Honeywell

Lab Exercise – Add Remaining Enhancements

57311007L
06/00

TotalPlant

Notices and Trademarks

This module supports **TotalPlant** Solution (TPS) system network.

TPS is the evolution of TDC 3000X.

Honeywell Inc.
Industrial Automation and Control
Automation College
2500 W. Union Hills Drive
Phoenix, AZ 85027
1-800 852-3211

Lab Exercise 7

Introduction

In the final lab exercise for this section , you will enhance your popup dialog and add the following functionality:

- A confirmation dialog that appears within the popup dialog whenever the operator selects a state change button.
- An error message that appears if there is an OnDataChange script runtime error.
- A Loop ID parameter that causes any related points to appear highlighted. This allows an operator to see what other points are in the same control strategy as the selected point. For example, all points related to a particular piece of equipment may highlight when one is selected.
- Feedback text that indicates to the operator that a requested state change is occurring.
- Fillcolor dynamics for the state change buttons.
- Two state and three state button dynamics. If the digital composite is a 2 state point, then the third button disappears and the button symmetry is adjusted.

Objectives

At the end of the lab exercise, you will be able to do the following:

- Create a confirmation dialog.
- Use a text object for error messages.
- Highlight related control points.
- Use a text object for state change feedback.
- Add fillcolor dynamics to the state change buttons.
- Update the dialog so that two state and three state buttons only appear when their respective points are invoked.

Design Criteria – Digital Target

The changes to the digital target are minor:

- You will add a display parameter that represents a Loop Id for related control points.
- Modify the OnLButtonUp and OnDataChange scripts that you coded earlier for the digital target so that you can highlight related control loop points. The selected point will have a gray highlighted target, while other points in the same control loop strategy will have a white highlighted target(s).

In the lab exercise, the display parameter that you will define is:

- **Loop_ID** – this parameter is defined as an integer data type and writes its value to dispdb.int01. Other targets in the same control strategy compare their Loop Ids and highlight if they have the same Loop ID number. When Loop_ID is set to –1 from script, the highlighted target(s) become invisible.

The Loop_ID script that you add to the digital target's OnLButtonScript is shown in the following **boldfaced** statement:

```
Sub OnLButtonUp()      'touchscreen and mouse click event
       Tag.external = pname      'send tagname string to dispdb.ent

       'send display.params to popup dialog
       display.params.obj.params.PName = pname
       display.params.obj.params.State0 = State0 'Button2
       display.params.obj.params.State1 = State1 'Button1
       display.params.obj.params.State2 = State2 'Button3
       display.params.obj.params.PtDesc = PtDesc
       display.params.obj.params.PntType = PntType
       display.params.obj.params.NoStates = NoStates
       display.params.obj.params.x = display.params.x
       display.params.obj.params.y = display.params.y
       display.params.obj.params.Info_File = display.params.Info

            'other targets check dispb.str01
       me.visible = true 'target is selected
       dispdb.int01 = display.params.Loop_ID
       dispdb.str01 = My_name

End Sub
```

The Loop_ID script that you add to the target's OnDataChange procedure is shown in the following **boldfaced** statements:

```
Sub OnDataChange()
       On Error Goto ODC_Error

       dim LP as integer
       LP = dispdb.int01

       Select Case LP
              Case display.params.Loop_ID 'related point
                     me.fillcolor = makecolor(255,255,255) 'white target
                     me.visible = true
              Case Else
                     me.visible = false
       End Select


       'collector uses inline called point
       ALM = collector("ackstat(\pe(point))")
       If My_Name = dispdb.str01 Then
              me.fillcolor = makecolor(200,200,200)
              me.visible = true
              display.params.obj.params.Info_Color = display.params.Info_Color
              display.params.obj.params.Alarm = ALM   'send status to dialog
       'Else --comment this line and next line, you do not need these in lab
              'me.visible = false --Loop_ID makes target visible/invisible
       End If

'only need to send these values once
       If not once then
              My_Name = display.name
              once = true
              pname = point.[name]
              State0 = point.State0
              State1 = point.State1
              PtDesc = point.ptdesc
              PntType = point.pntType
              NoStates = point.NoStates
              If NoStates = 3 Then State2 = point.State2
       End If
              Exit Sub
ODC_Error:
              End Sub
```

Design Criteria – Popup Dialog

Changes to your current popup dialog's OnDataChange script support the following enhancements:

- Use a text object for error messages.
- Create a confirmation dialog for a state change request.
- Provide operator feedback by making the button fillcolors change to the requested commanded state and give feedback text regarding the digital input status.

To update the popup dialog for the enhancements means that you have to modify the OnDataChange script you coded earlier with script shown in the following **boldfaced** statements.

