

## Application Note



## 2500 Series® Programmable Automation Control System

How to convert a Peer-to-Peer network from  
Simatic/TI PeerLink modules using RS485 to  
CTI JACP modules using Ethernet



# Description of Peerlink Setup and Operation

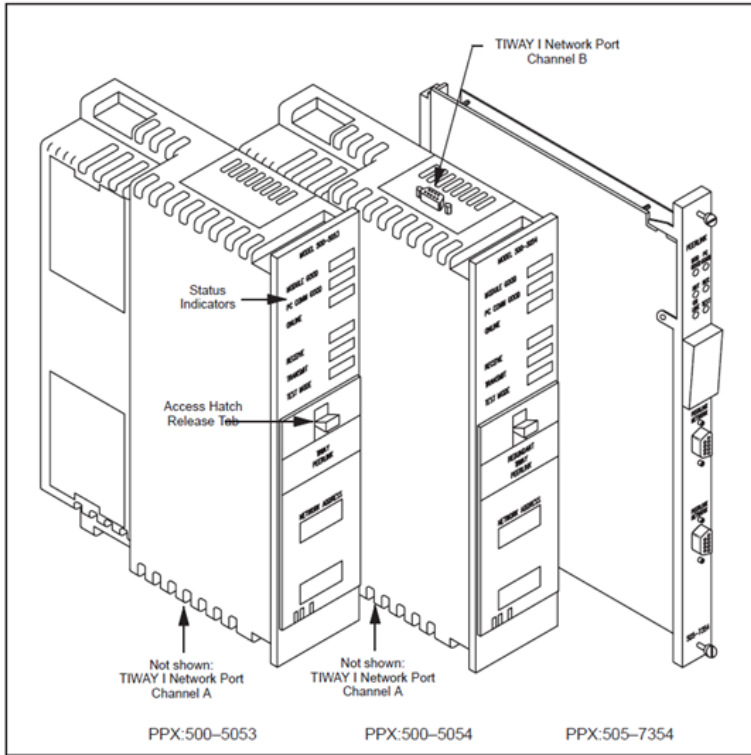


Figure 1-1 Peerlink Modules

This information is taken directly from the 'SIMATIC TIWAY 1 Peerlink Manual'

In Peerlink terminology, the term "station" refers to a PLC rack with a Peerlink module that is assigned a specific network address. Each station on a Peerlink network is capable of transmitting up to 16 words of data (16-bit words) to every other station on the network. This means that each station can receive up to 240 words of data if a system with a maximum load of 16 stations (each transmitting 16 words) is used.

Peerlink operates by using a broadcast method of data transmission, where one module is designated as the "active monitor." This means that this module is responsible for initiating all network communications. The active monitor polls each station on the network and each station responds by broadcasting its message on the network.

The basic network operation is the granting of a time slot to a station followed by a broadcast of data by

the station. The information frame sent by a network station contains a message field consisting of up to 16 words of data that is broadcast to all stations connected to the network.

For each Peerlink station on the network, there must be 16 words of V-Memory allocated in each PLC. For example, if you have two Peerlink stations, each PLC must have 32 words of contiguous V Memory space reserved. If you have 16 Peerlink stations, each PLC must have 256 words reserved. You specify the location of this V Memory space by using normal I/O output words 4 and 5 which specify the starting address of the Peerlink data table.

Modules on a Peerlink network are interconnected on a RS485 serial bus "multidrop" line (Local Line) consisting of a shielded twisted pair cable. The total cable length cannot exceed 10,000 feet using premium cable such as Belden) 9860. Figure 2-1 illustrates a typical multidrop bus configuration.

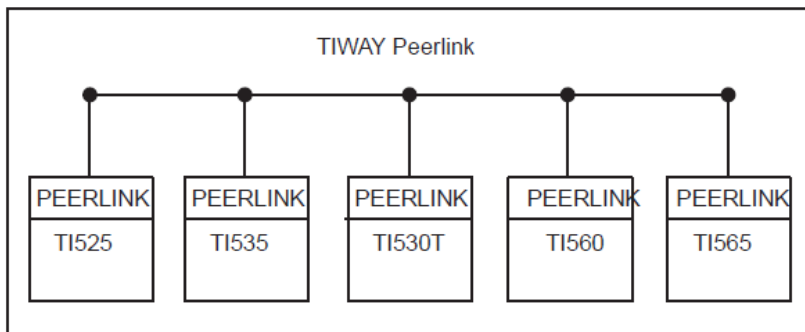


Figure 2-1 TIWAY Peerlink Multidrop Bus Configuration



For using the network data, the number of words allocated should equal 16 times the highest addressed station number on the network. The only thing that Peerlink requires is that you specify the starting address of this table through normal I/O. When you have done this, Peerlink will automatically begin logging in data. The first 16 words are designated for network address #1, the next 16 words for network address #2, and so on. The station designated as network address #1 would transmit in the first block of 16 words. It would receive data from the rest of the network in the remaining blocks. The station designated as network address #2 would transmit data in the second block of 16 words, and so on.

