```
' PicInfo bitmap file information class
' ©2006 Aivosto Oy (www.aivosto.com)
'
' This file is part of a sample project for Project Analyzer.
' Distribution of this file is only allowed along with Project Analyzer
' according to the Project Analyzer license terms.

Option Explicit

Implements IPicInfo

Private Enum EBMPType
    bmpUnknown
    bmpInfoHeader
    bmpCoreHeader
End Enum
Private BMPType As EBMPType

Private Type BITMAPFILEHEADER
        bfType As Integer       ' Specifies the file type, must be BM.
        bfSize As Long          ' Specifies the size, in bytes, of the bitmap file.
        bfReserved1 As Integer  ' Reserved; must be zero.
        bfReserved2 As Integer  ' Reserved; must be zero.
        bfOffBits As Long       ' Specifies the offset, in bytes, from the beginning of
         the BITMAPFILEHEADER structure to the bitmap bits.
End Type
Private Type BITMAPINFOHEADER
        biSize As Long
        biWidth As Long
        biHeight As Long
        biPlanes As Integer
        biBitCount As Integer
        biCompression As Long
        biSizeImage As Long
        biXPelsPerMeter As Long
        biYPelsPerMeter As Long
        biClrUsed As Long
        biClrImportant As Long
End Type
Private Type BITMAPCOREHEADER
        bcSize As Long
        bcWidth As Integer
        bcHeight As Integer
        bcPlanes As Integer
        bcBitCount As Integer
End Type
Private InfoHeader As BITMAPINFOHEADER
Private CoreHeader As BITMAPCOREHEADER

Private Const BI_RGB = 0&
Private Const BI_RLE4 = 2&
Private Const BI_RLE8 = 1&
Private Const BI_bitfields = 3&

' IsRLE indicates whether the bitmap is RLE compressed (True) or not (False)
' This variable is written but not read
' It might well be removed without affecting the functionality of the program
Private IsRLE As Boolean

' Filename used in the previous call to ReadFile
Private StoredFilename As String

```

```
Private Function ReadBitmapFile(ByVal Filename As String) As Boolean
' Read picture information from a bitmap file of type BMP/DIB/RLE
' [Filename] Name of picture file
' Return value:
' True - Picture information retrieved
' False - Not a valid bitmap file

Dim BMPFileHeader As BITMAPFILEHEADER ' Main BM file header
Dim FileNr As Integer  ' Number of open file
Dim HeaderSize As Long ' Size of header data in bytes

IsRLE = False        ' Clear the IsRLE flag. We will set it below to True if required.
BMPType = bmpUnknown ' Set file type to bmpUnknown until we can verify the real type

' Open the file for binary read access
FileNr = FreeFile
Open Filename For Binary Access Read Lock Write As #FileNr

If LOF(FileNr) > Len(BMPFileHeader) Then
    ' File length is "enough", read main BM file header header
    Get #FileNr, 1, BMPFileHeader
    If BMPFileHeader.bfType = &H4D42 Then ' BM
        If BMPFileHeader.bfReserved1 = 0 And BMPFileHeader.bfReserved2 = 0 Then
            ' Signature OK

            ' Retrieve size of following header
            Get #FileNr, , HeaderSize
            Seek #FileNr, Seek(FileNr) - 4 ' Rewind to start of header

            Select Case HeaderSize
                Case Len(CoreHeader)
                    Get #FileNr, , CoreHeader
                    BMPType = bmpCoreHeader
                    ReadBitmapFile = True ' File valid
                Case Len(InfoHeader)
                    Get #FileNr, , InfoHeader
                    BMPType = bmpInfoHeader

                    ' Determine bitmap compression
                    Select Case InfoHeader.biCompression
                        Case BI_RLE8, BI_RLE4
                            ' The bitmap is RLE compressed
                            IsRLE = True
                    End Select
                    ReadBitmapFile = True ' File valid
                Case Else
                    ReadBitmapFile = False ' Invalid/unsupported file type
            End Select
        End If
    End If
End If
Close FileNr

End Function


```

```
Private Property Get IPicInfo_Filename() As String
' Returns the filename used in the previous call to ReadFile

IPicInfo_Filename = StoredFilename

End Property

Private Property Get IPicInfo_MaxColors() As Variant

Select Case BMPType
    Case bmpCoreHeader
        IPicInfo_MaxColors = 2 ^ CoreHeader.bcBitCount
    Case bmpInfoHeader
        IPicInfo_MaxColors = 2 ^ InfoHeader.biBitCount
End Select


End Property

Private Function IPicInfo_ReadFile(ByVal Filename As String) As Boolean
' Read a picture file to retrieve picture information
' [Filename] File to read
' Return value:
' True - Picture information retrieved
' False - Error, information not retrieved

IPicInfo_ReadFile = ReadBitmapFile(Filename)

End Function


```

```
150 Private Property Get IPicInfo_Size() As TPicSize
151 ' Returns picture width and height in pixels
152
153 Select Case BMPType
154     Case bmpCoreHeader
155         IPicInfo_Size.Width = CoreHeader.bcWidth
156         IPicInfo_Size.Height = CoreHeader.bcHeight
157     Case bmpInfoHeader
158         IPicInfo_Size.Width = InfoHeader.biWidth
159         IPicInfo_Size.Height = InfoHeader.biHeight
160 End Select
161
162 End Property
163
164
```

```
' PicInfo main form
' ©2006 Aivosto Oy (www.aivosto.com)
'
' This file is part of a sample project for Project Analyzer.
