

PROGRAMMABLE LOGIC CONTROLLER

Syed Rizwan <ata786rz@hotmail.com>,
Zulfiqar Ali Syed <zulfiqaralis@hotmail.com>,
Shiraz Khan <shirazkhan1@yahoo.com>,
Muhammad Jawad Bukhari <mjawadbukhari@hotmail.com>

Institute of Industrial Electronics Engineering (IIEE), PCSIR,
ST-22/C, Block-6 Gulshan-e-Iqbal, Karachi-75300, Pakistan

ABSTRACT

Industrial automation is the now vital for its prosperity, since it reduces the production cost many folds and also increases productivity. For this purpose WLC (i.e. Wired Logic Controller) was introduced in 60s-70s. It was the first step in the industrial automation. Later on microcontrollers were introduced which had the advantage of greater flexibility over the WLC. With the evolution of technology, a very sophisticated version of a controller was introduced with a very large I/O handling capability and extreme flexibility called the Programmable Logic Controller, commonly called the PLC.

The PLCs are not alien to the industries of Pakistan. They have been around for quite a long period and are being employed extensively. Even though the demand of this product is considerable here, there has been no effort to develop it indigenously. Since there had been no development in this field in our country and being a requisite of the nation, so as an exercise of Logic Design, Digital Electronics, Interfacing and Software Engineering. As part of our coursework, we have to undertake a project in the final semester. This article describes the design and implementation process of a Programmable Logic Controller (PLC), in that connection. The task was undertaken as an exercise in digital circuit design, microcontroller application and interface since we already had taken up projects related to power and analog electronics in preceding semesters.

Keywords: Programmable Logic Controller (PLC), Automation, Microcontroller Applications, Industrial Electronics, Digital Control, On-Off Control, Digital Electronics

1. INTRODUCTION

1.1 PROBLEM DESCRIPTION

We all know that industrial automation is the backbone of a nation for its prosperity since, it reduces the production cost many folds and also increases productivity. For this purpose WLC (i.e. Wired Logic Controller) was introduced. It was the first step in the industrial automation. Later on microcontrollers were introduced which had the advantage of greater flexibility over the WLC.

Eventually, a very sophisticated version of a controller was introduced with a very large I/O handling capability and extreme flexibility called the Programmable Logic Controller, commonly called the PLC. Siemens introduced many series of controllers with different capabilities. One of the first series was the SIMATIC® 5 Controllers. The PLC available to us is the S5-100U-100. Later on Siemens introduced the SIMATIC® 7 series of controllers and now, we also have the S7-314IFM PLC.

