Analysis of Non-Steady State Physiological and Pathological Processes

Volume II

Nathan R Hill

Harris-Manchester College

Thesis Submitted for the Degree of Doctor of Philosophy

University of Oxford

Hilary Term 2008

Nuffield Department of Medicine
John Radcliffe Hospital
Headington
Oxford OX3 9DU

Chapter 1 – Easy TSA

The following code written in Visual Basic 6 is the final beta version of Easy TSA used to analyse the data in Volume I of this thesis. All commenting within the code is green.

The code (once compiled) will run on a Microsoft Windows platform.

Table of Contents for Easy TSA

```
Option Explicit
Public Time_Needed As Boolean



Private Sub cmdOpen_Click(Index As Integer)
   '=========Init Vars===========
   Dim FLX As MSFlexGrid

   CDBmain.DialogTitle = "File Open"
   CDBmain.Filter = "Text (*.txt)|*.txt|Excel (*.xls)|*.xls"

   If Index = 0 Then
      Time_Needed = True
      Huge_data = False


   Else    'Huge Pics
      Time_Needed = False
      Huge_data = True

   End If

   Set FLX = FRMHost.FLXMain

   '=========Position Form for Dialog Box=====
   Me.Visible = False
   Me.Width = 1
   Me.Height = 1
   Me.Left = FRMHost.Left + 2000
   Me.top = FRMHost.top + 1500
   '============Load Data=============
   CDBmain.ShowOpen

   If CDBmain.FileName <> "" Then
      Erase arrRaw
      Call Reset_Controls
      If LCase(Right(CDBmain.FileName, 4)) = ".txt" Then
         Call Load_Data_text
      Else
         Call Load_Data_Excel
      End If

      If Error_in_data = True Then
         GoTo Clean_UP
      End If

      Call Display_Data
      Call Choose_Time
      Call Enable_Buttons
      '============Set First Patient============
      FLX.Row = 1
      FRMHost.LBLInfo.Caption = FLX.TextMatrix(FLX.RowSel, 0)

      '========Check Raw Check box========
      FRMHost.CHKRaw.Value = 1

      '----------Headings---------
      FLX.TextMatrix(0, 0) = "Identifier"
      FLX.TextMatrix(0, 1) = "Assay"
      FLX.TextMatrix(0, 2) = "No. Of Values"
      FLX.TextMatrix(0, 3) = "Duplicate ?"
   End If

   '============Clean Up===============
Clean_UP:
   Set FLX = Nothing
   Unload Me
End Sub

Private Sub Form_Load()
   Border.Move 0, 0, Me.Width, Me.Height
```

```
   LBLInfo.Caption = "Please choose whether the Dataset is Time Dependent" & _
         " or Non-Time dependant." & vbCrLf & _
         "ie An image analysis would be Non-Time dependent"
End Sub
```

```
Option Explicit
Dim myRow As Integer 'saves current row of FLX
Dim myCol As Integer 'saves current Col of FLX
Public WithEvents CHTRaw As MSChart
Dim WithEvents FLXRaw_POI As MSFlexGrid
Dim CHTRaw_Location(3) As Integer

Private Sub CHKABS_Norm_Fourier_Click()
   If CHKABS_Norm_Fourier.Value = 1 Then
      If CHKABS_Norm_Fourier.Caption = "Make Relative" Then
         CHKABS_Norm_Fourier.Caption = "Make Absolute"
      Else
         CHKABS_Norm_Fourier.Caption = "Make Relative"
      End If


      Call CHKGrpPowerPeriod_Click
   End If
   CHKABS_Norm_Fourier.Value = 0
End Sub

Sub CHKAutocorrelate_Click()
   If CHKAutocorrelate.Value = 1 Then
      Call Create_Delete_Controls("AutoCorrelation", True)
      Call Draw_AutoCorrelation_Single
   Else
      Call Create_Delete_Controls("AutoCorrelation", False)
   End If

   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("AutoCorrelation")
   Call Select_Tab("AutoCorrelation")

End Sub

Private Sub CHKCrossCorrelate_Click()
   If CHKCrossCorrelate.Value = 1 Then
      Call Create_Delete_Controls("CrossCorrelation", True)

      '#Select Patients and Draw#
      Call Swap_Selected_Pats(True)
      Call Draw_CrossCorrelate
      Call Swap_Selected_Pats(False)

   Else
      Call Create_Delete_Controls("CrossCorrelation", False)
   End If

   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("CrossCorrelation")
   Call Select_Tab("CrossCorrelation")
End Sub

Sub CHKDeConvolution_Click()
   If CHKDeConvolution.Value = 1 Then

      Call Create_Delete_Controls("Deconvolution", True)
      Call Draw_Deconv
   Else
      Call Create_Delete_Controls("Deconvolution", False)
   End If
   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Deconvolution")
   Call Select_Tab("Deconvolution")
End Sub



Sub ChkFourier_Click()

   If CHKFourier.Value = 1 Then
      Call Create_Delete_Controls("Fourier", True)

      If cmdFourier_Abs_Norm.Caption = "Absolute" Then
         Call Draw_Fourier(True)
      Else
         Call Draw_Fourier(False)
```

```
      End If

      cmdFourier_Abs_Norm.Visible = True

   Else
      Call Create_Delete_Controls("Fourier", False)
      cmdFourier_Abs_Norm.Visible = False
   End If


   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Fourier")
   Call Select_Tab("Fourier")
End Sub

Private Sub CHKGrpAutocorrelation_Click()

   If CHKGrpAutocorrelation.Value = 1 Then
      Call Create_Delete_Controls("Grouped_Autocorrelation", True)

      '#Select Patients and Draw#
      Call Swap_Selected_Pats(True)
      Call Draw_AutoCorrelate_Group
      Call Swap_Selected_Pats(False)

   Else
      Call Create_Delete_Controls("Grouped_Autocorrelation", False)
   End If

   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Grouped_Autocorrelation")
   Call Select_Tab("Grouped_Autocorrelation")
End Sub



Sub CHKGrpPowerPeriod_Click()
   '-------Do a Absolute Fourier-------

   If CHKGrpPowerPeriod.Value = 1 Then
      '----Tick Fourier First---
      ' CHKABS_Norm_Fourier.Value = 0
      CHKFourier.Value = 1
      CHKPowerPeriod.Value = 1
      '-------------------------
      Call Create_Delete_Controls("Grouped_Power", True)
      '#Select Patints and Draw#
      Call Swap_Selected_Pats(True)
      'Call Draw_Power_Grouped
      Call Draw_PP
      Call Swap_Selected_Pats(False)

      '#Enable Normalise Button#
      CHKABS_Norm_Fourier.Visible = True



   Else
      Call Create_Delete_Controls("Grouped_Power", False)
      '#Disable Normalise Button#
      CHKABS_Norm_Fourier.Visible = False
   End If


   '===Refresh Other CHT's that can use this CHT===
   'Call Refresh_All("Grouped_Power")
   Call Select_Tab("Grouped_Power")
   'cmdFourier_Abs_Norm.Visible = False
End Sub

Sub chkMean_Click()
   If chkMean.Value = 1 Then
      Call Create_Delete_Controls("Mean_SEM", True)

      '#Select Patients and Draw#
      Call Swap_Selected_Pats(True)
      Call Draw_Mean_And_SEM
      Call Swap_Selected_Pats(False)
```

```
   Else
      Call Create_Delete_Controls("Mean_SEM", False)
   End If

   '===Refresh Other CHT's that can use this CHT===
   'Call Refresh_All("Mean")
   Call Select_Tab("Mean_SEM")

End Sub

Sub CHKObservedConcentration_Click()
   If CHKObservedConcentration.Value = 1 Then
      Call Create_Delete_Controls("ObservedConcentration", True)

      '#Select Patients and Draw#
      'Call Swap_Selected_Pats(True)
      Call Draw_ObservedConcentration
      'Call Swap_Selected_Pats(False)

   Else
      Call Create_Delete_Controls("ObservedConcentration", False)
   End If

   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("ObservedConcentration")
   Call Select_Tab("ObservedConcentration")
End Sub

Private Sub ChkPermutation_Click()
   If ChkPermutation.Value = 1 Then
      Call Swap_Selected_Pats(True)
      Call Permutation_test
      Call Swap_Selected_Pats(False)
   End If
End Sub

Sub CHKPowerPeriod_Click()
   If CHKPowerPeriod.Value = 1 Then
      '----Tick Fourier First---
      CHKFourier.Value = 1
      '-------------------------
      Call Create_Delete_Controls("Power", True)
      Call Draw_Power

   Else
      Call Create_Delete_Controls("Power", False)

   End If
   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Power")
   Call Select_Tab("Power")
End Sub

Sub CHKRaw_Click()
   If CHKRaw.Value = 1 Then
      '  CHKTrend_Line.Visible = True
      '  CHKTrend_Line.ZOrder (0)
      Call Create_Delete_Controls("Raw", True)
      Call Draw_Raw
   Else
      Call Create_Delete_Controls("Raw", False)
      '  CHKTrend_Line.Visible = False
   End If
   '===Refresh Other CHT's that can use this CHT===

   Call Refresh_All("Raw")
   Call Select_Tab("Raw")
End Sub

Private Sub chkRaw_Resample_Click()
   If chkRaw_Resample.Value = 1 Then
      Call Resample_Data
   Else
      Call CHKRaw_Click
   End If
End Sub
```

```
Private Sub chkSave_Click()
   If chkSave.Value = 1 Then
      Call Save_Data
   End If
End Sub

Private Sub CHKSerial_Single_Draw_Click()
   '----Init Vars-----
   Dim FLX As MSFlexGrid
   Dim irow As Integer, iPat As Integer, strName As String
   Set FLX = FRMHost.Controls("FLXSerial")

   If FRMHost.CHKSerial_Single_Draw.Value = 1 Then
      '=====Check that we have some POI's======
      If FLX.TextMatrix(1, 0) <> "" Then
         If Not IsNumeric(FLX.TextMatrix(1, 0)) Then
            '---Move Relevant Pats to Grouped FLX-----
            For irow = 1 To FLX.Rows - 1
               strName = FLX.TextMatrix(irow, 0)
               For iPat = 1 To FRMHost.FLXMain.Rows - 1
                  If strName = FRMHost.FLXMain.TextMatrix(iPat, 0) Then
                     FRMHost.FLXMain.Row = iPat
                     If FRMHost.FLXMain.CellBackColor <> Highlight Then
                        Call FLXMain_DblClick
                     End If
                  End If
               Next iPat
            Next irow

            '-----Call Drawing Function------
            If FRMHost.FLXGrouped.Rows > 3 Then
               '------Draw Grouped SAA-----
               Call Swap_Selected_Pats(True)
               Call Draw_Serial_Group
               Call Swap_Selected_Pats(False)
            Else
               '------Draw Single SAA------
               If FLX.Rows <= 2 Then
                  Call Serial_Analyse_Single(, True)
               Else
                  Call Serial_Analyse_Single(, False)
               End If
               'Set tmp = Nothing

            End If

            FRMHost.CHKSerial_Single_Draw.Value = 0
            FRMHost.CHKSerial_Single_Draw.Visible = False
         End If
      End If
   End If
End Sub

Sub CHKSerialArrayAveraging_Click()

   If CHKSerialArrayAveraging.Value = 1 Then
      Call Create_Delete_Controls("Serial", True)
      Call Draw_Serial_Start
      Call Add_on_POI(CHTSerial, Me.Controls("FLXSerial"))
      CHKSerial_Single_Draw.Visible = True
      CHKSerial_Single_Draw.ZOrder (0)
      'CHKSerial_Single_Draw.Enabled = False
   Else
      Call Create_Delete_Controls("Serial", False)
      CHKSerial_Single_Draw.Visible = False
   End If


   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Serial")
   Call Select_Tab("Serial")
End Sub

Private Sub CHKTrend_Line_Click()
   If CHKTrend_Line.Value = 1 Then
      Call Add_TrendLine
   Else
```

```
      Call CHKRaw_Click
   End If
End Sub

Private Sub CHTRaw_PointActivated(Series As Integer, Datapoint As Integer, MouseFlags As Integer, Cancel As
» Integer)
   '=============Init Vars===========
   Dim CHT As MSChart, FLX As MSFlexGrid
   Static Once As Boolean
   Set CHT = CHTRaw
   Set FLX = FRMHost.Controls("FLXRaw_POI")

   '---Format FLX---
   FLX.Move FLXRaw.Left, FLXRaw.top, FLXRaw.Width, FLXRaw.Height
   FLX.ColWidth(0) = FLXRaw.ColWidth(0)
   FLX.ColWidth(1) = FLXRaw.ColWidth(1)
   FLX.ZOrder (0)
   FLX.Visible = True
   If Once = False Then
      FLX.TextMatrix(0, 0) = "Identifier"
      FLX.TextMatrix(0, 1) = "P.O.I."
      FLX.Visible = True

      Once = True
   End If


   '====Select Deselect Point of interest===
   Call Cht_Point_Select(CHT, FLX, Datapoint)
End Sub
Private Sub CHTRaw_PointSelected(Series As Integer, Datapoint As Integer, MouseFlags As Integer, Cancel As
»Integer)
   CHTRaw.TitleText = FLXRaw.TextMatrix(Datapoint, 0)
   'FLXRaw_POI.ZOrder (0)

End Sub

Private Sub CHTSerial_PointActivated(Series As Integer, Datapoint As Integer, MouseFlags As Integer, Cancel
» As Integer)

   '=============================================================================================
   'DESCRIPTION: _
Double Click on the Serial Array averaging Graph - This sub finds where you clicked _
then inserts a Point of interest there
   '=============================================================================================

   '=============Init Vars===========
   Dim CHT As MSChart, FLX As MSFlexGrid



   Set CHT = CHTSerial
   Set FLX = FRMHost.Controls("FLXSerial")
   '===Add Dataset to CHT for POI's===
   Call Cht_Point_Select(CHT, FLX, Datapoint)

End Sub

Private Sub CHTSerial_PointSelected(Series As Integer, Datapoint As Integer, MouseFlags As Integer, Cancel
»As Integer)
   CHTSerial.TitleText = Me.Controls("FLXRaw").TextMatrix(Datapoint + 1, 0)
End Sub



Private Sub cmdExport_Click()
   Call Print_Out(False)
End Sub

Private Sub cmdFourier_Abs_Norm_Click()
   If cmdFourier_Abs_Norm.Caption = "Normalise" Then
      '----Format as Button-----
      cmdFourier_Abs_Norm.Caption = "Absolute"
      Call Draw_Fourier(True)

   Else
      '----Format as Button-----
```

```
      cmdFourier_Abs_Norm.Caption = "Normalise"
      Call Draw_Fourier(False)

   End If

   Call Refresh_All("Fourier")
End Sub

Private Sub cmdLoad_Click()
   FRMchoose.Show vbModal, FRMHost

End Sub




Private Sub cmdPrint_Click()
   Call Print_Out(True)
End Sub
```

```
Sub FLXMain_DblClick()
   '=================Init Vars=====================
   Dim irow As Integer, icol As Integer, jCol As Integer

   '==================Escape Clause================
   If FLXMain.TextMatrix(FLXMain.RowSel, 0) = "" Then Exit Sub


   '=================Colour & Copy==================
   If FLXMain.CellBackColor = Blank Then
      '#Colour It#
      For icol = 0 To FLXMain.Cols - 1
         FLXMain.Col = icol
         FLXMain.CellBackColor = Highlight
      Next icol

      '------------Copy it------------
      For irow = 1 To FLXGrouped.Rows - 1
         If FLXGrouped.TextMatrix(irow, 3) = "" Then
            '-------Add a row if needed------
            If irow = FLXGrouped.Rows - 1 Then
               FLXGrouped.Rows = FLXGrouped.Rows + 1
               '#Change Patient Name Label to Grouped#
               If FLXGrouped.Rows > 3 Then
                  LBLInfo = "Group Selected"
               End If
            End If
            '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            '------------Copy-------------
            For jCol = 0 To FLXMain.Cols - 1
               FLXGrouped.TextMatrix(irow, jCol) = FLXMain.TextMatrix(FLXMain.RowSel, jCol)
            Next jCol
            '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

            '-------Update Grouped TSA's--------
            If CHKGrpPowerPeriod.Value = 1 Then
               Call CHKGrpPowerPeriod_Click

            End If
            Call Refresh_All("all")
            '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         End If

      Next irow
      '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      '----Enable Grp Functions-----
      If FLXGrouped.TextMatrix(2, 0) <> "" Then
         chkMean.Enabled = True
         CHKGrpPowerPeriod.Enabled = True
         CHKGrpAutocorrelation.Enabled = True
      End If
      '==============Uncolour and uncopy it==============
   Else
      For icol = 0 To FLXMain.Cols - 1
         FLXMain.Col = icol
         FLXMain.CellBackColor = Blank
      Next icol

      '--------UnCopy it------------

      For irow = 1 To FLXGrouped.Rows - 1
         If FLXMain.TextMatrix(FLXMain.RowSel, 0) = FLXGrouped.TextMatrix(irow, 0) Then
            If FLXMain.TextMatrix(FLXMain.RowSel, 1) = FLXGrouped.TextMatrix(irow, 1) Then
               If FLXMain.TextMatrix(FLXMain.RowSel, 2) = FLXGrouped.TextMatrix(irow, 2) Then

                  FLXGrouped.RemoveItem (irow)

                  '#change Patient Name Lable#
                  If FLXGrouped.Rows < 3 Then
                     LBLInfo = ""
                  End If

                  '-------Update Grouped TSA's--------
                  If CHKGrpPowerPeriod.Value = 1 Then
                     Call CHKGrpPowerPeriod_Click
```

```
                End If
                Call Refresh_All("all")
                '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

                Exit For
             End If
          End If
       End If
    Next irow

    '----Disable Grp Functions-----
    If FLXGrouped.Rows > 2 Then
       If FLXGrouped.TextMatrix(2, 0) = "" Then
          chkMean.Enabled = False
          CHKGrpPowerPeriod.Enabled = False
          CHKGrpAutocorrelation.Enabled = False
       End If
    End If


 End If
End Sub



Private Sub FLXMain_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
 If Button = 2 Then          'if right mouse button clicked
    PopupMenu MNUnothing
 End If
End Sub

Sub FLXMain_SelChange()
 Static OLDROW As Integer
 '---escape clause---

 If SWAPPING_FLX = True Then Exit Sub

 '---redraw chts/flxs---
 If FLXMain.TextMatrix(FLXMain.RowSel, 0) <> "" Then
    Call Draw_Raw

    If GRP_POWER_ON <> True Then 'don't refresh if doing group power/period
       Call Refresh_All("Raw")
    Else
       If OLDROW <> FLXMain.Row Then 'don't keep refreshing same rows if are doing power/period
          OLDROW = FLXMain.Row
          Call Refresh_All("Raw")
       End If
    End If

    If What_Tab = "Raw" Then

       If FRMHost.Controls("FLXRaw_POI").TextMatrix(1, 0) <> "" Then
          FRMHost.Controls("FLXRaw_POI").ZOrder (0)
          FRMHost.Controls("FLXRaw_POI").Visible = True
          Call Add_on_POI(CHTRaw, FRMHost.Controls("FLXRaw_POI"))
       End If
    End If
 End If
 If LBLInfo <> "Group Selected" Then
    LBLInfo.Caption = FLXMain.TextMatrix(FLXMain.RowSel, 0)
 End If
End Sub

Private Sub Form_Load()
 '----------Headings---------
 FLXMain.TextMatrix(0, 0) = "Identifier"
 FLXMain.TextMatrix(0, 1) = "Assay"
 FLXMain.TextMatrix(0, 2) = "No. Of Values"
 FLXMain.TextMatrix(0, 3) = "Duplicate ?"
End Sub

Private Sub Form_Resize()
 If Me.Height < 361 Then 'don't resize when minimised
    Exit Sub
 End If

 If Me.Width < 12000 Then
```

```
      Me.Width = 12000
      Exit Sub
   End If

   If Me.Height < 8000 Then
      Me.Height = 8000
      Exit Sub
   End If
   If Me.Width > 2400 Then
      Call Resize
   End If
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
   On Error Resume Next

   Unload FRMchoose
   Unload frmWait
   Unload Me

End Sub

Private Sub iTB_Click(Index As Integer)
   If Index = 0 Then
      Me.WindowState = vbMinimized
   ElseIf Index = 1 Then
      Me.WindowState = vbMaximized
   Else
      Unload Me
   End If
End Sub

Private Sub MNUArea_Click()
   MNUArea.Checked = Not MNUArea.Checked
End Sub

Private Sub MNUGA_Click()
   MNUGA.Checked = Not MNUGA.Checked
End Sub

Private Sub MNURand_Click()
   MNURand.Checked = Not MNURand.Checked
End Sub

Private Sub MNUSelect_Click()
   '----Init Vars----
   Dim irow As Integer, icol As Integer, idata As Long

   '---Set Rows---
   FLXGrouped.Rows = FLXMain.Rows
   '----Add colour in FLX main-----
   For irow = 1 To FLXMain.Rows - 2
      For icol = 0 To FLXMain.Cols - 1
         FLXMain.Col = icol
         FLXMain.Row = irow
         FLXMain.CellBackColor = Highlight
         FLXGrouped.TextMatrix(irow, icol) = FLXMain.Text
      Next icol
   Next irow

   '----Copy all----


   chkMean.Enabled = True
   CHKGrpPowerPeriod.Enabled = True
   CHKGrpAutocorrelation.Enabled = True

End Sub

Private Sub MNUClear_Click()
   '------Init Vars-------
   Dim irow As Integer, icol As Integer

   '---Escape Clauses-------
   If FLXMain.TextMatrix(1, 0) = "" Then Exit Sub

   '---Reset Grouped FLX----
   FRMHost.FLXGrouped.Clear
   FRMHost.FLXGrouped.Rows = 2

   '----Clear colour in FLX main-----
   For irow = 1 To FLXMain.Rows - 1
      For icol = 0 To FLXMain.Cols - 1
         FLXMain.Col = icol
         FLXMain.Row = irow
         FLXMain.CellBackColor = Blank
      Next icol
   Next irow
End Sub
```

```
Sub OptDetrend_Click(Index As Integer)
   '========Choose which Button Pushed========
   If Index = 0 Then
      Call Create_Delete_Controls("Detrended", False)
   Else
      Call Create_Delete_Controls("Detrended", True)
      Call Draw_Detrend
   End If

   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Detrended")
   Call Select_Tab("Detrended")
End Sub

Sub OptMpa_Click(Index As Integer)

   '========Choose which Button Pushed========
   If Index = 0 Then
      Call Create_Delete_Controls("Mpa", False)
   Else
      Call Create_Delete_Controls("Mpa", True)
      Call Draw_MPA
   End If
   '===Refresh Other CHT's that can use this CHT===
   Call Refresh_All("Mpa")
   Call Select_Tab("Mpa")
End Sub

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
»~~~~
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
»~~~~
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
»~~~~


Sub FLXRAW_KeyPress(KeyAscii As Integer)
   '========FLXRAW EDIT CODE===========
   '##################################
   '==================================
   If FLXRaw.RowSel = 0 Or FLXRaw.Row = 0 Then Exit Sub

   If TXTedit.Visible Then TXTedit.SetFocus: Exit Sub
   MSHFlexGridEdit FLXRaw, TXTedit, KeyAscii
End Sub

Sub FLXRAW_DblClick()
   '========FLXRAW EDIT CODE===========
   '##################################
   '==================================
   If FLXRaw.RowSel = 0 Or FLXRaw.Row = 0 Then Exit Sub
   If TXTedit.Visible Then TXTedit.SetFocus: Exit Sub
   MSHFlexGridEdit FLXRaw, TXTedit, 32 ' Simulate a space.
End Sub

Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
      Edt As Control, KeyAscii As Integer)

   '========FLXRAW EDIT CODE===========
   '##################################
   '==================================
   If MSHFlexGrid.ColSel = 0 Then Exit Sub
   If MSHFlexGrid.RowSel = 0 Then Exit Sub
   ' Use the character that was typed.
   Select Case KeyAscii

      ' A space means edit the current text.
   Case 0 To 32
      Edt = MSHFlexGrid
      Edt.SelStart = 1000

      ' Anything else means replace the current text.
   Case Else
      Edt = Chr(KeyAscii)
      Edt.SelStart = 1
   End Select
```

```
' Show Edt at the right place.
Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
      MSHFlexGrid.top + MSHFlexGrid.CellTop, _
      MSHFlexGrid.CellWidth - 8, _
      MSHFlexGrid.CellHeight - 8
Edt.Visible = True

' And make it work.
```

```
   Edt.SetFocus
   Edt.ZOrder (0)


End Sub

Private Sub TabMain_Click()
   Dim Tab_Name As String
   Dim i As Integer
   For i = 1 To Tabmain.Tabs.Count
      If Tabmain.Tabs(i).Selected = True Then
         Tab_Name = Tabmain.Tabs(i).Image
         Exit For
      End If
   Next i

   '---escape clause----
   'stops clicking on any tabs b4 data is loaded
   If Tab_Name = "Raw" Then
      If FLXMain.TextMatrix(1, 0) = "" Then Exit Sub
   End If

   Me.Controls("FLX" & Tab_Name).ZOrder (0)
   Me.Controls("CHT" & Tab_Name).ZOrder (0)

   '-------View Hide extra buttons----------
   'hide'
   CHKTrend_Line.Visible = False
   CHKSerial_Single_Draw.Visible = False
   CHKABS_Norm_Fourier.Visible = False
   cmdFourier_Abs_Norm.Visible = False
   Me.chkRaw_Resample.Visible = False


   'unhide'
   Select Case Tab_Name
   Case "Raw"
      CHKTrend_Line.Visible = True
      CHKTrend_Line.ZOrder (0)
      chkRaw_Resample.Visible = True
   Case "Mean_SEM"
      CHKTrend_Line.Visible = True
      CHKTrend_Line.ZOrder (0)
   Case "Serial"
      CHKSerial_Single_Draw.Visible = True
      CHKSerial_Single_Draw.ZOrder (0)

   Case "Grouped_Power"
      CHKABS_Norm_Fourier.Visible = True
      CHKABS_Norm_Fourier.ZOrder (0)

   Case "Fourier", "Power"
      cmdFourier_Abs_Norm.Visible = True
      cmdFourier_Abs_Norm.ZOrder (0)



   End Select
End Sub

Private Sub TXTEdit_GotFocus()
   '========FLXRAW EDIT CODE===========
   '###################################
   '===================================
   If myRow = 0 Then
      myRow = FLXRaw.Row
      myCol = FLXRaw.Col
   End If
End Sub

Sub txtEdit_KeyPress(KeyAscii As Integer)
   '========FLXRAW EDIT CODE===========
   '###################################
   '===================================
   ' Delete returns to get rid of beep.
   If KeyAscii = Asc(vbCr) Then KeyAscii = 0
End Sub
```

```
Sub txtEdit_KeyDown(KeyCode As Integer, _
      Shift As Integer)
   '========FLXRAW EDIT CODE===========
   '###################################
   '===================================
   EditKeyCode FLXRaw, TXTedit, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
      Control, KeyCode As Integer, Shift As Integer)
   '========FLXRAW EDIT CODE===========
   '###################################
   '===================================
   ' Standard edit control processing.
   Select Case KeyCode

   Case 27   ' ESC: hide, return focus to MSHFlexGrid.
      Edt.Visible = False
      MSHFlexGrid.SetFocus

   Case 13   ' ENTER return focus to MSHFlexGrid.
      MSHFlexGrid.SetFocus

   Case 38      ' Up.
      MSHFlexGrid.SetFocus
      DoEvents
      If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
         MSHFlexGrid.Row = MSHFlexGrid.Row - 1
      End If

   Case 40      ' Down.
      MSHFlexGrid.SetFocus
      DoEvents
      If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
         MSHFlexGrid.Row = MSHFlexGrid.Row + 1
      End If
   End Select
End Sub




Private Sub TXTedit_LostFocus()
   '========FLXRAW EDIT CODE===========
   '###################################
   '===================================


   If IsNumeric(TXTedit.Text) = True Then
      TXTedit.ForeColor = &H0&
      '----Sucessful editing----
Its_Interpolated:
      '#Select Relevant Cell#
      FLXRaw.Row = myRow
      FLXRaw.Col = myCol
      FLXRaw = TXTedit
      '#Reset Attributes
      TXTedit.Visible = False
      'FLXRaw.RowSel = myRow
      ' FLXRaw.ColSel = myCol
      myRow = 0
      myCol = 0
      '#Update Data & CHT's#
      Call Update_Data
      Call Draw_Raw
      Call Refresh_All("Raw")
      '------------------------
   ElseIf TXTedit.Text = "" Then
      TXTedit.Visible = False
   ElseIf Right(TXTedit.Text, 1) = "*" And _
         IsNumeric(Left(TXTedit.Text, Len(TXTedit.Text) - 1)) = True Then
      GoTo Its_Interpolated

   Else
      TXTedit.Visible = True
      TXTedit.SetFocus
```

```
      TXTedit.ForeColor = &HFF&

   End If
End Sub
```

```
Option Explicit

Private Sub Form_Load()
   Border.Move 0, 0, Me.Width, Me.Height
End Sub
```

```
'Dim A As New Matrix, B As New Matrix, C As New Matrix 'BLUEBIT ACTIVEX MATRIX
Dim Mat As New cMatrixMathLib 'Define a variable as the matrix library class
Dim A() As Double, B() As Double, C() As Double



Function DET_Cubic(arrDetrend_Start() As Variant) As Variant
   '-init vars-
   ReDim X(UBound(arrDetrend_Start)) As Variant
   ReDim Y(UBound(arrDetrend_Start)) As Variant
   Dim tmp() As String
   ReDim arrDetrended(UBound(arrDetrend_Start), 1) As Variant
   Dim Yi As Double, Xi As Double
   '-set x,y arrays-
   For i = 0 To UBound(arrDetrend_Start)
      tmp() = Split(arrDetrend_Start(i, 0), " ")
      X(i) = tmp(0)
      Y(i) = arrDetrend_Start(i, 1)

      '-copy x vals to final array-
      arrDetrended(i, 0) = arrDetrend_Start(i, 0)
   Next i

   '-Get Best Coefficients to use to detrend store in matrix C-
   Call Calc_Coeffs(X(), Y(), 3)

   '-Detrend the data-
   For i = 0 To UBound(arrDetrend_Start)
      'Y= a + bx + cx2 + dx3
      Xi = X(i)
      Yi = C(0, 0) + (C(1, 0) * Xi) + (C(2, 0) * Xi ^ 2) + (C(3, 0) * Xi ^ 3)
      arrDetrended(i, 1) = arrDetrend_Start(i, 1) - Yi
   Next i

   DET_Cubic = arrDetrended
   Erase A
   Erase B
   Erase C
End Function
Private Sub Calc_Coeffs(X() As Variant, Y() As Variant, Order As Integer)
   '-init vars-

   'A.Size Order + 1, Order + 1
   'B.Size Order + 1, 1
   ReDim A(Order, Order)
   ReDim B(Order, 0)
   '-fill matrices-
   Call Fill_A(X())
   Call Fill_B(X(), Y())

   '-invert A-
   A = Mat.Inv(A)
   C = Mat.Multiply(A, B)
   'Set A = A.Inverse
   'Set C = A.Times(B)
   For i = 0 To UBound(C, 2) 'C.Rows - 1 '
      Debug.Print C(i, 0) & vbCrLf
   Next
End Sub

Private Sub Fill_A(X() As Variant)
   Dim i As Integer, j As Integer
   For i = 0 To UBound(A, 2) 'A.Rows - 1 '
      For j = 0 To UBound(A) 'A.Cols - 1 '
         A(i, j) = SumofXfactor(i, j, X())
      Next
   Next

End Sub

Private Function SumofXfactor(i As Integer, j As Integer, X() As Variant)
   Dim sum As Variant

   For k = 0 To UBound(X)
      sum = sum + (X(k) ^ (i + j))
   Next
```

```
   SumofXfactor = sum
End Function

Private Sub Fill_B(X() As Variant, Y() As Variant)
   Dim i As Integer
   For i = 0 To UBound(B) 'B.Rows - 1 '
      B(i, 0) = SumofXfactorAndY(i, X(), Y())
   Next

End Sub

Private Function SumofXfactorAndY(j As Integer, X() As Variant, Y() As Variant)
   Dim sum As Variant
   For k = 0 To UBound(X)
      sum = sum + ((X(k) ^ j) * Y(k))
   Next
   SumofXfactorAndY = sum
End Function
```

```
Option Explicit
' This module is split into two main parts. AutoCoreelation Single & Autocorrelation Grouped _
Single :- _
1) Gets the data to Correlate                                   (AutoCorrelate_Single) _
2) Makes 2 arrays of the data                                   (AutoCorrelate_Single) _
3) Compares the data using the correlation co-efficient         (AutoCorrelate_Single) _
4) Shifts the data out of phase, compares it and repeats        (AutoCorrelate_Single) _
5) Then it outputs the Graphs and                               (FLX Draw_AutoCorrelation_Single) _
Grouped :- _
1) Does everything in Single                                    (AutoCorrelate_Single) _
2) Saves the Data for that patient and does the next            (Draw_AutoCorrelate_Group) _
3) When all Patients done it Z transforms their data           (Draw_AutoCorrelate_Group) _
4) Adds it together and finds the Mean                          (Draw_AutoCorrelate_Group) _
5) it Then Reverse Z transforms it                              (Draw_AutoCorrelate_Group) _
6) And outputs the Cht & FLX                                    (Draw_AutoCorrelate_Group) _
 _
 There is a problem with The Group Function as the Z transform may be incorrect, need to check

Public Sub Draw_AutoCorrelation_Single()
   '======Init Vars=====
   Dim arrAutoCorr() As Variant

   arrAutoCorr = AutoCorrelate_Single()

   '===================Fill FLX=================
   Call Fill_Flx(arrAutoCorr, "AutoCorrelation")

   '==============Position Controls===============
   Call Format_Controls("AutoCorrelation", 3, "N", "R", "Limits")

   '==================Draw CHT=================
   FRMHost.Controls("ChtAutoCorrelation").ChartData = arrAutoCorr

   '======Clean Up=======
   Erase arrAutoCorr
End Sub

Function AutoCorrelate_Single(Optional Are_we_Cross As Boolean, Optional Doing_left_AC As Boolean) As
»Variant
   '================= Init Vars ================
   Dim R As Double, X As Double, Y As Double
   Dim SumTop As Double, SumBot As Double, SumBotX As Double, SumBotY As Double, RootBot As Double
   Dim MeanY As Double, MeanX As Double, Data_Items_Total As Long
   Dim idata As Long, i As Long, FLX As MSFlexGrid, iTab As Integer
   Dim Half As Long
   Dim MPA_Found As Boolean, Det_Found As Boolean
   Set FLX = FRMHost.FLXMain
   idata = 1
   '========= Get Data to AutoCorrelate =========
   Dim Arr() As Variant, cloneArr() As Variant

   If Are_we_Cross = False Then
      Call Draw_Raw
      Arr = What_Flx()
      cloneArr = Arr
   Else

      If Doing_left_AC = False Then
         '-------Get Data for 1st Patient--------
         FLX.Row = 1
         Call FRMHost.FLXMain_SelChange
         Arr = What_Flx()
         '----Change to next Selected Patient----
         FLX.Row = 2
         Call FRMHost.FLXMain_SelChange
         cloneArr = What_Flx()
      Else
         FLX.Row = 2
         Call FRMHost.FLXMain_SelChange
         Arr = What_Flx()
         '----Change to next Selected Patient----
         FLX.Row = 1
         Call FRMHost.FLXMain_SelChange
         cloneArr = What_Flx()
      End If
```

```
   End If

   '=============== Init More Vars =============
   Half = CLng(UBound(Arr) / 2) - 1
   ReDim arrOF_R_Vals(Half - 1, 3) As Variant



   '=========== Perform AutoCorrelation =========
   Do While idata <= Half

      '------ Reset Vars ------
      SumTop = 0
      SumBotX = 0
      SumBotY = 0
      SumBot = 0
      MeanX = 0
      MeanY = 0
      '~~~~~~~~~~~~~~~~~~~~~~

      '------ Calculate Means -----
      For i = 0 To UBound(cloneArr) - idata
         MeanX = MeanX + Arr(i, 1)
         MeanY = MeanY + cloneArr(i + idata, 1)
      Next i



      MeanX = MeanX / i
      MeanY = MeanY / i
      '~~~~~~~~~~~~~~~~~~~~~~~~~~

      '---------------- Calculate Co-efficient ------------------
      ' R = sum x-xbar*y-bar/ root(sum(x-xbar^2)* sum(Y-ybar^2))
      For i = 0 To UBound(cloneArr) - idata
         X = Arr(i, 1)
         Y = cloneArr(i + idata, 1)
         SumTop = SumTop + ((X - MeanX) * (Y - MeanY))
         SumBotX = SumBotX + ((X - MeanX) * (X - MeanX))
         SumBotY = SumBotY + ((Y - MeanY) * (Y - MeanY))
      Next i

      SumBot = 0
      SumBot = SumBotX * SumBotY
      RootBot = 0
      RootBot = Sqr(SumBot)

      '---------- Store Results ----------------
      arrOF_R_Vals(idata - 1, 1) = SumTop / RootBot
      arrOF_R_Vals(idata - 1, 0) = idata & " "

      idata = idata + 1
   Loop



   '=========Add Significance Bars of +-2/sqr(N)===========
   '1.96 = 95% CI
   '1.645 = 90% CI
   Data_Items_Total = UBound(Arr)
   For i = 0 To UBound(arrOF_R_Vals)
      arrOF_R_Vals(i, 2) = 1.645 / Sqr(Data_Items_Total)
      arrOF_R_Vals(i, 3) = -1.645 / Sqr(Data_Items_Total)
      Data_Items_Total = Data_Items_Total - 1
   Next i

   '========= Return Array ============
   AutoCorrelate_Single = arrOF_R_Vals()

   '======Clean Up=======
   Erase Arr
   Erase cloneArr
   Erase arrOF_R_Vals
End Function
Sub Draw_AutoCorrelate_Group()
   '==================== Init Vars ========================
   Dim arrOF_R_Vals() As Variant
```

```
Dim FLX As MSFlexGrid, idata As Long, iPat As Integer, iSum As Integer
Dim Sum_Total3 As Double, Sum_Total1 As Double, Sum_Total2 As Double, FRM As Form
Dim tmpTop As Double, tmpBot As Double, TopBot As Double
Set FLX = FRMHost.FLXMain
Set FRM = FRMHost

'--set first patient-
FLX.Row = 1
FRMHost.FLXMain_SelChange
'---------------------
'#Define Array to store Data#
'===Find largest No of dataitems and current number of Pats===
Dim max_Pat As Integer, Max_Data As Integer, i As Integer

max_Pat = FLX.Rows - 3
```

```
'---find smallest N in FLX----
Dim smallest As Integer
smallest = FLX.TextMatrix(1, 2)
For i = 1 To FLX.Rows - 2
   If FLX.TextMatrix(i, 2) < smallest Then
      smallest = FLX.TextMatrix(i, 2)
   End If
Next i

'---Only want 1/2 Of period---
smallest = (smallest / 2) - 1
Max_Data = smallest '- 2

'-Define Array to hold all Data-
'ReDim arrOf_R_ALL(Max_Data, 2, max_Pat) As Variant
ReDim arrOF_Z_Vals_Grouped(Max_Data, 3, max_Pat) As Variant

'=============== Loop Through all Patients ===============
For iPat = 0 To FLX.Rows - 3


   '================= Get Array of R vals ==================
   arrOF_R_Vals = AutoCorrelate_Single

   '======= Normalsize it by Z transform and Store it as R======
   ' Z = 1/2 * ln(1+r/1-r)

   '---- Init Array for Z  ------
   ReDim arrOf_Z_Vals(UBound(arrOF_R_Vals), 2) As String



   '----- Store Vars in Arrays -----
   For idata = 0 To UBound(arrOf_Z_Vals)
      '---- Perform Transform on data-----
      tmpTop = 1 + arrOF_R_Vals(idata, 1)
      tmpBot = 1 - arrOF_R_Vals(idata, 1)
      TopBot = tmpTop / tmpBot
      arrOf_Z_Vals(idata, 0) = 0.5 * excel.WorksheetFunction.Ln(TopBot)

      '--perform transfer onlimits----
      'upper
      tmpTop = 1 + arrOF_R_Vals(idata, 2)
      tmpBot = 1 - arrOF_R_Vals(idata, 2)
      TopBot = tmpTop / tmpBot
      arrOf_Z_Vals(idata, 1) = 0.5 * excel.WorksheetFunction.Ln(TopBot)

      'Lower
      tmpTop = 1 + arrOF_R_Vals(idata, 3)
      tmpBot = 1 - arrOF_R_Vals(idata, 3)
      TopBot = tmpTop / tmpBot
      arrOf_Z_Vals(idata, 2) = 0.5 * excel.WorksheetFunction.Ln(TopBot)

   Next idata


   '==================== Add to Huge Array ==================

   '----- Init Grouped Array -----


   '-------- Add Data in ---------
   For idata = 0 To UBound(arrOF_Z_Vals_Grouped())
      '-escape clause when mpa and detrend used array of data is shorter-
      If idata > UBound(arrOf_Z_Vals) Then Exit For

      arrOF_Z_Vals_Grouped(idata, 0, iPat) = (idata + 1)
      arrOF_Z_Vals_Grouped(idata, 1, iPat) = arrOf_Z_Vals(idata, 0)
      arrOF_Z_Vals_Grouped(idata, 2, iPat) = arrOf_Z_Vals(idata, 1)
      arrOF_Z_Vals_Grouped(idata, 3, iPat) = arrOf_Z_Vals(idata, 2)
   Next idata

   '========= Get next Patient ========
   FRM.FLXMain.Row = FRM.FLXMain.Row + 1
   Call FRM.FLXMain_SelChange

Next iPat
```

```
'============== Find Mean Normalised R Vals ==============
'---export Zvals for given values---
'Open "C:\nat.txt" For Output As #1
'    Write #1, "N=14"
'For iPat = 0 To UBound(arrOF_Z_Vals_Grouped, 3)
''
'    Write #1, arrOF_Z_Vals_Grouped(13, 1, iPat)
'
'Next iPat

'    Write #1, "N=12"
'For iPat = 0 To UBound(arrOF_Z_Vals_Grouped, 3)

'    Write #1, arrOF_Z_Vals_Grouped(11, 1, iPat)

'Next iPat

'Close #1
'--------- Init Z Mean array ----
ReDim arrOF_Z_MEAN(Max_Data, 3) As Variant

'----------- Find Mean ---------
For iSum = 0 To UBound(arrOF_Z_MEAN)
   '--reset vars--

   Sum_Total1 = 0
   Sum_Total2 = 0
   Sum_Total3 = 0
   For iPat = 0 To UBound(arrOF_Z_Vals_Grouped, 3)
      Sum_Total1 = Sum_Total1 + arrOF_Z_Vals_Grouped(iSum, 1, iPat)
      Sum_Total2 = Sum_Total2 + arrOF_Z_Vals_Grouped(iSum, 2, iPat)
      Sum_Total3 = Sum_Total3 + arrOF_Z_Vals_Grouped(iSum, 3, iPat)

   Next iPat

   arrOF_Z_MEAN(iSum, 0) = arrOF_Z_Vals_Grouped(iSum, 0, 0)
   arrOF_Z_MEAN(iSum, 1) = Sum_Total1 / iPat
   arrOF_Z_MEAN(iSum, 2) = Sum_Total2 / iPat
   arrOF_Z_MEAN(iSum, 3) = Sum_Total3 / iPat
Next iSum



'======== Do a reverse Z transform for real R's ==========

'------- Init R Mean Array & Ave for SEM-------
ReDim arrOF_R_Mean(UBound(arrOF_Z_MEAN), 3) As Variant



'---------- Fill Arrays -------------
For idata = 0 To UBound(arrOF_R_Mean)
   '-Escape Clause-
   If idata > UBound(arrOF_R_Vals()) Then
      Exit For
   End If
   '------- Reverse the Transform ---------
   'R = Tanh(Z)
   arrOF_R_Mean(idata, 0) = arrOF_Z_MEAN(idata, 0)
   arrOF_R_Mean(idata, 1) = excel.WorksheetFunction.Tanh(arrOF_Z_MEAN(idata, 1))
   arrOF_R_Mean(idata, 2) = excel.WorksheetFunction.Tanh(arrOF_Z_MEAN(idata, 2))
   arrOF_R_Mean(idata, 3) = excel.WorksheetFunction.Tanh(arrOF_Z_MEAN(idata, 3))

Next idata

'==create Array to say if data is real or interpolated==
ReDim arrOF_Reals(UBound(arrOF_R_Mean), 2) As Variant
For i = 0 To UBound(arrOF_Reals)
   arrOF_Reals(i, 2) = "Real"
Next i
'============= Calculate SEM ================

ReDim arrOf_Z_SEM(UBound(arrOF_R_Mean)) As Variant
ReDim arrOf_R_SEM(UBound(arrOF_R_Mean)) As Variant
ReDim arrOf_R_Merge(UBound(arrOF_R_Mean), 1) As Variant

'--Get SEM for Z transformed Cht's--
```

```
    arrOf_Z_SEM = SEM_Group(arrOF_Z_Vals_Grouped, arrOF_Reals)

    '-Z transform back-
    For i = 0 To UBound(arrOf_Z_SEM)
       arrOf_R_SEM(i) = excel.WorksheetFunction.Tanh(arrOf_Z_SEM(i))
    Next i

    arrOf_R_Merge = MergeSEM_Group(arrOf_R_SEM, arrOF_R_Mean)


    ReDim arrOf_R_Merge_Limits(UBound(arrOf_R_Merge), 3) As Variant
    '---add significance bars---
    Dim j As Long, k As Long
    For i = 0 To UBound(arrOf_R_Merge_Limits) Step 5
       For j = 0 To 4
          arrOf_R_Merge_Limits(i + j, 0) = arrOf_R_Merge(i + j, 0)
          arrOf_R_Merge_Limits(i + j, 1) = arrOf_R_Merge(i + j, 1)
          arrOf_R_Merge_Limits(i + j, 2) = arrOF_R_Mean(k, 2)
          arrOf_R_Merge_Limits(i + j, 3) = arrOF_R_Mean(k, 3)
       Next
       k = k + 1
    Next i

    Open "C:\nat.txt" For Output As #2

    For k = 0 To UBound(arrOF_R_Mean)

       Write #2, arrOF_R_Mean(k, 2)

    Next k
    Close #2
    '======Create an Array just for FLX=======
    ReDim arrOf_R_FLX(UBound(arrOF_R_Mean), 2) As Variant

    '#Add Ave#
    For idata = 0 To UBound(arrOf_R_FLX)
       arrOf_R_FLX(idata, 0) = arrOF_R_Mean(idata, 0)
       arrOf_R_FLX(idata, 1) = arrOF_R_Mean(idata, 1)
       arrOf_R_FLX(idata, 2) = arrOf_R_SEM(idata)
    Next idata

    '==================Fill FLX=================
    Call Fill_Flx(arrOf_R_FLX, "Grouped_Autocorrelation")



    '==============Position Controls===============
    Call Format_Controls("Grouped_Autocorrelation", 16, "N", "Mean of R", "SEM")

    '==================Draw CHT=================
    FRMHost.Controls("ChtGrouped_Autocorrelation").ChartData = arrOf_R_Merge
    '======Clean Up=======
    Set FLX = Nothing
    Set FRM = Nothing
    Erase arrOF_R_Vals
    Erase arrOf_Z_Vals
    'Erase arrOf_R_ALL
    Erase arrOF_Z_Vals_Grouped
    Erase arrOF_Z_MEAN
    Erase arrOF_R_Mean
    'Erase arrOF_R_AVE
    'Erase arrOf_R_SEM
    'Erase arrOf_R_Merge
    Erase arrOf_R_FLX

End Sub
```

```
Option Explicit
'This Module is very similar to AutoCorrelation but instead of using the same dataset to _
correlate to it uses a different one. _
1) Checks that 2 Pats only are Selected _
2) Uses the Autocorrelation Function to get Correlation Coefficient array _
3) Reverses the Array so that it appears to have correlated by shifting to the left as well _
4) Displays the Results
Sub Draw_CrossCorrelate()
   '==================Init Vars==================
   Dim FLX As MSFlexGrid
   Dim i As Integer, idata As Long, Middle_of_Data As Integer

   Set FLX = FRMHost.FLXMain

   '====Check that 2 Patients are Selected=======
   If FLX.Rows <> 4 Then
      MsgBox ("Two Patients only must be selected before performing a Cross-Correlation" & vbCr & "Please
»Try again")
      FRMHost.CHKCrossCorrelate.Value = 0
      Exit Sub
   End If
   '=========Call AutoCorrelation Function=======
   Dim arrCorrolate_Right() As Variant
   Dim arrCorrolate_Left() As Variant
   arrCorrolate_Right = AutoCorrelate_Single(True, False)
   arrCorrolate_Left = AutoCorrelate_Single(True, True)
   '=======Reverse Array So we Shift to Left=====


   '===Fing Best Match===
   Dim BestVal As Double, BestI As Integer
   BestVal = arrCorrolate_Right(0, 1)

   For i = 0 To UBound(arrCorrolate_Right)
      If Abs(arrCorrolate_Right(i, 1)) > BestVal Then
         BestVal = Abs(arrCorrolate_Right(i, 1))
         BestI = i
      End If
   Next i



   '============Merge The two Arrays=============
   ReDim arrCorrolate_Merge((UBound(arrCorrolate_Right) * 2) + 1, 3) As Variant
   Middle_of_Data = UBound(arrCorrolate_Right) + 1
   For idata = 0 To UBound(arrCorrolate_Right)
      '#Left Array# 'flip it so it counts down to ag of Zero instead of up from Lag of Zero
      arrCorrolate_Merge(idata, 0) = "N= " & CStr(0 - (arrCorrolate_Left(UBound(arrCorrolate_Left) - idata,
» 0)))
      arrCorrolate_Merge(idata, 1) = arrCorrolate_Left(UBound(arrCorrolate_Left) - idata, 1)
      arrCorrolate_Merge(idata, 2) = arrCorrolate_Left(UBound(arrCorrolate_Left) - idata, 2)
      arrCorrolate_Merge(idata, 3) = arrCorrolate_Left(UBound(arrCorrolate_Left) - idata, 3)

      '#Right Array#
      arrCorrolate_Merge(idata + Middle_of_Data, 0) = "N= " & CStr(arrCorrolate_Right(idata, 0))
      arrCorrolate_Merge(idata + Middle_of_Data, 1) = arrCorrolate_Right(idata, 1)
      arrCorrolate_Merge(idata + Middle_of_Data, 2) = arrCorrolate_Right(idata, 2)
      arrCorrolate_Merge(idata + Middle_of_Data, 3) = arrCorrolate_Right(idata, 3)
   Next idata

   '==================Fill FLX=================
   Call Fill_Flx(arrCorrolate_Merge, "CrossCorrelation")

   '==============Position Controls===============
   Call Format_Controls("CrossCorrelation", 3, "Period", "Power", "Limits")

   '==================Draw CHT=================
   FRMHost.Controls("CHTCrossCorrelation").ChartData = arrCorrolate_Merge

   '====Write best N Value found====
   FRMHost.Controls("CHTCrossCorrelation").Title = "N = " & BestI & " (" & arrCorrolate_Right(BestI, 1) &
»")"

   '======Clean Up=======
   Set FLX = Nothing
   Erase arrCorrolate_Right
```

```
    Erase arrCorrolate_Left
    Erase arrCorrolate_Merge
End Sub
```

```
Option Explicit
' This module deconvolutes the data, ie some assays have long halflifes and remain in _
the body for a while so the module aims to remove that data from subsequent measurements _
1) Set, Checks & Gets the Raw Dataset to start (Perform_Deconv) _
2) Adds some Rubbish Data to start of Data so that we can Deconv Properly , not to sure why
»(Perform_Deconv) _
3) Perform the Deconverlution by Working out the Decay rate of a given reading and (Perform_Deconv) _
4) Subtracting it from the Raw Data (Perform_Deconv) _
5) Returns the Array (Perform_Deconv) _
6) Outputs the FLX and CHT (Draw_Deconv) _


Sub Draw_Deconv()
   '====================Init Vars====================
   Dim arrDeconv_Main() As Variant

   arrDeconv_Main = Perform_Deconv

   '==================Fill FLX=================
   Call Fill_Flx(arrDeconv_Main, "Deconvolution")

   '==============Position Controls===============
   Call Format_Controls("Deconvolution", 1, "Time", "Data")

   '==================Draw CHT=================
   FRMHost.Controls("CHTDeconvolution").ChartData = arrDeconv_Main
End Sub

Function Perform_Deconv(Optional RESET As Boolean) As Variant
   '==================Init Vars====================
   Dim last_Dataitem As Integer, FLX As MSFlexGrid, idata As Long
   Dim Current_Pat As Integer, Assay As String, Answer As String, i As Integer, iDecay As Integer
   Dim Test_Case As Boolean, NegCount As Integer
   Static Half_Life As Double
   Dim continue As Boolean

   Set FLX = FRMHost.FLXMain
   last_Dataitem = Get_Least() - 1 'Find_Last_DataItem(FLX.Row) - 3
   Current_Pat = FLX.Row - 1

   Dim arrMain() As Variant


   '====RESET if Told=====
   If RESET = True Then
      Half_Life = 0
      Exit Function
   End If

   '===============Check Datasets===================
   If Duplicate = True And FRMHost.chkMean.Value = 0 Then
      For i = 1 To FRMHost.OptMpa.UBound
         If FRMHost.OptMpa(i).Value = True Then
            continue = True
            Exit For
         End If
      Next

      If continue = False Then
         MsgBox ("Deconverlution Can only be done on Single Data sets. Please Try again.")
         Perform_Deconv = arrMain
         Exit Function
      End If
   End If

   '==================Get Data=====================
   Dim tmp() As String
   arrMain = What_Flx()
   For idata = 0 To UBound(arrMain)
      tmp = Split(arrMain(idata, 0), " ")
      arrMain(idata, 0) = tmp(0)
   Next

   '===Set up other arrays for analysis===
   ReDim arrMain_Random(UBound(arrMain), 1) As Variant
   ReDim arrMain_Final(UBound(arrMain), 1) As Variant
```

```
'==============Identify Half Life================
Assay = arrRaw(1, Current_Pat)

'#Auto Find Half Life#
Select Case UCase(Assay)
Case "I":
   Half_Life = Insulin_Halflife
Case "INSULIN":
   Half_Life = Insulin_Halflife
Case "INS":
   Half_Life = Insulin_Halflife
Case "C-PEP":
   Half_Life = CPeptide_Halflife
Case "CP":
   Half_Life = CPeptide_Halflife
Case "C":
   Half_Life = CPeptide_Halflife
Case "C-PEPTIDE":
   Half_Life = CPeptide_Halflife
End Select

'#Manually Find Half Life#
'If Half_Life = 0 Then
Do While Test_Case = False
   Answer = InputBox("What is the Half Life of the Assay used?", , Half_Life)
   If Answer <> "" Then
      If IsNumeric(Answer) = True Then
         If Answer <> 0 Then
            Test_Case = True
            Half_Life = CDbl(Answer)
         End If
      End If
   End If
Loop
'End If

'arrMain_Random = arrMain 'QWERTY
'==========Insert Random 10 First Items==========
'#False Data#
'For idata = 0 To 9
'    arrMain_Random(idata, 1) = arrMain(0, 1)
'    arrMain_Random(idata, 0) = arrMain(0, 0)
'Next idata
'
'#Real Data#
For idata = 0 To UBound(arrMain_Random)
   arrMain_Random(idata, 0) = arrMain(idata, 0)
   arrMain_Random(idata, 1) = arrMain(idata, 1)
Next idata


'=============Perform Deconvolution==============
'Y = Y / 2 ^( X / Half )
For idata = 0 To UBound(arrMain_Random)
   '---reset Decay Array---
   ReDim arrMAin_Decayed(UBound(arrMain_Random)) As Variant
   '-------------------

   '#Work out Decay Rate#
   For i = 0 To UBound(arrMAin_Decayed)
      arrMAin_Decayed(i) = arrMain_Random(idata, 1) / (2 ^ (arrMain_Random(i, 0) / Half_Life))

   Next i


   iDecay = 1

   '#Remove Decay Rate from Sum Data#
   For i = (idata + 1) To UBound(arrMAin_Decayed)
      arrMain_Random(i, 1) = arrMain_Random(i, 1) - (arrMAin_Decayed(iDecay))
      iDecay = iDecay + 1
   Next i


   '---reset Decay Array---
   Erase arrMAin_Decayed
   '-------------------
Next idata
```

```
    '============Remove Random 10 items==============
    'For i = 0 To UBound(arrMain_Final)
    '    arrMain_Final(i, 0) = arrMain_Random(i + 10, 0) & " " & Time_Type
    '   arrMain_Final(i, 1) = arrMain_Random(i + 10, 1)
    'Next i

    For i = 0 To UBound(arrMain_Final)
       arrMain_Final(i, 0) = arrMain_Random(i, 0) & " " & Time_Type
       arrMain_Final(i, 1) = arrMain_Random(i, 1)
    Next i
    '===========Remove Negitive Vals===========
    For i = 0 To UBound(arrMain_Final)
       If Sgn(arrMain_Final(i, 1)) = -1 Then
          NegCount = NegCount + 1
          arrMain_Final(i, 1) = 0 'get rid of negative values
       End If
    Next i

    '============Output Neg Count====================
    FRMHost.Controls("chtdeconvolution").Title = "Negative Values = " & NegCount
    '================Return Array====================
    Perform_Deconv = arrMain_Final

    '====================Tidy Up==================
    Set FLX = Nothing
    Erase arrMain
    Erase arrMain_Random
    Erase arrMAin_Decayed
    Erase arrMain_Final

End Function
```

```
Option Explicit
'This module _
1) decides where to get it data from (Draw_Detrend) _
2) it then Detects what Detrending is chosen and goes one of 3 ways (Draw_Detrend) _
3a) One way is Difference + Mean detrend (DET_Difference) _
3b) Another way is End to End Detrend (DET_EndtoEnd) _
3c) The final way is Regressional Detrending in (DET_Reg) _
After doing one of these detrending the module outputs the data and Chart _
 _
Need to find out how to do detrending on duplicate data do we just average ??? as at mo _
we do nothing
Sub Draw_Detrend()
   '=========Init Vars========
   Dim iTab As Integer, MPA_Found As Boolean, FLX As MSFlexGrid, idata As Long, i As Integer

   '=======Find if Mpa tab exists===
   For iTab = 1 To FRMHost.Tabmain.Tabs.Count
      If FRMHost.Tabmain.Tabs(iTab).Image = "Mpa" Then
         MPA_Found = True
      End If
   Next iTab

   '======Decide where to get data from====
   If MPA_Found = True Then
      Set FLX = FRMHost.Controls("FLXMpa")

   Else
      Set FLX = FRMHost.Controls("FLXRaw")
   End If

   '==========Init and Fill Array==========
   ReDim arrDetrend_Start(FLX.Rows - 2, 1) As Variant

   For idata = 0 To UBound(arrDetrend_Start)
      arrDetrend_Start(idata, 0) = FLX.TextMatrix((idata + 1), 0)
      arrDetrend_Start(idata, 1) = FLX.TextMatrix((idata + 1), 1)
   Next idata

   '=============Remove * from Dataset=============
   arrDetrend_Start = Filter_StarsOut(arrDetrend_Start)

   '==============Init Other Det arrays============
   Dim arrDetrend_Finish() As Variant

   '===============Find Selected option============
   For i = 0 To FRMHost.OptDetrend.Count - 1
      If FRMHost.OptDetrend.Item(i).Value = True Then
         ' i = 5
         Select Case i
         Case 1
            arrDetrend_Finish = DET_Difference(arrDetrend_Start())
            Exit For
         Case 2
            arrDetrend_Finish = DET_Reg(arrDetrend_Start())
            Exit For
         Case 3
            arrDetrend_Finish = DET_EndtoEnd(arrDetrend_Start())
            Exit For
         Case 4
            arrDetrend_Finish = DET_19Point(arrDetrend_Start())
            Exit For
         Case 5
            arrDetrend_Finish = DET_Cubic(arrDetrend_Start())
            'Debug.Print DET_Cubic(arrDetrend_Start())
            Exit For
         End Select
      End If
   Next i

   '==================Fill FLX=================
   Call Fill_Flx(arrDetrend_Finish, "Detrended")

   '==============Position Controls===============
   Call Format_Controls("Detrended", 3, "Time", "Detrended Vals")

   '==================Draw CHT=================
   FRMHost.Controls("ChtDetrended").ChartData = arrDetrend_Finish
```

```
   Call Add_Options

End Sub

Function DET_Difference(arrDetrend_Start() As Variant) As Variant
   '==============Init Vars==================
   Dim idata As Long, Total As Long, Mean As Integer, missing As Integer
   ReDim arrDetrend_Diff(UBound(arrDetrend_Start) - 1, 1) As Variant
   '==============Work out Mean==============
   For idata = 0 To UBound(arrDetrend_Start)
      Total = Total + arrDetrend_Start(idata, 1)


   Next idata

   Mean = Total / (UBound(arrDetrend_Start) + 1)

   '==========Perform Differnce + Mean========
   For idata = 0 To UBound(arrDetrend_Diff)

      arrDetrend_Diff(idata, 1) = arrDetrend_Start(idata + 1, 1) - arrDetrend_Start(idata, 1) + Mean
      arrDetrend_Diff(idata, 0) = arrDetrend_Start(idata, 0)

   Next idata

   '=============Return Array==============
   DET_Difference = arrDetrend_Diff()
End Function

Function DET_EndtoEnd(arrDetrend_Start() As Variant) As Variant
   '==============Init Vars==============
   Dim idata As Long, Slope As Double, Slope_at_Time As Double
   ReDim arrDetrend_EndtoEnd(UBound(arrDetrend_Start), 1) As Variant

   '=======Work Out Slope of Data=======
   ' (last - first / N) + first = SLOPE
   Slope = (arrDetrend_Start(UBound(arrDetrend_Start), 1) - arrDetrend_Start(0, 1))
   Slope = Slope / (UBound(arrDetrend_Start) + 1)
   'Slope = Slope + arrDetrend_Start(0, 1)

   '============Do Data - Slope=========
   For idata = 0 To UBound(arrDetrend_EndtoEnd)
      arrDetrend_EndtoEnd(idata, 0) = arrDetrend_Start(idata, 0)
      arrDetrend_EndtoEnd(idata, 1) = arrDetrend_Start(idata, 1) - (Slope * idata * Time_Val)

   Next idata
   '=============Return Array===========

   DET_EndtoEnd = arrDetrend_EndtoEnd()


End Function

Function DET_Reg(arrDetrend_Reg() As Variant) As Variant
   '======================Init Vars====================
   Dim TotalTime As Long, iTime As Integer, Bestfit As Double
   Dim MeanX As Double, idata As Long, TotalBottom As Double
   Dim TotalData As Double, MeanY As Double, TotalTop As Double

   ReDim arrX(UBound(arrDetrend_Reg)) As Double
   ReDim arrY(UBound(arrDetrend_Reg)) As Double

   '-----------------------------------------------------
   ' Hard maths bit to calculate trend line
   ' sum of (x minus mean of x time y minus mean of y)
   ' all over the sum of (x minus the mean of x squared)
   '-----------------------------------------------------

   '==================Calculate X=======================
   For iTime = 0 To UBound(arrDetrend_Reg())
      TotalTime = TotalTime + iTime * Time_Val
   Next iTime
   '---------------Mean of Time (X)--------------------
   MeanX = TotalTime / (UBound(arrDetrend_Reg()) + 1)


   '---------------X minus Mean X----------------------
```

```
   For iTime = 0 To UBound(arrX)
      arrX(iTime) = (iTime * Time_Val) - MeanX
   Next iTime


   '==================Calculate Y=======================
   For idata = 0 To UBound(arrY)
      TotalData = TotalData + arrDetrend_Reg(idata, 1)
   Next idata
   '------------------Mean of Data (Y)------------------
   MeanY = TotalData / (UBound(arrDetrend_Reg()) + 1)

   '-----------------Y minus mean Y--------------------
   For idata = 0 To UBound(arrY)
      arrY(idata) = arrDetrend_Reg(idata, 1) - MeanY
   Next idata

   '================Sum of (X-Xm)(Y-Ym)================
   For idata = 0 To UBound(arrX)
      TotalTop = TotalTop + (arrX(idata) * arrY(idata))
   Next idata


   '================Sum of (X-Xm)(X-Xm)================
   For idata = 0 To UBound(arrX)
      TotalBottom = TotalBottom + arrX(idata) * arrX(idata)
   Next idata

   '====================Top over Bottom=================
   Bestfit = TotalTop / TotalBottom


   '======================Detrend array=================
   For idata = 0 To UBound(arrDetrend_Reg())
      arrDetrend_Reg(idata, 1) = _
            arrDetrend_Reg(idata, 1) - (Bestfit * Time_Val * (idata + 1))

   Next idata

   '==================Return Array=======================
   DET_Reg = arrDetrend_Reg()

End Function

Private Function DET_19Point(arrDetrend_Start() As Variant) As Variant
   '------Init Vars-------
   ReDim arrDetrend(UBound(arrDetrend_Start) - 18, 1) As Variant
   Dim iRange As Integer, idata As Long, iNo As Integer
   Dim Total As Double

   iRange = Round((19 - 1) / 2)

   '----Check Array is big enough----------
   If UBound(arrDetrend_Start) < 23 Then
      Call MsgBox("Array length is too short for this type of detrending", vbApplicationModal, "Easy TSA")
      DET_19Point = arrDetrend_Start
      Exit Function
   End If

   '-----Get Detrended Data----
   '========Do moving Average==========
   For idata = iRange To UBound(arrDetrend_Start) - iRange

      '-----Add Data up-----
      For iNo = -iRange To iRange
         Total = Total + arrDetrend_Start(iNo + idata, 1)
      Next iNo

      '-----Average it------
      arrDetrend((idata - iRange), 1) = Total / 19
      arrDetrend(idata - iRange, 0) = idata * Time_Val & " " & Time_Type
      Total = 0

   Next idata

   '-----Return Array------
   For iNo = 0 To UBound(arrDetrend)
      '    arrDetrend(iNo, 0) = arrDetrend_Start(iNo, 0)
```

```
      arrDetrend(iNo, 1) = arrDetrend_Start(iNo + 9, 1) - arrDetrend(iNo, 1)
   Next
   DET_19Point = arrDetrend()

   '-----Clean Up------
   Erase arrDetrend

End Function
```

```
Option Explicit

Sub Draw_Fourier(ByVal Normalise As Boolean)

   '==================Init Vars==================
   Dim PI As Double, Quarter As Double
   Dim iCount As Long, idata As Long, SumC As Double, SumS As Double, tmp As Double, Frequency As Double
   Dim SumTotal As Double, WhatToDo As String, i As Integer
   Dim Norm_Abs_Title As String


   '==========Init and Fill Array==========
   Dim arrFourier_Start() As Variant
   arrFourier_Start = What_Flx()


   '=========Init Pi and End of Dataset=========
   PI = 4 * Atn(1)
   'Quarter = Round((UBound(arrFourier_Start) + 1) / 4)
   Quarter = Round((UBound(arrFourier_Start) + 1) / 2)
   ReDim arrFourier_End(Quarter - 1, 1) As Variant
   ReDim arrFourier_Norm(UBound(arrFourier_End), 1) As Variant
   ReDim arrFourier_Final(UBound(arrFourier_End), 1) As Variant

   '========Work Out Fourier Transform======
   iCount = 1
   Do While iCount <= Quarter
      SumC = 0
      SumS = 0

      For idata = 1 To UBound(arrFourier_Start) + 1
         Frequency = idata / (UBound(arrFourier_Start) + 1)
         SumC = SumC + (Cos(2 * PI * iCount * Frequency) * arrFourier_Start(idata - 1, 1)) 'Frequency item
»is the current frequency from 1 to n/4
         SumS = SumS + (Sin(2 * PI * iCount * Frequency) * arrFourier_Start(idata - 1, 1))
      Next idata


      tmp = (SumC * SumC) + (SumS * SumS)
      arrFourier_End(iCount - 1, 1) = Sqr(tmp) / (UBound(arrFourier_Start) + 1)
      arrFourier_End(iCount - 1, 0) = "Freq = " & iCount
      iCount = iCount + 1
   Loop


   If Normalise = True Then
      '===========Normalise the Data=========

      '#Work out whether to Multiply up or Down to get to 100%#
      For idata = 0 To UBound(arrFourier_End)
         SumTotal = SumTotal + arrFourier_End(idata, 1)
      Next idata

      If SumTotal > 100 Then
         WhatToDo = "divide"
      Else
         WhatToDo = "multiply"
      End If

      'FRMHost.lblblank1.Caption = "100% = " & Sumtotal FIX THIS

      SumTotal = SumTotal / 100
      '#########################################################


      '####################Normalise it#########################
      For i = 0 To UBound(arrFourier_Norm)
         arrFourier_Norm(i, 0) = arrFourier_End(i, 0)

         '------Divide 0/0 Fix------------
         If arrFourier_End(i, 1) = 0 And SumTotal = 0 Then Exit For

         If WhatToDo = "multiply" Then
            If SumTotal < 1 Then
               arrFourier_Norm(i, 1) = arrFourier_End(i, 1) / SumTotal
            Else
```

```
            arrFourier_Norm(i, 1) = arrFourier_End(i, 1) * SumTotal
         End If

      Else
         arrFourier_Norm(i, 1) = arrFourier_End(i, 1) / SumTotal
      End If

   Next i
   '#########################################################
```

```
        '=====Output for CHT and FLX======
        arrFourier_Final = arrFourier_Norm
        Norm_Abs_Title = "Normalised Vals (%)"

    Else
        arrFourier_Final = arrFourier_End
        Norm_Abs_Title = "Absolute Values"

    End If

    '==================Fill FLX=================
    Call Fill_Flx(arrFourier_Final, "Fourier")

    '==============Position Controls===============
    Call Format_Controls("Fourier", 1, "Frequency", Norm_Abs_Title)

    '==================Draw CHT=================
    FRMHost.Controls("ChtFourier").ChartData = arrFourier_Final
    FRMHost.Controls("CHTFourier").Title = Norm_Abs_Title


End Sub
```

```
Option Explicit
'This module is really two disctinct parts: _
Singular : _
1) We decide on 3/5/ or X point moving ave (MPA_Selected) _
2) We perform the moving ave (MPA_Single) _
3) We work out the Standard Error of the Mean (SEM) _
4) We merge the two results and display the graph (Merge_Sem) _
Duplicate : _
1) We decide on 3/5/x moving point average (MPA_Selected) _
2) We perform the moving ave (MPA_Duplicate) _
3) We work out the Standard Error of the Mean (SEM) _
4) We merge the two results and display the graph (Merge_Sem) _
 _
Need to check if SEM on duplicate data is same on Singular though !!!!


Sub Draw_MPA()
   Call MPA_Selected
   Call Add_Options
End Sub
Sub MPA_Selected()
   '=====================Init Vars=================
   Dim i As Long, n As Integer, iRange As Long, idata As Long
   Dim tmp As String, Streams As Long, last_Dataitem As Long, Total_SEM_Data As Long, Total_NO_SEM_Data As
»Long
   last_Dataitem = Find_Last_DataItem(FRMHost.FLXMain.Row) - 3

   '===============Find Selected option============
   For i = 0 To FRMHost.OptMpa.Count - 1
      If FRMHost.OptMpa.Item(i).Value = True Then
         Select Case i
         Case 1
            n = 3
         Case 2
            n = 5
         Case 3
            Do While IsNumeric(tmp) = False
               tmp = InputBox("How many points in the moving average?")
               If tmp = "2" Then tmp = "Can not do two dude"
            Loop
            n = CInt(tmp)
            x_Point_MPA = n
         End Select
      End If
   Next i

   '================Init Arrays==================
   ReDim arrMPA_Ave(last_Dataitem - (n - 1), 1) As Variant
   ReDim arrMpa_SEM(UBound(arrMPA_Ave) - 2) As Double
   Total_SEM_Data = UBound(arrMpa_SEM) * 5
   Total_NO_SEM_Data = n - 2
   ReDim arrMPA_ALL(Total_SEM_Data + Total_NO_SEM_Data, 1) As Double
   ReDim arrMPA_FLX(last_Dataitem - (n - 1), 2) As Variant

   '======Perform Moving Average Functions======
   If Duplicate = False Then
      '----- Get Averaged Data----
      arrMPA_Ave = MPA_Single(n)
   Else
      '----- Get Averaged Data----
      arrMPA_Ave = MPA_Duplicate(n)
   End If
   '=========Work out SEM and add it on=======
   arrMpa_SEM = SEM(arrMPA_Ave, n)
   arrMPA_ALL = MergeSEM(arrMPA_Ave(), arrMpa_SEM(), n)

   '======Create on Array just for FLX=======
   iRange = CInt(Round((n - 1) / 2))

   '#Add Ave#
   For idata = 0 To UBound(arrMPA_Ave)
      arrMPA_FLX(idata, 0) = arrMPA_Ave(idata, 0)
      arrMPA_FLX(idata, 1) = arrMPA_Ave(idata, 1)
   Next idata

   '#Add SEM#
```

```
For idata = iRange To UBound(arrMPA_Ave) - iRange
   arrMPA_FLX(idata, 2) = arrMpa_SEM(idata - iRange)
Next idata
Debug.Print FRMHost.FLXMain.Row
Call Fill_Flx(arrMPA_FLX, "Mpa")

'==============Position Controls===============
Call Format_Controls("Mpa", 16, "Time", "Ave. Vals", "SEM")
```

```
   '=================Draw CHT================
   FRMHost.Controls("ChtMpa").ChartData = arrMPA_ALL


End Sub
Function MPA_Single(ByRef n As Integer) As Variant
   '===========Init Vars===========
   Dim idata As Long, iRange As Integer, last_Dataitem As Long, Current_Pat As Integer
   Dim iNo As Integer, Total As Double
   Dim patname As String, i As Integer

   last_Dataitem = Find_Last_DataItem(FRMHost.FLXMain.Row) - 3

   '-select diff curr pat if using grouped data-
   If FRMHost.CHKGrpAutocorrelation.Value = 1 Or FRMHost.CHKGrpPowerPeriod.Value = 1 Or
»FRMHost.ChkPermutation.Value = 1 Then
      patname = FRMHost.FLXMain.TextMatrix(FRMHost.FLXMain.Row, 0)

      For i = 0 To UBound(arrRaw, 2)
         If patname = arrRaw(0, i) Then
            Current_Pat = i
         End If
      Next i
   Else
      Current_Pat = (FRMHost.FLXMain.Row - 1)
   End If

   '===============Init Arrays===============
   ReDim arrMPA(last_Dataitem, 1) As Variant
   ReDim arrMPA_Ave(last_Dataitem - (n - 1), 1) As Variant

   '============Copy relevant data==============

   For idata = 0 To last_Dataitem
      arrMPA(idata, 0) = (Time_Val * idata) + 1 & " " & Time_Type
      arrMPA(idata, 1) = arrRaw(idata + 3, Current_Pat)
   Next idata


   arrMPA = Filter_StarsOut(arrMPA)

   '======Work out Range of Mpa====
   iRange = Round((n - 1) / 2)

   '========Do moving Average==========
   For idata = iRange To UBound(arrMPA) - iRange

      '-----Add Data up-----
      For iNo = -iRange To iRange
         Total = Total + arrMPA(iNo + idata, 1)
      Next iNo
      '-----Average it------
      arrMPA_Ave((idata - iRange), 1) = Total / n
      arrMPA_Ave(idata - iRange, 0) = idata * Time_Val & " " & Time_Type
      Total = 0
   Next idata



   '=========Return Array============
   MPA_Single = arrMPA_Ave()
End Function

Function MPA_Duplicate(ByRef n As Integer) As Variant
   '===========Init Vars===========
   Dim idata As Long, iRange As Integer, last_Dataitem As Long, Current_Pat As Integer
   Dim Star_Found As Boolean, iStar As Integer
   Dim iNo As Integer, Total As Double
   Dim patname As String, i As Integer
   last_Dataitem = Find_Last_DataItem(FRMHost.FLXMain.Row) - 3

   '-select diff curr pat if using grouped data-
   If FRMHost.CHKGrpAutocorrelation.Value = 1 Or FRMHost.CHKGrpPowerPeriod.Value = 1 Or
»FRMHost.ChkPermutation.Value = 1 Then
      patname = FRMHost.FLXMain.TextMatrix(FRMHost.FLXMain.Row, 0)
```

```
      For i = 0 To UBound(arrRaw, 2)
         If patname = arrRaw(0, i) Then
            Current_Pat = i
         End If
      Next i
   Else
      Current_Pat = (FRMHost.FLXMain.RowSel - 1)
   End If



   '===============Init Arrays===============
   ReDim arrMPA(last_Dataitem, 2) As Variant
   ReDim arrMPA_Ave(last_Dataitem - (n - 1), 1) As Variant


   '============Copy relevant data==============
   For idata = 0 To last_Dataitem
      arrMPA(idata, 0) = (Time_Val * idata) + 1 & " " & Time_Type
      arrMPA(idata, 1) = arrRaw(idata + 3, Current_Pat)
      arrMPA(idata, 2) = arrRaw(idata + 3, Current_Pat + Dupe_data_Start)
   Next idata

   '======Work out Range of Mpa====
   iRange = Round((n - 1) / 2)

   '========Do Moving Average===========
   For idata = iRange To UBound(arrMPA) - iRange
      '--------Reinit Vars-------
      Total = 0
      Star_Found = False
      '-----Detect if duplicate data is interpolated---
      For iStar = -iRange To iRange
         If Right(arrMPA(idata + iStar, 2), 1) = "*" Then
            Star_Found = True
            Exit For
         End If
      Next iStar
      '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      '#Use Only main Data set#
      If Star_Found = True Then
         '-----Add Data up-----
         For iNo = -iRange To iRange
            '#Strip out stars#
            If Right(arrMPA(iNo + idata, 1), 1) = "*" Then
               arrMPA(iNo + idata, 1) = Left(arrMPA(iNo + idata, 1), Len(arrMPA(iNo + idata, 1)) - 1)
            End If
            '#################

            Total = Total + arrMPA(iNo + idata, 1)
         Next iNo
         '-----Average it------
         arrMPA_Ave((idata - iRange), 1) = Total / n
         arrMPA_Ave(idata - iRange, 0) = idata * Time_Val & " " & Time_Type
         '#Use Both Datasets#
      Else
         '-----Add Data up-----
         For iNo = -iRange To iRange
            '#Strip out stars#
            If Right(arrMPA(iNo + idata, 1), 1) = "*" Then
               arrMPA(iNo + idata, 1) = Left(arrMPA(iNo + idata, 1), Len(arrMPA(iNo + idata, 1)) - 1)
            End If
            '###################
            Total = Total + IIf(arrMPA(iNo + idata, 1) = "", 0, arrMPA(iNo + idata, 1))
            Total = Total + IIf(arrMPA(iNo + idata, 2) = "", 0, arrMPA(iNo + idata, 2))
         Next iNo
         '-----Average it------
         arrMPA_Ave((idata - iRange), 1) = Total / (n * 2)
         arrMPA_Ave(idata - iRange, 0) = idata * Time_Val & " " & Time_Type
      End If

   Next idata

   '=========Return Array============
   MPA_Duplicate = arrMPA_Ave()
End Function
```

```
Option Explicit

Sub Draw_ObservedConcentration()
   '============Init Vars==========
   Dim arrOC_Start() As Variant, arrOC_Log() As Variant, arrBins_Vars() As String
   Dim arrOC_Bins() As Variant, arrOC_CumFreq() As Variant, idata As Long
   Dim arrOC_Probits() As Variant, L_Limit As Double, U_Limit As Double
   Dim arrOC_Probits_NOLOG() As Variant
   Dim i As Integer, uDiscard As Integer, lDiscard As Integer, iPat As Integer
   '=========Get Data from FLX======
   arrOC_Start = What_Flx()

   '==========Log the Data==========
   arrOC_Log() = arrOC_Start()

   For idata = 0 To UBound(arrOC_Log)


      arrOC_Log(idata, 1) = Log10(arrOC_Log(idata, 1))
      ' Debug.Print arrOC_Log(iData, 1)
   Next idata

   '==========Choose Bins===========
   'Returns = No_of_bins|Bin Size|Minimum|Maximum
   arrBins_Vars() = Split(Choose_Bins(arrOC_Log), "|")

   '=========Assign To Bins=========
   arrOC_Bins = Assign_Bins(arrOC_Log, arrBins_Vars)

   '========Calculate Frequency=====
   arrOC_CumFreq = Get_Cumalative_Frequency(arrOC_Bins)

   '=========Calculate Limits=======
   '[200/n]% to 100-[200/n]% ie 2% to 98%
   L_Limit = (200 / UBound(arrOC_Log))
   U_Limit = 100 - L_Limit

   '===Discard Bins Outside Limits==
   '---Get How many to Discard---
   uDiscard = 0
   lDiscard = 0

   For i = 0 To UBound(arrOC_CumFreq)
      If arrOC_CumFreq(i) < L_Limit Then
         lDiscard = lDiscard + 1
      ElseIf arrOC_CumFreq(i) > U_Limit Then
         uDiscard = uDiscard + 1
      End If
   Next i
   '---Init New Array----
   'ReDim arrOC_CumFreq_Short(UBound(arrOC_CumFreq) - iDiscard, 1)
   ReDim arrOC_CumFreq_Short(UBound(arrOC_CumFreq) - (lDiscard + uDiscard), 1)
