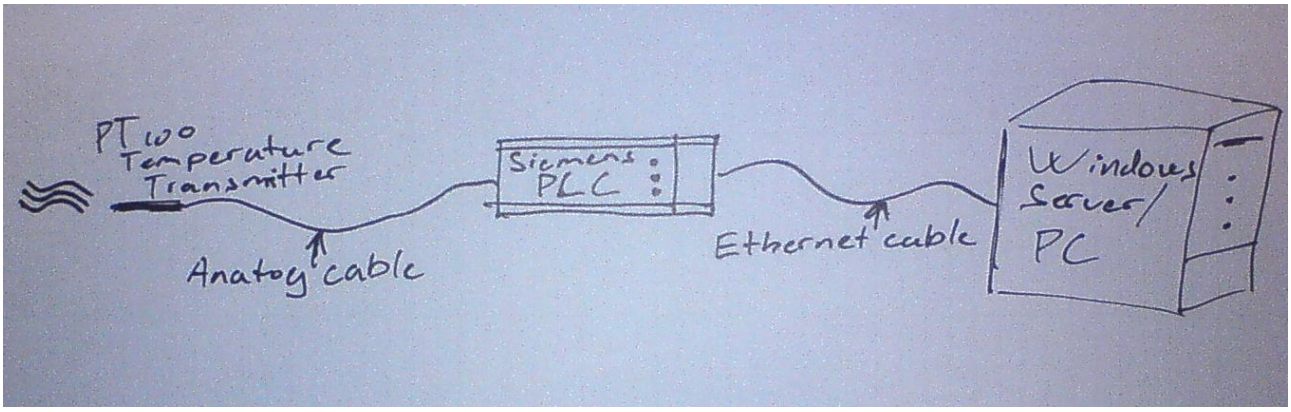
The guide about TCP/IP connections between PC's and PLC's

By Rasmus Frederiksen



Contents:

Intro

I am studying to become a type of electrician that is called control and regulation electrician. I am raising my English level to the level D. In this context I am writing this guide.

The reason I started exploring this area was that I had a project at home, where the job was to log a temperature to a graph. I searched the area for programs that was already made, but soon I found out that I would have to buy an expensive OPC server. The OPC server is a piece of software that reads a data-collection from a PLC, and makes it available on a PC. The PLC is a programmable logic controller which can handle electrical in- and out-puts very stable. Then I searched the internet for a solution for free, I found out that it is possible, but nobody has made a guide to tie the ends together. Therefore there were a lot of areas to explore, and it took many hours in front of the computer, to make the project finished. Now I am writing this guide, hoping that other people who are interested in the subject will find it useful.

Overview

This guide is intended for people, who want to make data acquisition from a PLC to a PC, via the protocol TCP/IP. The guide describes in details how to program the PLC. In the guide there will also be an example of how the code to receive the data on the PC could look like. The program on the PC will write the value sent from the PLC, to a text document.

The guide

For the example we will use a Siemens S7 1200 PLC, and program it with ladder diagrams. On the computer side we will use a PC with Windows XP installed. The program to collect the data will be programmed with VisualBasic.Net.

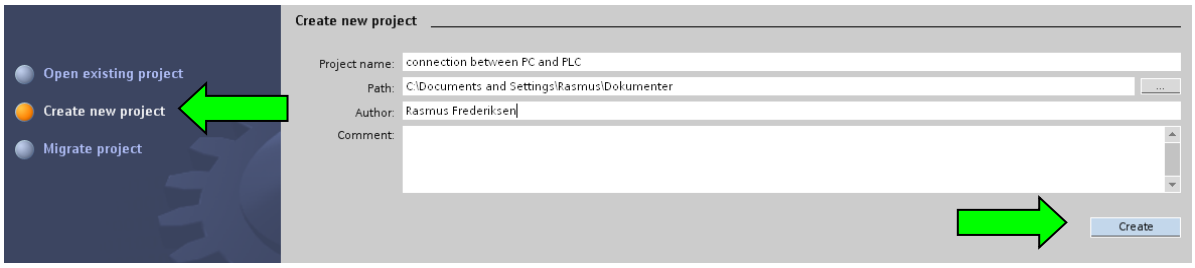
It will be an advantage if you know about these areas:

- PLC (Programmable Logic Controller), you should know how to program the controller, and how the controller works.
- PC (Personal Computer), generally knowledge on using a PC.
- TCP/IP (Transmission Control Protocol / Internet Protocol), generally knowledge on how to set up a static IP address on a network card.
- VB.Net (VisualBasic.net), VB.net is a programming language. I will strongly suggest that you learn the basics in this language, it will make the connection much more flexible and useful. You can make the connection without knowing vb.net, and just use the program example in the guide.

