Using VBA in Microsoft Excel for Instrument and Test Automation

We were recently asked by a customer for some example code which allowed them to use VBA in Excel to control a Keithley Series 3706A switch mainframe (sans internal DMM option) with a 3724 multiplexer card, routing the signals out to a Keithley Model 2002 8.5 digit multimeter to capture high-resolution DC voltage measurements. While, technically, the user could tether the instruments together and have the 3706A trigger the 2002 to make a measurement after each individual channel is closed, then have the 2002 trigger the 3706A when the measurement is complete, we were not asked for anything that complex. What I found the most challenging from this request was a decent "how to" with respect to setting things up in Excel. I Google-searched to find some content was out of date and some not giving me all the details I needed. No matter – I trudged my way through it and made it happen. However, there is little sense in hoarding the know-how only to be asked later, "do you have an example of….". The following are the basic steps needed to get your VBA+Excel test automation project up and running.

Step 1: Get VISA Installed on Your System

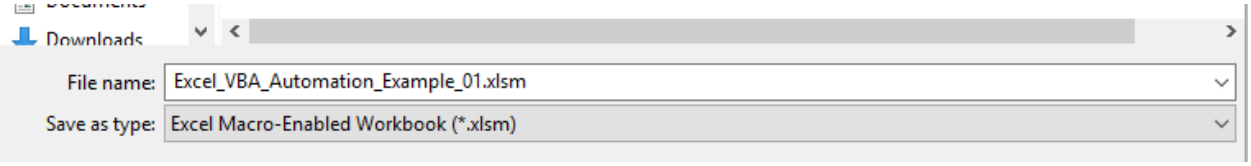
VISA is the foundation for this example – you need to have it installed if you want to move on. In my searches, I found people had put together samples using the NI-VISA and Keysight VISA – I use the former and you can obtain a copy of the 17.5 Run-Time Engine here.

Step 2: Get Microsoft Excel Installed on Your System

Yes, I know, you probably already had this installed. Had to throw this in, otherwise there will be comments like, "but you didn't say I had to install Excel…. How was I supposed to know?" For your information, we use Excel 2016 for this example.

Step 3: Open Excel and Save Your Workbook as a Macro-Enabled Workbook

Create a new workbook and after it opens use the ***File->Save As*** option to name your project, but make certain you use the **'Save as type'** drop-down control to select **"Excel Macro-Enabled Workbook"** option. The macros that you will end up defining are triggered by the controls you will create, and this file type enables the VBA tools you will need to continue.

