**Московский государственный технический университет им. Н.Э.Баумана**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Утверждаю: |  |
| Галкин В.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |

**Курсовая работа по дисциплине**

**«Сетевые технологии в АСОИУ»**

**«Локальная безадаптерная сеть»**

Описание программы

(вид документа)

писчая бумага

(вид носителя)

22

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты группы ИУ5-64 |  |
| Матвейчук И.А.  Журавлева У.В.  Повираева М.Л. | "\_\_\_"\_\_\_\_\_\_\_\_\_2017 г.  "\_\_\_"\_\_\_\_\_\_\_\_\_2017 г.  "\_\_\_"\_\_\_\_\_\_\_\_\_2017 г. |

Москва - 2017

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**1. Общие сведения.**

Наименование: «Локальная безадаптерная сеть»

Программа выполняется на языке программирования С# и работает под управлением операционной системы Windows 7 и выше.

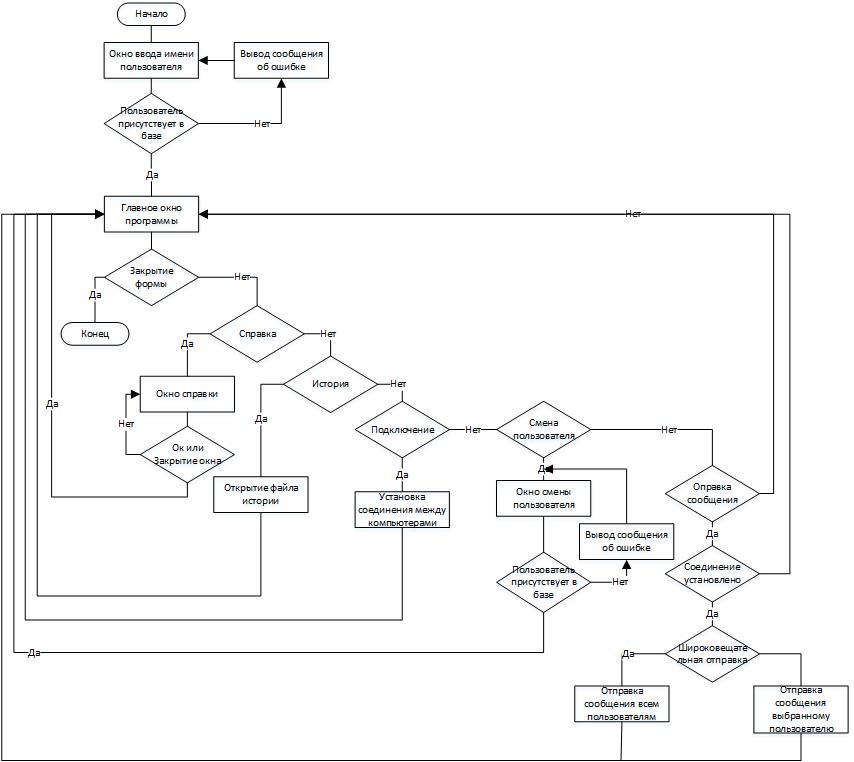
**2. Назначение разработки.**

Программа должна реализовывать функцию передачи коротких текстовых сообщений между ПЭВМ, соединенными через интерфейс RS-232C нуль-модемным кабелем с поддержкой широковещательной рассылки сообщений.

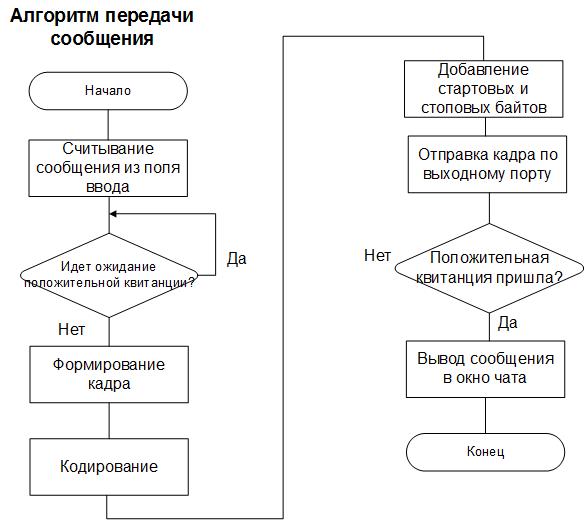
**3. Описание логической структуры.**

**3.1. Алгоритм интерфейсной части программы.**

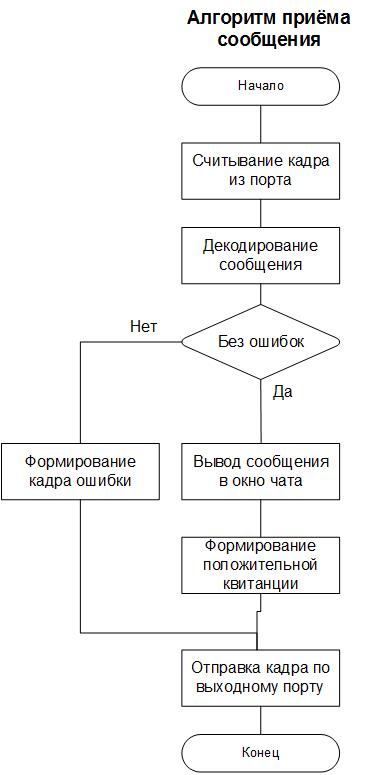
Алгоритм интерфейсной части приведен следующем рисунке.



