modGridIn - 1

```
Option Explicit

Sub getRefDoc(ByVal wp As String)

'\\  save reference to current grid gDoc
'\\  set loaded status of gDoc

   If MacroRefDoc = Zero Then

      '\\            MacroRefDoc = lDocCount + one

      MacroRefDoc = LoadNewDoc("untitled")

      '\\   Reset gDoc to Current Grid
      '\\            gDoc = CurDoc
      '\\            gCancel = False
      With fDoc(MacroRefDoc)
         .Caption = App.path & "\" & App.Title & strLocalID & "\" & Trim$(wp) & ".ing"
         .DocText_Filename = vbNullString     '\\   to signal an invisible system grid
         FileLoad MacroRefDoc, .Caption
         Select Case wp
         Case "TrainingCardDoc"
            If gCancel Then
               '\\   if no Reference preparation exits in LocalID
               '\\   then guide user to build a multigrid directory
               MsgBoxEx "MultiGrid preparation code should be run.", _
                       vbOKOnly + vbInformation, "WM_TCARD Setup"
               '\\   clean up

               UnloadDocument (MacroRefDoc)
               TCardRow = -One    '\\   do nothing
               Exit Sub
            Else
               GetGridTable MacroRefDoc, .Table
               .Table.col = Two
               .Table.RowSel = One
               .Table.ColSel = 73


               If .Table.Clip <> ToolBoxHeads Then
                  Select Case MsgBoxEx("ToolBoxheadings do not match the current version. Up
date?", _
                                       vbYesNo + vbInformation, "WM_TCard Macro")
                  Case vbYes
                     .Table.Clip = ToolBoxHeads
                     SetDirty Two, MacroRefDoc
                  Case vbNo
```

```
                UnloadDocument (MacroRefDoc)
                TCardRow = -One    '\\   do nothing
                Exit Sub
            End Select
        End If

        Dim J As Long
        '\\   check for WM_TCard construct
        If TCardRow = Zero Then
            For J = Two To .Table.Rows - Three
                If .Table.TextMatrix(J, One) = "WM_TCard" Then
                    TCardRow = J
                    Exit For
                End If
            Next
            For J = Two To .Table.Rows - Three
                If .Table.TextMatrix(J, One) = "MacroRow" Then
                    MacroRow = J
                    Exit For
                End If
            Next
        End If

    End If
Case "AtomixMP3Doc"

    If gCancel Then
        MsgBoxEx "fDoc.table unsaveable"
        Stop    ' testing needed to develop further
        savegr
        '\\                          gCancel = False
        FileCopy FilenameEx(fDoc(gDoc).DocText_Filename), FilenameEx(.Caption)
        FileLoad MacroRefDoc, .Caption
        If gCancel Then
            .Table.Rows = GridIn.Table.Rows
            .Table.Cols = GridIn.Table.Cols
            .Table.col = Zero
            .Table.row = Zero
            .Table.RowSel = .Table.Rows - One
            .Table.ColSel = .Table.Cols - One
            GridIn.Table.col = Zero
            GridIn.Table.row = Zero
            GridIn.Table.RowSel = GridIn.Table.Rows - One
            GridIn.Table.ColSel = GridIn.Table.Cols - One
            .Table.Clip = GridIn.Table.Clip
            .Table.col = One    '\\   to deselect
            .Table.row = One
```

```
                Else
                    GetGridTable MacroRefDoc, .Table
                End If
            Else
                GetGridTable MacroRefDoc, .Table

            End If
        Case Else
            MsgBoxEx wp & " error - not programmed yet"
        End Select
        '\\   lastly for saving as a System Grid must getGridTable into invisible cache
        .Visible = False
    End With
  End If
End Sub
Function TableClip(ByRef Table As MSFlexGrid) As String
  On Error GoTo errline
  Dim sTempFile As String
  sTempFile = GetTempFile()
  Dim iFileNum As Integer, i As Long, J As Long, sBuffer As String
  iFileNum = FreeFile
'   Stop
  Open sTempFile For Output As iFileNum
  With Table
    For i = .row To .RowSel
      For J = .col To .ColSel - One
        Print #iFileNum, .TextMatrix(i, J); vbTab;
      Next
      Print #iFileNum, .TextMatrix(i, J); vbCr;
    Next
    Print #iFileNum, vbLf;
  End With

  'Read contents of file
  Close #iFileNum
  Open sTempFile For Input As iFileNum
  TableClip = input$(LOF(iFileNum), iFileNum)
errline:
'   Stop
  Close #iFileNum
  Kill sTempFile
End Function

Sub transpose()
'\\  Transpose Grid
  On Error GoTo errorline
  Dim RowItem As Long, ColItem As Long, workz6 As String
```

```
   UnResolve
   grname FullGridName
   If Section = Three And StrComp(NormalCode, "Y", vbTextCompare) = Zero Then
      CellRevert
   End If

   Open FullGridName For Output As #One
   If inGridVER(Zero, gDoc) < 10.096 Then
      Transpose81411
   Else

      Print #One, "Version"
      Print #One, 10.096

      If Len(JPEGs(gDoc).toImgSrc.Text) <> Zero Then
         Print #One, GridTitle;
         Print #One, " <TITLE="""" & JPEGs(gDoc).toPageTitle & Quotes & TAGHref & Quotes &
 JPEGs(gDoc).toHref & Quotes & TAGImgSrc & Quotes & JPEGs(gDoc).toImgSrc & Quotes & AS
pace & TAGUseMap & Quotes & JPEGs(gDoc).toUseMap & """>"
      Else
         Print #One, GridTitle
      End If
      Print #One, inGridUser(gDoc)    '\\  ;
      Print #One, B9; vbTab; T9
      Print #One, GridDate
      Print #One, ViewType
      Print #One, NormalCode
      Print #One, WeightCol
      If StrComp(ElementTitle, "ABBREVIATED ELEMENTS", One) = Zero Then ElementTitle "El
ements"
      If StrComp(ConstructTitle, "ABBREVIATED CONSTRUCTS", One) = Zero Then ConstructTit
le "Constructs"
      Print #One, str(NmC + ConCols * Cent); ASpace; vbTab; ConstructTitle
      Print #One, str(NmE + CelCols * Cent); ASpace; vbTab; ElementTitle

      workz6 = GridIn.Table.TextMatrix(One, LastDataCol + Two)
      GridIn.Table.TextMatrix(One, LastDataCol + Two) = getPictureList
      putPictureList workz6



      For ColItem = One To LastDataCol + Three
         workz6 = GridIn.Table.TextMatrix(One, ColItem)
         Do While InStr(workz6, vbLf)
            Mid$(workz6, InStr(workz6, vbLf), One) = ASpace
         Loop
         Print #One, workz6;
         For RowItem = Two To NmC + Three
```

```
            workz6 = GridIn.Table.TextMatrix(RowItem, ColItem)
            Do While InStr(workz6, vbCr)
               Mid$(workz6, InStr(workz6, vbCr), One) = ASpace
            Loop
            Do While InStr(workz6, vbLf)
               '\\  for ans=1to len(workz6):print hex(mid$(workz6,ans,1));:next
               Mid$(workz6, InStr(workz6, vbLf), One) = ASpace
            Loop
            '\\  do while InStr(one, workz6, "world_wide")
            '\\      mid$(workz6, InStr(workz6, vbTab), one) = ASpace
            '\\  loop
            Print #One, vbTab; workz6;
         Next
         Print #One,
      Next
      Print #One, Zero
   End If
   Close #One

   If Section > Nine Then
      grload1
      analy0
      If Not gCancel Then
         Sect02
         Sect03
         Sect04
         Sect05
         savegr
         ReEnable
      End If
   End If
   grload1
   sect11
   If gCancel Then
      Exit Sub
   End If
   setAboutCaption "inGridX - transposed " & FullGridName
   Exit Sub
   Resume
errorline:       ' stop
   WarningError Err, "Transpose"

End Sub

Function CopyToClip() As Boolean
'Stop
   CopyToClip = True
```

```
   If GridIn.Table.RowSel = GridIn.Table.row And GridIn.Table.ColSel = GridIn.Table.col Then
      GridIn.Table.col = One
      GridIn.Table.row = One
      GridIn.Table.ColSel = GridIn.Table.Cols - One
      GridIn.Table.RowSel = GridIn.Table.Rows - One
   End If
   If LenB(GridIn.Table.Clip) = Zero Then
      If MsgBoxEx(Msg219_Search_whole_Grid, vbOKCancel) = vbCancel Then
         CopyToClip = False
         Exit Function
      End If
      GridIn.Table.col = One
      GridIn.Table.row = One
      GridIn.Table.ColSel = GridIn.Table.Cols - One
      GridIn.Table.RowSel = GridIn.Table.Rows - One
   End If
      ClipBoardClear
      Clipboard.SetText Trim(GridIn.Table.Clip)
'    ClipBoardClear
'    ClipBoardSetText (GridIn.Table.Clip), False
End Function




Sub GridInClick_cmdReplace()
   UnResolve
   Dim workz7 As String, workz6 As String
   If Len(GridIn.Table.Clip) <> Zero And (GridIn.Table.RowSel = GridIn.Table.row And GridIn.
Table.ColSel = GridIn.Table.col) Then
      workz7 = InputBox(IBS510_ReplaceWhat, IBS511_EntireTableReplacement, GridIn.Table.
Clip)
   Else
      workz7 = InputBox(IBS510_ReplaceWhat, IBS513_Search_within_Selection, GridIn.TextE
ntry.SelText)
   End If
   If Len(workz7) = Zero Then Exit Sub
   workz6 = InputBox(IBS514_ReplaceWith)
   '\\  If Len(workz6) = ZERO Then Exit Sub
   If CopyToClip Then PasteArea workz7, workz6

End Sub

Sub GridInClick_ColSort()
   With GridIn.Table
      inGrid.mnuSortCustomA.Enabled = False
      inGrid.mnuSortCustomD.Enabled = False
```

modGridIn - 7

```
    If .col = .ColSel And .row < Two Then
        If .ColSel < Two Then
            inGrid.mnuSortNone.Caption = "Emerge entire grid"    '\\  Cap624_Label_sort
        ElseIf .ColSel > NmE + One Then
            If WeightCol = "Y" And .ColSel = NmE + Two Then
                inGrid.mnuSortNone.Caption = Cap625_Weight_sort
            ElseIf .ColSel = LastDataCol + Two Then
                inGrid.mnuSortNone.Caption = Cap626_Construct_Description_sort
            Else
                inGrid.mnuSortNone.Caption = Cap627_Construct_extraHTML_sort
            End If
        Else
            inGrid.mnuSortNone.Caption = Cap628_Element & .TextMatrix(Zero, .col)
            inGrid.mnuSortCustomA.Enabled = True
            inGrid.mnuSortCustomD.Enabled = True
            inGrid.mnuSortCustomA.Caption = Cap629_AscendingForeColor & str(.col - One)
            inGrid.mnuSortCustomD.Caption = Cap630_DescendingForeColor & str(.col - One)
        End If
        inGrid.mnuSortGenericAscending.Visible = True
        inGrid.mnuSortGenericDescending.Visible = True
        inGrid.mnuSortNoCaseDescending.Visible = True
        inGrid.mnuSortNone.Visible = True
        inGrid.mnuSortNumericAscending.Visible = True
        inGrid.mnuSortNumericDescending.Visible = True
        inGrid.mnuSortStringAscending.Visible = True
        inGrid.mnuSortStringDescending.Visible = True
        inGrid.mnuSortStringNoCaseAsending.Visible = True

        inGrid.mnuSortColumnAscending.Visible = False
        inGrid.mnuSortColumnDescending.Visible = False

    ElseIf .row = .RowSel Then

        If .RowSel < Two Then

            inGrid.mnuSortNone.Caption = "Emerge entire grid"    '\\  Cap624_Label_sort Cap6
31_Element_Label_sort
        ElseIf .RowSel > NmC + One Then
            If .RowSel = NmC + Two Then
                inGrid.mnuSortNone.Caption = Cap632_Element_Description_sort
            Else
                inGrid.mnuSortNone.Caption = Cap633_Element_extraHTML_sort
            End If
        Else
            inGrid.mnuSortNone.Caption = Cap634_Construct_sort & .TextMatrix(.row, Zero)
            inGrid.mnuSortCustomA.Enabled = True
            inGrid.mnuSortCustomA.Caption = Cap635_AscendingBackColor & str(.row - One)
```

```
            inGrid.mnuSortCustomD.Enabled = True
            inGrid.mnuSortCustomD.Caption = Cap636_DescendingBackColor & str(.row - One)

        End If
        inGrid.mnuSortColumnAscending.Visible = True
        inGrid.mnuSortColumnDescending.Visible = True
        inGrid.mnuSortNone.Visible = True

        inGrid.mnuSortGenericAscending.Visible = False
        inGrid.mnuSortGenericDescending.Visible = False
        inGrid.mnuSortNoCaseDescending.Visible = False
        inGrid.mnuSortNumericAscending.Visible = False
        inGrid.mnuSortNumericDescending.Visible = False
        inGrid.mnuSortStringAscending.Visible = False
        inGrid.mnuSortStringDescending.Visible = False
        inGrid.mnuSortStringNoCaseAsending.Visible = False

    End If
  End With
  inGrid.PopupMenu inGrid.mnuSort
  '\\  GridSort
End Sub
Sub inGridClick_mnuEditCut()
  Dim i As Long, J As Long
  If GridIn.Visible = True And Len(GridIn.Table.Clip) <> Zero Then
    SetDirty True
'     ClipBoardClear
    ClipBoardSetText GridIn.Table.Clip, False
    If GridIn.Table.row <= One And GridIn.Table.RowSel = GridIn.Table.Rows - One And (Gr
idIn.Table.col > One Or GridIn.Table.ColSel < GridIn.Table.Cols - One) Then
        UnloadAnyLayers
        UnloadPopupMenuItems
        For J = GridIn.Table.col To GridIn.Table.ColSel
            DeleteCol
        Next
        grent1
    ElseIf GridIn.Table.col <= One And GridIn.Table.ColSel = GridIn.Table.Cols - One And (G
ridIn.Table.row > One Or GridIn.Table.RowSel < GridIn.Table.Rows - One) Then
        UnloadAnyLayers
        UnloadPopupMenuItems
        For i = GridIn.Table.row To GridIn.Table.RowSel
            DeleteRow
        Next
        grent1
    Else
        For i = GridIn.Table.row To GridIn.Table.RowSel
            For J = GridIn.Table.col To GridIn.Table.ColSel
```

```
                GridIn.Table.TextMatrix(i, J) = vbNullString
            Next
        Next
    End If
  Else

    If GridOut.FileText.Visible = True Then
        GridOut.FileText.SelText = vbNullString
        Exit Sub
    Else
        '\\             GridOut.FileText.Visible = True
        If GridOut.FileText.Visible = True Then
            If GridOut.Picture5.Picture <> Zero Then
                '\\   delete the current picture from the pictureList
                deleteFromPictureList
                Exit Sub
            End If
        ElseIf ReadDoc > Zero Then
            CutOccurances
        Else
            MsgBoxEx Msg216_Cut_for_Grid_table
        End If
    End If
  End If
End Sub


Sub GridInClose()
  Dim lWork As Long
  If SaveDirty(gDoc, Four) Then
    StoreGrid
    '\\  grloada
  End If
  If Analysed And dirty Then
    AlterSpace
    UnResolve
    GridOut.Hide

    analy0
    If Not gCancel Then
      Sect02
      Sect03
      Sect04
      Sect05
    End If
    GridIn.Hide
```

```
modGridIn - 10

   ElseIf Not Analysed And dirty Then
      AlterSpace
      Exit Sub
   Else
      GridIn.Hide
   End If
   Section = Ten
   lWork = ShowMe(GridOut)
   Front
   '\\  UserParams.Hide
   '\\  SetDirty False
   sect11
   ReEnable
End Sub


Sub WeightChange()
   With GridIn
      Dim RowItem As Long    '\\  , ColItem As Long
      .Table.Redraw = False
      If .Weight.value = vbChecked Then
         WeightCol = "Y"
         LastDataCol = NH(-NmE)
         .Table.Cols = LastDataCol + Four
         For RowItem = Zero To NmC + Three
            .Table.TextMatrix(RowItem, LastDataCol + Three) = .Table.TextMatrix(RowItem, La
stDataCol + Two)
            .Table.TextMatrix(RowItem, LastDataCol + Two) = .Table.TextMatrix(RowItem, Last
DataCol + One)
            If RowItem > Zero And RowItem < NmC + Two Then
               .Table.TextMatrix(RowItem, LastDataCol + One) = OneText
            End If
         Next
         .Table.TextMatrix(One, LastDataCol + One) = vbNullString
         .Table.TextMatrix(NmC + Two, LastDataCol + One) = vbNullString
         .Table.TextMatrix(NmC + Three, LastDataCol + One) = vbNullString
      Else
         For RowItem = Zero To NmC + Three
            .Table.TextMatrix(RowItem, LastDataCol + One) = .Table.TextMatrix(RowItem, Last
DataCol + Two)
            .Table.TextMatrix(RowItem, LastDataCol + Two) = .Table.TextMatrix(RowItem, Last
DataCol + Three)
            .Table.TextMatrix(RowItem, LastDataCol + Three) = vbNullString
         Next
         WeightCol = "N"
         LastDataCol = NH(-NmE)
         .Table.Cols = LastDataCol + Four
```

```
        End If
        inGridVER(Zero, gDoc) = 10.092
        here = LastDataCol + Two
        SetDirty True
    End With
End Sub



Sub EditCopy()

    If fDoc(gDoc).WindowState <> vbMinimized Then
'        ClipBoardClear
        ClipBoardSetText fDoc(gDoc).DocText.SelText, False
        Exit Sub
    ElseIf GridIn.Visible = True And Len(GridIn.Table.Clip) <> Zero Then
'        ClipBoardClear
        ClipBoardSetText GridIn.Table.Clip, False
    Else
        CopyGrid
    End If

End Sub
Sub EditPaste()

    If fDoc(gDoc).Visible And dirty Then Exit Sub
    PasteArea NotString, NotString
End Sub
Sub WeightCrossMultiply()
'\\  this routine takes amounts stored in the row descriptions and
'\\  spreads them across the columns according to the weighted components
'\\  and then sums the columns putting the values into the column labels

    Dim RowVal As Double, ComPercent As Double, workz6 As String, workz7 As String, DisburS
um As Double, i9 As Long, j9 As Long, k9 As Long
    For j9 = One To NmE
        eLoad(Zero, j9) = Zero
    Next



    For i9 = One To NmC
        RowVal = Val(getDescription(NmE + i9))
        DisburSum = Zero
        If WeightCol = "Y" Then
            workz6 = GridIn.Table.TextMatrix(i9 + One, NmE + Two)
            workz6 = Mid$(workz6, One + InStr(workz6, "/"))
            If InStr(workz6, "/") > Zero Then
```

```
            workz6 = Mid$(workz6, One + InStr(workz6, "/"))
            DisburSum = Val(workz6)
            If DisburSum < Zero Then
               '\\  Third weight parameter is used to specify fixed base
               '\\  as amount if -ve or as % if +ve
               DisburSum = -DisburSum
            Else
               DisburSum = RowVal * DisburSum * Cent
            End If
         End If
      End If

      For j9 = One To NmE
         eLoad(Zero, j9) = eLoad(Zero, j9) + DisburSum / NmE
      Next
      DisburSum = RowVal - DisburSum
      If ConDev(i9) = Zero Then
         For j9 = One To NmE
            eLoad(Zero, j9) = eLoad(Zero, j9) + (DisburSum) / NmE
