

LabVIEW Connectivity

This document provides information on using Origin as an Automation Server from a LabVIEW client.

Origin supports communication with LabVIEW versions 5.0, 6.1, and 7.0. There are two aspects to the LabVIEW connectivity in Origin:

1. Building-block VI's
2. LabVIEW VI Browser Tool

Building-block VI's

The Origin product is shipped with a collection of building-block VI's that can be included in your custom VI's to easily communicate with Origin. These building-block VI's all have names that start with OA, such as:

- OAOpenCommunication.VI – open communication with Origin
- OACloseCommunication.VI – close communication
- OAExecute.VI – send a LabTalk script command to Origin

....

These building-block VI's are contained in the file "Origin Automation Server.llb". When Origin is installed on a system in which LabVIEW was already installed, this file is placed under the "\user.lib" subfolder of the the LabVIEW installation folder. These VI's will then be visible from the Functions->User Libraries palette in LabVIEW. If Origin is installed before LabVIEW, this file should be copied from the "\Samples\Automation Server\LabVIEW" subfolder of the Origin installation to the \user.lib subfolder of the LabVIEW installation, in order for them to be accessible from the LabVIEW Functions palette. Note that there are three LabVIEW subfolders under the Origin Samples area, specific to LabVIEW versions 5.0, 6.1, and 7.0.

LabVIEW VI Browser Tool

In addition to the collection of building-block VI's shipped with Origin, a "LabVIEW VI Browser" tool is also provided. This tool, which can be launched from the Tools menu, provides a way for the Origin user to browse their LabVIEW VI's, select a particular VI, and then run the VI from within the Origin environment. For instance, if you create a particularly LabVIEW VI that collects data and then sends the data to an Origin worksheet or matrix, you can browse to that VI and run the VI to place the data in Origin, and then perform further graphing and analysis on the data with Origin. This tool also allows you to transfer data contained in a VI to an Origin worksheet or matrix.

Creating custom VI's using Origin's building-block VI's

In this section, three tutorials are presented that walk the user through the process of creating custom VI's using the building-block VI's provided with the Origin installation. It is assumed in these tutorials that the building-block VI's have been added to the user.lib folder of LabVIEW so that they are visible from the Functions palette. For more information on this, please refer to the previous sections.

Tutorial 1 – Sending a script command to Origin

In this tutorial, we will create a LabVIEW VI that sends script (LabTalk) commands to Origin.

1. Launch Origin.
2. Launch LabVIEW.
3. In LabVIEW, start a new VI.
4. Activate the Front Panel window, right-click in window, and from Controls palette, choose String and Path, and then a String Control. Place this control in the window and rename it to “Script Command”. Fill this control with the string “type –a Hello from LabVIEW!”.
5. Activate the Block Diagram window.
6. Right click inside the window, and from the Functions palette, choose User Libraries, and then paletteMenu. Then choose OAOpenCommunication.vi and place it in the window.



The icon for this VI is

7. Right-click again and add the OAEExecute VI from the same palette, and place it next to the OAOpenCommunication VI.



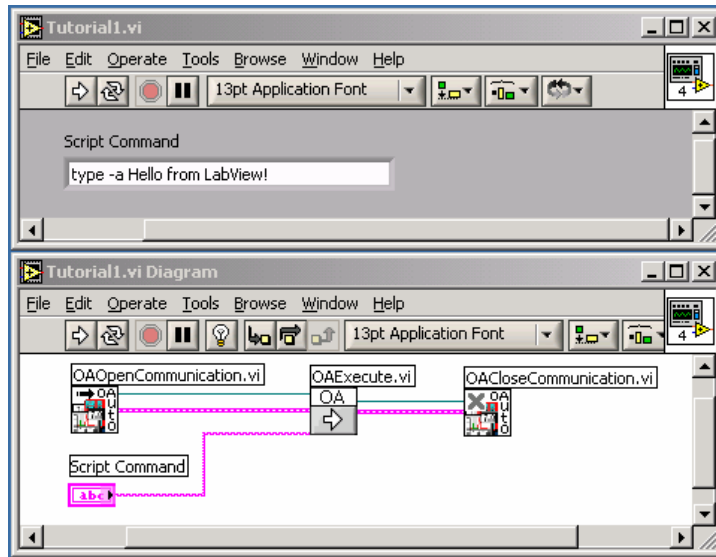
The OAEExecute VI looks like:

8. Wire the two VI's together.
9. Wire the “Script Command” string control to “script” node of the OAEExecute VI.
10. Right-click inside the window and then select the OACloseCommunication VI from the Origin VI palette (as in step 6). Place this VI next to the OAEExecute VI and connect. The Automation The



OACloseCommunications VI looks like:

Your final Front Panel and Block Diagram windows should look like:



11. Go to the Front Panel window and click the Run button. If you then switch to the Origin application, you will see the text “Hello from LabVIEW!” in the script window. You can change the script in the String Control and try sending more script commands.

Note: If an instance of Origin was not running at the time you ran the code from LabVIEW, then a hidden instance of Origin is launched, the script command is sent over, and then that instance of Origin is closed.

Tutorial 2 – Sending data from LabVIEW to an Origin Worksheet

In this tutorial, we will set up a VI to send data from LabVIEW to an Origin worksheet.

1. Launch Origin. If Origin is already running, start a new project. Make sure there is a Data1 worksheet available.
2. Launch LabVIEW, and then start a new VI.
3. Activate the Block Diagram window.
4. Right click inside the window, and from the Functions palette, choose User Libraries, and then paletteMenu. Then choose OAOpenCommunication.vi and place it in the window.



The icon for this VI is

5. Right-click in the window again, and this time select OAPutWorksheet.vi from the same palette as in the previous step.



The icon for this VI is

6. Wire the two VI's.
7. Add a string constant and fill it with the string “Data1”.

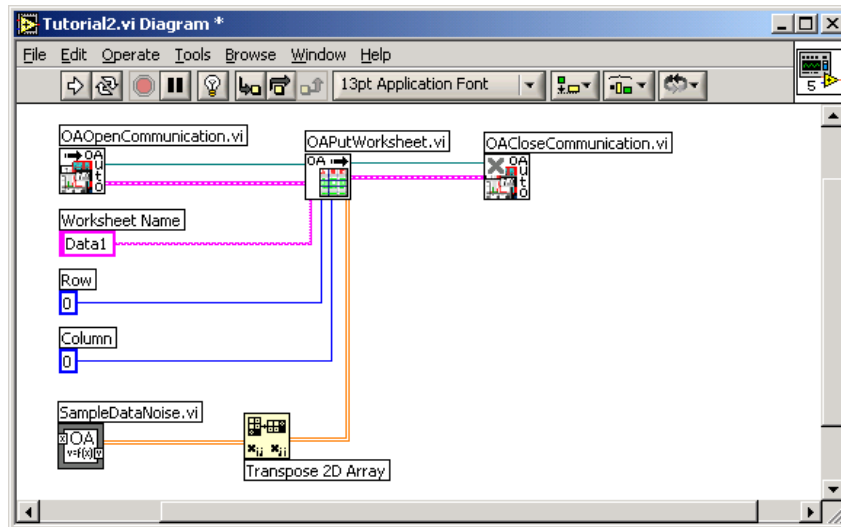
8. Add two numeric constants and set them both to 0.
9. Wire the string constant to the “name” connector and the two constants to the “r1” and “c1” connectors of the OAPutWorksheet VI.
Hint: You can click on any Origin VI you place in the window, and then you can go to the Help menu and select Show Context Help. This will bring up the context help window that will then show the specifics of the connectors on the VI.
10. Right-click in the window, then select the Origin palette (as in step 4), and add the VI named SampleDataNoise.VI.



The icon for this VI is

This VI just uses a formula to create a 2xn array of x and y value where y is some noise data.

11. The VI placed in the step above needs to be transposed before sending it to an Origin worksheet. Right-click next to this VI, and from the Functions palette select Array, and then Transpose 2D Array. Add this VI, and then connect the two.
12. Take the output of Transpose 2D Array and connect to the OAPutWorksheet VI.
13. Add an OACloseCommunications VI from the Origin VI palette next to the OAPutworksheet VI and wire them together.
14. Once all the wiring has been done, your Block Diagram window should look like below:

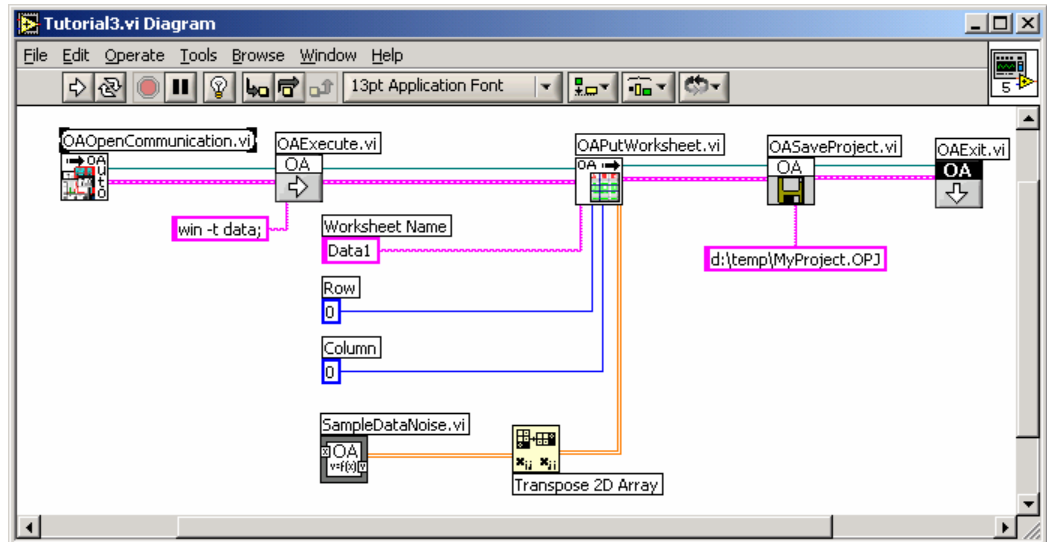


15. Click the Run button to send the data over to Origin. You should see the data in the Data1 worksheet. Try changing the Row and Column number controls to place data in a different location in the worksheet.
16. Save this VI to disk. We will modify this VI in the next tutorial.

Tutorial 3 – Saving an Origin project from LabVIEW

In this tutorial, we will launch Origin in hidden mode from LabVIEW, then place some data into an Origin worksheet, and then save the Origin project for future use.

1. Close all instances of Origin.
2. In LabVIEW, start where the previous tutorial ended.
3. Change the block diagram by introducing two more VI's, and replacing the last VI as shown below:



The two VI's that were added are OAEExecute and OASaveProject. The first one sends a LabTalk script command (`win -t data;`) to create a new worksheet. The second VI saves the project to a user-specified location (`d:\temp\MyProject.OPJ` in this example – you may want to change to your desired path).

Since we close all instances of Origin before running this VI, a new hidden instance of Origin is launched when this VI is run. We need to make sure that instance is terminated when the VI finishes executing. For this reason, the last VI in the previous tutorial, OACloseCommunication, has been replaced with OAEExit.

4. Run this VI. Then launch Origin and browse to the OPJ file location and open it. The data pushed to Origin will be found in the Data1 worksheet.

OASetLTStr.VI		Set value of LabTalk string variable
OAGetLTVar.VI		Get value of LabTalk numeric variable
OASetLTVar.VI		Set value of LabTalk numeric variable
OAGetPageString.VI		Get textual properties of specified Origin page (such as label of a worksheet page)
OASetPageString.VI		Set textual properties of specified Origin page (such as label of a worksheet page)
OA2DArrayToCluster.VI		Convert 2D array from GetWorksheet or GetMatrix VI's to cluster
OAVariant2DToArrayOrCluster.VI		Convert variant output of GetMatrix VI to 2D array or cluster
SampleDataNoise.VI		Create sample x-y array with noise data – not related to Origin – provided for testing purposes

OAOpenCommunication.vi

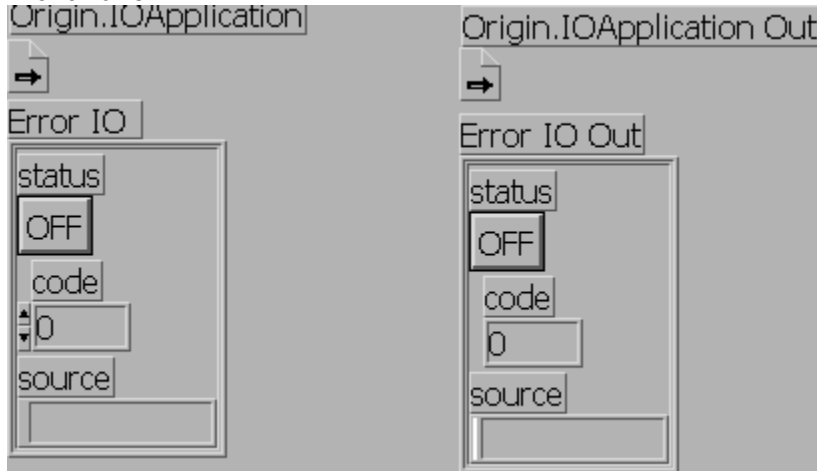
Call to Automation Open LabVIEW function to open communication with Origin.ApplicationSI if no error on input.

This SubVI (or its functionality) must be present in any project using the rest of the samples.

Connector Pane



Front Panel



Controls and Indicators

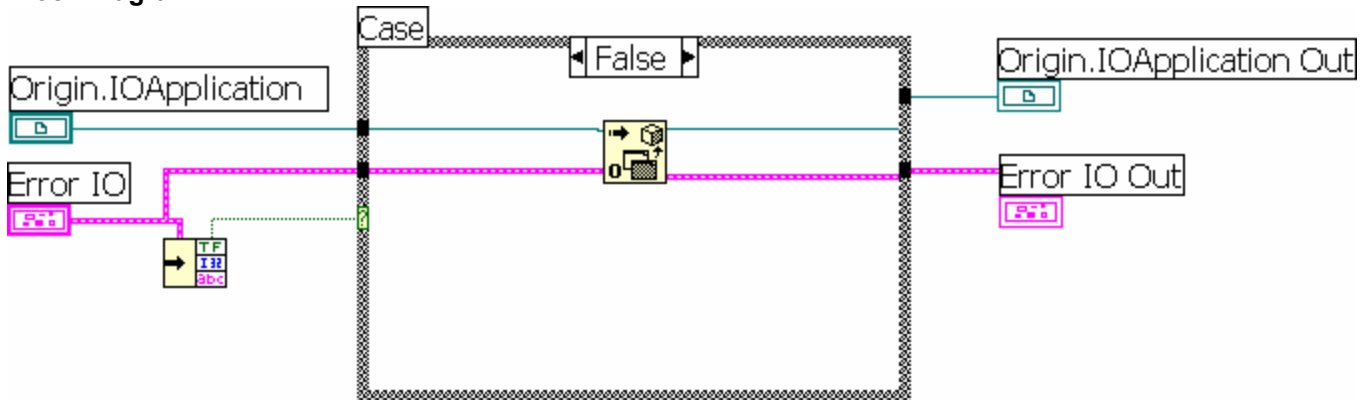
Origin.IOApplication

Error IO

Error IO Out

Origin.IOApplication Out

Block Diagram



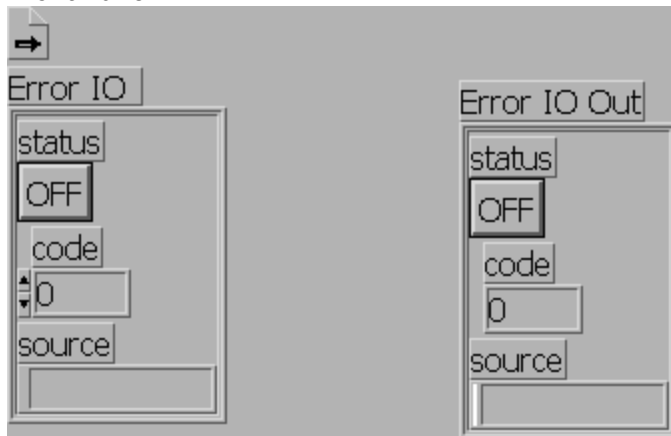
OACloseCommunication.vi

Call to Automation Close LabVIEW function if no error on input. Uses Origin.ApplicationSI.

