AUTODESK UNIVERSITY 2013

Single Minute Exchange of Design

Yudhistira Hasannudin – Hasannudin Bersaudara Maju

MA1581 Single-minute exchange of design provides a way to cut design time to a single digit using iLogic and the Visual Basic (VBA) application programming interface to Autodesk Inventor software. We cover basic programming in VBA and iLogic. We explain how to combine VBA and iLogic for design automation.

Learning Objectives

At the end of this class, you will be able to:

- Perform basic programming
- Use the Visual Basic interface to Inventor
- Use an iLogic component in Inventor
- Automate a design in Inventor

About the Speaker

Yudhistira or people know him as Yudhi graduated from Mercubuana University, Master of Industrial Engineering program in 2009 and from State Polytechnic of Bandung, Mechanical Engineering program in 2006. He started working as an Application Engineering and Autodesk Approve Instructor at one of Autodesk Gold Partner in Indonesia from 2006 to 2011. He was the first person to get Autodesk Inventor Associate Certificate from Indonesia, 2010. At 2011 he start his own company, Hasannudin Bersaudara Maju (HBM), a service company that provide Design and Engineering Services, such as CAD Training, Re-drawing, Re-Design, Prototype Maker, etc.

Email: yudhidesign@yahoo.co.id

Introduction

What is SMEDg (Single Minute Exchange of Design)

For you all who work in Industrial Engineering/Management maybe have heard about SMED. SMED or Single Minute Exchange of Die is one of the many lean production methods for reducing waste in a manufacturing process. It provides a rapid and efficient way of converting a manufacturing process from running the current product to running the next product.

This system is developed by DR. Shigeo Shingo, one of the Toyota's industrial consultants. He sees there are a major lost/waste in a production Setup. Therefor he develops a system that can cut a setup time, so the production cost can be reduce. The system that he develop also known as SMED (Single Minute Exchange of Die/tools).

It is also often referred to as Quick Changeover (QCO). Performing faster change-overs is important in manufacturing, or any process, because they make low cost flexible operations possible. However SMED's utility of is not limited to manufacturing. It also can be applied in design process, we called SMEDg (Single Minute Exchange of Design), a methods for reducing design time in a product development.

Why using SMEDg (Single Minute Exchange of Design)

In product development, there are two kind of development. The first is developing a new kind of product. Second, develop a same product, but have a different parameter, such as different size, capacity or length; this is what we called a variant product.

In fact, lots of product that has been developed in Industry is a variant product. For example, a company that manufacture a standard screw conveyor. Their product shape maybe all the same, but the different between their product is only in the parameters of screw conveyor.

Example:
The first product has parameter:
Length: 1500 mm
Outer Diameter: 250 mm



Second product has parameter:
Length: 2000 mm
Outer Diameter: 200 mm

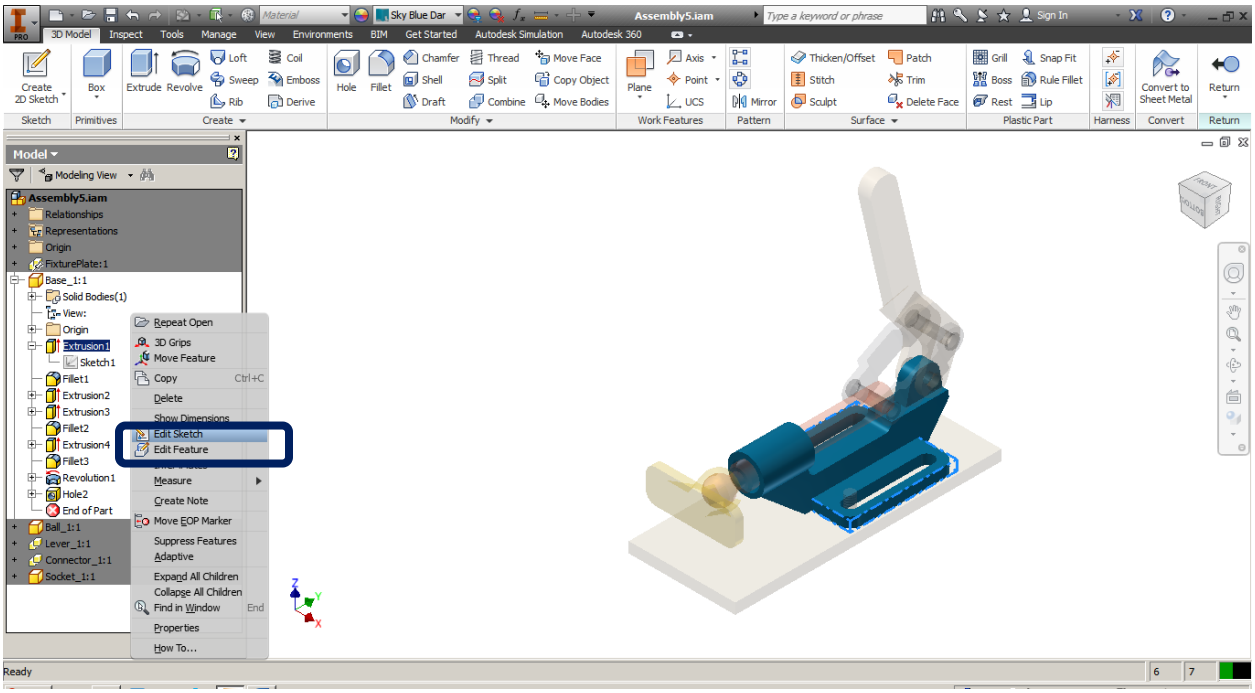


Even though their shape almost the same but to develop a variant product can take lots of time.

First scenario, if they used AutoCAD. In 2D, even though they use a template and to make a new product they just edit the template, still it take lots of time to edit the drawing. And to write a BOM they must write it manually. This can make a room for error, example if they write BOM manually; they could be by mistake wrote a wrong material name or wrong quantity of material.

Second, if they used parametric software, such as Autodesk Inventor. This can save lots of time. Inventor we can easily edit the design by editing the parameter that they have. Also we can automatically make a BOM. 2D drawing will be produce automatically also.

But, to edit a parameter in Autodesk still can also take lots of time. To edit parameter in Autodesk first, you must search the feature of parameter that will be edit in browser. Then right-click chooses what you want to edit, a feature or sketch. You must do this step to edit all parameter needed. This can be messy and take lots of time if you have to edit lots of parameter, such as a large assembly design.



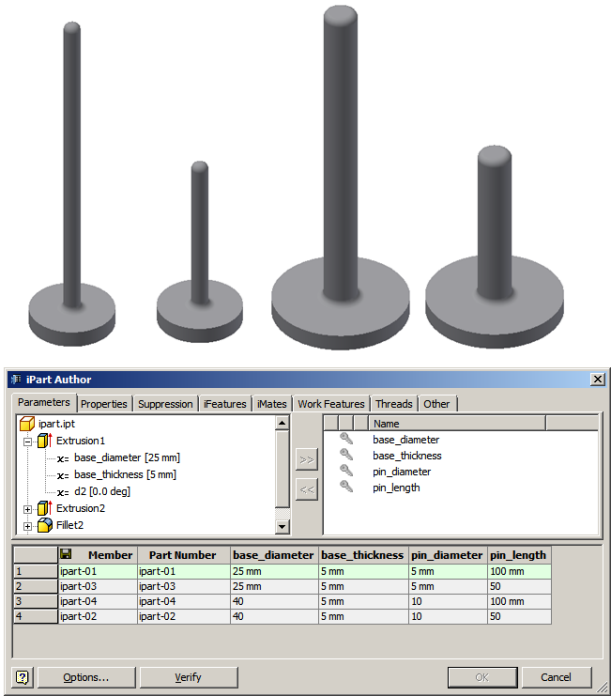
Design Automation

So, we must find a way to change a parameter in such easy way and in a short time. This can be happen if we used design automation in Autodesk Inventor. There are four kind of design automation in Autodesk Inventor:

- iPart
- Excel Driven Design
- VBA
- iLogic

iPart

An iPart is a table driven master part that configures standard parts to different sizes and states. The table can be edited within Inventor or externally in a spreadsheet.



Each row can control feature state (enabled or suppressed), and many other variables such as feature size, appearance, material, and part number. Table driven iFeatures can also be included in an iPart table.

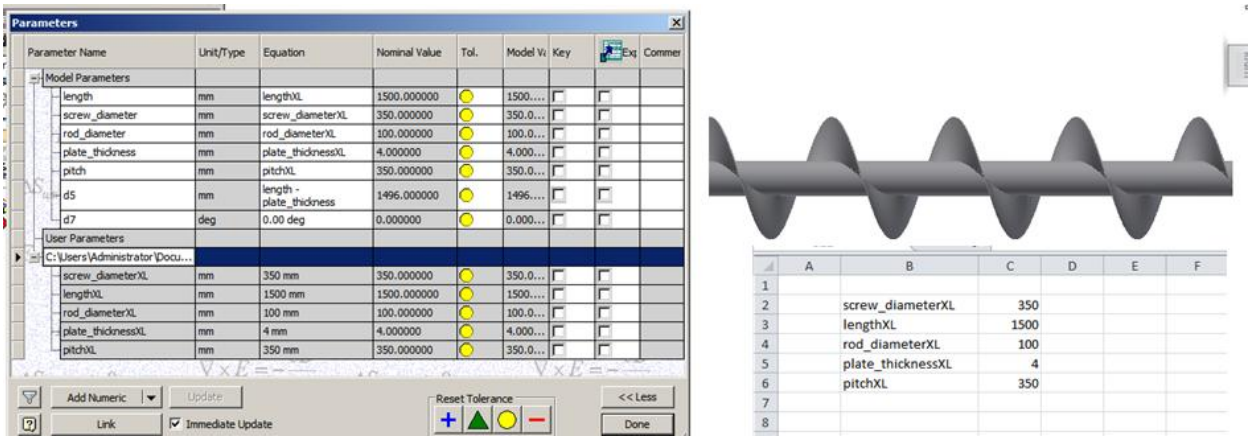
There is two kind iPart, Standard iPart and Custom iPart.

- A Standard iPart Factory defines all values in columns. When you publish an iPart using a standard iPart factory, the member parts cannot be modified after placement.
- A Custom iPart Factory contains at least one column identified as a Custom Parameter Column. When you publish an iPart using a custom iPart factory, the custom parameters in the member parts can be modified when the iPart member is placed. You can add features to a custom iPart member.

Excel Driven Design

If you use the same parameters in many models, you can define parameters in a Microsoft Excel spreadsheet. You can embed or link to the spreadsheet from assembly or part models.

In an assembly file, you can also link to a part file (.ipt). When you click Link and select the desired part file, the assembly is associatively linked with the exported parameters from the .ipt file.



Excel driven design is more flexible then iPart. We can use an excel equation capability. But, only numeric parameters can be linked from spreadsheets. We cannot link a text type, for example we cannot link and change material name from spreadsheets.

VBA/Macro

Visual Basic for Applications (VBA) is a programming environment developed by Microsoft and licensed by other companies to integrate into their applications. Autodesk licenses VBA and includes it in AutoCAD and Inventor. This provides all customers of Inventor with a free development environment. Below are the minimal steps to begin using VBA and writing a simple macro.

You can access VBA through Inventor using the Macro | Visual Basic Editor command in the Tools menu, or by pressing Alt - F11. You can choose which macro you want to run. In VBA, we have more flexibility rather then iPart or Excel driven design. VBA has capability to customize almost all the inventor component.

In VBA we can design our own custom form, so this can make us easly to work with and more attractive in visualization.

As mentioned earlier, VBA is using Visual Basic language program. So to work with VBA we must understand a visual basic language. For some engineer or designer this could be a problem.

iLogic

The last one is iLogic. In a simple word, iLogic is a simplified visual basic. In here, just like in VBA, we can customize almost every component of Inventor. iLogic is using a simple language programming.

We can make parameter rule driven in iLogic. A parameter that change based on a rules that driven by another parameter.

iLogic embeds rules as objects directly into part, assembly, and drawing documents. The rules determine and drive parameter and attribute values for your design. By controlling these values, you can define behavior of the attributes, features, and components of a model. Knowledge is saved and stored directly in the documents, like how geometric design elements are stored.

iLogic rules can utilize custom parameter types now available in Autodesk Inventor, such as text, true/false, and multi-value lists. You can use these parameter types to write rules that involve more than numeric input values.

The Autodesk Inventor Parameters dialog box supports these specialized parameters, with advanced filtering functions to assist in parameter input definition, management, and editing.

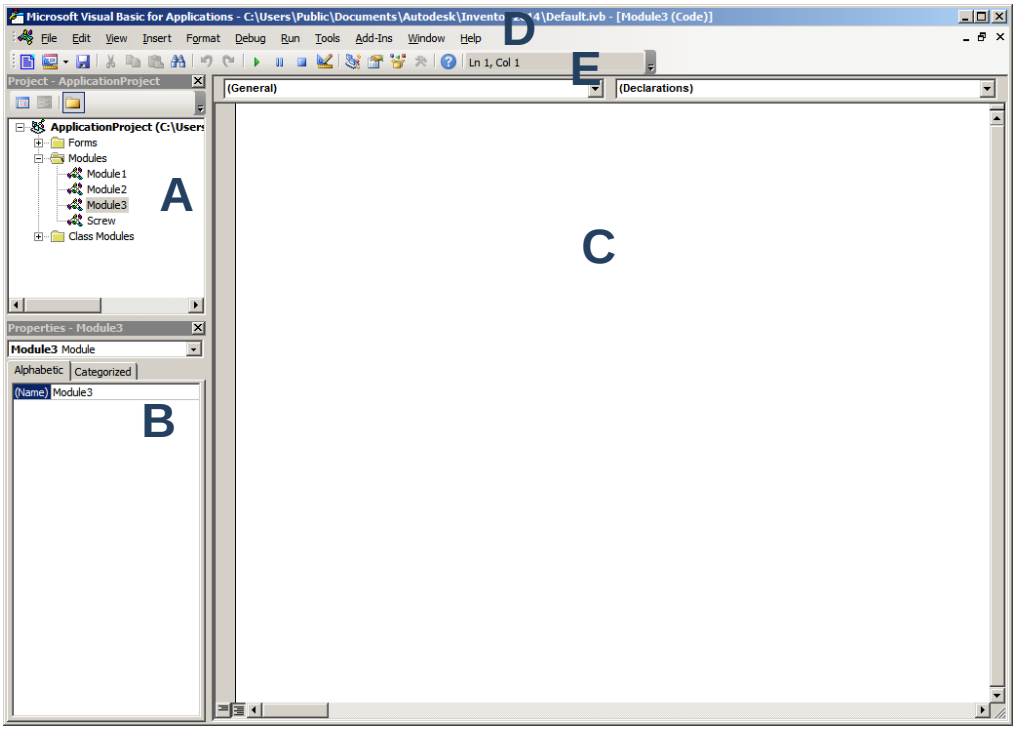
Visual Basic Application (VBA)

Visual Basic for Applications (VBA) is a programming environment developed by Microsoft and licensed by other companies to integrate into their applications. There are two feature in inventor related to VBA, “macro” and “VBA Editor”

VBA Editor

To build your Microsoft® Visual Basic® for Applications (VBA) program, you create a VBA project using the Visual Basic Editor.

Visual Basic Editor: the VBA development environment:



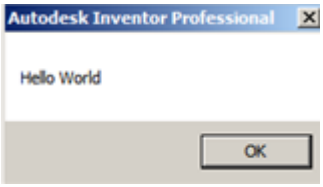
A. The **Project Explorer** displays a list of projects and project items in Autodesk Inventor documents.
B. The **Properties** window displays a list of the properties for the selected item.
C. The **Code** window
D. The menu bar displays the commands you use to build, run, and debug your program.
E. The toolbar provides quick access to commonly used commands in the development environment.

Create a new simple programming

- First, you must click a module in project explorer (module 1).
- Then write the code that you want in the code window
- In this example I want to make a simple program to write “Hello World”

```
Public Sub HelloWorld()
    MsgBox ("Hello World")
End Sub
```

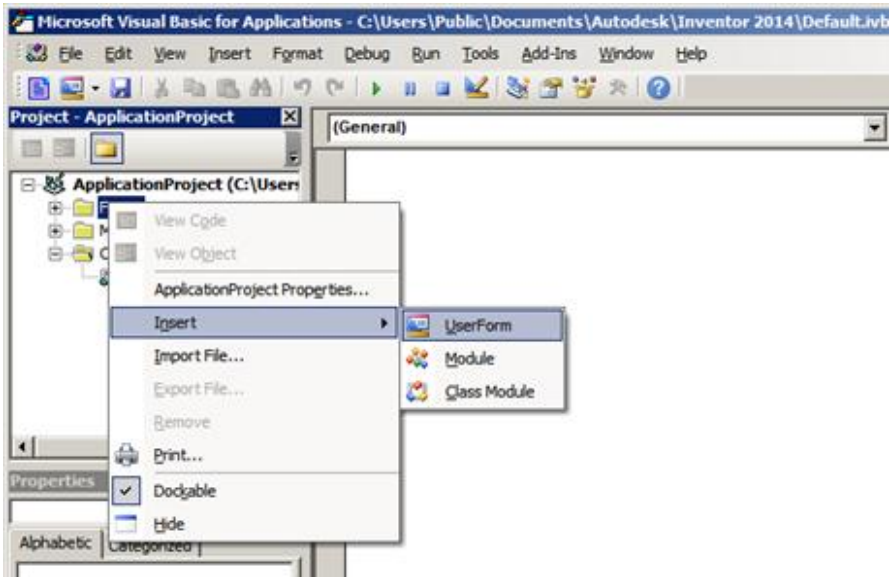
Click run on the toolbar to try our code.



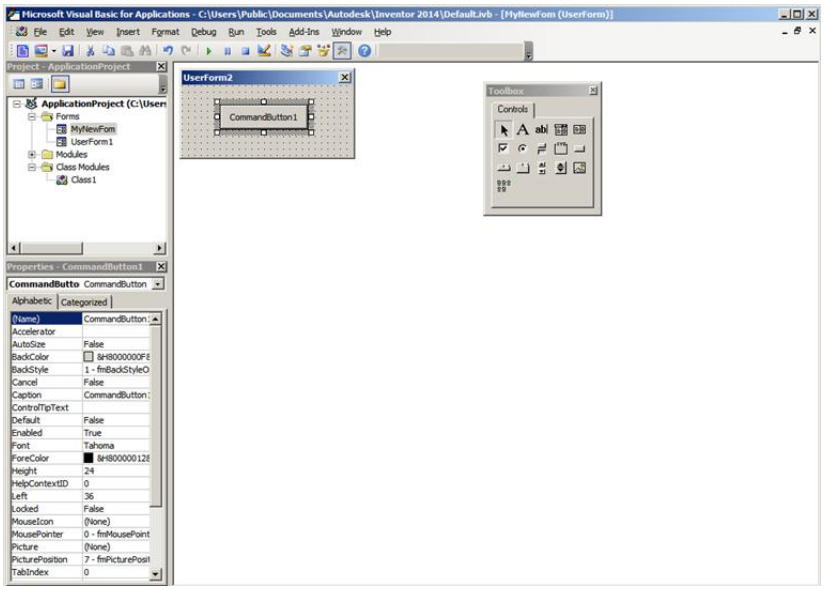
Using User Form

You can also design your own custom form to visualize your program. Here is the example to use a user form:

First you must make a new form. In Project Explorer, right click Form > Insert > User Form



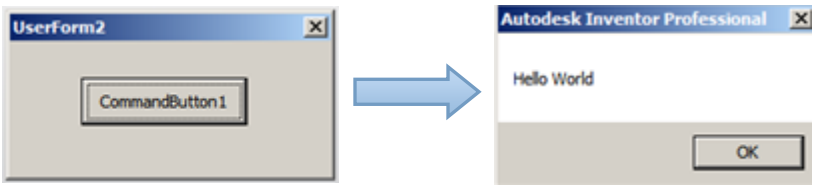
Next, you can add a command button from a toolbox. By default caption in button is “CommandButton1”. You can change the caption of button in button properties.



If I want the text is show when I click the button, then double click the button to put the code. Add the code that we want. In this case:

```
Private Sub CommandButton1_.click()
        MsgBox (“Hello World")
End Sub
```

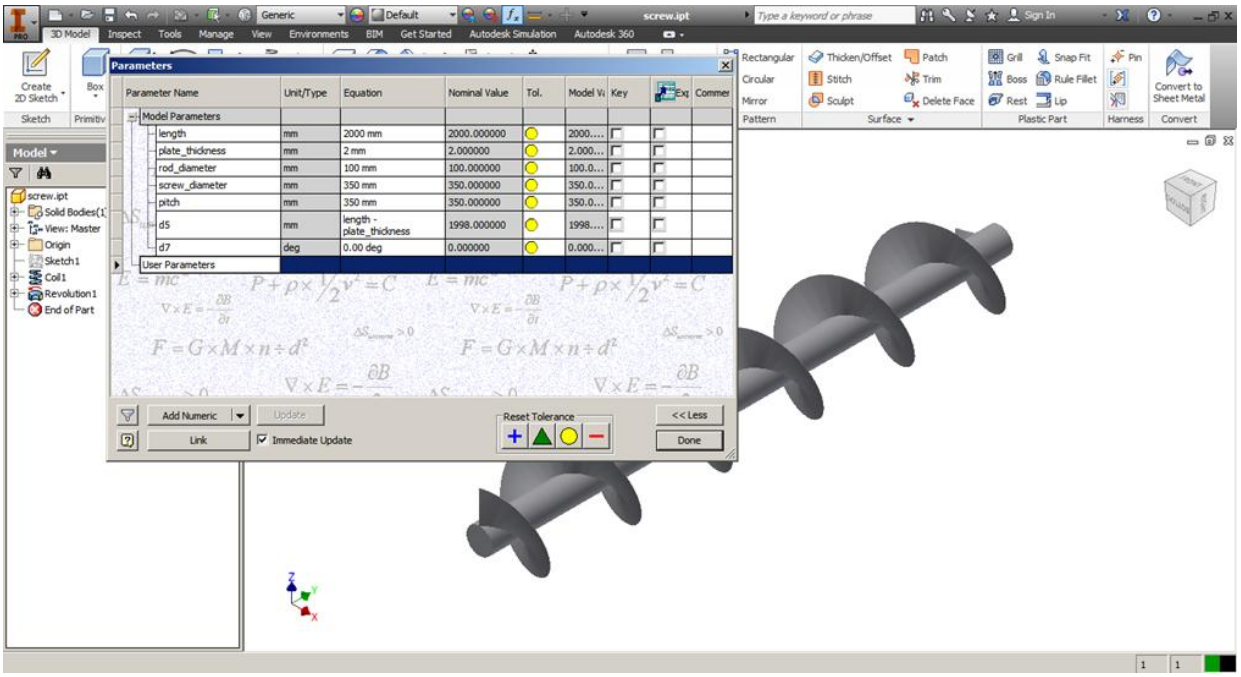
If we run the program, we can see first the form with command button. After we click the button, a message box with a text “Hello World” will be show to.



