as the network master using this communication driver. Communication with the ROC800 controllers is over an Ethernet or serial link. Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated.

Accordingly, always ensure that the latest driver is used in the application.

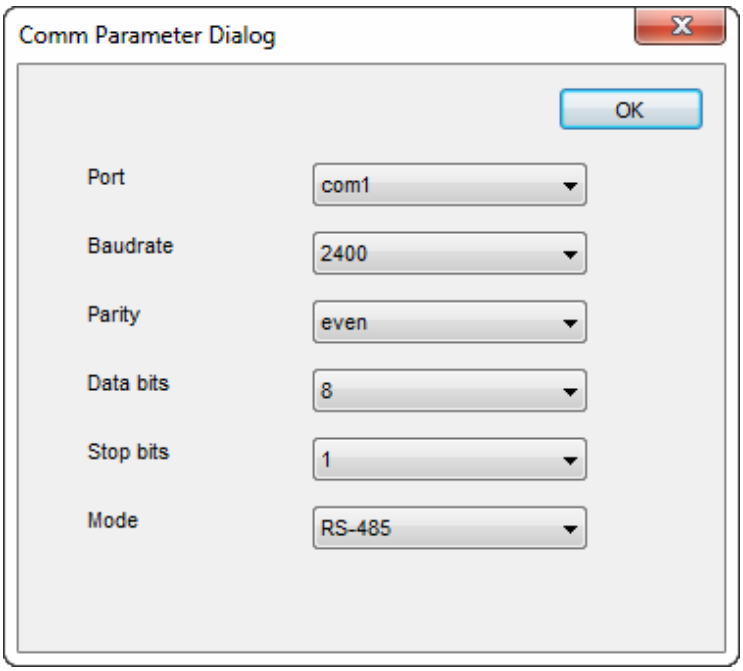
Protocol Editor Settings

Add (+) a driver in the Protocol editor and select the protocol called “ROC Plus” from the list of available protocols. The driver configuration dialog is shown in figure.



Element	Description
Media	Specify if the HMI is connected to the controller via serial communication link or Ethernet (TCP/IP)
TCP/IP Address	Ethernet IP address of the controller
TCP/IP Port	Port number used by the ROC plus driver; the default value can be changed when the

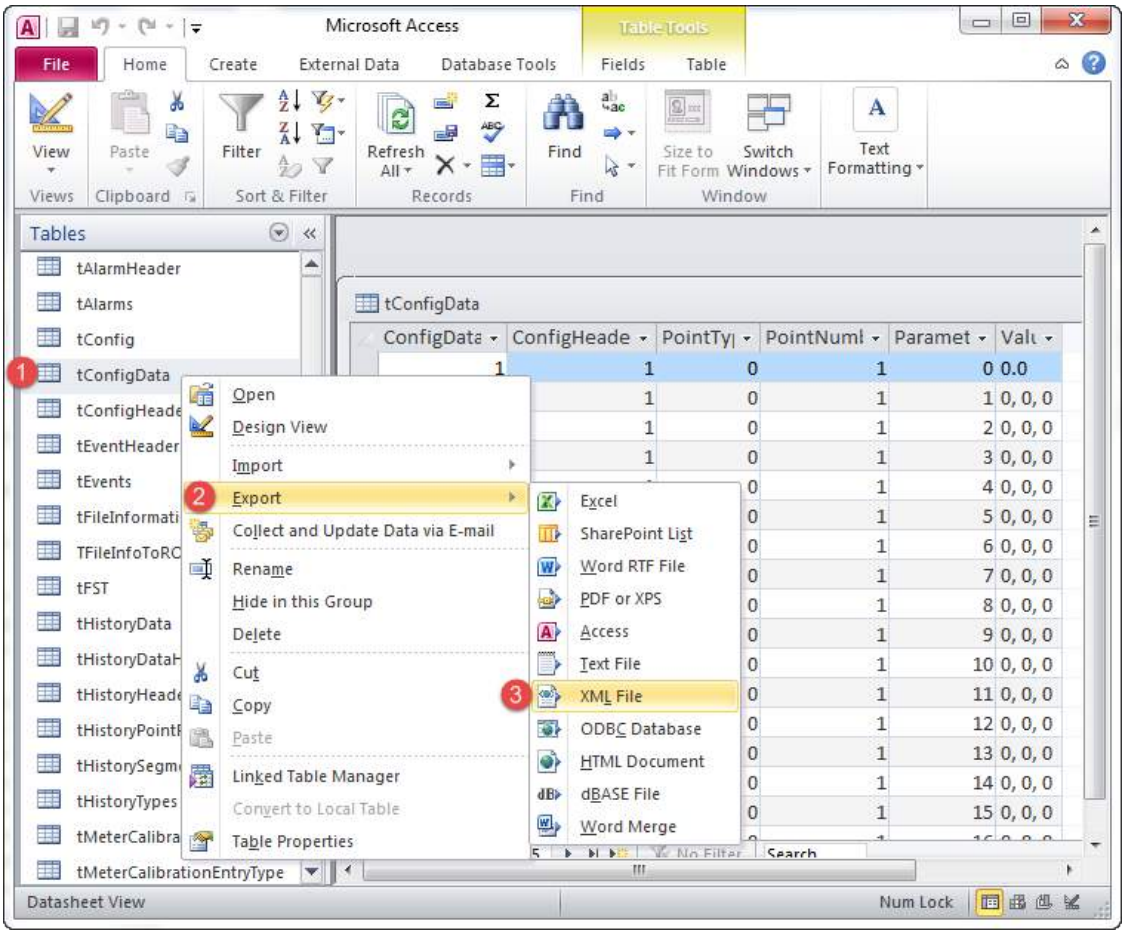
Element	Description
	communication goes through routers or Internet gateways where the default port number is already in use
Panel Address	Indicates the address of the HMI, this must be a unique number. 0 represents “broadcast within group” and 240 is the “direct connect address.”
Panel Group	Indicates the group code for the station address. This is user-configurable and usually set to 2.
Controller Address	Indicates the address of the controller, this must be unique
Controller Group	Indicates the group code for the station address. This is user-configurable and usually set to 2.
Operator ID	Sets operator identification code for the communications port through which communications are occurring. The operator identification is logged with an event, indicating the operator responsible for creating the event.
Password	A numerical value that is used as a password for the Operator Identifier
Access Level	A value that is used to limit access to parameters.
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from the server device. Value is expressed in milliseconds.
PLC Models	The driver supports the communication with a number of different Emerson controllers. Please check directly in the programming IDE software for a complete list of supported controllers.
PLC Network	The protocol allows the connection of multiple controllers to one HMI device. To set-up multiple connections, check “PLC network” checkbox and create your network using the command “Add” per each slave device you need to include in the network.
Comm…	Click on this button to configure the serial port on the panel to be used

Element	Description
	
Port	On UN20: • com1 is the HMI port labeled “PLC”, • com2 is the HMI port labeled “PC/Printer” On UN31 or UN30: • com1 is the integrated serial port, • com2 is an add-on module plugged in Slot#1 or #2 • com3 is an add-on module plugged in Slot#3 or #4
Baudrate, Parity, Data bits, Stop bits	Communication parameters for the serial line.
Mode	Serial port mode; options are: • RS-232, • RS-485 (2 wires) • RS-422 (4 wires)

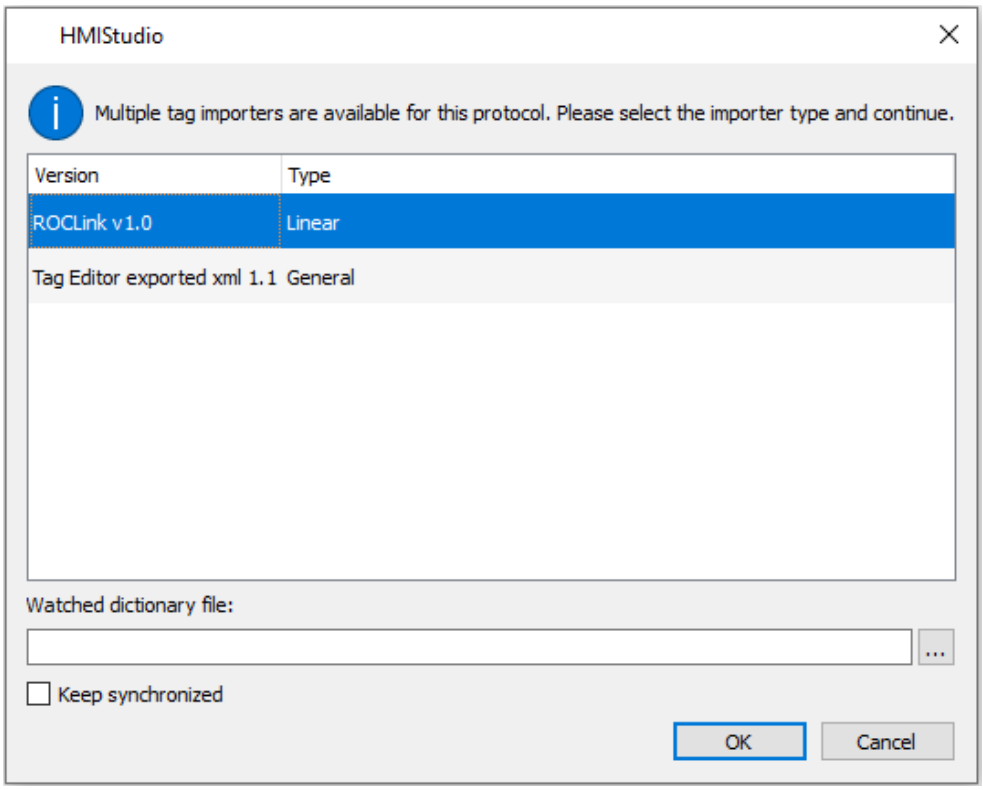
Tag Import

The ROC Plus driver, support the generic import of tags when provided in XML. Import procedure is described.

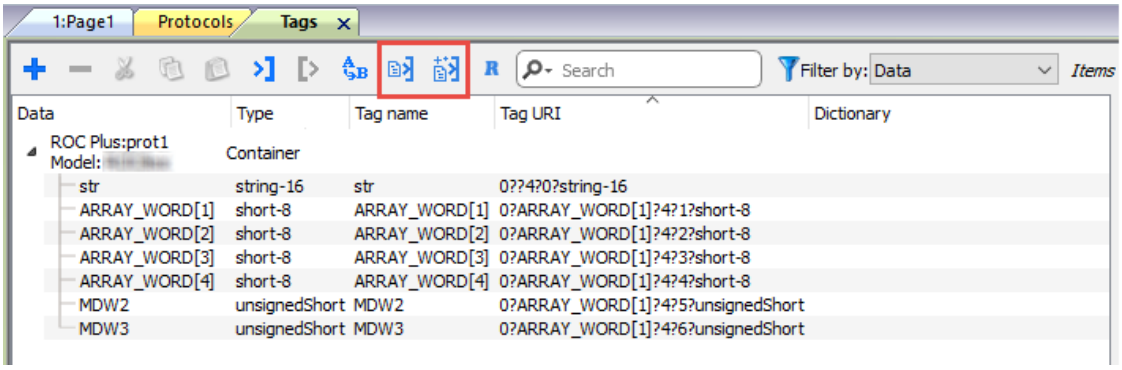
- make a copy of saved configuration file “.800” and rename as “.MDB”
- open the “.MDB” using Microsoft Access
- export the table "tConfigData" to a XML file choosing XML format



- In the tag editor select the driver and click on the “Import tag” button to start the importer.



- Locate the ".XML" file and confirm. The tags present in the exported document are listed in the tag dictionary from where they can be directly added to the project using the add tags button as shown in figure.



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
No response	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured to get network access
Not expected response TLP	The panel did receive from the controller a response with invalid Type Logical Parameter
Can't find the TLP location	The panel can't get the physical location of the type or the logical number in the ROC800
Not expected number of items	Controller did not accept write request; ensure the data programmed in the project are consistent with the controller resources
Wrong datagram data length	The panel did receive from controller a response frame contains wrong data length
PLC is in the firmware update mode	Firmware Update Mode – Extremely limited functionality is available.
Not expected response length	The panel did receive from the controller a response with invalid message length
Can't read port security mode	Security Access Mode for the port the request was not received
Can't read compatibility mode	Logical Compatibility Mode was not received

Error	Notes
Can't get IO point types	The ROC Plus database is broken into individual parameters. Each database parameter is uniquely associated by parameter number and point The panel did not receive the requested Point Type
Can't send the request	The panel cannot sent any request to the controller
Not expected response group/unit	The panel did receive from the controller a response with invalid Group/Unit
Not expected opcode in the response	The panel did receive from the controller a response contains an unexpected operation code action to perform.
Invalid format received	The panel did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
Message checksum error	The panel did receive from the controller a response contains an invalid checksum

SAIA S-BUS

The SAIA S-BUS communication driver has been designed to connect HMI devices to SAIA PLCs through serial connection.

HMIs from UN65 and UN70 platforms do not support PARITY mode on PLC configuration due hardware incompatibility.
DATA mode is supported in all HMI platforms.

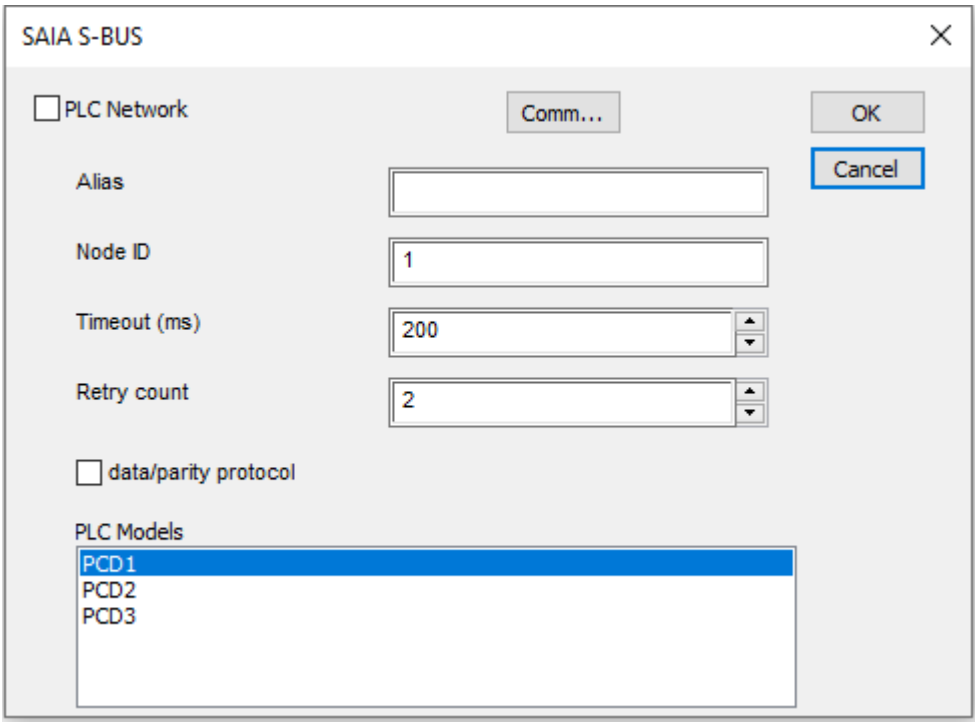
Protocol Editor Settings

Adding a protocol

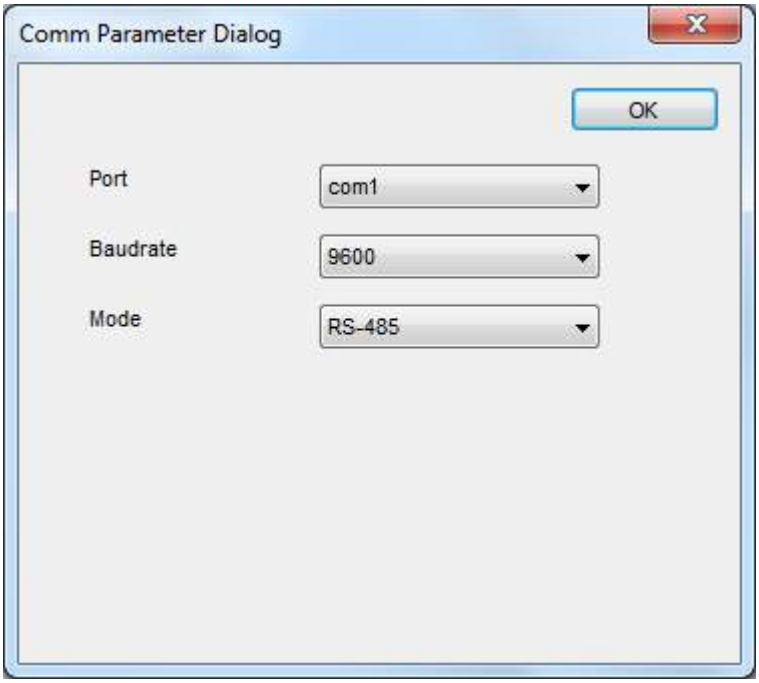
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
Node ID	SAIA PLC node on the serial network.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
Retry count	Defines the number of times a certain message will be sent to the controller

Element	Description
	before reporting the communication error status.
data/parity protocol	SAIA protocol mode: • **unchecked** (default): parity mode • **checked**: data mode
PLC Models	SAIA PLC models available: • **PCD1** • **PCD2** • **PCD3**
Comm...	If clicked displays the communication parameters setup dialog. 

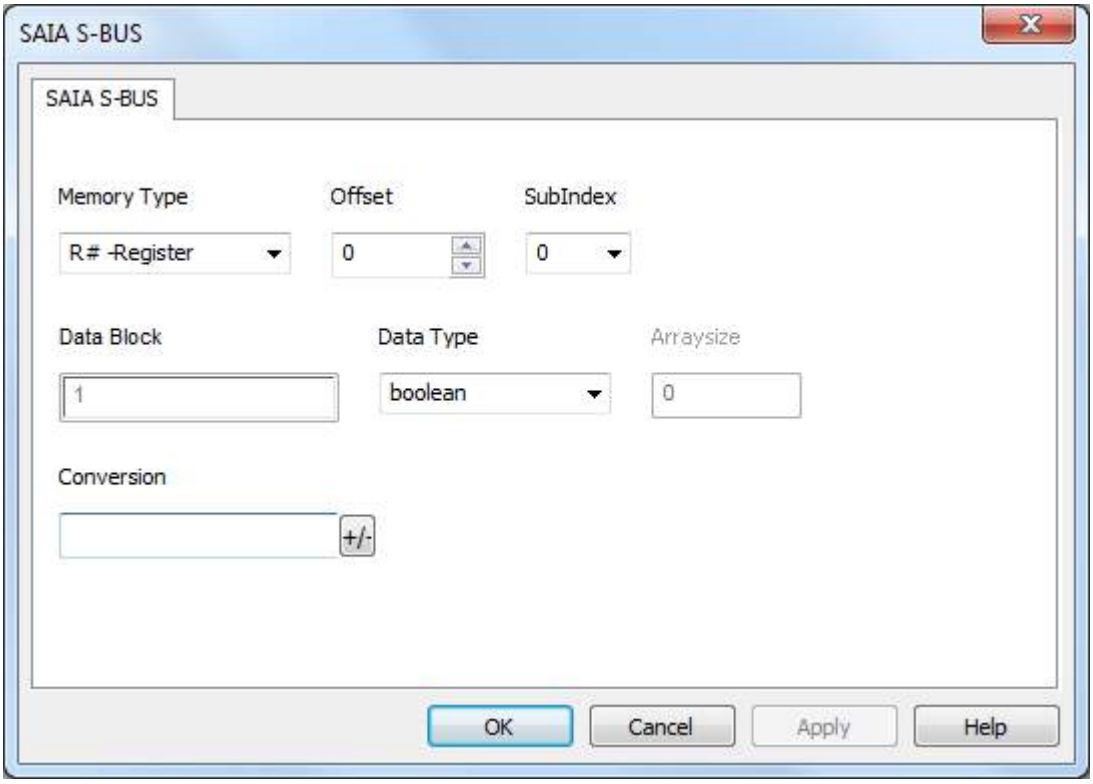
Element	Parameter
Port	Serial port selection. • **COM1**: device PLC port. • **COM2**: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board. • **COM3:** optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.
Baudrate	Serial baudrate. Available speeds:

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Element</th><th>Parameter</th></tr>
<tr><td></td><td>• 9600.
• 19200.
• 38400.
• 57600.</td></tr>
<tr><td>Mode</td><td>Serial port mode. Available modes:
• RS-232.
• RS-485 (2 wires).
• RS-422 (4 wires).</td></tr>
</table>
</td></tr>
<tr><td>PLC Network</td><td>Multiple controllers can be connected to one HMI device. To set-up multiple connections, select PLC networkand click Add to configure each node</td></tr>
</table>

Tag Editor Settings

*Path: **ProjectView**> **Config** > double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **SAIA S-BUS** from the **Driver** list: tag definition dialog is displayed.



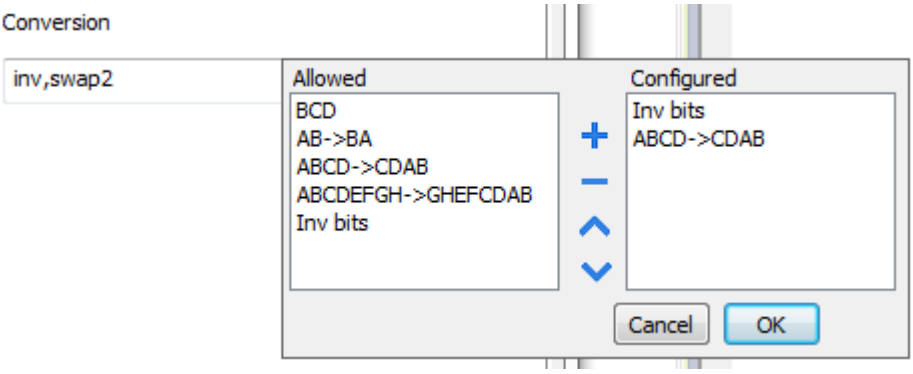
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>R # -Register</td><td>unsigned 32 bit data register (default)</td></tr>
<tr><td>C # -Counter</td><td>unsigned 32 bit data counter (default)</td></tr>
<tr><td>T # -Timer</td><td>unsigned 32 bit data timer (default)</td></tr>
<tr><td>F # -Flag</td><td>1 bit data flag</td></tr>
<tr><td>I # -Input</td><td>1 bit data input</td></tr>
<tr><td>O # -Output</td><td>1 bit data output</td></tr>
<tr><td>Data Block</td><td>unsigned 32 bit data block (default)</td></tr>
<tr><td>Real Time Clock</td><td>unsigned 8 bit real time clock (default) (see Special Data Types for mode details)</td></tr>
<tr><td>Node Override</td><td>protocol parameter (see Special Data Types for mode details)</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>
<table>
<tr><th>Memory Type</th><th>Offset PCD1</th><th>Offset PCD2</th><th>Offset PCD3</th></tr>
<tr><td>R # -Register</td><td>0 – 4095</td><td>0 – 4095</td><td>0 – 16383</td></tr>
<tr><td>C # -Counter</td><td>0 – 1599</td><td>0 – 1599</td><td>0 – 1599</td></tr>
<tr><td>T # -Timer</td><td>0 – 1599</td><td>0 – 1599</td><td>0 – 1599</td></tr>
<tr><td>F # -Flag</td><td>0 – 8191</td><td>0 – 8191</td><td>0 – 8191</td></tr>
<tr><td>I # -Input</td><td>0 – 512</td><td>0 – 8192</td><td>0 – 5120</td></tr>
<tr><td>O # -Output</td><td>0 – 512</td><td>0 – 8192</td><td>0 – 5120</td></tr>
<tr><td>Data Block</td><td>0 – 3333</td><td>0 – 3333</td><td>0 – 16383</td></tr>
<tr><td>Real Time Clock</td><td>1 – 8</td><td>1 – 8</td><td>1 – 8</td></tr>
<tr><td>Node Override</td><td>0</td><td>0</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>

float
string

See "Programming concepts" section in the main manual.
Note: To define arrays, select one of Data Type format followed by square brackets.
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Inv bits</td><td>inv: Invert all the bits of the tag.

Example:
1001 → 0110 (in binary format)
9 → 6 (in decimal format)</td></tr>
<tr><td>Negate</td><td>neg: Set the opposite of tag value.

Example:
25.36 → -25.36</td></tr>
<tr><td>AB -> BA</td><td>swapnibbles: Swap nibbles in a byte.

Example:
15D4 → 514D (in hexadecimal format)</td></tr>
</table>
</td></tr>
</table>

Element	Description

Value	Description
	5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Real Time Clock

The protocol provides the special data type Real Time Clock which allows you to change the date and time on PLC. This memory type is an unsigned byte.

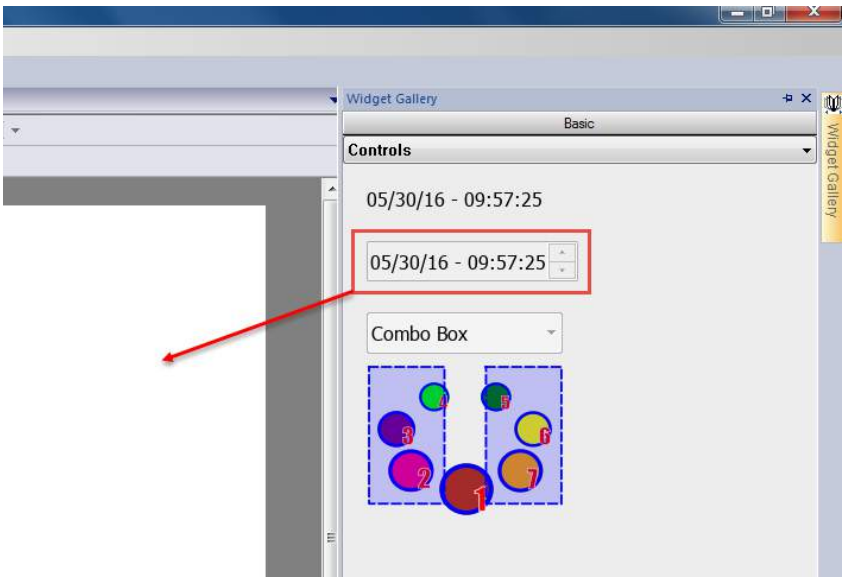
Offset	Description
1	Number of week
2	Day of week

Offset	Description
3	Year
4	Month
5	Day
6	Hours
7	Minutes
8	Seconds

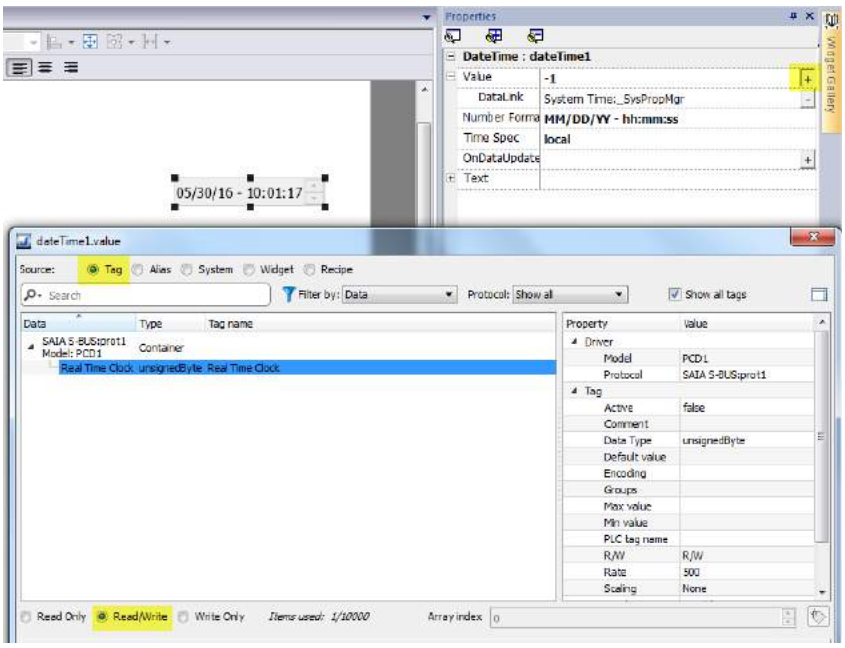
It is also possible to use the Date/Time control widget to directly write in Real Time Clock variable.

1) Define a Real Time Clock, as per above picture

2) Drag and drop the Date/Time control widget



3) From Property Pane, click on the + button beside **Value** property. Then locate the Real Time Clock variable from Tag source, and select Read/Write option.



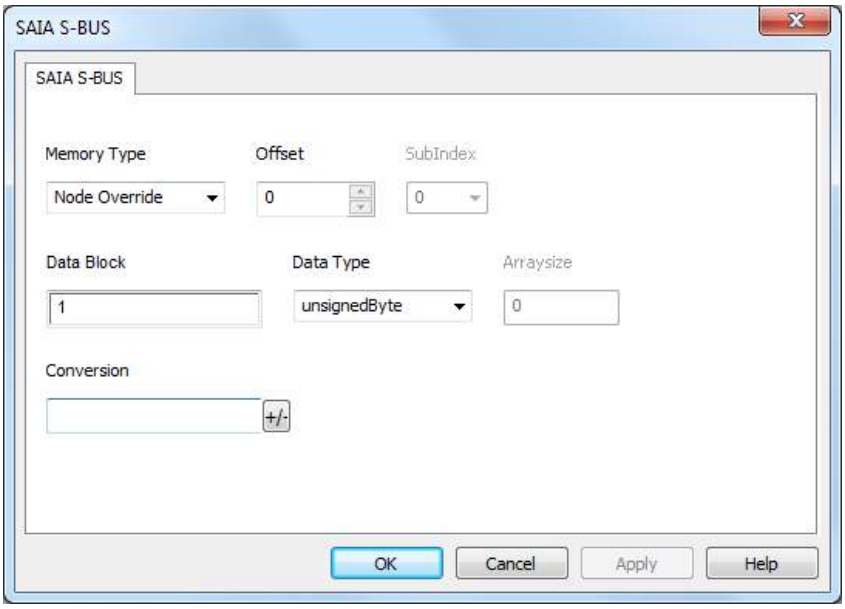
Node Override

The protocol provides the special data type Node Override which allows you to change the node ID of the slave at runtime. This memory type is an unsigned byte.

The node Override is initialized with the value of the node ID specified in the project at programming time.

Node Override	Description
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.

Note: Node Override ID value assigned at runtime is retained through power cycles.



Communication Status

The current communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The codes supported for this communication driver are:

Error	Notes
NAK	Returned in case the controller replies with a not acknowledge
Timeout	Returned when a request is not replied within the specified timeout period; ensure the controller is connected and properly configured for communication
Line Error	Returned when an error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits); ensure the communication parameter settings of the controller is compatible with panel communication setup
Invalid response	The panel did receive from the controller a response, but its format or its contents is not as expected; ensure the data programmed in the project are consistent with the controller resources

SAIA S-BUS ETH

The SAIA S-BUS ETH communication driver has been designed to connect HMI devices to SAIA PLCs through ethernet connection.

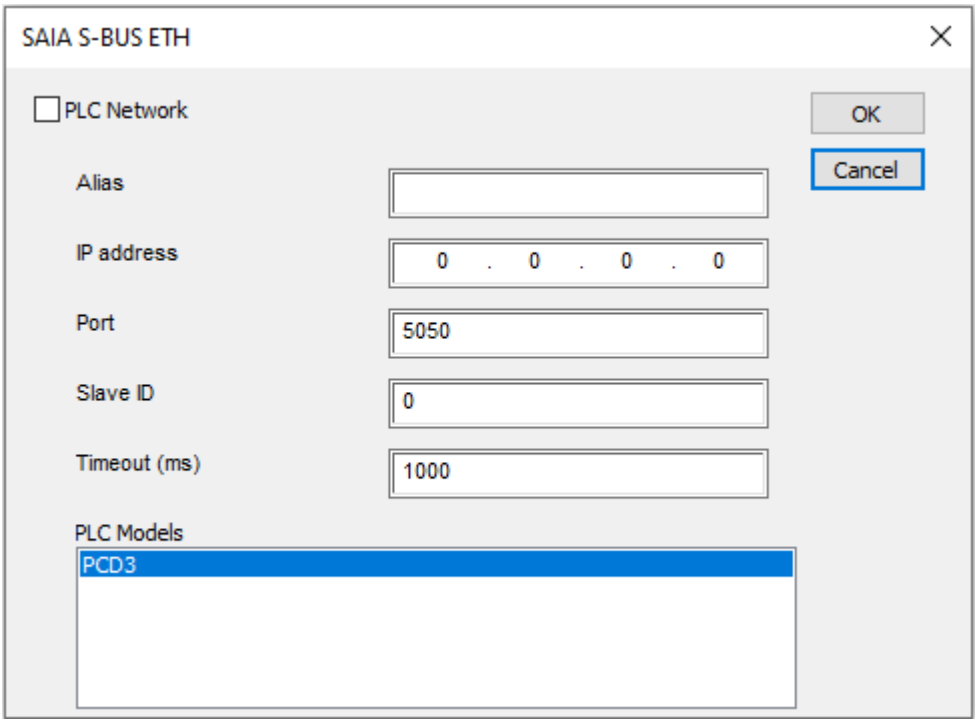
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



Element	Description
IP address	Ethernet IP address of the controller.
Port	Port number used by the driver. The default value is **5050**.
Slave ID	ID if the controller.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server device.
PLC Models	SAIA PLC models available:

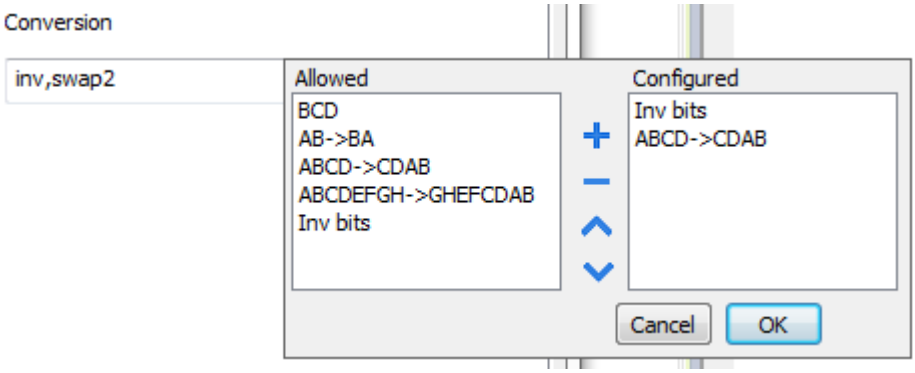
Element	Description
	• **PCD3**
PLC Network	Multiple controllers can be connected to one HMI device. To set-up multiple connections, select **PLC network**and click **Add** to configure each node

Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **SAIA S-BUS ETH** from the **Driver** list: tag definition dialog is displayed.

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>
<table>
<tr><th>Memory Type</th><th>Description</th></tr>
<tr><td>R # -Register</td><td>unsigned 32 bit data register (default)</td></tr>
<tr><td>C # -Counter</td><td>unsigned 32 bit data counter (default)</td></tr>
<tr><td>T # -Timer</td><td>unsigned 32 bit data timer (default)</td></tr>
<tr><td>F # -Flag</td><td>1 bit data flag</td></tr>
<tr><td>I # -Input</td><td>1 bit data input</td></tr>
<tr><td>O # -Output</td><td>1 bit data output</td></tr>
<tr><td>Data Block</td><td>unsigned 32 bit data block (default)</td></tr>
<tr><td>Real Time Clock</td><td>unsigned 8 bit real time clock (default) (see Special Data Types for mode details)</td></tr>
</table>
</td></tr>
<tr><td>Offset</td><td>
<table>
<tr><th>Memory Type</th><th>Offset</th></tr>
<tr><td>R # -Register</td><td>0 – 16383</td></tr>
<tr><td>C # -Counter</td><td>0 – 1599</td></tr>
<tr><td>T # -Timer</td><td>0 – 1599</td></tr>
<tr><td>F # -Flag</td><td>0 – 8191</td></tr>
<tr><td>I # -Input</td><td>0 – 5120</td></tr>
<tr><td>O # -Output</td><td>0 – 5120</td></tr>
<tr><td>Data Block</td><td>0 – 16383</td></tr>
<tr><td>Real Time Clock</td><td>1 – 8</td></tr>
</table>
</td></tr>
<tr><td>SubIndex</td><td>This allows resource offset selection within the register.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
string</td></tr>
</table>

Element	Description
	See "Programming concepts" section in the main manual. Note: To define arrays, select one of Data Type format followed by square brackets.
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word.

Element	Description

Value	Description
	Example: 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select conversion and click +. The selected item will be added to list **Configured**.

If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**).

Use the arrow buttons to order the configured conversions.

Real Time Clock

The protocol provides the special data type Real Time Clock which allows you to change the date and time on PLC. This memory type is an unsigned byte.

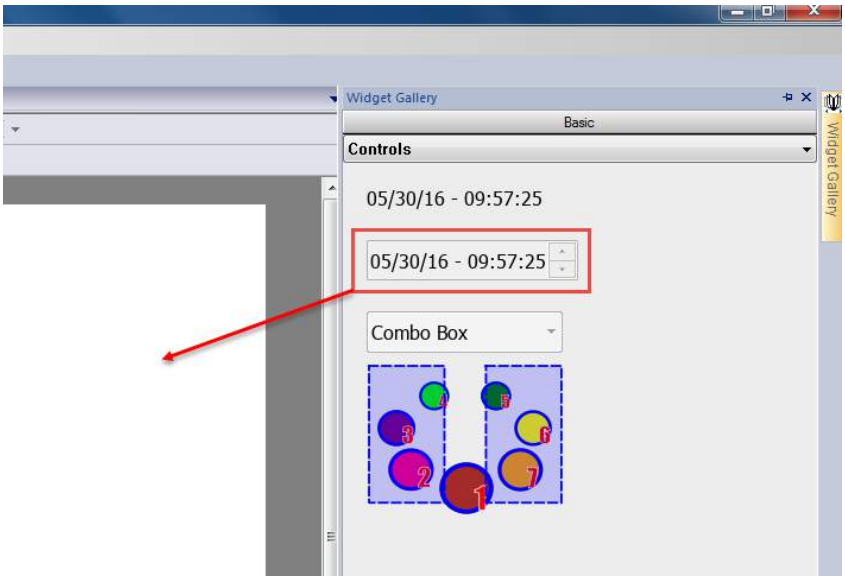
Offset	Description
1	Number of week
2	Day of week
3	Year
4	Month

Offset	Description
5	Day
6	Hours
7	Minutes
8	Seconds

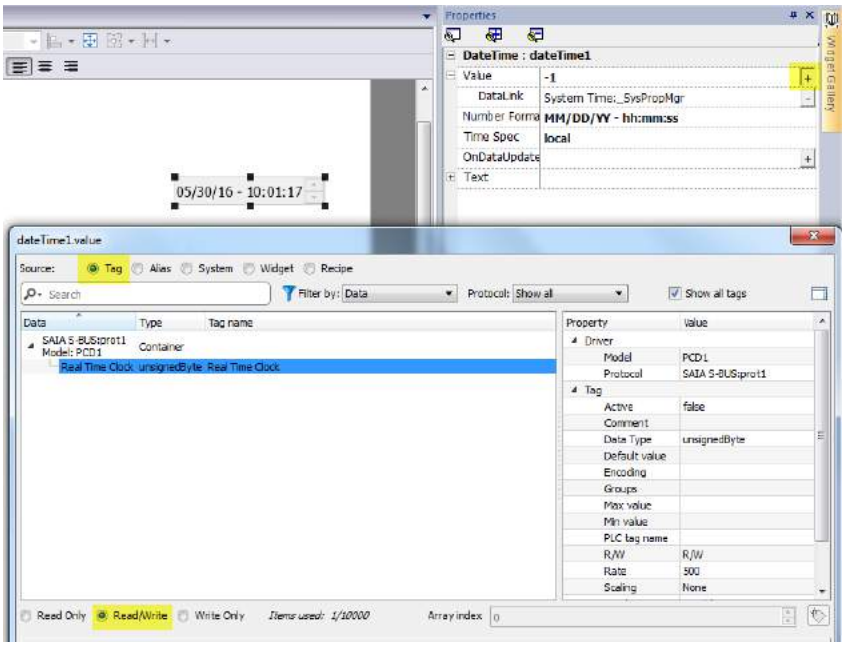
It is also possible to use the Date/Time control widget to directly write in Real Time Clock variable.

1) Define a Real Time Clock, as per above picture

2) Drag and drop the Date/Time control widget



3) From Property Pane, click on the + button beside **Value** property. Then locate the Real Time Clock variable from Tag source, and select Read/Write option.



Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid response	The device did received a response with invalid format or contents from the controller.	Check if the data programmed in the project are consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

Simatic S7 PPI

HMI devices can be connected to the Siemens Simatic S7-200 family of PLCs. The communication is performed via the PLC programming ports using the PPI and the PPI+ protocols.

This document describes the PPI+ protocol and includes the information needed for a successful connection.

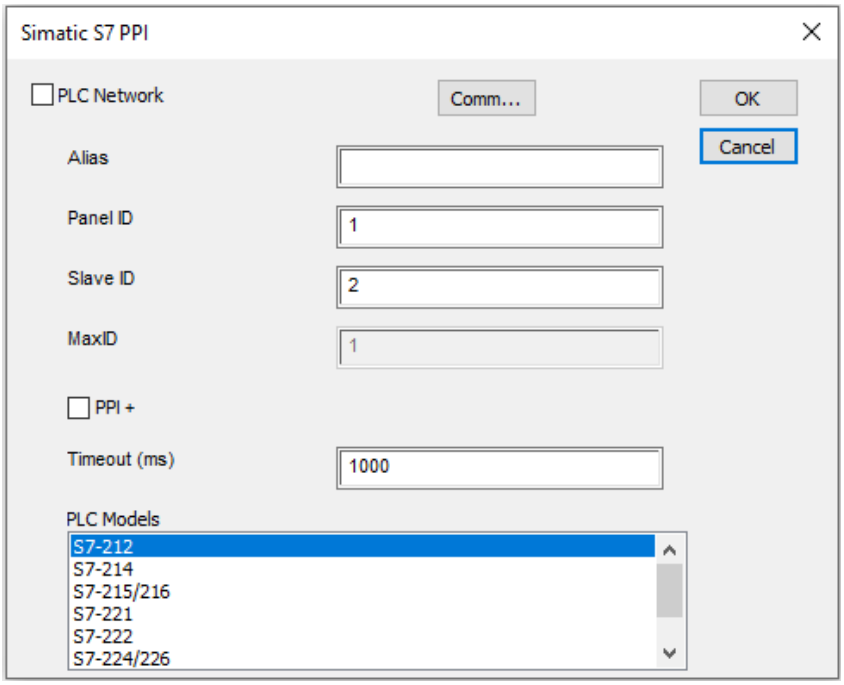
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

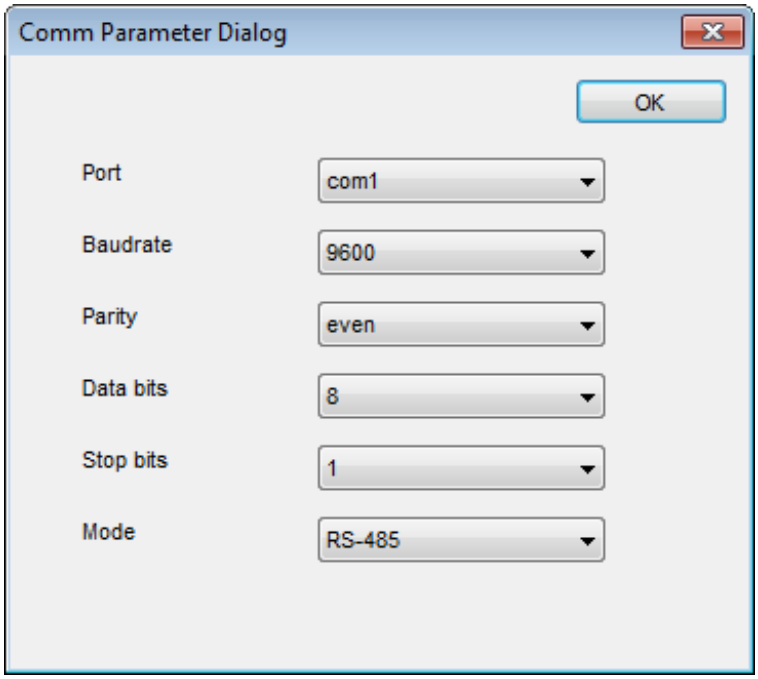
The protocol configuration dialog is displayed.



