DEVELOPED ENVIRONMENT FOR ADJUSTMENT AND DIAGNOSTICS OF SPV INDUSTRIAL CONTROLLERS

Stanimir Damyanov Mollov

Faculty of Electronic Engineering and Technologies, TU – Sofia, 1797, Sofia, Bulgaria,

E-mail: smollov@abv.bg

Keywords: adjustment, diagnostics, debugger, controller

The paper presents a developed environment for adjustment and diagnostics of SPV industrial controllers. The developed environment is a part of the software for personal computer. It provides all operations related with loading, adjustment and diagnostics of the controller's software and may communicate both control and debugger programs embedded in each of SPV industrial controllers. To accomplishing the communication between a personal computer and controllers is used a hardware-software network driver, which realizes all function related with receiving and transmitting the messages via industrial network. The developer environment is compatible with all Windows operating system. It is realized on high level program language Visual C++.

1. INTRODUCTION

Nowadays, the tendency in automated systems is to be based on personal computer as applied so-called **centralized management** (fig.1). This management is characterized with a request sending from the master station to the some slave station. By this way, the task of 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.

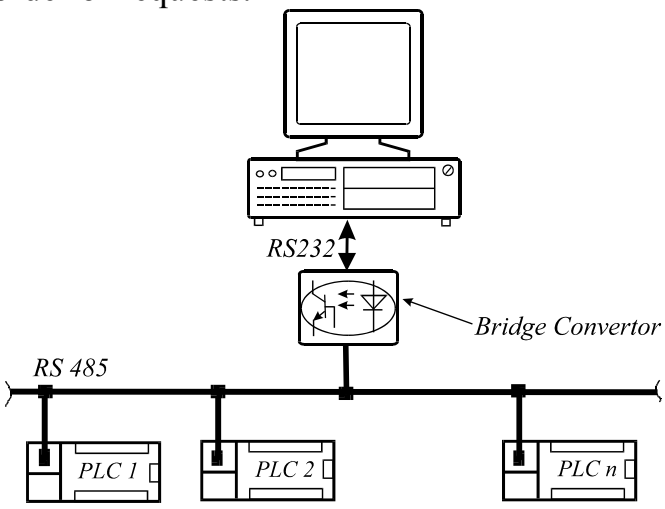

Fig. 1 Centralized menagement

Usually the main station consists of a personal computer, which is used as for acumodating and processing the information, as well as assistance instrument for adjustment and diagnostics of the slave modules. In the present paper the developed

environment for adjustment and diagnostics of SPV industrial controllers have been discussed. The developed enviroment is compatible with all Windows operating system and it is realized on high level language Visual C++.

2. DIAGNOSTIC FUNCTIONS EMBEDED IN SPV CONTROLLERS

The main idea, developed in the building of an industrial network with SPV-controllers, is that all operations, related with control, adjustment and diagnostics, to be performed in the same network [1]. For this purpose the software of the controllers must include: control (user) program and a program for adjustment and diagnostics – debugger. The control program and debugger must work. To perform this condition, it is necessary the debugger to be provided with own network driver and to does not use the network driver of the control program.

Including diagnostic functions in the network requires keeping exact rules. 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.

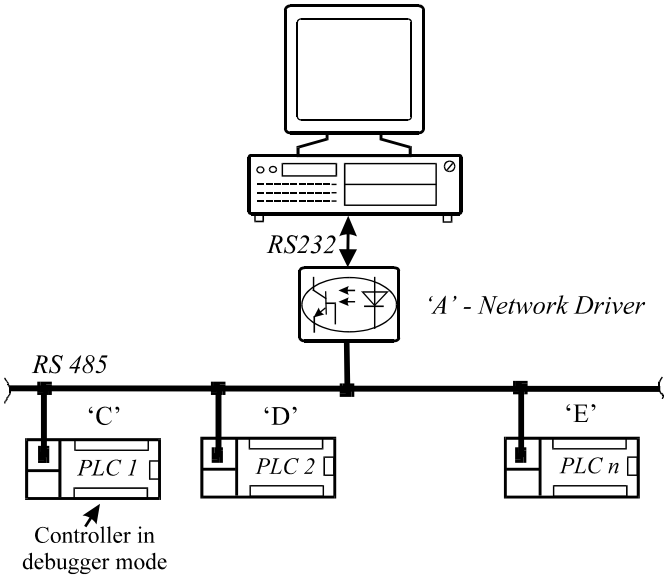

Fig. 2 Conception for correspondents naming

The developed environment for adjustment and diagnostics as a part of the software for the personal computer should communicate both control program and debugger embedded in each one of SPV controllers. To accomplish the communication between master station and slave stations in the industrial network is used a hardware/software network driver (fig.2). The network driver realizes all functions of receiving and sending of the messages, protecting times, repeating of the message when an error occurs and solving the conflict situation in the industrial network. By this way the hardware/software network driver in fact is the master station in the industrial network.