Setting parameters using VBA

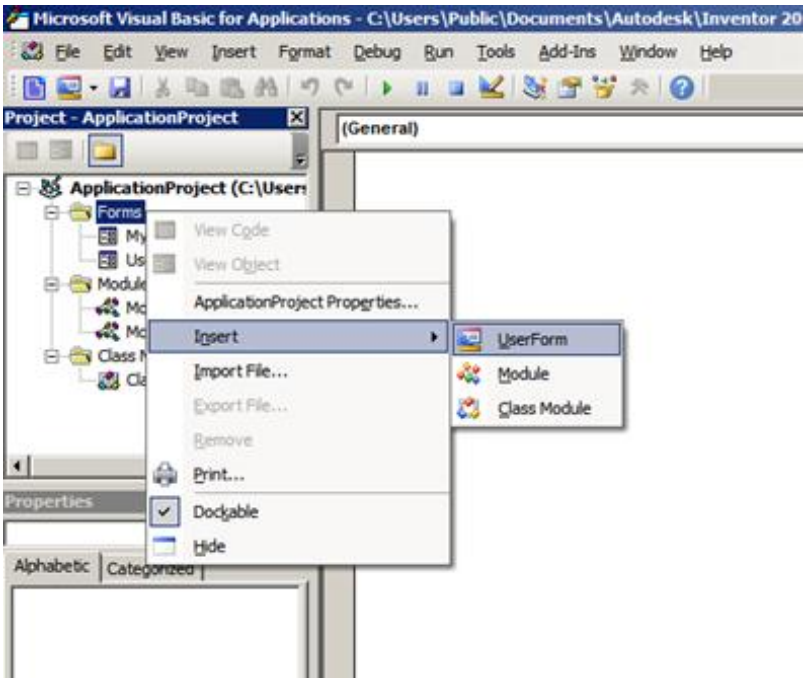
First design a part with all the parameter. In this example I will use a screw conveyor as part with parameter:

- screw_diameter : 350 mm
- length : 2000 mm
- rod_diameter: 100 mm
- plate_thickness: 2 mm
- pitch: 350 mm

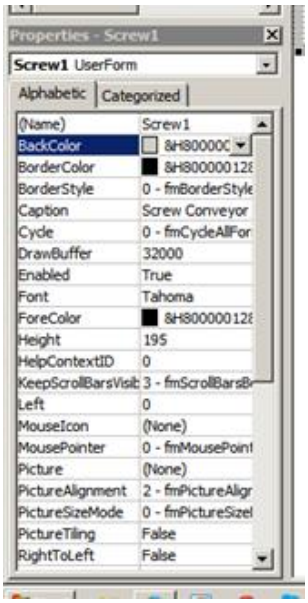


Next, go to ribbon menu tab “Tools” > “VBA Editor” to enter VBA Editor.

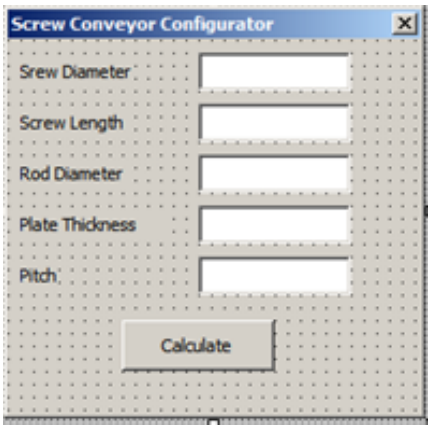
After we move to VBA editor, we must make a new user form. In project explorer, right click “Forms”>”Insert”>”User Form”.



Edit user form properties with:
Name: Screw1
Caption: Screw Conveyor Configurator



After that, we must resize the form and add some of component in user form with label, textbox and command button. Change the caption of labels and command button like the picture below.



To put a code, double click “commandbutton1”. The visual in VBA editor will be change from form editor into form code. First, is to make our program have access to part document, parameters, and model parameters

Code to access part document:

```
Dim InvDoc As PartDocument
Set InvDoc = ThisApplication.ActiveDocument
```

Code to access parameters:

```
Dim InvPar As Parameters
Set InvPar = InvDoc.ComponentDefinition.Parameters
```

Code to access model parameters:

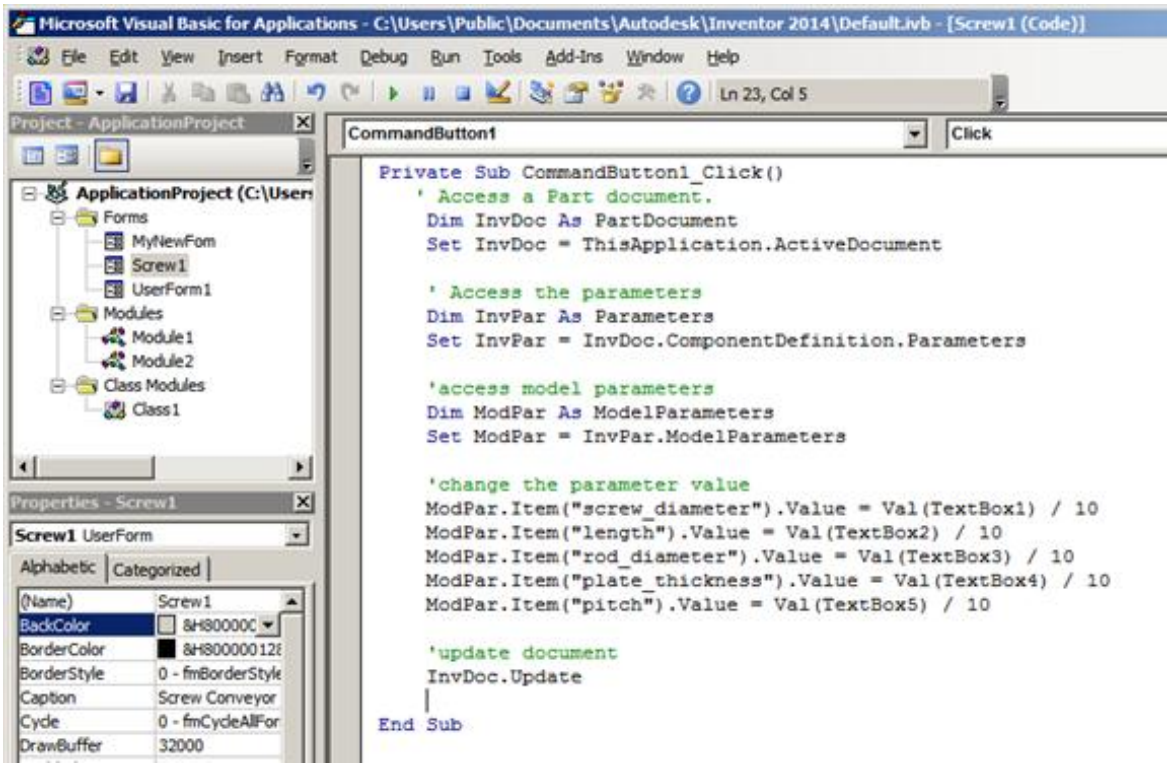
```
Dim ModPar As ModelParameters
Set ModPar = InvPar.ModelParameters
```

After our program have access to model parameters, and then we can set a parameter with any value. First, our program will read a value from textbox in user form, and then our program will set the parameter with value that has been read before. Here is the code:

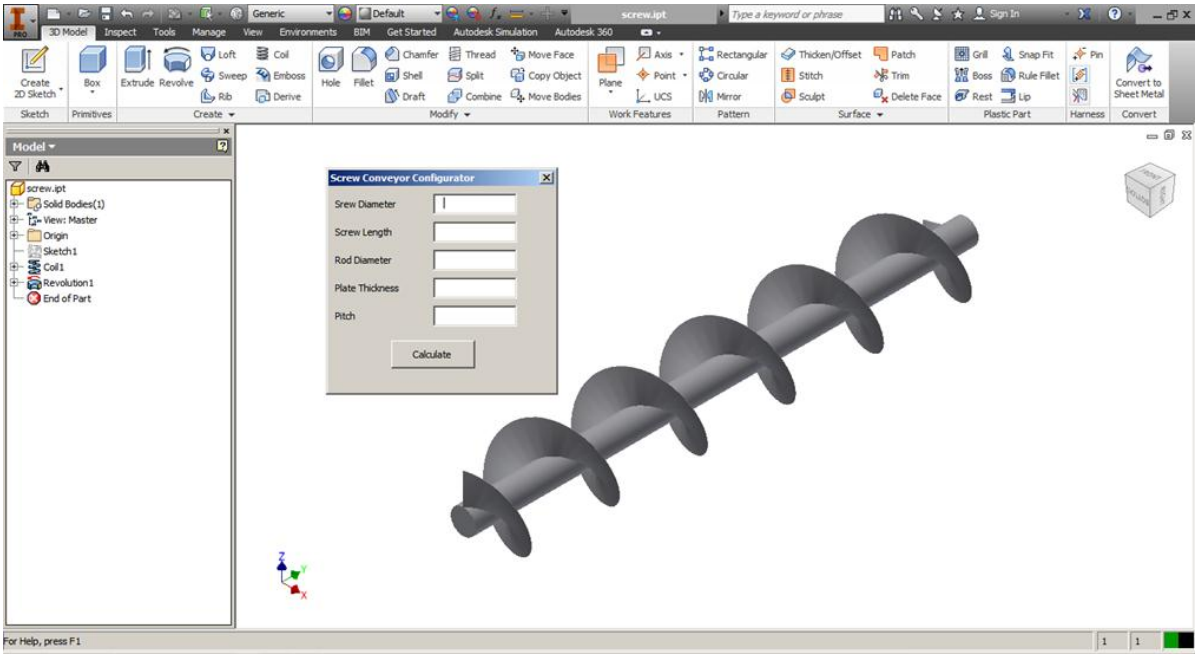
```
ModPar.Item("screw_diameter").Value = Val(TextBox1) / 10
ModPar.Item("length").Value = Val(TextBox2) / 10
ModPar.Item("rod_diameter").Value = Val(TextBox3) / 10
ModPar.Item("plate_thickness").Value = Val(TextBox4) / 10
ModPar.Item("pitch").Value = Val(TextBox5) / 10
```

And don't forget to put code for updating the document. This is to make sure after the parameter have been set, part model will be change based on the parameter value. Here is the code:

```
InvDoc.Update
```



Now we try to run the program. Click run on the toolbar.



Notice that the form column is blank, it doesn't read the parameter value. Close the form to go to VBA Editor.

To make our form read the existing parameter value when it activate, we must add another sub program in our form. Double click the command button to switch into code window. Add a sub program with a trigger when user form is activated. Here is the code:

```
Private Sub UserForm_Activate()

End sub
```