A text object is used for error messages in two of your OnDataChange scripts. The reason you use a text object instead of a Message Box, MsgBox, is that a text object is non-modal. In other words, you want the OnDataChange script to display an error but continue to execute. An example code fragment is shown below:

```
'make text object visible with Error description
            E_Msg.visible = true
            E_Msg.text = "Error " & ERR & "-" & Error$
```

Because there are three state change buttons, the following statements add the confirmation dialog through the use of a function call rather than hard-coding the dialog for each button. As you observed from the previous lab exercise, the use of functions saves you scripting effort and eases the management of your code.

One challenging aspect of the confirmation dialog is that you want it to appear within the popup dialog. As you have already observed, the popup dialog moves next to your process element based upon the x and y coordinates you entered as display parameters. This means that you have to use the same x and y coordinates to position the dialog within the popup dialog. The confirmation dialog itself is sized to fit within the popup. An example code fragment is shown below:

```
'confirmation dialog function for all state changes
Function Confirm(s as string) as boolean
      dx = .67 * display.params.x 'position inside popup
      dy = .63 * display.params.y 'position inside popup

      Begin Dialog UserDialog dx,dy,102,36,s
            PushButton 56,4,44,24,"NO",.PushButton2
            PushButton 4,4,44,24,"YES",.PushButton1
      End Dialog
```

```
      dim d as UserDialog
      If dialog(d) = 2 Then
            Confirm = true
      Else
            Confirm = false
      End If
End Function
```

The BUTTON#_LClick() procedures will be coded to call the confirm function. If the operator selects YES, a boolean value of true is returned from the function and the output command is sent to the LCN point's output (op) parameter. If the operator selects NO, a boolean value of false is returned from the function and no output command is sent to the LCN point. An example code fragment is shown below for a BUTTON#_LClick() procedure:

```
'procedure for State1 button click
Sub BUTTON1_LClick()
      On Error Goto Button_error
      dim st as string
      st = display.params.State1
      'call confirm function
      If confirm(st) Then display.params.Tag.op = st
      Exit Sub
      Button_error:
            INFO.fillcolor = tdc_red
            MsgBox "Error '" & Err & " - " & Error$ & "'"
End Sub
```

Your code for the EXIT button will have the following **boldfaced** statements added and appear as the following:

```
Sub OnDataChange()
       On Error Goto ODC_Error
       dim almstat as String ‘point alarm status
       dim pnttype as string ‘digital point type
       dim clr as long ‘info button color

       pnttype = display.params.Pnttype

       If (pnttype = "DEVCTL" or pnttype = "DIGCOM" or _
              pnttype = "DIGOUT") and dispdb.int01 <> -1 Then
              call Show(true) ‘position and make popup visible
       Else
              call Show(false) ‘make popup invisible
              Exit Sub
       End If

       almstat = display.params.Alarm
       ‘check alarm status
       if almstat = "NOALARM" Then
              ALM.visible = false
              ALM.blink = false
       elseif almstat = "UNAKALRM" Then
              ALM.visible = true
              ALM.blink = true
       elseif almstat = "AKDALRM" Then
              ALM.blink = false
              ALM.visible = true
       end if


       clr = display.params.Info_color
       If clr = 0 Then
              INFO.fillcolor = makecolor(204,204,204)
       Else
              INFO.fillcolor = display.params.Info_color
       End If
Exit Sub

       ODC_Error:
       ‘make text object visible with Error description
              E_Msg.visible = true
              E_Msg.text = “Error “ & ERR & “-“ & Error$
End Sub

‘procedure for INFO button click
Sub INFO_LClick()
       On Error Goto Info_Error

       INFO.fillcolor = makecolor(204,204,204)
       If FileExists(display.params.Info_File) Then
              dispdb.ent01G.external = display.params.Tag.[name]
              WinSize 8500,8500
              InvokeDisplay(display.params.Info_File)
       Else
              detail display.params.Tag.[name]
```

```
      End If
Exit Sub

      Info_Error:
      'make text object visible with Error description
            E_Msg.visible = true
            E_Msg.text = "Error " & ERR & "-" & Error$
End Sub

'confirmation dialog function for all state changes
Function Confirm(s as string) as boolean
      dx = .67 * display.params.x 'position inside popup
      dy = .63 * display.params.y 'position inside popup

      Begin Dialog UserDialog dx,dy,102,36,s
            PushButton 56,4,44,24,"NO",.PushButton2
            PushButton 4,4,44,24,"YES",.PushButton1
      End Dialog

      dim d as UserDialog
      If dialog(d) = 2 Then
            Confirm = true
      Else
            Confirm = false
      End If
End Function


'procedure for State1 button click
Sub BUTTON1_LClick()
      On Error Goto Button_error
      dim st as string
      st = display.params.State1
      'call confirm function
      If confirm(st) Then display.params.Tag.op = st
      Exit Sub
      Button_error:
            INFO.fillcolor = tdc_red
            MsgBox "Error '" & Err & " - " & Error$ & "'"
End Sub

'procedure for State0 button click
Sub BUTTON2_LClick()
      On Error Goto Button_error
      dim st as string
      st = display.params.State0
      'call confirm function
      If confirm(st) Then display.params.Tag.op = st
      Exit Sub
      Button_error:
            INFO.fillcolor = tdc_red
            MsgBox "Error '" & Err & " - " & Error$ & "'"
End Sub

'procedure for State2 button click
Sub BUTTON3_LClick()
      On Error Goto Button_error
```

```
       dim st as string
       st = display.params.State2
       'call confirm function
       If confirm(st) Then display.params.Tag.op = st
       Exit Sub
       Button_error:
              INFO.fillcolor = tdc_red
              MsgBox "Error '" & Err & " - " & Error$ & "'"
End Sub

'procedure for EXIT button
Sub OnLButtonUp()
       INFO.fillcolor = makecolor(204,204,204)
       Dispdb.int01 = -1 'clear Loop_ID targets
       Dispdb.str01 = "" '  clears the pname from the dialog
       E_Msg.visible = false 'clear err msg text
       Show(false)

End Sub
```