3. DEVELOPED ENVIRONMENT FOR ADJUSTMENT AND DIAGNOSTICS

The developed environment for adjustment and diagnostics realizes two kinds of operations. On one hand these are the operations of loading, adjustment and diagnostics of the applied program. On the other hand these are the operations for processing of the parameters of the control program. The place of the developed system into the composition of control informational system is shown on fig. 3.

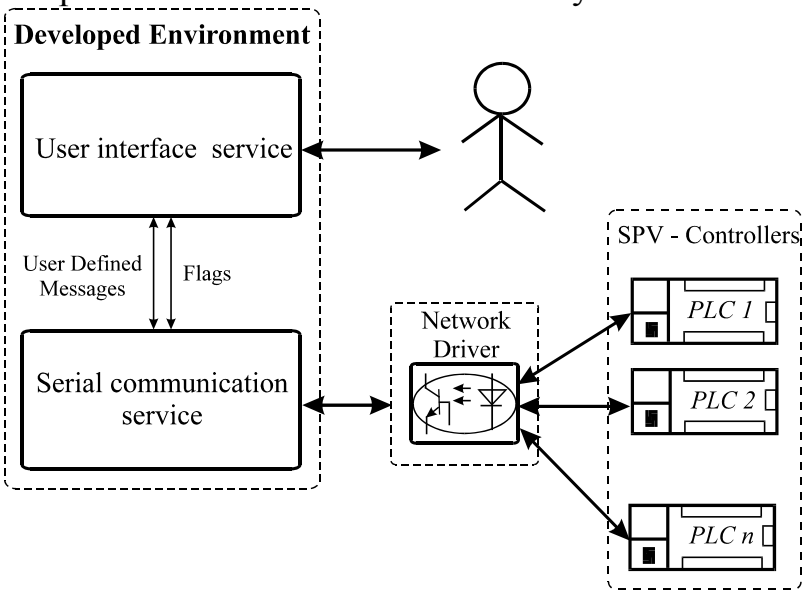

Fig. 3 Developed environment and its two processes

Developed environment for adjustment and diagnostics is realized as used so called multi-thread method. This method allows several processes to work at the same time under management of the computer operating system. In this case are used only two simultaneous worked processes. The first one (main process) services the communication with operator. The second one is managed from main process and services the serial communication interface. This imposes existence of a communication channel between these two processes. In the present work the communication channel is consist of flags and messages.

4. FUNCTIONS FOR SERIAL COMMUNICATION SERVICE

The developed enviroment is realized on high level language Visual C++ and used the standart API function for serial communication. These functions are two kinds:

- The functions for serial port configuration;
- The functions for communication via serial port.

4.1 Functions for serial port configuration

Creating a port handle

The serial port's handle is a handle that can be used to access the object of serial port. The function that is used to create the serial port handle is the **CreateFile** function. The following code shows the function that is used to create a handle:

```
hComm = CreateFile("COM1",                  // Specify port device: default "COM1"
            GENERIC_READ | GENERIC_WRITE,   // Specify mode that open device.
```

```
        0,                          // the devide isn't shared.
        NULL,                       // the object gets a default security
        OPEN_EXISTING,              // Specify which action to take on file.
        0,                          // default.
        NULL);                      // default.
```

Restoring a configuration

The restoration of serial port configuration is getting current configuration at control device. The configuration of serial port includes parameters that are used for setting a serial communications device. The **GetCommState** function is used to get the current device-control and then fills to a device-control block (a DBC structure) with the current control settings for a specified communications device:

```
GetCommState(hComm,&m_dcb)
```

Modifying a configuration

```
m_dcb.BaudRate = CBR_9600;          // Specify the baud rate
m_dcb.ByteSize = 8;                 //Specify the number of bits for byte
m_dcb.Parity = NOPARITY;            //Specify parity
m_dcb.StopBits = ONESTOPBIT;        //Specify stop bits
```

Storing a configuration

The next step is the storage of the new configuration that is modified already into device control. To store is used **SetCommState** API function. This function configures a communications device according to the specifications in a DBC structure. The function reinitializes all hardware and control settings.

```
SetCommState(hComm,&m_dcb)
```

