ÍNDICE PROGRAMA

CÓDIGO PROGRAMA PRINCIPAL

```
Imports System.Threading

Public Class FormOsciloscopio

#Region "Variables"

    Public VDIVA As Double
    Public VDIVB As Double
    Public TDIV As Double
    Public TRIGGER As Double
    Public BTRIGGER As Byte
    Public fichero As String

    Public ObjDatosSeñal As DatosSeñal
    Public ObjGrafica As Grafica
    Public ObjCapturaDat As CapturaDatos
    Public ObjArchivo As ArchivosTxt
    Public dibujoEjes As Graphics
    Public dibujoA As Graphics
    Public dibujoB As Graphics
    Public dibujoCP As Graphics
    Public btmp As Bitmap
    Public btmpCP As Bitmap
    Public PSize As Size
    Public ComandFFT As FormFFT

    Public CoordX As Integer
    Public CoordYA As Integer
    Public CoordYB As Integer

    Public threadREPRESENTAR As Thread
    Public threadTRIGGERAUTO As Thread
    Public threadTRIGGERSUB As Thread
    Public threadTRIGGERBAJ As Thread
    Public threadObtDatos As Thread
    Public threadTrigger As Thread
    Public threadFFT As Thread


#End Region

#Region "Constructor"

    Public Sub New()

        ' Llamada necesaria para el Diseñador de Windows Forms.
        InitializeComponent()

        ' Agregue cualquier inicialización después de la llamada a
InitializeComponent().
        ObjCapturaDat = New CapturaDatos
        ObjDatosSeñal = New DatosSeñal
        ObjGrafica = New Grafica
        ObjArchivo = New ArchivosTxt
```

```
        VDIVA = 1
        VDIVB = 1
        TDIV = 0.02
        TRIGGER = 0

        CoordX = TBEjeX.Value
        CoordYA = TBCanalA.Value
        CoordYB = TBCanalB.Value

        LblVDivA.Text = "CH A= " & VDIVA & "V/DIV"
        LblVDivB.Text = "CH B= " & VDIVB & "V/DIV"
        LblTimeDiv.Text = "Time= " & TDIV & "s/DIV"

        btmp = New Bitmap(PBGrafico.Width, PBGrafico.Height)
        btmpCP = New Bitmap(PBGrafico.Width, PBGrafico.Height)

        dibujoA = Graphics.FromImage(btmp)
        dibujoB = Graphics.FromImage(btmp)
        dibujoCP = Graphics.FromImage(btmpCP)
        dibujoEjes = Graphics.FromImage(btmp)

        ComandFFT = New FormFFT()

        threadTRIGGERAUTO = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturaAuto))
        threadTRIGGERSUB = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarSUB))
        threadTRIGGERBAJ = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarBAJ))
        threadREPRESENTAR = New Thread(New
ParameterizedThreadStart(AddressOf Representar))
        threadObtDatos = New Thread(New ParameterizedThreadStart(AddressOf
ObtenerDatos))
        threadFFT = New Thread(New ParameterizedThreadStart(AddressOf
ComandFFT.ObtDatosFFT))
        threadTrigger = New Thread(New ParameterizedThreadStart(AddressOf
ObjCapturaDat.ObtenerBTRG))

    End Sub

#End Region

#Region "Funciones modo Canal"

    Private Sub ChBHold_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ChBHold.CheckedChanged
        If ChBHold.CheckState = CheckState.Unchecked Then
            If RBTRGAuto.Checked = True Or RBTRGDef.Checked = True Or
ChBDatosArch.CheckState = CheckState.Checked Then
                If RBCanalA.Checked = True Or RBCanalB.Checked = True Or
RBDual.Checked = True Then
                    If threadREPRESENTAR.ThreadState = ThreadState.Running
Then
                        threadREPRESENTAR.Abort()
                    End If
                    If threadREPRESENTAR.ThreadState = ThreadState.Stopped
Or threadREPRESENTAR.ThreadState = ThreadState.Aborted Then
                        threadREPRESENTAR = New Thread(New
ParameterizedThreadStart(AddressOf Representar))
                        threadREPRESENTAR.Start()
                    End If
                    If threadREPRESENTAR.ThreadState =
ThreadState.Unstarted Then
```

```
                    threadREPRESENTAR.Start()
                End If
            End If
        End If
    End If
End Sub

    Private Sub ChBGNDA_CheckStateChanged(ByVal sender As Object, ByVal e
As System.EventArgs) Handles ChBGNDA.CheckStateChanged
        If ChBGNDA.CheckState = CheckState.Unchecked Then
            If RBTRGAuto.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    threadTRIGGERAUTO.Abort()
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Stopped Or
threadTRIGGERAUTO.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERAUTO = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturaAuto))
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Stopped Or
threadTRIGGERSUB.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERSUB = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarSUB))
```

```
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGBJD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERBAJ.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Stopped Or
threadTRIGGERBAJ.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERBAJ = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarBAJ))
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
            End If
            If ChBDatosArch.CheckState = CheckState.Checked Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    threadObtDatos.Abort()
                End If
                If threadObtDatos.ThreadState = ThreadState.Stopped Or
threadObtDatos.ThreadState = ThreadState.Aborted Then
                    threadObtDatos = New Thread(New
ParameterizedThreadStart(AddressOf ObtenerDatos))
                    threadObtDatos.Start(fichero)
                End If
                If threadObtDatos.ThreadState = ThreadState.Unstarted Then
```

```
                    threadObtDatos.Start(fichero)
                End If
            End If
        Else
            If ChBGNDA.CheckState = CheckState.Checked Then
                If threadTRIGGERAUTO.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Abort()
                End If
                If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Abort()
                End If
                If threadObtDatos.ThreadState <> ThreadState.Unstarted Then
                    threadObtDatos.Abort()
                End If
                For i As Integer = 0 To ObjDatosSeñal.Numelem / 2
                    ObjDatosSeñal.datosX1(i) = 0
                Next
            End If
        End If
    End Sub

    Private Sub ChBGNDB_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ChBGNDB.CheckedChanged
        If ChBGNDB.CheckState = CheckState.Unchecked Then
            If RBTRGAuto.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    threadTRIGGERAUTO.Abort()
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Stopped Or
threadTRIGGERAUTO.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERAUTO = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturaAuto))
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
```

```
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Stopped Or
threadTRIGGERSUB.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERSUB = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarSUB))
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGBJD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERBAJ.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Stopped Or
threadTRIGGERBAJ.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERBAJ = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarBAJ))
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
            End If
            If ChBDatosArch.CheckState = CheckState.Checked Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
```

```
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    threadObtDatos.Abort()
                End If
                If threadObtDatos.ThreadState = ThreadState.Stopped Or
threadObtDatos.ThreadState = ThreadState.Aborted Then
                    threadObtDatos = New Thread(New
ParameterizedThreadStart(AddressOf ObtenerDatos))
                    threadObtDatos.Start(fichero)
                End If
                If threadObtDatos.ThreadState = ThreadState.Unstarted Then
                    threadObtDatos.Start(fichero)
                End If
            End If
        Else
            If ChBGNDB.CheckState = CheckState.Checked Then
                If threadTRIGGERAUTO.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Abort()
                End If
                If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Abort()
                End If
                If threadObtDatos.ThreadState <> ThreadState.Unstarted Then
                    threadObtDatos.Abort()
                End If
                For i As Integer = 0 To ObjDatosSeñal.Numelem / 2
                    ObjDatosSeñal.datosX2(i) = 0
                Next
            End If
        End If
    End Sub

    Private Sub TBCanalA_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TBCanalA.Scroll
        CoordYA = TBCanalA.Value
    End Sub

    Private Sub TBCanalB_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TBCanalB.Scroll
        CoordYB = TBCanalB.Value
    End Sub

    Private Sub TBEjeX_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TBEjeX.Scroll
        CoordX = TBEjeX.Value
    End Sub
```

```
    Private Sub ChBMedidaDC_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ChBMedidaDC.CheckedChanged
        If ChBMedidaDC.CheckState = CheckState.Checked Then
            VDIVA = VDIVA / 2
            VDIVB = VDIVB / 2
        Else
            VDIVA = VDIVA * 2
            VDIVB = VDIVB * 2
        End If
    End Sub

#End Region

#Region "Funciones botones escala de Voltios Canal A"

    Private Sub RB5A_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB5A.CheckedChanged
        If RB5A.Checked = True Then
            VDIVA = 5
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB2A_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB2A.CheckedChanged
        If RB2A.Checked = True Then
            VDIVA = 2
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB1A_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB1A.CheckedChanged
        If RB1A.Checked = True Then
            VDIVA = 1
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB05A_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB05A.CheckedChanged
        If RB05A.Checked = True Then
            VDIVA = 0.5
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB02A_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB02A.CheckedChanged
        If RB02A.Checked = True Then
            VDIVA = 0.2
```

```
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB01A_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB01A.CheckedChanged
