Proceedings of the 12th
KACC, October 1997

사출기를 위한 자동제어 시스템의 개발

Development of the automatic control system for projection machines

°Kaldy Domonkos*, 정상범**, 이천희**

* Technical University of Budapest ** Chongju University

** 청주대학교 전자공학과(Tel:0431-229-8448;Fax:0431-55-6392;E-mail:yicheon@soback.kornet.nm.kr)

Abstract : The industrial automation market encompasses all aspects of the discrete process in the manufactring industries. Nowadays many manufacturers are increasingly moving to increase plant capacity, and are trying to produce better quality at the lower costs.

We have studied a modern projection-automation system to achieve the communication network which makes connection among all of the porjection machines and connect them to a central computer. This communication system have to be very flexible and fast. Also we have studied the communication interface for the automatic production system including the hardware elements and the communication software.

1. Introduction

One of the most popular research targets is to make an automatic production control system. In our case the production line is composed of several machines. Every machine is a stand-alone entity with its own control system. These machines can work alone, all of the functions are controlled by itself.

But the production line requires a complex cooperation among the machines. We need a central computer, a central unit to control the whole line and the whole process, which can communicate with all of the machines and which can supervise it.

The production control system is composed of two parts : the communication system and the central computer program. The communication system includes hardware elements and communication programs.

2. Configuration of system

2.1 Environment of system

To find the best solution we had to divide the exercise into two parts. It is very easy because there are two almost completely independent problems. The first problem is how to **transport the data and the commands** from the Central Computer to the PLC and from the PLC to the Central Computer. The second problem is the **control of the PLCs**, receiving the status information from them and sending commands to them. The first, crucial question is how to make a communication-system operating among the PLCs and the Central Computer. After that we can create the supervisor program which runs on the Central Computer with a large database, and with automatic decision algorithms, etc.

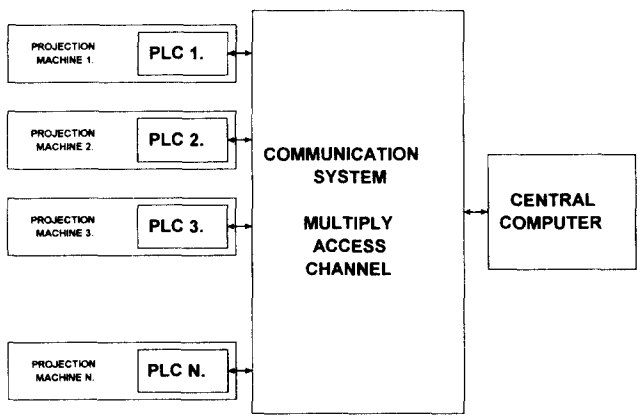

Figure 2-1 Block diagram of system

2.2 TOPOLOGY

There are three different types of topologies for the multiple access systems: star, ring, and common bus. The topology is almost independent from the used protocol type. The first point of view in our case is the system flexibility and the price. The star topology requires a large amount of devices compare to the common bus topology. Even though the ring topology is not too flexible, it could be a little bit complicated to expand it. We decided to use the **common bus** topology.

2.2.1 SERIAL BUS TYPE

There are several standard systems. We decided to use the **RS-485 line**. The RS-485 standard line has been already tested in the laboratory, and it shows the maximal competency in this application. It allows the maximum data rate to 10 megabit/s while maintaining the ability to drive up to 4000 ft and up to 32 receivers, with signal levels of -1.5 to -6 V and +1.5 to +6 V. It has two major advantages: the RS-485 is a differential standard, and allows the multidrop operation. Differential feature means that each signal is represented by a pair of wires, and the voltage difference across these wires is what is sensed at the receiver. This minimizes the effect of ground noise or the voltage drop along the signal leads. The signal arrives at the receiver with less noise since the ground wire noise is irrelevant, and with larger useful amplitude. The multidrop operation means that more than one transmitter can be connected to line -though not active- at a time. Although only a single transmitter can be operating at any instant, the standard specifies that up to 32 transmitters can be connected on the differential wire pair at the same time. This is done by making sure that line drivers for RS-485 can be disabled, using a single control pin on the line driver IC. When disabled, the line driver goes to a high impedance mode where it draws virtually no current and produces neither a binary 0 or 1. As far as the system is concerned, these disabled drivers are not present on the line. The RS-485 standard is useful when many user points must be connected together, sending data with a single user at any time. To ensure that one and only one sender is active, the system circuitry must take special actions to instruct the users as to whose turn it is to become active. This forms the basis for the backbone of a network where messages can be sent by any user to any of the other users. Receivers who have no need to listen to the message can simply ignore the data since there is no electrical conflict in having many listeners. The only conflict occurs when there are two or more talkers at the same t

