

Create a Windows Form, Display It, Get Instant Feedback From Your Choices, all from one iLogic Rule

Article By: Wesley Carihfield

Originally posted to <https://knowledge.autodesk.com/community/> site on: October 15, 2020

<https://knowledge.autodesk.com/support/inventor/learn-explore/caas/simplecontent/content/create-windows-form-display-it-get-instant-feedback-your-choices-all-one-ilogic-rule.html>

Many of us have likely struggled with the design and functionality of both the iLogic Form and the VBA UserForm, over the years. This goal of this post is to show you some basics about how to bypass both of those two form design options for a third option. Creating your own custom Windows Form, from scratch, using only code, within an iLogic rule.

Both the iLogic form and the VBA UserForm are based on the Windows form and expose their own specific means by which to design them and specify their settings and functionality. The iLogic form is usually the quickest and easiest to learn how to create and use but lacks the flexibility (and some capability) of the VBA UserForm. The VBA UserForm has more options and capability, but due to the complexity of the process, often takes a long while to learn, and the design is in two different places (a graphical design, and "form module" for specifying the content and functionality of the form. Because of these complications/restrictions/functionality issues, creating a Windows form entirely by code, within the same rule you're planning on interacting with it from, can sometimes seem like a "breath of fresh air".

Now to the point. Here is some code you can copy and paste into an iLogic rule, that when ran, will create and display a custom Windows form. Almost all the "controls" within this form will provide INSTANT feedback when you change something. This behavior is all set-up and customizable/changeable within the code, using "event handlers". I've kept those event handler codes simple, just to show how they work. Each "control" (TextBox, CheckBox, ComboBox, Button, RadioButton, etc.) has its own set of "Events" that you can play with. I have one, concerning the TextBox commented out (in two places), because, as my comments within the code state, the event handler it creates can be extremely annoying.

Here's the code: (Don't forget to put those top Reference and Import options into the Header of your rule {or just at the very top}.)

```
AddReference "System.Drawing"
Imports System.ComponentModel
Imports System.Drawing
Imports System.Windows.Forms
Public Class WinForm
    Inherits System.Windows.Forms.Form
    'declare any thing here that you want to use/access throughout all Subs & Functions
    Public oLargerFont As System.Drawing.Font = New Font("Arial",10)
    Public Sub New() 'creates the new instance
        oForm = Me
        With oForm
            .FormBorderStyle = FormBorderStyle.FixedToolWindow
            .StartPosition = FormStartPosition.CenterScreen
            .Width = 300
            .Height = 300
            .TopMost = True
            .Font = oLargerFont
            .Text = "Windows Form"
            .Name = "Windows Form"
            .ShowInTaskbar = False
        End With

        Dim oButton1 As New Button()
        With oButton1
            .Text = "TEST ME"
```

```

        .Top = 25
        .Left = 25
        .Enabled = True
        .AutoSize = True
    End With
    oForm.Controls.Add(oButton1)
    AddHandler oButton1.Click, AddressOf oButton1_Click

```

```

Dim oTextBox As New TextBox()
With oTextBox
    .Text = "Your interactive text here."
    .Top = oButton1.Bottom + 10
    .Left = 25
    .Width = 250
    .Height = 25
End With

```

'once the focus is within the TextBox, this event lets you know when the focus leaves it again
 'this is set to trigger the Sub below which checks the contents of the TextBox
 'to see if anything has changed from the default value.
 AddHandler oTextBox.Leave, AddressOf oTextBox_Leave

```
oForm.Controls.Add(oTextBox)
```

```

Dim oCheckBox As New CheckBox()
With oCheckBox
    .Text = "Activate Option/Setting?"
    .Top = oTextBox.Bottom + 10
    .Left = 25
    .AutoSize = True
End With
AddHandler oCheckBox.CheckedChanged, AddressOf oCheckBox_CheckedChanged
oForm.Controls.Add(oCheckBox)

```

```

Dim oOptions As New List(Of String)
oOptions.AddRange({"Option 1", "Option 2", "Option 3"})

```

```

Dim oComboBox As New ComboBox()
With oComboBox
    .DropDownStyle = ComboBoxStyle.DropDownList
    .Items.Clear()
    .Items.AddRange(oOptions.ToArray)
    .SelectedIndex = 1
    .Top = oCheckBox.Bottom + 10
    .Left = 25
    .Width = 150
End With
AddHandler oComboBox.SelectionChangeCommitted, AddressOf oComboBox_SelectionChangeCommitted
oForm.Controls.Add(oComboBox)

```

```

Dim oGroupBox As GroupBox = New GroupBox()
oGroupBox.Left = 25
oGroupBox.Width = 150
oGroupBox.Top = oComboBox.Bottom + 10
oGroupBox.Height = 100
oGroupBox.Text = "Choose"
oForm.Controls.Add(oGroupBox)