Connector Pane



Front Panel



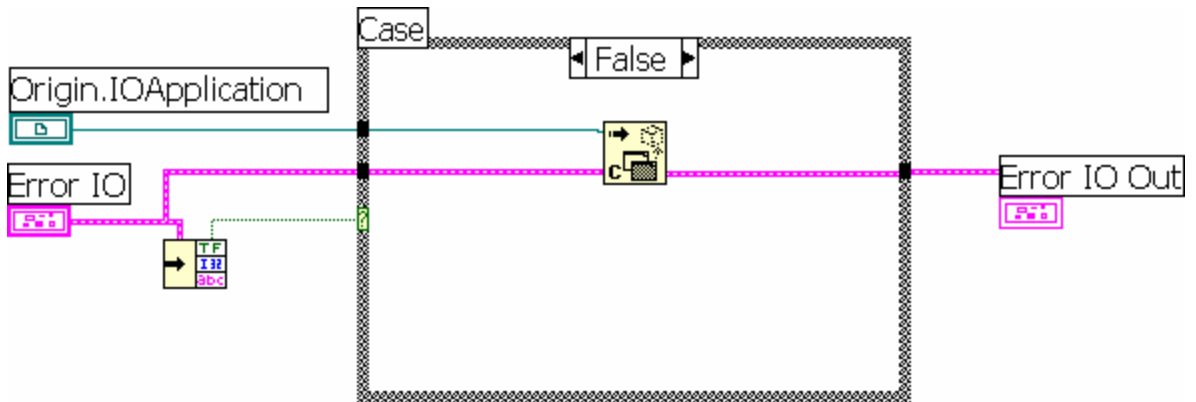
Controls and Indicators

 Origin.IOApplication

 Error IO

 Error IO Out

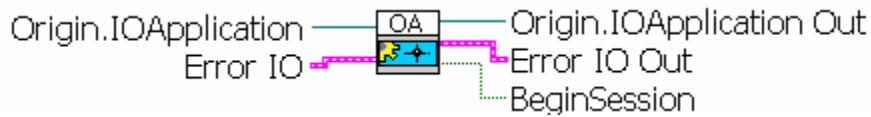
Block Diagram



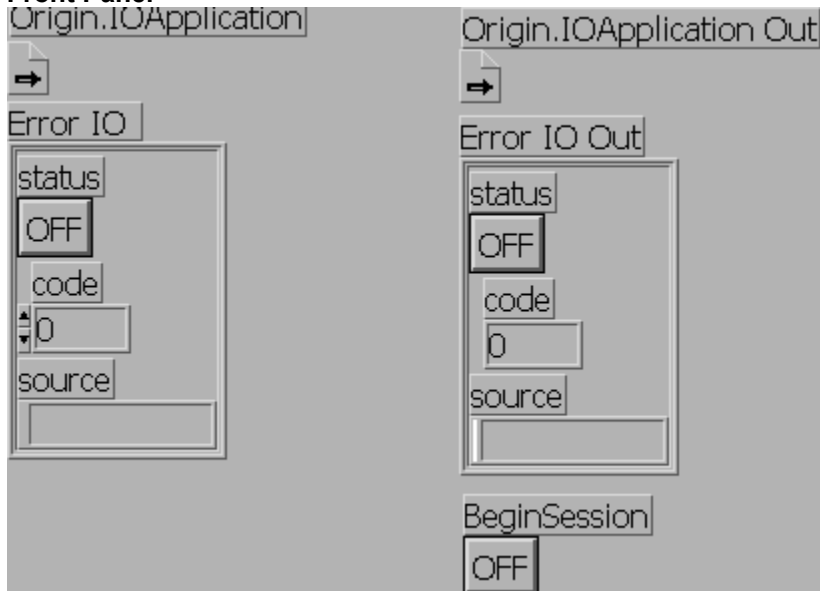
OABeginSession.vi

Call to Origin Automation Server method BeginSession if no error on input.
Uses Origin.ApplicationSI

Connector Pane



Front Panel



Controls and Indicators

Origin.IOApplication

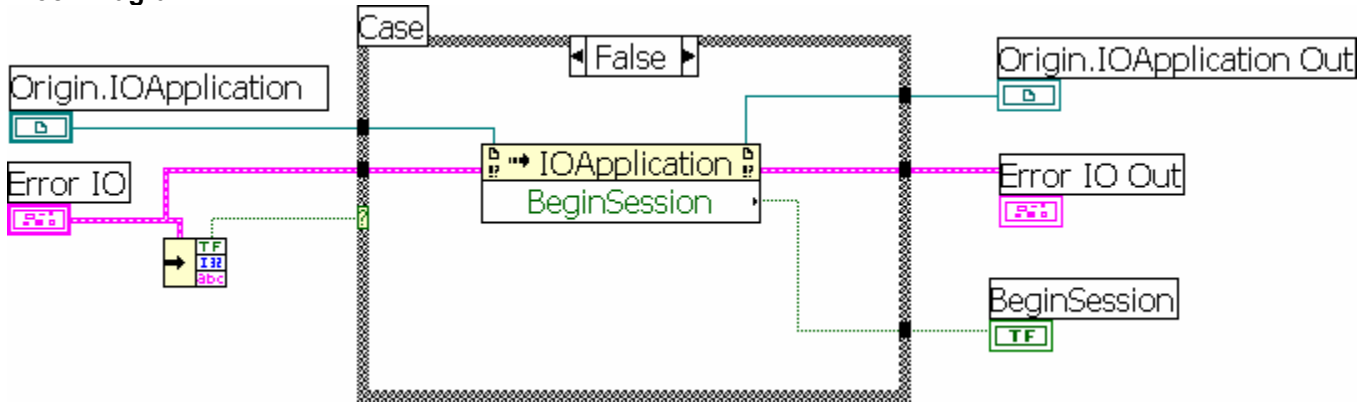
Error IO

BeginSession

 Error IO Out

 Origin.IOApplication Out

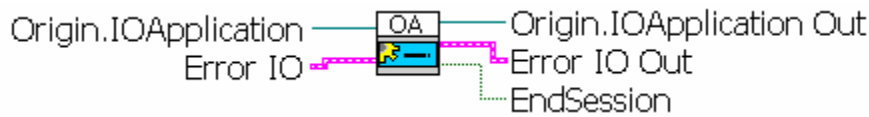
Block Diagram



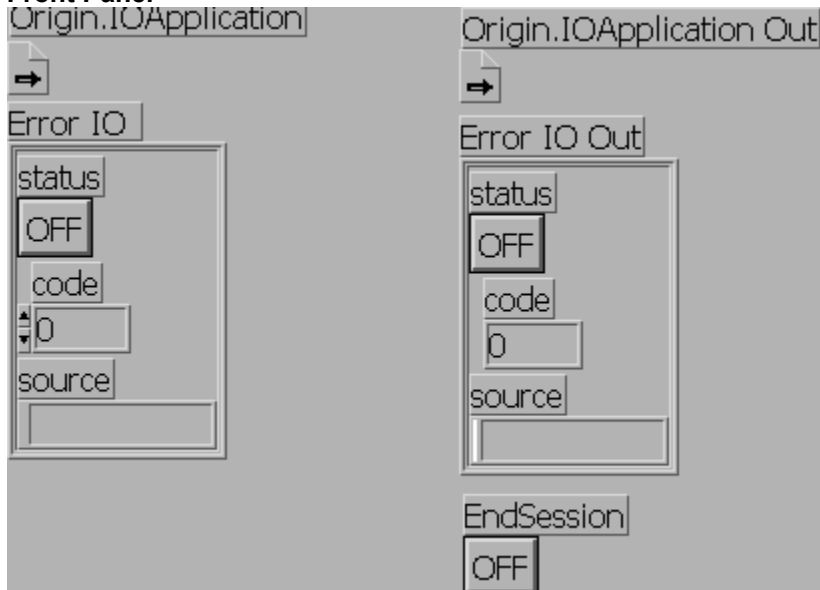
OAEndSession.vi

Call to Origin Automation Server method EndSession if no error on input.
Uses Origin.ApplicationSI






Connector Pane



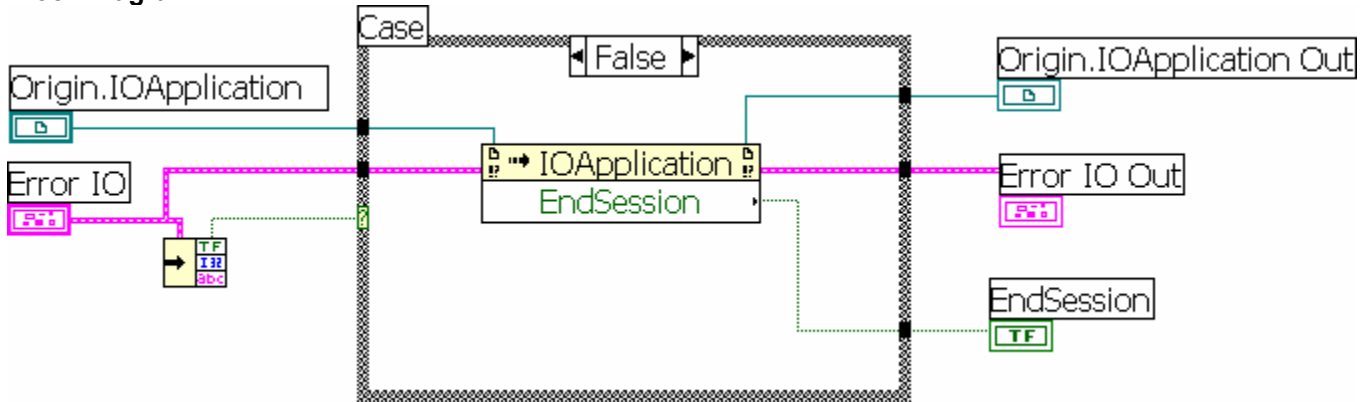
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  EndSession
-  Error IO Out
-  Origin.IOApplication Out

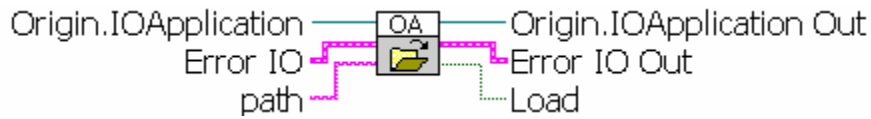
Block Diagram



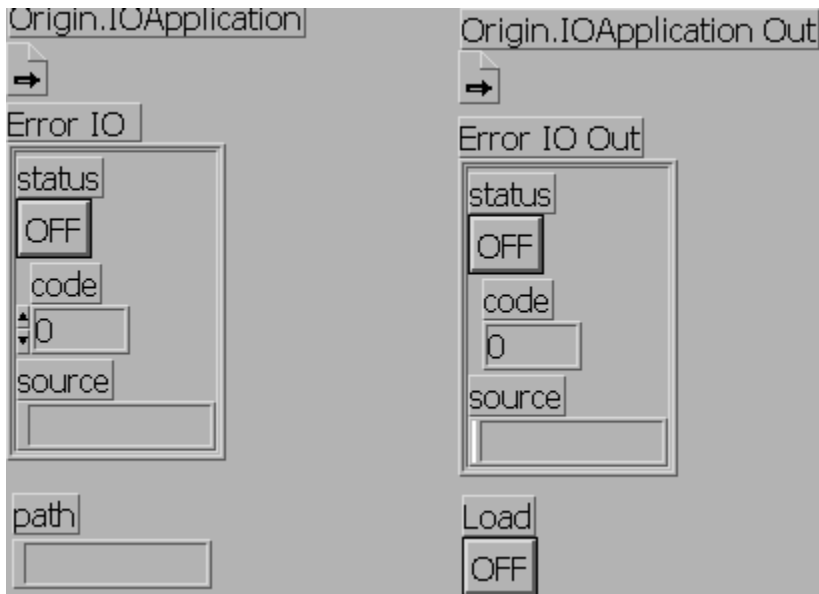
OALoadProject.vi

Call to Origin Automation Server method Load if no error on input.
Uses Origin.ApplicationSI.

Connector Pane



Front Panel



Controls and Indicators

 Origin.IOApplication

 Error IO

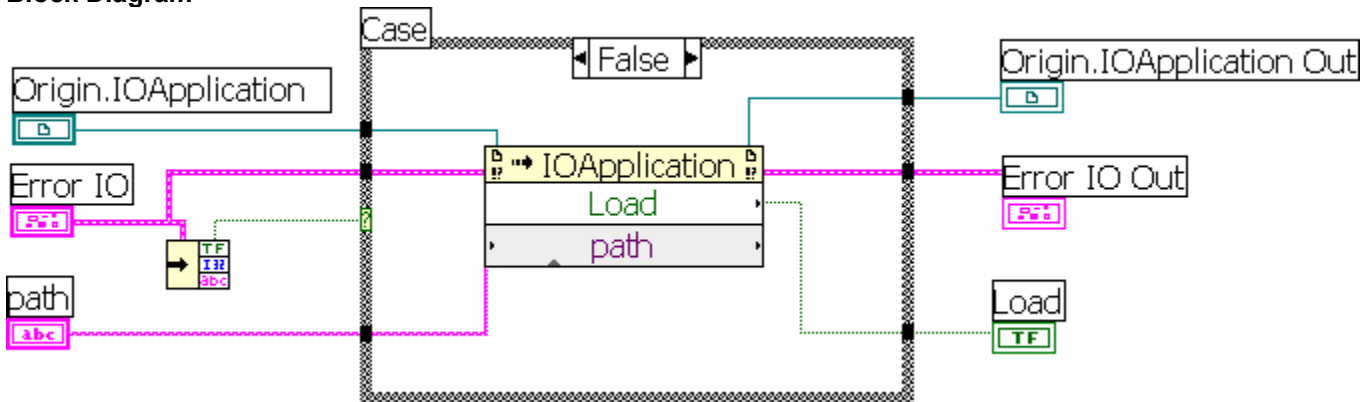
 path

 Load

 Error IO Out

 Origin.IOApplication Out

Block Diagram

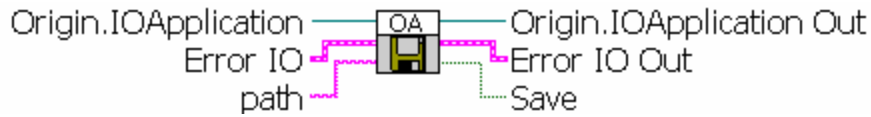


OASaveProject.vi

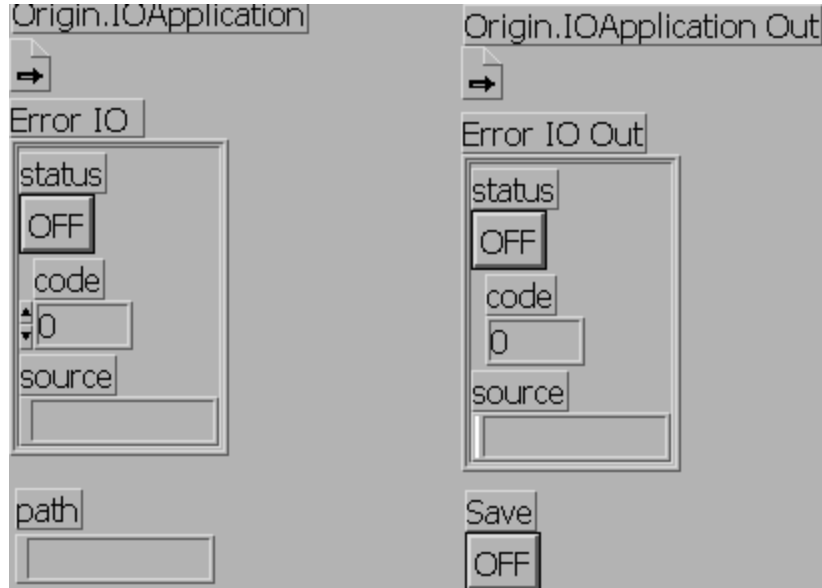
Call to Origin Automation Server method Save if no error on input. Save the project currently in Origin's memory.

