Honeywell

Lab Exercise – Creating a User-Defined DDB

57310301L
11/99

TotalPlant

Notices and Trademarks

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 following concept lab shows you how to build and load a user-defined database for DispDb variables.

Objectives

Upon completing this lab exercise, you will be able to

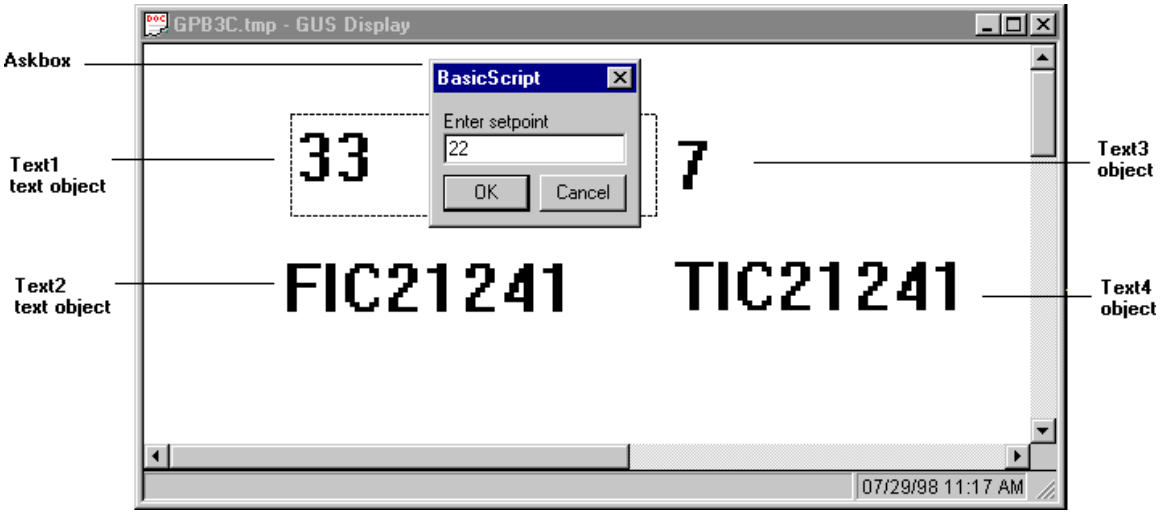
- Create a user-defined declare file.
- Load the user-defined database file into a GUS display.
- Access the user-defined database variables in a display.

Design Criteria

Your user-defined declare file, “user_def”, should include at least the following statements:

```
LOOP1:   entity,   300, global; {control loop entity}
LOOP1sp: variable, 301, global; {control loop setpoint}
LOOP2:   entity,   302, global; {control loop entity}
LOOP2sp: variable, 303, global; {control loop setpoint}
```

An example runtime display, with the four text objects that you will need to add, is shown below:



DDB_lab.gif

Your text object(s) in the display can use the following expressions:

```
'Text1
Sub OnDataChange()
      set dispdb.loop1sp = lcn.yourtag.sp       'SET buildtime binds
      me.text = dispdb.loop1sp
End Sub


Sub OnLButtonUp()
      dispdb.loop1sp = Askbox("Enter setpoint")    'runtime bind
End Sub



'Text2
Sub OnDataChange()
      set dispdb.loop1 = lcn.yourtag
      me.text = dispdb.loop1              'display tagname
End Sub



'Text3
Sub OnDataChange()
set dispdb.loop2sp = GetVar("yourtag.sp") 'getvar function, buildtime
me.text = dispdb.loop2sp
End Sub



'Text4
Sub OnDataChange()
set dispdb.loop2 = GetEnt("yourtag")  'get entity function, buildtime
me.text = dispdb.loop2
End Sub
```

You may have noticed some unfamiliar functions in our example scripts, GetVar and GetEnt. These represent Honeywell functions that allow you to bind, either at run time or build time, a variable or entity to a display database variable.

Display Building tip: Use the SET operator whenever possible with database variables to bind the variable at build time. The SET operator provides better performance when using display database variables.

Caution: Be careful about how you interpret the GetVar function. It's very easy to incorrectly think of it as one where you are bringing in a new tagname.parameter. It is actually returning the **value** of the new tagname.parameter.

For example, assume you have a code fragment such as

```
dispdb.loop2sp = GetVar(AskBox$("Enter the variable as tagname.sp"))
```

One would first think that you are changing the display database variable reference from one tag.param to another. But, in the code fragment above, you are actually assigning the **value** of another tag's parameter to this dispdb item, which may be indirectly linked to another tag.parameter.

Caution: The usage of the Askbox example as shown in the above example was to illustrate that an operator entry can be made. Because related AskBox examples may appear in documentation, there are several comments to be made about the usage of the Askbox (and InputBox) for process-related entries. ***The Askbox and InputBox may not be your best options for operator process related entries***. One reason is that, unless you add script, the CANCEL button selection on an AskBox returns a 0 if a numeric entry is required, or null string if a text entry is required. Similar behavior occurs with the InputBox. This could mean a runtime error that closes your display, or worse yet, a bad value sent to the process with unpredictable consequences. In other words, if you use the Askbox or Inputbox, you must include error handlers and code that compensates for the cancel button being selected and/or a null string being returned.

