

Public Class Voltmetro

Public Structure DDL_CHANNEL_CONTROL_DATA

Public bGainTimesTen As Boolean

Public GainuVBit As Integer

Public bCouplingState As Boolean

Public bActiveState As Boolean

End Structure

Private Declare Function DDL1M12_GetNumDevices Lib "ddl1m12" (ByRef NumDevices As Long) As Integer

Private Declare Function DDL1M12_GetDeviceSerialNumber Lib "ddl1m12" (ByVal SerialNumberIndex As Integer, ByVal SerialNumber As String, ByVal BufferSize As Integer) As Integer

Private Declare Function DDL1M12_OpenSpecifiedDevice Lib "ddl1m12" (ByVal SerialNumber As String, ByRef ddiHandle As Long) As Integer

Private Declare Function DDL1M12_OpenDevice Lib "ddl1m12" (ByRef ddiHandle As Long) As Integer

Private Declare Function DDL1M12_CloseDevice Lib "ddl1m12" (ByVal ddiHandle As Integer) As Integer

Private Declare Function DDL1M12_InitDevice Lib "ddl1m12" (ByVal ddiHandle As Integer) As Integer

Private Declare Function DDL1M12_ProgramDevice Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal FileName As String) As Integer

Private Declare Function DDL1M12_IsDeviceProgrammed Lib "ddl1m12" (ByVal ddiHandle As Integer, ByRef IpbProgramState As Boolean) As Integer

Private Declare Function DDL1M12_SetDeviceChannelState Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal bChannelAActiveState As Boolean, ByVal bChannelBActiveState As Boolean) As Integer

Private Declare Function DDL1M12_StartDeviceDataLogging Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal bTestMode As Boolean, ByVal SampleIntervalSecs As Integer, ByRef ChannelAControlData As DDL_CHANNEL_CONTROL_DATA, ByRef ChannelBControlData As DDL_CHANNEL_CONTROL_DATA) As Integer

Private Declare Function DDL1M12_StopDeviceDataLogging Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal Channel As Integer) As Integer

Private Declare Function DDL1M12_GetDeviceChannelData Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal ChannelACDLB As IntPtr, ByRef IpChannelANumValuesReturned As Long, ByVal ChannelBCDLB As IntPtr, ByRef IpChannelBNumValuesReturned As Long) As Integer

Private Declare Function DDL1M12_SetDeviceFunctionGeneratorState Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal bActiveState As Boolean) As Integer

Private Declare Function DDL1M12_GetWaveformFrequencyValues Lib "ddl1m12" (ByVal OutputWaveform As Integer, ByVal FrequencyValue As Integer, ByVal NumSamples As Integer, ByVal FileName As String, ByRef IpFrequencyHz As Long, ByRef IpMaxFreqRangeHz As Long, ByRef IpMinFreqRangeHz As Long) As Integer

Private Declare Function DDL1M12_SetDeviceFunctionGeneratorWaveform Lib "ddl1m12" (ByVal ddiHandle As Integer, ByVal OutputWaveform As Integer, ByVal FrequencyValue As Integer, ByVal PeakToPeakVolts As Integer, ByVal OffsetVolts As Integer, ByVal NumSamples As Integer, ByVal PulseDutyRatio As Integer, ByVal bInvertWaveform As Boolean, ByVal FileName As String) As Integer

Private Declare Function DDL1M12_GetDllVersion Lib "ddl1m12" (ByVal DllVersion As String, ByVal BufferSize As Integer) As Integer

Private Declare Function DDL1M12_GetErrorCodeString Lib "ddl1m12" (ByVal Language As String, ByVal StatusCode As Integer, ByVal ErrorMessage As String, ByVal BufferSize As Integer) As Integer

