cConvertYesNo - 1

```
Option Explicit
Option Compare Text
'Option Private Module

Public iDlg As Integer

Private dlg As DialogSheet
Private btnHelp As Button
Private btnCancel As Button
Private btnBack As Button
Private btnNext As Button
Private btnFinish As Button
Public obYes As OptionButton
Private obNo As OptionButton

'---------------------------------------------------------------------------
Public Sub Boot()
  'Set aliases for fast access
  Set dlg = rgDe(iDlg).dlg
  With dlg
    Set btnHelp = .Buttons("btnHelp")
    Set btnCancel = .Buttons("btnCancel")
    Set btnBack = .Buttons("btnBack")
    Set btnNext = .Buttons("btnNext")
    Set btnFinish = .Buttons("btnFinish")
    Set obYes = .OptionButtons("obYes")
    Set obNo = .OptionButtons("obNo")
  End With
End Sub
'---------------------------------------------------------------------------
Public Sub Init()
  obYes.Value = xlOff
  obNo.Value = xlOn
End Sub
'---------------------------------------------------------------------------
Public Sub Frame()
  dlg.Buttons.Enabled = True
  dlg.Focus = "btnNext"
End Sub
'---------------------------------------------------------------------------
Public Function Dismiss() As Boolean
  Dismiss = False
  Select Case btnCode
    Case bcNext, bcFinish
      btnCode = bcSetNext
      If obYes.Value = xlOn Then
        iDlgCur = cSelectForms.iDlg
      Else
        iDlgCur = cLastDialog.iDlg
      End If
      If fInFinishMode Then Exit Function
      dlg.Hide False
      Dismiss = True
    Case bcBack
      dlg.Hide True
    Case Else
      Assert False
  End Select
End Function
```

cLastDialog - 1

```
Option Explicit
Option Compare Text
'Option Private Module

Public iDlg As Integer

Private dlg As DialogSheet
Private btnHelp As Button
Private btnCancel As Button
Private btnBack As Button
Private btnNext As Button
Private btnFinish As Button
Private btnRoutingSlip As Button
Private lblTemplate As Label
Private lblDatabase As Label
Private fHasMailSystem As Boolean

'---------------------------------------------------------------------------
Public Sub Boot()
  'Set aliases for fast access
  Set dlg = rgDe(iDlg).dlg
  With dlg
    Set btnHelp = .Buttons("btnHelp")
    Set btnCancel = .Buttons("btnCancel")
    Set btnBack = .Buttons("btnBack")
    Set btnNext = .Buttons("btnNext")
    Set btnFinish = .Buttons("btnFinish")
    Set btnRoutingSlip = .Buttons("btnRoutingSlip")
    Set lblTemplate = .Labels("lblTemplate")
    Set lblDatabase = .Labels("lblDatabase")
  End With
'  fHasMailSystem = (Application.MailSystem <> xlNoMailSystem)
  fHasMailSystem = (Application.MailSystem = xlMAPI)
End Sub
'---------------------------------------------------------------------------
Public Sub Init()
End Sub
'---------------------------------------------------------------------------
Public Sub Frame()
  dlg.Buttons.Enabled = True
  btnNext.Enabled = False
  btnRoutingSlip.Enabled = fHasMailSystem
  dlg.Focus = "btnFinish"
  lblTemplate.Text = dsTemplate & stTemplate
  lblDatabase.Text = dsDatabase & DB.stName
End Sub
'---------------------------------------------------------------------------
Public Function Dismiss() As Boolean
  Dismiss = False
  Select Case btnCode
    Case bcNext, bcFinish
      Application.ScreenUpdating = False
      'create the template if it is not already created. The template could be already
      'created if the user added a routing slip to it
      If Not fTemplateIsForm Then If wbTemplate Is Nothing Then _
        If Not FCreateTemplate(stTemplate, wbForm) Then GoTo LFailure
      If fNewDB Then
        If Not FCreateDB(DB) Then GoTo LFailure
      Else
        If Not FCreateTemplateKeys(DB) Then GoTo LFailure
      End If

      CreateShtTemplate wbTemplate
      WriteShtTemplate wbTemplate.Worksheets(dsTemplateSheet), DB
      HideSheet wbTemplate.Worksheets(dsTemplateSheet)
      If Not fTemplateIsForm Then
        wbTemplate.Close savechanges:=True, routeworkbook:=False
      End If

      vWbSav.Activate 'activate the sheet on which we started
      Application.ScreenUpdating = True
      If lbFiles.ListCount <> 0 Then
        If (AddFilesToDB(DB) = False) Then GoTo LFailure
      End If
```

```
cLastDialog - 2

      If fInFinishMode Then Exit Function
      dlg.Hide False
      Dismiss = True
      Exit Function
LFailure:
      Application.ScreenUpdating = True
      SetErrorFocus "btnBack"
      Exit Function
    Case bcBack
      btnCode = bcSetNext
      If cConvertYesNo.obYes.Value = xlOn Then
        iDlgCur = cSelectForms.iDlg
      Else
        iDlgCur = cConvertYesNo.iDlg
      End If
      dlg.Hide True
    Case Else
      Assert False
  End Select
End Function
'--------------------------------------------------------------------------
Sub btnRoutingSlipClick()
  Dim wbsav As Workbook
  Dim fScreenUpdatingSav As Boolean

  Set wbsav = ActiveWorkbook
  fScreenUpdatingSav = Application.ScreenUpdating
  Application.ScreenUpdating = False

  If Not fTemplateIsForm Then If wbTemplate Is Nothing Then _
    If Not FCreateTemplate(stTemplate, wbForm) Then Exit Sub
  wbTemplate.Activate

  On Error GoTo LError
  Application.Dialogs(xlDialogRoutingSlip).Show 'can add arguments if required
  dlg.Focus = "btnFinish"
LError:

  wbsav.Activate
  Application.ScreenUpdating = fScreenUpdatingSav
End Sub
'--------------------------------------------------------------------------
' REVISE REVISE REVISE DOES NOT CLEANUP On ERRORS!!
'--------------------------------------------------------------------------
Private Function AddFilesToDB(DB As DBStruct) As Boolean
  Dim stFile As String
  Dim i As Integer
  Dim DBObj  As Object 'workbook or database object
  Dim daoDBEngine As Object
  Dim fWbWasAlreadyOpen As Boolean
  Dim fDBWasAlreadyOpen As Boolean
  Dim wb As Workbook

  AddFilesToDB = True
  'open the DB for writing
  Select Case DB.iType
    Case iExcelDBType
      If Not FOpenWorkbook(DB.stName, DBObj, fDBWasAlreadyOpen, False) Then GoTo LFailure
    Case Else
      On Error GoTo LDAOError
        Set daoDBEngine = CreateObject("DAO.DBEngine.36")
        Set DBObj = daoDBEngine.Workspaces(0).OpenDatabase( _
            DB.stName, False, False, rgDBType(DB.iType).stType)
        Set daoDBEngine = Nothing
      On Error GoTo 0
  End Select

  For i = 1 To lbFiles.ListCount
    stFile = lbFiles.List(i)
    If FOpenWorkbook(stFile, wb, fWbWasAlreadyOpen) Then
      Assert Not (wb Is Nothing)
      If FExtractRecordFromWb(wb, DB, Nothing) Then
        If (FAddRecordToDB(DB, DBObj, Nothing, False) = False) Then
          AddFilesToDB = False
        End If
      End If
```

```
cLastDialog - 3

      If Not fWbWasAlreadyOpen Then wb.Close savechanges:=False, routeworkbook:=False
    End If
    DoEvents
  Next i
  'close the database, saving the changes
  On Error GoTo LError
    Select Case DB.iType
      Case iExcelDBType
        If Not fDBWasAlreadyOpen Then DBObj.Close savechanges:=True
      Case Else
        DBObj.Close
    End Select
  On Error GoTo 0
  Exit Function
LDAOError:
  DisplayError ecUnexpectedError
  GoTo LFailure
LError:
  DisplayError ecUnexpectedError
LFailure:
  AddFilesToDB = False
End Function
```

cSelectDatabase - 1

```
Option Explicit
Option Compare Text
'Option Private Module

Public iDlg As Integer

Private dlg As DialogSheet
Private btnHelp As Button
Private btnCancel As Button
Private btnBack As Button
Private btnNext As Button
Private btnFinish As Button
Private btnBrowse As Button
Public ddDBType As DropDown 'these two are set in cSelectWorkbook
Public ebDBName As EditBox  '-----------------------------------

Public DB As DBStruct 'the database
Public fNewDB As Boolean 'is it a new database
Public fRefsOutDated As Boolean
'---------------------------------------------------------------------------
Public Sub Boot()
  Dim i As Integer
  'Set aliases for fast access
  Set dlg = rgDe(iDlg).dlg
  With dlg
    Set btnHelp = .Buttons("btnHelp")
    Set btnCancel = .Buttons("btnCancel")
    Set btnBack = .Buttons("btnBack")
    Set btnNext = .Buttons("btnNext")
    Set btnFinish = .Buttons("btnFinish")
    Set btnBrowse = .Buttons("btnBrowse")
    Set ddDBType = .DropDowns("ddDBType")
    Set ebDBName = .EditBoxes("ebDBName")
  End With
  With ddDBType
    .RemoveAllItems
    For i = 1 To cDBTypeMac
      .AddItem rgDBType(i).stDescription
    Next i
  End With
End Sub
'---------------------------------------------------------------------------
Public Sub Init()
  ebDBName.Text = ""
  ddDBType.Value = 1
  DB.iType = 0
End Sub
'---------------------------------------------------------------------------
Public Sub Frame()
  dlg.Buttons.Enabled = True
  If (ebDBName.Text = "") Then
    If (PutDBAfterName = True) Then
      ebDBName.Text = StFullPathName(stForm & " " & dsDatabaseName)
    Else
      ebDBName.Text = StFullPathName(dsDatabaseName & " " & stForm)
    End If
  End If
  SetDlgCtlEnable
  If (Not fStartedOnTemplate) Then
    SetFocus "ddDBType"
  Else
    SetFocus "ebDBName"
  End If
  dlg.Labels("ddDBTypeText2").Visible = fStartedOnTemplate
  dlg.Labels("ddDBTypeText1").Visible = Not fStartedOnTemplate
  ddDBType.Visible = Not fStartedOnTemplate
  dlg.Labels("ddDBTypeLabel").Visible = Not fStartedOnTemplate

'  If (Not fStartedOnTemplate) Then
'    dlg.EditBoxes("ebDBName").Text = stForm & " Database"
'    SetDlgCtlEnable
'  End If
End Sub
'---------------------------------------------------------------------------
Public Function Dismiss() As Boolean
```

```
cSelectDatabase - 2

  Dim fc As Integer
  Dim stDBNameNew As String

  Dismiss = False
  Select Case btnCode
    Case bcNext, bcFinish
      If ebDBName.Text = "" Then 'this could happen if Finish was pressed in the Ist dlg
        Assert fInFinishMode
        DisplayError ecDatabaseNameRequired
        SetErrorFocus "ebDBName"
        Exit Function
      End If

      stDBNameNew = StFullPathName(StAddDBTypeExtension(ddDBType.Value, ebDBName.Text))
      'check to see if the DB has changed
      If Not (ddDBType.Value = DB.iType And stDBNameNew = DB.stName) Then 'changed
        Dim fPreserve As Boolean

        'if the previous database was an New database, preserve previous setting of tables
        fPreserve = DB.iType <> 0 And fNewDB
        DB.iType = ddDBType.Value
        DB.stName = stDBNameNew
        If Not FInitDBStructure(DB, False, fPreserve) Then
          DB.iType = 0 'setting it to 0 will cause us to register a change when the
                       'user clicks "Next" again, since ddDBType.Value can never be zero
          SetErrorFocus "ebDBName"
          Exit Function
        End If
        'if the database now opened is also a new database, keep the refs
        If Not (fPreserve And fNewDB) Then
          fRefsOutDated = True
          'hack: ResetICurs should be called by cSelectFields.Frame, and not here, but
          'we have to call it, since the MakeFieldVisible function uses iCur values and
          'may get called if the user hit finish and an error occurred
          cSelectFields.ResetiCurs DB
          fTableViewOutdated = True
        End If
      End If

      If fRefsOutDated Then
        If fStartedOnTemplate And Not fNewDB Then 'fill out refs from the template
          LoadRefsFromTemplate DB, wbForm.Worksheets(dsTemplateSheet) 'refs will get validated in c
SelectFields.Dismiss
        Else
          BlankRefs DB
        End If
        fRefsOutDated = False
        fFieldViewOutdated = True
      End If

      If fInFinishMode Then Exit Function
      dlg.Hide False
      Dismiss = True
    Case bcBack
      dlg.Hide True
    Case Else
      Assert False
  End Select
End Function


Sub SetDlgCtlEnable()
  btnNext.Enabled = (ebDBName.Text <> "")
  btnFinish.Enabled = (ebDBName.Text <> "")
  If btnNext.Enabled Then btnNext.Parent.DefaultButton = btnNext.Name
End Sub

Public Sub ebDBNameChange()
  SetDlgCtlEnable
End Sub

'---------------------------------------------------------------------------
Sub btnBrowseClick()
  Dim File As Variant
  Dim i As Integer
```

```
cSelectDatabase - 3

  'bug 276
  i = cSelectDatabase.ddDBType.ListIndex

  'bug 284
  File = Application.GetOpenFilename(fileFilter:=rgDBType(i).stFilter, Title:=dsSelectDatabaseFile)
  'filterIndex:=2,
  If File <> False Then dlg.EditBoxes("ebDBName").Text = File
  'bug 637
  SetDlgCtlEnable
End Sub
```

cSelectFields - 1

```
Option Explicit
Option Compare Text
'Option Private Module

Private Const cColumn As Integer = 4 'the number of columns in the display
                                     'must be more than two and less than 10
                                     'and of course you need to change the dlgsheet!
Public iDlg As Integer

Private dlg As DialogSheet
Private btnHelp As Button
Private btnCancel As Button
Private btnBack As Button
Private btnNext As Button
Private btnFinish As Button
Private ddTable As DropDown
Private ebTable As EditBox
Private rglbln(1 To cColumn) As Label
Private rglblName(1 To cColumn) As Label
Private rgebName(1 To cColumn) As EditBox
Private picName As Picture
Private rgebRef(1 To cColumn) As EditBox
Private sbColumn As ScrollBar

'-------------------------------------------------------------------------
Private iTableCur
Private rgIFieldCur(1 To cTableMax) As Integer  'the current field in each of the tables
'Remark: xxCur values must be initialized to 1 even when xxMac in rgxx is zero.
'-------------------------------------------------------------------------

Public fTableViewOutdated As Boolean 'whether the tables view presented in this dlg is outdated
Public fFieldViewOutdated As Boolean
'-------------------------------------------------------------------------
Public Sub Boot()
  Dim i As Integer
  Dim stT As String * 1

  'Set aliases for fast access
  Set dlg = rgDe(iDlg).dlg
  With dlg
    Set btnHelp = .Buttons("btnHelp")
    Set btnCancel = .Buttons("btnCancel")
    Set btnBack = .Buttons("btnBack")
    Set btnNext = .Buttons("btnNext")
    Set btnFinish = .Buttons("btnFinish")

    Set ddTable = .DropDowns("ddTable")
    Set ebTable = .EditBoxes("ebTable")

    Set picName = .Pictures("picName")
    Set sbColumn = .ScrollBars("sbColumn")
    sbColumn.LargeChange = cColumn
    sbColumn.Min = 1

    For i = 1 To cColumn
      stT = StFromI(i)
      Set rglbln(i) = .Labels("lbln" & stT)
      Set rglblName(i) = .Labels("lblName" & stT)
      Set rgebName(i) = .EditBoxes("ebName" & stT)
      Set rgebRef(i) = .EditBoxes("ebRef" & stT)
    Next i
  End With
End Sub
'-------------------------------------------------------------------------
Public Sub Init()
End Sub
'-------------------------------------------------------------------------
Public Sub Frame()
  Dim iTable  As Integer
  Dim iColumn As Integer

  If fTableViewOutdated Then 'the tables have changed
    'make selective components visible based on whether it is a new table
    ebTable.Visible = fNewDB
    ddTable.Visible = Not fNewDB
```

```
    picName.Visible = Not fNewDB
    For iColumn = 1 To cColumn
      rglblName(iColumn).Visible = Not fNewDB
      rgebName(iColumn).Visible = fNewDB
    Next iColumn
    'Add the list of tables to the dropdown if it is an existing database
    If Not fNewDB Then
      With ddTable
        .RemoveAllItems
        For iTable = 1 To DB.cTable
          .AddItem DB.rgTable(iTable).stName
        Next iTable
'        .DropDownLines = Min(5, DB.cTable)
      End With
    End If
    If fNewDB Then
      ebTable.Text = DB.rgTable(iTableCur).stName
    Else
      ddTable.Value = iTableCur
    End If
    ConfigureScrollbar
    fTableViewOutdated = False
    fFieldViewOutdated = True
  End If
  If fFieldViewOutdated Then
    LoadDisplayColumns 1, cColumn
    fFieldViewOutdated = False
  End If
  dlg.Buttons.Enabled = True
  dlg.Labels("newLabel").Visible = Not (fStartedOnTemplate)
  dlg.Labels("existingLabel").Visible = fStartedOnTemplate

  dlg.Labels("lSheet").Visible = (DB.iType = iExcelDBType)
  dlg.Labels("lTable").Visible = (DB.iType <> iExcelDBType)


  SetFocus rgebRef(1).Name
End Sub
'-------------------------------------------------------------------------
Public Function Dismiss() As Boolean
  Dismiss = False
  Select Case btnCode
    Case bcNext, bcFinish
      If fNewDB Then
        If Not FFieldsFilled() Then Exit Function
        If Not FUniqueNames() Then Exit Function
        If Not FRefsValid() Then Exit Function
      Else
        If Not FRefsValid() Then Exit Function
      End If

      If fInFinishMode Then Exit Function
      dlg.Hide False
      Dismiss = True
    Case bcBack
      dlg.Hide True
    Case Else
      Assert False
  End Select
End Function

'-------------------------------------------------------------------------
Public Sub sbColumnChange()
  If sbColumn.Value <> rgIFieldCur(iTableCur) Then
    rgIFieldCur(iTableCur) = sbColumn.Value
    LoadDisplayColumns 1, cColumn
  End If
End Sub
'-------------------------------------------------------------------------
Public Sub ebRefChange()
  Dim iColumn As Integer
  Dim iField As Integer
  Dim stRef As String
  Dim rwOff As Integer, colOff As Integer
  Dim cellRef As Range
  Dim stCellName As String
```

```
cSelectFields - 3

  Dim stIColumn As String * 1
  Dim stFocus As String
  Dim i As Integer
  Dim fDispErr As Boolean
  Dim nErr As Integer

  stIColumn = Right$(Application.Caller, 1) 'bug 117308
  iColumn = IFromSt(stIColumn)
  iField = IFieldFromIColumn(iColumn)
  fDispErr = Not (dlg.Focus = "btnCancel" Or dlg.Focus = "btnBack") 'don't bother the poor fellow i
f he is trying to quit

  If rgebRef(iColumn).Text = "" Then GoTo LNoName

  'check if valid cell ref
  stRef = StValidRef(rgebRef(iColumn).Text, fDispErr)
  If stRef = "" Then 'invalid ref
    If fDispErr Then dlg.Focus = "ebRef" & stIColumn
    DB.rgTable(iTableCur).rgfield(iField).stRef = rgebRef(iColumn).Text 'save the junk that a user
types
    Exit Sub
  End If
  'bug 286 > if you do that we get the a result using A1 references
  'rgebRef(iColumn).Text = stRef 'standardize the ref
  DB.rgTable(iTableCur).rgfield(iField).stRef = stRef
  'bug 286
  Set cellRef = Evaluate(stRef)

  'Find a name for the Ref
  If Not fNewDB Then GoTo LNoName

  If rgebName(iColumn).Text <> "" Then GoTo LNoName 'name already filled in
  'check for a defined name for this cell here
  On Error Resume Next
  stCellName = cellRef.Name.Name
  nErr = Err
  On Error GoTo 0
  If nErr <> 0 Then 'name is not defined, search in vicinity of the cell
    rwOff = 0
    colOff = iFindLabelLeft(cellRef)
    If colOff = 0 Then
      rwOff = iFindLabelTop(cellRef)
      If rwOff = 0 Then GoTo LNoName
    End If
    'at this point only one of rwoff coloff is zero and a label has been found
    stCellName = cellRef.Offset(rwOff, colOff).Value
  End If
  rgebName(iColumn).Text = stCellName
  DB.rgTable(iTableCur).rgfield(iField).stName = stCellName

  stFocus = dlg.Focus 'save since AdjustColumns may change dlg.Focus
  AdjustColumns iColumn
  'adjust the focus. Modify focus only if the user clicked in the name
  'field for this ref field. and a name was found, and we are on the last
  'column.
