



NEAR EAST UNIVERSITY

Faculty of Engineering

**Department of Electrical and Electronic
Engineering**

LIQUID LEVEL CONTROL BY PLC

**Graduation Project
EE- 400**

Student: Nader Samir (20032224)

Supervisor: Dr.Özgür Cemal Özerdem

Nicosia - 2006

ACKNOWLEDGEMENT



The long day has end.

I would like to express my deepest appreciation to **God** who stood beside me all the time, who supported me in all my achievements and who has given me the power and patience to finish my college studies successfully.

I am very thankful to my Supervisor **Dr. Özgür ÖZERDEM** who was very generous with his help at every stage in the preparation of this project, with his valuable advices and comments.

Special thank goes to Near East University education staff, Especially to Electrical and Electronic Engineering stuff, for their generosity and special concern to me.

I would like to thank my friend **Shimaa** ABURAS for her support me at times to make this project possible.

I would like to thank my friends for their help during preparation of this project, especially S. JARADAT, H. GHANNAM, S. HUSSEIN, and N. BATAL.

Finally, I would like to thank **my parents** for their supports, and encourage for every stage of my education, and specially my Father and my Mother without their endless support and love, I would have never archived my current position.

Abstracts

In the first chapter in this project, an introduction to the PLC, and the history of the PLC, and discusses the Cuts Inside, and the explanation of each part in the PLC as an output and an input, an example INPUT RELAYS, INTERNAL UTILITY RELAYS, COUNTERS and TIMERS.

The first chapter discusses the PLC operations, and the response time of it, and it does manage how to make or how to build a program by managing many steps and how to make these steps in a very clearly way, then discussed the DC Inputs and AC Inputs of the PLC.

The chapter one talks also about AC Motor - Basics of AC Motor Design Engineering and about the Electro Mechanical Valve and the Actuator of the Elector Mechanical Valve.

In the second chapter of this project discusses the practical part of the project, the layout of the project and the way of the experiment connect.

The second chapter discussed the equipment of the Experiment that used in the experimental, the operation principles, and in the practical part there is a program that used in the experiment in order to Control the experiment by using a pc which is connected to The PLC's part.

TABLE OF CONTECTS

ACKNOWLEDGEMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
INTRODUCTION	v
CHAPTER 1:	
PLC STRUCTURE AND OPERATION	1
1.1 PLC History	1
1.2 What is a PLC?	2
1.3 Internal Structure	3
1.3.1 The Parts Inside	4
1.4 PLC Operation	5
1.4.1 Response Time	6
1.5 Creating Programs	7
1.5.1 Relays	7
1.5.2 Replacing Relays	8
1.5.3 Basic Instruction	10
1.5.3.1 Load	10
1.5.3.2 LoadBar	11
1.5.3.3 Out	12
1.5.3.4 OutBar	12
1.6 PLC Registers	14
1.6.1 A Level Application	16
1.6.2 Counters	18
1.6.3 Timers	22
1.6.3.1 Timer Accuracy	25
1.6.4 Shift Registers	25
1.6.5 Math Instructions	28
1.6.6 DC Inputs	28
1.6.7 AC Inputs	31
1.6.7.1 Relays Outputs	33
1.6.7.2 Transistor Outputs	35
1.7 AC Motor-Basic of AC Motor Design Engineering	38
1.7.1 Introduction to AC Motor	38

1.7.1.1	Polyphase of AC Motor	39
1.7.1.2	AC Motors-Design A and B	39
1.7.2	Wound-rotor AC Motors	40
1.7.3	Multispeed AC Motors	40
1.7.3.1	AC Motors-Variable Torque	41
1.7.3.2	AC Motors-Constant Torque	41
1.7.3.3	AC Motors-Constant Horsepower	41
1.7.3.4	AC Motors-Single-Phase AC Motors	41
1.7.4	Universal AC Motors	42
1.7.5	Synchronous AC Motors	43
1.7.6	AC Servo Motors	43
1.8	The Electromechanical Valve	45
1.8.1	The Electromechanical Valve Actuator	45
	SUMMARY	47

CHAPTER 2:

LIQUID LEVEL BY PLC	48	
2.1	The Experimental Setup	48
2.2	Design and the Connection of the Experiment	49
2.3	The Operation of the System	50
2.4	Networks Explanation	51
2.5	Equipment of the Experiment	57
2.6	PLC Program of the Experimental	58
2.7	Program Title Comments	60
	SUMMARY	63
	CONCLUSION	64
	REFERENCES	65

INTRODUCTION

A programmable controller was defined by Capiel (1982) as:

A digitally operating electronic system designed for use in an industrial environment, which uses a programmable memory for the internal storage of instructions for implementing specific functions such as logic, sequencing, timing, counting and arithmetic to control through analog or digital input/output models, various types of machines or processes.

The PLC (i.e. Programmable Logic Controller) is a device that was invented to replace the necessary sequential relay circuits for machine control. The PLC works by looking at its inputs and depending upon their state, turning on/off its outputs. The user enters a program, usually via software, that gives the desired results. technologies were sequencer state-machines and the bit-slice based CPU. The AMD 2901 and 2903 were quite popular in Modicon and A-B PLCs. Conventional microprocessors lacked the power to quickly solve PLC logic in all but the

PLCs are used in many "real world" applications. If there is industry present, chances are good that there is a plc present. If you are involved in machining, packaging, material handling, automated assembly or countless other industries you are probably already using them. If you are not, you are wasting money and time. Almost any application that needs some type of electrical control has a need for a plc.

The History of The PLC, In the late 1960's PLCs were first introduced. The primary reason for designing such a device was eliminating the large cost involved in replacing the complicated relay based machine control systems.

CHAPTER 1

PLC STRUCTURE AND OPERATION

1.1 PLC History

In the late 1960's PLCs were first introduced. The primary reason for designing such a device was eliminating the large cost involved in replacing the complicated relay based machine control systems. Bedford Associates (Bedford, MA) proposed something called a Modular Digital Controller (MODICON) to a major US car manufacturer. Other companies at the time proposed computer based schemes, one of which was based upon the PDP-8. The MODICON 084 brought the world's first PLC into commercial production.

When production requirements changed so did the control system. This becomes very expensive when the change is frequent. Since relays are mechanical devices they also have a limited lifetime which required strict adherence to maintenance schedules. Troubleshooting was also quite tedious when so many relays are involved. Now picture a machine control panel that included many, possibly hundreds or thousands, of individual relays. The size could be mind boggling. How about the complicated initial wiring of so many individual devices! These relays would be individually wired together in a manner that would yield the desired outcome.

These "new controllers" also had to be easily programmed by maintenance and plant engineers. The lifetime had to be long and programming changes easily performed. They also had to survive the harsh industrial environment. That's a lot to ask! The answers were to use a programming technique most people were already familiar with and replace mechanical parts with solid-state ones.

In the mid70's the dominant PLC technologies were sequencer state-machines and the bit-slice based CPU. The AMD 2901 and 2903 were quite popular in Modicon and A-B PLCs. Conventional microprocessors lacked the power to quickly solve PLC logic in all but the

smallest PLCs. As conventional microprocessors evolved, larger and larger PLCs were being based upon them. However, even today some are still based upon the 2903.(ref A-B's PLC-3) Modicon has yet to build a faster PLC than their 984A/B/X which was based upon the 2901.

Communications abilities began to appear in approximately 1973. The first such system was Modicon's Modbus. The PLC could now talk to other PLCs and they could be far away from the actual machine they were controlling. They could also now be used to send and receive varying voltages to allow them to enter the analog world. Unfortunately, the lack of standardization coupled with continually changing technology has made PLC communications a nightmare of incompatible protocols and physical networks. Still, it was a great decade for the PLC.

It was also a time for reducing the size of the PLC and making them software programmable through symbolic programming on personal computers instead of dedicated programming terminals or handheld programmers. Today the world's smallest PLC is about the size of a single control relay.

The 90's have seen a gradual reduction in the introduction of new protocols, and the modernization of the physical layers of some of the more popular protocols that survived the 1980's. The latest standard (IEC 1131-3) has tried to merge plc programming languages under one international standard. We now have PLCs that are programmable in function block diagrams, instruction lists, C and structured text all at the same time! PC's are also being used to replace PLCs in some applications. The original company who commissioned the MODICON 084 has actually switched to a PC based control system.

1.2 What is a PLC?

A PLC (i.e. Programmable Logic Controller) is a device that was invented to replace the necessary sequential relay circuits for machine control. The PLC works by looking at its inputs and depending upon their state, turning on/off its outputs. The user enters a program, usually via software, that gives the desired results.

PLCs are used in many "real world" applications. If there is industry present, chances are good that there is a plc present. *If you are involved in machining, packaging, material handling, automated assembly or countless other industries you are probably already using them. If you are not, you are wasting money and time. Almost any application that needs some type of electrical control has a need for a plc.*

For example, let's assume that when a switch turns on we want to turn a solenoid on for 5 seconds and then turn it off regardless of how long the switch is on for. We can do this with a simple external timer. But what if the process included 10 switches and solenoids? We would need 10 external timers. What if the process also needed to count how many times the switches individually turned on? We need a lot of external counters.

As you can see the bigger the process the more of a need we have for a PLC. We can simply program the PLC to count its inputs and turn the solenoids on for the specified time.

Here's enough information to be able to write programs far more complicated than the simple one above. We will take a look at what is considered to be the "top 20" plc instructions. It can be safely estimated that with a firm understanding of these instructions one can solve more than 80% of the applications in existence.

1.3 Internal Structure

The PLC mainly consists of a CPU, memory areas, and appropriate circuits to receive input/output data. We can actually consider the PLC to be a box full of hundreds or thousands of separate relays, counters, timers and data storage locations. Do these counters, timers, etc. really exist? No, they don't "physically" exist but rather they are simulated and can be considered software counters, timers, etc. These internal relays are simulated through bit locations in registers.

1.3.1 The Parts Inside

- INPUT RELAYS-(contacts) these are connected to the outside world. They physically exist and receive signals from switches, sensors, etc. Typically they are not relays but rather they are transistors.
- INTERNAL UTILITY RELAYS-(contacts) these do not receive signals from the outside world nor do they physically exist. They are simulated relays and are what enables a PLC to eliminate external relays. There are also some special relays that are dedicated to performing only one task. Some are always on while some are always off. Some are on only once during power-on and are typically used for initializing data that was stored.
- COUNTERS-These again do not physically exist. They are simulated counters and they can be programmed to count pulses. Typically these counters can count up, down or both up and down. Since they are simulated they are limited in their counting speed. Some manufacturers also include high-speed counters that are hardware based. We can think of these as physically existing. Most times these counters can count up, down or up and down.
- TIMERS-These also do not physically exist. They come in many varieties and increments. The most common type is an on-delay type. Others include off-delay and both retentive and non-retentive types. Increments vary from 1ms through 1s.
- OUTPUT RELAYS-(coils) these are connected to the outside world. They physically exist and send on/off signals to solenoids, lights, etc. They can be transistors, and relays depending upon the model chosen.
- DATA STORAGE-Typically there are registers assigned to simply store data. They are usually used as temporary storage for math or data manipulation. They can also typically be used to store data when power is removed from the PLC. Upon power-up they will still have the same contents as before power was removed. Very convenient and necessary.

1.4 PLC Operation

A PLC works by continually scanning a program. We can think of this scan cycle as consisting of 3 important steps. There are typically more than 3 but we can focus on the important parts and not worry about the others. Typically the others are checking the system and updating the current internal counter and timer values.

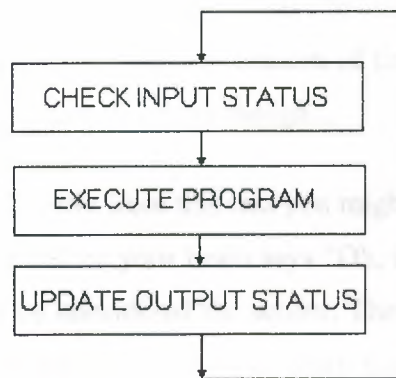


Figure 1.1 Scanning Steps of PLC Program

Step 1-CHECK INPUT STATUS-First the PLC takes a look at each input to determine if it is on or off. In other words, is the sensor connected to the first input on? How about the second input? How about the third... It records this data into its memory to be used during the next step.

