SNAP PAC RESTful API to Excel Spreadsheet

About this Tutorial

This technical note describes how to read from and write to an Opto 22 SNAP PAC programmable automation controller via the PAC's RESTful API, using Visual Basic for Applications (VBA) with Microsoft Excel. In this way you can securely share I/O point and variable data in your SNAP PAC controller with an Excel spreadsheet.

If you have not already done so, download sample spreadsheets in the SNAP PAC RESTful API to Excel Spreadsheet Example on our website at www.opto22.com. Once downloaded, you can immediately use them to read and write to your SNAP PAC.

The download includes these files:

- Opto22_RESTfulPAC_ReadingWritingCharting.xlsx—A macro-enabled Excel file containing three spreadsheets and Visual Basic code. These spreadsheets require only the name or IP address of a PAC and the login and password for a user with Read and Write privileges. (See "Prerequisite: Setting Up the PAC" on page 2.
- JsonConverter.bas—A third-party class for VBA created by MIT (copyright Tim Hall, https://github.com/VBA-tools/VBA-JSON). This file is available for general use as long as the provisions in its copyright are preserved.
- 2203_SNAP_PAC_RESTful_API_to_Excel_Spreadsheet_Technical Note.pdf—(This document)

The first part of this tech note explains the sample spreadsheets. The optional second part of the tech note explains step by step how to create a spreadsheet of your own, if you wish to.

Questions about this tutorial can be addressed to Support@opto22.com.

About the RESTful API

The REST API is a way for software applications to securely access data in an Opto 22 SNAP PAC controller by making GET and POST requests. The complete Opto 22 RESTful API is documented here:

http://developer.opto22.com/static/generated/pac-rest-api/swagger-ui/index.html

At this link you'll find URL paths to read any data available to the PAC Control strategy running on the PAC and to write to the same data, if it is a writable data type. All the GET and POST requests return information in standard JavaScript Object Notation (JSON).

The sample Excel file uses the PAC's RESTful API and common Excel features to create an interactive spreadsheet that can query your PAC and then build URL paths for the data in your control strategy. You don't need to know the I/O point and variable names on your PAC, because the Excel sample can read the I/O and variable names and will create the paths. To see how to construct the HTTP requests and parse the returned data, you'll look in the Visual Basic code included in the Excel sample. We'll explore the code below.

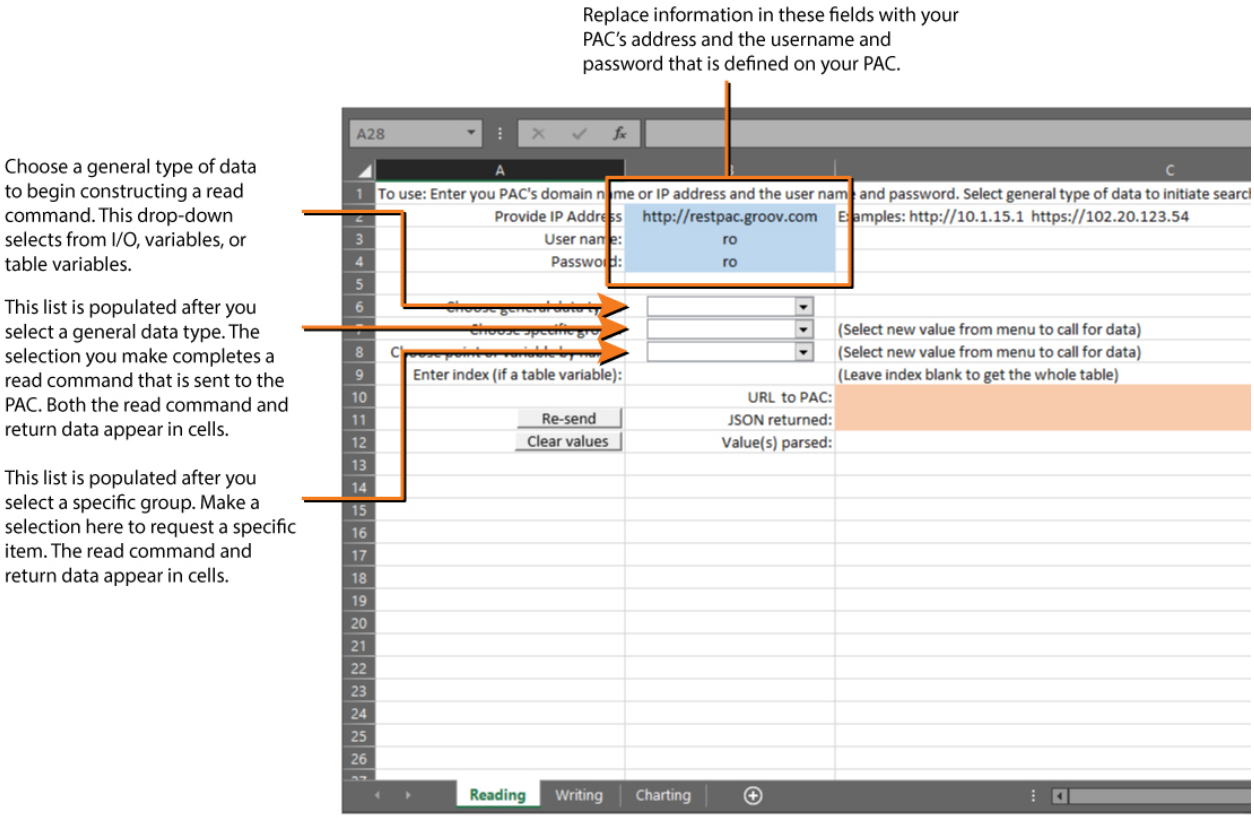
Prerequisite: Setting Up the PAC

Before you begin, enable the RESTful interface on the PAC and create a user account with read and write privileges as described in the SNAP PAC REST API Quick Start, which is available at http://developer.opto22.com/rest/pac/quickstart/

About the Sample File

If you're an experienced Excel user, you may wish to start by examining the sample file. The following images provide an overview.