**3.2. Алгоритм передачи сообщения.**



**3.3 Алгоритм приема сообщения.**



**4. Используемые технические средства.**

Программа должна работать на IBM-совместимой ЭВМ следующей конфигурации:

1. Центральный процессор Perntium I или выше;
2. Объем оперативной памяти 32 Мб;
3. Видеоадаптер и монитор VGA и выше;
4. Стандартная клавиатура;
5. Свободного пространства на жестком диске 3Мб;

Для работы программы требуются IBM-совместимые компьютеры, соединенных нуль-модемным кабелем через интерфейс RS-232C. Или же 1 компьютер, на котором установлена программа Virtual Serial Port Emulator с произведенными настройками (созданы 3 пары com-портов: COM1-COM2, COM3-COM4, COM5-COM6)

**5. Входные и выходные данные.**

**5.1. Входные данные.**

Входными данными является текстовое сообщение, набранное пользователем.

**5.2. Выходные данные.**

Выходными данными являются:

1. текст переданного сообщения на ПЭВМ;
2. сообщения об ошибках и выполнении передачи.

**6. Спецификация данных.**

**6.1. Внутренние данные.**

Данные указаны без учета стартовых и стоповых байтов.

Запрос на соединение:

|  |  |  |
| --- | --- | --- |
| Наименование | Тип поля | Размер (байт) |
| UPLINK | Byte | 1 |

Поддержание соединения:

|  |  |  |
| --- | --- | --- |
| Наименование | Тип поля | Размер (байт) |
| LINKACTIVE | Byte | 1 |

Положительная квитанция:

|  |  |  |
| --- | --- | --- |
| Наименование | Тип поля | Размер (байт) |
| ACK | Byte | 1 |

Разрыв соединения:

|  |  |  |
| --- | --- | --- |
| Наименование | Тип поля | Размер (байт) |
| DOWNLINK | Byte | 1 |

Информационный блок:

|  |  |  |
| --- | --- | --- |
| Наименование | Тип поля | Размер (байт) |
| DAT | Byte | 1 |

**6.2. Структура сообщения.**

Программа работает с текстовыми сообщениями стандарта ANSI размером не более 255 символов.

**7. Спецификация функций.**

public void SetParityToComboBox(object obj) – задает возможные значения четности

private void Form1\_Load(object sender, EventArgs e) – действия при загрузке формы настроек

private void button1\_Click(object sender, EventArgs e) – действия при нажатии на кнопку ОК на форме настроек

public void OpenForm1(bool \_this) – действия при открытии формы настроек

private void button1\_Click(object sender, EventArgs e) – действия при нажатии кнопки «Отправить»

private void Form2\_Load(object sender, EventArgs e) – действия при загрузке главной формы

private void connectButton\_Click(object sender, EventArgs e) – действия при нажатии на кнопку «Подключиться»

private void disconnectButton\_Click(object sender, EventArgs e) – действия при нажатии на кнопку «Сменить пользователя»

private void connectionTimer1\_Tick(object sender, EventArgs e) – таймер физического соединения первого COM-порта

private void connectionTimer2\_Tick(object sender, EventArgs e) – таймер физического соединения второго COM-порта

private void logicConnectionTimer1\_Tick(object sender, EventArgs e) - таймер логического соединения первого COM-порта

private void logicConnectionTimer2\_Tick(object sender, EventArgs e) - таймер логического соединения второго COM-порта

private void comPort1\_DataReceived(object sender, SerialDataReceivedEventArgs e) – событие получения данных на первый COM-порт

private void comPort2\_DataReceived(object sender, SerialDataReceivedEventArgs e) – событие получения данных на второй COM-порт

private void send\_timer1\_Tick(object sender, EventArgs e) – таймер ожидания квитанции первого COM-порта

private void send\_timer2\_Tick(object sender, EventArgs e) – таймер ожидания квитанции второго COM-порта

private void Form2\_FormClosing(object sender, FormClosingEventArgs e) – действия при закрытии главной формы

private void usersListBox\_Click(object sender, EventArgs e) – действия при клике на список активных пользователей

private void checkChanged(object sender, EventArgs e) – действия, при выборе «Отправить всем»

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e) – действия, при клике на пункт меню «О программе»

private void посмотретьИсториюToolStripMenuItem\_Click(object sender, EventArgs e) – действия, при клике на пункт меню «О программе»

private void SetParametrs(COMPORT port1, COMPORT port2) – установка начальных параметров COM-портов

private void Form4\_Load(object sender, EventArgs e) – действия, при загрузке формы смены пользователя

private void OKButton\_Click(object sender, EventArgs e) – действия, при нажатии на «ОК» на форме смены пользователя

public void Connect(COMPORT port1, COMPORT port2) – установка физического соединения

public void WriteData(string msg, FrameType currentFrameType, bool \_private, COMPORT comPort) - запись в COM-порт

private void DisplayData(MessageType type, string msg, bool isMsg, ListBox list, COMPORT port) - вывод сообщений на форму