Step 2-EXECUTE PROGRAM-Next the PLC executes your program one instruction at a time. Maybe your program said that if the first input was on then it should turn on the first output. Since it already knows which inputs are on/off from the previous step it will be able to decide whether the first output should be turned on based on the state of the first input. It will store the execution results for use later during the next step.

Step 3-UPDATE OUTPUT STATUS-Finally the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during the first step and the results of executing your program during the second step. Based on the example in step 2 it would

now turn on the first output because the first input was on and your program said to turn on the first output when this condition is true.

After the third step the PLC goes back to step one and repeats the steps continuously. One scan time is defined as the time it takes to execute the 3 steps listed above.

1.4.1 Response Time

The total response time of the PLC is a fact we have to consider when shopping for a PLC. Just like our brains, the PLC takes a certain amount of time to react to changes. In many applications speed is not a concern, in others though...

If you take a moment to look away from this text you might see a picture on the wall. Your eyes actually see the picture before your brain says "Oh, there's a picture on the wall". In this example your eyes can be considered the sensor. The eyes are connected to the input circuit of your brain. The input circuit of your brain takes a certain amount of time to realize that your eyes saw something. (If you have been drinking alcohol this input response time would be longer!) Eventually your brain realizes that the eyes have seen something and it processes the data. It then sends an output signal to your mouth. Your mouth receives this data and begins to respond to it.

Notice in this example we had to respond to 3 things:

INPUT- It took a certain amount of time for the brain to notice the input signal from the eyes.

EXECUTION- It took a certain amount of time to process the information received from the eyes. Consider the program to be: If the eyes see an ugly picture then output appropriate words to the mouth.

OUTPUT- The mouth receives a signal from the brain and eventually spits (no pun intended) out the words "Gee, that's a really ugly picture!" as seen in the Figure1.2.

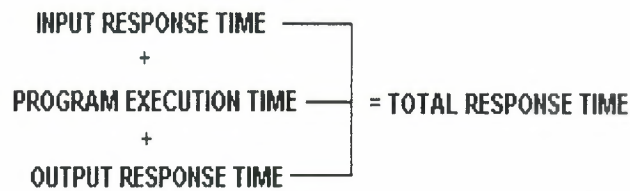


Figure 1.2 Response of PLC to the Execution Steps and Overall

1.5 Creating Programs

1.5.1 Relays

Now that we understand how the PLC processes inputs, outputs, and the actual program we are almost ready to start writing a program. But first let's see how a relay actually works. After all, the main purpose of a plc is to replace "real-world" relays.

We can think of a relay as an electromagnetic switch. Apply a voltage to the coil and a magnetic field is generated. This magnetic field sucks the contacts of the relay in, causing them to make a connection. These contacts can be considered to be a switch. They allow current to flow between 2 points thereby closing the circuit.

Let's consider the following example. Here we simply turn on a bell (Lunch time!) whenever a switch is closed. We have 3 real-world parts

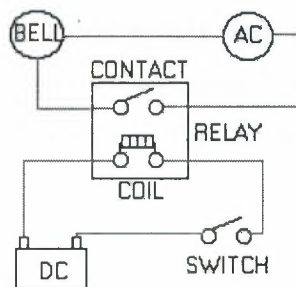


Figure 1.3 Relays Layout

Notice in the picture that we have 2 separate circuits. The bottom (blue) indicates the DC part. The top (red) indicates the AC part.

Here we are using a dc relay to control an AC circuit. That's the fun of relays! When the switch is open no current can flow through the coil of the relay. As soon as the switch is closed, however, current runs through the coil causing a magnetic field to build up. This magnetic field causes the contacts of the relay to close. Now AC current flows through the bell and we hear it, as seen above in the Figure 1.3.

1.5.2 Replacing Relays


Next, let's use a plc in place of the relay. (Note that this might not be very cost effective for this application but it does demonstrate the basics we need.) The first thing that's necessary is to create what's called a ladder diagram. After seeing a few of these it will become obvious why it's called a ladder diagram. We have to create one of these because, unfortunately, a plc doesn't understand a schematic diagram. It only recognizes code. Fortunately most PLCs have software which converts ladder diagrams into code. This shields us from actually learning the plc's code.

First step- We have to translate all of the items we're using into symbols the plc understands. The plc doesn't understand terms like switch, relay, bell, etc. It prefers input, output, coil, contact, etc. It doesn't care what the actual input or output device actually is. It only cares that it's an input or an output.

First we replace the battery with a symbol. This symbol is common to all ladder diagrams. We draw what are called bus bars. These simply look like two vertical bars. One on each side of the diagram. Think of the left one as being + voltage and the right one as being ground. Further think of the current (logic) flow as being from left to right. Next we give the inputs a symbol. In this basic example we have one real world input. (i.e. the switch) We give the input that the switch will be connected to, to the symbol shown below. This symbol can also be used as the contact of a relay.

 A contact symbol

Next we give the outputs a symbol. In this example we use one output (i.e. the bell). We give the output that the bell will be physically connected to the symbol shown below. This symbol is used as the coil of a relay.

 A coil symbol

The AC supply is an external supply so we don't put it in our ladder. The plc only cares about which output it turns on and not what's physically connected to it.

Second step- We must tell the plc where everything is located. In other words we have to give all the devices an address. Where is the switch going to be physically connected to the plc? How about the bell? We start with a blank road map in the PLCs town and give each item an address. Could you find your friends if you didn't know their address? You know they live in the same town but which house? The plc town has a lot of houses (inputs and outputs) but we have to figure out who lives where (what device is connected where). We'll get further into the addressing scheme later. The plc manufacturers each do it a different way! For now let's say that our input will be called "0000". The output will be called "500".

Final step- We have to convert the schematic into a logical sequence of events. This is much easier than it sounds. The program we're going to write tells the plc what to do when certain events take place. In our example we have to tell the plc what to do when the operator turns on the switch. Obviously we want the bell to sound but the plc doesn't know that. It's a pretty stupid device.

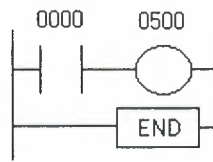


Figure 1.4 Example for Replacing Relays

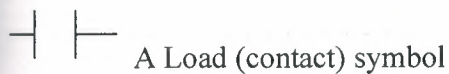
The picture above is the final converted diagram. Notice that we eliminated the real world relay from needing a symbol. It's actually "inferred" from the diagram.

1.5.3 Basic Instructions

Now let's examine some of the basic instructions in greater detail to see more about what each one does.

1.5.3.1 Load

The load (LD) instruction is a normally open contact. It is sometimes also called examine if on.(XIO) (as in examine the input to see if its physically on) The symbol for a load instruction is shown below.



This is used when an input signal is needed to be present for the symbol to turn on. When the physical input is on we can say that the instruction is true. We examine the input for an on signal. If the input is physically on then the symbol is on. An on condition is also referred to as logic 1 state.

This symbol normally can be used for internal inputs, external inputs and external output contacts. Remember that internal relays don't physically exist. They are simulated (software) relays.

1.5.3.2 LoadBar

The LoadBar instruction is a normally closed contact. It is sometimes also called LoadNot or examine if closed. (XIC) (as in examine the input to see if its physically closed) The symbol for a loadbar instruction is shown below.

 A LoadNot (normally closed contact) symbol

This is used when an input signal does not need to be present for the symbol to turn on. When the physical input is off we can say that the instruction is True. We examine the input for an off signal. If the input is physically off then the symbol is on. An off condition is also referred to as a logic 0 state.

This symbol normally can be used for internal inputs, external inputs and sometimes, external output contacts. Remember again that internal relays don't physically exist. They are simulated (software) relays. It is the exact opposite of the Load instruction.

*NOTE- With most PLCs this instruction (Load or Loadbar) MUST be the first symbol on the left of the ladder.

Table 1 Difference between LoadBar and Load

Logic State	Load	LoadBar
0	False	True
1	True	False

1.5.3.3 Out

The Out instruction is sometimes also called an OutputEnergize instruction. The output instruction is like a relay coil. Its symbol looks as shown below.



An OUT (coil) symbol

When there is a path of True instructions preceding this on the ladder rung, it will also be true. When the instruction is true it is physically On. We can think of this instruction as a normally open output. This instruction can be used for internal coils and external outputs.

1.5.3.4 Outbar

The Outbar instruction is sometimes also called an OutNot instruction. Some vendors don't have this instruction. The outbar instruction is like a normally closed relay coil. Its symbol looks like that shown below.



An OUTBar (normally closed coil) symbol

When there is a path of False instructions preceding this on the ladder rung, it will be True. When the instruction is True it is physically On. We can think of this instruction as a normally closed output. This instruction can be used for internal coils and external outputs. It is the exact opposite of the Out instruction.

Table 2 the Difference between OutBar and Out

Logic State	Out	OutBar
0	False	True
1	True	False

A Simple Example

Now let's compare a simple ladder diagram with its real world external physically connected relay circuit and SEE the differences.

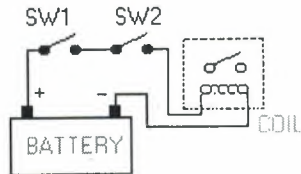


Figure 1.5 Relay Connected

In the above circuit, the coil will be energized when there is a closed loop between the + and - terminals of the battery. We can simulate this same circuit with a ladder diagram. A ladder diagram consists of individual rungs just like on a real ladder. Each rung must contain one or more inputs and one or more outputs. The first instruction on a rung must always be an input instruction and the last instruction on a rung should always be an output (or its equivalent).

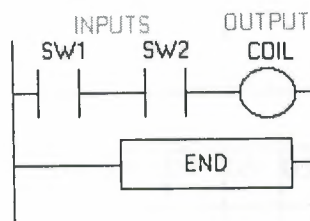


Figure 1.6 Ladder Diagram for Connection Relay.

Notice in this simple one rung ladder diagram we have recreated the external circuit above with a ladder diagram. Here we used the Load and Out instructions. Some manufacturers require that every ladder diagram include an END instruction on the last rung. Some PLCs also require an ENDH instruction on the rung after the END rung.

1.6 PLC Registers

We'll now take the previous example and change switch 2 (SW2) to a normally closed symbol (loadbar instruction). SW1 will be physically OFF and SW2 will be physically ON initially. The ladder diagram now looks like this:

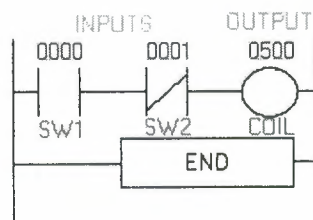


Figure 1.7 Ladder Diagram Registers

Notice also that we now gave each symbol (or instruction) an address. This address sets aside a certain storage area in the PLC's data files so that the status of the instruction (i.e. true/false) can be stored. Many PLCs use 16 slot or bit storage locations. In the example above we are using two different storage locations or registers.

Table 3 The Register 00 and 05

REGISTER 00															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
														1	0
REGISTER 05															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
															0

In the tables above we can see that in register 00, bit 00 (i.e. input 0000) was a logic 0 and bit 01 (i.e. input 0001) was a logic 1. Register 05 shows that bit 00 (i.e. output 0500) was a logic 0. The logic 0 or 1 indicates whether an instruction is False or True. *Although most of the items in the register tables above are empty, they should each contain a 0. They were left blank to emphasize the locations we were concerned with.

Table 4 Logical Condition Symbol

LOGIC BITS	LD	LDB	OUT
Logic 0	False	True	False
Logic 1	True	False	True

The plc will only energize an output when all conditions on the rung are TRUE. So, looking at the table above, we see that in the previous example SW1 has to be logic 1 and SW2 must be logic 0. Then and ONLY then will the coil be true (i.e. energized). If any of the instructions on the rung before the output (coil) are false then the output (coil) will be false (not energized).

