Honeywell

Lab Exercise – Pass Data with Object Data Type

57311002L
6/00

TotalPlant

Notices and Trademarks

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

TPS is the evolution of TDC 3000X.

Honeywell

Industrial Automation and Control

Automation College

2500 W. Union Hills Drive

Phoenix, AZ 85027

1-800 852-3211

Lab Exercise 2

Introduction

The following lab exercise introduces the use of the object data type to pass data between embedded displays. While it is possible to use other mechanisms to pass data, such as display database (DispDb) variables, the object data type represents a faster, more performant way to pass data. This does not mean that the DispDb variables are useless in our strategy, rather that they play a secondary role in the popup dialog design.

Object Data Type Role

To visualize the object data type's role in this lab exercise, think of it as a role where one display object (in this case, a rectangle target) temporarily "owns" another display object (in this case, the popup dialog). Because the target temporarily "owns" the popup dialog when the target is selected, the target can then pass data to the popup dialog's display parameters (display.params). Recall that the dialog had this behavior in the previous introduction lab exercise. Included in the data that is passed from the target to the popup dialog is an LCN tagname. The target 'releases its ownership' of the popup dialog when another target is selected. At that time, the newly selected target "owns" the popup dialog.

To use the object data type in this lab exercise requires essentially four steps:

1. Define a display parameter in the sending object, in this case the rectangle target, as object data type. Give the 'object type' display parameter a name you can easily remember (example: controller). This name will eventually be used as the name for the popup dialog.

2. Write script for the sending object that will pass data to the receiving object using the following syntax:

```
display.params.[receiving object name].params.[receiving object parameter]
```

3. Rename the receiving object to have the same display parameter name you had defined in Step 1 (Example: Controller).

4. Define display parameter(s) (display.params) in the receiving object to act as mailboxes to receive the data from the sending object display parameters.

For example, the following code fragment sends data to another display object (called “controller”).

Display.params.obj.params.PtDesc = PtDesc

Where:

> “Display.params.obj” refers to an object data type display parameter ‘obj’ defined in the sending object. “Controller” is the display object’s name of the receiving object; it’s easier to follow your code if you rename the objects that you insert into your container display from EmbeddedPicture# to some another name.
>
> “Params.PtDesc” refers to a parameter defined on the receiving object, which is another embedded picture. Params.PtDesc is assigned a value from the sending object’s display parameter, PtDesc.

In order to provide an orderly approach to pop-up dialog building and grasp some of the underlying concepts, the goals in this exercise are modest. You first build the targets and dialogs so that data can be passed to the dialog.

Objectives

At the end of this lab exercise, you will be able to

- Define the object data type as a display parameter.
- Pass data from one embedded display to another using the object data type.

Design Criteria

The following lab exercise consists of a partially completed popup dialog. You are certainly welcome to make your own popup dialog, but to save you lab display building time, much of the static object drawing is already done for you. Your pre-built popup dialog does not have any functionality coded into it yet; the necessary code is what you will add in the following lab exercises. Your pre-built popup dialog contains buttons for:

- A 3 state digital composite point. Later lab exercises add code that makes the popup dialog display 2 buttons for 2 state points and display 3 buttons for 3 state points. In the first lab exercises you will leave the third state button as a "dumb" non-functional static object. (You will add code in a later lab exercise that makes the popup dialog appear when needed and change from 2 buttons to 3 buttons, as required.)

- Two other buttons that will be used for calling up related displays are the following:

  - an INFO button and

  - an EXIT button to close the dialog. (The INFO and EXIT buttons are non-functional at this point in the beginning lab exercises. You will add code to them in later lab exercises.)

The design criteria for this lab exercise is limited to the following:

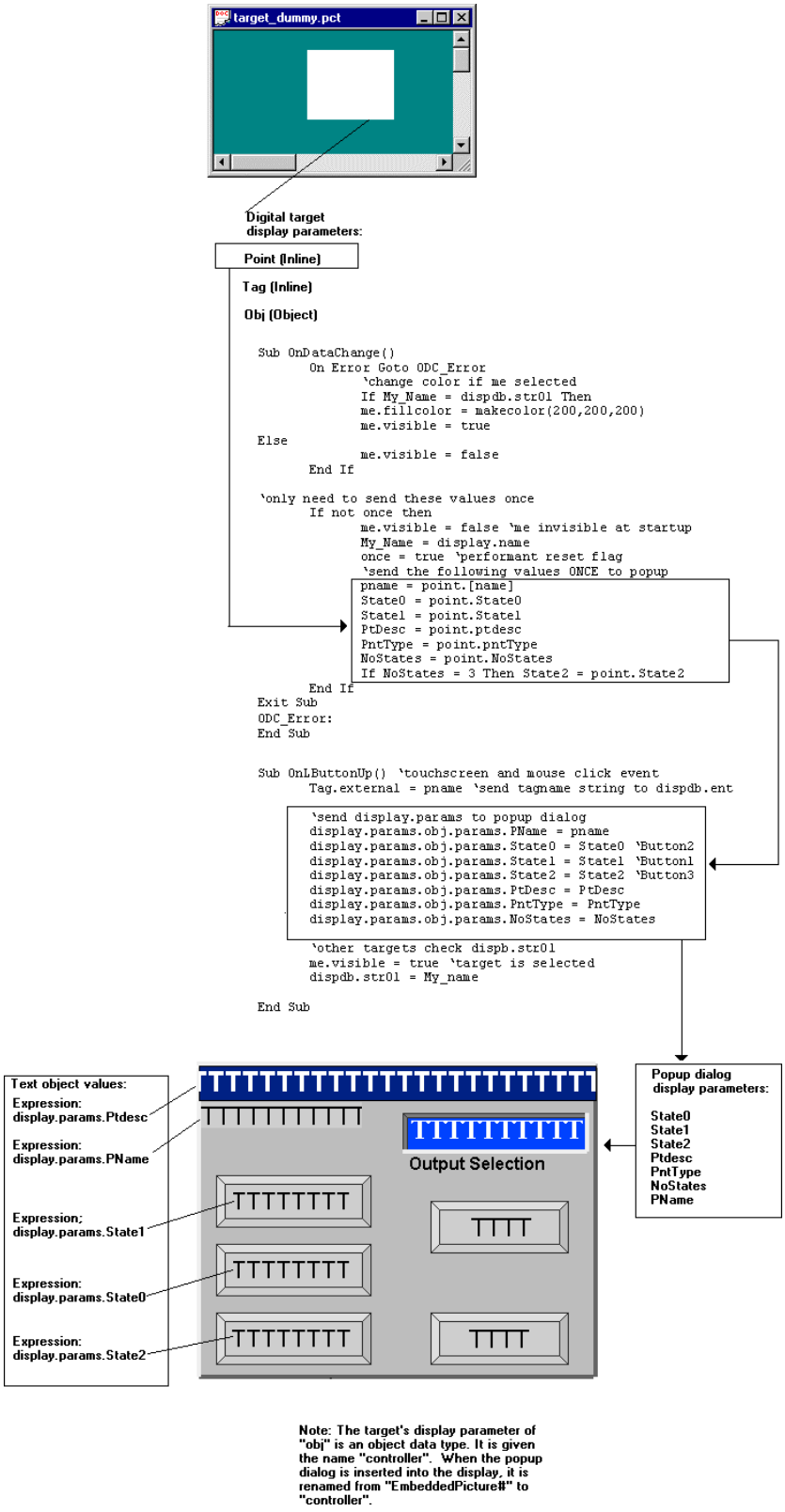
- Build a rectangle target that passes data to the popup dialog. Here is where you define the display parameters to do this, more specifically, here is where you define the object data type display parameter. Next, code script on the target to pass data using the object data type and display.params references.

- Modify the popup dialog to receive the data. To provide this functionality, you define the receiving object display parameters to act as mailboxes that receive the digital target's data.

- Embed the target and popup dialog into another display.

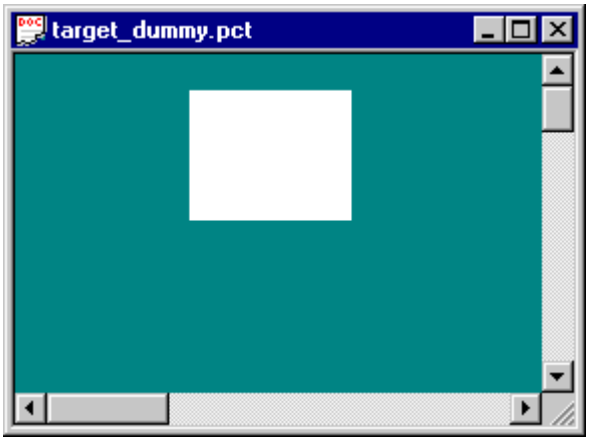
- At display runtime, select the target to send data to the popup dialog.

The overall flowchart is shown in the following figure.