        If RB01A.Checked = True Then
            VDIVA = 0.1
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB005A_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB005A.CheckedChanged
        If RB005A.Checked = True Then
            VDIVA = 0.05
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB002A_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB002A.CheckedChanged
        If RB002A.Checked = True Then
            VDIVA = 0.02
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

    Private Sub RB001A_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB001A.CheckedChanged
        If RB001A.Checked = True Then
            VDIVA = 0.01
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVA = VDIVA / 2
            End If
        End If
    End Sub

#End Region

#Region "Funciones botones escala de Voltios Canal B"

    Private Sub RB5B_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB5B.CheckedChanged
        If RB5B.Checked = True Then
            VDIVB = 5
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
```

```
        End If
    End Sub

    Private Sub RB2B_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB2B.CheckedChanged
        If RB2B.Checked = True Then
            VDIVB = 2
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB1B_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB1B.CheckedChanged
        If RB1B.Checked = True Then
            VDIVB = 1
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB05B_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB05B.CheckedChanged
        If RB05B.Checked = True Then
            VDIVB = 0.5
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB02B_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB02B.CheckedChanged
        If RB02B.Checked = True Then
            VDIVB = 0.2
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB01B_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB01B.CheckedChanged
        If RB01B.Checked = True Then
            VDIVB = 0.1
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB005B_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB005B.CheckedChanged
        If RB005B.Checked = True Then
            VDIVB = 0.05
            RefrescarEscalas()
```

```
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB002B_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB002B.CheckedChanged
        If RB002B.Checked = True Then
            VDIVB = 0.02
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

    Private Sub RB001B_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB001B.CheckedChanged
        If RB001B.Checked = True Then
            VDIVB = 0.01
            RefrescarEscalas()
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                VDIVB = VDIVB / 2
            End If
        End If
    End Sub

#End Region

#Region "Funciones escala de Tiempo"

    Private Sub RB1T_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB1T.CheckedChanged
        If RB1T.Checked = True Then
            TDIV = 1
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB05T_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB05T.CheckedChanged
        If RB05T.Checked = True Then
            TDIV = 0.5
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB02T_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB02T.CheckedChanged
        If RB02T.Checked = True Then
            TDIV = 0.2
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB01T_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB01T.CheckedChanged
        If RB01T.Checked = True Then
            TDIV = 0.1
            RefrescarEscalas()
        End If
    End Sub
```

```
    Private Sub RB50mT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB50mT.CheckedChanged
        If RB50mT.Checked = True Then
            TDIV = 0.05
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB20mT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB20mT.CheckedChanged
        If RB20mT.Checked = True Then
            TDIV = 0.02
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB10mT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB10mT.CheckedChanged
        If RB10mT.Checked = True Then
            TDIV = 0.01
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB5mT_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB5mT.CheckedChanged
        If RB5mT.Checked = True Then
            TDIV = 0.005
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB2mT_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB2mT.CheckedChanged
        If RB2mT.Checked = True Then
            TDIV = 0.002
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB1mT_CheckedChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles RB1mT.CheckedChanged
        If RB1mT.Checked = True Then
            TDIV = 0.001
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB500uT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB500uT.CheckedChanged
        If RB500uT.Checked = True Then
            TDIV = 0.0005
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB200uT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB200uT.CheckedChanged
        If RB200uT.Checked = True Then
            TDIV = 0.0002
            RefrescarEscalas()
        End If
```

```
    End Sub

    Private Sub RB100uT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB100uT.CheckedChanged
        If RB100uT.Checked = True Then
            TDIV = 0.0001
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB50uT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB50uT.CheckedChanged
        If RB50uT.Checked = True Then
            TDIV = 0.00005
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB20uT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB20uT.CheckedChanged
        If RB20uT.Checked = True Then
            TDIV = 0.00002
            RefrescarEscalas()
        End If
    End Sub

    Private Sub RB10uT_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles RB10uT.CheckedChanged
        If RB10uT.Checked = True Then
            TDIV = 0.00001
            RefrescarEscalas()
        End If
    End Sub

#End Region

#Region "Funciones botones"

    Private Sub BtnNuevaCapt_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnNuevaCapt.Click
        ChBDatosArch.CheckState = CheckState.Unchecked
        If RBTRGAuto.Checked = False And RBTRGDef.Checked = False Then
            MsgBox("Por Favor, seleecione el modo de disparo o TRIGGER.")
        Else
            If RBTRGAuto.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    threadTRIGGERAUTO.Abort()
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    If threadFFT.ThreadState <> ThreadState.Unstarted Then
```

```
                        threadFFT.Abort()
                    End If
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Stopped Or
threadTRIGGERAUTO.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERAUTO = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturaAuto))
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Stopped Or
threadTRIGGERSUB.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERSUB = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarSUB))
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERBAJ.Abort()
                End If
```

```
                If threadTRIGGERBAJ.ThreadState = ThreadState.Stopped Or
threadTRIGGERBAJ.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERBAJ = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarBAJ))
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
            End If
            If threadREPRESENTAR.ThreadState <> ThreadState.Running Then
                If threadREPRESENTAR.ThreadState = ThreadState.Running Then
                    threadREPRESENTAR.Abort()
                End If
                If threadREPRESENTAR.ThreadState = ThreadState.Stopped Or
threadREPRESENTAR.ThreadState = ThreadState.Aborted Then
                    threadREPRESENTAR = New Thread(New
ParameterizedThreadStart(AddressOf Representar))
                    threadREPRESENTAR.Start()
                End If
                If threadREPRESENTAR.ThreadState = ThreadState.Unstarted
Then
                    threadREPRESENTAR.Start()
                End If
            End If
        End If
    End Sub

    Private Sub ButValMedio_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButValMedio.Click
        Dim aux1, aux2 As Double
        If RBCanalA.Checked = True Then
            aux1 = ObjDatosSeñal.ValorMedioCanA
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                For i As Integer = 0 To ObjDatosSeñal.Numelem - 1
                    ObjDatosSeñal.datosX1(i) =
Math.Abs(ObjDatosSeñal.datosX1(i)) * 2
                Next
                aux1 = ObjDatosSeñal.ValorMedioCanA
            End If
            MsgBox(aux1, MsgBoxStyle.Information, "V. Medio CanalA")
        ElseIf RBCanalB.Checked = True Then
            aux2 = ObjDatosSeñal.ValorMedioCanB
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                For i As Integer = 0 To ObjDatosSeñal.Numelem - 1
                    ObjDatosSeñal.datosX2(i) =
Math.Abs(ObjDatosSeñal.datosX2(i)) * 2
                Next
                aux2 = ObjDatosSeñal.ValorMedioCanB
            End If
            MsgBox(aux2, MsgBoxStyle.Information, "V. Medio CanalB")
        Else
            MsgBox("Por favor, seleccione un único canal.")
        End If
    End Sub

    Private Sub ButValeficaz_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ButValeficaz.Click
        Dim aux1, aux2 As Double
        If RBCanalA.Checked = True Then
            aux1 = ObjDatosSeñal.ValorEficazCanA
            If ChBMedidaDC.CheckState = CheckState.Checked Then
```

```
                For i As Integer = 0 To ObjDatosSeñal.Numelem - 1
                    ObjDatosSeñal.datosX1(i) =
Math.Abs(ObjDatosSeñal.datosX1(i)) * 2
                Next
                aux1 = ObjDatosSeñal.ValorEficazCanA
            End If
            MsgBox(aux1, MsgBoxStyle.Information, "V. Eficaz CanalA")
        ElseIf RBCanalB.Checked = True Then
            aux2 = ObjDatosSeñal.ValorEficazCanB
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                For i As Integer = 0 To ObjDatosSeñal.Numelem - 1
                    ObjDatosSeñal.datosX2(i) =
Math.Abs(ObjDatosSeñal.datosX2(i)) * 2
                Next
                aux2 = ObjDatosSeñal.ValorEficazCanB
            End If
            MsgBox(aux2, MsgBoxStyle.Information, "V. Eficaz CanalB")
        Else
            MsgBox("Por favor, seleccione un único canal.")