Let's now look at a truth table of our previous program to further illustrate this important point. Our truth table will show ALL possible combinations of the status of the two inputs.

Table 5 Previous Program of PLC Register

Inputs		Outputs	Register Logic Bits		
SW1(LD)	SW2(LDB)	COIL(OUT)	SW1(LD)	SW2(LDB)	COIL(OUT)
False	True	False	0	0	0
False	False	False	0	1	0
True	True	True	1	0	1
True	False	False	1	1	0

Notice from the chart that as the inputs change their states over time, so will the output. The output is only true (energized) when all preceding instructions on the rung are true.

1.6.1 A Level Application

Now that we've seen how registers work, let's process a program like PLCs do to enhance our understanding of how the program gets scanned. Let's consider the following application:

We are controlling lubricating oil being dispensed from a tank. This is possible by using two sensors. We put one near the bottom and one near the top, as shown in the picture below.

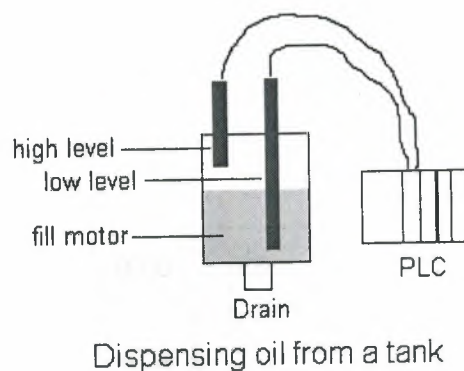


Figure 1.8 Example of Level Application

Here, we want the fill motor to pump lubricating oil into the tank until the high level sensor turns on. At that point we want to turn off the motor until the level falls below the low level sensor. Then we should turn on the fill motor and repeat the process, as shown in Figure 1.8.

Here we have a need for 3 I/O (i.e. Inputs/Outputs). 2 are inputs (the sensors) and 1 is an output (the fill motor). Both of our inputs will be NC (normally closed) fiber-optic level sensors. When they are NOT immersed in liquid they will be ON. When they are immersed in liquid they will be OFF.

We will give each input and output device an address. This lets the plc know where they are physically connected. The addresses are shown in the following tables:

Table 6 Inputs and Outputs Addresses

Inputs	Address	Output	Address	Internal Utility Relay
Low	0000	Motor	0500	1000
High	0001			

Below is what the ladder diagram will actually look like. Notice that we are using an internal utility relay in this example. You can use the contacts of these relays as many times as required. Here they are used twice to simulate a relay with 2 sets of contacts. Remember, these relays DO NOT physically exist in the plc but rather they are bits in a register that you can use to SIMULATE a relay.

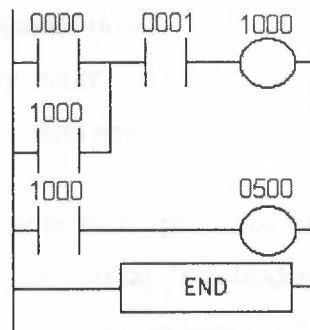


Figure 1.9 Example of a Simulate Relay

We should always remember that the most common reason for using PLCs in our applications is for replacing real-world relays. The internal utility relays make this action possible. It's impossible to indicate how many internal relays are included with each brand of plc. Some include 100's while other includes 1000's while still others include 10's of 1000's! Typically, plc size (not physical size but rather I/O size) is the deciding factor. If

we are using a micro-plc with a few I/O we don't need many internal relays. If however, we are using a large plc with 100's or 1000's of I/O we'll certainly need many more internal relays.

If ever there is a question as to whether or not the manufacturer supplies enough internal relays, consult their specification sheets. In all but the largest of large applications, the supplied amount should be MORE than enough.

1.6.2 Counters

A counter is a simple device intended to do one simple thing - count. Using them, however, can sometimes be a challenge because every manufacturer (for whatever reason) seems to use them a different way. Rest assured that the following information will let you simply and easily program anybody's counters.

What kinds of counters are there? Well, there are up-counters (they only count up 1, 2, 3...). These are called CTU, (count up) CNT, C, or CTR. There are down counters (they only count down 9, 8, 7....).

These are typically called CTD (count down) when they are a separate instruction. There are also up-down counters (they count up and/or down 1,2,3,4,3,2,3,4,5,...) These are typically called UDC(up-down counter) when they are separate instructions.

Many manufacturers have only one or two types of counters but they can be used to count up, down or both. Confused yet? Can you say "no standardization"? Don't worry, the theory is all the same regardless of what the manufacturers call them. A counter is a counter is a counter.

To further confuse the issue, most manufacturers also include a limited number of high-speed counters. These are commonly called HSC (high-speed counter), CTH (CounTer High-speed).

Typically a high-speed counter is a "hardware" device. The normal counters listed above are typically "software" counters. In other words they don't physically exist in the plc but

rather they are simulated in software. Hardware counters do exist in the plc and they are not dependent on scan time.

A good rule of thumb is simply to always use the normal (software) counters unless the pulses you are counting will arrive faster than 2X the scan time. (i.e. if the scan time is 2ms and pulses will be arriving for counting every 4ms or longer then use a software counter. If they arrive faster than every 4ms (3ms for example) then use the hardware (high-speed) counters. ($2 \times \text{scan time} = 2 \times 2\text{ms} = 4\text{ms}$).

To use them we must know 3 things:

1. Where the pulses that we want to count are coming from. Typically this is from one of the inputs.(a sensor connected to input 0000 for example)
2. How many pulses we want to count before we react. Let's count 5 widgets before we box them, for example.
3. When/how we will reset the counter so it can count again. After we count 5 widgets lets reset the counter, for example.

When the program is running on the plc the program typically displays the current or "accumulated" value for us so we can see the current count value.

Typically counters can count from 0 to 9999, -32,768 to +32,767 or 0 to 65535. Why the weird numbers? Because most PLCs have 16-bit counters. We'll get into what this means in a later chapter but for now suffice it to say that 0-9999 is 16-bit BCD (binary coded decimal) and that -32,768 to 32767 and 0 to 65535 is 16-bit binary.

Here are some of the instruction symbols we will encounter (depending on which manufacturer we choose) and how to use them. Remember that while they may look different they are all used basically the same way. If we can setup one we can setup any of them.

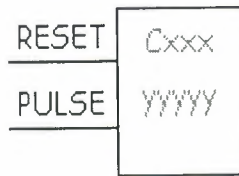


Figure 1.10 Counters

In this counter we need 2 inputs; one goes before the reset line. When this input turns on the current (accumulated) count value will return to zero, the second input is the address where the pulses we are counting are coming from as shown in Figure 1.10.

For example, if we are counting how many widgets pass in front of the sensor that is physically connected to input 0001 then we would put normally open contacts with the address 0001 in front of the pulse line.

Cxxx is the name of the counter. If we want to call it counter 000 then we would put "C000" here.

yyyyy is the number of pulses we want to count before doing something. If we want to count 5 widgets before turning on a physical output to box them we would put 5 here. If we wanted to count 100 widgets then we would put 100 here, etc. When the counter is finished (i.e. we counted yyyyy widgets) it will turn on a separate set of contacts that we also label Cxxx.

Note that the counter accumulated value ONLY changes at the off to on transition of the pulse input.

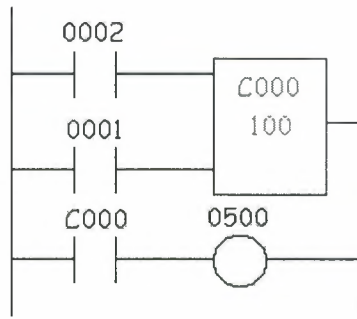


Figure 1.11 the Counter Example

Here's the symbol on a ladder showing how we set up a counter (we'll name it counter 000) to count 100 widgets from input 0001 before turning on output 500. Sensor 0002 resets the counter.

Below is one symbol we may encounter for an up-down counter. We'll use the same abbreviation as we did for the example above.(i.e. UDCxxx and yyyy)

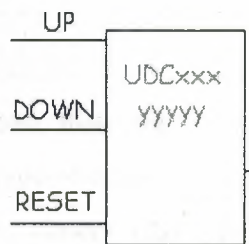


Figure 1.12 Up Down Counter

In this up-down counter we need to assign 3 inputs. The reset input has the same function as above. However, instead of having only one input for the pulse counting we now have 2. One is for counting up and the other is for counting down. In this example we will call the counter UDC000 and we will give it a preset value of 1000. (We'll count 1000 total pulses) For inputs we'll use a sensor which will turn on input 0001 when it sees a target and another sensor at input 0003 will also turn on when it sees a target. When input 0001 turns on we

count up and when input 0003 turns on we count down. When we reach 1000 pulses we will turn on output 500.

1.6.3 Timers

Let's now see how a timer works. What is a timer? It's exactly what the word says... it is an instruction that waits a set amount of time before doing something. Sounds simple doesn't it.

When we look at the different kinds of timers available the fun begins. As always, different types of timers are available with different manufacturers. Here are most of them:

- **On-Delay timer**-This type of timer simply "delays turning on". In other words, after our sensor (input) turns on we wait x-seconds before activating a solenoid valve (output). This is the most common timer. It is often called TON (timer on-delay), TIM (timer) or TMR (timer).
- **Off-Delay timer**- This type of timer is the opposite of the on-delay timer listed above. This timer simply "delays turning off". After our sensor (input) sees a target we turn on a solenoid (output). When the sensor no longer sees the target we hold the solenoid on for x-seconds before turning it off. It is called a TOF (timer off-delay) and is less common than the on-delay type listed above. (i.e. few manufacturers include this type of timer)
- **Retentive or Accumulating timer**- This type of timer needs 2 inputs. One input starts the timing event (i.e. the clock starts ticking) and the other resets it. The on/off delay timers above would be reset if the input sensor wasn't on/off for the complete timer duration. This timer however holds or retains the current elapsed time when the sensor turns off in mid-stream. For example, we want to know how long a sensor is on for during a 1 hour period. If we use one of the above timers they will keep resetting when the sensor turns off/on. This timer however, will give us a total or accumulated time. It is often called an RTO (retentive timer) or TMRA (accumulating timer).

Let's now see how to use them. We typically need to know Two things:

1. **What will enable the timer?** Typically this is one of the inputs.(a sensor connected to input 0000 for example)
2. **How long we want to delay before we react?** Let's wait 5 seconds before we turn on a solenoid, for example.

When the instructions before the timer symbol are true the timer starts "ticking". When the time elapses the timer will automatically close its contacts. When the program is running on the plc the program typically displays the elapsed or "accumulated" time for us so we can see the current value. Typically timers can tick from 0 to 9999 or 0 to 65535 times.

Why the weird numbers? Again its because most PLCs have 16-bit timers. We'll get into what this means in a later chapter but for now suffice it to say that 0-9999 is 16-bit BCD (binary coded decimal) and that 0 to 65535 is 16-bit binary. Each tick of the clock is equal to x-seconds.

Typically each manufacturer offers several different ticks. Most manufacturers offer 10 and 100 ms increments (ticks of the clock). An "ms" is a milli-second or 1/1000th of a second. Several manufacturers also offer 1ms as well as 1 second increments. These different increment timers work the same as above but sometimes they have different names to show their time base. Some are TMH (high speed timer), TMS (super high speed timer) or TMRAF (accumulating fast timer)

Shown below is a typical timer instruction symbol we will encounter (depending on which manufacturer we choose) and how to use it. Remember that while they may look different they are all used basically the same way. If we can setup one we can setup any of them, as shown in figure1.13.

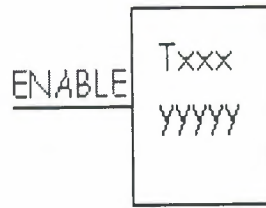


Figure 1.13 Timer

This timer is the on-delay type and is named Txxx. When the enable input is on the timer starts to tick. When it ticks yyyyy (the preset value) times, it will turn on its contacts that we will use later in the program. Remember that the duration of a tick (increment) varies with the vendor and the time base used. (I.e. a tick might be 1ms or 1 second or...)