   '------Fill Array-----
   For i = 0 To UBound(arrOC_CumFreq)
      If arrOC_CumFreq(i) < L_Limit Then

      ElseIf arrOC_CumFreq(i) > U_Limit Then

      Else
         arrOC_CumFreq_Short(iPat, 0) = arrBins_Vars(1) * (i + 0.5) + arrBins_Vars(2) 'binsize/2 + minimum
         arrOC_CumFreq_Short(iPat, 1) = arrOC_CumFreq(i)
         iPat = iPat + 1
      End If
   Next i

   '----Update Number of Bins----
   arrBins_Vars(0) = iPat
   '======Find Correct Probits=====
   arrOC_Probits = Get_Probits(arrOC_CumFreq_Short, arrBins_Vars())

   '===log for display===
   'ReDim arrOC_Probits_LOG(UBound(arrOC_CumFreq_Short), UBound(arrOC_CumFreq_Short, 2))
   'For i = 0 To UBound(arrOC_CumFreq_Short)
   '    arrOC_Probits_LOG(i, 1) = Log10(arrOC_CumFreq_Short(i, 1))
   '    arrOC_Probits_LOG(i, 0) = arrOC_CumFreq_Short(i, 0)
   'Next i
```

```
   '===========Fill FLX=============
   Call Fill_Flx(arrOC_Probits, "ObservedConcentration")

   '=======Position Controls========
   Call Format_Controls("ObservedConcentration", 16, "Bin Mean", "Probit")

   '==========Draw CHT==============
   FRMHost.Controls("ChtObservedConcentration").ChartData = arrOC_Probits

   '========Add OC Title to CHT=====
   Call Update_Graph
End Sub


Private Function Choose_Bins(ByRef arrLog() As Variant) As String
   '=============Init Vars==============
   Dim Dblminimum As Double, iCount As Integer
   Dim Dblmaximum As Double, Minimum As Integer, Maximum As Integer
   Dim Scope As Double, Bincount As Double, Noofbins As Integer, Bin_Size As Double

   Dblminimum = arrLog(0, 1)
   Dblmaximum = arrLog(0, 1)
   '====Find Largest and Smallest Vals====

   '----Find----
   For iCount = 0 To UBound(arrLog)
      If arrLog(iCount, 1) > Dblmaximum Then
         Dblmaximum = arrLog(iCount, 1)
      End If

      If arrLog(iCount, 1) < Dblminimum Then
         Dblminimum = arrLog(iCount, 1)
      End If
   Next iCount

   '----Format----
   'Maximum = Ceiling(Dblmaximum)
   'Minimum = Floor(Dblminimum)

   '==========Define Number of Bins to Use========
   'If we have less than 6 points away Then just use a few bins
   'ElseIf we have lots of Difference Then use lots of bins
   'Else find a number between 6 and and 16 bins
   '----Init Vars----
   'Scope = Maximum - Minimum
   'Recheck_Bins:

   '----Define Bins----
   'If Scope <= 16 Then
   '
   '    If Scope > 6 Then
   '        Noofbins = Ceiling(Scope)
   '        Bin_Size = (Maximum - Minimum) / Noofbins
   '    Else
   '        Bincount = Scope
   '        Do While Bincount <= 6
   '            Bincount = Bincount * 2
   '        Loop
   '        Scope = Bincount
   '        GoTo Recheck_Bins
   '    End If
   '
   'Else
   '
   '    Bincount = Scope
   '    Do While Bincount > 16
   '        Bincount = Bincount / 2
   '    Loop
   '    Scope = Bincount
   '    GoTo Recheck_Bins
   '
   'End If
   Noofbins = 16
   If Dblminimum < 0 Then Dblminimum = 0
   Bin_Size = (Dblmaximum - Dblminimum) / Noofbins
   '========Return========
```

```
   Choose_Bins = Noofbins & "|" & Bin_Size & "|" & Dblminimum & "|" & Dblmaximum
End Function
Private Function Assign_Bins(ByRef arrLog() As Variant, ByRef arrBin_Vars() As String) As Variant
   '=========Init Vars========
   Dim No_of_Bins As Integer, Size_of_Bins As Double, i As Integer
   Dim idata As Long, iBin As Integer, Minimum As Double
   Dim UpperLimitofBin As Double
   No_of_Bins = arrBin_Vars(0)
   Size_of_Bins = arrBin_Vars(1)
   Minimum = arrBin_Vars(2)

   ReDim arrBin_Vals(UBound(arrLog), No_of_Bins - 1) As Variant

   '========Assign Data to Bin======
   For idata = 0 To UBound(arrLog)
      For iBin = 1 To No_of_Bins
         UpperLimitofBin = (Minimum + (Size_of_Bins * iBin))
         If arrLog(idata, 1) <= UpperLimitofBin Then
            '---find next empty---
            For i = 0 To UBound(arrBin_Vals)
               If arrBin_Vals(i, iBin - 1) = "" Then
                  Exit For
               End If
            Next i

            '-----Insert Data-----
            arrBin_Vals(i, iBin - 1) = arrLog(idata, 1)
            Exit For
         End If
      Next iBin
   Next idata

   '==========Return=========
   Assign_Bins = arrBin_Vals()
   '=========Clean Up========
   Erase arrBin_Vals
End Function

Private Function Get_Cumalative_Frequency(ByRef arrBins_Vals() As Variant) As Variant
   '========Init Vars=======
   Dim iBin As Integer, iCount As Integer, iTotal_Count As Integer

   ReDim arrBins_Freq(UBound(arrBins_Vals, 2)) As Double
   ReDim arrBins_CumFreq(UBound(arrBins_Vals, 2)) As Variant
   '======Set Frequency Per Bin========

   '----Count of Data Items per Bin---
   For iBin = 0 To UBound(arrBins_Freq)
      iCount = 0
      Do While arrBins_Vals(iCount, iBin) <> ""
         iCount = iCount + 1
      Loop
      arrBins_Freq(iBin) = iCount
      iTotal_Count = iTotal_Count + iCount
   Next iBin

   '--log this data--
   'For iBin = 0 To UBound(arrBins_Freq)
   '    arrBins_Freq(iBin) = 10 ^ (arrBins_Freq(iBin))
   'Next


   '----% Freq of Data Items per Bin---
   For iBin = 0 To UBound(arrBins_Freq)
      arrBins_Freq(iBin) = (arrBins_Freq(iBin) / iTotal_Count) * 100
   Next iBin


   '=======Calculate Cumalative Frequency====
   '----Set First Value----
   arrBins_CumFreq(0) = arrBins_Freq(0)
   '---Set Rest of Vals----
   For iBin = 1 To UBound(arrBins_CumFreq)
      arrBins_CumFreq(iBin) = arrBins_CumFreq(iBin - 1) + arrBins_Freq(iBin)
   Next iBin

   '--Adjust all 100% vals to 99.9%===
   For iBin = 0 To UBound(arrBins_CumFreq)
```

```
      If CInt(arrBins_CumFreq(iBin)) = 100 Then
         arrBins_CumFreq(iBin) = 99.9
      End If
   Next iBin

   '======Return======
   Get_Cumalative_Frequency = arrBins_CumFreq()

   '======Clean Up=====
   Erase arrBins_Freq
   Erase arrBins_CumFreq
End Function


Private Function Get_Probits(ByRef arrBins_CumFreq() As Variant, ByRef arrBin_Vars() As String) As Variant
   '============Init Vars============
   Dim No_of_Bins As Integer, iBin As Integer, Size_of_Bins As Double, Minimum As Double

   No_of_Bins = arrBin_Vars(0)
   Size_of_Bins = arrBin_Vars(1)
   Minimum = arrBin_Vars(2)

   ReDim arrbins_probits(No_of_Bins - 1, 1) As Variant

   '======copy Means of Bins====
   For iBin = 0 To UBound(arrbins_probits)
      arrbins_probits(iBin, 0) = arrBins_CumFreq(iBin, 0)
   Next iBin

   '=======Get & Set Probit Vals======
   For iBin = 0 To UBound(arrbins_probits)
      arrbins_probits(iBin, 1) = Return_Probit(CDbl(arrBins_CumFreq(iBin, 1)))
   Next iBin

   '===============Return=============
   Get_Probits = arrbins_probits()
   Dim i As Integer
   For i = 0 To UBound(arrbins_probits)
      Debug.Print (arrbins_probits(i, 1))
   Next
   '============Clean Up==============
   Erase arrbins_probits

End Function

Private Sub Update_Graph()
   '===========Init Vars===========
   Dim CHT As MSChart, tmp() As String, tmp2() As String
   Dim Gradient As Double, Intercept As Double, arrOC_Vals(2) As Double

   Set CHT = FRMHost.Controls("ChtObservedConcentration")
   '===========Draw Graph==========
   Call Select_Tab("ObservedConcentration")
   Call Print_Out(False, True)
   excel.Application.Visible = True
   Sheets(1).Select
   '==========Add Trend Line=======
   ActiveChart.SeriesCollection(1).Select

   ActiveChart.SeriesCollection(1).Trendlines.Add(Type:=xlLinear, Forward:=0, _
         Backward:=0, DisplayEquation:=True, DisplayRSquared:=False).Select

   '=========Get Equation==========
   tmp = Split(ActiveChart.SeriesCollection(1).Trendlines(1).DataLabel.Text, "x")
   tmp2 = Split(tmp(0), "=")
   Gradient = CDbl(tmp2(1))
   Intercept = CDbl(tmp(1))

   '=======Reverse Equation=======
   'Y= Mx+C to X = (Y-C)/M

   '========Calculate X from Y======
   '---- Y = 5% 50% & 95% -----
   '---- Y = 3.3551, 5.0, 6.6449
   'arrOC_Vals(0) = ((3.3551 - Intercept) / (Gradient))
   'arrOC_Vals(1) = ((5 - Intercept) / Gradient)
   'arrOC_Vals(2) = ((6.6449 - Intercept) / Gradient)
```

```
    arrOC_Vals(0) = 10 ^ ((3.3551 - Intercept) / (Gradient))
    arrOC_Vals(1) = 10 ^ ((5 - Intercept) / Gradient)
    arrOC_Vals(2) = 10 ^ ((6.6449 - Intercept) / Gradient)

    '=====Add OC to Graph Title======
    CHT.Title = "5% = " & arrOC_Vals(0) & _
          " 50% = " & arrOC_Vals(1) & _
          " 95% = " & arrOC_Vals(2)
    'MsgBox (Gradient)
    '============Clean Up============
    Erase tmp
    Erase tmp2
    Erase arrOC_Vals
    Set CHT = Nothing
    excel.Application.DisplayAlerts = False
    excel.Workbooks.Close
    excel.Application.Quit
    excel.Application.DisplayAlerts = True

End Sub


Private Function Return_Probit(ByRef ipos As Double) As Double
    '========Init Var===========
    Dim arrProbit() As Variant
    arrProbit = Array( _
          0, 1.91, 2.12, 2.25, 2.35, 2.42, 2.49, 2.54, 2.59, 2.63, 2.67, 2.71, 2.74, 2.77, 2.8, 2.83, 2.86,
»2.88, 2.9, 2.93, 2.95, 2.97, 2.99, 3, 3.02, 3.04, 3.06, 3.07, 3.09, 3.1, 3.12, 3.13, 3.15, 3.16,
» 3.17, 3.19, 3.2, 3.21, 3.23, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.3, 3.32, 3.33, 3.34, 3.35, 3
».36, 3.36, 3.37, 3.38, 3.39, 3.4, 3.41, 3.42, 3.43, 3.44, 3.45, 3.45, 3.46, 3.47, 3.48, 3.49, 3.
»49, 3.5, 3.51, 3.52, 3.52, 3.53, 3.54, 3.55, 3.55, 3.56, 3.57, 3.57, 3.58, 3.59, 3.59, 3.6, 3.61
», 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.65, 3.66, 3.67, 3.67, 3.68, 3.68, 3.69, 3.7, 3.7, 3.71,
»3.71, 3.72, 3.72, 3.73, 3.74, 3.74, 3.75, 3.75, 3.76, 3.76, 3.77, 3.77, 3.78, 3.78, 3.79, 3.79,
»3.8, 3.8, 3.81, 3.81, 3.82, 3.83, 3.83, 3.83, 3.84, 3.84, 3.85, 3.85, 3.86, 3.86, 3.87, 3.87, 3.
»88, 3.88, 3.89, 3.89, 3.9, 3.9, 3.91, 3.91, 3.92, 3.92, 3.92, 3.93, 3.93, 3.94, 3.94, 3.95, 3.95
», 3.95, 3.96, 3.96, 3.97, 3.97, 3.98, 3.98, 3.98, 3.99, 3.99, 4, 4, 4.01, 4.01, 4.01, 4.02, 4.02
», 4.03, 4.03, 4.03, 4.04, 4.04, _
          4.05, 4.05, 4.05, 4.06, 4.06, 4.07, 4.07, 4.07, 4.08, 4.08, 4.08, 4.09, 4.09, 4.1, 4.1, 4.1, 4.11,
» 4.11, 4.11, 4.12, 4.12, 4.13, 4.13, 4.13, 4.14, 4.14, 4.14, 4.15, 4.15, 4.15, 4.16, 4.16, 4.17,
» 4.17, 4.17, 4.18, 4.18, 4.18, 4.19, 4.19, 4.19, 4.2, 4.2, 4.2, 4.21, 4.21, 4.21, 4.22, 4.22, 4.
»22, 4.23, 4.23, 4.23, 4.24, 4.24, 4.24, 4.25, 4.25, 4.25, 4.26, 4.26, 4.26, 4.27, 4.27, 4.27, 4.
»28, 4.28, 4.28, 4.29, 4.29, 4.29, 4.3, 4.3, 4.3, 4.31, 4.31, 4.31, 4.32, 4.32, 4.32, 4.33, 4.33,
» 4.33, 4.33, 4.34, 4.34, 4.34, 4.35, 4.35, 4.35, 4.36, 4.36, 4.36, 4.37, 4.37, 4.37, 4.38, 4.38,
» 4.38, 4.38, 4.39, 4.39, 4.39, 4.4, 4.4, 4.4, 4.41, 4.41, 4.41, 4.41, 4.42, 4.42, 4.42, 4.43, 4.
»43, 4.43, 4.43, 4.44, 4.44, 4.44, 4.45, 4.45, 4.45, 4.46, 4.46, 4.46, 4.46, 4.47, 4.47, 4.47, 4.
»48, 4.48, 4.48, 4.48, 4.49, 4.49, 4.49, 4.5, 4.5, 4.5, 4.5, 4.51, 4.51, 4.51, 4.52, 4.52, 4.52,
»4.52, 4.53, 4.53, 4.53, 4.54, 4.54, 4.54, 4.54, 4.55, 4.55, 4.55, 4.55, 4.56, 4.56, 4.56, 4.57,
»4.57, 4.57, 4.57, 4.58, 4.58, 4.58, _
          4.58, 4.59, 4.59, 4.59, 4.6, 4.6, 4.6, 4.6, 4.61, 4.61, 4.61, 4.61, 4.62, 4.62, 4.62, 4.63, 4.63,
»4.63, 4.63, 4.64, 4.64, 4.64, 4.64, 4.65, 4.65, 4.65, 4.65, 4.66, 4.66, 4.66, 4.67, 4.67, 4.67,
»4.67, 4.68, 4.68, 4.68, 4.68, 4.69, 4.69, 4.69, 4.69, 4.7, 4.7, 4.7, 4.71, 4.71, 4.71, 4.71, 4.
»72, 4.72, 4.72, 4.72, 4.73, 4.73, 4.73, 4.73, 4.74, 4.74, 4.74, 4.74, 4.75, 4.75, 4.75, 4.75, 4.
»76, 4.76, 4.76, 4.76, 4.77, 4.77, 4.77, 4.78, 4.78, 4.78, 4.78, 4.79, 4.79, 4.79, 4.79, 4.8, 4.8
», 4.8, 4.8, 4.81, 4.81, 4.81, 4.81, 4.82, 4.82, 4.82, 4.82, 4.83, 4.83, 4.83, 4.83, 4.84, 4.84,
»4.84, 4.84, 4.85, 4.85, 4.85, 4.85, 4.86, 4.86, 4.86, 4.86, 4.87, 4.87, 4.87, 4.87, 4.88, 4.88,
»4.88, 4.88, 4.89, 4.89, 4.89, 4.89, 4.9, 4.9, 4.9, 4.9, 4.91, 4.91, 4.91, 4.91, 4.92, 4.92, 4.92
», 4.92, 4.93, 4.93, 4.93, 4.93, 4.94, 4.94, 4.94, 4.94, 4.95, 4.95, 4.95, 4.96, 4.96, 4.96
», 4.96, 4.97, 4.97, 4.97, 4.97, 4.98, 4.98, 4.98, 4.98, 4.99, 4.99, 4.99, 4.99, 5, 5, 5, 5.01, 5
».01, 5.01, 5.01, 5.02, 5.02, 5.02, _
          5.02, 5.03, 5.03, 5.03, 5.03, 5.04, 5.04, 5.04, 5.04, 5.05, 5.05, 5.05, 5.05, 5.06, 5.06, 5.06, 5.
»06, 5.07, 5.07, 5.07, 5.07, 5.08, 5.08, 5.08, 5.08, 5.09, 5.09, 5.09, 5.09, 5.1, 5.1, 5.1, 5.1,
»5.11, 5.11, 5.11, 5.11, 5.12, 5.12, 5.12, 5.12, 5.13, 5.13, 5.13, 5.13, 5.14, 5.14, 5.14, 5.14,
»5.15, 5.15, 5.15, 5.15, 5.16, 5.16, 5.16, 5.16, 5.17, 5.17, 5.17, 5.17, 5.18, 5.18, 5.18, 5.18,
»5.19, 5.19, 5.19, 5.19, 5.2, 5.2, 5.2, 5.2, 5.21, 5.21, 5.21, 5.21, 5.22, 5.22, 5.22, 5.22, 5.23
», 5.23, 5.23, 5.24, 5.24, 5.24, 5.24, 5.25, 5.25, 5.25, 5.25, 5.26, 5.26, 5.26, 5.26, 5.27, 5.27
», 5.27, 5.27, 5.28, 5.28, 5.28, 5.28, 5.29, 5.29, 5.29, 5.29, 5.3, 5.3, 5.3, 5.31, 5.31, 5.31, 5
».31, 5.32, 5.32, 5.32, 5.32, 5.33, 5.33, 5.33, 5.33, 5.34, 5.34, 5.34, 5.35, 5.35, 5.35, 5.35, 5
».36, 5.36, 5.36, 5.36, 5.37, 5.37, 5.37, 5.37, 5.38, 5.38, 5.38, 5.39, 5.39, 5.39, 5.39, 5.4, 5.
»4, 5.4, 5.4, 5.41, 5.41, 5.41, 5.42, 5.42, 5.42, 5.42, 5.43, 5.43, 5.43, 5.43, 5.44, 5.44, 5.44,
» 5.45, 5.45, 5.45, 5.45, 5.46, 5.46, 5.46, _
          5.46, 5.47, 5.47, 5.47, 5.48, 5.48, 5.48, 5.48, 5.49, 5.49, 5.49, 5.5, 5.5, 5.5, 5.5, 5.51, 5.51,
»5.51, 5.52, 5.52, 5.52, 5.52, 5.53, 5.53, 5.53, 5.54, 5.54, 5.54, 5.54, 5.55, 5.55, 5.55, 5.56,
»5.56, 5.56, 5.57, 5.57, 5.57, 5.57, 5.58, 5.58, 5.58, 5.59, 5.59, 5.59, 5.59, 5.6, 5.6, 5.6, 5.
»61, 5.61, 5.61, 5.62, 5.62, 5.62, 5.62, 5.63, 5.63, 5.63, 5.64, 5.64, 5.64, 5.65, 5.65, 5.65, 5.
```

```
»66, 5.66, 5.66, 5.67, 5.67, 5.67, 5.67, 5.68, 5.68, 5.68, 5.69, 5.69, 5.69, 5.7, 5.7, 5.7, 5.71,
» 5.71, 5.71, 5.72, 5.72, 5.72, 5.73, 5.73, 5.73, 5.74, 5.74, 5.74, 5.75, 5.75, 5.75, 5.76, 5.76,
» 5.76, 5.77, 5.77, 5.77, 5.78, 5.78, 5.78, 5.79, 5.79, 5.79, 5.8, 5.8, 5.8, 5.81, 5.81, 5.81, 5.
»82, 5.82, 5.82, 5.83, 5.83, 5.83, 5.84, 5.84, 5.85, 5.85, 5.85, 5.86, 5.86, 5.86, 5.87, 5.87, 5.
»87, 5.88, 5.88, 5.89, 5.89, 5.89, 5.9, 5.9, 5.9, 5.91, 5.91, 5.92, 5.92, 5.92, 5.93, 5.93, 5.93,
» 5.94, 5.94, 5.95, 5.95, 5.95, 5.96, 5.96, 5.97, 5.97, 5.97, 5.98, 5.98, 5.99, 5.99, 5.99, 6, 6,
» 6.01, 6.01, 6.02, 6.02, 6.02, 6.03, _
         6.03, 6.04, 6.04, 6.05, 6.05, 6.05, 6.06, 6.06, 6.07, 6.07, 6.08, 6.08, 6.08, 6.09, 6.09, 6.1, 6.1
», 6.11, 6.11, 6.12, 6.12, 6.13, 6.13, 6.14, 6.14, 6.15, 6.15, 6.16, 6.16, 6.17, 6.17, 6.17, 6.18
», 6.19, 6.19, 6.2, 6.2, 6.21, 6.21, 6.22, 6.22, 6.23, 6.23, 6.24, 6.24, 6.25, 6.25, 6.26, 6.26,
»6.27, 6.28, 6.28, 6.29, 6.29, 6.3, 6.3, 6.31, 6.32, 6.32, 6.33, 6.33, 6.34, 6.35, 6.35, 6.36, 6.
»37, 6.37, 6.38, 6.39, 6.39, 6.4, 6.41, 6.41, 6.42, 6.43, 6.43, 6.44, 6.45, 6.45, 6.46, 6.47, 6.
»48, 6.48, 6.49, 6.5, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.55, 6.56, 6.57, 6.58, 6.59, 6.6, 6.61
», 6.62, 6.63, 6.64, 6.64, 6.65, 6.66, 6.67, 6.68, 6.7, 6.71, 6.72, 6.73, 6.74, 6.75, 6.76, 6.77,
» 6.79, 6.8, 6.81, 6.83, 6.84, 6.85, 6.87, 6.88, 6.9, 6.91, 6.93, 6.94, 6.96, 6.98, 7, 7.01, 7.03
», 7.05, 7.07, 7.1, 7.12, 7.14, 7.17, 7.2, 7.23, 7.26, 7.29, 7.33, 7.37, 7.41, 7.46, 7.51, 7.58,
»7.65, 7.75, 7.88, 8.09)

   '========Return=========
   Return_Probit = arrProbit(CInt(ipos * 10))
   '========Clean up=======
   Erase arrProbit
End Function
```

```
'#########################################################################################################
'#                                                                                                       #
'# The function performs a permutation test on two sets of data to prove that they are the sa #
'# me. It first runs through all the selected patients then inverts the selection and runs    #
'# through the send group                                                                                #
'# Calculate_T0 Function: Returns the initial difference between the means of the two groups  #
'#########################################################################################################
Option Explicit
Dim FLX As MSFlexGrid
Dim FRM As Form
Dim T0 As Double 'T0 difference in means of groups at start
Dim GrpA_Mean As Double
Dim GrpB_Mean As Double


Private Sub init_vars()
   Set FLX = FRMHost.FLXMain
   Set FRM = FRMHost
   FRM.CHKPowerPeriod.Value = 1

End Sub

Sub Permutation_test()
   Dim i As Integer
   ReDim arrTs(999) As String
   Dim p As Double, msg As String
   Call init_vars
   ReDim arrOFALL(FRM.FLXMain2Grouped.Rows - 3, 1) As Double
   '-calculate diff at turn zero-
   arrOFALL = Calculate_T0()
   Debug.Print T0

   '-Build array of T values based on 1000 different grp assignments-
   For i = 0 To UBound(arrTs)
      arrOFALL = Randomise_GRPS(arrOFALL)
      arrTs(i) = Calculate_T(arrOFALL)
   Next

   '-Sort arrof T's--
   arrTs = Bubble_Sort(arrTs)

   '--Get P Value--
   p = get_P(arrTs())

   p = Format((1 - CDbl(p)), "0.00")
   If p > 0.05 Then
      If p <> 1 Then
         msg = vbCrLf & "Null Hypothesis Rejected. These groups are  probably the same"
      Else
         msg = "(tending towards)" & vbCrLf & "Null Hypothesis Rejected. These groups are  probably the
»same"
      End If
   Else
      msg = vbCrLf & "Null Hypothesis Accepted. These groups are probably different"
   End If
   Call MsgBox("Probability of these two groups being the same is " & p & " " & msg)
End Sub
Function get_P(arrTs() As String) As Double
   Dim i As Integer
   Dim dTMP As Double

   For i = 0 To UBound(arrTs)
      dTMP = FormatSF(CDbl(arrTs(i)), 4)
      If FormatSF(T0, 4) <= dTMP Then
         Exit For
      End If

   Next i

   get_P = (((i + 1) / (UBound(arrTs) + 1)))
   If get_P > 1 Then get_P = 1
End Function
Function Randomise_GRPS(arrOFALL() As Double) As Double()
   Dim Half As Integer
   Dim idata As Integer
   Dim myRand As Integer
```

```
Dim GRP0 As Integer

'-define half the array-
Half = CInt((UBound(arrOFALL) + 1) / 2)
'-define GRP 0 size-
For idata = 0 To UBound(arrOFALL)
   If arrOFALL(idata, 1) = 0 Then
      GRP0 = GRP0 + 1
   End If

   '-clear all grpings-
   arrOFALL(idata, 1) = -777
Next idata

If GRP0 > Half Then
   GRP0 = UBound(arrOFALL) + 1 - GRP0
End If

'==============GET RANDOM NUMBERS=================
'-init vars-
Const max As Long = 2147483647
Dim part As Variant, moi As Variant
Dim Value As Long
Dim i As Integer, imoi As Integer
Dim start As Long, ending As Long
Dim tmpVar As Variant
'-Define PArtition of Maximum number that each value will cover-
part = max / (UBound(arrOFALL) + 1)


For idata = 0 To GRP0
   '-Get random numbers-
   moi = MDLRANDOM.ISAAC_Prng(1, False)
   '-set random number-
   Value = Abs(moi(0))

   '-Equate Random value to the part of the array that each number covers-
   For i = 1 To UBound(arrOFALL) + 1
      '-define area to use-
      start = ((i - 1) * part)
      tmpVar = (i * part)
      If tmpVar <= max Then
         ending = i * part
      Else
         ending = max
      End If

      '-check if val fit-
      If Value >= start And Value <= ending Then
         '    Debug.Print i
         Exit For
      End If
   Next

   '-Set Random chosen to be in GRP0-
   If arrOFALL(i - 1, 1) <> 0 Then
      arrOFALL(i - 1, 1) = 0
   Else
      idata = idata - 1
   End If
Next idata

'-assign rest to grp 1-
For idata = 0 To UBound(arrOFALL)
   If arrOFALL(idata, 1) = -777 Then
      arrOFALL(idata, 1) = 1
   End If

Next idata

'-output randoms-
Dim strout As String
Open "C:\nat.txt" For Append As #1
For idata = 0 To UBound(arrOFALL)
   If arrOFALL(idata, 1) = 0 Then
      strout = strout & "|" & idata
   End If
Next
```

```
   Write #1, strout
   Close #1

   Randomise_GRPS = arrOFALL

End Function

Function Calculate_T(arrOFALL() As Double) As Double
   Dim idata As Integer
   Dim SumA As Double, Sumb As Double
   Dim countA As Integer, CountB As Integer
   Dim MeanA As Double, MeanB As Double

   For idata = 0 To UBound(arrOFALL)
      If arrOFALL(idata, 1) = 0 Then
         SumA = SumA + arrOFALL(idata, 0)
         countA = countA + 1
      Else
         Sumb = Sumb + arrOFALL(idata, 0)
         CountB = CountB + 1
      End If

   Next

   MeanA = SumA / countA
   MeanB = Sumb / CountB
   Calculate_T = Abs(MeanA - MeanB)

End Function
Function Calculate_T0() As Double()
   '-Init Vars-
   Dim irow As Integer
   Dim idata As Integer
   Dim arrperiod() As Double
   Dim dHigh As Double
   Dim sum As Double
   Dim isel As Integer
   Dim lowest As Integer

   ReDim arrBothGRP(FRM.FLXMain2Grouped.Rows - 3, 1) As Double
   Dim arrBothGRP_COUNT As Integer
   '---SET GLOBAL VAR TO SHOW WE ARE RUNNING---
   GRP_POWER_ON = True

   '-Find shortest dataset-
   lowest = FLX.TextMatrix(1, 2)
   For isel = 1 To FLX.Rows - 2
      If FLX.TextMatrix(isel, 2) <= lowest Then
         lowest = FLX.TextMatrix(isel, 2)
      End If
   Next isel

   '-Cycle through the selected group first, then invert the selected pats and run again-
   For isel = 0 To 1
      '-Define Array to store all-
      ReDim Arr_of_All(FLX.Rows - 3, 1) As Double

      '-Cycle thru data-
      For irow = 0 To UBound(Arr_of_All)
         '-select subject-
         FLX.Row = irow + 1
         Call FRM.FLXMain_SelChange

         '--Make Power/Period CHT Normalised--
         If LCase(FRMHost.cmdFourier_Abs_Norm.Caption) = "normalise" Then
            FRMHost.cmdFourier_Abs_Norm.Caption = "Absolute"
         End If

         '-Get Data-
         arrperiod = Get_Period_Data()

         '-Find Highest Val-
         dHigh = 0


         For idata = 0 To UBound(arrperiod)
            If arrperiod(idata, 1) > dHigh Then
               '-slight bodge fix here to only compare powers in same range-
```

```
            If arrperiod(idata, 0) <= lowest Then
               dHigh = arrperiod(idata, 1)
            End If
         End If
      Next


      '-Add to array-
      Arr_of_All(irow, 0) = dHigh
      Arr_of_All(irow, 1) = isel
   Next irow

   '-Calc Mean-
   sum = 0
   For idata = 0 To UBound(Arr_of_All)
      sum = sum + Arr_of_All(idata, 0)
   Next idata

   If isel = 0 Then
      GrpA_Mean = sum / (UBound(Arr_of_All) + 1)

      '-Select Inverse Group-
      Call Swap_Selected_Pats(False)
      Call Select_inverse
      Call Swap_Selected_Pats(True)
   Else
      GrpB_Mean = sum / (UBound(Arr_of_All) + 1)
   End If

   '-Save Array of highest powers for later randomisation-
   For idata = 0 To UBound(Arr_of_All)

      arrBothGRP(arrBothGRP_COUNT, 0) = Arr_of_All(idata, 0)
      arrBothGRP(arrBothGRP_COUNT, 1) = Arr_of_All(idata, 1)

      arrBothGRP_COUNT = arrBothGRP_COUNT + 1
   Next idata
Next isel

'-set return var-
T0 = Abs(GrpA_Mean - GrpB_Mean)
Calculate_T0 = arrBothGRP
'---SET GLOBAL VAR TO SHOW WE ARE RUNNING---
GRP_POWER_ON = False
End Function

Private Sub Select_inverse()
   Dim irow As Integer

   For irow = 1 To FLX.Rows - 1
      FLX.Row = irow
      Call FRM.FLXMain_DblClick

   Next irow
End Sub

Function Get_Period_Data() As Double()
   Dim FLX_P As MSFlexGrid
   Dim i As Integer
   Set FLX_P = FRMHost.Controls("FLXPOWER")
   ReDim arrperiod(FLX_P.Rows - 2, 1) As Double

   For i = 1 To FLX_P.Rows - 1
      arrperiod(i - 1, 0) = FLX_P.TextMatrix(i, 0)
      arrperiod(i - 1, 1) = FLX_P.TextMatrix(i, 1)
   Next i

   Get_Period_Data = arrperiod
End Function
```

```
Option Explicit
' This module is very complex it performs  fourier transforms for all grouped data _
and gets the power /period graphs for each which it adds together. _
1) Gets data from Power/Period Flexigrid converts period to integer _
2) Interpolates Data between Periods _
3) Then stores the data and gets the next patient _
4) When all Patients data is done it merges it and averages it _
5) Then it draws the SEM (SEM_Group, MergeSEM_Group) _
6) Then it outputs the FLX and CHT
Dim FLX As MSFlexGrid
Dim FRM As Form


Private Sub init_vars()
   Set FLX = FRMHost.FLXMain
   Set FRM = FRMHost
End Sub



Sub Draw_PP()
   '-Init Vars-
   Call init_vars
   Dim Least As Integer
   Dim arrperiod() As Double, arrPeriod_lg() As Variant
   Dim idata As Integer, iPat As Integer
   Dim irow As Integer
   Least = (Get_Least() * Time_Val)



   '---SET GLOBAL VAR TO SHOW WE ARE RUNNING---
   GRP_POWER_ON = True

   '-Define Array to store all-
   ReDim Arr_of_All(Least, 2, FLX.Rows - 3) As Variant
   '-Cycle thru data-
   For irow = 1 To FLX.Rows - 2
      '-select subject-
      FLX.Row = irow
      Call FRM.FLXMain_SelChange
      '-select grp cht button here---

      '-Get Data-
      arrperiod = Get_Period_Data

      '-Interpolate array-
      arrPeriod_lg = Expand_Interpolate_data(arrperiod)

      '-Resize least if too big-
      If Least - 1 >= UBound(arrPeriod_lg) Then
         Least = UBound(arrPeriod_lg)
      End If
      '-Add to array-
      For idata = 0 To Least

         Arr_of_All(idata, 0, irow - 1) = arrPeriod_lg(idata, 0)
         Arr_of_All(idata, 1, irow - 1) = arrPeriod_lg(idata, 1)
         Arr_of_All(idata, 2, irow - 1) = arrPeriod_lg(idata, 2)
      Next idata

   Next irow

   '====Normalise Array if option selected=========
   If LCase(FRMHost.CHKABS_Norm_Fourier.Caption) = "make absolute" Then
      Dim sTotal As Double
      Dim mTotal As Double


      For iPat = 0 To UBound(Arr_of_All, 3)

         sTotal = 0
         '-Sum total-
         For idata = 3 To UBound(Arr_of_All)
            If Arr_of_All(idata, 2, iPat) = "" Then Exit For
            sTotal = sTotal + Arr_of_All(idata, 1, iPat)
```

```
      Next idata

      '-Calculate % of power-
      For idata = 3 To UBound(Arr_of_All)
         If Arr_of_All(idata, 2, iPat) = "" Then Exit For
         Arr_of_All(idata, 1, iPat) = (Arr_of_All(idata, 1, iPat) / sTotal) * 100
      Next idata
   Next iPat

End If

Open "C:\nat.txt" For Output As #1
Write #1, "Period=15"
For iPat = 0 To UBound(Arr_of_All, 3)
   Write #1, Arr_of_All(15, 1, iPat)
Next iPat
Close #1

'---------Merge and Average Data--------
ReDim arrPeriod_Ave(Least, 2) As Variant
'###############Merge################
For iPat = 0 To FLX.Rows - 3
   For idata = 0 To UBound(arrPeriod_Ave)
      arrPeriod_Ave(idata, 0) = Arr_of_All(idata, 0, iPat)
      arrPeriod_Ave(idata, 1) = arrPeriod_Ave(idata, 1) + Arr_of_All(idata, 1, iPat)

      If arrPeriod_Ave(idata, 2) <> "Real" Then
         arrPeriod_Ave(idata, 2) = Arr_of_All(idata, 2, iPat)
      End If
   Next idata
Next iPat



'#################Average################
For idata = 0 To UBound(arrPeriod_Ave)
   arrPeriod_Ave(idata, 1) = arrPeriod_Ave(idata, 1) / (FLX.Rows - 2)
Next idata



'=========Work out SEM and add it on=======
ReDim arrPower_SEM(UBound(arrPeriod_Ave)) As Variant
ReDim arrPower_ALL(UBound(arrPeriod_Ave), 1) As Variant

arrPower_SEM = SEM_Group(Arr_of_All, arrPeriod_Ave)
arrPower_ALL = MergeSEM_Group(arrPower_SEM, arrPeriod_Ave)


'======Create on Array just for FLX=======
ReDim arrPower_FLX(UBound(arrPeriod_Ave), 2) As Variant

'#Add Ave#
For idata = 0 To UBound(arrPower_FLX)
   arrPower_FLX(idata, 0) = arrPeriod_Ave(idata, 0)
   arrPower_FLX(idata, 1) = arrPeriod_Ave(idata, 1)
   arrPower_FLX(idata, 2) = arrPower_SEM(idata)
Next idata


Call Fill_Flx(arrPower_FLX, "Grouped_Power")


'==============Position Controls===============
Call Format_Controls("Grouped_Power", 16, "Period", "Power", "SEM")

'==================Draw CHT=================
FRMHost.Controls("ChtGrouped_Power").ChartData = arrPower_ALL

'=======Show Area Under the Curve=========
Call Select_Tab("Grouped_Power")
Call AreaUnderCurve
'---Hide Progress Bar---

frmWait.Hide

'==============
Set FLX = Nothing
```

```
   Set FRM = Nothing
   Erase Arr_of_All


   '---SET GLOBAL VAR TO SHOW WE ARE NOT RUNNING---
   GRP_POWER_ON = False

End Sub

Function Get_Period_Data() As Double()
   Dim FLX_P As MSFlexGrid
   Dim i As Integer
   Set FLX_P = FRMHost.Controls("FLXPOWER")
   ReDim arrperiod(FLX_P.Rows - 2, 1) As Double

   For i = 1 To FLX_P.Rows - 1
      arrperiod(i - 1, 0) = FLX_P.TextMatrix(i, 0)
      arrperiod(i - 1, 1) = FLX_P.TextMatrix(i, 1)
   Next i

   Get_Period_Data = arrperiod
End Function

Function Expand_Interpolate_data(arrperiod() As Double) As Variant
   '--Init Vars--
   Dim iReal As Integer, ismall As Integer, i As Integer, iLeft As Integer, iRight As Integer
   Dim begining As Double, ending As Double, data As Double, iint As Integer
   ReDim arrPeriod_Large(arrperiod(UBound(arrperiod), 0), 2) As Variant
   Dim Startpoint As Integer, Endpoint As Integer, Gap As Integer
   '----------------------Insert the data that we do have----------------------
   '------Insert the Period Numbering----
   For i = 0 To UBound(arrPeriod_Large)
      arrPeriod_Large(i, 0) = i
   Next i


   '----Insert the data that we do have-----
   ismall = 0
   For iReal = 0 To UBound(arrPeriod_Large)
      If arrPeriod_Large(iReal, 0) = CInt(arrperiod(ismall, 0)) Then    'Time

         arrPeriod_Large(iReal, 1) = arrperiod(ismall, 1)        'Data
         arrPeriod_Large(iReal, 2) = "Real"                      'Label
         ismall = ismall + 1
         '#Reset counter as we have multiple valus for the same period#
         If iReal <> UBound(arrPeriod_Large) Then
            iReal = 0
         End If
      End If

   Next iReal


   '================Interpolate The Data we don't============
   '----Find start of real data------
   Startpoint = CLng(arrperiod(0, 0))
   '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   '-insert Tag 'No Data' for the bit we can't interpolate-
   For i = 0 To Startpoint - 1
      arrPeriod_Large(i, 2) = "No Data"
   Next i

   '--------Interpolate the data we don't have-------------
   For i = Startpoint To UBound(arrPeriod_Large)
      If arrPeriod_Large(i, 2) = "" Then

         '####################Find Real Data to Start From##################
         For iLeft = arrPeriod_Large(i, 0) To Startpoint Step -1
            If arrPeriod_Large(iLeft, 2) <> "" Then
               begining = arrPeriod_Large(iLeft, 0)
               Exit For
            End If
         Next iLeft
         '#########################################################################
         '######################Find Real Data to End At#####################
         For iRight = arrPeriod_Large(i, 0) To UBound(arrPeriod_Large)
            If arrPeriod_Large(iRight, 2) <> "" Then
               ending = arrPeriod_Large(iRight, 0)
```

```
                Exit For
             End If
          Next iRight
          '#########################################################################
          '###################Interpolate all Data Between#########################
          Gap = ending - begining
          data = (arrPeriod_Large(iRight, 1) - arrPeriod_Large(iLeft, 1)) / Gap
          For iint = 1 To Gap - 1
             arrPeriod_Large(i + (iint - 1), 1) = arrPeriod_Large(i - 1, 1) + (data * iint)
             arrPeriod_Large(i + (iint - 1), 2) = "Interpolated"
          Next iint
          '#########################################################################
       End If
    Next i
    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Expand_Interpolate_data = arrPeriod_Large()

End Function

'Sub Draw_Power_Grouped()
''==================Init Vars====================
'Dim FLX As MSFlexGrid, FRM As Form
'Dim iReal As Integer, iSmall As Integer, i As Integer, Startpoint As Integer, _
'ileft As Integer, iRight As Integer, Begining As Integer, Ending As Integer, _
'Gap As Integer, iInt As Integer, iData As Long, lastreading As Integer
'Dim data As Double, iPat As Integer, FLX_P As MSFlexGrid, Pat_Total As Integer
'
''---SET GLOBAL VAR TO SHOW WE ARE RUNNING---
'GRP_POWER_ON = True
'
''FRMHost.Tabmain.Tabs(3).Selected = True
'Set FRM = FRMHost
'Set FLX = FRMHost.FLXMain
'Set FLX_P = FRMHost.Controls("FLXPOWER")
'
'Static Once_Only As Boolean
'Static Finished As Boolean
'Static ExitCounter As Integer
'
'If Once_Only = False Then
'    FLX.RowSel = 1
'    FLX.Row = 1
'    Call FRM.FLXMain_SelChange
'    Once_Only = True
'
'    '----Hide FLX and CHT----
'    FRMHost.Controls("ChtGrouped_Power").Visible = False
'    FLX_P.Visible = False
'    '---Show Progress Bar---
'    frmWait.Show , FRMHost
'    frmWait.Bar.Value = 0
'    frmWait.Bar.Max = FLX.Rows - 1
'End If
'
'
'
'
''=========Take/Store Data From FLXPower=========
'ReDim arrPeriod(FLX_P.Rows - 2, 1) As Double
'For i = 0 To UBound(arrPeriod)
'
'        FLX_P.Row = i + 1
'        FLX_P.Col = 0
'        arrPeriod(i, 0) = FLX_P.Text
'        FLX_P.Row = i + 1
'        FLX_P.Col = 1
'        arrPeriod(i, 1) = FLX_P.Text
'
'Next i
'
'
''================Prepare the Dataset==============
''--convert Periodicity from Double to Int--
''---------NB We lose integrity here--------
'
'For i = 0 To UBound(arrPeriod)
'   arrPeriod(i, 0) = CDbl(CLng(arrPeriod(i, 0))) '* timeslot)
```

```
'   arrPeriod(i, 1) = arrPeriod(i, 1)
'Next i
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
''-----Create new larger array to hold all new interpolated data---
''##Get the last Time at which a data reading is taken##
'lastreading = arrPeriod(UBound(arrPeriod), 0)
''LastReading = ((LastReading / timeslot) - 1)
'
''#########################################################
'ReDim arrPeriod_Large(lastreading, 2) As Variant
''ReDim arrPeriod_Large(UBound(arrPeriod), 2) As Variant
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
''------Insert the Period Numbering----
'For i = 0 To UBound(arrPeriod_Large)
'   arrPeriod_Large(i, 0) = i '* timeslot
'Next i
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
''---------------------Insert the data that we do have----------------------
'iSmall = 0
'For iReal = 0 To UBound(arrPeriod_Large)
'    If arrPeriod_Large(iReal, 0) = arrPeriod(iSmall, 0) Then    'Time
'
'        arrPeriod_Large(iReal, 1) = arrPeriod(iSmall, 1)        'Data
'        arrPeriod_Large(iReal, 2) = "Real"                      'Label
'        iSmall = iSmall + 1
'        '#Reset counter as we have multiple valus for the same period#
'        If iReal <> UBound(arrPeriod_Large) Then
'            iReal = 0
'        End If
'    End If
'
'Next iReal
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
''================Interpolate The Data============
'
'    '----Find start of real data------
'    Startpoint = CLng(arrPeriod(0, 0))
'    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'    '-insert Tag 'No Data' for the bit we can't interpolate-
'    For i = 0 To Startpoint - 1
'        arrPeriod_Large(i, 2) = "No Data"
'    Next i
'    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'    '--------Interpolate the data we don't have-------------
'    For i = Startpoint To UBound(arrPeriod_Large)
'        If arrPeriod_Large(i, 2) = "" Then
'
'            '###################Find Real Data to Start From##################
'            For ileft = arrPeriod_Large(i, 0) To Startpoint Step -1
'                If arrPeriod_Large(ileft, 2) <> "" Then
'                    Begining = arrPeriod_Large(ileft, 0)
'                    Exit For
'                End If
'            Next ileft
'            '#########################################################################
'            '######################Find Real Data to End At#####################
'            For iRight = arrPeriod_Large(i, 0) To UBound(arrPeriod_Large)
'                If arrPeriod_Large(iRight, 2) <> "" Then
'                    Ending = arrPeriod_Large(iRight, 0)
'                    Exit For
'                End If
'            Next iRight
'            '#########################################################################
'            '###################Interpolate all Data Between#####################
'            Gap = Ending - Begining
'            data = (arrPeriod_Large(iRight, 1) - arrPeriod_Large(ileft, 1)) / Gap
'            For iInt = 1 To Gap - 1
'                arrPeriod_Large(i + (iInt - 1), 1) = arrPeriod_Large(i - 1, 1) + (data * iInt)
'                arrPeriod_Large(i + (iInt - 1), 2) = "Interpolated"
'            Next iInt
'            '#########################################################################
'        End If
'    Next i
'    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
''========Store New Data in Huge Split Array======
```

```
''----------Create Split Array----------
'
'Static SDArray(((Max_Data_Items + 1) * 5) - 1, 2, Max_Patients) As Variant
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
'
''------Store current patients data-----
'For iData = 0 To UBound(arrPeriod_Large)
'    DoEvents
'    SDArray(iData, 0, FLX.RowSel - 1) = arrPeriod_Large(iData, 0) '* Time_Val
'    SDArray(iData, 1, FLX.RowSel - 1) = arrPeriod_Large(iData, 1)
'    SDArray(iData, 2, FLX.RowSel - 1) = arrPeriod_Large(iData, 2)
'Next iData
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
''---Update Waiting Form----
'frmWait.Bar.Value = frmWait.Bar.Value + 1
'
''================Call Next Patient===============
'If FLX.RowSel < FLX.Rows - 2 Then
'    FLX.RowSel = FLX.RowSel + 1
'    If Finished = False Then
'        FRM.FLXMain.Row = FRM.FLXMain.Row + 1
'        Call FRM.FLXMain_SelChange
'        Erase arrPeriod
'        Debug.Print FRM.FLXMain.Row
'
'        '-escape clause-
'        'If FRM.FLXMain.Row <> FLX.Rows - 2 Then
'        Call Draw_Power_Grouped
'       ' Else
'
'
'
'        '----Hide FLX and CHT----
'        FRMHost.Controls("ChtGrouped_Power").Visible = False
'        FLX_P.Visible = False
'
'    End If
'    '-----V IMP Get out Clause--
'    If Finished = True Then
'        ExitCounter = ExitCounter + 1
'        '#####On Last Run reset Finished Flag####
'        If ExitCounter = FLX.Rows - 3 Then
'            Finished = False
'            ExitCounter = 0
'        End If
'        '##########################################
'        Exit Sub
'    End If
'    '--------------------------
'End If
''================================================
'
''=========Store All Data in Merged Array=========
''---find lowest number of datapoints----
'Dim Lowest As Integer
'For iData = 0 To UBound(SDArray)
'    For iPat = 0 To FLX.Rows - 3 'Minus 3 as 1 is as tile, 1 is blank and it begins @ 0
'        If SDArray(iData, 2, iPat) = "" Then
'            '----First Pass---
'            If Lowest = 0 Then
'                Lowest = iData
'            End If
'            '--Consecutive passes--
'            If iData < Lowest Then
'                Lowest = iData
'            End If
'        End If
'    Next iPat
'Next iData
'Lowest = Lowest - 1 'as Base is 0
''---------Create Arevaged Data Array----
'ReDim arrPeriod_Ave(Lowest, 2) As Variant
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
''================Normalise SD Array data=========================
'If FRMHost.CHKABS_Norm_Fourier.Value = 1 Then
```

```
'    Dim sTotal As Double
'    Dim mTotal As Double
'
'
'    For iPat = 0 To UBound(SDArray, 3)
'
'        sTotal = 0
'        '-Sum total-
'        For iData = 3 To UBound(SDArray)
'           If SDArray(iData, 2, iPat) = "" Then Exit For
'           sTotal = sTotal + SDArray(iData, 1, iPat)
'
'        Next iData
'
'        '-Calculate % of power-
'        For iData = 3 To UBound(SDArray)
'           If SDArray(iData, 2, iPat) = "" Then Exit For
'           SDArray(iData, 1, iPat) = (SDArray(iData, 1, iPat) / sTotal) * 100
'        Next iData
'    Next iPat
'
'End If
''---------Merge and Average Data--------
''###############Merge################
'For iPat = 0 To FLX.Rows - 3
'    For iData = 0 To Lowest
'        arrPeriod_Ave(iData, 0) = SDArray(iData, 0, iPat)
'        arrPeriod_Ave(iData, 1) = arrPeriod_Ave(iData, 1) + SDArray(iData, 1, iPat)
'        If arrPeriod_Ave(iData, 2) <> "Real" Then
'            arrPeriod_Ave(iData, 2) = SDArray(iData, 2, iPat)
'        End If
'    Next iData
'Next iPat
''#################################
'
'
''#################Average################
'For iData = 0 To UBound(arrPeriod_Ave)
'     arrPeriod_Ave(iData, 1) = arrPeriod_Ave(iData, 1) / (FLX.Rows - 2)
'Next iData
''########################################
'
'
''=========Work out SEM and add it on=======
'ReDim arrPower_SEM(UBound(arrPeriod_Ave)) As Variant
'ReDim arrPower_ALL(UBound(arrPeriod_Ave), 1) As Variant
'
'arrPower_SEM = SEM_Group(SDArray, arrPeriod_Ave)
'arrPower_ALL = MergeSEM_Group(arrPower_SEM, arrPeriod_Ave)
'
'
''======Create on Array just for FLX=======
'ReDim arrPower_FLX(UBound(arrPeriod_Ave), 2) As Variant
'
''#Add Ave#
'For iData = 0 To UBound(arrPower_FLX)
'    arrPower_FLX(iData, 0) = arrPeriod_Ave(iData, 0)
'    arrPower_FLX(iData, 1) = arrPeriod_Ave(iData, 1)
'    arrPower_FLX(iData, 2) = arrPower_SEM(iData)
'Next iData
'
'
'Call Fill_Flx(arrPower_FLX, "Grouped_Power")
'
'
''==============Position Controls===============
'Call Format_Controls("Grouped_Power", 16, "Period", "Power", "SEM")
'
''==================Draw CHT=================
'FRMHost.Controls("ChtGrouped_Power").ChartData = arrPower_ALL
'
''=======Show Area Under the Curve=========
'Call Select_Tab("Grouped_Power")
'Call AreaUnderCurve
'
'
''----Show FLX and CHT----
'FRMHost.Controls("ChtGrouped_Power").Visible = True
```

```
'FLX_P.Visible = True
''---Hide Progress Bar---
'
'frmWait.Hide
''=====================Tidy Up====================
''=======Interpolate works by being called by the period function and then _
'dealing with that periods data and calling for the next one unfortunatly it _
'gets stuck in a loop as it calls period which in turn call itself which call _
'period and so on this bit of code is it's get out clause==================
'Finished = True
''==============
'Set FLX = Nothing
'Set FRM = Nothing
'Erase SDArray
'Once_Only = False
'
''---SET GLOBAL VAR TO SHOW WE ARE NOT RUNNING---
'GRP_POWER_ON = False
'End Sub
'
'
'Sub Convert_Absolute_to_Normalised()
''=============Init Vars==============
'Dim FLX As MSFlexGrid, icol As Integer, irow As Integer, Sum_Total As Double
'Dim Count_of_RealData As Integer, N_Count As Integer
'Set FLX = FRMHost.Controls("FLXGrouped_Power")
'Dim Sum_Total1 As Double
''=====Find How much Real Data we Have=====
'For irow = 1 To FLX.Rows - 1
'    If FLX.TextMatrix(irow, 2) <> "" Then
'        Count_of_RealData = Count_of_RealData + 1
'    End If
'Next irow
'
'ReDim arrData(Count_of_RealData - 1, 2) As Variant
''==============Get Real Data only==============
'For icol = 1 To FLX.Cols - 1
'    N_Count = 0
'
'    For irow = 1 To FLX.Rows - 2
'        If FLX.TextMatrix(irow, 2) <> "" Then
'            '-------Copy Data-------
'            arrData(N_Count, 0) = FLX.TextMatrix(irow, 0)
'            arrData(N_Count, 1) = FLX.TextMatrix(irow, 1)
'            arrData(N_Count, 2) = FLX.TextMatrix(irow, 2)
'            '-----Total up Data-----
'            If icol = 1 Then
'                Sum_Total = Sum_Total + arrData(N_Count, 1)
'                Sum_Total1 = Sum_Total1 + arrData(N_Count, 2)
'            End If
'            '----------------------
'
'            N_Count = N_Count + 1
'        End If
'    Next irow
'Next icol
''==========Normalise Data============
'For irow = 0 To UBound(arrData)
'    arrData(irow, 1) = ((arrData(irow, 1) / Sum_Total) * 100)
'    arrData(irow, 2) = ((arrData(irow, 2) / Sum_Total1) * 100)
'Next irow
'
'
''===Create CHT array===
'ReDim arrCHT(((UBound(arrData) + 1) * 5) - 1, 1) As Variant
'Dim iData  As Integer
'For iData = 0 To UBound(arrData) Step 5
'
'    '#Time#
'    arrCHT(iData, 0) = arrData(iData, 0)
'    arrCHT(iData + 1, 0) = arrData(iData, 0)
'    arrCHT(iData + 2, 0) = arrData(iData, 0)
'    arrCHT(iData + 3, 0) = arrData(iData, 0)
'    arrCHT(iData + 4, 0) = arrData(iData, 0)
'
'    '#Data#
'    arrCHT(iData, 1) = arrData(iData, 1)
'    arrCHT(iData + 1, 1) = arrData(iData, 1) + arrData(iData, 2)
```

```
'    arrCHT(iData + 2, 1) = arrData(iData, 1)
'    arrCHT(iData + 3, 1) = arrData(iData, 1) - arrData(iData, 2)
'    arrCHT(iData + 4, 1) = arrData(iData, 1)
'
'
'Next iData
'
''===============Fill FLX=============
'Call Fill_Flx(arrData, "Grouped_Power")
'
'
''==============Position Controls===============
'Call Format_Controls("Grouped_Power", 16, "Period", "% Power", "SEM")
'
''==================Draw CHT=================
'FRMHost.Controls("ChtGrouped_Power").ChartData = arrCHT


'End Sub
```

```
Option Explicit
'This module produces a power period graph using the fourier data _
1) Gets Data _
2) Changes Frquency to Periodicity _
3) Reverses array (why?) _
4) Draws it _
 _
 I need to check it frequency to periodicity is correct

Public Sub Draw_Power()
   '=====================Init Vars======================
   Dim Data_items As Integer, FLX As MSFlexGrid, idata As Long, tmp() As String, SumTotal As Integer
   Dim i As Integer, ColTitle As String
   Dim No_of_Items As Long ' remove data items soon!!!
   Set FLX = FRMHost.Controls("FLXFourier")
   ReDim arrPower_Period(FLX.Rows - 2, 1) As Variant
   ReDim arrPower_Flip(UBound(arrPower_Period), 1) As Variant


   '==========Init and Fill Array from Fourier==========
   For idata = 0 To UBound(arrPower_Period)
      arrPower_Period(idata, 0) = FLX.TextMatrix((idata + 1), 0)
      arrPower_Period(idata, 1) = FLX.TextMatrix((idata + 1), 1)
   Next idata

   '===Get number of Data Items used to produce Fourier====
   '#Detrend#
   For i = 0 To FRMHost.OptDetrend.Count - 1
      If FRMHost.OptDetrend.Item(i).Value = True Then
         Select Case i
         Case 0  'None
            SumTotal = SumTotal
            Exit For
         Case 1  'Diff + Mean
            SumTotal = SumTotal - 1
            Exit For
         Case 2  'Regression
            SumTotal = SumTotal
            Exit For
         Case 3  'End to End
            SumTotal = SumTotal
            Exit For
         End Select
      End If
   Next i

   '#Mpa#
   For i = 0 To FRMHost.OptMpa.Count - 1
      If FRMHost.OptMpa.Item(i).Value = True Then
         Select Case i
         Case 0  'None
            SumTotal = SumTotal
            Exit For
         Case 1  '3pt
            SumTotal = SumTotal - 1
            Exit For
         Case 2  '5pt
            SumTotal = SumTotal - 2
            Exit For
         Case 3  'xpt
            SumTotal = SumTotal - Round((x_Point_MPA - 1) / 2)
            Exit For
         End Select
      End If
   Next i

   'Data_items = (FRMHost.FLXRaw.Rows - 1) + SumTotal
   No_of_Items = (FRMHost.FLXRaw.Rows - 1) + SumTotal
   '==============Convert Frequency to Period=============
   'Periodicity = Frequency / Time
   For idata = 0 To UBound(arrPower_Period)

      tmp = Split(arrPower_Period(idata, 0), "=")
      arrPower_Period(idata, 0) = (No_of_Items / tmp(1)) * Time_Val
   Next idata

   '====================Flip Array Visuallly==============
```

```
    For idata = 0 To UBound(arrPower_Period)
       arrPower_Flip(idata, 1) = arrPower_Period(UBound(arrPower_Period) - idata, 1)
       arrPower_Flip(idata, 0) = arrPower_Period(UBound(arrPower_Period) - idata, 0)
    Next idata
    '---------------------------------------------------------------------

    '======Normalise or Absolute Graph?=======
    If FRMHost.cmdFourier_Abs_Norm.Caption = "Normalise" Then
       ColTitle = "Absolute Power"
    Else
       ColTitle = "Normalised Power (%)"
    End If

    '==================Fill FLX=================
    Call Fill_Flx(arrPower_Flip, "Power")

    '==============Position Controls===============
    Call Format_Controls("Power", 16, "Period", ColTitle)

    '==================Draw CHT=================
    FRMHost.Controls("ChtPower").ChartData = arrPower_Flip
    FRMHost.Controls("ChtPower").Title = ColTitle
    Call Select_Tab("Power")
    Call AreaUnderCurve



End Sub
```

```
Option Explicit
'This module just plots the unchanged data in single or duplicate


Public Sub Draw_Raw()
   '=========Init Vars==============
   Dim last_Dataitem As Long, idata As Long, Streams As Integer
   Dim Heading_3 As String, Current_Pat_Name As String
   Dim FLX As MSFlexGrid, Current_Pat As Integer, iPat As Long

   Set FLX = FRMHost.FLXMain

   last_Dataitem = Find_Last_DataItem(FLX.Row) - 3

   If last_Dataitem < 0 Then
      FRMHost.Controls("chtraw").ColumnCount = 0
      FRMHost.Controls("FLXraw").Clear
      Exit Sub
   End If

   Current_Pat_Name = FLX.TextMatrix(FRMHost.FLXMain.Row, 0)
   FRMHost.LBLInfo = Current_Pat_Name
   '============Init Data Streams=========
   If Duplicate = True Then
      Streams = 2
      Heading_3 = "Values"
   Else
      Streams = 1
   End If

   '=========Find Corrasponding Patient Data=======
   For iPat = 0 To UBound(arrRaw)
      If arrRaw(0, iPat) = Current_Pat_Name Then
         Current_Pat = iPat
         Exit For
      End If
   Next iPat


   '============Copy relevant data==============
   ReDim arrMain(last_Dataitem, Streams) As Variant

   For idata = 0 To last_Dataitem
      arrMain(idata, 0) = CStr((Time_Val * idata)) & " " & Time_Type
      ' If Duplicate = False Then
      '    arrMain(idata, 0) = arrMain(idata, 0) & " " & Time_Type
      ' End If
      arrMain(idata, 1) = arrRaw(idata + 3, Current_Pat)

      If Duplicate = True Then
         arrMain(idata, 2) = arrRaw(idata + 3, Current_Pat + Dupe_data_Start)
      End If
   Next idata

   '===temp normalise test====
   If FRMHost.Check1.Value = 1 Then
      'Dim arrnorm As Variant
      arrMain = Normalise_RAW(arrMain())
   End If
   '===============Fill FLX =================
   Call Fill_Flx(arrMain, "Raw")

   '==========Format CHT & FLX===============
   'If Duplicate = True Then
   '  Call Format_Controls("Raw", 16, "Time", "Values", Heading_3)
   'Else
   Call Format_Controls("Raw", 3, "Time", "Values", Heading_3)
   'End If
   '===============Fill CHT==================
   arrMain = Filter_StarsOut(arrMain())
   FRMHost.Controls("Chtraw").ChartData = arrMain
   'FRMHost.Controls("Chtraw").Plot.SeriesCollection(1).SeriesMarker.Show = True
   '====IF Duplicate redraw second line undotted====
   'If Duplicate = True Then
   '    FRMHost.Controls("Chtraw").Plot.SeriesCollection(2).Pen.VtColor.Set 13, 145, 13
   '    FRMHost.Controls("Chtraw").Plot.SeriesCollection(2).Pen.Style = VtPenStyleSolid
   '    FRMHost.Controls("Chtraw").Plot.SeriesCollection(2).Pen.Width = 1
```

```
   'End If

   '#OPTIONS
   If FRMHost.chkMean.Value = 0 Then
      Call Add_Options
   End If

End Sub

Public Sub Draw_Mean_And_SEM()
   '---Init vars---
   Dim arrData() As Variant

   Dim irow As Integer, idata As Integer, Least As Integer, iPat As Integer
   Dim FLX As MSFlexGrid
   Set FLX = FRMHost.FLXMain

   '-Get least datapoints-
   Least = Get_Least

   '-Define Muge array-
   ReDim Arr_of_All(Least - 1, 2, FLX.Rows - 3) As Variant

   '-SWap pats-
   Call Swap_Selected_Pats(True)

   '-Loop thru pats-
   For irow = 1 To FLX.Rows - 2
      FLX.Row = irow
      FRMHost.FLXMain_SelChange

      arrData() = What_Flx("Raw")


      '-Resize Least if needed-
      If Least - 1 >= UBound(arrData) Then
         Least = (UBound(arrData) + 1)
      End If

      For idata = 0 To Least - 1

         Arr_of_All(idata, 0, irow - 1) = arrData(idata, 0)
         Arr_of_All(idata, 1, irow - 1) = arrData(idata, 1)
         Arr_of_All(idata, 2, irow - 1) = "Real"
      Next idata
   Next


   '-swap Pats-
   Call Swap_Selected_Pats(False)

   '=========Average Data==========
   ReDim arrGroup_Mean(Least - 1, 2) As Variant

   For idata = 0 To UBound(arrGroup_Mean)
      For iPat = 0 To UBound(Arr_of_All, 3)
         arrGroup_Mean(idata, 0) = Arr_of_All(idata, 0, iPat)
         arrGroup_Mean(idata, 1) = arrGroup_Mean(idata, 1) + CDbl(Arr_of_All(idata, 1, iPat))


         arrGroup_Mean(idata, 2) = "Real"

      Next iPat
   Next idata

   '====Get Mean====
   Dim tmp() As String
   For idata = 0 To UBound(arrGroup_Mean)
      tmp = Split(arrGroup_Mean(idata, 0), " ")
      arrGroup_Mean(idata, 0) = tmp(0)
      arrGroup_Mean(idata, 1) = arrGroup_Mean(idata, 1) / (FLX.Rows - 2)
   Next idata

   '============= Calculate SEM ================
   ReDim arrOf_SEM(UBound(arrGroup_Mean)) As Variant
   ReDim arrOf_Merge(UBound(arrGroup_Mean), 1) As Variant

   arrOf_SEM = SEM_Group(Arr_of_All, arrGroup_Mean)
```

```
   arrOf_Merge = MergeSEM_Group(arrOf_SEM, arrGroup_Mean)


   '======Create on Array just for FLX=======
   ReDim arrOf_FLX(UBound(arrGroup_Mean), 2) As Variant

   '#Add Ave#
   For idata = 0 To UBound(arrOf_FLX)
      arrOf_FLX(idata, 0) = (Time_Val * idata) + 1 & " " & Time_Type
      arrOf_FLX(idata, 1) = arrGroup_Mean(idata, 1)
      arrOf_FLX(idata, 2) = arrOf_SEM(idata)
   Next idata

   '==================Fill FLX=================
   Call Fill_Flx(arrOf_FLX, "Mean_SEM")



   '==============Position Controls===============
   Call Format_Controls("Mean_SEM", 16, "N", "Mean of Raw", "SEM")

   '==================Draw CHT=================
   'arrOf_Merge = Filter_StarsOut(arrOf_Merge())
   FRMHost.Controls("ChtMean_SEM").ChartData = arrOf_Merge

   FRMHost.Tabmain.Tabs(FRMHost.Tabmain.Tabs.Count).Selected = True
   Call Add_Options

End Sub


Sub Add_TrendLine()
   'y=b1X +B0
   'B1 = (Sum(XY)-(Sum(X)-Sum(Y)/N)) / Sum(X^2) - Sum(X)^2/N
   'B0 = MeanY-B1*MeanX
   'NOT SET UP FOR DUPLICATE DATA

   '---Init Vars---
   Dim SumXY As Double
   Dim SumX As Double
   Dim SumY As Double
   Dim SumX2 As Double
   Dim n As Long
   Dim MeanX As Double
   Dim MeanY As Double
   Dim B0 As Double, B1 As Double
   Dim FLX As MSFlexGrid, CHT As MSChart

   Dim sum As Double, dValue As Double
   Dim irow As Long, i As Long
   Dim arrTmp() As String

   '---Draw on Raw or Mean---
   If FRMHost.Tabmain.SelectedItem.Image = "Raw" Then
      Set FLX = FRMHost.FLXRaw
      Set CHT = FRMHost.CHTRaw
   Else
      Set FLX = FRMHost.Controls("FLXMean_SEM")
      Set CHT = FRMHost.Controls("CHTMean_SEM")
   End If

   ReDim arrFLX_RAW(FLX.Rows - 2, 1) As Variant 'Array for Raw Data

   '---Get Raw FLX Data---
   For irow = 1 To FLX.Rows - 1
      arrTmp() = Split(FLX.TextMatrix(irow, 0), " ")
      arrFLX_RAW(irow - 1, 0) = arrTmp(0)
      arrFLX_RAW(irow - 1, 1) = FLX.TextMatrix(irow, 1)
   Next irow

   arrFLX_RAW = Filter_StarsOut(arrFLX_RAW)

   '---Calculate Vars---
   For i = 0 To UBound(arrFLX_RAW)
      SumXY = SumXY + (arrFLX_RAW(i, 0) * arrFLX_RAW(i, 1))
      SumX = SumX + arrFLX_RAW(i, 0)
      SumY = SumY + arrFLX_RAW(i, 1)
      SumX2 = SumX2 + (arrFLX_RAW(i, 0) * arrFLX_RAW(i, 0))
```

```
   Next i

   n = i
   MeanX = SumX / n
   MeanY = SumY / n

   '---Calculate B0 & B1---
   B1 = (SumXY - ((SumX * SumY) / n)) / (SumX2 - ((SumX ^ 2) / n))
   B0 = MeanY - B1 * MeanX

   '---calculate trend line array---
   ReDim arrTrend(UBound(arrFLX_RAW), 2) As String

   For i = 0 To UBound(arrTrend)
      arrTrend(i, 0) = arrFLX_RAW(i, 0) & " " & Time_Type
      arrTrend(i, 1) = arrFLX_RAW(i, 1)
      arrTrend(i, 2) = (B1 * arrFLX_RAW(i, 0)) + B0
   Next i

   '---Add to CHT Raw or Mean---
   CHT.ChartData = arrTrend
   CHT.Plot.SeriesCollection(2).Pen.VtColor.Set 0, 0, 0
   CHT.Plot.SeriesCollection(2).Pen.Style = VtPenStyleDitted
   CHT.Plot.SeriesCollection(2).Pen.Width = 2
   '--title--
   Dim sign As String
   If Sgn(B0) = 1 Then
      sign = "+ "
   ElseIf Sgn(B0) = 0 Then
      sign = ""
   Else
      sign = "- "
   End If

   FRMHost.CHTRaw.TitleText = FRMHost.CHTRaw.TitleText & " (Trend Y = " & Format(B1, "0.00") & "X " & sign
»& Format(B0, "0.00") & ")"


End Sub

Function Normalise_RAW(ByRef arrInput()) As Variant

   '----init vars----
   Dim idata As Integer, i As Integer, SumTotal As Double, WhatToDo As String
   ReDim arrOutput(UBound(arrInput), UBound(arrInput, 2)) As Variant

   '===========Normalise the Data=========

   '#Work out whether to Multiply up or Down to get to 100%#
   For idata = 0 To UBound(arrInput)
      SumTotal = SumTotal + arrInput(idata, 1)
   Next idata

   If SumTotal > 100 Then
      WhatToDo = "divide"
   Else
      WhatToDo = "multiply"
   End If


   SumTotal = SumTotal / 100
   '#########################################################


   '####################Normalise it#########################
   For i = 0 To UBound(arrInput)
      arrOutput(i, 0) = arrInput(i, 0)

      '------Divide 0/0 Fix------------
      If arrInput(i, 1) = 0 And SumTotal = 0 Then Exit For

      If WhatToDo = "multiply" Then
         If SumTotal < 1 Then
            arrOutput(i, 1) = arrInput(i, 1) / SumTotal
         Else
            arrOutput(i, 1) = arrInput(i, 1) * SumTotal
         End If
```

```
      Else
         arrOutput(i, 1) = arrInput(i, 1) / SumTotal
      End If

   Next i
   '#########################################################

   '=====Output for FLX======
   Normalise_RAW = arrOutput
End Function
```

```
Sub Calculate_Randomness() 'As Boolean
   On Error GoTo Handler
   '=Get Data=
   Dim arrPatient() As Variant
   Dim dblTMP As Double, Median As Double, dblTMP2 As Double, strTMP As String
   Dim No_Of_A As Long, No_Of_B As Long
   Dim i As Long, j As Long, iTmp As Long, Runs As Long
   arrPatient = What_Flx()

   '---escape clause---
   'If UBound(arrPatient) > 100 Then Exit Sub
   '--reformat arrPatient-
   ReDim arrPat(UBound(arrPatient), UBound(arrPatient, 2)) As String
   Dim tmp() As String
   For i = 0 To UBound(arrPat)
      tmp = Split(arrPatient(i, 0), " ")
      arrPat(i, 0) = CInt(tmp(0))
      arrPat(i, 1) = CDbl(arrPatient(i, 1))
   Next i

   '=Find Median=
   '-Bubble Sort it-
   For i = 0 To UBound(arrPat)
      For j = 0 To UBound(arrPat)
         If CDbl(arrPat(i, 1)) < CDbl(arrPat(j, 1)) Then
            'swap em-
            dblTMP = arrPat(i, 1)
            arrPat(i, 1) = arrPat(j, 1)
            arrPat(j, 1) = dblTMP

            dblTMP2 = arrPat(i, 0)
            arrPat(i, 0) = arrPat(j, 0)
            arrPat(j, 0) = dblTMP2
         End If
      Next j
   Next i

   '-Take middle (median)-
   If IsEven(UBound(arrPat) + 1) Then
      iTmp = Floor(UBound(arrPat) / 2)
      Median = CDbl((arrPat(iTmp, 1)) + CDbl(arrPat(iTmp + 1, 1))) / 2
   Else
      iTmp = CInt(UBound(arrPat) / 2)
      Median = arrPat(iTmp, 1)
   End If

   '=Varible Replace=
   'Also counting A,B's
   For i = 0 To UBound(arrPat)
      If arrPat(i, 1) < Median Then
         arrPat(i, 1) = "A"
         No_Of_A = No_Of_A + 1
      ElseIf arrPat(i, 1) = Median Then
         arrPat(i, 1) = ""
      Else
         arrPat(i, 1) = "B"
         No_Of_B = No_Of_B + 1
      End If
   Next i

   '===Resort array back===
   For i = 0 To UBound(arrPat)
      For j = 0 To UBound(arrPat)
         If CDbl(arrPat(i, 0)) < CDbl(arrPat(j, 0)) Then
            'swap em-
            strTMP = arrPat(i, 1)
            arrPat(i, 1) = arrPat(j, 1)
            arrPat(j, 1) = strTMP

            dblTMP2 = arrPat(i, 0)
            arrPat(i, 0) = arrPat(j, 0)
            arrPat(j, 0) = dblTMP2
         End If
      Next j
   Next i

   '=Count Runs of Letters=
```

```
   Runs = 1
   For i = 0 To UBound(arrPat) - 1
      If arrPat(i, 1) <> "" Then
         If arrPat(i, 1) <> arrPat(i + 1, 1) Then
            Runs = Runs + 1
         End If
      End If
   Next i

   Debug.Print ("Median " & Median & " A " & No_Of_A & " B" & No_Of_B & " Runs " & Runs)

   '---set vars for probability calc---
   Dim m As Integer, n As Integer
   If No_Of_A >= No_Of_B Then
      n = No_Of_A
      m = No_Of_B
   Else
      n = No_Of_B
      m = No_Of_A
   End If

   Dim p As Double
   Dim p_out As String
   p = Calculate_Probability(m, n, Runs)

   Select Case p
   Case Is > 0.05
      p_out = vbCrLf & "P is N.S. Null Hypothesis is false"
   Case Is <= 0.01
      p_out = vbCrLf & "P <0.01 Null Hypothesis is true"
   Case Is <= 0.05
      p_out = vbCrLf & "P <0.05 Null Hypothesis is true"
   End Select

   MsgBox "Probability of this being random data P = " & Format(p, "0.00000") & " (range 0 to 1)" & p_out
   Debug.Print "Probability of this being random data P=" & p & " or " & Format((p * 100), "0.00000") & "%
»" & p_out

   Exit Sub
   Handler:
   If Err.Number = 6 Then
      MsgBox ("Dataset too large to calculate Randomness")
   Else
      MsgBox Err.Number & " " & Err.Description
   End If

End Sub

Function Calculate_Probability(m As Integer, n As Integer, U_prime As Long) As Double
   'Equation  from tables for testing randomness of grouping by frieda s swed et al

   '---init vars---
   Dim p As Double, Left_Bit As Variant, Right_Bit As Variant, Right_Bit1 As Variant, Right_Bit2 As Variant
», Right_Bit3 As Variant, Right_Bit4 As Variant
   Dim k As Integer, U As Integer

   '===left bit = 1/ total!/M! * n!===
   Left_Bit = Combitorial(m + n, m)
   'Left_Bit = 1 / Left_Bit

   For U = 2 To U_prime
      '-Set K-
      If IsEven(U) = True Then
         k = U / 2
      Else
         k = (U + 1) / 2
      End If

      If U = 17 Then
         Debug.Print ""
      End If
      'Right bit = 2 * total!/ m-1! * k-1! * total!/ n-1! * k-1!
      If IsEven(U) = True Then
         Right_Bit = 2 * Combitorial(m - 1, k - 1) * Combitorial(n - 1, k - 1)
      Else
         rightbit1 = Combitorial(m - 1, k - 1)
         rightbit2 = Combitorial(n - 1, k - 2)
```

```
         rightbit3 = Combitorial(m - 1, k - 2)
         rightbit4 = Combitorial(n - 1, k - 1)

         Right_Bit = (rightbit1 * rightbit2) + (rightbit3 * rightbit4)
      End If

      p = p + (Right_Bit / Left_Bit)

   Next U
   'P = P * Left_Bit

   Calculate_Probability = p

End Function

Function Factorial(ByVal i As Integer) As Variant
   Dim sum As Variant
   '-FIX FOR zERO-
   If i = 0 Then
      Factorial = 1: Exit Function
   ElseIf i = 1 Then
      Factorial = 1: Exit Function
   End If
   '-Calc Factorial-
   sum = i
   Do While i > 1
      i = i - 1
      sum = sum * i
   Loop

   Factorial = sum
End Function

Public Function IsEven(ByVal Number As Long) As Boolean
   IsEven = (Number Mod 2 = 0)
End Function

Public Function Combitorial(m As Integer, n As Integer) As Variant

   Dim top As Variant, bot As Variant
   top = Factorial(m)
   bot = Factorial(n) * Factorial(m - n)
   Combitorial = top / bot
   Exit Function
   Handler:

End Function
```

```
Option Explicit

Sub Draw_Serial_Start()
   '===============Init Vars=================
   Dim iTab As Integer, iCht_Type As Integer
   iCht_Type = 3

   '=========Get Data from Raw,Mpa,or Det======
   Dim arrSerial_Start() As Variant
   arrSerial_Start = What_Flx()

   '==============Position Controls============
   Call Format_Controls("Serial", iCht_Type, "Patient", "P.O.I.")

   '=================Draw CHT=================
   FRMHost.Controls("ChtSerial").ChartData = arrSerial_Start

   '===============Clean Up==================
End Sub

Function Serial_Analyse_Single(Optional Points_to_analyse As String, Optional No_SEM As Boolean, Optional
»Which_Pat As Integer) As Variant
   '==================Init Vars================
   Dim arrSerial_Start() As Variant, FLX As MSFlexGrid, irow As Integer, idata As Long
   Dim jData As Integer, iPOI As Integer, i As Integer, No_Of_Pats As Integer
   Dim jRow As Integer
   Set FLX = FRMHost.Controls("FLXserial")


   '=================Get Data==================
   arrSerial_Start = What_Flx()


   '============Get Points of Interest===========
   '---Find Out Count per Pat---
   If Which_Pat = 0 Then Which_Pat = FRMHost.FLXMain.RowSel

   '---Init Array for POI---
   ReDim arrSerial_POI(0) As String

   '---Fill Array---
   For irow = 1 To FLX.Rows - 1
      If FLX.TextMatrix(irow, 0) = FRMHost.FLXMain.TextMatrix(Which_Pat, 0) Then
         If idata > UBound(arrSerial_POI) Then
            ReDim Preserve arrSerial_POI(idata)
         End If

         arrSerial_POI(idata) = FLX.TextMatrix(irow, 1)
         idata = idata + 1

      End If
   Next irow

   '---Sort Array---
   arrSerial_POI() = Bubble_Sort(arrSerial_POI())

   '======Request How many points to Analyse======
   '----Get----
   If Points_to_analyse = "" Then
      Points_to_analyse = InputBox("How many datapoints would you like to analyse?")
   End If
   '---Checks---
   '#Numeric#
   Do While IsNumeric(Points_to_analyse) = False
PassBack:
      Points_to_analyse = InputBox("How many datapoints would you like to analyse?")
   Loop
   '#In Range#
   For idata = 0 To UBound(arrSerial_POI)
      If CInt(Points_to_analyse) + CInt(arrSerial_POI(idata)) > UBound(arrSerial_Start) Then
         MsgBox "The number of datapoints chosen is outside the range of data, please choose again"
         GoTo PassBack
      End If


   Next idata
```

```
   '#Overlaps#
   For idata = 0 To UBound(arrSerial_POI)
      For jData = idata + 1 To UBound(arrSerial_POI)
         If CInt(arrSerial_POI(idata)) + CInt(Points_to_analyse) > arrSerial_POI(jData) Then
            MsgBox ("The number of datapoints chosen overlaps another Point of interest,please choose
»again")
            GoTo PassBack
         End If
      Next jData
   Next idata

   '=====Select Relevant part of Data Array=======

   '----Init Array-----
   ReDim arrSerial_Cut(CInt(Points_to_analyse) - 1, 2, UBound(arrSerial_POI)) As Variant

   '-----Get Data-----
   For iPOI = 0 To UBound(arrSerial_POI)
      For idata = 0 To Points_to_analyse - 1
         arrSerial_Cut(idata, 0, iPOI) = arrSerial_Start(CInt(arrSerial_POI(iPOI) - 1) + idata, 0)
         arrSerial_Cut(idata, 1, iPOI) = arrSerial_Start(CInt(arrSerial_POI(iPOI) - 1) + idata, 1)
         If Duplicate = True And (FRMHost.Tabmain.Tabs(1).Image = "Raw" And FRMHost.Tabmain.Tabs(1).
»Selected = True) Then
            arrSerial_Cut(idata, 2, iPOI) = arrSerial_Start(CInt(arrSerial_POI(iPOI) - 1) + idata, 2)
         Else
            arrSerial_Cut(idata, 2, iPOI) = "Real"
         End If
      Next idata
   Next iPOI

   '======Average Relevant Parts of Data Array====
   ReDim arrSerial_AVE(UBound(arrSerial_Cut), 2) As Variant
   '----Get N----
   Dim n As Integer
   If Duplicate = True Then
      n = ((UBound(arrSerial_Cut, 3) + 1) * 2)
   Else
      n = (UBound(arrSerial_Cut, 3) + 1)
   End If

   For idata = 0 To UBound(arrSerial_Cut)
      For iPOI = 0 To UBound(arrSerial_Cut, 3)
         arrSerial_AVE(idata, 0) = idata 'arrSerial_Cut(iData, 0, iPOI)
         arrSerial_AVE(idata, 1) = CDbl(arrSerial_AVE(idata, 1)) + CDbl(arrSerial_Cut(idata, 1, iPOI))
         If Duplicate = True And (FRMHost.Tabmain.Tabs(1).Image = "Raw" And FRMHost.Tabmain.Tabs(1).
»Selected = True) Then
            arrSerial_AVE(idata, 1) = CDbl(arrSerial_AVE(idata, 1)) + CDbl(arrSerial_Cut(idata, 2, iPOI))
         End If
         arrSerial_AVE(idata, 2) = "Real"
      Next iPOI

      arrSerial_AVE(idata, 1) = arrSerial_AVE(idata, 1) / n
   Next idata


   '=========Work out SEM and add it on=======
   If No_SEM = False Then
      Dim arrSerial_SEM() As Variant
      Dim arrSerial_ALL() As Variant

      arrSerial_SEM = SEM_Group(arrSerial_Cut(), arrSerial_AVE())
      arrSerial_ALL = MergeSEM_Group(arrSerial_SEM(), arrSerial_AVE())
      '-------------------Fill FLX-----------------
      For i = 0 To UBound(arrSerial_AVE)
         arrSerial_AVE(i, 2) = arrSerial_SEM(i)
      Next i
      Call Fill_Flx(arrSerial_AVE, "Serial")

      '-------------------Fill CHT------------------
      FRMHost.Controls("ChtSerial").ChartData = arrSerial_ALL
      '--------------Format CHT & FLX---------------
      Call Format_Controls("Serial", 16, "N", "Ave. Values", "SEM")
   ElseIf No_SEM = True And Which_Pat = 0 Then 'ONLY draw if 1 point on 1 cht selected
      '-------------------Fill FLX-----------------
      Call Fill_Flx(arrSerial_AVE, "Serial")
```

```
      '-------------------CUT Down Array------------
      ReDim arrCut(UBound(arrSerial_AVE), 1) As String

      For i = 0 To UBound(arrSerial_AVE)
         arrCut(i, 0) = arrSerial_AVE(i, 0)
         arrCut(i, 1) = arrSerial_AVE(i, 1)
      Next i
      '-------------------Fill CHT------------------
      FRMHost.Controls("ChtSerial").ChartData = arrCut
      '--------------Format CHT & FLX---------------
      Call Format_Controls("Serial", 16, "N", "Ave. Values")
   End If

   '====Return main Array====
   Serial_Analyse_Single = arrSerial_AVE()

   '------remove CHT title--------
   FRMHost.CHTSerial.TitleText = ""

   '----Clean Up-----
   Set FLX = Nothing
End Function


Sub Draw_Serial_Group()
   '--------Init Vars---------
   Dim FLX As MSFlexGrid, iPat As Integer, Points_to_analyse As String
   Dim arrOF_Vals() As Variant, FRM As Form, iPOI As Integer, idata As Long
   Dim Sum_Total As Double, i As Integer
   Static arrOf_ALL(Max_Data_Items, 2, Max_Patients) As Variant

   Set FLX = FRMHost.FLXMain: Set FRM = FRMHost
   FLX.Row = 1
   '====Get Points to analyse====
   Points_to_analyse = InputBox("How many datapoints would you like to analyse?")
   '=============== Loop Through all Patients ===============
   For iPat = 0 To FLX.Rows - 3

      '----- Set Row on FLX -----
      FLX.Row = iPat + 1
      '================= Get Array of vals ===================

      '    If IsEmpty(Serial_Analyse_Single(Points_to_analyse, True)) Then
      '        MsgBox ("You haven't chosen points for all patients")
      '        Exit Sub
      '    Else
      arrOF_Vals() = Serial_Analyse_Single(Points_to_analyse, True, FLX.Row)
      '    End If
      '----- Store Vars in Array -----
      For idata = 0 To UBound(arrOF_Vals)
         '-----Store Averaged Data ------
         arrOf_ALL(idata, 0, FLX.Row - 1) = arrOF_Vals(idata, 0)
         arrOf_ALL(idata, 1, FLX.Row - 1) = arrOF_Vals(idata, 1)
         arrOf_ALL(idata, 2, FLX.Row - 1) = "Real"

      Next idata


      '===Move to Next Patient===

      FRM.FLXMain.Row = FRM.FLXMain.Row + 1
      Call FRM.FLXMain_SelChange

   Next iPat

   '====Average Vals Array=========
   '======Average Relevant Parts of Data Array====
   ReDim arrOF_Ave(UBound(arrOF_Vals), 2) As Variant

   For idata = 0 To UBound(arrOf_ALL)
      '-----Escape Clause------
      If arrOf_ALL(idata, 1, 0) = "" Then Exit For

      For iPOI = 0 To UBound(arrOf_ALL, 3)
         '----Escape Clause---
         If arrOf_ALL(idata, 1, iPOI) = "" Then Exit For

         arrOF_Ave(idata, 0) = idata + 1
```

```
            arrOF_Ave(idata, 1) = CDbl(arrOF_Ave(idata, 1)) + CDbl(arrOf_ALL(idata, 1, iPOI))
            arrOF_Ave(idata, 2) = "Real"
        Next iPOI

        arrOF_Ave(idata, 1) = arrOF_Ave(idata, 1) / (iPOI)

    Next idata



    '----Add on SEM----
    Dim arrOf_SEM() As Variant
    Dim arrOf_Final() As Variant

    arrOf_SEM = SEM_Group(arrOf_ALL(), arrOF_Ave())
    arrOf_Final = MergeSEM_Group(arrOf_SEM(), arrOF_Ave())


    '-------------------Fill FLX-----------------
    For i = 0 To UBound(arrOF_Ave)
        arrOF_Ave(i, 1) = Format(arrOF_Ave(i, 1), "0.000")
    Next i

    Call Fill_Flx(arrOF_Ave, "Serial")

    '-------------------Fill CHT-----------------
    FRMHost.Controls("ChtSerial").ChartData = arrOf_Final

    '--------------Format CHT & FLX---------------
    Call Format_Controls("Serial", 16, "N", "Ave. Values")



    '----Clean Up-----
    Erase arrOf_ALL
    Erase arrOF_Vals
    Erase arrOf_SEM
    Erase arrOf_Final
    Set FLX = Nothing
    Set FRM = Nothing
End Sub
```

EASY TSA - MODULE - MDL_X_Factor.bas **MDL_X_Factor**

```
Option Explicit
Dim Bin_Range(10, 2) As Double
Dim myDupe As Boolean
'This Module Calculates the X Factor of a patients Glucose Control
Public Sub Display_GA_Value()
   '-Init Vars-
   Dim GA_Value As String, GA_Value2 As Double
   Dim FLX As MSFlexGrid
   Dim CHT As MSChart
   Dim tmp() As String

   Set FLX = FRMHost.FLXMain

   Set CHT = FRMHost.Controls("CHT" & What_Tab)

   '---Calc GA Value---
   GA_Value = Calc_GA

   '-Display X Factor-
   'lose existing stuff-
   CHT.TitleText = ""
   If myDupe = True Then
      tmp = Split(GA_Value, "|")
      GA_Value = tmp(0)
      GA_Value2 = tmp(1)
      CHT.TitleText = CHT.TitleText & "[GA Value 1 is " & Format(GA_Value, "0.00") & ",GA Value 2 is " &
»Format(GA_Value2, "0.00") & "]"
   Else

      CHT.TitleText = CHT.TitleText & "[GA Value is " & Format(GA_Value, "0.00") & "]"

   End If




   '---Clean Up---
   Set FLX = Nothing
   Set CHT = Nothing
End Sub
Private Function Calc_GA() As String
   On Error GoTo Handler
   '---Init Vars---
   Dim FLX As MSFlexGrid
   Set FLX = FRMHost.FLXRaw                'FLX Raw
   Dim sum As Double, dValue As Double
   Dim irow As Integer, i As Integer
   Dim Sum2 As Double, dValue2 As Double

   Dim arrFLX_RAW() As Variant
   '---Get Raw FLX Data---
   arrFLX_RAW = What_Flx()

   '--Calc GA--
   For i = 0 To UBound(arrFLX_RAW)
      dValue = 10 ^ (85 * (Log10(Log10(arrFLX_RAW(i, 1))) - Log10(Log10(5))) ^ 2)
      Debug.Print dValue
      sum = sum + dValue

      If Duplicate = True Then
         If FRMHost.Tabmain.SelectedItem.Key = "Raw" Then
            myDupe = True
            dValue2 = 10 ^ (85 * (Log10(Log10(arrFLX_RAW(i, 2))) - Log10(Log10(5))) ^ 2)
            Sum2 = Sum2 + dValue2
         End If
      End If
   Next i

   '---return GA Value----
   If myDupe = True Then
      Calc_GA = (sum / i) & "|" & (Sum2 / i)
   Else
      Calc_GA = Log10(sum / i)
   End If
   Exit Function
   Handler:
```

```
   dValue = 0
   dValue2 = 0
   Resume Next
End Function


'Private Function Distribute_Data_To_Bins() As Double 'OLD STUFF
''---Init Vars---
'Dim arrFreq(10) As Integer              'Array for Freq of Raw
'Dim FLX As MSFlexGrid
'Set FLX = FRMHost.FLXRaw                'FLX Raw
'ReDim arrFLX_RAW(FLX.Rows - 2, 1) As Variant 'Array for Raw Data
'Dim X_Factor As Integer, irow As Integer
'Dim G_Value As Double
'
''---Get Raw FLX Data---
'For irow = 1 To FLX.Rows - 1
'    arrFLX_RAW(irow - 1, 0) = FLX.TextMatrix(irow, 0)
'    arrFLX_RAW(irow - 1, 1) = FLX.TextMatrix(irow, 1)
'Next irow
'
'arrFLX_RAW = Filter_StarsOut(arrFLX_RAW)
'
'
''---Distibute to Bins---
'For irow = 0 To UBound(arrFLX_RAW)
'    '---Set Glucose Var---
'    G_Value = arrFLX_RAW(irow, 1)
'    '---Assign to Bin---
'
'        If (G_Value >= Bin_Range(0, 0)) And (G_Value <= Bin_Range(0, 1)) Then
'            X_Factor = X_Factor + 10
'        ElseIf (G_Value >= Bin_Range(1, 0)) And (G_Value <= Bin_Range(1, 1)) Then
'            X_Factor = X_Factor + 8
'        ElseIf (G_Value >= Bin_Range(2, 0)) And (G_Value <= Bin_Range(2, 1)) Then
'            X_Factor = X_Factor + 6
'        ElseIf (G_Value >= Bin_Range(3, 0)) And (G_Value <= Bin_Range(3, 1)) Then
'            X_Factor = X_Factor + 4
'        ElseIf (G_Value >= Bin_Range(4, 0)) And (G_Value <= Bin_Range(4, 1)) Then
'            X_Factor = X_Factor + 2
'        ElseIf (G_Value >= Bin_Range(5, 0)) And (G_Value <= Bin_Range(5, 1)) Then
'            X_Factor = X_Factor + 0 'NORMAL
'        ElseIf (G_Value >= Bin_Range(6, 0)) And (G_Value <= Bin_Range(6, 1)) Then
'            X_Factor = X_Factor + 2
'        ElseIf (G_Value >= Bin_Range(7, 0)) And (G_Value <= Bin_Range(7, 1)) Then
'            X_Factor = X_Factor + 4
'        ElseIf (G_Value >= Bin_Range(8, 0)) And (G_Value <= Bin_Range(8, 1)) Then
'            X_Factor = X_Factor + 6
'        ElseIf (G_Value >= Bin_Range(9, 0)) And (G_Value <= Bin_Range(9, 1)) Then
'            X_Factor = X_Factor + 8
'        ElseIf (G_Value >= Bin_Range(10, 0)) And (G_Value <= Bin_Range(10, 1)) Then
'            X_Factor = X_Factor + 10
'        Else
'            MsgBox ("Error in Sub")
'            Exit Function
'        End If
'
'Next irow
'
''--return Var--
'Distribute_Data_To_Bins = X_Factor / UBound(arrFLX_RAW)
'
''--Clean Up--
'Set FLX = Nothing
'End Function
'
'Private Sub Define_Bins() 'OLD STUFF
''---Set Bin Size for local Use---
'
'Bin_Range(0, 0) = 0
'Bin_Range(0, 1) = 2
'Bin_Range(0, 2) = 10
'
'Bin_Range(1, 0) = 2
'Bin_Range(1, 1) = 2.5
'Bin_Range(1, 2) = 8
'
'Bin_Range(2, 0) = 2.5
```

```
'Bin_Range(2, 1) = 3
'Bin_Range(2, 2) = 6
'
'Bin_Range(3, 0) = 3
'Bin_Range(3, 1) = 3.5
'Bin_Range(3, 2) = 4
'
'Bin_Range(4, 0) = 3.5
'Bin_Range(4, 1) = 4
'Bin_Range(4, 2) = 2
'
''----NORMAL---->
'Bin_Range(5, 0) = 4
'Bin_Range(5, 1) = 7.8
'Bin_Range(5, 2) = 0
''-------------->
'
'Bin_Range(6, 0) = 7.8
'Bin_Range(6, 1) = 10.85
'Bin_Range(6, 2) = 2
'
'Bin_Range(7, 0) = 10.85
'Bin_Range(7, 1) = 13.9
'Bin_Range(7, 2) = 4
'
'Bin_Range(8, 0) = 13.9
'Bin_Range(8, 1) = 16.95
'Bin_Range(8, 2) = 6
'
'Bin_Range(9, 0) = 16.95
'Bin_Range(9, 1) = 20
'Bin_Range(6, 2) = 8
'
'Bin_Range(10, 0) = 20
'Bin_Range(10, 1) = 999
'Bin_Range(10, 2) = 10
'
'
'End Sub
'
'Public Sub Display_M_Value()
''-Init Vars-
'Dim M_Value As Double
'Dim FLX As MSFlexGrid
'Dim CHT As MSChart
'
'Set FLX = FRMHost.FLXMain
'Set CHT = FRMHost.CHTRaw
''---check if assay is glucose---
'If UCase(FLX.TextMatrix(FLX.Row, 1)) = "GLUCOSE" Then
'    M_Value = Calc_M_Value()
'
'End If
'
'CHT.TitleText = CHT.TitleText & " - [M Value is " & Format(M_Value, "0.00") & "/10]"
'
''--Clean Up--
'Set FLX = Nothing
'Set CHT = Nothing
'End Sub
'
'Private Function Calc_M_Value() As Double
''---Init Vars---
'Dim FLX As MSFlexGrid
'Set FLX = FRMHost.FLXRaw
'ReDim arrFLX_RAW(FLX.Rows - 2, 1) As Variant 'Array for Raw Data
'Dim M_Value As Double, irow As Integer
'Dim MBS As Double, BS As Double
'
''---Get Raw FLX Data---
'For irow = 1 To FLX.Rows - 1
'    arrFLX_RAW(irow - 1, 0) = FLX.TextMatrix(irow, 0)
'    arrFLX_RAW(irow - 1, 1) = FLX.TextMatrix(irow, 1)
'Next irow
'
'arrFLX_RAW = Filter_StarsOut(arrFLX_RAW)
'
''---calculate M Vals----
```

```
''--Mbs = (|10*log10(BS/120)|^3) /N-- BS=Glucose*18 --
''--Mw = big-low /N of sets--
''-- M = Mbs + Mw--
'For irow = 0 To UBound(arrFLX_RAW)
'    BS = Floor(arrFLX_RAW(irow, 1) * 18)
'    MBS = (10 * log10(BS / 120)) ^ 3
'
'    MBS = Abs(Floor(MBS))
'
'    M_Value = M_Value + MBS
'
'Next irow
'
''----Final Calculation----
'M_Value = M_Value / UBound(arrFLX_RAW)
'
'Calc_M_Value = Floor(M_Value)
'End Function
```

```
Option Explicit

Sub AreaUnderCurve()

   ' Sum of ((x+x1)/2) /y-y1 _
Where x = data and y=time

   '==========Init Vars===============
   Dim FLX As MSFlexGrid, Sum_Total As Double, CHT As MSChart, Data_Col_Count As Integer
   Dim irow As Long, top As Double, Bottom As Double, chop_off As Integer, icol As Integer
   Dim idata As Long, i As Integer, lastcol As Integer
   Dim var2() As String
   Dim var1() As String
   Set FLX = FRMHost.Controls("FLX" & What_Tab)
   Set CHT = FRMHost.Controls("CHT" & What_Tab)
   Dim currtab As String

   '===Choose How to use Time Data===
   If What_Tab = "Raw" Then
      'chop_off = 7
      '-----Is the Data in Duplicate---
      If Duplicate = True Then
         Data_Col_Count = 2
      Else
         Data_Col_Count = 1
      End If

      lastcol = 1
   ElseIf What_Tab = "Grouped_Power" Then
      ' chop_off = 0
      Data_Col_Count = 1
      lastcol = 2
   ElseIf What_Tab = "Power" Then
      ' chop_off = 0
      Data_Col_Count = 1
      lastcol = 1

   Else
      Exit Sub
   End If



   '============Get Data=============
   '---------Init Vars-----
   ReDim arrData(FLX.Rows - 2, Data_Col_Count) As Variant

   '----------Read Data-----------
   For icol = 0 To FLX.Cols - lastcol
      For irow = 1 To FLX.Rows - 1
         arrData(irow - 1, icol) = FLX.TextMatrix(irow, icol)
      Next irow
   Next icol

   '-------Strip Stars------
   arrData = Filter_StarsOut(arrData)

   '========Calculate Area===========
   '-----------Init Vars----------
   ReDim arrSum_Total(Data_Col_Count - 1) As Double
   idata = 1
   '--------Get Area from Data---
   Do While idata <= Data_Col_Count
      '#Calculate Area#
      For irow = 0 To UBound(arrData) - 1
         '---escape clause---
         'If arrData(irow + 1, iData) = "" Then GoTo End_of_Data
         top = (CDbl(arrData(irow + 1, idata)) + CDbl(arrData(irow, idata))) / 2
         'Bottom = Left(arrData(iRow + 1, 0), Len(arrData(iRow + 1, 0)) - chop_off) - Left(arrData(iRow,
»0), Len(arrData(iRow, 0)) - chop_off)
         var2 = Split(arrData(irow + 1, 0), " ")
         var1 = Split(arrData(irow, 0), " ")
         Bottom = CLng(var2(0)) - CLng(var1(0))
         ' If Bottom < 1 Then
         ' Sum_Total = Sum_Total + (top * Bottom)
         '  Else
```

```
         If Bottom <> 0 Then
            Sum_Total = Sum_Total + (top / Bottom)
         End If
         ' End If
      Next irow

      '#Store Area#
      arrSum_Total(idata - 1) = Sum_Total
      Sum_Total = 0
      idata = idata + 1
   Loop

   '==========OutPut Answer==========
   CHT.TitleText = "Area : "
   For i = 0 To UBound(arrSum_Total)
      CHT.TitleText = CHT.TitleText & " (Col " & i + 1 & ") = " & arrSum_Total(i)
   Next i

End Sub
```

```
Option Explicit

Sub Create_Delete_Controls(ByRef Type_Name As String, ByRef CreateIt As Boolean)

   '=================Init Vars=================
   Dim i As Integer
   Dim Tab_Number As Integer
   Dim tmpControl As Control
   Dim CHT_Exists As Boolean, FLX_Exists As Boolean
   Dim FRM As Form
   Set FRM = FRMHost
   '===============Check For Mpa Tab===========

   For i = 1 To FRM.Tabmain.Tabs.Count
      If FRMHost.Tabmain.Tabs(i).Image = Type_Name Then
         Tab_Number = i
         Exit For
      End If
   Next i

   '========Choose which Button Pushed========
   If CreateIt = False Then
      '~~~~~~~~~Delete MPA Tab~~~~~~~~~
      If Tab_Number <> 0 Then
         FRM.Tabmain.Tabs.Remove (Tab_Number)
      End If

      '~~~~~~~~Delete CHT & FLX~~~~~~~~~
      '#Check CHT does exist#

      For Each tmpControl In FRM.Controls
         If tmpControl.Name = "CHT" & Type_Name Then
            CHT_Exists = True
         ElseIf tmpControl.Name = "FLX" & Type_Name Then
            FLX_Exists = True
         End If


      Next tmpControl

      '#delete/hide the controls#
      If CHT_Exists = True Then
         If Type_Name <> "Serial" And Type_Name <> "Raw" Then
            FRM.Controls.Remove FRM.Controls("CHT" & Type_Name)
            '        ElseIf Type_Name <> "Raw" Then
            '            FRM.Controls.Remove FRM.Controls("CHT" & Type_Name)
         End If
      End If

      If FLX_Exists = True Then
         If Type_Name <> "Raw" Then
            FRM.Controls.Remove FRM.Controls("FLX" & Type_Name)
         Else
            FRM.Controls("FLX" & Type_Name).Visible = False
         End If
      End If


   Else
      '~~~~~~~~~~Add Tab~~~~~~~~~~
      If Tab_Number = 0 Then
         FRMHost.Tabmain.Tabs.Add , Type_Name, , Type_Name
      End If


      '~~~~~~~~~~Add CHT & FLX~~~~~~~~~
      '#Check CHT doesn't exist#

      For Each tmpControl In FRMHost.Controls
         If tmpControl.Name = "CHT" & Type_Name Then
            CHT_Exists = True
         ElseIf tmpControl.Name = "FLX" & Type_Name Then
            FLX_Exists = True
         End If
      Next tmpControl

      '#Add the controls#
```

```
      If CHT_Exists = False Then
         If Type_Name = "Raw" Then
            Set FRMHost.CHTRaw = FRM.Controls.Add("MSChart20lib.MSChart", "CHTRaw")
         Else
            FRM.Controls.Add "MSChart20lib.MSChart", "CHT" & Type_Name

         End If
      End If

      If FLX_Exists = False Then
         FRM.Controls.Add "MSFlexGridlib.MSFlexGrid", "FLX" & Type_Name

      Else
         FRM.Controls("FLX" & Type_Name).Visible = True
      End If

      '#Position Controls#
      Call PositionControls(Type_Name)


   End If
   '==========Resize Everything to Fit==========
   If FRMHost.Height > 361 Then
      Call Resize
   End If
   '=================Clean UP===================
   Set tmpControl = Nothing

End Sub

Sub PositionControls(ByRef Type_Name As String)
   '=================Init Vars=================
   Dim tmpCHT As Control
   Dim tmpFLX As Control
   Dim FRM As Form, Item As Control
   Set FRM = FRMHost
   '===============Position CHT================
   Set tmpCHT = FRM.Controls("CHT" & Type_Name)




   tmpCHT.Visible = True
   tmpCHT.Move _
         FRMHost.FRAMenu.Left + FRMHost.FRAMenu.Width + 230, _
         FRMHost.Tabmain.ClientTop + FRMHost.FRAGraphs.top + 100, _
         FRMHost.Tabmain.ClientWidth * 0.65, _
         FRMHost.Tabmain.ClientHeight - 250

   tmpCHT.ZOrder (0)
   '================Position FLX===============
   Set tmpFLX = FRM.Controls("FLX" & Type_Name)

   tmpFLX.Visible = True
   tmpFLX.Move _
         tmpCHT.Left + tmpCHT.Width + 100, _
         tmpCHT.top, _
         FRMHost.Tabmain.ClientWidth * 0.3, _
         FRMHost.Tabmain.ClientHeight - 250

   tmpFLX.ZOrder (0)

   '=================Clean UP===================
Clean_UP:
   Set tmpCHT = Nothing
   Set tmpFLX = Nothing
   Set FRM = Nothing
End Sub



Sub Reset_Controls()
   On Error Resume Next
   '========Init Vars============
   Dim cnt As Control

   '==========Reset Check Boxes============
   For Each cnt In FRMHost.Controls
```

```
      If UCase(Left(cnt.Name, 3)) = "CHK" Then
         cnt.Value = 0
      ElseIf UCase(Left(cnt.Name, 3)) = "OPT" Then
         cnt.Value = False
      End If
   Next cnt

   '============Reset Flx's============
   FRMHost.FLXMain.Clear
   FRMHost.FLXMain.Rows = 2
   FRMHost.FLXGrouped.Clear
   FRMHost.FLXGrouped.Rows = 2

   '=========Reset Option buttons========
   FRMHost.OptDetrend(0).Value = True
   FRMHost.OptMpa(0).Value = True

   '---reset Static Variable---
   If FRMHost.CHKDeConvolution.Value = 1 Then Call Perform_Deconv(True)

End Sub

Sub Fill_Flx(ByRef arrFill(), ByRef Control_Name As String)
   '=============Init Vars==============
   Dim FLX As MSFlexGrid
   Dim idata As Long, icol As Integer
   Set FLX = FRMHost.Controls("FLX" & Control_Name)

   '==========Format FLX========
   With FLX
      .Clear
      .Rows = UBound(arrFill) + 2
      .Cols = UBound(arrFill, 2) + 1
      .Col = 0
      .Appearance = vbFlat
   End With


   '==========Fill FLX=========
   For icol = 0 To UBound(arrFill, 2)
      FLX.Col = icol
      For idata = 0 To UBound(arrFill)
         FLX.Row = (idata + 1)
         If icol = 0 Then
            FLX.Text = arrFill(idata, icol)
         Else
            FLX.Text = Format(arrFill(idata, icol), "0.000000")
         End If
      Next idata
   Next icol
   '==========Clean Up==================

   Set FLX = Nothing
End Sub

Sub Format_Controls(Control_Name As String, CHT_Type As Integer, FLX_Heading_1 As String, FLX_Heading_2 As
»String, Optional FLX_Heading_3 As String)
   On Error GoTo Handler:
   '============Init Vars=========
   Dim FLX As MSFlexGrid
   Set FLX = FRMHost.Controls("FLX" & Control_Name)
   Dim CHT As MSChart
   Set CHT = FRMHost.Controls("CHT" & Control_Name)
   Dim iWidth As Double, i As Integer
   '==========Format CHT=========
   CHT.ChartType = CHT_Type
   CHT.Plot.Axis(VtChAxisIdX).AxisGrid.MajorPen.Style = VtPenStyleNull
   CHT.Plot.Axis(VtChAxisIdY).AxisGrid.MajorPen.Style = VtPenStyleNull
   CHT.Plot.Axis(VtChAxisIdY2).AxisGrid.MajorPen.Style = VtPenStyleNull
   '#Colours#
   CHT.Plot.Wall.Brush.FillColor.Set 255, 255, 255
   CHT.Plot.Wall.Brush.Style = VtBrushStyleSolid
   CHT.Plot.SeriesCollection(1).DataPoints(-1).Brush.FillColor.Set 0, 0, 255
   CHT.Plot.SeriesCollection(1).DataPoints(-1).EdgePen.VtColor.Set 0, 0, 0
   CHT.Plot.SeriesCollection(1).Pen.VtColor.Set 0, 0, 255
```

```
With CHT.Plot.Backdrop.Fill
   .Brush.FillColor.Set 255, 255, 255 '192, 192, 255
   .Style = VtFillStyleBrush
End With

CHT.Backdrop.Fill.Style = VtFillStyleBrush
CHT.Backdrop.Fill.Brush.FillColor.Set 255, 255, 255 '192, 192, 255


'#Lines & colours#
If CHT.Plot.SeriesCollection.Count = 2 Then

   CHT.Plot.SeriesCollection(2).Pen.VtColor.Set 255, 0, 255
ElseIf CHT.Plot.SeriesCollection.Count > 2 Then
   CHT.Plot.SeriesCollection(2).Pen.VtColor.Set 0, 0, 0
   CHT.Plot.SeriesCollection(3).Pen.VtColor.Set 0, 0, 0
   CHT.Plot.SeriesCollection(2).Pen.Style = VtPenStyleDotted
   CHT.Plot.SeriesCollection(3).Pen.Style = VtPenStyleDotted
End If

'#Sctterplots only
CHT.Plot.Axis(VtChAxisIdY).AxisScale.Hide = False

If Duplicate = True And Control_Name = "Raw" Then
   For i = 1 To CHT.Plot.SeriesCollection.Count
      CHT.Plot.SeriesCollection(i).ShowLine = False
      CHT.Plot.SeriesCollection(i).SeriesMarker.Show = True
      CHT.Plot.SeriesCollection(i).SeriesMarker.Auto = False

      CHT.Plot.SeriesCollection(i).Pen.Style = VtPenStyleSolid
      If i = 1 Then
         CHT.Plot.SeriesCollection(i).Pen.VtColor.Set 0, 0, 0
         CHT.Plot.SeriesCollection(i).DataPoints(-1).Marker.Style = VtMarkerStyleFilledCircle
      Else
         CHT.Plot.SeriesCollection(i).Pen.VtColor.Set 0, 0, 0
         CHT.Plot.SeriesCollection(i).DataPoints(-1).Marker.Style = VtMarkerStyleFilledSquare
      End If
      CHT.Plot.SeriesCollection(i).DataPoints(-1).Marker.Size = 120

   Next

Else
   For i = 1 To CHT.Plot.SeriesCollection.Count
      CHT.Plot.SeriesCollection(i).ShowLine = True
      CHT.Plot.SeriesCollection(i).SeriesMarker.Show = False
   Next

End If
'#Lines Only#
For i = 1 To CHT.Plot.SeriesCollection.Count
   CHT.Plot.SeriesCollection(i).Pen.Width = 1
Next i

CHT.Plot.UniformAxis = False

CHT.Plot.Axis(VtChAxisIdY2).AxisScale.Hide = True
CHT.AllowSeriesSelection = False

'-----Specialist CHTs------
If CHT.Name = "CHTSerial" Then
   '---Check if we are starting or finishing SAA---
   If FRMHost.CHKSerial_Single_Draw.Value = 1 Then
      CHT.AllowSelections = False
      CHT.Plot.Axis(VtChAxisIdY).AxisScale.Hide = False
   Else
      CHT.AllowSelections = True
      CHT.Plot.Axis(VtChAxisIdY).AxisScale.Hide = True
   End If
ElseIf CHT.Name = "CHTRaw" Then
   CHT.AllowSelections = True
Else
   CHT.AllowSelections = False
End If
```

```
   '=========Format FLX==========

   '-----Decide on 2 or 3 Cols---
   If FLX_Heading_3 <> "" Then
      FLX.Cols = 3
      FLX.Col = 2
      FLX.Row = 0
      '###Format extra col###
      FLX.Text = FLX_Heading_3
      iWidth = (FLX.Width / 3) - 40
      FLX.ColWidth(2) = iWidth
      FLX.ColAlignment(2) = 4
   Else
      iWidth = (FLX.Width / 2) - 40
      FLX.Cols = 2
   End If
   '~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   '------Generic Format--------
   With FLX
      .FixedCols = 0
      .ScrollBars = flexScrollBarVertical
      .Row = 0
      .Col = 0
      .Text = FLX_Heading_1
      .ColWidth(0) = iWidth
      .ColAlignment(0) = 1
      .Col = 1
      .Text = FLX_Heading_2
      .ColWidth(1) = iWidth
      .ColAlignment(1) = 4
      .BackColor = &HFFFFFF
      .BackColorBkg = &HFFC0C0
      .BackColorFixed = &HC00000
      .BackColorSel = &H8000000D
      .ForeColorFixed = &HFFC0C0
      .ForeColorSel = &HFFFFFF
      .GridColorFixed = &HFFC0C0
      .GridColor = &HFFFFFF
      .Gridlines = flexGridFlat
      .GridLinesFixed = flexGridFlat
      .Appearance = flexFlat
      .BorderStyle = flexBorderNone
   End With
   '~~~~~~~~~~~~~~~~~~~~~~~~~~~~


   '===========Clean Up============
   Set FLX = Nothing
   Set CHT = Nothing

   Exit Sub
   Handler:
   MsgBox ("VB is being crap need to restart proggy")
   End
End Sub

Sub Enable_Buttons()
   Dim cnt As Control
   On Error Resume Next
   For Each cnt In FRMHost.Controls

      If cnt.Container.Name = "FRAMenu" Then
         cnt.Enabled = True
      End If
   Next cnt

   '---Exceptions---
   If FRMHost.FLXGrouped.Rows < 3 Then
      If FRMHost.FLXGrouped.TextMatrix(2, 0) = "" Then
         FRMHost.chkMean.Enabled = False
         FRMHost.CHKGrpPowerPeriod.Enabled = False
         FRMHost.CHKGrpAutocorrelation.Enabled = False
      End If
   End If
End Sub
```

```
Sub Cht_Point_Select(ByRef CHT As MSChart, ByRef FLX As MSFlexGrid, Datapoint As Integer)
   Dim irow As Integer, sum As Double, Mean_of_Serial As Double, iTab As Integer, MPA_Found As Boolean,
»Cols_Needed As Integer, Deleted As Boolean

   '----Set Number of CHT cols-----
   If Duplicate = False Then
      Cols_Needed = 2
   Else
      Cols_Needed = 3
   End If

   '------Find out which CHT we Started From----
   For iTab = 1 To FRMHost.Tabmain.Tabs.Count
      If FRMHost.Tabmain.Tabs(iTab).Image = "Mpa" Then
         Cols_Needed = 2
      End If
   Next iTab
   '------Decide if we Need another Data Col----
   If Cols_Needed > CHT.ColumnCount Then
      '#Add Another Data Col#
      CHT.ColumnCount = CHT.ColumnCount + 1
      '#Insert Blank Data#
      CHT.Column = CHT.ColumnCount
      For irow = 1 To CHT.RowCount
         CHT.Row = irow
         CHT.data = 0
      Next irow
   End If

   '========Get Mean of DataSet=======
   For irow = 1 To CHT.RowCount
      sum = sum + CHT.ChartData(irow, 1)
   Next irow

   Mean_of_Serial = sum / (irow - 1)

   '====Mean = 0 Workaround=====
   If Mean_of_Serial > -0.5 And Mean_of_Serial < 0.5 Then
      Mean_of_Serial = 0.6
   End If


   '==========Add/Delete POI to CHT===========
   If Duplicate = True And (FRMHost.Tabmain.Tabs(1).Selected = True And FRMHost.Tabmain.Tabs(1).Image =
»"Raw") Then
      CHT.Row = Datapoint
      CHT.Column = 3
   Else

      CHT.Row = Datapoint
      CHT.Column = 2
   End If
   If CHT.data = 0 Then
      CHT.data = Mean_of_Serial
      Deleted = False
   Else
      CHT.data = 0
      Deleted = True
   End If
   CHT.Plot.SeriesCollection(1).Position.Order = 1
   CHT.Plot.SeriesCollection(2).Position.Order = 0


   '========Add/Delete POI to FLXserial=======
   If Deleted = False Then
      '------See if we need another Row----
      If FLX.TextMatrix(FLX.Rows - 1, 0) <> "" Then
         FLX.Rows = FLX.Rows + 1
      End If

      '-----------Insert Data--------------
      FLX.TextMatrix(FLX.Rows - 1, 0) = FRMHost.LBLInfo.Caption
      FLX.TextMatrix(FLX.Rows - 1, 1) = Datapoint
      If CHT.Name = "CHTSerial" Then
         FRMHost.CHKSerial_Single_Draw.Enabled = True
      Else
         FRMHost.chkRaw_Resample.Visible = True
```

```
            'FRMHost.chkSave.Visible = True
         End If
      Else
         For irow = 1 To FLX.Rows
            If FLX.TextMatrix(irow, 0) = FRMHost.LBLInfo.Caption And _
                  FLX.TextMatrix(irow, 1) = Datapoint Then
               If FLX.Rows <> 2 Then
                  FLX.RemoveItem (irow)
               Else
                  FLX.TextMatrix(irow, 0) = ""
                  FLX.TextMatrix(irow, 1) = ""
                  If CHT.Name = "CHTSerial" Then
                     FRMHost.CHKSerial_Single_Draw.Enabled = False
                  Else
                     FRMHost.chkRaw_Resample.Visible = True
                     'FRMHost.chkSave.Visible = True
                  End If
               End If
               Exit For
            End If
         Next irow
      End If

End Sub

Sub Add_on_POI(ByRef CHT As MSChart, ByRef FLX As MSFlexGrid)
   '------Init Vars--------
   Dim irow As Integer
   Dim myName As String, iData_Count As Integer, Mean_of_Serial As Double
   Dim iPOI As Integer, sum As Double, idata As Long
   ReDim arrPOI(0) As Integer


   'Set CHT = FRMHost.Controls("CHTSerial")
   'Set FLX = FRMHost.Controls("FLXserial")
   myName = FRMHost.LBLInfo

   '---Escape Clause---
   If FLX.TextMatrix(1, 0) = "" Then Exit Sub

   '-------Get POI---------
   For irow = 1 To FLX.Rows - 1
      If myName = FLX.TextMatrix(irow, 0) Then
         '---Inc Arr if needed---
         If iData_Count > UBound(arrPOI) Then
            ReDim Preserve arrPOI(iData_Count) As Integer
         End If

         '----Add in Field-----
         arrPOI(iData_Count) = FLX.TextMatrix(irow, 1)
         iData_Count = iData_Count + 1
      End If
   Next irow
   '------Escape Clause----
   If arrPOI(0) = 0 Then Exit Sub

   '------Extend Graph-----
   '---Get Mean----
   For irow = 1 To CHT.RowCount
      sum = sum + CHT.ChartData(irow, 1)
   Next irow

   Mean_of_Serial = sum / (irow - 1)

   '---Add Column for data---
   CHT.ColumnCount = CHT.ColumnCount + 1
   CHT.Column = CHT.ColumnCount

   '------Add in POI-------
   For irow = 1 To CHT.RowCount
      CHT.Row = irow

      '---Mean Data---
      If irow = arrPOI(idata) Then
         CHT.data = Mean_of_Serial
         idata = idata + 1

         '---Workaround Fix---
```

```
            If idata > UBound(arrPOI) Then
               idata = idata - 1
            End If

            '---Blank Data---
         Else
            CHT.data = 0
         End If
      Next irow



      '-------Clean Up--------
      Erase arrPOI

End Sub
```

```
Option Explicit
'The Module Does the Following for either Text or Excel _
The Text Path _
1) Loads the data (load_Data_text) _
2) Puts it into 1 or 2 Data Arrays (load & delimit) _
3) Checks DataSets Length (load) _
4) Verifies Datas Format (verify_data) _
5) Splits it up (Split_up_Patients) _
6) Fills in the -999 missing data (Missing_Data) _
7) Stores the Array Globally _
8) Displays it on FLX _
 _
The Excel Path _
1) Loads the Data (Load_Data_Excel) _
2) Delimits it into one or Two Arrays _
3) Does steps 3 to 8 of The Text Path _
 _
We also have the Update_data function which takes in modified data from FLXRaw



Public Sub Load_Data_text()
   'On Error GoTo Error_Handler
   '===============Init Vars=============
   Dim CurrFile As String
   Dim strTemp As String
   Dim Delimiter As String
   Dim LineCounter As Long
   Dim FileEnd As Long
   Dim ErrCounter As Integer
   Dim arrDataLine() As String
   Dim chkArr1 As Boolean, chkArr2 As Boolean
   LineCounter = 0
   CurrFile = FRMchoose.CDBmain.FileName
   '===============Open File==============
   If CurrFile <> "" Then
      Open CurrFile For Input As #1
      FRMHost.Caption = "TSA / " & FRMchoose.CDBmain.FileTitle
   Else
      Exit Sub
   End If


   '============Investigate File============
   '#Find delimiter#
   Delimiter = Delimit_Data()
   If Delimiter <> "" Then
      Duplicate = True
   Else
      Duplicate = False
   End If
   '#Find LengthofFile#
   Open CurrFile For Input As #2

   Do While EOF(2) = False
      Input #2, strTemp
      FileEnd = FileEnd + 1
   Loop

   Close #2

   '=========Set up Progress Bar==============
   frmWait.Bar.max = FileEnd

   frmWait.Show , FRMHost
   '=============Init Data Arrays=============
   FileEnd = FileEnd - 1
   ReDim arrDataSet1(FileEnd) As String
   ReDim arrDataSet2(FileEnd) As String

   '=================Load data==============


   Do Until (EOF(1) = True)
      DoEvents
      '---------------Read Data-----------
```

```
      Input #1, strTemp

      '---------Check & Insert Data---------
      If strTemp <> "" Then

         arrDataLine() = Split(strTemp, Delimiter) 'split the string using the delimiter
         arrDataSet1(LineCounter) = TrimALL(arrDataLine(0), True)

         If UBound(arrDataLine) <> 0 Then
            '##Insert main data if dupe is blank##
            If arrDataLine(1) = "" Then
               arrDataSet2(LineCounter) = TrimALL(arrDataLine(0), True)
            Else
               arrDataSet2(LineCounter) = TrimALL(arrDataLine(1), True)
            End If
            '########################################
         Else
            If Duplicate = True Then
               arrDataSet2(LineCounter) = TrimALL(arrDataLine(0), True)

            End If
         End If

      Else
         '--------If we get an error-------
         MsgBox ("We have reached an empty line (#" & LineCounter + 1 & ") before we have got to the end
»of the file " & vbCrLf & "Please make sure there is no extra whitspace after or in the data" &
»vbCrLf & "Attempting to read next line. Try " & ErrCounter + 1 & " of 2")
         If ErrCounter = 3 Then
            Close 1
            MsgBox ("Failed to read all the data, please check it then try again")
            Error_in_data = True
            Unload frmWait
            Exit Sub
         End If
         ErrCounter = ErrCounter + 1
         '~~~~~~~~~~~~~~~~~~~~~~~~~
      End If

      '======Update Progress Bar========
      frmWait.Bar.Value = LineCounter

      LineCounter = LineCounter + 1



   Loop

   Unload frmWait
   Close 1

   '===========Verify Data is in correct Format=============
   chkArr1 = Verify_Data(arrDataSet1())

   If chkArr1 = False Then Exit Sub

   If Duplicate = True Then

      chkArr2 = Verify_Data(arrDataSet2())
   Else
      chkArr2 = True
   End If

   If chkArr1 = True And chkArr2 = True Then
      Error_in_data = False
      Call Split_up_Patients(arrDataSet1(), arrDataSet2())

   Else
      Error_in_data = True
   End If
   '=========================================================
   Exit Sub
Error_Handler:
   'Exit Sub

End Sub
```

```
Private Function Delimit_Data() As String
   '============Init Vars===========
   Dim CurrFile As String
   Dim strtoCheck As String
   Dim tmpDelim As String
   Dim DelimitertoCheck(3) As String
   Dim i As Integer
   CurrFile = FRMchoose.CDBmain.FileName
   Delimit_Data = ""
   DelimitertoCheck(0) = " "
   DelimitertoCheck(1) = vbTab
   DelimitertoCheck(2) = ","
   DelimitertoCheck(3) = "|"
   '===========Input File=========
   Open CurrFile For Input As #2

   Input #2, strtoCheck
   Input #2, strtoCheck
   Input #2, strtoCheck
   Input #2, strtoCheck 'does this four times to get the forth data item which is the data

   Close #2

   '====Search for Common delimiters====
   For i = 0 To UBound(DelimitertoCheck)
      If InStr(1, strtoCheck, DelimitertoCheck(i)) <> 0 Then
         If InStr(1, strtoCheck, DelimitertoCheck(i)) <> Len(strtoCheck) Then
            Delimit_Data = DelimitertoCheck(i)
            Exit Function
         End If
      End If
   Next i

   '===================================


End Function

Private Function Verify_Data(ByRef Array2Check() As String) As Boolean
   '===================Init Vars=======================
   Dim NoofVals As Long
   Dim Counter As Long
   Dim CurrName As String
   Dim i As Long, Xtra As Long
   '============Check Data Format is Correct=========
   For i = 0 To UBound(Array2Check) Step 3 'CHECK
      Counter = 0

      '------assign First Patient name------
      If IsNumeric(Array2Check(i)) = False Then
         CurrName = Array2Check(i)
         If UCase(CurrName) = "RUIZ1P52/707" Then
            Debug.Print ""
         End If
      End If
      '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      '-----checks to see if current data item is numeric and the item before it _
    is numeric so that we can get the assays 'cos the formatting of the data should _
    be name,assay,no. of vals, vals ------------------------------------------

      '---------Check Name, Assay---------- 'PROBLEM HERE WITH shorter number given but more data there
      If UBound(Array2Check) = i Then
         MsgBox ("Not enough Data points explicitly specified in header of file")
         Verify_Data = False
         Exit Function
         'Fixes above
      End If

      If i = 862 Then
         Debug.Print
      End If
      If IsNumeric(Array2Check(i)) = False And _
            IsNumeric(Array2Check(i + 1)) = False Then

         '######Check No. of Vals#######
         If IsNumeric(Array2Check(i + 2)) Then
```

```
            NoofVals = Array2Check(i + 2)
            Verify_Data = True
         Else
            MsgBox "Error in Range Specified for " & CurrName
            Verify_Data = False
            Exit Function
         End If
         '##############################


         Counter = 0
         '#######Check all Vals are present######
         Do While Counter < NoofVals
            '-------Special Check  for last PAt---------
            If (i + 3 + Counter) > UBound(Array2Check) Then
               MsgBox "Not Enough Datapoints, the amount specified for " & CurrName _
                     & " was " & Array2Check(i + 2) & " but only " & Counter & " points found" & " on line
»(#" & i & ")"
               Verify_Data = False
               Exit Function
            End If

            '-------Check for rest-----------
            If IsNumeric(Array2Check(i + 3 + Counter)) Then
               Counter = Counter + 1
               Verify_Data = True
            Else
               Call MsgBox("Not Enough Datapoints, the amount specified for " & CurrName _
                     & " was " & Array2Check(i + 2) & " but only " & Counter & " points found. The next
»line is non numeric" & " on line (#" & i & ")" & vbCrLf & "OR no delimiter found in
»one val for this subject")
               Verify_Data = False
               Exit Function
            End If
         Loop
         '##########################################
         i = i + NoofVals

      Else
         '-----------------Too much Data found-----------------

         '##########Find end of data#############
         Do While IsNumeric(Array2Check(i + Xtra)) = True
            Xtra = Xtra + 1
            If i + Xtra > UBound(Array2Check) Then
               Exit Do
            End If
         Loop
         '#######################################

         '############Display Error Message##############
         MsgBox ("Error in Patient data, expected next patient Name OR Assay and got " _
               & Array2Check(i) & "." & VBA.vbCr & "There are " & Xtra & " extra datapoints for patient " _
               & CurrName & ", making the total " & _
               (NoofVals + Xtra) & " and the number specfied was only " & NoofVals & " on line (#" & i &
»")" & vbCrLf & "Please fix the data then try again")
         '##############################################
         Verify_Data = False
         Exit Function

      End If

   Next i

   Verify_Data = True

End Function

Private Sub Split_up_Patients(arrMain() As String, arrDupe() As String)

   '=================Init Vars=========================
   If Duplicate = False Then
      ReDim arrPats(Max_Data_Items, Max_Patients) As String
   Else
      ReDim arrPats(Max_Data_Items, (Max_Patients * 2) + 1) As String
   End If
```

```
Dim idata As Long, iPat As Integer, i As Long, j As Long

'========Initialise Status Bar==========
frmWait.Show , FRMHost
frmWait.Bar.Value = 0
frmWait.LBLInfo = "Adding please wait ..."
If Duplicate = True Then
   frmWait.Bar.max = (UBound(arrMain) * 2) + 1
Else
   frmWait.Bar.max = UBound(arrMain) + 1
End If
'===========Split Main Data into Individual Patients============
For i = 1 To UBound(arrMain)
   DoEvents
   '#Update status Bar#
   frmWait.Bar.Value = i
   '####################

   '---escape clause---
   If iPat > UBound(arrPats, 2) Then MsgBox ("Please Increase Max Number of Subjects Allowed"): End
   If IsNumeric(arrMain(i)) = False And IsNumeric(arrMain(i - 1)) = False Then
      '####Assign Name, Assay & Vals ######
      arrPats(idata, iPat) = TrimALL(arrMain(i - 1))
      arrPats(idata + 1, iPat) = TrimALL(arrMain(i))
      arrPats(idata + 2, iPat) = TrimALL(arrMain(i + 1))
      '####################################

      '#######Assign Vals###########
      For j = 1 To arrPats(idata + 2, iPat)
         '----escape clause----
         If (idata + 2 + j) > Max_Data_Items Then MsgBox ("Please Increase Max Data Items Allowed"): End

         arrPats(idata + 2 + j, iPat) = TrimALL(arrMain(i + 1 + j))
      Next j
      '#################################
      iPat = iPat + 1

   End If
Next i

'===========Split Duplicate Data into Individual Patients============
If Duplicate = True Then

   For i = 0 To UBound(arrDupe)

      DoEvents
      '#Update status Bar#
      frmWait.Bar.Value = frmWait.Bar.Value + 1
      '####################

      If IsNumeric(arrDupe(i)) = False Then
         If i <> 0 Then
            If IsNumeric(arrDupe(i - 1)) = False Then
               arrPats(idata, iPat) = TrimALL(arrDupe(i - 1)) 'assign name

               arrPats(idata + 1, iPat) = TrimALL(arrDupe(i)) 'assign assay
               arrPats(idata + 2, iPat) = TrimALL(arrDupe(i + 1)) 'assign no of vals

               For j = 1 To arrPats(idata + 2, iPat)
                  arrPats(idata + 2 + j, iPat) = TrimALL(arrDupe(i + 1 + j))
               Next j 'assign all data vals

               iPat = iPat + 1
            End If
         End If
      End If


   Next i
End If
'----set global var of Noof Pats---
Dupe_data_Start = (iPat / 2) '- 1
'================================================================
Unload frmWait

If Missing_Data(arrPats()) = True Then
   Exit Sub
End If
```

```
End Sub


Public Function Missing_Data(arrPats() As String) As Boolean

   '==================Init Vars=====================
   Dim limit As Long
   Dim Start_of_Dupe As Long, Dupe_data_Start As Long
   Dim iPat As Long, idata As Long

   '==================Find last Pat=================
   For iPat = 0 To UBound(arrPats, 2)
      If arrPats(0, iPat) = "" Then Exit For
   Next iPat

   limit = (iPat - 1)
   Start_of_Dupe = (limit + 1) / 2
   Dupe_data_Start = Start_of_Dupe 'Set global flag
   iPat = 0
   '======================Find -999 Data points============
   Do While iPat <= limit
      For idata = 3 To Max_Data_Items
         '#Find -999#
         If arrPats(idata, iPat) = "-999" Then
            '#if duplicate data available use that#
            If Duplicate = True Then
               '#decide to copy from real or duplicate#
               If iPat >= Start_of_Dupe Then
                  '-----------the duplicate pats---------
                  If arrPats(idata, iPat - Start_of_Dupe) <> "-999" Then

                     If Right(arrPats(idata, iPat - Start_of_Dupe), 1) <> "*" Then
                        arrPats(idata, iPat) = arrPats(idata, iPat - Start_of_Dupe) & "*"       '[Copy
»Data]
                     Else
                        GoTo Int_Data:
                     End If

                  Else
Int_Data:
                     '#if duplicate is also -999 and it is first data item#
                     If idata = 3 Then
                        Missing_Data = True
                        MsgBox ("There is an error in " & arrPats(0, iPat) & " Dataset. There is a missing
»value at the Start")
                        End
                     ElseIf idata = (arrPats(2, iPat) + 2) Then
                        '#Last Data Item for this Patient#
                        Missing_Data = True
                        MsgBox ("There is an error in " & arrPats(0, iPat) & " Dataset. There is a missing
»value at the End")
                        End
                     Else
                        arrPats() = Interpolate_data(idata - 1, arrPats(), iPat)   '[Copy Data]
                     End If
                  End If
                  '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
               Else
                  '----------the main pats------------
                  If arrPats(idata, iPat + Start_of_Dupe) <> "-999" Then

                     If Right(arrPats(idata, iPat + Start_of_Dupe), 1) <> "*" Then
                        arrPats(idata, iPat) = arrPats(idata, iPat + Start_of_Dupe)         '[Copy Data]
                     Else
                        GoTo Int_Data2:
                     End If
                  Else
Int_Data2:
                     '#if duplicate is also -999 and it is first data item#
                     If idata = 3 Then
                        Missing_Data = True
                        MsgBox ("There is an error in " & arrPats(0, iPat) & " Dataset. There is a missing
»value at the Start")
                        End
```

```
                    ElseIf idata = (arrPats(2, iPat) + 2) Then
                       '#Last Data Item for this Patient#
                       Missing_Data = True
                       MsgBox ("There is an error in " & arrPats(0, iPat) & " Dataset. There is a missing
»value at the End")
                       End
                    Else
                       arrPats() = Interpolate_data(idata - 1, arrPats(), iPat) '[Copy Data]

                    End If
                 End If
                 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
              End If
           Else

              If idata = 3 Then
                 '#First Data Item for this Patient#
                 Missing_Data = True
                 MsgBox ("There is an error in " & arrPats(0, iPat) & " Dataset. There is a missing value
»at the Start")
                 End
              ElseIf idata = (arrPats(2, iPat) + 2) Then
                 '#Last Data Item for this Patient#
                 Missing_Data = True
                 MsgBox ("There is an error in " & arrPats(0, iPat) & " Dataset. There is a missing value
»at the End")
                 End
              Else
                 arrPats() = Interpolate_data(idata - 1, arrPats(), iPat) '[Copy Data]



              End If


           End If
        End If
     Next idata
     iPat = iPat + 1
  Loop

  '---Check for duplicate Unique Identifiers---
  Dim i As Integer, j As Integer
  If Duplicate = False Then 'Will need to fix for future release
     For i = 0 To UBound(arrPats, 2)
        For j = 0 To UBound(arrPats, 2)
           '-Escape clause-
           If arrPats(0, i) = "" Then Exit For

           If UCase(arrPats(0, i)) = UCase(arrPats(0, j)) And i <> j Then
              MsgBox ("Each Dataset must have a unique identifier there are two or more with the
»identifier " & arrPats(0, i) & vbCrLf & "Please fix and try again")
              End
           End If
        Next j
     Next i
  End If


  Call Store_data(arrPats(), limit)
End Function

Function Interpolate_data(ByRef Position_A As Long, ByRef arrPats() As String, ByRef iPat As Long) As
»Variant
  On Error GoTo Handler:
  '=============Init Vars===================
  Dim Expand As Long, Position_B As Long, Gap_Size As Integer, Counter As Integer
  Dim DataPoint_B As Double, DataPoint_A As Double, Gap_Value As Double, Interval As Double

  '============Error Checker================
  If Right(arrPats(Position_A, iPat), 1) = "*" Then
     DataPoint_A = Left(arrPats(Position_A, iPat), Len(arrPats(Position_A, iPat)) - 1)
  Else
     DataPoint_A = arrPats(Position_A, iPat)
  End If
```

```
   Expand = 1
   Counter = 1
   '=======Find Next Real Data Point=========
   Do While arrPats(Position_A + Expand, iPat) = "-999"
      Expand = Expand + 1
   Loop

   '===========Init DataPoint============
   Position_B = Position_A + Expand
   DataPoint_B = CDbl(arrPats(Position_B, iPat))

   '========Workout Interpolated Data=====
   Gap_Size = Position_B - Position_A
   Gap_Value = DataPoint_B - DataPoint_A
   Interval = Gap_Value / Gap_Size

   '=====Fill in Interpolated Data=======
   Do While Position_A + Counter <= Position_B - 1
      arrPats(Position_A + Counter, iPat) = _
            CStr(CDbl(DataPoint_A) + (Interval * Counter)) & "*"
      Counter = Counter + 1
   Loop

   Interpolate_data = arrPats
   Exit Function
   Handler:
   MsgBox ("Too many -999 found at end of dataset for " & arrPats(0, iPat) & "Please fix this and try
»again.")
   End
End Function

Sub Store_data(ByRef arrPats() As String, ByRef Pat_Count)

   '==============Init Vars=================
   Dim iPat As Integer, idata As Long

   '=============Store Data================
   For iPat = 0 To Pat_Count
      idata = 0
      Do While arrPats(idata, iPat) <> ""
         arrRaw(idata, iPat) = arrPats(idata, iPat)
         idata = idata + 1
         '        If iPat = 42 Then
         '        If iData = 120 Then
         ''        MsgBox ("")
         '        End If
         '        End If
      Loop
   Next iPat


End Sub

Sub Display_Data()

   '========Init Vars===========
   Dim FLX As MSFlexGrid
   Dim irow As Integer, iPat As Integer, icol As Integer, apat As Integer, Endpoint As Integer
   Dim strDupe As String


   Set FLX = FRMHost.FLXMain

   '=========Find Last Patient=========
   For iPat = 0 To UBound(arrRaw, 2)
      If arrRaw(0, iPat) = "" Then Exit For
   Next iPat


   '=======Decide how much or arrRaw to use=====
   If Duplicate = False Then
      strDupe = "No"
      Endpoint = iPat - 1 'use all of the array
   Else
      strDupe = "Yes"
      Endpoint = (iPat / 2) - 1 'use half of the array
   End If
```

```
   irow = 1
   '=============Display Data===============
   For apat = 0 To Endpoint
      '----Insert data-----
      For icol = 0 To FLX.Cols - 2
         FLX.TextMatrix(irow, icol) = arrRaw(icol, apat)
      Next icol
      '--------------------
      '----Insert Duplicate Y/N col----
      FLX.TextMatrix(irow, icol) = strDupe
      '--------------------------------
      irow = irow + 1

      '-----Extend FLX if needs be----
      If FLX.Rows = irow Then
         FLX.Rows = FLX.Rows + 1
      End If
      '-------------------------------
   Next apat

   '==========Clean Up==============
   Set FLX = Nothing
End Sub

Sub Update_Data()
   '================Init Vars===============
   Dim FLX As MSFlexGrid
   Dim FRM As Form
   Set FLX = FRMHost.FLXRaw
   Set FRM = FRMHost
   Dim currPat As Integer, CurrEnd As Integer, idata As Long
   FLX.Col = 1
   '==============Get patient Number==========
   currPat = FRM.FLXMain.Row

   '========Get last data item===============
   CurrEnd = Find_Last_DataItem(currPat)

   '============Copy FLX data to Array=======
   For idata = 3 To CurrEnd
      FLX.Row = (idata - 2)
      arrRaw(idata, currPat) = FLX.Text
   Next idata

   '------If Duplicate copy duplicate data also---
   If Duplicate = True Then
      FLX.Col = 2
      currPat = currPat + Dupe_data_Start
      For idata = 3 To CurrEnd
         FLX.Row = (idata - 2)
         arrRaw(idata, currPat) = FLX.Text
      Next idata

   End If
   '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
End Sub

Sub Load_Data_Excel()
   '=============Init Vars============
   '----------Excel---------
   Dim mySheet As excel.Worksheet
   Dim myBook As excel.Workbook

   excel.Workbooks.Open (FRMchoose.CDBmain.FileName)
   Set myBook = excel.ActiveWorkbook
   Set mySheet = myBook.ActiveSheet

   '--------Others-----------
   Dim irow As Long, icol As Integer, iData_Counter As Integer, iDatacol As Integer, imax As Long
   Dim lEnd_of_Row As Long
   icol = 1: irow = 1
   Dim arrData() As String

   '=============Find Number of Data Cols==============
   Do While mySheet.Cells(1, icol) <> ""
      If IsNumeric(mySheet.Cells(1, icol)) = False Then
         iData_Counter = iData_Counter + 1
```

```
      End If
      icol = icol + 1
   Loop

   '===========Set up Waiting Screen================
   imax = ((iData_Counter - 1) * 65336)
   frmWait.Bar.max = imax
   frmWait.Show , FRMHost

   '===========Find Data for each Datacol=============
   For iDatacol = 1 To iData_Counter
      '#update Progress label#
      frmWait.LBLInfo = "Processing " & iDatacol & "/" & iData_Counter

      '#Get Data into an Array#
      Do While mySheet.Cells(irow, iDatacol) <> ""
         DoEvents
         '----Read in Data---
         '#Define Array#
         If irow + lEnd_of_Row = 1 Then
            ReDim Preserve arrData(0)
         ElseIf (irow + lEnd_of_Row - 1) > UBound(arrData) Then
            ReDim Preserve arrData(irow + lEnd_of_Row - 1)
         End If

         '#Insert Data#
         arrData(irow + lEnd_of_Row - 1) = mySheet.Cells(irow, iDatacol)

         '---Replace Blanks with -777-----


         '---------Next Row----------
         irow = irow + 1

         '-----Update Status Bar----------
         If irow > frmWait.Bar.max Then
            frmWait.Bar.max = frmWait.Bar.max + 5000
         End If
         frmWait.Bar.Value = irow
      Loop
      '#Store Endpoint#
      lEnd_of_Row = irow - 1

      '#Reset Progress Bar#
      frmWait.Bar.Value = 0
      frmWait.Bar.max = imax
      irow = 1
   Next iDatacol

   Unload frmWait
   '======Is Data in Duplicate========



   '===========Verifys Data==========

   '===========Clean Up=============

End Sub

Sub Save_Data()
   '----Init Vars-----
   Dim CDB As CommonDialog, Result As Integer
   Dim arrTmp() As String, last_pat As Integer
   Dim arrTMP2() As String
   Dim irow As Integer, iPat As Integer

   arrTmp() = Split(FRMHost.Caption, "/")
   arrTMP2() = Split(arrTmp(1), ".")
   Set CDB = FRMchoose.CDBmain
   '----Set Up Document----
   CDB.CancelError = True
   CDB.FileName = Trim(arrTMP2(0)) & "_" & Replace(Date, "/", "-") & ".txt"
   CDB.Filter = "*.txt (Text)|*.txt"
   On Error GoTo ending
   CDB.ShowSave

   '---Set up for duplicate data---
```

```
   If Duplicate = True Then
      last_pat = Dupe_data_Start - 1
   Else
      last_pat = UBound(arrRaw)
   End If

   '---Copy all Data from Array---

   Open CDB.FileName For Output As #1 'Opens The File
   For iPat = 0 To last_pat

      If arrRaw(irow, iPat) = "" Then
         Exit For
      End If

      For irow = 0 To UBound(arrRaw, 2)

         If arrRaw(irow, iPat) <> "" Then
            If Duplicate = True Then
               If irow > 2 Then
                  Print #1, IIf(Right(arrRaw(irow, iPat), 1) = "*", -999, arrRaw(irow, iPat)) & vbTab & IIf
»(Right(arrRaw(irow, iPat + Dupe_data_Start), 1) = "*", -999, arrRaw(irow, iPat +
»Dupe_data_Start))

               Else
                  Print #1, arrRaw(irow, iPat)
               End If
            Else
               Print #1, IIf(Right(arrRaw(irow, iPat), 1) = "*", -999, arrRaw(irow, iPat))
            End If
         Else
            irow = 0
            Exit For
         End If

      Next irow
   Next iPat

   Close #1
   MsgBox ("File has been Saved")
   FRMHost.chkSave.Value = 0
   FRMHost.chkSave.Visible = False
   '---Close Document----
   Exit Sub
ending:
End Sub
```

```
Option Explicit
'Maximum Array size is about 130,000,000 we are going for 100,000,000 _


Global Const Max_Data_Items = 1999 '82550 '
Global Const Max_Patients = 399 '1
Global Duplicate As Boolean                 'A global flag set if data is in duplicate
Global arrRaw(Max_Data_Items, Max_Patients) As String
Global Dupe_data_Start As Integer           'Where the duplicate dataset begins within arrRaw
Global Time_Val As Integer
Global Time_Type As String
Global x_Point_MPA As Integer               'number of points in moving point average
Global Error_in_data As Boolean             ' if true then stop accessing data
Global Huge_data As Boolean                 'If true then we are doing a image
'=====Colours=====
Global Const Blank = 0
Global Const Highlight = &HFFC0C0

'======Half Lifes=====
Global Const Insulin_Halflife = 3.8
Global Const CPeptide_Halflife = 28

Global SWAPPING_FLX As Boolean ' USed in FLX swapping Sub
Global GRP_POWER_ON As Boolean 'USED TO SHOW we are in middle of grp power/period

Public Sub Init_Global_Vars(ByRef Big As Boolean)

   'If Big = True Then
   '    Global arrRaw(Max_Data_Items_Lg, Max_Patients_Lg) As String
   '    Max_Patients = Max_Patients_Lg
   '    Max_Data_Items = Max_Data_Items_Lg
   'Else
   '    Global arrRaw(Max_Data_Items_Sm, Max_Patients_Sm) As String
   '    Max_Patients = Max_Patients_Sm
   '    Max_Data_Items = Max_Data_Items_Sm
   'End If


End Sub
```

```
Option Explicit

Sub Resize()

    '===============Init Vars================
    Dim FRM As Form, FLX As MSFlexGrid, ipos As Double, icol As Integer, tmpWidth As Double
    Dim cnt As Control, tmpname As String
    Set FRM = FRMHost
    Set FLX = FRM.FLXMain
    ipos = 0.01

    '====================Menu Start===================
    FRM.FRAMenu.Move _
          0, _
          0, _
          2400, _
          FRM.ScaleHeight '- (FRM.ScaleHeight * ipos * 2)
    'L
    'T
    'W
    'H


    '-----------------Template------------------
    FRM.arrShape(0).Move _
          30, _
          0, _
          FRM.FRAMenu.Width - 30, _
          500


    FRM.arrShape(1).Move _
          50, _
          FRM.arrShape(0).Height / 2, _
          FRM.FRAMenu.Width - 60, _
          FRM.FRAMenu.Height - 250


    FRM.LBLMenu.Move _
          90, _
          50, _
          FRM.arrShape(0).Width - 120, _
          180

    '-----------------------------------------------

    '--------------------Buttons-------------------
    FRM.cmdLoad.Move _
          190, _
          FRM.FRAMenu.Height - 430

    FRM.cmdExport.Move _
          190, _
          FRM.cmdLoad.top - 480

    FRM.cmdPrint.Move _
          190, _
          FRM.cmdExport.top - 480

    '#Xtra Menu Buttons#
    FRM.CHKABS_Norm_Fourier.Move _
          190, _
          FRM.cmdPrint.top - 480

    FRM.CHKSerial_Single_Draw.Move _
          190, _
          FRM.cmdPrint.top - 480

    FRM.cmdFourier_Abs_Norm.Move _
          190, _
          FRM.cmdPrint.top - 480

    FRM.chkRaw_Resample.Move _
          190, _
          FRM.cmdPrint.top - 960

    FRM.CHKTrend_Line.Move _
```

```
      190, _
      FRM.cmdPrint.top - 480

FRM.chkSave.Move _
      190, _
      FRM.cmdPrint.top - 1140
'-----------------------------------------------


'==================Data Start=====================
FRM.FRAData.Move _
      FRM.FRAMenu.Left + FRM.FRAMenu.Width + 50, _
      FRM.FRAMenu.top, _
      FRM.ScaleWidth - (FRM.FRAMenu.Left + FRM.FRAMenu.Width + 100), _
      (FRM.ScaleHeight - 200) * 0.4

'-----------------Template-----------------
FRM.arrShape(3).Move _
      30, _
      0, _
      1600, _
      500


FRM.arrShape(2).Move _
      50, _
      FRM.arrShape(0).Height / 2, _
      FRM.FRAData.Width - 50, _
      FRM.FRAData.Height - 300


FRM.LBLMenu.Move _
      90, _
      50, _
      FRM.arrShape(3).Width - 120, _
      180

'-----------------------------------------------


'--------------------FLX---------------------
FLX.Move _
      100, _
      FRM.arrShape(0).Height / 2 + 100, _
      FRM.FRAData.Width - 200, _
      FRM.FRAData.Height - 430

'#Resize FLX cols#
tmpWidth = FLX.Width / 4
For icol = 0 To FLX.Cols - 1
   FLX.ColWidth(icol) = tmpWidth
Next icol
'-----------------------------------------------

'==================Graphs Start=====================
FRM.FRAGraphs.Move _
      FRM.FRAMenu.Left + FRM.FRAMenu.Width + 50, _
      FRM.FRAData.top + FRM.FRAData.Height + 50, _
      FRM.ScaleWidth - (FRM.FRAMenu.Left + FRM.FRAMenu.Width + 20), _
      (FRM.ScaleHeight) * 0.61
FRM.Tabmain.Move _
      30, _
      0, _
      FRM.FRAGraphs.Width - 30, _
      FRM.FRAGraphs.Height

'-----------------Template-----------------
FRM.FRAGraph_Inner.Move _
      FRMHost.Tabmain.ClientLeft, _
      FRMHost.Tabmain.ClientTop, _
      FRMHost.Tabmain.ClientWidth, _
      FRMHost.Tabmain.ClientHeight

FRM.arrShape(4).Move _
      20, _
      20, _
      FRM.FRAGraph_Inner.Width - 40, _
      FRM.FRAGraph_Inner.Height - 40
```

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

   '--------------------FRALBLINFO---------------
   Call FRM.FRALableInfo.Move(FRM.FRAData.Left + FRM.arrShape(3).Width + 20, FRM.FRAData.top,
»FRM.FRAData.Width - FRM.arrShape(3).Width - 20, 250)
   FRM.LBLInfo.Move 0, 0, FRM.FRALableInfo.Width, FRM.FRALableInfo.Height
   '---------Resize all CHT's & FLX's-------
   Dim FLXTemp As MSFlexGrid, iWidth As Double

   For Each cnt In FRM.Controls
      If UCase(Left(cnt.Name, 3)) = "CHT" Then
         tmpname = Right(cnt.Name, Len(cnt.Name) - 3)
         '#The one exception is the Serial FLX on Start up as it doesn't exist#
         If FRM.CHKSerialArrayAveraging.Value = 0 And tmpname = "Serial" Then

            'DO nothing at all
         Else '#All other Controls are Fine#

            Call PositionControls(tmpname)
            '---Resize FLx's Cols-----
            Set FLXTemp = FRM.Controls("FLX" & tmpname)

            iWidth = (FLXTemp.Width / FLXTemp.Cols) - 40

            For icol = 0 To FLXTemp.Cols - 1
               FLXTemp.ColWidth(icol) = iWidth
            Next icol
         End If
      End If
   Next cnt
   '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   '================Clean Up================
Clean_UP:
   Set FLX = Nothing
   Set FRM = Nothing
End Sub
```

```
Option Explicit
'This Module is the Print and Exporting Module it relies heavily on Excel. _
1) Sets up an Excel Sheet (Print_Export_Data) _
2) Copies over all the data (Print_Export_Data) _
3) Draws a Graph using the Data (Print_Export_Graph) _
4) Either Prints it or Displays it (Print_Export_Data) _


Sub Print_Out(ByRef Print_This As Boolean, Optional Dont_Show_This As Boolean)
   Call Print_Export_Data
   Call Print_Export_Graph(Print_This, Dont_Show_This)
End Sub

Sub Print_Export_Data()
   '================Init Vars==============
   Dim CurrName As String, FLX As MSFlexGrid, iSheet As Integer, iDel As Integer, strTitle As String
   Dim icol As Integer, irow As Integer, strProgram_Coords As String, CHT As MSChart, FRM As Form
   Dim Mpa As String, Det As String, i As Integer
   CurrName = What_Tab & " for " & FRMHost.LBLInfo
   Dim lcol As Long, lrow As Long

   Set FLX = FRMHost.Controls("FLX" & What_Tab)
   Set CHT = FRMHost.Controls("CHT" & What_Tab)
   Set FRM = FRMHost
   '========Set Up Excel Worksheet=========
   excel.Workbooks.Add
   Sheets(1).Name = "Data"

   excel.Application.DisplayAlerts = False

   For iSheet = 2 To Sheets.Count - iDel
      Sheets(iSheet - iDel).Delete
      iDel = iDel + 1
   Next iSheet

   excel.Application.DisplayAlerts = True

   Sheets(1).Select

   excel.Range("A1:G1").Select
   Selection.Merge

   '-----Build Title---
   If FRMHost.Tabmain.SelectedItem.Key = "Grouped_Power" Or _
         FRMHost.Tabmain.SelectedItem.Key = "Grouped_Autocorrelation" _
         Then
      strTitle = What_Tab
   Else
      strTitle = CurrName
   End If

   For i = 0 To FRMHost.OptMpa.Count - 1
      If FRMHost.OptMpa.Item(i).Value = True Then
         Mpa = FRMHost.OptMpa.Item(i).Caption
         Exit For
      Else
         Mpa = "No"
      End If
   Next i

   For i = 0 To FRMHost.OptDetrend.Count - 1
      If FRMHost.OptDetrend.Item(i).Value = True Then
         Det = FRMHost.OptDetrend.Item(i).Caption
         Exit For
      Else
         Det = "No"
      End If
   Next i


   strTitle = strTitle & " (with " & Mpa & " M.P.A, " & Det & " Detrending)"
   '---Format Title-----
   With Selection
      .Value = strTitle
      .HorizontalAlignment = xlCenter
      .Font.FontStyle = "Bold"
      .Font.Size = 12
```

```
   End With

   With excel.Range("H1")
      .FormulaR1C1 = "=NOW()"
   End With

   With excel.Rows(2)
      .Font.FontStyle = "Bold"
      .HorizontalAlignment = xlCenter
   End With


   '=========Copy Data from FLX============
   For lcol = 0 To FLX.Cols - 1
      For lrow = 0 To FLX.Rows - 1
         Cells(lrow + 2, lcol + 1) = FLX.TextMatrix(lrow, lcol)
      Next lrow
   Next lcol

   '=====Get Cht Title if it had one ======
   If CHT.Title <> "" Then
      strProgram_Coords = "A" & lrow + 3 & ":H" & lrow + 3
      excel.Range(strProgram_Coords).Select
      Selection.Merge
      With Selection
         .Value = CHT.Title
      End With
   End If
   '========Format Data on Excel===========
   Columns("A:Z").EntireColumn.AutoFit

   strProgram_Coords = "A" & lrow + 5 & ":H" & lrow + 5
   excel.Range(strProgram_Coords).Select
   Selection.Merge
   With Selection
      .Value = "Produced by Easy TSA"
      .HorizontalAlignment = xlCenter
      .Font.FontStyle = "Bold"
      .Font.Size = 8
   End With


   '===============Clean Up================
   Set FLX = Nothing

End Sub

Sub Print_Export_Graph(ByRef Should_It_Be_Printed As Boolean, Optional Dont_Show_This As Boolean)
   '===============Init Vars================
   Dim CHT As MSChart, irow As Integer, Final_DataRow As Long
   Dim AllData As String
   irow = 1
   Set CHT = FRMHost.Controls("CHT" & What_Tab)
   Dim lrow As Long, lcol As Long
   lrow = 1
   '============Find Data Range=============
   Do While Cells(lrow, 1) <> ""
      lrow = lrow + 1
   Loop
   Final_DataRow = lrow

   '============Insert Graph================
   '#Add#
   Cells(10, 3).Select
   Charts.Add
   ActiveChart.Name = What_Tab & " Chart"
   ActiveChart.ChartType = What_CHT_Type(CHT.ChartType)
   '#Range of Data#
   AllData = "A2:A" & Final_DataRow & ",B2:B" & Final_DataRow
   '#Position of Graph#
   ActiveChart.SetSourceData Source:=Sheets("Data").Range(AllData), PlotBy _
         :=xlColumns

   '============Format Graph================
   Charts(1).Select
   With ActiveChart
      .HasTitle = True
      .ChartTitle.Font.Bold = True
```

```
      .ChartTitle.Characters.Text = Sheets("Data").Range("A1")
      .Axes(xlCategory, xlPrimary).HasTitle = True
      .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = Sheets("Data").Range("A2")
      .Axes(xlValue, xlPrimary).HasTitle = True
      .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = Sheets("Data").Range("B2")
      .PlotArea.Border.LineStyle = xlNone
      .ChartArea.Border.LineStyle = 0
   End With

   With ActiveChart.Axes(xlCategory)
      .HasMajorGridlines = False
      .HasMinorGridlines = False
   End With

   With ActiveChart.Axes(xlValue)
      .HasMajorGridlines = False
      .HasMinorGridlines = False
   End With

   ActiveChart.HasLegend = False
   ActiveChart.HasDataTable = False

   ActiveChart.PlotArea.Select
   With Selection.Border
      .ColorIndex = 16
      .Weight = xlThin
      .LineStyle = xlNone
   End With

   With Selection.Interior
      .ColorIndex = 2
      .PatternColorIndex = 1
      .Pattern = xlSolid
   End With


   '==========Special Add ons==============



   '---Add Significance Bars for AC & CC---
   Sheets("Data").Select
   If InStr(1, UCase(Cells(1, 1)), "CORR") <> 0 Then
      If InStr(1, UCase(Cells(1, 1)), "GROUPED") = 0 Then
         Call Add_Significance_Bars(Final_DataRow)
      End If
   End If
   '--------Extra Data Rows to Plot--------

   '-------------SEM to Plot---------------
   Sheets("Data").Select
   If Cells(2, 3) = "SEM" Then
      Call Add_Sem_to_Graph(Final_DataRow)
   End If

   '==============Print or Not=============
   If Should_It_Be_Printed = True Then
      excel.ActiveSheet.PrintOut
      ActiveWorkbook.Close (0)
      excel.Application.Quit
   Else
      If Dont_Show_This = False Then
         excel.Application.Visible = True
      Else
         excel.Application.Visible = False
      End If
   End If
End Sub


Private Sub Add_Sem_to_Graph(ByRef Final_DataRow As Long)
   '================Init Vars==================
   Dim Sem_Data As String

   Sem_Data = "C3:C" & Final_DataRow - 1
   Charts(1).Select
   '=================Add Sem===================
   ActiveChart.SeriesCollection(1).ErrorBar Direction:=xlY, Include:=xlBoth, _
```

```vba
        Type:=xlCustom, Amount:=Sheets("Data").Range(Sem_Data), _
        MinusValues:=Sheets("Data").Range(Sem_Data)

End Sub

Sub Add_Significance_Bars(ByRef Final_DataRow As Long)
   '==============Init Vars==============
   Dim irow As Long, AllData As String
   Dim Data_Count_L As Long
   Data_Count_L = (Final_DataRow - 3) * 2 - 1
   AllData = "A3:B" & Final_DataRow & ",C3:D" & Final_DataRow
   '===============Add Bar Data=============
   Sheets("Data").Select
   For irow = 3 To Final_DataRow - 1
      Cells(irow, 4) = (0 - Cells(irow, 3))
      '    Cells(Final_DataRow + 2 - irow, 5) = 1 '-2 / Sqr(Data_Count_L)
      '    Data_Count_L = Data_Count_L - 1
   Next irow

   '===============Update Graph=============
   Charts(1).Select
   ActiveChart.SetSourceData Source:=Sheets("Data").Range(AllData), PlotBy _
         :=xlColumns

   'ActiveChart.Axes(xlCategory).ReversePlotOrder = True

   '=============Colour Bars================
   ActiveChart.SeriesCollection(2).Select
   With Selection.Border
      .ColorIndex = 1
      .Weight = xlHairline
      .LineStyle = xlDot
   End With

   ActiveChart.SeriesCollection(3).Select
   With Selection.Border
      .ColorIndex = 1
      .Weight = xlHairline
      .LineStyle = xlDot
   End With

End Sub
'Sub PrintData(ByRef CHTtoPrint As MSChart, ByRef Headingspace As Integer, ByRef chtName As String, ByRef
»PorE As Boolean, ByRef CHTorDB As Boolean, Optional ColHeading As String, Optional ColLocation As
»MSFlexGrid)
''##########
''##INPUTS##
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
''CHTtoPrint = chart to get data from
''HeadingSpace = Space needed for heading info in excel
''CHTName = Name to put on excel sheet
''PorE = Print or Export the data
''CHTorDB = Get the data from the graph or from a datastore
''OPTIONAL ColHeading = what heading to use instead of time
''OPTIONAL ColLocation = Where the data is located to use instead of time
''~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'
'
'If CHTtoPrint.ColumnCount >= 1 Then
'    If UCase(CHTtoPrint.Name) = "CHTPERIOD" Then
'        Offset = 0
'    ElseIf UCase(CHTtoPrint.Name) = "CHTINT" Then
'        Offset = 0
'    Else
'    Offset = 1
'    End If
'Else
'    Offset = 1
'End If
'
'excel.Workbooks.Add
''-------Insert Headings--------
'excel.Range("A1:G1").Select
'Selection.Merge
'With Selection
'.Value = chtName
'.HorizontalAlignment = xlCenter
```

```
'.Font.FontStyle = "Bold"
'.Font.Size = 12
'End With
'
'With excel.Range("H1")
'.FormulaR1C1 = "=NOW()"
'End With
'
'If ColHeading = "" Then
'    With excel.Range("A2")
'        .Value = "Time (" & frmhost.txtTimeType & ")"
'        .Font.FontStyle = "Bold"
'    End With
'Else
'    With excel.Range("A2")
'        .Value = ColHeading
'        .Font.FontStyle = "Bold"
'    End With
'End If
'
'With excel.Range("B2")
'.Value = "Data"
'.Font.FontStyle = "Bold"
'End With
'
'
''------------------------------
'
''----Print data from graph or from datastore----
''################################################
'If CHTorDB = True Then
'
'    '----Insert data from graph----
'    For E_Col = 1 To CHTtoPrint.ColumnCount
'        For e_row = 1 To CHTtoPrint.RowCount
'            CHTtoPrint.Row = e_row
'            CHTtoPrint.Column = E_Col
'            ActiveSheet.Cells(e_row + Headingspace, E_Col + Offset) = CHTtoPrint.Data
'        Next e_row
'    Next E_Col
'    '------------------------------
'
'    If ColHeading = "" Then
'        '-----Insert Times-----
'        For e_row = 1 To CHTtoPrint.RowCount
'            ActiveSheet.Cells(e_row + Headingspace, 1) = frmhost.txtTimeVal * e_row
'        Next e_row
'        '------------------------------
'    Else
'        '-----Insert data for OTHER-----
'            For e_row = 1 To ColLocation.Rows
'                ActiveSheet.Cells(e_row + Headingspace, 1) = ColLocation.TextMatrix(e_row - 1, 0)
'            Next e_row
'        '------------------------------
'    End If
'
'
'Else
'
'
'    '----Insert data from datastore----
'    For E_Col = 1 To A_FRMData.Controls(CHTtoPrint.Name).Cols
'        For e_row = 1 To A_FRMData.Controls(CHTtoPrint.Name).Rows
'            A_FRMData.Controls(CHTtoPrint.Name).Row = e_row - 1
'            A_FRMData.Controls(CHTtoPrint.Name).Col = E_Col - 1
'            ActiveSheet.Cells(e_row + Headingspace, E_Col + Offset) =
»A_FRMData.Controls(CHTtoPrint.Name).Text
'        Next e_row
'    Next E_Col
'    '------------------------------
'
'    If ColHeading = "" Then
'        '-----Insert data for Time-----
'        If CHTtoPrint.ColumnCount >= 1 Then
'
'            For e_row = 1 To (A_FRMData.Controls(CHTtoPrint.Name).Rows)
'                ActiveSheet.Cells(e_row + Headingspace, 1) = frmhost.txtTimeVal * e_row
```

```
'            Next e_row
'
'        End If
'        '-----------------------------
'    ElseIf ColHeading = "Period" Then
'    'don't insert anything else
'    Else
'
'        For e_row = 1 To ColLocation.Rows
'                ActiveSheet.Cells(e_row + Headingspace, 1) = ColLocation.TextMatrix(e_row - 1, 0)
'        Next e_row
'
'    End If
'
'End If
'
''---Style Sheet-----------------
'Columns("A:A").EntireColumn.AutoFit
'Columns("B:B").EntireColumn.AutoFit
'Columns("C:C").EntireColumn.AutoFit
'Columns("H:H").EntireColumn.AutoFit
'Columns("A:C").HorizontalAlignment = xlLeft
'
'txtProgName = "A" & e_row + Headingspace + 1 & ":H" & e_row + Headingspace + 1
'excel.Range(txtProgName).Select
'Selection.Merge
'With Selection
'    .Value = "Produced by Graph++"
'    .HorizontalAlignment = xlCenter
'    .Font.FontStyle = "Bold"
'    .Font.Size = 8
'End With
'
'
''---------------------------------
'If PorE = True Then excel.ActiveSheet.PrintOut 'print it
'
'
'End Sub
'
'Sub PrintGraph(CHTtoPrint As MSChart, Headingspace As Integer, ByRef PorE As Boolean, Optional GraphType)
'
'If IsMissing(GraphType) Then
'    EndofData = CHTtoPrint.RowCount
'    StartofData = 1
'Else
'    If GraphType = "SEM" Then
'        EndofData = A_FRMData.Controls(CHTtoPrint.Name).Rows
'        StartofData = 1
'    ElseIf GraphType = "FOURIER" Then
'        EndofData = CHTtoPrint.RowCount - 1
'        StartofData = 2
'    Else
'        EndofData = CHTtoPrint.RowCount
'        StartofData = 1
'    End If
'End If
'
''=============Select Range of Data============
'If UCase(CHTtoPrint.Name) = "CHTRAW" And _
'frmhost.chkduplicate = 1 Then
'    AllData = "A" & Headingspace + StartofData & ":C" & EndofData + Headingspace
'
'ElseIf UCase(CHTtoPrint.Name) = "CHTCORRALATION" Then
'    AllData = "A" & Headingspace + StartofData & ":D" & EndofData + Headingspace
'Else
'    AllData = "A" & Headingspace + StartofData & ":B" & EndofData + Headingspace
'End If
''=============================================
'DataSheetName = ActiveSheet.Name
'excel.Range(AllData).Select
'
'    Charts.Add
'    If IsMissing(GraphType) Then
'        ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
'    Else
'        If GraphType = "FOURIER" Or GraphType = "DECONV" Then
'            ActiveChart.ChartType = xlColumnClustered
```

```
'        Else 'GraphType = "SEM" Then
'            ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
'        End If
'    End If
'
'    ActiveChart.SetSourceData Source:=Sheets(DataSheetName).Range(AllData), PlotBy _
'        :=xlColumns
'    ActiveChart.Location Where:=xlLocationAsObject, Name:=DataSheetName
'
'    With ActiveChart
'        .HasTitle = True
'        .ChartTitle.Characters.Text = Sheets(DataSheetName).Range("A1")
'        .Axes(xlCategory, xlPrimary).HasTitle = True
'        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = Sheets(DataSheetName).Range("A2")
'        .Axes(xlValue, xlPrimary).HasTitle = True
'        .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = Sheets(DataSheetName).Range("B2")
'        .PlotArea.Border.LineStyle = xlNone
'        .ChartArea.Border.LineStyle = 0
''         .DrawingObjects.RoundedCorners = False
' '        .DrawingObjects.Shadow = False
'    End With
'    With ActiveChart.Axes(xlCategory)
'        .HasMajorGridlines = False
'        .HasMinorGridlines = False
'    End With
'    With ActiveChart.Axes(xlValue)
'        .HasMajorGridlines = False
'        .HasMinorGridlines = False
'    End With
'    ActiveChart.HasLegend = False
'    ActiveChart.HasDataTable = False
'    ActiveChart.PlotArea.Select
'    With Selection.Border
'        .ColorIndex = 16
'        .Weight = xlThin
'        .LineStyle = xlNone
'
'    End With
'    With Selection.Interior
'        .ColorIndex = 2
'        .PatternColorIndex = 1
'        .Pattern = xlSolid
'    End With
'
'    If IsMissing(GraphType) Then
'        'do nothing else
'    Else
'        If GraphType = "SEM" Then
'            SEMData = "C" & Headingspace + 1 & ":C" & A_FRMData.Controls(CHTtoPrint.Name).Rows +
»Headingspace
'
'            ActiveChart.SeriesCollection(1).ErrorBar Direction:=xlY, Include:=xlBoth, _
'            Type:=xlCustom, Amount:=Sheets(DataSheetName).Range(SEMData),
»MinusValues:=Sheets(DataSheetName).Range(SEMData)
'
'        ElseIf GraphType = "CORRA" Then
'            ActiveChart.Axes(xlCategory).ReversePlotOrder = True
'        End If
'    End If
'
'
'If PorE = True Then
'    excel.ActiveSheet.PrintOut
'    ActiveWorkbook.Close (0)
'    excel.Application.Quit
'Else
'   'excel.Application.DisplayFullScreen = True
'   excel.Application.Visible = True
'End If
'
'End Sub
```

EASY TSA - MODULE - REAL_RANDOM.bas MDLRANDOM

```
'Attribute VB_Name = "basISAAC"
' ************************************************************************
' Module:        basISAAC.bas
'
'
' ************************************************************************
'  ISAAC has an output within these ranges:
'
'      -0.9999999999990, 0.9999999999990    Double Precision
'      -2147483648, 2147483647              Long Integer
'
' ************************************************************************
Option Explicit

' ************************************************************************
' Constants
' ************************************************************************
Private Const MAXINT_4          As Long = 2147483647
Private Const MAXDBL_4          As Double = 4294967296#

' Other constants defined to be used in the Visual Basic version:
' Powers of 2: k2_X means 2^X
Private Const k2_8              As Long = 256
Private Const k2_16             As Long = 65536
Private Const k2_24             As Long = 16777216

Private Const MASK_16LONG       As Long = 65536
Private Const MASK_16           As Long = &HFFFF
Private Const MASK_15           As Long = &H7FFF
Private Const MASK_31           As Long = &H7FFFFFFF
Private Const MASK_COMMON       As Long = &HFFFFFFFF
Private Const k2_31             As Double = 2147483648#   ' 2^31   ==  2147483648 == 80000000
Private Const k2_31Neg          As Double = 0# - k2_31    ' -2^31  == -2147483648 == 80000000
Private Const k2_31b            As Double = k2_31 - 1#    ' 2^31-1 ==  2147483647 == 7FFFFFFF
Private Const k2_32             As Double = 2# * k2_31    ' 2^32   ==  4294967296 == 0
Private Const k2_32b            As Double = k2_32 - 1#    ' 2^32-1 ==  4294967295 (m32Double)
Private Const kMT_1             As Double = 1# / k2_32b
Private Const kMT_2             As Double = 1# / k2_32
Private Const kMT_3             As Double = kMT_2
Private Const kMT_Gap           As Double = 0.0000000000005          ' 5.0E-13
Private Const kMT_Gap2          As Double = 2# * kMT_Gap             ' 1.0E-12
Private Const kMT_GapInterval   As Double = 1# - kMT_Gap2            ' 0.9999999999990
Private Const kMT_2b            As Double = kMT_GapInterval / k2_32b
Private Const kMT_3b            As Double = kMT_2b

' Constants for shift operations:
Private Const kShift1           As Long = 2             ' 2==2^1
Private Const kShift2           As Long = 4             ' 4==2^2
Private Const kShift3           As Long = 8             ' 8==2^3
Private Const kShift4           As Long = 16            ' 16==2^4
Private Const kShift5           As Long = 32            ' 32==2^5
Private Const kShift6           As Long = 64            ' 64==2^6
Private Const kShift7           As Long = 128           ' 128==2^7
Private Const kShift8           As Long = 256           ' 256==2^8
Private Const kShift9           As Long = 512           ' 512==2^9
Private Const kShift10          As Long = 1024          ' 1024==2^10
Private Const kShift11          As Long = 2048          ' 2048==2^11
Private Const kShift12          As Long = 4096          ' 4096==2^12
Private Const kShift13          As Long = 8192          ' 8192==2^13
Private Const kShift14          As Long = 16384         ' 16384==2^14
Private Const kShift15          As Long = 32768         ' 32768==2^15
Private Const kShift16          As Long = 65536         ' 65536==2^16
Private Const kShift17          As Long = 131072        ' 131072==2^17
Private Const kShift18          As Long = 262144        ' 262144==2^18
Private Const kShift19          As Long = 524288        ' 524288==2^19
Private Const kShift20          As Long = 1048576       ' 1048576==2^20
Private Const kShift21          As Long = 2097152       ' 2097152==2^21
Private Const kShift22          As Long = 4194304       ' 4194304==2^22
Private Const kShift23          As Long = 8388608       ' 8388608==2^23
Private Const kShift24          As Long = 16777216      ' 16777216==2^24
Private Const kShift25          As Long = 33554432      ' 33554432==2^25
Private Const kShift26          As Long = 67108864      ' 67108864==2^26
Private Const kShift27          As Long = 134217728     ' 134217728==2^27
Private Const kShift28          As Long = 268435456     ' 268435456==2^28
Private Const kShift29          As Long = 536870912     ' 536870912==2^29
Private Const kShift30          As Long = 1073741824    ' 1073741824==2^30
Private Const kShift31          As Double = 2147483648# ' 2147483648==2^31
```

```
Private Const kShift32          As Double = 4294967296#   ' 4294967296==2^32

' ************************************************************************
' API Declares
' ************************************************************************
' The GetTickCount() API will capture the time in milliseconds.  The
' counter overflows after 1192.8 hours (49.7 days) from the last reboot.
Private Declare Function GetTickCount Lib "kernel32" () As Long

' ************************************************************************
' Module variables
'
' readable.c
'
'    /* a ub4 is an unsigned 4-byte quantity */
'    /* external results */
'    ub4 randrsl[256];
'
'    /* internal state */
'    static    ub4 mm[256];
'    static    ub4 aa=0, bb=0, cc=0;  /* seed values */
' ************************************************************************
Private malngRand(256)   As Long
Private malngMem(256)    As Long
Private mlngSeed1        As Long
Private mlngSeed2        As Long
Private mlngSeed3        As Long

' ************************************************************************
' Routine:       ISAAC_Prng
'
' Description:   A quantity of random values will be generated based on the
'                user request.
'
' Parameters:    lngArraySize - [Optional] Number of elements in return array.
'                     Default return number of 1.
'                blnReturnFloat - [Optional] Return random values in an array.
'                     TRUE - Double precision
'                     FALSE - Long integer
'
' Returns:       An array of random generated values
'
' ========================================================================
'
' ************************************************************************
Public Function ISAAC_Prng(Optional ByVal lngArraySize As Long = 1, _
      Optional ByVal blnReturnFloat As Boolean = True) As Variant

   Dim lngIndex   As Long
   Dim lngRand    As Long
   Dim lngLoop    As Long
   Dim lngCount   As Long
   Dim alngData() As Long
   Dim dblRand    As Double
   Dim adblData() As Double
   Dim blnLoaded  As Boolean

   On Error GoTo ISAAC_Prng_Error

   Erase malngRand()
   Erase malngMem()
   lngCount = 0
   blnLoaded = False

   ReDim adblData(lngArraySize)  ' resize the return arrays
   ReDim alngData(lngArraySize)
   Call RandomInit(True)         ' full seeding and calculations

   ' Start creating the random data
   For lngIndex = 0 To lngArraySize - 1

      Call RandomInit(False)    ' partial mixing and calculations
      Call ISAAC_Calc           ' load the random data array

      ' unload the random data array into it's appropriate return array
      For lngLoop = 0 To 255

         lngRand = malngRand(lngLoop)      ' capture one generated value
```

```
         ' Return float value
         If blnReturnFloat Then
            dblRand = lngRand / MAXINT_4  ' create a float value
            adblData(lngCount) = dblRand  ' -0.9999999999990, 0.9999999999990
         Else
            ' return long integer
            alngData(lngCount) = lngRand  ' -2147483648, 2147483647
         End If

         lngCount = lngCount + 1

         ' if the requested number of elements have been
         ' collected then exit this loop
         If lngCount = lngArraySize Then
            Exit For
         End If

      Next lngLoop

      ' if the requested number of elements have been
      ' collected then exit this loop
      If lngCount = lngArraySize Then
         Exit For
      End If

   Next lngIndex

ISAAC_Prng_CleanUp:
   If blnReturnFloat Then
      ISAAC_Prng = adblData()
   Else
      ISAAC_Prng = alngData()
   End If

   Erase alngData()   ' empty the temp arrays
   Erase adblData()
   On Error GoTo 0    ' nullify this error trap
   Exit Function

ISAAC_Prng_Error:
   MsgBox "Err: " & CStr(Err.Number) & "  " & Err.Description
   ReDim alngData(1)    ' resize arrays to one element, return value = 0
   ReDim adblData(1)
   Resume ISAAC_Prng_CleanUp

End Function

' void isaac()
Private Function ISAAC_Calc() As Long

   ' register ub4 i,x,y;
   Dim lngIndex  As Long
   Dim xx        As Long
   Dim yy        As Long
   Dim lngTemp   As Long

   ' cc = cc + 1;    /* cc just gets incremented once per 256 results */
   ' bb = bb + cc;   /* then combined with bb */
   mlngSeed3 = uAdd(mlngSeed3, 1)          ' mlngSeed3 just gets incremented once per 256 results
   mlngSeed2 = uAdd(mlngSeed2, mlngSeed3)  ' then combined with mlngSeed2
   Erase malngRand()

   ' for (i=0; i<256; ++i)
   For lngIndex = 0 To 255

      ' x = mm[i];
      xx = malngMem(lngIndex)

      ' switch (i%4)
      '  {
      '   case 0: aa = aa^(aa<<13); break;
      '   case 1: aa = aa^(aa>>6); break;
      '   case 2: aa = aa^(aa<<2); break;
      '   case 3: aa = aa^(aa>>16); break;
      '  }
      lngTemp = lngIndex Mod 4
      Select Case lngTemp
```

```
      Case 0: mlngSeed1 = mlngSeed1 Xor uMult(mlngSeed1, kShift13)
      Case 1: mlngSeed1 = mlngSeed1 Xor uDiv(mlngSeed1, kShift6)
      Case 2: mlngSeed1 = mlngSeed1 Xor uMult(mlngSeed1, kShift2)
      Case 3: mlngSeed1 = mlngSeed1 Xor uDiv(mlngSeed1, kShift16)
      End Select

      ' aa = mm[(i+128)%256] + aa;
      mlngSeed1 = uAdd(malngMem((lngIndex + 128) Mod 256), mlngSeed1)

      ' mm[i] = y = mm[(x>>2)%256] + aa + bb;
      yy = uAdd(uAdd(malngMem((uDiv(xx, kShift2)) Mod 256), mlngSeed1), mlngSeed2)
      malngMem(lngIndex) = yy

      ' randrsl[i] = bb = mm[(y>>10)%256] + x;
      mlngSeed2 = uAdd(malngMem((uDiv(yy, kShift10)) Mod 256), xx)
      malngRand(lngIndex) = mlngSeed2

      ' Note that bits 2..9 are chosen from x but 10..17 are chosen
      ' from y.  The only important thing here is that 2..9 and 10..17
      ' do not overlap.  2..9 and 10..17 were then chosen for speed in
      ' the optimized version.
      '

   Next lngIndex

End Function

' void randinit(flag)
Private Sub RandomInit(ByRef blnSeed As Boolean)

   ' word flag;
   ' word i;
   ' ub4 a,b,c,d,e,f,g,h;
   Dim lngIndex As Long
   Dim AA As Long
   Dim BB As Long
   Dim CC As Long
   Dim DD As Long
   Dim EE As Long
   Dim FF As Long
   Dim GG As Long
   Dim HH As Long

   ' aa=bb=cc=0;   /* See CreateSeed()  Ken Ives */
   If blnSeed Then
      Call CreateSeed  ' fill the seeds with something
   End If

   ' a=b=c=d=e=f=g=h=0x9e3779b9;  /* the golden ratio */
   AA = &H9E3779B9
   BB = &H9E3779B9
   CC = &H9E3779B9
   DD = &H9E3779B9
   EE = &H9E3779B9
   FF = &H9E3779B9
   GG = &H9E3779B9
   HH = &H9E3779B9

   ' for (i=0; i<4; ++i)          /* scramble it */
   For lngIndex = 0 To 4
      ' mix(a,b,c,d,e,f,g,h);
      Call Mix(AA, BB, CC, DD, EE, FF, GG, HH)
   Next lngIndex

   ' for (i=0; i<256; i+=8)   /* fill in mm[] with messy stuff */
   For lngIndex = 0 To 248 Step 8
      ' Note that I am looping 0 to 248 because the last index
      ' will be 248 + 7 == 255
      '
      ' if (flag)  /* use all the information in the seed */
      If blnSeed Then
         ' a+=randrsl[i  ]; b+=randrsl[i+1]; c+=randrsl[i+2]; d+=randrsl[i+3];
         ' e+=randrsl[i+4]; f+=randrsl[i+5]; g+=randrsl[i+6]; h+=randrsl[i+7];
         AA = uAdd(AA, malngRand(lngIndex))
         BB = uAdd(BB, malngRand(lngIndex + 1))
         CC = uAdd(CC, malngRand(lngIndex + 2))
         DD = uAdd(DD, malngRand(lngIndex + 3))
         EE = uAdd(EE, malngRand(lngIndex + 4))
```

```
            FF = uAdd(FF, malngRand(lngIndex + 5))
            GG = uAdd(GG, malngRand(lngIndex + 6))
            HH = uAdd(HH, malngRand(lngIndex + 7))
        End If

        ' mix(a,b,c,d,e,f,g,h);
        Call Mix(AA, BB, CC, DD, EE, FF, GG, HH)

        ' mm[i  ]=a; mm[i+1]=b; mm[i+2]=c; mm[i+3]=d;
        ' mm[i+4]=e; mm[i+5]=f; mm[i+6]=g; mm[i+7]=h;
        malngMem(lngIndex) = AA
        malngMem(lngIndex + 1) = BB
        malngMem(lngIndex + 2) = CC
        malngMem(lngIndex + 3) = DD
        malngMem(lngIndex + 4) = EE
        malngMem(lngIndex + 5) = FF
        malngMem(lngIndex + 6) = GG
        malngMem(lngIndex + 7) = HH

    Next lngIndex

    ' if (flag)
    If blnSeed Then
        ' do a second pass to make all of
        ' the seed affect all of malngMem()
        '
        ' for (i=0; i<256; i+=8)
        For lngIndex = 0 To 248 Step 8
            ' Note that I am looping 0 to 248 because the last index
            ' will be 248 + 7 == 255
            '
            ' a+=mm[i  ]; b+=mm[i+1]; c+=mm[i+2]; d+=mm[i+3];
            ' e+=mm[i+4]; f+=mm[i+5]; g+=mm[i+6]; h+=mm[i+7];
            AA = uAdd(AA, malngMem(lngIndex))
            BB = uAdd(BB, malngMem(lngIndex + 1))
            CC = uAdd(CC, malngMem(lngIndex + 2))
            DD = uAdd(DD, malngMem(lngIndex + 3))
            EE = uAdd(EE, malngMem(lngIndex + 4))
            FF = uAdd(FF, malngMem(lngIndex + 5))
            GG = uAdd(GG, malngMem(lngIndex + 6))
            HH = uAdd(HH, malngMem(lngIndex + 7))

            ' mix(a,b,c,d,e,f,g,h);
            Call Mix(AA, BB, CC, DD, EE, FF, GG, HH)

            ' mm[i  ]=a; mm[i+1]=b; mm[i+2]=c; mm[i+3]=d;
            ' mm[i+4]=e; mm[i+5]=f; mm[i+6]=g; mm[i+7]=h;
            malngMem(lngIndex) = AA
            malngMem(lngIndex + 1) = BB
            malngMem(lngIndex + 2) = CC
            malngMem(lngIndex + 3) = DD
            malngMem(lngIndex + 4) = EE
            malngMem(lngIndex + 5) = FF
            malngMem(lngIndex + 6) = GG
            malngMem(lngIndex + 7) = HH
        Next lngIndex
    End If

    ' isaac();  /* fill in the first set of results */
    Call ISAAC_Calc

End Sub

' #define mix(a,b,c,d,e,f,g,h) \
Private Sub Mix(ByRef AA As Long, _
        ByRef BB As Long, _
        ByRef CC As Long, _
        ByRef DD As Long, _
        ByRef EE As Long, _
        ByRef FF As Long, _
        ByRef GG As Long, _
        ByRef HH As Long)

    ' a^=b<<11; d+=a; b+=c; \
    AA = AA Xor uMult(BB, kShift11)
    DD = uAdd(DD, AA)
    BB = uAdd(BB, CC)
```

```
   ' b^=c>>2;  e+=b; c+=d; \
   BB = BB Xor uDiv(CC, kShift2)
   EE = uAdd(EE, BB)
   CC = uAdd(CC, DD)

   ' c^=d<<8;  f+=c; d+=e; \
   CC = BB Xor uMult(CC, kShift8)
   EE = uAdd(EE, BB)
   DD = uAdd(DD, EE)

   ' d^=e>>16; g+=d; e+=f; \
   DD = DD Xor uDiv(EE, kShift16)
   GG = uAdd(GG, DD)
   EE = uAdd(EE, FF)

   ' e^=f<<10; h+=e; f+=g; \
   EE = EE Xor uMult(FF, kShift10)
   HH = uAdd(HH, EE)
   FF = uAdd(FF, GG)

   ' f^=g>>4;  a+=f; g+=h; \
   FF = FF Xor uDiv(GG, kShift4)
   AA = uAdd(AA, FF)
   GG = uAdd(GG, HH)

   ' g^=h<<8;  b+=g; h+=a; \
   GG = GG Xor uMult(HH, kShift8)
   BB = uAdd(BB, GG)
   HH = uAdd(HH, AA)

   ' h^=a>>9;  c+=h; a+=b; \
   HH = HH Xor uDiv(AA, kShift9)
   CC = uAdd(CC, HH)
   AA = uAdd(AA, BB)

End Sub

' ************************************************************************
'
'
' Note from Mark Hutchinson's presentation about Microsoft's VB random number
' generator, invoke Rnd (-1) prior to Randomize statement in order to actually
' initialize with a new seed.
'
'        Rnd -1     ' Reset the random number generator
'        Randomize  ' Reseed the generator
' ************************************************************************
Private Sub CreateSeed()


   Dim intIndex      As Integer
   Dim lngTemp       As Long
   Dim alngData(50)  As Long

   Erase alngData()
   alngData(0) = (GetTickCount() And MASK_COMMON)

   For intIndex = 1 To 50
      lngTemp = alngData(intIndex - 1)
      alngData(intIndex) = uAdd(uMult(1812433253, (lngTemp Xor uDiv(lngTemp, kShift30))), intIndex)
   Next intIndex

   DoEvents
   Rnd (-1)  ' Reset the RNG
   Randomize GetTickCount()

   mlngSeed1 = alngData(Int(Rnd * 15))
   mlngSeed2 = alngData(Int(Rnd * 15) + 20)
   mlngSeed3 = alngData(Int(Rnd * 20) + 30)
   Erase alngData()

End Sub

' ************************************************************************
' Routine:       uAdd
'
' Description:   Unsigned Add: adds the two (signed) Long parameters, treated
'                as unsigned long, and returns the result as a (signed) Long
```

```
'               result
'
' Parameters:    x - Value1 to be added
'               y - Value2 to be added
'
' Returns:       New value
'
' ============================================================================

' ****************************************************************************
Private Function uAdd(ByVal X As Long, ByVal Y As Long) As Long

   Dim tmp As Double

   tmp = CDbl(X) + Y

   If tmp < k2_31Neg Then
      uAdd = CLng(k2_32 + tmp)
   Else
      If tmp > k2_31b Then
         uAdd = CLng(tmp - k2_32)
      Else
         uAdd = CLng(tmp)
      End If
   End If

End Function

' ****************************************************************************
' Routine:       uMulti
'
' Description:   Shifts the bits to the left the specified number of
'               positions and returns the new value.  Bits "falling off"
'               the left edge do not wrap around.  Fill bits coming in from
'               right are 0.  A shift left is effectively a multiplication
'               by 2.  Some common languages like C/C++ or Java have an
'               operator for this job: "<<".
'
' Parameters:    x - Number to be manipulated
'               y - number of shift positions
'
' Returns:       New manipulated value
'
' ============================================================================
' ****************************************************************************
Private Function uMult(ByVal X As Long, ByVal Y As Long) As Long
   'Unsigned Mult: multiplies the two (signed) Long parameters, treated as
   'unsigned long, and returns the lowest 4 bytes of the 8 bytes result
   'as a (signed) Long result:

   'This function emulates the multiplication of two (unsigned long) numbers,
   'and is needed because the type Double has only a 53 bits mantissa
   'and the result of multiplying 2 Long variables might need 64 bits to accurately
   'represent the result in some cases.

   'x   == ABCD  == A000 + B00 + C0 + D
   'y   == EFGH  == E000 + F00 + G0 + H

   '       Note: in the following, "ae" means the 2 bytes result of A*E, "bg" of B*G, etc:
   '
   'x*y == ae 000 000 +  'discard, result is too high
   '       af 000  00 +  'discard, result is too high
   '       ag 000   0 +  'discard, result is too high
   '       ah 000     +  'take lowest byte

   '       be  00 000 +  'discard, result is too high
   '       bf  00  00 +  'discard, result is too high
   '       bg  00   0 +  'take lowest byte
   '       bh  00     +  'take both bytes

   '       ce   0 000 +  'discard, result is too high
   '       cf   0  00 +  'take lowest byte
   '       cg   0   0 +  'take both bytes
   '       ch   0     +  'take both bytes

   '       de     000 +  'take lowest byte
   '       df      00 +  'take both bytes
   '       dg       0 +  'take both bytes
```

```
'       dh            'take both bytes


'Given that "aa" and "ee" are used just once, they are replaced by their
'values: aa == (x \ k2_24) and ee == (y \ k2_24), and they are not declared:
'Dim aa As Long, ee As Long

Dim BB As Long, CC As Long, DD As Long
Dim FF As Long, GG As Long, HH As Long
Dim r3 As Long, r2 As Long, r1 As Long, r0 As Long
Dim tmp As Double

'x==ABCD, y==EFGH
BB = (X \ k2_16) Mod k2_8: CC = (X \ k2_8) Mod k2_8: DD = X Mod k2_8
FF = (Y \ k2_16) Mod k2_8: GG = (Y \ k2_8) Mod k2_8: HH = Y Mod k2_8


'get the 1st (lowest) byte of the result, r0:
'       dh             'take both bytes
r0 = DD * HH
```

```
   'get the 2nd byte of the result, r1, and add carry from r0:
   '       ch   0      +  'take both bytes
   '       dg       0     'take both bytes
   r1 = CC * HH + DD * GG + r0 \ k2_8

   'get the 3rd byte of the result, r2, and add carry from r1:
   '       bh  00      +  'take both bytes
   '       cg   0    0 +  'take both bytes
   '       df       00    'take both bytes
   r2 = BB * HH + CC * GG + DD * FF + r1 \ k2_8

   'get the 4th (highest) byte of the result, r3, and add carry from r2:
   '       ah 000      +  'take lowest byte
   '       bg  00    0 +  'take lowest byte
   '       cf   0   00 +  'take lowest byte
   '       de      000    'take lowest byte
   r3 = (((X \ k2_24) * HH + BB * GG + CC * FF + DD * (Y \ k2_24)) Mod k2_8) + r2 \ k2_8

   'tmp = CDbl(r3) * k2_24 + r2 * k2_16 + r1 * k2_8 + r0
   tmp = CDbl(r3 Mod k2_8) * k2_24 + (r2 Mod k2_8) * k2_16 + (r1 Mod k2_8) * k2_8 + (r0 Mod k2_8)

   'now we have a 32 bits number (tmp) that can be processed without losing precision
   'using the 53 bits mantissa of the Double type

   If tmp < k2_31Neg Then
      uMult = CLng(k2_32 + tmp)
   Else
      If tmp > k2_31b Then
         uMult = CLng(tmp - k2_32)
      Else
         uMult = CLng(tmp)
      End If
   End If

End Function

' ****************************************************************************
' Routine:       uDiv
'
'' Description:  Shifts the bits to the right the specified number of
'               positions and returns the new value.  Bits "falling off"
'               the right edge do not wrap around.  Fill bits coming in from
'               left are 0.  A shift right is effectively a multiplication
'               by 2.  Some common languages like C/C++ or Java have an
'               operator for this job: ">>".
'
' Parameters:    x - Number to be manipulated
'               y - number of shift positions
'
' Returns:       New manipulated value
'
' ============================================================================
' ****************************************************************************
Private Function uDiv(ByVal X As Long, ByVal Y As Long) As Long
   'Unsigned Divide: divides the two (signed) Long parameters, treated as
   'unsigned long, and returns the result as a (signed) Long result:

   If Y < 0 Then
      Y = k2_32 + Y
   End If

   If X < 0 Then
      uDiv = (Abs(Fix((k2_32 + X) / Y)))
   Else
      uDiv = CLng(Fix(X / Y))
   End If

End Function

Public Sub Main()
   ' ***************************************************************************
   ' For testing the ISAAC random number generator only.  Will create two files.
   ' One for float values and the other for long integers.  Only 1 in every SAMPLE
   ' values will be written to the file. Rename this routine when not wanted.
   ' ***************************************************************************
```

```
Dim hFile       As Long
Dim lngIndex    As Long
Dim lngColCount As Long
Dim lngCount    As Long
Dim alngData()  As Long
Dim adblData()  As Double
Dim dblTotal    As Double
Dim dblLow      As Double
Dim dblHigh     As Double
Dim strFmt      As String
Dim strTemp     As String
Dim strTitle    As String

Const FN_DBL    As String = "C:\Isaac_Dbl.txt"
Const FN_LNG    As String = "C:\Isaac_Lng.txt"
Const FCOUNT    As Long = 100000
Const SAMPLE    As Long = 100

'*****************************************************************
' Generate random float values  -0.9999999999990, 0.9999999999990
'*****************************************************************
Screen.MousePointer = vbHourglass
Erase adblData()
lngColCount = 0
lngCount = 0
dblLow = 1#
dblHigh = 0#
dblTotal = 0#
strFmt = String$(15, "@")
strTitle = "ISAAC Double precision values" & vbCrLf & _
      Format$(FCOUNT, "#,0") & " random generated numbers" & vbCrLf & _
      "Saving every " & CStr(SAMPLE) & "th value for this display"

hFile = FreeFile                        ' get first free file handle
Open FN_DBL For Output As #hFile        ' create empty output file

Print #hFile, strTitle & vbCrLf         ' print the title
adblData = ISAAC_Prng(FCOUNT, True)     ' generate float values

' Gather the output statistics
For lngIndex = 0 To FCOUNT - 1

   dblTotal = dblTotal + adblData(lngIndex)  ' Accumulate overall total

   If dblLow > adblData(lngIndex) Then
      dblLow = adblData(lngIndex)           ' check for lowest value
   ElseIf adblData(lngIndex) > dblHigh Then
      dblHigh = adblData(lngIndex)          ' check for highest value
   End If

Next lngIndex

' format and write the statistics to the file
Print #hFile, "Average: " & Format$(FormatNumber(dblTotal / CDbl(FCOUNT), 12), strFmt)
Print #hFile, " Lowest: " & Format$(FormatNumber(dblLow, 12), strFmt)
Print #hFile, "Highest: " & Format$(FormatNumber(dblHigh, 12), strFmt)
Print #hFile, " "

' dump the contents of the array to the output file
For lngIndex = 0 To FCOUNT - 1

   ' Just use every 100th value for the output file
   If lngCount Mod SAMPLE = 0 Then
      ' write to the test file.
      strTemp = Format$(FormatNumber(adblData(lngIndex), 12), strFmt)
      Print #hFile, strTemp & Space$(2);    ' write to the file
      lngColCount = lngColCount + 1         ' increment column counter
   End If

   ' see if we have 5 columns
   If lngColCount = 5 Then
      Print #hFile, ""     ' prints Chr$(13) + Chr$(10) at the end of the line
      lngColCount = 0      ' reset column counter
   End If

   lngCount = lngCount + 1
```

```
   Next lngIndex

   Close #hFile
   Erase adblData()
   Screen.MousePointer = vbNormal
   MsgBox FN_DBL, vbOKOnly, "ISAAC Float Values"

   '*****************************************************************
   ' Generate long integer values  -2147483648, 2147483647
   '*****************************************************************
   Screen.MousePointer = vbHourglass
   Erase alngData()
   lngColCount = 0
   lngCount = 0
   dblLow = MAXINT_4
   dblHigh = 0#
   dblTotal = 0#
   strFmt = String$(11, "@")
   strTitle = "ISAAC Long Integer values" & vbCrLf & _
         Format$(FCOUNT, "#,0") & " random generated numbers" & vbCrLf & _
         "Saving every " & CStr(SAMPLE) & "th value for this display"

   hFile = FreeFile                     ' get first free file handle
   Open FN_LNG For Output As #hFile     ' create empty output file

   Print #hFile, strTitle & vbCrLf      ' print the title
   alngData = ISAAC_Prng(FCOUNT, False)  ' generate long integer values

   ' Gather the output statistics
   For lngIndex = 0 To FCOUNT - 1

      dblTotal = dblTotal + alngData(lngIndex)  ' Accumulate overall total

      If dblLow > alngData(lngIndex) Then
         dblLow = alngData(lngIndex)          ' check for lowest value
      ElseIf alngData(lngIndex) > dblHigh Then
         dblHigh = alngData(lngIndex)         ' check for highest value
      End If

   Next lngIndex

   ' format and write the statistics to the file
   Print #hFile, "Average: " & Format$(Int(dblTotal / CDbl(FCOUNT)), strFmt)
   Print #hFile, " Lowest: " & Format$(dblLow, strFmt)
   Print #hFile, "Highest: " & Format$(dblHigh, strFmt)
   Print #hFile, " "

   ' dump the contents of the array to the output file
   For lngIndex = 0 To FCOUNT - 1

      ' Just use every 100th value for the output file
      If lngCount Mod SAMPLE = 0 Then
         ' write to the test file.
         strTemp = Format$(alngData(lngIndex), strFmt)
         Print #hFile, strTemp & Space$(2);   ' write to the file
         lngColCount = lngColCount + 1        ' increment column counter
      End If

      ' see if we have 6 columns
      If lngColCount = 6 Then
         Print #hFile, ""         ' prints Chr$(13) + Chr$(10) at the end of the line
         lngColCount = 0          ' reset column counter
      End If

      lngCount = lngCount + 1

   Next lngIndex

   Close #hFile
   Erase alngData()
   Screen.MousePointer = vbNormal
   MsgBox FN_LNG & vbCrLf & vbCrLf & "TESTING COMPLETE!", vbOKOnly, "ISAAC Long Integers"
   End

End Sub

Public Sub Main_DH()
   ' ***************************************************************************
```

```
    ' Rename this routine to Main() and press F5 to execute. This will build the
    ' approximately 11mb binary input file needed when using Diehard or ENT for
    ' randomness testing.  On a 500mhz, 256mb RAM PC, this will take approximately
    ' two minutes to complete.
    ' ************************************************************************

    Dim hFile       As Integer
    Dim alngData()  As Long
    Dim lngPointer  As Long
    Dim lngLoop     As Long
    Dim lngIndex    As Long

    Const FN_TEST   As String = "C:\Isaac.bin"
    Const FCOUNT    As Long = 10240

    Screen.MousePointer = vbHourglass
    lngPointer = 1                          ' init pointer for output file

    hFile = FreeFile                        ' capture first free file handle
    Open FN_TEST For Output As #hFile       ' make sure the file is empty
    Close #hFile                            ' close the file

    hFile = FreeFile                        ' capture first free file handle
    Open FN_TEST For Binary Access Write As #hFile

    ' Generate some random data
    For lngLoop = 1 To 274

       Erase alngData()                     ' empty the array
       alngData = ISAAC_Prng(FCOUNT, False)  ' Get some random integers

       ' Unload data to output file
       For lngIndex = 0 To FCOUNT - 1
          Put #hFile, lngPointer, alngData(lngIndex) ' write to ouput file
          lngPointer = lngPointer + 4               ' update pointer within output file
       Next lngIndex

    Next lngLoop

    Close #hFile
    Erase alngData()
    Screen.MousePointer = vbNormal
    MsgBox FN_TEST & vbCrLf & vbCrLf & "FINISHED!", vbOKOnly, "ISAAC Diehard/ENT test file"
    End

End Sub
```

```
Option Explicit

Sub Resample_Data()
   '-----Init Vars------
   Dim FLX As MSFlexGrid
   Dim irow As Integer, iStart As Integer, iEnd As Integer
   Dim iCheck As Integer, strName As String

   Set FLX = FRMHost.FLXRaw_POI

   '-----Get Limits-----
   For irow = 1 To FLX.Rows - 1
start:
      '---Set name---
      strName = FLX.TextMatrix(irow, 0)
      iEnd = 0
      iStart = 0

      '--Escape Clause--
      If strName = "" Then
         FRMHost.FLXMain_SelChange
         'FRMHost.chkRaw_Resample.Visible = False
         'FRMHost.chkSave.Visible = False
         Exit Sub
      End If
      '---Set Start---
      iStart = FLX.TextMatrix(irow, 1)

      '---Find Next Name---
      For iCheck = irow + 1 To FLX.Rows - 1
         If FLX.TextMatrix(iCheck, 0) = strName Then
            '---Set End---
            iEnd = FLX.TextMatrix(iCheck, 1)
            Exit For
         End If
      Next iCheck

      '---Error Check---
      If iEnd = 0 Then
         MsgBox ("No End Point Selected for " & strName)
         FRMHost.chkRaw_Resample.Value = 0

         If FLX.Rows <> 2 Then
            FLX.RemoveItem (iCheck - 1)
         Else
            FLX.TextMatrix(1, 0) = ""
            FLX.TextMatrix(1, 1) = ""
         End If
         GoTo start
      ElseIf iEnd < iStart Then
         MsgBox ("One or more POI's Chosen are no good for " & strName & ", please select again")
         FRMHost.chkRaw_Resample.Value = 0

         If FLX.Rows <> 3 Then
            FLX.RemoveItem (irow)
            FLX.RemoveItem (iCheck - 1)

         Else
            FLX.TextMatrix(1, 0) = ""
            FLX.TextMatrix(1, 1) = ""
            FLX.TextMatrix(2, 0) = ""
            FLX.TextMatrix(2, 1) = ""
         End If

         GoTo start
      End If

      '-----Apply to Main Data Array-----
      If Duplicate = True Then
         Call Resize_Array(strName, iStart, iEnd, True)
      End If

      Call Resize_Array(strName, iStart, iEnd, False)

      '----remove done rows-----
      FLX.RemoveItem (irow)
      If FLX.Rows <> 2 Then
```

```
            FLX.RemoveItem (iCheck - 1)
         Else
            Exit For
         End If
         irow = 0
      Next irow
      '-----Reset all CHT's & FLX's-----
      FLX.Rows = 2
      FLX.Clear
      FLX.Visible = False
      FRMHost.chkRaw_Resample.Value = 0
      FRMHost.chkSave.Visible = True
      FRMHost.chkSave.Value = 0
      FRMHost.chkRaw_Resample.Visible = False

      '----Refresh Pat FLX-----
      Call Display_Data
      FRMHost.CHKRaw_Click
      '----Clean Up---
      Set FLX = Nothing

End Sub
Sub Resize_Array(strName, iStart, iEnd, FirstStream As Boolean)
      '-----Init Vars-----
      Dim iPat As Integer, iSize_New As Integer, iSize_Old As Integer
      Dim idata As Long, Current_Pat As Integer

      iSize_New = iEnd - iStart - 1

      '---Init Arrays------
      ReDim arrcopy(iSize_New)
      '----Find Correct Part of Array-----
      For iPat = 0 To UBound(arrRaw)
         If arrRaw(0, iPat) = strName Then
            Current_Pat = iPat
            If FirstStream = True Then
               Current_Pat = Current_Pat + Dupe_data_Start
            End If
            Exit For
         End If
      Next iPat


      '---Get/Set Size---
      iSize_Old = arrRaw(2, iPat)
      arrRaw(2, iPat) = iSize_New + 1

      '--Copy Relevant Data--
      For idata = 0 To UBound(arrcopy)
         arrcopy(idata) = arrRaw(idata + 3 + iStart, Current_Pat)
      Next idata

      '---Clear Old Data---
      For idata = 3 To (iSize_Old + 3)
         arrRaw(idata, Current_Pat) = ""
      Next idata

      '---Update With New Data---
      For idata = 0 To UBound(arrcopy)
         arrRaw(idata + 3, Current_Pat) = arrcopy(idata)
      Next idata


End Sub
```

```
Option Explicit

Function SEM(ByRef arrMPA_Ave(), ByRef n As Integer) As Variant
   '==========Init Vars============
   Dim iRange As Integer, idata As Long, iNo As Integer
   ReDim arrMpa_SEM(UBound(arrMPA_Ave) - (n - 1)) As Double
   Dim Total As Double, MeanX As Double, Count As Integer, XmeanX As Double
   Dim XMeanXsqd As Double, SD As Double

   '======Work out Range of Mpa====
   iRange = Round((n - 1) / 2)

   '========Work Out SEM==========
   For idata = iRange To UBound(arrMPA_Ave) - iRange


      '-----Add Data up-----
      For iNo = -iRange To iRange
         Total = Total + arrMPA_Ave(iNo + idata, 1)
      Next iNo
      '-----Find Mean------
      MeanX = Total / n
      Total = 0
      XMeanXsqd = 0

      '-----now do SD calculation using the mean------
      '--------SD= sum (x- meanx)^2/N-1---------------
      For Count = -iRange To iRange
         XmeanX = (MeanX - arrMPA_Ave(idata - Count, 1)) * (MeanX - arrMPA_Ave(idata - Count, 1))

         XMeanXsqd = XMeanXsqd + XmeanX

      Next Count

      SD = Sqr((XMeanXsqd) / n)