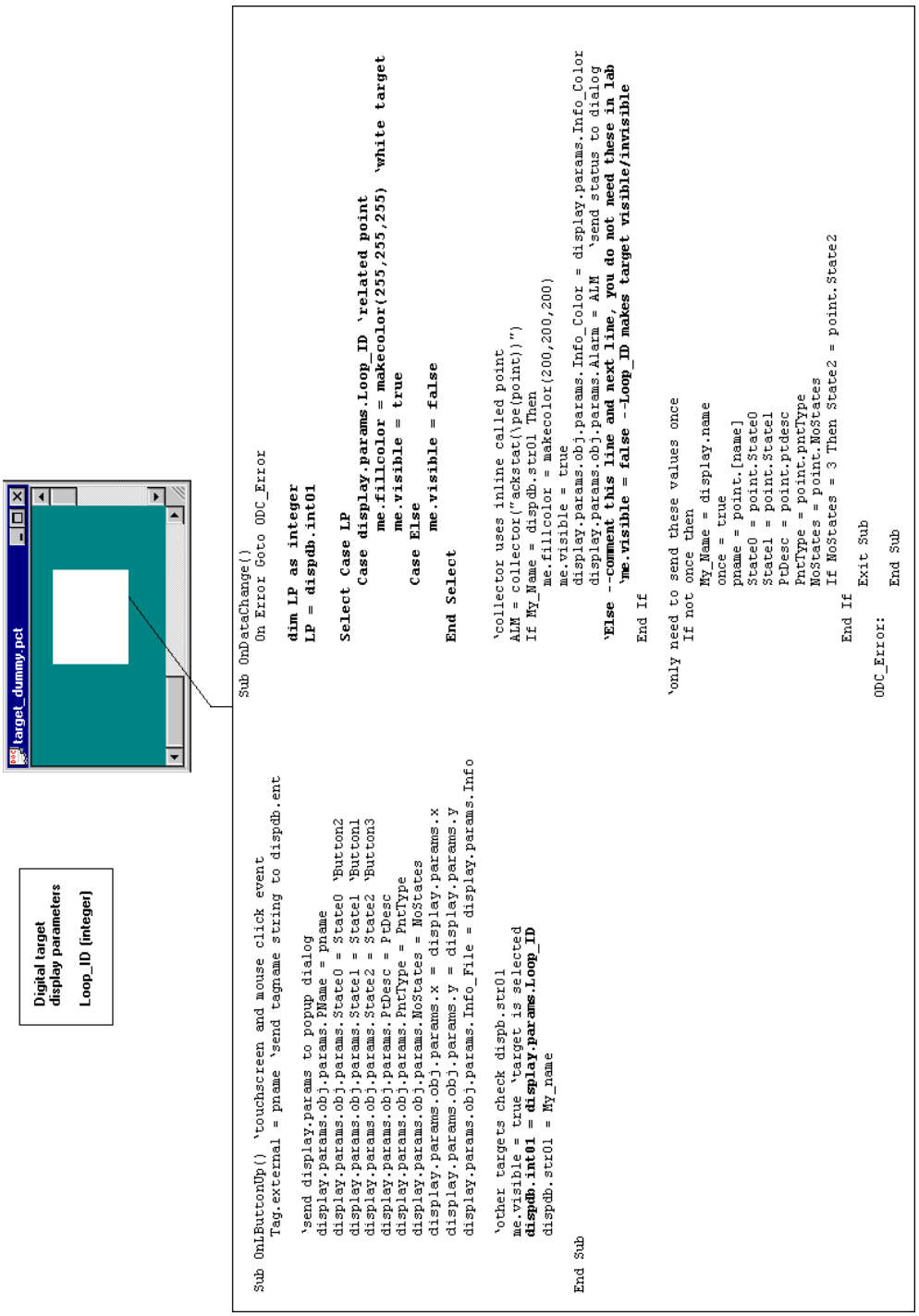
The popup dialog also provides the operator feedback that the requested output change is taking place. Two text objects will be added that provide feedback. The following OnDataChange script is added to the Feedback text object, FB_text, (not the Exit button) because it is not likely to run as often as the Exit button's OnDataChange script. The Feedback text object updates with the following script:

```
Sub OnDataChange()
      On Error Goto ODC_Error
      dim feedback as string
      dim c0 as long    'store TDC color for on/off
      dim c1 as long    'store TDC color for on/off

      'feedback text set input
      feedback = display.params.Tag.PV
      c0 = tdc_red
      c1 = tdc_green
      me.text = feedback
      'change button fillcolors to show change request
      If feedback = display.params.State1 Then
            Button1_State1.fillcolor = c1
            Button2_State0.fillcolor = makecolor(204,204,204)
            Button3_State2.fillcolor = makecolor(204,204,204)
            FB.visible = false
      Elseif feedback = display.params.State0 Then
            Button1_State1.fillcolor = makecolor(204,204,204)
            Button2_State0.fillcolor = c0
            Button3_State2.fillcolor = makecolor(204,204,204)
            FB.visible = false
      Elseif feedback = display.params.State2 Then
            Button1_State1.fillcolor = makecolor(204,204,204)
            Button2_State0.fillcolor = makecolor(204,204,204)
            Button3_State2.fillcolor = tdc_yellow
            FB.visible = false
      Else
            Button1_State1.fillcolor = makecolor(204,204,204)
            Button2_State0.fillcolor = makecolor(204,204,204)
            Button3_State2.fillcolor = makecolor(204,204,204)
            FB.visible = true

      End IF
      Exit Sub
ODC_error:
      me.text = "---"
      FB.visible = true
End Sub
```

