

Automating Spectral Measurements

Fred T. Goldstein*

FTG Software Associates, P.O. Box 579, Princeton, NJ USA 08542

Copyright 2008 Society of Photo-Optical Instrumentation Engineers (SPIE)

ABSTRACT

This paper discusses the architecture of software utilized in spectroscopic measurements. As optical coatings become more sophisticated, there is mounting need to automate data acquisition (DAQ) from spectrophotometers. Such need is exacerbated when 100% inspection is required, ancillary devices are utilized, cost reduction is crucial, or security is vital. While instrument manufacturers normally provide point-and-click DAQ software, an application programming interface (API) may be missing. In such cases automation is impossible or expensive.

An API is typically provided in libraries (*.dll, *.ocx) which may be embedded in user-developed applications. Users can thereby implement DAQ automation in several Windows languages. Another possibility, developed by FTG as an alternative to instrument manufacturers' software, is the ActiveX application (*.exe). ActiveX, a component of many Windows applications, provides means for programming and interoperability. This architecture permits a point-and-click program to act as automation client and server. Excel, for example, can control and be controlled by DAQ applications. Most importantly, ActiveX permits ancillary devices such as barcode readers and XY-stages to be easily and economically integrated into scanning procedures. Since an ActiveX application has its own user-interface, it can be independently tested. The ActiveX application then runs (visibly or invisibly) under DAQ software control.

Automation capabilities are accessed via a built-in spectro-BASIC language with industry-standard (VBA-compatible) syntax. Supplementing ActiveX, spectro-BASIC also includes auxiliary serial port commands for interfacing programmable logic controllers (PLC). A typical application is automatic filter handling.

Keywords: spectrophotometer, quality control, data acquisition, spectrometer, automation, software

1. INTRODUCTION

Optical coating technicians normally measure a single witness sample from each coating run. In many cases the witness reliably characterizes the entire batch; but as spectral requirements become more critical and filters more selective, it eventually becomes necessary to measure each part. It is our view that custom automation solutions can be implemented for relatively low cost if off-the-shelf hardware and software components are utilized.

We discuss relatively simple and straightforward automation ideas which can be implemented by coating engineers and technicians who are not specialists but who have, or are willing to acquire, sufficient familiarity with *Microsoft* Excel VBA (Visual Basic for Applications) to follow and adapt the examples in this paper.

1.1 Need for automation

Consider a requirement to measure circular variable filters in one degree increments and store results in Excel. Manually, this would be accomplished by moving the filter with a rotation stage, clicking *Scan* followed by *Save As*. Each file is opened, data copied to the clipboard and pasted into Excel. A motor-driven rotation stage with RS-232 controller is assembled and a small program in Visual Basic is created to rotate the stage (easy since the vendor provides examples). The next step is to integrate code for automatic scans: rotate, scan and export to Excel. This might be accomplished in a macro language within the spectrophotometer software; another possibility is to use Excel's VBA language as client and the spectrophotometer software as server. Note that automation does not only apply to the spectrophotometer and rotation stage; it also applies to entering transmittance values in Excel columns.

What if there is no way to control the spectrophotometer except by point and click? In that case automation is impossible. One can only hope that measurement technicians are not confused by a sequence involving 360 scans followed by 360 copy-and-paste Excel operations.

*fredg@ftgsoftware.com; phone +1 609 924-6222; fax +1 609 482-8060; <http://ftgsoftware.com>

Automation also applies to manual scanning procedures. An example is the characterization of architectural windows with heat-reflecting coated plastic film applied to one surface. Three measurements are required from 250 to 2500 nm: reflectance from filmed and unfilmed surfaces and transmittance. The resultant spectra are exported in a file compatible with the Lawrence Berkeley Window program that computes solar performance. Training time and costs are minimized by restricting user options and replacing printed instructions by ‘What to do next’ dialogs. Instrument manufacturers who tout the advanced capabilities of their measurement software (*i.e.* Process menu with 25 functions), often miss the point that some users are better served with a program that starts by displaying a ‘Click OK when ready’ button.

1.2 Background

We originally developed spectrophotometer DAQ software as an alternative to programs offered by instrument vendors. Our programs had several advantages: compatibility with thin film design software, same software for diverse instruments, correct treatment of reflectance, ability to display and store IR spectra in microns.

Automation was not a consideration until the 1990’s when we were contacted by an oil company who had developed a method for characterizing oils by UV spectroscopy. Oil is dissolved in a solvent and the mixture pumped into a spectrophotometer cell. A barcode reader enters sample ID and an electronic balance records sample and solvent weights. The pumping system had been designed and could be controlled by serial commands. Spectral analysis would be performed by integration with *Galactic’s* (now *Thermo Galactic*) GRAMS application.

Our task was to automate the procedure. One solution would be a single program controlling the spectrophotometer and other devices. That approach was deemed unsupportable. At the same time (1994) Sax Software began distributing its Sax Basic VBX which added programming capabilities to Visual Basic 3.0 applications. We could easily add specialized spectro-BASIC commands like *Scan* to a fairly standard and familiar macro language. We implemented separate DDE (see Section 1.3) server programs (now ActiveX EXE servers) to control the barcode reader, balance, and pumping module. Our approach enabled us to add new capabilities to existing software without need to compile special versions for a single customer. As new instruments (*PerkinElmer* Lambda 9, Lambda 19, Lambda 850) were introduced, we were able to update our scanning software without changing spectro-BASIC programs. Replacing a barcode reader with a newer model only required changes in a module, not in the host scanning application.

More recently it was realized that technicians wasted too much time waiting for pumping sequences and UV scans to complete. Further automation was achieved with the addition of an autosampler (essentially a liquid-sampling robot) holding 135 liquid samples. Labor costs were dramatically lowered by implementing overnight unattended scans.

1.3 API possibilities

In our experience, spectrophotometers are often selected according to optical performance and price with little regard to the abilities of controlling software and automation possibilities. Purchasers interested in automation need to ask about the ‘application programming interface’ (API). There are several ways an API can be provided. These are listed below and discussed in Section 2 in some detail.

- Communications interface: low-level commands and responses via RS-232, GPIB, etc.
- Dynamic linked library (DLL), OLE control extension (OCX), out-of-process EXE server
- Dynamic data exchange (DDE, obsolete but still useful)
- Macro language
- Sending keystrokes (not recommended)
- ActiveX EXE application

These are loosely, perhaps erroneously, referred to as drivers. Certain instruments, such as diode-array spectrometers, are intended for remote use and include drivers with examples for Excel, Labview, Visual Basic, etc. Bench instruments are more problematic and some provide no automation support at all. In any case, a purchaser must ascertain the level of API support provided by the spectrophotometer manufacturer; sometimes perseverance is required to communicate with a company representative who appreciates and understands your requirements. Those who find the next section somewhat obscure might now contemplate our conclusion: if a spectrophotometer can be automated by *Microsoft* Excel, its controlling software includes a well implemented API

2. API OPTIONS IN DETAIL

There are hierarchies of API facilities, ranging from simple DOS programs to software embedded within other software. Terminology and methods overlap. For example, an ActiveX EXE might be an end-user application, like Excel, which can also run in the background, or it might be a hidden server like MSLambdaServer (handles communication with *PerkinElmer* Lambda 650-1050 instruments). Ultimately the hidden server is utilized by the end-user application.

2.1 Communications interface

A communications interface specifies commands such as "\$GO 550" which, when sent via RS-232, slews a Lambda 9 instrument to 550 nm. Upon arriving at 550 nm the instrument returns error code "0000" (indicating no error). This is the minimum level of support that an instrument manufacturer can provide. In general, these commands are intended for software developers implementing scanning applications and/or higher level drivers. FTG Software has utilized interface commands to create ActiveX EXE applications controlling a number of instruments.

Interface commands communicate directly with the instrument's firmware and can potentially damage the instrument. In one spectrometer, slewing to zero wavelength is useful for alignment; a similar command in another instrument destroys the gears. Nevertheless, a communications interface may be all that is needed when the same scan is repeated again and again, with no need for menus, dialogs, graphics, etc. The following GWBASIC code (popularly known as *Gee-Whiz Basic* and freely available on the Internet) for PE Lambda 2/10 and similar scans from 400 to 800 nm x 1 nm and stores data in a text file. Program communications were tested in Windows XP and Vista. (Lines 51 and 52 are not a mistake. The scan command \$SC results in an acknowledgement followed by a status line.)

Table. 1. GWBASIC code for scanning PE Lambda 2.

```
10 OPEN "COM1:4800,N,8,1" AS #1
11 PRINT #1, "RE 0" & CHR$(17)
12 LINE INPUT #1, R$
20 PRINT #1, "$AL 400"
21 LINE INPUT #1, R$
30 PRINT #1, "$AH 800"
31 LINE INPUT #1, R$
40 PRINT #1, "$DI 1"
41 LINE INPUT #1, R$
50 PRINT #1, "$SC"
51 LINE INPUT #1, R$
52 LINE INPUT #1, R$
60 OPEN "DATA.TXT" FOR OUTPUT AS #2
70 FOR I = 1 TO 9
80 LINE INPUT #1, R$
90 PRINT #2, R$
100 NEXT I
110 CLOSE #2
120 CLOSE #1
130 END
```

Assuming that the instrument manufacturer has provided no other API support, the existence of a documented communications interface indicates that automation will be possible. Care must be taken with legacy instruments (no longer supported by the manufacturer), even if documentation is available. While RS-232 interfaces remain compatible with today's fast computers, it is our understanding that *PerkinElmer* Lambda 4/6 and *Varian* Cary 2200/2400 IEEE-488 (GPIB) models can only be driven by the 286 computers originally supplied with those instruments.

If no communications protocol is available, reverse engineering with serial monitoring software is an option. By studying log files of commands to and from an instrument it may be possible to deduce enough of the interface protocol to implement automatic scans. We have, in fact, used serial monitoring to enhance our understanding of poorly documented interfaces. Finally (scraping the bottom of the barrel of possibilities) automation might be implemented by sending keystrokes to the scanning application. This is programming of last resort, but may just save the day. Any trick that works is acceptable, even 'handshaking by stopwatch' wherein the scan command is followed by a known delay.

2.2 DLL/OCX/EXE servers

The next level of support encapsulates control commands in a server component. That is, the *server* runs the instrument and delivers spectral data to the main application or *client*. This minimizes programming tasks. A server normally provides a SCAN command which relieves the programmer of low-level communication details.

Windows includes a great number of libraries; typically there are thousands of DLL files in c:\Windows\System32 and elsewhere. Unlike libraries that are linked when programs are compiled, a DLL is linked when the program is run. The same DLL can be used by more than one application. DLLs may themselves depend on other DLLs. When a dependent file is missing, we may experience *DLL Hell*. An OCX control is similar to a DLL and often includes a visible object such as a data entry grid or graphics window. Most typically OCX's are utilized in programmer toolkits; for example, our software makes heavy use of ProEssential's pesgo32.dll scientific graphics engine.

An example of an invisible OCX control is Cary32.ocx for running Varian Cary spectrophotometers. The following VB6 code snippets illustrate procedures for initiating a scan and capturing data. It is not necessary to fully understand this code to appreciate that it far simpler than controlling the instrument directly via its GPIB instrument bus. Complex handshaking and control details are offloaded to the instrument manufacturer. The following code is incomplete and only intended to show main features, but it should be clear that it is quite easy to implement. Once again, if the same scan is required over and over, there is no need for an elaborate user interface.

Table. 2. VB6 code snippets to run Varian Cary spectrophotometers with Cary32.ocx.

```
With cary1
    .caryStop
    .UVVisInterval = -1
    .ScanStart = 800
    .ScanStop = 400
    .ScanMode = scanWavelengthSNR
    .CarySetup
    .caryWaitNotBusy
    .CaryStart           ' start scanning
    .carywaitnotscanning ' can be triggered by STOP key via .CaryStop
End With

Private Sub Cary1_CaryDataMessage(ByVal y As Single)
    If cary1.ScanMode = scanWavelengthSNR Then
        While cary1.caryRead(w, t1, t2)
            yData(k) = t1
            k = k + 1
        Wend
    End If
End Sub
```

OCX and DLL servers are considered to run *in-process*, and do not appear in Windows Task Manager. Another option is the ActiveX EXE *out-of-process* server. This is an executable program with or without its own user-interface running outside the application. The program name appears in Windows Task Manager under Processes but not under Applications. If there is no user interface, running the program by itself usually does nothing useful. DLL/OCX/EXE server solutions are all a significant step up from a firmware communications interface. From a developer's point of view all server variations can be made to work. What is important is that the server is well supported and documented and includes examples in several programming languages, such as Visual Basic, Excel, LabView, C++.

