NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

DELTA WYE START OF THE MOTOR WITH PLC

Graduation Project
EE – 400

Student: Muammer Özavcı (20032991)

Supervisor:Asisst.Prof.Dr Özgür Cemal Özerdem

NICOSIA 2007

ACKNOWLEDGMENTS

First I want to thank Asisst.Prof.Dr. Özgür Cemal Özerdem to be my advisor. under his guidance, I succesfully overcome many difficulties and learn a lot about Programmable Logic Controllers. In each discussion,he explained my questions patiently, and I felt my quick progress from his advises . He always helps me a lot either in my study or my life. I asked him many questions in Logic Circuit Desing and Programmable Logic Controllers and he always answered my questions quickly and in detail.

Special thanks mrs Samet Biricik. With his kind help, I use Siemens S7-200 succesfully to perform computational environment.

Finally, I want to thank my family, especially my parents.Without their endless support and love for me, I would never achieve my cuurent position. I wish my mother lives happily always, and my father in the heaven be proud of me.

ABSTRACT

As PLC technology has advanced, so have programming languages and communications capabilities, along with many other important features. Today's PLCs offer faster scan times, space efficient high-density input/output systems, and special interfaces to allow non-traditional devices to be attached directly to the PLC. Not only can they communicate with other control systems, they can also perform reporting functions and diagnose their own failures, as well as the failure of a machine or process. Size is typically used to categorize today's PLC, and is often an indication of the features and types of applications it will accommodate.

In this graduation project Simatic S7-200 CPU 212 8 inputs, 6 outputs, 240 volt AC PLC is used, and an automation of a delta wye start of the motor is realized by PLC programs.

TABLE OF CONTENTS

AKNOWLEDGEMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	Iii,iv
INTRODUCTION	v
Chapter 1	
ROGRAMMABLE LOGIC CONTROLLERS (PLC)	
1.1 What is a PLC	1
1.2 History of PLC	2
1.3 General Physical Build System	3
1.3.1 Compact PLCs	3
1.3.2 Modular PLCs	4
1.4 Programmable Controller	4
1.4.1 Overview	4
1.4.2 Background	5
1.4.3 Terminology PC or PLC	6
1.5 Types of PLC	7
1.5.1 Small PLCs	8
1.5.2 Medium-Sized PLC	8
1.5.3 Large PLCs	9
1.5.4 Remote Input\Output	10
1.5.5 Programming Large PLCs	10
1.5.6 Developments	11
1.6 Today's PLC	11
Chapter 2	
STRUCTURE AND OPERATION OF PLC	
2.1 Basic Functional Structure of Programmable Controller System	12
2.1.1 Central Processing Unit(CPU)	15
2.1.2 Memory	15
2.1.3 Contacts and Coils	16
2.1.4 Data Storage	16
2.1.5 Functions	17
2.2 PLC Operation	17
2.3 Response Time	19
2.3.1 Response Time Concerns	19
2.4 Relays	22
2.4.1 Replacing Relays	23
2.5 Basic Instructions	25
2.5.1 Load	26
2.5.2 Loadbar	26
2.5.3 Out	27
2.5.4 Outbar	28
2.6 A Simple Example	28
2.7 PLC Registers	30
2.8 A Level Application	32
2.8.1 The Program Scan	34
2.9 Latch Instructions	37
2.10 Counters	38
2.11 Timers	42
2.11.1Timer Accuracy	47
2.12 One-Shots	49
2.13 Shift Registers	52
2.14 Getting and Moving Data	58

2.15 Math Instructions 61
2.16 DC Inputs 65
2.17 AC Inputs 68
2.18 Relay Outputs 70
2.19 Transistor Outputs 72
Chapter 3
SIEMENS S7-200 MICRO-CONTROLLER
3.1 Overview of an S7- 200 74
3.2 Introduction to the Simatic S7-200 Micro PLC 74
3.3 Comparing The Futures of the S7-200 Micro PLCs 75
3.3.1 Equipment Requirements 75
3.3.2 Capalities of the S7-200 CPUs 76
3.4 Major Components of the S7-200 Micro PLC 77
3.4.2 Expansion Modules 79
3.5 Programming Languages 80
3.5.1 Ladder Programs 80
3.5.2 STL Programs 80
3.6 CPU Memory 80
3.7 Simatic S7-200 Application Areas 81
3.7.1 The S7-200 Characteristics 81
3.7.2 Mechanical Features 82
3.7.3 Design Features 82
3.7.4 Benefits of the S7-200 82
Chapter 4
EQUIPMENT OF THE PROJECT
4.1 AC Motors 84
4.1.1 A Shalded Pole AC Motor 86
4.2 A Universal Motor 86
4.3 A 3- Phase AC Motor Demonstrator 87
4.3.1 Three Phase AC Motor Stator 88
4.4 Linear Motors 89
4.5 Contactor 90
4.5.1 Operating Principle 92
4.5.2 Ratings 93

Chapter 5
GRADUATION PROJECT

5.1 Delta Wye Start of the motor with PLC 94
5.2 Photograpies 95
5.3 Figures 98
CONCLUSION 99
REFERENCES 100

INTRODUCTION

A Programmable Logic 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 modules, various types of machines or processes." Which explains the device perfectly.

In the late 1960's PLC's were first introduced. The first PLC can be traced back to 1968 when Bedford Associates, a company in Bedford, MA, developed a device called a Modular Digital Controller for General Motors (GM). The MODICON, as it was known

The aim of this thesis is the control of the delta wye start of the Motor with SIMATIC S7-200 MICRO/WIN PLC.

The Thesis consists of the introduction, four chapters, and conclusion.
Chapter-1 presents history of PLC, what is PLC?, types of PLC's, hardware of PLC's, and chapter concluded with a brief information about today's PLC's.
Chapter-2 Structure and Opertion of PLC
Chapter-3 Presents the informations about Siemens SIMATIC S7-200
Chapter-4 Equipment of the project ac motors and contactors
Chapter-5 Delta wye connection of the motor with PLC

Chapter 1

PROGRAMMABLE LOGIC CONTROLLERS (PLC)

1.1 What is a PLC?

PLC are often defined as miniature industrial computers that contain hardware and software that is used to perform control functions. A PLC consists of two basic sections: the central processing unit (CPU) and the input/output interface system. The CPU, which controls all PLC activity, can further be broken down into the processor and memory system. The input/output system is physically connected to field devices (e.g., switches, sensors, etc.) and provides the interface between the CPU and the information providers (inputs) and controllable devices (outputs). To operate, the CPU "reads" input data from connected field devices through the use of its input interfaces, and then "executes", or performs the control program that has been stored in its memory system. Programs are typically created in ladder logic, a language that closely resembles a relay-based wiring schematic, and are entered into the CPU's memory prior to operation. Finally, based on the program, the PLC "writes", or updates output devices via the output interfaces. This process, also known as scanning, continues in the same sequence without interruption, and changes only when a change is made to the control program.

