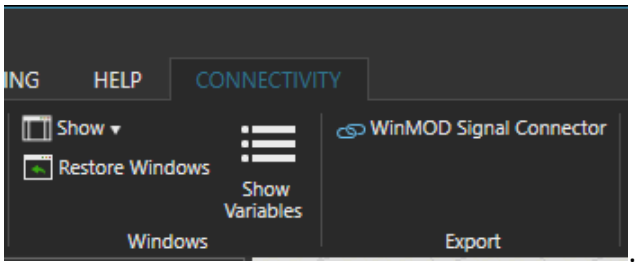
WinMOD Signal Connector

WinMOD Signal Connector add-on can be used to automate connecting variables in WinMOD and Visual Components. Add-on creates two files. Signal list file (.txt) can be used to create operands in WinMOD and also expose them for connectivity. Connection file (.xml) can be used to import connectivity setup in VC. XML is needed as there's no API at the moment to create connections in VC.

How to install

To install the add-on copy add-on folder under your **Documents\Visual Components\4.X\My Commands** folder. This will add a new button under Connectivity tab.



How to use

The idea on the add-no is that you may define which variables are connected on the VC model. This is done by using templates. For every component in the layout you can associated a template file with any string property in the components. By default add-on uses *Category* property which is a standard property in every VC component. With a *Category* value of *Conveyor* add-on will try to find a template called *Conveyor.txt* under add-on's *Templates* folder. If template is found then add-on uses the template to define connected variables for that component on the output files.

Basic workflow is to first model simulation components compatible with co-simulation. Then you create template for every component type. You build a layout and use add-on to export variables for WinMOD. Then you build the WinMOD model for co-simulation. Finally you use the add-on to connect both WinMOD and VC layout.

Component modeling

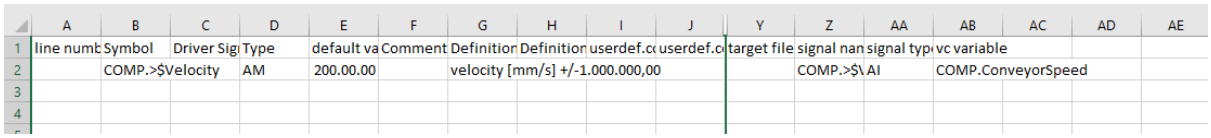
In component modeling you can use component properties, signal behaviors and behavior properties and the add-on supports them all. Connected properties should either Boolean, Integer and Real so that they are compatible with WinMOD operand types which are Binary, Digital and Analog.

Templates

When creating templates you can start by copying *Example.txt* template, rename it to some other type and define variables inside the file. File format is basically comma separated file (.csv) except the separator character is tab (by default) or semicolon ";". First row in the file is the header and you should not modify it. Following lines contain the connection variables.

Template has total of 28 columns. 27 columns are standard WinMOD columns and the final 28th column is the VC variable. Most of the columns are optional so you don't need to fill any value to them. Here we list only the most important columns which you need to pay attention to. These **important 7 columns** in the template file are called ***symbol***, ***type***, ***default value***, ***definition internal format***, ***signal name***, ***signal type***, ***vc variable***. Columns *symbol*, *type*, *default value* and *definition internal format* are used in creating operands in WinMOD. Columns *signal name* and *signal type* are used in exposing connection variable in WinMOD communication element. Last column *vc variable* marks the VC variable.

Below there's an example template for type *Conveyor*. Note that it's easiest to modify template files using Excel.



	A	B	C	D	E	F	G	H	I	J	Y	Z	AA	AB	AC	AD	AE
1	line numb	Symbol	Driver Sig	Type	default va	Comment	Definition	Definition	userdef.c	userdef.c	target file	signal nan	signal typ	vc variable			
2		COMP.>$Velocity		AM	200.00.00		velocity [mm/s] +/-1.000.000,00					COMP.>$\	AI	COMP.ConveyorSpeed			
3																	
4																	

In template's variable fields ***symbol*** and ***signal name*** usually have the same value. Value syntax is *COMP.xxx* where *COMP* is a placeholder for the device ID and *xxx* is the variable in WinMOD. In the default template naming convention is used where *COMP.>$xxx* notation is used for inputs coming to VC from WinMOD and *COMP.<$xxx* notation is used for outputs going from VC to WinMOD.

Field ***type*** is the operand type in WinMOD and its value should be either *BM* (binary), *DM* (digital) or *AM* (analog). Field ***default value*** can have a default value for the operand but it can also be empty. Field ***definition internal format*** can have some user-defined WinMOD format but it can also be empty. Field ***signal type*** is the type in WinMOD communication element and its value should be either *BI* (binary input), *BO* (binary output), *DI* (digital input), *DO* (digital output), *AI* (analog input) or *AO* (analog output).

Last field ***vc variable*** should have the value given as one of the following notations:

COMP.xxx for component property of name *xxx*.

