

LightningChart Basic - Plotting timer-generated real-time data

Content

This document describes how to use LightningChart Basic (LC) PointLineSeries series to plot fixed-interval data in a MS Visual Studio 2010 .NET C# Windows Forms project. Installing of LC to Visual Studio is described in another document, found in installation folder or web site.

Prerequisites

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

Description

A simple strip-chart recorder application. Chart displays timer-generated data. When chart is full, it starts scrolling and destroying out-scrolled data.

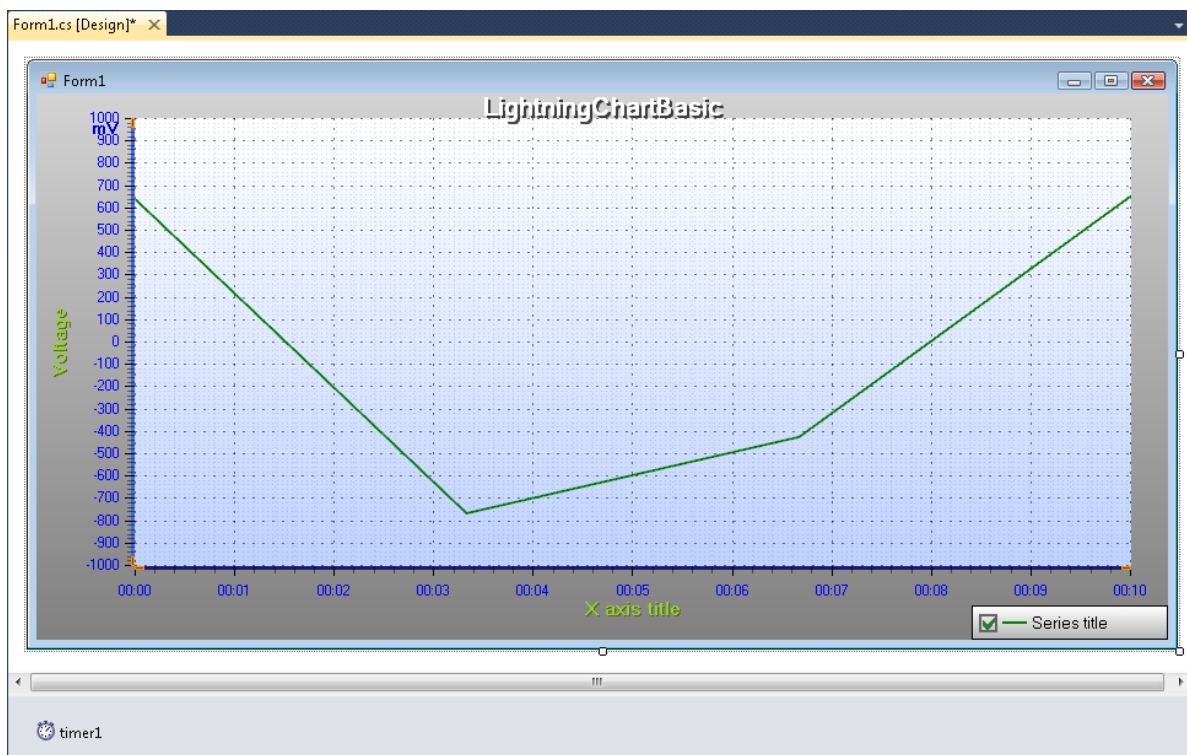
Add a LightningChart control and a Timer to the form

Create a Windows Forms application, which creates and opens main Form automatically. Set Target Framework to .NET Framework 3.5 from project properties (Compile->Advanced Compile Options -> Target framework).

Find LightningChart control from the Toolbox and drag it on the form.

Find Timer from the Toolbox and drag it to the form, as well.

The form should look like the following:



LightningChart Basic - Plotting timer-generated real-time data

Initialize LightningChart control

Edit source code of Form1.cs (Right-click on Form1.cs in Solution Explorer and select “View Code”, or press F7). Here is a sample code to show how to set up chart to display some incoming data. This could be also made with Properties window, in most parts, but here it’s all made by code.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Arction.LightningChartBasic;
using Arction.LightningChartBasic.Series;

namespace LCBasicTimer
{
    public partial class Form1 : Form
    {
        //Data interval in seconds
        private double dataInterval;
        private int pointCount = 0;
        Random rand = new Random();

        public Form1()
        {
            InitializeComponent();

            InitializeChart();

            dataInterval = timer1.Interval / 1000.0;
        }

        private void InitializeChart()
        {
            //Disable repaints
            lightningChartBasic1.BeginUpdate();

            //The chart has one Y axis and PointLineSeries to go.

            lightningChartBasic1.YAxes[0].SetRange(0, 100);
            lightningChartBasic1.YAxes[0].Title.Text = "Voltage (V)";
            lightningChartBasic1.YAxes[0].Units.Visible = false;

            //Set 10 second X range
            lightningChartBasic1.XAxis.SetRange(0, 10);
            lightningChartBasic1.XAxis.Title.Text = "Time";
            lightningChartBasic1.XAxis.Units.Visible = false;