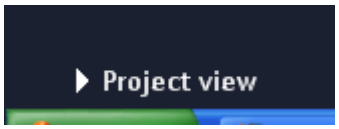
The PLC side of the connection

The first thing we will do is to program the PLC. The PLC we use is, as mentioned before, a Siemens S7 1200, and the Siemens TIA Portal V10.5 for programming it.

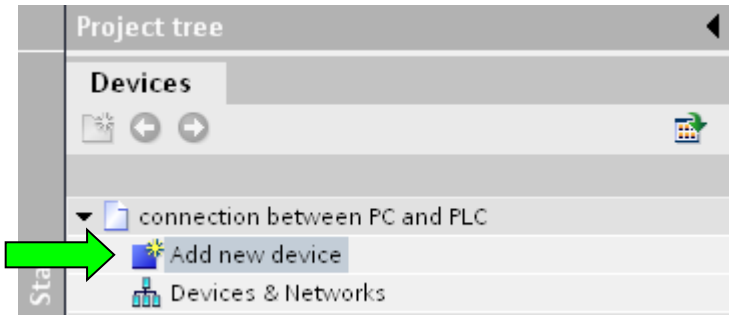
- First we will have to open the TIA portal software, and make a new project, as shown below:



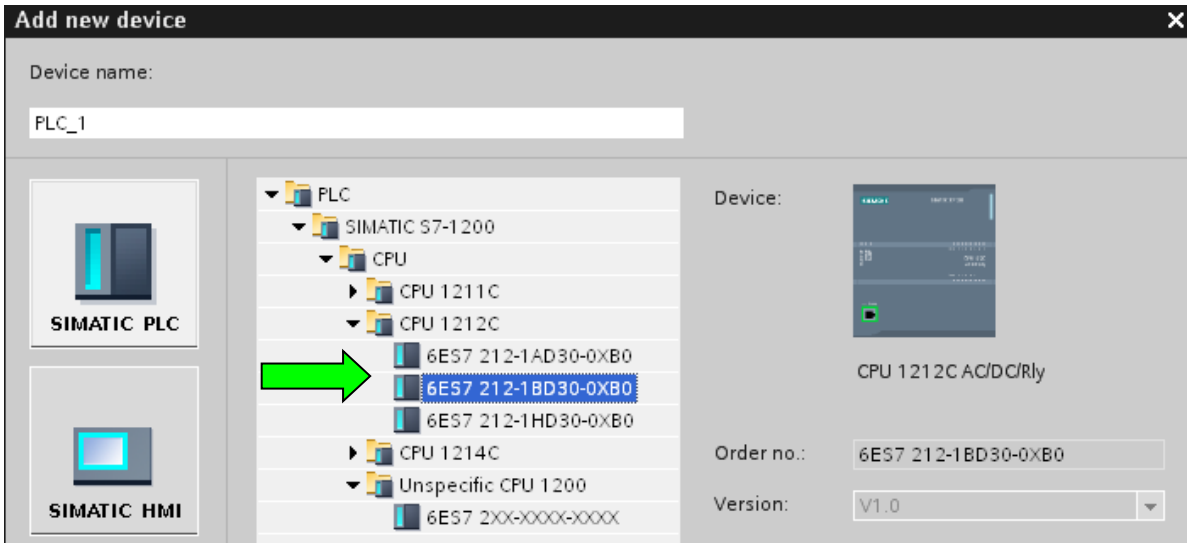
- When the project is created, we will connect the PLC to the PC with a network cable.
- We will now go to the project view by clicking the button in left lower corner:



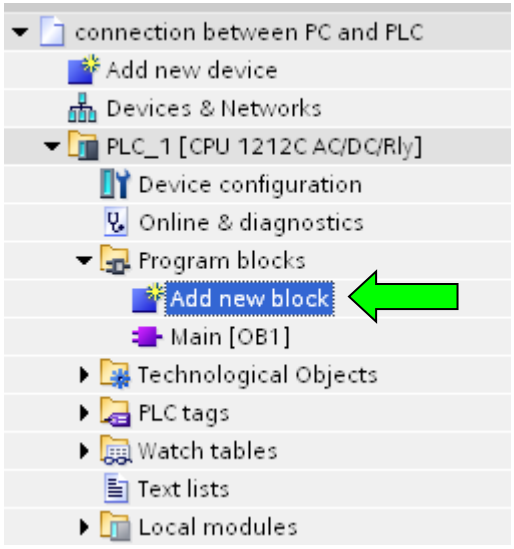
- To add the PLC to the project, we can click the Add new device, in the left side of the window:



- Then the Add new device window will appear. Here we can choose the PLC name. It is also here we select our CPU. We can either select an unspecified PLC/CPU to select your type of CPU later, or we can select a specified CPU. It would be an advantage to select later if you have expansion modules, because then the program can detect the expansion modules automatically. Here we will choose the CPU now:

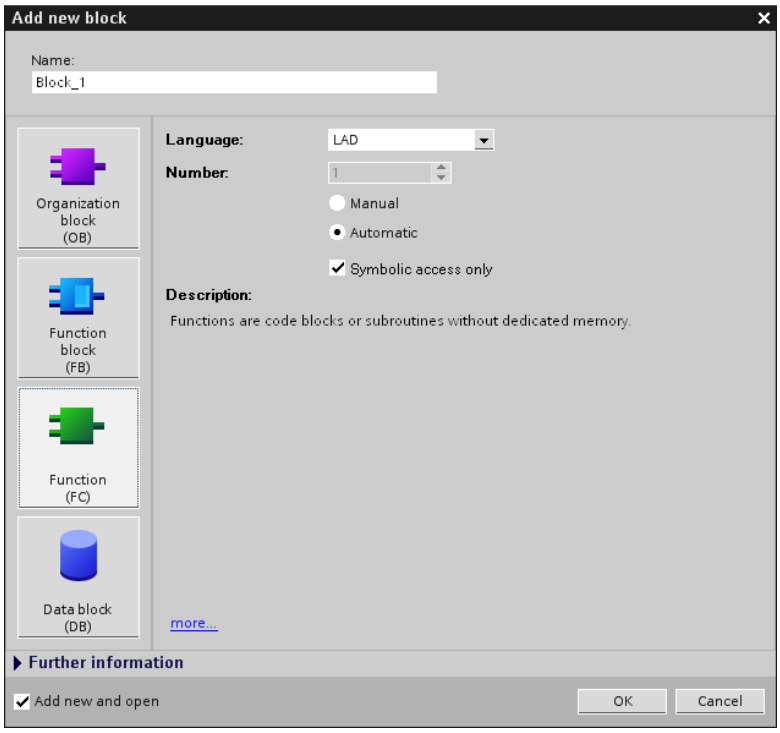


- When the CPU is chosen, it will appear in the project tree. We can now add a new block to the PLC program, by clicking the add new block button:

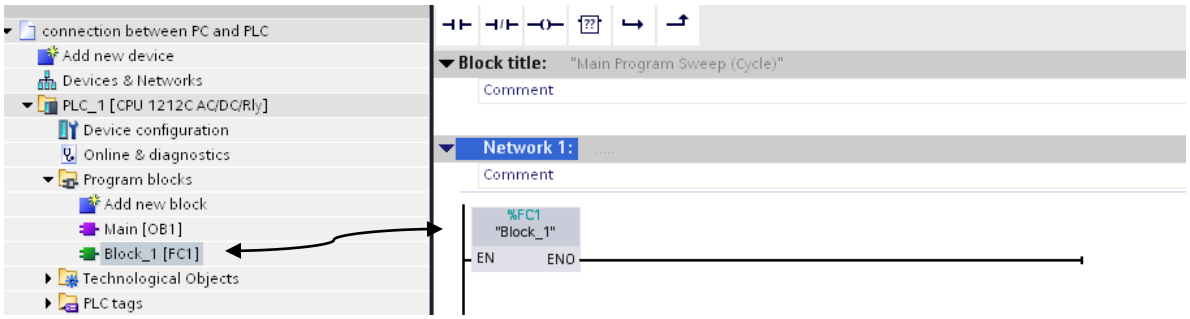


- In the Add new block window we can choose four types of blocks:
  - The first type is an Organization block (OB). It is the block where the other blocks are called. If a block is not called, the program will not be executed. In other words the operating block is the main procedure, functions and function blocks are sub procedures.
    The main block is by default added to the project.
  - Function blocks (FB) are a code block with dedicated memory. The dedicated memory can retain data when the CPU has power off. Function blocks acts like a sub routine.
  - Functions (FC) are also code blocs, but without dedicated memory. It is in this subroutine most of all programming is made.
  - A Data block (DB) is a place where data can be defined. When you define a type of data, you write a name and the data type you whishes, the name will then be bound to a place in the data memory. All data types can be retained.
    There is a variety of data types, the most used is:

Type:	Bit length:	Value range:
▪ Boolean	1	true/false
▪ Byte	8	255
▪ Word	16	65535
▪ Unsigned integer	16	65535
▪ Signed integer	16	-32768 to 32767
▪ Real	32	Signed with floating point
▪ String	255 byte	255 ASCII chars

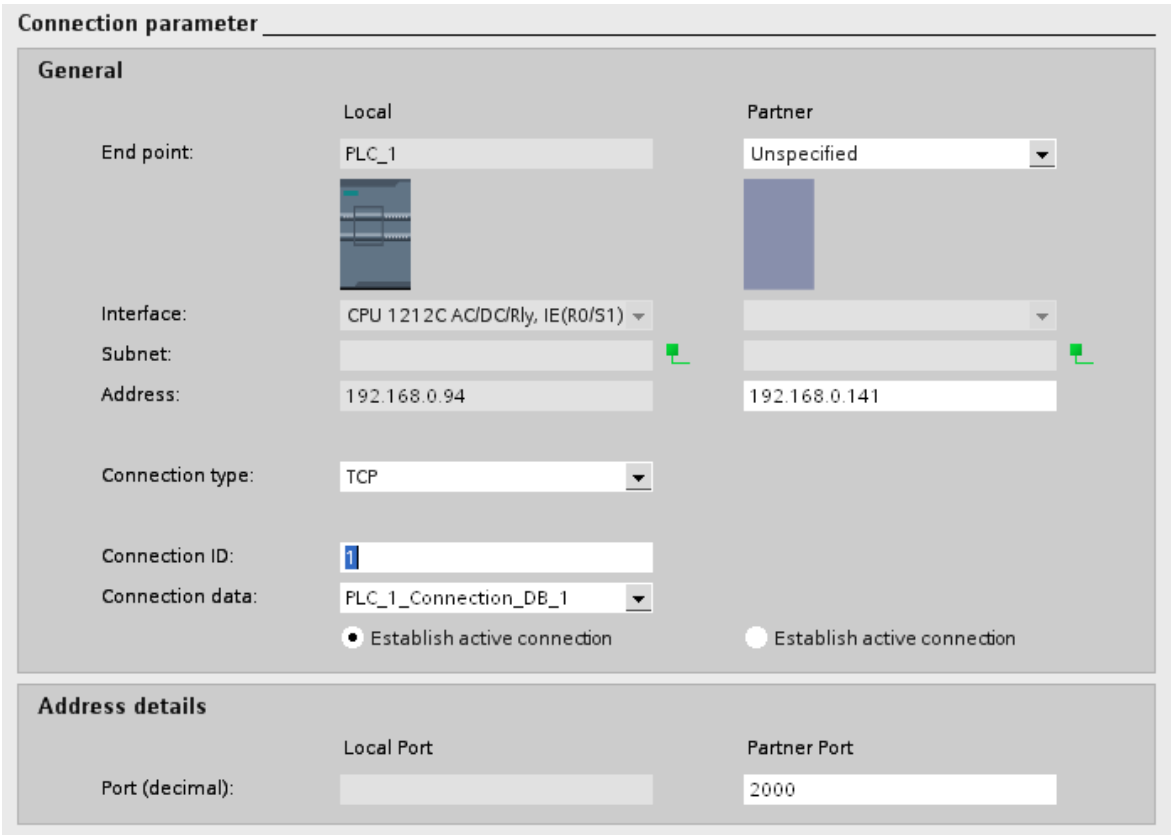


- We will use a FC (Function) to handle the program.
- When you have clicked the FC and ok, the block will appear in the project tree.
- Now we will add it to the Main block, so that it will be executed when the PLC runs. To add it to the Main block, open the Main block and simply drag the FC from the project tree to the first network in the Main block:



- Open the FC block Block_1.
- We will also be using a Data block. To add it, follow the same procedure as before, just without adding it to the Main block.
- We are now ready to add the TSEND_C object. TSEND_C is an object that controls a socket connection over TCP/IP.
  Socket is the definition of a port and an IP-address, on a network card, where an object can write to, or read from. A network card is defined by one IP-address, and ports from 0 to 9999. Said in another way, a network card can contain several connections on different ports, but on the same IP-address.
- Now we will add the TSEND_C object. You find it in the right tree of the screen under Extended instructions and under Communications. Again simply drag the object to network 1 in Block_1:

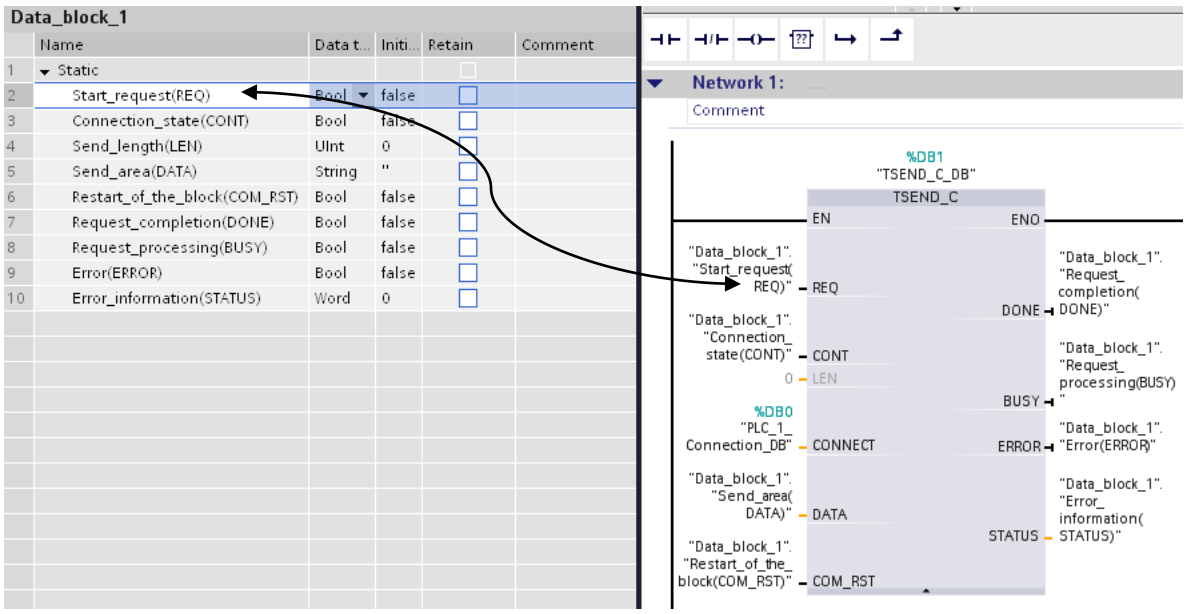
- Now we will see the TSEND_C object, and if we click on it we have a lot of options in the properties window below:



- In this project we will use the parameters as above. Here is a description of the parameters:
  - End point: The end points are the two devices, between which the connection is made. The local endpoint is the PLC we are sending from, and the partner is the device we are sending to. If we would like to send to another PLC S7 1200 we could add it to the project, and connect them via Ethernet in the Devices and networks window. Then we would be able to see it in the partner list. We are not sending to a PLC but to a PC, therefore we will use the unspecified partner.
  - Address: The address is the IP addresses we would like to use. The local address is by default the IP address of the network card on the CPU of the PLC. The partner IP address is the IP address of the network card we would like to send to. In our project it would be the IP address of the PC.
  - Connection ID: Each connection has their own ID.

- Port — The partner port, is the port on the network card of the PC, the PLC is writing the data to.