COMP.xxx for signal behavior of name *xxx*.

COMP.yyy.xxx for behavior property, where *yyy* is the behavior name and *xxx* is the property name.

Again *COMP* is a placeholder on previous notations. Placeholder is replaced with the component name on the output.

Running the example template *Conveyor.txt* to a component with name *CNV_1* will produce a following entry in the signal list .txt and connection .xml:

	A	B	C	D	E	F	G	H	I	J	Z	AA	AB	AC	AD
1	line numb	Symbol	Driver Sig	Type	default va	Comment	Definition	Definition	userdef.c	userdef.c	signal nan	signal typ	vc variable		
2		CNV_1.>$Velocity		AM	200.00.00		velocity [mm/s]	1.000.000,00			CNV_1.>$'	AI	CNV_1.ConveyorSpeed		

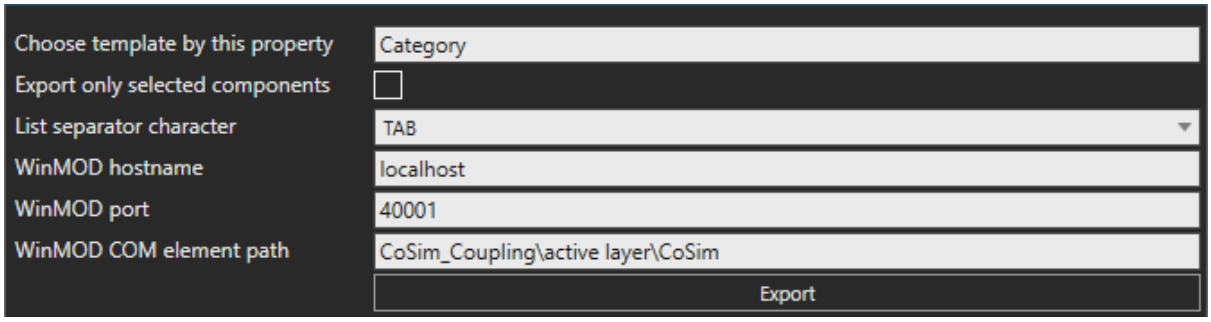
Figure 1 Signal list file entry

```
<Array:anyType i:type="VariableGroupItem">
  <DisplayName i:nil="true" />
  <ServerItem i:type="WinmodNet:WinmodVariableID">
    <WinmodNet:CommunicationElementName>CoSim_Coupling\active layer\CoSim
    </WinmodNet:CommunicationElementName>
    <WinmodNet:DataDirection>Input</WinmodNet:DataDirection>
    <WinmodNet:SignalName>CNV_1.&gt;$Velocity</WinmodNet:SignalName>
  </ServerItem>
  <SimulationVariable>
    <Component>CNV_1</Component>
    <Behaviour i:nil="true" />
    <Property>ConveyorSpeed</Property>
  </SimulationVariable>
</Array:anyType>
```

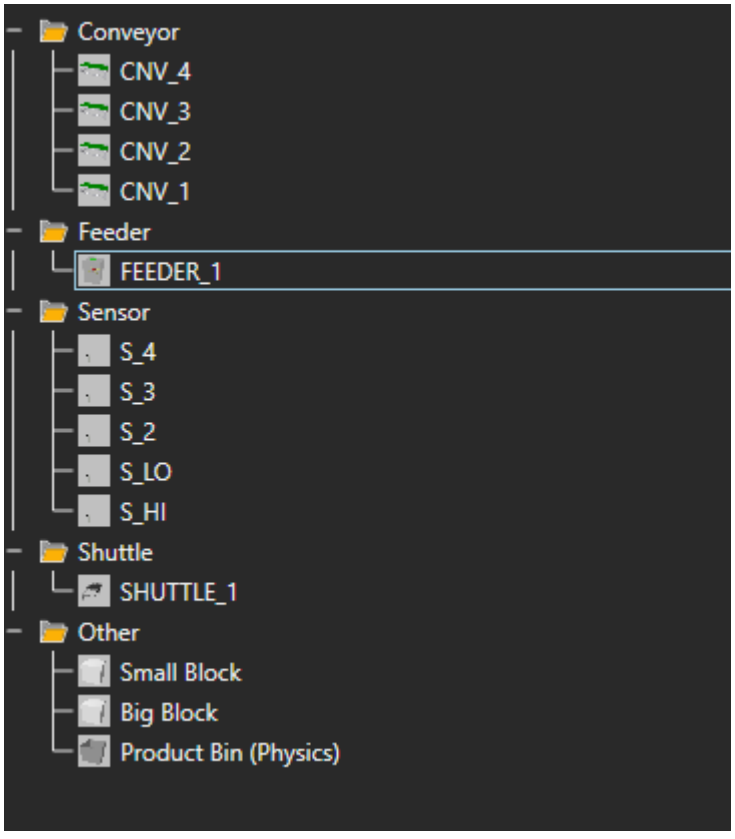
Figure 2 Connection xml entry

Add-on settings

Launching the add-on will open the action panel with some settings:



Choose template by this property value is the property name which you use to associate the components with your templates. By default it uses the value of *Category*. Category is convenient property to be used for selecting the template because you can use the cell graph to quickly view which device types you have on your layout. In the following example of a cell graph you see that there are 5 types (Categories) for which 4 have templates and *Other* Category lists components that don't need to be connected.