Setting a Time-Out communication

The final step in serial port opening is setting communication Time-out by using the **COMMTIMEOUTS** data-structure and calling **SetCommTimeouts** function.

```
m_ComOuts.ReadIntervalTimeout = 20;          //Specify time-out between charactor for receiving.
m_ComOuts.ReadTotalTimeoutMultiplier = 10; //Specify value that is multiplied by the requested number of bytes to be read
m_ComOuts.ReadTotalTimeoutConstant = 10;//Specify value is added to the product of theReadTotalTimeoutMultiplier member
m_ComOuts.WriteTotalTimeoutMultiplier = 10; // Specify value that is multiplied by the requested number of bytes to be sent.
m_ComOuts.WriteTotalTimeoutConstant = 10; //Specify value is added to the product of the WriteTotalTimeoutMultiplier member
SetCommTimeouts(hComm,&m_ComOuts)       // Set the time-out parameter into device control
```

4.2 Functions for communication via serial port

Sending data

The **WriteFile** function is a function used to send data in serial port communication.

```
WriteFile(hComm,            // handle to file to write to
        outputData,         // pointer to data to write to file
        sizeBuffer,         // number of bytes to write
        &length, NULL)      // pointer to number of bytes written
```

Receiving data

The **ReadFile** function is the function that handles reading data via serial port.

```
ReadFile(hComm,             // handle of file to read
```

```
inputData,      // handle of file to read
sizeBuffer,     // number of bytes to read
&length,        // pointer to number of bytes read
NULL)           // pointer to structure for data
```

5. DEMONSTRATION PROGRAM FOR WORK WITH SPV CONTROLLERS

The demonstration program provides full control over SPV industrial controllers, which include:

- Loading, adjustment and diagnostics of the software of the controllers;
- Management the work of the controllers (start and stop);
- Change the parameters and the name (address) of the controllers;
- Cyclical inquiry and displaying the data from all controllers in the network (so called an information system).

The result of the demonstration program work is shown on fig.4. In this case, the slave controller works in the debugger mode. The command, which may be executed in this mode are:

- loading the user program from a file;
- writing the user program or a memory area into file;
- reading and displaying the memory area;
- memory modification;
- make a copy of the memory area.

