Honeywell

Lab Exercise – Create a Generic Display with DDBs

57310302L
11/99

TotalPlant

Notices and Trademarks

Copyright 1999 by Honeywell Inc.
Revision 01 Date 11/99

Honeywell IAC courseware is subject to change without notice.

FLEXTRAINING courseware is copyrighted and all rights are reserved by Honeywell Inc. These materials are intended solely for use in conjunction with Honeywell products. The materials comprising the courseware may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form with out the prior, express written consent of Honeywell Inc.

Honeywell and **TotalPlant** are U.S. registered trademarks of Honeywell, Inc.

Other brand or product names are trademarks of their respective owners.

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

TPS is the evolution of TDC 3000^{X}.

Honeywell Inc.
Industrial Automation and Control
Automation College
2820 West Kelton Lane
Phoenix, AZ 85053-3028
1-800 852-3211

Lab Exercise

Introduction

The Display database (DispDB) supports the creation of generic displays. A reusable generic display can reduce authoring effort through the use of global display database variables. By using global display database variables, values can be passed from one display to another upon display upon invocation using the InvokeDisplay command.

Objectives

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

- Create a generic display that is passed database values from a menu.
- Interpret the behavior of global display database variables.

Design Criteria

The design criteria follows the generic display shown in the Display Authoring Tutorial, in the section called "Display Database Model." Sample displays and scripts are shown on the following pages.

The display used as the reactor menu requires script similar to the following for each menu item. Be sure to change the entity references to 4 sets of LCN points and the string references to display titles (Reactor 1, Reactor 2, Reactor 3, and Reactor 4). You can access points outside of your partition, but be sure to let the other users know you are doing that.

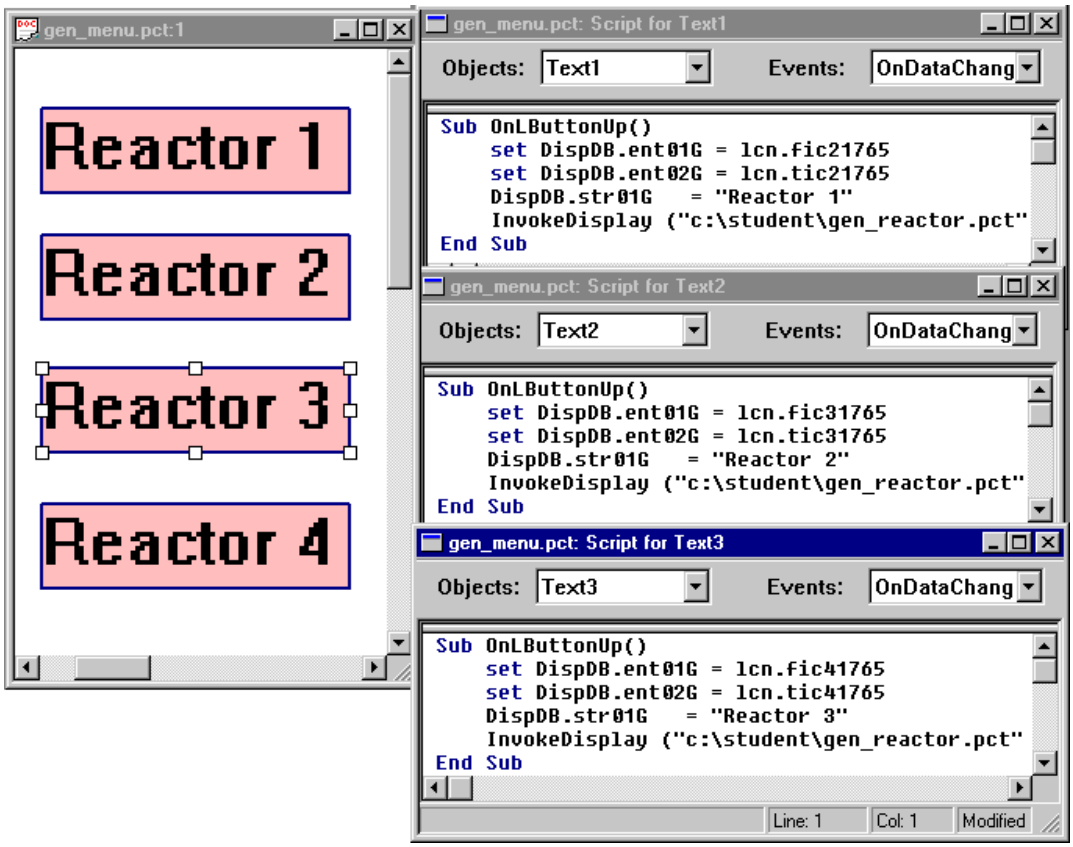
```
Sub OnLButtonUp()
Set dispDB.Ent01G = lcn.fic21###  'flow point
Set dispDB.Ent02G = lcn.tic21###  'temperature point
DispDB.Str01G = "Reactor 1"       'display title
InvokeDisplay ("c:\student\gen_reactor.pct")
End Sub
```