      '#Assign SEM to Array#
      arrMpa_SEM((idata - iRange)) = SD / Sqr(n)

   Next idata

   '=========Return Array============
   SEM = arrMpa_SEM()

End Function
Function MergeSEM(arrMPA_Ave() As Variant, arrMpa_SEM() As Double, ByRef n As Integer) As Variant

   '============Init Vars===========
   Dim Total_NO_SEM_Data As Integer, Total_SEM_Data As Long, iRange As Integer, idata As Long
   Dim iCount As Long

   Total_SEM_Data = UBound(arrMpa_SEM) * 5
   Total_NO_SEM_Data = n - 2
   ReDim arrMPA_ALL(Total_SEM_Data + Total_NO_SEM_Data, 1) As Double

   '======Work out Range of Mpa=====
   iRange = CInt(Round((n - 1) / 2))

   '========Build Array of Merged Data======
   For idata = iRange To UBound(arrMPA_ALL) - iRange Step 5

      '#Time#
      arrMPA_ALL(idata, 0) = (iCount + iRange) * Time_Val
      arrMPA_ALL(idata + 1, 0) = (iCount + iRange) * Time_Val
      arrMPA_ALL(idata + 2, 0) = (iCount + iRange) * Time_Val
      arrMPA_ALL(idata + 3, 0) = (iCount + iRange) * Time_Val
      arrMPA_ALL(idata + 4, 0) = (iCount + iRange) * Time_Val

      '#Data#
      arrMPA_ALL(idata, 1) = arrMPA_Ave(iCount + iRange, 1)
      arrMPA_ALL(idata + 1, 1) = arrMPA_Ave(iCount + iRange, 1) + arrMpa_SEM(iCount)
      arrMPA_ALL(idata + 2, 1) = arrMPA_Ave(iCount + iRange, 1)
      arrMPA_ALL(idata + 3, 1) = arrMPA_Ave(iCount + iRange, 1) - arrMpa_SEM(iCount)
      arrMPA_ALL(idata + 4, 1) = arrMPA_Ave(iCount + iRange, 1)

      iCount = iCount + 1
   Next idata
```

```
   '=======Insert First & Last Non Sem Values=====
   '#First#
   For idata = 0 To iRange - 1 '/ 2
      arrMPA_ALL(idata, 0) = idata * Time_Val
      arrMPA_ALL(idata, 1) = arrMPA_Ave(idata, 1)
   Next idata
   '#Last#
   For idata = 0 To iRange - 1 '/ 2
      arrMPA_ALL(UBound(arrMPA_ALL) - idata, 0) = (UBound(arrMPA_Ave) - (idata + 1)) * Time_Val
      arrMPA_ALL(UBound(arrMPA_ALL) - idata, 1) = arrMPA_Ave(UBound(arrMPA_Ave) - idata, 1)
   Next idata
   '=============================================

   '=========Return Array============
   MergeSEM = arrMPA_ALL()

End Function

Function SEM_Group(ByRef arrData() As Variant, ByRef RealCheckArray() As Variant) As Variant
   '================Init Vars==============
   Dim Pat_Total As Integer, Point_Counter As Integer, Pat_Counter As Integer
   Dim SumX As Double, n As Integer
   Dim MeanX As Double, X As Double
   Dim XmeanX As Double
   Dim SumXmeanX As Double
   Dim XMeanXsqd As Double
   Dim SD As Double
   Dim SEM As Double
   ReDim SEMArray(UBound(RealCheckArray())) As Variant


   '=========Work out Number of Datasets used=======
   Pat_Total = UBound(arrData, 3) 'UBound(RealCheckArray, 1)
   'If Duplicate = True Then
   '    N = (Pat_Total + 1) * 2
   'Else
   n = Pat_Total + 1
   'End If
   '========Work Out SEM==========
   For Point_Counter = 0 To UBound(RealCheckArray)
      If RealCheckArray(Point_Counter, 2) = "Real" Then

         '-------Find Mean-----
         For Pat_Counter = 0 To UBound(arrData, 3)
            SumX = SumX + IIf(arrData(Point_Counter, 1, Pat_Counter) = "", 0, arrData(Point_Counter, 1,
»Pat_Counter))

            If Duplicate = True Then
               If IsNumeric(arrData(Point_Counter, 2, Pat_Counter)) = True Then
                  SumX = SumX + arrData(Point_Counter, 2, Pat_Counter)
               End If
            End If

         Next Pat_Counter

         MeanX = SumX / n


         '----Reset Vars----
         SumX = 0
         XMeanXsqd = 0

         '-----Find SD-------
         'SD= sum (x- meanx)^2/N-1
         For Pat_Counter = 0 To UBound(arrData, 3)
            If Duplicate = True And IsEmpty(arrData(Point_Counter, 2, Pat_Counter)) = False And IsNumeric(
»arrData(Point_Counter, 2, Pat_Counter)) Then
               X = (CDbl(arrData(Point_Counter, 1, Pat_Counter)) + CDbl(arrData(Point_Counter, 2,
»Pat_Counter))) / 2
            Else
               X = CDbl(IIf(arrData(Point_Counter, 1, Pat_Counter) = "", 0, arrData(Point_Counter, 1,
»Pat_Counter)))
            End If

            XmeanX = ((MeanX - X) _
```

```
                * (MeanX - X))

            XMeanXsqd = XMeanXsqd + XmeanX

        Next Pat_Counter

        SD = Sqr((XMeanXsqd) / n)

        '------Store SEM------
        SEMArray(Point_Counter) = SD / Sqr(n)

      End If
   Next Point_Counter
   '-----------------------------

   SEM_Group = SEMArray()

End Function

Function MergeSEM_Group(ByRef SEMArray(), ByRef RealCheckArray()) As Variant
   '================Init Vars===============
   Dim iPower As Integer, iPeriod As Integer, idata As Long
   Dim Max_Data As Integer

   '==========Define Maximum Array Size========
   Max_Data = ((UBound(RealCheckArray) + 1) * 5) - 1 'number of data points
   ReDim arrMerged(Max_Data) As Double
   ReDim arrMerged_BIG(Max_Data, 1) As Variant

   '============Merge Sem if Present==============
   idata = 0
   For iPower = 0 To UBound(RealCheckArray)

      arrMerged(idata) = RealCheckArray(iPower, 1) 'put on value


      If RealCheckArray(iPower, 2) = "Real" Or (IsEmpty(RealCheckArray(iPower, 2)) = False And IsNumeric(
»RealCheckArray(iPower, 2)) = True) Then
         arrMerged(idata + 1) = RealCheckArray(iPower, 1) + SEMArray(iPower) 'put on upper sem
      Else
         arrMerged(idata + 1) = RealCheckArray(iPower, 1)
      End If


      arrMerged(idata + 2) = RealCheckArray(iPower, 1) ' put on value


      If RealCheckArray(iPower, 2) = "Real" Or (IsEmpty(RealCheckArray(iPower, 2)) = False And IsNumeric(
»RealCheckArray(iPower, 2)) = True) Then
         arrMerged(idata + 3) = RealCheckArray(iPower, 1) - SEMArray(iPower) 'put on lower sem
      Else
         arrMerged(idata + 3) = RealCheckArray(iPower, 1)
      End If


      arrMerged(idata + 4) = RealCheckArray(iPower, 1) 'put on value


      idata = idata + 5
   Next iPower

   idata = 0
   For iPeriod = 0 To UBound(RealCheckArray)
      arrMerged_BIG(idata, 1) = arrMerged(idata)
      arrMerged_BIG(idata + 1, 1) = arrMerged(idata + 1)
      arrMerged_BIG(idata + 2, 1) = arrMerged(idata + 2)
      arrMerged_BIG(idata + 3, 1) = arrMerged(idata + 3)
      arrMerged_BIG(idata + 4, 1) = arrMerged(idata + 4)
      arrMerged_BIG(idata, 0) = RealCheckArray(iPeriod, 0)
      arrMerged_BIG(idata + 1, 0) = RealCheckArray(iPeriod, 0)
      arrMerged_BIG(idata + 2, 0) = RealCheckArray(iPeriod, 0)
      arrMerged_BIG(idata + 3, 0) = RealCheckArray(iPeriod, 0)
      arrMerged_BIG(idata + 4, 0) = RealCheckArray(iPeriod, 0)
      idata = idata + 5
   Next iPeriod
```

```
    MergeSEM_Group = arrMerged_BIG()

End Function
```

```
Option Explicit
Option Base 0

Function DET_Cubic_spline(arrDetrend_Start() As Variant) As Variant
   '-init vars-
   ReDim arrX(UBound(arrDetrend_Start)) As Double
   ReDim arrY(UBound(arrDetrend_Start)) As Double
   Dim i As Integer
   Dim arrTmp() As String

   '---Fill Arrays----
   For i = 0 To UBound(arrDetrend_Start)
      arrTmp = Split(arrDetrend_Start(i, 0), " ")
      arrX(i) = CDbl(arrTmp(0))
      arrY(i) = arrDetrend_Start(i, 1)
   Next i

   '-Detrend array-
   For i = 0 To UBound(arrDetrend_Start)
      arrTmp = Split(arrDetrend_Start(i, 0), " ")
      Debug.Print (cubspline(1, CDbl(arrTmp(0) - 1), arrX, arrY) & vbCrLf)
   Next i

End Function
Function cubspline(Metode As Integer, Xi As Double, xx() As Double, yy() As Double) As Double
   '#####################################
   'Metode 1 = original spline
   'Metode 3 = special one
   'Xi = X val wanting a Y val for
   'xx = array of x coords
   'yy = array of y coords

   '#####################################
   Dim i As Integer
   Dim Yi As Double
   Dim X() As Double
   Dim Y() As Double
   Dim y2() As Double
   Dim j As Integer

   If Metode = 1 Then
      'Numerical Recipes are 1 based
      j = 0
   Else
      'Others are 0 based
      j = -1
   End If

   For i = 1 To UBound(xx())
      'If yy(i) <> "" Then
      j = j + 1
      ReDim Preserve X(j)
      ReDim Preserve Y(j)
      X(j) = CDbl(xx(i))
      Y(j) = CDbl(yy(i))
      'End If
   Next i

   If Metode = 1 Then
      'NR cubic spline
      'Get y2
      ReDim y2(1 To UBound(X()))
      ' Call spline(x(), y(), UBound(x()), 10 ^ 30, 10 ^ 30, y2())
      Call spline(X(), Y(), UBound(X()), Y(0), Y(UBound(Y)), y2())
      'Get y
      Call splint(X(), Y(), y2(), UBound(X()), Xi, Yi)
   ElseIf Metode = 3 Then
      'Own cubic spline
      Yi = SplineX3(Xi, X(), Y())
   End If

   'Return
   cubspline = Yi


End Function
```

```
Sub spline(X() As Double, Y() As Double, n As Integer, yp1 As Double, ypn As Double, y2() As Double)

   'Given arrays x(1:n) and y(1:n) containing a tabulated function, i.e., y i = f(xi), with
   'x1<x2< :::<xN , and given values yp1 and ypn for the first derivative of the inter-
   'polating function at points 1 and n, respectively, this routine returns an array y2(1:n) of
   'length n which contains the second derivatives of the interpolating function at the tabulated
   'points xi. If yp1 and/or ypn are equal to 1 * 10^30 or larger, the routine is signaled to set
   'the corresponding boundary condition for a natural spline, with zero second derivative on
   'that boundary.
   'Parameter: NMAX is the largest anticipated value of n.

   Dim Nmax As Integer
   Nmax = 500

   Dim i As Integer
   Dim k As Integer

   Dim p As Double
   Dim qn As Double
   Dim sig As Double
   Dim un As Double
   ReDim U(Nmax) As Double

   'The lower boundary condition is set either to be natural
   If (yp1 > 9.9E+29) Then
      y2(1) = 0#
      U(1) = 0#
   Else
      'or else to have a specicied first derivative.
      y2(1) = -0.5
      U(1) = (3# / (X(2) - X(1))) * ((Y(2) - Y(1)) / (X(2) - X(1)) - yp1)
   End If

   'This is the decomposition loop of the tridiagonal
   'algorithm. y2 and u are used for temporary
   'storage of the decomposed factors.

   For i = 2 To n - 1
      sig = (X(i) - X(i - 1)) / (X(i + 1) - X(i - 1))
      p = sig * y2(i - 1) + 2#
      y2(i) = (sig - 1#) / p
      U(i) = (6# * ((Y(i + 1) - Y(i)) / (X(i + 1) - X(i)) - (Y(i) - Y(i - 1)) / (X(i) - X(i - 1))) / (X(i +
» 1) - X(i - 1)) - sig * U(i - 1)) / p
   Next i

   'The upper boundary condition is set either to be natural
   If (ypn > 9.9E+29) Then
      qn = 0#
      un = 0#
   Else
      'or else to have a specified first derivative.
      qn = 0.5
      un = (3# / (X(n) - X(n - 1))) * (ypn - (Y(n) - Y(n - 1)) / (X(n) - X(n - 1)))
   End If
   y2(n) = (un - qn * U(n - 1)) / (qn * y2(n - 1) + 1#)

   'This is the backsubstitution loop of the tridiagonal algorithm.
   For k = n - 1 To 1 Step -1
      y2(k) = y2(k) * y2(k + 1) + U(k)
   Next k

End Sub

Sub splint(xa() As Double, ya() As Double, y2a() As Double, n As Integer, X As Double, Y As Double)

   'Given the arrays xa(1:n) and ya(1:n) of length n, which tabulate a function (with the
   'xai 's in order), and given the array y2a(1:n), which is the output from spline above,
   'and given a value of x, this routine returns a cubic-spline interpolated value y.

   Dim k As Integer
   Dim khi As Integer
   Dim klo As Integer

   Dim A As Double
   Dim B As Double
   Dim h As Double
```

```
   'We will the right place in the table by means of bisection.
   klo = 1
   khi = n

   While (khi - klo > 1)
      k = (khi + klo) / 2
      If (xa(k) > X) Then
         khi = k
      Else
         klo = k
      End If
   Wend

   'klo and khi now bracket the input value of x.
   'h = xa(khi) - xa(klo)
   h = CInt((n + 1) / 2)
   If (h = 0) Then MsgBox ("bad xa input in splint")

   'Cubic spline polynomial is now evaluated.
   A = (xa(khi) - X) / h
   B = (X - xa(klo)) / h
   Y = A * ya(klo) + B * ya(khi) + ((A ^ 3 - A) * y2a(klo) + (B ^ 3 - B) * y2a(khi)) * (h ^ 2) / 6#

End Sub

Public Function SplineX3(X As Double, xx() As Double, yy() As Double) As Double
   '|-------------------------------------------------------------------------------
   '| Function returns y value for a corresponding x value, based on cubic spline.
   '| Will never oscillates or overshoot. No need to solve matrix.
   '| Also calculate constants for cubic in case needed (for integration).
   '|
   '| xx(0 to No_of_lines) is x values
   '|    * Must be unique (no two consequetive ones the same)
   '|    * Must be in ascending order
   '|    * No of lines = Number of points - 1
   '| yy(0 to No_of_lines) is y values
   '|
   '| Uses function dxx to prevent div by zero.
   '
   '|-------------------------------------------------------------------------------

   Dim i As Integer
   Dim j As Integer
   Dim Nmax As Integer
   Dim Num As Integer

   '1st and 2nd derivative for left and right ends of line
   Dim gxx(0 To 1) As Double
   Dim ggxx(0 To 1) As Double

   'Constants for cubic equations
   Dim A As Double      'Also for linear extrapolation
   Dim B As Double      'Also for linear extrapolation
   Dim C As Double
   Dim D As Double

   'Number of lines = points - 1
   Nmax = UBound(xx())

   '(1a) Find LineNumber or segment. Linear extrapolate if outside range.
   Num = 0
   If X < xx(0) Or X > xx(Nmax) Then
      'X outisde range. Linear interpolate
      'Below min or max?
      If X < xx(0) Then Num = 1 Else Num = Nmax
      B = (yy(Num) - yy(Num - 1)) / dxx(xx(Num), xx(Num - 1))
      A = yy(Num) - B * xx(Num)
      SplineX3 = A + B * X
      Exit Function

      '(1b) Find LineNumber or segment. Linear extrapolate if outside range.
   Else
      'X in range. Get line.
      For i = 1 To Nmax
         If X <= xx(i) Then
            Num = i
            Exit For
         End If
```

```
      Next i
   End If

   '(2) Calc first derivative (slope) for intermediate points

   For j = 0 To 1          'Two points around line
      'i = Num - 1 + j

      If i = 0 Or i = Nmax Then
         'Set very large slope at ends
         gxx(j) = 10 ^ 30
      ElseIf (yy(i + 1) - yy(i) = 0) Or (yy(i) - yy(i - 1) = 0) Then
         'Only check for 0 dy. dx assumed NEVER equals 0 !
         gxx(j) = 0
      ElseIf ((xx(i + 1) - xx(i)) / (yy(i + 1) - yy(i)) + (xx(i) - xx(i - 1)) / (yy(i) - yy(i - 1))) = 0
»Then
         'Pos PLUS neg slope is 0. Prevent div by zero.
         gxx(j) = 0
      ElseIf (yy(i + 1) - yy(i)) * (yy(i) - yy(i - 1)) < 0 Then
         'Pos AND neg slope, assume slope = 0 to prevent overshoot
         gxx(j) = 0
      Else
         'Calculate an average slope for point based on connecting lines
         gxx(j) = 2 / (dxx(xx(i + 1), xx(i)) / (yy(i + 1) - yy(i)) + dxx(xx(i), xx(i - 1)) / (yy(i) - yy(i
»- 1)))
      End If
   Next j

   '(3) Reset first derivative (slope) at first and last point
   If Num = 1 Then
      'First point has 0 2nd derivative
      gxx(0) = 3 / 2 * (yy(Num) - yy(Num - 1)) / dxx(xx(Num), xx(Num - 1)) - gxx(1) / 2
   End If
   If Num = Nmax Then
      'Last point has 0 2nd derivative
      gxx(1) = 3 / 2 * (yy(Num) - yy(Num - 1)) / dxx(xx(Num), xx(Num - 1)) - gxx(0) / 2
   End If

   '(4) Calc second derivative at points
   ggxx(0) = -2 * (gxx(1) + 2 * gxx(0)) / dxx(xx(Num), xx(Num - 1)) + 6 * (yy(Num) - yy(Num - 1)) / dxx(xx(
»Num), xx(Num - 1)) ^ 2
   ggxx(1) = 2 * (2 * gxx(1) + gxx(0)) / dxx(xx(Num), xx(Num - 1)) - 6 * (yy(Num) - yy(Num - 1)) / dxx(xx(
»Num), xx(Num - 1)) ^ 2

   '(5) Calc constants for cubic
   D = 1 / 6 * (ggxx(1) - ggxx(0)) / dxx(xx(Num), xx(Num - 1))
   C = 1 / 2 * (xx(Num) * ggxx(0) - xx(Num - 1) * ggxx(1)) / dxx(xx(Num), xx(Num - 1))
   B = (yy(Num) - yy(Num - 1) - C * (xx(Num) ^ 2 - xx(Num - 1) ^ 2) - D * (xx(Num) ^ 3 - xx(Num - 1) ^ 3))
»/ dxx(xx(Num), xx(Num - 1))
   A = yy(Num - 1) - B * xx(Num - 1) - C * xx(Num - 1) ^ 2 - D * xx(Num - 1) ^ 3

   'Return function
   Dim split1
   Dim split2
   Dim split3

   split1 = (B * X)
   split2 = (C * X ^ 2)
   split3 = (D * X ^ 3)
   SplineX3 = A + split1 + split2 + split3

   ''Alternative method based on Numerical Recipes.
   ''Shorter but does not calc cubic constants A, B, C, D
   'i = Num
   'A = (xx(i) - x) / (xx(i) - xx(i - 1))
   'B = 1 - A
   'Cy = 1 / 6 * (A ^ 3 - A) * (6 * (yy(i) - yy(i - 1)) - 2 * (gxx(i) + 2 * gxx(i - 1)) * (xx(i) - xx(i -
»1)))
   'Dy = 1 / 6 * (B ^ 3 - B) * (2 * (2 * gxx(i) + gxx(i - 1)) * (xx(i) - xx(i - 1)) - 6 * (yy(i) - yy(i -
»1)))
   ''Return function
   'SplineX3 = A * yy(i - 1) + B * yy(i) + Cy + Dy

End Function

Public Function dxx(x1 As Double, x0 As Double) As Double
   'Calc Xi - Xi-1 to prevent div by zero
```

```
    dxx = x1 - x0
    If dxx = 0 Then dxx = 10 ^ 30

End Function
```

EASY TSA - MODULE - MdlTime.bas **MdlTime**

```
Option Explicit

Public Sub Choose_Time()
   '=============Init Vars====================

   Dim Responce As String
   Dim TimeType As String
   Dim smell As String, smelly As String

   '===========Check that we need to Get time==========
   If FRMchoose.Time_Needed = False Then
      Time_Val = 1
      Time_Type = "Pixels"
      Exit Sub
   End If
   '==================Get Intervals=====================
   Responce = InputBox("What are the Time Intervals?", "Time Interval", 1, FRMHost.Left + (FRMHost.Width /
»2) - 2000, FRMHost.top + (FRMHost.Height / 2) - 1000)



   Do While IsNumeric(Responce) = False
      smell = MsgBox("Invalid Time Entry", vbCritical, "Error in Time Range")
      Responce = InputBox("What are the NEW Time Intervals?", "Time Interval", 1, FRMHost.Left + (
»FRMHost.Width / 2) - 2000, FRMHost.top + (FRMHost.Height / 2) - 1000)
   Loop
   Time_Val = Responce

   '================Get Denomination==================
   TimeType = InputBox("Of what type are the intervals? (Hours/Mins/Secs)", "Time Type", "Minutes",
»FRMHost.Left + (FRMHost.Width / 2) - 2000, FRMHost.top + (FRMHost.Height / 2) - 1000)

   Select Case UCase(TimeType)
   Case "HOURS":    Time_Type = "Hours"
   Case "HOUR":     Time_Type = "Hours"
   Case "H":        Time_Type = "Hours"
   Case "SECS":     Time_Type = "Seconds"
   Case "SECONDS":  Time_Type = "Seconds"
   Case "SECOND":   Time_Type = "Seconds"
   Case "S":        Time_Type = "Seconds"
   Case "SEC":      Time_Type = "Seconds"
   Case "MIN":      Time_Type = "Minutes"
   Case "MINS":     Time_Type = "Minutes"
   Case "MINUTES":  Time_Type = "Minutes"
   Case "MINUTE":   Time_Type = "Minutes"
   Case "M":        Time_Type = "Minutes"
   Case Else:       Time_Type = TimeType

      'Case Else: smelly = MsgBox("Invalid Time Type (H/M/S)", vbCritical, "Error in Time Type")
      'TimeType = InputBox("Of what type are the intervals? (Hours/Mins/Secs)", "Time Type", "Minutes",
»FRMHost.Left + (FRMHost.Width / 2) - 2000, FRMHost.Top + (FRMHost.Height / 2) - 1000)
   End Select

End Sub
```

```
Option Explicit
Private Declare Function GetTickCount& Lib "kernel32" ()

Function RealRand(Lowb As Integer, upb As Integer)
   Randomize
   Dim t As Long
   Dim q As Long
   Dim p As Integer
   t = GetTickCount
   Do Until t >= Lowb And t <= upb

      q = t / (Rnd * 100)
      t = q
      DoEvents
      '  t = t / Int(upb - Lowb) * Rnd
      If t < Lowb Then
         t = upb * Rnd
      End If
   Loop

   RealRand = t
End Function

Function Find_Last_DataItem(ByRef iPat As Integer) As Long
   '=========Init Vars=========
   'iPat = iPat - 1
   Dim idata As Long, i As Integer
   Dim strName As String
   '====Get UID=====
   strName = FRMHost.FLXMain.TextMatrix(iPat, 0)

   '==========Find when data ends=======
   '---Match Name---
   Do While strName <> arrRaw(0, i)

      i = i + 1
      '---Error Check---
      If i > UBound(arrRaw, 2) Then
         MsgBox ("File has too many extra blank lines in file OR last dataset has incorrect number of data
»items specified")
         End
      End If
   Loop
   '---Find EOF Data---
   For idata = 0 To UBound(arrRaw)
      If arrRaw(idata, i) = "" Then
         Find_Last_DataItem = (idata - 1)
         Exit Function
      End If
   Next idata

End Function


Function Filter_StarsOut(ByRef arrMain(), Optional ThreeD As Boolean) As Variant
   '=========Init Vars=========
   Dim idata As Long
   Dim icol As Integer

   '=======Filter out * ========
   If ThreeD = False Then
      For icol = 0 To UBound(arrMain, 2)
         For idata = 0 To UBound(arrMain)
            If Right(arrMain(idata, icol), 1) = "*" Then
               arrMain(idata, icol) = Trim(Left(arrMain(idata, icol), Len(arrMain(idata, icol)) - 1))
            Else
               arrMain(idata, icol) = Trim(arrMain(idata, icol))
            End If
         Next idata
      Next icol
   Else
      Dim iPat As Integer
      For iPat = 0 To UBound(arrMain, 3)
         For icol = 0 To UBound(arrMain, 2)
            For idata = 0 To UBound(arrMain)
               If Right(arrMain(idata, icol, iPat), 1) = "*" Then
```

```
                arrMain(idata, icol, iPat) = Left(arrMain(idata, icol, iPat), Len(arrMain(idata, icol,
»iPat)) - 1)
              End If
            Next idata
         Next icol
      Next iPat
   End If
   Filter_StarsOut = arrMain()
End Function


Sub Refresh_All(strSender As String)
   '====================Init Vars==================
   Dim FRM As Form, i As Integer
   Set FRM = FRMHost

   '====Send to Correct PArt======
   Select Case UCase(strSender)
   Case "CHANGE IN ARRAY": GoTo Raw
   Case "RAW": GoTo Mpa_Det
   Case "MPA": GoTo Det
   Case "DETRENDED": GoTo All
   Case "FOURIER": GoTo Power
   Case Else: Exit Sub

   End Select

   '=======Re-Click All Selected Chk's and opt's============
Raw:
   If FRM.chkMean.Value = 1 Then
      Call FRM.chkMean_Click
   End If

Mpa_Det:
   For i = 1 To FRM.OptMpa.Count - 1
      If FRM.OptMpa(i).Value = True Then Call FRM.OptMpa_Click(i): Exit For
   Next i



Det:
   For i = 1 To FRM.OptDetrend.Count - 1
      If FRM.OptDetrend(i).Value = True Then Call FRM.OptDetrend_Click(i): Exit For
   Next i


All:
   If FRM.CHKFourier.Value = 1 Then
      Call FRM.ChkFourier_Click
   End If

   If FRM.CHKAutocorrelate.Value = 1 Then
      Call FRM.CHKAutocorrelate_Click
   End If

   If FRM.CHKDeConvolution.Value = 1 Then
      Call FRM.CHKDeConvolution_Click
   End If

   If FRM.CHKSerialArrayAveraging.Value = 1 Then
      Call FRM.CHKSerialArrayAveraging_Click
   End If

   If FRM.CHKObservedConcentration.Value = 1 Then
      Call FRM.CHKObservedConcentration_Click
   End If



Power:
   If FRM.CHKPowerPeriod.Value = 1 Then
      Call FRM.CHKPowerPeriod_Click
   End If


   '----Zorder the Chts-----
   If FRMHost.Tabmain.Tabs.Count <> 0 Then
```

```
      Dim Curr_Name As String
      Curr_Name = FRMHost.Tabmain.Tabs(FRMHost.Tabmain.Tabs.Count).Key
      Call PositionControls(Curr_Name)
   End If
End Sub

Sub Select_Tab(ByRef Type_Name As String)
   Dim i As Integer
   '-------Select Tab----------
   For i = 1 To FRMHost.Tabmain.Tabs.Count
      If FRMHost.Tabmain.Tabs(i).Image = Type_Name Then
         FRMHost.Tabmain.Tabs(i).Selected = True
         Exit For
      End If
   Next i
   '-------------------------
End Sub

Function What_Flx(Optional Choice As String) As Variant
   '================Init Vars==================
   Dim iTab As Integer, MPA_Found As Boolean, Det_Found As Boolean, FLX As MSFlexGrid
   Dim idata As Long, Lanes As Integer, iLanes As Integer, Mean_Found As Boolean

   Lanes = 1
   '==========Find if Mpa or Det tab exists=======
   If Choice = "" Then
      For iTab = 1 To FRMHost.Tabmain.Tabs.Count
         If FRMHost.Tabmain.Tabs(iTab).Key = "Mpa" Then
            MPA_Found = True
         ElseIf FRMHost.Tabmain.Tabs(iTab).Key = "Detrended" Then
            Det_Found = True
         ElseIf FRMHost.Tabmain.Tabs(iTab).Key = "Mean_SEM" Then
            Mean_Found = True
         End If
      Next iTab
   Else
      If Choice = "Mpa" Then
         MPA_Found = True
      ElseIf Choice = "Detrended" Then
         Det_Found = True
      ElseIf Choice = "Mean_SEM" Then
         Mean_Found = True
      End If
   End If
   '======Decide where to get data from====
   If Det_Found = True Then
      Set FLX = FRMHost.Controls("FLXdetrended")

   ElseIf MPA_Found = True Then
      Set FLX = FRMHost.Controls("FLXmpa")

   ElseIf Mean_Found = True Then
      Set FLX = FRMHost.Controls("FLXMean_SEM")
   Else
      Set FLX = FRMHost.Controls("FLXRaw")

      If Duplicate = True Then
         Lanes = 2
      End If
   End If

   '=========Init Storage Array==========

   ReDim arrData(FLX.Rows - 2, Lanes) As Variant

   For idata = 0 To UBound(arrData)
      For iLanes = 0 To Lanes
         arrData(idata, iLanes) = FLX.TextMatrix((idata + 1), iLanes)
         If FLX.TextMatrix((idata + 1), iLanes) = "" Then
            'Debug.Print
            arrData(idata, iLanes) = 0
         End If
      Next iLanes
   Next idata


   What_Flx = Filter_StarsOut(arrData())
End Function
```

```
Function What_CHT_Type(ByRef Id As Integer) As String

   Select Case Id
   Case 3: What_CHT_Type = excel.xlLine
   Case 16: What_CHT_Type = excel.xlXYScatterSmoothNoMarkers
   Case 1: What_CHT_Type = xlColumnClustered
   End Select

End Function

Function What_Tab() As String
   Dim i As Integer
   For i = 1 To FRMHost.Tabmain.Tabs.Count
      If FRMHost.Tabmain.Tabs(i).Selected = True Then
         What_Tab = FRMHost.Tabmain.Tabs(i).Key
         Exit For
      End If
   Next i
End Function


Sub Swap_Selected_Pats(ByRef Swap_to_Main As Boolean)
   '==============Init Vars===============
   Dim FLX As MSFlexGrid, FLXGrouped As MSFlexGrid, FLXMain2Grouped As MSFlexGrid
   Dim i As Integer, EndofArrayFLX As Long, EndofArrayGrouped As Long, irow As Integer, jRow As Integer
   Dim jCol As Integer

   Set FLX = FRMHost.FLXMain
   Set FLXGrouped = FRMHost.FLXGrouped
   Set FLXMain2Grouped = FRMHost.FLXMain2Grouped
   '---set global flag to disable cht update---
   SWAPPING_FLX = True

   If Swap_to_Main = True Then
      If FLXGrouped.TextMatrix(1, 0) <> "" Then
         '=========Swap Contents of Flexigrids over=======
         'We need to swap the contents as flxmain is the
         'one preocessed by Interpolate
         'main to temp, grp to main

         '-------Define Temp FLX-------
         FLXMain2Grouped.Cols = FLX.Cols
         FLXMain2Grouped.Rows = FLX.Rows
         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

         '----------Main FLX to Temp--------
         EndofArrayFLX = FLX.Cols * FLX.Rows - 2
         For i = 0 To EndofArrayFLX
            FLXMain2Grouped.TextArray(i) = FLX.TextArray(i)
         Next i
         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

         '------Define FLX Main again----
         FLX.Cols = FLXGrouped.Cols
         FLX.Rows = FLXGrouped.Rows
         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         FLX.Clear
         '---------Grouped FLX to Main--------------
         EndofArrayGrouped = FLXGrouped.Cols * FLXGrouped.Rows - 2
         For i = 0 To EndofArrayGrouped
            FLX.TextArray(i) = FLXGrouped.TextArray(i)
         Next i
         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         '===============================================
      End If
      FLX.RowSel = 1
      FLX.Row = 1
      'Call Draw_Raw
   Else
      If FLXGrouped.TextMatrix(1, 0) <> "" Then
         '=========Swap Contents of Flexigrids back=======
         'We need to swap the contents as flxmain is the
         'one preocessed by Interpolate
         'temp to main

         '------Define FLX Main again----
         FLX.Cols = FLXMain2Grouped.Cols
```

```
         FLX.Rows = FLXMain2Grouped.Rows
         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         FLX.Clear
         '--------- Temp to Main--------------
         EndofArrayGrouped = FLX.Cols * FLX.Rows - 2
         For i = 0 To EndofArrayGrouped

            FLX.TextArray(i) = FLXMain2Grouped.TextArray(i)
         Next i
         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

         '---------Colour Grouped ones--------
         jRow = 1
         For irow = 1 To FLXGrouped.Rows - 2
            For jRow = 1 To FLX.Rows - 2
               If FLX.TextMatrix(jRow, 0) = FLXGrouped.TextMatrix(irow, 0) Then
                  If FLX.TextMatrix(jRow, 1) = FLXGrouped.TextMatrix(irow, 1) Then
                     If FLX.TextMatrix(jRow, 2) = FLXGrouped.TextMatrix(irow, 2) Then
                        FLX.Row = jRow
                        For jCol = 0 To FLX.Cols - 1
                           FLX.Col = jCol
                           FLX.CellBackColor = Highlight
                        Next jCol
                     End If
                  End If
               End If
            Next jRow
         Next irow

         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         '===================================================
      End If
   End If

   '---set global flag to disable cht update---
   SWAPPING_FLX = False
End Sub

Function Floor(dblIn As Double) As Long
   '========Rounds Down========
   If (dblIn - CLng(dblIn)) < 0 Then
      Floor = CLng(dblIn - 1)
   Else
      Floor = CLng(dblIn)
   End If
End Function

Function Ceiling(dblIn As Double) As Integer
   '=========Rounds Up=========

   If (dblIn - CInt(dblIn)) = 0.5 Then
      'rounds up 0.5 values as VB sucks @ rounding up 0.5's
      Ceiling = CInt(dblIn + 1) - 1

   ElseIf (dblIn - CInt(dblIn)) > 0 Then
      'rounds up all the values below 0.5
      Ceiling = CInt(dblIn + 1)

   Else
      'just rounds rest over 0.5
      Ceiling = CInt(dblIn)

   End If
End Function


Function Bubble_Sort(ByRef arrStart() As String) As Variant
   '-----Init Vars-----
   Dim idata As Long, Value_data As Double
   Dim iloop As Integer


   '-------Sort--------
   For iloop = 0 To UBound(arrStart) - 1
      For idata = 0 To UBound(arrStart) - 1
         Value_data = arrStart(idata)
         If Value_data > arrStart(idata + 1) Then
            '-----Swap----
```

```
            arrStart(idata) = arrStart(idata + 1)
            arrStart(idata + 1) = Value_data

         End If

      Next idata
   Next iloop

   '------Return-------
   Bubble_Sort = arrStart()

End Function

Function Copy_Control(ByVal cnt_Original As Control)
   Set Copy_Control = cnt_Original

End Function

Public Function TrimALL(ByVal TextIN As String, Optional NonPrints As Boolean) As String



   TrimALL = Trim(TextIN)

   If NonPrints Then
      Dim X As Long
      ' remove all non-printable characters
      While InStr(TrimALL, vbCrLf) > 0
         TrimALL = Replace(TrimALL, vbCrLf, " ")
      Wend

      While InStr(TrimALL, vbTab) > 0
         TrimALL = Replace(TrimALL, vbTab, " ")
      Wend

      For X = 0 To 31
         While InStr(TrimALL, Chr(X)) > 0
            TrimALL = Replace(TrimALL, Chr(X), " ")
         Wend
      Next X

      For X = 127 To 255
         While InStr(TrimALL, Chr(X)) > 0
            TrimALL = Replace(TrimALL, Chr(X), " ")
         Wend
      Next X
   End If

   While InStr(TrimALL, String(2, " ")) > 0
      TrimALL = Replace(TrimALL, String(2, " "), " ")
   Wend

End Function

Function Get_Least() As Integer
   Dim FLX As MSFlexGrid
   Dim Least As Integer, irow As Integer
   Set FLX = FRMHost.FLXMain
   '-Get Least Datapoints-
   Least = FLX.TextMatrix(1, 2)
   For irow = 1 To FLX.Rows - 2
      If FLX.TextMatrix(irow, 2) < Least Then
         Least = FLX.TextMatrix(irow, 2)
      End If
   Next irow

   Get_Least = Least
End Function

Sub Add_Options()
   If FRMHost.MNURand.Checked = True Then
      Call Calculate_Randomness
   End If

   If FRMHost.MNUArea.Checked = True Then
      Call AreaUnderCurve
   End If
```

```
   If FRMHost.MNUGA.Checked = True Then
      Call Display_GA_Value
   End If
End Sub

Static Function Log10(X)
   Log10 = Log(X) / Log(10#)
End Function

Function FormatSF(dblInput As Double, intSF As Integer) As String

   Dim intCorrPower As Integer         'Exponent used in rounding calculation
   Dim intSign As Integer              'Holds sign of dblInput since logs are used in calculations

   '-- Store sign of dblInput --
   intSign = Sgn(dblInput)

   '-- Calculate exponent of dblInput --
   intCorrPower = Int(Log10(Abs(dblInput)))

   FormatSF = Round(dblInput * 10 ^ ((intSF - 1) - intCorrPower))   'integer value with no sig fig
   FormatSF = FormatSF * 10 ^ (intCorrPower - (intSF - 1))         'raise to original power


   '-- Reconsitute final answer --
   FormatSF = FormatSF * intSign

   '-- Answer sometimes needs padding with 0s --
   If InStr(FormatSF, ".") = 0 Then
      If Len(FormatSF) < intSF Then
         FormatSF = Format(FormatSF, "##0." & String(intSF - Len(FormatSF), "0"))
      End If
   End If

   If intSF > 1 And Abs(FormatSF) < 1 Then
      Do Until Left(Right(FormatSF, intSF), 1) <> "0" And Left(Right(FormatSF, intSF), 1) <> "."
         FormatSF = FormatSF & "0"
      Loop
   End If


End Function
```

```
Option Explicit

Public Function Non_Repeating_Numbers(ByVal lngQtyRequested As Long, _
      ByVal lngMaxSeedValue As Long, arlngMixed() As Long, _
      Optional lngBaseNumber As Long = 0, _
      Optional bSortData As Boolean = True) As Boolean

   ' ************************************************************************
   '
   '               '
   ' ========================================================================

   ' ************************************************************************

   ' ------------------------------------------------------------------------
   ' define local variables
   ' ------------------------------------------------------------------------
   Dim lngIndex      As Long
   Dim lngTmpNbr     As Long
   Dim arlngTemp()   As Long

   ' ------------------------------------------------------------------------
   ' Initialize variables.
   ' ------------------------------------------------------------------------
   Erase arlngMixed()                  ' Verify this array is empty
   Erase arlngTemp()                   ' Verify this array is empty
   lngTmpNbr = 0

   ' ------------------------------------------------------------------------
   ' Test data input.  Leave if nothing to work with.
   ' ------------------------------------------------------------------------
   If lngQtyRequested <= 0 Or _
         lngMaxSeedValue <= 0 Then
      MsgBox "Cannot process values less than one.", _
            vbInformation + vbOKOnly, "Non-Repeating Numbers"
      Non_Repeating_Numbers = False
      Exit Function
   End If

   ' ------------------------------------------------------------------------
   ' Test the lowest possible number
   ' ------------------------------------------------------------------------
   If lngBaseNumber >= lngMaxSeedValue Then
      MsgBox "The lowest possible number is greater than or equal to the maximum value.", _
            vbInformation + vbOKOnly, "Non-Repeating Numbers"
      Non_Repeating_Numbers = False
      Exit Function
   End If

   ' ------------------------------------------------------------------------
   ' Test the base number
   ' ------------------------------------------------------------------------
   If lngBaseNumber > (lngMaxSeedValue - lngQtyRequested) Then
      MsgBox "The base number is greater than or equal to the maximum value.", _
            vbInformation + vbOKOnly, "Non-Repeating Numbers"
      Non_Repeating_Numbers = False
      Exit Function
   End If

   ' ------------------------------------------------------------------------
   ' Test data input.  Leave if quantity requested is greater than the amount
   ' to work with.
   ' ------------------------------------------------------------------------
   If lngQtyRequested > lngMaxSeedValue Then
      MsgBox "Quantity requested is greater than the value to choose from.", _
            vbInformation + vbOKOnly, "Non-Repeating Numbers"
      Non_Repeating_Numbers = False
      Exit Function
   End If

   ' ------------------------------------------------------------------------
   ' Resize arrays based on values passed to this routine.
   ' ------------------------------------------------------------------------
   ReDim arlngMixed(1 To lngQtyRequested)  ' number of elements to be returned
   ReDim arlngTemp(lngMaxSeedValue)        ' number of elements to pick from

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

```
   ' If the user only wants 1 number returned then random generate for five
   ' interations and return the last value generated.
   ' ------------------------------------------------------------------------
   If lngQtyRequested = 1 Then
      For lngIndex = 1 To 5
         arlngMixed(1) = Int(Rnd2(CSng(lngBaseNumber), CSng(lngMaxSeedValue)))
      Next
      Non_Repeating_Numbers = True
      Exit Function
   End If

   ' ------------------------------------------------------------------------
   ' Preload the seed array.  These are the values to choose from.
   ' ------------------------------------------------------------------------
   For lngIndex = 0 To lngMaxSeedValue
      arlngTemp(lngIndex) = lngIndex   ' load seed array
   Next

   ' ------------------------------------------------------------------------
   ' Mix the numbers
   ' ------------------------------------------------------------------------
   arlngTemp = Mix_The_Numbers(arlngTemp())

   ' ------------------------------------------------------------------------
   ' Transfer the data from the array that was just created back to the original
   ' array.
   ' ------------------------------------------------------------------------
   For lngIndex = 0 To lngMaxSeedValue - 1
      If arlngTemp(lngIndex) >= lngBaseNumber Then
         ' if the value that was mixed is greater than
         ' or equal to the base value requested then
         ' add it to the output array
         lngTmpNbr = lngTmpNbr + 1
         arlngMixed(lngTmpNbr) = arlngTemp(lngIndex)

         ' see if we have the return quantity requested
         If lngTmpNbr = lngQtyRequested Then
            Exit For
         End If
      End If
   Next

   ' ------------------------------------------------------------------------
   ' If more than one number is requested and the user wants to sort the data,
   ' we will use a Sort algorithym to sort in ascending order.
   ' ------------------------------------------------------------------------
   If bSortData Then

      Select Case lngQtyRequested

         ' do a bubble sort
      Case 2 To 25:
         For lngIndex = 1 To (lngQtyRequested - 1)
            If arlngMixed(lngIndex) > arlngMixed(lngIndex + 1) Then
               Swap_Data arlngMixed(lngIndex), arlngMixed(lngIndex + 1)
               lngIndex = 0    ' Reset the index and start over
            End If
         Next

         ' do a quicksort
      Case Is > 25:
         QuickSort arlngMixed(), LBound(arlngMixed), UBound(arlngMixed)

      Case Else:  ' fall thru and do nothing to the array
      End Select
   End If

   ' ------------------------------------------------------------------------
   ' Successful finish
   ' ------------------------------------------------------------------------
   Non_Repeating_Numbers = True

End Function

Public Function Mix_The_Numbers(arlngInput() As Long) As Variant

   ' ************************************************************************
   ' Routine:       Mix_The_Numbers
```

```
    '
    ' Description:   An array of numbers is passed to this routine.
    ' ************************************************************************

    ' ------------------------------------------------------------------------
    ' define local variables
    ' ------------------------------------------------------------------------
    Dim lngIndex1      As Long
    Dim lngIndex2      As Long
    Dim lngNewIndex    As Long
    Dim intOuterLoop   As Integer
    Dim sngMin         As Single
    Dim sngMax         As Single

    ' ------------------------------------------------------------------------
    ' Determine the low and high end of the array
    ' ------------------------------------------------------------------------
    ' Get minimum number of elements in array
    sngMin = LBound(arlngInput)

    ' Get maximum number of elements in array
    If sngMin = 0 Then
       sngMax = UBound(arlngInput) - 1
    Else
       sngMax = UBound(arlngInput)
    End If

    ' ------------------------------------------------------------------------
    ' See if anything was passed that has to be mixed
    ' ------------------------------------------------------------------------
    If sngMin >= sngMax Then
       Exit Function
    End If

    ' ------------------------------------------------------------------------
    ' Calculate the number of times the array will be mixed.
    ' ------------------------------------------------------------------------
    intOuterLoop = Int(Rnd2(5, 10))   ' create value between 5-10 inclusive

    ' ------------------------------------------------------------------------
    ' The array will now undergo multiple mixing operations in a series of
    ' loops (5 to 10 times).
    ' ------------------------------------------------------------------------
    For lngIndex1 = 1 To intOuterLoop

       ' go thru the input array and randomly pick an element to move to
       ' another position within the array.
       For lngIndex2 = sngMin To sngMax

          ' generate a new index that does mot
          ' match the current index
          Do
             lngNewIndex = Int(Rnd2(sngMin, sngMax))
          Loop Until lngNewIndex <> lngIndex2

          ' swap the the data
          Swap_Data arlngInput(lngIndex2), arlngInput(lngNewIndex)
       Next
    Next

    ' ------------------------------------------------------------------------
    ' Return the rearranged data
    ' ------------------------------------------------------------------------
    Mix_The_Numbers = arlngInput()

End Function

Public Sub QuickSort(arNumeric() As Long, lngLow As Long, lngHi As Long)

    ' ************************************************************************
    ' Routine:       QuickSort
    '
    ' Description:   This routine will accept and sort in ascending order a
    '                numeric array of data.
    ' ************************************************************************

    ' ------------------------------------------------------------------
    ' Define local variables
```

```
' ---------------------------------------------------------------------
Dim lngMidPoint  As Long      ' midpoint of the array to be sorted
Dim lngTmpLow    As Long      ' Index pointer
Dim lngTmpHi     As Long      ' Index pointer

' ---------------------------------------------------------------------
' See if this is an empty array by checking to see if there is data in
' the first element.  If not, then leave.
' ---------------------------------------------------------------------
If Len(Trim(arNumeric(1))) = 0 Then
   Exit Sub
End If

' ---------------------------------------------------------------------
' Leave if there is nothing to sort
' ---------------------------------------------------------------------
If lngLow >= lngHi Then
   Exit Sub
End If

' ---------------------------------------------------------------------
' Save the count of the minimum and maximum number of elements in the
' array to be sorted.
' ---------------------------------------------------------------------
lngTmpLow = lngLow
lngTmpHi = lngHi

' ---------------------------------------------------------------------
' Calculate the midpoint of the array
' ---------------------------------------------------------------------
lngMidPoint = arNumeric((lngLow + lngHi) / 2)

' ---------------------------------------------------------------------
' Start the sorting process
' ---------------------------------------------------------------------
While (lngTmpLow <= lngTmpHi)

   ' Always process the low end first.  Loop as long the array data
   ' element is LESS than the data in the temporary holding area
   ' and the temporary low value is LESS than the maximum number of
   ' array elements.
   While (arNumeric(lngTmpLow) < lngMidPoint And lngTmpLow < lngHi)
      lngTmpLow = lngTmpLow + 1  ' Increment the temp low counter
   Wend

   ' Now, we will process the high end.  Loop as long the data in the
   ' temporary holding area is LESS than the array data element
   ' and the temporary high value is GREATER than the minimum number
   ' of array elements.
   While (lngMidPoint < arNumeric(lngTmpHi) And lngTmpHi > lngLow)
      lngTmpHi = lngTmpHi - 1    ' Decrement the temp high counter
   Wend

   ' if the temp low end is LESS than or equal to the temp high end,
   ' then swap places
   If (lngTmpLow <= lngTmpHi) Then
      Swap_Data arNumeric(lngTmpLow), arNumeric(lngTmpHi)
      lngTmpLow = lngTmpLow + 1    ' Increment the temp low counter
      lngTmpHi = lngTmpHi - 1      ' Decrement the temp high counter
   End If
Wend

' ---------------------------------------------------------------------
' If the minimum number of elements in the array is LESS than the temp
' high end, then make a recursive call to this routine.  Always sort
' the low end of the array first.  This gives you a solid base.
' ---------------------------------------------------------------------
If (lngLow < lngTmpHi) Then
   QuickSort arNumeric(), lngLow, lngTmpHi
End If

' ---------------------------------------------------------------------
' If the temp low end is LESS than the maximum number of elements in
' the array, then make a recursive call to this routine.  The high end
' is always sorted last.
' ---------------------------------------------------------------------
If (lngTmpLow < lngHi) Then
   QuickSort arNumeric(), lngTmpLow, lngHi
```

```
   End If

End Sub

Private Function Swap_Data(lngValue1 As Long, lngValue2 As Long)

   ' ************************************************************************
   ' Routine:       Swap_Data
   '
   ' Description:   Swap data with each other.
   ' ************************************************************************

   ' ------------------------------------------------------------------------
   ' Define local variables
   ' ------------------------------------------------------------------------
   Dim lngTmp As Long

   ' ------------------------------------------------------------------------
   ' Swap the values with each other
   ' ------------------------------------------------------------------------
   lngTmp = lngValue1
   lngValue1 = lngValue2
   lngValue2 = lngTmp

End Function

Public Function Rnd2(sngLow As Single, sngHigh As Single) As Single

   ' ************************************************************************
   ' Routine:       Rnd2
   '
   ' Description:   Create a random value between two given values.
   ' ************************************************************************
   Static every22 As Integer
   If every22 = 22 Then
      Rnd (-1)
      every22 = 0
   Else
      every22 = every22 + 1
   End If
   DoEvents             ' gives the internal clock time to change milliseconds
   Randomize (CDbl(Now()) + Timer)      ' always use a unique value for a seed
   Rnd2 = (Rnd * (sngHigh - sngLow)) + sngLow

End Function
```

```
''-----------------------------------------------------------------------
''
'' Description : Matrix Operations Library
''
''-----------------------------------------------------------------------

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' The dimensions of the matrix are checked
' Here
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Function Find_R_C(Mat() As Double) As Double()
   Dim Rows As Integer, Columns As Integer
   Dim i As Integer, j As Integer
   Dim Result() As Double
   Columns = 0
   If Mat_1D(Mat, Rows) Then
      ReDim Result(Rows, 1)
      Result(0, 0) = Rows
      Result(0, 1) = Columns + 1

      For i = 1 To Rows
         Result(i, 1) = Mat(i - 1)
      Next i
   Else
      Call Mat_2D(Mat, Rows, Columns)
      ReDim Result(Rows, Columns)
      Result(0, 0) = Rows
      Result(0, 1) = Columns

      For i = 1 To Rows
         For j = 1 To Columns '- 1
            Result(i, j) = Mat(i - 1, j - 1)
         Next j
      Next i
   End If
   Find_R_C = Result
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Check if matrix has only one column
' shift the matrix one level and keep
' its dimensions details in Mat(0,0) and Mat(0,1)
' Mat(0,0)= no of rows
' Mat(0,1)= no of columns
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Function Mat_1D(Mat() As Double, m As Integer) As Boolean
   Dim Temp_MAT As Double
   On Error GoTo Error_Handler
   Temp_MAT = Mat(0, 0)
   Mat_1D = False
   Exit Function
Error_Handler:
   Mat_1D = True
   m = UBound(Mat) + 1
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Check if matrix has more than one column
' if so return the dimension as described above
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Private Sub Mat_2D(Mat() As Double, m As Integer, n As Integer)
   Dim Temp_MAT As Double, i As Integer
   i = 0
   m = UBound(Mat) + 1
   On Error GoTo Error_Handler
   Do Until i < -1
      Temp_MAT = Mat(0, i)
      i = i + 1
   Loop
Error_Handler:
   n = i
End Sub
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Add two matrices, their dimensions should be compatible!
' Function returns the summation or errors due to
' dimensions incompatibility
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function Add(Mat_1() As Double, Mat_2() As Double) As Double()
   Dim Mat1() As Double, Mat2() As Double
   Dim sol() As Double
   Dim i As Integer, j As Integer

   On Error GoTo Error_Handler

   Mat1 = Find_R_C(Mat_1)
   Mat2 = Find_R_C(Mat_2)

   If Mat1(0, 0) <> Mat2(0, 0) Or Mat1(0, 1) <> Mat2(0, 1) Then
      GoTo Error_Dimension
   End If

   ReDim sol(Mat1(0, 0) - 1, Mat1(0, 1) - 1)
   For i = 1 To Mat1(0, 0)
      For j = 1 To Mat1(0, 1)
         sol(i - 1, j - 1) = Mat1(i, j) + Mat2(i, j)
      Next j
   Next i

   Add = sol
   Erase sol

   Exit Function

Error_Dimension:
   Err.Raise "5005", , "Dimensions of the two matrices do not match !"

Error_Handler:
   If Err.Number = 5005 Then
      Err.Raise "5005", , "Dimensions of the two matrices do not match !"
   Else
      Err.Raise "5022", , "One or both of the matrices are null, this operation cannot be done !!"
   End If

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Subtracts two matrices from each other, their
' dimensions should be compatible!
' Function returns the solution or errors due to
' dimensions incompatibility
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function Subtract(Mat_1() As Double, Mat_2() As Double) As Double()
   Dim Mat1() As Double, Mat2() As Double
   Dim i As Integer, j As Integer, sol() As Double

   On Error GoTo Error_Handler

   Mat1 = Find_R_C(Mat_1)
   Mat2 = Find_R_C(Mat_2)

   If Mat1(0, 0) <> Mat2(0, 0) Or Mat1(0, 1) <> Mat2(0, 1) Then
      GoTo Error_Dimension
   End If

   ReDim sol(Mat1(0, 0) - 1, Mat1(0, 1) - 1)

   For i = 1 To Mat1(0, 0)
      For j = 1 To Mat1(0, 1)
         sol(i - 1, j - 1) = Mat1(i, j) - Mat2(i, j)
      Next j
   Next i

   Subtract = sol
   Erase sol
   Exit Function
```

```
Error_Dimension:
   Err.Raise "5007", , "Dimensions of the two matrices do not match !"

Error_Handler:
   If Err.Number = 5007 Then
      Err.Raise "5007", , "Dimensions of the two matrices do not match !"
   Else
      Err.Raise "5022", , "One or both of the matrices are null, this operation cannot be done !!"
   End If

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Multiply two matrices, their dimensions should be compatible!
' Function returns the solution or errors due to
' dimensions incompatibility
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function Multiply(Mat_1() As Double, Mat_2() As Double) As Double()
   Dim Mat1() As Double, Mat2() As Double, l As Integer
   Dim i As Integer, j As Integer, OptiString As String
   Dim sol() As Double, MulAdd As Double

   On Error GoTo Error_Handler

   MulAdd = 0

   Mat1 = Find_R_C(Mat_1)
   Mat2 = Find_R_C(Mat_2)

   If Mat1(0, 1) <> Mat2(0, 0) Then
      GoTo Error_Dimension
   End If

   ReDim sol(Mat1(0, 0) - 1, Mat2(0, 1) - 1)

   For i = 1 To Mat1(0, 0)
      For j = 1 To Mat2(0, 1)
         For l = 1 To Mat1(0, 1)
            MulAdd = MulAdd + Mat1(i, l) * Mat2(l, j)
         Next l
         sol(i - 1, j - 1) = MulAdd
         MulAdd = 0
      Next j
   Next i

   Multiply = sol
   Erase sol
   Exit Function

Error_Dimension:
   Err.Raise "5009", , "Dimensions of the two matrices not suitable for multiplication !"

Error_Handler:
   If Err.Number = 5009 Then
      Err.Raise "5009", , "Dimensions of the two matrices not suitable for multiplication !"
   Else
      Err.Raise "5022", , "One or both of the matrices are null, this operation cannot be done !!"
   End If

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Determinant of a matrix should be (nxn)
' Function returns the solution or errors due to
' dimensions incompatibility
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function Det(Mat() As Double) As Double
   Dim DArray() As Double, S As Integer
   Dim k As Integer, i As Integer, j As Integer
   Dim save As Double, ArrayK As Double, k1 As Integer
   Dim M1 As String, Mat1() As Double

   On Error GoTo Error_Handler
```

```
   Mat1 = Find_R_C(Mat)

   If Mat1(0, 0) <> Mat1(0, 1) Then GoTo Error_Dimension

   S = Mat1(0, 0)
   Det = 1
   DArray = Mat1()

   For k = 1 To S
      If DArray(k, k) = 0 Then
         j = k
         Do While ((j < S) And (DArray(k, j) = 0))
            j = j + 1
         Loop
         If DArray(k, j) = 0 Then
            Det = 0
            Exit Function
         Else
            For i = k To S
               save = DArray(i, j)
               DArray(i, j) = DArray(i, k)
               DArray(i, k) = save
            Next i
         End If

         Det = -Det
      End If
      ArrayK = DArray(k, k)
      Det = Det * ArrayK
      If k < S Then
         k1 = k + 1
         For i = k1 To S
            For j = k1 To S
               DArray(i, j) = DArray(i, j) - DArray(i, k) * (DArray(k, j) / ArrayK)
            Next j
         Next i
      End If
   Next

   Exit Function

Error_Dimension:
   Err.Raise "5011", , "Matrix should be a square matrix !"

Error_Handler:
   If Err.Number = 5011 Then
      Err.Raise "5011", , "Matrix should be a square matrix !"
   Else
      Err.Raise "5022", , "In order to do this operation values must be assigned to the matrix !!"
   End If
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Inverse of a matrix, should be (nxn) and det(Mat)<>0
' Function returns the solution or errors due to
' dimensions incompatibility
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function Inv(Mat() As Double) As Double()
   Dim sol() As Double
   Dim AI() As Double, AIN As Double, AF As Double, _
         Mat1() As Double
   Dim LL As Integer, LLM As Integer, L1 As Integer, _
         L2 As Integer, LC As Integer, LCA As Integer, _
         LCB As Integer, i As Integer, j As Integer

   On Error GoTo Error_Handler

   Mat1 = Find_R_C(Mat)
   If Mat1(0, 0) <> Mat1(0, 1) Then GoTo Error_Dimension

   If Det(Mat1) = 0 Then GoTo Error_Zero
   ReDim sol(Mat1(0, 0) - 1, Mat1(0, 0) - 1)

   LL = Mat1(0, 0)
   LLM = Mat1(0, 1)
   ReDim AI(LL, LL)
```

```
   For L2 = 1 To LL
      For L1 = 1 To LL
         AI(L1, L2) = 0
      Next
      AI(L2, L2) = 1
   Next

   For LC = 1 To LL
      If Abs(Mat1(LC, LC)) < 0.0000000001 Then
         For LCA = LC + 1 To LL
            If LCA = LC Then GoTo 1090
            If Abs(Mat1(LC, LCA)) > 0.0000000001 Then
               For LCB = 1 To LL
                  Mat1(LCB, LC) = Mat1(LCB, LC) + Mat1(LCB, LCA)
                  AI(LCB, LC) = AI(LCB, LC) + AI(LCB, LCA)
               Next
               GoTo 1100
            End If
            1090 Next
         End If

         1100
         AIN = 1 / Mat1(LC, LC)
         For LCA = 1 To LL
            Mat1(LCA, LC) = AIN * Mat1(LCA, LC)
            AI(LCA, LC) = AIN * AI(LCA, LC)
         Next

         For LCA = 1 To LL
            If LCA = LC Then GoTo 1150
            AF = Mat1(LC, LCA)
            For LCB = 1 To LL
               Mat1(LCB, LCA) = Mat1(LCB, LCA) - AF * Mat1(LCB, LC)
               AI(LCB, LCA) = AI(LCB, LCA) - AF * AI(LCB, LC)
            Next
            1150 Next

         Next

         For i = 1 To LL
            For j = 1 To LL
               sol(i - 1, j - 1) = AI(i, j)
            Next j
         Next i

         Inv = sol
         Erase sol

         Exit Function

Error_Zero:
         Err.Raise "5012", , "Determinent equals zero, inverse can't be found !"

Error_Dimension:
         Err.Raise "5014", , "Matrix should be a square matrix !"

Error_Handler:
         If Err.Number = 5012 Then
            Err.Raise "5012", , "Determinent equals zero, inverse can't be found !"
         ElseIf Err.Number = 5014 Then
            Err.Raise "5014", , "Matrix should be a square matrix !"
         End If

      End Function

      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      ' Multiply two vectors, dimensions should be (3x1)
      ' Function returns the solution or errors due to
      ' dimensions incompatibility
      ' Example:
      '  Check Main Form !!
      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Public Function MultiplyVectors(Mat_1() As Double, Mat_2() As Double) As Double()
         Dim Mat1() As Double, Mat2() As Double
         Dim i As Double, j As Double, k As Double
         Dim sol(2) As Double
```

```
         On Error GoTo Error_Handler

         Mat1 = Find_R_C(Mat_1)
         Mat2 = Find_R_C(Mat_2)

         If Mat1(0, 0) <> 3 Or Mat1(0, 1) <> 1 Then
            GoTo Error_Dimension
         End If

         If Mat2(0, 0) <> 3 Or Mat2(0, 1) <> 1 Then
            GoTo Error_Dimension
         End If

         i = Mat1(2, 1) * Mat2(3, 1) - Mat1(3, 1) * Mat2(2, 1)
         j = Mat1(3, 1) * Mat2(1, 1) - Mat1(1, 1) * Mat2(3, 1)
         k = Mat1(1, 1) * Mat2(2, 1) - Mat1(2, 1) * Mat2(1, 1)

         sol(0) = i: sol(1) = j: sol(2) = k

         MultiplyVectors = sol

         Exit Function

Error_Dimension:
         Err.Raise "5016", , "Dimension should be (3 x 1) for both matrices in order to do cross
»multiplication !"

Error_Handler:

         If Err.Number = 5016 Then
            Err.Raise "5016", , "Dimension should be (3 x 1) for both matrices in order to do cross
»multiplication !"
         Else
            Err.Raise "5022", , "One or both of the matrices are null, this operation cannot be done !!"
         End If

      End Function

      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      ' Magnitude of a Vector, vector should be (3x1)
      ' Function returns the solution or errors due to
      ' dimensions incompatibility
      ' Example:
      '  Check Main Form !!
      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Public Function VectorMagnitude(Mat() As Double) As Double

         Dim Mat1() As Double

         On Error GoTo Error_Handler

         Mat1 = Find_R_C(Mat)

         If Mat1(0, 0) <> 3 Or Mat1(0, 1) <> 1 Then
            GoTo Error_Dimension
         End If

         VectorMagnitude = Sqr(Mat1(1, 1) * Mat1(1, 1) + Mat1(2, 1) * Mat1(2, 1) + Mat1(3, 1) * Mat1(3, 1))

         Exit Function

Error_Dimension:
         Err.Raise "5018", , "Dimension of the matrix should be (1 x 3) in order to find the vector's norm
»!"

Error_Handler:
         If Err.Number = 5018 Then
            Err.Raise "5018", , "Dimension of the matrix should be (3 x 1) in order to find the vector's
»magnitude !"
         Else
            Err.Raise "5022", , "In order to do this operation values must be assigned to the matrix !!"
         End If

      End Function

      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      ' Transpose of a matrix
```

```vb
' Function returns the solution or errors
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function Transpose(Mat() As Double) As Double()
   Dim Mat1() As Double, Tr_Mat() As Double
   Dim i As Integer, j As Integer, sol() As Double

   On Error GoTo Error_Handler

   Mat1() = Find_R_C(Mat())

   ReDim Tr_Mat(0 To Mat1(0, 1), 0 To Mat1(0, 0))
   ReDim sol(Mat1(0, 1) - 1, Mat1(0, 0) - 1)
   Tr_Mat(0, 0) = Mat1(0, 1)
   Tr_Mat(0, 1) = Mat1(0, 0)

   For i = 1 To Mat1(0, 0)
      For j = 1 To Mat1(0, 1)
         Tr_Mat(j, i) = Mat1(i, j)
      Next j
   Next i

   For i = 1 To Tr_Mat(0, 0)
      For j = 1 To Tr_Mat(0, 1)
         sol(i - 1, j - 1) = Tr_Mat(i, j)
      Next j
   Next i

   Transpose = sol
   Erase sol
   Exit Function

Error_Handler:
   Err.Raise "5028", , "In order to do this operation values must be assigned to the matrix !!"

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Multiply a matrix or a vector with a scalar quantity
' Function returns the solution or errors
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function ScalarMultiply(Value As Double, Mat() As Double) As Double()
   Dim i As Integer, j As Integer
   Dim Mat1() As Double, sol() As Double

   On Error GoTo Error_Handler

   Mat1 = Find_R_C(Mat)
   ReDim sol(Mat1(0, 0) - 1, Mat1(0, 1) - 1)

   For i = 1 To Mat1(0, 0)
      For j = 1 To Mat1(0, 1)
         sol(i - 1, j - 1) = Mat1(i, j) * Value
      Next j
   Next i

   ScalarMultiply = sol

   Exit Function

Error_Handler:
   Err.Raise "5022", , "Matrix was not assigned"
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Divide matrix elements or a vector by a scalar quantity
' Function returns the solution or errors
' Example:
'  Check Main Form !!
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function ScalarDivide(Value As Double, Mat() As Double) As Double()
   Dim i As Integer, j As Integer
   Dim Mat1() As Double, sol() As Double

   On Error GoTo Error_Handler
```

```
         Mat1 = Find_R_C(Mat)
         ReDim sol(Mat1(0, 0) - 1, Mat1(0, 1) - 1)

         For i = 1 To Mat1(0, 0)
            For j = 1 To Mat1(0, 1)
               sol(i - 1, j - 1) = Mat1(i, j) / Value
            Next j
         Next i

         ScalarDivide = sol

         Exit Function

Error_Handler:
         Err.Raise "5022", , "Matrix was not assigned"
      End Function

      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      ' Print a matrix to multitext text box
      ' Function returns the solution or errors
      ' Example:
      '  Check Main Form !!
      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Public Function PrintMat(Mat_1() As Double) As String
         Dim N_Rows As Integer, N_Columns, k As Integer, _
               i As Integer, j As Integer, m As Integer
         Dim StrElem As String, StrLen As Long, _
               Greatest() As Integer, LarString As String
         Dim OptiString As String, sol As String

         Dim Mat1() As Double

         Mat1 = Find_R_C(Mat_1)

         sol = ""
         OptiString = ""

         N_Rows = Mat1(0, 0)
         N_Columns = Mat1(0, 1)

         ReDim Greatest(N_Columns)

         For i = 1 To N_Rows
            For j = 1 To N_Columns
               If i = 1 Then
                  Greatest(j) = 0
                  For m = 1 To N_Rows
                     StrElem = Format$(Mat1(m, j), "0.0000")
                     StrLen = Len(StrElem)
                     If Greatest(j) < StrLen Then
                        Greatest(j) = StrLen
                        LarString = StrElem
                     End If
                  Next m
                  If Mid$(LarString, 1, 1) = "-" Then Greatest(j) = Greatest(j) + 1
               End If
               StrElem = Format$(Mat1(i, j), "0.0000")
               If Mid$(StrElem, 1, 1) = "-" Then
                  StrLen = Len(StrElem)
                  If Greatest(j) >= StrLen Then
                     For k = 1 To (Greatest(j) - StrLen)
                        OptiString = OptiString & "  "
                     Next k
                     OptiString = OptiString & " "
                  End If
               Else
                  StrLen = Len(StrElem)
                  If Greatest(j) > StrLen Then
                     For k = 1 To (Greatest(j) - StrLen)
                        OptiString = OptiString & "  "
                     Next k
                  End If
               End If
               OptiString = OptiString & "  " & Format$(Mat1(i, j), "0.0000")
            Next j
            If i <> N_Rows Then
               sol = sol & OptiString & vbCrLf
```

```
            OptiString = ""
         End If
         sol = sol & OptiString
         OptiString = ""
      Next i

      PrintMat = sol

      Exit Function

End Function

Private Function Cutting(M_L As Integer)
      Dim Num As Integer
      Num = 0
      Num = M_L \ 20
      If M_L Mod 20 <> 0 Then Num = Num + 1
      Cutting = Num
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Return the maximum of two numbers
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function max(ByVal X As Double, Y As Double) As Double
      If X >= Y Then
         max = X
      ElseIf X < Y Then
         max = Y
      End If
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Return the minimum of two numbers
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function min(ByVal X As Double, Y As Double) As Double
      If X >= Y Then
         min = Y
      ElseIf X < Y Then
         min = X
      End If
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This routine finds the
' atan2(y,x) vlaue
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function atan2(ByVal Y As Double, ByVal X As Double) As Double
      Dim yy As Double, xx As Double
      yy = Abs(Y)
      xx = Abs(X)
      If Y = 0 And X > 0 Then
         atan2 = 0
      ElseIf Y > 0 And X > 0 Then
         atan2 = Atn(yy / xx)
      ElseIf Y > 0 And X = 0 Then
         atan2 = PI / 2
      ElseIf Y > 0 And X < 0 Then
         atan2 = PI - Atn(yy / xx)
      ElseIf Y = 0 And X < 0 Then
         atan2 = PI
      ElseIf Y < 0 And X < 0 Then
         atan2 = Atn(yy / xx) - PI
      ElseIf Y < 0 And X = 0 Then
         atan2 = PI / 2
      ElseIf Y < 0 And X > 0 Then
         atan2 = -Atn(yy / xx)
      End If
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This routine returns Pi value
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function PI() As Double
      PI = 4 * Atn(1)
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This routine returns the
```

```
' arc sin vlaue of an angle
''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function asin(ByVal X As Double) As Double
   asin = Atn(X / (Sqr(Abs(1 - X * X)) + 1E-200))
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''
' This routine returns the
' arc cos vlaue of an angle
''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function acos(ByVal X As Double) As Double
   acos = Atn(-X / (Sqr(1 - X * X))) + 2 * Atn(1)
End Function
```

Chapter 2 – Automated Retinal differencing Code

The following code written in Visual Basic.Net is the final beta version of ARID used to analyse the data in Volume I of this thesis. All commenting within the code is green.

The code (once compiled) will run on a Microsoft Windows platform.