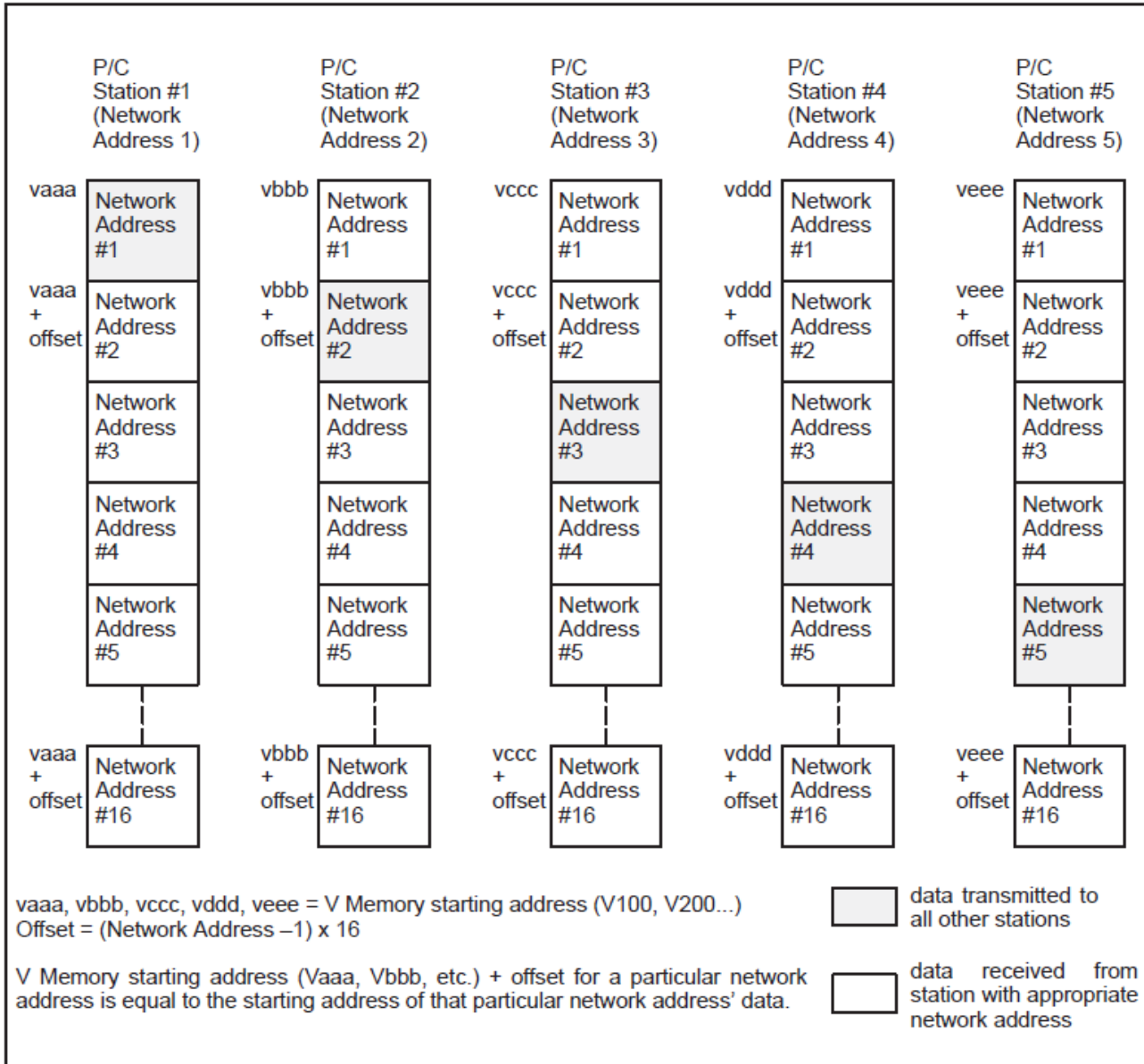
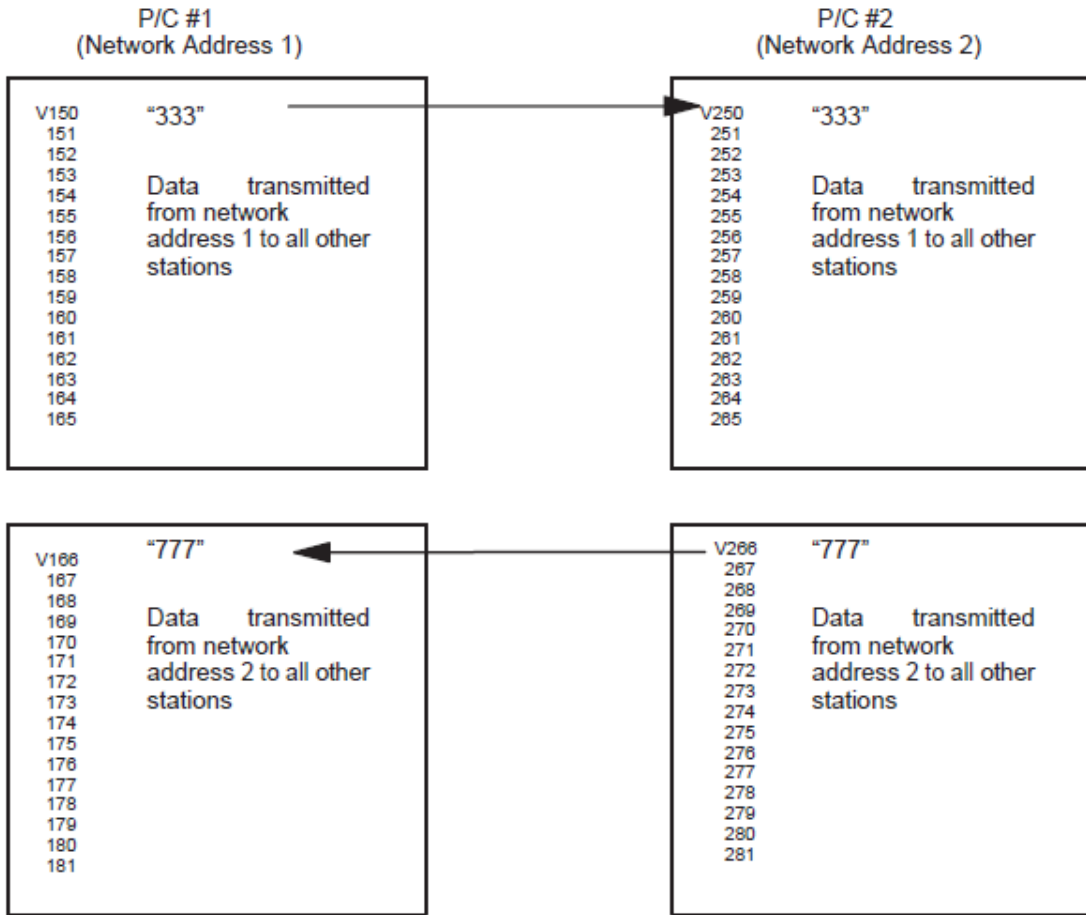


Figure 2-4 P/C V Memory Tables Allocated for Peerlink Network Data

**Peerlink V-Memory setup for two stations:** For each PLC on the network, you need to specify a sequential block of V memory words equal to 16 times the highest network address. For this example, in which the highest network address is 2, you need to allocate 32 words of V memory in each PLC. If you defined the V memory table starting address for station 1 as V150 in PLC #1, and for station 2 as V250 in PLC #2, the chart below would illustrate the organization of the V memory tables. For example, if you wrote the value 333 to V150 in PLC #1, you would see 333 appear in V250 in PLC #2. If you wrote the value 777 to V266 in P/C #2, you would see 777 appear in V166 in PLC #1.



## Using 2500P-JACP Janus Application Coprocessor to Replace Peerlink

This sample application involves using two separate PLC bases, with each containing a Simatic® 505 PLC and a 2500P-JACP module. The Ethernet ports on the JACP modules are plugged into an ethernet switch.

- Janus WorkBench programming software will be used to develop and test the JACP software applications.
- 505WorkShop programming software will be used to test the operation of the two JACP modules functionality by reading and writing V-memory registers.
- No programs are required to be created for either PLC.
- The JACP modules only require configuration of the data communications.

The following JACP Ethernet Data Communications Protocols will be used:

- Block Transfer (enables communications with Series 505 and CTI controllers using the Special Function I/O protocol) – *this is how the JACP module communicates to the PLC through the base backplane.* The ‘Block Transfer’ I/O Driver will be used to read and write V-Memory in a Simatic/TI 555 PLC and a CTI C400 PLC. These V-Memory registers will show up as symbols in the JACP applications.
- Network Data Exchange (binding) – *this is how the JACP modules communicate with each other.* The ‘Network Data Exchange’ Publisher and Subscriber communication method will be used to exchange data between the two JACP applications. This data exchange is achieved by using ‘Binding’ to link symbols in one JACP application into the other JACP application.

### Prerequisites

- JACP modules are installed, configured and operating in each PLC base.
- Janus Workbench programming software is installed and operational on the computer to be used.
- The appropriate knowledge of the Workbench software exists in order to configure / modify and test these applications.
- 505Workshop programming software is installed and operational on the computer to be used.
- The appropriate knowledge of the 505WorkShop software exists in order to test/verify JACP data exchange.
- Each JACP module and the programming computer are configured appropriately and are on the same ethernet network.

From the *Simatic TIWAY 1 PEERLINK* user manual, the network application example given in Appendix B.2.6 will be replicated using JACP hardware. In that example:

- PLC1 writes the contents of registers V151-V165 into PLC2 registers V250-V265
- PLC2 writes the contents of registers V266-V281 into PLC1 registers V166-V181

### Building the Workbench Projects for each JACP Module

1. Create a new Project List in Workbench – in this case it is named ‘PeerLink\_conversion\_to\_JACP’.
2. Create each of the JACP projects – in this case they are named:

PLC1\_JACP1 (PLC1\_JACP1 is assigned IP address 172.18.9.230)

PLC2\_JACP2 (PLC2-JACP2 is assigned IP address 172.18.9.231)