Reading spreadsheet



Writing spreadsheet

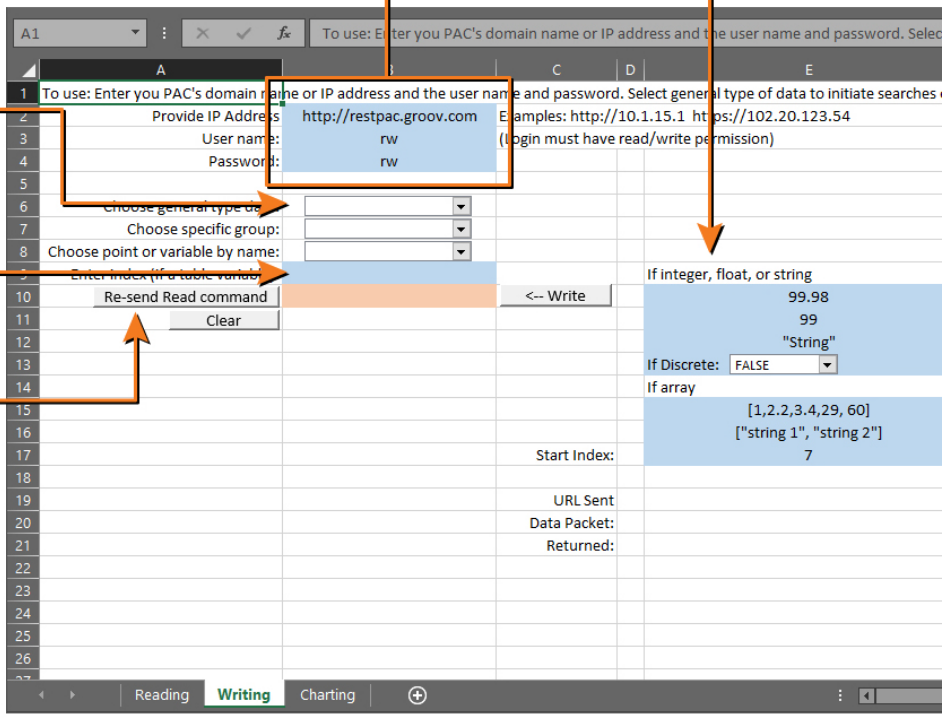
Replace the information in these fields with your PAC's address and the username and password that is defined on your PAC.

Enter the information to write in these fields. Each cell is used for a different type of variable. The write function in Excel will choose the appropriate field based on the type of data you select.

Use these three drop-down lists to identify a specific I/O point, variable, or table variable to write to.

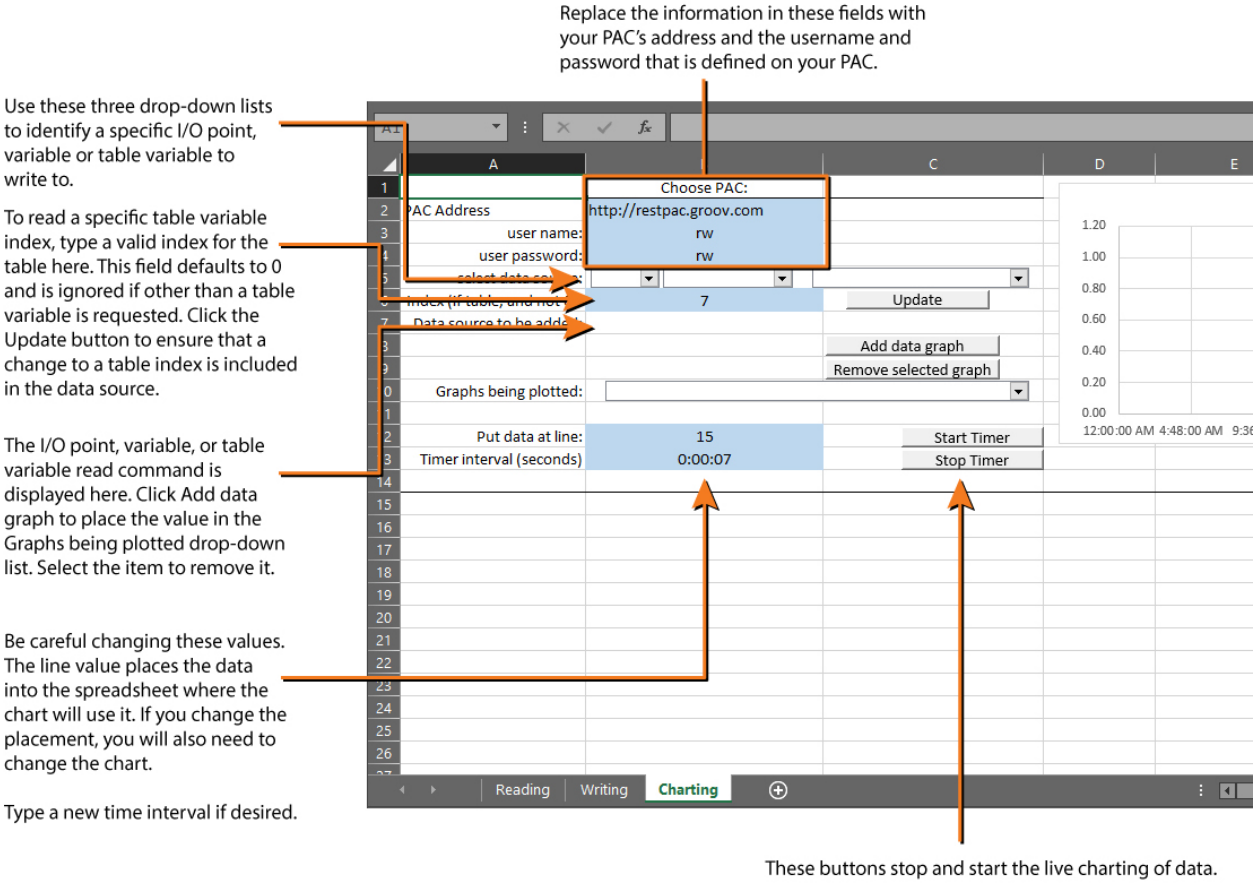
To read a specific table variable index, type a valid index for the table here. This field defaults to 0 and is ignored if other than a table variable is requested.

Click this button to re-send the read command so you can verify the write command.

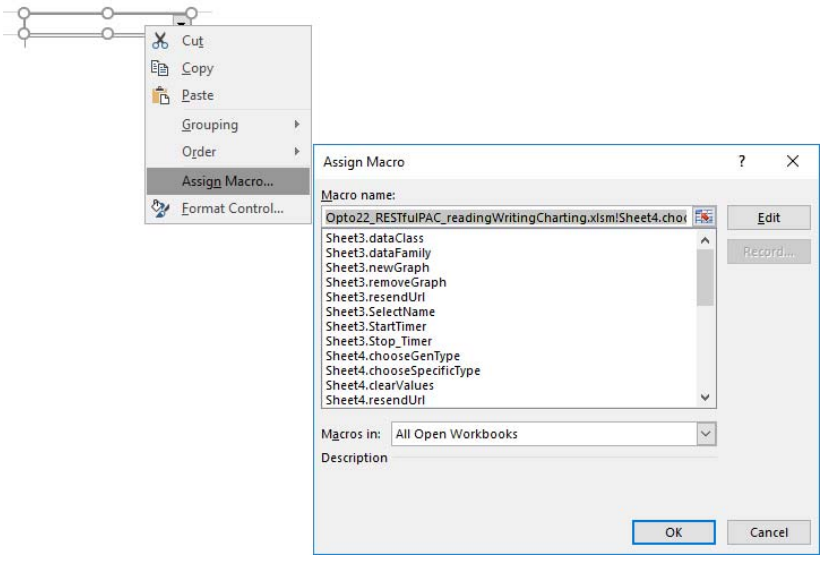


Charting spreadsheet

The Charting sheet shows how to call data on a timed interval. The data are displayed in cells and in a chart.