Table of Contents for ARID

```
timerform.vb [ProjectItem] ..............................
  timerform [Class] .....................................
    ProgressBar_Click [Function] ........................
    timerform_Load [Function] ...........................
```

```
Public Class CEnhance
    Dim myImage As Bitmap
    Dim myChoice As Object
    Dim strSample_Size As String 'USED IN THRESHOLD
    Dim StartingP As Point 'USED IN THRESHOLD

    Function Equalise_Histogram() As Bitmap
        '----Init Vars-----

        Dim arrVars(100, 1), arrCumVars(100, 1) As Int32
        Dim arrHue(3600), arrCumHue(3600) As Int32
        Dim iX, iY As Int32
        Dim inorm, tmpSat, tmpLum As Double
        'Dim myPixel As HLSRGB
        'Dim best As Double

        '---Create Freq  arrays----
        For iX = 0 To myImage.Width - 1
            For iY = 0 To myImage.Height - 1

                tmpSat = myImage.GetPixel(iX, iY).GetSaturation
                tmpLum = myImage.GetPixel(iX, iY).GetBrightness


                tmpSat = Math.Round(tmpSat * 100)
                tmpLum = Math.Round(tmpLum * 100)


                arrVars(tmpSat, 0) = arrVars(tmpSat, 0) + 1
                arrVars(tmpLum, 1) = arrVars(tmpLum, 1) + 1

            Next iY
        Next iX

        '---Normalised constant---
        inorm = (100 + 1) / ((myImage.Width - 1) * (myImage.Height - 1))


        '========Create Normalised Cum Freq Array==========
        arrCumVars(0, 0) = arrVars(0, 0)
        arrCumVars(0, 1) = arrVars(0, 1)


        '----Sat & Lum----
        For iX = 1 To UBound(arrCumVars)
            arrCumVars(iX, 0) = arrCumVars(iX - 1, 0) + arrVars(iX, 0)
            arrCumVars(iX, 1) = arrCumVars(iX - 1, 1) + arrVars(iX, 1)
        Next iX

        For iX = 0 To UBound(arrCumVars)
            arrCumVars(iX, 0) = arrCumVars(iX, 0) * inorm
            arrCumVars(iX, 1) = arrCumVars(iX, 1) * inorm


        Next iX


        '------Use Look up table just created-----
        Dim Colour_converted_Pixel As HLSRGB
        Dim newHue, newLum, newSat As Double
        For iX = 0 To myImage.Width - 1
            For iY = 0 To myImage.Height - 1
                '-set attributes-
                newHue = myImage.GetPixel(iX, iY).GetHue
                newLum = myImage.GetPixel(iX, iY).GetBrightness
                newSat = myImage.GetPixel(iX, iY).GetSaturation

                '----Look up value----
                If myChoice = 1 Then
                    newLum = (arrCumVars(newLum * 100, 1) / 100)
                ElseIf myChoice = 2 Then
                    newSat = (arrCumVars(newSat * 100, 0) / 100)
                ElseIf myChoice = 3 Then
                    newLum = (arrCumVars(newLum * 100, 1) / 100)
```



```
                    newSat = (arrCumVars(newSat * 100, 0) / 100)
```

```
            Else
                'do nothing with original pixels

            End If



            '---checks---
            If newSat > 1 Then newSat = 1
            If newLum > 1 Then newLum = 1


            '---insert new pixel-----
            Colour_converted_Pixel = New HLSRGB(newHue, newLum, newSat)
            myImage.SetPixel(iX, iY, Colour_converted_Pixel.Color)
        Next iY
    Next iX

    Equalise_Histogram = myImage
End Function

Function Thresholding() As Bitmap
    '---Init Vars---
    Dim N_Pixels, iData As Int32
    Dim V_Max, V_Min As Double
    Dim Threshold, Threshold25, Threshold75 As Double
    Dim ix, iy As Int32
    Dim dblBright, Sum_Top, Sum_Bottom As Double
    Dim iVal As Int32
    '--Set Vars--
    N_Pixels = CInt((myImage.Width * myImage.Height) * 0.1)


    If strSample_Size = "" Then
        iVal = 999
    Else
        iVal = CInt(strSample_Size)
    End If

    Dim arr_Max(iVal) As Double 'N_Pixels - 1) As Double
    Dim arr_Min(iVal) As Double 'N_Pixels - 1) As Double



    '---Get top/bottom 10% of Pixels---

    For iy = 0 To myImage.Height - 1
        ' For iy = StartingP.Y To StartingP.Y + 100
        For ix = 0 To myImage.Width - 1
            '    For ix = StartingP.X To StartingP.X + 100 '
            '-Get Brightness-
            If (ix > myImage.Width - 1) Or (iy > myImage.Height - 1) Then Exit For
            dblBright = myImage.GetPixel(ix, iy).GetBrightness

            '-Escape-
            If dblBright = 0 Or dblBright = 1 Then GoTo Skip_all

            '==Top 10==
            For iData = 0 To UBound(arr_Max)
                '-Escape-
                '-Check-
                If dblBright >= arr_Max(iData) Then
                    arr_Max(iData) = dblBright

                End If
            Next iData

            '==Bottom 10==
            For iData = 0 To UBound(arr_Min)
                '-Escape-

                '-Check-
                If dblBright <= arr_Min(iData) Or arr_Min(iData) = 0 Then
```

```
                    arr_Min(iData) = dblBright

                End If
            Next iData
Skip_All:

        Next ix
    Next iy

    '===Average Top/Bottom Pixels===
    For iData = 0 To UBound(arr_Min)
        Sum_Top = Sum_Top + arr_Max(iData)
        Sum_Bottom = Sum_Bottom + arr_Min(iData)
    Next iData

    V_Max = Sum_Top / UBound(arr_Max)
    V_Min = Sum_Bottom / UBound(arr_Min)

    '==Set Threshold Level===
    '===(Vmax+vmin)/2====
    Threshold = (V_Max + V_Min) / 2
    Threshold25 = Threshold * 0.5
    Threshold75 = Threshold * 1.5


    'Threshold = V_Max * 0.9

    '==Redraw Pic===
    Dim Colour_converted_Pixel As HLSRGB
    Dim newHue, newLum, newSat As Double
    For iy = 0 To myImage.Height - 1
        '     For iy = StartingP.Y To StartingP.Y + 100
        For ix = 0 To myImage.Width - 1
            '    For ix = StartingP.X To StartingP.X + 100 '

            '-Escape-
            If (ix > myImage.Width - 1) Or (iy > myImage.Height - 1) Then Exit For

            '-set attributes-
            newHue = myImage.GetPixel(ix, iy).GetHue
            newLum = myImage.GetPixel(ix, iy).GetBrightness
            newSat = myImage.GetPixel(ix, iy).GetSaturation

            '-Set Threshold Level-
            If newLum <= Threshold25 Then
                newLum = 0
            ElseIf newLum <= Threshold Then
                'If newLum <= Threshold Then
                '  newLum = 0
                newLum = 0.33
            ElseIf newLum <= Threshold75 Then
                newLum = 0.66
            Else
                newLum = 1
            End If

            '---insert new pixel-----
            Colour_converted_Pixel = New HLSRGB(newHue, newLum, newSat)
            myImage.SetPixel(ix, iy, Colour_converted_Pixel.Color)
        Next ix
    Next iy

    Thresholding = myImage
End Function

Public Sub New(ByVal myImg As Bitmap, ByVal sender As Integer, ByVal strThreshold As
»       String, ByVal myP As Point)
    Dim tmpI As New Bitmap(myImg)
    myImage = tmpI
    myChoice = sender
    strSample_Size = strThreshold

    '--Reset Starting P to Center of Filter--
    StartingP.X = myP.X - (100 / 2)
    StartingP.Y = myP.Y - (100 / 2)
```

```
    End Sub
End Class
```

ARID CFS.vb

```
Imports System.IO.FileSystemInfo
Imports System.IO.File
Imports System.IO.Directory
Imports System.IO

Public Class CFSO


    '-file system object-
    Dim Curr_Dir As String

    Sub Set_Dir(ByVal strdir As String)
        Curr_Dir = strdir
    End Sub

    Function Get_Image_Pair(ByVal iPair) As Image()
        Dim dir_info As New DirectoryInfo(Curr_Dir)
        Dim Pics(1) As Image

        Dim subFiles() As FileInfo = dir_info.GetFiles

        Pics(0) = Bitmap.FromFile(subFiles(iPair).FullName)
        Pics(1) = Bitmap.FromFile(subFiles(iPair + 1).FullName)
        Get_Image_Pair = Pics
    End Function



    Function Filecount() As Int32


        Dim dir_info As New DirectoryInfo(Curr_Dir)

        Dim count As Int32
        Dim subFiles() As FileInfo = dir_info.GetFiles

        count = (UBound(subFiles))


        Filecount = count

    End Function

    Function Getname(ByVal iwhich As Int32) As String


        Dim dir_info As New DirectoryInfo(Curr_Dir)

        Dim sname As String
        Dim subFiles() As FileInfo = dir_info.GetFiles

        sname = Left(subFiles(iwhich).Name, Len(subFiles(iwhich).Name) - 5)


        Getname = sname

    End Function

End Class
```

ARID CNorm_Corralate.vb

```
Imports System.Math
Public Structure Vector
    Public myP As Point
    Public myT As Double
End Structure

Public Class CNorm_Corralate

    Dim myMASK As Image
    Dim myImage As Image
    Dim ix, iY As Int32
    Const MAX_ARRAY_SIZE = 10000000
    Dim MAX_Rows As Int64
    Dim Partition_StartP As Point
    Dim tmpI As Int32
    Dim arrOFR(0) As Point
    Dim arrBest(2, 9) As Double ' used in norm final

    Function GET_arrofR() As Point()
        ReDim Preserve arrOFR(UBound(arrOFR) - 1)
        Return arrOFR
    End Function

    '---extract Mask into Array---
    Private Function Get_Mask() As Double(,)
        '---init vars---
        Dim arrMask(myMASK.Width - 1, myMASK.Height - 1) As Double
        Dim myB As New Bitmap(myMASK)

        '---get data---
        For ix = 0 To myMASK.Width - 1
            For iY = 0 To myMASK.Height - 1
                arrMask(ix, iY) = myB.GetPixel(ix, iY).GetBrightness
            Next iY
        Next ix

        '-clean & return -
        Get_Mask = arrMask
        Erase arrMask
        myB = Nothing
    End Function
    '---extract part/all of image into array---
    Private Function Scan_Image(ByVal myI As Image) As Double(,)
        '---init vars---
        'Dim Need_To_Split As Boolean
        Dim arrImage(,) As Double
        Dim myB As New Bitmap(myI)


        '---define array size---
        If (myI.Height * myI.Width) > MAX_ARRAY_SIZE Then
            'Need_To_Split = True

            ReDim arrImage(myI.Width - 1, MAX_Rows)




        Else
            'Need_To_Split = False
            ReDim arrImage(myI.Width - 1, myI.Height - 1)
        End If

        '---scan image---
        For ix = 0 To UBound(arrImage)
            For iY = 0 To UBound(arrImage, 2)
                '---Escape clause---
                If (iY + Partition_StartP.Y) >= myI.Width Then
                    Exit For
                End If
                arrImage(ix, iY) = myB.GetPixel(ix + Partition_StartP.X, iY +
»                   Partition_StartP.Y).GetBrightness
            Next iY
```

```
        Next ix

        '-clean & return -
        Scan_Image = arrImage
        Erase arrImage
        myB = Nothing
    End Function
    '---Cross corralate given region---
    Private Function Corralate(ByVal arrImage(,) As Double, ByVal arrMask(,) As Double,
»       ByVal startP As Point) As Double(,)
        '================= Init Vars ================
        Dim X As Double, Y As Double
        Dim SumTop As Double, SumBot As Double, SumBotX As Double, SumBotY As Double,
»           RootBot As Double
        Dim MeanY As Double, MeanX As Double ', Data_Items_Total As Long
        Dim Max As Long

        Dim Main_X, Main_Y As Int32
        Dim arrMask_N(,), arrImage_N(,) As Double
        '=============== Init More Vars =============
        Dim arrOF_R_Vals(UBound(arrMask), UBound(arrMask, 2)) As Double
        Max = (UBound(arrOF_R_Vals) + 1) * (UBound(arrOF_R_Vals, 2) + 1)

        '====Normalise Areas to Correlate===
        Dim tmpP As Point
        arrMask_N = Normalise(arrMask, tmpP, tmpP)
        tmpP.X = startP.X + UBound(arrMask)
        tmpP.Y = startP.Y + UBound(arrMask, 2)
        arrImage_N = Normalise(arrImage, startP, tmpP)
        arrMask_N = arrMask
        arrImage_N = arrImage
        '=========== Perform AutoCorrelation =========
        For Main_X = 0 To UBound(arrOF_R_Vals)
            For Main_Y = 0 To UBound(arrOF_R_Vals, 2)

                '------ Reset Vars ------
                SumTop = 0
                SumBotX = 0
                SumBotY = 0
                SumBot = 0
                MeanX = 0
                MeanY = 0
                '~~~~~~~~~~~~~~~~~~~~~~

                '------ Calculate Means -----
                For ix = 0 To UBound(arrMask_N)
                    For iY = 0 To UBound(arrMask_N, 2)
                        MeanX = MeanX + arrImage_N(ix + Partition_StartP.X + startP.X, iY
»                           + Partition_StartP.Y + startP.Y)
                        MeanY = MeanY + arrMask_N(ix, iY)
                    Next iY
                Next ix

                MeanX = MeanX / Max
                MeanY = MeanY / Max
                '~~~~~~~~~~~~~~~~~~~~~~~~~~~

                '---------------- Calculate Co-efficient -----------------
                ' R = sum x-xbar*y-bar/ root(sum(x-xbar^2)* sum(Y-ybar^2))
                For ix = 0 To UBound(arrMask_N)
                    For iY = 0 To UBound(arrMask_N, 2)
                        X = arrImage_N(ix + Partition_StartP.X + startP.X, iY +
»                           Partition_StartP.Y + startP.Y)
                        Y = arrMask_N(ix, iY)
                        SumTop = SumTop + ((X - MeanX) * (Y - MeanY))
                        SumBotX = SumBotX + ((X - MeanX) * (X - MeanX))
                        SumBotY = SumBotY + ((Y - MeanY) * (Y - MeanY))
                    Next iY
                Next ix

                SumBot = 0
                SumBot = SumBotX * SumBotY
                RootBot = 0
                RootBot = Sqrt(SumBot)
```

```
                '---------- Store Results ----------------
                If RootBot <> 0 Then
                    arrOF_R_Vals(Main_X, Main_Y) = (SumTop / RootBot)
                Else
                    arrOF_R_Vals(Main_X, Main_Y) = 0
                End If
            Next Main_Y
        Next Main_X



        '========= Return Array ============
        Corralate = arrOF_R_Vals

        '======Clean Up=======
        Erase arrImage
        Erase arrMask
        Erase arrImage_N
        Erase arrMask_N
        Erase arrOF_R_Vals
        tmpP = Nothing
    End Function

    Private Function Normalised_CrossCorrelation(ByVal arrMask(,) As Double, ByVal arrImg2
»       (,) As Double, ByVal i As Int32, ByVal j As Int32) As Double
        'S(i,j) = Sum (arrimg2(x+i,y+i) * arrmask(x,y))
        '          ------------------------------------
        '         Sqrt(Sum (arrimg2(x+i,y+i)* arrimg2(x+i,y+i) * SUM(arrmask(x,y)*arrmask
»           (x,y)))

        '-init vars-
        Dim Sum_Top As Double
        Dim Sum_Bot1, Sum_Bot2 As Double

        '-Calculate Sum parts of Equation-
        For ix = 0 To UBound(arrMask)
            For iY = 0 To UBound(arrMask, 2)

                If ((ix + i) >= 0) And ((iY + j) >= 0) Then 'EXCLUDE AREAS OUTSIDE OF IMAG
»                   E
                    If ((ix + i) <= UBound(arrMask)) And ((iY + j) <= UBound(arrMask, 2))
»                       Then 'EXCLUDE AREAS OUTSIDE OF IMAGE
                        ' If (arrImg2(ix + i, iY + j) > 0.1) And (arrMask(ix, iY) > 0.1) T
»                           hen 'EXCLUDE BLACK AREAS
                        '-Calc top part-
                        Sum_Top = Sum_Top + (arrImg2(ix + i, iY + j) * arrMask(ix, iY))
                        '-Calc Bot1 & Bot2-
                        Sum_Bot1 = Sum_Bot1 + (arrImg2(ix + i, iY + j) * arrImg2(ix + i,
»                           iY + j))
                        Sum_Bot2 = Sum_Bot2 + (arrMask(ix, iY) * arrMask(ix, iY))
                        ' End If
                    End If
                End If
            Next iY
        Next ix

        '-Finish off equation-
        Normalised_CrossCorrelation = Sum_Top / Sqrt(Sum_Bot1 * Sum_Bot2)

    End Function

    '---Cross correlate Mask against image---
    Function Find_Match() As Image
        '---init vars---
        Dim mMask(,), mImage(,), mCoef(,), mBest(,) As Double
        Dim i, j, X, Y As Int32
        Dim startP As Point

        Dim SumR, MeanR, BestR As Double, BestP As Point
        Dim SumM, meanM, SumA, MeanA As Double
        Dim TimerF As New timerform()
        '---Get Mask---
        mMask = Get_Mask()
```

```
        '---mean Intensity of mask---
        For ix = 0 To UBound(mMask)
            For iY = 0 To UBound(mMask, 2)
                SumM = SumM + mMask(ix, iY)
            Next iY
        Next ix

        meanM = SumM / ((UBound(mMask) + 1) * (UBound(mMask, 2) + 1))

        '--Get Part/All of Image--
        Partition_StartP.X = 0
        Partition_StartP.Y = 0

        ' Do While (Partition_StartP.Y + MAX_Rows) > myImage.Height
        '----set up timer form---
        TimerF.Visible = True

        TimerF.ProgressBar.Maximum = ((myImage.Width - myMASK.Width) * (myImage.Height -
»           myMASK.Height)) + 1

        TimerF.ProgressBar.Increment(1)
        '---Scan relevant part of image--
        mImage = Scan_Image(myImage)
        For X = 0 To myImage.Width - 1 - myMASK.Width 'Step 5
            For Y = 0 To myImage.Height - 1 - myMASK.Height 'Step 5
                '---inc progress bar---
                TimerF.ProgressBar.Value = TimerF.ProgressBar.Value + 1

                '--set vars--
                startP.X = X
                startP.Y = Y
                SumR = 0
                SumA = 0

                '---Decide if this areas worth looking at---
                For i = 0 To UBound(mMask)
                    For j = 0 To UBound(mMask, 2)
                        SumA = SumA + mImage(startP.X + i, startP.Y + j)
                    Next j
                Next i

                MeanA = SumA / ((UBound(mMask) + 1) * (UBound(mMask, 2) + 1))

                '--Is mean intensity with 1% of masks--
                If (MeanA >= (meanM * 0.99)) And (MeanA <= (meanM * 1.01)) Then

                    '--Correlate--
                    mCoef = Corralate(mImage, mMask, startP)

                    '---Calc MeanR---
                    For i = 0 To UBound(mCoef)
                        For j = 0 To UBound(mCoef, 2)
                            SumR = SumR + mCoef(i, j)
                        Next j
                    Next i

                    MeanR = SumR / ((UBound(mCoef) + 1) * (UBound(mCoef, 2) + 1))
                    '---Save Best---
                    If Abs(MeanR) > BestR Then
                        BestR = Abs(MeanR)
                        mBest = mCoef
                        BestP.X = X
                        BestP.Y = Y
                    End If

                    tmpI = tmpI + 1
                    TimerF.Text = TimerF.ProgressBar.Value.ToString & "MATCH"
                    TimerF.Refresh()
                Else
                    TimerF.Text = TimerF.ProgressBar.Value.ToString & "NO MATCH"
                    TimerF.Refresh()
                End If


            Next Y
        Next X
```

```
        TimerF = Nothing

        MsgBox(tmpI & " Candidate regions, Best R= " & BestR)
        '---Set next start Point---
        Partition_StartP.X = 0
        Partition_StartP.Y = Partition_StartP.Y + MAX_Rows + 1
        'Loop

        '---draw---
        Dim myPiccyG As Graphics
        myPiccyG = Graphics.FromImage(myImage)
        myMASK = InVert_IMG(myMASK)
        myPiccyG.DrawImage(myMASK, BestP)



        Return myImage
    End Function

    Function Correlate_match(ByVal Factor As Int32) As Vector
        '--Init Vars--
        Dim mImage2(,), mImage(,) As Double
        Dim myImg As New Bitmap(myImage)
        Dim myImg2 As New Bitmap(myMASK)
        Dim R, bR As Double, X, Y As Int32
        Dim Offset_X, Offset_Y As Int32
        Dim MOA As Int16 = 5 'Maximum_Offset_Ratio
        Dim MOS As Int16 = 1  'Maximum Offset Step
        Dim pBest As Point
        Dim Total_Pixels As Int64
        Dim X_percentage_IMG_used, Y_percentage_IMG_used, bTheta As Double

        '---Scan image 1 --
        mImage = Scan_Image(myImg)
        Total_Pixels = UBound(mImage, 2) * Factor * UBound(mImage) * Factor
        '---Scan Image 2---
        mImage2 = Scan_Image(myImg2)
        Debug.WriteLine("Starting Correlate at " & Now())
        '--correlate images--
        For Offset_Y = -UBound(mImage, 2) / MOA To UBound(mImage, 2) + (UBound(mImage, 2)
»           / MOA) Step MOS
            For Offset_X = -UBound(mImage) / MOA To UBound(mImage) + (UBound(mImage) / MOA
»               ) Step MOS


                If Offset_X > 10 And Offset_Y > 0 Then

                    Debug.Write("")
                End If
                '-Calc missing pixels-

                X = (UBound(mImage) * Factor)
                X = X - (Abs(Offset_X) * Factor)
                Y = (UBound(mImage, 2) * Factor)
                Y = Y - (Abs(Offset_Y) * Factor)



                '--calc %of image used in CC---
                X_percentage_IMG_used = X / (UBound(mImage) * Factor)
                Y_percentage_IMG_used = Y / (UBound(mImage, 2) * Factor)

                If (X_percentage_IMG_used > 0.8) And (Y_percentage_IMG_used > 0.8) Then
                    R = Normalised_CrossCorrelation(mImage, mImage2, Offset_X, Offset_Y)

                    If R > bR Then
                        bR = R
                        pBest.X = Offset_X
                        pBest.Y = Offset_Y



                    End If
                End If
            Next Offset_X
```

```
        Next Offset_Y
        ' Debug.WriteLine("Ending Correlate at " & Now())
        Dim myVector As New Vector()
        myVector.myP.X = pBest.X
        myVector.myP.Y = pBest.Y


        '====BEST ROATATION====
        '-Translate to best found match-
        Debug.WriteLine("Starting Rotate at " & Now())
        Dim wid = myImg.Width
        Dim Hgt = myImg.Height

        Dim gr_out As Graphics
        Dim myCorners As Point() = { _
           New Point(0, 0), _
           New Point(wid, 0), _
           New Point(0, Hgt)}


        myCorners = Translate_XY_Points(myCorners, pBest)
        Dim myIMG_T As New Bitmap(CInt(myImg.Width), CInt(myImg.Height))
        gr_out = Graphics.FromImage(myIMG_T)
        gr_out.DrawImage(myImg, myCorners)
        '  Dim myF As New FRMFlip()
        'myF.Pic2.Image = myIMG_T
        'myF.PIC1.Image = myImg2
        'myF.Visible = True
        'myF.Pic2.Visible = True

        '---Rotate around Best Find---
        Dim MAX_Theta As Int16 = 5
        Dim theta, myTheta As Double
        Dim Origin As Point
        Dim mImage3(,) As Double
        Dim bm_out As Bitmap
        Dim Rotated_Corners As Point()



        Origin.X = (wid / 2) + pBest.X
        Origin.Y = (Hgt / 2) + pBest.Y
        '===Rotate and Diff Points===
        For theta = -MAX_Theta To MAX_Theta Step 0.1
            '-reset corners-
            Rotated_Corners = New Point() { _
               New Point(0 + pBest.X, 0 + pBest.Y), _
               New Point(wid + pBest.X, 0 + pBest.Y), _
               New Point(0 + pBest.X, Hgt + pBest.Y)}

            '-CRAP VB.NET FIX-
            If theta < 0 And theta > -0.1 Then
                theta = 0
            End If
            '-------Rotate Points-------
            If theta < 0 Then
                myTheta = 360 + theta
            Else
                myTheta = theta
            End If

            If theta = 0 Then
                Debug.Write("")
            End If

            Rotated_Corners = Rotate_XY_Points(Rotated_Corners, myTheta, Origin)
            bm_out = New Bitmap(CInt(myImg.Width), CInt(myImg.Height))
            gr_out = Graphics.FromImage(bm_out)
            gr_out.DrawImage(myImg, Rotated_Corners)



            '  myF.Pic2.Image = bm_out
            ' Exit For
            '---Scan image 3--
```

```
            mImage3 = Scan_Image(bm_out)
            R = Normalised_CrossCorrelation(mImage2, mImage3, 0, 0)
            If R > bR Then
                bR = R
                bTheta = myTheta

            End If


        Next
        Debug.WriteLine("Ending Rotate at " & Now())
        myVector.myT = bTheta

        Debug.Write("BEST X = " & pBest.X & ",Best Y = " & pBest.Y & " Theta = " &
»           myVector.myT & " R= " & bR)
        Return myVector
    End Function
    Function Final_Correlation(ByVal Factor As Int32) As Vector 'Worse than minimatch
        '---Init Vars---
        Dim mImage2(,), mImage(,) As Double
        Dim myImg As New Bitmap(myImage)
        Dim myImg2 As New Bitmap(myMASK)
        Dim iR, R, bR As Double, X, Y As Int32
        Dim Offset_X, Offset_Y As Int32
        Dim MOA As Int16 = 4 'Maximum_Offset_Ratio
        Dim MOS As Int16 = 3  'Maximum Offset Step
        Dim pBest As Point
        Dim pixels_not_used, Total_Pixels, bTheta As Int64
        Dim X_percentage_IMG_used, Y_percentage_IMG_used As Double
        Dim i, j As Int16
        Dim temp1, temp2, temp3 As Double
        Dim myVector As New Vector()

        '---Scan image 1 --
        mImage = Scan_Image(myImg)
        Total_Pixels = UBound(mImage, 2) * Factor * UBound(mImage) * Factor
        '---Scan Image 2---
        mImage2 = Scan_Image(myImg2)
        Debug.WriteLine("Starting Correlate at " & Now())

        '===========scan X=================
        For Offset_X = -UBound(mImage) / MOA To UBound(mImage) + (UBound(mImage) / MOA) 'S
»           tep MOS

            X = (UBound(mImage) * Factor)
            X = X - (Abs(Offset_X) * Factor)



            '--calc %of image used in CC---
            X_percentage_IMG_used = X / (UBound(mImage) * Factor)

            If (X_percentage_IMG_used > 0.8) Then
                R = Normalised_CrossCorrelation(mImage, mImage2, Offset_X, Offset_Y)


                '-Store Best 10  matches-
                For iR = 0 To UBound(arrBest, 2)
                    If R > arrBest(2, iR) Then
                        arrBest(0, iR) = Offset_X
                        arrBest(1, iR) = Offset_Y
                        arrBest(2, iR) = R

                        For i = 0 To UBound(arrBest, 2)
                            For j = 1 To UBound(arrBest, 2)
                                If arrBest(2, j) < arrBest(2, i) Then
                                    '-Store Val-
                                    temp1 = arrBest(0, i)
                                    temp2 = arrBest(1, i)
                                    temp3 = arrBest(2, i)
                                    '-1st swap-
                                    arrBest(0, i) = arrBest(0, j)
                                    arrBest(1, i) = arrBest(1, j)
                                    arrBest(2, i) = arrBest(2, j)
```

```
                                '-2nd Swap-
                                arrBest(0, j) = temp1
                                arrBest(1, j) = temp2
                                arrBest(2, j) = temp3

                            End If
                        Next j
                    Next i

                    Exit For

                End If
            Next



            If R > bR Then
                bR = R
                pBest.X = Offset_X
                ' pBest.Y = Offset_Y
            End If


        End If


    Next
    '--Search just these on Y axis--
    For i = 0 To UBound(arrBest, 2)
        For Offset_Y = -UBound(mImage, 2) / MOA To UBound(mImage, 2) + (UBound(mImage,
»            2) / MOA) Step MOS
            Y = (UBound(mImage, 2) * Factor)
            Y = Y - (Abs(Offset_Y) * Factor)
            Y_percentage_IMG_used = Y / (UBound(mImage, 2) * Factor)
            If (Y_percentage_IMG_used > 0.8) Then
                R = Normalised_CrossCorrelation(mImage, mImage2, arrBest(0, i),
»                   Offset_Y)
                If R > bR Then
                    bR = R
                    ' pBest.X = Offset_X
                    pBest.Y = Offset_Y
                End If
            End If
        Next
    Next

    Debug.WriteLine("Ending Correlate at " & Now())

    myVector.myP.X = pBest.X
    myVector.myP.Y = pBest.Y

    '====BEST ROATATION====
    '-Translate to best found match-
    Debug.WriteLine("Starting Rotate at " & Now())
    Dim wid = myImg.Width
    Dim Hgt = myImg.Height

    Dim gr_out As Graphics
    Dim myCorners As Point() = { _
       New Point(0, 0), _
       New Point(wid, 0), _
       New Point(0, Hgt)}

    Dim centerP As Point
    centerP.X = wid / 2
    centerP.Y = Hgt / 2
    myCorners = Translate_XY_Points(myCorners, pBest)
    Dim myIMG_T As New Bitmap(CInt(myImg.Width), CInt(myImg.Height))

    gr_out = Graphics.FromImage(myIMG_T)
    gr_out.DrawImage(myImg, myCorners)


    '---Rotate around Best Find---
    Dim MAX_Theta As Int16 = 5
    Dim theta, myTheta As Double
```

```
        Dim Origin As Point
        Dim mImage3(,) As Double
        Dim bm_out As Bitmap
        '--origin--
        Origin.X = myIMG_T.Width / 2
        Origin.Y = myIMG_T.Height / 2

        '===Rotate and Diff Points===
        For theta = -MAX_Theta To MAX_Theta Step 0.1
            '-CRAP VB.NET FIX-
            If theta < 0 And theta > -0.1 Then
                theta = 0
            End If
            '-------Rotate Points-------
            If theta < 0 Then
                myTheta = 360 + theta
            Else
                myTheta = theta
            End If

            If CInt(myTheta) = 4.0 Then
                Debug.Write("")
            End If


            myCorners = Rotate_XY_Points(myCorners, myTheta, Origin)
            bm_out = New Bitmap(CInt(myImg.Width), CInt(myImg.Height))
            gr_out = Graphics.FromImage(bm_out)
            gr_out.DrawImage(myIMG_T, myCorners)

            '---Scan image 3--
            mImage3 = Scan_Image(bm_out)
            R = Normalised_CrossCorrelation(mImage2, mImage3, 0, 0)
            If R > bR Then
                bTheta = myTheta

            End If


        Next
        Debug.WriteLine("Ending Rotate at " & Now())
        myVector.myT = bTheta

        '  MsgBox("BEST X = " & pBest.X & ",Best Y = " & pBest.Y & " Theta = " & bTheta &
»           " R= " & bR)
        Return myVector
    End Function
    Function Correlate_Mini_match(ByVal Factor As Int32) As Vector 'THIS IS THE ONE WE USE
        '--Init Vars--
        Dim mImage2(,), mImage(,) As Double
        Dim myImg As New Bitmap(myImage)
        Dim myImg2 As New Bitmap(myMASK)
        Dim R, bR As Double, X, Y As Int32
        Dim Offset_X, Offset_Y As Int32
        Dim MOA As Int16 = 4 'Maximum_Offset_Ratio
        Dim MOS As Int16 = 10  'Maximum Offset Step
        Dim pBest As Point
        Dim bMiss, imiss, pixels_not_used, Total_Pixels, bTheta As Int64
        Dim X_percentage_IMG_used, Y_percentage_IMG_used As Double

        '---Scan image 1 --
        mImage = Scan_Image(myImg)
        Total_Pixels = UBound(mImage, 2) * Factor * UBound(mImage) * Factor
        '---Scan Image 2---
        mImage2 = Scan_Image(myImg2)
        Debug.WriteLine("Starting Correlate at " & Now())
        '--correlate images--
        For Offset_Y = -UBound(mImage, 2) / MOA To UBound(mImage, 2) + (UBound(mImage, 2)
»           / MOA) Step MOS
            For Offset_X = -UBound(mImage) / MOA To UBound(mImage) + (UBound(mImage) / MOA
»               ) Step MOS


                If Offset_X > 10 And Offset_Y > 0 Then

                    Debug.Write("")
```

```
            End If
            '-Calc missing pixels-
            ' If (Offset_Y < 0 Or Offset_X < 0) Or (Offset_Y > UBound(mImage, 2) Or Of
»               fset_X > UBound(mImage)) Then
            '  pixels_not_used = (Abs(Offset_Y) * Factor) * (Abs(Offset_X) * Factor)
            'pixels_not_used = (Abs(Offset_Y)) * (Abs(Offset_X))
            X = (UBound(mImage) * Factor)
            X = X - (Abs(Offset_X) * Factor)
            Y = (UBound(mImage, 2) * Factor)
            Y = Y - (Abs(Offset_Y) * Factor)

            ' End If


            '--calc %of image used in CC---
            X_percentage_IMG_used = X / (UBound(mImage) * Factor)
            Y_percentage_IMG_used = Y / (UBound(mImage, 2) * Factor)
            If (X_percentage_IMG_used > 0.8) And (Y_percentage_IMG_used > 0.8) Then
                R = Normalised_CrossCorrelation(mImage, mImage2, Offset_X, Offset_Y)

                '-Store all GOOD matches-
                If R > 0.9 Then
                    Debug.WriteLine("R= " & R & " (" & Offset_X & ", " & Offset_Y & ")
»                       ")
                    arrOFR(UBound(arrOFR)).X = Offset_X
                    arrOFR(UBound(arrOFR)).Y = Offset_Y
                    ReDim Preserve arrOFR(UBound(arrOFR) + 1)
                End If

                If R > bR Then
                    bR = R
                    pBest.X = Offset_X
                    pBest.Y = Offset_Y
                    bMiss = pixels_not_used


                End If
            End If
        Next Offset_X


    Next Offset_Y
    'MsgBox("BEST X = " & pBest.X & ",Best Y = " & pBest.Y & " Missing = " & bMiss & "
»        Theta = " & bTheta & "R= " & bR)
    Debug.WriteLine("Ending Correlate at " & Now())
    Dim myVector As New Vector()
    myVector.myP.X = pBest.X
    myVector.myP.Y = pBest.Y


    '====BEST ROATATION====
    '-Translate to best found match-
    Debug.WriteLine("Starting Rotate at " & Now())
    Dim wid = myImg.Width
    Dim Hgt = myImg.Height

    Dim gr_out As Graphics
    Dim myCorners As Point() = { _
       New Point(0, 0), _
       New Point(wid, 0), _
       New Point(0, Hgt)}

    Dim centerP As Point
    centerP.X = wid / 2
    centerP.Y = Hgt / 2
    myCorners = Translate_XY_Points(myCorners, centerP)
    Dim myIMG_T As New Bitmap(CInt(myImg.Width), CInt(myImg.Height))
    gr_out = Graphics.FromImage(myIMG_T)

    gr_out.DrawImage(myImg, myCorners)

    ' Dim myF As New FRMFlip()
    ' myF.Pic2.Image = myIMG_T
    ' myF.PIC1.Image = myImg2
    ' myF.Visible = True
    ' myF.Pic2.Visible = True
```

```
        '---Rotate around Best Find---
        Dim MAX_Theta As Int16 = 5
        Dim theta, myTheta As Double
        Dim Origin As Point
        Dim mImage3(,) As Double
        Dim bm_out As Bitmap



        ' Origin.X = wid / 2
        'Origin.Y = Hgt / 2
        Origin.X = (wid / 2) + pBest.X
        Origin.Y = (Hgt / 2) + pBest.Y
        '===Rotate and Diff Points===
        For theta = -MAX_Theta To MAX_Theta Step 0.1
            '-CRAP VB.NET FIX-
            If theta < 0 And theta > -0.1 Then
                theta = 0
            End If
            '-------Rotate Points-------
            If theta < 0 Then
                myTheta = 360 + theta
            Else
                myTheta = theta
            End If

            If CInt(myTheta) = 4.0 Then
                Debug.Write("")
            End If

            'myTheta = 4
            myCorners = Rotate_XY_Points(myCorners, myTheta, Origin)

            bm_out = New Bitmap(CInt(myImg.Width), CInt(myImg.Height))

            gr_out = Graphics.FromImage(bm_out)

            gr_out.DrawImage(myIMG_T, myCorners)


            '  myF.Pic2.Image = bm_out
            '  Exit For
            '---Scan image 3--
            mImage3 = Scan_Image(bm_out)
            R = Normalised_CrossCorrelation(mImage2, mImage3, 0, 0)
            If R > bR Then
                bTheta = myTheta

            End If


        Next
        Debug.WriteLine("Ending Rotate at " & Now())
        myVector.myT = bTheta

        '  MsgBox("BEST X = " & pBest.X & ",Best Y = " & pBest.Y & " Theta = " & bTheta &
»           " R= " & bR)
        Return myVector
    End Function

    '---Cross correlate Mask against image---
    Function Find_Whole_Image_Match() As Vector
        '---init vars---
        Dim mImage2(,), mImage(,) As Double ', mCoef(,), mBest(,) As Double
        Dim i, X, Y As Int32

        Dim TimerF As New timerform()
        Dim myImg As New Bitmap(myImage)
        Dim myImg2 As New Bitmap(myMASK)
        'Dim startP As Point
        Dim R, bR As Double
        '-Set Point-
        'startP.X = 0
        'startP.Y = 0
        'EndP.X = myImg.Width - 1
```

```
'EndP.Y = myImg.Height - 1

'----set up timer form---
'TimerF.Visible = True
'TimerF.ProgressBar.Maximum = ((myImg.Width * myImg.Height) * 2) + 1
'TimerF.ProgressBar.Increment(1)

'---Scan image 1 --
mImage = Scan_Image(myImg)
' mImage = Normalise(mImage, startP, EndP)
'---Scan Image 2---
mImage2 = Scan_Image(myImg2)
' mImage2 = Normalise(mImage2, startP, EndP)

Dim pImg1, pImg2, pBest As Point, iMiss, bMiss As Int32
Dim Pixels_OK As Boolean, diff, sDiff, bDiff, mDiff As Double
Dim Offset_X, Offset_Y As Int32, First_Run As Boolean = True
Dim MOA As Int16 = 1 '3 'Maximum_Offset_Ratio
Dim MOS As Int16 = 1 '10 'Maximum Offset Step



For Offset_Y = -UBound(mImage, 2) / MOA To UBound(mImage, 2) / MOA Step MOS
    For Offset_X = -UBound(mImage) / MOA To UBound(mImage) / MOA Step MOS

        '-RESET VARS-
        iMiss = 0
        ' sDiff = 0
        If Offset_Y >= -5 And Offset_X >= -63 Then
            Debug.Write("")
        End If

        ' For Y = (0 + Offset_Y) To (UBound(mImage, 2) + Offset_Y)
        ' For X = (0 + Offset_X) To (UBound(mImage) + Offset_X)


        '--set vars--
        '---image 1 co ords never changes this way----
        ' pImg1.X = X - Offset_X '+ (UBound(mImage) / MOA)
        'pImg1.Y = Y - Offset_Y '+ (UBound(mImage, 2) / MOA)

        '---image 2 moves about----
        ' pImg2.X = X
        ' pImg2.Y = Y

        Pixels_OK = False



        '--Count missing Pixels---
        If (pImg1.X Or pImg2.X) < 0 Then
            iMiss = iMiss + 1
        ElseIf (pImg1.X Or pImg2.X) > UBound(mImage) Then
            iMiss = iMiss + 1
        ElseIf (pImg1.Y Or pImg2.Y) < 0 Then
            iMiss = iMiss + 1
        ElseIf (pImg1.Y Or pImg2.Y) > UBound(mImage, 2) Then
            iMiss = iMiss + 1
        Else
            Pixels_OK = True
        End If

        '--Don't include very dark pixels---
        If Pixels_OK = True Then
            If mImage(pImg1.X, pImg1.Y) < 0.1 Then
                ' iMiss = iMiss + 1
                Pixels_OK = False
            ElseIf mImage2(pImg2.X, pImg2.Y) < 0.1 Then
                ' iMiss = iMiss + 1
                Pixels_OK = False
            End If
        End If
        '--Difference--
        If Pixels_OK = True Then
            '    diff = Abs(mImage(pImg1.X, pImg1.Y) - mImage2(pImg2.X, pImg2.Y))
            '    sDiff = sDiff + diff
```

```
                'End If
                '-Cross correlation-
                R = Normalised_CrossCorrelation(mImage, mImage2, X, Y)
            End If
            'TimerF.ProgressBar.Increment(1)
            'TimerF.Refresh()
            '  Next X
            ' Next Y

            '--calc mean diff--
            ' mDiff = sDiff / ((UBound(mImage) + 1) * (UBound(mImage, 2) + 1) - iMiss)

            '--Save best--
            ' If First_Run = True Then
            '        First_Run = False
            '         bDiff = mDiff
            ' End If

            'If iMiss < 10000 Then
            If R > bR Then
                bR = R
                pBest.X = Offset_X
                pBest.Y = Offset_Y
                bMiss = iMiss
            End If
            'End If
        Next Offset_X
    Next Offset_Y


    '---Rotate around Best Find---
    Dim MAX_Theta As Int16 = 5
    Dim theta, bTheta, myTheta As Int16
    Dim m As New Drawing.Drawing2D.Matrix()
    Dim Origin As PointF
    '--Calculate Best Origin--
    Origin.X = ((UBound(mImage) + pBest.X) - (0 + pBest.X)) / 2
    Origin.Y = ((UBound(mImage) + pBest.Y) - (0 + pBest.Y)) / 2

    '--Build array of points to rotate--
    Dim arrP(0) As Point

    For Y = (0 + pBest.Y) To (UBound(mImage, 2) + pBest.Y)
        For X = (0 + pBest.X) To (UBound(mImage) + pBest.X)

            arrP(i).X = X
            arrP(i).Y = Y
            i = i + 1

            '-redim arrp-
            If i >= UBound(arrP) Then
                ReDim Preserve arrP(i)
            End If
        Next X
    Next Y

    '--remove last i as empty--
    ReDim Preserve arrP(i - 1)
    '===Rotate and Diff Points===
    For theta = -MAX_Theta To MAX_Theta
        '-------Rotate Points-------
        If myTheta < 0 Then
            theta = 360 + theta
        Else
            myTheta = theta
        End If

        m.RotateAt(myTheta, Origin)
        m.TransformPoints(arrP)

        ' --- Reset(vars)---
        sDiff = 0
        i = 0
        iMiss = 0

        '---comapare diff---
        For Y = 0 To UBound(mImage, 2)
```

```
                For X = 0 To UBound(mImage)

                    Pixels_OK = False

                    '--Count missing Pixels---
                    If (arrP(i).X) < 0 Then
                        iMiss = iMiss + 1
                    ElseIf (arrP(i).X) > UBound(mImage) Then
                        iMiss = iMiss + 1
                    ElseIf (arrP(i).Y) < 0 Then
                        iMiss = iMiss + 1
                    ElseIf (arrP(i).Y) > UBound(mImage, 2) Then
                        iMiss = iMiss + 1
                    Else
                        Pixels_OK = True
                    End If

                    '--Don't include very dark pixels---
                    If Pixels_OK = True Then
                        If mImage2(arrP(i).X, arrP(i).Y) < 0.1 Then
                            Pixels_OK = False
                        ElseIf mImage2(arrP(i).X, arrP(i).Y) < 0.1 Then
                            Pixels_OK = False
                        End If
                    End If

                    '--Diff--
                    If Pixels_OK = True Then
                        diff = Abs(mImage(X, Y) - mImage2(arrP(i).X, arrP(i).Y))
                        sDiff = sDiff + diff

                    End If

                    i = i + 1
                Next X
            Next Y

            '--calc mean diff--
            mDiff = sDiff / ((UBound(mImage) + 1) * (UBound(mImage, 2) + 1) - iMiss)

            '--Save Best--
            If mDiff < bDiff Then
                bDiff = mDiff
                bTheta = myTheta
                bMiss = iMiss
            End If
        Next theta


        MsgBox("BEST X = " & pBest.X & ",Best Y = " & pBest.Y & " Missing = " & bMiss & "
»           Theta = " & bTheta)
        '---destroy timer form----
        '  TimerF = Nothing

        '---draw---
        'Dim myPiccyG As Graphics
        ' myPiccyG = Graphics.FromImage(myImg2)
        'myImg = InVert_IMG(myImg)
        'myPiccyG.DrawImage(myImg, pBest)
        Dim myVector As New Vector()
        myVector.myP.X = pBest.X
        myVector.myP.Y = pBest.Y
        myVector.myT = bTheta


        Return myVector
    End Function

    '---invert image function---
    Private Function InVert_IMG(ByVal imgBefore As Image) As Image
        '======Init Vars=====

        Dim bm As New Bitmap(imgBefore)
        Dim X As Integer
        Dim Y As Integer
        Dim r As Integer
```

```
        Dim g As Integer
        Dim b As Integer

        '=====Invert Colors====
        For X = 0 To bm.Width - 1
            For Y = 0 To bm.Height - 1
                r = 255 - bm.GetPixel(X, Y).R
                g = 255 - bm.GetPixel(X, Y).G
                b = 255 - bm.GetPixel(X, Y).B
                bm.SetPixel(X, Y, Color.FromArgb(128, r, g, b))
            Next Y
        Next X
        '====Return Image=====
        InVert_IMG = bm


    End Function

    Function Normalise(ByRef arrInput(,) As Double, ByVal startP As Point, ByVal endP As
»       Point) As Double(,)
        '----init vars----
        Dim ix, iY As Integer, SumTotal As Double, WhatToDo As String
        Dim EndpointX, EndpointY As Int32
        Dim arrOutput(UBound(arrInput), UBound(arrInput, 2)) As Double

        '---set end points to scan to in arrays---
        If endP.Equals(startP) = False Then
            EndpointX = endP.X - startP.X
            EndpointY = endP.Y - startP.Y
        Else
            EndpointX = UBound(arrInput)
            EndpointY = UBound(arrInput, 2)
        End If

        '===========Normalise the Data=========
        '#Work out whether to Multiply up or Down to get to 100%#
        For ix = 0 To EndpointX
            For iY = 0 To EndpointY
                SumTotal = SumTotal + arrInput(ix + startP.X, iY + startP.Y)
            Next iY
        Next ix

        If SumTotal > 100 Then
            WhatToDo = "divide"
        Else
            WhatToDo = "multiply"
        End If

        SumTotal = SumTotal / 100

        '####################Normalise it#######################
        For ix = 0 To EndpointX
            For iY = 0 To EndpointY

                '------Divide 0/0 Fix------------
                If arrInput(ix + startP.X, iY + startP.Y) = 0 And SumTotal = 0 Then Exit
»                   For

                If WhatToDo = "multiply" Then
                    If SumTotal < 1 Then
                        arrOutput(ix + startP.X, iY + startP.Y) = arrInput(ix + startP.X,
»                           iY + startP.Y) / SumTotal
                    Else
                        arrOutput(ix + startP.X, iY + startP.Y) = arrInput(ix + startP.X,
»                           iY + startP.Y) * SumTotal
                    End If

                Else
                    arrOutput(ix + startP.X, iY + startP.Y) = arrInput(ix + startP.X, iY +
»                        startP.Y) / SumTotal
                End If

            Next iY
        Next ix
        '#########################################################
```

```
        '=====Output for FLX======
        Normalise = arrOutput

    End Function

    Public Sub New(ByVal a As Image, ByVal b As Image)
        myMASK = b
        myImage = a
        MAX_Rows = CInt(MAX_ARRAY_SIZE / myImage.Width)
    End Sub

    Function Rotate_XY_Points(ByVal myPs() As Point, ByRef Angle As Double, ByVal origin
»       As Point) As Point()
        '------Init Vars------
        Rotate_XY_Points = myPs
        ' Dim i As Int32
        Dim m As New Drawing.Drawing2D.Matrix()
        Dim OriginF As PointF
        OriginF.X = origin.X
        OriginF.Y = origin.Y
        '-------Rotate Points-------
        m.RotateAt(Angle, OriginF)
        m.TransformPoints(Rotate_XY_Points)



    End Function

    Private Function Translate_XY_Points(ByRef myPoints() As Point, ByVal Offset As Point)
»        As Point()
        '-----Init Vars----
        Dim m As New Drawing.Drawing2D.Matrix()


        '---Translate Image---
        m.Translate(Offset.X, Offset.Y)
        m.TransformPoints(myPoints)
        Translate_XY_Points = myPoints

    End Function
    Private Function Rotate_an_Image(ByVal picSource As Image, ByVal Angle As Double) As
»       Image
        ' Copy the output bitmap from the source image.
        Dim bm_in As New Bitmap(picSource)

        ' Make an array of points defining the
        ' image's corners.
        Dim wid As Single = bm_in.Width
        Dim hgt As Single = bm_in.Height
        Dim corners As Point() = { _
            New Point(0, 0), _
            New Point(wid, 0), _
            New Point(0, hgt), _
            New Point(wid, hgt)}

        ' Translate to center the bounding box at the origin.
        Dim cx As Single = wid / 2
        Dim cy As Single = hgt / 2
        Dim i As Long
        For i = 0 To 3
            corners(i).X -= cx
            corners(i).Y -= cy
        Next i

        ' Rotate.
        Dim theta As Single = Single.Parse(Angle) * PI _
            / 180.0
        Dim sin_theta As Single = Sin(theta)
        Dim cos_theta As Single = Cos(theta)
        Dim X As Single
        Dim Y As Single
        For i = 0 To 3
```

```
            X = corners(i).X
            Y = corners(i).Y
            corners(i).X = X * cos_theta + Y * sin_theta
            corners(i).Y = -X * sin_theta + Y * cos_theta
        Next i

        ' Translate back to original position
        cx = wid / 2
        cy = (hgt / 2)

        For i = 0 To 3
            corners(i).X += cx
            corners(i).Y += cy
        Next i

        '' Translate so X >= 0 and Y >=0 for all corners.
        Dim xmin As Single = corners(0).X
        Dim ymin As Single = corners(0).Y
        'For i = 1 To 3
        '    If xmin > corners(i).X Then xmin = corners(i).X
        '    If ymin > corners(i).Y Then ymin = corners(i).Y
        'Next i
        'For i = 0 To 3
        '    corners(i).X -= xmin
        '    corners(i).Y -= ymin
        'Next i



        ' Create an output Bitmap and Graphics object.
        Dim bm_out As New Bitmap(CInt(-2 * xmin), CInt(-2 * _
            ymin))
        Dim gr_out As Graphics = Graphics.FromImage(bm_out)

        ' Drop the last corner lest we confuse DrawImage,
        ' which expects an array of three corners.
        ReDim Preserve corners(2)

        ' Draw the result onto the output Bitmap.
        gr_out.DrawImage(bm_in, corners)

        ' Display the result.
        Rotate_an_Image = bm_out
    End Function
End Class
```

ARID CVessels.vb

```
Imports System.Math
Public Class CVessels
    Dim iX, iY As Int32

    Dim myPic As Bitmap
    Dim Points_H(0) As Point
    Dim Points_V(0) As Point
    Dim Points_Best(0) As Point
    Dim nGrp_Size As String 'used for grp function
    Dim nJoin As Int32 'used for join function
    Private Function GET_Ave_Intensity() As Double
        '---Init Vars---
        Dim dblBright, sum, ave As Double
        ' Dim newHue, newlum, newSat As Double
        ' Dim Colour_converted_Pixel As HLSRGB
        Dim missed As Int64
        '---Scan for Candidates---
        For iY = 0 To myPic.Height - 1
            For iX = 0 To myPic.Width - 1
                dblBright = myPic.GetPixel(iX, iY).GetBrightness
                If dblBright > 0.02 Then
                    sum = sum + dblBright
                Else
                    missed = missed + 1
                End If
            Next iX
        Next iY
        ave = sum / (((myPic.Height - 1) * (myPic.Width - 1)) - missed)

        Return ave

    End Function
    Function Get_Actual_Retina() As Rectangle
        '-get center of image
        'scan H until black found
        'repeat for v
        Dim x, y As Int64
        Dim centerP As Point
        Dim dblBright As Double
        Dim Hedge1, Hedge2, vedge1, vedge2 As Int64
        Dim Found1 As Boolean
        Dim iVariableH As Int32 = myPic.Width / 10
        Dim iVariableV As Int32 = myPic.Height / 10
        Dim Reset_Count As Int16
        '-center-
        centerP.X = myPic.Width / 2
        centerP.Y = myPic.Height / 2
        dblBright = myPic.GetPixel(centerP.X, centerP.Y).GetBrightness

        '=============Scan H==================

        '--search Right --
        x = centerP.X
        Do While Hedge1 = 0
            x = x + iVariableH
            If x > myPic.Width - 1 Then x = myPic.Width - 1 : Reset_Count = Reset_Count +
» 1
            dblBright = myPic.GetPixel(x, centerP.Y).GetBrightness

            If Reset_Count = 2 Then Exit Do

            '-step down search pattern as it gets closer to an edge-
            If dblBright < 0.02 Then
                Found1 = True
                x = x - iVariableH
                If iVariableH > 2 Then
                    iVariableH = iVariableH - (iVariableH / 5)
                Else
                    Hedge1 = x
                    Exit Do
                End If
            Else
                If Found1 = True Then
                    'we have found an edge and now moved past it'
                    Hedge1 = x '- iVariableH
```

```
                    Exit Do
                End If
            End If
            '-special escape clause-
            If x = myPic.Width - 1 And Hedge1 = 0 Then Exit Do
        Loop

        '--search LEFT --
        x = centerP.X
        Found1 = False
        Reset_Count = 0
        Do While Hedge2 = 0
            x = x - iVariableH
            If x < 0 Then x = iVariableH : Reset_Count = Reset_Count + 1
            dblBright = myPic.GetPixel(x, centerP.Y).GetBrightness

            If Reset_Count = 2 Then Exit Do

            '-step down search pattern as it gets closer to an edge-
            If dblBright < 0.02 Then
                Found1 = True
                x = x + iVariableH
                If iVariableH > 2 Then
                    iVariableH = iVariableH - (iVariableH / 5)
                Else
                    Hedge2 = x
                    Exit Do
                End If
            Else
                If Found1 = True Then
                    'we have found an edge and now moved past it'
                    Hedge2 = x '+ iVariableH
                    Exit Do
                End If
            End If
            '-special escape clause-
            If x = myPic.Width - 1 And Hedge2 = 0 Then Exit Do
        Loop

        '=============Scan V==================

        '--search Down --
        y = centerP.Y
        Found1 = False
        Reset_Count = 0
        Do While vedge1 = 0
            y = y + iVariableV
            If y > myPic.Height - 15 Then y = myPic.Height - 15 : Reset_Count =
»               Reset_Count + 1
            dblBright = myPic.GetPixel(centerP.X, y).GetBrightness

            If Reset_Count = 2 Then Exit Do
            '-step down search pattern as it gets closer to an edge-
            If dblBright < 0.025 Then
                '-Special check for images that have white at the bottom-
                Found1 = True
                y = y - iVariableV
                If iVariableV > 2 Then
                    iVariableV = iVariableV - (iVariableV / 5)
                Else
                    vedge1 = y
                    Exit Do
                End If
            Else
                If Found1 = True Then
                    'we have found an edge and now moved past it'
                    vedge1 = y '+ iVariableV
                    Exit Do
                End If
            End If
Skippy:
            '-special escape clause-
            If y = myPic.Height - 1 And vedge1 = 0 Then Exit Do
        Loop

        '--search up --
```

```
        y = centerP.Y
        Found1 = False
        Reset_Count = 0
        Do While vedge2 = 0
            y = y - iVariableV
            If y < 0 Then y = iVariableV : Reset_Count = Reset_Count + 1
            dblBright = myPic.GetPixel(centerP.X, y).GetBrightness

            If Reset_Count = 2 Then Exit Do

            '-step down search pattern as it gets closer to an edge-
            If dblBright < 0.02 Then
                Found1 = True
                y = y + iVariableV
                If iVariableV > 2 Then
                    iVariableV = iVariableV - (iVariableV / 5)
                Else
                    vedge2 = y
                    Exit Do
                End If
            Else
                If Found1 = True Then
                    'we have found an edge and now moved past it'
                    vedge2 = y '+ iVariableV
                    Exit Do
                End If
            End If
            '-special escape clause-
            If y = myPic.Height - 1 And vedge2 = 0 Then Exit Do
        Loop

        Dim wid, hig As Int64
        wid = Abs(Hedge1 - Hedge2)
        hig = Abs(vedge1 - vedge2)

        If wid = 0 Then wid = myPic.Width - 1
        If hig = 0 Then hig = myPic.Height - 1
        Get_Actual_Retina = New Rectangle(Hedge2, vedge2, wid, hig)

    End Function

    Private Function Scan_H() As Point()
        Debug.WriteLine("Scan H Start: " & Now())
        '---Init Vars---
        Dim dblBright As Double
        Dim dblDiff, dblEDGE As Double
        Dim i, Pixels_MatchedX, Pixels_MatchedY, Y_Dir As Int32
        Dim Too_Many_Found, Too_few_Found As Boolean
        dblEDGE = 0.1 'Edge detection threshold

        '--Define segmentation threshold based on average intensity--
Too_Many_Few_Restart_Point:
        If Too_Many_Found = True Then
            dblEDGE = dblEDGE * 1.25
            Too_Many_Found = False
        End If

        If Too_few_Found = True Then
            dblEDGE = dblEDGE / 1.2
            Too_few_Found = False
        End If


        '#Exclusion zones
        Dim xZone, xZone2 As Point
        xZone.X = 0.2 * myPic.Width
        xZone.Y = 0.3 * myPic.Height
        '#exclusion zones sides
        xZone2.X = 0.05 * myPic.Width
        xZone2.Y = 0.05 * myPic.Height
        '---Find Pixels---
        For iY = 6 To myPic.Height - 7 Step 3 '+-5 as we check +-5 in Y Direction
            For iX = 0 To myPic.Width - 5 'Minus 5 as we check +4 pixels on X axis
                dblBright = myPic.GetPixel(iX, iY).GetBrightness


                '--Escape--
```

```
If dblBright < 0.1 Then GoTo Skip_Pixel
'--Avoid scaning corners of images-
If iX < xZone.X And iY < xZone.Y Then
    GoTo Skip_Pixel
ElseIf iX > myPic.Width - xZone.X And iY < xZone.Y Then
    GoTo Skip_Pixel
ElseIf iX < xZone.X And iY > myPic.Height - xZone.Y Then
    GoTo Skip_Pixel
ElseIf iX > myPic.Width - xZone.X And iY > myPic.Height - xZone.Y Then
    GoTo Skip_Pixel
End If

'--Avoid sides of images--
If iX < xZone2.X Then
    GoTo Skip_Pixel
ElseIf iX > myPic.Width - xZone2.X Then
    GoTo Skip_Pixel
ElseIf iY < xZone2.Y Then
    GoTo Skip_Pixel
ElseIf iY > myPic.Height - xZone2.Y Then
    GoTo Skip_Pixel
End If

'---Find 4 Pixels in a row---
'  If dblBright < AveBright Then
For i = 1 To 4
    dblDiff = Abs(myPic.GetPixel(iX + i, iY).GetBrightness - dblBright)
    '--If Diff < 5% then inc counter--
    If (dblDiff / dblBright) < 0.05 Then '0.13 Then '
        Pixels_MatchedX = Pixels_MatchedX + 1
    End If

Next i
'Debug.WriteLine(Pixels_MatchedX)

'--check 6 either way in Y direction--
If Pixels_MatchedX = 4 Then
    For Y_Dir = -6 To 6
        If myPic.GetPixel(iX, iY + Y_Dir).GetBrightness > 0.1 Then

            dblDiff = Abs(myPic.GetPixel(iX, iY + Y_Dir).GetBrightness -
» dblBright)


            If (dblDiff / dblBright) > dblEDGE Then
                Pixels_MatchedY = Pixels_MatchedY + 1
            End If
        End If
    Next Y_Dir



End If

'--Save good Pixels--
If Pixels_MatchedX = 4 Then
    If Pixels_MatchedY > 2 Then
        '--inc array--
        If Points_H(UBound(Points_H)).IsEmpty = False Then
            ReDim Preserve Points_H(UBound(Points_H) + 1)
        End If
        '--Save--
        Points_H(UBound(Points_H)).X = iX + 2
        Points_H(UBound(Points_H)).Y = iY

        '--Set Vars--
        Pixels_MatchedX = 0
        Pixels_MatchedY = 0
        iX = iX + 4
    End If
End If
Pixels_MatchedX = 0
Pixels_MatchedY = 0
If UBound(Points_H) > 3000 Then
    Too_Many_Found = True
    ReDim Points_H(0)
```

```
                GoTo Too_Many_Few_Restart_Point

            End If

Skip_Pixel:
        Next iX
    Next iY
exit_Search:
    If UBound(Points_H) < 1500 Then
        Too_few_Found = True
        ReDim Points_H(0)
        GoTo Too_Many_Few_Restart_Point
    End If
    Scan_H = Points_H
    '--clean up--
    Debug.WriteLine("Scan H End: " & Now())
End Function

Private Function Scan_V() As Point()
    Debug.WriteLine("Scan V Start: " & Now())
    '---Init Vars---
    Dim dblBright As Double
    Dim dblDiff, dblEdge As Double
    Dim i, Pixels_MatchedX, Pixels_MatchedY, x_Dir As Int32
    Dim Too_Many_Found, Too_Few_Found As Boolean
    dblEdge = 0.1
    '--Define segmentation threshold based on average intensity--
Too_Many_Few_Restart_Point:
    If Too_Many_Found = True Then
        dblEdge = dblEdge * 1.25
        Too_Many_Found = False
    End If

    If Too_Few_Found = True Then
        dblEdge = dblEdge / 1.2
        Too_Few_Found = False
    End If



    '#Exclusion zones corners
    Dim xZone, xZone2 As Point
    xZone.X = 0.2 * myPic.Width
    xZone.Y = 0.3 * myPic.Height
    '#exclusion zones sides
    xZone2.X = 0.05 * myPic.Width
    xZone2.Y = 0.05 * myPic.Height
    '---Find Pixels---
    For iX = 6 To myPic.Width - 7 Step 3
        For iY = 0 To myPic.Height - 5

            dblBright = myPic.GetPixel(iX, iY).GetBrightness

            '--Escape--
            If dblBright < 0.1 Then GoTo Skip_Pixel
            '--Avoid scaning corners of images-
            If iX < xZone.X And iY < xZone.Y Then
                GoTo Skip_Pixel
            ElseIf iX > myPic.Width - xZone.X And iY < xZone.Y Then
                GoTo Skip_Pixel
            ElseIf iX < xZone.X And iY > myPic.Height - xZone.Y Then
                GoTo Skip_Pixel
            ElseIf iX > myPic.Width - xZone.X And iY > myPic.Height - xZone.Y Then
                GoTo Skip_Pixel
            End If

            '--Avoid sides of images--
            If iX < xZone2.X Then
                GoTo Skip_Pixel
            ElseIf iX > myPic.Width - xZone2.X Then
                GoTo Skip_Pixel
            ElseIf iY < xZone2.Y Then
                GoTo Skip_Pixel
            ElseIf iY > myPic.Height - xZone2.Y Then
                GoTo Skip_Pixel
            End If
```

```
                '---Find 5 Pixels in a row---
                'If dblBright < AveBright Then
                For i = 1 To 4
                    dblDiff = Abs(myPic.GetPixel(iX, iY + i).GetBrightness - dblBright)
                    '--If Diff > 10% then inc counter--
                    If (dblDiff / dblBright) < 0.05 Then '< 0.13 Then '
                        Pixels_MatchedY = Pixels_MatchedY + 1
                    End If

                Next i

                '--check 5 either way in Y direction--
                If Pixels_MatchedY = 4 Then
                    For x_Dir = -6 To 6
                        dblDiff = Abs(myPic.GetPixel(iX + x_Dir, iY).GetBrightness -
» dblBright)
                        If (dblDiff / dblBright) > dblEdge Then
                            Pixels_MatchedX = Pixels_MatchedX + 1
                        End If

                    Next x_Dir



                End If

                '--Save good Pixels--
                If Pixels_MatchedY = 4 Then
                    If Pixels_MatchedX > 2 Then
                        '--inc array--
                        If Points_V(UBound(Points_V)).IsEmpty = False Then
                            ReDim Preserve Points_V(UBound(Points_V) + 1)
                        End If
                        '--Save--
                        Points_V(UBound(Points_V)).X = iX
                        Points_V(UBound(Points_V)).Y = iY + 2

                        '--Set Vars--
                        Pixels_MatchedX = 0
                        Pixels_MatchedY = 0
                        iY = iY + 4
                    End If
                End If
                'If UBound(Points_V) = 999 Then GoTo exit_search
                Pixels_MatchedX = 0
                Pixels_MatchedY = 0

                If UBound(Points_V) > 3000 Then
                    Too_Many_Found = True
                    ReDim Points_V(0)
                    GoTo Too_Many_Few_Restart_Point

                End If

Skip_Pixel:
            Next iY
        Next iX
exit_Search:
        If UBound(Points_V) < 1500 Then
            Too_Few_Found = True
            ReDim Points_V(0)
            GoTo Too_Many_Few_Restart_Point
        End If
        Scan_V = Points_V
        '--clean up--
        Debug.WriteLine("Scan V End: " & Now())
    End Function

    Private Sub Grp_Pixels()
        Debug.WriteLine("Grouping Start: " & Now())
        '--init Vars--
        Dim i, icurr, Pixel_Match As Int32
        Dim CurrP As Point
        Dim arrCombine(UBound(Points_H) + UBound(Points_V)) As Point

        '--Combine Arrays--
```

```
        For i = 0 To UBound(Points_H)
            arrCombine(i) = Points_H(i)
        Next i

        For i = 0 To UBound(Points_V)
            arrCombine(i + UBound(Points_H)) = Points_V(i)
        Next i

        '  )

        '--find grp size to use-
        'Dim aveBright As Double
        Dim grpsize As Int32

        If nGrp_Size = "" Then
            'aveBright = GET_Ave_Intensity()
            'Select Case aveBright
            '    Case Is < 0.2 : grpsize = 6
            '    Case 0.2 To 0.3 : grpsize = 5
            '    Case 0.3 To 0.4 : grpsize = 6
            '    Case 0.4 To 0.5 : grpsize = 6 '4
            '    Case 0.5 To 0.6 : grpsize = 2
            '    Case Else : grpsize = 6
            'End Select
            grpsize = 7

        Else
            grpsize = CInt(nGrp_Size)
        End If

        '--Look for neighbour pixels--
        For icurr = 0 To UBound(arrCombine) 'Curr Pixel Loop
            CurrP = arrCombine(icurr)

            For i = 0 To UBound(arrCombine) 'Array to Search Loop
                For iX = -5 To +5
                    '-Within 2 in X dir?-
                    If CurrP.X = arrCombine(i).X + iX Then
                        '-Within 2 in Y dir?-
                        For iY = -5 To +5
                            If CurrP.Y = arrCombine(i).Y + iY Then
                                Pixel_Match = Pixel_Match + 1
                                GoTo Get_next_Pixel
                            End If

                        Next iY

                    End If

                Next iX
Get_next_Pixel:
            Next i


            '--group--
            If Pixel_Match >= grpsize Then

                '--inc array--
                If Points_Best(UBound(Points_Best)).IsEmpty = False Then
                    ReDim Preserve Points_Best(UBound(Points_Best) + 1)
                End If
                '--save pixel--
                Points_Best(UBound(Points_Best)) = CurrP
            End If


            Pixel_Match = 0
        Next icurr

        Debug.Write(UBound(Points_Best))
        Debug.WriteLine("Grouping End: " & Now())
    End Sub

    Function Interpolate_Points_Best()
        Dim myP(0) As Point
        Dim ix, iy, i, j, distance, icount, jcount As Int64
```

```
        distance = 9
        j = 0

        For i = 0 To UBound(Points_Best) - 2
            If Abs(Points_Best(i + 1).X - Points_Best(i).X) < distance Then
                If Abs(Points_Best(i + 1).Y - Points_Best(i).Y) < distance Then

                    For icount = 1 To distance
                        For jcount = 1 To distance
                            myP(j).X = Points_Best(i).X + icount
                            myP(j).Y = Points_Best(i).Y + jcount
                            j = j + 1
                            ReDim Preserve myP(j)
                        Next
                    Next

                End If
            End If
        Next

        Debug.Write(UBound(myP))
        Points_Best = myP
    End Function

    Function Get_Best_Pixels() As Point()
        '---Init Vars--
        Dim i As Int32
        Dim Pixels_Matched1(), Pixels_Matched2() As Point
        Dim TForm As New timerform()
        TForm.ProgressBar.Maximum = 50
        TForm.ProgressBar.Value = 0
        TForm.Show()

        Call Scan_H()
        TForm.ProgressBar.Value = 25
        Call Scan_V()
        TForm.ProgressBar.Value = 50
        Call Grp_Pixels()
        'TForm.ProgressBar.Value = 100
        ' Interpolate_Points_Best()
        TForm.Hide()
        TForm = Nothing
        Return Points_Best
    End Function


    Function Draw(ByVal Draw_What As Int16) As Image
        Dim i As Int32
        Dim myPiccy As New Bitmap(myPic)

        '--snake vessels---
        'For i = 0 To UBound(Points_Best)
        '    Dim myS As New CSnake(Points_Best(i), myPiccy)
        '    myS.Trace_Path()
        '    If i = 99 Then Exit For
        'Next i

        If Draw_What = 0 Then 'HORIZONTAL
            '---Draw Pixels--
            For i = 0 To UBound(Points_H)
                myPiccy.SetPixel(Points_H(i).X, Points_H(i).Y, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X + 1, Points_H(i).Y, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X - 1, Points_H(i).Y, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X, Points_H(i).Y + 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X, Points_H(i).Y - 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X + 1, Points_H(i).Y - 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X - 1, Points_H(i).Y + 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X + 1, Points_H(i).Y + 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X - 1, Points_H(i).Y - 1, Color.Blue)
            Next i

        ElseIf Draw_What = 1 Then 'VERTICAL
            '---Draw Pixels--
            For i = 0 To UBound(Points_V)
```

```
                myPiccy.SetPixel(Points_V(i).X, Points_V(i).Y, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X + 1, Points_V(i).Y, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X - 1, Points_V(i).Y, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X, Points_V(i).Y + 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X, Points_V(i).Y - 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X + 1, Points_V(i).Y - 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X - 1, Points_V(i).Y + 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X + 1, Points_V(i).Y + 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X - 1, Points_V(i).Y - 1, Color.LimeGreen)

            Next i

        ElseIf Draw_What = 2 Then 'GROUPED
            ' ---Draw Pixels--
            If UBound(Points_Best) > 1 Then
                For i = 0 To UBound(Points_Best)
                    myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y, Color.Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y + 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y - 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y - 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y + 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y + 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y - 1, Color.
» Turquoise)

                Next i
            End If
        Else ' ALL
            For i = 0 To UBound(Points_H)
                myPiccy.SetPixel(Points_H(i).X, Points_H(i).Y, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X + 1, Points_H(i).Y, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X - 1, Points_H(i).Y, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X, Points_H(i).Y + 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X, Points_H(i).Y - 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X + 1, Points_H(i).Y - 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X - 1, Points_H(i).Y + 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X + 1, Points_H(i).Y + 1, Color.Blue)
                myPiccy.SetPixel(Points_H(i).X - 1, Points_H(i).Y - 1, Color.Blue)

            Next i
            For i = 0 To UBound(Points_V)
                myPiccy.SetPixel(Points_V(i).X, Points_V(i).Y, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X + 1, Points_V(i).Y, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X - 1, Points_V(i).Y, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X, Points_V(i).Y + 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X, Points_V(i).Y - 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X + 1, Points_V(i).Y - 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X - 1, Points_V(i).Y + 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X + 1, Points_V(i).Y + 1, Color.LimeGreen)
                myPiccy.SetPixel(Points_V(i).X - 1, Points_V(i).Y - 1, Color.LimeGreen)


            Next i
            If UBound(Points_Best) > 1 Then
                For i = 0 To UBound(Points_Best)
                    myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y, Color.Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y + 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y - 1, Color.
» Turquoise)
                    myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y - 1, Color.
```

```
»                   Turquoise)
                myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y + 1, Color.
»                   Turquoise)
                myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y + 1, Color.
»                   Turquoise)
                myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y - 1, Color.
»                   Turquoise)


            Next i
        End If
    End If


    ''---Set Vars---




    Return myPiccy
End Function

Function Create_Dot_Image() As Image
    '-Init Vars-
    Dim myPiccy As New Bitmap(myPic.Width, myPic.Height)
    Dim i As Int32
    '-draw black background-
    Dim myG As Graphics = Graphics.FromImage(myPiccy)
    Dim pn As Pen = New Pen(Color.Black)
    Dim myregion = New Region()
    Dim myRect = New Rectangle(0, 0, myPic.Width, myPic.Height)

    myG.DrawRectangle(pn, myRect)

    '-apply grp dots found-
    If UBound(Points_Best) > 1 Then
        For i = 0 To UBound(Points_Best)
            myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y, Color.White)
            myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y, Color.White)
            myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y, Color.White)
            myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y + 1, Color.White)
            myPiccy.SetPixel(Points_Best(i).X, Points_Best(i).Y - 1, Color.White)
            myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y - 1, Color.White)
            myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y + 1, Color.White)
            myPiccy.SetPixel(Points_Best(i).X + 1, Points_Best(i).Y + 1, Color.White)
            myPiccy.SetPixel(Points_Best(i).X - 1, Points_Best(i).Y - 1, Color.White)

        Next i
    End If
    Return myPiccy
End Function

Function Join_Up() As Image
    Dim i, j, iFill, iloopX, iLoopY, XStep, YStep As Int32
    Dim myPiccy As New Bitmap(myPic)
    Dim arrFill_IN(1) As Point
    Dim N As Int32 = nJoin



    '---Find Neighbouring grped pixels----
    For i = 0 To UBound(Points_Best)
        For j = 0 To UBound(Points_Best)
            '==skip identical Co-ords==
            If Points_Best(i).Equals(Points_Best(j)) = False Then
                '==Is it Within N pixels on X axis==
                If Points_Best(i).X >= (Points_Best(j).X - N) And Points_Best(i).X <=
»                   (Points_Best(j).X + N) Then
                    '==Is it Within N pixels on Y axis==
                    If Points_Best(i).Y >= (Points_Best(j).Y - N) And Points_Best(i).Y
»                       <= (Points_Best(j).Y + N) Then
                        iX = Points_Best(i).X - Points_Best(j).X
```

```
                            iY = Points_Best(i).Y - Points_Best(j).Y



                            '-Resize array-
                            If iFill >= UBound(arrFill_IN) Then
                                ReDim Preserve arrFill_IN(iFill + 1)
                            End If



                            '-Add coords of start and end of line-

                            arrFill_IN(iFill).X = Points_Best(i).X
                            arrFill_IN(iFill).Y = Points_Best(i).Y
                            arrFill_IN(iFill + 1).X = Points_Best(j).X
                            arrFill_IN(iFill + 1).Y = Points_Best(j).Y
                            iFill = iFill + 2



                        End If
                    End If
                End If
            Next j

        Next i


        '---draw results---
        Dim myG As Graphics = Graphics.FromImage(myPiccy)
        For i = 0 To UBound(arrFill_IN) Step 2
            myG.DrawLine(Pens.WhiteSmoke, arrFill_IN(i), arrFill_IN(i + 1))
        Next i


        Return myPiccy
    End Function 'PRETTY but no real use

    Function Find_Paths() As Image
        '---Init Vars---
        Dim iPoint, jPoint, i As Int32
        Dim Diff, BestDiff, BestJ As Int32
        Dim FirstRun As Boolean
        Dim arrPath() As Point
        Dim arrBad(0) As Point
        Dim Partition_StartP, EndP As Point
        '---Take Any Point---
        For iPoint = 0 To UBound(Points_Best)
            Partition_StartP = Points_Best(iPoint)
ReFind:
            '---Reset Vars--
            FirstRun = True

            '--Take Point closest to it--
            For jPoint = 0 To UBound(Points_Best)
                If Partition_StartP.Equals(Points_Best(jPoint)) = False Then
                    Diff = Abs(Points_Best(iPoint).X - Points_Best(jPoint).X)
                    Diff = Diff + Abs(Points_Best(iPoint).Y - Points_Best(jPoint).Y)

                    If FirstRun = True Then
                        '--Check Point Not already tried--
                        For i = 0 To UBound(arrBad)
                            If Points_Best(jPoint).Equals(arrBad(i)) Then
                                GoTo Exit_IF
                            End If
                        Next i
                        '---SavePoint---
                        BestDiff = Diff
                        BestJ = jPoint
                        FirstRun = False
                    Else
                        If Diff < BestDiff Then
                            '--Check Point Not already tried--
                            For i = 0 To UBound(arrBad)
```

```
                                If Points_Best(jPoint).Equals(arrBad(i)) Then
                                    GoTo Exit_IF
                                End If
                            Next i
                            '--Save Point--
                            BestDiff = Diff
                            BestJ = jPoint
                        End If
                    End If
                End If
Exit_IF:
            Next jPoint
            '--Can we trace a path ?--
            EndP = Points_Best(BestJ)
            arrPath = Snake_Between_points(Partition_StartP, EndP)


            '---If Path Found what is that path ----
            If UBound(arrPath) <> 1 Then
                'need to save this array once completed
                For i = 0 To UBound(arrPath)
                    myPic.SetPixel(arrPath(i).X, arrPath(i).Y, Color.WhiteSmoke)

                Next i

                arrBad(UBound(arrBad)) = Partition_StartP
                Partition_StartP = EndP
            Else
                '--If No take Next nearest point & repeat---
                arrBad(UBound(arrBad)) = EndP

            End If
            '---Loop Back into Array---
            ReDim Preserve arrBad(UBound(arrBad) + 1)
            'WORK OUT A WAY TO TELL DIFF BETWEEN FALSE PATH AND END OF PATH
            GoTo ReFind
        Next iPoint
    End Function

    Function Snake_Between_points(ByVal Partition_StartP As Point, ByVal endP As Point) As
»        Point()

        '---init Vars---
        Dim Point_Depth As Int16 = 3
        Dim arrV As Int16 = Point_Depth * 2 - 1
        Dim negV As Int16 = 0 - Point_Depth
        Dim posV As Int16 = Point_Depth - 1
        Dim totV As Int16 = Math.Pow((Point_Depth * 2), 2) - 1

        Dim Intensity_Primary As Double
        Dim Intensity_Array(arrV, arrV) As Double     'Intensity Store
        Dim CoOrd_Array(arrV, arrV) As Point          'Coords of Intensity Store
        Dim Order_of_Match(totV, 1) As Object     'Order of Best Match to Original
        Dim Swap_array(0, 1) As Object          'Swap Array for Bubble Sort
        Dim Diff_Matrix(totV, 1) As Object     'Order of Best Match to Original

        Dim Curr_X, Curr_Y, m_Val As Int32
        Dim dblDiff, sum_I As Double, myP As Point
        Dim myPen As Pen

        Dim Path_Points(1) As Point

        Dim X_Direction, Y_Direction As Int16
        Dim NO_Path As Boolean

        'Counters
        Dim iX, iY, i, j, iTrace As Int16

        '===Calculate General Direction===
        X_Direction = Sign(endP.X - Partition_StartP.X)
        Y_Direction = Sign(endP.Y - Partition_StartP.Y)

        '--Set starting intensity---
        Path_Points(0).X = Partition_StartP.X
        Path_Points(0).Y = Partition_StartP.Y
        Intensity_Primary = myPic.GetPixel(Path_Points(0).X, Path_Points(0).Y).
```

```
»           GetBrightness

        '===Snake thru vessels===
        Do While iTrace < 100 'Path less than 100 pixels long
            '---Reset Vars---
            i = 0
            sum_I = 0
            NO_Path = False
            '---check surrounding pixels to a depth of X---
            For iX = negV To posV
                For iY = negV To posV
                    '--store Brightness--
                    Curr_X = Path_Points(UBound(Path_Points) - 1).X + iX
                    Curr_Y = Path_Points(UBound(Path_Points) - 1).Y + iY

                    '---escape clause---
                    If Curr_X < 0 Or Curr_Y < 0 Then Exit For

                    Intensity_Array(iX - negV, iY - negV) = myPic.GetPixel(Curr_X, Curr_Y)
»                       .GetBrightness
                    '--Store CoOrds--
                    myP.X = Curr_X
                    myP.Y = Curr_Y
                    CoOrd_Array(iX - negV, iY - negV) = myP


                Next iY
            Next iX
            Debug.Write("")
            '===assign Vals dependent upon match to originals position and intensity===
            '--store Difference in Intensity---
            For iX = 0 To UBound(Intensity_Array)
                For iY = 0 To UBound(Intensity_Array, 2)
                    '--store diff--
                    dblDiff = Abs(Intensity_Primary - Intensity_Array(iX, iY))

                    '--store diffs--
                    Order_of_Match(i, 0) = dblDiff
                    Order_of_Match(i, 1) = CoOrd_Array(iX, iY)
                    i = i + 1
                Next iY
            Next iX
            '---Sort Diff in Intensity---
            'Bubble Sort
            For i = 0 To UBound(Order_of_Match)
                For j = 0 To UBound(Order_of_Match)
                    If Order_of_Match(i, 0) < Order_of_Match(j, 0) Then
                        '-Store Val-
                        Swap_array(0, 0) = Order_of_Match(i, 0)
                        Swap_array(0, 1) = Order_of_Match(i, 1)

                        '-1st swap-
                        Order_of_Match(i, 0) = Order_of_Match(j, 0)
                        Order_of_Match(i, 1) = Order_of_Match(j, 1)

                        '-2nd Swap-
                        Order_of_Match(j, 0) = Swap_array(0, 0)
                        Order_of_Match(j, 1) = Swap_array(0, 1)

                    End If
                Next j
            Next i
            Debug.Write("")
            '---Is one of the top 9 least intensity differences is in same direction as wa
»               nted end point--
            Dim X_Dir, Y_Dir As Int16

            For i = 1 To 9 ' 1 to 9 as match 0 = original point
                X_Dir = Sign(Order_of_Match(i, 1).X - Path_Points(UBound(Path_Points) - 1)
»                   .X)
                Y_Dir = Sign(Order_of_Match(i, 1).Y - Path_Points(UBound(Path_Points) - 1)
»                   .Y)

                If (X_Dir = X_Direction) And (Y_Dir = Y_Direction) Then
                    'SUCCESS
                    NO_Path = False
```

```
                Path_Points(UBound(Path_Points)) = Order_of_Match(i, 1)

                '---Inc Array---
                ReDim Preserve Path_Points(UBound(Path_Points) + 1)
                Exit For
            Else
                'FAILED
                NO_Path = True
            End If
        Next i


        '--need to return HERE if No path between pixels--
        '---create and return Zero length Array---
        If NO_Path = True Then
            Dim NoPath(0) As Point
            Return NoPath
        Else
            If Path_Points(UBound(Path_Points) - 1).Equals(endP) Then
                Return Path_Points
            End If
        End If

        '--Store new Start point--
        'Dim found As Boolean

        '---is it already stored as a point?----
        'For j = 0 To UBound(Path_Points)
        '    For i = 0 To UBound(Start_points)
        '        '---escape clause---
        '        If Start_points(i).X = 0 And Start_points(i).Y = 0 Then
        '            Exit For
        '        End If

        '        If Order_of_Match(j, 1).x = Start_points(i).X And _
        '            Order_of_Match(j, 1).y = Start_points(i).Y Then
        '            found = True
        '        End If

        '    Next i

        'If found = False Then
        '-Save-
        ' Start_points(UBound(Start_points)) = Order_of_Match(i, 1)
        '-Redim Array-
        ' ReDim Preserve Start_points(UBound(Start_points) + 1)
        '    Exit For
        'Else
        '    found = False
        'End If
        'Next j
        iTrace += 1
    Loop

    '--Return successful array here--
    Return Path_Points
End Function

'===DOESN'T WORK===
'Sub Pixel_compare_Probability(ByRef arrOne() As Point, ByRef arrTwo() As Point, ByVal
»       pic1 As Bitmap, ByVal pic2 As Bitmap)
'    '---Init Vars--
'    Dim arrOne_META(UBound(arrOne), 3)
'    Dim arrTwo_META(UBound(arrTwo), 3)
'    Dim i, j As Int32
'    Dim dblDiff9, dblDiff25, dblDiff_Back As Double
'    Dim dblDiff9_Best, dblDiff25_Best, dblDiff_Back_Best As Double

'    Dim arrPixels(UBound(arrOne_META), 1) As Point

'    '---Get Meta Data---
'    arrOne_META = GET_Meta_Data(arrOne, pic1)
'    arrTwo_META = GET_Meta_Data(arrTwo, pic2)

'    '--Compare Arrays---
'    For i = 0 To UBound(arrOne_META)
```

```
    '        '--save pixel--
    '        arrPixels(i, 0) = arrOne_META(i, 0)

    '        For j = 0 To UBound(arrTwo_META)
    '            '--Compare Attribs to find closest match---
    '            dblDiff9 = Abs(arrTwo_META(j, 1) - arrOne_META(i, 1))
    '            dblDiff25 = Abs(arrTwo_META(j, 2) - arrOne_META(i, 2))
    '            dblDiff_Back = Abs(arrTwo_META(j, 3) - arrOne_META(i, 3))

    '            '---Save Best Pixel match---
    '            If j = 0 Then
    '                dblDiff9_Best = dblDiff9
    '                dblDiff25_Best = dblDiff25
    '                dblDiff_Back_Best = dblDiff_Back
    '                '--Save pixel match--
    '                arrPixels(i, 1) = arrTwo_META(j, 0)
    '            End If

    '            If (dblDiff9 <= dblDiff9_Best) And _
    '                (dblDiff25 <= dblDiff25_Best) And _
    '                (dblDiff_Back <= dblDiff_Back_Best) Then

    '                dblDiff9_Best = dblDiff9
    '                dblDiff25_Best = dblDiff25
    '                dblDiff_Back_Best = dblDiff_Back
    '                '--Save pixel match--
    '                arrPixels(i, 1) = arrTwo_META(j, 0)

    '            End If
    '        Next j
    '    Next i

    '    '--Calc Trans & Rot needed--
    '    Call Rotate_Translate(arrPixels)
    'End Sub

    'Private Function GET_Meta_Data(ByVal arrStart() As Point, ByVal myImage As Bitmap)
    '    '---Ave diff of closest 8,25 pixels & diff from background---
    '    '---&maybe number of other red pixels within 10 pixels---

    '    '---Init Vars--
    '    Dim i As Int32
    '    Dim dblDiff9, myI, SumDiff9, dblDiff25, sumDiff25 As Double
    '    Dim dblBack_1, dblBack_2 As Double
    '    Dim arr_META(UBound(arrStart), 3)

    '    '=======ARRAY ONE=======
    '    '--Get Ave Background I--
    '    dblBack_1 = GET_Ave_Intensity(myImage)

    '    For i = 0 To UBound(arrStart)
    '        '--Get meta data--
    '        arr_META(i, 0) = arrStart(i)

    '        '-Set Vars-
    '        myI = myImage.GetPixel(arrStart(i).X, arrStart(i).Y).GetBrightness

    '        '-Get ave diff of closest 9 and next closest 25 pixels-
    '        For iX = -2 To 2
    '            For iY = -2 To 2
    '                If (iX = -1 Or iX = 0 Or iX = 1) And (iY = -1 Or iY = 0 Or iY = 1) Th
»       en '9 closest
    '                    dblDiff9 = Abs(myI - myImage.GetPixel(arrStart(i).X + iX, arrStar
»       t(i).Y + iY).GetBrightness)
    '                    SumDiff9 = SumDiff9 + dblDiff9
    '                Else
    '                    dblDiff25 = Abs(myI - myImage.GetPixel(arrStart(i).X + iX, arrSta
»       rt(i).Y + iY).GetBrightness)
    '                    sumDiff25 = sumDiff25 + dblDiff25
    '                End If
    '            Next iY
    '        Next iX

    '        '--Get ave diff of closest 8 pixels--
    '        arr_META(i, 1) = SumDiff9 / 9
```

```
'        '--Get ave diff of next closest 25 pixels--
'        arr_META(i, 2) = sumDiff25 / 25
```

```
    '        '--Get ave diff of pixel intensity from ave background
    '        arr_META(i, 3) = myI - dblBack_1
    '    Next i


    '    Return arr_META

    'End Function

    Sub Rotate_Translate(ByRef arrMain(,) As Point)
        Debug.WriteLine("Rotate Translate start: " & Now())
        'Rotate up to 20 Translate up to max diff in X,Y
        '--Init Vars--
        Dim iAngle As Int16
        Dim i, j As Int32
        Dim iDifference_X, iDifference_Y, iDifference_Least As Int64
        Dim Best_T As Point, Best_A As Int16

        'Dim arrMax_PositiveDiff(CInt(UBound(arrMain) * 0.1))
        'Dim arrMax_NegativeDiff(CInt(UBound(arrOne) * 0.1), 1)
        Dim DiffX, DiffY, SumDiffX, SumDiffY, Ave_Moved_DiffX, Ave_Moved_DiffY As Double

        '=====Calc Max Diff in X, Y for pixels=====
        For i = 0 To UBound(arrMain)
            DiffX = Abs(arrMain(i, 0).X - arrMain(i, 1).X)
            DiffY = Abs(arrMain(i, 0).Y - arrMain(i, 1).Y)
            SumDiffX = SumDiffX + DiffX
            SumDiffY = SumDiffY + DiffY
            '--save if best--
            'For j = 0 To UBound(arrMax_PositiveDiff)
            '    '--Assign best positives--
            '    If arrMax_PositiveDiff(j) = 0 Then
            '        arrMax_PositiveDiff(j) = DiffX + DiffY
            '    Else
            '        If (DiffX + DiffY) > arrMax_PositiveDiff(j) Then
            '            arrMax_PositiveDiff(j) = DiffX + DiffY
            '        End If

            '    End If
            'Next j
        Next i
        '-Get Ave-
        Ave_Moved_DiffX = SumDiffX / (UBound(arrMain) + 1)
        Ave_Moved_DiffY = SumDiffY / (UBound(arrMain) + 1)
        'For i = 0 To UBound(arrMax_PositiveDiff)
        '    SumDiff = SumDiff + arrMax_PositiveDiff(i)
        'Next i

        'Ave_Moved_Diff = SumDiff / (UBound(arrMax_PositiveDiff) + 1)
        Debug.Write(Ave_Moved_DiffX & " - " & Ave_Moved_DiffY)
        '--Split Array into Two--
        Dim arrOne(UBound(arrMain)) As Point
        Dim arrTwo(UBound(arrMain)) As Point

        For i = 0 To UBound(arrMain)
            arrOne(i) = arrMain(i, 0)
            arrTwo(i) = arrMain(i, 1)
        Next i

        '---Rotate & Translate--
        For iY = 0 To CInt(Ave_Moved_DiffY)
            For iX = 0 To CInt(Ave_Moved_DiffX)
                For iAngle = 0 To 359
                    '--Rotate by Angle--
                    arrTwo = Rotate_XY_Points(arrTwo, iAngle)
                    '--compare points--
                    For i = 0 To UBound(arrOne)
                        For j = 0 To UBound(arrTwo)
                            iDifference_X = iDifference_X + Abs(arrOne(i).X - arrTwo(j).X)
                            iDifference_Y = iDifference_Y + Abs(arrOne(i).Y - arrTwo(j).Y)
                        Next j
                    Next i

                    '--store best match--
                    If iAngle = 0 Then
                        iDifference_Least = iDifference_X + iDifference_Y
```

```
                    Else
                        If iDifference_X + iDifference_Y <= iDifference_Least Then
                            iDifference_Least = iDifference_X + iDifference_Y
                            Best_T.X = iX
                            Best_T.Y = iY
                            Best_A = iAngle
                        End If
                    End If
                    iDifference_X = 0
                    iDifference_Y = 0
                Next iAngle
            Next iX
        Next iY

        Debug.WriteLine("Rotate Translate End: " & Now())
        Debug.WriteLine(Best_T.X & "," & Best_T.Y & ":" & Best_A)
    End Sub

    Private Function Translate_XY_Points(ByRef myPoints() As Point, ByVal X As Int32,
»       ByVal Y As Int32) As Point()
        '-----Init Vars----
        Dim m As New Drawing.Drawing2D.Matrix()
        Dim i As Int32


        '---Translate Image---
        m.Translate(X, Y)
        m.TransformPoints(myPoints)
        Translate_XY_Points = myPoints

    End Function

    Private Function Rotate_XY_Points(ByVal BeforeP() As Point, ByRef Angle As Int16) As
»       Point()
        '------Init Vars------
        Rotate_XY_Points = BeforeP
        Dim i As Int32
        Dim m As New Drawing.Drawing2D.Matrix()
        '-------Rotate Points-------
        m.Rotate(Angle)

        m.TransformPoints(Rotate_XY_Points)

    End Function



    Public Sub New(ByVal pic As Bitmap, ByVal grp_Size As String, ByVal nJoin_UP As String
»       )
        '---Init Vars---
        myPic = New Bitmap(pic)
        nGrp_Size = grp_Size
        nJoin = IIf(nJoin_UP = "", 3, nJoin_UP)
    End Sub
End Class
```