As mentioned above, an OCX can optionally include a visible object such as a graphics window. We made use of this several years ago when *Galactic Industries* (now *Thermo Galactic*) published their *My Instrument* specification whereby instrument scanning could be integrated within their GRAMS application. While designed for GRAMS, our OCX also worked in other applications. The screenshot below illustrates how this OCX immediately adds scanning capabilities to *Microsoft Excel*. In the Excel Visual Basic for Applications (VBA) editor, one need only add a user-form, add the scan control from the tools collection and resize it to fit the form.

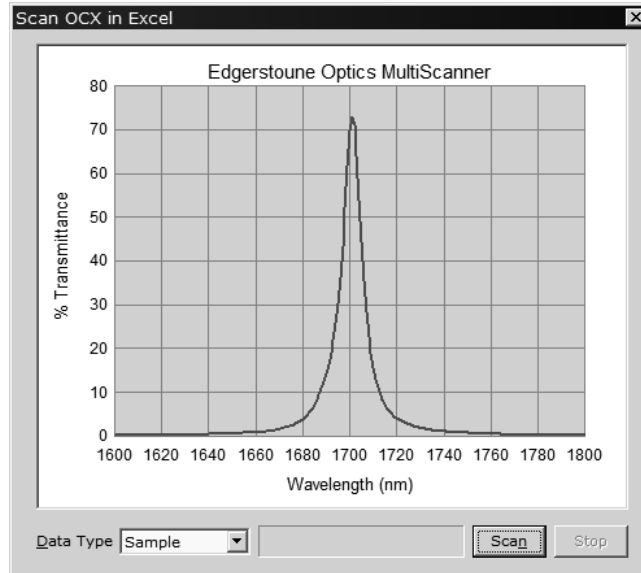


Fig. 1. Scanning control Ope983.ocx as it ultimately appears in *Microsoft Excel*.

The object browser shows the scan commands now available in Excel VBA.

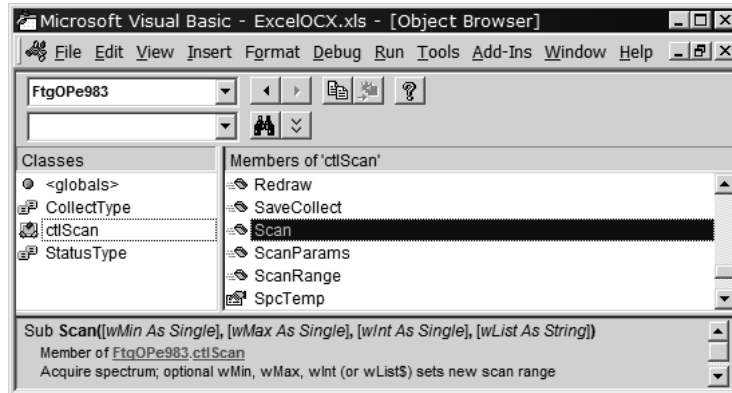


Fig. 2. Ope983.ocx adds spectrophotometer scanning capabilities to *Microsoft Excel*.

All that is needed is minimal code to transfer data to the spreadsheet.

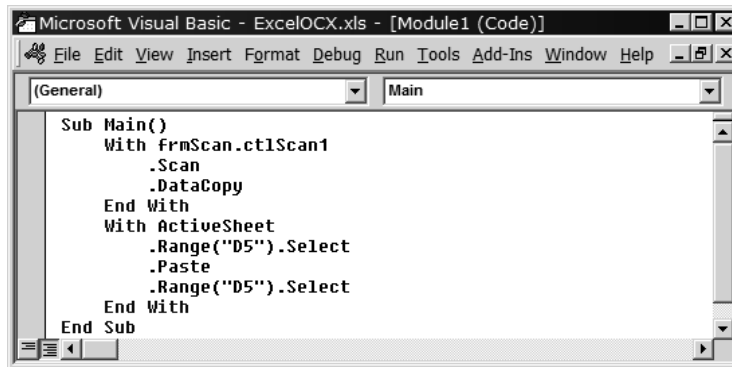


Fig. 3. Ope983.ocx adds spectrophotometer scanning capabilities to *Microsoft Excel*.

My Instrument was largely ignored by the spectroscopy industry. Due to lack of interest, we stopped supporting OCX versions in favor of ActiveX EXE applications (discussed below).

2.3 DDE servers

Dynamic data exchange has been replaced by newer methods for transferring data and commands between programs. It is still useful, however, and provides means to transfer data between 16 and 32-bit software. FTG Software has integrated a 16-bit PerkinElmer GX (FTIR) server into our 32-bit scanning applications. A 16-bit server for Lambda 19 is also available, but requires Windows 98 to operate the Lambda 19 transputer board. A problem with DDE commands is that the client program proceeds directly to the next statement without waiting for the previous statement to be executed by the server program. This is worked around with handshaking, for example waiting for a server text box to change from BUSY to READY. If the server application is storing a file on disk, checking that the file date and time have been updated is another handshaking method.

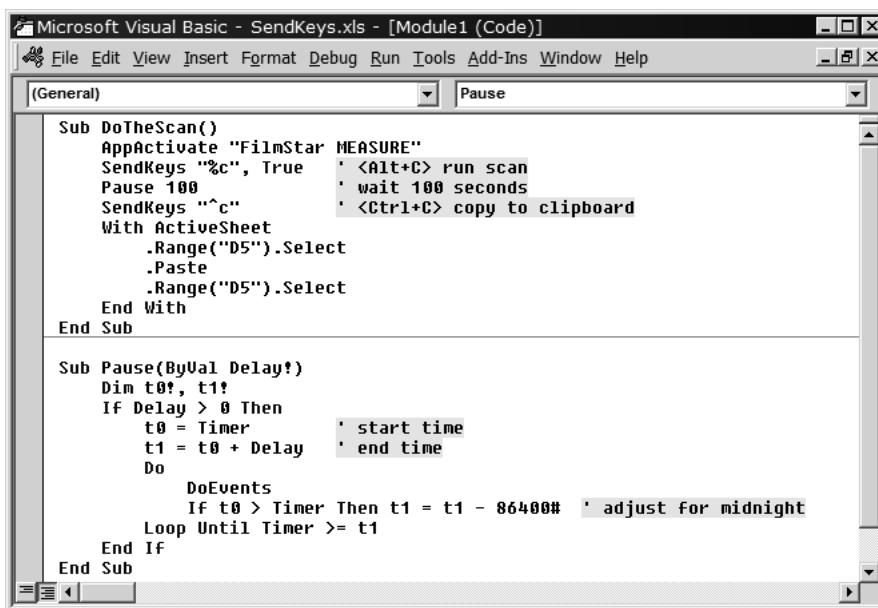
2.4 Macro language

Scanning applications may include built-in programming languages, sometimes called macro languages or procedure builders. Our own scanning applications include a VBA-clone to which scanning functions have been added. The underlying language with familiar syntax is supplemented by keywords devoted to host program capabilities.