Element	Description
PLC Network	Enable access to multiple networked controllers. For every controller (slave) set the proper option.
Panel ID	Node number of the operator panel.
Slave ID	Node number of the connected PLC.
Max ID	Available only if PPI+ protocol is in use. Contains the highest node number in PPI+ network.

Element	Description
PPI+	Checked to use PPI+ protocol instead of PPI protocol.
Timeout (ms)	Time delay in milliseconds between two retries of the same message when no answer is received from the controller.

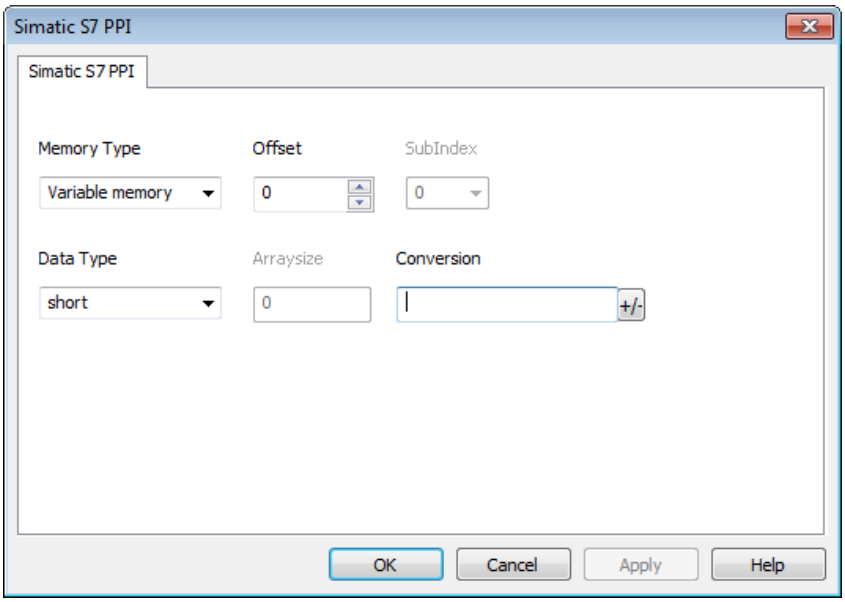
Element	Description
PLC Models	Several Siemens controllers are supported. Please check directly in the programming IDE software for a complete list of supported controllers.
Comm…	If clicked displays the communication parameters setup dialog.



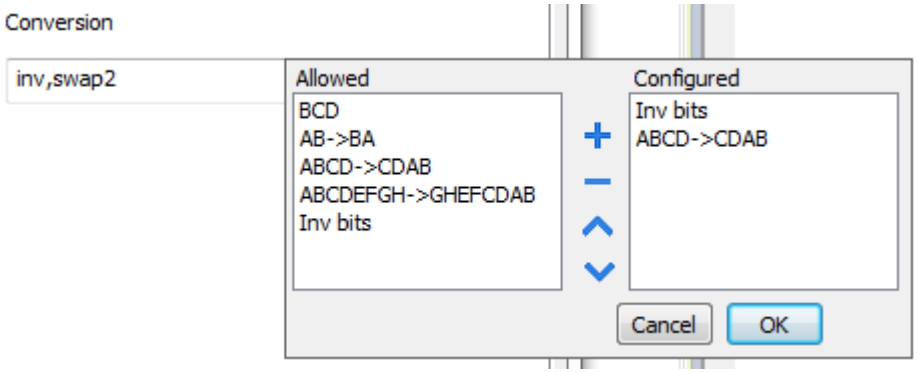
Element	Parameter
Port	Serial port selection. On UN20: • **COM1**: device PLC port. • **COM2**: PC/printer port On UN31 or UN30: • **COM1**: integrated serial port • **COM2**: optional module plugged on Slot 1/2 • **COM3**: optional module plugged on Slot 3/4
Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.
Mode	Serial port mode. Available modes: • **RS-232**. • **RS-485** (2 wires). • **RS-422** (4 wires).

Tag Editor Settings

In the Tag Editor select Simatic S7 PPI from the list of defined protocols and click **+** to add a tag.



Element	Description
Memory Type	Area of PLC where tag is located.
Offset	Offset address where tag is located.
SubIndex	In case of Boolean data type, this is the offset of single bit.
Data Type	Available data types: • **boolean** • **byte** • **short** • **int** • **unsignedByte** • **unsignedShort** • **unsignedInt** • **float** • **string** See "Programming concepts" section in the main manual.
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character

Element	Description
	requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example:

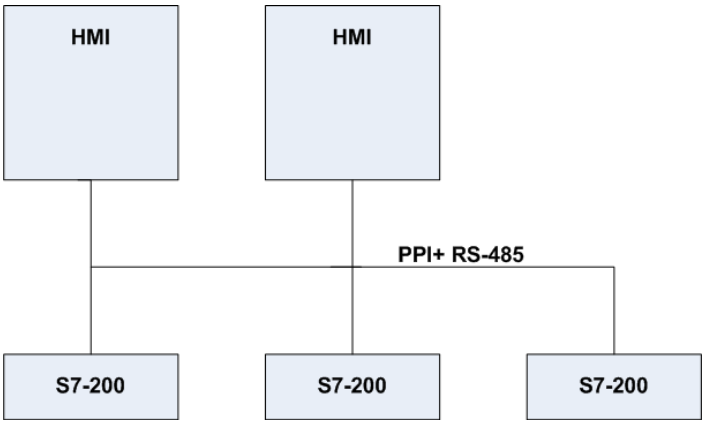
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000001
→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>
Select conversion and click +. The selected item will be added to list Configured.
If more conversions are configured, they will be applied in order (from top to bottom of list Configured).
Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

PPI+ Connectivity

HMI devices can be connected to more than one CPU S7-200, more than one operator panel can also be connected to the same PLC.

Operator panels will not interfere with PPI+ communication between the PLC's.



PPI+ protocol allows you to use more complex configurations than the standard PPI protocol.

Each PLC can execute read and write operations to and from other PLCs. At the same time more than one panel can be connected on the PPI network and can access all the variables from all the PLCs.

PLC programming software can be used and online programming can be performed without interfering with the panel-PLC communication .

Communication Status

Current communication status can be displayed using System Variables. See "System Variables" section in the main manual.

Codes supported for this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid response	The device did received a response with invalid format or contents from the controller .	Ensure the data programmed in the project are consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

Siemens S7 Optimized

Siemens S7 Optimized communication driver has been designed to communicate with Siemens PLCs through Ethernet connection.

PLC must either have an on-board Ethernet port or be equipped with an appropriate Ethernet interface (either built-in or with a module).

This communication driver allows communication with PLCs which have been programmed using optimized Data Blocks.

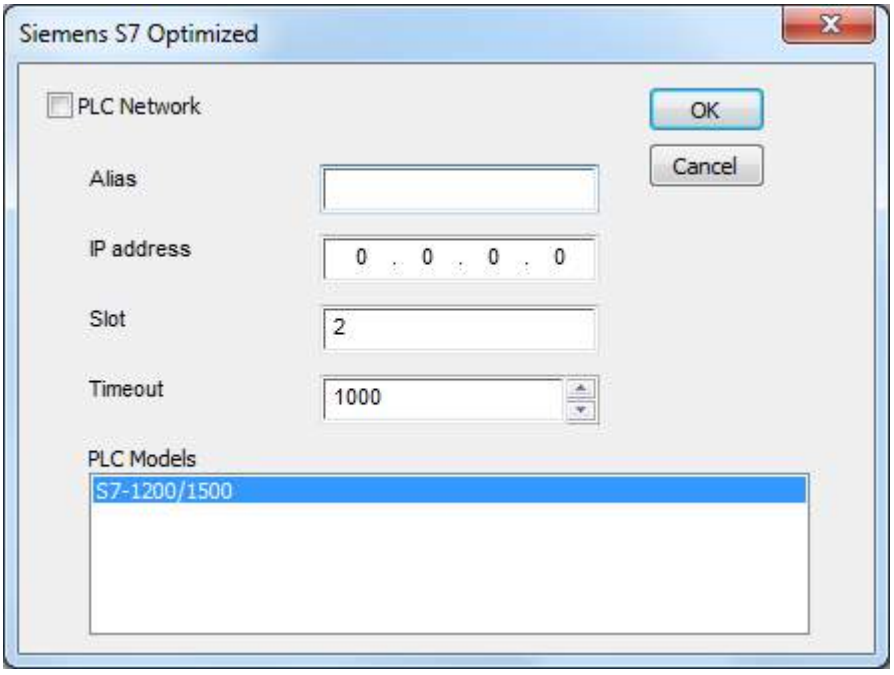
Protocol Editor Settings

Adding a protocol

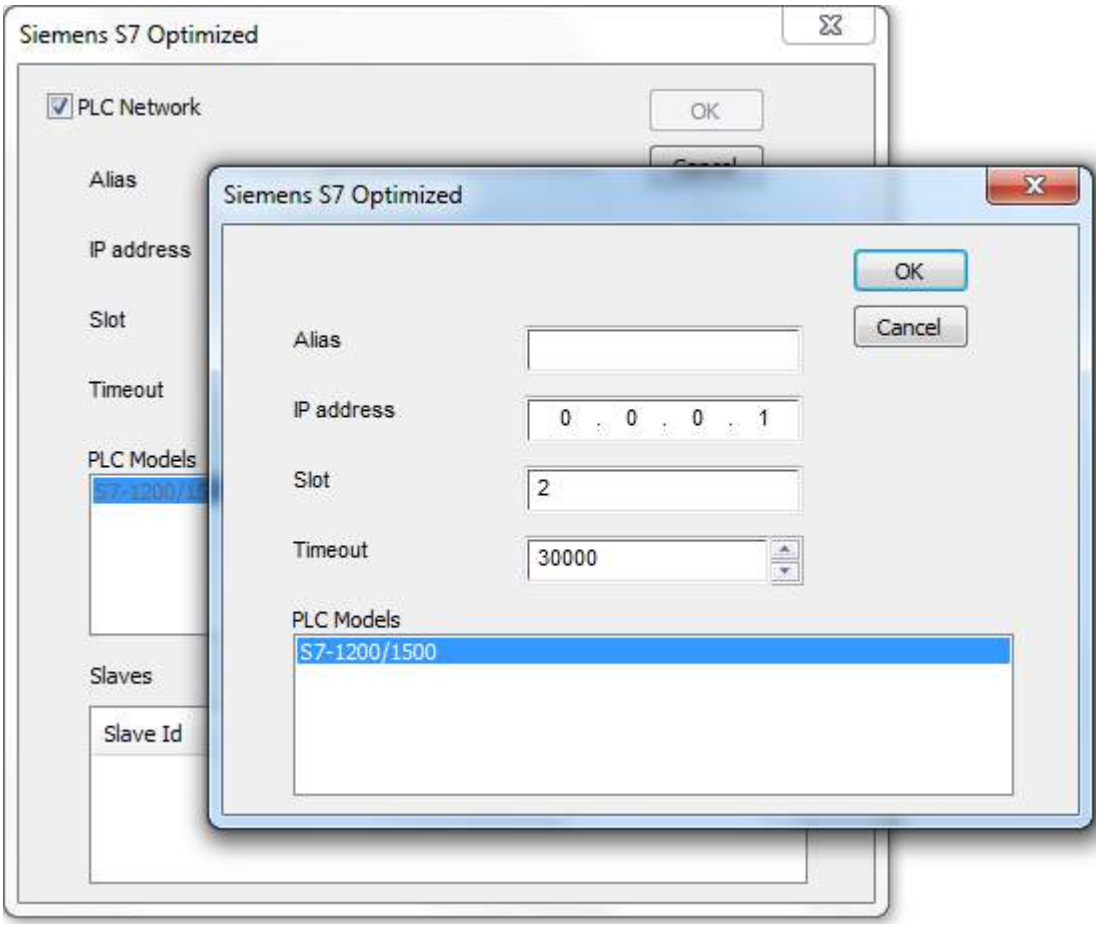
To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



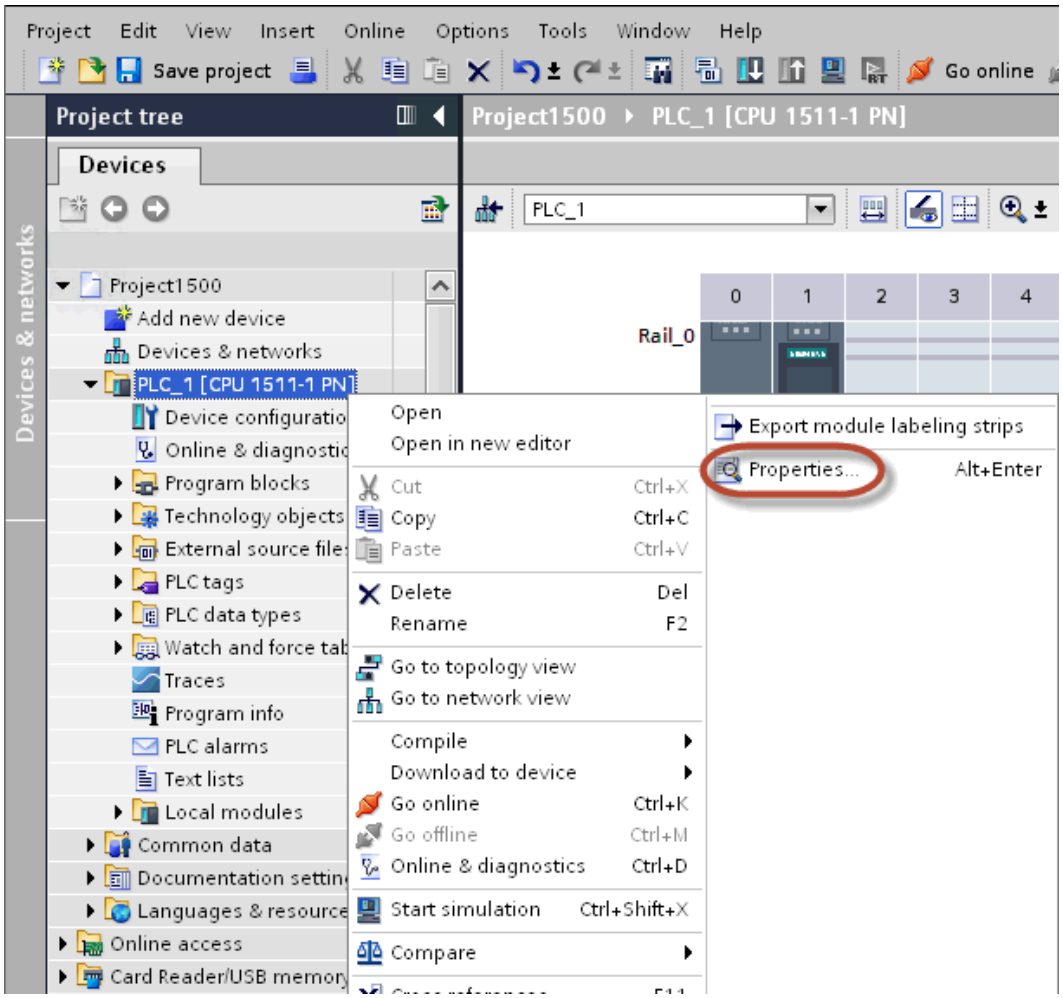
Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP	Ethernet IP address of PLC.

Element	Description
address	
Slot	Number of the slot where the CPU is mounted.
PLC Models	List of compatible PLCs.
PLC Network	Enable access to multiple networked PLCs. For every PLC set the proper option. 

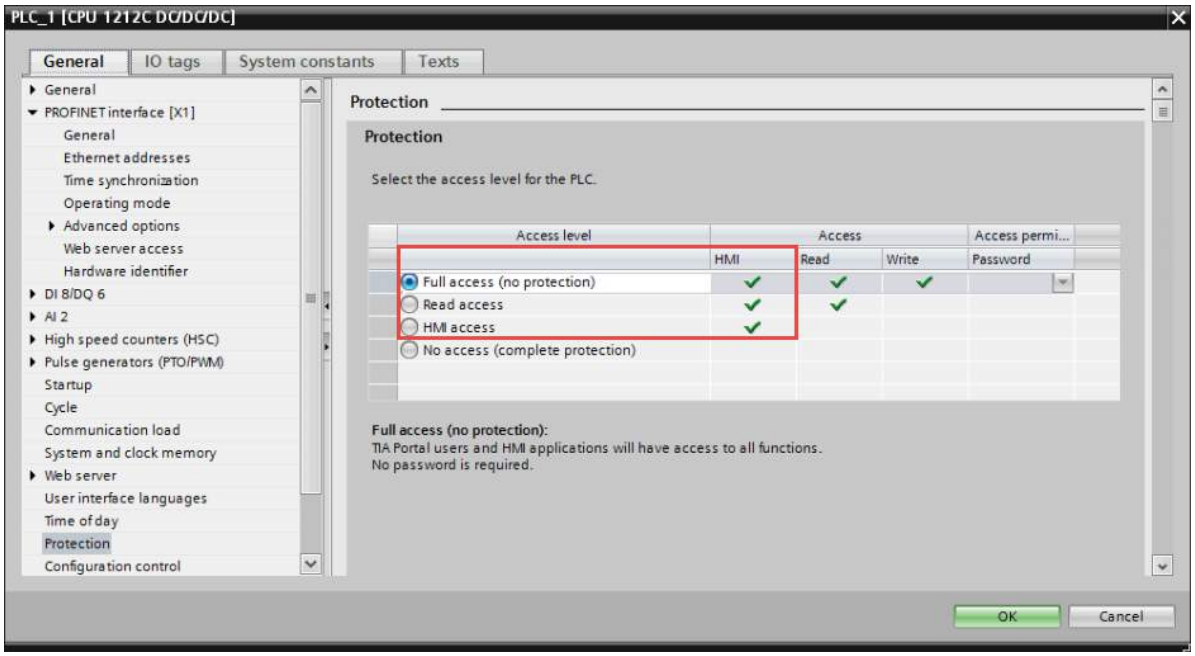
S7-1200 and S7-1500 PLC configuration

S7-1200 (starting from firmware version 4.0) and S7-1500 PLC Series from Siemens have a built-in firewall; by default the maximum protection level is enabled. To establish communication with these PLC models it is necessary to enable S7 communication with 3rd party devices; this setting is available in TIA Portal programming software.

1. Open the PLC project in TIA Portal.
2. Select the PLC from the project tree and open PLC Properties.

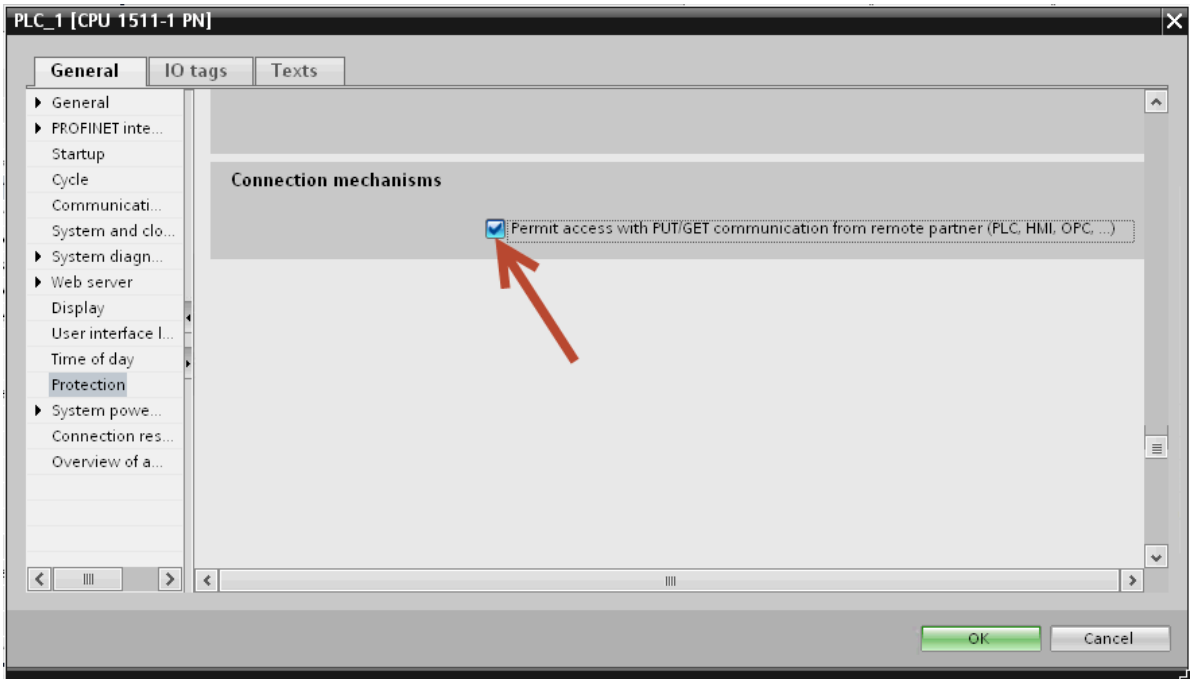


3. In General > Protection choose a permission between the top three (make sure that the tick is present on HMI column).



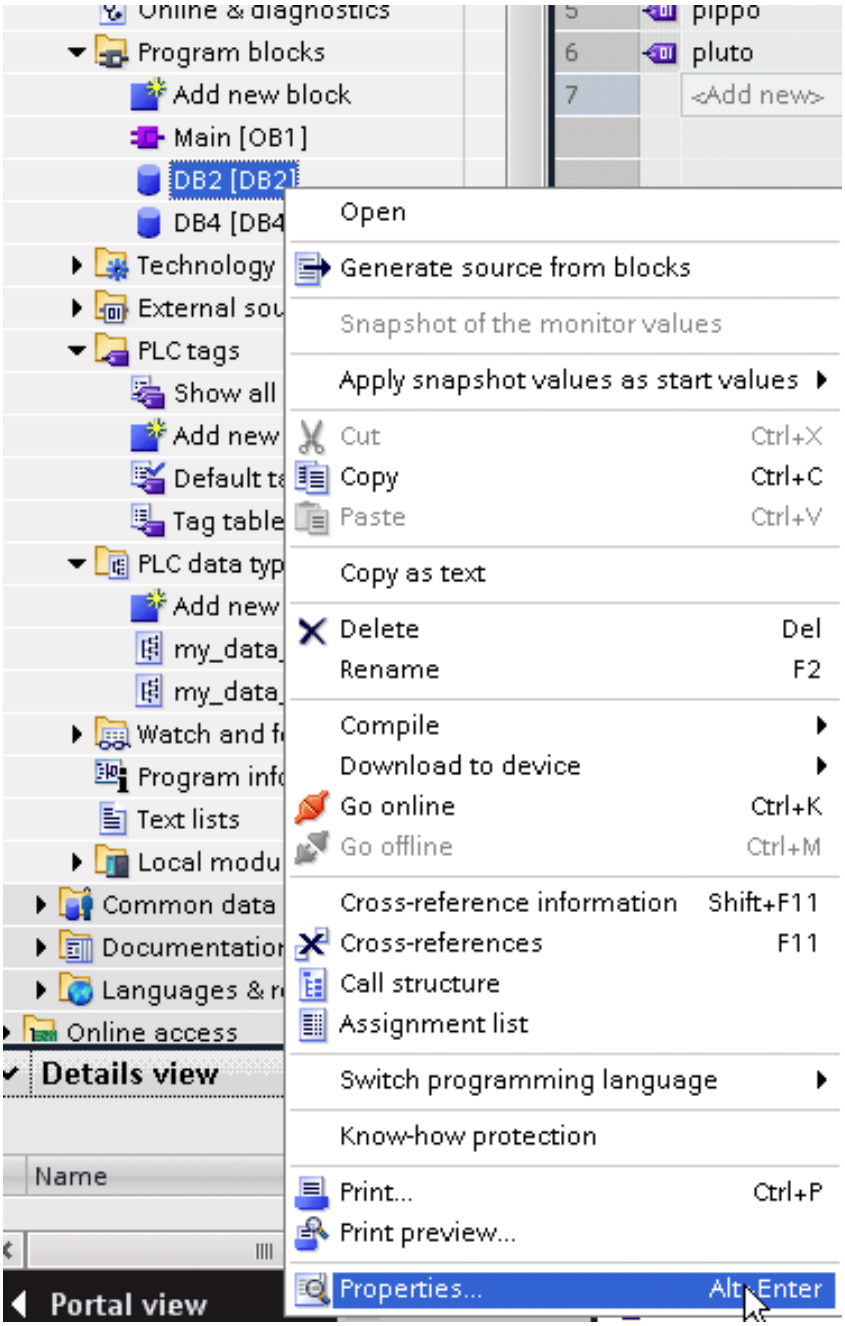
Note: If "No access" is selected, the communication with the panel will not be established.

4. Scroll down the page and check "Permit access with PUT/GET communication from remote partner".

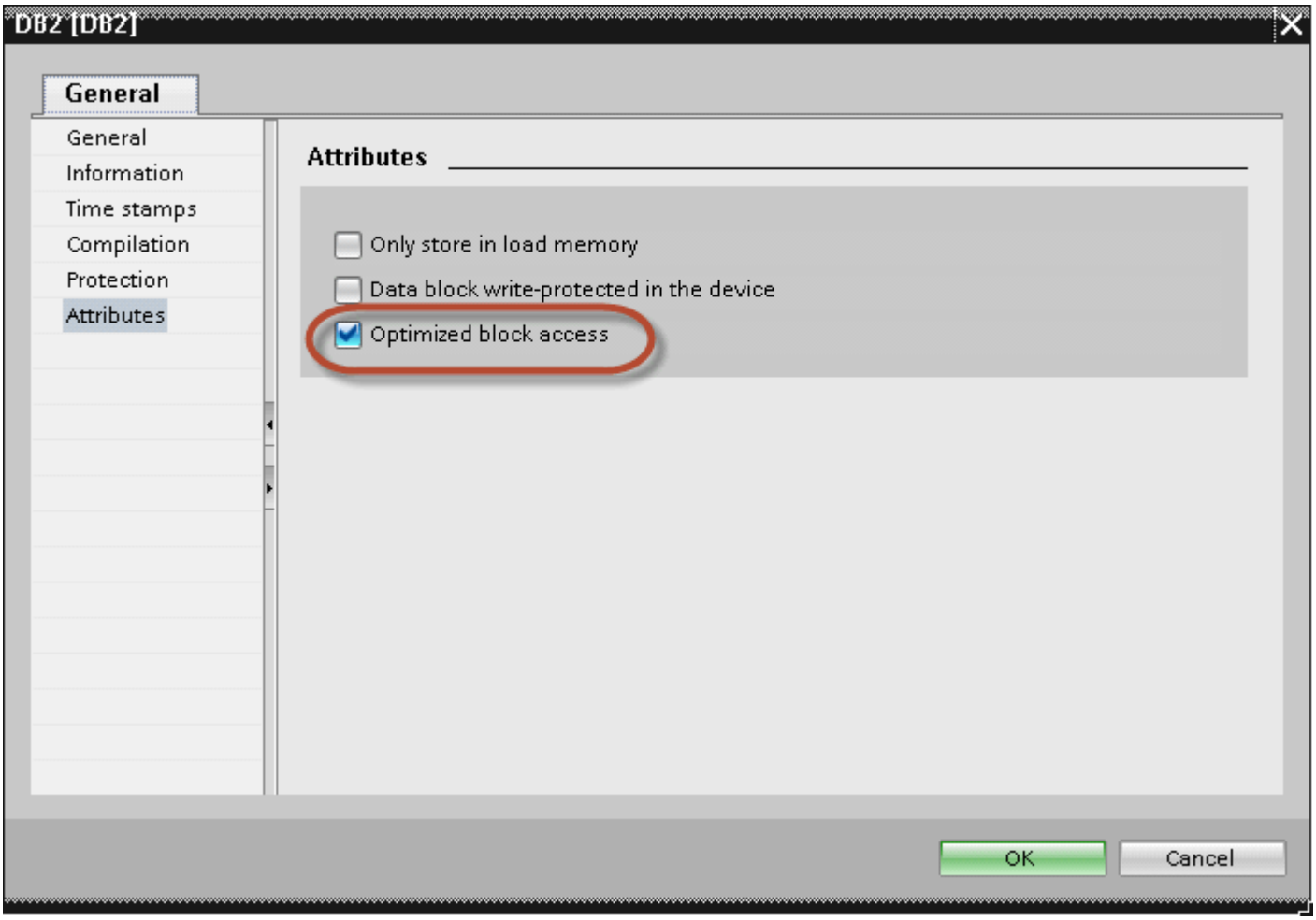


Note: If variables are defined in "Program blocks", DB must configured as "Optimized".

To check or change DB optimization, open DB Properties:



In General > Attributes check "Optimized block access":



If check box "Optimized block access" is not available (grayed-out) it could be because DB is an "instance DB" linked to an "optimized access FB".

After compiling the project, tag offsets will be shown close to variable name.

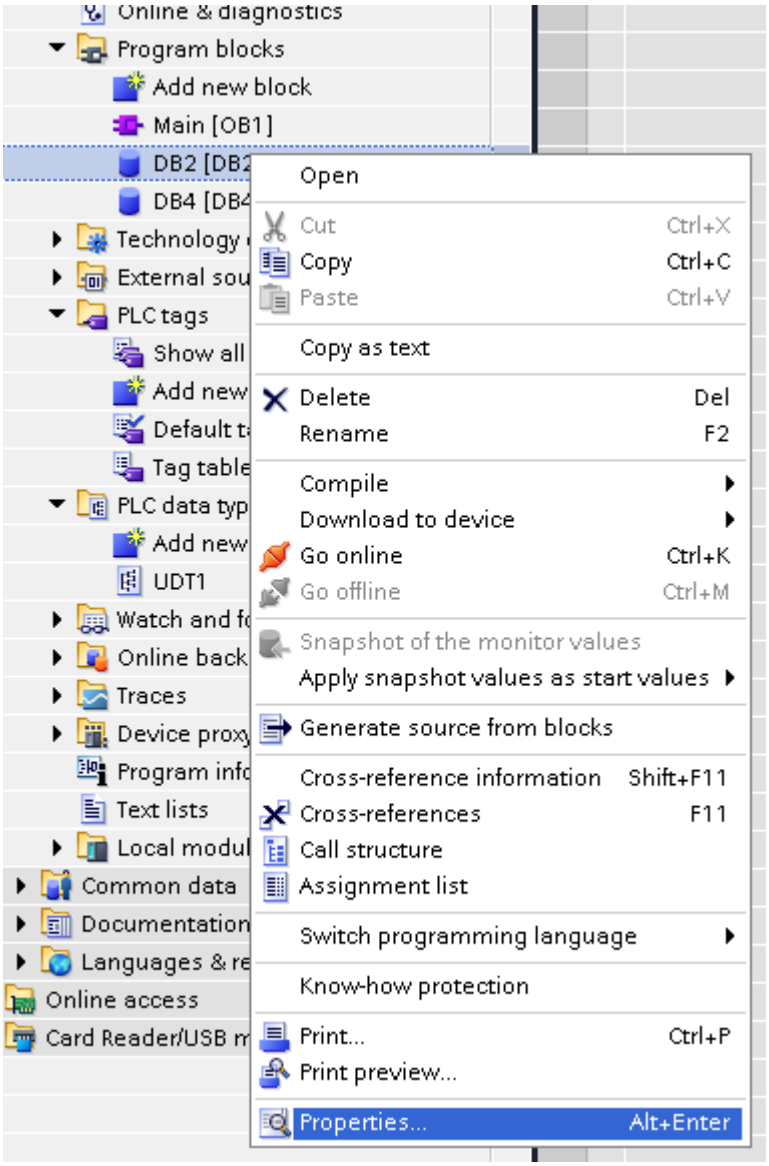
These settings can be applied to TIA Portal programming software, S7-1200 PLC family starting from PLC firmware version 4.0 and S7-1500 PLC family.

Direct Import of TIA Portal project

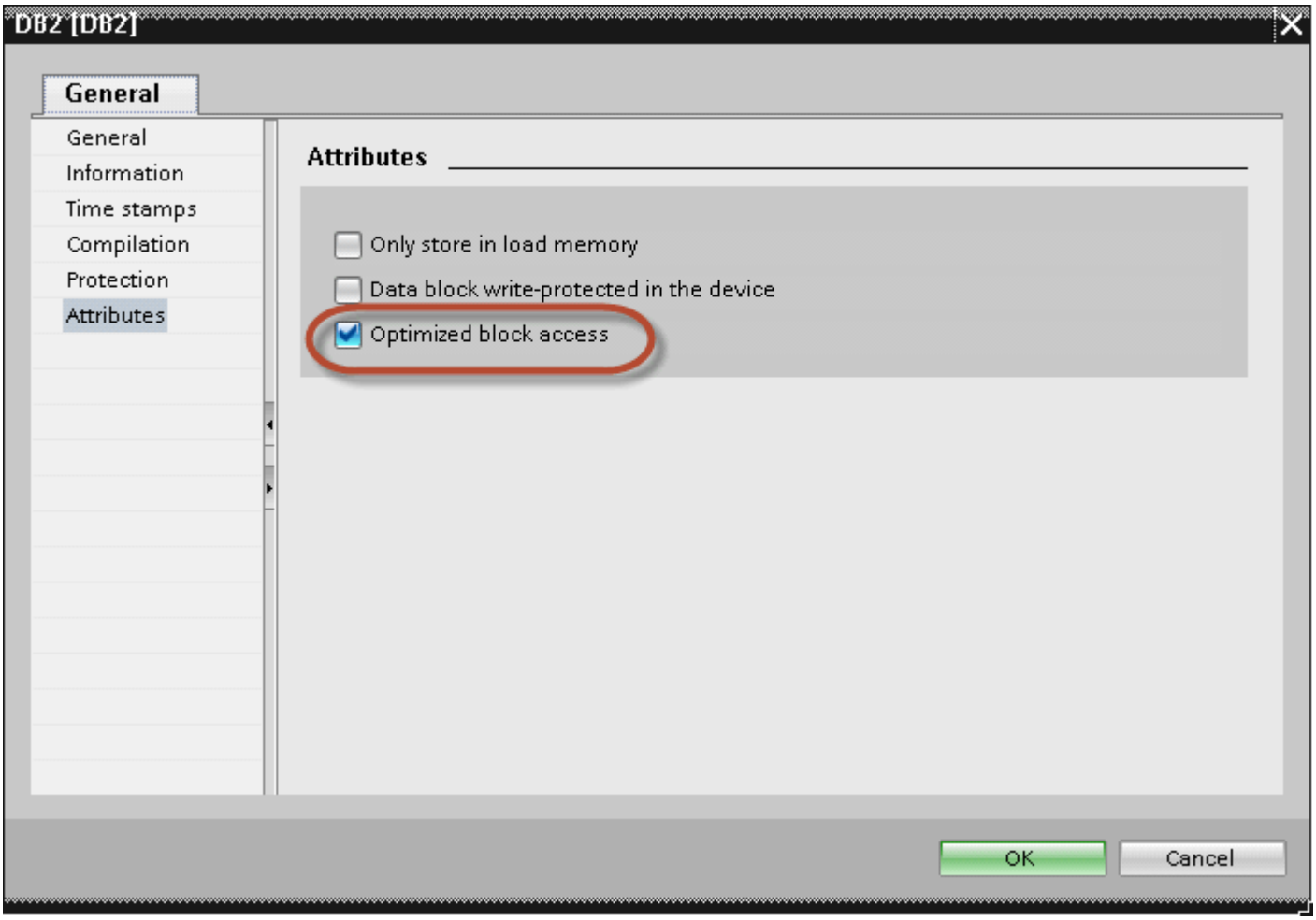
It is possible to import TIA Portal variables directly from TIA Portal project, by selecting "TIA Portal Project v12 or newer" from import selection (refer to "Tag Import" chapter).

Data Blocks must be set as Optimized:

1. Configure the Data Block as **Optimized**.
2. Right-click on the Data Block and choose **Properties**:



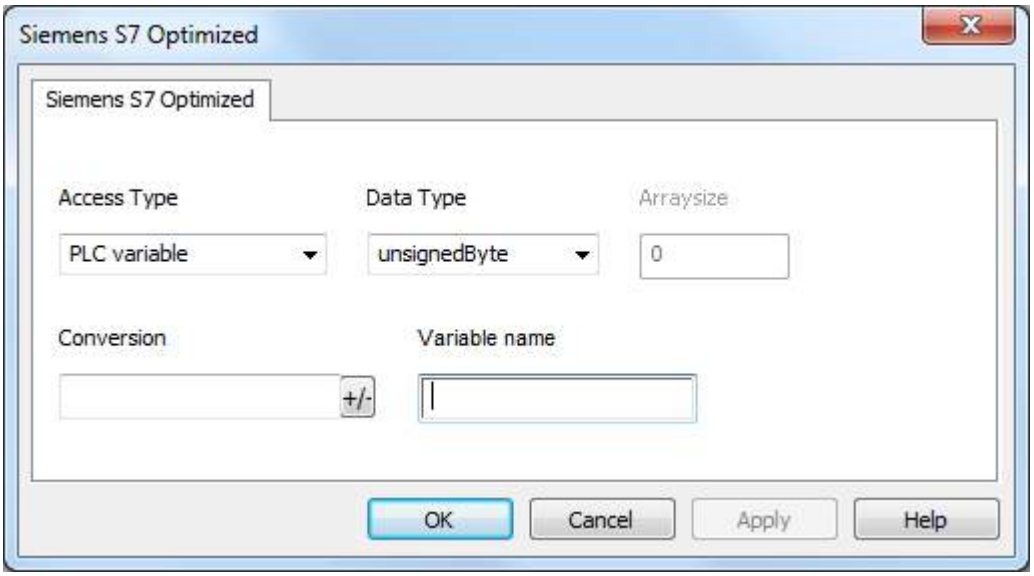
3. In the **General** tab select **Attributes** and select **Optimized block access**.



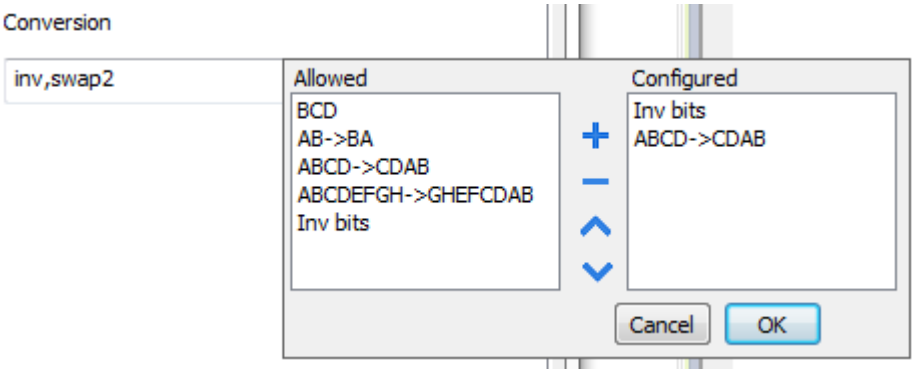
Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

Tag Editor Settings

In the Tag Editor select “Simatic S7 ETH” from the list of defined protocols and click **+** to add a tag.



Element	Description
Memory Type	Area of PLC where tag is located. **Type** / **Description** PLC variable — Variables imported from TIA Portal project. Node Override IP — Check "Special data type" chapter
Data Type	**Data Type** / **Memory Space** / **Limits** **boolean** — 1-bit data — 0 ... 1 **byte** — 8-bit data — -128 ... 127 **short** — 16-bit data — -32768 ... 32767 **int** — 32-bit data — -2.1e9 ... 2.1e9 **unsignedByte** — 8-bit data — 0 ... 255 **unsignedShort** — 16-bit data — 0 ... 65535 **unsignedInt** — 32-bit data — 0 ... 4.2e9 **float** — IEEE single-precision 32-bit floating point type — 1.17e-38 ... 3.4e38 **double** — IEEE double-precision 64-bit floating point type — 2.2e-308 ... 1.79e308 **string** — Array of elements containing character code defined by selected encoding

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the **Allowed** list shows one or more conversions, listed below.

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)

Element	Description

Value	Description
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	Separate the byte in two nibbles, and reads them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select the conversion and click on plus button. The selected item will be added on **Configured** list.

If more conversions are configured, they will be applied in order (from top to bottom of **Configured** list).

Use the arrow buttons to order the configured conversions.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

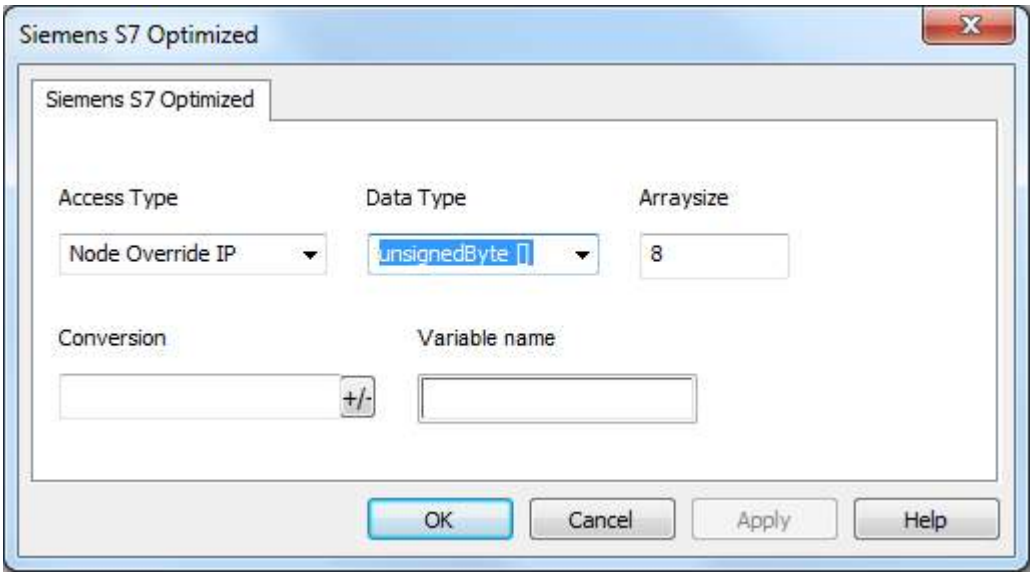
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

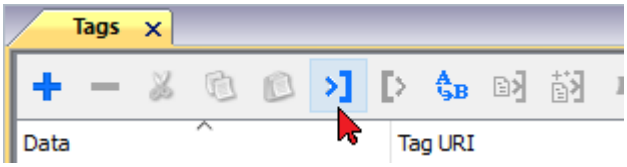
Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.

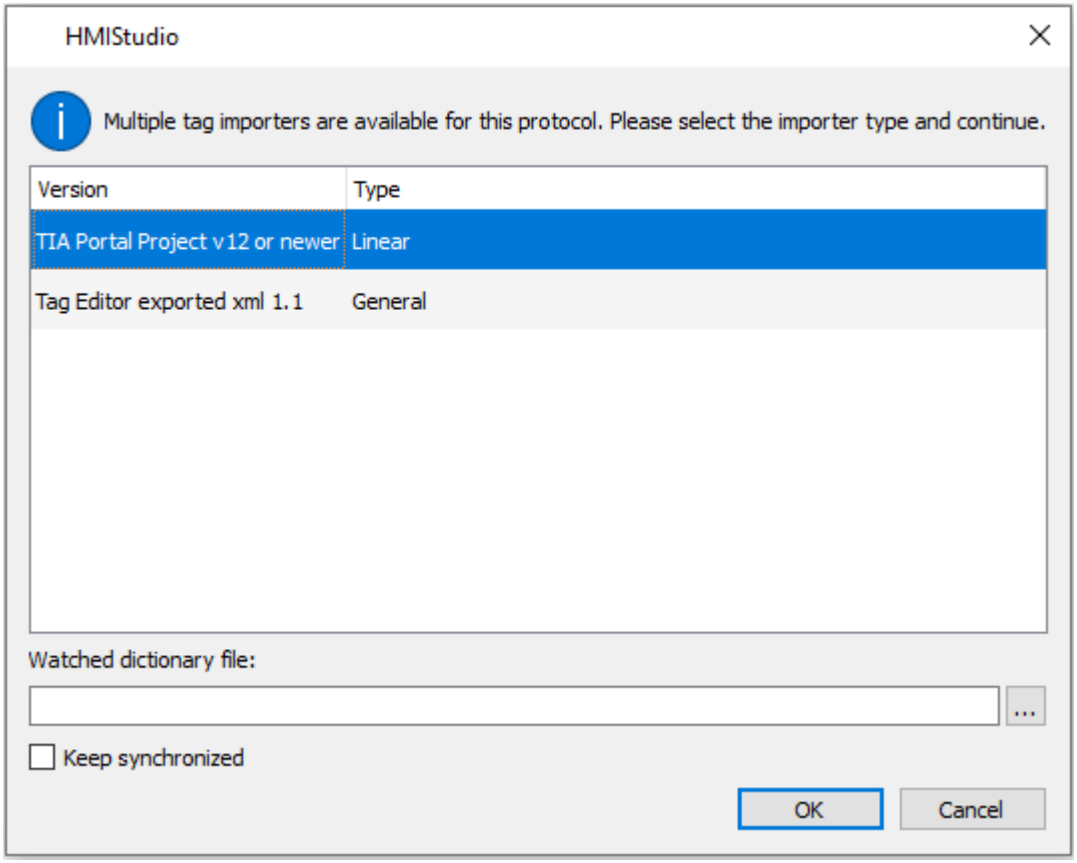


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



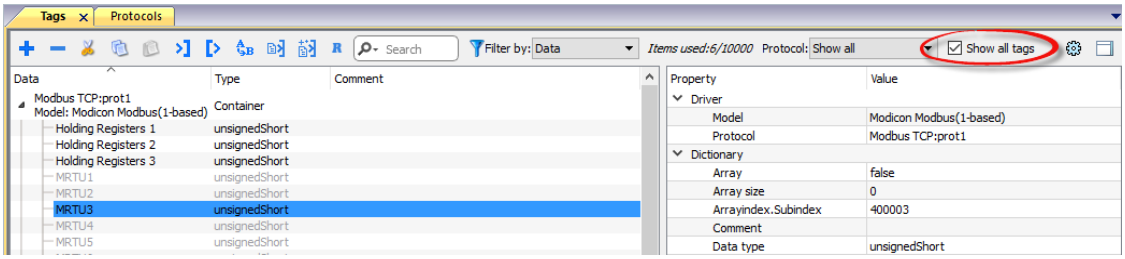
The following dialog shows which importer type can be selected.