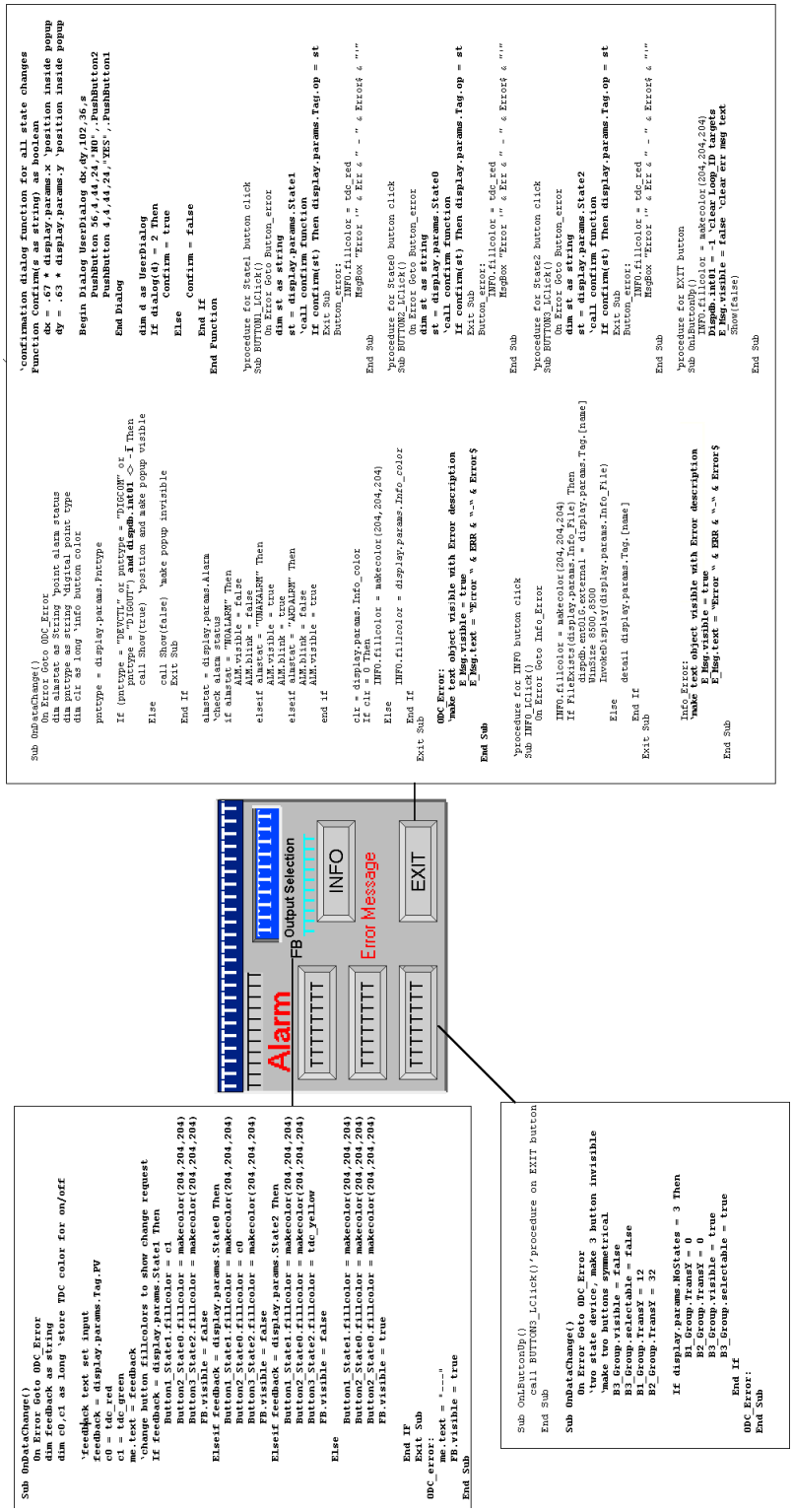
The popup dialog updates to show 2 buttons if there is a 2 state device, 3 buttons if there is a 3 state device. The following **boldfaced** statements will be added to the button3 rectangle object.

```
Sub OnLButtonUp()
      call BUTTON3_LClick()'procedure on EXIT button
End Sub

Sub OnDataChange()
      On Error Goto ODC_Error
      'two state device, make 3 button invisible
      'make two buttons symmetrical
      B3_Group.visible = false
      B3_Group.selectable = false
      B1_Group.TransY = 12
      B2_Group.TransY = 32

      If display.params.NoStates = 3 Then
            B1_Group.TransY = 0
            B2_Group.TransY = 0
            B3_Group.visible = true
            B3_Group.selectable = true
      End If
ODC_Error:
End Sub
```

