Honeywell

# Lab Exercise – Approach to Bad Status Handling

57310701L
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

Bad status handling is a good programming practice common to almost all scripts, even if it represents nothing more than a branch to a subroutine exit or making a text object change color. Several common error-trapping techniques that are shown in the course material are frequently encountered in “real world” scripts.

## Objectives

Upon completing this lab exercise, you will be able to

- Add an On Error handler to an existing script.
- Test the On Error handler to see if it performs as desired in terms of:
  - ⇒ Changing an object’s color and
  - ⇒ Providing HOPC status.
- Provide a viewport that gives an operator point information.

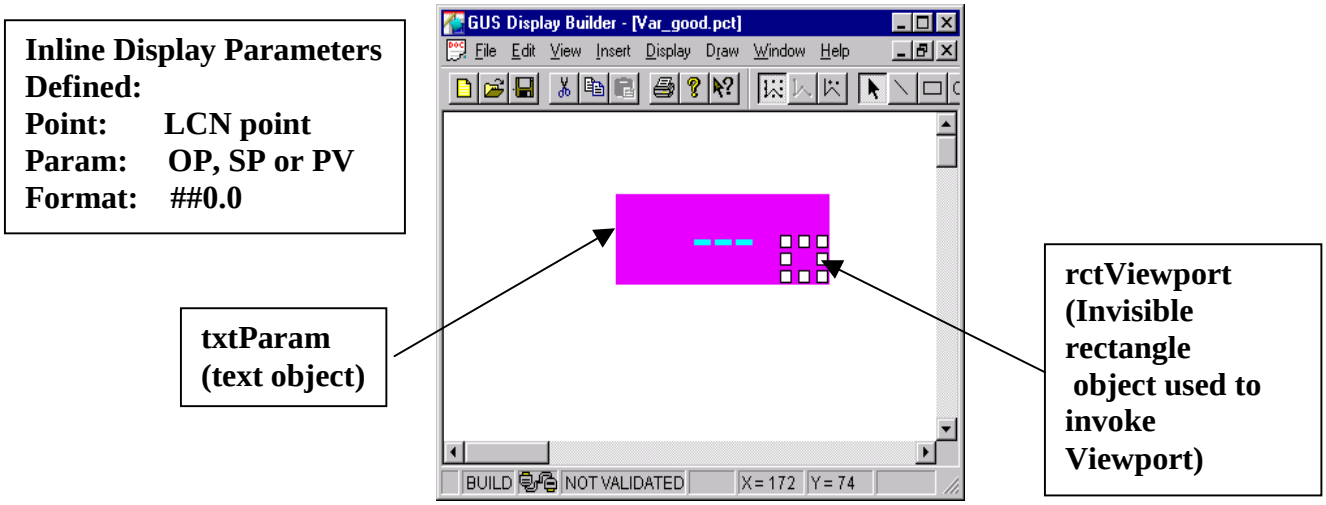
## Design Criteria

The embedded display used in your lab exercise is a relatively simple looking display at first glance, but it has several functions that lend itself to good display building practices. The display is based upon an example display from a customer site. Although you could simply hard code the example as a text object for this lab exercise, it is built as an embedded display to support re-usability for your points. At the end of the lab exercise, you will have an example that you can begin using in your own plant displays.

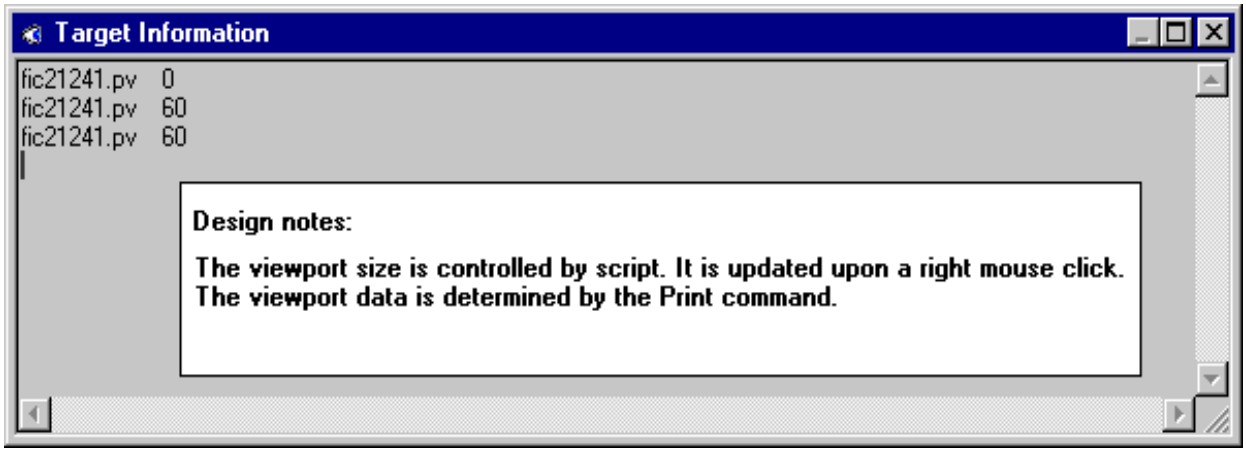
The display requires that you add an On Error handler. The On Error handler will do the following:

- Change the text object fillcolor to magenta (222,0,255)
- Change the text color to black
- Provide the correct HOPC error and bad PV indication

The following figure shows the overall design of the embedded display:




An example viewport is shown below.



Most of the code on this object is already provided for you. The following **boldfaced** code represents an example of what you will enter for the value indicator. The error handler code for the “Lab Exercise” is in **bold**:

```
Sub OnDataChange()
      dim almstat as string

      On Error Goto ODC_error

      If  "-\p(format)-" = "--" Then
            fmt = ""
      else
            fmt ="\p(format)"
      end If

      almstat = collector("ackstat(\pe(point))")

      if almstat = "NOALARM" Then
            me.blink = false
            me.visible = true

            If UCase$("\p(param)") = "OP" Then
                  me.textcolor = makecolor(253,253,128) ‘yellow
                  me.fillcolor = makecolor(0,0,0) ‘black
            Else
                  me.textcolor = makecolor(0,255,255) ‘cyan
                  me.fillcolor = makecolor(0,0,0)     ‘black
            End If
      elseif almstat = "UNAKALRM" Then
            me.textcolor = makecolor(0,255,255) ‘cyan
            me.fillcolor = tdc_red
            me.blink = true
      elseif almstat = "AKDALRM" Then
            me.textcolor = makecolor(0,255,255) ‘cyan
            me.fillcolor = tdc_red
            me.blink = false
            me.visible = true
      end if

      me.text = Format$(point.param,fmt)


Exit Sub
      ODC_error:
            me.text = "---"
            me.fillcolor = makecolor(226,0,255) ‘magenta
            me.textcolor = makecolor(0,0,0)     ‘black
End Sub
```

(Display Building tip: You may have noticed some functions in the provided code that are new to you, such as Ucase$ and Format$. Ucase$ and Format$ are BasicScript formatting functions that present the values in the format that the Honeywell system requires. Using formatting functions in an embedded display assures that another user of this display would not incorrectly format a value when typing the value in as a display parameter. Refer to your Basic Script Language Reference Manual for more details about using these functions.)

After testing your code to see if it works correctly, modify the error handling code so that it provides HOPC status. Example code is listed below:

```
ODC_error:
            select case err.number
                  case 1053
                        me.text = "-----"
                  case HOPC_COMMUNICATION_ERROR
                        me.text = "?????"
                  case HOPC_CONFIGURATION_ERROR
                        me.text = "@@@@@"
                  case else
                        me.text = "!!!!!"
            end select
            me.fillcolor = makecolor(226,0,255)
            me.textcolor = makecolor(0,0,0)
```

A good programming practice is to provide an operator some form of help if there is a problem. One technique is to use a viewport. The viewport itself is non-modal. By right mouse clicking on the invisible rectangle object located in the lower right of the text object displaying the value, the operator receives point status that the display sends the viewport via a Print command. The viewport can be invoked at any time, even when the point status is OK. The following code represents an example of what you can enter for the viewport:

```
'display process info to operator
Sub OnRButtonClick()
      Viewport.Open "Target Information",0,0,600,200
      Print  "\pe(point).\p(param)  " & "  " & txtParam.text
End Sub
```

Display building tip: Display builders often use the right mouse click event to provide the operator some form of help, even if it is a message "No help available for this item." That way, the operator becomes aware that right mouse clicking is available for help, while left mouse clicking is for display and process commands.

## Lab Prerequisites

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

- GUS Display Builder.
- Native Window loaded with access to a process network point.