Private Const DDL_SUCCESS As Integer = 0

Private Const CHANNEL_A As Integer = 1

Private Const CHANNEL_B As Integer = 2

Private Const GAIN_100UV_PER_BIT As Integer = 1

Private Const GAIN_200UV_PER_BIT As Integer = 2

Private Const GAIN_500UV_PER_BIT As Integer = 3

Private Const GAIN_1MV_PER_BIT As Integer = 4

Private Const GAIN_2MV_PER_BIT As Integer = 5

Private Const GAIN_5MV_PER_BIT As Integer = 6

Private Const GAIN_10MV_PER_BIT As Integer = 7

Private Const WAVEFORM_DC As Integer = 1

Private Const WAVEFORM_SQUARE As Integer = 2

Private Const WAVEFORM_SAWTOOTH As Integer = 3

Private Const WAVEFORM_SIN_COS As Integer = 4

Private Const WAVEFORM_TRIANGULAR As Integer = 5

Private Const WAVEFORM_PULSE As Integer = 6

Private Const WAVEFORM_CUSTOM As Integer = 7

Private Const DATA_LOGGER_BUFFER_SIZE As Integer = 16384

Private Const NUM_DATA_LOG_ELEMENTS As Integer = 4

```
Private Const DATA_VALUE_INDEX As Integer = 0
Private Const MISSING_VALUE_INDEX As Integer = 1
Private Const MAX_VALUE_CLIPPED_INDEX As Integer = 2
Private Const MIN_VALUE_CLIPPED_INDEX As Integer = 3

Private Const MAX_NUM_SERIAL_NUMBER_CHARS As Integer = 50

Private Const MAX_NUM_DLL_VERSION_CHARS As Integer = 10

Private Const MAX_NUM_ERROR_MESSAGE_CHARS As Integer = 100
```

```
Private ddlHandle As Long = 0
```

```
Dim ddlStatus As Integer = DDL_SUCCESS
Dim NumDevices As Long = 0
Dim SerialNumber As String = New String(ChrW(0), MAX_NUM_SERIAL_NUMBER_CHARS)
Dim SN As String
Dim bProgramState As Boolean = False
Dim ChannelAControlData As DDL_CHANNEL_CONTROL_DATA
Dim ChannelBControlData As DDL_CHANNEL_CONTROL_DATA
Dim ChannelADataLoggerBuffer() As Integer = New Integer(DATA_LOGGER_BUFFER_SIZE) {}
Dim ChannelBDataLoggerBuffer() As Integer = New Integer(DATA_LOGGER_BUFFER_SIZE) {}
Dim iDataValue As Integer
Dim cbLD As Integer = Marshal.SizeOf(iDataValue)
Dim ChannelACDLBPtr As IntPtr
Dim ChannelBCDLBPtr As IntPtr
Dim ChannelANumValuesReturned As Long = 0
Dim ChannelBNumValuesReturned As Long = 0
Dim iLoopCntr As Integer = 0
Dim bValueMissed As Boolean
Dim FrequencyHz As Long = 0
Dim MaxFreqRangeHz As Long = 0
Dim MinFreqRangeHz As Long = 0
```

```
Dim ErrorMessage As String = New String(ChrW(0), MAX_NUM_ERROR_MESSAGE_CHARS)
```

```
Private Function ConvertDLLStringToVBString(ByRef DLLString As String) As String
```

```
    Dim VBString As String
    Dim iCharCntr As Integer = 0
    Dim CharArray() As Char
```

```
    While (DLLString.Chars(iCharCntr) <> vbNullChar)
        iCharCntr = iCharCntr + 1
    End While
```

```
    CharArray = New Char((iCharCntr - 1)) {}
```

```
    iCharCntr = 0
    While (DLLString.Chars(iCharCntr) <> vbNullChar)
        CharArray(iCharCntr) = DLLString.Chars(iCharCntr)
        iCharCntr = iCharCntr + 1
    End While
```

```
    VBString = New String(CharArray)
```

```
    Return VBString
```

```
End Function
```

```
Private Sub CopyIntPtr_To_Logged_Data_Array(ByVal CDLBPtr As IntPtr, ByVal ChannelDataLogBuffer() As Integer, ByVal NumValuesReturned As Long)
```

```
    Dim iBuffIndex As Integer = 0
    Dim cbint As Integer = Marshal.SizeOf(iBuffIndex)
```

```

For iBuffIndex = 0 To (NumValuesReturned - 1)
    ChannelDataLogBuffer(iBuffIndex)(DATA_VALUE_INDEX) = Marshal.ReadInt32(CDLBPtr)
    CDLBPtr = New IntPtr(CDLBPtr.ToInt32() + cbint)
    ChannelDataLogBuffer(iBuffIndex)(MISSING_VALUE_INDEX) = Marshal.ReadInt32(CDLBPtr)
    CDLBPtr = New IntPtr(CDLBPtr.ToInt32() + cbint)
    ChannelDataLogBuffer(iBuffIndex)(MAX_VALUE_CLIPPED_INDEX) = Marshal.ReadInt32(CDLBPtr)
    CDLBPtr = New IntPtr(CDLBPtr.ToInt32() + cbint)
    ChannelDataLogBuffer(iBuffIndex)(MIN_VALUE_CLIPPED_INDEX) = Marshal.ReadInt32(CDLBPtr)
    CDLBPtr = New IntPtr(CDLBPtr.ToInt32() + cbint)