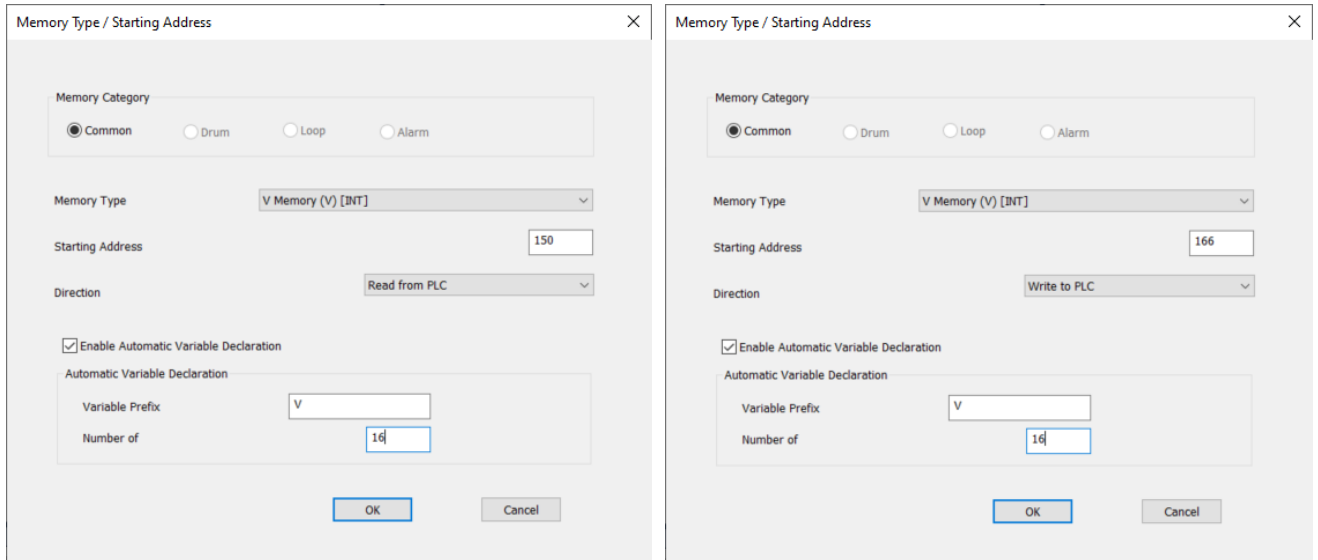
3. Next, we'll put a configuration in the PLC1\_JACP1 project to allow the module to read and write V-Memory in PLC1. In the Fieldbus Configuration insert the 'Block Transfer' I/O driver into this project.
4. Now configure the V-Memory to be used:

For PLC1\_JACP1,

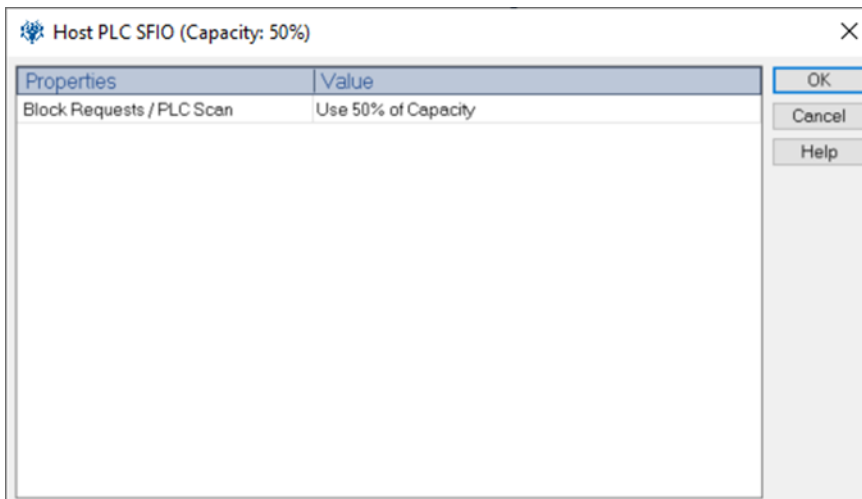
Read from PLC = V150 starting address; configure for 16 registers (V150 – V165)

Write to PLC = V166 starting address; configure for 16 registers (V166 - V181)

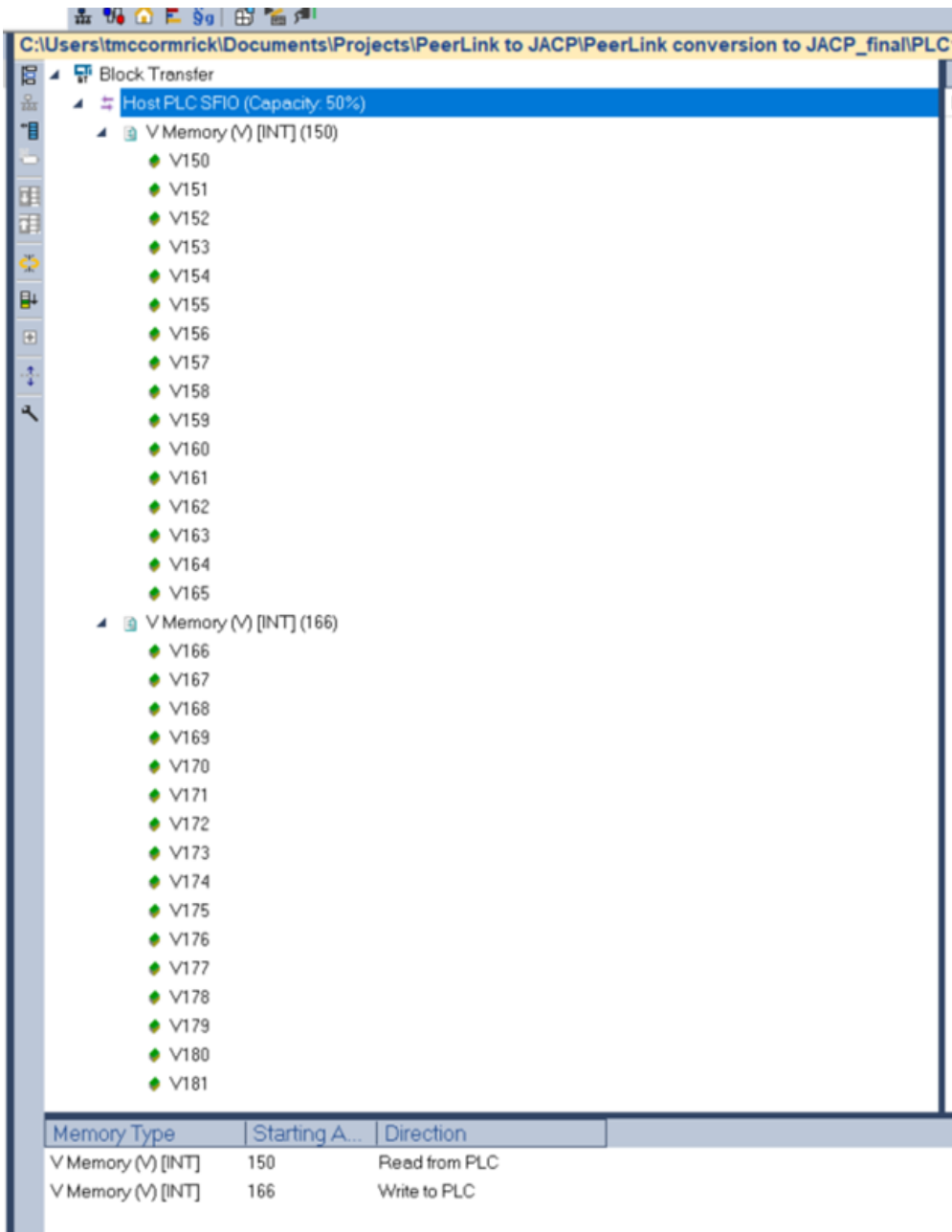
The configurations should look like this:



Also, the Host PLC SFIO capacity needs to be modified for this particular application because the Simatic/TI 555 PLC only supports 8 SFIO task codes per scan, the capacity must be set to 50%. If both PLC's were CTI 2500 series then this parameter can be left at 100%.