         Next
      Else
         For k9 = One To N7
            ComPercent = (cLoad(i9, k9) * ConDev(i9)) ^ Two
            For j9 = One To NmE
               eLoad(Zero, j9) = eLoad(Zero, j9) + DisburSum / NmE * ComPercent * (One + S
gn(cLoad(i9, k9)) * eLoad(k9, j9))
            Next
         Next
      End If
   Next
   DisburSum = Zero
   For j9 = One To NmE
      workz7 = Trim$(GridIn.Table.TextMatrix(eLabelRow, j9 + One))
      workz6 = Trim$(str(Val(workz7)))
      If InStr(One, workz7, workz6) = One Then
         workz7 = Trim$(Mid$(workz7, Len(workz6) + One))
         If Left$(workz7, One) = "-" Then workz7 = Trim$(Mid$(workz7, Two))
      End If
      DisburSum = DisburSum + eLoad(Zero, j9)
      GridIn.Table.TextMatrix(eLabelRow, j9 + One) = str(Int(eLoad(Zero, j9) * Hundred + Hal
f) * Cent) & "-" & workz7
   Next
   GridIn.Table.TextMatrix(eLabelRow, NmE + Two) = str(Int(DisburSum * Hundred + Half) *
Cent)

   grent1
End Sub
```

```
Function conLabel(ByRef RowItem As Long, Optional CaseText As String = "Any") As String

   If CaseText = "Any" Then CaseText = inGrid.SetConstructs.list(inGrid.ConSlider.value)

   Select Case LCase$(CaseText)

   Case "full"
      conLabel = getDescription(RowItem, ZeroText)
   Case "split"
      If RowItem > Zero Then
         conLabel = Trim$(getDescription(RowItem, ZeroText))
         If InStr(conLabel, "/") Then
            conLabel = Trim$(Left$(conLabel, InStr(conLabel, "/") - 1))
         Else
            conLabel = Trim$(conLabel & str(RowItem))
         End If
      Else
         conLabel = getDescription((-RowItem), ZeroText)
         If InStr(conLabel, "/") Then
            conLabel = Trim$(Right$(conLabel, Len(conLabel) - InStr(conLabel, "/")))
         Else
            conLabel = str(RowItem)
            Mid$(conLabel, One) = "c"
         End If
      End If
   Case "left"
      conLabel = Trim$(getDescription(Abs(RowItem), ZeroText))
      conLabel = Left$(conLabel, InStr(conLabel & "/", "/") - One)
   Case "right"
      conLabel = Trim$(getDescription(RowItem, ZeroText))
      conLabel = Right$(conLabel, Len(conLabel) - InStr(conLabel, "/"))
   Case "label"
      conLabel = Trim$(GridIn.Table.TextMatrix(One + RowItem, cLabelCol))
   Case "nmbr"
      conLabel = str(RowItem)
      If RowItem > Zero Then
         Mid$(conLabel, One) = "C"
      Else
         Mid$(conLabel, One) = "c"
      End If
   Case "cell"
      conLabel = str(RowItem)
      If RowItem > Zero Then
         Mid$(conLabel, One) = "C"
      Else
```

```
            Mid$(conLabel, One) = "c"
        End If
    Case "off"
        conLabel = vbNullString
    End Select
End Function

Function eLabel(ByRef ColItem As Long, Optional CaseText As String = "Any") As String

    If CaseText = "Any" Then CaseText = inGrid.SetElements.list(inGrid.ElSlider.value)
    Dim pos As Long, ret As Long
    Select Case LCase$(CaseText)
    Case "speak"
        eLabel = fDoc(ReadDoc).Table.TextMatrix(eLabelRow, One + ColItem)
    Case "full"
        eLabel = getDescription(ColItem)
        If Len(eLabel) = Zero Then eLabel = Trim$(GridIn.Table.TextMatrix(eLabelRow, One + C
olItem))

    Case "left"
        eLabel = getDescription(ColItem)
        eLabel = Trim$(Left$(eLabel, InStr(eLabel & "/", "/") - One))
    Case "right"
        pos = Zero
        ret = Zero
        eLabel = getDescription(ColItem)
        Do
            pos = InStr(pos + One, eLabel, "/")
            If pos Then ret = pos
        Loop While pos
        eLabel = Trim$(Right$(eLabel, Len(eLabel) - ret))
    Case "label"
        eLabel = GridIn.Table.TextMatrix(eLabelRow, One + ColItem)
    Case "nmbr", "cell"
        eLabel = Trim$(str(ColItem))
    Case "off"
        eLabel = vbNullString
    End Select

End Function



Function nC(xaxis, yaxis, zaxis, ConstructNo) As Double
    On Error GoTo errorline
    nC = Abs(ncsq1(Int(PointNine + Abs(xaxis)), Int(PointNine + Abs(yaxis)), Int(PointNine + A
bs(zaxis)), ConstructNo) * ConDev(ConstructNo) ^ Two)
    Exit Function
```

```
modGridIn - 15

errorline:        ' stop
   nC = Zero

End Function

Function ncsq1(xaxis, yaxis, zaxis, ConstructNo) As Double
   Dim ans As Double


   If xaxis <> Zero Then
      ans = Int(PointNine + Abs(xaxis))
      ncsq1 = cLoad(ConstructNo, ans) ^ Two
   End If
   If yaxis <> Zero Then
      ans = Int(PointNine + Abs(yaxis))
      ncsq1 = ncsq1 + cLoad(ConstructNo, ans) ^ Two
   End If
   If zaxis <> Zero Then
      ans = Int(PointNine + Abs(zaxis))
      ncsq1 = ncsq1 + cLoad(ConstructNo, ans) ^ Two
   End If
End Function

Function ne(xaxis, yaxis, zaxis, el) As Double
   ne = nesq1(Int(PointNine + Abs(xaxis)), Int(PointNine + Abs(yaxis)), Int(PointNine + Abs(za
xis)), el) / ElVar(el)
End Function

Function nesq1(xaxis, yaxis, zaxis, el) As Double
   Dim ans As Double


   If xaxis <> Zero Then
      ans = Int(PointNine + Abs(xaxis))
      '\\  If eLoad(Ans, el) = zero Then
      '\\      eLoad(Ans, el) = Val(fTableLeft.flex01.TextMatrix(Ans, el))
      '\\  End If
      nesq1 = eLoad(ans, el) ^ Two
   End If
   If yaxis <> Zero Then
      ans = Int(PointNine + Abs(yaxis))
      '\\  If eLoad(Ans, el) = zero Then
      '\\      eLoad(Ans, el) = Val(fTableLeft.flex01.TextMatrix(Ans, el))
      '\\  End If
      nesq1 = nesq1 + eLoad(ans, el) ^ Two
   End If
   If zaxis <> Zero Then
```

```
      ans = Int(PointNine + Abs(zaxis))
      '\\  If eLoad(Ans, el) = zero Then
      '\\      eLoad(Ans, el) = Val(fTableLeft.flex01.TextMatrix(Ans, el))
      '\\  End If
      nesq1 = nesq1 + eLoad(ans, el) ^ Two
   End If

End Function


Function NH(ByVal NmE As Long, Optional ByVal Weighted As String = vbNullString) As Long
   If Len(Weighted) = Zero Then
      Weighted = WeightCol
   End If
   NH = Abs(NmE) + InStr(One, "Y", Weighted, vbTextCompare)

End Function

Function dMax(ByRef x As Double, ByRef y As Double) As Double
   If x > y Then dMax = x Else dMax = y
End Function
Function sMax(ByRef x As Single, ByRef y As Single) As Single
   If x > y Then sMax = x Else sMax = y
End Function
Function lMax(ByRef x As Long, ByRef y As Long) As Long
   If x > y Then lMax = x Else lMax = y
End Function
Function dMin(ByRef x As Double, ByRef y As Double) As Double
   If x > y Then dMin = y Else dMin = x
End Function
Function sMin(ByRef x As Single, ByRef y As Single) As Single
   If x > y Then sMin = y Else sMin = x
End Function
Function lMin(ByRef x As Long, ByRef y As Long) As Long
   If x > y Then lMin = y Else lMin = x
End Function

Sub LayersEload()
   Dim workz7 As String

   On Error GoTo errorline
   If InStr(One, inGrid.mnuLayersEload.Caption, "Load Picture", vbTextCompare) > Zero Then



      With fCommonDialog
         '\\  To Do
```

```
        '\\  set the flags and attributes of the
        .flags = OFN_FILEMUSTEXIST
        workz7 = getExtraHTML(LayeredItem)
        .FileName = vbNullString

        If Len(workz7) <> Zero And InStr(One, workz7, "coords", vbTextCompare) = Zero Th
en
            .FileName = ExtractHTML(workz7, TAGImgSrc)
            '\\  If Ans > zero Then
            '\\      workz7 = mid$(workz7, Ans + nine)
            '\\      workz7 = left$(workz7, InStr(workz7, Quotes) - ONE)
            '\\      .filename = workz7
            '\\  End If
        End If
        '\\  common dialog control
        .Filter = STR105_Image_Files & " |*.gif;*.jpg;*.bmp;*.ico"
        .ShowOpen

        If Len(.FileName) = Zero Then
            Exit Sub
        End If
        workz7 = ReplaceHTML(workz7, TAGImgSrc, .FileName)
        If InStr(.FileName, ".gif") Then
            If MsgBoxEx(Msg204_load_as_animation, vbYesNo) = vbYes Then
                Set GridOut.Gif89a1.Container = GridOut.Layers(LayeredItem)
                GridOut.Gif89a1.FileName = .FileName
                GridOut.Gif89a1.Visible = True
                GridOut.Gif89a1.Play
                GridOut.Gif89a1.Top = 0.25
                GridOut.Gif89a1.Left = Half
                GridOut.Layers(LayeredItem).Height = GridOut.Gif89a1.Height + Half
                GridOut.Layers(LayeredItem).Width = GridOut.Gif89a1.Width + One
            End If
        Else
            Set GridOut.Layers(LayeredItem).Picture = Nothing
            Set GridOut.Layers(LayeredItem).Picture = LoadPicture(.FileName)
        End If
        If LayeredItem > NmE Then
            putExtraHTML LayeredItem, workz7
        Else
            putExtraHTML LayeredItem, workz7
            GridIn.Table.row = Zero
            GridIn.Table.col = LayeredItem + One
            Set GridIn.Table.CellPicture = GridOut.Layers(LayeredItem).Picture
            GridIn.Table.RowHeight(Zero) = lMax(GridIn.Table.RowHeight(Zero), GridIn.Table.
CellPicture.Height)
            GridIn.Table.ColWidth(LayeredItem + One) = lMax(GridIn.Table.ColWidth(LayeredIt
```

```
modGridIn - 18

em + One), GridIn.Table.CellPicture.Width)
            End If
            '\\  CellRow = one
            '\\  CellCol = 28
        End With
        '\\  To Do
        '\\  process the opened file

    End If
    Exit Sub
errorline:        ' stop
    '\\  ans = Err
    If Err <> 340 Then
        WarningError Err, "LayersEload"
        fCommonDialog.FileName = vbNullString

        Exit Sub
    End If
    LoadGridOutLayers LayeredItem, Three
    Resume

End Sub


Sub TransposeAs()
    FullGridName = FileNameAs(STR188_Save_As_Transposed, FullGridName, "inGrid Files (*.in
g)|*.ing")
    If gCancel Then Exit Sub
    UnResolve
    UnloadPopupMenuItems
    fCommonDialog.InitDir = CurDir
    AddRecent Five
    setAboutCaption "inGridX - Transposed " & FullGridName

    transpose
End Sub

Sub grinp2()
'\\  load grid part2
    With GridIn
        Dim RowItem As Single, ColItem As Single

        Dim worka As String
        '\\  On Error goto ErrorLine
        .Table.col = One
        .Table.row = NmC + Two
        .Table.Text = StripInputEx
```

```
    If InStr(getPictureList, Chr$(31)) Then
       inGrid.NextImage.Enabled = True
    Else
       inGrid.NextImage.Enabled = False
    End If
    If inGridVER(Zero, gDoc) <> 8.13 Then
       workbuffer = StripInputEx
    Else
       inGrid.FontNames = One
    End If

    workbuffer = workbuffer & ASpace
    If StrComp(ElementTitle, "ABBREVIATED ELEMENTS", One) = Zero Or Len(.Table.TextMa
trix(eLabelRow, One + NmE)) = Zero Then
       ColItem = Zero
       Do While Len(workbuffer) <> Zero
          Do While Left$(workbuffer, One) = ASpace
             workbuffer = Mid$(workbuffer, Two)
          Loop
          If Len(workbuffer) <> Zero Then

             ColItem = ColItem + One
             .Table.TextMatrix(eLabelRow, One + ColItem) = Left$(workbuffer, InStr(workbuf
fer, ASpace) - One)
             putDescription ColItem, Left$(workbuffer, InStr(workbuffer, ASpace) - One)
             workbuffer = Mid$(workbuffer, InStr(workbuffer, ASpace))
          End If
       Loop
    End If

    .Table.col = LastDataCol + Two
    .Table.row = NmC + Two

    .Table.Text = StripInputEx
    '\\      Dim StartSel As Long, LengthSel As Long
    '\\      StartSel = fDoc(gDoc).DocText_SelStart
    '\\      LengthSel = fDoc(gDoc).DocText.SelLength


    PicClipLoad
    If inGridVER(Zero, gDoc) >= 10.095 Then
       .Table.col = Zero
       .Table.row = Two
       .Table.ColSel = LastDataCol + Three
       .Table.RowSel = NmC + Three
       .Table.Clip = Mid$(fDoc(gDoc).DocText.Text, StartLine)
       '\\          fDoc(gDoc).DocText_SelStart = StartLine - One
```

```
        '\\         fDoc(gDoc).DocText.SelLength = fDoc(gDoc).DocText.Find(vbCr, StartLine) -
StartLine
        '\\         .Table.Clip = fDoc(gDoc).DocText.SelText

        StartLine = StartLine + Len(.Table.Clip) + 2
        If inGridVER(Zero, gDoc) = 10.095 Then
            For RowItem = One To NmC
                worka = .Table.TextMatrix(RowItem + One, cLabelCol)
                .Table.TextMatrix(RowItem + One, cLabelCol) = Trim$(worka)
            Next
        End If

    Else
        For RowItem = One To NmC
            ColItem = Zero
            workbuffer = vbNullString
            If RowItem <= NmC Then workbuffer = StripInputEx



            workbuffer = workbuffer & ASpace
            Do
                Dim i9 As Long
                i9 = Zero
                Do While Mid$(workbuffer, i9 + One, One) = ASpace And Len(Mid$(workbuffer, i
9 + One, One)) <> Zero
                    i9 = i9 + One
                Loop
                If Len(Mid$(workbuffer, i9 + One, One)) = Zero Then Exit Do
                workbuffer = Right$(workbuffer, Len(workbuffer) - i9)
                If ColItem > LastDataCol Then
                    If Left$(workbuffer, One) = Quotes Then
                        workbuffer = Mid$(workbuffer, Two)
                        workbuffer = Left$(workbuffer, InStr(workbuffer, Quotes) - One)
                    Else
                        workbuffer = Trim$(workbuffer)
                    End If
                    If Len(.Table.TextMatrix(One + RowItem, LastDataCol + Two)) = Zero Then



                        .Table.col = LastDataCol + Two
                        .Table.ColSel = LastDataCol + Three
                        .Table.row = RowItem + One
                        .Table.RowSel = RowItem + One
                        .Table.Clip = Trim$(workbuffer)
                    ElseIf LCase$(Right(workbuffer, 4)) = "<br>" Then
                        Mid$(workbuffer, Len(workbuffer) - Three, Four) = "    "
                        workbuffer = Trim$(workbuffer)
```

```
                End If


                Exit Do
            End If
            worka = Left$(workbuffer, InStr(workbuffer, ASpace) - One)
            If ColItem = Zero Then
                If .Table.TextMatrix(One + RowItem, cLabelCol) = "      " Or .Table.TextMatrix
(One + RowItem, cLabelCol) = ASpace Or Len(.Table.TextMatrix(One + RowItem, cLabelCol))
= Zero Then
                    .Table.TextMatrix(One + RowItem, cLabelCol) = worka
                    .Table.col = One
                    .Table.row = RowItem + One
                    .Table.Text = worka



                End If
            ElseIf ColItem <> NmE And ColItem = LastDataCol Then
                .Table.col = LastDataCol + One
                .Table.row = RowItem + One
                .Table.Text = worka


            Else
                .Table.col = ColItem + One
                .Table.row = RowItem + One
                .Table.Text = worka

            End If
            ColItem = ColItem + One
            workbuffer = Right$(workbuffer, Len(workbuffer) - Len(worka))

        Loop

      Next
    End If
  End With
  Exit Sub
errorline:      ' stop
  Close #One
  gCancel = True
  inGrid.Visible = True

  WarningError Err, "inGridIO"

End Sub
```

modGridIn - 22

```
Sub GetEigenValues()

   On Error GoTo errorline

   '\\  ColItem = InStr(WorkBuffer, "EigenValues =")
   Dim ComponentValues() As String, RowItem As Long
   n6 = SplitX(ExtractXML(gDoc, "<EigenValues>", True), "=", ComponentValues()) - One
   RedimEigenValues n6
   N7 = Zero
   For RowItem = One To n6 - One
      EigenValues(RowItem) = Val(ComponentValues(RowItem))
      If EigenValues(RowItem) > EigenValues(RowItem - One) Then N7 = RowItem
   Next
   Exit Sub
   Resume
errorline:        ' stop
   WarningError Err, "GetEigenValues"
End Sub


Public Sub MultiGrid()
   On Error Resume Next
   Dim workz7 As String

   UnResolve



   With fCommonDialog
      '\\  To Do
      '\\  set the flags and attributes of the
      .DialogTitle = STR191_Consensus_Series
      .flags = OFN_EXPLORER Or OFN_LONGNAMES Or OFN_ALLOWMULTISELECT
      '\\  common dialog control
      .Filter = STR192_inGrid_and_Text & " (*.ing)|*.ing"
      .MaxFileSize = 4000
      .FileName = vbNullString
      .ShowOpen
      If Len(.FileName) = Zero Then


         Exit Sub
      End If
      workz7 = .FileName


   End With
```

```
   Do While InStr(workz7, vbNullChar)
      Mid$(workz7, InStr(workz7, vbNullChar)) = Quotes
   Loop
   '\\  To Do
   '\\  process the opened file
   consen workz7
   Resume
End Sub


Function StripInputEx() As String

   On Error GoTo errorline


   StripInputEx = StripInput(Zero)
   Do While Left$(StripInputEx, One) < Quotes
      Mid$(StripInputEx, One, One) = ASpace
      StripInputEx = Trim$(StripInputEx)
   Loop

   Exit Function
   Resume
errorline:        ' stop
   If LenB(StripInputEx) = Zero Then
      StripInputEx = StripInput(Zero)
      If Len(StripInputEx) > Zero Then
         Resume Next
      End If
   End If
End Function


Sub GetAnalysedFlag()
'Stop
   On Error GoTo errorline
   Dim RowItem As Single, ColItem As Single

   Section = Zero
   NextSection = Zero
   workbuffer = StripInputEx

   GridOut.Caption = vbNullString
   GridOut.Image1.ToolTipText = vbNullString
   Set GridOut.Picture2.Picture = Nothing
   Set GridOut.Picture5.Picture = Nothing
```

```
   GridOut.Picture2.Picture = LoadPicture(NotString)
   GridOut.Picture5.Picture = LoadPicture(NotString)

   If inGridVER(Zero, gDoc) > 12.7 Then
      workbuffer = StripInputEx
      '\\  ans = NextNumber
      If NextNumber = One Then
         Analyzed (True)
      Else
         Analyzed (False)
      End If
      workbuffer = StripInputEx
   ElseIf Val(Left$(workbuffer, Six)) = One Then
      Analyzed (True)
   Else
      Analyzed (False)
   End If

   ColItem = InStr(workbuffer, "EigenValues =")
   N7 = Zero
   If ColItem > One Then
      For RowItem = One To n6
         ColItem = InStr(ColItem + One, workbuffer, "=")
         EigenValues(RowItem) = Val(Mid$(workbuffer, ColItem + One))
         If EigenValues(RowItem) > EigenValues(RowItem - One) Then N7 = RowItem
      Next
   End If