Lab Design Criteria for Rectangle Target

The rectangle target in the following figure passes LCN process point type data for digital points (i.e., digital composite, digital output, device control, etc.).

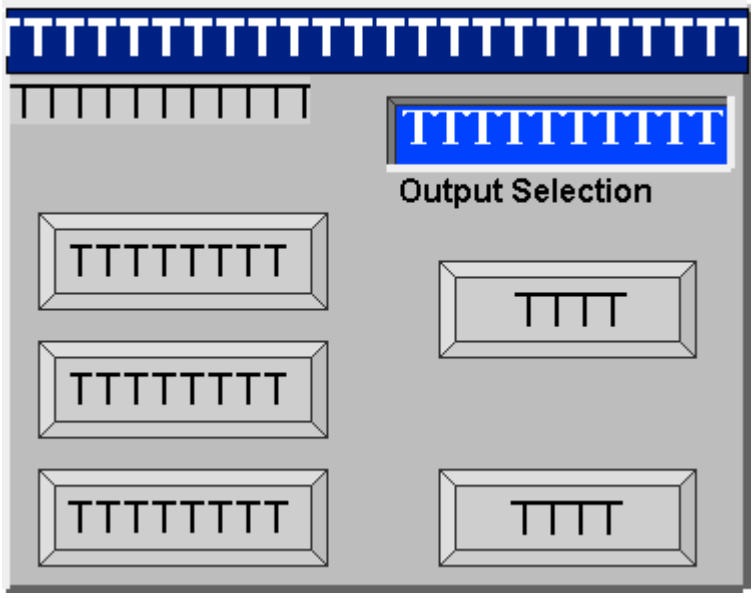


In the first lab exercise, you will define the following display parameters to pass data:

- Point – an inline parameter that passes point parameters.
- Tag- an inline parameter that accesses the LCN entity itself via a display database (DispDb) entity.
- Obj- an object data type that permits a target to write to another display object.

Lab Design Criteria for the PopUp Dialog

This popup dialog is used for digital point types only.



Display parameters that you define for the dialog are the following:

- Tag- an entity type parameter that is set to read the same display database entity (dispdb.ent01) as the digital target.

The remaining parameters receive values from the digital target and are self-explanatory as to what they represent:

- Ptdesc - string represents the point descriptor.
- PntType – string representing the point type.
- State0 – string represents the State0 descriptor.
- State1 – string represents the State1 descriptor.
- State2 – string represents the State2 descriptor.
- NoStates – integer represents the number of states.
- PName – string represents the point name.

Lab Prerequisites

Lab prerequisites are:

- Native Window is loaded.
- Two off process LCN digital composites points.
- A pre-built dialog and target are available in the EmbedLab2 folder to save you display building time.

Lab Procedure

In the following procedure, you will complete a simple rectangle target that passes data to the popup dialog. For now, the target and popup dialog has limited functionality. This lab exercise goal is to help you become familiar with using the object data type to pass data between displays.

Step	Action
1.	From your EmbedLab2 folder, open the display target_dummy.pct. Result: a rectangle object appears.
2.	Save the display target2.pct into your EmbedLab2 folder.
3.	Define parameters for your target2.pct display. (Choose Display>Define Parameters). Result: The Enter/Add Parameters dialog appears.

4.	Enter the parameters (choose Add>Enter Parameter> DataType> Prompt> OK, then repeat for remaining parameter entries):
	`Parameter: Point`
	`Data Type: Inline`
	`Prompt: Enter the point name. Examples:`
	`LCN.FVL21241 , LCN.[75m1]`
	(Note: The parameter "Point" is used to pass an entity's parameters to an embedded display. The prompt reminds the end user to enter a tagname.)
	`Parameter: Tag`
	`Data Type: Inline`
	`Prompt: Enter the DispDb entity to use Example:`
	`DispDB.ent01`
	(Note: The parameter "Tag" is used to reference the entity itself. The popup dialog also references the same dispdb entity, DispDB.ent01. A DispDB entity is used because other displays access this value.)
	`Parameter: Obj`
	`Data Type: Object`
	`Prompt: Enter the popup dialog name. Example:`
	`Controller`
	(Note: The parameter "Obj" is the most important parameter in this design. It allows you to pass data to another display.)

5.	Choose Close in the dialog after entering all of the display parameters.
6.	Save your display.
7.	Open a script edit window for your rectangle object.