Then like the sub program before, we must access part document, parameters and model parameter that we want to read. After we have the access then we can read the model parameter value and write it into textbox at our user form. Here the complete sub-program to read value of model parameter when user form is activates:

```
Private Sub UserForm_Activate()

  ' access a Part document.
    Dim InvDoc As PartDocument
    Set InvDoc = ThisApplication.ActiveDocument

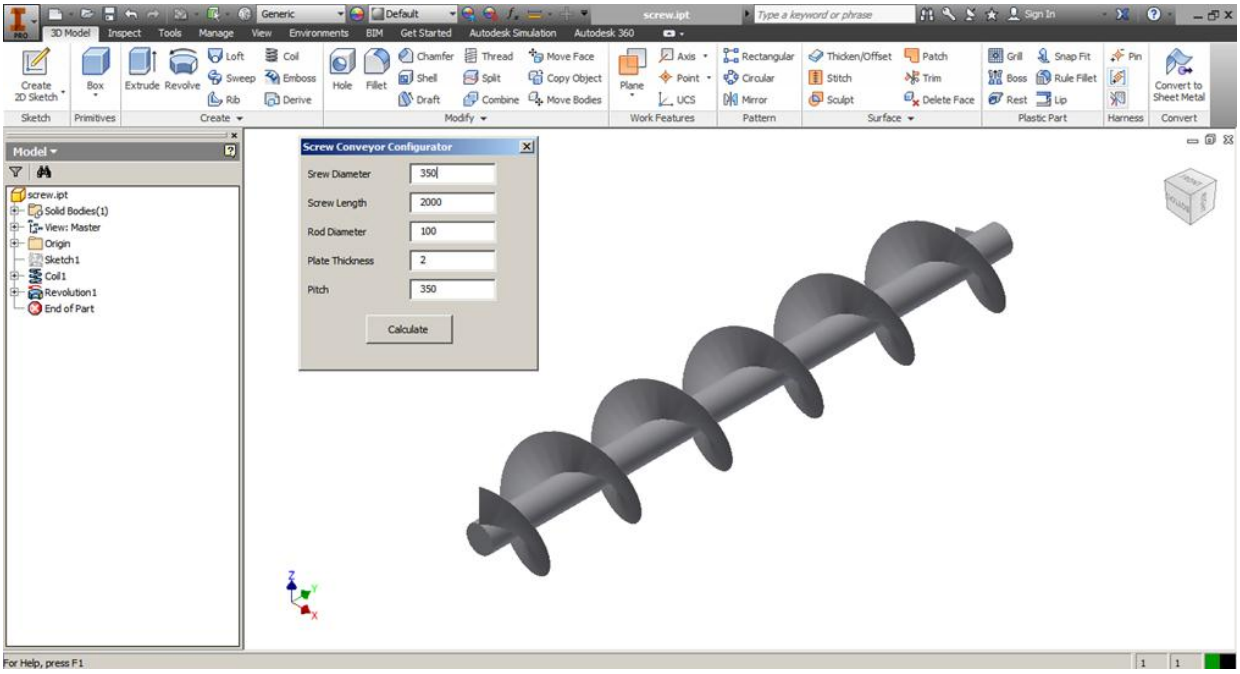
  ' access the parameters
    Dim InvPar As Parameters
    Set InvPar = InvDoc.ComponentDefinition.Parameters
```

```
'access model parameters
  Dim ModPar As ModelParameters
  Set ModPar = InvPar.ModelParameters

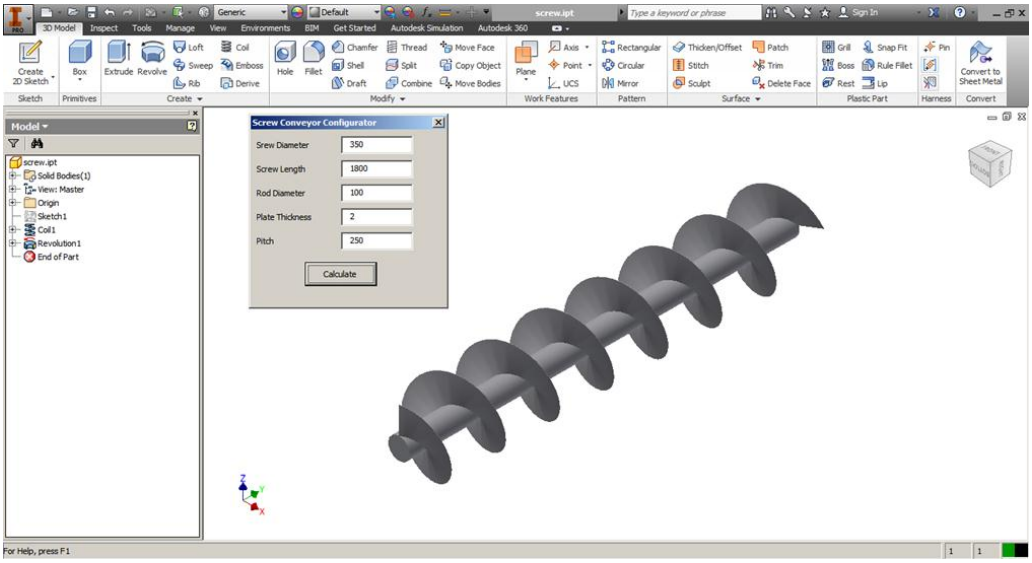
'read the parameter value to textbox
  TextBox1.Text = (ModPar.Item("screw_diameter").Value) * 10
  TextBox2.Text = (ModPar.Item("length").Value) * 10
  TextBox3.Text = (ModPar.Item("rod_diameter").Value) * 10
  TextBox4.Text = (ModPar.Item("plate_thickness").Value) * 10
  TextBox5.Text = (ModPar.Item("pitch").Value) * 10

End Sub
```

We run the program again. Now we can see the program can read model parameter value when the form is activate.



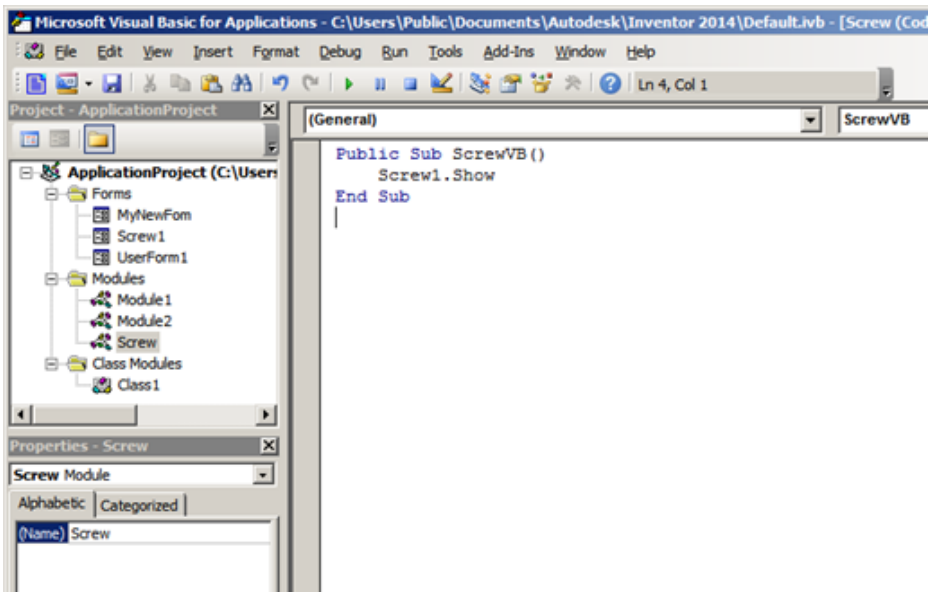
And also if we change the value in user form, our part is also change based on the value that we have set in user form.



But still, we cannot run the program through run macro. We must add some code in VBA module to show the user form.

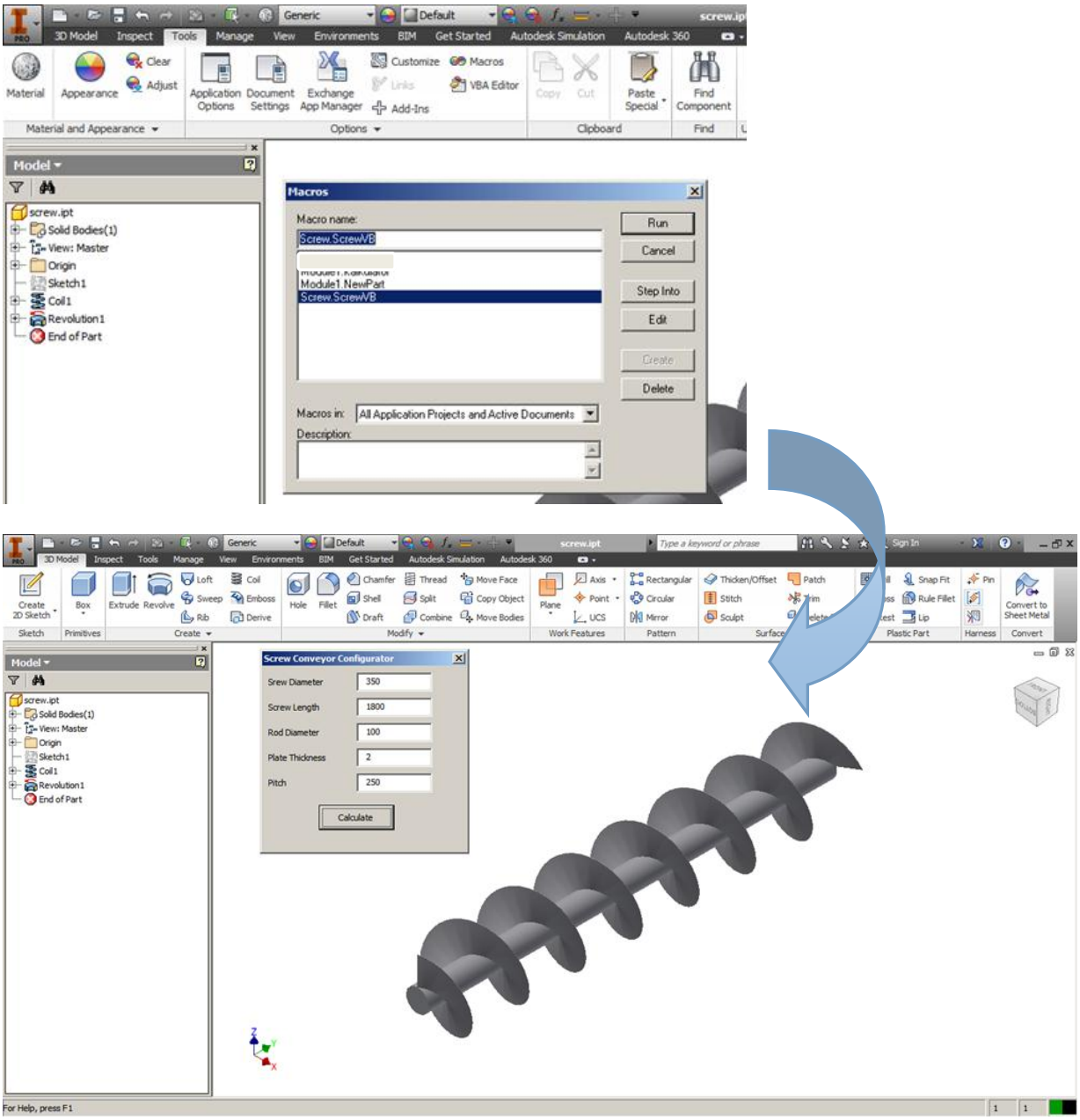
In project browser, right click Module > Insert > Module, to make a new module. We can rename the module, in this example I will rename it: ”Screw”. Then add this code into the module:

```
Public Sub ScrewVB()
        Screw1.show
End Sub
```



Save the project in VBA editor. Then close it.

Now we try to run macro “ScrewVB”. Go to ribbon tab “Tools” > “Macro”’. Select the macro name “Screw.ScrewVB”, click run.

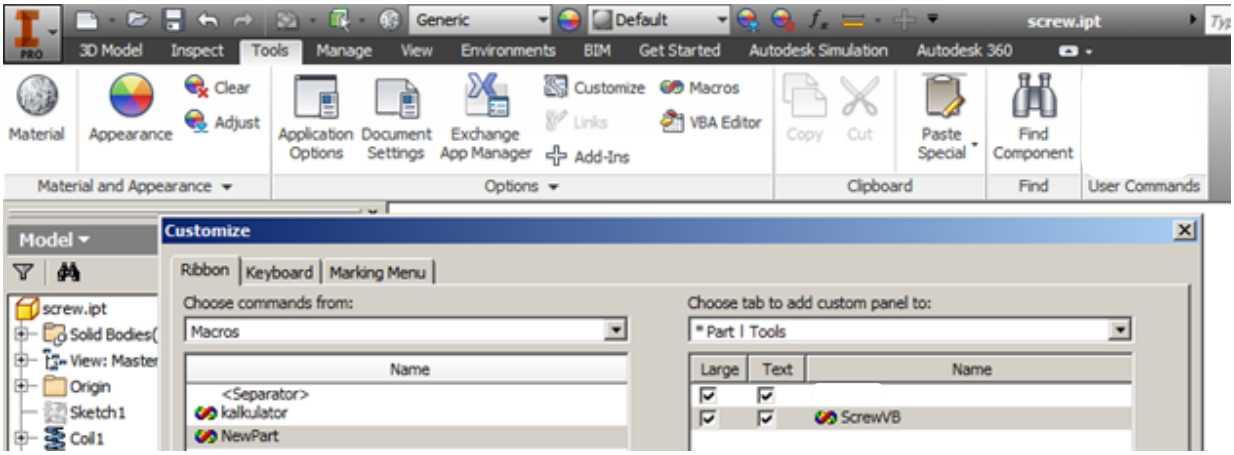


To Make Ribbon Menu Button

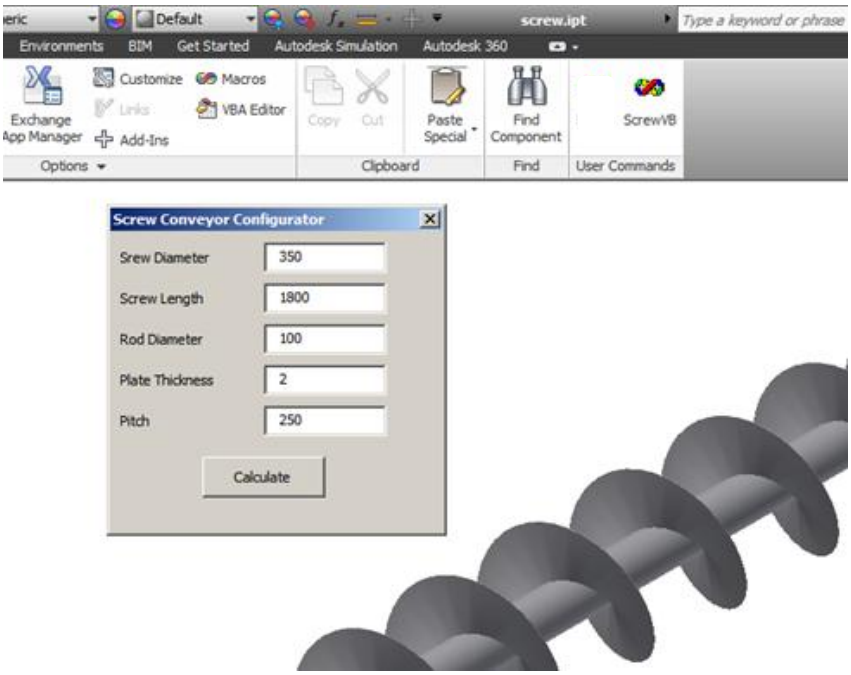
How to make a button that can run our ScrewVB macro through Ribbon Menu. First go to “Tools” tab > “Options” panel > “Customize, this will open customize window.

