TEKLA

[Document type]

[Confidentiality]

Excel Incorporation Into Custom Components

Advanced Custom Component Features

Manual

Description

Author	Joni Virkki
Date	
Reviewed by	
Review date	
Approved by	
Approval date	

Change history

Date	Changed by	Issue

Tekla Corporation

www.tekla.com

Headquarters
Metsänpojankuja 1
P.O.Box 1
FI-02131 Espoo
FINLAND

Tel. +358 30 661 10
Fax +358 30 661 1500

Trade Reg. 183.128
Business Identity Code 0196634-1
Domicile: Espoo

Table of Contents

Pictures

No table of figures entries found.

Tables

No table of figures entries found.

EXCEL SHEET DESIGN INCORPORATION WITH TEKLA STRUCTURES

1. Background

This document instructs how to link your own Excel sheet calculations to Tekla Structures. It is expected that you have reasonable experience with Microsoft Excel 2007, creating custom components in Tekla Structures as well as an appropriate Excel calculation sheet ready. The Excel sheet used in this demonstration has been created from scratch and is extensively modified, but the methods demonstrated herein are universally applicable for similar situations.

2. The Custom Component

2.1 Creating a Custom Component

In this example, we intend to calculate in-situ values for a simple beam-to-beam reinforcement. Create a custom component in Tekla Structures that relates to a similar situation as your own Excel calculation sheet. Name the component according to the purpose, in this case '**BeamToBeam**'.

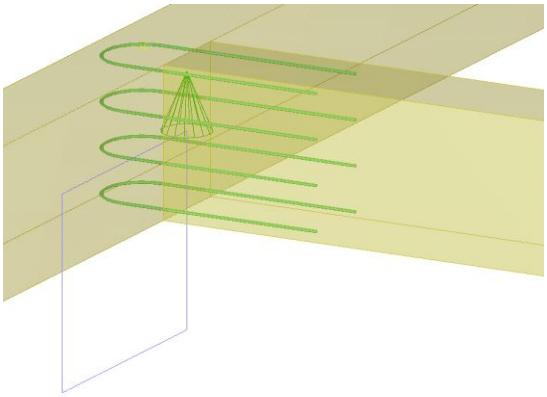


Figure 2.1 'BeamToBeam' custom component

2.2 Preparing the Custom Component

1. Right-click on your newly created custom component and click **Edit custom component.**

2. Edit the custom component as you normally would: add **magnetic user planes** along the reinforcement handle points and bind the planes to the beam **boundary planes**. This allows the reinforcement to react and resize itself according to changes in the beams' profile sizes.

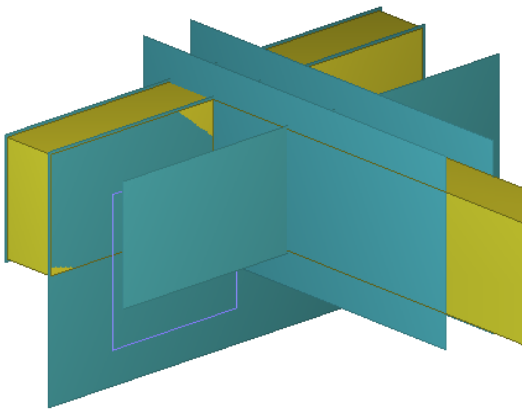


Figure 2.2 Magnetic plane binding

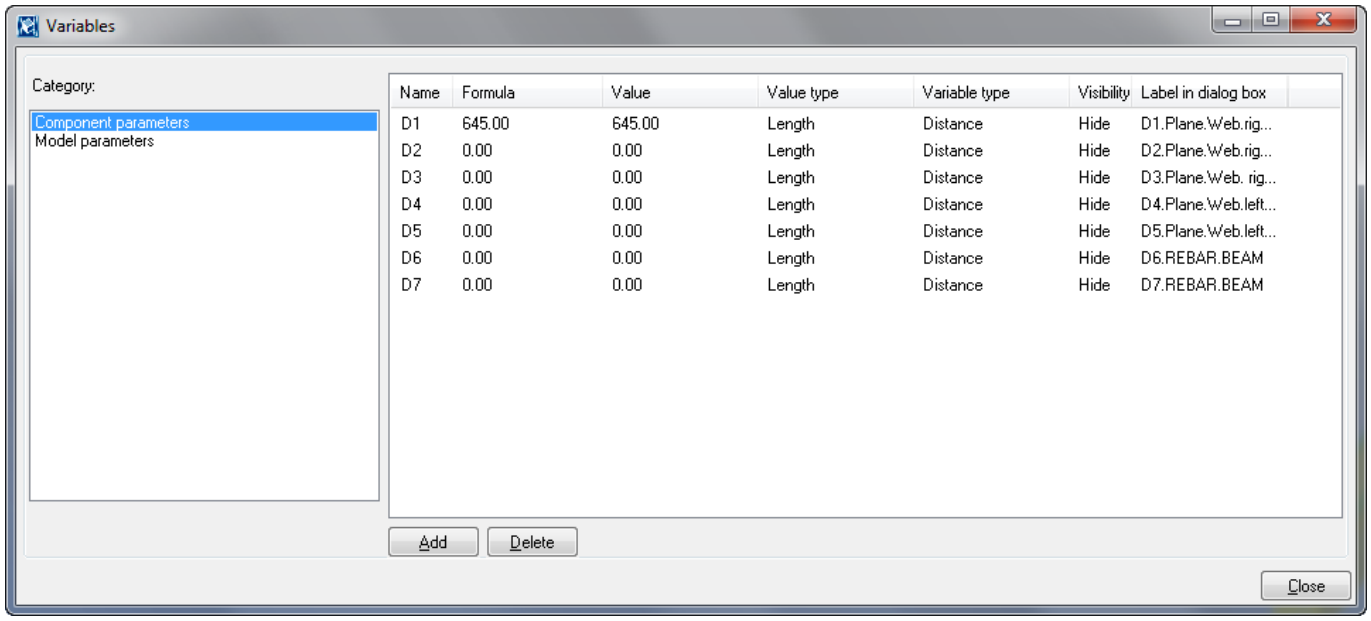


Figure 2.3 Distance Variables

3. Add a new parameter and name it "use_externaldesign". Change the **Value type** to **Yes/No** and set the **Formula** to **1** (1=Yes 0=No).

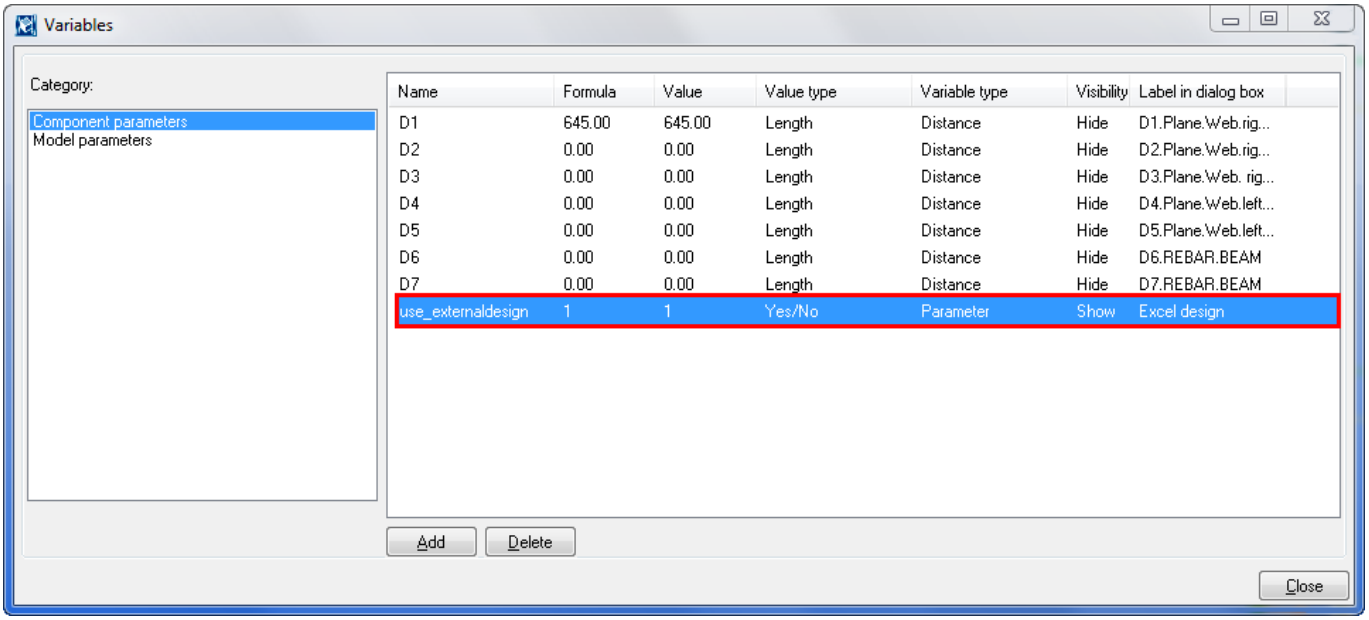


Figure 2.4 Setting use of external design

4. Add any other parameters needed for this component and the calculations. Name the parameters according to their purpose and then link the parameters to necessary places in the **Custom component browser**.

   In this example we have created the following parameters:

   - Reinforcing bar diameter
   - Number of reinforcing bars
   - Cover thickness on plane
   - Cover thickness from plane
   - Shear force

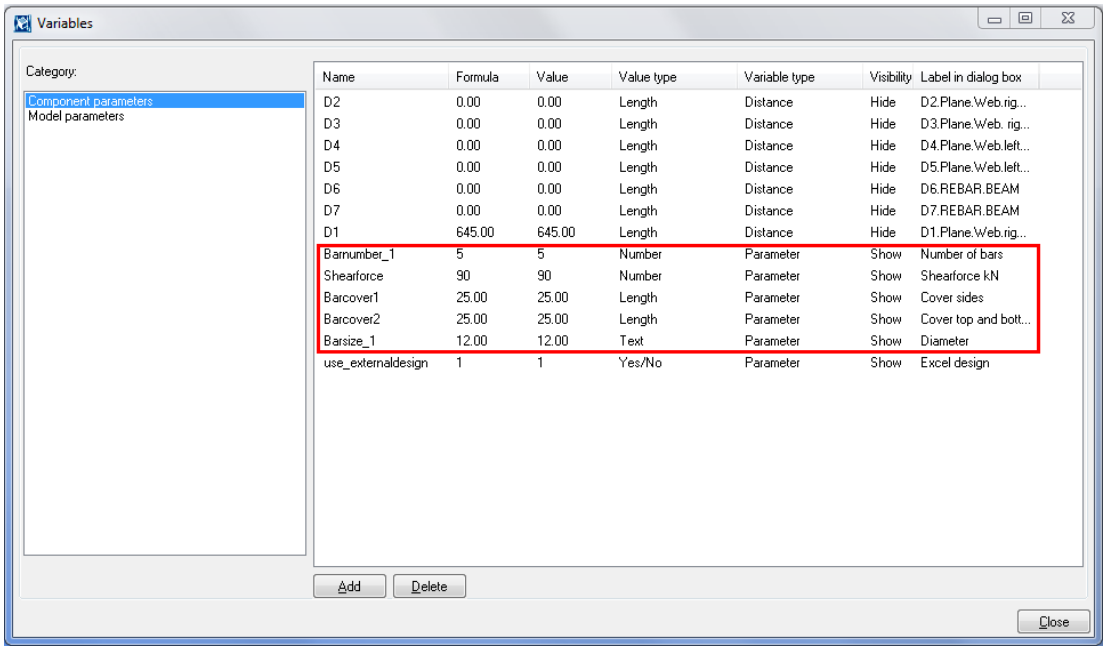


Figure 2.5 Custom parameters

In this case, the custom parameters must be bound to properties found in the **Custom component browser** under **Component > Component objects > Rebar group > General properties** and **…> Group properties** according to their names specified in the **Variables** window.