   GridIn.Table.FormatString = vbNullString
   Exit Sub
   Resume
errorline:        ' stop

   WarningError Err, "GetAnalysedFlag", , , vbYes
End Sub



Function NextNumber(Optional ByRef i As Long = Zero) As Single
   On Error GoTo errorline
   If i > Zero Then
      NextNumber = Val(Tools(i))
      i = i + One
      Exit Function
   End If

   If Len(workbuffer) = Zero Then
      workbuffer = StripInputEx
```

```
        If gCancel Then
            NextNumber = Zero
            Exit Function
        End If
    End If
    If InStr(workbuffer, ",") > Zero Then Mid$(workbuffer, InStr(workbuffer, ","), One) = ASpac
e
    If InStr(workbuffer, ASpace) = Zero Then
        NextNumber = Val(workbuffer)
        workbuffer = vbNullString
    Else
        NextNumber = Val(Left$(workbuffer, InStr(workbuffer, ASpace)))
        workbuffer = Trim$(Mid$(workbuffer, InStr(workbuffer, ASpace)))
        If Left$(workbuffer, One) = "<" Then workbuffer = Trim$(Mid$(workbuffer, InStr(workbu
ffer, ">") + One))
    End If
    Exit Function

errorline:        ' stop
    WarningError Err, "inGridIO"
    gCancel = True
End Function
Public Function ReadUntil() As Single
    On Error GoTo errorline
    If Len(workbuffer) = Zero Then
        workbuffer = StripInput(-Half)
        If gCancel Then
            ReadUntil = Zero
            Exit Function
        End If
    End If
    If Left$(workbuffer, One) = vbTab Then Mid$(workbuffer, One) = ASpace
    workbuffer = Trim$(workbuffer)
    If InStr(workbuffer, ASpace) = Zero Then
        ReadUntil = Val(workbuffer)
        workbuffer = vbNullString
    Else
        ReadUntil = Val(Left$(workbuffer, InStr(workbuffer, ASpace)))
        workbuffer = Trim$(Mid$(workbuffer, InStr(workbuffer, ASpace) + One))
    End If
    Exit Function

errorline:        ' stop
    WarningError Err, "inGridIO"
    gCancel = True
    Exit Function
```

```
End Function



Sub grname(ByVal workz6 As String)
'\\  Set FullGridName and extension
    FullGridName = Trim$(workz6)
    If InStr(workz6, ".") = Zero Then
        FullGridName = workz6 & ".ing"
    End If
End Sub

Sub gropen()
'\\   Open Grid control
    grname FullGridName
    '\\  call ShowMe(Gridout)

    grload
End Sub

Public Sub grloadb(Optional ByVal Restart As Boolean = False)
    Dim lWork As Long
    On Error GoTo errorline
    Dim workz6 As String
    lWork = Screen_MousePointer(, vbDefault)
    If InStr(LCase$(GridTitle), "chess3") Then
        SetfBoard

        ConRows = Thousand    '\\   flag value only
        '\\          CelRows =zero'ven
        If MigoyaDoc <> gDoc Or fBoard.lbl_playno.Caption = Zero Then
            fBoard.OpenChessGame GridTitle, False

            MigoyaDoc = gDoc
            If NmC <> 64 Then
                ProcessingFlag = PF_ExitGrid
                '\\                fBoard.OpenIngridChessGame GridTitle
                OpenIngridChessGame GridTitle
                '\\                savegr
            End If
        Else
            '\\  doesn't do this at startup and takes saved windowstate
            If StartUpTypeIsNormal Then fBoard.WindowState = vbNormal

        End If
        fBoard.draw_board
```

```
      '\\  here because taskbar autohide
      '\\  sometimes doesn't and minimized bar gets hidden
      '\\         If Not StartUpTypeIsNormal Then fBoard.Visible = False
   Else
      '\\         MigoyaDoc = ZERO

   End If
   If Not m_bFrameRelayTo Then
      FrameRelayTo = SettingsGet(iniName, RegSettings, "FrameRelayTo", FrameRelayTo)
      FrameRelayFrom = SettingsGet(iniName, RegSettings, "FrameRelayFrom", FrameRelayFro
m)
   End If
   workz6 = LCase$(getPictureList)
   If workz6 = "select" And InStr(LCase$(FullGridName), LCase$(App.path)) > Zero Then
      If Not StartUpTypeIsPreview Then lWork = ShowMe(GridOut)
      Select Case MsgBoxEx(FullGridName & " needs you to select Pictures from your system to
 add texture to the display. Click No to set up as Screensaver", vbYesNoCancel)
      Case vbYes
         PictOpen
         savegr
      Case vbNo
         Select Case MsgBoxEx("Or Do you want to Install as well as a ScreenSaver?" & vbCr &
"Pictures are needed for palletes and textores.", vbYesNo)
         Case vbYes
            ShellExecute Zero, "install", "ingrid.scr", Zero, Zero, One
         Case vbNo
            putPictureList NotString
            savegr
         End Select
      Case vbCancel
         gCancel = True
         SetExitType PF_StartExit, "grloadb problem"
         Exit Sub
      End Select

   ElseIf (LenB(workz6) = Zero Or workz6 = "select") And (Not StartUpTypeIsNormal Or inGrid
_mnuViewAutoRedraw_Checked = True) Then
      GridOutPicture App.path & "\splash1.jpg", "Now"
      If StartUpTypeIsNormal Then
         GridOut.Picture2.Visible = True
         lWork = ShowMe(GridOut)
      End If

      Exit Sub

   End If
```

```
   GridOut.Picture2.ScaleMode = vbCharacters
   If FullGridName = " .ing" Then
      If Not StartUpTypeIsPreview Then UserParams.Show vbModeless
      Exit Sub
   End If

   If workz6 <> NotString Then
      If Restart = False Then
         GridOut.Picture2.Visible = False
      End If
      NextPicture "Fetched", Restart
      SetPicture
      If StartUpTypeIsNormal Then
         '\\             GridOut.Visible = False
         ProcessingFlag = PF_Null
         '\\  call ShowMe(Gridout)
      End If
   Else
      GridJPEG = Trim$(FullGridName)
      If InStr(GridJPEG, ".ing") Then
         Mid$(GridJPEG, InStr(GridJPEG, ".ing")) = ".jpg"

         If InStr(One, GridJPEG, "tp:", vbTextCompare) = Zero Then
            If GridOutPicture(GridJPEG, "no error") <> "no error" Then
               GridJPEG = vbNullString
               GridOut.Caption = vbNullString
               GridOut.Image1.ToolTipText = vbNullString
            End If
         Else
            GridJPEG = vbNullString
         End If
         '\\  max255 = lMax(lMin(255, GridOut.Picture2.ScaleWidth), 76)
         Max255 = SetMax255
         If Not Analysed Then
            QuadBottom = GridOut.Picture2.ScaleHeight / Two - 2:    '\\  11.5
            QuadRight = Max255 / Two - 2:    '\\  38.5
         End If

      End If
   End If



   Exit Sub
   Resume
errorline:        ' stop
   WarningError Err, "grloadb " & GridOut.Caption
```

```
modGridIn - 29

End Sub

Sub OpenIngridChessGame(Optional ByVal GameName As String = vbNullString)
   If PF_Ending Then Exit Sub
   '\\          Dim x As Long, y As Long, z As Long, aux2 As Long
   On Error GoTo errorline    '\\  required
   If GameName = vbNullString Then GameName = "MigoyaChess_game"
   Open App.path & "\" & GameName & ".mig" For Input As #One
   Dim Piece As String    '\\  , x As Long, y As Long, z As Long, Play As Long, aux2 As Long
   Dim x As Integer, y As Integer, z As Integer, PlayNbr As Integer, aux2 As Variant

   Input #One, PlayNbr
   Input #One, y    '\\  user_color
   Input #One, y    '\\  dificulty

   NmE = PlayNbr + One
   NmC = 64
   m8 = NmE + 2
   n8 = NmC + 2
   N7 = 3
   grinit "Resize"
   For y = One To Eight
      For x = One To Eight
         GridIn.Table.TextMatrix((y - One) * Eight + x + One, One) = Format(y * Ten + x, "00
")
      Next
   Next

   For z = Zero To NmE - One
      GridIn.Table.TextMatrix(One, z + Two) = "p" & Format(z, "00")
      For y = -One To Ten
         For x = -One To Ten
            Input #One, aux2    '\\  (.mig) format
            If Not (x < One Or x > Eight Or y < One Or y > Eight) Then
               Piece = vbNullString
               If aux2 > Zero Then
                  Select Case aux2
                  Case pawn
                     Piece = "bP"
                  Case knight
                     Piece = "bK"
                  Case bishop
                     Piece = "bB"

                  Case rook
                     Piece = "bR"
```

```
                    Case queen
                        Piece = "bQ"

                    Case black_king
                        Piece = "bK"
                    End Select
                Else
                    Select Case aux2
                    Case -pawn
                        Piece = "WP"
                    Case -knight
                        Piece = "WK"
                    Case -bishop
                        Piece = "WB"

                    Case -rook
                        Piece = "WR"

                    Case -queen
                        Piece = "WQ"

                    Case -white_king
                        Piece = "WK"
                    End Select
                End If

                GridIn.Table.TextMatrix((x - One) * Eight + y + One, z + Two) = Piece
            End If
        Next
    Next
Next
Close
params
ConRows = Thousand
UserParams.UpdateUserGame
SetDirty True, MigoyaDoc
Exit Sub
Resume
errorline:        ' stop
If Err <> 53 And Err <> 91 Then WarningError Err, "Opening_Migoya_Chess", , , vbYes

End Sub




Public Sub ReadControls(Optional ByVal WhichDoc As Long = Zero)
```

```
   Dim lWork As Long


   If WhichDoc > Zero Then
      '\\  WorkBuffer = StripInput(-1, "<ToolboxSettings>", WhichDoc)
      '\\  GetToolbox
      workbuffer = vbNullString
      lWork = GetToolbox(StripInput(-1.5, "<ToolboxSettings>", WhichDoc))
      If XDoEventsX Then Exit Sub
   ElseIf inGridVER(Zero, nDoc) >= 9.5 And Not ReadOnlyStatus(True) Then
      '\\  WorkBuffer = StripInputEx

      '\\  GetToolbox
      workbuffer = vbNullString
      lWork = GetToolbox(StripInput(Half))
      If FullScreen = False And Not StartUpTypeIsScreensaver Then
         GridOut.WindowState = vbNormal
      Else

         FullScreen = True
         GridOut.WindowState = vbMaximized

      End If

      Exit Sub
   Else    '\\  ingridver(two,gDoc)
      workbuffer = vbNullString
      Do While Len(workbuffer) < 70
         '\\  Input #ONE, WorkBuffer
         workbuffer = StripInputEx
      Loop

   End If

End Sub


Function SaveEigenValues() As String
   Dim ColItem As Long    '\\  , RowItem As Long
   Dim e() As String
   ReDim e(Zero)
   e(Zero) = "<EigenValues> - Cumulated EigenValues"
   For ColItem = One To n6
      ReDim Preserve e(ColItem)
      e(ColItem) = str(EigenValues(ColItem))
   Next
   SaveEigenValues = JoinX(e(), " =") & "</EigenValues>"
```

```
modGridIn - 32

End Function


Function SplitGridTitle(ByRef gDoc As Long, ByRef workz6 As String) As String
   Dim ans As Long
   Do While InStr(workz6, "<")    '\\   fixes for XML requirement in 1.2.7
      Mid$(workz6, InStr(workz6, "<")) = "_"
      workz6 = ReplaceHTML(workz6, TAGUseMap, ExtractHTML(workz6, TAGUseMap & Quote
s))
   Loop

   ans = InStr(workz6, "_")
   If ans Then
      If ans > One Then
         SplitGridTitle = Trim$(Left$(workz6, ans - One))    '\\
      End If
      JPEGs(gDoc).toHref.Text = ExtractHTML(workz6, TAGHref)
      JPEGs(gDoc).toImgSrc.Text = ExtractHTML(workz6, TAGImgSrc)
      JPEGs(gDoc).toUseMap.Text = ExtractHTML(workz6, TAGUseMap)
      JPEGs(gDoc).toPageTitle.Text = ExtractHTML(workz6, "title=")
      JPEGs(gDoc).tBrowseURL.Text = ExtractHTML(workz6, "browse=")
      If Len(SplitGridTitle) = Zero Then
         SplitGridTitle = JPEGs(gDoc).toPageTitle.Text
      End If
   Else
      SplitGridTitle = Trim$(workz6)    '\\

   End If
End Function


Function SplitGridUser(ByRef gDoc As Long, ByRef workz6 As String) As String
   Dim ans As Long, workz7 As String
   ans = InStr(workz6, "_")
   If ans Then
      If ans > One Then
         SplitGridUser = Trim$(Left$(workz6, ans - One))    '\\  inGridUser
      End If
      With FTPs(gDoc)
         workz7 = ExtractHTML(workz6, "server=")
         If workz7 <> NotString Then .tServer.Text = workz7
         If FTPServer = vbNullString Then
            FTPServer = .tServer.Text
         ElseIf .tServer.Text = vbNullString Then
            .tServer.Text = FTPServer
         End If
         workz7 = ExtractHTML(workz6, "username=")
```

```
        If workz7 <> NotString Then .tUser.Text = workz7
        If FTPUser = vbNullString Then
            FTPUser = .tUser.Text
        ElseIf .tUser.Text = vbNullString Then
            .tUser.Text = FTPUser
        End If
        workz7 = ExtractHTML(workz6, "directory=")
        If workz7 <> NotString Then .tRemoteFile(Zero).Text = workz7
        If FTPstrDirectory = vbNullString Then
            FTPstrDirectory = .tRemoteFile(Zero).Text
        ElseIf .tRemoteFile(Zero).Text = vbNullString Then
            .tRemoteFile(Zero).Text = FTPstrDirectory
        End If
        If Len(SplitGridUser) = Zero Then
            SplitGridUser = .tUser.Text
        End If
    End With
  Else
    SplitGridUser = Trim$(workz6)    '\\  inGridUser

  End If
End Function

Sub StoreGrid()
  On Error GoTo errorline
  If gDoc = ReadDoc Then
    Exit Sub
  End If
  Dim ans As Single
  SetReadOnly False

  FileType = ReplaceHTML(FileType, "Title=", "inGridX Version ", "Title =")


  Dim workz7 As String
  workz7 = SettingsGet(iniName, "Software Distribution", "inGridX Version", NotString)
  '\\  repair faux pas < 12.9
  If InStr(One, workz7, "</Filetype", vbTextCompare) Then
    workz7 = Left$(workz7, InStr(One, workz7, "</Filetype", vbTextCompare) - One)
  End If
  ans = Val(ExtractHTML(workz7, "value=", , "value ="))
  If ans >= inGridVER(Zero, gDoc) And ans > GridVerNum Then
    FileType = workz7
  Else
    If GridVerNum >= ans Then
      FileType = ReplaceHTML(FileType, "Value=", Trim$(str(sMax(Val(ExtractHTML(FileTyp
e, "Value=")), sMax(inGridVER(Zero, gDoc), 15.7)))), "Value =")
```

```
        If Len(ExtractHTML(workz7, TAGHref)) <> Zero Then If Len(ExtractHTML(FileType, TA
GHref)) = Zero Then FileType = ReplaceHTML(FileType, TAGHref, ExtractHTML(workz7, TAGHr
ef))
        If Len(ExtractHTML(workz7, TAGAlt)) <> Zero Then If Len(ExtractHTML(FileType, TAG
Alt)) = Zero Then FileType = ReplaceHTML(FileType, TAGAlt, ExtractHTML(workz7, TAGAlt))

        '\\  MsgBoxEx "' code to update software repository goes here"
        If inGrid.mnuFileMRU(One).Caption = ExtractHTML(workz7, TAGHref) Then
           fVBZip.Timer1.Enabled = True
        End If
     End If
     SettingsSave iniName, "Software Distribution", "inGridX Version", FileType
  End If
  GridIn.Table.Redraw = False
  GridIn.Table.Visible = False

  GridIn.Table.row = One
  GridIn.Table.col = One
  GridIn.Table.Rows = NmC + Four
  GridIn.Table.Cols = LastDataCol + Four
  GridIn.Table.RowSel = NmC + Three
  GridIn.Table.ColSel = LastDataCol + Three
  fDoc(gDoc).DocText.Text = ""
  '    fDoc(gDoc).DocText_SelStart = Zero

  fDoc(gDoc).DocText.SelText = TAGXMLHeader & vbCrLf & IngridXMLHeader & vbCrLf
  '     SaveHeader
'   On Error GoTo errorline
  Dim H() As String
  ReDim H(2 To 11)


  H(Two) = "<Filetype>" & FileType & "</Filetype>"
  If inGridVER(Zero, gDoc) <> maxInt And Analysed Then inGridVER(Zero, gDoc) = Zero
  If GridVerNum = Zero Then
     H(Three) = "<Version>" & str(sMax(inGridVER(Zero, gDoc), 15.7)) & "</Version>"
  Else
     H(Three) = "<Version>" & str(sMax(inGridVER(Zero, gDoc), GridVerNum)) & "</Version
>"
  End If
  H(Four) = "<DifferentScore>" & str(B9) & "</DifferentScore>"
  H(Five) = "<RelatedScore>" & str(T9) & "</RelatedScore>"
  H(Six) = "<Date>" & GridDate & "</Date>"
  H(Seven) = "<DecisionView>" & ViewType & "</DecisionView>"
  H(Eight) = "<Normalised>" & NormalCode & "</Normalised>"
  H(Nine) = "<Weighted>" & WeightCol & "</Weighted>"
  H(10) = "<NumberOfElements>" & str(NmE + CelCols * Cent) & "</NumberOfElements>"
```

```
   H(11) = "<NumberOfConstructs>" & str(NmC + ConCols * Cent) & "</NumberOfConstructs
>"
   If NmC < 150 And NmE < 150 And ViewType = "C" Then
      fDoc(gDoc).DocText.SelText = CompressString(JoinX(H(), vbCrLf) & SaveAnalysis)
   Else
      If NmC < 150 And NmE < 150 Then
         fDoc(gDoc).DocText.SelText = JoinX(H(), vbCrLf) & SaveAnalysis
      Else
         fDoc(gDoc).DocText.SelText = JoinX(H(), vbCrLf) & "<inGridTable>" & vbCrLf

         fDoc(gDoc).DocText.SelText = TableClip(GridIn.Table)

         '\\  FLAG: SET TO ZERO IF CHANGES ARE MADE or to omit the following for compatab
ility";
         fDoc(gDoc).DocText.SelText = vbCrLf & "</inGridTable>" & vbCrLf & "<AnalyzedFlag
>" & " 0 " & "</AnalyzedFlag>"
      End If
      JobName = ASpace
   End If

   fDoc(gDoc).DocText.SelText = vbCrLf & IngridXMLFooter
   GridIn.Table.RowSel = One
   GridIn.Table.ColSel = One
   GridIn.Table.Visible = True
   inGridVER(One, gDoc) = True
   Exit Sub
   Resume
errorline:        ' stop
   WarningError Err, "inGridIO", , , vbYes
End Sub

Function StripInput(ByVal file As Single, Optional ByVal StartSearch As String = vbNullString, O
ptional ByVal tDoc As Long = Zero) As String
   On Error GoTo errorline

   Dim InputString As String, Endline As Long, lPos As Long

   gCancel = False
   StripInput = vbNullString
   If file >= One Then
      If file = Billion Then
         StripInput = Mid$(fDoc(Zero).DocText.Text, StartLine)
      Else    '\\  If Not EOF Then
         Line Input #file, StripInput
      End If

      Exit Function
```

```
   Else
      If file <= -One Then
         If tDoc = Zero Then
            nDoc = gDoc
         Else
            nDoc = tDoc
         End If
         If Len(StartSearch) = Zero Then
            StartLine = -file
         Else
            fDoc(gDoc).DocText_SelStart = Zero
            fDoc(gDoc).DocText.SelLength = fDoc(gDoc).MeTextLength

            StartLine = fDoc(nDoc).DocText.Find(StartSearch, fDoc(nDoc).DocText_SelStart, , rt
fNoHighlight) + One    '\\  InStr(one, fDoc(nDoc).DocText.SelText, StartSearch)

         End If
      End If
      If StartLine = Zero Then
         StartLine = One
      Else
         '\\  StartLine = StartLine + Len(StartSearch) ????