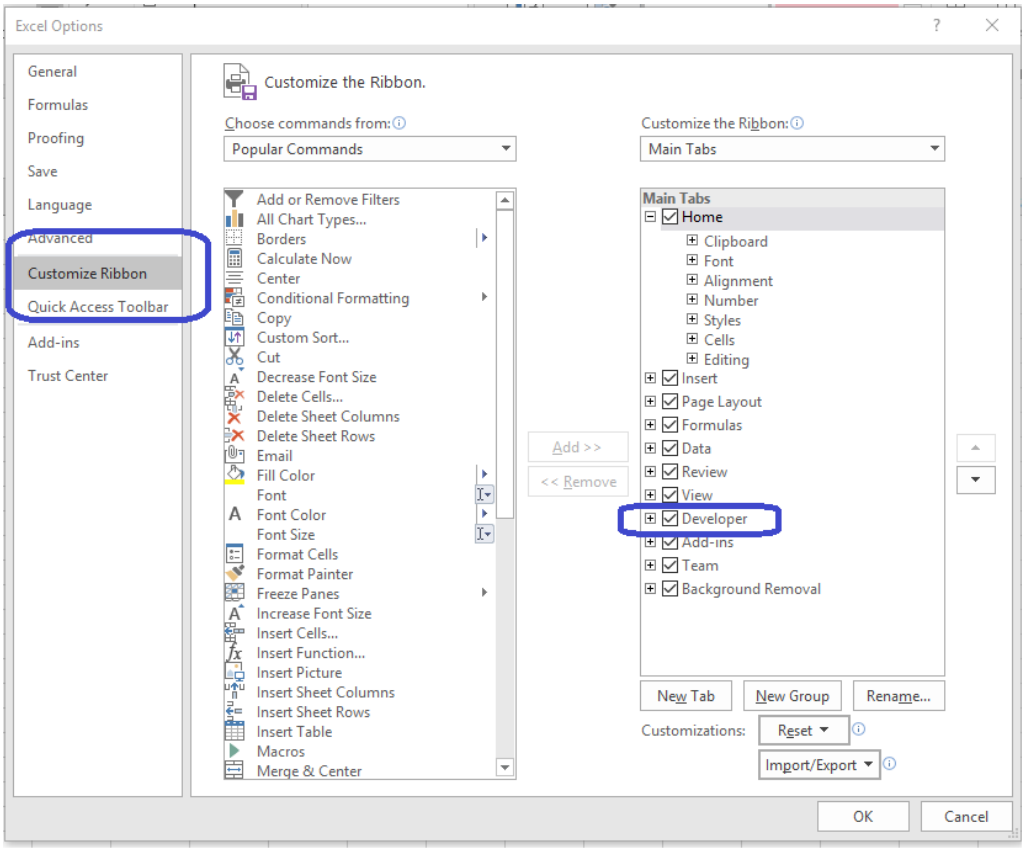


Step 4: Ensure You Have the Developer Tab Available

If you do not see the Developer tab at the top of you Excel interface:

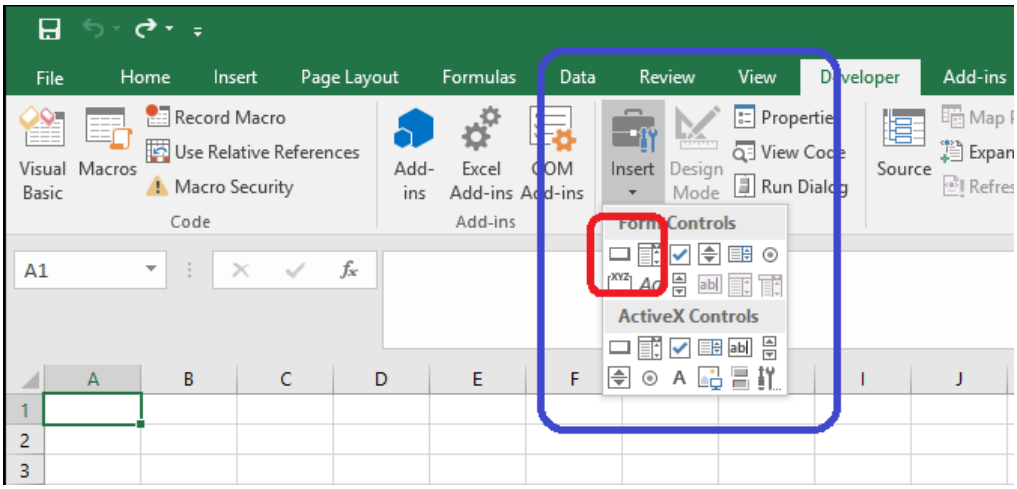
1. Navigate to ***File->Options***
2. Click on the **Customize Ribbon** option in the left-hand pane
3. Ensure the checkbox next to the **Developer** option (right-hand pane under *Menu Tabs*) is checked.

