1 Use of Conditional Statements

1.1 Overview

Conditional statements help you perform actions based on specific criteria.

For example: Consider a situation where a set of instructions needs to be performed if a condition is true. In case the condition is false, a different set of instructions needs to be performed. You can use conditional statements in such a scenario to perform one set of actions if the condition is true and another set of actions if the condition is false.

If-Then Statement

The If-Then statement is used when you need to perform an action or a set of actions only when certain criteria or conditions are met. In case the condition is not met, no action is to be performed.

In an If-Then statement, you can limit an action, or a set of actions to be performed, only if a condition is true, and no action is performed if the condition is false.

Example with syntax:

```
If Length = "long" Then
  Boxsize = 10.0
End If
```
If-Then-Else Statement

The **If-Then-Else** statement is used when you need to perform an action or a set of actions when a certain criteria or conditions are met and another set of actions in case that particular criteria/conditions are not met.

Example with syntax:

```plaintext
If Length = "long" Then
    Boxsize = 10.0
Else
    Boxsize = 6.0
End If
```

The **End If** statement closes the **If** loop and the program resumes.

**Single line If statement**

The general form of using an **If** statement is in its block form along with an **End If**. However, you can also use the **If** statement in one single line. The **End If** statement is not used while using the single line **If** statement.

Example with syntax:

```plaintext
If Width < 100 Then MessageBox.Show("Width value entered is very small", "Low Value Warning")
If Density < 1000 Then volume = 60 Else volume = 30
```
If-Then-ElseIf statement

The If-Then-ElseIf or the extended block If statement allows you to insert an additional condition between the ‘If’ and the ‘Else’ parts of the code. The keyword to be used is ‘ElseIf’.

Using an If-Then-ElseIf statement

Example with syntax:

```plaintext
If Liquid = "Water" Then
    Density = 1000
    ElseIf Liquid = "Turpentine" Then
        Density = 868.20
        ElseIf Liquid = "Kerosene" Then
            Density = 817.15
    End If
End If
```

Without Using an If-Then-ElseIf statement

Example with syntax:

```plaintext
If Liquid = "Water" Then
    Density = 1000
Else
    If Liquid = "Turpentine" Then
        Density = 868.20
        Else
            If Liquid = "Kerosene" Then
                Density = 817.15
            End If
        End If
    End If
End If
```

The working of an If-Then-ElseIf conditional statement

Boolean Variables in conditional statements

A Boolean variable has either a ‘true’ or a ‘false’ value associated with it. These are used in relational expressions to check for a specific characteristic and perform an action.

```plaintext
If Feature.IsActive("Inch Thread") Then
    Feature.ThreadDesignation("Inch Thread") = "7/16-14 UNC"
End If

If Feature.IsActive("Metric Thread") Then
    Feature.ThreadDesignation("Metric Thread") = "M10x1.5"
End If
```
Select Case statement

A Select Case statement provides the user with a variety of options by specifying an expression to be tested.

The Case statements that follow in the code compare other expressions to the test expression. The comparison of other expressions is based on its sequence in the program. The first expression that fulfills the test condition (as per the sequence), has its action executed. Once an action is executed, the program moves to the End Select statement. It is good practice in a Select Case statement to use a final Case Else to handle faulty user input.

```
START

Is the value = 3, 4, or 5?

Value = 3  Value = 4  Value = 5

Execute the statements here  Execute the statements here  Execute the statements here

STOP
```
Example with syntax:

```vbnet
Select Case Liquid
    Case "Water"
        Density = 1000
    Case "Turpentine"
        Density = 868.20
    Case "Kerosene"
        Density = 817.15
End Select
```

The `Select Case` statement can also be used without an association with specific variables. It can be used to work with other conditions (such as Boolean conditions) as shown below:

Example with syntax:

```vbnet
Select Case True
    Case Viscosity >= 5
        Density = Viscosity + 10
        Volume = 100/Density
    Case Viscosity < 5
        Density = Viscosity + 5
        Volume = 50/Density
End Select
```
1.2 Project: Use Conditional Statements

1. Using the iLogic Projects.ipj

2. Open the Assignment1_without_rule.ipt file.

3. In the Manage tab, from the iLogic tab, click the Rule Browser.
   - The iLogic Browser appears.

4. Dock the Rule Browser below the Model Browser.
5. In the **Manage** ribbon, from the **iLogic** tab, click **Add Rule**
   - The **Rule Name** dialog appears.
   - Enter **Pattern_Selection**.