' Distribution of this file is only allowed along with Project Analyzer
' according to the Project Analyzer license terms.

Option Explicit

Private Sub DisplayPictureInfo(ByVal Filename As String)
' Display information about a picture file
' [Filename] Name of picture file

Dim IPicInfo As IPicInfo ' Object that implements the IPicInfo interface
Dim PicSize As TPicSize  ' Picture size in pixels
Dim MaxColors As Variant ' Maximum number of colors in this picture

' Determine file type
Filename = LCase$(Filename)
If Filename Like "*.gif" Then
    ' GIF file, get a PicGIF object to handle it
    Set IPicInfo = New PicGIF
Else
    ' Assume it is a bitmap file, get a PicBMP object to handle it
    Set IPicInfo = New PicBMP
End If

' Read the picture file
If IPicInfo.ReadFile(Filename) Then
    ' Retrieve picture size in pixels
    PicSize = IPicInfo.Size
    ' Retrieve maximum number of colors in this picture
    MaxColors = IPicInfo.MaxColors

    ' Display retrieved information
    Me.Filename = "File: " & Filename
    Me.Size = "Size: " & PicSize.Width & " x " & PicSize.Height & " pixels"
    Me.MaxColors = "Max colors: " & MaxColors

    ' Display the picture as well
    Pict.Picture = LoadPicture(Filename)
End If

End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)

If KeyAscii = 15 Then
    ' Ctrl+O = Open
    OpenFile_Click
End If

End Sub

Private Sub Form_Load()
' As the form loads, _
  show the program title in the form caption

Me.Caption = ProgramTitle ' Global variable access

End Sub


Private Sub OpenFile_Click()
' User has clicked the "Open picture..." command button

Dim Filename As String

' Ask the user to select a picture file
Filename = ShowFileOpenDialog(Me.hWnd, "", "Picture files (*.gif;*.bmp)|*.gif;*.bmp")

If Len(Filename) Then
    ' We have received a filename to open
    DisplayPictureInfo Filename
End If


End Sub


Private Sub Pict_OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer,
 Shift As Integer, X As Single, Y As Single)
' If user has dropped one or more files on the picture area
' try to load the first file and show its picture info

If Data.GetFormat(vbCFFiles) Then
    Effect = vbDropEffectCopy
    DisplayPictureInfo Data.Files(1)
End If

End Sub

Private Sub Pict_OLEDragOver(Data As DataObject, Effect As Long, Button As Integer,
 Shift As Integer, X As Single, Y As Single, State As Integer)
' If user has dragged one or more files, indicate we will accept them

If Data.GetFormat(vbCFFiles) Then
    Effect = vbDropEffectCopy
End If

End Sub


Private Sub SaveFile_Click()
' Save file not implemented

Stop

End Sub

```

```
' PicInfo GIF file information class
' ©2006 Aivosto Oy (www.aivosto.com)
'
' This file is part of a sample project for Project Analyzer.
' Distribution of this file is only allowed along with Project Analyzer
' according to the Project Analyzer license terms.