        End If
    End Sub

    Private Sub BTNCopiarPantalla_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BTNCopiarPantalla.Click
        ChBHold.CheckState = CheckState.Checked
        ' Copiar la imagen del PictureBox1 en el portapapeles
        btmpCP = New Bitmap(PBGrafico.Width, PBGrafico.Height)
        dibujoCP = Graphics.FromImage(btmpCP)
        dibujoCP.CopyFromScreen(Me.Location.X + 14, Me.Location.Y + 52, 0,
0, PBGrafico.Size)
        Clipboard.SetDataObject(btmpCP, True)
    End Sub

    Private Sub BTNGuardarDatos_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BTNGuardarDatos.Click
        ChBHold.CheckState = CheckState.Checked
        Dim SFDialog As New SaveFileDialog
        SFDialog.InitialDirectory = "C:\"
        SFDialog.Filter = "Text Files (*.txt)|*.txt"
        If (SFDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK)
Then
            Dim ObjAtxt As New ArchivosTxt(SFDialog.FileName,
ObjDatosSeñal)
            MsgBox("Guardado con éxito.", MsgBoxStyle.DefaultButton1,
"Guardado")
        End If
    End Sub

    Private Sub BTNAbrirDatos_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BTNAbrirDatos.Click
        ChBHold.CheckState = CheckState.Checked
        If threadTRIGGERAUTO.ThreadState <> ThreadState.Unstarted Then
            threadTRIGGERAUTO.Abort()
        End If
        If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted Then
            threadTRIGGERSUB.Abort()
        End If
        If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted Then
            threadTRIGGERBAJ.Abort()
        End If
        Dim OFDialog As New OpenFileDialog
        OFDialog.InitialDirectory = "C:\"
        OFDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*"
```

```
        If (OFDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK)
Then
            fichero = OFDialog.FileName
            If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                If threadTRIGGERAUTO.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Abort()
                End If
                If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Abort()
                End If
                threadObtDatos.Abort()
            End If
            If threadObtDatos.ThreadState = ThreadState.Stopped Or
threadObtDatos.ThreadState = ThreadState.Aborted Then
                threadObtDatos = New Thread(New
ParameterizedThreadStart(AddressOf ObtenerDatos))
                threadObtDatos.Start(fichero)
            End If
            If threadObtDatos.ThreadState = ThreadState.Unstarted Then
                threadObtDatos.Start(fichero)
            End If
            MsgBox("Archivo leído.", MsgBoxStyle.DefaultButton1, "Abrir
Archivo")
            ChBDatosArch.CheckState = CheckState.Checked
            ChBHold.CheckState = CheckState.Unchecked
        End If

    End Sub

    Private Sub ButEspectroFFT_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ButEspectroFFT.Click
        If RBTRGAuto.Checked = False And RBTRGDef.Checked = False Then
            MsgBox("Por Favor, seleecione el modo de disparo o TRIGGER.")
        Else
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                MsgBox("Por Favor, desmarque la casilla Medida en DC e
intentelo de nuevo.")
            Else
                ComandFFT = New FormFFT
                ComandFFT.Show()

                If threadFFT.ThreadState = ThreadState.Running Then
                    threadFFT.Abort()
                End If
                If threadFFT.ThreadState = ThreadState.Stopped Or
threadFFT.ThreadState = ThreadState.Aborted Then
                    threadFFT = New Thread(New
ParameterizedThreadStart(AddressOf ComandFFT.ObtDatosFFT))
                    threadFFT.Start(ObjDatosSeñal)
                End If
                If threadFFT.ThreadState = ThreadState.Unstarted Then
                    threadFFT.Start(ObjDatosSeñal)
                End If
            End If
```

```
        End If
    End Sub

    Private Sub BSalir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BSalir.Click
        End
    End Sub

#End Region

#Region "Funciones de barra de menús"

    Private Sub AbrirArchivoTxtToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AbrirArchivoTxtToolStripMenuItem.Click
        ChBHold.CheckState = CheckState.Checked
        If threadTRIGGERAUTO.ThreadState <> ThreadState.Unstarted Then
            threadTRIGGERAUTO.Abort()
        End If
        If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted Then
            threadTRIGGERSUB.Abort()
        End If
        If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted Then
            threadTRIGGERBAJ.Abort()
        End If
        Dim OFDialog As New OpenFileDialog
        OFDialog.InitialDirectory = "C:\"
        OFDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*"
        If (OFDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK)
Then
            fichero = OFDialog.FileName
            If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                If threadTRIGGERAUTO.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Abort()
                End If
                If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Abort()
                End If
                threadObtDatos.Abort()
            End If
            If threadObtDatos.ThreadState = ThreadState.Stopped Or
threadObtDatos.ThreadState = ThreadState.Aborted Then
                threadObtDatos = New Thread(New
ParameterizedThreadStart(AddressOf ObtenerDatos))
                threadObtDatos.Start(fichero)
            End If
            If threadObtDatos.ThreadState = ThreadState.Unstarted Then
                threadObtDatos.Start(fichero)
            End If
            MsgBox("Archivo leído.", MsgBoxStyle.DefaultButton1, "Abrir
Archivo")
            ChBDatosArch.CheckState = CheckState.Checked
            ChBHold.CheckState = CheckState.Unchecked
        End If
```

```
    End Sub

    Private Sub GuardarToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
GuardarToolStripMenuItem.Click
        ChBHold.CheckState = CheckState.Checked
        Dim SFDialog As New SaveFileDialog
        SFDialog.InitialDirectory = "C:\"
        SFDialog.Filter = "Text Files (*.txt)|*.txt"
        If (SFDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK)
Then
            Dim ObjAtxt As New ArchivosTxt(SFDialog.FileName,
ObjDatosSeñal)
            MsgBox("Guardado con éxito.", MsgBoxStyle.DefaultButton1,
"Guardado")
        End If
    End Sub

    Private Sub SalirToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles SalirToolStripMenuItem.Click
        End
    End Sub

    Private Sub CanalAToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CanalAToolStripMenuItem.Click
        RBCanalA.Checked = True
    End Sub

    Private Sub CanalBToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CanalBToolStripMenuItem.Click
        RBCanalB.Checked = True
    End Sub

    Private Sub DualToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles DualToolStripMenuItem.Click
        RBDual.Checked = True
    End Sub

    Private Sub ModoXYToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ModoXYToolStripMenuItem.Click
        RBModoXY.Checked = True
    End Sub

    Private Sub ValorMedioToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ValorMedioToolStripMenuItem.Click
        If RBCanalA.Checked = True Then
            MsgBox(ObjDatosSeñal.ValorMedioCanA)
        ElseIf RBCanalB.Checked = True Then
            MsgBox(ObjDatosSeñal.ValorMedioCanB)
        Else
            MsgBox("Por favor, seleccione un único canal.")
        End If
    End Sub

    Private Sub ValorEficazToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ValorEficazToolStripMenuItem.Click
        If RBCanalA.Checked = True Then
            MsgBox(ObjDatosSeñal.ValorEficazCanA)
```

```
        ElseIf RBCanalB.Checked = True Then
            MsgBox(ObjDatosSeñal.ValorEficazCanB)
        Else
            MsgBox("Por favor, seleccione un único canal.")
        End If
    End Sub

    Private Sub NuevaCaptToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NuevaCaptToolStripMenuItem.Click
        ChBDatosArch.CheckState = CheckState.Unchecked
        If RBTRGAuto.Checked = False And RBTRGDef.Checked = False Then
            MsgBox("Por Favor, seleecione el modo de disparo o TRIGGER.")
        Else
            If RBTRGAuto.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    threadTRIGGERAUTO.Abort()
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    If threadFFT.ThreadState <> ThreadState.Unstarted Then
                        threadFFT.Abort()
                    End If
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Stopped Or
threadTRIGGERAUTO.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERAUTO = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturaAuto))
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERAUTO.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERAUTO.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
```

```
                    End If
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Stopped Or
threadTRIGGERSUB.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERSUB = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarSUB))
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERBAJ.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Stopped Or
threadTRIGGERBAJ.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERBAJ = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarBAJ))
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Unstarted
Then
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
            End If
            If threadREPRESENTAR.ThreadState <> ThreadState.Running Then
                If threadREPRESENTAR.ThreadState = ThreadState.Running Then
                    threadREPRESENTAR.Abort()
                End If
                If threadREPRESENTAR.ThreadState = ThreadState.Stopped Or
threadREPRESENTAR.ThreadState = ThreadState.Aborted Then
                    threadREPRESENTAR = New Thread(New
ParameterizedThreadStart(AddressOf Representar))
                    threadREPRESENTAR.Start()
                End If
                If threadREPRESENTAR.ThreadState = ThreadState.Unstarted
Then
                    threadREPRESENTAR.Start()
                End If
            End If
        End If
    End Sub
```

```
    Private Sub CopiarGraficaToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CopiarGraficaToolStripMenuItem.Click
        ChBHold.CheckState = CheckState.Checked
        ' Copiar la imagen del PictureBox1 en el portapapeles
        btmpCP = New Bitmap(PBGrafico.Width, PBGrafico.Height)
        dibujoCP = Graphics.FromImage(btmpCP)
        dibujoCP.CopyFromScreen(Me.Location.X + 14, Me.Location.Y + 52, 0,
0, PBGrafico.Size)
        Clipboard.SetDataObject(btmpCP, True)
    End Sub

    Private Sub DiagramaEspectralToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
DiagramaEspectralToolStripMenuItem.Click
        If RBTRGAuto.Checked = False And RBTRGDef.Checked = False Then
            MsgBox("Por Favor, seleecione el modo de disparo o TRIGGER.")
