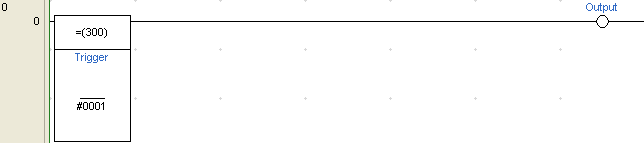
C# – Send FINS command from C# to PLC

**c++finsplcudp**

I try to send **FINS** commands through UDP from PC (C#.NET) to PLC Ethernet module (Omron), but is not getting any responds from the PLC and have no clue where I can start troubleshooting.

PLC has a very simple ladder logic as follow:  
If DM100 has value of #0001, then trigger on output 101.00. (Here, "Trigger" is just a symbol name for memory area D100, and "Output" is a symbol for output 101.00)  


Then I wrote a piece of C# that performs FINS command of "Memory Area Write" which has command code of 01 02, followed by beginning address, number of items to be written, and the data. The C# code should write a value of #0001 to PLC's D100 area to trigger ON on 101.00.

*[deleted code that doesn't work]..*

The output 101.00 did not get triggered, nor I receive any exception.  
I have made sure the following:

1. Port, node and address configured correctly as confirmed by "Work Online" in CX-Programmer. I have also ping each IP to make sure nodes are connected.
2. The UdpClient code is valid since I wrote a very simple server / client code that successfully send and receive packets.
3. Ladder logic has no problem. I transfered the ladder to PLC and test out by Work Online in Monitor mode and setting D100 a value manually.

I suspect there is mistake in the fins\_cmnd array, but as seen in my code, I have commented as detail as possible on each value; I can't possibly find myself missing anything. I suspect I may not be parsing the hexadecimal correctly, but again, I have no exception to guide me.

I have no idea where and how I can troubleshoot. Hope someone here with FINS programming or PLC experience can offer me some help.

**[ANSWER]**  
Thanks **Porge** for the link – that got me found out the problem. After a couple trails finally get it to work. See below for the working code.

string SERV\_IP\_ADDR = "192.168.250.1";

const int FINS\_UDP\_PORT = 9600;

byte[] sendPacket = new byte[]

{

// Full UDP packet: 80 00 02 00 00 00 00 05 00 19 01 02 82 00 64 00 00 01 00 01

// Header

0x80, //0.(ICF) Display frame information: 1000 0001

0x00, //1.(RSV) Reserved by system: (hex)00

0x02, //2.(GCT) Permissible number of gateways: (hex)02

0x00, //3.(DNA) Destination network address: (hex)00, local network

0x00, //4.(DA1) Destination node address: (hex)00, local PLC unit

0x00, //5.(DA2) Destination unit address: (hex)00, PLC

0x00, //6.(SNA) Source network address: (hex)00, local network

0x05, //7.(SA1) Source node address: (hex)05, PC's IP is 192.168.250.5

0x00, //8.(SA2) Source unit address: (hex)00, PC only has one ethernet

0x19, //9.(SID) Service ID: just give a random number 19

// Command

0x01, //10.(MRC) Main request code: 01, memory area write

0x02, //11.(SRC) Sub-request code: 02, memory area write

// PLC Memory Area

0x82, //12.Memory area code (1 byte): 82(DM)

// Address information

0x00, //13.Write start address (2 bytes): D100

0x64,

0x00, //15.Bit address (1 byte): Default 0

0x00, //16.No. of items (2 bytes): only one address which is D100

0x01,

// Write Data

0x00, //18.Data to write (2 bytes): value is 1

0x01,

};

UdpClient client = new UdpClient(); //create a UdpClient instance

try

{

client.Send(sendPacket, sendPacket.Length, SERV\_IP\_ADDR, FINS\_UDP\_PORT);

}

catch (SocketException se)

{

Console.WriteLine(se.ErrorCode + ": " + se.Message);

}

client.Close();

Best Solution

None of those strings will be parsed as hexadecimal. NumberStyles.AllowHexSpecifier *allows* the hexadecimal prefix "0x" but doesn't parse the number as hex unless it is present.

So you'd want (char)Int16.Parse("0x64", NumberStyles.AllowHexSpecifier).

However, numeric literals in C# can be hexadecimal, so instead of doing all that you can just write 0x64 instead.