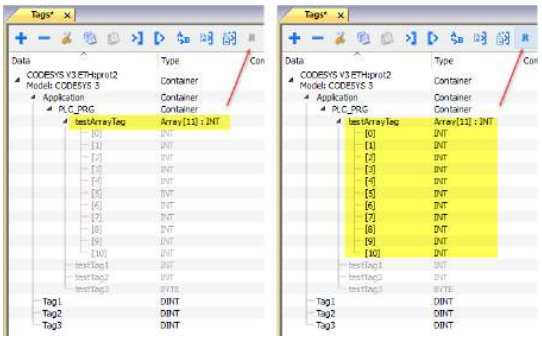
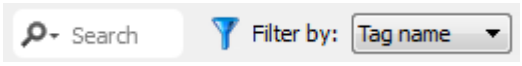


Importer	Description
TIA Portal Project v12 or newer **Linear**	Allows to import the whole TIA Portal project file using **.apxx** file (where "xx" is the TIA Portal version, example: for TIA Portal 13 , file name is "project.ap13"). All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported by this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid	The device did received a response with invalid	Ensure the data programmed in the project are

Error	Cause	Action
response	format or contents from the controller .	consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

Simatic S7 ETH

Simatic S7 ETH communication driver has been designed to communicate with Simatic controllers through Ethernet connection.

The Simatic controller must either have an on-board Ethernet port or be equipped with an appropriate Ethernet interface (either built-in or with a module).

Communication is based on the PG/OP (ISO on TCP) communication functions.

This documents describes the driver settings to be applied in programming IDE software and in S7 PLC programming software.

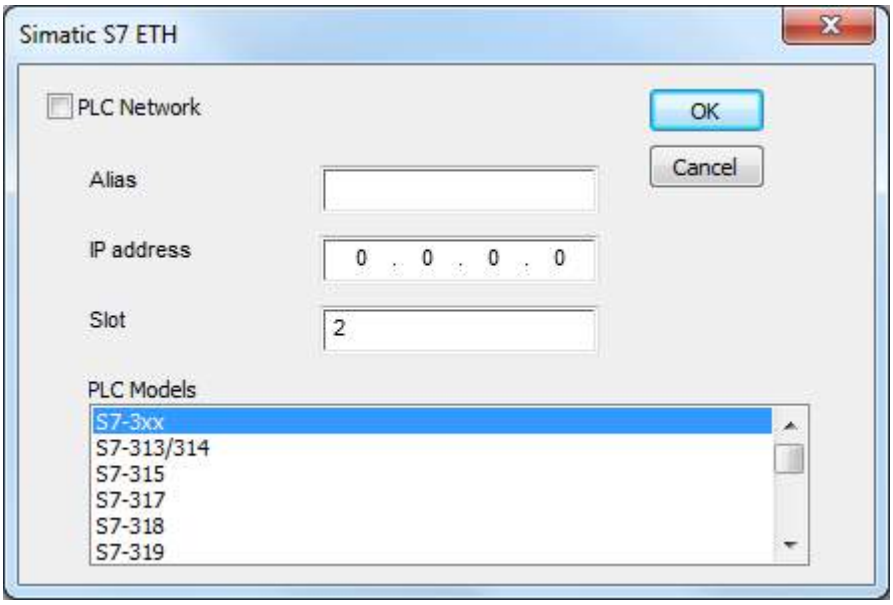
Protocol Editor Settings

Adding a protocol

To configure the protocol:

1. In **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The protocol configuration dialog is displayed.



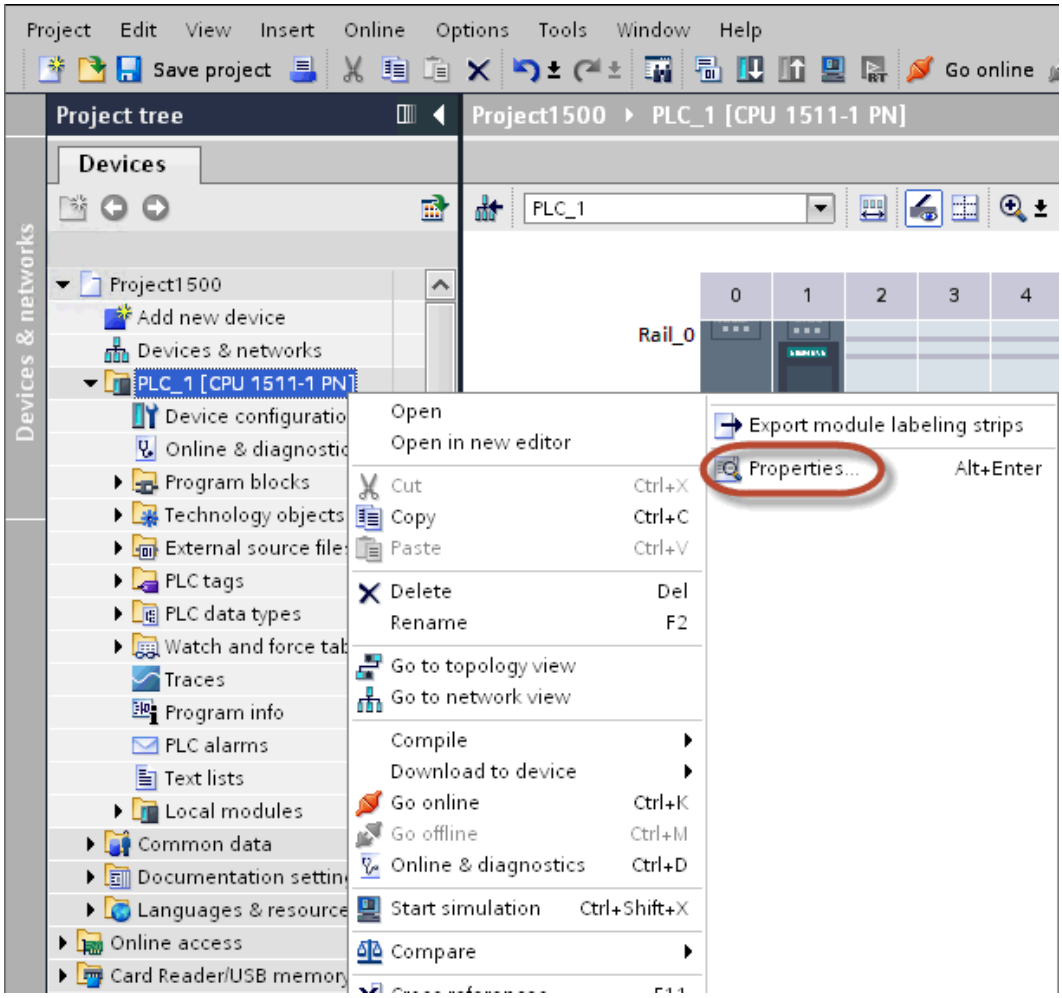
Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Slot	Number of the slot where the CPU is mounted. 2 for S7-300, may take a higher value for S7-

Element	Description
	400 systems.
PLC Models	List of compatible controller models. Make sure to select the correct PLC model in this list when configuring the protocol.
PLC Network	Enable access to multiple networked controllers. For every controller (slave) set the proper option.

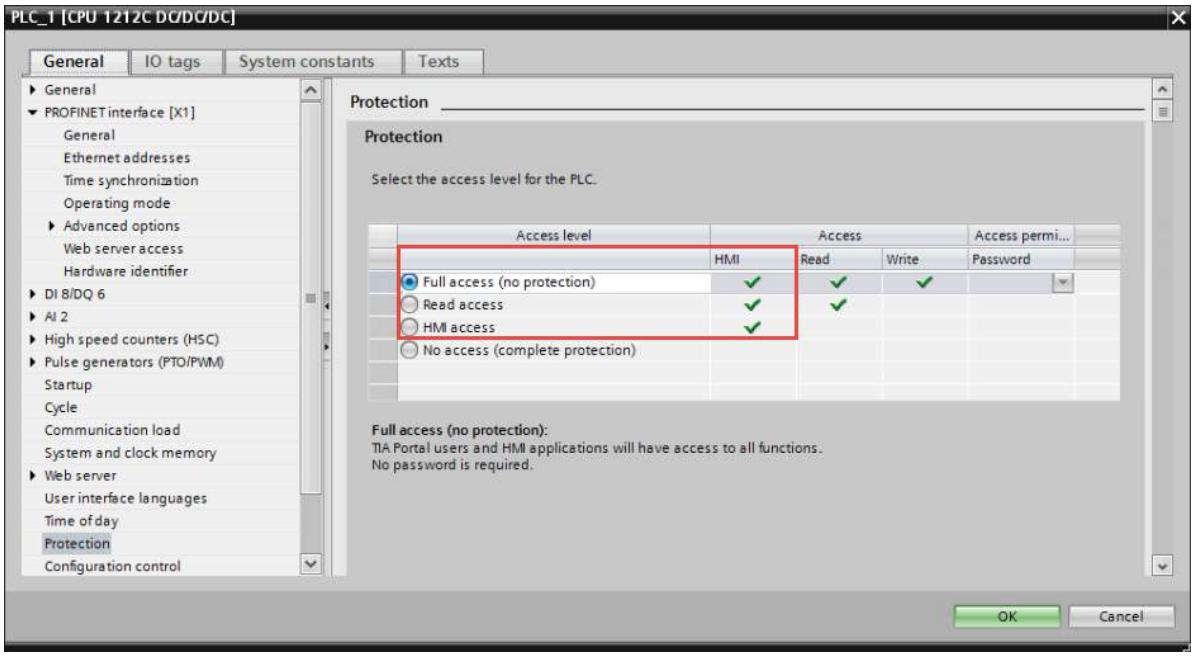
S7-1200 and S7-1500 PLC configuration

S7-1200 (starting from firmware version 4.0) and S7-1500 PLC Series from Siemens have a built-in firewall; by default the maximum protection level is enabled. To establish communication with these PLC models it is necessary to enable S7 communication with 3rd party devices; this setting is available in TIA Portal programming software.

1. Open the PLC project in TIA Portal.
2. Select the PLC from the project tree and open PLC Properties.

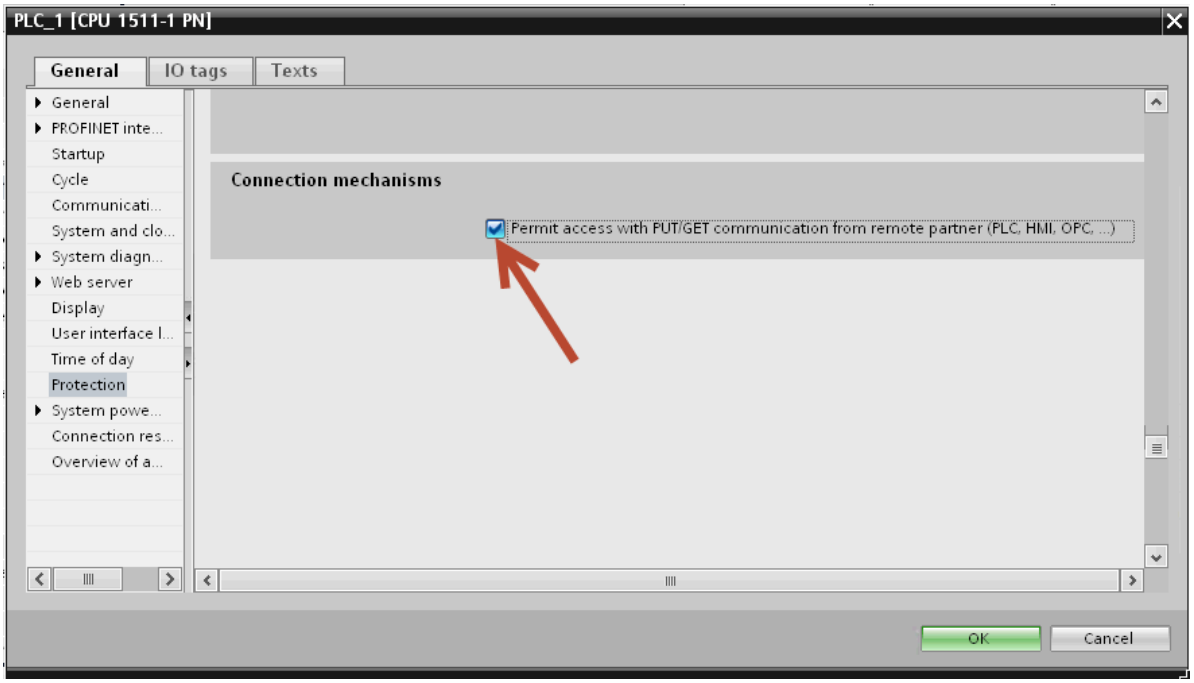


3. In General > Protection choose a permission between the top three (make sure that the tick is present on HMI column).



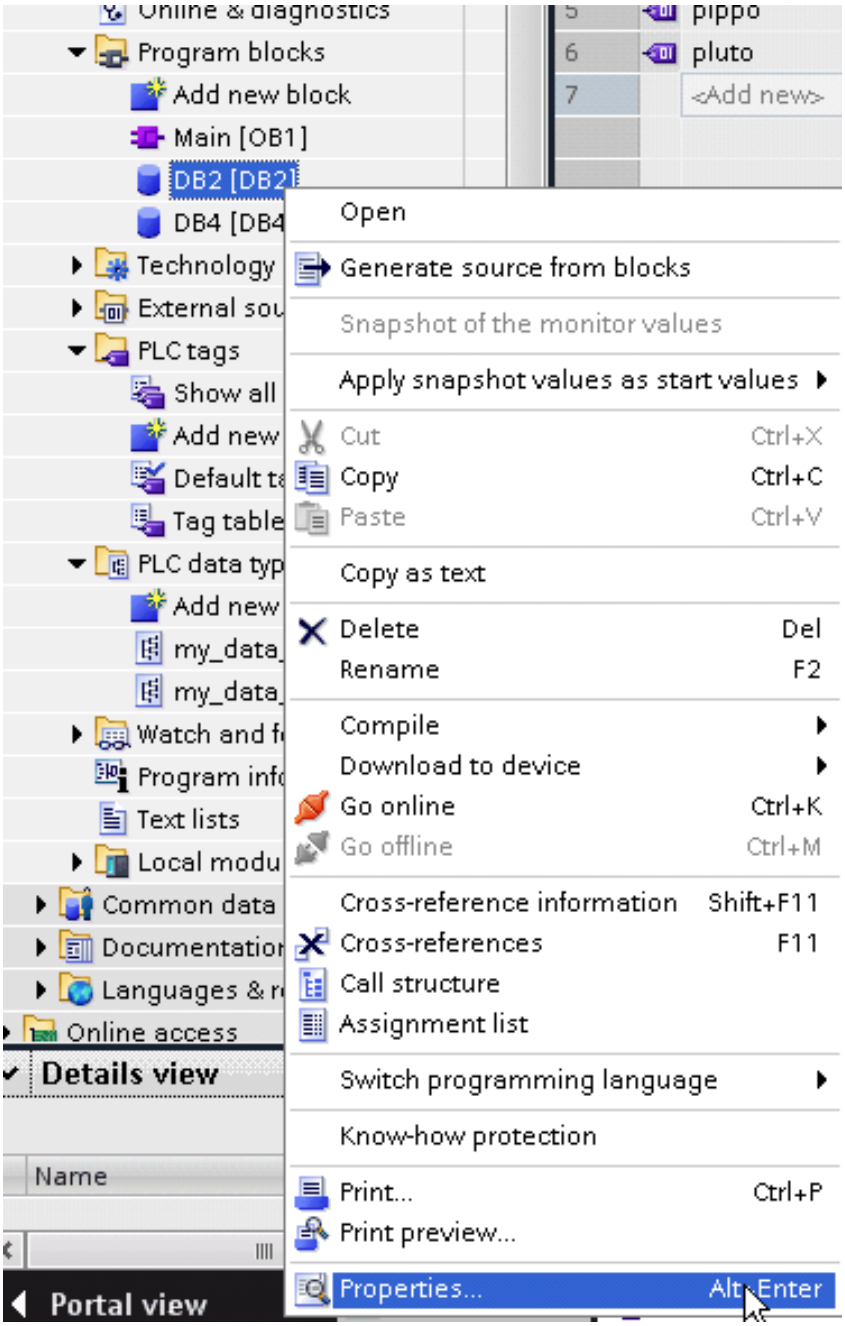
Note: If "No access" is selected, the communication with the panel will not be established.

4. Scroll down the page and check "Permit access with PUT/GET communication from remote partner".

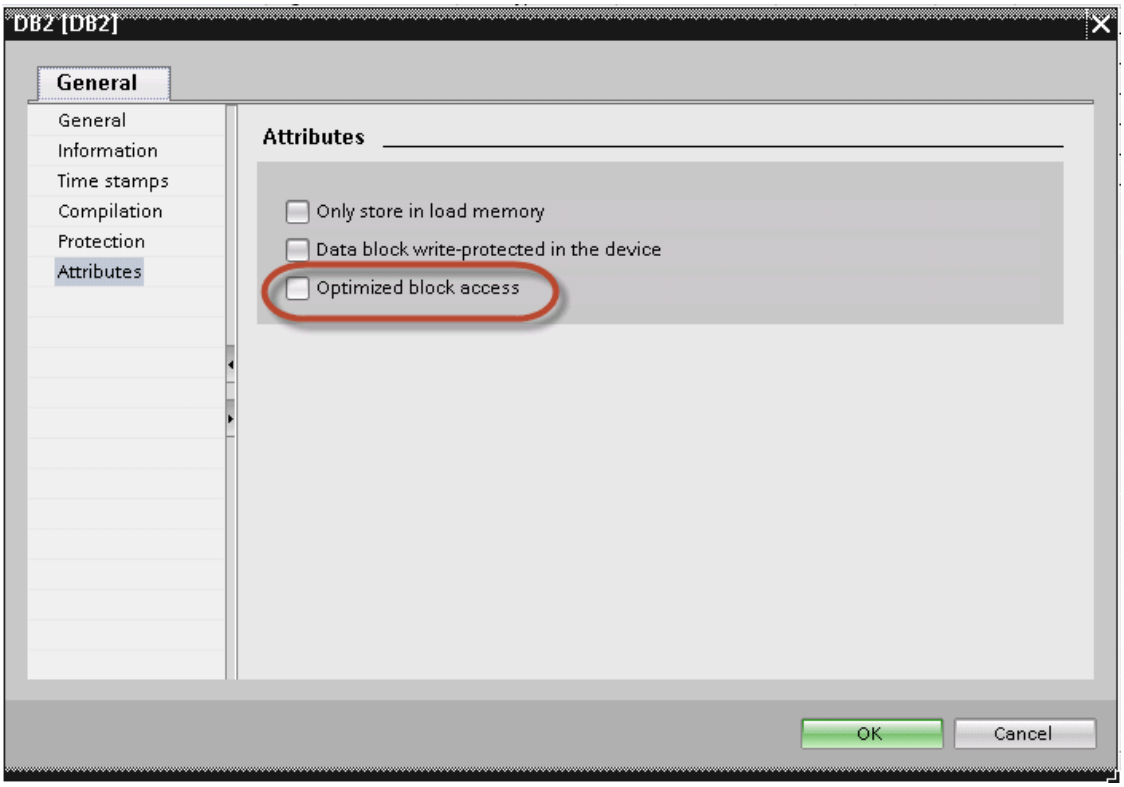


Note: If variables are defined in "Program blocks", DB must configured as "Not optimized".

To check or change DB optimization, open DB Properties:



In General > Attributes uncheck "Optimized block access":



If check box "Optimized block access" is not available (grayed-out) it could be because DB is an "instance DB" linked to an "optimized access FB".

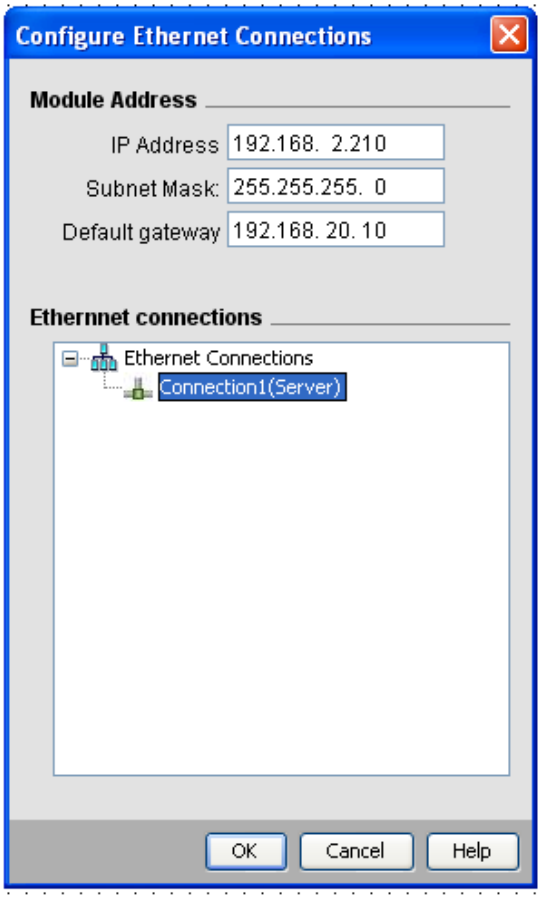
After compiling the project, tag offsets will be shown close to variable name.

These settings can be applied to TIA Portal programming software, S7-1200 PLC family starting from PLC firmware version 4.0 and S7-1500 PLC family.

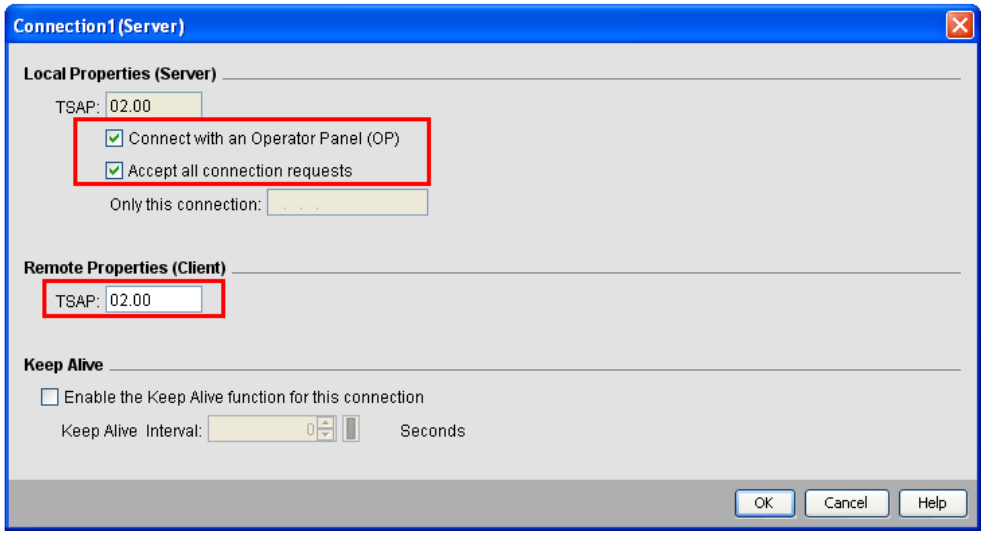
Logo! PLC configuration

To configure communication with Logo! PLC:

1. Open the Logo!Soft Comfort project.
2. Select **Tools** > **Ethernet Connections**: the Configure Ethernet Connections dialog is displayed.



3. Right-click on **Ethernet Connections** and add a server connection.
4. Double-click on the newly created connection: the connection properties dialog is displayed.



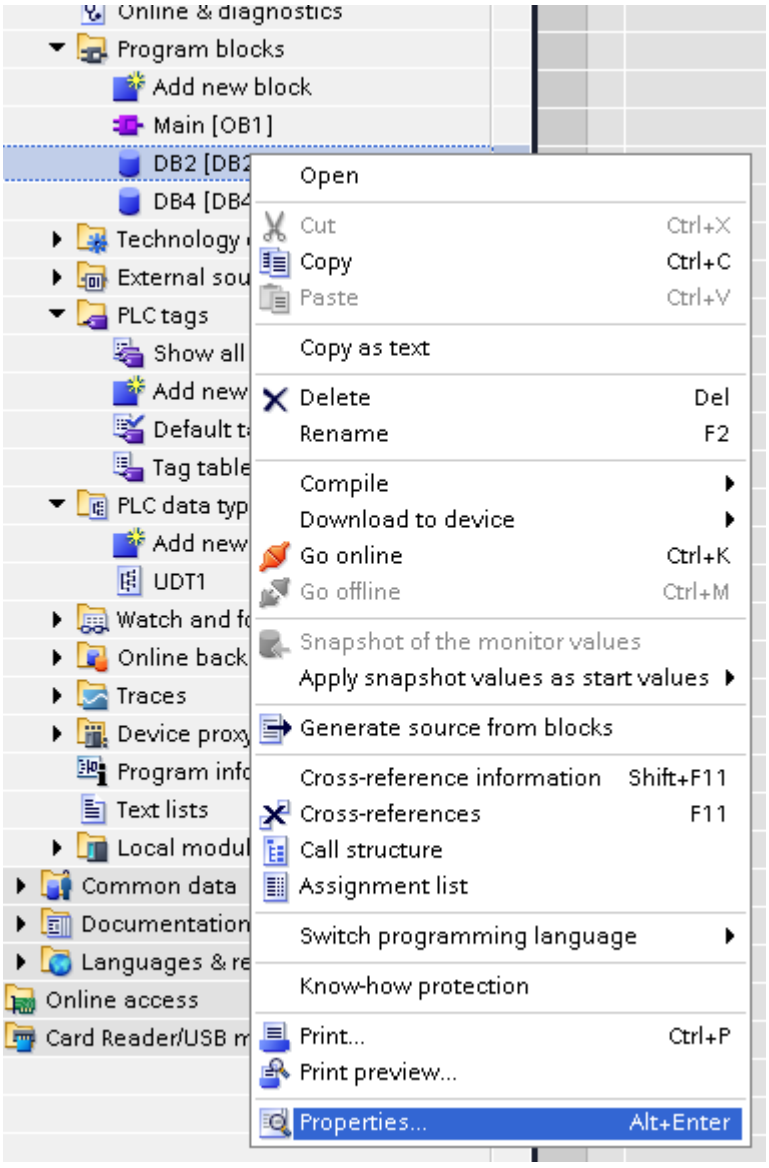
5. Select the **Connect with an operator panel (OP)** (0BA7 model only, do noth check for Logo! 0BA8 model)
6. Select Accept all connection requests options.
7. In the **Remote Properties (Client)** section, set **TSAP** to 02.00.

Direct Import of TIA Portal project

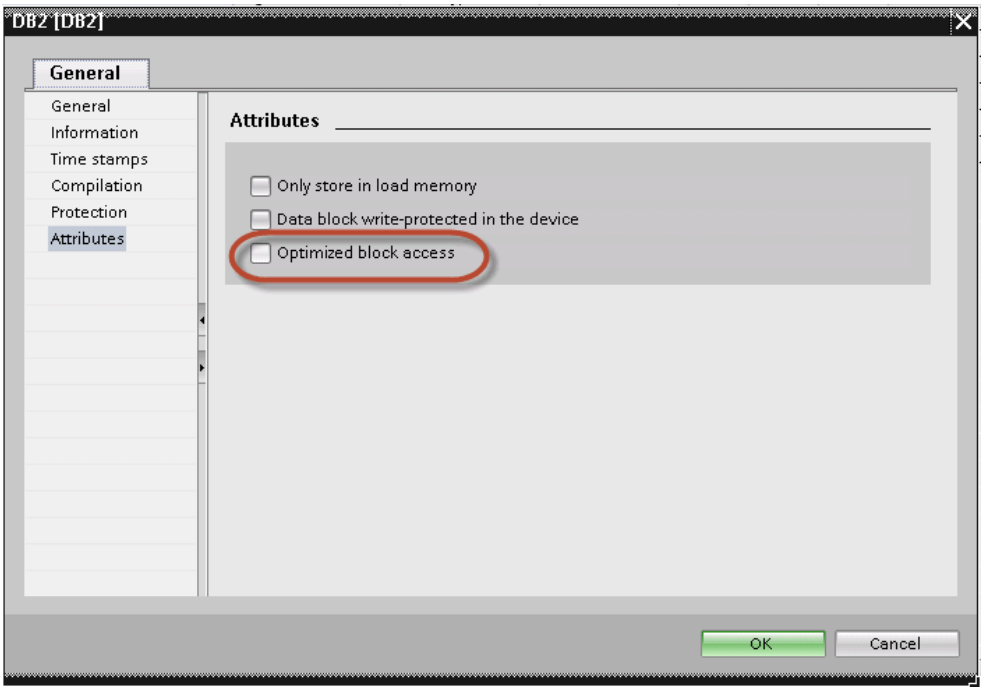
It is possible to import TIA Portal variables directly from TIA Portal project, by selecting "TIA Portal Project v12 or newer" from import selection (refer to "Tag Import" chapter).

Data Blocks must be set as Not optimized:

1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:



3. In the **General** tab select **Attributes** and unselect **Optimized block access**.



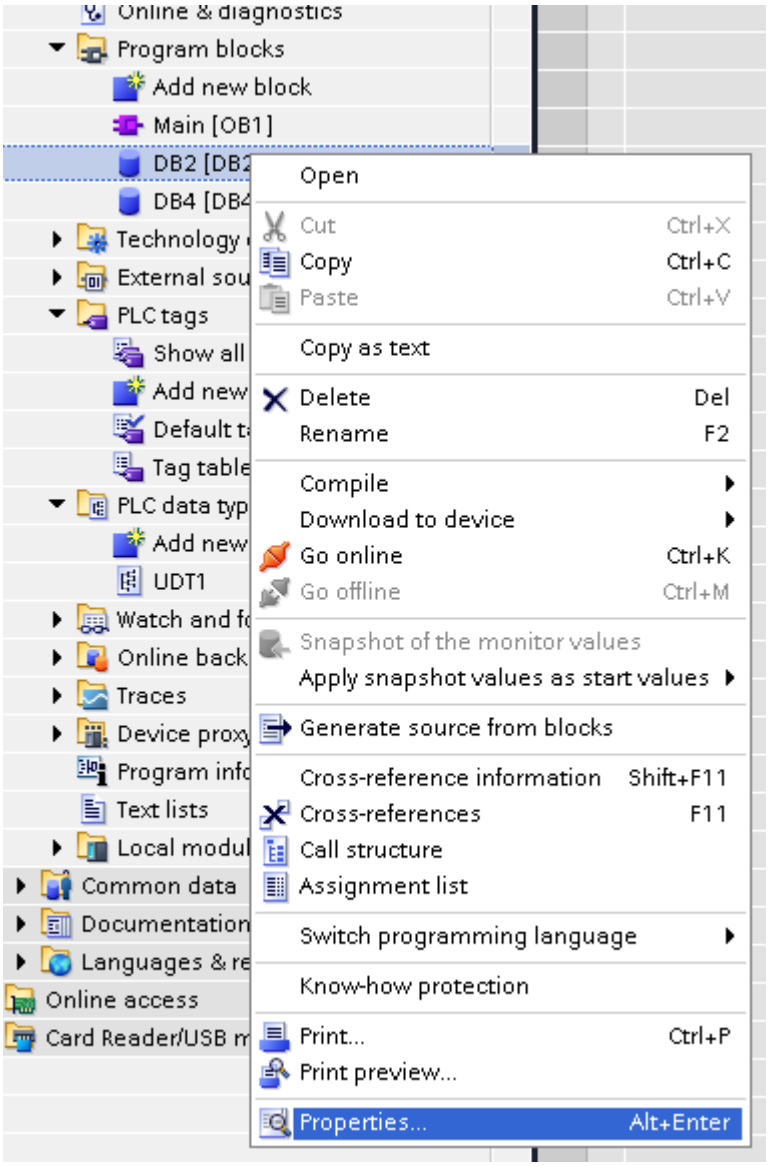
Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

Export using TIA Portal v13, v14 or newer

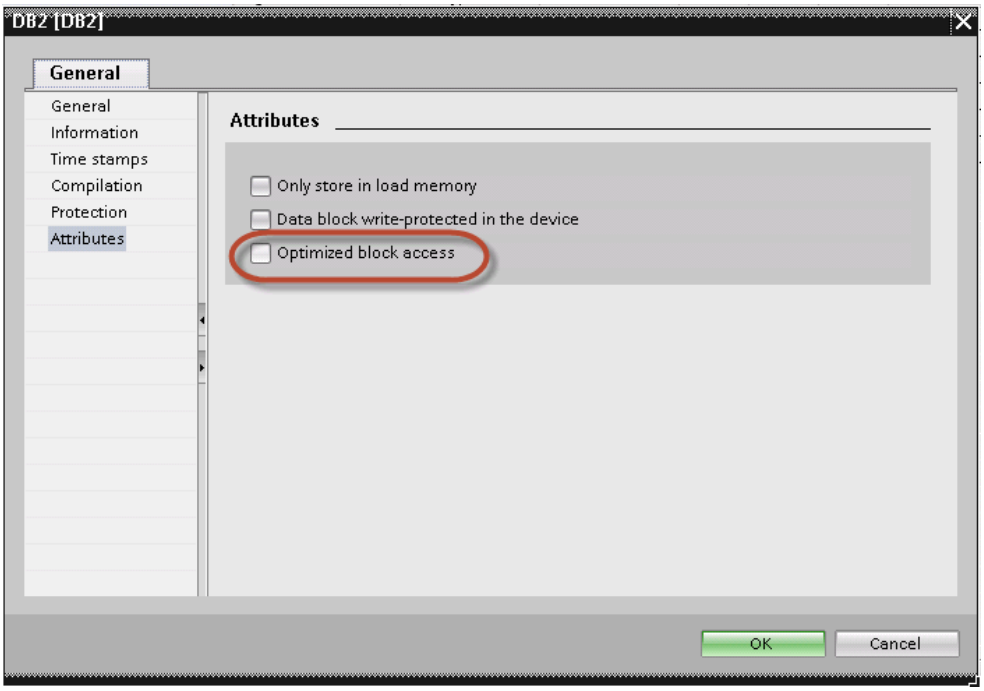
Exporting Program blocks

These files refer to DB tags defined in **Program blocks**.

1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:

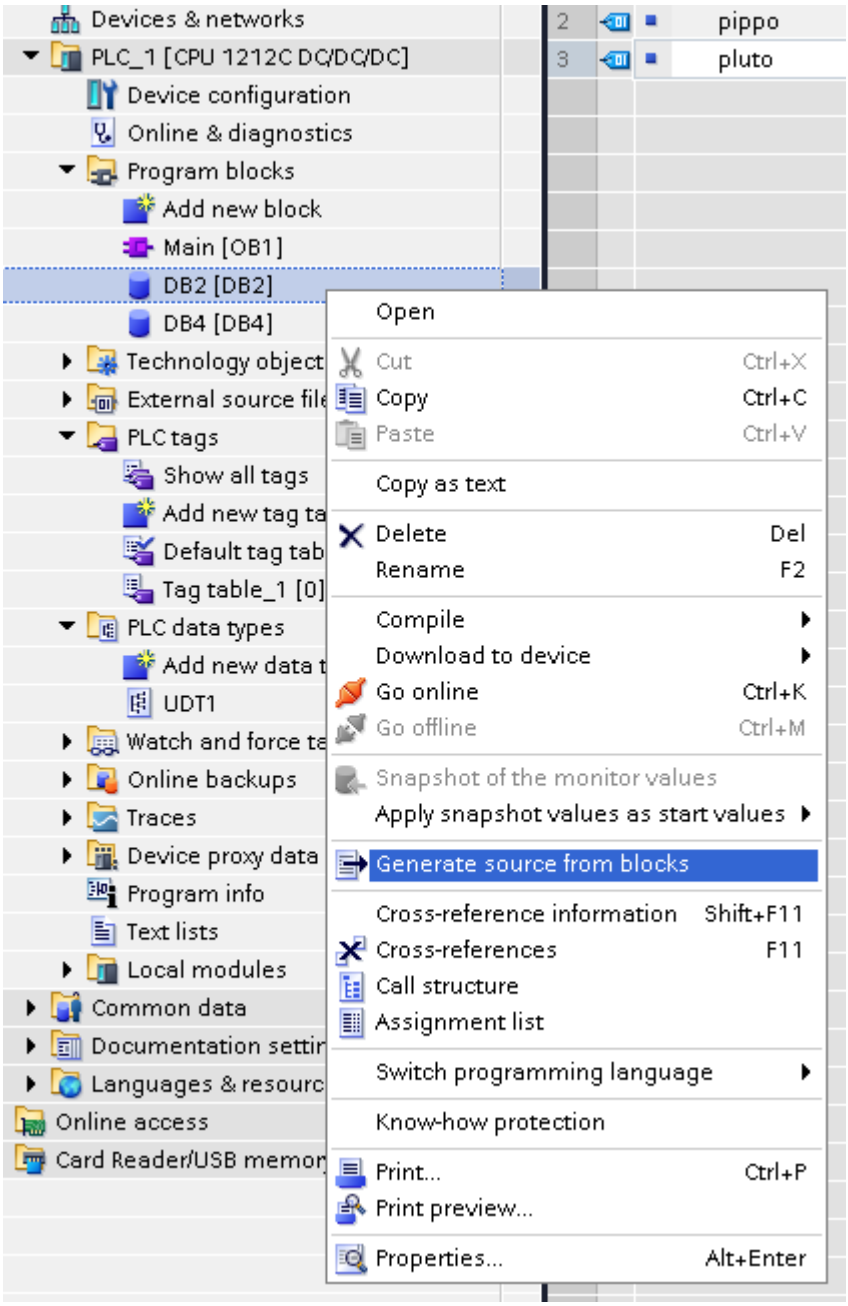


3. In the **General** tab select **Attributes** and unselect **Optimized block access**.

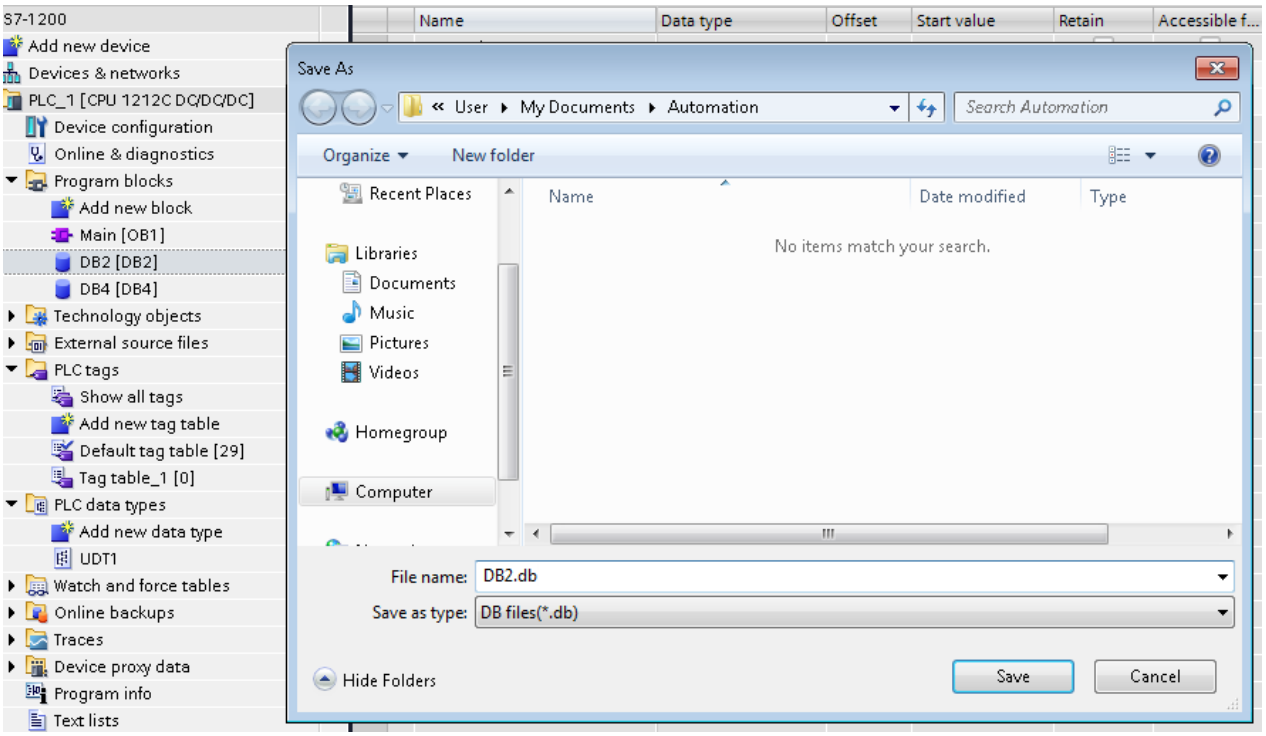


Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

4. Right-click on the Data Block and choose **Generate source from blocks**:



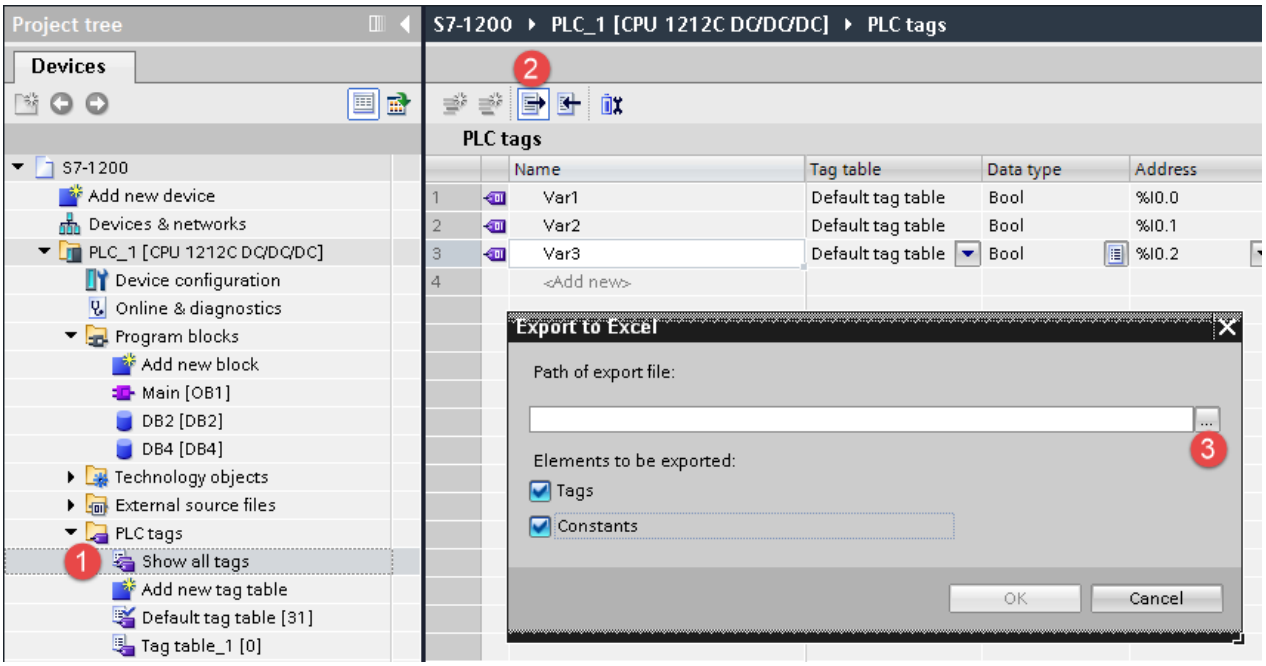
5. Save the file as DBxxx.db, where xxx=number of DB.



