```
Sub Main
        ' First we need to get important design information from an Excel spreadsheet
        ' This information let's us know how many shell plates (and their sizes) we need based on the length of the tank
        GetParametersFromExcel

        ' Before creating a new assembly, we need to make sure the inlet does not interfere with a hatch or tank seam on the top of the tank
        ' This is only applicable for tanks that have the inlet located on the top side of the tank body
        ' This will give an example of how to check inputs before committing to generating geometry
        If ValidateSeamsAndHatches = False Then Exit Sub

        ' Setup project folders, including sub-folders for sub-assemblies
        ' Then create a new copy of the this file, and rename it based on the PROJECT_ID
        If SetupProjectAndTopAssembly = False Then Exit Sub

        ' Create a new copy of the tank body assembly based on an existing tank body assembly template
        ' Then add it to the assembly at the origin and update all the parts based on user inputs
        CreateAndConfigureTankBody

        ' Create a new copy of the skid assembly based on an existing skid assembly template
        ' Then add it to the assembly at the origin and update all the parts based on user inputs and calculations
        CreateAndConfigureSkid

        ' Create a new copy of the gunline assembly based on an existing gunline assembly template
        ' Then add it to the assembly at the origin and update all the parts based on user inputs and calculations
        ' Note that gunline assemblies can only be used on tanks with an OD of 60" or greater
        If GUNLINE = True And TANK_OD >= 60 in Then CreateAndConfigureGunline

        ' Place the selected manway into the assembly, on the rear dish plate
        ' Note that manway assemblies can only be used on tanks with an OD of 48" or greater
        If MANWAY = True And TANK_OD >= 48 in Then InsertManwayIntoAssembly

        ' Place the hatch into the assembly on top of the tank body assembly
        ' User may have none, 1 or 2 total hatches in the assembly
        ' User may select hatches in the front or in the back, or both
        ' Note that hatch assemblies can only be used on tanks with an OD of 60" or greater
        InsertHatchesIntoAssembly

        ' Place the drain nozzles in the assembly
        ' Drain nozzles are always located near the bottom of the dish plates on each end
        ' User may select whether or not they want drains in the front and back, and what end connections to use
        ' Notes that if the tank diameter is 48" or below, 3" drains will be used; otherwise 4" drains will be used
        InsertDrainNozzlesIntoAssembly

        ' Place the Sump Nozzle
        If SUMP = True And TANK_OD >= 60 in Then CreateAndConfigureSump

        ' Place the Inlet Nozzle as required
        InsertInletIntoAssembly
End Sub

Sub CopyComponents(strFilePath As String, strAssemblyName As String, strFolderName As String)
        ' This function copies an assembly and all of its components to a new location
        ' It also updates the references in the new assembly so they point to the new components created
        ' If there are files in the assembly you do not want new copies of, then this routine will not work for you
        ' This uses the Inventor API to make a variable that references the assembly name passed to this routine
        ' It opens it up in the Inventor interface to see what is happening to the file
        Dim oAsmDoc As AssemblyDocument
        oAsmDoc = ThisApplication.Documents.Open(strFilePath & strAssemblyName, True)
        ' This is like selecting "Save As" in the Inventor interface, and saves a new copy of the assembly that was passed in
        oAsmDoc.SaveAs(PROJECT_PATH & PROJECT_ID & "\" & strFolderName & "\" & Left(strAssemblyName, strAssemblyName.Length - 4) & " - " & PROJECT_ID & ".iam", False)

        ' This continues to use the more advanced API routines available with Inventor
        ' A DocumentsEnumerator object will let us access the names of all the files referenced inside of the assembly
        Dim oRefDocs As DocumentsEnumerator
        oRefDocs = oAsmDoc.AllReferencedDocuments

        ' A document object represents a specific file reference in the Inventor assembly
        Dim oRefDoc As Document
        ' We will iterate through all of the currently referenced documents, and make copies of them
        For Each oRefDoc In oRefDocs
                Dim strNewFileName As String
                Dim strOldFileName As String
                strOldFileName = strFilePath & oRefDoc.DisplayName
                strNewFileName = PROJECT_PATH & PROJECT_ID & "\" & strFolderName & "\" & Left(oRefDoc.DisplayName, oRefDoc.DisplayName.Length - 4) & " - " & PROJECT_ID & ".ipt"
                ' Here is where a new copy of the referenced files are made, with a new name based on PROJECT_PATH and PROJECT_ID
                oRefDoc.SaveAs(strNewFileName, True)
                ' This code is like the "Replace Components" command inside of Inventor
                ' It replaces the reference to the old part, with a reference to the new part we recently created
                oAsmDoc.File.ReferencedFileDescriptors.Item(strOldFileName).ReplaceReference(strNewFileName)
        ' Next repeats this same process for all the files in our assembly, until we get to the end of the list
        Next
End Sub

Sub GetParametersFromExcel()
        ' This information is found on the "Shell Length Calcs" tab of the spreadsheet, and the first column represents the length of the tank
        ' Each tank body can have up to two different shell plate widths to cover the exact length of the tank
        ' SP1 represents the width of the first shell plate, and SP2 represents the width of the second shell plate
        ' Length Check and Total Plates columns are just for verification of the data, and are not read into this rule
        i = GoExcel.FindRow("C:\Automation Starter Kit\SK Excel File.xlsx", "Shell Length Calcs", "Length", "=", TANK_L)
        SHELL_W_1 = GoExcel.CurrentRowValue("SP1 Width")
        SHELL_W_2 = GoExcel.CurrentRowValue("SP2 Width")
        SHELL_Q_1 = GoExcel.CurrentRowValue("SP1 Qty")
        SHELL_Q_2 = GoExcel.CurrentRowValue("SP2 Qty")

        ' This information let's us know key parameters that are used to design the skid assembly
        ' This grabs several columns from the "Dish Depths" tab of the spreadsheet
        j = GoExcel.FindRow("C:\Automation Starter Kit\SK Excel File.xlsx", "Dish Depths", "Tank Diameter", "=", TANK_OD)
        DISH_DEPTH = GoExcel.CurrentRowValue("Dish Depth")
        SKID_FW = GoExcel.CurrentRowValue("Width")
        SKID_FH = GoExcel.CurrentRowValue("Height")
        SKID_FL_THK = GoExcel.CurrentRowValue("Flange")
        If TANK_OD >= 36 in Then SKID_WEB_THK = GoExcel.CurrentRowValue("Web")
        SKID_BEND_L = GoExcel.CurrentRowValue("Bend_L")
        SKID_ROD_D = GoExcel.CurrentRowValue("Rod_D")
        DRAIN_DISH_OFF = GoExcel.CurrentRowValue("Drain Offset Dish")

        ' This grabs information from the "Tubes" tab so that we can test if the input nozzle interferes with hatches or shell plate seams
        k = GoExcel.FindRow("C:\Automation Starter Kit\SK Excel File.xlsx", "Tubes", "Size", "=", Inlet_Size)
        INLET_PIPE_OD = GoExcel.CurrentRowValue("OD")
End Sub

Function ValidateSeamsAndHatches() As Boolean
        ' This function validates that the inlet location does not interfere with a hatch or tank seam
        ' If there's no interference, the value "True" is returned
        ' Otherwise, a messagebox let's the user know there was an error, and values will need to be re-entered
        Dim blnValid As Boolean = True

        If INLET_LOC = "Top" Then
                Dim dblFrontLoc, dblBackLoc As Double
                ' These two variables represent the two Z coordinates of the inlet pipe, including the front and back sides
                dblFrontLoc = INLET_OFF - INLET_PIPE_OD / 2
                dblBackLoc = INLET_OFF + INLET_PIPE_OD / 2
                ' This If statement does the math for the front hatch
                ' F_HATCH_OFF represents the offset value of the hatch from the front of the tank body (not including dish depths)
                ' SEAM_CLEAR_MIN represents the minimum clearance you want enforced to place inlets around seams
                ' 20 in represents how wide the actual hatch is; this will need to become a variable if more hatches are used in the future
                If F_HATCH Then
                        If (dblFrontLoc > F_HATCH_OFF - SEAM_CLEAR_MIN And dblFrontLoc < F_HATCH_OFF + 20 in + SEAM_CLEAR_MIN) Or _
                                (dblBackLoc > F_HATCH_OFF - SEAM_CLEAR_MIN And dblBackLoc < F_HATCH_OFF + 20 in + SEAM_CLEAR_MIN) Then
                                MessageBox.Show("The tank inlet will need to be moved to avoid interference with the front hatch." & vbCrLf & _
                                        "Avoid an inlet offset between " & F_HATCH_OFF - INLET_PIPE_OD / 2 - SEAM_CLEAR_MIN & Chr(34) & " and " & _
                                        F_HATCH_OFF + 20 in + INLET_PIPE_OD / 2 + SEAM_CLEAR_MIN & Chr(34) & ".")