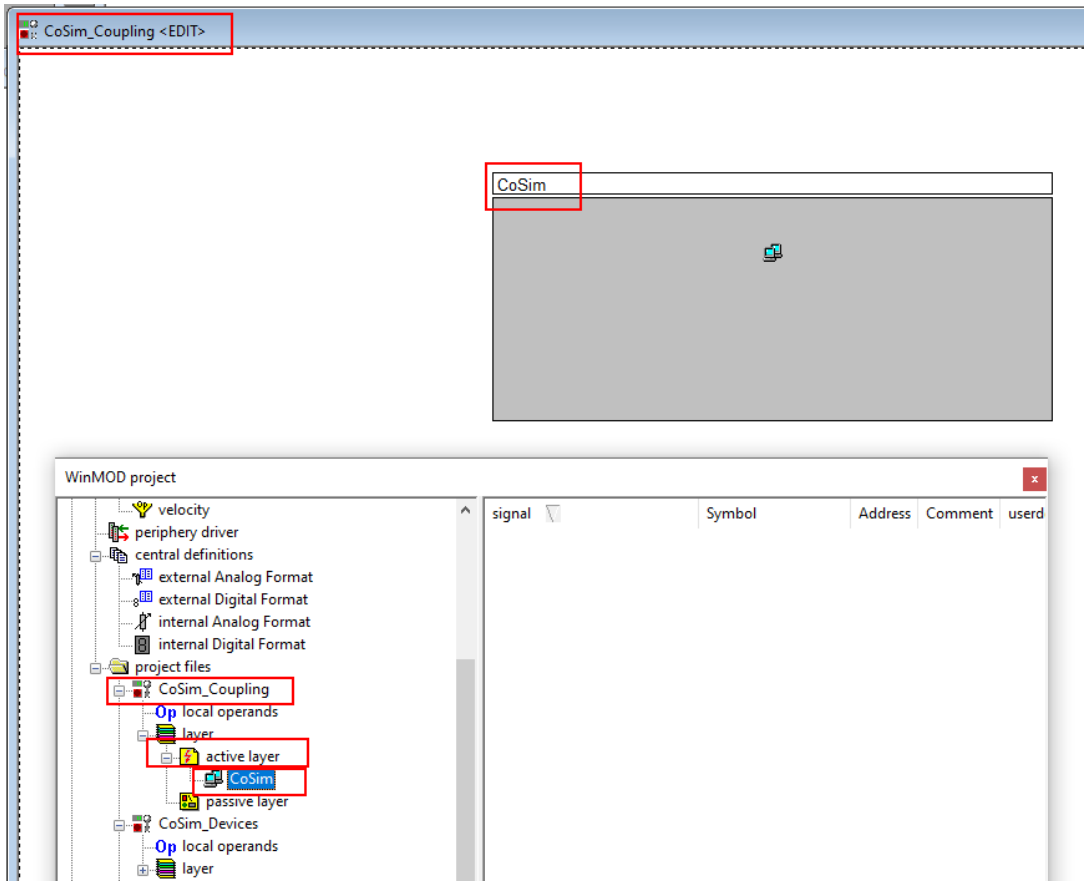


Export only selected components setting allows you to export signals only from selected components. If unchecked all components in the layout are used for export if they have a matching template.

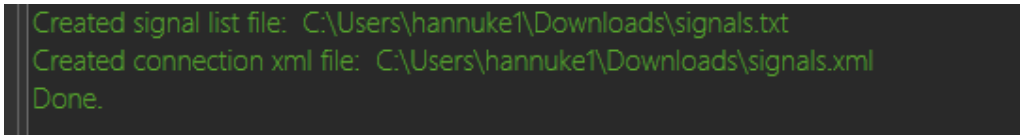
List separator character setting lets you choose if you want to use tab (default) or semicolon ";" as the list separator character in templates and the output signal list file.

WinMOD hostname and ***WinMOD port*** settings define where your WinMOD server will be running. These settings affect the settings on the connection xml.

WinMOD Com element path points to the location where the WinMOD variables are on the connection xml. Path name is *xxx\yyy\zzz* where *xxx* is the name of the WinMOD simulation, *yyy* is the name of layer and *zzz* is the name of the com element. In the picture below elements of example path *CoSim_Coupling\active_layer\CoSim* are highlighted on the WinMOD project.

