Developing and Using Communication Driver for Serial Communication Between PCs and Industrial PLCs

Zoran M. Milić[1], Petar B. Nikolić[2], Dragana Krstić[3] and Miljana Lj. Sokolović[4]

***Abstract* – The basic principles of the asinhronius serial communication between PLC controllers and PC based applications are presented in this paper. In order to create HMI interface between operator and the controller at the machine, the Windows based application for vizualization and data acquisition was developed. This application is using self-oriented driver for serial communication which was developed and used for purposes of communication and data exchange between industrial controllers working at machines in the plant floor, and PCs. Basic principle of generating communication packets and examples of writing data into PLC's memory registers are given. This driver is also used in self-oriented SCADA programming for data exchange between SCADA application and controllers on the industrial network.**

***Keywords* – PLC, industrial network, network layer.**

I. INTRODUCTION

For visualization of the interface between the operator and the machine, it is possible to use different hardware solutions, from the industrial panels, intended only for these purposed and related to a particular PLC type, to the industrial and non-industrial PC's with a proper operating system installed as well as the appropriate user's application program. For the realization, it is suitable to use common personal computer since it enables the enlargement of the system and the integration into the wider information system.

The industrial application specific panels have no ability for multi-tasking work. They need additional communication with PCs or other controllers in the control system where complex calculations are needed. With the respect to those calculations, a change of the data in the registers of the controllers takes place as well as the activation of the appropriate outputs. These panels have neither additional ports, nor the ability to install additional cards (PCI, PCE express) that are obtained as a part of the sensors' and actuators' kits. When the industrial panels are connected to the larger industrial network, and one of the devices blocks its message transmition, this can break the communication of the entire network.

For the communication between the PC and the controller, communicating applications of the controllers' manufacturer can be used, or a specific communication driver can be developed. The communication software enables an easy integration into the application for the machine control, or into the SCADA software, and this decreases the time needed for the entire system realization. Some of these communication software are in the same time OPC servers so that they allow relatively easy integration of the data sources devices from different manufacturers into the unique visualization and acquisition systems.

Writing its own communication driver allows good controller management. It is possible to adjust the operating mode, which is switching between the programs, run and test modes, which is very useful for application testing. Managing the program upload and download to and from the controller is possible to be achieved directly, through the serial interface, or through the network, using the proper network module. In this way, the change of the status registers, control of the restart or switching of the controller, and the memory initialization can be achieved. Direct changing the parameters of the communication link (for example, baud rate, parity bit, stop bit, hardware/software control, BCC/CRC) is possible, and this represents the advantage considering the entire system testing. HMI (Human Machine Interface) applications, which communicate with the controller directly over the driver, perform it in the real time.

This solution has better diagnostics abilities, enables an acceptable real-time response of the machine, gives a bigger independence, but also increases the entire system design and development. Beside the time needed for the design, the important factor in the solution choice is the prize. In the case of SCADA application design, the prizes of the both system are comparable, since

In the case of SCADA application design, the prizes of both systems are similar since the number of the acquisition places, and though the places than need the communication software, relatively small, comparing to the number of PLC's in the system. By the HMI applications, the situation is different. The number of places that need the communication software is the equal as the number of PLC controllers - that is, each machine must have PLC and the particular interface toward the operator, which communicate with that PLC. In this case the prize of developing particular driver is much lower than buying a new solution.

[1]Zoran M. Milić is with the Tigar MH, Pirot, Serbia E-mail: zrondjul@yahoo.com

[2]Petar Nikoić is with the Tigar MH, Pirot, Serbia E-mail: nikpetar@tigar.com

[3]Dragana S. Krstć is with the Faculty of Electronic Engineering, University of Niš, Serbia E-mail: dragana@ elfak.ni.ac.yu

[4]Miljana Lj. Sokolović is with the Faculty of Electronic Engineering, University of Niš, Serbia E-mail: miljana@venus.elfak.ni.ac.yu

II. NETWORK LAYERS OF THE COMMUNICATION MODEL

Network architecture of the DF1 industrial network has physical layer, data layer, network layer and the application layer [1].

In the case of serial RS232/RS422 communication, physical layer consists of the RS232/RS422 ports at the PC and PLC, RS232/RS422 cable that connects them, voltage values (zeros and ones), number of data bits, number of stop bits, parity bit, bit rate, and the way of establishing and breaking the connection after PC and PLC finish the data transfer [3].

The data connection layer controls the correctness of the data transfer, and the protocol at this layer should provide the mechanism for the acknowledgement of the correct data transmission and reception [2].

The network layer controls the packet transmission to its destination, which is for establishing the connection between the nods of the network [4].

Application layer in the case of PLC and PC communication contains the commands that are set and executed by the PC and the PLC applications.

III. DF1 LAYER OF THE DATA CONNECTION

DF1 is the Allen Bradley's protocol for the data connection layer, which is based on the ANSI x3.28 specification. The basic principle of the DF1 protocol will be explained at the example of Full Duplex data exchange [1].

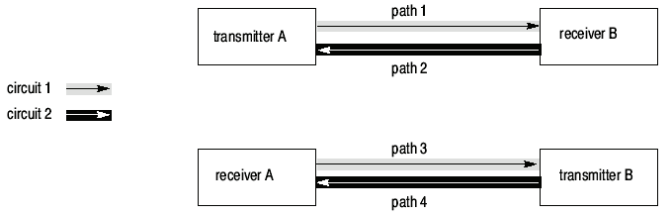

Fig. 1. The data routes for alternating simultaneous communication (Full duplex)

At the Full Duplex protocol (Fig. 1), link uses two physically separated circuits for the simultaneous data exchange. These circuits provide communication at four communication channels.

- In the first circuit the transmitter A sends messages to the receiver B (the route 1) and the receiver A send the returning control messages to the transmitter B (route 3).
- In the second circuit the receiver B sends messages to the receiver A (the route 4) and the receiver B sends the returning control messages to the transmitter A (the route 2).

All messages and the symbols in each of these circuits are transmitted in one direction; from A to B in the first, and from B to A in the second.

- In order to implement 4 logical routes in 2 physically separated circuits a software multiplexer must be used. Its purpose is to combine the command messages (from the transmitter) with the returning messages (from the receiver) as well as with the replies from the transmitter sent in the same direction.
- At the other end of the link, the separator software separates the command messages from the returning reply messages. The separator software should send the command messages to the particular receiver and the returning messages to the corresponding transmitter.
- Although the command and the returning messages in the same circuit exist independently from one another, there is a certain relation between them. For example, the command message in the AB circuit will be delayed if the returning message of the receiver A is inserted into the sequence of the common messages of the transmitter A. Each hardware problem that influences the command symbols in one circuit will also have the influence at the returning symbols in the same circuit [1].

IV. GENERATION AND SEPARATION OF THE DATA FRAME IN THE FULL DUPLEX PROTOCOL

The data frame in the Full Duplex protocol has different forms depending on the observed network layer, and considering the fact that different message parts are generated in different network layer. Fig. 2 illustrates how the particular network layer influences generating the message frame [5]. The influence of the physical layer is not shown.

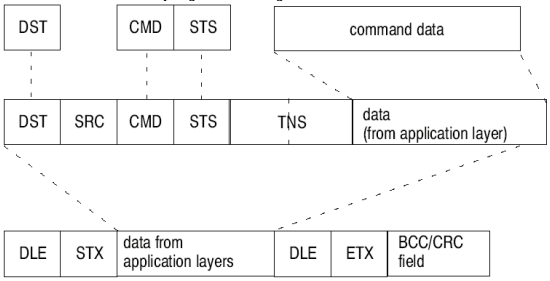

Fig. 2. Data frame: from the top to the bottom: application layer, network layer, data connection layer

The command (for example, read or write) is generated at the application layer, as well as the destination (the address of the PLC controller in the network) and the data related to that command (for example, in the case of reading, this would be the specification of the first memory location and the size of the block to be read).

The network layer is responsible for establishing connection between the communicating nodes, and the address from which the message is transmitted is added here (so that returning message has the correct destination address), transmission status field (which contains the error code for the eventual transmission errors), and the identifier, which is a unique for each message (in order to know which returning message is the reply to the given command).

Data connection layer is intended for the control of the correctness of the transmitted data – one more field is added for the beginning and one for the end of the message as well as the field for the error control.

In order to design the corresponding HMI solution one needs to develop a driver which cover a wide spectrum of communication messages that enable: reading and writing into and from the controllers' registers, changing the content of the status registers, changing the parameters of the

communication link itself, and simultaneous covering relatively large number of communicating devices.

The driver is the application specific for the asynchronous serial communication with the Allen Bradley PLC 5 and the SLC 5 families of the controllers as well as for the Allen Bradley KF2 communicating module [6].

The developed driver supports the following set of commands:

For PLC 5 family of controllers and a KF2 module:

- "Word Range Read" – reading the block of words from the controller's memory
- "Word Range Write" – writing the block of words into the controller's memory
- "Typed Read" – reading the block of data from the controller's memory (this command is also supported for the SLC 5/03 and SLC 5/04 processors from the SLC500 family)
- "Typed Write" – writing the block of data into the controller's memory (this command is also supported for the SLC 5/03 and SLC 5/04 processors from the SLC500 family)
- "Read – Modify - Write" – bit write command
- "Set Variables" – adjusting the parameter of the serial link – the number of the ENQ packets, the number of the NAK packets and the timeout interval
- "Set CPU Mode" – changing the controllers' operating mode: Test, Program and Run
- "Diagnostic Status" – reading the content of the controllers' status registers

For the SLC 500 and the MicroLogix 1000 family of controllers:

- "Protected Typed Logical Read With Three Address Fields" – reading the block of data from the controllers' memory, starting from the given address
- "Protected Typed Logical Write With Three Address Fields" – writing the block of data into the controllers' memory, starting from the given address
- "Change mode" – changing the controllers' operating mode: Test, Program and Run
- "Diagnostic Status" – – reading the content of the controllers' status registers

V. EXAMPLE

For communication between the PCs and the PLCs in the systems that control machines in the tires industry, asynchronous serial communication is commonly used. The role of the interface between the controller and the operator is given to the applications installed at the industrial PC. The reason for that is the additional processor power that system needs for processing the parameters obtained from the smart sensors, which is the task that most of PLCs cannot satisfy, or the prize of it implementation is to high.

The communication during one working cycle of the different machines is performed in a similar manner. The controller sends a request for the recipe data from the user application over the asynchronous serial communication. The data are read from the particular sensors and according to their content, the application reads some parameters from the data bases and calculates the parameters of the recipe. According to these data, the machine's parameters are adjusted, and the corresponding machine's operating cycle can begin. During the cycle, several communication sessions between a PC and a PLC can appear. During one session, a certain amount of data needed for the machine control is successively written into, and read from the registers of the controller in order to obtain the data needed for the additional analysis in the PC.

After completing the cycle and the analysis, performed by the user's application, the results of the processing should be written into the data base, and the signal for the initialization of the next cycle should be sent.

The described HMI application is implemented using the Microsoft Visual Basic.NET 2005 developing tool [7].

The example shows writing four integer words (1111, decimal) into the u Allen Bradley PLC 5/20 processor over the serial link, starting from the N8:31 memory location. To do this a command "Word Range Write" was used. In a similar manner, an entire communication over the serial link is performed, where the packets contain all necessary commands, addresses and data. During this procedure the following data exchange takes place (Fig. 3):

- The application sends a data for writing into the driver, that is, it specifies the writing command, all four words of the data and the destination address of the controller
- The driver generates the entire data frame for sending
- The transmitter of the driver sends the packet to the receiver of the PLC
- The receiver of the PLC receives the packet, forwards it for processing to the processor and sends the control message about the successful reception of the packet (DLE ACK) to the link
- After the writing command is successfully completed, PLC processor forwards the message about the successful reception to the transmitter.
- The transmitter of the controller sends the message to the link
- The receiver of the driver receives the message about the successful writing, the application forwards that message and sends the control message about the successful data reception (DLE ACK) to the link.

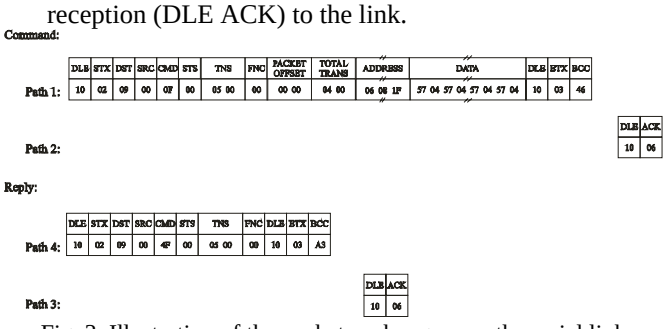

Fig. 3. Illustration of the packet exchange over the serial link

A. The driver

The driver was developed as a C#.NET Class Library. Since it represents an interface between the user's HMI application and the physical layer of the network, the following class properties were implemented:

- command – it is a command that forwards the application.

- address – represents the address of the controller in the network
- mem_address - specifies the starting address in the controller's memory where the writing is taking place
- packet_offset – specifies the offset in regard to the of the given mem_address value
- total_trans – specifies the total number of the data words for writing into the controllers' memory during the entire transaction
- sent_data – dada, forwarded by the application
- error_check – can have CRC or BCC
- receive_data – the driver forwards these data to the application
- nak – the number of the NAK packets
- enq – the number of the ENQ packets
- time_out – time interval for timeout
- mode – operating mode
- comm_status

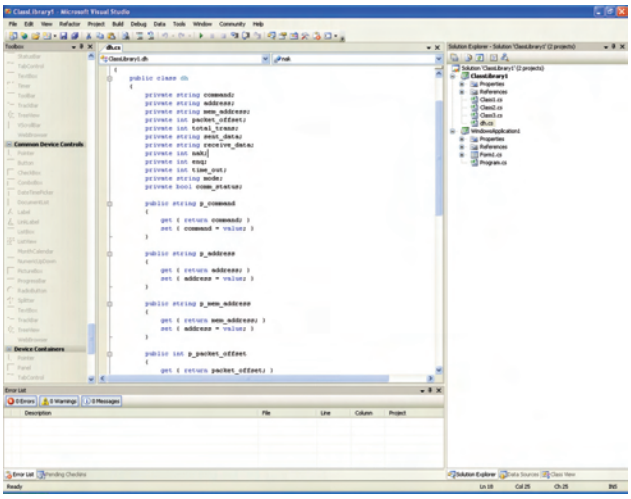

Fig. 4. The example of the class code

The class also contains the method "communicate" that is used by the application for forwarding the request for establishing communication with the controller.

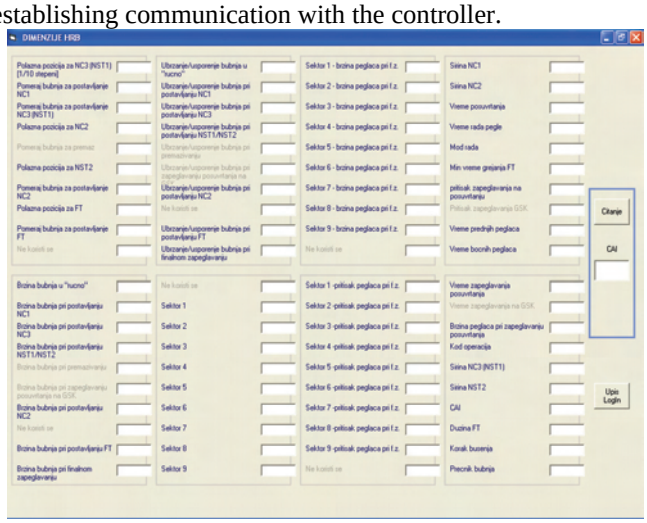

Fig. 5. Writing the dimensions parameters

The procedure in which the class is used for forwarding the data between the HMI application and the controller is given next:

- A new object with a dh class is created in the application
- The appropriate class properties are set – depending on the command properties, some of the properties are not necessary
- In order to inflict the controller to send the command, the method "communicate" must be invoked
- Coding the memory address, generating the TNS value, calculating CRC or BCC field and assembling the entire packet is performed within the class frame
- The class itself performs the communication toward teh controller
- When class extracts all corresponding data, it sets the "receive_data“ property
- After a certain amount of time the application reads the "receive_data“ and "comm_status“ properties
- If the property "comm_status“ is set to "True“ value, depending on the program logic, the property "receive_data“ is used further
- If the property "comm_status“ is set to "False“ value, the property "receive_data“ is not used further, and depending on the program logic, the command could or could not be repeated.

VI. CONCLUSION

Developing the unique own communication driver for the communication between PCs and PLCs in the machine control HMI interface design process enables better diagnostics, good real-time response of the machine and gives larger independence comparing to the use of the communication application offered by the machine manufacturer. The own solution and its integration into the HMI application increases the time needed for the design of the entire system, but at the same time the prize of developing the own driver much lower than the prize for buying the entire solution and no addition license expenses are necessary for each application installed later. The paper describes the communication model that uses asynchronous serial communication, and one practical realization of writing data obtained in a PC and needed for machine control, into the controllers' registers, and reading data needed for additional PC analysis or for updating the data shown on the screen.

REFERENCES

[1] -, “DF 1 Protocol and command set – Reference Manual”, Publication 1770 – 6.5.16, Allen_Bradley, Milwaukee, USA, 1996.

[2] Andrew S. Tanenbaum, “Computer Networks”, London, Prentice Hall PTR, 2002.

[3] -, “Data Highway or Data Highway Plus Asynchronous (RS-232-C or RS-422-A) Interface Module” User Manual, Allen_Bradley, Milwaukee, USA, March 1989.

[4] Anthony Chiarella, “Networks in Cisco and Microsoft technology (in Serbian)”, Čačak, Computer library, 2005.

[5] -, “Data Highway Plus and DF1 Communication Protocols”, Allen_Bradley, Milwaukee, USA, 2004.

[6] -, “Allen-Bradley DF1 Serial Communication Interface API”, DASTEC Corporation, 2003.

[7] Michael Halvarson, “Visual Basic.NET Step by Step”, CET Computer Equimpent and Trade, 2002.