                                blnValid = False
                                ' Show the main form before finishing the test and returning a value of False
                                iLogicForm.Show("Configure Tank")
                        End If
                End If

                ' This If statement does the math for the rear hatch
                ' R_HATCH_OFF represents the offset value of the hatch from the rear of the tank body (not including dish depths)
                ' SEAM_CLEAR_MIN represents the minimum clearance you want enforced to place inlets around seams
                ' 20 in represents how wide the actual hatch is; this will need to become a variable if more hatches are used in the future
                If R_HATCH Then
                        If (dblFrontLoc > TANK_L - R_HATCH_OFF - 20 in - SEAM_CLEAR_MIN And dblFrontLoc < TANK_L - R_HATCH_OFF + SEAM_CLEAR_MIN) Or _
                                (dblBackLoc > TANK_L - R_HATCH_OFF - 20 in - SEAM_CLEAR_MIN And dblBackLoc < TANK_L - R_HATCH_OFF + SEAM_CLEAR_MIN) Then
                                MessageBox.Show("The tank inlet will need to be moved to avoid interference with the rear hatch." & vbCrLf & _
                                        "Avoid an inlet offset between " & TANK_L - R_HATCH_OFF - 20 in - INLET_PIPE_OD / 2 - SEAM_CLEAR_MIN & Chr(34) & " and " & _
                                        TANK_L - R_HATCH_OFF + INLET_PIPE_OD / 2 + SEAM_CLEAR_MIN & Chr(34) & ".")
                                blnValid = False
                                ' Show the main form before finishing the test and returning a value of False
                                iLogicForm.Show("Configure Tank")
                        End If
                End If

                ' This If statement does the math for the seam clearance calculations
                ' TANK_L represents the length of the tank (not including dish depths)
                ' SHELL_Q_1 and SHELL_Q_2 represent how many plates of width 1 and width 2 are required to create the tank body
                ' SHELL_W_1 and SHELL_W_2 represent the widths of the shell plates used to create the tank shell body
                If TANK_L > 72 in Then
                        Dim dblSeamLocation As Double
                        ' The For statement will take us from one seam of the tank to the next, until we pass where the inlet is located
                        For seam = 1 To SHELL_Q_1 + SHELL_Q_2 - 1
                                If seam <= SHELL_Q_1 Then
                                        dblSeamLocation = SHELL_W_1 * seam
                                Else
                                        dblSeamLocation = SHELL_Q_1 * SHELL_W_1 + (seam - SHELL_Q_1) * SHELL_W_2
                                End If
                                ' This statement will show you how to create a compound If statement using "And" and "Or" operators
                                If (dblFrontLoc > dblSeamLocation - SEAM_CLEAR_MIN And dblFrontLoc < dblSeamLocation + SEAM_CLEAR_MIN) Or _
                                        (dblBackLoc > dblSeamLocation - SEAM_CLEAR_MIN And dblBackLoc < dblSeamLocation + SEAM_CLEAR_MIN) Or _
                                        (INLET_OFF > dblSeamLocation - SEAM_CLEAR_MIN And INLET_OFF < dblSeamLocation + SEAM_CLEAR_MIN) Then
                                        MessageBox.Show("The tank inlet will need to be moved to avoid interference with one of the seams." & vbCrLf & _
                                                "Avoid an inlet offset between " & dblSeamLocation - INLET_PIPE_OD / 2 - SEAM_CLEAR_MIN & Chr(34) & " and " & _
                                                dblSeamLocation + INLET_PIPE_OD / 2 + SEAM_CLEAR_MIN & Chr(34) & ".")
