

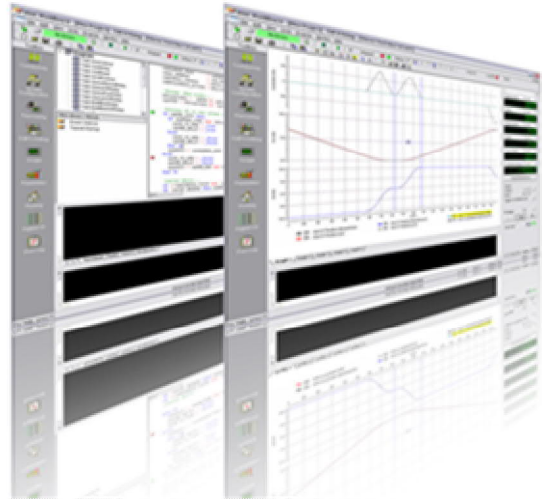
# Application note

## Using Mint ActiveX controls with VBA

AN00135

Rev A (EN)

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. The ActiveX controls that are included with Mint Workbench allow PC applications to communicate with controllers to allow complete machine control from a PC



### Introduction

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
- Microsoft Visual C#
- Borland Delphi
- National Instruments LabView
- **Microsoft Visual Basic for Applications (VBA)**

As VBA is included with a large number of third party applications (Microsoft Excel and Rockwell Software's RSView are examples of these) this provides the platform for simple, easy to use connectivity between these applications and Mint controllers.

The Mint ActiveX controls are installed as part of Mint Workbench (freely downloadable from [new.abb.com/motion](http://new.abb.com/motion)) and are suitable for all versions of Windows (depending on the version of Workbench installed – see the support website for further details). Note: USB support is only provided under Win2000 onwards).

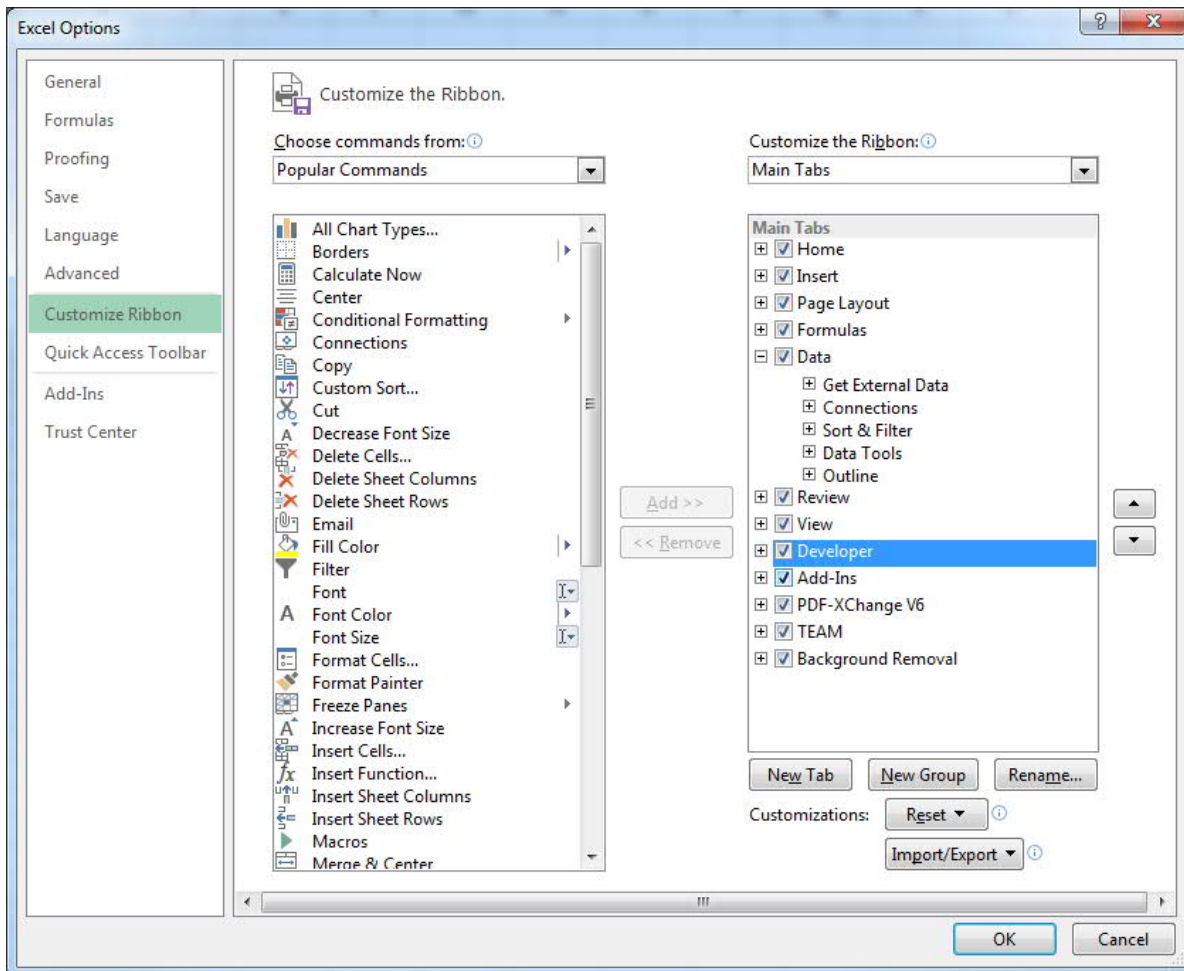
For the purposes of this application note we will illustrate how VBA can be used within Microsoft Excel to communicate with (and control) a legacy MintDrive<sup>II</sup> via a RS232 serial connection. The same principles apply to all Mint controllers (we will highlight specific differences throughout the text – for example, how to connect to a USB based controller such as NextMove ESB-2 or an Ethernet based drive such as MicroFlex e190).

