A Review: Calling FORTRAN From VBA (Excel 2003) Utilizing MicroSoft PowerStation Version 4.0 and Windows XP

Edward M. Rosen
EMR Technology Group

Introduction

Calling FORTRAN code from VBA [1] is useful when it is desired to avoid recoding FORTRAN into VBA or it is desired to take advantage of the greater execution speed of FORTRAN. The steps to call FORTRAN from VBA , however, depend on the FORTRAN compiler and the operating system being used.

If the FORTRAN code is in the form of a main program (.exe file) then the Shell function of VBA is utilized [2] . However, if the FORTRAN code is in the form of a subroutine, then generating a DLL is required [3] .

Previous communications on this topic [2,3] utilized Digital Visual Fortran Professional Edition and DOS. This communication utilizes Microsoft PowerStation Version 4.0 and Windows XP.

The paper is divided into two parts. First the generation of a FORTRAN .exe file is reviewed and an example is given using the Shell function of VBA. Then the generation of a DLL file is detailed followed by an example illustrating how the DLL is called in a VBA procedure.

Creating an .exe File

Creation of an exe file utilizing Microsoft PowerStation (Professional Edition) Version 4.0.

1. Create a FORTRAN main program utilizing a text editor Call it SOURCE1.FOR and place the file in directory C:\TMP (Figure 1)
2. Invoke PowerStation. Click on the MicroSoft Developer Studio
3. Select File from the menu
4. Click on New
5. Select Project Workspace. Click OK
6. Console Application is highlighted. Name the workspace (ggg)
7. Click Create
8. Select File again from the menu
9. Click New
10. Select Text File then OK
11. Select Insert from menu. Highlight ‘ Files Into Project’
12. Double Click on the c:directory

13. Scroll down to highlight TMP. Click OK Highlight Source1.For. Click OK.
14. Select 'build' from the menu. Then select ' build ggg.exe.' Note the compile and link messages in the lower window
15. When the compile and link are completed without an error, copy the ggg.exe file from C:\MSDEV\Projects\ggg\debug\ggg.exe to C:\TMP\ggg.exe

The Source1.For file of Figure 1 (a main FORTRAN program) reads an input file C:\TMP\TSIN1 shown in Figure 2. Calculations (shown in Figure 1) using the input file are carried out and placed into C:\TMP\TSOUT shown in Figure 3.The ggg.exe program can be tested by varying the TSIN1 input file.

The Fortran Program Source1.FOR (Figure 1) is based on illustration 3 page 234 of Hougan and Watson [4].

```
**************************************************************
     Program to read a file, do calculations and output the results
!
      Program  Source1

!     PP(1)  = Heat of Vaporization at NBP  cal/gm
!     PP(2)  = Normal Boiling Point Deg C
!     PP(3)  = Critical Temperature in Deg C
!     PP(4)  = Temperature at which Heat of Vaporization is desired  Deg C
!     Y      = Heat of Vaporization at desired temperature cal/gm
!
      Real  PP(4), Y, Y1, XL1, XBP, XTC, XT, TR1, TR2
!
      OPEN (Unit =9, FILE = 'C:\TMP\TSIN1')
      OPEN (Unit =8, FILE = 'C:\TMP\TSOUT')
!
      READ (9, *, END = 100) (PP(I), I = 1,4)

!     Miscellaneous Calculations
!
100   XL1 = PP(1)
      XBP = PP(2)
      XTC = PP(3)
      XT  = PP(4)
      TR1 = (273+XBP)/(273+XTC)
      TR2 = (273+XT)/(273 +XTC)

       Y1   =  (1-TR2)/(1-TR1)
       Y    =  XL1*(Y1**0.38 )
!
       WRITE(8, 1000) Y

 1000  FORMAT (F7.2)

       END    Program  Source1
```

Figure 1 A FORTRAN Main Program

****

204, 78, 243, 180

Figure 2 Input File C:\TMP\TSIN1

****

****

141.49

Figure 3 Output file C:\TMP\TSOUT

Utilizing The Shell Function

The Shell Function of VBA can only be used with an executable file.