        Else
            ComandFFT.Show()

            If threadFFT.ThreadState = ThreadState.Running Then
                threadFFT.Abort()
            End If
            If threadFFT.ThreadState = ThreadState.Stopped Or
threadFFT.ThreadState = ThreadState.Aborted Then
                threadFFT = New Thread(New
ParameterizedThreadStart(AddressOf ComandFFT.ObtDatosFFT))
                threadFFT.Start(ObjDatosSeñal)
            End If
            If threadFFT.ThreadState = ThreadState.Unstarted Then
                threadFFT.Start(ObjDatosSeñal)
            End If
        End If
    End Sub

#End Region

#Region "Funciones definidas"

    Public Function Representar()
        While ChBHold.CheckState = CheckState.Unchecked
            If ChBMedidaDC.CheckState = CheckState.Checked Then
                For i As Integer = 0 To ObjDatosSeñal.Numelem - 1
                    ObjDatosSeñal.datosX1(i) =
Math.Abs(ObjDatosSeñal.datosX1(i))
                    ObjDatosSeñal.datosX2(i) =
Math.Abs(ObjDatosSeñal.datosX2(i))
                Next
            End If
            btmp = New Bitmap(PBGrafico.Width, PBGrafico.Height)
            dibujoEjes = Graphics.FromImage(btmp)
            dibujoA = Graphics.FromImage(btmp)
            dibujoB = Graphics.FromImage(btmp)
            ObjGrafica.LimpiarPantallaT(dibujoEjes, PBGrafico, btmp)
            If RBCanalA.Checked = True Then
                ObjGrafica.RepresentarCanalA(dibujoA, btmp, PBGrafico,
ObjDatosSeñal, VDIVA, TDIV, CoordX, CoordYA)
            Else
                If RBCanalB.Checked = True Then
                    ObjGrafica.RepresentarCanalB(dibujoB, btmp, PBGrafico,
ObjDatosSeñal, VDIVB, TDIV, CoordX, CoordYB)
                Else
                    If RBDual.Checked = True Then
```

```
                        ObjGrafica.RepresentarCanalAB(dibujoA, dibujoB,
btmp, PBGrafico, ObjDatosSeñal, VDIVA, VDIVB, TDIV, CoordX, CoordYA,
CoordYB)
                    Else
                        If RBModoXY.Checked = True Then
                            ObjGrafica.RepresentarModoXY(dibujoA, btmp,
PBGrafico, ObjDatosSeñal, VDIVA, VDIVB)
                        End If
                    End If
                End If
            End If
            PBGrafico.Image = btmp
        End While
        Return PBGrafico
    End Function

    Private Sub FormOsciloscopio_FormClosing(ByVal sender As Object, ByVal
e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        threadTRIGGERAUTO.Abort()
        threadTRIGGERSUB.Abort()
        threadTRIGGERBAJ.Abort()
        threadREPRESENTAR.Abort()
        threadObtDatos.Abort()
        threadFFT.Abort()
    End Sub

    Public Function RefrescarEscalas()
        LblVDivA.Text = "CH A= " & VDIVA & "V/DIV"
        LblVDivB.Text = "CH B= " & VDIVB & "V/DIV"
        LblTimeDiv.Text = "Time= " & TDIV & "s/DIV"
        Return VDIVA
    End Function

    Public Function ObtenerDatos(ByVal fichero As String)
        Dim fich As New ArchivosTxt
        fich.ObtenerStringT("T", fichero, ObjDatosSeñal)
        fich.ObtenerStringX1("X1", fichero, ObjDatosSeñal)
        fich.ObtenerStringX2("X2", fichero, ObjDatosSeñal)
        Return ObjDatosSeñal
    End Function

#End Region

#Region "Funciones botones TRIGGER"

    Private Sub RBTRGAuto_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RBTRGAuto.CheckedChanged
        If RBTRGAuto.Checked = True Then
            If threadTrigger.ThreadState <> ThreadState.Unstarted Then
                threadTrigger.Abort()
            End If
            GBOpTRG.Enabled = False
            If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                threadTRIGGERAUTO.Abort()
                If threadTRIGGERSUB.ThreadState <> ThreadState.Unstarted
Then
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState <> ThreadState.Unstarted
Then
```

```
                    threadTRIGGERBAJ.Abort()
                End If
                If threadObtDatos.ThreadState <> ThreadState.Unstarted Then
                    threadObtDatos.Abort()
                End If
            End If
            If threadTRIGGERAUTO.ThreadState = ThreadState.Stopped Or
threadTRIGGERAUTO.ThreadState = ThreadState.Aborted Then
                threadTRIGGERAUTO = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturaAuto))
                threadTRIGGERAUTO.Start(ObjDatosSeñal)
            End If
            If threadTRIGGERAUTO.ThreadState = ThreadState.Unstarted Then
                threadTRIGGERAUTO.Start(ObjDatosSeñal)
            End If
        End If
    End Sub

    Private Sub RBTRGDef_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RBTRGDef.CheckedChanged
        If RBTRGDef.Checked = True Then
            GBOpTRG.Enabled = True
        End If
    End Sub

    Private Sub BtnOKTRG_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnOKTRG.Click
        If RBTRGBJD.Checked = False And RBTRGSBD.Checked = False Then
            MsgBox("Por favor, seleccione el tipo de flanco del TRIGGER.")
        Else
            If Convert.ToDouble(TBValTRG.Text) < (-5) Then
                MsgBox("Por favor, Introduzca un número entre -5 y 5.")
            Else
                If Convert.ToDouble(TBValTRG.Text) > 5 Then
                    MsgBox("Por favor, Introduzca un número entre -5 y 5.")
                Else
                    TRIGGER = Convert.ToDouble(TBValTRG.Text)
                    BTRIGGER = ((TRIGGER / 27) * (128 / 0.4)) + 128
                    ObjCapturaDat.ObtenerBTRG(BTRIGGER)
                End If
            End If
            If RBTRGSBD.Checked = True Then
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERBAJ.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERBAJ.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERSUB.Abort()
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Stopped Or
threadTRIGGERSUB.ThreadState = ThreadState.Aborted Then
```

```
                    threadTRIGGERSUB = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarSUB))
                    ObjCapturaDat.ObtenerBTRG(BTRIGGER)
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERSUB.ThreadState = ThreadState.Unstarted
Then
                    ObjCapturaDat.ObtenerBTRG(BTRIGGER)
                    threadTRIGGERSUB.Start(ObjDatosSeñal)
                End If
            Else
                If threadObtDatos.ThreadState = ThreadState.Running Or
threadTRIGGERAUTO.ThreadState = ThreadState.Running Or
threadTRIGGERSUB.ThreadState = ThreadState.Running Or
threadTRIGGERBAJ.ThreadState = ThreadState.Running Then
                    If threadTRIGGERAUTO.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERAUTO.Abort()
                    End If
                    If threadTRIGGERSUB.ThreadState <>
ThreadState.Unstarted Then
                        threadTRIGGERSUB.Abort()
                    End If
                    If threadObtDatos.ThreadState <> ThreadState.Unstarted
Then
                        threadObtDatos.Abort()
                    End If
                    threadTRIGGERBAJ.Abort()
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Stopped Or
threadTRIGGERBAJ.ThreadState = ThreadState.Aborted Then
                    threadTRIGGERBAJ = New Thread(New
ParameterizedThreadStart(AddressOf ObjCapturaDat.CapturarBAJ))
                    ObjCapturaDat.ObtenerBTRG(BTRIGGER)
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
                If threadTRIGGERBAJ.ThreadState = ThreadState.Unstarted
Then
                    ObjCapturaDat.ObtenerBTRG(BTRIGGER)
                    threadTRIGGERBAJ.Start(ObjDatosSeñal)
                End If
            End If
        End If
    End Sub

#End Region

End Class
```

CÓDIGO MÓDULO ESTRUCTURA DE DATOS

```
Public Class DatosSeñal

#Region "Variables"

    Public Nmax As Integer = 22050
    Public Nmax2 As Integer = 16384

    Dim X1(Nmax) As Double
    Dim X2(Nmax) As Double
    Dim S1(Nmax2) As Double
    Dim S2(Nmax2) As Double
    Dim T(Nmax) As Double
    Dim F(Nmax2) As Double

#End Region

#Region "Propiedades"

    Public ReadOnly Property Numelem() As Integer
        Get
            Return Nmax
        End Get
    End Property

    Public ReadOnly Property NumelemF() As Integer
        Get
            Return Nmax2
        End Get
    End Property

    Public Property datosX1() As Double()
        Get
            Return X1
        End Get
        Set(ByVal value As Double())

        End Set
    End Property

    Public Property datosX2() As Double()
        Get
            Return X2

        End Get
        Set(ByVal value As Double())

        End Set
    End Property

    Public Property datosS1() As Double()
        Get
            Return S1
        End Get
        Set(ByVal value As Double())
```

```
        End Set
