Example 6: A Simple Realtime Display

Example 6 is similar to Example 5b, but this time we use the obtained profiles to show a very simple realtime display of ProfileDataPoints on a Windows form:

The scanning thread forces the scan head into TimedSyncMode and proceeds to poll for Profiles, just like in the previous examples. Meanwhile, in the UI, a standard Charting component (System.Windows.Forms.DataVisualization.Charting.Chart) is updated every 80 milliseconds with data from a Profile. The important part is the locking: both the main thread and the scan thread have access to the variable latestProfile, so in order to avoid a conflict, we have to lock access to the variable for the short time it takes to write new data into it.

```
        using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using JoeScan.JCamNet;
using Timer = System.Windows.Forms.Timer;

namespace Examples
{
    public partial class Form1 : Form
    {
        private Thread scanThread;
        private volatile bool isRunning;
        private Scanner scanner;
        private Profile latestProfile;

        public Form1()
        {
            InitializeComponent();
            scanner = SetupScanner();
            Timer liveUpdateTimer = new Timer {Interval = 80};
            liveUpdateTimer.Tick += new EventHandler(UpdateLiveProfiles);
            liveUpdateTimer.Start();
            StartScanning();
        }

        private void UpdateLiveProfiles(object sender, EventArgs e)
        {
            if (latestProfile != null)
            {
                Series series = LiveView.Series[0];
                series.Points.Clear();
                // to prevent that the scanning thread overwrites
                // latestProfile, we lock around the code that reads the variable
                lock (this)
                {
                    LaserOnTimeLabel.Text = String.Format("{0} ms", latestProfile.LaserOnTime/1000.0);
                    foreach (var pt in latestProfile)
                    {
                        series.Points.AddXY(pt.X, pt.Y);
                    }
                }
            }
        }

        private void ScanThreadMain()
        {
            scanner.EnterTimedSyncMode();
```

```
            while (isRunning)
            {
                List<Profile> received = scanner.GetQueuedProfiles(1);
                if (received.Count > 0)
                {
                    // to prevent overwriting the variable while it's being
                    // read in the GUI thread, we need to lock
                    lock (this)
                    {
                        latestProfile = received[0];
                    }
                }
            }

            scanner.ExitSyncMode();
        }

        private void StartScanning()
        {
            if (scanThread == null || !scanThread.IsAlive)
            {
                scanThread = new Thread(new ThreadStart(this.ScanThreadMain))
                {
                    IsBackground = true,
                    Name = "ScanThreadMain",
                    Priority = ThreadPriority.Highest
                };
                isRunning = true;
                scanThread.Start();
            }
        }

        private void StopScanning()
        {
            isRunning = false;
            if (scanThread != null)
            {
                scanThread.Join(100);
                scanThread = null;
            }
        }

        private Scanner SetupScanner()
        {
            // for clarity, no error checking here
            Scanner s = Scanner.Connect(new IPAddress(new byte[] { 192, 168, 1, 150 }));
            using (TextReader parametersReader = File.OpenText("param.dat"))
            {
                string parametersString = parametersReader.ReadToEnd();
                s.SetParameters(parametersString, true);
            }
            return s;
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            StopScanning();
        }
    }
}
```