ARID DeShadow.vb

```
Imports System.Math
Public Class DeShadow
    Dim Percentage_to_sample As Double
    Dim myIMG As Bitmap
    Dim ix, iy As Int32
    Dim dThreshold, d2Dark As Double
    Public Sub New(ByVal Percentage As Int32, ByVal img As Bitmap)
        Percentage_to_sample = 1 / Percentage
        Dim myI As Double
        myIMG = New Bitmap(img)
        myI = GET_Ave_Intensity(myIMG)
        Debug.Write(myI)
        dThreshold = myI
        d2Dark = 0.05
    End Sub

    Function deshadow() As Image
        '-init vars-
        Dim arrSample() As Double
        Dim arrDist() As Int32, arrDARKDist(,) As Object
        Dim No_of_Darks As Int64, iScale As Double

        '-sample random region-
        arrSample = Sample_regions()
        '-Get distribution generally-
        arrDist = Get_Distribution(arrSample)

        '-Invert-

        '-Get Noof DArks-
        No_of_Darks = Scan_Image_For_Noof_Shadows()

        '-Scaleup distribution-
        iScale = CInt(No_of_Darks / UBound(arrSample))
        For ix = 0 To UBound(arrDist)
            arrDist(ix) = arrDist(ix) * iScale
        Next ix

        '-Get distribution of dark pixels-
        arrDARKDist = Get_Dark_Distribution(No_of_Darks)

        '-remap and recolour dark pixels-
        myIMG = Remap_and_recolour(arrDist, arrDARKDist)


        Return myIMG
    End Function

    Private Function GET_Ave_Intensity(ByVal pic As Bitmap) As Double
        '---Init Vars---
        Dim dblBright, sum, ave As Double
        Dim newHue, newlum, newSat As Double
        Dim Colour_converted_Pixel As HLSRGB

        '---Scan for Candidates---
        For iy = 0 To pic.Height - 1
            For ix = 0 To pic.Width - 1
                dblBright = pic.GetPixel(ix, iy).GetBrightness
                sum = sum + dblBright
            Next ix
        Next iy
        ave = sum / ((pic.Height - 1) * (pic.Width - 1))

        Return ave

    End Function

    Private Function Invert_array(ByVal arrStart() As Int32) As Int32()
        Dim tmpI As Int32
        For ix = 0 To UBound(arrStart) / 2
            tmpI = arrStart(ix)
            arrStart(ix) = arrStart(UBound(arrStart) - ix)
```

```
            arrStart(UBound(arrStart) - ix) = tmpI
        Next
        Invert_array = arrStart
    End Function

    Private Function Sample_regions() As Double()
        '-init vars-
        Dim Pixels_to_get, i, RND_X, RND_Y As Int32
        Dim dbl_I As Double
        Pixels_to_get = ((myIMG.Width - 1) * (myIMG.Height - 1)) * Percentage_to_sample

        '-dim array-
        Dim arrSample(Pixels_to_get - 1) As Double

        '-get pixels-
        For i = 0 To Pixels_to_get - 1
            RND_X = CInt(myIMG.Width * Rnd()) - 1
            RND_Y = CInt(myIMG.Height * Rnd()) - 1

            If RND_X < 0 Then RND_X = 0
            If RND_Y < 0 Then RND_Y = 0

            dbl_I = myIMG.GetPixel(RND_X, RND_Y).GetBrightness
            If dbl_I < dThreshold Then
                If i > 0 Then i = i - 1
            Else
                arrSample(i) = dbl_I
            End If


        Next

        '-return & clean-
        Return arrSample

    End Function

    Private Function Get_Distribution(ByVal arrSample() As Double) As Int32()
        Dim arrFreq(100) As Int32
        Dim tmplum As Double
        '---Create Freq arrays----
        For ix = 0 To UBound(arrSample)
            tmplum = Math.Round(arrSample(ix) * 100)
            arrFreq(tmplum) = arrFreq(tmplum) + 1
        Next

        Get_Distribution = arrFreq
    End Function

    Private Function Scan_Image_For_Noof_Shadows() As Int64
        '---init vars---
        Dim myB As New Bitmap(myIMG)
        Dim iDark As Int64
        Dim tmp As Double

        '---scan image---
        For ix = 0 To myIMG.Width - 1
            For iy = 0 To myIMG.Height - 1
                tmp = myB.GetPixel(ix, iy).GetBrightness
                If tmp > d2Dark And tmp < dThreshold Then
                    iDark = iDark + 1

                End If
            Next iy
        Next ix

        '-clean & return -
        Scan_Image_For_Noof_Shadows = iDark
        myB = Nothing
    End Function

    Private Function Get_Dark_Distribution(ByVal No_of_Darks As Int32) As Object(,)
        '-Init Vars-
        Dim arrDark(No_of_Darks - 1, 2) As Double
```

```
        Dim idark As Int32, dblDark As Double
        '-get all dark_regions-
        For ix = 0 To myIMG.Width - 1
            For iy = 0 To myIMG.Height - 1
                dblDark = myIMG.GetPixel(ix, iy).GetBrightness()
                If dblDark > d2Dark And dblDark < dThreshold Then
                    arrDark(idark, 0) = dblDark
                    arrDark(idark, 1) = ix
                    arrDark(idark, 2) = iy
                    idark = idark + 1
                End If
            Next
        Next

        '-Calc Distribution-
        Dim arrFreq(100, 2) As Object
        Dim tmplum As Double
        '---Create Freq arrays----
        For ix = 0 To UBound(arrDark)
            tmplum = Math.Round(arrDark(ix, 0) * 100)
            arrFreq(tmplum, 0) = arrFreq(tmplum, 0) + 1
            '-A cheating way of storind coordinates-
            arrFreq(tmplum, 1) = arrFreq(tmplum, 1) & arrDark(ix, 1) & "|"
            arrFreq(tmplum, 2) = arrFreq(tmplum, 2) & arrDark(ix, 2) & "|"
        Next


        '-return-
        Get_Dark_Distribution = arrFreq

    End Function


    Private Function Remap_and_recolour(ByVal arrDist() As Int32, ByVal arrDARKDist As
»       Object) As Image
        '-init vars-
        Dim arrX(), arrY() As String
        Dim iDark_Freq, iLight_Freq, iLeft, LastiY, iStart, iEnd As Int32
        Dim FinalIMG As New Bitmap(myIMG)
        Dim MoreThan As Boolean
        Dim dark_left, dark_iy As Int32
        '-loop thru-
        For ix = 0 To UBound(arrDist)
            If arrDist(ix) <> 0 Then
                '---set iFREQ----
                iLight_Freq = arrDist(ix)
                If ix = 15 Then Debug.Write("")
                '-find corresponding normal dist-
                For iy = dark_iy To UBound(arrDARKDist)
                    If CInt(arrDARKDist(iy, 0)) <> 0 Then

                        iDark_Freq = arrDARKDist(iy, 0)
                        '-split coords found-
                        arrX = Split(arrDARKDist(iy, 1), "|")
                        arrY = Split(arrDARKDist(iy, 2), "|")


                        If iEnd > iDark_Freq Then iEnd = 0

                        '-remap some/all-
                        Call remap_IT(iLight_Freq, ix, arrX, arrY, FinalIMG, iStart, iEnd)

                        '-set session state of dark array-
                        dark_left = (iDark_Freq - iStart) - iLight_Freq
                        If dark_left = 0 Then
                            dark_iy = iy + 1
                            Exit For
                        ElseIf dark_left > 0 Then
                            dark_iy = iy
                            iStart = iLight_Freq
                            Exit For
                        ElseIf dark_left < 0 Then
                            'not sure yet
                            dark_iy = iy + 1
                            '--Flags so we know how many we got left to place--
                            iEnd = Abs(dark_left)
```

```
                        iLight_Freq = Abs(dark_left)
                        iStart = 0
                    End If

                End If
            Next iy

            iEnd = 0
        End If
    Next

    '-retun piccy-
    Remap_and_recolour = FinalIMG
End Function

Private Function remap_IT(ByVal iLight_Freq, ByVal ix, ByVal arrX(), ByVal arrY(),
»       ByVal FinalIMG, ByVal iStart, ByVal iEnd) As Image
    Dim myC As New HLSRGB()
    Dim i As Int32
    Dim Ending As Int32
    '-decide on end point-
    If iEnd = 0 Then
        Ending = UBound(arrX) - 1
    Else
        Ending = iEnd - 1
    End If
    For i = 0 + iStart To Ending
        '-build colour-
        myC.Hue = FinalIMG.GetPixel(arrX(i), arrY(i)).GetHue
        myC.Saturation = FinalIMG.GetPixel(arrX(i), arrY(i)).GetSaturation
        myC.Luminance = ix / 100
        '-apply-
        FinalIMG.SetPixel(arrX(i), arrY(i), myC.Color)
    Next


End Function



End Class
```

