

## Content

This document describes how to use LightningChart (LC) SampleDataSeries series to plot fixed interval data in a MS Visual Studio 2008 .NET C# project. The procedures to interface with LC are similar in other Visual Studio versions.

Installing of LC to Visual Studio is described in another document found in installation folder.

## Prerequisites

Basic knowledge of C# and Visual Studio is assumed. LightningChart is installed and demo projects are found to be functional.

## Description

Chart displays sample data generated by signal generator. Chart must contain a SampleDataSeries series where the data is added by event handler connected to signal generator. Event handler receives data as double value array, which is given directly to series.

Signal generator has an UI, which can be used to edit produced waveform. In this example the signal generator is initialized in code and UI updated accordingly.

## SampleData series

SampleData series allows representing fixed-interval signal data. There's no need to give point X values when adding data, as the point X locations are automatically calculated with FirstSampleTimeStamp, SamplingFrequency and the index of the sample.

## Add LightningChart and SignalGenerator controls to Form

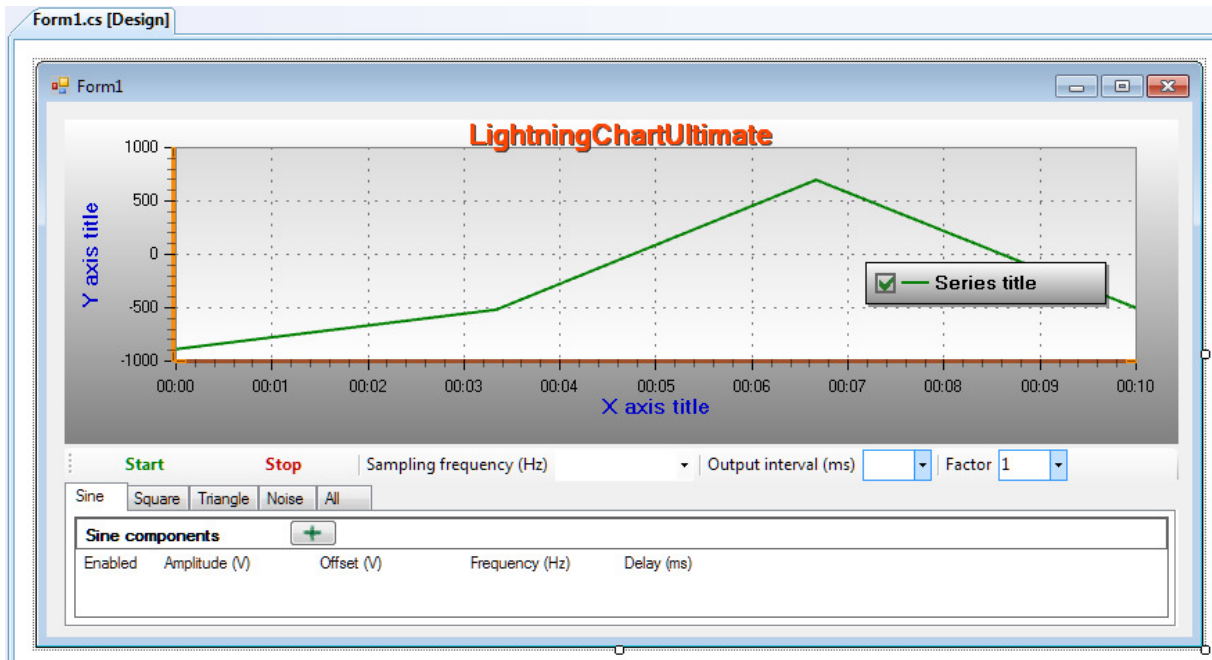
Create a Windows Forms application, which creates and opens main Form automatically. Find LightningChart control from the Toolbox and drag it on the form. Do same to SignalGenerator.

The view should look like following. Set LC Anchor property in Form editor to Top, Bottom, Left, Right and SignalGenerator Anchor to Bottom, Left, Right so they are positioned nicely and don't overlap on resize.

## Constructor

Call initializers in constructor.

```
public Form1()  
{  
    InitializeComponent();  
    InitializeChart();  
    InitializeSignalGenerator();  
}
```