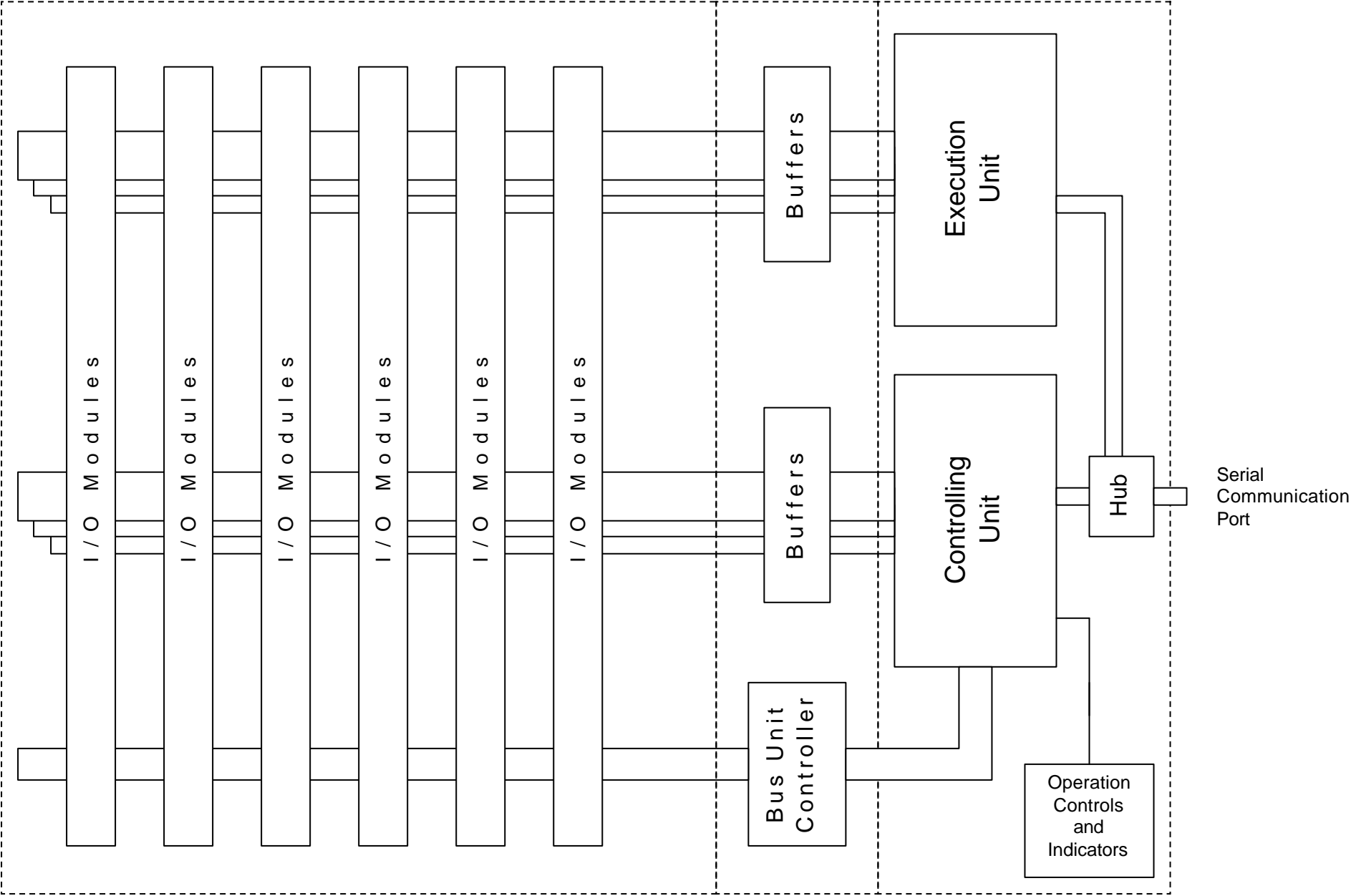
The PLCs are not alien to the industries of Pakistan. They have been around for quite a long period and are being employed extensively. Even though the demand of this product is considerable here, there has been no effort to develop it indigenously. Since there had been no development in this field in our country and being a requisite of the nation, so as an exercise of Logic Design, Digital Electronics, Interfacing and Software Engineering, we undertook this task of making a PLC better than the S5 controllers and comparable to some extent to the S7 controllers. The PLC that we have designed can also be upgraded to support Remote Data Acquisition.

1.2 BLOCK DIAGRAM AND BRIEF DESCRIPTION

The complete general block diagram of the PLC is depicted in Fig.1-1. Just like the SIMATIC® 5 systems, our PLC can also be broadly classified into three distinct units:

- a. CPU
- b. Bus unit, and
- c. Modules (I/O and function)

Fig.1-1 Block diagram of the PLC



Each unit can be discussed as follows:

1.2.1 CPU

The CPU, as the name suggests, is the brain of the PLC system. Without its brain, the PLC simply cannot function.

In our system, the CPU comprises of two ATMEL® 89C52 microcontrollers. As in every multi processor system, one is the master and the other is the slave. The functions performed by the master or the controlling unit (Fig.1-1) can be summarized as follows:

- Monitor and control the operating modes of the PLC using the operation controls.
- Configure the PLC modules.
- Check program integrity by monitoring scan cycles.

The slave microcontroller (executing unit, Fig.1-1) functions to run the user program. It is responsible for:

- Taking the appropriate actions during startup.
- Issuing the \overline{CONV} signal at the start of every scan cycle and the subsequent addressing and transaction of data with the modules.
- Handling the counters and timers.

The CPU section also has the responsibility of communicating the programming device i.e. the computer through a serial port. The communication channel works according to the semaphore algorithm (commonly known as token algorithm). It is to be noted here that the communication link currently does not function, as mentioned earlier, because it was a part of the planned GUI, which has not been implemented due to the constraint of time.