When complete, the JACP1 'Block Transfer' window should look like this:



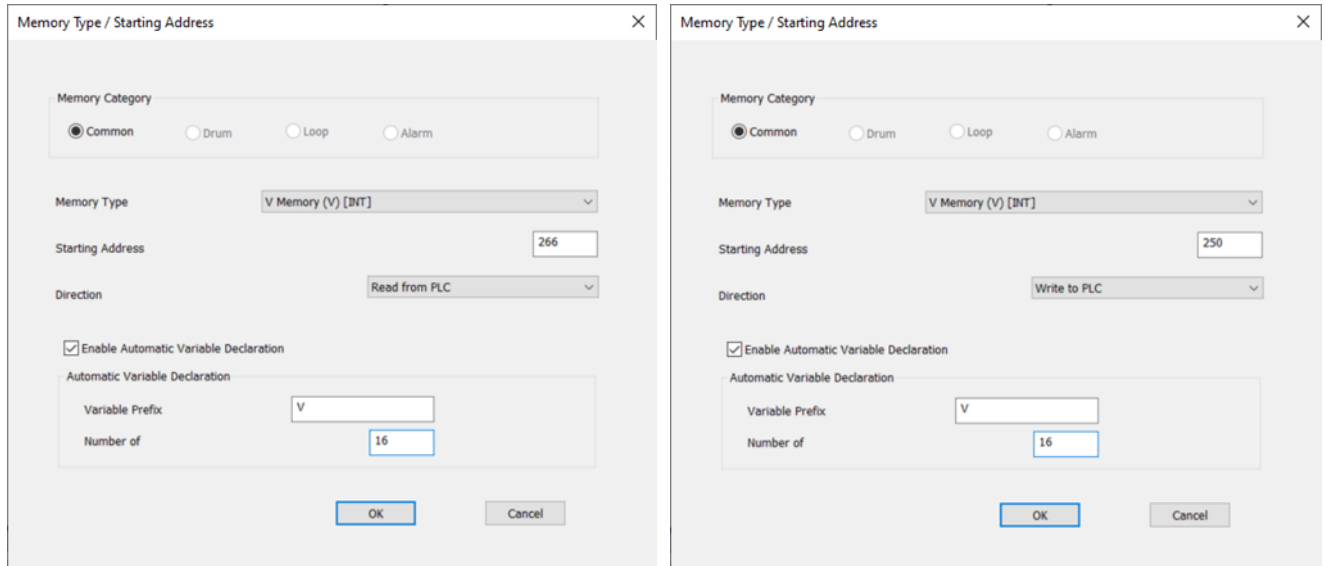
5. Next, we'll put a configuration in the PLC2\_JACP2 project to allow the module to read and write V-Memory in PLC2. In the Fieldbus Configuration insert the 'Block Transfer' I/O driver into this project.
6. Now configure the V-Memory to be used:

For PLC2\_JACP2,

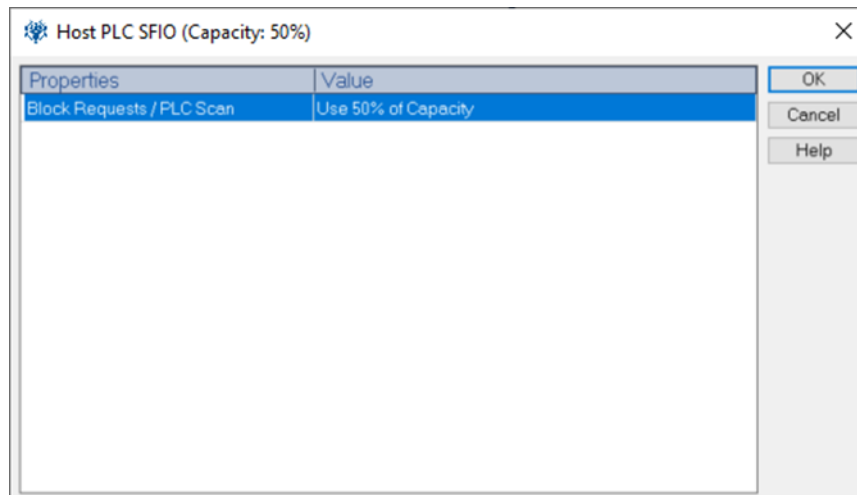
Read from PLC = V266 starting address; configure for 16 registers (V266 – V291)

Write to PLC = V250 starting address; configure for 16 registers (V250 - V265)

The configurations should look like this:

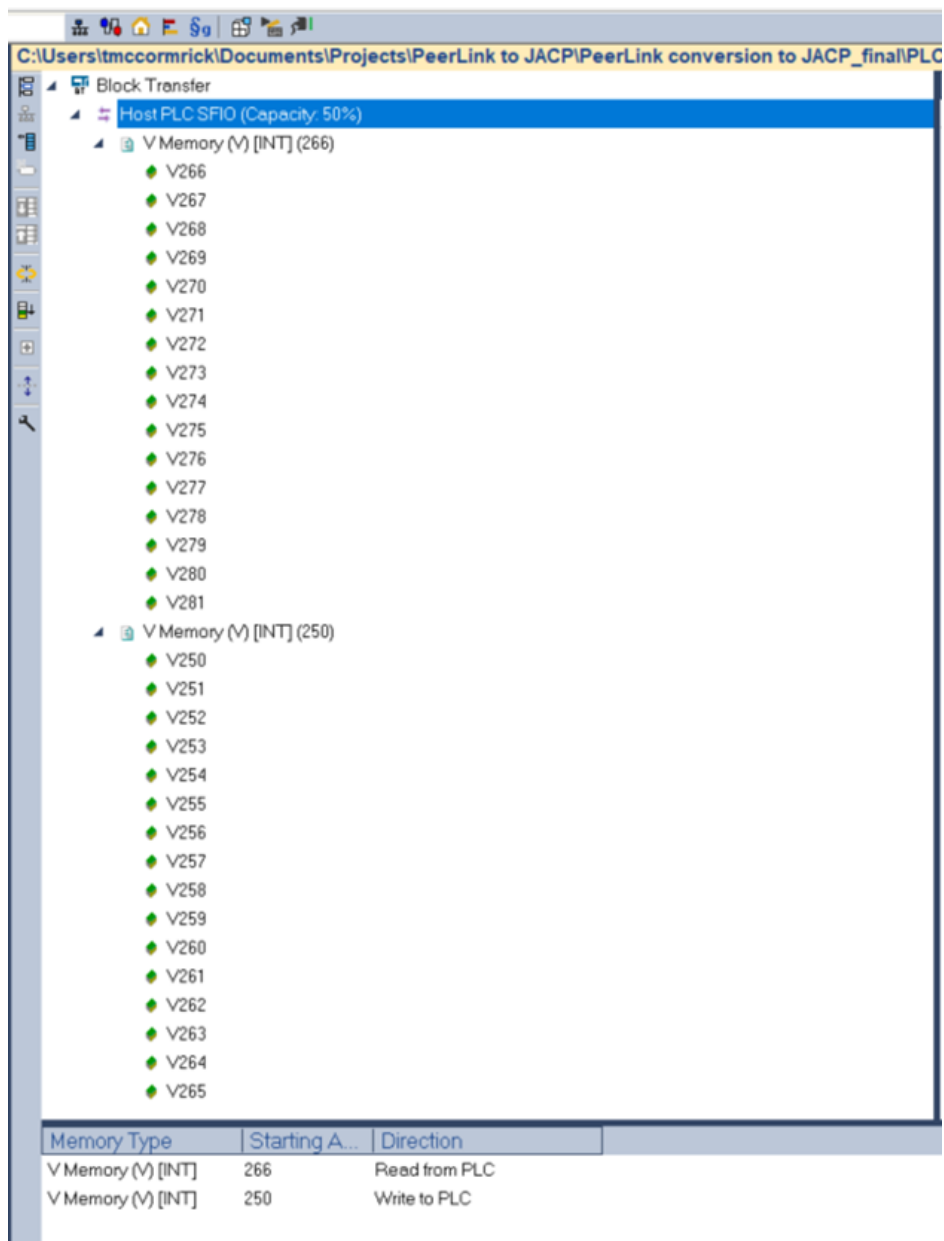


As in the JACP1 application, the Host PLC SFIO capacity needs to be modified for this particular application because the Simatic/TI 555 PLC only supports 8 SFIO task codes per scan, the capacity must be set to 50%. If both PLC's were CTI 2500 series then this parameter can be left at 100%.