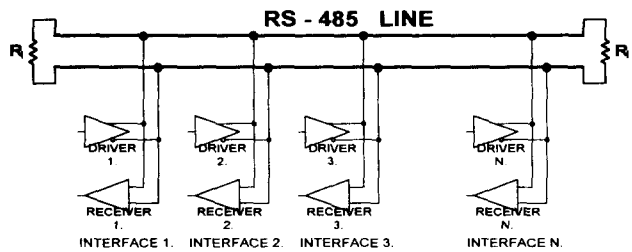

Figure 2-2 RS-485 line

For the RS-485 standard there is a special transceiver IC on the market, and it has been also tested in the laboratory. Its model number is : **75176**. It has signal inputs/outputs and enable pins on the "interface-side" and the bus connection (2 pins) on the RS-485 side. It requires more only single +5V and the GND. In addition, the device are handled by a thermal shutdown circuit, which forces the driver outputs into a high impedance state. For more information see the appendix.

2.2.2 FUNCTIONS OF INTERFACES

In our system the interfaces have three important functions: to make contact with the other interfaces, and with the RS-485 bus, to make contact with the PLC (or in the case of the master: with the Central Computer), buffering all of the received data. Almost all of the functions of the interface is interrupt controlled. The serial receiver routines must to place the received data at an auxiliary store.

Every interface is a stand-alone entity, that is they are operating alone. We have to control them from the Central Computer with special messages, named system control messages. Every interface has a command interpreter and a System-Buffer for that kind of messages. See the figure 2 - 3 and 2 - 4.

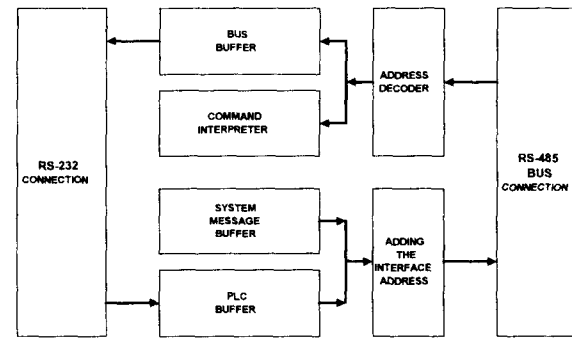

Figure 2 - 3 Functions of the slave interface

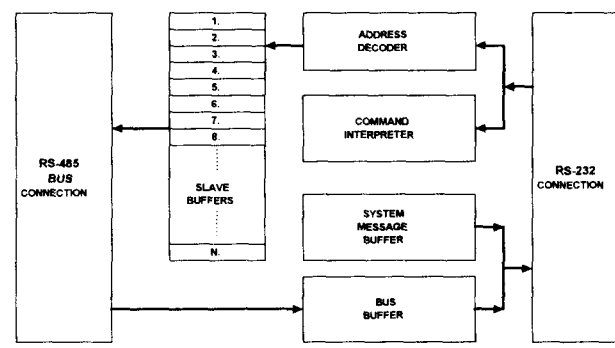

Figure 2 - 4 Functions of the master interface

3. PROTOCOL DESCRIPTION

3.1. FUNCTIONAL DESCRIPTION

Before making a communication protocol we have to know the message types which could happen in the whole system. See the figure 3 - 1

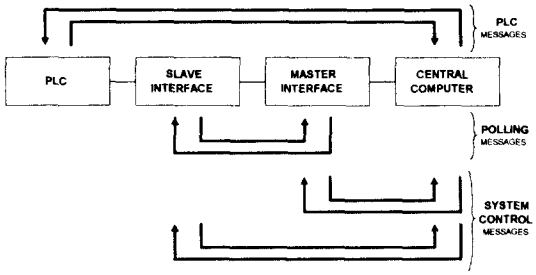

Figure 3 - 1 Message Types

We can recognize only three types of messages.