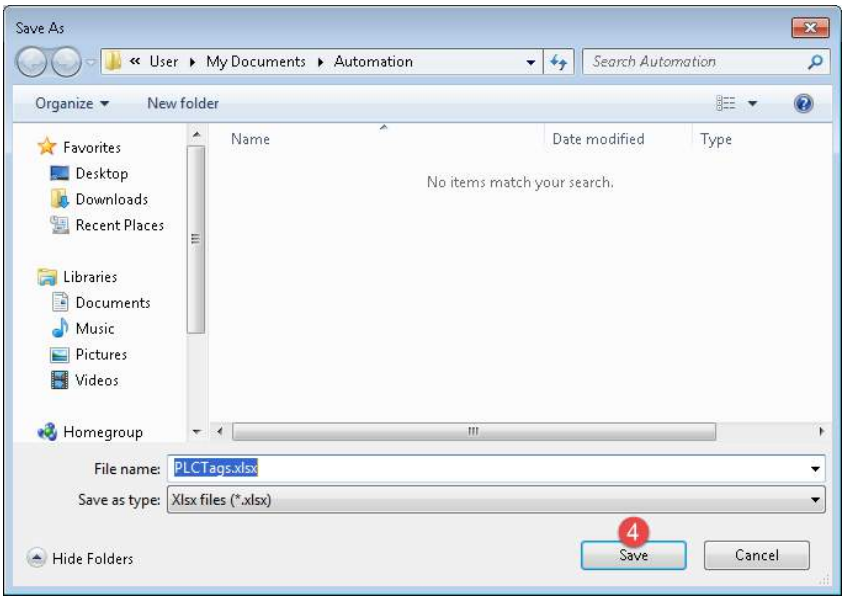
Exporting PLC tags

An Excel file refers to PLC tags.

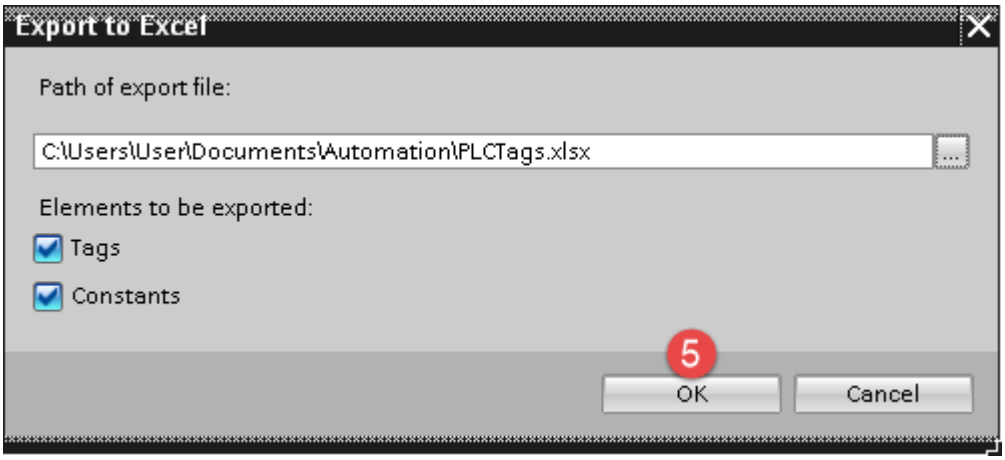
1. Double-click **Show all tags**: the tag table is displayed.
2. Click the **Export** button and browse for path file.
3. Define file name.



4. Click **Save** to confirm.

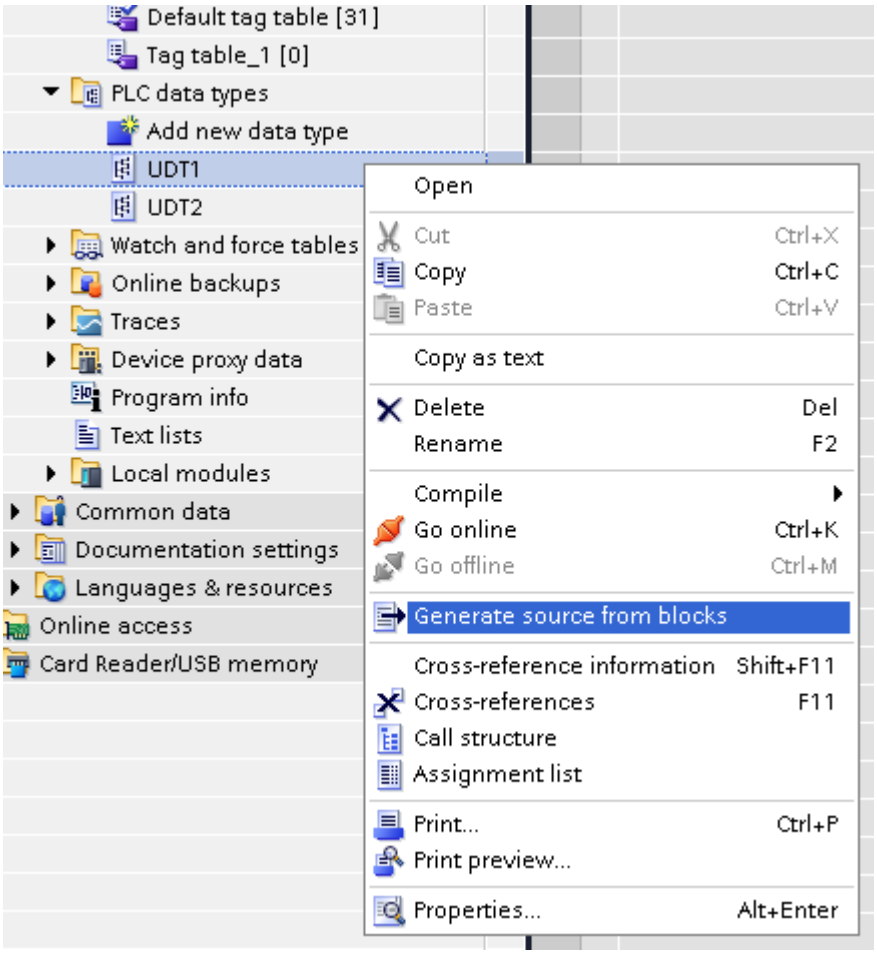


5. Click **OK** to export.

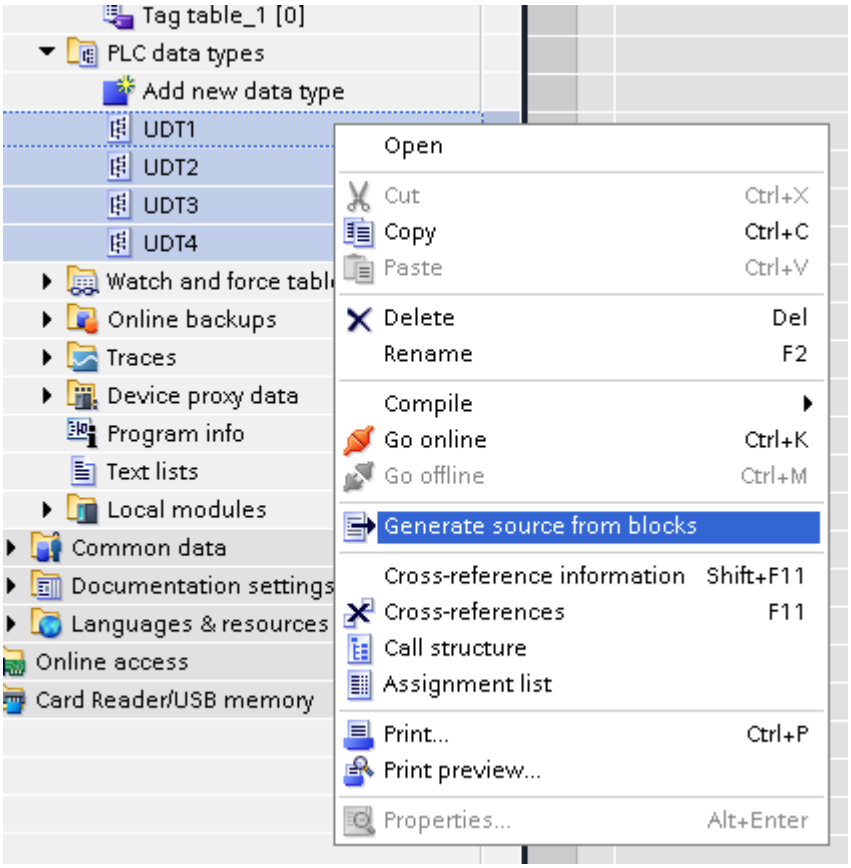


Exporting PLC data types

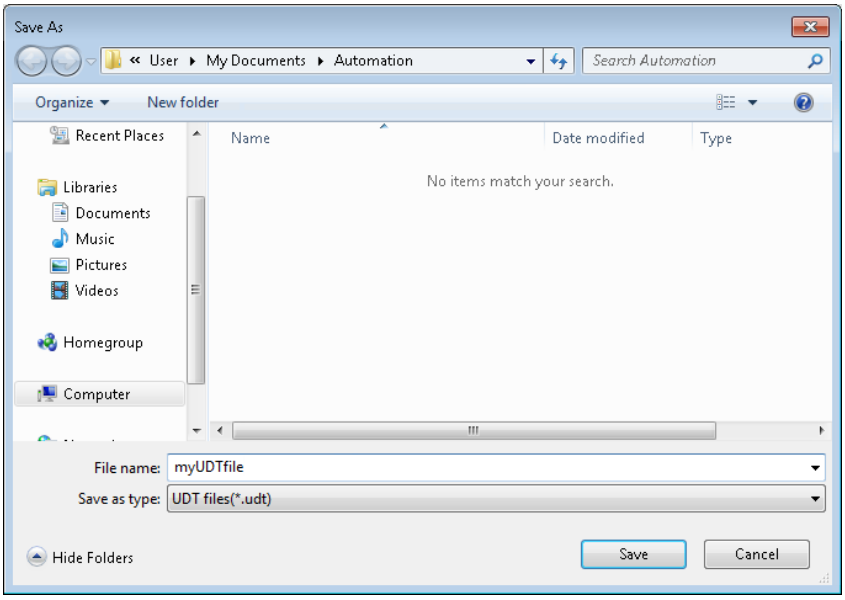
To create the file, expand **PLC data types** item from TIA Portal project tree and right click on the user defined structure. Then click on **Generate source from blocks**.



In case of multiple PLC data types in PLC project, it is necessary to select them all from **PLC data types** list, right click and select **Generate source from blocks** to create the .UDT file that contains all the PLC data types defined.



In the next step, give a name to the .UDT file and choose the path to where to save the file.



This file will content all the PLC data types and it can be used for importing tags in Tag Editor.

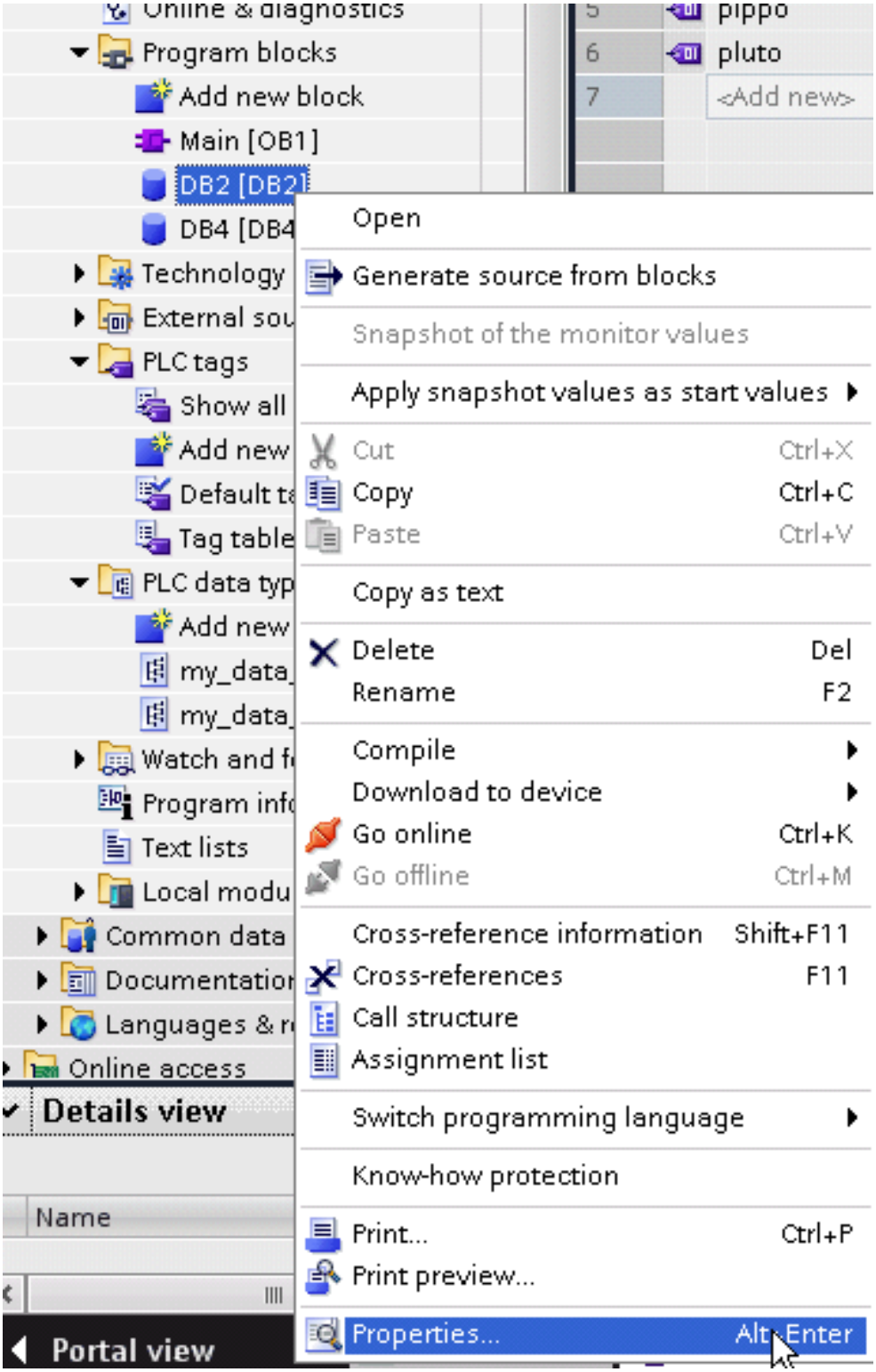
Check **Tag Import** chapter for more details.

Export using TIA Portal v10, v11, v12

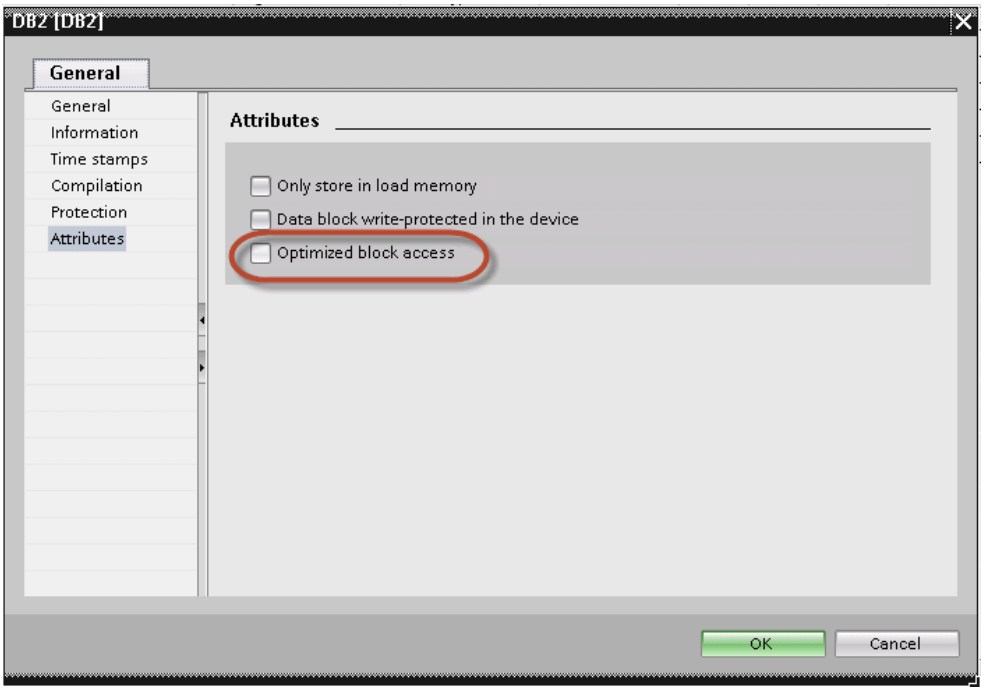
Exporting Program blocks

These files refer to DB tags defined in **Program blocks**.

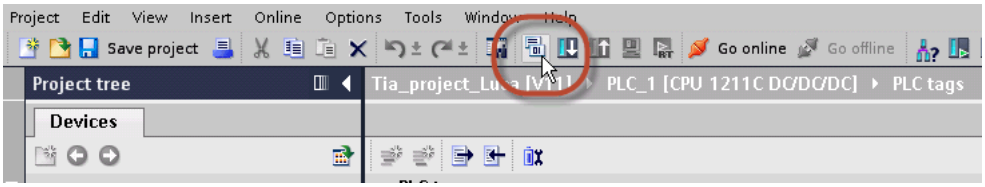
1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:



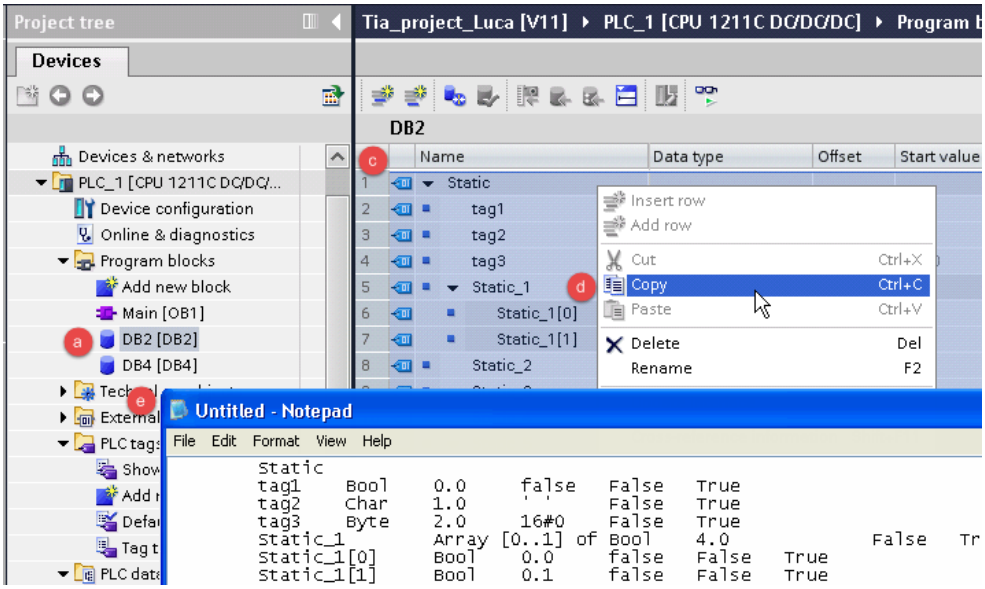
3. In the **General** tab select **Attributes** and unselect **Optimized block access**.



Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".



4. Build the project to make sure TIA Portal calculates the tags offset.

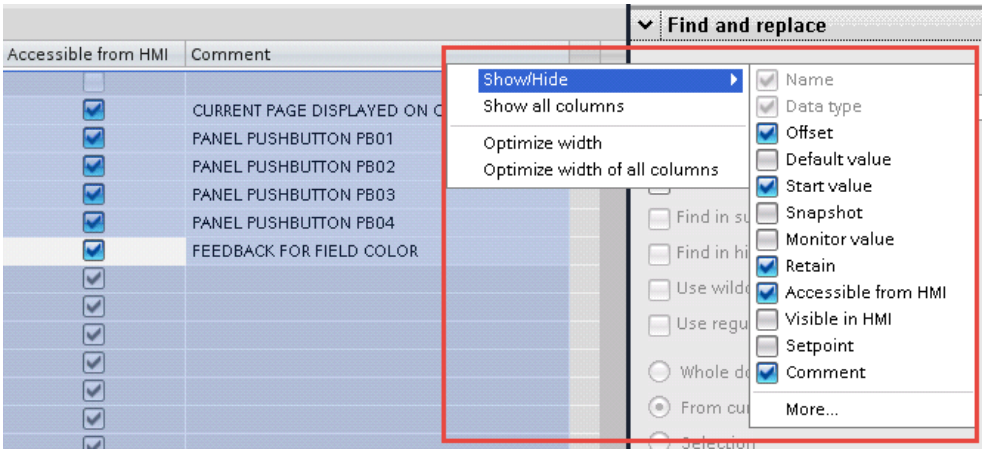


5. Double-click on a DB name.
6. Expand the view of program block selected.
7. Select all rows.
8. Copy and paste into any text editor.
9. Save the file as DBxxx.tia, where xxx=number of DB.

Note: Make sure you use the **Save As** function or the file will be named DB2.tia.txt and will not be visible from the importer.

10. Repeat from step 5 for all program blocks.

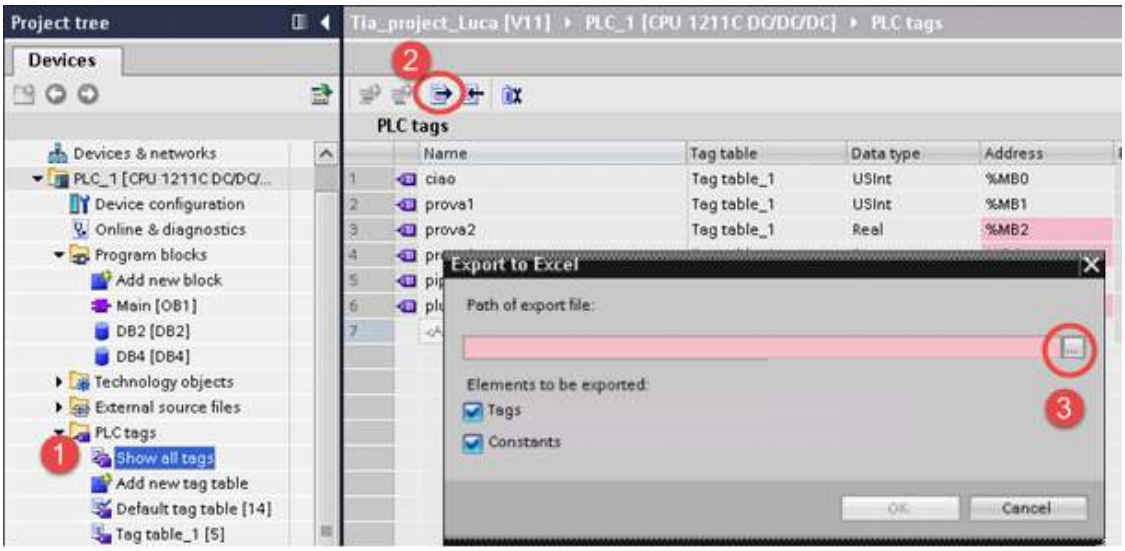
Note: Make sure that only the following columns are shown in DB editor before copying all data in the txt file



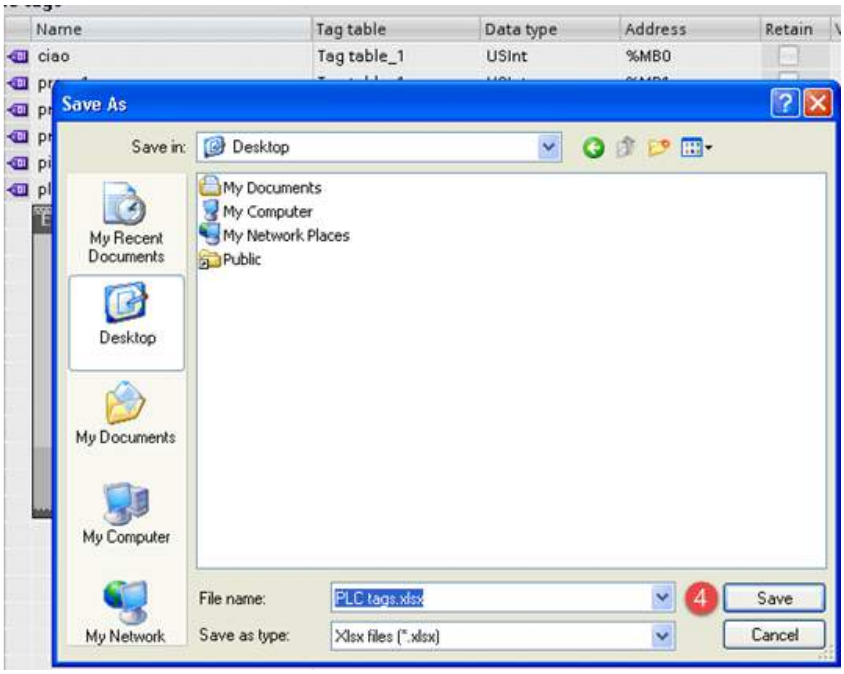
Exporting PLC tags

An Excel file refers to PLC tags.

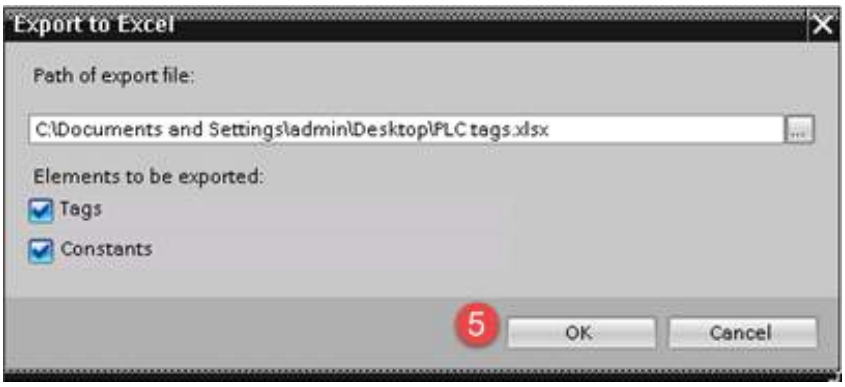
1. Double-click **Show all tags**: the tag table is displayed.



2. Click the **Export** button and browse for path file.
3. Define file name.
4. Click **Save** to confirm.

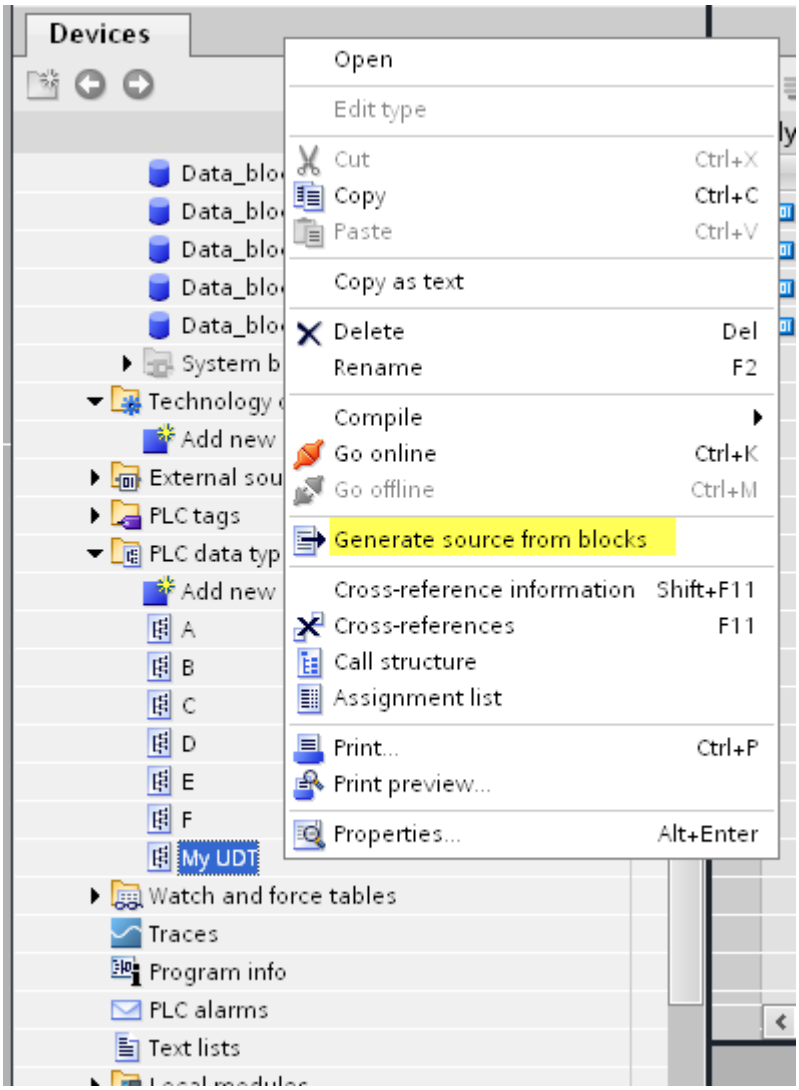


5. Click **OK** to export.

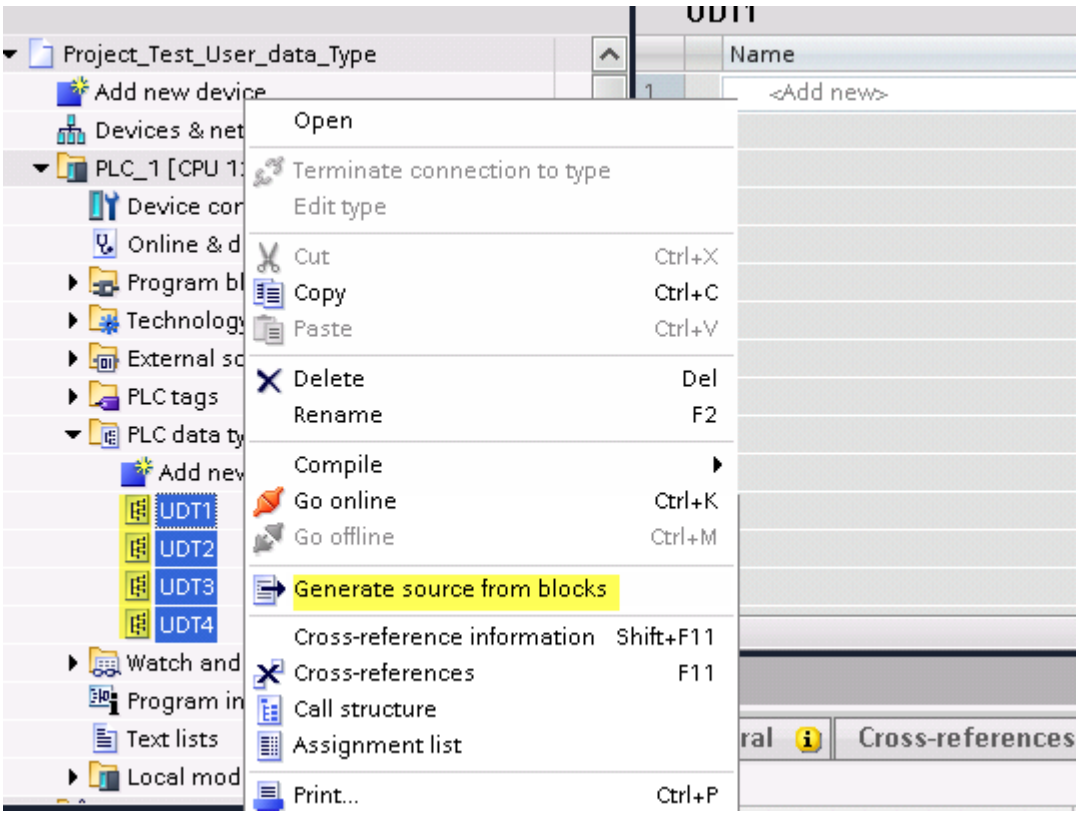


Exporting PLC data types

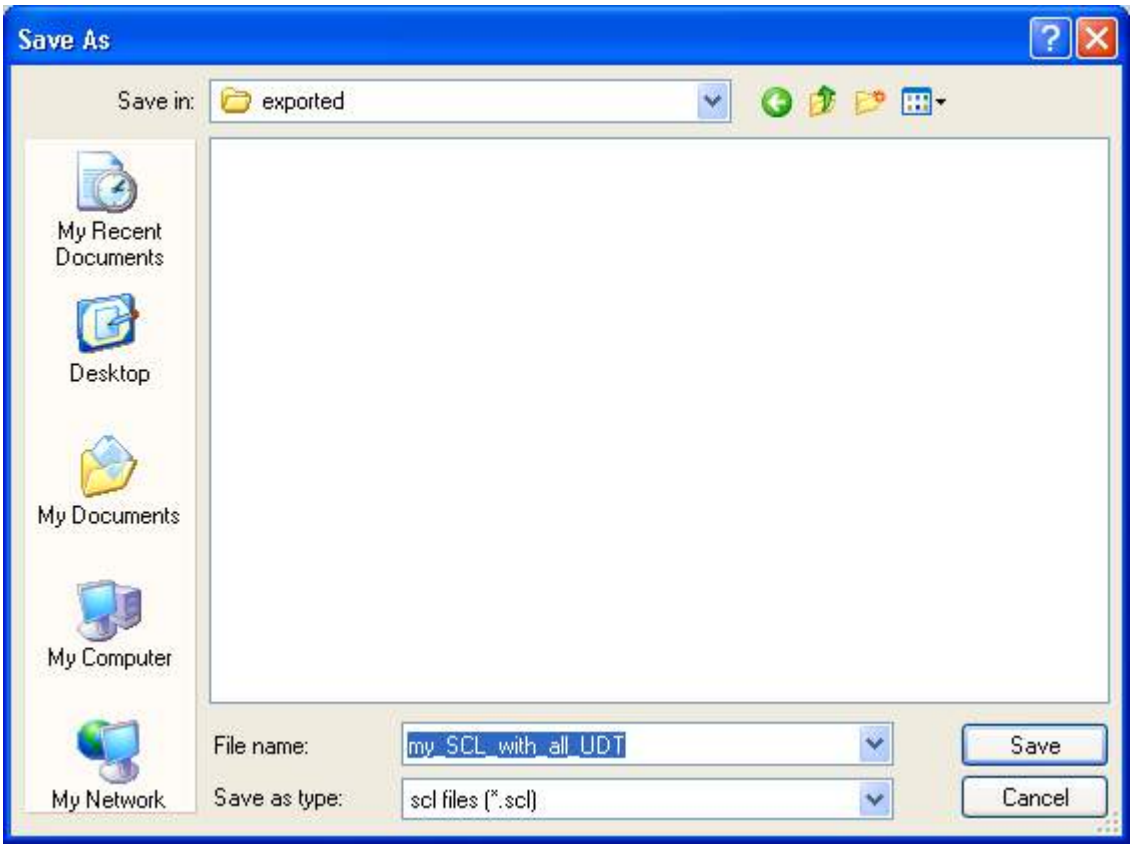
To create the file, expand **PLC data types** item from TIA Portal project tree and right click on the user defined structure. Then click on **Generate source from blocks**.



In case of multiple PLC data types in PLC project, it is necessary to select them all from **PLC data types** list, right click and select **Generate source from blocks** to create the .SCL file that contains all the PLC data types defined.



In the next step, give a name to the .SCL file and choose the path to where to save the file.



This file will content all the PLC data types and it can be used for importing tags in Tag Editor.

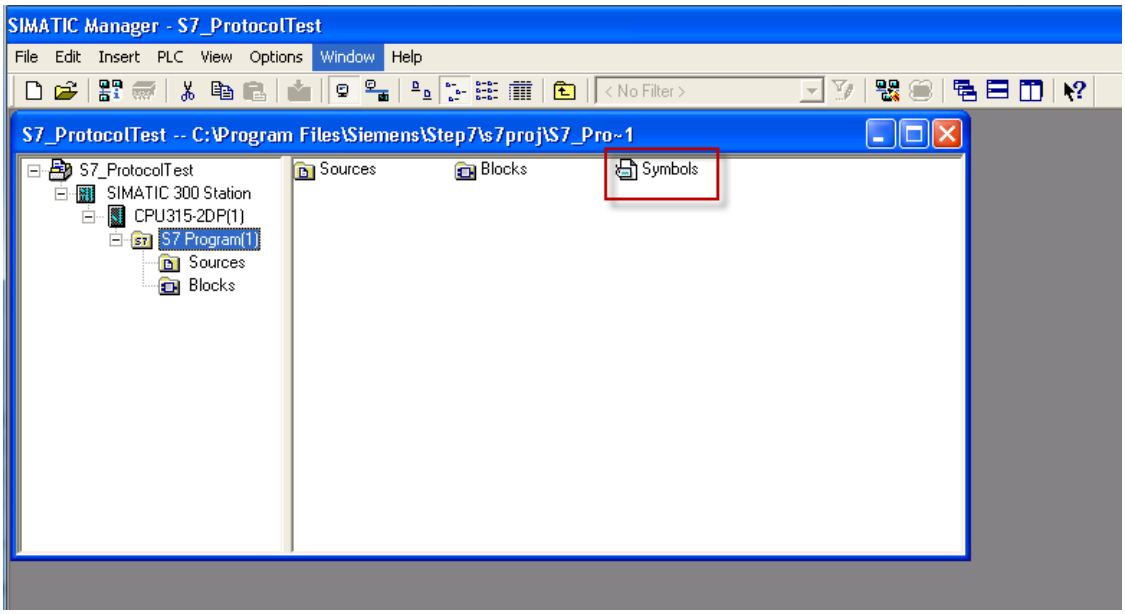
Check **Tag Import** chapter for more details.

Export using STEP7

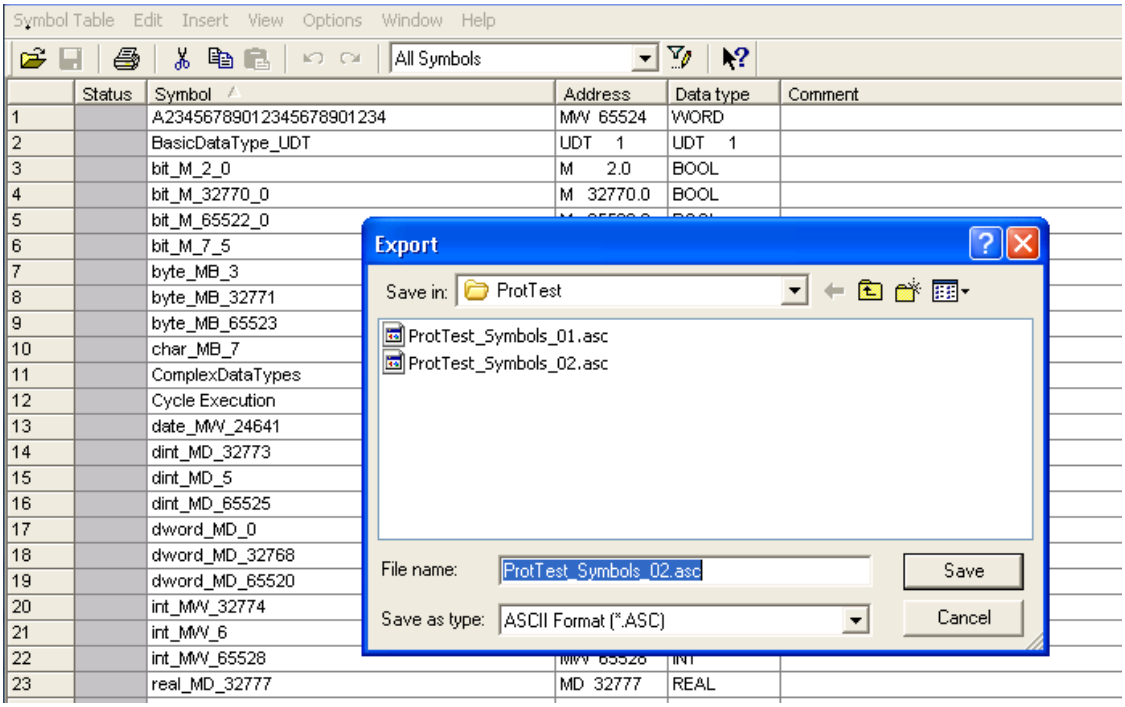
The Simatic S7 ETH Tag importer accepts symbol files (ASCII format .asc) and source files (.awl extension) created by the Simatic Step7. The symbol file can be previously exported using the Step7 symbol table utility.

Exporting Symbols table

Symbol files (.asc) can be exported from the symbol table utility.



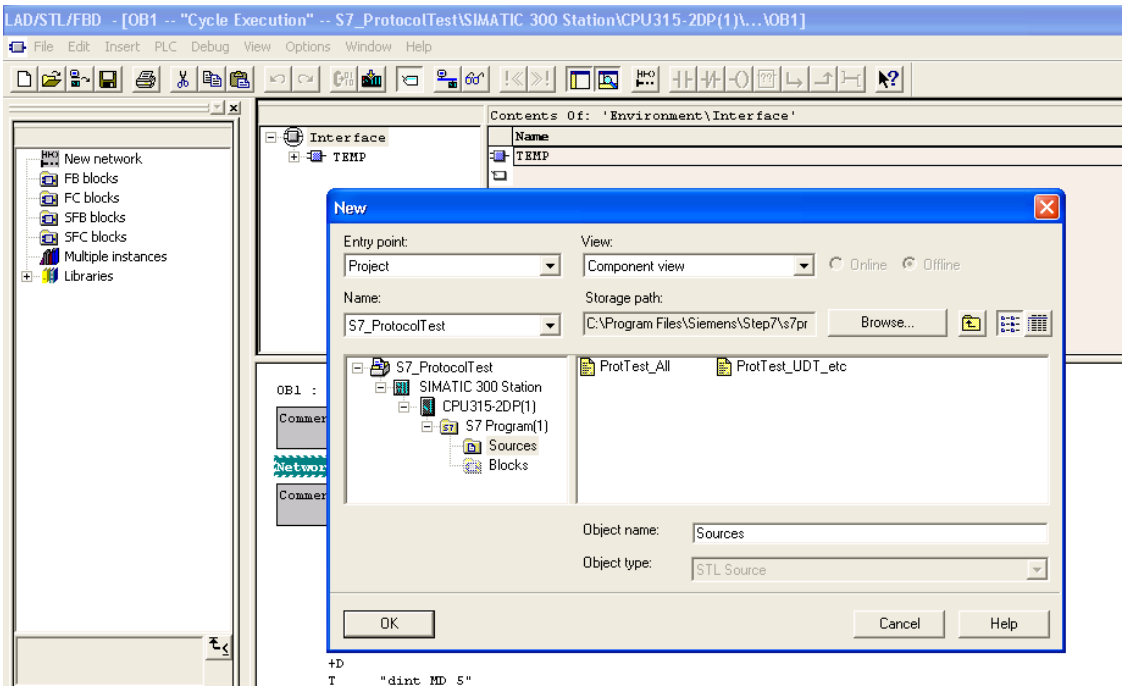
1. From the **Symbol Table** menu in the Symbol Editor choose **Export**.
2. Assign a name and save the symbol table as ASCII file.