Uses Origin.ApplicationSI







Connector Pane



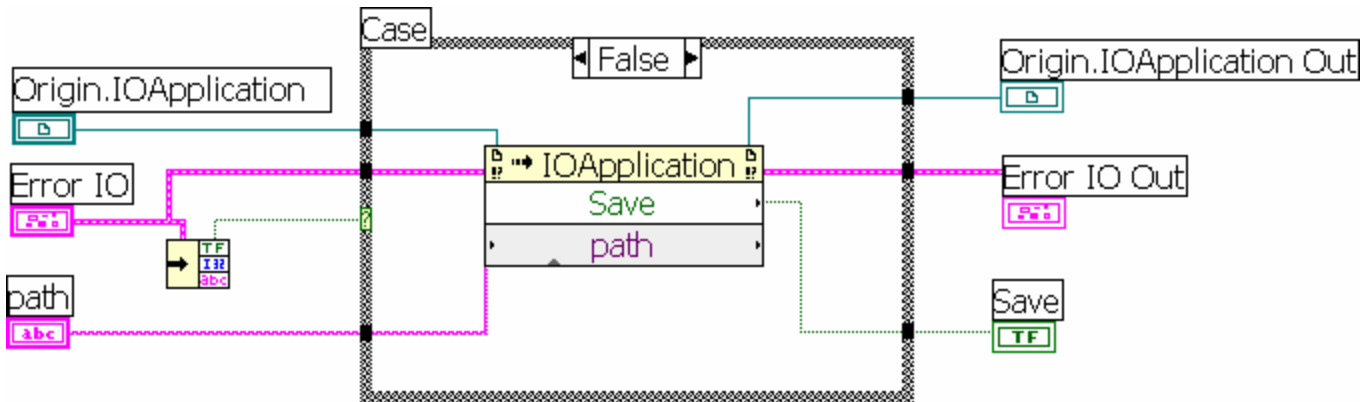
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  path
-  Save
-  Error IO Out
-  Origin.IOApplication Out

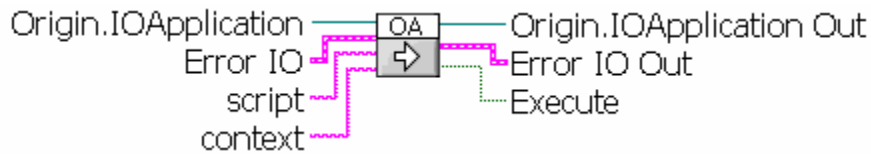
Block Diagram



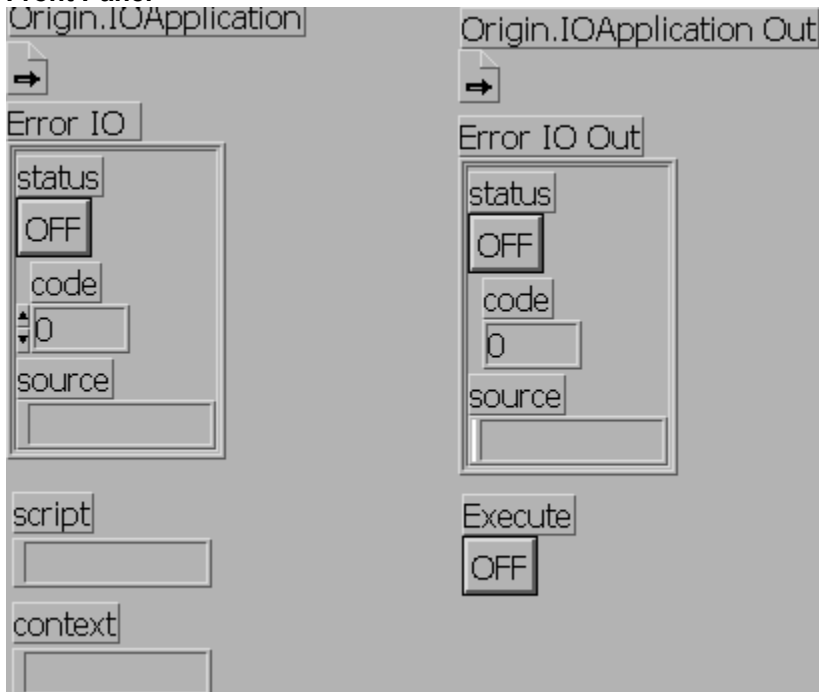
OAEExecute.vi

Call to Origin Automation Server method Execute if no error on input.
Uses Origin.ApplicationSI

Connector Pane








Front Panel



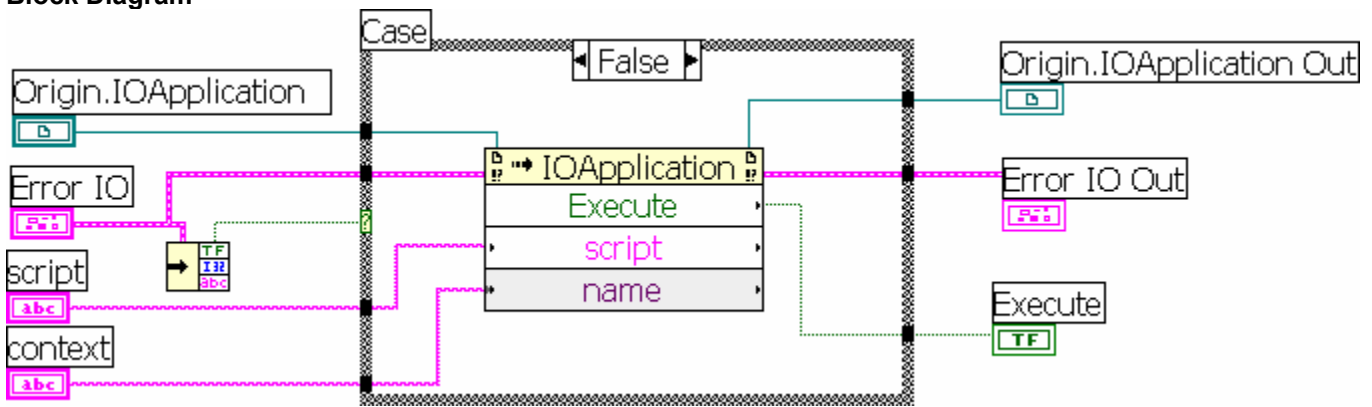
Controls and Indicators

 Origin.IOApplication

 Error IO

-  script
-  context
-  Execute
-  Error IO Out
-  Origin.IOApplication Out

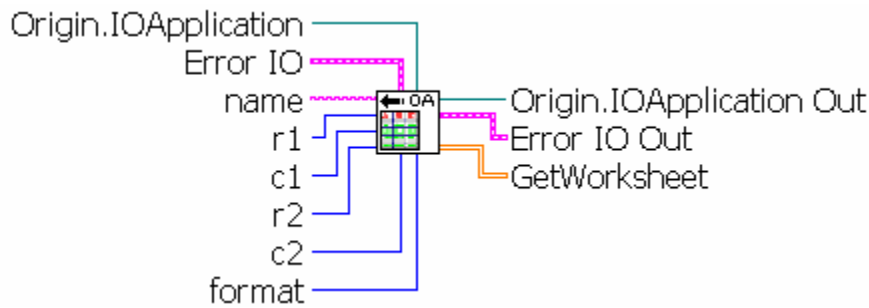
Block Diagram



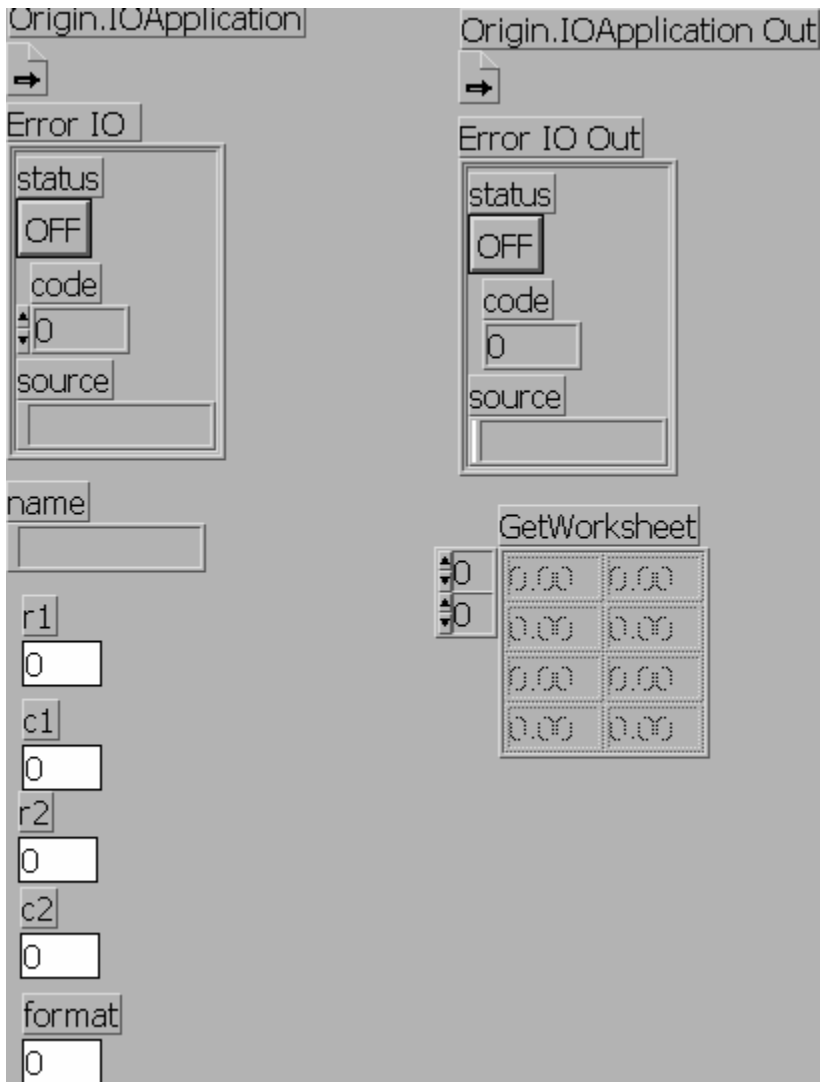
OAGetWorksheet.vi

Call to Origin Automation Server method GetWorksheet if no error on input.
Uses Origin.ApplicationSI.

Connector Pane



Front Panel



Controls and Indicators

 Origin.IOApplication

 Error IO

 name

 r1

 c1

 r2

 c2

 format

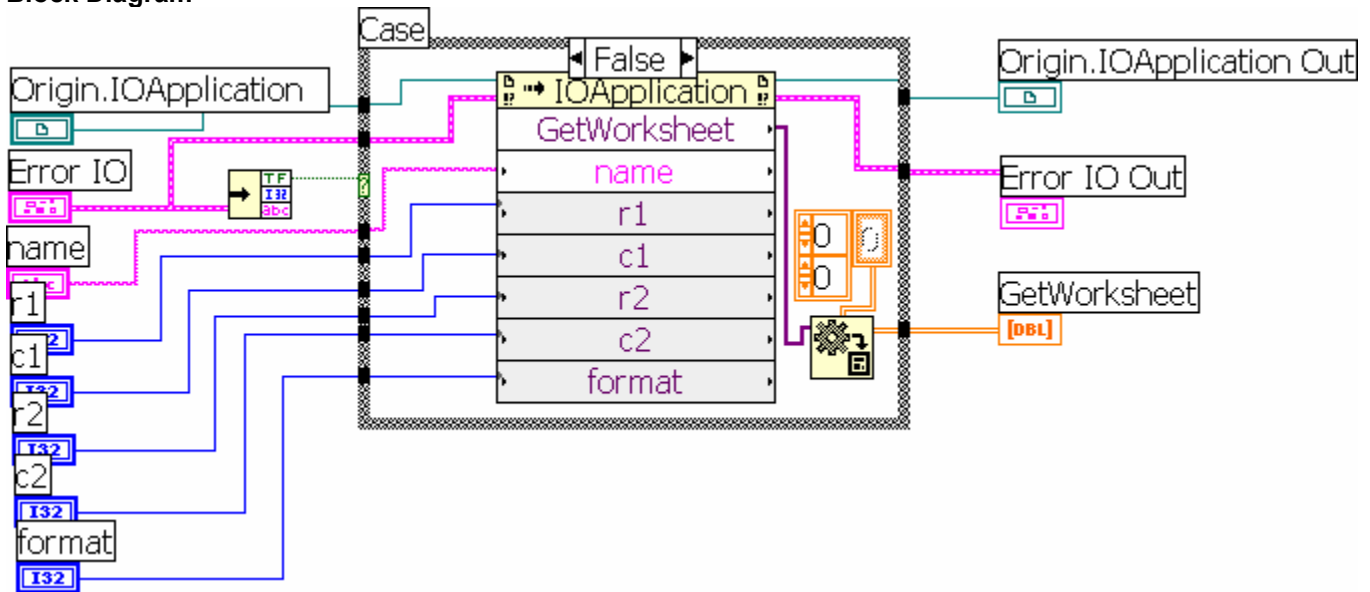
 Error IO Out

 Origin.IOApplication Out

 GetWorksheet

 DBL

Block Diagram

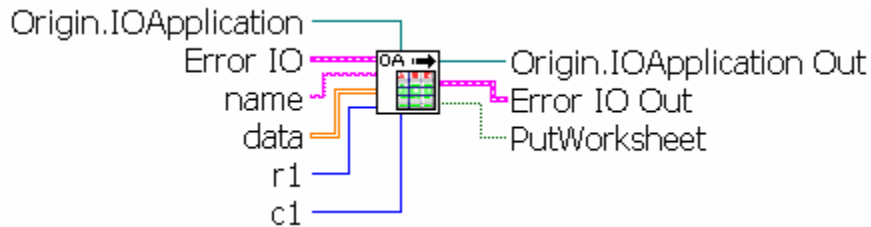


OAPutWorksheet.vi

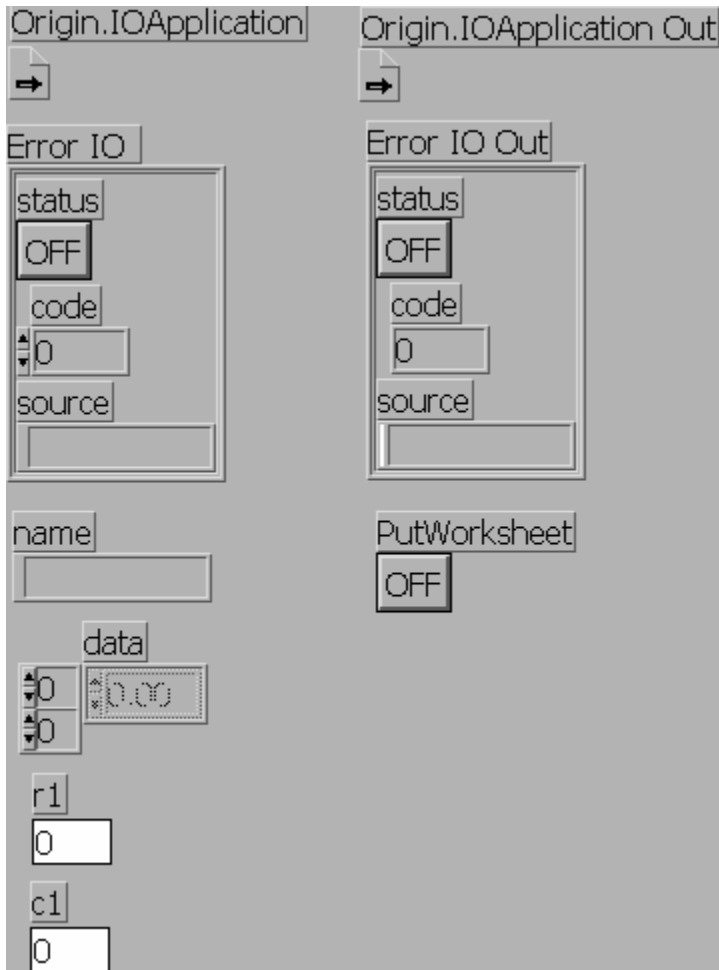
Call to Origin Automation Server method PutWorksheet if no error on input. 2D array is passed to Origin.

Uses Origin.ApplicationSI.