In customize window, go to tab “Ribbon”, in right collom (Choose command from), select “macro”. In left colom (choose tab to add custom panel to), select “ * Part | Tools”.

Then, select macro name that we want to add, in this example “ScrewVB” and click “add” button to add the macro in part environments at “tools” tab. Click “OK” to finish the job.



Now we can see there is a new button in tools tab, “ScrewVB” button. If click it will run the program that we have develop before.



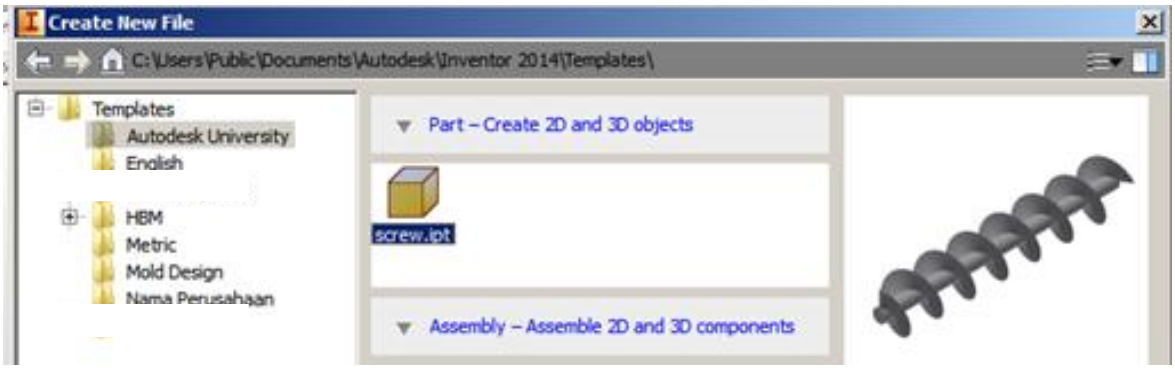
Add a new template

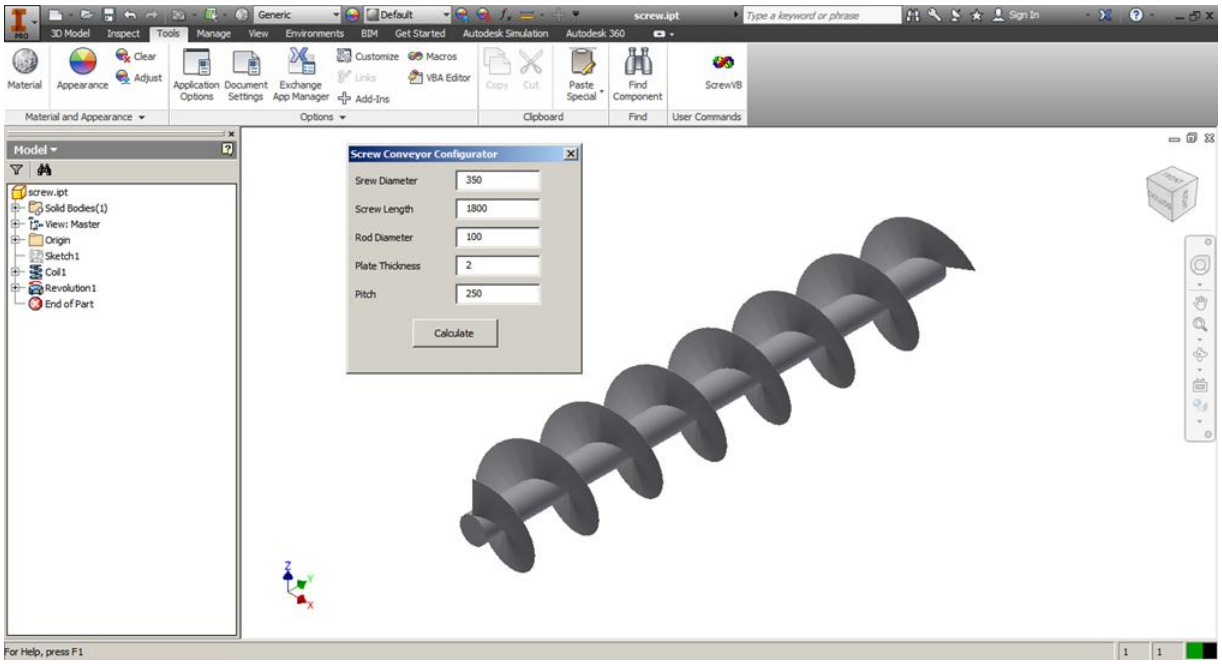
To finish our SMEDg system using VB, next we must save our model into template. So if we want to develop a new screw conveyor, we can load the template and change the value using macro that we have developed before.

To make a new template, go to Application Menu (upper left corner menu), “Save as” > “Save copy as template”.

Then make a new folder (example: “University” folder), and save model inside the folder.

Now, we can try to make a new screw conveyor part and if we want to change value of model parameter, just click “ScrewVB” in “tools” tab at the ribbon menu.

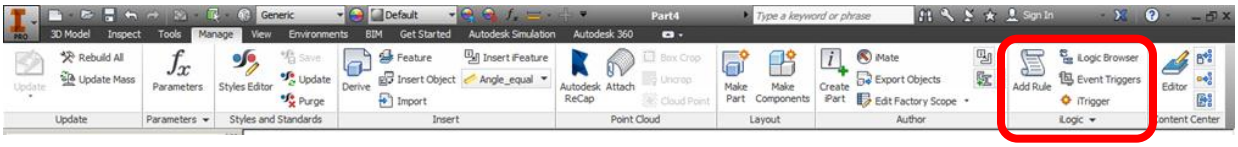




iLogic

How about iLogic? How to make a design automation using iLogic?
Like I have explained before, iLogic is a simplified VBA. Where in iLogic, lots of VBA sintax has been simplified into a simple logic language program. All we need is a simple logic, no programming experiences require.

Before we continue, for information we can access iLogic menu in “Manage” tab > “iLogic” panel. There are some menus related to iLogic, such as “add rules”, “iLogic browser”, “event trigger”, etc.



The most important is “iLogic Browser”. If we activate, iLogic browser will appear by default upper side of “browser”. iLogic browser is consist of “rules”, “form”, “global forms” and “external rules”.



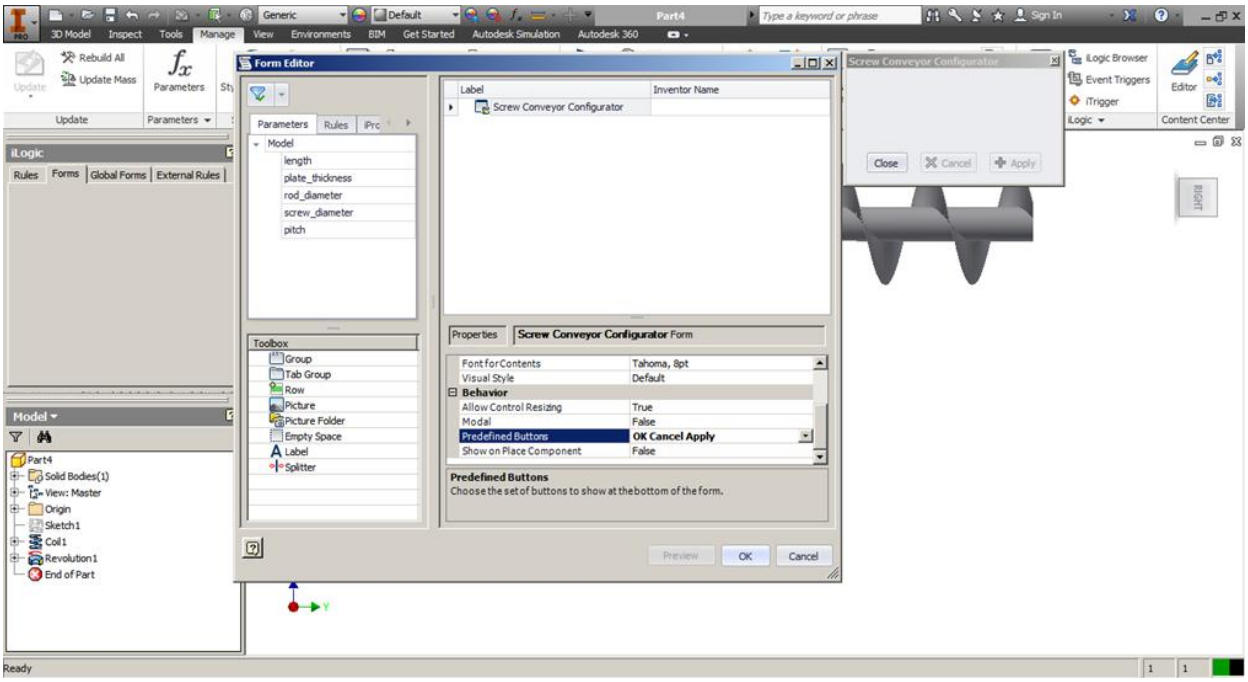
How to set a parameter using iLogic

To set a parameter in iLogic is very simple. We don’t have to access part document, parameter and model parameter using code just like in VBA.

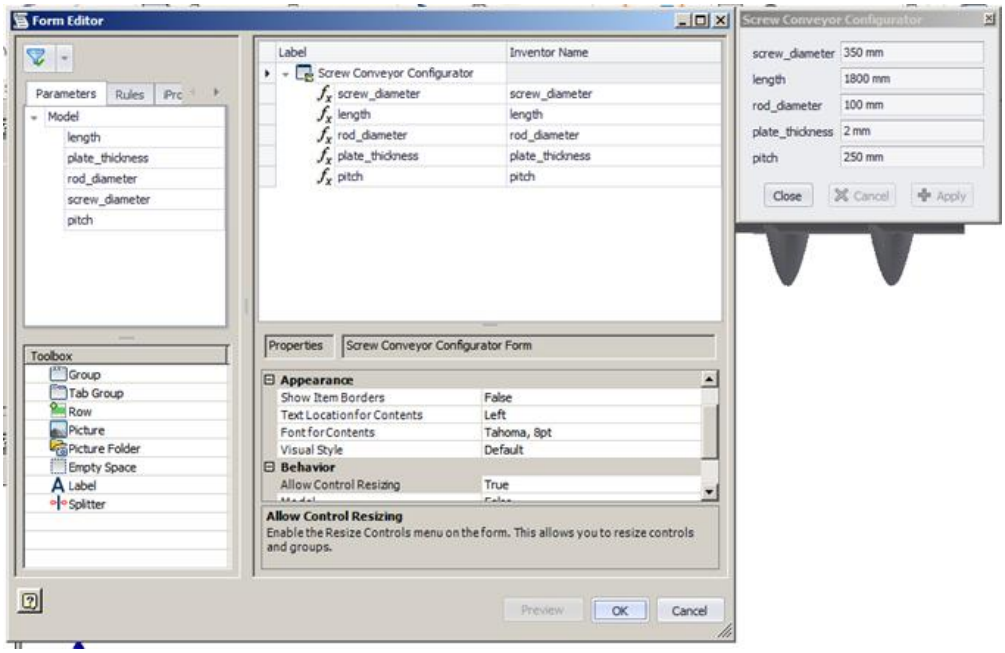
First, activate “iLogic Browser”, go to “Manage” tab > “iLogic” panel > “iLogic Browser”

Then in iLogic browser, go to “form” tab, right click “add form”. This will make a new user form in iLogic and open a form editor. Also, we can see next to form editor we have a preview for the form that we design.