### Getting Started

If you intend to follow the practical example in this application note you will need the following:

- A PC running Windows 7 with an Ethernet adaptor configured to communicate with a MicroFlex e190 servo drive (i.e. the Ethernet adaptor should be on the same subnet as the drive – 192.168.0 being the default, so the PC adaptor may be set to 192.168.0.201 for example)
- A copy of Microsoft Excel (we used Excel 2013 for our example)
- A copy of Mint Workbench (Build 5852 or later ideally)
- A MicroFlex e190 drive complete with AC servo motor (linear or rotary) – it is suggested that you initially setup the drive if necessary via the Commissioning Wizard provided with Mint Workbench and make a note of the drive's IP address (e.g. 192.168.0.1) and the Mint SCALEFACTOR that has been set (e.g. if an ESM motor with Smarttabs feedback is used and the axis is scaled into a user unit of 'revs' then this may be 131072)

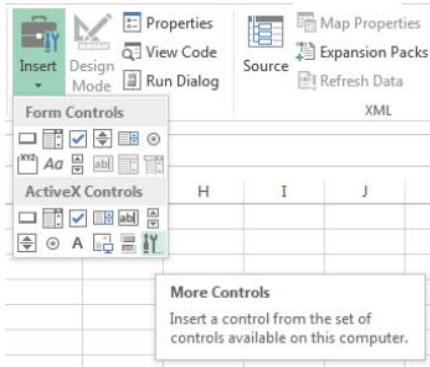
Start Microsoft Excel – by default you may not have the 'Developer' toolbar available. The method of adding this varies according to the version of Excel being used....in Excel 2013 this is added by selecting the 'File' tab, clicking 'Options' to enter the Excel options dialog, clicking 'Customise ribbon' and then in the right hand pane selecting 'Main Tabs' from the 'Customise ribbon' drop down list. Ensure the 'Developer' check box is selected...



Once the developer tab is present in the Excel main menu you can now add the Mint ActiveX controls to the workbook/sheet. Click on the 'Developer' tab and then click on 'Insert' button...