                                        blnValid = False
                                        ' Show the main form before finishing the test and returning a value of False
                                        iLogicForm.Show("Configure Tank")
                                End If
                        Next
                End If
        End If
        ValidateSeamsAndHatches = blnValid
End Function

Function SetupProjectAndTopAssembly() As Boolean
        ' This function checks to make sure a project doesn't already exist that the user is requesting
        ' It then sets up a folder structure, and does a "Save As" to create a new top-level assembly that will be used
        ' This first set of statements uses the Windows System object to create a folder structure for our files
        Dim blnSetupSucceeded As Boolean = True
        ' This statement first checks to see if the folder already exists
        ' If it does already exist, it won't bother creating the folder again
        If System.IO.Directory.Exists(PROJECT_PATH & PROJECT_ID) = False Then
                System.IO.Directory.CreateDirectory(PROJECT_PATH & PROJECT_ID)
                System.IO.Directory.CreateDirectory(PROJECT_PATH & PROJECT_ID & "\Tank Body Assy")
                System.IO.Directory.CreateDirectory(PROJECT_PATH & PROJECT_ID & "\Skid Assy")
                ' If user has configured a gunline, then create a folder to store its files
                If GUNLINE Then System.IO.Directory.CreateDirectory(PROJECT_PATH & PROJECT_ID & "\Gunline Assy")
                ' If a user have configured a sump, then create a folder to store its files
                If SUMP Then System.IO.Directory.CreateDirectory(PROJECT_PATH & PROJECT_ID & "\Sump Assy")
        End If

        ' Now that we have our folder structure in place, we are ready to save the Master Assembly file
        ' It uses the path stored in the PROJECT_PATH parameter, and the PROJECT_ID to give the assembly a unique name
        Dim sMasterAssy As String
        sMasterAssy = PROJECT_PATH & PROJECT_ID & "\Tank Assembly - " & PROJECT_ID & ".iam"
        ' We first check to make sure the master assembly file doesn't already exist, then save it if it doesn't
        If System.IO.File.Exists(sMasterAssy) = False Then
                ThisDoc.Document.SaveAs(sMasterAssy , False)
        Else
                MessageBox.Show("Assembly Already Exists", "Master")
                iLogicForm.Show("Configure Tank")
                blnSetupSucceeded = False
        End If
        ' Return whether or not we were successful creating the new folder structure and master assembly file
        SetupProjectAndTopAssembly = blnSetupSucceeded
End Function

Sub CreateAndConfigureTankBody()
        ' This code creates a new copy of the tank body assembly template in our new folder structure
        ' It then updates the tank body geometry based on values we pass to the assembly
        Dim strNewTankBodyFileName As String
        ' This string represents the new name of our unique, copied tank body assembly file
        strNewTankBodyFileName = PROJECT_PATH & PROJECT_ID & "\Tank Body Assy\Tank Body Assy - " & PROJECT_ID & ".iam"

        ' We first check to make sure the tank body assembly file has not been previously created
        If System.IO.File.Exists(strNewTankBodyFileName) = False Then
                ' This here is a sample of how to make a variable that represents an assembly document
                ' It uses the Inventor API, which you can use freely (for the most part) throughout iLogic rules
                Dim subAssy1 As AssemblyDocument
                ' This statement tells subAssy1 to represent the template file, and opens it up in the Inventor interface
                subAssy1 = ThisApplication.Documents.Open(TEMPLATE_PATH & "Tank Body Assy\Tank Body Assy.iam", True)
                ' This code is like selecting "File Save As" in the Inventor interface, and we now have our new file saved
                subAssy1.SaveAs(strNewTankBodyFileName, False)
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts our newly created tank body assembly into our master tank assembly file
                Dim componentA = Components.Add("Tank Body Assy:1", strNewTankBodyFileName, position := Nothing, grounded := True, visible := True, appearance := Nothing)
                ' This will now close the new tank body assembly file
                subAssy1.Close
                ' This code calculates the horizontal and vertical locations of our gunline assembly (for later use)
                GUNLINE_HOR_OFF = Round((TANK_OD / 2) * .6667)
                GUNLINE_VERT_OFF = Round((TANK_OD / 2) * .25)
                ' This code represents our typical manway offset, which is 26" above the bottom of the tank
                MANWAY_VERT_OFF = -TANK_OD / 2 + 26
                ' These statements pass parameters from our master assembly file into the tank body assembly file
                Parameter("Tank Body Assy:1", "PROJECT_ID") = PROJECT_ID
                Parameter("Tank Body Assy:1", "PROJECT_PATH") = PROJECT_PATH
                Parameter("Tank Body Assy:1", "TANK_OD") = TANK_OD
                Parameter("Tank Body Assy:1", "TANK_L") = TANK_L
                Parameter("Tank Body Assy:1", "SHELL_W_1") = SHELL_W_1
                Parameter("Tank Body Assy:1", "SHELL_W_2") = SHELL_W_2
                Parameter("Tank Body Assy:1", "SHELL_Q_1") = SHELL_Q_1
                Parameter("Tank Body Assy:1", "SHELL_Q_2") = SHELL_Q_2
                Parameter("Tank Body Assy:1", "GUNLINE_SIZE") = GUNLINE_SIZE
                Parameter("Tank Body Assy:1", "GUNLINE_VERT_OFF") = GUNLINE_VERT_OFF
                Parameter("Tank Body Assy:1", "GUNLINE_HOR_OFF") = GUNLINE_HOR_OFF
                Parameter("Tank Body Assy:1", "MANWAY_VERT_OFF") = MANWAY_VERT_OFF
                Parameter("Tank Body Assy:1", "GUNLINE") = GUNLINE
                Parameter("Tank Body Assy:1", "MANWAY") = MANWAY
                Parameter("Tank Body Assy:1", "F_HATCH") = F_HATCH
                Parameter("Tank Body Assy:1", "F_HATCH_OFF") = F_HATCH_OFF
                Parameter("Tank Body Assy:1", "R_HATCH") = R_HATCH
                Parameter("Tank Body Assy:1", "R_HATCH_OFF") = R_HATCH_OFF
                Parameter("Tank Body Assy:1", "INLET_OFF") = INLET_OFF
                Parameter("Tank Body Assy:1", "INLET_PIPE_OD") = INLET_PIPE_OD
                ' Once all the parameters are updated in the tank body assembly file, we want to run their rules
                ' This will allow the tank body assembly to update all its own parts and components itself
                iLogicVb.RunRule("Tank Body Assy:1", "Size Dish")
                iLogicVb.RunRule("Tank Body Assy:1", "Calculate and Place Shells")
        End If
End Sub

Sub CreateAndConfigureSkid()
        ' This code creates a new copy of the skid assembly template in our new folder structure
        ' It then updates the skid geometry based on values we pass to the assembly
        Dim strNewSkidFilename As String
        ' This string represents the new name of our unique, copied skid assembly file
        strNewSkidFilename = PROJECT_PATH & PROJECT_ID & "\Skid Assy\Skid Assy - " & PROJECT_ID & ".iam"