    End Property

    Public Property datosS2() As Double()
        Get
            Return S2
        End Get
        Set(ByVal value As Double())

        End Set
    End Property

    Public Property datosT() As Double()
        Get
            Return T
        End Get
        Set(ByVal value As Double())

        End Set
    End Property

    Public Property datosF1() As Double()
        Get
            Return F
        End Get
        Set(ByVal value As Double())

        End Set
    End Property

#End Region

#Region "Funciones"

    Public Function ConvX1TEnS1F1()
        Dim FFT As New FastFourierTrf
        FFT.FourierTransform(X1, X1, S1, F)
        Return S1
    End Function

    Public Function ConvX2TEnS2F2()
        Dim FFT As New FastFourierTrf
        FFT.FourierTransform(X2, T, S2, F)
        Return S2
    End Function

    Public Function ValorMedioCanA()
        Dim ValMedA As Double
        For cont As Integer = 0 To Numelem - 1
            ValMedA += X1(cont)
        Next
        ValMedA /= Numelem
        Return ValMedA
    End Function

    Public Function ValorMedioCanB()
        Dim ValMedB As Double
        For cont As Integer = 0 To Numelem - 1
            ValMedB += X2(cont)
        Next
        ValMedB /= Numelem
        Return ValMedB
    End Function
```

```
    Public Function ValorEficazCanA()
        Dim ValEfA As Double
        For cont As Integer = 0 To Numelem - 1
            ValEfA += (X1(cont) * X1(cont))
        Next
        ValEfA /= Numelem
        ValEfA = Math.Sqrt(ValEfA)
        Return ValEfA
    End Function

    Public Function ValorEficazCanB()
        Dim ValEfB As Double
        For cont As Integer = 0 To Numelem - 1
            ValEfB += (X2(cont) * X2(cont))
        Next
        ValEfB /= Numelem
        ValEfB = Math.Sqrt(ValEfB)
        Return ValEfB
    End Function

#End Region


End Class
```

CÓDIGO MÓDULO ARCHIVOS .TXT

```
Imports System.IO

Public Class ArchivosTxt

#Region "Variables"

    Public strFilename As String

    Public Shared ind1 As Integer = 1

#End Region

#Region "Constructores"

    Public Sub New()

    End Sub

    ' Constructor, acepta un nombre de fichero (si no existe se creará)
    Public Sub New(ByVal Filename As String, ByVal datos As DatosSeñal)
        strFilename = Filename
        'Escribimos en el fichero de configuracion las opciones por defecto
        If (System.IO.File.Exists(strFilename) = False) Then
            Dim sw As New StreamWriter(strFilename) ' crea un fichero
            sw.WriteLine("T") 'escribe el encabezado de una seccion
            For con = 0 To FormOsciloscopio.ObjDatosSeñal.Numelem
                sw.WriteLine(datos.datosT(con).ToString) 'escribe la clave
seguida del dato
            Next
            sw.WriteLine("X1")
            For con = 0 To FormOsciloscopio.ObjDatosSeñal.Numelem
                sw.WriteLine(datos.datosX1(con).ToString) 'escribe la clave
seguida del dato
            Next
            sw.WriteLine("X2")
            For con = 0 To FormOsciloscopio.ObjDatosSeñal.Numelem
                sw.WriteLine(datos.datosX2(con).ToString) 'escribe la clave
seguida del dato
            Next
            sw.Close()
        End If
    End Sub

#End Region

#Region "Propiedades"

    ' Propiedad para Read-only
    ReadOnly Property FileName() As String
        Get
            Return strFilename
        End Get
    End Property

#End Region
```

```
#Region "Funciones"

    Public Function ObtenerStringT(ByVal Seccion As String, ByVal fichero
As String, ByVal datos As DatosSeñal)
        Using sr As StreamReader = File.OpenText(fichero)
            Dim valor As String
            Dim ind As Integer = 0
1:          valor = sr.ReadLine
            If valor = "T" Then
                While valor <> "X1"
                    valor = sr.ReadLine
                    If valor <> "X1" Then
                        datos.datosT(ind) = CDbl(valor)
                        ind += 1
                    End If
                End While
            Else
                GoTo 1
            End If
        End Using
        Return datos
    End Function

    Public Function ObtenerStringX1(ByVal Seccion As String, ByVal fichero
As String, ByVal datos As DatosSeñal)
        Using sr As StreamReader = File.OpenText(fichero)
            Dim valor As String
            Dim ind As Integer = 0
2:          valor = sr.ReadLine
            If valor = "X1" Then
                While valor <> "X2"
                    valor = sr.ReadLine
                    If valor <> "X2" Then
                        datos.datosX1(ind) = CDbl(valor)
                        ind += 1
                    End If
                End While
            Else
                GoTo 2
            End If
        End Using
        Return datos
    End Function

    Public Function ObtenerStringX2(ByVal Seccion As String, ByVal fichero
As String, ByVal datos As DatosSeñal)
        Using sr As StreamReader = File.OpenText(fichero)
            Dim valor As String
            Dim ind As Integer = 0
3:          valor = sr.ReadLine
            If valor = "X2" Then
                While valor <> Nothing
                    valor = sr.ReadLine
                    If valor <> Nothing Then
                        datos.datosX2(ind) = CDbl(valor)
                        ind += 1
                    End If
                End While
            Else
                GoTo 3
            End If
        End Using
```

```
        Return datos
    End Function

#End Region


End Class
```

CÓDIGO MÓDULO CAPTURA DE DATOS

```
#Region "Definir librerías"

Imports MDS = Microsoft.DirectX.DirectSound
Imports System.IO

#End Region

Public Class CapturaDatos

#Region "Decalración variables"

    Public Captura As MDS.Capture
    Public PtrBC As Integer = 0
    Public BuffCapDesc As MDS.CaptureBufferDescription
    Public BuffCaptura As MDS.CaptureBuffer
    Public BuffOutput As MDS.Buffer
    Public BuffDec As MDS.BufferDescription
    Public Format As MDS.WaveFormat
    Public wave(50000) As Byte
    Public Datos As MemoryStream
    Public Notifysize As Double
    Public CaptureBuffSize As Integer
    Public Num_Buffer As Integer
    Public bytetrg As Byte
    Public Control As Boolean = False
    Public captOK As Boolean = False

#End Region

#Region "Constructor"

    Public Sub New()


        Num_Buffer = 2
        Format.SamplesPerSecond = 44100
        Format.BitsPerSample = 8
        Format.Channels = 2
        Format.FormatTag = MDS.WaveFormatTag.Pcm
        Format.BlockAlign = Convert.ToInt16((Format.Channels *
(Format.BitsPerSample / 8)))
        Format.AverageBytesPerSecond = Format.SamplesPerSecond *
Format.BlockAlign

        Notifysize = Math.Max(44096, Format.AverageBytesPerSecond / 8)
        Notifysize -= (Notifysize Mod Format.BlockAlign)
        CaptureBuffSize = Num_Buffer * Notifysize


        Datos = New MemoryStream(wave)

        Captura = New MDS.Capture
        BuffCapDesc = New MDS.CaptureBufferDescription
        BuffCapDesc.Format = Format
        BuffCapDesc.BufferBytes = CaptureBuffSize
```

```
        BuffCaptura = New MDS.CaptureBuffer(BuffCapDesc, Captura)


    End Sub

#End Region

#Region "Propiedades"
    Public ReadOnly Property capturaOK() As Boolean
        Get
            Return captOK
        End Get
    End Property
#End Region

#Region "Funciones"

    Public Function ObtenerBTRG(ByVal btrigger As Byte)
        Me.bytetrg = btrigger
        Return bytetrg
    End Function

    Public Function CapturaAuto(ByVal datocap As DatosSeñal)
        While Control = False
            Dim T1Byte As Double
            Dim TLect As Integer = 1000
            Dim BytesSecond As Integer
            Dim NBytes As Integer
            Dim posicion As Integer
            captOK = False
            BytesSecond = (Format.BitsPerSample / 8) *
Format.SamplesPerSecond
            NBytes = BytesSecond * (TLect / 1000)
            T1Byte = 1 / Format.SamplesPerSecond
            Datos.Position = 0
            BuffCaptura.Start(True)
            Threading.Thread.Sleep(TLect)
            While captOK = False
                BuffCaptura.Read(PtrBC, Datos, NBytes, 0)
                Datos.Position = 0
                bytetrg = 128
                For count As Integer = 0 To wave.Length - 20 Step 2
                    If wave(count) = bytetrg Then
                        If wave(count) < wave(count + 20) Then
                            posicion = count
                            For i As Integer = 0 To (datocap.Numelem) - 1
                                If i = 0 Then
                                    datocap.datosT(i) = 0
                                Else
                                    datocap.datosT(i) = datocap.datosT(i -