Visual Basic Behind the Spreadsheets

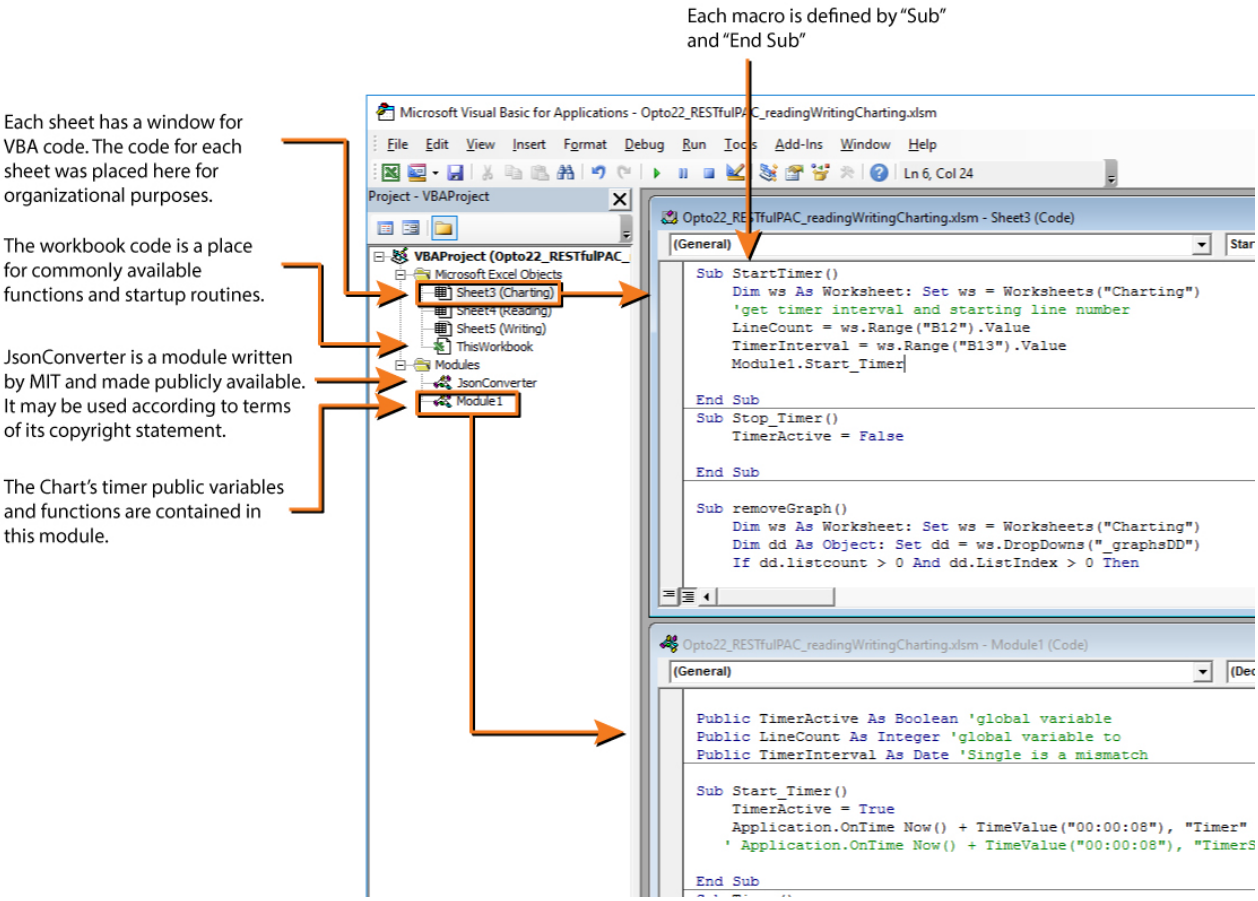


The code that reads, writes, and plots data resides in the spreadsheet as modules and macros. Macros are run from controls placed in the spreadsheet. To see the list of macros that can be assigned to a control (such as a drop-down list or button), right-click the control.

To see the content of macros, first open the Visual Basic Editor. The VBA editor is available from the Developer menu. Note that the Developer menu is not displayed by default.

To display the Developer menu:

1. Choose File -> Options.
2. In the options win, select Customize Ribbon.
3. Select Developer under Main Tabs and close the Options window.
4. Select the Developer menu.
5. Select the Visual Basic icon.



(Optional) Setting up the Excel Development Environment

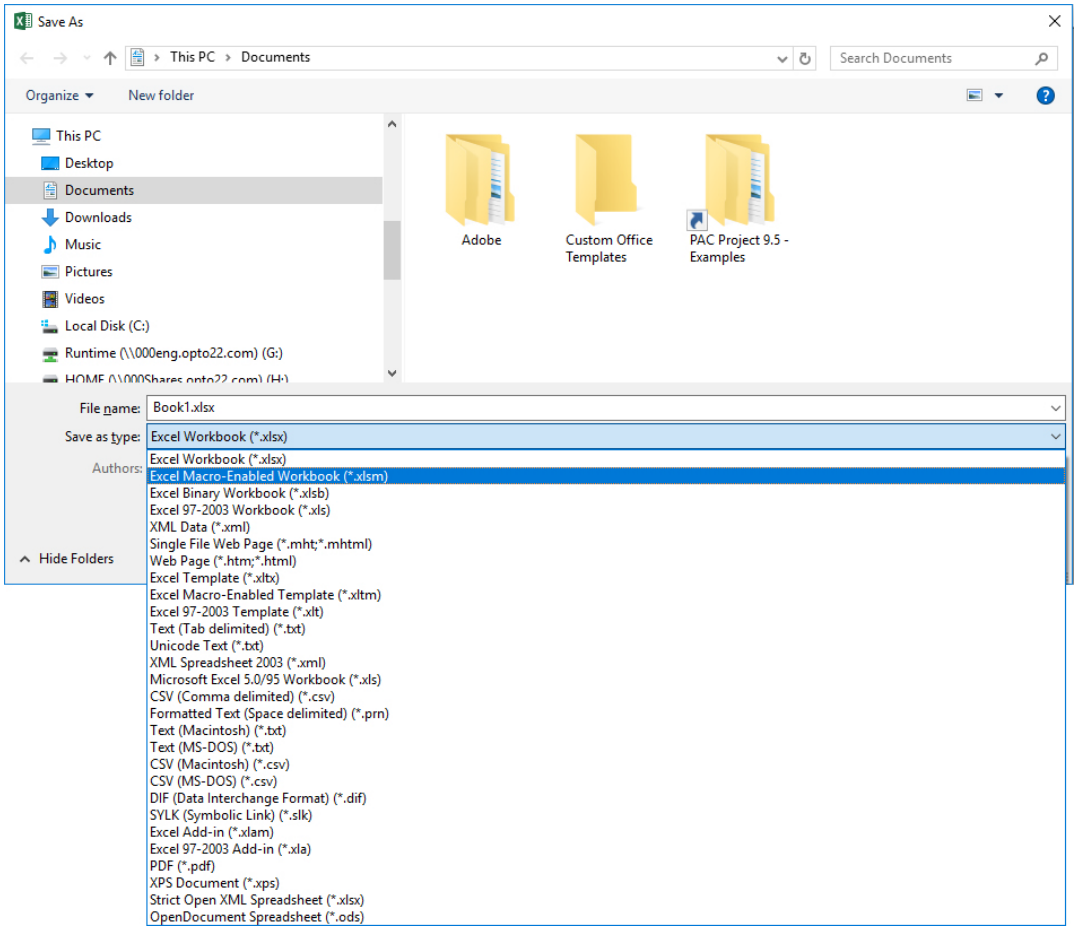
You can use the spreadsheets just as they are to read and write to your SNAP PAC. However, if you want to understand more about how it was built or build your own, this last section shows you step by step how to create a new spreadsheet, use the Visual Basic Editor, and write a macro to read data from your PAC.