        ' We first check to make sure the skid assembly file has not been previously created
        If System.IO.File.Exists(strNewSkidFilename) = False Then
                ' This here is a sample of how to make a variable that represents an assembly document
                ' It uses the Inventor API, which you can use freely (for the most part) throughout iLogic rules
                Dim subAssy1 As AssemblyDocument
                ' This code uses the "CopyComponents" subroutine (see above) to copy the skid assembly, and all its children
                ' This will not work if you don't want some of the parts in the assembly to have unique copies
                ' The function also changes the references in the skid assembly to point to the newly created part files
                CopyComponents(TEMPLATE_PATH & "Skid Assy\", "Skid Assy.iam", "Skid Assy")
                ' This statement tells subAssy1 to represent the template file, and opens it up in the Inventor interface
                subAssy1 = ThisApplication.Documents.Open(strNewSkidFilename, True)
                ' This statement changes the occurrence names of the existing skid components in the model browser
                ' This will allow the rule that passes parameters in our skid sub-assembly to still work
                subAssy1.ComponentDefinition.Occurrences(1).Name = "Skid-1:1"
                subAssy1.ComponentDefinition.Occurrences(2).Name = "Skid-1:2"
                subAssy1.Save
                subAssy1.Close
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts our newly created skid assembly into our master tank assembly file
                Dim componentB = Components.Add("Skid Assy:1", strNewSkidFilename, position := Nothing, grounded := True, visible := True, appearance := Nothing)
                ' Change our flange radius if the TANK_OD is 30" or less
                If TANK_OD <= 30 in    Then SKID_FLG_RAD = .1 in
                ' These statements pass parameters from our master assembly file into the skid assembly file
                Parameter("Skid Assy:1", "PROJECT_ID") = PROJECT_ID
                Parameter("Skid Assy:1", "PROJECT_PATH") = PROJECT_PATH
                Parameter("Skid Assy:1", "TANK_OD") = TANK_OD
                Parameter("Skid Assy:1", "TANK_L") = TANK_L
                Parameter("Skid Assy:1", "SHELL_W_1") = SHELL_W_1
                Parameter("Skid Assy:1", "SHELL_W_2") = SHELL_W_2
                Parameter("Skid Assy:1", "SHELL_Q_1") = SHELL_Q_1
                Parameter("Skid Assy:1", "SHELL_Q_2") = SHELL_Q_2
                Parameter("Skid Assy:1", "SKID_FW") = SKID_FW
                Parameter("Skid Assy:1", "SKID_FH") = SKID_FH
                Parameter("Skid Assy:1", "SKID_FL_THK") = SKID_FL_THK
                Parameter("Skid Assy:1", "SKID_FLG_RAD") = SKID_FLG_RAD
                Parameter("Skid Assy:1", "SKID_WEB_THK") = SKID_WEB_THK
                Parameter("Skid Assy:1", "SKID_BEND_L") = SKID_BEND_L
                Parameter("Skid Assy:1", "SKID_ROD_D") = SKID_ROD_D
                Parameter("Skid Assy:1", "DISH_DEPTH") = DISH_DEPTH
                ' Once all the parameters are updated in the skid assembly file, we want to run its creation rule
                ' This will allow the skid assembly to update all its own parts and components itself
                iLogicVb.RunRule("Skid Assy:1", "Create Skid")
        End If
End Sub

Sub CreateAndConfigureGunline()
        ' This code creates a new copy of the gunline assembly template in our new folder structure
        ' It then updates the gunline geometry based on values we pass to the assembly
        Dim strNewGunlineFilename As String
        ' This string represents the new name of our unique, copied gunline assembly file
        strNewGunlineFilename = PROJECT_PATH & PROJECT_ID & "\Gunline Assy\Gunline Assy - " & PROJECT_ID & ".iam"

        ' We first check to make sure the gunline assembly file has not been previously created
        ' If it hasn't yet been created, we do a Windows Copy operation to make a new copy in our new folder
        If System.IO.File.Exists(strNewGunlineFilename) = False Then
                System.IO.File.Copy(TEMPLATE_PATH & "Gunline Assy\Gunline Assy.iam", strNewGunlineFilename)
        End If

        ' In order to locate where to put the gunline assembly in our master assembly file, we will use matrix positioning
        ' See presentation included in this kit that explains how matrix positioning works - it's easier than it looks or sounds
        Dim matrixC = ThisDoc.Geometry.Matrix(-1, 0, 0, GUNLINE_HOR_OFF, 0, 1, 0, -GUNLINE_VERT_OFF, 0, 0, -1, TANK_L / 2 + DISH_DEPTH,    0, 0, 0, 1)
        ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
        ' This code inserts our newly created gunline assembly into our master tank assembly file
        ' Instead of placing at the origin, it places it based on our input matrix we created (matrixC)
        ' Note that we are grounding all geometry, and we are not using any constraints to place the assembly
        Dim componentC = Components.Add("Gunline Assy:1", strNewGunlineFilename, position := matrixC, grounded := True, visible := True, appearance := Nothing)
        ' These statements pass parameters from our master assembly file into the gunline assembly file
        Parameter("Gunline Assy:1", "TANK_OD") = TANK_OD
        Parameter("Gunline Assy:1", "TANK_L") = TANK_L
        Parameter("Gunline Assy:1", "PROJECT_ID") = PROJECT_ID
        Parameter("Gunline Assy:1", "PROJECT_PATH") = PROJECT_PATH
        Parameter("Gunline Assy:1", "SHELL_Q_1") = SHELL_Q_1
        Parameter("Gunline Assy:1", "SHELL_Q_2") = SHELL_Q_2
        Parameter("Gunline Assy:1", "GUNLINE_SIZE") = GUNLINE_SIZE
        Parameter("Gunline Assy:1", "GUNLINE_F_FL_TYPE") = GUNLINE_F_FL_TYPE
        Parameter("Gunline Assy:1", "GUNLINE_R_FL_TYPE") = GUNLINE_R_FL_TYPE
        Parameter("Gunline Assy:1", "GUNLINE_F_FL_END") = GUNLINE_F_FL_END
        Parameter("Gunline Assy:1", "GUNLINE_R_FL_END") = GUNLINE_R_FL_END
        Parameter("Gunline Assy:1", "DISH_DEPTH") = DISH_DEPTH
        ' Once all the parameters are updated in the gunline assembly file, we want to run its creation rule
        ' This will allow the gunline assembly to update all its own parts and components itself
        iLogicVb.RunRule("Gunline Assy:1", "Set Gunline Size")
        iLogicVb.RunRule("Gunline Assy:1", "Calculate Gunline Spacing")
        iLogicVb.RunRule("Gunline Assy:1", "Push Parameters")
        iLogicVb.RunRule("Gunline Assy:1", "Spray Nozzle Length")
        iLogicVb.RunRule("Gunline Assy:1", "Assemble Flanges")
End Sub

Sub InsertManwayIntoAssembly()
        ' This code places the selected manway into our assembly, if applicable
        ' Manways are always place on the rear dish head plate
        ' We first need to calculate the Z-value to place our manway so it doesn't interfere with the dish head plate
        Dim dblHorizontalOffset As Double
        ' This calculates our initial horizontal offset based on the length of the tank, and placement on the dish head plate
        If MANWAY_VERT_OFF < 0 Then
                dblHorizontalOffset = -(TANK_L / 2 - (MANWAY_VERT_OFF / (TANK_OD / 2)) * DISH_DEPTH + MANWAY_HOR_OFF)
        Else
                dblHorizontalOffset = -(TANK_L / 2 + (MANWAY_VERT_OFF / (TANK_OD / 2)) * DISH_DEPTH + MANWAY_HOR_OFF)
        End If

        ' Even though we made an initial calculation for horizontal placement, there was some interference with the dish head plate
        ' With more time, I could have come up with a better calculation than the one above that would have been more accurate
        ' For the sake of timing and getting this done, I added different offset values based on empirical testing
        ' First, we start with the code to place a 21 inch manway, if that has been selected
        If MANWAY_SIZE = 21 in Then
                If TANK_OD >= 54 in  And TANK_OD <= 90 in  Then dblHorizontalOffset -= 6 in
                If TANK_OD >= 96 in  And TANK_OD <= 102 in   Then dblHorizontalOffset -= 4 in
                If TANK_OD >= 108 in  And TANK_OD <= 114 in   Then dblHorizontalOffset -= 2.5 in
                If TANK_OD = 120 Then dblHorizontalOffset -= 1 in
                If TANK_OD = 138 Then dblHorizontalOffset += 1 in
                If TANK_OD = 144 Then dblHorizontalOffset += 2 in
                ' In order to locate where to put the manway assembly in our master assembly file, we will use matrix positioning
                ' See presentation included in this kit that explains how matrix positioning works - it's easier than it looks or sounds
                Dim matrixD = ThisDoc.Geometry.Matrix(-1, 0, 0, 0, 0, 1, 0, MANWAY_VERT_OFF, 0, 0, -1, dblHorizontalOffset, 0, 0, 0, 1)
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the selected manway assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixD)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the manway
                Dim componentD = Components.Add("Manway 21 Inch:1", LIBRARY_PATH & "Manways\21 Inch\21 in Manway.iam", _
                                                                position := matrixD, grounded := True, visible := True, appearance := Nothing)
        ' Next is the code to place the 22 inch manway, if that has been selected
        ElseIf MANWAY_SIZE = 22 in  Then
                If TANK_OD >= 54 in And TANK_OD <= 60 in Then dblHorizontalOffset -= 3 in
                If TANK_OD >= 66 in And TANK_OD <= 78 in Then dblHorizontalOffset -= 4 in
                If TANK_OD >= 84 in And TANK_OD <= 90 in Then dblHorizontalOffset -= 2 in
                If TANK_OD >= 96 in And TANK_OD <= 102 in Then dblHorizontalOffset -= 1 in
                If TANK_OD >= 132 in And TANK_OD <= 138 in Then dblHorizontalOffset += 2 in
                If TANK_OD = 144 Then dblHorizontalOffset += 3 in
                ' In order to locate where to put the manway assembly in our master assembly file, we will use matrix positioning
                ' See presentation included in this kit that explains how matrix positioning works - it's easier than it looks or sounds
                Dim matrixE = ThisDoc.Geometry.Matrix(0, 0, 1, 0, -1, 0, 0, MANWAY_VERT_OFF, 0, -1, 0, dblHorizontalOffset, 0, 0, 0, 1)
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the selected manway assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixE)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the manway
                Dim componentE = Components.Add("Manway 22 Inch:1", LIBRARY_PATH & "Manways\22 Inch\22 in Manway.iam", _
                                                                position := matrixE, grounded := True, visible := True, appearance := Nothing)
        Else
                If TANK_OD >= 114 in And TANK_OD <= 126 in  Then dblHorizontalOffset += 2 in
                If TANK_OD >= 132 in And TANK_OD <= 138 in   Then dblHorizontalOffset += 3 in
                If TANK_OD = 144 in Then dblHorizontalOffset += 4.5 in
                ' In order to locate where to put the manway assembly in our master assembly file, we will use matrix positioning
                ' See presentation included in this kit that explains how matrix positioning works - it's easier than it looks or sounds
                Dim matrixF = ThisDoc.Geometry.Matrix(0, 0, 1, 0, -1, 0, 0, MANWAY_VERT_OFF, 0, -1, 0, dblHorizontalOffset, 0, 0, 0, 1)
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the selected manway assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixF)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the manway
                Dim componentF = Components.Add("Manway 25 Inch:1", LIBRARY_PATH & "Manways\25 Inch\25 in Manway.iam", _
                                                                position := matrixF, grounded := True, visible := True, appearance := Nothing)

        End If
End Sub

Sub InsertHatchesIntoAssembly()
        ' This code places the hatch into our assembly (up to two times), if applicable
        ' Hatches are always place on the top of the tank body assembly, at either end of the tank
        ' This set of commands is to place the front hatch, if the user has opted to include one
        If F_HATCH Then
                ' In order to locate where to put the hatch assembly in our master assembly file, we will use matrix positioning
                ' See presentation included in this kit that explains how matrix positioning works - it's easier than it looks or sounds
                Dim matrixG = ThisDoc.Geometry.Matrix(1, 0, 0, 10.625, 0, 1, 0, TANK_OD / 2 + 3, 0, 0, 1, TANK_L / 2 - F_HATCH_OFF - 21.5, 0, 0, 0, 1)
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the hatch assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixG)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the hatch
                Dim componentG = Components.Add("Front Hatch:1", LIBRARY_PATH & "Hatches\20 Inch\MW-SW 103.iam", _
                                                                position := matrixG, grounded := True, visible := True, appearance := Nothing)
        End If

        If R_HATCH Then
                ' In order to locate where to put the hatch assembly in our master assembly file, we will use matrix positioning
                ' See presentation included in this kit that explains how matrix positioning works - it's easier than it looks or sounds
                Dim matrixH = ThisDoc.Geometry.Matrix(-1, 0, 0, -10.625, 0, 1, 0, TANK_OD / 2 + 3, 0, 0, -1, -TANK_L / 2 + R_HATCH_OFF + 21.5, 0, 0, 0, 1)
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the selected manway assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixH)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the manway
                Dim componentH = Components.Add("Rear Hatch:1", LIBRARY_PATH & "Hatches\20 Inch\MW-SW 103.iam", _
                                                                                position := matrixH, grounded := True, visible := True, appearance := Nothing)
        End If
End Sub

Sub InsertDrainNozzlesIntoAssembly()
        ' This code places the drain nozzles into our assembly, if applicable
        ' Each drain nozzle consists of a pipe, and an end connection (i.e. flange, capped flange, or valve)
        ' One drain nozzle can be placed on the front head dish plate, and another can be placed on the rear head dish plate
        ' First, we set the size of the drain nozzles based on the OD of the tank
        If TANK_OD <= 48 in Then
                DRAIN_SIZE = 3 in
        Else
                DRAIN_SIZE = 4 in
        End If

        ' If they have selected to have a drain in front, then place it at the bottom of the tank on the front dish head plate
        Dim strDrainValveName, strDrainPipeName As String
        ' These strings represent the filenames (and paths) for the both the pipe and the end connection
        strDrainValveName = LIBRARY_PATH & "Valves\Butterfly\" & DRAIN_SIZE & " Inch\Slip-On Welding to Threaded Valve - " & DRAIN_SIZE & ".iam"
        strDrainPipeName = LIBRARY_PATH & "Flanges\ANSI B36.10 XS - " & DRAIN_SIZE & ".ipt"

        ' This variable will represent the offset in the front based on the end connection type
        Dim dblFrontHorOffset As Double
        If DRAIN_F_FL_END = "Valve" Then
                dblFrontHorOffset = 12
        Else
                dblFrontHorOffset = 9
        End If

        ' This variable will represent the offset in the rear based on the end connection type
        Dim dblRearHorOffset As Double
        If DRAIN_F_FL_END = "Valve" Then
                dblRearHorOffset = 12
        Else
                dblRearHorOffset = 9
        End If

        ' This code will determine if a front drain is required, and then run code to place it if it is
        If DRAIN_F Then
                ' This uses the "GetFlangeFilename" function (near the bottom of this rule)
                ' It will automatically determine the filename based on flange type, flange end connection, and drain size
                Dim strFrontDrainFlangeName = GetFlangeFilename(DRAIN_F_FL_TYPE, DRAIN_F_FL_END, DRAIN_SIZE)
                ' This uses the "GetFrontOrRearMatrix" function (near the bottom of this rule)
                ' This will automatically determine the location matrix based on several factors
                Dim matrixI = GetFrontOrRearMatrix(DRAIN_F_FL_TYPE, DRAIN_F_FL_END, DRAIN_SIZE, dblFrontHorOffset, DRAIN_SIZE, "Front", "Bottom")
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the selected end connection part or assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixI)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the end connection
                Dim componentI = Components.Add("Front Drain:1", strFrontDrainFlangeName, position := matrixI, grounded := True, visible := True, appearance := Nothing)
                ' We create a location matrix and place the pipe, to complete the components needed for the front drain
                Dim matrixJ = ThisDoc.Geometry.Matrix(-1, 0, 0, 0, 0, 1, 0, -TANK_OD / 2 + DRAIN_SIZE, 0, 0, -1, TANK_L / 2 + (DRAIN_SIZE / (TANK_OD / 2)) * DISH_DEPTH + 9 in, 0,
0, 0, 1)
                Dim componentJ = Components.Add("Front Drain Pipe:1", strDrainPipeName, position := matrixJ, grounded := True, visible := True, appearance := Nothing)
        End If
        ' This code will determine if a rear drain is required, and then run code to place it if it is
        If DRAIN_R Then
                ' This uses the "GetFlangeFilename" function (near the bottom of this rule)
                ' It will automatically determine the filename based on flange type, flange end connection, and drain size
                Dim strRearDrainFlangeName = GetFlangeFilename(DRAIN_R_FL_TYPE, DRAIN_R_FL_END, DRAIN_SIZE)
                ' This uses the "GetFrontOrRearMatrix" function (near the bottom of this rule)
                ' This will automatically determine the location matrix based on several factors
                Dim matrixK = GetFrontOrRearMatrix(DRAIN_R_FL_TYPE, DRAIN_R_FL_END, DRAIN_SIZE, dblRearHorOffset, DRAIN_SIZE, "Rear", "Bottom")
                ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
                ' This code inserts the selected end connection part or assembly into our master tank assembly file
                ' Instead of placing at the origin, it places it based on our input matrix we created (matrixK)
                ' Note that we are grounding all geometry, and we are not using any constraints to place the end connection
                Dim componentK = Components.Add("Rear Drain:1", strRearDrainFlangeName, position := matrixK, grounded := True, visible := True, appearance := Nothing)
                ' We create a location point and place the pipe, to complete the components needed for the rear drain
                Dim pointL = ThisDoc.Geometry.Point(0, -TANK_OD / 2 + DRAIN_SIZE, -(TANK_L / 2 + (DRAIN_SIZE / (TANK_OD / 2)) * DISH_DEPTH + 9 in))
                Dim componentL = Components.Add("Rear Drain Pipe:1", strDrainPipeName, position := pointL, grounded := True, visible := True, appearance := Nothing)
        End If
End Sub

Sub CreateAndConfigureSump()
        ' This code creates a new copy of the sump assembly template and places it in our new folder structure
        ' It then updates the sump geometry based on values we pass to the assembly
        Dim strNewSumpFilename As String
        ' This string represents the new name of our unique, copied sump assembly file
        strNewSumpFilename = PROJECT_PATH & PROJECT_ID & "\Sump Assy\Sump Pipe Assy - " & PROJECT_ID & ".iam"

        ' We first check to make sure the sump assembly file has not been previously created
        ' If it hasn't yet been created, we do a Windows Copy operation to make a new copy in our new folder
        ' We also make copies of the part files that will go into our sump assembly
        If System.IO.File.Exists(strNewSumpFilename) = False Then
                System.IO.File.Copy(TEMPLATE_PATH & "Sump Assy\Sump Pipe Assy.iam", PROJECT_PATH & PROJECT_ID & "\Sump Assy\Sump Pipe Assy - " & PROJECT_ID & ".iam")
                System.IO.File.Copy(TEMPLATE_PATH & "Sump Assy\Sump-Angled Pipe.ipt", PROJECT_PATH & PROJECT_ID & "\Sump Assy\Sump-Angled Pipe - " & PROJECT_ID & ".ipt")
                System.IO.File.Copy(TEMPLATE_PATH & "Sump Assy\Sump-Straight Pipe.ipt", PROJECT_PATH & PROJECT_ID & "\Sump Assy\Sump-Straight Pipe - " & PROJECT_ID & ".ipt")
                ' This here is a sample of how to make a variable that represents an assembly document
                ' It uses the Inventor API, which you can use freely (for the most part) throughout iLogic rules
                Dim oSumpAssy As Inventor.AssemblyDocument
                ' This statement tells oSumpAssy to represent the newly created file, and opens it up in the Inventor interface
                oSumpAssy = ThisApplication.Documents.Open(strNewSumpFilename, True)
                ' When the copied sump assembly initially opens, it will reference the old part files in our template folder
                ' We need to change that so that the newly copied angle and straight pipe files are referenced by the assembly
                ' The following code uses the Inventor API functionality to do that
                ' This is similar to selecting the "Replace Components" command in the Inventor application
                Dim strOldAnglePipe, strNewAnglePipe As String
                strOldAnglePipe = TEMPLATE_PATH & "Sump Assy\Sump-Angled Pipe.ipt"
                strNewAnglePipe = PROJECT_PATH & PROJECT_ID & "\Sump Assy\Sump-Angled Pipe - " & PROJECT_ID & ".ipt"
                oSumpAssy.File.ReferencedFileDescriptors.Item(strOldAnglePipe).ReplaceReference(strNewAnglePipe)
                Dim strOldStraightPipe, strNewStraightPipe As String
