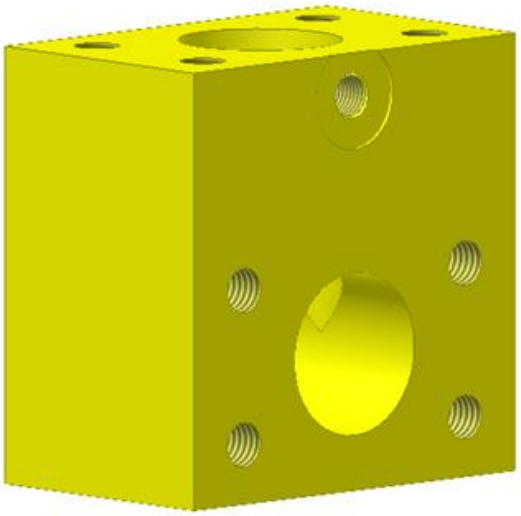
INTRODUCTION



In this tutorial, we will continue your introduction to Autodesk® Inventor® iLogic 2009. This is the second of three tutorials to help you become familiar with iLogic and some of the common features that you will be using.

This tutorial expands upon the information presented in the Basic Tutorial. If you are new to iLogic and haven't gone through that, we recommend that you take the time to do that before you go through this one.

In this tutorial we will be adding rules to a parametric part. Inventor iLogic provides you with the ability to write rules that can drive the parameters, features, attributes, iProperties, and other elements in an Inventor model. The rules are stored within the part or assembly document. Rules are written in a language that is a slightly modified version of Visual Basic .Net (VB.Net). It is easy to get started in this language and to learn more advanced features if needed. In the next tutorial you will learn how to add rules at an assembly level to affect Inventor parts and iParts. In the course of these tutorials, you will be provided with 3D parametric models to add rules to.

The skills you will learn include:

- Using the iLogic parameter interface
- Adding a rule
- Writing a rule
- Running a rule
- Editing a rule
- Editing the iLogic Tree
- Read data from an embedded spreadsheet
- Setting feature and component activity

Autodesk®

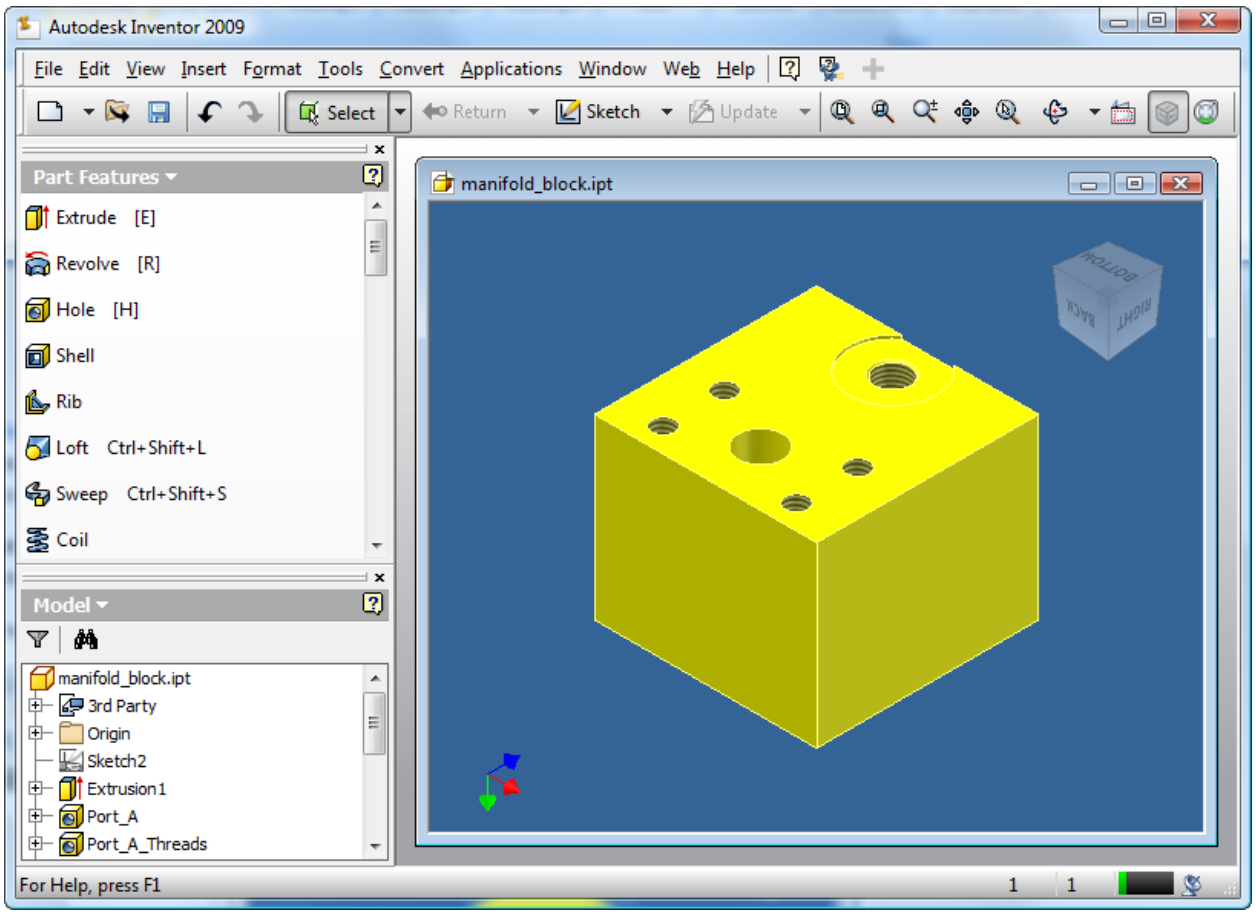
OPEN A PART DOCUMENT

First, make sure that the "iLogic 2009 Tutorials" project is active. This will provide easier access to the relevant files, and will support the work in the next tutorial.

Now that the project is set, open **Manifold_Block_no_rules.ipt**. This is the part that model rules will be added to throughout this tutorial.

To avoid making changes to this file, so you can easily get back to the original, first save this file as a new part file, named **manifold_block.ipt**. Use the File -> Save As command to save the file.