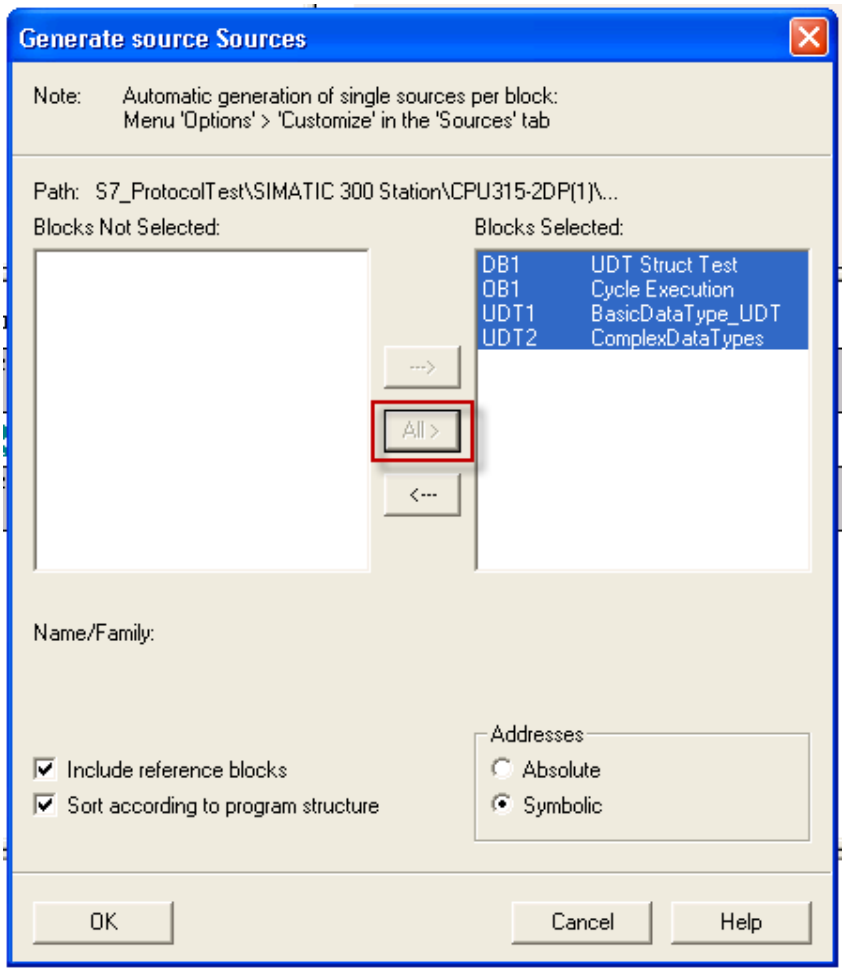
Exporting Sources

These files are created exporting source code.

1. Open any program block in the editor, "OB1" in this example.
2. From the **File** menu choose **Generate Source**: the following dialog is displayed:



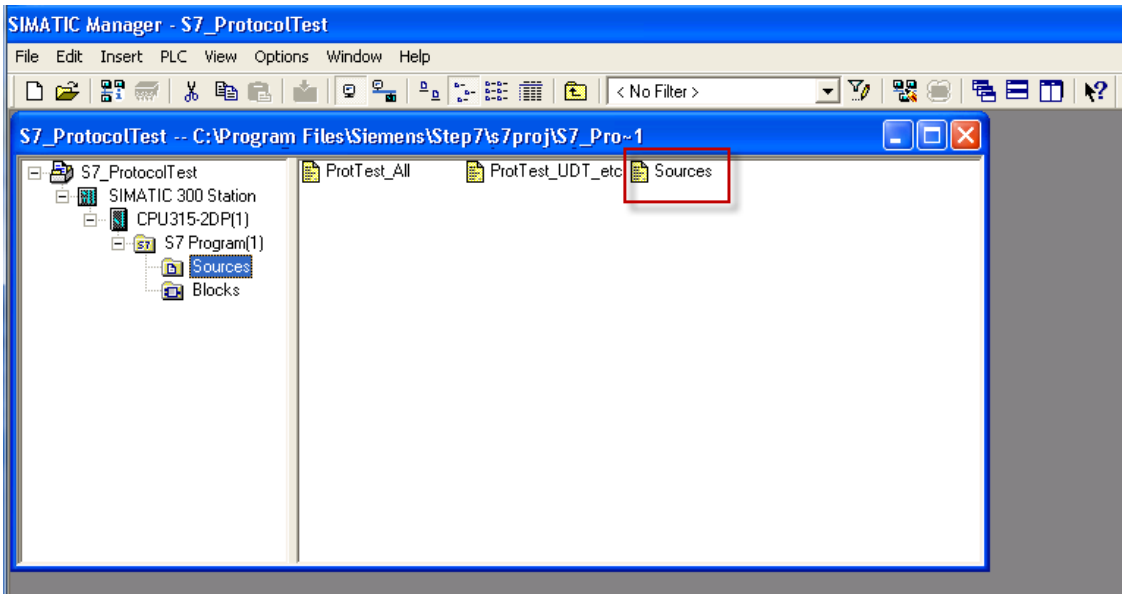
1. Assign a name, "Sources" in the example, and click **OK**: the **Generate source Sources** dialog is displayed.



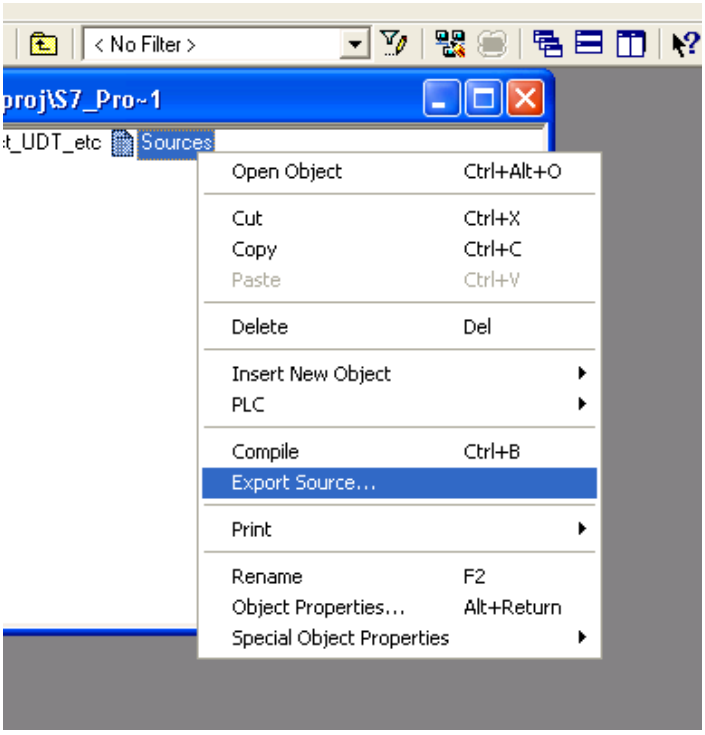
2. Click **All >** to generate source for all blocks.
3. Select the following options:

- **Include reference blocks**
- **Sort according to program structure**
- **Symbolic** address

4. Click **OK** to confirm: the "Sources" object is generated in the Step7 project as in the example.



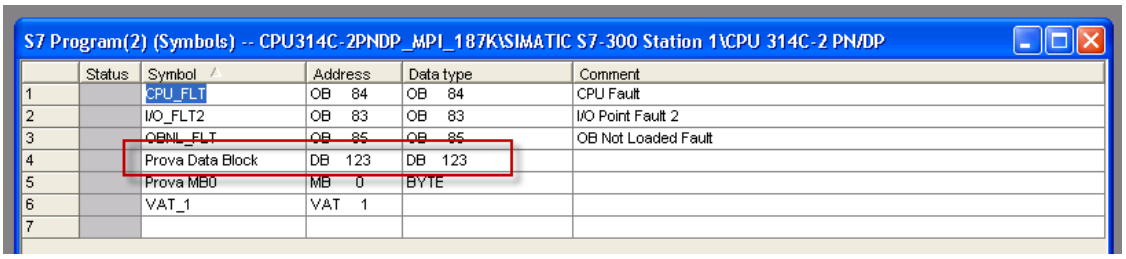
5. Right click on the object and select **Export Sources**.



The generated .awl file can be imported in the Tag Editor.

Note: The .awl file contains additional information not included in the .asc file exported from the symbol table.

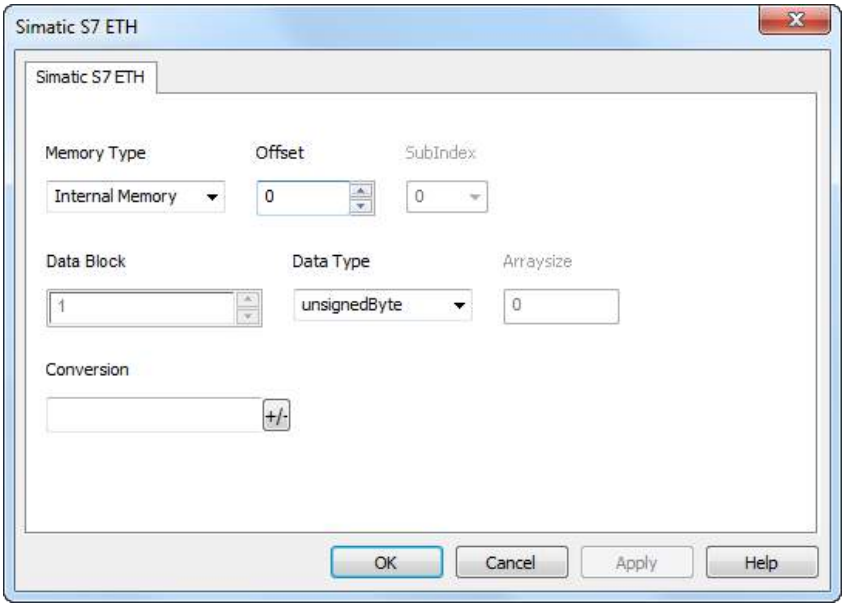
Make sure that reference to all data blocks is inserted in the symbol table. The tags from a data block are imported only if the symbol table contains a line with the data block name and related comment.



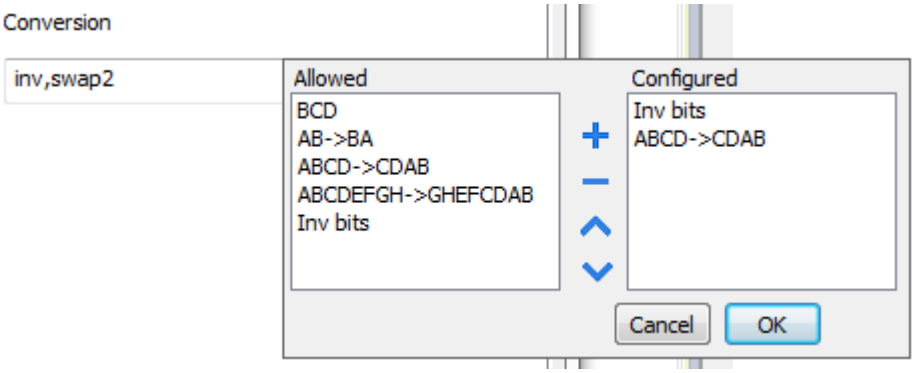
Each entry enables the import filter to import the tags related to the specified data block.

Tag Editor Settings

In the Tag Editor select “Simatic S7 ETH” from the list of defined protocols and click **+** to add a tag.



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Area of PLC where tag is located.<table><tr><th>Data Type</th><th>Simatic Type</th></tr><tr><td>Internal Memory</td><td>M</td></tr><tr><td>Data Block</td><td>DB</td></tr><tr><td>Input</td><td>I (E)</td></tr><tr><td>Output</td><td>O (A)</td></tr><tr><td>Timer value</td><td>T</td></tr><tr><td>Counter value</td><td>C</td></tr></table></td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.</td></tr>
<tr><td>SubIndex</td><td>Resource offset within the register.</td></tr>
<tr><td>Data Block</td><td>Data block number for Data Block Memory Type.</td></tr>
<tr><td>Data Type</td><td>Available data types:
boolean
byte
short
int
unsignedByte
unsignedShort
unsignedInt
float
string
See "Programming concepts" section in the main manual.
Note: To define arrays, select one of Data Type format followed by square brackets.</td></tr>
</table>

Element	Description
Arraysize	• In case of array tag, this property represents the number of array elements. • In case of string tag, this property represents the maximum number of bytes available in the string tag. Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the **Allowed** list shows one or more conversions, listed below.

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)

Element	Description

Value	Description
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	Separate the byte in two nibbles, and reads them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)
S5timer(BCD)	Used to support S5timer. Check **Simatic S5timer special data type** for more details.
S5timer(BIN)	Legacy transformation for S5timer in binary format.

Select the conversion and click on plus button. The selected item will be added on **Configured** list.

If more conversions are configured, they will be applied in order (from top to bottom of **Configured** list).

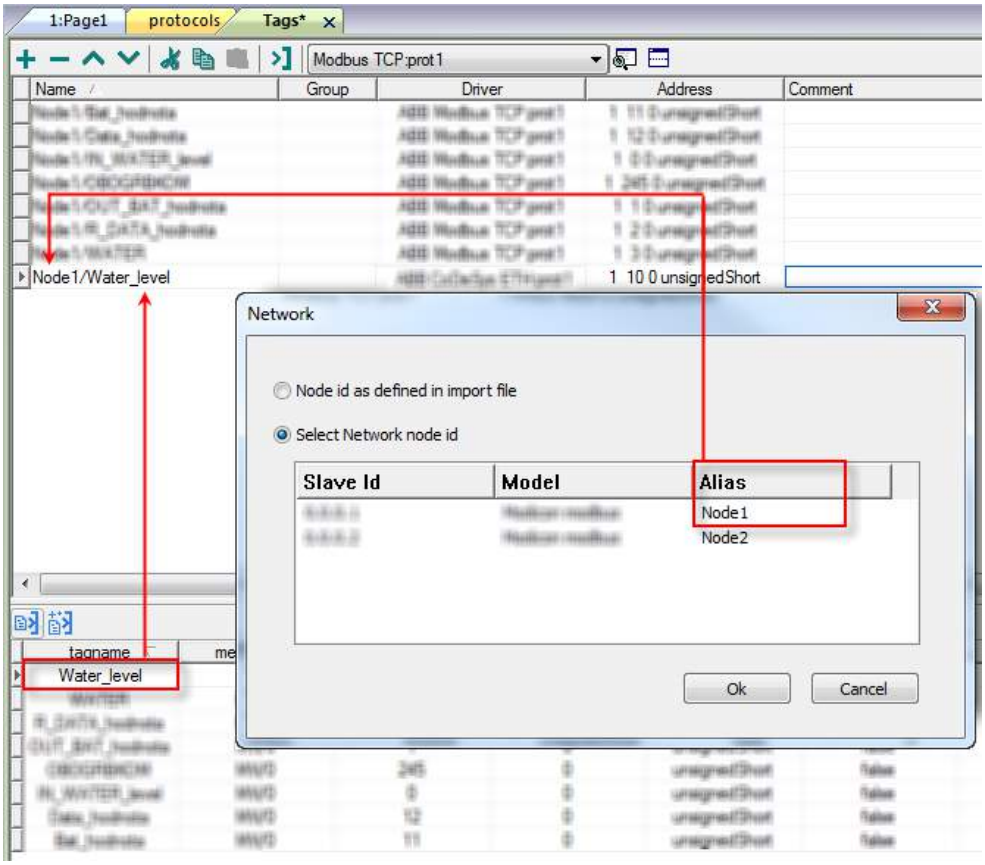
Use the arrow buttons to order the configured conversions.

Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



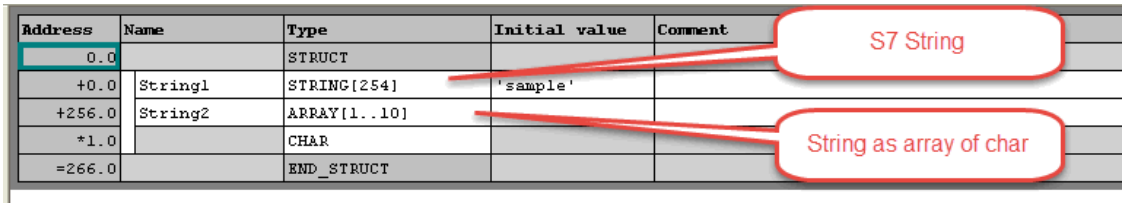
Note: Aliasing tag names is only available for imported tags. Tags added manually in the Tag Editor cannot have the Alias prefix in the tag name.
The Alias string is attached at the time of tag import. If you modify the Alias string after the tag import has been completed, there will be no effect on names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

String data type

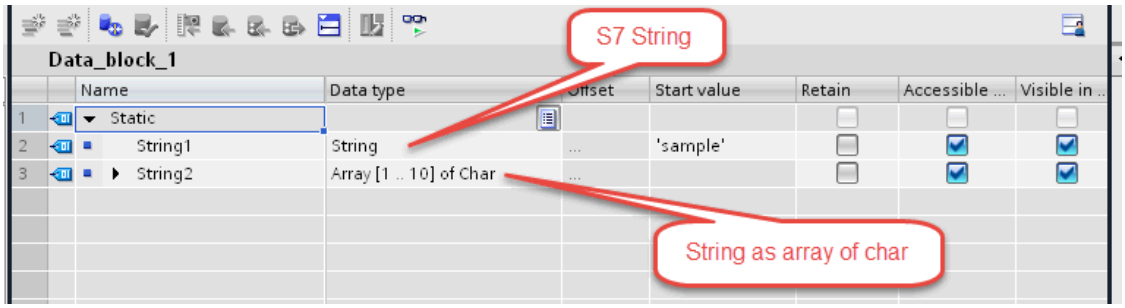
In Simatic S7 PLC two different types of tags manage string variables:

- as Array [1..xx] of characters,
- as String[xx].

Step7 string declaration is shown in this example:



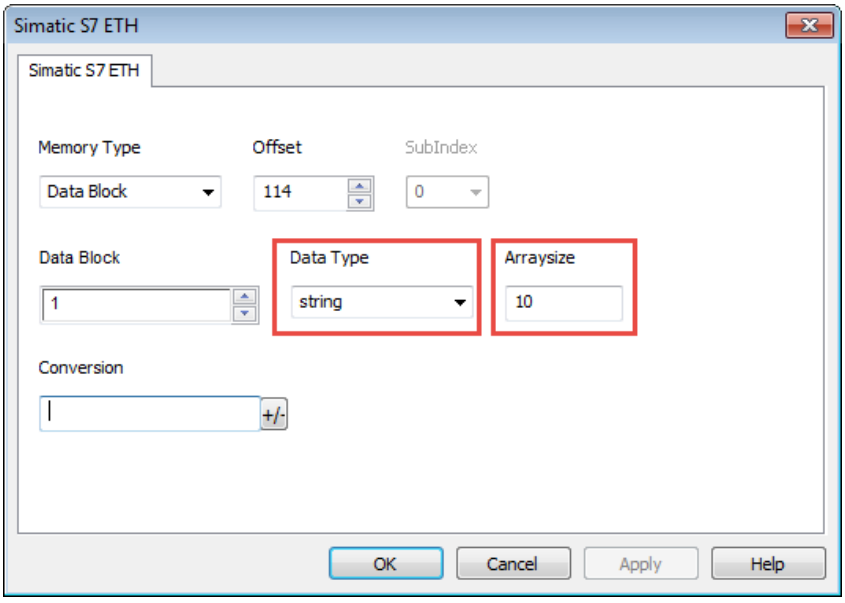
TIA Portal string declaration is shown in this example:



Note: When using String[xx] data type specific a conversion must be applied to the tag. If the tag dictionary is imported from TIA Portal or Step7 using the import tool, however, conversion of the string tags is performed automatically and no further action is required.

To add a string as an array of characters:

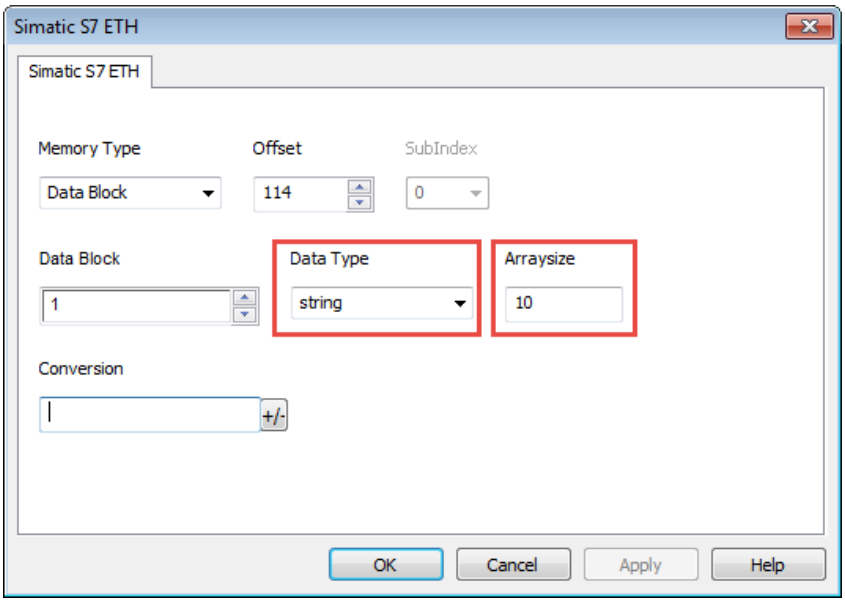
1. Press the **+** in the Tag Editor.



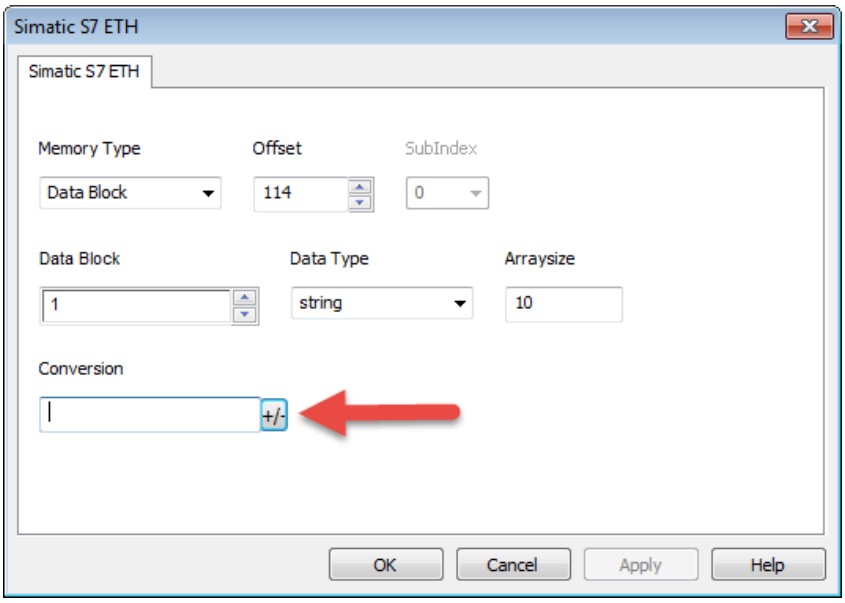
2. Select **string** as **Data Type.**
3. Enter string length in **Arraysize**.
4. Click **OK** to confirm.

To add a string data type:

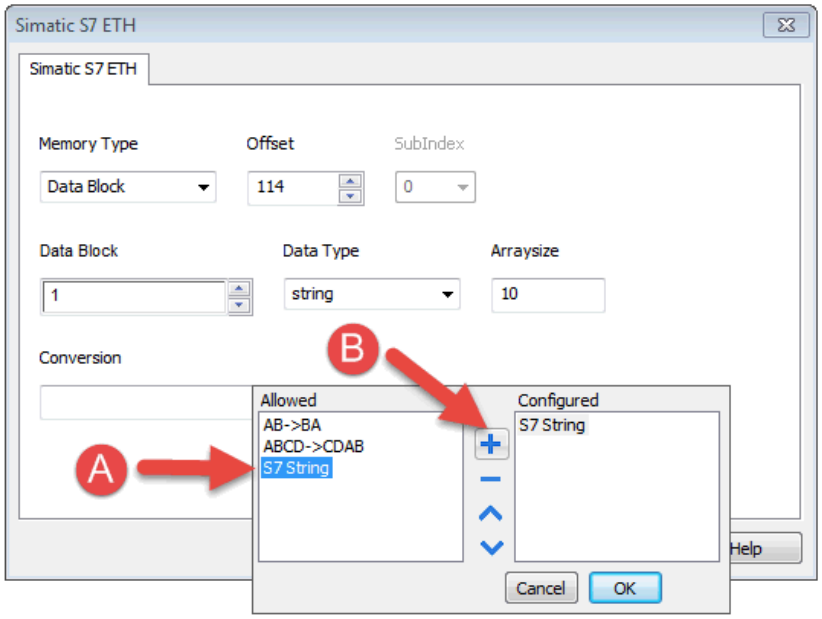
1. Press the **+** in the Tag Editor.



2. Select **string** as **Data Type.**
3. Enter string length in **Arraysize**.
4. Click **+/-** to open the Conversion dialog.



5. In the conversion dialog select the **S7 String** conversion type.

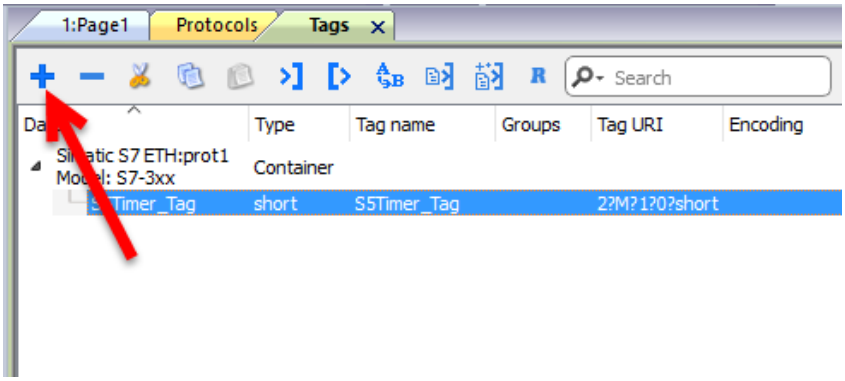


6. Click **+** to add the conversion: the conversion will be listed into the **Configured** list on the right.
7. Click **OK** to confirm.

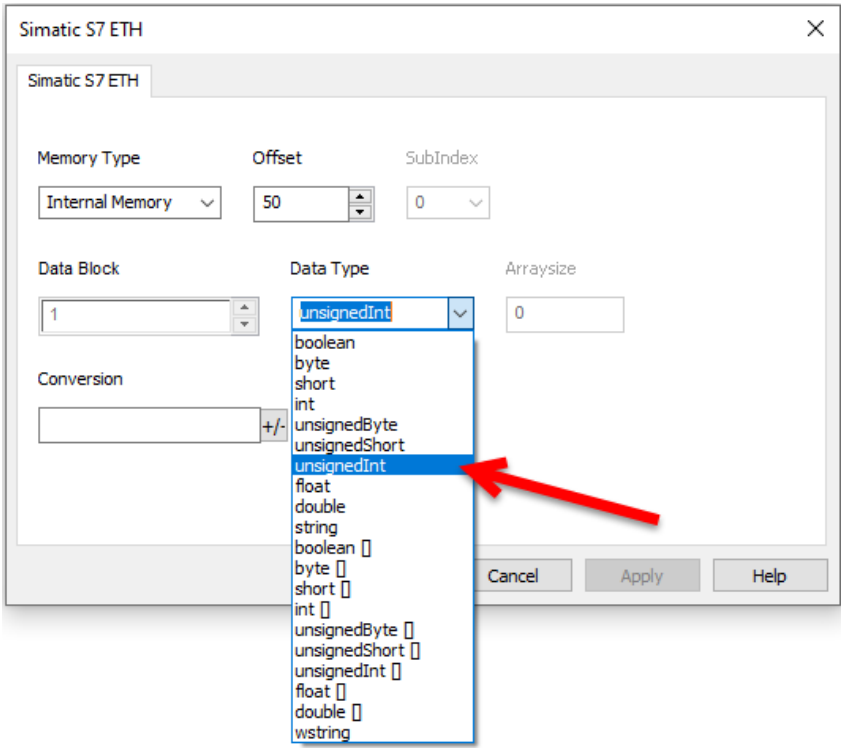
Simatic S5Timer data type

Simatic drivers support a special data type, the S5Timer data type.
The tag must be configured with a specific data type and a conversion must be applied to the tag to correctly read/write a Simatic S5Timer Variable.

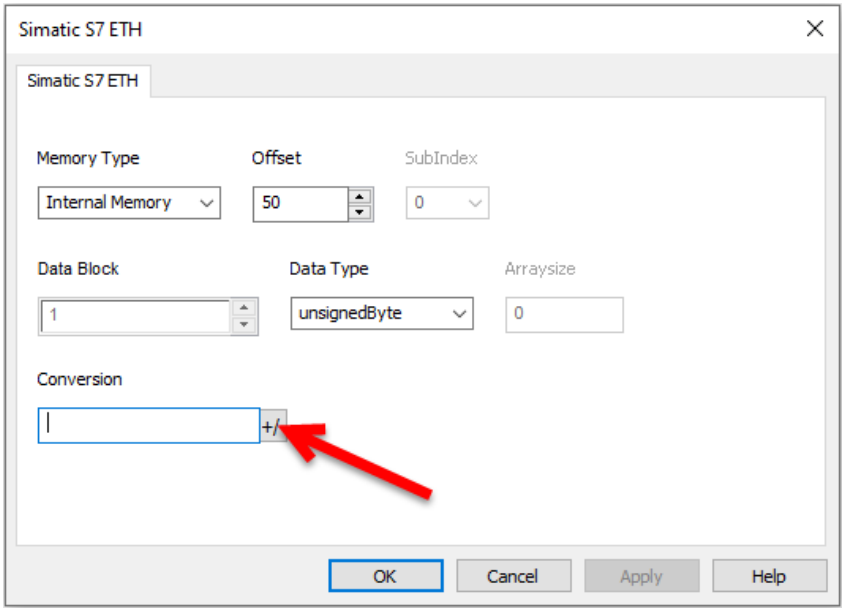
1. In the Tag Editor click **+** to add a tag.



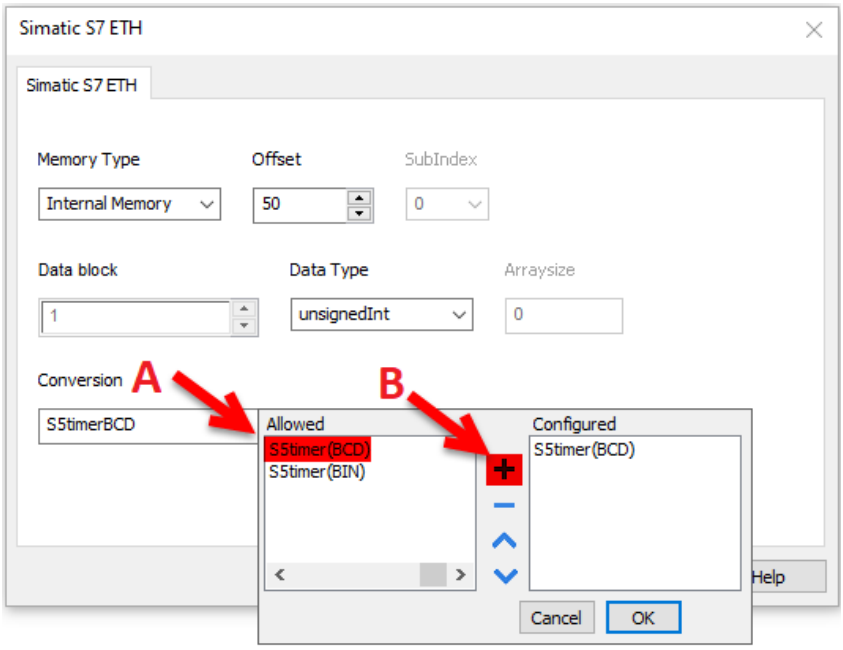
2. Select **unsignedInt** as **Data Type.**



3. Click **+/-** to open the Conversion dialog.



4. In the conversion dialog select the **S5timer(BCD)** conversion type.
5. Click **+** to add the conversion: the conversion will be listed into the **Configured** list on the right.



6. Click **OK** to confirm.

Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

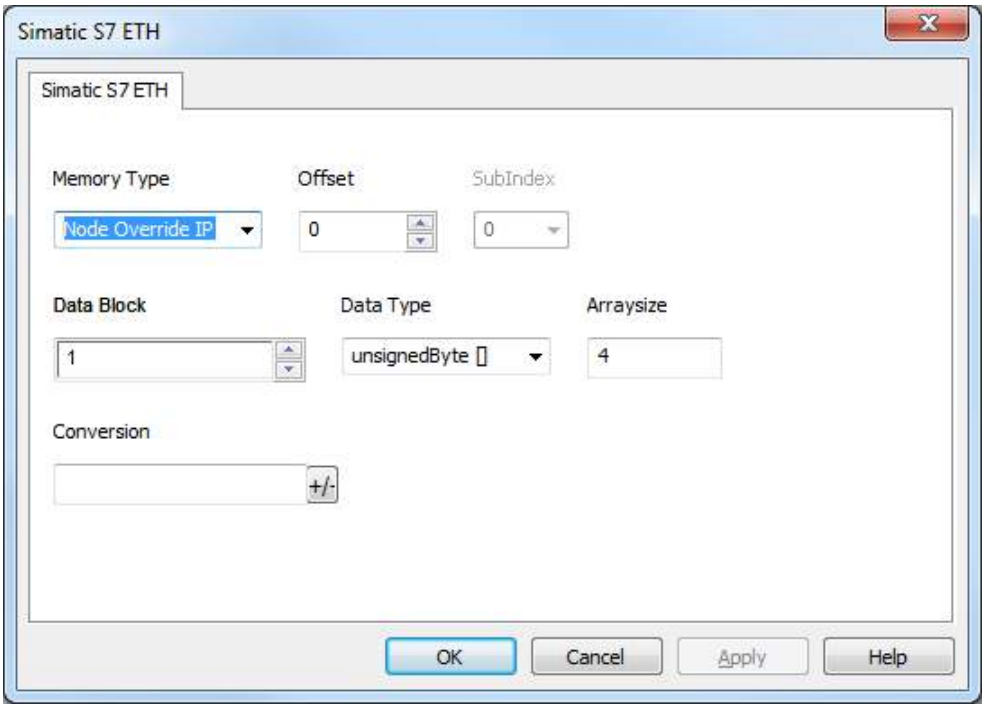
Node Override IP	PLC operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.

Note: Node Override IP values assigned at runtime are retained through power cycles.

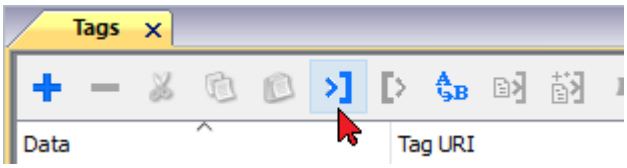
Hostname DNS or mDNS

In addition to the array of bytes, string memory type can be selected to be able use the DNS or mDNS hostname as an alternative to the IP Address.

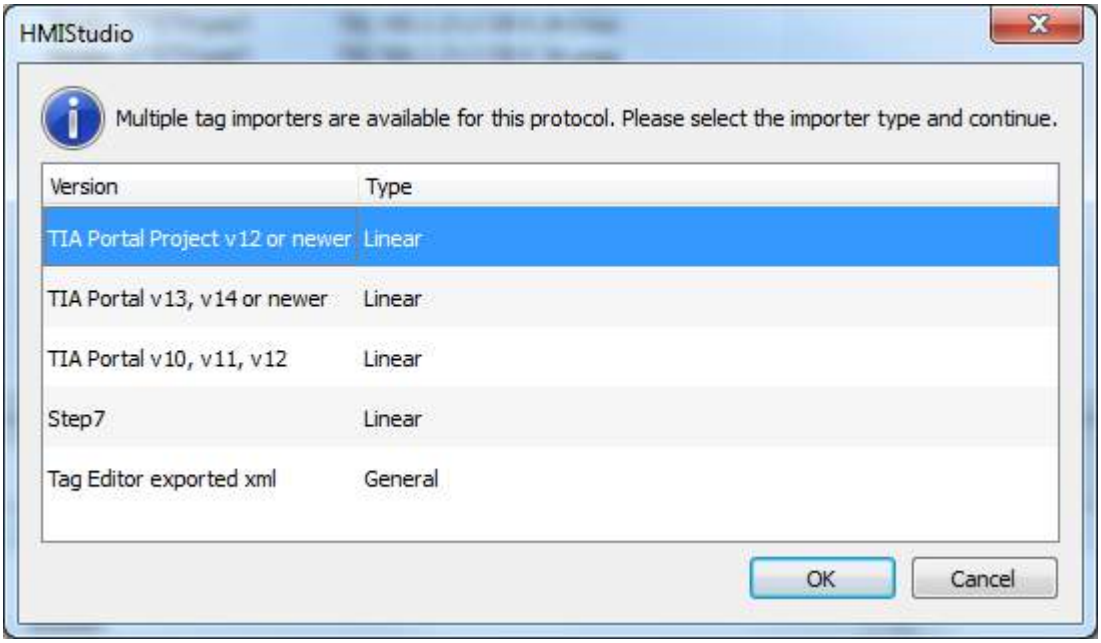


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



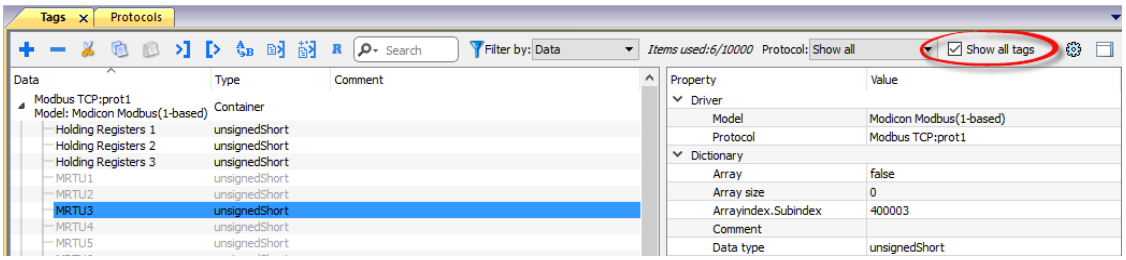
The following dialog shows which importer type can be selected.

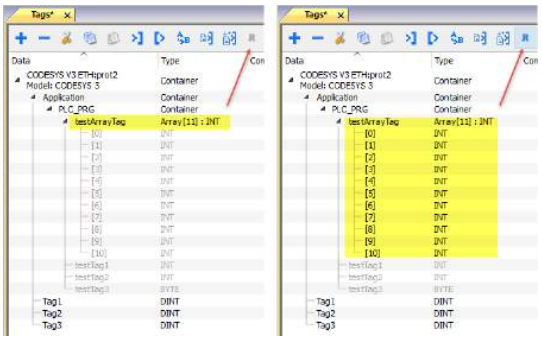
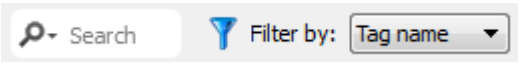


Importer	Description
TIA Portal Project v12 or newer **Linear**	Allows to import the whole TIA Portal project file using **.apxx** file (where "xx" is the TIA Portal version, example: for TIA Portal 13 , file name is "project.ap13"). All variables will be displayed at the same level.
TIA Portal v13, v14 or newer **Linear**	Allows to import: • Program blocks using **.db** file • PLC tags using **.xlsx** file • PLC data types using **.udt** file Check **Export using TIA Portal v13, v14 or newer** for more details. All variables will be displayed at the same level.
TIA Portal v10, v11, v12 **Linear**	Allows to import: • Program blocks using **.tia** file • PLC tags using **.xlsx** file • PLC data types using **.scl** file Check **Export using TIA Portal v10, v11, v12** for more details. All variables will be displayed at the same level.
Step7 **Linear**	Allows to import: • Symbols table **.asc** file • Sources using **.awl** file Check **Export using STEP7** for more details. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

Current communication status can be displayed using system variables. See "System Variables" section in the main manual.

Codes supported by this communication driver:

Error	Cause	Action
NAK	The controller replies with a not acknowledge.	-
Timeout	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
Invalid	The device did received a response with invalid	Ensure the data programmed in the project are

Error	Cause	Action
response	format or contents from the controller .	consistent with the controller resources.
General Error	Unidentifiable error. Should never be reported.	Contact technical support.

Simatic S7 MPI

HMI products support direct Siemens MPI communication without any additional module.

The driver supports the standard communication speed 187Kbit/s.

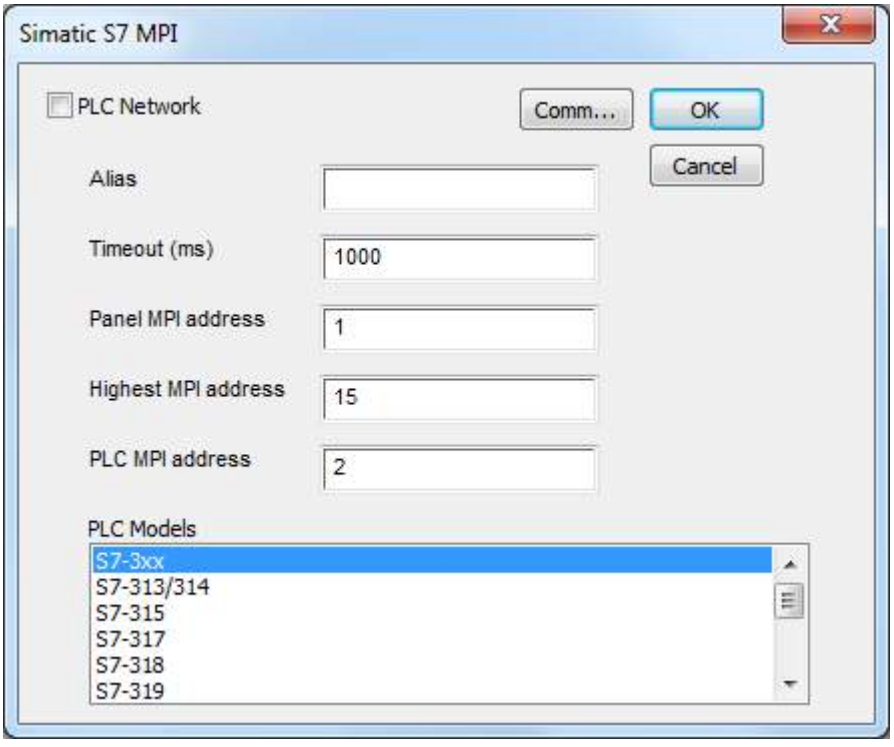
here is a minimum requirement also for the version of operating system running in the HMI (this is normally referenced as BSP version). See in user manual how to read the BSP version with the System Settings menu. The minimum requirements are shown in the following table.

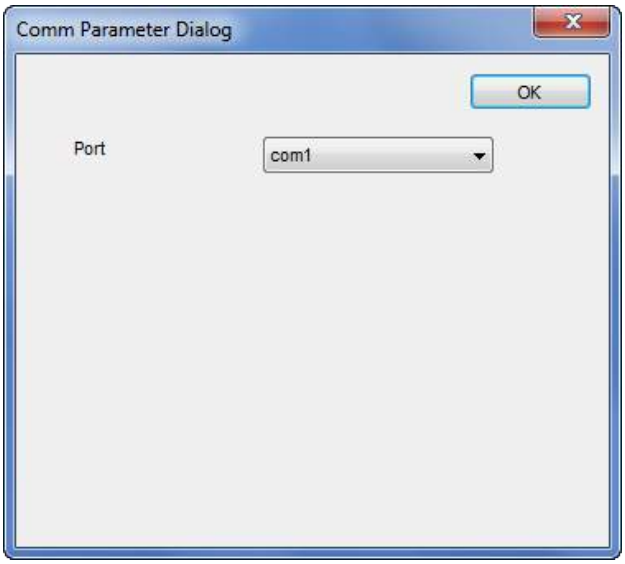
Platform	BSP Version
UN30/31	v1.38 or newer
UN65/UN71	v1.0.300 or newer
UN60/UN70	v1.0.413 or newer
UN73	v1.0.142 or newer

Protocol Editor Settings

Add [+] a driver in the Protocol editor and select the “Simatic S7 MPI” protocol from the list of available protocols.

The protocol type can be selected from the dedicated combo box in the dialog.