  If stFocus = "ebName" & stIColumn _
    And iField = DB.rgTable(iTableCur).cField _
    And iField <> cFieldMax Then
    iColumn = IColumnFromIField(iField + 1) 'the next col
    If iColumn >= 1 And iColumn <= cColumn Then
      dlg.Focus = "ebRef" & StFromI(iColumn)
    End If
  End If
  Exit Sub

LNoName:  'if no name found save ref, adjust columns and exit
  DB.rgTable(iTableCur).rgfield(iField).stRef = rgebRef(iColumn).Text
  AdjustColumns iColumn
End Sub
'-----------------------------------------------------------------------
Public Sub EbNameChange()
  Dim iColumn As Integer
  Dim iField As Integer

  Assert fNewDB
```

```
cSelectFields - 4

  iColumn = IFromSt(Right$(Application.Caller, 1)) 'bug 117308
  iField = IFieldFromIColumn(iColumn)
  DB.rgTable(iTableCur).rgfield(iField).stName = rgebName(iColumn).Text
  AdjustColumns iColumn
End Sub
'-----------------------------------------------------------------------
Public Sub ddTableChange()
  iTableCur = ddTable.Value
  LoadDisplayColumns 1, cColumn
  ConfigureScrollbar
End Sub
'-----------------------------------------------------------------------
Public Sub EbTableChange()
  DB.rgTable(iTableCur).stName = ebTable.Text
End Sub

'-----------------------------------------------------------------------
Sub ConfigureScrollbar()
  With DB.rgTable(iTableCur)
    'cField+1 is the number of columns (one for the blank column (fNewDB)
    'or disabled col (not fNewDB)). if it is less than or equal to cColumn,
    'then disable scrollbar
    If .cField + 1 <= cColumn Then
      sbColumn.Enabled = False
      sbColumn.Max = 1
    Else
      sbColumn.Enabled = True
      sbColumn.Max = (.cField + 1) - cColumn + 1
    End If
    sbColumn.Value = rgIFieldCur(iTableCur)
  End With
End Sub
'-----------------------------------------------------------------------
'function to load the columns according to the current table
'iFirst specifies the first column to load, iLast, the last.
'iLast is set to cColumn if it is more than that.
'-----------------------------------------------------------------------
Sub LoadDisplayColumns(iFirst As Integer, iLast As Integer)
  Dim iColumn As Integer
  Dim iField As Integer
  Dim iUpperT As Integer

  If iLast > cColumn Then iLast = cColumn
  With DB.rgTable(iTableCur)
    If fNewDB Then
      'if cField = cFieldmax we wont show the blank column
      iUpperT = Min(.cField + 1, cFieldMax)
      For iColumn = iFirst To iLast
        iField = IFieldFromIColumn(iColumn)
        If iField <= iUpperT Then 'activate one more than actual noof fields
          rglbln(iColumn).Text = CStr(iField)
          rgebName(iColumn).Enabled = True
          rgebName(iColumn).Text = .rgfield(iField).stName
          rgebRef(iColumn).Enabled = True
          'bug 466
          If .rgfield(iField).stRef = "" Then
                rgebRef(iColumn).Text = .rgfield(iField).stRef
          Else
                rgebRef(iColumn).Text = GetCorrectRef(iTableCur, iField)
          End If
        Else
          rglbln(iColumn).Text = "-"
          rgebName(iColumn).Enabled = False
          rgebName(iColumn).Text = ""
          rgebRef(iColumn).Enabled = False
          rgebRef(iColumn).Text = ""
        End If
      Next iColumn
    Else
      For iColumn = iFirst To iLast
        iField = IFieldFromIColumn(iColumn)
        If iField <= .cField Then
          rglbln(iColumn).Text = CStr(iField)
          rglblName(iColumn).Text = .rgfield(iField).stName
          rgebRef(iColumn).Enabled = True
          'bug 467
```

```
cSelectFields - 5

          If .rgfield(iField).stRef = "" Then
                rgebRef(iColumn).Text = .rgfield(iField).stRef
          Else
                rgebRef(iColumn).Text = GetCorrectRef(iTableCur, iField)
          End If
        Else
          rglbln(iColumn).Text = "-"
          rglblName(iColumn).Text = ""
          rgebRef(iColumn).Enabled = False
          rgebRef(iColumn).Text = ""
        End If
      Next iColumn
    End If
  End With
End Sub
'-----------------------------------------------------
'bug 467
'-----------------------------------------------------
Function GetCorrectRef(ByVal iTableCur As Integer, ByVal iField As Integer) As String
Dim iPos As Integer
Dim iLen As Integer

    If Not IsError(Application.ConvertFormula(Formula:=DB.rgTable(iTableCur).rgfield(iField).stRef,
 _
                fromreferencestyle:=xlR1C1, toReferenceStyle:=Application.ReferenceStyle)) Then
        GetCorrectRef = Application.ConvertFormula(Formula:=DB.rgTable(iTableCur).rgfield(iField).s
tRef, _
                        fromreferencestyle:=xlR1C1, toReferenceStyle:=Application.ReferenceStyle)

    Else
        GetCorrectRef = Application.ConvertFormula(Formula:=DB.rgTable(iTableCur).rgfield(iField).s
tRef, _
        fromreferencestyle:=xlA1, toReferenceStyle:=Application.ReferenceStyle)

    End If

    iPos = InStr(1, GetCorrectRef, "]", 0)
    iLen = Len(GetCorrectRef)

    If Left(GetCorrectRef, 1) = "'" Then
        GetCorrectRef = "'" & Right(GetCorrectRef, iLen - iPos)
    Else
        GetCorrectRef = Right(GetCorrectRef, iLen - iPos)
    End If

End Function


'-------------------------------------------------------------------------
'makes the given field of the given table visible
'-------------------------------------------------------------------------
Sub MakeFieldVisible(iTable As Integer, iField As Integer)
  If iTableCur <> iTable Then
    iTableCur = iTable
    If fNewDB Then
      ebTable.Text = DB.rgTable(iTableCur).stName
    Else
      ddTable.Value = iTableCur
    End If
    GoTo LScrollTable
  End If
  If iField < rgIFieldCur(iTableCur) Or iField > rgIFieldCur(iTableCur) + cColumn - 1 Then 'it is n
ot currently visible
LScrollTable:
    If iField < rgIFieldCur(iTableCur) Then rgIFieldCur(iTableCur) = iField
    If iField > rgIFieldCur(iTableCur) + cColumn - 1 Then rgIFieldCur(iTableCur) = iField - cColumn
 + 1
    LoadDisplayColumns 1, cColumn
    ConfigureScrollbar
  End If
End Sub
'-------------------------------------------------------------------------
'Adjust the number of columns and the display. iColumn is the index of the
'column that was changed.
'-------------------------------------------------------------------------
Sub AdjustColumns(iColumn As Integer)
```

```
cSelectFields - 6

  Dim iField As Integer
  Dim i As Integer
  Dim stEbFocus As String
  Dim iFieldFocus As Integer
  Dim iColumnFocus As Integer

  If Not fNewDB Then Exit Sub 'for now we adjust only new DBs

  stEbFocus = dlg.Focus
  iColumnFocus = IFromSt(Right$(stEbFocus, 1))
  stEbFocus = Left$(stEbFocus, 6)
  If stEbFocus <> "ebName" Then
    stEbFocus = Left$(stEbFocus, 5)
    If stEbFocus <> "ebRef" Then stEbFocus = "" 'to denote that focus was not in a column
  End If

  With DB.rgTable(iTableCur)
    iField = IFieldFromIColumn(iColumn)
    Assert iField <= .cField + 1
    If iField = .cField + 1 Then 'this is the blank column
      If .rgfield(iField).stRef <> "" Or .rgfield(iField).stName <> "" Then
        'add another blank column
        .cField = .cField + 1
        If .cField < cFieldMax Then 'don't blank next col. if max no of cols is reached
          With .rgfield(.cField + 1)
            .stName = ""
            .stRef = ""
          End With
        End If
        If iColumn = cColumn Then 'scroll so that the new column is visible
          'scroll only if the focus column will still remain visible
          If stEbFocus = "" Or iColumnFocus <> 1 Then
            rgIFieldCur(iTableCur) = rgIFieldCur(iTableCur) + 1
            iColumnFocus = iColumnFocus - 1 'maintain focus in same logical field
            LoadDisplayColumns 1, cColumn
            If stEbFocus <> "" Then dlg.Focus = stEbFocus & StFromI(iColumnFocus)
          Else
            'Nothing. The added field will not be displayed
          End If
        Else
          LoadDisplayColumns iColumn + 1, iColumn + 1
        End If
        ConfigureScrollbar
      End If
    Else
      If .rgfield(iField).stRef = "" And .rgfield(iField).stName = "" Then
        .cField = .cField - 1
        'Compact fields, get rid of this blank field
        For i = iField To .cField
          .rgfield(i).stName = .rgfield(i + 1).stName
          .rgfield(i).stRef = .rgfield(i + 1).stRef
        Next i
        'blank col at cField + 1
        Assert .cField + 1 <= cFieldMax
        .rgfield(.cField + 1).stName = ""
        .rgfield(.cField + 1).stRef = ""
        If stEbFocus <> "" Then
          iFieldFocus = IFieldFromIColumn(iColumnFocus)
          If iFieldFocus > iField Then iFieldFocus = iFieldFocus - 1 'logical column has shifted le
ft
        End If

        If IFieldFromIColumn(cColumn) > .cField + 1 Then 'will cause more than one dummy col
          rgIFieldCur(iTableCur) = Max(1, rgIFieldCur(iTableCur) - 1)
          LoadDisplayColumns 1, cColumn
        Else
          LoadDisplayColumns iColumn, cColumn
        End If
        If stEbFocus <> "" Then
          Assert iFieldFocus <= .cField + 1
          iColumnFocus = IColumnFromIField(iFieldFocus)
          dlg.Focus = stEbFocus & StFromI(iColumnFocus)
        End If
        ConfigureScrollbar
      End If
    End If
```

```
cSelectFields - 7

  End With
End Sub
'-----------------------------------------------------------------------
'resets all the xxCur variables to 1
'-----------------------------------------------------------------------
Public Sub ReseticurS(DB As DBStruct)
  Dim iTable As Integer
  iTableCur = 1
  For iTable = 1 To DB.cTable
    rgIFieldCur(iTable) = 1
  Next iTable
End Sub
'-----------------------------------------------------------------------
'conversion function to deal with columns' correspondence to fields
'-----------------------------------------------------------------------
Private Function IColumnFromIField(iField As Integer) As Integer
  IColumnFromIField = iField - rgIFieldCur(iTableCur) + 1
End Function
'-----------------------------------------------------------
Private Function IFieldFromIColumn(iColumn As Integer) As Integer
  IFieldFromIColumn = iColumn + rgIFieldCur(iTableCur) - 1
End Function
'-----------------------------------------------------------
Private Function IFromSt(st As String) As Integer
  Assert Len(st) = 1
  IFromSt = Asc(st) - Asc("0")
End Function
'-----------------------------------------------------------
Private Function StFromI(i As Integer) As String
  StFromI = Chr$(Asc("0") + i)
End Function
'-----------------------------------------------------------------------
'Function to check if the given string refers to a single cell. If so it
'standardizes the string, else returns an empty string
'-----------------------------------------------------------------------
Private Function StValidRef(stRef As String, Optional fDispErr)
  Dim cellRef As Range
  Dim st As String
  Dim i As Integer
  Dim nErr As Integer

  If IsMissing(fDispErr) Then fDispErr = True

  On Error Resume Next
  'bug 286
  Set cellRef = Evaluate(stRef)
  nErr = Err
  On Error GoTo 0

  If nErr <> 0 Then
    If fDispErr Then DisplayError ecInvalidRef, """ & stRef & """
    GoTo LBadRef
  End If
  If cellRef.count <> 1 Then
    If fDispErr Then DisplayError ecMultiSelect
    GoTo LBadRef
  End If
  st = cellRef.AddressLocal(external:=True)
  i = InStr(1, st, "]", 0)
  Assert i <> 0
  'bug 275 (use the parent property) and 286
  If Evaluate(st).Parent.Parent.Name <> wbForm.Name Then
    If fDispErr Then DisplayError ecDifferentWorkbook, wbForm.Name
    GoTo LBadRef
  End If
  If Left$(st, 1) = "'" Then
    StValidRef = "'" & Mid$(st, i + 1)
  Else
    StValidRef = Mid$(st, i + 1)
  End If
  Exit Function
LBadRef:
    StValidRef = ""
End Function
'-----------------------------------------------------------------------
'checks to see that all refs are valid, standardizes the ref as well
```

```
cSelectFields - 8

'--------------------------------------------------------------------------
Private Function FRefsValid() As Boolean
  Dim iTable As Integer
  Dim iField As Integer
  Dim stT As String

  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      For iField = 1 To .cField
        If .rgfield(iField).stRef <> "" Then
          stT = StValidRef(.rgfield(iField).stRef)
          If stT = "" Then 'invalid ref
            FRefsValid = False
            MakeFieldVisible iTable, iField
            SetErrorFocus "ebRef" & StFromI(IColumnFromIField(iField))
            Exit Function
          Else
            .rgfield(iField).stRef = stT
          End If
        End If
      Next iField
    End With
  Next iTable
  FRefsValid = True
End Function
'--------------------------------------------------------------------------
'checks to see that all fields of a table have unique names.
'--------------------------------------------------------------------------
Private Function FUniqueNames() As Boolean
  Dim iTable As Integer
  Dim i As Integer
  Dim j As Integer
  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      For i = 1 To .cField - 1 'using n*n alg. nlgn is not worth it
        For j = i + 1 To .cField
          If .rgfield(i).stName = .rgfield(j).stName Then
            DisplayError ecDuplicateName, i, j
            FUniqueNames = False
            MakeFieldVisible iTable, j
            SetErrorFocus "ebName" & StFromI(IColumnFromIField(j))
            Exit Function
          End If
        Next j
      Next i
    End With
  Next iTable
  FUniqueNames = True
End Function
'--------------------------------------------------------------------------
'checks to see that a table name is provided and that the field names are
'all filled and references to fields are all filled. If not returns false
'and sets focus to the bad field. although new dbs currently have only a
'single table this function can handle multiple tables.