Lab Prerequisites

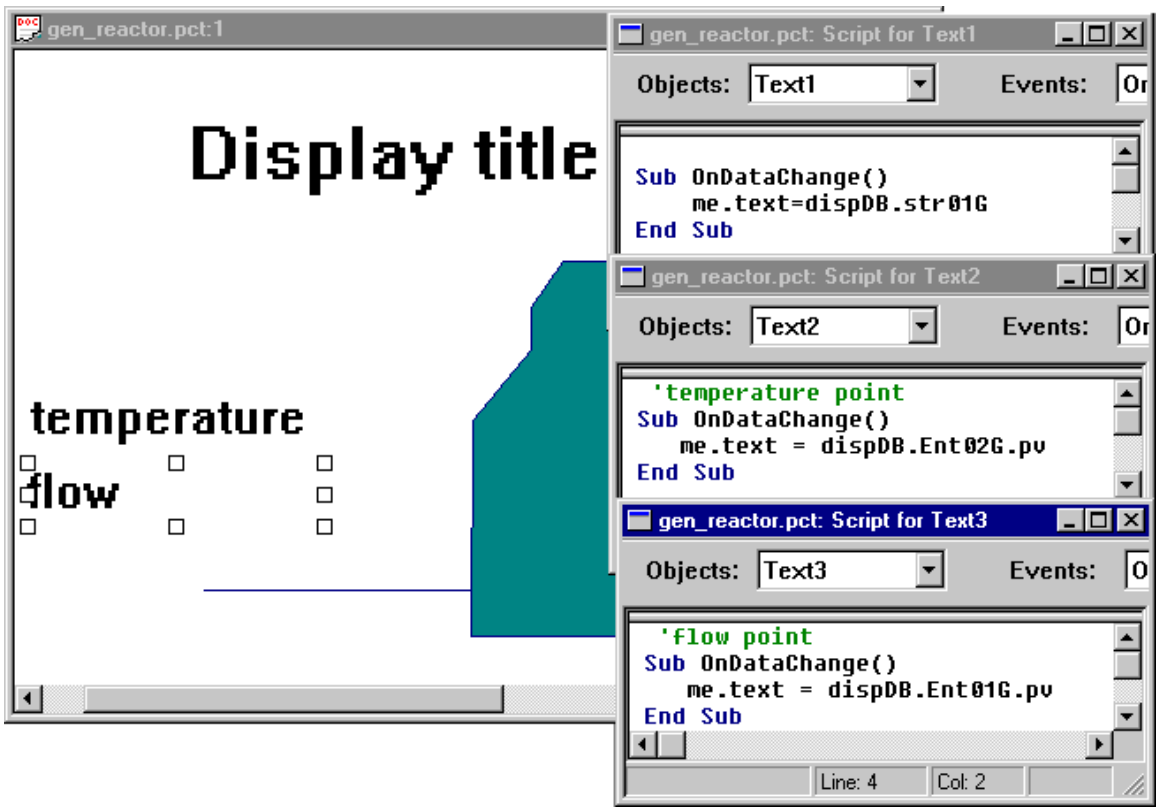
Lab prerequisites are the following:

- GUS Display Builder
- Several off-process control points
- Native Window is running

An example display used as the reactor menu is shown below. (You do not have to be too concerned about how aesthetically pleasing your display is; this lab is more concerned about the concepts behind display database scripting.) This reactor menu has four text items (Reactor 1, Reactor 2, etc.) that will act as targets to call specific information for each reactor.



An example display used as the reactor is shown in the following figure. The below example will be invoked by selecting a “Reactor” target on the menu display. The display’s title, temperature value and flow value will be different based on the menu target selected to call the reactor. The global display database will be used to pass information from the invoking display (the menu) to the invoked display (the reactor).



The above generic reactor display requires three text objects with the following script.

```
‘display title
Sub OnDataChange ()
Me.text=DispDB.str01G
End Sub

‘flow point
Sub OnDataChange ()
Me.text= dispDB.Ent01G.PV
End Sub

‘temperature point
Sub OnDataChange ()
Me.text= dispDB.Ent02G.PV
End Sub
```

Lab Procedure

Step	Action
1.	From the GUS Display Builder, open a new display as Gen_menu.pct
2.	Build four text objects with solid fillcolor patterns and label them Reactor 1, Reactor 2, Reactor 3, Reactor 4.
3.	Add script similar to the following for each text object. (Refer to the Design Criteria section for more details.) **Sub OnLButtonUp()** **Set dispDB.Ent01G = lcn.fic21### 'flow point** **Set dispDB.Ent02G = lcn.tic21### 'temperature point** **DispDB.Str01G = "Reactor 1" 'display title** **InvokeDisplay ("c:\student\gen_reactor.pct")** **End Sub**
4.	Syntax check your display.
5.	Validate your display.
6.	Save your display as Gen_menu.pct in your student folder.
7.	From the GUS Display Builder, open another new display.
8.	Create a simple vessel shape. (Do not worry about how pleasing your display vessel appears; you are mainly interested in scripting display database variables.)
9.	Add script to the objects according to the information shown in the Design Criteria section. **'display title** **Sub OnDataChange ()** **Me.text=DispDB.str01G** **End Sub** **'flow point** **Sub OnDataChange ()** **Me.text= dispDB.Ent01G.PV** **End Sub** **'temperature point** **Sub OnDataChange ()** **Me.text= dispDB.Ent02G.PV** **End Sub**
10.	Syntax check your display.
11.	Validate your display.
12.	Save your display as Gen_reactor.pct.
13.	Run your Gen_menu display that contains your menu items.
14.	Invoke your generic reactor display from each of the menu items. Expected result: The Gen_reactor display is invoked with global variables passed to it. The display updates to show the new entities. (Note: Run time errors, such as Error 1053, may be due to the point not having a good PV value. Normally you would code On Error handlers in your scripts to check for these conditions.)

Last Page