Diagnostic Functions Embedding in Industrial Control Systems

Stanimir D. Mollov[1], Georgy Sl. Mihov[2], Ratcho M. Ivanov[3], Stoyan N. Jilov[4]

Abstract – **In the paper problems of diagnostic functions embedding in industrial control systems have been discussed. The idea for co-operation between control functions and functions related with loading, diagnostics and adjustment of the software in a single communication channel has been developed. The realization of this idea requires some conditions to be determined. In respect on these particular conditions, the rules for industrial networks building with embedded diagnostic functions are defined. Possibilities for existing conflict situations have been analyzed. The variants for their overcoming are offered.**

Keywords – **Industrial control system, Programmable Logical Controller, communication channel, debugger, networking.**

I. Introduction

One of the main features of Programmable Logical Controllers (PLC) or Industrial Controllers is a capability of networking. Usually each PLC is provided with a primary local area network trough which it is able to exchange information with controllers of the same rang and with the central computer [1].

However, in many applications, it is necessary to support secondary communication channel, which is used for diagnostic purposes. The main diagnostic function is software adjustment and diagnostics. Usually the secondary communication channel is different kind from the primary. More over, different channels may support different physical and logical levels.

The family controllers SIMATIC of SIEMENS is an example for PLCs with divided diagnostic and control functions (fig.1). Functions of loading and adjustment of the software are accomplished by Multi Point Interface (MPI). This interface is specially developed for connection between PLC and programmable station and may serve up to 126 correspondents. Control functions are accomplished by another interfaces: ProfiBus, Industrial Ethernet and AS-Interface. Usually these interfaces are built as additional modules [2].

MODICOM controllers produced by Telemechanique use similar strategy. These controllers are provided by an embedded primary interface, which is used for loading and adjusting of the software. However, this interface is Point- to-Point kind and does not have a capability of networking. Modbus interface is used for control functions between the same type of controllers.

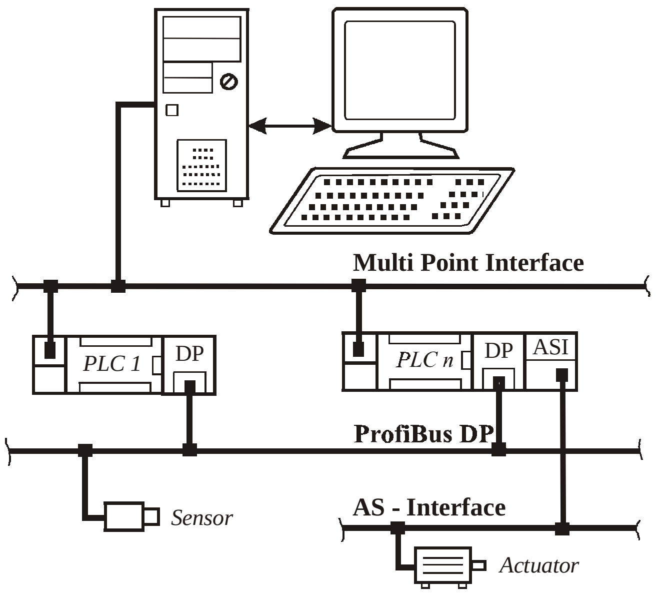

Fig.1 Communication options via the integrated interfaces of the SIMATIC

In the present paper, a solution is applied where the primary and the secondary communication channels are united in just one common channel. This solution allows controllers, having only one communication port, to use the control information network for loading, adjusting and diagnosing of the software. For this purpose, the functions for loading of the operation system and for loading, adjusting and diagnosing of the software have to be realized via the common channel, using common communication environment for different protocols.