Figure 4 is a VBA procedure (in Spreadsheet named FullProgram.XLS) that uses the Shell Function to execute the ggg exe file. Input data is retrieved from the spreadsheet (Cells method) and placed in the file C:\Tmp\Tsin1. The Shell function is then invoked to carry out the calculations. The output file C:\Tmp\Tsout is transferred back to the spreadsheet again using the Cells method.

It is noted that the procedure runs asynchronously and enough time must be given to the Fortran program to complete its calculations. A ‘ MsgBox’ command which generates a pause is a simple way to insure this rather than attempting to calculate the elapsed time of execution before proceeding.

Figure 5 lists the spreadsheet (FullProgram.XLS) that contains the VBA procedure CallFile2() of Figure 4.

****

```
Sub CallFile2()
Reset
' Clear D3 on Sheet 1
Call moda
'  Write Data From Spreadheet (Column 1) to TSin1
FileIn = "C:\Tmp\Tsin1"
Call WriteDataToTsin1(FileIn)
' Execute Source1 FORTRAN program on C;\tmp
Retval = Shell("C:\TMP\ggg.exe", 1)

MsgBox "  Pause After Shell Call   Task ID Number  =   " & Retval
'Define Output File
```

```
FileOut = "C:\Tmp\Tsout"
' Read Output Results From Tsout and Place Into Spreadsheet
Call ReadBackFromTsout(FileOut)

End Sub

Sub moda()
 Sheets("Sheet1").Select
 Range("D3:D3").Select
 Selection.ClearContents
End Sub

Sub WriteDataToTsin1(FileIn)
Dim A3 As Single
Dim A4 As Single
Dim A5 As Single
Dim A6 As Single

A3 = Cells(3, 1)
A4 = Cells(4, 1)
A5 = Cells(5, 1)
A6 = Cells(6, 1)

Open FileIn For Output As #1
Write #1, A3, A4, A5, A6
Close #1
End Sub

Sub ReadBackFromTsout(FileOut)

Open FileOut For Input As #1

Do Until EOF(1)
   Input #1, D1
Loop

Close #1

Cells(3, 4) = D1


End Sub
```

Figure 4 VBA Procedure CallFile2() Using the Shell Function to execute FORTRAN Program ggg.exe

****

This is the VBA spreadsheet FullProgram.XLS					
Input Data	Data For Ethyl Alcohol		Output in D3		
204	Heat Vap at NBP calories per gram		141.49	cal/ gram at Temp Deg	180
78	Normal Boiling Point Deg C				
243	Critical Temperature Deg C				
180	Temp (Deg C) at which Heat of Vap is Sought				

Figure 5 Spreadsheet FullPrigram.XLS

****

Creating a FORTRAN DLL

Figure 6 is listing of a FORTRAN subroutine MULLX. Two lines have been added starting with !MS$. These are metacommands which allow for passing variables and procedures between FORTRAN PowerStation and VBA.

```
**************************************************************
!     Fortran Subroutine as DLL
      Subroutine MULLX (X1,X2,X3,X4, D)

!MS$ATTRIBUTES DLLEXPORT::MULLX
!MS$ATTRIBUTES ALIAS: 'MULLX'::MULLX

      REAL XL1, XBP, XTC, XT, TR1, TR2, Y1, Y
!
!     X1   = Heat of Vaporization at NBP cal/gm
!     X2   = Normal Boiling Point Deg C
!     X3   = Critical Tempertaure in Degree C
!     X4   = Temperature at Which Heat of Vaporization is Sought
!
!     D    = Output, Heat of Vapoization at X4  cal/gm

      XL1 = X1
      XBP = X2
      XTC = X3
      XT  = X4

      TR1 = (273+XBP)/(273+XTC)
      TR2 = (273+XT)/(273+XTC)
      Y1  = (1-TR2)/(1-TR1)
      Y   = XL1*(Y1**.38)
      D = Y
      END Subroutine
```

