

Mint^{MT} Multi-Tasking
Application Note**AN00138-001 - Interfacing RSView® with Mint Controllers****Overview**

ActiveX controls, formerly known as OLE controls or OCX controls, are components (or objects) that can be inserted into an application to reuse packaged functionality. For example, the ActiveX controls that are included with Mint Workbench allow PC applications to communicate with controllers to allow complete machine control from a PC.

A key advantage of ActiveX controls is that they can be used in applications written in many programming languages, including development environments such as:

- Microsoft Visual C++
- Microsoft Visual Basic
- Borland Delphi
- National Instruments LabView
- Microsoft Visual Basic for Applications (VBA)
- **Rockwell Software's RSView® SCADA package**

VBA provides the platform for simple, easy to use connectivity between these applications and Mint controllers.

The Mint ActiveX Control is installed as part of Workbench v5.5 (freely downloadable from www.baldormotion.com) and is suitable for use under Windows 95, 98, NT, ME, 2000, XP and Vista (Note: USB support is only provided under Win2000, WinXP and Vista).

For the purposes of this Application Note we will illustrate how VBA can be used within RSView® to communicate with (and control) a Nextmove (ESB/ES/ST/e100) via an USB connection. The same principles apply to all Mint controllers (we will highlight specific differences throughout the text – for example, how to connect to a serial based controller such as Mintdrive^{II}).

It may also be useful to make reference to Application Note AN00135 – Communicating to Mint Controllers from VBA.

Supported Controllers

NextMovePCI	<input checked="" type="checkbox"/>
NextMoveBX^{II}	<input checked="" type="checkbox"/>
NextMoveST	<input checked="" type="checkbox"/>
NextMoveES	<input checked="" type="checkbox"/>
NextMoveESB	<input checked="" type="checkbox"/>
NextMoveE100	<input checked="" type="checkbox"/>
MintDrive^{II}	<input checked="" type="checkbox"/>
Flex+Drive^{II}	<input checked="" type="checkbox"/>
MicroFlex	<input checked="" type="checkbox"/>

Mint^{MT} Application Note

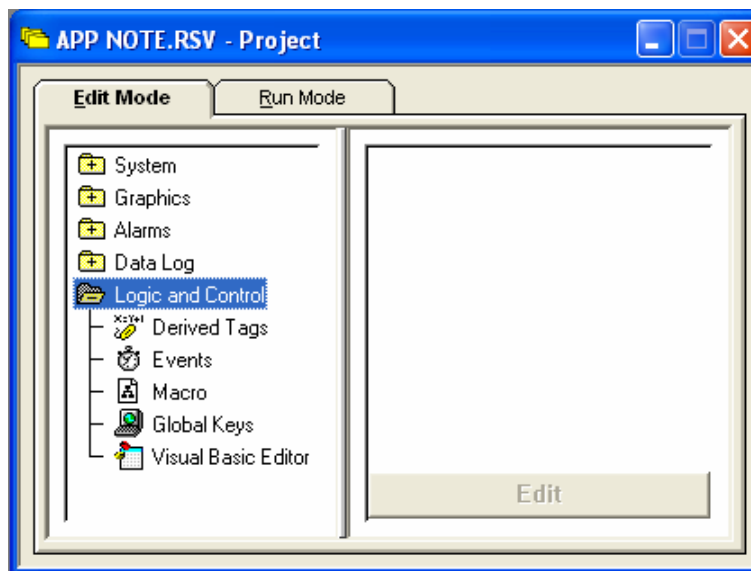
Getting Started

If you intend to follow the practical example in this Application Note you will need the following:

- A PC running Windows 2000/XP/Vista with a USB connection
- A copy of Rockwell Software's RSView32 Works ® (Version 7.2.00 onwards)
- A copy of Mint Workbench v5.5 (Build 5560 onward)
- A Nextmove motion controller with USB connectivity (e.g. ESB/ES/ST/e100). It is suggested that you initially setup a virtual axis as Axis 0 with Mint Workbench v5.5. Make a note of the controller's USB node address (e.g. Node 2) and the Mint SCALEFACTOR that has been set for the axis (e.g. 4000)

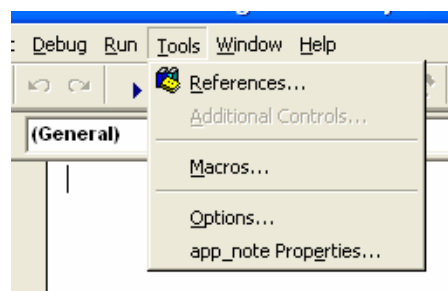
Start RSView32 Works ® and start a new project. The first thing we must do is include a reference to the Mint ActiveX control in the RSView® project.

On the 'Edit Mode' tab of the RSView® project explorer double-click the 'Logic and Control' folder...