Element	Description
Alias	Name to be used to identify nodes in the plc network configuration. The name will be added as a prefix to each tag name imported for each network node.
Timeout (ms)	Defines the time inserted by the protocol between two retries of the same message in case of missing response from controller. Value is expressed in milliseconds.
Panel MPI Address	MPI node number assigned to the device.
Highest MPI Address	The highest node number in the MPI network where the device is operating and communicating.
PLC MPI Address	The MPI address of the controller to which the device needs to communicate.
PLC Models	List of compatible controller models. Make sure to select the correct PLC model in this list when configuring the protocol.
Comm…	Click on this button to configure the serial port on the device to be used as MPI port (see example in the following figure)  Communication parameters for Simatic S7 MPI are fixed at: • Baud rate=187500 • Parity=Even • Data=bits8 • Stop=bit1 On UN20: • com1 is the HMI port labeled “PLC”, • com2 is the HMI port labeled “PC/Printer”

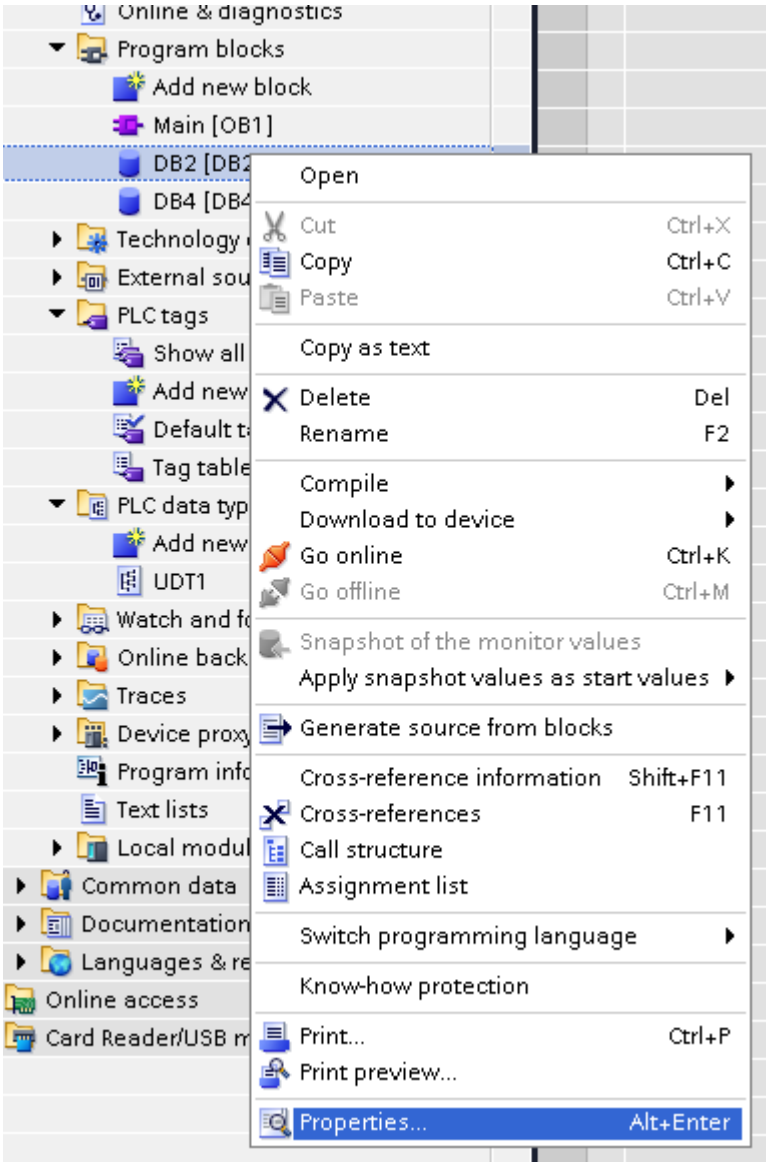
Element	Description
	On UN31 or UN30: • com1 is the integrated serial port, • com2 is an add-on module plugged in Slot#1 or #2 • com3 is an add-on module plugged in Slot#3 or #4 The connection between device and PLC can be made: • Creating a custom cable following the scheme provided with document CA255 "*eTOP400/500 serie PLC Port to MPI Port*" • Using a standard MPI cable with ADP-0001 "*MPI wiring adapter*"
PLC Network	The protocol supports connection to multiple controllers. To enable this option, check the "PLC Network" check box and enter the configuration per each controller node.

Direct Import of TIA Portal project

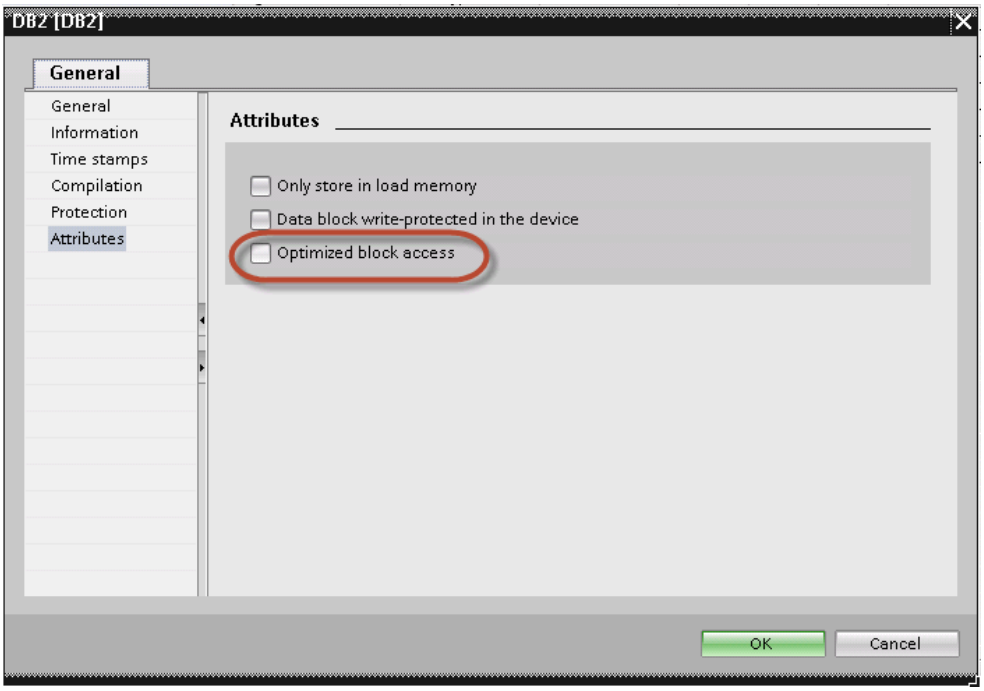
It is possible to import TIA Portal variables directly from TIA Portal project, by selecting "TIA Portal Project v12 or newer" from import selection (refer to "Tag Import" chapter).

Data Blocks must be set as Not optimized:

1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:



3. In the **General** tab select **Attributes** and unselect **Optimized block access**.



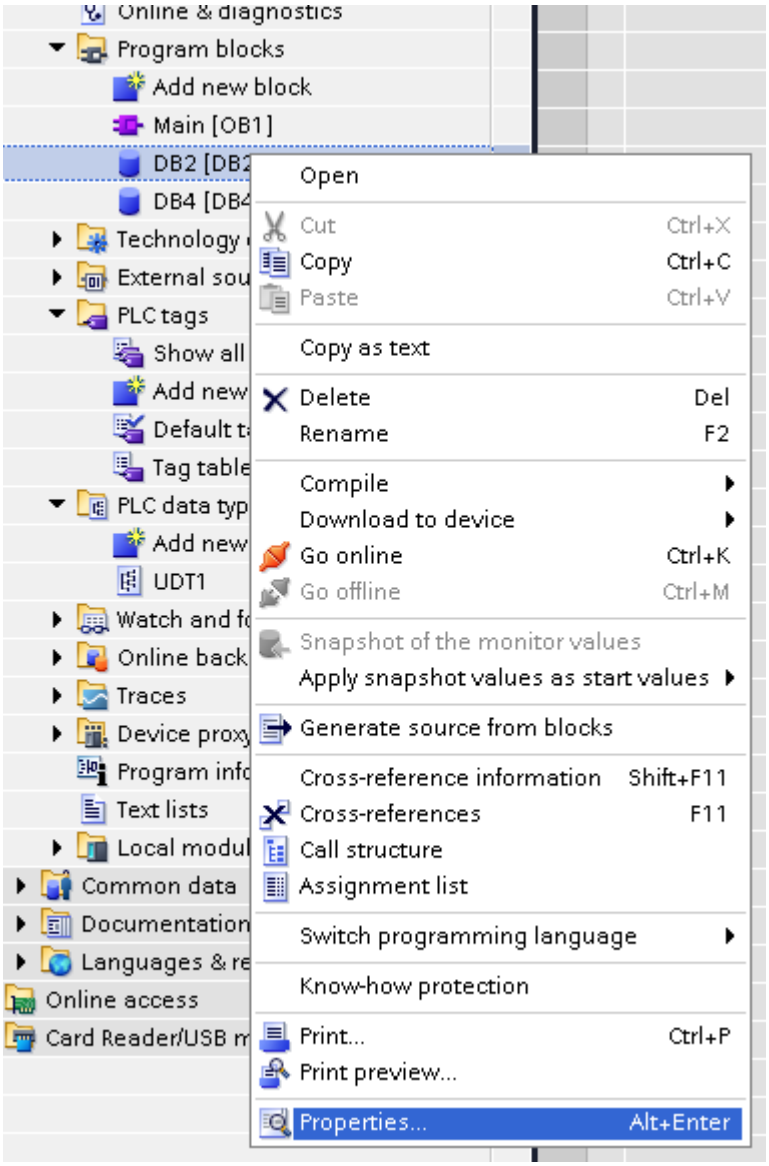
Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

Export using TIA Portal v13, v14 or newer

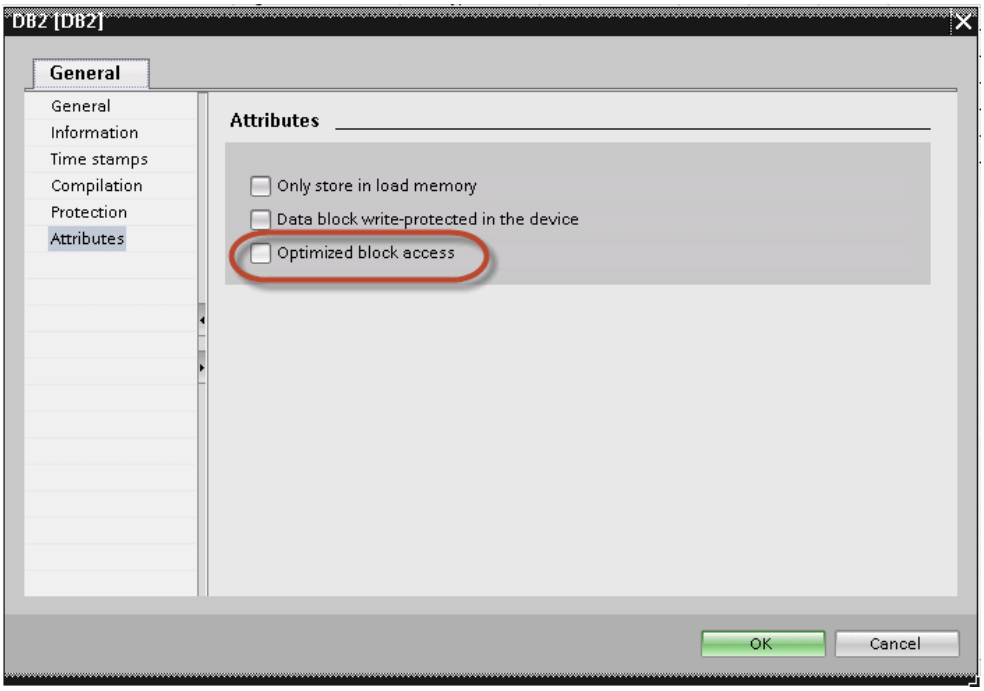
Exporting Program blocks

These files refer to DB tags defined in **Program blocks**.

1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:

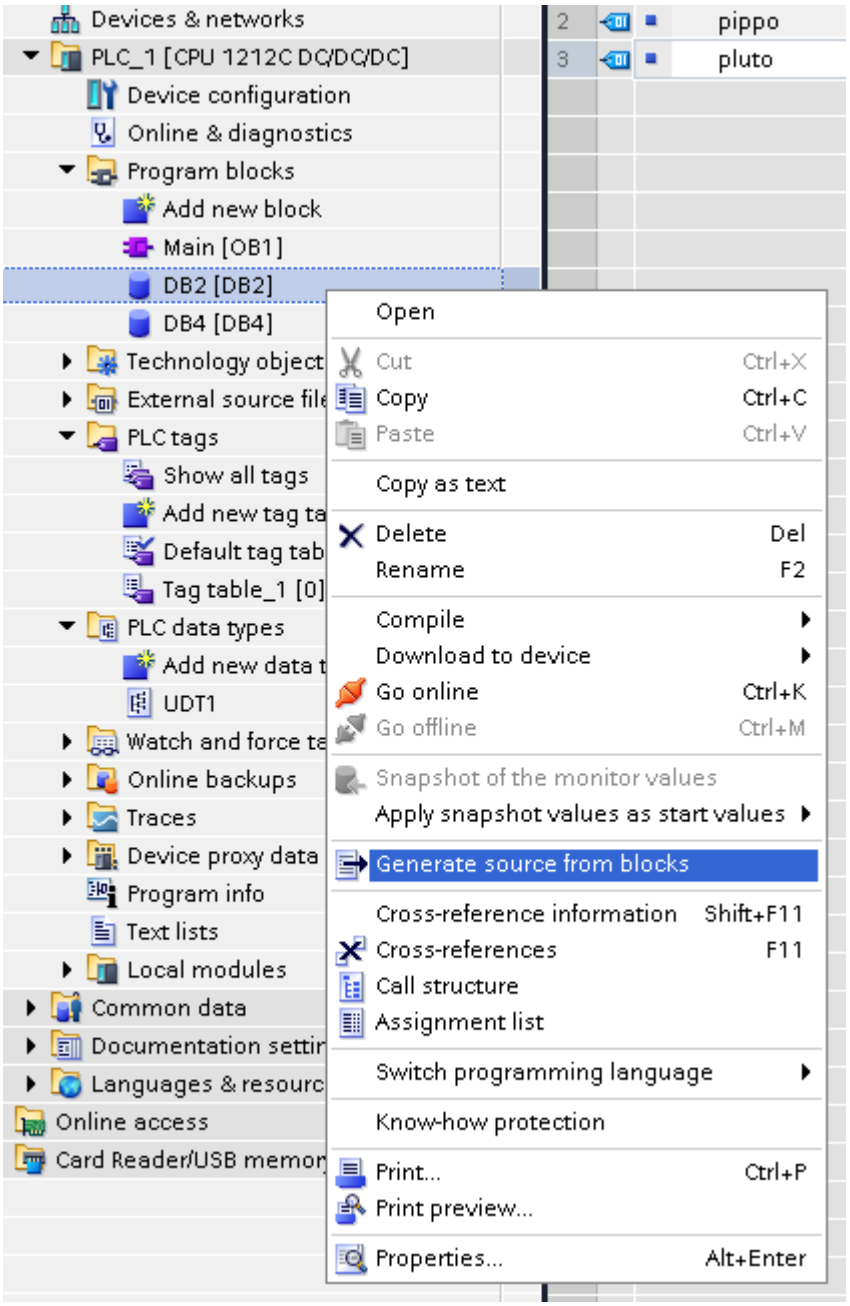


3. In the **General** tab select **Attributes** and unselect **Optimized block access**.

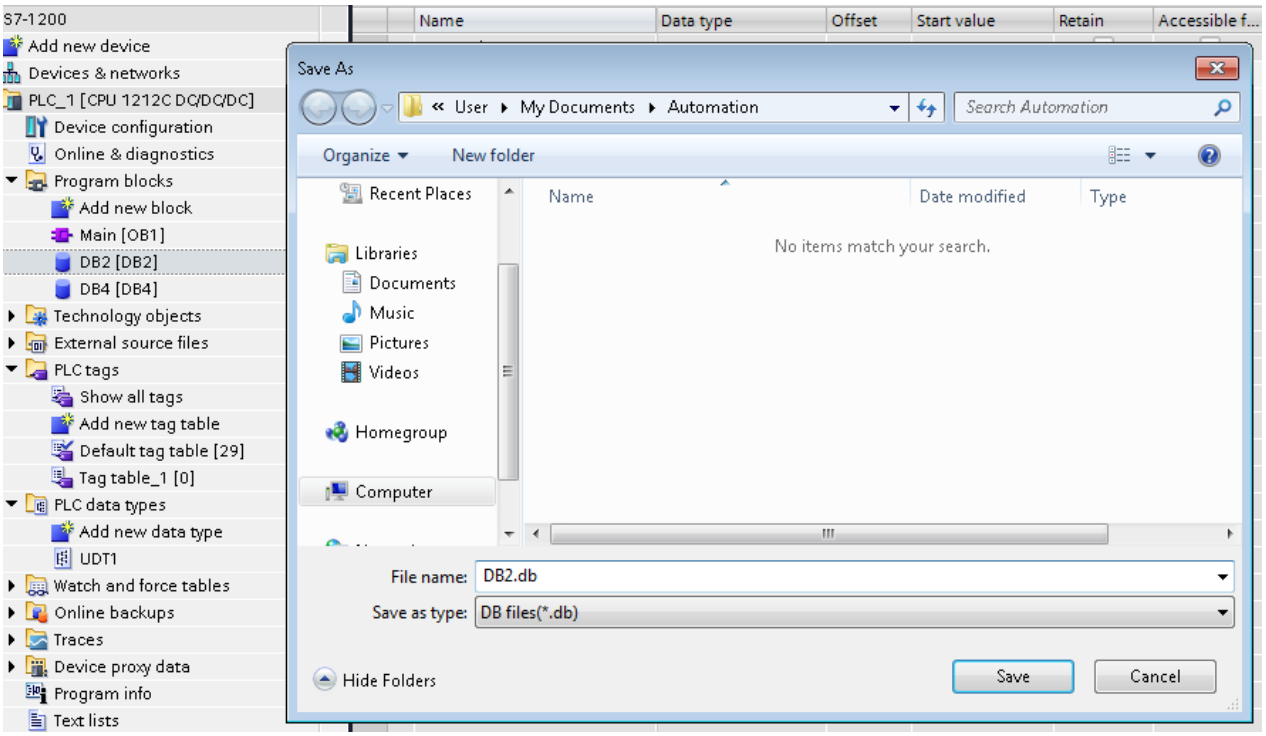


Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".

4. Right-click on the Data Block and choose **Generate source from blocks**:



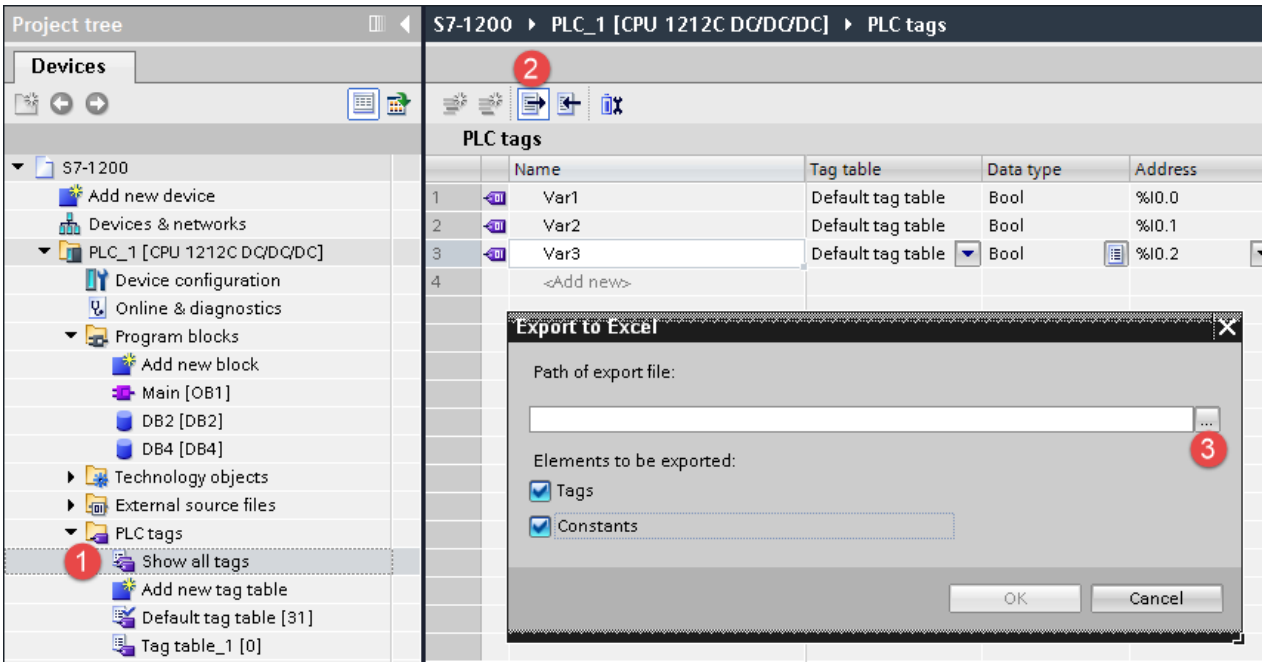
5. Save the file as DBxxx.db, where xxx=number of DB.



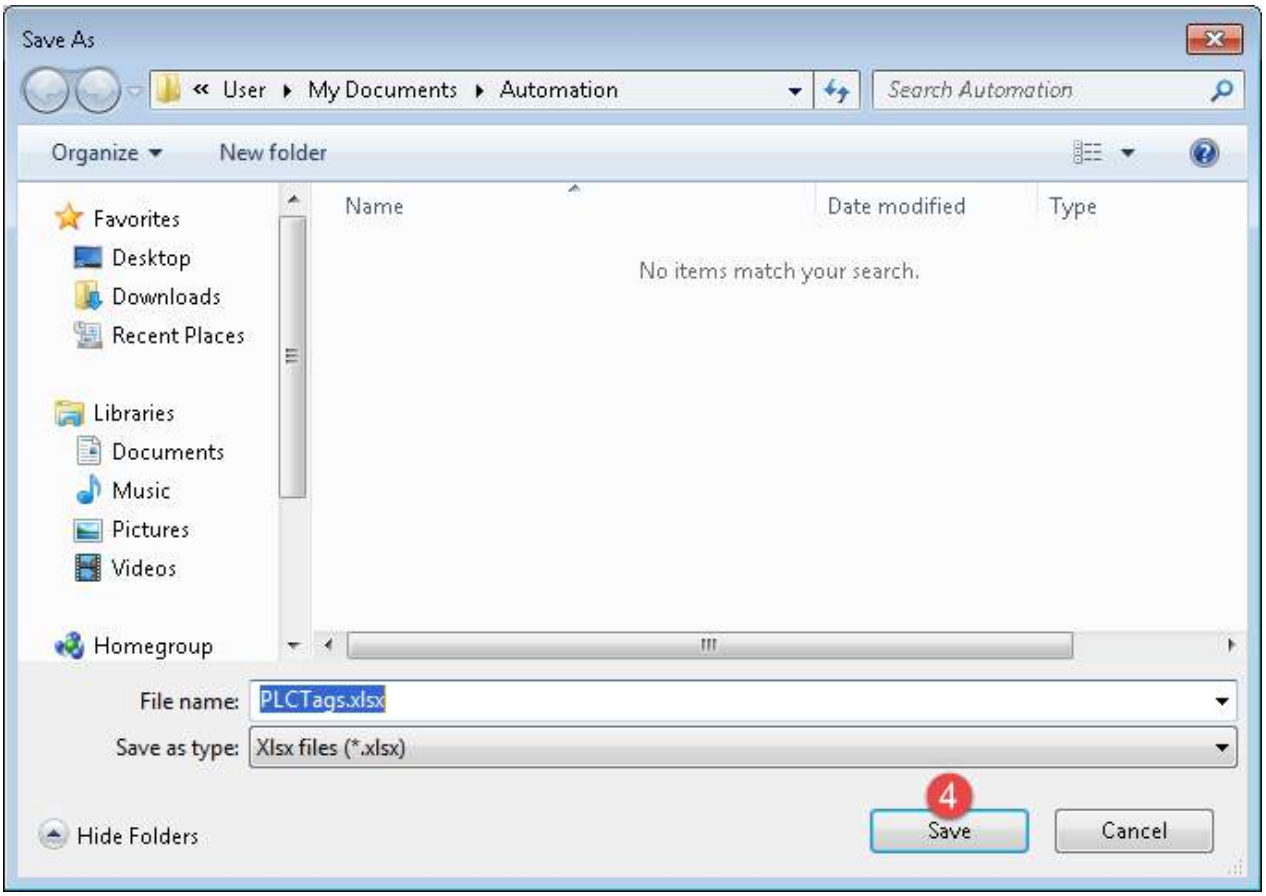
Exporting PLC tags

An Excel file refers to PLC tags.

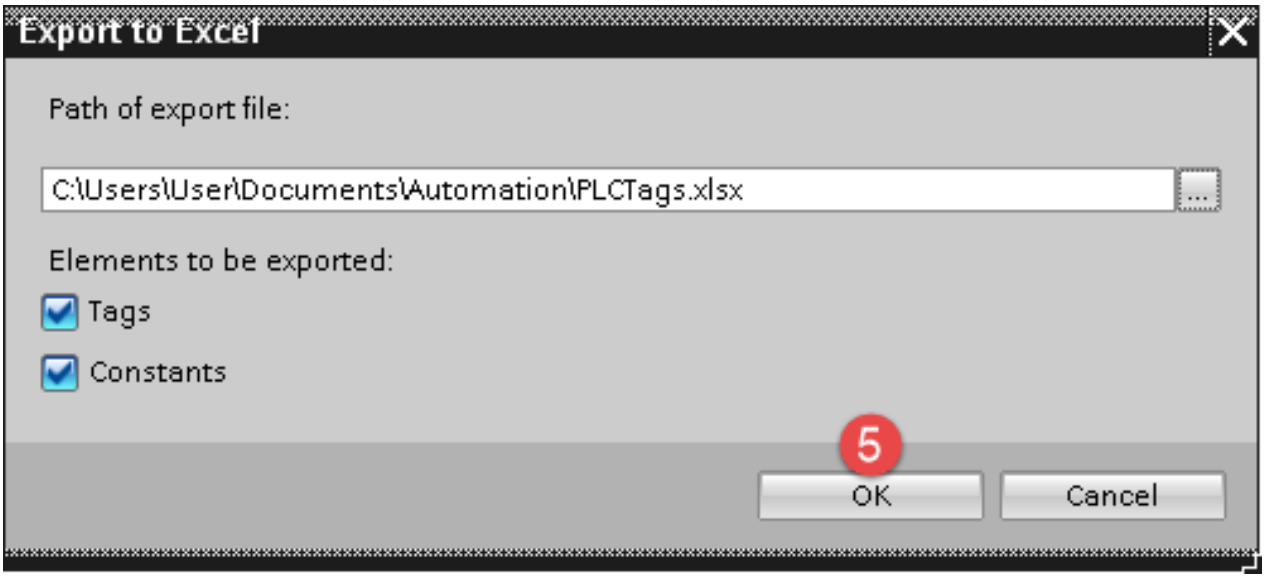
1. Double-click **Show all tags**: the tag table is displayed.
2. Click the **Export** button and browse for path file.
3. Define file name.



4. Click **Save** to confirm.

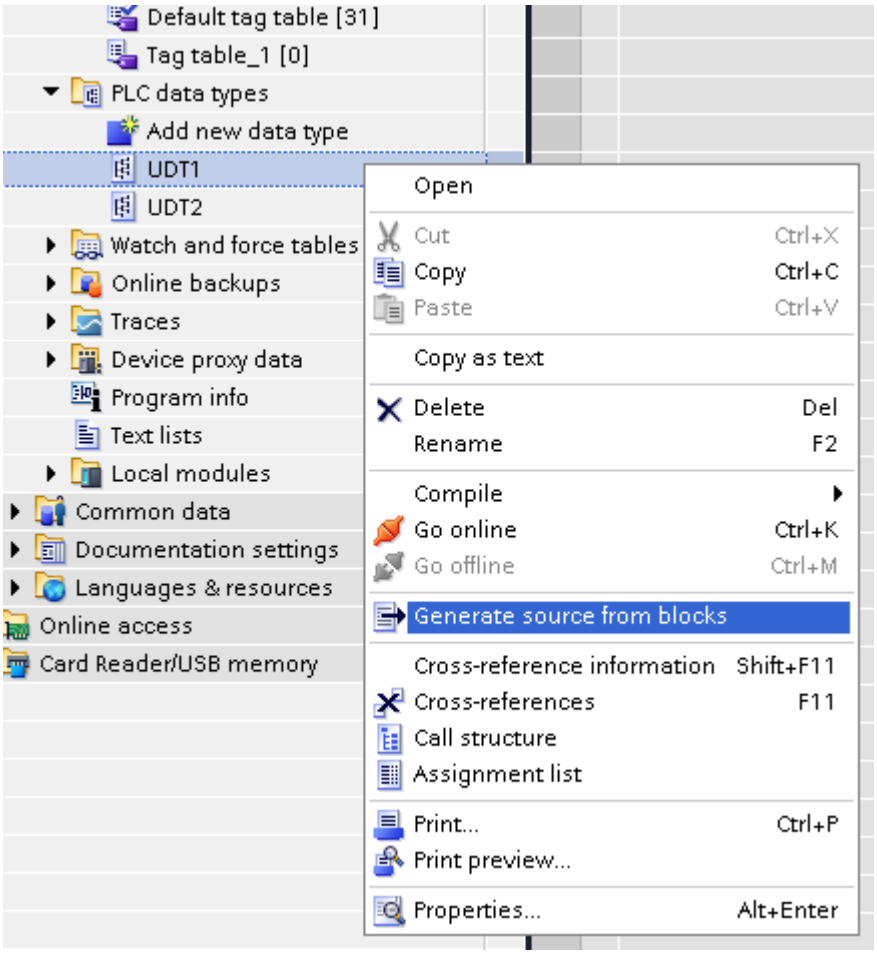


5. Click **OK** to export.

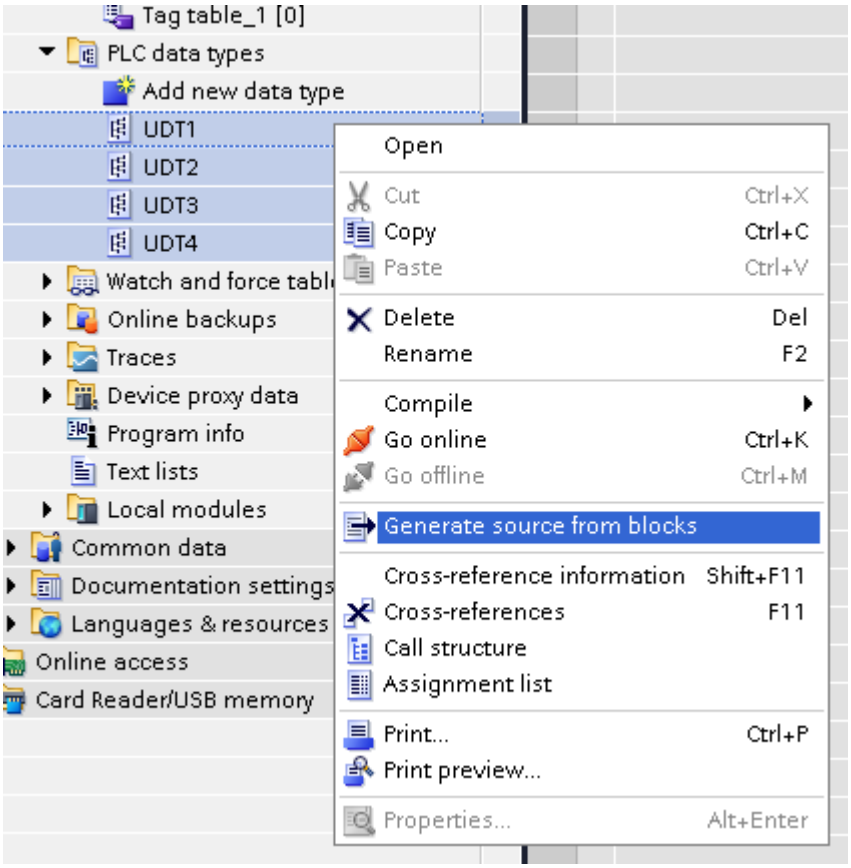


Exporting PLC data types

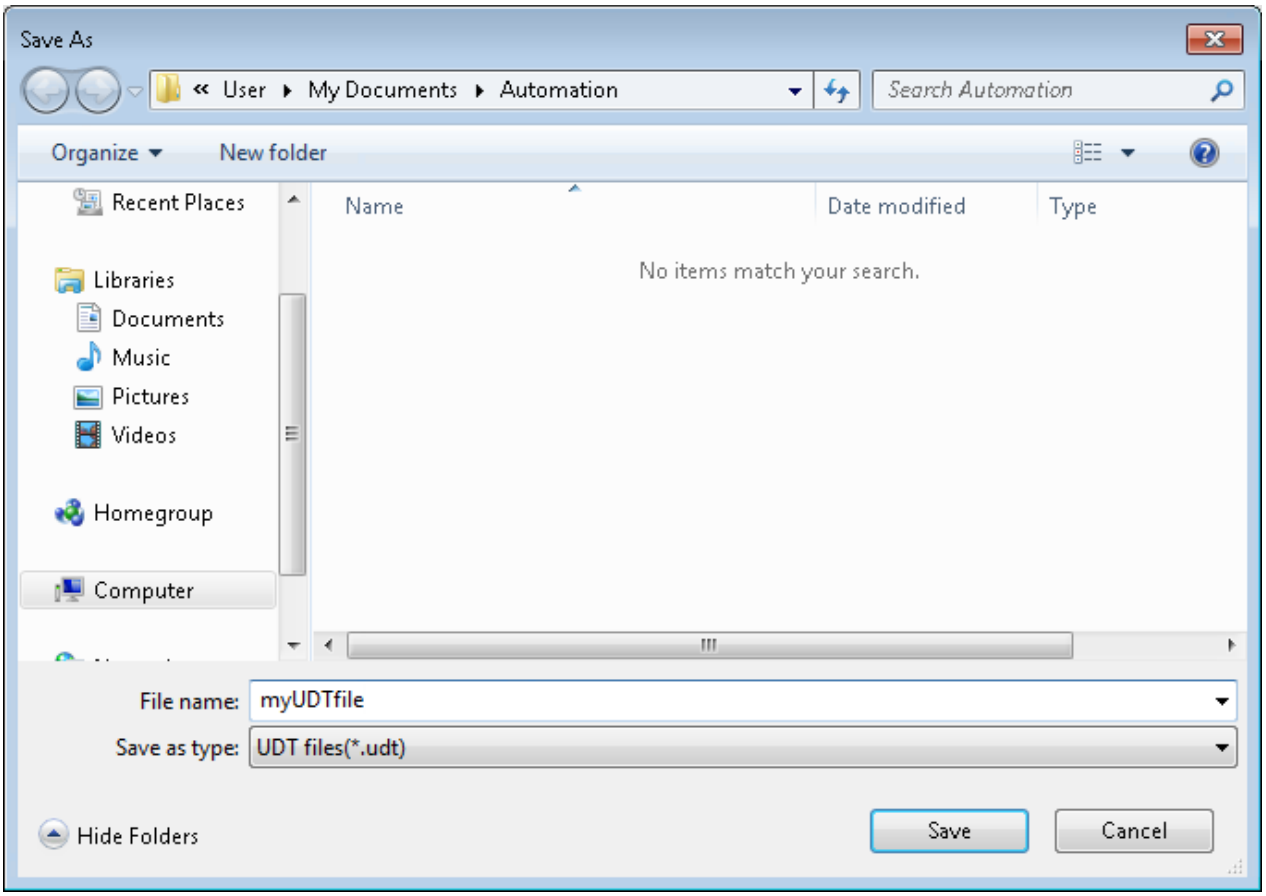
To create the file, expand **PLC data types** item from TIA Portal project tree and right click on the user defined structure. Then click on **Generate source from blocks**.



In case of multiple PLC data types in PLC project, it is necessary to select them all from **PLC data types** list, right click and select **Generate source from blocks** to create the .UDT file that contains all the PLC data types defined.



In the next step, give a name to the .UDT file and choose the path to where to save the file.



This file will content all the PLC data types and it can be used for importing tags in Tag Editor.

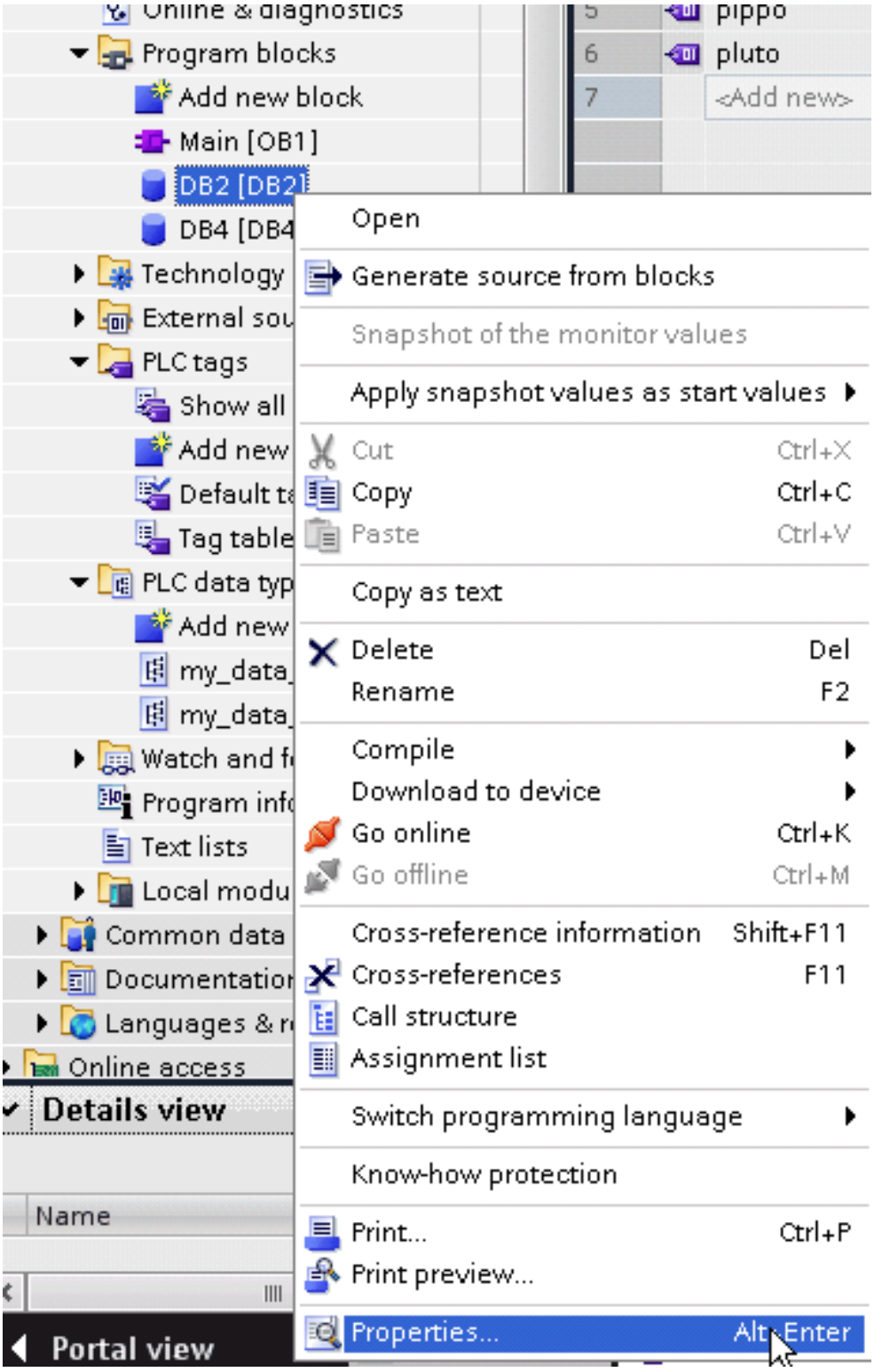
Check **Tag Import** chapter for more details.

Export using TIA Portal v10, v11, v12

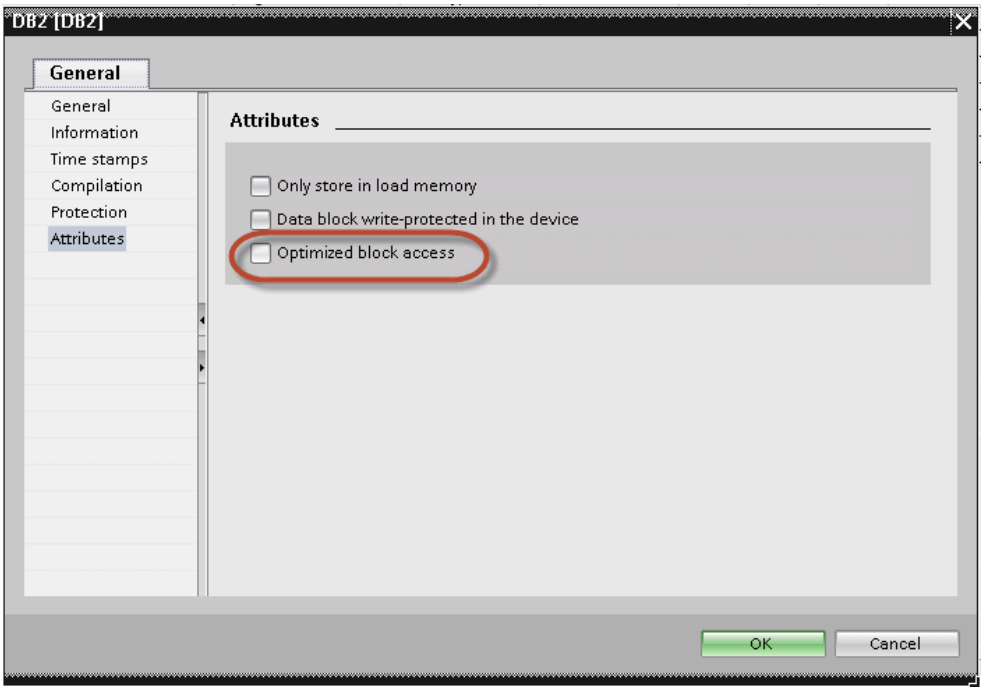
Exporting Program blocks

These files refer to DB tags defined in **Program blocks**.

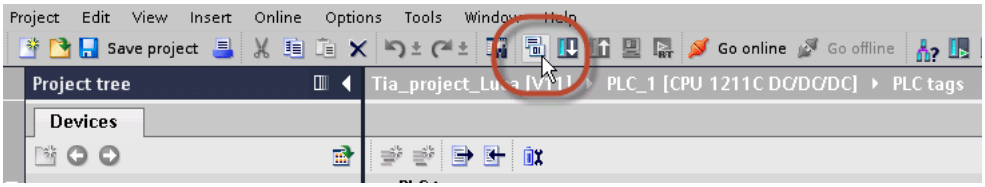
1. Configure the Data Block as **Not optimized**.
2. Right-click on the Data Block and choose **Properties**:



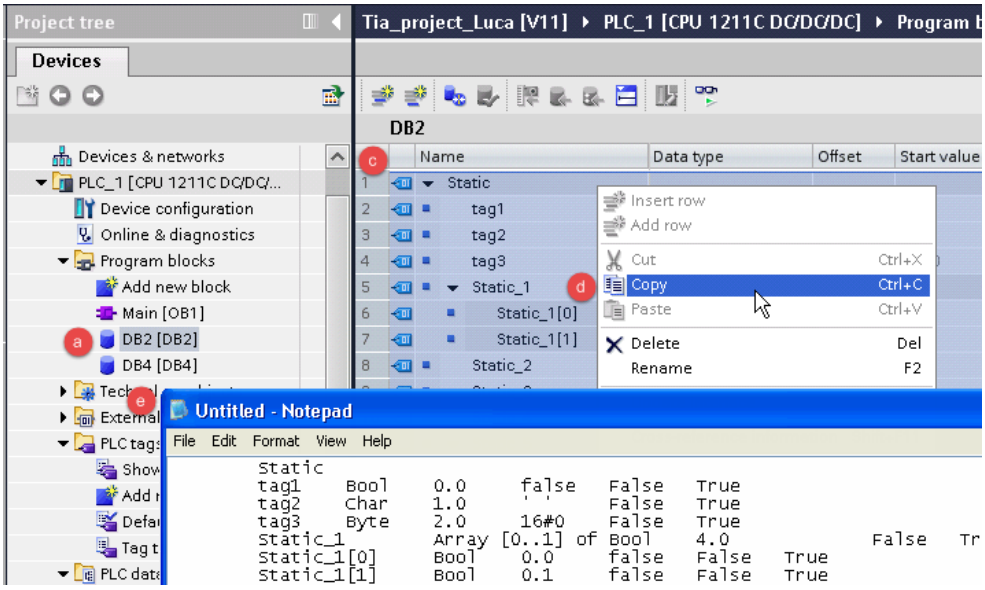
3. In the **General** tab select **Attributes** and unselect **Optimized block access**.