```

```

Next
End Sub

```

```

Private Sub MainForm_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

```

```

    Timer1.Interval = 1000
    Timer1.Stop()
    Iniciar_VOLTIMETRO()
    Timer1.Start()

```

```

End Sub

```

```

Private Sub MainForm_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.CancelEventArgs) Handles MyBase.Closing

```

```

    Timer1.Stop()
    Encerrar_VOLTIMETRO()
    Dim ddlStatus As Integer = DDL_SUCCESS

```

```

    If (ddlHandle <> 0) Then
        ddlStatus = DDL1M12_CloseDevice(ddlHandle)

```

```

        ddlHandle = 0

```

```

    End If
End Sub

```

```

Private Sub Ler_DADOS()

```

```

    If (ddlStatus = DDL_SUCCESS) Then
        ddlStatus = DDL1M12_GetDeviceChannelData(ddlHandle, ChannelACDLBPtr, ChannelANumValuesReturned, ChannelBCDLBPtr, ChannelBNumValuesReturned)
        CopyIntPtr_To_Logged_Data_Array(ChannelACDLBPtr, ChannelADataLoggerBuffer, ChannelANumValuesReturned)

```

```

        For iBuffIndex As Integer = 0 To (ChannelANumValuesReturned - 1)
            If (ChannelADataLoggerBuffer(iBuffIndex)(MISSING_VALUE_INDEX) <> False) Then
                bValueMissed = True
            End If

```

```

        Next

```

```

        If (ddlStatus = DDL_SUCCESS) And (ChannelANumValuesReturned > 0) Then
            ' Process Data
            lbValor.Text = Math.Round(ChannelADataLoggerBuffer(0)(0) / 1000000, 2)
        End If

```

```

    End If

```

```

    If (ddlStatus <> DDL_SUCCESS) Then
        ddlStatus = DDL1M12_GetErrorCodeString("EN", ddlStatus, ErrorMessage, MAX_NUM_ERROR_MESSAGE_CHARS)
        Timer1.Stop()
        MsgBox(ErrorMessage)