You should now have the **manifold_block.ipt** file open in Inventor:



INTRODUCTION TO THE SAMPLE MODEL

The model that we will work with throughout this tutorial is a simple manifold block. This block contains a set of 3 available ports. These ports are referred to as “A”, “B”, and “C”. Each port is on a different side of a simple block. Each port consists of a center hole (of variable size), and a set of surrounding threaded bolt holes, which will be used to mount union caps in a later tutorial.

This manifold block is capable of being either a “tee”-style block, which has all 3 ports, or an “elbow”-style block, with only 2 ports. Also, we can either create a standard block, which we could order off the shelf, or a custom block, which we would manufacture ourselves. A standard block uses the same size for each of its ports, whereas a custom block can have different sizes for each of the ports.

Finally, the part contains an embedded Microsoft Excel spreadsheet, which is used to specify the values for various parameters as the port sizes are changed.

Now, let’s start adding additional parameters to the model to support the rules well write later.

CREATE ILOGIC PARAMETERS

Open the iLogic Parameter Editor, using the Parameters button on the iLogic panel bar or toolbar.

Most of the parameters have been named in the parameter editor already. It is good practice to name your parameters for future reference when creating a parametric design. Parameters with meaningful names make the rules that drive or reference them easier to read and understand.

Keep in mind that parameter names in iLogic are case sensitive. Please be sure to follow the case being used in the parameter editor, and while creating rules.

CREATE PORT SIZE PARAMETERS

First, we need a set of parameters to control the size of 3 ports on our manifold block.

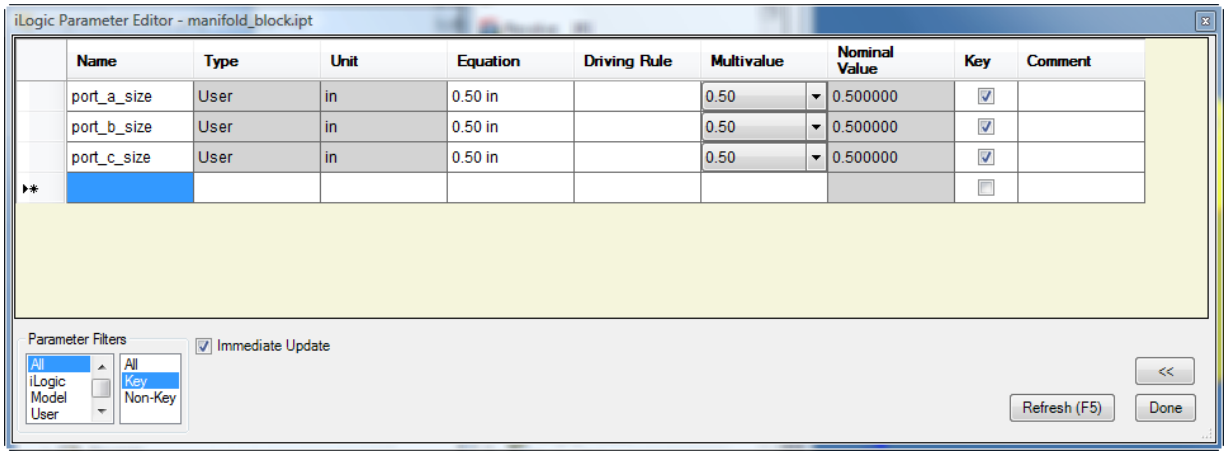
To begin, create a new iLogic parameter named “port_a_size”. Set its Unit value to “in”. Give it an initial equation of “0.50”. Make it a multi-value list, with the values:

0.50
0.75
1.00
1.25
1.50
2.00
2.50
3.00

If you need more details on the exact steps needed to create a new iLogic parameter, please revisit the Basic Tutorial. It covers these steps in greater detail. Remember that you can cut and paste the values above to set the values for the multi-value list.

Finally, make **port_a_size** a key parameter.

Create two more parameters, named **port_b_size** and **port_c_size** with the same settings and multi-value lists. Set both of these as **Key** parameters. You should now have the following key parameters:



CREATE BLOCK AND COMPONENT TYPE PARAMETERS

Now, we need to create two more parameters, which will control whether we are modeling a "tee" block or an "elbow" block, and also whether we are creating a standard or a custom block.

First, create a new parameter named "**block"**. Make this be a String type parameter. Make it a multi-value parameter, with the values:

tee
elbow

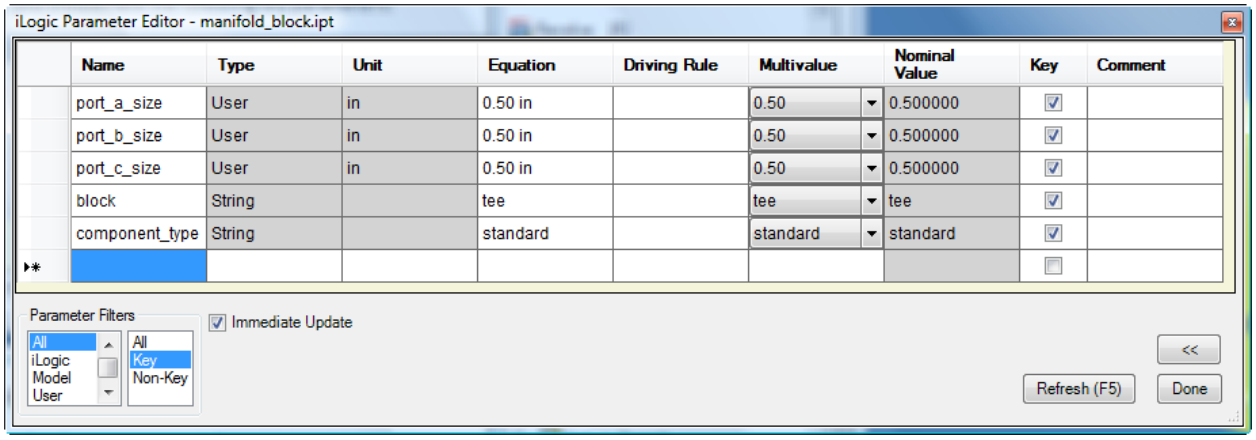
Set the current value to be "tee", and make this a key parameter.

Create a second parameter, named "component_type". Make this a String parameter also, and make it a multi-value parameter, with the values:

standard
custom

Set the current value to be "standard", and make this a key parameter.

Now, you should have the following key parameters:



Click **Done** at this point to exit the iLogic Parameter Editor.