'--------------------------------------------------------------------------
Private Function FFieldsFilled() As Boolean
  Dim iTable As Integer
  Dim iField As Integer

  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      If .stName = "" Then
        DisplayError ecMissingTableName, iTable
        SetErrorFocus "ebTable"
        GoTo LNotFilled
      End If
      If .cField = 0 Then
        DisplayError ecNoFields, iTable, .stName
        MakeFieldVisible iTable, 1
        SetErrorFocus "ebRef1"
        GoTo LNotFilled
      End If
      For iField = 1 To .cField
        If .rgfield(iField).stName = "" Then
          DisplayError ecMissingFieldName, iField
          MakeFieldVisible iTable, iField
```

```
          SetErrorFocus "ebName" & StFromI(IColumnFromIField(iField))
          GoTo LNotFilled
        End If
        If .rgfield(iField).stRef = "" Then
          DisplayError ecMissingReference, iField
          MakeFieldVisible iTable, iField
          SetErrorFocus "ebRef" & StFromI(IColumnFromIField(iField))
          GoTo LNotFilled
        End If
      Next iField
    End With
  Next iTable
  FFieldsFilled = True
  Exit Function
LNotFilled:
  FFieldsFilled = False
End Function
'---------------------------------------------------------------------------
'Function to search for a label. Returns the offset of the label or zero
'if no label is found
'---------------------------------------------------------------------------
Private Function iFindLabelLeft(cellRef As Range) As Integer
  Dim colOff As Integer
  With cellRef
    For colOff = -1 To 1 - .Column Step -1
      If Not IsEmpty(.Offset(0, colOff)) Then
        iFindLabelLeft = colOff
        Exit Function
      End If
    Next colOff
  End With
  iFindLabelLeft = 0
End Function
'-------------------------------------------------
Private Function iFindLabelTop(cellRef As Range) As Integer
  Dim rwOff As Integer
  With cellRef
    For rwOff = -1 To 1 - .Row Step -1
      If Not IsEmpty(.Offset(rwOff, 0)) Then
        iFindLabelTop = rwOff
        Exit Function
      End If
    Next rwOff
  End With
  iFindLabelTop = 0
End Function
```

cSelectForms - 1

```
Option Explicit
Option Compare Text
'Option Private Module

Public iDlg As Integer

Private dlg As DialogSheet
Private btnHelp As Button
Private btnCancel As Button
Private btnBack As Button
Private btnNext As Button
Private btnFinish As Button
Private btnSelect As Button
Private btnDelete As Button
Private btnPreview As Button
Private tbPreview As TextBox
Private ddTable As DropDown
Private tbCFile As TextBox
Public lbFiles As ListBox
Private sbColumn As ScrollBar
Private picFields As Picture
'Private picBook As Picture

Private Const cColDisp As Integer = 4
Private Const cColBook As Integer = 6
Private Const cRowBook As Integer = 8

Private iTableCur As Integer 'the table to be displayed in the preview
Private rgIFieldCur(1 To cTableMax) As Integer 'the left most field being displayed in the preview

Private shtFields As Worksheet
Private stFilePreview As String 'The workbook being displayed in the preview, or "" if none

'-------------------------------------------------------------------------
Public Sub Boot()
  'Set aliases for fast access
  Set dlg = rgDe(iDlg).dlg
  With dlg
    Set btnHelp = .Buttons("btnHelp")
    Set btnCancel = .Buttons("btnCancel")
    Set btnBack = .Buttons("btnBack")
    Set btnNext = .Buttons("btnNext")
    Set btnFinish = .Buttons("btnFinish")
    Set btnSelect = .Buttons("btnSelect")
    Set btnDelete = .Buttons("btnDelete")
    Set btnPreview = .Buttons("btnPreview")
    Set tbPreview = .TextBoxes("tbPreview")
    Set ddTable = .DropDowns("ddTable")
    Set tbCFile = .TextBoxes("tbCFile")
    Set lbFiles = .ListBoxes("lbFiles")
    Set sbColumn = .ScrollBars("sbColumn")
    Set picFields = .Pictures("picFields")
  End With
  lbFiles.RemoveAllItems
  Set shtFields = wbAtw.Worksheets("sFormPreview")
  sbColumn.LargeChange = cColDisp - 1
  sbColumn.Min = 1
End Sub
'-------------------------------------------------------------------------
Public Sub Init()
End Sub
'-------------------------------------------------------------------------
Public Sub Frame()
  ResetPreview
  ConfigureSbColumn
  SetPicFields
  SetTbPreviewText
  SetTbCFileText
  dlg.Buttons.Enabled = True
  SetDlgCtrlEnable
  dlg.Focus = "btnSelect"
End Sub
'-------------------------------------------------------------------------
Public Function Dismiss() As Boolean
  Dismiss = False
```

```
  Select Case btnCode
    Case bcNext, bcFinish
      If fInFinishMode Then Exit Function
      dlg.Hide False
      Dismiss = True
    Case bcBack
      btnCode = bcSetNext
      iDlgCur = cSelectFields.iDlg
      dlg.Hide True
    Case Else
      Assert False
  End Select
End Function
'-------------------------------------------------------------------------------
Sub sbColumnChange()
  rgIFieldCur(iTableCur) = sbColumn.Value
  SetPicFields
End Sub
'-------------------------------------------------------------------------------
Sub ddTableChange()
  iTableCur = ddTable.Value
  ConfigureSbColumn
  SetPicFields
End Sub
'-------------------------------------------------------------------------------
Sub lbFilesChange()
  SetDlgCtrlEnable
End Sub
'-------------------------------------------------------------------------------
Sub btnDeleteClick()
  Dim ListIndexSav As Integer

  Assert lbFiles.ListIndex <> 0
  If stFilePreview = lbFiles.List(lbFiles.ListIndex) Then
    stFilePreview = ""
    SetTbPreviewText
    ClearFieldValues
  End If
  ListIndexSav = lbFiles.ListIndex
  lbFiles.RemoveItem Index:=lbFiles.ListIndex
  If lbFiles.ListCount <> 0 Then 'select the next file
    lbFiles.ListIndex = Min(ListIndexSav, lbFiles.ListCount)
  End If
  SetTbCFileText
  SetDlgCtrlEnable
End Sub
'-------------------------------------------------------------------------------
Sub btnSelectClick()
  Dim Files As Variant
  Dim i As Integer

  ' filterIndex:=2 to default to XL files
  'bug 284
  Files = Application.GetOpenFilename( _
    fileFilter:=dsFileFilter, _
    Title:=dsSelectFilesToConvert, _
    MultiSelect:=True _
    )

  If IsArray(Files) Then
    For i = LBound(Files) To UBound(Files)
      lbFiles.AddItem Files(i)
    Next i
  Else
    If Files <> False Then lbFiles.AddItem Files
  End If
  SetTbCFileText
  SetDlgCtrlEnable
End Sub
'-------------------------------------------------------------------------------
Sub btnPreviewClick()
  Dim wb As Workbook
  Dim fWbWasAlreadyOpen As Boolean
  Dim stFileName As String
  Dim iTable As Integer, iField As Integer
  Dim iCol As Integer, iRow As Integer
```

```
  Assert lbFiles.ListIndex <> 0
  If stFilePreview = lbFiles.List(lbFiles.ListIndex) Then Exit Sub

  Application.ScreenUpdating = False

  stFileName = lbFiles.List(lbFiles.ListIndex)
  If FOpenWorkbook(stFileName, wb, fWbWasAlreadyOpen) Then
    Assert Not (wb Is Nothing)
    If FExtractRecordFromWb(wb, DB, Nothing) Then
      iRow = 2
      For iTable = 1 To DB.cTable
        iCol = 1
        For iField = 1 To DB.rgTable(iTable).cField
          shtFields.Cells(iRow, iCol).Value = DB.rgTable(iTable).rgfield(iField).Value
          iCol = iCol + 1
        Next iField
        iRow = iRow + 2
      Next iTable
      stFilePreview = stFileName
      SetTbPreviewText
    End If
    If Not fWbWasAlreadyOpen Then wb.Close savechanges:=False, routeworkbook:=False
  End If
LExit:
  Application.ScreenUpdating = True
End Sub
'---------------------------------------------------------------------------
Sub SetDlgCtrlEnable()
  If lbFiles.ListIndex <> 0 Then
    btnDelete.Enabled = True
    btnPreview.Enabled = True
    dlg.DefaultButton = "btnPreview"
  Else
    btnDelete.Enabled = False
    btnPreview.Enabled = False
    dlg.DefaultButton = "btnSelect"
  End If
End Sub
'---------------------------------------------------------------------------
Sub SetTbPreviewText()
  If stFilePreview = "" Then
    tbPreview.Text = dsPreview
  Else
    tbPreview.Text = dsPreviewOf & StripExtension(StripPath(stFilePreview))
  End If
End Sub
'---------------------------------------------------------------------------
Sub SetTbCFileText()
  Select Case lbFiles.ListCount
    Case 0
      tbCFile.Text = dsNoFiles
    Case 1
      tbCFile.Text = "1" & dsFile
    Case Else
      tbCFile.Text = lbFiles.ListCount & dsFiles
  End Select
End Sub
'---------------------------------------------------------------------------
'blank out everything in the preview. Reload the tables in ddTable, write
'field names in pic.
'---------------------------------------------------------------------------
Sub ResetPreview()
  Dim iTable As Integer
  Dim iField As Integer
  Dim iRow As Integer
  Dim iCol As Integer

  stFilePreview = ""
  ddTable.RemoveAllItems
  shtFields.Cells.ClearContents
  shtFields.Cells.HorizontalAlignment = xlLeft
  iRow = 1
  For iTable = 1 To DB.cTable
    ddTable.AddItem DB.rgTable(iTable).stName
    iCol = 1
```

```
cSelectForms - 4

    For iField = 1 To DB.rgTable(iTable).cField
      shtFields.Cells(iRow, iCol).Value = DB.rgTable(iTable).rgfield(iField).stName
      iCol = iCol + 1
    Next iField
    shtFields.Cells(iRow, iCol).Value = "  "      'write empty string here which will not display,
    shtFields.Cells(iRow + 1, iCol).Value = "  "  'but will cause the contents of the cell to its
                                                  'left from spilling into this
    iRow = iRow + 2
  Next iTable
'  ddTable.DropDownLines = Min(5, DB.cTable)
  ReseteiCurs
  ddTable.Value = iTableCur
End Sub
'-------------------------------------------------------------------------
'clear values of fields from the shtFields
'-------------------------------------------------------------------------
Sub ClearFieldValues()
  Dim iRow As Integer
  Dim iTable As Integer

  iRow = 2
  For iTable = 1 To DB.cTable
    With shtFields
      .Range(.Cells(iRow, 1), .Cells(iRow, DB.rgTable(iTable).cField)).ClearContents
    End With
    iRow = iRow + 2
  Next iTable
End Sub
'-------------------------------------------------------------------------
Sub ConfigureSbColumn()
  With DB.rgTable(iTableCur)
    'cField+1 is the number of columns (one for a blank column). if it is
    'less than or equal to cColDisp, then disable scrollbar
    If .cField + 1 <= cColDisp Then
      sbColumn.Enabled = False
      sbColumn.Max = 1
    Else
      sbColumn.Enabled = True
      sbColumn.Max = (.cField + 1) - cColDisp + 1
    End If
    sbColumn.Value = rgIFieldCur(iTableCur)
  End With
End Sub
'-------------------------------------------------------------------------
Sub SetPicFields()
  Dim iRow As Integer
  Dim iCol As Integer

  iRow = iTableCur * 2 - 1
  iCol = rgIFieldCur(iTableCur)
  With shtFields
    picFields.Formula = "=" & _
      .Range(.Cells(iRow, iCol), .Cells(iRow + 1, iCol + cColDisp - 1)).Address _
        (ReferenceStyle:=xlA1, external:=True)
  End With
End Sub
'-------------------------------------------------------------------------
'resets all the xxCur variables to 1
'-------------------------------------------------------------------------
Sub ReseteiCurs()
  Dim iTable As Integer
  iTableCur = 1
  For iTable = 1 To DB.cTable
    rgIFieldCur(iTable) = 1
  Next iTable
End Sub
```

cSelectWorkbook - 1

```
Option Explicit
Option Compare Text
'Option Private Module

Public iDlg As Integer

Private dlg As DialogSheet
Private btnHelp As Button
Private btnCancel As Button
Private btnBack As Button
Private btnNext As Button
Private btnFinish As Button
Public ddBooks As DropDown 'value is read by other modules to determine if it changed
Private ebTemplate As EditBox

Private febTemplateDirty As Boolean 'has the user typed in the template field

Public stTemplate As String 'absolute path Name of the template being created
Public stForm As String     'Name of the workbook without the extension
Public fStartedOnTemplate As Boolean 'if we started on a template
Public fTemplateIsForm As Boolean 'whether the form itself is going to be updated

Private iBook As Integer 'the index of the selected form
'---------------------------------------------------------------------------
Public Sub Boot()
  'Set aliases for fast access
  Set dlg = rgDe(iDlg).dlg
  With dlg
    Set btnHelp = .Buttons("btnHelp")
    Set btnCancel = .Buttons("btnCancel")
    Set btnBack = .Buttons("btnBack")
    Set btnNext = .Buttons("btnNext")
    Set btnFinish = .Buttons("btnFinish")
    Set ddBooks = .DropDowns("ddBooks")
    Set ebTemplate = .EditBoxes("ebTemplate")
  End With
End Sub
'---------------------------------------------------------------------------
Public Sub Init()
  Dim i As Integer
  Dim wb As Workbook

  'add open workbooks to ddBooks
  With ddBooks
    .RemoveAllItems
    i = 1 'to keep track of the active workbook in the list
    For Each wb In Application.Workbooks
      'add all workbooks that contains at least one worksheet and are not same as this
      If Not (wb.Name = ThisWorkbook.Name) And wb.Worksheets.count > 0 And wb.Windows.count > 0 The
n
        If wb.Windows(1).Visible = True Then
            .AddItem wb.Name
            If wb.Name = ActiveWorkbook.Name Then i = .ListCount
        End If
      End If
    Next
    If .ListCount = 0 Then 'No workbooks
      Beep
      DisplayError ecNoWorkbooks
      End
    End If
'    .DropDownLines = Min(4, .ListCount)
    .Value = i
    ddBooksChange 'sets fields depending on state
  End With
  'put startup path into ebTemplate
  febTemplateDirty = False
  iBook = 0
  Set wbTemplate = Nothing
  fTemplateIsForm = False
  stTemplate = ""
End Sub
'---------------------------------------------------------------------------
Public Sub Frame()
  dlg.Buttons.Enabled = True
```

```
  btnBack.Enabled = False
  SetDlgCtlEnable
  SetFocus "ebTemplate"
End Sub
'-----------------------------------------------------------------------------
Public Function Dismiss() As Boolean
  Dim iType As Integer
  Dim stName As String
  Dim stType As String
  Dim fc As Integer 'code returned by FcFileIsOpen
  Dim stTemplateNew As String

  Dismiss = False
  Select Case btnCode
    Case bcNext, bcFinish
      ' Can't make a template out of a protected book.  Make the template, then protect it.
      If wbForm.ProtectStructure Then
        DisplayError ecProtectedWorkbook, stTemplate
        SetErrorFocus "ebTemplate"
        stTemplate = ""
        Exit Function
      End If

      If iBook <> ddBooks.Value Then 'the form being automated has changed.
        'check to see if we have been started on a template
        If FIsAutoTemplate(wbForm) Then
          fStartedOnTemplate = True
          GetDBNameAndType wbForm.Worksheets(dsTemplateSheet), stName, stType
          iType = IFromStType(stType)
          If iType = 0 Then 'template does not contain a valid DB type
            iType = 1
          End If
          cSelectDatabase.ddDBType.Value = iType
          cSelectDatabase.ebDBName.Text = stName
        Else
          fStartedOnTemplate = False
        End If
        fRefsOutDated = True
        'if wbTemplate existed, kill it, since wbForm has changed now
        If Not (wbTemplate Is Nothing) Then If Not fTemplateIsForm Then KillWorkbook wbTemplate
        cSelectForms.lbFiles.RemoveAllItems 'clear the list of selected files since form changed
        iBook = ddBooks.Value
      End If

      ' dirty the book just in case it's the magic initial book
      If (fStartedOnTemplate = False And wbForm.Worksheets.count > 0 And _
          wbForm.Name = dsProject & "1") Then
        wbForm.Worksheets(1).Cells(1, 1).Value = wbForm.Worksheets(1).Cells(1, 1).Value
      End If

      stTemplateNew = AddExtension(ebTemplate.Text, stTemplateExtension)
      If stTemplate <> stTemplateNew Then 'template name has changed
        'kill old template file if it exists
        If Not (wbTemplate Is Nothing) Then If Not fTemplateIsForm Then KillWorkbook wbTemplate
        stTemplate = stTemplateNew
        fc = FcFileIsOpen(stTemplate)
        fTemplateIsForm = False
        Select Case fc
          Case fcNotOpen
            Set wbTemplate = Nothing 'will open when needed..
          Case fcSameName
            DisplayError ecSameNameFileOpen, StripPath(stTemplate)
            SetErrorFocus "ebTemplate"
            stTemplate = ""
            Exit Function
          Case fcSameFile
            If stPathNameFromWb(wbForm) = stTemplate And wbForm.FileFormat = xlTemplate Then
              fTemplateIsForm = True
              Set wbTemplate = wbForm
            Else
              DisplayError ecAlreadyOpen, stTemplate
              SetErrorFocus "ebTemplate"
              stTemplate = ""
              Exit Function
            End If
          Case Else
```

```
            Assert False
        End Select
      End If

      If fInFinishMode Then Exit Function
      dlg.Hide False
      Dismiss = True
'      dlg.Labels("lblExistTemp1").Visible = True
'      dlg.Labels("lblExistTemp2").Visible = True
'      dlg.Labels("lblNewTemp1").Visible = True
'      dlg.Labels("lblNewTemp2").Visible = True

    Case bcBack
      'back button is disabled in this dlg. we will never come here.
      Assert False
    Case Else
      Assert False
  End Select
End Function

'---------------------------------------------------------------------------
Sub SetDlgCtlEnable()
  Dim fAutoTemp As Boolean
  btnNext.Enabled = ebTemplate.Text <> ""
  btnFinish.Enabled = ebTemplate.Text <> ""

  If (FIsAutoTemplate(wbForm)) Then fAutoTemp = True Else fAutoTemp = False
  dlg.Labels("lblExistTemp1").Visible = fAutoTemp
  dlg.Labels("lblExistTemp2").Visible = fAutoTemp
  dlg.Labels("lblNewTemp1").Visible = Not fAutoTemp
  dlg.Labels("lblNewTemp2").Visible = Not fAutoTemp

End Sub
'---------------------------------------------------------------------------
Public Sub ddBooksChange()
  Dim st As String

  With ddBooks: Set wbForm = Workbooks(.List(.Value)): End With
  wbForm.Activate
  stForm = StripExtension(wbForm.Name)

  If Not febTemplateDirty Then
    If FIsAutoTemplate(wbForm) And wbForm.FileFormat = xlTemplate Then
      st = StripExtension(stPathNameFromWb(wbForm))
    Else
      If TestOS = OS_WIN Then
        st = Application.TemplatesPath & stForm
      Else
        st = Application.StartupPath & ":" & stForm
      End If
    End If
    ebTemplate.Text = st
    ebTemplateChange
  End If
End Sub
'---------------------------------------------------------------------------
Public Sub ebTemplateChange()
  febTemplateDirty = febTemplateDirty And ebTemplate.Text <> ""
  SetDlgCtlEnable
End Sub
```

```
Option Explicit

' DAO 3.0 Constants, hardcoded here to avoid a typelib reference to DAO
Global Const LOCAL_dbOpenDynaset = 2
Global Const LOCAL_dbSystemObject = -2147483646
Global Const LOCAL_dbHiddenObject = 1

' Pre-DAO3.0 constants copied from VB3 library

' Field Data Types
Global Const DB_LONG = 4
Global Const DB_TEXT = 10

' CreateDatabase and CompactDatabase Language constants
Global Const DB_LANG_GENERAL = ";LANGID=0x0809;CP=1252;COUNTRY=0"
```

```
Option Explicit
Option Compare Text
Option Private Module
'--------------------------------------------------------------------------
Public Const cFieldMax As Integer = 100
Public Const cTableMax As Integer = 25
Public Const cDBTypeMax As Integer = 10 'the max number of supported DBTypes

Public Const MaxLong As Long = 2147483647
'--------------------------------------------------------------------------
Type DBTypeEntry
  stDescription As String
  stType As String
  stFilter As String
  stDefaultExtension As String
End Type
'--------------------------------------------------------------------------
Public Type FieldMapEntry
  stName As String 'name of the field
  stRef As String  'the cell reference it refers to
  Value As Variant 'the value of this field (in a record, when the record is retrived)
End Type

Public Type TableMapEntry
  stName As String                          'name of the table
  cField As Integer                         'number of fields
  fKeyExists As Boolean                     'whether the atwiz key is present in the table
  ATWKey As Long                            'the value of the ATW key
  rgfield(1 To cFieldMax) As FieldMapEntry 'The fields in this table
End Type

'all you need to know about the database plus more..