8.	To send data to the popup dialog requires that you enter script that responds to an operator selecting the target. Enter the following script:
	Script Text:
	`Global Ptdesc as string`
	`Global pname as string`
	`Dim My_Name as string`
	`dim State0 as string`
	`Dim State1 as string`
	`Dim State2 as string`
	`Dim PntType as string`
	`Dim NoStates as integer`
	`Dim once as boolean`
	
	`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.State1 = State1 'Button1`
	`display.params.obj.params.State0 = State0 'Button2`
	`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`
	
	`'other targets check dispb.str01`
	`me.visible = true 'target is selected`
	`dispdb.str01 = My_name`
	`End Sub`

9.	Note: If you do a "Syntax" check on the above script you will receive an error on Tag.external = pname. Inline parameters cause validation errors; ignore the error.

10.	Next, to cause an update to the popup dialog from the rectangle target requires an OnDataChange script on your target. Right now, the script only checks to see if the target is selected. By using a 'once' flag, the code makes a performant one time pass of point type data to the display parameters. Enter the following OnDataChange script on your digital target.
	
	`Sub OnDataChange()`
	`  On Error Goto ODC_Error`
	`  'change color if me selected`
	`  If My_Name = dispdb.str01 Then`
	`  me.fillcolor = makecolor(200,200,200)`
	`  me.visible = true`
	`  Else`
	`  me.visible = false`
	`  End If`
	
	`'only need to send these values once`
	`  If not once then`
	`  me.visible = false 'me invisible at startup`
	`  My_Name = display.name`
	`  once = true 'performant reset flag`
	`  'send the following values ONCE to popup`
	`  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`

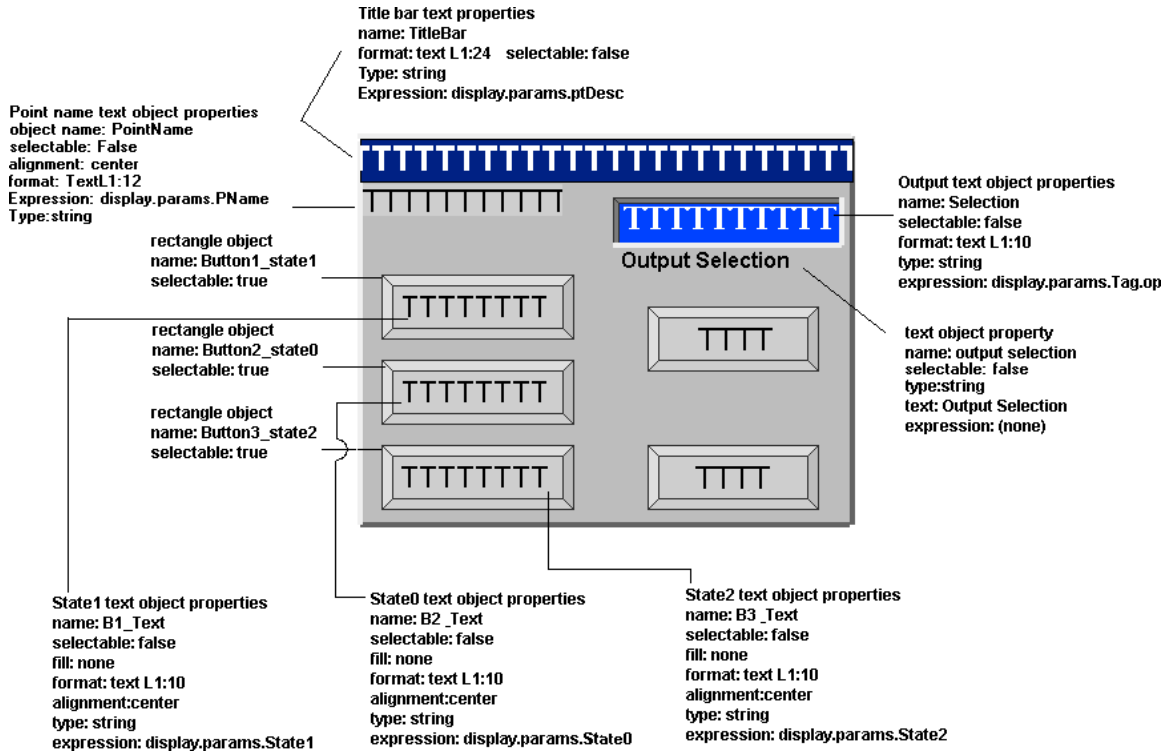
11.	Syntax check the display. (Note: You may get false syntax check errors against the inline data types that are used in this display. Ignore errors against inline data types, but fix any other errors.)
12.	Close the script editor window.
13.	Validate the display. (Note: You may get false validation errors against the inline data types that are used in this display. Ignore errors against inline data types, but fix any other validation errors.)
14.	Save the display as target2.pct in your EmbedLab2 folder.
15.	Close the display.