public void FrameAnalysis(byte mybyte, COMPORT comPort, ListBox list, ListBox list2, COMPORT port) - анализ приходящих кадров

public void ShowUsers(string name, ListBox list) - обновление активных пользователей на форме

public void WriteHistory(string msg) - запись в файл истории

public string Decode(byte[] array) – декодирует набор байтов в строку типа string с помощью циклического кода

public byte[] Code(string str) – кодирует строку типа string при помощи циклического кода

public static string division(string ostatok, string divider) – алгоритм деления двоичных векторов и получения остатка

public static string conc(string str1, string str2) – алгоритм конкатенации двоичных векторов

public static string CycleCoding(string str) – кодирует двоичный 11-ти разрядный вектор циклическим кодом

public static string CycleDecoding(string str) – декодирует двоичный 11-ти разрядный вектор циклическим кодом и сообщает об ошибке

public static byte[] StrToByte(string str) – перевод строки, состоящей из 0 и 1 в вектор байт

public byte[] Code1(string str) – переводит строку типа string в массив байтов

public string Decode1(byte[] arr) - переводит массив байтов в строку типа string

***Листинг основных функций:***

**Form1.cs (форма настроек)**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.IO.Ports;

namespace ST

{

public partial class Form1 : Form

{

public static string CurrentUser;

public static string com1\_num;

public static string com2\_num;

public static string com1\_speed;

public static string com2\_speed;

public static SerialPort ComPort1 = new SerialPort();

public static SerialPort ComPort2 = new SerialPort();

public Form1 current\_form = (ST.Form1)Form1.ActiveForm;

public bool port1\_check;

public Form1()

{

InitializeComponent();

}

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

{

errorLabel.Visible = false;

string[] ports = SerialPort.GetPortNames();

comPortNum1.Items.Clear();

comPortNum1.Items.AddRange(ports);

comPortNum2.Items.Clear();

comPortNum2.Items.AddRange(ports);

comPortNum1.SelectedIndex = 0;

comPortNum2.SelectedIndex = 2;

speedComboBox1.SelectedIndex = 0;

speedComboBox2.SelectedIndex = 0;

}

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

{

port1\_check = false;

var allUsers = new List<string> { "1", "2", "3", "Ира", "Уля", "Марина", "Ira", "Ulya", "Marina"};

if (!allUsers.Contains(userName.Text))

errorLabel.Visible = true;

else

{

#region Настойка параметров передачи

com1\_speed = speedComboBox1.Items[speedComboBox1.SelectedIndex].ToString();

com2\_speed = speedComboBox2.Items[speedComboBox2.SelectedIndex].ToString();

try

{

com1\_num = comPortNum1.Items[comPortNum1.SelectedIndex].ToString();

ComPort1.PortName = com1\_num;

ComPort1.Open();

ComPort1.DiscardInBuffer();

port1\_check = true;

}

catch

{

MessageBox.Show("Не удалось открыть указанные порты");

}

if (port1\_check)

{

try

{

com2\_num = comPortNum2.Items[comPortNum2.SelectedIndex].ToString();

ComPort2.PortName = com2\_num;

ComPort2.Open();

ComPort2.DiscardInBuffer();

ConnectionManager.Speeds.Add(ComPort1, Convert.ToInt32(com1\_speed));

ConnectionManager.Speeds.Add(ComPort2, Convert.ToInt32(com2\_speed));

CurrentUser = userName.Text;

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

catch

{

ComPort1.Close();

MessageBox.Show("Не удалось открыть указанные порты");

}

}

#endregion

}

}

public void OpenForm1(bool \_this)

{

if (\_this)

this.Show();

Form ifrm = Application.OpenForms[0];

ifrm.Show();

}

}

}

**Form2.cs (главная форма)**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.IO;

using System.IO.Ports;

using System.Threading;

namespace ST

{

public class COMPORT

{

public SerialPort port;

public Int32 speed;

public bool my\_logic\_state; //логическое состояние этого порта

public bool p\_connected; //флаг на установку физического подключения

public bool p\_active; //флаг на поддержание физического соединения

public bool l\_connected; //флаг на установку логического соединения

public bool linkactive; //флаг на поддержание логического соединения

public int linkactive\_try; //колво попыток для поддержания пользователя в активном состоянии

public bool timerstop;

public int count\_try;

public System.Windows.Forms.Timer logicConnectionTimer = new System.Windows.Forms.Timer(); //таймер на установку логического соединения

public System.Windows.Forms.Timer send\_timer = new System.Windows.Forms.Timer(); //таймер на отправку сообщения

public System.Windows.Forms.Timer connectionTimer = new System.Windows.Forms.Timer(); //таймер физического соединения

}

public partial class Form2 : Form

{

private const byte START = 0xFF;

private ConnectionManager \_manager = new ConnectionManager();

private PhysicalConnection \_phys\_manager = new PhysicalConnection();

public static string selected\_user;

public static string this\_path = Directory.GetCurrentDirectory();

public static string path;

public static COMPORT port1 = new COMPORT();

public static COMPORT port2 = new COMPORT();

public static List<COMPORT> ports = new List<COMPORT>();

public static string current\_user;

private string message; //сообщение

public Form2()

