Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет

Информатики и радиоэлектроники

Факультет компьютерный систем и сетей

Кафедра электронных вычислительных машин

Отчёт

По лабораторной работе №1

На тему

“Асинхронная двунаправленная побайтная передача данных”

Выполнил: Проверил:

ст. гр. 050503 Марцинкевич В.А.

Бедюк С.П.

Минск, 2022

*Program.cs*

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace PCComm

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FrmMain());

}

}

}

*CommunicationManager.cs*

using System;

using System.Text;

using System.Drawing;

using System.IO.Ports;

using System.Windows.Forms;

using System.Linq;

using System.Collections.Generic;

namespace PCComm

{

class CommunicationManager

{

#region Manager Enums

public enum TransmissionType { Text }

public enum MessageType { Incoming, Outgoing, Normal, Warning, Error };

#endregion

#region Manager Variables

private readonly Color[] MessageColor = { Color.Blue, Color.Green, Color.Brown, Color.Orange, Color.Red };

private readonly SerialPort comPort = new SerialPort();

#endregion

#region Manager Properties

public string BaudRate { get; set; }

public string Parity { get; set; }

public string StopBits { get; set; }

public string DataBits { get; set; }

public string PortName { get; set; }

public TransmissionType CurrentTransmissionType { get; set; }

public RichTextBox DisplayWindow { get; set; }

#endregion

#region Manager Constructors

public CommunicationManager(string baud, string par, string sBits, string dBits, string name, RichTextBox rtb)

{

BaudRate = baud;

Parity = par;

StopBits = sBits;

DataBits = dBits;

PortName = name;

DisplayWindow = rtb;

comPort.DataReceived += new SerialDataReceivedEventHandler(ComPort\_DataReceived);

}

public CommunicationManager()

{

BaudRate = string.Empty;

Parity = string.Empty;

StopBits = string.Empty;

DataBits = string.Empty;

PortName = "COM1";

DisplayWindow = null;

comPort.DataReceived += new SerialDataReceivedEventHandler(ComPort\_DataReceived);

}

#endregion

#region WriteData

public void WriteData(string msg)

{

if (!comPort.IsOpen) comPort.Open();

comPort.Write(msg);

DisplayData(MessageType.Outgoing, "-> " + msg + "\n");

}

#endregion

#region DisplayData

[STAThread]

private void DisplayData(MessageType type, string msg)

{

DisplayWindow.Invoke(new EventHandler(delegate

{

DisplayWindow.SelectedText = string.Empty;

DisplayWindow.SelectionFont = new Font(DisplayWindow.SelectionFont, FontStyle.Bold);

DisplayWindow.SelectionColor = MessageColor[(int)type];

DisplayWindow.AppendText(msg);

DisplayWindow.ScrollToCaret();

}));

}

#endregion

#region OpenPort

public bool OpenPort()

{

try

{

if (comPort.IsOpen) comPort.Close();

comPort.BaudRate = int.Parse(BaudRate); //BaudRate

comPort.DataBits = int.Parse(DataBits); //DataBits

comPort.StopBits = (StopBits)Enum.Parse(typeof(StopBits), StopBits); //StopBits

comPort.Parity = (Parity)Enum.Parse(typeof(Parity), Parity); //Parity

comPort.PortName = PortName; //PortName

comPort.Open();

DisplayData(MessageType.Normal, "Port opened at " + DateTime.Now + "\n");

return true;

}

catch (Exception ex)

{

DisplayData(MessageType.Error, ex.Message);

return false;

}

}

#endregion

public bool ChangeBaudRate()

{

if (comPort.IsOpen)

{

comPort.BaudRate = int.Parse(BaudRate); //BaudRate

return true;

}

return false;

}

public bool ClosePort()

{

try

{

if (comPort.IsOpen)

{

comPort.Close();

}

DisplayData(MessageType.Normal, "Port closed at " + DateTime.Now + "\n");

return true;

}

catch (Exception ex)

{

DisplayData(MessageType.Error, ex.Message);

return false;

}

}

#region SetParityValues

public void SetParityValues(object obj)

{

foreach (string str in Enum.GetNames(typeof(Parity)))

{

((ComboBox)obj).Items.Add(str);

}

}

#endregion

#region SetStopBitValues

public void SetStopBitValues(object obj)

{

foreach (string str in Enum.GetNames(typeof(StopBits)))

{

((ComboBox)obj).Items.Add(str);

}

}

#endregion

#region SetPortNameValues

public void SetPortNameValues(object obj)

{

foreach (string str in SerialPort.GetPortNames())

{

((ComboBox)obj).Items.Add(str);

}

}

#endregion

#region comPort\_DataReceived

void ComPort\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

string msg = comPort.ReadExisting();

DisplayData(MessageType.Incoming, "<- " + msg + "\n");

}

#endregion

}

}

*app.config*

<?xml version="1.0"?>

<configuration>

<configSections>

<sectionGroup name="userSettings" type="System.Configuration.UserSettingsGroup, System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">

<section name="PCComm.Properties.Settings" type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" allowExeDefinition="MachineToLocalUser" requirePermission="false"/>

</sectionGroup>

</configSections>

<userSettings>

<PCComm.Properties.Settings>

<setting name="PortName" serializeAs="String">

<value>COM1</value>

</setting>

<setting name="BaudRate" serializeAs="String">

<value>9600</value>

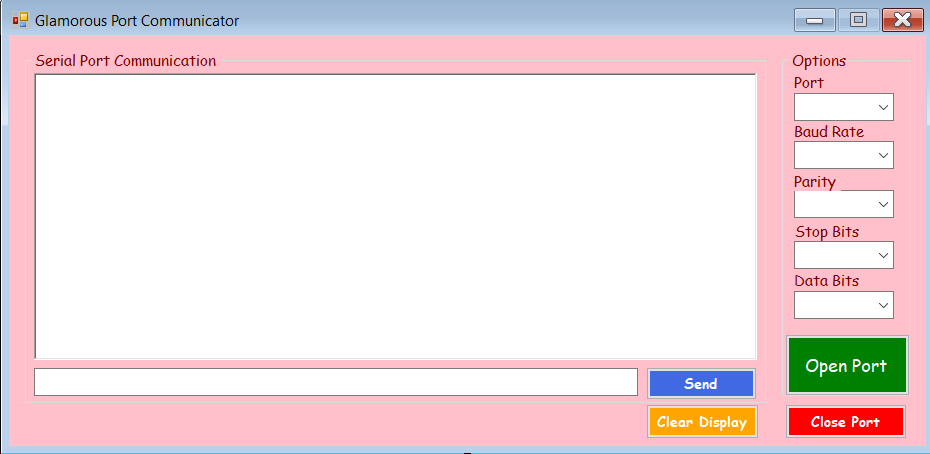
</setting>

</PCComm.Properties.Settings>

</userSettings>

<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup></configuration>

*frmMain.cs*



Задание к лабораторной работе:

1 Разработать программный модуль реализации процедуры передачи (приема) байта информации через последовательный интерфейс.

2 В программах синхронно изменить скорости передачи и приема байта до минимальной и максимальной. Проверить функционирование звена приемопередачи.

3 Установить различные скорости для приемника и передатчика. Проверить функционирование звена приемопередачи.