

**Packet.cs:**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

namespace Packets

{

    public class Packet

    {

        protected const byte PCK_START1 = (byte)0x52;

        protected const byte PCK_START2 = (byte)0x65;

        protected ushort PCK_SIZE;

        protected ushort PCK_CNT;

        protected byte PCK_ID;

        protected ushort PCK_CRC;

        protected const byte PCK_END1 = (byte)0x4A;

        protected const byte PCK_END2 = (byte)0x6F;

        protected byte PCK_SID;

        protected ushort PCK_SCNT;

        private byte[] packet;

        private int idx;

        public Packet()

        {
```

```
this.PCK_SIZE = (ushort)0x0000;
this.PCK_CNT = (ushort)0x0000;
this.PCK_ID = (byte)0x00;
this.PCK_CRC = (ushort)0x0000;
this.PCK_SID = (byte)0x00;
this.PCK_SCNT = (ushort)0x0000;
this.idx = 0;
}
```

```
#region PACKETU GENERATORIAI
```

```
#region TCP CLIENT packetai
```

```
public void init_CLIENT_P1(ushort cnt)
{
    const ushort pck_size = (ushort)0x0008;
    const byte pck_id = (byte)0x01;

    initPacket(pck_size + 4);
    addByte((byte)PCK_START1);
    addByte((byte)PCK_START2);
    addWord((ushort)pck_size);
    addWord((ushort)pck_id);
    addWord((ushort)cnt);
    addWord((ushort)0x0000);//crc
    updateCRC();
    addByte((byte)PCK_END1);
    addByte((byte)PCK_END2);
}
```

```

public void init_CLIENT_P2(ushort cnt, byte data)
{
    const ushort pck_size = (ushort)0x0009; //6 + 1
    const byte pck_id = (byte)0x02;

    initPacket(pck_size + 4);
    addByte((byte)PCK_START1);
    addByte((byte)PCK_START2);
    addWord((ushort)pck_size);
    addWord((ushort)pck_id);
    addWord((ushort)cnt);
    addByte((byte)data);
    addWord((ushort)0x0000); //crc
    addByte((byte)PCK_END1);
    addByte((byte)PCK_END2);
    updateCRC();
}

public void init_CLIENT_P3(ushort cnt, byte[] more_data)
{
    ushort pck_size = (ushort)(8 + more_data.Length);
    const byte pck_id = (byte)0x03;

    initPacket(pck_size + 4);
    addByte((byte)PCK_START1);
    addByte((byte)PCK_START2);
    addWord((ushort)pck_size);
    addWord((ushort)pck_id);

```

```

    addWord((ushort)cnt);
    for (int i = 0; i < more_data.Length; i++)
    {
        addByte((byte)more_data[i]);
    }
    addWord((ushort)0x0000);//crc
    addByte((byte)PCK_END1);
    addByte((byte)PCK_END2);
    updateCRC();
}

```

#endregion

#region TCP SERVER packetai

```

public void init_SERVER_P1(ushort cnt, byte pck_sid, ushort pck_scnt)
{
    const ushort pck_size = (ushort)0x000C;
    const byte pck_id = (byte)0x01;

    initPacket(pck_size + 4);
    addByte((byte)PCK_START1);
    addByte((byte)PCK_START2);
    addWord((ushort)pck_size);
    addWord((ushort)pck_id);
    addWord((ushort)cnt);
    addWord((ushort)pck_sid);
    addWord((ushort)pck_scnt);
}

```

```

        addWord((ushort)0x0000);//crc
        addByte((byte)PCK_END1);
        addByte((byte)PCK_END2);
        updateCRC();
    }

```

```

public void init_SERVER_P2(ushort cnt, byte pck_sid, ushort pck_scnt, byte data)
{
    const ushort pck_size = (ushort)0x000D; //9 + 1
    const byte pck_id = (byte)0x02;

    initPacket(pck_size + 4);
    addByte((byte)PCK_START1);
    addByte((byte)PCK_START2);
    addWord((ushort)pck_size);
    addWord((ushort)pck_id);
    addWord((ushort)cnt);
    addByte((byte)pck_sid);
    addWord((ushort)pck_scnt);
    addByte((byte)data);
    addWord((ushort)0x0000);//crc
    addByte((byte)PCK_END1);
    addByte((byte)PCK_END2);
    updateCRC();
}

```