```

'These radio buttons can also just be added to the form directly without being in a GroupBox
 Dim oRadioButton1 As New RadioButton()
 AddHandler oRadioButton1.CheckedChanged, AddressOf oRadioButton1_CheckedChanged
 oGroupBox.Controls.Add(oRadioButton1)
 'oRadioButton1.Checked = True 'you can use this to set the default
 'be careful though, because it will trigger the Sub below if uncommented

```

oRadioButton1.Text = "Choice 1"
'>>> This location is in reference to the bounds of the GroupBox, not the Form <<<
oRadioButton1.Top = 20 'oComboBox.Top + 20
oRadioButton1.Left = 35
oRadioButton1.AutoSize = True

```

```

Dim oRadioButton2 As New RadioButton()
AddHandler oRadioButton2.CheckedChanged, AddressOf oRadioButton2_CheckedChanged
oGroupBox.Controls.Add(oRadioButton2)
oRadioButton2.Text = "Choice 2"
'>>> This location is in reference to the bounds of the GroupBox, not the Form <<<
oRadioButton2.Top = oRadioButton1.Bottom + 10
oRadioButton2.Left = 35
oRadioButton2.AutoSize = True

```

```

'This is the end of the main Sub that defines the Form
End Sub

```

```

Private Sub WinForm_FormClosing(ByVal oSender As Object, ByVal oFormCloseEvents As FormClosingEventArgs)
Handles Me.FormClosing
    If MsgBox("Are you sure you want to close this Form?",vbYesNo+vbQuestion, "CLOSE") = vbYes Then
    Else
        oFormCloseEvents.Cancel = True
    End If
End Sub

```

```

Private Sub oButton1_Click(ByVal oSender As System.Object, ByVal oEventArgs As System.EventArgs)
'oSender is a Button
MsgBox("You just clicked the [" & oSender.Text & "] button.",vbOKOnly+vbInformation,"EVENT FEEDBACK")
End Sub

```

```

Private Sub oTextBox_Leave(ByVal oSender As System.Object, ByVal oEventArgs As System.EventArgs)
If oSender.Text <> oOriginalText Then 'oSender is a TextBox
    MsgBox("The contents of the TextBox have changed.", vbOKOnly + vbInformation, "EVENT
FEEDBACK")
End If
End Sub

```

```

Private Sub oCheckBox_CheckedChanged(ByVal oSender As System.Object, ByVal oEventArgs As System.EventArgs)
If oSender.Checked = True Then 'Sender is a CheckBox
    MsgBox("You just Checked the [" & oSender.Text & "] checkbox.", vbOKOnly + vbInformation, "EVENT
FEEDBACK")
Else
    MsgBox("You just UnChecked the [" & oSender.Text & "] checkbox.", vbOKOnly + vbInformation, "EVENT
FEEDBACK")
End If
End Sub

```

```

Private Sub oComboBox_SelectionChangeCommitted(ByVal oSender As System.Object, ByVal oEventArgs As
System.EventArgs)
'oSender is a ComboBox
MsgBox("You just chose " & oSender.Text & " from the ComboBox.",vbOKOnly+vbInformation,"EVENT
FEEDBACK")
End Sub

```

```

Private Sub oRadioButton1_CheckedChanged(ByVal oSender As System.Object, ByVal oEventArgs As
System.EventArgs)
If oSender.Checked Then 'oSender is a RadioButton
    MsgBox("You just changed the Radio Button option to '" & oSender.Text & "'.", vbOKOnly +
vbInformation, "EVENT FEEDBACK")
Else

```

```

        MsgBox("You just changed the Radio Button option to 'Choice 2'.", vbOKOnly + vbInformation, "EVENT
FEEDBACK")
    End If
End Sub

' Private Sub oRadioButton2_CheckedChanged(ByVal oSender As System.Object, ByVal oEventArgs As
System.EventArgs)
'     MsgBox("You just changed the Radio Button option to " & oSender.Text & ".", vbOKOnly+vbInformation,"EVENT
FEEDBACK")
' End Sub
End Class

```

```

'This is the code that actually shows/runs the Form
Public Class RunMyForm
    Private Sub Main
        Dim oMyForm As New WinForm
        oMyForm.Show
    End Sub
End Class

```

This Windows Form includes a Button, a TextBox, a CheckBox, a ComboBox, a GroupBox, and two RadioButtons. The two radio buttons are just placed into the inside of a GroupBox to show how a GroupBox acts. The radio buttons can just be added directly to the form, without being inside of a GroupBox. When specifying the locations of controls within a GroupBox, those locations are going to be relative to the bounds of the GroupBox, instead of the bounds of the form itself. If you want to delete the GroupBox, you may then have to also edit the locations (Top & Left values) of those two RadioButtons also.