Lab Procedure: Overview

In the following procedure, you make initial modifications to a popup dialog so that it can display the data it received from the target. Its functions are intentionally limited in the beginning lab exercises to help you become familiar with the object data type and how it is used to pass data.

Design Criteria: Text objects in popup dialog

Use the following figure as a guide for your data entry.



Lab Procedure: Modify the popup dialog to receive data

Step	Action
1.	From your EmbedLab2 folder, open the display dig_dialog_dummy.pct. Result: a popup dialog appears with pre-built buttons. (Note: these buttons are already built as static objects to help save lab time. We suggest that you accept them "as is' for now until you see how the dialog operates.)
2.	Save the display dig_dialog2.pct into your EmbedLab2 folder
3.	Define parameters for your dig_dialog2.pct display. (Choose Display>Define Parameters). Result: The Enter/Add Parameters dialog appears.

4.	Enter the following parameters (
	`Parameter: Tag`
	`Data Type: Entity`
	`Prompt: Enter the DispDb entity to use Example: DispDB.ent01`
	(Note: The display database entry, in this case DispDB.ent01, must be the same as the entry used for the display parameter Tag from your digital target)
	The remaining entries represent display parameters that receive values from the digital target. They do not require prompts.
	`Parameter: PtDesc`
	`Data Type: string`
	
	`Parameter: PntType`
	`Data Type: string`
	
	`Parameter: State0`
	`Data Type: string`
	
	`Parameter: State1`
	`Data Type: string`
	
	`Parameter: State2`
	`Data Type: string`
	
	`Parameter: NoStates`
	`Data Type: Integer`
	
	`Parameter: Pname`
	`Data Type: string`

5.	After entering all display parameters, choose Close in the dialog.
6.	Validate your popup dialog display.
7.	Save your display.
8.	Define the text objects to reference the display parameter values as shown in the previous figure in "Design Criteria: Text objects in Popup dialog."
9.	Validate your popup dialog display.
10.	Save your display.

Lab Procedure: Use the modified target and popup dialog

In the following procedure, you use your modified target and popup dialog to pass data from the selected target to the popup dialog.

Step	Action
1.	From your EmbedLab2 folder, open the dialog_pass.pct display. Result: A display with static objects appears.
2.	Save the display as embed2.pct in your EmbedLab2 folder.
3.	Insert the partially built dialog that you had just completed in the earlier procedure, dig_dialog2.pct, as an embedded display into your embed2.pct display. • Choose Insert>Display. • Choose dig_dialog2.pct (This is the dialog that you have just modified)
4.	When the Enter Parameters dialog appears, enter the following: • Tag: DispDB.Ent01 (Note: the other parameters do not need definition, they receive their values from the digital target.)
5.	Click on a position within your embed2.pct display. (You may want to move the dialog so that it does not cover up any process elements).
6.	Select the dialog and rename the dialog from “EmbeddedPicture1” to “controller”.
7.	Insert the digital target that you had just completed in the earlier procedure, target2.pct, as an embedded display into your embed2.pct display. • Choose Insert>Display. • Choose target2.pct (This is the target that you have just modified)
8.	When the Enter Parameters dialog appears, enter the following: • Point: LCN.FVL21###(where ### represents your partition number.) Scroll down to the next parameter • Tag: DispDb.Ent01 Scroll down to the next parameter • Obj: controller (Note: Controller becomes your popup dialog’s name. Enter the string ‘controller’ in the Initial Value expression field.)
9.	Choose OK
10.	Click on a position over a process element within your embed2.pct display. Result: The target covers the valve.
11.	Move the digital target behind the process element (Draw>Send to Back).
12.	Repeat the five previous steps for inserting another digital target’s composite point, FVL22###.
13.	Validate the display.

14.	Save the display as Embed2.pct.
15.	Run the display. Result: The popup dialog appears (Note: In later lab exercises you add code that makes the dialog visible on demand and go to a relative position in the display.)
16.	Select a process element. Result: The target backlights behind the process element when it is selected. The popup dialog shows the current tag and data. (Note: At this time the dialog is not coded to send an output command to the process. Additional functionality will be added in later lab exercises.)

Last Page