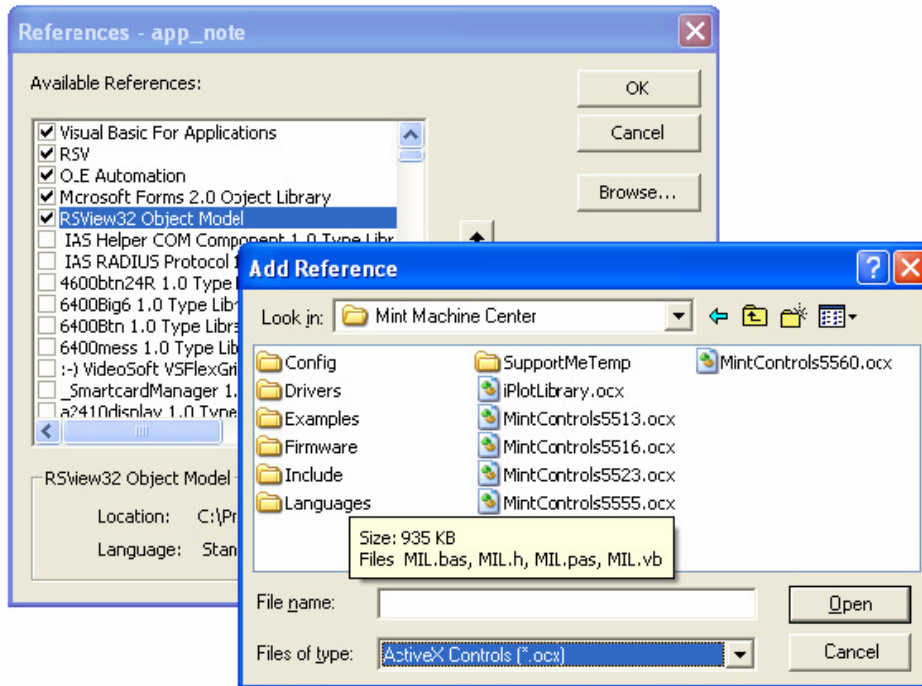
This reveals a number of options including the 'Visual Basic Editor' – double click this to open the VBA programming environment.

From the VBA menu select Tools>References...



Mint^{MT} Multi-Tasking Application Note

Now click on the Browse button, navigate to the Workbench install directory (e.g. C:\Program Files\Mint Machine Center) and select 'ActiveX Controls (*.ocx)' from the 'Files of type' dropdown...



Select the latest ActiveX version (in our case we used MintControls5560.ocx) and click the 'Open' button.

MintControls Build 5560 will now appear in the list of project references – Click OK.

You will now be back in the editor window for the 'General Declarations' section of the 'ThisProject' RSView® object. We can now create an instance of a Mintcontroller object using the following two lines of code...

```
Public WithEvents Mintcontroller As MintControls5560Lib.Mintcontroller  
Dim NMESB As New Mintcontroller
```

We were using a NextmoveESB so we called our object NMESB. It's a good idea to keep to a short name as this will be typed a lot in a large project!

Also add the following variable declaration...

```
Dim bControllerOK As Boolean
```

We will use this later to indicate to the rest of the code whether or not the controller has been connected successfully or not.

Mint^{Multi-Tasking} MT Application Note

Establishing Communication

Before the ActiveX control can be used to issue any Mint commands we need to establish a communications link between RSView® and the Mint controller. When our RSView® project starts (i.e. enters Run Mode) it would be good if it could automatically establish this connection. To do this we will create a VBA subroutine (again in the General section of the 'ThisProject' object) as follows...

```

Sub Init()
  On Error GoTo ErrHandler
  Dim IBuild As Long
  NMESB.SetUSBControllerLink 2
  IBuild = NMESB.AAABuild
  bControllerOK = True
Exit Sub

ErrHandler:
  bControllerOK = False
  MsgBox "Failed to Connect to Mint Controller"

End Sub

```

This code configures a USB connection to Node 2 and then checks that the controller is present by reading back the firmware build version. If it fails to communicate correctly then a Windows Messagebox informs the user. The variable bControllerOK then indicates (True/False) whether the controller is connected.

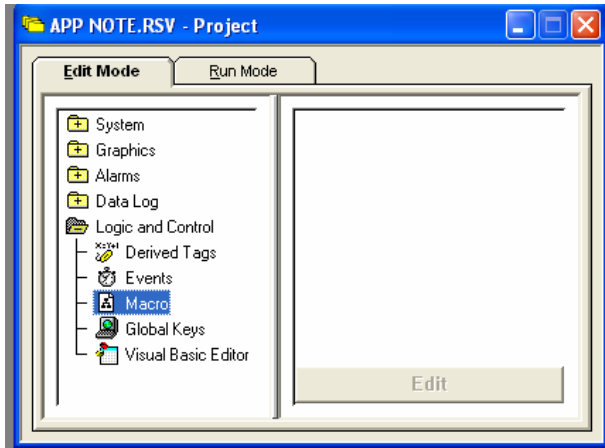
The table below details which ActiveX method should be selected to connect to a particular Mint controller...

Controller	Connection Type	ActiveX Method
MintdriveII	Serial (RS232/RS422)	setMintDrive2Link
FlexPlusDriveII	Serial (RS232/RS422)	setFlexPlusDrive2Link
NextMoveBXII	Serial (RS232/RS422)	setNextMoveBX2Link
NextMoveESB	Serial (RS232/RS422)	setNextMoveESBLink
NextMoveESB	USB	setUSBControllerLink
NextMoveST	Serial (RS232/RS422)	setNextMoveSTLink
NextMoveST	USB	setUSBControllerLink
NextMoveES	Serial (RS232/RS422)	setNextMoveESLink
NextMoveES	USB	setUSBControllerLink
NextMovePCI	Dual Port Ram	setNextMovePCILink
NextMove e100	Serial (RS232/RS422)	setNextMoveE100Link
NextMove e100	USB	setUSBControllerLink
NextMove e100	Ethernet	setEthernetControllerLink

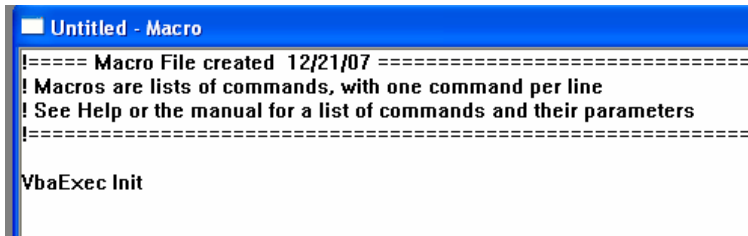
Mint^{MT} Multi-Tasking Application Note

We now need to setup RSVIEW® to automatically call our VBA Init subroutine on startup.