6. The **Edit Rule** dialog appears.
7. In the **Edit Rule** dialog, from the **Model** tab, right-click on **Rectangular Pattern1**.
   - The **Capture Current State** option appears.
   - The **Model** and **User** parameters and features appear in the program space.
   - Click **Ok** to save and exit the rule.
8. **Add Text** parameter named **Hole_density**. The parameter is displayed.

   In the **Manage** tab, click **Parameters**. The **Parameters** dialog box appears.

   1. In the **Parameters** dialog, select **Add Text**. A new user parameter slot is created.
   2. Place the cursor in the **Parameter Name** section and name it as **Hole_density**
   3. Note that **Pattern_selection** shows up in the **Driving Rule** column for all the parameters due to the **Capture Current State** performed in a previous step.
   4. Place a check mark in the **Key** column in the **Hole_density** row.

![Image of the Parameters dialog with the Hole_density parameter added and marked as Key]

   - In the **Parameters** dialog, select **Add Text**. A new user parameter slot is created.
   - Place the cursor in the **Parameter Name** section and name it as **Hole_density**
   - Place a check mark in the **Key** column in the **Hole_density** row.
9. Right click **Hole_density** and select **Make Multi-Value** and add the value items **Low**, **Medium** and **High**

1. Right-click the **Equation** section and select **Make Multi-Value**. The **Value List Editor** dialog appears.
2. In the **Add New Items** group, enter 'Low', 'Medium' and 'High' and click **Add**. The values are displayed in the **Value** group box.
3. Click **OK**. You have added a new user parameter.
10. Modify the parameter in Sketch1.
   - Change $d0$ to Width
   - Change $d1$ to Length

11. In the Manage tab, click Parameters. The Parameters dialog box appears.
   - Change $d7$ to RowQuantity
   - Change $d8$ to RowSpacing
   - Change $d9$ to ColumnQuantity
   - Change $d10$ to ColumnSpacing
   - Click Done

12. Right click on Pattern_selection and select Edit Rule.
   - NOTE: All changes made to the parameter names in the previous step are reflected in the rules
13. Use the conditional statement from the panel above the program space to insert conditions in the rule.

- Place an *If* statement.
- Change *My_Expression* to *Hole_density = "Low"*
- Set *ColumnSpacing* equal to 160 mm
- Set *RowSpacing* equal to 160 mm
- Set *ColumnQuantity* equal to 4 ul
- Set *RowQuantity* equal to 4 ul

```
If Hole_density = "Low" Then
  ***Rectangular Pattern1***
  Feature.IsActive("Rectangular Pattern1") = True
  ColumnSpacing = 60 mm
  RowSpacing = 60 mm
  ColumnQuantity = 9 ul
  RowQuantity = 9 ul
End If
```

14. Add a *UpdateAfterChange* snippet

- In the *Snippets* menu, expand the *Parameters* snippet
- Double click on *UpdateAfterChange* snippet.
- The *UpdateAfterChange* snippet is displayed in the program space.
15. Use the **ElseIf...Then** conditional statement from the panel above the program space to insert conditions in the rule.

- Place an **ElseIf...Then** statement.
- Change **My_Expression** to **Hole_density = "Medium"**
- Set **ColumnSpacing** equal to **120 mm**
- Set **RowSpacing** equal to **120 mm**
- Set **ColumnQuantity** equal to **5 ul**
- Set **RowQuantity** equal to **5 ul**
- Add the **UpdateAfterChange** snippet

```plaintext
If Hole_density = "Low" Then
  ' ***Rectangular Pattern1***
  Feature.IsActive("Rectangular Pattern1") = True
  ColumnSpacing = 160 mm
  RowSpacing = 160 mm
  ColumnQuantity = 4 ul
  RowQuantity = 4 ul
  Parameter.UpdateAfterChange = True
ElseIf Hole_density = "Medium" Then
  ' ***Rectangular Pattern1***
  Feature.IsActive("Rectangular Pattern1") = True
  ColumnSpacing = 120 mm
  RowSpacing = 120 mm
  ColumnQuantity = 5 ul
  RowQuantity = 5 ul
  Parameter.UpdateAfterChange = True
End If
```
16. Use the **Else** conditional statement from the panel above the program space to insert conditions in the rule.