The following figure summarizes changes to the digital target.



The following figure summarizes changes to the popup dialog.



Lab Procedure – Modify the Digital Target to Support Loop ID

Step	Action
1.	From your EmbedLab6 folder, open the digital target display from the earlier Lab 6 exercise called target6.pct.
2.	Save this target as target7.pct into your EmbedLab7 folder.
3.	Add the following display parameter to your digital target (Choose Display> Define Parameters): • Parameter: Loop_ID • Data Type: Integer • Initial value: 01 (Note: In R120 systems a port appears) • Prompt: Enter a number representing related points. Example: 01.

4.	Add script to your rectangle target's OnLButtonUp event so that the Loop_ID can highlight any related points of the control strategy. The script will follow the Design Criteria scripting example; the code fragment that you need to add is listed below in **bold**:
	`Sub OnLButtonUp() 'touchscreen and mouse click event`
	
	`{rest of your code that you entered earlier}`
	`'other targets check dispb.str01`
	`me.visible = true 'target is selected`
	`dispdb.int01 = display.params.Loop_ID`
	`dispdb.str01 = My_name`
	`End Sub`

5.	Check your syntax. (Ignore any errors against inline data types).

6.	Add script to your rectangle target's OnDataChange code so that the target can be highlighted in white when its Loop_ID is the same as related control points. The script can follow the Design Criteria scripting example, the code fragment that you need to add is listed below in **bold** (Note: Code fragments in bold are provided in a text file, Lab_code7.txt, in your EmbedLab7 folder to save you typing time):
	
	`Sub OnDataChange()`
	`On Error Goto ODC_Error`
	`dim LP as integer`
	`LP = dispdb.int01`
	
	`Select Case LP`
	`Case display.params.Loop_ID 'related point`
	`me.fillcolor = makecolor(255,255,255) 'white target`
	`me.visible = true`
	`Case Else`
	`me.visible = false`
	`End Select`
	
	`'collector uses inline called point`
	`ALM = collector("ackstat(\pe(point))")`
	{rest of your code that you entered earlier }
	
	`'Else --comment this line and next line, you do not need in lab`
	`'me.visible = false --Loop_ID makes target visible/invisible`
	{rest of your code that you entered earlier }

Step	Action
7.	Check your syntax.
8.	Validate the target7 display. (Note: You may get false validation errors against the inline data type. Ignore them.)
9.	Save the display as target7.pct into your EmbedLab7 folder.

Lab Procedure – Modify the Popup Dialog's Scripts

Step	Action
1.	From your EmbedLab6 folder, open up your popup dialog, dig_dialog6.pct, from the previous exercise.
2.	Save the display as dig_dialog7.pct into your Embed Lab7 folder.
3.	Add an Error Message text object so that it can report errors from your OnDataChange scripts. (Refer to the Design example on the previous page.) It is easier to build the text object off of the dialog. Leave the object off the dialog until a later step in the exercise instructs you to move it back on to the dialog. The Error Message text object has the following properties: • Text object name: E_Msg • Visible: False • Selectable: False • Fill: None • Multiline: true • Text: Error Message • Text color: red: • Alignment: center • Type: string • Expression: none

4.	Modify the OnDataChange script that is on the **Exit** button to include the changes from the Design Criteria section for the error handler code. (To access the script for the EXIT button, select the grouped popup dialog and access it's script window. From the script editor window, go to the object browser and select the EXIT object.)
	
