Autodesk Inventor Tutorials

by Sean Dotson
www.sdotson.com
sean@sdotson.com

VBA Functions in Parts
Part One

Latest Revision: 3/17/03
For R6

This tutorial will deal with a subject that may send some of you running for the hills. It involves using VBA functions in parts and assemblies. But don't pack it in yet. It doesn't take a Visual Basic guru to use these simple functions.

The advantage of using VBA functions in your parts and assemblies is the ability to use VB conditional functions to change parameters. In the parameter dialogue you can use min, min, ceiling, floor and many other functions but you do not have access to If…Then..ElseIf.. or Select Case functions. These functions greatly simply the equations that are needed in the parameters dialogue.

The data set for this tutorial can be downloaded from:
http://www.sdotson.com/tutparts/VBAFunctions.zip

Unzip the dataset and open **Flange.ipt.** This is a simple flange that we will use to illustrate the concept of If…Then functions.

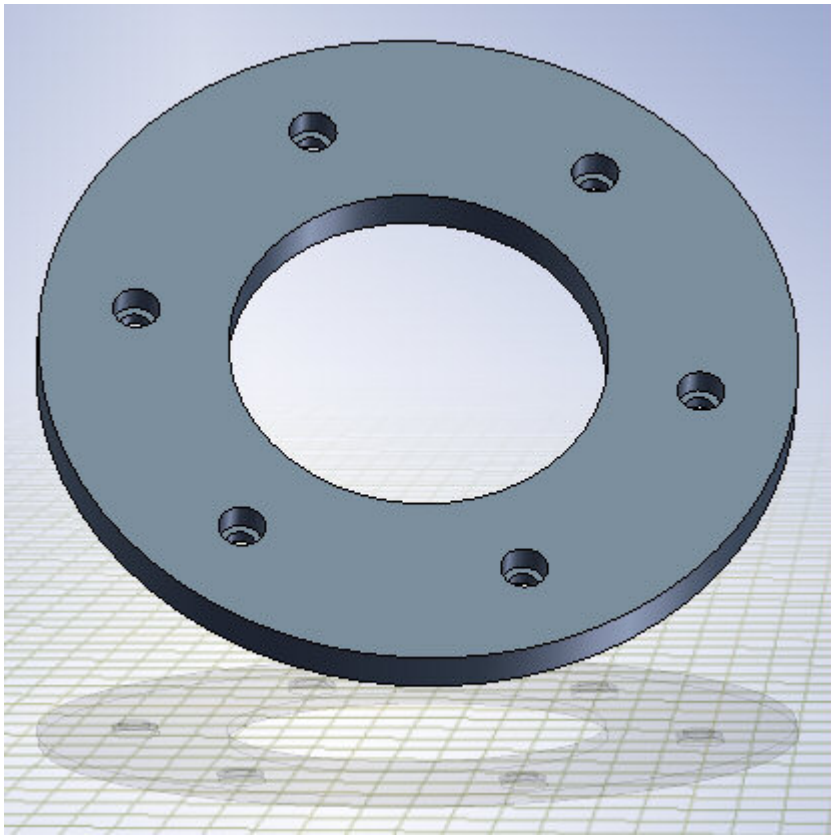

Figure 1 - Basic Flange

Take a moment to look at the sketches that make up the OD, ID and bolt circle of the flange. Note that the ID is defined as the OD/2. Also note that the bolt circle (BC) diameters is defined as the average of the OD and ID (OD+ID)/2. See figure 2.