SAVE THE INVENTOR PART DOCUMENT

This would be a good time to save your document, before we move on to creating rules.

DEFINE THE RULES OF THE MODEL

Now, we define a set of rules that will drive the geometry of the model based on the values of the key parameters we defined above. In this section, we will construct each rule in segments. The entire text of all of the rules can be found in the appendix at the end of the document for easy review later.

ADDING A RULE TO CONTROL PORT VISIBILITY

The first rule we need to add will make model changes to the Port B features, based on whether the user wants an "elbow" or a "tee" block. This will involve suppressing or enabling the Port B-related features, based on the value of the **block** parameter.

Create a new rule named "block_shape_rule". Recall that you add a new rule using the "Add Rule" button on the iLogic panel bar or toolbar:

ILOGIC RULE EDITOR

Once you've specified the new rule name in the Rule Name dialog, the iLogic Rule Editor dialog will be displayed.

Recall that, in the rule editor, you can use parameter names from the model as variables in your rule. Also, you can select from lists of the available parameters, as well as features and other model entities, for inclusion in your rule.

The first part of your new rule will handle the case where the block is a "tee"-style block.

```
If block = "tee" Then
```

Recall that for a tee-style block, all 3 ports will be active. In case Port B was previously disabled, we add the steps to (re)enable it, which involves activating two features in the part:

```
Feature.IsActive("Port_B") = True
Feature.IsActive("Port_B_Threads") = True
```

Reminder: The Feature.IsActive function is available for selection on the Rule Syntax tab, in the Component/Feature function category. You can access the names of the available features on the Features tab of the rule editor.

We have now defined the behavior that we wish our model to take when **block** = "tee". Now we will define the model behavior when the **block** = "elbow":

```
ElseIf block = "elbow" Then
```

When we're creating an elbow block, we want to suppress the Port B-related features. Since this is simply the opposite of what we did for a "tee" block, we use the same function, but with opposite values:

```
Feature.IsActive("Port_B") = False
Feature.IsActive("Port_B_Threads") = False
```

The easiest way to add these lines is to copy and paste the text from above, and then change "True" to "False" in the new lines.

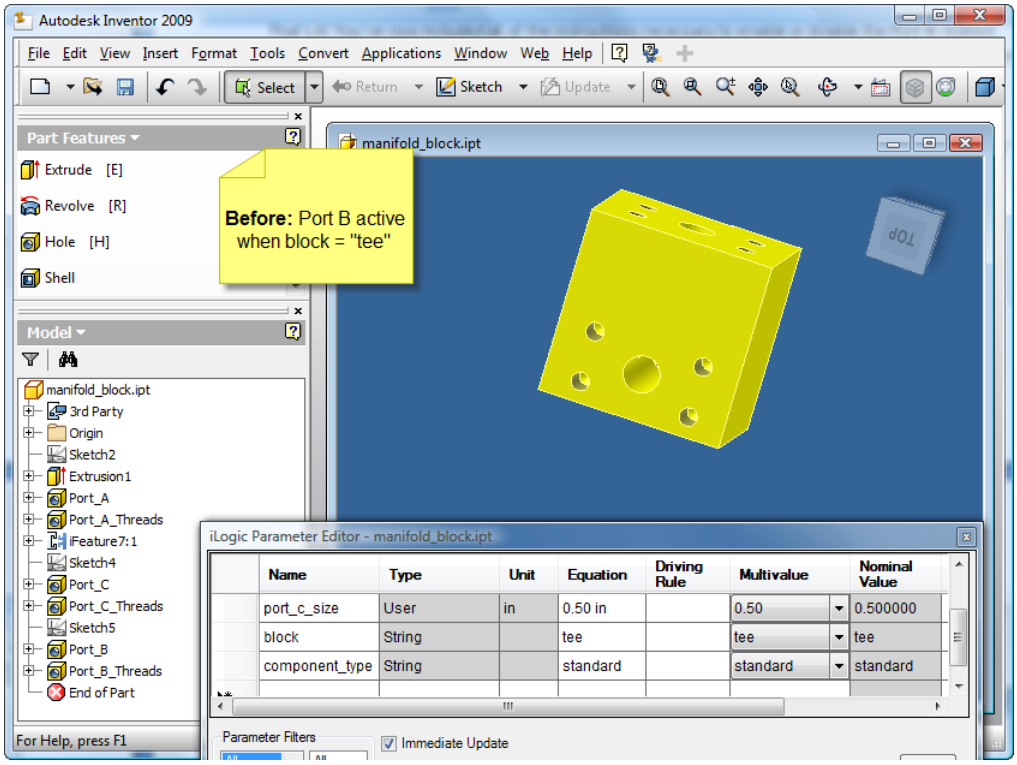
This is all we need to do for the **block** = "elbow" case, so we can end the **If** block:

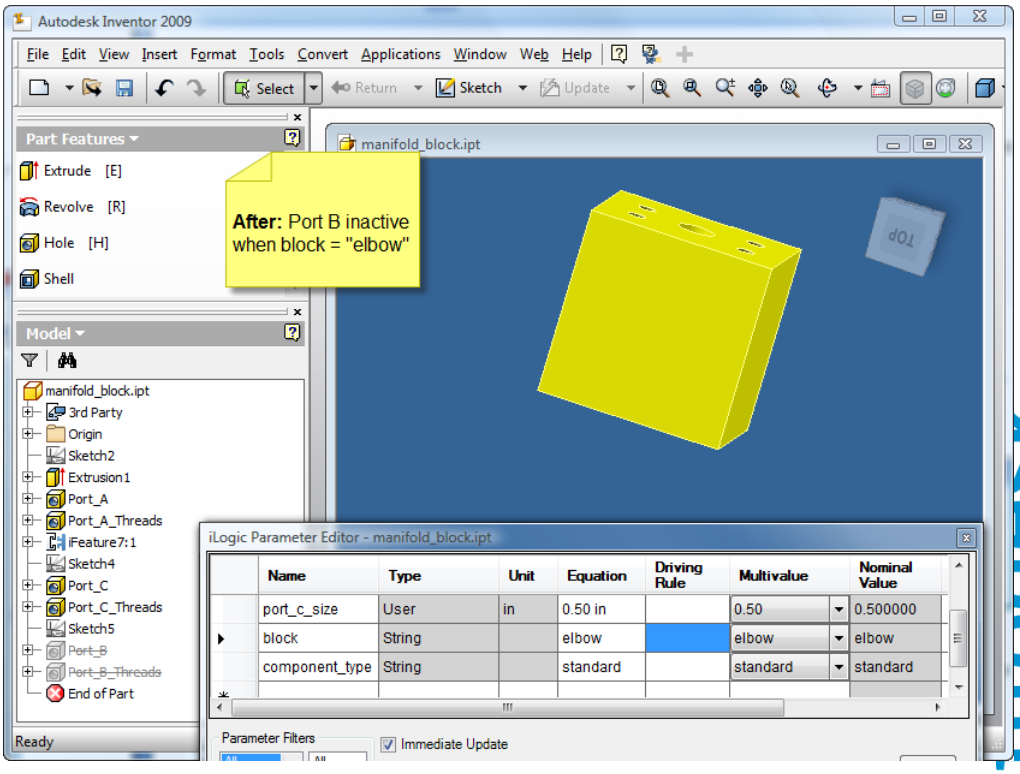
```
End If
```

That's it! You've now included all of the instructions necessary to enable or disable the Port B-related features based on the type of block to be used. If you haven't already done so, click **OK** in the rule editor dialog to save the completed rule.

TEST THE RULE (BLOCK_SHAPE_RULE)

To verify this rule is really in control of our model. Now, open the iLogic Parameter Editor. In the "block" row, change **tee** to **elbow** using the Multi-value selection box.





MANAGING PART CONFIGURATIONS

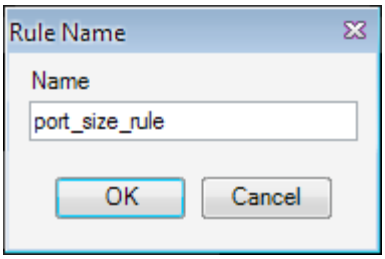
We will now cover the last two topics listed in the beginning of this tutorial:

- Read data from an embedded spreadsheet
- Set feature and component activity

Now we will write a rule that uses values from the embedded Excel spreadsheet to set the values for parameters that control the port geometry, based on a specified size. We will look up the port size in the spreadsheet to identify the row of values that we want, and then read fields from that row to get the parameter values to use.

Go to the iLogic Parameter Editor and change the **block** parameter from **elbow** back to **tee**. When we have a Tee style block, there are three ports. Each port is listed in the iLogic parameter editor, but changing the port size in the parameter editor will not change the port size in our model. We have to add rules to drive the different port sizes.

ADD PORT_SIZE_RULE

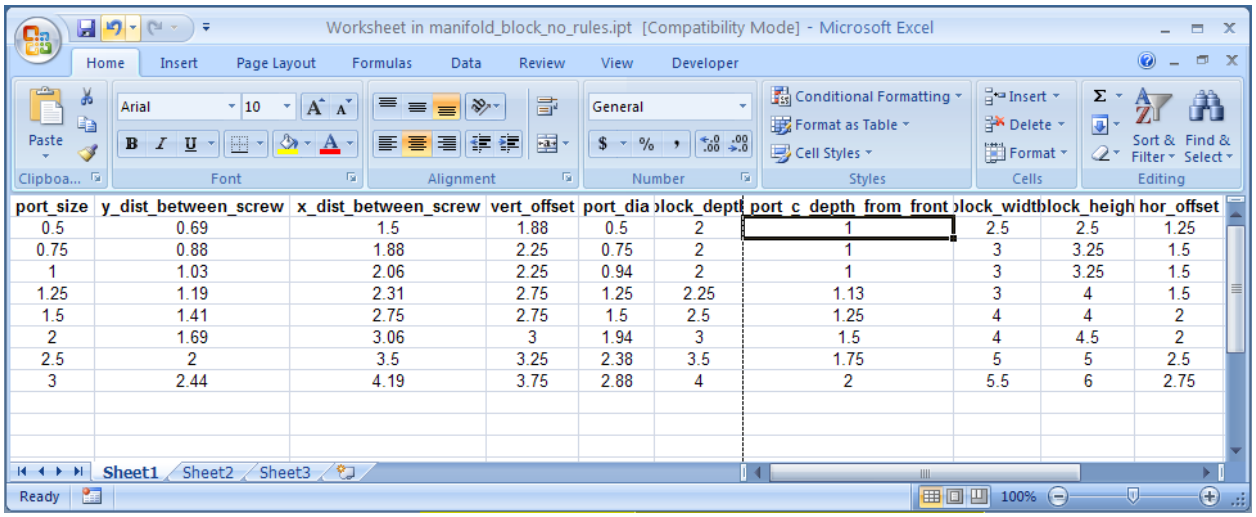


Our first step will be to add a rule that will set the size of the ports and the dimensions of the screw pattern around each port. The screw pattern will be used in the assembly to hold a flange onto the block.

Add a rule named “port_size_rule”. Click **OK**. The rule editor will appear.

iLogic provides built-in functions that read information from Excel spreadsheets. These functions are available on the Rule Syntax tab, under the Data Links function category.

To access the embedded spreadsheet, go to the Inventor model browser, and expand the “3rd Party” item in the tree. Right-click on the “Embedding 1” entry, and choose “Edit”. The spreadsheet looks like this:



port_size	y_dist_between_screw	x_dist_between_screw	vert_offset	port_dia	lock_depth	port_c_depth_from_front	lock_width	block_heigh	hor_offset
0.5	0.69	1.5	1.88	0.5	2	1	2.5	2.5	1.25
0.75	0.88	1.88	2.25	0.75	2	1	3	3.25	1.5
1	1.03	2.06	2.25	0.94	2	1	3	3.25	1.5
1.25	1.19	2.31	2.75	1.25	2.25	1.13	3	4	1.5
1.5	1.41	2.75	2.75	1.5	2.5	1.25	4	4	2
2	1.69	3.06	3	1.94	3	1.5	4	4.5	2
2.5	2	3.5	3.25	2.38	3.5	1.75	5	5	2.5
3	2.44	4.19	3.75	2.88	4	2	5.5	6	2.75

The first thing we need to do is to locate the row that contains the values to be used for Port A. We look up the value matching the **port_a_size** parameter in a column named "port_size". The function to use is labeled *i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "columnName", "<=", 0.2, "columnName", "<=", 4.1)* in the rule editor. Once you have inserted this function template into your rule, replace **"columnName"** with **"port_size"**, **"<="** with **"="**, and **0.2** with **port_size.**

```
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_a_size)
```

This indicates that we want to find the row in the embedded spreadsheet that has a **port_size** column that equals the value of the **port_a_size** parameter.