	`Sub OnDatachange()`
	`{rest of your code appears here}`
	`ODC_Error:`
	`'make text object visible with Error description`
	`E_Msg.visible = true`
	`E_Msg.text = "Error " & ERR & "-" & Error$`
	`End Sub`
	
	Also add the following bold script to the statement that checks the PNTTYPE and calls the SHOW subroutine.
	
	`If (pnttype = "DEVCTL" or pnttype = "DIGCOM" or _`
	`pnttype = "DIGOUT") and dispdb.int01 <> -1 Then`
	`call Show(true) 'position and make popup visible`

Step	Action

5.	Modify the `INFO_LClick()` script that is on the EXIT button to include these changes from the Design Criteria section for the INFO error handler code.
	
	`Sub INFO_LClick()`
	`{rest of your code appears here}`
	`Info_Error:`
	`'make text object visible with Error description`
	`E_Msg.visible = true`
	`E_Msg.text = "Error " & ERR & "-" & Error$`
	`End Sub`

6.	Add a function following the OnDataChange script that is on the EXIT button. The function creates a dialog within the popup dialog. The confirmation dialog asks the operator to confirm the requested state change. If needed, refer to the Design Criteria section for the code placement.
	
	`'confirmation dialog function for all state changes`
	`Function Confirm(s as string) as boolean`
	`dx = .67 * display.params.x 'position inside popup`
	`dy = .63 * display.params.y 'position inside popup`
	
	`Begin Dialog UserDialog dx,dy,102,36,s`
	`PushButton 56,4,44,24,"NO",.PushButton2`
	`PushButton 4,4,44,24,"YES",.PushButton1`
	`End Dialog`
	
	`dim d as UserDialog`
	`If dialog(d) = 2 Then`
	`Confirm = true`
	`Else`
	`Confirm = false`
	`End If`
	`End Function`

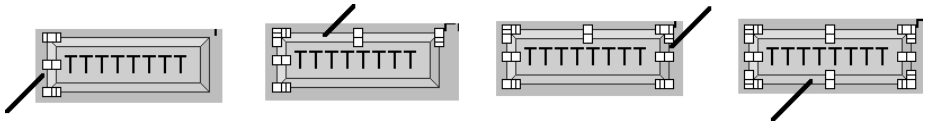
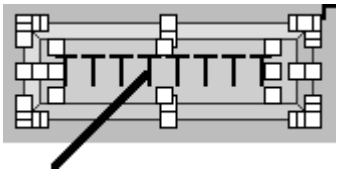
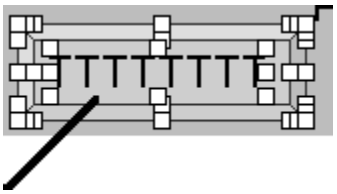
7.	Check your syntax.

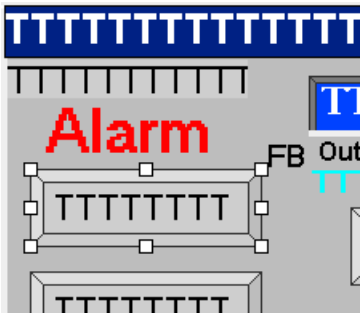
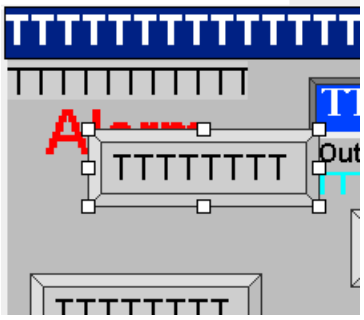
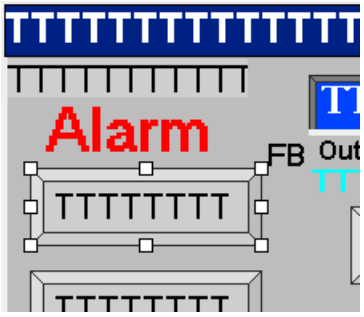
Step	Action

8.	On the EXIT button, modify the custom subroutine scripts for all 3 of your state-change buttons. The actual buttons, when selected, use an OnLButtonUp event to call these subroutines which are located on the EXIT button. When the following code is placed in the called subroutine, whenever a state change request is made, a confirmation dialog will now appear. An example script for one of the three state change buttons is listed below:
	
	`'state1 button, button1`
	`Sub BUTTON1_Lclick()`
	`On Error Goto Button_error`
	`dim st as string`
	`st = display.params.State1`
	`'call confirm function`
	`If confirm(st) Then display.params.Tag.op = st`
	`Exit Sub`
	`Button_error:`
	`INFO.fillcolor = tdc_red`
	`MsgBox "Error '" & Err & " - " & Error$ & "'"`
	`End Sub`
	(Note: This code fragment is in your Lab7_code.txt file in the EmbedLab7 folder.)