Note: If the options **Optimized block access** is not enabled (checkbox grayed out) this might mean that the Data Block is an "instance DB" linked to an "optimized access FB".



4. Build the project to make sure TIA Portal calculates the tags offset.

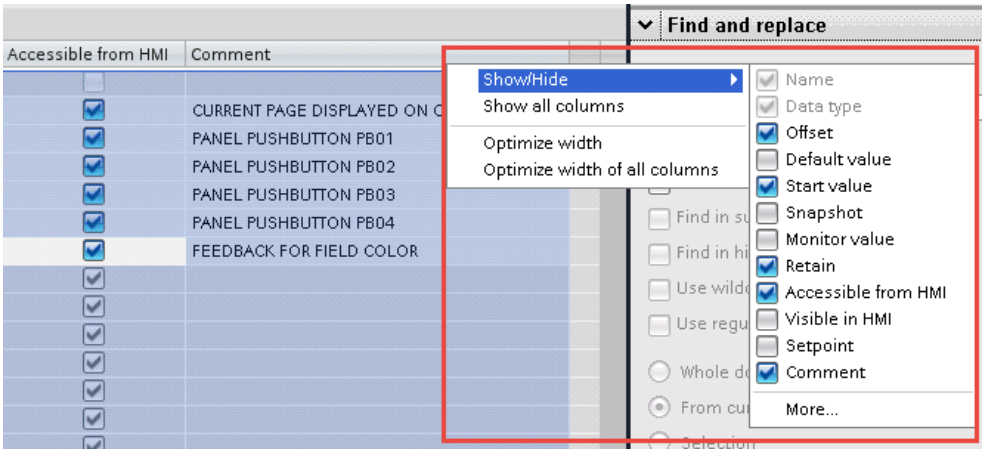


5. Double-click on a DB name.
6. Expand the view of program block selected.
7. Select all rows.
8. Copy and paste into any text editor.
9. Save the file as DBxxx.tia, where xxx=number of DB.

Note: Make sure you use the **Save As** function or the file will be named DB2.tia.txt and will not be visible from the importer.

10. Repeat from step 5 for all program blocks.

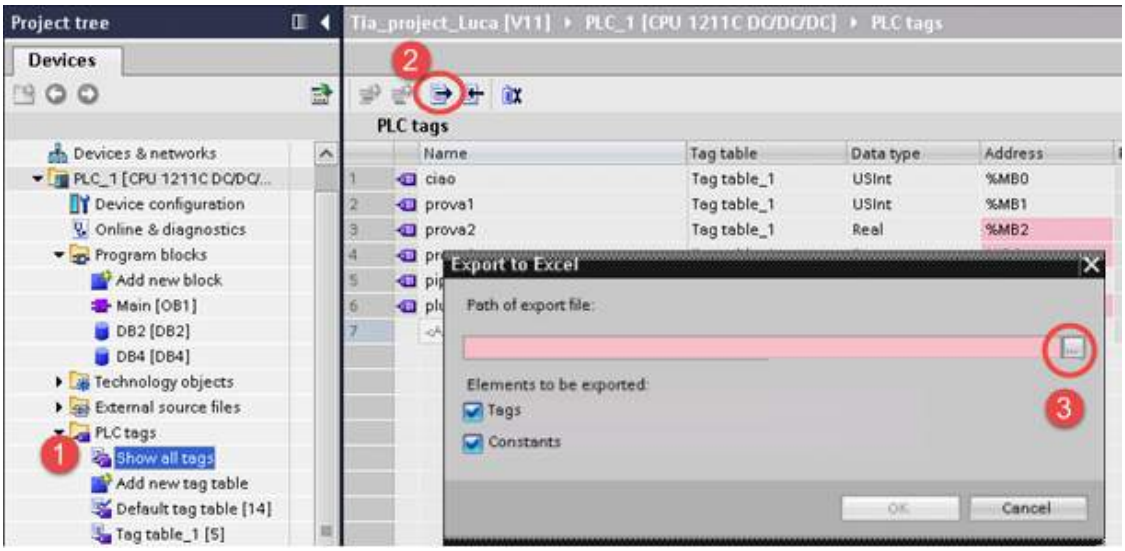
Note: Make sure that only the following columns are shown in DB editor before copying all data in the txt file



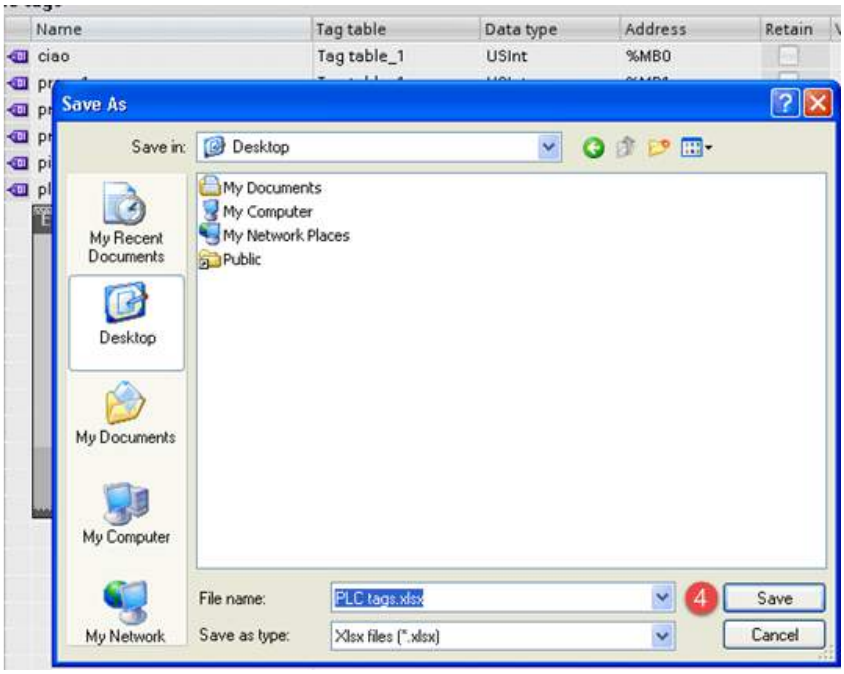
Exporting PLC tags

An Excel file refers to PLC tags.

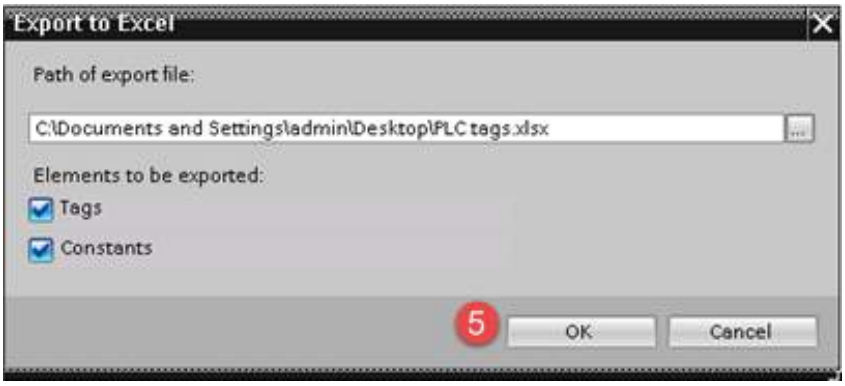
1. Double-click **Show all tags**: the tag table is displayed.



2. Click the **Export** button and browse for path file.
3. Define file name.
4. Click **Save** to confirm.

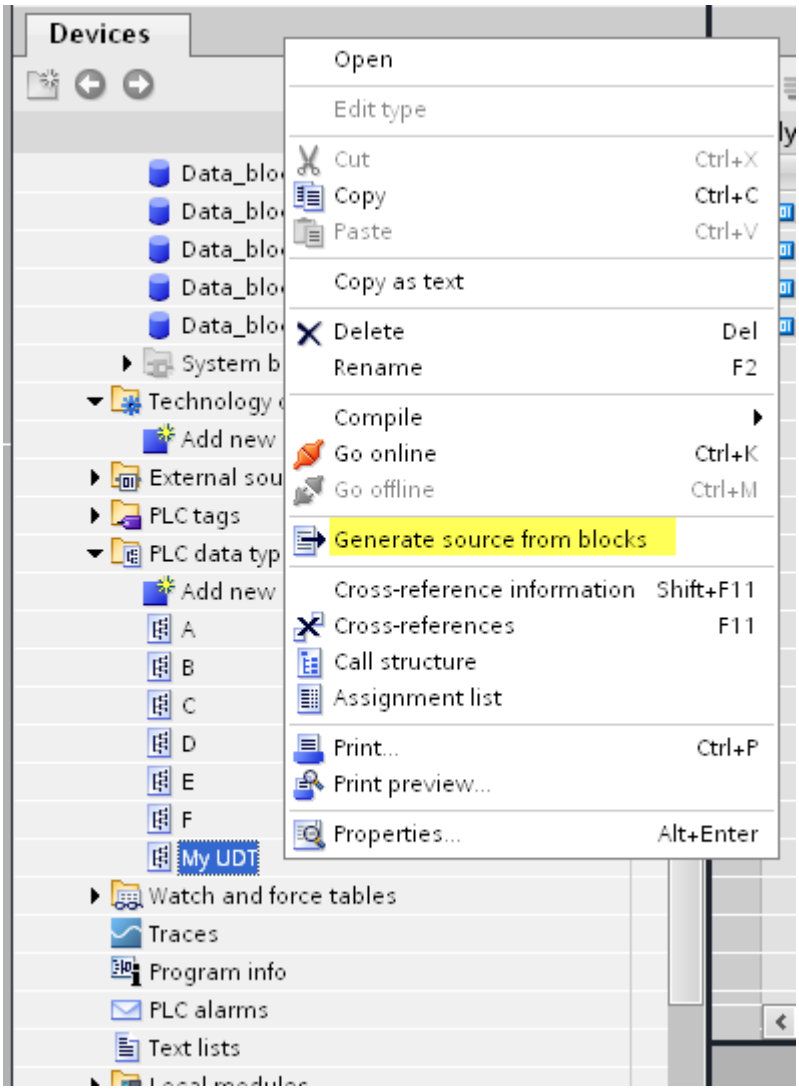


5. Click **OK** to export.

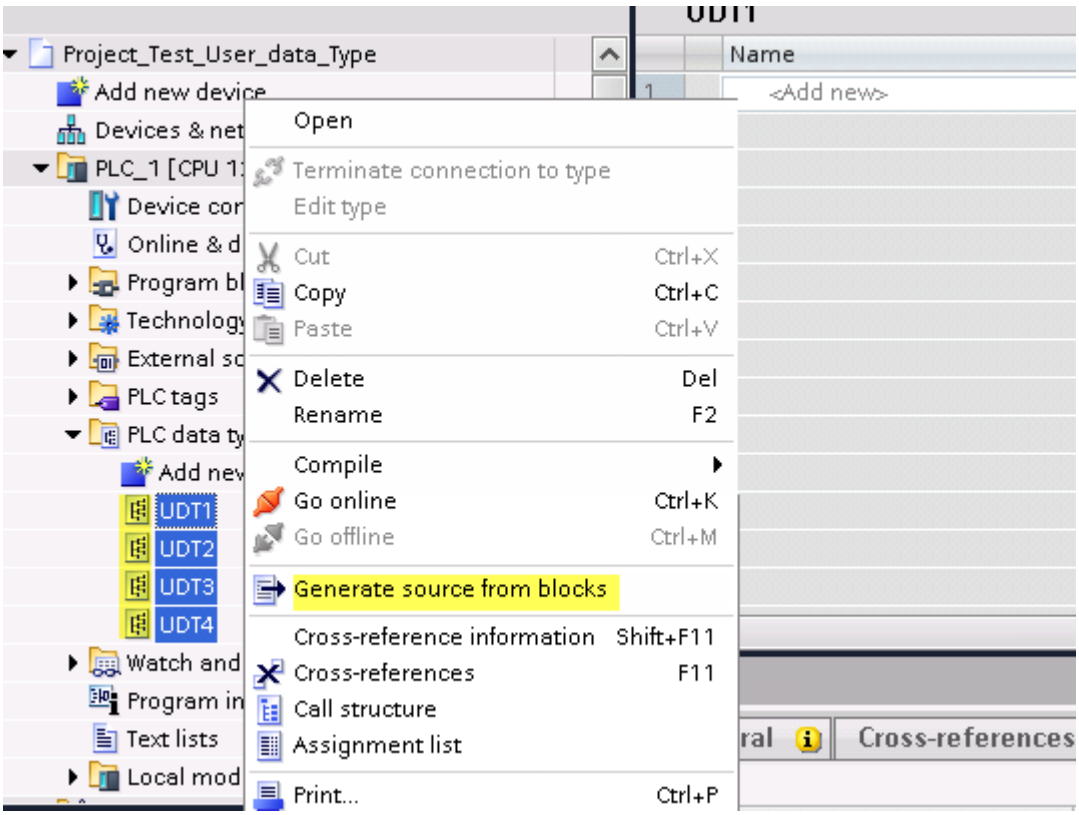


Exporting PLC data types

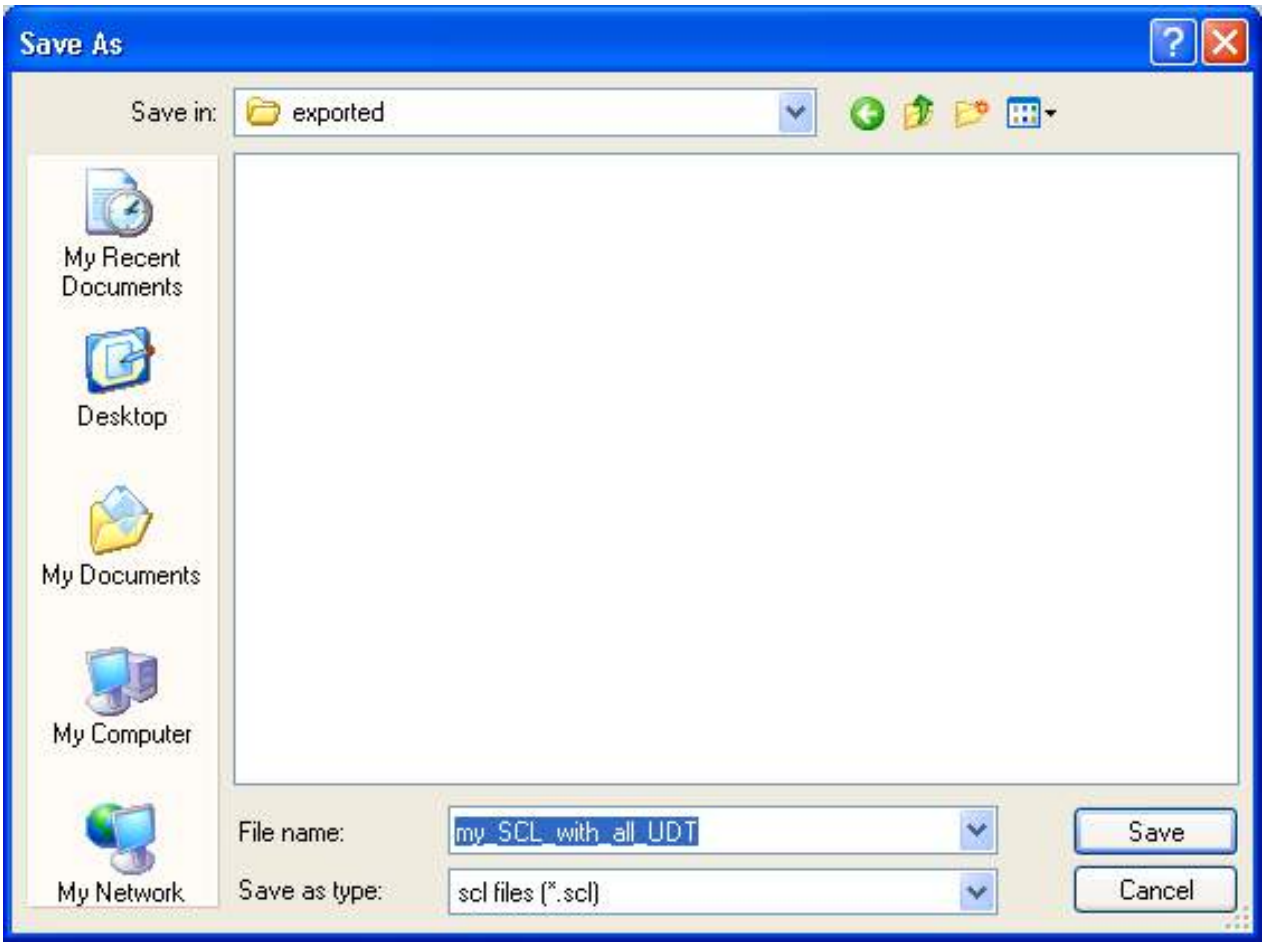
To create the file, expand **PLC data types** item from TIA Portal project tree and right click on the user defined structure. Then click on **Generate source from blocks**.



In case of multiple PLC data types in PLC project, it is necessary to select them all from **PLC data types** list, right click and select **Generate source from blocks** to create the .SCL file that contains all the PLC data types defined.



In the next step, give a name to the .SCL file and choose the path to where to save the file.



This file will content all the PLC data types and it can be used for importing tags in Tag Editor.

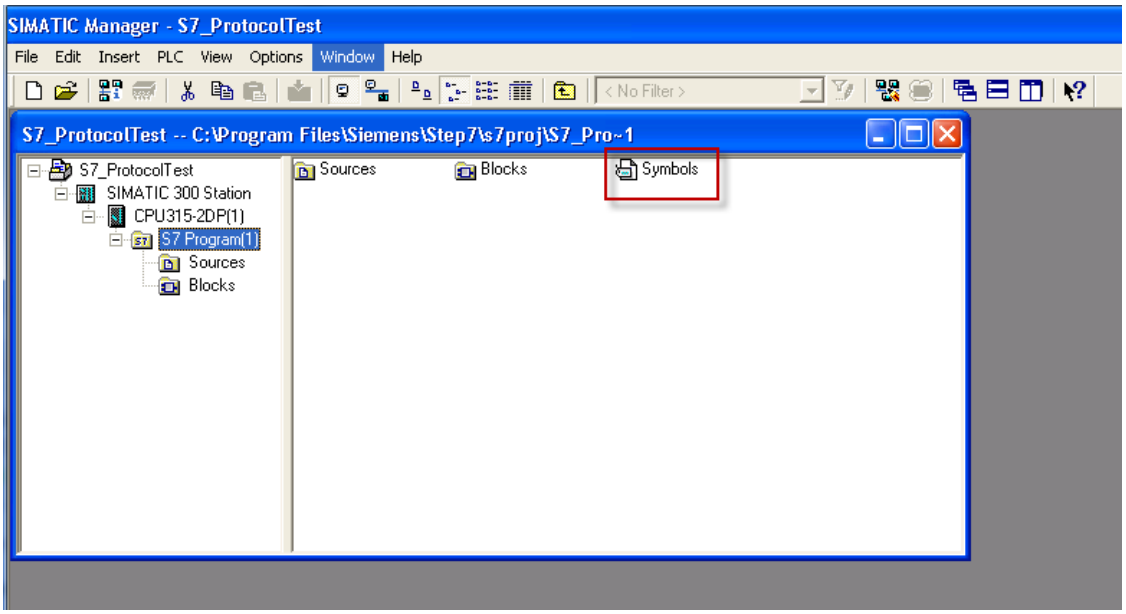
Check **Tag Import** chapter for more details.

Export using STEP7

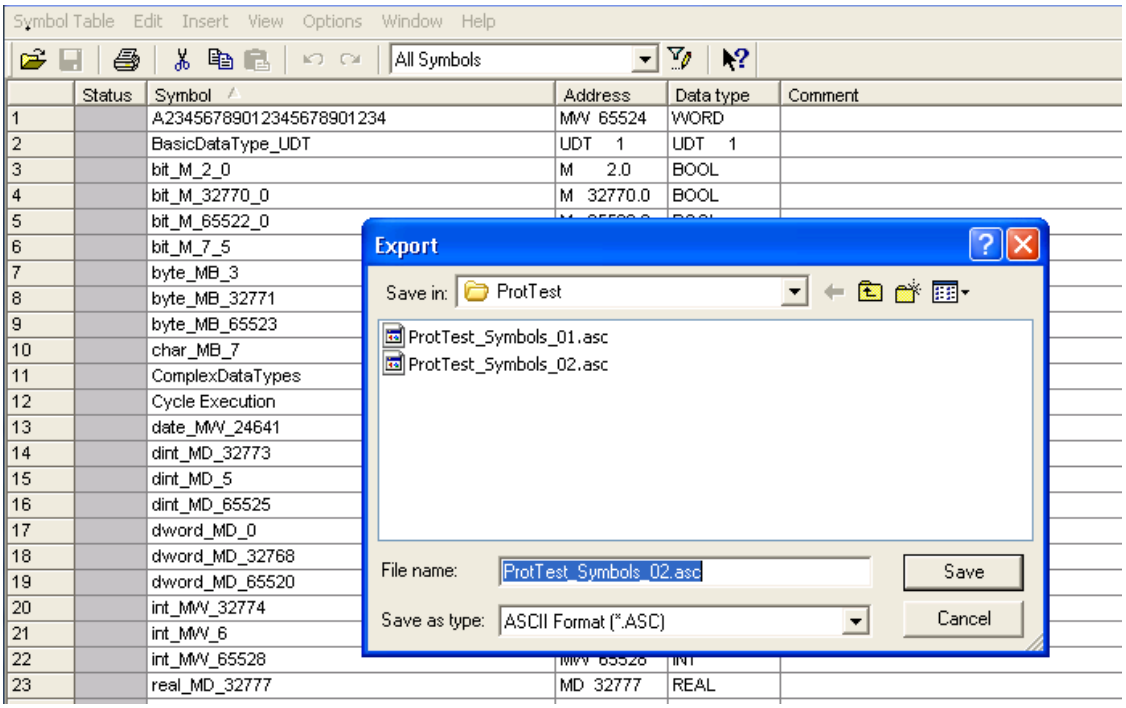
The Simatic S7 MPI Tag importer accepts symbol files (ASCII format .asc) and source files (.awl extension) created by the Simatic Step7. The symbol file can be previously exported using the Step7 symbol table utility.

Exporting Symbols table

Symbol files (.asc) can be exported from the symbol table utility.



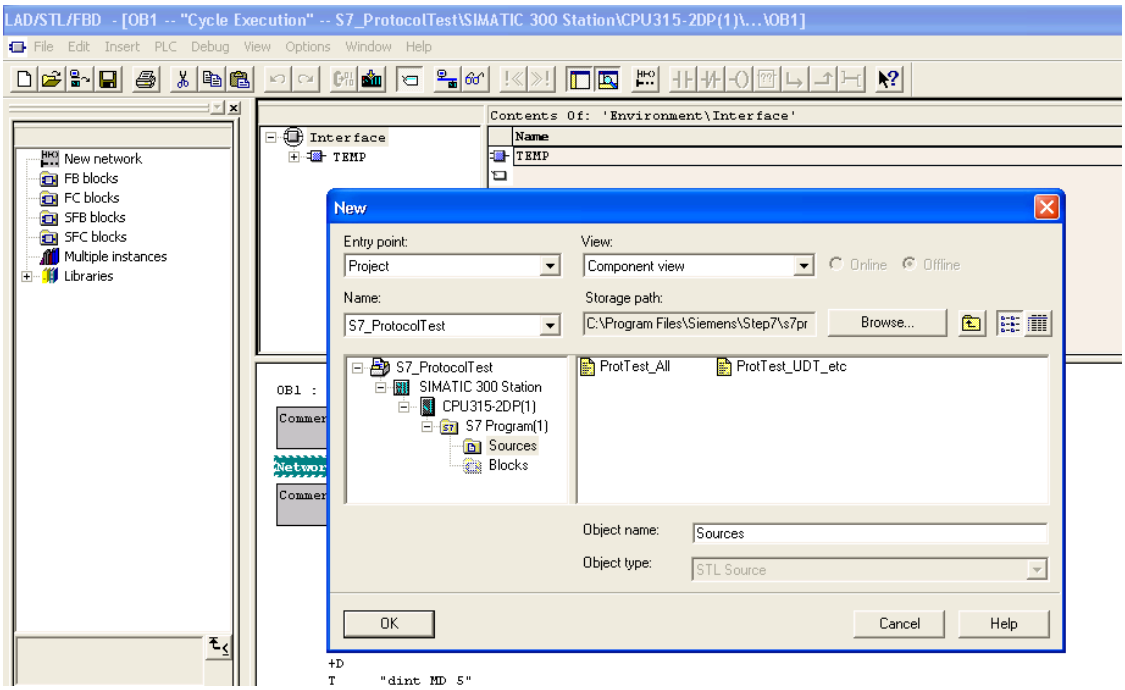
1. From the **Symbol Table** menu in the Symbol Editor choose **Export**.
2. Assign a name and save the symbol table as ASCII file.



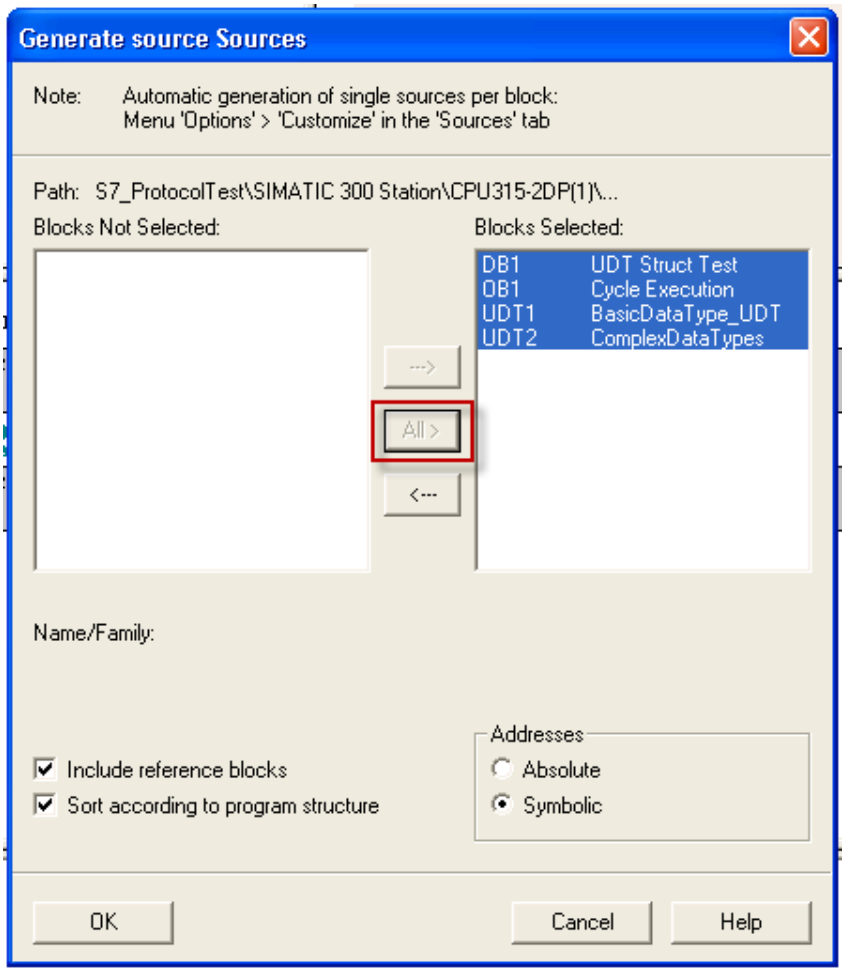
Exporting Sources

These files are created exporting source code.

1. Open any program block in the editor, "OB1" in this example.
2. From the **File** menu choose **Generate Source**: the following dialog is displayed:



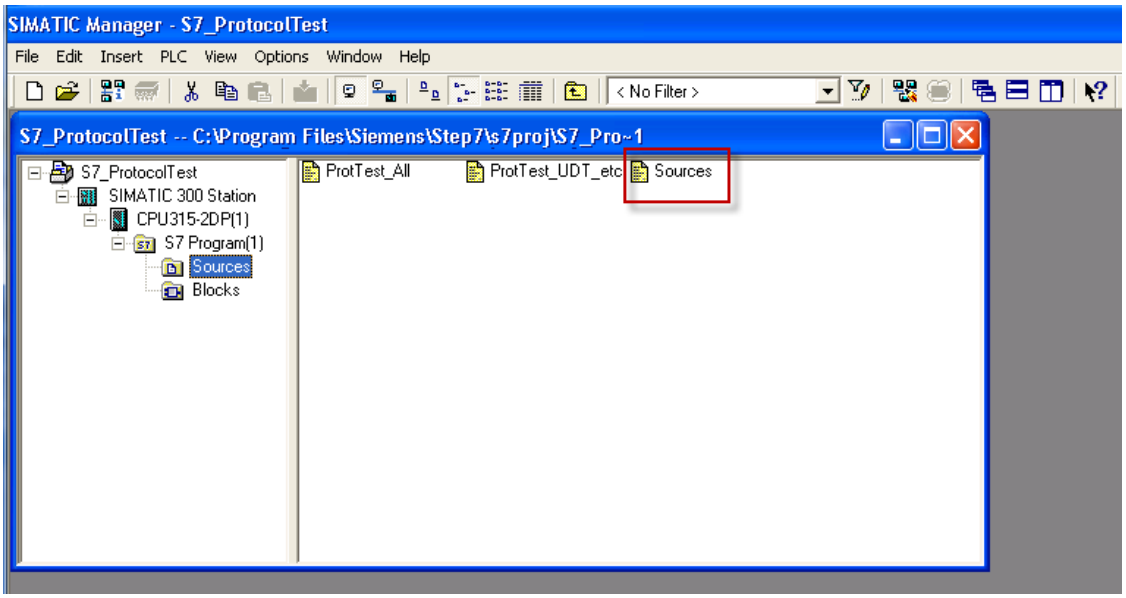
1. Assign a name, "Sources" in the example, and click **OK**: the **Generate source Sources** dialog is displayed.



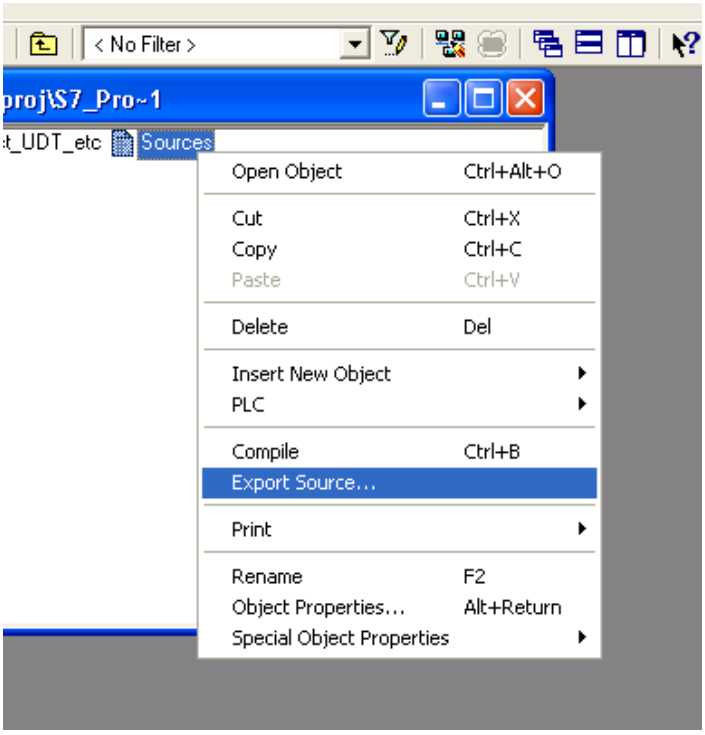
2. Click **All >** to generate source for all blocks.
3. Select the following options:

- **Include reference blocks**
- **Sort according to program structure**
- **Symbolic** address

4. Click **OK** to confirm: the "Sources" object is generated in the Step7 project as in the example.



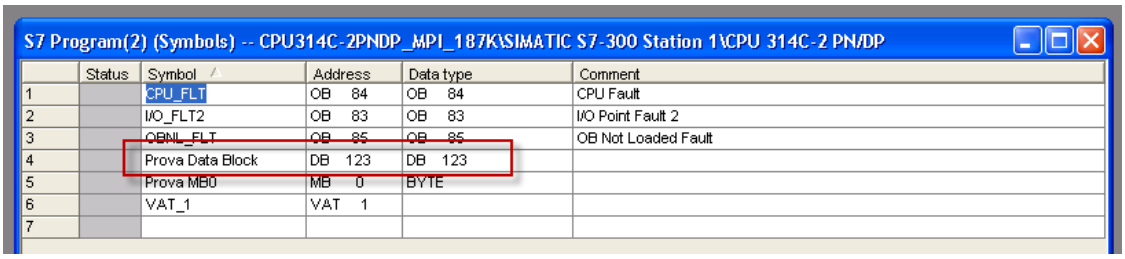
5. Right click on the object and select **Export Sources**.



The generated .awl file can be imported in the Tag Editor.

Note: The .awl file contains additional information not included in the .asc file exported from the symbol table.

Make sure that reference to all data blocks is inserted in the symbol table. The tags from a data block are imported only if the symbol table contains a line with the data block name and related comment.



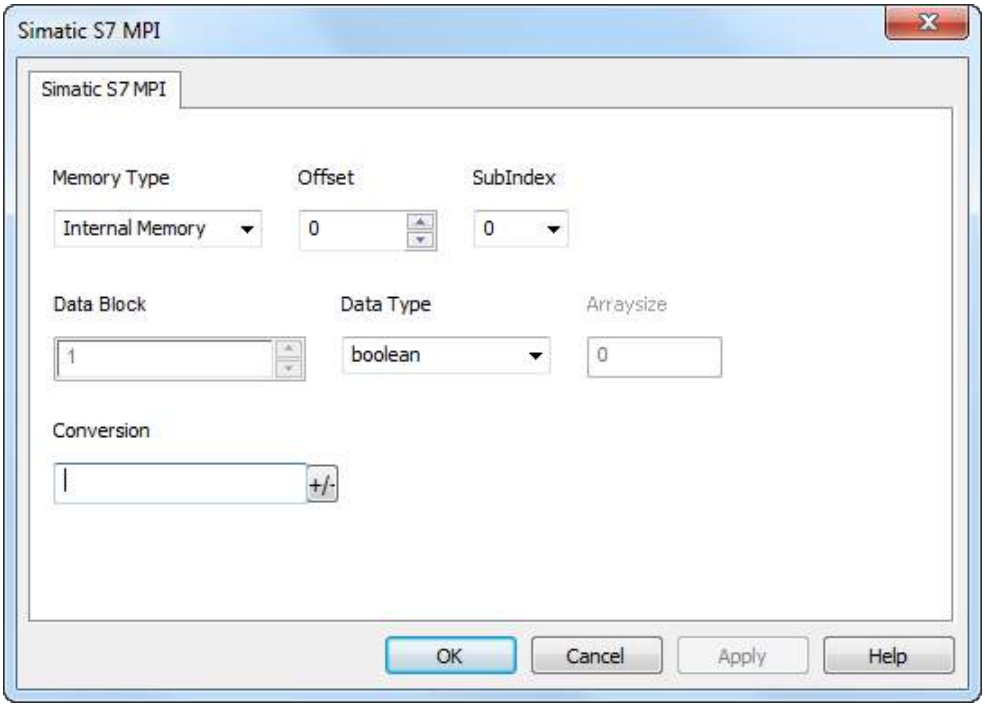
	Status	Symbol	Address	Data type	Comment
1		CPU_FLT	OB 84	OB 84	CPU Fault
2		I/O_FLT2	OB 83	OB 83	I/O Point Fault 2
3		OBNL_FLT	OB 85	OB 85	OB Not Loaded Fault
4		Prova Data Block	DB 123	DB 123	
5		Prova MB0	MB 0	BYTE	
6		VAT_1	VAT 1		
7					

Each entry enables the import filter to import the tags related to the specified data block.

Tag Editor Settings

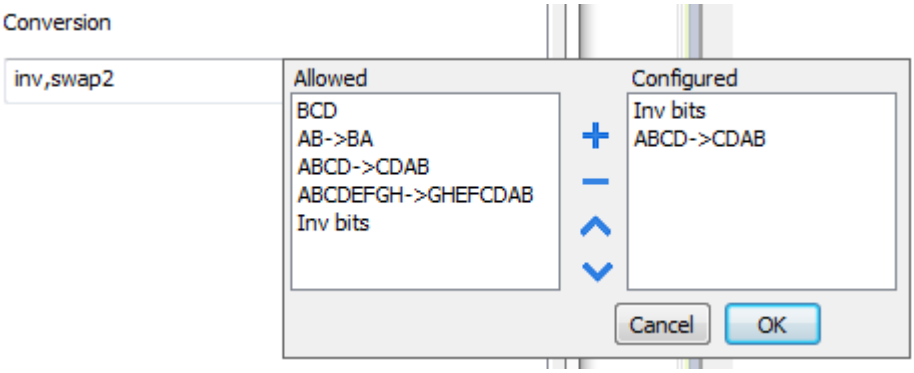
Into Tag editor select the protocol “Simatic S7 MPI” from the list of defined protocols and add a tag using [+] button.

Tag settings can be defined using the following dialog:



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Area of PLC where tag is located.
<table>
<tr><th>Data Type</th><th>Simatic Type</th></tr>
<tr><td>Internal Memory</td><td>M</td></tr>
<tr><td>Data Block</td><td>DB</td></tr>
<tr><td>Input</td><td>I (E)</td></tr>
<tr><td>Output</td><td>O (A)</td></tr>
<tr><td>Timer value</td><td>T</td></tr>
<tr><td>Counter value</td><td>C</td></tr>
</table></td></tr>
<tr><td>Offset</td><td>Offset address where tag is located.</td></tr>
<tr><td>SubIndex</td><td>In case of Boolean data type, this is the offset of single bit.</td></tr>
<tr><td>Data Block</td><td>If Memory Type is “Data Block”, this will identify the DB number.</td></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
<tr><td>string</td><td colspan="2">Refer to “String data type channel”</td></tr>
</table>

Note: to define arrays, select one of Data Type format followed by square brackets like “byte[]”, “short[]”…</td></tr>
<tr><td>Arraysize</td><td>• In case of array tag, this property represents the number of array elements.
• In case of string tag, this property represents the maximum number of bytes available in the string tag.</td></tr>
</table>

Element	Description
	Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor. If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
Conversion	Conversion to be applied to the tag.  Depending on data type selected, the **Allowed** list shows one or more conversions, listed below.

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format)

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>855441236 → 1426062386 (in decimal format)</td></tr>
<tr><td>ABC...NOP -> OPM...DAB</td><td>Swap bytes of a long word.

Example:
142.366 → -893553517.588905 (in decimal format)
0 10000000110
00011100101110110110010001011010000111001010110000001
→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>Separate the byte in two nibbles, and reads them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
<tr><td>S5timer(BCD)</td><td>Used to support S5timer. Check Simatic S5timer special data type for more details.</td></tr>
<tr><td>S5timer(BIN)</td><td>Legacy transformation for S5timer in binary format.</td></tr>
</table>

Select the conversion and click on plus button. The selected item will be added on Configured list.

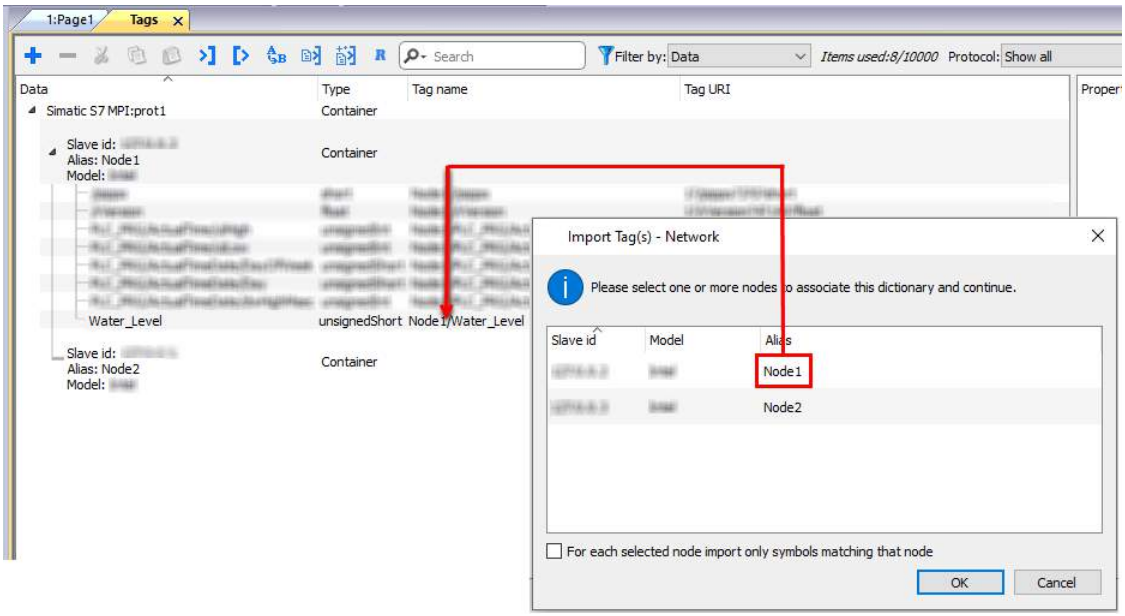
If more conversions are configured, they will be applied in order (from top to bottom of Configured list).

Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

Aliasing Tag Names in Network Configurations

Tag names must be unique at project level; it often happens that the same tag names have to be used for different controller nodes (for example when the HMI is connected to two devices that are running the same application). Since tags include also the identification of the node and Tag Editor does not support duplicate tag names, the import facility in Tag Editor has an aliasing feature that can automatically add a prefix to imported tags. With this feature tag names can be done unique at project level.

The feature works when importing tags for a specific protocol. Each tag name will be prefixed with the string specified by the “Alias”. As shown in the figure below, the connection to a certain controller is assigned the name “Node1”. When tags are imported for this node, all tag names will have the prefix “Node1” making each of them unique at the network/project level.



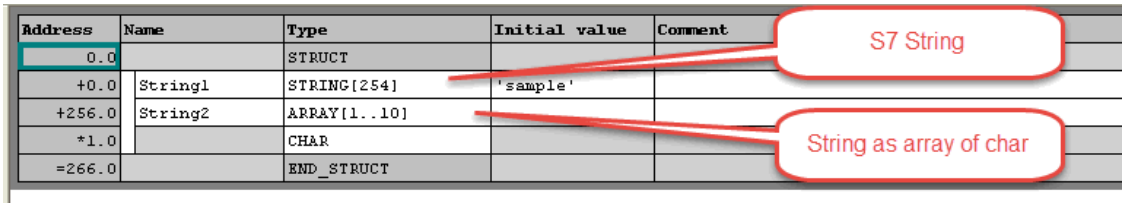
Note: Aliasing tag names are only available when tags can be imported. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.
The Alias string is attached to the tag name only at the moment the tags are imported using Tag Editor. If Alias string is modified after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are imported again, all tags will be imported again with the new prefix string.

String data type

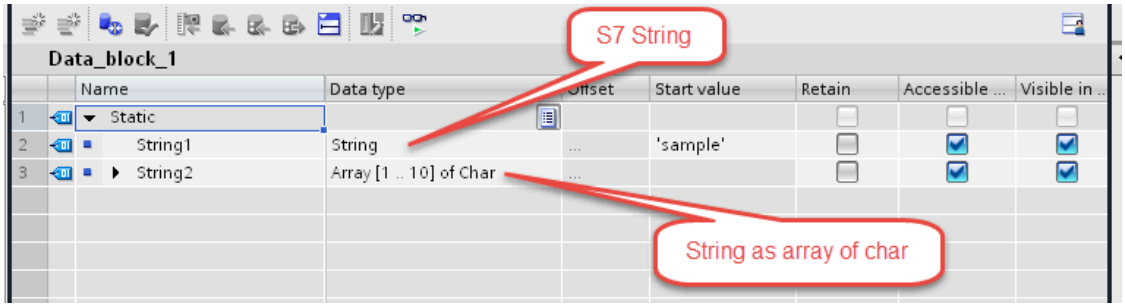
In Simatic S7 PLC it's possible to define two different types of tags to manage string variables.

- as Array [1..xx] of Chars.
- as String[xx].