      End If
      Endline = One + fDoc(nDoc).DocText.Find(vbCrLf, StartLine - One, , rtfNoHighlight)
      If Endline = Zero Then
         Endline = One + fDoc(nDoc).DocText.Find(vbCr, StartLine - One, , rtfNoHighlight)    '\\
 InStr(StartLine, fDoc(nDoc).DocText.Text, vbCr)
         If Endline = Zero Then
            Endline = One + fDoc(nDoc).DocText.Find(vbLf, StartLine - One, , rtfNoHighlight)
'\\  InStr(StartLine, fDoc(nDoc).DocText.Text, vbLf)
            If Endline = Zero Then
               InputString = Mid$(fDoc(nDoc).DocText.Text, StartLine)
               StartLine = fDoc(nDoc).MeTextLength + 2
               '\\  Error = 11
            Else
               InputString = Mid$(fDoc(nDoc).DocText.Text, StartLine, Endline - StartLine)
               StartLine = Endline + One
               Endline = InStr(InputString, "<br>")
               If Endline > Zero Then
                  InputString = Left$(InputString, Endline - One)
               End If
            End If
         Else
            InputString = Mid$(fDoc(nDoc).DocText.Text, StartLine, Endline - StartLine)
            StartLine = Endline + One
         End If
```

```
      Else
         InputString = Mid$(fDoc(nDoc).DocText.Text, StartLine, Endline - StartLine)
         StartLine = Endline + 2
      End If
   End If
   If file = Int(file) Then
      lPos = InStr(InputString, vbTab)
      Do While lPos
         Mid$(InputString, lPos, One) = ASpace
         lPos = InStr(lPos, InputString, vbTab)
      Loop
      lPos = InStr(InputString, "\tab")
      Do While lPos
         Mid$(InputString, lPos, Four) = "    "
         lPos = InStr(lPos, InputString, "\tab")
      Loop
   Else
      StripInput = InputString
      If file = -1.5 Then Exit Function
   End If
   If Left$(InputString, Four) = "\par" Then
      StripInput = Trim$(Mid$(InputString, Five))
   Else
      StripInput = Trim$(InputString)
   End If



   If Left$(StripInput, One) = "<" Then
      '\\  XML?
      lPos = InStr(StripInput, ">")
      If lPos = Zero Then Exit Function
      '\\  lpos = InStr(left$(StripInput, lpos), ASpace)
      If InStr(Left$(StripInput, lPos), ASpace) Then Exit Function
      '\\  stack = stack + One
      '\\  ReDim Preserve XML(stack)
      '\\  XML(stack) = left$(StripInput, lpos)
      If lPos = Len(StripInput) Then
         StripInput = StripInput(file)
      Else
         StripInput = Trim$(Mid$(StripInput, lPos + One))
      End If
   End If
   Exit Function
   Resume
errorline:        ' stop
   gCancel = True
   WarningError Err, "inGridIO"
```

```
modGridIn - 38

End Function



Public Sub Transpose81411()
   Dim RowItem As Single, ColItem As Single
   Print #One, "VERSION"; ","; 8.1411; NotString
   Print #One, GridTitle
   Print #One, inGridUser(gDoc)
   Print #One, B9, T9, GridDate; ","; ViewType; ","; NormalCode; ","; WeightCol
   Print #One, NmC; ConstructTitle
   Print #One, NmE; ElementTitle

   Print #One, GridIn.Table.TextMatrix(One, One)



   For RowItem = One To NmC
      Print #One, GridIn.Table.TextMatrix(One + RowItem, cLabelCol); ASpace; vbTab; GridIn.
Table.TextMatrix(One + RowItem, LastDataCol + Two);
      If Len(GridIn.Table.TextMatrix(One + RowItem, LastDataCol + Three)) <> Zero Then
         Print #One, vbTab; GridIn.Table.TextMatrix(One + RowItem, LastDataCol + Three)
      Else
         Print #One,
      End If

   Next
   Print #One, GridIn.Table.TextMatrix(One, LastDataCol + Two)

   For ColItem = One To NmE
      Print #One, GridIn.Table.TextMatrix(eLabelRow, One + ColItem); ASpace; vbTab; getDes
cription(ColItem);
      If Len(getExtraHTML(ColItem)) <> Zero Then
         Print #One, vbTab; getExtraHTML(ColItem)
      Else
         Print #One,
      End If

   Next
   Print #One, getPictureList
   Print #One, Tab(Seven);
   For RowItem = One To NmC
      Print #One, GridIn.Table.TextMatrix(One + RowItem, cLabelCol); ASpace;
   Next
   Print #One,
   Print #One, getPictureClip
   For ColItem = One To LastDataCol
```

```
      Print #One, GridIn.Table.TextMatrix(eLabelRow, One + ColItem),
      For RowItem = One To NmC
         If fnDataZ1(RowItem, ColItem) = maxInt Then
            Print #One, vbTab; " ? ";
         Else
            Print #One, vbTab; fnDataZ1(RowItem, ColItem);
         End If
      Next
      Print #One, "  "; vbTab; Quotes; getDescription(ColItem); Quotes
   Next
   Print #One, Zero

End Sub


Public Sub savez()
   On Error GoTo errorline
   If ReadOnlyStatus Or InStr(One, FullGridName, ".txt", vbTextCompare) Or InStr(One, FullGri
dName, ".rtf", vbTextCompare) Or IsFullURL(FullGridName) Then
      Select Case MsgBoxEx(Msg170_Read_Only & vbCr & Msg171_temp_file_to_FTP & vbCr &
 Msg172_use_Save_As & vbCr & Msg173_Cancel_nothing, vbYesNoCancel)
      Case vbCancel
         Exit Sub
      Case vbNo
         SaveAs
      Case vbYes
         Dim workz7 As String

         GridOut.Listw.Clear
         GridOut.Listz6.Clear
         workz7 = iniDirectory & "temp\" & "~temp.ing"
         '\\  set local file
         GridOut.Listw.AddItem workz7
         GridOut.Listz6.AddItem FullGridName
         If Len(FTPs(gDoc).tServer) = Zero Then
            FTPs(gDoc).tServer = InputBox(IBS507_Server, , FullGridName)
         End If
         If Len(FTPs(gDoc).tRemoteFile(Zero)) = Zero Then
            FTPs(gDoc).tRemoteFile(Zero) = InputBox(IBS508_Remote_file, , FullGridName)
         End If
         savegr workz7
         If IsFullURL(FullGridName) Then    '\\  InStr(FullGridName, ".htm") > ZERO
            If Not fDoc(Zero) Is Nothing Then Unload fDoc(Zero)
            Set fDoc(Zero) = New frmDocument
            '\\                    gCancel = False
            FileLoad Zero, workz7
            fDoc(Zero).DocText.Text = ReplaceXML(fDoc(gDoc).DocText, "<!--" & fDoc(Zero).D
```

```
modGridIn - 40

ocText.Text) & "-->"
            If fDoc(Zero).MeTextLength = Zero Then
               MsgBoxEx Msg174_XML_corrupted, vbCritical
               gCancel = True
               Exit Sub
            End If
            fDoc(Zero).DocText.SaveFile FilenameEx(workz7), rtfText

         End If
         UploadLocalFile gDoc, GridOut.Listw, GridOut.Listz6
      End Select
      If gCancel Then Exit Sub
      If Left$(GridJPEG, 7) = "http://" Then
         GridJPEG = Left$(FullGridName, Len(FullGridName) - Four) & ".jpg"
         FileCopy iniDirectory & "temp\" & "~temp.jpg", GridJPEG
      End If
   Else
      savegr
   End If
   Exit Sub
   Resume
errorline:        ' stop
   WarningError Err, "inGridIO"
End Sub

Public Sub SaveAs()
   Dim lWork As Long
   '\\   Save As..
   gCancel = False
   If InStr(One, FullGridName, "tp://", vbTextCompare) Then
      FullGridName = Right$(FullGridName, Len(FullGridName) - InStrRev(FullGridName, "/") +
 One)
   End If
   If Not (InStr(One, FullGridName, ".ing", vbTextCompare) > Zero) Then
      If InStr(FullGridName, ".") Then
         Do While Right$(FullGridName, One) <> "."
            FullGridName = Left$(FullGridName, Len(FullGridName) - 1)
         Loop
         FullGridName = Left$(FullGridName, Len(FullGridName) - 1)
      End If
   End If
   FullGridName = Trim$(FileNameAs(STR197_Save_As_inGrid_File, FullGridName, "inGrid Files
 (*.ing)|*.ing"))
   If gCancel Then Exit Sub
   UnResolve
   grname FullGridName
   lWork = Screen_MousePointer(, vbHourglass)
```

```
   fDoc(gDoc).DocText_Filename = Filename_Ini(FullGridName)
   savegr
   lWork = Screen_MousePointer(, vbDefault)
   fCommonDialog.InitDir = CurDir
   AddRecent Five
   setAboutCaption "inGridX - " & FullGridName
   Front
End Sub

Public Sub savegr(Optional ByVal TempGridName As String = vbNullString)
   Dim lWork As Long
   '\\   Save Grid
   'Dim ret As Long
   'ret = GridIn.Visible
   'GridIn.Visible = False

   If Len(TempGridName) = Zero Then
      If IsFullURL(FullGridName) Then
         FullGridName = "untitled.ing"
      End If
      '\\         If lcase$(Right(FullGridName, 11)) = "zipfdoc.ing" Then
      '\\            MsgBox "terrible zipfdoc error"
      '\\            Exit Sub
      '\\         End If
      TempGridName = FullGridName
      If (TempGridName = "untitled" Or TempGridName = " .ing" Or Len(TempGridName) = Ze
ro) Then
         SaveAs
         Exit Sub
      End If

      On Error GoTo errorline
      '\\  If FileType) = zero then
      '\\      Error 321
      '\\  End If
      If InStr(Len(TempGridName) - Five, TempGridName, ".") Then
         TempGridName = Left$(TempGridName, InStr(Len(TempGridName) - Five, TempGridN
ame, ".")) & "ing"
      End If
   End If

   StoreGrid
   '    lWork = SendSound("msgsent.wav")
   fDoc(gDoc).DocText.SaveFile FilenameEx(TempGridName), rtfText

   SetDirty False
   '\\  GridIn.Visible = ret
```

```
    Exit Sub
    Resume
errorline:        ' stop
    If InStr(One, Error, "Path", vbTextCompare) > Zero Then
        If MsgBoxEx(Error & Msg175_Use_SaveAs, vbOKCancel) = vbOK Then SaveAs
        Exit Sub
    End If
    inGrid.Visible = True
    '\\  warning = vbCancel
    inGrid.SetFocus
    WarningError Err, "inGridIO"


End Sub
Sub SetDirty(boolcase As Integer, Optional ByVal pDoc As Long = -1)
    On Local Error Resume Next
    If pDoc = Zero Then Exit Sub
    '\\      If pDoc = six Then Stop
    If pDoc = -One Then pDoc = gDoc
    If boolcase < Two Then dirty = boolcase
    If boolcase Mod Five = Zero Then    '\\   five means W.I.P. text changed warning
        inGrid.SaveGridX.Enabled = False
        inGridVER(Three, pDoc) = Zero               '\\   SaveDirty
    Else
        If Analysed And inGridVER(One, pDoc) = Five And ReadDoc <> Zero Then
            If fDoc(ReadDoc).ReaderReStartTotalTime = Zero Then
                If boolcase = True Then
                    DealToText
                ElseIf MsgBoxEx("Deal with the text", vbYesNo, "Unidentified text will be scrapped")
 = vbYes Then
                    DealToText
                End If
            End If
        End If
        If boolcase = Four Then
            inGridVER(Three, pDoc) = boolcase
        End If
        If pDoc = gDoc Then
            inGrid.SaveGridX.Enabled = True
        End If
    End If
    inGridVER(One, pDoc) = boolcase                 '\\   current grid status
End Sub

Function NEXTWORD(Optional ByRef i As Long = Zero) As String
    If i > Zero Then
        NEXTWORD = Tools(i)
```

```
      If Left$(NEXTWORD, One) = Quotes Then
         NEXTWORD = Mid$(NEXTWORD, Two)
         If Right$(NEXTWORD, One) = Quotes Then
            NEXTWORD = Left$(NEXTWORD, Len(NEXTWORD) - One)
         End If
      End If
      i = i + One
      Exit Function
   End If
   If Len(workbuffer) = Zero Then
      workbuffer = StripInputEx
      If Len(workbuffer) = Zero Then
         NEXTWORD = vbNullString
         Exit Function
      End If
   End If
   If InStr(workbuffer, ",") Then Mid$(workbuffer, InStr(workbuffer, ","), One) = ASpace
   If Left$(Trim$(workbuffer), One) = Quotes Then
      workbuffer = Mid$(Trim$(workbuffer), Two)
      NEXTWORD = Left$(workbuffer, InStr(workbuffer, Quotes) - One)
      workbuffer = Trim$(Mid$(workbuffer, InStr(workbuffer, Quotes) + One))
   ElseIf InStr(workbuffer, ASpace) = Zero Then
      NEXTWORD = workbuffer
      workbuffer = vbNullString
   Else
      NEXTWORD = Left$(workbuffer, InStr(workbuffer, ASpace) - One)
      workbuffer = Trim$(Mid$(workbuffer, InStr(workbuffer, ASpace)))
   End If
   If Left$(NEXTWORD, One) = "<" Then workbuffer = Trim$(Mid$(workbuffer, InStr(workbuff
er, ">") + One))
   If InStr(NEXTWORD, "<") > One Then
      NEXTWORD = Left$(NEXTWORD, InStr(NEXTWORD, "<") - One)
      If LenB(NEXTWORD) = Zero Then NEXTWORD = NEXTWORD
   End If
   If Len(NEXTWORD) = Zero Then NEXTWORD = NEXTWORD

End Function

Public Sub PicClipLoad()
   Dim lWork As Long
   Dim workz7 As String

   On Error GoTo errorline
   workbuffer = getPictureClip
   If MigoyaDoc > Zero Then
      If Not fBoard Is Nothing Then
         fBoard.draw_board
```

```
        End If

    ElseIf Len(workbuffer) <> Zero Then
        Set GridOut.PictureClip1.Picture = Nothing
        If InStr(One, workbuffer, "tp://", vbTextCompare) Then

            lWork = GetImage(workbuffer, iniDirectory & "temp\" & "~temp.jpg")

            If lWork Then
                GridOut.PictureClip1.Picture = LoadPicture(iniDirectory & "temp\" & "~temp.jpg")
            Else
                Exit Sub
            End If
        Else

            workz7 = workbuffer
            If InStr(workz7, "\") = Zero Then
                workz7 = fCommonDialog.InitDir & workz7
            End If

            GridOut.PictureClip1.Picture = LoadPicture(workz7)
        End If
    Else
        If Len(JPEGs(gDoc).tiHref.Text) = Zero Then Exit Sub

        GoTo errorline


    End If

    Exit Sub

errorline:        ' stop
    Set GridOut.PictureClip1.Picture = Nothing
    GridOut.PictureClip1.Picture = LoadPicture(NotString)
    '\\  ans = MsgBoxEx("PictureClip " & workz7 & " Not found. Find it yourself?", vbOKCancel)

    If MsgBoxEx(Msg176_PictureClip & workz7 & Msg177_Not_found_Find_it, vbOKCancel) = vb
OK Then
        Dim sFile As String



        With fCommonDialog
            '\\  To Do
            '\\  set the flags and attributes of the
            .flags = OFN_FILEMUSTEXIST
```

```
        .FileName = workz7
        '\\  common dialog control
        .Filter = STR198_Picture_Files & " |*.gif;*.jpg;*.bmp"
        .ShowOpen
        If Len(.FileName) = Zero Then

            Exit Sub
        End If
        sFile = .FileName

    End With

    '\\  To Do
    '\\  process the opened file
    Set GridOut.PictureClip1.Picture = Nothing
    GridOut.PictureClip1.Picture = LoadPicture(sFile)
    JPEGs(gDoc).tiImgSrc.Text = sFile
  Else
    JPEGs(gDoc).tiImgSrc.Text = vbNullString

  End If
End Sub

Sub ColorTable()
  On Error GoTo errorline
  Dim RowItem As Single, ColItem As Single
  UnResolve

  With GridIn.Table
    .Redraw = False
    For RowItem = One To NmC
      eLoad(RowItem, Zero) = Zero
    Next
    For ColItem = One To NmE
      cLoad(Zero, ColItem) = Zero
    Next
    .row = One
    .col = One
    .RowSel = NmC + One
    .ColSel = NmE + One
    .CellBackColor = GridWhite
    .CellForeColor = vbBlack
    For RowItem = One To NmC
      For ColItem = One To NmE
        cLoad(Zero, ColItem) = cLoad(Zero, ColItem) + cLoad(RowItem, ColItem) ^ Two
        eLoad(RowItem, Zero) = eLoad(RowItem, Zero) + eLoad(RowItem, ColItem) ^ Tw
o
```

```
            Next
        Next
        For RowItem = One To N7
            .row = RowItem + One
            If eLoad(RowItem, Zero) = Zero Then
                N7 = RowItem - One
                Exit For
            Else
                For ColItem = One To NmE
                    .col = ColItem + One
                    .CellBackColor = fMidColor + Sgn(eLoad(RowItem, ColItem)) * fColorRange * eL
oad(RowItem, ColItem) ^ Two / eLoad(RowItem, Zero)
                Next
            End If
        Next
        For ColItem = One To N7
            .col = ColItem + One
            If cLoad(Zero, ColItem) = Zero Then
                N7 = ColItem - One
                Exit For
            Else
                For RowItem = One To NmC
                    .row = RowItem + One
                    .CellForeColor = bMidColor - Sgn(cLoad(RowItem, ColItem)) * fColorRange * bC
olorFactor * (cLoad(RowItem, ColItem) ^ Two / cLoad(Zero, ColItem))
                Next
            End If
        Next
        .row = One
        .col = One
        .Redraw = True
        If .Visible = True Then GridinTextEntry
    End With
    Exit Sub
    Resume
errorline:        ' stop
    WarningError Err, "ColorTable", , , vbYes

End Sub
Function FindVer(ByVal FindVersion As String) As Single
    FindVer = Val(ExtractHTML(FindVersion, "value=", , "value ="))
    If FindVer = Zero Then FindVer = Val(FindVersion)
End Function

Function FindinGridX(ByVal FindVersion As String) As String
    Dim lWork As Long
```

```
    lWork = FindinGrid(FindVersion)
    If CellRow < One Or (CellCol < Two Or CellCol > LastDataCol) Or InStr(One, GridIn.Table.Te
xtMatrix(CellRow, CellCol), FindVersion, vbTextCompare) = Zero Then
        FindinGridX = vbNullString
        '\\  MsgBoxEx "not a valid archive"

        Exit Function
    End If
    GridIn.Table.col = CellCol
    GridIn.Table.ColSel = CellCol
    GridSort (One)
    GridIn.Table.row = NmC + One
    FindinGridX = GridIn.Table.Text


End Function

Function fnWeight(ByVal DataRow As Long) As Single

    Dim worka As String    '\\  , item As Long, ans As Single

    gAns = Zero
    worka = Trim$(GridIn.Table.TextMatrix(DataRow + One, LastDataCol + One))
    Select Case worka
    Case OneText    '\\  If worka = "?" Then
        fnWeight = One

    Case Else
        If InStr(worka, "/") Then
            If InStr(One, worka, "chr$(", vbTextCompare) = One Then
                gAns = Val(Mid$(worka, Five))
            Else
                gAns = Val(worka)
            End If
            worka = Mid$(worka, InStr(worka, "/") + One)
            fnWeight = One / (Val(worka) - gAns)

        Else
            fnWeight = Val(worka)
        End If
    End Select
End Function

Sub xDataIn(ByVal i9 As Long, ByVal j9 As Long, ByVal amount As Single)
    If amount = maxInt Then
        GridIn.Table.TextMatrix(i9 + One, j9 + One) = "?"
