Log Horizontal with VBScript

Create a Logarithmic Frequency Scale using VBScripts

Logarithmic sweeps are not common in oscilloscopes. Even oscilloscopes with Fast Fourier Transform (FFT) capability usually display frequency spectra on linear horizontal scales. The creation of a logarithmically scaled FFT display is a useful and instructive application for Visual Basic Scripting (VBScript). The ability to perform custom math functions based on VB scripting or MatLab is a unique capability in LeCroy's WaveMaster™ series oscilloscopes.

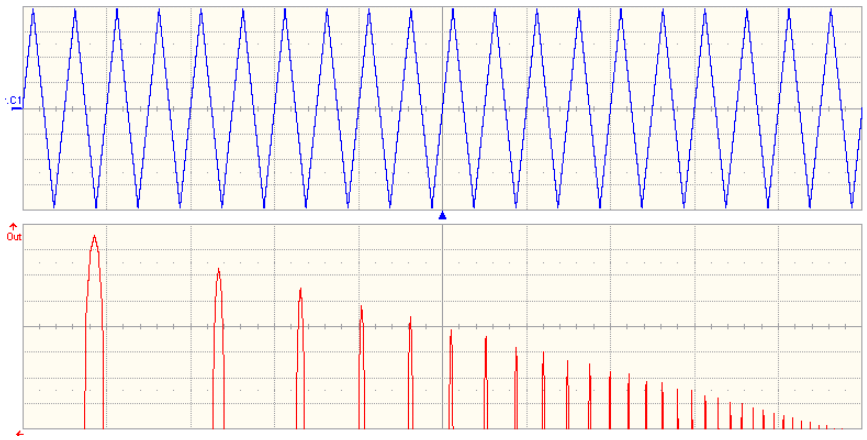

Figure 1 A logarithmic horizontal axis for an FFT spectral display created using a VBScript

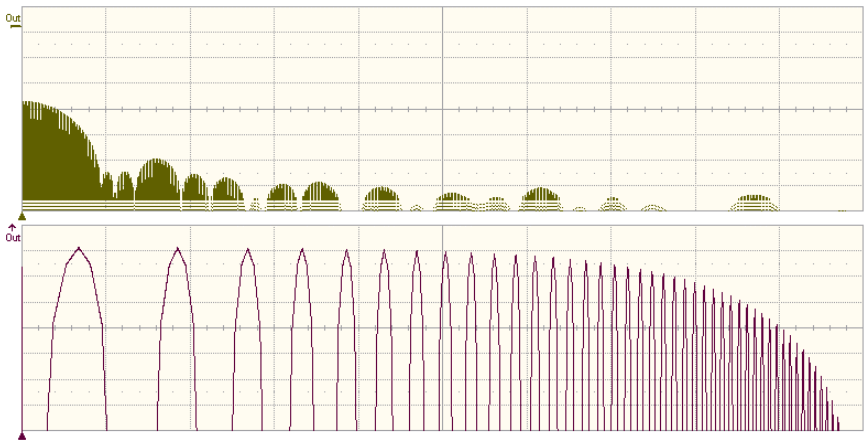

Figure 2 Another example of a log horizontal display in the lower trace. This is the output spectrum of a digital filter shown in the upper trace where it has a linear frequency scle

The VBscript is shown in its entirety on page 2. It reads in the data from the FFT and then spaces the data points logarithmically over the linear horizontal scale. The input data, containing the values of the FFT is called 'InResult'. There are two properties for this object. The first is the number of samples contained, InResult.Sample. The second are the data, 'InResult.DataArray (False)'. Data are read in as 16 bit signed integers (range 32767 to -32768) which is indicated by the Boolean expression 'False'. The principle part of the script is a For-Next loop which reads each data point and calculates where it should be located in the logarithmic display. If necessary, interpolation is used to add data points to fill in the display. Note is made of the point at which a compaction routine might be inserted if the data are spaced too closely. After each data point is properly positioned the data array is read out to the output data array, 'OutResult.DataArray'. The output data array is also unscaled integer data.

This is a very practical application
for custom function generation within the WaveMaster scope. The end result of this calculation is returned to the scope and displayed just like any other math function. Figure 2 Another example of a log horizontal display in the lower trace. This is the output spectrum of a digital filter shown in the upper trace where it has a linear frequency scale

```
' Example - create a logarithmic freqency scale.

' Input data
OutResult.Samples = InResult.Samples
startData = 0 : endData = OutResult.Samples
newNumPoints = endData - startData

ReDim newDataArray(OutResult.Samples)
unscaledData = InResult.DataArray(False)

EndOfData = Csng (endData)     ' Make a real number.
LastSample = endData - 1       ' Last sample to use
Log10 = Log(10)
Decades = 2.5  ' Number of frequency decades to show
ScaleFactor = endData/Decades  ' Needed to fit log scale on screen
OldLogPos = 1  ' Initialise position memory.

        For Sample = 1 To LastSample
        RSample = Csng(Sample)    ' Make a real number
        Y = unscaledData(Sample)  ' Input value
        ' Calculate position of sample on logarithmic scale.
        LogPos = endData + ScaleFactor*(log(RSample/EndOfData))/log10
        ' Calculate spacing from previous sample.
        DeltaLP = LogPos - OldLogPos : DeltaY = Y - OldY
        ' If spacing greater than 1 we need to interpolate.
                if DeltaLP > 1 then
                        for NewRelPos = 0 to DeltaLP
                        ' Calculate the horizontal position.
                        NewPos = NewRelPos + OldLogPos
                                if ((NewPos > 1) and (NewPos <=LastSample)) then
                                ' Calculate value of the interpolated sample.
                                newDataArray(NewPos) = OldY + (DeltaY * NewRelPos) / DeltaLP
                                end if
                        next
        ' We do not need to interpolate.
                else
                        if ((LogPos>1) and (LogPos<=LastSample)) then
        ' We would need to do some compaction if the peaks
        ' became too closely spaced.
                        newDataArray(LogPos) = Y
                        end if
                end if
        ' Preserve last calculated values.
        OldLogPos = LogPos : OldY = Y
        Next
' Set the output array equivalent to the calculated array.
OutResult.DataArray(False) = newDataArray
```