Step7 string declaration is showed in the following figure:

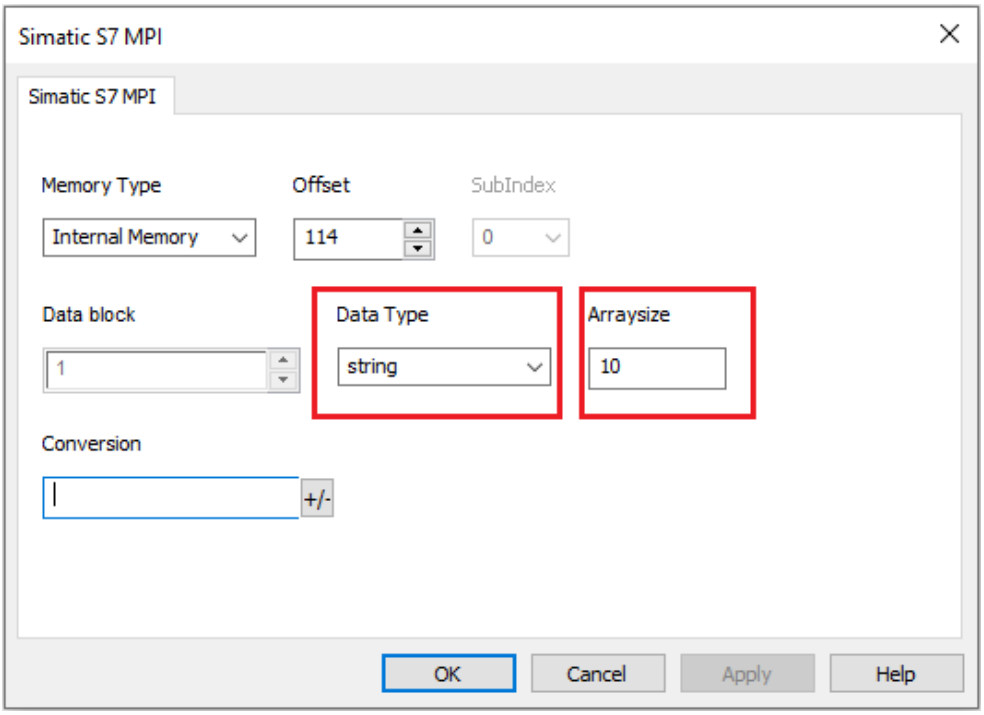


TIA Portal string declaration is showed in the following figure:



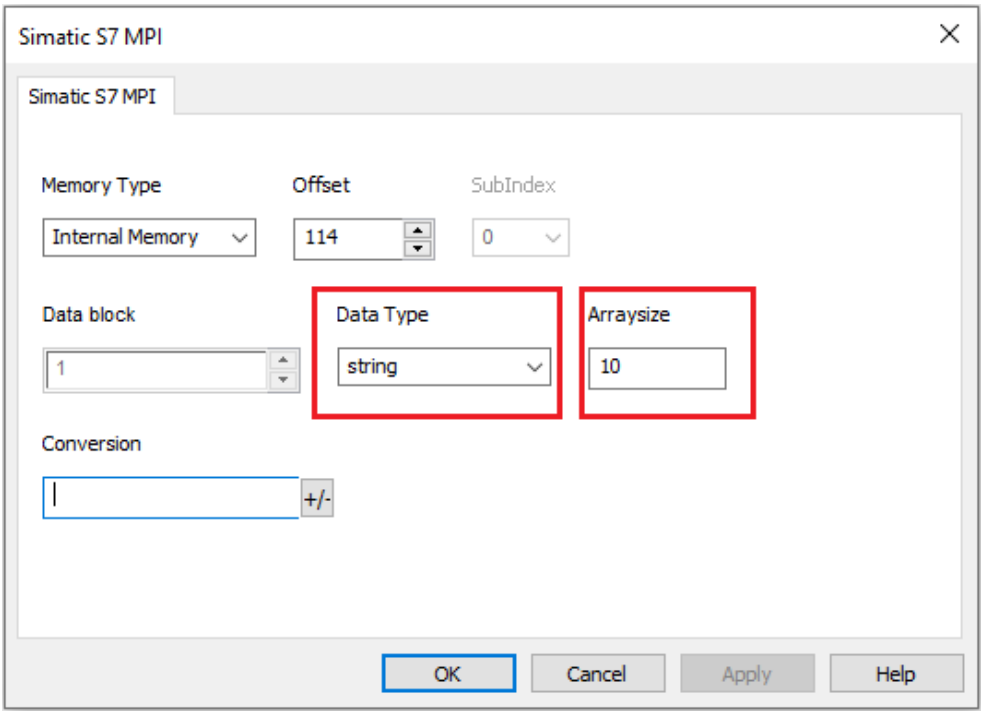
Note: Usage of String[xx] data type is allowed but a specific Conversion must be applied to the tag. Anyway using tag importer to import tag dictionary from TIA Portal or Step7 string tags are automatically configured and no changes/conversion are needed.

To manually add an "Array [1..xx] of Chars" data type tag, press the [+] button in the Tag Editor, then select "string" as Data Type of the Tag and type the string length in the "Arraysize" field:

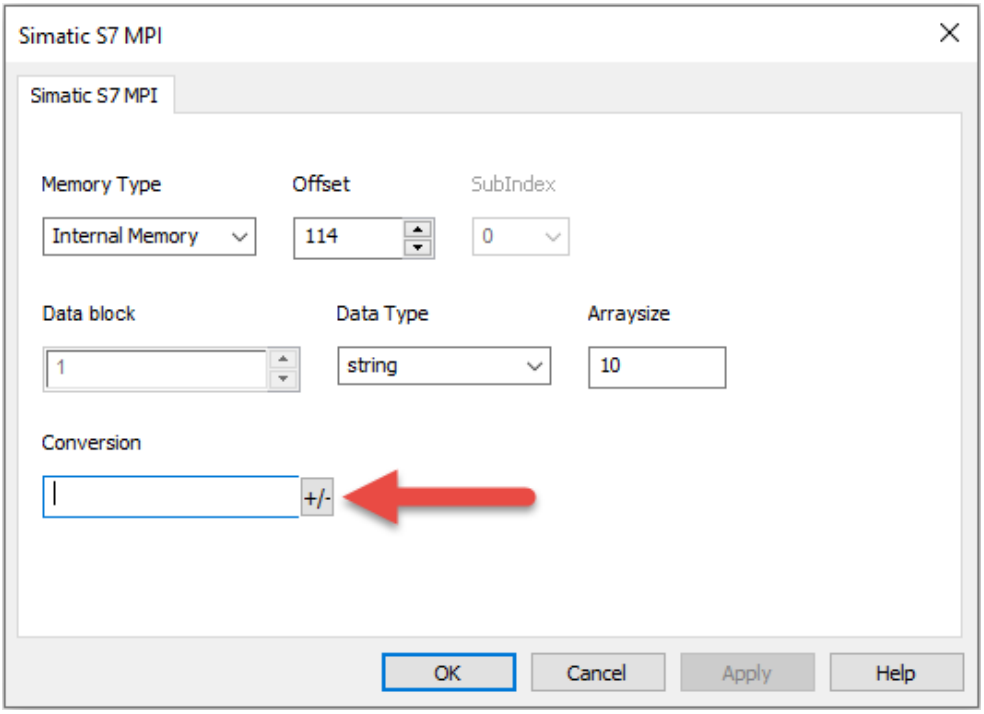


and confirm with OK button.

To manually add a "String[xx]" data type tag, press the [+] button in the Tag Editor, then select "string" as Data Type of the Tag and type the string length in the "Arraysize" field,

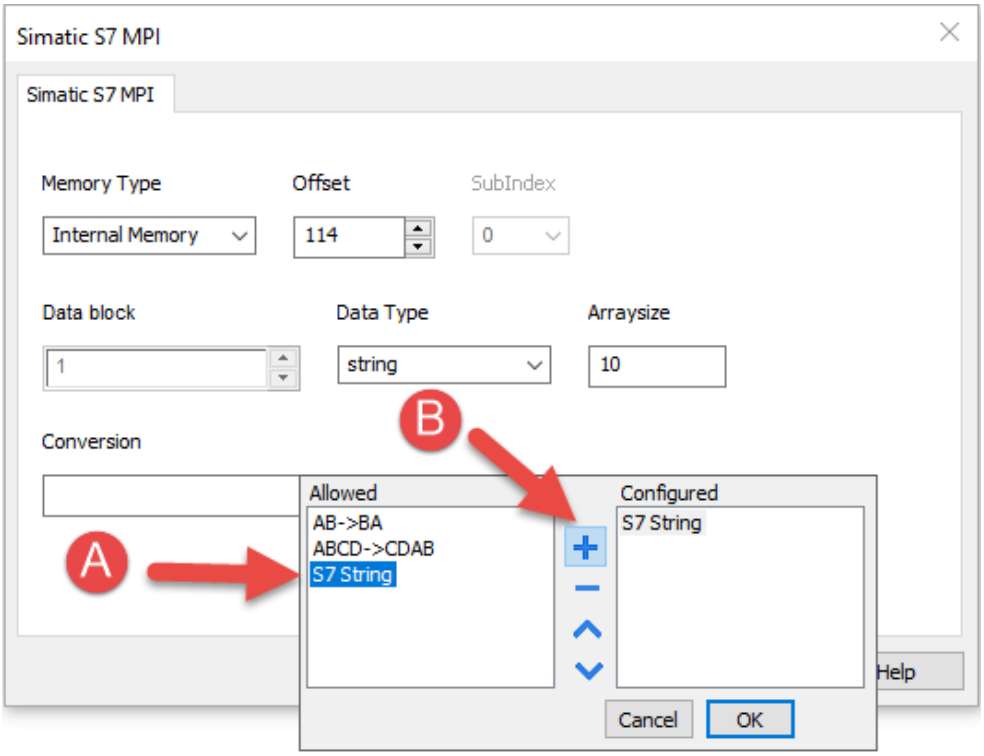


then click on [+/-] button to open the Conversion dialog.



Into conversion dialog:

- select the "S7 String" conversion type
- click on [+] button to add the conversion.



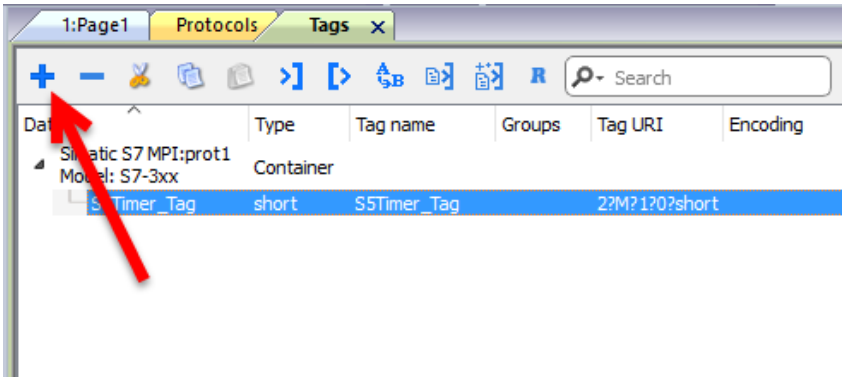
The conversion will be listed into the Configured window on the right.

Confirm with OK button.

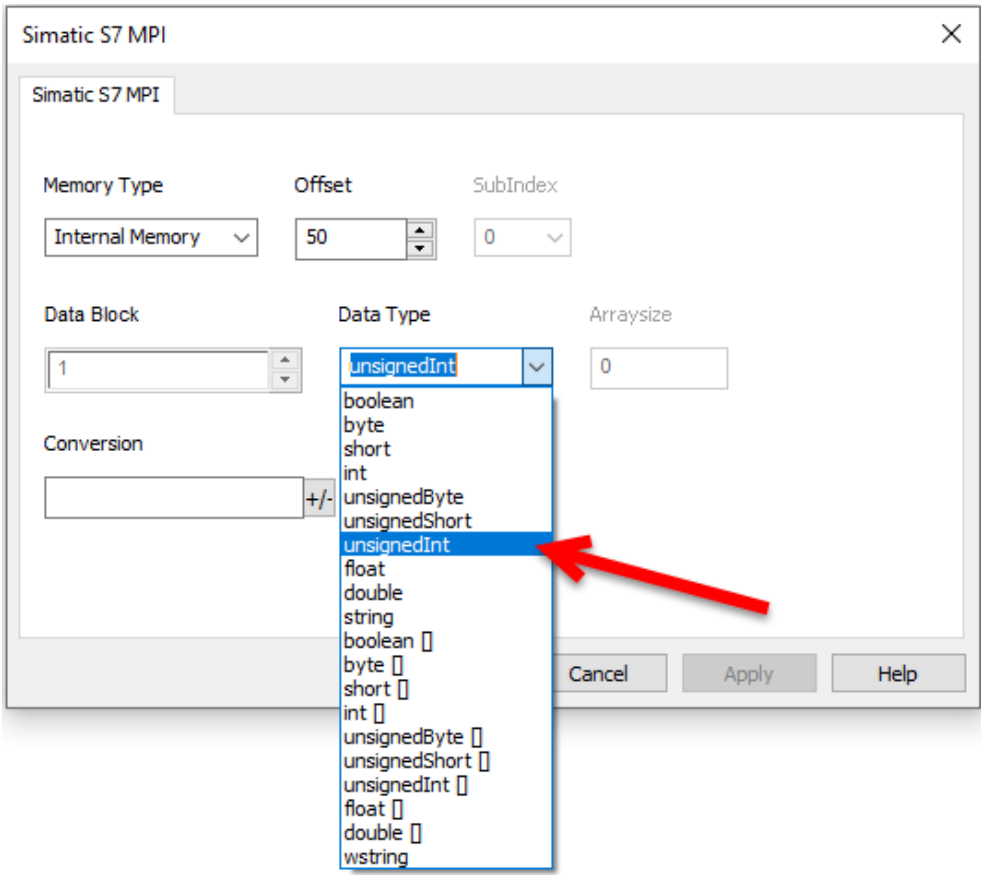
Simatic S5timer data type

Simatic drivers support a special data type, called S5Timer.
The tag must be configured with a specific data type and a conversion must be applied to the Tag to correctly read/write a Simatic S5Timer Variable.

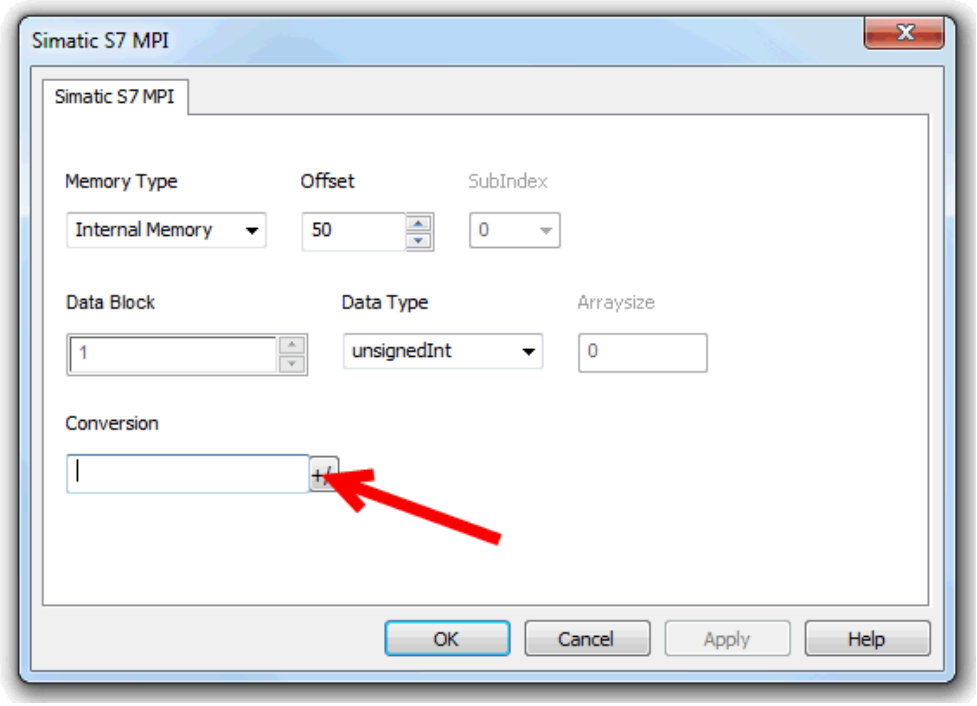
Open the Tag Editor and add a Tag pressing the Plus button.



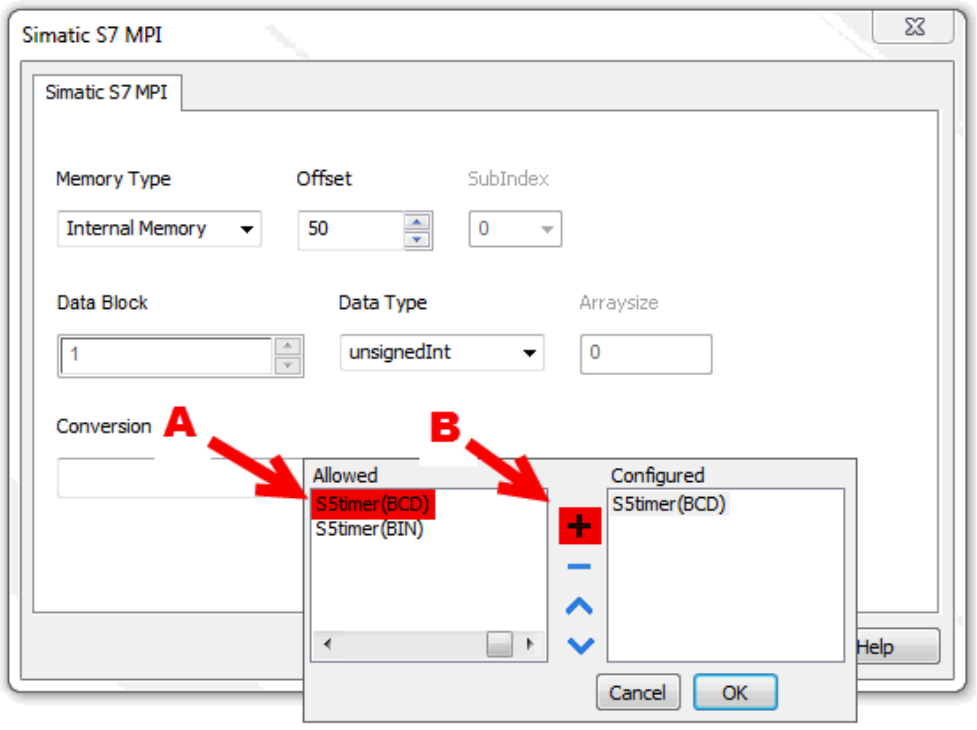
Select “unsignedInt” as Data Type of the Tag.



Click on +/- button to open the Conversion dialog.

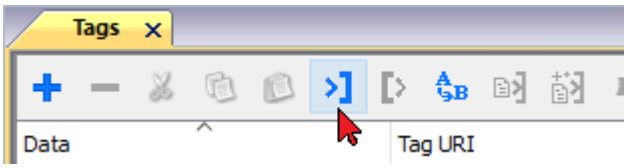


In the Conversion dialog select the S5timer(BCD) conversion type [A] then click on Plus button [B] to add the conversion, the configured conversion will be listed into the Configured window on the right. Then confirm with OK.

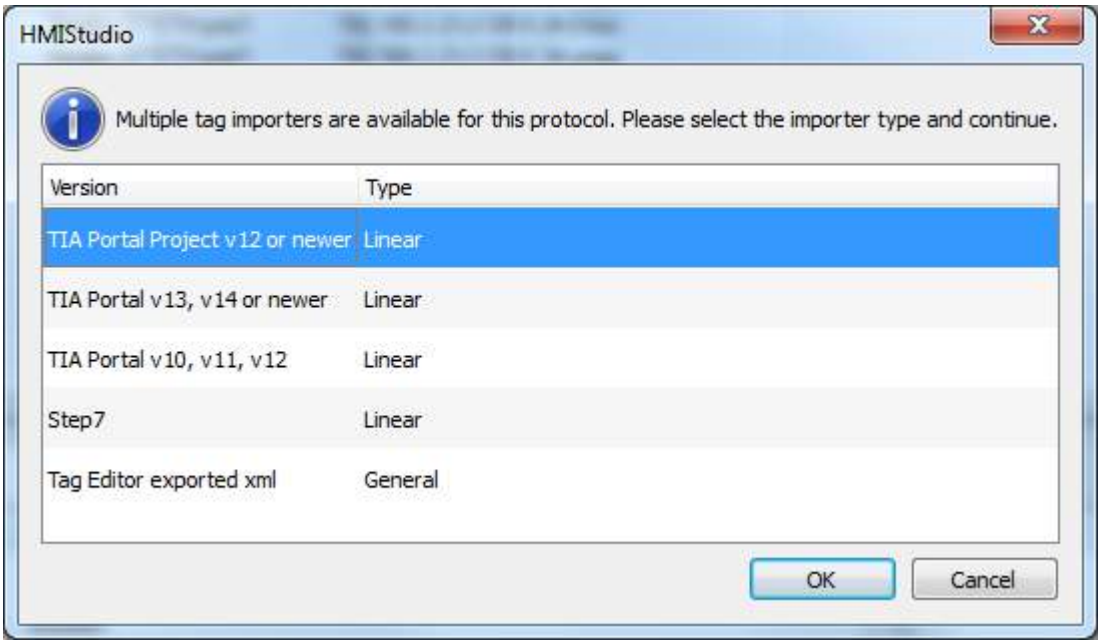


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.



The following dialog shows which importer type can be selected.

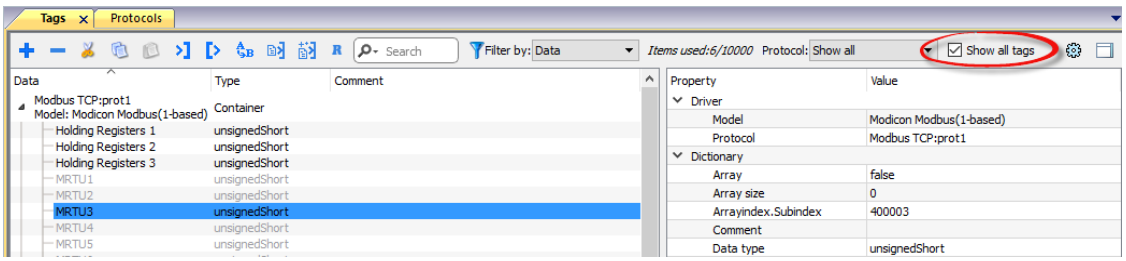


Importer	Description
TIA Portal Project v12 or newer **Linear**	Allows to import the whole TIA Portal project file using **.apxx** file (where "xx" is the TIA Portal version, example: for TIA Portal 13 , file name is "project.ap13"). All variables will be displayed at the same level.
TIA Portal v13, v14 or newer **Linear**	Allows to import: • Program blocks using **.db** file • PLC tags using **.xlsx** file • PLC data types using **.udt** file Check **Export using TIA Portal v13, v14 or newer** for more details. All variables will be displayed at the same level.
TIA Portal v10, v11, v12 **Linear**	Allows to import: • Program blocks using **.tia** file • PLC tags using **.xlsx** file • PLC data types using **.scl** file Check **Export using TIA Portal v10, v11, v12** for more details. All variables will be displayed at the same level.

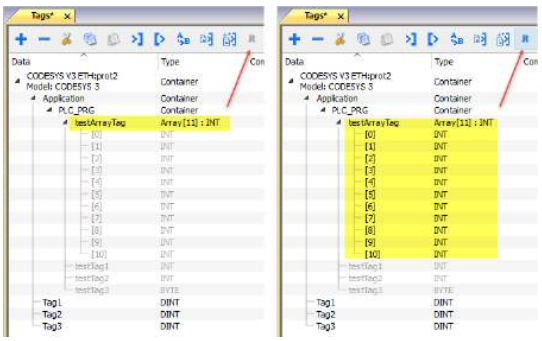
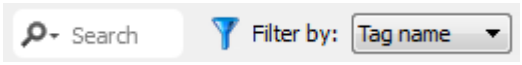
Importer	Description
Step7 Linear	Allows to import: • Symbols table **.asc** file • Sources using **.awl** file Check **Export using STEP7** for more details. All variables will be displayed at the same level.
Tag Editor exported xml	Select this importer to read a generic XML file exported from Tag Editor by appropriate button. 

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result:

Toolbar item	Description
	
	Searches tags in the dictionary basing on filter combo-box item selected.

Communication status

The communication status can be displayed using the dedicated system variables. Please refer to the User Manual for further information about available system variables and their use.

The status codes supported for this communication driver are:

Error	Notes
NAK	Controller replies with a not acknowledge.
Timeout	Request is not replied within the specified timeout period; ensure the controller is connected and properly configured for network access
Invalid response	The device did receive from the controller a response, but its format or its contents or its length is not as expected; ensure the data programmed in the project are consistent with the controller resources.
General Error	Error cannot be identified; should never be reported; contact technical support

System Variables

System Variables communication driver allows to create Tags that point to system information.

Refer to *System Variables > Protocol* chapter of User's Manual.

Protocol Editor Settings

System Variables communication driver allows to create Tags that point to system information.

Refer to *System Variables > Protocol* chapter of User's Manual.

Uni-Telway

Uni-Telway is a field bus used to communicate between devices of the same type according to a protocol defined by Schneider Electric.

The physical access is based on a Serial Link transmission (half-duplex type). The electrical interface allows multi-point mode connection.

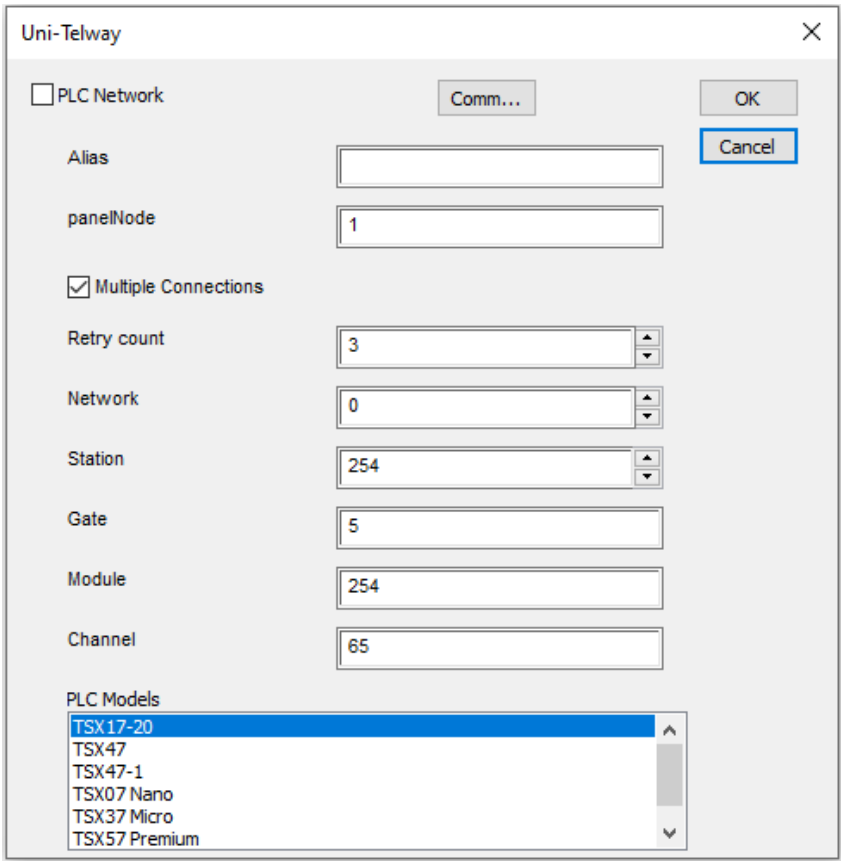
Numerous proprietary or third-party devices can be used on this bus, which has become one of the industry standards.

The operator panels can be connected to a Uni-Telway controller using this communication driver.

Protocol Editor Settings

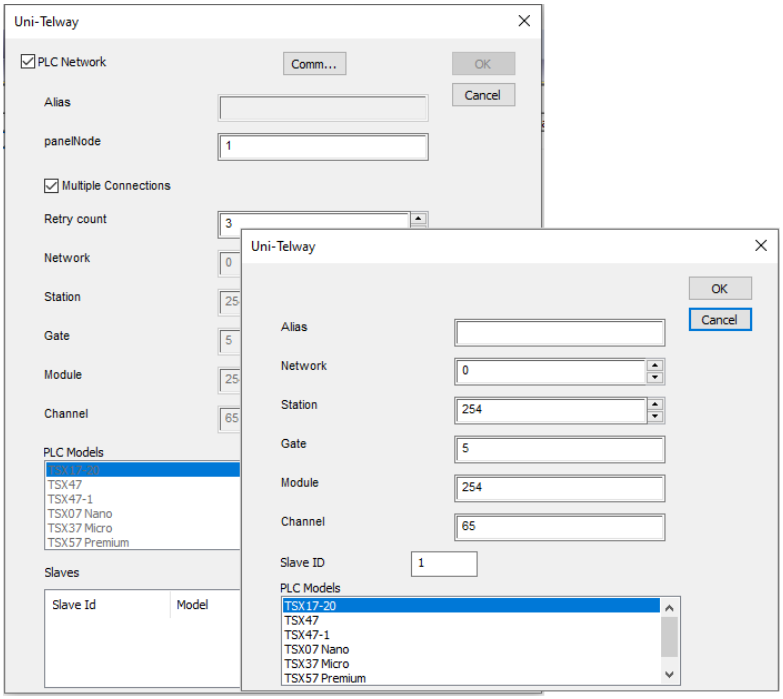
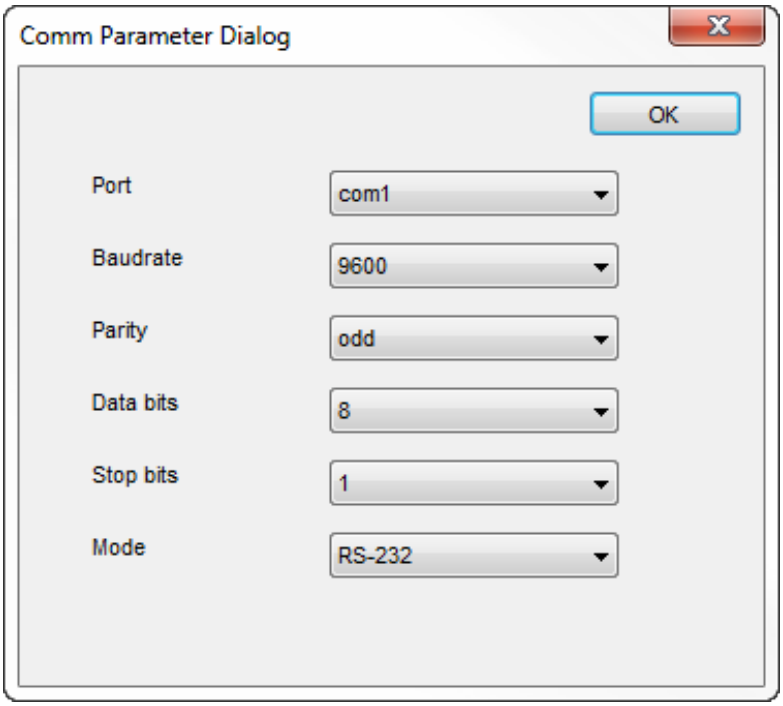
Add (+) a new driver in the Protocol editor and select the protocol called “Uni-Telway” from the list of available protocols.

The driver configuration dialog is shown in figure.



Element	Description
panelNode	Node of the panel into the Uni-Telway network.
Multiple Connections	Not used. Available for future implementation.
Retry count	This parameter defines the number of times a certain message will be sent to the

Element	Description
	controller before reporting the communication error status. A value of 1 for the parameter “Retry count” means that the panel will eventually report the communication error status if the response to the first request packet is not correct.
Network, Station, Gate, Module, Channel	Controller’s parameters as defined into controller’s programming tool.
PLC Models	The driver supports communication with different controllers. Please check directly in the programming IDE software for a complete list of supported controllers.

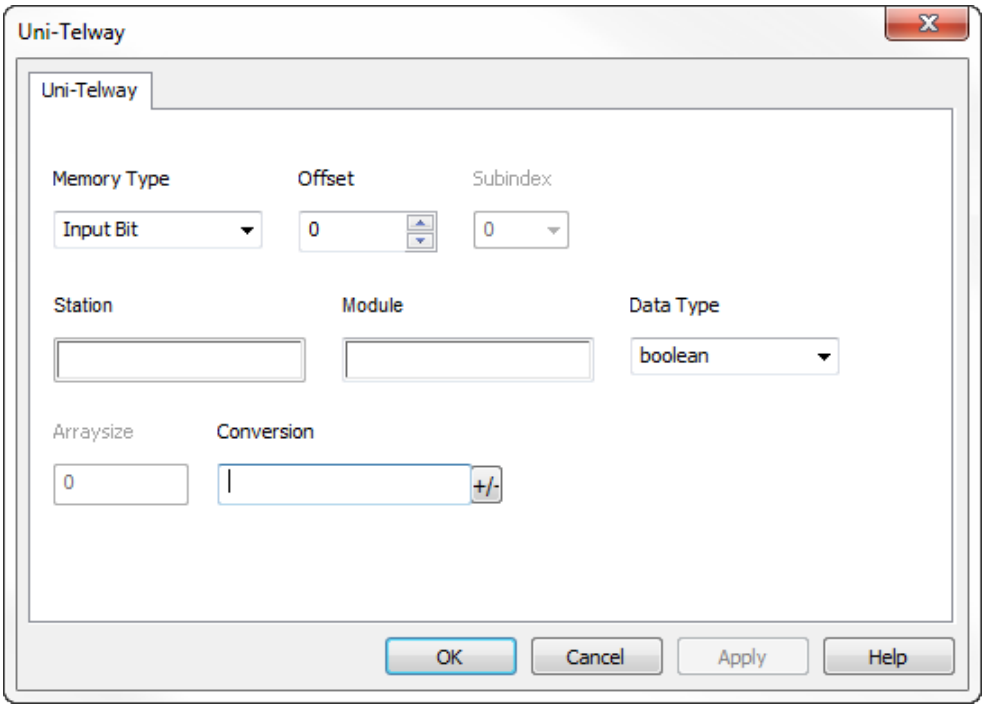
Element	Description
PLC Network	The protocol allows the connection of multiple controllers to one operator panel. To set-up multiple connections, check “PLC network” checkbox and configure all controllers. 
Comm…	Gives access to the serial port configuration parameters as shown in the following figure: 

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Port</td><td>Serial port selection:
<table>
<tr><th></th><th>Series 400</th><th>Series 500/600</th></tr>
<tr><td>com1</td><td>PLC Port</td><td>Onboard Serial Port</td></tr>
<tr><td>com2</td><td>PC/Printer Port</td><td>Optional Module
on slot #1 or #2</td></tr>
<tr><td>com3</td><td>Not available</td><td>Optional Module
on slot #3 or #4</td></tr>
</table>
</td></tr>
<tr><td>Baud rate, Parity, Data bits, Stop bits</td><td>Communication parameters for serial communication</td></tr>
<tr><td>Mode</td><td>Serial port mode; available options:
RS-232,
RS-485 (2 wires)
RS-422 (4 wires)</td></tr>
</table>
</td></tr>
</table>

Tag Editor Settings

Into Tag editor select the protocol “Uni-Telway” from the list of defined protocols and add a tag using [+] button.

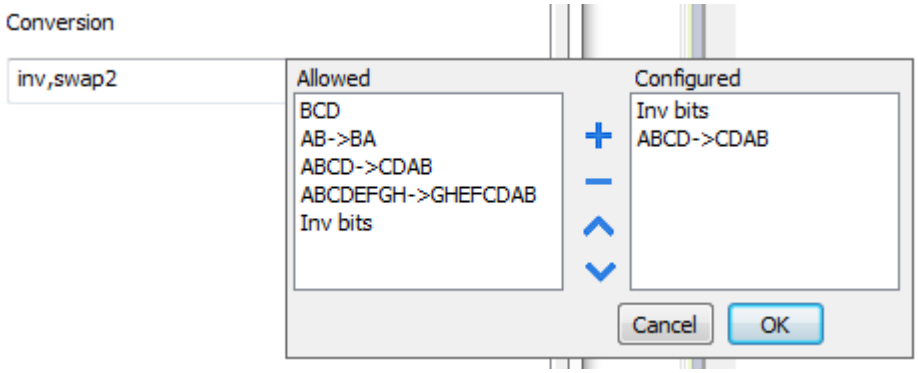
Tag settings can be defined using the following dialog:



Element	Description
Memory Type	Memory resource where tag is located.
Offset	Offset address where tag is located.
SubIndex	This allows resource offset selection within the register.
Station	Station number. Property available only for Memory Type “Common Word”.
Module	Module number. Property available for Memory Type: • Input Bit • Output Bit • Input Word • Output Word

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1 bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>float</td><td>IEEE single-precision
32-bit floating point type</td><td>1.17e-38 ... 3.40e38</td></tr>
</table>
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>
Conversion to be applied to the tag.


Depending on data type selected, the list Allowed shows one or more conversion types.
</td></tr>
</table>

Element	Description

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011000001 → 1 10000011100 1010101000010100010111011011011001011011000010011101 (in binary format)
BCD	**bcd**: Separate byte in two nibbles, read them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Element	Description
	Select conversion and click +. The selected item will be added to list **Configured**. If more conversions are configured, they will be applied in order (from top to bottom of list **Configured**). Use the arrow buttons to order the configured conversions.

Variables

Variables communication driver allows to define Tags which points to HMI internal memory.

Variables Tags are not retentive: when the project starts, the starting value of any Variables Tag is 0 (or "" in case of string Tag).

Variables communication driver is not counted as physical protocol.
Refer to **Table of functions and limits** from main manual in "Number of physical protocols" line.

Protocol Editor Settings

Adding a protocol

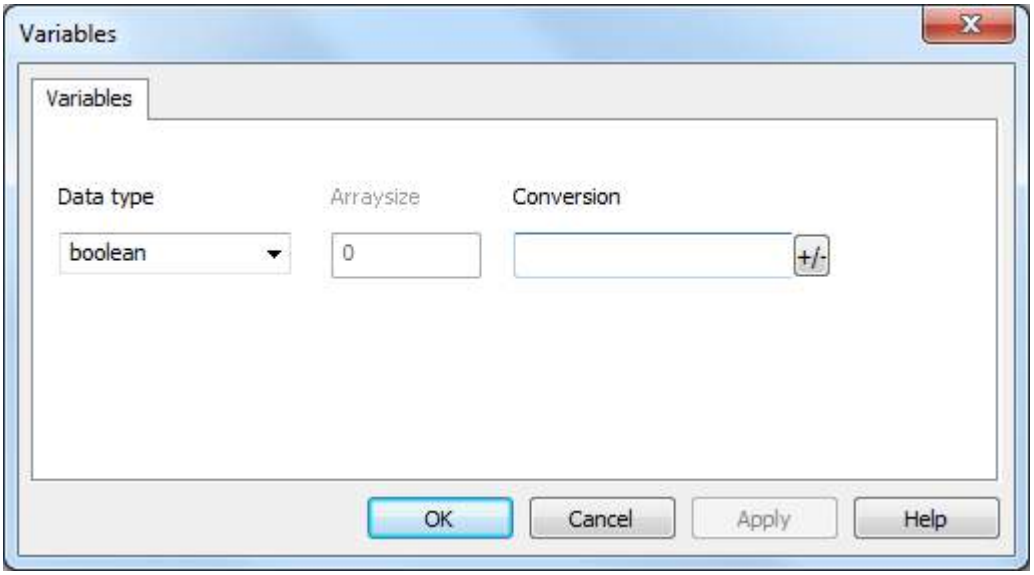
To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the **Variables** protocol from the **PLC** list.

Tag Editor Settings

*Path: **ProjectView**> **Config** > double-click **Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **Variables** from the protocol list: tag definition dialog is displayed.

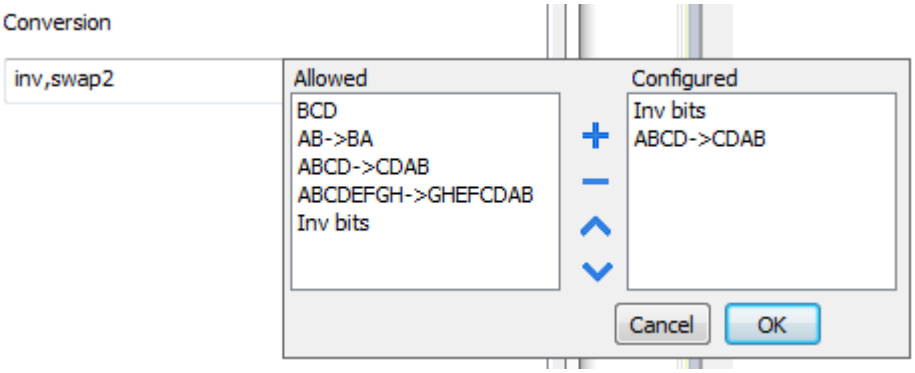


<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array tag, this property represents the number of array elements.
In case of string tag, this property represents the maximum number of bytes available in the string tag.

Note: number of bytes corresponds to number of string characters if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one character requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the tag.</td></tr>
</table>

Element	Description
	 Depending on data type selected, the list **Allowed** shows one or more conversion types.

Value	Description
Inv bits	**inv**: Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	**neg**: Set the opposite of tag value. *Example:* 25.36 → -25.36
AB -> BA	**swapnibbles**: Swap nibbles in a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	**swap2**: Swap bytes in a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	**swap4**: Swap bytes in a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	**swap8**: Swap bytes in a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 0001110010111011011001000101101000011100101011 00

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>0001
→
1 10000011100
10101010000101000101101101101100101101100001001111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>bcd: Separate byte in two nibbles, read them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>

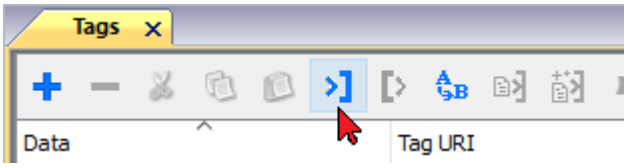
Select conversion and click +. The selected item will be added to list Configured.

If more conversions are configured, they will be applied in order (from top to bottom of list Configured).

Use the arrow buttons to order the configured conversions.</td></tr>
</table>

Tag Import

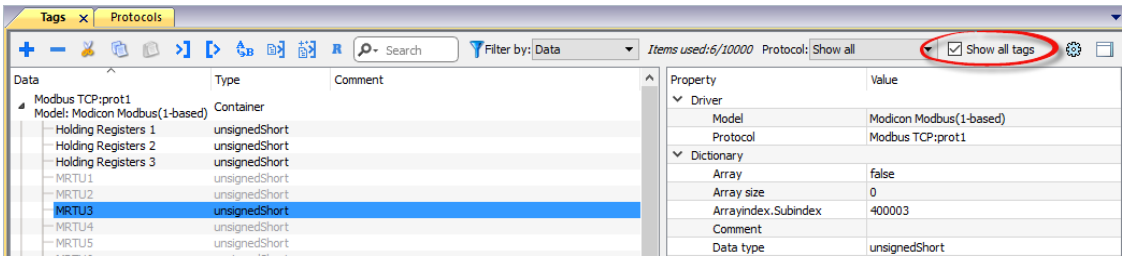
Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.

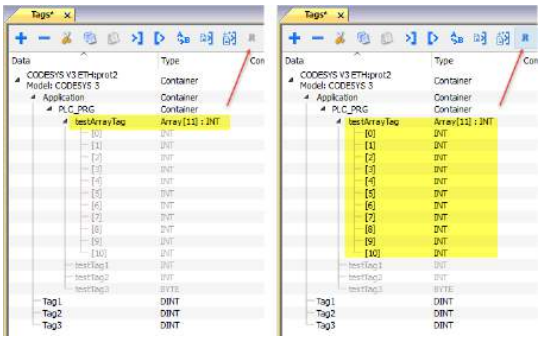
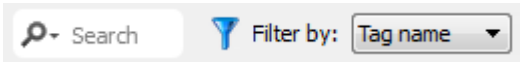


The system will require a generic XML file exported from Tag Editor by appropriate button.

Once the importer has been selected, locate the symbol file and click **Open**.

The tags available within the Dictionary but not imported into the project are gray and are visible only when the "Show all tags" check box is selected.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.
	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
	Searches tags in the dictionary basing on filter combo-box item selected.

Exor International S.p.A.
Via Monte Fiorino, 9
37057 San Giovanni Lupatoto (Verona)
Italy

info@exorint.com
phone: +39 045 8750404
fax: +39 045 8779023

Communication Protocols
User Manual

2022-05-18

Copyright © 2010-2022