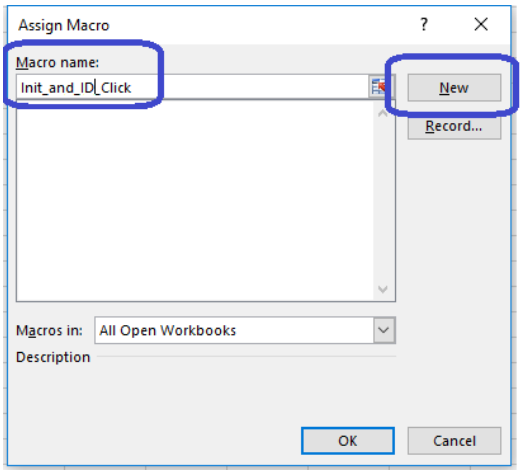
4. Click **OK**

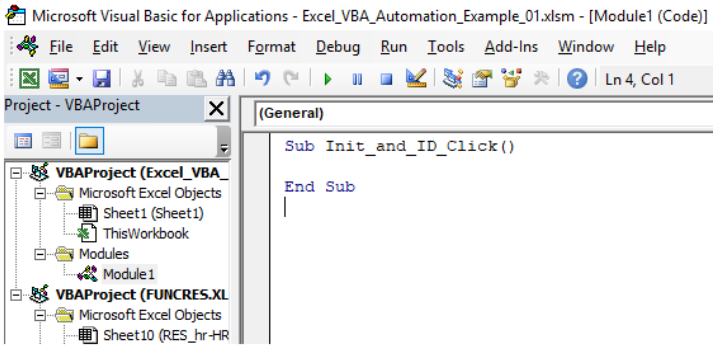


Step 5: Add a Button to Connect to and Query Your Instrument's ID

Click on the **Developer** tab, locate, and click on the **Insert** (toolbox) option to reveal a set of form controls from which to choose. Select the button control (shown below circled in red, and you will get a pop-up if you hover long enough) and draw a button on the worksheet. After placing and sizing your button, you will be presented with the **Assign Macro** dialog, where you can define the macro event that will call a function to run some control code that you will define. Note how we named this macro *"Init_and_ID_Click"* because that is what this button will do. Click the **New** button and this will open up the Visual Basic (VB) editor, showing your *"Init_and_ID_Click"* subroutine template.

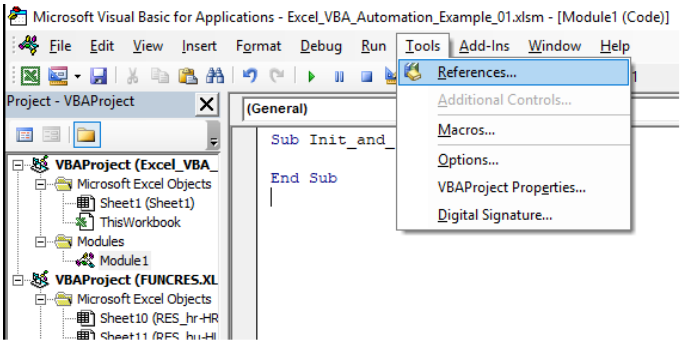


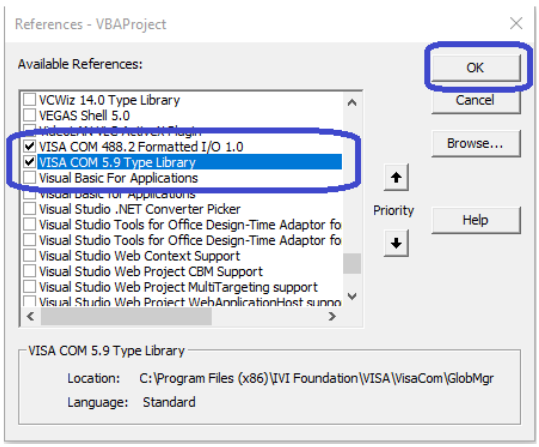




Step 6: Add Your VISA References

From the toolbar in the VB editor, navigate to *Tools->References*. Scroll down through the list of available references to locate both “VISA COM 488.2 Formatted I/O 1.0” and “VISA COM 5.9 Type Library”, selecting each, and then click **OK**.





Step 7: Add Your Control Code

Review the example below and enter it into your editor. I attempted to capture the high-level concepts via the comments. Note that we use an inline system function reference to `Sleep()` in order to add delays where needed – this comes in handy. I am personally a big fan of creating function wrappers, so the `commSend()` and `commQuery()` were an attempt to clean up the overall code.

Hopefully you already know how to power on your instrument; establish its communications settings; connect your instrument to your computer; and obtain the VISA instrument ID. If not then we will leave that as another topic for another day. Also, note our instrument control string for the DMM6500 – seems to make sense to show you how to connect to one of Keithley's latest digital multimeter options.

```
' Define resource manager and instrument objects here...
Dim rscrMgr As VisaComLib.ResourceManager
Dim instDMM6500 As VisaComLib.FormattedIO488
Dim sndBuffer As String
Dim rcvBuffer As String
Dim myDMM6500Str As String
Dim myStatus As Integer

' Add this to be able to call the Sleep() function per Windows
Public Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr)
```

```
Sub Init_and_ID_Click()
    ' Connect to the instruments and write the ID strings to the
    ' worksheet for verification of connection.
    Set rscrMgr = New VisaComLib.ResourceManager
    Set instDMM6500 = New VisaComLib.FormattedIO488

    ' Designate a cell the worksheet to hold the ID...
    ActiveSheet.Cells(2, 4) = ""        ' Use row 2, column 4
    Sleep (500)

    ' Connect to DMM6500 via LAN...
    myDMM6500Str = "TCPIP0::192.168.1.19::inst0::INSTR"
    Set instDMM6500.IO = rscrMgr.Open(myDMM6500Str)

    ' Query the instrument ID and populate in the worksheet....
    rcvBuffer = commQuery(instDMM6500, "*IDN?")
    ActiveSheet.Cells(2, 4) = rcvBuffer
End Sub
```