- We will also have to set up our Block parameters. To do that, we need to define some data types in our Data block. Open the Data_Block_1 in the project tree, and type the following names and data types:

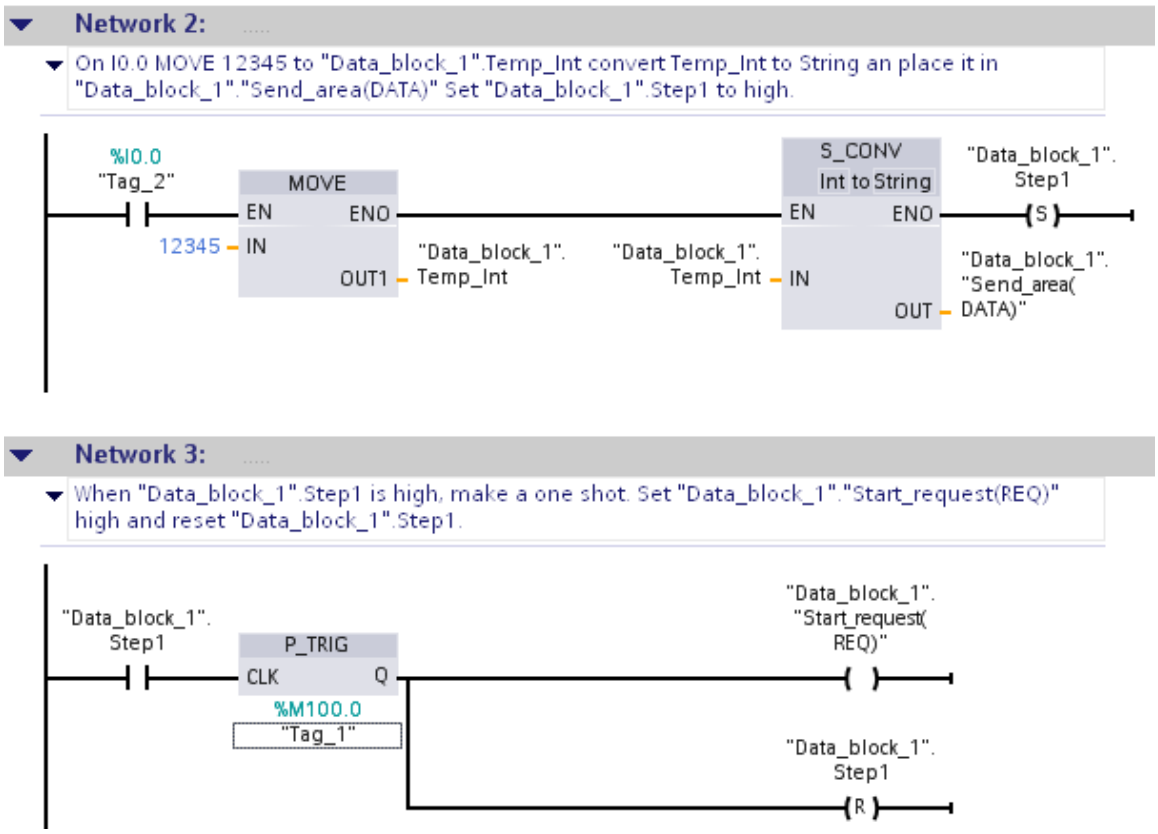


- When we have defined all the data areas that we need, we can drag then to their places on the TSEND_C block, as seen above. The above picture is split, so that we can see the Data_block_1 and the Block_1 at the same time. To do that, click the split window icon at the toolbar:

- In this project we will only use two of the parameters, to send the data:
  - Start_request(REQ) — A Boolean input we will set high when we wish to send.
  - Send_area(DATA) — A String containing the data we wish to send.

  The other parameters are used to maintain the connection, and see the occurring problems. There is a fine description of the block and the parameters in the help section of the program. To access it click on the TSEND_C block and press F1.

- Here is the function we want: When an input is high at I0.0, the PLC will move a constant (12345) in to a temporary integer, convert the temporary integer to a String and put it to the Send_area String. When that is done, it will give a one shot pulse to the Send_request, the PLC will then try to establish a connection and send the data. The ladder would look as shown below: (The temporary integer is defined as integer and named Temp_Int in data_block_1)



- Now the PLC side of the program is made.

The PC side of the connection

This subject will not be explained in details. The reason is that there are a lot of good guides on the internet about making a program using Visual Basic.net, and other programming language.

The program on the PC side is as mentioned before, written in the language VB.net. VB.net is a very user friendly program language, because it is an object language. It is a part of the .net (dot-net) framework, which make it very useful.

I strongly suggest you to read and learn the language, if you want to use the data you receive from the PLC in your own way. I have written the program in Sharp Development, which is a great free alternative to Microsoft's Visual studio. Both of the programs are used to write, compile and debug code in.