1.2.2 BUS UNIT

The bus unit functions similar to the motherboard in a computer. The bus unit that we have constructed holds up to 4 modules and the CPU; we named it the *main bus unit*. Moreover, the bus unit also carries the extension connector by which other bus units can be cascaded to increase the I/O handling capability of the PLC. The bus units that will be cascaded will not have a slot for connecting the CPU; only modules can be connected.

The main function of the bus unit is to buffer all the signals i.e. the data buses, address buses, and the control signals, to eliminate the loading of the CPU. This function is also performed by the main bus unit but this also performs an extra function during the configuration stage. The bus unit controller, see Fig.1-1, selects each slot one at a time so that it can be configured for the module connected in it and also sends a signal to the controlling unit when all the slots have been selected and configured once.

1.2.3 MODULES

The modules serve to provide the interface of the PLC with the plant to be controlled. The modules, which we had designed were digital input, digital output, analog input and analog output modules (verified using OrCAD® PSpice). But we only managed to manufacture the digital input and output modules, each capable of handling 16 devices. The standards of the modules are similar to that of commercially available modules.

The modules are equipped with an address decoder circuit, which is common in all the modules, as the circuit has been designed to support auto configuration.

1.3 POSSIBLE SOLUTIONS

Now that the overview of the project is present before us, we will take a look into the possible solutions that came to our mind when we commenced brainstorming. The ideas can be listed as follows along with their supporting and disagreement comments.

- Use of a single processor/microcontroller was the first idea that came to mind because it was most simple and easy to implement. The problem with such an idea was that we were striving for a good scan time due to which we needed a fast processor (which was unavailable) or had to go for multiple processors.
- The use of microprocessor instead of a microcontroller was also among the options because we have been taught the assembly language of this processor and we were well aware of it. The drawback of such a selection was that in a PLC, bit operations are plenty while the microprocessor instruction is not very powerful for such operations.
- Using a 16-bit microcontroller for ease of work was a great idea but its materialization was not possible as it was not available here.
- After all these options, we were left only with the option of using two microcontrollers in tandem to achieve our desired task.

1.4 REASONS FOR SELECTION

Of all the above solutions mentioned above, we decided to implement the multi processor CPU with the bus unit scheme for connecting the modules. The advantages, which also form the reasons for the selection of this design, are enumerated below.

- This solution was the first seemingly feasible solution that came to our minds. During the course of work, many more came but then it was too late.
- The design was based on equipment and technology easily available in our city.
- The use of microcontroller instead of a microprocessor is justified by the fact that the microprocessor is incapable of bit operations, which are numerous in a PLC.
- A fast microcontroller was unavailable hence we decided to distribute the work load and run tasks simultaneously by using two controllers so that the speed is not compromised.
- The auto configuration feature allows the design of the modules to be quite generic as the address decoding section is identical hence ensuring ease of manufacture.
- The use of the bus unit allows cost effectiveness as specialized circuits in each module are not required as found in the SIMATIC® 7 controllers.

2. ANALYSIS AND SIMULATION

Since our system is a digital system based on microcontrollers, it was not possible to obtain its mathematical model. Instead, we have simulated our circuit design with all the real life limitations in OrCAD® 9.1 PSpice. The computer hardware simulation results and their analysis are discussed below.

2.1 COMPUTER SIMULATIONS - HARDWARE

The simulations were carried in different stages; the auto configuration process, in which the control word is read and then the appropriate address, is assigned to the module. Once this was tested, the complete sequence of reading data from a digital input module was tested. Similarly, the sequence of writing to a digital output module was also tested.

Before discussing the processes, one thing that must be stressed is that the signals used for testing the performance of the circuits were constructed using actual timing delays that would

be introduced by the microcontrollers. All the propagation delays, pulse widths, etc. have been carefully implemented so that our simulations are as near to practical results as possible.

2.1.1 THE AUTO CONFIGURATION PROCESS

The configuration cycle is handled purely and solely by the second microcontroller. It is the job of this controller to read the control words of the various modules and then assign addresses to each. The process is carried out as follows.