```

public void init_SERVER_P3(ushort cnt, byte pck_sid, ushort pck_scnt, byte[] more_data)
{
    ushort pck_size = (ushort)(12 + more_data.Length);

```

```

const byte pck_id = (byte)0x03;

initPacket(pck_size + 4);
addByte((byte)PCK_START1);
addByte((byte)PCK_START2);
addWord((ushort)pck_size);
addWord((ushort)pck_id);
addWord((ushort)cnt);
addByte((byte)pck_sid);
addWord((ushort)pck_scnt);
for (int i = 0; i < more_data.Length; i++)
{
    addByte((byte)more_data[i]);
}
addWord((ushort)0x0000);//crc
addByte((byte)PCK_END1);
addByte((byte)PCK_END2);
updateCRC();
}

```

```

#endregion

```

```

#endregion

```

```

#region PACKET FIELD ACCESSORS

```

```

#region PCK_SIZE

```

```
public void setPCK_SIZE(ushort pck_size)
{
    addWordAt(pck_size, 2);
    updateCRC();
}
```

```
public ushort getPCK_SIZE()
{
    ushort data = bytes2word(this.packet[2], this.packet[3]);
    return data;
}
```

```
public string getPCK_SIZE_HexString()
{
    ushort data = bytes2word(this.packet[2], this.packet[3]);
    return word2hexstr(data);
}
```

```
#endregion
```

```
#region PCK_CNT
```

```
public void setPCK_CNT(ushort pck_cnt)
{
    addWordAt(pck_cnt, 4);
    updateCRC();
}
```

```
public ushort getPCK_CNT()
```

```
{  
    ushort data = bytes2word(this.packet[4], this.packet[5]);  
    return data;  
}
```

```
public string getPCK_CNT_HexString()  
{  
    ushort data = bytes2word(this.packet[4], this.packet[5]);  
    return word2hexstr(data);  
}
```

```
#endregion
```

```
#region PCK_ID
```

```
public byte getPCK_ID()  
{  
    byte data = this.packet[6];  
    return data;  
}
```

```
public string getPCK_ID_HexString()  
{  
    byte data = this.packet[6];  
    return byte2hexstr(data);  
}
```

```
#endregion
```



```
#region PCK_CRC
```

```
public ushort getPCK_CRC()  
{  
    int crc_index = this.packet.Length - 3;  
    ushort data = bytes2word(this.packet[crc_index], this.packet[crc_index+1]);  
    return data;  
}
```

```
public string getPCK_CRC_HexString()  
{  
    int crc_index = this.packet.Length - 3;  
    ushort data = bytes2word(this.packet[crc_index], this.packet[crc_index + 1]);  
    return word2hexstr(data);  
}
```

```
#endregion
```

```
/*unique to SERVER packets*/
```

```
#region PCK_SID
```

```
public void setPCK_SID(byte pck_sid)  
{  
    addByteAt(pck_sid, 7);  
    updateCRC();  
}
```

```
public byte getPCK_SID()
```

```
{
```

```
    byte data = this.packet[7];
```

```
    return data;
```

```
}
```

```
public string getPCK_SID_HexString()
```

```
{
```

```
    byte data = this.packet[7];
```

```
    return byte2hexstr(data);
```

```
}
```

```
#endregion
```

```
#region PCK_SCNT
```

```
public void setPCK_SCNT(ushort pck_scnt)
```

```
{
```

```
    addWordAt(pck_scnt, 8);
```

```
    updateCRC();
```

```
}
```

```
public ushort getPCK_SCNT()
```

```
{
```

```
    ushort data = bytes2word(this.packet[8], this.packet[9]);
```

```
    return data;
```

```
}
```

```
public string getPCK_SCNT_HexString()
{
    ushort data = bytes2word(this.packet[8], this.packet[9]);
    return word2hexstr(data);
}
```