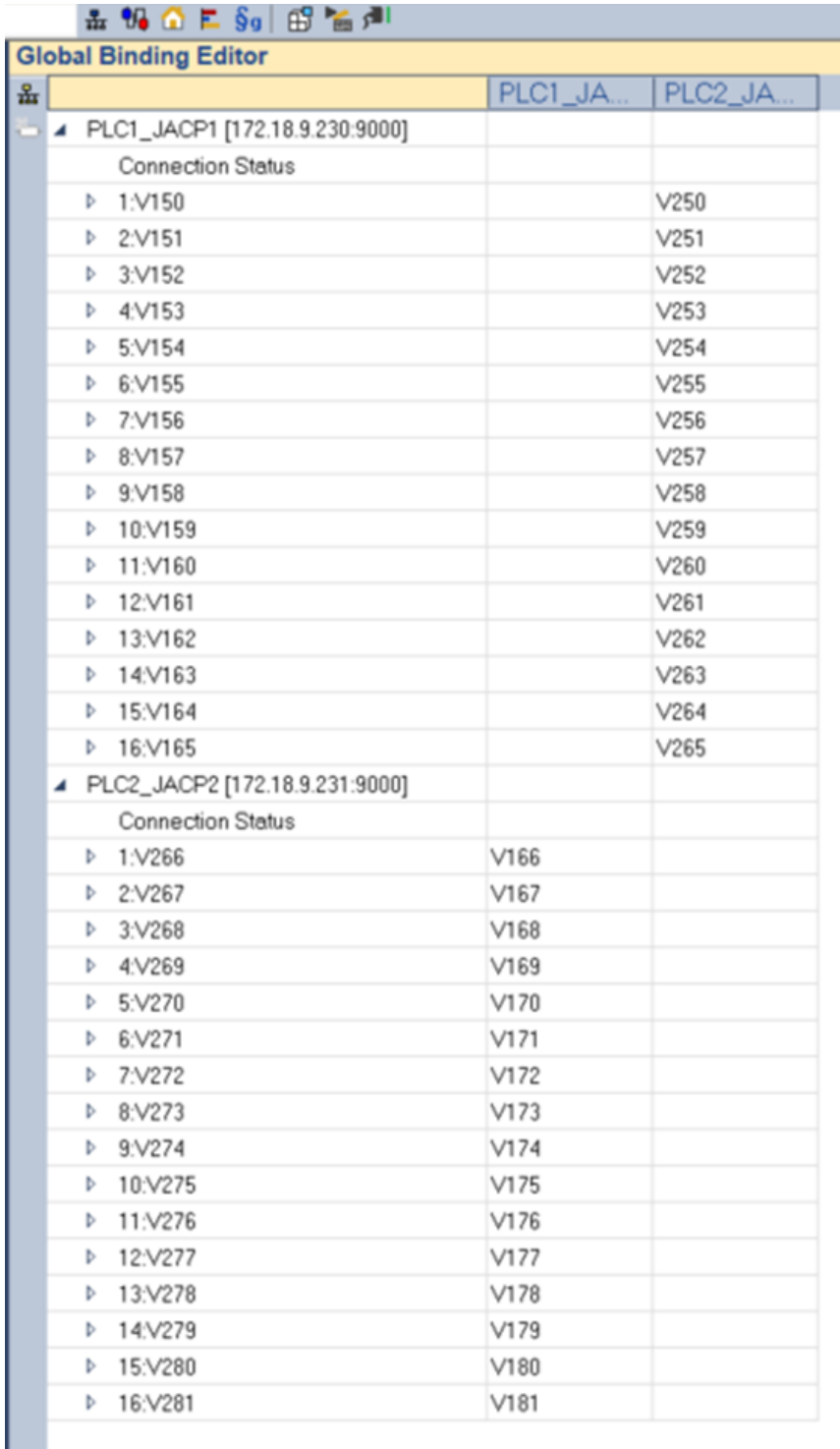


When complete, the JACP2 'Block Transfer' window should look like this:



7. Now we will configure how JACP1 and JACP2 will exchange data using the Global Binding editor in Workbench. Below is an overview of what the result of this step should look like. You can find detailed instruction on using the Global Binding Editor in the Appendix at the end of this document.

- From the 'Tools' menu select 'Global Binding Editor' to open the editor.
- Now add the two projects that you have already created.
- Next you should populate the fields as shown below:



Global Binding Editor		
	PLC1_JA...	PLC2_JA...
<ul style="list-style-type: none"> <li>PLC1_JACP1 [172.18.9.230:9000]           <ul style="list-style-type: none"> <li>Connection Status</li> <li>1:V150 → V250</li> <li>2:V151 → V251</li> <li>3:V152 → V252</li> <li>4:V153 → V253</li> <li>5:V154 → V254</li> <li>6:V155 → V255</li> <li>7:V156 → V256</li> <li>8:V157 → V257</li> <li>9:V158 → V258</li> <li>10:V159 → V259</li> <li>11:V160 → V260</li> <li>12:V161 → V261</li> <li>13:V162 → V262</li> <li>14:V163 → V263</li> <li>15:V164 → V264</li> <li>16:V165 → V265</li> </ul> </li> <li>PLC2_JACP2 [172.18.9.231:9000]           <ul style="list-style-type: none"> <li>Connection Status</li> <li>1:V266 → V166</li> <li>2:V267 → V167</li> <li>3:V268 → V168</li> <li>4:V269 → V169</li> <li>5:V270 → V170</li> <li>6:V271 → V171</li> <li>7:V272 → V172</li> <li>8:V273 → V173</li> <li>9:V274 → V174</li> <li>10:V275 → V175</li> <li>11:V276 → V176</li> <li>12:V277 → V177</li> <li>13:V278 → V178</li> <li>14:V279 → V179</li> <li>15:V280 → V180</li> <li>16:V281 → V181</li> </ul> </li> </ul>		



The first column is the Publish or Write registers, while the 2<sup>nd</sup> and 3<sup>rd</sup> columns are the Subscribe or Read registers.

Notice how the columns have the name of each partner.

In this case then, the contents of register V150 in PLC1\_JACP1 will write into V250 in PLC2\_JACP2. Likewise, the contents of register V266 in PLC2\_JACP2 will write into V166 in PLC1\_JACP1.

## Compile and Download

Next compile and download both projects into their respective JACP module.

If configured as listed above, there should not be any compilation or download issues.

## Monitor / test

Launch 505WorkShop and go online to both of these PLC's. Next create a data window to view the contents of these registers.

The screenshot displays two data windows from the 505 WorkShop software. The left window, titled 'C400\_JACP1.wdt - JACP\_TEST\_C400 (Online)', shows a list of registers for PLC1\_JACP1. The right window, titled 'T1555\_JACP2.wdt - JACP\_TEST\_T1555 (Online)', shows a list of registers for PLC2\_JACP2. Both windows have a table with columns for Row, Address, Tag, Description, Value, Time Stamp, and Status.