- Place an **Else** statement.
- Set `ColumnSpacing` equal to 60 mm
- Set `RowSpacing` equal to 60 mm
- Set `ColumnQuantity` equal to 9 ul
- Set `RowQuantity` equal to 9 ul
- Add the **UpdateAfterChange** snippet

```plaintext
If Hole_density = "Low" Then
    ' ***Rectangular Pattern1***
    Feature.IsActive("Rectangular Pattern1") = True
    ColumnSpacing = 160 mm
    RowSpacing = 160 mm
    ColumnQuantity = 4 ul
    RowQuantity = 4 ul
    Parameter.UpdateAfterChange = True
ElseIf Hole_density = "Medium" Then
    ' ***Rectangular Pattern1***
    Feature.IsActive("Rectangular Pattern1") = True
    ColumnSpacing = 120 mm
    RowSpacing = 120 mm
    ColumnQuantity = 5 ul
    RowQuantity = 5 ul
    Parameter.UpdateAfterChange = True
Else
    ' ***Rectangular Pattern1***
    Feature.IsActive("Rectangular Pattern1") = True
    ColumnSpacing = 60 mm
    RowSpacing = 60 mm
    ColumnQuantity = 9 ul
    RowQuantity = 9 ul
    Parameter.UpdateAfterChange = True
End If
```
17. Add a Comment at the beginning of the Rule describing what the Rule does. Refer to the image to compare if the rule you created matches.

| 'The execution of this rule enables you to choose between 3 values of hole density |
| ' - High, Medium, and Low through a list box |
| If Hole_density = "Low" Then |
| ' ***Rectangular Pattern1*** |
| Feature.IsActive("Rectangular Pattern1") = True |
| ColumnSpacing = 160 mm |
| RowSpacing = 160 mm |
| ColumnQuantity = 4 ul |
| RowQuantity = 4 ul |
| Parameter.UpdateAfterChange = True |
| ElseIf Hole_density = "Medium" Then |
| ' ***Rectangular Pattern1*** |
| Feature.IsActive("Rectangular Pattern1") = True |
| ColumnSpacing = 120 mm |
| RowSpacing = 120 mm |
| ColumnQuantity = 5 ul |
| RowQuantity = 5 ul |
| Parameter.UpdateAfterChange = True |
| Else |
| ' ***Rectangular Pattern1*** |
| Feature.IsActive("Rectangular Pattern1") = True |
| ColumnSpacing = 60 mm |
| RowSpacing = 60 mm |
| ColumnQuantity = 9 ul |
| RowQuantity = 9 ul |
| Parameter.UpdateAfterChange = True |
| End If |
18. Change the **Hole_density** parameter in the **Parameters** dialog box.
   - Select **Medium** from the pull down menu.
   - Select **Low** from the pull down menu.
1.3 Project: Use iLogic Wizards

19. In the **Manage** tab, click **Parameters**. The **Parameters** dialog box appears.
   - Change $d_2$ to **Thickness**
   - Change $d_4$ to **HoleDiameter**
   - Click **Done**

20. In the **Manage** ribbon, from the **iLogic** tab, click **Add Rule**
   - The **Rule Name** dialog appears.
   - Enter **Hole_Size**.

21. The **Edit Rule** dialog appears.
22. In the **Edit Rule** dialog box, from the **Wizards** tab, click on **Parameter Limits**.
   - The **iLogic Limits Wizard** dialog box appears.
   - Enter **Name** equal to **HoleDiameter**.
   - Enter **Max. Value** equal to **50**.
   - Enter **Min. Value** equal to **10**.

   ![iLogic Limits Wizard](image)

   - Click **Apply**.
   - An If-Then-Else statement is generated in the program space according to the parameters entered in the wizard.

   ![Generated If-Then-Else Statement](image)

   - Click **Close** to exit the wizard.
   - Click **Ok** to save and exit the rule.
23. Change the **HoleDiameter** parameter in the **Parameters** dialog box.
   - Set **HoleDiameter** to 5.
   - The **Minimum Value Rule** warning dialog is displayed.
• Click OK.
• The HoleDiameter parameter is reset to the minimum value and the model is updated accordingly.
- Set **HoleDiameter** to 55.
- The **Maximum Value Rule** warning dialog is displayed.
• Click **OK**.
• The **HoleDiameter** parameter is reset to the maximum value and the model is updated accordingly.

• The File result has been provided… it is named *Assignment1_with_rule.ipt*