RESEARCH PAPERS
FACULTY OF MATERIALS SCIENCE AND TECHNOLOGY IN TRNAVA
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

2021 Volume 29, Number 49

INDUSTRY COMMUNICATION BASED ON TCP/IP PROTOCOL

Martin BARTOŇ[1], Roman BUDJAČ[1], Pavol TANUŠKA[1],
Peter SCHREIBER[1], Tibor HORÁK[1]

[1]SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA,
FACULTY OF MATERIALS SCIENCE AND TECHNOLOGY IN TRNAVA,
INSTITUTE OF APPLIED INFORMATICS, AUTOMATION AND MECHATRONICS,
ULICA JÁNA BOTTU 2781/25, 917 24 TRNAVA, SLOVAK REPUBLIC
e-mail: martin.barton@stuba.sk, roman.budjac@stuba.sk, pavol.tanuska@stuba.sk,
peter.schreiber@stuba.sk, tibor.horak@stuba.sk

Received 26 August 2021, Accepted 27 October 2021, Published 24 November 2021

Abstract

The design of control systems needs to ensure communication between multiple PLCs or external IoT devices. Furthermore, there are several ways of communication between them. This article dealt with the PLC communication based on TCP/IP protocol, and compared several communication options between two PLCs of the S7-300 and S7-1200 series connected using Profinet with a solution built in C# connected to the PLC via an OPC server. We used MODBUS TCP, TCON, and PUT/GET program blocks for comparison. We transmitted a digital Boolean data type and an analogue value of the integer data type via Profinet to read data from the S7-300 PLC and send data to the S7-1200. We compared the programming instructions standardly used as a client/server PLC programming with a Windows Forms application, and evaluated the advantages and disadvantages of this solution even when using external IoT devices. The solution was applied and successfully tested for communication between PLC S7-1200 and Nvidia Jetson Nano. We wrote program instructions for PLC in Siemens Tia Portal V15.

Keywords

PLC, TCP/IP, communication

INTRODUCTION

Programming complex control systems requires ensuring the proper communication of industrial automats and IoT devices. The control of automated industrial lines always require several PLCs which must be able to communicate with each other and transmit the necessary information, as well as send data about the controlled system to superior systems for their archiving and further processing. Further processing can be, for instance, the analysis of machines for predictive maintenance, analysis of production capacity utilization to optimize

production processes and eliminate unnecessary downtime. In this context, communication needs to take place effectively.

In this article, we centred the attention on the actual method of communication between individual PLCs. There are several options of how to communicate between programmable logic automats. The most common way of such communication is to use pre-programmed graphic blocks, where it is necessary to configure individual parameters. Our article focused on the blocks implemented in the development environment TIA Portal V15, especially the block TCON, PUT/GET and MODBUS. Besides the existing solutions, we presented our own one, which regards the implementation method differently and prefers implementation of the TCP/IP communication protocol in one of the object-oriented programming languages, rather than focus on programming blocks supported by the manufacturer.

This paper also clarifies the differences in implementation, using the standard IEC language PLC protocol using pre-programmed LAD blocks, and our own algorithmic solution utilizing C# and OPC server. We implemented a communication algorithm in C# for sending and receiving data through the MODBUS protocol between PLC S7-300 and PLC S7-1200. In this case, various empirical studies have been conducted to use TCP/IP protocol communication [1] [2] [3] [4]; however, the PLC communication blocks explanation was not detailed enough in relation to the research topic.

MATERIALS AND METHODOLOGY OF EXPERIMENT

PLC as a fundamental device in automation has long history and, at present, we can see various manufacturers bringing different development environment particularly for their own devices. The lack of open-source development environment, causing differences and there is no universal way to develop a program using standard IEC language. A partly free solution is Codesys which brings its own approach to programming respecting IEC. We attempted to propose the OOP approach to programming into automation, as well. This lack of diversity is due to the fact, that the chosen approaches to communication must copy the approaches supplied by the manufacturer, and there is no room for own solutions of complex tasks, which would be supported by the manufacturer and use other than the predefined language of the PLC manufacturer.