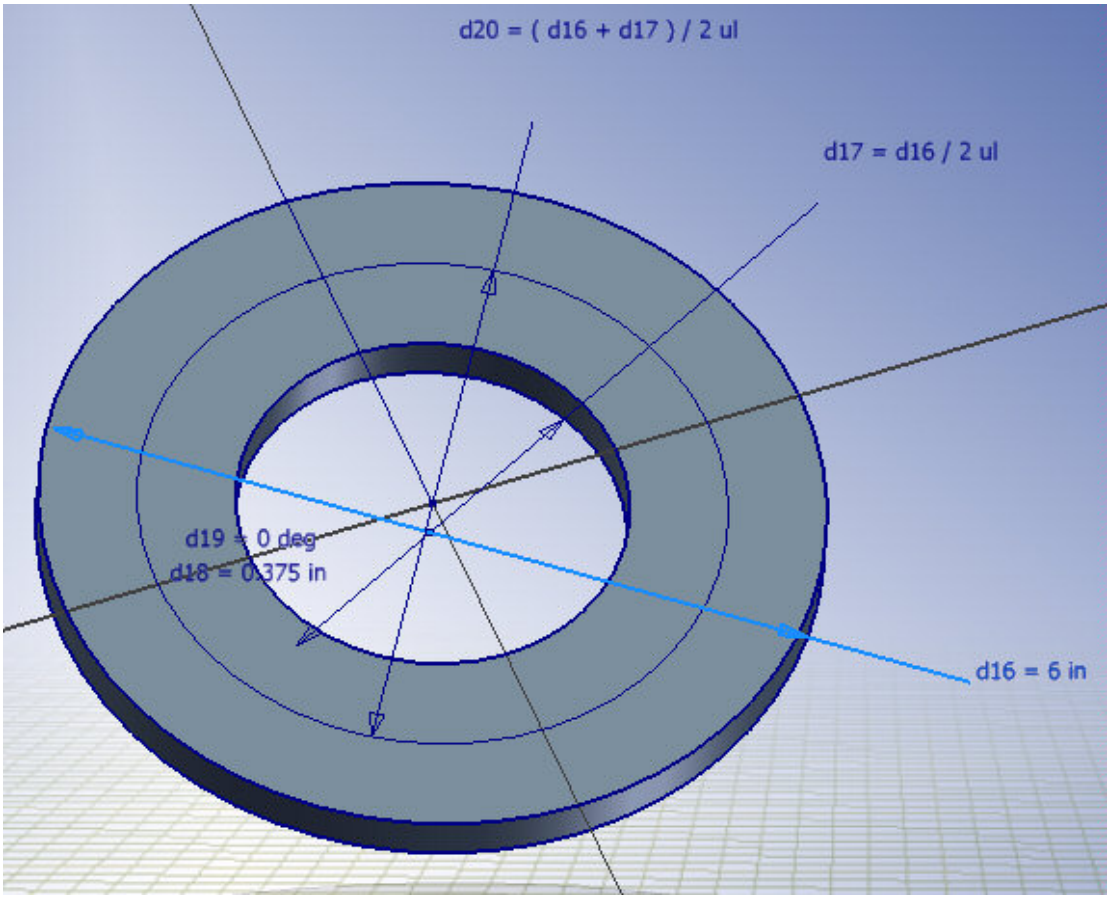

Figure 2 - Sketch Relationships

So we have a flange whose OD, ID and BC are all defined by the OD. By changing the OD in the sketch or via the parameters menu the ID and BC change in hand.

Now it's time to jump into the programming. On the main menu choose **Tools>Macros>Visual Basic Editor**. You will be presented with the VBA programming environment. (See Figure 3)

In the upper left you will see a list of the open projects and open documents. If you have multiple documents open you will need to determine which one is the flange.ipt document. The filename is listed beside each document project name. Find the document project for the current file **flange.ipt** and expand the browser so that the functions tab is visible. Click on the functions tab. The right hand side of the screen is where you type your code. At this point I will not going to go into great detail about the VB Editing environment (there are other great tutorials for this subject).