## Initialize LightningChart control

Edit source code of Form1.cs (Right-click on Form1.cs in Solution Explorer and select "View Code") to add a single SampleDataSeries to LC. Here is a sample code to do that (which should be called in constructor).

```
private void InitializeChart()
{
    // Disable repaints caused by property changes
    lightningChartUltimate1.BeginUpdate();

    // Remove default point-line series
    lightningChartUltimate1.ViewXY.PointLineSeries.Clear();

    // Properties could be also set at Form editor
    lightningChartUltimate1.Name = "SampleDataSeries chart";

    // Hide legend box
    lightningChartUltimate1.ViewXY.LegendBox.Visible = false;

    // Add SampleData series with sample format as double (default).
    // Double format due easy sample data update on generated sample
    // data (see below)
    SampleDataSeries series = new
    SampleDataSeries(lightningChartUltimate1.ViewXY,
    lightningChartUltimate1.ViewXY.YAxes[0]);

    // Add the created series into SampleDataSeries list
    lightningChartUltimate1.ViewXY.SampleDataSeries.Add(series);

    // Allow automatic destruction of outscrolled data
    lightningChartUltimate1.ViewXY.DropOldSeriesData = true;

    // Allow chart drawing
    lightningChartUltimate1.EndUpdate();
}
```

### Initialize SignalGenerator

SignalGenerator produces sample data used by LC. Signal generator is initialized to produce 1 Hz sine wave samples at 1 kHz sampling rate with a random noise component. Here is a sample method to initialize signal generator. Naturally you can do the same initialization with Properties editor of Visual Studio.

```
private void InitializeSignalGenerator()
{
    // Set name
    signalGenerator1.Name = "Signal";
    // Clear sine waveform table
    signalGenerator1.WaveFormSines.Clear();
    // Clear random noise table
    signalGenerator1.WaveFormRandomNoises.Clear();
    // Set sampling frequency
    signalGenerator1.SamplingFrequency = 1000;
    // Set signal generator output interval
    signalGenerator1.OutputIntervalMs = 2;
    // Set thread type
    signalGenerator1.ThreadType = ThreadType.Timer;
    // This is master generator
    signalGenerator1.MasterGenerator = null;
    // Add some event listeners
    signalGenerator1.NewSignalPointsGenerated += new
SignalGenerator.NewSignalPointsGeneratedHandler(signalGenerator1_New
SignalPointsGenerated);

    signalGenerator1.Started += new
SignalGenerator.StartedHandler(signalGenerator1_Started);
    signalGenerator1.Stopped += new
SignalGenerator.StoppedHandler(signalGenerator1_Stopped);

    // Create sine component
    SineComponent sineComp = new SineComponent();
    sineComp.Amplitude = 10.0;
    sineComp.DelayMs = 0;
    sineComp.Frequency = 1;
    sineComp.Offset = 0;
    sineComp.Enabled = true;
    signalGenerator1.WaveFormSines.Add(sineComp);

    // Create noise component
    RandomNoiseComponent noiseComp = new RandomNoiseComponent();
    noiseComp.Amplitude = 1.0;
    noiseComp.Offset = 0;
    noiseComp.Enabled = true;
    signalGenerator1.WaveFormRandomNoises.Add(noiseComp);

    // Update signal generator ui from added components
    signalGenerator1.UpdateUIFromWaveFormComponents();
}
```

## LightningChart Ultimate - Real-time monitoring with SampleDataSeries

---

Signal generator produces events which should be used to initialize, display and end sampling display. Here are the methods referenced in signal generator initialize method.

```
void signalGenerator1_Started()
{
    // Prepare chart for property updates, in batch
    lightningChartUltimate1.BeginUpdate();

    // Update title
    lightningChartUltimate1.Title.Text = string.Format("LightningChart
    Ultimate, sfreq = {0} kHz",
    (double)signalGenerator1.SamplingFrequency / 1000.0);

    // Set x axis range
    lightningChartUltimate1.ViewXY.XAxis.SetRange(0, 10);

    // Set scroll position to the beginning
    lightningChartUltimate1.ViewXY.XAxis.ScrollPosition = 0;

    // Scrollmode to scrolling
    lightningChartUltimate1.ViewXY.XAxis.ScrollMode =
    XAxisScrollMode.Scrolling;

    // Chart has one Y axis ready to go. Just set the range
    lightningChartUltimate1.ViewXY.YAxes[0].SetRange(-10, 10);

    // Disable zooming
    lightningChartUltimate1.ViewXY.ZoomPanOptions.ZoomEnabled = false;

    // Update series properties
    SampleDataSeries series =
    lightningChartUltimate1.ViewXY.SampleDataSeries[0];
    series.Clear();
    series.FirstSampleTimeStamp = 1.0 /
    signalGenerator1.SamplingFrequency;
    series.SamplingFrequency = signalGenerator1.SamplingFrequency;
    series.MouseInteraction = false;

    // Allow chart to update view
    lightningChartUltimate1.EndUpdate();
}
```

```
void signalGenerator1_Stopped()
{
    lightningChartUltimate1.ViewXY.XAxis.ScrollMode =
    XAxisScrollMode.None;

    // Prepare chart for property update
    lightningChartUltimate1.BeginUpdate();

    // Allow zooming
    lightningChartUltimate1.ViewXY.ZoomPanOptions.ZoomEnabled = true;

    // Set view to data end
    double dMin = lightningChartUltimate1.ViewXY.XAxis.ScrollPosition -
    (lightningChartUltimate1.ViewXY.XAxis.Maximum -
    lightningChartUltimate1.ViewXY.XAxis.Minimum);

    lightningChartUltimate1.ViewXY.XAxis.SetRange(Math.Max(0, dMin),
    lightningChartUltimate1.ViewXY.XAxis.ScrollPosition);

    // Allow mouse interaction with series
    lightningChartUltimate1.ViewXY.SampleDataSeries[0].MouseInteraction
    = true;

    // Allow chart to update view
    lightningChartUltimate1.EndUpdate();
}

void signalGenerator1_NewSignalPointsGenerated(ref double[][] samples,
double firstSampleTimeStamp)
{
    // This event handler is invoked when new data points are generated.
    // 'samples' is a multi-channel input. In this example, only the
    // first channel (samples[0]) is used.

    if (samples.Length == 0)
        return;

    // Prepare chart for property update
    lightningChartUltimate1.BeginUpdate();

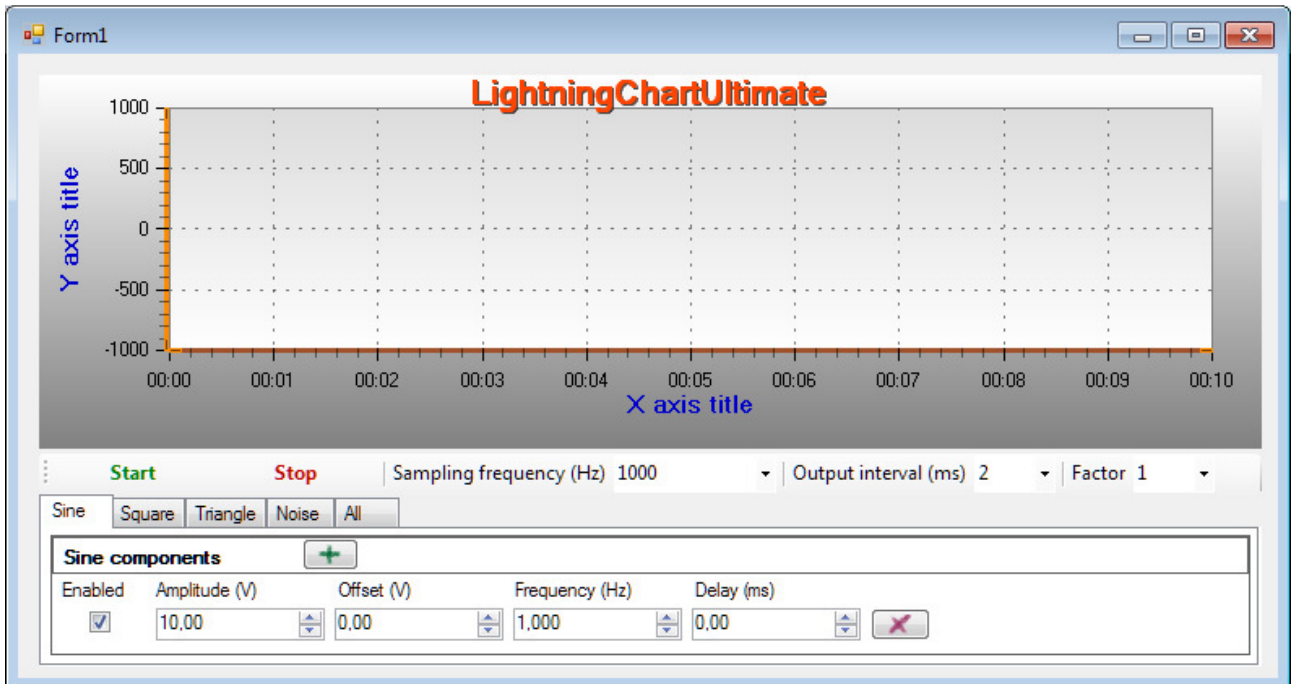
    // Add double precision samples
    lightningChartUltimate1.ViewXY.SampleDataSeries[0].AddSamples(
    samples[0], false);

    // Update scroll position to last sample time stamp
    lightningChartUltimate1.ViewXY.XAxis.ScrollPosition =
    firstSampleTimeStamp + (double)(samples[0].Length - 1) /
    signalGenerator1.SamplingFrequency;

    // Allow chart to update view
    lightningChartUltimate1.EndUpdate();
}
```

## Result

Compile and run the project. The result should look like this.



Press Start button to start signal generator. LC should be drawn by sine curve with a random noise component.