Connector Pane



Front Panel



Controls and Indicators



data



Origin.IOApplication



Error IO



name



r1



c1



Error IO Out

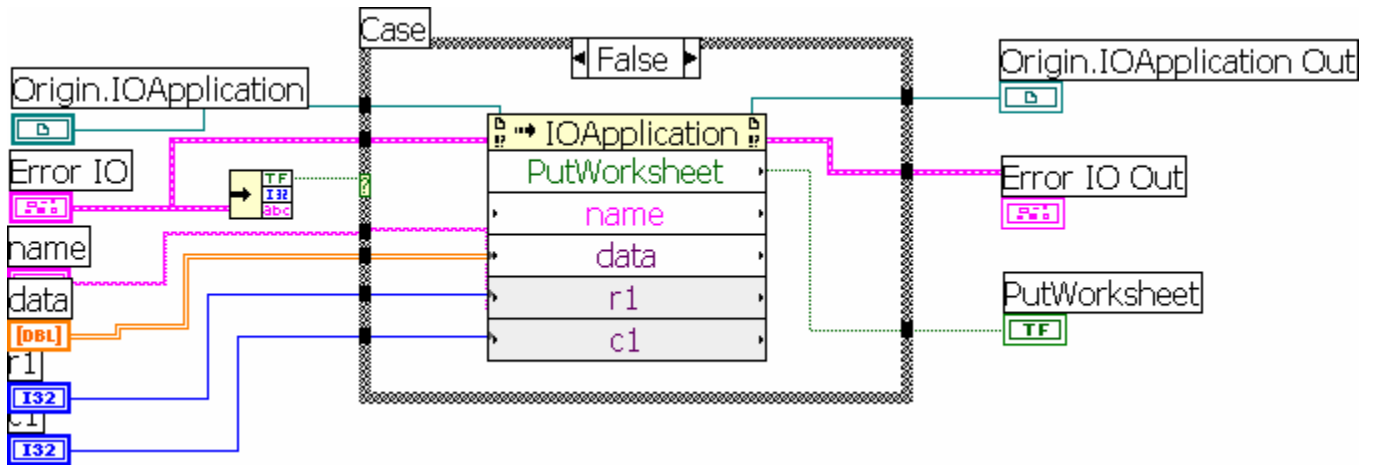


Origin.IOApplication Out



PutWorksheet

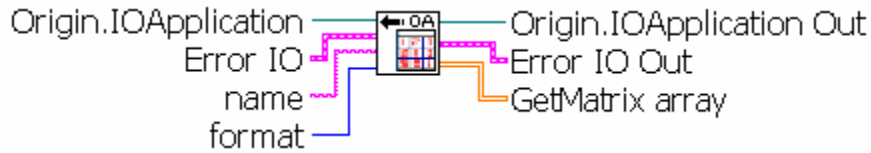
Block Diagram



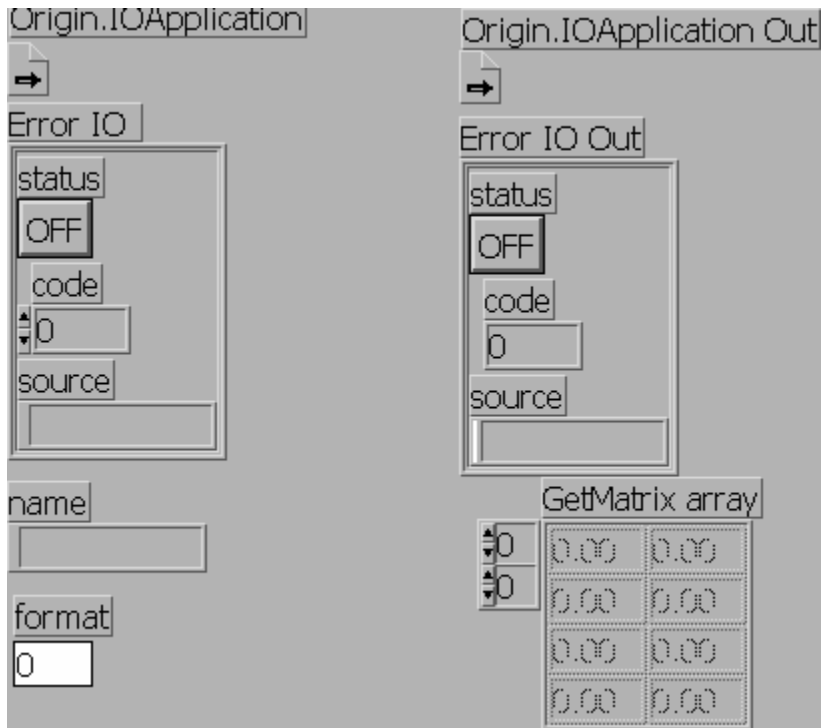
OAGetMatrix.vi

Call to Origin Automation Server method GetMatrix if no error on input.
Uses Origin.ApplicationSI.









Connector Pane



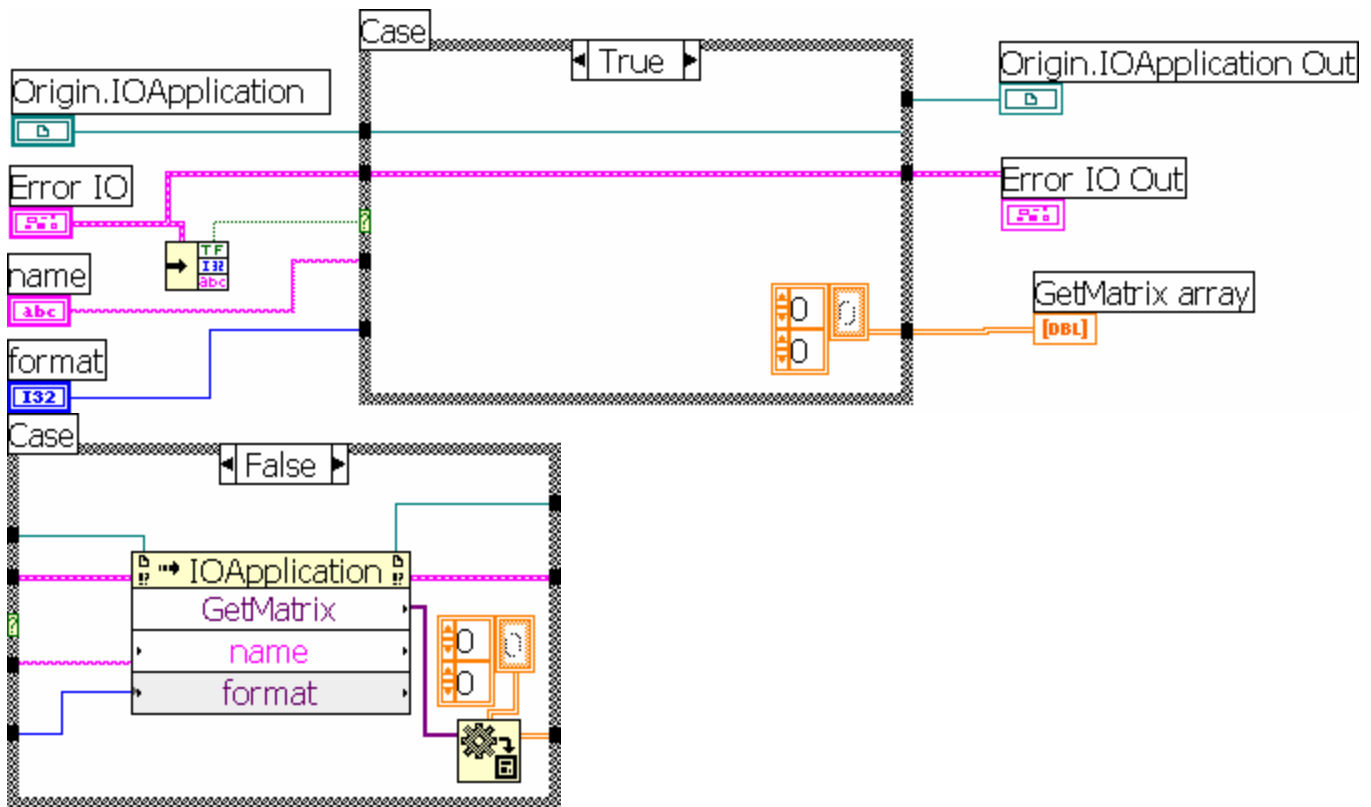
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  name
-  format
-  Error IO Out
-  Origin.IOApplication Out
-  GetMatrix array
-  DBL

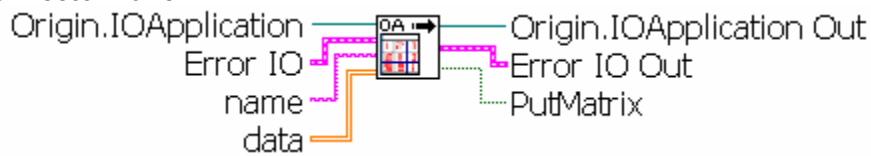
Block Diagram



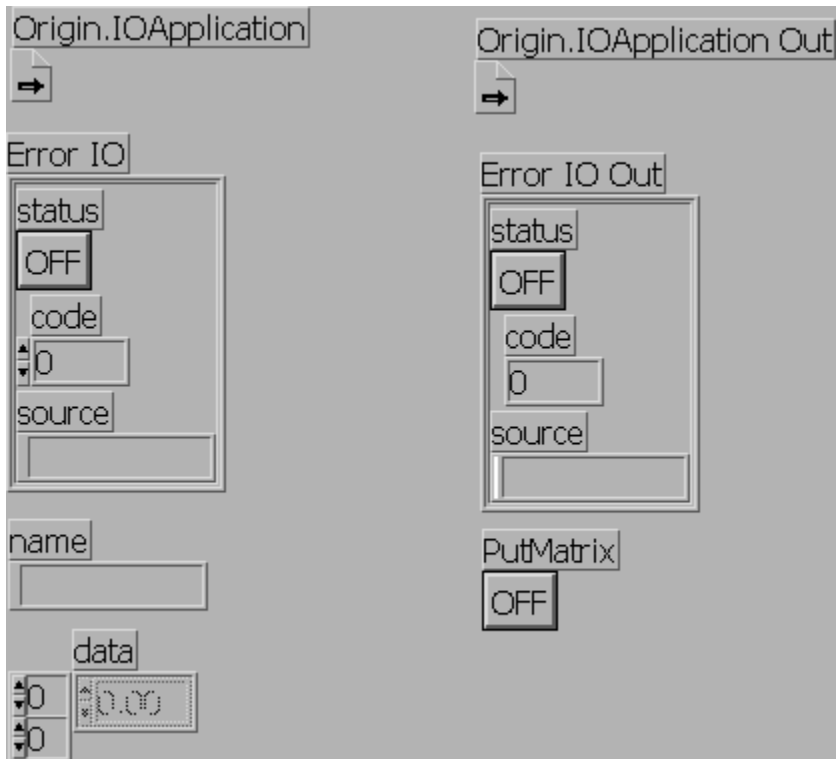
OAPutMatrix.vi

Call to Origin Automation Server method PutMatrix if no error on input. 2D array passed to Origin
 Uses Origin.ApplicationSI.

Connector Pane



Front Panel



Controls and Indicators



Error IO



Origin.IOApplication



name



data



Origin.IOApplication Out

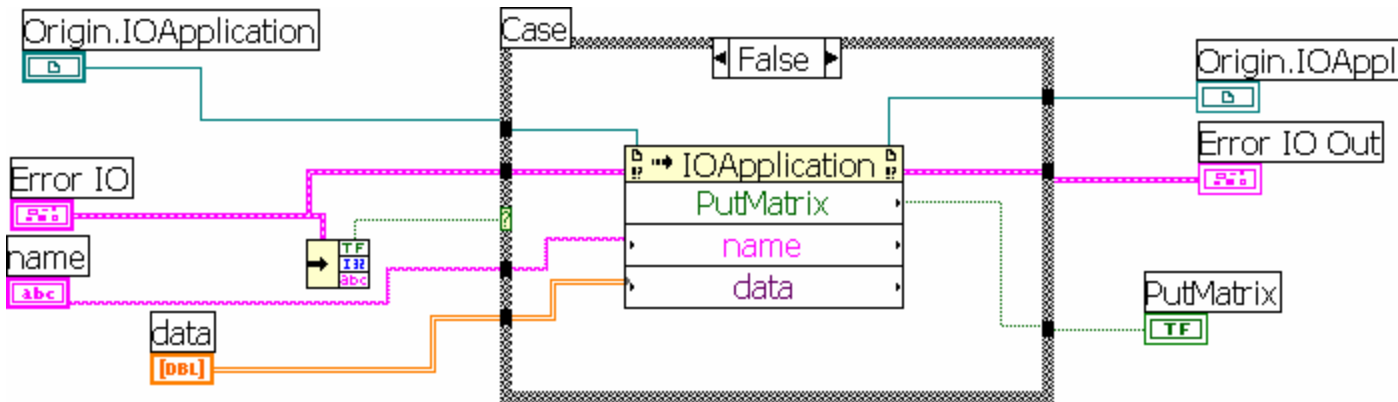


Error IO Out



PutMatrix

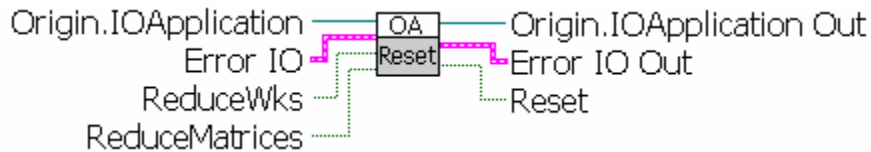
Block Diagram



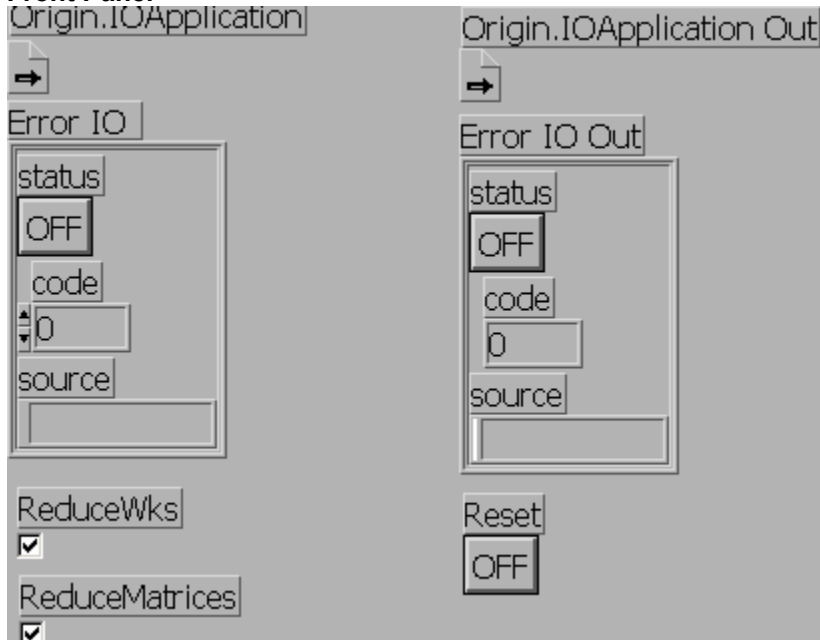
OAResetProject.vi

Call to Origin Automation Server method Reset if no error on input.
Uses Origin.ApplicationSI.

Connector Pane








Front Panel



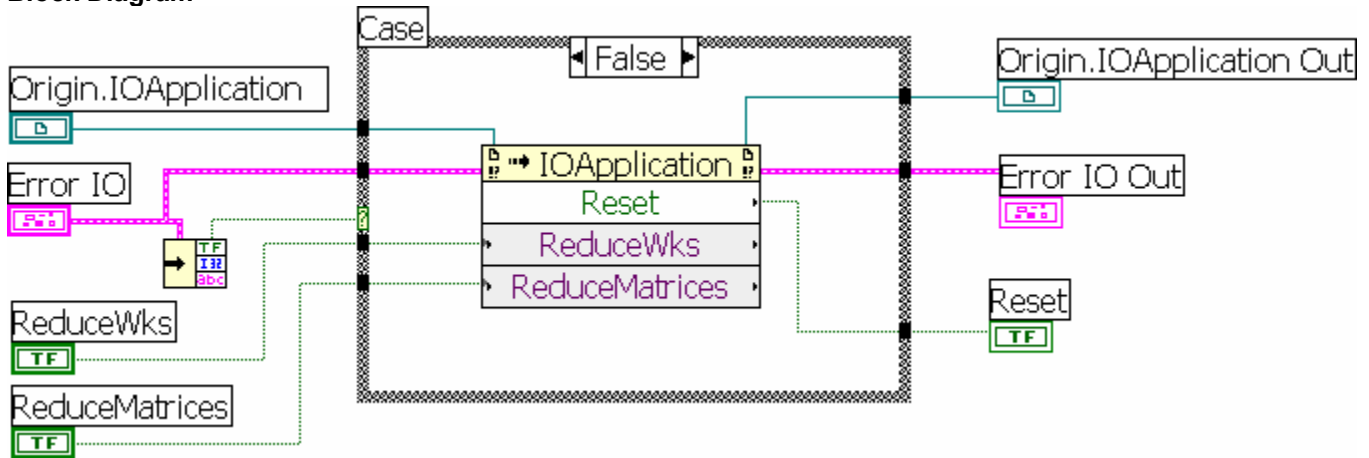
Controls and Indicators

 **Origin.IOApplication**

 **Error IO**

-  ReduceWks
-  ReduceMatrices
-  Reset
-  Error IO Out
-  Origin.IOApplication Out

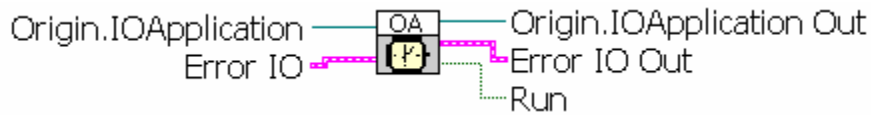
Block Diagram



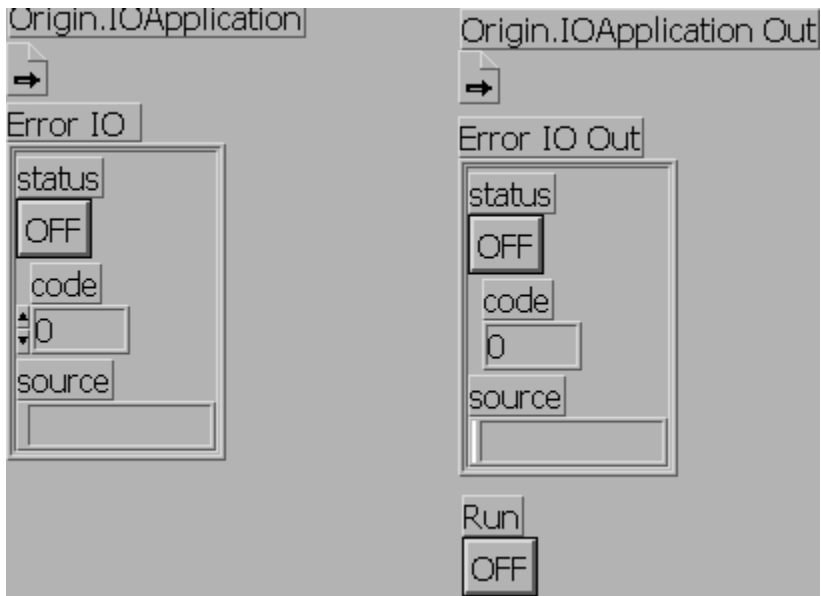
OARun.vi

Call to Origin Automation Server method Run if no error on input.
 Uses Origin.ApplicationSI.






Connector Pane



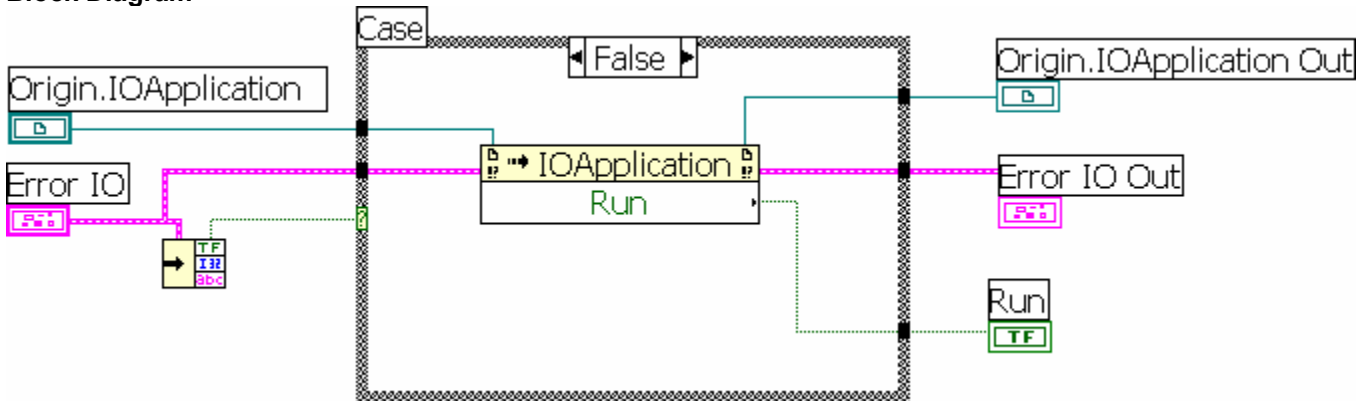
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  Run
-  Error IO Out
-  Origin.IOApplication Out

Block Diagram

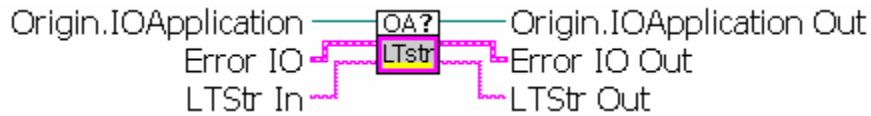


Origin Automation Get/Set Methods and Properties

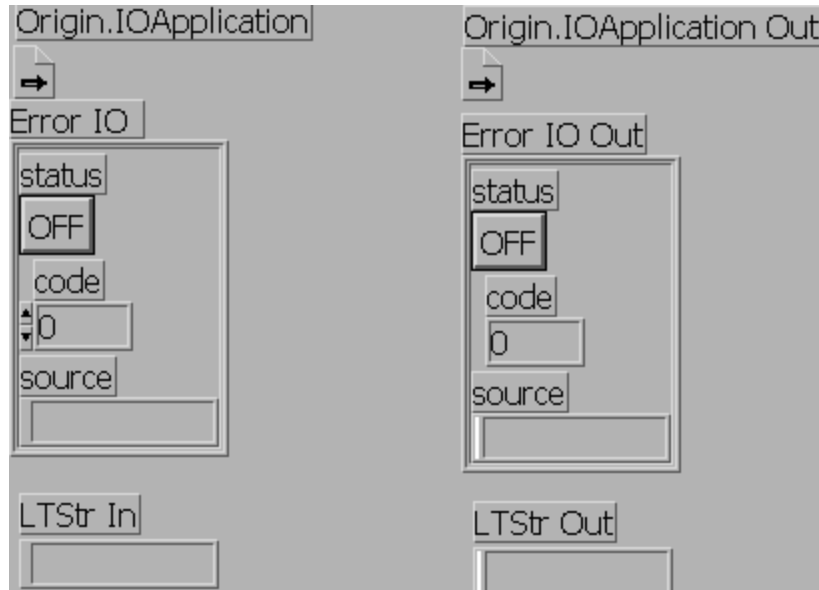
OAGetLTStr.vi

Read Origin Automation Server property LTStr if no error on input.
Uses Origin.ApplicationSI.







Connector Pane



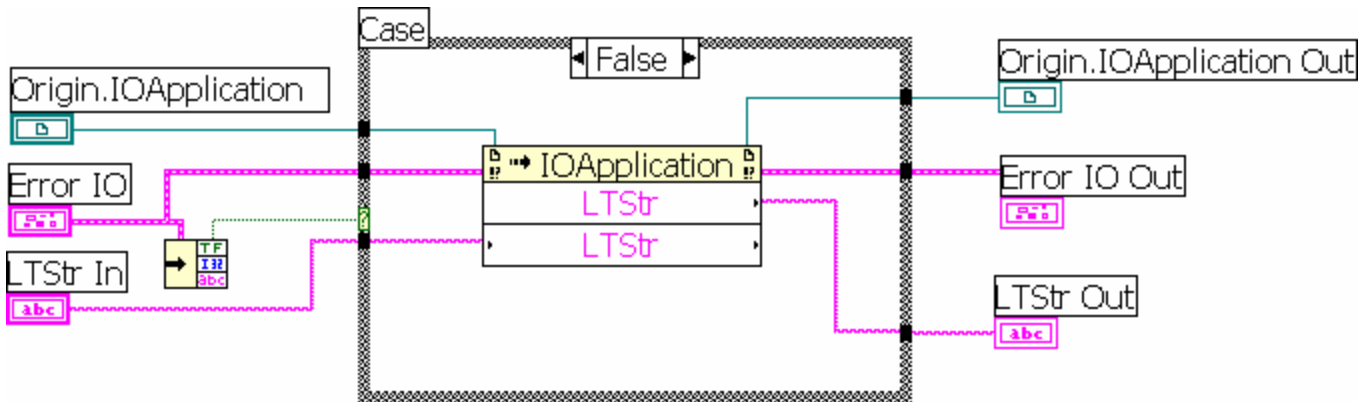
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  LTStr In
-  Error IO Out
-  Origin.IOApplication Out
-  LTStr Out

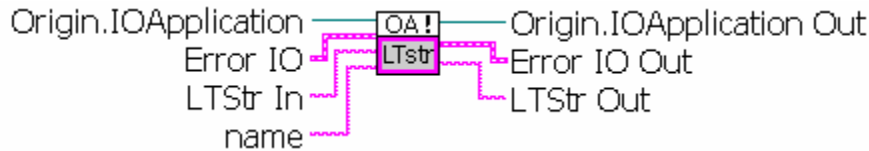
Block Diagram



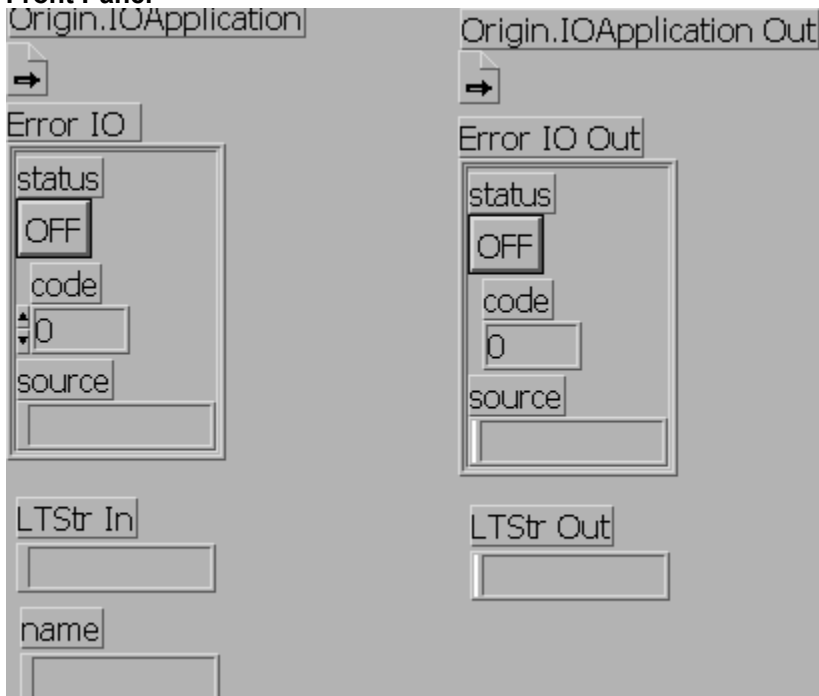
OASetLTStr.vi

Write to Origin Automation Server property LTStr if no error on input.
Uses Origin.ApplicationSI

Connector Pane









Front Panel

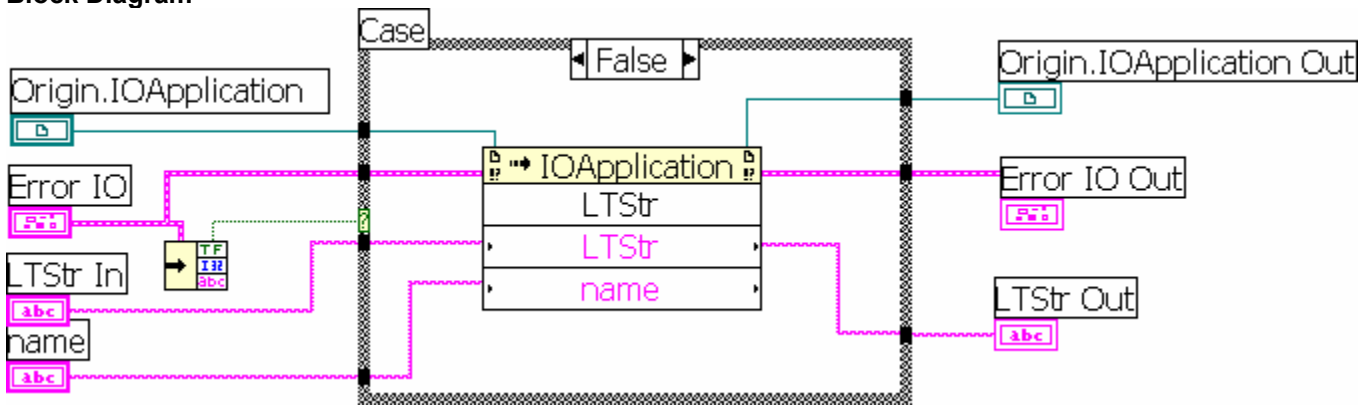


Controls and Indicators

 Origin.IOApplication

-  Error IO
-  name
-  LTStr In
-  Error IO Out
-  Origin.IOApplication Out
-  LTStr Out

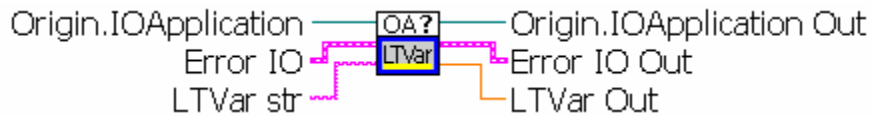
Block Diagram



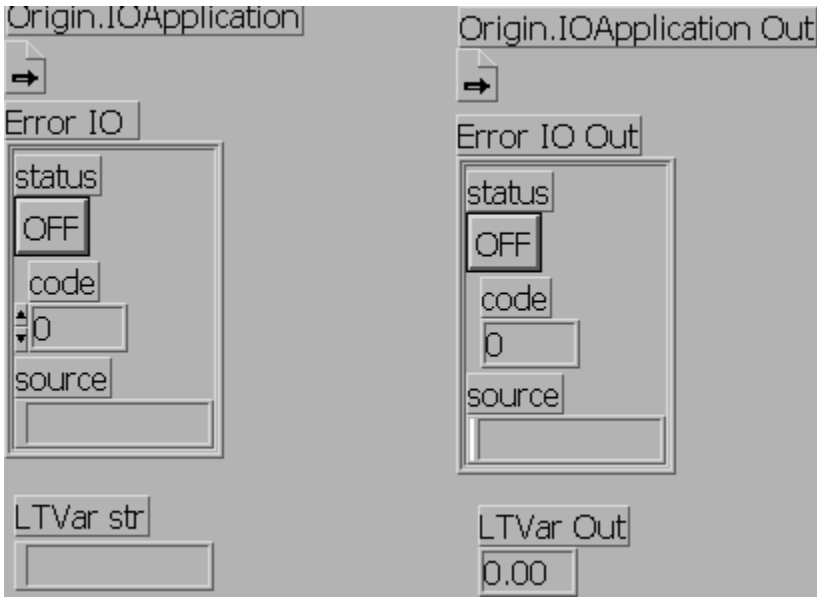
OAGetLTVar.vi

Read Origin Automation Server property LTVar if no error on input.
Uses Origin.ApplicationSI.