The existence of a macro language does not imply consistency. *Thermo Scientific* Nicolet OMNIC software includes a non-standard graphics macro builder while *Varian Cary* WinUV software offers its VBA-like Applications Development Language (ADL). Potential users need to determine whether these macro languages support external applications (as required for hardware automation) or are limited to sequencing built-in functions. Note that VBA-compatibility does not necessarily include client and server capabilities. A scanning application might act as client (control other applications), but not as server (controlled by other applications). This may or may not be important.

2.5 Sending keystrokes

Sending keystrokes to the scan application is a possibility for instruments with no API. Assuming the application is amenable to receiving keystrokes (at least enough to scan and save data), the main problem (similar to DDE) is handshaking. Note the use of a *Pause* statement in the code below. While 'handshaking by stopwatch' is not elegant, it can do the job. In this example we copy the spectrum to the clipboard. If that is impossible, file transfer might work. As with DDE transfer, handshaking is also possible by monitoring the spectral file's timestamp. Using Excel VBA, it is quite easy to experiment with SendKeys.



```
Microsoft Visual Basic - SendKeys.xls - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
(General) Pause
Sub DoTheScan()
  AppActivate "FilmStar MEASURE"
  SendKeys "%c", True ' <Alt+C> run scan
  Pause 100 ' wait 100 seconds
  SendKeys "^c" ' <Ctrl+C> copy to clipboard
  With ActiveSheet
    .Range("D5").Select
    .Paste
    .Range("D5").Select
  End With
End Sub

Sub Pause(ByVal Delay!)
  Dim t0!, t1!
  If Delay > 0 Then
    t0 = Timer ' start time
    t1 = t0 + Delay ' end time
    Do
      DoEvents
      If t0 > Timer Then t1 = t1 - 86400# ' adjust for midnight
    Loop Until Timer >= t1
  End If
End Sub
```

Fig. 4. Excel VBA code illustrating the use of SendKeys.

2.6 ActiveX EXE solutions

There are two types of executable programs (EXE) in Windows: standard and ActiveX. The difference is that an ActiveX application expose its objects to other Windows programs and can be run in the background. ActiveX applications utilize COM (component object model) to transfer data and commands between Windows programs. Note that terms such as ActiveX, OLE, OCX, etc. are often lumped under the term COM. Adding to the confusion is the fact that an ActiveX EXE server can be utilized (much as an OCX) in a standard or ActiveX EXE end-user application.

An ‘object’ is a diffuse term best explained by an example that readers with Windows can try. In *Microsoft Excel*, access the Visual Basic editor (Alt+F11) and select Tools...References. Find and check *Microsoft Word...Object Library*. Now click View...Object Browser (F2) and select Word in the upper left list box. It is unlikely, of course, that anyone would want to check Word spelling by running a macro in Excel, but it is possible! The point is that Word objects are now available in Excel. This is extremely useful and not well known to most Windows users.

How is all this implemented? Windows programs include forms (dialogs, etc.), code modules (subroutines, etc.) and class modules. A *class* is an object that can be exposed to other applications. VBA (Visual Basic for Applications) provides means to utilize such objects in all *Microsoft Office* applications. The Word VBA subroutine CheckSpelling might appear in Excel as WordObj.CheckSpelling. This is similar to the OCX formalism in the previous section. One difference is that an OCX always requires a host application, while an ActiveX EXE application can run by itself.

Suppose we want to run a spectrophotometer from Excel using our ActiveX MEASURE scanning application Mpe983.exe as server. We first switch to VBA and add a reference to FtgMpe983.exe in Tools...References

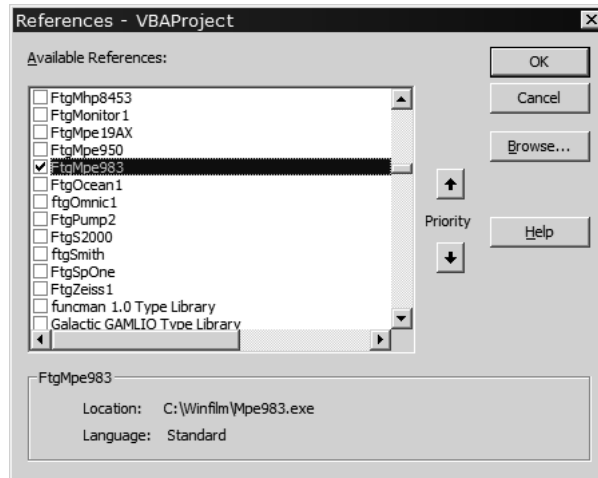


Fig. 5. Adding a reference to Mpe983.exe via Tools...References in the Excel VBA editor.

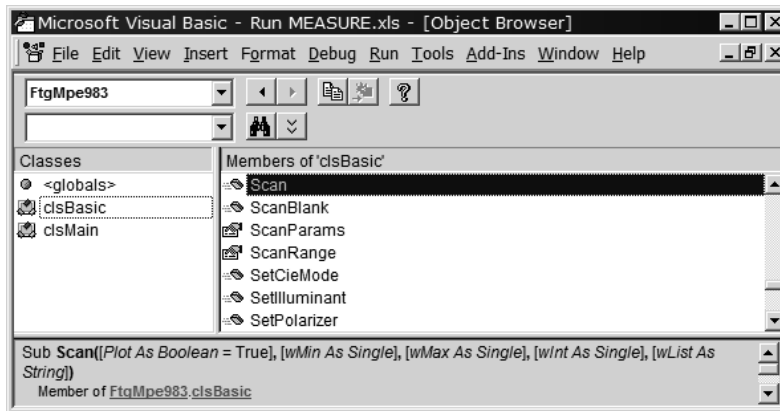


Fig. 6. Members of ‘clsBasic’ (subroutines, functions and properties) that are now available in Excel VBA.