    Else
```

```
        GridIn.Table.TextMatrix(i9 + One, j9 + One) = Trim$(str(amount))
    End If
End Sub

Function ExtractHTML(ByRef HTMLString As String, ByRef HTMLTag As String, Optional ByRef D
efault As String = vbNullString, Optional ByRef AltTag As String = vbNullString, Optional ByRef
lPos As Long = Zero, Optional ByVal delim As String = Quotes) As String
'Dim lPos As Long
    ExtractHTML = Default

    lPos = InStr(lPos + One, HTMLString, HTMLTag, vbTextCompare)
    Dim sQuotes As String, lWork As Long
    sQuotes = Mid$(HTMLString, lPos + Len(HTMLTag), One)
    If sQuotes <> """" Then sQuotes = delim
    If lPos > Zero Then
        If AltTag = "<" Then
            lPos = lPos + Len(HTMLTag)
            sQuotes = AltTag
        Else
            lPos = lPos + Len(HTMLTag) + One
        End If
    Else
        If Len(AltTag) <> Zero Then
            lPos = InStr(One, HTMLString, AltTag, vbTextCompare)
            If lPos > Zero Then
                lPos = lPos + Len(AltTag) + One
            End If
        End If
    End If
    If lPos > Zero Then
        lWork = InStr(lPos, HTMLString, sQuotes)
        If lWork = Zero Then
            lWork = InStr(lPos, HTMLString, vbLf)    '\\  Google Chrome
            If lWork = Zero Then lWork = InStr(lPos, HTMLString, vbCr)    '\\  just for some weird
case
        End If
        If lWork > lPos Then
            ExtractHTML = Mid$(HTMLString, lPos, lWork - lPos)
        Else
            ExtractHTML = Mid$(HTMLString, lPos)
        End If
    End If
End Function


Function CopyGrid(Optional ByRef GetPicture As Boolean = True) As Boolean
    On Error GoTo errorline
```

```
   CopyGrid = True


'    ClipBoardClear

   If GetPicture = True Then
      JPEGs(gDoc).picTest.Picture = Nothing
      JPEGs(gDoc).picTest.Picture = CaptureClient(GridOut)    '\\  GridOut.Picture2.Image
      Clipboard.SetData JPEGs(gDoc).picTest.Picture, Two
      '\\  Stop
   End If
   If Section > Nine Then

      JobName = MapThis
      MapThsHTML = vbNullString
      PlotInit
      RePlot
      Resolv
      Set JPEGs(gDoc).picTest.Picture = CaptureClient(GridOut)
      '\\  JPEGs(gDoc).picTest.Picture = GridOut.Picture2.Image
      JPEGs(gDoc).HTMLTextBox.Text = MapThsHTML
      JobName = vbNullString

   ElseIf Section < Nine Then
      If fTableLeft.flex01.Cols > One Then
         fTableLeft.flex01.row = Zero
         fTableLeft.flex01.col = Zero
         fTableLeft.flex01.RowSel = fTableLeft.flex01.Rows - One
         fTableLeft.flex01.ColSel = fTableLeft.flex01.Cols - One
      End If
      If fTableRight.flex01.Cols > One Then
         If NextSection < Four Then
            fTableRight.flex01.row = One
            fTableRight.flex01.col = One
         Else
            fTableRight.flex01.row = Zero
            fTableRight.flex01.col = Zero
         End If
         fTableRight.flex01.RowSel = fTableRight.flex01.Rows - One
         fTableRight.flex01.ColSel = fTableRight.flex01.Cols - One



      End If
      Select Case NextSection
      Case 2, 3
         If fTableRight.flex01.RowSel = Zero Then
            ClipBoardSetText fTableLeft.flex01.Clip, False
```

```
modGridIn - 50

            Else
                ClipBoardSetText fTableLeft.flex01.Clip & vbCrLf & fTableRight.flex01.Clip, False
            End If
        Case 4, 6, 7
            ClipBoardSetText fTableLeft.flex01.Clip, False
        Case 5, 8
            ClipBoardSetText fTableRight.flex01.Clip, False
        End Select
        fTableLeft.flex01.RowSel = Zero
        fTableRight.flex01.RowSel = Zero
        fTableLeft.flex01.ColSel = Zero
        fTableRight.flex01.ColSel = Zero
    End If
    If inGrid.DrawState.value = vbChecked Then
        If SettingsGet(iniName, "Should this be shown", "Persistent", "Yes") = "Yes" Then
            Select Case MsgBoxEx(Msg122_NoPersistent_Autoredraw & vbCr & Msg123_CancelAbo
rt, vbYesNoCancel, Msg124_show_again)
            Case vbNo
                SettingsSave iniName, "Should this be shown", "Persistent", "No"
            Case vbCancel
                CopyGrid = False
                Exit Function
            End Select
        End If
    End If
    Exit Function
    Resume
errorline:        ' stop
    WarningError Err, "CopyGrid"
End Function
Public Function FindinGrid(Optional ByRef FindString As String = vbNullString, Optional ByRef p
Doc As Long = -1) As Long
    Static i As Long, J As Long, FoundString As String    '\\   c As Long,
    If pDoc = -One Then
        pDoc = gDoc
    End If

    With GridIn.Table
        If Len(FindString) = Zero Then
            FindString = InputBox(IBS502_text_to_find, , FindString)
            If .row = .RowSel Then .RowSel = .Rows - One
            If .col = .ColSel Then .ColSel = .Cols - One

            For i = .row To .RowSel
                For J = .col To .ColSel
                    If InStr(One, .TextMatrix(i, J), FindString, vbTextCompare) > Zero Then
                        CellCol = J
```

```
                CellRow = i
                FindinGrid = CellCol - One
                grent1
                Exit Function
            End If
        Next
    Next
Else
    '\\   search element descriptions and extraHTML
    .row = .Rows - Two
    .col = One
    .RowSel = .Rows - One
    .ColSel = .Cols - Two
    For i = .row To .RowSel
        For J = .col To .ColSel
            If InStr(One, .TextMatrix(i, J), FindString, vbTextCompare) > Zero Then
                CellCol = J
                CellRow = i
                FindinGrid = CellCol - One
                Exit Function
            End If
        Next
    Next

    '\\   search construct descriptions and extraHTML

    .row = One
    .col = .Cols - Two
    .RowSel = .Rows - Three
    .ColSel = .Cols - One
    For J = .col To .ColSel
        For i = .row To .RowSel
            If InStr(One, .TextMatrix(i, J), FindString, vbTextCompare) > Zero Then
                CellCol = J
                CellRow = i
                FindinGrid = CellRow - One + NmE
                Exit Function
            End If
        Next
    Next

    '\\   search body of grid

    .row = Two
    .col = Two
    .RowSel = .Rows - Three
    .ColSel = .Cols - Three
```

```
        For i = .row To .RowSel
            For J = .col To .ColSel
                If InStr(One, .TextMatrix(i, J), FindString, vbTextCompare) > Zero Then
                    CellCol = J
                    CellRow = i
                    FindinGrid = CellCol - One
                    Exit Function
                End If
            Next
        Next

        '\\   Search Row 1 element labels ++

        .row = One
        .col = One
        .RowSel = .row
        .ColSel = .Cols - One
        For i = .row To .RowSel
            For J = .col To .ColSel
                If InStr(One, .TextMatrix(i, J), FindString, vbTextCompare) > Zero Then
                    CellCol = J
                    CellRow = i
                    FindinGrid = CellCol - One
                    Exit Function
                End If
            Next
        Next

        '\\   Search Col 1 construct labels ++

        .row = One
        .col = One
        .RowSel = .Rows - One
        .ColSel = .col
        For i = .row To .RowSel
            For J = .col To .ColSel
                If InStr(One, .TextMatrix(i, J), FindString, vbTextCompare) > Zero Then
                    CellCol = J
                    CellRow = i
                    FindinGrid = CellRow - One + NmE
                    Exit Function
                End If
            Next
        Next
    End If
```

```
    End With
    CellCol = Zero
    CellRow = Zero
    FindinGrid = Zero
End Function



Public Function GridSort(Optional ByRef ans As Long = -One)
    If ans = -One Then
        ans = Val(InputBox(IBS503_Enter_Sort_Method, IBS504_SortMethod_0to9, Three))
    End If
    UnResolve
    Dim SortRow As Long, SortRowSel As Long, SetRow As Long
    Dim SortCol As Long, SortColSel As Long, SetCol As Long
    SortRow = GridIn.Table.row
    SortRowSel = GridIn.Table.RowSel
    SortCol = GridIn.Table.col
    SortColSel = GridIn.Table.ColSel
    SetRow = SortRow
    SetCol = SortCol

    If SortRow = One And SortCol = One And SortRowSel = GridIn.Table.Rows - One And SortC
olSel = GridIn.Table.Cols - One Then
        SortCol = Zero
        SortColSel = Zero
    End If
    If SortRow = SortRowSel Then
        SortRow = Two
        SortRowSel = NmC + One
    End If
    GridIn.Table.row = lMax(2, SortRow)
    GridIn.Table.col = SortCol
    GridIn.Table.RowSel = lMin(NmC + One, SortRowSel)
    GridIn.Table.ColSel = SortColSel
    If ans = Three Then
        If SortColSel = One Then
            Dim i9 As Long, j9 As Long, workstring As String
            For i9 = One To NmC
                workstring = GridIn.Table.TextMatrix(One + i9, One)
                j9 = InStr(workstring, "_#")
                If j9 > Zero Then GridIn.Table.TextMatrix(One + i9, One) = Mid(workstring, j9 + T
hree) & "_#_" & Left(workstring, j9 - One)
            Next

        End If
```

```
    End If

    GridIn.Table.Sort = ans
    If ans = Three Then
        For j9 = One To NmE
            workstring = GridIn.Table.TextMatrix(One, One + j9)
            i9 = InStr(workstring, "_#")
            If i9 > Zero Then GridIn.Table.TextMatrix(One, One + j9) = Mid(workstring, i9 + Thre
e) & "_#_" & Left(workstring, i9 - One)
        Next
        If SortColSel = One Then
            GridIn.Table.row = One
            inGrid.CustomSort "N"
            For i9 = One To NmC
                workstring = GridIn.Table.TextMatrix(One + i9, One)
                j9 = InStr(workstring, "_#_")
                If j9 > Zero Then GridIn.Table.TextMatrix(One + i9, One) = Mid(workstring, j9 + T
hree)
            Next

            For j9 = One To NmE
                workstring = GridIn.Table.TextMatrix(One, One + j9)
                i9 = InStr(workstring, "_#_")
                If i9 > Zero Then GridIn.Table.TextMatrix(One, One + j9) = Mid(workstring, i9 + T
hree)
            Next
        End If

    End If
    SaveAs
    GridIn.Table.row = SetRow
    GridIn.Table.col = SetCol
    SetDirty True

End Function

Sub LoadURL(workz6 As String)
    Dim lWork As Long


    ShellExecute Zero, "open", workz6, Zero, Zero, One

    lWork = SendSound("websearch.wav", SND_SYNC + SND_NOSTOP)

End Sub
```

```
Sub UncompressING(ByRef gDoc As Long)    '\\  , IngridXMLFooter As String, Optional ByRef  .
DocText_SelStart + one As Long = ZERO) As String

   On Error GoTo errorline

   Dim lEndPos As Long    '\\  , .DocText_SelStart As Long
   With fDoc(gDoc)
      lEndPos = .DocText.Find(IngridXMLHeader, , , rtfNoHighlight)
      If lEndPos >= Zero Then
         '\\         If CreateXMLGrid(gDoc) Then WELLFORMED

         .DocText_SelStart = lEndPos + Len(IngridXMLHeader) + Two
         '\\             .DocText.SelLength = fDoc(gDoc).MeTextLength - .DocText_SelStart
         lEndPos = .DocText.Find(IngridXMLFooter, .DocText_SelStart, , rtfNoHighlight)
         .DocText.SelLength = lEndPos - .DocText_SelStart
         workbuffer = Left$(.DocText.SelText, Thirty)
         Dim lUncompressedLen As Long
         Dim lColonPosition As Long
         lUncompressedLen = Val(workbuffer)
         lColonPosition = Len(CStr(lUncompressedLen)) + One
         If Mid$(workbuffer, lColonPosition, One) = ":" Or Mid$(workbuffer, lColonPosition, One
) = ";" Then
            '\\               if next tag says that this is compressed Zlib then
            '\\               strings and flags must be set to redirect text stream to zlib output

            .DocText.Text = Left$(.DocText.Text, .DocText_SelStart) & UncompressString(.Doc
Text.SelText) & Mid$(.DocText.Text, lEndPos + One)
         End If
      End If
   End With
   Exit Sub
   Resume
errorline:       ' stop
   gCancel = True
End Sub


Sub ReplaceTextBoxXML(ByRef XMLString As RichTextBox, Optional ByVal ReplaceString As Stri
ng = vbNullString, Optional ByRef lPos As Long = Zero, Optional ByRef GridClip As String)
   On Error GoTo errorline
   Dim XMLstart As String, XMLend As String, ans As Long
   Dim lStart As Long, lLength As Long
   If ReplaceString = vbNullString Then
      XMLstart = "<inGridTable>" & vbCrLf
      XMLend = "</inGridTable>"
   Else
      ans = InStr(ReplaceString, ">") - One
```

```
        XMLstart = Left$(ReplaceString, ans)
        ans = InStrRev(ReplaceString, "<")
        '\\      do while Mid$(ReplaceString, ans, One) <> "<"
        '\\          ans = ans - One
        '\\      loop
        XMLend = Right$(ReplaceString, Len(ReplaceString) - ans + One)
        XMLend = Left$(XMLend, InStr(XMLend, ">"))
    End If
    lStart = XMLString.SelStart
    lLength = XMLString.SelLength
    XMLString.SelStart = XMLString.Find(XMLstart, lPos + One, , rtfNoHighlight)
    XMLString.SelLength = XMLString.Find(XMLend, XMLString.SelStart + One, , rtfNoHighlight)
- XMLString.SelStart
    If ReplaceString = vbNullString Then

        XMLString.SelText = XMLstart & GridClip
    Else
        XMLString.SelText = Left$(ReplaceString, ans - One)
    End If
    lPos = XMLString.SelStart + ans - One + Len(XMLend)
    XMLString.SelStart = lStart
    XMLString.SelLength = lLength
    Exit Sub

errorline:        ' stop
    '    gCancel = True
End Sub
Function ReplaceXML(ByRef XMLString As String, ReplaceString As String, Optional ByRef lPos
As Long = Zero) As String
    On Error GoTo errorline
    Dim XMLstart As String, XMLend As String, ans As Long

    ans = InStr(ReplaceString, ">") - One
    XMLstart = Left$(ReplaceString, ans)

'    ans = Len(ReplaceString)
'    do while Mid$(ReplaceString, ans, One) <> "<"
'        ans = ans - One
'    loop
'    XMLend = Right$(ReplaceString, Len(ReplaceString) - ans + One)
    ans = InStrRev(ReplaceString, "<") '20240825
    XMLend = Mid$(ReplaceString, ans)

    XMLend = Left$(XMLend, InStr(XMLend, ">"))
    lPos = InStr(lPos + One, XMLString, XMLstart, vbTextCompare) - One
    ReplaceXML = Left$(XMLString, lPos) & Left$(ReplaceString, ans - One) & XMLend & Mid$(X
MLString, InStr(lPos + One, XMLString, XMLend, vbTextCompare) + Len(XMLend))
```

```
    lPos = lPos + ans - One + Len(XMLend)
    Exit Function

errorline:        ' stop
    gCancel = True
End Function


Sub HideTable()
    If Not StartUpTypeIsPreview Then
        fTableLeft.Hide
        fTableRight.Hide
    End If

    '\\  fTableLeft.flex01.Visible = False
    fTableLeft.flex01.Redraw = False
    '\\  fTableRight.flex01.Visible = False
    fTableRight.flex01.Redraw = False
End Sub

Sub ParamsUp(ByRef Button As Integer)
    On Error Resume Next
    UnloadPopupMenuItems
    If Not (GridIn.Visible = False Or UserParams.Visible = False Or GridIn.WindowState = vbMini
mized Or UserParams.WindowState = vbMinimized) Then
        GridIn.WindowState = vbMinimized
        UserParams.WindowState = vbMinimized
        GridIn.Hide    '\\  WindowState = vbMinimized
        UserParams.Hide    '\\  WindowState = vbMinimized

        Exit Sub
    End If
    '\\  UnResolve
    ParamInit
    params
    GridIn.WindowState = vbNormal
    UserParams.WindowState = vbNormal
    grent1
    GridIn.Show vbModeless
    UserParams.Show vbModeless
    With UserParams

        If Button = vbRightButton Then
            .Visible = False
            '\\  if no range of significance bandwidths then signal message

            .EnableMDS.value = vbChecked
```

```
        PlotReset
        inGrid.SetSig = -Abs(inGrid.SetSig)
        SessionClear
        inGrid.SetAxisZ.value = inGrid.SetAxisZ.value + One
        inGrid.SetAxisZ.value = inGrid.SetAxisZ.value + One
        ListCopy
        .MDSFlexGrid1.row = One
        .MDSFlexGrid1.col = Five
        .MDSFlexGrid1.RowSel = .MDSFlexGrid1.Rows - One
        .MDSFlexGrid1.ColSel = Five
        .MDSFlexGrid1.Sort = Three
        Paramsort

    End If
  End With
End Sub
Sub Paramsort()
  Dim count As Long, ans As Double
  With UserParams.MDSFlexGrid1
    For count = Two To NmC + One
      GridIn.Table.TextMatrix(count, Zero) = ZeroText
    Next
    For count = .Rows - One To One Step -1
      ans = Int(Val(.TextMatrix(count, Six)))
      If ans > Zero Then
        GridIn.Table.TextMatrix(ans + One, Zero) = str(.Rows - count + Val(GridIn.Table.T
extMatrix(ans + One, Zero)))
      Else
        GridIn.Table.TextMatrix(Zero, -ans + One) = str(.Rows - count)
      End If
    Next

    '\\  sort grid both ways
    If inGrid.SetConstructs.list(inGrid.ConSlider.value) <> "OFF" Then
      For count = Two To NmC + One
        GridIn.Table.TextMatrix(count, Zero) = "zzzz"
      Next
      inGrid.mnuSortCustomA.Caption = Cap613_Construct
      GridIn.Table.col = Zero
    End If
    inGrid.mnuSortCustomA.Caption = Cap612_Element
    GridIn.Table.row = Zero
    inGrid.CustomSort "A"
    '\\  inGrid.mnuSortCustomA_Click
    dirty = True
    SessionClear
    UserParams.EnableMDS.value = Unchecked
```

```
        UserParams.Hide
        '\\  update grid
        '\\  inGrid.Components_Click
        grent1

    End With
End Sub




Public Sub SetTextEntry(ByRef frm As Form, Optional ByVal NotZipfDoc As Boolean = True)
    With frm
        .TextEntry.Text = .Table.Text
        .TextEntry.Height = .Table.CellHeight
        .TextEntry.Width = .Table.CellWidth
        If NotZipfDoc Then
            .Caption = Cap622_Grid_entry & .Table.TextMatrix(.Table.row, Zero) & Cap623_AND
& .Table.TextMatrix(Zero, .Table.col) & " = " & .TextEntry.Text
        End If
        .TextEntry.Top = .Table.CellTop + .Table.Top
        .TextEntry.Left = .Table.CellLeft + .Table.Left
        If .Table.row = .Table.Rows - One And .Table.col = One Then
            If StartPic = Zero Then
                StartPic = lMax(Zero, InStr(.TextEntry.Text, Chr$(31)) - One)
                .TextEntry.SelStart = StartPic
            Else
                .TextEntry.SelStart = lMax(StartPic, One)
            End If
            .TextEntry.SelLength = lMax(Zero, InStr(.TextEntry.Text, Chr$(31)) - One - .TextEntry.