A programmable logic controller (PLC) 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.

PLC's are used in many real world applications. If there is industry present, changes are good that there is a PLC present. If you are involved in machining, packaging material handling, and 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 PLC.

For example, let's assume that when a switch turned on, a solenoid is wanted to turn on for 5 seconds and then turn it off regularly of how long the switch is on. It can be done with a simple external timer. But what will happen if the process included 10 switches and solenoids? 10 external timers would be needed. What will happen if the process also needed to count how many times the switches individually turned on? A lot of external counters would be needed.

As seen, if the process becomes more complicated, then we have to use a device the simplify that. We use PLC for this process. We can program the PLC to count its inputs and turn the solenoids for the specified time.

1.2 History of 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. 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 adhesion 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.

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!

The 80's saw an attempt to standardize communications with General Motor's manufacturing automation protocol(MAP). 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.

1.3 General Physical Build System

PLCs are separated into two according to their building mechanisms.

1.3.1 Compact PLCs

Compact PLCs are manufactured such that all units forming the PLC are placed in a case. They are low price PLC with lower capacity. Small or medium size machine manufacturers usually prefer them. In some types compact enlargement module is present.

1.3.2 Modular PLCs

Combining separate modules together in a board forms them. They can have different memory capacity, I / O numbers, power supply up to the necessary limits.

Some examples: SIEMENS S5-115U, SIEMENS S7-200, MITSUBISHI PC40, TEXAS INSTRUMENT PLC'S, KLOCKNER-MOELLER PS316, OMRON C200H.

1.4 Programmable Controller

1.4.1 Overview

The need for low cost, versatile and easily commissioned controllers has resulted in the development of programmable-control systems standard units based on a hardware CPU and memory for the control of machines or processes. Originally designed as a replacement for the hard-wired relay and timer logic to be found in traditional control panels, PLC's provides ease and flexibility of control based on programming and executing simple logic instructions. PLC's have internal functions such as timers, counters and shift registers, making sophisticated control possible using even the smallest PLC.

A programmable control operates by examining the input signals from a process and carrying out logic instructions on these input signals, producing output signal to drive process equipment or machinery. Standard interfaces build into PLC's allow them to be

directly connected to process actuators and transducers (pumps and valves) without the need for intermediate circuitry or relays.

Through using PLC's it became possible to modify a control system without having the disconnect or re-route a signal wire. It was necessary to change only the control program using a keypad or VDU terminal. Programmable controllers also require shorter installation and commissioning times than do hardwired systems. Although PLC's are similar to conventional computers in terms of hardware technology, they have specific features suited to industrial control:

1 Rugged, noise immune equipment;
2 Modular plug-in construction, allowing easy replacement\addition of units (input\output);
3 Standard input\output connections and signal levels;
4 Easily understood programming language (ladder diagram and function chart),
5 Ease of programming and reprogramming in-plant.

These features make programmable controllers highly desirable in a wide variety of industrial-plant and process-control situations.

1.4.2 Background

The programmable controller was initially conceived by a group of engineers from General Motors in 1968, where an initial specification was provided: the controller must be:

1- Easily programmed and reprogrammed, preferably in-plant to alter its sequence of operations.

2- Easily maintained and repaired- preferably using plug-in modules.

a) More reliable in plant environment.
b) Smaller than it is relay equivalent.

3- Cost competitive, with solid-state and relay panels than in use.

This provoked a keen interest from engineers of all disciplines in how to PLC could be used for industrial control. With this came demands for additional PLC capabilities and facilities, which were rapidly implemented as the technology became available. The instruction sets quickly moved from simple logic instructions to include counters, timers and shift registers, than onto more advanced mathematical functions on the machines. Developments hardware were also occurring, with larger memory and greater numbers of input / output points featuring on new models. In 1976 became possible to control remote I / O racks, where large numbers of distant I / O points were monitored updated via a communications link, often several hundred meters from the main PLC. The Allan-Bradley Corporation in America introduced a microprocessor-based PLC in 1977. It was based on an 8080 microprocessor but used an extra processor to handle bit logic instruction at high speed.

The increased rate of application of programmable controllers within industry has encouraged manufacturers to develop whole families of microprocessor-based systems having various levels of performance. The range of available PLC's now extends from small self-contained units with 20 digital I / O points and 500 program steps, up to modular systems with add-on function modules:

1 Analogue I/O;
2 PID control (proportional, integral and derivative terms);
3 Communications;
4 Graphics display;
5 Additional I/O;
6 Additional memory.

This modular approach allows the expansion or upgrading of a control system with minimum cost and disturbance.

Programmable controllers are developing at a virtually the same pace as microcomputers, with particular emphasis on small controllers, positioning\numeric control and communication networks. The market for small controllers has grown rapidly since the

early 1980's when a number of Japanese companies introduced very small, low cost units that were much cheaper than others available at that time. This brought programmable controllers within the budget of many potential users in the manufacturing and process industries, and this trend continues with PLC's offering ever-increasing performance at ever-decreasing cost.

1.4.3 Terminology-PC or PLC

There are several different terms used to describe programmable controllers, most referring to the functional operation of the machine in question:

1 PC programmable controller
2 PLC programmable logic controller
3 PBS programmable binary system

By their nature these terms tend to describe controllers that normally work in a binary environment. Since all but the smallest programmable controllers can now be equipped to process analogue inputs and outputs these labels are not representative of their capabilities. For these reason the overall term programmable controller has been widely adopted to describe the family of freely programmable controllers. However, to avoid confusion with the personal computer PC, this text uses the abbreviation PLC for programmable (logic) controller.