We have now exposed Mpe983 clsBasic class objects to Excel. These are the same functions we could access directly in Mpe983 via spectro-BASIC. We now appreciate that there are several ways to accomplish the same tasks when both programs (Mpe983 and Excel) can be treated as client or server.

```

Microsoft Visual Basic - Run MEASURE.xls - [GetSpectrum (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
(General) (Declarations)
Dim mBasic As FtqMpe983.clsBasic
Sub DoTheScan()
  Set mBasic = New FtqMpe983.clsBasic
  mBasic.Scan
  mBasic.DataCopy ' copy to clipboard
  With ActiveSheet
    .Range("D5").Select
    .Paste
    .Range("D5").Select
  End With
End Sub

```

Fig. 7. Code to run the spectrophotometer and place the spectrum in an Excel worksheet.

The inverse process, illustrated in Figure 8, uses our Mpe983.exe as client (via spectro-BASIC) and Excel as server. The macro scans, copies data to the clipboard, pastes the data and enters Excel formulas for average and standard deviation. Note that we first had to establish a reference to the *Microsoft* Excel object library in Edit...References.

```

FilmStar BASIC - C:\Winfilm\Basic32\Scan Calculate in Excel.bas
File Edit View Macro Debug Sheet Help
Object: (General) Proc: ExcelCalc
1 ' Scan Spectrum in MEASURE and compute
  ' Reflectance Average and Std Dev in Excel
  Dim nData As Integer
  Sub Main
    ReDim xd(1) As Single
    Scan ' do the scan
    xd = Spectrum_X: nData = UBound(xd) ' get number of data points for Excel range
    DataCopy: ExcelCalc ' copy spectrum to clipboard
  End Sub
  Private Sub ExcelCalc()
    Dim xlApp As Excel.Application
    Dim xlBook As Excel.Workbook
    Dim xlSheet As Excel.Worksheet
    Set xlApp = New Excel.Application
    Set xlBook = xlApp.Workbooks.Add
    Set xlSheet = xlBook.Worksheets.Add
    xlApp.DisplayAlerts = False ' no save prompt upon quitting Excel
    xlSheet.Paste Destination:=xlSheet.Range("A5") ' paste spectrum at A5
    xlSheet.Range("A1").Formula = "=AVERAGE(B5:B" & CStr(4 + nData) & ")"
    xlSheet.Range("A2").Formula = "=STDEV(B5:B" & CStr(4 + nData) & ")"
    Ave$ = Format$(xlSheet.Range("A1").Value, "0.00000")
    Dev$ = Format$(xlSheet.Range("A2").Value, "0.00000")
    MsgBox "Ref1 Average = " & Ave$ & vbCrLf & "Ref1 Std Dev = " & Dev$, vbOkOnly, "Excel Calculation"
    xlApp.Quit ' quit Excel
    Set xlApp = Nothing ' release objects
    Set xlBook = Nothing
    Set xlSheet = Nothing
  End Sub

```

Fig. 8. spectro-BASIC program (client) to scan and compute average and standard deviation in Excel (server).

While it is ideal if instrument software can act as both client and server, the server function is really more important. An example is provide by *Thermo Scientific's* Omnic software for Nicolet FTIR spectrometers. Their software is a model that should be emulated by other manufacturers. An FTG user recently acquired a Nicolet 380 on our guarantee that it could use the same software already controlling a *PerkinElmer* 983 double-beam spectrophotometer. Following our previous ActiveX server examples we add OmnicCom as a reference in the Excel VBA editor.

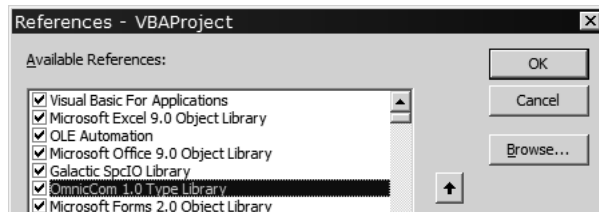


Fig. 9. Adding a reference to omnic32.exe via Tools...References in the Excel VBA editor. The Galactic SpcIO Library simplifies file transfer from OMNIC to Excel.

```

Microsoft Visual Basic - OmnicExcel.xls - [basOmnic (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
[General] RunOmnic

Option Explicit
' References to Galactic SpcIO library and OmnicCom Type Library
Dim SpcIO As GSpclLib.GSpcIO ' SpcIO object for loading spectrum (Thermo Scientific)
Dim OmApp As OmnicComLib.OmnicApp ' Omnic object (Thermo Scientific)
Const fData$ = "C:\Windows\Temp\OmnicTemp.spc"
Const fParam$ = "C:\Program Files\Omnic\Param\SingleBeam.exp"
Dim nPts%, Wave!(), Back!(), Samp!()

Sub RunOmnic(ByVal kType) ' kType = 1 Background (Reference) 2 Sample
    Dim i%, qte$, cmd$
    qte$ = Chr$(34) ' Double quote character
    If kType = 1 Then
        MsgBox "REMOVE SAMPLE, then click OK", vbExclamation, "Reference Scan"
        cmd$ = "Invoke CollectBackground Auto"
    Else
        MsgBox "INSERT SAMPLE, then click OK", vbExclamation, "Sample Scan"
        cmd$ = "Invoke CollectSample Auto"
    End If
    Set OmApp = New OmnicComLib.OmnicApp
    With OmApp
        .ExecuteCommand "RestoreWindow"
        .ExecuteCommand "ShowToolBar Off" ' looks neater
        .ExecuteCommand "LoadParameters " & qte$ & fParam$ & qte$
        .ExecuteCommand cmd$
        .ExecuteCommand "Export " & qte$ & fData$ & qte$
        .ExecuteCommand "Clear"
        .ExecuteCommand "MinimizeWindow"
    End With
    SpcFileOpen kType
    If kType = 2 Then
        For i = 1 To nPts
            shSpectrum.Cells(i, 1) = Wave(i - 1)
            shSpectrum.Cells(i, 2) = Samp(i - 1) / Back(i - 1)
        Next i
    End If
End Sub

Sub SpcFileOpen(ByVal kType)
    Dim i%, xFirst!, xLast!, xDelta!
    Set SpcIO = New GSpclLib.GSpcIO
    SpcIO.OpenFile fData$
    nPts = SpcIO.NumPoints: xFirst = SpcIO.FirstPoint: xLast = SpcIO.LastPoint
    If kType = 1 Then
        ReDim Wave(0 To nPts - 1), Back(0 To nPts - 1)
        xDelta = (xLast - xFirst) / (nPts - 1)
        For i = 0 To nPts - 1
            Wave(i) = xLast - (nPts - i - 1) * xDelta
        Next i
        Back = SpcIO.YPoints
    Else
        ReDim Samp(0 To nPts - 1)
        Samp = SpcIO.YPoints
    End If
End Sub

```

Fig. 10. Excel VBA code to perform reference sample scans and put results into worksheet cells.