Type DBStruct
  iType As Integer            'index into rgDBType
  stName As String            'name of the database
  cTable As Integer           'the number of tables in the database
  rgTable(1 To cTableMax) As TableMapEntry
                              'the tables in the database
End Type

'Remark: entry in cField + 1 must be initialized and blank for new DBs
'--------------------------------------------------------------------------
Public rgDBType(1 To 10) As DBTypeEntry
Public cDBTypeMac As Integer

'--------------------------------------------------------------------------
Public iExcelDBType As Integer  'the index of Excel DBs in the rgDBType
Public iODBCDBType As Integer   'the index of ODBC  DBs in the rgDBType
'--------------------------------------------------------------------------

Sub InitrgDBType()
  Dim iDBType As Integer

  cDBTypeMac = 0
  If TestDBType1 Then
    If TestOS = OS_WIN Then
        iDBType = RegisterDBType(DBType1Desc, DBType1Type, DBType1Filter, DBType1Ext)
    Else
        iDBType = RegisterDBType(DBType1Desc, DBType1Type, DBType1FilterMac, DBType1ExtMac)
    End If
  End If
  If (iExcelDBTypeString = 1) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 1) Then iODBCDBType = iDBType

  If TestDBType2 Then iDBType = RegisterDBType(DBType2Desc, DBType2Type, DBType2Filter, DBType2Ext)
  If (iExcelDBTypeString = 2) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 2) Then iODBCDBType = iDBType

  If TestDBType3 Then iDBType = RegisterDBType(DBType3Desc, DBType3Type, DBType3Filter, DBType3Ext)
  If (iExcelDBTypeString = 3) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 3) Then iODBCDBType = iDBType

  If TestDBType4 Then iDBType = RegisterDBType(DBType4Desc, DBType4Type, DBType4Filter, DBType4Ext)
  If (iExcelDBTypeString = 4) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 4) Then iODBCDBType = iDBType
```

```
  If TestDBType5 Then iDBType = RegisterDBType(DBType5Desc, DBType5Type, DBType5Filter, DBType5Ext)
  If (iExcelDBTypeString = 5) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 5) Then iODBCDBType = iDBType

  If TestDBType6 Then iDBType = RegisterDBType(DBType6Desc, DBType6Type, DBType6Filter, DBType6Ext)
  If (iExcelDBTypeString = 6) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 6) Then iODBCDBType = iDBType

  If TestDBType7 Then iDBType = RegisterDBType(DBType7Desc, DBType7Type, DBType7Filter, DBType7Ext)
  If (iExcelDBTypeString = 7) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 7) Then iODBCDBType = iDBType

  If TestDBType8 Then iDBType = RegisterDBType(DBType8Desc, DBType8Type, DBType8Filter, DBType8Ext)
  If (iExcelDBTypeString = 8) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 8) Then iODBCDBType = iDBType

  If TestDBType9 Then iDBType = RegisterDBType(DBType9Desc, DBType9Type, DBType9Filter, DBType9Ext)
  If (iExcelDBTypeString = 9) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 9) Then iODBCDBType = iDBType

  If TestDBType10 Then iDBType = RegisterDBType(DBType10Desc, DBType10Type, DBType10Filter, DBType1
0Ext)
  If (iExcelDBTypeString = 10) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 10) Then iODBCDBType = iDBType

  If TestDBType11 Then iDBType = RegisterDBType(DBType11Desc, DBType11Type, DBType11Filter, DBType1
1Ext)
  If (iExcelDBTypeString = 11) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 11) Then iODBCDBType = iDBType

  If TestDBType12 Then iDBType = RegisterDBType(DBType12Desc, DBType12Type, DBType12Filter, DBType1
2Ext)
  If (iExcelDBTypeString = 12) Then iExcelDBType = iDBType
  If (iODBCDBTypeString = 12) Then iODBCDBType = iDBType

End Sub

Function RegisterDBType(stDescription As String, stType As String, stFilter As String, Optional stD
efaultExtension)
  cDBTypeMac = cDBTypeMac + 1
  With rgDBType(cDBTypeMac)
    .stDescription = stDescription
    .stType = stType
    .stFilter = stFilter
    If IsMissing(stDefaultExtension) Then
      .stDefaultExtension = ""
    Else
      .stDefaultExtension = stDefaultExtension
    End If
    .stFilter = stFilter
  End With
  RegisterDBType = cDBTypeMac
End Function
'----------------------------------------------------------------------------
'function to search for the DB type in recognized DB types,
'and return its index, or zero if not found
'----------------------------------------------------------------------------
Function IFromStType(stType As String) As Integer
  Dim i As Integer

  For i = 1 To cDBTypeMac
    If rgDBType(i).stType = stType Then
      IFromStType = i
      Exit Function
    End If
  Next i
  IFromStType = 0
End Function
'----------------------------------------------------------------------------
Function StAddDBTypeExtension(iType As Integer, stName As String) As String
  StAddDBTypeExtension = AddExtension(stName, rgDBType(iType).stDefaultExtension)
End Function
'----------------------------------------------------------------------------
'FInitDBStructure loads the DB structure from the DB if the DB exists,
'initializes it to a table with no fields if the DB does not exist. It also
```

```
'sets fNewDB to indicate whether the DB exists or not. If fMustExist is True
'then it genereatea an error if the DB does not exist. Default for fMustExist
'is false. If fPreserve is true, then if the database is a new database, then
'it will leave the DB.rgTable alone instead of initializing it to a blank table
'fPreserve defaults to false
'-----------------------------------------------------------------------------
Function FInitDBStructure(DB As DBStruct, Optional fMustExist, Optional fPreserve) As Boolean
  Dim fExists As Boolean

  If IsMissing(fMustExist) Then fMustExist = False
  If IsMissing(fPreserve) Then fPreserve = False
  'if it is an excel DB, check to see if file is already open in Excel. If
  'the file is open in Excel we consider the database to be old (fnewdb=false)
  'even if the file does not exist on disk (it has not been saved even once)
  fExists = FFileExists(DB.stName)
  If Not fExists And (DB.iType = iExcelDBType) Then _
    fExists = (FcFileIsOpen(DB.stName) = fcSameFile)

  If fExists Then
    Select Case DB.iType
      Case iExcelDBType
        If Not FLoadExcelDBStructure(DB) Then GoTo LFailure
      Case Else
        If Not FLoadDAODBStructure(DB) Then GoTo LFailure
    End Select
    fNewDB = False
  Else
    If fMustExist Then
      DisplayError ecUnableToOpenDatabase, rgDBType(DB.iType).stDescription, DB.stName
      GoTo LFailure
    ElseIf Not fPreserve Then
      'it is a new database initialize it as one blank table
      DB.cTable = 1
      With DB.rgTable(1)
        .stName = dsInitialTableName
        .cField = 0
        .fKeyExists = False
        'one field to be initialized blank, this is required by the cSelectField Dialog
        With .rgfield(1)
          .stName = ""
          .stRef = ""
        End With
      End With
      fNewDB = True
    End If
  End If
  FInitDBStructure = True
  Exit Function
LFailure:
  FInitDBStructure = False
End Function
'-----------------------------------------------------------------------------
'loads the DB structure from a workbook
'-----------------------------------------------------------------------------
Function FLoadExcelDBStructure(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim iField As Integer
  Dim iRow As Integer
  Dim iCol As Integer
  Dim sht As Worksheet
  Dim val As Variant
  Dim wbDB As Workbook
  Dim fWasAlreadyOpen As Boolean

  'open the database (readonly)
  If Not FOpenWorkbook(DB.stName, wbDB, fWasAlreadyOpen, True) Then GoTo LFailure
  'load fields from the database
  iTable = 0: iRow = 1
  For Each sht In wbDB.Worksheets
    If (iTable = cTableMax) Then GoTo LEndLoadTables
    iTable = iTable + 1
    With DB.rgTable(iTable)
      .stName = sht.Name
      .fKeyExists = False: iField = 0: iCol = 1
      val = sht.Cells(iRow, iCol)
      Do While (Not IsEmpty(val) And iField < cFieldMax)
```

```
        If IsError(val) Then 'bug 732
            DisplayError ecUnableToLoadFields ' Bogus field name
            GoTo LFailure
        End If
        If val <> dsATWKey Then
          iField = iField + 1
          .rgfield(iField).stName = val
        Else
          .fKeyExists = True
        End If
        iCol = iCol + 1: val = sht.Cells(iRow, iCol)
      Loop
    If (iField >= cFieldMax) Then
      DisplayError ecTooManyFields, DB.stName
      GoTo LFailure
    End If
      .cField = iField
      If .cField = 0 Then iTable = iTable - 1 'ignore a sheet with no fields
    End With
  Next
LEndLoadTables:
  DB.cTable = iTable
  If iTable = 0 Then
    DisplayError ecUnableToLoadFields
    If Not fWasAlreadyOpen Then wbDB.Close savechanges:=False, routeworkbook:=False
    GoTo LFailure
  End If
  'close the database
'  Dim wbsav As Workbook        '************BUG IN EXCEL**************
'  Set wbsav = ActiveWorkbook   '************BUG IN EXCEL**************
  If Not fWasAlreadyOpen Then wbDB.Close savechanges:=False, routeworkbook:=False
'  wbsav.Activate               '************BUG IN EXCEL**************
  FLoadExcelDBStructure = True
  Exit Function
LFailure:
  FLoadExcelDBStructure = False
End Function
'-----------------------------------------------------------------------------
Function FLoadDAODBStructure(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim iField As Integer
  Dim iTableDef As Integer
  Dim iDaoField As Integer
  Dim DaoDB As Object 'Database
  Dim daoDBEngine As Object

  'open the database (readonly)
  On Error Resume Next
    Set daoDBEngine = CreateObject("DAO.DBEngine.36")
    Set DaoDB = daoDBEngine.Workspaces(0).OpenDatabase( _
        DB.stName, False, True, rgDBType(DB.iType).stType)
    Set daoDBEngine = Nothing
  On Error GoTo 0
  If Err <> 0 Or DaoDB Is Nothing Then
    DisplayError ecUnableToOpenDatabase, rgDBType(DB.iType).stDescription, DB.stName
    GoTo LFailure
  End If
  'load fields from database
  iTable = 0
'  If (DaoDB.TableDefs.count - 1 >= cTableMax) Then GoTo LUnableToLoadFields ' Too many tables
  For iTableDef = 0 To DaoDB.TableDefs.count - 1 'these objects are 0 based
    With DaoDB.TableDefs(iTableDef)
      If (.Attributes And (LOCAL_dbSystemObject Or LOCAL_dbHiddenObject)) = 0 Then
        If (.Fields.count >= cFieldMax) Then  ' Too many fields
          DisplayError ecTooManyFields, DB.stName
          GoTo LCloseAndFail
        End If
        If (iTable >= cTableMax) Then GoTo LEndLoadTables
        iTable = iTable + 1
        DB.rgTable(iTable).stName = .Name
        DB.rgTable(iTable).fKeyExists = False: iField = 0
        For iDaoField = 0 To .Fields.count - 1
          If .Fields(iDaoField).Name <> dsATWKey Then
            iField = iField + 1
            DB.rgTable(iTable).rgfield(iField).stName = .Fields(iDaoField).Name
          Else
```

```
            DB.rgTable(iTable).fKeyExists = True
          End If
        Next
        DB.rgTable(iTable).cField = iField
        If iField = 0 Then iTable = iTable - 1 'ignore a table with no fields
      End If
    End With
  Next
LEndLoadTables:
  DB.cTable = iTable
  If DB.cTable = 0 Then
LUnableToLoadFields:
    DisplayError ecUnableToLoadFields
LCloseAndFail:
    DaoDB.Close
    GoTo LFailure
  End If
  'close the database
  DaoDB.Close
  FLoadDAODBStructure = True
  Exit Function
LFailure:
  FLoadDAODBStructure = False
End Function
'--------------------------------------------------------------------------------
'blanks the ref fields of all tables in DB.rgTable, and set the xxCur variables to 1
'--------------------------------------------------------------------------------
Sub BlankRefs(DB As DBStruct)
  Dim iTable As Integer
  Dim iField As Integer

  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      For iField = 1 To .cField
        .rgfield(iField).stRef = ""
      Next iField
    End With
  Next iTable
End Sub
'--------------------------------------------------------------------------------
'Creates the database DB.
'--------------------------------------------------------------------------------
Public Function FCreateDB(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim iField As Integer

  Assert fNewDB
  Assert DB.cTable > 0
  Select Case DB.iType
    Case iExcelDBType
      FCreateDB = FCreateExcelDB(DB)
    Case Else
      FCreateDB = FCreateDaoDB(DB)
  End Select
End Function
'--------------------------------------------------------------------------------
Function FCreateExcelDB(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim iField As Integer
  Dim iCol As Integer
  Dim sht As Worksheet
  Dim wbDB As Workbook
  Dim wbsav As Workbook

  Set wbsav = ActiveWorkbook
  Set wbDB = Nothing
  On Error GoTo LError
    Set wbDB = Workbooks.Add(xlWorksheet)
    If DB.cTable > 1 Then _
      wbDB.Worksheets.Add count:=DB.cTable - 1, Type:=xlWorksheet
    For iTable = 1 To DB.cTable
      With DB.rgTable(iTable)
        Assert .cField > 0
        Set sht = wbDB.Worksheets(iTable)
        sht.Name = .stName
        sht.Cells(1, 1).EntireRow.Font.Bold = True
```

```
        iCol = 1
        For iField = 1 To .cField
          sht.Cells(1, iCol).Value = .rgfield(iField).stName
          iCol = iCol + 1
        Next iField
        'add the atw key
        With sht.Cells(1, iCol): .Value = dsATWKey: .EntireColumn.ColumnWidth = 0: End With
      End With
    Next iTable
    wbDB.SaveAs filename:=DB.stName, FileFormat:=xlWorkbook
    wbDB.Close savechanges:=False, routeworkbook:=False
  On Error GoTo 0
  Set wbDB = Nothing
  FCreateExcelDB = True
  Exit Function
LError:
  If Not (wbDB Is Nothing) Then _
    wbDB.Close savechanges:=False, routeworkbook:=False
  DisplayError ecUnableToCreateDatabase, rgDBType(DB.iType).stDescription, DB.stName
  FCreateExcelDB = False
End Function
'---------------------------------------------------------------------------
'
'---------------------------------------------------------------------------
Function FCreateDaoDB(DB As DBStruct) As Boolean
  Dim dbDao As Object 'database
  Dim daoDBEngine As Object
  Dim tbl As Object   'tabledef
  Dim fld As Object   'field
  Dim iTable As Integer
  Dim iField As Integer

  On Error Resume Next
  Set daoDBEngine = CreateObject("DAO.DBEngine.36")
'  This code used to work but doesn't now.  Now create a directory first
  Set dbDao = daoDBEngine.Workspaces(0).CreateDatabase _
  (DB.stName, rgDBType(DB.iType).stType & DB_LANG_GENERAL, 0)
  If (dbDao Is Nothing) Then
    Set dbDao = daoDBEngine.Workspaces(0).OpenDatabase _
      (DB.stName, False, False, rgDBType(DB.iType).stType)
    If (dbDao Is Nothing) Then
      MkDir (DB.stName) ' Create the directory for the database
      Set dbDao = daoDBEngine.Workspaces(0).OpenDatabase _
        (DB.stName, False, False, rgDBType(DB.iType).stType)
    End If
  End If

  Set daoDBEngine = Nothing
  On Error GoTo 0
  If dbDao Is Nothing Then GoTo LFailure

  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      Set tbl = dbDao.CreateTableDef(.stName)

      For iField = 1 To .cField
        Set fld = tbl.CreateField(.rgfield(iField).stName, DB_TEXT)
        tbl.Fields.Append fld
      Next iField
      'create the ATW key
      Set fld = tbl.CreateField(dsATWKey, DB_LONG)
      tbl.Fields.Append fld

      dbDao.TableDefs.Append tbl
    End With
  Next iTable
  dbDao.Close
  FCreateDaoDB = True
  Exit Function
LFailure:
  DisplayError ecUnableToCreateDatabase, rgDBType(DB.iType).stDescription, DB.stName
  FCreateDaoDB = False
End Function
'---------------------------------------------------------------------------
Function IColumnFromStField(shtTable As Worksheet, stField As String) As Integer
  Dim iCol As Integer
```

```
  Dim val As Variant

  iCol = 1
  val = shtTable.Cells(1, iCol).Value
  Do While Not IsEmpty(val)
    If CStr(val) = stField Then
      IColumnFromStField = iCol
      Exit Function
    End If
    iCol = iCol + 1
    val = shtTable.Cells(1, iCol).Value
  Loop
  IColumnFromStField = 0
End Function
'-----------------------------------------------------------------------------
'returns a non zero random Long value
'-----------------------------------------------------------------------------
Function iRandomKey() As Long
  Do
    iRandomKey = Int(2 * (Rnd() - 0.5) * MaxLong)
  Loop While iRandomKey = 0
End Function
'-----------------------------------------------------------------------------
'REVISE REVISE REVISE NO ERROR CHECK!!
'Adds record to the database. For every table that a key exists, and a key is
'provided, it searches for the record with the key. For every table that a key
'exists but no key is provided (is zero) it adds a new record, and places the
'value of the generated key into the DB.rgtable(itable).atwkey. For every table
'that a key does not exist, it appends the record to the table. DBObj is the
'database, which should have been opened writable

'NOTE: We use a random key and then see if it is already in the database. This
'method is FASTER in the average case on indexed databases, and almost as fast
'as finding Max or something like that, on non indexed databases. Consider this:
'What is the probability of finding a random 32bit value in a database with
'say 100000 records?