I've just looked at [this page as a reference](http://deltamotion.com/support/webhelp/rmctools/Communications/Ethernet/Supported_Protocols/Ethernet_FINS_UDP.htm) for the protocol, and it would also be better to create the message directly as bytes rather than specifying Unicode code points and decoding those to bytes as ASCII. You can also use the array specification syntax to remove a lot of clutter:

var message = new byte[]

{

// header

0x80, //(ICF) Display frame information: 1000 0001

0x00, //(RSV) Reserved by system: (hex)00

0x02, //(GCT) Permissible number of gateways: (hex)02

0x00, //(DNA) Destination network address: (hex)00, local network

0x00, //(DA1) Destination node address: (hex)00, local PLC unit

0x00, //(DA2) Destination unit address: (hex)00, PLC

0x00, //(SNA) Source network address: (hex)00, local network

0x05, //(SA1) Source node address: (hex)05, PC's IP is 192.168.250.5

0x00, //(SA2) Source unit address: (hex)00, PC only has one ethernet

0x19, //(SID) Service ID: just give a random number 19

// command

0x01, //(MRC) Main request code: 01, memory area write

0x02, //(SRC) Sub-request code: 02, memory area write

// data

0x82, //Memory area code, 2 bytes: 82(DM)

0x00, //Write start address DM100

0x64,

0x00,

0x00, //Word write: only one address

0x01,

0x00, //Write value of 1 to address DM100 (0000 0000 0000 0001)

0x01, // - this value is 0xaabbccdd -> cc dd aa bb

0x00,

0x00,

};

It also looks like there are some problems with your data section - according to the linked document the memory address should be 4 bytes, the write length 2 bytes, and the values 4 bytes each. These don't match with what you have, so I've expanded that section.

Related Solutions

[C# – How to update the GUI from another thread](https://itecnote.com/tecnote/c-how-to-update-the-gui-from-another-thread/)

The **simplest** way is an anonymous method passed into [Label.Invoke](https://msdn.microsoft.com/en-us/library/zyzhdc6b(v=vs.110).aspx):

// Running on the worker thread

string newText = "abc";

form.Label.Invoke((MethodInvoker)delegate {

// Running on the UI thread

form.Label.Text = newText;

});

// Back on the worker thread

Notice that Invoke blocks execution until it completes--this is synchronous code. The question doesn't ask about asynchronous code, but there is lots of [content on Stack Overflow](https://stackoverflow.com/search?q=c%23+await+async+pattern) about writing asynchronous code when you want to learn about it.

[C# – Get int value from enum in C#](https://itecnote.com/tecnote/c-get-int-value-from-enum-in-c/)

Just cast the enum, e.g.

int something = (int) Question.Role;

The above will work for the vast majority of enums you see in the wild, as the default underlying type for an enum is int.

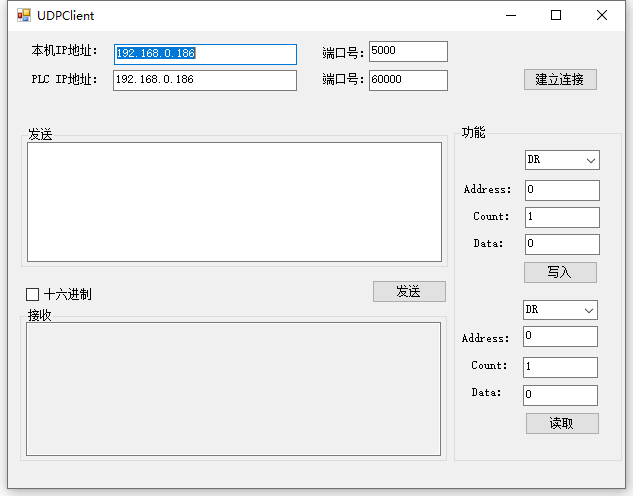
However, as [cecilphillip](https://stackoverflow.com/users/333082/cecilphillip) points out, enums can have different underlying types. If an enum is declared as a uint, long, or ulong, it should be cast to the type of the enum; e.g. for

enum StarsInMilkyWay:long {Sun = 1, V645Centauri = 2 .. Wolf424B = 2147483649};

you should use

long something = (long)StarsInMilkyWay.Wolf424B;

C# OMRON FINS UDP通信



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;

namespace WindowsFormsApplication2

{

public partial class Form1 : Form

{

UdpClient udpClient;

byte Registertype=0x82;

IPAddress locateIp;

IPEndPoint locatePoint;

IPAddress remoteIp;

IPEndPoint remotePoint;

public Form1()

{

InitializeComponent();

}

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

{

txtLocateIP.Text = getIPAddress();

txtRemoteIP.Text = getIPAddress();