Communication blocks PUT/GET

For its simple (straightforward) solution, PUT/GET is one of the most commonly used blocks for programming a connection between two PLCs. PUT is used to send data, and GET to receive sent data. The PUT/GET communication system works on the client/server connection. This S7 connection is not only possible between two PLCs directly in one project, but also supports connection with an unspecified virtual PLC partner. However, this protocol cannot be used to communicate with external IoT devices. In S7-300, the PUT creates its own data block PUT_DB, and the function block PUT_E. The GET creates its own GET_DB data block in the S7-1200.

PUT/GET is used to send multiple variables between two PLCs using one communication block, but it is always necessary to allocate each variable using pointers.

PUT/GET outputs to determine the status of communication with DONE / NDR flags - logical 1, if the communication was successful, ERROR - logical 1, if an error occurred during the communication, and STATUS - a numeric variable to determine the type of error message. These outputs can be used as markers to run the code based on logical conditions, depending on the successful connection of the PLC.

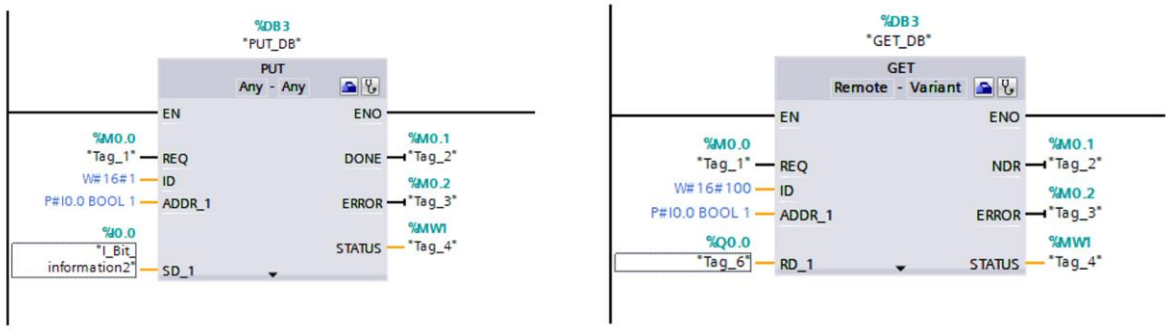

Fig. 1 *PUT S7-300 communication block and GET S7-1200 communication block*

Communication block TCON

It is a programmable block that can establish a TCP/IP connection between a PLC and a PLC. The TCON block must be used in both PLCs between which creates the connection. If these blocks are used in the S7-300 PLC, the TSEND must also be used, and the TRCV block must be used for the S7-1200 PLC. During compilation on the S7-300 PLC, TCON creates an additional data block TCON_DB and the TCON function block. Compiling of Block TSEND creates data block TSEND_DB and function block TSEND. Using of TCON communication generates a data block PLC_2_Connection_DB.

In the case of the S7-1200, TCON creates the TCON_DB data block, the TRCV block creates the TRCV_DB auxiliary data block in the S7-1200. To start communication, TCON creates a PLC_1_Connection_DB data block. Both communication data blocks are necessary for proper communication. It should be noted that TCON, TSEND, and TRCV run asynchronously. This fact causes running process over multiple executed instruction and executes instructions sequence. [5]

For these purposes, it is necessary to maintain data consistency during the connection between CPUs. The instruction level CPU provides necessary data consistency by verifying all data types used during the process and controlling system data structures. Another tool given to the programmer is to disable alarm interrupt and enable alarm interrupt. These two blocks ensure that an interrupted OB is not executable during the read/write operation. Nonetheless, these two instructions cause additional latency to interruption time. [6]

Similarly to PUT/GET, TCON has outputs to determine the communication status. With TCON, it is also possible to use the BUSY status in case the communication has not been finished yet.

Similarly to PUT/GET, the communication protocol cannot be used to communicate with external IoT devices.

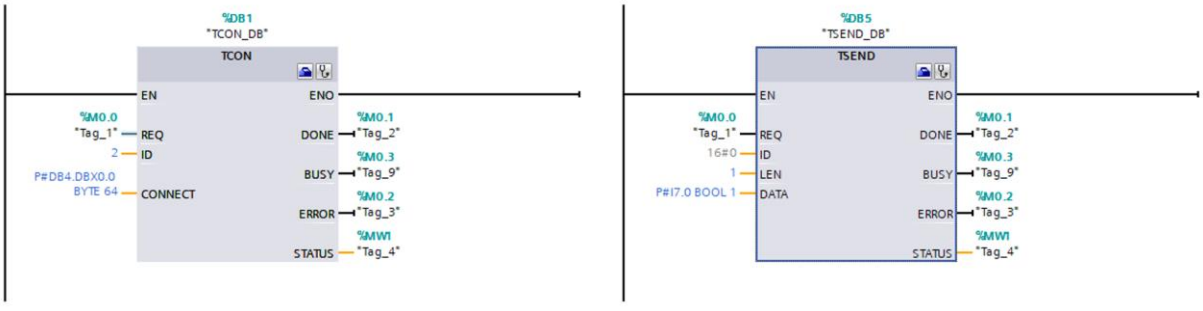

Fig. 2 *TCON and TSEND S7-300 communication blocks*

Communication MODBUS TCP/MODBUS PN

It is a programmable block that establishes connection via industrial Ethernet. As all mentioned before work so on TCP- Client and TCP-Server architecture. Modbus client and Modbus Server establishing connection via Profinet interface of the CPU. Client – server instruction is called for every connection using unique ID. Also instance data blocks are used when Modbus connection is called. It is specifically Write holding register and Read holding register. These registers are bool data type Read/Write register is used as a parameter for dataBuffer, which is pointer to buffer storing data that was written or read to the Modbus server. Function block Modbus Client is called cyclically in OB1. Client is established on the port number 505 with connection number = 2. Modbus dispose with Mode parameter. This parameter selects direction of data transmission. Data is sent via Modbus protocol in packets, consisting of a message header, function code and transmitted data. The message header consists of 7 bytes. The function code consists of 1 byte, and the data consists of n bytes, which depends on the length of the sent message. [1]

In our case, MODBUS on the S7-1200 side works as a client, on the S7-300 side as a server. Modbus in S7-1200 creates data block MB_CLIENT_DB and function block MB_CLIENT. [7] In S7-300, Modbus is called MODBUS PN. It creates data block MODBUSPN_DB and function blocks MOD_CLI, MOD_SERV, and MODBUSPN, while based on the Write / Read parameter, we decide which Modbus blocks in S7-300 will be used. Server supports IP v4 TCP type of connection and TCP port number: 20,21,25,80,102,123,5001,34962,34963 and 34964 are restricted. [8] The Modbus TCP communication protocol can also be used for communication with external IoT devices.

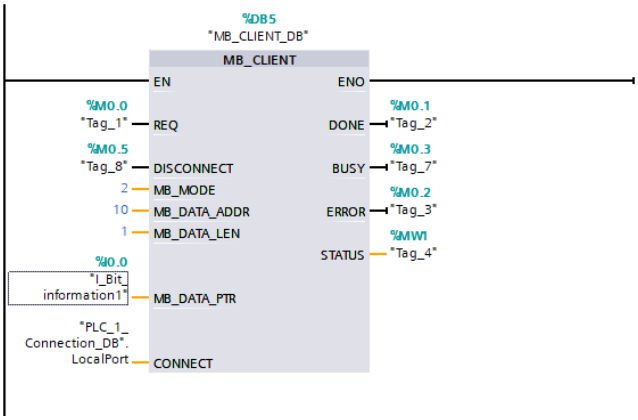

Fig. 3 *MODBUS TCP client S7-1200 communication block*

All types of communication mentioned above are suitable for use between two PLCs, but the blocks are challenging to configure in TIA Portal and when creating communication blocks, new data and function blocks are created, which also fills the PLC memory. The solution we propose will be implemented in the C# program and connected via an OPC server with two PLCs. The algorithm that the program will operate can be seen in the Fig. 4.

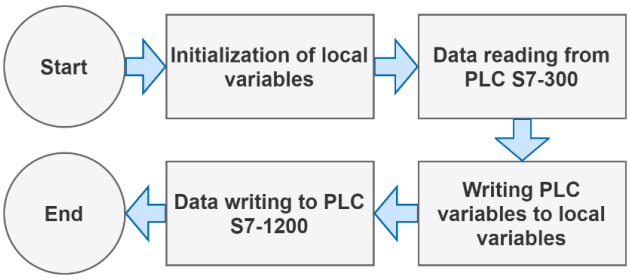

Fig. 4 *Proposal of communication sequence in C# application*

ATTAINED RESULTS

All the solutions we presented are complex to configure and set up when programming the PLCs. Our presented approach bypasses this problem. It architecture is a more complex than that in the case of direct PLC connection via Profinet, since it uses a connection to a PC via an OPC server, as shown in Fig. 5. We also connected Jetson to OPC server similarly to PLC S7-300 to read data.

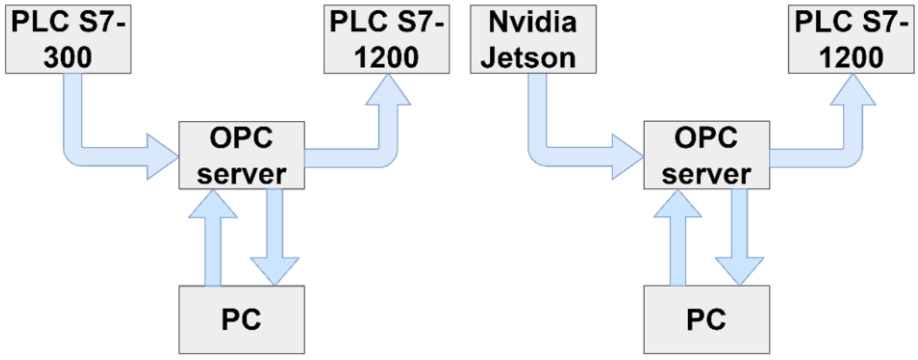

Fig. 5 *PLC/Nvidia-PC communication via OPC server*

As an OPC server, we chose Nettoplcsim, which is freely available and offers sufficient functionality to provide a functional solution. The solution is designed in Visual Studio in C# as a Windows Form Application. It uses the S7.Net library, which provides the options for working with PLCs from Siemens from the S7-200 series to the S7-1500 series, such as reading/writing inputs, outputs, markers or data in data blocks from/to PLC. [9] Our program works as an intermediate element in reading data from PLC S7-300 and writing read data to PLC S7-1200. The following block of code is used for this purpose:

```
bool db1BoolVariable;
int db1WordVariable;
using (var plc = new Plc(CpuType.S7300, "10.1.1.18", 0, 2))
{
    db1BoolVariable = (bool)plc.Read("DB1.DBX0.0");
    db1WordVariable = (int)plc.Read("DB1.DBW2.0");
}
using (var plc = new Plc(CpuType.S71200, "10.1.1.62", 0, 2))
{
    plc.Write("DB1.DBX0.1", db1BoolVariable);
    plc.Write("DB1.DBW3.0", db1WordVariable);
}
```

The script begins by allocating bool and integer variables in the memory which are used to transfer data. Line 3 starts the sequence of reading data from PLC S7-300. In this line, it is necessary to define the PLC type, PLC IP address, rack number and slot number. The part of the code in curly braces accessing the PLC variables directly and loads the data into local variables. The *plc.Read* command must be preceded by the data type of the read variables in parentheses, and the specific variable read from the PLC is specified in the braces after the command. Line 8 runs the sequence of sending data to PLC S7-1200. The *plc.Write* command has two flags, the first one is the address in the PLC where the data is written, while the second is the data itself or a variable sent to the PLC. The OPC server establishing a communication layer between PC and PLC and ensures the correct reading and sending data from/to the PLC. The solution is based on a star network topology putting the TCP/IP protocol in service and is designed according to the design mentioned above. Communication via C# application and OPC server can also be applied to connect external devices to PLCs, such as Jetson Nano, which would provide an effective solution for communication with tasks runs artificial intelligence algorithm, where the application could also be used to evaluate data from devices and settings required action on the PLC.

DISCUSSION

Communication with the PLC via the OPC server provides a way to efficiently transfer, send or read the values from the PLC. The solution offers certain advantages over the conventional way of communication that has to be taken into the account.

As the proposed program takes over part of the processes performed on the PC, the PLC itself is relieved. It is not necessary to use functional communication blocks such as TCON or MODBUS, which creates fewer data and functional blocks and relieves program memory. This advantage applies to the S7-300 PLC, as it creates more blocks than the S7-1200 PLC when creating communication and has less performance compared to the S7-1200.

The advantage is the absence of complicated setting and configuration of parameters in the TIA Portal environment to establish communication between PLCs. This is particularly advantageous if you need to connect a large number of PLC devices that need to be interconnected. Using the C# application, it is possible to send a large amount of data between multiple PLCs without the need to configure each PLC separately in the TIA Portal.

On the other hand, it is necessary to mention the weaknesses of the proposed communication. In terms of network topology, this is a more complex architecture than in the case of direct PLC connection and it is necessary to use an additional OPC server for the correct function of communication between PLC and C# application. With this type of communication, it is also necessary to consider the security of the solution if the used PC is connected to the external Internet network. However, advantages significantly exceed the disadvantages of connection, especially in the case of using Jetson Nano communicating with the PLC as an external device. In this case, the OPC server is difficult to avoid.

CONCLUSION

The paper deals with the communication between programmable logic controllers using Siemens PLC and external devices. We focused on traditional solution provided by a manufacturer in the Siemens Library. We also explored the communication methods using OPC server and C# language. In the discussion section, we compared those approaches and highlighted the advantages and disadvantages.

There are a lot more ways for communication of PLC for future research. The future most important aim is reducing the technology gap between new technologies like deep learning computer vision task and industry automation control mainly based on PLC devices.

Principally, we see opportunities in MODBUS TCP/IP protocol and implementing it to create a communication channel for transmitting data from edge devices such as Jetson super computer, and bringing deep learning recognition information to PLC and move to higher SCADA system.

As we proved, applying an OPC server is also advantageous and functional for establishing a connection between Jetson devices and PLCs. For further research, it is essential to examine the speed of communication and the delay between the PLC and the Jetson Nano tensor operations in a time-intensive task.

The research described in the article used an OPC server as a replacement for the complex configuration of communication between PLCs, while we could easily add additional devices to our hardware infrastructure. The OPC server can also be used to connect IoT devices, while using the same way of communication between PLC and IoT devices.

Acknowledgement

This publication covers partial results of the project of Young Scientist 1360 (2021-2022): *Applying machine learning at industry environment*, supported by the Young Scientist internal grant of the Slovak University of Technology in Bratislava.

References

[1] YING, L., YAO, L. 2016. Research on Ethernet Communication and Remote Control of PLC and PC. In: *Proccedings of the 2016 3rd International Conference on Materials Enginnering, Manufacturing Technology and Control*, 29 Avenue Lavmiere, Paris, 75019, France, 2016.

[2] TAMBOLI, S., RAWALE, M., THORAIET, R., and AGASHE, S. 2015. Implementation of Modbus RTU and Modbus TCP communication using Siemens S7-1200 PLC for batch process. In: *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*.

[3] KIM, Y., LEE, S. Y., LIM, S. 2020. Implementation of PLC controller connected Gazebo-ROS to support IEC 61131-3. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.

[4] MIYAZAWA, I., Et Al.. 2011. OPC UA information model, data exchange, safety and security for IEC 61131–3. In: *SICE Annual Conference 2011*.

[5] www.siemens.com, "S7-1200 Programmable controller (V4.2.3, 08/2018, English)," 2020. [Online]. [Accessed 06-2021] Available at: https://support.industry.siemens.com/cs/mdm/109759862?c=112125944459&lc=en-CN.

[6] www.siemens.com, Understanding data consistency, [Online].[Accessed 07-2021]. Available at: https://support.industry.siemens.com/cs/mdm/109759862?c=65872045195&lc=en-CA.

[7] www.siemens.com, Modbus/TCP with instructions "MB_CLIENT" and "MB_SERVER", [Online]. [Accessed: 05-2021] Available at: https://cache.industry.siemens.com/dl/files/340/102020340/att_118119/v6/net_modbus_tcp_s7-1500_s7-1200_en.pdf.

[8] www.siemens.com, Address the memory areas in the SIMATIC S7-1200/S7-1500 and in the Modbus device in the case of Modbus/TCP data communication, [Online]. [Accessed: 07-2021], Available at: https://support.industry.siemens.com/cs/document/100633819/how-do-you-address-the-memory-areas-in-the-simatic-s7-1200-s7-1500-and-in-the-modbus-device-in-the-case-of-modbus-tcp-data-communication-?dti=0&lc=en-WW.

[9] STŘELEC, P., BARTOŇ, M,. TANUŠKA, P., KEBÍSEK., M., ŠPENDLA., L. 2020. Real Time Data Acquisition as a Part of Data Processing from Production Line According to. In: *WCICSS*, London.

ORCID

Pavol Tanuška 0000-0001-7025-1911
Peter Važan 0000-0001-6943-2635
Tibor Horák 0000-0002-1620-6199