Our next task is to understand, from examples or documentation, how to communicate with the instrument. In Omnic nearly everything is handled via ExecuteCommand. As seen in the above code listing, few commands are required. Note

the *LoadParameters* command. It is simpler to specify setup parameters (*.exp) stored on disk rather than set parameters from a macro. Duplicating complex FTIR setup dialogs in Excel would be foolish. Since we already had the Galactic-supplied I/O library (freely available from *Thermo Scientific*), transferring spectra was best accomplished by SPC file transfer. Another option is CSV file transfer. The next screen shows what the end-user experiences.

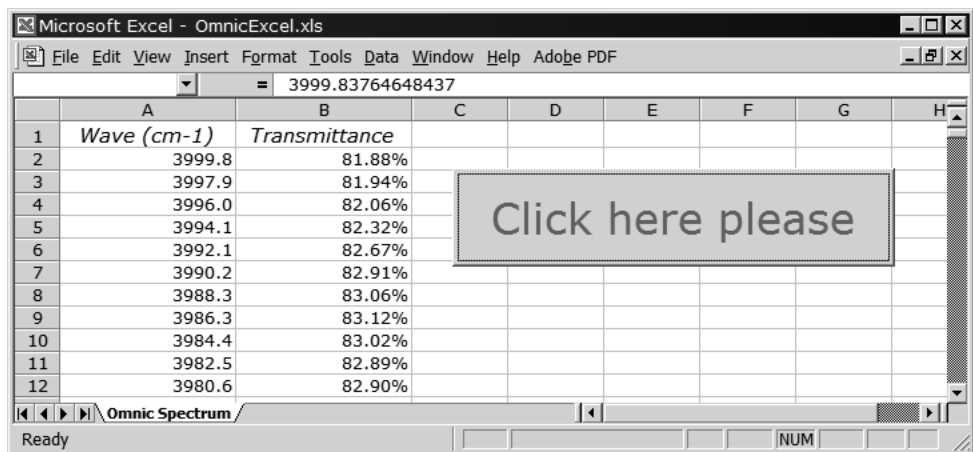


Fig. 11. The user sees only a button to push and needs no understanding of VBA or Omnic software.

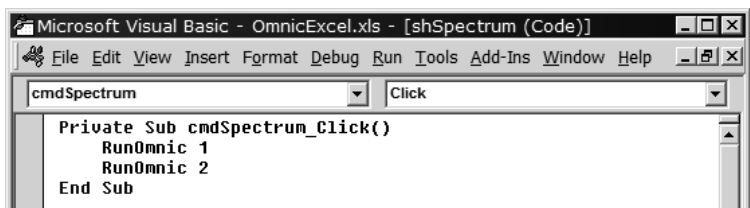


Fig. 12. Code executed by activating the 'Click here please' button.

The next screenshots (Figures 13-15) illustrate the use of Omnic.exe as a server for FTG application Mpe983.exe.

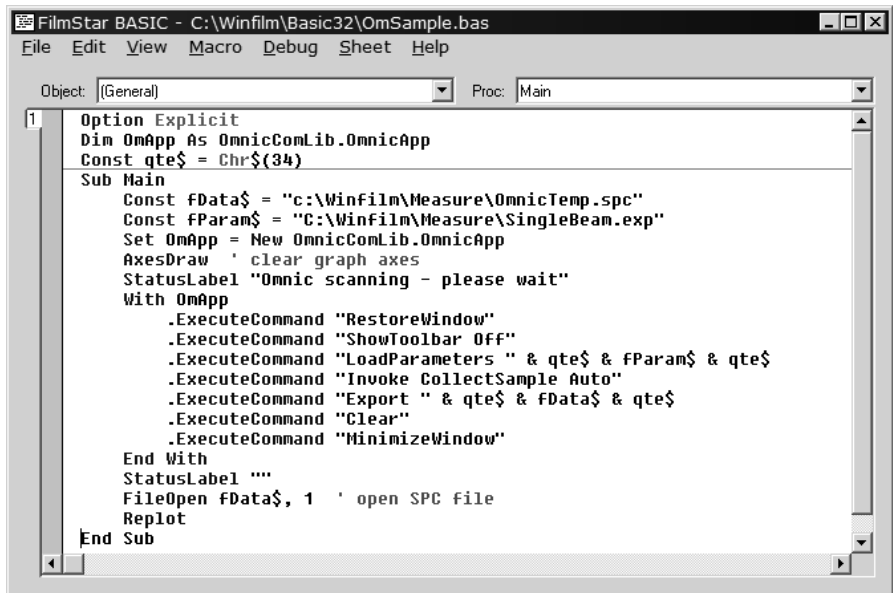


Fig. 13. spectro-BASIC program to acquire data using Omnic. Note that the data acquisition code is virtually identical to the Excel example in Figure 10. Here the code is shorter because FileOpen already includes SpcIO functions.

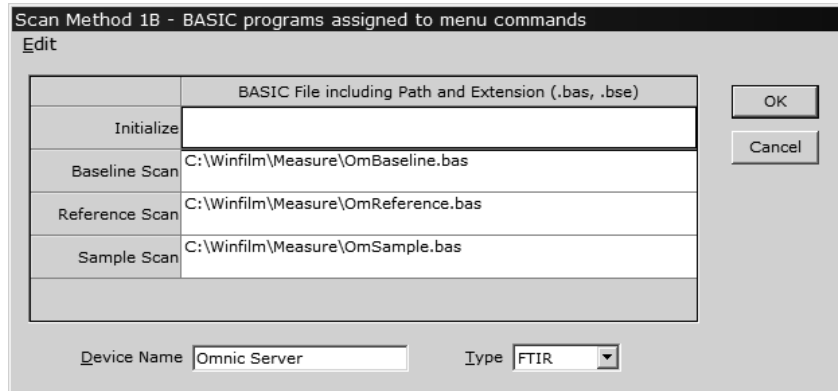


Fig. 14. Assigning spectro-BASIC scan commands to the Scan button.