The **PLC messages** are the most important among all of them. The communication system is to transport them.

All of the others are for the correct operation of the communication system, for this reason they are very important. We can divide them into two groups:

a; **Polling messages**: the sent **questions** (by the master to one of slaves) , and **answers** (from the slave to the master) can be named in this section.

b; **System control messages**: all of the interfaces are directly controlled by the central computer. Almost all of the configuration data and other more information for the correct operation can be loaded from the central computer to the interfaces. The interfaces could have some important messages from itself to send to the central computer. (For example: buffer full error). We have to ensure the transfer of this kind of messages.

3.2 DATA TRANSFER ON THE SERIAL BUS

How to transfer any data with this algorithm? We must recognize 3 different cases:

A;

Neither the master nor the slave do not want transfer any data to each other. The master send a QUESTION / NO MESSAGE byte to the slave, and the slave resend the same byte. (QUESTION / NO MESSAGE) See the figure 5 - 3

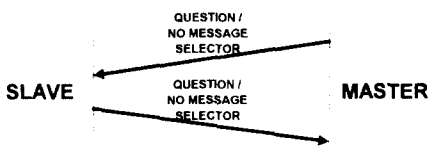

Figure 3 - 2

B;

The slave has a message to send to the master. The master sends a QUESTION / NO MESSAGE byte to the slave, and the slave answers a PLC SELECTOR byte or a SYSTEM CONTROL MESSAGE SELECTOR byte, and after that sends the DATA-BYTE and the DATA-BYTE-INVERT. If the transfer is completed, there is no error occurred, the master sends an OK SELECTOR BYTE. See the figure 3 - 3.

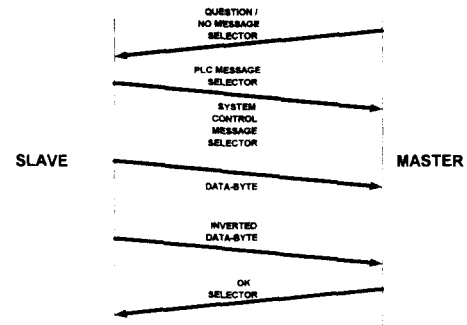

Figure 3 - 3

C;

The master has a message to send to the slave. The master sends a PLC SELECTOR byte or a SYSTEM CONTROL MESSAGE SELECTOR byte after send the DATA-BYTE and the DATA-BYTE-INVERT to the slave. If the transfer is completed, there is no error occurred, the slave sends an OK SELECTOR BYTE. See the figure 3 - 4.

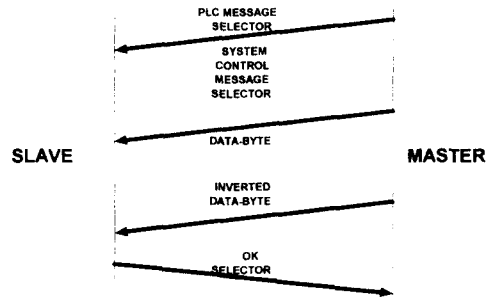

Figure 3 - 4

4. COMMUNICATION SOFTWARE DESCRIPTION

4.1 FIFO TYPE BUFFERS

All of our buffers are FIFO type buffers. The FIFO aspect is realized by **two pointers** : the input and the output pointer which are independently operating. When an input data arrives, we load it to the current address, and after that we increment the input pointer. The reading from the buffer means the reading of the pointed byte by the output-pointer and after increment this pointer. The FIFO seems like a ring. So when the input pointer catch up the output-pointer, a buffer-full error happens. When the output pointer catch up the input-pointer, the buffer is empty.

Their sizes are different depending on the application. We have two basic type of FIFO buffer: **small** and **large**.

The **small** ones are composed only of **256 bytes**. By this way the addressing method will be very simple. Even if the buffer-pointer is 16 bits length, the MS byte can be unmodified,

the LS byte can be increased continuously. We do not need to recognize the top or the bottom of the buffer, it really seems like a ring. See the figure: 4 - 1.