```
#endregion
```

```
#endregion
```

```
#region PACKET TOOLS
```

```
protected void initPacket(int size)
{
    this.packet = new byte[size];
}
```

```
protected void addByte(byte data)
{
    this.packet[idx++] = data;
}
```

```
protected void addWord(ushort data)
{
    byte hbyte = (byte)0x00;
    byte lbyte = (byte)0x00;
    word2bytes(data, ref hbyte, ref lbyte);
    this.packet[idx++] = hbyte;
```

```
    this.packet[idx++] = lbyte;
}
```

```
protected void addByteAt(byte data, int index)
{
    this.packet[index] = data;
}
```

```
protected void addWordAt(ushort data, int index)
{
    byte hbyte = (byte)0x00;
    byte lbyte = (byte)0x00;
    word2bytes(data, ref hbyte, ref lbyte);
    this.packet[index] = hbyte;
    this.packet[index + 1] = lbyte;
}
```

```
private ushort calcCRC()
{
    ushort crc = (ushort)0xffff;
    ushort index;
    byte b;

    for (index = 4; index < this.packet.Length - 3; index++)
    {
        crc ^= ((ushort)((this.packet[index] << 8) & 0x0000ffff));
        for (b = 0; b < 8; b++)
        {
            if ((crc & (ushort)0x8000) == (ushort)0x8000)
```

```

        crc = (ushort)((ushort)((crc << 1) & 0x0000ffff) ^ (ushort)0x1021);
    else
        crc = (ushort)((crc << 1) & 0x0000ffff);
    }
}

return crc;
}

```

```

protected void updateCRC()
{
    ushort crc = calcCRC();
    addWordAt((ushort)crc, this.packet.Length - 4);
}

```

```

public string toHexString()
{
    return BitConverter.ToString(this.packet);
}

```

```

public void setPacket(byte[] p)
{
    this.packet = p;
}

```

```

public byte[] getRawPacket()
{
    return this.packet;
}

```

```
#endregion
```

```
#region BYTE TOOLS
```

```
private ushort bytes2word(byte hb, byte lb)
```

```
{  
    ushort data = (ushort)(hb << 8 | lb);  
    return data;  
}
```

```
private void word2bytes(ushort data, ref byte hb, ref byte lb)
```

```
{  
    hb = (byte)((data >> 8) & 0x000000FF);  
    lb = (byte)(data & (ushort)0x00FF);  
}
```

```
private string byte2hexstr(byte data)
```

```
{  
    StringBuilder sb = new StringBuilder(4);  
    sb.Append("0x");  
    sb.AppendFormat("{0:x2}", data);  
    return sb.ToString();  
}
```

```
private string word2hexstr(ushort data)
```

```
{  
    StringBuilder sb = new StringBuilder(6);  
    byte hb = (byte)((data >> 8) & 0x000000FF);
```

```

        byte lb = (byte)(data & 0x00FF);
        sb.Append("0x");
        sb.AppendFormat("{0:X2}", hb);
        sb.AppendFormat("{0:X2}", lb);
        return sb.ToString();
    }

```

```

#endregion

```

```

    }
}

```

#### **SocketClient application code:**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Packets;

```

```

namespace MySocketClient
{
    public partial class Form1 : Form

```

```

{
    private ushort packet_cnt;

    public delegate void UpdateTextBox1Callback(string strMessage);
    public delegate void UpdateTextBox2Callback(string strMessage);

    public Form1()
    {
        InitializeComponent();
        packet_cnt = 1;
    }

    public void updateTextBox1(string data)
    {
        this.textBox1.AppendText(data + "\r\n");
    }

    public void updateTextBox2(string data)
    {
        this.textBox2.AppendText(data + "\r\n");
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Packet cp = new Packet();
        cp.init_CLIENT_P1(packet_cnt++);

        this.textBox1.Text = "Packet: " + cp.toHexString() + "\r\n";
        this.textBox1.Text += "-----" + "\r\n";
    }
}