Option Explicit

Implements IPicInfo

Private StoredFilename As String ' Filename used in the previous call to ReadFile

' FileOK is a flag indicating whether a valid GIF file has been read
' True - The variables below contain valid data
' False - The variables below contain undefined data and should not be used
Private FileOK As Boolean

' Brief introduction to GIF file structure
'
' Every GIF file starts with the following bytes:
' 6 bytes for Signature (either GIF87a or GIF89a in ASCII)
' 7 bytes for LogicalScreenDescriptor

Private Signature As String * 6  ' GIF87a, GIF89a

Private Type TLogicalScreenDescriptor
    Width As Integer             ' uint - Width, in pixels
    Height As Integer            ' uint - Height, in pixels
    Packed As Byte               ' Byte packed with color information
    BackgroundColorIndex As Byte ' Background Color index to Global Color Table
    PixelAspectRatio As Byte     ' Approximation of aspect ratio of pixel in the
     original image
End Type
Private LogicalScreenDescriptor As TLogicalScreenDescriptor ' Storage for
 LogicalScreenDescriptor
Private MaxColors As Long        ' Maximum number of colors in .gif

Private Function UInt(ByVal i As Integer) As Long
' Convert integer to unsigned integer
' [i] Signed integer to convert to an unsigned value
' Return value:
' Unsigned integer (fits in the Long datatype)

If i < 0 Then
    UInt = 65536 + i
Else
    UInt = i
End If

End Function



Private Function ReadGIF(ByVal GIFFilename As String) As Boolean
' Reads a GIF file and retrieves image information
' [GIFFilename] Name of file to read
' Return value:
' True - GIF file was read and information retrieved
' False - Error or invalid file

Dim ColorResolution As Integer   ' Number of color bits in the file (1..8)
Dim FileNr As Integer            ' Number of open data file

StoredFilename = GIFFilename ' Store Filename for later retrieval
FileOK = False               ' Clear the FileOK flag until we have successfully read a
 valid .gif file

' Open the .gif file for binary read access
FileNr = FreeFile
Open GIFFilename For Binary Access Read Lock Write As #FileNr

If LOF(FileNr) > Len(Signature) + Len(LogicalScreenDescriptor) Then
    ' The file size is "enough"

    ' Read 6 GIF signature bytes, which must be either GIF87a or GIF89a in ASCII
    Get #FileNr, 1, Signature
    If Signature = "GIF87a" Or Signature = "GIF89a" Then
        ' Signature bytes OK

        ' Read logical screen descriptor, which contains picture size and color info
        Get #FileNr, , LogicalScreenDescriptor

        ' Determine max number of colors
        ColorResolution = (LogicalScreenDescriptor.Packed And (64 + 32 + 16)) \ 16 + 1
        MaxColors = 2 ^ ColorResolution

        FileOK = True
        ReadGIF = True
    End If
End If
Close FileNr
FileNr = 0

End Function



Friend Property Get Version() As String
' Returns GIF format version (either "GIF87a" or "GIF89a")

' This procedure is not used by the sample project
' Project Analyzer reports it as a dead procedure

If FileOK Then
    Version = Signature
End If

End Property


Private Property Get IPicInfo_Filename() As String
' Returns the filename used in the previous call to ReadFile

IPicInfo_Filename = StoredFilename

End Property

Private Property Get IPicInfo_MaxColors() As Variant
' Returns the maximum number of colors possible to represent in picture

If FileOK Then
    ' We have the color information available
    IPicInfo_MaxColors = MaxColors
End If

End Property

Private Function IPicInfo_ReadFile(ByVal Filename As String) As Boolean
' Read a picture file to retrieve picture information
' [Filename] File to read
' Return value:
' True - Picture information retrieved
' False - Error, information not retrieved

IPicInfo_ReadFile = ReadGIF(Filename)

End Function


Private Property Get IPicInfo_Size() As TPicSize
' Returns picture width and height in pixels

If FileOK Then
    ' We have retrieved the picture size

    ' As the width and height are unsigned integers but VB6 supports no unsigned
     integer datatype,
    ' we need to convert the values with UInt() before returning to caller.