II Control Information System

Functionally, the control information system may be divided into two main parts. The first part is the control part and it consists of a personal computer. The personal computer may combine the functions of an operation station and of a diagnostics station as well. According to those purposes, the

[1]Stanimir D. Mollov is with the Faculty of Electronic Engineering, and Technologies, TU – Sofia, 1797, Sofia, Bulgaria, E-mail: smollov@abv.bg
[2]Georgy Sl. Mihov is with the Faculty of Electronic Engineering, and Technologies, TU – Sofia, 1797, Sofia, Bulgaria, E-mail: gsm@tu-sofia.bg
[3]Ratcho M. Ivanov is with the Faculty of Electronic Engineering, and Technologies, TU – Sofia, 1797, Sofia, Bulgaria, E-mail: rmi@tu-sofia.bg
[4]Stoyan N. Jilov is with the SPV Ltd. Rakovsky 135, 6000, St. Zagora, Bulgaria, E-mail: spv@abv.bg

personal computer is mainly characterized with a powerful processing unit, and with embedded interfaces RS232 and USB. Thanks of the powerful processing unit of the computer, the main part of the network's protocol have to be realized in the computer environment. At the same time, a suitable hardware module has to be used to connect the embedded interfaces to the industrial interface.

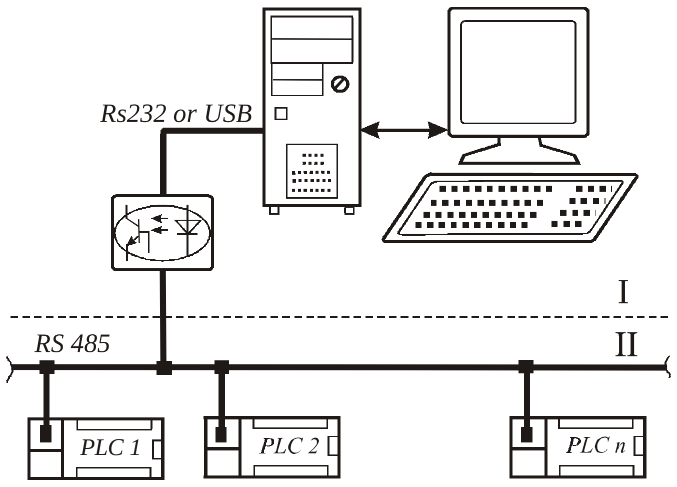

Fig. 2. Control information system

The secondary part of the control information system is a group of slave modules. The group includes intelligent sensors, PLCs and intelligent actuators. They are characterized by low power processing units. In most cases they are provided an embedded serial communication interface (SCI). This interface allows the network's protocol to be built easier and the connection to the industrial network to be realized directly via a suitable hardware adapter.

For the presented control information system is applied so called centralized management. This management is characterized with a request sending from the master station to the some slave station. By this way, the task of the slave module is to identify the request from the master station and to replay at a fixed time interval. The master station, considering the specified necessity defines the order of requests.

A. Hardware realization of a communication environment

The communication of the presented control information system is based on a serial communication interface, which is embedded in most controllers. To achieve maximum noise immunity it is necessary to use a differential transmission and a galvanic isolation. In the present case, the physical standard RS485 is used for a communication between correspondents. The using of a half duplex line requires an additional control signal for the data direction exchange.

B. Software realization of a communication environment

Due to the high noise level in the communication environment, it is necessary the transferred data to be blocked in comparatively short packets. Single data packet of the local area network has the following structure (fig. 3):

1. *SOH* – 'start of header' (1 byte). This is the leading symbol, which indicates the start of the data block transmission. There are two types of leading symbols: *SOH1* (start of message transmission) and *SOH2* (start of data block transmission). The first symbol is located at the start of the message. The second one is located at the beginning of the second and the following blocks of the multi-block message.
2. *RCV* (1 byte). It indicates the address of the message receiver (destination address).
3. *TRN* (1 byte). It indicates the address of the message transmitter (source address).
4. *COL* (1 byte). It may have value 0 or 0xFF, which is alternatively changed each other in the multi-block message transmission.
5. *LENGTH* (1 byte). It indicates the block length, e.g. the number of bytes in the informative part.
6. *DATA*. This is an informative part and can be sized from 0 to 255 (0xFF).
7. *CKSUM* (1 byte for a check sum). It is the least byte of the sum of all transmitted bytes, from *SOH* to end of *DATA*. It is used to check the block receive accuracy.
8. *EOT* (1 symbol 'end of transmission'). It indicates the end of the data block transmission. There are two types of end symbols: *EOT1* (end of message transmission) and *EOT2* (end of data block transmission). The first symbol is located at the end of the last block of the message. The second one is located at the end of the first and following blocks (except the last block) of multi-block message.

