SMALL LOCAL NETWORK FOR MICROPROCESSOR CONTROLLERS

Tsvetan Petkov Shoshkov
Georgy Slavchev Mihov, Ventsislav Draganov Manoev

Departmen of Electronics, Technical University – Sofia, Kl. Ohridsky str. No 8, bl. 1, 1797 Sofia, Bulgaria, phone: +3952 965 3241, e-mail: bladejuven@gmail.com; gsm@tu-sofia.bg;

This article deals with the matter of developing of a driver for small local field network for microprocessor controllers in information controlling systems. As a basis it is used the structure of a small local network driver, developed in Technical University – Sofia. It is processed to work according to a standard protocol under the specification of Modbus. Thus the existing network is made universal and there is a possibility for connecting it in other systems, using the open standard of the Modbus specification. The existing logical rules for the organization of the user and service programs of the controllers and the rules for communication are not changed. The driver is written in C programming language that makes it easy for using by different microprocessors. The driver is with simple structure, it is easy for using and optimization.

Keywords: Communication, Controllers, Local network, Modbus

1. INTRODUCTION

The complex automation of a manufacturing process requires the organization of information controlling systems by communication between controllers in a local network. Field local networks are used when the information exchange must be accomplished in real time mode [1]. The most perspective for the future are the open specifications.

There is an information controlling system with a small field local network, developed in Technical University – Sofia [2, 3]. One of its applications is in inspecting of weight in the manufacture lines for bottling. For its local network it is typical that it allows connecting of 256 equal subscribers (senders). Each of them has the right to be a master and a slave. This conception requires a relatively complex organization of the protocol for information exchange and of the messages format.

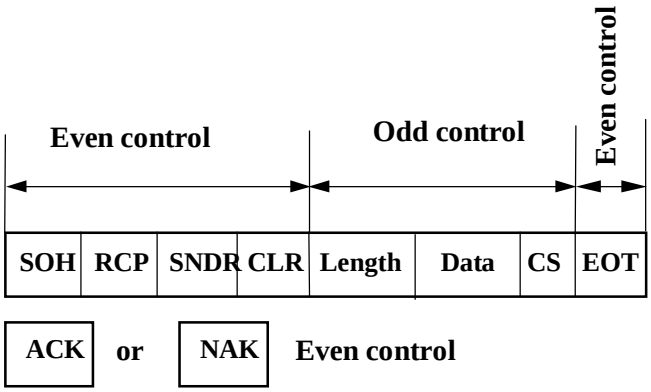

Fig. 1. Single block message transmission.

Schematically, the transmission of a single block message is shown on Fig. 1.

1. *Leading symbol*, 1 byte. It indicates the start of the data block transmission. There are two types of leading symbols – a start of message transmission (SOH) and a start of data block transmission (STX). SOH is used for the start of the message and STH is used for the

following blocks of a multi-block message.

2. *Receiver*, 1 byte. It indicates the address of the message's recipient. The number of the users is allowed to be up to 256.

3. *Transmitter*, 1 byte. It indicates the address of the message's sender.

4. *Block color*, 1 byte. It contains 00 or FF_{16} , which are changing alternatively when a multi-block message transmission is present. This is so called coloring of the block in "blue" or "orange".

5. *Block length*, 1 byte. It indicates the number of the bytes in the informative part. It does not exceed 256.

6. *Informative part*, 1 to 255 bytes data information.

7. *Checksum*, 1 byte. It is the less byte of the sum of the all transmitted bytes and is used to check the block receive accuracy.

8. *End symbol*, 1 byte. It indicates the end of the data block transmission. There are two types of end symbols – End of message transmission (EOT) and End of data block transmission (ETX). EOT is used for the last block of the message and ETX is used for each block (except the last block) of a multi-block message.

In order to have a difference between the system information and the informative part, the system information is transmitted with even control, and the informative part's bytes are transmitted with odd control.