1.5 Types of PLC

The increasing demand from industry for programmable controllers that can be applied to different forms and sizes of control tasks has resulted in most manufacturers producing a range of PLC's with various levels of performance and facilities.

Typical rough definitions of PLC size are given in terms of program memory size and the maximum number of input\output points the system can support. Table 7.1 gives an example of these categories.

Table 1.1 Categories of PLC

PC size	Max I \ O points	Use memory size
Small	40 / 40	1K
Medium	128 / 128	4K
Large	> 128 / > 128	> 4K

However, to evaluate properly any programmable controller we must consider many additional features such as its processor, cycle time language facilities, functions, and expansion capabilities.

A brief outline of the characteristics of small, medium of large programmable controller is given below, together with typical applications.

1.5.1 Small PLCs

In general, small and 'mini' PLCs are designed as robust, compact units, which can be mounted on or beside the equipment to be controlled. They are mainly used the replaced hard-wired logic relays, timers, counters. That control individual items of plant or machinery, but can also be used to coordinate several machines working in conjunction with each other.

Small programmable controllers can normally have their total I/ O expanded by adding one or two I/ O modules, but if any further developments are required this will often mean replacement of the complete unit. This end of the market is very much concerned with non-specialist and users, therefore ease of programming and a 'familiar' circuit format are desirable. Competition between manufacturers is extremely fierce in this field, as they vie to obtain a maximum share in this partially developed sector of the market.

A single processor is normally used, and programming facilities are kept a fairly basic level, including conventional sequencing controls and simple standard functions: e.g. timers and counters. Programming of small PLC's is by way of logic instruction list (mnemonics) or relay ladder diagrams.

Program storage is given by EPROM or battery-backed RAM. There is now a trend towards EEPROM memory with on-board programming facilities on several controllers.

1.5.2 Medium-sized PLCs

In this range modular construction predominates with plug-in modules based around the Euro card 19-inch rack format or another rack mounting system. This construction allows the simple upgrading or expansion of the system. This construction allows the simple upgrading or expansion of the system by fitting additional I/ O cards in to the cards into the rack, since most rack, systems have space for several extra function cards. Boards are usually 'rugged zed' to allow reliable operation over a range of environments.

In general this type of PLC is applied to logic control tasks that can not be met by small controllers due to insufficient I\O provision, or because the control task is likely to be extended in the future. This might require the replacement of a small PLC, where as a modular system can be expanded to a much greater extent, allowing for growth. A medium-sized PLC may therefore be financially more attractive in the long term.

Communications of a single and multi-bit processor are likely within the CPU. For programming, standard instructions or ladder and logic diagrams are available. Programming is normally carried out via a small keypad or a VDU terminal. If different sizes of PLC are purchased from a single manufacturer, it is likely that programs and programming panels will be compatible between the machines.

1.5.3 Large PLCs

Where control of very large numbers of input and output points is necessary and complex control functions are required, a large programmable controller is the obvious choice. Large

PLC's are designed for use in large plants or on large machines requiring continuous control. They are also employed as supervisory controllers to monitor and control several other PLC's or intelligent machines. e.g. CNC tools.

Modular construction in Euro card format is standard, with a wide range of function cards available including analogue input output modules. There is a move towards 16-bit processor, and also multi-processor usage in order to efficiently handle a large range of differing control tasks.

For example;

7 16-bit processor as main processor for digital arithmetic and text handling.

8 Single-bit processor as co-or parallel processor for fast counting, storage etc.

9 Peripheral processor for handling additional tasks which are time-dependent or time-critical, such as:

- Closed-loop (PID) control
- Position controls
- Floating-point numerical calculations
- Diagnostic and monitoring
- Communications for decentralized
- Remote input\output racks.

This multi-processor solution optimizes the performance of the overall system as regards versatility and processing speed, allowing to PLC to handle very large programs of 100 K instructions or more. Memory cards can now provide several megabytes of CMOS RAM or EPROM storage.

1.5.4. Remote input\output

When large numbers of input / output points are located a considerable distance away from the programmable controller, it is uneconomic to run connecting cables to every point. A solution of this problem is to site a remote I/ O unit near to the desired I/ O points. This acts as a concentrator to monitor all inputs and transmit their status over a single serial

communications link to the programmable controller. Once output signals have been produced by the PLC they are feedback along the communications cable to the remote I/ O unit, which converts the serial data into the individual output signals to drive the process.

1.5.5 Programming large PLCs

Virtually any function can be programmed, using the familiar ladder symbols via a graphics terminal or personal computer. Parameters are passed to relevant modules either by incorporating constants in to the ladder, or via on screen menus for that module.

There may in addition be computer-oriented languages, which allow programming of function modules and subroutines.

There is progress towards standardization of programming languages; with programs becoming easier to over-view through improvement of text handling, hand improved documentation facilities. This is assisted by the application of personal computers as workstations.

1.5.6 Developments

Present trends include the integration of process data from a PLC into management databases, etc. This allows immediate presentation of information to those involved in scheduling, production and planning.

The need to pass process information between PC's and PLC sand other devices within a automated plants has resulted in the provision of a communications capability on all but the smallest controller. The development of local area networks (LAN) and in particular the recent MAP specification by General Motors (manufacturing automation protocol) provides the communication link to integrate all levels of control systems.

1.6 Today's PLC

As PLC technology has advanced, so have programming languages and communications capabilities, along with many other important features. Today's PLCs offer faster scan times, space efficient high-density input/output systems, and special interfaces to allow non-traditional devices to be attached directly to the PLC. Not only can they communicate with other control systems, they can also perform reporting functions and diagnose their own failures, as well as the failure of a machine or process. Size is typically used to categorize today's PLC, and is often an indication of the features and types of applications it will accommodate. Small, non-modular PLCs (also known as fixed I/O PLCs) generally have less memory and accommodate a small number of inputs and outputs in fixed configurations. Modular PLCs have bases or racks that allow installation of multiple I/O modules, and will accommodate more complex applications. When you consider all of the advances PLCs have made and all the benefits they offer, it's easy to see how they've become a standard in the industry, and why they will most likely continue their success in the future.

Chapter 2

STRUCTURE AND OPERATION OF PLC

2.1 Basic functional structure of a programmable controller system

The PLC mainly consists of a CPU, memory areas, and appropriate circuits to receive input/output data. Actually the PLC can be considered as a box full of hundreds or thousands of separate relays, counters, timers and data storage locations.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.