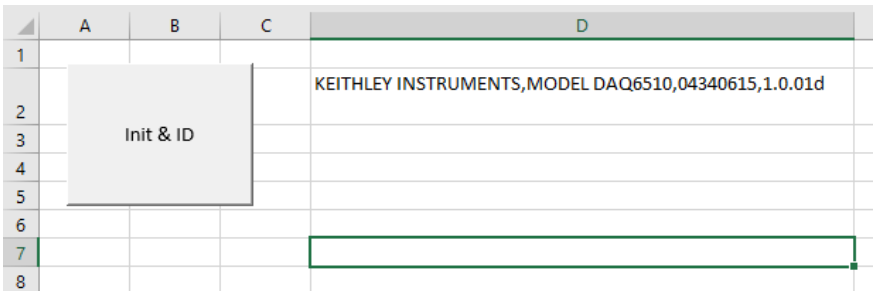
```
Public Function commSend(myInst As VisaComLib.FormattedIO488, sndBuffer As String)
    ' Wrapper function for instrument Writes
    myInst.WriteString (sndBuffer)
    commSend = 0
End Function
```

```
Public Function commQuery(myInst As VisaComLib.FormattedIO488, sndBuffer As String) As String
    ' Wrapper functio for instrument Write + Read
    Dim status As Integer
    status = 0

    status = commSend(myInst, sndBuffer)
    rcvBuffer = myInst.ReadString()
    commQuery = rcvBuffer
End Function
```

Step 8: Click Your Button, Make Stuff Happen

Navigate back to your main Excel worksheet and click your button. You should see the instrument ID string show up on the cells similar to what is shown below.



That is pretty much all you need to get started – all additional control code to customize your particular application is all up to your imagination or test specifications.

A Quick Word on Using the VISA Serial Options

While VISA makes instrument connectivity simple, those using a serial port (RS-232) will need to take a few extra steps during setup. Attributes such as baud rate, flow control, termination character, and others must align with the instrument configuration. There is an additional object that you must instantiate and link to the instrument object to make these modifications. Note how this is handled in code in the image below, which is a portion of an DMM6500 example using the 2001-TCSCAN card to capture thermocouple measurements once a minute for two hours.

```
' Define resource manager and instrument objects here...
Dim rscrMgr As VisaComLib.ResourceManager
Dim instDMM6500 As VisaComLib.FormattedIO488
Dim sndBuffer As String
Dim rcvBuffer As String
Dim myDMM6500Str As String
Dim myStatus As Integer

' Add this to be able to call the Sleep() function per Windows
Public Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilli

Sub Connect_To_DMM6500_Click()
    ' Connect to the instruments and write the ID strings to t
    ' worksheet for verification of connection.
    ' Open a serial VISA session to the DMM6500....
    Set rscrMgr = New VisaComLib.ResourceManager
    Set instDMM6500 = New VisaComLib.FormattedIO488

    ActiveSheet.Cells(2, 2) = ""
    Sleep (500)

    ' Connect to Model DMM6500
    myDMM6500Str = "ASRL4::INSTR"
    Set instDMM6500.IO = rscrMgr.Open(myDMM6500Str)

    ' Set serial interface-specific attributes...
    Dim serInfc As VisaComLib.ISerial
    Set serInfc = instDMM6500.IO
    serInfc.BaudRate = 9600
    serInfc.FlowControl = ASRL_FLOW_NONE
    serInfc.Timeout = 10000
    instDMM6500.IO.SendEndEnabled = True
    instDMM6500.IO.TerminationCharacter = 10          ' could
    instDMM6500.IO.TerminationCharacterEnabled = True
    instDMM6500.IO.Timeout = 10000

    rcvBuffer = commQuery(instDMM6500, "*IDN?")       ' query
    ActiveSheet.Cells(6, 5) = rcvBuffer
End Sub
```

Examples to Share

We started this post with a reference to a customer example that controls the 3706A and 2002, but in reality we used a DMM7510 to perform the measurements. This example has been posted in the Tek Forum and can be found here.

Additionally, the serial example (using the DMM6500) can also be found here.

Enjoy!