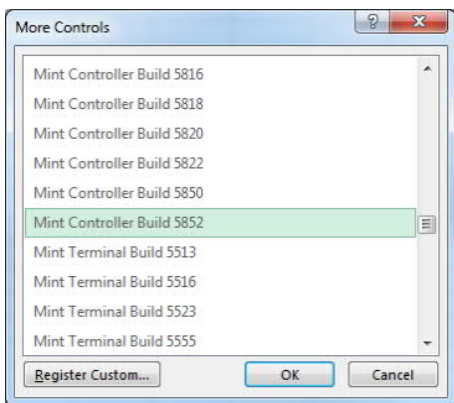


Now click on the 'More Controls' button in the Control Toolbox....

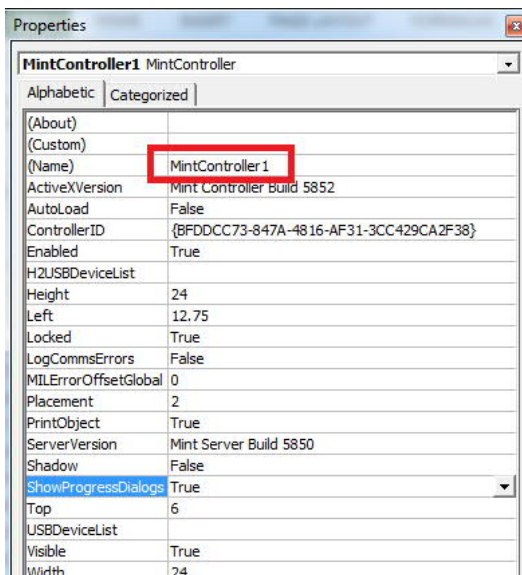
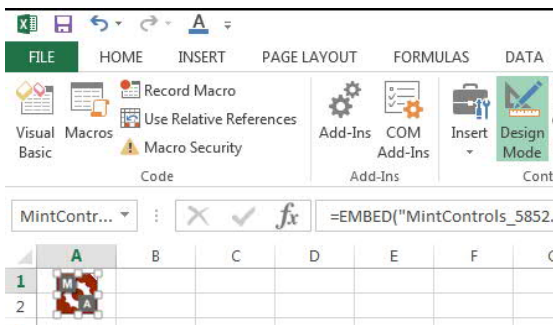


Clicking the 'More Controls' button will cause your PC to read the system registry to discover what ActiveX controls are installed so don't be worried if there is a small delay whilst it does this

From the list of available ActiveX controls select the latest version of the 'Mint Controller' (in our example this is Build 5852). Be sure to select the Mint Controller as there are also controls available for the Mint Terminal Window and the Mint Command Prompt...



The cursor will change to a crosshair – click and drag the mouse to draw a box on the Excel sheet (putting this at cell A1 is a good idea so you know where it is later).




Right click on the Mint icon and select 'Properties' from the resulting menu. The Properties dialog will show a name for the control (MintController1 by default). This name can be changed if required to something more meaningful to the application...we might call it 'axE190' for example (for ActiveX e190)

