USB Logic Analyzer

Shom Bandopadhaya
Advisor: Dr. James Irwin, Jr.

Final Report
Senior Capstone Project

Bradley University
Department of Electrical and Computer Engineering

Abstract

A logic analyzer displays logic level of digital signals. A logic analyzer differs from an oscilloscope which is a more powerful instrument but typically only has a few channels. This project seeks to continue work on a digital logic analyzer which has sixteen channels. The logic analyzer will display three logic levels: low, high, and indeterminate. The sixteen channels are sampled by an external conditioning hardware called a POD. The POD interfaces to the computer using Universal Serial Bus [USB] protocol. The graphical user interface [GUI] on the computer will display the possible three states of the sixteen lines, and allow the user to interact with and interpret the data. The user is able to select what lines are displayed, choose lines to combine into a data bus, zoom in on a particular segment of the data, and save the data in numeric or image format.

Table of Contents

I. Introduction

A logic analyzer is a measurement instrument that allows an user to check logic levels at multiple parts in a circuit. A logic analyzer is similar to an oscilloscope in that regard, but an oscilloscope is capable of measuring both analog and digital signals whereas a logic analyzer only displays digital signals. The idea behind creating a PC based USB Logic Analyzer is to devise an economically feasible option to provide logic analyzers to all lab stations in the Electrical Engineering Department's Junior Laboratory. The goal of the project is to create a sixteen channel digital logic analyzer that can sample data at a minimum speed of 100MHz and display one of three possible logic levels of low, indeterminate, or high. The Graphical User Interface [GUI] allows the user to select which channels are being displayed, what segment of data is being displayed and combine selected channels into a data bus. This project is primarily concerned with developing the C# software that runs on a personal computer.

II. Functional Description

The USB Logic Analyzer uses an external PC board to sample sixteen channels of data. The PC board or POD has high speed comparators that compare the inputs against a TTL or CMOS reference voltage. The outputs of the comparators are fed into the Opal Kelly XEM3001 FPGA board. The FPGA processes the data and stores the data in an internal First In First out [FIFO] buffer. The FIFO buffer is accessed by an application written in C++ (the interface API provided by Opal Kelly for the board is written for C++) from the host PC. The application then parses the data into discreet logic lines and prepares it for the main core application written in C#. The C# application takes the data and displays the data as discreet logic lines on a GUI. The GUI allows the user to perform several tasks such as selecting the lines to be displayed, zooming in on a particular piece of data, selecting lines that are combined into a data bus, etc.

This whole process is visually represented in Figure 2.1. The analog inputs go into the POD which interfaces with the FPGA board and then communicates with the host PC over USB. The scope of this project is the software on the PC. Everything other than the FPGA board and the POD are part of this project, this can be seen in Figure 2.2

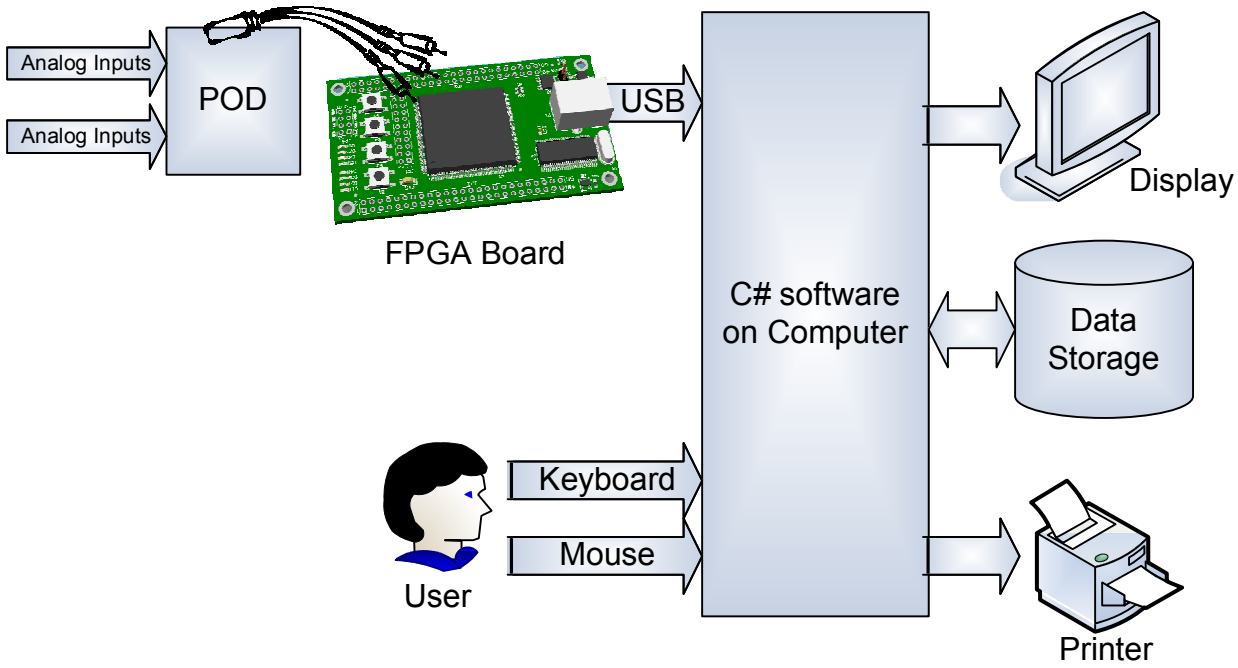


Figure 2.1 Functional Diagram of the USB Logic Analyzer

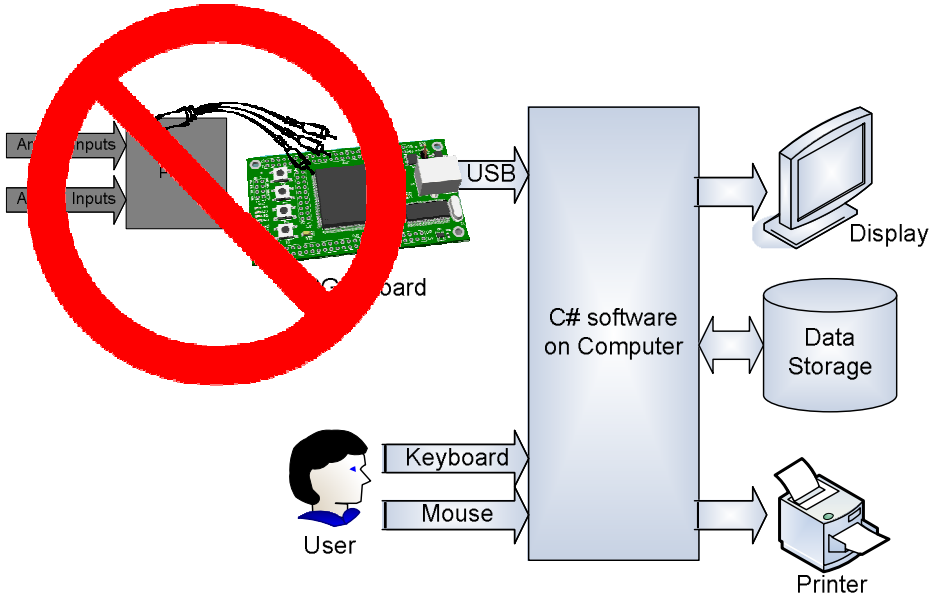


Figure 2.2 Functional Diagram showing the scope of the project

However, the software for the USB Logic Analyzer was completed ahead of schedule, with the remaining time the FPGA board was interfaced to the system so the USB communication could be verified. The final completed system can be seen in Figure 3.1. The FPGA board simulates real data in the absence of the POD. Data generated by a binary counter is saved to the FIFO buffer and then that information is acquired by the main core software.

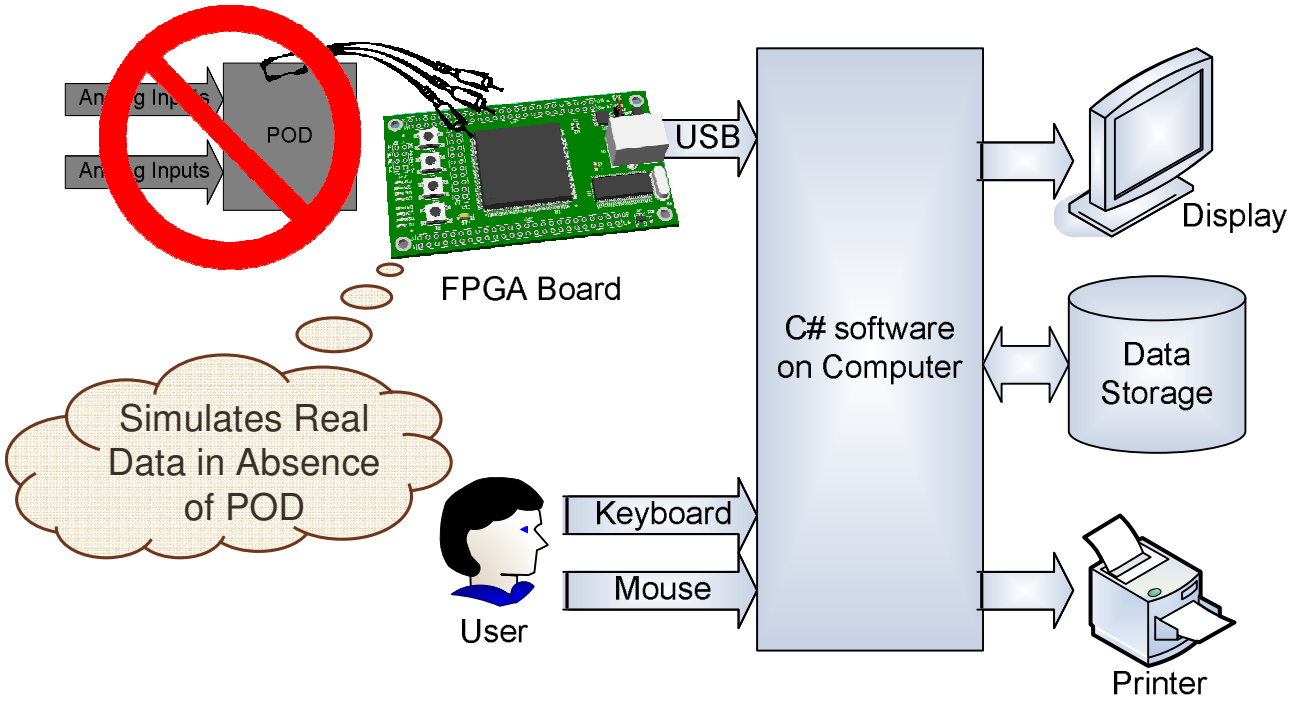


Figure 3.1 Functional Diagram demonstrating the final integration of the project

III. Project Goals

After having looked at the scope of the project the software goals can be concretely summarized as:

- Display using NPlot graphing library
- Plot acquired data as discreet logic signals
- Combine data lines into data bus
- Efficient data processing

- Print/Save data capture

Although the hardware is not a part of this project for reference the hardware goals can be concretely summarized as:

- Design 16 channel PC board which:
  - Works with CMOS or TTL logic levels
  - Samples at upwards of 100 MHz
  - Acquires data based on Trigger Event
  - Interfaces with XEM 3001 FPGA board
- Create VHDL code to interface FPGA board to PC

IV. System Block Diagram

The system as a whole has been described by the Functional Description section. The system can be broken down into smaller sections by separating them into functional blocks. The System Block Diagram, Figure 4.1, represents the whole system as smaller interconnected blocks. The function of each of these blocks will be discussed along with the design approach and the software flowchart.

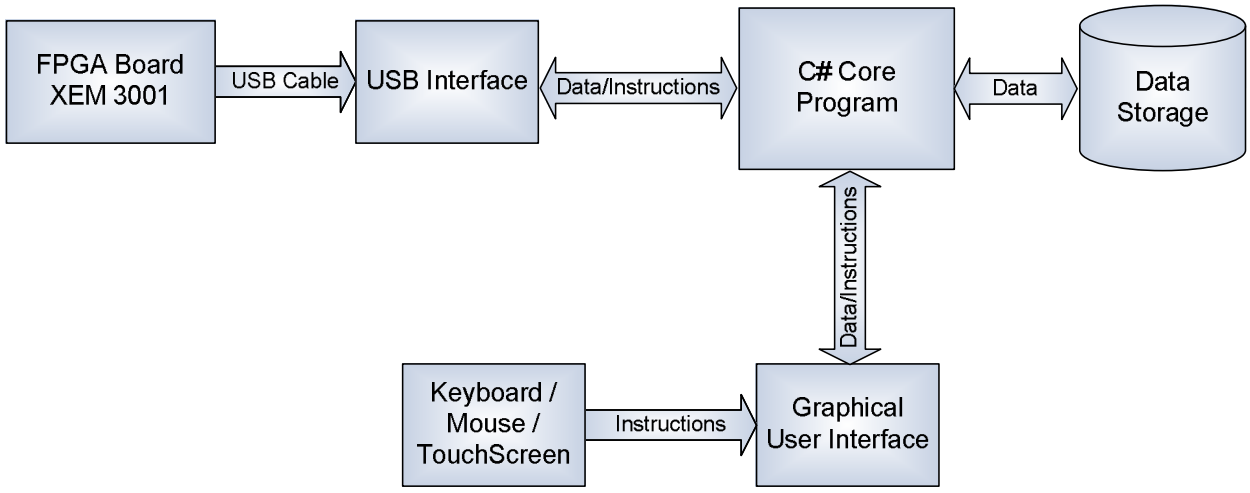


Figure 4.1 Overall System Block Diagram

IV. a. FPGA Board

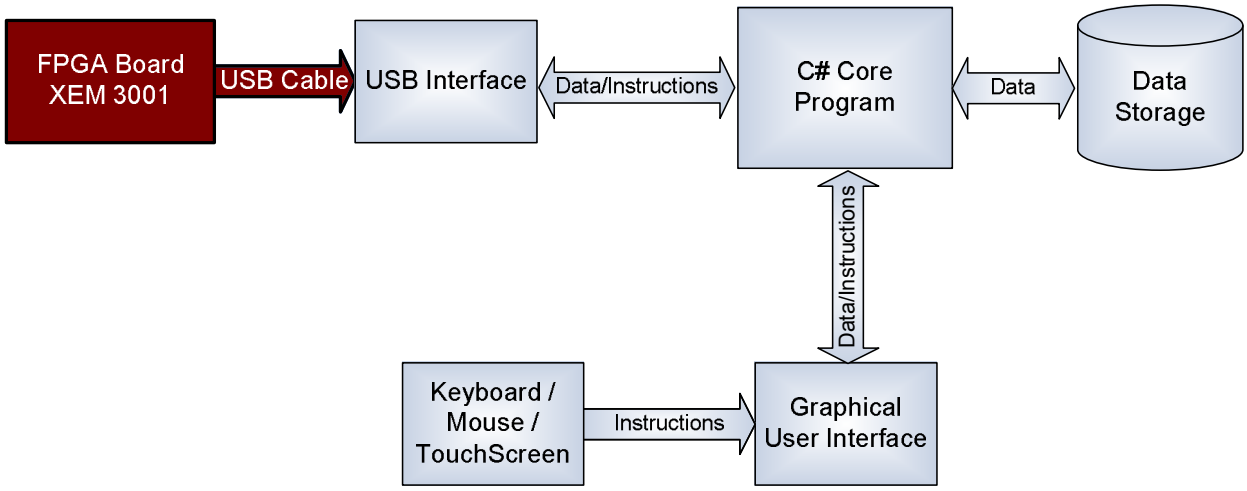
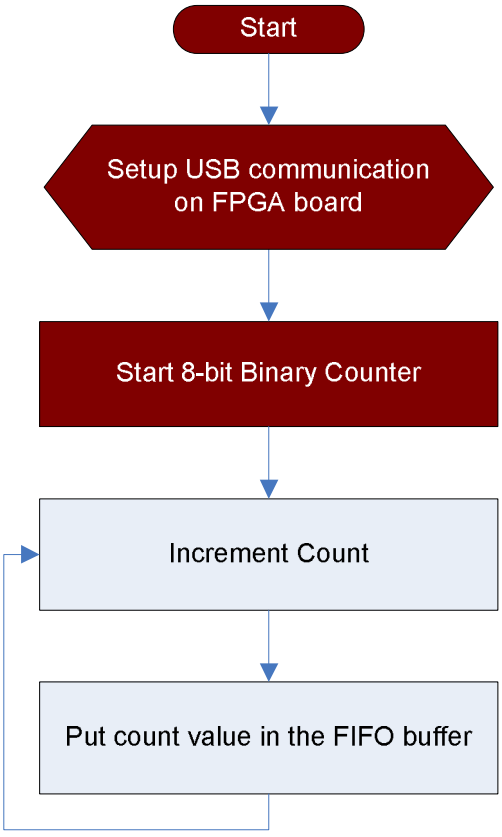


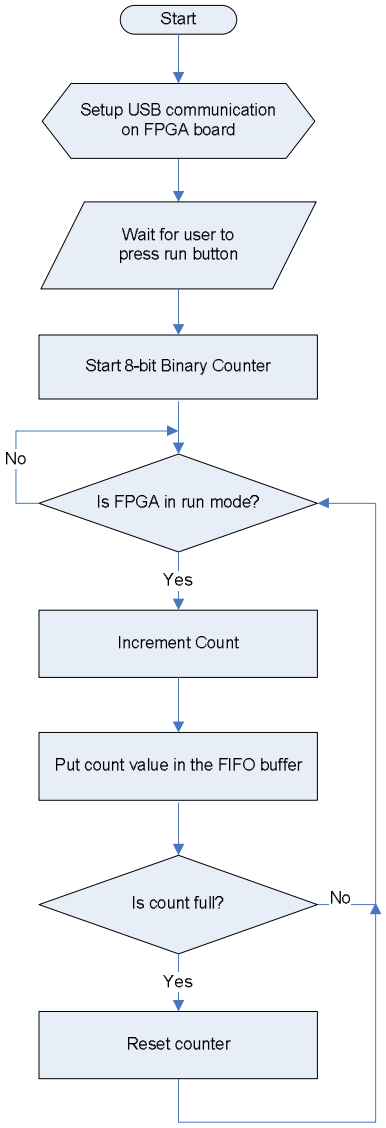
Figure 5.1 FPGA Board [XEM 3001] in System Block Diagram

The Opal Kelly XEM 3001 board contains a Xilinx Spartan-3 FPGA chip, an onboard Phase Lock Loop [PLL] chip, an onboard Voltage Controlled Oscillator [VCO], and an USB communication chip. The board natively runs at 48 MHz, but using the PLL and VCO it can be made to operate at 400 MHz. The operating frequency can be changed instantaneously by changing the PLL divider value using Opal Kelly's front end GUI called FrontPanel.

Figure 5.1 highlights how the FPGA board fits in with the whole system. The biggest challenge was to setup the FPGA/PC interface. Opal Kelly provides a library that instantiates VHDL blocks through port maps which the user's code can access to communicate with the PC. The inclusion and execution of these instantiated blocks is a difficult task. Opal Kelly provides these *.ngc files which are just the precompiled binaries of these blocks, that need to be copied to the project directory and they are often not detected when the project is compiled. Once the interface is setup it defines endpoints for the PC to connect to. For example, pin 78 on the chip is defined as the "wireIn" pin, on which the FPGA expects to receive instructions. When the PC connects it looks for "wireIn" and then sends the instructions on that pin.

After the FPGA can communicate with the PC, VHDL programs can be loaded and run. In this case an eight bit binary counter loaded and started. This binary counter is incremented and then the resulting value is placed in the FIFO buffer to be read by the PC. Essentially each bit of the eight bit binary counter corresponds to a channel that the FPGA would sample if a POD was present. Figure 6.1 (a) demonstrates the process that the FPGA performs; the highlighted section represents the iterative process. The VHDL code for this process can be seen in Appendix V.





(a) Comprehensive flowchart (b) Detailed flowchart

Figure 6.1 FPGA software flowchart

IV. b. USB Interface

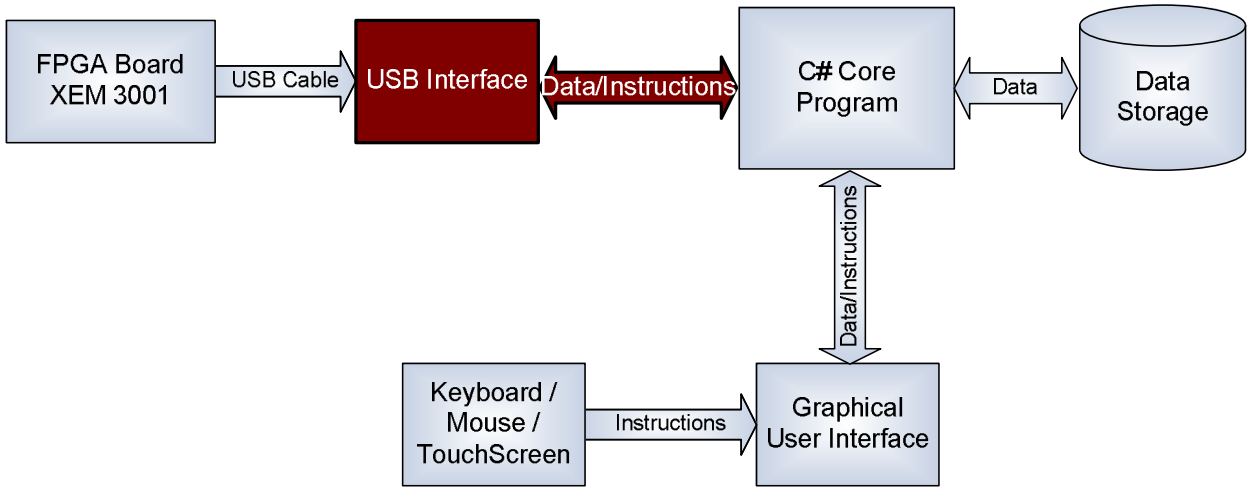


Figure 7.1 USB Interface in System Block Diagram

The USB Interface on the PC is analogous to the communication interface on the FPGA as discussed earlier. The USB Interface program [OKPipeRead] written in C++ (Appendix IV) utilizes the API provided by Opal Kelly to connect to the endpoints that have been defined on the FPGA using the Opal Kelly VHDL block instantiations. First, the program needs to be executed by the C# main core. Once the program is executed, an object of the board is created and assigned to the currently connected board. Then the endpoints that have been predetermined are connected from the PC to the FPGA. After this setup process is completed the PC reads the FIFO buffer on the FPGA. This buffer is then parsed into discreet logic lines, by isolating each bit through masking and shifting, and assigning it to its relevant channel. This data is now directly used by the C# main core program for further processing and manipulation.

A simplified logic flowchart can be seen in Figure 8.1. The section that is highlighted emphasizes the data processing section of the code. Based on how many channels worth of information is being transmitted (in this case sixteen) and based on how many points are being sampled in (by default 512) this is the section of code that will have variable execution time.

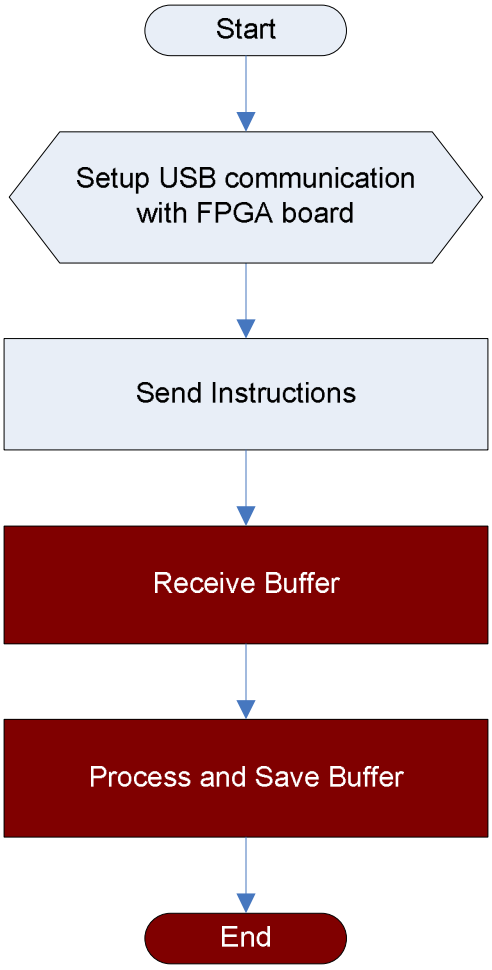


Figure 8.1 USB Interface software flowchart

IV. c. C# Main Core

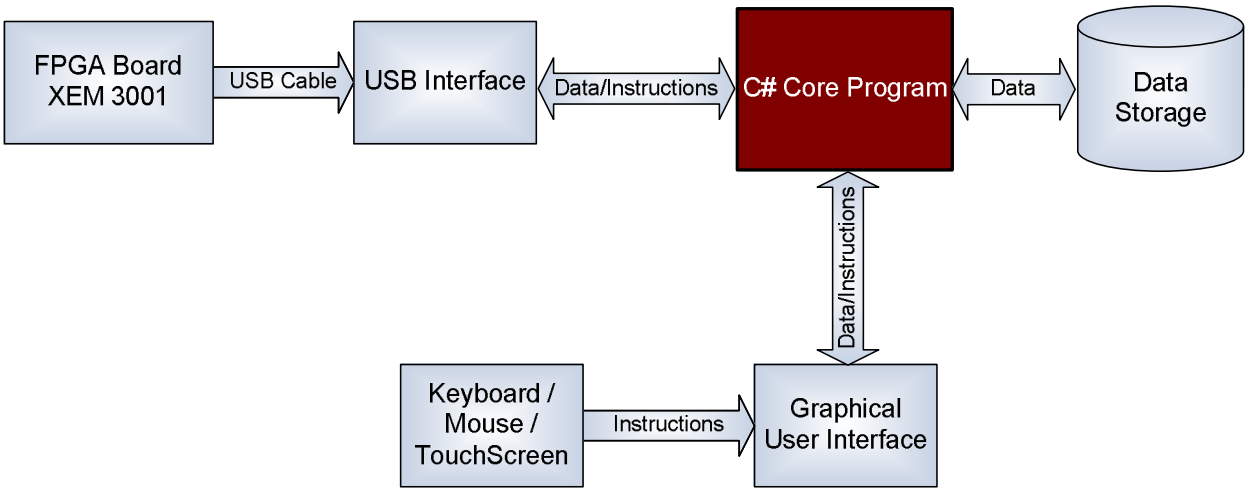


Figure 8.2 C# Main Core Program in System Block Diagram

The C# Main Core is the brain of the whole system. It receives and sends instructions to and from all the other blocks, and based on the input it receives it processes data that it either acquires or has previously acquired. Figure 8.2 shows how the Main Core is the central block of the whole system. Figure 9.2 shows the three main tasks that the Main Core performs. The Main Core is driven by user input that is obtained through the GUI. Based on what the user input is, the Main Core either invokes the USB Interface to acquire new data from the FPGA, or displays the data based on the user's manipulation, or saves/loads the data that is currently being displayed.

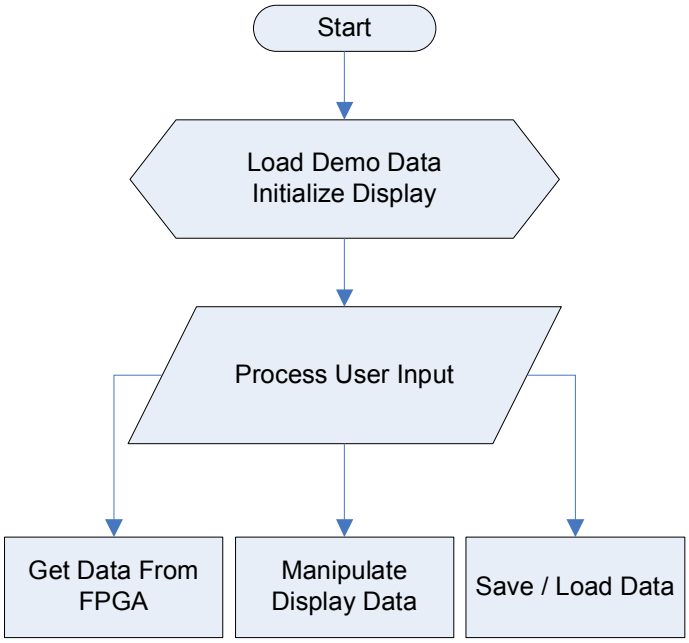


Figure 9.1 C# Main Core flowchart

There are several complex processes that are involved in all three of the main tasks that are performed by the Main Core. The details of the functions that enable these processes can be found by looking at the HTML documentation generated by doxygen (Appendix VII). The documentation was created by extracting inline code documentation which can be found in Appendix I.