*//禁止夸线程监视*

Control.CheckForIllegalCrossThreadCalls = false;*//这种方法不好，只是方便，不提倡*

}

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

{

if (udpClient != null)

return;

locateIp = IPAddress.Parse(txtLocateIP.Text);

locatePoint = new IPEndPoint(locateIp, Convert.ToInt32(txtLocatePort.Text));

udpClient = new UdpClient(locatePoint);

remoteIp = IPAddress.Parse(txtRemoteIP.Text);

remotePoint = new IPEndPoint(remoteIp, Convert.ToInt32(txtRemotePort.Text));

*//监听创建好后，就开始接收信息，并创建一个线程*

Thread th = new Thread(Receive);

th.IsBackground = true;

th.Start();

}

*/// <summary>*

*/// 接收线程方法*

*/// </summary>*

void Receive()

{

byte[] recBuffer;

*//远端IP*

IPEndPoint remotePoint = new IPEndPoint(IPAddress.Any, 0);

while (true)

{

try

{

recBuffer = udpClient.Receive(ref remotePoint);

if (cBoxHex.Checked)

{

rtxtRec.AppendText(StringtoHex(recBuffer) "\r\n");

}

else {

if (recBuffer != null)

{

byte[] Num = new byte[2] { recBuffer[recBuffer.Length-2], recBuffer[recBuffer.Length-1] };

string zas = StringtoHex(Num);

Int32 xx = Convert.ToInt32(zas, 16);

rtxtRec.AppendText(xx "\r\n");

}

}

}

catch (Exception e)

{

MessageBox.Show(e.ToString());

}

}

}

string StringtoHex(byte[] myBytes)

{

string result=null;

foreach (byte b in myBytes)

{

string val = b.ToString("X");

result = (val.Length == 2 ? val : "0" val);

}

return result;

}

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

{

if (string.IsNullOrEmpty(rtxtSend.Text.Trim()))

return;

byte[] buffer = System.Text.Encoding.UTF8.GetBytes(rtxtSend.Text.Trim());

udpClient.Send(buffer, buffer.Length, remotePoint);

}

*/// <summary>*

*/// 获取本地IP的方法*

*/// </summary>*

*/// <returns></returns>*

private string getIPAddress()

{

*//获取本地所有IP地址*

IPHostEntry ipe = Dns.GetHostEntry(Dns.GetHostName());

IPAddress[] ip = ipe.AddressList;

for (int i = 0; i < ip.Length; i )

{

if (ip[i].AddressFamily.ToString().Equals("InterNetwork"))

{

return ip[i].ToString();

}

}

return null;

}

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

{

byte waddressL = Convert.ToByte((Convert.ToUInt16(wtxtaddress.Text) % 256).ToString());

byte waddressH= Convert.ToByte(((Convert.ToUInt16(wtxtaddress.Text) /256)%256).ToString());

byte wCountL = Convert.ToByte((Convert.ToUInt16(wtxtCount.Text) % 256).ToString());

byte wCountH = Convert.ToByte(((Convert.ToUInt16(wtxtCount.Text) / 256) % 256).ToString());

byte wdataL = Convert.ToByte((Convert.ToUInt16(wdata.Text) % 256).ToString());

byte wdataH = Convert.ToByte(((Convert.ToUInt16(wdata.Text) / 256) % 256).ToString());

IPAddress remoteIp = IPAddress.Parse(txtRemoteIP.Text);

IPEndPoint remotePoint = new IPEndPoint(remoteIp, Convert.ToInt32(txtRemotePort.Text));

byte[] wbuffer = { 0x80, 0x00, 0x02, 0x00, remoteIp.GetAddressBytes()[3], 0x00, 0x00, locateIp.GetAddressBytes()[3], 0x00, 0x00, 0x01, 0x02, Registertype, waddressH, waddressL, 0x00, wCountH, wCountL, wdataH, wdataL };

udpClient.Send(wbuffer, wbuffer.Length, remotePoint);

rtxtSend.Text = Convert.ToString(BitConverter.ToString(wbuffer, 0));

}

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

{

switch (cnbRegistertype.Text)

{

case "DR": Registertype = 0x82; break;

case "WR": Registertype = 0xB1; break;

case "CIO": Registertype = 0xB0;break;

case "ER": Registertype = 0xA0; break;

case "TIM": Registertype = 0x89;break;

case "CNT": Registertype = 0x89;break;

default: Registertype = 0x82; break;

}

}

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

{

*// 800002000100002800000101820068000001*

byte RaddressL = Convert.ToByte((Convert.ToUInt16(Rtxtaddress.Text) % 256).ToString());

byte RaddressH = Convert.ToByte(((Convert.ToUInt16(Rtxtaddress.Text) / 256) % 256).ToString());

byte wCountL = Convert.ToByte((Convert.ToUInt16(RtxtCount.Text) % 256).ToString());

byte RCountH = Convert.ToByte(((Convert.ToUInt16(RtxtCount.Text) / 256) % 256).ToString());