{

InitializeComponent();

}

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

{

if (!port1.send\_timer.Enabled && !port2.send\_timer.Enabled)

{

message = MsgTextBox.Text;

ConnectionManager.last\_msg = message;

if (message == "") emptyMsgLabel.Visible = true;

else

{

if (sendToAllCheck.Checked == true)

{

if (usersListBox.Items.Count == 0) MessageBox.Show("Нет пользователей в сети");

else

{

for (var i=0; i<usersListBox.Items.Count; i++)

{

if (ConnectionManager.Users.ContainsValue(Convert.ToString(usersListBox.Items[i])))

{

foreach (var a in ConnectionManager.Users.Keys)

{

if (ConnectionManager.Users[a] == Convert.ToString(usersListBox.Items[i]))

{

\_manager.WriteData(message, ConnectionManager.FrameType.DAT, false, a);

a.timerstop = false;

a.send\_timer.Start();

}

}

}

}

msgListBox.Items.Add("Я > " + message);

msgListBox.TopIndex = msgListBox.Items.Count - 1;

\_manager.WriteHistory("Я > " + message);

MsgTextBox.Clear();

ConnectionManager.last\_msg\_stat = false;

}

}

else

{

if (ConnectionManager.Users.ContainsValue(selected\_user))

{

foreach (var a in ConnectionManager.Users.Keys)

{

if (ConnectionManager.Users[a] == selected\_user)

{

\_manager.WriteData(message, ConnectionManager.FrameType.DAT, true, a);

a.timerstop = false;

a.send\_timer.Start();

}

}

}

msgListBox.Items.Add("Я [" + selected\_user + "]> " + message);

msgListBox.TopIndex = msgListBox.Items.Count - 1;

\_manager.WriteHistory("Я [" + selected\_user + "]> " + message);

MsgTextBox.Clear();

ConnectionManager.last\_msg\_stat = true;

}

}

}

else MessageBox.Show("Подождите, идет отправка");

}

private void Form2\_Load(object sender, EventArgs e)

{

this.FormClosing += Form2\_FormClosing;

usersListBox.Click += usersListBox\_Click;

path = Path.Combine(this\_path, "History\_" + current\_user + ".txt");

SetParametrs(port1, port2);

sendToAllCheck.CheckedChanged += checkChanged;

current\_user = Form1.CurrentUser;

disconnectButton.Enabled = false;

sendButton.Enabled = false;

path = Path.Combine(this\_path, "History\_" + current\_user + ".txt");

if (File.Exists(path))

File.Delete(path);

File.Create(path).Dispose();

ConnectionManager.f2.CreateForm2(sendButton, connectButton, disconnectButton, msgListBox, MsgTextBox, usersListBox, progressBar1);

}

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

{

connectButton.Enabled = false;

progressBar1.Visible = true;

\_phys\_manager.Connect(port1, port2);

}

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

{

port1.my\_logic\_state = false;

if (port1.p\_active)

\_manager.WriteData(null, ConnectionManager.FrameType.DOWNLINK, false, port1);

port2.my\_logic\_state = false;

if (port2.p\_active)

\_manager.WriteData(null, ConnectionManager.FrameType.DOWNLINK, false, port2);

usersListBox.Items.Clear();

msgListBox.Items.Clear();

Form4 f4 = new Form4();

f4.ShowDialog();

}

#region Timer на установку физического соединения

private void connectionTimer1\_Tick(object sender, EventArgs e)

{

if (port1.port.DsrHolding && !port1.p\_active)

{

progressBar1.Value = 25;

\_manager.WriteData(null, ConnectionManager.FrameType.UPLINK, false, port1);

port1.my\_logic\_state = true;

port1.p\_active = true;

port1.logicConnectionTimer.Start();

}

else if (!port1.port.DsrHolding && port1.p\_active)

{

msgListBox.Items.Add("SYSTEM> Проверьте соединение 1 кабеля");

msgListBox.TopIndex = msgListBox.Items.Count - 1;

\_manager.WriteHistory("SYSTEM> Проверьте соединение 1 кабеля");

port1.logicConnectionTimer.Stop();

//ConnectionManager.ActivePorts[port1.port] = false;

port1.l\_connected = false;

port1.linkactive = false;

port1.my\_logic\_state = false;

port1.p\_connected = false;

port1.p\_active = false;

if (usersListBox.Items.Contains(ConnectionManager.Users[port1]))

usersListBox.Items.Remove(ConnectionManager.Users[port1]);

}

}

private void connectionTimer2\_Tick(object sender, EventArgs e)

{

if (port2.port.DsrHolding && !port2.p\_active)

{

progressBar1.Value = 25;

\_manager.WriteData(null, ConnectionManager.FrameType.UPLINK, false, port2);

port2.my\_logic\_state = true;

port2.p\_active = true;

port2.logicConnectionTimer.Start();

}

else if (!port2.port.DsrHolding && port2.p\_active)

{

msgListBox.Items.Add("SYSTEM> Проверьте соединение 2 кабеля");

msgListBox.TopIndex = msgListBox.Items.Count - 1;

\_manager.WriteHistory("SYSTEM> Проверьте соединение 2 кабеля");

port2.logicConnectionTimer.Stop();