SelStart)
        Else
            .TextEntry.SelStart = Zero
            .TextEntry.SelLength = Len(.TextEntry.Text)
        End If
        .TextEntry.Font.size = .Table.CellFontSize
        .TextEntry.Font.Bold = .Table.Font.Bold
        .TextEntry.FontName = .Table.CellFontName
        .Table.RowSel = .Table.row
        .Table.ColSel = .Table.col    '\\  s - One

    End With
End Sub

Sub ListCopy()
    On Error Resume Next
```

```
    Dim RowItem As Long, ColItem As Long
    Dim workstring As String

    workstring = vbNullString
'   ClipBoardClear
    If inGrid.SendSpeech.value = vbChecked Then SpeakThis UserParams.MDSFlexGrid1.Clip
    ClipBoardSetText UserParams.MDSFlexGrid1.Clip, False
    UserParams.MDSFlexGrid1.Rows = One

    snake = Zero
    For RowItem = One To NmC
        If WorkCon(RowItem, Two) > Zero Then
            snake = snake + One
            UserParams.MDSFlexGrid1.AddItem GridIn.Table.TextMatrix(One + RowItem, cLabelCo
l) & vbTab & conLabel(RowItem) & "." & vbTab & ASpace & vbTab & ASpace & vbTab & ASpac
e & vbTab & ASpace & str(WorkCon(RowItem, One) / WorkCon(RowItem, Two)) & vbTab & AS
pace & str(RowItem) & vbTab & ASpace & str(WorkCon(RowItem, Two)) & vbTab & ASpace &
str(WorkCon(RowItem, One))
        End If
    Next

    If inGrid.SetConstructs.list(inGrid.ConSlider.value) = "Split" Then
        For RowItem = One To NmC
            If WorkSplit(RowItem, Two) > Zero Then
                snake = snake + One
                UserParams.MDSFlexGrid1.AddItem GridIn.Table.TextMatrix(One + RowItem, cLabe
lCol) & vbTab & conLabel(-RowItem) & "." & vbTab & ASpace & vbTab & ASpace & vbTab & AS
pace & vbTab & ASpace & str(WorkSplit(RowItem, One) / WorkSplit(RowItem, Two)) & vbTab
& ASpace & str(RowItem + Half) & vbTab & ASpace & str(WorkSplit(RowItem, Two)) & vbTab
& ASpace & str(WorkSplit(RowItem, One))
            End If
        Next
    End If

    For ColItem = One To NmE
        If WorkEl(ColItem, Two) > Zero Then
            snake = snake + One
            UserParams.MDSFlexGrid1.AddItem GridIn.Table.TextMatrix(eLabelRow, One + ColIte
m) & vbTab & eLabel(ColItem) & ". " & vbTab & ASpace & vbTab & ASpace & vbTab & ASpace
& vbTab & ASpace & str(WorkEl(ColItem, One) / WorkEl(ColItem, Two)) & vbTab & ASpace & s
tr(-ColItem) & vbTab & ASpace & str(WorkEl(ColItem, Two)) & vbTab & ASpace & str(WorkEl(C
olItem, One))    '\\  getDescription( ColItem)
        End If
    Next
    UserParams.Update.SetFocus

End Sub
```

```
Public Function getPictureList() As String
   Dim Grid As Object
   If gDoc = MacroRefDoc And MacroRefDoc > Zero Then
      Set Grid = fDoc(gDoc).Table
   Else
      Set Grid = GridIn.Table
   End If
   '

   Static PictureList As String
   If LCase$(Grid.TextMatrix(Grid.Rows - Two, One)) = "select" Or Not SystemPictureListJustSe
t Then
      If PictureList = vbNullString Then
         PictureList = SettingsGet(iniName, RegSettings, "PictureList")
      End If
      If PictureList = vbNullString Then
         PictureList = Grid.TextMatrix(Grid.Rows - Two, One)

      End If

      getPictureList = PictureList
      SystemPictureListJustSet = True
   Else
      getPictureList = Grid.TextMatrix(Grid.Rows - Two, One)
   End If
   Set Grid = Nothing
End Function

Public Sub putExtraHTML(ByVal itemlink, ByVal workz6 As String, Optional ByVal itemtype = vb
NullString)

   If Len(itemtype) = Zero Then
      If itemlink > NmE Or itemlink < Zero Then
         itemtype = Construct
         If itemlink > NmE Then
            itemlink = itemlink - NmE
         Else
            itemlink = -itemlink - One
         End If
      Else
         itemtype = element
      End If
   End If
```

```
   Select Case itemtype
   Case element
      GridIn.Table.TextMatrix(NmC + Three, One + itemlink) = workz6
   Case Construct
      GridIn.Table.TextMatrix(One + itemlink, LastDataCol + Three) = workz6
   Case Else
      Error = 13
      WarningError Err, "Casing"
   End Select
End Sub

Public Sub putDescription(ByVal itemlink, ByVal workz6 As String, Optional ByVal itemtype = vb
NullString)
   If Len(itemtype) = Zero Then
      If itemlink > NmE Or itemlink < Zero Then
         itemtype = Construct
         If itemlink > NmE Then
            itemlink = itemlink - NmE
         Else
            itemlink = -itemlink - One
         End If
      Else
         itemtype = element
      End If
   End If

   Select Case itemtype
   Case element
      GridIn.Table.TextMatrix(NmC + Two, One + itemlink) = workz6
   Case Construct
      GridIn.Table.TextMatrix(One + itemlink, LastDataCol + Two) = workz6
   Case Else
      Error = 13
      WarningError Err, "Casing"
   End Select
End Sub

Public Function getDescription(ByVal itemlink As Long, Optional ByVal itemtype As String = vbN
ullString) As String
   Dim workz7 As String
   workz7 = itemtype
   If workz7 = "Desc" Then
      itemtype = vbNullString
   Else
      workz7 = vbNullString
   End If
   If Len(itemtype) = Zero Then
```

```
    If itemlink > NmE Or itemlink < Zero Then
        itemtype = workz7 & Construct
        If itemlink > NmE Then
            itemlink = itemlink - NmE
        Else
            itemlink = -itemlink - One
        End If
    Else
        itemtype = workz7 & element
    End If
End If
Select Case itemtype
Case OneText
    getDescription = GridIn.Table.TextMatrix(NmC + Two, One + itemlink)
    If Len(getDescription) = Zero Then
        getDescription = Trim$(GridIn.Table.TextMatrix(eLabelRow, One + itemlink))
    End If
Case ZeroText
    getDescription = GridIn.Table.TextMatrix(One + Abs(itemlink), LastDataCol + Two)
    If Len(getDescription) = Zero Then
        getDescription = Trim$(GridIn.Table.TextMatrix(One + Abs(itemlink), cLabelCol))
    End If
Case "Desc1"
    getDescription = ElementDesc(itemlink)
Case "Desc0"
    getDescription = ConstructDesc(itemlink)
Case "Popup"
    Select Case itemlink
    Case Is > NmE
        Static ConstructName As String
        getDescription = conLabel(itemlink - NmE, "label")
        If ConstructName = vbNullString Then
            ConstructName = ConstructTitle
            If ConstructName = vbNullString Then ConstructName = Cap613_Construct
        End If
        workz7 = ConstructName
    Case Else
        Static ElementName As String
        getDescription = eLabel(itemlink, "label")
        If ElementName = vbNullString Then
            ElementName = ElementTitle
            If ElementName = vbNullString Then ElementName = Cap612_Element
        End If
        workz7 = ElementName
    End Select
    If Right$(getDescription, 4) = ".ing" Then
        inGrid.mnuPopupItemLoad(itemlink).Caption = getDescription
```

```
        Else
            inGrid.mnuPopupItemLoad(itemlink).Caption = workz7 & str(itemlink) & ASpace & get
Description
            GridIn.Table.CellBackColor = GridOut.Picture2.ForeColor
        End If

    Case Else
        Error = 13
        WarningError Err, "Casing"
    End Select
End Function


Public Sub putPictureList(ByVal workz6 As String)
    GridIn.Table.TextMatrix(NmC + Two, One) = workz6
    '\\      SetDirty True
End Sub
Public Sub HTMLExtras()
    On Error GoTo errorline

    GridIn.Table.Rows = NmC + Four
    GridIn.Table.Cols = LastDataCol + Four
    GridIn.Table.TextMatrix(NmC + 3, Zero) = "ExtraHTML and Action Settings for"
    GridIn.Table.TextMatrix(Zero, LastDataCol + Three) = "ExtraHTML and Action Settings for"

    Exit Sub
errorline:        ' stop
    WarningError Err, "Casing"
End Sub

Sub ScoreSwap()
    Dim ColItem As Single
    Dim workz6 As String
    Dim workz7 As String

    If T9 <> K1 And T9 = B9 Then
        UserParams.WorstScore.Text = str(K1)
        If MsgBoxEx(Msg224_swap_descriptions, vbYesNo) = vbYes Then
            For ColItem = One To NmC
                workz6 = Trim$(Left$(GridIn.Table.TextMatrix(One + ColItem, LastDataCol + Two),
 InStr(GridIn.Table.TextMatrix(One + ColItem, LastDataCol + Two) & "/", "/") - One))
                workz7 = Trim$(Right$(GridIn.Table.TextMatrix(One + ColItem, LastDataCol + Two
), Len(GridIn.Table.TextMatrix(One + ColItem, LastDataCol + Two)) - InStr(GridIn.Table.TextM
atrix(One + ColItem, LastDataCol + Two), "/")))
                If workz6 <> workz7 Then
                    GridIn.Table.TextMatrix(One + ColItem, LastDataCol + Two) = workz7 & "/" & w
orkz6
```

```
            End If
        Next
        K1 = Zero
        For ColItem = One To NmE
            workz6 = Trim$(Left$(getDescription(ColItem), InStr(getDescription(ColItem) & "/",
 "/") - One))
            workz7 = Trim$(Right$(getDescription(ColItem), Len(getDescription(ColItem)) - InS
tr(getDescription(ColItem), "/")))
            If workz6 <> workz7 Then
                If K1 = Zero Then
                    K1 = MsgBoxEx(Msg225_same_for_elements, vbYesNo)
                End If
                If K1 = vbYes Then putDescription ColItem, workz7 & "/" & workz6
            End If
        Next
    End If
  End If
End Sub

Sub SessionClear()
  UnResolve
  UserParams.MDSFlexGrid1.Rows = One

  UserParams.MDSFlexGrid1.row = Zero
  UserParams.MDSFlexGrid1.col = Zero
  UserParams.MDSFlexGrid1.RowSel = Zero
  UserParams.MDSFlexGrid1.ColSel = Zero

  Erase WorkCon, WorkEl
  ReDim WorkCon(NmC, Two), WorkEl(NmE, Two)
  snake = One
  UserParams.MDSFlexGrid1.Clip = ASpace
  On Local Error Resume Next
  UserParams.Update.SetFocus
  UserParams.MDSFlexGrid1.Rows = 3
End Sub

Sub setTitleTag(ByRef Grid As Object, ByVal Tag As String, ByVal TAGValue As String)
  Grid.TextMatrix(Grid.Rows - One, Grid.Cols - One) = ReplaceHTML(Grid.TextMatrix(Grid.Row
s - One, Grid.Cols - One), Tag, TAGValue)

End Sub
Sub setUserTagOptional(ByRef FTP As Form, ByVal Tag As String, ByVal TAGValue As String, O
ptional ByVal AltTag As String = vbNullString)
  Dim Grid As Object
  If Val(FTP.Caption) > Zero Then
    Set Grid = fDoc(Val(FTP.Caption)).Table
```

```
    Else
        Set Grid = GridIn.Table
    End If
    Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One) = ReplaceHTML(Grid.TextMatrix(Grid.Row
s - Two, Grid.Cols - One), Tag, TAGValue, AltTag)
    Set Grid = Nothing
End Sub

Sub setUserTag(ByRef Grid As Object, ByVal Tag As String, ByVal TAGValue As String, Optional
ByVal AltTag As String = vbNullString)
    Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One) = ReplaceHTML(Grid.TextMatrix(Grid.Row
s - Two, Grid.Cols - One), Tag, TAGValue, AltTag)

End Sub
Sub setTemplateTag(ByVal Tag As String, ByVal TAGValue As String)
    Dim Grid As Object
    If gDoc = MacroRefDoc And MacroRefDoc > Zero Then
        Set Grid = fDoc(gDoc).Table
    Else
        Set Grid = GridIn.Table
    End If
    Grid.TextMatrix(One, One) = ReplaceHTML(Grid.TextMatrix(One, One), Tag, TAGValue)
    Set Grid = Nothing

End Sub

Function getTemplateTag(ByVal Tag As String) As String
    Dim Grid As Object
    If gDoc = MacroRefDoc And MacroRefDoc > Zero Then
        Set Grid = fDoc(gDoc).Table
    Else
        Set Grid = GridIn.Table
    End If
    getTemplateTag = ExtractHTML(Grid.TextMatrix(One, One), Tag)
    Set Grid = Nothing

End Function
Function getUserTag(ByVal Tag As String, Optional ByVal Default As String = vbNullString, Opti
onal ByVal AltTag As String = vbNullString) As String
    If PF_Ending Then Exit Function
    If Len(AltTag) = Zero Then
        AltTag = Trim$(Left$(Tag, Len(Tag) - One)) & " ="
    End If
    getUserTag = ExtractHTML(GridIn.Table.TextMatrix(NmC + Two, LastDataCol + Three), Tag
, Default, AltTag)

End Function
```

modGridIn - 67

```
Function getTitleTag(ByVal Tag As String, Optional ByVal Default As String = vbNullString, Opti
onal ByVal AltTag As String = vbNullString) As String
    Dim Grid As Object
    If gDoc = MacroRefDoc And MacroRefDoc > Zero Then
        Set Grid = fDoc(gDoc).Table
    Else
        Set Grid = GridIn.Table
    End If
    If Len(AltTag) = Zero Then
        AltTag = Trim$(Left$(Tag, Len(Tag) - One)) & " ="
    End If
    getTitleTag = ExtractHTML(Grid.TextMatrix(Grid.Rows - One, Grid.Cols - One), Tag, Default,
AltTag)
    Set Grid = Nothing

End Function

Public Function inGridUser(ByRef gDoc As Long, Optional ByVal workz6 As String = vbNullString
, Optional ByRef Grid As Object = Nothing) As String
    If PF_Ending Then Exit Function
    If Grid Is Nothing Then
        Set Grid = GridIn.Table
    End If

    If Len(workz6) = Zero Then
        inGridUser = SplitGridUser(gDoc, Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One))
        If Right$(inGridUser, One) = ">" Then
            inGridUser = Left$(inGridUser, Len(inGridUser) - One)
        End If
    Else
        If workz6 = "_" Then
            workz6 = Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One)
            If workz6 = vbNullString Then workz6 = "_"    '\\  this should replicate FTP settings
        End If
        inGridUser = SplitGridUser(gDoc, workz6)
    End If
    If Len(FTPs(gDoc).tUser) <> Zero Or Len(FTPs(gDoc).tServer) <> Zero Or Len(FTPs(gDoc).