                strOldStraightPipe = TEMPLATE_PATH & "Sump Assy\Sump-Straight Pipe.ipt"
                strNewStraightPipe = PROJECT_PATH & PROJECT_ID & "\Sump Assy\Sump-Straight Pipe - " & PROJECT_ID & ".ipt"
                oSumpAssy.File.ReferencedFileDescriptors.Item(strOldStraightPipe).ReplaceReference(strNewStraightPipe)
                ' Once we've updated the file references in the sump assembly file, we can save and then close it
                oSumpAssy.Save
                oSumpAssy.Close
        End If

        ' The sump assembly was created in the exact same orientation as our master tank assembly
        ' This means we don't need to rotate the sump assembly when placing it into the master tank assembly
        ' That means we don't need a matrix, but can just define a point (X, Y, Z coordinates) of where to place it
        Dim pointO = ThisDoc.Geometry.Point(0, -TANK_OD / 2 + SUMP_H, TANK_L / 2)
        ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
        ' This code inserts our newly created sump pipe assembly into our master tank assembly file
        ' Instead of placing at the origin, it places it based on our input point we created (pointO)
        ' Note that we are grounding all geometry, and we are not using any constraints to place the assembly
        Dim componentO = Components.Add("Sump Pipe Assembly:1", strNewSumpFilename, position := pointO, grounded := True, visible := True, appearance := Nothing)
        ' These statements pass parameters from our master assembly file into the sump pipe assembly file
        Parameter("Sump Pipe Assembly:1", "PROJECT_ID") = PROJECT_ID
        Parameter("Sump Pipe Assembly:1", "PROJECT_PATH") = PROJECT_PATH
        Parameter("Sump Pipe Assembly:1", "SUMP_SIZE") = SUMP_SIZE
        Parameter("Sump Pipe Assembly:1", "SUMP_H") = SUMP_H
        Parameter("Sump Pipe Assembly:1", "SUMP_PIPE_PROJ") = SUMP_PIPE_PROJ
        Parameter("Sump Pipe Assembly:1", "TANK_OD") = TANK_OD
        Parameter("Sump Pipe Assembly:1", "DISH_DEPTH") = DISH_DEPTH
        ' Once all the parameters are updated in the sump pipe assembly file, we want to run its update rule
        ' This will allow the sump pipe assembly to update all its own parts and components itself
        iLogicVb.RunRule("Sump Pipe Assembly:1", "Update Children Parts")
        ' Once the sump pipe assembly is created and placed, it still needs an end connection
        ' This uses the "GetFlangeFilename" function (near the bottom of this rule)
        ' It will automatically determine the filename based on flange type, flange end connection, and drain size
        Dim strFlangeName As String = GetFlangeFilename(SUMP_FL_TYPE, SUMP_FL_END, SUMP_SIZE)
        ' This uses the "GetFrontOrRearMatrix" function (near the bottom of this rule)
        ' This will automatically determine the location matrix based on several factors
        Dim matrixP = GetFrontOrRearMatrix(SUMP_FL_TYPE, SUMP_FL_END, SUMP_H, SUMP_PIPE_PROJ, SUMP_SIZE, "Front", "Bottom")
        ' This can be taken from an iLogic snippet, and is used to insert components into assemblies
        ' This code inserts the selected end connection part or assembly into our master tank assembly file
        ' Instead of placing at the origin, it places it based on our input matrix we created (matrixP)
        ' Note that we are grounding all geometry, and we are not using any constraints to place the end connection
        Dim componentP = Components.Add("Sump Valve:1", strFlangeName, position := matrixP, grounded := True, visible := True, appearance := Nothing)
End Sub

Sub InsertInletIntoAssembly()
        ' This code places the inlet nozzle into our assembly
        ' The inlet nozzle consists of a pipe and an end connection (i.e. flange, capped flange, or valve)
        ' The user has the option to place the inlet nozzle on the top of the tank, or the front dish head plate
        ' If they place it on the dish head plate, it must be located near the top of the tank, and not the bottom half
        Dim strInletTubeName, strFlangeFile As String

        ' No new geometry is created for inlets - they only use existing parts from the library
        ' This let's us find the right name of the tube (or pipe) based on the inlet size
        strInletTubeName = LIBRARY_PATH & "Flanges\ANSI B36.10 XS - " & INLET_SIZE & ".ipt"

        ' This code uses our "GetFlangeFilename" function to find the name of the end connection based on
        '     flange type, flange end connection, and inlet size
        strFlangeFile = GetFlangeFilename(INLET_FL_TYPE, INLET_FL_END, INLET_SIZE)

        ' Define the matrices that will be needed to place the inlet nozzle, including the pipe and flange
        Dim matrixM, matrixN As DocumentUnitsMatrix
        Dim strInletPipeBrowserName, strInletFlangeBrowserName As String
        ' If the user wants to place the inlet nozzle on the top, use these locating matrices
        If INLET_LOC = "Top" Then
                ' These strings will be used to set the occurrence names in the browser to indicate they are installed on top of the tank
                strInletPipeBrowserName = "Top Inlet Pipe - " & INLET_SIZE & " Inch:1"
                strInletFlangeBrowserName = "Top Inlet Flange - " & INLET_SIZE & " Inch:1"
                ' This matrix represents the orientation required for the pipe on top of the tank
                matrixM = ThisDoc.Geometry.Matrix(1, 0, 0, 0, 0, 0, -1, TANK_OD / 2 + 6, 0, 1, 0, TANK_L / 2 - INLET_OFF, 0, 0, 0, 1)
                ' The locating matrix will be different for open, capped and valve end connection choices
                If INLET_FL_END = "Open" Then
                        ' If the user chooses a welding neck flange, a different offset matrix value will be required for the Y (up) direction
                        If INLET_FL_TYPE = "Welding Neck" Then
                                matrixN = ThisDoc.Geometry.Matrix(0, 1, 0, 0, -1, 0, 0, TANK_OD / 2 + dblFlangeOffsetDistance + 9 in, 0, 0, 1, TANK_L / 2 - INLET_OFF, 0, 0, 0, 1)
                        Else
                                matrixN = ThisDoc.Geometry.Matrix(0, 1, 0, 0, -1, 0, 0, TANK_OD / 2 + dblFlangeOffsetDistance + 6 in, 0, 0, 1, TANK_L / 2 - INLET_OFF, 0, 0, 0, 1)
                        End If
                ElseIf INLET_FL_END = "Capped" Then
                        matrixN = ThisDoc.Geometry.Matrix(1, 0, 0, 0, 0, 0, 1, TANK_OD / 2 + 6, 0, -1, 0, TANK_L / 2 - INLET_OFF, 0, 0, 0, 1)
                Else
                        matrixN = ThisDoc.Geometry.Matrix(1, 0, 0, 0, 0, 0, 1, TANK_OD / 2 + dblFlangeOffsetDistance + 7 in, 0, -1, 0, TANK_L / 2 - INLET_OFF, 0, 0, 0, 1)
                End If
        ' If the user wants to place the inlet nozzle on the front, this is the code that will be used to create the location matrices
        Else
                ' These strings will be used to set the occurrence names in the browser to indicate they are installed on top of the tank
                strInletPipeBrowserName = "Front Inlet Pipe - " & INLET_SIZE & " Inch:1"
                strInletFlangeBrowserName = "Front Inlet Flange" & INLET_SIZE & " Inch:1"
                Dim dblDishOffset As Double = TANK_L / 2 + (INLET_OFF / (TANK_OD / 2)) * DISH_DEPTH + 6
                matrixM = ThisDoc.Geometry.Matrix(-1, 0, 0, 0, 0, 1, 0, TANK_OD / 2 - INLET_OFF, 0, 0, -1, dblDishOffset + 4, 0, 0, 0, 1)