To do this double-click the 'Macro' icon in the Edit Mode project explorer window...

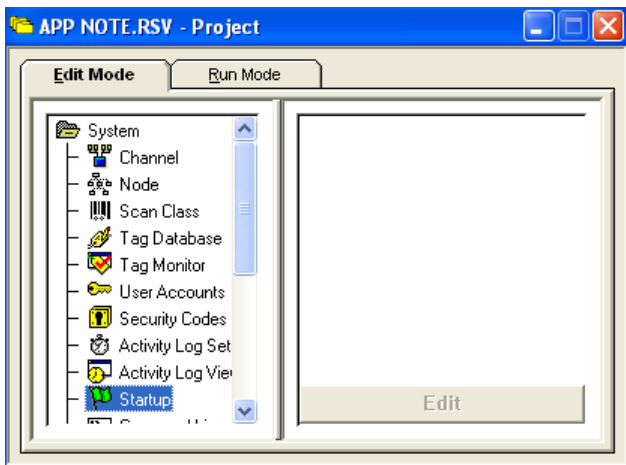


This opens RSVIEW's macro editing window. Here we need to use the VbaExec command to call our VBA subroutine. Enter the following code into the Macro editor...



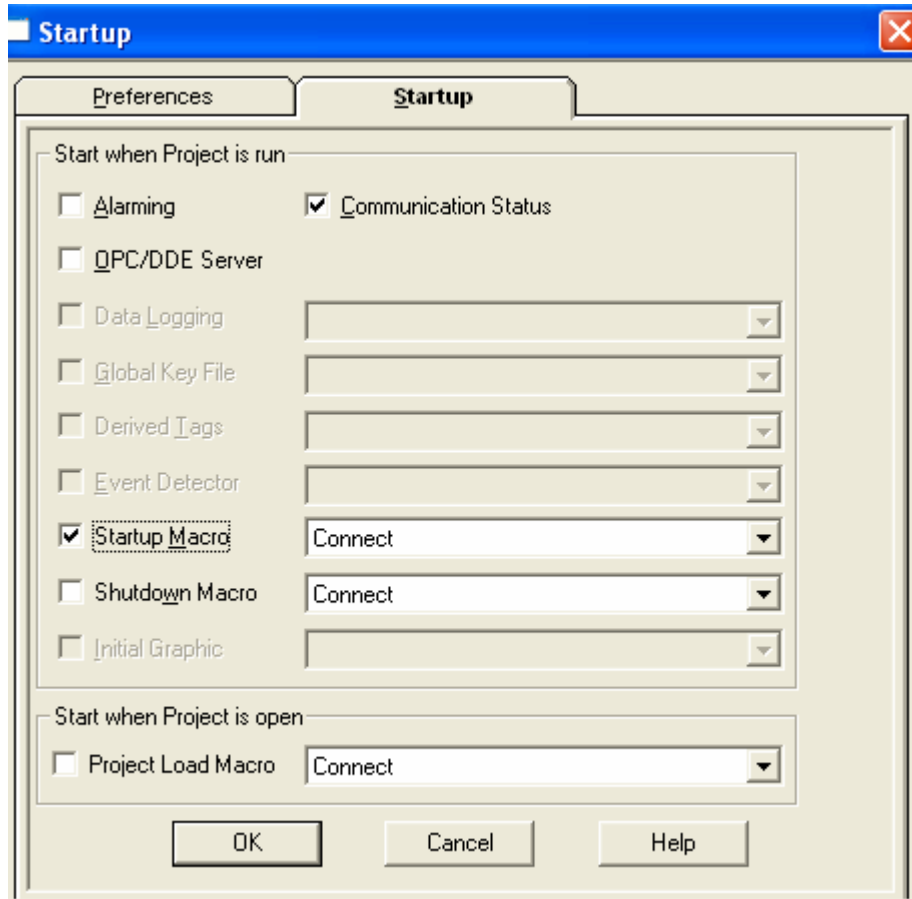
Close this window, RSVIEW® will ask you to save the macro (we called ours Connect).

In the RSVIEW® project explorer double-click the System icon and then double-click the Startup icon....



Mint^{MT} Multi-Tasking Application Note

This opens a new dialog with two tabs (Preferences and Startup). Select the Startup tab and then tick the 'Startup Macro' checkbox to show we want to run the Connect macro we just created ...the dropdown box to the right allows us to select from all available macros but at this stage we have only created the one macro.



Click OK to save these settings. We are now ready to test the connection to the controller and the operation of our macro and VBA code.

Switch to the 'Run Mode' tab on the RStudio® project explorer and click the 'Start Project' button

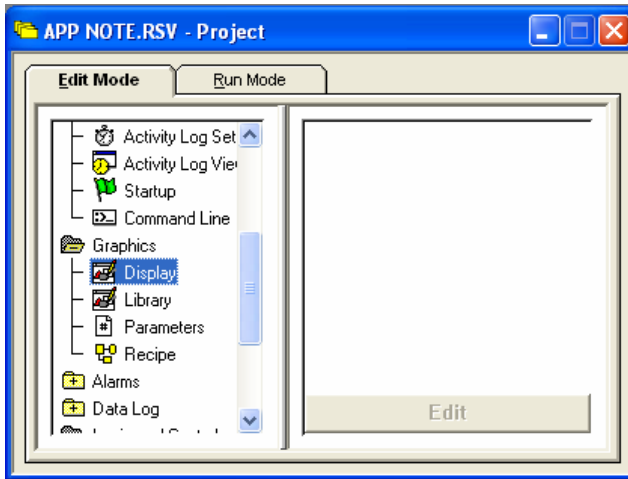
If a Windows message box appears stating the connection failed then you need to check the physical connection to the controller, check the baud rate etc.. (if using a serial controller) and reread this Application Note to ensure you have followed the instructions correctly. Otherwise, if no message appears then everything is working and it's time to create a screen to display some data and provide a means of sending commands to the controller – Click on the 'Stop Project' button and return to the 'Edit Mode' tab.