//ConnectionManager.ActivePorts[port2.port] = false;

port2.l\_connected = false;

port2.linkactive = false;

port2.my\_logic\_state = false;

port2.p\_connected = false;

port2.p\_active = false;

if (usersListBox.Items.Contains(ConnectionManager.Users[port2]))

usersListBox.Items.Remove(ConnectionManager.Users[port2]);

}

}

#endregion

#region Timer на уставку логического соединения

private void logicConnectionTimer1\_Tick(object sender, EventArgs e)

{

if (!port1.l\_connected && port1.my\_logic\_state)

\_manager.WriteData(null, ConnectionManager.FrameType.UPLINK, false, port1);

else if (port1.my\_logic\_state)

{

if (port1.linkactive\_try != 3)

{

\_manager.WriteData(null, ConnectionManager.FrameType.LINKACTIVE, false, port1);

port1.linkactive\_try--;

}

else

{

//ConnectionManager.ActivePorts[port1.port] = false;

port1.l\_connected = false;

port1.linkactive\_try = 0;

}

}

}

private void logicConnectionTimer2\_Tick(object sender, EventArgs e)

{

if (!port2.l\_connected && port2.my\_logic\_state)

\_manager.WriteData(null, ConnectionManager.FrameType.UPLINK, false, port2);

else if (port2.my\_logic\_state)

{

if (port2.linkactive\_try != 3)

{

\_manager.WriteData(null, ConnectionManager.FrameType.LINKACTIVE, false, port2);

port2.linkactive\_try--;

}

else

{

port2.l\_connected = false;

port2.linkactive\_try = 0;

}

}

}

#endregion

#region Событие получения данных на порт

private void comPort1\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

Thread.Sleep(1000);

if (Form1.ComPort1.BytesToRead != 0)

{

switch (Form1.ComPort1.ReadByte())

{

case START:

{

\_manager.FrameAnalysis((byte)Form1.ComPort1.ReadByte(), port2, msgListBox, usersListBox, port1);

}

break;

default:

\_manager.WriteData(null, ConnectionManager.FrameType.RET, true, port1);

break;

}

}

}

private void comPort2\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

Thread.Sleep(1000);

if (Form1.ComPort2.BytesToRead != 0)

{

switch (Form1.ComPort2.ReadByte())

{

case START:

{

\_manager.FrameAnalysis((byte)Form1.ComPort2.ReadByte(), port1, msgListBox, usersListBox, port2);

}

break;

default:

\_manager.WriteData(null, ConnectionManager.FrameType.RET, true, port2);

break;

}

}

}

#endregion

private void send\_timer1\_Tick(object sender, EventArgs e)

{

port1.count\_try++;

if (port1.timerstop && port1.count\_try<10)

{

port1.send\_timer.Stop();

port1.count\_try = 0;

}

else if (port1.count\_try==10)

{

port1.send\_timer.Stop();

port1.count\_try = 0;

\_manager.WriteData(null, ConnectionManager.FrameType.RET\_DAT, true, port1);

}

}

private void send\_timer2\_Tick(object sender, EventArgs e)

{

port2.count\_try++;

if (port2.timerstop && port2.count\_try < 10)

{

port2.send\_timer.Stop();

port2.count\_try = 0;

}

else if (port2.count\_try == 10)

{

port2.send\_timer.Stop();

port2.count\_try = 0;

\_manager.WriteData(null, ConnectionManager.FrameType.RET\_DAT, true, port2);

}

}

private void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{

Form ifrm = Application.OpenForms[0];

ifrm.Close();

}

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

{

selected\_user = Convert.ToString(usersListBox.Items[usersListBox.SelectedIndex]);

if (!sendToAllCheck.Checked)

sendToLabel.Text = "Получатель: " + selected\_user;

}

private void checkChanged(object sender, EventArgs e)

{

if (sendToAllCheck.Checked)

{

sendToLabel.Text = "Получатель: все";

}

else

{

if (selected\_user == "")

sendToLabel.Text = "Получатель не выбран";

else sendToLabel.Text = "Получатель: " + selected\_user;

}

}

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

{

InfoForm info = new InfoForm();

info.ShowDialog();

}

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

{

System.Diagnostics.Process.Start(path);

}

private void SetParametrs(COMPORT port1, COMPORT port2)

{

port1.port = Form1.ComPort1;

port1.port.DataReceived += comPort1\_DataReceived;

port1.speed = Convert.ToInt32(Form1.com1\_speed);

port1.l\_connected = false;

port1.linkactive = false;

port1.my\_logic\_state = false;

port1.p\_active = false;

port1.p\_connected = false;

port1.timerstop = false;

port1.linkactive\_try = 0;

port1.logicConnectionTimer.Tick += new EventHandler(logicConnectionTimer1\_Tick);

port1.connectionTimer.Tick += new EventHandler(connectionTimer1\_Tick);

port1.send\_timer.Tick += new EventHandler(send\_timer1\_Tick);

port1.logicConnectionTimer.Interval = 7000;

port1.send\_timer.Interval = 500;

port2.port = Form1.ComPort2;

port2.port.DataReceived += comPort2\_DataReceived;

port2.speed = Convert.ToInt32(Form1.com2\_speed);

port2.l\_connected = false;

port2.linkactive = false;

port2.my\_logic\_state = false;

port2.p\_active = false;

port2.p\_connected = false;

port1.timerstop = false;

port2.linkactive\_try = 0;

port2.logicConnectionTimer.Tick += new EventHandler(logicConnectionTimer2\_Tick);

port2.connectionTimer.Tick += new EventHandler(connectionTimer2\_Tick);

port1.send\_timer.Tick += new EventHandler(send\_timer2\_Tick);

port2.logicConnectionTimer.Interval = 7000;

port2.send\_timer.Interval = 500;