                ' Since we created a function (GetFrontOrRearMatrix) that figures out location matrices on the front and rear dish plates,
                '     we can take advantage of that and don't need to figure them out separately, like we had to for the top
                matrixN = GetFrontOrRearMatrix(INLET_FL_TYPE, INLET_FL_END, INLET_OFF, 10, INLET_SIZE, "Front", "Top")
        End If
        ' These are the iLogic commands to add the pipe and flange components to the assembly, and place them properly based on the matrices
        Dim componentM = Components.Add(strInletPipeBrowserName, strInletTubeName, position := matrixM, grounded := True, visible := True, appearance := Nothing)
        Dim componentN = Components.Add(strInletFlangeBrowserName, strFlangeFile, position := matrixN, grounded := True, visible := True, appearance := Nothing)
End Sub

Function GetFlangeFilename(strFlangeType As String, strFlangeEnd As String, dblSize As Double) As String
        ' This function determines the full path and filename of the end connection that is needed, based on the flange type,
        '     flange end connection, and size
        Dim strFilename As String

        ' If the end connection is "Open", then we just return a flange part
        If strFlangeEnd = "Open" Then
                strFilename = LIBRARY_PATH & "Flanges\ASME B16.5 Flange " & strFlangeType & " - Class 150 " & dblSize & ".ipt"
        ' If the end connection is "Capped", then we find which pre-created assembly includes the desired flange and cap
        ' The files in the library were setup with a consistent naming convention so that it was easy to derive the filenames
        '     based on this information
        ElseIf strFlangeEnd = "Capped" Then
                strFilename = LIBRARY_PATH & "Flanges\" & strFlangeType & " to Blind - " & dblSize & ".iam"
        ' If the end connection is "Valve", then we find which pre-created assembly includes the desired flange and butterfly valve
        ' The files in the library were setup with a consistent naming convention so that it was easy to derive the filenames
        '     based on this information
        Else
                strFilename = LIBRARY_PATH & "Valves\Butterfly\" & dblSize & " Inch\" & strFlangeType & " to Threaded Valve - " & dblSize & ".iam"
        End If

        ' Set our resulting filename string to the GetFlangeFilename function so that it can be returned to our calling statement
        GetFlangeFilename = strFilename
End Function

Function GetFrontOrRearMatrix(strFlangeType As String, strFlangeEnd As String, dblVertOffset As Double, dblCustomHorOffset As Double, _
                                                            dblFlangeSize As Double, strSide As String, strTopOrBottom As String) As DocumentUnitsMatrix
        ' This function returns a matrix object that is derived based on all of its inputs
        ' It is only good for matrices on the front dish head plate, and the rear dish head plate, and only for end connections
```

```
	' That includes flanges, caps and valves
	Dim matrixReturn As DocumentUnitsMatrix
	' This variable calculates the length from the center of the tank to the outside edge of the tank body
	' It then approximates the dish head plate depth using a linear formula (which isn't always the most accurate)
	' The goal is to get the distance as from tank centerline to the outside edge of the tank, including the dish head plate
	Dim dblDishOffset As Double = TANK_L / 2 + (dblVertOffset / (TANK_OD / 2)) * DISH_DEPTH
	' This uses the "GetFlangeOffsetDistance" function to get the initial offset values based on the type of end connection
	Dim dblFlangeOffset As Double = GetFlangeOffsetDistance(strFlangeType, strFlangeEnd, dblFlangeSize)
	Dim dblYValue, dblZValue As Double
	' We need to know if the end connection will be on the upper half of the tank, or the lower half of the tank
	' If it's on the upper half, our Y location value will be positive
	' If it's on the lower half, our Y location value will be negative
	If strTopOrBottom = "Top" Then
		dblYValue = TANK_OD / 2 - dblVertOffset
	Else
		dblYValue = -TANK_OD / 2 + dblVertOffset
	End If
	' For "Open" end connections, calculate our Z location value, and create one matrix for the front, and one for the rear
	' The reason front and rear placement matrices differ, is that a flange has to be rotated 180-degrees if it's placed
	'     on the rear dish head; in other words, you always want the flanges pointing away from the tanks
	If strFlangeEnd = "Open" Then
		dblZValue = dblDishOffset + dblFlangeOffset + dblCustomHorOffset - 6 in
		If strSide = "Front" Then
			matrixReturn = ThisDoc.Geometry.Matrix(0, 0, 1, 0, 0, 1, 0, dblYValue, -1, 0, 0, dblZValue, 0, 0, 0, 1)
		Else
			matrixReturn = ThisDoc.Geometry.Matrix(0, 0, -1, 0, 0, 1, 0, dblYValue, 1, 0, 0, -dblZValue, 0, 0, 0, 1)
		End If
	' For "Capped" end connections, calculate our Z location value, and create one matrix for the front, and one for the rear
	ElseIf strFlangeEnd = "Capped" Then
		dblZValue = dblDishOffset + dblCustomHorOffset
		If strSide = "Front" Then
			matrixReturn = ThisDoc.Geometry.Matrix(1, 0, 0, 0, 0, 1, 0, dblYValue, 0, 0, 1, dblZValue, 0, 0, 0, 1)
		Else
			matrixReturn = ThisDoc.Geometry.Matrix(-1, 0, 0, 0, 0, 1, 0, dblYValue, 0, 0, -1, -dblZValue, 0, 0, 0, 1)
		End If
	' For "Valve" end connections, calculate our Z location value, and create one matrix for the front, and one for the rear
	Else
		dblZValue = dblDishOffset + dblCustomHorOffset + 1 in
		If strSide = "Front" Then
			matrixReturn = ThisDoc.Geometry.Matrix(1, 0, 0, 0, 0, 1, 0, dblYValue, 0, 0, 1, dblZValue, 0, 0, 0, 1)
		Else
			matrixReturn = ThisDoc.Geometry.Matrix(-1, 0, 0, 0, 0, 1, 0, dblYValue, 0, 0, -1, -dblZValue, 0, 0, 0, 1)
		End If
	End If
	' Set our resulting matrix to the GetFrontOrRearMatrix function so that it can be returned to our calling statement
	GetFrontOrRearMatrix = matrixReturn
End Function

Function GetFlangeOffsetDistance(strFlangeType As String, strFlangeEnd As String, dblFlangeSize As Double) As Double
	' This function determines what the initial flange offset distance should be for any end connection based on its
	'     flange type, flange end connection, and size
	' It's pretty straight forward and just assigns empirically derived offset values based on the type of end connection
	Dim dblFlangeOffsetDistance As Double
	If strFlangeType = "Welding Neck" Then
		If dblFlangeSize = 3 in Then dblFlangeOffsetDistance = 8.5 in
		If dblFlangeSize = 4 in  Then dblFlangeOffsetDistance = 9 in
		If dblFlangeSize = 6 in Then dblFlangeOffsetDistance = 9.5 in
```

```
                If dblFlangeSize = 8 in Then dblFlangeOffsetDistance = 10 in
        Else
                If dblFlangeSize = 3 in Then dblFlangeOffsetDistance = 6.25 in
                If dblFlangeSize = 4 in Then dblFlangeOffsetDistance = 6.31 in
                If dblFlangeSize = 6 in Then dblFlangeOffsetDistance = 6.56 in
                If dblFlangeSize = 8 in Then dblFlangeOffsetDistance = 6.5 in
        End If
        If strFlangeEnd = "Valve" Then dblFlangeOffsetDistance = dblFlangeOffsetDistance + 1 in
        GetFlangeOffsetDistance = dblFlangeOffsetDistance
End Function
```