```

```

    End If
End Sub

```

```

Private Sub Iniciar_VOLTIMETRO()
    Dim iDataValueIndex As Integer = 0

    ChannelACDLBPtr = Marshal.AllocHGlobal(DATA_LOGGER_BUFFER_SIZE * (cbLD *
NUM_DATA_LOG_ELEMENTS))

    For iDataValueIndex = 0 To (DATA_LOGGER_BUFFER_SIZE - 1)
        ChannelADataLoggerBuffer(iDataValueIndex) = New Integer((NUM_DATA_LOG_ELEMENTS - 1)) {}
    Next

    ChannelBCDLBPtr = Marshal.AllocHGlobal(DATA_LOGGER_BUFFER_SIZE * (cbLD *
NUM_DATA_LOG_ELEMENTS))

    For iDataValueIndex = 0 To (DATA_LOGGER_BUFFER_SIZE - 1)
        ChannelBDataLoggerBuffer(iDataValueIndex) = New Integer((NUM_DATA_LOG_ELEMENTS - 1)) {}
    Next

    ddlStatus = DDL1M12_GetNumDevices(NumDevices)
    If ((ddlStatus = DDL_SUCCESS) And (NumDevices > 0)) Then
        If (NumDevices = 1) Then
            ddlStatus = DDL1M12_GetDeviceSerialNumber(0, SerialNumber,
MAX_NUM_SERIAL_NUMBER_CHARS)
            If (ddlStatus = DDL_SUCCESS) Then
                SN = New String(ConvertDLLStringToVBString(SerialNumber))
                ddlStatus = DDL1M12_OpenSpecifiedDevice(SN, ddlHandle)
            End If
        End If
    End If

    If ((ddlHandle <> 0) And (ddlStatus = DDL_SUCCESS)) Then
        ddlStatus = DDL1M12_InitDevice(ddlHandle)

        If (ddlStatus = DDL_SUCCESS) Then
            ddlStatus = DDL1M12_IsDeviceProgrammed(ddlHandle, bProgramState)

            If (ddlStatus = DDL_SUCCESS) Then
                If (bProgramState = False) Then
                    ddlStatus = DDL1M12_ProgramDevice(ddlHandle, "C:\DDL1M12.RBF")
                End If
            End If
        End If

        If (ddlStatus = DDL_SUCCESS) Then
            ddlStatus = DDL1M12_SetDeviceChannelState(ddlHandle, True, False)
        End If

        If (ddlStatus = DDL_SUCCESS) Then
            ChannelAControlData.bGainTimesTen = False
            ChannelAControlData.GainuVBit = GAIN_10MV_PER_BIT 'GAIN_200UV_PER_BIT
            ChannelAControlData.bCouplingState = False
            ChannelAControlData.bActiveState = True

            ChannelBControlData.bGainTimesTen = False
            ChannelBControlData.GainuVBit = GAIN_10MV_PER_BIT 'GAIN_200UV_PER_BIT
            ChannelBControlData.bCouplingState = False
            ChannelBControlData.bActiveState = False 'True
        End If
    End If

    ddlStatus = DDL1M12_StartDeviceDataLogging(ddlHandle, False, 1000, ChannelAControlData,

```

ChannelBControlData)

```
    If (ddlStatus <> DDL_SUCCESS) Then
        ddlStatus = DDL1M12_GetErrorCodeString("EN", ddlStatus, ErrorMessage,
MAX_NUM_ERROR_MESSAGE_CHARS)
        Timer1.Stop()
        MsgBox(ErrorMessage)
    End If
End Sub
```

```
Private Sub Encerrar_VOLTIMETRO()
    If (ddlStatus = DDL_SUCCESS) Then
        ddlStatus = DDL1M12_StopDeviceDataLogging(ddlHandle, CHANNEL_A)
    End If
```

```
    If (ddlStatus = DDL_SUCCESS) Then
        ddlStatus = DDL1M12_SetDeviceChannelState(ddlHandle, False, False)
    End If
```

```
    DDL1M12_CloseDevice(ddlHandle)
    ddlHandle = 0
```

```
    Marshal.FreeHGlobal(ChannelACDLBPtr)
    Marshal.FreeHGlobal(ChannelBCDLBPtr)
End Sub
```

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    Ler_DADOS()
End Sub
```

End Class