}

}

}

**Form4.cs (форма смены пользователя)**

using System.Windows.Forms;

using System.IO;

namespace ST

{

public partial class Form4 : Form

{

private PhysicalConnection \_phys\_manager = new PhysicalConnection();

private ConnectionManager \_manager = new ConnectionManager();

public Form4()

{

InitializeComponent();

}

private void Form4\_Load(object sender, EventArgs e)

{

errorLabel.Visible = false;

}

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

{

var allUsers = new List<string> { "1", "2", "Ира", "Уля", "Марина", "Ira", "Ulya", "Marina" };

if (!allUsers.Contains(userTextBox.Text))

errorLabel.Visible = true;

else

{

Form2.current\_user = userTextBox.Text;

Form2.port1.my\_logic\_state = true;

if (Form2.port1.l\_connected)

\_manager.WriteData(null, ConnectionManager.FrameType.SUNH, false, Form2.port1);

Form2.port2.my\_logic\_state = true;

if (Form2.port2.l\_connected)

\_manager.WriteData(null, ConnectionManager.FrameType.SUNH, false, Form2.port2);

Form2.path = Path.Combine(Form2.this\_path, "History\_" + Form2.current\_user + ".txt");

if (File.Exists(Form2.path))

File.Delete(Form2.path);

File.Create(Form2.path).Dispose();

this.Close();

}

}

}

}

**ConnectionManager.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

using System.IO.Ports;

using System.Windows.Forms;

namespace ST

{

class Form2Form

{

public Button send;

public Button disconnect;

public Button connect;

public ListBox messages;

public TextBox msg\_to\_send;

public ListBox users;

public ProgressBar bar;

public void CreateForm2(Button send1, Button connect1, Button disconnect1, ListBox messages1, TextBox msg\_to\_send1, ListBox users1, ProgressBar bar1)

{

send = send1;

connect = connect1;

disconnect = disconnect1;

messages = messages1;

msg\_to\_send = msg\_to\_send1;

users = users1;

bar = bar1;

}

}

class ConnectionManager

{

public static Form2Form f2 = new Form2Form();

public PhysicalConnection \_phys\_connect = new PhysicalConnection();

private CycleCode coder = new CycleCode();

public enum MessageType

{

Public,

Private

};

public enum FrameType : byte

{

UPLINK, //кадр установки соединения

ACK\_UPLINK,

LINKACTIVE,

ACK\_LINKACTIVE, //кадры -квитанция

DAT,

ACK,

RET\_DAT,

RET, //кадры - ARQ

DOWNLINK, //кадр разрыва соединения

SUNH, //Кадр синхронизации пользователей

ACK\_SUNH,

SUNHP, //Кадр синхронизации параметров

}

private const byte Start = 0xFF; //стартовый бит

public static Dictionary<COMPORT, string> Users = new Dictionary<COMPORT, string>();

public static Dictionary<SerialPort, Int32> Speeds = new Dictionary<SerialPort, Int32>();

//public static Dictionary<SerialPort, bool> ActivePorts = new Dictionary<SerialPort, bool>();

private int reSendCount = 3;

private byte previousOperation;

public static string last\_msg;

public static bool last\_msg\_stat;

public void WriteData(string msg, FrameType currentFrameType, bool \_private, COMPORT comPort) //Запись в порт

{

//lastMsg = msg;

var frameFields = new List<byte> { Start, (byte)currentFrameType };

switch (currentFrameType)

{

case FrameType.DAT:

try

{

if (string.IsNullOrEmpty(msg))

return;

if (\_private) { frameFields.Add(1); }

else { frameFields.Add(0); }

frameFields.Add((byte)coder.Code(msg).Length);

frameFields.AddRange(coder.Code(msg));

comPort.port.Write(frameFields.ToArray(), 0, frameFields.Count);

}

catch (FormatException ex)

{

DisplayData(MessageType.Public, ex.Message, false, f2.messages, comPort);

}

break;

case FrameType.ACK\_SUNH:

//frameFields = new List<byte> { Start, (byte)currentFrameType };

frameFields.Add((byte)coder.Code1(msg).Length);

frameFields.AddRange(coder.Code1(msg));

comPort.port.Write(frameFields.ToArray(), 0, frameFields.Count);

break;

case FrameType.SUNHP:

//frameFields = new List<byte> { Start, (byte)currentFrameType };

frameFields.Add((byte)coder.Code1(msg).Length);

frameFields.AddRange(coder.Code1(msg));

comPort.port.Write(frameFields.ToArray(), 0, frameFields.Count);

break;

default:

//DisplayData(MessageType.Outgoing, DateTime.Now + " отправили " + currentFrameType + "\n", false, f2.messages, comPort);

comPort.port.Write(frameFields.ToArray(), 0, 2);

break;