Now, we want to set a series of parameters based on the values of cells from this row in the spreadsheet. These parameters will control the port diameter, drill depth, and the distance between the bolt holes. Use the function labeled *"= GoExcel.CurrentRowValue("columnName")".*

```
port_a_y_dist_between_screw  = GoExcel.CurrentRowValue("y_dist_between_screw")
port_a_x_dist_between_screw  = GoExcel.CurrentRowValue("x_dist_between_screw")
port_a_port_dia = GoExcel.CurrentRowValue("port_dia")
Port_A_Drill_Depth = GoExcel.CurrentRowValue("tap_drill_depth")
```

Remember that you can use the items in the Parameters tab of the rule editor to access various sets of parameters in the model. The ones used above are all Model parameters.

We have one more line of code to add for this part of the rule. This line of code will define the thread of the tapped holes. Under **Categories**, click on **Component/Feature**. Near the bottom of the list of functions, choose *Feature.ThreadDesignation("featurename")= "3/8-16 UNC"*. We will replace the feature name with "Port_A_Threads", and will use another value from the spreadsheet row for its value.

```
Feature.ThreadDesignation("Port_A_Threads") = GoExcel.CurrentRowValue("tap_dim")
```

Here, we have indicated that we should use the **tap_dim** cell to get the thread designation for the bolt holes.

Now you've completed the instructions to set the parameters related to Port A. Since we have three ports, we will now copy the rule text that we have created thus far and paste it two more times. For the second grouping of code, change **port_a** text to be **port_b**. For the third grouping, change **port_a** to be **port_c**.

The additional rule lines should look like this:

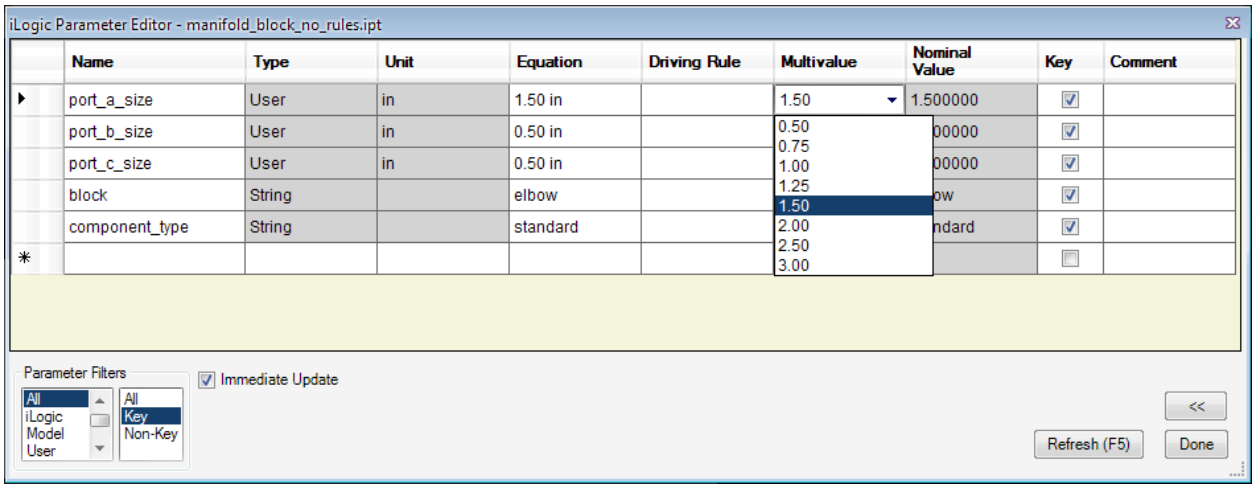
```
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_b_size)
port_b_y_dist_between_screw  = GoExcel.CurrentRowValue("y_dist_between_screw")
port_b_x_dist_between_screw  = GoExcel.CurrentRowValue("x_dist_between_screw")
port_b_port_dia = GoExcel.CurrentRowValue("port_dia")
Port_B_Drill_Depth = GoExcel.CurrentRowValue("tap_drill_depth")
Feature.ThreadDesignation("Port_B_Threads") = GoExcel.CurrentRowValue("tap_dim")

i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_c_size)
port_c_y_dist_between_screw  = GoExcel.CurrentRowValue("y_dist_between_screw")
port_c_x_dist_between_screw  = GoExcel.CurrentRowValue("x_dist_between_screw")
port_c_port_dia = GoExcel.CurrentRowValue("port_dia")
Port_C_Drill_Depth = GoExcel.CurrentRowValue("tap_drill_depth")
Feature.ThreadDesignation("Port_C_Threads") = GoExcel.CurrentRowValue("tap_dim")
```

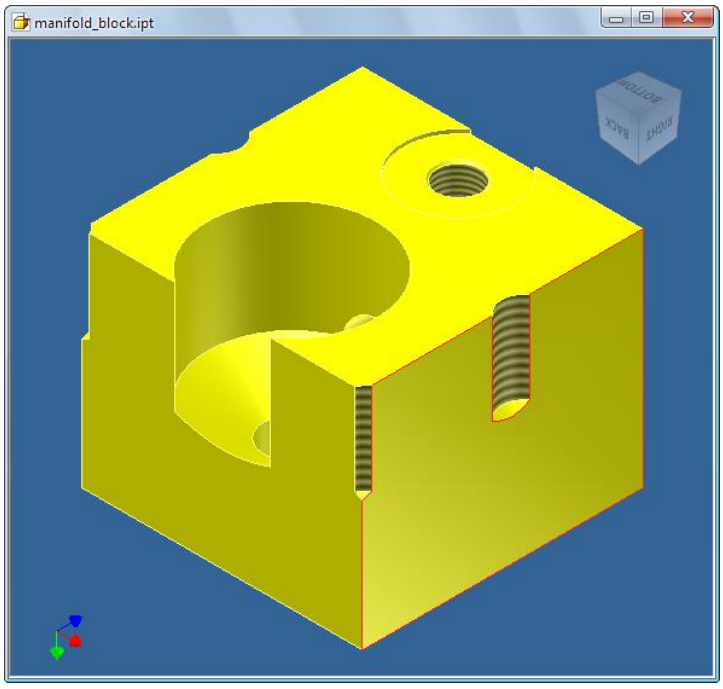
That's all we need to set up the port geometry. Click OK in the iLogic rule editor to save your **port_size_rule**. Your model may or may not update at this point depending upon how the iLogic port size parameters were initially set.