1) + (2 * T1Byte)
                                End If
                            Next
                            Dim h As Integer = 0
                            For int As Integer = posicion To (wave.Length)
- 2 Step 2
                                If h < datocap.Numelem Then
                                    datocap.datosX1(h) = (wave(int) - 128)
* (0.4 / 128) * 10
                                    datocap.datosX2(h) = (wave(int + 1) -
128) * (0.4 / 128) * 10
                                    h += 1
                                Else
```

```
                                    Exit For
                                End If
                            Next
                            captOK = True
                            Exit While
                        End If
                    End If
                Next
            End While
        End While
        Return datocap
    End Function

    Public Function CapturarSUB(ByVal DatoCap As DatosSeñal)
        While Control = False
            Dim T1Byte As Double
            Dim TLect As Integer = 1000
            Dim BytesSecond As Integer
            Dim NBytes As Integer
            Dim posicion As Integer
            captOK = False
            BytesSecond = (Format.BitsPerSample / 8) *
Format.SamplesPerSecond
            NBytes = BytesSecond * (TLect / 1000)
            T1Byte = 1 / Format.SamplesPerSecond
            Datos.Position = 0
            BuffCaptura.Start(True)
            Threading.Thread.Sleep(TLect)
            While captOK = False
                BuffCaptura.Read(PtrBC, Datos, NBytes, 0)
                Datos.Position = 0
                bytetrg = bytetrg
                For count As Integer = 0 To wave.Length - 20 Step 2
                    If wave(count) = bytetrg Then
                        If wave(count) < wave(count + 20) Then
                            posicion = count
                            For i As Integer = 0 To (DatoCap.Numelem) - 1
                                If i = 0 Then
                                    DatoCap.datosT(i) = 0
                                Else
                                    DatoCap.datosT(i) = DatoCap.datosT(i -
1) + (2 * T1Byte)
                                End If
                            Next
                            Dim h As Integer = 0
                            For int As Integer = posicion To (wave.Length)
- 2 Step 2
                                If h < DatoCap.Numelem Then
                                    DatoCap.datosX1(h) = (wave(int) - 128)
* (0.4 / 128) * 10
                                    DatoCap.datosX2(h) = (wave(int + 1) -
128) * (0.4 / 128) * 10
                                    h += 1
                                Else
                                    Exit For
                                End If
                            Next
                            captOK = True
                            Exit While
                        End If
                    End If
                Next
            End While
```

```
        End While
        Return DatoCap
    End Function

    Public Function CapturarBAJ(ByVal DatoCap As DatosSeñal)
        While Control = False
            Dim T1Byte As Double
            Dim TLect As Integer = 1000
            Dim BytesSecond As Integer
            Dim NBytes As Integer
            Dim posicion As Integer
            captOK = False
            BytesSecond = (Format.BitsPerSample / 8) *
Format.SamplesPerSecond
            NBytes = BytesSecond * (TLect / 1000)
            T1Byte = 1 / Format.SamplesPerSecond
            Datos.Position = 0
            BuffCaptura.Start(True)
            Threading.Thread.Sleep(TLect)
            While captOK = False
                BuffCaptura.Read(PtrBC, Datos, NBytes, 0)
                Datos.Position = 0
                bytetrg = bytetrg
                For count As Integer = 0 To wave.Length - 20 Step 2
                    If wave(count) = bytetrg Then
                        If wave(count) > wave(count + 20) Then
                            posicion = count
                            For i As Integer = 0 To (DatoCap.Numelem) - 1
                                If i = 0 Then
                                    DatoCap.datosT(i) = 0
                                Else
                                    DatoCap.datosT(i) = DatoCap.datosT(i -
1) + (2 * T1Byte)
                                End If
                            Next
                            Dim h As Integer = 0
                            For int As Integer = posicion To (wave.Length)
- 2 Step 2
                                If h < DatoCap.Numelem Then
                                    DatoCap.datosX1(h) = (wave(int) - 128)
* (0.4 / 128) * 10
                                    DatoCap.datosX2(h) = (wave(int + 1) -
128) * (0.4 / 128) * 10
                                    h += 1
                                Else
                                    Exit For
                                End If
                            Next
                            captOK = True
                            Exit While
                        End If
                    End If
                Next
            End While
        End While
        Return DatoCap
    End Function

#End Region

End Class
```

CÓDIGO MÓDULO TRANSFORMADA FOURIER

```
Imports System.Math

Public Class FastFourierTrf

#Region "Variables"

    Dim NumSamples As Integer
    Dim AngleNumerator As Double

#End Region

#Region "Constructor"

    Public Sub New()
        NumSamples = 16384
        AngleNumerator = 2 * PI
    End Sub

#End Region

#Region "Funciones definidas"

    Public Function NumberOfBitsNeeded(ByVal PowerOfTwo As Integer) As
Short
        Dim i As Integer
        For i = 0 To 16
            If (PowerOfTwo And (1 << i)) <> 0 Then
                NumberOfBitsNeeded = i
            End If
        Next
    End Function

    Public Function ReverseBits(ByVal Index As Integer, ByVal NumBits As
Integer) As Short
        Dim i, rev As Integer
        rev = 0
        For i = 0 To NumBits - 1
            rev = (rev << 1) Or (Index And 1)
            Index = Index >> 1
        Next
        ReverseBits = rev
    End Function

    Public Function FourierTransform(ByVal RealIn() As Double, ByVal ImIn()
As Double, ByVal RealOut() As Double, ByVal ImOut() As Double)
        Dim NumBits, i, j, k, n, BlockEnd, BlockSize As Integer
        Dim delta_angle, delta_ar As Double
        Dim alpha, beta As Double
        Dim tr, ti, ar, ai As Double

        NumBits = NumberOfBitsNeeded(NumSamples)
        For i = 0 To NumSamples - 1
            j = ReverseBits(i, NumBits)
            RealOut(j) = RealIn(i)
            ImOut(j) = ImIn(i)
```

```
        Next
        BlockEnd = 1
        BlockSize = 2
        While BlockEnd < NumSamples
            delta_angle = AngleNumerator / BlockSize
            alpha = Sin(0.5 * delta_angle)
            alpha = 2 * alpha * alpha
            beta = Sin(delta_angle)
            i = 0
            While i < NumSamples
                ar = 1
                ai = 0
                j = i
                For n = 0 To BlockEnd - 1
                    k = j + BlockEnd
                    tr = ar * RealOut(k) - ai * ImOut(k)
                    ti = ar * ImOut(k) + ai * RealOut(k)
                    RealOut(k) = RealOut(j) - tr
                    ImOut(k) = ImOut(j) - ti
                    RealOut(j) = RealOut(j) + tr
                    ImOut(j) = ImOut(j) + ti
                    delta_ar = alpha * ar + beta * ai
                    ai = ai - (alpha * ai - beta * ar)
                    ar = ar - delta_ar
                    j += 1
                Next
                i = i + BlockSize
            End While
            BlockEnd = BlockSize
            BlockSize = BlockSize << 1
        End While
        For i = 0 To NumSamples - 1
            RealOut(i) = Abs(RealOut(i))
            ImOut(i) = Abs(ImOut(i))
        Next
        Return ImOut
    End Function

#End Region


End Class
```

CÓDIGO MÓDULO FORMULARIO FFT

```
Imports System.Threading

Public Class FormFFT

#Region "variables"
    Public datosFFT As DatosSeñal
    Public GraficaFFT As Grafica

    Public dibujoEjesF As Graphics
    Public dibujoAF As Graphics
    Public dibujoBF As Graphics
    Public dibujoCPFFT As Graphics

    Public btmpFFT As Bitmap
    Public btmpCPFFT As Bitmap

    Public HOLDFFT_Activado As Boolean

    Public threadREPRESENTARFFT As Thread

#End Region

#Region "Constructor"

    Public Sub New()

        ' Llamada necesaria para el Diseñador de Windows Forms.
        InitializeComponent()

        ' Agregue cualquier inicialización después de la llamada a
InitializeComponent().