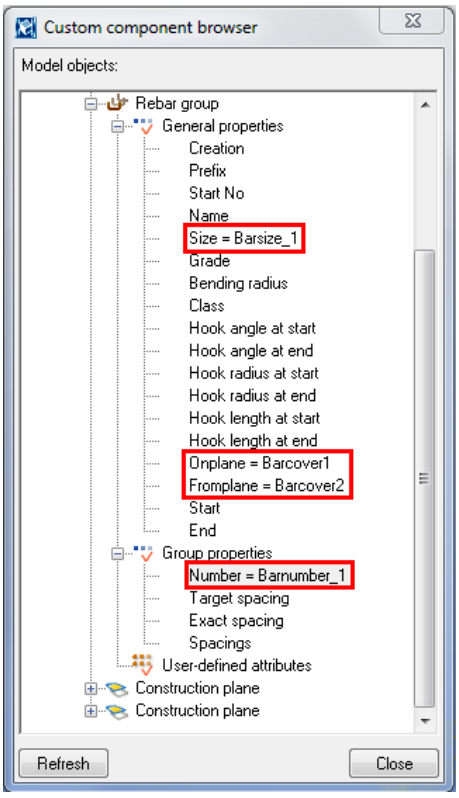


Figure 2.6 Custom parameter binding

Note that, as shear force is only used for calculations, it is not bound to anything in the **Custom component browser**.

Save the component and test it to make sure it works as expected.

3. The Excel sheet

3.1 Activating Debugging

The computer-automated workflow done performing cross-application tasks such as linking Tekla Structures to Excel may be less-than-obvious for even the more experienced user. It is therefore useful to activate *debugging* in Excel in order to more precisely follow the workflow and more easily detect possible bugs, defects or shortcomings.

In order to activate debugging and be able to test and see how the Excel sheet functions, we need to make a few simple changes to the **`Excel.vb`** file

In Tekla Structures versions up to ver.16.0, the **`Excel.vb`** file is located in
`… \ TeklaStructures \ < version number > \ nt \ bin \ plugins`

- Open the **`Excel.vb`** file and make the following changes:

  Set `Const DEBUG As Boolean =` **`False`** to equal **`True`**
  Set `Const SHOW_EXCEL As Boolean =` **`False`** to equal **`True`**

In Tekla Structures 16.1 and onward, the **`Excel.vb`** file has been moved to a new location in order to streamline file structure. It can be accessed with the following steps:

1. The `Excel.vb` file can now be found in **`... \ Tekla Structures\ < version number >\ Environments\ Common\ Exceldesign`**

2. Open the **`Excel.vb`** file and make sure the following settings are in effect (if not, change them):

   ```
   Const DEBUG As Boolean = True
   Const SHOW_EXCEL As Boolean = True
   ```

For more information on file locations and folder structure in Tekla Structures, visit https://extranet.tekla.com/BC/tekla-structures-en/product/version_downloads/16.0/Downloads/Renewed_Tekla_Structures_installation.pdf

Debugging is now activated.

3.2 Creating the Excel calculation sheet

We will now combine the custom component – specific Excel sheet with the standard component calculation Excel template built into Tekla Structures.

The component calculation template in question is **component_template.xls**.

- In Tekla Structures 16.1 and onwards, the template is located in the same directory as the **Excel.vb** file.
- In Tekla Structures 11.0-16.0, the **component_template.xls** file is located in …**\ Tekla Structures\ < version number > \ Environments\ common\ Exceldesign**

1. Copy the **component_template.xls** file to the same file location as your custom Excel sheet and rename the copy to '**component_[**your component's name**].xls**' (in this case component_BeamToBeam.xls)

You can place the Excel sheets inside the Tekla Structures model folder as well. Create a folder within named **Exceldesign** and place the files in there.

Open your calculation sheet and component_*.xls sheet in Excel at the same time. Copy the calculation sheet to the component Excel template by right-clicking on top of the *sheet name* and click **Move or Copy**.

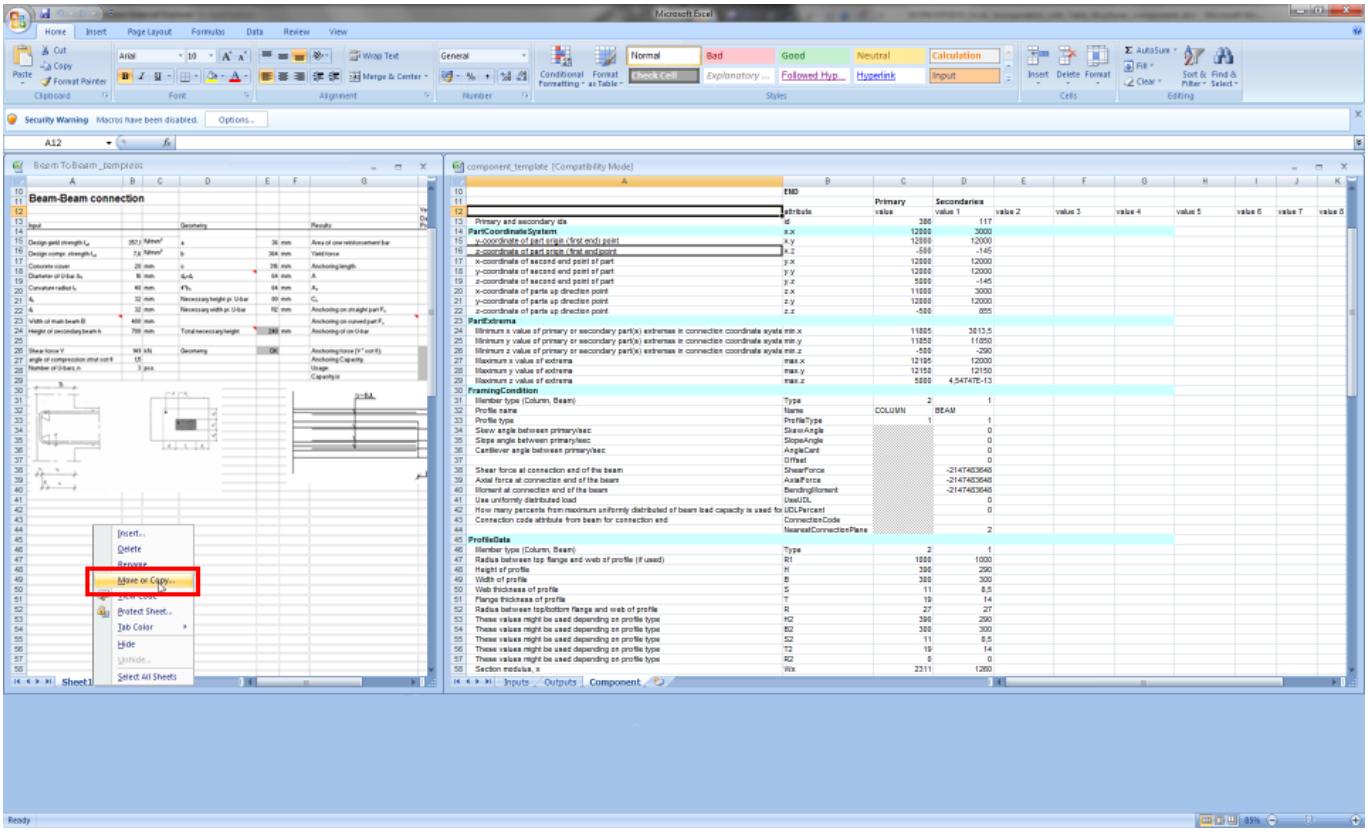


Figure 3.1 Move or Copy

1. Select the **component_*.xls** file(component_BeamToBeam.xlsx) from the "**To book:**" drop-down box and check **Create a copy**.

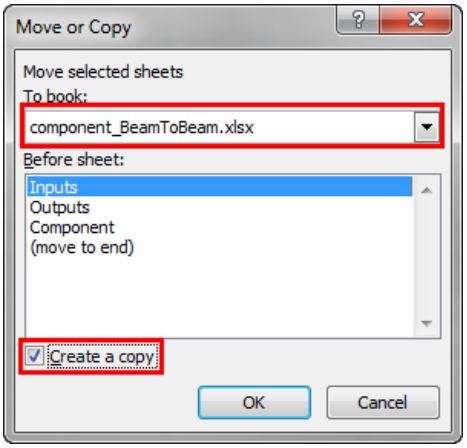


Figure 3.2 Create a copy

2. Click **OK**. The calculation sheet is copied to the component Excel.
3. Close the original calculation sheet.
4. Rename the copied sheet (**Sheet 1**) to **Calculation** by right-clicking and clicking **Rename**.