Mint^{MT} Multi-Tasking Application Note

Creating a RSVIEW® Screen

Now we've established a connection we can consider how to read and write data. For the purposes of this application note we'll add a command button (to issue a relative move) and a numeric label (to display axis position) to a screen.

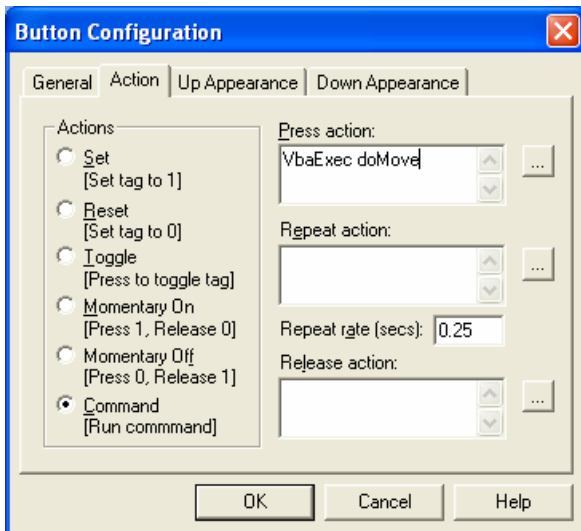
Double-click the 'Graphics' icon in the RSVIEW® project explorer and then double-click the 'Display' icon...



RSVIEW® will now display a blank template for a new screen. Select the 'button' tool from the toolbar and click and drag on the screen to draw a button...



Having drawn the button RSVIEW® will display the button configuration dialog. Configure the button as shown below...



Mint^{MT} Multi-Tasking Application Note

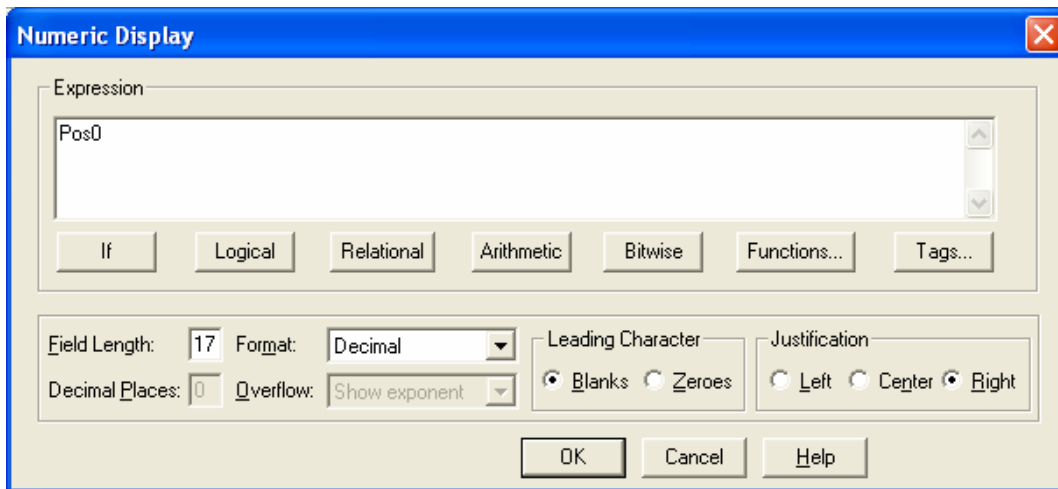
This sets up the button to call a VBA subroutine called doMove whenever the button is pressed (we will write this routine in a moment).

If you like you can also use the 'Up Appearance' and 'Down Appearance' tabs to configure a caption for the button (we added the text 'MOVE' to each).

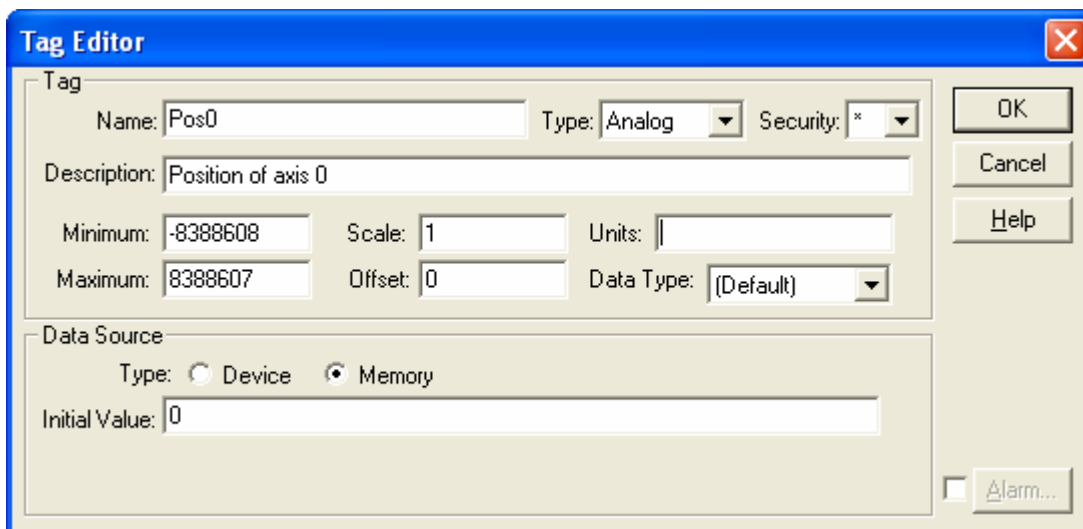
Now select the Numeric display toolbar button and draw a box for a numeric label on the screen...