```



```

        this.textBox1.Text += "PCK_SIZE = " + cp.getPCK_SIZE() + " (" + cp.getPCK_SIZE_HexString() + ")" +
"\r\n";

        this.textBox1.Text += "PCK_CNT = " + cp.getPCK_CNT() + " (" + cp.getPCK_CNT_HexString() + ")" +
"\r\n";

        this.textBox1.Text += "PCK_ID = " + cp.getPCK_ID() + " (" + cp.getPCK_ID_HexString() + ")" +
"\r\n";

        this.textBox1.Text += "PCK_CRC = " + cp.getPCK_CRC_HexString() + "\r\n";

        this.textBox1.Text += "-----" + "\r\n";

```

```

SendThread st = new SendThread(cp.getRawPacket(), this, this.textBoxIP.Text);

```

```

Thread t = new Thread(new ThreadStart(st.ThreadProc));
t.Start();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    Packet cp = new Packet();
    cp.init_CLIENT_P2(packet_cnt++, (byte)0x58);

    this.textBox1.Text = "Packet: " + cp.toHexString() + "\r\n";

    this.textBox1.Text += "-----" + "\r\n";

    this.textBox1.Text += "PCK_SIZE = " + cp.getPCK_SIZE() + " (" + cp.getPCK_SIZE_HexString() + ")" +
"\r\n";

    this.textBox1.Text += "PCK_CNT = " + cp.getPCK_CNT() + " (" + cp.getPCK_CNT_HexString() + ")" +
"\r\n";

    this.textBox1.Text += "PCK_ID = " + cp.getPCK_ID() + " (" + cp.getPCK_ID_HexString() + ")" +
"\r\n";

    this.textBox1.Text += "PCK_CRC = " + cp.getPCK_CRC_HexString() + "\r\n";
}

```

```

this.textBox1.Text += "-----" + "\r\n";

SendThread st = new SendThread(cp.getRawPacket(), this, this.textBoxIP.Text);

Thread t = new Thread(new ThreadStart(st.ThreadProc));
t.Start();
}

private void button3_Click(object sender, EventArgs e)
{
    Packet cp = new Packet();
    byte[] more_data = new byte[4] { 0x58, 0x74, 0x4f, 0xfa };
    cp.init_CLIENT_P3(packet_cnt++, more_data);

    this.textBox1.Text = "Packet: " + cp.toHexString() + "\r\n";
    this.textBox1.Text += "-----" + "\r\n";
    this.textBox1.Text += "PCK_SIZE = " + cp.getPCK_SIZE() + " (" + cp.getPCK_SIZE_HexString() + ")" +
"\r\n";
    this.textBox1.Text += "PCK_CNT = " + cp.getPCK_CNT() + " (" + cp.getPCK_CNT_HexString() + ")" +
"\r\n";
    this.textBox1.Text += "PCK_ID = " + cp.getPCK_ID() + " (" + cp.getPCK_ID_HexString() + ")" +
"\r\n";
    this.textBox1.Text += "PCK_CRC = " + cp.getPCK_CRC_HexString() + "\r\n";
    this.textBox1.Text += "-----" + "\r\n";

    SendThread st = new SendThread(cp.getRawPacket(), this, this.textBoxIP.Text);

    Thread t = new Thread(new ThreadStart(st.ThreadProc));

```

```
t.Start();  
}
```

```
public class SendThread  
{  
    private byte[] pck_data;  
    private Form1 form;  
    enum RX_STATE { WAIT_FOR_SYNC, RX_LENGTH, RX_DATA, RX_CRC, RX_END }  
    private string server_ip;  
  
    public SendThread(byte[] d, Form1 f, string ip)  
    {  
        pck_data = d;  
        form = f;  
        server_ip = ip;  
    }  
  
    public void ThreadProc()  
    {  
        Form1.UpdateTextBox1Callback tb1 = new  
Form1.UpdateTextBox1Callback(form.updateTextBox1);  
  
        Form1.UpdateTextBox2Callback tb2 = new  
Form1.UpdateTextBox2Callback(form.updateTextBox2);  
  
        form.Invoke(tb1, new object[] { "sendPacketToServer() started" });  
  
        TcpClient client = new TcpClient();
```

```

try
{

    form.Invoke(tb1, new object[] { "Connecting..." });

    client.Connect(server_ip, 7777);

    Socket soc = client.Client;
    soc.SendTimeout = 10000;
    soc.ReceiveTimeout = 10000;
    soc.NoDelay = true;

    form.Invoke(tb1, new object[] { "Connected" });

    soc.Send(pck_data, pck_data.Length, 0);

    form.Invoke(tb1, new object[] { "Data sent, waiting for response..." });

    byte b = (byte)0x00;
    byte[] data = new byte[1];
    int bytes = 0;
    int counter = 0;
    ushort sizeOfReceivingFrame = (ushort)0x0000;
    ushort rxCRC = (ushort)0x0000;
    ushort crc = (ushort)0x0000;
    byte crc_hb = (byte)0x00;
    byte crc_lb = (byte)0x00;
    bool pck_error = false;
    bool crc_error = false;

```

```

bool end_error = false;

bool rx_len_error = false;

bool rxOK = false;

byte[] rxPacket = new byte[512]; //cia tik atvaizdavimui reikalinga!

int idx = 0;


RX_STATE rxState = RX_STATE.WAIT_FOR_SYNC;


while (pck_error == false && rxOK == false && ((bytes = soc.Receive(data, 1, 0)) > 0))
{
    //form.Invoke(tb1, new object[] { "byte received, bytes_size=" + bytes });

    b = (byte)data[0];

    switch (rxState)
    {
        case RX_STATE.WAIT_FOR_SYNC:
            if (b != (byte)0x21)
            {
                counter = 0;

                break;
            }

            counter++;

            rxPacket[idx++] = b;

            if (counter == 2)
            {
                counter = 0;

                rxState = RX_STATE.RX_LENGTH;
            }

```

```

        break;
    case RX_STATE.RX_LENGTH:
        counter++;

        if (counter == 1)
        {
            sizeOfReceivingFrame = b;
            rxPacket[idx++] = b;
            break;
        }

        if (counter == 2)
        {
            counter = 0;
            sizeOfReceivingFrame = (ushort)((sizeOfReceivingFrame << 8) | b);
            sizeOfReceivingFrame -= 3; // atmetam CRC ir END
            if (sizeOfReceivingFrame <= 0)
            {
                pck_error = true;
                rx_len_error = true;
                form.Invoke(tb2, new object[] { "RX_STATE.RX_LENGTH: pck_error" });
                break;
            }
            rxPacket[idx++] = b;
            rxState = RX_STATE.RX_DATA;
            break;
        }
        break;
    case RX_STATE.RX_DATA:

```

```

    rxPacket[idx++] = b;

    counter++;

    if (counter == sizeofReceivingFrame)
    {
        counter = 0;

        rxState = RX_STATE.RX_CRC;

        break;
    }

    break;

case RX_STATE.RX_CRC:

    counter++;

    if (counter == 1)
    {
        crc_hb = b;

        rxPacket[idx++] = b;

        break;
    }

    if (counter == 2)
    {
        crc_lb = b;

        rxPacket[idx++] = b;

        counter = 0;

        crc = (ushort)((crc_hb << 8) | crc_lb);

        rxCRC = calcCRC(rxPacket, idx);

        if (crc != rxCRC)

```

```

        {
            form.Invoke(tb2, new object[] { "BAD CRC: calcCRC=" + word2hexstr(rxCRC) + " |
crc=" + word2hexstr(rxCRC) });

            form.Invoke(tb2, new object[] { "packet: " + toHexString(rxPacket) });

            pck_error = true;

            crc_error = true;
        }
    else
    {
        rxState = RX_STATE.RX_END;
    }

    break;
}

break;
case RX_STATE.RX_END:
    counter = 0;
    if (b != (byte)0x2E) // discard
    {
        form.Invoke(tb2, new object[] { "MISSING END! ..." });

        form.Invoke(tb2, new object[] { "packet discarded: " + toHexString(rxPacket) });

        pck_error = true;

        end_error = true;
    }
    else
    {
        rxPacket[idx++] = b;

        rxOK = true;
    }
}