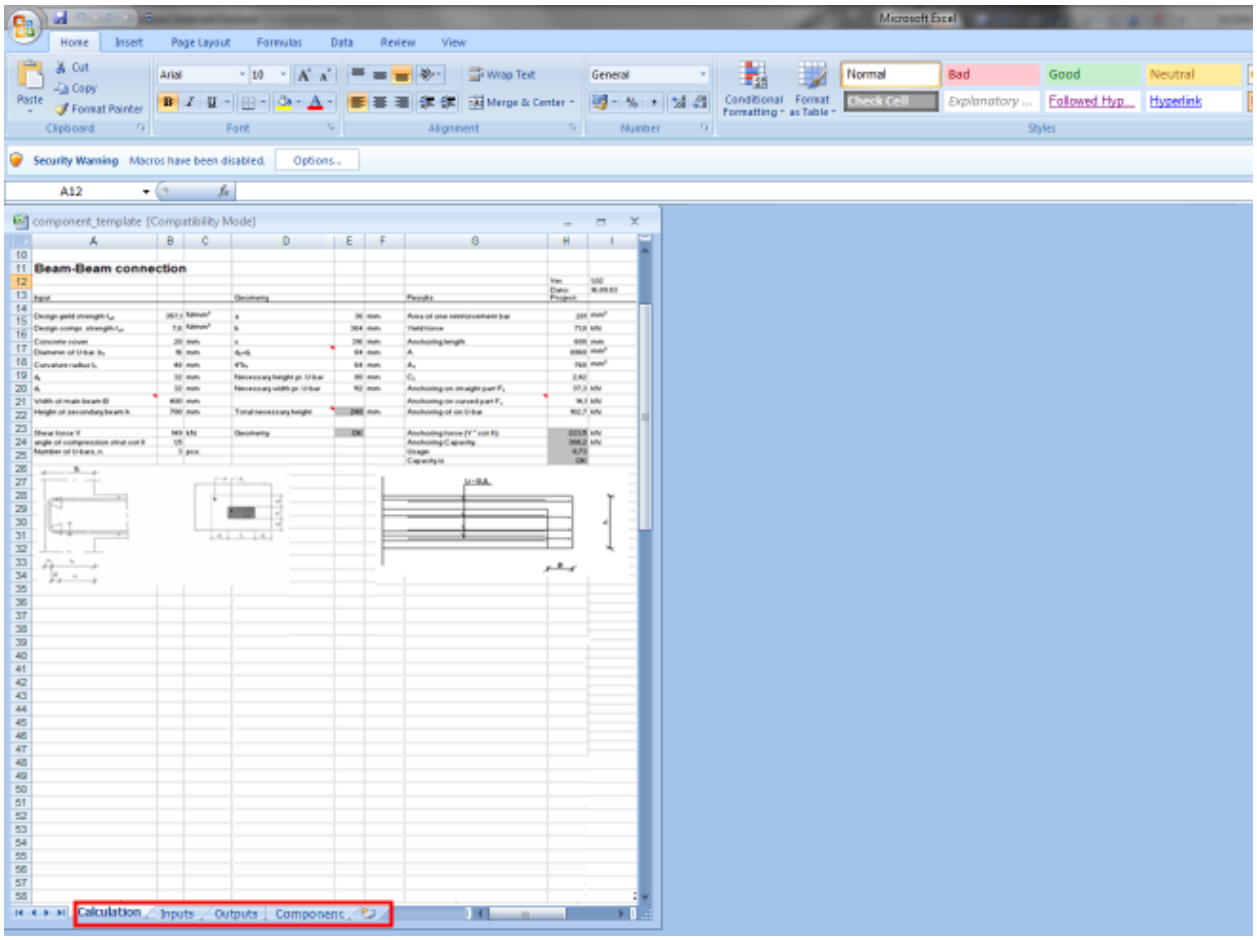


Figure 3.3 Incorporated and renamed sheet

3.3 Editing the component calculation sheet

Quite often you might need or want the Excel calculation sheet to work differently with Tekla Structures than as a stand-alone Excel sheet.

In this case, the original calculations are defined by given values according to whether the reinforcement ratio is under or above 1. Since the calculations will soon be connected to Tekla Structures, it would be wise to edit the calculation sheet so that it calculates the required amount of reinforcements and transfers the information back to the custom component for execution.

The Excel sheet has been changed so that, instead of calculating the ratio of a given number of reinforcing bars, the sheet calculates the ratio of a single rebar. If the anchoring capacity isn't enough, the Excel sheet adds another rebar and calculates the ratio again, continuing until there are enough reinforcing bars to properly withstand the shear force.

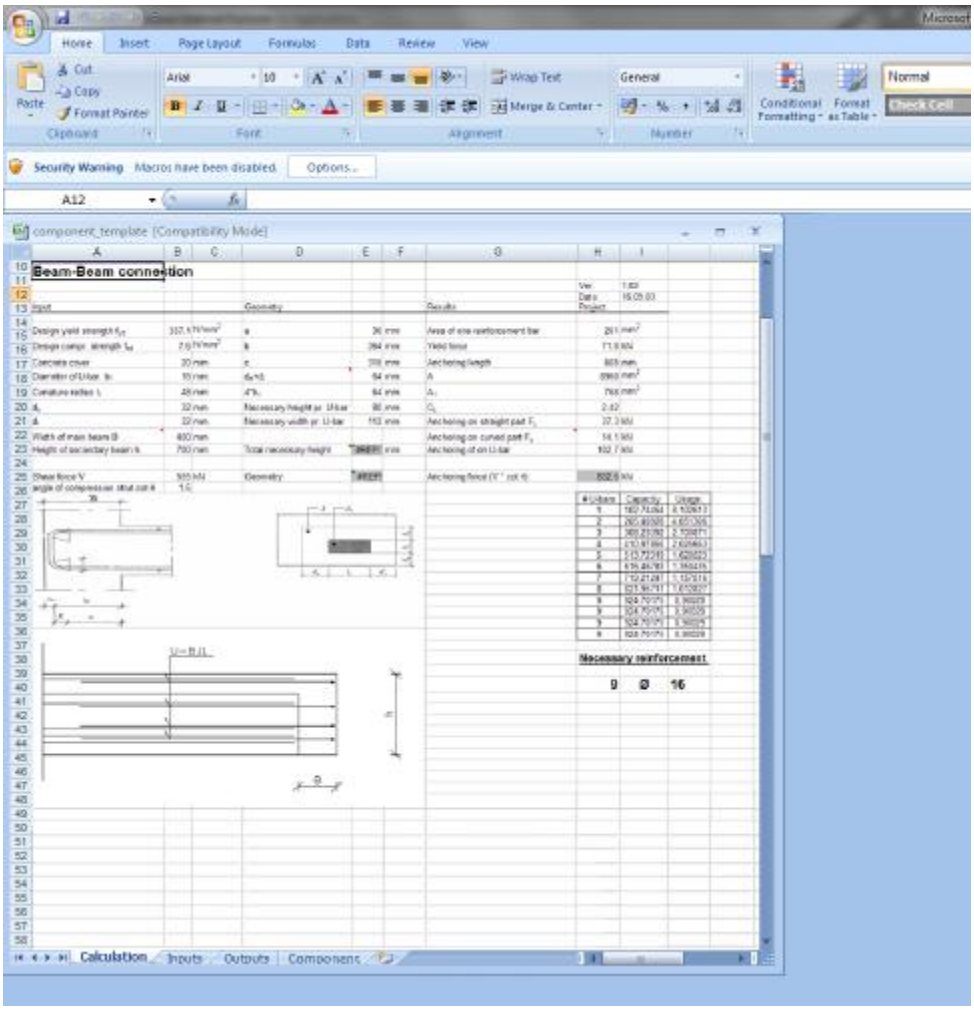


Figure 3.4 Calculation sheet

3.4 Adding input and output fields

The **Inputs** sheet in the standard component Excel template is used to input information *from* the Tekla Structures component into the Excel sheet. Similarly, the **Outputs** sheet returns calculated information back into Tekla Structures to be used in the component.

1. Go to the **Inputs** sheet

2. Add the necessary parameters required for the calculations as shown in figure 3.5 from the **Variables** window (adding both **parameter name** and **type**).

The available types are **string**, **double** and **int**.

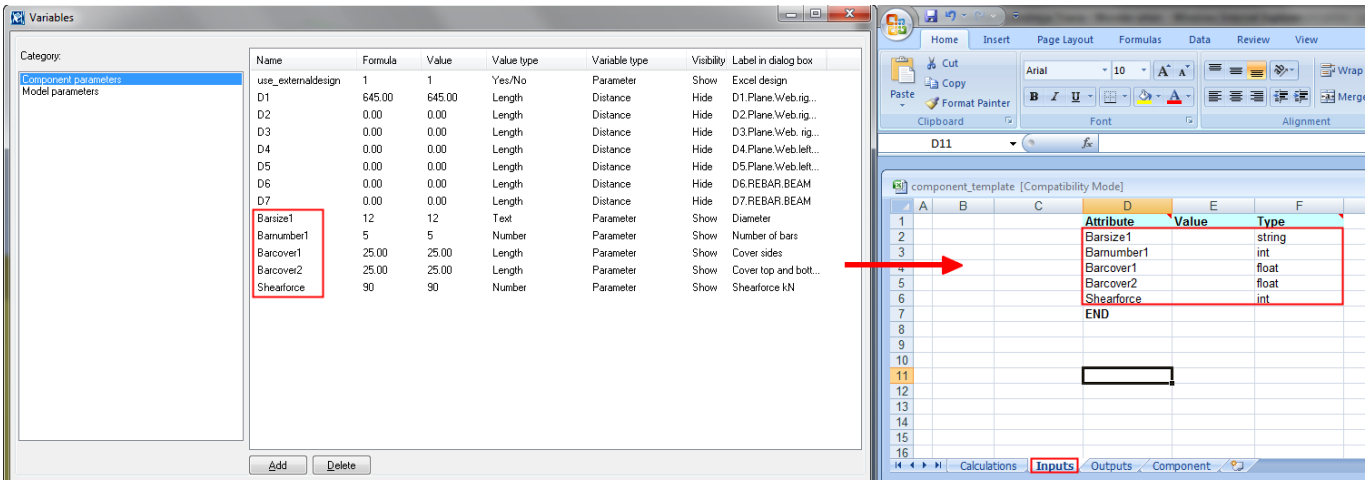


Figure 3.5 Adding Input parameters

3. Similarly, add the variables that you want to output back to the custom component to the **Outputs** sheet.

   In this example we only calculate the number of required reinforcing bars, so the only parameter added to the **Outputs** sheet is the amount of re-bars (parameter = Barnumber1)

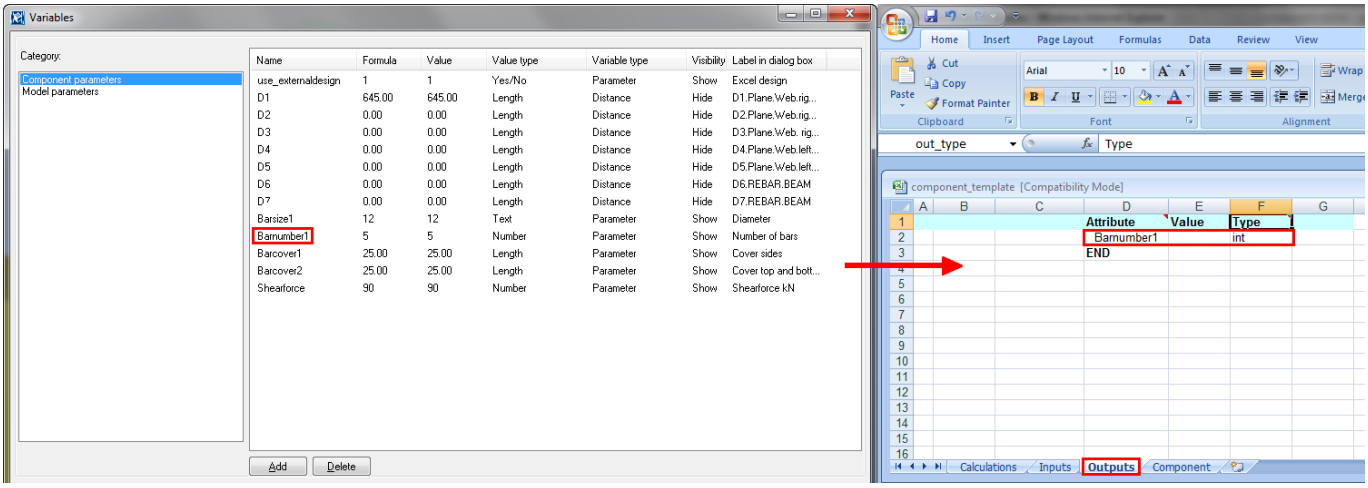


Figure 3.6 Adding Output parameters

3.5 Linking input, output and component fields to calculations

Whenever the component in Tekla Structures is modified, the calculation needs to be redone. In order for this to happen automatically, we need to link the input and necessary component fields to the calculation sheet and from then on from the calculation sheet to the output sheet, which then returns the new values to the component and updates it accordingly.