### Using a Reference instead of an ActiveX object

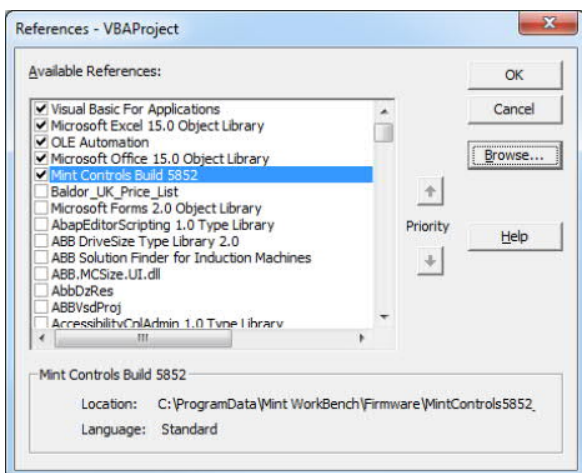
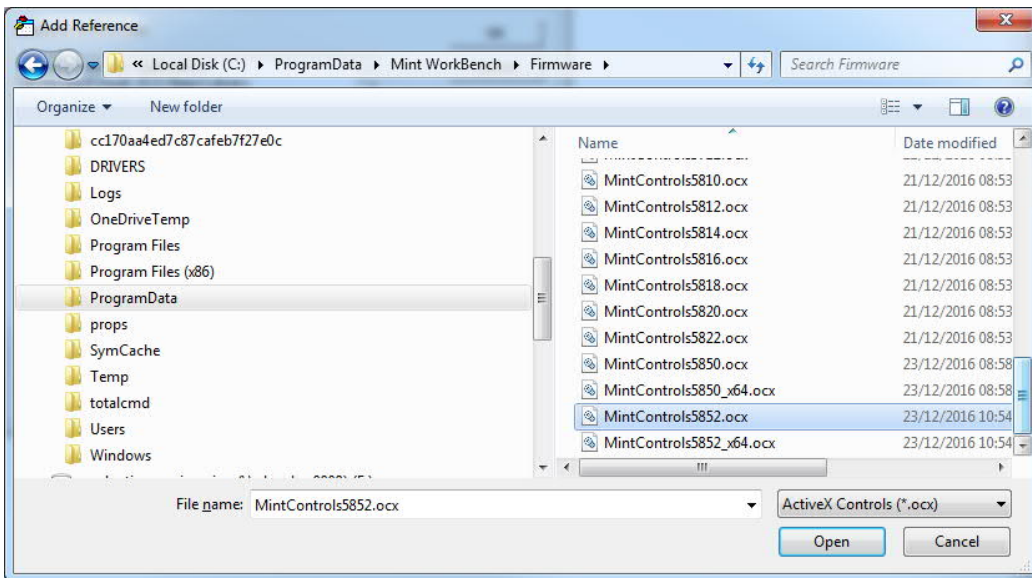
Some applications that support VBA may not provide a 'container' for the ActiveX control to be placed in (such as a form or a sheet). In these cases it is necessary to create a reference to the ActiveX control instead of placing an instance of the control in the container.

To include a Reference to the Mint ActiveX Control navigate to the VBA Editor (the procedure for this will vary according to the application in use).

In Excel 2013 the VBA editor is displayed by clicking on the 'View code' button on the Developer tab 

Once in the VBA editor environment select Tools>References... from the top menu and then use the Browse... button to navigate to the installation directory for the Mint ActiveX components (e.g. if using Windows 7 you will find this at C:\ProgramData\Mint Workbench\Firmware by default).

Change the 'Files of Type' dropdown to 'ActiveX Controls (\*.ocx)' and the available references will be shown (again select the latest version – 5852 in our example below). Click on 'Open' and then on 'OK' at the next dialog.