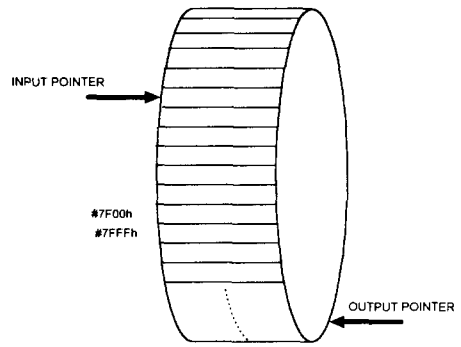

Figure 4 - 1 small size buffer

The **large** one is composed of one or **more slices**. Every slice is 256 bytes long. We have to make a ring, so we have to check the pointers if they are at the top of the buffer. If it is pointing there, we have to reload the bottom value to it. It takes more time, because of this the large buffers are more slow than the small ones.

Also the size can be modified easily at the beginning of the program. The MS bytes and only the MS bytes are defined over there. For ex.: if you want to create your buffer between the addresses: #0000H and #3F00H, you have to set the MIN value : #0000H and the MAX value : #4000H. Do not worry about this difference. The program will not write on the #4000H address. After writing something to #3F00H, it will increment the input-pointer, and it will notice the MAX value and set the pointer to the MIN value, #0000H . So the next buffer placement could be: i.e.: MIN: #4000H and MAX: #8000H. See the figure: 4 - 2.

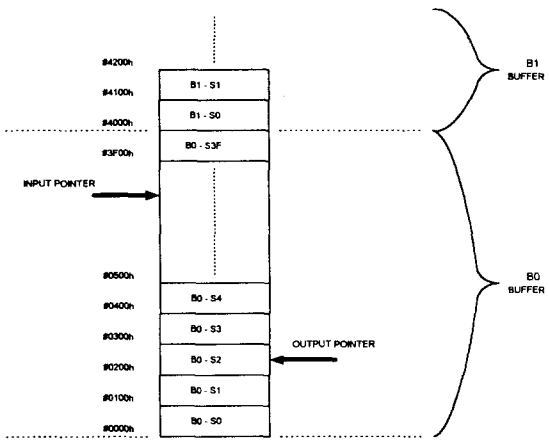

Figure 4 - 2 large size buffer

4.2 INTERRUPT CONTROLLED PROGRAM OPERATING

Our serial interface is connected to two serial lines. Both of them are **asynchronous** types. In this case the interrupt controlled program operating is the best solution to avoid the data lost. Also we have some timer functions, which have to be serviced by an interrupt routine.

The micro-controller has a large amount of interrupt lines. It has 8 interrupt levels, and each level has more interrupt source. On a certain level the priority is fixed (by hardware) , but the priority of the levels can be modified. These circuits use the devices such as the built in Serial Controller receiver, transmitter interrupts, the Timer C and D interrupts, and the external P2.5 interrupt. (PLC or Central Computer connection, reception.)

All of these interrupts are pending. The pending bit is cleared automatically in the P2.5 case and by software in the other cases.

The Bus reception and transmission is preferred against the PLC / CC interrupt, but every interrupt routine must be as short as possible.

5. Conclusions

The goal of this research has been to develop for the automatic production system including the hardware elements and the communication software. We described environment of system, configuration of system and hardware description of serial interface. Based on this information, we then deduced that performance-driven simultaneous protocol description and communication software description were a solution to this practicality problem.

The communication network which makes connection among all of the projection machines and connect them to a central computer. This communication system have to be very flexible and fast. It includes hardware elements which cost is considerable. So we had to make decisions very providently. We use new modern devices to develop another advanced systems.

Another part which contain the communication interface for the automatic production system including the hardware elements and communication software. Every unit includes one Samsung Micro-controller, and we employed the SMDS(Samgsung Microprocessor Development System) tool during the program development procedure.

Succinctly, we very much like the superior quality of the results, but future work should continue to address a complex projection control program which include a large database, automatic decision algorithm, automatic testing and supervisor services.

Reference

1. Samsung KS88C0116 user's manual, Revision 1, 1996.
2. Samsung SMDS2 user's guide, 1995.
3. William Schweber, Electronic communication systems, Prentice-Hall, 1996.
4. Leon W. Couch, Digital and analog communication systems, Macmillan publishing company, 1993.