RSView® will now display a dialog to allow you to setup this numeric display. In our case we want it to display the value of a RSView® tag called 'Pos0' (we will create this tag in a moment) so edit the dialog so it looks like this...

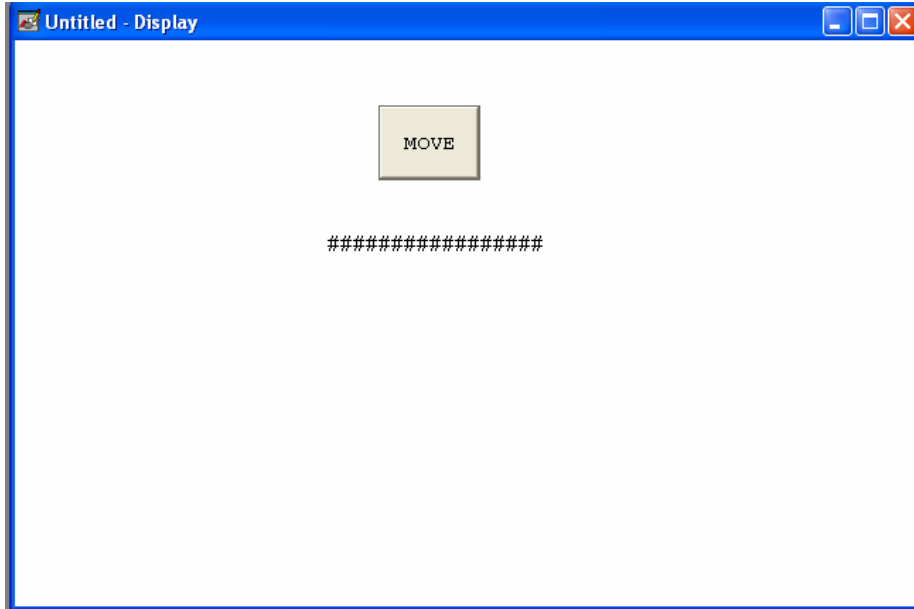


Click OK and RSView® will detect that the tag called Pos0 does not exist. Say Yes to create it now...



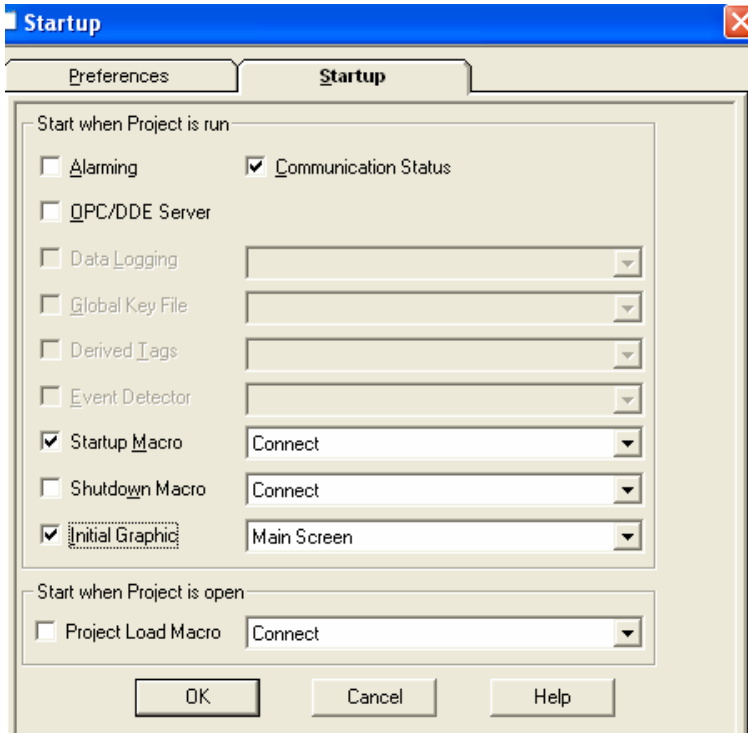
Mint^{MT} Multi-Tasking Application Note

Your finished screen should look something like this...



Close the window and save the screen when prompted (we called ours 'Main Screen'). RSVIEW® will now show this in the list of available screens.

Now select the Startup icon in the RSVIEW® Project Explorer again and now setup our new screen as the Initial Graphic (on the Startup tab)...



Mint^{Multi-Tasking} MT Application Note

Before we look at the code required to display the axis position we'll now create the VBA subroutine to issue the Move command (i.e. the doMove routine we programmed for the button). Either use ALT+TAB to switch back to the VBA editor or double-click 'Visual Basic Editor' in the RSVIEW® Project Explorer and then enter the following code...

```
Sub doMove()  
  On Error GoTo ErrHandler  
  If bControllerOK Then  
    If NMESB.DriveEnable(0) = False Then  
      NMESB.DriveEnable(0) = True  
      NMESB.DoWait (100)  
    End If  
    NMESB.MoveR(0) = 100  
    NMESB.DoGo1 (0)  
  End If  
  Exit Sub  
  
ErrHandler:  
  MsgBox "Error issuing move : " & Err.Description  
End Sub
```

This code checks if axis 0 on the controller is enabled (and enables it if it isn't) and then issues a relative move of 100 user units.