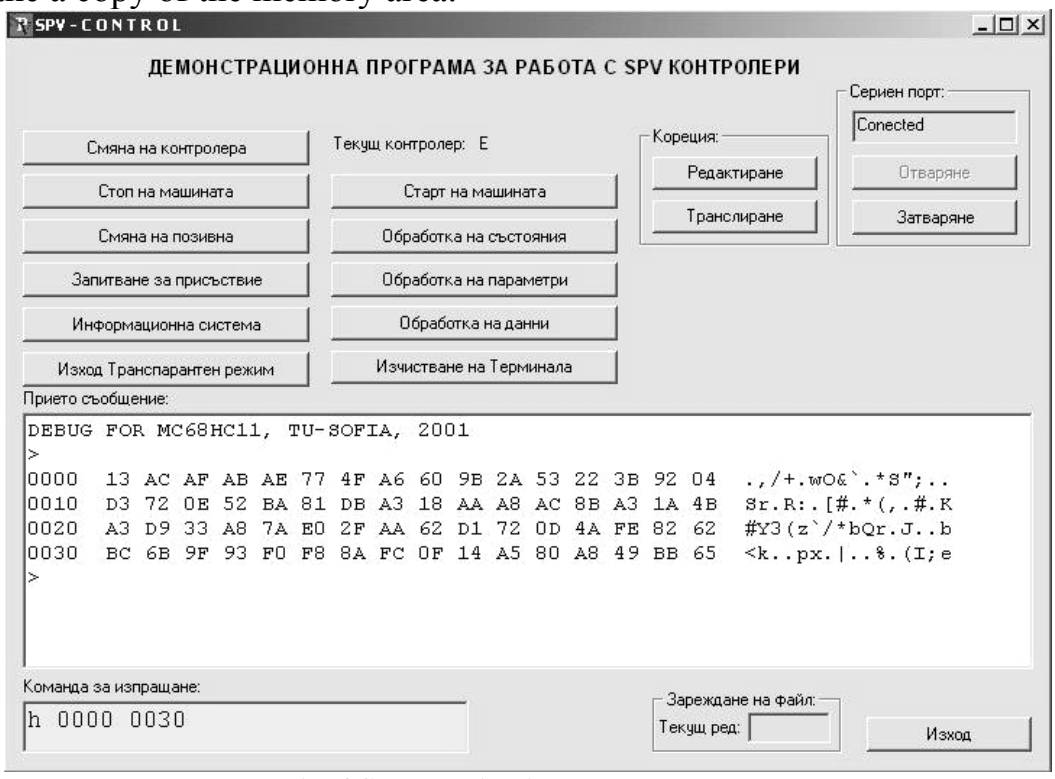

Fig. 4 Slave station into debugger mode

The commands are written in the bottom edit box. After pressing the ENTER key each command is included in the packet structure of the previously defined protocol [2]. After this the message is sending via serial port. The received answer is displayed

in upper edit box. It is consist of only usefully part without service part of the received message

The result of the work of the demonstration program as an information system is shown on fig.5. In this mode the main station in the network cyclical inquiring and displaying the data from the all controllers in the network.

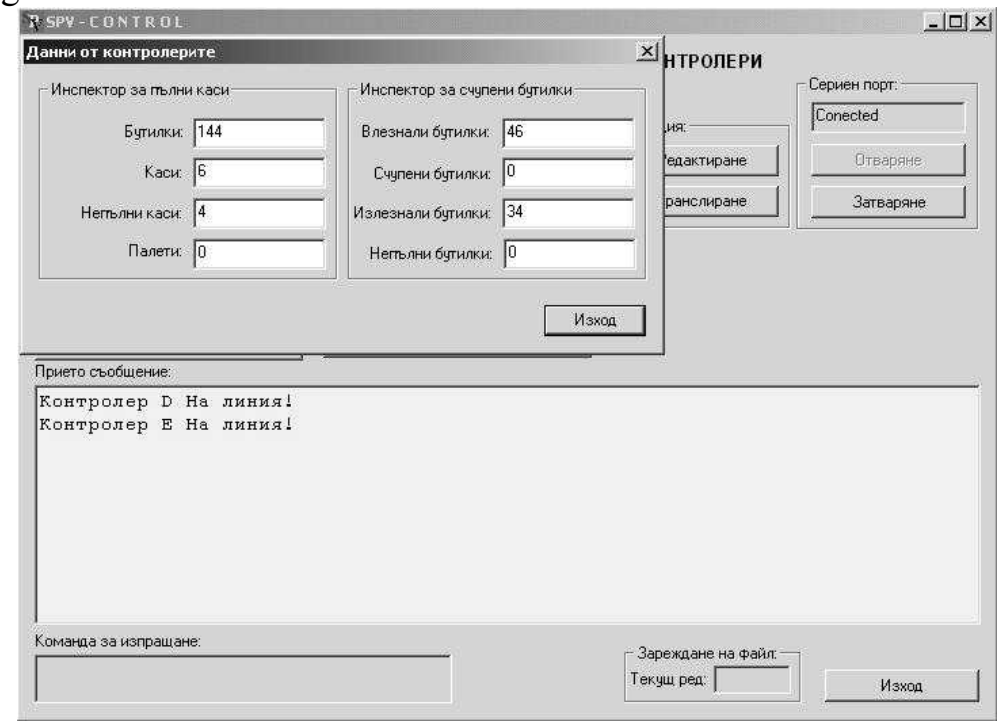

Fig. 5 The information system

6. CONCLUSION

The developed environment for adjustment and diagnostics of SPV industrial controllers can provide a full control over the SPV slave controllers. It is compatible with all Windows operating system after Windows 95. By using the environment for adjustments and diagnostics, full or particular changes of the controller's applied program and parameters can be done.

7. REFERENCES

[1] Mollov, S., G. Mihov, R. Ivanov, S. Jilov,. *Diagnostics Functions Embedding Industrial Control System*. Composing of Different Local Area Networks for Industrial Controllers on Common Physical Layer. XXXIX International Scientific Conference on Information, Communication and Energy Systems and Technologies ICEST '2004. June 16- 19, 2004, Bitola, Macedonia.

[2] Dimitrov, E., G. Mihov, I. Tashev, M. Mitev, *Local array network for industrial controller*, Proceedings of the Int. Scientific Conference ENERGY AND INFORMATION SYSTEM AND TEHNOLOGIES, . vol. 3 pp. 608-613, June 7- 8, 2001, Bitola, Macedonia.

[3] www.msdn.microsoft.com

[4] www.codeproject.com

The paper has been reviewed by Associate Professor Ph.D. Emil Dimitrov from department of Electronics, Technical University – Sofia.