    IPicInfo_Size.Width = UInt(LogicalScreenDescriptor.Width)
```

```
147         IPicInfo_Size.Height = UInt(LogicalScreenDescriptor.Height)
148     End If
149
150     End Property
151
152
```

```
1  ' PicInfo IPicInfo interface
2  ' ©2006 Aivosto Oy (www.aivosto.com)
3  '
4  ' This file is part of a sample project for Project Analyzer.
5  ' Distribution of this file is only allowed along with Project Analyzer
6  ' according to the Project Analyzer license terms.
7
8  Option Explicit
9
10
11 ' Type for picture size
12 Public Type TPicSize
13     Width As Long   ' Picture width in pixels
14     Height As Long  ' Picture height in pixels
15 End Type
16 ------------------------------------------------------------------------
17 Public Property Get Filename() As String
18 Attribute Filename.VB_Description = "Returns the filename used in the previous call to
    ReadFile"
19 ' Returns the filename used in the previous call to ReadFile
20
21 End Property
22
23 ------------------------------------------------------------------------
24 Public Property Get MaxColors() As Variant
25 Attribute MaxColors.VB_Description = "Returns the maximum number of colors possible to
    represent in picture"
26 ' Returns the maximum number of colors possible to represent in picture.
27 ' Return value:
28 ' The return value is always numeric. The maximum value may exceed
29 ' the size of the Long data type, therefore we use the Variant data type.
30
31 End Property
32 ------------------------------------------------------------------------
33 Public Function ReadFile(ByVal Filename As String) As Boolean
34 Attribute ReadFile.VB_Description = "Read a picture file to retrieve picture
    information"
35 ' Read a picture file to retrieve picture information
36 ' [Filename] File to read
37 ' Return value:
38 ' True - Picture information retrieved
39 ' False - Error, information not retrieved
40
41 End Function
42 ------------------------------------------------------------------------
43 Public Property Get Size() As TPicSize
44 Attribute Size.VB_Description = "Returns picture width and height in pixels"
45 ' Returns picture width and height in pixels
46
47 End Property
48
49
```

```
' PicInfo main module
' ©2006 Aivosto Oy (www.aivosto.com)
'
' This file is part of a sample project for Project Analyzer.
' Distribution of this file is only allowed along with Project Analyzer
' according to the Project Analyzer license terms.

Option Explicit

' Global variable containing the title of this program
Public ProgramTitle As String


' API declarations for FileDialog
Private Declare Function GetOpenFileNameA Lib "comdlg32.dll" (pOpenfilename As OpenFilename) As Long

' This API Declare is dead
Private Declare Function GetSaveFileNameA Lib "comdlg32.dll" (pOpenfilename As OpenFilename) As Long

Private Type OpenFilename
        lStructSize As Long
        hwndOwner As Long
        hInstance As Long
        lpstrFilter As String
        lpstrCustomFilter As String
        nMaxCustFilter As Long
        nFilterIndex As Long
        lpstrFile As String
        nMaxFile As Long
        lpstrFileTitle As String
        nMaxFileTitle As Long
        lpstrInitialDir As String
        lpstrTitle As String
        Flags As Long
        nFileOffset As Integer
        nFileExtension As Integer
        lpstrDefExt As String
        lCustData As Long
        lpfnHook As Long
        lpTemplateName As String
End Type

Public Enum EFileDlgFlags
    OFN_ALLOWMULTISELECT = &H200            ' box allows multiple selections
    OFN_CREATEPROMPT = &H2000               ' If the user specifies a file that does not exist, this flag causes the dialog box to prompt the user for permission to create the file.
    OFN_FILEMUSTEXIST = &H1000              ' user can type only names of existing files in the File Name entry field
    OFN_HIDEREADONLY = &H4                  ' Hides the Read Only check box.

    OFN_NOCHANGEDIR = &H8
    ' Restores the current directory to its original value if the user changed the directory while searching for files.
    ' Windows NT 4.0/2000/XP: This flag is ineffective for GetOpenFileName.


    OFN_NODEREFERENCELINKS = &H100000
    ' Directs the dialog box to return the path and file name of the selected shortcut (.LNK) file.

    OFN_NONETWORKBUTTON = &H20000
    ' Hides and disables the Network button.

    OFN_NOREADONLYRETURN = &H8000&
    ' Specifies that the returned file does not have the Read Only check box selected and is not in a write-protected directory.