Connector Pane



Front Panel



Controls and Indicators

Origin.IOApplication

Error IO

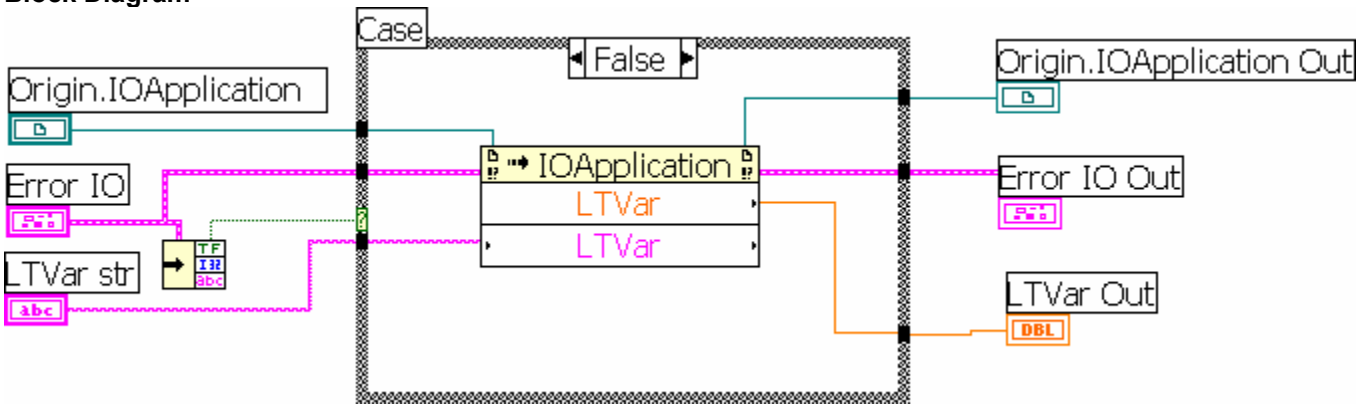
LTVar str

Error IO Out

Origin.IOApplication Out

LTVar Out

Block Diagram



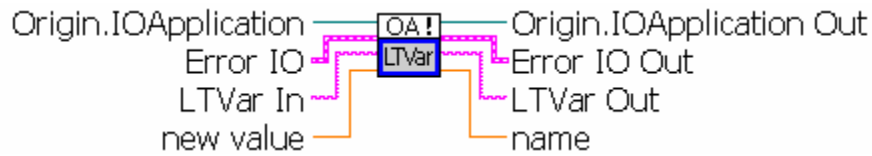
OASetLTVar.vi

Do Not Use in LabVIEW 5.0!

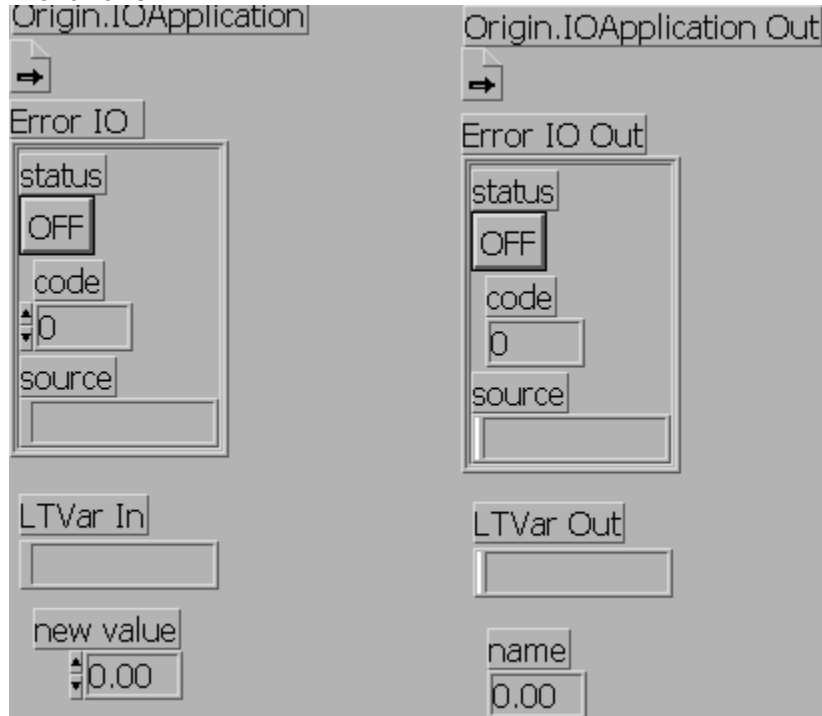
It should write to Origin Automation Server property LTVar if no error on input. But LabVIEW 5.0

does not support this kind of methods/properties with arguments.
 Uses Origin.ApplicationSI.









Connector Pane



Front Panel

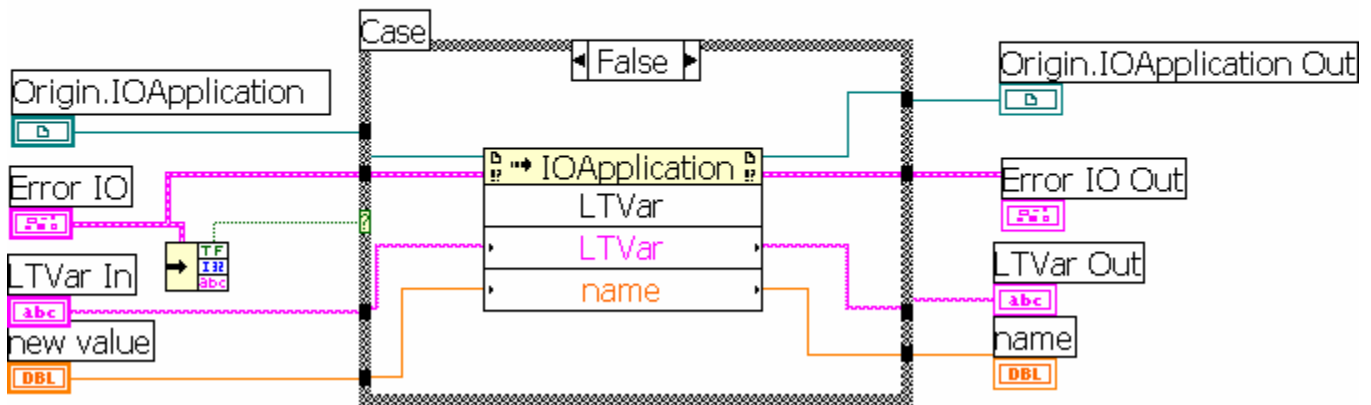


Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  new value
-  LTVar In
-  Error IO Out
-  Origin.IOApplication Out
-  LTVar Out
-  name

Block Diagram

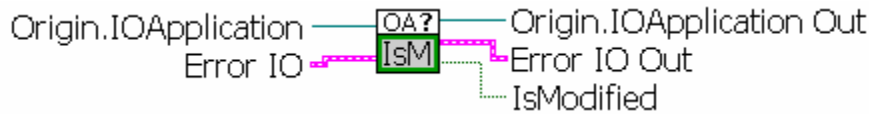
No way to set LTVar in LabVIEW 5.0



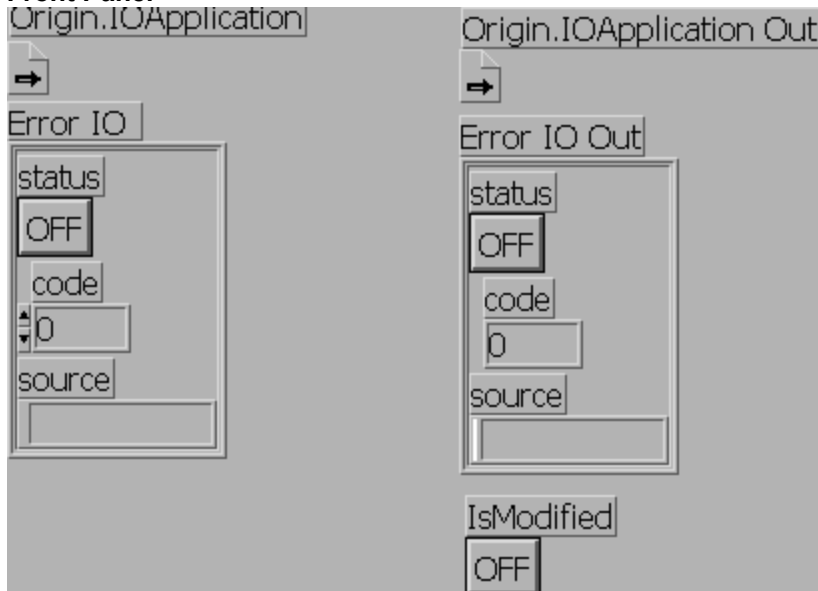
OAGetIsModified.vi

Read Origin Automation Server property IsModified if no error on input. Read IsModified property. query if Origin project is modified. Uses Origin.ApplicationSI.

Connector Pane







Front Panel

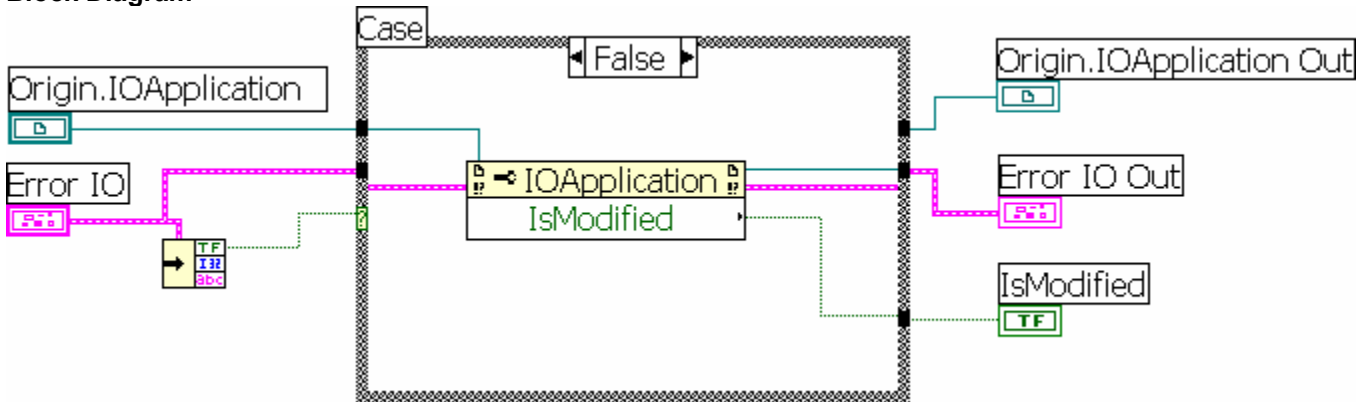


Controls and Indicators

 Origin.IOApplication

-  Error IO
-  Error IO Out
-  Origin.IOApplication Out
-  IsModified

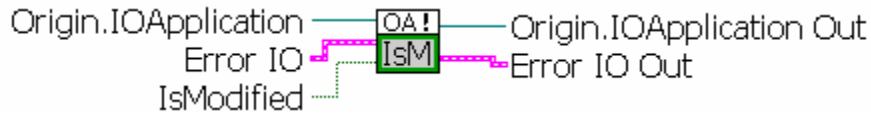
Block Diagram



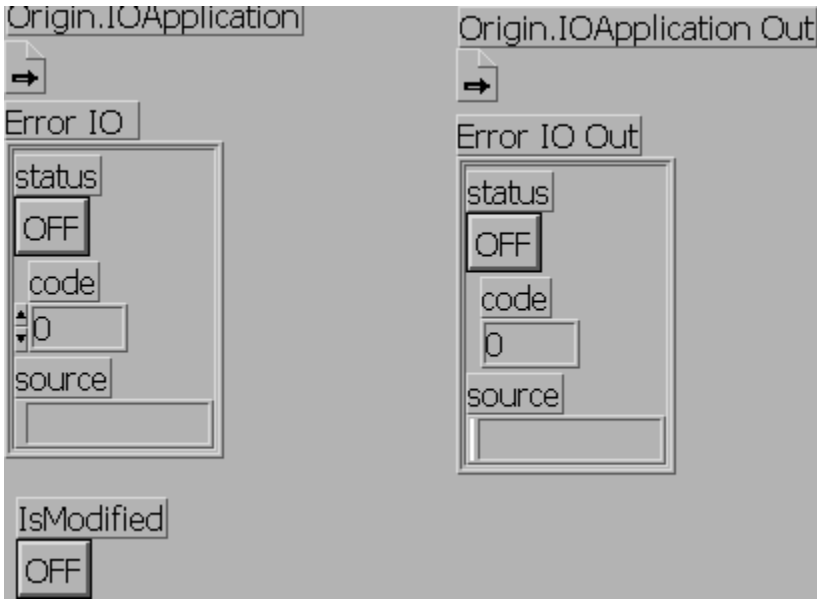
OASetIsModified.vi

Write to Origin Automation Server property IsModified if no error on input.
 Uses Origin.ApplicationSI.






Connector Pane



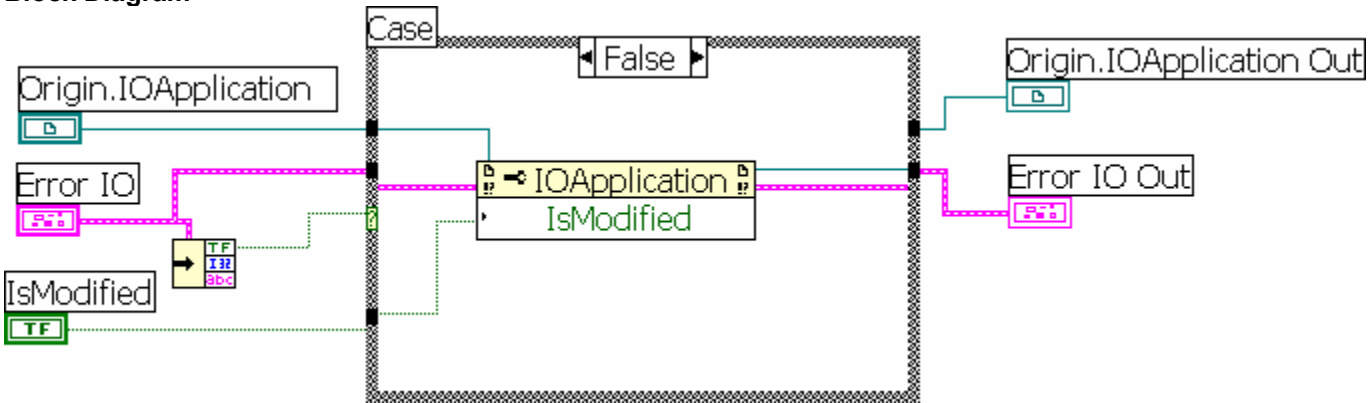
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  IsModified
-  Error IO Out
-  Origin.IOApplication Out

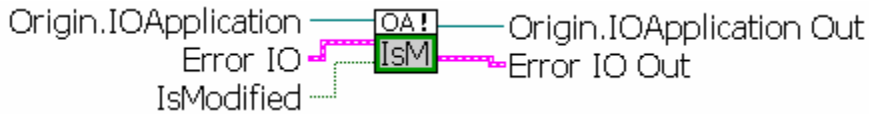
Block Diagram



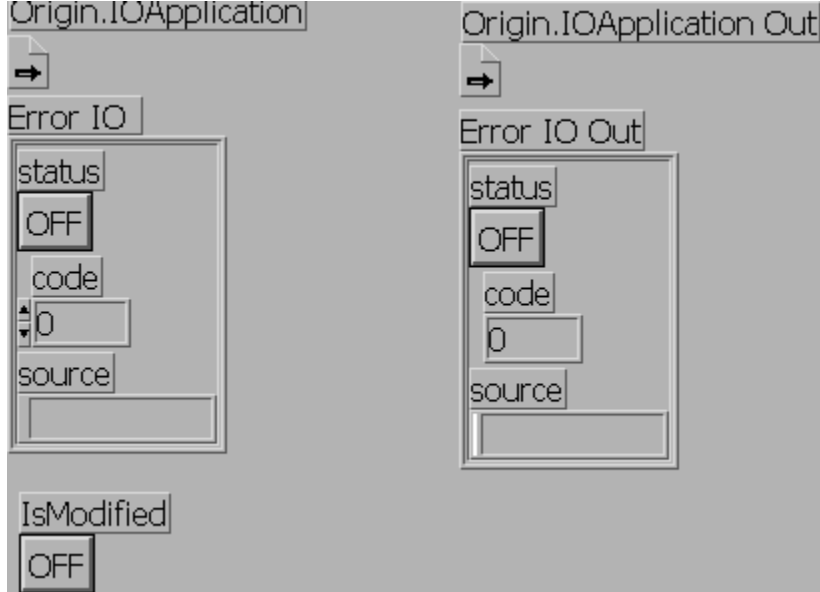
OASetsIsModified.vi

Write to Origin Automation Server property IsModified if no error on input. Set Origin project as modified or clean.
 Uses Origin.ApplicationSI.