'-----------------------------------------------------------------------------
Public Function FAddRecordToDB(DB As DBStruct, DBObj As Object, _
                  shtTemplate As Worksheet, fPrompt As Boolean) As Boolean

  Dim iTable As Integer

  Select Case DB.iType
    Case iExcelDBType
      FAddRecordToDB = FAddRecordToExcelDB(DB, DBObj, fPrompt)
    Case Else
      FAddRecordToDB = FAddRecordToDaoDB(DB, DBObj, fPrompt)
  End Select
  If Not (shtTemplate Is Nothing) Then 'save the key values in shttemplate
    For iTable = 1 To DB.cTable
      With DB.rgTable(iTable)
        If .fKeyExists Then ATWKey(shtTemplate, iTable) = .ATWKey
      End With
    Next iTable
  End If
End Function
'-----------------------------------------------------------------------------
Function FAddRecordToExcelDB(DB As DBStruct, ByVal wbDB As Workbook, fPrompt As Boolean) As Boolean
  Dim iTable As Integer
  Dim iField As Integer
  Dim iCol As Integer
  Dim shtTable As Worksheet
  Dim rng As Range
  Dim rngKeyColumn As Range

  On Error GoTo LFailure

  If Not fPrompt Then
    ThisWorkbook.DialogSheets("dSaveDBAlert").OptionButtons("obCreateNew").Value = xlOn ' Set Defau
lt
    GoTo EndAlerts
  End If

  If Not (DB.rgTable(1).fKeyExists) Then GoTo AskCreateNew
  Set shtTable = wbDB.Worksheets(DB.rgTable(1).stName)
  If (shtTable.ProtectContents = True) Then
```

```
    DisplayError ecProtectedDatabase
    GoTo LFailure
  End If
  iCol = IColumnFromStField(shtTable, dsATWKey)
  Set rngKeyColumn = shtTable.Cells(1, iCol).EntireColumn
  rngKeyColumn.ColumnWidth = 20
  Set rng = rngKeyColumn.Find(what:=DB.rgTable(1).ATWKey, after:=rngKeyColumn.Cells(2, 1), lookin:=
xlValues, _
                              lookat:=xlWhole, searchorder:=xlByColumns)
  rngKeyColumn.ColumnWidth = 0

  Application.ScreenUpdating = True

  If Not (rng Is Nothing) Then
    ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obUpdateExist").Value = xlOn ' Set D
efault

    If (ThisWorkbook.DialogSheets("dUpdateDBAlert").Show = False) Then GoTo LFailure
    If (ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obNoUpdate").Value = xlOn) Then
GoTo LFailure
  Else
AskCreateNew:
    ThisWorkbook.DialogSheets("dSaveDBAlert").OptionButtons("obCreateNew").Value = xlOn ' Set Defau
lt

    If (ThisWorkbook.DialogSheets("dSaveDBAlert").Show = False) Then GoTo LFailure
    If (ThisWorkbook.DialogSheets("dSaveDBAlert").OptionButtons("obNoUpdate").Value = xlOn) Then Go
To LFailure
    ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obCreateNew").Value = xlOn
  End If
EndAlerts:

  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      Set shtTable = wbDB.Worksheets(.stName)
      If .fKeyExists Then
        iCol = IColumnFromStField(shtTable, dsATWKey)
        Set rngKeyColumn = shtTable.Cells(1, iCol).EntireColumn
        rngKeyColumn.ColumnWidth = 20
        If .ATWKey = 0 Then 'find a new key
          Randomize
          Do
            .ATWKey = iRandomKey
            Set rng = rngKeyColumn.Find(what:=.ATWKey, after:=rngKeyColumn.Cells(2, 1), lookin:=xlV
alues, _
                                        lookat:=xlWhole, searchorder:=xlByColumns)
          Loop Until rng Is Nothing
          rngKeyColumn.ColumnWidth = 0
        Else                  'search for the record to update
          Set rng = rngKeyColumn.Find(what:=.ATWKey, after:=rngKeyColumn.Cells(2, 1), lookin:=xlVal
ues, _
                                      lookat:=xlWhole, searchorder:=xlByColumns)
          If Not (rng Is Nothing) Or ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obC
reateNew").Value = xlOn Then
            If ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obCreateNew").Value = xlO
n Then
              Randomize
              Do
               .ATWKey = iRandomKey
               Set rng = rngKeyColumn.Find(what:=.ATWKey, after:=rngKeyColumn.Cells(2, 1), lookin:=
xlValues, _
                                           lookat:=xlWhole, searchorder:=xlByColumns)
              Loop Until rng Is Nothing
              rngKeyColumn.ColumnWidth = 0
              GoTo LCreateNew
            End If
            rngKeyColumn.ColumnWidth = 0
            If ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obNoUpdate").Value = xlOn
 Then GoTo LFailure

            Set rng = rng.EntireRow 'set rng to the record to be updated
            GoTo LWriteFields       'skip the appending part
          End If
        End If
        rngKeyColumn.ColumnWidth = 0
      End If
```

```
LCreateNew:
      'append a new record
      Set rng = shtTable.Cells(1, 1).CurrentRegion
      Set rng = rng.Cells(rng.Rows.count + 1, 1).EntireRow
      'save the ATWKey in the record, if the table has ATWKey.
      If .fKeyExists Then rng.Cells(1, iCol).Value = .ATWKey
LWriteFields:
      'at this point rng points to the record to be written. write the record
      For iField = 1 To .cField
        With .rgfield(iField)
          iCol = IColumnFromStField(shtTable, .stName)
          If Not IsEmpty(.Value) Then rng.Cells(1, iCol).Value = .Value
        End With
      Next iField
    End With
  Next iTable

  On Error GoTo 0
  FAddRecordToExcelDB = True
  Application.ScreenUpdating = False
  Exit Function
LFailure:
  Application.ScreenUpdating = False
  FAddRecordToExcelDB = False
End Function
'---------------------------------------------------------------------------
Function FAddRecordToDaoDB(DB As DBStruct, ByVal dbDao As Object, fPrompt As Boolean) As Boolean
  Dim iTable As Integer
  Dim iField As Integer
  Dim rs As Object 'recordset

  On Error GoTo LFailure

  If Not fPrompt Then
    ThisWorkbook.DialogSheets("dSaveDBAlert").OptionButtons("obCreateNew").Value = xlOn ' Set Defau
lt
    GoTo EndAlerts
  End If

  If Not (DB.rgTable(1).fKeyExists) Then GoTo AskCreateNew

  With DB.rgTable(1)
    If .ATWKey = 0 Then GoTo AskCreateNew

    Set rs = dbDao.OpenRecordset(.stName, LOCAL_dbOpenDynaset)
    If (rs.Updatable = False) Then
      rs.Close
      GoTo LErrorWritingFields
    End If

    rs.FindFirst dsATWKey & " = " & .ATWKey 'should use Seek ...
    If rs.NoMatch Then
      rs.Close
      GoTo AskCreateNew
    End If
    rs.Close
  End With

  ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obUpdateExist").Value = xlOn ' Set Def
ault

  If (ThisWorkbook.DialogSheets("dUpdateDBAlert").Show = False) Then GoTo LFailure
  If (ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obNoUpdate").Value = xlOn) Then Go
To LFailure
  GoTo EndAlerts

AskCreateNew:
  ThisWorkbook.DialogSheets("dSaveDBAlert").OptionButtons("obCreateNew").Value = xlOn ' Set Default

  If (ThisWorkbook.DialogSheets("dSaveDBAlert").Show = False) Then GoTo LFailure
  If (ThisWorkbook.DialogSheets("dSaveDBAlert").OptionButtons("obNoUpdate").Value = xlOn) Then GoTo
 LFailure
  ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obCreateNew").Value = xlOn
EndAlerts:
```

```
  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      Set rs = dbDao.OpenRecordset(.stName, LOCAL_dbOpenDynaset)
      If (rs.Updatable = False) Then
        rs.Close
        GoTo LErrorWritingFields
      End If

      If .fKeyExists Then
        If .ATWKey = 0 Or ThisWorkbook.DialogSheets("dUpdateDBAlert").OptionButtons("obCreateNew").
Value = xlOn Then 'find a new key
          Randomize
          Do
            .ATWKey = iRandomKey
            rs.FindFirst dsATWKey & " = " & .ATWKey 'should use Seek ...
          Loop Until rs.NoMatch
        Else                'search for the record to update
          rs.FindFirst dsATWKey & " = " & .ATWKey 'should use Seek ...
          If Not rs.NoMatch Then
            rs.Edit 'set up current record for editing
            GoTo LWriteFields 'skip the appending part
          End If
        End If
      End If
      rs.AddNew
'      rs.Move 0, rs.LastModified 'make the new record current
      'save the ATWKey in the record, if the table has ATWKey
      If .fKeyExists Then rs(dsATWKey) = .ATWKey
LWriteFields:

      On Error GoTo LErrorWritingFields

      'at this point rs's current record points to the record to be written. write the record
      For iField = 1 To .cField
        With .rgfield(iField)
          If Not IsEmpty(.Value) Then rs(.stName) = .Value
        End With
      Next iField
      rs.Update: rs.Close
    End With
  Next iTable

  On Error GoTo 0
  FAddRecordToDaoDB = True
  Exit Function

LErrorWritingFields:
  DisplayError ecCantWriteFields
  ' fall through
LFailure:
  FAddRecordToDaoDB = False
End Function
'---------------------------------------------------------------------------
Public Sub CommitWorkBook()
  Dim wb As Workbook
  Dim sht As Worksheet
  Dim stType As String
  Dim fDBWasAlreadyOpen As Boolean
  Dim DBObj As Object
  Dim daoDBEngine As Object

  On Error GoTo LError
  Set wb = ActiveWorkbook
  On Error GoTo 0
  If Not FIsAutoTemplate(wb) Then
    Beep
    DisplayError ecNotOnTemplate
    GoTo LFailure
  End If
  Set sht = wb.Worksheets(dsTemplateSheet)
  GetDBNameAndType sht, DB.stName, stType
  DB.iType = IFromStType(stType)
  If DB.iType = 0 Then
    DisplayError ecDBTypeNotSupported, stType
    Exit Sub
  End If
```

```
  Application.ScreenUpdating = False
  If Not FInitDBStructure(DB, True) Then GoTo LFailure
  LoadRefsFromTemplate DB, sht
  'open the DB for writing
  Select Case DB.iType
    Case iExcelDBType
      If Not FOpenWorkbook(DB.stName, DBObj, fDBWasAlreadyOpen, False) Then GoTo LFailure
      If DBObj.ReadOnly = True Then
        DisplayError ecDatabaseAlreadyOpen, StripPath(DBObj.Name)
        DBObj.Close savechanges:=False
        GoTo LFailure
      End If
    Case Else
      On Error GoTo LError
        Set daoDBEngine = CreateObject("DAO.DBEngine.36")
        Set DBObj = daoDBEngine.Workspaces(0).OpenDatabase( _
            DB.stName, False, False, rgDBType(DB.iType).stType)
        Set daoDBEngine = Nothing
      On Error GoTo 0
  End Select
  If FExtractRecordFromWb(wb, DB, sht) Then
    FAddRecordToDB DB, DBObj, sht, True 'do we care about the return value here?
  End If

  'close the database, saving the changes
  On Error GoTo LError
    Select Case DB.iType
      Case iExcelDBType
        If Not fDBWasAlreadyOpen Then
          DBObj.Close savechanges:=True
        End If
      Case Else
        DBObj.Close
    End Select
  On Error GoTo 0
  Application.ScreenUpdating = True
  Exit Sub
LError:
  Application.ScreenUpdating = True
  DisplayError ecUnexpectedError
LFailure:
  Application.ScreenUpdating = True
End Sub
'--------------------------------------------------------------------------------
Public Function FCreateTemplateKeys(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim fKeysExist As Boolean

  'check if we need to create any key fields
  fKeysExist = True
  For iTable = 1 To DB.cTable
    fKeysExist = fKeysExist And DB.rgTable(iTable).fKeyExists
  Next iTable

  If Not fKeysExist Then
    Select Case DB.iType
      Case iExcelDBType
        FCreateTemplateKeys = FExcelCreateTemplateKeys(DB)
      Case Else
        FCreateTemplateKeys = FDaoCreateTemplateKeys(DB)
    End Select
  Else
    FCreateTemplateKeys = True
  End If
End Function
'--------------------------------------------------------------------------------
'little or no error checking done here. REVISE * REVISE * REVISE
'--------------------------------------------------------------------------------
Function FExcelCreateTemplateKeys(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim wbDB As Workbook
  Dim fWasAlreadyOpen As Boolean
  Dim wsSheet As Worksheet
  Dim rgCell As Range

  On Error GoTo LError
```

```
  'open the database for writing
  Set wbDB = Nothing
  If Not FOpenWorkbook(DB.stName, wbDB, fWasAlreadyOpen, False) Then GoTo LFailure
  For iTable = 1 To DB.cTable
    If Not DB.rgTable(iTable).fKeyExists Then
      Set wsSheet = wbDB.Worksheets(DB.rgTable(iTable).stName)
      If (wsSheet.ProtectContents = True) Then
        DisplayError ecProtectedDatabase
        If Not fWasAlreadyOpen Then wbDB.Close savechanges:=False, routeworkbook:=False
        GoTo LFailure
      End If

      'access the cell after the last filled column
      Set rgCell = wsSheet.Cells(1, 1).End(xlToRight)
      If (IsEmpty(rgCell)) Then
        Set rgCell = wsSheet.Cells(1, 2) ' only 1 column
      Else
        Set rgCell = rgCell.Cells(1, 2)
      End If
      rgCell.Value = dsATWKey
      rgCell.EntireColumn.ColumnWidth = 0

      DB.rgTable(iTable).fKeyExists = True
    End If
  Next iTable
  If Not fWasAlreadyOpen Then wbDB.Close savechanges:=True, routeworkbook:=False
  Set wbDB = Nothing
  FExcelCreateTemplateKeys = True
  On Error GoTo 0
  Exit Function
LError:
  If Not (wbDB Is Nothing) Then _
    If Not fWasAlreadyOpen Then wbDB.Close savechanges:=False, routeworkbook:=False
  DisplayError ecUnexpectedError 'catch all! should do finer grain error checks
LFailure:
  FExcelCreateTemplateKeys = False
End Function
'-------------------------------------------------------------------------------
Function FDaoCreateTemplateKeys(DB As DBStruct) As Boolean
  Dim iTable As Integer
  Dim daoDBEngine As Object
  Dim dbDao As Object 'database
  Dim tbl As Object 'TableDef
  Dim fld As Object 'Field

  On Error GoTo LError
  Set daoDBEngine = CreateObject("DAO.DBEngine.36")
  'open the database for writing
  Set dbDao = daoDBEngine.Workspaces(0).OpenDatabase( _
        DB.stName, False, False, rgDBType(DB.iType).stType)
  Set daoDBEngine = Nothing

  For iTable = 1 To DB.cTable
    If Not DB.rgTable(iTable).fKeyExists Then
      With dbDao.TableDefs(DB.rgTable(iTable).stName)
        Set fld = .CreateField(dsATWKey, DB_LONG)
        .Fields.Append fld
      End With
      DB.rgTable(iTable).fKeyExists = True
    End If
  Next iTable
  dbDao.Close
  Set dbDao = Nothing
  FDaoCreateTemplateKeys = True
  On Error GoTo 0
  Exit Function
LError:
  If Not (dbDao Is Nothing) Then dbDao.Close
  DisplayError ecUnexpectedError 'catch all! should do finer grain error checks
LFailure:
  FDaoCreateTemplateKeys = False
End Function
```

```
Driver - 1


Option Explicit
Option Compare Text
Option Private Module


'--------------------------------------------------------------------
'This module contains the main entry point for the ATW and the driver that
'controls the order of top level dialogs
'--------------------------------------------------------------------
Type DialogEntry
  dlg As DialogSheet     'the dialog
  stDismissSub As String 'sub to call when this dialog is dismissed
  hc As Integer          'the help context for this dialog
End Type

'--------------------------------------------------------------------
Public Const cDialogMax As Integer = 6

'----------------------------ButtonCodes-------------------------------
Public Const bcHelp = 1
Public Const bcBack = 2
Public Const bcNext = 3
Public Const bcCancel = 4
Public Const bcFinish = 5
Public Const bcSetNext = 6  'a dialog will return this code if it decides
                            'the next dialog itself.

'----------------------------Variables---------------------------------
Public fBooted As Boolean 'whether atw has been booted
Public fBootedCommit   'whether sufficiently initialized for commitworkbook

Public wbForm As Workbook 'The workbook being converted to autotemplate
Public vWbSav As Workbook 'The workbook that was active when atw was called
Public wbAtw As Workbook 'This workbook
Public wbTemplate As Workbook

Public rgDe(1 To cDialogMax) As DialogEntry
Public cDialog As Integer
Public iDlgCur As Integer 'The currently active dialog

Public btnCode As Integer 'The code for button that was pressed

Public fInFinishMode As Boolean 'whether we are in the "finish mode"
Public stErrorFocus As String 'whether focus is to be forced when a dlg appears

Public stPathSeparator As String * 1 'the path separator char
Public stCurrentDir As String 'the directory of Excel when autotemplate wizard was started
                              'we save it since CurDir() might change..
'--------------------------------------------------------------------
'Initialization
'--------------------------------------------------------------------
Sub Boot()
  stPathSeparator = Application.PathSeparator
  Set wbAtw = ThisWorkbook
  InitErrorDef
  InitDialogs
  'bug 280
  EnumerateSubkeysOfJetFiles
  '---------
  InitrgDBType
  cSelectWorkbook.Boot
  cSelectDatabase.Boot
  cSelectFields.Boot
  cConvertYesNo.Boot
  cSelectForms.Boot
  cLastDialog.Boot
  fBooted = True
  fBootedCommit = True
End Sub
Sub BootCommit()
  Set wbAtw = ThisWorkbook
  stPathSeparator = Application.PathSeparator
  InitErrorDef
  'bug 280
  EnumerateSubkeysOfJetFiles
```

```
Driver - 2

  '---------
  InitrgDBType
  fBootedCommit = True
End Sub

Sub Init()
  stCurrentDir = CurDir()
  Set vWbSav = ActiveWorkbook
  fStartedOnTemplate = False
  cSelectWorkbook.Init
  cSelectDatabase.Init
  cSelectFields.Init
  cConvertYesNo.Init
  cSelectForms.Init
  cLastDialog.Init
End Sub
Sub InitCommit()
End Sub
'------------------------------------------------------------------------
'Sub to initialize the rgDe structure.