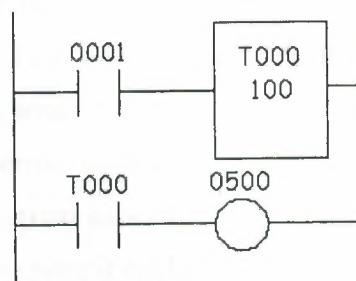


Figure 1.14 On Delay Timer

In this diagram we wait for input 0001 to turn on. When it does, timer T000 (a 100ms increment timer) starts ticking. It will tick 100 times. Each tick (increment) is 100ms so the timer will be a 10000ms (i.e. 10 second) timer. $100\text{ticks} \times 100\text{ms} = 10,000\text{ms}$. When 10 seconds have elapsed, the T000 contacts close and 500 turns on. When input 0001 turns off(false) the timer T000 will reset back to 0 causing its contacts to turn off(become false) thereby making output 500 turn back off.

1.6.3.1 Timer Accuracy

Now that we've seen how timers are created and used, let's learn a little about their precision. When we are creating a timer that lasts a few seconds, or more, we can typically not be very concerned about their precision because it's usually insignificant. However, when we're creating timers that have duration in the millisecond (1ms= 1/1000 second) range we MUST be concerned about their precision.

There are general two types of errors when using a timer. The first is called an input error. The other is called an output error. The total error is the sum of both the input and output errors.

- **Input error-** An error occurs depending upon when the timer input turns on during the scan cycle. When the input turns on immediately after the plc looks at the status of the inputs during the scan cycle, the input error will be at its largest. (i.e. more than 1 full scan time!). This is because, as you will recall, the inputs are looked at once during a scan. If it wasn't on when the plc looked and turns on later in the scan we obviously have an error. Further we have to wait until the timer instruction is executed during the program execution part of the scan. If the timer instruction is the last instruction on the rung it could be quite a big error!
- **Output error-** An another error occurs depending upon when in the ladder the timer actually "times out" (expires) and when the plc finishes executing the program to get to the part of the scan when it updates the outputs. This is because the timer finishes during the program execution but the plc must first finish executing the remainder of the program before it can turn on the appropriate output.

1.6.4 Shift Registers

In many applications it is necessary to store the status of an event that has previously happened. As we've seen in past chapters this is a simple process. But what do we do if we must store many previous events and act upon them later.

We use a register or group of registers to form a train of bits (cars) to store the previous on/off status. Each new change in status gets stored in the first bit and the remaining bits get shifted down the train. Huh? Read on.

The shift register goes by many names. SFT (ShiFT), BSL (Bit Shift Left), SFR (Shift Forward Register) are some of the common names. These registers shift the bits to the left. BSR (Bit Shift Right) and SFRN (Shift Forward Register Not) are some examples of instructions that shift bits to the right. We should note that not all manufacturers have shift registers that shift data to the right but most all do have left shifting registers.

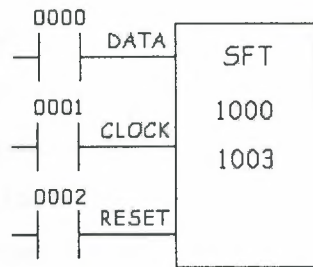


Figure 1.15 Shift Register Instruction

A typical shift register instruction has a symbol like that shown above. Notice that the symbol needs 3 inputs and has some data inside the symbol.

The reasons for each input are as follows:

- **Data-** The data input gathers the true/false statuses that will be shifted down the train. When the data input is true the first bit (car) in the register (train) will be a 1. This data is only entered into the register (train) on the rising edge of the clock input.
- **Clock-** The clock input tells the shift register to "do its thing". On the rising edge of this input, the shift register shifts the data one location over inside the register and enters the status of the data input into the first bit. On each rising edge of this input the process will repeat.

- **Reset-** The reset input does just what it says. It clears all the bits inside the register we're using to 0.

The 1000 inside the shift register symbol is the location of the first bit of our shift register. If we think of the shift register as a train (a choo-choo train that is) then this bit is the locomotive. The 1003 inside the symbol above is the last bit of our shift register. It is the caboose. Therefore, we can say that 1001 and 1002 are cars in between the locomotive and the caboose. They are intermediate bits. So, this shift register has 4 bits.(i.e. 1000,1001,1002,1003)

Imagine an ice-cream cone machine. We have 4 steps. First we verify the cone is not broken. Next we put ice cream inside the cone.(turn on output 500) Next we add peanuts.(turn on output 501) And finally we add sprinkles.(turn on output 502) If the cone is broken we obviously don't want to add ice cream and the other items. Therefore we have to track the bad cone down our process line so that we can tell the machine not to add each item. We use a sensor to look at the bottom of the cone. (Input 0000) If its on then the cone is perfect and if its off then the cone is broken. An encoder tracks the cone going down the conveyor. (Input 0001) A push button on the machine will clear the register (Input 0002).

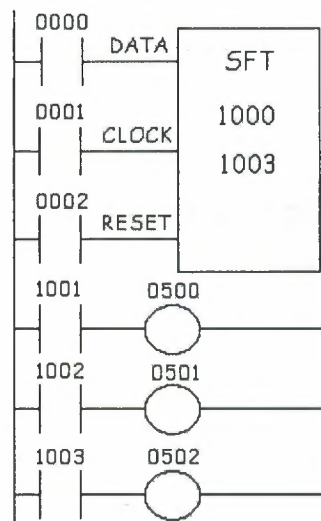


Figure 1.16 SFT Register Instruction Example

1.6.5 Math Instructions

Let's now look at using some basic math functions on our data. Many times in our applications we must execute some type of mathematical formula on our data. It's a rare occurrence when our data is actually exactly what we needed.

As an example, let's say we are manufacturing widgets. We don't want to display the total number we've made today, but rather we want to display how many more we need to make today to meet our quota. Let's say our quota for today is 1000 pieces. We'll say X is our current production. Therefore, we can figure that $1000 - X = \text{widgets left to make}$. To implement this formula we obviously need some math capability.

In general, PLCs almost always include these math functions:

- **Addition-** The capability to add one piece of data to another. It is commonly called ADD.
- **Subtraction-** The capability to subtract one piece of data from another. It is commonly called SUB.
- **Multiplication-** The capability to multiply one piece of data by another. It is commonly called MUL.
- **Division-** The capability to divide one piece of data from another. It is commonly called DIV.

1.6.6 DC Inputs

Let's now take a look at how the input circuits of a plc work. This will give us a better understanding of how we should wire them up. Bad things can happen if we wire them up incorrectly!

Typically, dc input modules are available that will work with 5, 12, 24, and 48 volts. Be sure to purchase the one that fits your needs based upon the input devices you will use.

We'll first look at how the dc inputs work. DC input modules allow us to connect either PNP (sourcing) or NPN (sinking) transistor type devices to them. If we are using a regular

switch (i.e. toggle or pushbutton, etc.) we typically don't have to worry about whether we wire it as NPN or PNP. We should note that most PLCs won't let us mix NPN and PNP devices on the same module. When we are using a sensor (photo-eye, prox, etc.) we do, however, have to worry about its output configuration. Always verify whether it's PNP or NPN. (Check with the manufacturer when unsure)

The difference between the two types is whether the load (in our case, the plc is the load) is switched to ground or positive voltage. An NPN type sensor has the load switched to ground whereas a PNP device has the load switched to positive voltage.

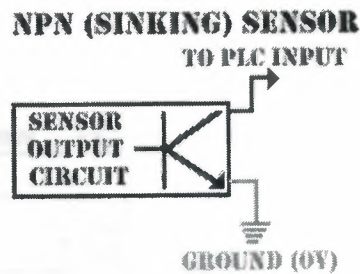


Figure 1.17 NPN Sensor

On the NPN sensor we connect one output to the PLCs input and the other output to the power supply ground. If the sensor is not powered from the same supply as the plc, we should connect both grounds together. NPN sensors are most commonly used in North America.

Many engineers will say that PNP is better (i.e. safer) because the load is switched to ground, but whatever works for you is best. Just remember to plan for the worst.

On the PNP sensor we connect one output to positive voltage and the other output to the PLCs input. If the sensor is not powered from the same supply as the plc, we should connect both V+'s together. PNP sensors are most commonly used in Europe.

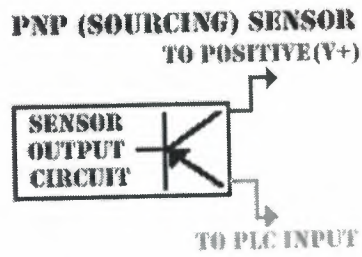


Figure 1.18 PNP Sensor

Inside the sensor, the transistor is just acting as a switch. The sensor's internal circuit tells the output transistor to turn on when a target is present. The transistor then closes the circuit between the 2 connections shown above. (V+ and plc input).

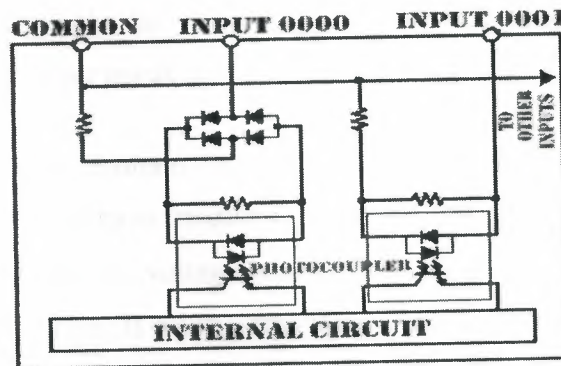


Figure 1.19 Internal Circuit of DC inputs

The only things accessible to the user are the terminals labeled COMMON, INPUT 0000, INPUT 0001, INPUTxxxx... The common terminal either gets connected to V+ or ground. Where it's connected depends upon the type of sensor used. When using an NPN sensor this terminal is connected to V+. When using a PNP sensor this terminal is connected to 0V (ground).

A common switch (i.e. limit switch, pushbutton, toggle, etc.) would be connected to the inputs in a similar fashion. One side of the switch would be connected directly to V+. The other end goes to the plc input terminal. This assumes the common terminal is connected to

0V (ground). If the common is connected to V+ then simply connect one end of the switch to 0V (ground) and the other end to the plc input terminal.

The photo couplers are used to isolate the PLCs internal circuit from the inputs. This eliminates the chance of any electrical noise entering the internal circuitry. They work by converting the electrical input signal to light and then by converting the light back to an electrical signal to be processed by the internal circuit.

1.6.7 AC Inputs

Now that we understand how dc inputs work, let's take a close look at ac inputs. An ac voltage is non-polarized. Put simply, this means that there is no positive or negative to "worry about". However, ac voltage can be quite dangerous to work with if we are careless. (Remember when you stuck the knife in the toaster and got a shock? Be careful) typically, ac input modules are available that will work with 24, 48, 110, and 220 volts. Be sure to purchase the one that fits your needs based upon the input devices (voltage) you will use.

AC input modules are less common these days than dc input modules. The reason being that today's sensors typically have transistor outputs. A transistor will not work with an ac voltage. Most commonly, the ac voltage is being switched through a limit switch or other switch type. If your application is using a sensor it probably is operating on a dc voltage.

We typically connect an ac device to our input module as shown above. Commonly the ac "hot" wire is connected to the switch while the "neutral" goes to the plc common. The ac ground (3rd wire where applicable) should be connected to the frame ground terminal of the plc.(not shown) As is true with dc, ac connections are typically color coded so that the individual wiring the device knows which wire is which. This coding varies from country to country but in the US is commonly white (neutral), black (hot) and green (3rd wire ground when applicable). Outside the US it's commonly coded as brown (hot), blue (neutral) and green with a yellow stripe (3rd wire ground where applicable).