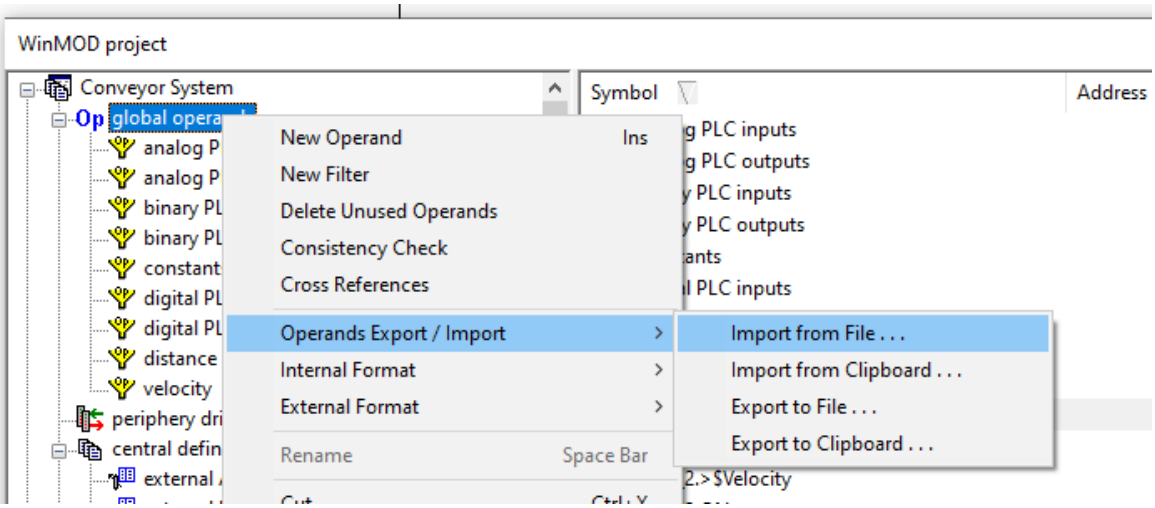


Export button in action panel opens up a save dialog where you define the name and location of the signal list text file. Connection xml is saved on the same location with the same name but only with *.xml* file extension instead of *.txt*.