If you are creating a new Excel spreadsheet, you will need to create a macro-enabled spreadsheet and add resources to the Visual Basic editor.

Create a macro-enabled spreadsheet

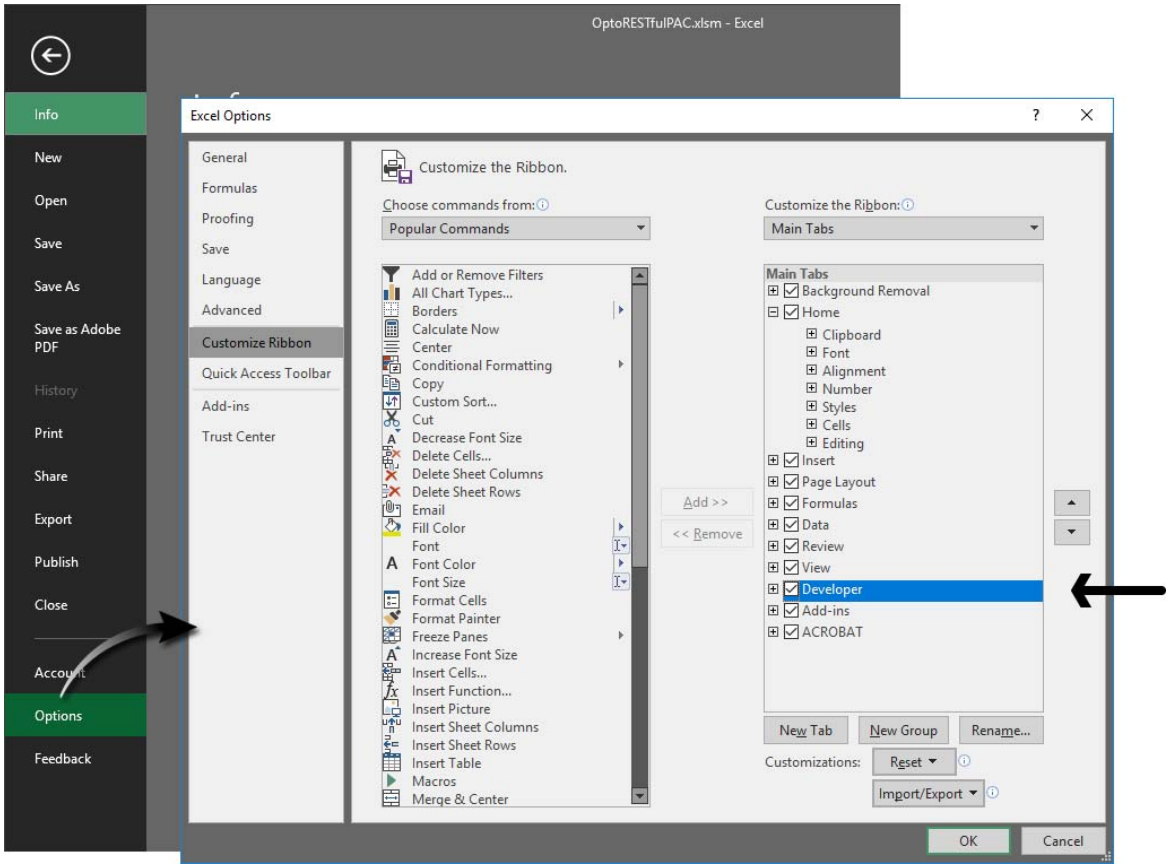
1. Create a new file in Excel.
2. When saving, choose Excel Macro-Enabled Workbook.

   When you save the file this way, the filename ends in .xlsm.



Enable the developer tool bar

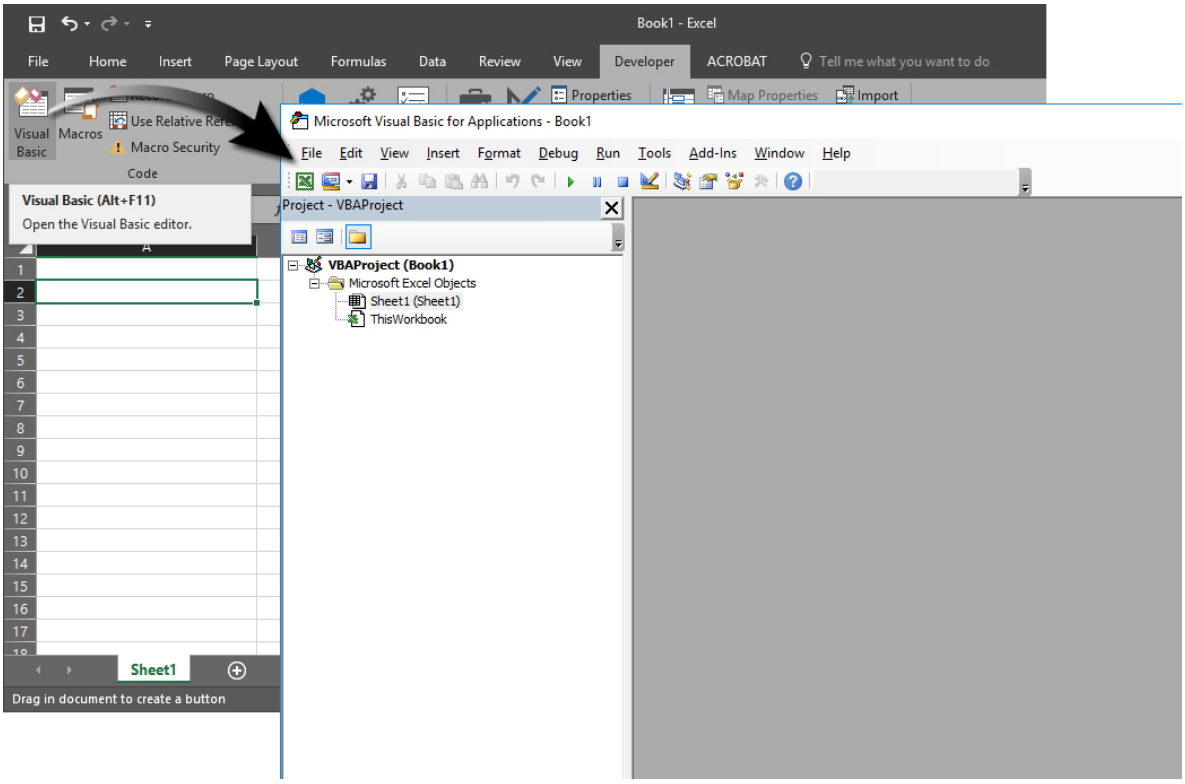
1. Choose File > Options.
2. Choose Customize Ribbon.
3. Under Main Tabs, select Developer.