Whenever the PLC is switched ON, the microcontroller will carry out the configuration cycle. To initiate this cycle, the microcontroller sends an active low signal \overline{PSEN} pulse. At the first rising edge of the CLK, this low state is transferred to P0, the signal for selecting the module at the first slot. Meanwhile, \overline{PSEN} now goes back high. Now the microcontroller sends out $A_1A_0=00$, which causes the control word to be addressed. Hence when the \overline{RD} signal goes low, the control word of the module at the first slot is read and stored in the RAM of the microcontroller. Now the next module is selected on the next rising edge of the clock. Similarly, its control word is read and stored. Once a bus unit has been completely checked, it raises a flag bit. This flag bit of each bus unit is ANDed with the flag bit from the next bus unit (i.e. NBUS) and the output of the AND gate goes to the input of the preceding bus unit. This chain continues until the output of the final AND gate goes to the microcontroller. This output goes high when all the modules have been selected once and their control words have been read.

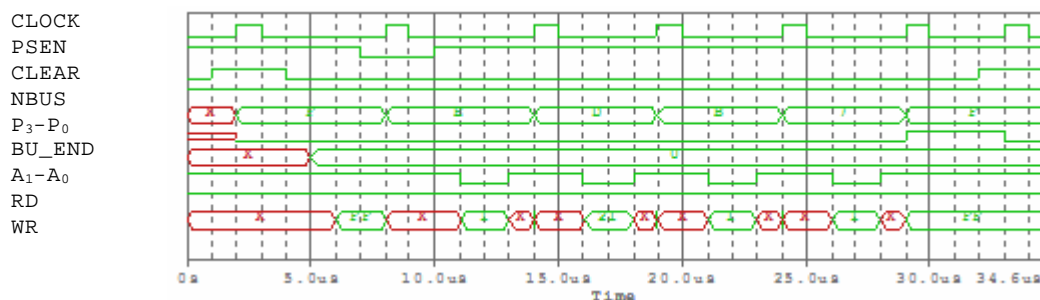


Figure 2.1

The timing diagram shown below depicts the sequential selection of each module and the reading of their control words. Also note that the BU_END signal goes high when all the modules have been read.

Now the microcontroller reads all the stored control words and overwrites them with proper addresses, each 16-bit long. Once all the control words are converted to addresses, these are to be transferred to the modules. Hence, once again, the \overline{PSEN} goes low and on the first rising edge of the clock, the first module is selected. However, this time the microcontroller sends out $A_1A_0=10$, which indicates that the lower byte of the address latch is being pointed to. Then the lower byte of the address appears on the data bus of the microcontroller and when the \overline{WR} signal goes low, the address latch latches the data. Once lower byte has been written, the microcontroller sends out $A_1A_0=11$, which shows that the high byte is being addressed. Again, the high byte is transferred to the address latch when \overline{WR} goes low. This completes the auto configuration for one module. In a similar manner, all the remaining modules are configured, as each is selected one by one on successive rising edges of the clock. The addressing assigning process is depicted in the following timing diagram.

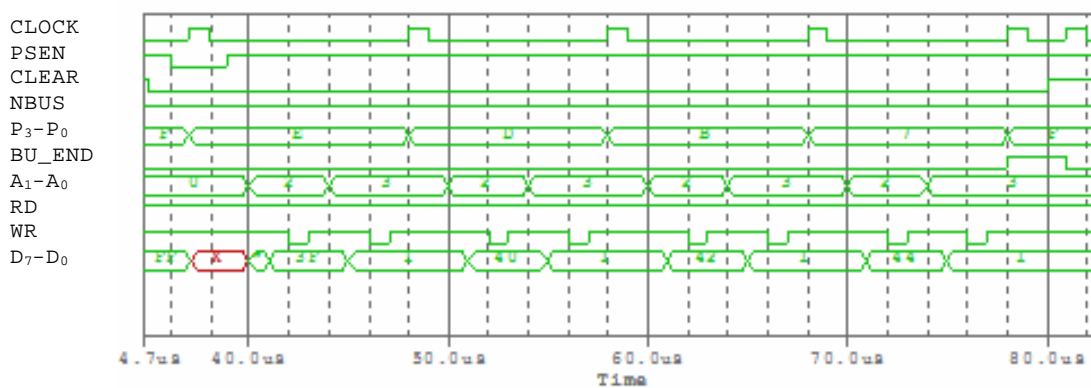
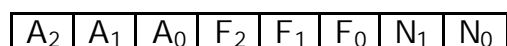


Figure 2.2

2.1.2 CONTROL WORD

To implement this feature, it was necessary that the modules were able to ‘tell’ who they are. This was done by using unique 8-bit control words for each module. These control words are constant for a given type of module and built into the modules during their creation. The bit format of the control word is given below.