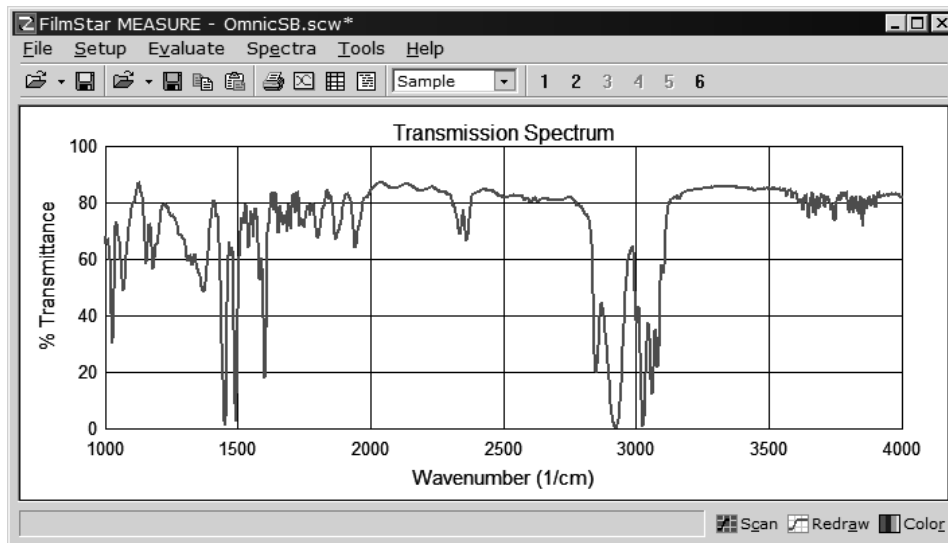


Fig. 15. Omnic-acquired test spectrum as it finally appears in our application. The spectro-BASIC program shown in Figure 13 is triggered by clicking the Scan button. Technicians need have no knowledge of BASIC or Omnic.

If we had used a double-beam *PerkinElmer* 983 instead of the *Thermo Scientific* Nicolet 380, the following screen would have appeared instead of Figure 14.

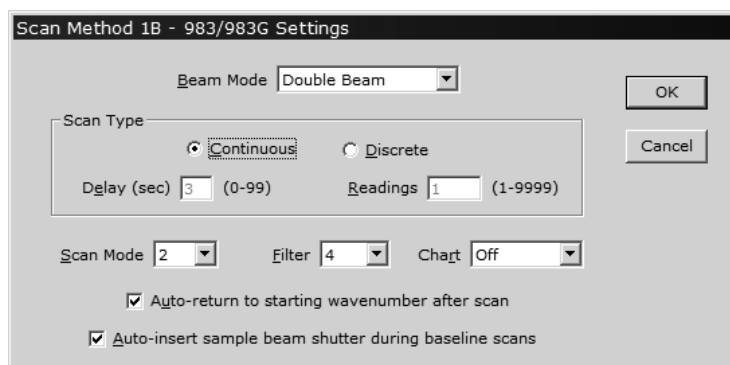


Fig. 16. *PerkinElmer* 983 Scan Method dialog. A corresponding screen for a more complex instrument such as a PE Lambda 950 will have far more options.

3. DISCUSSION

Given all the possibilities, we summarize our efforts in implementing spectrometer software which readily lends itself to automation. Essentially, we have escalated various API implementations into consistent end-user applications.

3.1 Scanning applications

Our applications are ActiveX and communicate with *Microsoft* Office and other Windows applications at the object level. Several versions support different instruments, but all have the same appearance, features, and automation capabilities. From the user's point of view, operating a legacy double-beam *PerkinElmer* 983 is virtually identical to operating their new Spectrum 100 Optica FTIR (except for the time required to scan, of course). The end-user is hardly aware that we use serial commands for the 983 and an out-of-process server for the FTIR. Since our applications are client/server they can both control and be controlled by external software.

Our scanning applications include a test mode whereby procedures can be tested offline with meaningful spectral data. Here the program reads a data file instead of actually scanning. Sometimes test mode is referred to as *soft instrument*. Surprisingly test mode is not always provided in control software; in our opinion that is a glaring omission.

3.2 Automating procedures

Our built-in VBA-like spectro-BASIC macro language supports custom dialogs and prompts in any Windows-supported language, thereby ensuring that complex procedures are followed. During macro execution users are locked out of program menus. In addition an Administrator is able to prevent users from editing spectrophotometer methods. A startup macro provides complete customization, turning highly flexible software into single-purpose procedures. The following prompt is taken from the solar performance calculation mentioned in the Introduction.

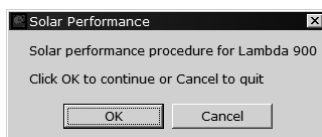


Fig. 17. 'What to do next' dialogs replace complex menu choices.

The macro language also supports external DLLs. This provides seamless means to include complex spectral calculations in macros. Our main use of this was the inclusion of GRAMS partial least squares prediction routines in oil analysis. Existing DOS (BASIC, Fortran, C, etc.) code can often be utilized by recompiling as a DLL instead of an EXE.

3.3 Ancillary hardware

A barcode ActiveX EXE provides an example of our approach to ancillary devices. Once tested in standalone mode, the application runs as an out-of-process server under macro control and ultimately appears to be built into the scanning software. Most importantly, these programs also include a test mode.



Fig. 18. Barcode reader window appears as if built into the scanning application.

At the high end of complexity, laboratory-scale 'pick and place' robots, such as those manufactured by *Thermo Fisher Scientific*, offer promising automation possibilities. A tray of optics is loaded and the robot places each part in the spectrophotometer beam and waits for the scan. Since these robots are controlled by ActiveRobot™ (ActiveX) software, there is no problem in integrating our scanning applications. To the best of our knowledge, such robots are not often used in optics. One proffered reason is that there are too many shapes. It seems to us, however, that the ability to easily adapt to multiple shapes is an advantage of programmable robots.