        datosFFT = New DatosSeñal
        GraficaFFT = New Grafica

        HOLDFFT_Activado = False

        threadREPRESENTARFFT = New Thread(New
ParameterizedThreadStart(AddressOf RepresentarFFT))

    End Sub

#End Region

#Region "Botones"

    Private Sub BFFTSalir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BFFTSalir.Click
        Close()
    End Sub

    Private Sub BFFTCoPantalla_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BFFTCoPantalla.Click
        ChBHOLDFFT.CheckState = CheckState.Checked
        ' Copiar la imagen del PictureBox1 en el portapapeles
```

```
        btmpCPFFT = New Bitmap(PBFFT.Width, PBFFT.Height)
        dibujoCPfft = Graphics.FromImage(btmpCPFFT)
        dibujoCPFFT.CopyFromScreen(Me.Location.X + 11, Me.Location.Y + 33,
0, 0, PBFFT.Size)
        PBFFT.Image = btmpCPFFT
        Clipboard.SetDataObject(btmpCPFFT, True)
    End Sub

    Private Sub BfftInicio_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BfftInicio.Click
        If threadREPRESENTARFFT.ThreadState = ThreadState.Running Then
            threadREPRESENTARFFT.Abort()
        End If
        If threadREPRESENTARFFT.ThreadState = ThreadState.Stopped Or
threadREPRESENTARFFT.ThreadState = ThreadState.Aborted Then
            threadREPRESENTARFFT = New Thread(New
ParameterizedThreadStart(AddressOf RepresentarFFT))
            threadREPRESENTARFFT.Start()
        End If
        If threadREPRESENTARFFT.ThreadState = ThreadState.Unstarted Then
            threadREPRESENTARFFT.Start()
        End If
    End Sub

    Private Sub ChBHOLDFFT_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ChBHOLDFFT.CheckedChanged
        If ChBHOLDFFT.CheckState = CheckState.Unchecked Then
            If threadREPRESENTARFFT.ThreadState = ThreadState.Running Then
                threadREPRESENTARFFT.Abort()
            End If
            If threadREPRESENTARFFT.ThreadState = ThreadState.Stopped Or
threadREPRESENTARFFT.ThreadState = ThreadState.Aborted Then
                threadREPRESENTARFFT = New Thread(New
ParameterizedThreadStart(AddressOf RepresentarFFT))
                threadREPRESENTARFFT.Start()
            End If
            If threadREPRESENTARFFT.ThreadState = ThreadState.Unstarted
Then
                threadREPRESENTARFFT.Start()
            End If
        End If
    End Sub

#End Region

#Region "Funciones definidas"

    Public Function ObtDatosFFT(ByVal datos As DatosSeñal)
        While ChBHOLDFFT.CheckState = CheckState.Unchecked
            datosFFT = datos
            datosFFT.ConvX1TEnS1F1()
            datosFFT.ConvX2TEnS2F2()
        End While
        Return datosFFT
    End Function

    Public Function RepresentarFFT()
        While ChBHOLDFFT.CheckState = CheckState.Unchecked
            btmpFFT = New Bitmap(PBFFT.Width, PBFFT.Height)
            dibujoEjesF = Graphics.FromImage(btmpFFT)
            dibujoAF = Graphics.FromImage(btmpFFT)
            dibujoBF = Graphics.FromImage(btmpFFT)
            GraficaFFT.LimpiarPantallaF(dibujoEjesF, PBFFT, btmpFFT)
```

```
            If RBFFTCanalA.Checked = True Then
                GraficaFFT.RepresentarFFTCanA(dibujoAF, btmpFFT, PBFFT,
datosFFT)
            Else
                If RBFFTCanalB.Checked = True Then
                    GraficaFFT.RepresentarFFTCanB(dibujoBF, btmpFFT, PBFFT,
datosFFT)
                Else
                    If RBFFTDual.Checked = True Then
                        GraficaFFT.RepresentarFFTCanAB(dibujoAF, dibujoBF,
btmpFFT, PBFFT, datosFFT)
                    End If
                End If
            End If
            PBFFT.Image = btmpFFT
        End While
        Return PBFFT
    End Function

    Private Sub FormFFT_FormClosing(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        threadREPRESENTARFFT.Abort()
    End Sub

#End Region



End Class
```

CÓDIGO MÓDULO GRÁFICA

```
Public Class Grafica

#Region "Variables"

    Public EscalaX As Double
    Public EscalaY As Double

    Public Px1 As Integer
    Public Px2 As Integer
    Public Py1 As Integer
    Public Py2 As Integer

    Public CentrCoordX As Integer
    Public CentrCoordYA As Integer
    Public CentrCoordYB As Integer
#End Region

#Region "Constructor"

    Public Sub New()
        EscalaX = 100
        EscalaY = 80
    End Sub

#End Region

#Region "Funciones"

    Public Function RepresentarCanalA(ByRef dibujo As Graphics, ByRef btm
As Bitmap, ByRef PBdibujo As PictureBox, ByVal datos As DatosSeñal, ByVal
VDIVA As Double, ByVal TDIV As Double, ByVal CoordX As Integer, ByVal
CoordYA As Integer)
        dibujo.TranslateTransform(CoordX, CoordYA) 'tranladamos cordenadas
        dibujo.ScaleTransform(1, -1)                'convertimos a
cordenadas normales
        Dim Ind, a, b, c, d As Integer
        For Ind = 0 To (datos.Numelem) - 2
            Px1 = datos.datosT(Ind) * (PBdibujo.Width / 20) / TDIV
            Py1 = datos.datosX1(Ind) * (PBdibujo.Height / 20) / VDIVA
            Px2 = datos.datosT(Ind + 1) * (PBdibujo.Width / 20) / TDIV
            Py2 = datos.datosX1(Ind + 1) * (PBdibujo.Height / 20) / VDIVA
            a = Math.Round(Px1)
            b = Math.Round(Py1)
            c = Math.Round(Px2)
            d = Math.Round(Py2)
            dibujo.DrawLine(Pens.Blue, a, b, c, d)
        Next
        Return PBdibujo
    End Function

    Public Function RepresentarCanalB(ByRef dibujo As Graphics, ByRef btm
As Bitmap, ByRef PBdibujo As PictureBox, ByVal datos As DatosSeñal, ByVal
VDIVB As Double, ByVal TDIV As Double, ByVal CoordX As Integer, ByVal
CoordYB As Integer)
        dibujo.TranslateTransform(CoordX, CoordYB) 'tranladamos cordenadas
```

```
        dibujo.ScaleTransform(1, -1)                'convertimos a
cordenadas normales

        Dim Ind, a, b, c, d As Integer
        For Ind = 0 To (datos.Numelem) - 2
            Px1 = datos.datosT(Ind) * (PBdibujo.Width / 20) / TDIV
            Py1 = datos.datosX2(Ind) * (PBdibujo.Height / 20) / VDIVB
            Px2 = datos.datosT(Ind + 1) * (PBdibujo.Width / 20) / TDIV
            Py2 = datos.datosX2(Ind + 1) * (PBdibujo.Height / 20) / VDIVB
            a = Math.Round(Px1)
            b = Math.Round(Py1)
            c = Math.Round(Px2)
            d = Math.Round(Py2)
            dibujo.DrawLine(Pens.Red, a, b, c, d)
        Next
        Return PBdibujo
    End Function

    Public Function RepresentarModoXY(ByRef dibujo As Graphics, ByRef btm
As Bitmap, ByRef PBdibujo As PictureBox, ByVal datos As DatosSeñal, ByVal
VDIVA As Double, ByVal VDIVB As Double)
        CentrCoordX = PBdibujo.Width / 2
        CentrCoordYA = PBdibujo.Height / 2
        dibujo.TranslateTransform(CentrCoordX, CentrCoordYA) 'tranladamos
cordenadas
        dibujo.ScaleTransform(1, -1)                'convertimos a
cordenadas normales

        Dim Ind, a, b, c, d As Integer
        For Ind = 0 To datos.Numelem - 2
            Px1 = datos.datosX2(Ind) * (PBdibujo.Width / 20) / VDIVB
            Py1 = datos.datosX1(Ind) * (PBdibujo.Height / 20) / VDIVA