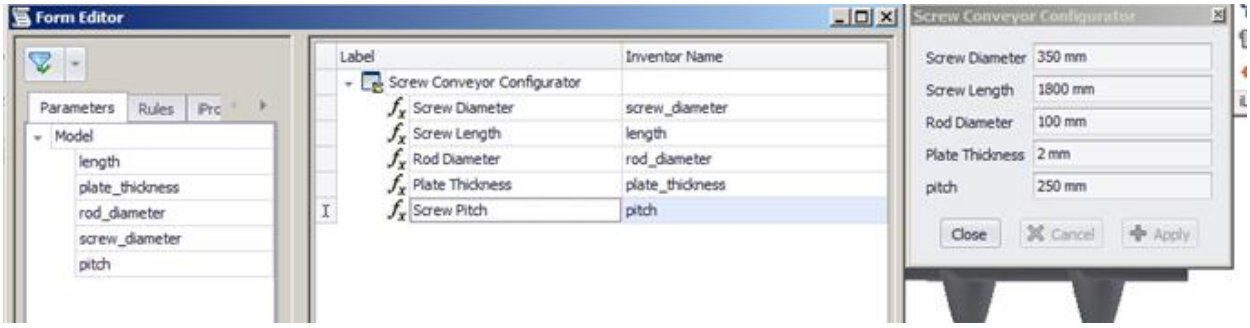
Next, we have to change visualization of form that we will use. We can change the label of form. (Example: “Screw Conveyor Configurator”). Also if we want to change the button type, go to “Behavior” > “Predefine Button” > change to “OK Cancel Apply”.



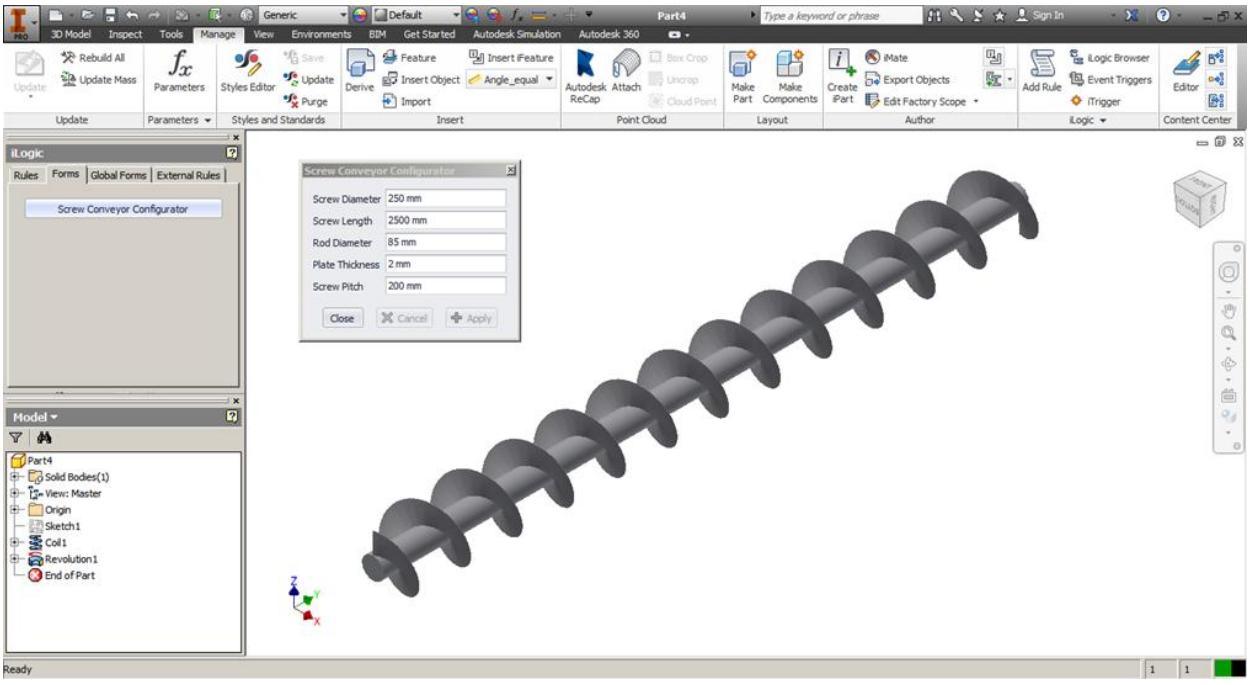
After that, you just simply drag and drop a parameter name that you want to change into form. We can see in the form preview, all the parameter we select has been inserting into our user form.



To make this more user friendly and pritier, we can edit the label of parameter into a word that we can understand.



After everything is done, click “OK” to finish. Now, we can see in “form” tab at iLogic browser, there is a button with caption “Screw Conveyor Configurator”. We can click the button to show iLogic form that we have been develop before. If we change parameter value in iLogic form, our part model will also change based on parameter value in iLogic form.

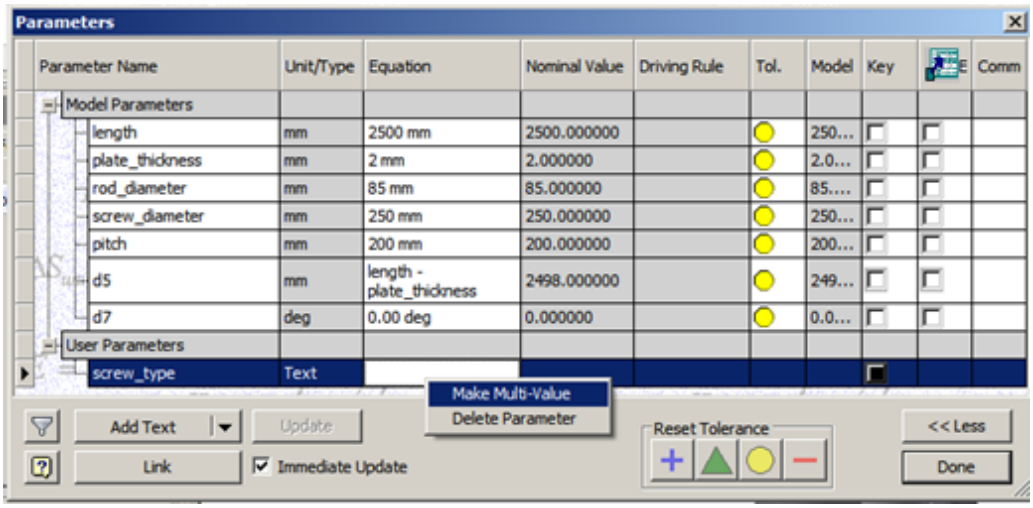


Rule Based Parameter in iLogic

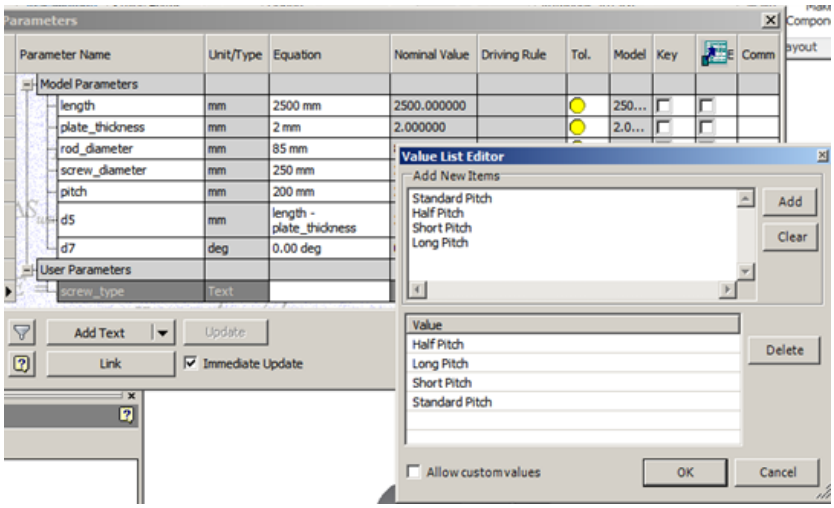
We can also apply rules for parameter using iLogic. Parameter value will be change based on rules that driven by another parameter. For example, in screw conveyor there are some pitch types. Pitch type will determine how much pitch value according to screw diameter.

First, we make a new user parameter. Go to “Manage” tab in ribbon menu > “Parameters”. In parameters windows, click “add text” to add a text based parameter.

Give the parameter name, in this example: “screw_type”. Because there are four type of pitch type, then we must use multi-value. Right click in equation colom > “Make Multi-Value”.



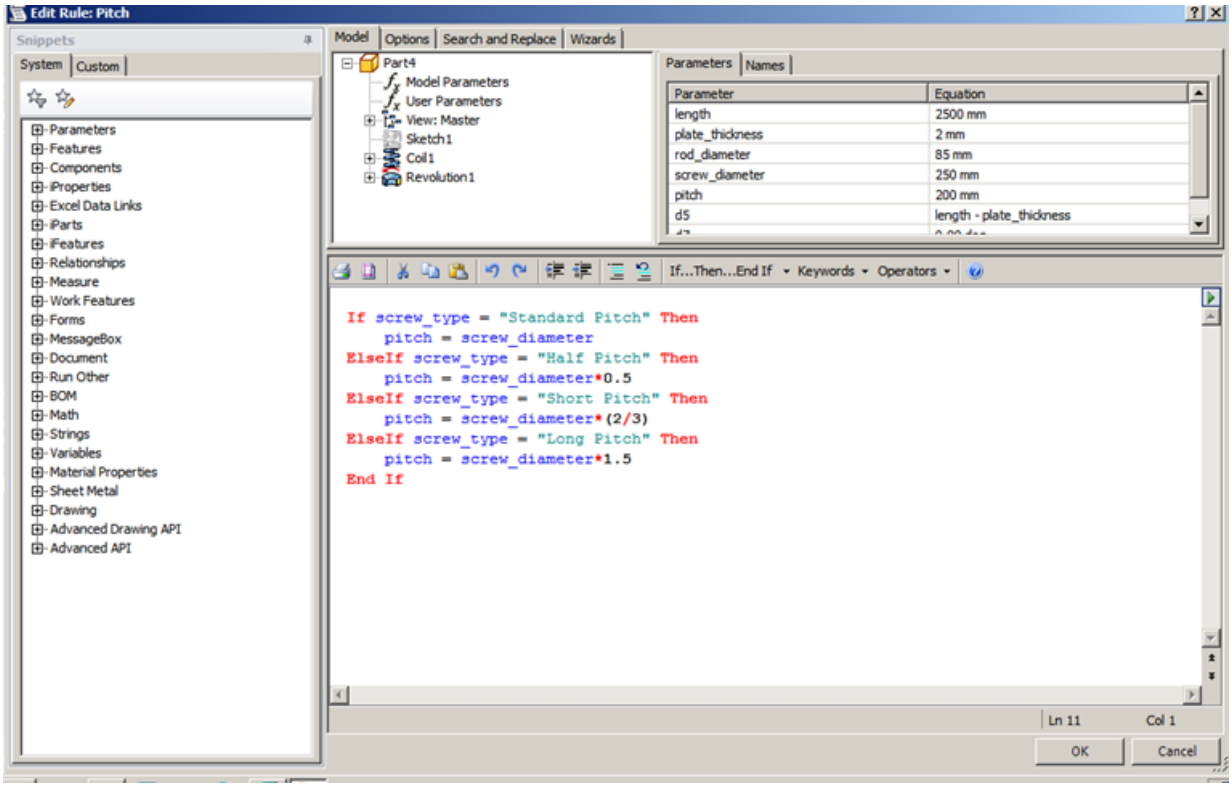
Write all type of pitch type, click add then OK to finish. After that, close the parameter window.



Back to iLogic browser, rules tab. Right click and select “add rules”. Give rule name “Pitch”.