    OFN_NOTESTFILECREATE = &H10000
    ' Specifies that the file is not created before the dialog box is closed.
    ' This flag should be specified if the application saves the file on a create-nonmodify network share.
    ' When an application specifies this flag, the library does not check for write protection, a full disk,
    ' an open drive door, or network protection.

    OFN_OVERWRITEPROMPT = &H2
    ' Causes the Save As dialog box to generate a message box if the selected file already exists. The user must confirm whether to overwrite the file.

    OFN_PATHMUSTEXIST = &H800
    ' Specifies that the user can type only valid paths and file names.

    OFN_READONLY = &H1
    ' Causes the Read Only check box to be selected initially when the dialog box is created.
    ' This flag indicates the state of the Read Only check box when the dialog box is closed.

    OFN_SHAREAWARE = &H4000
    ' Specifies that if a call to the OpenFile function fails because of a network sharing violation, the error is ignored and the dialog box returns the selected file name.

    ' Open dialog defaults:
    ' File and path must exist
    ' No Read Only checkbox is displayed
    OFN_OPENDEFAULTS = OFN_FILEMUSTEXIST Or OFN_PATHMUSTEXIST Or OFN_HIDEREADONLY

    ' Save dialog defaults:
    ' User must confirm overwrite
    ' Path must exist
    ' * Read-only files can be selected, caller must use error handling if need to kill/overwrite/append
    OFN_SAVEDEFAULTS = OFN_OVERWRITEPROMPT Or OFN_PATHMUSTEXIST Or OFN_HIDEREADONLY Or OFN_NOREADONLYRETURN

End Enum

Const MAXFILE = 512 ' Size of file name buffer in chars. Should be at least 256.

' Error code retrieval
Private Declare Function CommDlgExtendedError Lib "comdlg32.dll" () As Long
Private Const CDERR_DIALOGFAILURE = &HFFFF
Private Const CDERR_FINDRESFAILURE = &H6
Private Const CDERR_NOHINSTANCE = &H4
Private Const CDERR_INITIALIZATION = &H2
Private Const CDERR_NOHOOK = &HB
Private Const CDERR_LOCKRESFAILURE = &H8
Private Const CDERR_NOTEMPLATE = &H3
Private Const CDERR_LOADRESFAILURE = &H7
Private Const CDERR_STRUCTSIZE = &H1
Private Const CDERR_LOADSTRFAILURE = &H5
Private Const FNERR_BUFFERTOOSMALL = &H3003
Private Const CDERR_MEMALLOCFAILURE = &H9
Private Const FNERR_INVALIDFILENAME = &H3002
Private Const CDERR_MEMLOCKFAILURE = &HA
Private Const FNERR_SUBCLASSFAILURE = &H3001


Public Function ShowFileOpenDialog(ByVal hwndOwner As Long, ByVal DefaultExtension As String, ByVal Filter As String, Optional ByRef FilterIndex As Long, Optional ByVal InitialDir As String, Optional ByVal DialogTitle As String, Optional ByRef Flags As EFileDlgFlags = OFN_OPENDEFAULTS) As String
' Show the Open file dialog box using comdlg32.dll
' [hwndOwner] In. Window handle of owner window. Can be zero.
' [DefaultExtension] In. Default file extension such as "txt".
' [Filter] In. Filters in format "File type|*.ext;*.txt|All files|*.*".
' [FilterIndex] In/out. Selected file type in filter. First filter is number 1.
' [InitialDir] In. Initial directory. If not given or empty, uses system default initial directory. See FileDialog for explanation.
' [DialogTitle] In. Dialog caption. If not given or empty, uses system default caption.
' [Flags] In/out. Flags for the dialog. Should be given as "OFN_OPENDEFAULTS or ...". On return contains any altered flags.