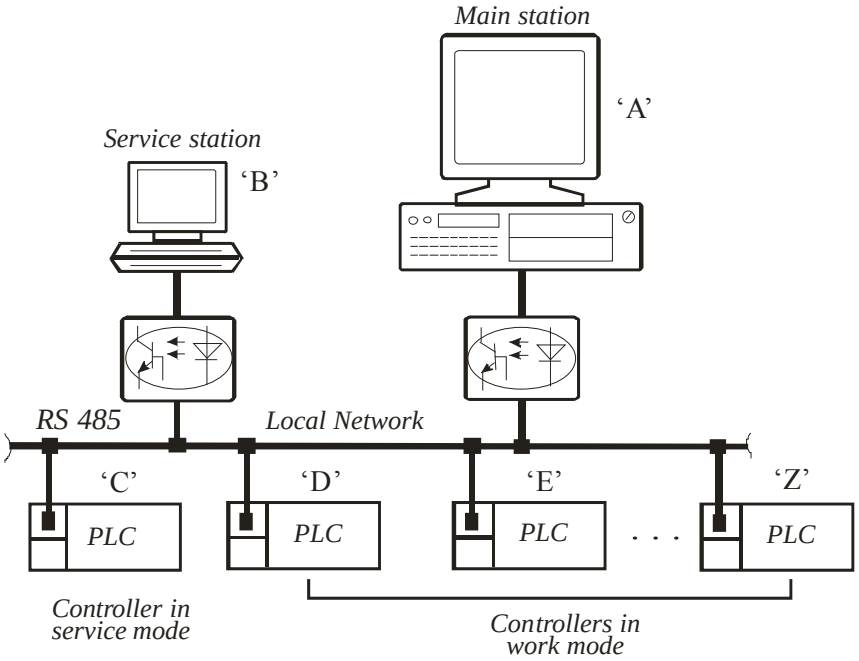

Fig. 2. Naming rules of the network subscribers.

The information system and the local network are developed, using the conception in [1, 5] that is modified for the particular purposes. Each controller can be identified in the local network by its name. Functionally the information system consists of two parts – controlling (working) and diagnostic. The controlling part is organized between controllers with names that are composed of the main letters of the Latin alphabet – from 'D' to 'Z'. A controller that enters a diagnostic mode is identified with name 'C', no matter what its name is. Such controller cannot perform

controlling functions in the same time. Only one controller in the network can be in diagnostic mode in a particular moment. 'A' is defined for a name of the main station.

The commands of the diagnostic program in the controllers are different from these of the working program. The diagnostic activities usually are done by a service staff, so there is an individual name 'B' for the service (diagnostic) station. In order to be possible the diagnostics to be performed by the main and the service station, the debugger program is made in a way that the controller in diagnostic mode to respond to the commands of the station that have sent the request. The concept of the rules for naming of the subscribers in a local network is shown on Fig. 2.

The information in the Data field is processed by the command interpreter. The first byte of this field is the command (functional code), that defines the task of the subscriber.

The program memory of the controllers, that forms the information controlling system, is divided in two banks (pages) as it is shown on Fig. 3. They take up the same space in the address area (the same logical address area). The debugger program is loaded in one of the two banks. The working program is loaded in the other. There is a particular command used to switch over to another bank. This segmentation discounts the possibility for the two programs to work in the same time.

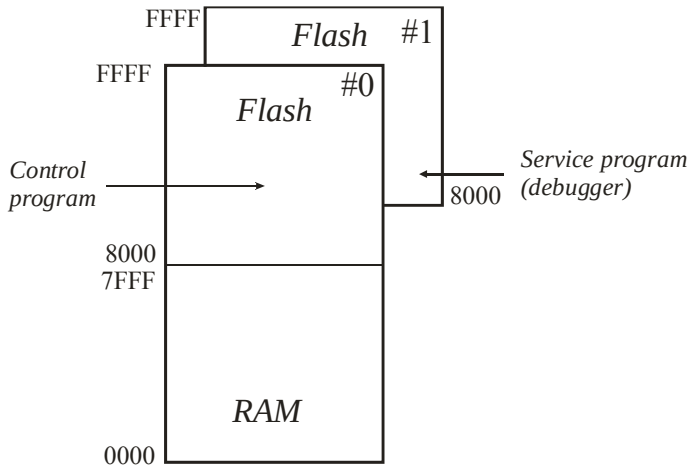

Fig. 3. Organization of the controlling and the service program.

In spite of its high reliability and simplicity, the created local network has a disadvantage – it is applicable in limited number of information controlling systems and it is not logically compliant with the standard field networks.

In order to cope with this problem the existing structure of the driver is processed to work under the standard Modbus. In the same time the organization concept of the network subscribers is saved.