Message

SOH	*RSV*	*TRN*	*COL*	*LENGTH*	*DATA*	*CKSUM*	*EOT*

Reply

ACK	*Or*	*NAK*

Fig. 3 Transmitting a single data packet

After the transmitter sends the message, the receiver sends back a positive or negative acknowledges (fig.4). Positive acknowledge (ACK) means that the transmission had finished successfully and the calculated checksum equals to the received checksum. Negative acknowledge (NAK) means that the transmitter have to resend this message, because an error occurs. All actions in the local area network have own protected times. If the transmitting station had not received the acknowledgment at a fixed time interval, it repeats the transmission of the current data block. If the acknowledgment from the receiving station was positive, but it had not received, due to the noises in the network, the transmitting station have to repeat the same message. By the value of the *COL* field of the message the receiving station understands that the transmitting sequence is destroyed. The sending a *BREAK* signal causes the transmission interruption. All users, including the sender recognize the *BREAK* signal like an error in the format of the transmission data. After that, all users are listening to the line continuously and start transmitting a message only when the line is not busy. The time for one symbol's transmission is limited. The users of the network will know, that the line is busy after the time of a one symbol's transmission.

A conflict situation is possible to appear when more than one station from local area network start transmission at the same time. Each sender listens to the line (receives its own transmission) and compares the sent and the received information. In case of a difference (someone transmits over the line too) the sender interrupts the transmission and sends a *BREAK* signal to inform the others users for the conflict situation. After that each user continues to receive, but does not start transmission before a certain time, that depends on user's address in the network. By this way the conflict situations in the local area network are solved.

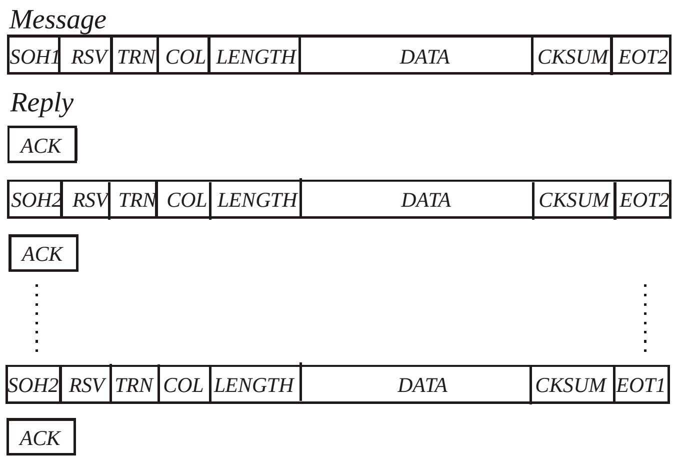

Fig. 4 Multi-block message transmission

III. Program Environment for Adjustment and Diagnostics

One of the main software tools for developing, adjustment and diagnostics of the micro processing systems is so called Debugger. The debugger's quality depends on used system resources. The main functions of the debugger are: program loading and executing, including break points, executing a single instruction, disassembling, reading and modification memory and registers.

One part of debugger functions is related with the specific of the central processing unit. The specification depends on the number, the kind and the structure of central processing unit registers as well as on the addressing mechanism.

Another part of debugger functions is related with the specific hardware realization of the controller. There are functions, which provide an access to the hardware resources. Most important of them are: inputs reading, writing and modification of controller's outputs, access to keyboard, indication, etc.