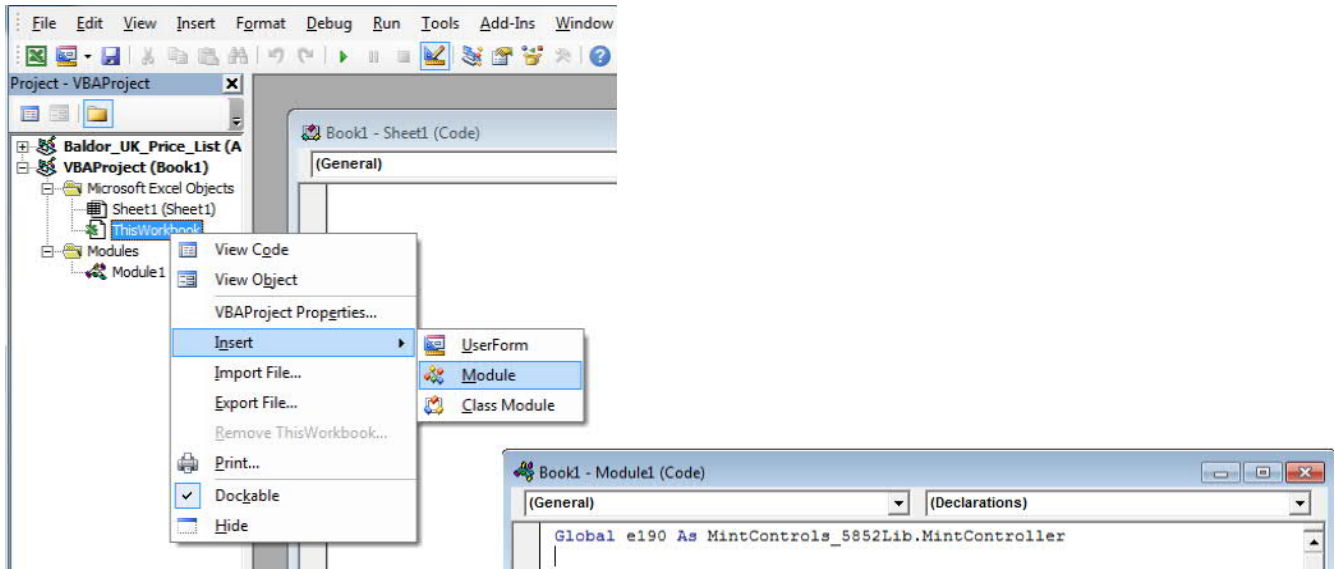


Now the Reference has been created we need to declare an instance of the MintController in the VBA application (and make this globally available to the whole project).

In Excel we could do this by including a new VBA code module in the Workbook and adding the line...

```
Global e190 As New MintControls5852Lib.MintController
```

...to the General>Declarations section of the module (Note that 'e190 is the name we've decided to give the control in this example, in much the same way as the Properties dialog could be used to give a name such as 'axE190' to the ActiveX component earlier).



...and then including a subroutine in this code module to initialise an instance of this as follows:

```
Sub Initialise()
    Set e190 = New MintControls_5852Lib.MintController
End Sub
```

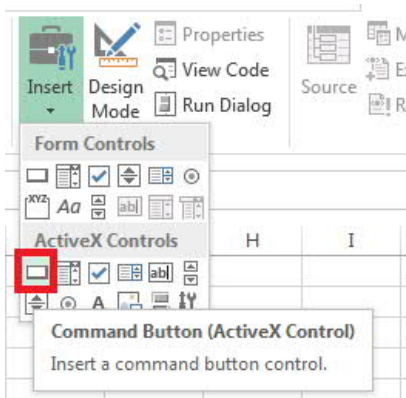
### Establishing Communications

Before the ActiveX control can be used to issue any Mint commands we need to establish a communications link between the PC and the Mint controller. We will add a button to our Excel sheet that does this when pressed (to automate the process as far as possible this could be added to the inbuilt Excel 'Workbook – Open' procedure instead, but closing and opening the Excel document during development is time consuming so we've stuck to using a button for this example).

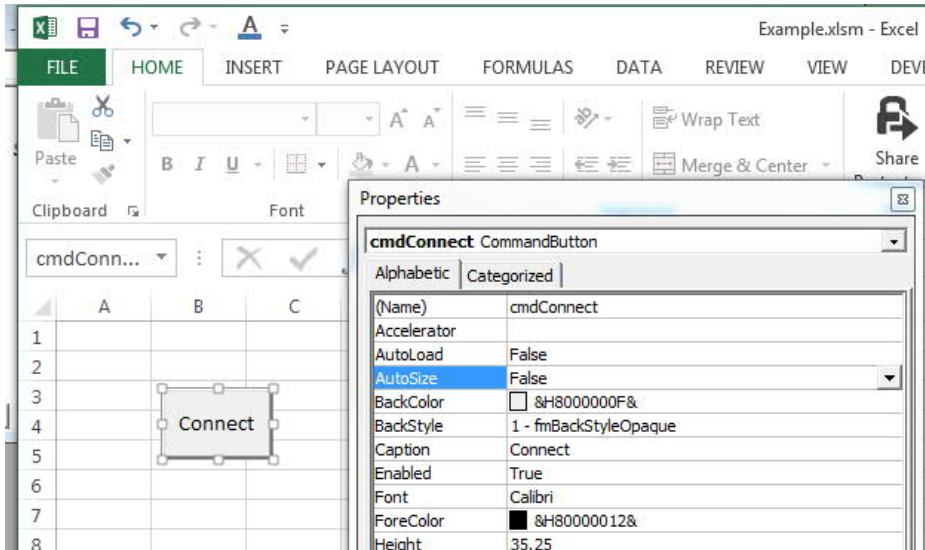
Switch back to the standard Excel view by clicking the Excel icon button



Click on Insert again and this time select a command button from the tool palette...