TEST THE RULE (PORT_SIZE_RULE)

Activate the iLogic Parameter Editor to see how our new rule will update the model when parameter values are changed. Change **port_a_size** to **1.50** by selecting from the multi-value list. When you click in another column (e.g., Driving Rule) or hit the Tab key, the parameter equation change will be made, and the **port_size_rule** will run. This will apply the changes to the parameters specified in the rule.



Notice how the model changes as you set this or other values. Using the value **1.50**, your model should look something like this:



Clearly, we have some more work to do to get other aspects of the model (e.g., its size) to update according to the selected port size.

THE BLOCK_SIZE RULE

Now that we can change the size of each port, we will need to decide which face has the largest port so that the block can be sized appropriately. This will require another rule.

ADD THE RULE

Add a new rule named "block_size". Proceed to the rule editor.

First, we need to figure out which port is the largest. To do this, we examine the values of the three port size parameters, and hold onto the value of the largest one. As we did for the **block_shape_rule**, we will need to have different behavior for tee-style vs. elbow-style blocks.

For tee-style blocks, all three ports will be used, so we need to check the sizes for all of them. For elbow-style blocks, we are not interested in the size of Port B, since it will be suppressed. We use the ***MaxOfMany*** function to get the largest value out of a set of input values. We use the appropriate set of input values for each case. This function is available from the Math category on the Rule Syntax tab.

```
If block = "tee" Then
port = MaxOfMany(port_a_size,port_b_size,port_c_size)
ElseIf block = "elbow" Then
port = MaxOfMany(port_a_size,port_c_size)
End If
```

Notice that we've used a new "local variable" named "port" to hold the size of the largest available port. Now we have to tell the model what to do with this information. This model is getting its information from an embedded Excel spreadsheet. We will need to look at the spreadsheet to update the overall sizes of this model.

Press **Enter** twice to add some white space in the rule.

As with the previous rule, we will use information from the embedded spreadsheet to get the values for other parameters. We first need to locate the row of information in the embedded spreadsheet.

```
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port)
```

Here, we're again using the **port_size** column for the lookup, and using the value of the variable assigned above as the value to look for.

We will now set some model parameters with information from the embedded Excel spreadsheet, using the row we found for the largest port size.

```
block_depth = GoExcel.CurrentRowValue("block_depth")
port_c_depth_from_front = GoExcel.CurrentRowValue("port_c_depth_from_front")
block_width = GoExcel.CurrentRowValue("block_width")
port_a_hor_offset = GoExcel.CurrentRowValue("hor_offset")
port_b_hor_offset = GoExcel.CurrentRowValue("hor_offset")
port_c_hor_offset = GoExcel.CurrentRowValue("hor_offset")
```

So far we have figured out which port is the largest, and we are sizing the top of the block according to these sizes. We will examine the tee and elbow to determine which port size is bigger on the port a / port b face to determine height of the block.

For this, we will use another local variable, “porta”, to hold onto this value. Again, since Port B is not used for elbow-style blocks, we have different steps depending on this setting.

```
If block = "tee" Then
porta = MaxOfMany(port_a_size, port_b_size)
ElseIf block = "elbow"
porta = port_a_size
End If
```

Notice that, for elbow-style blocks, we don’t use the ***MaxOfMany*** function, since there’s only one value to consider. We can just set the variable from that value.

Now we need to tell the model where to get the values for the height of the block. We will go to the Excel spreadsheet again to get this information.

```
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", porta)
```

Here, we find the correct row to use, based on the local variable we just set. We will now set the height of our block.

```
port_a_vert_offset = GoExcel.CurrentRowValue("vert_offset")
port_b_vert_offset = GoExcel.CurrentRowValue("vert_offset")
```

Almost finished. We need to set the value for two additional parameters. One of these will set the block height. The other will set the vertical offset of Port C. For this value, we have some special logic, which adds some extra space beyond the vertical offset used for the other ports. We get this from another spreadsheet cell. Note that we only do this for elbow-style blocks.

```
If block = "elbow" Then
port_c_vert_offset = GoExcel.CurrentRowValue("vert_offset") + (GoExcel.CurrentRowValue("port_dia")/4)
Else
port_c_vert_offset = GoExcel.CurrentRowValue("vert_offset")
End If
block_height = GoExcel.CurrentRowValue("block_height")
```

This finishes up the **block_size_rule** rule. Click **OK** in the iLogic Rule Editor to see the effect of this rule. Change any of the port sizes in the iLogic Parameter Editor and see the model update.

SETTING THE COMPONENT TYPE

This next rule could have been created first, but we have left it until the end in order to show how rules can be re-ordered after they are created.

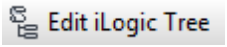
Create a new rule named “component_type_rule”, and open the rule editor.

This rule is very simple: for standard components, we want to make sure that all port sizes are the same. We do this by setting the sizes for ports B and C to be the same as Port A.

```
If component_type = "standard" Then
port_b_size = port_a_size
port_c_size = port_a_size
End If
```

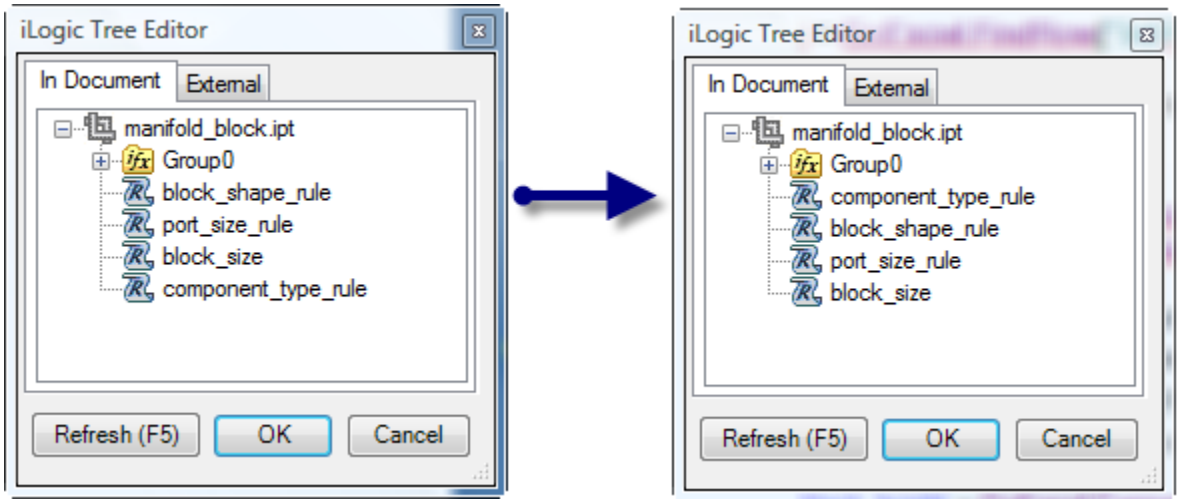
Click **OK** when you are done to save this rule.