By reading this control word, the microcontroller is able to recognize the module and then assign an address to it. The control word for any module can be constructed using the following table.

A ₂	A ₁	A ₀	BASIC MODULE CLASS	F ₂	F ₁	F ₀	MODULE SUB-CLASS	N ₁	N ₀	NO. OF CHANNELS
0	0	0	Digital Input (DI)	X	X	X		0	0	8 Channels
								0	1	16 Channels
								1	X	32 Channels
0	0	1	Digital Output (DO)	X	X	X		0	0	8 Channels
								0	1	16 Channels
								1	X	32 Channels
0	1	0	Analog Input (AI)	0	0	1	8-Bit Resolution	0	0	4 Channels
				0	1	0	12-Bit Resolution	0	1	8 Channels
				1	0	0	16-Bit Resolution	1	X	16 Channels
0	1	1	Analog Output (AO)	0	0	1	8-Bit Resolution	0	0	4 Channels
				0	1	0	12-Bit Resolution	0	1	8 Channels
				1	0	0	16-Bit Resolution	1	X	16 Channels
1	0	0	Digital Input/Output (DIQ)	X	X	X		0	0	4+4 Channels
								0	1	8+8 Channels
								1	X	16+16 Channels
1	0	1	Analog Input/Output (AIQ)	0	0	1	8-Bit Resolution	0	0	2+2 Channels
				0	1	0	12-Bit Resolution	0	1	4+4 Channels
				1	0	0	16-Bit Resolution	1	X	8+8 Channels
1	1	0	Function Module	X	X	X	Not decided yet...	X	X	Not decided yet...
1	1	1	No Module	X	X	X		X	X	

Table 2.1: Configuration control word

2.1.3 ADDRESSING THE MODULES

Since the addressing of each module will be the same, this section has been separated, as it will be common to all the modules.

The addressing of the modules will begin once the configuration cycle has been completed and the user program has been downloaded to the main microcontroller. It will be this microcontroller, which will be responsible for addressing the modules and making them perform according to the user program. Moreover, the user need not be worried about the addresses of the modules because the addresses will be handled by the microcontroller; he will simply have to use the normal syntax, as in other PLCs.

Any module is addressed in one machine cycle using the microcontroller's external memory mode. When the address is completely placed on the address bus, the 16-bit equality comparators of each of the modules compare this address with their assigned address. If it matches their address, a \overline{CS} signal goes low and access to the remaining module is permitted.

The timing waveform shown below shows a digital input module being decoded at 13Eh – 13Fh address.

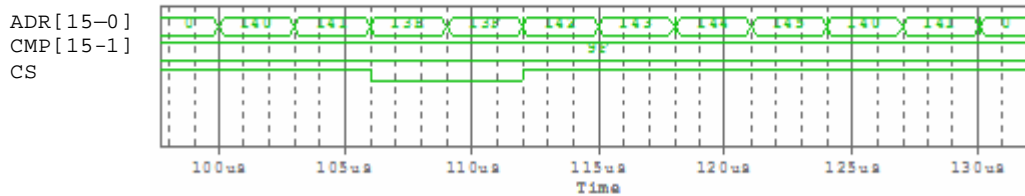


Figure 2.3

2.1.4 DIGITAL INPUT MODULE

As illustrated by the control words, a digital input module could have eight, sixteen or thirty-two inputs. The module that we simulated is a 16-input module. The input voltage standards are maintained from those of the industry i.e. 13-24 V represents logic high. The industry standards are converted to TTL levels by the use of optoisolators, which also isolate the following circuits from the high voltage side.

Accessing the Module

The address of the module is decided in the auto configuration stage after reading the control word and the user need not be worried about it.