IV. d. Data Storage

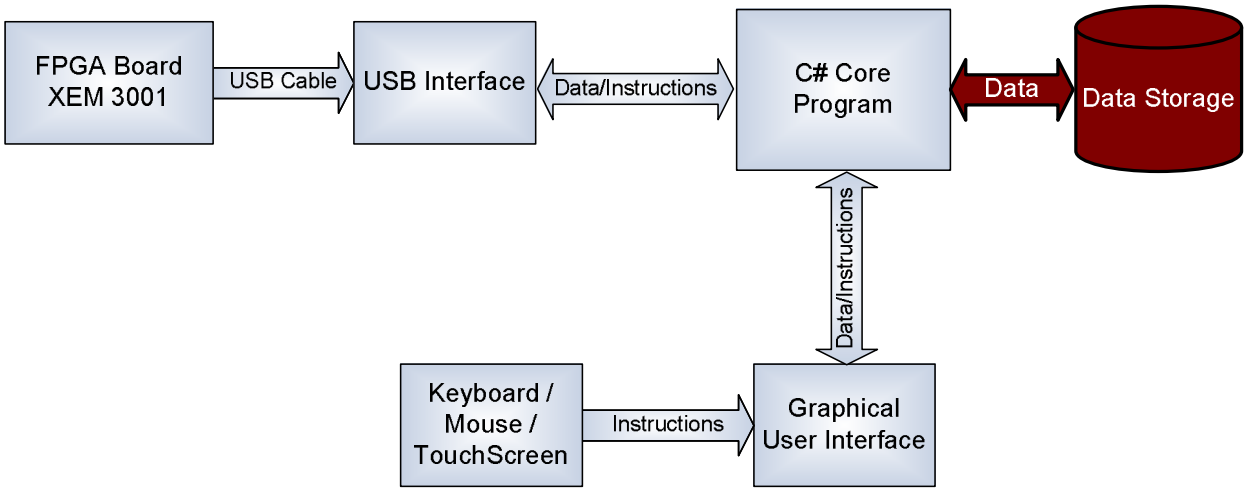


Figure 10.1 Data Storage in System Block Diagram

Once the data has been acquired through the USB Interface and it has been processed by the Main Core, the user has the option to save the data. Figure 10.1 shows that the data storage block is connected to the Main Core; once the user input is processed the Main Core can store the data in either a numeric format or a graphical format.

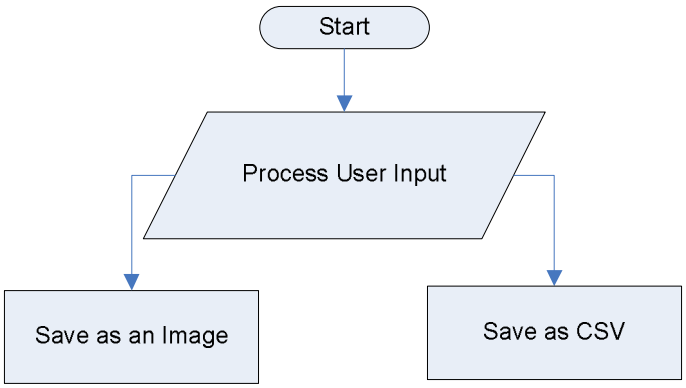


Figure 10.2 Data Storage flow chart

Figure 10.2 shows that the user can choose to save the data as a Comma Separated Value [CSV] file, which is a numeric format, or as an image. If the user chooses to save the current display as an image then only the segment of data that is currently zoomed in on is saved as one of four image formats based on user's need. The user is provided with four image formats to suit

what type of application the image will be used in. The user can choose to save the image as BMP [Bitmap], GIF [Graphic Interchange Format], PNG [Portable Network Graphics], and JPEG [Joint Photographic Experts Group] format. When the user saves these images, the custom names that the user has given to the channels automatically gets embedded to the image, so as to avoid confusion later.

The user can also choose to save the data in the numeric format as CSV file. This format saves all the data for all the lines that are currently being displayed, regardless of the zoom level. This feature is useful because the user can save the data that (s)he acquired as a CSV and then at a later time load the data back into the GUI and manipulate it and use all the features of the software as if the data was live data. This allows for time shifting of data acquisition; data can be acquired and saved and analyzed at a later time.

IV. e. Graphical User Interface

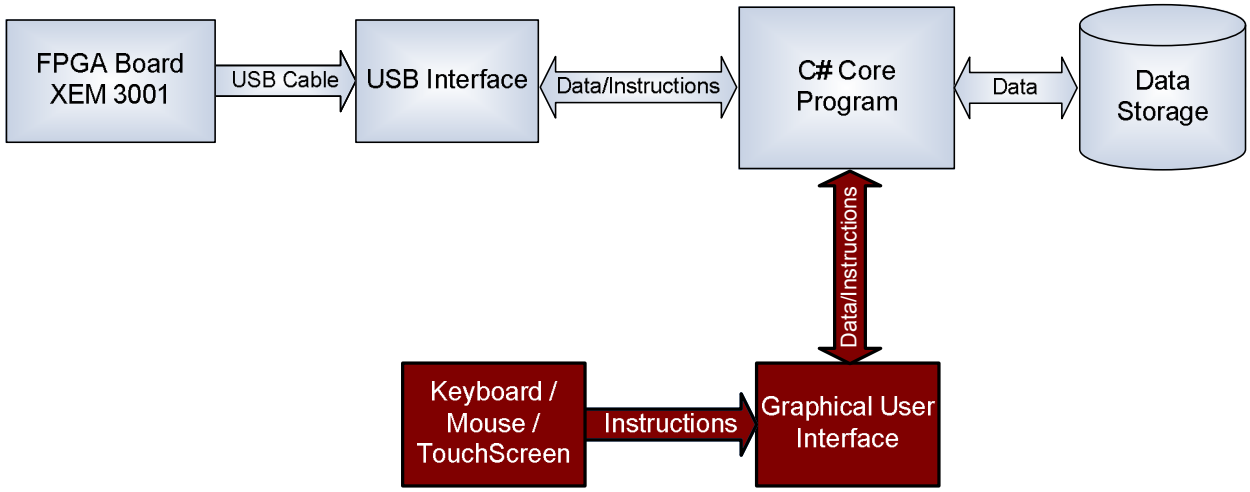


Figure 11.1 GUI in System Block Diagram

The Graphical User Interface is the front end to the whole application. All the application features are accessed through the GUI and all instructions that are sent to the different blocks of the system originate from the GUI. Figure 11.1 shows that the GUI is what

ties the system to the user, the user interacts with the GUI using the mouse, keyboard and optionally through a touch screen monitor. Figure 12.1 shows the functioning of the GUI simplistically, it does not discuss each of the features are implemented.

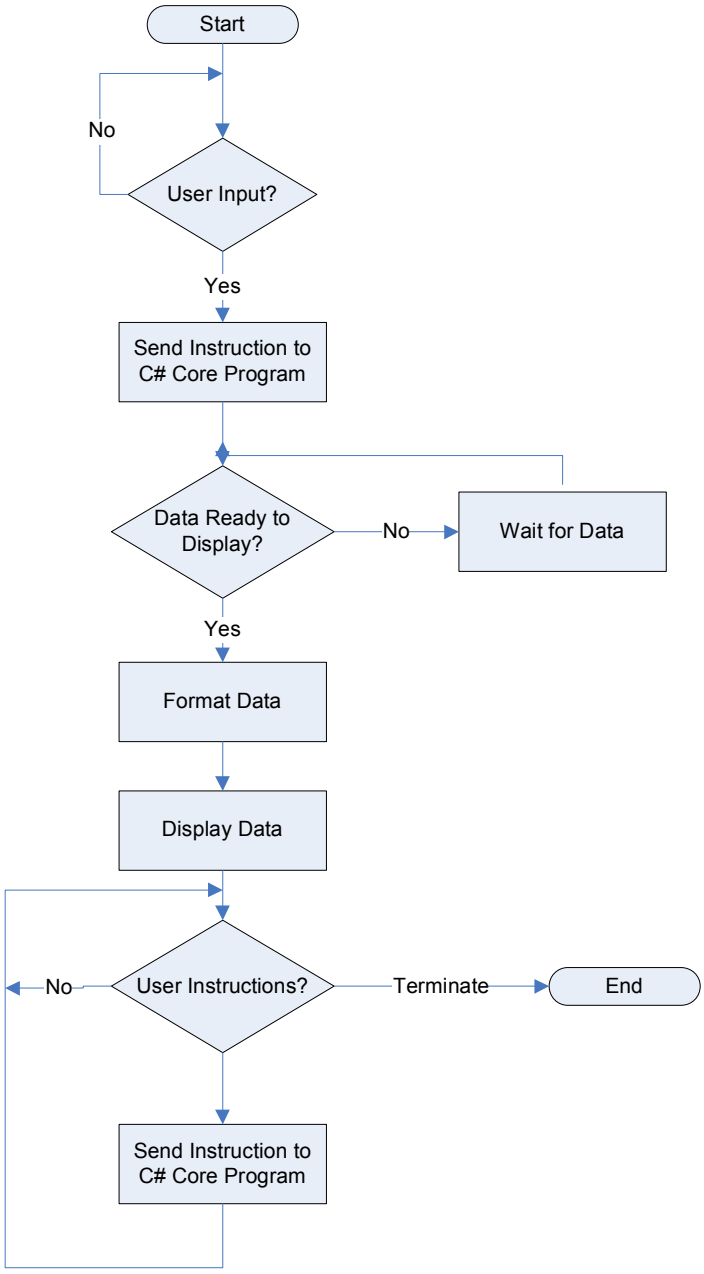


Figure 12.1 GUI Flowchart

The GUI allows the user to access several features. The main window as seen in Figure 13.1 displays the channels that have been selected by the user.

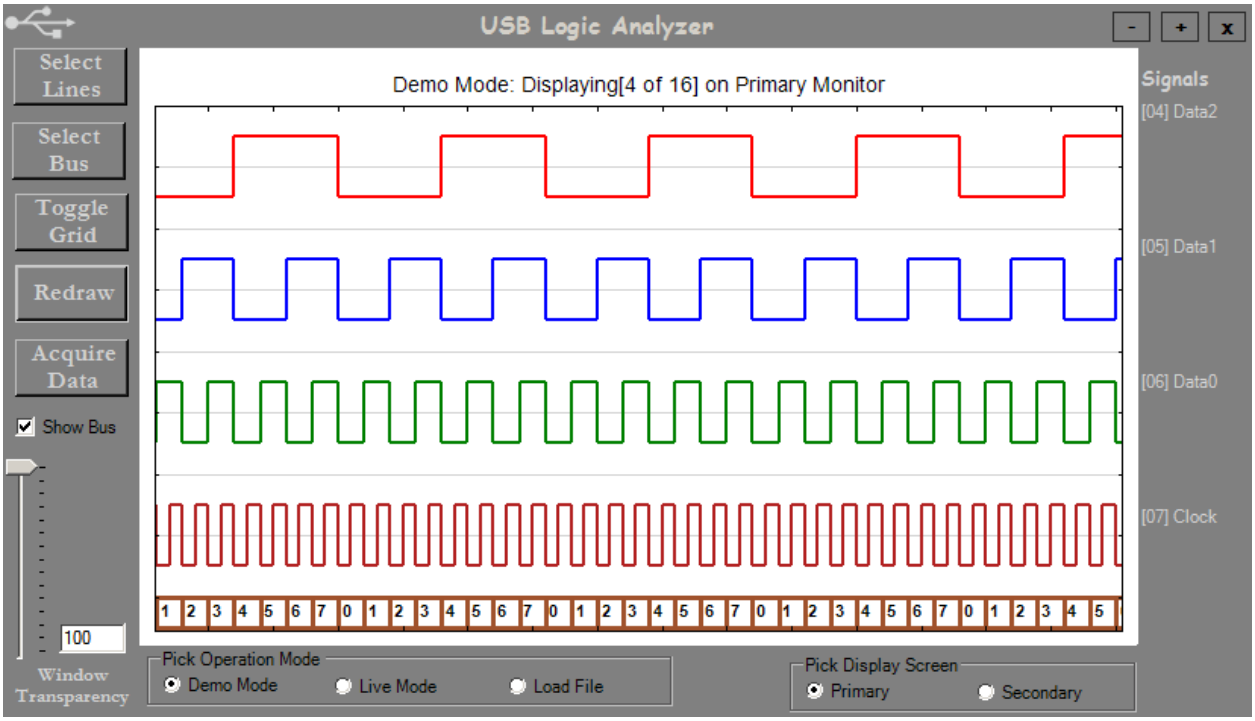


Figure 13.1 GUI Main Window

The main window shows the plotting area (white space) that is showing four channels of digital logic. Near the bottom of the plotting area the data bus is displayed. The data bus shows the hexadecimal value of all the channels that have been added to the bus. In this case channels Data0, Data1, and Data2 have been added to the data bus, where Data0 (the lowest channel part of the bus) is the least significant bit, and where Data2 (the highest channel of the bus) is the most significant bit.

The area to the right of the plotting area shows the names of the channels, these are custom names that the user has chosen to give to the signals so that they make more intuitive sense. The area at the bottom of the plotting area has several radio buttons denoting different options. The set on the left let the user pick between three operational modes: Demo, Live, and Load mode. In the Demo mode (the default startup mode) the application displays data from a text file that has been preloaded. In the Live mode the application queries the FPGA and

acquires data from the FPGA via the USB Interface. In the Load mode, the user can load previously saved data from a *.csv file and then use the application to analyze the data as if it was live data.

The set on the right lets the user move the application windows from displaying on the primary monitor to the secondary monitor and vice-versa. This is helpful if the application is being used on a touch screen monitor, the application starts up on the primary monitor by default, but moving it to the secondary touch screen monitor gives it more of an instrument feel.

The left panel provides more options to the user. It lets the user access two other application windows Select Lines and Select Bus, to be discussed later. It also allows the user to impose a grid onto the plotting area; the vertical grid aligns itself with the rising edge of the fastest changing signal, presumably the clock. The two other buttons, let the user reset the viewing area and update any changes that were made to the options and acquire data directly from the FPGA at that given instance of time.

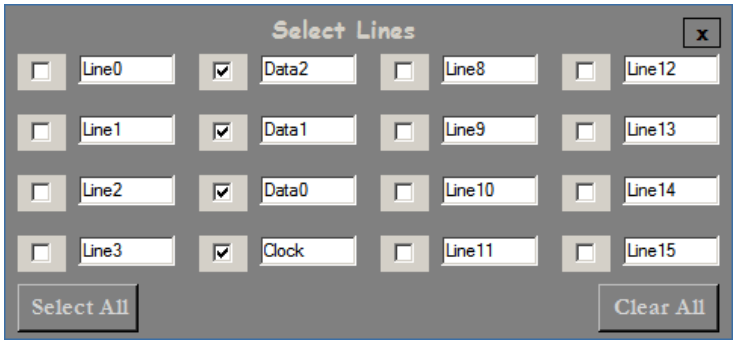


Figure 14.1 Select Lines Window

Figure 14.1 shows the Select Lines window, this window allows the user to do two things. First, it lets the user to select which lines will be displayed in the plotting area. Second, it lets the user to give custom names to the channels, instead of the default names. Once the user

assigns custom names and clicks the Redraw button in the main window the name changes propagate throughout the application.

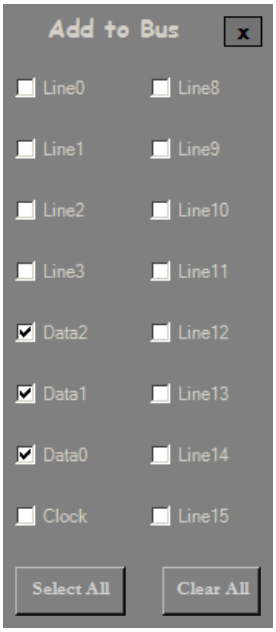


Figure 15.1 Add to Bus Window

Figure 15.1 shows the Add to Bus window, this lets the user select which channels will be part of the data bus. In this case Data0, Data1, and Data2 have been selected so they make up the data bus. The Clock channel has not been selected despite the fact that it is currently being displayed in the plotting area. This allows the user the flexibility of choosing which lines are displayed and which are part of the data bus, so that instruction lines and data lines can be treated separately. A line can be selected for display, but it does not need to be a part of the data bus and vice-versa.

All the windows have their custom window moving handles. The default Microsoft Windows constructs were not used to provide better handling of graphical elements and in handling multiple windows. The main window can be maximized and minimized just like any other MS Windows application; the tertiary windows can only be opened or closed. More information and details can be found in Appendix I and VII.

V. Results

Looking back at the original goals of the project all the primary goals were accomplished:

- ✓ Display using NPlot graphing library
- ✓ Plot acquired data as discreet logic signals
- ✓ Combine data lines into data bus
- ✓ Efficient data processing
- ✓ Print/Save data capture

In addition to these goals, several other goals were accomplished that have been discussed previously, but can be more concisely represented as:

- Hardware
  - ✓ Interface a Spartan3 FPGA board [XEM 3001] over USB
  - ✓ Place data in buffer to transmit over USB
- Software
  - ✓ Acquire instantaneous data on user demand
  - ✓ Parse acquired data into discreet logic signals
  - ✓ Load previously saved data acquisition session

All the features of the software have already been discussed; this section is going to discuss how the application performed when it was acquiring data live from the FPGA board. To reiterate, the board was run at 1 MHz for testing purposes. The FPGA ran an 8 bit binary counter. The value of the count was put on the FIFO buffer which was read by the application and then parsed into discreet logic channels.

The binary count was saved to the FIFO buffer and all the bits were assigned to I/O pins on the FPGA so that they could be observed using an oscilloscope. The least significant bit and

the bit immediately more significant bit were monitored using an oscilloscope. Figure 17.1 shows that these channels are operating at a frequency of 500 KHz and 250 KHz. This proves that the FPGA is operating at the correct frequency of 1 MHz and that the binary counter is functioning correctly. Figure 17.2 shows the same effect with four bits – two bits had to be saved to reference channels on the scope – again proving the advantages of a logic analyzer.

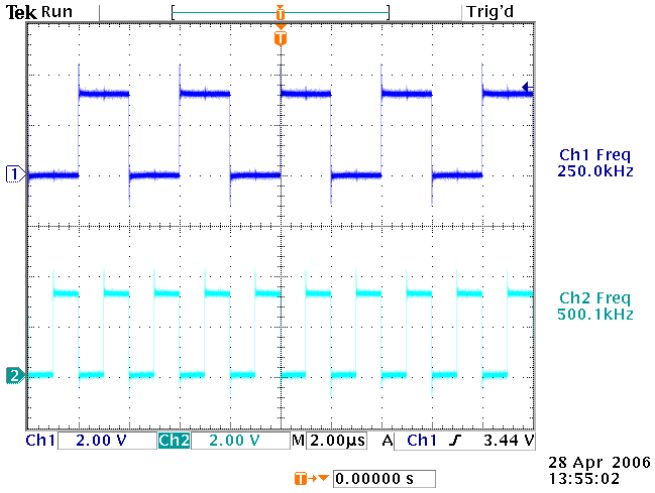


Figure 17.1 Lowest two bits of a binary counter running at 1 MHz

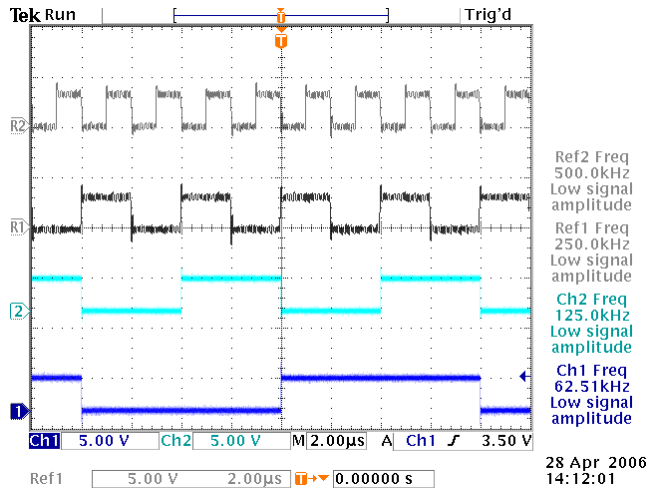


Figure 17.2 Lowest four bits (using reference channels) of a binary counter running at 1 MHz

To show that the data is being correctly transmitted from the FPGA to the C# application the counting was paused when the lowest four bits were displaying 0101_2 or 5_{16}. The scope

capture can be seen in Figure 18.1, and the same data was acquired using the application and the data was saved as an image and can be seen in Figure 18.2

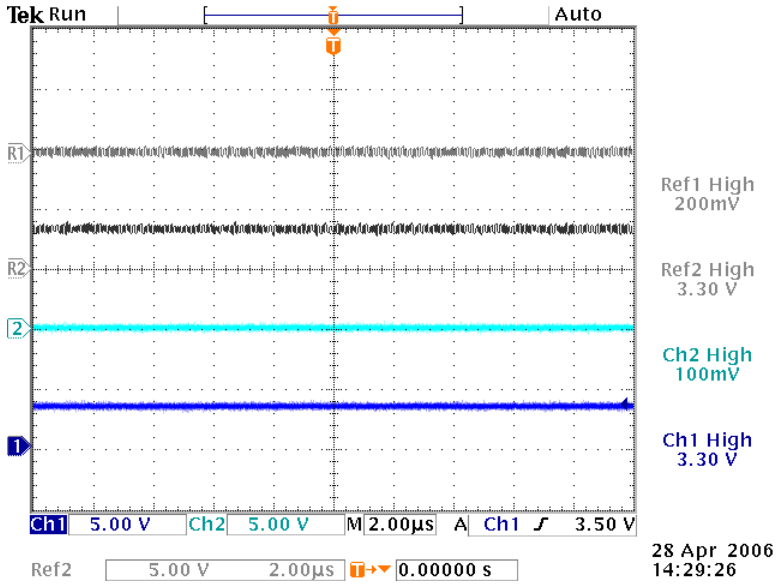


Figure 18.1 Scope capture of lowest four bits representing 0101_2

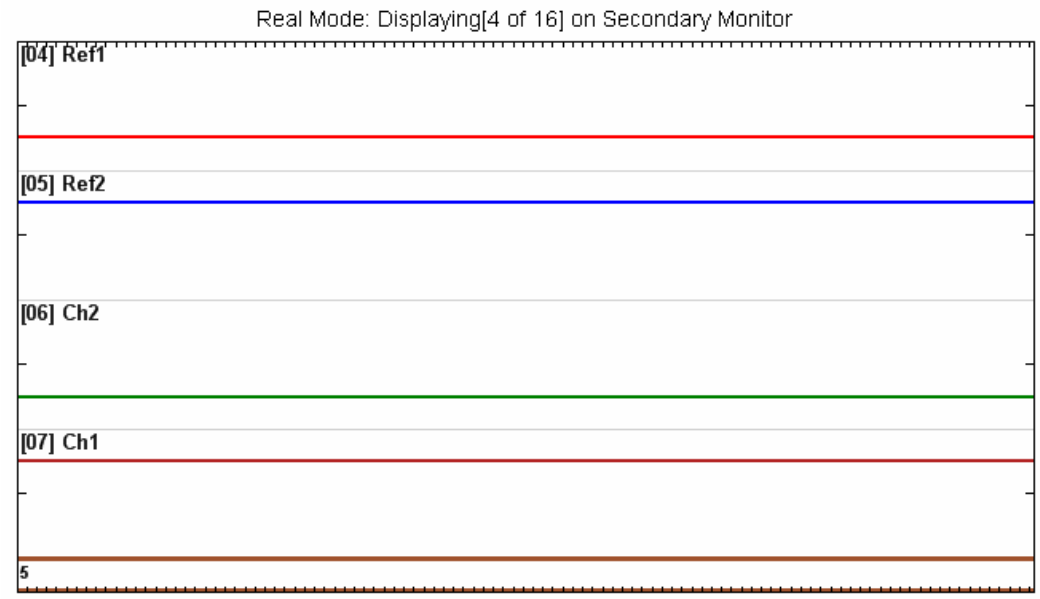


Figure 18.2 Image saved by the application with custom names to match scope shows 0101_2 and 5_{16}

Finally the board was run at 1 MHz and then at 0.5 MHz and data was captured and saved as images. Figure 19.1 shows the data capture at 1 MHz and Figure 19.2 shows the data

capture at 0.5MHz. Visually it is easy to see that the signals are changing twice as fast in Figure 19.1 than they are in Figure 19.2. This also demonstrates the capability of the logic analyzer to acquire data correctly despite the operating frequency.

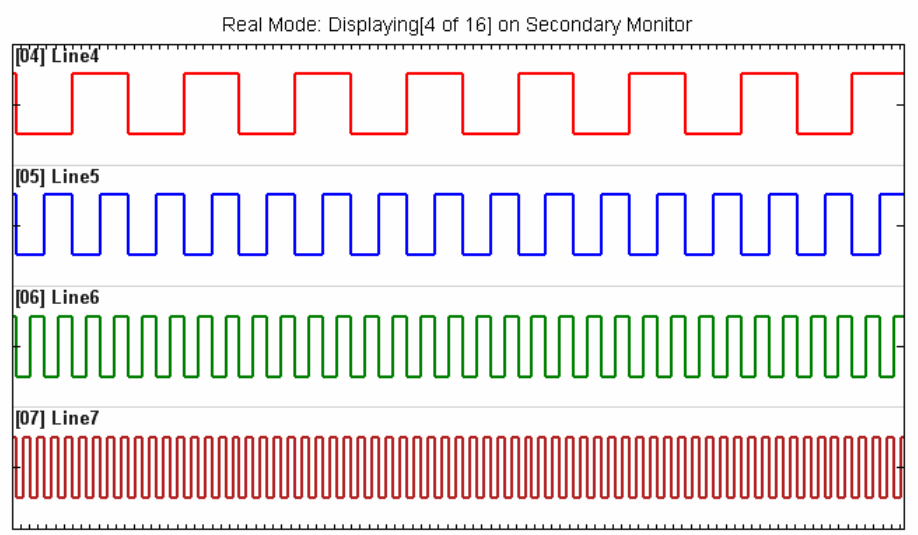


Figure 19.1 Data acquired at 1 MHz

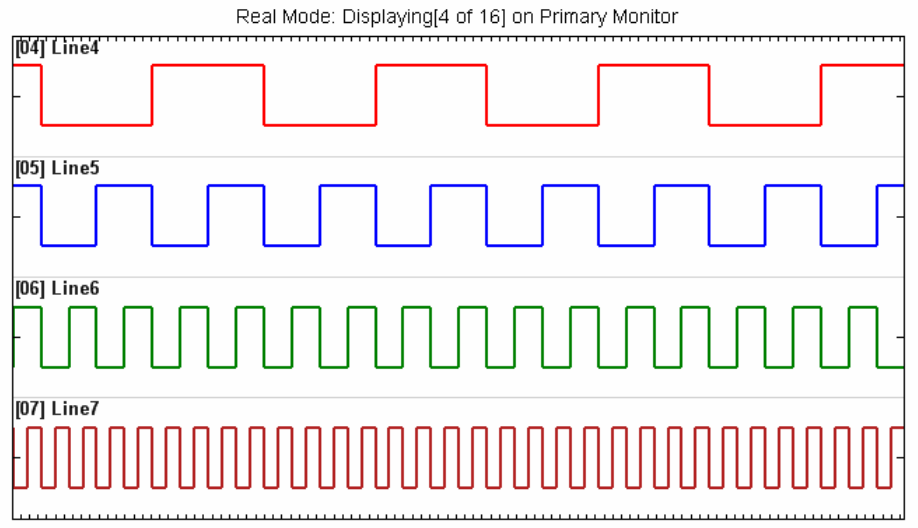


Figure 19.2 Data acquired at 0.5 MHz

To prove the point further and to demonstrate another feature of the application, the data captured at 1 MHz was saved as an image and then the board was run at 0.5 MHz and then the application window was layered on top of the image captured before. The application window's

opacity can be changed to make it translucent. This makes it very easy to compare currently acquired data with previously acquired data, as demonstrated in Figure 20.1. Figure 20.1 also reaffirms the fact that the signals in the background window (board running at 1 MHz) is indeed changing twice as fast as the data that has currently been acquired (board running at 0.5 MHz).

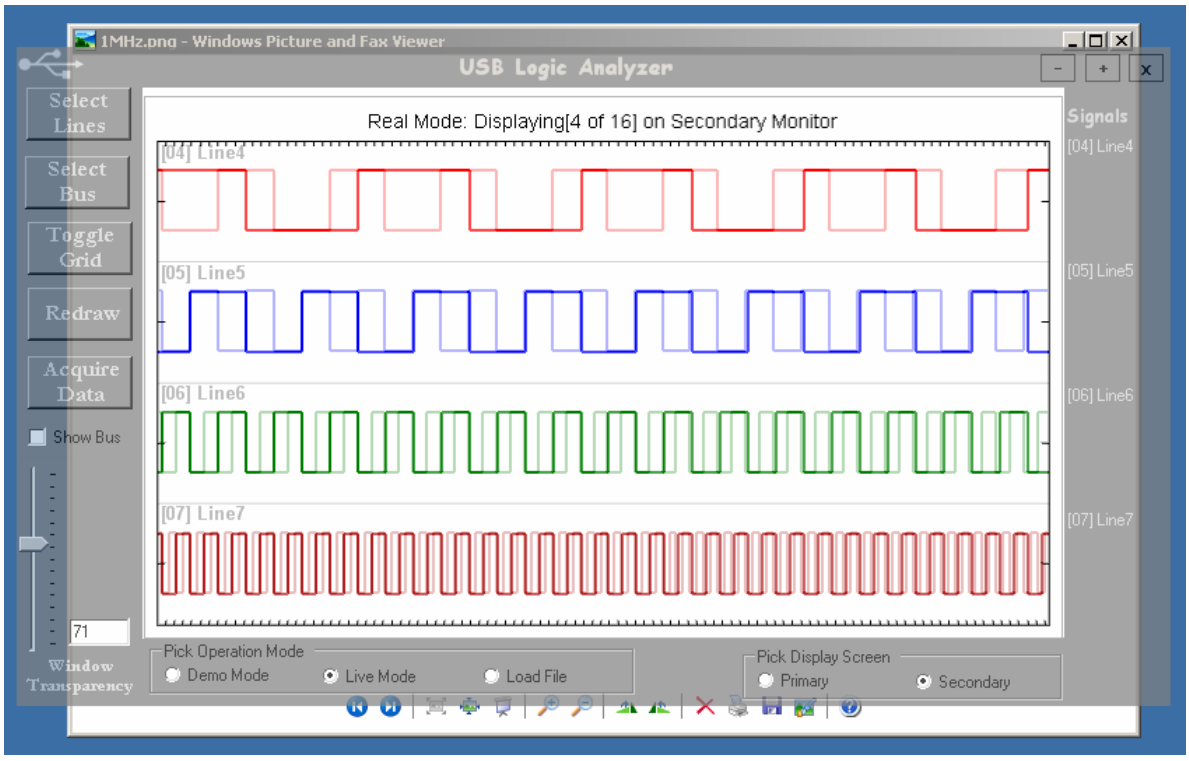


Figure 20.1 Application made translucent and layered on previously capture data

VI. Conclusion

The idea behind the project is to provide a cost effective logic analyzer for use in the Junior Laboratory. The junior lab work stations are equipped with a personal computer and a two channel Tektronix oscilloscope. The USB Logic Analyzer application runs on the personal computer eliminating the need for extra equipment, and it provides sixteen digital channels, bettering the current measurement capability. The juniors are working with EMAC 8-bit µcontroller Boards (running at 12MHz), CPLDs, GALs, Altera FPGA Boards (running at 25 MHz), which makes the desired sampling rate 50 MHz, and the desired data width 8 bits. The

USB Logic Analyzer samples over 100 MHz (pending POD construction), and provides a data width of 16 bits, hence exceeding the requirements.

The application is stable for all end users and the FPGA board is flexible enough to be used for other applications as well. The USB Logic Analyzer has met and exceeded all its goals; the project was an enormous success.

USB Logic Analyzer

Appendix I:

MainCore Code [C#]

```
/// \mainpage USB Logic Analyzer - Abstract
/// Documentation front page info follows.
///
/// A logic analyzer displays logic level of digital signals. A logic analyzer differs from an oscilloscope
/// which is a more powerful instrument but typically only has a few channels. This project seeks to continue
/// work on a digital logic analyzer which has sixteen channels. The logic analyzer will display three logic levels:
/// low, high, and indeterminate. The sixteen channels are sampled by an external conditioning hardware called a
POD.
/// The POD interfaces to the computer using Universal Serial Bus [USB] protocol. The graphical user interface [GUI]
/// on the computer will display the possible three states of the sixteen lines, and allow the user to interact with
/// and interpret the data.  The user is able to select what lines are displayed, choose lines to combine into a
data bus,
/// zoom in on a particular segment of the data, and save the data in numeric or image format.
///
/// Shom Bandopadhaya \n
/// Advisor: Dr. James Irwin, Jr \n
/// Bradley University \n
/// Department of Electrical and Computer Engineering

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Threading;
using System.Diagnostics;
using System.Runtime.InteropServices;//for moving forms

using NPlot;

//! USB Logic Analyzer (usbla) namespace contains main core which is the main class
//! that provides all the functions and objects on the main form.
namespace usbla
{
  //! Multistep will plot multiple step plots on one surface
  public class mainCore : System.Windows.Forms.Form
  {
    #region publicVariables
    //! [User set] number of data lines to be plotted
    public const int numlines = 16;
    //! [User set] number of points per data line
    public const int points = 512;
    public int [,] linedata = new int [numlines,points];
    #endregion

    #region privateVariables
    private System.ComponentModel.Container components = null;
    private NPlot.Windows.PlotSurface2D plot;
    private StepPlot[] lineplots = new StepPlot [numlines];
    private System.Windows.Forms.TrackBar Opacity;
    private System.Windows.Forms.Label OpacityLabel;
    private System.Windows.Forms.TextBox CurrentOpacity;
    private System.Windows.Forms.Button Redraw;
    private System.Windows.Forms.Button SelectLines;
    private Color[] linecolor = new Color[16]
{Color.Purple,Color.Crimson,Color.DarkBlue,Color.DarkMagenta,Color.Red,Color.Blue,Color.Green,Color.Firebrick,Color.
DarkBlue,Color.Red,Color.Blue,Color.Green,Color.Purple,Color.DarkGreen,Color.DarkMagenta,Color.MidnightBlue};
    public System.Windows.Forms.Label []LineLabels = new System.Windows.Forms.Label[16];
    private lineSelect SelectForm = new lineSelect();
    private About AboutForm = new About();
    private busSelect busSelectForm = new busSelect();
    private System.Windows.Forms.Form displayForm_ = null;
    private System.Windows.Forms.Button quit;
    private System.Windows.Forms.Button maximize;
    private int ScaleFactor = 0;
    private int defaultwidth, defaultheight, defaultx, defaulty;
    private System.Windows.Forms.Label usblatitle;
    private System.Windows.Forms.Button ToggleGrid;
    private System.Windows.Forms.Label LineNameLabel;
    private System.Windows.Forms.PictureBox AboutIcon;
    private System.Windows.Forms.Button minimize;
    private System.Windows.Forms.Label label0;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Label label5;
    private System.Windows.Forms.Label label6;
    private System.Windows.Forms.Label label7;
```

```
    private System.Windows.Forms.Label label8;
    private System.Windows.Forms.Label label9;
    private System.Windows.Forms.Label label10;
    private System.Windows.Forms.Label label11;
    private System.Windows.Forms.Label label12;
    private System.Windows.Forms.Label label13;
    private System.Windows.Forms.Label label14;
    private System.Windows.Forms.Label label15;
    private System.Windows.Forms.Label ErrorBase;
    private int [,] selectdata = new int[numlines,points];
    private System.Windows.Forms.Button acquiredata;
    private int counter = new int();
    private int[] busdata = new int[points];
    private System.Windows.Forms.CheckBox showBusButton;
    private TextItem [] busdatatext = new TextItem[points];
    private System.Windows.Forms.RadioButton demoradio;
    private System.Windows.Forms.GroupBox mode;
    private System.Windows.Forms.GroupBox screen;
    private System.Windows.Forms.RadioButton liveradio;
    private SaveFileDialog save = new SaveFileDialog();
    private System.Windows.Forms.RadioButton primaryradio;
    private System.Windows.Forms.RadioButton secondaryradio;
    private System.Windows.Forms.Button addbusbutton;
    private System.Windows.Forms.RadioButton loadradio;
    private int secondaryScreen = new int();
    private OpenFileDialog open = new OpenFileDialog();
    private string curdir = Environment.CurrentDirectory.ToString();
    #endregion


    //! Constructor intializes components and calls the generatePlot() function on this instance of the class
    //! for the number of lines defined by the varibale numlines.
    public mainCore()
    {
      InitializeComponent();
      this.getData();
      this.initializeLabels();
      this.generatePlot();
      if (primaryradio.Checked == true)
                this.primaryDisplay();
      else
        this.secondaryDisplay();
    }

    //!Reads data depending on operation mode to get data for 16 data lines and stores them into the linedata array
    private void getData()
    {
      StreamReader s = new StreamReader(curdir+"./datademo.txt");
      string myline;
      string[] mysplitline;
      int myint;
      int mynumlines = numlines;

      // For cases when switching from user loaded file mode to any other mode
      for (int i=0; i < numlines; i++)
      {
        SelectForm.Lines[i].Visible = true;
        SelectForm.LineNames[i].Visible = true;
        busSelectForm.LineArray[i].Visible = true;
      }

      if (liveradio.Checked == true)
        s = File.OpenText(curdir+"./datareal.txt");
      else if (loadradio.Checked == true)
      {
        open.Filter = "CSV Files (*.csv)|*.csv|" + "All Files|";
        open.Title = "Load LA data from";
        open.AddExtension= true;
        open.InitialDirectory = Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop);
        if (open.ShowDialog() == DialogResult.OK)
          s = File.OpenText(open.FileName.ToString());
        else
        {
          MessageBox.Show("File open failed, reverting to demo data!");
          demoradio.Checked = true;
          return;
        }

        //Read first line with line names and determine number of lines that were saved
        myline = s.ReadLine();
        mynumlines = myline.Split(',').Length-1;
```

```
      //! If the user is loading previously acquired data then all lines that data was not saved for
      //! need to be disabled, which is done by making the checkboxes and nameboxes invisible.
      for (int i=0; i < numlines; i++)
      {
        SelectForm.Lines[i].Checked = false;
        SelectForm.Lines[i].Visible = false;
        SelectForm.LineNames[i].Visible = false;
        busSelectForm.LineArray[i].Checked = false;
        busSelectForm.LineArray[i].Visible = false;
      }
      for (int i=0; i < mynumlines; i++)
      {
        SelectForm.Lines[i].Checked = true;
        SelectForm.Lines[i].Visible = true;
        SelectForm.LineNames[i].Visible = true;
        busSelectForm.LineArray[i].Checked = true;
        busSelectForm.LineArray[i].Visible = true;
      }
      Environment.CurrentDirectory = curdir.ToString();
    }

    try //So that it doesn't crash if user select wrong file format
    {
      for (int i=0; i<points; i++)
      {
        myline = s.ReadLine();
        if (myline != null)
        {
          mysplitline = myline.Split(',');
          for (int j=0; j<mynumlines; j++)
          {
            myint = Convert.ToInt16(mysplitline[j]);
            linedata[j,i] = myint;
          }
        }
      }
    }
    catch{}
      s.Close();

    return;
  }


  //! Generates Step Plot based on user data specified by numlines, points, and linedata
  private void generatePlot()
  {
    // --- Plotting ---
    plot.Clear();

    // --- Grid Code ---
    Grid mygrid = new Grid();
    mygrid.HorizontalGridType = Grid.GridType.None;
    mygrid.VerticalGridType = Grid.GridType.Coarse;
    plot.Add(mygrid);

    //If no lines are selected don't plot
    if (checkNullError())
      return;


    //for (int i=0; i<numlines; i++) //plot low to bottom
    for (int i=numlines-1; i>=0; i--) //plot high to bottom
    {
      if (SelectForm.Lines[i].Checked == true)
      {
        plotLine(i);
        ScaleFactor = ScaleFactor + 2;
      }
    }
    configurePlotSurface();
    addLineLabels();
    mouseInteraction();

    if (showBusButton.Checked==true)
      addBus();

    plot.Refresh();

    ScaleFactor = 0; //Reset number selected to 0 for subsequent redraws
```

```
      counter = 0;
      return;
    }

    //! Adds a data bus consisting of selected signals onto the bottom of the plot.
    //! The line at the bottom of the screen is the most significant bit [MSB] and the one
    //! at the top is the least significant bit [LSB].
    private void addBus()
    {

      generateBusData();
      showBusStructure();

      TextItem [] bustext = new TextItem[points];
      double y = new double();
      y = plot.Height - 20.00;
      string [] temps = busDataTextArray();

      for (int i=0; i<points; i++)
      {
        bustext[i] = new TextItem(new PointD(i+.1,0-.1),temps[i]);
        bustext[i].TextFont = new System.Drawing.Font("Arial",8,System.Drawing.FontStyle.Bold);
        plot.Add(bustext[i]);
      }
      return;
    }

    //! Generates the combined value of selected lines as busdata
    private void generateBusData()
    {
      int temp, total = new int();
      temp = 0;
      total = 0;

      for (int i=0; i<numlines; i++)
      {
        if (busSelectForm.LineArray[i].Checked==true)
        {
          for (int j=0; j<points; j++)
          {
            //Adding a bus code
            if (linedata[i,j] == -1)
              selectdata[total,j]=0;
            else if (linedata[i,j] == 1)
              selectdata[total,j]=1;
            else if (linedata[i,j] == 0)
              selectdata[total,j]=-5;   //indeterminate case
          }
          total++;
        }
      }

      for (int i=0; i<points; i++)
      {
        for (int j=0; j<total; j++)
        {
          if(selectdata[j,i] == 1)
            temp = temp + Convert.ToInt32(Math.Pow(2,total-j-1));
          else if (selectdata[j,i] == -5) //indeterminate case
            temp = -5;
        }
        busdata[i] = temp;
        temp = 0;
      }
      total = 0;
      return;
    }

    //! Returns all the bus values converted to HEX as one continuous string
    private string busDataText()
    {

      string temps = new string(' ',1);
      for(int i=0; i<points; i++)
      {
        if (busdata[i] != -5)
          temps = temps + '<' + Convert.ToString(busdata[i],16) + "> ";
        else
          temps = temps + "<X> ";
      }
      temps = temps.ToUpper();
```

```
      return temps;
    }

    //! Returns all the bus values converted to HEX as a string array
    //! Each element contains the combined HEX value of all the visible lines at that point
    private string[] busDataTextArray()
    {
      string [] temps = new string[points];
      int old = new int();
      for(int i=0; i<points; i++)
      {
        if (busdata[i] == -5)
          temps[i] = "X";
        else if (busdata[i] == old) //Displays blank if the bus value hasn't changed
          temps[i] = " ";
        else
          temps[i] = Convert.ToString(busdata[i],16).ToUpper();//.PadLeft((int)numSelected()/3,'0');
        old = busdata[i];
      }
      return temps;
    }

    //! Shows a busstructure at the bottom of the plot
    private void showBusStructure()
    {
      /*//Histogram to overlay bus value
      HistogramPlot x = new HistogramPlot();
      x.DataSource = busdata;
      x.Filled= true;
      x.RectangleBrush = new RectangleBrushes.VerticalCenterFade( Color.Red, Color.Blue );
      plot.Add(x);*/


      //Bus looking overlap stepplot structure
      StepPlot bustop = new StepPlot();
      StepPlot busbottom = new StepPlot();
      bustop.Pen = new Pen(Color.Sienna,3);
      busbottom.Pen = new Pen(Color.Sienna,3);
      double [] array1 = new double[points];
      double [] array2 = new double[points];
      bool toggle = new bool();
      toggle = true;
      int tempcur = new int();
      int tempold = new int();
      tempold = -5;

      for (int i=0; i<points; i++)
      {
        tempcur = busdata[i];
        if (tempcur == tempold)
        {
          array1[i] = array1[i-1];
          array2[i] = array2[i-1];
        }
        else
        {
          if(tempcur == -5)
          {
            array1[i] = -0.5;
            array2[i] = -0.5;
          }
          else
          {
            array1[i] = Convert.ToInt16(toggle)-1;
            array2[i] = Convert.ToInt16(!toggle)-1;
            toggle = !toggle;
          }
        }
        tempold = tempcur;
      }

      bustop.OrdinateData = array1;
      busbottom.OrdinateData = array2;
      LinearAxis ly1 = (LinearAxis)plot.YAxis1;
      ly1.WorldMin = -1;
      plot.Add(bustop);
      plot.Add(busbottom);
      return;
    }
```

```
    //! Plots line number specified by the integer argument i.
    private void plotLine(int i)
    {
      lineplots[i] = new StepPlot();
      lineplots[i].Pen = new Pen (linecolor[i], 2);
      int [] array = new int[points];
      for(int j=0; j<points; j++)
      {
        array[j]=linedata[i,j]+(2*ScaleFactor+2); //assign value and scale based on line number
      }
      lineplots[i].OrdinateData = array;
      lineplots[i].HideVerticalSegments = false;
      plot.Add(lineplots[i]);
      counter++;
      return;
    }


    //! Configures the x and y axes on the plot, adds legend.
    private void configurePlotSurface()
    {
      //! --- Axis Configure ---
      LinearAxis ly1 = (LinearAxis)plot.YAxis1;
      ly1.LargeTickStep = 4;
      ly1.LargeTickSize = 0;
      ly1.SmallTickSize = 5;
      ly1.NumberOfSmallTicks = 1;
      ly1.HideTickText = true;
      ly1.WorldMax = ScaleFactor*2;
      ly1.WorldMin = 0;

      LinearAxis lx1 = (LinearAxis)plot.XAxis1;
      lx1.HideTickText = true;
      lx1.NumberOfSmallTicks = 0;
      lx1.LargeTickStep = 5;
      lx1.LargeTickSize = plot.Height/100;

      plot.XAxis1.Label="Time";
      plot.YAxis1.Label="Data Lines";
      // --- end Axis Configure ---

      // --- Title Configure ---
      string title = new string(' ',1);
      if (demoradio.Checked == true)
        title = "Demo Mode: ";
      else if (liveradio.Checked == true)
        title = "Real Mode: ";
      else
        title = "User Loaded File: ";
      title = title + "Displaying[" + numSelected().ToString() + " of " + numlines.ToString() + "]";
      if (primaryradio.Checked == true)
        title = title + " on Primary Monitor";
      else
        title = title + " on Secondary Monitor";
      plot.Title = title;

      //! --- ContextMenu Configure ---
      //! Default ContextMenu contains "Show World Coordinate" which messes up the plot surface with
      //! ghost lines left over from dragging, it also contains "Copy Data To Clipboard" which copies
      //! NPlot coordinate information to the clipboard which is not useful.  So the default menu is first
      //! acquired and then stripped of those two functions.

      //plot.RightMenu = NPlot.Windows.PlotSurface2D.DefaultContextMenu;
      NPlot.Windows.PlotSurface2D.PlotContextMenu menu = new NPlot.Windows.PlotSurface2D.PlotContextMenu();
      ArrayList a = menu.MenuItems;
      a.RemoveAt(1);
      a.RemoveAt(5);
      a.Add( new NPlot.Windows.PlotSurface2D.PlotContextMenu.PlotMenuItem( "Save as Image", 4, new
EventHandler(saveAsImage)) );
      a.Add( new NPlot.Windows.PlotSurface2D.PlotContextMenu.PlotMenuItem( "Save as CSV", 5, new
EventHandler(saveAsCSV)) );
      menu.SetMenuItems(a);

      plot.RightMenu = menu;

      //! --- end ContextMenu Configure ---
    }

    //! Lets the user save the plot areas as an image (PNG, JPEG, BMP, or GIF).
    //! It embeds the customs names of the logic lines onto the image for ease of reference later.
    private void saveAsImage(object sender, System.EventArgs e)
```

```
    {
      addLabelsToImage();
      //plot.Refresh();

      // Code copied from NPlot PlotSurface2D class to copy plotsurface image to clipboard
      System.Drawing.Bitmap b = new System.Drawing.Bitmap( plot.Width, plot.Height );
      System.Drawing.Graphics g = Graphics.FromImage( b );
      g.Clear(Color.White);
      this.Draw( g, new Rectangle( 0, 0, b.Width-1, b.Height-1 ) );
      Clipboard.SetDataObject( b, true );

      // Code image as bitmap to the clipboard and prepare to save based on user input
      IDataObject data = Clipboard.GetDataObject();
      Image image = (Image)data.GetData(DataFormats.Bitmap,true);


      save.Filter = "PNG Files (*.png)|*.png|" + "BMP Files (*.bmp)|*.bmp|" +
                "JPG Files (*.jpg,*.jpeg)|*.jpg,*.jpeg|" + "GIF Files (*.gif)|*.gif|" + "All Files|";
      save.Title = "Save Image to";
      save.AddExtension = true;
      save.InitialDirectory = Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop);
      if( save.ShowDialog() == DialogResult.OK )
      {
        switch(save.FilterIndex)
        {
          case 1: image.Save(save.FileName.ToString(),System.Drawing.Imaging.ImageFormat.Png);
            break;
          case 2: image.Save(save.FileName.ToString(),System.Drawing.Imaging.ImageFormat.Bmp);
            break;
          case 3: image.Save(save.FileName.ToString(),System.Drawing.Imaging.ImageFormat.Jpeg);
            break;
          case 4: image.Save(save.FileName.ToString(),System.Drawing.Imaging.ImageFormat.Gif);
            break;
          default:
            image.Save(save.FileName.ToString(),System.Drawing.Imaging.ImageFormat.Png);
            break;
        }
      }

      return;
    }

    //! Saves the numeric data that is currently being plotted in the csv (Comma Separated Value) format.
    private void saveAsCSV(object sender, System.EventArgs e)
    {
      save.Filter = "CSV Files (*.csv)|*.csv|" + "All Files|";
      save.Title = "Save Data as CSV to";
      save.AddExtension = true;
      save.InitialDirectory = Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop);
      int flag = new int();
      flag = 0;

      if( save.ShowDialog() == DialogResult.OK )
      {
        StreamWriter sw = new StreamWriter(save.FileName.ToString(), false);
        for (int i=0; i<points; i++)
        {
          for (int j=0; j<numlines; j++)
          {
            if (SelectForm.Lines[j].Checked==true)
            {
              if (flag == 0 && i==0)
              {
                sw.Write(SelectForm.LineNames[j].Text.ToString());
                sw.Write(",");
              }
              else
              {
                sw.Write(linedata[j,i].ToString());
                if (i!=points)
                  sw.Write(",");
              }
            }
          }
          flag = 1;
          sw.Write(sw.NewLine);
        }
        sw.Close();
      }
      return;
    }
```

```
    //! Draw function is part of NPlot PlotSurface2D class, it had to extracted as code
    //! so that the functionality could be use to copy image to clipboard and then save it
    private void Draw( Graphics g, Rectangle bounds )
    {
      // If we are not in design mode then draw as normal.
      if (LicenseManager.UsageMode == LicenseUsageMode.Designtime)

      {
        g.DrawRectangle( new Pen(Color.Black), bounds.X + 2, bounds.Y + 2, bounds.Width-4, bounds.Height-4 );
      }

      plot.Draw( g, bounds );

      return;
    }

    //! Adds text labels to the plot denoting the custom name and index number of the lines
    //! being displayed when an image is being saved. Labels stays as long as plot is not
    //! redrawn, but it can only be seen if plot is refreshed (by zooming).
    private void addLabelsToImage()
    {
      int total = new int();
      total = numSelected();
      TextItem [] imagelabel = new TextItem[total];
      string [] selectindeces = new string[numlines];
      int curindex = new int();
      curindex = 0;
      for (int i=0; i<numlines; i++)
      {
        if (SelectForm.Lines[i].Checked == true)
        {
          selectindeces[curindex] = LineLabels[i].Text.ToString();
          curindex++;
        }
      }
      for(int i=0; i<total; i++)
      {
        imagelabel[i] = new TextItem(new PointD(0, i*4+4),selectindeces[total-i-1]);
        plot.Add(imagelabel[i]);
        imagelabel[i].TextFont = new System.Drawing.Font("Arial",10,System.Drawing.FontStyle.Bold);
      }
      return;
    }

    //! Adds mouse interaction to the plot, the user can click and drag along x-axis to zoom in.
    private void mouseInteraction()
    {
      plot.AddInteraction(new NPlot.Windows.PlotSurface2D.Interactions.VerticalGuideline());
      plot.AddInteraction(new NPlot.Windows.PlotSurface2D.Interactions.HorizontalRangeSelection());
      plot.AddInteraction(new NPlot.Windows.PlotSurface2D.Interactions.AxisDrag(true));
      //plot.AddInteraction(new NPlot.Windows.PlotSurface2D.Interactions.RubberBandSelection());

      //! TODO: figure out how to zoom in or out based on mouse direction, seems like code incomplete in NPlot
library
      plot.AddInteraction(new NPlot.Windows.PlotSurface2D.Interactions.MouseWheelZoom());

      return;
    }

    //!void initializeLabels() assigns the line labels to an array to help in itteration
    private void initializeLabels()
    {
      LineLabels[0] =  label0;
      LineLabels[1] =  label1;
      LineLabels[2] =  label2;
      LineLabels[3] =  label3;
      LineLabels[4] =  label4;
      LineLabels[5] =  label5;
      LineLabels[6] =  label6;
      LineLabels[7] =  label7;
      LineLabels[8] =  label8;
      LineLabels[9] =  label9;
      LineLabels[10] = label10;
      LineLabels[11] = label11;
      LineLabels[12] = label12;
      LineLabels[13] = label13;
      LineLabels[14] = label14;
      LineLabels[15] = label15;
    }
```

```
    //!void addLineLabels() reads the user defined line names from the SelectForm form
    //!and places them as lables on the main form.  The labels are shown/hidden based
    //!on what lines are currently selected and their position is manipulated to match
    //!the position of the line that is plotted.
    private void addLineLabels()
    {
      int delta = (plot.Height-LineNameLabel.Height)/numSelected();
      int x = LineNameLabel.Location.X;
      int y = LineNameLabel.Location.Y+LineNameLabel.Height;
      int offset = 0;
      for (int i=0; i<numlines; i++)
      {
        LineLabels[i].Size = new System.Drawing.Size(75, 30);
        LineLabels[i].Text = "["+i.ToString().PadLeft(2,'0')+"] " + SelectForm.LineNames[i].Text;
        LineLabels[i].Location = new System.Drawing.Point(x,y+offset*delta);
        if (SelectForm.Lines[i].Checked != true)
          LineLabels[i].Visible = false;
        else
        {
          LineLabels[i].Visible = true;
          LineLabels[i].BringToFront();
          offset++;
        }
      }
      offset = 0;
      return;
    }

    //!int numSelected() function counts the number of lines that are selected
    //in the SelectForm form and returns the count as an integer.
    private int numSelected()
    {
      int numlinesSelected = new int();
      for (int i=0; i<numlines; i++)
        if (SelectForm.Lines[i].Checked == true)
          numlinesSelected++;
      return numlinesSelected;
    }

    //!bool checkNullError() function checks if no lines are selected.
    //!If no lines are selected then it hides user interaction features which would
    //!cause the application to crash.
    private bool checkNullError()
    {
      //!Error Handling required if no lines are selected
      if (numSelected() == 0)
      {
        SelectForm.ErrorLabel.Visible = true;
        SelectForm.BringToFront();
        //Hiding these elements will prevent user interaction
        //that can lead to the application crashing
        this.plot.Hide();
        this.maximize.Hide();
        this.minimize.Hide();
        return true;
      }
      else
      {
        SelectForm.ErrorLabel.Visible = false;
        this.plot.Show();
        this.maximize.Show();
        this.minimize.Show();
        return false;
      }
    }

    //! Configures all display windows to display and resize correctly on primary display monitor
    private void primaryDisplay()
    {
      secondaryScreen = 0;
      this.Left = Screen.PrimaryScreen.WorkingArea.X + 15;
      this.Top = 40;
      SelectForm.Left = Screen.PrimaryScreen.WorkingArea.X + 15;
      SelectForm.Top = this.Height + 45;
      busSelectForm.Left = this.Left + this.Width + 5;
      busSelectForm.Top = this.Top;
      AboutForm.Left = this.Left + this.Width/4;
      AboutForm.Top = this.Top + this.Height/4;
      return;
    }
```

```
    //! Configures all display windows to display and resize correctly on secondary display monitor
    private void secondaryDisplay()
    {
      secondaryScreen = Screen.PrimaryScreen.WorkingArea.Width;
      this.Left = secondaryScreen + 20;
      this.Top = 20;
      SelectForm.Left = Screen.PrimaryScreen.WorkingArea.X + secondaryScreen + 20;
      SelectForm.Top = this.Height + 25;
      busSelectForm.Left = this.Left + this.Width + 5;
      busSelectForm.Top = this.Top;
      AboutForm.Left = this.Left + this.Width/4;
      AboutForm.Top = this.Top + this.Height/4;
      return;
    }

  #region Generated Code
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if (components != null)
        {
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(mainCore));
      this.plot = new NPlot.Windows.PlotSurface2D();
      this.Opacity = new System.Windows.Forms.TrackBar();
      this.OpacityLabel = new System.Windows.Forms.Label();
      this.CurrentOpacity = new System.Windows.Forms.TextBox();
      this.Redraw = new System.Windows.Forms.Button();
      this.SelectLines = new System.Windows.Forms.Button();
      this.quit = new System.Windows.Forms.Button();
      this.maximize = new System.Windows.Forms.Button();
      this.minimize = new System.Windows.Forms.Button();
      this.usblatitle = new System.Windows.Forms.Label();
      this.ToggleGrid = new System.Windows.Forms.Button();
      this.AboutIcon = new System.Windows.Forms.PictureBox();
      this.LineNameLabel = new System.Windows.Forms.Label();
      this.label0 = new System.Windows.Forms.Label();
      this.label1 = new System.Windows.Forms.Label();
      this.label2 = new System.Windows.Forms.Label();
      this.label3 = new System.Windows.Forms.Label();
      this.label4 = new System.Windows.Forms.Label();
      this.label5 = new System.Windows.Forms.Label();
      this.label6 = new System.Windows.Forms.Label();
      this.label7 = new System.Windows.Forms.Label();
      this.label8 = new System.Windows.Forms.Label();
      this.label9 = new System.Windows.Forms.Label();
      this.label10 = new System.Windows.Forms.Label();
      this.label11 = new System.Windows.Forms.Label();
      this.label12 = new System.Windows.Forms.Label();
      this.label13 = new System.Windows.Forms.Label();
      this.label14 = new System.Windows.Forms.Label();
      this.label15 = new System.Windows.Forms.Label();
      this.ErrorBase = new System.Windows.Forms.Label();
      this.acquiredata = new System.Windows.Forms.Button();
      this.showBusButton = new System.Windows.Forms.CheckBox();
      this.mode = new System.Windows.Forms.GroupBox();
      this.demoradio = new System.Windows.Forms.RadioButton();
      this.liveradio = new System.Windows.Forms.RadioButton();
      this.loadradio = new System.Windows.Forms.RadioButton();
      this.screen = new System.Windows.Forms.GroupBox();
      this.primaryradio = new System.Windows.Forms.RadioButton();
      this.secondaryradio = new System.Windows.Forms.RadioButton();
      this.addbusbutton = new System.Windows.Forms.Button();
      ((System.ComponentModel.ISupportInitialize)(this.Opacity)).BeginInit();
      this.mode.SuspendLayout();
```

```
      this.screen.SuspendLayout();
      this.SuspendLayout();
      //
      // plot
      //
      this.plot.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
        | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
      this.plot.AutoScaleAutoGeneratedAxes = false;
      this.plot.AutoScaleTitle = false;
      this.plot.BackColor = System.Drawing.SystemColors.ControlLightLight;
      this.plot.Cursor = System.Windows.Forms.Cursors.Cross;
      this.plot.DateTimeToolTip = false;
      this.plot.Legend = null;
      this.plot.LegendZOrder = -1;
      this.plot.Location = new System.Drawing.Point(90, 35);
      this.plot.Name = "plot";
      this.plot.Padding = 10;
      this.plot.RightMenu = null;
      this.plot.ShowCoordinates = false;
      this.plot.Size = new System.Drawing.Size(658, 388);
      this.plot.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.None;
      this.plot.TabIndex = 0;
      this.plot.Text = "plot";
      this.plot.Title = "USB Logic Analyzer - Shom Bandopadhaya 2006";
      this.plot.TitleFont = new System.Drawing.Font("Arial", 14F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Pixel);
      this.plot.XAxis1 = null;
      this.plot.XAxis2 = null;
      this.plot.YAxis1 = null;
      this.plot.YAxis2 = null;
      //
      // Opacity
      //
      this.Opacity.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
      this.Opacity.Cursor = System.Windows.Forms.Cursors.NoMoveVert;
      this.Opacity.LargeChange = 10;
      this.Opacity.Location = new System.Drawing.Point(0, 292);
      this.Opacity.Maximum = 100;
      this.Opacity.Minimum = 30;
      this.Opacity.Name = "Opacity";
      this.Opacity.Orientation = System.Windows.Forms.Orientation.Vertical;
      this.Opacity.Size = new System.Drawing.Size(42, 148);
      this.Opacity.TabIndex = 5;
      this.Opacity.TickFrequency = 5;
      this.Opacity.Value = 100;
      this.Opacity.Scroll += new System.EventHandler(this.Opacity_Scroll);
      //
      // OpacityLabel
      //
      this.OpacityLabel.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
      this.OpacityLabel.Font = new System.Drawing.Font("Garamond", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.OpacityLabel.ForeColor = System.Drawing.Color.LightGray;
      this.OpacityLabel.Location = new System.Drawing.Point(1, 436);
      this.OpacityLabel.Name = "OpacityLabel";
      this.OpacityLabel.Size = new System.Drawing.Size(90, 28);
      this.OpacityLabel.TabIndex = 15;
      this.OpacityLabel.Text = "Window Transparency";
      this.OpacityLabel.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      //
      // CurrentOpacity
      //
      this.CurrentOpacity.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
      this.CurrentOpacity.Location = new System.Drawing.Point(37, 408);
      this.CurrentOpacity.Name = "CurrentOpacity";
      this.CurrentOpacity.Size = new System.Drawing.Size(44, 20);
      this.CurrentOpacity.TabIndex = 6;
      this.CurrentOpacity.Text = "100";
      this.CurrentOpacity.TextChanged += new System.EventHandler(this.CurrentOpacity_TextChanged);
      //
      // Redraw
      //
      this.Redraw.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.Redraw.ForeColor = System.Drawing.Color.LightGray;
      this.Redraw.Location = new System.Drawing.Point(8, 172);
```

```
      this.Redraw.Name = "Redraw";
      this.Redraw.Size = new System.Drawing.Size(75, 38);
      this.Redraw.TabIndex = 18;
      this.Redraw.Text = "Redraw";
      this.Redraw.Click += new System.EventHandler(this.Redraw_Click);
      //
      // SelectLines
      //
      this.SelectLines.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.SelectLines.ForeColor = System.Drawing.Color.LightGray;
      this.SelectLines.Location = new System.Drawing.Point(7, 29);
      this.SelectLines.Name = "SelectLines";
      this.SelectLines.Size = new System.Drawing.Size(75, 38);
      this.SelectLines.TabIndex = 19;
      this.SelectLines.Text = "Select Lines";
      this.SelectLines.Click += new System.EventHandler(this.SelectLines_Click);
      //
      // quit
      //
      this.quit.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.quit.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
      this.quit.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.quit.Location = new System.Drawing.Point(795, 5);
      this.quit.Name = "quit";
      this.quit.Size = new System.Drawing.Size(25, 20);
      this.quit.TabIndex = 20;
      this.quit.Text = "x";
      this.quit.Click += new System.EventHandler(this.quit_Click);
      //
      // maximize
      //
      this.maximize.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.maximize.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
      this.maximize.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.maximize.Location = new System.Drawing.Point(764, 5);
      this.maximize.Name = "maximize";
      this.maximize.Size = new System.Drawing.Size(25, 20);
      this.maximize.TabIndex = 21;
      this.maximize.Text = "+";
      this.maximize.Click += new System.EventHandler(this.maximize_Click);
      //
      // minimize
      //
      this.minimize.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.minimize.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
      this.minimize.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.minimize.Location = new System.Drawing.Point(732, 5);
      this.minimize.Name = "minimize";
      this.minimize.Size = new System.Drawing.Size(25, 20);
      this.minimize.TabIndex = 22;
      this.minimize.Text = "-";
      this.minimize.Click += new System.EventHandler(this.minimize_Click);
      //
      // usblatitle
      //
      this.usblatitle.Anchor = System.Windows.Forms.AnchorStyles.Top;
      this.usblatitle.Font = new System.Drawing.Font("Comic Sans MS", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.usblatitle.ForeColor = System.Drawing.SystemColors.Control;
      this.usblatitle.Location = new System.Drawing.Point(54, 4);
      this.usblatitle.Name = "usblatitle";
      this.usblatitle.Size = new System.Drawing.Size(677, 20);
      this.usblatitle.TabIndex = 23;
      this.usblatitle.Text = "USB Logic Analyzer";
      this.usblatitle.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      this.usblatitle.DoubleClick += new System.EventHandler(this.usblatitle_DoubleClick);
      this.usblatitle.MouseDown += new System.Windows.Forms.MouseEventHandler(this.Title_MouseDown);
      //
      // ToggleGrid
      //
      this.ToggleGrid.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.ToggleGrid.ForeColor = System.Drawing.Color.LightGray;
      this.ToggleGrid.Location = new System.Drawing.Point(8, 125);
```

```
      this.ToggleGrid.Name = "ToggleGrid";
      this.ToggleGrid.Size = new System.Drawing.Size(75, 38);
      this.ToggleGrid.TabIndex = 24;
      this.ToggleGrid.Text = "Toggle Grid";
      this.ToggleGrid.Click += new System.EventHandler(this.ToggleGrid_Click);
      //
      // AboutIcon
      //
      this.AboutIcon.Image = ((System.Drawing.Image)(resources.GetObject("AboutIcon.Image")));
      this.AboutIcon.Location = new System.Drawing.Point(0, 0);
      this.AboutIcon.Name = "AboutIcon";
      this.AboutIcon.Size = new System.Drawing.Size(59, 24);
      this.AboutIcon.TabIndex = 25;
      this.AboutIcon.TabStop = false;
      this.AboutIcon.Click += new System.EventHandler(this.AboutIcon_Click);
      //
      // LineNameLabel
      //
      this.LineNameLabel.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.LineNameLabel.Font = new System.Drawing.Font("Comic Sans MS", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.LineNameLabel.ForeColor = System.Drawing.Color.LightGray;
      this.LineNameLabel.Location = new System.Drawing.Point(749, 34);
      this.LineNameLabel.Name = "LineNameLabel";
      this.LineNameLabel.Size = new System.Drawing.Size(75, 30);
      this.LineNameLabel.TabIndex = 26;
      this.LineNameLabel.Text = "Signals";
      this.LineNameLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
      //
      // label0
      //
      this.label0.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label0.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label0.ForeColor = System.Drawing.Color.LightGray;
      this.label0.Location = new System.Drawing.Point(749, 68);
      this.label0.Name = "label0";
      this.label0.Size = new System.Drawing.Size(75, 10);
      this.label0.TabIndex = 27;
      this.label0.Text = "Lines";
      //
      // label1
      //
      this.label1.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label1.ForeColor = System.Drawing.Color.LightGray;
      this.label1.Location = new System.Drawing.Point(749, 84);
      this.label1.Name = "label1";
      this.label1.Size = new System.Drawing.Size(75, 10);
      this.label1.TabIndex = 28;
      this.label1.Text = "Lines";
      //
      // label2
      //
      this.label2.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label2.ForeColor = System.Drawing.Color.LightGray;
      this.label2.Location = new System.Drawing.Point(749, 101);
      this.label2.Name = "label2";
      this.label2.Size = new System.Drawing.Size(75, 10);
      this.label2.TabIndex = 29;
      this.label2.Text = "Lines";
      //
      // label3
      //
      this.label3.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label3.ForeColor = System.Drawing.Color.LightGray;
      this.label3.Location = new System.Drawing.Point(749, 117);
      this.label3.Name = "label3";
      this.label3.Size = new System.Drawing.Size(75, 10);
      this.label3.TabIndex = 30;
      this.label3.Text = "Lines";
```

```
      //
      // label4
      //
      this.label4.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label4.ForeColor = System.Drawing.Color.LightGray;
      this.label4.Location = new System.Drawing.Point(749, 137);
      this.label4.Name = "label4";
      this.label4.Size = new System.Drawing.Size(75, 10);
      this.label4.TabIndex = 31;
      this.label4.Text = "Lines";
      //
      // label5
      //
      this.label5.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label5.ForeColor = System.Drawing.Color.LightGray;
      this.label5.Location = new System.Drawing.Point(749, 156);
      this.label5.Name = "label5";
      this.label5.Size = new System.Drawing.Size(75, 10);
      this.label5.TabIndex = 32;
      this.label5.Text = "Lines";
      //
      // label6
      //
      this.label6.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label6.ForeColor = System.Drawing.Color.LightGray;
      this.label6.Location = new System.Drawing.Point(749, 176);
      this.label6.Name = "label6";
      this.label6.Size = new System.Drawing.Size(75, 10);
      this.label6.TabIndex = 33;
      this.label6.Text = "Lines";
      //
      // label7
      //
      this.label7.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label7.ForeColor = System.Drawing.Color.LightGray;
      this.label7.Location = new System.Drawing.Point(749, 197);
      this.label7.Name = "label7";
      this.label7.Size = new System.Drawing.Size(75, 10);
      this.label7.TabIndex = 34;
      this.label7.Text = "Lines";
      //
      // label8
      //
      this.label8.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label8.ForeColor = System.Drawing.Color.LightGray;
      this.label8.Location = new System.Drawing.Point(749, 218);
      this.label8.Name = "label8";
      this.label8.Size = new System.Drawing.Size(75, 10);
      this.label8.TabIndex = 35;
      this.label8.Text = "Lines";
      //
      // label9
      //
      this.label9.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label9.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label9.ForeColor = System.Drawing.Color.LightGray;
      this.label9.Location = new System.Drawing.Point(749, 243);
      this.label9.Name = "label9";
      this.label9.Size = new System.Drawing.Size(75, 10);
      this.label9.TabIndex = 36;
      this.label9.Text = "Lines";
      //
      // label10
      //
```

```
      this.label10.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label10.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label10.ForeColor = System.Drawing.Color.LightGray;
      this.label10.Location = new System.Drawing.Point(749, 264);
      this.label10.Name = "label10";
      this.label10.Size = new System.Drawing.Size(75, 10);
      this.label10.TabIndex = 37;
      this.label10.Text = "Lines";
      //
      // label11
      //
      this.label11.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label11.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label11.ForeColor = System.Drawing.Color.LightGray;
      this.label11.Location = new System.Drawing.Point(749, 288);
      this.label11.Name = "label11";
      this.label11.Size = new System.Drawing.Size(75, 10);
      this.label11.TabIndex = 38;
      this.label11.Text = "Lines";
      //
      // label12
      //
      this.label12.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label12.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label12.ForeColor = System.Drawing.Color.LightGray;
      this.label12.Location = new System.Drawing.Point(749, 307);
      this.label12.Name = "label12";
      this.label12.Size = new System.Drawing.Size(75, 10);
      this.label12.TabIndex = 39;
      this.label12.Text = "Lines";
      //
      // label13
      //
      this.label13.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label13.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label13.ForeColor = System.Drawing.Color.LightGray;
      this.label13.Location = new System.Drawing.Point(749, 323);
      this.label13.Name = "label13";
      this.label13.Size = new System.Drawing.Size(75, 10);
      this.label13.TabIndex = 40;
      this.label13.Text = "Lines";
      //
      // label14
      //
      this.label14.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label14.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label14.ForeColor = System.Drawing.Color.LightGray;
      this.label14.Location = new System.Drawing.Point(749, 341);
      this.label14.Name = "label14";
      this.label14.Size = new System.Drawing.Size(75, 10);
      this.label14.TabIndex = 41;
      this.label14.Text = "Lines";
      //
      // label15
      //
      this.label15.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.label15.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.label15.ForeColor = System.Drawing.Color.LightGray;
      this.label15.Location = new System.Drawing.Point(749, 358);
      this.label15.Name = "label15";
      this.label15.Size = new System.Drawing.Size(75, 10);
      this.label15.TabIndex = 42;
      this.label15.Text = "Lines";
      //
      // ErrorBase
      //
      this.ErrorBase.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
        | System.Windows.Forms.AnchorStyles.Left)
```

```
        | System.Windows.Forms.AnchorStyles.Right)));
      this.ErrorBase.BackColor = System.Drawing.SystemColors.ActiveCaptionText;
      this.ErrorBase.Font = new System.Drawing.Font("Microsoft Sans Serif", 15.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.ErrorBase.ForeColor = System.Drawing.Color.Red;
      this.ErrorBase.Location = new System.Drawing.Point(90, 29);
      this.ErrorBase.Name = "ErrorBase";
      this.ErrorBase.Size = new System.Drawing.Size(659, 393);
      this.ErrorBase.TabIndex = 43;
      this.ErrorBase.Text = "Please select at least one line to plot and then click Refresh!";
      this.ErrorBase.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      //
      // acquiredata
      //
      this.acquiredata.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.acquiredata.ForeColor = System.Drawing.Color.LightGray;
      this.acquiredata.Location = new System.Drawing.Point(8, 221);
      this.acquiredata.Name = "acquiredata";
      this.acquiredata.Size = new System.Drawing.Size(75, 38);
      this.acquiredata.TabIndex = 18;
      this.acquiredata.Text = "Acquire Data";
      this.acquiredata.Click += new System.EventHandler(this.acquiredata_Click);
      //
      // showBusButton
      //
      this.showBusButton.Location = new System.Drawing.Point(8, 267);
      this.showBusButton.Name = "showBusButton";
      this.showBusButton.Size = new System.Drawing.Size(75, 24);
      this.showBusButton.TabIndex = 45;
      this.showBusButton.Text = "Show Bus";
      this.showBusButton.CheckedChanged += new System.EventHandler(this.showBusButton_CheckedChanged);
      //
      // mode
      //
      this.mode.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
      this.mode.Controls.Add(this.demoradio);
      this.mode.Controls.Add(this.liveradio);
      this.mode.Controls.Add(this.loadradio);
      this.mode.Location = new System.Drawing.Point(95, 425);
      this.mode.Name = "mode";
      this.mode.Size = new System.Drawing.Size(347, 38);
      this.mode.TabIndex = 46;
      this.mode.TabStop = false;
      this.mode.Text = "Pick Operation Mode";
      //
      // demoradio
      //
      this.demoradio.Checked = true;
      this.demoradio.Location = new System.Drawing.Point(10, 15);
      this.demoradio.Name = "demoradio";
      this.demoradio.Size = new System.Drawing.Size(104, 18);
      this.demoradio.TabIndex = 0;
      this.demoradio.TabStop = true;
      this.demoradio.Text = "Demo Mode";
      this.demoradio.CheckedChanged += new System.EventHandler(this.demoradio_CheckedChanged);
      //
      // liveradio
      //
      this.liveradio.Location = new System.Drawing.Point(123, 16);
      this.liveradio.Name = "liveradio";
      this.liveradio.Size = new System.Drawing.Size(104, 18);
      this.liveradio.TabIndex = 0;
      this.liveradio.Text = "Live Mode";
      this.liveradio.CheckedChanged += new System.EventHandler(this.liveradio_CheckedChanged);
      //
      // loadradio
      //
      this.loadradio.Location = new System.Drawing.Point(238, 16);
      this.loadradio.Name = "loadradio";
      this.loadradio.Size = new System.Drawing.Size(104, 18);
      this.loadradio.TabIndex = 0;
      this.loadradio.Text = "Load File";
      this.loadradio.Click += new System.EventHandler(this.loadradio_Click);
      //
      // screen
      //
      this.screen.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
      this.screen.Controls.Add(this.primaryradio);
```

```
      this.screen.Controls.Add(this.secondaryradio);
      this.screen.Location = new System.Drawing.Point(519, 429);
      this.screen.Name = "screen";
      this.screen.Size = new System.Drawing.Size(229, 38);
      this.screen.TabIndex = 46;
      this.screen.TabStop = false;
      this.screen.Text = "Pick Display Screen";
      //
      // primaryradio
      //
      this.primaryradio.Checked = true;
      this.primaryradio.Location = new System.Drawing.Point(10, 15);
      this.primaryradio.Name = "primaryradio";
      this.primaryradio.Size = new System.Drawing.Size(104, 18);
      this.primaryradio.TabIndex = 0;
      this.primaryradio.TabStop = true;
      this.primaryradio.Text = "Primary";
      this.primaryradio.CheckedChanged += new System.EventHandler(this.primaryradio_CheckedChanged);
      //
      // secondaryradio
      //
      this.secondaryradio.Location = new System.Drawing.Point(123, 16);
      this.secondaryradio.Name = "secondaryradio";
      this.secondaryradio.Size = new System.Drawing.Size(104, 18);
      this.secondaryradio.TabIndex = 0;
      this.secondaryradio.Text = "Secondary";
      this.secondaryradio.CheckedChanged += new System.EventHandler(this.secondaryradio_CheckedChanged);
      //
      // addbusbutton
      //
      this.addbusbutton.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.addbusbutton.ForeColor = System.Drawing.Color.LightGray;
      this.addbusbutton.Location = new System.Drawing.Point(6, 78);
      this.addbusbutton.Name = "addbusbutton";
      this.addbusbutton.Size = new System.Drawing.Size(75, 38);
      this.addbusbutton.TabIndex = 19;
      this.addbusbutton.Text = "Select Bus";
      this.addbusbutton.Click += new System.EventHandler(this.addbusbutton_Click);
      //
      // multistep
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.BackColor = System.Drawing.SystemColors.ControlDark;
      this.ClientSize = new System.Drawing.Size(825, 471);
      this.Controls.Add(this.label15);
      this.Controls.Add(this.label14);
      this.Controls.Add(this.label13);
      this.Controls.Add(this.label12);
      this.Controls.Add(this.label11);
      this.Controls.Add(this.label10);
      this.Controls.Add(this.label9);
      this.Controls.Add(this.label8);
      this.Controls.Add(this.label7);
      this.Controls.Add(this.label6);
      this.Controls.Add(this.label5);
      this.Controls.Add(this.label4);
      this.Controls.Add(this.label3);
      this.Controls.Add(this.label2);
      this.Controls.Add(this.label1);
      this.Controls.Add(this.label0);
      this.Controls.Add(this.mode);
      this.Controls.Add(this.showBusButton);
      this.Controls.Add(this.plot);
      this.Controls.Add(this.OpacityLabel);
      this.Controls.Add(this.CurrentOpacity);
      this.Controls.Add(this.Opacity);
      this.Controls.Add(this.LineNameLabel);
      this.Controls.Add(this.AboutIcon);
      this.Controls.Add(this.ToggleGrid);
      this.Controls.Add(this.usblatitle);
      this.Controls.Add(this.minimize);
      this.Controls.Add(this.maximize);
      this.Controls.Add(this.quit);
      this.Controls.Add(this.SelectLines);
      this.Controls.Add(this.Redraw);
      this.Controls.Add(this.ErrorBase);
      this.Controls.Add(this.acquiredata);
      this.Controls.Add(this.screen);
      this.Controls.Add(this.addbusbutton);
      this.Cursor = System.Windows.Forms.Cursors.Arrow;
```

```
      this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
      this.Location = new System.Drawing.Point(11, 10);
      this.Name = "multistep";
      this.StartPosition = System.Windows.Forms.FormStartPosition.Manual;
      this.Text = "Multiline";
      ((System.ComponentModel.ISupportInitialize)(this.Opacity)).EndInit();
      this.mode.ResumeLayout(false);
      this.screen.ResumeLayout(false);
      this.ResumeLayout(false);

    }
  #endregion

    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.Run(new mainCore());
    }
    #endregion

  #region User Interaction Handling

    //!User Interaction: void Opacity_Scroll changes the opacity of the form based on
    //!the position of the opacity slider
    private void Opacity_Scroll(object sender, System.EventArgs e)
    {
      CurrentOpacity.Text=Opacity.Value.ToString();
      mainCore.ActiveForm.Opacity=(double)(Opacity.Value)/100;
    }

    //!User Interaction: void CurrentOpacity_TextChanged changes the opacity of the form
    //whenever a valid value is entered in the text box
    private void CurrentOpacity_TextChanged(object sender, System.EventArgs e)
    {
      double curop;
      try
      {
        curop = Convert.ToDouble(CurrentOpacity.Text);
        if (curop >= 30)
        {
          mainCore.ActiveForm.Opacity=curop/100;
          Opacity.Value=Convert.ToInt32(CurrentOpacity.Text);
        }
      }
      catch {}
    }

    //!User Interaction: void Redraw_Click calls the generatePlot() function
    private void Redraw_Click(object sender, System.EventArgs e)
    {
      generatePlot();
      updateLineNames();
    }

    //! Threaded process for showing the lineSelect Form
    private void WindowThread()
    {
      displayForm_.ShowDialog();
    }

    //! User Interaction: Displays the lineSelect Form
    private void SelectLines_Click(object sender, System.EventArgs e)
    {
      displayForm_ = SelectForm;
      System.Threading.Thread t = new Thread( new ThreadStart(WindowThread) );
      t.Start();
      if (t.IsAlive == true)
      {
        SelectForm.BringToFront();
      }
    }

    //!User Interaction: void AboutIcon_Click loads the About form as a threaded process
    private void AboutIcon_Click(object sender, System.EventArgs e)
    {
      displayForm_ = AboutForm;
      System.Threading.Thread s = new Thread( new ThreadStart(WindowThread) );
```

```
      s.Start();
      if (s.IsAlive == true)
      {
        AboutForm.BringToFront();
      }
    }

    //! User Interaction: Starts the busSelectForm as a new thread
    private void addbusbutton_Click(object sender, System.EventArgs e)
    {
      displayForm_ = busSelectForm;
      System.Threading.Thread busthread = new Thread( new ThreadStart(WindowThread) );
      busthread.Start();
      if (busthread.IsAlive == true)
      {
        busSelectForm.BringToFront();
      }
      updateLineNames();
      return;
    }

    //! Updates the custom names of logic lines on the busSelectForm
    private void updateLineNames()
    {
      for (int i=0; i<numlines; i++)
        busSelectForm.LineArray[i].Text = SelectForm.LineNames[i].Text;
      return;
    }

    //!User Interaction: void quit_Click exits the application and closes all related forms
    private void quit_Click(object sender, System.EventArgs e)
    {
      Application.Exit();
    }

    //!User Interaction: void maximize_Click maximizes the form to the max screen resolution
    //!after doing some error checking, on second click the original window dimensions and
    //!locations are restored.  Optimized for display on secondary monitor.
    private void maximize_Click(object sender, System.EventArgs e)
    {
      if (checkNullError())
        return;
      //this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
      if (this.Width != Screen.PrimaryScreen.WorkingArea.Width)
      {
        defaultwidth = this.Width;
        defaultheight = this.Height;
        defaultx = this.Location.X;
        defaulty = this.Location.Y;
        this.Width = Screen.PrimaryScreen.Bounds.Width;//WorkingArea.Width;
        this.Height = Screen.PrimaryScreen.WorkingArea.Height;//.Bounds.Height;//
        this.Left = Screen.PrimaryScreen.WorkingArea.X + secondaryScreen;
        this.Top = Screen.PrimaryScreen.WorkingArea.Y;
      }
      else
      {
        this.Width = defaultwidth;
        this.Height = defaultheight;
        this.Left = defaultx;
        this.Top = defaulty;
      }

      //Fix grid if grid is on
      LinearAxis lx1 = (LinearAxis)plot.XAxis1;
      if (lx1.LargeTickStep == 2)
      {
        lx1.LargeTickValue = 1;
        lx1.LargeTickStep = 2;
        lx1.LargeTickSize = plot.Height/2;
      }
      plot.Refresh();
      addLineLabels();
      return;
    }

    //!User Interaction: void minimize_Click uses predefined minimize to taskbar feature to
    //!minimize the form
    private void minimize_Click(object sender, System.EventArgs e)
    {
      this.WindowState = System.Windows.Forms.FormWindowState.Minimized;
    }
```

```
    //!User Interaction: ToggleGrid_Click overlays a vertical grid aligned to the rising edge
    //!of the fastest changing line being plotted (presumably the clock).  This is achieved
    //!by manipulating the tick size of the LinearAxis provided by NPlot against which the data
    //!is being plotted.
    private void ToggleGrid_Click(object sender, System.EventArgs e)
    {
      LinearAxis lx1 = (LinearAxis)plot.XAxis1;
      if (lx1.LargeTickStep != 2)
      {
        lx1.LargeTickValue = 1;
        lx1.LargeTickStep = 2;
        lx1.LargeTickSize = plot.Height/2;
      }
      else
      {
        lx1.LargeTickStep = 4;
        lx1.LargeTickSize = plot.Height/100;
      }
      plot.Refresh();
    }

    //! API functions to move the form (www.c-sharpcorner.com)
    public const int WM_NCLBUTTONDOWN = 0xA1;
    //! API functions to move the form
    public const int HTCAPTION = 0x2;
    [DllImport("user32.dll")]
    //! API functions to move the form
    public static extern bool ReleaseCapture();
    [DllImport("user32.dll")]
    //! API functions to move the form
    public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

    //!User Interaction: void Title_MouseDown lets the user to move the form around when the
    //!title is clicked and dragged
    private void Title_MouseDown(object sender, MouseEventArgs e)
    {
      //If the left mouse is pressed, release form for movement
      if (e.Button == MouseButtons.Left)
      {
        ReleaseCapture();
        SendMessage(Handle, WM_NCLBUTTONDOWN, HTCAPTION, 0);
      }
    }

    //! User Interaction: Reloads data into memory and updates the plot area. If in real mode
    //! then it queries the FPGA for new data
    private void acquiredata_Click(object sender, System.EventArgs e)
    {
      Process proc = null;
      ProcessStartInfo procInfo = new ProcessStartInfo("./OKPipeRead.exe");
      procInfo.UseShellExecute = false;
      procInfo.CreateNoWindow = true;
      proc = Process.Start(procInfo);
      proc.WaitForExit();
      this.getData();
      this.generatePlot();
    }

    //! User Interaction: If the Show Bus checkbox is checked it adds the databus to plot surface
    private void showBusButton_CheckedChanged(object sender, System.EventArgs e)
    {
      generatePlot();
      return;
    }

    //! User Interaction: Configures windows to display on primary monitor
    private void primaryradio_CheckedChanged(object sender, System.EventArgs e)
    {
      primaryDisplay();
      return;
    }

    //! User Interaction: Configures windows to display on secondary monitor
    private void secondaryradio_CheckedChanged(object sender, System.EventArgs e)
    {
      secondaryDisplay();
      return;
    }

    //! User Interaction: Configures application to run in Live mode with the FPGA connected.
```

```
    //! All data acquisition is done from the board directly via the OKPipeRead.exe program written in C++
    private void liveradio_CheckedChanged(object sender, System.EventArgs e)
    {
      if (liveradio.Checked == true)
      {
        Process proc = null;
        ProcessStartInfo procInfo = new ProcessStartInfo("./OKPipeRead.exe");
        procInfo.UseShellExecute = false;
        procInfo.CreateNoWindow = true;
        proc = Process.Start(procInfo);
        proc.WaitForExit();
        if (proc.ExitCode == 4)
        {
          MessageBox.Show("USB Logic Analyzer did not detect XEM3001. Using demo data.");
          demoradio.Checked = true;
        }
        this.getData();
        this.generatePlot();
      }
    }

    //! User Interaction: Configures application to run in demonstration mode which uses fake data
    private void demoradio_CheckedChanged(object sender, System.EventArgs e)
    {
      if (demoradio.Checked == true)
      {
        this.getData();
        this.generatePlot();
      }
    }

    //! User Interaction: Maximizes the main window on double-clicking the title
    //! (for some reason it only works with right double-click, not left double-click)
    private void usblatitle_DoubleClick(object sender, System.EventArgs e)
    {
      maximize_Click(sender, e);
    }


    //! User Interaction: Allows the user to load previously saved numeric data into the application.
    //! Once loaded the user has can interact with data with all available application features.
    private void loadradio_Click(object sender, System.EventArgs e)
    {
      this.getData();
      this.generatePlot();
    }

  #endregion
  }
}
```

USB Logic Analyzer

Appendix II:

LineSelect Code [C#]

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Runtime.InteropServices;//for moving forms

namespace usbla
{
  /// <summary>
  /// Displays a form for the user to select which lines will be plotted on the plot surface.
  /// </summary>
  public class lineSelect : System.Windows.Forms.Form
  {
    public System.Windows.Forms.CheckBox []Lines = new System.Windows.Forms.CheckBox[16];
    public System.Windows.Forms.TextBox []LineNames = new System.Windows.Forms.TextBox[16];
    public System.Windows.Forms.Label ErrorLabel;

    #region Private Variable
    private System.Windows.Forms.CheckBox Line0;
    private System.Windows.Forms.CheckBox Line1;
    private System.Windows.Forms.CheckBox Line2;
    private System.Windows.Forms.CheckBox Line3;
    private System.Windows.Forms.CheckBox Line4;
    private System.Windows.Forms.CheckBox Line5;
    private System.Windows.Forms.CheckBox Line6;
    private System.Windows.Forms.CheckBox Line7;
    private System.Windows.Forms.CheckBox Line8;
    private System.Windows.Forms.CheckBox Line9;
    private System.Windows.Forms.CheckBox Line10;
    private System.Windows.Forms.CheckBox Line11;
    private System.Windows.Forms.CheckBox Line12;
    private System.Windows.Forms.CheckBox Line13;
    private System.Windows.Forms.CheckBox Line14;
    private System.Windows.Forms.CheckBox Line15;
    private System.Windows.Forms.Button quit;
    private System.Windows.Forms.Label SelectLinestitle;
    private System.Windows.Forms.TextBox LineName0;
    private System.Windows.Forms.TextBox LineName1;
    private System.Windows.Forms.TextBox LineName2;
    private System.Windows.Forms.TextBox LineName3;
    private System.Windows.Forms.TextBox LineName4;
    private System.Windows.Forms.TextBox LineName5;
    private System.Windows.Forms.TextBox LineName6;
    private System.Windows.Forms.TextBox LineName7;
    private System.Windows.Forms.TextBox LineName8;
    private System.Windows.Forms.TextBox LineName9;
    private System.Windows.Forms.TextBox LineName10;
    private System.Windows.Forms.TextBox LineName11;
    private System.Windows.Forms.TextBox LineName12;
    private System.Windows.Forms.TextBox LineName13;
    private System.Windows.Forms.TextBox LineName14;
    private System.Windows.Forms.TextBox LineName15;
    private System.Windows.Forms.Button SelectAll;
    private System.Windows.Forms.Button ClearAll;
    #endregion


    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public lineSelect()
    {
      //
      // Required for Windows Form Designer support
      //
      InitializeComponent();
      Lines[0] = Line0;
      Lines[1] = Line1;
      Lines[2] = Line2;
      Lines[3] = Line3;
      Lines[4] = Line4;
      Lines[5] = Line5;
      Lines[6] = Line6;
      Lines[7] = Line7;
      Lines[8] = Line8;
      Lines[9] = Line9;
      Lines[10] = Line10;
      Lines[11] = Line11;
```

```
      Lines[12] = Line12;
      Lines[13] = Line13;
      Lines[14] = Line14;
      Lines[15] = Line15;

      LineNames[0] = LineName0;
      LineNames[1] = LineName1;
      LineNames[2] = LineName2;
      LineNames[3] = LineName3;
      LineNames[4] = LineName4;
      LineNames[5] = LineName5;
      LineNames[6] = LineName6;
      LineNames[7] = LineName7;
      LineNames[8] = LineName8;
      LineNames[9] = LineName9;
      LineNames[10] = LineName10;
      LineNames[11] = LineName11;
      LineNames[12] = LineName12;
      LineNames[13] = LineName13;
      LineNames[14] = LineName14;
      LineNames[15] = LineName15;
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if(components != null)
        {
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(lineSelect));
      this.Line1 = new System.Windows.Forms.CheckBox();
      this.Line2 = new System.Windows.Forms.CheckBox();
      this.Line3 = new System.Windows.Forms.CheckBox();
      this.Line4 = new System.Windows.Forms.CheckBox();
      this.Line5 = new System.Windows.Forms.CheckBox();
      this.Line6 = new System.Windows.Forms.CheckBox();
      this.Line0 = new System.Windows.Forms.CheckBox();
      this.Line7 = new System.Windows.Forms.CheckBox();
      this.Line8 = new System.Windows.Forms.CheckBox();
      this.Line9 = new System.Windows.Forms.CheckBox();
      this.Line15 = new System.Windows.Forms.CheckBox();
      this.Line14 = new System.Windows.Forms.CheckBox();
      this.Line13 = new System.Windows.Forms.CheckBox();
      this.Line12 = new System.Windows.Forms.CheckBox();
      this.Line11 = new System.Windows.Forms.CheckBox();
      this.Line10 = new System.Windows.Forms.CheckBox();
      this.quit = new System.Windows.Forms.Button();
      this.LineName0 = new System.Windows.Forms.TextBox();
      this.LineName1 = new System.Windows.Forms.TextBox();
      this.LineName2 = new System.Windows.Forms.TextBox();
      this.LineName3 = new System.Windows.Forms.TextBox();
      this.LineName4 = new System.Windows.Forms.TextBox();
      this.LineName5 = new System.Windows.Forms.TextBox();
      this.LineName6 = new System.Windows.Forms.TextBox();
      this.LineName7 = new System.Windows.Forms.TextBox();
      this.LineName8 = new System.Windows.Forms.TextBox();
      this.LineName9 = new System.Windows.Forms.TextBox();
      this.LineName10 = new System.Windows.Forms.TextBox();
      this.LineName11 = new System.Windows.Forms.TextBox();
      this.LineName12 = new System.Windows.Forms.TextBox();
      this.LineName13 = new System.Windows.Forms.TextBox();
      this.LineName14 = new System.Windows.Forms.TextBox();
      this.LineName15 = new System.Windows.Forms.TextBox();
      this.SelectLinestitle = new System.Windows.Forms.Label();
      this.SelectAll = new System.Windows.Forms.Button();
```

```
        this.ClearAll = new System.Windows.Forms.Button();
        this.ErrorLabel = new System.Windows.Forms.Label();
        this.SuspendLayout();
        //
        // Line1
        //
        this.Line1.BackColor = System.Drawing.SystemColors.Control;
        this.Line1.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line1.ForeColor = System.Drawing.SystemColors.Control;
        this.Line1.Location = new System.Drawing.Point(8, 72);
        this.Line1.Name = "Line1";
        this.Line1.Size = new System.Drawing.Size(32, 24);
        this.Line1.TabIndex = 1;
        //
        // Line2
        //
        this.Line2.BackColor = System.Drawing.SystemColors.Control;
        this.Line2.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line2.ForeColor = System.Drawing.SystemColors.Control;
        this.Line2.Location = new System.Drawing.Point(8, 112);
        this.Line2.Name = "Line2";
        this.Line2.Size = new System.Drawing.Size(32, 24);
        this.Line2.TabIndex = 2;
        //
        // Line3
        //
        this.Line3.BackColor = System.Drawing.SystemColors.Control;
        this.Line3.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line3.ForeColor = System.Drawing.SystemColors.Control;
        this.Line3.Location = new System.Drawing.Point(8, 152);
        this.Line3.Name = "Line3";
        this.Line3.Size = new System.Drawing.Size(32, 24);
        this.Line3.TabIndex = 3;
        //
        // Line4
        //
        this.Line4.BackColor = System.Drawing.SystemColors.Control;
        this.Line4.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line4.Checked = true;
        this.Line4.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Line4.ForeColor = System.Drawing.SystemColors.Control;
        this.Line4.Location = new System.Drawing.Point(128, 32);
        this.Line4.Name = "Line4";
        this.Line4.Size = new System.Drawing.Size(32, 24);
        this.Line4.TabIndex = 4;
        //
        // Line5
        //
        this.Line5.BackColor = System.Drawing.SystemColors.Control;
        this.Line5.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line5.Checked = true;
        this.Line5.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Line5.ForeColor = System.Drawing.SystemColors.Control;
        this.Line5.Location = new System.Drawing.Point(128, 72);
        this.Line5.Name = "Line5";
        this.Line5.Size = new System.Drawing.Size(32, 24);
        this.Line5.TabIndex = 5;
        //
        // Line6
        //
        this.Line6.BackColor = System.Drawing.SystemColors.Control;
        this.Line6.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line6.Checked = true;
        this.Line6.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Line6.ForeColor = System.Drawing.SystemColors.Control;
        this.Line6.Location = new System.Drawing.Point(128, 112);
        this.Line6.Name = "Line6";
        this.Line6.Size = new System.Drawing.Size(32, 24);
        this.Line6.TabIndex = 6;
        //
        // Line0
        //
        this.Line0.BackColor = System.Drawing.SystemColors.Control;
        this.Line0.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line0.ForeColor = System.Drawing.SystemColors.Control;
        this.Line0.Location = new System.Drawing.Point(8, 32);
        this.Line0.Name = "Line0";
        this.Line0.Size = new System.Drawing.Size(32, 24);
        this.Line0.TabIndex = 0;
        //
        // Line7
```

```
        //
        this.Line7.BackColor = System.Drawing.SystemColors.Control;
        this.Line7.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line7.Checked = true;
        this.Line7.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Line7.ForeColor = System.Drawing.SystemColors.Control;
        this.Line7.Location = new System.Drawing.Point(128, 152);
        this.Line7.Name = "Line7";
        this.Line7.Size = new System.Drawing.Size(32, 24);
        this.Line7.TabIndex = 7;
        //
        // Line8
        //
        this.Line8.BackColor = System.Drawing.SystemColors.Control;
        this.Line8.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line8.ForeColor = System.Drawing.SystemColors.Control;
        this.Line8.Location = new System.Drawing.Point(248, 32);
        this.Line8.Name = "Line8";
        this.Line8.Size = new System.Drawing.Size(32, 24);
        this.Line8.TabIndex = 8;
        //
        // Line9
        //
        this.Line9.BackColor = System.Drawing.SystemColors.Control;
        this.Line9.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line9.ForeColor = System.Drawing.SystemColors.Control;
        this.Line9.Location = new System.Drawing.Point(248, 72);
        this.Line9.Name = "Line9";
        this.Line9.Size = new System.Drawing.Size(32, 24);
        this.Line9.TabIndex = 9;
        //
        // Line15
        //
        this.Line15.BackColor = System.Drawing.SystemColors.Control;
        this.Line15.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line15.ForeColor = System.Drawing.SystemColors.Control;
        this.Line15.Location = new System.Drawing.Point(368, 152);
        this.Line15.Name = "Line15";
        this.Line15.Size = new System.Drawing.Size(32, 24);
        this.Line15.TabIndex = 15;
        //
        // Line14
        //
        this.Line14.BackColor = System.Drawing.SystemColors.Control;
        this.Line14.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line14.ForeColor = System.Drawing.SystemColors.Control;
        this.Line14.Location = new System.Drawing.Point(368, 112);
        this.Line14.Name = "Line14";
        this.Line14.Size = new System.Drawing.Size(32, 24);
        this.Line14.TabIndex = 14;
        //
        // Line13
        //
        this.Line13.BackColor = System.Drawing.SystemColors.Control;
        this.Line13.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line13.ForeColor = System.Drawing.SystemColors.Control;
        this.Line13.Location = new System.Drawing.Point(368, 72);
        this.Line13.Name = "Line13";
        this.Line13.Size = new System.Drawing.Size(32, 24);
        this.Line13.TabIndex = 13;
        //
        // Line12
        //
        this.Line12.BackColor = System.Drawing.SystemColors.Control;
        this.Line12.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line12.ForeColor = System.Drawing.SystemColors.Control;
        this.Line12.Location = new System.Drawing.Point(368, 32);
        this.Line12.Name = "Line12";
        this.Line12.Size = new System.Drawing.Size(32, 24);
        this.Line12.TabIndex = 12;
        //
        // Line11
        //
        this.Line11.BackColor = System.Drawing.SystemColors.Control;
        this.Line11.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.Line11.ForeColor = System.Drawing.SystemColors.Control;
        this.Line11.Location = new System.Drawing.Point(248, 152);
        this.Line11.Name = "Line11";
        this.Line11.Size = new System.Drawing.Size(32, 24);
        this.Line11.TabIndex = 11;
        //
```

```
      // Line10
      //
      this.Line10.BackColor = System.Drawing.SystemColors.Control;
      this.Line10.CheckAlign = System.Drawing.ContentAlignment.MiddleCenter;
      this.Line10.ForeColor = System.Drawing.SystemColors.Control;
      this.Line10.Location = new System.Drawing.Point(248, 112);
      this.Line10.Name = "Line10";
      this.Line10.Size = new System.Drawing.Size(32, 24);
      this.Line10.TabIndex = 10;
      //
      // quit
      //
      this.quit.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.quit.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
      this.quit.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.quit.Location = new System.Drawing.Point(448, 8);
      this.quit.Name = "quit";
      this.quit.Size = new System.Drawing.Size(25, 20);
      this.quit.TabIndex = 34;
      this.quit.Text = "x";
      this.quit.Click += new System.EventHandler(this.quit_Click);
      //
      // LineName0
      //
      this.LineName0.Location = new System.Drawing.Point(48, 32);
      this.LineName0.Name = "LineName0";
      this.LineName0.Size = new System.Drawing.Size(64, 20);
      this.LineName0.TabIndex = 16;
      this.LineName0.Text = "Line0";
      //
      // LineName1
      //
      this.LineName1.Location = new System.Drawing.Point(48, 72);
      this.LineName1.Name = "LineName1";
      this.LineName1.Size = new System.Drawing.Size(64, 20);
      this.LineName1.TabIndex = 17;
      this.LineName1.Text = "Line1";
      //
      // LineName2
      //
      this.LineName2.Location = new System.Drawing.Point(48, 112);
      this.LineName2.Name = "LineName2";
      this.LineName2.Size = new System.Drawing.Size(64, 20);
      this.LineName2.TabIndex = 18;
      this.LineName2.Text = "Line2";
      //
      // LineName3
      //
      this.LineName3.Location = new System.Drawing.Point(48, 152);
      this.LineName3.Name = "LineName3";
      this.LineName3.Size = new System.Drawing.Size(64, 20);
      this.LineName3.TabIndex = 19;
      this.LineName3.Text = "Line3";
      //
      // LineName4
      //
      this.LineName4.Location = new System.Drawing.Point(168, 32);
      this.LineName4.Name = "LineName4";
      this.LineName4.Size = new System.Drawing.Size(64, 20);
      this.LineName4.TabIndex = 20;
      this.LineName4.Text = "Line4";
      //
      // LineName5
      //
      this.LineName5.Location = new System.Drawing.Point(168, 72);
      this.LineName5.Name = "LineName5";
      this.LineName5.Size = new System.Drawing.Size(64, 20);
      this.LineName5.TabIndex = 21;
      this.LineName5.Text = "Line5";
      //
      // LineName6
      //
      this.LineName6.Location = new System.Drawing.Point(168, 112);
      this.LineName6.Name = "LineName6";
      this.LineName6.Size = new System.Drawing.Size(64, 20);
      this.LineName6.TabIndex = 22;
      this.LineName6.Text = "Line6";
      //
      // LineName7
```

```
      //
      this.LineName7.Location = new System.Drawing.Point(168, 152);
      this.LineName7.Name = "LineName7";
      this.LineName7.Size = new System.Drawing.Size(64, 20);
      this.LineName7.TabIndex = 23;
      this.LineName7.Text = "Line7";
      //
      // LineName8
      //
      this.LineName8.Location = new System.Drawing.Point(288, 32);
      this.LineName8.Name = "LineName8";
      this.LineName8.Size = new System.Drawing.Size(64, 20);
      this.LineName8.TabIndex = 24;
      this.LineName8.Text = "Line8";
      //
      // LineName9
      //
      this.LineName9.Location = new System.Drawing.Point(288, 72);
      this.LineName9.Name = "LineName9";
      this.LineName9.Size = new System.Drawing.Size(64, 20);
      this.LineName9.TabIndex = 25;
      this.LineName9.Text = "Line9";
      //
      // LineName10
      //
      this.LineName10.Location = new System.Drawing.Point(288, 112);
      this.LineName10.Name = "LineName10";
      this.LineName10.Size = new System.Drawing.Size(64, 20);
      this.LineName10.TabIndex = 26;
      this.LineName10.Text = "Line10";
      //
      // LineName11
      //
      this.LineName11.Location = new System.Drawing.Point(288, 152);
      this.LineName11.Name = "LineName11";
      this.LineName11.Size = new System.Drawing.Size(64, 20);
      this.LineName11.TabIndex = 27;
      this.LineName11.Text = "Line11";
      //
      // LineName12
      //
      this.LineName12.Location = new System.Drawing.Point(408, 32);
      this.LineName12.Name = "LineName12";
      this.LineName12.Size = new System.Drawing.Size(64, 20);
      this.LineName12.TabIndex = 28;
      this.LineName12.Text = "Line12";
      //
      // LineName13
      //
      this.LineName13.Location = new System.Drawing.Point(408, 72);
      this.LineName13.Name = "LineName13";
      this.LineName13.Size = new System.Drawing.Size(64, 20);
      this.LineName13.TabIndex = 29;
      this.LineName13.Text = "Line13";
      //
      // LineName14
      //
      this.LineName14.Location = new System.Drawing.Point(408, 112);
      this.LineName14.Name = "LineName14";
      this.LineName14.Size = new System.Drawing.Size(64, 20);
      this.LineName14.TabIndex = 30;
      this.LineName14.Text = "Line14";
      //
      // LineName15
      //
      this.LineName15.Location = new System.Drawing.Point(408, 152);
      this.LineName15.Name = "LineName15";
      this.LineName15.Size = new System.Drawing.Size(64, 20);
      this.LineName15.TabIndex = 31;
      this.LineName15.Text = "Line15";
      //
      // SelectLinestitle
      //
      this.SelectLinestitle.Anchor = System.Windows.Forms.AnchorStyles.None;
      this.SelectLinestitle.Font = new System.Drawing.Font("Comic Sans MS", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.SelectLinestitle.ForeColor = System.Drawing.SystemColors.Control;
      this.SelectLinestitle.Location = new System.Drawing.Point(0, 0);
      this.SelectLinestitle.Name = "SelectLinestitle";
      this.SelectLinestitle.Size = new System.Drawing.Size(448, 32);
      this.SelectLinestitle.TabIndex = 38;
```

```
      this.SelectLinestitle.Text = "Select Lines";
      this.SelectLinestitle.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      this.SelectLinestitle.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.SelectLinestitle_MouseDown);
      //
      // SelectAll
      //
      this.SelectAll.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.SelectAll.ForeColor = System.Drawing.Color.LightGray;
      this.SelectAll.Location = new System.Drawing.Point(8, 184);
      this.SelectAll.Name = "SelectAll";
      this.SelectAll.Size = new System.Drawing.Size(80, 32);
      this.SelectAll.TabIndex = 32;
      this.SelectAll.Text = "Select All";
      this.SelectAll.Click += new System.EventHandler(this.SelectAll_Click);
      //
      // ClearAll
      //
      this.ClearAll.Font = new System.Drawing.Font("Garamond", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.ClearAll.ForeColor = System.Drawing.Color.LightGray;
      this.ClearAll.Location = new System.Drawing.Point(392, 184);
      this.ClearAll.Name = "ClearAll";
      this.ClearAll.Size = new System.Drawing.Size(80, 32);
      this.ClearAll.TabIndex = 33;
      this.ClearAll.Text = "Clear All";
      this.ClearAll.Click += new System.EventHandler(this.ClearAll_Click);
      //
      // ErrorLabel
      //
      this.ErrorLabel.Font = new System.Drawing.Font("Garamond", 14.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.ErrorLabel.ForeColor = System.Drawing.Color.Red;
      this.ErrorLabel.Location = new System.Drawing.Point(112, 184);
      this.ErrorLabel.Name = "ErrorLabel";
      this.ErrorLabel.Size = new System.Drawing.Size(256, 32);
      this.ErrorLabel.TabIndex = 39;
      this.ErrorLabel.Text = "Please select at least one line!";
      this.ErrorLabel.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      this.ErrorLabel.Visible = false;
      //
      // LineSelect
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.BackColor = System.Drawing.SystemColors.ControlDark;
      this.ClientSize = new System.Drawing.Size(480, 220);
      this.ControlBox = false;
      this.Controls.Add(this.ErrorLabel);
      this.Controls.Add(this.ClearAll);
      this.Controls.Add(this.SelectAll);
      this.Controls.Add(this.SelectLinestitle);
      this.Controls.Add(this.LineName8);
      this.Controls.Add(this.LineName9);
      this.Controls.Add(this.LineName10);
      this.Controls.Add(this.LineName11);
      this.Controls.Add(this.LineName12);
      this.Controls.Add(this.LineName13);
      this.Controls.Add(this.LineName14);
      this.Controls.Add(this.LineName15);
      this.Controls.Add(this.LineName4);
      this.Controls.Add(this.LineName5);
      this.Controls.Add(this.LineName6);
      this.Controls.Add(this.LineName7);
      this.Controls.Add(this.LineName2);
      this.Controls.Add(this.LineName3);
      this.Controls.Add(this.LineName1);
      this.Controls.Add(this.LineName0);
      this.Controls.Add(this.Line0);
      this.Controls.Add(this.Line3);
      this.Controls.Add(this.Line2);
      this.Controls.Add(this.Line1);
      this.Controls.Add(this.Line7);
      this.Controls.Add(this.Line6);
      this.Controls.Add(this.Line5);
      this.Controls.Add(this.Line4);
      this.Controls.Add(this.quit);
      this.Controls.Add(this.Line9);
      this.Controls.Add(this.Line15);
      this.Controls.Add(this.Line14);
      this.Controls.Add(this.Line13);
```

```
      this.Controls.Add(this.Line12);
      this.Controls.Add(this.Line11);
      this.Controls.Add(this.Line10);
      this.Controls.Add(this.Line8);
      this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
      this.Location = new System.Drawing.Point(12, 460);
      this.Name = "LineSelect";
      this.ShowInTaskbar = false;
      this.SizeGripStyle = System.Windows.Forms.SizeGripStyle.Show;
      this.StartPosition = System.Windows.Forms.FormStartPosition.Manual;
      this.Text = "LineSelect";
      this.ResumeLayout(false);

    }
    #endregion

    private void quit_Click(object sender, System.EventArgs e)
    {
      this.Close();
    }

    //! API functions to move the form (www.c-sharpcorner.com)
    public const int WM_NCLBUTTONDOWN = 0xA1;
    //! API functions to move the form
    public const int HTCAPTION = 0x2;
    [DllImport("user32.dll")]
    //! API functions to move the form
    public static extern bool ReleaseCapture();
    [DllImport("user32.dll")]
    //! API functions to move the form
    public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

    //! User Interaction: Lets the user move the window around
    private void SelectLinestitle_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)
    {
      //If the left mouse is pressed, release form for movement
      if (e.Button == MouseButtons.Left)
      {
        ReleaseCapture();
        SendMessage(Handle, WM_NCLBUTTONDOWN, HTCAPTION, 0);
      }
    }

    //! User Interaction: Checks all check boxes that are visible
    //! (boxes are not visible when lines not available in Load File mode)
    private void SelectAll_Click(object sender, System.EventArgs e)
    {
      for (int i=0; i < 16; i++)
      {
        if (Lines[i].Visible == true)
          Lines[i].Checked = true;
      }
    }

    //! User Interaction: Clears all check boxes that are visible
    //! (boxes are not visible when lines not available in Load File mode)
    private void ClearAll_Click(object sender, System.EventArgs e)
    {
      for (int i=0; i < 16; i++)
      {
        Lines[i].Checked = false;
      }
    }
  }
}
```

USB Logic Analyzer

Appendix III:

BusSelect Code [C#]

busSelect.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Runtime.InteropServices;//for moving forms

namespace usbla
{
  /// <summary>
  /// Summary description for lineSelect.
  /// </summary>
  public class busSelect : System.Windows.Forms.Form
  {
    public System.Windows.Forms.CheckBox []LineArray = new System.Windows.Forms.CheckBox[16];

  #region Private Variable
    private System.Windows.Forms.CheckBox Line0;
    private System.Windows.Forms.CheckBox Line1;
    private System.Windows.Forms.CheckBox Line2;
    private System.Windows.Forms.CheckBox Line3;
    private System.Windows.Forms.CheckBox Line4;
    private System.Windows.Forms.CheckBox Line5;
    private System.Windows.Forms.CheckBox Line6;
    private System.Windows.Forms.CheckBox Line7;
    private System.Windows.Forms.CheckBox Line8;
    private System.Windows.Forms.CheckBox Line9;
    private System.Windows.Forms.CheckBox Line10;
    private System.Windows.Forms.CheckBox Line11;
    private System.Windows.Forms.CheckBox Line12;
    private System.Windows.Forms.CheckBox Line13;
    private System.Windows.Forms.CheckBox Line14;
    private System.Windows.Forms.CheckBox Line15;
    private System.Windows.Forms.Button quit;
    private System.Windows.Forms.Label SelectLinestitle;
    private System.Windows.Forms.Button SelectAll;
    private System.Windows.Forms.Button ClearAll;
    public System.Windows.Forms.Label ErrorLabel;
  #endregion

    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public busSelect()
    {
      //
      // Required for Windows Form Designer support
      //
      InitializeComponent();
      LineArray[0] = Line0;
      LineArray[1] = Line1;
      LineArray[2] = Line2;
      LineArray[3] = Line3;
      LineArray[4] = Line4;
      LineArray[5] = Line5;
      LineArray[6] = Line6;
      LineArray[7] = Line7;
      LineArray[8] = Line8;
      LineArray[9] = Line9;
      LineArray[10] = Line10;
      LineArray[11] = Line11;
      LineArray[12] = Line12;
      LineArray[13] = Line13;
      LineArray[14] = Line14;
      LineArray[15] = Line15;
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if(components != null)
        {
          components.Dispose();
        }
      }
```

```
      base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(busSelect));
      this.Line1 = new System.Windows.Forms.CheckBox();
      this.Line2 = new System.Windows.Forms.CheckBox();
      this.Line3 = new System.Windows.Forms.CheckBox();
      this.Line4 = new System.Windows.Forms.CheckBox();
      this.Line5 = new System.Windows.Forms.CheckBox();
      this.Line6 = new System.Windows.Forms.CheckBox();
      this.Line0 = new System.Windows.Forms.CheckBox();
      this.Line7 = new System.Windows.Forms.CheckBox();
      this.Line8 = new System.Windows.Forms.CheckBox();
      this.Line9 = new System.Windows.Forms.CheckBox();
      this.Line15 = new System.Windows.Forms.CheckBox();
      this.Line14 = new System.Windows.Forms.CheckBox();
      this.Line13 = new System.Windows.Forms.CheckBox();
      this.Line12 = new System.Windows.Forms.CheckBox();
      this.Line11 = new System.Windows.Forms.CheckBox();
      this.Line10 = new System.Windows.Forms.CheckBox();
      this.quit = new System.Windows.Forms.Button();
      this.SelectLinestitle = new System.Windows.Forms.Label();
      this.SelectAll = new System.Windows.Forms.Button();
      this.ClearAll = new System.Windows.Forms.Button();
      this.ErrorLabel = new System.Windows.Forms.Label();
      this.SuspendLayout();
      //
      // Line1
      //
      this.Line1.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line1.ForeColor = System.Drawing.SystemColors.Control;
      this.Line1.Location = new System.Drawing.Point(8, 80);
      this.Line1.Name = "Line1";
      this.Line1.Size = new System.Drawing.Size(75, 30);
      this.Line1.TabIndex = 1;
      this.Line1.Text = "[01]";
      //
      // Line2
      //
      this.Line2.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line2.ForeColor = System.Drawing.SystemColors.Control;
      this.Line2.Location = new System.Drawing.Point(8, 120);
      this.Line2.Name = "Line2";
      this.Line2.Size = new System.Drawing.Size(75, 30);
      this.Line2.TabIndex = 2;
      this.Line2.Text = "[02]";
      //
      // Line3
      //
      this.Line3.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line3.ForeColor = System.Drawing.SystemColors.Control;
      this.Line3.Location = new System.Drawing.Point(8, 160);
      this.Line3.Name = "Line3";
      this.Line3.Size = new System.Drawing.Size(75, 30);
      this.Line3.TabIndex = 3;
      this.Line3.Text = "[03]";
      //
      // Line4
      //
      this.Line4.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line4.Checked = true;
      this.Line4.CheckState = System.Windows.Forms.CheckState.Checked;
      this.Line4.ForeColor = System.Drawing.SystemColors.Control;
      this.Line4.Location = new System.Drawing.Point(8, 200);
      this.Line4.Name = "Line4";
      this.Line4.Size = new System.Drawing.Size(75, 30);
      this.Line4.TabIndex = 4;
      this.Line4.Text = "[04]";
      //
      // Line5
      //
      this.Line5.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line5.Checked = true;
      this.Line5.CheckState = System.Windows.Forms.CheckState.Checked;
```

```
        this.Line5.ForeColor = System.Drawing.SystemColors.Control;
        this.Line5.Location = new System.Drawing.Point(8, 240);
        this.Line5.Name = "Line5";
        this.Line5.Size = new System.Drawing.Size(75, 30);
        this.Line5.TabIndex = 5;
        this.Line5.Text = "[05]";
        //
        // Line6
        //
        this.Line6.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line6.Checked = true;
        this.Line6.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Line6.ForeColor = System.Drawing.SystemColors.Control;
        this.Line6.Location = new System.Drawing.Point(8, 280);
        this.Line6.Name = "Line6";
        this.Line6.Size = new System.Drawing.Size(75, 30);
        this.Line6.TabIndex = 6;
        this.Line6.Text = "[06]";
        //
        // Line0
        //
        this.Line0.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line0.ForeColor = System.Drawing.SystemColors.Control;
        this.Line0.Location = new System.Drawing.Point(8, 40);
        this.Line0.Name = "Line0";
        this.Line0.Size = new System.Drawing.Size(75, 30);
        this.Line0.TabIndex = 0;
        this.Line0.Text = "[00]";
        //
        // Line7
        //
        this.Line7.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line7.Checked = true;
        this.Line7.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Line7.ForeColor = System.Drawing.SystemColors.Control;
        this.Line7.Location = new System.Drawing.Point(8, 320);
        this.Line7.Name = "Line7";
        this.Line7.Size = new System.Drawing.Size(75, 30);
        this.Line7.TabIndex = 7;
        this.Line7.Text = "[07]";
        //
        // Line8
        //
        this.Line8.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line8.ForeColor = System.Drawing.SystemColors.Control;
        this.Line8.Location = new System.Drawing.Point(96, 40);
        this.Line8.Name = "Line8";
        this.Line8.Size = new System.Drawing.Size(75, 30);
        this.Line8.TabIndex = 8;
        this.Line8.Text = "[08]";
        //
        // Line9
        //
        this.Line9.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line9.ForeColor = System.Drawing.SystemColors.Control;
        this.Line9.Location = new System.Drawing.Point(96, 80);
        this.Line9.Name = "Line9";
        this.Line9.Size = new System.Drawing.Size(75, 30);
        this.Line9.TabIndex = 9;
        this.Line9.Text = "[09]";
        //
        // Line15
        //
        this.Line15.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line15.ForeColor = System.Drawing.SystemColors.Control;
        this.Line15.Location = new System.Drawing.Point(96, 320);
        this.Line15.Name = "Line15";
        this.Line15.Size = new System.Drawing.Size(75, 30);
        this.Line15.TabIndex = 15;
        this.Line15.Text = "[15]";
        //
        // Line14
        //
        this.Line14.BackColor = System.Drawing.SystemColors.ControlDark;
        this.Line14.ForeColor = System.Drawing.SystemColors.Control;
        this.Line14.Location = new System.Drawing.Point(96, 280);
        this.Line14.Name = "Line14";
        this.Line14.Size = new System.Drawing.Size(75, 30);
        this.Line14.TabIndex = 14;
        this.Line14.Text = "[14]";
        //
```

```
      // Line13
      //
      this.Line13.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line13.ForeColor = System.Drawing.SystemColors.Control;
      this.Line13.Location = new System.Drawing.Point(96, 240);
      this.Line13.Name = "Line13";
      this.Line13.Size = new System.Drawing.Size(75, 30);
      this.Line13.TabIndex = 13;
      this.Line13.Text = "[13]";
      //
      // Line12
      //
      this.Line12.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line12.ForeColor = System.Drawing.SystemColors.Control;
      this.Line12.Location = new System.Drawing.Point(96, 200);
      this.Line12.Name = "Line12";
      this.Line12.Size = new System.Drawing.Size(75, 30);
      this.Line12.TabIndex = 12;
      this.Line12.Text = "[12]";
      //
      // Line11
      //
      this.Line11.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line11.ForeColor = System.Drawing.SystemColors.Control;
      this.Line11.Location = new System.Drawing.Point(96, 160);
      this.Line11.Name = "Line11";
      this.Line11.Size = new System.Drawing.Size(75, 30);
      this.Line11.TabIndex = 11;
      this.Line11.Text = "[11]";
      //
      // Line10
      //
      this.Line10.BackColor = System.Drawing.SystemColors.ControlDark;
      this.Line10.ForeColor = System.Drawing.SystemColors.Control;
      this.Line10.Location = new System.Drawing.Point(96, 120);
      this.Line10.Name = "Line10";
      this.Line10.Size = new System.Drawing.Size(75, 30);
      this.Line10.TabIndex = 10;
      this.Line10.Text = "[10]";
      //
      // quit
      //
      this.quit.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
      this.quit.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
      this.quit.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.quit.Location = new System.Drawing.Point(144, 8);
      this.quit.Name = "quit";
      this.quit.Size = new System.Drawing.Size(25, 20);
      this.quit.TabIndex = 34;
      this.quit.Text = "x";
      this.quit.Click += new System.EventHandler(this.quit_Click);
      //
      // SelectLinestitle
      //
      this.SelectLinestitle.Anchor = System.Windows.Forms.AnchorStyles.None;
      this.SelectLinestitle.Font = new System.Drawing.Font("Comic Sans MS", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.SelectLinestitle.ForeColor = System.Drawing.SystemColors.Control;
      this.SelectLinestitle.Location = new System.Drawing.Point(0, 0);
      this.SelectLinestitle.Name = "SelectLinestitle";
      this.SelectLinestitle.Size = new System.Drawing.Size(144, 32);
      this.SelectLinestitle.TabIndex = 38;
      this.SelectLinestitle.Text = "Add to Bus";
      this.SelectLinestitle.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      this.SelectLinestitle.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.SelectLinestitle_MouseDown);
      //
      // SelectAll
      //
      this.SelectAll.Font = new System.Drawing.Font("Garamond", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.SelectAll.ForeColor = System.Drawing.Color.LightGray;
      this.SelectAll.Location = new System.Drawing.Point(8, 368);
      this.SelectAll.Name = "SelectAll";
      this.SelectAll.Size = new System.Drawing.Size(72, 32);
      this.SelectAll.TabIndex = 32;
      this.SelectAll.Text = "Select All";
      this.SelectAll.Click += new System.EventHandler(this.SelectAll_Click);
      //
```

```
      // ClearAll
      //
      this.ClearAll.Font = new System.Drawing.Font("Garamond", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.ClearAll.ForeColor = System.Drawing.Color.LightGray;
      this.ClearAll.Location = new System.Drawing.Point(104, 368);
      this.ClearAll.Name = "ClearAll";
      this.ClearAll.Size = new System.Drawing.Size(64, 32);
      this.ClearAll.TabIndex = 33;
      this.ClearAll.Text = "Clear All";
      this.ClearAll.Click += new System.EventHandler(this.ClearAll_Click);
      //
      // ErrorLabel
      //
      this.ErrorLabel.BackColor = System.Drawing.SystemColors.ControlDark;
      this.ErrorLabel.Font = new System.Drawing.Font("Garamond", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.ErrorLabel.ForeColor = System.Drawing.Color.Red;
      this.ErrorLabel.Location = new System.Drawing.Point(0, 352);
      this.ErrorLabel.Name = "ErrorLabel";
      this.ErrorLabel.Size = new System.Drawing.Size(176, 16);
      this.ErrorLabel.TabIndex = 39;
      this.ErrorLabel.Text = "Please select at least one line!";
      this.ErrorLabel.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      this.ErrorLabel.Visible = false;
      //
      // busSelect
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.BackColor = System.Drawing.SystemColors.ControlDark;
      this.ClientSize = new System.Drawing.Size(176, 408);
      this.ControlBox = false;
      this.Controls.Add(this.ClearAll);
      this.Controls.Add(this.SelectAll);
      this.Controls.Add(this.SelectLinestitle);
      this.Controls.Add(this.Line7);
      this.Controls.Add(this.Line0);
      this.Controls.Add(this.Line6);
      this.Controls.Add(this.Line5);
      this.Controls.Add(this.Line4);
      this.Controls.Add(this.Line3);
      this.Controls.Add(this.Line2);
      this.Controls.Add(this.Line1);
      this.Controls.Add(this.quit);
      this.Controls.Add(this.Line9);
      this.Controls.Add(this.Line15);
      this.Controls.Add(this.Line14);
      this.Controls.Add(this.Line13);
      this.Controls.Add(this.Line12);
      this.Controls.Add(this.Line11);
      this.Controls.Add(this.Line10);
      this.Controls.Add(this.Line8);
      this.Controls.Add(this.ErrorLabel);
      this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
      this.Location = new System.Drawing.Point(12, 460);
      this.Name = "busSelect";
      this.ShowInTaskbar = false;
      this.SizeGripStyle = System.Windows.Forms.SizeGripStyle.Show;
      this.StartPosition = System.Windows.Forms.FormStartPosition.Manual;
      this.Text = "LineSelect";
      this.ResumeLayout(false);

    }
    #endregion

    private void quit_Click(object sender, System.EventArgs e)
    {
      this.Close();
    }

    //! API functions to move the form (www.c-sharpcorner.com)
    public const int WM_NCLBUTTONDOWN = 0xA1;
    //! API functions to move the form
    public const int HTCAPTION = 0x2;
    [DllImport("user32.dll")]
    //! API functions to move the form
    public static extern bool ReleaseCapture();
    [DllImport("user32.dll")]
    //! API functions to move the form
    public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);
```

```
    //! User Interaction: Lets the user move the window
    private void SelectLinestitle_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)
    {
      //If the left mouse is pressed, release form for movement
      if (e.Button == MouseButtons.Left)
      {
        ReleaseCapture();
        SendMessage(Handle, WM_NCLBUTTONDOWN, HTCAPTION, 0);
      }
    }

    //! User Interaction: Checks all check boxes that are visible
    //! (boxes are not visible when lines not available in Load File mode)
    private void SelectAll_Click(object sender, System.EventArgs e)
    {
      for (int i=0; i < 16; i++)
      {
        if (LineArray[i].Visible == true)
          LineArray[i].Checked = true;
      }
    }

    //! User Interaction: Clears all check boxes that are visible
    //! (boxes are not visible when lines not available in Load File mode)
    private void ClearAll_Click(object sender, System.EventArgs e)
    {
      for (int i=0; i < 16; i++)
      {
        LineArray[i].Checked = false;
      }
    }
  }
}
```

USB Logic Analyzer

Appendix IV:

OKPipeRead Code [C++]

OKPipeRead.cpp

```
#include <iostream>
#include <fstream>
#include "okCUsbXEM.h"
#include <string>
using namespace std;

int main(int flashFPGA)
{
  string fileName;
  fileName = "./datareal.txt";

  string fileNamebd;
  fileNamebd = "./bufdump.txt";
  ofstream bufdump;
  cout << "Opening File...";
  bufdump.open(fileNamebd.c_str());
  if (!bufdump.is_open()) {
    cout << "ERROR: Cannot open file for writing." << endl;
    return(1);
  }
  cout << "OK" << endl;


  ofstream outFile;
  cout << "Opening File...";
  outFile.open(fileName.c_str());
  if (!outFile.is_open()) {
    cout << "ERROR: Cannot open file for writing." << endl;
    return(1);
  }
  cout << "OK" << endl;

  // --- Opal Kelly pipe code ---
  //  unsigned char *buf;
  long xfer_len = 1024;
  long pkt_len = 1024;
  long len;

  // Pipe Out Test
  unsigned char *buf2;
  buf2 = new unsigned char[pkt_len];

  // Perform transfer.
  bool ret2;

  cout << "Testing pipe read..." << endl;

  // Create an instance of the okCUsbXEM.
  okCUsbXEM *xem = new okCUsbXEM();

  // Open the first available device.
  bool check = xem->Open(0);
  if (check == false)
    return (4); //FPGA not found exit code


  if (flashFPGA == 0)
  {
    // Download a configuration file to the FPGA.
    check = xem->ConfigureFPGA("./pipetest.bit");
    if (check == false)
      cout << "bit file not found!" << endl;
  }


  xem->SetWireInValue(0x01,0x01);
  xem->UpdateWireIns();

  for (int i=0; i<xfer_len/pkt_len; i++)
  {
    len = pkt_len;
    ret2 = xem->ReadFromPipeOut(0xa0, len, buf2);
    if (false == ret2) {
      cout << "ReadFromPipeOut(...) failed.\n";
      break;
    }
    if (pkt_len != len) {
      cout << "Wanted " << pkt_len << " but got " << len << " (diff=" << pkt_len-len << ").\n";
      break;
    }
  }
```

```
  xem->SetWireInValue(0x01,0x00); // set the "write to buffer" signal on FPGA to '0'(ie don't write)
  xem->UpdateWireIns();

  // --- dump the whole buffer (unparsed) to a text file ---
  for(int i = 0;i<pkt_len;i++)
  {
    bufdump << buf2[i] << endl;
  }

  // --- Data parsing code for text file ---
  int numlines = 16;
  unsigned short parse_data;
  unsigned short mask;
  unsigned short pd;
  int count = 0;
  for(int i = 0;i<pkt_len; i++)
  {
    parse_data = buf2[i];
    mask = 0x80;
    for(int j = 0;j<8;j++)
    {
      pd = (parse_data & mask);
      pd = pd >> 7-j;
      mask = mask >> 1;
      switch (pd)
      {
      case 0x0:
        outFile << "-1,";
        break;
      case 0x1:
        outFile << "1,";
        break;
      default:
        outFile << "Probe Line Parsing Error";
      }
    }
    count++;
    if (count == 2)
    {
      outFile << endl;
      count = 0;
    }
  }
  delete buf2;
  cout << "OK" << endl;
  // --- End Opal Kelley Pipe Code ---

  bufdump.close();

  cout << "close file...";
  outFile.close();
  cout << "OK" << endl;
  //  int x;
  //  cin >> x;
  return (0);
}
```

USB Logic Analyzer

Appendix V:

FPGA Code [VHDL]

```
------------------------------------------------------------------------
-- FrontPanel Library Module Declarations (VHDL)
--
-- Copyright (c) 2004
-- Opal Kelly Incorporated
------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library UNISIM;
use UNISIM.VComponents.all;
entity okHostInterface is
  port (
    hi_clk    : in std_logic;
    hi_rdwr   : in std_logic;
    hi_cs   : in std_logic;
    hi_addr   : in std_logic_vector(3 downto 0);
    hi_data   : inout std_logic_vector(7 downto 0);
    hi_irq    : out std_logic;
    hi_busy   : out std_logic;
    ti_clk    : out std_logic;
    ti_control  : out std_logic_vector(12 downto 0);
    ti_data   : inout std_logic_vector(7 downto 0)
  );
end okHostInterface;

architecture arch of okHostInterface is
  component BUFG port (
    I : in std_logic;
    O : out std_logic);
  end component;

  component IOBUF port (
    T : in std_logic;
    O : out std_logic;
    I : in std_logic;
    IO : inout std_logic);
  end component;

  component okHostInterfaceCore port (
    hi_clk : in std_logic;
    hi_rdwr : in std_logic;
    hi_cs : in std_logic;
    hi_irq : out std_logic;
    hi_busy : out std_logic;
    hi_dir : out std_logic;
    hi_addr : in std_logic_vector(3 downto 0);
    hi_datain : in std_logic_vector(7 downto 0);
    hi_dataout : out std_logic_vector(7 downto 0);
    ti_control : out std_logic_vector(12 downto 0);
    ti_data : inout std_logic_vector(7 downto 0));
  end component;

  signal hi_datain : std_logic_vector(7 downto 0);
  signal hi_dataout : std_logic_vector(7 downto 0);
  signal hi_dir : std_logic;
  signal ti_clk_int : std_logic;
begin

ti_clk <= ti_clk_int;

-- Clock buffer for the Host Interface clock.
clkbuf : BUFG port map (I => hi_clk, O => ti_clk_int);

-- Instantiate bidirectional IOBUFs for the hi_data lines.
iobuf0 : IOBUF port map (T => hi_dir, O => hi_datain(0), I => hi_dataout(0), IO => hi_data(0) );
iobuf1 : IOBUF port map (T => hi_dir, O => hi_datain(1), I => hi_dataout(1), IO => hi_data(1) );
iobuf2 : IOBUF port map (T => hi_dir, O => hi_datain(2), I => hi_dataout(2), IO => hi_data(2) );
iobuf3 : IOBUF port map (T => hi_dir, O => hi_datain(3), I => hi_dataout(3), IO => hi_data(3) );
iobuf4 : IOBUF port map (T => hi_dir, O => hi_datain(4), I => hi_dataout(4), IO => hi_data(4) );
iobuf5 : IOBUF port map (T => hi_dir, O => hi_datain(5), I => hi_dataout(5), IO => hi_data(5) );
```

```
iobuf6 : IOBUF port map (T => hi_dir, O => hi_datain(6), I => hi_dataout(6), IO => hi_data(6) );
iobuf7 : IOBUF port map (T => hi_dir, O => hi_datain(7), I => hi_dataout(7), IO => hi_data(7) );

-- Instantiate the core Host Interface.
hicore : okHostInterfaceCore port map(
  hi_clk => ti_clk_int,
  hi_rdwr => hi_rdwr,
  hi_cs => hi_cs,
  hi_irq => hi_irq,
  hi_busy => hi_busy,
  hi_addr => hi_addr,
  hi_dir => hi_dir,
  hi_datain => hi_datain,
  hi_dataout => hi_dataout,
  ti_control => ti_control,
  ti_data => ti_data);

end arch;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okHostInterfaceCore is
  port (
    hi_clk    : in std_logic;
    hi_rdwr   : in std_logic;
    hi_cs   : in std_logic;
    hi_irq    : out std_logic;
    hi_busy   : out std_logic;
    hi_dir    : out std_logic;
    hi_addr   : in std_logic_vector(3 downto 0);
    hi_datain : in std_logic_vector(7 downto 0);
    hi_dataout  : out std_logic_vector(7 downto 0);
    ti_control  : out std_logic_vector(12 downto 0);
    ti_data   : inout std_logic_vector(7 downto 0)
  );
end okHostInterfaceCore;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okWireIn is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : in std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_dataout  : out std_logic_vector(7 downto 0)
  );
end okWireIn;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okWireOut is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_datain : in std_logic_vector(7 downto 0)
  );
end okWireOut;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okTriggerIn is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
```

```
    ti_data   : in std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_clk    : in std_logic;
    ep_trigger  : out std_logic_vector(7 downto 0)
  );
end okTriggerIn;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okTriggerOut is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_clk    : in std_logic;
    ep_trigger  : in std_logic_vector(7 downto 0)
  );
end okTriggerOut;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okPipeIn is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : in std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_write  : out std_logic;
    ep_dataout  : out std_logic_vector(7 downto 0)
  );
end okPipeIn;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okPipeOut is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_read   : out std_logic;
    ep_datain : in std_logic_vector(7 downto 0)
  );
end okPipeOut;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity okBufferedPipeIn is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_clk    : in std_logic;
    ep_reset  : in std_logic;
    ep_read   : in std_logic;
    ep_dataout  : out std_logic_vector(7 downto 0);
    ep_full   : out std_logic;
    ep_empty  : out std_logic;
    ep_status : out std_logic_vector(3 downto 0)
  );
end okBufferedPipeIn;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity okBufferedPipeOut is
  port (
    ti_clk    : in std_logic;
    ti_control  : in std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in std_logic_vector(7 downto 0);
    ep_clk    : in std_logic;
    ep_reset  : in std_logic;
    ep_write  : in std_logic;
    ep_datain : in std_logic_vector(7 downto 0);
    ep_full   : out std_logic;
    ep_empty  : out std_logic;
    ep_status : out std_logic_vector(3 downto 0)
  );
end okBufferedPipeOut;


------------------------------------------------------------------------
-- PipeTest.vhd
--
-- tabstop: 3
--
-- Copyright (c) 2004
-- Opal Kelly Incorporated
------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_misc.all;
use IEEE.std_logic_unsigned.all;
entity PipeTest is
  port (
    hi_clk    : in  STD_LOGIC;
    hi_cs     : in  STD_LOGIC;
    hi_rdwr   : in  STD_LOGIC;
    hi_busy   : out STD_LOGIC;
    hi_irq    : out STD_LOGIC;
    hi_addr   : in  STD_LOGIC_VECTOR(3 downto 0);
    hi_data   : inout STD_LOGIC_VECTOR(7 downto 0);

    clk1      : in STD_LOGIC;
    start     : in STD_LOGIC;
    stop      : in STD_LOGIC;
    pause     : in STD_LOGIC;
    ybus      : out STD_LOGIC_VECTOR(15 downto 0);
    led     : out STD_LOGIC_VECTOR(7 downto 0)
  );
end PipeTest;

architecture arch of PipeTest is
  component okHostInterface port (
    hi_clk    : in  std_logic;
    hi_rdwr   : in  std_logic;
    hi_cs     : in  std_logic;
    hi_irq    : out std_logic;
    hi_busy   : out std_logic;
    hi_addr   : in  std_logic_vector(3 downto 0);
    hi_data   : inout std_logic_vector(7 downto 0);
    ti_clk    : out std_logic;
    ti_control  : out std_logic_vector(12 downto 0);
    ti_data   : inout std_logic_vector(7 downto 0));
  end component;

  component okWireIn port (
    ti_clk    : in  std_logic;
    ti_control  : in  std_logic_vector(12 downto 0);
    ti_data   : in  std_logic_vector(7 downto 0);
    ep_addr   : in  std_logic_vector(7 downto 0);
    ep_dataout  : out std_logic_vector(7 downto 0));
  end component;
```

```
  component okWireOut port (
    ti_clk    : in  std_logic;
    ti_control  : in  std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in  std_logic_vector(7 downto 0);
    ep_datain : in  std_logic_vector(7 downto 0));
  end component;

  component okTriggerIn port (
    ti_clk    : in  std_logic;
    ti_control  : in  std_logic_vector(12 downto 0);
    ti_data   : in  std_logic_vector(7 downto 0);
    ep_addr   : in  std_logic_vector(7 downto 0);
    ep_clk    : in  std_logic;
    ep_trigger  : out std_logic_vector(7 downto 0));
  end component;

  component okPipeIn port (
    ti_clk    : in  std_logic;
    ti_control  : in  std_logic_vector(12 downto 0);
    ti_data   : in  std_logic_vector(7 downto 0);
    ep_addr   : in  std_logic_vector(7 downto 0);
    ep_write    : out std_logic;
    ep_dataout  : out std_logic_vector(7 downto 0));
  end component;

  component okPipeOut port (
    ti_clk    : in  std_logic;
    ti_control  : in  std_logic_vector(12 downto 0);
    ti_data   : out std_logic_vector(7 downto 0);
    ep_addr   : in  std_logic_vector(7 downto 0);
    ep_read   : out std_logic;
    ep_datain : in  std_logic_vector(7 downto 0));
  end component;

  component RAMB16_S9_S9 port (
    DIA : in  STD_LOGIC_VECTOR (7 downto 0);
    DOA : out STD_LOGIC_VECTOR (7 downto 0);
    DIPA  : in  STD_LOGIC_VECTOR (0 downto 0);
    DOPA  : out STD_LOGIC_VECTOR (0 downto 0);
    DIB : in  STD_LOGIC_VECTOR (7 downto 0);
    DOB : out STD_LOGIC_VECTOR (7 downto 0);
    DIPB  : in  STD_LOGIC_VECTOR (0 downto 0);
    DOPB  : out STD_LOGIC_VECTOR (0 downto 0);
    ADDRA : in  STD_LOGIC_VECTOR (10 downto 0);
    ADDRB : in  STD_LOGIC_VECTOR (10 downto 0);
    CLKA  : in  STD_LOGIC;
    CLKB  : in  STD_LOGIC;
    ENA : in  STD_LOGIC;
    ENB : in  STD_LOGIC;
    SSRA  : in  STD_LOGIC;
    SSRB  : in  STD_LOGIC;
    WEA : in  STD_LOGIC;
    WEB : in  STD_LOGIC);
  end component;

  signal ti_clk : STD_LOGIC;
  signal ti_control : STD_LOGIC_VECTOR(12 downto 0);
  signal ti_data : STD_LOGIC_VECTOR(7 downto 0);

  signal ep00wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep20wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep21wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep22wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep23wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep24wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep25wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep26wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep27wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep28wire : STD_LOGIC_VECTOR(7 downto 0);
```

```
  signal ep29wire : STD_LOGIC_VECTOR(7 downto 0);
  signal ep40trig : STD_LOGIC_VECTOR(7 downto 0);
  signal pipeI_data : STD_LOGIC_VECTOR(7 downto 0);
  signal pipeO_data : STD_LOGIC_VECTOR(7 downto 0);
  signal pipeA1_data : STD_LOGIC_VECTOR(7 downto 0);
  signal pipeI_write : STD_LOGIC;
  signal pipeO_read : STD_LOGIC;
  signal pipeA1_read : STD_LOGIC;

  signal count_pipeI : STD_LOGIC_VECTOR(31 downto 0);
  signal count_pipeO : STD_LOGIC_VECTOR(31 downto 0);
  signal count_reset : STD_LOGIC;
  signal pipeI_xor : STD_LOGIC_VECTOR(7 downto 0);
  signal pulse : STD_LOGIC_VECTOR(27 downto 0);
  signal tick : STD_LOGIC;

  ----------------------------------------------------
  signal run    : STD_LOGIC;               -- control bit for counter status
  signal divider : STD_LOGIC_VECTOR (3 downto 0);
  ----------------------------------------------------
begin

-- Pulse counter
--led <= count_pipeO(7 downto 0);
ep24wire <= tick & tick & tick & tick & tick & tick & tick & tick;
process (clk1) begin
  if rising_edge(clk1) then
    pulse <= pulse - "1";
    if (pulse = x"0000000") then
      pulse <= x"17d7840";  -- Integer 25,000,000
      tick <= not tick;
    end if;
  end if;
end process;


-- IN Pipe counter
count_reset <= ep40trig(0);
ep20wire <= count_pipeI(7 downto 0);
ep21wire <= count_pipeI(15 downto 8);
ep22wire <= count_pipeI(23 downto 16);
ep23wire <= count_pipeI(31 downto 24);
ep25wire <= pipeI_xor;
process (ti_clk) begin
  if rising_edge(ti_clk) then
    if (count_reset = '1') then
      count_pipeI <= x"00000000";
      pipeI_xor <= x"00";
    elsif (pipeI_write = '1') then
      count_pipeI <= count_pipeI + "1";
      pipeI_xor <= pipeI_xor xor pipeI_data;
    end if;
  end if;
end process;


-- OUT Pipe counter
pipeO_data <= count_pipeO(7 downto 0);
ep26wire <= count_pipeO(7 downto 0);
ep27wire <= count_pipeO(15 downto 8);
ep28wire <= count_pipeO(23 downto 16);
ep29wire <= count_pipeO(31 downto 24);
process (clk1) begin
  if rising_edge(clk1) then
    if (start = '0') then
      run <= '0';
    elsif (stop = '0') then
      run <= '1';
    end if;

    if (count_reset = '1') then
```

```
      count_pipeO <= x"00000000";
      run <= '1';
      divider <= x"F";
    elsif (run = '0' and pause = '1') then
      if (divider = x"F") then
--      if (count_pipeO = x"000000FF") then
--        count_pipeO <= x"00000000";
--      else
        count_pipeO <= count_pipeO + "1";
--      end if;
--        divider <= "0000";
      else
        divider <= divider + "1";
      end if;
    end if;
  end if;
end process;
led <= not count_pipeO;
ybus <= count_pipeO(15 downto 0);
-------------------------------------------------
-- Instantiate the input block RAM
-- Interface A is a write-only interface from the Pipe In (0x80).
-- Interface B is a read-only interface to the Pipe Out (0xA1).
ram_I : RAMB16_S9_S9 port map(
    CLKA => ti_clk, SSRA => count_reset, ENA => '1',
    WEA => pipeI_write, ADDRA => count_pipeI(10 downto 0),
    DIA => pipeI_data, DIPA => "0",
    CLKB => ti_clk, SSRB => count_reset, ENB => '1',
    WEB => '0', ADDRB => count_pipeO(10 downto 0),
    DIB => x"00", DIPB => "0", DOB => pipeA1_data);

-- Instantiate the okHostInterface and connect endpoints to
-- the target interface.
okHI : okHostInterface port map (
    hi_clk => hi_clk,
    hi_rdwr => hi_rdwr,
    hi_cs => hi_cs,
    hi_irq => hi_irq,
    hi_busy => hi_busy,
    hi_addr => hi_addr,
    hi_data => hi_data,
    ti_clk => ti_clk,
    ti_control => ti_control,
    ti_data => ti_data
  );

ep00 : okWireIn port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"00", ep_dataout => ep00wire);

ep40 : okTriggerIn port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"40", ep_clk => ti_clk, ep_trigger => ep40trig);

ep20 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"20", ep_datain => ep20wire);
ep21 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"21", ep_datain => ep21wire);
ep22 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"22", ep_datain => ep22wire);
ep23 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"23", ep_datain => ep23wire);
ep24 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"24", ep_datain => ep24wire);
ep25 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
```

```
    ep_addr => x"25", ep_datain => ep25wire);
ep26 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"26", ep_datain => ep26wire);
ep27 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"27", ep_datain => ep27wire);
ep28 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"28", ep_datain => ep28wire);
ep29 : okWireOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"29", ep_datain => ep29wire);

ep80 : okPipeIn port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"80", ep_write => pipeI_write, ep_dataout => pipeI_data);

epA0 : okPipeOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"A0", ep_read => pipeO_read, ep_datain => pipeO_data);
epA1 : okPipeOut port map (
    ti_clk => ti_clk, ti_control => ti_control, ti_data => ti_data,
    ep_addr => x"A1", ep_read => pipeA1_read, ep_datain => pipeA1_data);

end arch;
```

USB Logic Analyzer

Appendix VI:

FPGA Pin Listing [UCF]

```
#------------------------------------------------------------------------
# XEM3001 - Xilinx constraints file
#
# Pin mappings for the XEM3001.  Use this as a template and comment out
# the pins that are not used in your design.  (By default, map will fail
# if this file contains constraints for signals not in your design).
#------------------------------------------------------------------------

#------------------------------
# FrontPanel Host Interface pins
#------------------------------
NET "hi_clk"      LOC = "P79";
NET "hi_cs"       LOC = "P57";
NET "hi_rdwr"     LOC = "P58";
NET "hi_busy"     LOC = "P81";
NET "hi_irq"      LOC = "P85";
NET "hi_addr<0>"  LOC = "P64";
NET "hi_addr<1>"  LOC = "P63";
NET "hi_addr<2>"  LOC = "P62";
NET "hi_addr<3>"  LOC = "P61";
NET "hi_data<0>"  LOC = "P67";
NET "hi_data<1>"  LOC = "P68";
NET "hi_data<2>"  LOC = "P72";
NET "hi_data<3>"  LOC = "P74";
NET "hi_data<4>"  LOC = "P86";
NET "hi_data<5>"  LOC = "P87";
NET "hi_data<6>"  LOC = "P90";
NET "hi_data<7>"  LOC = "P92";

#---------------
# PLL Clock pins
#---------------
NET "clk1"    LOC = "P80";
#NET "clk2"   LOC = "P77";
#NET "clk3"   LOC = "P76";

#----------------
# Port JP1 (zbus)
#----------------
#NET "zclk1"  LOC = "P183";
#NET "zclk2"  LOC = "P181";
#NET "zbus<0>"  LOC = "P187";
#NET "zbus<1>"  LOC = "P185";
#NET "zbus<2>"  LOC = "P182";
#NET "zbus<3>"  LOC = "P178";
#NET "zbus<4>"  LOC = "P176";
#NET "zbus<5>"  LOC = "P175";
#NET "zbus<6>"  LOC = "P172";
#NET "zbus<7>"  LOC = "P171";
#NET "zbus<8>"  LOC = "P169";
#NET "zbus<9>"  LOC = "P168";
#NET "zbus<10>" LOC = "P167";
#NET "zbus<11>" LOC = "P166";
#NET "zbus<12>" LOC = "P165";
#NET "zbus<13>" LOC = "P162";

#----------------
# Port JP2 (ybus)
#----------------
#NET "yclk1"    LOC = "P184";
NET "ybus<0>" LOC = "P52";
NET "ybus<1>" LOC = "P51";
NET "ybus<2>" LOC = "P50";
NET "ybus<3>" LOC = "P48";
NET "ybus<4>" LOC = "P46";
NET "ybus<5>" LOC = "P45";
NET "ybus<6>" LOC = "P44";
NET "ybus<7>" LOC = "P43";
NET "ybus<8>" LOC = "P42";
NET "ybus<9>" LOC = "P40";
NET "ybus<10>"  LOC = "P39";
```

```
NET "ybus<11>"  LOC = "P37";
NET "ybus<12>"  LOC = "P36";
NET "ybus<13>"  LOC = "P35";
NET "ybus<14>"  LOC = "P34";
NET "ybus<15>"  LOC = "P33";
#NET "ybus<16>" LOC = "P31";
#NET "ybus<17>" LOC = "P29";
#NET "ybus<18>" LOC = "P28";
#NET "ybus<19>" LOC = "P27";
#NET "ybus<20>" LOC = "P26";
#NET "ybus<21>" LOC = "P24";
#NET "ybus<22>" LOC = "P22";
#NET "ybus<23>" LOC = "P21";
#NET "ybus<24>" LOC = "P20";
#NET "ybus<25>" LOC = "P19";
#NET "ybus<26>" LOC = "P18";
#NET "ybus<27>" LOC = "P16";
#NET "ybus<28>" LOC = "P15";
#NET "ybus<29>" LOC = "P13";
#NET "ybus<30>" LOC = "P12";
#NET "ybus<31>" LOC = "P11";
#NET "ybus<32>" LOC = "P10";
#NET "ybus<33>" LOC = "P9";
#NET "ybus<34>" LOC = "P7";
#NET "ybus<35>" LOC = "P5";

#----------------
# Port JP3 (xbus)
#----------------
#NET "xclk1"  LOC = "P180";
#NET "xbus<0>"  LOC = "P156";
#NET "xbus<1>"  LOC = "P155";
#NET "xbus<2>"  LOC = "P154";
#NET "xbus<3>"  LOC = "P152";
#NET "xbus<4>"  LOC = "P150";
#NET "xbus<5>"  LOC = "P149";
#NET "xbus<6>"  LOC = "P148";
#NET "xbus<7>"  LOC = "P147";
#NET "xbus<8>"  LOC = "P146";
#NET "xbus<9>"  LOC = "P144";
#NET "xbus<10>" LOC = "P143";
#NET "xbus<11>" LOC = "P141";
#NET "xbus<12>" LOC = "P140";
#NET "xbus<13>" LOC = "P139";
#NET "xbus<14>" LOC = "P138";
#NET "xbus<15>" LOC = "P137";
#NET "xbus<16>" LOC = "P135";
#NET "xbus<17>" LOC = "P133";
#NET "xbus<18>" LOC = "P132";
#NET "xbus<19>" LOC = "P131";
#NET "xbus<20>" LOC = "P130";
#NET "xbus<21>" LOC = "P128";
#NET "xbus<22>" LOC = "P126";
#NET "xbus<23>" LOC = "P125";
#NET "xbus<24>" LOC = "P124";
#NET "xbus<25>" LOC = "P123";
#NET "xbus<26>" LOC = "P122";
#NET "xbus<27>" LOC = "P120";
#NET "xbus<28>" LOC = "P119";
#NET "xbus<29>" LOC = "P117";
#NET "xbus<30>" LOC = "P116";
#NET "xbus<31>" LOC = "P115";
#NET "xbus<32>" LOC = "P114";
#NET "xbus<33>" LOC = "P113";
#NET "xbus<34>" LOC = "P111";
#NET "xbus<35>" LOC = "P109";

#------------
# Peripherals
#------------
NET "led<0>"  LOC = "P205";
```

File: xem3001v1.ucf

```
NET "led<1>"  LOC = "P204";
NET "led<2>"  LOC = "P203";
NET "led<3>"  LOC = "P200";
NET "led<4>"  LOC = "P199";
NET "led<5>"  LOC = "P198";
NET "led<6>"  LOC = "P197";
NET "led<7>"  LOC = "P196";

NET "start" LOC = "P194";
NET "stop" LOC = "P191";
NET "pause" LOC = "P190";
#NET "button<3>" LOC = "P189";
```

USB Logic Analyzer

Appendix VII:

Code Documentation [HTML]

Documentation generated by Doxygen can be viewed at the project website:
http://cegt201.bradley.edu/projects/proj2006/usbla/documentation/

USB Logic Analyzer

Appendix VIII:

Original Functional Description

USB Logic Analyzer

Functional Description

Shom Bandopadhaya
Advisor: Dr. James H. Irwin

October 26, 2005

Senior Capstone Project
Bradley University
Department of Electrical and Computer Engineering

Introduction

A logic analyzer displays logic level of digital signals. A logic analyzer differs from an oscilloscope which is a more powerful instrument but typically only has a few channels. This project seeks to continue work on a digital logic analyzer which has sixteen channels. The logic analyzer will display four logic levels: low, high, tri-state and indeterminate. The sixteen channels are sampled by an external conditioning hardware called a POD. The POD consists of an Opal Kelly XEM 3001 FPGA board and some external hardware. The POD interfaces to the computer using Universal Serial Bus [USB] protocol. The graphical user interface [GUI] on the computer will display one of the four possible logic levels sampled by the POD. The overall system block diagram can be seen in Figure 1: Overall system block diagram for USB Logic Analyzer.

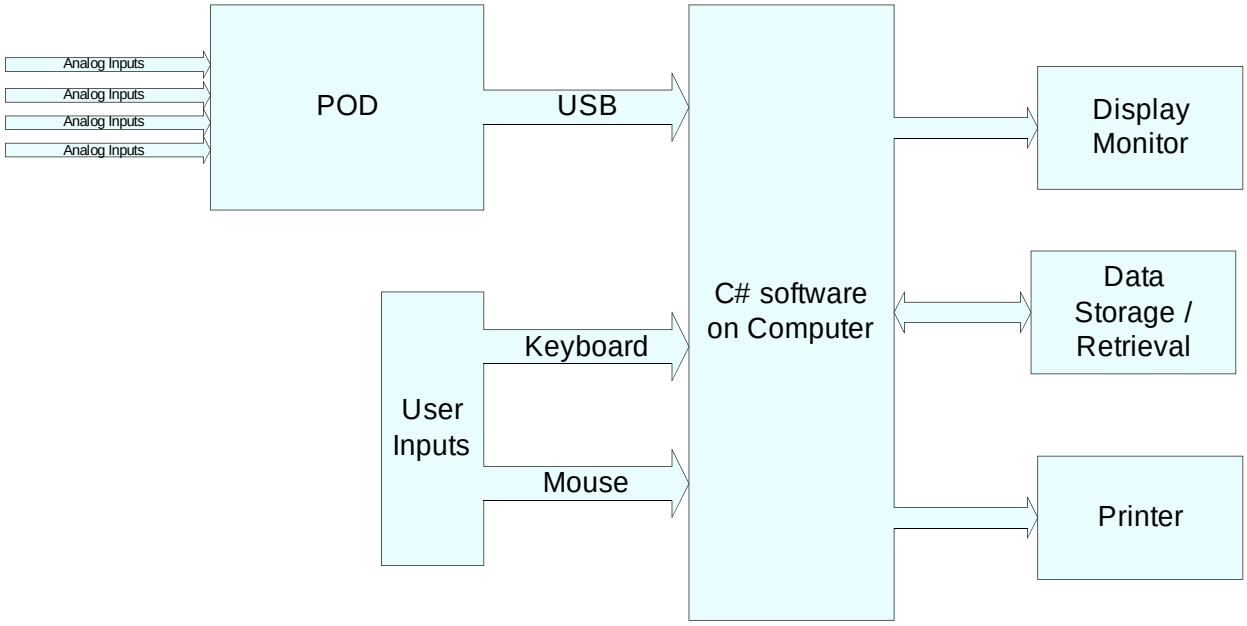


Figure 1: Overall system block diagram for USB Logic Analyzer

Inputs

Analog inputs:	These are the analog voltages on the probes of the POD. The POD handles CMOS or TTL levels.
User inputs:	The user interacts with the software GUI using the keyboard and mouse, and is able to select options, change display parameters, and access additional features like zoom, scroll and save.

Outputs

Display Monitor:	Displays the GUI. The user can see the current logic levels, triggering points and also observe the effects of their interactions with the GUI.
Data Storage/Retrieval:	Saves the captured data frame to disk. The GUI can also open and display previously saved captures.
Printer:	Prints the current screen.

Operation Modes

Option selection:	Allows the user to select from the available options, such as logic type (TTL or CMOS), display window range, etc.
Single Capture:	Captures and displays information from the time the capture was started till the time software buffer was filled.
Continuous Capture:	Continuously captures information into the buffer and displays the current buffer, so the logic levels are shown at “pseudo real-time”.

Goals

- Primary
    - New display engine
    - DLL interface for USB
    - Efficient data processing
    - Print/Save data capture
    - Continuous Update (Pseudo Real Time)
- Secondary
    - Remote Viewing
    - Reverse Assembly code decoding

USB Logic Analyzer

Appendix IX:

Original System Block Diagram

USB Logic Analyzer

System Block Diagram

Shom Bandopadhaya
Advisor: Dr. James H. Irwin

November 22, 2005

Senior Capstone Project
Bradley University
Department of Electrical and Computer Engineering

Introduction

A logic analyzer displays logic level of digital signals. A logic analyzer differs from an oscilloscope which is a more powerful instrument but typically only has a few channels. This project seeks to continue work on a digital logic analyzer which has sixteen channels. The logic analyzer will display four logic levels: low, high, tri-state and indeterminate. The sixteen channels are sampled by an external conditioning hardware called a POD. The POD consists of an Opal Kelly XEM 3001 FPGA board and some external hardware. The POD interfaces to the computer using Universal Serial Bus [USB] protocol. The graphical user interface [GUI] on the computer will display one of the four possible logic levels sampled by the POD. The overall system block diagram can be seen in Figure 1.

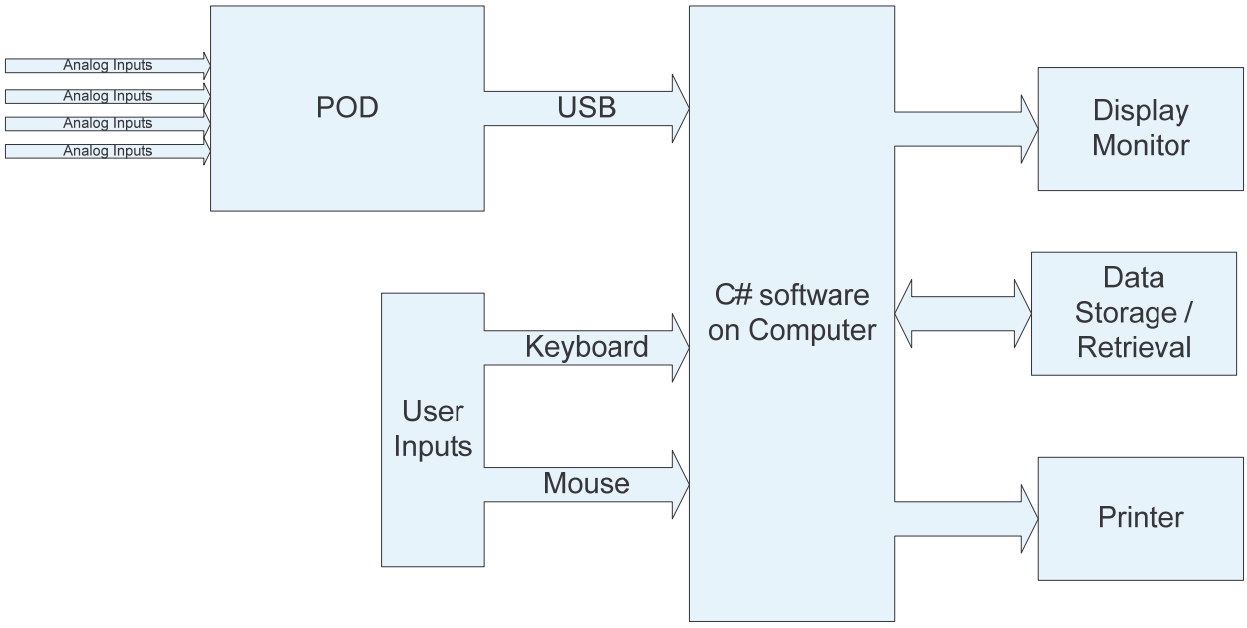


Figure 1: Overall system block diagram for USB Logic Analyzer

Inputs

Analog inputs:	These are the analog voltages on the probes of the POD. The POD handles CMOS or TTL levels.
User inputs:	The user interacts with the software GUI using the keyboard and mouse, and is able to select options, change display parameters, and access additional features like zoom, scroll and save.

Outputs

Display Monitor:	Displays the GUI. The user can see the current logic levels, triggering points and also observe the effects of their interactions with the GUI.
Data Storage/Retrieval:	Saves the captured data frame to disk. The GUI can also open and display previously saved captures.
Printer:	Prints the current screen.

Operation Modes

Option selection:	Allows the user to select from the available options, such as logic type (TTL or CMOS), display window range, etc.
Single Capture:	Captures and displays information from the time the capture was started till the time software buffer was filled.
Continuous Capture:	Continuously captures information into the buffer and displays the current buffer, so the logic levels are shown at “pseudo real-time”.

Goals

- Primary
    - New display engine
    - DLL interface for USB
    - Efficient data processing
    - Print/Save data capture
    - Continuous Update (Pseudo Real Time)
- Secondary
    - Remote Viewing
    - Reverse Assembly code decoding

Software

Conditioned Signals: Signals sent from the POD to the PC with all sampled data in packet form USB interface.

Keyboard/Mouse: Commands entered using keyboard and mouse to setup trigger conditions, change display format, save the waveform, setup, and hardcopies.

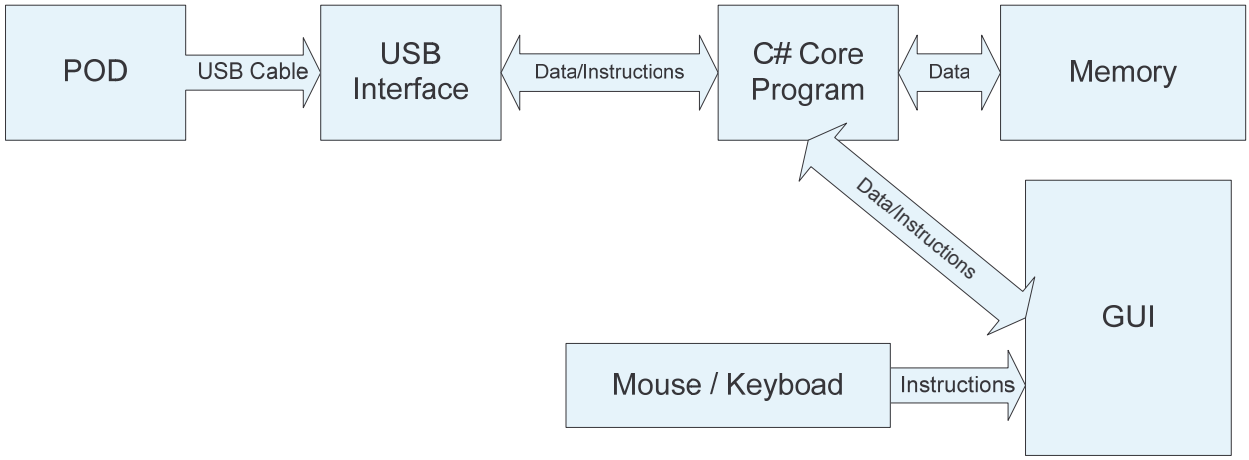


Figure 4.1 Software Subsystem Block Diagram

There are three major sub-blocks, the USB Interface, the C# Core Program, and the GUI as can be seen in Figure 4.1.

USB Interface

The USB interface uses the Opal Kelly provided library to connect to the FPGA using a USB 2.0 connection. It acquires data when requested by the C# core program and transmits the data back to the C# core program through the USB buffer. Figure 5.1 shows the process through which it is accomplished.

C# Core Program

The C# Core Program receives the instructions via the GUI, contacts the USB interface to start Data Acquisition, collects the data and then saves it to memory and sends it to the GUI to be displayed. Figure 5.2 describes the process.

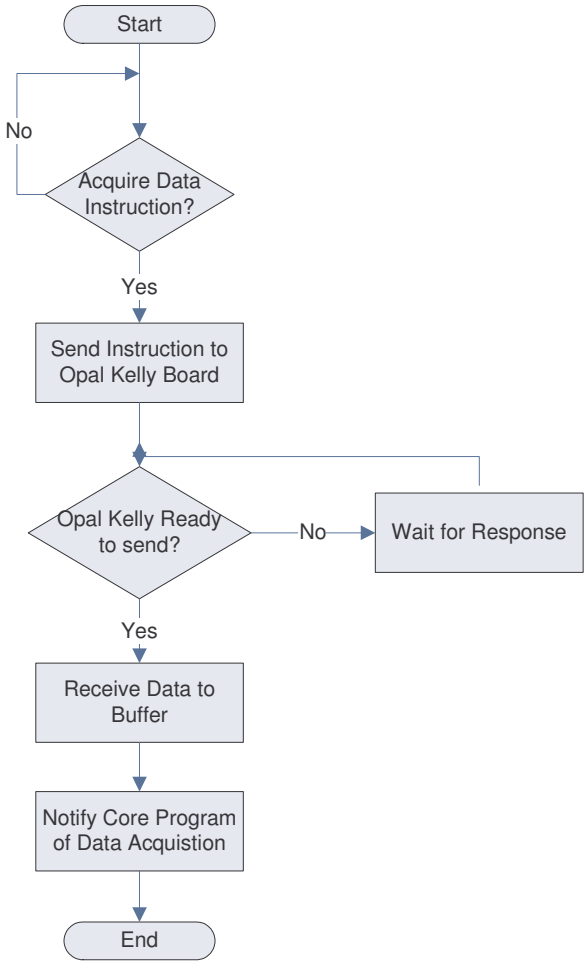


Figure 5.1 USB Interface Tasks

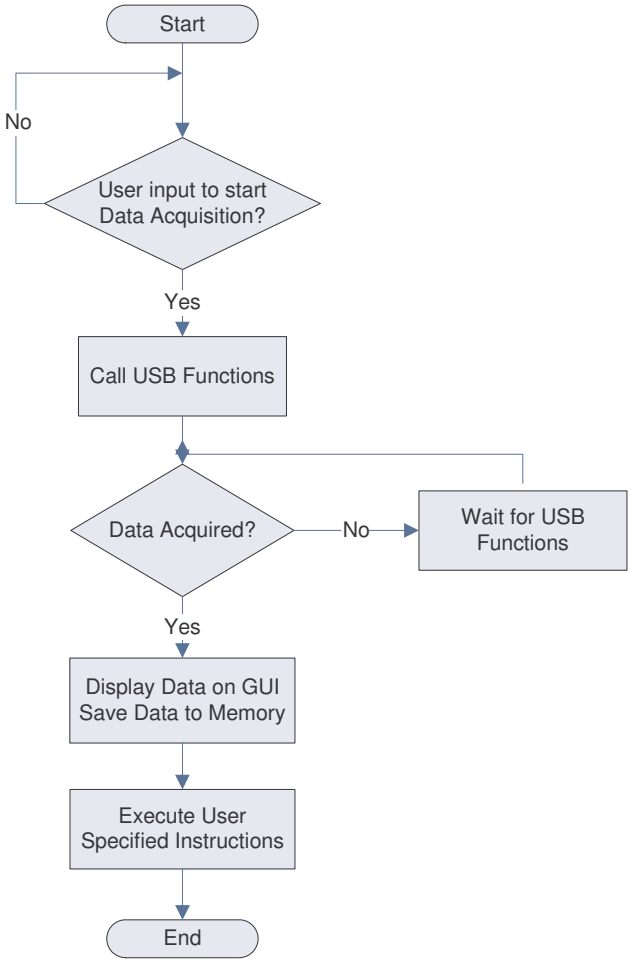


Figure 5.2 C# Core Program Tasks

GUI

The GUI interfaces the user to the C# core program. It receives instructions via the keyboard and mouse and displays the information on the screen. Once it receives any instructions it contacts the Core C# Program to perform those tasks. It idles till it receives the next set of instructions. If the user decides to terminate the GUI it terminates the whole software. Figure 6.1 shows this process.