1. Link the **shear force** value from the **Inputs** sheet to the **Calculations** sheet. Click the shear force value field in the **Calculations** sheet and add an equality sign (**=**).

2. Go to the **Inputs** sheet, select the *value field* for the *shear force* and press **Enter**.

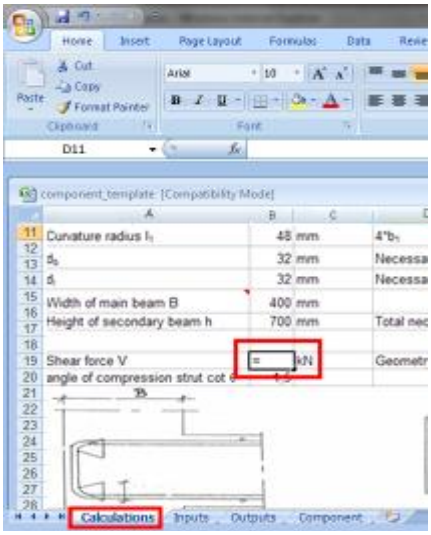


Figure 3.7 Adding = to value field in calculation page

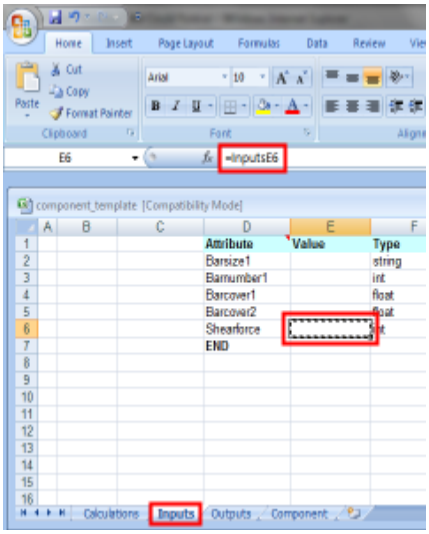


Figure 3.8 Selecting shear force on Inputs sheet

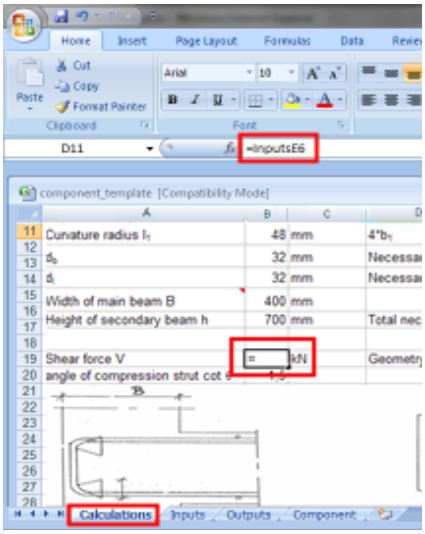


Figure 3.9 Linking value to Calculations sheet

3. Link the *main beam width value* from the **Component** sheet to the **Calculations** sheet. As above, create an equality sign (**=**) into the **Width of main beam B** value field, go to the **Component** sheet, select the *main part width* value field and press **Enter**.

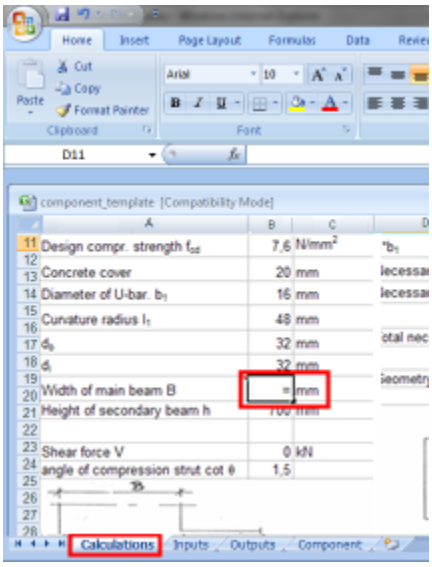


Figure 3.10 Adding an equality sign

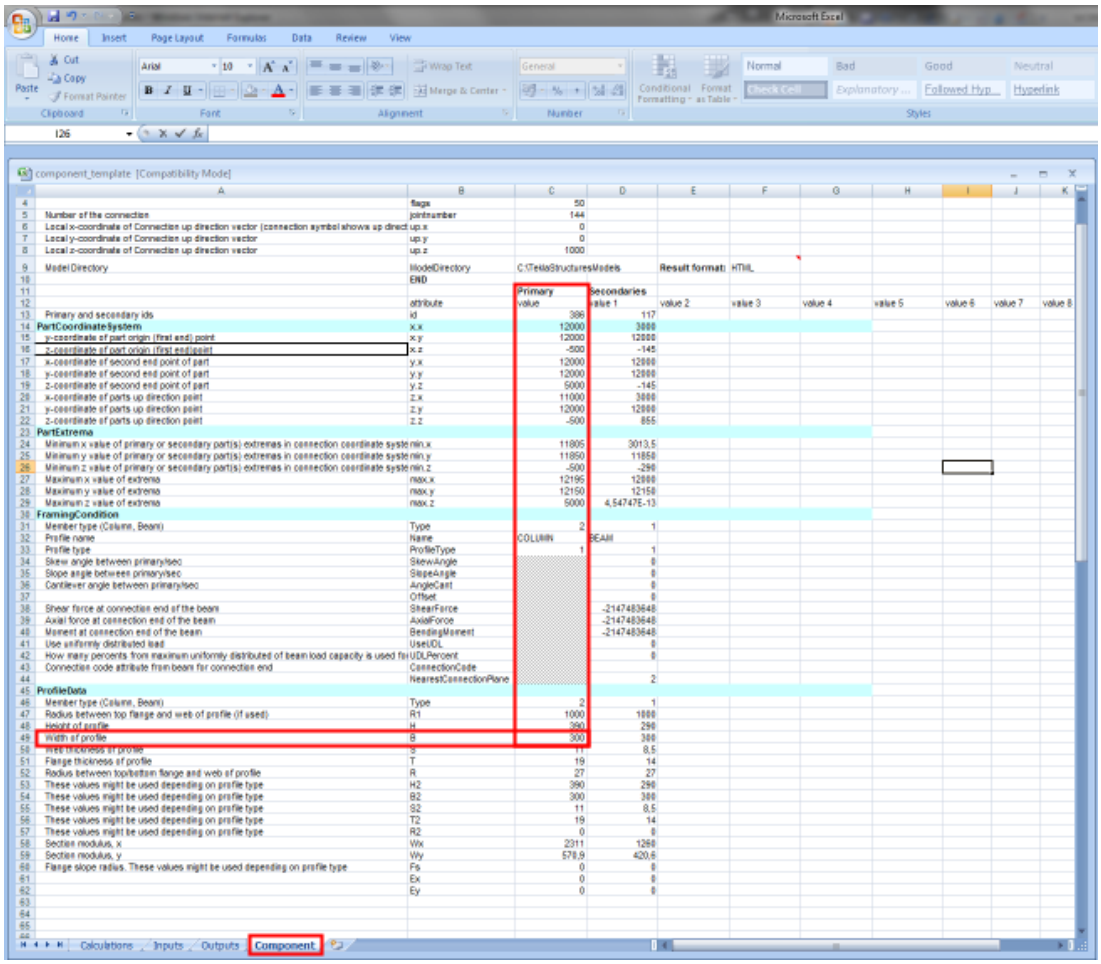


Figure 3.11 Binding 'Width of Profile'

4. Select the **Width of profile** *value field* on the **Component** sheet (the value from the model is brought here when the component is created or modified).

5. Link the value into the **Calculations** sheet as shown below.

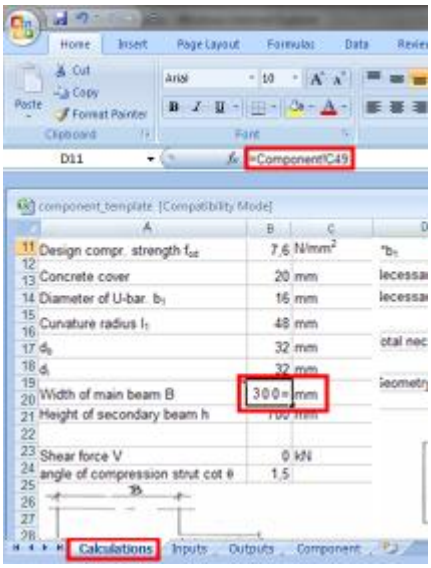


Figure 3.12 Beam width linked

6. In a similar fashion, link any other necessary fields from the **Inputs** and **Component** sheets to the **Calculations** sheet

   You can also link the component ID from the **Component** sheet to the **Calculations** sheet.

7. Lastly, the calculation results must be linked to the **Output** sheet.

   - Add an equality sign to the *Barnumber1* value field on the **Outputs** sheet.

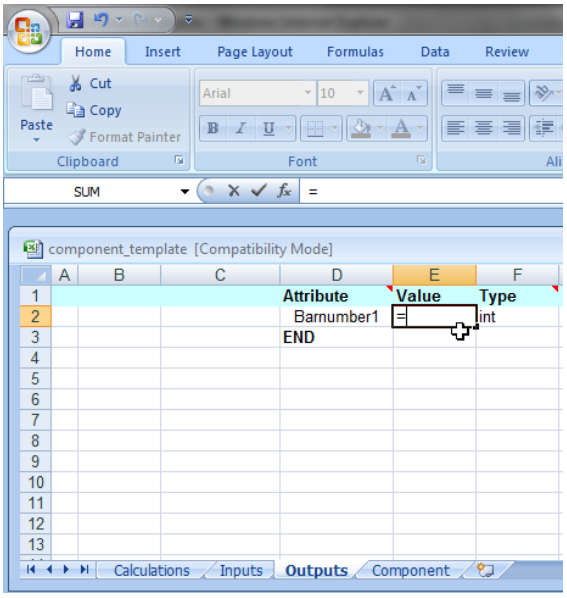


Figure 3.13 Adding an equality sign

- Select the final value in the *number of re-bars field* on the **Calculations** sheet.

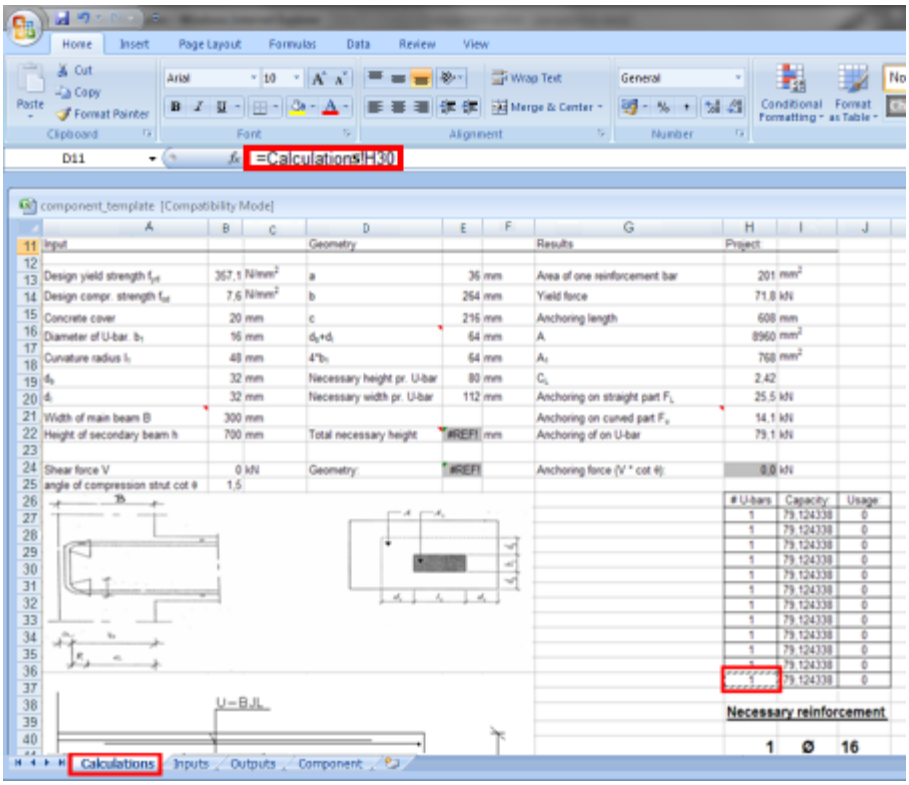


Figure 3.14 =Calculations!H30

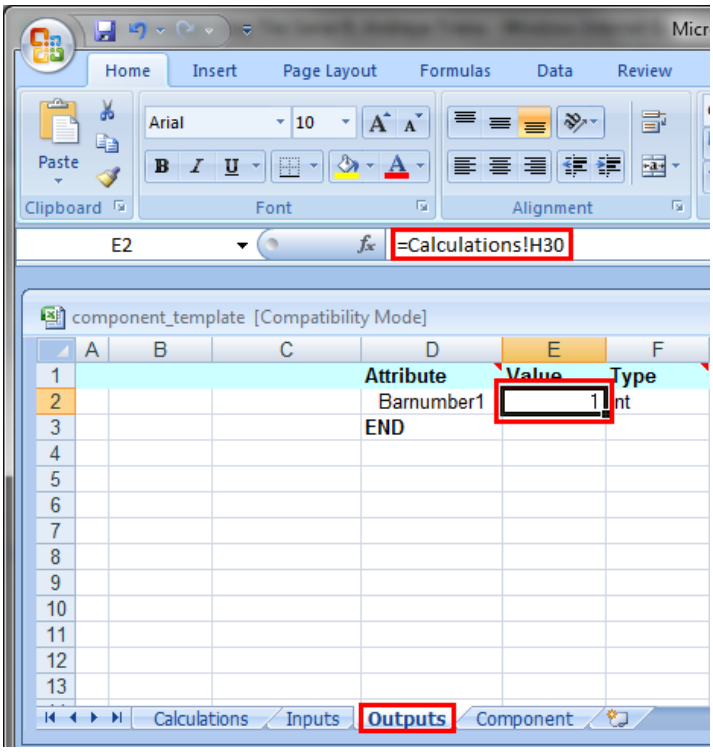


Figure 3.15 Re-bar amount linked to Barnumber1 value field

3.6 Test Excel sheet

After linking all the necessary fields, test the functionality. Give or change values in the **Inputs** sheet (and/or **Component** sheet) and check that values are transferred correctly to the **Calculations** sheet. It is also wise to check that the calculations are performed correctly. Check that the calculation results are transferred to the **Output** sheet as well. Once you have verified full functionality, **Save** and close Excel.

4. Test the incorporation of Excel sheet and Component

1. Double-click on the custom component in Tekla Structures.

2. Fill in values into the component dialog. Note that **any and all fields which will have information sent back to them from Excel must be empty** – in this case, the **Number of bars** value field. All other default values are sent to Excel.

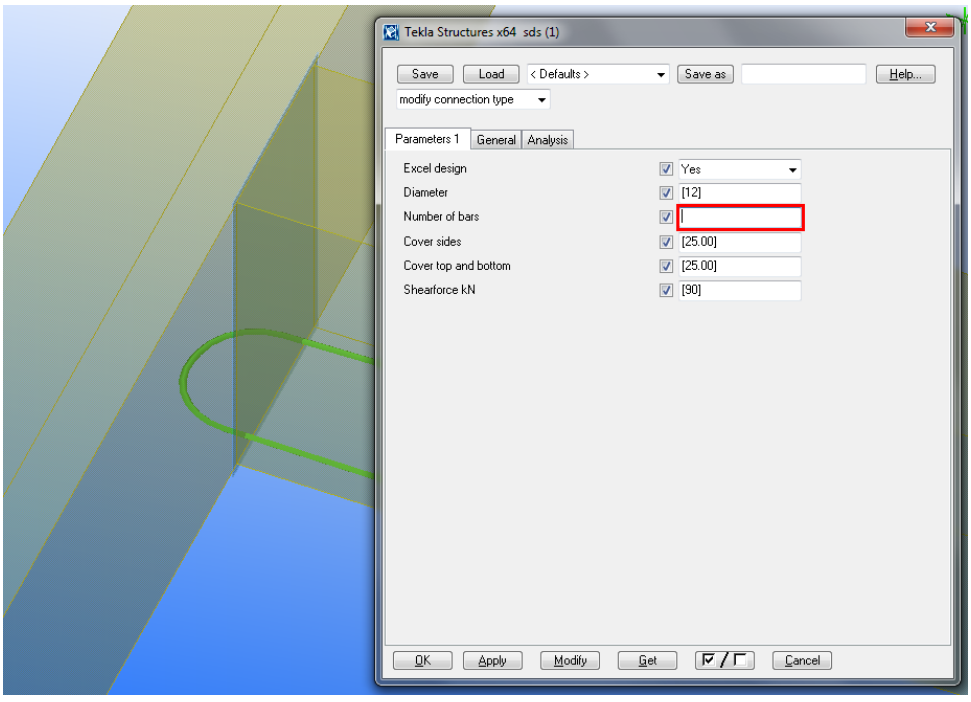


Figure 4.1 Custom component dialog box

3. Click **Modify**. The compiler service will start in a separate DOS window

Normally Excel performs the intended calculations automatically. Since we have activated debugging, however (see section 3.1), we can follow the calculations step by step.

1. The Excel **Inputs** sheet pops up displaying the input values from the component dialog. After the values have been set a small dialog box

appears telling you that the input values have been set. Click **OK** to continue.

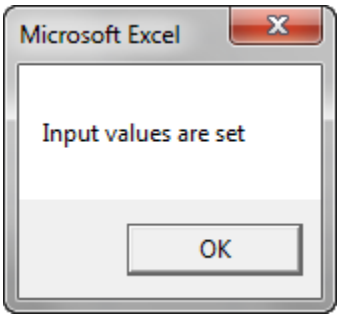


Figure 4.2 Input values set

2. Next, you will see all the values on the **Component** sheet update themselves. Another dialog box pops up, informing you that the component data has been set. Click **OK** to continue.

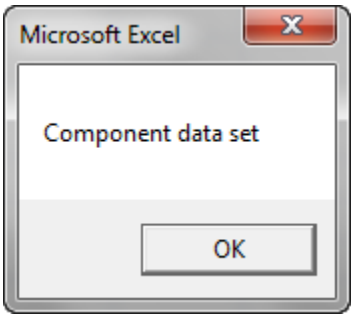


Figure 4.3 Component data set

3. Finally, the output data is set and read. Another dialog box notifies you of this occurrence. Click **OK** to continue.

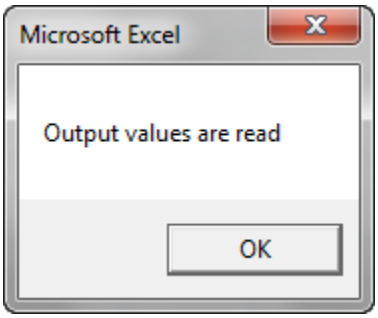


Figure 4.4 Output values

Excel may loop a few times before finishing if you are working with a custom component.

After Excel has gone through the calculation the values are transferred to component and the component is updated accordingly.

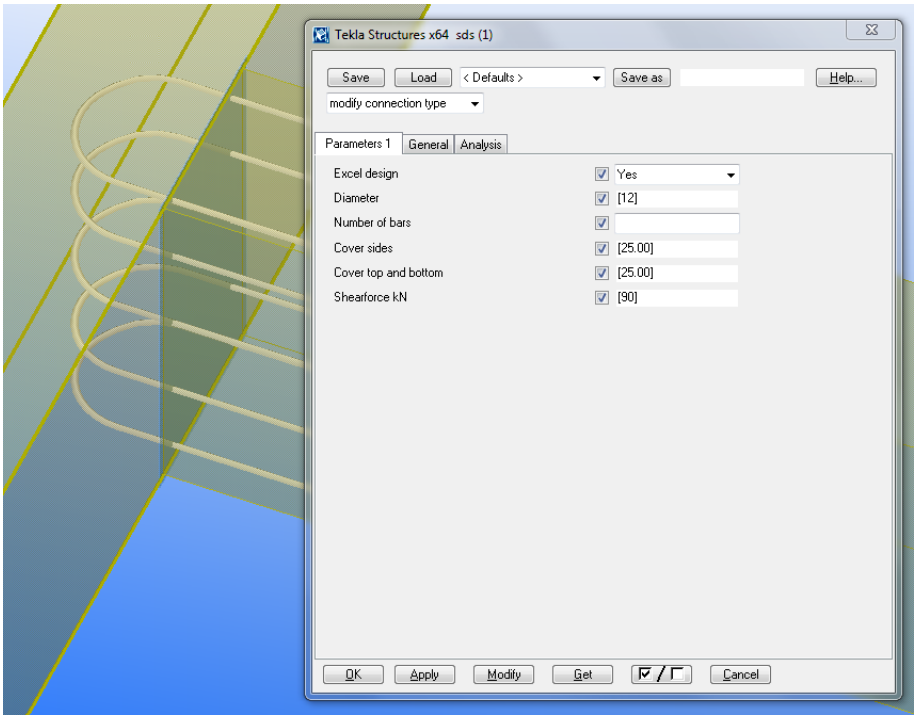


Figure 4.5 Modified custom component

5. Saving results and other editing

There are a lot of nifty features you can add to the Excel sheet! As an example, below are instructions on how to create a macro function that saves the calculation results in either .htm or .xls format.

5.1 Defining result file and path

The '*ModelDirectory*' field in the **Component** sheet is automatically updated according to the location of the component. Let's change the definition of the cell by defining a set folder to send the results to. The model folder itself being the root folder, we will define a subfolder directory for the results.

1. Pick a vacant cell and write `=C9 & "\" & "ExcelDesignResults"`. In this case, **C9** is the cell number of the '*ModelDirectory*' field in the **Components** sheet.