Programming in iLogic is very easy all you need is a simple logic rules. In this case we will use “if the end if” logic statement. In here is the statement:

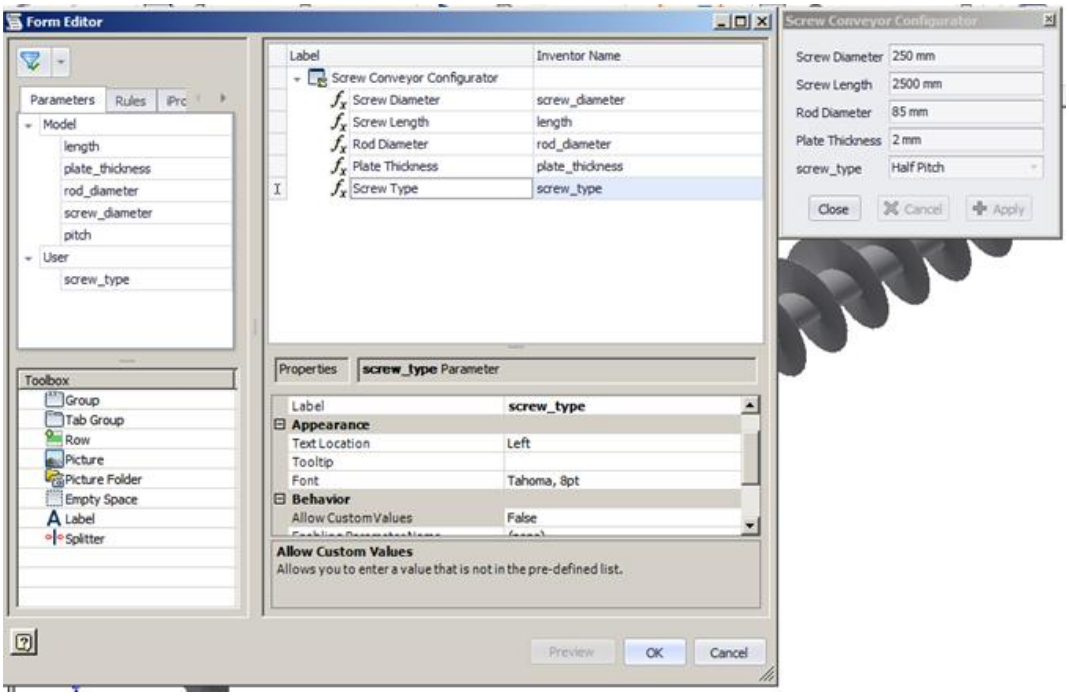
```
If screw_type = "Standard Pitch" Then
    pitch = screw_diameter
ElseIf screw_type = "Half Pitch" Then
    pitch = screw_diameter*0.5
ElseIf screw_type = "Short Pitch" Then
    pitch = screw_diameter*(2/3)
ElseIf screw_type = "Long Pitch" Then
    pitch = screw_diameter*1.5
End If
```



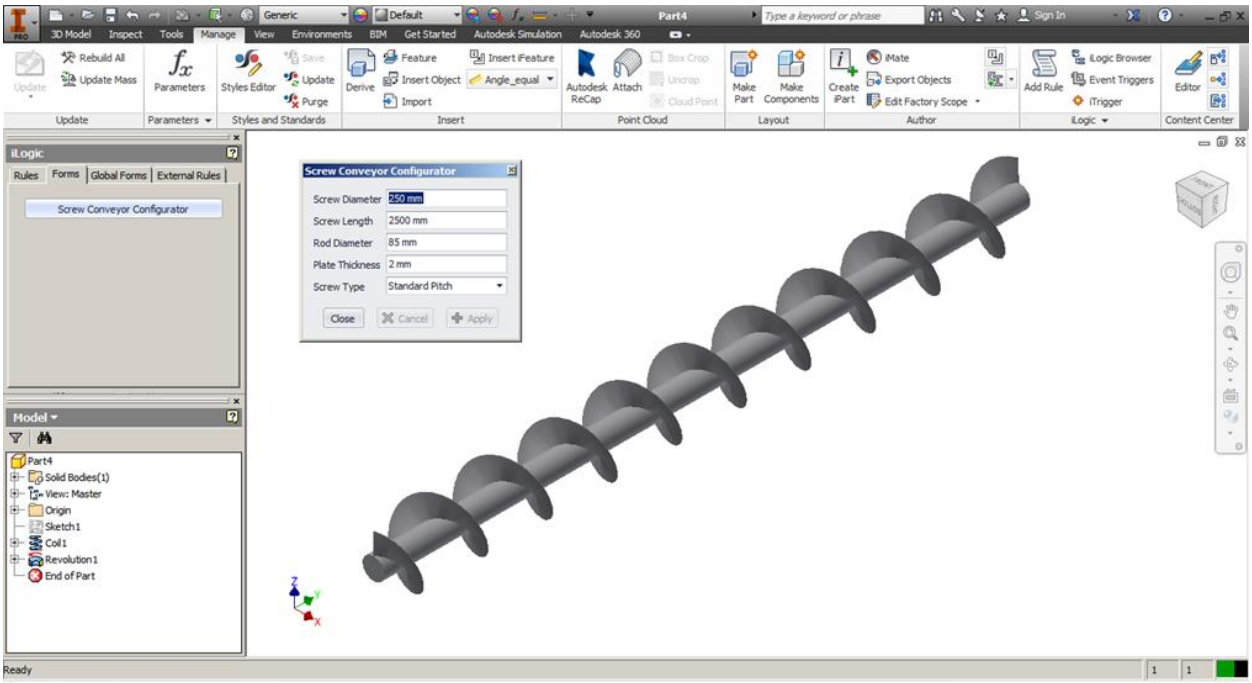
If you can see the programming that iLogic used is very simple. Just use a simple logic statement. Don’t have to add another code to access parameter like we did in VBA.

After that we must edit iLogic form that we have made before. Because it still using a pitch parameter not pitch_type. In iLogic browser, go to form tab, right click in “Screw Conveyor Button” > “edit”.

We delete “pitch” parameter and replace it with “screw_type” from user parameter.



Now pitch value is change based in what kind of pitch type that we select.

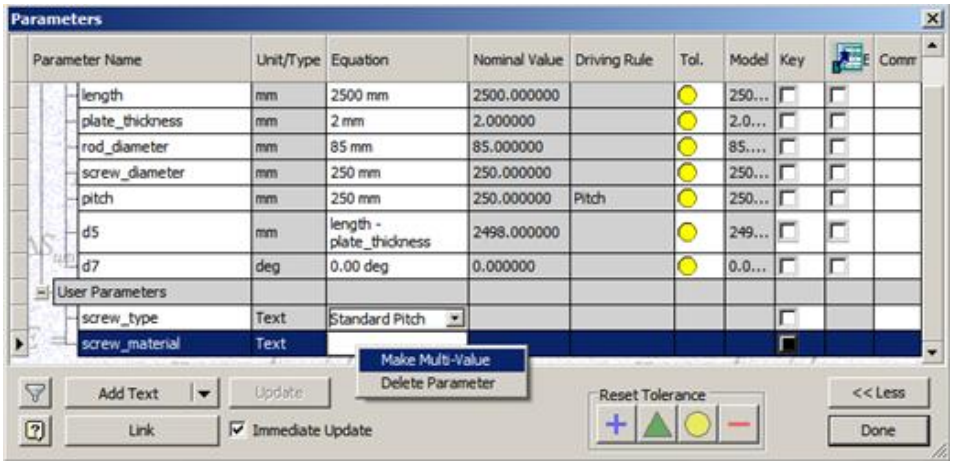


Access iProperties using iLogic

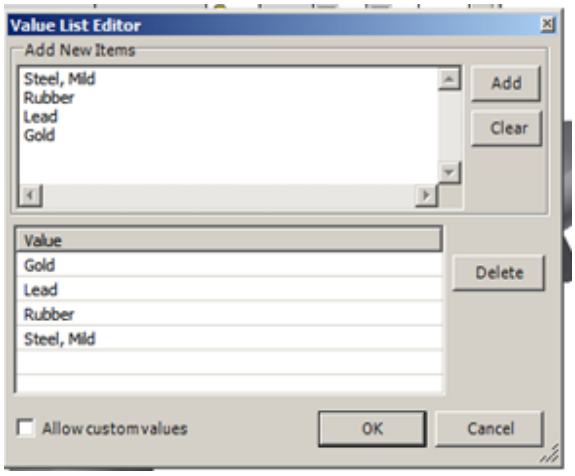
Now, how about material? Can we change material using iLogic?
In conventional step, we can change material trough iProperties. But it take some steps, we want to change material just trough iLogic form.

The steps is almost the same like rule base parameter above. First, we make a new user parameter. Go to “Manage” tab > “Parameters”. In parameters windows, click “add text” to add a text based parameter.

Give the parameter name, in this example: “screw_material”. Because there is more than one type of material, then we must use multi-value. Right click in equation colom > “Make Multi-Value”.



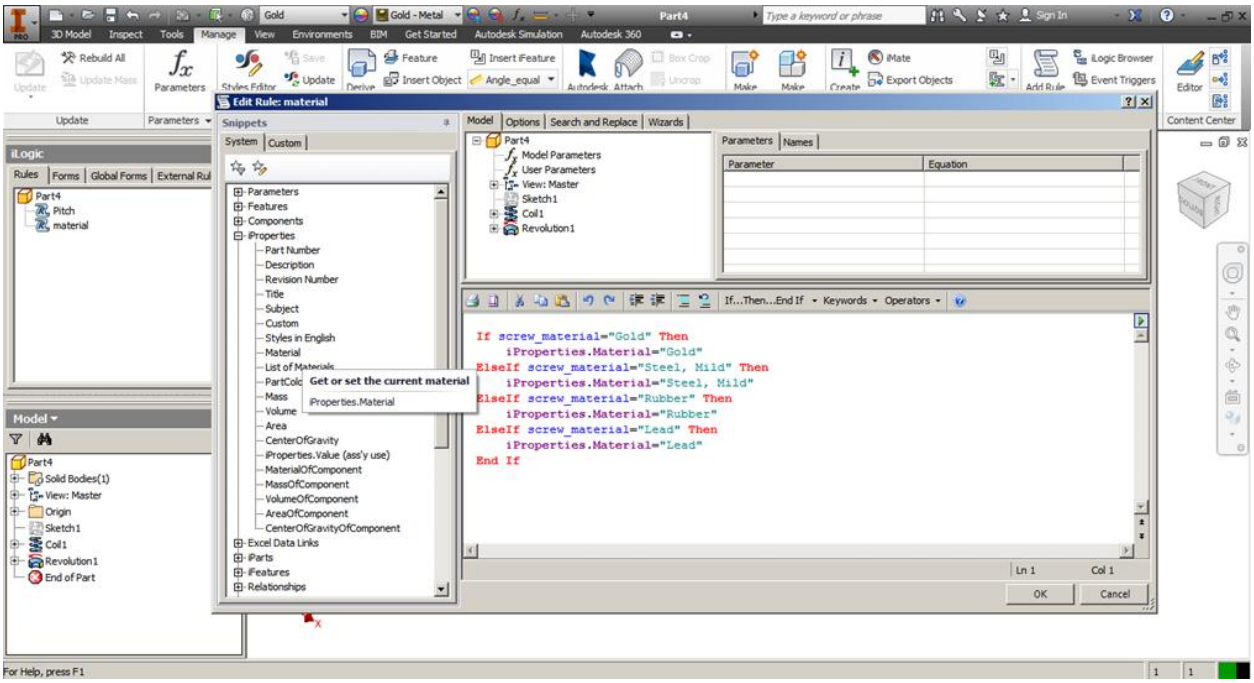
Write all type of material (in this example we only use four materials), click add then OK to finish. After that, close parameter window.



Back to iLogic browser, rules tab. Right click and select “add rules”. Give rule name “material”.

