

22114 - How To Read and Write Tags in Display Client VBA Code

How Can I Read and Write Tags in FactoryTalk View SE Display Client VBA Code?

Background:

When creating VBA code in the FactoryTalk View SE or RSView SE Client graphics, there may be a need for this code to read or set tag values from the VBA code. The preferred method for accessing tag values is using the Tag Object Model. Additionally, there are two techniques using Display objects that can also be used to read and set these tag values.

1. Using the Tag Object Model in VBA:

The tag group and tag objects in the Tag Object Model can be used in the VBA code to access tags. This method is preferred since it does not require a display object for each tag being accessed or rely on the screen update rate for tag values. The procedure for using this method is 1) Create a tag group, 2) Add tag references to the tag group 3) Set a tag object to the matching tag reference in the tag group 4) Read value of tag object or write value to the tag object.

Sample code is included below. For this code to function as is, the following items are needed:

1. The display hosting the code needs to contain a button named "btnReadWriteTag"
2. The FactoryTalk View SE or RSView SE tag database needs an analog tag named "MyTestTag" contained within a folder named "TestTags"
3. This example also uses a direct reference tag and requires a reference [FTViewSE_VBA] <tagname> where [FTViewSE_VBA] is the RSLinx Enterprise shortcut name or OPC Topic name.

The examples below use a wrapper module for logging informative diagnostic messages that needs to be included in the project. For the examples to function as is, they need to have the module imported into the VBA IDE for any screen using the example code. The module can be found in [AID 54424](#) Sample VBA: Using a Wrapper Module to Log Standard Error Messages.

Option Explicit

'Global declarations

Public MyTagGroup As TagGroup

Private Sub Display_AnimationStart()

'Tag group is created when display is started in order to allow time for tags to be put on scan

'On Error GoTo ErrorTrap

'Check to ensure that the tag group doesn't already exist

If MyTagGroup Is Nothing Then

'Create a tag group based on the Area that the Display is located in

Set MyTagGroup = Application.CreateTagGroup(Me.AreaName)

'Add tags to tag group

'HMI Tag Example 1 – system tag

MyTagGroup.Add "system\Second"

'HMI Tag Example 2 – HMI device tag

MyTagGroup.Add "TestTags\MyTestTag" 'Note: this is a HMI device tag

```

'Direct Referencing Example where [FTViewSE_VBA] is the OPC topic name
MyTagGroup.Add "{[FTViewSE_VBA]TestDirRefTag}"
End If

Exit Sub

ErrorTrap:
'Log a diagnostic message using the SystemLogModule.bas wrapper module
WriteSystemLog ftDiagAudienceOperator, ftDiagSeverityError, Me.Name,
Me.ActiveElement.Name, "Error Creating Tag Group", Err.Number, Err.Description
Set MyTagGroup = Nothing
End Sub

Private Sub btnReadWriteTag_Released()
'Local declarations
Dim SystemSecondTag As Tag
Dim HMITestTag As Tag
Dim DirRefTag As Tag

Dim SystemSecondTagValue As Integer
Dim SystemSecondTagHighEU As Integer
Dim SystemSecondTagLowEU As Integer
Dim ErrorCatch As Integer ' Used to determine where error occurred

On Error GoTo ErrorTrap
ErrorCatch = 1

'Check to see if tag group was succesfully created on display start
If MyTagGroup Is Nothing Then
WriteSystemLog ftDiagAudienceOperator, ftDiagSeverityError, Me.Name,
Me.ActiveElement.Name, "Tag Group Not Created on Display Start", Err.Number,
Err.Description
Exit Sub
End If

***** Tag Read Example *****
'Note: assignment of tag objects could have been done in the above If clause: the
TagGroup add method
'returns the added tag (as shown for the direct ref tag example above)

'Set the tag variable to the tag group entry
Set SystemSecondTag = MyTagGroup.Item("system\Second")
'Read tag value and set it to a variable value for later VBA use
SystemSecondTagValue = SystemSecondTag.Value
'Additional tag properties, like engineering units, can also be read from the tag object
SystemSecondTagHighEU = SystemSecondTag.GetProperty(tagPropIdHighEU)
SystemSecondTagLowEU = SystemSecondTag.GetProperty(tagPropIdLowEU)

ErrorCatch = 2
***** Tag Write Example *****
'Set the tag variable to the tag group entry
Set HMITestTag = MyTagGroup.Item("TestTags\MyTestTag")

'Add one to current tag value and write back to tag
HMITestTag.Value = HMITestTag.Value + 1

'This example increments a Direct Reference Tag
Set DirRefTag = MyTagGroup.Item("{[FTViewSE_VBA]TestDirRefTag}")

'Add one to current tag value and write back to tag
DirRefTag.Value = DirRefTag.Value + 1

```

```

'Destroy tag object and tag group
Set SystemSecondTag = Nothing
Set HMITestTag = Nothing
Set DirRefTag = Nothing

Exit Sub

ErrorTrap:
'If error occurs, sends information to the Diagnostics log to help
' you debug errors and notify the operator of problems with VBA code
'Error message based on where error occurred using ErrorCatch Variable

'Log a diagnostic message using the SystemLogModule.bas wrapper module  Select
Case ErrorCatch
    Case 1 ' Error occurred during tag tag read example
        WriteSystemLog ftDiagAudienceOperator, ftDiagSeverityError, Me.Name,
Me.ActiveElement.Name, "Error Reading Tags", Err.Number, Err.Description
    Case 2 ' Error occurred during tag write example
        WriteSystemLog ftDiagAudienceOperator, ftDiagSeverityError, Me.Name,
Me.ActiveElement.Name, "Error Writing Tags", Err.Number, Err.Description
    End Select
Resume Next

End Sub

Private Sub Display_BeforeAnimationStop()

'Destroy tag group
Set MyTagGroup = Nothing

End Sub