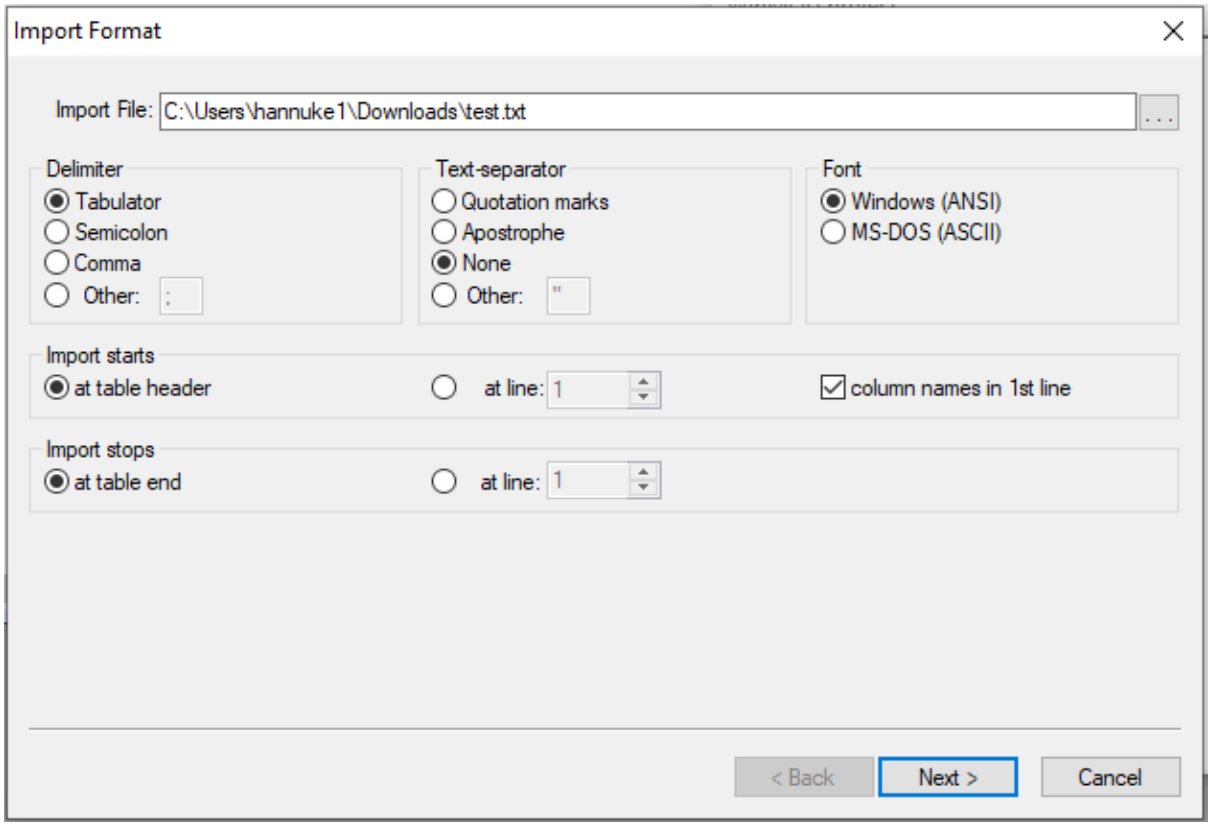


Signal list text file

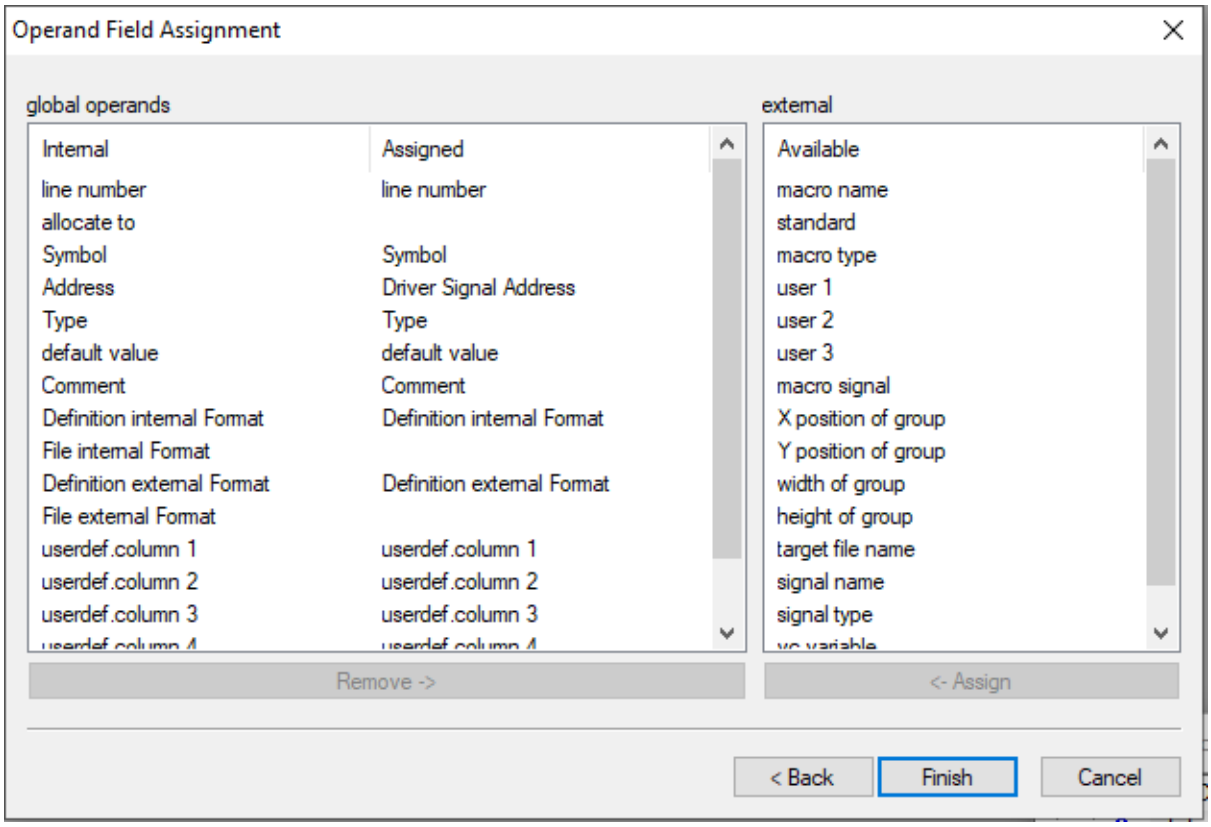
Text file that you create with the add-on can be used on WinMOD to create operands and expose them for connectivity. To create operands right-click over project's *global operands* and select *Operands Export / Import / Import from File.*



Use the following settings on the dialog and click *Next*.