## Lab Procedure

| Step | Action |
|---|---|
| **1.** | From your ErrorExample folder, open the display called var_dummy.pct. |
| **2.** | Save this display as var_good.pct into your ErrorExample folder. |
| **3.** | Select the object and open a script editor window. |
| **4.** | From the script editor's object browser window, verify that you are in the script window for the object called txtParam. |
| **5.** | Refer to the Design Criteria section, if necessary, and modify the error handling script as shown below (add the bold type to the txtParam object):<br>`Sub OnDataChange()`<br>`  dim almstat as string`<br>**`  On Error Goto ODC_error`**<br>`{rest of the code}`<br>**`Exit Sub`**<br>**`  ODC_error:`**<br>**`  me.text = "---"`**<br>**`  me.fillcolor = makecolor(226,0,255)`**<br>**`  me.textcolor = makecolor(0,0,0)`**<br>`End Sub` |
| **6.** | Syntax check your code. |
| **7.** | From the script editor's object browser window, select the object called rctViewport. |
| **8.** | Refer to the Design Criteria section on the previous page and modify the error handling script for the rctViewport object as shown below:<br>**`'display process info to operator`**<br>**`Sub OnRButtonClick()`**<br>**`Viewport.Open "Target Information",0,0,600,200`**<br>**`Print  "\pe(point).\p(param)  " & " " & txtParam.text`**<br>**`End Sub`** |
| **9.** | Syntax check your code. |
| **10.** | Attempt to validate the display; you will receive validation errors. Note: There are inline parameters defined in this embedded display. You cannot validate an embedded display with inline parameters. |
| **11.** | Save this display as var_good.pct into your ErrorExample folder. |
| **12.** | Open a new display that will contain the var_good.pct. |
| **13.** | Insert your var_good.pct into the new display. (Choose Insert>Display.) |

| Step | Action |
|---|---|
| **14.** | Respond to the embedded display prompts and enter the following parameters:<br>Parameter: **point**<br>(Enter a regulatory control point from your database partition. Example: LCN.FIC21###.)<br>Parameter: **param**<br>Enter: PV<br>Parameter: **format**<br>Enter: ###0.0 |
| **15.** | Validate the display. |
| **16.** | Run the display. |
| **17.** | Open a Native Window detail display for your control point and set up error conditions. Ways to do this include making changes to limits or PVSOURCE in order to make the PV go bad, put the point in MAN mode and set the OP value to –6, or make the point INACTIVE.<br>Expected result: The var_good embedded display should show an error condition. |
| **18.** | Right mouse click on the lower right hand corner of your var_good value indicator.<br>Expected result: A non-modal viewport opens that displays point data. The viewport's point data is only updated when you right mouse click the invisible rectangle that contains the Viewport script. |
| **19.** | The approach of changing a text object's color on an error condition is very common and acceptable in plants. The rationale is that if there is a problem, the operator should then notify someone else. However, you can provide the operator more detailed status by including HOPC status, the subject of the next steps.<br>Close your runtime display that contains var_good.pct. |
| **20.** | Modify your var_good.pct to check for HOPC errors. Your script can be similar to the following:<br>**ODC_error:**<br>**select case err.number**<br>**case 1053**<br>**me.text = "-----"**<br>**case HOPC_COMMUNICATION_ERROR**<br>**me.text = "?????"**<br>**case HOPC_CONFIGURATION_ERROR**<br>**me.text = "@@@@@"**<br>**case else**<br>**me.text = "!!!!!"**<br>**end select**<br>**me.fillcolor = makecolor(226,0,255)**<br>**me.textcolor = makecolor(0,0,0)** |
| **21.** | Syntax check your code. |
| **22.** | Validate the display. (Ignore any errors against inline data types.) |

| Step | Action |
|---|---|
| **23.** | Save this display as var_good2.pct into your ErrorExample folder. |
| **24.** | From the display that contains your regulatory control point, replace the previous var_good.pct display with your var_good2.pct display. (Choose Edit>Replace Embedded display) |
| **25.** | Run the display. |
| **26.** | (Optional step) Setup error conditions to verify that HOPC errors are reported correctly.<br>(Note: To set up some of the error conditions may require course manager intervention. One condition that you can set up, if your time permits, is to reset the Native Window. This will require you to take the time to re-load the Native Window.) |

**Last Page**