'Dialogs will appear in the order they are registered.
'------------------------------------------------------------------------
Sub InitDialogs()
  cDialog = 0
  cSelectWorkbook.iDlg = RegisterDialog("SelectWorkbook", 2017)
  cSelectDatabase.iDlg = RegisterDialog("SelectDatabase", 2018)
  cSelectFields.iDlg = RegisterDialog("SelectFields", 2019)
  cConvertYesNo.iDlg = RegisterDialog("ConvertYesNo", 2020)
  cSelectForms.iDlg = RegisterDialog("SelectForms", 2030)
  cLastDialog.iDlg = RegisterDialog("LastDialog", 2021)
End Sub

Function RegisterDialog(stDialog As String, hc As Integer) As Integer
  cDialog = cDialog + 1
  With rgDe(cDialog)
    Set .dlg = wbAtw.DialogSheets("d" & stDialog)
    .stDismissSub = "c" & stDialog & ".Dismiss"
    .hc = hc
  End With
  RegisterDialog = cDialog
End Function


'------------------------------------------------------------------------
'The main entry point into atw. It drives the dialogs and controls sequencing
'------------------------------------------------------------------------
'THE ACTUAL SUB IS IN MODULE "PUBLIC" IN ORDER TO BE ACCESSIBLE FROM OUTSIDE
'Sub StartWizard()
'End Sub
Sub MainLoop()
  fInFinishMode = False
  stErrorFocus = ""
  iDlgCur = cSelectWorkbook.iDlg
  Do
    If fInFinishMode Then 'simulate a user click on next
      btnCode = bcNext
      Application.Run rgDe(iDlgCur).stDismissSub
    Else
      rgDe(iDlgCur).dlg.Show
    End If
    'set the next dialog
    Select Case btnCode
      Case bcNext
        iDlgCur = iDlgCur + 1
      Case bcBack
        iDlgCur = iDlgCur - 1
      Case bcSetNext
        'Nothing. The next dialog has been set by the current dialog
      Case bcFinish
        'just go ahead as if user had hit Next. fInFinishMode will take care of the rest
        iDlgCur = iDlgCur + 1
      Case bcCancel
        Exit Do
      Case Else
        'finish, help and cancel are handled by defbuttonclick
        Assert False
```

```
Driver - 3

    End Select
  Loop Until iDlgCur = cLastDialog.iDlg + 1
End Sub

Sub SetFocus(stFocus As String)
  If stErrorFocus = "" Then
    rgDe(iDlgCur).dlg.Focus = stFocus
  Else
    rgDe(iDlgCur).dlg.Focus = stErrorFocus
    stErrorFocus = ""
  End If
End Sub

Sub SetErrorFocus(stFocus As String)
  If fInFinishMode Then
    EndFinishMode
    stErrorFocus = stFocus
  Else
    rgDe(iDlgCur).dlg.Focus = stFocus
  End If
End Sub

Sub EndFinishMode()
  Assert fInFinishMode
  Application.ScreenUpdating = True
  fInFinishMode = False
  btnCode = bcSetNext
  'now when we go in the main loop we will call dlg.show for this dlg.
End Sub

Sub StartFinishMode()
  Application.ScreenUpdating = False
  fInFinishMode = True
End Sub

'---------------------------------------------------------------------------
' Function to be called when user chooses cancel. It undoes all that has
' been done till now.
'---------------------------------------------------------------------------
Sub ProcessCancel()
  If Not (wbTemplate Is Nothing) Then If Not fTemplateIsForm Then _
    KillWorkbook wbTemplate
  vWbSav.Activate
End Sub
```

```
Errors - 1


Option Explicit
Option Compare Text
Option Private Module

'--------------------------------------------------------------------------
'This module contains most of the error handling setup code.
'In an error handler if the error code is not ATWError then it is an error
'generated by the system. If it is ATWError then it is an error generated by
'ATW, and the errorcode is available in the global variable vec i.e.
'((V)global (E)rror (C)ode
'--------------------------------------------------------------------------
Public Const stHelpFile As String = "xladdin.chm"
'--------------------------------------------------------------------------
'The list of errors is maintained as a array.
'NOTE>>>when project is complete make nErrorMax be equal to the number
'of errors defined at that time.
'If an error string contains "[]" as a substring, an optional parameter is
'placed inside the []s when the displayerror is called on this string
'--------------------------------------------------------------------------
Public Const svrFatal As Integer = 1
Public Const svrWarning As Integer = 2
Public Const svrInform As Integer = 3

Type ErrorDefEntry
  stMsg As String     'The error message
  severity As Integer 'warning or error
  hc As Integer       'The help context
End Type

Const nErrorMax As Integer = 100
Dim nErrorMac As Integer
Dim rgErrorDef(0 To nErrorMax - 1) As ErrorDefEntry


'--------------------------------------------------------------------------
'Error codes
'--------------------------------------------------------------------------
Public Const ecNoError As Integer = 0
Public ecUnexpectedError As Integer 'we missed something!
Public ecNoWorkbooks As Integer
Public ecInvalidRef As Integer
Public ecMultiSelect As Integer
Public ecMissingTableName As Integer
Public ecMissingFieldName As Integer
Public ecMissingReference As Integer
Public ecDuplicateName As Integer
Public ecNoFields As Integer
Public ecUnableToCreateTemplate As Integer
Public ecTooManyFields As Integer
Public ecTemplateNameRequired As Integer
Public ecDatabaseNameRequired As Integer
Public ecDifferentWorkbook As Integer
Public ecUnableToOpenDatabase As Integer
Public ecUnableToOpenFile As Integer
Public ecUnableToFindDatabase As Integer
Public ecUnableToLoadFields As Integer
Public ecFileWasAlreadyOpen As Integer
Public ecSameNameFileOpen As Integer
Public ecExtractRecordError As Integer
Public ecNotSaved As Integer
Public ecAlreadyOpen As Integer
Public ecUnableToCreateDatabase As Integer
Public ecDBStructureChanged As Integer
Public ecDBTypeNotSupported As Integer
Public ecNotOnTemplate As Integer
Public ecProtectedWorkbook As Integer
Public ecProtectedDatabase As Integer
Public ecCantRunIfCantSave As Integer
Public ecExcel7OrGreater As Integer
Public ecCantWriteFields As Integer
Public ecCantInitDAO As Integer
Public ecDatabaseAlreadyOpen As Integer
'--------------------------------------------------------------------------
'Subroutine to initialize the ErrorDef structure with error messages
'--------------------------------------------------------------------------
```

```
Public Sub InitErrorDef()
  nErrorMac = 0
  ecUnexpectedError = RegisterError(esUnexpectedError, svrFatal, 0)
  ecNoWorkbooks = RegisterError(esNoWorkbooks, svrFatal, 0)
  ecInvalidRef = RegisterError(esInvalidRef, svrWarning, 0) 'fill in appropriate hc later
  ecMultiSelect = RegisterError(esMultiSelect, svrWarning, 0)
  ecMissingTableName = RegisterError(esMissingTableName, svrWarning, 0)
  ecMissingFieldName = RegisterError(esMissingFieldName, svrWarning, 0)
  ecMissingReference = RegisterError(esMissingReference, svrWarning, 0)
  ecDuplicateName = RegisterError(esDuplicateName, svrWarning, 0)
  ecNoFields = RegisterError(esNoFields, svrWarning, 0)
  ecUnableToCreateTemplate = RegisterError(esUnableToCreateTemplate, svrWarning, 0)
  ecTemplateNameRequired = RegisterError(esTemplateNameRequired, svrWarning, 0)
  ecDatabaseNameRequired = RegisterError(esDatabaseNameRequired, svrWarning, 0)
  ecDifferentWorkbook = RegisterError(esDifferentWorkbook, svrWarning, 0)
  ecUnableToOpenDatabase = RegisterError(esUnableToOpenDatabase, svrWarning, 0)
  ecUnableToFindDatabase = RegisterError(esUnableToFindDatabase, svrWarning, 0)
  ecUnableToLoadFields = RegisterError(esUnableToLoadFields, svrWarning, 0)
  ecTooManyFields = RegisterError(esTooManyFields, svrWarning, 0)
  ecUnableToCreateDatabase = RegisterError(esUnableToCreateDatabase, svrWarning, 0)
  ecFileWasAlreadyOpen = RegisterError(esFileWasAlreadyOpen, svrWarning, 0)
  ecSameNameFileOpen = RegisterError(esSameNameFileOpen, svrWarning, 0)
  ecExtractRecordError = RegisterError(esExtractRecordError, svrWarning, 0)
  ecNotSaved = RegisterError(esNotSaved, svrWarning, 0)
  ecDBStructureChanged = RegisterError(esDBStructureChanged, svrInform, 0)
  ecDBTypeNotSupported = RegisterError(esDBTypeNotSupported, svrFatal, 0)
  ecUnableToOpenFile = RegisterError(esUnableToOpenFile, svrWarning, 0)
  ecAlreadyOpen = RegisterError(esAlreadyOpen, svrWarning, 0)
  ecNotOnTemplate = RegisterError(esNotOnTemplate, svrFatal, 0)
  ecProtectedWorkbook = RegisterError(esProtectedWorkbook, svrFatal, 0)
  ecProtectedDatabase = RegisterError(esProtectedDatabase, svrFatal, 0)
  ecCantRunIfCantSave = RegisterError(esCantRunIfCantSave, svrFatal, 0)
  ecExcel7OrGreater = RegisterError(esExcel7OrGreater, svrFatal, 0)
  ecCantWriteFields = RegisterError(esCantWriteFields, svrFatal, 0)
  ecCantInitDAO = RegisterError(esCantInitDAO, svrFatal, 0)
  ecDatabaseAlreadyOpen = RegisterError(esDatabaseAlreadyOpen, svrFatal, 0)
End Sub


Function RegisterError(stMsg As String, severity As Integer, hc As Integer) As Integer
  With rgErrorDef(nErrorMac)
    .stMsg = stMsg
    .severity = severity
    .hc = hc
  End With
  RegisterError = nErrorMac
  nErrorMac = nErrorMac + 1
End Function


'--------------------------------------------------------------------------
'DisplayError displays an error message
'--------------------------------------------------------------------------

Public Sub DisplayError(ec As Integer, ParamArray rgArg())
  Dim st As String
  Dim stArg As String
  Dim stMsg As String
  Dim iArg As Integer
  Dim iArgLast As Integer
  Dim i As Integer
  Dim iPrev As Integer
  Dim imsgIcon As Integer

  Const stMissingArg As String = "?"

  iArg = LBound(rgArg)
  iArgLast = UBound(rgArg)

  'construct message in stMsg, from st
  st = rgErrorDef(ec).stMsg
  stMsg = "": iPrev = 1
  i = InStr(iPrev, st, "[", 0)
  Do While i <> 0
    If iArg <= iArgLast Then
      stArg = rgArg(iArg)
```

```
Errors - 3

      iArg = iArg + 1
    Else
      stArg = stMissingArg
    End If
    stMsg = stMsg & Mid$(st, iPrev, i - iPrev) & stArg
    iPrev = InStr(i, st, "]", 0) 'skip the "[.*]" pair
    Assert iPrev <> 0
    iPrev = iPrev + 1 'skip over "]"
    i = InStr(iPrev, st, "[", 0)
  Loop
  stMsg = stMsg & Mid$(st, iPrev)

  Select Case rgErrorDef(ec).severity
    Case svrFatal
      imsgIcon = vbCritical
    Case svrWarning
      imsgIcon = vbExclamation
    Case svrInform
      imsgIcon = vbInformation
  End Select

  'bug 86066
  If rgErrorDef(ec).hc = 0 Then
    MsgBox stMsg, imsgIcon, dsWizardName
  Else
    MsgBox stMsg, imsgIcon, dsWizardName, stHelpFile, rgErrorDef(ec).hc
  End If
End Sub
```

Localization_Code - 1

```
Option Explicit

'-------------------------------------------------
' SetupSummaryInfo
'
' Called only at localization compile time
'-------------------------------------------------
Sub DoUpdateCode()
    ThisWorkbook.Title = dsAddinLongname
    ThisWorkbook.Author = dsAddinAuthor
    ThisWorkbook.Comments = dsAddinDescription
End Sub
```

```
Public - 1


Option Explicit
Option Compare Text
'This is the only module not set to 'Option Private Module. It contains the entrypoint for the
'wizard.

Private Declare Function RegCreateKeyA Lib "ADVAPI32" (ByVal hkey As Long, ByVal sKey As String, By
Ref plKeyReturn As Long) As Long
Private Declare Function RegCloseKey Lib "ADVAPI32" (ByVal hkey As Long) As Long
Private Declare Function RegDeleteValueA Lib "ADVAPI32" (ByVal hkey As Long, ByVal sValueName As St
ring) As Long
Private Declare Function RegSetValueExA Lib "ADVAPI32" (ByVal hkey As Long, ByVal sValueName As Str
ing, ByVal dwReserved As Long, ByVal dwType As Long, ByVal sBuffer As String, ByVal dwLen As Long)
As Long
Private Declare Function RegOpenKeyA Lib "ADVAPI32" (ByVal hkey As Long, ByVal sKey As String, ByRe
f plKeyReturn As Long) As Long

Public Const HKEY_CURRENT_USER = &H80000001
Public Const ERROR_SUCCESS = 0
Public Const REG_SZ = 1

'bug 280
Declare Function RegEnumKeyA Lib "ADVAPI32.DLL" (ByVal hkey As Long, ByVal lIndex As Long, ByVal sS
ubkeyName As String, ByVal lSubkeyBufferLen As Long) As Long
Private Const HKEY_LOCAL_MACHINE = &H80000002
Private Const ERROR_NO_MORE_ITEMS = 259
Private Const SUBKEY_BUFFER_LEN = 500
Public TestDBType1, TestDBType2, TestDBType3, TestDBType4, TestDBType5, TestDBType6 As Boolean
Public TestDBType7, TestDBType8, TestDBType9, TestDBType10, TestDBType11, TestDBType12 As Boolean

Global Const OS_WIN = 0
Global Const OS_MAC = 1

'--------------------------------------------------------------------------
'The main entry point into atw. It drives the dialogs and controls sequencing
'The comment below appears in the tool tip statusbar
'--------------------------------------------------------------------------

Public Sub StartWizard()
'  On Error GoTo LUnExpectedError
  Application.EnableCancelKey = xlDisabled
  If Not fBooted Then Driver.Boot
  Driver.Init

  If TestOS = OS_WIN And (val(Left$(Application.Version, 2)) < 7) Then
        DisplayError ecExcel7OrGreater
        Exit Sub
  ElseIf (val(Left$(Application.Version, 2)) < 5) Then
        DisplayError ecExcel7OrGreater
        Exit Sub
  End If

  On Error GoTo TryAnyway
  If (MenuBars(7).Menus(1).MenuItems(5).Enabled = False Or _
      Toolbars(1).ToolbarButtons(3).Enabled = False) Then
    DisplayError ecCantRunIfCantSave
    Exit Sub
  End If

TryAnyway:
  On Error GoTo 0

  MainLoop
  Exit Sub
LUnExpectedError:
  Select Case Err
    Case 18 'user interrupt occurred
      ProcessCancel
    Case Else
      DisplayError ecUnexpectedError
      'End
      Debug.Print Error$()
      Stop
  End Select
End Sub
```

```
Public - 2

'publicly accessible wrapper around commitworkbook
Public Sub Commit(Optional fSaveAs As Variant)
  Dim st As Variant

  Application.EnableCancelKey = xlDisabled
  If Not fBootedCommit Then Driver.BootCommit
  Driver.InitCommit
'  On Error GoTo LUnExpectedError
  CommitWorkBook
  Exit Sub
LUnExpectedError:
  Select Case Err
    Case 18 'user interrupt occurred
      ProcessCancel
    Case Else
      DisplayError ecUnexpectedError
      'End
      Debug.Print Error$()
      Stop
  End Select
End Sub

'AutoOpen to set up menu
Sub Auto_Open()
    AddToMenuBar
End Sub

Sub Auto_Close()
    RemoveFromMenuBar
End Sub

' AddAdd to handle Addin Manager stuff
Sub Auto_Add()
    AddInitCommand
End Sub

Sub Auto_Remove()
    RemoveInitCommand
End Sub

' Entry point when auto-loaded.