```



```

        break;
    }
}

if (rxOK == true)
{
    byte[] p = new byte[idx];
    Array.Copy(rxPacket, p, idx);
    Packet server_packet = new Packet();
    server_packet.setPacket(p);
    form.Invoke(tb2, new object[] { "OK: packet=" + server_packet.toHexString() });
}
else
{
    form.Invoke(tb2, new object[] { "RX failed" });
}

client.Close();

form.Invoke(tb1, new object[] { "Closed" });

}
catch (Exception ex)
{
    form.Invoke(tb2, new object[] { ex.Message });
    client.Close();
}

```

```
}
```

```
private ushort calcCRC(byte[] packet, int length)
```

```
{
```

```
    ushort crc = (ushort)0xffff;
```

```
    ushort index;
```

```
    byte b;
```

```
    for (index = 4; index < length - 2; index++)//~2 : atmetam CRC kuri gavom ir idejom i packet
```

```
    {
```

```
        crc ^= ((ushort)((packet[index] << 8) & 0x0000ffff));
```

```
        for (b = 0; b < 8; b++)
```

```
        {
```

```
            if ((crc & (ushort)0x8000) == (ushort)0x8000)
```

```
                crc = (ushort)((ushort)((crc << 1) & 0x0000ffff) ^ (ushort)0x1021);
```

```
            else
```

```
                crc = (ushort)((crc << 1) & 0x0000ffff);
```

```
        }
```

```
    }
```

```
    return crc;
```

```
}
```

```
public string toHexString(byte[] packet)
```

```
{
```

```
    return BitConverter.ToString(packet);
```

```
}
```

```

private string word2hexstr(ushort data)
{
    StringBuilder sb = new StringBuilder(6);
    byte hb = (byte)((data >> 8) & 0x000000FF);
    byte lb = (byte)(data & 0x00FF);
    sb.Append("0x");
    sb.AppendFormat("{0:X2}", hb);
    sb.AppendFormat("{0:X2}", lb);
    return sb.ToString();
}
}
}
}

```

**SocketServer application code:**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Packets;

namespace MySocketServer
{

```

```

public partial class Form1 : Form
{
    private ushort packet_cnt;

    TcpListener server = null;

    Int32 port = 7777;

    Boolean srvRunning = false;

    private delegate void UpdateTextBox1Callback(string strMessage);
    private delegate void UpdateTextBox2Callback(string strMessage);
    private Thread listenThread;

    enum RX_STATE { WAIT_FOR_SYNC, RX_LENGTH, RX_DATA, RX_CRC, RX_END };

    public Form1()
    {
        InitializeComponent();
        packet_cnt = 1;
    }

    private void updateTextBox1(string data)
    {
        this.textBox1.Text += data + "\r\n";
    }

    private void updateTextBox2(string data)
    {
        this.textBox2.Text += data + "\r\n";
    }
}

```

```

private void button1_Click(object sender, EventArgs e)
{

    try
    {
        server = new TcpListener(IPAddress.Parse(this.textBoxIP.Text), port);

        server.Start();

        srvRunning = true;
        this.listenThread = new Thread(accept_n_process);
        this.listenThread.Start();

        this.button1.Enabled = false;
        this.button2.Enabled = true;
    }
    catch (SocketException ex)
    {
        this.textBox1.Text = "SocketException: " + ex;
    }
    catch (Exception exx)
    {
        this.textBox1.Text = "Exception: " + exx;
    }
}

private void accept_n_process()
{
    UpdateTextBox1Callback tb1 = new UpdateTextBox1Callback(updateTextBox1);

```