The general structure with main functional components in a programmable controller system is illustrated in the figures 2.1 and 2.2. These functions communicate with each other and with the signals of the machine/process to be controlled.

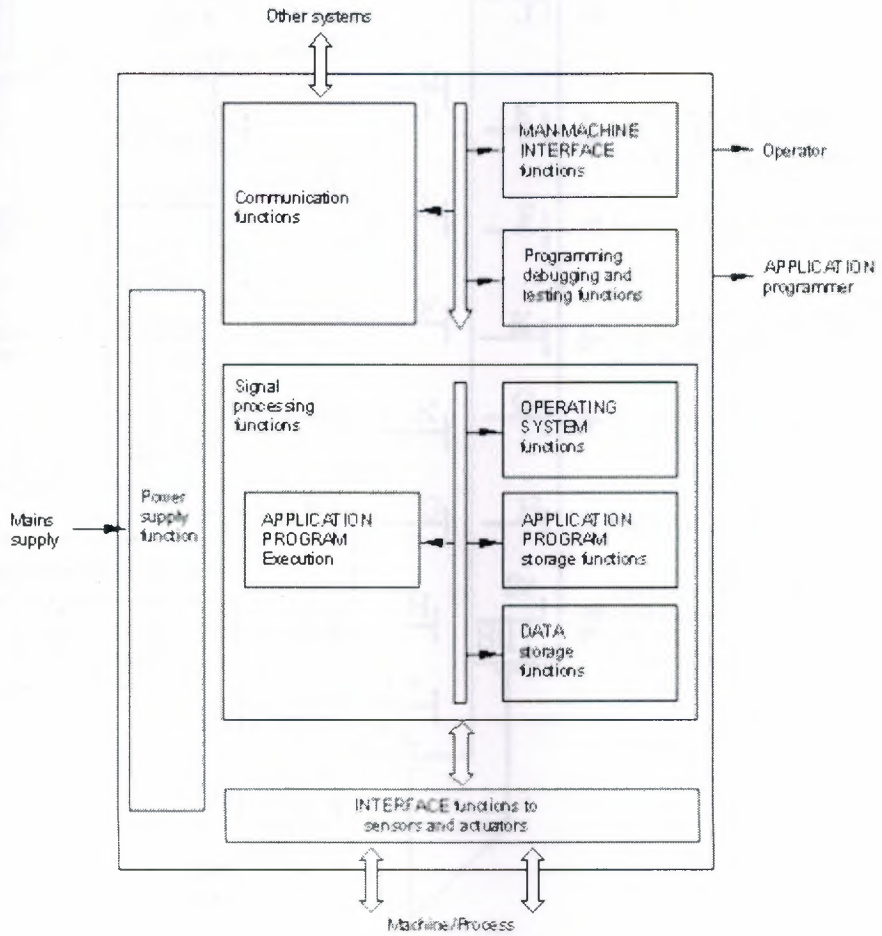


Figure 2.1 Basic functional structure of a PLC-system

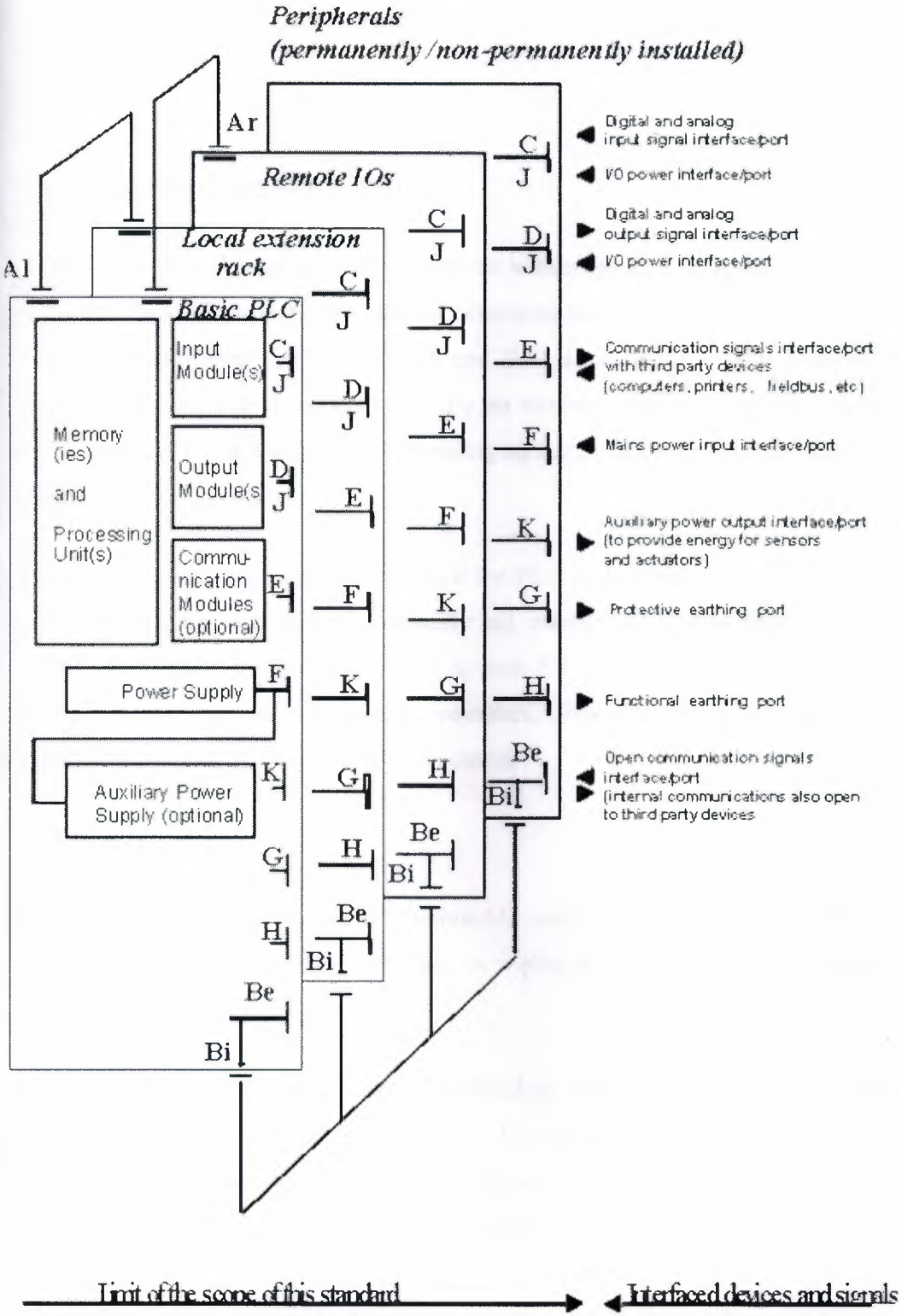


Figure 2.2 Typical Interface/Port Diagram of a PLC-system (from IEC 61131 - part 2)

2.1.1 Central Processing Unit (CPU)

The CPU controls and supervises all operations within the PLC, carrying out programmed instructions stored in memory. An internal communications highway or bus system carries information to and from the CP, memory and I/ O units, under control of the CPU. The CPU is supplied with a clock frequency by an external quartz crystal or RC oscillator, typically between 1 and 8 megahertz depending on the microprocessor used and the area of application.

The clock determines the operating speed of the PLC and provides timing\synchronization for all elements in the system. Virtually all modern programmable controllers are microprocessor based using a micro as a system CPU. Some larger PLC's also employ additional microprocessor to control complex, time-consuming functions such as mathematical processing, three terms PID control.

2.1.2 Memory

For program storage all modern programmable controllers use semiconductor memory devices such as RAM read\write memory, or a programmable read-only memory of the EPROM or EEPROM families.

In the virtually all cases RAM is used for initial program development and testing, as it follows changes to be easily made in program. The current trend is to be providing CMOS RAM because of it is very low power consumption, to provide battery back up to this RAM in order to maintain the contents when the power is removed from the PLC system. This battery has a lifespan of at least one year before replacement is necessary, or alternatively a rechargeable type may be supplied with the system being recharge whenever the main PLC power supply is on.

This feature makes programs stored in RAM virtually permanent. Many users operate their PLC systems on this basis alone, since it permits future program alterations if and when necessary.

After a program is fully developed and tested it may be loaded (blown) into a PROM or EPROM memory chip, which are normally cheaper than RAM devices. PROM programming is usually carried out with a special purpose programming unit, although many programmable controllers now have this facility built-in, allowing programs in the PLC RAM to be down loaded into a PROM IC placed in a socket provided on the PLC itself.

2.1.3 Contacts and Coils

Input Relays:

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:

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.

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, relays, or triacs depending upon the model chosen.

2.1.4 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!!

2.1.5 Functions

Communication function :

The communication function provides the data exchange with other systems (third party devices) such as other PLC-systems, robot controllers, computers, etc.

Human-machine interface (HMI) function :

The HMI function provides for interaction between the operator, the signal processing function and the machine/process.

Programming, debugging, testing and documentation functions :

These functions provide for application program generation and loading, monitoring, testing and debugging as well as for application program documentation and archiving.

Power supply functions :

The power supply functions provide for conversion and isolation of the PLC-system power from the mains supply.

2.2 PLC Operation

A PLC works by continually scanning a program. This scan cycle can be thought as consisting of 3 important steps. There are typically more than 3 but the important parts is focused on. Typically the others are checking the system and updating the current internal counter and timer values.

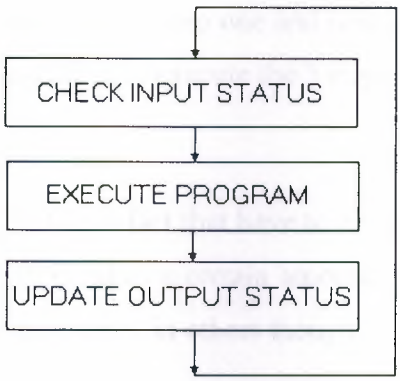


Figure 2.3 Operation Flow of PLC

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 the 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 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.

2.3 Response Time

The total response time of the PLC is a fact that have to be considered 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. 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. Eventually your mouth utters the words "That's a really ugly picture!" .

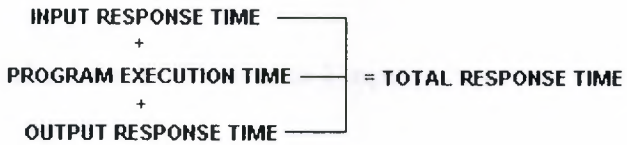


Figure 2.5 Total response time

2.3.1 Response Time Concerns

The PLC can only see an input turn on/off when it's looking. In other words, it only looks at its inputs during the check input status part of the scan. This part shows in an application what the response time is.

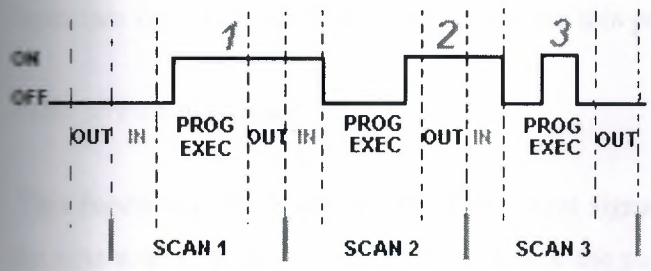


Figure 2.6 Application time graphic

In the diagram, input 1 is not seen until scan 2. This is because when input 1 turned on, scan 1 had already finished looking at the inputs.Input 2 is not seen until scan 3. This is also because when the input turned on scan 2 had already finished looking at the inputs. Input 3 is never seen. This is because when scan 3 was looking at the inputs, signal 3 was not on yet. It turns off before scan 4 looks at the inputs. Therefore signal 3 is never seen by the plc.

To avoid this the input should be on for at least **1 input delay time + one scan time.**

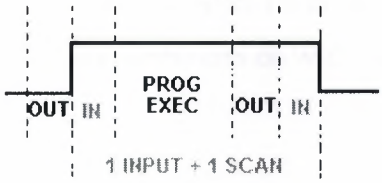


Figure 2.7 1 Input + 1 Scan time graphic

But what if it was not possible for the input to be on this long? Then the plc doesn't see the input turn on. There are 2 ways to get around this problem.

Pulse stretch function :

This function extends the length of the input signal until the plc looks at the inputs during the next scan.(i.e. it stretches the duration of the pulse.)

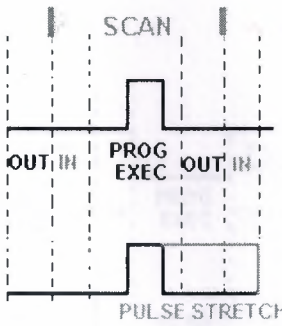


Figure 2.8 Pulse stretch function

Interrupt function :

This function interrupts the scan to process a special routine that you have written. i.e. As soon as the input turns on, regardless of where the scan currently is, the plc immediately stops what its doing and executes an interrupt routine. (A routine can be thought of as a

mini program outside of the main program.) After its done executing the interrupt routine, it goes back to the point it left off at and continues on with the normal scan process.

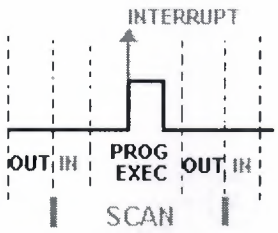


Figure 2.9 Interrupt function graphic

Now let's consider the longest time for an output to actually turn on. Let's assume that when a switch turns a load is needed to turn onconnected to the plc output. The diagram below shows the longest delay (worst case because the input is not seen until scan 2) for the output to turn on after the input has turned on. The maximum delay is thus **2 scan cycles - 1 input delay time.**

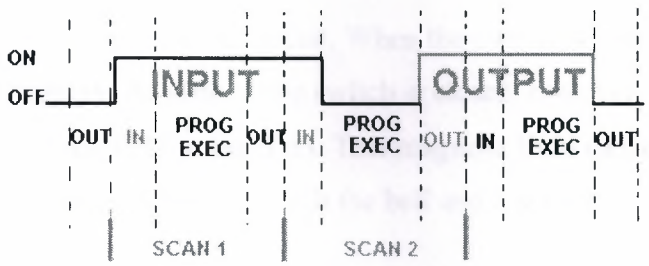


Figure 2.10 The maximum delay graphic

2.4 Relays

A relay can thought 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.

There is an example below. Bell is simply turned on whenever a switch is closed. There are 3 real-world parts. A switch, a relay and a bell. Whenever the switch closes we apply a current to a bell causing it to sound.

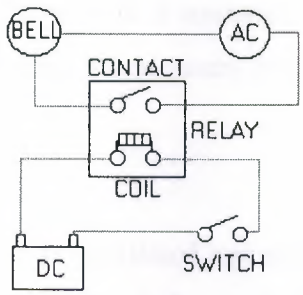


Figure 2.11 Operation of relay

Notice in the figure 2.11 that there are 2 separate circuits. The bottom(blue) indicates the DC part. The top(red) indicates the AC part.

Here dc relay is used to control an AC circuit. 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 it sounds.

Figure 2.12 A typical industrial relay

2.4.1 Replacing Relays

Let's use a plc in place of the relay. 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 its called a ladder diagram. One of these has to be created because, unfortunately, a plc doesn't understand a schematic diagram. It only recognizes code. Fortunately most PLCs have software which convert ladder diagrams into code. This shields users from actually learning the plc's code.

First step:

All of the items we're using have to be translated 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 its an input or an output.

First the battery is replaced with a symbol. This symbol is common to all ladder diagrams. Bus bars are drawn. 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 a symbol is given to the input. In this basic example we there is one real world input. (i.e. the switch) The symbol is given to the input that the switch will be connected to. This symbol can also be used as the contact of a relay.

Figure 2.13 A contact symbol

Next a symbol is given to the outputs. In this example one output (i.e. the bell) is used.the symbol is givn to the output that the bell will be physically connected to. This symbol is used as the coil of a relay.

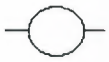


Figure 2.14 A coil symbol

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:

It must be told the plc where everything is located. In other words addresses have to be given all the devices. It starts with a blank road map in the PLCs town and an address is given to each item. The plc town has a lot of houses (inputs and outputs) but have to be figured out who lives where (what device is connected where). For now the input will be called and The output will be called "500".

Final step:

The schematic has to be converted into a logical sequence of events. This is much easier than it sounds. The program going to be written tells the plc what to do when certain events take place. In this example has to be said to the plc what to do when the operator turns on the switch.

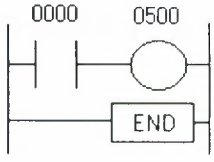


Figure 2.14 Ladder diagram of the example

The picture above is the final converted diagram.

2.5 Basic Instructions

In the ladder diagram there are few instructions shows inputs and outputs. These are explained at each part one by one.

2.5.1 Load

The load (LD) instruction is a normally open contact. The symbol for a load instruction is shown below.

Figure 2.15 A Load (contact) symbol

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

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

2.5.2 Loadbar

The Loadbar instruction is a normally closed contact. The symbol for a loadbar instruction is shown below.

Figure 2.16 A Loadbar (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 it can be said that the instruction is true. 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. 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 2.1 Logic values of Load and Loadbar

Logic State	Load	Loadbar
0	False	True
1	True	False

2.5.3 Out

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

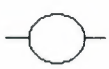


Figure 2.17 An Out (coil) symbol

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. This instruction can be thought as a normally open output. This instruction can be used for internal coils and external outputs.

2.5.4 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.

Figure 2.18 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. This instruction can be thought 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 2.2 Logic values of Out and Outbar

Logic State	Out	Outbar
0	False	True
1	True	False

2.6 A Simple Example

The two figures below shows simple ladder diagram with its real world external physically connected relay circuit and the differences are easily seen.

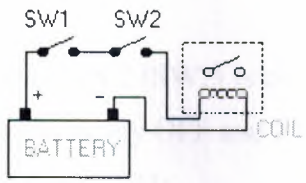


Figure 2.19 Circuit of example

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).

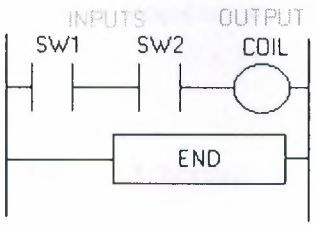


Figure 2.20 Ladder diagram of the example

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.

2.7 PLC Registers

The previous example is taken and switch 2 (SW2) is changed 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 figure 2.21:

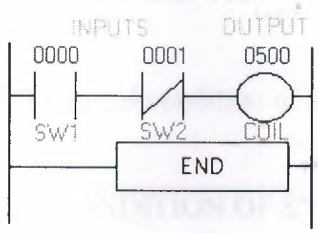


Figure 2.21 Addressed ladder diagram of previous example

Notice also that an address was given to each symbol (or instruction). This address sets aside a certain storage area in the PLCs 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 two different storage locations or registers were used.

Table 2.3 Register 00

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

Table 2.4 Register 05

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

In the tables 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.

Table 2.5 Logical condition of diagram

LOGICAL CONDITION OF 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 2.5, 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. Truth table will show all possible combinations of the status of the two inputs.

Table 2.6 The truth table

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.

2.8 A Level Application

In this application lubricating oil being dispensed from a tank is being controlled. This is possible by using two sensors. One near the bottom and one near the top, as shown in the picture (figure 2.22) below.

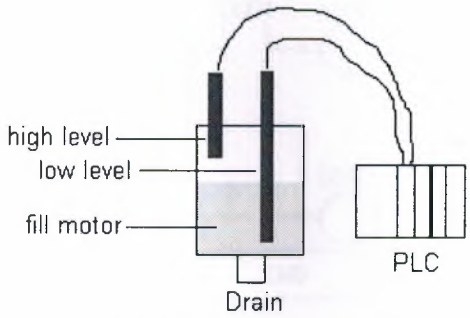


Figure 2.22 Application picture

Here, pump lubricating oil is wanted to pump into the tank by the fill motor until the high level sensor turns on. At that point motor is wanted to be turned off until the level falls below the low level sensor. Then the fill motor has to be turned on and the process will be repeated.

Here 3 I/O (i.e. Inputs/Outputs) are needed. 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.

Addresses have to be given to each input and output device. This lets the plc know where they are physically connected. The addresses are shown in the table 2.7:

Table 2.7 Address chart

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

Below (figure 2.23) is what the ladder diagram will actually look like. Notice that an internal utility relay is used in this example. The contacts of these relays can be used as many times as required. Here they are used twice to simulate a relay with 2 sets of contacts.

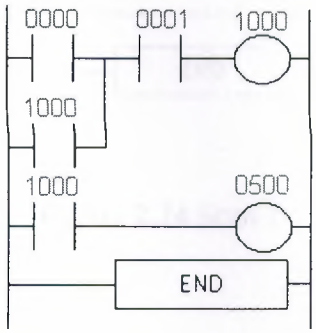


Figure 2.23 Ladder diagram of the application

The most common reason for using PLCs in 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 others include 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 a micro-plc with a few I/O is used, many internal relays aren't needed. If however, a large plc with 100's or 1000's of I/O is used, many more internal relays are certainly needed.

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.

2.8.1 The Program Scan

Below it is seen that what happens in this program scan by scan.

Initially the tank is empty. Therefore, input 0000 is true and input 0001 is also true.

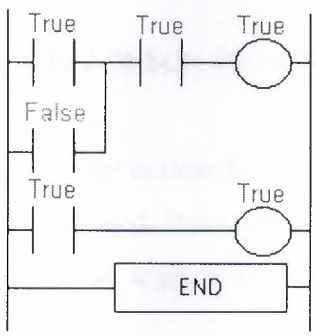


Figure 2.24 Scan 1

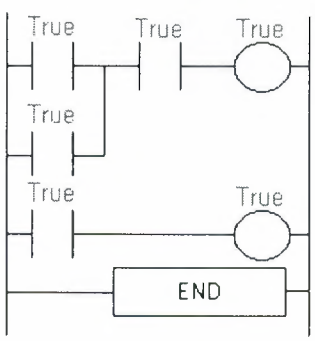


Figure 2.25 Scan 2-100

Gradually the tank fills because 500(fill motor) is on. After 100 scans the oil level rises above the low level sensor and it becomes open. (i.e. false)

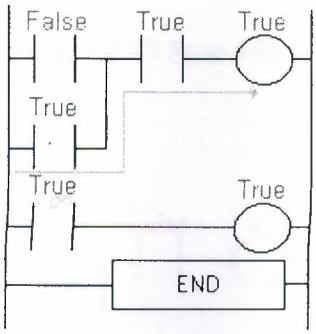


Figure 2.26 Scan 101-1000

Notice that even when the low level sensor is false there is still a path of true logic from left to right. This is why an internal relay is used. Relay 1000 is latching the output (500) on. It will stay this way until there is no true logic path from left to right. (i.e. when 0001 becomes false)

After 1000 scans the oil level rises above the high level sensor at it also becomes open (i.e. false)

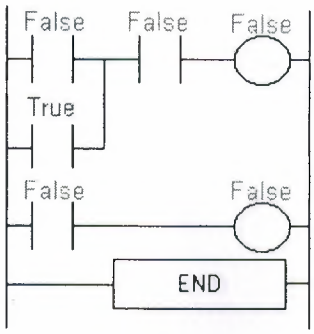


Figure 2.27 Scan 1001

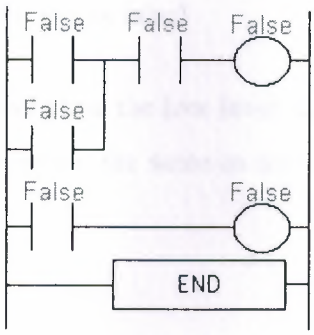


Figure 2.28 Scan 1002

Since there is no more true logic path, output 500 is no longer energized (true) and therefore the motor turns off.

After 1050 scans the oil level falls below the high level sensor and it will become true again.

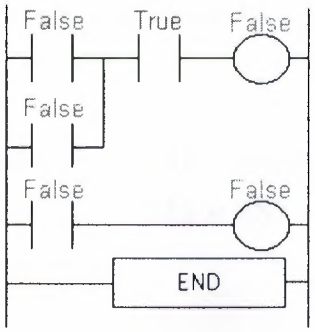


Figure 2.29 Scan 1050

Notice that even though the high level sensor became true there still is no continuous true logic path and therefore coil 1000 remains false!

After 2000 scans the oil level falls below the low level sensor and it will also become true again. At this point the logic will appear the same as scan 1 above and the logic will repeat as illustrated above.