9.	Check your syntax.
10.	Save the display as dig_dialog7.pct into your Embed Lab7 folder.
11.	The popup dialog also provides the operator feedback that the requested output change is taking place. Add two text objects that provide feedback. (Note: Build these objects off of the dialog so that is easier to create them. Later in this exercise you position them on the dialog.) The FB_text text object has the following properties. Its properties are changed by script at runtime: • Name: FB_text • Visible: True • Selectable: False • Fill: None • Text: TTTTTTTTTTT • Format: MS Sans Serif, 8 pt, aqua • Multiline: true • Alignment: left • Type: string • Expression: none The FB_label text object has the following properties: • Name: FB_label • Visible: True • Selectable: False • Fill: None • Text: FB • Format: MS Sans Serif, 8 pt, black • Alignment: Left • Type: string • Expression: none
12.	While the two feedback text objects are off of the dialog, group just these two text objects (Choose Draw>Group).

Step	Action
13.	Give the feedback text objects a group name of FB, and set the selectable property to false. (Note: The FB visibility is controlled at runtime. You can leave it enabled.)
14.	Select the popup dialog and ungroup it. (You can OK the message about destroying group scripts because there are no scripts on the group itself. Your other scripts, such as those on the EXIT button, remain intact.)
15.	Click in an open area of the dialog display so that the selection handles on your dialog no longer appear.
16.	Select the Error Message text object and position it on your dialog according the Design Criteria section (between the INFO button and the EXIT button).
17.	Select the FB text group and position it on your dialog according the Design Criteria section.
18.	Save your display.

Step	Action
19.	The 3 state change buttons are currently an ungrouped collection of objects. Each button consists of 4 polygons that make up the beveled button edges, a text object, and a rectangle object. You will need to group each button as its own object and give it a name. The reason you need to group the button objects is that you add script to the dialog to show only 2 buttons if there is a 2 state device and 3 buttons if there is a 3 state device. Making the buttons invisible (and symmetrical) is easier from script if you group each button. Having said that, selecting smaller objects that are on top of other objects can be tricky in any kind of graphic program. The following diagrams show a way to easily select smaller objects residing on larger objects. • First, make sure no other display object is selected. You can click anywhere off of the popup dialog to make sure no other object is selected. • Press the Shift key and click each of the polygons that make up Button1. Use the diagonal line as a guide for clicking on the object.  • While keeping the Shift key down, click on the text object.  • While keeping the Shift key down, click the rectangle object just slightly below the text object. 

Step	Action
20.	Group all the button objects (Choose Draw>Group).  To verify that you now have a grouped button, move the button slightly out of position. (If you grouped correctly, all objects should move together as one object.)  To return the button group to its original position, choose Edit>Undo (or Ctrl + Z) to undo the move. 
21.	If your button is correctly grouped, from the grouped object's General Property tab, give the Button 1 group an object name of B1_Group. Verify that its selectable property is enabled. (Note: Script that you add later in this exercise references this button grouping as B1_group, so use the name B1_group.)
22.	Repeat the above steps for your Button 2 group. Give the grouping for button2 a name of B2_group.
23.	Repeat the above steps for your Button 3 group. Give the grouping for button3 a name of B3_group.
24.	Select all objects in your dialog (Choose Edit>Select All).
25.	Group all objects for the dialog (Choose Draw>Group).
26.	Once again, give the grouped dialog objects a name of "Pop." (Remember: "Pop" is the name used for the dialog in your script.)
27.	Click on the popup dialog and open a script editor window. Result: A script window opens for the object named "Pop."

Step	Action
28.	From the script editor window's object browser, select the object called FB_text. Result: A script window opens for FB_text.
29.	Add the following OnDataChange script to the text object, FB_text. (Note: You can copy this script from the .txt file in your EmbedLab 7 folder.)

```
Sub OnDataChange()
      On Error Goto ODC_Error
      dim feedback as string
      dim c0 as long    'store TDC color for on/off
      dim c1 as long    'store TDC color for on/off

            'feedback string set equal to PV
      feedback = display.params.Tag.PV
      c0 = tdc_red
      c1 = tdc_green
      me.text = feedback
        'change button fillcolors to show change request
      If feedback = display.params.State1 Then
      Button1_State1.fillcolor = c1
      Button2_State0.fillcolor = makecolor(204,204,204)
      Button3_State2.fillcolor = makecolor(204,204,204)
            FB.visible = false
      Elseif feedback = display.params.State0 Then
      Button1_State1.fillcolor = makecolor(204,204,204)
      Button2_State0.fillcolor = c0
      Button3_State2.fillcolor = makecolor(204,204,204)
            FB.visible = false
      Elseif feedback = display.params.State2 Then
      Button1_State1.fillcolor = makecolor(204,204,204)
      Button2_State0.fillcolor = makecolor(204,204,204)
      Button3_State2.fillcolor = tdc_yellow
            FB.visible = false
      Else
      Button1_State1.fillcolor = makecolor(204,204,204)
      Button2_State0.fillcolor = makecolor(204,204,204)
      Button3_State2.fillcolor = makecolor(204,204,204)
      FB.visible = true
      End IF
      Exit Sub
ODC_error:
      me.text = "---"
      FB.visible = true
End Sub
```