The PLCs ac input module circuit typically looks like this:

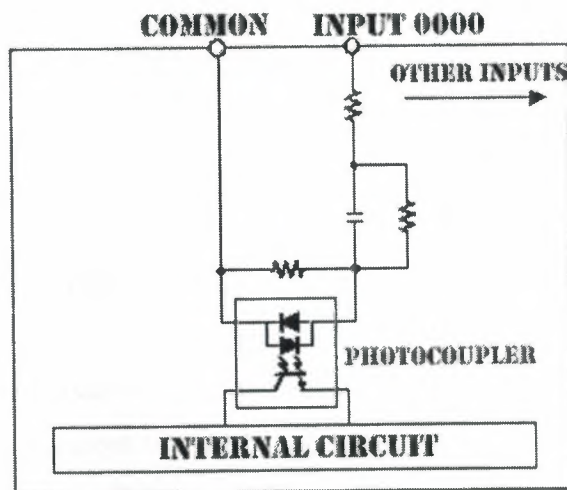


Figure 1.20 Internal Circuits of AC Inputs

The only things accessible to the user are the terminals labeled COMMON, INPUT 0000, INPUTxxxx... The common terminal gets connected to the neutral wire.

A common switch (i.e. limit switch, pushbutton, toggle, etc.) would be connected to the input terminals directly. One side of the switch would be connected directly to INPUT XXX. The other end goes to the ac hot wire. This assumes the common terminal is connected to neutral. Always check the manufacturer's specifications before wiring, to be sure AND SAFE.

The photo couplers are used to isolate the PLCs internal circuit from the inputs. This eliminates the chance of any electrical noise entering the internal circuitry. They work by converting the electrical input signal to light and then by converting the light back to an electrical signal to be processed by the internal circuit.

One last note, typically an ac input takes longer than a dc input for the plc to see. In most cases it doesn't matter to the programmer because an ac input device is typically a mechanical switch and mechanical devices are slow. It's quite common for a plc to require

that the input be on for 25 or more milliseconds before it's seen. This delay is required because of the filtering which is needed by the plc internal circuit. Remember that the plc internal circuit typically works with 5 or less volts dc.

1.6.7.1 Relay Outputs

By now we should have a good understanding of how the inputs are used. Next up is the output circuits.

One of the most common types of outputs available is the relay output. A relay can be used with both AC and DC loads. A load is simply a fancy word for whatever is connected to our outputs. We call it a load because we are "loading the output" with something. If we connected no load to the output (i.e. just connect it directly to a power supply) we would certainly damage the outputs. This would be similar to replacing the light bulb in the lamp you're using to read this with a piece of wire. If you did this, the lamp would draw a tremendous amount of current from the outlet and certainly pop your circuit breaker or blow your fuse or your brains.

Some common forms of a load are a solenoid, lamp, motor, etc. These "loads" come in all sizes. Electrical sizes, that is. Always check the specifications of your load before connecting it to the plc output. You always want to make sure that the maximum current it will consume is within the specifications of the plc output. If it is not within the specifications (i.e. draws too much current) it will probably damage the output. When in doubt, double check with the manufacturer to see if it can be connected without potential damage.

Some types of loads are very deceiving. These deceiving loads are called "inductive loads". These have a tendency to deliver a "back current" when they turn on. This back current is like a voltage spike coming through the system.

A good example of an inductive load that most of us see about 6 months per year is an air conditioning unit. Perhaps in your home you have an air conditioner. (Unless you live in the arctic you probably do!) Have you ever noticed that when the air conditioner "kicks on"

the lights dim for a second or two. Then they return to their normal brightness. This is because when the air conditioner turns on it tries to draw a lot of current through your wiring system. After this initial "kick" it requires less current and the lights go back to normal. This could be dangerous to your PLCs output relays. It can be estimated that this kick is about 30 times the rated current of the load. Typically a diode, varistor, or other "snubber" circuit should be used to help combat any damage to the relay. Enough said. Let's see how we can use these outputs in the "real plc world".

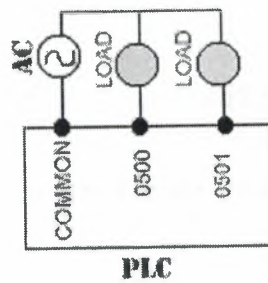


Figure 1.21 Typical Method of Connection the Outputs to the PLC Relays

Shown above is a typical method of connecting our outputs to the plc relays. Although our diagram shows the output connected to an AC supply, DC can be used as well. A relay is non-polarized and typically it can switch either AC or DC. Here the common is connected to one end of our power supply and the other end of the supply is connected to the load.

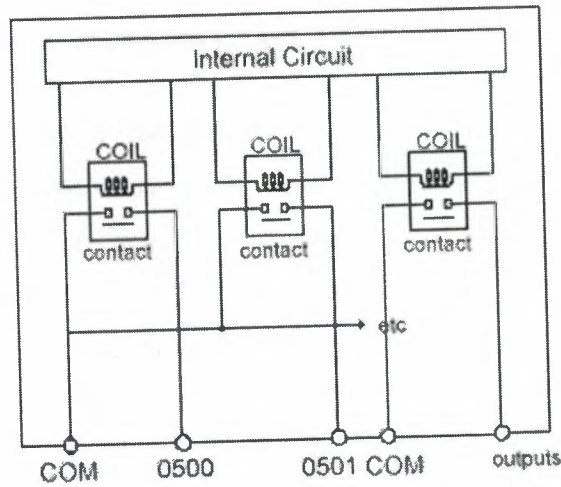


Figure 1.22 Internal Circuit of PLC Output Relay

The relay is internal to the plc. Its circuit diagram typically looks like that shown above in Figure 1.22.

When our ladder diagram tells the output to turn on, the plc will internally apply a voltage to the relay coil. This voltage will allow the proper contact to close. When the contact closes, an external current is allowed to flow through our external circuit. When the ladder diagram tells the plc to turn off the output, it will simply remove the voltage from the internal circuit thereby enabling the output contact to release. Our load will than have an open circuit and will therefore be off.

1.6.7.2 Transistor Outputs

The next type of output we should learn about is our transistor type outputs. It is important to note that a transistor can only switch a dc current. For this reason it cannot be used with an AC voltage.

We can think of a transistor as a solid-state switch. Or more simply put, an electrical switch. A small current applied to the transistors "base" (i.e. input) lets us switch a much larger current through its output. The plc applies a small current to the transistor base and the transistor output "closes". When it's closed, the device connected to the plc output will

be turned on. The above is a very simple explanation of a transistor. There are, of course, more details involved but we don't need to get too deep.

We should also keep in mind that as we saw before with the input circuits, there are generally more than one type of transistor available. Typically a plc will have either NPN or PNP type outputs. The "physical" type of transistor used also varies from manufacturer to manufacturer. Some of the common types available are BJT and MOSFET. A BJT type (Bipolar Junction Transistor) often has less switching capacity (i.e. it can switch less current) than a MOS-FET (Metal Oxide Semiconductor- Field Effect Transistor) type. The BJT also has a slightly faster switching time. Once again, please check the output specifications of the particular plc you are going to use. Never exceed the manufacturer's maximum switching current.

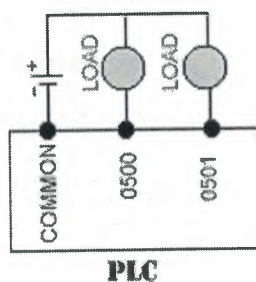


Figure 1.23 Transistor Outputs

Shown above is how we typically connect our output device to the transistor output. Please note that this is an NPN type transistor. If it were a PNP type, the common terminal would most likely be connected to V+ and V- would connect to one end of our load. Note that since this is a DC type output we must always observe proper polarity for the output. One end of the load is connected directly to V+ as shown above.

Let's take a moment and see what happens inside the output circuit. Shown below is a typical output circuit diagram for an NPN type output.

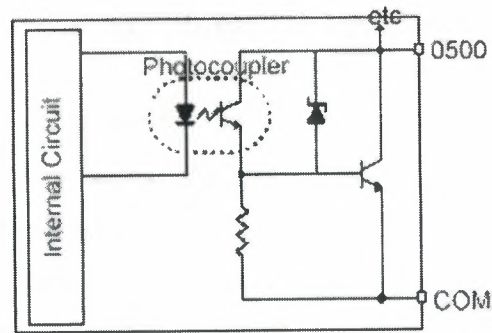


Figure 1.24 Internal Circuit of the Transistor Output

Notice that as we saw with the transistor type inputs, there is a photo coupler isolating the "real world" from the internal circuit. When the ladder diagram calls for it, the internal circuit turns on the photo coupler by applying a small voltage to the LED side of the photo coupler. This makes the LED emit light and the receiving part of the photo coupler will see it and allow current to flow. This small current will turn on the base of the output transistor connected to output 0500. Therefore, whatever is connected between COM and 0500 will turn on. When the ladder tells 0500 to turn off, the LED will stop emitting light and hence the output transistor connected between 0500 and COM will turn off, as shown in Figure 1.24.

One other important thing to note is that a transistor typically cannot switch as large a load as a relay. Check the manufacturer's specifications to find the largest load it can safely switch. If the load current you need to switch exceeds the specification of the output, you can connect the plc output to an external relay. Then connect the relay to the large load. You may be thinking, "why not just use a relay in the first place"? The answer is because a relay is not always the correct choice for every output. A transistor gives you the opportunity to use external relays when and only when necessary.

In summary, a transistor is fast, switches a small current, has a long lifetime and works with dc only. Whereas a relay is slow, can switch a large current, has a shorter lifetime and works with ac or dc. Select the appropriate one based upon your actual application needs.

1.7 AC Motor - Basics of AC Motor Design Engineering

Synchronous and asynchronous electric motors are the two main categories of ac motors. The induction ac motor is a common form of asynchronous motor and is basically an ac transformer with a rotating secondary. The primary winding (stator) is connected to the power source and the shorted secondary (rotor) carries the induced secondary current. Torque is produced by the action of the rotor (secondary) currents on the air-gap flux. The synchronous motor differs greatly in design and operational characteristics, and is considered a separate class of ac motor.

1.7.1 Induction AC Motors:

Induction ac motors are the simplest and most rugged electric motor and consists of two basic electrical assemblies: the wound stator and the rotor assembly. The induction ac motor derives its name from currents flowing in the secondary member (rotor) that are induced by alternating currents flowing in the primary member (stator). The combined electromagnetic effects of the stator and rotor currents produce the force to create rotation.

AC motors typically feature rotors, which consist of a laminated, cylindrical iron core with slots for receiving the conductors. The most common type of rotor has cast-aluminum conductors and short-circuiting end rings. This ac motor "squirrel cage" rotates when the moving magnetic field induces a current in the shorted conductors. The speed at which the ac motor magnetic field rotates is the synchronous speed of the ac motor and is determined by the number of poles in the stator and the frequency of the power supply: $n_s = 120f/p$, where n_s = synchronous speed, f = frequency and p = the number of poles.

Synchronous speed is the absolute upper limit of ac motor speed. If the ac motor's rotor turns exactly as fast as the rotating magnetic field, then no lines of force are cut by the rotor conductors, and torque is zero. When ac motors are running, the rotor always rotates slower than the magnetic field. The ac motor's rotor speed is just slow enough to cause the proper amount of rotor current to flow, so that the resulting torque is sufficient to overcome wind age and friction losses, and drive the load. The speed difference between the ac motor's

rotor and magnetic field, called slip, is normally referred to as a percentage of synchronous speed: $s = 100 (n_s - n_a)/n_s$, where s = slip, n_s = synchronous speed, and n_a = actual speed.