REORDERING RULES

The order of rule execution sometimes affects the results of these rules. You can alter the order of execution from the iLogic Tree Editor. 

You can also switch to the iLogic browser panel to view the tree of iLogic rules and parameters.

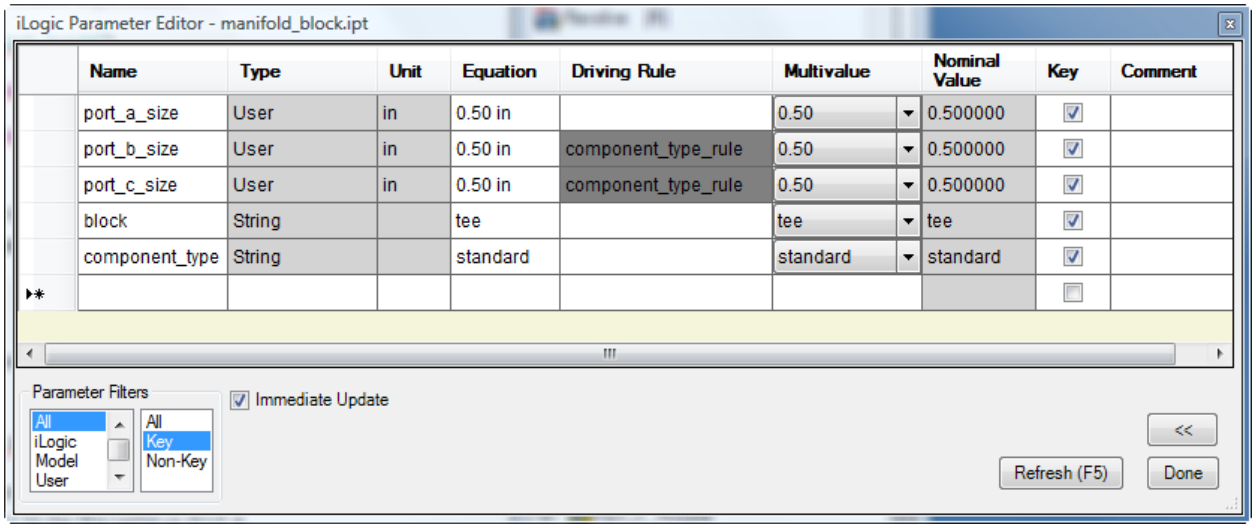
The **iLogic Tree Editor** dialog box will be displayed. Left-click on the **component_type** rule that we just created. Drag and drop this rule above the **block_shape_rule**.



Click **OK** when you are done. You might notice that the inventor Update icon is active. Click on it to update the model.

"DRIVING" RULES

Activate the iLogic Parameter Editor dialog.



Notice the Driving Rule column value for **port_b_size** and **port_c_size** parameters. These column values indicate that there is a driving rule (**component_type**) in place for these two parameters. When component_type equals "standard", these two port sizes are set automatically by the rule to be equal to **port_a_size**. Attempts to change these port values while component_type is equal to "standard" will fail, as the rule fires and controls these values. Notice that after changing **component_type** to "custom", you are free to choose independent values for port_b_size and port_c_size.

Try it. Change **component type** from **standard** to **custom**. Now change **port_b_size** to **3** inch and **port_c_size** to be **.75** inch. Notice that independent port sizes are possible. Change your **component type** back to **standard**. What happed to your model? All of the ports should have updated to match port_a_size.

UPDATING IPROPERTIES

Let's add one more rule. This rule will update some of the iProperties of the manifold block part. Add a new rule named "part_number_rule". This rule will set the Inventor Part Number iProperty value.

For standard components, we can look up the Part Number in the embedded spreadsheet. Then, we use the value in the "model_code" cell to set the Part Number property for the part. To make it easy to do this, iLogic provides the ***iProperties.Value*** function, in the iProperties category on the Rule Syntax tab.

As with earlier rules, we first need to locate the row in the embedded spreadsheet to read values from. We locate the row using **port_a_size**.

```
If component_type = "standard" Then
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_a_size)
iProperties.Value("Project", "Part Number") = GoExcel.CurrentRowValue("model_code")
```

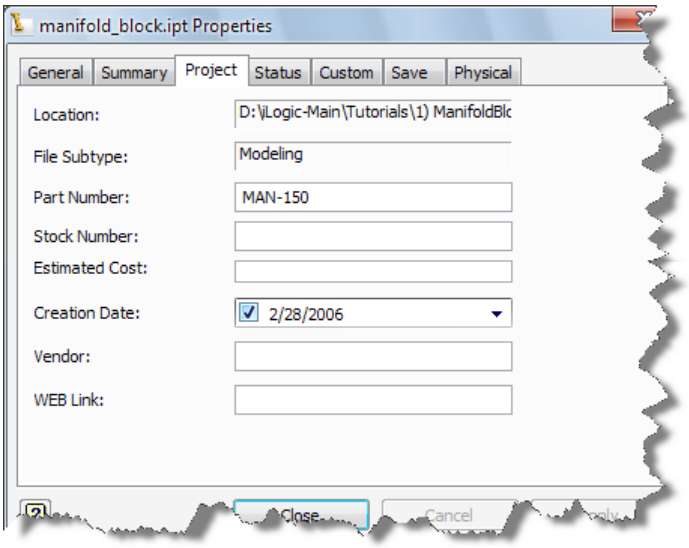
For custom components, we just used a fixed string for the part number, so we don't need to use any information from the spreadsheet.

```
Else
iProperties.Value("Project", "Part Number") = "HomeMade"
End If
```

Click **OK** on the Rule Editor to close this rule.