```
UpdateTextBox2Callback tb2 = new UpdateTextBox2Callback(updateTextBox2);
```

```
this.Invoke(tb1, new object[] { "accept_n_process started" });
```

```
this.Invoke(tb2, new object[] { "accept_n_process started" });
```

```
TcpClient client = null;
```

```
try
```

```
{
```

```
    while (srvRunning == true)
```

```
    {
```

```
        client = server.AcceptTcpClient();
```

```
        Socket soc = client.Client;
```

```
        soc.SendTimeout = 10000;
```

```
        soc.ReceiveTimeout = 10000;
```

```
        soc.NoDelay = true;
```

```
        byte b = (byte)0x00;
```

```
        byte[] data = new byte[1];
```

```
        int bytes = 0;
```

```
        int counter = 0;
```

```
        ushort sizeOfReceivingFrame = (ushort)0x0000;
```

```
        ushort rxCRC = (ushort)0x0000;
```

```
        ushort crc = (ushort)0x0000;
```

```
        byte crc_hb = (byte)0x00;
```

```
        byte crc_lb = (byte)0x00;
```

```
        bool pck_error = false;
```

```

bool rxOK = false;

bool crc_error = false;

bool end_error = false;

bool rx_len_error = false;

byte[] rxPacket = new byte[512]; //cia tik atvaizdavimui reikalinga!

int idx = 0;


RX_STATE rxState = RX_STATE.WAIT_FOR_SYNC;


while ((pck_error == false && rxOK == false && (bytes = soc.Receive(data, 1, 0)) > 0))
{
    //this.Invoke(tb1, new object[] { "byte received, bytes_size=" + bytes });


    b = (byte)data[0];

    switch (rxState)
    {
        case RX_STATE.WAIT_FOR_SYNC:
            if (b != (byte)0x21)
            {
                counter = 0;

                break;
            }


            counter++;

            rxPacket[idx++] = b;


            if (counter == 2)
            {
                counter = 0;
            }
        }
    }
}

```

```

        rxState = RX_STATE.RX_LENGTH;
    }
    break;
case RX_STATE.RX_LENGTH:
    counter++;

    if (counter == 1)
    {
        sizeOfReceivingFrame = b;
        rxPacket[idx++] = b;
        break;
    }

    if (counter == 2)
    {
        counter = 0;
        sizeOfReceivingFrame = (ushort)((sizeOfReceivingFrame << 8) | b);
        sizeOfReceivingFrame -= 3; // atmetam CRC ir END
        if (sizeOfReceivingFrame <= 0)
        {
            pck_error = true;
            rx_len_error = true;
            this.Invoke(tb1, new object[] { "RX_STATE.RX_LENGTH: pck_error" });
            break;
        }
        rxPacket[idx++] = b;
        rxState = RX_STATE.RX_DATA;
        break;
    }
}

```



```

        break;
    case RX_STATE.RX_DATA:
        rxPacket[idx++] = b;
        counter++;
        if (counter == sizeofReceivingFrame)
        {
            counter = 0;
            rxState = RX_STATE.RX_CRC;
            break;
        }
        break;
    case RX_STATE.RX_CRC:
        counter++;
        if (counter == 1)
        {
            crc_hb = b;
            rxPacket[idx++] = b;
            break;
        }

        if (counter == 2)
        {
            crc_lb = b;
            rxPacket[idx++] = b;
            counter = 0;

            crc = (ushort)((crc_hb << 8) | crc_lb);

            rxCRC = calcCRC(rxPacket, idx);

```

```

        if (crc != rxCRC)
        {
            this.Invoke(tb1, new object[] { "BAD CRC: calcCRC=" + word2hexstr(rxCRC) + " | " +
            crc=" + word2hexstr(rxCRC) });

            this.Invoke(tb1, new object[] { "packet: " + toHexString(rxPacket) });

            pck_error = true;

            crc_error = true;
        }
        else
        {
            rxState = RX_STATE.RX_END;
        }

        break;
    }

    break;
case RX_STATE.RX_END:
    counter = 0;
    if (b != (byte)0x2E) // discard
    {
        this.Invoke(tb1, new object[] { "MISSING END! ..." });

        this.Invoke(tb1, new object[] { "packet discarded: " + toHexString(rxPacket) });

        pck_error = true;

        end_error = true;
    }
    else
    {
        //this.Invoke(tb1, new object[] { "rxOK! ..." });

```

```

        rxPacket[idx++] = b;

        rxOK = true;
    }

    //

    break;
}
}

if (rxOK == true) // response
{
    byte[] p = new byte[idx];
    Array.Copy(rxPacket, p, idx);
    Packet client_packet = new Packet();
    client_packet.setPacket(p);
    this.Invoke(tb1, new object[] { "OK: packet=" + client_packet.toHexString() });