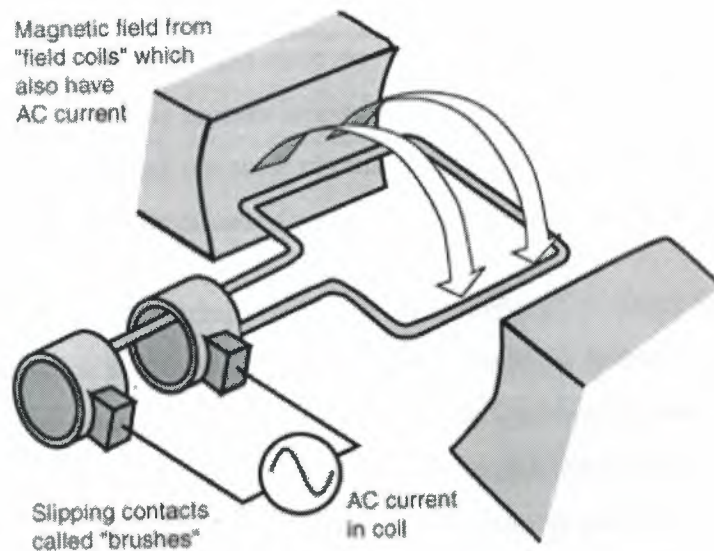


Figure 1.25 AC Motor

1.7.1.1 Polyphase AC Motors

Polyphase squirrel-cage ac motors are basically constant-speed machines, but some degree of flexibility in operating characteristics results from modifying the rotor slot design. These variations in ac motors produce changes in torque, current, and full-load speed. Evolution and standardization have resulted in four fundamental types of ac motors.

1.7.1.2 AC Motors - Designs A and B

General-purpose ac motors with normal starting torques and currents, and low slip. Fractional-horsepower polyphase ac motors are generally design B. Because of the drooping characteristics of design B, a polyphase ac motor that produces the same breakdown (maximum) torque as a single-phase ac motor cannot attain the same speed-torque point for full-load speed as single-phase ac motors. Therefore, breakdown torque

must be higher (a minimum of 140% of the breakdown torque of single-phase, general-purpose ac motors) so that full-load speeds are comparable.

1.7.2 Wound-rotor AC Motors

Squirrel-cage ac motors are relatively inflexible with regard to speed and torque characteristics, but a special wound-rotor ac motor has controllable speed and torque. Application of wound-rotor ac motors is markedly different from squirrel-cage ac motors because of the accessibility of the rotor circuit. AC motor performance characteristics are obtained by inserting different values of resistance in the rotor circuit.

Wound-rotor ac motors are generally started with secondary resistance in the rotor circuit. The ac motor resistance is sequentially reduced to permit the motor to come up to speed. Thus, ac motors can develop substantial torque while limiting locked-rotor current. This secondary ac motor resistance can be designed for continuous service to dissipate heat produced by continuous operation at reduced speed, frequent acceleration, or acceleration with a large inertia load. External resistance gives ac motors a characteristic that results in a large drop in rpm for a fairly small change in load. Reduced ac motor speed is provided down to about 50% rated speed, but efficiency is low.

1.7.3 Multispeed AC Motors

Consequent-pole ac motors are designed for one speed. By physically reconnecting the leads, a 2:1 speed ratio can be obtained. Typical synchronous speeds for 60-Hz ac motors are: 3,600/1,800 rpm (2/4 pole), 1,800/900 rpm (4/8 pole), and 1,200/600 rpm (6/12 pole).

Two-winding ac motors have two separate windings that can be wound for any number of poles so that other speed ratios can be obtained. However, ratios greater than 4:1 are impractical because of ac motor size and weight. Single-phase multispeed ac motors are usually variable-torque design, but constant-torque and constant-horsepower ac motors are available.

Power output of multispeed ac motors can be proportioned to each different speed. These ac motors are designed with output horsepower capacity in accordance with one of the following load characteristics.

1.7.3.1 AC Motors - Variable torque

AC motors have a speed torque characteristic that varies as the square of the speed. For example, a 1,800/900-rpm electrical motor that develops 10 hp at 1,800 rpm produces 2.5 hp at 900 rpm. Since ac motors face loads, such as centrifugal pumps, fans, and blowers, have a torque requirement that varies as the square or cube of the speed, this ac motor characteristic is usually adequate.

1.7.3.2 AC Motors - Constant torque

These ac motors can develop the same torque at each speed, thus power output varies directly with speed. For example, an ac motor rated at 10 hp at 1,800 rpm produces 5 hp at 900 rpm. These ac motors are used in applications with constant torque requirements such as mixers, conveyors, and compressors.

1.7.3.3 AC Motors - Constant horsepower

These ac motors develop the same horsepower at each speed and the torque is inversely proportional to the speed. Typical applications for ac motors include machine tools such as drills, lathes, and milling machines.

1.7.3.4 AC Motors - Single-phase AC Motors

Single-phase induction ac electric motors are commonly fractional-horsepower types, although single-phase integral-horsepower are available in the lower horsepower range. The most common fractional-horsepower single-phase ac motors are split-phase, capacitor-start, permanent split-capacitor, and shaded pole.

The ac motors come in multispeed types, but there is a practical limit to the number of speeds obtained. Two, three, and four-speed motors are available, and speed selection may be accomplished by consequent-pole or two-winding methods.

Single-phase ac electric motors run in the direction in which they are started; and they are started in a predetermined direction according to the electrical connections or mechanical setting of the starting means. General-purpose ac motors may be operated in either direction, but the standard ac motor rotation is counterclockwise when facing the end opposite the drive shaft. AC motors can be reconnected to reverse the direction of rotation.

1.7.4 Universal AC Motors

Universal ac motors operate with nearly equivalent performance on direct current or alternating current up to 60 Hz. AC motors differ from a dc motors due to the winding ratios and thinner iron laminations. DC motors runs on ac, but with poor efficiency. Universal ac motors can operate on dc with essentially equivalent ac motor performance, but with poorer commutation and brush life than for an equivalent dc motor.

An important characteristic of universal ac motors is that it has the highest horsepower-per-pound ratio of any ac motor because it can operate at speeds many times higher than that of any other 60-Hz electric motor.

When operated without load, universal ac motors tend to run away, speed being limited only by wind age, friction, and commutation. Therefore, large universal ac motors are nearly always connected directly to a load to limit speed. On portable tools such as electric saws, the load imposed by the gears, bearings, and cooling fan is sufficient to hold the no-load speed down to a safe value.

With a universal ac motor, speed control is simple, since electric motor speed is sensitive to both voltage and flux changes. With a rheostat or adjustable autotransformer, ac motor speed can be readily varied from top speed to zero.

1.7.5 Synchronous AC Motors

Synchronous ac motors are inherently constant-speed electric motors and they operate in absolute synchronism with line frequency. As with squirrel-cage induction ac motors, speed is determined by the number of pairs of poles and is always a ratio of the line frequency.

Synchronous ac motors are made in sizes ranging from sub fractional self-excited units to large-horsepower, direct-current-excited ac motors for industrial drives. In the fractional-horsepower range, synchronous ac motors are used primarily where precise constant speed is required.

In large horsepower sizes applied to industrial loads, synchronous ac motors serve two important functions. First, ac motors provide highly efficient means of converting ac energy to mechanical power. Second, ac motors can operate at leading or unity power factor, thereby providing power-factor correction.

There are two major types of synchronous ac motors: no excited and direct-current excited electric motors

1.7.6 AC Servo Motors

Timing Electric Motors (cont.) Ac and dc electric motors can be used as timing motors. Dc electric timing motors are used for portable applications, or where high acceleration and low speed variations are required. These electric motors offer advantages, which include starting torque as high as ten times running torque, efficiency from 50 to 70%, and relatively easy speed control. But some form of speed governor, either mechanical or electronic, is required.

Ac motors use readily available power, are lower in cost, have improved life, and do not generate RFI. However, ac motors cannot be readily adapted to portable applications, have relatively low starting torques, and are much less efficient than dc motors.

Ac servo motors are used in ac servomechanisms and computers which require rapid and accurate response characteristics. To obtain these characteristics, servo motors have small-

diameter high-resistance rotors. The small diameter provides low inertia for fast starts, stops, and reversals, while the high resistance provides a nearly linear speed-torque relationship for accurate control.

Servo motors are wound with two phases physically at right angles or in space quadrature. Servo motors feature a fixed or reference winding is excited from a fixed voltage source, while the control winding is excited by an adjustable or variable control voltage, usually from a servo amplifier. The servo motor windings are usually designed with the same voltage-turns ratio, so that power inputs at maximum fixed-phase excitation and at maximum control-phase signal are in balance.

In an ideal servo motor, torque at any speed is directly proportional to the servo motor's control-winding voltage. In practice, however, this relationship exists only at zero speed because of the inherent inability of an induction servo motor to respond to voltage input changes under conditions of light load.

The inherent damping of servo motors decreases as ratings increase and the servo motors have a reasonable efficiency at the sacrifice of speed-torque linearity. Larger servo motors have integral auxiliary blowers to maintain temperatures within safe operating ranges. Servo motors are available in power ratings from less than 1 to 750 W, in sizes ranging from 0.5 to 7-in. OD. Most servo motors are available with modular or built-in gear heads.

One of the drawbacks of this kind of AC motor is the high current which must flow through the rotating contacts. Sparking and heating at those contacts can waste energy and shorten the lifetime of the motor. In common AC motors the magnetic field is produced by an electromagnet powered by the same AC voltage as the motor coil. The coils which produce the magnetic field are sometimes referred to as the "stator", while the coils and the solid core which rotates is called the "armature". In an AC motor the magnetic field is sinusoidally varying, just as the current in the coil varies.

1.8 The Electromechanical Valve

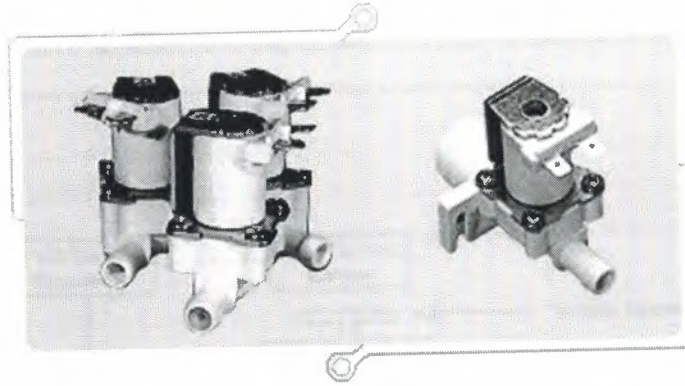


Figure 1.26 the Electromechanical Valve

Electromechanical valve actuators have received much attention recently due to their potential for improving the performance of the internal combustion engine. Various control schemes for the EVA have been proposed, however stability is often neglected due to the bounded motion of the EVA or proven based on a linearized plant model. Here, we demonstrate and prove that our controller renders the system GAS without any assumption of linearity.

1.8.1 The Electromechanical Valve Actuator

The actuator consists of an armature mounted between two opposing magnetic coils and springs. The experimental setup consists of a 200 V power supply, two Pulse Width Modulated (PWM) drivers, an eddy current sensor mounted on the rear of the actuator, a laser vibrometer, and a 1103 dSpace processing board. During operation the dSpace processing board regulates the duty cycle command to the PWM drivers in order to govern the motion of the armature. The commands from the dSpace processing board are based on either open loop instructions or measurements from the various sensors. The position of the armature/valve is measured by an eddy current sensor that detects changes in the magnetic field due to the motion of the sensor target. The velocity of the armature/valve is measured

by a laser vibrometer. Lastly, the current in each the electromagnet is measured by sensors build into each PWM driver. The controller, which is presented in Sec. III, is implemented using the eddy current sensor and the sensors bundled with the PWM drivers. The laser vibrometer is used to verify the system performance for the experimental resThe electromechanical valve actuator and experimental setup are shown in Figure1.27.