Sub StartWizardStub()
    AddToMenuBar
    StartWizard
End Sub

Sub AddToMenuBar()
    Dim mnuToplevel As Menu
    Dim mnuItem As MenuItem

    On Error Resume Next
    Set mnuToplevel = MenuBars(xlWorksheet).Menus(dsMenuName)
    If Not (mnuToplevel Is Nothing) Then
        Set mnuItem = mnuToplevel.MenuItems(dsMenuCmdName)
        If mnuItem Is Nothing Then
            ' make sure the before item exists, otherwise place at end of menu
            If mnuToplevel.MenuItems(dsMenuCmdBelow) Is Nothing Then
                mnuToplevel.MenuItems.Add Caption:=dsMenuCmdName, OnAction:="'" & ThisWorkbook.Name
 & "'!StartWizard", _
                    StatusBar:=dsMenuStatusBar, HelpFile:=stHelpFile, HelpContextID:=2017
            Else
                mnuToplevel.MenuItems.Add Caption:=dsMenuCmdName, OnAction:="'" & ThisWorkbook.Name
 & "'!StartWizard", _
                    before:=dsMenuCmdBelow, StatusBar:=dsMenuStatusBar, HelpFile:=stHelpFile, HelpC
ontextID:=2017
            End If
        Else
            ' menuitem already exists, make sure it points to the right location
            mnuItem.OnAction = "'" & ThisWorkbook.Name & "'!StartWizard"
        End If

    End If
End Sub

Sub RemoveFromMenuBar()
    On Error Resume Next
```

```
    MenuBars(xlWorksheet).Menus(dsMenuName).MenuItems(dsMenuCmdName).Delete
End Sub

Sub AddInitCommand()
    Dim hkey As Long
    Dim sResult As String

    If TestOS = OS_WIN Then
        sResult = "10," & dsMenuName & "," & dsMenuCmdName & ",'" & ThisWorkbook.FullName & "'!Star
tWizardStub"
        sResult = sResult & "," & dsMenuCmdBelow & ",," & dsMenuStatusBar & "," & stHelpFile & "!20
17"

        If RegCreateKeyA(HKEY_CURRENT_USER, dsInitCmdsKey, hkey) = ERROR_SUCCESS Then
            RegSetValueExA hkey, dsInitCmdIdentifier, 0, REG_SZ, sResult, Len(sResult)
            RegCloseKey hkey
        End If
     End If
End Sub

Sub RemoveInitCommand()
    Dim hkey As Long

    If TestOS = OS_WIN Then
        If RegOpenKeyA(HKEY_CURRENT_USER, dsInitCmdsKey, hkey) = ERROR_SUCCESS Then
            RegDeleteValueA hkey, dsInitCmdIdentifier
            RegCloseKey hkey
        End If
    End If
End Sub

'---------------------------------------------------------------------------
Sub AutoOpenFromTemplate()
  If TestOS = OS_WIN And (val(Left$(Application.Version, 2)) < 7) Then
        MsgBox esExcel7OrGreater, vbOKOnly
    Exit Sub
  ElseIf (val(Left$(Application.Version, 2)) < 5) Then
        MsgBox esExcel7OrGreater, vbOKOnly
    Exit Sub
  End If

  ActiveWorkbook.OnSave = ThisWorkbook.Name & "!Public.Commit"
End Sub

'---------------------------------------------------------------------------
' the default button press handler, for wizard-buttons
'---------------------------------------------------------------------------
Public Sub DefButtonClick()
  Dim stCaller As String
  Dim fDismissed As Boolean

  stCaller = Application.Caller
  Select Case stCaller
    Case "btnCancel"
      btnCode = bcCancel
      ProcessCancel
      rgDe(iDlgCur).dlg.Hide True
      Exit Sub 'don't need to call the dismiss sub
    Case "btnNext"
      btnCode = bcNext
    Case "btnFinish"
      btnCode = bcFinish
    Case "btnBack"
      btnCode = bcBack
    Case "btnHelp"
      Application.Help stHelpFile, rgDe(iDlgCur).hc
      Exit Sub
    Case Else
      Assert False
  End Select

  fDismissed = Application.Run(rgDe(iDlgCur).stDismissSub)

  If stCaller = "btnFinish" Then
    'if the current dlg has been hidden ==> there was no error in it, then go into finishmode
    If fDismissed Then StartFinishMode
```

```
  End If
End Sub


'---------------------------------------------------------------------
' bug 280 > check the ISAM driver available
'---------------------------------------------------------------------
Sub EnumerateSubkeysOfJetFiles()
    Dim hkeyJetRoot As Long
    Dim i As Integer
    Dim s As String
    Dim lResult As Long
    Dim r As Range

    TestDBType1 = True
    TestDBType2 = False
    TestDBType3 = False
    TestDBType4 = False
    TestDBType5 = False
    TestDBType6 = False
    TestDBType7 = False
    TestDBType8 = False
    TestDBType9 = False
    TestDBType10 = False
    TestDBType11 = False
    TestDBType12 = False

    If TestOS = OS_WIN Then
        If RegOpenKeyA(HKEY_LOCAL_MACHINE, "Software\Microsoft\Jet\4.0\ISAM Formats", hkeyJetRoot)
<> ERROR_SUCCESS Then
            Exit Sub
        End If

        For i = 0 To 50
            s = Space(SUBKEY_BUFFER_LEN)
            lResult = RegEnumKeyA(hkeyJetRoot, i, s, SUBKEY_BUFFER_LEN)
            Select Case lResult
            Case ERROR_SUCCESS:
                s = Left(s, InStr(1, s, Chr(0), 0) - 1)
                Select Case s
                    Case "dBase 5.0"
                    Case "dBase III"
                        TestDBType3 = True
                        TestDBType2 = True
                    Case "dBase IV"
                        TestDBType4 = True
                        TestDBType2 = True
                    Case "FoxPro 2.0"
                        TestDBType5 = True
                        TestDBType2 = True
                    Case "FoxPro 2.5"
                        TestDBType6 = True
                        TestDBType2 = True
                    Case "FoxPro 2.6"
                    Case "FoxPro 3.0"
                    Case "dBase 5.0"
                    Case "Jet 2.x"
                    Case "Paradox 3.X"
                    Case "Paradox 4.X"
                    Case "Paradox 5.X"
                    Case "Text"
                End Select
            Case ERROR_NO_MORE_ITEMS:
                Exit For

            Case Else
                MsgBox "Unknown error from registry enumeration."
                GoTo Bail
            End Select
        Next

Bail:
        RegCloseKey hkeyJetRoot
    End If
End Sub
```

```
Public - 5

'-------------------------------------------------
' Test OS
'-------------------------------------------------
Function TestOS() As Integer
    If InStr(1, Application.OperatingSystem, "win", 1) > 0 Then TestOS = OS_WIN
    If InStr(1, Application.OperatingSystem, "mac", 1) > 0 Then TestOS = OS_MAC
End Function
```

SetupStub Code - 1

```
Option Explicit
Option Private Module

Private Function SubstituteNames(ByVal sInput As String) As String
    Dim sNew As String

    sNew = Application.Substitute(sInput, "%ADDINNAME%", ThisWorkbook.Name)
    sNew = Application.Substitute(sNew, "%CANNOTFINDALERT%", esCannotFindWizard)
    SubstituteNames = sNew
End Function

Function SetupLoaderStubInTemplate(wbToModify As Workbook) As Worksheet
    Dim wsNewXLM As Worksheet
    Dim rCodeBase As Range
    Dim rCodeStatements As Range
    Dim rCurrentStatement As Range
    Dim iRowOffset As Integer
    Dim shtAct As Worksheet

    Set shtAct = wbToModify.ActiveSheet

    '\ remove any existing AutoOpen plies
    On Error GoTo AddLoadStubSheet
    wbToModify.Sheets("AutoOpen Stub Data").Delete

AddLoadStubSheet:
    On Error GoTo 0

    '\ add new intl macro sheet.  BUG: Excel doesn't allow you to insert at the end of
    '\ a book.  Put at beginning, then should probably move/hide.
    Set wsNewXLM = wbToModify.Sheets.Add(before:=wbToModify.Sheets(1), Type:=xlExcel4IntlMacroSheet
)

    wsNewXLM.Name = "AutoOpen Stub Data"

    '\ add AutoOpen macro to this sheet.  References are relative so the code can be
    '\ repositioned as necessary.
    Set rCodeBase = wsNewXLM.Cells(1, 1)
    iRowOffset = 0

    Set rCodeStatements = ThisWorkbook.Sheets("AutoOpen Stub Data").Range("StubData")
    For Each rCurrentStatement In rCodeStatements
        rCodeBase.Offset(iRowOffset, 0).FormulaR1C1 = "=" & SubstituteNames(rCurrentStatement.Value
)
        iRowOffset = iRowOffset + 1
    Next

    '\ Define the topmost cell as an AutoOpen for the template book.
    '\ BUG: SHOULD PROBABLY USE A HIDDEN NAME HERE, WHICH REQUIRES USE OF NAMES::ADD
    rCodeBase.Name = "Auto_Open21"

    wsNewXLM.Visible = xlVeryHidden

    shtAct.Activate

    Set SetupLoaderStubInTemplate = wsNewXLM
End Function
```

```
Option Explicit
Option Compare Text
'Option Private Module

'------------------------------------------------------------------------
'This module contains most of the strings used in the messages. It does not
'handle all the localization problems. Labels on buttons etc have to be done
'in a different manner.
'Each string starts with the prefix ds which stands for Defined Strings (??)
'------------------------------------------------------------------------


'------------------------------------------------------------------------
'Messages and other defined strings
'------------------------------------------------------------------------

' Don't localize these identifiers!!!
Public Const dsIdentificationCookie = "AutoTemplateWizardDONTMESSWITHIT"
Public Const dsTemplateSheet = "TemplateInformation"
Public Const dsInitCmdsKey = "Software\Microsoft\Office\8.0\Init Commands"
Public Const dsInitCmdIdentifier = "WZTemplt"
Public Const dsATWKey = "WT_RECID"

' Localize these identifiers:
' NOTE: Don't forget to localize the summary info
Public Const dsWizardName = "Assistant Modèle avec suivi des données"
Public Const dsTemplateTitle = _
  "Modèle créé par l'Assistant Modèle"
Public Const dsDatabaseType = "Type de base de données:"
Public Const dsDatabaseLocation = "Emplacement de la base de données:"
Public Const dsTableName = "Nom de la table:"
Public Const dsNumberOfTables = "Nombre de tables:"
Public Const dsNumberOfFields = "Nombre de champs:"
Public Const dsFieldName = "Nom de champ:"
Public Const dsRefersTo = "Réfère à:"
Public Const dsPreview = "Aperçu"
Public Const dsPreviewOf = "Aperçu de "
Public Const dsFile = " Fichier"
Public Const dsNoFiles = "Aucun fichier"
Public Const dsFiles = " Fichiers"
Public Const dsInitialTableName = "Table1"
Public Const dsExcelTemplates = "Excel\" ' name of Excel subdirectory in the \office\templates dir
Public Const dsProject = "Class" ' name of initial book when you boot XL

Public Const dsDatabase = "Base de données: "
Public Const dsTemplate = "Modèle: "

Public Const dsTypeNameForTemplate = "Tapez un nom de modèle: "

'bug 284
Public Const dsSelectDatabaseFile = "Sélectionnez un fichier de base de données"
Public Const dsSelectFilesToConvert = "Sélectionnez les fichiers à convertir"


Public Const dsOpenTip = _
"Pour ouvrir un modèle dans Microsoft Excel, maintenez MAJ enfoncée et choisissez OK dans la boîte
de dialogue Fichier Ouvrir"

Public Const dsFileFilter = "Fichiers Microsoft Excel (*.xl*),*.xl*, Tous (*.*),*.*"

Public Const dsMenuName = "Données"
Public Const dsMenuCmdBelow = "Consolider..."
Public Const dsMenuCmdName = "Assistant &Modèle..."
Public Const dsMenuStatusBar = "Crée des modèles de feuilles de calcul avec un suivi de données"

Public Const dsAddinLongname = "Assistant Modèle avec suivi des données"
Public Const dsAddinAuthor = "(c) 1995 Microsoft Corporation"
Public Const dsAddinDescription = "Permet de créer des modèles de formulaires avec suivi des donnée
s"


'------------------------------------------------------------------------
'Error Strings: All strings should start with es for Error String
```

```
'--------------------------------------------------------------------------
Public Const esUnexpectedError = "Erreur inattendue."
'bug 283
Public Const esNoWorkbooks = "Aucun classeur ouvert ne contient de feuille de calcul valide."
Public Const esInvalidRef = "[refstr] n'est pas une référence de cellule valide."
Public Const esMultiSelect = "Vous ne pouvez sélectionner qu'une seule cellule à la fois."
Public Const esMissingTableName = "Vous devez spécifier un nom pour la table [tablenum]."
Public Const esMissingFieldName = "Vous devez spécifier un nom pour le champ [fieldnum]."
Public Const esMissingReference = "Vous devez spécifier une référence de cellule pour le champ [fie
ldnum]."
Public Const esDuplicateName = "Les champs [fieldnum1] et [fieldnum2] possèdent le même nom."
Public Const esNoFields = "La table [tablenumber] ([tablename]) ne contient pas de champ."
Public Const esUnableToCreateTemplate = "Impossible de créer le modèle [templatename]."
Public Const esTemplateNameRequired = "Vous devez spécifier un nom de modèle."
Public Const esDatabaseNameRequired = "Vous devez spécifier un nom de base de données."
Public Const esDifferentWorkbook = "Vous devez sélectionner des cellules du classeur [workbookname]
."
Public Const esFileWasAlreadyOpen = "Un fichier nommé [filename] est déjà ouvert."
Public Const esSameNameFileOpen = "Un document nommé [docName] est déjà ouvert."

Public Const esUnableToOpenDatabase = "Impossible d'ouvrir [dbdescription] [dbName]."
Public Const esUnableToFindDatabase = "Impossible de trouver [dbdescription] [dbName]."
Public Const esUnableToCreateDatabase = "Impossible de créer [dbdescription] [dbname]."
Public Const esUnableToLoadFields = "Impossible de lire la base de données."
Public Const esTooManyFields = "Trop de champs se trouvent dans la base de données [dbName]."
Public Const esExtractRecordError = "Impossible de trouver les données dans le classeur."
Public Const esNotSaved = "[Filename] n'est pas enregistré."
Public Const esCantWriteFields = "Impossible d'écrire des données dans la base de données. Assurez-
vous que la base de données n'est pas protégée en écriture et que les noms de champs ne sont pas mo
difiés. "
Public Const esCantInitDAO = "Impossible d'initialiser DAO."

Public Const esDBStructureChanged = "La structure de la base de données est différente de celle gér
ée par le modèle."
Public Const esDBTypeNotSupported = "Le type de base de données [type] n'est pas pris en charge."
Public Const esUnableToOpenFile = "Impossible d'ouvrir le fichier [filename]."
Public Const esAlreadyOpen = "Le fichier [filename] est déjà ouvert."
Public Const esNotOnTemplate = "Ce classeur n'a pas été créé à partir de l'Assistant Modèle."
Public Const esCannotFindWizard = "Impossible d'ouvrir l'Assistant Modèle."
Public Const esProtectedWorkbook = "Ce classeur est protégé. Sélectionnez un autre classeur."
Public Const esProtectedDatabase = "Cette base de données est protégée. Sélectionnez-en une autre."
Public Const esCantRunIfCantSave = "Impossible d'exécuter l'Assistant Modèle lorsque Excel est inco
rporé dans une autre application."
Public Const esExcel7OrGreater = "Pour utiliser l'Assistant Modèle, vous devez exécuter Excel 97 ou
 une version ultérieure."
Public Const esDatabaseAlreadyOpen = "La base de donnees [dbName] est en lecture seule ou est ouver
te par un autre utilisateur."