Click in the right hand pane and type the following:

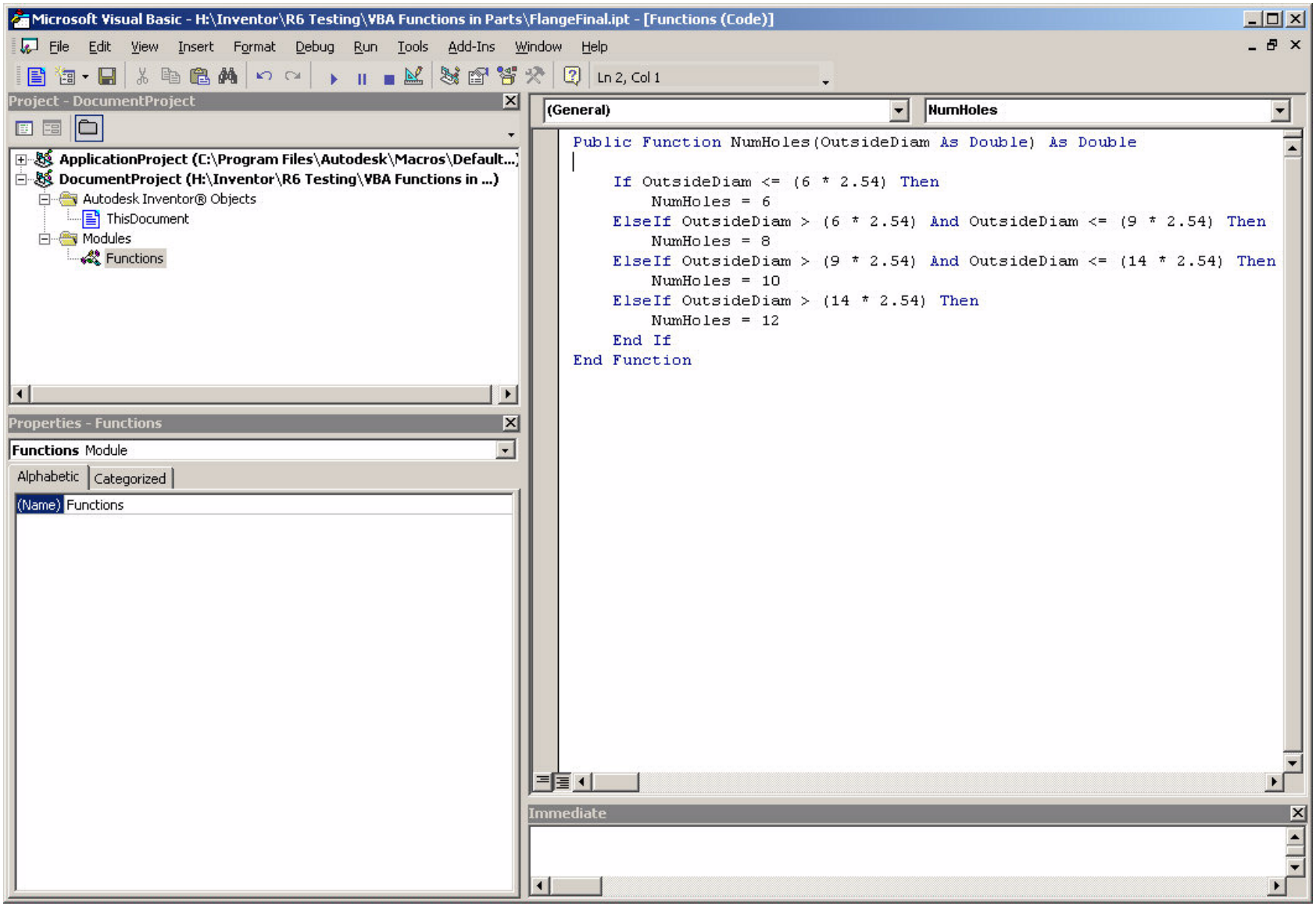

Figure 3 - VBA Editor Interface

```
Public Function NumHoles(OutsideDiam As Double) As Double

    If OutsideDiam <= (6 * 2.54) Then
        NumHoles = 6
    ElseIf OutsideDiam > (6 * 2.54) And OutsideDiam <= (9 * 2.54) Then
        NumHoles = 8
    ElseIf OutsideDiam > (9 * 2.54) And OutsideDiam <= (14 * 2.54) Then
        NumHoles = 10
    ElseIf OutsideDiam > (14 * 2.54) Then
        NumHoles = 12
    End If

End Function
```

Let's take a look at what this code means:

We first declare the function and give it a name (**NumHoles**). We then define the arguments of the function (**OutsideDiam**). The arguments are the information we are using to make our decision in the function. We define **OutsideDiam** as a double precision variable. We define the output of the function **NumHoles** as a double precision variable as well. Due to the way these functions work define all variables as double precision variables even if the variable is an integer. If you do not do so the function will not execute properly.