Lab Prerequisites

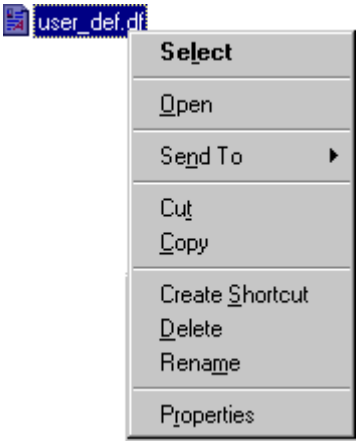
In order to complete the lab exercise, you will need the following:

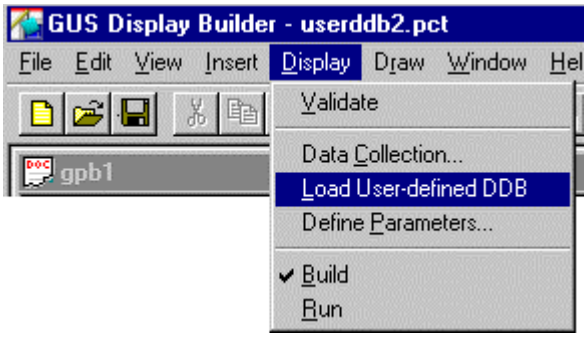
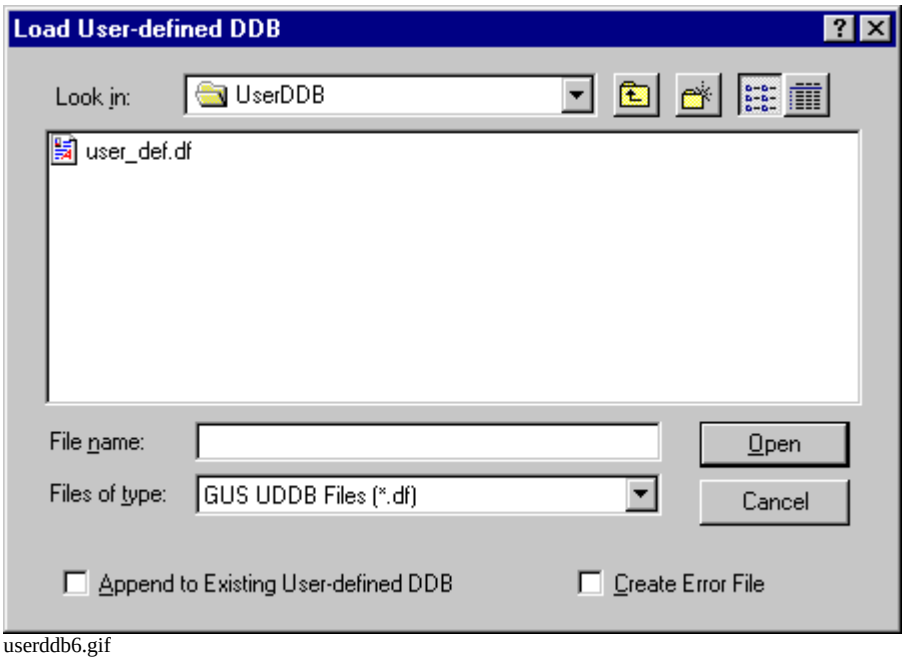
- GUS Display Builder.
- Access to a text editor, such as NotePad.
- Native Window running with access to two off-process regulatory control points.

Lab Procedure

Step	Action

1.	From a text editor, create the user defined text file that contains the following variables:.
	`LOOP1:  entity,  300, global; {control loop entity}`
	`LOOP1sp: variable, 301, global; {control loop setpoint}`
	`LOOP2:  entity,  302, global; {control loop entity}`
	`LOOP2sp: variable, 303, global; {control loop setpoint}`

2.	Save this file with a **.DF** extension as **user_def** in your UserDDBfolder. (Note: You may have to remove the .txt extension from your file and rename your file to have a .df extension.)  userddb4.gif)
3.	Open a new display from the GUS display builder

4.	Load your user-defined database file so that it may be used with this display: (Choose Display>Load User-defined DDB)  userddb5.gif Result: the User-Defined DDB dialog appears.  userddb6.gif
5.	Open the User-Defined DDB .df file that you just created. Expected result: A loaded successful message appears.
6.	Build four text objects in this display.
7.	Refer to the Design Criteria section for adding the script to your text objects.
8.	Syntax check your code.
9.	Validate the display.
10.	Save this display as userDDB.pct into your UserDDB folder.
11.	Run the display.
12.	Click on the text object that requires an entry. Expected results: An input box appears for the setpoint entry.

Last Page