tRemoteFile(Zero).Text) <> Zero Then
        If inGridUser <> FTPs(gDoc).tUser Then
            Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One) = inGridUser & " _server=""" & FTP
s(gDoc).tServer & """ username=""" & FTPs(gDoc).tUser & """ directory=""" & FTPs(gDoc).tRe
moteFile(Zero) & """>"
        Else
            Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One) = "_server=""" & FTPs(gDoc).tServ
er & """ username=""" & FTPs(gDoc).tUser & """ directory=""" & FTPs(gDoc).tRemoteFile(Zero)
 & """>"
        End If
```

```
   Else
      Grid.TextMatrix(Grid.Rows - Two, Grid.Cols - One) = inGridUser
   End If
End Function
Sub TableCopy()
   If gAns = Zero Then
      fTableLeft.flex01.row = Zero
      fTableLeft.flex01.col = Zero
      fTableLeft.flex01.RowSel = fTableLeft.flex01.Rows - One
      fTableLeft.flex01.ColSel = fTableLeft.flex01.Cols - One
      ClipBoardSetText fTableLeft.flex01.Clip, False
   Else
      fTableRight.flex01.row = Zero
      fTableRight.flex01.col = Zero
      fTableRight.flex01.RowSel = fTableRight.flex01.Rows - One
      fTableRight.flex01.ColSel = fTableRight.flex01.Cols - One
      ClipBoardSetText fTableRight.flex01.Clip, False
   End If

   fTableLeft.flex01.RowSel = Zero
   fTableRight.flex01.RowSel = Zero
   fTableLeft.flex01.ColSel = Zero
   fTableRight.flex01.ColSel = Zero
End Sub
Sub GridinTextEntry()
   SetTextEntry GridIn
   With GridIn
      If .Check1.Visible = False Then
         If .TextEntry.FontName = "Wingdings" Then
            .Check1.Left = .Table.CellLeft + .Table.Left
            .Check1.Top = .TextEntry.Top
            .Check1.Visible = True
            If .Check1.Enabled = True Then
               If .TextEntry.Text = "p" Then
                  .TextEntry.Text = "u"
                  .Check1 = Checked
               Else
                  .TextEntry.Text = "p"
                  .Check1 = Unchecked
               End If
               .Check1.SetFocus

            Else
               If .TextEntry.Text = "p" Then
                  .Check1 = Unchecked
               ElseIf .TextEntry.Text = "s" Then
                  .Check1 = vbGrayed
```

```
                Else
                    .Check1 = Checked
                End If

                .Check1.Enabled = True
                .Check1.SetFocus

            End If

        Else
            '\\  .Check1.Visible = False
        End If
    End If
  End With
End Sub
Sub SetGridHead()
  GridIn.Table.TextMatrix(NmC + Three, LastDataCol + Two) = GridDate & "," & ViewType &
"," & NormalCode & "," & WeightCol
End Sub
Public Function getPictureClip() As String
  If inGridVER(Zero, gDoc) < 12.9 Then
    getPictureClip = ExtractHTML(GridIn.Table.TextMatrix(NmC + Two, LastDataCol + Two),
TAGImgSrc)
    '\\  inGridVER(zero,gDoc) = 12.9
    GridIn.Table.TextMatrix(NmC + Two, LastDataCol + Two) = vbNullString
    JPEGs(gDoc).tiImgSrc.Text = getPictureClip
  Else
    getPictureClip = getTemplateTag(TAGImgSrc)
  End If
End Function

Function ReadOnlyStatus(Optional FullTest As Boolean = False) As Boolean
  ReadOnlyStatus = inGridVER(Two, gDoc)

  If FullTest Then
    ReadOnlyStatus = ReadOnlyStatus And Not (InStr(One, FullGridName, "tp://", vbTextCom
pare) > Zero Or InStr(One, FullGridName, ".htm", vbTextCompare) > Zero) And inGridVER(Two
, gDoc) <> Two
  End If
End Function


Sub ContinueMRU(ByVal MRU As Integer)

  UnloadAnyLayers
  Dim lPos As Long
  lPos = Zero
```

```
   GridOut.Caption = exitmode("ShowPath")
   If Not SlaveBot Then
      Do While InStr(LCase$(inGrid.mnuFileMRU(MRU).Caption), "copy of") > Zero
         MRU = MRU + One
      Loop
      If InStr(LCase$(inGrid.mnuFileMRU(One).Caption), "chess3") > Zero And m_bShowChess
Then
         '\\             MigoyaDoc = gDoc
         FileMRU MRU
         Exit Sub
      End If
      If InStr(LCase$(inGrid.mnuFileMRU(One).Caption), "copy of") > Zero Then  '\\  Or (gDoc
> one And (Abs(CycleZ) > Half Or inGrid_mnuViewAutoRedraw_Checked)) Then
         If MRU = One And inGridVER(Four, gDoc) > Hundred * Thousand Then
            lPos = MsgBoxEx("Continue running Most Recent grid." & vbCr & "This will stop Ing
rid.scr (screensaver) from launching", vbYesNoCancel)
         Else
            lPos = vbYes
         End If

         If lPos = vbCancel Then Exit Sub

         If lPos = vbYes Then
            SetDirty True    '\\  why?
            If ReadDoc = Zero Then
               ReadDoc = gDoc    '\\  this is so font selection works
               '\\                    SetFont
            End If
         End If
      End If
      If lPos = vbYes Or lPos = Zero Then
         lPos = gDoc
         FileMRU MRU
         If MRU = One And Analysed Then spin    '\\   difference between loading and pasting r
egards Unresolve

      End If
   End If

End Sub

Function FileLoad(ByVal file As Long, ByVal FileName As String, Optional ByVal AbortMsg As Stri
ng) As Boolean
   Dim lWork As Long
   On Error GoTo errorline
   gCancel = False
   FileLoad = True    '\\   inGridVer(two,gDoc) = true
```

```
   If InStr(One, FileName, "tp://", vbTextCompare) Then
      '\\  If Not GetConnected Then
      '\\      Cancel = True
      '\\      Exit Sub
      '\\  End If
      fDoc(file).DocText.Text = vbNullString
      fDoc(file).DocText.Text = WSAPI_OpenURL(FileName)
      If fDoc(file).MeTextLength = Zero Then
         lWork = Screen_MousePointer(, vbDefault)

         gCancel = True
         MsgBoxEx Msg147_network_error & vbCrLf & AbortMsg
         Exit Function

      End If
   Else
      FileName = FilenameEx(FileName)
      If Not FileExists(FileName) Then
         gCancel = True
         Exit Function
      End If
      If Right(FileName, Four) = ".rtf" Then
         fDoc(file).DocText.LoadFile FilenameEx(FileName)    '\\  , rtfText
      Else
         fDoc(file).DocText.LoadFile FilenameEx(FileName), rtfText
      End If
      SetDirty False, file
      If InStr(One, FileName, ".ing", vbTextCompare) Then FileLoad = False    '\\  ReadOnlySta
tus
   End If
   Exit Function
   Resume
errorline:        ' stop
   gCancel = True

End Function


Function GetToolbox(Optional ControlsNew As String = vbNullString, Optional ByVal delim As St
ring = vbTab) As Boolean
   On Error GoTo errorline
   GetToolbox = True
   PerturbateOff = KarmaGunAfterStartup ' False
   Static Gettoolboxcount As Long
   Gettoolboxcount = Gettoolboxcount + One
   If Gettoolboxcount > One Then GoTo errorline
   Dim values(14) As Single, VerNUM As Single, lPos As Long, i As Long    '\\  , lWork As Long
```

```
   lPos = InStr(ControlsNew, delim)
   If lPos > Zero And IngridLoaded Then
      If ControlsNew = vbTab & vbCr Then
         GetToolbox = False
         Gettoolboxcount = Zero
         Exit Function    '\\   no toolbox in Registry SettingSaves
      End If
      If ControlsNew <> vbTab Then
         If Not SplitTools(ControlsNew, delim) Then
            GetToolbox = False
            Gettoolboxcount = Zero
            Exit Function    '\\   vbtab is a macro flag
         End If
      End If
      If Tools(One) >= 15.7 Then
         i = One
      Else
         i = Zero
         Do While lPos
            Mid$(ControlsNew, lPos, One) = ASpace
            lPos = InStr(lPos, ControlsNew, vbTab)
         Loop
         If Len(ControlsNew) <> Zero Then workbuffer = Trim$(ControlsNew)
      End If
   Else
      GetToolbox = False
      Gettoolboxcount = Zero
      Exit Function    '\\   no toolbox in Registry SettingSaves
   End If
   '\\  Split WorkBuffer, vbnullchar, t()

   With inGrid
      '\\  If vernum >= 9.79 Then Input #one, fore(ColorSwitch,1), fore(ColorSwitch,2), fore(C
olorSwitch,3), fore(ColorSwitch,4), fore(ColorSwitch,5), fore(ColorSwitch,6)
      '\\  Input #ONE,
      Disable
      values(One) = NextNumber(i)
      If values(One) >= 13.7 Then
         VerNUM = values(One)
         If VerNUM > 13.92 And VerNUM < 15.7 Then
            TextValues(Five) = NEXTWORD(i)
         End If
         values(One) = NextNumber(i)
      Else
         VerNUM = inGridVER(Zero, nDoc)
      End If
```

```
        values(Two) = NextNumber(i)
        values(Three) = NextNumber(i)
        If VerNUM < 15.11 Then values(Four) = NextNumber(i)
        values(Five) = NextNumber(i)
        If VerNUM < 15.12 Then values(Six) = NextNumber(i)
        If VerNUM > 15.13 Then values(Six) = NextNumber(i)
        values(Seven) = NextNumber(i)
        values(Eight) = NextNumber(i)
        values(Nine) = NextNumber(i)
        TextValues(Nine) = NEXTWORD(i)
        If Not PerturbateOff Then
            .BackToFront.value = values(One)
            .UpsideDown.value = values(Two)
            .InsideOut.value = values(Three)
            '\\  .SigLines.Value = values(Four)
            inGrid_Cycler_Value = values(Five)
            If VerNUM > 15.13 Then
                '\\  .Off.BackColor = values(Six)

                '\\  inGrid.Off.Picture = Nothing
                '\\  If inGrid.Off.BackColor = vbblack Then
                '\\      inGrid.Off.Picture ,18)
                '\\  Else
                '\\      inGrid.Off.Picture ,20)
                '\\  End If
            End If
            .Off.value = values(Seven)
            .JetTrails.value = values(Eight)
            .CenterLines.value = values(Nine)
            '\\  dummy (was user) .ForDoevents.Value = values(ten)
        End If
        If VerNUM >= 9.73 Then
            '\\  Input #ONE, values(ONE), values(TWO), values(THREE), values(FOUR), values(fiv
e), values(six), values(seven), values(eight)
            '\\  WorkBuffer = StripInputEx
            values(One) = NextNumber(i)
            values(Two) = NextNumber(i)
            values(Three) = NextNumber(i)
            values(Four) = NextNumber(i)
            values(Five) = NextNumber(i)
            values(Six) = NextNumber(i)
            values(Seven) = NextNumber(i)
            values(Eight) = NextNumber(i)
            If Not PerturbateOff Then

                .BrushFill.value = Val(TextValues(Nine))
                .Outlines.value = values(One)
```

```
            .ScopeOff.value = values(Two)
            .ClipOff.value = values(Three)
            If inGrid_ForDoEvents_BackColor <> vbBlack Then
               .ForDoevents.value = values(Four)
               inGrid_ForDoevents_Value = .ForDoevents.value
            End If
            .Transparent.value = values(Five)
            .DrawState.value = values(Six)
            .SendSpeech.value = values(Seven)
            .BlankPictures.value = values(Eight)

         End If
      End If
      TextValues(Two) = NEXTWORD(i)  '\\  stripinput(ZERO)
      TextValues(Four) = NEXTWORD(i)  '\\  stripinput(ZERO)
      '\\  Input #ONE, QuadBottom, QuadRight, QuadTop, QuadLeft, GridFont
      '\\  WorkBuffer = StripInputEx
      values(11) = NextNumber(i)   '\\  QuadBottom = must be set after Viewsize but before F
ullscreen check
      values(12) = NextNumber(i)  '\\  QuadRight =
      values(13) = NextNumber(i)  '\\  QuadTop =
      values(14) = NextNumber(i)  '\\  QuadLeft =
      If VerNUM < 14.11 Then
         values(One) = NextNumber(i)
         If Not PerturbateOff Then GridFont = values(One)
      End If
      If Not PerturbateOff Then
         '\\  .SetConstructs.text = TextValues(Two)
         .ConSlider.value = SendMessageStr(.SetConstructs.hwnd, CB_FINDSTRING, -1, ByVal
TextValues(Two))  '\\  .SetConstructs.ListIndex
         '\\  .SetElements.list(inGrid.ElSlider.Value) = TextValues(Four)
         .ElSlider.value = SendMessageStr(.SetElements.hwnd, CB_FINDSTRING, -1, ByVal Text
Values(Four))  '\\  .SetElements.ListIndex
      End If
      If VerNUM <= 9.73 Then
         '\\  Input #ONE, values(ONE), values(TWO), values(FOUR), values(five), values(six), v
alues(seven), values(eight), values(nine), values(ten)
         '\\  WorkBuffer = StripInputEx
         values(One) = NextNumber(i)
         values(Two) = NextNumber(i)
         values(Four) = NextNumber(i)
         values(Five) = NextNumber(i)
         values(Six) = NextNumber(i)
         values(Seven) = NextNumber(i)
         values(Eight) = NextNumber(i)
         values(Nine) = NextNumber(i)
         values(Ten) = NextNumber(i)
```

```
            '\\  .Consensus.Value = values(one)
            '\\  .Wizard.Value = values(two)
            '\\  .Pause.Value = values(four)
            '\\  .ResetPlot.Value = values(five)
            '\\  .NowRefresh.Value = values(six)
            '\\  .Similarities.Value = values(seven)
            '\\  .Components.Value = values(eight)
            '\\  .Contributions.Value = values(nine)
            '\\  .SpinGridX.Value = values(ten)
        End If
        '\\  Input #ONE, values(ONE)
        '\\  WorkBuffer = StripInputEx
        values(One) = NextNumber(i)

        TextValues(One) = NEXTWORD(i)  '\\  stripinput(ZERO)
        '\\  Input #ONE, values(THREE), values(FOUR), values(five), values(six), values(seven), v
alues(eight)
        '\\  WorkBuffer = StripInputEx
        values(Three) = NextNumber(i)
        values(Four) = NextNumber(i)
        values(Five) = NextNumber(i)
        values(Six) = NextNumber(i)
        values(Seven) = NextNumber(i)
        values(Eight) = NextNumber(i)

        If Not PerturbateOff Then

            .ElementBackColor.value = values(One)
            If VerNUM >= 9.71 Then
                '\\  .SetElements.ListIndex = Zero
                '\\  .SetElements.ListIndex = Abs(Val(TextValues(one)))
            End If    '\\  was .HScroll2.Value = values(two)
            .ConstructForeColor.value = values(Three)
            .SmallChanges.value = values(Four)
            .ElementForeColor.value = values(Five)
            .ConstructBackColor.value = values(Six)
'           Stop
            .GridBackColor.value = values(Seven)
            .GridForeColor.value = values(Eight)

        End If
        .SetAxisX.Max = DEG * lMax(Two, N7)
        .SetAxisX.Min = Zero
        .SetAxisY.Max = DEG * lMax(Three, N7)
        .SetAxisY.Min = Zero
        values(One) = N7
        If VerNUM >= 13.7 Then
```

```
        values(One) = NextNumber(i)  '\\   N7 = number of components
    End If
    If VerNUM >= 13.6 Then
        values(Three) = lMax(DEG, NextNumber(i))
        values(Two) = NextNumber(i)
        If Not PerturbateOff Then
            If values(Three) < .SetAxisZ.Max Then
                .SetAxisZ.Min = values(Three)
            Else
                .SetAxisZ.Min = .SetAxisZ.Max - One
            End If
            If values(Two) < Zero Or values(Two) > maxInt Then
                MsgBoxEx "setZaxis.max fatal error condition"
                GetToolbox = False
                Gettoolboxcount = Zero
                Exit Function
            End If
            .SetAxisZ.Max = values(Two)
            If values(Two) > .SetAxisZ.Min Then
                .SetAxisZ.Max = sMin(values(Two), DEG * (lMax(Three, N7) + One) - One)
            Else
                .SetAxisZ.Max = .SetAxisZ.Min + One
            End If
        End If
    ElseIf Not PerturbateOff Then
        .SetAxisZ.Max = DEG * sMin(N7 + One, values(One)) - One    '\\  lMin(NmE - One, Nm
C - One))
        .SetAxisZ.Min = 1080
    End If
    If VerNUM >= 13.8 Then
        values(One) = NextNumber(i)
        values(Two) = NextNumber(i)
        If Not PerturbateOff Then
            .ZaxisLabel.ForeColor = values(One)
            .ZaxisLabel.BackColor = values(Two)
        End If
    End If
    '\\  Input #ONE, values(ONE), values(TWO), values(THREE), values(FOUR)
    '\\  WorkBuffer = StripInputEx
    values(One) = NextNumber(i)
    values(Two) = NextNumber(i)
    values(Three) = NextNumber(i)
    values(Four) = NextNumber(i)
    TextValues(One) = NEXTWORD(i)  '\\  WorkBuffer

    IngridMixTCP = NEXTWORD(i)
```

```
        values(Seven) = NextNumber(i)
        values(Eight) = NextNumber(i)
        If Right$(IngridMixTCP, Three) = "bpm" Then
            If dmDrumsLoaded Then

                If dmDrums.LabelAlign.ForeColor <> &H80FF80 Then  'light green = manually selec
ted forced sever of beatmix
                    GetToolbox = Not MixingThis
                    If bEnqueuedTrackWaiting Then
                        If dmDrums.LabelBPM_ForeColor > vbRed Then
                            MouseUp_LabelBPM vbMiddleButton
                        Else
                            bEnqueuedTrackWaiting = bEnqueuedTrackWaiting
                        End If
'                           dmDrums.LabelBPM_ForeColor = Int(vbRed / Two)
                    End If
                End If
'               If ListAdvanceColor = vbGreen And GetToolbox Then
''                  ListAdvanceColor = vbRed
''                  PlaylistAdvance True ' stops below setting trackselected to false in ok condition
'                   fHIDJ.IngridMixSync
'               End If
            End If
        End If
        If Not PerturbateOff Then
            .SetAxisX.value = values(One)
            .SetAxisY.value = values(Two)
            .SetAxisZ.value = sMax(sMin(values(Three), .SetAxisZ.Max), .SetAxisZ.Min)
            xaxis = .SetAxisX.value / DEG
            yaxis = .SetAxisY.value / DEG
            zaxis = .SetAxisZ.value / DEG
            .SetSig.value = values(Four)
            If VerNUM >= 9.72 Then
                .OpCode.ListIndex = Val(TextValues(One))
                .OpCodeSlider.value = .OpCode.ListIndex
            End If    '\\   was .Blending.Value = values(five)
            .TimeStep.value = Val(IngridMixTCP)


            .Updown.value = values(Seven)
            .Sideways.value = values(Eight)
        End If

        '\\  Input #ONE, values(ONE), values(TWO), values(THREE), values(FOUR)
        '\\  WorkBuffer = StripInputEx
        values(One) = NextNumber(i)
        values(Two) = NextNumber(i)
```

```
    values(Nine) = NextNumber(i)       '\\   ViewSize must be set after FullScreen check
    values(Five) = NextNumber(i)       '\\   fontscroll set further on down after viewsize but b
efore pointsize
    If Not PerturbateOff Then
       .Zoom.value = values(One)
       .BandSize.value = values(Two)
    End If
    '\\  .FontScroll.Value = values(Four)
    If VerNUM >= 9.51 Then
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       If Not PerturbateOff Then CycleZ = Sgn(values(One)) * CycleZ
       '\\  Input #ONE, CycleZ
    End If
    If VerNUM >= 9.54 Then
       '\\  Input #ONE, values(ONE)
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       If Not PerturbateOff Then
          .ColorLines.value = values(One)