4. Close the Options dialog box and return to the spreadsheet.

Open the Visual Basic editor

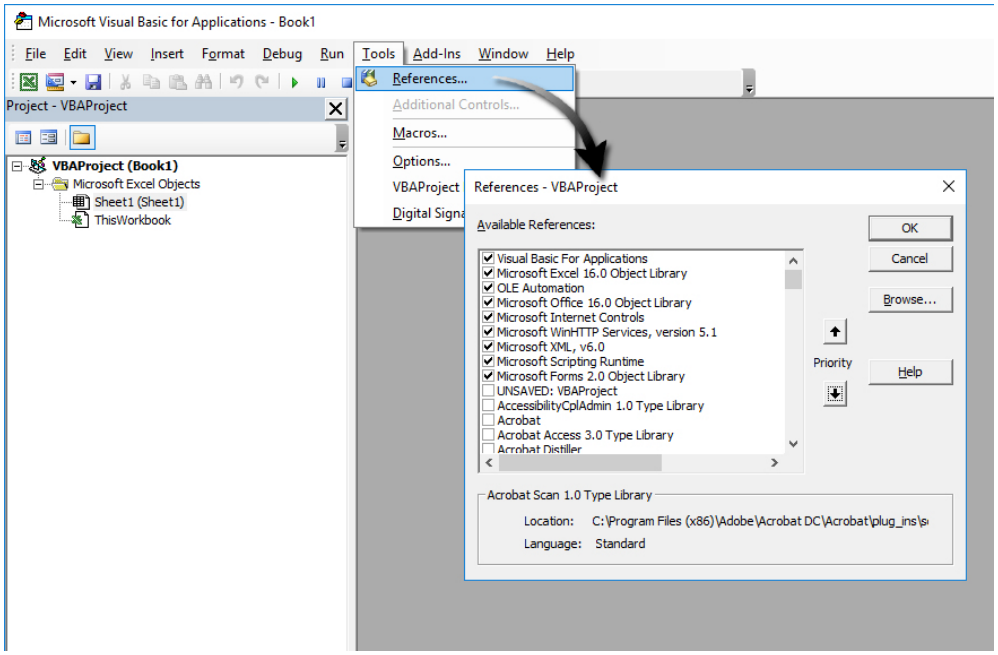
1. Choose the Developer tab.
2. Choose the Visual Basic button. This opens the VBA Editor in its own window.



Add VBA references

The VBA editor defaults to a minimum set of libraries enabled. You will need to add more.

1. Choose Tools > References.
2. Find the following libraries and select each.
   - Visual Basic For Applications
   - Microsoft Excel 16.0 Object Library
   - OLE Automation
   - Microsoft Office 16.0 Object Library
   - Microsoft Internet Controls
   - Microsoft WinHTTP Services version 5.1
   - Microsoft XML v6.0
   - Microsoft Scripting Runtime
   - Microsoft Forms 2.0 Object Library



3. Click OK to close.

Create a Read command

Write code to read from the PAC

To read data from the PAC, you'll create a macro, which is Visual Basic code attached to the spreadsheet.

1. In the VBA Editor, double click "Sheet1" (or the name you used if you renamed the sheet).
2. Create a subroutine by typing: `Sub ReadPoint()`

   As soon as you press Return, the text `End Sub` will be added. All code added between `Sub` and `End Sub` will make up the body of the macro.

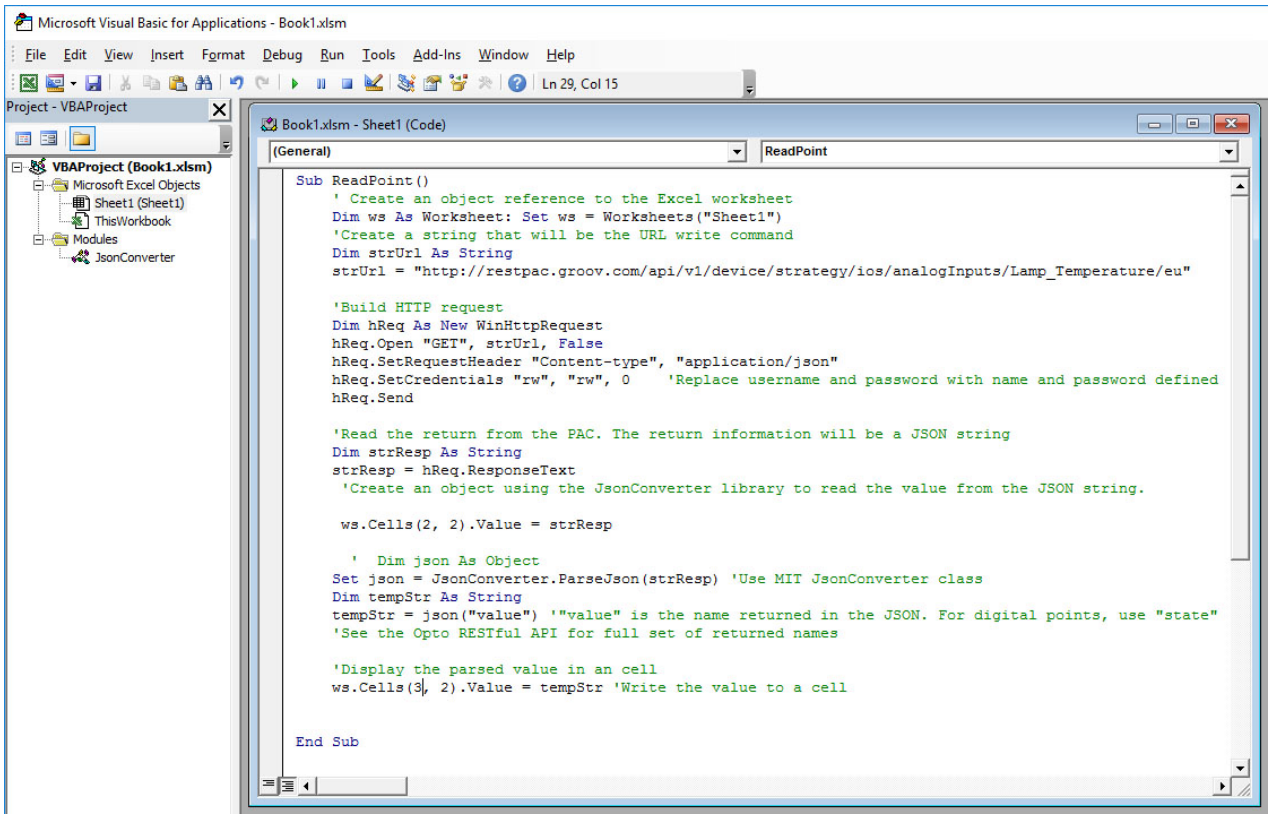
   In this tutorial, you'll enter the body of the macro in two parts. First, you'll enter the code that will retrieve data and display it so that you can see the format of the data returned. Second, you will import a JSON parsing module to convert the returned JSON string into an easy-to-use value.