Figure 6. FORTRAN Subroutine MULLX to be made into a DLL

****

To create the FORTRAN DLL for subroutine MULLX.FOR

1. Create a FORTRAN Subroutine (MULLX.FOR) with a text editor
   Add the ATTRIBUTE DLLEXPORT and ALIAS (Figure 6)
2. Place the file in directory C:\TMP1\
3. Invoke PowerStation. Click on MicroSoft Developer Studio
4.. Click File then New
5. Highlight Project Workspace. Press OK
6. Specify MULLX as the name (The name is arbitrary)
7. Highlight Dynamic Link Library and then 'Create'
8. Click File then New
9. Choose Text File then OK
10 Select Insert from the Menu. Then, 'Insert Files Into Project'
11 Double Click on c:\
12. Scroll down to TMP1 and click on it Then OK
13. Highlight MULLX.FOR
14. Click OK
15. Select Build from the menu. Then 'Build MULLX.DLL'
16. When compile and link are completed successfully the DLL is on
    C:\MSDEV\Projects\MULLX\DEBUG\MULLX.DLL

Using the FORTRAN DLL

The following is a VBA procedure that uses the MULLX.DLL

```
**********************************************************************
Private Declare Sub MULLX Lib "C:\msdev\projects\MULLX\debug\MULLX.DLL" _
(X1 As Single, X2 As Single, X3 As Single, X4 As Single, ByRef D As Single)
Sub Mullnew()
Dim X1, X2, X3, X4 As Single
Dim D As Single
X1 = Cells(7, 2)
X2 = Cells(8, 2)
X3 = Cells(9, 2)
X4 = Cells(10, 2)
Call MULLX(X1, X2, X3, X4, D)
Cells(13, 5) = D
End Sub
```

Figure 7 VBA Procedure Mullnew Calling MULLX. DLL

The VBA procedure (Figure 7) is called Mullnew. The arguments X1 X2, X3 and X4 are taken from the spreadsheet and passed to the sub MULLX. The sub

carries out the calculations indicated in Figure 6. Finally the result is placed in the D variable and passed to the spreadsheet using the Cells method.

Figure 8 is the spreadsheet used in the Mullnew procedure which utilizes the MULLX.DLL

****

This utilizes the MULLX.DLL for the Fortran Subroutine (MULLX.FOR)						
Heat of Vaporization from Hougen and Watson Illustration 3 page 234						
Data for Ethyl Alcohol						
Variable	Data	Definition				
x1	204	Heat of Vaporization at NBP cal/gm				
x2	78	Normal Boiling Point Deg C				
x3	243	Critical Temperature Deg C				
x4	180	Temperature at Which Heat of Vaporization is Sought				
OutPut						
Heat of Vaporization At X4 in cal/gm				141.4932		

Figure 8 Spreadsheet MullX.XLS

Conclusions

The use of the Shell function of VBA and/or use of a FORTRAN DLL provides a convenient way of incorporating FORTRAN into VBA programs.

Caution must be used to insure compatibility across different FORTRAN compilers and operating systems in calling FORTRAN from VBA.

References

1. "How to Call a FORTRAN DLL from VBA (FORTRAN DLL Overview)" www.emagenit.com/FORTRAN%20 DLL.htm

2. Rosen, Edward, "Executing FORTRAN Programs from Excel: Use of the Shell Function" *CACHE News*, No 47, Fall 1998

3. Rosen, Edward "Calling Fortran Subroutines from Excel 7.0". *CACHE News*, No 48 Spring 1999.

4. Hougen,O. and Watson.K, *Chemical Process Principles. Part One Material And Energ Balances*, John Wiley New York (1943).