Example 7: High-Speed Scanning with Multiple Heads

Example 7 shows three important concepts:

- how to deal with multiple scan heads,
- how to interleave calls for maximum scan speed, and
- how to use pulse sync mode.

Using multiple heads is as simple as keeping a connection open (i.e., retaining a Scanner object) for each of them. If the objects are kept in an enumerable collection, it is very convenient to loop over them with a foreach statement. The factory function Scanner.Connect() will give you back an array of scanners, but be aware that you must check that you connected to as many heads as you requested. In the example code, we skip this check.

If you have connections to more than six scanners, the total roundtrip for requesting and reading a profile from each one can take a while. Fortunately, there's an easy way to parallelize requests and reads from groups of scanners. The key is to send all the requests at once and then read all the profiles back at once. This causes all the scanners' processing time and network travel time to happen simultaneously.

In the example, we execute two foreach loops – one for requesting profiles and one for reads.

In the code, we also demonstrate how to use Pulse Sync Mode. This mode allows for a master head to drive slave heads. The slave heads wait for a StartIO pulse on the cable. The code designates the first head the master and the second a slave.

```
     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 JoeScan.JCamNet;

namespace Examples
{
    public partial class Form1 : Form
    {
        private Thread scanThread;
        private volatile bool isRunning;
        private Scanner[] scanners;

        public Form1()
        {
            InitializeComponent();
            scanners = SetupScanners();
        }

        private void ScanThreadMain()
        {
            foreach (var scanner in scanners)
            {
                // reset encoders
                scanner.SetEncoder(0);
                // EnterPulsSyncMode() will make all heads trigger on receiving a
                // Start Scan pulse, that is emitted by the "master" head.
                scanner.EnterPulseSyncMode();
            }
            // sleep for a millisecond to give all heads to get ready
            Thread.Sleep(1);
            // and tell the "master" head to send out a pulse train every second
            scanners[0].StartPulseMaster(1000 * 1000);

            // this is called on a thread, so it can't manipulate objects on the UI thread, hence the
            // need to use Invoke()
            this.Invoke(new Action(() => this.textBox.Clear()));
            this.Invoke(new Action(() => this.textBox.AppendText("Scanning Started.\n")));

            while (isRunning)
            {
                // This is the recommended way to parallelize the readout from multiple scanners.
```

```
            // first, request profiles
            foreach (var scanner in scanners)
            {
                scanner.BeginGetQueuedProfiles(1);
            }
            // then, read them out
            foreach (var scanner in scanners)
            {
                List<Profile> received = scanner.EndGetQueuedProfiles();
                if (received.Count > 0)
                {
                    // we need to make temp copies because of the way
                    // captured variables work with lambda expressions
                    int tempShort = scanner.CableID;
                    int tempInt = received[0].Location;
                    this.Invoke(new Action(() =>
                        this.textBox.AppendText(String.Format
                        ("Scanner {0}: Profile measured at Encoder Value {1}\n",
                        tempShort, tempInt))));
                }
            }
            // Keep processing time in this loop to a minimum if you want maximum scan speed.
            // Key to a fast system is to only gather the data in this loop, and let another thread
            // (e.g. your main GUI thread) do heavy processing asynchronously. You must use thread
            //  safe methods of exchanging data or interacting with the UI.
        }
        // and make sure every scanner is out of sync mode again
        foreach (var scanner in scanners)
        {
            scanner.ExitSyncMode();
        }
    }

    private void StartButton_Click(object sender, EventArgs e)
    {
        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 StopButton_Click(object sender, EventArgs e)
    {
        StopScanning();
    }

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

    private void QuitButton_Click(object sender, EventArgs e)
    {
        StopScanning();
        Close();
    }

    private Scanner[] SetupScanners()
    {
```

```
            // for clarity, no error checking here
            Scanner[] scanners = Scanner.Connect(new IPAddress(new byte[] {192, 168, 1, 150}),
                new short[] {0,1});
            using (TextReader parametersReader = File.OpenText("param.dat"))
            {
                string parametersString = parametersReader.ReadToEnd();
                foreach (var scanner in scanners)
                {
                    scanner.SetParameters(parametersString, true);
                }
            }
            return scanners;
        }
    }
}
```

The running application from Example 7 looks like this: