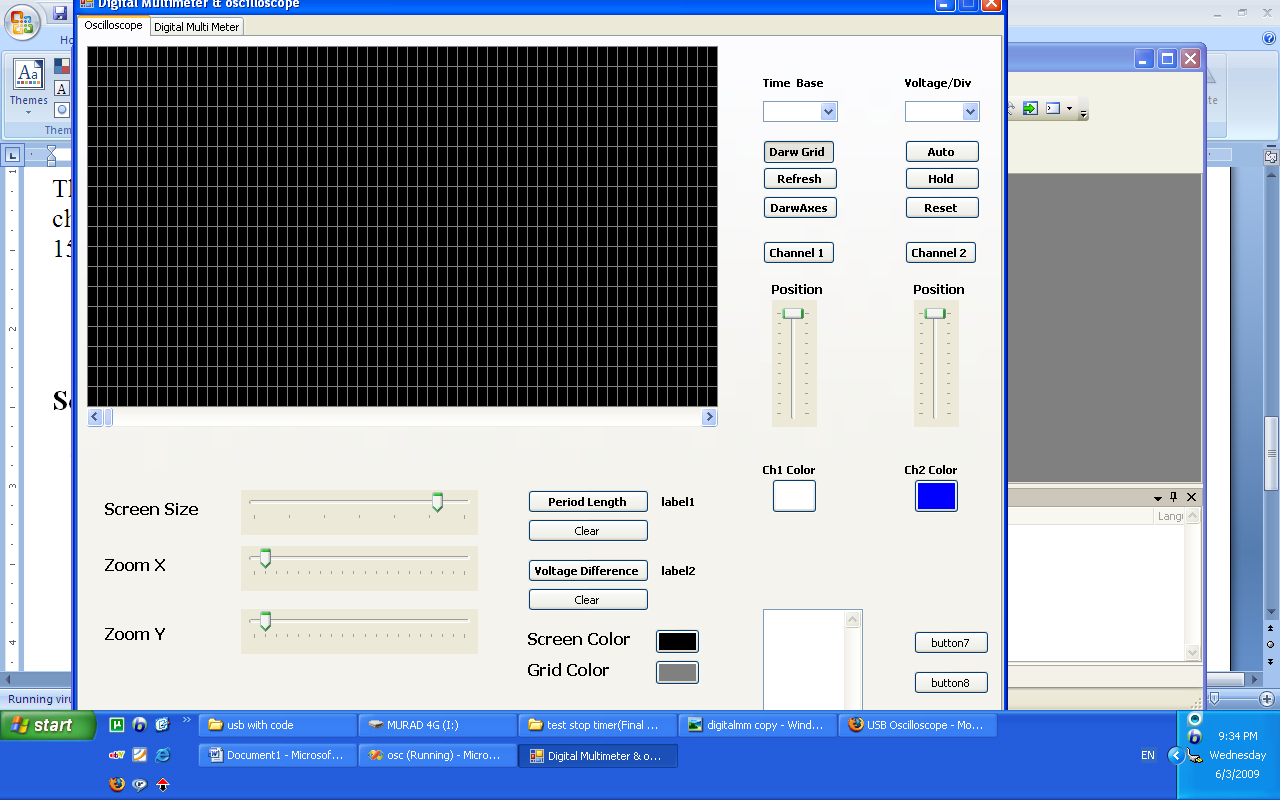
USB Oscilloscope &Digital Multimeter



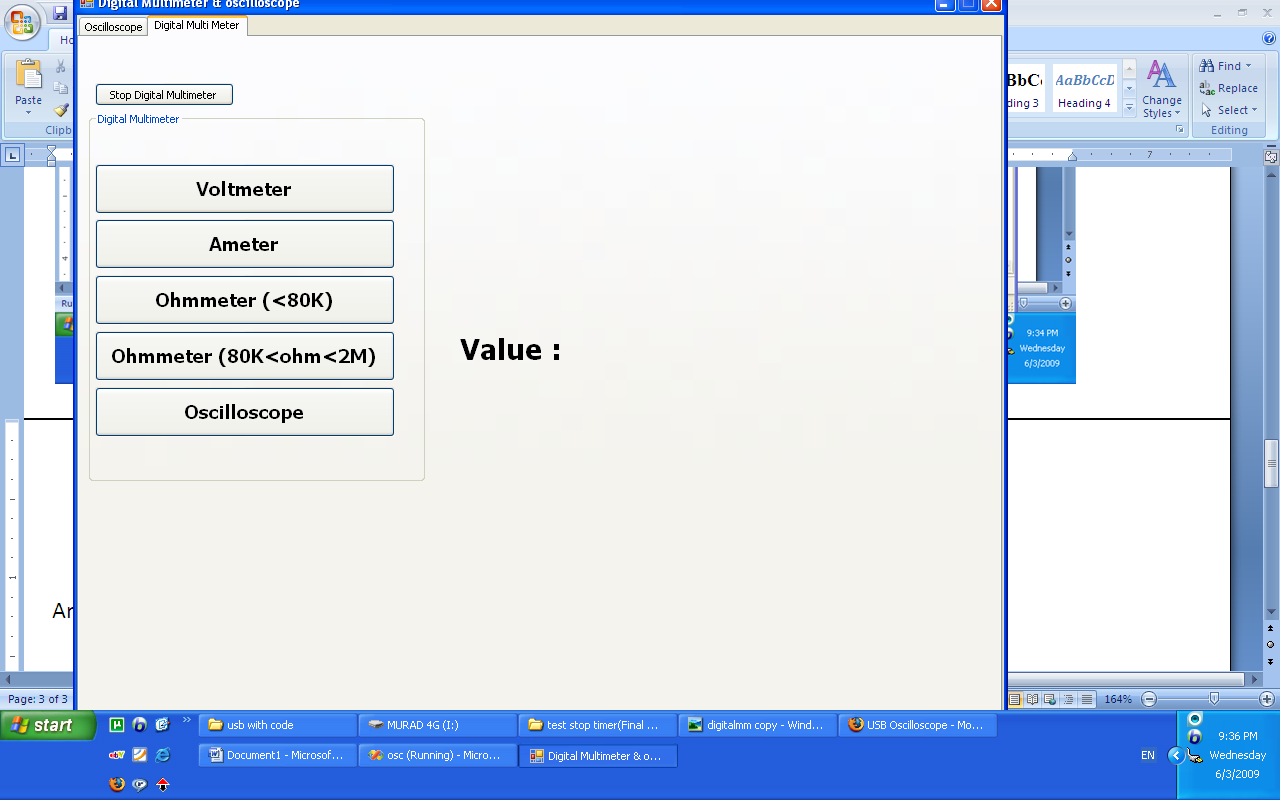
This one's an oscilloscope connected to a computer using the USB bus. This is probably the cheapest oscilloscope on the market, but maybe also the worst... The scope can only record 15479 samples per second.

**Screenshots:**

The signals in the screenshots have been generated using a Meter International Corp. FG-503 DDS function generator.



And this is the screen shots for multimeter



And this c# code for the project

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Collections;

using PicWinUSB;

namespace osc

{

public partial class Form1 : Form

{

IntPtr iHandle; // global device handle definition

Color BackColor;

PicWinUSBAPI picwinusbapi = new PicWinUSBAPI();

bool channel1Visible;

bool channel2Visible;

int picWidth = 500;

int picHeight = 250;

int Yshift;

int Xshift;

int Yshift2;

int Xshift2;

Pen grid;

Pen ch1;

Pen ch2;

int Zoom;

Timer t;

Timer t2;

int x=0;

int y;

int prev\_x;

int prev\_y;

int x2=0;

int y2;

int prev\_x2;

int prev\_y2;

StreamReader file;

StreamReader file2;

string line;

int counter = 0;

int counter2 = 0;

Graphics g;

string[] coordinates;

ArrayList dataStore;

ArrayList dataStore2;

int lastMinimumValue;

int lastMinimumValue2;

int MinimumValue = 0;

int MaximumValue = 500;

int zoomX, zoomY;

int Mouse1\_X;

int Mouse2\_X;

int Mouse1\_Y;

int Mouse2\_Y;

int MouseCount = 0;

bool GridVisiable;

int GridWidth;

int GridHeight;

int DatarecievedCount = 0;

int DatarecievedCount2 = 0;

Timer t3;

public Form1()

{

InitializeComponent();

channel1Visible = false;

channel2Visible = false;

panel1.Width = picWidth;

panel1.Height = picHeight;

hScrollBar1.Maximum = picWidth;

prev\_x = 20;

prev\_y = picHeight / 4;

prev\_x2 = 20;

prev\_y2 = 3\*picHeight / 4;

zoomX = 1;

zoomY = 1;

GridVisiable = false;

GridWidth = 10;

//comboBox1.SelectedIndex = 0;

GridHeight = 20;

Yshift = picHeight / 4;

Xshift = 0;

Yshift2 =3\* picHeight / 4;

Xshift2 = 0;

grid = new Pen(Brushes.Gray, 0.4f);

ch1 = new Pen(Brushes.White, 1f);

ch2 = new Pen(Brushes.Blue, 1f);

Zoom = 0;

dataStore = new ArrayList();

dataStore2 = new ArrayList();

t = new Timer();

t.Interval = 1;

//file = new StreamReader(@"c:\data.txt");

//file2 = new StreamReader(@"c:\data2.txt");

t2= new Timer();

t2.Interval = 10;

zoomX = trackBar2.Value;

zoomY = trackBar3.Value;

button3.BackColor = ch1.Color;

button4.BackColor = ch2.Color;

button5.BackColor = pictureBox1.BackColor;

button6.BackColor = grid.Color;

Guid InterfaceGuid = new Guid("31415926-5358-9793-2384-626433832795"); // .Inf defined Guid

iHandle = picwinusbapi.Init\_PicWinUSB(InterfaceGuid);

BackColor = pictureBox1.BackColor;

backPen=new Pen(BackColor,grid.Width);

t3 = new Timer();

t3.Interval = 1000;

}

public void translate\_coor(ref int y,ref int x)

{

y = -1 \* y \* zoomY + Yshift;

x = x\*zoomX + 20;

}

public void translate\_coor\_without\_x(ref int y, ref int x)

{

y = -1 \* y \* zoomY + Yshift;

x = x\*zoomX ;

}

public void translate\_coor2(ref int y2, ref int x2)

{

y2 = -1 \* y2 \* zoomY + Yshift2;

x2 = x2 \* zoomX + 20;

}

public void translate\_coor\_without\_x2(ref int y2, ref int x2)

{

y2 = -1 \* y2 \* zoomY + Yshift2;

x2 = x2 \* zoomX;

}

public void clear(Graphics g)

{

Pen p=new Pen(Brushes.Yellow,3f);

g.Clear(pictureBox1.BackColor);

if (GridVisiable)

{

DrawGrid(grid, GridWidth, GridHeight);

}

g.DrawLine(p, new Point(20, 0), new Point(20, picHeight));

g.DrawLine(p, new Point(0, picHeight / 2), new Point(picWidth, picHeight / 2));

p = null;

}

public void clearWtihoutX(Graphics g)

{

g.Clear(pictureBox1.BackColor);

if (GridVisiable)

{

DrawGrid(grid, GridWidth, GridHeight);

}

Pen p = new Pen(Brushes.Yellow, 2f);

//g.DrawLine(p, new Point(20, 0), new Point(20, 250));

g.DrawLine(p, new Point(0, picHeight / 2), new Point(picWidth, picHeight / 2));

p = null;

}

private void Form1\_Load(object sender, EventArgs e)

{

label12.Text = "";

}

private void button1\_Click(object sender, EventArgs e)

{

Graphics g=pictureBox1.CreateGraphics();

Pen p = new Pen(Brushes.Yellow, 3f);

g.DrawLine(p, new Point(20, 0), new Point(20, picHeight));

g.DrawLine(p, new Point(0, picHeight / 2), new Point(picWidth, picHeight / 2));

p = null;

}

int packetlength=1;

void t\_Tick(object sender, EventArgs e)

{

t.Stop();

// byte ss = 125;

sdBuffer[0] = 1;

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

if (bres)

{

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

}

//while (packetlength < 64)

//{

if ((rdBuffer[0] == 1))

{

y = Convert.ToInt32((rdBuffer[1]) \* 5 / 254)\*10;

x = DatarecievedCount;

DatarecievedCount++;

textBox1.AppendText("x= " + DatarecievedCount.ToString() + " y= " + rdBuffer[packetlength].ToString() + "\n");

}

DataClass Dataobject = new DataClass(x, y);

dataStore.Add(Dataobject);

if (x >= hScrollBar1.Minimum && x <= hScrollBar1.Maximum)

{

hScrollBar1.Value = x;

}

translate\_coor(ref y, ref x);

if (x == ((counter + 1) \* picWidth))

{

hScrollBar1.Maximum = picWidth \* (counter + 2);

prev\_x = 0;

prev\_y = picHeight / 4;

counter++;

clearWtihoutX(g);

}

x = x - (counter \* picWidth);

g.DrawLine(ch1, new Point(prev\_x, prev\_y), new Point(x, y));

prev\_x = x;

prev\_y = y;

//}

//packetlength = 1;

t.Start();

//}

//else

//{

// prev\_x = 20;

// prev\_y = picHeight/4;

// file.Close();

// t.Stop();

//}

}

public void Drawline(Pen p ,int x, int y, int prev\_x, int prev\_y, Graphics g,bool start)

{

if (x % picWidth == 0)

{

prev\_x = 0;

}

if (!start)

{

x = x % picWidth;

prev\_x = prev\_x % picWidth;

translate\_coor\_without\_x(ref prev\_y, ref prev\_x);

translate\_coor\_without\_x(ref y, ref x);

}

else

{

x = x % picWidth;

prev\_x = prev\_x % picWidth;

translate\_coor(ref prev\_y, ref prev\_x);

translate\_coor(ref y, ref x);

}

g.DrawLine(p, new Point(prev\_x, prev\_y), new Point(x, y));

}

public void Drawline2(Pen p, int x2, int y2, int prev\_x2, int prev\_y2, Graphics g, bool start2)

{

if (x2 % picWidth == 0)

{

prev\_x2 = 0;

}

if (!start2)

{

x2 = x2% picWidth;

prev\_x2 = prev\_x2 % picWidth;

translate\_coor\_without\_x2(ref prev\_y2, ref prev\_x2);

translate\_coor\_without\_x2(ref y2, ref x2);

}

else

{

x2 = x2 % picWidth;

prev\_x2 = prev\_x2 % picWidth;

translate\_coor2(ref prev\_y2, ref prev\_x2);

translate\_coor2(ref y2, ref x2);

}

g.DrawLine(p, new Point(prev\_x2, prev\_y2), new Point(x2, y2));

}

public void DrawFunction(int minimum,int maximum,Graphics g)

{

int x, y, prev\_x=minimum, prev\_y=0;

DataClass d;

if (minimum == 0)

{

clear(g);

}

else

{

clearWtihoutX(g);

}

bool start;

try

{

//Pen p = new Pen(Brushes.White, 1f);

for (int i = 0; i < dataStore.Count; i++)

{

d = (DataClass)dataStore[i];

if (d.x == (minimum / zoomX))

{

prev\_x = d.x;

prev\_y = d.y;

}

if ((d.x >= (minimum / zoomX)) && (d.x <= (maximum / zoomX)))

{

x = d.x;

y = d.y;

if (x \* zoomX > picWidth)

start = false;

else

start = true;

Drawline(ch1, x - (minimum / zoomX), y, prev\_x - (minimum / zoomX), prev\_y, g, start);

prev\_x = x;

prev\_y = y;

}

}

}

catch (Exception ex)

{

}

}

public void DrawFunction2(int minimum, int maximum, Graphics g)

{

int x2, y2, prev\_x2 = minimum, prev\_y2 = 0;

DataClass d;

//if (minimum == 0)

//{

// clear(g);

//}

//else

//{

// clearWtihoutX(g);

//}

bool start;

bool start2;

try

{

//Pen p = new Pen(Brushes.White, 1f);

for (int i = 0; i < dataStore2.Count; i++)

{

d = (DataClass)dataStore2[i];

if (d.x == (minimum / zoomX))

{

prev\_x2 = d.x;

prev\_y2 = d.y;

}

if ((d.x > (minimum / zoomX)) && (d.x <= (maximum / zoomX)))

{

x2 = d.x;

y2 = d.y;

if (x2 \* zoomX > picWidth)

start2 = false;

else

start2 = true;

Drawline2(ch2, x2 - (minimum / zoomX), y2, prev\_x2 - (minimum / zoomX), prev\_y2, g, start2);

prev\_x2 = x2;

prev\_y2 = y2;

}

}

}

catch (Exception ex)

{

}

}

private void hScrollBar1\_Scroll(object sender, ScrollEventArgs e)

{

t.Stop();

t2.Stop();

lastMinimumValue = MinimumValue;

MinimumValue = (hScrollBar1.Value / picWidth)\* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

Graphics g=pictureBox1.CreateGraphics();

if (lastMinimumValue!=MinimumValue)

{

DrawFunction(MinimumValue, MaximumValue,g );

DrawFunction2(MinimumValue, MaximumValue, g);

}

}

public void DrawGrid(Pen grid,int width, int height)

{

Graphics g=pictureBox1.CreateGraphics();

for (int i = 0; i < picWidth; i = i + width)

{

g.DrawLine(grid, new Point(i, 0), new Point(i, picHeight));

}

for (int i = 0; i < picHeight; i = i + height)

{

g.DrawLine(grid, new Point(0, i), new Point(picWidth,i));

}

}

private void button8\_Click(object sender, EventArgs e)

{

}

private void button9\_Click(object sender, EventArgs e)

{

//Pen grid = new Pen(Brushes.Black, 0.4f);

DrawGrid(grid, GridWidth, GridHeight);

GridHeight = 2 \* GridHeight;

GridWidth = 2 \* GridWidth;

//grid = new Pen(Brushes.Yellow, 0.4f);

DrawGrid(grid, GridWidth, GridHeight);

}

private void button10\_Click(object sender, EventArgs e)

{

//Pen grid = new Pen(Brushes.Black, 0.4f);

DrawGrid(grid, GridWidth, GridHeight);

if (GridWidth > 10)

{

GridWidth = GridWidth/2;

}

if (GridHeight > 10)

{

GridHeight = GridHeight/2;

}

grid = new Pen(Brushes.Yellow, 0.4f);

DrawGrid(grid, GridWidth, GridHeight);

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (!checkBox1.Checked)

{

Graphics g = pictureBox1.CreateGraphics();

Pen p = new Pen(pictureBox1.BackColor, 1f);

g.DrawLine(p, new Point(Mouse1\_X, Mouse1\_Y - (picHeight / 4)), new Point(Mouse1\_X, Mouse1\_Y + (picHeight / 4)));

g.DrawLine(p, new Point(Mouse2\_X, Mouse2\_Y - (picHeight / 4)), new Point(Mouse2\_X, Mouse2\_Y + (picHeight / 4)));

MouseCount = 0;

}

}

private void pictureBox1\_MouseHover(object sender, EventArgs e)

{

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

}

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e)

{

Graphics g = pictureBox1.CreateGraphics();

if (checkBox1.Checked)

{

if (MouseCount < 2)

{

MouseCount++;

if (MouseCount == 1)

{

Mouse1\_X = e.X;

Mouse1\_Y = e.Y;

Pen p = new Pen(Brushes.Blue, 1f);

g.DrawLine(p, new Point(e.X, Mouse1\_Y - (picHeight / 4)), new Point(e.X, Mouse1\_Y + (picHeight / 4)));

}

if (MouseCount == 2)

{

Mouse2\_X = e.X;

Mouse2\_Y = e.Y;

label1.Text = Convert.ToString(Math.Abs(Mouse1\_X - Mouse2\_X)/zoomX)+" ms";

Pen p = new Pen(Brushes.Blue, 1f);

g.DrawLine(p, new Point(e.X, Mouse2\_Y - (picHeight / 4)), new Point(e.X, Mouse2\_Y + (picHeight / 4)));

}

}

}

if (checkBox2.Checked)

{

if (MouseCount < 2)

{

MouseCount++;

if (MouseCount == 1)

{

Mouse1\_X = e.X;

Mouse1\_Y = e.Y;

Pen p = new Pen(Brushes.Blue, 1f);

g.DrawLine(p, new Point(Mouse1\_X - picWidth / 4, e.Y), new Point(Mouse1\_X + picWidth / 4, e.Y));

}

if (MouseCount == 2)

{

Mouse2\_X = e.X;

Mouse2\_Y = e.Y;

Pen p = new Pen(Brushes.Blue, 1f);

g.DrawLine(p, new Point(Mouse2\_X - picWidth / 4, e.Y), new Point(Mouse2\_X + picWidth / 4, e.Y));

label2.Text = Convert.ToString(Math.Abs(Mouse1\_Y - Mouse2\_Y)/(10\*zoomY)) + " V ";

}

}

}

}

private void button11\_Click(object sender, EventArgs e)

{

Graphics g = pictureBox1.CreateGraphics();

Pen p = new Pen(pictureBox1.BackColor, 1f);

g.DrawLine(p, new Point(Mouse1\_X, Mouse1\_Y - picHeight / 4), new Point(Mouse1\_X, Mouse1\_Y + picHeight / 4));

g.DrawLine(p, new Point(Mouse2\_X, Mouse2\_Y - picHeight / 4), new Point(Mouse2\_X, Mouse2\_Y + picHeight / 4));

label1.Text = "";

MouseCount=0;

}

private void button12\_Click(object sender, EventArgs e)

{

Graphics g = pictureBox1.CreateGraphics();

Pen p = new Pen(pictureBox1.BackColor, 1f);

g.DrawLine(p, new Point(Mouse1\_X - picWidth / 4, Mouse1\_Y), new Point(Mouse1\_X + picWidth / 4, Mouse1\_Y));

g.DrawLine(p, new Point(Mouse2\_X - picWidth / 4, Mouse2\_Y), new Point(Mouse2\_X + picWidth / 4, Mouse2\_Y));

label2.Text = "";

MouseCount = 0;

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

if (!checkBox2.Checked)

{

Graphics g = pictureBox1.CreateGraphics();

Pen p = new Pen(pictureBox1.BackColor, 1f);

g.DrawLine(p, new Point(Mouse1\_X - picWidth / 4, Mouse1\_Y), new Point(Mouse1\_X + picWidth / 4,Mouse1\_Y));

g.DrawLine(p, new Point(Mouse2\_X - picWidth / 4, Mouse2\_Y), new Point(Mouse2\_X + picWidth / 4,Mouse2\_Y));

label2.Text = "";

MouseCount = 0;

}

}

private void button14\_Click(object sender, EventArgs e)

{

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

Graphics g=pictureBox1.CreateGraphics();

DrawFunction(MinimumValue, MaximumValue, g);

DrawFunction2(MinimumValue, MaximumValue,g);

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

GridVisiable = true;

if (comboBox1.SelectedIndex == 0)

{

// zoomX = 100;

t.Interval = 1;

t2.Interval = 1;

GridWidth = 10;

t.Stop();

lastMinimumValue = MinimumValue;

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

hScrollBar1.Maximum = dataStore.Count \* zoomX;

DrawFunction(MinimumValue, MaximumValue, pictureBox1.CreateGraphics());

}

else if (comboBox1.SelectedIndex == 1)

{

//zoomX = 10;

t.Interval = 1;

t2.Interval = 1;

GridWidth = 100;

//TimeLabel.Text=Convert.ToString(

t.Stop();

lastMinimumValue = MinimumValue;

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

hScrollBar1.Maximum = dataStore.Count \* zoomX;

DrawFunction(MinimumValue, MaximumValue, pictureBox1.CreateGraphics());

}

else if (comboBox1.SelectedIndex == 2)

{

// zoomX = 1;

t.Interval = 10;

t2.Interval = 10;

GridWidth = 1000;

t.Stop();

lastMinimumValue = MinimumValue;

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

hScrollBar1.Maximum = dataStore.Count \* zoomX;

DrawFunction(MinimumValue, MaximumValue, pictureBox1.CreateGraphics());

}

}

private void button15\_Click(object sender, EventArgs e)

{

//DatarecievedCount = 0;

//DatarecievedCount2 = 0;

g = pictureBox1.CreateGraphics();

clear(g);

if (channel1Visible)

{

t.Tick += new EventHandler(t\_Tick);

t.Start();

}

if (channel2Visible)

{

t2.Tick += new EventHandler(t2\_Tick);

t2.Start();

}

}

bool bres;

byte[] sdBuffer=new byte[1]; // Define send DataBuffer size

byte[] rdBuffer = new byte[2];

void t2\_Tick(object sender, EventArgs e)

{

//throw new Exception("The method or operation is not implemented.");

//if ((line = file2.ReadLine()) != null)

//{

// coordinates = line.Split('\*');

// coordinates[0] = coordinates[0].Trim();

// coordinates[1] = coordinates[1].Trim();

// x2 = Convert.ToInt32(coordinates[0]);

// y2 = Convert.ToInt32(coordinates[1]);

t2.Stop();

bool bres;

// Define recieve DataBuffer size

sdBuffer[0] = 0x02; //Mode

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

if (rdBuffer[0] == 2)

{

y2=Convert.ToInt32(rdBuffer[1]\*5/254)\*10;

x2 = DatarecievedCount2;

DatarecievedCount2++;

}

DataClass Dataobject = new DataClass(x2, y2);

dataStore2.Add(Dataobject);

if (x2 >= hScrollBar1.Minimum && x2 <= hScrollBar1.Maximum)

{

hScrollBar1.Value = x2;

}

translate\_coor2(ref y2, ref x2);

if (x2 == ((counter2 + 1) \* picWidth))

{

hScrollBar1.Maximum = picWidth \* (counter2 + 2);

prev\_x2 = 0;

prev\_y2 = Yshift2;

counter2++;

clearWtihoutX(g);

}

x2 = x2 - (counter2 \* picWidth);

g.DrawLine(ch2, new Point(prev\_x2, prev\_y2), new Point(x2, y2));

prev\_x2 = x2;

prev\_y2 = y2;

t2.Start();

}

private void button17\_Click(object sender, EventArgs e)

{

t.Stop();

t2.Stop();

}

private void trackBar1\_Scroll\_1(object sender, EventArgs e)

{

if (channel1Visible)

t.Stop();

if (channel2Visible)

t2.Stop();

if (trackBar1.Value >= Zoom)

{

panel1.Width = panel1.Width + trackBar1.Value-Zoom;

panel1.Height = panel1.Height + trackBar1.Value -Zoom;

Zoom = trackBar1.Value;

}

else

{

panel1.Width = panel1.Width + trackBar1.Value-Zoom;

panel1.Height = panel1.Height + trackBar1.Value - Zoom;

Zoom = trackBar1.Value;

}

pictureBox1.Width = panel1.Width;

pictureBox1.Height = panel1.Height - 20;

pictureBox1.Refresh();

picWidth = panel1.Width;

picHeight = panel1.Height;

hScrollBar1.Maximum = picWidth;

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

g = pictureBox1.CreateGraphics();

if (channel1Visible)

DrawFunction(MinimumValue, MaximumValue, g);

if (channel2Visible)

DrawFunction2(MinimumValue, MaximumValue, g);

if (GridVisiable)

DrawGrid(grid, picWidth, picHeight);

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

zoomX = trackBar2.Value ;

GridWidth = trackBar2.Value \* GridWidth;

t.Stop();

t2.Stop();

lastMinimumValue = MinimumValue;

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

if (dataStore2.Count >= dataStore.Count)

{

hScrollBar1.Maximum = dataStore2.Count \* zoomX;

}

else

{

hScrollBar1.Maximum = dataStore.Count \* zoomX;

}

g = pictureBox1.CreateGraphics();

if (channel1Visible)

DrawFunction(MinimumValue, MaximumValue, g);

if (channel2Visible)

DrawFunction2(MinimumValue, MaximumValue, g);

}

private void trackBar3\_Scroll(object sender, EventArgs e)

{

zoomY = trackBar3.Value ;

if (comboBox2.SelectedIndex == 0)

{

GridHeight = trackBar3.Value \* 50;

}

else if (comboBox2.SelectedIndex == 1)

{

GridHeight = trackBar3.Value \* 10;

}

t.Stop();

t2.Stop();

lastMinimumValue = MinimumValue;

MinimumValue = (hScrollBar1.Value / picWidth) \* picWidth;

MaximumValue = ((hScrollBar1.Value / picWidth) + 1) \* picWidth;

g = pictureBox1.CreateGraphics();

if(channel1Visible)

DrawFunction(MinimumValue, MaximumValue, g);

if (channel2Visible)

DrawFunction2(MinimumValue, MaximumValue,g);

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void osc\_Click(object sender, EventArgs e)

{

}

private void channel1\_CheckedChanged(object sender, EventArgs e)

{

channel1Visible = channel1.Checked;

}

private void channel2\_CheckedChanged(object sender, EventArgs e)

{

channel2Visible = channel2.Checked;

}

private void trackBar4\_Scroll(object sender, EventArgs e)

{

Yshift = (10-trackBar4.Value) \* picHeight / 10;

}

private void trackBar5\_Scroll(object sender, EventArgs e)

{

Yshift2 = (10 - trackBar5.Value) \* picHeight / 10;

}

private void button2\_Click(object sender, EventArgs e)

{

t.Stop();

t2.Stop();

dataStore.Clear();

dataStore2.Clear();

clear(pictureBox1.CreateGraphics());

panel1.Width = picWidth;

panel1.Height = picHeight;

hScrollBar1.Maximum = picWidth;

prev\_x = 20;

prev\_y = picHeight / 4;

prev\_x2 = 20;

prev\_y2 = 3 \* picHeight / 4;

zoomX = 1;

zoomY = 1;

GridWidth = 10;

GridHeight = 20;

Yshift = picHeight / 4;

Xshift = 0;

Yshift2 = 3 \* picHeight / 4;

Xshift2 = 0;

grid = new Pen(Brushes.Yellow, 0.4f);

ch1 = new Pen(Brushes.White, 1f);

ch2 = new Pen(Brushes.Blue, 1f);

Zoom = 0;

dataStore = new ArrayList();

dataStore2 = new ArrayList();

DatarecievedCount = 0;

DatarecievedCount2 = 0;

t = new Timer();

t.Interval = 1;

//file = new StreamReader(@"c:\data.txt");

//file2 = new StreamReader(@"c:\data2.txt");

t2 = new Timer();

t2.Interval = 10;

}

private void button3\_Click(object sender, EventArgs e)

{

DialogResult res = colorDialog1.ShowDialog();

if (res == DialogResult.OK)

{

button3.BackColor = colorDialog1.Color;

ch1.Color = button3.BackColor;

button14.Click+=new EventHandler(button14\_Click);

}

}

private void button4\_Click(object sender, EventArgs e)

{

DialogResult res = colorDialog1.ShowDialog();

if (res == DialogResult.OK)

{

button4.BackColor = colorDialog1.Color;

ch2.Color = button4.BackColor;

}

}

private void button5\_Click(object sender, EventArgs e)

{

DialogResult res = colorDialog1.ShowDialog();

if (res == DialogResult.OK)

{

button5.BackColor = colorDialog1.Color;

pictureBox1.BackColor= button5.BackColor;

BackColor = pictureBox1.BackColor;

}

}

private void label11\_Click(object sender, EventArgs e)

{

}

private void button6\_Click(object sender, EventArgs e)

{

DialogResult res = colorDialog1.ShowDialog();

if (res == DialogResult.OK)

{

button6.BackColor = colorDialog1.Color;

grid.Color = button6.BackColor;

button14\_Click(sender, e);

}

}

//private void radioButton5\_CheckedChanged(object sender, EventArgs e)

//{

// if (!fun1.Checked)

// {

// bool bres;

// byte[] sdBuffer = new byte[2]; // Define send DataBuffer size

// sdBuffer[0] = 0x06; //Mode

// sdBuffer[1] = 0x01;

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

// }

//}

private void dmm\_Click(object sender, EventArgs e)

{

}

private void radioButton8\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton8.Checked)

{

t3.Stop();

tabControl1.SelectedTab = osc;

}

}

private void panel3\_Paint(object sender, PaintEventArgs e)

{

}

private void button7\_Click(object sender, EventArgs e)

{

bool bres;

sdBuffer[0] = 0x01; //Mode

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

if (rdBuffer[0] == 1)

{

y = Convert.ToInt32(rdBuffer[1] \* 5 / 254) \* 5;

x++;

}

textBox1.AppendText("x= " + x.ToString() + " y= " + y.ToString() + "\n");

}

private void volt\_CheckedChanged(object sender, EventArgs e)

{

t3.Tick += new EventHandler(t3\_Tick);

t3.Start();

// double sum = 0;

// int count = 0;

//volt:

// if (volt.Checked)

// {

// //MessageBox.Show("mmm");

// bool bres;

// byte[] sdBuffer = new byte[1]; // Define send DataBuffer size

// byte[] rdBuffer = new byte[2]; // Define recieve DataBuffer size

// sdBuffer[0] = 0x03; //Mode

// //while ((count<10))

// //{

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

// bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

// sum += Convert.ToDouble(rdBuffer[1]);

// count++;

// //}

// double volt1 = (sum / count) \* 4.9 \* 5 / 254;

// //MessageBox.Show(Convert.ToString(sum / count) + " V");

// label12.Text = Convert.ToString(volt1) + " V";

// count = 0;

// sum = 0;

// //goto volt;

// }

}

void t3\_Tick(object sender, EventArgs e)

{

//throw new Exception("The method or operation is not implemented.");

double sum = 0;

int count = 0;

byte[] sdBuffer; // Define send DataBuffer size

byte[] rdBuffer ; // Define recieve DataBuffer size

if (volt.Checked)

{

//MessageBox.Show("mmm");

bool bres;

sdBuffer = new byte[1]; // Define send DataBuffer size

rdBuffer = new byte[2]; // Define recieve DataBuffer size

sdBuffer[0] = 0x03; //Mode

//while ((count<10))

//{

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

sum += Convert.ToDouble(rdBuffer[1]);

count++;

//}

double volt1 = (sum / count) \* 4.9 \* 5 / 254;

//MessageBox.Show(Convert.ToString(sum / count) + " V");

volt1 = Math.Round(volt1, 2);

label12.Text = Convert.ToString(volt1) + " V";

count = 0;

sum = 0;

}

else if (Ameter.Checked)

{

bool bres;

sdBuffer = new byte[1]; // Define send DataBuffer size

rdBuffer = new byte[2]; // Define recieve DataBuffer size

sdBuffer[0] = 0x04; //Mode

// while (count<10)

// {

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

sum += Convert.ToDouble(rdBuffer[1]);

count++;

//}

double ampere = (sum / count) \* 5 / (254 \* 5);

//decimal dd = new decimal(ampere);

ampere= Math.Round(ampere, 4);

label12.Text = Convert.ToString(ampere) + " A";

count = 0;

sum = 0;

// goto ameterr;

}

else if (ohm1.Checked)

{

bool bres;

sdBuffer = new byte[1]; // Define send DataBuffer size

rdBuffer = new byte[2]; // Define recieve DataBuffer size

sdBuffer[0] = 0x05; //Mode

//while (count<10)

//{

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

sum += Convert.ToDouble(rdBuffer[1]);

count++;

//}

double volt1 =Convert.ToDouble( (sum / count) \* 5 / 255);

double current = volt1 / 324;

double resist = (5 - volt1) / current;

resist = Math.Round(resist);

//decimal dd = new decimal(resist);

label12.Text = resist + " Ohm";

count = 0;

sum = 0;

//goto ohm11;

}

else if (ohm2.Checked)

{

bool bres;

sdBuffer = new byte[1]; // Define send DataBuffer size

rdBuffer = new byte[2]; // Define recieve DataBuffer size

sdBuffer[0] = 0x06; //Mode

//while (count<10)

//{

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

sum += Convert.ToDouble(rdBuffer[1]);

count++;

//}

double volt1 = (sum / count) \* 5 / 254;

double current = volt1 / 370000;

double resist = (5 - volt1) / current;

//decimal dd = new decimal(resist);

resist = Math.Round(resist / 1000);

label12.Text = Convert.ToString(resist) + " K Ohm";

count = 0;

sum = 0;

// goto ohm22;

}

}

private void Ameter\_CheckedChanged(object sender, EventArgs e)

{

t3.Tick += new EventHandler(t3\_Tick);

t3.Start();

// double sum = 0;

// int count = 0;

//ameterr:

// if (Ameter.Checked)

// {

// bool bres;

// byte[] sdBuffer = new byte[1]; // Define send DataBuffer size

// byte[] rdBuffer = new byte[2]; // Define recieve DataBuffer size

// sdBuffer[0] = 0x04; //Mode

// // while (count<10)

// // {

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

// bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

// sum += Convert.ToDouble(rdBuffer[1]);

// count++;

// //}

// double ampere = (sum / count) \* 5\*2 / (254 \* 5);

// label12.Text = Convert.ToString(ampere) + " A";

// count = 0;

// sum = 0;

// // goto ameterr;

// }

}

private void ohm1\_CheckedChanged(object sender, EventArgs e)

{

t3.Tick += new EventHandler(t3\_Tick);

t3.Start();

// double sum = 0;

// int count = 0;

//ohm11:

// if (ohm1.Checked)

// {

// bool bres;

// byte[] sdBuffer = new byte[1]; // Define send DataBuffer size

// byte[] rdBuffer = new byte[2]; // Define recieve DataBuffer size

// sdBuffer[0] = 0x05; //Mode

// //while (count<10)

// //{

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

// bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

// sum += Convert.ToDouble(rdBuffer[1]);

// count++;

// //}

// double volt = (sum/count) \* 5 / 255;

// double current = volt / 324;

// double resist = (5 - volt) / current;

// decimal dd = new decimal(resist);

// label12.Text = Convert.ToString(dd.ToString("0.##")) + " Ohm";

// count = 0;

// sum = 0;

// //goto ohm11;

// }

}

private void ohm2\_CheckedChanged(object sender, EventArgs e)

{

t3.Tick += new EventHandler(t3\_Tick);

t3.Start();

// double sum = 0;

// int count = 0;

//ohm22:

// if (ohm2.Checked)

// {

// bool bres;

// byte[] sdBuffer = new byte[1]; // Define send DataBuffer size

// byte[] rdBuffer = new byte[2]; // Define recieve DataBuffer size

// sdBuffer[0] = 0x06; //Mode

// //while (count<10)

// //{

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

// bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

// sum += Convert.ToDouble(rdBuffer[1]);

// count++;

// //}

// double volt = (sum / count) \* 5 / 254;

// double current = volt / 100000;

// double resist = (5 - volt) / current;

// label12.Text = Convert.ToString(resist) + " Ohm";

// count = 0;

// sum = 0;

// // goto ohm22;

// }

}

private void fun2\_CheckedChanged(object sender, EventArgs e)

{

//if (fun2.Checked)

//{

// MessageBox.Show("gfhfhf");

// bool bres;

// byte[] sdBuffer = new byte[2]; // Define send DataBuffer size

// sdBuffer[0] = 0x07; //Mode

// sdBuffer[1] = 0x01;

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

//}

}

Pen backPen;

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

backPen=new Pen(BackColor,grid.Width);

if (comboBox2.SelectedIndex == 0)

{

DrawGrid(backPen, GridWidth, GridHeight);

GridHeight = 50;

DrawGrid(grid, GridWidth, GridHeight);

}

else if (comboBox2.SelectedIndex == 1)

{

DrawGrid(backPen, GridWidth, GridHeight);

GridHeight = 10;

DrawGrid(grid, GridWidth, GridHeight);

}

if (!GridVisiable)

{

//Pen grid = new Pen(Brushes.Yellow, 0.4f);

DrawGrid(backPen, GridWidth, GridHeight);

}

else

{

//Pen grid = new Pen(Brushes.Black, 0.4f);

DrawGrid(grid, GridWidth, GridHeight);

}

//GridVisiable = !GridVisiable;

}

private void checkBox3\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox3.Checked)

GridVisiable = true;

else GridVisiable = false;

if (!GridVisiable)

{

//Pen grid = new Pen(Brushes.Yellow, 0.4f);

DrawGrid(backPen, GridWidth, GridHeight);

}

else

{

//Pen grid = new Pen(Brushes.Black, 0.4f);

DrawGrid(grid, GridWidth, GridHeight);

}

}

Timer t4;

private void button7\_Click\_1(object sender, EventArgs e)

{

bool bres;

sdBuffer[0] = 0x01; //Mode

// bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

t4 = new Timer();

t4.Interval = 1;

t4.Tick += new EventHandler(t4\_Tick);

t4.Start();

//while(packetlength<=10)

//{

}

void t4\_Tick(object sender, EventArgs e)

{

//throw new Exception("The method or operation is not implemented.");

bres = picwinusbapi.Write\_PicWinUSB(iHandle, sdBuffer);

if (bres)

{

bres = picwinusbapi.Read\_PicWinUSB(iHandle, rdBuffer);

}

//while (packetlength < 64)

//{

if ((rdBuffer[0] == 1)||(rdBuffer[0] == 1))

{

if (rdBuffer[0] == 1)

{

y = Convert.ToInt32((rdBuffer[1]+127) \* 5 / 254);

x = DatarecievedCount;

DatarecievedCount++;

}

else

{

y = Convert.ToInt32(rdBuffer[1] \* 5 / 254);

x = DatarecievedCount;

DatarecievedCount++;

}

textBox1.AppendText("x= " + DatarecievedCount.ToString() + " y= " + rdBuffer[packetlength].ToString() + "\n");

}

// packetlength++;

//}

//packetlength = 1;

}

private void button8\_Click\_1(object sender, EventArgs e)

{

t4.Stop();

}

private void button9\_Click\_1(object sender, EventArgs e)

{

t3.Stop();

}

private void label15\_Click(object sender, EventArgs e)

{

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace osc

{

class DataClass

{

public int x;

public int y;

public DataClass()

{

x = 0;

y = 0;

}

public DataClass(int x1,int y1)

{

x = x1;

y = y1;

}

}

}

using System;

using System.Runtime.InteropServices;

using System.IO;

using Microsoft.Win32.SafeHandles;

namespace PicWinUSB

{

class DeviceInterfaceImports

{

public const int DIGCF\_PRESENT = 0x00000002;

public const int DIGCF\_DEVICEINTERFACE = 0x00000010;

public const int FILE\_ATTRIBUTE\_NORMAL = 0x00000080;

public const int FILE\_FLAG\_OVERLAPPED = 0x40000000;

[StructLayout(LayoutKind.Sequential)]

public struct SP\_DEVICE\_INTERFACE\_DATA

{

public int cbSize;

public Guid InterfaceClassGuid;

public int Flags;

public IntPtr Reserved;

}

[StructLayout(LayoutKind.Sequential)]

public struct SP\_DEVICE\_INTERFACE\_DETAIL\_DATA

{

public int cbSize;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]

public string DevicePath;

}

[DllImport(@"setupapi.dll", CharSet = CharSet.Auto, SetLastError = true)]

public static extern IntPtr SetupDiGetClassDevs(

ref Guid ClassGuid,

[MarshalAs(UnmanagedType.LPTStr)] string Enumerator,

IntPtr hwndParent,

UInt32 Flags);

[DllImport(@"setupapi.dll", CharSet = CharSet.Auto, SetLastError = true)]

public static extern Boolean SetupDiEnumDeviceInterfaces(

IntPtr hDevInfo,

IntPtr devInfo,

ref Guid interfaceClassGuid,

UInt32 memberIndex,

ref SP\_DEVICE\_INTERFACE\_DATA deviceInterfaceData);

[DllImport(@"setupapi.dll", SetLastError = true)]

public static extern Boolean SetupDiGetDeviceInterfaceDetail(

IntPtr hDevInfo,

ref SP\_DEVICE\_INTERFACE\_DATA deviceInterfaceData,

IntPtr deviceInterfaceDetailData,

UInt32 deviceInterfaceDetailDataSize,

out UInt32 requiredSize,

IntPtr deviceInfoData);

[DllImport(@"setupapi.dll", SetLastError = true)]

public static extern Boolean SetupDiGetDeviceInterfaceDetail(

IntPtr hDevInfo,

ref SP\_DEVICE\_INTERFACE\_DATA deviceInterfaceData,

ref SP\_DEVICE\_INTERFACE\_DETAIL\_DATA deviceInterfaceDetailData,

UInt32 deviceInterfaceDetailDataSize,

out UInt32 requiredSize,

IntPtr deviceInfoData);

[DllImport(@"setupapi.dll", CharSet = CharSet.Auto, SetLastError = true)]

public static extern UInt16 SetupDiDestroyDeviceInfoList(

IntPtr hDevInfo);

[DllImport("kernel32.dll")]

public static extern IntPtr CreateFile(

string fileName,

FileAccess fileAccess,

FileShare fileShare,

IntPtr securityAttributes,

FileMode creationDisposition,

UInt32 flags,

IntPtr template);

[DllImport("kernel32.dll")]

public static extern bool CloseHandle(

IntPtr hObject);

}

}

using System;

using System.Diagnostics;

using System.IO;

using System.Windows.Forms;

using Microsoft.Win32.SafeHandles;

using System.Runtime.InteropServices;

namespace PicWinUSB

{

unsafe public class PicWinUSBAPI

{

#region Imported functions from winusb.dll ... some functions has not been checked yet.

[DllImport("winusb.dll")]

private static extern bool WinUsb\_AbortPipe(IntPtr InterfaceHandle, byte PipeID);

//[DllImport("winusb.dll")]

//private static extern bool WinUsb\_ControlTransfer(IntPtr InterfaceHandle, WINUSB\_SETUP\_PACKET SetupPacket, byte\* Buffer, ulong BufferLength, out ulong\* LengthTransferred); // retocar

[DllImport("winusb.dll")]

private static extern bool WinUsb\_FlushPipe(IntPtr InterfaceHandle, byte PipeID);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_Free(IntPtr InterfaceHandle);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_GetAssociatedInterface(IntPtr InterfaceHandle, byte AssociatedInterfaceIndex, out IntPtr AssociatedInterfaceHandle);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_GetCurrentAlternateSetting(IntPtr InterfaceHandle, out byte\* AlternateSetting);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_GetDescriptor(IntPtr InterfaceHandle, byte DescriptorType, byte Index, ushort LanguageID, out byte\* Buffer, ulong BufferLength, out ulong\* LengthTransferred);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_GetInterfacePowerPolicy(IntPtr InterfaceHandle, ulong PolicyType, out ulong\* ValueLength, out void\* Value); // raro

[DllImport("winusb.dll")]

private static extern bool WinUsb\_GetPipePolicy(IntPtr DeviceHandle, byte PipeID, ulong PolicyType, out ulong\* ValueLength, out void\* Value); // raroo

[DllImport("winusb.dll")]

private static extern bool WinUsb\_Initialize(IntPtr DeviceHandle, out IntPtr InterfaceHandle); // out inter

[DllImport("winusb.dll")]

private static extern bool WinUsb\_QueryDeviceInformation(IntPtr DeviceHandle, ulong InformationType, out ulong BufferLength, out ulong Buffer); // raro

[DllImport("winusb.dll")]

private static extern bool WinUsb\_QueryInterfaceSettings(IntPtr InterfaceHandle, byte AlternateSettingNumber, IntPtr UsbAltInterfaceDescriptor); // falta lo ultimo

//[DllImport("winusb.dll")]

//private static extern bool WinUsb\_QueryPipe(IntPtr InterfaceHandle, byte AlternateInterfaceNumber, byte PipeIndex, out WINUSB\_PIPE\_INFORMATION PipeInformation);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_ReadPipe(IntPtr InterfaceHandle, byte PipeID, byte[] Buffer, int BufferLength, out uint LengthTransferred, IntPtr Overlapped);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_ResetPipe(IntPtr InterfaceHandle, byte PipeID);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_SetCurrentAlternateSetting(IntPtr InterfaceHandle, byte AlternateSetting);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_SetInterfacePowerPolicy(IntPtr InterfaceHandle, ulong PolicyType, ulong ValueLength, void\* Value); // creo q mal

[DllImport("winusb.dll")]

private static extern bool WinUsb\_SetPipePolicy(IntPtr InterfaceHandle, byte PipeID, ulong PolicyType, ulong ValueLength, void\* Value);

[DllImport("winusb.dll")]

private static extern bool WinUsb\_WritePipe(IntPtr InterfaceHandle, byte PipeID, byte[] Buffer, int BufferLength, out uint LengthTransferred, IntPtr Overlapped);

#endregion

public IntPtr Init\_PicWinUSB(Guid InterfaceGuid)

{

bool bResult = false;

UInt32 requiredLength = 0;

UInt32 Instance = 0; // Instance > 0 if various PicWinUSB are attached

IntPtr iHandle;

IntPtr hDev;

// [1] Get a handle to the device information set by passing the device interface GUID that you defined in the INF to SetupDiGetClassDevs.

IntPtr deviceInfo = DeviceInterfaceImports.SetupDiGetClassDevs(ref InterfaceGuid, null, IntPtr.Zero, (DeviceInterfaceImports.DIGCF\_PRESENT | DeviceInterfaceImports.DIGCF\_DEVICEINTERFACE));

// [2] Call SetupDiEnumDeviceInterfaces to enumerate the system’s device interfaces and obtain information on your device interface

DeviceInterfaceImports.SP\_DEVICE\_INTERFACE\_DATA interfaceData = new DeviceInterfaceImports.SP\_DEVICE\_INTERFACE\_DATA();

interfaceData.cbSize = Marshal.SizeOf(interfaceData);

bResult = DeviceInterfaceImports.SetupDiEnumDeviceInterfaces(deviceInfo, IntPtr.Zero, ref InterfaceGuid, Instance, ref interfaceData);

// [3] Call SetupDiGetDeviceInterfaceDetail to get detailed data for the device interface.

DeviceInterfaceImports.SetupDiGetDeviceInterfaceDetail(deviceInfo, ref interfaceData, IntPtr.Zero, 0, out requiredLength, IntPtr.Zero);

DeviceInterfaceImports.SP\_DEVICE\_INTERFACE\_DETAIL\_DATA detailData = new DeviceInterfaceImports.SP\_DEVICE\_INTERFACE\_DETAIL\_DATA();

if (IntPtr.Size == 8) detailData.cbSize = 8;// x64 OS

else detailData.cbSize = 5; // x86 OS

bResult = DeviceInterfaceImports.SetupDiGetDeviceInterfaceDetail(deviceInfo, ref interfaceData, ref detailData, requiredLength, out requiredLength, IntPtr.Zero);

// [4] Destroy non used anymore info

DeviceInterfaceImports.SetupDiDestroyDeviceInfoList(deviceInfo);

// [5] Pass the device path to CreateFile to obtain a file handle for the device.

hDev = DeviceInterfaceImports.CreateFile(detailData.DevicePath, FileAccess.ReadWrite, FileShare.ReadWrite, IntPtr.Zero, FileMode.Open, (DeviceInterfaceImports.FILE\_ATTRIBUTE\_NORMAL | DeviceInterfaceImports.FILE\_FLAG\_OVERLAPPED), IntPtr.Zero);

Debug.WriteLine(detailData.DevicePath);

// [6] Call WinUsb\_Initialize to obtain the WinUSB handle for the device.

bResult = WinUsb\_Initialize(hDev, out iHandle);

//if(bResult)Debug.WriteLine("PicWinUSB found!!!!!");

return iHandle;

}

public bool Write\_PicWinUSB(IntPtr iHandle, byte[] dBuffer)

{

byte PipeID = 0x01; // send pipe

int szBuffer = dBuffer.GetLength(0);

uint bytesWritten;

bool bResult;

bResult = WinUsb\_WritePipe(iHandle, PipeID, dBuffer, szBuffer, out bytesWritten, IntPtr.Zero);

//Debug.WriteLine(bytesWritten);

return bResult;

}

public bool Read\_PicWinUSB(IntPtr iHandle, byte[] dBuffer)

{

byte PipeID = 0x81; // recieve pipe

int szBuffer = dBuffer.GetLength(0);

uint bytesRead;

bool bResult;

bResult = WinUsb\_ReadPipe(iHandle, PipeID, dBuffer, szBuffer, out bytesRead, IntPtr.Zero);

//Debug.WriteLine(bytesRead);

return bResult;

}

}

}

And this the code for pic using PICC

#include "C:\Documents and Settings\Administrator\My Documents\test\test.h"

#include <LCD.C>

#define LCD\_TYPE 1

//#include <lcd.c>

void main()

{

// lcd\_init();

setup\_adc\_ports(NO\_ANALOGS|VSS\_VDD);

setup\_adc(ADC\_OFF|ADC\_TAD\_MUL\_0);

setup\_psp(PSP\_DISABLED);

setup\_spi(FALSE);

setup\_wdt(WDT\_OFF);

setup\_timer\_0(RTCC\_INTERNAL);

setup\_timer\_1(T1\_DISABLED);

setup\_timer\_2(T2\_DISABLED,0,1);

setup\_timer\_3(T3\_DISABLED|T3\_DIV\_BY\_1);

setup\_comparator(NC\_NC\_NC\_NC);

setup\_vref(FALSE);

lcd\_init();

setup\_oscillator(False);

printf(lcd\_putc,"'fkhkh\n");

// TODO: USER CODE!!

while (1) {

restart\_wdt();

}

}

The layout and schematics in attachments file with the CD