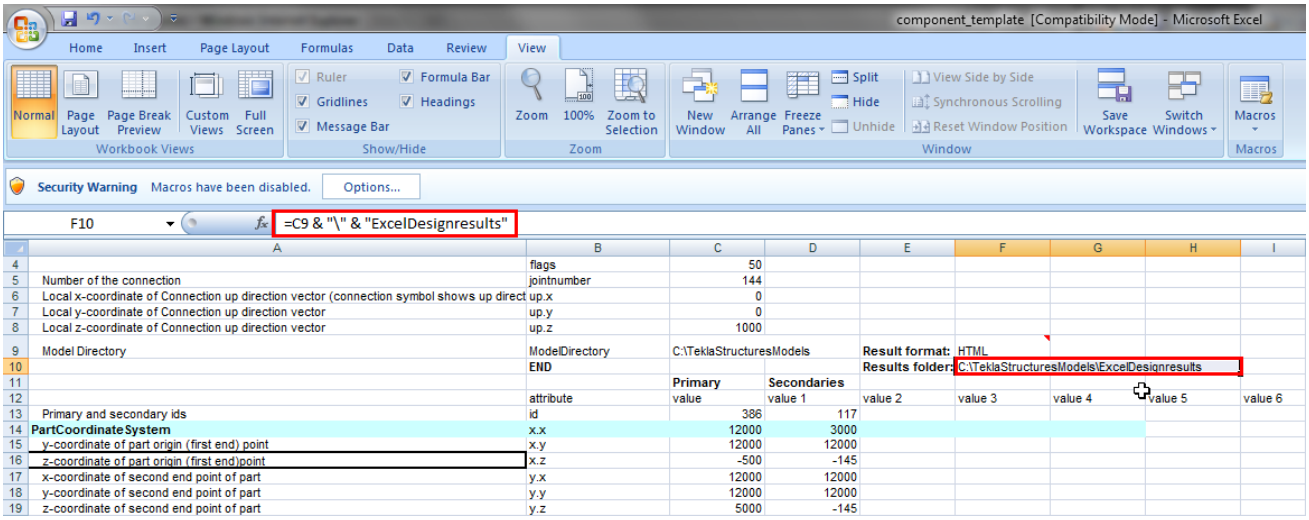


Figure 5.1 Result file path definition

2. To save the file path and ID on the **Calculations** sheet as well, pick a vacant cell and write:

   ```
   =Component!F10 & "\" & ComponentID & ".htm"
   ```

   (**F10** being the cell definition for the earlier defined result folder on the **Component** sheet.)

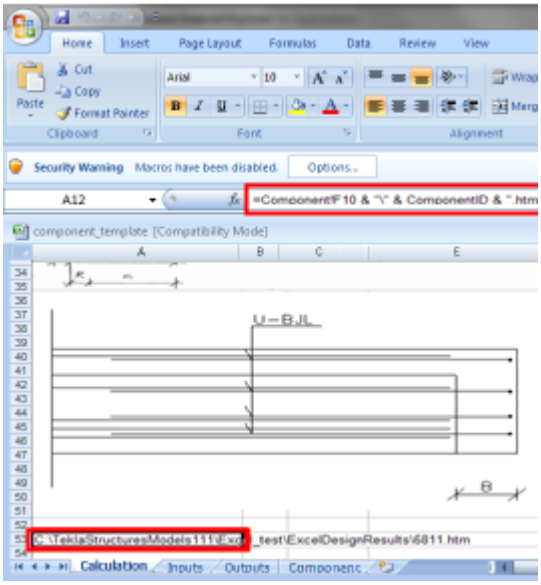


Figure 5.2 Path definition on Calculations sheet

5.2 Recording and editing an Excel macro

The next step is to create an *Excel Macro* that publishes the document in .htm format.

1. Go to the **View** tab in the Excel ribbon.

Figure 5.3 View tab

2. Click the **Macros** button and select **Record Macro…**

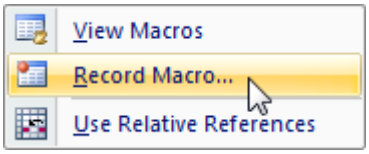


Figure 5.4 Macro recording

3. Define a macro name.

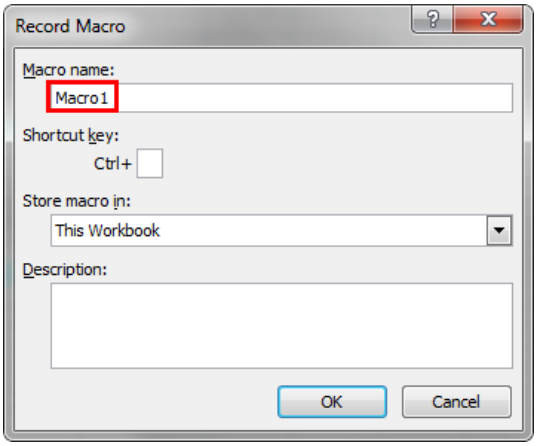


Figure 5.5 Record Macro

4. Click **OK** to start recording, then:

   a. Click the **Windows Office** button. 

   b. Select **Save as > Other Formats** in the drop-down box.

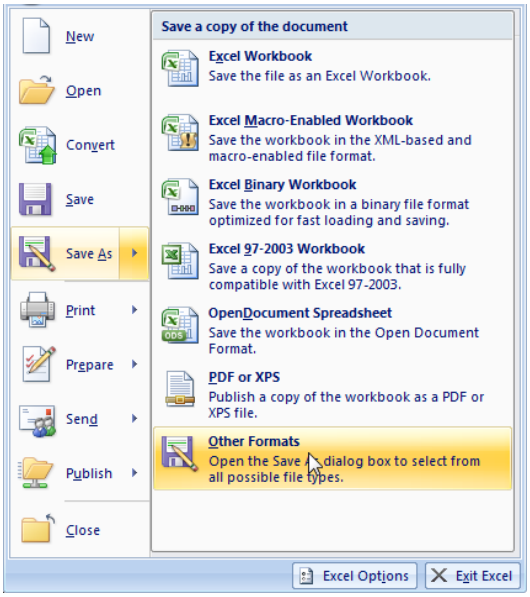


Figure 5.6 Save As – Other Formats

c. Select **Web Page** from the **Save as type** drop-down box and click the emerging **Publish** button.

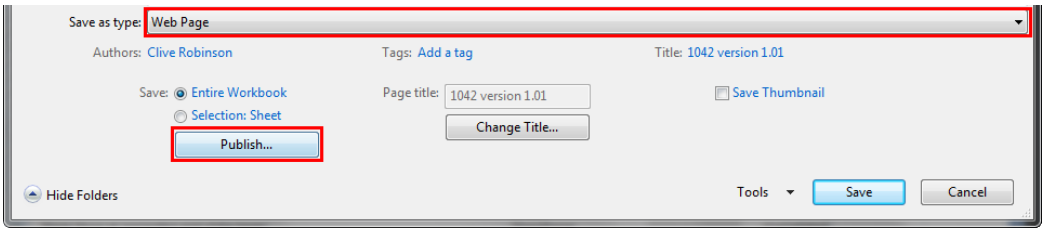


Figure 5.7 Publish Web Page

d. Select *Item on Calculations* from the **Choose** drop-down box and pick **Sheet All contents of Calculations**

e. Change the **File name** file path so that the results end up in the model folder – in this case the *Exceldesignresults* folder.

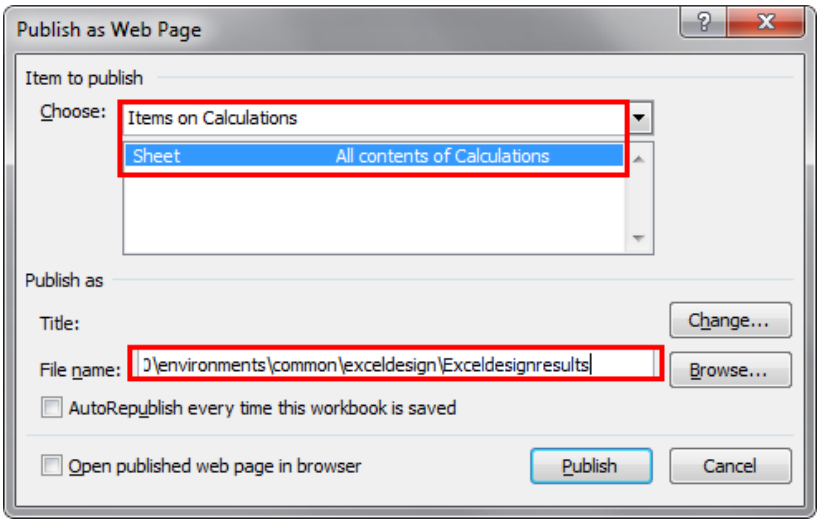


Figure 5.8 Publish as Web Page

f. Click **Publish**.

g. Go back to the **Macros** and click **Stop recording**.

The recorded macro publishes the **Calculations** sheet as a web page. However, it is still a "dumb" macro that contiuously uses the same file name and destination folder, effectively overwriting itself each time the macro is run. We will therefore configure the macro script itself.

1. Go to **Macros** in the **View** tab and click **View Macros** in the drop-down box.

2. Select the macro You recorded and click **Edit**. This opens the *Visual Basic editor*.

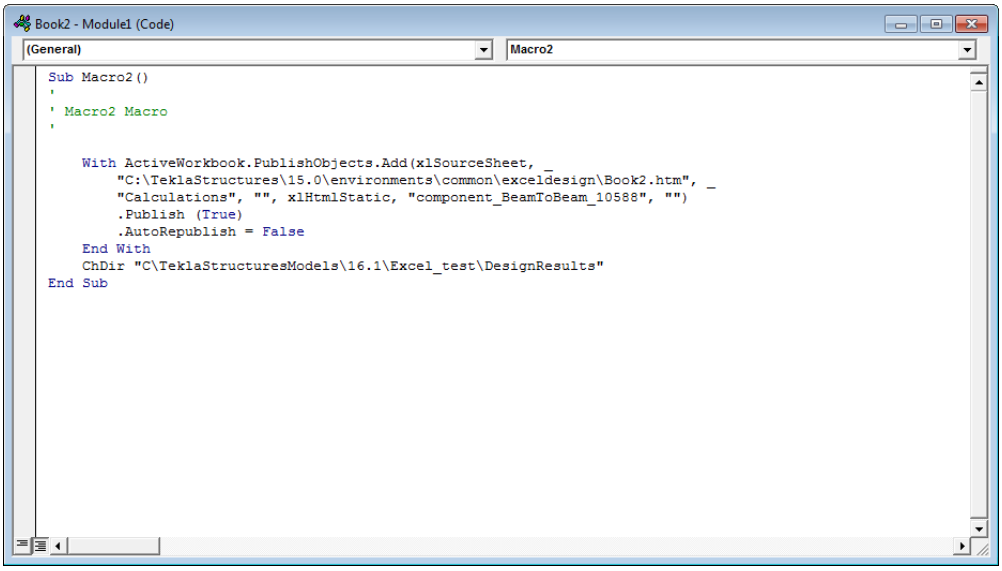


Figure 5.9 Visual Basic module

First, let's add a script that automatically creates the results' folder into the model folder (if necessary). Add

```
' Save Results Sheet
    report_folder = Worksheets("component").Range("F10").Value
    Set fs = CreateObject("Scripting.FileSystemObject")
    new_folder = fs.FolderExists(report_folder)
    If new_folder = 0 Then
    fs.CreateFolder (report_folder)
    End If
```

to the beginning of the main script as shown in figure 5.10. **Notice** that **F10** refers to the predefined cell in the **Component** sheet which defines the folder file path for the results (see figure 5.1).