3. Copy and paste (or type) the following text between `Sub ReadPoint()` and `Sub End`. The text is annotated to explain what each section does. Lines beginning with an apostrophe are comments and will appear in green in the VBA Editor. You'll need to customize the text in italics to your application as described in the accompanying annotations.

```
' Create an object reference to the Excel worksheet
Dim ws As Worksheet: Set ws = Worksheets("Sheet1")
'Create a string that will be the URL write command
Dim strUrl As String
```

```
‘In the following line, substitute restpac.groov.com and
Lamp_Temperature/eu
‘with the address of your PAC and name of your point or variable.
For points, include “/eu”.
strUrl =
"https://restpac.groov.com/api/v1/device/strategy/ios/analogInputs
/Lamp_Temperature/eu"
'Build HTTP request
Dim hReq As New WinHttpRequest
hReq.Open "GET", strUrl, False
hReq.SetRequestHeader "Content-type", "application/json"
hReq.SetCredentials "username", "password", 0    'Replace username
and password with name and password
                                                    ‘defined for your PAC
hReq.Send
'Read the return from the PAC. The return information will be a
JSON string
Dim strResp As String
strResp = hReq.ResponseText
‘This will place the returned JSON string in cell 2B.
ws.Cells(2, 2).Value = strResp
```

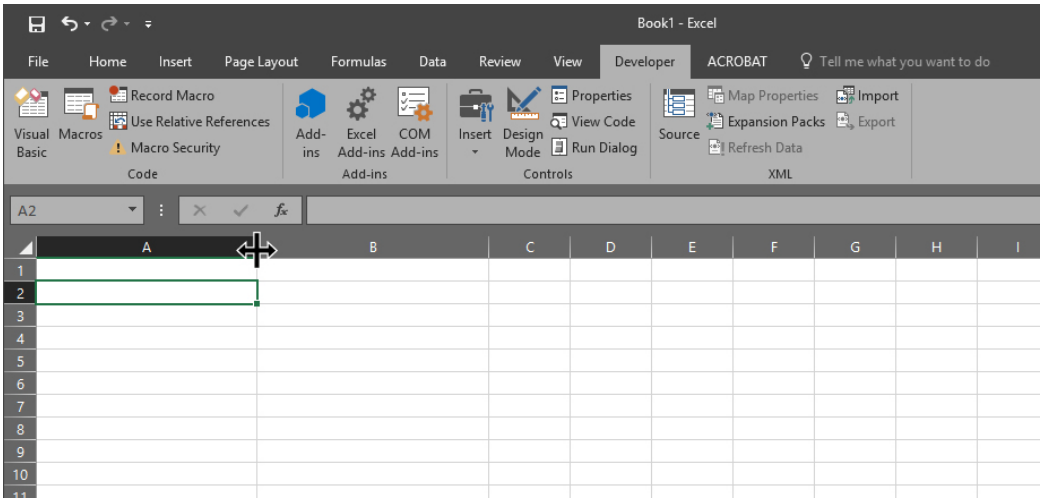
After entering this code, your macro should appear as shown here:



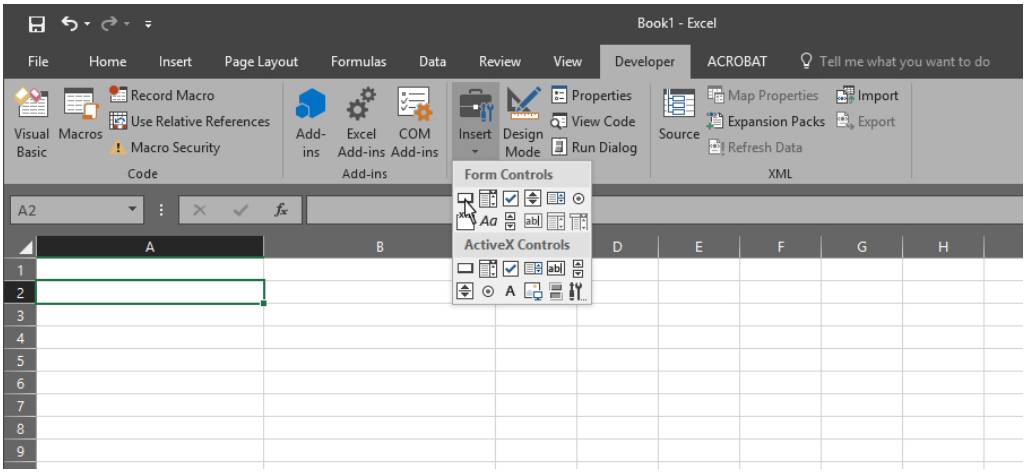
4. Save your changes.

Call the macro from the spreadsheet

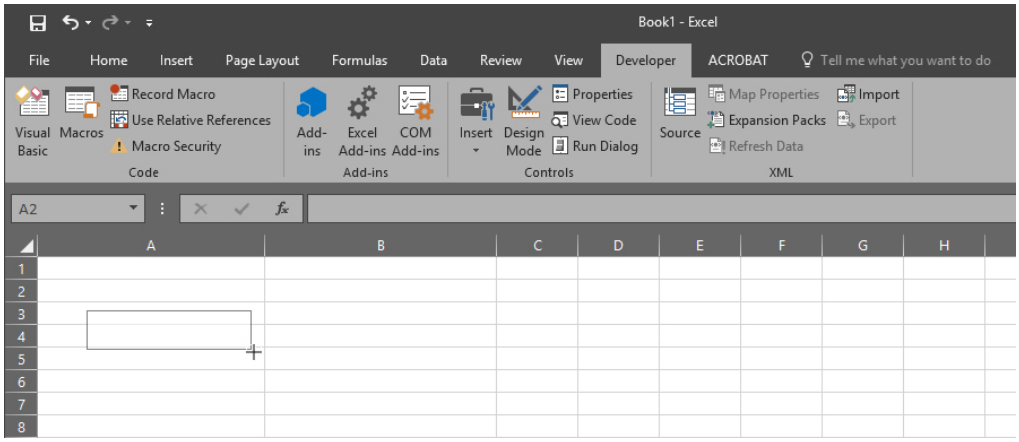
1. Return to the Excel sheet.
2. Resize columns A and B to provide space for drawing a button. Resize by positioning the mouse over the column divider. When the cursor changes to a divider, click and drag the column divider.