The adjustment and the diagnostics may be accomplished either by the control panel or by the terminal station via communication environment. Due to the limited information, which could be shown on the control panel, its application is limited. The terminal station, used for adjustment and diagnostics is built on a personal computer, as a rule. Using appropriate software, it can provide a full control over the slave controllers and also it is able to execute full or particular changes of the controller's applied program and parameters.

IV. Diagnostic Functions of the Information System Based on SPV-4xx Industrial Controllers

The main idea, developed in the building of an industrial network with SPV-4xx, is that all operations, related with control, adjustment and diagnostics, to be performed in the same network. The realizing of the idea requires:

– the control program and the debugger to be switched;

– the control program and the debugger to work independently.

To perform these conditions, it is necessary the debugger to be provided with own network driver and to does not use the network driver of the control program.

The industrial controller SPV 4xx is built on microcontroller MC68HC11. For the 'first time' loading the controller is set in Bootstrap mode. In this mode a small Boot program is enabled, which is located in its internal ROM memory. After the microcontroller is run, this program initializes the serial communication port, receives a program up to 256 bytes throw which and writes them in the internal RAM. The manufacture requirements, defined for bootstrap mode, must be strongly kept. They include data format and transmission speed. The bootstrap loading uses binary format. After each received byte, the microcontroller sends an acknowledgment. These special futures must be taken into account when the special software for the programmable station is developed.

After the bootstrap loading, the program in the internal RAM is started. This program has to provide the followed possibilities:

– the SCI interface to be initialized, if it is necessary;

– the execution operation cod of the debugger to be received via the network and to be loaded into the FLASH memory of the microprocessor system.

Including diagnostic functions in the network requires keeping exact rules, e.g. so called 'discipline of work'. These rules determine the conception: each correspondent in network has to have an own name. The main station has the name 'A'. The control information part of the information system is organized between controllers used capital letters from D to Y as a name. The controller, which enters into diagnostic functions, is identified as a controller with name 'C'. Such controller cannot perform control and information functions at the same time. The name 'Z' is reserved for future applications, like a calibration function controller.

In this paper the diagnostic functions are minded as an adjustment and a diagnostics of the applied software. If the diagnostic functions have to be performed at same time when the user program is run, they should be embedded in it.

The debugger commands are different from commands of the control information program. The main station may run them but it is not desirable. Especially for diagnostics function using, a new station is included in the network. It is called 'service station' and has got the name 'B' (fig. 5). Run in the diagnostic mode controller responds to the one, which asks him. From security point of view, just one controller in the network is allowed running in the diagnostic mode.

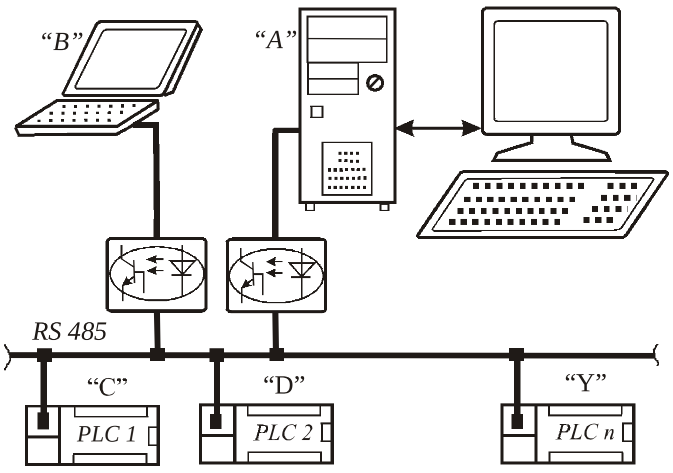

Fig. 5 Conception for correspondents naming

A proper mechanism for switching between the debugger and the applied program is built. The hardware realization of this mechanism consists of the program memory division in two banks (fig. 6). These banks are located in the same area of the addresses. The user program is loaded is loaded in the first bank, while the debugger is loaded in the second one. The switching between the banks is provided by a special command in the network protocol. By this way, the simultaneous running of the debugger and the applied program is excluded.