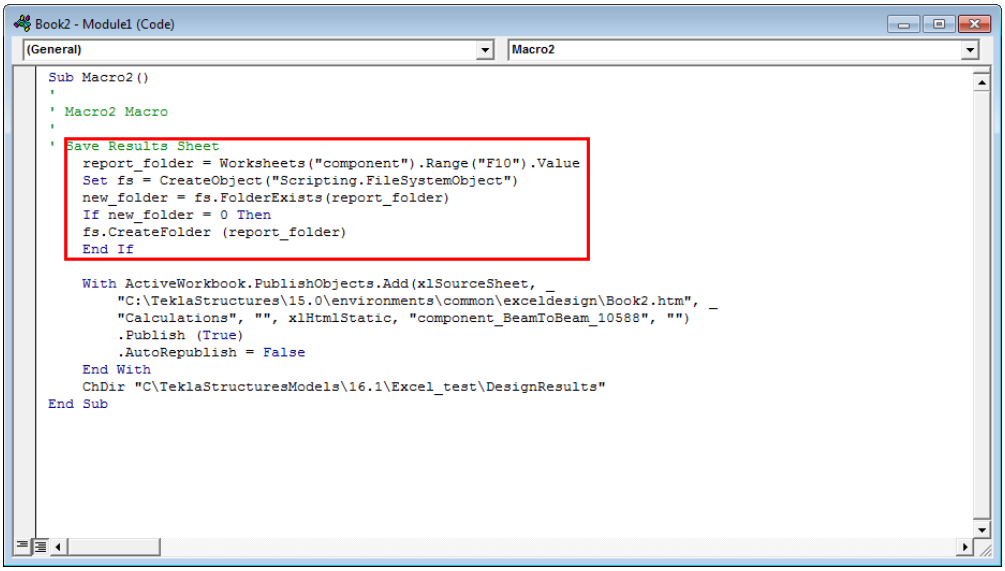
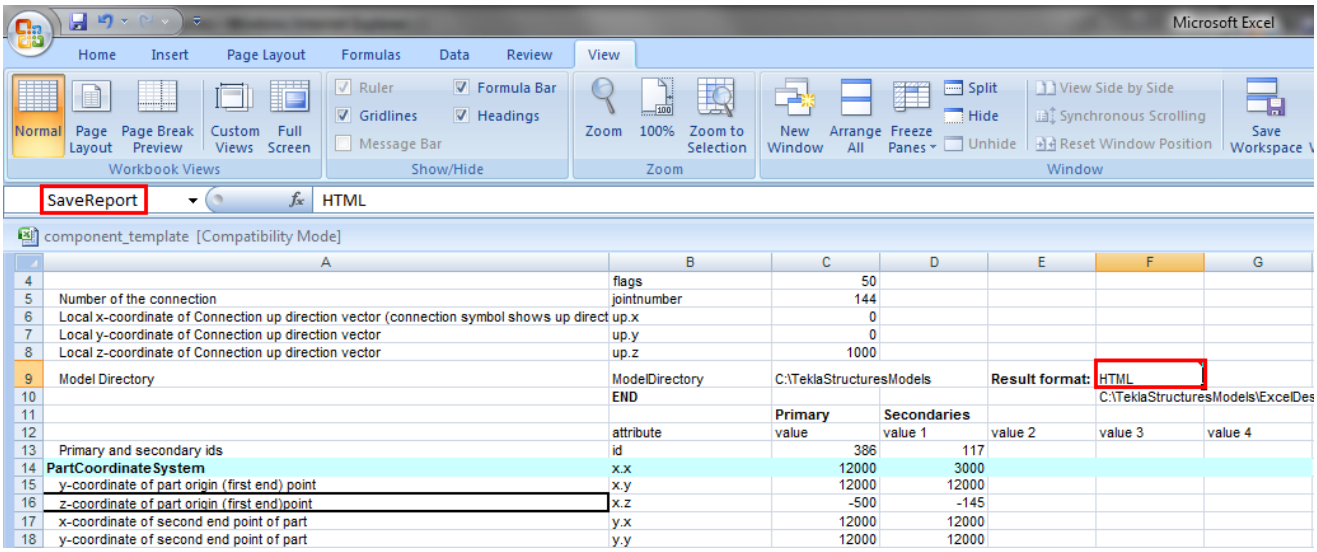


Figure 5.10 Adding script

Another script must be added, one which defines whether the file is saved in .htm or .xls format. If you select cell **F9** on the **Component** sheet, you will notice that the field is also named **SaveReport**.



	A	B	C	D	E	F	G
4		flags	50				
5	Number of the connection	jointnumber	144				
6	Local x-coordinate of Connection up direction vector (connection symbol shows up direct	up.x	0				
7	Local y-coordinate of Connection up direction vector	up.y	0				
8	Local z-coordinate of Connection up direction vector	up.z	1000				
9	Model Directory	ModelDirectory	C:\TeklaStructuresModels		Result format:	HTML	
10		END				C:\TeklaStructuresModels\ExcelDes	
11			Primary	Secondaries			
12		attribute	value	value 1	value 2	value 3	value 4
13	Primary and secondary ids	id	386	117			
14	PartCoordinateSystem	x.x	12000	3000			
15	y-coordinate of part origin (first end) point	x.y	12000	12000			
16	z-coordinate of part origin (first end)point	x.z	-500	-145			
17	x-coordinate of second end point of part	y.x	12000	12000			
18	y-coordinate of second end point of part	y.y	12000	12000			

Figure 5.11 SaveReport

Add the following script to the macro script as shown in figure 5.12. If the value in the '**SaveReport**' cell is **HTML** then the file is saved with a component ID number and a .htm extension.

```
If Names("SaveReport").RefersToRange.Value = "HTML" Then
        Sheets("Calculation").Select
        report_path = report_folder & "\" &
Worksheets("component").Range("C2").Value & ".htm"
```

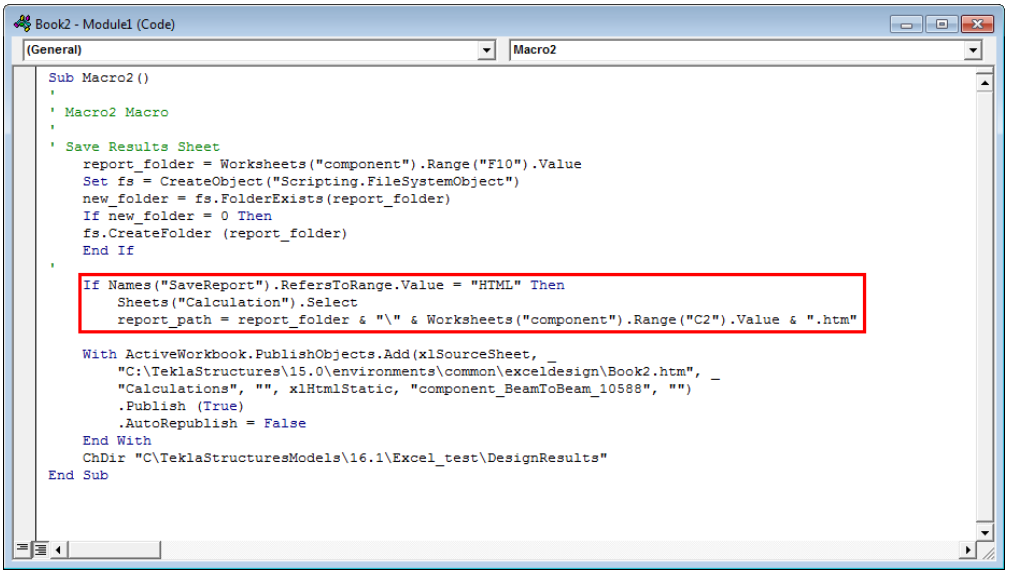


Figure 5.12 HTML extension script

Because of the added lines of code to the script which define where to save the results as well as the name of the file to be saved, we need to edit the original code a little bit to accommodate the changes.

Change the code from:

```
With ActiveWorkbook.PublishObjects.Add(xlSourceSheet, _
      "C:\TeklaStructuresModels111\Excel_test\ExcelDesignResults\Page.mht", _
      "Calculation", "", xlHtmlStatic, "component_beamtobeam_10588", "")
      .Publish (True)
      .AutoRepublish = False
   End With
   ChDir "C:\TeklaStructuresModels111\Excel_test\ExcelDesignResults"
```

To:

```
With ActiveWorkbook.PublishObjects("component_beamtobeam_10588")
      .HtmlType = xlHtmlStatic
      .Filename = _
      report_path
      .Publish (False)
      .AutoRepublish = False
End With
```

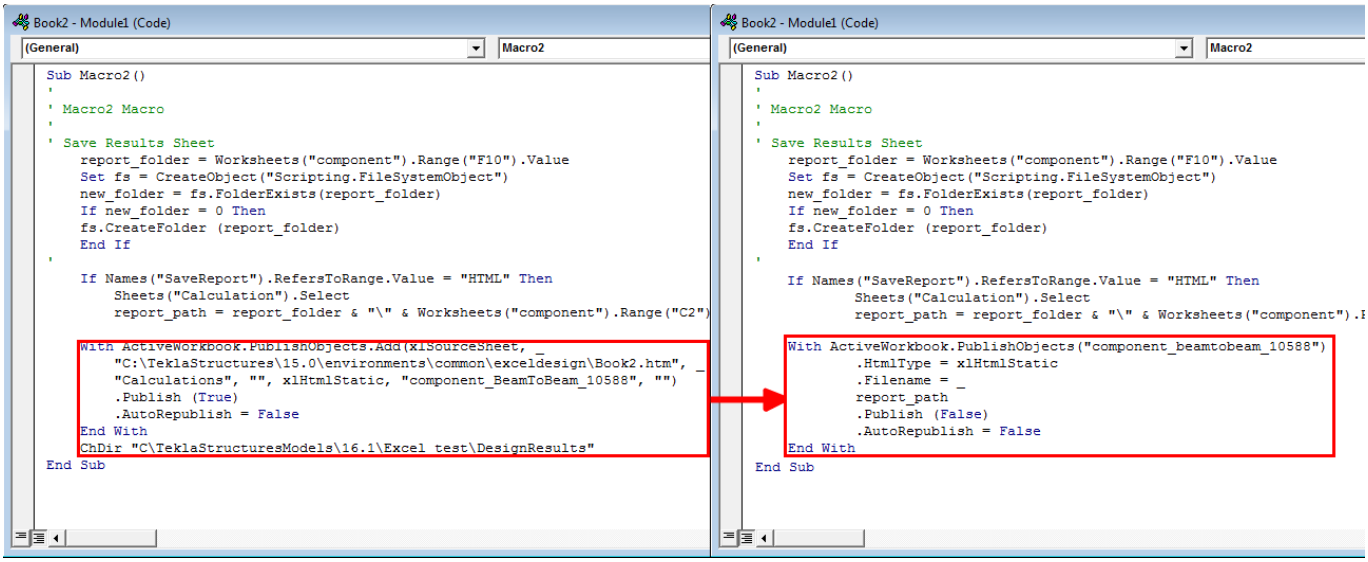


Figure 5.13 Core script alteration

Remember to copy/paste the **`component_name_number`** value from the original code to the new code (in this case, the "component_beamtobeam_10588" part marked with red).