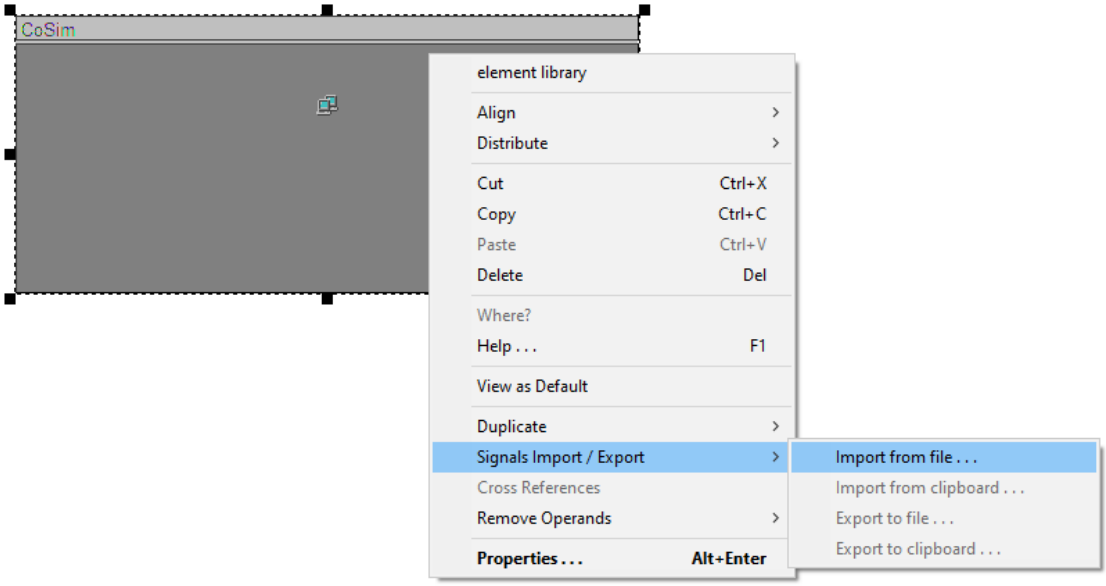


On the final page make the following column mapping and click *Finish*.

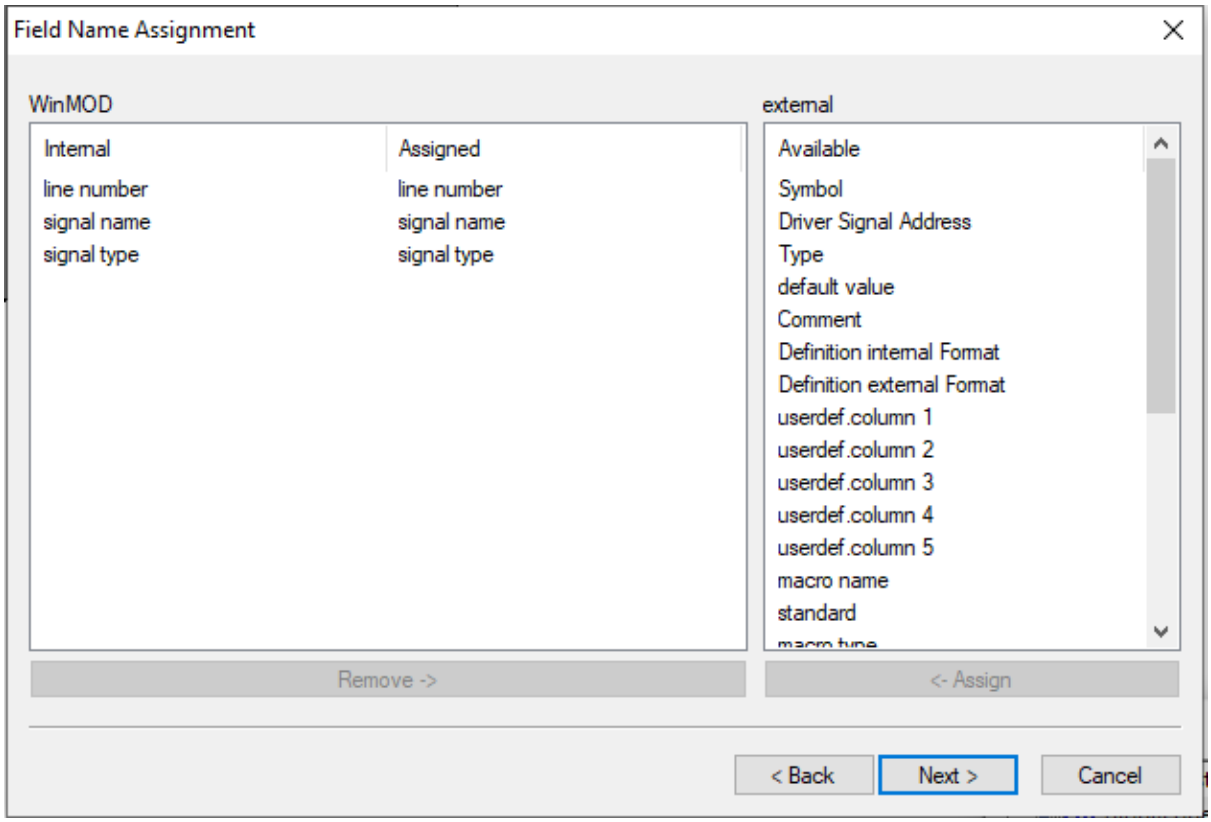


Wizard will create global operands from the signal list. If you already had operands with the same name then wizard uses the old operand and doesn't create a new one. You can now use these operands in your WinMOD simulation to create the simulation behaviors for drives etc.