            Px2 = datos.datosX2(Ind + 1) * (PBdibujo.Width / 20) / VDIVB
            Py2 = datos.datosX1(Ind + 1) * (PBdibujo.Height / 20) / VDIVA
            a = Math.Round(Px1)
            b = Math.Round(Py1)
            c = Math.Round(Px2)
            d = Math.Round(Py2)
            dibujo.DrawLine(Pens.Blue, a, b, c, d)
        Next
        Return PBdibujo
    End Function

    Public Function RepresentarCanalAB(ByRef dibujoCA As Graphics, ByRef
dibujoCB As Graphics, ByRef btm As Bitmap, ByRef PBdibujo As PictureBox,
ByVal datos As DatosSeñal, ByVal VDIVA As Double, ByVal VDIVB As Double,
ByVal TDIV As Double, ByVal CoordX As Integer, ByVal CoordYA As Integer,
ByVal CoordYB As Integer)
        CentrCoordX = CoordX
        CentrCoordYA = CoordYA
        CentrCoordYB = CoordYB
        RepresentarCanalA(dibujoCA, btm, PBdibujo, datos, VDIVA, TDIV,
CentrCoordX, CentrCoordYA)
        RepresentarCanalB(dibujoCB, btm, PBdibujo, datos, VDIVB, TDIV,
CentrCoordX, CentrCoordYB)
        Return PBdibujo
    End Function

    Public Function RepresentarFFTCanA(ByRef dibujoF As Graphics, ByRef btm
As Bitmap, ByRef PBdibujoF As PictureBox, ByVal datos As DatosSeñal)
        Dim PCanAF As New Pen(Color.Blue)
        CentrCoordX = 10
        CentrCoordYA = PBdibujoF.Height - 25
```

```
        dibujoF.TranslateTransform(CentrCoordX, CentrCoordYA) 'tranladamos
cordenadas al centro
        dibujoF.ScaleTransform(1, -1)    'convertimos a cordenadas normales

        Dim Ind, a, b, c, d As Integer
        For Ind = 1 To datos.NumelemF - 5
            Px1 = Math.Log10(Ind) * 162
            Py1 = datos.datosS1(Ind) / (PBdibujoF.Height / 4)
            Px2 = Math.Log10(Ind + 1) * 162
            Py2 = datos.datosS1(Ind + 1) / (PBdibujoF.Height / 4)
            a = Math.Round(Px1)
            b = Math.Round(Py1)
            c = Math.Round(Px2)
            d = Math.Round(Py2)
            dibujoF.DrawLine(PCanAF, a, b, c, d)
        Next
        Return PBdibujoF
    End Function

    Public Function RepresentarFFTCanB(ByRef dibujoF As Graphics, ByRef btm
As Bitmap, ByRef PBdibujoF As PictureBox, ByVal datos As DatosSeñal)
        Dim PCanBF As New Pen(Color.Red)
        CentrCoordX = 10
        CentrCoordYA = PBdibujoF.Height - 10
        dibujoF.TranslateTransform(CentrCoordX, CentrCoordYA) 'tranladamos
cordenadas al centro
        dibujoF.ScaleTransform(1, -1)    'convertimos a cordenadas normales

        Dim Ind, a, b, c, d As Integer
        For Ind = 0 To PBdibujoF.Width - 10
            Px1 = Ind
            Py1 = datos.datosS2(Ind) / (PBdibujoF.Height / 10)
            Px2 = (Ind + 1)
            Py2 = datos.datosS2(Ind + 1) / (PBdibujoF.Height / 10)
            a = Math.Round(Px1)
            b = Math.Round(Py1)
            c = Math.Round(Px2)
            d = Math.Round(Py2)
            dibujoF.DrawLine(PCanBF, a, b, c, d)
        Next
        Return PBdibujoF
    End Function

    Public Function RepresentarFFTCanAB(ByRef dibujoFCA As Graphics, ByRef
dibujoFCB As Graphics, ByRef btm As Bitmap, ByRef PBdibujoF As PictureBox,
ByVal datos As DatosSeñal)
        RepresentarFFTCanA(dibujoFCA, btm, PBdibujoF, datos)
        RepresentarFFTCanB(dibujoFCB, btm, PBdibujoF, datos)
        Return PBdibujoF
    End Function

    Public Function DibEjesT(ByRef dibujo As Graphics, ByRef PBdibujo As
PictureBox, ByRef btm As Bitmap)
        Dim Ejes As New Pen(Color.Black)
        EscalaX = 100
        EscalaY = 80
        CentrCoordX = PBdibujo.Width / 2
        CentrCoordYA = PBdibujo.Height / 2
        dibujo.TranslateTransform(CentrCoordX, CentrCoordYA) 'tranladamos
cordenadas al centro
        dibujo.ScaleTransform(1, -1)                'convertimos a
cordenadas normales
```

```
        ' DIBUJAMOS EJES X-Y
        dibujo.DrawLine(Ejes, CentrCoordX * -1, 0, PBdibujo.Width -
CentrCoordX, 0)  'eje X
        dibujo.DrawLine(Ejes, 0, CentrCoordYA, 0, CentrCoordYA * -1) 'eje Y

        'DIBUJAMOS PUNTOS +X
        Dim a As Integer
        For a = 0 To PBdibujo.Width Step (PBdibujo.Width / EscalaX)
            dibujo.DrawLine(Ejes, a, 3, a, -3)
            dibujo.DrawLine(Ejes, 10 * a, 5, 10 * a, -5)
        Next
        'DIBUJAMOS PUNTOS -X
        For a = 0 To CentrCoordX * -1 Step (-1 * PBdibujo.Width / EscalaX)
            dibujo.DrawLine(Ejes, a, 3, a, -3)
            dibujo.DrawLine(Ejes, 10 * a, 5, 10 * a, -5)

        Next

        'DIBUJAMOS PUNTOS Y
        For a = 0 To PBdibujo.Height Step (PBdibujo.Height / EscalaY)
            dibujo.DrawLine(Ejes, 3, a, -3, a)
            dibujo.DrawLine(Ejes, 5, 10 * a, -5, 10 * a)
        Next
        'DIBUJAMOS PUNTOS -Y
        For a = 0 To PBdibujo.Height * -1 Step ((-1 * PBdibujo.Height) /
EscalaY)
            dibujo.DrawLine(Ejes, 3, a, -3, a)
            dibujo.DrawLine(Ejes, 5, 10 * a, -5, 10 * a)
        Next
        Return PBdibujo
    End Function

    Public Function DibEjesF(ByRef dibujoF As Graphics, ByRef PBdibujo As
PictureBox, ByRef btm As Bitmap)
        Dim EjesF As New Pen(Color.Black)
        EscalaX = 50
        EscalaY = 12
        CentrCoordX = 10
        CentrCoordYA = PBdibujo.Height - 25
        dibujoF.TranslateTransform(CentrCoordX, CentrCoordYA) 'tranladamos
cordenadas al centro
        dibujoF.ScaleTransform(1, -1)                'convertimos a
cordenadas normales

        ' DIBUJAMOS EJES X-Y
        dibujoF.DrawLine(EjesF, CentrCoordX * -1, 0, PBdibujo.Width -
CentrCoordX, 0)  'eje X
        dibujoF.DrawLine(EjesF, 0, CentrCoordYA, 0, CentrCoordYA * -1) 'eje
Y

        'DIBUJAMOS PUNTOS +X
        Dim a As Integer
        For a = 0 To PBdibujo.Width Step (PBdibujo.Width / EscalaX)
            dibujoF.DrawLine(EjesF, 5 * a, 5, 5 * a, -5)
            dibujoF.DrawLine(EjesF, 10 * a, 8, 10 * a, -8)
        Next
        'DIBUJAMOS PUNTOS -X
        For a = 0 To CentrCoordX * -1 Step (-1 * PBdibujo.Width / EscalaX)
            dibujoF.DrawLine(EjesF, 5 * a, 5, 5 * a, -5)
            dibujoF.DrawLine(EjesF, 10 * a, 8, 10 * a, -8)

        Next
```

```
        'DIBUJAMOS PUNTOS Y
        For a = 0 To PBdibujo.Height Step (PBdibujo.Height / EscalaY)
            dibujoF.DrawLine(EjesF, 3, a, -3, a)
            dibujoF.DrawLine(EjesF, 5, 10 * a, -5, 10 * a)
        Next
        'DIBUJAMOS PUNTOS -Y
        For a = 0 To PBdibujo.Height * -1 Step ((-1 * PBdibujo.Height) /
EscalaY)
            dibujoF.DrawLine(EjesF, 3, a, -3, a)
            dibujoF.DrawLine(EjesF, 5, 10 * a, -5, 10 * a)
        Next
        Return PBdibujo
    End Function

    Public Function LimpiarPantallaT(ByRef dibujo As Graphics, ByRef
PBdibujo As PictureBox, ByRef btm As Bitmap)
        dibujo.Clear(Color.Azure)
        DibEjesT(dibujo, PBdibujo, btm)
        Return dibujo
    End Function

    Public Function LimpiarPantallaF(ByRef dibujoF As Graphics, ByRef
PBdibujo As PictureBox, ByRef btm As Bitmap)
        dibujoF.Clear(Color.Azure)
        DibEjesF(dibujoF, PBdibujo, btm)
        Return dibujoF
    End Function

#End Region

End Class
```