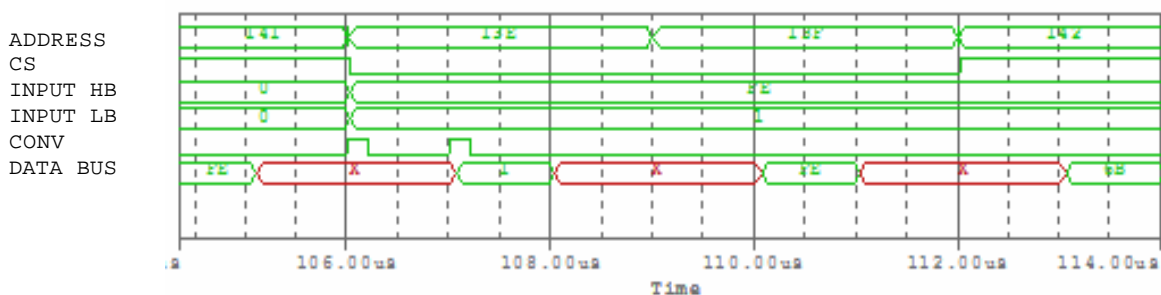


Figure 2.4

The 16-input module is decoded at two addresses; one for the lower byte and the other for the higher byte. The \overline{CONV} signal, from the main microcontroller causes the two latches at the input stage to latch the present status of the inputs simultaneously. Now when we read from the lower address (i.e. $A_0=0$), we are reading the lower eight inputs while reading from the next address (i.e. $A_0=1$), we get the status of the higher eight inputs. So a digital module with 16 inputs is read in two cycles while a module with 32 inputs would require four cycles.

2.1.5 DIGITAL OUTPUT MODULE

Similar to the input module, a digital output module could have eight, sixteen or thirty-two outputs. The module that we simulated is a 16-output module. The output voltage standards are once again maintained from those of the industry i.e. 24 V represents logic high. The module is isolated from the plant by the use of opto-isolators at the outputs and the industry standard is obtained by the use of opto-isolators.

Accessing the Module

Once again, the 16-output module is decoded at two addresses; one for the lower byte and the other for the higher byte. However, here the sequence of events is the reverse of what it was for the input module.

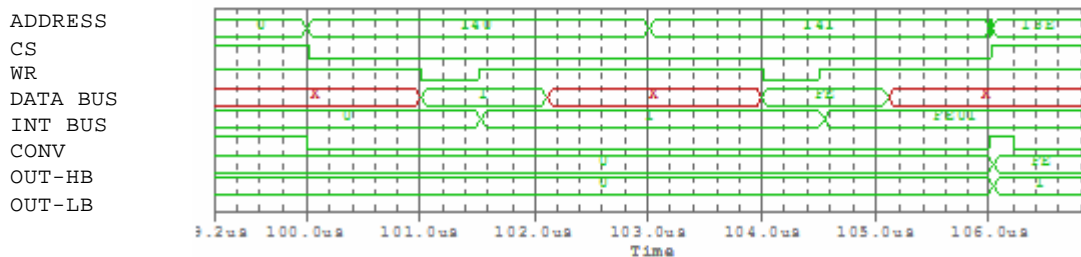


Figure 2.5

After the decisions have been taken according to the user program, the logic states are transferred to the output module in two cycles. First the lower byte is written to the output latch by using the lower address and then the higher byte is written to the latch. Once this has been done, the \overline{CONV} signal appears and all the states are transferred to the output latches simultaneously. This cycle is repeated in every scan cycle monotonously.

2.1.6 ANALOG INPUT MODULE

The analog module that we simulated is an 8-channel 12-bit signed output module. The MAX120 ADC used for converting the analog voltages to digital provides 500 kilo samples per second and hence a conversion time of 2 µsec including tracking time. It has an internal -5.0 V reference.

The differential input voltage is limited to $\pm 5V$ by the use of a voltage divider at every input, as this is the limit of the ADC. The input voltages and their respective references are multiplexed using analog multiplexers. The differential inputs to be converted are selected by a sequential circuit (implemented in a PLD) and are applied to the two inputs of a unity

gain differential amplifier. The differential output voltage is passed to the MAX120 for conversion.

Accessing the Module

When a user program is downloaded to the main microcontroller, in which analog inputs are being used, the microcontroller first figures out which inputs are to be converted and according to it, it creates a control word that would be sent to the analog module. This control word is written to address, which was assigned to the module. The sequential circuit interprets the control word, selects and converts the proper input and transfers it to latches from where it is read by the microcontroller.

Although we did manage to successfully simulate the module but constraint of time and facilities forced us to give up its implementation.

2.1.7 ANALOG OUTPUT MODULE

The analog output designed by us is also an 8-channel 12-bit signed module. The MX7547 DAC used at the output provides a good conversion time of 2 μ sec. The output of each channel is equipped with a track and hold buffer circuit to prevent loading of the DAC.