2. DATA EXCHANGE PROTOCOL MODBUS

Modbus is a communication protocol, developed by Modicon systems [6]. The controller that requests information is called Master (main). The controller that supplies the information is Slave (subordinate). There is one Master device and up to 247 Slave devices in the standard Modbus network. Each one of them has an unique address from 1 to 247. Modbus is an open protocol, free of charge.

Each message in Modbus has the same structure (Fig. 4). The message consists of four elements. In the Modbus network the „conversation" is started only by the Master controller. The Slave controller launches a task and sends a response, depending on the content of the message.

Field	Description
Device Address	Receiver address.
Function Code	Code defining the type of the message.
Data	Data block with additional information.
Error Check	Testing check for errors in the communication.

Fig. 4. Modbus message structure.

The Modbus connection allows two basic modes of transmitting – ASCII and RTU (Remote Terminal Unit). The Modbus/ASCII messages are in ASCII format. The Modbus/RTU format uses binary coding. All subscribers of a Modbus network must use the same mode of data transmitting.

Each Modbus/ASCII message begins with the transmission of a symbol ':' (ASCII code 0x3A) and ends with a Carriage Return (CR) Line Feed (LF) – Fig. 5.

':'	Slave address	Function Code	Data	LRC	CRLF
1 byte	2 bytes	2 bytes	N bytes	2 bytes	2 bytes

Fig. 5. Message structures in ASCII mode.

In Modbus/RTU mode the transmission of each message is went before an empty time interval, longer than the time, needed to transmit 3.5 chars. If the receiver detects a „gap" more than 1.5 chars, it is considered that a new message is arriving and the receiving buffer is empty.

Slave address	Function Code	Data	CRC
1 byte	1 byte	0 - 252 bytes	2 bytes

Fig. 6. Message structure in RTU mode.

The first element of each message is the address of the receiver. This parameter consists of one byte information. The Modbus/ASCII address is coded with two hex chars. The Modbus/RTU address is only one byte.

The second byte of the message is the function code. It defines the task that will be launched by the Slave controller. When a Modbus receiver „responds", it uses the same function code as in the request. When an error occurs, the master bit in the function code is set to one. This way the Master controller is able to make difference between the correct and the wrong answer.

As a result of a received request there might be four different cases:

– The request is successfully received by the Slave controller and there is a valid response.

– The request is not received by the Slave controller and there is no response.

– The request is received by the Slave device with parity, CRC or LRC error. The Slave device ignores the request and doesn't send a response.

– The request is received without any errors but it cannot be launched by the Slave device because of another reason. The master bit of the function code in the response is set to one.

The Modbus standard uses preservatives for error detecting. While transmitting each byte of the message an odd/even check is made. The last block of each message is a CRC or LRC.

3. MODBUS COMMUNICATION DRIVER

The network driver is a part of the working program of a particular controller in the network. The driver is an instrument used to realize the transfer of information between the industrial network and the applied program of the controller. Its special feature is that it works as a separate task in interruption mode. In basics the network driver consists of an initialization part, a receiver and a transmitter. The receiver and the transmitter can be in one of the following statuses: wait, active, end of operation.

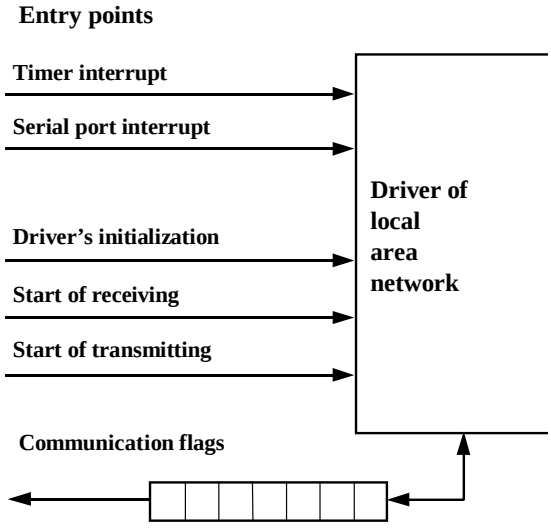

Fig. 7. Network driver entry points.

Timer interrupt. The counters used for realizing the pause time are modified in this entry point.

Serial port interrupt. The serial port interrupt makes a request when a symbol from the network arrives or when a symbol is sent into the network.

Driver's initialization. This entry point is called for an initialization of the serial port. After the initialization both the receiver and the transmitter are in inactive status.

The work algorithm of the driver is shown on Fig. 8. The driver observes the network for a pause bigger than 1.5 chars. This indicates the beginning of a new message.

The received message blocks are written in an input buffer - IBUFDT. If there are no errors while receiving, the information is transferred from the input buffer into a queue - QUEUEA. After that the operating system, that uses the driver, transfers the command message from QUEUEA into a command buffer - CBUF. Then the received command is analyzed and are executed the defined by the message tasks. Then a response is written in the command buffer and the information is transferred in a queue QUEUEB. Eventually, before the transmitting of the response starts, the information is transferred into an output buffer - OBUFDT.

The debugger program is a standard monitoring program. There is a personal local network driver in the debugger. Input-output commands of the debugger are transferred through the driver.

The personal network driver of the debugger is comparatively simplified, as the setting of the connected in the network microcontrollers is performed one at a time. It doesn't work in an interruption mode and handles the serial port by observing the flags in the status register (poling mode). The basic function of the debugger is to load the working program in the microcontroller.

4. CONCLUSIONS

The developed driver is tested in real conditions with controllers performing weight inspection. The Master controller is an IBM personal computer. Its existing driver is in the form shown on figure 1 (written in Assembler programming language). It is processed into Modbus/RUT driver, shown on figure 6.

The naming concept of the Slave controllers is saved and they have an address from 66 to 91, corresponding to the ASCII symbols from 'B' to 'Z'.

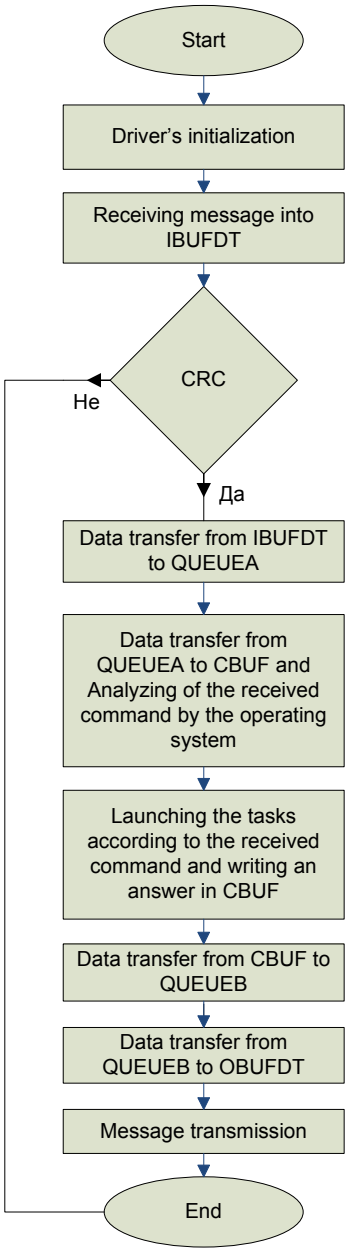

Fig. 8. Work algorithm of the local network driver

The messages consisted of many blocks are transferred by transmitting of every block as a separate message.

The first byte of the Data field is accepted to be the address of the sender. This gives the opportunity (out of the logical protocol of Modbus) the command interpreter of the Slave device to process a message, sent by another controller and to send a response to it.

2. REFERENCES

[1] Mollov, S. *Interfaces research about connection between devices and controllers in controlling systems*. Dissertation for educational and scientific degree 'doctor', Sofia, 2006, (in Bulgarian).

[2] Tashev, I. *Methods, devices and systems for gathering and processing of information.* Textbook for distance teaching, Technical University – Sofia, 1997, (in Bulgarian).

[3] Dimitrov, E., G. Mihov, I. Tashev, M. Mitev. *Local Area Network for Industrial Controller.* The International Scientific Conference EIST2001. vol. III, Bitola, Macedonia, June 7-8, pp. 608-613, 2001.

[4] Mihov, G., E. Dimitrov, V. Gunev. *Industrial Controller for Weight Measurement.* The International Scientific Conference EIST2001. vol. III, Bitola, Macedonia, June 7-8, pp. 649-654, 2001.

[5] Mollov, S., G. Mihov, R. Ivanov, S. Jilov. *System for Technological Data Collecting in Pipe-Lined Equipment for Beverage Industry*. mag. "Electrotechnica & Electronica", No 7-8, pp. 3-9, 2006, (in Bulgarian).

[6] http://www.Modbus-IDA.org