}

}

private void DisplayData(MessageType type, string msg, bool isMsg, ListBox list, COMPORT port)

{

string user;

string msg\_to\_display;

try { user = Users[port]; }

catch { user = ""; }

if (type == MessageType.Private)

{

msg\_to\_display = user + " [личное] > " + msg;

}

else

{

if (isMsg) msg\_to\_display = user + "> " + msg;

else msg\_to\_display = "SYSTEM> " + msg;

}

list.Invoke(new Action(() => { list.Items.Add(msg\_to\_display); }));

list.Invoke(new Action(() => { list.TopIndex = list.Items.Count - 1; }));

WriteHistory(msg\_to\_display);

list.SelectedItem = -1;

} //Вывод сообщений на форму

public void FrameAnalysis(byte mybyte, COMPORT comPort, ListBox list, ListBox list2, COMPORT port) //Анализ приходящих кадров

{

previousOperation = mybyte;

switch (mybyte)

{

case (byte)FrameType.UPLINK:

if (port.my\_logic\_state)

{

WriteData(null, FrameType.ACK\_UPLINK, false, port);

}

break;

case (byte)FrameType.ACK\_UPLINK:

f2.bar.Invoke(new Action(() => { f2.bar.Value = 50; }));

port.l\_connected = true;

if (port.p\_connected)

WriteData(null, FrameType.SUNH, false, port);

else

{

WriteData(Convert.ToString(port.speed), FrameType.SUNHP, false, port);

}

break;

case (byte)FrameType.LINKACTIVE:

if (port.my\_logic\_state)

{

WriteData(null, FrameType.ACK\_LINKACTIVE, false, port);

}

break;

case (byte)FrameType.ACK\_LINKACTIVE:

port.linkactive\_try = 0;

break;

case (byte)FrameType.SUNHP:

var Count = Convert.ToInt32(port.port.ReadByte());

var text = new byte[Count];

port.port.Read(text, 0, Count);

var decoded = coder.Decode1(text);

if (Convert.ToInt32(decoded) >= port.speed)

port.port.BaudRate = port.speed;

else if (Convert.ToInt32(decoded) < port.speed)

port.port.BaudRate = Convert.ToInt32(decoded);

port.p\_connected = true;

f2.bar.Invoke(new Action(() => { f2.bar.Value = 75; }));

WriteData(null, FrameType.SUNH, false, port);

break;

case (byte)FrameType.SUNH:

WriteData(Form2.current\_user, FrameType.ACK\_SUNH, false, port);

break;

case (byte)FrameType.ACK\_SUNH:

f2.bar.Invoke(new Action(() => { f2.bar.Value = 100; }));

Count = Convert.ToInt32(port.port.ReadByte());

text = new byte[Count];

port.port.Read(text, 0, Count);

decoded = coder.Decode1(text);

if (Users.ContainsKey(port))

Users.Remove(port);

try

{

Users.Add(port, decoded);

}

catch

{

Form2.port1.port = Form1.ComPort1;

Form2.port2.port = Form1.ComPort2;

}

f2.disconnect.Invoke(new Action(() => { f2.disconnect.Enabled = true; }));

f2.messages.Invoke(new Action(() => { DisplayData(MessageType.Public, "Пользователь " + decoded + " активен", false, f2.messages, port); }));

f2.messages.Invoke(new Action(() => { f2.messages.TopIndex = f2.messages.Items.Count - 1; }));

ShowUsers(decoded, list2);

f2.bar.Invoke(new Action(() => { f2.bar.Visible = false; }));

f2.send.Invoke(new Action(() => { f2.send.Enabled = true; }));

break;

case (byte)FrameType.DOWNLINK:

DisplayData(MessageType.Public, "Пользоваель " + Users[port] + " вышел из сети", false, list, port);

f2.messages.Invoke(new Action(() => { f2.messages.TopIndex = f2.messages.Items.Count - 1; }));

port.l\_connected = false; //сделали порт логически неактивным

if (f2.users.Items.Contains(Users[port]))

f2.users.Invoke(new Action(() => { f2.users.Items.Remove(Users[port]); })); //убрали имя из списка

break;

case (byte)FrameType.DAT:

var sendUser = port.port.ReadByte();

Count = Convert.ToInt32(port.port.ReadByte());

text = new byte[Count];

port.port.Read(text, 0, Count);

decoded = coder.Decode(text);

if (decoded == "") WriteData(null, FrameType.RET\_DAT, false, port);

else

{

if (sendUser == 0)

DisplayData(MessageType.Public, decoded, true, list, port);

else if (sendUser == 1)

{

DisplayData(MessageType.Private, decoded, true, list, port);

}

WriteData(null, FrameType.ACK, false, port);

}

break;

case (byte)FrameType.ACK:

port.timerstop = true;

list);

break;

default:

DisplayData(MessageType.Public, DateTime.Now + " unknow frame \n", false, list, port);

break;

case (byte)FrameType.RET\_DAT:

if (reSendCount > 0)

{

reSendCount--;

if (last\_msg\_stat)

WriteData(last\_msg, FrameType.DAT, true, port);

else

WriteData(last\_msg, FrameType.DAT, false, port);