Connector Pane



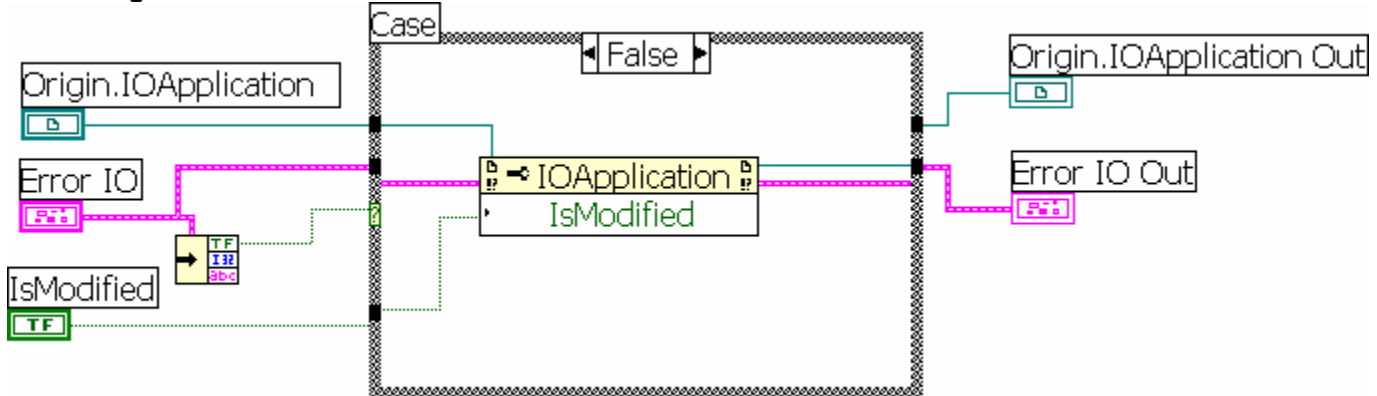
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  IsModified
-  Error IO Out
-  Origin.IOApplication Out

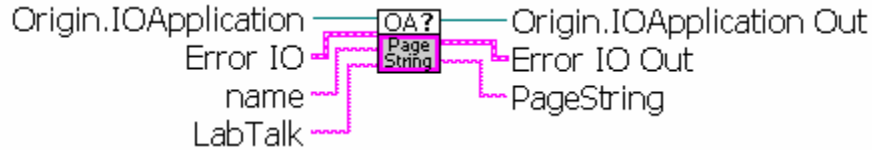
Block Diagram



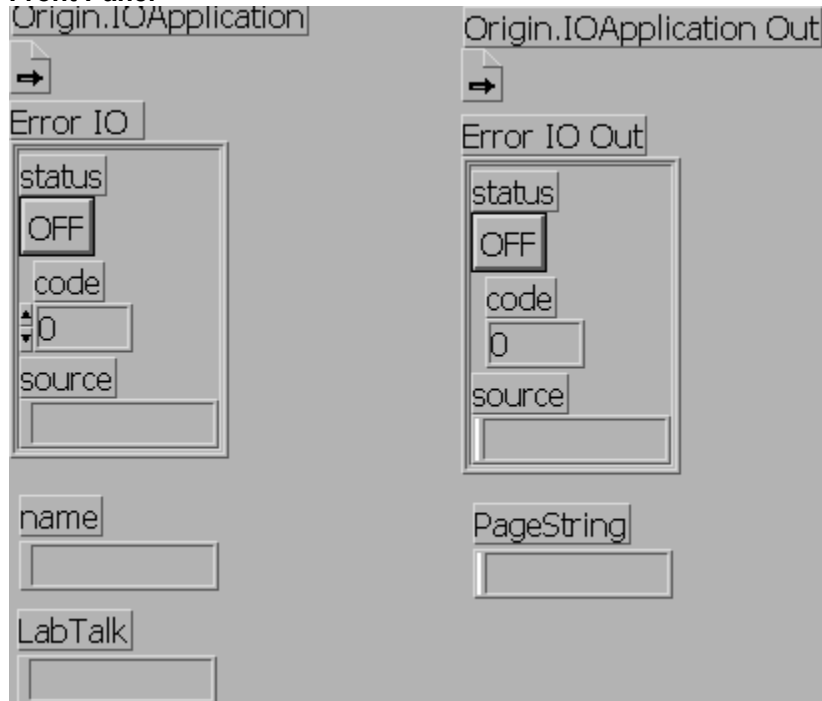
OAGetPageString.vi

Read Origin Automation Server property PageString if no error on input.
Uses Origin.ApplicationSI.








Connector Pane



Front Panel

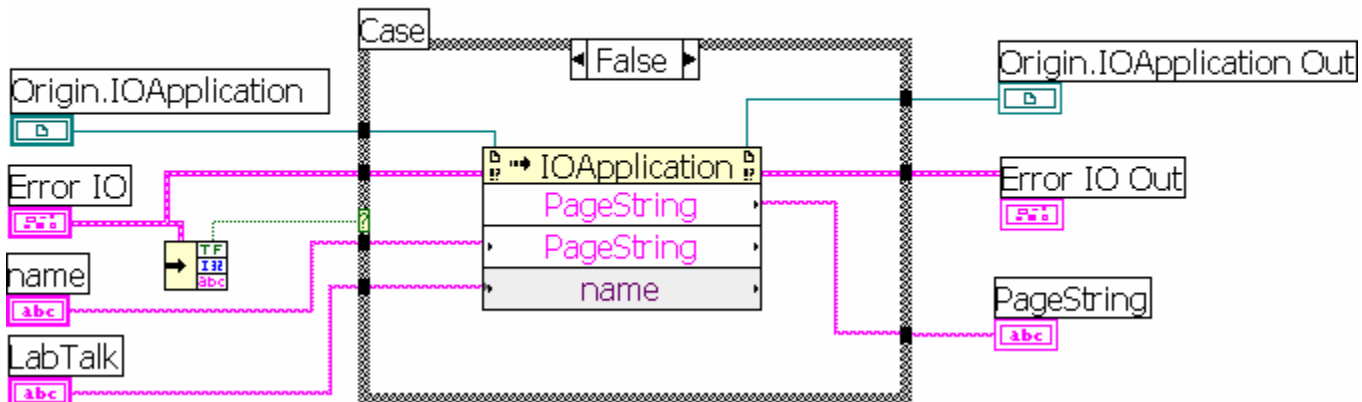


Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  name
-  LabTalk
-  Error IO Out
-  Origin.IOApplication Out
-  PageString

Block Diagram

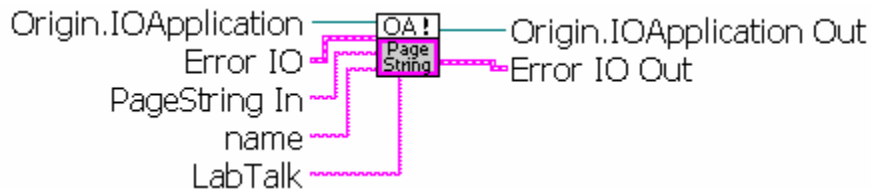
Ignore what you see in Invoke Node function
LabVIEW 5.0 problem with Method/Property



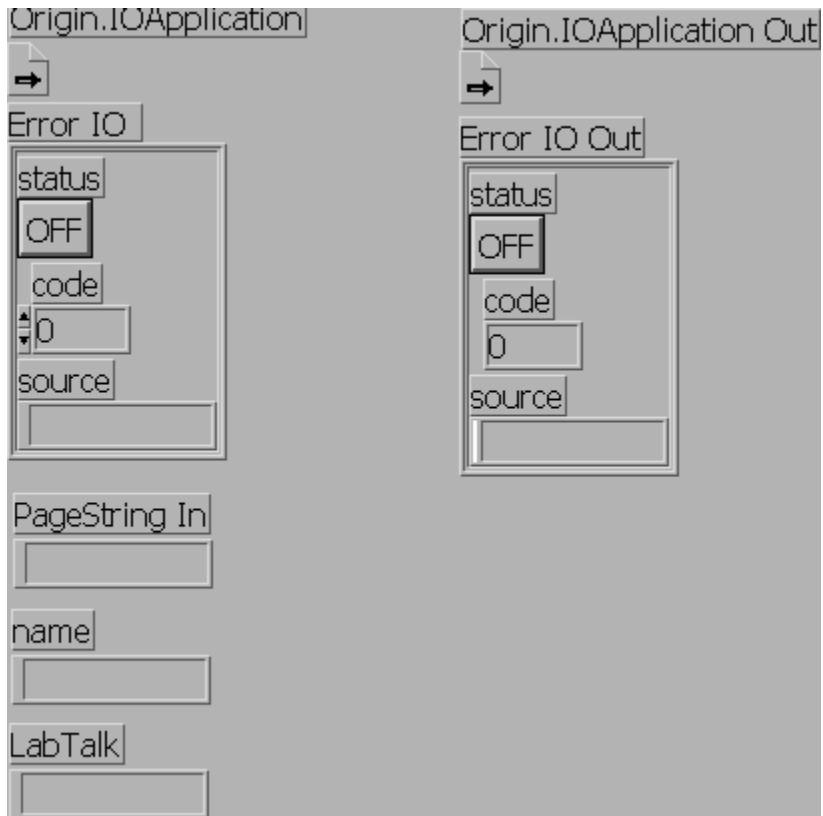
OASetPageString.vi

Write to Origin Automation Server property PageString if no error on input.
Uses Origin.ApplicationSI.








Connector Pane



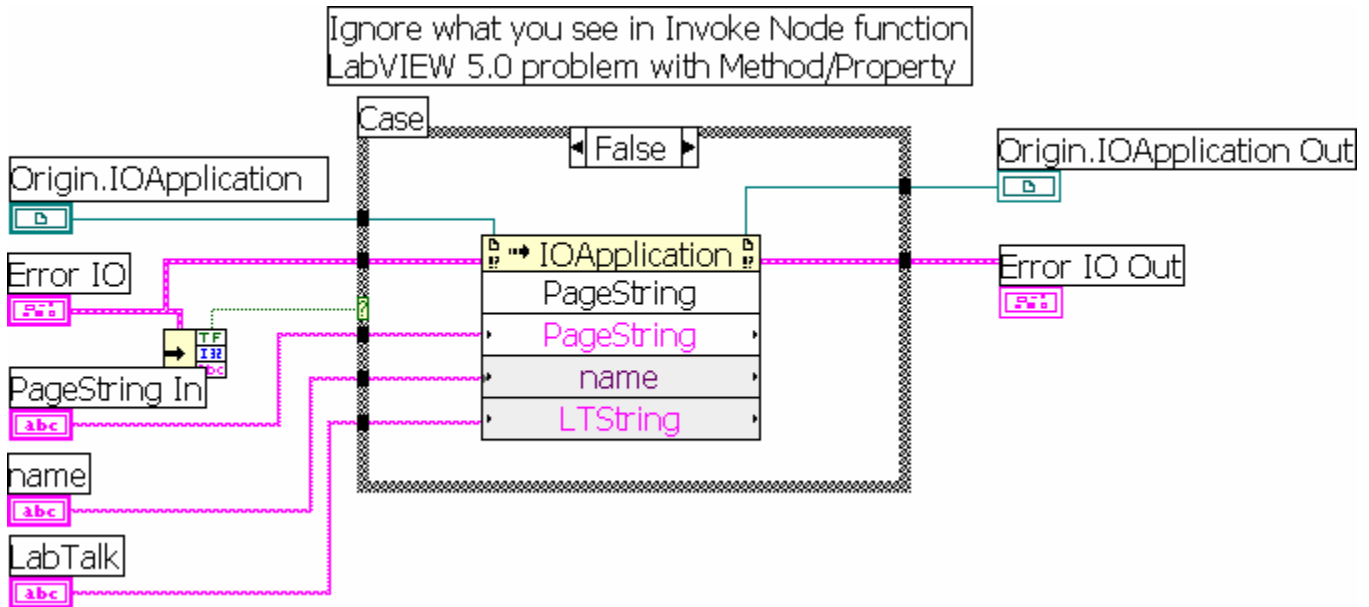
Front Panel



Controls and Indicators

-  **Origin.IOApplication**
-  **Error IO**
-  **name**
-  **LabTalk**
-  **PageString In**
-  **Error IO Out**
-  **Origin.IOApplication Out**

Block Diagram

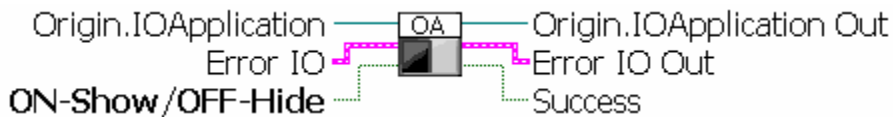


Origin Automation Utilities

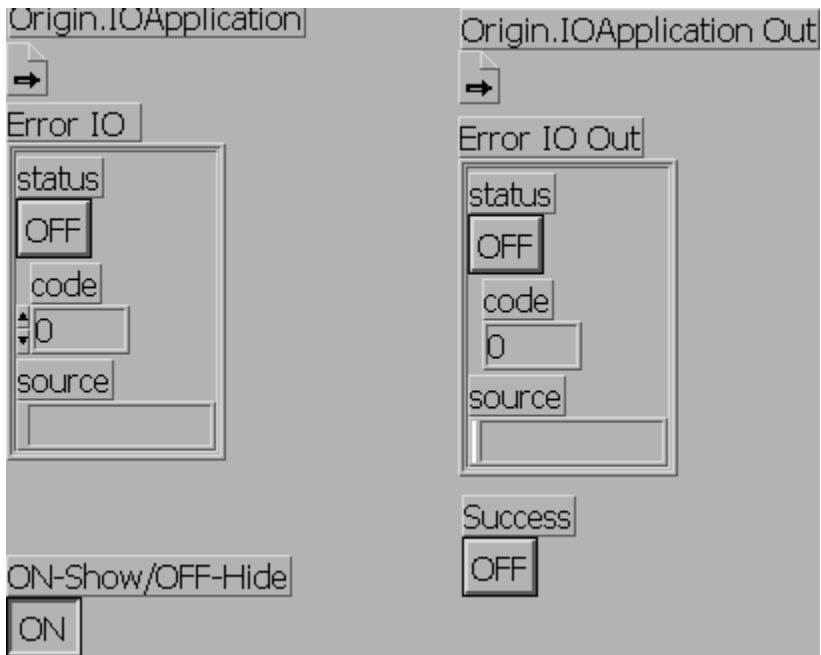
OAShowHide.vi

Shows/Hides Origin using Origin Automation Server method Execute("doc -mc 1/0") if no error on input. LabTalk command Doc -mc 1|0 turns on user control (AfxOleGetUserCtrl) and shows (1) hides (0) Origin.
Uses Origin.ApplicationSI.