The cursor will now let you draw a button on the Excel sheet....once added to the sheet right click the button and access the Properties dialog. Give the button a name (we called ours cmdConnect) and change the Caption property to 'Connect' for example....



Now double-click the Connect button and Excel will open the VBA editor and display the click event procedure for this button automatically...



This is where we will add some code to establish our communication connection with the controller. If you placed a Mint control instance on the sheet then type the name of the ActiveX control followed by a dot (.). If a reference was used instead then we need to include an additional step to instantiate the Mintcontroller object....if you used the reference method simply add the line...

Initialise

...or whatever you called your subroutine that was added earlier to the VB code module. You can now also type the name of the object followed by a dot. The screenshots below show how the click event would look in these two cases...

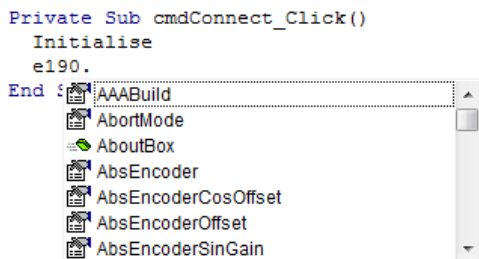
ActiveX object on form:

```
Private Sub cmdConnect_Click()
    axE190.
End Sub
```

Reference to Mintcontroller:

```
Private Sub cmdConnect_Click()
    Initialise
    e190.
End Sub
```

As soon as the dot is entered you should see that VBA displays a list of all of the available ActiveX methods and properties that can be coded (a lot of these correspond to equivalent Mint keywords, some may be prefixed with set, get or do).



Scroll down through the list until the appropriate connection method is highlighted (the table below details which method should be selected for each Mint controller)...

Controller	Connection Type	ActiveX Method	Note
NextMove ESB-2	Serial (RS232/RS422)	SetSerialControllerLink	Use platform type 36
NextMove ESB-2	USB	SetUSBControllerLink	
NextMove e100	Serial (RS232/RS422)	SetSerialControllerLink	Use platform type 31
NextMove e100	USB	SetUSBControllerLink	
NextMove e100	Ethernet	SetEthernetControllerLink	
MicroFlex e190	Ethernet	SetEthernetControllerLink	
MotiFlex e180	Ethernet	SetEthernetControllerLink	

As we're using a MicroFlex e190 in this example we'll select SetEthernetControllerLink from the list and then press the SPACEBAR on our keyboard (we will use the reference in our example, if you have used an ActiveX object on the sheet then simply replace all instances of 'e190' with 'axE190', or whatever you named your ActiveX control).

VBA displays a list of any parameters that are needed to complete the ActiveX method...

```

Private Sub cmdConnect_Click()
    Initialise
    e190.SetEthernetControllerLink |
End Sub SetEthernetControllerLink(sLinkName As String)
    
```

In this case we need to enter the drive's IP address (our drive is using the default address of 192.168.0.1).

```

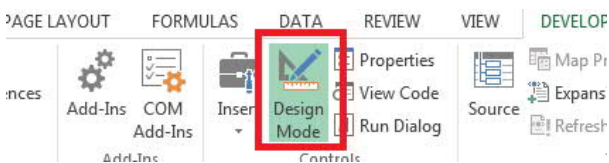
Private Sub cmdConnect_Click()
    Initialise
    e190.SetEthernetControllerLink "192.168.0.1"
End Sub
    
```

Setting the appropriate link is the only command needed to setup a connection to the Mint controller but it would be good to know if the connection has been successful or not. One easy way to do this is to attempt to read something from the controller. We could read the firmware version loaded for example by calling the AAABuild method and passing the result of this to the VBA 'Msgbox' function. The code to do this will look like this....