' Describe Databases here
Public Const DBType1Desc = "Classeur Microsoft Excel"
Public Const DBType1Type = "Excel 5.0"
Public Const DBType1Ext = "xls"
Public Const DBType1ExtMac = ""
Public Const DBType1Filter = "Fichiers Microsoft Excel (*.xls),*.xls"
Public Const DBType1FilterMac = "XLS"


Public Const DBType2Desc = "Base de données Access"
Public Const DBType2Type = ""
Public Const DBType2Ext = "mdb" 'connect string for jet dbms is empty
Public Const DBType2Filter = "Fichiers Microsoft Access (*.mdb),*.mdb"


Public Const DBType3Desc = "Base de données dBASE III"
Public Const DBType3Type = "dBASE III"
Public Const DBType3Ext = ""
Public Const DBType3Filter = "Fichiers dBase (*.db*),*.db*"


Public Const DBType4Desc = "Base de données dBASE IV"
Public Const DBType4Type = "dBASE IV"
Public Const DBType4Ext = ""
Public Const DBType4Filter = "Fichiers dBase (*.db*),*.db*"


Public Const DBType5Desc = "Base de données FoxPro 2.0"
```

```
Public Const DBType5Type = "FoxPro 2.0"
Public Const DBType5Ext = ""
Public Const DBType5Filter = "Fichiers Microsoft FoxPro (*.db*),*.db*"

Public Const DBType6Desc = "Base de données FoxPro 2.5"
Public Const DBType6Type = "FoxPro 2.5"
Public Const DBType6Ext = ""
Public Const DBType6Filter = "Fichiers Microsoft FoxPro (*.db*),*.db*"


Public Const DBType7Desc = "Base de données Paradox 3.x"
Public Const DBType7Type = "Paradox 3.x"
Public Const DBType7Ext = ""
Public Const DBType7Filter = "Fichiers Paradox (*.db),*.db"


Public Const DBType8Desc = "Base de données Paradox"
Public Const DBType8Type = "Paradox 4.x"
Public Const DBType8Ext = ""
Public Const DBType8Filter = "Fichiers Paradox (*.db),*.db"


Public Const DBType9Desc = "Source de données ODBC"
Public Const DBType9Type = "ODBC"
Public Const DBType9Ext = ""
Public Const DBType9Filter = "Tous les fichiers (*.*),*.*"


Public Const DBType10Desc = ""   ' Put "" for Desc of blank DBs
Public Const DBType10Type = ""
Public Const DBType10Ext = ""
Public Const DBType10Filter = ""


Public Const DBType11Desc = ""
Public Const DBType11Type = ""
Public Const DBType11Ext = ""
Public Const DBType11Filter = ""


Public Const DBType12Desc = ""
Public Const DBType12Type = ""
Public Const DBType12Ext = ""
Public Const DBType12Filter = ""

Public Const iExcelDBTypeString = 1 ' Excel is the 1st DBType
Public Const iODBCDBTypeString = 9 ' ODBC is the 9th DBType

Public Const dsDatabaseName = "Base de données"
Public Const PutDBAfterName = False  ' True gives "sheet Database", false "Database sheet"
Public Const stWorkbookExtension = "xls"
Public Const stTemplateExtension = "xlt"
```

TemplateUtil - 1

```
Option Explicit
Option Compare Text

Option Private Module

'---------------------------------------------------------------------------
Public Function FIsAutoTemplate(wb As Workbook) As Boolean
  Dim sht As Worksheet

  On Error GoTo LNotAutoTemplate
    Set sht = wb.Worksheets(dsTemplateSheet)
  On Error GoTo 0
  If sht.Cells(1, 1).Value <> dsIdentificationCookie Then GoTo LNotAutoTemplate
  FIsAutoTemplate = True
  Exit Function
LNotAutoTemplate:
  FIsAutoTemplate = False
End Function
'---------------------------------------------------------------------------
'adds a blank sheet named dsTemplateSheet to the workbook. If such a sheet
'already exists it clears out the contents of the sheet
'---------------------------------------------------------------------------
Sub CreateShtTemplate(wb As Workbook)
  Dim sht As Worksheet
  Dim shtAct As Worksheet
  Dim shtStub As Worksheet

  On Error GoTo LError
    Set sht = wb.Worksheets(dsTemplateSheet)
  On Error GoTo 0
  sht.Cells.Clear
  Exit Sub
LError:
  With wb.Sheets:
    Set shtAct = wb.ActiveSheet
    If (wb.Sheets.count > 1) Then
      Set sht = .Add(after:=.Item(.count - 1), Type:=xlWorksheet)
    Else
      Set sht = .Add(Type:=xlWorksheet)
    End If
    sht.Move after:=.Item(.count)
    sht.Name = dsTemplateSheet
    shtAct.Activate
  End With
  Set shtStub = SetupLoaderStubInTemplate(wb)
End Sub
'---------------------------------------------------------------------------
'removes dsTemplateSheet from wb if it exists
'---------------------------------------------------------------------------
Sub RemoveShtTemplate(wb As Workbook)
  Application.DisplayAlerts = False
  On Error Resume Next
    With wb.Worksheets(dsTemplateSheet): .Visible = True: .Delete: End With
  On Error GoTo 0
  Application.DisplayAlerts = True
End Sub
'---------------------------------------------------------------------------
Sub WriteShtTemplate(shtTemplate As Worksheet, DB As DBStruct)
  Dim rng As Range
  Dim iRow As Integer
  Dim iCol As Integer
  Dim iField As Integer
  Dim iTable As Integer
  Dim nErr As Integer

  On Error GoTo LFailure

  Assert Not (shtTemplate Is Nothing)
  Set rng = shtTemplate.Cells

  iRow = 1: iCol = 1

  rng.Cells(iRow, iCol).Value = dsIdentificationCookie: iRow = iRow + 1

  rng.Cells(iRow, iCol).Value = dsDatabaseType: iCol = iCol + 1
```

```
  rng.Cells(iRow, iCol).Value = rgDBType(DB.iType).stType: iCol = 1: iRow = iRow + 1

  rng.Cells(iRow, iCol).Value = dsDatabaseLocation: iCol = iCol + 1
  rng.Cells(iRow, iCol).Value = DB.stName: iCol = 1: iRow = iRow + 1

  rng.Cells(iRow, iCol).Value = "Reserved": iRow = iRow + 1

  rng.Cells(iRow, iCol).Value = dsNumberOfTables: iCol = iCol + 1
  rng.Cells(iRow, iCol).Value = DB.cTable: iCol = 1: iRow = iRow + 1

  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      rng.Cells(iRow, iCol).Value = iTable: iCol = iCol + 1
      rng.Cells(iRow, iCol).Value = dsTableName: iCol = iCol + 1
      rng.Cells(iRow, iCol).Value = .stName: iCol = iCol + 1
      rng.Cells(iRow, iCol).Value = dsNumberOfFields: iCol = iCol + 1
      rng.Cells(iRow, iCol).Value = .cField: iCol = 1: iRow = iRow + 1

      rng.Cells(iRow, iCol).Value = dsFieldName
      rng.Cells(iRow + 1, iCol).Value = dsRefersTo: iCol = iCol + 1

      For iField = 1 To .cField
        On Error Resume Next
        rng.Cells(iRow, iCol).Value = .rgfield(iField).stName
        nErr = Err
        On Error GoTo LFailure
        If nErr <> 0 Then DisplayError ecUnableToLoadFields ' Bogus field name

        rng.Cells(iRow + 1, iCol).FormulaLocal = "=" & .rgfield(iField).stRef: iCol = iCol + 1
      Next iField
      iCol = 1: iRow = iRow + 2
      rng.Cells(iRow, iCol).Value = "Reserved": iRow = iRow + 1
    End With
  Next iTable
  Exit Sub

LFailure:
  DisplayError ecUnableToLoadFields
End Sub
'---------------------------------------------------------------------------
Property Get ATWKey(sht As Worksheet, iTable As Integer) As Long
  Dim val As Variant

  val = sht.Cells(2 + iTable * 4, 7)
  If IsEmpty(val) Or Not IsNumeric(val) Then
    ATWKey = 0
  Else
    ATWKey = Int(val)
  End If
End Property
'---------------------------------------------------------------------------
Property Let ATWKey(sht As Worksheet, iTable As Integer, keyvalue As Long)
  sht.Cells(2 + iTable * 4, 7).Value = keyvalue
End Property
'---------------------------------------------------------------------------
'Get the name and type of database from the template information sheet
'---------------------------------------------------------------------------
Public Sub GetDBNameAndType(shtTemplate As Worksheet, stName As String, stType As String)
  With shtTemplate
    stName = .Cells(3, 2)
    stType = .Cells(2, 2)
  End With
End Sub
'---------------------------------------------------------------------------
Public Function FExtractRecordFromWb(wb As Workbook, DB As DBStruct, _
  shtTemplate As Worksheet) As Boolean

  Dim iTable As Integer
  Dim iField As Integer
  Dim stRef As String
  Dim st As String
  Dim i As Integer
  Dim ws As Worksheet
  Dim rg As Range

  On Error GoTo LError
```

```
'  st = "[" & wb.Name & "]"
  For iTable = 1 To DB.cTable
    With DB.rgTable(iTable)
      For iField = 1 To .cField
        stRef = .rgfield(iField).stRef
        If stRef <> "" Then
          'bug 275 and 286 and 2003
          Set ws = Evaluate(stRef).Parent
          Set rg = Evaluate(stRef)
          st = "'[" & wb.Name & "]" & ws.Name & "'!" & rg.Address
          'st = Evaluate(stRef).Address(external:=True)
          DB.rgTable(iTable).rgfield(iField).Value = Evaluate(st).Value
        Else
          DB.rgTable(iTable).rgfield(iField).Value = Empty
        End If
      Next iField
    End With
    If Not (shtTemplate Is Nothing) Then
      DB.rgTable(iTable).ATWKey = ATWKey(shtTemplate, iTable)
    Else
      DB.rgTable(iTable).ATWKey = 0
    End If
  Next iTable
  FExtractRecordFromWb = True
  Exit Function
LError:
  FExtractRecordFromWb = False
  DisplayError ecExtractRecordError, wb.Name
End Function
'-------------------------------------------------------------------------------
'intimately related to UpdateTemplate
'merges information about refs available from the template, into the information about
'the fields available from the database. uses n*n alg. can be done in nlgn (don't even
'think about it!)
'-------------------------------------------------------------------------------
Sub LoadRefsFromTemplate(DB As DBStruct, shtTemplate As Worksheet)
  Dim rng As Range
  Dim iRow As Integer
  Dim iCol As Integer
  Dim iField As Integer
  Dim iTable As Integer
  Dim iTableSht As Integer
  Dim iFieldSht As Integer
  Dim cTableSht As Integer
  Dim cFieldSht As Integer
  Dim st As String
  Dim fStructureChanged As Boolean

  Assert Not (shtTemplate Is Nothing)
  BlankRefs DB
  fStructureChanged = False
  Set rng = shtTemplate.Cells
  cTableSht = rng.Cells(5, 2).Value
  iRow = 6
  For iTableSht = 1 To cTableSht
    st = rng.Cells(iRow, 3).Value 'the name of the table
    cFieldSht = rng.Cells(iRow, 5).Value: iRow = iRow + 1
    'find a table with the same name in DB.rgTable
    For iTable = 1 To DB.cTable
      If DB.rgTable(iTable).stName = st Then Exit For
    Next iTable
    If iTable <= DB.cTable Then 'table found
      With DB.rgTable(iTable)
        iCol = 2
        For iFieldSht = 1 To cFieldSht
          'find a field with the same name in DB.rgTable.rgField
          st = rng.Cells(iRow, iCol).Value
          For iField = 1 To .cField
            If .rgfield(iField).stName = st Then Exit For
          Next iField
          If iField <= .cField Then 'field found
            .rgfield(iField).stRef = Mid$(rng.Cells(iRow + 1, iCol).FormulaLocal, 2)
          Else
            fStructureChanged = True
          End If
```

```
          iCol = iCol + 1
        Next iFieldSht
      End With
    Else
      fStructureChanged = True
    End If
    iRow = iRow + 3
  Next iTableSht
  If fStructureChanged Then
    DisplayError ecDBStructureChanged
  End If
End Sub
'---------------------------------------------------------------------------
'FcreateTemplate creates the template workbook. If the workbook is same as
'the form (fTemplateIsForm), it just sets the wbTemplate to wbForm. Otherwise
'it copies wbForm into a new workbook and set wbTemplate to the new workbook
'wbTemplate file exists on the disk when it is done. And ofcourse, wbTemplate
'is open in Excel.
'---------------------------------------------------------------------------
Function FCreateTemplate(stTemplate, wbForm As Workbook) As Boolean
  Dim wbsav As Workbook
  Dim stTmpFile As String
  Dim fTmpFileExists As Boolean

  Set wbsav = ActiveWorkbook

  If fTemplateIsForm Then
    Set wbTemplate = wbForm
    GoTo LSuccess
  End If

  Set wbTemplate = Nothing
  stTmpFile = StTmpFilename()
  On Error GoTo LError
  wbForm.SaveCopyAs stTmpFile
  fTmpFileExists = True
  Set wbTemplate = Workbooks.Open(filename:=stTmpFile, updateLinks:=False, ReadOnly:=True)

  If (wbForm Is Nothing) Then GoTo LError:

  wbTemplate.SaveAs filename:=stTemplate, FileFormat:=xlTemplate, CreateBackup:=False

  On Error Resume Next
  Kill stTmpFile
  On Error GoTo 0
  fTmpFileExists = False
LSuccess:
  wbsav.Activate
  FCreateTemplate = True
  Exit Function
LError:
  DisplayError ecUnableToCreateTemplate, stTemplate
  If Not (wbTemplate Is Nothing) Then _
    wbTemplate.Close savechanges:=False, routeworkbook:=False
  If fTmpFileExists Then Kill stTmpFile
  Set wbTemplate = Nothing
  FCreateTemplate = False
End Function
'---------------------------------------------------------------------------
Sub HideSheet(sht As Worksheet)
  sht.Visible = xlVeryHidden
  sht.Cells.ColumnWidth = 0
  sht.Cells.RowHeight = 0
End Sub
Sub UnHideSheet(sht As Worksheet)
  sht.Visible = True
  sht.Cells.ColumnWidth = sht.StandardWidth
  sht.Cells.RowHeight = sht.StandardHeight
End Sub
```

```
Option Explicit
Option Compare Text
Option Private Module

'---------------------------------------------------------------------
'Extern procedures
'---------------------------------------------------------------------
Private Declare Function GetFullPathNameA Lib "KERNEL32.DLL" _
  (ByVal lpszFile As String, ByVal cch As Integer, ByVal lpszPath As String, _
    ByVal ppsz As Integer) As Integer

Private Declare Function GetTempPathA Lib "KERNEL32.DLL" _
  (ByVal cchBuffer As Integer, ByVal lpszTempPath As String) As Integer


'---------------------------------------------------------------------
'This module contains common code and utility procedures that used in
'other modules.
'---------------------------------------------------------------------
Private Const stTmpFileExtension = "tmp"
Private Const stTmpFilePrefix = "ATW"
'code for fcFileIsOpen

Public Const fcNotOpen = 0
Public Const fcSameName = 1
Public Const fcSameFile = 2

Private Const cchPathMax As Integer = 1024
'---------------------------------------------------------------------
'Procedures
'---------------------------------------------------------------------
Public Sub Assert(f As Boolean)
'  If Not f Then
'    MsgBox "Assertion Failed"
'    Stop
'  End If
End Sub

Public Function Min(x As Integer, y As Integer) As Integer
  If x > y Then
    Min = y
  Else
    Min = x
  End If
End Function

Public Function Max(x As Integer, y As Integer) As Integer
  If x > y Then
    Max = x
  Else
    Max = y
  End If
End Function

'Return the filename without the extension
Public Function StripExtension(st As String) As String
  Dim i As Integer

  i = InStr(1, st, ".", 0)
  If i = 0 Then
    StripExtension = st
  Else
    StripExtension = Left$(st, i - 1)
  End If
End Function

Public Function AddExtension(stFile As String, stExt As String, Optional fForce As Variant) As Stri
ng
  Dim i As Integer

  If TestOS = OS_MAC Then
    AddExtension = stFile
    Exit Function
  End If
```

```
  i = InStr(1, stFile, ".", 0)

  If IsMissing(fForce) Then fForce = False
  If i = 0 Then
    AddExtension = stFile & "." & stExt
  Else
    If fForce Then
      AddExtension = Left$(stFile, i) & stExt
    Else
      AddExtension = stFile
    End If
  End If
End Function

'strips the path of the name from it
Public Function StripPath(st As String) As String
  Dim i As Integer
  Dim iPrev As Integer

  i = 0
  Do
    iPrev = i + 1
    i = InStr(iPrev, st, stPathSeparator, 0)
  Loop While i <> 0
  StripPath = Mid$(st, iPrev)
End Function


Function FFileExists(stFile As String) As Boolean
  On Error GoTo LError
    GetAttr stFile
  On Error GoTo 0
  FFileExists = True
  Exit Function
LError:
  FFileExists = False
End Function

'stpath is the absolute path to the file, including an extension if it is there
Function FcFileIsOpen(stPath As String) As Integer
  Dim wb As Workbook
  Dim st As String
  Dim stPathPart As String
  Dim stNamePart As String
  Dim i As Integer
  Dim iPrev As Integer

  i = 0
  Do
    iPrev = i + 1
    i = InStr(iPrev, stPath, stPathSeparator, 0)
  Loop While i <> 0
  stNamePart = Mid$(stPath, iPrev)
  ' Assert iPrev > 1 'since it was an absolute path
  If (iPrev > 1) Then stPathPart = Left$(stPath, iPrev - 2)
  On Error GoTo LError
    Set wb = Workbooks(stNamePart)
  On Error GoTo 0
  If wb.Path = stPathPart Then
    FcFileIsOpen = fcSameFile
  Else
    FcFileIsOpen = fcSameName
  End If
  Exit Function
LError:
  FcFileIsOpen = fcNotOpen
End Function

Public Function StTmpFilename() As String
  Dim stPathBuf As String * cchPathMax
  Dim cchPath As Integer
  Static i As Integer
  Dim stTmpPath As String
  Dim st As String
  Dim j As Integer
  Dim sTmpDir As String
```

```
  'get the path for temp files
  If TestOS = OS_WIN Then
    'bug 730
    On Error Resume Next
        j = 1
        Do
            sTmpDir = PathFile(ActiveWorkbook.Path, "wzkfpb" & j)
            If Not FFileExists(sTmpDir) Then
                MkDir sTmpDir
                If FFileExists(sTmpDir) Then
                    cchPath = 0
                    RmDir sTmpDir
                    Exit Do
                Else
                    cchPath = GetTempPathA(cchPathMax, stPathBuf)
                    Exit Do
                End If
            End If
            j = j + 1
        Loop
    On Error GoTo 0
  Else
     cchPath = 0
  End If

  If cchPath = 0 Or cchPath > cchPathMax Then
    'stTmpPath = CurDir$() bug 730
    stTmpPath = ActiveWorkbook.Path
    If Right$(stTmpPath, 1) <> stPathSeparator Then _
      stTmpPath = stTmpPath & stPathSeparator
  Else
    stTmpPath = Left$(stPathBuf, cchPath)
  End If
  Assert Right$(stTmpPath, 1) = stPathSeparator
  Do
    st = stTmpPath & stTmpFilePrefix & Format$(i, "00000")
    If Not FFileExists(st) Then
    If Not FcFileIsOpen(st) Then
      StTmpFilename = st
      Exit Function
    End If
    End If
    i = i + 1
  Loop Until i = 0
  'if we come here, the user deserves it!
  Assert False
End Function

Function PathFile(sPath As String, sFile As String) As String
    If Len(sPath) = 3 Or Right(sPath, 1) = stPathSeparator Then
        'this returns only directories to the list
        PathFile = sPath & sFile
    Else
        'append a "\" to the list and then get the directories there
        PathFile = sPath & stPathSeparator & sFile
    End If
End Function

Function stPathNameFromWb(wb As Workbook) As String
  Assert Not (wb Is Nothing)
  stPathNameFromWb = wb.Path & stPathSeparator & wb.Name
End Function

'close the given workbook and remove all traces of its existence from the disk!
Sub KillWorkbook(wb As Workbook)
  Dim stFile As String

  stFile = stPathNameFromWb(wb)
  wb.Close savechanges:=False, routeworkbook:=False
  Kill stFile
  Set wb = Nothing
End Sub
'---------------------------------------------------------------------------
'Opens a workbook specified by stFilename. If the book is already open in
'excel, it sets the wbRet to that. If a samename book is open, it returns false
```

```
'and shows appropriate error. fReadOnly defaults to True
'--------------------------------------------------------------------------
Function FOpenWorkbook(stFileName As String, _
                       wbRet As Workbook, fWasAlreadyOpenRet As Boolean, _
                       Optional fReadOnly) As Boolean
  Dim fc As Integer
  Dim wbsav As Workbook

  If IsMissing(fReadOnly) Then fReadOnly = True
  fc = FcFileIsOpen(stFileName)
  Select Case fc
    Case fcNotOpen
      Set wbsav = ActiveWorkbook
      On Error Resume Next
        Set wbRet = Workbooks.Open(filename:=stFileName, ReadOnly:=fReadOnly)
        If Err <> 0 Then
          DisplayError ecUnableToOpenFile, stFileName
          GoTo LFailure
        End If
      On Error GoTo 0
      wbsav.Activate
      fWasAlreadyOpenRet = False
    Case fcSameName
      DisplayError ecSameNameFileOpen, StripPath(stFileName)
      GoTo LFailure
    Case fcSameFile 'the same file is open, use it
      Set wbRet = Workbooks(StripPath(stFileName))
      fWasAlreadyOpenRet = True
    Case Else
      Assert False
  End Select
  FOpenWorkbook = True
  Exit Function
LFailure:
  FOpenWorkbook = False
End Function

'--------------------------------------------------------------------------
'Function to return the complete path given a path to a book
'--------------------------------------------------------------------------
Public Function StFullPathName(stFileName As String) As String
  Dim stPath As String * cchPathMax 'must use fixed len string here
  Dim cchPath As Integer
  Dim iLenDir As Integer

  If TestOS = OS_WIN Then
    cchPath = GetFullPathNameA(stFileName, cchPathMax, stPath, 0)
    If cchPath = 0 Or cchPath > cchPathMax Then
        StFullPathName = stFileName 'try the name directly.
    Else
        StFullPathName = Left$(stPath, cchPath)
    End If
  Else
    iLenDir = Len(CurDir())
    If Left(stFileName, iLenDir) = CurDir() Then
        StFullPathName = stFileName
    Else
        StFullPathName = CurDir() & ":" & stFileName
    End If
  End If
End Function

'--------------------------------------------------------------------------
```