The only thing left is to define an option to save the file in .xls format. Add the following script to the macro script – if the *value* written into the '**SaveReport**' cell is *WorkBook*, then the file is saved with a component ID number and a .xls extension.

```
    ElseIf Names("SaveReport").RefersToRange.Value = "WorkBook" Then
    'Save Workbook
    ActiveWorkbook.SaveAs Filename:=report_folder & "\" &
Worksheets("component").Range("C2").Value & ".xls"
    End If
```

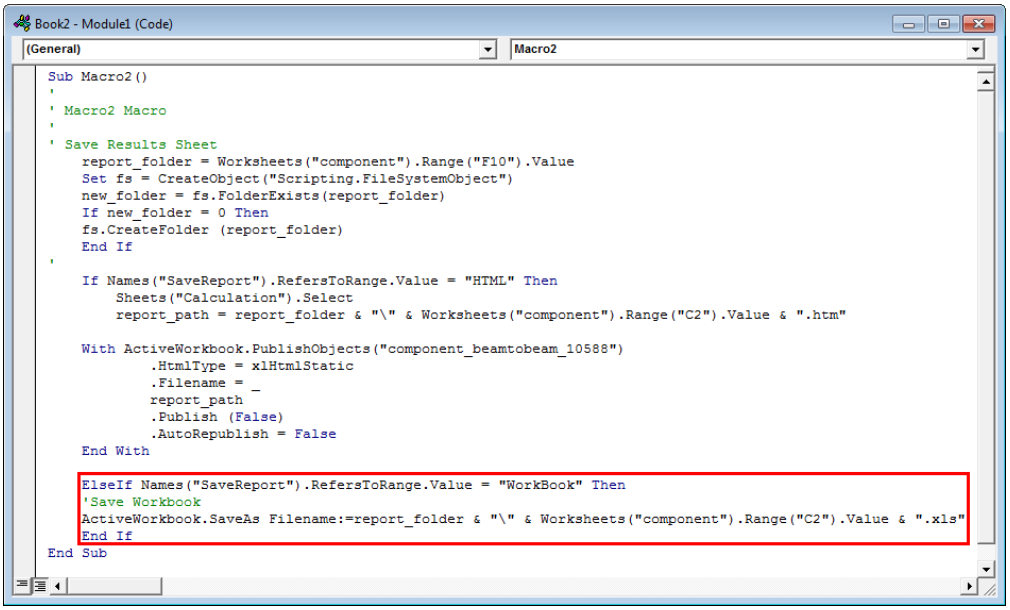


Figure 5.14 XLS extension script

Save the changes and close the editor.

Finally, test the macro by going to the **Macros** screen and selecting the macro from the list.

Click **Run**.

Check that the macro created the wanted folder in the designated file path folder.

5.2.1 Automating a macro

To make the created macro run automatically each time the component is modified we still need to add a *call function* for it.

1. Select **View Macros** in the **Macros** drop-down box.
2. Select the **StartCalculation** macro.
3. Click **Edit**.

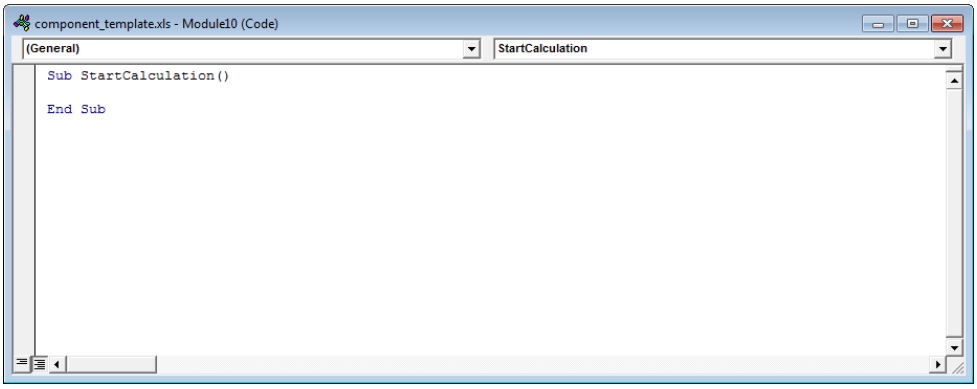


Figure 5.15 StartCalculation macro

4. Add a call for the previously created macro - in our case named *SaveResults* - by adding the line of code **`Call SaveResults`** as shown in figure 5.16.

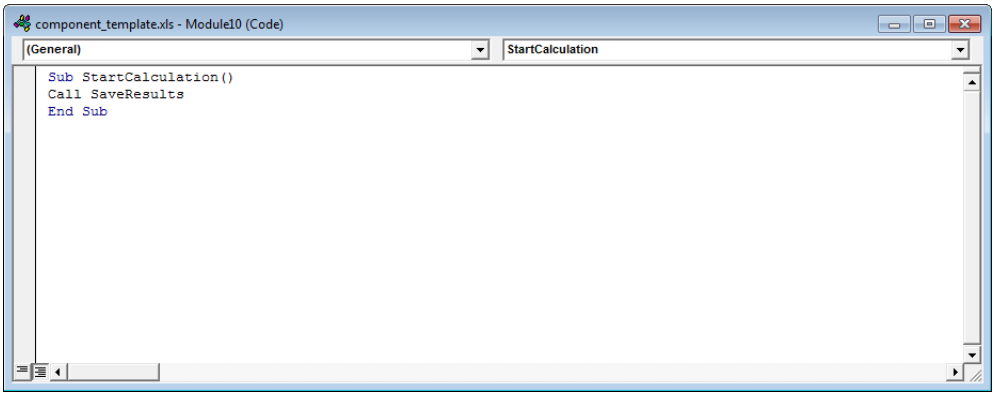
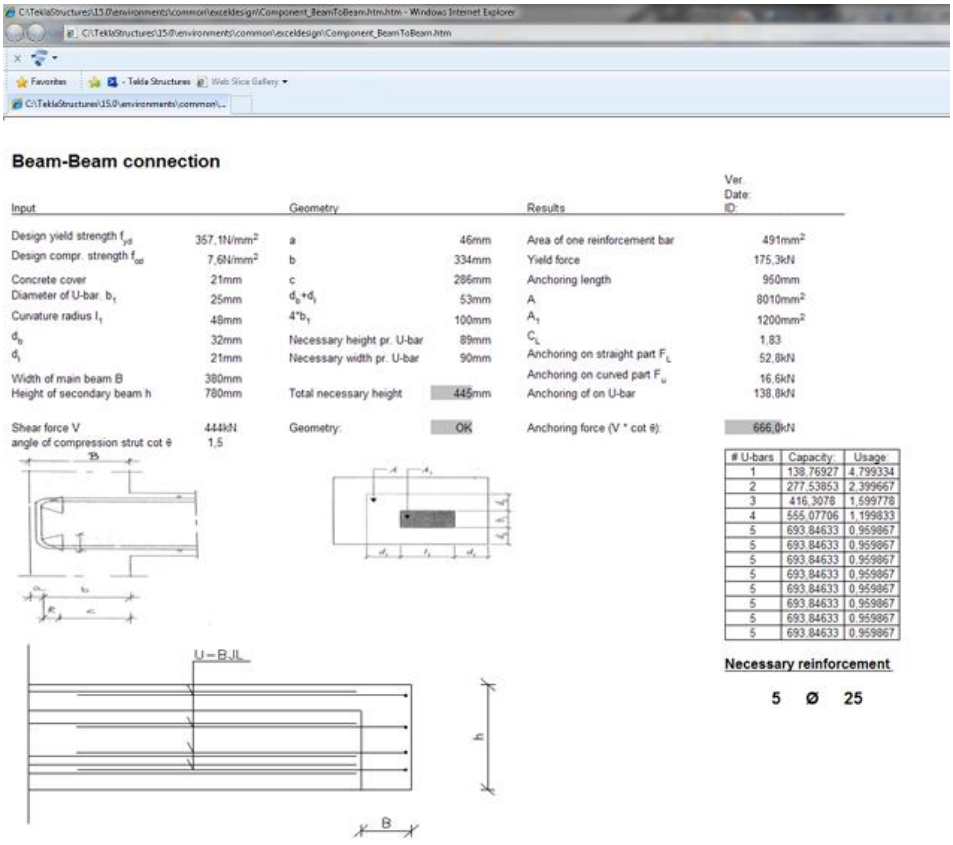


Figure 5.16 Call SaveResults

5. Save the changes and close the debugger.
6. Test the component.
   - Delete the *ExcelDesignResults* folder from the model folder.
   - Modify the component in Tekla Structures – the Excel macro should now automatically add both the folder and the result file.
   - Check that the content is correct when the result file is made in .htm format as well.



Beam-Beam connection

Ver.
Date:
ID:

Input		Geometry		Results	
Design yield strength f_{yd}	357,1N/mm²	a	46mm	Area of one reinforcement bar	491mm²
Design compr. strength f_{cd}	7,6N/mm²	b	334mm	Yield force	175,3kN
Concrete cover	21mm	c	286mm	Anchoring length	950mm
Diameter of U-bar b_1	25mm	d_0+d_1	53mm	A	8010mm²
Curvature radius l_1	48mm	$4*b_1$	100mm	A_1	1200mm²
d_0	32mm	Necessary height pr. U-bar	89mm	C_L	1,83
d_1	21mm	Necessary width pr. U-bar	90mm	Anchoring on straight part F_L	52,8kN
				Anchoring on curved part F_u	16,6kN
Width of main beam B	380mm				
Height of secondary beam h	780mm	Total necessary height	445mm	Anchoring of on U-bar	138,8kN
Shear force V	444kN	Geometry:	OK	Anchoring force (V * cot θ):	666,0kN
angle of compression strut cot θ	1,5				

# U-bars	Capacity:	Usage:
1	138,76927	4,799334
2	277,53853	2,399667
3	416,3078	1,599778
4	555,07706	1,199833
5	693,84633	0,959867
5	693,84633	0,959867
5	693,84633	0,959867
5	693,84633	0,959867
5	693,84633	0,959867
5	693,84633	0,959867
5	693,84633	0,959867
5	693,84633	0,959867

Necessary reinforcement

5 Ø 25

Figure 5.17 Results in .htm format

6. Finalizing the creation of the component

Turn off debugging so that the Excel calculations work “behind the scenes” again. Open **`Excel.vb`** and change the values back to their original state (see section 3.1 for location of file and changed values).

Test the component once more – the calculation runs its course and the results should be saved as before, but no windows nor dialog boxes should pop up.