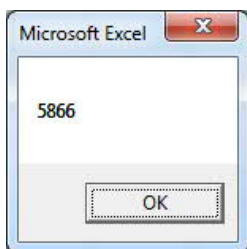
```

Private Sub cmdConnect_Click()
    Initialise
    e190.SetEthernetControllerLink "192.168.0.1"
    MsgBox e190.AAABuild
End Sub
    
```

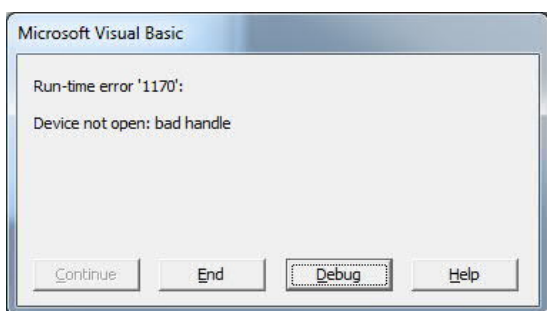
If we now switch back to our Excel sheet view and exit design mode by deselecting the 'Design Mode' button....



...we can now click on our Connect button to see if a successful connection is made or not. If all is well we should see a dialog like this appear...



If the connection fails a run-time error dialog will appear like this...

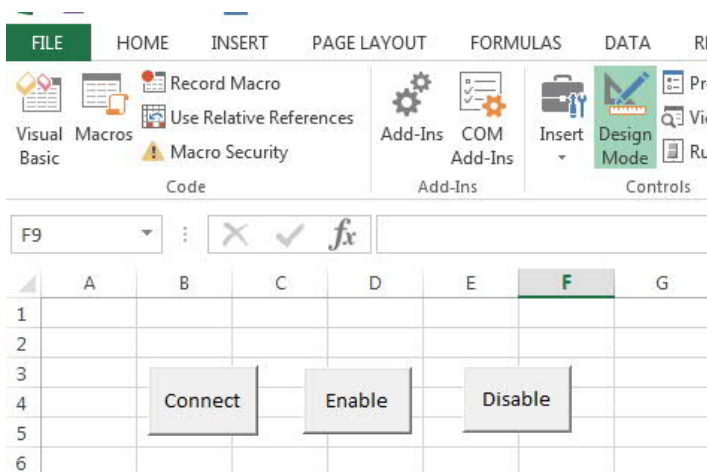


In this case click the 'End' button and check your code and check the connection details for the controller you're using (e.g. have you used the correct IP, USB or serial address). If you make any changes repeat the test until the controller firmware version is reported.

### Enabling/Disabling an axis

Now we've established a connection we can consider how to enable and disable the axis. For the purposes of this application note we'll add two buttons to our sheet.

Repeat the previous steps to add two buttons to the Excel sheet and give these appropriate names and captions (we called ours cmdEnable, cmdDisable and captions of Enable and Disable as shown below)...



Now we need to add some code to make the buttons 'work'. Double click the 'Enable' button to open its click event procedure. In this routine we will call Mintcontroller ActiveX methods to...

- Clear any errors (CANCEL)
- Wait for this to complete (WAIT)
- Enable the axis (DRIVEENABLE)

Providing the axis is ready to be enabled (e.g. we are using an e190 drive so the AC supply must be present, the STO must be inactive etc...) clicking the button should enable the drive.



Our enable button click event procedure looks like this once completed...

```
Private Sub cmdEnable_Click()
    e190.DoCancel (0)
    e190.DoWait (100)
    e190.DriveEnable(0) = True
End Sub
```

In most cases (particularly if using a reference to the Mintcontroller) the ActiveX methods match the corresponding Mint keywords. There are some exceptions, usually where the Mint keyword would clash with a pre-existing Windows method or property, DoCancel instead of Cancel is an example of this.

Note that most methods take one or more parameters (so in the above example DoCancel and DriveEnable use a parameter for the axis number (0 in our case) and DoWait takes a time in ms (as in the Mint instruction).

We can now add some code to the disable button as below...