Row	Address	Tag	Description	Value	Time Stamp	Status
1	V150			333	02:18:43:768 PM 07/17/23	Success
2	V151			0	02:18:43:768 PM 07/17/23	Success
3	V152			0	02:18:43:768 PM 07/17/23	Success
4	V153			0	02:18:43:768 PM 07/17/23	Success
5	V154			0	02:18:43:768 PM 07/17/23	Success
6	V155			0	02:18:43:768 PM 07/17/23	Success
7	V156			0	02:18:43:768 PM 07/17/23	Success
8	V157			0	02:18:43:768 PM 07/17/23	Success
9	V158			0	02:18:43:768 PM 07/17/23	Success
10	V159			0	02:18:43:768 PM 07/17/23	Success
11	V160			0	02:18:43:768 PM 07/17/23	Success
12	V161			0	02:18:43:768 PM 07/17/23	Success
13	V162			0	02:18:43:768 PM 07/17/23	Success
14	V163			0	02:18:43:768 PM 07/17/23	Success
15	V164			0	02:18:43:768 PM 07/17/23	Success
16	V165			0	02:18:43:768 PM 07/17/23	Success
17						
18	V166			777	02:18:43:768 PM 07/17/23	Success
19	V167			0	02:18:43:768 PM 07/17/23	Success
20	V168			0	02:18:43:768 PM 07/17/23	Success
21	V169			0	02:18:43:768 PM 07/17/23	Success
22	V170			0	02:18:43:768 PM 07/17/23	Success
23	V171			0	02:18:43:768 PM 07/17/23	Success
24	V172			0	02:18:43:768 PM 07/17/23	Success
25	V173			0	02:18:43:768 PM 07/17/23	Success
26	V174			0	02:18:43:768 PM 07/17/23	Success
27	V175			0	02:18:43:768 PM 07/17/23	Success
28	V176			0	02:18:43:768 PM 07/17/23	Success
29	V177			0	02:18:43:768 PM 07/17/23	Success
30	V178			0	02:18:43:768 PM 07/17/23	Success
31	V179			0	02:18:43:768 PM 07/17/23	Success
32	V180			0	02:18:43:768 PM 07/17/23	Success
33	V181			0	02:18:43:768 PM 07/17/23	Success

Row	Address	Tag	Description	Value	Time Stamp	Status
1	V266			777	02:18:43:768 PM 07/17/23	Success
2	V267			0	02:18:43:768 PM 07/17/23	Success
3	V268			0	02:18:43:768 PM 07/17/23	Success
4	V269			0	02:18:43:768 PM 07/17/23	Success
5	V270			0	02:18:43:768 PM 07/17/23	Success
6	V271			0	02:18:43:768 PM 07/17/23	Success
7	V272			0	02:18:43:768 PM 07/17/23	Success
8	V273			0	02:18:43:768 PM 07/17/23	Success
9	V274			0	02:18:43:768 PM 07/17/23	Success
10	V275			0	02:18:43:768 PM 07/17/23	Success
11	V276			0	02:18:43:768 PM 07/17/23	Success
12	V277			0	02:18:43:768 PM 07/17/23	Success
13	V278			0	02:18:43:768 PM 07/17/23	Success
14	V279			0	02:18:43:768 PM 07/17/23	Success
15	V280			0	02:18:43:768 PM 07/17/23	Success
16	V281			0	02:18:43:768 PM 07/17/23	Success
17						
18	V250			333	02:18:43:768 PM 07/17/23	Success
19	V251			0	02:18:43:768 PM 07/17/23	Success
20	V252			0	02:18:43:768 PM 07/17/23	Success
21	V253			0	02:18:43:768 PM 07/17/23	Success
22	V254			0	02:18:43:768 PM 07/17/23	Success
23	V255			0	02:18:43:768 PM 07/17/23	Success
24	V256			0	02:18:43:768 PM 07/17/23	Success
25	V257			0	02:18:43:768 PM 07/17/23	Success
26	V258			0	02:18:43:768 PM 07/17/23	Success
27	V259			0	02:18:43:768 PM 07/17/23	Success
28	V260			0	02:18:43:768 PM 07/17/23	Success
29	V261			0	02:18:43:768 PM 07/17/23	Success
30	V262			0	02:18:43:768 PM 07/17/23	Success
31	V263			0	02:18:43:768 PM 07/17/23	Success
32	V264			0	02:18:43:768 PM 07/17/23	Success
33	V265			0	02:18:43:768 PM 07/17/23	Success

Notice in these two windows how PLC1\_JACP1 register V150 has the value of '333' and has written that value into PLC2\_JACP2 register V250.

Also, notice that PLC2\_JACP2 register V266 has the value of '777' and has written that value into PLC1\_JACP1 register V166.



# Appendix

The following section is taken from the Janus Workbench Training Course 2, version 2.0

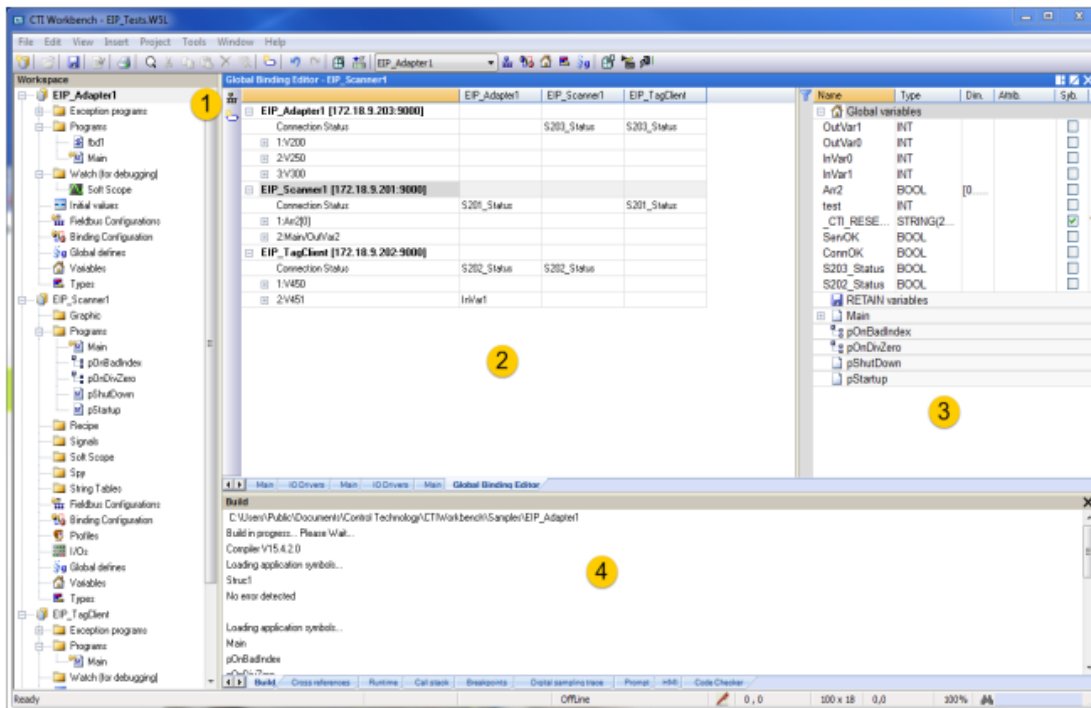
## 8.3 Global Binding Editor

The **Global Binding Editor** provides a simplified interface for building the Binding configuration for each of the networked projects. The information entered in this utility is saved into the Binding configuration for each project and can be viewed and/or edited later in the project's **Binding Configuration Editor**.

The Global Binding Editor is started by selecting the Tools / Global Binding Editor menu command.