ARID **Form1.vb**

```
Imports System.Math
Imports System.IO
Imports System.Drawing.Drawing2D

Public Class FRMArid
    Inherits System.Windows.Forms.Form

    Dim arrOF_Points_PIC1(1) As Point
    Dim arrOF_Points_PIC2(1) As Point
    Dim arrOF_Points_Zoomed(1) As Point
    Dim iPoints_PIC1 As Int32
    Dim iPoints_PIC2 As Int32
    Dim Translate_by As Point
    Dim Rotate_By As Single
    Dim Origin As PointF
    Dim myArea As GraphicsPath
    Dim Grid_On As Boolean
    Dim Retina_rectangle As Rectangle ' USED in GET Blue cmd and ARID auto
    Dim tmp2Del As Point ' delete @ later date

    '--Zoom bits---
    Dim ZoomRatio1, ZoomRatio2, old_Zoomratio1, old_Zoomratio2 As Int32
    Dim OLD_Selected_Part1, OLD_Selected_Part2 As Point
    '---------------
    Dim main_img1_TL, main_img2_TL As Point

    Dim Track_Bar_Value As Int32 = 25 ' For use with FRM FLip

    '---global vars for cursor moving--
    Dim Pic1_Cursor_one_Region, Pic1_Cursor_two_Region, Pic2_Cursor_one_Region,
»       Pic2_Cursor_two_Region As Region

    Dim Pic1_Zoom_Cursor_Region, Pic2_Zoom_Cursor_Region As Region
    Dim Before_XY, Zoom_Before_XY As Point

    Dim Which_cursor As Single
    '---------------

    'Private Function Translate_XY_Points(ByRef myPoints() As Point) As Point()
    '    '-----Init Vars----
    '    Dim m As New Drawing.Drawing2D.Matrix()
    '    Dim i As Int32, X, Y As Single

    '    '---Define Offset between Images---
    '    X = arrOF_Points_PIC2(0).X - arrOF_Points_PIC1(0).X
    '    Y = arrOF_Points_PIC2(0).Y - arrOF_Points_PIC1(0).Y

    '    '----Save Translation Vars----
    '    Translate_by.X = X
    '    Translate_by.Y = Y

    '    '---Translate Image---
    '    m.Translate(X, Y)
    '    m.TransformPoints(myPoints)
    '    Translate_XY_Points = myPoints

    '    '----Save origin for rotation----
    '    Origin.X = arrOF_Points_PIC2(0).X
    '    Origin.Y = arrOF_Points_PIC2(0).Y

    'End Function

    'Private Function Translate_XY_Points(ByRef myPoints() As Point, ByVal myVector As Vec
»       tor) As Point()
    '    '-----Init Vars----
    '    Dim m As New Drawing.Drawing2D.Matrix()
    '    Dim i As Int32, X, Y As Single

    '    '---Define Offset between Images---
    '    X = myVector.myP.X
    '    Y = myVector.myP.Y

    '    '----Save Translation Vars----
    '    Translate_by.X = X
    '    Translate_by.Y = Y
```

```
    '    '---Translate Image---
    '    m.Translate(X, Y)
    '    m.TransformPoints(myPoints)
    '    Translate_XY_Points = myPoints

    '    '----Save origin for rotation----
    '    'Origin.X = arrOF_Points_PIC2(0).X
    '    'Origin.Y = arrOF_Points_PIC2(0).Y

    'End Function

    'Private Function Rotate_Points(ByRef myPoints() As Point) As Point()

    '    '=============Init Vars============
    '    Dim iPoint, iSum_X, iSum_Y, i As Int32, iAngle As Single
    '    Dim iSmallest_X, iSmallest_Y, iSmallest_XY As Int32, sngBest_Angle As Single


    '    '=========Rotate=============

    '    Dim arrOF_Points_Rot(UBound(myPoints)) As Point
    '    Dim arrOF_Points_Normal(UBound(myPoints)) As Point
    '    For iAngle = 0 To 360
    '        '=======Reset Vars=======
    '        iSum_X = 0
    '        iSum_Y = 0

    '        '====Create Fresh Array===
    '        For i = 0 To UBound(myPoints)
    '            arrOF_Points_Normal(i) = myPoints(i)
    '        Next
    '        '====Rotate Points====
    '        arrOF_Points_Rot = Rotate_XY_Points(arrOF_Points_Normal, iAngle)

    '        '================Get Diffs==================
    '        For iPoint = 0 To UBound(myPoints)
    '            '---X Differences---
    '            iSum_X = iSum_X + Abs(Abs(arrOF_Points_PIC2(iPoint).X) - Abs(arrOF_Points
» _Rot(iPoint).X))
    '            '---Y Differences---
    '            iSum_Y += iSum_Y + Abs(Abs(arrOF_Points_PIC2(iPoint).Y) - Abs(arrOF_Point
» s_Rot(iPoint).Y))

    '        Next

    '        '==============Store Diffs if Smaller==================
    '        If iSum_X + iSum_Y <= iSmallest_XY Or iAngle = 0 Then

    '            iSmallest_X = iSum_X
    '            iSmallest_Y = iSum_Y
    '            iSmallest_XY = iSmallest_X + iSmallest_Y

    '            sngBest_Angle = iAngle

    '        End If
    '    Next iAngle

    '    Me.Text = "Best Fit Angle = " & sngBest_Angle
    '    '----Save optimum Angle----
    '    Rotate_By = sngBest_Angle
    '    Rotate_Points = Rotate_XY_Points(myPoints, iAngle)

    'End Function
    Function Rotate_XY_Points(ByVal myPs() As Point, ByRef Angle As Single) As Point()
        '------Init Vars------
        Rotate_XY_Points = myPs
        Dim i As Int32
        Dim m As New Drawing.Drawing2D.Matrix()
        '-------Rotate Points-------
        m.RotateAt(Angle, Origin)
        m.TransformPoints(Rotate_XY_Points)


    End Function

    Function Rotate_XY_Points(ByVal myPs() As Point, ByRef Angle As Single, ByVal
```

```
»       NEWOrigin As PointF) As Point()
        '------Init Vars------
        Rotate_XY_Points = myPs
        Dim i As Int32
        Dim m As New Drawing.Drawing2D.Matrix()
        '-------Rotate Points-------
        m.RotateAt(Angle, NEWOrigin)
        m.TransformPoints(Rotate_XY_Points)



    End Function
    'Private Sub Do_Points()
    '    '---Init Vars---
    '    Dim myPoints() As Point
    '    myPoints = arrOF_Points_PIC1
    '    '---Translate---
    '    myPoints = Translate_XY_Points(myPoints)
    '    '-----Rotate----
    '    myPoints = Rotate_Points(myPoints)


    'End Sub



    'Sub start_up()
    '    '--------------------Init Vars----------------------
    '    Dim myImg As Image
    '    myImg = PIC1.Image

    '    '-------------Define Corners of Image---------------
    '    Dim wid = myImg.Width
    '    Dim Hgt = myImg.Height
    '    Dim myCorners As Point() = { _
    '       New Point(0, 0), _
    '       New Point(wid, 0), _
    '       New Point(0, Hgt), _
    '       New Point(wid, Hgt)}

    '    picHidden.Image = myImg

    '    '----Work Out Translation & Rotation from Points----
    '    Call Do_Points()
    '    '--------Use Those Vals to Manipulate image---------
    '    myCorners = Do_Image(myCorners)

    '    '----save corners as workspace area----
    '    myArea = New GraphicsPath()
    '    myArea.StartFigure()
    '    myArea.AddLine(myCorners(0), myCorners(1))
    '    myArea.AddLine(myCorners(1), myCorners(3))
    '    myArea.AddLine(myCorners(3), myCorners(2))
    '    ' myArea.AddLine(myCorners(2), myCorners(0))

    '    myArea.CloseFigure()



    '    '-------Display Results-----
    '    Dim bm_out As New Bitmap(CInt(PIC2.Image.Width), CInt(PIC2.Image.Height))

    '    Dim gr_out As Graphics = Graphics.FromImage(bm_out)
    '    ReDim Preserve myCorners(2)
    '    gr_out.DrawImage(myImg, myCorners)

    '    '------Redraw Hidden original-----
    '    Dim bm_Original As New Bitmap(CInt(PIC2.Image.Width), CInt(PIC2.Image.Height))
    '    Dim gr_Original As Graphics = Graphics.FromImage(bm_Original)
    '    gr_Original.DrawImage(pic1before.Image, myCorners)

    '    '----Update Pics----
    '    'picHidden.Image = bm_out
    '    PIC1.Image = bm_out
    '    picHidden.Image = bm_Original
```

```
    '    'PicDisplay.Image = PIC2.Image



    'End Sub
    'Private Function Do_Image(ByVal myCorners() As Point) As Point()

    '    '--Translate--
    '    Dim i As Int32
    '    For i = 0 To 3
    '        myCorners(i).X = myCorners(i).X + Translate_by.X
    '        myCorners(i).Y = myCorners(i).Y + Translate_by.Y
    '    Next i


    '    '---Rotate---
    '    myCorners = Rotate_XY_Points(myCorners, Rotate_By)
    '    Do_Image = myCorners

    'End Function


    Private Function InVert_IMG(ByVal imgBefore As Image) As Image
        '======Init Vars=====

        Dim bm As New Bitmap(imgBefore)
        Dim X As Integer
        Dim Y As Integer
        Dim r As Integer
        Dim g As Integer
        Dim b As Integer

        '=====Invert Colors====
        For X = 0 To bm.Width - 1
            For Y = 0 To bm.Height - 1
                r = 255 - bm.GetPixel(X, Y).R
                g = 255 - bm.GetPixel(X, Y).G
                b = 255 - bm.GetPixel(X, Y).B
                bm.SetPixel(X, Y, Color.FromArgb(128, r, g, b))
            Next Y
        Next X
        '====Return Image=====
        InVert_IMG = bm


    End Function

    Private Sub Blend_Pics(ByVal Orig_IMG As Image)
        '-------Init Vars---------
        Dim bm_Pic2 As New Bitmap(Orig_IMG)
        Dim bm_Final As New Bitmap(picHidden.Image)
        Dim X As Single = Translate_by.X
        Dim Y As Single = Translate_by.Y

        Dim gr As Graphics = Graphics.FromImage(bm_Pic2)

        gr.DrawImage(bm_Final, 0, 0)


        '------Display the result--------------
        PicDisplay.Image = bm_Pic2
        picHidden.Image = bm_Pic2
    End Sub

    Private Sub cmdLoad_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
»       Handles cmdLoad_A.Click, cmdLoad_B.Click
        '---Init Diaglog Box----
        OpenBox.Filter = "Image Files (*.jpg,*.tif,*.bmp)|*.jpg;*.tif;*.bmp|All files (*.*
»           )|*.*"
        'OpenBox.InitialDirectory = "j:\guest\arid\data\"
        OpenBox.ShowDialog()
        '---Load Filename is image Holder----
        If OpenBox.FileName <> "" Then

            '----Get file Title----
```

```
            Dim tmpName(), tmpfile(), file_name As String
            tmpName = Split(OpenBox.FileName, "\")
            tmpfile = Split(tmpName(UBound(tmpName)), ".")
            file_name = tmpfile(0)
            '--------------------
            '----Draw on cursor at default co-ords---
            Dim wid, hgt As Int32

            If sender Is cmdLoad_A Then
                PIC1.Image = Bitmap.FromFile(OpenBox.FileName)
                Call FRMArid_Load(sender, e)
                '---info label---
                lblInfo_A.Text = file_name '& " (Points Selected = 0)"

            Else
                PIC2.Image = Bitmap.FromFile(OpenBox.FileName)
                Call FRMArid_Load(sender, e)
                '---info label---
                lblInfo_B.Text = file_name '& " (Points Selected = 0)"
            End If

        End If
    End Sub
    'Private Sub cmdRegister_Click(ByVal sender As System.Object, ByVal e As System.EventA
»       rgs)
    '    Call start_up()
    '    'Call Shrink_Canvas()
    'End Sub

    'Sub Shrink_Canvas()
    '    '----REsize Canvas to fit image------------


    '    'tmpgfx.DrawPath(Pens.DarkBlue, myArea)


    '    '----Work out width and height of useable area----
    '    Dim LowPoint_X, LowPoint_Y, HighPoint_X, HighPoint_Y As Int32
    '    Dim i, wid, hgt As Int32
    '    LowPoint_X = HighPoint_X = myArea.PathPoints.GetValue(0).X
    '    LowPoint_Y = HighPoint_Y = myArea.PathPoints.GetValue(0).Y

    '    For i = 0 To 3
    '        '---Find Lowest/Highest X---
    '        If myArea.PathPoints.GetValue(i).X < LowPoint_X Then
    '            LowPoint_X = myArea.PathPoints.GetValue(i).X
    '        ElseIf myArea.PathPoints.GetValue(i).X > HighPoint_X Then
    '            HighPoint_X = myArea.PathPoints.GetValue(i).X
    '        End If

    '        '---Find Lowest/Highest X---
    '        If myArea.PathPoints.GetValue(i).Y < LowPoint_Y Then
    '            LowPoint_Y = myArea.PathPoints.GetValue(i).Y
    '        ElseIf myArea.PathPoints.GetValue(i).Y > HighPoint_Y Then
    '            HighPoint_Y = myArea.PathPoints.GetValue(i).Y
    '        End If

    '    Next
    '    wid = Abs(HighPoint_X - LowPoint_X)
    '    hgt = Abs(HighPoint_Y - LowPoint_Y)

    '    Dim bm_Source As Graphics = Graphics.FromImage(PIC1.Image)
    '    Dim bm_Destination As New Bitmap(wid, hgt)
    '    Dim gr_Destination As Graphics = Graphics.FromImage(bm_Destination)
    '    gr_Destination.DrawImage(PIC1.Image, 0, 0, bm_Destination.Width, bm_Destination.H
»       eight)
    '    PIC1.Image = bm_Destination


    'End Sub

    'Private Sub cmdPoints_Click(ByVal sender As System.Object, ByVal e As System.EventArg
»       s)
    '    Call Do_Points()
    'End Sub
```

```
    Private Sub FRMArid_Resize(ByVal sender As Object, ByVal e As System.EventArgs)
»       Handles MyBase.Resize
        '---Init Vars----
        Dim HW_Ratio As Double

        '---Size Ratio---
        HW_Ratio = PIC1.Image.Width / PIC1.Image.Height

        'Picture One
        PIC1.Left = 5
        If cmdLoad_A.Visible = True Then
            PIC1.Top = cmdLoad_A.Top + cmdLoad_A.Height + 5
        Else
            PIC1.Top = 5
        End If

        PIC1.Width = (Me.Width / 2) - 5
        PIC1.Height = PIC1.Width / HW_Ratio

        'Picture Two
        If cmdLoad_B.Visible = True Then
            PIC2.Top = cmdLoad_B.Top + cmdLoad_B.Height + 5
        Else
            PIC2.Top = 5
        End If

        PIC2.Width = (Me.Width / 2) - 5
        PIC2.Height = PIC2.Width / HW_Ratio
        PIC2.Left = (Me.Width - 5) - PIC2.Width

        'Label 1 & 2
        lblInfo_A.Top = PIC1.Top + PIC1.Height + 1
        lblInfo_A.Left = 5
        lblInfo_B.Top = PIC2.Top + PIC2.Height + 1
        lblInfo_B.Left = PIC2.Left

        'Label Zoom 1 & 2
        lblZoom1.Top = cmdEnhance1.Top
        lblZoom1.Left = PIC1.Left + PIC1.Width - lblZoom1.Width
        lblZoom2.Top = cmdEnhance1.Top
        lblZoom2.Left = PIC2.Left

        'Picture Display
        PicDisplay.Left = 5
        PicDisplay.Top = lblInfo_A.Top + lblInfo_A.Height + 5
        PicDisplay.Height = (Me.Height - 35) - (PIC1.Height + lblInfo_A.Height + PIC1.Top)
        PicDisplay.Width = Me.Width * 0.33

    End Sub

    Private Sub FRMArid_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
»       MyBase.Load
        Call FRMArid_Resize(Me, e.Empty)

        Dim wid, hgt As Int32
        wid = PIC1.Width / 2
        hgt = PIC1.Height / 2


        If sender Is cmdLoad_A Then
            '--load images--
            pic1before.Image = PIC1.Image
            Pic1_Original.Image = PIC1.Image
            pic1zoom.Image = PIC1.Image
            pic1_cursor1.Image = PIC1.Image
            pic1_cursor2.Image = PIC1.Image

            OLD_Selected_Part1 = Nothing
            ZoomRatio1 = 0

            Call Draw_Cursor(wid + 50, hgt + 50, 0, PIC1)
            Call Draw_Cursor(wid, hgt, 1, PIC1)
            Call fix_zoom_image(PIC1)

        ElseIf sender Is cmdLoad_B Then
            '--load images--
```

```
            Pic2Before.Image = PIC2.Image
            Pic2_Original.Image = PIC2.Image
            pic2zoom.Image = PIC2.Image
            pic2_Cursor1.Image = PIC1.Image
            pic2_Cursor2.Image = PIC1.Image

            OLD_Selected_Part2 = Nothing
            ZoomRatio2 = 0

            Call Draw_Cursor(wid + 50, hgt + 50, 0, PIC2)
            Call Draw_Cursor(wid, hgt, 1, PIC2)
            Call fix_zoom_image(PIC2)
        Else
            Call FRMArid_Load(cmdLoad_A, e)
            Call FRMArid_Load(cmdLoad_B, e)
        End If


    End Sub


    Private Sub cmdPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        '  Dim bm As New Bitmap(picHidden.Image)
        '   bm.Save("ARID.jpg", Imaging.ImageFormat.Jpeg)

        '
        Dim pd As New Printing.PrintDocument()

        AddHandler pd.PrintPage, AddressOf Me.pd_PrintPage
        pd.Print()

    End Sub

    Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As Printing.
»       PrintPageEventArgs)
        ' Draw a picture.
        ev.Graphics.DrawImage(PicDisplay.Image, 0, 0, PicDisplay.Height, PicDisplay.Width)


        ' Indicate that this is the last page to print.
        ev.HasMorePages = False
    End Sub



    Sub Equalise_HSL(ByVal do_Brightness As Boolean)
        '========= Init Vars ===========
        Dim imgOne, imgTwo As Bitmap, Pixel_Count, iCol, iRow, iData As Long
        Dim imgA_Sat, imgB_Sat, imgA_Hue, imgB_Hue, imgA_Lum, imgB_Lum, iDiff_S, iDiff_L
»           As Double
        Dim A_Total_L, A_Total_S, B_Total_L, B_Total_S As Double
        Dim iBlank_Pixels As Int32
        Dim wid, hgt As Int32
        imgOne = PIC1.Image
        imgTwo = PIC2.Image

        '---create blank canvas for painting----
        Dim bm_out As New Bitmap(CInt(imgOne.Width), CInt(imgOne.Height))
        Dim bm_out2 As New Bitmap(CInt(imgOne.Width), CInt(imgOne.Height))
        PicDisplay.Image = bm_out
        picHidden.Image = bm_out2
        Dim gfxA As Graphics = Graphics.FromImage(PicDisplay.Image)
        Dim gfxB As Graphics = Graphics.FromImage(picHidden.Image)

        'Pixel_Count = ((imgOne.Height + 1) * (imgOne.Width + 1)) - 1
        '----Calculate number of Pixel used in registration----
        Dim TR_X, TR_Y As Int32

        If myArea.PathData.Points(1).X < 0 Then
            TR_X = 0
        ElseIf myArea.PathData.Points(1).X > imgOne.Width Then
            TR_X = imgOne.Width
        Else
```

```
        TR_X = myArea.PathData.Points(1).X
    End If

    If myArea.PathData.Points(1).Y < 0 Then
        TR_Y = 0
    ElseIf myArea.PathData.Points(1).Y > imgOne.Height Then
        TR_X = imgOne.Height
    Else
        TR_Y = myArea.PathData.Points(1).Y
    End If

    wid = TR_X - myArea.PathData.Points(0).X
    hgt = TR_Y - myArea.PathData.Points(3).Y

    Pixel_Count = Abs(wid * hgt) - 1
    Dim Pixel_Array(Pixel_Count, 1) As HLSRGB

    '---Show wait Form----
    ' Dim myWait As New timerform()
    ' myWait.ProgressBar.Maximum = imgOne.Height.ToString * imgOne.Width.ToString * 2
    ' myWait.Show()
    '-----------------------

    '----GET HSL of all pixels in Region-----
    For iRow = 0 To imgOne.Height.ToString - 1
        'myWait.ProgressBar.Value += 1
        For iCol = 0 To imgOne.Width.ToString - 1
            ' myWait.ProgressBar.Value += 1
            '------------Get HSL---------------
            imgA_Hue = imgOne.GetPixel(iCol, iRow).GetHue

            If imgA_Hue <> 0 Then
                If do_Brightness = True Then
                    imgA_Lum = imgOne.GetPixel(iCol, iRow).GetBrightness
                    imgB_Lum = imgTwo.GetPixel(iCol, iRow).GetBrightness
                Else
                    imgA_Sat = imgOne.GetPixel(iCol, iRow).GetSaturation
                    imgB_Sat = imgTwo.GetPixel(iCol, iRow).GetSaturation

                End If

                '---Sum Total---

                A_Total_L = A_Total_L + imgA_Lum
                B_Total_L = B_Total_L + imgB_Lum

                A_Total_S = A_Total_S + imgA_Sat
                B_Total_S = B_Total_S + imgB_Sat

            End If

        Next
    Next

    '--------Work out mean difference----------
    Dim L_Ratio, S_Ratio As Double
    Dim AL, ASa As Boolean
    If do_Brightness = True Then
        If A_Total_L > B_Total_L Then
            ' L_Ratio = (A_Total_L - B_Total_L) / Pixel_Count
            L_Ratio = (A_Total_L / B_Total_L) - 1
            AL = True
        Else
            'L_Ratio = (B_Total_L - A_Total_L) / Pixel_Count
            L_Ratio = (B_Total_L / A_Total_L) - 1
            AL = False
        End If
    Else
        If A_Total_S > B_Total_S Then
            ' S_Ratio = (A_Total_S - B_Total_S) / Pixel_Count
            S_Ratio = (A_Total_S / B_Total_S) - 1
            ASa = True
        Else
            ' S_Ratio = (B_Total_S - A_Total_S) / Pixel_Count
            S_Ratio = (B_Total_S / A_Total_S) - 1
            ASa = False
```

```
            End If
        End If



        lblinfo.Text = ("Lum Ratio: " & L_Ratio & " Sat Ratio: " & S_Ratio & " Add Lum to
»           Pic 2 = " & AL & " Add Sat to Pic 2 = " & ASa)

        '----------Update Images--------------
        For iRow = 0 To imgOne.Height.ToString - 1
            ' myWait.ProgressBar.Value += 1
            For iCol = 0 To imgOne.Width.ToString - 1
                'myWait.ProgressBar.Value += 1

                '------------Get HSL---------------
                imgA_Hue = imgOne.GetPixel(iCol, iRow).GetHue

                If imgA_Hue <> 0 Then

                    imgB_Hue = imgTwo.GetPixel(iCol, iRow).GetHue
                    imgA_Sat = imgOne.GetPixel(iCol, iRow).GetSaturation
                    imgB_Sat = imgTwo.GetPixel(iCol, iRow).GetSaturation
                    imgA_Lum = imgOne.GetPixel(iCol, iRow).GetBrightness
                    imgB_Lum = imgTwo.GetPixel(iCol, iRow).GetBrightness


                    '-----Equalise Pics either Lum or Sat------
                    If do_Brightness = False Then
                        If ASa = True Then
                            '  imgA_Sat = imgA_Sat * (1 - S_Ratio)
                            imgB_Sat = imgB_Sat * (1 + S_Ratio)
                        Else
                            imgA_Sat = imgA_Sat * (1 + S_Ratio)
                            '  imgB_Sat = imgB_Sat * (1 - S_Ratio)
                        End If

                    Else
                        If AL = True Then
                            '   imgA_Lum = imgA_Lum * (1 - L_Ratio)
                            imgB_Lum = imgB_Lum * (1 + L_Ratio)
                        Else
                            imgA_Lum = imgA_Lum * (1 + L_Ratio)
                            '   imgB_Lum = imgB_Lum * (1 - L_Ratio)
                        End If

                    End If


                    '---Error Check Vals----
                    If imgA_Lum > 1 Then
                        imgA_Lum = 1
                    ElseIf imgA_Lum < 0 Then
                        imgA_Lum = 0
                    End If

                    If imgA_Sat > 1 Then
                        imgA_Sat = 1
                    ElseIf imgA_Sat < 0 Then
                        imgA_Sat = 0
                    End If

                    If imgB_Lum > 1 Then
                        imgB_Lum = 1
                    ElseIf imgB_Lum < 0 Then
                        imgB_Lum = 0
                    End If

                    If imgB_Sat > 1 Then
                        imgB_Sat = 1
                    ElseIf imgB_Sat < 0 Then
                        imgB_Sat = 0
                    End If

                    '----Create new pixels----
                    Pixel_Array(iRow + iCol, 0) = New HLSRGB(imgA_Hue, imgA_Lum, imgA_Sat)
```

```
Pixel_Array(iRow + iCol, 1) = New HLSRGB(imgB_Hue, imgB_Lum, imgB_Sat)
```

```
                    '----Draw Pixels Back----
                    Dim colorPenA As Pen = New Pen(Pixel_Array(iRow + iCol, 0).Color, 1)
                    Dim colorPenB As Pen = New Pen(Pixel_Array(iRow + iCol, 1).Color, 1)

                    gfxA.DrawLine(colorPenA, iCol, iRow, iCol + 1, iRow + 1)
                    gfxB.DrawLine(colorPenB, iCol, iRow, iCol + 1, iRow + 1)

                    colorPenA.Dispose()
                    colorPenB.Dispose()
                End If
            Next
        Next

        '---hide wait form---
        '  myWait.Hide()
        '  myWait = Nothing
        '--------------------

        '-----update original images---------
        PIC1.Image = Nothing
        PIC2.Image = Nothing
        PIC1.Image = PicDisplay.Image
        PIC2.Image = picHidden.Image

        PicDisplay.Image = Nothing
        picHidden.Image = Nothing

    End Sub

    Private Sub cmdBright_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       )
        Call Equalise_HSL(True)
    End Sub

    'Private Sub CMDDiff_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    '    '----Invert & Merge Pics-----
    '    If sender Is CMDDiff Then
    '        picHidden.Image = InVert_IMG(picHidden.Image)
    '        Call Blend_Pics(Pic2Before.Image)
    '    Else
    '        picHidden.Image = InVert_IMG(PIC2.Image) 'FIX this to work without crosshairs
    '        Call Blend_Pics(pic1before.Image)
    '    End If


    '    '---------Pics Display----------
    '    '---Pics---
    '    PIC1.Top = 5
    '    PIC1.Left = 5
    '    PIC2.Top = 5
    '    PIC2.Left = Me.Width - (PIC2.Width + 5)


    '    '-cmds-
    '    Me.cmdLoad_A.Visible = False
    '    Me.cmdLoad_B.Visible = False
    '    Me.cmdARID.Visible = False
    '    '-lbls-
    '    Dim strTMP() As String
    '    strTMP = Split(lblInfo_A.Text, vbCrLf)
    '    lblInfo_A.Text = strTMP(0)
    '    lblInfo_A.Height = lblInfo_A.Height / 2
    '    lblInfo_A.Top = PIC1.Top + PIC1.Height

    '    strTMP = Split(lblInfo_B.Text, vbCrLf)
    '    lblInfo_B.Text = strTMP(0)
    '    lblInfo_B.Height = lblInfo_B.Height / 2
    '    lblInfo_B.Top = PIC2.Top + PIC2.Height

    '    lblinfo.Visible = False

    '    '--Arid Pic--
    '    ' Dim Width_Ratio, Height_Ratio As Double
```

```
    '    'Width_Ratio = PicDisplay.Image.Width / PicDisplay.Width
    '    ' Height_Ratio = PicDisplay.Image.Height / PicDisplay.Height

    '    '---also in right click of sender sub--
    '    PicDisplay.Top = lblInfo_A.Top + lblInfo_A.Height
    '    PicDisplay.Left = PIC1.Left
    '    PicDisplay.Width = PicDisplay.Image.Width
    '    PicDisplay.Height = (Me.Height - 35) - (PIC1.Height + lblInfo_A.Height + PIC1.Top
»       )
    '    PicDisplay.Width = Me.Width * 0.33
    '    PicDisplay.Visible = True


    '    '--------------------------------
    '    PicDisplay.BringToFront()
    '    cmdPrint.BringToFront()
    '    cmdClear.BringToFront()
    '    cmdSave.BringToFront()
    '    GrpEnhance.BringToFront()
    '    GrpEnhance.Visible = True
    '    cmdGrid.BringToFront()
    '    cmdReset2.Visible = False
    'End Sub

    Private Sub sender_MouseDown(ByVal sender As System.Object, ByVal e As System.Windows.
»       Forms.MouseEventArgs) Handles PIC2.MouseDown, PIC1.MouseDown
        tmp2Del.X = e.X 'DELete @ later date
        tmp2Del.Y = e.Y 'DELete @ later date

        '----------Init Vars----------
        Dim myImg As Bitmap
        Dim iX, iY As Single
        Dim arrTMP() As String
        Dim Cursor_Region1, Cursor_Region2 As Region
        '----Decide on Img Clicked-----
        If sender Is PIC1 Then
            myImg = New Bitmap(PIC1.Image)
        Else
            myImg = New Bitmap(PIC2.Image)
        End If

        '----Calc Coords----
        Dim Width_Ratio, Height_Ratio As Double
        Dim X, Y As Int32
        Width_Ratio = sender.Image.Width / sender.Width
        Height_Ratio = sender.Image.Height / sender.Height

        X = CInt(e.X * Width_Ratio)
        Y = CInt(e.Y * Height_Ratio)
        If X > sender.image.width - 1 Then X = sender.image.width - 1
        If Y > sender.image.height - 1 Then Y = sender.image.height - 1

        '---decide if left or right mouse clicked---
        If e.Button = MouseButtons.Left Then
            '---  Zoom(Out)---
            If OptZoom.Checked = True Then

                If sender Is PIC1 Then

                    If ZoomRatio1 - 1 < 0 Then
                        ZoomRatio1 = 0
                    Else
                        ZoomRatio1 = ZoomRatio1 - 1
                        lblZoom1.Text = ZoomRatio1 * 100 & "%"
                    End If
                Else
                    If ZoomRatio2 - 1 < 0 Then
                        ZoomRatio2 = 0
                    Else
                        ZoomRatio2 = ZoomRatio2 - 1
                        lblZoom2.Text = ZoomRatio2 * 100 & "%"
                    End If
                End If


                Call Img_Zoom(sender, X, Y, 0, 0)
```

```
            '---zoom cursor pic---
            If sender Is PIC1 Then
                Pic1_Zoom_Cursor.Image = sender.image
            Else
                Pic2_Zoom_Cursor.Image = sender.image
            End If
        ElseIf optMove.Checked = True Then
            '---save coords---
            Before_XY.X = X
            Before_XY.Y = Y
            Me.Cursor = Cursors.Hand
        Else '===========Cursor Moving===============

            If lblZoom1.Text = "0%" Then
                If sender Is PIC1 Then
                    Cursor_Region1 = Pic1_Cursor_one_Region
                    Cursor_Region2 = Pic1_Cursor_two_Region

                Else
                    Cursor_Region1 = Pic2_Cursor_one_Region
                    Cursor_Region2 = Pic2_Cursor_two_Region
                End If

                If Cursor_Region1.IsVisible(X, Y) Then
                    '-change cursor-
                    Me.Cursor = Cursors.Hand

                    '---save coords---
                    Before_XY.X = X
                    Before_XY.Y = Y
                    Which_cursor = 0
                ElseIf Cursor_Region2.IsVisible(X, Y) Then
                    '-change cursor-
                    Me.Cursor = Cursors.Hand
                    '---save coords---
                    Before_XY.X = X
                    Before_XY.Y = Y
                    Which_cursor = 1
                Else
                    Before_XY = Nothing

                End If
            Else '===========ZOOMED Cursor Moving================
                '---save zoomed image cursor coords---
                Dim Zoom_Cursor_Region As Region, ZoomRatio As Int32
                If sender Is PIC1 Then
                    Zoom_Cursor_Region = Pic1_Zoom_Cursor_Region
                    ZoomRatio = ZoomRatio1
                Else
                    Zoom_Cursor_Region = Pic2_Zoom_Cursor_Region
                    ZoomRatio = ZoomRatio2
                End If

                If Zoom_Cursor_Region.IsVisible(X, Y) Then
                    '-change cursor-
                    Me.Cursor = Cursors.Hand

                    '---save coords---
                    Zoom_Before_XY.X = X * ZoomRatio
                    Zoom_Before_XY.Y = Y * ZoomRatio
                Else
                    Before_XY = Nothing
                End If
                '================================================
            End If


        End If
    Else
        '---Zoom in----
        If OptZoom.Checked = True Then
            If sender Is PIC1 Then
                If ZoomRatio1 + 1 > 15 Then
                    ZoomRatio1 = 15
                Else
                    ZoomRatio1 = ZoomRatio1 + 1
                End If
```

```
                lblZoom1.Text = ZoomRatio1 * 100 & "%"
            Else
                If ZoomRatio2 + 1 > 15 Then
                    ZoomRatio2 = 15
                Else
                    ZoomRatio2 = ZoomRatio2 + 1
                End If
                lblZoom2.Text = ZoomRatio2 * 100 & "%"
            End If


            Call Img_Zoom(sender, X, Y, 0, 0)

            '---zoom cursor pic---
            If sender Is PIC1 Then
                Pic1_Zoom_Cursor.Image = sender.image
            Else
                Pic2_Zoom_Cursor.Image = sender.image
            End If
        Else
            '---Expand Picture----
            If sender.top <> 0 Then
                sender.left = 0
                sender.top = 0
                'sender.sizemode = PictureBoxSizeMode.StretchImage
                sender.width = Me.Width - (Me.GrpEnhance.Width + 20)
                sender.height = Me.Height
                sender.bringtofront()
                If sender Is PIC1 Then
                    PIC2.Visible = False
                End If
                '---Zoom Out----
            Else

                Call FRMArid_Resize(sender, e)

                If sender Is PIC1 Then
                    PIC2.Visible = True
                End If


            End If
        End If
    End If
    '--------Clean Up-----------

    myImg = Nothing
End Sub


'Private Sub sender_MouseMove(ByVal sender As System.Object, ByVal e As System.Windows
»       .Forms.MouseEventArgs) Handles PIC2.MouseMove, PIC1.MouseMove
'    'Dim imgA_Hue, imgA_Sat, imgA_Lum As Double
'    'Dim Width_Ratio, Height_Ratio As Double
'    'Dim X, Y As Int32
'    'Width_Ratio = sender.Image.Width / sender.Width
'    'Height_Ratio = sender.Image.Height / sender.Height

'    'X = CInt(e.X * Width_Ratio)
'    'Y = CInt(e.Y * Height_Ratio)
'    'If X > sender.image.width - 1 Then X = sender.image.width - 1
'    'If Y > sender.image.height - 1 Then Y = sender.image.height - 1
'    'If X < 0 Then X = 0
'    'If Y < 0 Then Y = 0

'    'imgA_Hue = sender.image.GetPixel(X, Y).GetHue
'    'imgA_Sat = sender.image.GetPixel(X, Y).GetSaturation
'    'imgA_Lum = sender.image.GetPixel(X, Y).GetBrightness

'    'Me.Text = "ARID v15 [" & e.X & ", " & e.Y & "]" & _
'    '" HSL [" & imgA_Hue & ", " & imgA_Sat & ", " & imgA_Lum & "]"



'End Sub
```

```
    Private Sub cmdSat_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Call Equalise_HSL(False)
    End Sub


    'Private Sub picHidden_DoubleClick(ByVal sender As System.Object, ByVal e As System.Ev
»       entArgs)
    '    Dim Point_Loc As Point
    '    Point_Loc.X = 0
    '    Point_Loc.Y = 0
    '    picHidden.Location = Point_Loc
    '    picHidden.Height = Me.Height
    '    picHidden.Width = Me.Width


    'End Sub


    'Private Sub cmdARID_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
»       Handles cmdARID.Click
    '    If UBound(arrOF_Points_PIC1) >= 1 And UBound(arrOF_Points_PIC2) >= 1 Then
    '        Call cmdRejig_Click(sender, e)
    '        Call cmdRegister_Click(sender, e)
    '        'Call cmdBright_Click(sender, e)
    '        Call CMDDiff_Click(CMDDiff, e)
    '        cmdColour.BringToFront()

    '    Else
    '        MsgBox("Not enought Points chosen, must be at least 2 per image")
    '    End If

    'End Sub




    Sub Export_Data()
        Dim imgOne, imgTwo As Bitmap, irow, iCol As Int32
        Dim icol_Count As Int32, iFLX As Int32 = 0
        Dim arrColours() As String
        Dim Pixel_Count As Long
        imgOne = PIC1.Image
        imgTwo = PIC2.Image
        Dim tmpArr() As String



        Pixel_Count = ((imgOne.Height) * (imgOne.Width)) - 1
        Dim arrEye_Data(Pixel_Count, 2, 1) As Double




        For irow = 0 To imgOne.Height.ToString - 1
            For iCol = 0 To imgOne.Width.ToString - 1

                '=============Use for Large Images=======================
                arrEye_Data(iFLX, 0, 0) = imgOne.GetPixel(iCol, irow).GetHue
                arrEye_Data(iFLX, 1, 0) = imgOne.GetPixel(iCol, irow).GetSaturation
                arrEye_Data(iFLX, 2, 0) = imgOne.GetPixel(iCol, irow).GetBrightness

                arrEye_Data(iFLX, 0, 1) = imgTwo.GetPixel(iCol, irow).GetHue
                arrEye_Data(iFLX, 1, 1) = imgTwo.GetPixel(iCol, irow).GetSaturation
                arrEye_Data(iFLX, 2, 1) = imgTwo.GetPixel(iCol, irow).GetBrightness
                iFLX = iFLX + 1


            Next iCol

        Next irow

        '======Init Vars=========
        Dim Filename, Filename2, strOut As String
        Dim imgWidth As Int32, Row_count As Int32
```

```
        '======Open File to save to==========
        Filename = "Exported1.txt"
        Filename2 = "Exported2.txt"

        Dim MyStream As New IO.FileStream(Filename, IO.FileMode.OpenOrCreate)
        Dim MyWriter As New IO.StreamWriter(MyStream)
        Dim MyStream2 As New IO.FileStream(Filename2, IO.FileMode.OpenOrCreate)
        Dim MyWriter2 As New IO.StreamWriter(MyStream2)

        Dim ipic As Int32
        imgWidth = imgOne.Width
        '=========Copy image Data 1 to text======

        For irow = 0 To UBound(arrEye_Data)
            '-----Header info-----
            If irow = imgWidth * Row_count Then
                MyWriter.WriteLine("Row " & Row_count)
                MyWriter.WriteLine("Pixels")
                MyWriter.WriteLine(imgWidth)
                Row_count = Row_count + 1
            End If

            MyWriter.WriteLine(arrEye_Data(irow, 0, 0) & "," & arrEye_Data(irow, 1, 0) & "
»               "," & arrEye_Data(irow, 2, 0))
        Next irow


        MyWriter.Close()
        MyStream.Close()

        '=========Copy image Data 2 to text======
        Row_count = 0
        For irow = 0 To UBound(arrEye_Data)
            '-----Header info-----
            If irow = imgWidth * Row_count Then
                MyWriter2.WriteLine("Row " & Row_count)
                MyWriter2.WriteLine("Pixels")
                MyWriter2.WriteLine(imgWidth)
                Row_count = Row_count + 1
            End If

            MyWriter2.WriteLine(arrEye_Data(irow, 0, 1) & "," & arrEye_Data(irow, 1, 1) &
»               "," & arrEye_Data(irow, 2, 1))
        Next irow


        MyWriter2.Close()
        MyStream2.Close()

    End Sub



    'Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    '    Call Shrink_Canvas()
    'End Sub

    'Private Sub cmdGet_Hues_Click(ByVal sender As System.Object, ByVal e As System.EventA
»       rgs)

    '    '=========Init Vars============
    '    Dim irow, iCol, idata As Int64, tmp As String
    '    Dim icol_Count As Int32, MyImg As Bitmap
    '    Dim Pixel_Count As Long, tmpArr() As String
    '    Dim iGreen, iRed, iBlue, iPic As Int16
    '    Dim tmpTop, tmpBot As Double

    '    MyImg = PIC1.Image
    '    Pixel_Count = ((MyImg.Height + 1) * (MyImg.Width + 1)) - 1
    '    Dim arrEye_Data(Pixel_Count, 1) As Double
    '    Dim arrEye_Final(Pixel_Count) As Double

    '    '=========Scan Every Pixel==========
```

```
    '    For iPic = 0 To 1
    '        '---Select Pic to Scan---
    '        If iPic = 0 Then
    '            MyImg = PIC1.Image
    '        Else
    '            MyImg = PIC2.Image
    '        End If

    '        For irow = 0 To MyImg.Height.ToString - 1
    '            For iCol = 0 To MyImg.Width.ToString - 1
    '                '----Get RGB Components-----
    '                tmpArr = Split(MyImg.GetPixel(iCol, irow).ToString, ",")
    '                iRed = Mid(tmpArr(1), 4)
    '                iGreen = Mid(tmpArr(2), 4)
    '                tmp = Mid(tmpArr(3), 4)
    '                iBlue = Mid(tmp, 1, Len(tmp) - 1)

    '                '---Calculate Invariant Hue---
    '                'log(R) - Log(G) / Log(R)+ Log(G) - 2Log(B)
    '                tmpTop = Log(iRed) - Log(iGreen)
    '                tmpBot = Log(iRed) + Log(iGreen) - 2 * Log(iBlue)

    '                '---Store Hue---
    '                arrEye_Data(irow + iCol, iPic) = tmpTop / tmpBot


    '            Next iCol
    '        Next irow
    '    Next iPic

    '    '===========Difference Pixel Data=============
    '    For idata = 0 To UBound(arrEye_Data) - 1

    '        arrEye_Final(idata) = arrEye_Data(idata, 0) - arrEye_Data(idata, 1)
    '        'Debug.WriteLine(arrEye_Final(idata) & "%")
    '    Next

    '    '  Call Write_to_image(arrEye_Final)
    'End Sub



    'Private Sub cmdMorph_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       )
    '    Dim N As Int32
    '    'If picHidden.Image Is Nothing Then
    '    picHidden.Image = New Bitmap(CInt(PIC1.Image.Width), CInt(PIC1.Image.Height))
    '    'End If
    '    picHidden.Left = 100
    '    picHidden.Top = 100
    '    N = InputBox("How many Sections? (9+)", "9")
    '    Dim myMorph As New CMorph(N, PIC1.Image, PIC2.Image, picHidden)
    '    myMorph.Scan_AND_Compare()
    'End Sub




    'Private Sub cmdRejig_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       )
    '    Dim bob As New MiniReg(pic1before.Image, Pic2Before.Image, 100, 10)
    '    Dim i As Int32
    '    bob.Get_Best_Match(arrOF_Points_PIC1, arrOF_Points_PIC2)


    '    '----Refresh Image----
    '    'Dim myIMG As New Bitmap(Pic2Before.Image())

    '    Draw_Cursor(arrOF_Points_PIC2(0).X, arrOF_Points_PIC2(0).Y, 0, PIC2)
    '    Draw_Cursor(arrOF_Points_PIC2(1).X, arrOF_Points_PIC2(1).Y, 1, PIC2)
    '    'Dim Gfx As Graphics = Graphics.FromImage(myIMG)
    '    'For i = 0 To UBound(arrOF_Points_PIC2)
    '    '    'Gfx.DrawLine(Pens.White, (arrOF_Points_PIC2(i).X - 1), (arrOF_Points_PIC2(i
»       ).Y - 1), (arrOF_Points_PIC2(i).X - 1), (arrOF_Points_PIC2(i).Y - 1))
```

```
    '    '    myIMG.SetPixel(arrOF_Points_PIC2(i).X + 1, arrOF_Points_PIC2(i).Y, Color.Whi
»       te)
    '    '    myIMG.SetPixel(arrOF_Points_PIC2(i).X, arrOF_Points_PIC2(i).Y, Color.White)
    '    '    myIMG.SetPixel(arrOF_Points_PIC2(i).X - 1, arrOF_Points_PIC2(i).Y, Color.Whi
»       te)
    '    '    myIMG.SetPixel(arrOF_Points_PIC2(i).X, arrOF_Points_PIC2(i).Y + 1, Color.Whi
»       te)
    '    '    myIMG.SetPixel(arrOF_Points_PIC2(i).X, arrOF_Points_PIC2(i).Y - 1, Color.Whi
»       te)

    '    'Next i
    '    '  Gfx.DrawLine(Pens.White, 0, 0, 100, 100)

    '    'PIC2.Image = myIMG

    '    '----Clean up-----
    '    bob = Nothing
    'End Sub







    'Private Sub cmdSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    '    Dim bm As New Bitmap(Pic3dOne.Image)
    '    Dim bm2 As New Bitmap(Pic3dTwo.Image)
    '    Dim bm3 As New Bitmap(picHidden.Image)
    '    bm.Save("c:\one.tif", Imaging.ImageFormat.Tiff)
    '    bm2.Save("c:\two.tif", Imaging.ImageFormat.Tiff)
    '    'bm3.Save("c:\arid.tif", Imaging.ImageFormat.Tiff)
    'End Sub



    Private Sub Colourise_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles cmdColourise.Click, cmdColourise2.Click, cmdColourise3.Click
        '---green colour bits-----
        ' Dim myColour As New CColourise(picHidden.Image, 1, 1)
        'myColour.recolour_Pixels()
        Dim myHist As CEnhance
        Dim bob As Int16
        '----Equalise Histogram----
        If sender Is cmdColourise Then
            bob = 1
        ElseIf sender Is cmdColourise2 Then
            bob = 2
        Else
            bob = 3
        End If

        myHist = New CEnhance(picHidden.Image, bob, "", tmp2Del)
        PicDisplay.Image = myHist.Equalise_Histogram()
        PicDisplay.Refresh()

        '---hidden image without grid on---
        PicEnhance_Hidden.Image = PicDisplay.Image
        PicEnhance_Hidden.Text = bob

        '---Add Grid if turned on---
        If cmdGrid.Text = "Grid Off" Then
            Call Grid()
            Call Grid()
        End If


    End Sub



    Private Sub Grid()
        If PicDisplay.Image Is Nothing = False Then
            If Grid_On = False Then
```

```
            Dim bm As New Bitmap(PicDisplay.Image)

            Dim myG As Graphics = Graphics.FromImage(bm)

            Dim X_line, Y_Line As Int32
            Dim myPen As Pen = Pens.Goldenrod
            X_line = PicDisplay.Image.Width / 3
            Y_Line = PicDisplay.Image.Height / 3

            myG.DrawLine(myPen, X_line, 0, X_line, PicDisplay.Height)
            myG.DrawLine(myPen, X_line * 2, 0, X_line * 2, PicDisplay.Height)
            myG.DrawLine(myPen, 0, Y_Line, PicDisplay.Width, Y_Line)
            myG.DrawLine(myPen, 0, Y_Line * 2, PicDisplay.Width, Y_Line * 2)
            PicDisplay.Refresh()



            Grid_On = True
            cmdGrid.Text = "Grid Off"

            PicDisplay.Image = bm

            myG = Nothing

        Else
            Select Case PicEnhance_Hidden.Text
                Case 1, 2, 3
                    PicDisplay.Image = PicEnhance_Hidden.Image
                Case Else
                    PicDisplay.Image = picHidden.Image
            End Select


            PicDisplay.Refresh()

            cmdGrid.Text = "Grid On"
            Grid_On = False
        End If
    End If
End Sub

Private Sub PicDisplay_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles PicDisplay.Click
    If PicDisplay.Top <> 0 Then
        'PicDisplay.SizeMode = PictureBoxSizeMode.StretchImage
        PicDisplay.Top = 0
        PicDisplay.Left = 0
        PicDisplay.Width = Me.Width - (GrpEnhance.Width + 20)
        PicDisplay.Height = Me.Height
        PIC2.Visible = False
        PicDisplay.BringToFront()
        PicDisplay.Refresh()
    Else
        'PicDisplay.SizeMode = PictureBoxSizeMode.Normal
        PicDisplay.Top = lblInfo_A.Top + lblInfo_A.Height + 5
        PicDisplay.Left = PIC1.Left
        PicDisplay.Height = (Me.Height - 35) - (PIC1.Height + lblInfo_A.Height + PIC1.
»           Top)
        PicDisplay.Width = Me.Width * 0.33
        PIC2.Visible = True
        PicDisplay.Refresh()
    End If
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Dim bob As New CEnhance(PIC1.Image, 3, "", tmp2Del)
    bob.Equalise_Histogram()
    PIC1.Refresh()
End Sub



Private Sub cmdUndo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
»       Handles cmdUndo.Click
        PicDisplay.Image = picHidden.Image
        PicEnhance_Hidden.Text = ""
        If cmdGrid.Text = "Grid Off" Then
            Call Grid()
            Call Grid()
        End If
    End Sub



    Private Sub cmdClear_Click_1(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles cmdClear.Click, cmdReset2.Click
        Erase arrOF_Points_PIC1
        Erase arrOF_Points_PIC2
        ReDim arrOF_Points_PIC1(1)
        ReDim arrOF_Points_PIC2(1)
        iPoints_PIC1 = 0
        iPoints_PIC2 = 0
        'If sender Is cmdClear Then
        '    '   lblInfo_A.Text = "Points Selected = "
        '    '  lblInfo_B.Text = "Points Selected = "
        'Else
        '    lblInfo_A.Text = Mid(lblInfo_A.Text, 1, Len(lblInfo_A.Text) - 2) & "0)"
        '    lblInfo_B.Text = Mid(lblInfo_B.Text, 1, Len(lblInfo_B.Text) - 2) & "0)"
        'End If
        PIC1.Image = pic1before.Image
        PIC2.Image = Pic2Before.Image
        PIC2.Visible = True
        'PicDisplay.Image.Dispose()
        'picHidden.Image.Dispose()
        Translate_by = Nothing
        Rotate_By = Nothing
        Origin = Nothing
        PicDisplay.Visible = False
        Grid_On = False
        cmdGrid.Text = "Grid On"
        GrpEnhance.Visible = False
        ' Call start_up()
        Me.lblinfo.Visible = True
        ' Me.cmdARID.Visible = True
        Me.cmdLoad_A.Visible = True
        Me.cmdLoad_B.Visible = True
        Me.cmdReset2.Visible = True


        Call Me.FRMArid_Load(sender, e)
        Call Me.FRMArid_Resize(sender, e)
    End Sub

    Private Sub cmdGrid_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles cmdGrid.Click
        Call Grid()
    End Sub




    'Private Sub Colour_Click(ByVal which_colour As Int32)
    '    Dim myColour As New CColourise(picHidden.Image)
    '    PicDisplay.Image = myColour.Change_Colour(which_colour)
    '    'PicDisplay.Refresh()
    '    'PicDisplay.Visible = True
    'End Sub


    Private Sub cmdArid_Auto_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles cmdArid_Auto.Click

        '-crop so as to only process retina-
        Dim myRec1, myRec2 As Rectangle
        myRec1 = Crop(Pic1_Original.Image, 1)
        myRec2 = Crop(Pic2_Original.Image, 2)


        PIC1.Refresh()
```

```
        PIC2.Refresh()
        Retina_rectangle = myRec1 'global var set so i can use it in get blue cmd
        '--make all image analysis the same size--
        Call zReduce_Images(576, 768)
        '  Call zReduce_Images(768, 576)

        '-store ratio of image size increase/decrease-
        Dim ratioX1, ratioX2, ratioY1, ratioY2 As Double

        ratioX1 = myRec1.Width / Pic1_Original.Image.Width
        ratioY1 = myRec1.Height / Pic1_Original.Image.Height

        ratioX2 = myRec2.Width / Pic2_Original.Image.Width
        ratioY2 = myRec2.Height / Pic2_Original.Image.Height
        'ratioX1 = 1
        'ratioY1 = 1

        'ratioX2 = 1
        'ratioY2 = 1

        '-find vessels-
        Call cmdVessel_Click(sender, e)

        '-Correlate-
        Call Correlate_Images(ratioX1, ratioX2, ratioY1, ratioY2)

    End Sub


    'Sub Blend_3D(ByVal img1 As Bitmap, ByVal img2 As Bitmap, ByVal Alpha As Int32)
    '    '----Combine & Output'-------
    '    Dim bm_Pic2 As New Bitmap(img1)

    '    Dim bm_Final As New Bitmap(img2)
    '    Dim iX, iY, r, g, b As Int32
    '    Dim tmp, tmp2(), tmp3() As String

    '    '---Halve the Alpha Lvl of Pic2---
    '    For iX = 0 To bm_Pic2.Width - 1
    '        For iY = 0 To bm_Pic2.Height - 1
    '            '---get rgb----
    '            tmp = bm_Pic2.GetPixel(iX, iY).ToString
    '            tmp2 = Split(tmp, ",")
    '            tmp3 = Split(tmp2(1), "=")
    '            r = tmp3(1)
    '            tmp3 = Split(tmp2(2), "=")
    '            g = tmp3(1)
    '            tmp3 = Split(tmp2(3), "=")
    '            b = Mid(tmp3(1), 1, Len(tmp3(1)) - 1)
    '            '---------------------
    '            bm_Pic2.SetPixel(iX, iY, Color.FromArgb(Alpha, r, g, b))
    '        Next
    '    Next


    '    Dim gr As Graphics = Graphics.FromImage(bm_Final)
    '    'gr.DrawBezier(Pens.Aquamarine, 10, 10, 20, 20, 40, 40, 100, 100)

    '    gr.DrawImage(bm_Pic2, -5, 0)


    '    '------Display the result--------------
    '    PicDisplay.Image = bm_Final
    '    picHidden.Image = bm_Final
    '    PicDisplay.Visible = True
    '    PicDisplay.Refresh()
    'End Sub
    'Private Sub cmdColour_Click(ByVal sender As System.Object, ByVal e As System.EventArg
»       s)
    '    Call Colour_Click(0)
    'End Sub



    Private Sub cmdFlip_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
        Dim myFRMFlip As New FRMFlip()
        '---init frm before display---
        myFRMFlip.TrackBar.Value = Track_Bar_Value
        myFRMFlip.PIC1.Image = PIC1.Image
        myFRMFlip.Pic2.Image = PIC2.Image
        myFRMFlip.Timer_Flip.Enabled = True
        myFRMFlip.Show()
    End Sub



    Sub Draw_Cursor(ByVal x As Single, ByVal y As Single, ByVal Which_Cursor As Single,
»       ByVal curr_Image As PictureBox)
        '--init vars----
        Dim myG As Graphics
        Dim bm_out As Bitmap

        Dim bm_cursor, cursor_Image1, cursor_Image2, original_Image As Bitmap
        Dim myRegion1, myRegion2 As Region
        '---Define Image Ratio---
        Dim Width_Ratio, Height_Ratio As Double
        Width_Ratio = PIC1.Image.Width / PIC1.Width
        Height_Ratio = PIC1.Image.Height / PIC1.Height
        Dim myRect = New Rectangle(x, y, (25 * Width_Ratio), (25 * Height_Ratio))
        '---Defined for region info---
        Dim myPAth As New GraphicsPath()
        myPAth.AddRectangle(myRect)

        '---choose which image we are using---
        If curr_Image Is PIC1 Then
            If ZoomRatio1 = 0 Then
                cursor_Image1 = pic1_cursor1.Image
                cursor_Image2 = pic1_cursor2.Image
                original_Image = pic1before.Image
                myRegion1 = Pic1_Cursor_one_Region
                myRegion2 = Pic1_Cursor_two_Region
            Else
                cursor_Image2 = PIC1.Image

                cursor_Image1 = cursor_Image2
                original_Image = New Bitmap(Pic1_Zoom_Cursor.Image)
                myRegion1 = Pic1_Zoom_Cursor_Region
            End If
        Else
            If ZoomRatio2 = 0 Then
                cursor_Image1 = pic2_Cursor1.Image
                cursor_Image2 = pic2_Cursor2.Image
                original_Image = Pic2Before.Image
                myRegion1 = Pic2_Cursor_one_Region
                myRegion2 = Pic2_Cursor_two_Region
            Else
                cursor_Image2 = PIC2.Image
                cursor_Image1 = cursor_Image2
                original_Image = New Bitmap(Pic2_Zoom_Cursor.Image)
                myRegion1 = Pic2_Zoom_Cursor_Region
            End If
        End If

        '---decide which pic to use depending on cursor clicked----
        If Which_Cursor = 0 Then
            bm_out = New Bitmap(cursor_Image2)
        Else
            bm_out = New Bitmap(cursor_Image1)
        End If

        '---draw moved cursor on Pic1---
        myG = Graphics.FromImage(bm_out)
        ' myG.DrawImage(ILST_Cursors.Images(Which_Cursor), myRect)
        curr_Image.Image = bm_out
        curr_Image.Refresh()
        myG = Nothing

        If Which_Cursor = 0 Then
            '---update cursor images---
            cursor_Image1 = New Bitmap(original_Image)
```

```
            bm_cursor = cursor_Image1

            '---save region defined---
            myRegion1 = New Region(myPAth)
        Else
            cursor_Image2 = New Bitmap(original_Image)
            bm_cursor = cursor_Image2

            '---save region defined---
            myRegion2 = New Region(myPAth)
        End If

        myG = Graphics.FromImage(bm_cursor)
        ' myG.DrawImage(ILST_Cursors.Images(Which_Cursor), myRect)


        '---Update Cursor piccys and regions----
        If curr_Image Is PIC1 Then
            If ZoomRatio1 = 0 Then
                If Which_Cursor = 0 Then
                    pic1_cursor1.Image = bm_cursor
                    Pic1_Cursor_one_Region = myRegion1
                Else
                    pic1_cursor2.Image = bm_cursor
                    Pic1_Cursor_two_Region = myRegion2
                End If
            Else
                ' Pic1_Zoom_Cursor.Image = bm_cursor
                Pic1_Zoom_Cursor_Region = myRegion1
            End If



        Else
            If ZoomRatio2 = 0 Then
                If Which_Cursor = 0 Then
                    pic2_Cursor1.Image = bm_cursor
                    Pic2_Cursor_one_Region = myRegion1
                Else
                    pic2_Cursor2.Image = bm_cursor
                    Pic2_Cursor_two_Region = myRegion2
                End If
            Else
                '  Pic2_Zoom_Cursor.Image = bm_cursor
                Pic2_Zoom_Cursor_Region = myRegion1
            End If
        End If


    End Sub


    Private Sub Sender_MouseUp(ByVal sender As System.Object, ByVal e As System.Windows.
»       Forms.MouseEventArgs) Handles PIC2.MouseUp, PIC1.MouseUp
        '===choose co-ords to compare===
        Dim co_Ords As Point, zoomratio As Int32

        If sender Is PIC1 Then
            zoomratio = ZoomRatio1
        Else
            zoomratio = ZoomRatio2
        End If

        If optMove.Checked = True And zoomratio <> 0 Then 'moving pic
            co_Ords = Before_XY
        ElseIf optCursors.Checked = True And zoomratio <> 0 Then 'moving cursor
            co_Ords = Zoom_Before_XY
        ElseIf optCursors.Checked = True And zoomratio = 0 Then
            co_Ords = Before_XY
        Else
            Exit Sub
        End If
        '---Move image or cursors---
        If ((co_Ords.X <> e.X And co_Ords.X <> 0) Or (co_Ords.Y <> e.Y And co_Ords.Y <> 0)
»           ) _
        And e.X <= sender.width And e.Y <= sender.height And e.X > 0 And e.Y > 0 Then
```

```
'---init Vars---
Dim X, Y, middle_X, middle_Y, image_X, image_Y As Int32
Dim Width_Ratio, Height_Ratio As Double
Dim myRec As RectangleF
Dim myG As Graphics
Dim myRegion1, myRegion2 As Region
Dim offset_X, offset_Y As Int32

'---workout pixel ratio used---
Width_Ratio = sender.Image.Width / sender.Width
Height_Ratio = sender.Image.Height / sender.Height
X = CInt(e.X * Width_Ratio)
Y = CInt(e.Y * Height_Ratio)

'--Calc movement of mouse---
offset_X = X - co_Ords.X
offset_Y = Y - co_Ords.Y


If optMove.Checked = True Then
    '

    '---Zoom to right part---
    If sender Is PIC1 Then
        If ZoomRatio1 <> 0 Then
            Call Img_Zoom(sender, X, Y, offset_X, offset_Y)
            Pic1_Zoom_Cursor.Image = PIC1.Image
        End If
    Else
        If ZoomRatio2 <> 0 Then
            Call Img_Zoom(sender, X, Y, offset_X, offset_Y)
            Pic2_Zoom_Cursor.Image = PIC2.Image
        End If
    End If

ElseIf OptZoom.Checked = True Then
    'do nothing 'ERROR HERE OTHERWISE
Else

    '---get correct starting cursor image and region---
    If sender Is PIC1 Then
        If ZoomRatio1 = 0 Then
            sender.Image = pic1before.Image
            myG = Graphics.FromImage(PIC1.Image)
            myRegion1 = Pic1_Cursor_one_Region
            myRegion2 = Pic1_Cursor_two_Region
        Else

            Dim tmpI As Bitmap = New Bitmap(Pic1_Zoom_Cursor.Image)
            sender.image = tmpI
            myG = Graphics.FromImage(tmpI)
            myRegion1 = Pic1_Zoom_Cursor_Region
            myRegion2 = myRegion1
        End If
    Else
        If ZoomRatio2 = 0 Then
            sender.Image = Pic2Before.Image
            myG = Graphics.FromImage(PIC2.Image)
            myRegion1 = Pic2_Cursor_one_Region
            myRegion2 = Pic2_Cursor_two_Region
        Else
            Dim tmpI2 As Bitmap = New Bitmap(Pic2_Zoom_Cursor.Image)
            sender.image = tmpI2
            myG = Graphics.FromImage(tmpI2)
            myRegion1 = Pic1_Zoom_Cursor_Region
            myRegion2 = myRegion1
        End If
    End If

    '---get centre of cursor---
    middle_X = (25 * Width_Ratio) / 2
    middle_Y = (25 * Height_Ratio) / 2
    image_X = X - middle_X
    image_Y = Y - middle_Y

    '---Move cursor to new position---
```

```
                If Which_cursor = 0 Then

                    myRec = myRegion2.GetBounds(myG)
                    Draw_Cursor(image_X, image_Y, Which_cursor, sender)

                    '---draw extra cursor if needed--
                    If zoomratio = 0 Then
                        Draw_Cursor(myRec.X, myRec.Y, 1, sender)
                    Else
                        arrOF_Points_Zoomed(0).X = image_X
                        arrOF_Points_Zoomed(0).Y = image_Y
                    End If
                Else
                    myRec = myRegion1.GetBounds(myG)
                    Draw_Cursor(image_X, image_Y, Which_cursor, sender)

                    If zoomratio = 0 Then
                        Draw_Cursor(myRec.X, myRec.Y, 0, sender)
                    Else
                        arrOF_Points_Zoomed(1).X = image_X
                        arrOF_Points_Zoomed(1).Y = image_Y
                    End If

                End If


                sender.Refresh()
                Before_XY = Nothing

                '--------Add new Position to correct Array----
                Dim curr_Point As New Point(image_X, image_Y)
                If sender Is PIC1 Then
                    arrOF_Points_PIC1(Which_cursor) = curr_Point
                Else
                    arrOF_Points_PIC2(Which_cursor) = curr_Point
                End If

            End If

        End If
        Me.Cursor = Cursors.Default
    End Sub



    'Private Sub cmdPixel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       )
    '    '---Init Vars---
    '    Dim mySnake As CSnake
    '    Dim Width_ratio, Height_ratio As Double
    '    Dim myP As Point

    '    Width_ratio = PIC1.Image.Width / PIC1.Width
    '    Height_ratio = PIC1.Image.Height / PIC1.Height

    '    myP.X = tmp2Del.X * Width_ratio
    '    myP.Y = tmp2Del.Y * Height_ratio

    '    mySnake = New CSnake(myP, Pic1_Original.Image)

    '    '--trace vessel path--
    '    PIC1.Image = mySnake.Trace_Path



    'End Sub



    Private Sub cmdEnhance1_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles cmdEnhance1.Click, cmdEnhance2.Click, cmdEnhance3.Click,
»       cmdEnhance4.Click
        Dim ID As Int32
        Dim myE As CEnhance
        If sender Is cmdEnhance1 Then
            ID = 1
```

```
        ElseIf sender Is cmdEnhance2 Then
            ID = 2
        ElseIf sender Is cmdEnhance3 Then
            ID = 3

        Else
            PIC1.Image = Pic1_Original.Image

            Call FRMArid_Load(cmdLoad_A, e)
            PIC1.Refresh()
            Exit Sub
            'ID = 4
        End If

        myE = New CEnhance(pic1before.Image, ID, "", tmp2Del)

        PIC1.Image = myE.Equalise_Histogram()


        Call FRMArid_Load(cmdLoad_A, e)
        PIC1.Refresh()
    End Sub

    Private Sub cmdEnhance5_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles cmdEnhance5.Click, cmdEnhance6.Click, cmdEnhance7.Click,
»       cmdEnhance8.Click
        Dim ID As Int32
        If sender Is cmdEnhance5 Then
            ID = 1
        ElseIf sender Is cmdEnhance6 Then
            ID = 2
        ElseIf sender Is cmdEnhance7 Then
            ID = 3
        Else
            PIC2.Image = Pic2_Original.Image
            'pic2_Cursor1.Image = PIC2.Image
            'pic2_Cursor2.Image = PIC2.Image
            'Pic2Before.Image = PIC2.Image
            Call FRMArid_Load(cmdLoad_B, e)
            PIC2.Refresh()
            Exit Sub
        End If

        Dim myE As New CEnhance(Pic2Before.Image, ID, "", tmp2Del)
        PIC2.Image = myE.Equalise_Histogram()
        Call FRMArid_Load(cmdLoad_B, e)
        PIC2.Refresh()
    End Sub



    Sub Img_Zoom(ByVal sender As Object, ByVal X As Int32, ByVal Y As Int32, ByVal
»       offset_X As Int32, ByVal offset_y As Int32)


        '---Init Vars----
        Dim baseIMG As Bitmap
        Dim Zoom As Double


        Dim Width_Ratio, Height_Ratio As Double


        If sender Is PIC1 Then
            Zoom = 1 + (ZoomRatio1 / 10) ' ZoomRatio is class var
            baseIMG = New Bitmap(Pic1_Original.Image())

            If Zoom = 1 Then
                OLD_Selected_Part1 = Nothing
                sender.image = baseIMG

                Exit Sub
            End If
```

```
        Else
            baseIMG = New Bitmap(Pic2_Original.Image())
            Zoom = 1 + (ZoomRatio2 / 10) ' ZoomRatio is class var

            If Zoom = 1 Then
                OLD_Selected_Part2 = Nothing
                sender.image = baseIMG

                Exit Sub
            End If

        End If



        Width_Ratio = sender.Width / sender.image.Width
        Height_Ratio = sender.Height / sender.image.Height


        Dim Apparant_W = CInt(sender.Image.width * Width_Ratio)
        Dim Apparant_H = CInt(sender.Image.height * Height_Ratio)


        '----------------------
        '---Zoom Hidden Piccy----
        Dim piccy2 As New Bitmap(CInt(Apparant_W * Zoom), CInt(Apparant_H * Zoom))
        Dim myg4 As Graphics = Graphics.FromImage(piccy2)
        Dim newrec As New Rectangle(0, 0, CInt(Apparant_W * Zoom), CInt(Apparant_H * Zoom)
»           )
        Dim currrec As New Rectangle(0, 0, baseIMG.Width, baseIMG.Height)

        myg4.DrawImage(baseIMG, newrec, currrec, GraphicsUnit.Pixel)


        If sender Is PIC1 Then
            pic1zoom.Image = New Bitmap(piccy2)
            pic1zoom.Refresh()

        Else
            pic2zoom.Image = New Bitmap(piccy2)
            pic2zoom.Refresh()

        End If

        myg4 = Nothing
        newrec = Nothing
        currrec = Nothing

        '----------------------
        Call Select_Part_of_Zoom_piccy(sender, X, Y, Apparant_W, Apparant_H, Width_Ratio,
»           Height_Ratio, piccy2, offset_X, offset_y)

    End Sub
    Sub Select_Part_of_Zoom_piccy(ByVal sender, ByVal x, ByVal y, ByVal apparant_w, ByVal
»       apparant_h, ByVal width_ratio, ByVal height_ratio, ByVal piccy2, ByVal offset_X,
»       ByVal offset_Y)
        Dim OLD_Selected_Part As Point
        Dim OLD_ZoomRatio, Curr_ZoomRatio As Double
        '---Cut & paste------
        '--select correct bit---

        If sender Is PIC1 Then
            OLD_Selected_Part = OLD_Selected_Part1
            OLD_ZoomRatio = old_Zoomratio1
            Curr_ZoomRatio = ZoomRatio1
        Else
            OLD_Selected_Part = OLD_Selected_Part2
            OLD_ZoomRatio = old_Zoomratio2
            Curr_ZoomRatio = ZoomRatio2
        End If

        If offset_X = 0 Then
            If offset_Y = 0 Then
                '--Zoom Out--
                If OLD_ZoomRatio > Curr_ZoomRatio Then
```

```
                x = ((x * 0.9 * width_ratio) + (OLD_Selected_Part.X * 0.9))
                y = ((y * 0.9 * height_ratio) + (OLD_Selected_Part.Y * 0.9))
                '--Zoom In--
            Else
                x = ((x * 1.1 * width_ratio) + (OLD_Selected_Part.X * 1.1))
                y = ((y * 1.1 * height_ratio) + (OLD_Selected_Part.Y * 1.1))

                End If

            Else
                x = OLD_Selected_Part.X - offset_X + (apparant_w) / 2
                y = OLD_Selected_Part.Y - offset_Y + (apparant_h) / 2

            End If
        Else
            x = OLD_Selected_Part.X - offset_X + (apparant_w) / 2
            y = OLD_Selected_Part.Y - offset_Y + (apparant_h) / 2

        End If

        Dim Selected_Part As New Rectangle(x - ((apparant_w) / 2), y - ((apparant_h) / 2),
»            apparant_w, apparant_h)


        If sender Is PIC1 Then
            OLD_Selected_Part1.X = Selected_Part.X
            OLD_Selected_Part1.Y = Selected_Part.Y
            old_Zoomratio1 = ZoomRatio1

        Else
            OLD_Selected_Part2.X = Selected_Part.X
            OLD_Selected_Part2.Y = Selected_Part.Y
            old_Zoomratio2 = ZoomRatio2
        End If


        Dim piccy3 As New Bitmap(pic1zoom.Image)
        Dim myg99 As Graphics = Graphics.FromImage(piccy3)
        myg99.DrawRectangle(Pens.Aqua, Selected_Part)
        'piccy3.SetPixel(x, y, Color.White)
        'piccy3.SetPixel(x + 1, y, Color.White)
        'piccy3.SetPixel(x, y + 1, Color.White)
        'piccy3.SetPixel(x, y - 1, Color.White)
        'piccy3.SetPixel(x - 1, y, Color.White)
        pic1zoom.Image = piccy3

        Dim blank_Img As New Bitmap(Selected_Part.Width, Selected_Part.Height)
        Dim tmprec As New Rectangle(0, 0, Selected_Part.Width, Selected_Part.Height)


        ''---draw onto correct piccy---
        Dim myG2 As Graphics = Graphics.FromImage(blank_Img)
        myG2.DrawImage(piccy2, tmprec, Selected_Part, GraphicsUnit.Pixel)

        sender.image = blank_Img

        sender.refresh()


        '---put cursors on---
        Dim e As System.EventArgs
        If sender Is PIC1 Then
            ' Call FRMArid_Load(cmdLoad_A, e)
        Else
            '  Call FRMArid_Load(cmdLoad_B, e)
        End If



    End Sub

    Sub fix_zoom_image(ByVal sender)
        Dim baseIMG As Bitmap
        Dim Width_Ratio, Height_Ratio As Double
```

```
        '---set ratios for actual size---
        Width_Ratio = sender.Width / sender.Image.Width
        Height_Ratio = sender.Height / sender.Image.Height

        If sender Is PIC1 Then
            baseIMG = New Bitmap(Pic1_Original.Image)
        Else
            baseIMG = New Bitmap(Pic2_Original.Image)
        End If

        Dim piccy1 As New Bitmap(CInt(sender.Image.width * Width_Ratio), CInt(sender.Image
»           .height * Height_Ratio))
        Dim myG3 As Graphics = Graphics.FromImage(piccy1)
        Dim currrec As New Rectangle(0, 0, sender.Image.width, sender.Image.height)
        Dim newrec As New Rectangle(0, 0, piccy1.Width, piccy1.Height)
        myG3.DrawImage(baseIMG, newrec, currrec, GraphicsUnit.Pixel)

        If sender Is PIC1 Then
            pic1zoom.Image = piccy1
        Else
            pic2zoom.Image = piccy1
        End If


        myG3 = Nothing
    End Sub

    Private Sub pic1zoom_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms
»       .MouseEventArgs) Handles pic1zoom.MouseMove
        Me.Text = "(" & e.X & ", " & e.Y & ")"
    End Sub



    Private Sub optCursors_CheckedChanged(ByVal sender As System.Object, ByVal e As System
»       .EventArgs) Handles optCursors.CheckedChanged, optMove.CheckedChanged, OptZoom.
»       CheckedChanged
        Dim Width_ratio, Height_ratio As Double

        If sender.Checked = True Then
            sender.BackColor = Color.AliceBlue
            If sender Is optCursors Then
                '---add cursors to zoomed pics---
                If ZoomRatio1 <> 0 Then

                    PIC1.Image = Pic1_Zoom_Cursor.Image
                    Call Draw_Cursor(PIC1.Width / 2, PIC1.Height / 2, 0, PIC1)
                    arrOF_Points_Zoomed(0).X = PIC1.Width / 2
                    arrOF_Points_Zoomed(0).Y = PIC1.Height / 2
                End If

                If ZoomRatio2 <> 0 Then
                    Call Img_Zoom(PIC2, (PIC2.Width / 2), (PIC2.Height / 2), 0, 0)
                    Pic2_Zoom_Cursor.Image = PIC2.Image
                    Call Draw_Cursor(PIC2.Width / 2, PIC2.Height / 2, 0, PIC2)
                    arrOF_Points_Zoomed(1).X = PIC2.Width / 2
                    arrOF_Points_Zoomed(1).Y = PIC2.Height / 2
                End If

            End If
        Else
            sender.BackColor = Color.Transparent
        End If
    End Sub




    'Private Sub cmdOptic_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
```

```
»       )
    '    Dim myC As New CClock(Pic1_Original.Image)
    '    myC = New CClock(Pic1_Original.Image)
    '    '----Best Intensity----
    '    PIC1.Image = myC.get_Best_Intensity(Pic1_Original.Image)
    'End Sub

    Private Sub PIC1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
»       Handles PIC1.Click

    End Sub

    Private Sub cmdThreshold_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs)
        Dim Width_Ratio, Height_Ratio As Double

        Width_Ratio = PIC1.Image.Width / PIC1.Width
        Height_Ratio = PIC1.Image.Height / PIC1.Height
        Dim myP As Point
        myP.X = tmp2Del.X * Width_Ratio
        myP.Y = tmp2Del.Y * Height_Ratio
        Dim myE As New CEnhance(Pic1_Original.Image, 1, Me.txtThreshold.Text, myP)
        PIC1.Image = myE.Thresholding
    End Sub

    'Private Sub cmdT_AND_S_Click(ByVal sender As System.Object, ByVal e As System.EventAr
»       gs)
    '    Dim Width_Ratio, Height_Ratio As Double

    '    Width_Ratio = PIC1.Image.Width / PIC1.Width
    '    Height_Ratio = PIC1.Image.Height / PIC1.Height
    '    Dim myP As Point
    '    myP.X = tmp2Del.X * Width_Ratio
    '    myP.Y = tmp2Del.Y * Height_Ratio
    '    Dim myComplicated As New CT_and_S(Pic1_Original.Image, myP)
    '    PIC1.Image = myComplicated.Snake_Path_With_Thresholding
    'End Sub

    Function Crop(ByVal mypic As Image, ByVal WhichOne As Int16) As Rectangle
        '--crop image--
        Dim recCrop, recDest As Rectangle
        Dim bmpImage, bmpCrop As Image
        Dim gphCrop As Graphics
        Dim myP As CVessels
        myP = New CVessels(mypic, txtGrp.Text, "")
        recCrop = myP.Get_Actual_Retina()
        ' recCrop.X = recCrop.X + recCrop.Width * 0.1
        ' recCrop.Y = recCrop.Y + recCrop.Width * 0.1
        ' recCrop.Width = recCrop.Width * 0.8
        ' recCrop.Height = recCrop.Height * 0.8

        bmpImage = New Bitmap(mypic)

        bmpCrop = New Bitmap(recCrop.Width, recCrop.Height, bmpImage.PixelFormat)
        gphCrop = Graphics.FromImage(bmpCrop)
        recDest = New Rectangle(0, 0, recCrop.Width, recCrop.Height)
        gphCrop.DrawImage(bmpImage, recDest, recCrop.X, recCrop.Y, recCrop.Width, _
            recCrop.Height, GraphicsUnit.Pixel)
        If WhichOne = 1 Then
            PIC1.Image = bmpCrop
        Else
            PIC2.Image = bmpCrop
        End If
        Crop = recCrop
    End Function

    Public Sub cmdVessel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
»       Handles cmdVessel_A.Click
        '---Init Vars--
        Dim arrOne(), arrTwo() As Point
        Dim myP As CVessels

        '-crop so as to only process retina-
        Dim myRec1, myRec2 As Rectangle
```

```
        myRec1 = Crop(Pic1_Original.Image, 1)
        myRec2 = Crop(Pic2_Original.Image, 2)
        PIC1.Refresh()
        PIC2.Refresh()

        ''--shrink image--
        Call zReduce_Images(576, 768)

        myP = New CVessels(PIC1.Image, txtGrp.Text, "")
        ''--Get Pixels in A--


        arrOne = myP.Get_Best_Pixels
        ''reset pic between drawings

        ''--Draw Images--
        pic1_v_H.Image = myP.Draw(0)
        pic1_V_V.Image = myP.Draw(1)
        pic1_V_GRP.Image = myP.Draw(2)
        pic1_V_All.Image = myP.Draw(999)
        pic1_v_BW.Image = myP.Create_Dot_Image()
        PIC1.Image = pic1_v_BW.Image
        PIC1.Refresh()
        myP = Nothing


        myP = New CVessels(PIC2.Image, txtGrp.Text, "")
        '--Get Pixels in B--


        arrTwo = myP.Get_Best_Pixels

        PIC2_V_Grp.Image = myP.Draw(2)
        pic2_V_H.Image = myP.Draw(0)
        PIC2_V_V.Image = myP.Draw(1)
        pic2_V_All.Image = myP.Draw(999)
        pic2_v_bw.Image = myP.Create_Dot_Image()
        myP = Nothing
        PIC2.Image = pic2_v_bw.Image



    End Sub

    Private Sub lblInfo_B_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles lblInfo_B.Click

    End Sub

    Private Sub cmdVessel_Pic_Change_Click(ByVal sender As System.Object, ByVal e As
»       System.EventArgs) Handles cmdVessel_Pic_Change.Click, cmdVessel_Pic_Change2.Click,
»        cmdVessel_Pic_Change3.Click, cmdVessel_Pic_Change4.Click, cmdVessel_Pic_Change5.
»       Click
        If sender Is cmdVessel_Pic_Change Then
            PIC1.Image = pic1_v_H.Image
            PIC2.Image = pic2_V_H.Image
        ElseIf sender Is cmdVessel_Pic_Change2 Then
            PIC1.Image = pic1_V_V.Image
            PIC2.Image = PIC2_V_V.Image
        ElseIf sender Is cmdVessel_Pic_Change3 Then
            PIC1.Image = pic1_V_All.Image
            PIC2.Image = pic2_V_All.Image
        ElseIf sender Is cmdVessel_Pic_Change4 Then
            PIC1.Image = pic1_V_GRP.Image
            PIC2.Image = PIC2_V_Grp.Image
        ElseIf sender Is cmdVessel_Pic_Change5 Then
            PIC1.Image = pic1_v_BW.Image
            PIC2.Image = pic2_v_bw.Image
        End If
    End Sub


    Private Sub Time_Generic_Tick(ByVal sender As System.Object, ByVal e As System.
```

```
»       EventArgs) Handles Time_Generic.Tick
        Static Secs As Int32

        Secs = Secs + 1
    End Sub

    Public Sub Correlate_Images(ByVal ratioX1, ByVal ratioX2, ByVal ratioY1, ByVal ratioY2
»       )
        '-Init Vars-
        Dim Factor As Int32 = 5
        Dim myVector As New Vector()
        Dim myE As CEnhance
        Dim tmp As Point
        Dim myShadow As DeShadow, myShadow2 As DeShadow
        '-shrink images-
        '
        Call zReduce_Images(Factor)

        '-deshadow-
        ' myShadow = New DeShadow(10, PIC1.Image)
        ' myShadow2 = New DeShadow(10, PIC2.Image)
        ' Call zRestore_Images()
        '  PIC1.Image = myShadow.deshadow
        ' PIC2.Image = myShadow2.deshadow

        ' Call zReduce_Images(Factor)
        '-Equalise Histogram-
        '    myE = New CEnhance(PIC1.Image, 1, "", tmp)
        '    PIC1.Image = myE.Equalise_Histogram()
        '   myE = New CEnhance(PIC2.Image, 1, "", tmp)
        '    PIC2.Image = myE.Equalise_Histogram()
        ' PIC1.Refresh()
        '  PIC2.Refresh()


        '-build class-
        Dim myCor = New CNorm_Corralate(PIC1.Image, PIC2.Image)

        '-Calc Translation and rotation needed to match-
        'myVector = myCor.Correlate_Mini_match(Factor)
        '  myVector = myCor.Final_Correlation(Factor)
        myVector = myCor.Correlate_match(Factor)
        'Crop originals again now
        Dim myRec1, myRec2 As Rectangle
        myRec1 = Crop(Pic1_Original.Image, 1)
        myRec2 = Crop(Pic2_Original.Image, 2)
        PIC1.Refresh()
        PIC2.Refresh()
        PIC1.Image = Pic1_Original.Image
        PIC2.Image = Pic2_Original.Image
        PIC1.Refresh()
        PIC2.Refresh()
        '-Increase Vector by Factor-
        Dim Ratio_From_First_Resize_X, Ratio_From_First_Resize_Y As Double
        ' Ratio_From_First_Resize_X = 1
        '  Ratio_From_First_Resize_Y = 1
        Ratio_From_First_Resize_X = Pic1_Original.Image.Width / 768
        Ratio_From_First_Resize_Y = Pic1_Original.Image.Height / 568
        myVector.myP.X = myVector.myP.X * Factor * ratioX1 * Ratio_From_First_Resize_X
        myVector.myP.Y = myVector.myP.Y * Factor * ratioY1 * Ratio_From_First_Resize_Y
        Debug.WriteLine("MyVector X=" & myVector.myP.X & ", Y=" & myVector.myP.Y & ", Thet
»           a=" & myVector.myT)


        '-Unshrink images-
        'Call zRestore_Images()

        '-Apply Change-

        'picHidden.Image = zTranslate_rotate_Image(myVector)
        PicDisplay.Image = zMerge_Images(myVector)
        'picHidden.Image = PicDisplay.Image
        '-make it big-
        '---also in right click of sender sub--
        PicDisplay.Top = lblInfo_A.Top + lblInfo_A.Height
        PicDisplay.Left = PIC1.Left
```

```
        PicDisplay.Width = PicDisplay.Image.Width
        PicDisplay.Height = (Me.Height - 35) - (PIC1.Height + lblInfo_A.Height + PIC1.Top)
        PicDisplay.Width = Me.Width * 0.33
        PicDisplay.Visible = True
        PicDisplay.Refresh()

        picHidden.Image = PicDisplay.Image
        '--------------------------------
        PicDisplay.BringToFront()
        cmdPrint.BringToFront()
        cmdClear.BringToFront()
        cmdSave.BringToFront()
        GrpEnhance.BringToFront()
        GrpEnhance.Visible = True
        cmdGrid.BringToFront()
        cmdReset2.Visible = False
        '-Clean up-
        myE = Nothing
        tmp = Nothing
        myVector = Nothing
    End Sub

    Sub zReduce_Images(ByVal Factor As Int32)
        Dim img As Image
        img = PIC1.Image
        'img = Pic1_Original.Image
        img = New Bitmap(img, New Size(img.Width * (1 / Factor), img.Height * (1 / Factor)
»           ))
        PIC1.Image = img
        img = PIC2.Image
        '   img = Pic2_Original.Image
        img = New Bitmap(img, New Size(img.Width * (1 / Factor), img.Height * (1 / Factor)
»           ))
        PIC2.Image = img

        PIC1.Refresh()
        PIC2.Refresh()
    End Sub
    Sub zReduce_Images(ByVal Height As Int32, ByVal width As Int32)
        '-init vars-
        Dim myImg = PIC1.Image  '#Image One is the one that moves
        Dim myImg2 = PIC2.Image
        Dim wid As Int32 = width
        Dim Hgt As Int32 = Height
        Dim bm_out As New Bitmap(wid, Hgt)
        Dim gr_out As Graphics = Graphics.FromImage(bm_out)

        '-set graphics objects-
        gr_out.InterpolationMode = InterpolationMode.High
        gr_out.CompositingQuality = CompositingQuality.HighQuality

        '--Define Corners of Image-
        Dim myCorners As Point() = { _
           New Point(0, 0), _
           New Point(wid, 0), _
           New Point(0, Hgt)}


        '---Redraw New Image 1 ---
        gr_out.DrawImage(myImg, myCorners)
        PIC1.Image = bm_out


        '---Redraw image 2---
        bm_out = New Bitmap(wid, Hgt)
        gr_out = Graphics.FromImage(bm_out)
        gr_out.InterpolationMode = InterpolationMode.HighQualityBicubic
        gr_out.CompositingQuality = CompositingQuality.HighQuality
        gr_out.DrawImage(myImg2, myCorners)
        PIC2.Image = bm_out
```

```
        PIC1.Refresh()
        PIC2.Refresh()


    End Sub
    Sub zRestore_Images()
        PIC1.Image = Pic1_Original.Image
        PIC2.Image = Pic2_Original.Image
    End Sub


    Function zMerge_Images(ByVal myVector As Vector) As Image '====23/03/2005====
        '---Init Vars---
        Dim i As Int32

        '-Set Vars-
        Dim myImg = PIC1.Image  '#Image One is the one that moves
        Dim wid = myImg.Width
        Dim Hgt = myImg.Height
        Dim bm_out As New Bitmap(CInt(PIC2.Image.Width), CInt(PIC2.Image.Height))
        Dim gr_out As Graphics = Graphics.FromImage(bm_out)

        '--Define Corners of Image-
        Dim myCorners As Point() = { _
           New Point(0, 0), _
           New Point(wid, 0), _
           New Point(0, Hgt), _
           New Point(wid, Hgt)}



        '--Translate--
        For i = 0 To 3
            myCorners(i).X = myCorners(i).X + myVector.myP.X
            myCorners(i).Y = myCorners(i).Y + myVector.myP.Y
        Next i

        '---Rotate---
        Dim NEWOrigin As PointF
        NEWOrigin.X = (wid / 2) + myVector.myP.X
        NEWOrigin.Y = (Hgt / 2) + myVector.myP.Y

        myCorners = Rotate_XY_Points(myCorners, myVector.myT, NEWOrigin)

        '---Redraw New Image 1 ---
        ReDim Preserve myCorners(2)
        gr_out.DrawImage(myImg, myCorners)

        '---Invert Colours of Image 1---
        bm_out = InVert_IMG(bm_out)

        '---Blend Two Pics----
        Dim Final_BM As Bitmap = New Bitmap(PIC2.Image)
        Dim gr As Graphics = Graphics.FromImage(Final_BM)

        gr.DrawImage(bm_out, 0, 0)

        '----Update Pics----
        Return Final_BM

    End Function
    'Function zTranslate_rotate_Image(ByVal myVector As Vector) As Image '====23/03/2005==
»       ==
    '    '---Init Vars---
    '    Dim i As Int32

    '    '-Set Vars-
    '    Dim myImg = PIC1.Image  '#Image One is the one that moves
    '    Dim wid = myImg.Width
    '    Dim Hgt = myImg.Height

    '    '--Define Corners of Image-
    '    Dim myCorners As Point() = { _
```

```
    '       New Point(0, 0), _
    '       New Point(wid, 0), _
    '       New Point(0, Hgt), _
    '       New Point(wid, Hgt)}



    '    '--Translate--
    '    For i = 0 To 3
    '        myCorners(i).X = myCorners(i).X + myVector.myP.X
    '        myCorners(i).Y = myCorners(i).Y + myVector.myP.Y
    '    Next i

    '    '---Rotate---
    '    myCorners = Rotate_XY_Points(myCorners, myVector.myT)

    '    Dim bm_out As New Bitmap(CInt(PIC1.Image.Width), CInt(PIC1.Image.Height))
    '    Dim gr_out As Graphics = Graphics.FromImage(bm_out)
    '    ReDim Preserve myCorners(2)
    '    gr_out.DrawImage(myImg, myCorners)



    '    '---Draw Pic----
    '    '  Dim Final_BM As Bitmap = New Bitmap(CInt(myImg.Width), CInt(myImg.Height))
    '    '  Dim gr As Graphics = Graphics.FromImage(Final_BM)

    '    '  gr.DrawImage(bm_out, 0, 0)

    '    '----Update Pics----
    '    Return bm_out
    '    'picHidden.Image = bm_out
    'End Function

    Private Sub cmdDeshadow_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles cmdDeshadow.Click
        'Call zReduce_Images(5)
        Dim Shadow As New DeShadow(10, Pic1_Original.Image)
        Dim Shadow2 As New DeShadow(10, Pic2_Original.Image)
        'Call zRestore_Images()
        PIC1.Image = Shadow.deshadow
        PIC2.Image = Shadow2.deshadow
    End Sub

    Private Sub cmdAutoReg_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs)
        Call Comapare_R_to_Vessels()
    End Sub
    Sub Comapare_R_to_Vessels()
        '-init vars-
        Dim arrPic1(), arrPic2() As Point
        Dim arrRs() As Point
        Dim myVessels As CVessels
        Dim myVector As New Vector()
        Dim Factor As Int32 = 5

        '###################GET ARRAYS#######################
        '-Get Vessels on pic1-
        myVessels = New CVessels(Pic1_Original.Image, txtGrp.Text, "")
        arrPic1 = myVessels.Get_Best_Pixels

        '-Get Vessels on pic2-
        myVessels = New CVessels(Pic2_Original.Image, txtGrp.Text, "")
        arrPic2 = myVessels.Get_Best_Pixels

        '=====Get R Values======
        Call zReduce_Images(Factor)
        '-build class-
        Dim myCor = New CNorm_Corralate(PIC1.Image, PIC2.Image)

        '-Calc Translation and rotation needed to match-
        myVector = myCor.Correlate_Mini_match(Factor)

        '-Increase by Factor-
        myVector.myP.X = myVector.myP.X * Factor
```

```
        myVector.myP.Y = myVector.myP.Y * Factor

        '-Unshrink images-
        Call zRestore_Images()

        '-Get R's-
        arrRs = myCor.get_arrofR

        '#########################################################
        '========== translate arrays by pixels found from r=========
        '-show timer form-
        Dim Timer As New timerform()
        Timer.ProgressBar.Maximum = UBound(arrRs) + 1
        Timer.ProgressBar.Step = 1
        Timer.Show()
        Dim Pixel1, pixel2 As Point
        Dim i1, iR, i2 As Int32
        Dim bR As Point
        Dim bDiff, iDiff As Int64
        'bDiff = 999999999
        For iR = 0 To UBound(arrRs)
            iDiff = 0
            Timer.ProgressBar.Value = Timer.ProgressBar.Value + 1 ' update wait bar

            If arrRs(iR).X = 10 And arrRs(iR).Y = -6 Then
                Debug.Write("")
            End If
            For i1 = 0 To UBound(arrPic1)
                '-set new pixels-
                Pixel1.X = arrPic1(i1).X + arrRs(iR).X
                Pixel1.Y = arrPic1(i1).Y + arrRs(iR).Y
                For i2 = 0 To UBound(arrPic2)
                    '-set new pixels-
                    pixel2.X = arrPic2(i2).X
                    pixel2.Y = arrPic2(i2).Y

                    '-calc difference-
                    iDiff = iDiff + Abs(Pixel1.X - pixel2.X)
                    iDiff = iDiff + Abs(Pixel1.Y - pixel2.Y)
                Next i2
            Next i1
            '-Store best matches-
            If bDiff = 0 Then
                bDiff = iDiff
                bR = arrRs(iR)
            ElseIf iDiff < bDiff Then
                bDiff = iDiff
                bR = arrRs(iR)
            End If
        Next iR
        Timer.Hide()
        Timer = Nothing
        Call MsgBox("Best Translation = " & bR.ToString)
    End Sub


    'Private Sub cmdFFT_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    '    Dim myFFT As New FFT(Pic1_Original.Image)
    '    myFFT.Do_FFT(True)
    'End Sub



    Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles Button2.Click
        Dim tmp As New MEDIAN_CUT()
        PIC1.Image = tmp.RemapColourTable(PIC1.Image)
    End Sub

    Private Sub cmd3D_Display_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles cmd3D_Display.Click
        Dim my3DForm As New frm3D()
        Call my3DForm.Display3d(picHidden.Image, PIC2.Image)
        Call my3DForm.frm3D_Load(sender, e)
```

```
    End Sub


    Private Sub cmdSaveAs_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles cmdSaveAs.Click
        Dim image As Drawing.Image = PicDisplay.Image
        SaveBox.Filter = "Image Files (*.jpg,*.tif,*.bmp)|*.jpg;*.tif;*.bmp|All files (*.*
»           )|*.*"
        'OpenBox.InitialDirectory = "j:\guest\arid\data\"
        SaveBox.FileName = lblInfo_A.Text & "-Diff"

        SaveBox.ShowDialog()
        If SaveBox.FileName <> "" Then

            Call image.Save(SaveBox.FileName, Drawing.Imaging.ImageFormat.Tiff)
        End If
    End Sub

    Private Sub SaveFileDialog1_FileOk(ByVal sender As System.Object, ByVal e As System.
»       ComponentModel.CancelEventArgs)

    End Sub

    Private Sub OpenBox_FileOk(ByVal sender As System.Object, ByVal e As System.
»       ComponentModel.CancelEventArgs) Handles OpenBox.FileOk

    End Sub

    Private Sub picHidden_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles picHidden.Click

    End Sub

    'Private Sub cmdBlue_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
»        Handles cmdBlue.Click
    '    Dim ColorFind As New CBlue(PIC1.Image, 30) ' For testing
    '    PIC1.Image = ColorFind.Return_imagemap



    'End Sub

    Private Sub cmdBatch_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
»        Handles cmdBatch.Click
        Dim myCFSO As New CFSO()
        Dim FilesToDo, ipair As Int32
        Dim Pics(1) As Image
        Dim myFile As String

        If (DialogResult.OK = FolderBrowser1.ShowDialog()) Then

            myCFSO.Set_Dir(FolderBrowser1.DirectoryPath)
            '---Get Image Pair---
            FilesToDo = myCFSO.Filecount

            For ipair = 0 To FilesToDo Step 2
                Pics = myCFSO.Get_Image_Pair(ipair)
                myFile = myCFSO.Getname(ipair)
                '---load image pair
                PIC1.Image = Pics(0)
                PIC2.Image = Pics(1)
                Call FRMArid_Load(cmdLoad_A, e)
                Call FRMArid_Load(cmdLoad_B, e)

                '--Run Arid--
                Call cmdArid_Auto_Click(sender, e)
                '--Save Output--
                PicDisplay.Refresh()
                Dim tmp As String = InputBox("oh noes", MsgBoxStyle.Critical, "toot")
                Dim myimage As New Bitmap(PicDisplay.Image)
```

```
                Call myimage.Save("c:\ARID\" & myFile & "_DIFF.tiff", Drawing.Imaging.Imag
»                   eFormat.Tiff)
                'image = Nothing
                ' Debug.Write(PicDisplay)

                Erase Pics
                Call cmdClear_Click_1(sender, e)
            Next

        End If
    End Sub
End Class
```

ARID **Form1.resx**

```
Public Class frm3D
    Inherits System.Windows.Forms.Form
    Dim rDimensions As Rectangle

    Function resize_Img(ByVal img As Image) As Image
        Dim myG As Graphics
        Dim hgt, wid As Int32
        Dim sX, sY As Single


        Dim NEWimg As Image
        NEWimg = img
        NEWimg = New Bitmap(NEWimg, New Size(rDimensions.Width, rDimensions.Height))
        resize_Img = NEWimg

    End Function

    Function Get_Interlaced(ByRef imgL As Image, ByRef imgR As Image) As Image
        Dim myG As Graphics
        Dim iX, iY As Int32
        Dim bOut As New Bitmap(1280, 1024)
        Dim fr_rect, to_rect As Rectangle


        myG = Graphics.FromImage(bOut)

        For iX = 0 To bOut.Width
            fr_rect = New Rectangle(iX, 0, 1, 1280)
            to_rect = New Rectangle(iX, 0, 1, 1280)
            If IsOdd(iX) Then
                myG.DrawImage(imgL, to_rect, fr_rect, GraphicsUnit.Pixel)
            Else
                myG.DrawImage(imgR, to_rect, fr_rect, GraphicsUnit.Pixel)
            End If
        Next

        Get_Interlaced = bOut

    End Function

    Private Function IsOdd(ByVal lngNumber As Long) As Boolean
        IsOdd = IIf((lngNumber Mod 2) = 0, False, True)
    End Function

    Public Sub Display3d(ByVal picleft As Image, ByVal picRight As Image)
        Dim iLeft, iRight As Image
        iLeft = resize_Img(picleft)
        iRight = resize_Img(picRight)
        Debug.WriteLine(iLeft.Height & " " & iRight.Height)
        picCombined.Image = Get_Interlaced(iLeft, iRight)
        picCombined.Top = 0
        picCombined.Left = 0


    End Sub

    Sub frm3D_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.
»       Load


        If Screen.AllScreens.Length >= 2 Then

            Me.Location = New Point(Screen.AllScreens(1).Bounds.X, Screen.AllScreens(1).
»               Bounds.Y)

            Me.Height = rDimensions.Height
            Me.Width = rDimensions.Width
            picCombined.Height = rDimensions.Height
            picCombined.Width = rDimensions.Width
            Me.Visible = True
```

```
        Else
            Call MsgBox("No secondary screen detected")
            Exit Sub
        End If

    End Sub


End Class
```

ARID **FRMFlip.resx**

```
Public Class FRMFlip
    Inherits System.Windows.Forms.Form


    Private Sub Timer_Flip_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles Timer_Flip.Tick
        Pic2.Visible = Not Pic2.Visible
    End Sub

    Private Sub TrackBar_Scroll(ByVal sender As System.Object, ByVal e As System.EventArgs
»       ) Handles TrackBar.Scroll
        Timer_Flip.Interval = (TrackBar.Value ^ 2)
        Me.Text = Timer_Flip.Interval
    End Sub


    Private Sub TrackBar_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms
»       .MouseEventArgs) Handles TrackBar.MouseMove

    End Sub

End Class
```

ARID HSLRGB.vb

```
Imports System.Drawing
Public Class HLSRGB
    Private m_red As Byte = 0
    Private m_green As Byte = 0
    Private m_blue As Byte = 0

    Private m_hue As Double = 0
    Private m_luminance As Double = 0
    Private m_saturation As Double = 0

    Public Property Red() As Byte
        Get
            Return m_red
        End Get
        Set(ByVal Value As Byte)
            m_red = Value
            ToHLS()
        End Set
    End Property

    Public Property Green() As Byte
        Get
            Return m_green
        End Get
        Set(ByVal Value As Byte)
            m_green = Value
            ToHLS()
        End Set
    End Property

    Public Property Blue() As Byte
        Get
            Return m_blue
        End Get
        Set(ByVal Value As Byte)
            m_blue = Value
            ToHLS()
        End Set
    End Property

    Public Property Luminance() As Double
        Get
            Return m_luminance
        End Get
        Set(ByVal Value As Double)
            If ((Value < 0.0F) Or (Value > 1.0F)) Then
                Throw New ArgumentOutOfRangeException("Luminance", "Luminance must be betw
»                   een 0.0 and 1.0")
            End If
            m_luminance = Value
            ToRGB()
        End Set
    End Property

    Public Property Hue() As Double
        Get
            Return m_hue
        End Get
        Set(ByVal Value As Double)
            If ((Value < 0.0F) Or (Value > 360.0F)) Then
                Throw New ArgumentOutOfRangeException("Hue", "Hue must be between 0.0 and
»                   360.0")
            End If
            m_hue = Value
            ToRGB()
        End Set
    End Property

    Public Property Saturation() As Double
        Get
```

```
            Return m_saturation
        End Get
        Set(ByVal Value As Double)
            If ((Value < 0.0F) Or (Value > 1.0F)) Then
                Throw New ArgumentOutOfRangeException("Saturation", "Saturation must be be
»                   tween 0.0 and 1.0")
            End If
            m_saturation = Value
            ToRGB()
        End Set
    End Property

    Public Property Color() As Color
        Get
            Dim c As Color = Color.FromArgb(m_red, m_green, m_blue)
            Return c
        End Get
        Set(ByVal Value As Color)
            m_red = Value.R
            m_green = Value.G
            m_blue = Value.B
            ToHLS()
        End Set
    End Property

    Public Sub LightenColor(ByVal lightenBy As Double)
        m_luminance *= (1.0F + lightenBy)
        If (m_luminance > 1.0F) Then
            m_luminance = 1.0F
        End If
        ToRGB()
    End Sub

    Public Sub DarkenColor(ByVal darkenBy As Double)
        m_luminance *= darkenBy
        ToRGB()
    End Sub


    Public Sub New(ByVal c As Color)
        m_red = c.R
        m_green = c.G
        m_blue = c.B
        ToHLS()
    End Sub

    Public Sub New(ByVal hue As Double, ByVal luminance As Double, ByVal saturation As
»       Double)
        If ((saturation < 0.0F) Or (saturation > 1.0F)) Then
            Throw New ArgumentOutOfRangeException("Saturation", "Saturation must be betwee
»               n 0.0 and 1.0")
        End If
        If ((hue < 0.0F) Or (hue > 360.0F)) Then
            Throw New ArgumentOutOfRangeException("Hue", "Hue must be between 0.0 and 360.
»               0")
        End If
        If ((luminance < 0.0F) Or (luminance > 1.0F)) Then
            Throw New ArgumentOutOfRangeException("Luminance", "Luminance must be between
»               0.0 and 1.0")
        End If
        m_hue = hue
        m_luminance = luminance
        m_saturation = saturation
        ToRGB()
    End Sub

    Public Sub New(ByVal red As Byte, ByVal green As Byte, ByVal blue As Byte)
        m_red = red
        m_green = green
        m_blue = blue
```

```
    End Sub

    Public Sub New(ByVal hlsrgb As HLSRGB)
        m_red = hlsrgb.Red
        m_blue = hlsrgb.Blue
        m_green = hlsrgb.Green
        m_luminance = hlsrgb.Luminance
        m_hue = hlsrgb.Hue
        m_saturation = hlsrgb.Saturation
    End Sub

    Public Sub New()

    End Sub

    Sub ToHLS()
        Dim minval As Byte = Math.Min(m_red, Math.Min(m_green, m_blue))
        Dim maxval As Byte = Math.Max(m_red, Math.Max(m_green, m_blue))

        Dim mdiff As Double = (maxval * 1.0 - minval * 1.0)
        Dim msum As Double = (maxval * 1.0 + minval * 1.0)

        m_luminance = msum / 510.0F

        If (maxval = minval) Then
            m_saturation = 0.0F
            m_hue = 0.0F
        Else
            Dim rnorm As Double = (maxval - m_red) / mdiff
            Dim gnorm As Double = (maxval - m_green) / mdiff
            Dim bnorm As Double = (maxval - m_blue) / mdiff

            If (m_luminance <= 0.5F) Then
                m_saturation = (mdiff / msum)
            Else
                m_saturation = (mdiff / (510.0F - msum))
            End If

            If (m_red = maxval) Then
                m_hue = 60.0F * (6.0F + bnorm - gnorm)
            End If
            If (m_green = maxval) Then
                m_hue = 60.0F * (2.0F + rnorm - bnorm)
            End If
            If (m_blue = maxval) Then
                m_hue = 60.0F * (4.0F + gnorm - rnorm)
            End If
            If (m_hue > 360.0F) Then
                m_hue = m_hue - 360.0F
            End If
        End If
    End Sub

    Private Sub ToRGB()
        If (m_saturation = 0.0) Then
            m_red = CByte(m_luminance * 255.0F)
            m_green = m_red
            m_blue = m_red
        Else
            Dim rm1 As Double
            Dim rm2 As Double

            If (m_luminance <= 0.5F) Then
                rm2 = m_luminance + m_luminance * m_saturation
            Else
                rm2 = m_luminance + m_saturation - m_luminance * m_saturation
            End If
            rm1 = 2.0F * m_luminance - rm2
            m_red = ToRGB1(rm1, rm2, m_hue + 120.0F)
            m_green = ToRGB1(rm1, rm2, m_hue)
            m_blue = ToRGB1(rm1, rm2, m_hue - 120.0F)
        End If
    End Sub
```

```
    Private Function ToRGB1(ByVal rm1 As Double, ByVal rm2 As Double, ByVal rh As Double)
»       As Byte
        If (rh > 360.0F) Then
            rh -= 360.0F
        ElseIf (rh < 0.0F) Then
            rh += 360.0F
        End If

        If (rh < 60.0F) Then
            rm1 = rm1 + (rm2 - rm1) * rh / 60.0F
        ElseIf (rh < 180.0F) Then
            rm1 = rm2
        ElseIf (rh < 240.0F) Then
            rm1 = rm1 + (rm2 - rm1) * (240.0F - rh) / 60.0F
        End If

        ToRGB1 = CByte(rm1 * 255)
    End Function


End Class
```

ARID **Median_Cut.vb**

```
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Runtime.InteropServices

Public Class MEDIAN_CUT
    'Inherits System.Windows.Forms.Form

    Function RemapColourTable(ByVal img As Image) As Image
        Dim pic As Image = img
        ' Dim image_Out As Image
        RemapColourTable = SaveGIFWithNewColorTable(pic, 256, False)
    End Function

    ' Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) H
»       andles MyBase.Load

    '   Dim pic As Image = Image.FromFile("c:\test.jpg")

    '     SaveGIFWithNewColorTable(pic, "c:\test.gif", 256, False)

    '  End Sub

    Class Win32API

        <DllImport("KERNEL32.DLL", EntryPoint:="RtlMoveMemory", _
           SetLastError:=True, CharSet:=CharSet.Auto, _
           ExactSpelling:=True, _
           CallingConvention:=CallingConvention.StdCall)> _
        Public Shared Sub CopyArrayTo(<[In]()>, MarshalAs(UnmanagedType.I4)> ByVal hpvDest
»           As Int32, <[In]()>, Out()> ByVal hpvSource() As Byte, ByVal cbCopy As Integer)
            ' Leave function empty - DLLImport attribute forwards calls to CopyArrayTo to
            ' RtlMoveMemory in KERNEL32.DLL.
        End Sub


    End Class

    Private Function GetColorPalette(ByVal nColors As Integer) As ColorPalette
        ' Assume monochrome image.
        Dim bitscolordepth As PixelFormat = PixelFormat.Format1bppIndexed
        Dim palette As ColorPalette 'The Palette we are stealing
        Dim bitmap As Bitmap        'The source of the stolen palette

        ' Determine number of colors.
        If nColors > 2 Then
            bitscolordepth = PixelFormat.Format4bppIndexed
        End If
        If (nColors > 16) Then
            bitscolordepth = PixelFormat.Format8bppIndexed
        End If

        ' Make a new Bitmap object to get its Palette.
        bitmap = New Bitmap(1, 1, bitscolordepth)

        palette = bitmap.Palette    ' Grab the palette

        bitmap.Dispose()            ' cleanup the source Bitmap

        Return palette              ' Send the palette back
    End Function


    Private Function SaveGIFWithNewColorTable(ByVal image As Image, ByVal nColors As
»       Integer, ByVal fTransparent As Boolean) As Image

        ' GIF codec supports 256 colors maximum, monochrome minimum.
        If (nColors > 256) Then
            nColors = 256
        End If
        If (nColors < 2) Then
            nColors = 2
```

```
        End If


        ' Make a new 8-BPP indexed bitmap that is the same size as the source image.
        Dim Width As Integer = image.Width
        Dim Height As Integer = image.Height

        ' Always use PixelFormat8BppIndexed because that is the color
        ' table based interface to the GIF codec.
        Dim bitmap As Bitmap = New Bitmap(Width, Height, PixelFormat.Format8bppIndexed)

        ' Create a color palette big enough to hold the colors you want.
        Dim pal As ColorPalette = GetColorPalette(nColors)

        ' Initialize a new color table with entries that are determined
        ' by some optimal palette-finding algorithm; for demonstration
        ' purposes, use a grayscale.
        Dim i As Integer
        For i = 0 To nColors - 1
            Dim Alpha As Integer = 255             ' Colors are opaque
            Dim Intensity As Double = CDbl(i) * 255 / (nColors - 1) ' even distribution

            ' The GIF encoder makes the first entry in the palette
            ' with a ZERO alpha the transparent color in the GIF.
            ' Pick the first one arbitrarily, for demonstration purposes.

            If (i = 0 And fTransparent) Then    ' Make this color index...
                Alpha = 0                       ' Transparent
            End If

            ' Create a gray scale for demonstration purposes.
            ' Otherwise, use your favorite color reduction algorithm
            ' and an optimum palette for that algorithm generated here.
            ' For example, a color histogram, or a median cut palette.
            pal.Entries(i) = Color.FromArgb(Alpha, Intensity, Intensity, Intensity)

        Next i

        ' Set the palette into the new Bitmap object.
        bitmap.Palette = pal


        ' Use GetPixel below to pull out the color data of
        ' image because GetPixel isn't defined on an Image; make a copy
        ' in a Bitmap instead. Next, make a new Bitmap that is the same
        ' size as the image that you want to export. Or, try to interpret
        ' the native pixel format of the image by using a LockBits
        ' call. Use PixelFormat32BppARGB so you can wrap a graphics
        ' around it.
        Dim BmpCopy As Bitmap = New Bitmap(Width, Height, PixelFormat.Format32bppArgb)

        Dim g As Graphics
        g = Graphics.FromImage(BmpCopy)

        g.PageUnit = GraphicsUnit.Pixel

        ' Transfer the Image to the Bitmap.
        g.DrawImage(image, 0, 0, Width, Height)

        ' Force g to release its resources, namely BmpCopy.
        g.Dispose()

        ' Lock a rectangular portion of the bitmap for writing.
        Dim bitmapData As BitmapData
        Dim rect As Rectangle = New Rectangle(0, 0, Width, Height)

        bitmapData = bitmap.LockBits(rect, ImageLockMode.WriteOnly, PixelFormat.
» Format8bppIndexed)

        ' Write to a temporary buffer, and then copy to the buffer that
        ' LockBits provides. Copy the pixels from the source image in this
        ' loop. Because you want an index, convert RGB to the appropriate
        ' palette index here.
        Dim pixels As IntPtr = bitmapData.Scan0
        Dim bits As Byte()      ' the working buffer
        ' Get the pointer to the image bits.
```

```
        Dim pBits As Int32

        If (bitmapData.Stride > 0) Then
            pBits = pixels.ToInt32()
        Else
            ' If the Stide is negative, Scan0 points to the last
            ' scanline in the buffer. To normalize the loop, obtain
            ' a pointer to the front of the buffer that is located
            ' (Height-1) scanlines previous.
            pBits = pixels.ToInt32() + bitmapData.Stride * (Height - 1)
        End If

        Dim stride As Integer = Math.Abs(bitmapData.Stride)
        ReDim bits(Height * stride) ' Allocate the working buffer.

        Dim row As Integer
        Dim col As Integer

        For row = 0 To Height - 1
            For col = 0 To Width - 1
                ' Map palette indices for a gray scale.
                ' Put your favorite color reduction algorithm here.
                ' If you use some other technique to color convert.
                Dim pixel As Color      ' The source pixel.

                ' The destination pixel.
                Dim i8BppPixel As Integer = row * stride + col

                pixel = BmpCopy.GetPixel(col, row)

                ' Use luminance/chrominance conversion to get grayscale.
                ' Basically, turn the image into black and white TV.
                ' Do not calculate Cr or Cb because you
                ' discard the color anyway.
                ' Y = Red * 0.299 + Green * 0.587 + Blue * 0.114

                ' This expression should be integer math for performance;
                ' however, because GetPixel above is the slowest part of
                ' this loop, the expression is left as floating point
                ' for clarity.
                Dim luminance As Double = (pixel.R * 0.299) + _
                                          (pixel.G * 0.587) + _
                                          (pixel.B * 0.114)

                ' Gray scale is an intensity map from black to white.
                ' Compute the index to the grayscale entry that
                ' approximates the luminance, and then round the index.
                ' Also, constrain the index choices by the number of
                ' colors to do, and then set that pixel's index to the byte
                ' value.
                Dim colorIndex As Double = Math.Round((luminance * (nColors - 1) / 255))

                bits(i8BppPixel) = CByte(colorIndex)

                ' /* end loop for col */
            Next col
            ' /* end loop for row */
        Next row

        ' Put the image bits definition into the bitmap.
        Win32API.CopyArrayTo(pBits, bits, Height * stride)

        ' To commit the changes, unlock the portion of the bitmap.
        bitmap.UnlockBits(bitmapData)
        'SaveGIFWithNewColorTable = bitmap.Clone()
        SaveGIFWithNewColorTable = image
        '  bitmap.Save(filename, ImageFormat.Gif)

        ' Bitmap goes out of scope here and is also marked for
        ' garbage collection.
        ' Pal is referenced by bitmap and goes away.
        ' BmpCopy goes out of scope here and is marked for garbage
        ' collection. Force it, because it is probably quite large.
        ' The same applies for bitmap.
        BmpCopy.Dispose()
        bitmap.Dispose()
```

```
    End Function

End Class
```

ARID **Timerform.resx**

```
Public Class timerform
    Inherits System.Windows.Forms.Form
    Public txttitle As String

    Private Sub timerform_Load(ByVal sender As Object, ByVal e As System.EventArgs)
»       Handles MyBase.Load
        '  lbltitle.text = txttitle
    End Sub

    Private Sub ProgressBar_Click(ByVal sender As System.Object, ByVal e As System.
»       EventArgs) Handles ProgressBar.Click

    End Sub
End Class
```