30.	The popup dialog needs to update and show 2 buttons if there is a 2 state device, 3 buttons if there is a 3 state device. From the script editor window's object browser, select the object called Button3_State2. Result: A script window opens for Button3_State2.

Step	Action

31.	Add the following **boldfaced** script statements to the **Button3** rectangle object, Button3_State2.
	`Sub OnLButtonUp()`
	`  call BUTTON3_LClick()  'procedure on EXIT button`
	`End Sub`
	
	`Sub OnDataChange()`
	`  On Error Goto ODC_Error`
	`  'two state device, make 3 button invisible`
	`  'make two buttons symmetrical`
	`  B3_Group.visible = false`
	`  B3_Group.selectable = false`
	`  B1_Group.TransY = 12`
	`  B2_Group.TransY = 32`
	`  If display.params.NoStates = 3 Then`
	`  B1_Group.TransY = 0`
	`  B2_Group.TransY = 0`
	`  B3_Group.visible = true`
	`  B3_Group.selectable = true`
	`  End If`
	`ODC_Error:`
	`End Sub`
	(Note: This code can also be found in the .txt file in your EmbedLab 7 folder.)

32.	The popup dialog requires script to clear the highlighted targets when the EXIT button is selected. From the script editor window's object browser, select the object called EXIT. Result: A script window opens for EXIT.

33.	Add the following **boldfaced** script statements to the EXIT rectangle object.
	`  'procedure for EXIT button`
	`Sub OnLButtonUp()`
	`  INFO.fillcolor = makecolor(204,204,204)`
	`  Dispdb.int01 = -1  'clear Loop_ID targets`
	`  Dispdb.str01 = ""  'clear name from string`
	`  E_Msg.visible = false  'clear err msg text`
	`  Show(false)`
	`End Sub`

34.	Validate the dig_dialog7.pct display. (Note: You may get false validation errors against the inline data types. Ignore them.)
35.	Save the display as dig_dialog7.pct in your Embed Lab7 folder.

Lab Procedure – Modify the Display

Step	Action
1.	From your EmbedLab6 folder, open your embed6.pct display from the previous lab exercise.
2.	Save your embed6.pct display as embed7.pct into your EmbedLab7 folder.
3.	Replace your previous target6.pct from the embed7 display and insert the new target7.pct. (Choose Edit>Replace Embedded Display).
4.	Move the popup dialog if it is obstructing the view to any of your digital targets.
5.	Select each of the targets and set the display parameter for Loop_Id to "1". (Choose Edit>Enter Parameters).
6.	Replace your previous dig_dialog6.pct from the embed6 display and insert the new dialog, dig_dialog7pct (Choose Edit>Replace Embedded Display).
7.	Save the display as embed7.pct into your EmbedLab7 folder.
8.	Move the popup dialog back to the upper left-hand corner of the display so that its transX and transY properties work correctly.
9.	Validate your embed7.pct display.
10.	Save the display as embed7.pct into your EmbedLab7 folder.
11.	Run the display.
12.	Select a digital target Expected result: The non-selected target changes to white, the selected target changes to a darker gray color. The popup dialog now shows two state buttons (symmetrical to the INFO and EXIT buttons) instead of three buttons. The buttons have a fillcolor for the current state.
13.	From the popup dialog, make a state change request. Expected result: A confirmation dialog appears within the popup dialog.
14.	From your embed7 display, select a digital composite point that is in Program Manual (P-MAN) mode and try to change the state. Expected result: Error message box appears indicating that the mode is incorrect. (The INFO button changes to red because a runtime error occurred).
15.	Clear the runtime error message.
16.	To create a feedback alarm in the lab exercise environment, you can change PVSOURCE to Manual. Make a state change request. Expected result: The Alarm text appears.
17.	Put your point's PVSOURCE back to Auto. Now set the point to INACTIVE. Expected result: The feedback text appears as BAD.
18.	Select another target in your display. Expected result: The non-selected target changes to white, the selected target changes to a darker gray color.
19.	Select the EXIT button. Expected result: The popup dialog and targets become invisible.

Last Page