3. Click the Developer tab to display the Developer tools.
4. Open the list of Form Controls by clicking the Insert button, and choose the button icon (top left Form Control in this picture):

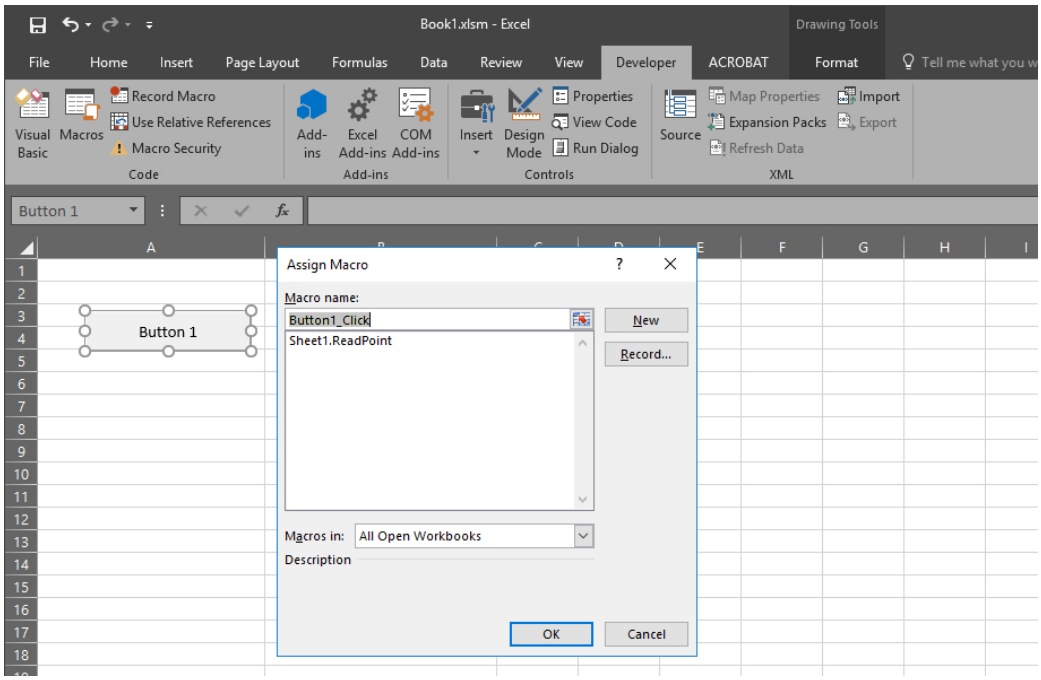


5. Click and drag in the spreadsheet to create the button.



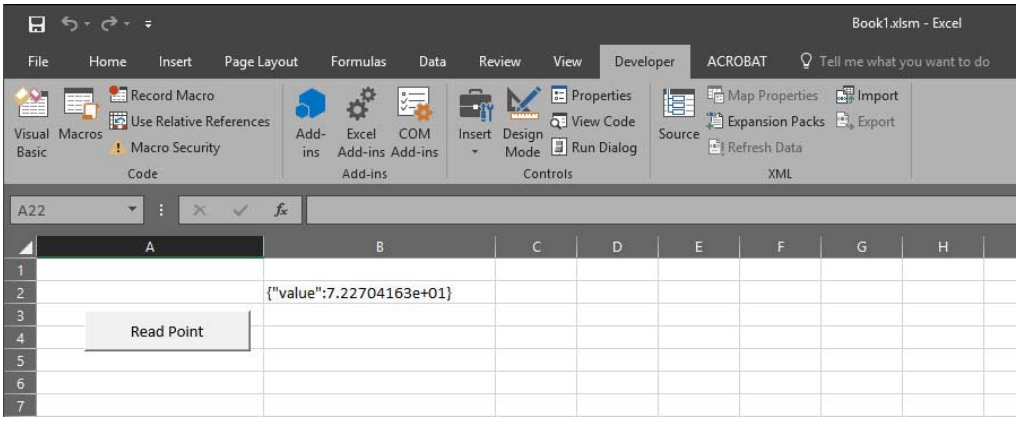
You can resize and reposition the button after you draw it.

As soon as you draw the button, a Macro window appears. The macro you created should be visible in the window:



6. Select Sheet1.ReadPoint and click OK.
7. Change the name of the new button by right-clicking the button. Position the cursor over the text, click the text, and type a new name, such "Read Point."
8. Test by clicking the Read Point button.

   The return string appears in cell 2B:

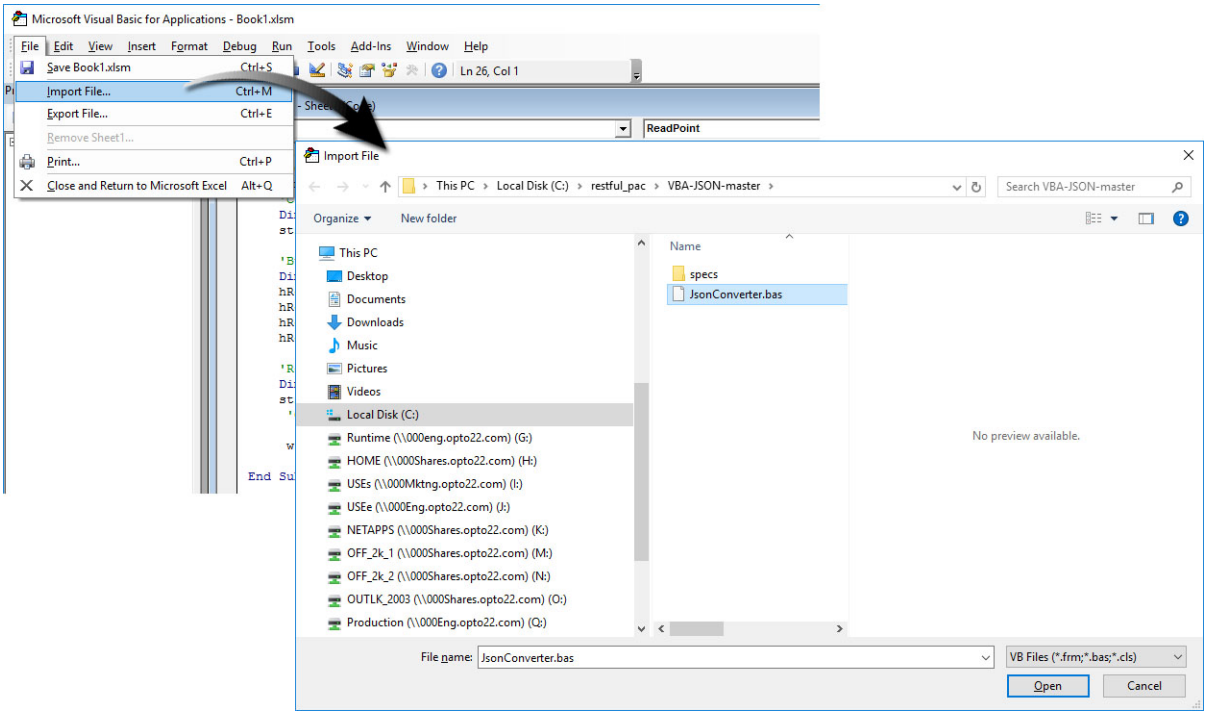


The return string's value varies in type according to the type of data returned. This example returned a float variable that needs to isolated from the string and converted to a standard number.