Accessing the Module

Just like a control word was required to access the analog input module, a similar control word generated by the microcontroller is required for accessing the output module. This control word will be used by a sequential circuit to determine which outputs are to receive the following data.

Once this control word is send at the beginning of the user program, on its turn, the address of the output module is sent out. Then the data for the various channels is sent one after the other. The channel to which the data goes is governed by the logic circuit implemented in the PLD. The logic circuit works as a controlled FIFO, as it transfers the incoming data to the outputs starting at the first and proceeding to the last. However, the data goes only to those channels, which were referred to in the control word. As the DAC is very fast, there are no serious timing problems.

Although we did manage to successfully simulate the module but constraint of time and facilities forced us to give up its implementation.

2.2 COMPUTER SIMULATIONS - SOFTWARE

Apart from the hardware simulation, the software simulation which comprised of the microcontroller programs was also conducted. The software used in testing the developed logic was Pinnacle® (software which was cracked by our own classmate). The user friendly environment of the simulator allowed us to easily simulate our programs before implementing them in the controllers. Since the software simulations simply comprised of running the software step by step and correcting the necessary mistakes, the simulation results cannot be presented as done above. It is enough to say that the simulator allowed us to develop an error free program to a very great extent.

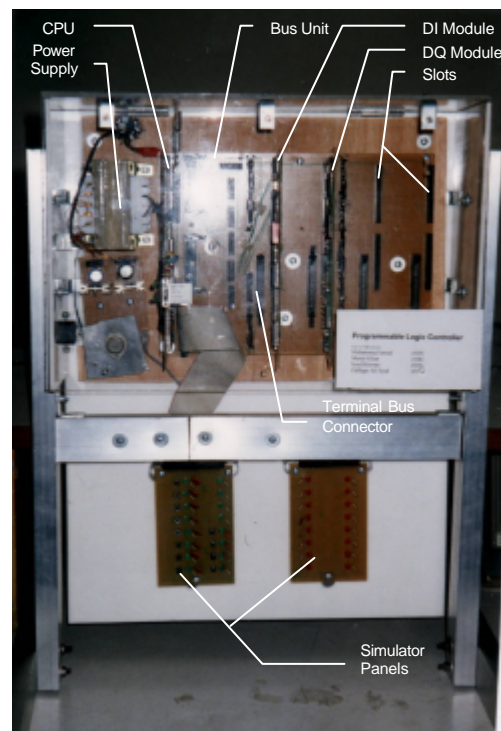
2.3 PERFORMANCE ANALYSIS

The simulation results that were obtained above were done so in a perfect environment i.e. there were no sources of errors present. But the results do guarantee that the design procedure and the plan are fine and might work if implemented properly and carefully. All the devices employed in simulation are available in Karachi and hopefully they will work in the same manner as they did in the simulations.

3. HARDWARE IMPLEMENTATION

The hardware implementation is shown in the following schematics. All the schematics have been developed in OrCAD® 9.1 and are the same that were used for simulation.

Not only the circuits were a part of the implementation but also the casing of the PLC has been manufactured indigenously. The neat looking plastic casing was assembled in an orderly manner so that all the size requirements were met with consummate ease. Since we were providing the casing, we also decided to provide the power supply along with it and hence the power supply of the PLC



Picture of Prototype

is also integrated with it, unlike many other projects.

Since a PLC also needs a simulator panel to simulate the inputs and outputs, we have provided a digital input panel comprising of 16 momentary contact push buttons while an array of 16 red LEDs forms the digital output panel. These, too, have been mounted in a neat and appropriate manner on the PLC frame.

The CPU card that we have implemented has an NVRAM mounted on it, courtesy of Dallas Semiconductors. This non volatile memory device forms the hard disk of our PLC where the user program resides and the system parameters are also recorded. Due to its uniqueness, the complete datasheet of this device is available in the appendix.

Since we were short of time, we were unable to implement the complete STL instruction set of the S7 PLCs but we did accomplish to construct the complete set of bit logical operations of the S7. The architect of the instruction is kept similar to that of S& so that compatibility is retained.

Moreover, the practical implementation is incomplete without the software of the microcontrollers. Hence, instead of the program listing, the detailed program flowcharts have also been provided in this section. The flowcharts have been divided into two sections; one for the execution unit and one for the controlling unit, for ease of understanding.