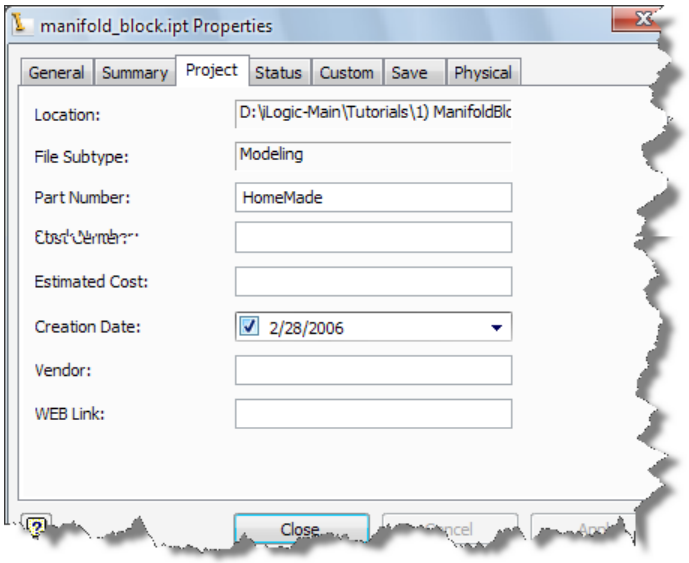
TEST THE RULE (PART_NUMBER_RULE)

To verify that the new rule is working, click on **File** from the Inventor pull down menu. Click on **iProperties**. The iProperties dialog box will appear. Click on the **Project** tab. Text is entered next to the Part Number.



Click **Close** to close this dialog box.

Go to the **iLogic Parameter Editor** and change **component** from **standard** to **custom**. Click **OK** are reopen the Inventor iPorperties dialog box. You will notice that the Part Number has updated to be “HomeMade”.



CONGRATULATIONS!

You have covered a lot of new ground in this tutorial. You now understand some of the basic iLogic methods for turning a parametric single model into an intelligent super model!

In this tutorial you have:

- Used the iLogic parameter interface
- Added a rule
- Written a rule
- Caused a rule to run
- Edited a rule
- Edited the iLogic Tree
- Read data from an embedded spreadsheet
- Set feature and component activity
- Set iProperty values from a rule

Save your work. We will be using this model in the next tutorial to build an assembly and add logic at the assembly level.

APPENDIX – COMPLETE RULE TEXT

The following sections contain the complete text of all of the rules presented in this tutorial. These rules are also available in completed form in the manifold_block_complete.ipt file, included in the tutorials directory.

BLOCK_SHAPE_RULE

```
If block = "tee" Then
Feature.IsActive("Port_B") = True
Feature.IsActive("Port_B_Threads") = True
ElseIf block = "elbow" Then
Feature.IsActive("Port_B") = False
Feature.IsActive("Port_B_Threads") = False
End If
```

PORT_SIZE_RULE

```
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_a_size)
port_a_y_dist_between_screw  = GoExcel.CurrentRowValue("y_dist_between_screw")
port_a_x_dist_between_screw  = GoExcel.CurrentRowValue("x_dist_between_screw")
port_a_port_dia = GoExcel.CurrentRowValue("port_dia")
Port_A_Drill_Depth = GoExcel.CurrentRowValue("tap_drill_depth")
Feature.ThreadDesignation("Port_A_Threads") = GoExcel.CurrentRowValue("tap_dim")

i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_b_size)
port_b_y_dist_between_screw  = GoExcel.CurrentRowValue("y_dist_between_screw")
port_b_x_dist_between_screw  = GoExcel.CurrentRowValue("x_dist_between_screw")
port_b_port_dia = GoExcel.CurrentRowValue("port_dia")
Port_B_Drill_Depth = GoExcel.CurrentRowValue("tap_drill_depth")
Feature.ThreadDesignation("Port_B_Threads") = GoExcel.CurrentRowValue("tap_dim")

i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_c_size)
port_c_y_dist_between_screw  = GoExcel.CurrentRowValue("y_dist_between_screw")
port_c_x_dist_between_screw  = GoExcel.CurrentRowValue("x_dist_between_screw")
port_c_port_dia = GoExcel.CurrentRowValue("port_dia")
Port_C_Drill_Depth = GoExcel.CurrentRowValue("tap_drill_depth")
Feature.ThreadDesignation("Port_C_Threads") = GoExcel.CurrentRowValue("tap_dim")
```

BLOCK_SIZE_RULE

```
If block = "tee" Then
port = MaxOfMany(port_a_size,port_b_size,port_c_size)
ElseIf block = "elbow" Then
port = MaxOfMany(port_a_size,port_c_size)
End If

i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port)
block_depth = GoExcel.CurrentRowValue("block_depth")
port_c_depth_from_front = GoExcel.CurrentRowValue("port_c_depth_from_front")
block_width = GoExcel.CurrentRowValue("block_width")
port_a_hor_offset = GoExcel.CurrentRowValue("hor_offset")
port_b_hor_offset = GoExcel.CurrentRowValue("hor_offset")
port_c_hor_offset = GoExcel.CurrentRowValue("hor_offset")

If block = "tee" Then
porta = MaxOfMany(port_a_size, port_b_size)
ElseIf block = "elbow"
porta = port_a_size
End If

i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", porta)
port_a_vert_offset = GoExcel.CurrentRowValue("vert_offset")
port_b_vert_offset = GoExcel.CurrentRowValue("vert_offset")

If block = "elbow" Then
port_c_vert_offset = GoExcel.CurrentRowValue("vert_offset") + (GoExcel.CurrentRowValue("port_dia")/4)
Else
port_c_vert_offset = GoExcel.CurrentRowValue("vert_offset")
End If
block_height = GoExcel.CurrentRowValue("block_height")
```

COMPONENT_TYPE_RULE

```
If component_type = "standard" Then
port_b_size = port_a_size
port_c_size = port_a_size
End If
```

PART_NUMBER_RULE

```
If component_type = "standard" Then
i = GoExcel.FindRow("3rd Party:Embedding 1", "Sheet1", "port_size", "=", port_a_size)
iProperties.Value("Project", "Part Number") = GoExcel.CurrentRowValue("model_code")
Else
iProperties.Value("Project", "Part Number") = "HomeMade"
End If
```