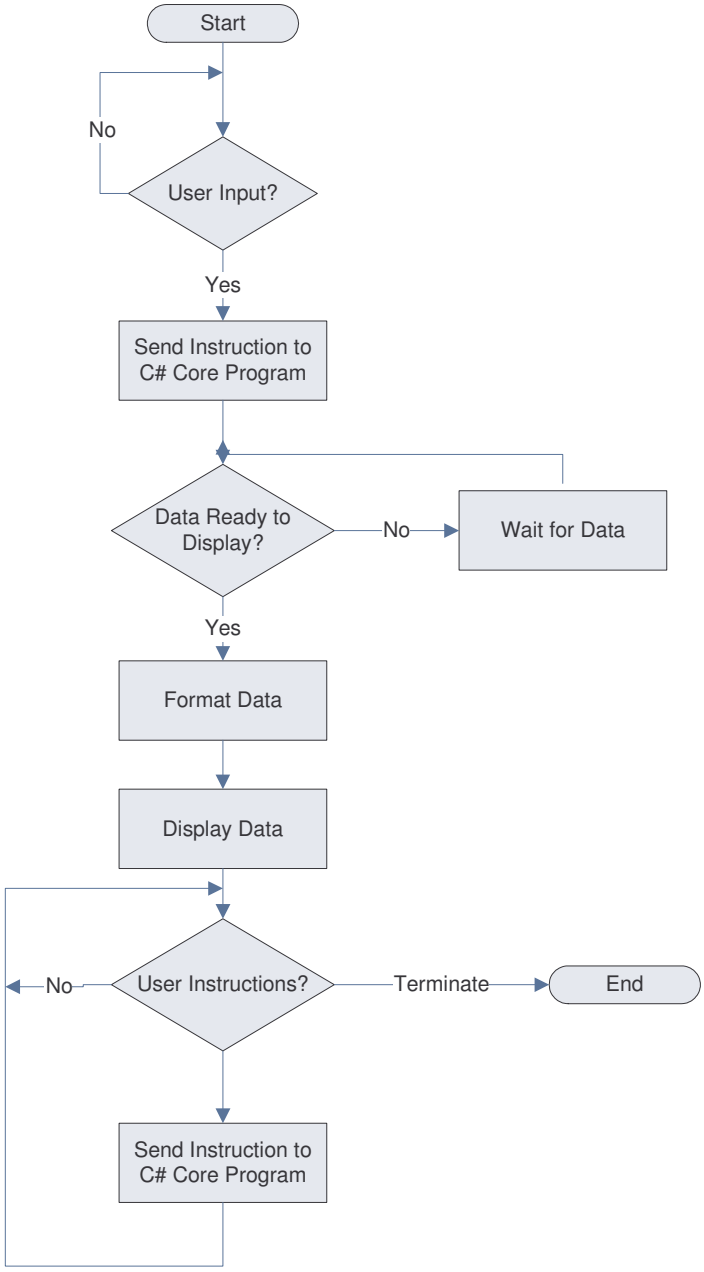


Figure 6.1 GUI Tasks

USB Logic Analyzer

Appendix X:

Original Project Proposal

USB Logic Analyzer

Project Proposal

Shom Bandopadhaya
Advisor: Dr. James H. Irwin

December 10, 2005

Senior Capstone Project
Bradley University
Department of Electrical and Computer Engineering

Introduction

A logic analyzer displays logic level of digital signals. A logic analyzer differs from an oscilloscope which is a more powerful instrument but typically only has a few channels. This project seeks to continue work on a digital logic analyzer which has sixteen channels. The logic analyzer will display four logic levels: low, high and indeterminate. The sixteen channels are sampled by external conditioning hardware called a POD. The POD consists of an Opal Kelly XEM 3001 FPGA board and some additional hardware. The POD interfaces to the computer using the Universal Serial Bus [USB] protocol. The graphical user interface [GUI] on the computer will display one of the four possible logic levels sampled by the POD. The overall system block diagram can be seen in Figure 1.

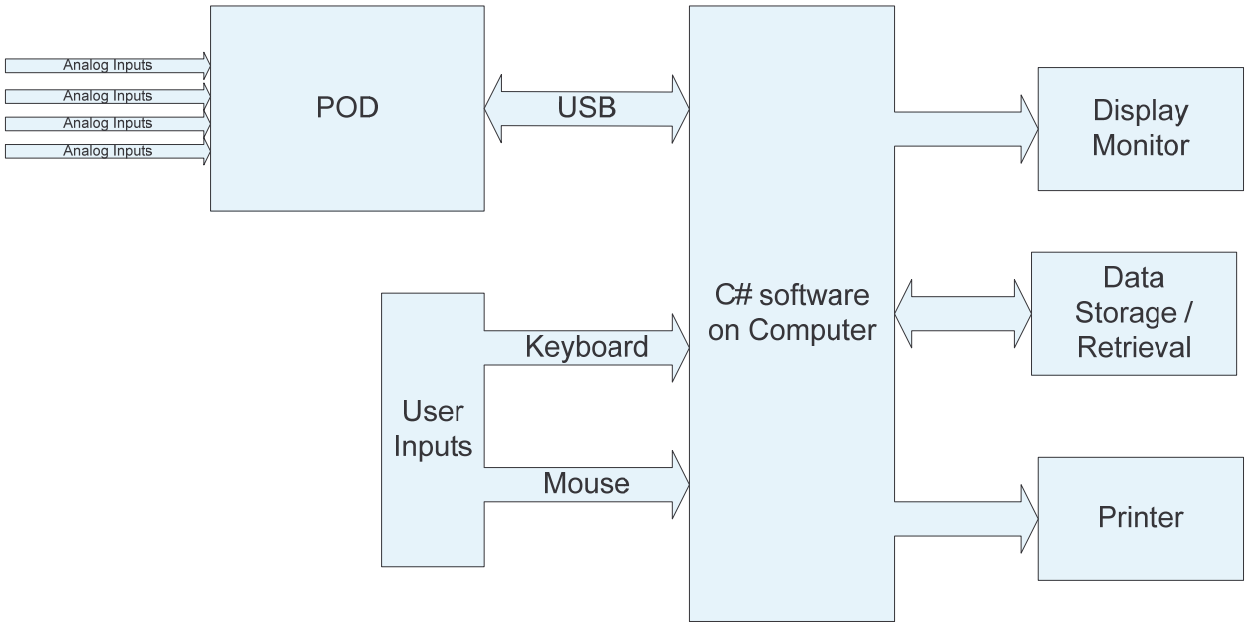


Figure 1: Overall system block diagram for USB Logic Analyzer

Inputs

Analog inputs:	These are the analog voltages on the probes of the POD. The POD can properly CMOS or TTL levels.
User inputs:	The user interacts with the software GUI using the keyboard and mouse, and is able to select options, change display parameters, and gain access additional features like zoom, scroll and save.

Outputs

Display Monitor:	Displays the GUI. The user can see the current logic levels, triggering points and also observe the effects of their interactions with the GUI.
Data Storage/Retrieval:	Saves the captured data frame to disk. The GUI can also open and display previously saved captures.
Printer:	Prints the current screen.

Operation Modes

Option selection:	Allows the user to select from the available options, such as logic type (TTL or CMOS), display window range, etc.
Single Capture:	Captures and displays information from the time the capture was started till the time software buffer was filled.
Continuous Capture:	Continuously captures information into the buffer and displays the current buffer, so the logic levels are shown at “pseudo real-time”.

Goals

- Primary
    - New display engine
    - DLL interface for USB
    - Efficient data processing
    - Print/Save data capture
    - Continuous Update (Pseudo Real Time)

Software

Conditioned Signals:	Signals sent from the POD to the PC with all sampled data in packet form USB interface.
Keyboard/Mouse:	Commands entered using keyboard and mouse to setup trigger conditions, change display format, save the waveform, setup, and hardcopies.

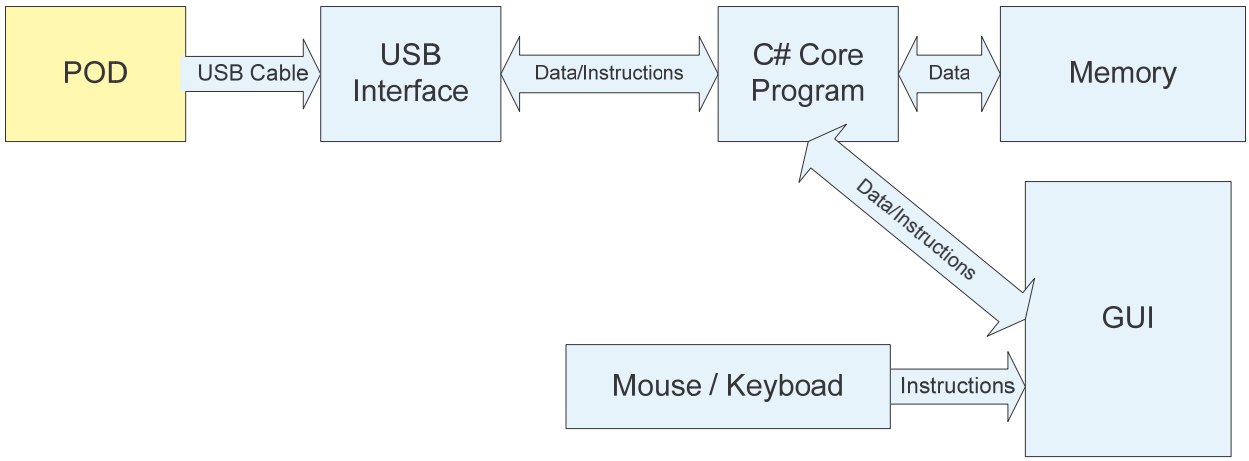


Figure 3.1 Software Subsystem Block Diagram

The USB Interface, the C# Core Program, and the GUI can be seen in Figure 3.1.

USB Interface

The USB interface uses the Opal Kelly provided library to connect to the FPGA using a USB 2.0 connection. It acquires data when requested by the C# core program and transmits the data back to the C# core program through the USB buffer. Figure 4.1 shows the process through which it is accomplished.

C# Core Program

The C# Core Program receives the instructions via the GUI, contacts the USB interface to start Data Acquisition, collects the data and then saves it to memory and sends it to the GUI to be displayed. Figure 4.2 describes the process.

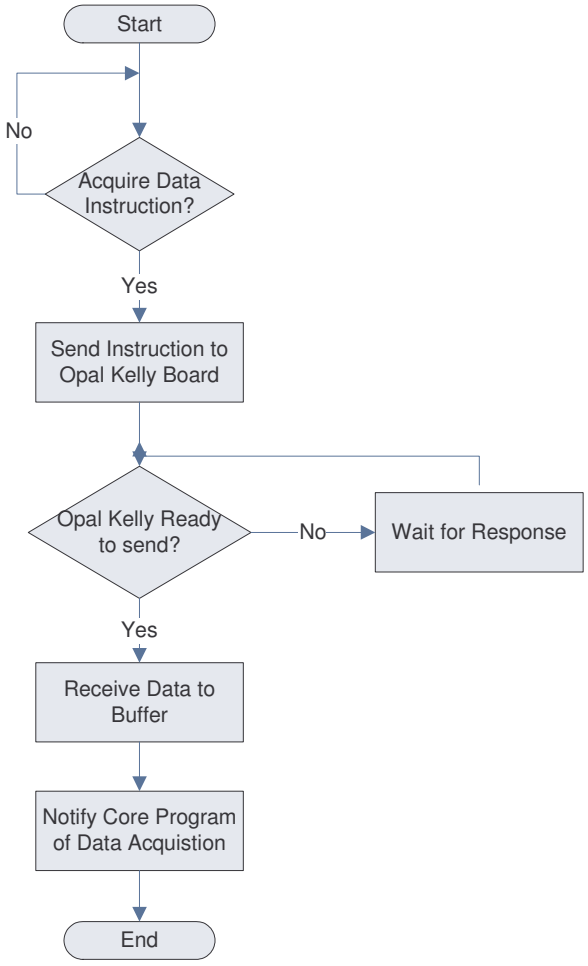


Figure 4.1 USB Interface Tasks

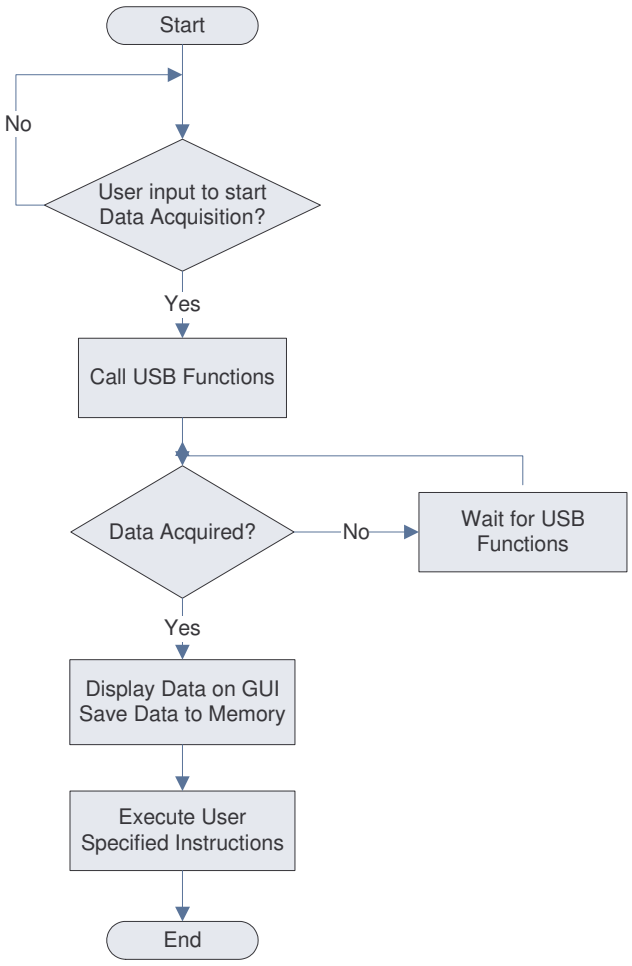


Figure 4.2 C# Core Program Tasks

GUI

The GUI permits the user to gain access to the functionality provided by the C# core program. It receives instructions via the keyboard and mouse and displays the information on the screen. Once it receives any instructions it contacts the Core C# Program to perform those tasks. It idles till it receives the next set of instructions. If the user decides to terminate the GUI it terminates the whole application. Figure 5.1 shows these processes.

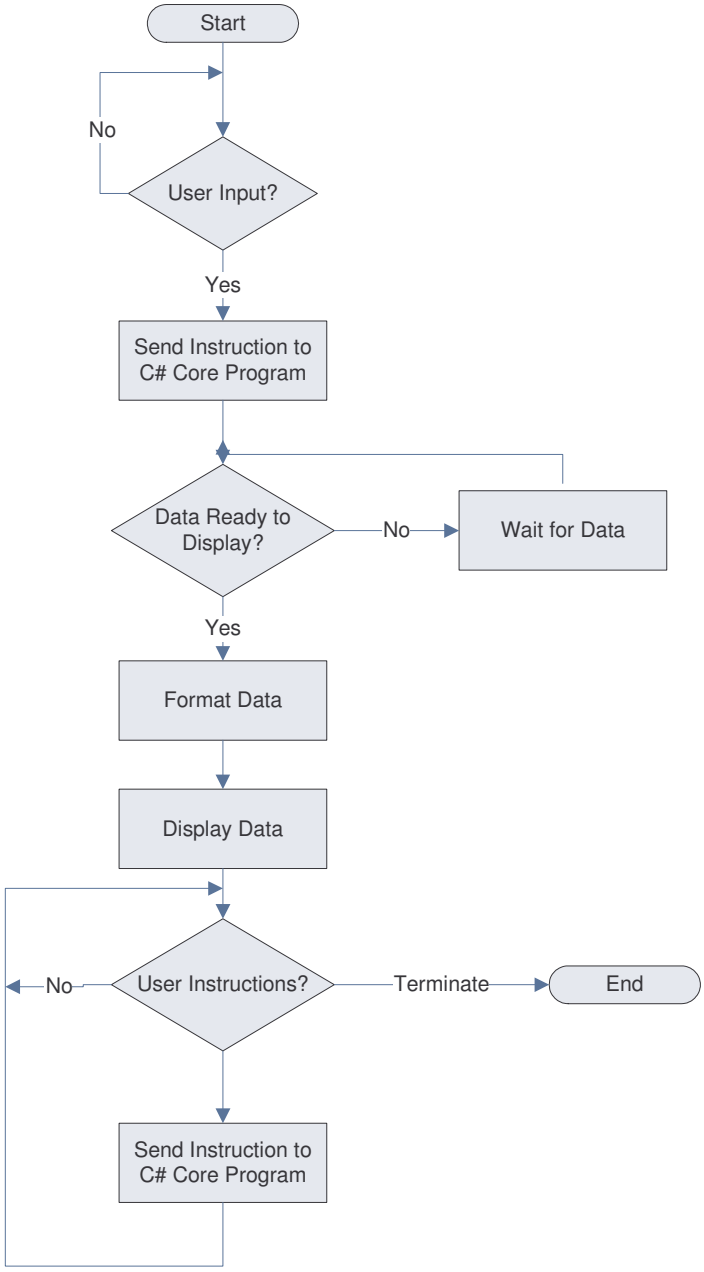


Figure 5.1 GUI Tasks

Integrating the goals into the schedule

The following are the primary goals of the project are detailed in the schedule:

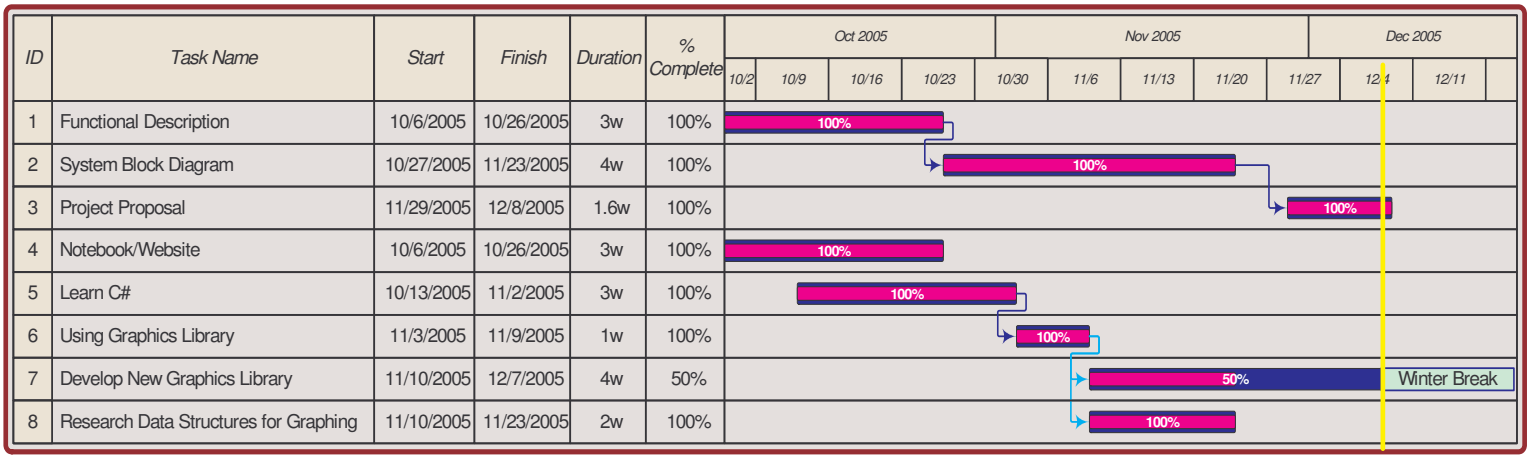
- New display engine for NPlot
- Combine lines into busses
- DLL interface for USB
- Data processing through memory instead of hard drive
- Print/Save data capture
- Continuous Update (Pseudo Real Time)

These goals and their dependencies are reflected in the timeline in the next section.

Schedule

The GANTT charts show the schedule for the project. The start and stop dates, the duration of each task, the percentage of completion of each task, and the interdependencies of the tasks can be seen in the GANTT charts.

The Fall semester schedule can be seen in Figure 6.1. As can be seen all the tasks scheduled for this semester have been completed, except for "Develop New Graphics Library" which will be completed over Winter Break as denoted by the schedule slip block.



ID	Task Name	Start	Finish	Duration	% Complete
1	Functional Description	10/6/2005	10/26/2005	3w	100%
2	System Block Diagram	10/27/2005	11/23/2005	4w	100%
3	Project Proposal	11/29/2005	12/8/2005	1.6w	100%
4	Notebook/Website	10/6/2005	10/26/2005	3w	100%
5	Learn C#	10/13/2005	11/2/2005	3w	100%
6	Using Graphics Library	11/3/2005	11/9/2005	1w	100%
7	Develop New Graphics Library	11/10/2005	12/7/2005	4w	50%
8	Research Data Structures for Graphing	11/10/2005	11/23/2005	2w	100%

Figure 6.1 Fall Semester Schedule

Not a lot of time was initially delegated to the deliverables task of the project, which caused some delay in other tasks. In addition to that there were some tasks were attempted in parallel, which also caused some schedule slip. These factors were considered in developing the Spring semester schedule which can be seen in figure 7.1.

ID	Task Name	Start	Finish	Duration	% Complete
1	DLL USB Interface to board	1/19/2006	2/8/2006	3w	0%
2	GUI	2/9/2006	3/15/2006	5w	0%
3	Data Saving/Retrieval	3/16/2006	3/29/2006	2w	0%
4	Testing with printed board	3/30/2006	4/12/2006	2w	0%
5	Testing with prototype board	4/13/2006	4/19/2006	1w	0%
6	Preparation for Expo	4/13/2006	4/26/2006	2w	0%
7	Final Paper	4/27/2006	5/10/2006	2w	0%

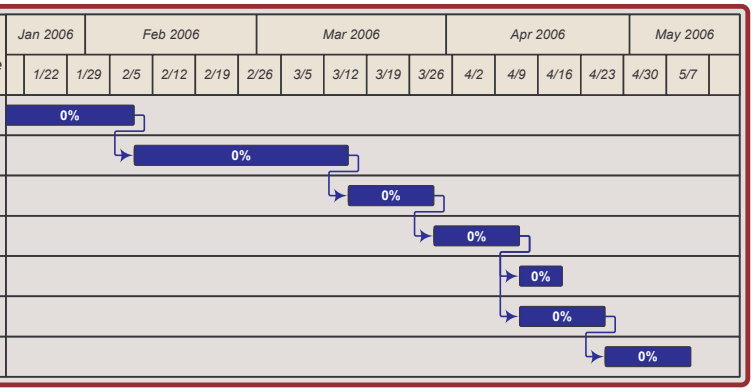


Figure 7.1 Spring semester schedule

Cost Justification

The Opal Kelly XEM 3001 FPGA board is a development platform with a Xilinx Spartan 3 FPGA onboard. The development platform was used to speed up the development time since it implements USB protocol natively, and the hardware part of the project can just be concerned with the external circuitry and VHDL development. The cost of the development platform was $200.00, whereas the Spartan 3 FPGA single unit costs $32.00. If the development board was not used there would be additional costs associated with the fabrication of the external circuitry along with the cost of the signal conditioning circuitry. The development board however can be used in Lab as a test board for other projects as well. If this project was ever to go to production, then the cost of the product is going to be significantly cheaper than equitable commercial solutions that sell for $400.00.

Progress

As the schedule demonstrates a lot of tasks were accomplished this semester. The deliverables have culminated in this project proposal and the research done will be briefly described. After researching and learning C#, which was a significant time investment, the NPlot graphics library was tackled. Test values were then plotted and some user interaction was added to the plot, specifically, the ability to select a region and zoom in on it. After understanding the fundamentals of the design a better GUI can be developed during the Spring semester. Figure 8.1 shows four lines of data being plotted together.

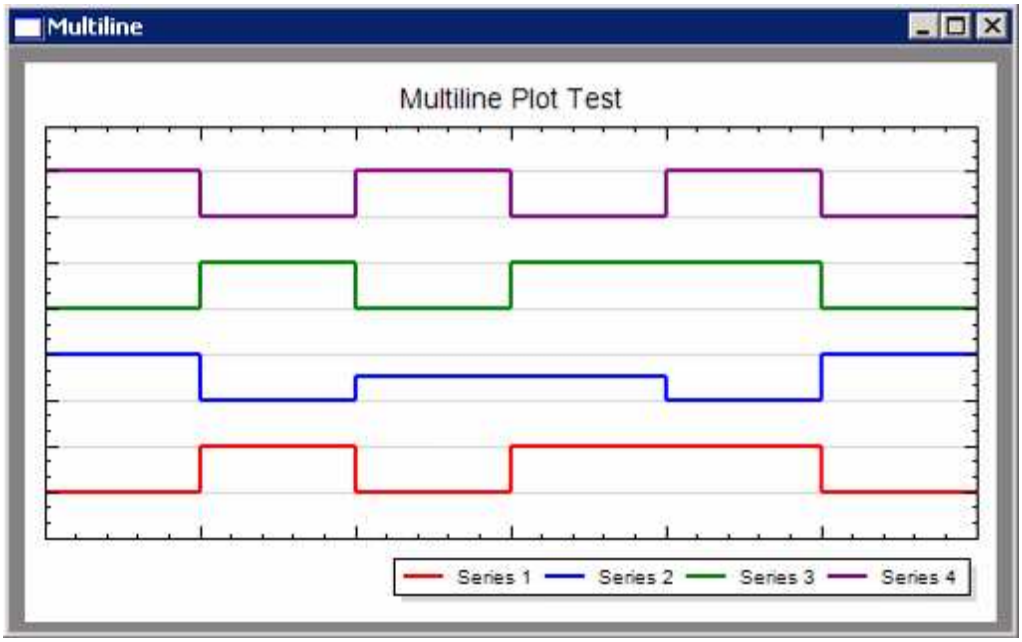


Figure 8.1 Multiple Lines being plotted on one surface

Eventually, the GUI will be modified to hold more information, to resemble Figure 8.2.

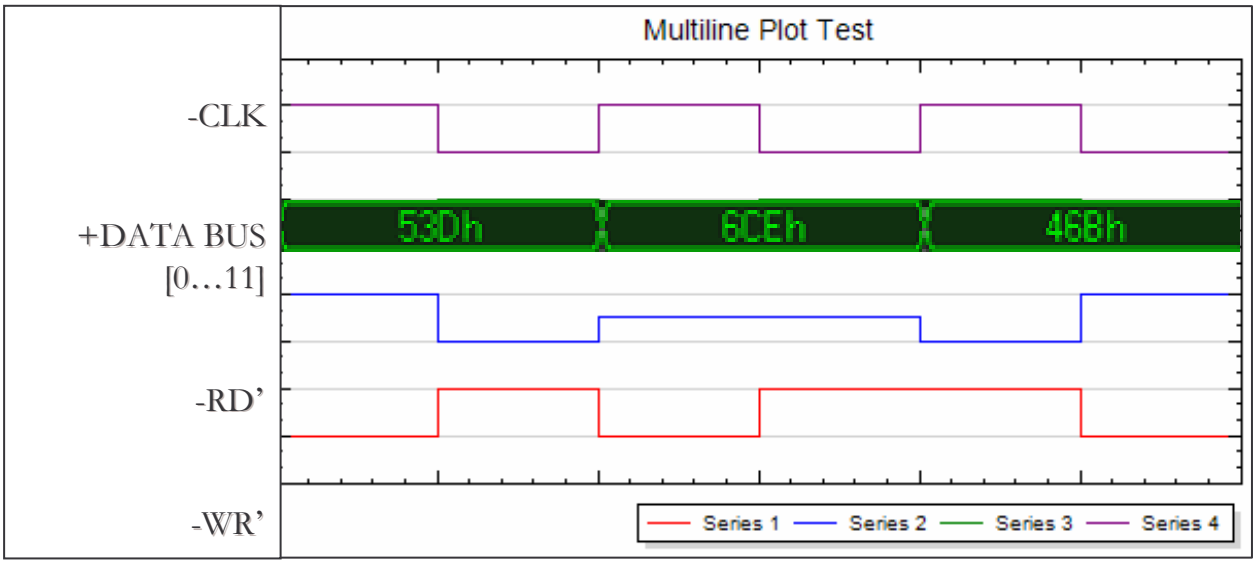


Figure 8.2 Possible future look of the GUI

In order to hold all the required information, two different data structures will be required. Table 9.1 lists the parameters of each of the data structures.

Line Class	Group Class
Data to be plotted	Internal Group Name
Internal Signal Name	User Defined Group Name
User Defined Signal Name	Number of total lines
Line Color	Lines part of this Group
Part of Group	Number Base (bin, dec, hex)*
Visible/Invisible	*defaults to hex

Table 9.1 Data structures and their contents

Finally, the project website tracks the project's progress. The website is located under Bradley University Electrical Engineering project pages at the following location: http://cegt201.bradley.edu/projects/proj2006/usbla/. Figure 9.1 shows a snapshot of the website for reference.

Figure 9.1 Project website snapshot