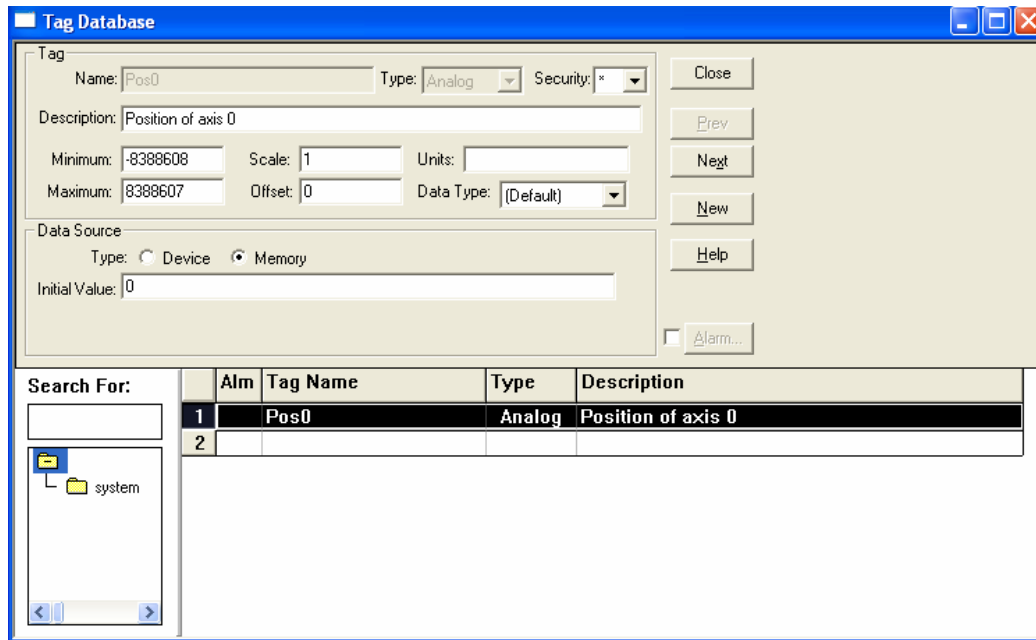
Although we haven't written any code to display position yet we could at this stage run the Project and check that the axis moves when we click on the button (Start Mint Workbench and use the Axis or Monitor tabs in the Spy Window to check the axis is moving...unless you're using a real axis of course in which case it should be a little more obvious!!).

Mint^{MT} Multi-Tasking Application Note

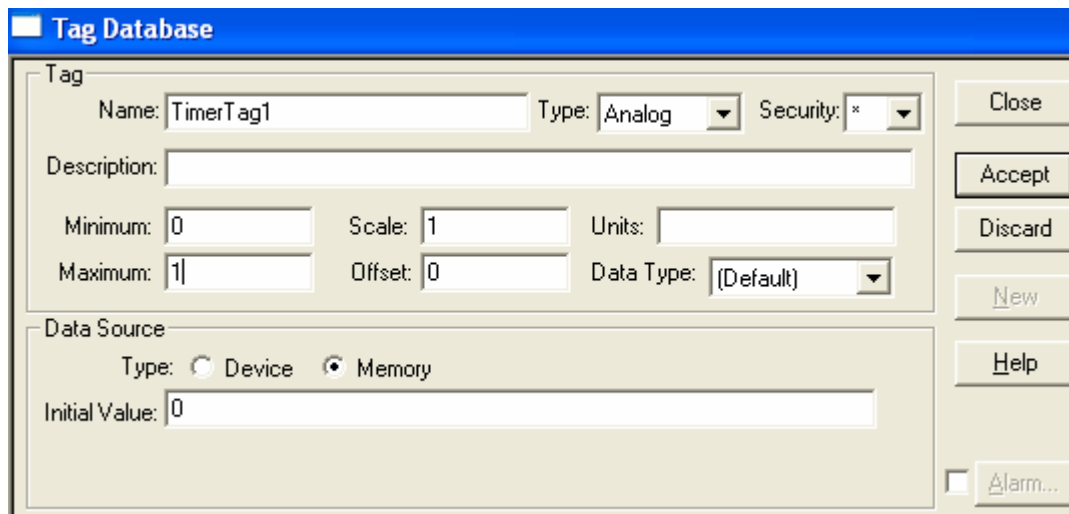
Displaying Data

RSView® (and VBA) does not provide a timer object (like that found in Visual Basic for example) so we must use RSView's 'Derived Tags' and 'Events' features to create our own timer mechanism.

Firstly double-click 'Tag Database' in the Project Explorer....the resulting dialog will show the Pos0 tag we created earlier...



Click the 'New' button and create a new tag called TimerTag1 as follows...