Connector Pane



Front Panel



Controls and Indicators

Origin.IOApplication

Error IO

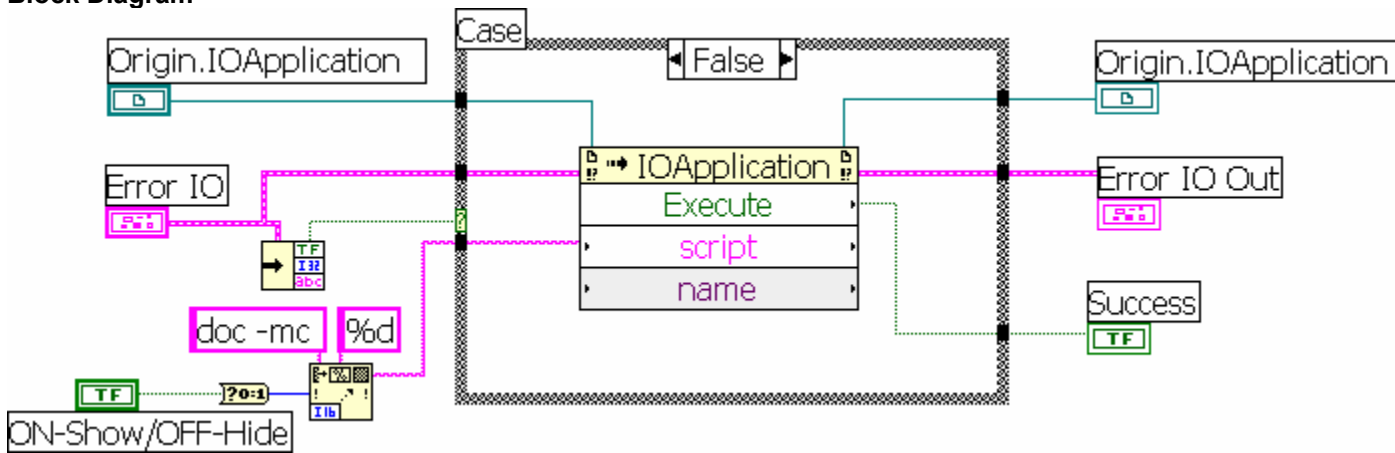
ON>Show/OFF-Hide

Success

Error IO Out

Origin.IOApplication Out

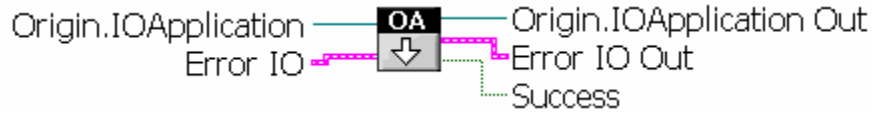
Block Diagram



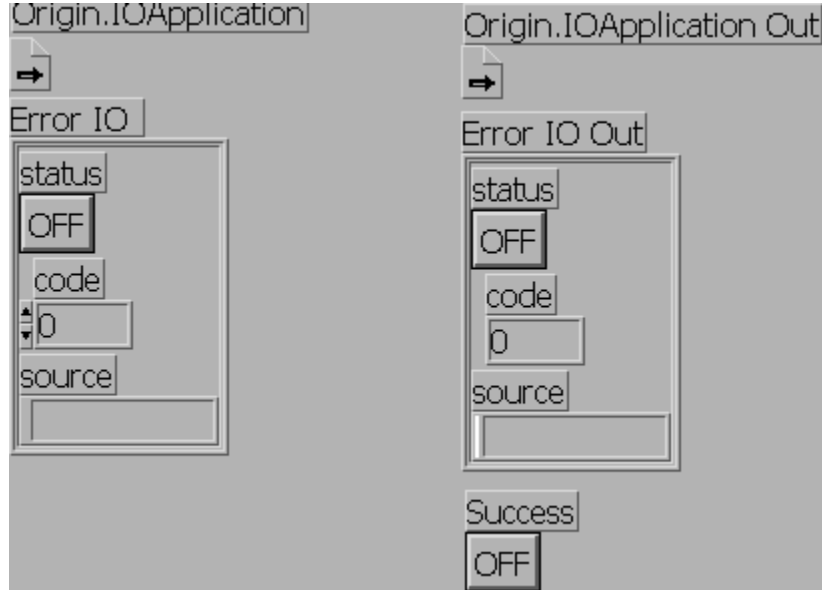
OAExit.vi

Shuts down Origin if no error on input.
Uses Origin.ApplicationSI






Connector Pane



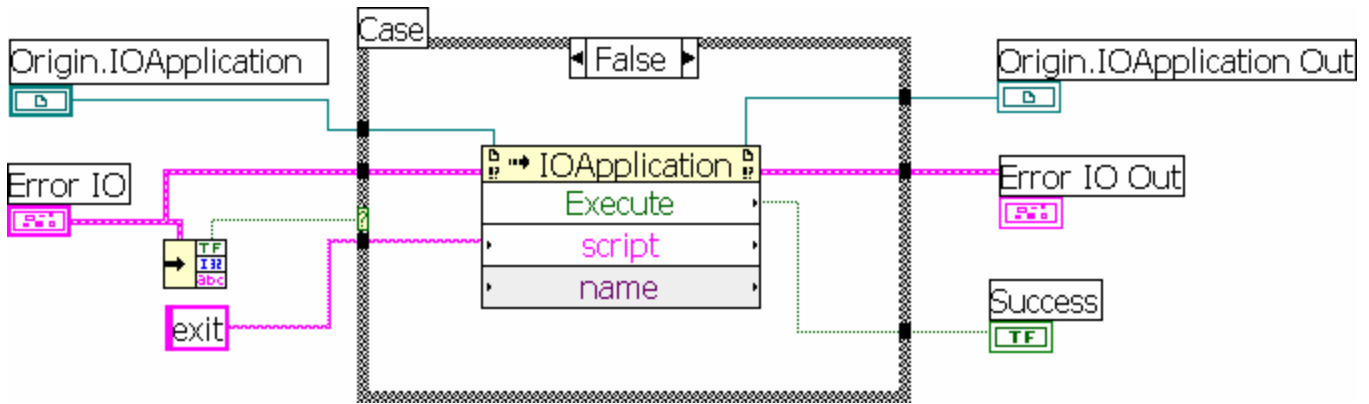
Front Panel



Controls and Indicators

-  Origin.IOApplication
-  Error IO
-  Success
-  Error IO Out
-  Origin.IOApplication Out

Block Diagram



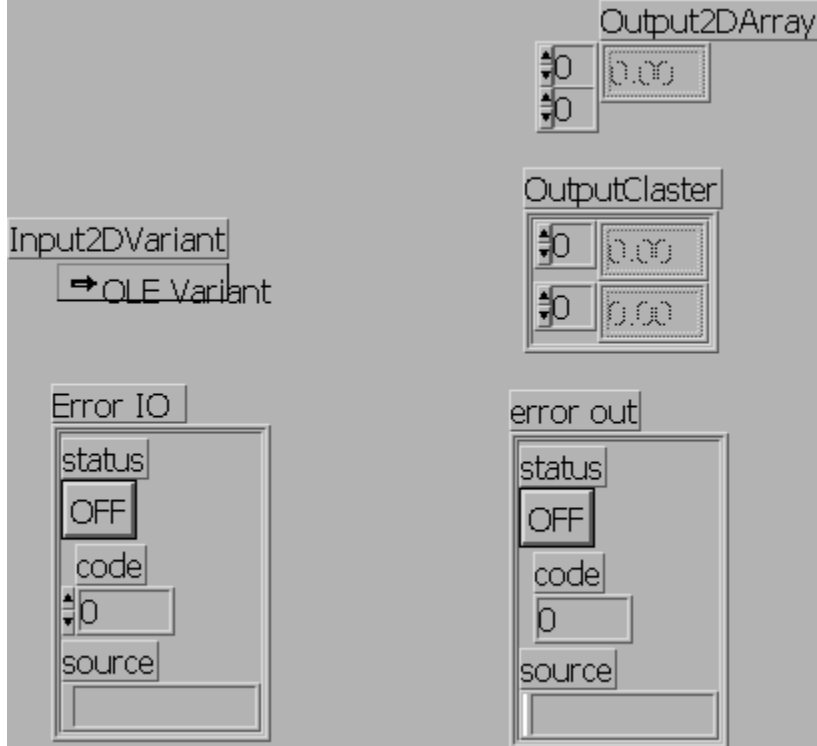
OAVariant2DToArrayOrCluster.vi

Variant found as output of GetMatrix is converted to 2D array and cluster.

Connector Pane




Front Panel



Controls and Indicators

 Error IO

 Input2DVariant

 error out

 Output2DArray

 DBL

 OutputCluster

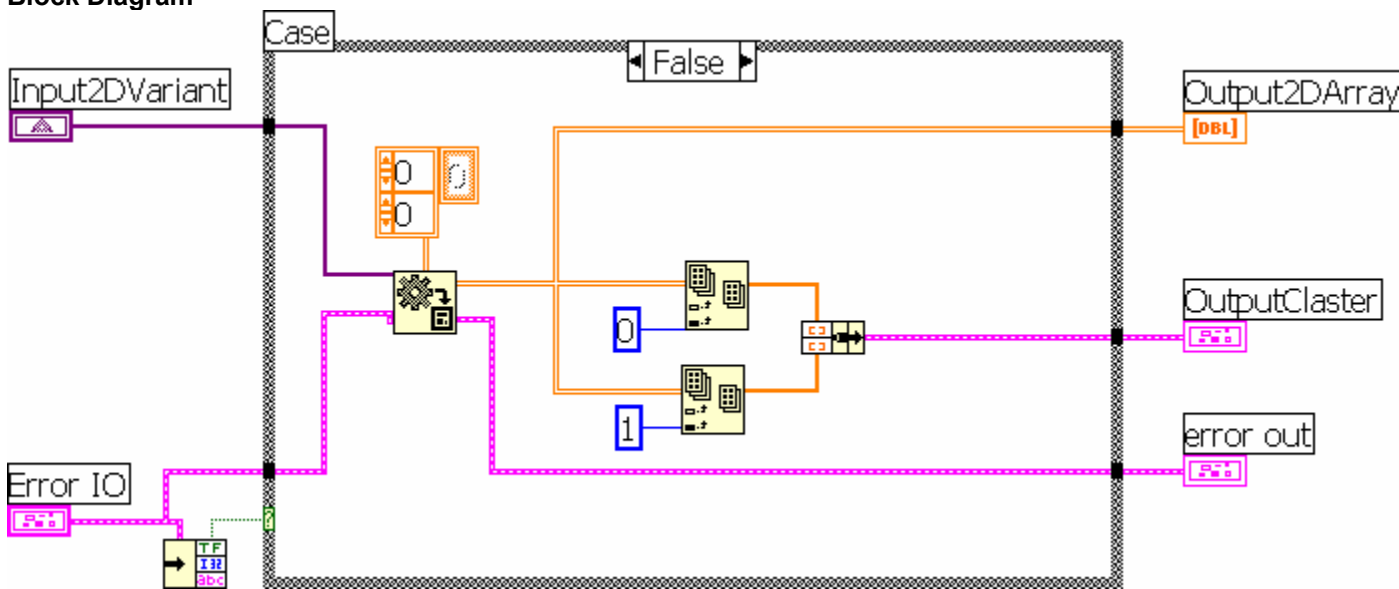
 DBL

 DBL

 DBL

 DBL

Block Diagram



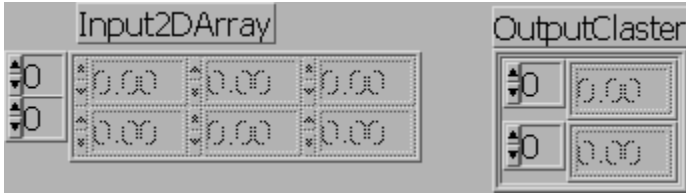
OA2DArrayToCluster.vi

2D array found as output of GetWorksheet or GetMatrix is converted to cluster for input to plotting

Connector Pane



Front Panel



Controls and Indicators

Input2DArray

DBL

OutputCluster

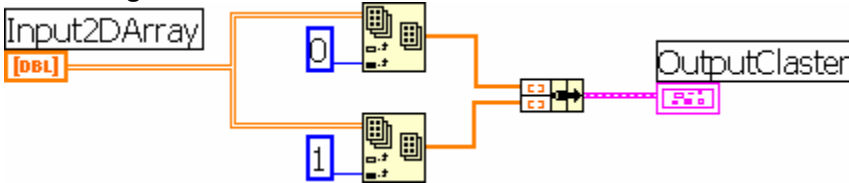
DBL

DBL

DBL

DBL

Block Diagram



ExponentialDecayNoise.vi

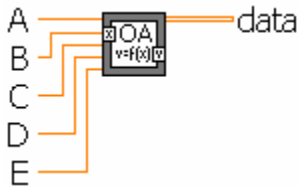
Sample data set.

100 points of:

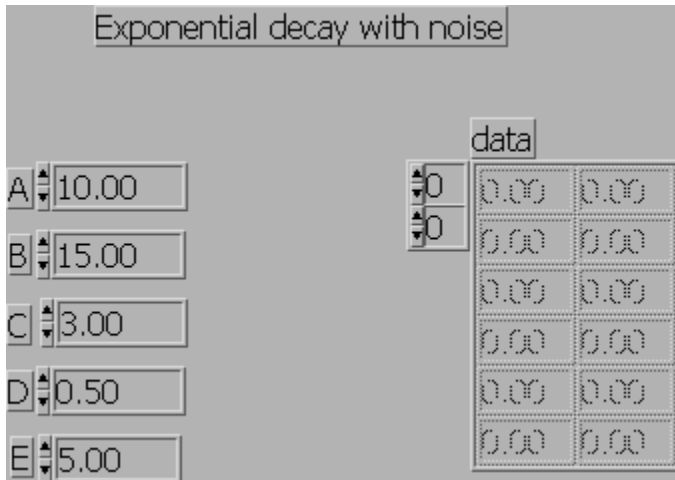
$$f = A * \exp(-x / B) + C * (D - \text{rand}()) + E$$

as 2D array

Connector Pane



Front Panel



Controls and Indicators

A

B

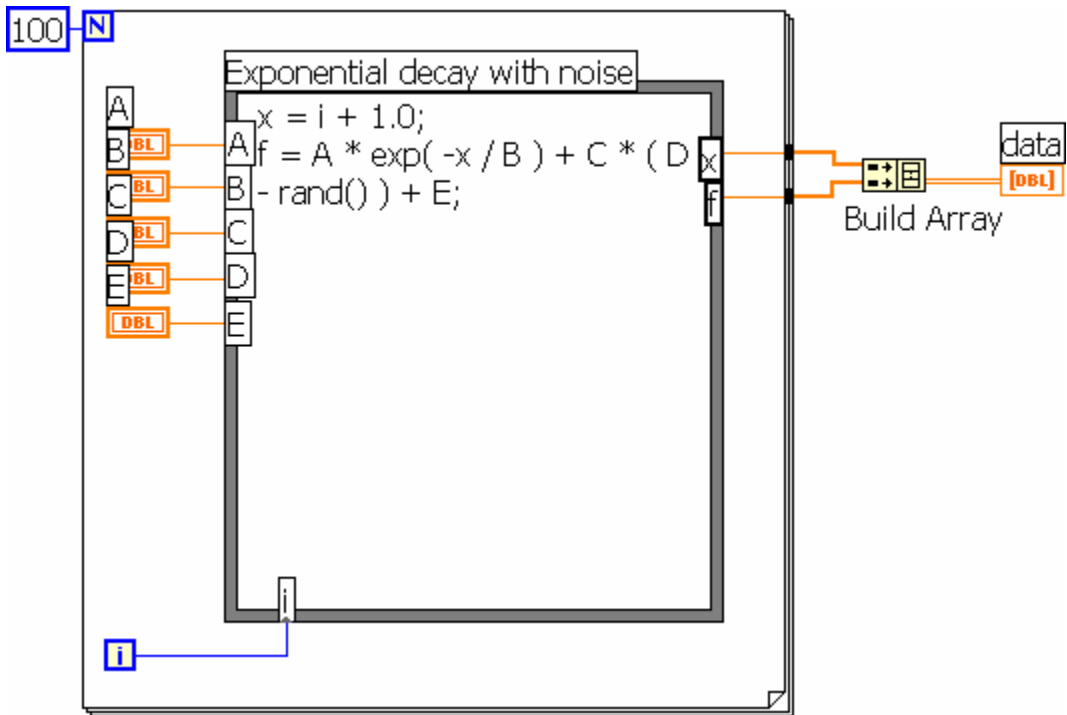
C

D

E

data

Block Diagram



SampleDataNoise.vi

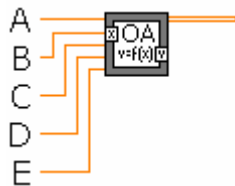
Sample data set.

250 points of:

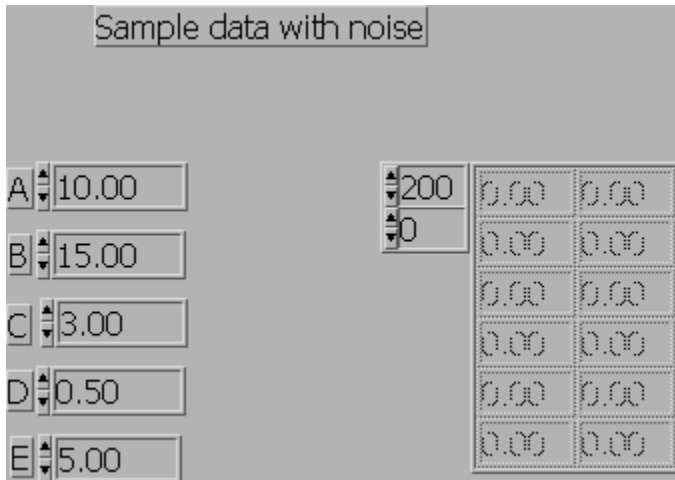
$$f = A * \exp(-(x - B)^2 / C) + D * (E + \text{rand}())$$

as 2D array

Connector Pane



Front Panel



Controls and Indicators

[DBL] A

[DBL] B

[DBL] C

[DBL] D

[DBL] E

[DBL]

[DBL]

Block Diagram