```

2. Numeric or String Display Objects:

The numeric or string display object can be used to provide a tag or expression value to VBA by reading the object's Value property. The following list tips and points to be aware of when using this method:

- To use this technique, there would need to be one display object per tag or expression required
- The numeric or string display object can be hidden at runtime by setting the object's Visible property to False in the Property Panel
- The update rate of these objects is defined in the graphic display update rate setting
- If the VBA Code needs to react to changes in the tag or expression value, code can be added to the object's Changed() event
- If the numeric or string display object is wire-framed and the Value property is read, the property will return an error variant. See the RSView SE VBA help for more information

Code sample from the RSView SE VBA Help for using the Numeric Input object.

```

' Sample code to read the numeric display object's value into VBA code.
Private Sub NumericDisplay1_Change()
' Check that the expression or tag is valid
If IsError(NumericDisplay1.Value) Then
Application.LogMessage "NumericDisplay1 is in error state."
Else
' Here is where the code would use this value... for the sample,
' just report it to the activity log
Application.LogMessage "NumericDisplay1.Value = " & NumericDisplay1.Value
End If
End Sub

```

3. Numeric or String Input Objects:

VBA code can use the numeric or string input objects to read and set tag values by reading the input object's Value property, writing back to the Value property, and then calling the Download method of the input object.

- This will require one input object per tag or expression required on the display
- It is recommended that the input object be set to continuously update. If the checkbox "Continuously update" is not set, then the Upload method needs to be called before reading the Value property
- Ensure the "Discard input and resume updating when focus is lost" checkbox is not set
- The numeric or string input object can be hidden at runtime by setting the object's Visible property to False in the Property Panel. NOTE: The Upload and Download methods require that the input object is visible when they are called. The code example below shows how this can be achieved.
- The update rate of the object is defined in the graphic display update rate setting
- If the VBA Code needs to react to changes to a tag value, code can be added to the object's Changed() event
- If the numeric or string display object is wire-framed and the Value property is read, the property will return an error variant. See the RSView SE VBA help for more information

' Sample code to read the numeric input field's value into VBA code.

```
Sub ReadNumericInputValue()
```

```
' If the input is not set to be "continuously updating", uncomment the following If-Then-Else-EndIf code
```

```
'If NumericInput1.Visible = False Then
```

```
'NumericInput1.Visible = True
```

```
'NumericInput1.Upload
```

```
'NumericInput1.Visible = False
```

```
'Else
```

```
'NumericInput1.Upload
```

```
'End If
```

```
' Read the Numeric input's Value
```

```
If IsError(NumericInput1.Value) Then
```

```
Application.LogMessage "NumericInput1 is in error state."
```

```
Else
```

```
' Here is where the code would use this value... for the sample,
```

```
' just report it to the activity log
```

```
Application.LogMessage "NumericInput1.Value = " & NumericInput1.Value
```

```
End if
```

```
End Sub
```

```
' Set the Numeric Input field value and then set the tag value.
```

```
,
```

```
Sub SetNumericInputValue(vValue As Variant)
```

```
Dim blsVisible As Boolean
```

```
' Set up Error handler
```

```
On Error GoTo Errhandler
```

```
blsVisible = NumericInput1.Visible ' Keep track of visible state
```

```
NumericInput1.Value = vValue ' Set the Input Value
```

```
If blsVisible = False Then
```

```
' Element is not visible, make visible and download the value
```

```
NumericInput1.Visible = True ' Make the Input visible
```

```
NumericInput1.Download ' Download the value to the Tag
```

```
NumericInput1.Visible = False ' Make the Input invisible.
```

```
Else
```

```
' already visible, just download the value
NumericInput1.Download      ' Download the value to the Tag
End If
Exit Sub
```

```
Errhandler:
LogMessage "Unable to set Value for Numeric Input: " & NumericInput1.Name,
cliActivitySeverityError, cliActivityCategoryCommunication
End Sub
```

Details

Answer ID 22114
Products Software
 Performance and Visualization (HMI)
 FactoryTalk View SE
Categories General
Date Created 06/14/2002 12:00 AM
Date Updated 04/20/2010 11:17 AM
Prev. TN# A12142740

DISCLAIMER

This knowledge base web site is intended to provide general technical information on a particular subject or subjects and is not an exhaustive treatment of such subjects. Accordingly, the information in this web site is not intended to constitute application, design, software or other professional engineering advice or services. Before making any decision or taking any action, which might affect your equipment, you should consult a qualified professional advisor.

ROCKWELL AUTOMATION DOES NOT WARRANT THE COMPLETENESS, TIMELINESS OR ACCURACY OF ANY OF THE DATA CONTAINED IN THIS WEB SITE AND MAY MAKE CHANGES THERETO AT ANY TIME IN ITS SOLE DISCRETION WITHOUT NOTICE. FURTHER, ALL INFORMATION CONVEYED HEREBY IS PROVIDED TO USERS "AS IS." IN NO EVENT SHALL ROCKWELL BE LIABLE FOR ANY DAMAGES OF ANY KIND INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, LOSS PROFIT OR DAMAGE, EVEN IF ROCKWELL AUTOMATION HAVE BEEN ADVISED ON THE POSSIBILITY OF SUCH DAMAGES.

ROCKWELL AUTOMATION DISCLAIMS ALL WARRANTIES WHETHER EXPRESSED OR IMPLIED IN RESPECT OF THE INFORMATION (INCLUDING SOFTWARE) PROVIDED HEREBY, INCLUDING THE IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, AND NON-INFRINGEMENT. Note that certain jurisdictions do not countenance the exclusion of implied warranties; thus, this disclaimer may not apply to you.