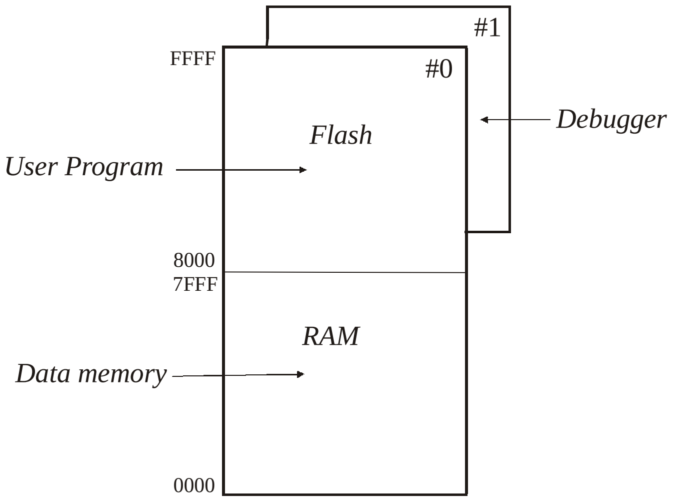

Fig. 6 Address memory map

The bootstrap loading of some controller may provoke a conflict situation if it is done using the network, at the same time, when the network is used by another controller for an normal (control and information) work. It may occur when some controller, loaded in advance, is laid and caught the information for the bootstrap loading as a valid for it. There are some ways to avoid this conflict situation:

1. The information for original loading is analyzed and checked for a stream of symbols sequence, which is the same as the sequence *SOH* and *RCV* (fig. 3). If there is such sequence, it must be avoided. Mostly, it is done including an empty operation cod between these symbols.

2. If it is not possible to include an empty operation cod in the program of the bootstrap operation code, it is necessary to exclude the controller and to do the bootstrap loading out the network.

Another way to avoid generally conflict situations is the bootstrap loading to be controlled by additional device (mediator). Ordinary it is also a microcontroller, which listens the network information. If the mediator detects the command for the bootstrap loading, it disconnects the controller from the network, gets the bootstrap information and translates it to the controller's processor. In this case, the bootstrap information has to be formatted according the network protocol.

V. CONCLUSIONS

In the paper the idea for co-operation between the operations related with loading, adjustment and diagnostics software in common communication channel is developed. The necessary conditions for its realization are discussed. As a result, a conclusion that the control program and debugger must have their own network protocol is done.

Including the diagnostic functions in the industrial network requires specific rules to be kept. They bring a conception for the naming of controllers. In additional, a protected mechanism, which does not allow the debugger and the control program to be run simultaneous, is created.

It the present paper the possibility for conflict situations are discussed. The variants for their solving are offered.

REFERENCES

[1] Mihov, G., E. Dimitrov, S. Jilov, A. Kostadinov. Composing of Different Local Area Networks for Industrial Controllers on Common Physical Layer. XXXVII International Scientific Conference on Information, Communication and Energy Systems and Technologies ICEST '2002. vol. 2 pp. 406-409, October 1-4, 2002, Niš, Yugoslavia.

[2] SIMATIC Programmable Controller System Manual, SIEMENS Berlin, Siemens Aktiengesellschaft 1999, C79000-G7076-C230-02

[3] Mihov G., I. Tashev. Industrial Controller For Discrete Manufacture National Scientific Conference "Electronics '96", book I, 31-36, Sozopol, Bulgaria, September, 1996.

[4] Dimitrov E., G. Mihov, I. Tashev, M. Mitev. Local Area Network for Industrial Controller. The International Scientific Conference ENERGY AND INFORMATION SYSTEMS AND TECHNOLOGIES. vol. III, pp. 608-613. Bitola, Macedonia, June 7-8, 2001.

[5] http://www.schneiderelectric.com/