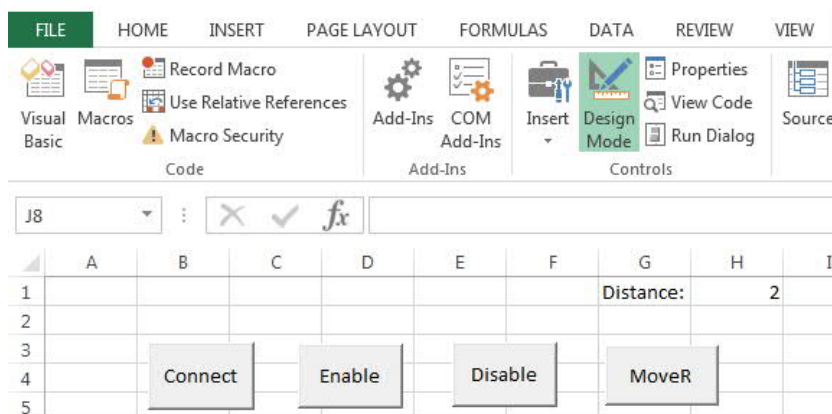
```
Private Sub cmdDisable_Click()
    e190.DriveEnable(0) = False
End Sub
```

We're now ready to try these commands out, switch to run-time mode (deselect the design time button), make sure the drive/axis is ready to be enabled and click on the enable button on the spreadsheet – the seven segment display should indicate the drive is enabled.

Click on the disable button on the spreadsheet and the drive should disable (the seven segment display should indicate -).

## Performing Motion

Switch back to design mode and add one more command button to the spreadsheet. This button is going to perform a relative move when clicked. Give this button a suitable name and caption (e.g. cmdMoveR and MoveR). To make the move distance variable we'll enter a value on the sheet and use this to set our relative move distance (in user units according to the scale factor set on the drive/axis....our MicroFlex e190 is scaled for revs as a user unit). Add some text to the spreadsheet to indicate to the user where to enter the move distance and enter a value in a specific cell....we used cell H1 as shown below:



Double-click the MoveR command button and enter the following two lines of code in the click event procedure..

```
Private Sub cmdMoveR_Click()
    e190.MoveR(0) = Range("H1")
    e190.DoGo1 (0)
End Sub
```

The Range function in VBA allows us to specify a single cell's contents as the value to use in conjunction with the MoveR method.

Just as in Mint, the MoveR must be triggered with a GO (refer to TRIGGERMODE in the Mint Help File for details on how moves can be auto triggered – i.e. without needing to issue the GO command). We have used the DoGo1 method (which is dedicated to triggering motion on a single axis – in this case axis 0).

You can now switch out of Design Mode and try to move the axis using the MoveR button (note that the move will occur using the values of SPEED, ACCEL and DECEL currently defined on the controller for the axis). A more advanced version of this code might only issue the move if the drive is enabled (which can be tested by using the ActiveX DriveEnable method to read the state of the drive)....

```
Private Sub cmdMoveR_Click()  
    If e190.DriveEnable(0) = True Then  
        e190.MoveR(0) = Range("H1")  
        e190.DoGo1 (0)  
    End If  
End Sub
```

Our example Excel 2013 spreadsheet is included with this application note.

### Summary

You should now have a working application allowing you to trigger motion on a Mint drive/axis from a PC application (in this case Excel). You should see that this is extremely simple to use, identical principles are used for any application support ActiveX/COM components.

In this example we triggered a simple relative move, in practice any of the available motion commands may be used. In some applications it may be preferable to issue all motion commands from within a Mint program on the controller and use the ActiveX just to exchange data (variables, comms, netdata etc...) between the PC and the controller. Or a combination of the two schemes is also possible, there are no limits.

Please refer to the other ActiveX related application notes on the support website for further information.

### Contact us

For more information please contact your local ABB representative or one of the following:

[new.abb.com/motion](http://new.abb.com/motion)  
[new.abb.com/drives](http://new.abb.com/drives)  
[new.abb.com/drivespartners](http://new.abb.com/drivespartners)  
[new.abb.com/PLC](http://new.abb.com/PLC)

© Copyright 2012 ABB. All rights reserved.  
Specifications subject to change without notice.