          inGridClick_ColorLines
       End If

    End If
    If VerNUM >= 9.8 Then
       '\\  Input #ONE, values(ONE)
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       If Not PerturbateOff Then
          .DisplayTables.value = values(One)
          inGrid.mnuViewTable.Checked = values(One)
       End If
    End If
    If VerNUM >= 9.83 Then
       '\\  Input #ONE, values(ONE), values(TWO), values(THREE), values(FOUR)
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       values(Two) = NextNumber(i)
       values(Three) = NextNumber(i)
       values(Four) = NextNumber(i)
       If Not PerturbateOff Then
          .Layer(element).value = values(One)
          .Layer(Construct).value = values(Two)
          .PictureStretch.value = values(Three)
          .PictureSize.value = values(Four)
       End If
    End If
```

```
    If VerNUM >= 9.84 Then
       '\\  Input #ONE, values(ONE)
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       If Not PerturbateOff Then .AutoChanger.value = values(One)
    End If
    If VerNUM >= 9.85 Then
       '\\  Input #ONE, values(ONE)
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       If Not PerturbateOff Then .Blending.value = values(One)
    End If
    If VerNUM >= 9.55 Then
       '\\  Input #ONE, AutoZoom
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       If Not PerturbateOff Then
          AutoZoom = values(One)
       End If
    End If
    TextValues(One) = vbNullString
    TextValues(Two) = vbNullString
    TextValues(Three) = vbNullString
    If VerNUM >= 9.6 Then
       '\\  Input #ONE, values(ONE), values(TWO)
       '\\  WorkBuffer = StripInputEx
       values(One) = NextNumber(i)
       values(Two) = NextNumber(i)
       If VerNUM < 14.1 Then
          TextValues(One) = NEXTWORD(i) & ASpace & NEXTWORD(i)   '\\  stripinput(ZERO)
          TextValues(Two) = NEXTWORD(i)  '\\  stripinput(ZERO)
          TextValues(Three) = Trim$(NEXTWORD(i) & ASpace & NEXTWORD(i))
       End If
       If Not PerturbateOff Then
          .Mode.value = values(One)
          .StyleWidth.value = values(Two)
       End If
    End If


    If VerNUM >= 9.61 Then
       '\\  Input #ONE, PointSize, ForeSize, BackSize, sized
       '\\  WorkBuffer = StripInputEx
       If VerNUM >= 10.103 Then
          values(One) = NextNumber(i)
          TextValues(Four) = Trim$(NEXTWORD(i)) 'looks like a throwaway
          If Not PerturbateOff Then
```

```
                .FontInt = values(One)
                '\\  inGrid.mnuFullScreen.Checked = NextNumber(i)

                '\\  If inGrid.mnuFullScreen.Checked = False Then
                '\\      GridOut.WindowState = vbNormal
                '\\  Else
                '\\      GridOut.WindowState = vbMaximized
                '\\
                '\\  End If
                '\\  GridOutResize_Form
                If StartUpTypeIsNormal Then

                   If .ViewSize.Enabled = False Then .ViewSize.Enabled = True               '\\  onl
y set for first grid loaded
                End If
                GridOut.ScaleMode = vbCharacters
                GridOut.Picture2.ScaleMode = vbCharacters
                '\\  If Not StartUpTypeIsNormal   Then
                '\\      Pic2Row = GridOut.Picture2.ScaleY(GridOut.Picture2.Height, vbTwips, vbC
haracters)
                '\\      Pic2Col = GridOut.Picture2.ScaleX(GridOut.Picture2.Width, vbTwips, vbCha
racters)
                '\\  Else
                Pic2Row = GridOut.Picture2.ScaleHeight
                Pic2Col = GridOut.Picture2.ScaleWidth
                '\\  End If

                If VerNUM > 12.8 Then

                   QuadBottom = sMin(Half, values(11)) * Pic2Row
                   QuadRight = sMin(Half, values(12)) * Pic2Col
                   QuadTop = values(13) * Pic2Row
                   If QuadTop > QuadBottom Then QuadTop = 3

                   QuadLeft = values(14) * Pic2Col
                   If QuadLeft > QuadRight Then QuadLeft = 8
                Else
                   QuadTop = values(13)    '\\  QuadTop =
                   QuadLeft = values(14)    '\\  QuadLeft =
                   QuadBottom = sMin(Pic2Row - QuadTop, values(11))    '\\  QuadBottom = mu
st be set after Viewsize and but before Fullscreen check
                   QuadRight = sMin(Pic2Col - QuadLeft, values(12))    '\\  QuadRight =

                   Pic2Row = QuadBottom
                   Pic2Col = QuadRight
                End If
                GridOut.ScaleMode = vbTwips
```

```
        End If

        '\\  call ShowMe(Gridout)

        If VerNUM >= 10.105 Then    '\\   not used anymore
           values(One) = NextNumber(i)
           values(Two) = NextNumber(i)
           values(Three) = NextNumber(i)
           If Not PerturbateOff Then
              .Beginner = values(One)
              .Intermediate = values(Two)
              .Advanced = values(Three)
           End If
        End If

     End If
     values(One) = NextNumber(i)
     '\\  pointsize = NextNumber(i)
     If Not PerturbateOff Then
        .FontScroll.value = values(Five)
        If values(One) < One Then    '\\  pointsize
           inGrid.FontInt.Picture = inGrid_MenuPics_GraphicsCell(Six)
           PointSize = values(One)
        Else
           inGrid.FontInt.Picture = inGrid_toolclip_GraphicsCell(Six)
        End If
        If PointSize = Zero Then PointSize = Cent
        GridOut.Picture2.Font.size = PointSize
     End If
     If VerNUM >= 15.7 Then
        values(One) = NextNumber(i)
        If Not PerturbateOff Then .PathBar.Appearance = values(One)
     Else
        values(One) = NextNumber(i)  '\\   was <15.7 ForeSize = nextnumber( i)
        values(One) = NextNumber(i)  '\\   was <15.7 BackSize = nextnumber( i)
     End If
     values(One) = NextNumber(i)
     If Not PerturbateOff Then sized = Int(values(One))
  End If
  TextValues(Four) = vbNullString
  If VerNUM >= 9.62 Then
     TextValues(Four) = NEXTWORD(i)

  End If
  If Not PerturbateOff Then
     If .ForDoevents.value = Unchecked Then ReEnable
     inGridClick_DrawState
```

```
      End If
   End With
   Gettoolboxcount = Zero
   Exit Function
   Resume
errorline:        ' stop
   Close #One
   If i > Zero Then
      WarningError Err, i & " GetToolbox " & Gettoolboxcount, , , vbYes
   Else
      OneBell " GetToolbox "
   End If
   Gettoolboxcount = Zero
   GetToolbox = False
   If dmDrumsLoaded Then
      dmDrums.LabelBPM_Enabled = False
   End If
End Function
Function PutToolBox(Optional ByVal Include As Boolean = True, Optional ByVal delim As String
= vbTab, Optional ByVal UserAction As String = vbNullString) As String
   On Error GoTo errorline
   Dim i As Long
   i = -One    '\\   error trap for next statement only
   If ProcessingFlag <> PF_FinishedD3D Then
      If ZaxisMarker > Zero Then
         ZaxisMarker = Zero
         KarmaGun
      Else
         RestoreD3D
      End If
   End If
   i = Zero
   With inGrid
      Tools(i) = "<ToolboxSettings>"
      Tools(i) = "15.7"
      Tools(i) = str(.BackToFront.value)
      Tools(i) = str(.UpsideDown.value)
      Tools(i) = str(.InsideOut.value)  '\\  & ASpace & str(.SigLines.Value) & ASpace out 15.2
      Tools(i) = str(.Cycler.value)
      Tools(i) = str(.Off.BackColor)  '\\  & ASpace & str(.Stand.Value)
      Tools(i) = str(.Off.value)
      Tools(i) = str(.JetTrails.value)
      Tools(i) = str(.CenterLines.value)
      Tools(i) = str(.BrushFill.value)
      Tools(i) = str(.Outlines.value)
      Tools(i) = str(.ScopeOff.value)
      Tools(i) = str(.ClipOff.value)
```

```
    Tools(i) = str(.ForDoevents.value)
    Tools(i) = str(.Transparent.value)
    Tools(i) = str(.DrawState.value)
    Tools(i) = str(.SendSpeech.value)
    Tools(i) = str(.BlankPictures.value)
    Tools(i) = .SetConstructs.list(inGrid.ConSlider.value): If Trim$(.SetConstructs.list(inGrid.C
onSlider.value)) = vbNullString Then MsgBoxEx ".SetConstructs.Text looking for error"
    Tools(i) = .SetElements.list(inGrid.ElSlider.value)  '\\  GridOut.ScaleMode = vbCharacters
    Tools(i) = str(QuadBottom / Pic2Row)
    Tools(i) = str(QuadRight / Pic2Col)
    Tools(i) = str(QuadTop / Pic2Row)
    Tools(i) = str(QuadLeft / Pic2Col)
    Tools(i) = str(.ElementBackColor.value)
    Tools(i) = str(.SetElements.ListIndex)
    Tools(i) = str(.ConstructForeColor.value)
    Tools(i) = str(.SmallChanges.value)
    Tools(i) = str(.ElementForeColor.value)
    Tools(i) = str(.ConstructBackColor.value)
    Tools(i) = str(.GridBackColor.value)
    Tools(i) = str(.GridForeColor.value)  '\\  stop  inGridVER(zero,gDoc) 13.7
    Tools(i) = str(N7)  '\\  stop   inGridVER(zero,gDoc) 13.6)
    Tools(i) = str(.SetAxisZ.Min)
    Tools(i) = str(.SetAxisZ.Max)  '\\  stop    inGridVER(zero,gDoc) 13.8
    Tools(i) = str(.ZaxisLabel.ForeColor)
    Tools(i) = str(.ZaxisLabel.BackColor)
    Tools(i) = CSng(Abs(xaxis * DEG))    '\\  str(.SetAxisX.Value)
    Tools(i) = CSng(Abs(yaxis * DEG))    '\\  str(.SetAxisY.Value)
    Tools(i) = CSng(Abs(zaxis * DEG))    '\\  str(.SetAxisZ.Value)
    Tools(i) = str(.SetSig.value)
    Tools(i) = str(.OpCode.ListIndex)

    Tools(i) = BeatMix(.TimeStep.value)
    Tools(i) = str(.Updown.value)
    Tools(i) = str(.Sideways.value)
    Tools(i) = str(.Zoom.value)
    Tools(i) = str(.BandSize.value)
    Tools(i) = str(.ViewSize.value)
    Tools(i) = str(.FontScroll.value)
    Tools(i) = str(Sgn(CycleZ))
    Tools(i) = str(.ColorLines.value)
    Tools(i) = str(.DisplayTables.value)
    Tools(i) = str(.Layer(element).value)
    Tools(i) = str(.Layer(Construct).value)
    Tools(i) = str(.PictureStretch.value)
    Tools(i) = str(.PictureSize.value)
    Tools(i) = str(.AutoChanger.value)
    Tools(i) = str(.Blending.value)
```

```
    Tools(i) = str(AutoZoom)    '\\  Stop    inGridVER(zero,gDoc) 9.6)
    Tools(i) = str(.Mode.value)
    Tools(i) = str(.StyleWidth)
    Tools(i) = str(.FontInt)
    Tools(i) = str(FullScreen)  '\\  stop  inGridVER(zero,gDoc) 10.105)
    Tools(i) = str(.Beginner)
    Tools(i) = str(.Intermediate)
    Tools(i) = str(.Advanced)  '\\  stop  inGridVER(zero,gDoc) 9.61)
    Tools(i) = str(PointSize)
    Tools(i) = str(.PathBar.Appearance)
    Tools(i) = str(sized)  '\\  stop  inGridVER(zero,gDoc) 9.62)
    If UserAction = "Perturbate" Then
       Tools(i) = Quotes & UserAction & Quotes
    Else
       Tools(i) = Quotes & fLogin.txtUserName.Text & Quotes
    End If
    Dim workz6 As String
    If Not MacroPlaying Then
       '\\   if recording system grid then
       workz6 = JoinMacro
       If Len(workz6) <> Zero Then
          If MacroRow > Zero And nMacro > One Then
             fDoc(MacroRefDoc).Table.TextMatrix(MacroRow, Val(Mid$(Macro(nMacro - One),
12)) + Two) = workz6
             If fDoc(MacroRefDoc).Table.TextMatrix(MacroRow, NmE) <> "?" Then
                WarningError Err, "PutToolBox MacroRow <>?"    '\\  Stop
             End If

             SetDirty Two, MacroRefDoc
          End If
       End If
    End If

    If Include = True Then
       Tools(i) = workz6 & "</ToolboxSettings>"
    Else
       Tools(i) = "</ToolboxSettings>"
    End If


  End With

  PutToolBox = JoinTools(delim)    '\\  (Tools(), vbTab)
  If inDesign(Three) Then
     If Len(PutToolBox) < Ten Then MsgBoxEx ("Puttoolbox problem")
  End If
  Exit Function
```

```
   Resume
errorline:        ' stop
   If inDesign And Not i < Zero Then
      If Len(PutToolBox) < Ten Then MsgBoxEx ("Puttoolbox problem")
      WarningError Err, "PutToolBox", , , vbYes
   End If
End Function


Sub writeINI(Optional FirstSetup As Boolean = False)
   On Error GoTo errorline
   SettingsSave iniName, "Music", "ExternalHelper", Abs(ExternalHelper)
   If Not inGrid Is Nothing Then
      With inGrid

         If iniName <> NotString Then
            SettingsSave iniName, RegSettings, "StartupType", "DLL"
         End If
         nMacro = Zero
         If gDoc > Zero Then
            If Not KarmaGunAfterStartup Then Call SettingsSave(iniName, RegSettings, "ToolBo
x", PutToolBox)
            Call SettingsSave(iniName, RegSettings, "PictureList", getPictureList)
         End If
         If GridOutLoaded Then
            SettingsSave iniName, RegSettings, "FreeMemory", Val(ExtractHTML(GridOut.Captio
n, "FreeMemory="))
         ElseIf PF_Ending Then
            Exit Sub
         ElseIf Not FirstSetup Then
            SetExitType PF_StartExit, "Writeini not FirstSetup"
         End If
         '\\  If PF_Ending Then Exit Sub

         If .WindowState = vbNormal And Not SaveOnExit = vbNo And StartUpTypeIsNormal T
hen
            If .Left >= Zero And .Top > DEG Then
               SettingsSave iniName, RegSettings, "MainHeight", .Height
               SettingsSave iniName, RegSettings, "MainWidth", .Width
               SettingsSave iniName, RegSettings, "MainLeft", .Left
               If IngridHidden = Zero Then SettingsSave iniName, RegSettings, "MainTop", .To
p
            End If
            SettingsSave iniName, RegSettings, "SendSound", inGrid_SendSpeech_BackColor

         End If
         If Not fToolTips Is Nothing Then
```

```
            If fToolTips.WindowState = vbNormal And fToolTips.Left >= Zero Then
                SettingsSave iniName, RegSettings, "ToolTipsLeft", fToolTips.Left
                SettingsSave iniName, RegSettings, "ToolTipsTop", fToolTips.Top
                SettingsSave iniName, RegSettings, "ToolTipsWidth", fToolTips.Width

            End If
        End If
        Call SettingsSave(iniName, RegSettings, "CycleZ", CycleZ)
        If FullScreen Then inGridClick_mnuFullScreen
        If Not KarmaGunAfterStartup Then SaveGridOutSettings
        If gDoc > Zero Then
            If Not JPEGs(gDoc) Is Nothing Then
                SettingsSave iniName, RegSettings, "SearchPrefix", JPEGs(gDoc).SearchPrefix
                SettingsSave iniName, RegSettings, "SearchSuffix", JPEGs(gDoc).SearchSuffix
            End If
        End If
        SettingsSave iniName, RegSettings, "FTPtxtServer", FTPServer
        SettingsSave iniName, RegSettings, "FTPtxtUser", FTPUser
        SettingsSave iniName, RegSettings, "FTPtxtPassword", FTPPassword
        SettingsSave iniName, RegSettings, "FTPstrDirectory", FTPstrDirectory

        If Not GridOut Is Nothing Then
            SettingsSave iniName, "Font", "font", .FontNames.ListIndex & "  " & GridOut.Picture
2.Font
        End If
        If Not KarmaGunAfterStartup Then
            SettingsSave iniName, "Font", "ColorSwitch", Colorswitch
            SettingsSave iniName, "Font", "ColorForeground", fore(One, Six)
            SettingsSave iniName, "Font", "ColorBackground", fore(One, Five)
            SettingsSave iniName, "Font", "ColorForeConstruct", fore(One, Four)
            SettingsSave iniName, "Font", "ColorBackConstruct", fore(One, Three)
            SettingsSave iniName, "Font", "ColorForeElement", fore(One, Two)
            SettingsSave iniName, "Font", "ColorBackElement", fore(One, One)
            SettingsSave iniName, "Font", "MonoForeground", fore(Two, Six)
            SettingsSave iniName, "Font", "MonoBackground", fore(Two, Five)
            SettingsSave iniName, "Font", "MonoForeConstruct", fore(Two, Four)
            SettingsSave iniName, "Font", "MonoBackConstruct", fore(Two, Three)
            SettingsSave iniName, "Font", "MonoForeElement", fore(Two, Two)
            SettingsSave iniName, "Font", "MonoBackElement", fore(Two, One)
        End If

        If Not HostPending Then
            If LCase$(Right(FullGridName, 4)) = ".ing" Then
                SettingsSave iniName, "Files", "RecentFile1", FullGridName
            Else
                SettingsSave iniName, "Files", "RecentFile1", .mnuFileMRU(One).Caption
            End If
```

```
            SettingsSave iniName, "Files", "RecentFile2", .mnuFileMRU(Two).Caption
            SettingsSave iniName, "Files", "RecentFile3", .mnuFileMRU(Three).Caption
            SettingsSave iniName, "Files", "RecentFile4", .mnuFileMRU(Four).Caption
        End If
        SettingsSave iniName, "Files", "InitDir", CurDir
        SettingsSave iniName, "Databases", "RecentBase1", .mnuGridBaseMRU(One).Caption
        SettingsSave iniName, "Databases", "RecentBase2", .mnuGridBaseMRU(Two).Caption
        SettingsSave iniName, "Databases", "RecentBase3", .mnuGridBaseMRU(Three).Caption
        If Not GridOut Is Nothing Then
            If Not (Abs(GridOut.Top + GridOut.Height / Two - Screen.Height / Two) < Twenty)
Then ' And (GridOut.Left + GridOut.Width / Two - Screen.Width / Two) < Ten)
      '          Stop '20131128
                If Not KarmaGunAfterStartup Then SaveGridOutSettings '20220210
            End If
        End If
        SettingsSave iniName, "Screen", "KnownD3DAdapter", str(KnownD3DAdapter)
'         SettingsSave iniName, "Music", "DefaultGeminiAPIKey", myKey
'20260126 Unfinished
'         If myKey <> vbNullString Then SettingsSave iniName, "Music", "GeminiAPIKey", myKe
y
        SettingsSave iniName, "Music", "ExternalHelper", ExternalHelper

        Call SettingsSave(iniName, RegSettings, "SchedulePath", Filename_Ini(SchedulePath))
  '5/5/10

      End With
   End If

   Exit Sub
   Resume
errorline:       ' stop
   WarningError Err, "Writeini " & Error, , , vbYes
End Sub




Sub CompressING(ByRef pDoc As Long)

   On Error GoTo errorline

   Dim lEndPos As Long    '\\  , .DocText_SelStart As Long
   With fDoc(pDoc)
      .DocText_SelStart = .DocText.Find(IngridXMLHeader, , , rtfNoHighlight)
```

```
      If .DocText_SelStart >= Zero Then
         '\\         If CreateXMLGrid(gDoc) Then WELLFORMED

         .DocText_SelStart = .DocText_SelStart + Len(IngridXMLHeader) + Two
         '\\             .DocText.SelLength = fDoc(gDoc).MeTextLength - .DocText_SelStart
         lEndPos = .DocText.Find(IngridXMLFooter, .DocText_SelStart, , rtfNoHighlight)
         .DocText.SelLength = lEndPos - .DocText_SelStart

         .DocText.Text = Left$(.DocText.Text, .DocText_SelStart) & CompressString(.DocText.S
elText) & Mid$(.DocText.Text, lEndPos + One)
      End If
   End With
   Exit Sub
   Resume
errorline:       ' stop
   gCancel = True
End Sub

Function Analyzed(flag) As Boolean
   If flag = Zero Then
      Analysed = False
      inGrid.SpinGridX.Enabled = False
      inGrid.Pause.Enabled = False
      '\\  inGrid.ResetPlot.Enabled = False

   Else
      Analysed = True
      inGrid.SpinGridX.Enabled = True
      inGrid.Pause.Enabled = True
      '\\  inGrid.ResetPlot.Enabled = True
      If gDoc > Zero Then SetDirty Two
   End If

End Function
```