*// byte RdataL = Convert.ToByte((Convert.ToUInt16(Rdata.Text) % 256).ToString());*

*// byte RdataH = Convert.ToByte(((Convert.ToUInt16(Rdata.Text) / 256) % 256).ToString());*

IPAddress remoteIp = IPAddress.Parse(txtRemoteIP.Text);

IPEndPoint remotePoint = new IPEndPoint(remoteIp, Convert.ToInt32(txtRemotePort.Text));

byte[] Rbuffer = { 0x80, 0x00, 0x02, 0x00, remoteIp.GetAddressBytes()[3], 0x00, 0x00, locateIp.GetAddressBytes()[3], 0x00, 0x00, 0x01, 0x01, Registertype, RaddressH, RaddressL, 0x00, RCountH, wCountL };

udpClient.Send(Rbuffer, Rbuffer.Length, remotePoint);

}

}

}

Here is the code to create some public strings and handle the time function every 100 msec.

Public Class Form1

Public TX As String

Public FCS As String

Public RXD As String

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

Timer1.Enabled = False

'Display current date and time

Label2.Text = System.DateTime.Now

'Open the serial port

If SerialPort1.IsOpen = False Then

SerialPort1.Open()

End If

Dim charreturn As Integer

'Check DM AREA DM0000 to DM0009 data update

TX = "@00RD00000010"

Call GetFCS()

Label25.Text = TX + FCS + "\*"

Call communicate()

SerialPort1.Close()

' Set information on the screen

Label26.Text = RXD

If RXD.Substring(5, 2) = "00" Then

Label4.Text = RXD.Substring(7, 4)

Label5.Text = RXD.Substring(11, 4)

Label7.Text = RXD.Substring(15, 4)

Label9.Text = RXD.Substring(19, 4)

Label11.Text = RXD.Substring(23, 4)

Label13.Text = RXD.Substring(27, 4)

Label15.Text = RXD.Substring(31, 4)

Label17.Text = RXD.Substring(35, 4)

Label19.Text = RXD.Substring(39, 4)

Label21.Text = RXD.Substring(43, 4)

End If

Timer1.Enabled = True

End Sub

This is the code for the FCS calculation.

Private Sub GetFCS()

'This will calculate the FCS value for the communications

Dim L As Integer

Dim A As String

Dim TJ As String

L = Len(TX)

A = 0

For J = 1 To L

TJ = Mid$(TX, J, 1)

A = Asc(TJ) Xor A

Next J

FCS = Hex$(A)

If Len(FCS) = 1 Then FCS = "0" + FCS

End Sub

Here is the code to actually send and receive the information from the Omron CP1H PLC.  
Note: Once information is sent on the port, we delay for 50 msec. This is more for the computer wait time. It can be 0 if nothing else is using computer ports. 50 works well when we are running CX-Programmer (USB Serial) and our VB.Net application (COM5).

Private Sub communicate()

'This will communicate to the Omron PLC

Dim BufferTX As String

Dim fcs\_rxd As String

Try

RXD = ""

BufferTX = TX + FCS + "\*" + Chr(13)

'Send the information out the serial port

SerialPort1.Write(BufferTX)

'Sleep for 50 msec so the information can be sent on the port

System.Threading.Thread.Sleep(50)

'Set the timeout for the serial port at 100 msec

SerialPort1.ReadTimeout = 100

'Read up to the carriage return

RXD = (SerialPort1.ReadTo(Chr(13)))

Catch ex As Exception

'If an error occurs then indicate communication error

RXD = "Communication Error"

End Try

'Get the FCS of the returned information

fcs\_rxd = RXD.Substring(RXD.Length - 3, 2)

If RXD.Substring(0, 1) = "@" Then

TX = RXD.Substring(0, RXD.Length - 3)

ElseIf RXD.Substring(2, 1) = "@" Then

TX = RXD.Substring(2, RXD.Length - 5)

RXD = RXD.Substring(2, RXD.Length - 1)

End If

'Check the FCS of the return information. If they are not the same then an error has occurred.

Call GetFCS()

If FCS <> fcs\_rxd Then

RXD = "Communication Error"

End If

End Sub

End Class

Here is the code running:

You can download this program as well as the[source code here](https://www.dropbox.com/s/zzsqzi5enbz8iyn/Omron%20Serial%20Communication%20VB%20net.zip?dl=0).