    Packet server_packet = processAnswer(client_packet);

    if (server_packet != null)
    {
        soc.Send(server_packet.getRawPacket(), server_packet.getRawPacket().Length, 0);
        this.Invoke(tb2, new object[] { "packet=" + server_packet.toHexString() });
    }
    else
    {
        this.Invoke(tb2, new object[] { "failed at processAnswer(): server_packet = null" });
    }
}
else

```

```

        {
            this.Invoke(tb1, new object[] { "RX failed" });
        }

        client.Close();
        this.Invoke(tb1, new object[] { "Connection closed" });
    }
}

catch (SocketException ex)
{
    this.Invoke(tb1, new object[] { ex.Message });
    this.Invoke(tb2, new object[] { ex.Message });
    client.Close();
    this.Invoke(tb1, new object[] { "Connection closed after SocketException" });
}

catch (Exception exx)
{
    this.Invoke(tb1, new object[] { exx.Message });
    this.Invoke(tb2, new object[] { exx.Message });
    client.Close();
    this.Invoke(tb1, new object[] { "Connection closed after Exception" });
}

}

private Packet processAnswer(Packet client_packet)
{
    UpdateTextBox2Callback tb2 = new UpdateTextBox2Callback(updateTextBox2);
    byte pck_id = client_packet.getPCK_ID();

```

```

switch (pck_id)
{
    case 0x01:
        Packet sp1 = new Packet();
        sp1.init_SERVER_P1(packet_cnt++, client_packet.getPCK_ID(), client_packet.getPCK_CNT());
        return sp1;
    case 0x03:
        Packet sp2 = new Packet();
        sp2.init_SERVER_P2(packet_cnt++, client_packet.getPCK_ID(), client_packet.getPCK_CNT(),
(byte)0x58);
        return sp2;
    case 0x05:
        Packet sp3 = new Packet();
        byte[] more_data = new byte[4] { 0x58, 0x74, 0x4f, 0xfa };
        sp3.init_SERVER_P3(packet_cnt++, client_packet.getPCK_ID(), client_packet.getPCK_CNT(),
more_data);
        return sp3;
    default:
        this.Invoke(tb2, new object[] { "UNKNOWN CLIENT PACKET ID" });
        break;
}

return null;
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    this.srvRunning = false;
}

```

```

server.Stop();

this.button1.Enabled = true;

this.button2.Enabled = false;
}

```

```

private ushort calcCRC(byte[] packet, int length)

```

```

{
    ushort crc = (ushort)0xffff;
    ushort index;
    byte b;

```

```

    for (index = 4; index < length-2; index++)// -2 : atmetam CRC kuri gavom ir idejom i packet

```

```

    {
        crc ^= ((ushort)((packet[index] << 8) & 0x0000ffff));
        for (b = 0; b < 8; b++)
        {
            if ((crc & (ushort)0x8000) == (ushort)0x8000)
                crc = (ushort)((ushort)((crc << 1) & 0x0000ffff) ^ (ushort)0x1021);
            else
                crc = (ushort)((crc << 1) & 0x0000ffff);
        }
    }
}

```

```

    return crc;
}

```

```

public string toHexString(byte[] packet)

```

```

{
    return BitConverter.ToString(packet);
}

```

```
}
```

```
private string word2hexstr(ushort data)
```

```
{
```

```
    StringBuilder sb = new StringBuilder(6);
```

```
    byte hb = (byte)((data >> 8) & 0x000000FF);
```

```
    byte lb = (byte)(data & 0x00FF);
```

```
    sb.Append("0x");
```

```
    sb.AppendFormat("{0:X2}", hb);
```

```
    sb.AppendFormat("{0:X2}", lb);
```

```
    return sb.ToString();
```

```
}
```

```
private ushort bytes2word(byte hb, byte lb)
```

```
{
```

```
    ushort data = (ushort)(hb << 8 | lb);
```

```
    return data;
```

```
}
```

```
}
```

```
}
```