}

else

{

f2.messages.Invoke(new Action(() => { f2.messages.Items.Add(" ...не удалось отправить сообщение"); }));

port.send\_timer.Stop();

}

break;

case (byte)FrameType.RET:

if (previousOperation != (byte)FrameType.DAT)

FrameAnalysis(previousOperation, comPort, list, list2, port);

break;

}

}

public void ShowUsers(string name, ListBox list) //Обновление активных пользователей на форме

{

list.Invoke(new Action(() => { list.Items.Add(name); }));

}

public void WriteHistory(string msg) //Запись в файл истории

{

File.AppendAllText(Form2.path, msg + Environment.NewLine, Encoding.Default);

}

}

}

**PhysicalConnection.cs**

namespace ST

{

class PhysicalConnection

{

public void Connect(COMPORT port1, COMPORT port2)

{

port1.port.DtrEnable = true;

port1.my\_logic\_state = true;

port1.connectionTimer.Start();

port2.port.DtrEnable = true;

port2.my\_logic\_state = true;

port2.connectionTimer.Start();

}

}

}

**CycleCode.cs**

using System;

using System.Text;

namespace ST

{

class CycleCode

{

public static bool mistake;

public string Decode(byte[] array)

{

string single;

string[] str\_arr = new string[array.Length];

string result = "";

for (int i = 0; i < array.Length; i++) //переводим массив в двоичный формат

str\_arr[i] = Convert.ToString(array[i], 2);

for (int i = 0; i < str\_arr.Length; i++) //увеличиваем длинну каждого элемента массива до 8 символов

{

if (str\_arr[i].Length < 8)

{

for (int j = str\_arr[i].Length; j < 8; j++)

str\_arr[i] = "0" + str\_arr[i];

}

}

for (int i = 0; i < str\_arr.Length; i++) //объединяем в одну двоичную строку

result += str\_arr[i];

result = result.TrimStart('0');

while (result.Length % 15 != 0)

result = "0" + result;

byte[] array1 = new byte[result.Length / 15];

int k = 0;

for (int i = 0; i < result.Length; i += 15) //раскодируем в массив байт

{

single = result.Substring(i, 15);

single = CycleDecoding(single);

if (mistake) break;

array1[k] = Convert.ToByte(single, 2);

k++;

}

if (mistake) return "";

else

{

char[] char\_array = Encoding.Unicode.GetChars(array1);

string str = new string(char\_array);

return str;

}

}

public byte[] Code(string str)

{

string[] str\_arr = new string[str.Length \* 2];

string result = ""; // выходная строка

byte[] array = new byte[str.Length \* 2];

array = Encoding.Unicode.GetBytes(str); //получаем массив байтов исходного сообщения

for (int i = 0; i < array.Length; i++) //переводим массив в двоичный формат

str\_arr[i] = Convert.ToString(array[i], 2);

for (int i = 0; i < str\_arr.Length; i++) //увеличиваем длинну каждого элемента массива до 11 символов

{

if (str\_arr[i].Length < 11)

{

for (int j = str\_arr[i].Length; j < 11; j++)

str\_arr[i] = "0" + str\_arr[i];

}

}

for (int i = 0; i < str\_arr.Length; i++) //кодируем каждый элемент

{

str\_arr[i] = CycleCoding(str\_arr[i]);

}

for (int i = 0; i < str\_arr.Length; i++) //объединяем в одну двоичную строку

result += str\_arr[i];

array = StrToByte(result); //переводим строку в массив байт

return array;

}

public static string division(string ostatok, string divider)

{

while (ostatok.Length >= divider.Length)

{

string sub = divider;

while (!(ostatok.Length == sub.Length))

sub += "0";

ostatok = conc(sub, ostatok).TrimStart('0');

}

return ostatok;

}

public static string conc(string str1, string str2)

{

while (!(str1.Length == str2.Length))

str2 = "0" + str2;

string otv = "";

for (int i = 0; i < str1.Length; i++)

otv += str1[i] ^ str2[i];

return otv;

}

public static string CycleCoding(string str)

{

string g = "10011"; //порождающий полином

str += "0000"; //сдвиг

string ost = division(str, g); //деление на порождающий полином

str = conc(str, ost); //конкатениция с остатком

return str;

}

public static string CycleDecoding(string str)

{

string g = "10011"; //порождающий полином

string ost = division(str, g); //деление на порождающий полином

if (ost == "") //проверяем синдром

{

mistake = false;

return str.Substring(3, 8);

}

else

{

mistake = true;

return str.Substring(3, 8);

}

}

public static byte[] StrToByte(string str)

{

str = str.TrimStart('0');

while (str.Length % 8 != 0)

str = "0" + str;

byte[] arr = new byte[str.Length / 8];

int j = 0;

for (int i = 0; i < str.Length; i += 8)

{

arr[j] = Convert.ToByte((str.Substring(i, 8)), 2);

j++;

}

return arr;

}

public byte[] Code1(string str)

{

byte[] arr = Encoding.Unicode.GetBytes(str);

return arr;

}

public string Decode1(byte[] arr)

{

char[] char\_array = Encoding.Unicode.GetChars(arr);

string str = new string(char\_array);

return str;

}

}

}