            //Set real-time scrolling mode. It occurs whenn ScrollPosition reaches end of
            the X axis.
            lightningChartBasic1.XAxis.ScrollMode = XAxisScrollMode.Scrolling;
            lightningChartBasic1.XAxis.ScrollPosition = 0;
        }
    }
}
```

LightningChart Basic - Plotting timer-generated real-time data

```
//Allow destruction of out-scrolled data, to keep RAM reservation minimal
lightningChartBasic1.DropOldData = true;

//Set some point-line series properties
//Don't use mouse interaction in real-time monitoring application
lightningChartBasic1.PointLineSeries[0].MouseInteraction = false;
lightningChartBasic1.PointLineSeries[0].LineStyle.Color = Color.Red;

//Enable repaints, and repaint
lightningChartBasic1.EndUpdate();
}
}
}
```

Initialize timer and add the Tick event handler

Edit `timer1` object with Properties editor.

```
Interval = 20
Enabled = true.
```

Define event handler for Tick event, by double-clicking the `timer1` object in the form designer.

```
private void timer1_Tick(object sender, EventArgs e)
{
    //Disable repaints
    lightningChartBasic1.BeginUpdate();

    //Append one point in the point-line series
    pointCount++;

    double x = (double)pointCount * dataInterval;
    double y = 100.0 * rand.NextDouble();

    SeriesPoint[] pointsArray = new SeriesPoint[1];
    pointsArray[0].X = x;
    pointsArray[0].Y = y;

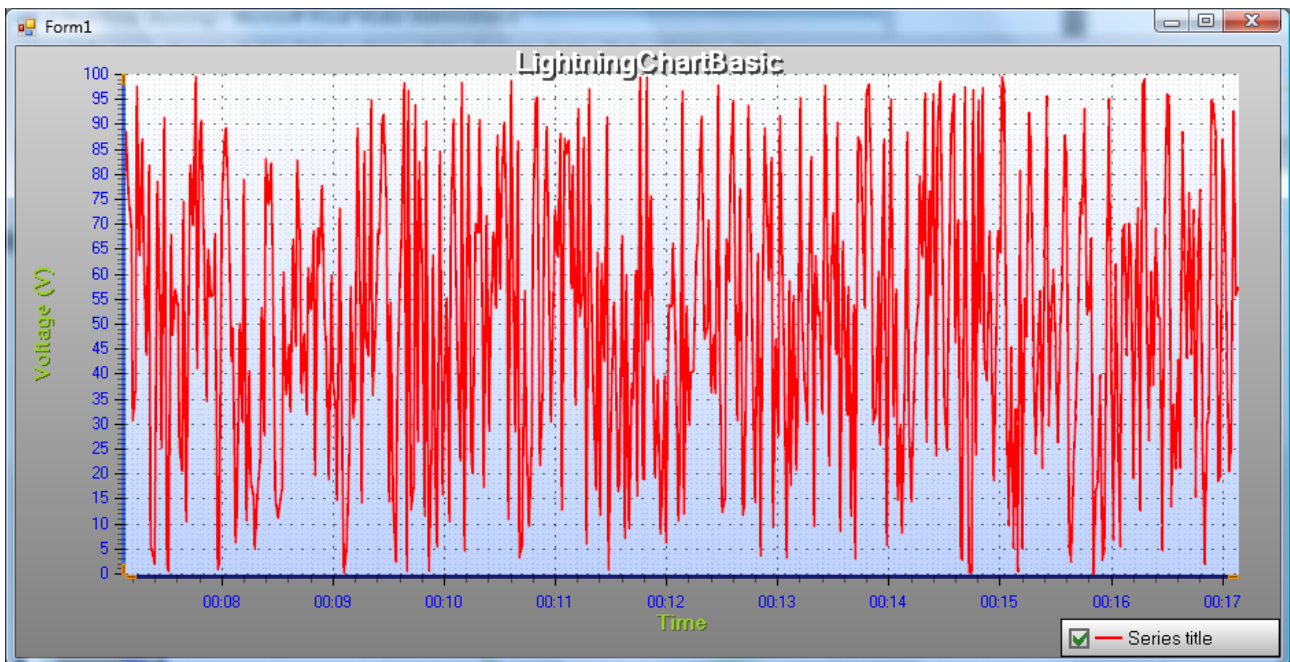
    //Add the point into the end of first point-line series
    lightningChartBasic1.PointLineSeries[0].AddPoints(pointsArray, false);

    //Set real-time scrolling position
    lightningChartBasic1.XAxis.ScrollPosition = x;

    //Enable repaints, and repaint
    lightningChartBasic1.EndUpdate();
}
```

LightningChart Basic - Plotting timer-generated real-time data

Compile and run the project. The result should look like this, and the chart scrolls when it gets full.



Note! In this example, data is added with fixed timer interval. Timer invoke interval varies, in fact, and causes some delay to happen in long monitoring measurements. To fix it, get the X values from system clock `TimeSpan.FromTicks(DateTime.Now.Ticks).TotalSeconds`, or update the chart by measurement hardware driven control.