In the next step of the code we begin our first **If** statement. Since **OutsideDiam** is our only argument we want to check to see what the value of the variable is. In the line:

```
    If OutsideDiam <= (6 * 2.54) Then
```

we check to see if **OutsideDiam** is less than or equal to 6". Now you will notice that we check this value against (6*2.54). **This is due to the fact that Inventor uses internal units of centimeters and radians.** Anytime you pass a value to an Inventor VBA function it translates the model units into centimeters. So when we want to check if the value is less than or equal to 6" we need to check to see if it's less than or equal to 6*2.54 = 15.24 cm. This is probably the most difficult part of the process. If your model units are in units other than inches you will need to use the appropriate conversion factor to translate them into centimeters.

You also need to keep this in mind when getting values out of the function. A function that returns a value of 5 is really retuning 5cm. In your parameters dialogue you will need to convert this into the appropriate units.

So if the argument **OutsideDiam** is less than or equal to 6" we specify the function **NumHoles** to return a value of 6.

The next line is an **ElseIf**statement. This means if the first If statement was not true then check this statement. In this first **ElseIf** we check to see if **OutsideDiam** is greater than 6 or less than or equal to 9. Notice we did not use greater than or equal to. If you did then

both the first and second IF statements would be true if OutsideDiam was = 6. You must be careful in this regard.

If this statement is true we return 8 as the value of the function. We continue this logic for two more iterations, each time returning a different value for NumHoles.

We then end the If statement with an EndIf and then end the function.

Save the document project (which also saves the part file) and close the VB editor. We now need to set up the link to this function in the parameters dialogue.

Open the parameters dialogue and scroll down to parameter d28. This is the parameter that controls the number of occurrences in the circular pattern. In the equation cell for the parameter type:

VBA:NumHoles(d16) * 1.000 ul

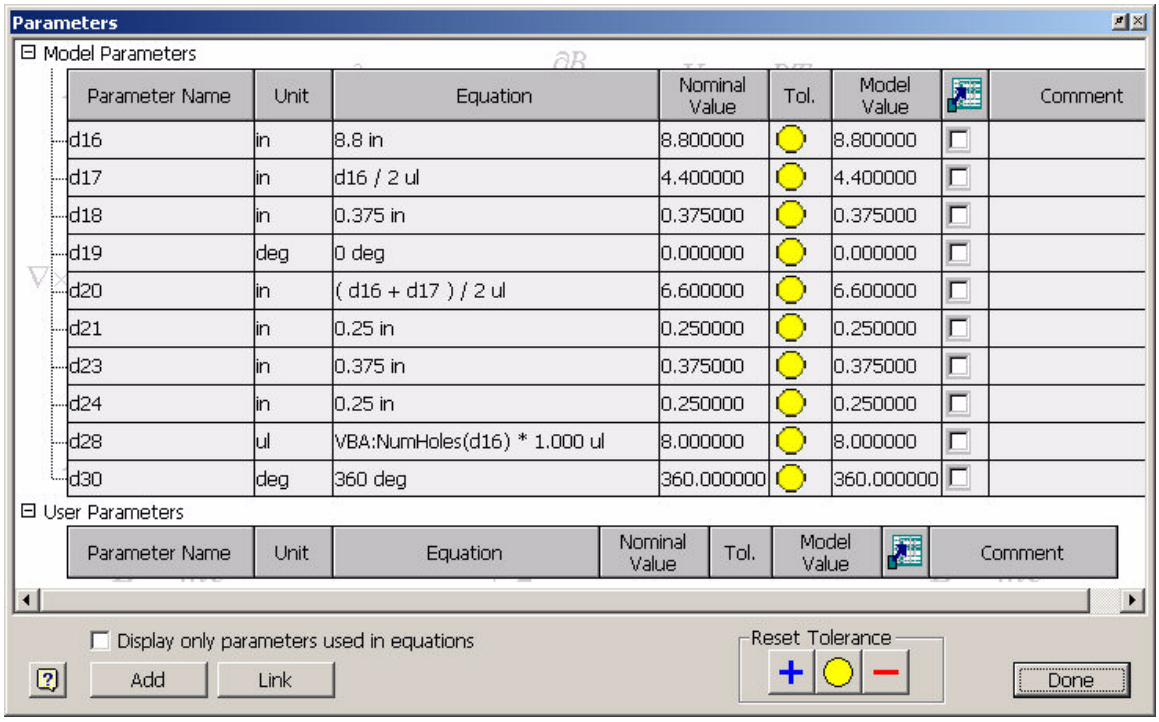

Figure 4 - Parameter Dialogue

The **VBA:** portion tells Inventor this is a function. **NumHoles** is the name of the function. **d16** is the parameter we are passing to the function to be used as the argument **OutsideDiam**. We multiply this by 1ul to turn the value unitless.

If we were passing more than one argument we would separate them with a semicolon such as:

VBA:NumHoles(d16;d20) * 1.000 ul

Keep in mind that your function would also need to have two arguments.

Exit the parameters dialogue and update the model. We should still only have 6 holes. Now change the OD of the plate to 9.5" then update the model. The number of holes should have changed to 10. You can continue to vary the OD to explore all the possible combinations.

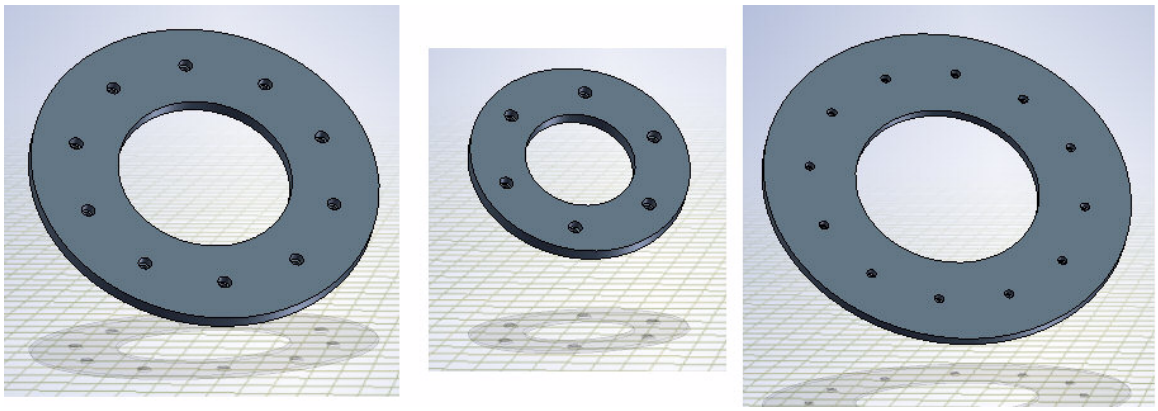

Figure 5 - Various Flange Permutations

The VBA function is contained within the document so this part can be used in a number of different projects without having to worry about linked spreadsheets.

You can have multiple functions in a part document to change different aspects of the part. For example instead of computing it in the parameters menu, we could have computed the value of the BC based on the OD and ID by a function like the following:

```
Public Function BCDiam(OD As Double, ID As Double) As Double

    BCDiam = ((OD + ID) / 2) / 2.54

End Function
```

and in the parameter's dialogue we would define d20 as

VBA:BCDiam(d16;d17) * 1.000 in

One of the only drawbacks of this method is that you cannot access any other Inventor API calls in your function. That is to say you cannot alter the visibility or color of parts, alter other parameters, suppress or compute features etc.

You can however access external applications such as Excel and Access. This would allow you to have a very lengthy lookup table and select values for your model from this table based on user input.

If there is enough interest in this tutorial, I will expand this method and explain how it can be used in assemblies for motion simulation.

(See, now programming wasn't that bad was it?) ☺