### 8.3.1 Editor Workspace

Below is the general form of the Global Binding Editor:

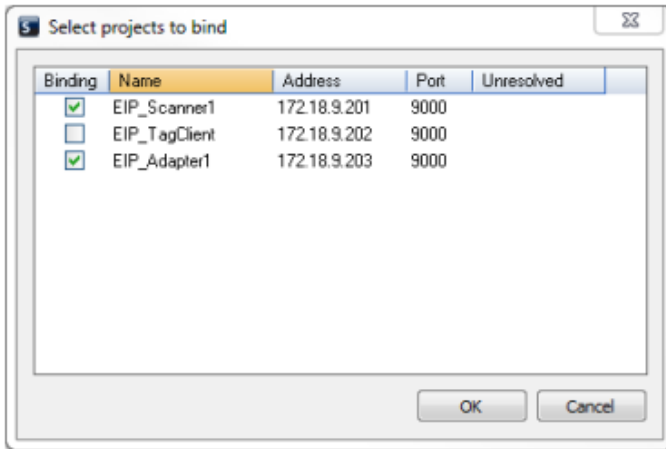


- Network Node Selector Icon
- Editing Grid
- Variable List of the selected project
- Output window for reports



## 8.3.2 Network Node Selection

The project configurations for all runtime targets that will act as Publisher or Subscriber must be included in current Workspace. Projects to be included in the Global Binding configuration are selected by clicking on the Add/Remove Projects button to display the following dialog:



All Projects in the current Workspace are included in the list. Check Binding box next to each Project to include it in the Network Binding configuration. Un-check Binding box to exclude a Project from Network Binding configuration.

Enter IP Address and Port Number for runtime target where project will run.

**Note: Binding Port Number for all CTI devices is 9000.**

## 8.3.3 Editing Grid

The Editing Grid consists of a matrix with projects and its produced variables listed in the left column with additional columns to hold the variables consumed in each destination project.

**Example**

Global Binding Editor - EIP_Scanner1		
	EIP_Adapter1	EIP_Scanner1
<b>EIP_Adapter1 [172.18.9.203:9000]</b>		
Connection Status		S201_Status
1:V200		Recv1
2:V250		
3:V300		
<b>EIP_Scanner1 [172.18.9.201:9000]</b>		
Connection Status	S203_Status	
1:OutVar0	InVar0	
2:OutVar1	InVar1	

This example shows that the project **EIP\_Adapter1** produces its variables V200, V250 and V300, and the project EIP\_Scanner1 produces its variables OutVar0 and OutVar1. The selected cell shows that the variable V200 of source project **EIP\_Adapter1** is bound to the variable Recv1 of destination project **EIP\_Scanner1**. Also, variables OutVar0 and OutVar1 of source project

**EIP\_Scanner1** are bound to destination project **EIP\_Adapter1** variables InVar0 and InVar1 respectively. The link identifier of each source variable is listed in front of the published variable name (“1” in this example for variable V200).

Note that each project also publishes by default a “Connection Status” variable which can be consumed by, or bound to, other projects. In this example, the project **EIP\_Adapter1** publishes “Connection Status”, which is bound to the “S201\_Status” variable in the project **EIP\_Scanner1**.

Variables can also be added by selecting a variable in Variable List in the top-right window of the Editing Grid and dragging it to the proper location in the matrix, or by selecting a cell in the matrix and then choosing Insert Variable from the menu displayed by right-clicking the mouse.

At any time, you can double-click on any source project in the first column to display the Network Node Selection dialog described above.

### 8.3.4 Produced Variables

A variable may be assigned for publishing by selecting the project name in the first column, and "drag and drop" variable from the Variable List to the cell with project name. You can also right-click on the project name and select Insert Variable from the drop-down list.

When a numerical data type (i.e. INTEGER OR REAL) source variable is selected, double click on the variable name to define a hysteresis for its change detection. If no hysteresis value is assigned, any change in the variable value will cause the new variable value to be published.

### 8.3.5 Consumed Variables

To add a consumed variable, double-click on the appropriate cell in the matrix corresponding to the published source variable and the destination project. Then select a variable of the project variable list displayed in the pop-up window. Variables can also be added by selecting the destination project name in the first column, and "drag and drop" a variable from the Variable List in the upper right-hand corner to the appropriate cell in the matrix


### 8.3.6 Variable Status Information

The status of each TCP connection between Subscriber and Publisher projects can be monitored by adding a variable to the "Connection Status" cell that corresponds to the source/destination projects. In the Editing Grid example shown above, variable S203\_Status in project **EIP\_Adapter1** is used to monitor TCP binding connection status with project **EIP\_Scanner1**. A value of 0 (or FALSE for a BOOLEAN variable) indicates the connection is OK. A value of 1 (or TRUE) indicates an error condition.





Additionally, each produced variable packet includes a set of information that can be saved and used by the Subscriber application. The variable data set is displayed by expanding the produced variable name, and each item can be linked to destination variables as shown below:

 **Example**

	EIP_Adapter1	EIP_Scanner1
[-] EIP_Adapter1 [172.18.9.203:9000]		
Connection Status		
[-] 1:V200		Recv1
Error Status		Recv1_Error
Date Stamp		Recv1_Date
Time Stamp		Recv1_Time

This example shows that the V200 variable data from source project **EIP\_Adapter1** is bound to the following variables in the destination project **EIP\_Scanner1**:

Variable Recv1 contains the last received value.

Variable Recv1\_Error contains the status of the previous produced variable message packet (0 or FALSE = OK / 1 or TRUE = Error).

Variable Recv1\_Date contains a Date stamp of the last variable update. **Variable used for Date stamp must be declared as DINT. Real-Time Clock functions can be used to extract Date (Year/Month/Day).**

Variable Recv1\_Time contain a Time stamp of the last variable update. **Variable used for Time stamp must be declared as DINT or 'Time' data type. When DINT is used, the Real-Time Clock functions can be used to extract Time (Hour/Minute/Second/Millisecond).**

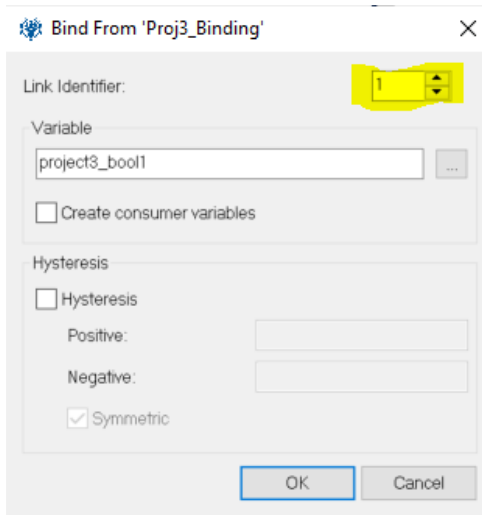
The Date/Time stamps are inserted into each variable message by the Publisher. These stamps are automatically updated with the current values from the Real-Time Clock unless the VSISTAMP function is used in source project to force an update to the Date and/or Time stamp of a particular variable. Once the Time/Date stamp is updated by this function, that value is reported in all future messages published for that variable unless the Time/Date stamp is updated by running the function again.

### 8.3.7 Declaring Variables

The Global Binding Editor allows the declaration of new variables in source and destination projects without need of opening the projects. Right-click on a cell in the editor matrix and select Insert Variable from the drop-down menu. The variable name and data type can then be declared for the associated project.

## 8.3.8 Link identifiers

The Global Binding Editor automatically allocates identifiers to associate each produced variable with its matching consumed variable(s). These identifiers are not visible in this editor, but you can see them by double-clicking on a produced variable to bring up the properties window. This is also where the hysteresis settings are found.



**IMPORTANT SIDE NOTE:** The Link Identifier highlighted above is also used by the CTI 2500P-ECC1 when using Network Data Exchange. This common technology allows an ECC1 to Subscribe to data that is published using Binding on a 2500P-ACP1 or Janus Processor.

## 8.3.9 Practice Exercise 8A – Global Binding Editor

In this exercise we will create two new projects and use the Global Binding Editor to share variables between them.

First, create two new projects, Global\_Binding1 (IP address 10.25.1.106) and Global\_Binding2 (IP address 10.25.1.103). Use ST as the default programming language.