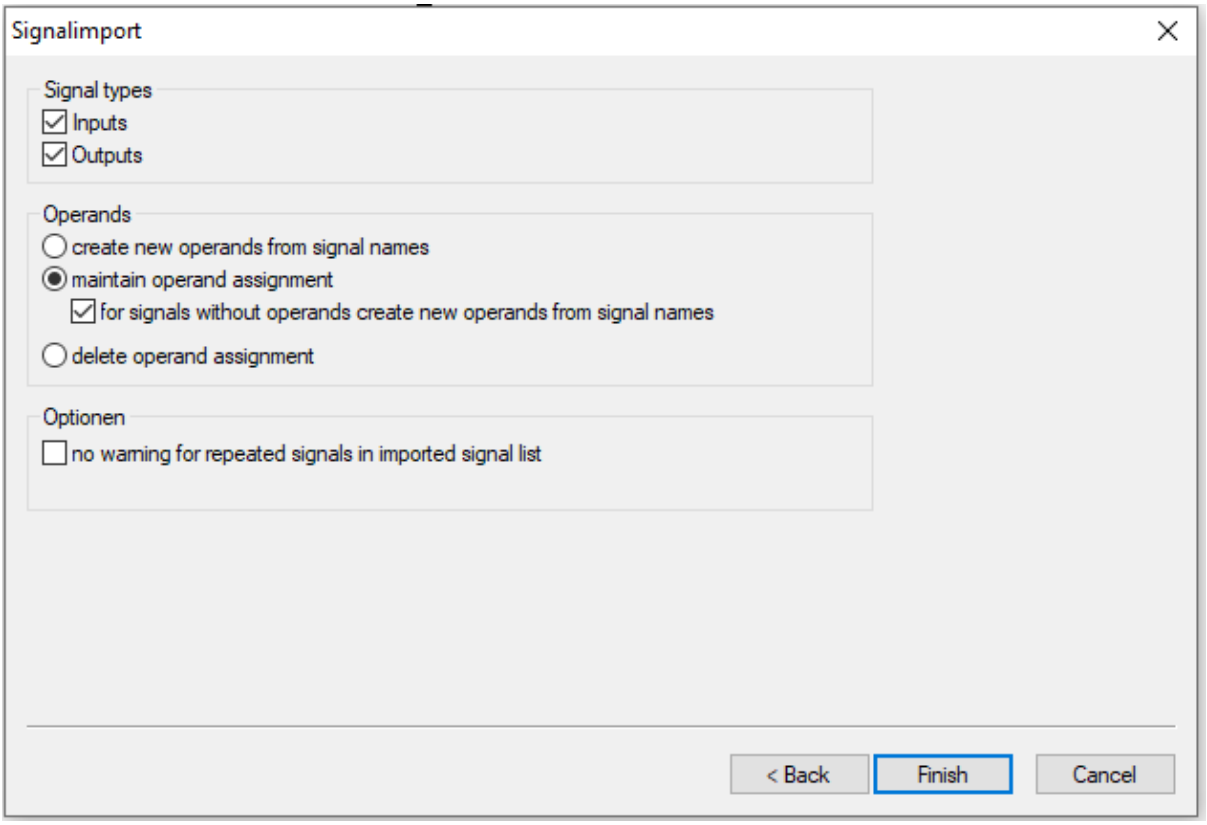
Another function that the signal list text files serves is to define the signals on a communication element in WinMOD. To define these signals first place a new communication element on your WinMOD simulation. Then right-click over it and select *Signals Improt / Export / Import from file.*



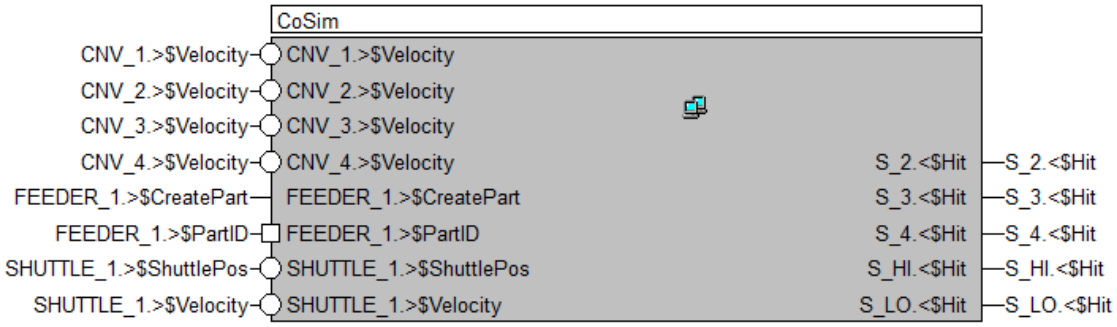
Use the same *Import format* settings as when creating operands but define column mapping like in the picture below.



On final page of the Import wizard use following settings and click *Finish*.



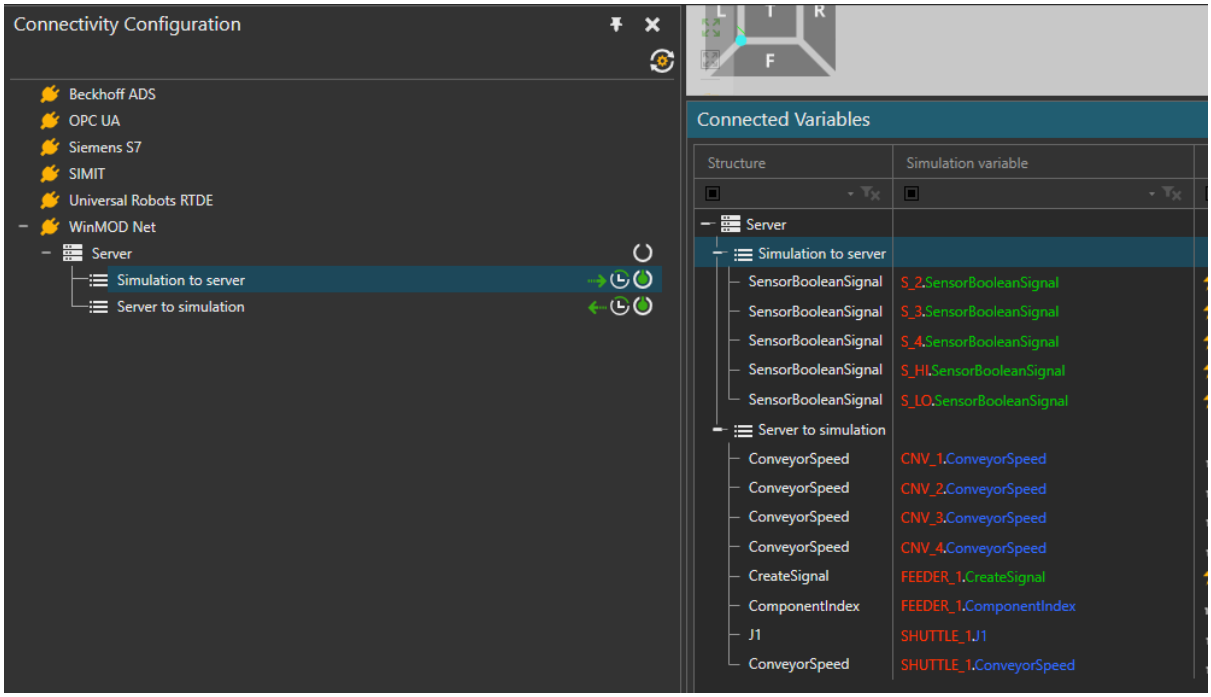
Wizard will create signals on the communication element.



Before you start to test connectivity make sure that connectivity is enabled on the WinMOD projects by right-clicking the project and selecting *Properties* and *Communication* tab. Enable incoming connections and set the port for them. After that start WinMOD simulation and you should be able to see variables when you connect to WinMOD server from Visual Components.

Connection XML file

After WinMOD operands are created and exposed for connectivity it's time to define variable pairings on Visaul Components model. You could do pairings manually but the add-on's connection XML can be used to automatically define pairings for you. Just go to *Connectivity* tab and under *Configuration* group click *Import*. Select your connection XML file and click *Open*. VC should recognize the file and create WinMOD server entry and variable pairings under it like shown on the picture below.



Now enable the server connection and start the simulation and co-simulation should run. If connection is not working make sure that WinMOD hostname and port are correctly set and also check that COM element path in variables is set correctly. If COM element path is wrong you can use add-on again and set path correctly and after that import new connection XML.

Example

There's an example bundled with this add-on that you can use to test the add-on itself. *Conveyor System.vcmx* is the VC model and *Conveyor System WMP* contains the WinMOD project. Before trying the example make sure you have the add-on properly installed.

Open example project on Visual Components (4.3 or newer). You can study the layout for example on Cell Graphs to see which kind of devices there are in the layout. In this case there are 4 conveyors, 1 shuttle, 5 sensors and 1 feeder that need to be connected to WinMOD. For all device types there's a template in the add-on. Use the add-on on the model and save signals to .txt and .xml files. Default setting should work on the add-on.

Now open the WinMOD project (*Conveyor System.wmp*). Co-sim operands are already creasted on the example as globals. On *CoSim_Coupling* simulation there's an communication element that you need to define the signals into. Import signal list .txt to com element like shown on this manual. Import should define 8 inputs and 5 outputs.

Next go back to Visual Components and import communication .xml. Establish a connection to server and start simulation on both WinMOD and VC. You can now test the co-simulation by enabling the line on WinMOD's HMI simulation and creating small and big blocks on the line. WinMOD will feed conveyor speeds and shuttle position to VC model and VC will send sensor information back to WinMOD. In this example there's no PLC involved but the line logic is built into the WinMOD project and its PLC simulation. Example logic will route small blocks to right lane and big blocks to left lane.