Supplementing ActiveX communications our macro language now includes serial port communications, independent of the spectrophotometer control port. This facilitates communications with programmable logic controllers (PLC).

3.4 Ancillary software

Excel is a near-universal and easily automated vehicle for transferring data to optics end-users; most importantly it can run in the background while completely hidden from view. QA technicians can reliably create secure Excel files while unaware that Excel is being used. When mapping filter uniformity, compare the complexity of 100 data files with the simplicity of 100 columns in a single Excel file.

For spectral data to be truly useful, we need to add information about the data. That could range from customer name to coating chamber parameters to calculated color coordinates. Furthermore, there are security and network access issues. The sales department should be able to view data files but not ‘improve’ them. Thin film engineers need to analyze coatings that fail specifications, examine trends in measured spectra, etc.

To this end, we have integrated *FileMaker Inc.*’s FileMaker Pro database. This ActiveX server application permits us to store and retrieve spectra according to numerous criteria. FileMaker is a workgroup (coating department) database which can be integrated with enterprise (company wide) databases. While normally considered to be ‘business’ software, improvements in recent FileMaker versions make it an excellent tool for managing spectral data. FileMaker now includes ActiveX automation, enhanced security options, a container field, and object-level integration with SQL databases. The container field can be used to store pictures as well as complete files. In other words, spectral data files can be stored directly in the database, eliminating the usual confusion of thousands of files.

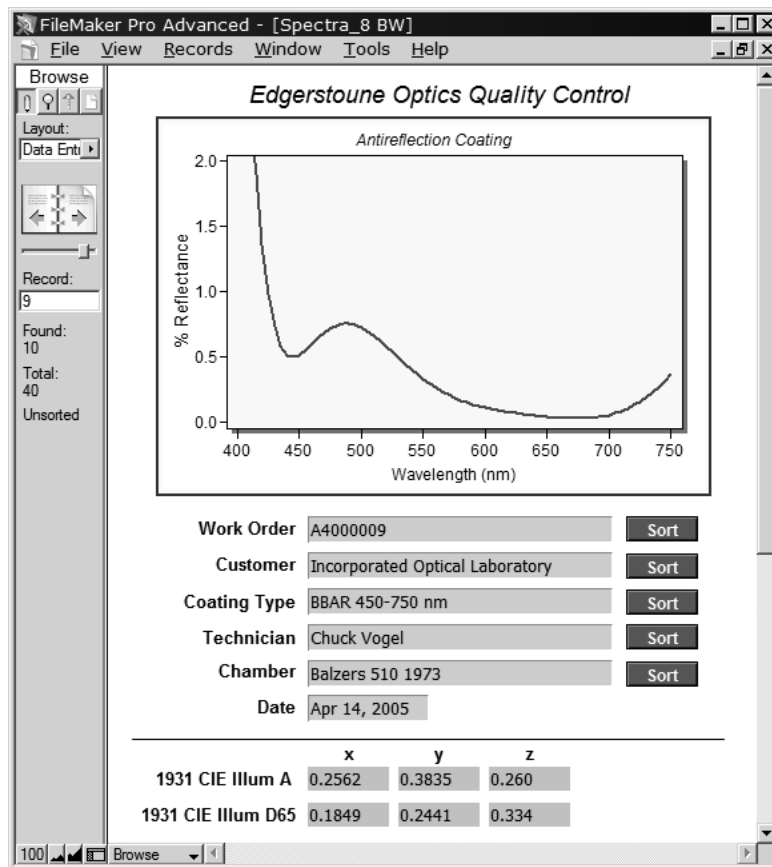


Fig. 19. Prototype FileMaker Pro database, our final step in automation.

As illustrated in Figure 19, our scanning application runs a macro (which runs FileMaker scripts) to automatically create a new blank record, insert the picture, data file, and calculated CIE values into a new database record. Once the record is created it cannot be deleted, nor can another user create a new blank record. (The picture is optional; the actual data is stored separately.) A design engineer, reviewing the day’s spectra and disappointed with the bump at 480 nm, can open the same database, and retrieve the spectrum for analysis. Another possibility, in this case, is to select curves matching a particular color specification, etc.

By adapting a database application with millions of users, companies eliminate numerous unforeseen problems that will likely occur in a custom solution. Given our prototype database as a starting point, how difficult is it to implement a FileMaker solution? As we have seen in actual cases, the task can be accomplished by a summer intern college student with no previous experience in FileMaker or in our data acquisition software.

4. CONCLUSION

Automation should be as straightforward and supportable as possible. We think object-level (ActiveX/COM) Windows compatibility is the key and therefore propose an Excel automation test: *If a spectrophotometer can be readily controlled by Excel and/or Excel controlled by the scanning application, it is likely that automation can be implemented rapidly and cost-effectively.* The code samples given above should be helpful.

4.1 Excel automation test

Most coating facilities have Excel and consequently *Microsoft* VBA. VBA provides ready means to test spectrophotometer software in both client and server modes. A *scan software as server* example is provided in Figures 9-12. Figure 8 displays a *scan software as client* example.

Since Excel is so universal, it is possible that the spectrometer manufacturer already provides examples. Assuming that the manufacturer does not supply sample code, but does support a DLL/OCX/EXE API, a VBA programmer should be able to implement test code similar to Figures 9-12 in a day or two. Third-party developers may offer solutions beyond those provided by the instrument manufacturer. The extra cost of third-party software must be weighed against the amount of time to translate a communications interface into a finished application.

If no API exists, but there are still compelling reasons for choosing an instrument, there remains the possibility of SendKeys automation as illustrated in Figure 4. This requires a copy of the spectrophotometer software set to run in test mode (soft instrument). Unless dealing with a diode-array, FTIR or other rapid-scan device, a test mode is essential for developing automation procedures.

4.2 ActiveX EXE advantages

There are many ways to implement automation. Much depends on the experience of the engineers assigned to the task. In our experience, the use of ActiveX/COM EXE has several advantages:

- Modules may be developed and tested independently. Reusability is enhanced and development costs are less.
- Support issues are minimized because the end-user never modifies scanning software. Fixes due to hardware upgrades, firmware revisions, Windows updates, CPU speed improvements, etc. are the responsibility of the instrument manufacturer or third-party software vendor.
- Complex software like Excel can run in the background, thereby increasing security and reliability.