Create the following variables in each project:

### Global\_Binding1

Binding1_bool1	Boolean
Binding1_integer1	integer
Binding1_bool2	boolean
Binding1_integer2	integer

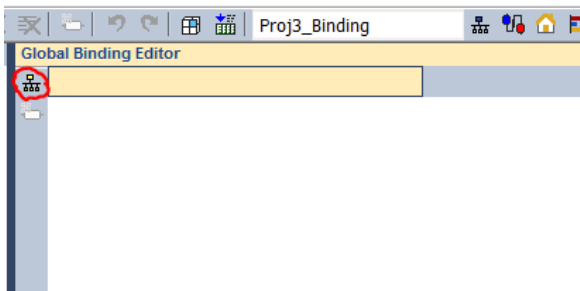
### Global\_Binding2

Binding2_bool1	boolean
Binding2_integer1	integer
Binding2_bool2	boolean
Binding2_integer2	integer



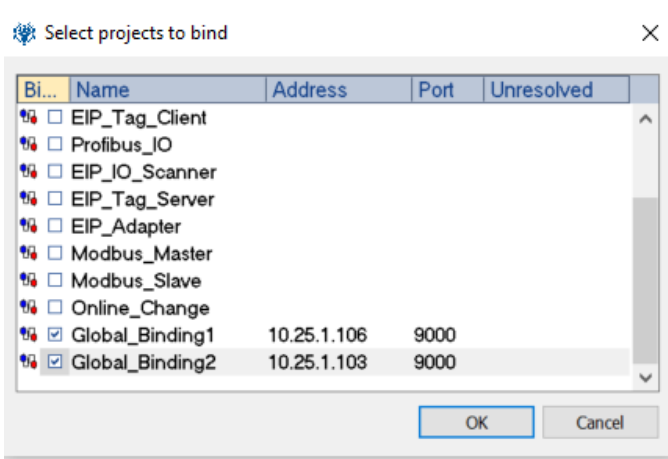
In this exercise, each project will publish `Bindingn_bool1` and `Bindingn_integer1`. Each project will subscribe to the other's `bool1` and `integer1` and store those in `Bindingn_bool2` and `Bindingn_integer2`

Go to Tools / Global Binding Editor to start the editor.

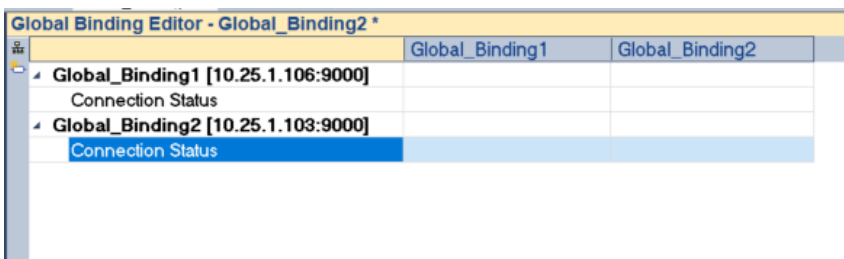


Click the Add / Remove Projects icon.

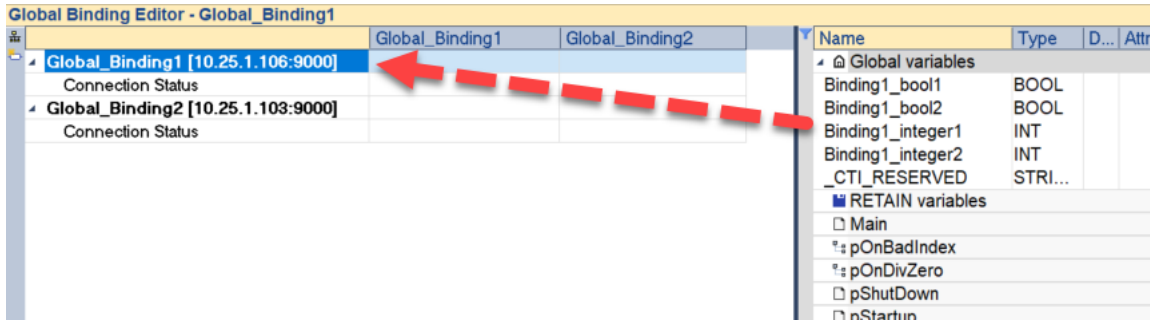
Tick the "Binding" box for `Global_Binding1` and `Global_Binding2`, and enter the IP addresses (if not already present).



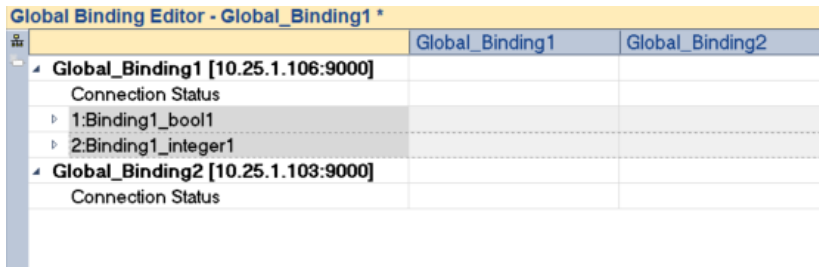
Click OK to return to the editor.



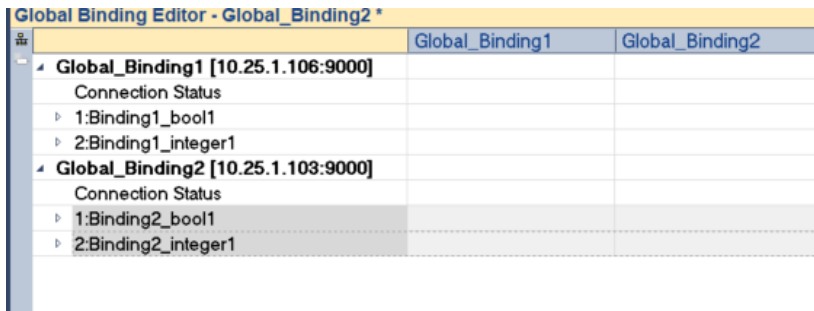
Highlight **Global\_Binding1** as shown above. Then drag the variables Binding1\_bool1 and Binding1\_integer1 from the variable window to the “Proj3\_Binding” heading in the editor.



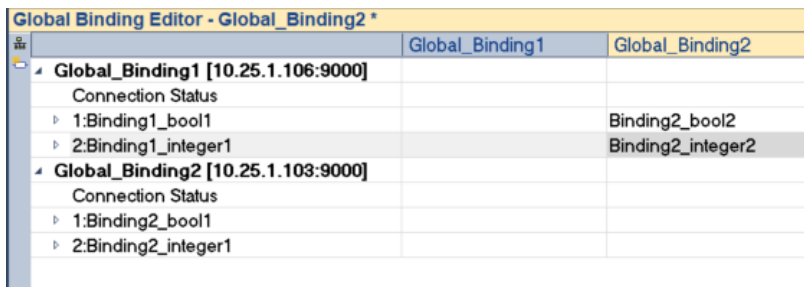
Your editor should now look like this:



Repeat this process for **Global\_Binding2**, using Binding2\_bool1 and Binding2\_integer1.



Now we will make the subscriptions for **Global\_Binding2**. Highlight **Global\_Binding2** then drag the variables Binding2\_bool2 and Binding2\_integer2 into the “Global\_Binding2” column and in-line with Binding1\_bool1 and Binding1\_integer1.



Repeat this process for Global\_Binding1 and the variables Binding1\_bool2 and Binding1\_integer2. The finished configuration looks like this:

Global Binding Editor - Global_Binding1 *		
	Global_Binding1	Global_Binding2
Global_Binding1 [10.25.1.106:9000]		
Connection Status		
▶ 1:Binding1_bool1		Binding2_bool2
▶ 2:Binding1_integer1		Binding2_integer2
Global_Binding2 [10.25.1.103:9000]		
Connection Status		
▶ 1:Binding2_bool1	Binding1_bool2	
▶ 2:Binding2_integer1	Binding1_integer2	

That completes our Global Binding Exercise!