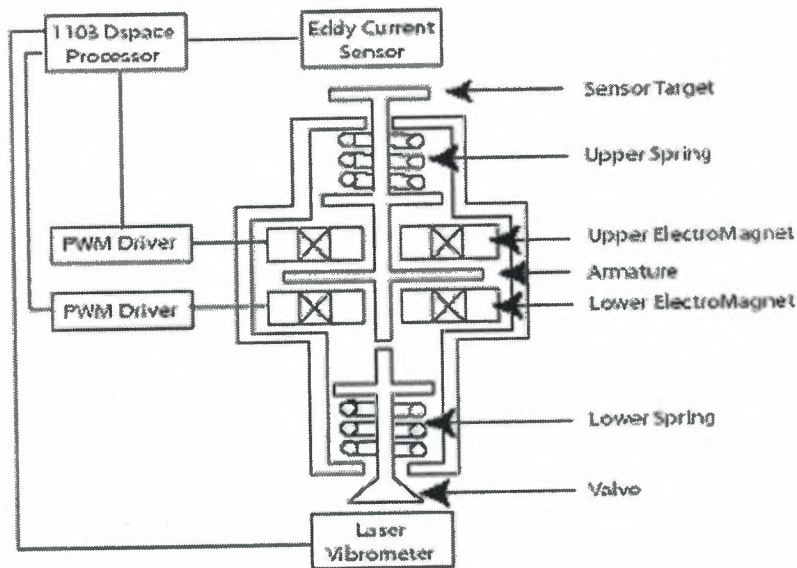


Figure 1.27 Electromechanical Valve Actuator and Experimental Setup

At the beginning of a valve opening/closing procedure the armature is held in contact with one of the two magnetic coils, referred to as the releasing coil, creating a force imbalance between the two springs. The current in the releasing coil is reduced to zero and the springs drive the armature across the gap, where it is then caught and held in place by the opposing magnetic coil, referred to as the catching coil. The forcing of the armature between the two extreme positions thereby opens or shuts the valve. If the voltage to the catching coil is not carefully regulated large impacts can occur between it and the armature. These impacts must be kept below 0.1 m/s to avoid damage and excessively loud operation.

SUMMARY

In this chapter an introduction to the PLC and the history of it, the PLC's Structure, the operation, Types of PLC, Programming of PLC, Technical Program of PLC, The Hints for writing good ladder programs, the PLC communications (parallel and series communications), and the applications examples.

Last part of the chapter discussed the AC Motor and the Electro mechanical Valve, of designing and the operation of the AC Motor and the Electro Mechanical Valve.

CHAPTER 2

LIQUID LEVEL CONTROL BY PLC

2.1 The Experimental Setup



Figure 2.1 the Experimental Setup

2.2 Design and the Connection of the Experiment

Liquid Level Control mechanism is based on three liquid tanks, one main tank and two sub-tanks. The volume of main tank has same volume of sum of sub-tanks. For the main tank, liquid flow is provided by means of Valve3; for the sub-tank1, flow is let by Valve1 and Valve2 is for sub-tank2, as show in Figure2.2 below.

There is a water-pumping motor feeding main tank. The motor takes water from Valve1 and Valve2. A switch is used for starting-stopping the system.

There are three lamps, two red lamps and one orange lamp. First red lamp indicates that the motor is operating. Second red lamp is for mixer indicating mixer is operating. The third lamp (orange lamp) shows heater is in operation. Mixer and heater are located in the main tank. Mixer is for mixing the water inside the main tank. Heater heats the water for hot water need.

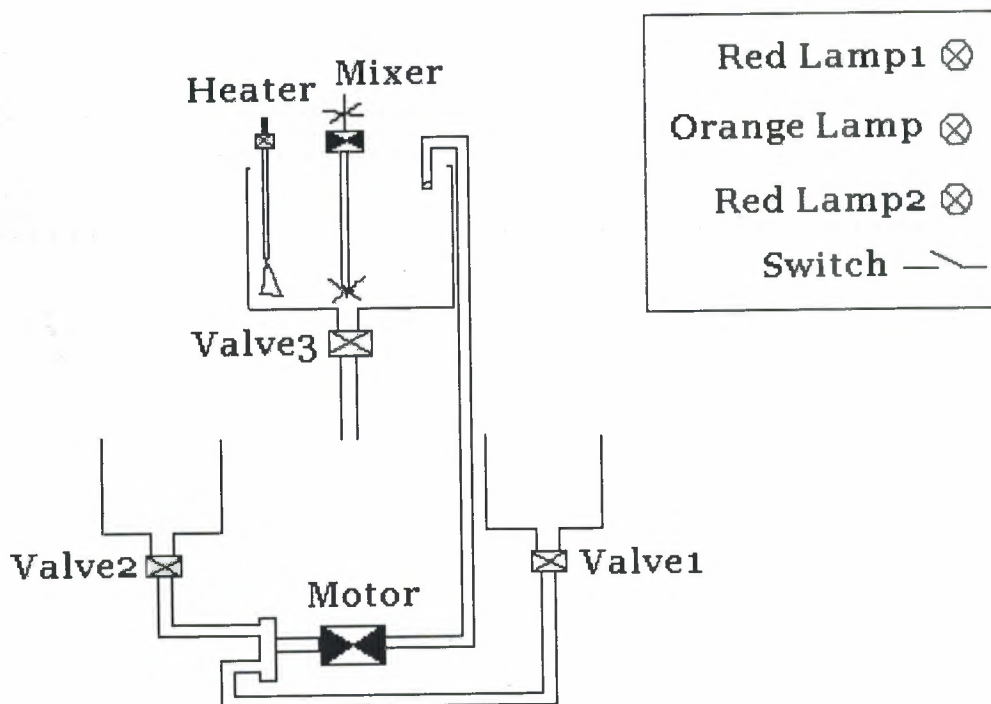


Figure 2.2 the Experimental Layout

2.3 The Operation of the System

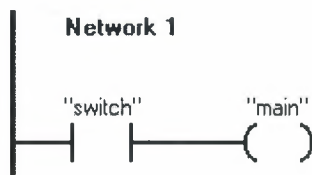
The motor will start pumping when the switch is turned on. This will be introduced in networks explanation as starting the “main”. First red lamp will operate at the same time with motor. After motor started, directly Valve1 will start operation. Valve1 start time is controlled by timer parameters. After Valve1 completes its operation, it will let Valve2 to operate for specific time. After Valve2 stopped, mixer and second red lamp will operate together for predefined time by the PLC program. Heater will start operation after mixer stopped.

Orange lamp will start with heater and stops when heater stopped. Heater operation period can be adjusted by changing related timer parameter.

After mixing and heating the water in the main tank, the water will be ready for use. Valve3 will start emptying the tank to the receiver part.

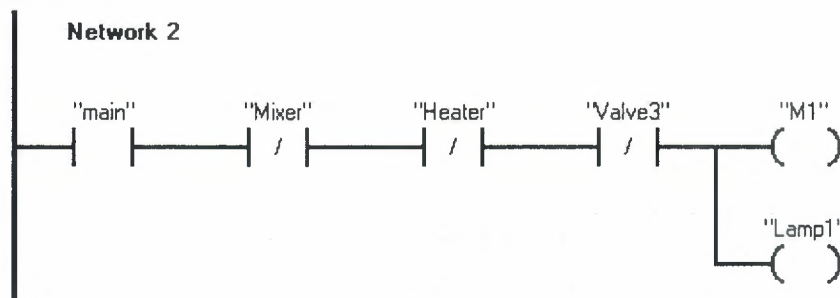
2.4 Networks Explanation:

Network # 1:



“Switch” contact is normally open contact in the program. When the “switch” contact is on, it will make “main” coil on (“logic 1”). Thus normally open contact of “main” will let power flow logically in anywhere in the program.

Network # 2:

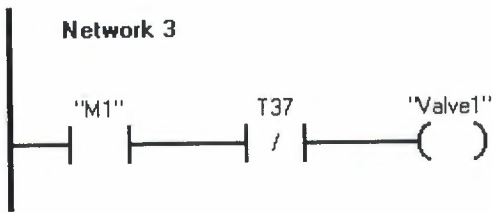


In the Network1, we set “main” coil on. Now all the operations depend on the operation of the switch so that, we put normally open contact of “main” coil in series (series because it is must condition not in parallel).

Motor operates (“M1” coil is on) when “main” contact is on and mixer-heater-Valve3 is in off position (it means that when mixer or heater or Valve3 is on it will draw the motor to off state).

When these conditions are satisfied “M1” coil will be on (it operates motor by driving it) and makes “Lamp1” coil on (Lamp1 is on) at the same time (because of parallel connection with “M1”). Here, “Lamp1” coil is a logical statement but Lamp1 is physical electrical equipment (like physical and logical disks in a computer).

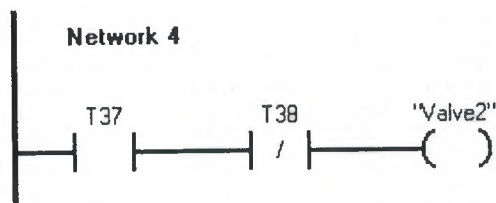
Network # 3:



In Network3, we put normally open contact of "M1" declaring that Valve1 will be operating due to "Valve1" coil on-state if the motor is on and T37 is off. T37 is a normally closed contact of Time-On Delay (TON) type timer. Here, closed contact says that when T37 named timer is on T37 normally closed will be off.

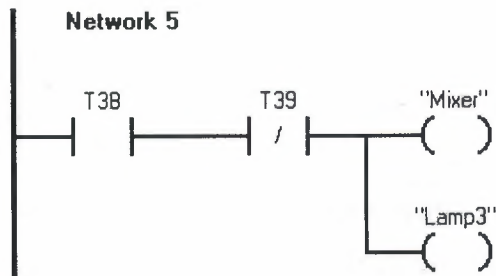
This structure is used especially when we want to operate a device for a specific time and stop it i.e. operating motor for five minutes. T37 is for operating Valve1 60 seconds after motor operation started. When preset value of T37 is reached, Valve1 will be stopped. Notice that motor will continue because it is not depending on T37.

Network # 4:



Valve2 operates when T37 normally open contact (N.O contact) is on and T38 normally closed contact (N.C contact) is off. Here, T37 N.O contact is for operating Valve2 after 60 seconds of motor start and T38 N.C contact is for operating Valve2 for 60 seconds duration. Valve2 will be off when T38 reaches preset time.

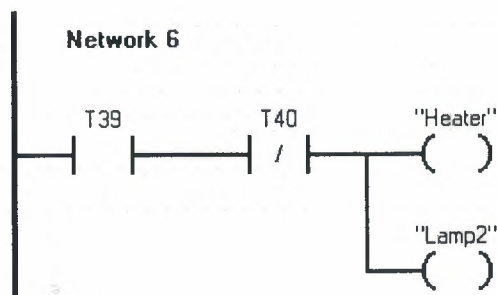
Network # 5:



Mixer operates after Valve2 stops and continues operation 30 seconds. This is provided by putting N.O contact of T38. Operating mixer 30 seconds is made by T39 N.C contact. T39 timer starts timing after mixer started and reaches preset value after 30 seconds later. N.C contact of T39 becomes off when preset value of timer T39 is reached.

Lamp3 indicates mixer is operating and it becomes on at the same time with the mixer. Therefore, we made "Lamp3" parallel connection with "Mixer".

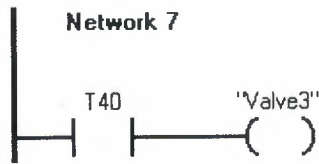
Network # 6:



Heater operates after mixer stops and continues operation 30 seconds. We can do this task by putting N.O contact of T39 timer (T39 starts timing after mixer is on). Here N.C contact T40 is for operating heater 30 seconds only. When heater is on, T40 starts timing and heater will stop when T40 preset time is reached.

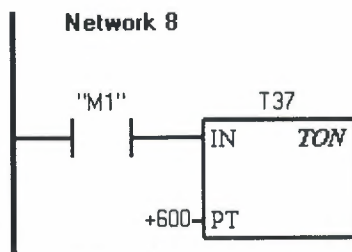
Lamp2 indicates heater is operating and it becomes on at the same time with heater. For this application, we made "Lamp2" parallel connection with "Heater".

Network # 7:



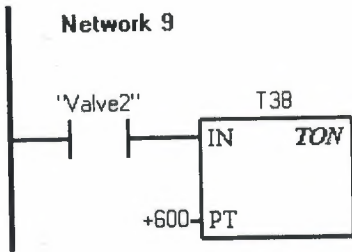
Valve3 operates after heater stopped. T40 timer becomes on when it reached its preset time value. This value makes 30 seconds delay which means that heater completed its operation. Here, there is no time limit for Valve3 operation. It stops when the switch(for start) is turned off manually.