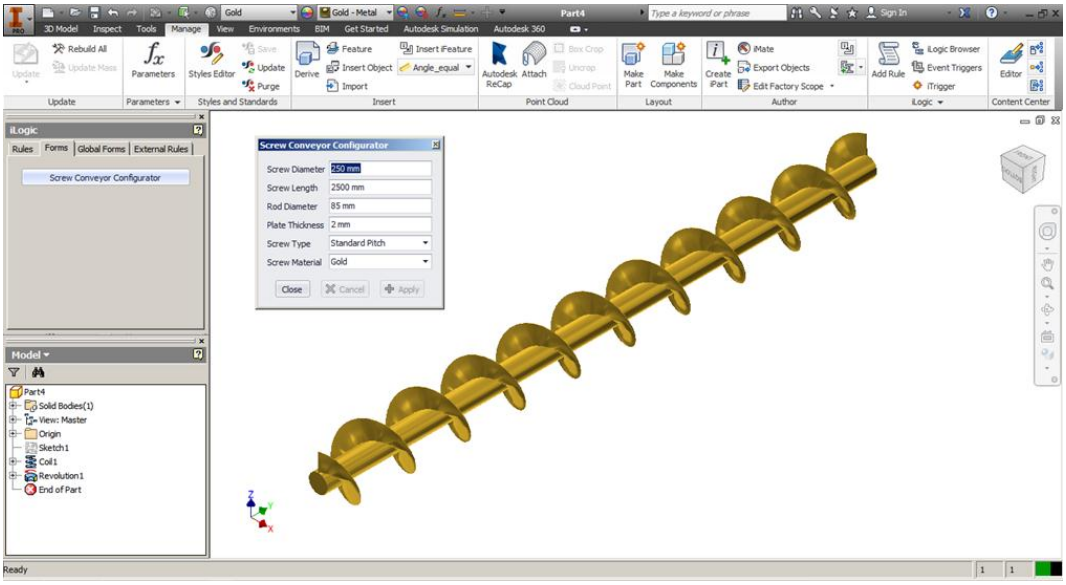
Like I have said before, programming in iLogic is very easy all you need is a simple logic rules. In this case we will use “if the end if” logic statement. In here is the statement:

```
If screw_material="Gold" Then
    iProperties.Material="Gold"
ElseIf screw_material="Steel, Mild" Then
    iProperties.Material="Steel, Mild"
ElseIf screw_material="Rubber" Then
    iProperties.Material="Rubber"
ElseIf screw_material="Lead" Then
    iProperties.Material="Lead"
End If
```



Click OK to finish editing rules. Next we go to “form” tab in iLogic explorer, right click to “Screw Conveyor Configurator” button > “Edit” to edit iLogic form. Now we can add material parameter to iLogic form.

To test, click “Screw Conveyor Configurator” in form tab. Right now we can edit material through iLogic form. Example if we change the material into “Gold”, the color of our part model will also change based on material that we have select in iLogic form.

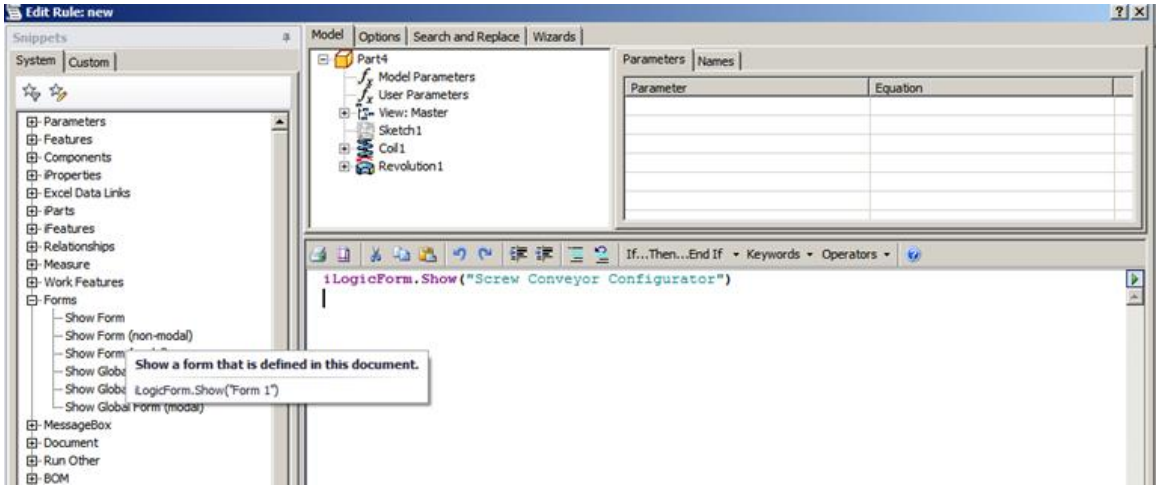


Event Trigger

Event trigger is a trigger to run an iLogic rules based on event. Example, we want to show iLogic form when file is open or when make a new file. Because iLogic form cannot show by itself, then first we must make a new rule to show iLogic form.

In iLogic browser, go to “rules” tab, right click and select “add rule”. Give the rule name: ”new”

In iLogic rule editor,there are some preset code that we can use. For example if we want to show form, go to Snippets colom > “System” > “Form”>”Show form”. By default it will write a code: **iLogicForm.Show(**"Form 1"**)**. Change “Form1” into our iLogic name, “Screw Conveyor Configurator”.



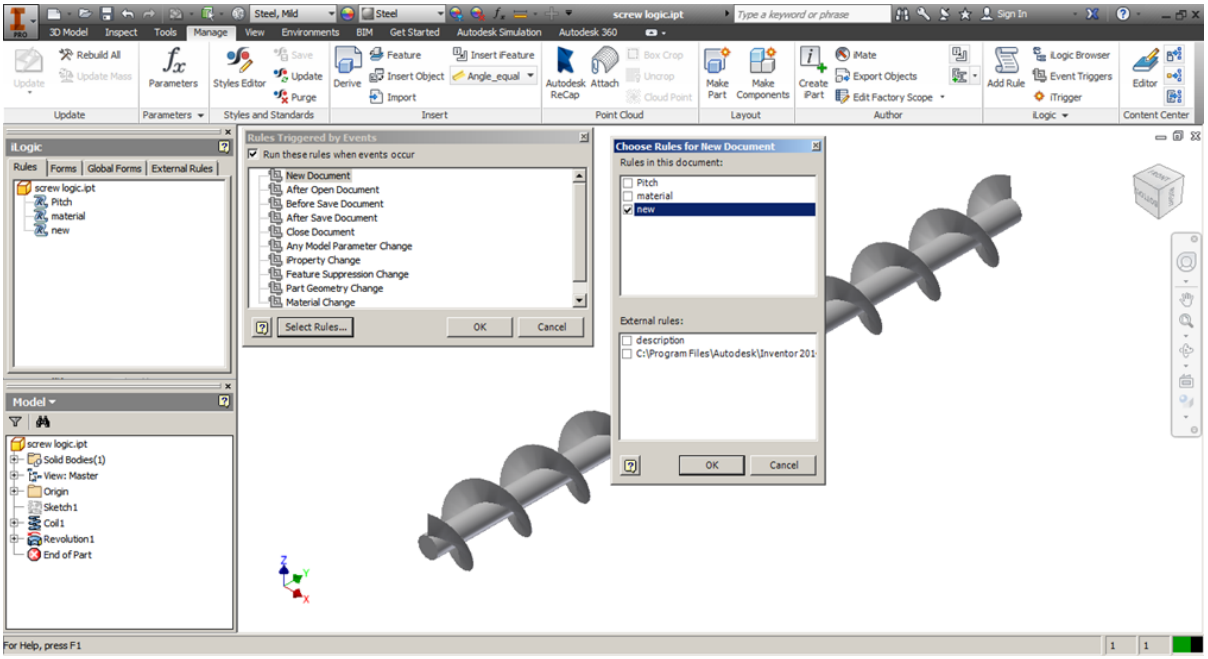
Beside have lots of preset code, we also can use a visual basic code in iLogic. For example if we want to zoom all/zoom extent the model using iLogic. First go to “Snippets” colom > “System” > “Advanced API” > “ThisApplication”. This will write code: **ThisApplication**.

Then add another code behind ThisApplication. Just like in VBA, we must access Active view to zoom all. Here is the full code:

```
iLogicForm.Show("Screw Conveyor Configurator")
ThisApplication.ActiveView.Fit
```

Clik “OK” to finish editing a rule.

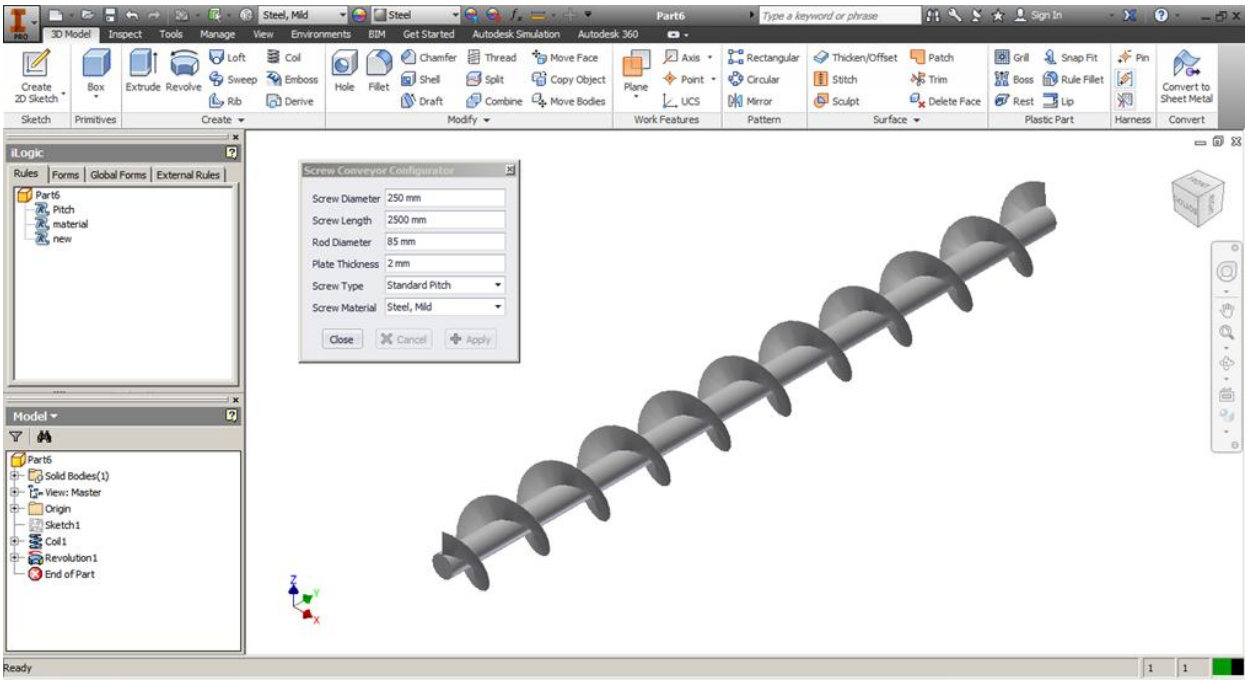
Now, to set an event trigger go to “Manage” tab in ribbon menu > “iLogic” panel > “Event Triggers”. Then select “New Document”, click “select rules” button and check list “new” rules. Do the same with “after open document”. This mean, after we open this file, rules “new” will be run and show our iLogic form.



To test it with new document event, we have to save this file to become a new template. Go to Inventor application menu (upper left corner menu) > “Save As” > “Save Copy As Template”. In this example we save it in university folder with the name: “”Screw Logic.ipt”

After save as template, now we try to open a new document using template that we have been save before. We can see that after new document is open, iLogic form will automatically show. This can save lots of time when developing a new variant product.


Create New File
C:\Users\Public\Documents\Autodesk\Inventor 2014\Templates\
Templates
Autodesk University
English
HBM
Metric
Mold Design
Nama Perusahaan
Part – Create 2D and 3D objects
screw.ipt
screw logic.ipt
Assembly – Assemble 2D and 3D components


Screw Conveyor Configurator
Screw Diameter 250 mm
Screw Length 2500 mm
Rod Diameter 85 mm
Plate Thickness 2 mm
Screw Type Standard Pitch
Screw Material Steel, Mild
Close
Cancel
Apply
iLogic
Model
Part6
Solid Bodies(1)
Origin
Sketch1
Coil1
Revolution1
End of Part
Ready