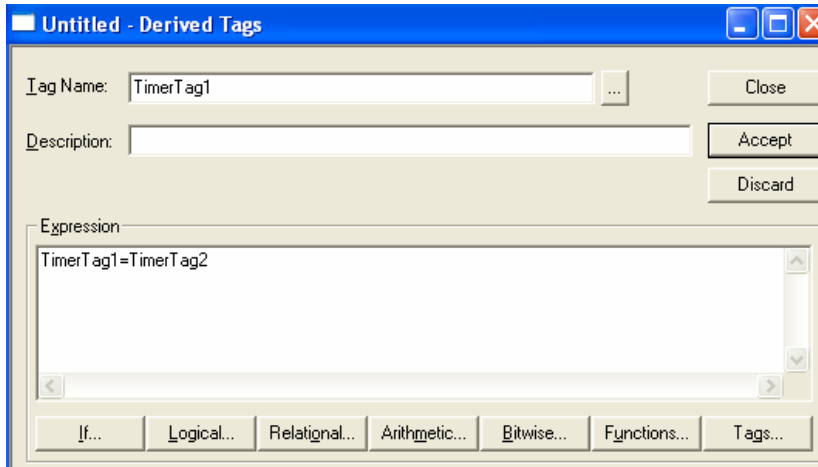


Mint^{MT} Multi-Tasking Application Note

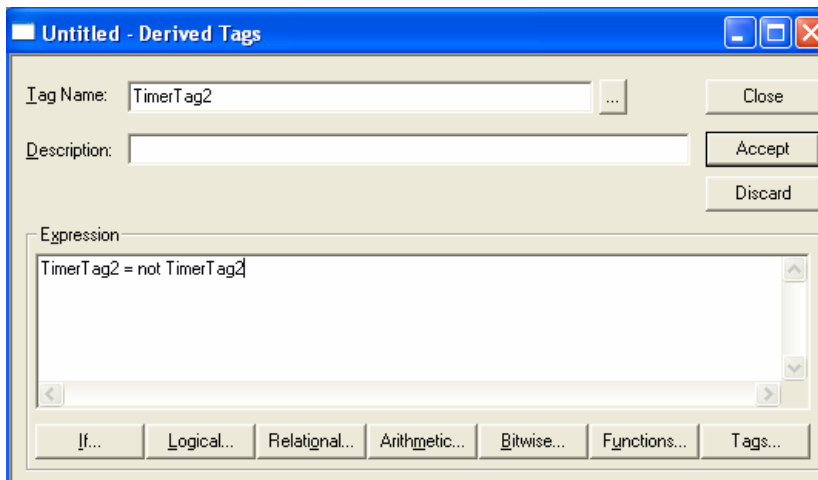
Now click on 'Accept' and then create another tag called TimerTag2 (the setup of this is identical to TimerTag1 but should have an initial value of 1 instead). Accept the second tag and close the dialog.

Now double-click 'Derived Tags' in the RSView® Project Explorer...

Enter TimerTag1 as the derived tag name and complete the expression as shown below...



Accept this, click on the second row in the Derived Tags table and then enter the following to define TimerTag2...

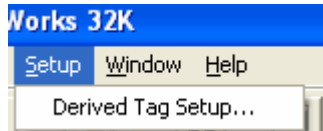


Accept this too and you should end up with two derived tags as shown below...

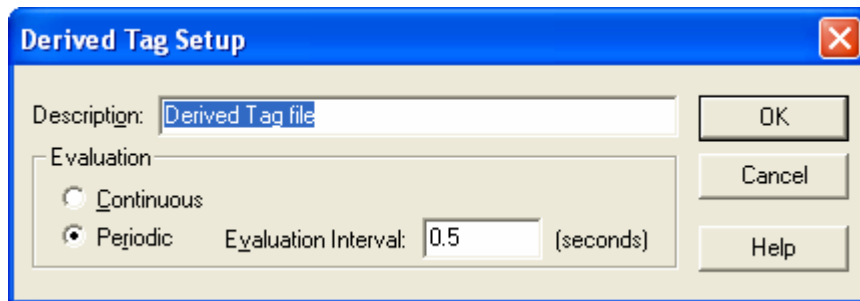
	Derived Tag Name	Expression	Description
1	TimerTag1	TimerTag1 = TimerTag2	
2	TimerTag2	TimerTag2 = not TimerTag2	

Whilst the derived tag setup dialog is still open select Setup>Derived Tag Setup... from the main menu

Mint^{MT} Multi-Tasking Application Note



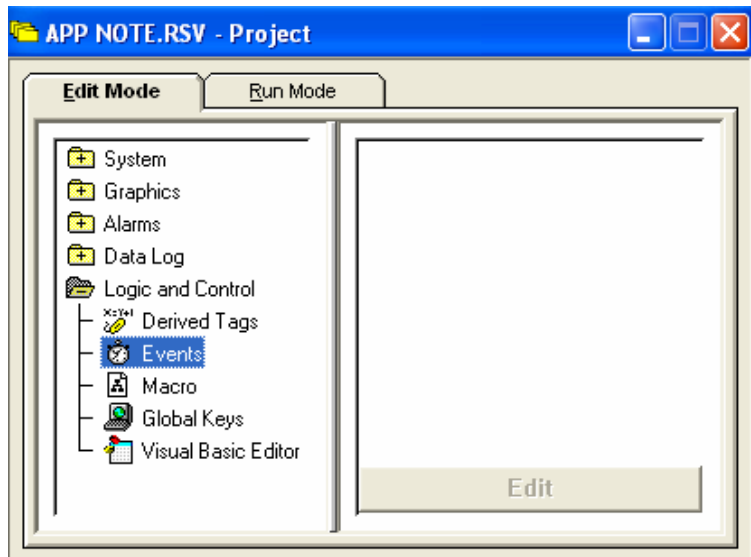
Ensure this particular tag file (which just contains our two timer tags) is setup as 'Periodic' and enter an appropriate evaluation interval (the display will actually update at double this time so we entered 0.5 seconds to allow the display to update once every second).



Close the Derived Tag setup window and RSVIEW® will ask you to save this and name the DTS file (we called ours Timer).