4. PRACTICAL PERFORMANCE

4.1 DESCRIPTION OF PRACTICAL RESULTS

After the long and arduous struggle to implement the electronic circuits of the PLC, we found out that there were some bugs which were fixed, by the Will of Allah. Since the circuit lay on vero boards, the performance of the PLC was really a question mark. Due to the suspect connections, the PLC functioned inconsistently. Some times it worked and sometimes it did not. But when it did work, it functioned according to plan.

Although the 12 MHz crystals that were connected were different from the planned value of 24 MHz, the microcontrollers worked smoothly. Some logical bugs did appear when we

downloaded their software (even after thorough simulation) but we managed to overcome them, thanks to Allah Almighty.

4.2 COMPARISON WITH SIMULATED RESULTS

Although the comparison of practical timing diagrams and the simulated ones is not possible due to the fact that a logic analyzer is unavailable in our institute, but the fact that the PLC functions properly indicates that the simulation results were true. We have not been able to achieve the clock speed which was used in simulation but a lower speed causes no problems. It is only the high frequencies that affect propagation time considerations.

The microcontroller signals appear on the buses in the same order as simulated hence the sequence of events is also verified.

In brief, the practical results are in agreement with the simulated results except that they are a little slower than the simulated ones.

4.3 SOURCES OF DISCREPANCIES

The biggest source of deviation from the simulated results is the fact that all the ICs available in Karachi are mostly used ones. This causes them to be slightly fatigued due to which they do not perform according to the specifications mentioned in the datasheets. Hence the propagation delays are generally greater than those mentioned. Not only this, but this problem also forced us to reassess our microcontroller software because the signal widths were critical.

Noise in the form of supply voltage fluctuations was also a cause for concern as the high speeds of operation caused oscillations to be set up in the V_{CC} of the ICs. However, this was overcome to a great extent by the use of bypass capacitors.

Apart from these sources, the discrepancies introduced by the errors of implementing the circuit on vero boards were troublesome and in some cases unavoidable. Although we have tried to keep them to a minimum by checking and rechecking, there are still some that have not yet been detected.

5. CONCLUSION

In a nut shell, the whole exercise of designing the PLC and then implementing it was a thorough learning experience, although one with a heavy price tag. There is no denying that the project was too long to be taken as a final year project and our teachers should have prevented us from taking such a task but, alls well that ends well. There were many positives to come out of this whole exercise which may be enlisted as follows:

1. The enormity of the project taught us the art of dividing a big task into several small ones so that it becomes easier to meet targets.
2. The use of OrCAD® Pspice® and Layout for simulation and artwork design became known to us.
3. Designing of complex state machines and cumbersome combinational logics in PLDs was also something new.
4. By designing the PLC according to the computer's design, we gained a brief insight into how computers are designed.
5. The project the served the purpose of teaching how to work as a team.
6. Our skills of recognizing bugs and then troubleshooting them came to the fore when we worked on the circuits.
7. The development of this PLC would certainly open doors for work to begin in this area to make our country self-reliant in this respect.

These were some of the many benefits that we gained out this coursework. But the project itself does not end here. Just like the commercially available PLCs, this one too has many prospective improvements and enhancements which can be taken up by succeeding batches. For example, the analog modules can be manufactured by someone as it has already been designed, and the instruction set can be made even more powerful by working on it. Hence the setup of this project can easily come in handy to encourage work in this direction.

ACKNOWLEDGMENTS

We would like to take this opportunity to acknowledge *Mr. Shahid Ahmad, Mr. Farhan Khan, Ms. Farah Bokhari and Mr. Ashab Mirza* for motivating us to take this challenging study; and for guidance and support in carrying out this project.

REFERENCES

Books Referred

- Siemens Statement List (STL) For S7-300 and S7-400 Programming Reference Manual
- Digital Fundamentals by **Thomas L. Floyd**
- Digital Design by **Morris Mano**
- Microprocessor and Interfacing by **Douglas V. Hall**
- Op-Amps and Linear Integrated Circuits by **Ramakant A. Gayakwad**
- Operational Amplifier Characteristics and Applications by **Robert G. Irvine**
- The 8051 Microcontroller by **I. Scott McKenzie**

Web Sites Visited

- www.8052.com
- www.laticesemi.com
- www.atmel.com
- www.maxim.com