' Return value:
' Selected file name if user pressed OK
' Empty string if user pressed Cancel or there was an error
```

```
' Caller can verify error status by calling CommDlgExtendedError

ShowFileOpenDialog = FileDialog(hwndOwner, DefaultExtension, Filter, FilterIndex,
 InitialDir, DialogTitle, Flags)

End Function

Private Function FileDialog(ByVal hwndOwner As Long, ByVal DefaultExtension As String,
 ByVal Filter As String, ByRef FilterIndex As Long, ByVal InitialDir As String, ByVal
 DialogTitle As String, ByRef Flags As EFileDlgFlags) As String
' Implementation of ShowFileOpenDialog (see it for description)

Dim OFN As OpenFilename  ' Structure with dialog information
Dim Result As Long       ' API return value
Dim indNull As Long      ' Index of Null character (ASCII 0)

Dim ErrMsg As String, ErrCode As Long ' Error information

' Populate OFN structure with dialog initialization information
With OFN
    .lStructSize = Len(OFN)         ' Length of structure
    .hwndOwner = hwndOwner          ' Owner window handle
    .lpstrFilter = Replace(Filter, "|", vbNullChar) & vbNullChar & vbNullChar   '
     Filters
    .nFilterIndex = FilterIndex     ' Selected filter
    .lpstrFile = String(MAXFILE, 0) ' Filename
    .nMaxFile = MAXFILE             ' Max length of filename

    ' Initial directory
    If LenB(InitialDir) = 0 Then
        .lpstrInitialDir = CurDir     ' No initial directory given, use current
         directory
    Else
        .lpstrInitialDir = InitialDir ' Use given initial directory
    End If
    .lpstrTitle = DialogTitle         ' Title of dialog
    .Flags = Flags                    ' Various flags passed by caller
    .lpstrDefExt = DefaultExtension   ' Default file extension
End With

' Show the Open dialog
Result = GetOpenFileNameA(OFN)

If Result <> 0 Then
    ' Success
    With OFN
        ' Retrieve selected filename and set it as the function return value
        indNull = InStr(.lpstrFile, vbNullChar)
        If indNull > 0 Then
            FileDialog = Left$(.lpstrFile, indNull - 1) ' Get rid of terminating null
             character
        Else
            FileDialog = .lpstrFile ' No terminating null
        End If

        Flags = .Flags              ' Return flags to caller - they may have changed
        FilterIndex = .nFilterIndex ' Return selected filter to caller
    End With
Else
    ' Error or cancel
    ' Call CommDlgExtendedError to get error code
    ErrCode = CommDlgExtendedError()

    ' Select error message
    Select Case ErrCode
        Case 0
            ' Cancel pressed, no error
        Case CDERR_DIALOGFAILURE
            ErrMsg = "Dialog box could not be created."
        Case CDERR_FINDRESFAILURE
            ErrMsg = "Failed to find a resource."
        Case CDERR_NOHINSTANCE
            ErrMsg = "Instance handle missing."
        Case CDERR_INITIALIZATION
            ErrMsg = "Failure during initialization. Possibly out of memory."
        Case CDERR_NOHOOK
            ErrMsg = "Hook procedure missing."
        Case CDERR_LOCKRESFAILURE
            ErrMsg = "Failed to lock a resource."
        Case CDERR_NOTEMPLATE
            ErrMsg = "Template missing."
        Case CDERR_LOADRESFAILURE
            ErrMsg = "Failed to load a resource."
        Case CDERR_STRUCTSIZE
            ErrMsg = "Internal error - invalid struct size."
        Case CDERR_LOADSTRFAILURE
            ErrMsg = "Failed to load a string."
        Case FNERR_BUFFERTOOSMALL
            ErrMsg = "File name buffer too small"
        Case CDERR_MEMALLOCFAILURE
            ErrMsg = "Unable to allocate memory for internal dialog structures."
        Case FNERR_INVALIDFILENAME
            ErrMsg = "Invalid file name."
        Case CDERR_MEMLOCKFAILURE
            ErrMsg = "Unable to lock memory."
        Case FNERR_SUBCLASSFAILURE
            ErrMsg = "Subclass failure, out of memory."
        Case Else
            ErrMsg = "Unknown error."
    End Select
    If LenB(ErrMsg) Then
        ' An error occurred. Display it.
        If LenB(DialogTitle) Then
            ErrMsg = "Error #" & ErrCode & " with dialog " & DialogTitle & vbCrLf &
             ErrMsg
        Else
            ErrMsg = "Error #" & ErrCode & " with file open dialog" & vbCrLf & ErrMsg
        End If
        MsgBox ErrMsg, vbExclamation
    End If
End If

End Function

Sub Main()
' PicInfo main procedure
' The program starts here

' Store program title in global variable
ProgramTitle = App.Title

' Show main form
PicForm.Show

End Sub
```