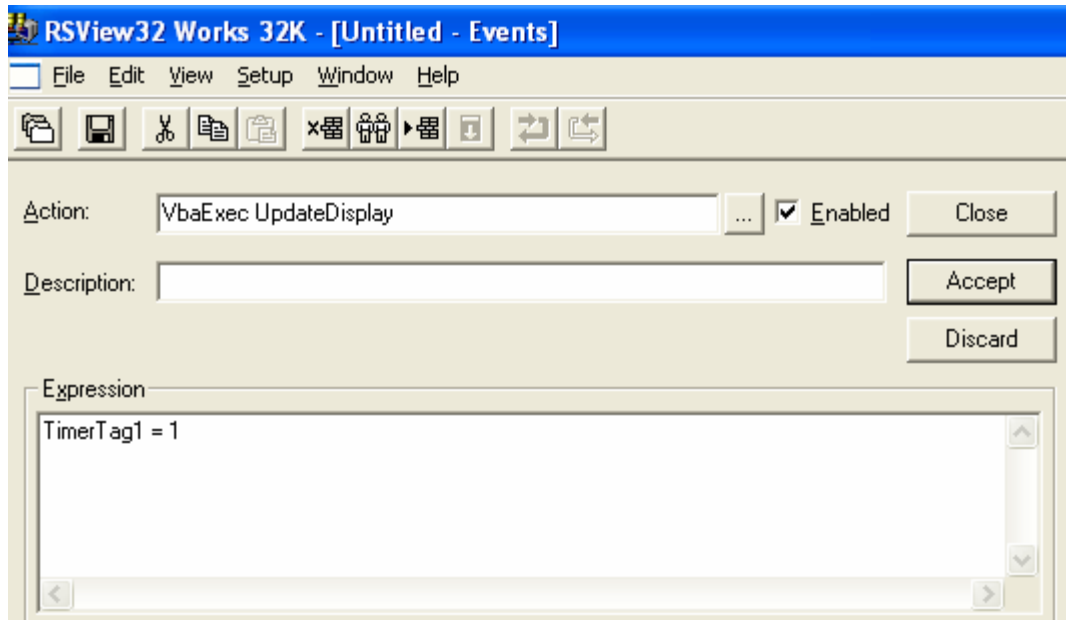
Now we need to create a RSVIEW® Event – in this case we want to create an Event whenever our TimerTag1 tag goes from 0 to 1...

Double-click the 'Events' icon in the RSVIEW® Project Explorer...

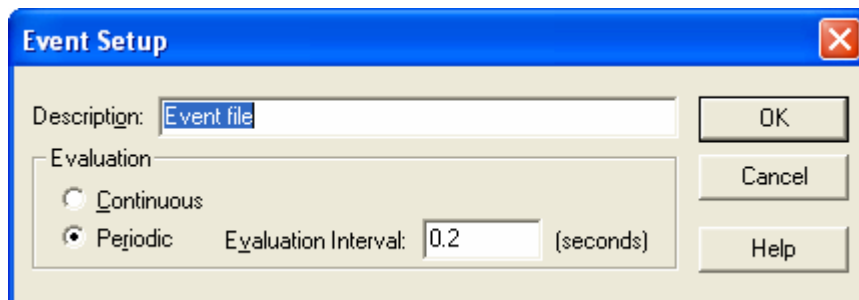


The following dialog shows how to setup an event that runs a VBA routine called UpdateDisplay (which we will add to our VBA code later) whenever Timertag1 switches from 0 to 1...

Mint^{MT} Application Note

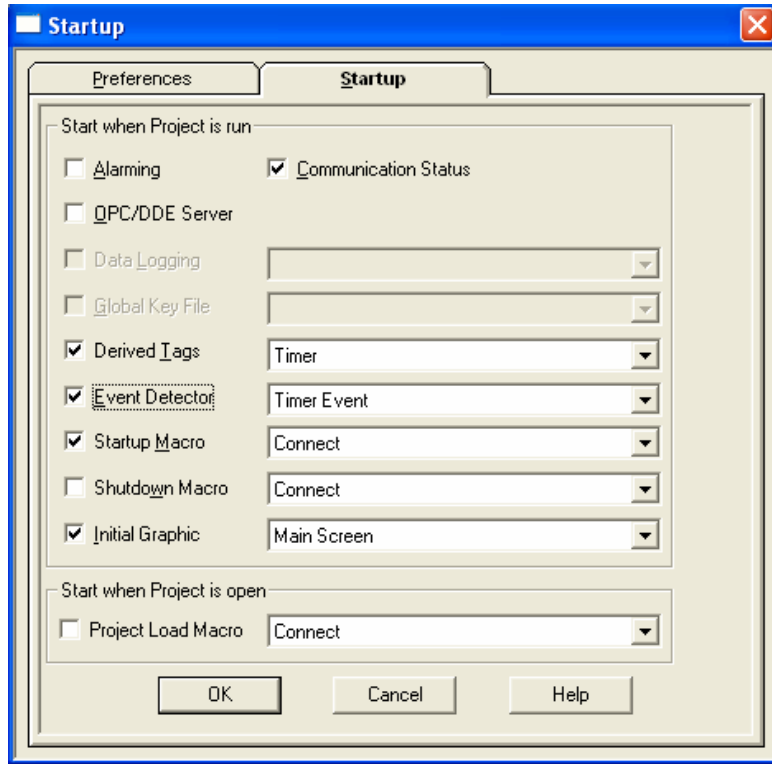


Accept this and then select the Setup>Event Setup... menu. We need to run the event detection at least twice the rate the derived tags are evaluated to ensure we don't miss the 0->1 transition of our TimerTag1 so set the evaluation period to 0.2 seconds...



Click OK and then close the Event setup dialog and save the resulting EDS file (we called ours Timer Event).

We now have to configure RSView® so it starts our Derived Tag file and Event detection file on Startup. Double-click the Startup icon in RSView's Project Explorer again and setup the Startup tab as shown below...

Mint^{MT} Multi-Tasking Application Note

Now we need to create the UpdateDisplay VBA subroutine. This code will read the position of Axis 0 from the Mint controller and assign this to our 'Pos0' RSVIEW® tag. Switch to the VBA editor again.

Firstly we need to create a tag variable so add this code below our other declarations...

Dim Pos0 As Tag

Now add the following subroutine to the VBA code...

```
Sub UpdateDisplay()  
  If bControllerOK = True Then  
    Set Pos0 = gTagDb.GetTag("Pos0")  
    Pos0.Value = NMESB.Pos(0)  
  End If  
End Sub
```

We are now ready to Run the project and test the full operation – Start the project, click the Move button and if you've followed these instructions correctly your axis will move and the label will show the current axis position.

RSView is a registered trademark of Rockwell Automation, Inc