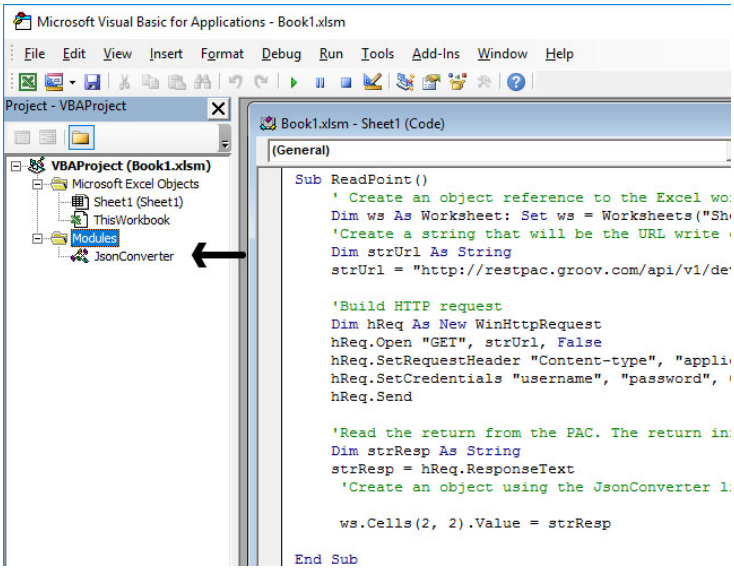
Parse the JSON string

Import the JsonConverter module.

1. In the VBA Editor, choose File > Import File.
2. Navigate to the file JsonConverter.bas, select it, and choose Open.



The JsonConverter module appears in the Project tree. Expand the Modules folder to see the JsonConverter. You do not need to open JsonConverter for this tutorial.

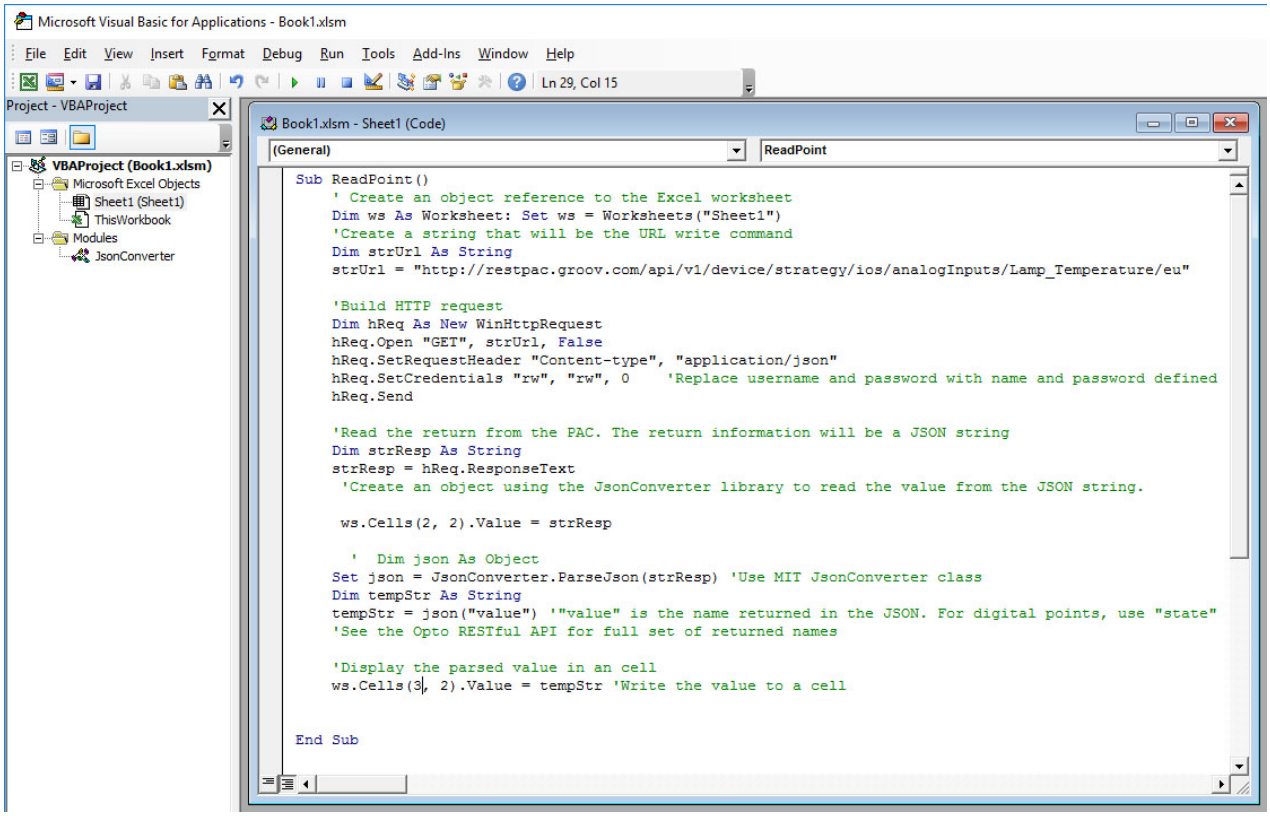


3. Add the following code at the end of your subroutine (macro) within the body of the macro:

```
'Create an object using the JsonConverter library to read the value
from the JSON string.
Dim json As Object
Set json = JsonConverter.ParseJson(strResp) 'Use MIT JsonConverter
class
Dim tempStr As String
tempStr = json("value") '"value" is the name returned in the JSON.
For digital points, use "state"
'See the Opto RESTful API for full set of returned names
'Display the parsed value in an cell
ws.Cells(3, 2).Value = tempStr 'Write the value to a cell
```

4. Save all changes.

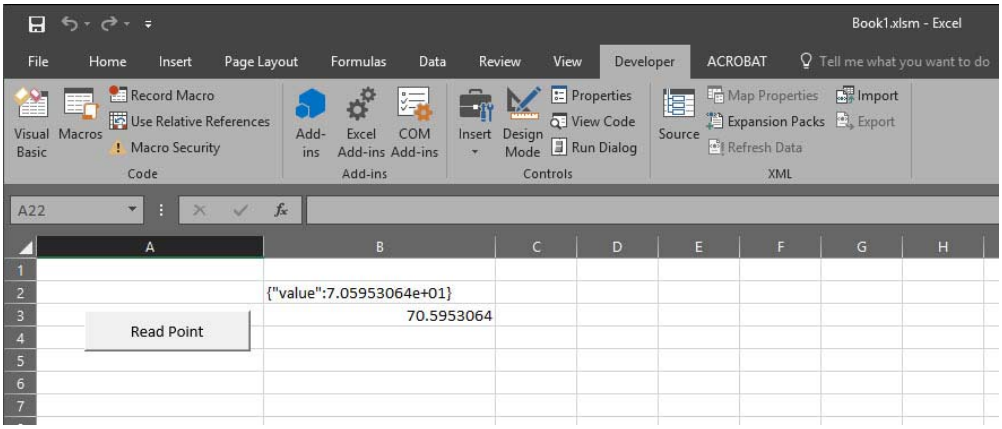
   Your macro should appear as shown here:



Test

1. Return to the Excel sheet.
2. Click Read Point.

   Cell 3B displays the parsed value:



For Further Exploration

The Excel sample OptoRESTfulPAC.xlsm contains interactive spreadsheets to build any read or write command supported by your PAC. Also in this file is a sample chart that plots data on a time interval. These samples construct the URL paths and parse the various forms of data returned, in addition to writing to the PAC.