Network # 8:



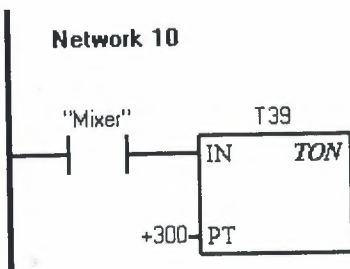
T37 is a TON (Time-On Delay) type timer and activates (becomes on) when it reached the preset time value. Here, it is used to operate Valve1 for 60 seconds. It is used in Network3 and Network4. In Network3, N.C contact of T37 let the Valve1 to operate for 60 seconds and when the preset time is reached the N.C contact will be off and stop Valve1. For Network4, it activates Valve2 after 60 seconds of Valve1 operation.

Network # 9:



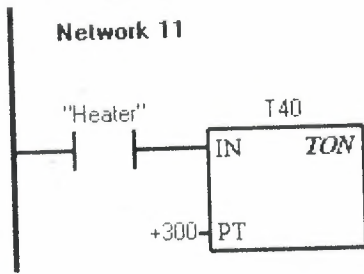
T38 is a TON type timer and it is used in Network4 and Network5. In Network4 its N.C contact lets Valve2 to operate for 60 seconds and when the preset time is reached, it stops Valve2 due to normally closed contact behavior. In Network5, it forces Mixer to start after 60 seconds of Valve2.

Network # 10:



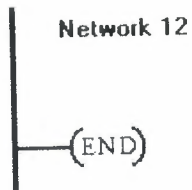
T39 is a TON type timer and it is used in Network5 and Network6. It is used in the same manner as above operations of Timers. In Network5, its N.C contact allows mixer to operate for 30 seconds after preset time value is reached, N.C contact will be off(logic "0") due to closed contact. In Network6, after 30 second of mixer operation, heater must be operating. This can be done by putting N.O contact of T39. Thus, when the preset time value is reached the heater will be on. The only way to stop it after 30 seconds can be done by putting N.C contact of T40 timer. It will be explained in the next page.

Network # 11:



T40 is a TON type timer and it is used in Network6 and Network7. In Network6, its N.C contact is used to operate the heater for 30 seconds and stop it at the end of 30 second same as above timers T37, T38 and T39. In Network7, it is used to start Valve3 after heater stopped. That is to say after 30 seconds of heater operation it will stop heater due to N.C contact in Network6 and starts Valve3 operation. N.O contact of T40 is used in Network7 and when the preset time is reached, it makes Valve3 on. Valve3 will not stop its operation. The only way to stop Valve3 is to move switch to its off position.

Network # 12:



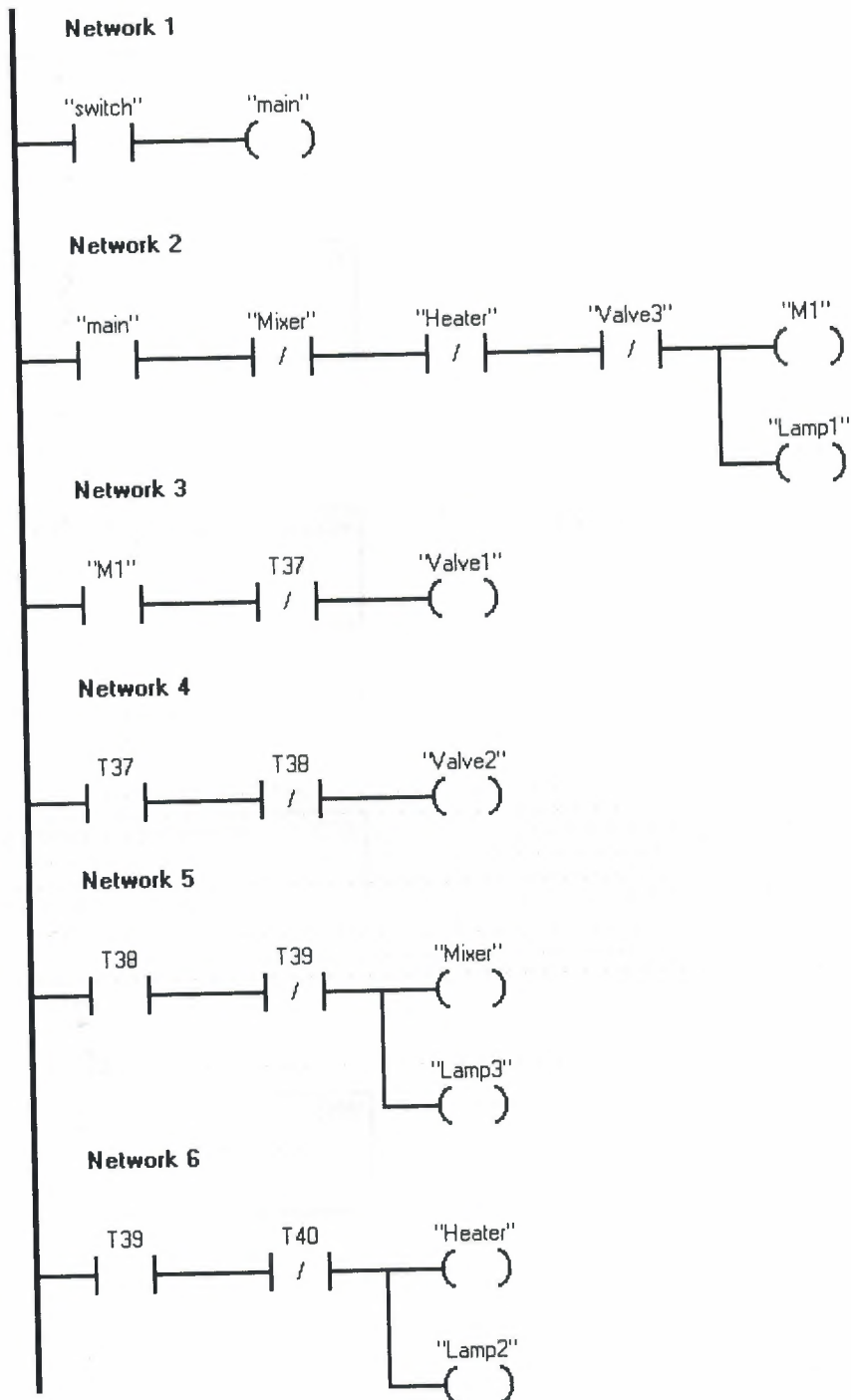
Network12 states that PLC program finished. It is an obligation to end the PLC program by END command. Otherwise, the PLC will not be compiled and downloaded to PLC. Be attention here the parallel connections for the lamps its electrical parallel connection not PLC parallel connection in the plc program.

2.5 Equipment of the Experiment

- 1- Three Tanks:
 - a- Main Tank.
 - b- Two Sub-Tanks.

- 2- Three Valves `Electro Mechanical Valve`.
- 3- Two Ac Motors
 - a- Mixer Motor.
 - b- Water Pump.
- 4- Heater.
- 5- Three Lumps (one orange, and two red).
- 6- Switch (on/off).
- 7- Four Pipes.
- 8- Pc.
- 9- Relay.
- 10- Siemens Semitic S7-200.....240 V

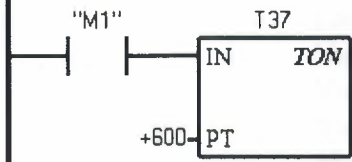
2.6 PLC Program of the Experiment



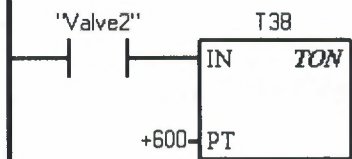
Network 7



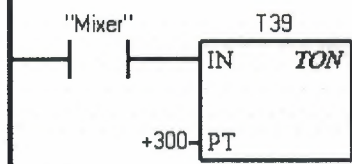
Network 8



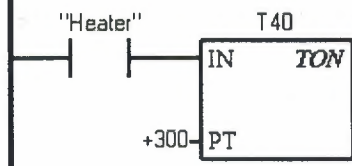
Network 9



Network 10



Network 11



Network 12

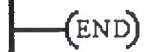


Table 2 Symbol Table of PLC Program

Symbol Name	Address	Comments
switch	I0.0	
main	I0.1	
M1	Q0.0	
Lamp1	Q0.1	Red1 Lamp
Lamp2	Q0.2	Orange Lamp
Lamp3	Q0.3	Red2 Lamp
Valve1	Q0.4	
Valve2	Q0.5	
Valve3	Q0.6	
Heater	Q0.7	
Mixer	Q1.0	

2.7 Program Title Comments

```
//  
//PROGRAM TITLE COMMENTS  
//  
//Press F1 for help and example program  
//  
NETWORK //NETWORK TITLE (single line)  
//  
//NETWORK 1 // Starting the system  
//  
LD I0.0  
= I0.1  
  
NETWORK 2 // Starting the Motor and Lamp1  
LD I0.1  
UN Q1.0  
UN Q0.7  
UN Q0.6  
= Q0.0  
= Q0.1  
  
NETWORK 3 // Opening Valve1 with Timer T37  
LD Q0.0  
UN T37  
= Q0.4
```

NETWORK 4 // Opening Valve2 with Timer T38

```
LD T37
UN T38
= Q0.5
```

NETWORK 5 // Starting the Mixer and Lamp3 with Timer T39

```
LD T38
UN T39
= Q1.0
= Q0.3
```

NETWORK 6 // Starting the Heater and Lamp2 with Timer T40

```
LD T39
UN T40
= Q0.7
= Q0.2
```

NETWORK 7 // Opening Valve3 when T40 Finished

```
LD T40
= Q0.6
```

NETWORK 8 // Start Timer T37 when Motor start working

```
LD Q0.0
TON T37, +600
```

NETWORK9 // Start Timer T38 when Valve2 opening

```
LD Q0.5
TON T38, +600
```

NETWORK 10 // Start Timer T39 when Mixer start working

```
LD Q1.0
TON T39, +300
```

NETWORK 11 // Start Timer T40 when Heater start working

```
LD Q0.7
TON T40, +300
```

NETWORK 12 // End the Main Program

```
MEND
```



Figure 2.3 the Experimental Review2

SUMMARY

In this chapter discussed the design of the experiment, the operation of the experimental, the equipment of the experiment, the program of controlling the experiment (liquid level control by PLC), and the writing the project all the explanations of the networks are presented.

CONCLUSION

The summary of this project is divided into two parts:

The first part which is presented about the general information about the PLC, what the PLC is, the History of the PLC, and how to great a program and to run according to get the aim of the project.

In the second part the experimental setup, the equipment that used in the experimental, and the way the connection of the equipment in the experimental are presented.

A liquid level control system is build practically and operated. The system is build such as to be used for experimental purpose in the PLC laboratory of the Electrical and Electronic Engineering Department.

The experimental procedure and the steps were carried out are presented, and the experimental 6 outputs and these 6 outputs contacts into the program of the experimental in order to control liquid level by using PLC.

The aim of this project is to control the liquid level by using the PLC.

In this experiment we can connect sensors for tank1 and tank2. This sensors aim is to specify the liquid level inside the tanks to control the motor.

If the tank is empty the sensor will give us a signal to stop the motor working or if we want a specific level for liquid this sensor will give us a signal to stop the motor working at that level, but in this case the output numbers will increase and we need plc part with more output.

REFERENCES

- [1]. Petruzella, F., Programmable Logic Controllers, Second Edition, McGraw-Hill Publishing Co., 1998.
- [2]. Crispin, A.J., "Programmable Logic Controllers and Their Engineering Applications", Books Britain, 1996.
- [3]. Bolton, w., Programmable Logic Controllers: An Introduction, Butterworth-Heinemann, 1997.
- [4]. Stenerson, J., "Fundamentals of Programmable Logic Controllers, Sensors and Communications", Prentice Hall, 1998.
- [5]. Özgür Özerdem, Programmable Logic Controllers and Programming, Iefkoşa, 2001.
- [6]. Webb, J.W., Reis, R.A., "Programmable Logic Controllers, Principles and Applications", Prentice Hall, 1995.