When a program sends or reads a string, to or from a port on a network card, which is defined by an IP address, it is called a socket connection.

Here is an example of a program that reads the data, that the PLC has send to the network card of the PC, via a socket connection:

(Lines marked with a ' first, are comments)

```
'This is a console application!
```

```
Imports System
Imports System.IO
Imports System.Net
Imports System.Net.Sockets
Imports System.Text
Imports Microsoft.VisualBasic

Class MyTcpListener

Public Shared Sub Main()

    'Finds a new available file number
        Dim filNummer As Integer
        filNummer=FreeFile()

    'Finds the current date
        Dim datoOgTid, dato As Date
        datoOgTid = Datetime.Now
        dato = datoOgTid.Date

    'Makes a file for the current hour
        FileOpen(filNummer, "C:\Documents and Settings\Rasmus\Dokumenter\Test af _
socket temperature\" & datoOgTid.Date & " - " & datoOgTid.Hour & ".txt", OpenMode.output)

    'Writes, in the console, where the file is saved
        Console.WriteLine("Gemmer i: C:\users\server\documents\pasturaserings temperature\")

    'Makes a new TCP listener
        Dim server As TcpListener
        server=Nothing

    'Try to stabilize the connection
      Try
      'Define the TCPListener on port 2000,  and the IP address as 192.168.0.141
        Dim port As Int32 = 2000
        Dim localAddr As IPAddress = IPAddress.Parse("192.168.0.141")

      'Binds the defined port and IP address to the TCP listener we defined as server
        server = New TcpListener(localAddr, port)

      'Start listening for client requests.
        server.Start()
```

```
        'Buffer for reading data
          Dim bytes(1024) As Byte
          Dim data, dataNu, dataForrige As String
          Dim time As Date

        'Enter the listening loop
           While True
          Console.Write("Waiting for a connection from the PLC...")

        ' Perform a blocking call to accept requests
        ' You could also user server.AcceptSocket() here
          Dim client As TcpClient = server.AcceptTcpClient()
          Console.WriteLine("Connected!")
          Console.WriteLine("Writes data to the file.")

          data = Nothing

        ' Get a stream object for reading and writing
          Dim stream As NetworkStream = client.GetStream()
          Dim i As Int32

          ' Loop to receive all the data sent by the client
            i = stream.Read(bytes, 0, bytes.Length)
            While (i <> 0)
            time = time.Now
            datoOgTid = Datetime.Now

            If time.Minute = 0 Then
               FileClose(filNummer)
               filNummer=FreeFile
               FileOpen(filNummer, "C:\users\server\documents\pasturaserings temperature\" &
datoOgTid.Date & " - " & datoOgTid.Hour & ".txt", OpenMode.output)
            End If

          ' Translate data bytes to a ASCII string.
            data = System.Text.Encoding.ASCII.GetString(bytes, 0, i)
            dataNu = mid(data, 6, 3)
            dataNu = dataNu / 10

          'Writes the incoming data to the text file, with a new line when the changes
             If dataNu <> dataForrige Then
            time = Time.Now
            PrintLine(filNummer, time.Hour & ":" & time.Minute & ":" & time.Second & " - " & dataNu )
```

```
            End If

            dataForrige = dataNu
            i = stream.Read(bytes, 0, bytes.Length)

          End While

      ' Shutdown and end connection
          client.Close()
        End While
  Catch e As SocketException
    Console.WriteLine("SocketException: {0}", e)
  Finally
    server.Stop()
  End Try
    Console.WriteLine(ControlChars.Cr + "Hit enter to continue....")
    Console.Read()

 End Sub 'Main

End Class 'MyTcpListener
```

Source references

- Books:
  - VB.NET By Poul Müller Olsen and Karsten Skov
- Internet sites:
  - The program example my program example is build on http://msdn.microsoft.com/en-us/library/system.net.sockets.tcplistener.start%28v=vs.71%29.aspx
  - A online free VB.net tutorial http://www.homeandlearn.co.uk/NET/vbNet.html
  - An example of a VB.net socket connection http://www.eggheadcafe.com/articles/20020323.asp
  - A overview of how socket works, and some VB.net program examples http://vb.net-informations.com/communications/vb.net_socket_programming.htm
  - A guide about data transferring with TCP from Siemens http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=19033929&caller=view
- And of course the help section in the Siemens TIA portal. (Can be accessed anywhere in the program, by pressing F1)