2.9 Latch Instructions

Regular output coils are an essential part of our programs but must be remembered that they are only true when all instructions before them on the rung are also true. If they are not, then the output will become false.(turn off)

Think back to the bell example is done a few parts ago. What would've happened if a "push on/push off" switch couldn't be found? Then it would have been needed to keep pressing the button for as long as we wanted the bell to sound. (A momentary switch) The latching instructions let users use momentary switches and program the plc so that when it is push one time the output turns on and when it is pushed another time the output turns off.

Let's do a real world example to understand well;

Picture the remote control for TV. It has a button for on and another for off. When on button is pushed the TV turns on. When off button is pushed the TV turns off. It is not

essary to keep pushing the on button to keep the TV on. This would be the function of a ching instruction.

The latch instruction is often called a SET or OTL (output latch). The unlatch instruction is often called a RES (reset), OUT (output unlatch) or RST (reset). The diagram below (figure 2.30) shows how to use them in a program.

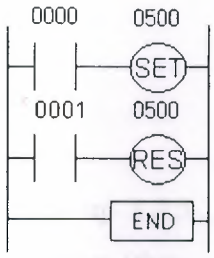


Figure 2.30 Sample application diagram of latch instructions

Here 2 momentary push button switches are being used. One is physically connected to input 0000 while the other is physically connected to input 0001. When the operator pushes switch 0000 the instruction "set 0500" will become true and output 0500 physically turns on. Even after the operator stops pushing the switch, the output (0500) will remain on. It is latched on. The only way to turn off output 0500 is turn on input 0001. This will cause the instruction "res 0500" to become true thereby unlatching or resetting output 0500.

One more think that have to be thought. What would happen if input 0000 and 0001 both turn on at the exact same time.

Will output 0500 be latched or unlatched?

To answer this question scanning sequence has to be thought. The ladder is always scanned from top to bottom, left to right. The first thing in the scan is to physically look at the inputs. 0000 and 0001 are both physically on. Next the plc executes the program. Starting from the top left, input 0000 is true therefore it should set 0500. Next it goes to the next

ung and since input 0001 is true it should reset 0500. The last thing is to reset 0500. Therefore on the last part of the scan when it updates the outputs it will keep 0500 off. (i.e. reset 0500).

2.10 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.

Kinds of counters:

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. The theory is all the same regardless of what the manufacturers call them. A counter is a counter is a counter...

Most manufacturers also include a limited number of high-speed counters. These are commonly called HSC (high-speed counter), CTH (Counter High-speed) or whatever. 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 that are being counted 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. (2xscan time = 2x2ms= 4ms)

To use them 3 things must be known:

1. Where the pulses that wanted 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 wanted to count before we react. Let's count 5 widgets before we box them, for example.
3. When/how the counter will be reset so it can count again. After 5 widgets are counted lets reset the counter, for example.

When the program is running on the plc the program typically displays the current or "accumulated" value so the current count value can be seen.

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.

Here are some of the instruction symbols that will be encountered (depending on which manufacturer be chosen) and how to use them. Should be remembered that while they may look different they are all used basically the same way.

Figure 2.31 Count-up counter

In this counter 2 inputs are needed.

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.

Cxxx is the name of the counter.If it is wanted to call it ''000'',then ''C000'' would be put.

yyyyy is the number of pulses we want to count before doing something. If 5 widgets are wanted to count before turning on a physical output to box them 5 would be put here. 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.

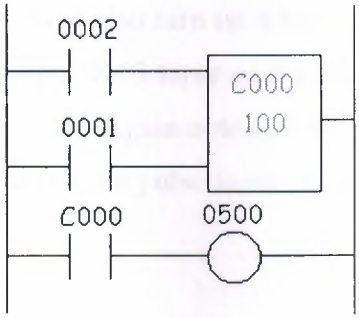


Figure 2.31 Count-up counter ladder diagram sample

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

Following figure (figure 2.32) is one symbol to encounter for an up-down counter. The same abbreviation is used as done for the example at the previous part (i.e. UDCxxx and yyyyy)

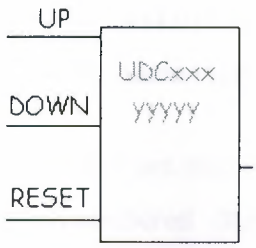


Figure 2.32 Count up/down counter

In this up-down counter has 3 inputs is needed to be assigned. The reset input has the same function as above. However, instead of having only one input for the pulse counting there are 2 now. One is for counting up and the other is for counting down. In this example the counter will be called UDC000 and a preset value of 1000 is given. (1000 total pulses will be counted) For inputs a sensor which will turn on input 0001 is used 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 it counts up and when input 0003 turns on it counts down. When 1000 pulses are reached, output 500 will be turned on. Again note that the counter accumulated value only changes at the off to on transition of the pulse input. The ladder diagram is shown below (figure 2.33).

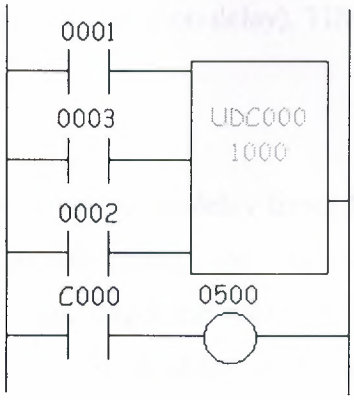


Figure 2.33 Count up/down counter ladder diagram sample

One important thing to note is that counters and timers can't have the same name (in most PLCs). This is because they typically use the same registers.

Counters certainly are an essential tool. They are also one of the least "standardized" basic instructions. However, always be remembered that the theory is the same from manufacturer to manufacturer!

2.11 Timers

Timer is exactly what the word says... It is an instruction that waits a set amount of time before doing something.

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 the sensor (input) turns on x-seconds are waited 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 the sensor (input) see a target, a solenoid (output) turns on. When the sensor no longer sees the target the solenoid is held 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, how long a sensor is on for during a 1 hour period is wanted to know. If one of the above timers is used they will keep resetting when the sensor turns off/on. This timer however, will give a total or accumulated time. It is often called an RTO (retentive timer) or TMRA (accumulating timer).

2 things are needed to know to use a timer:

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.

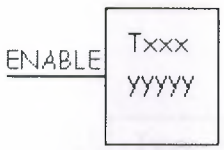
Why the weird numbers?

It is because most PLCs have 16-bit timers. For 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 millisecond 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 timebase. Some are TMH (high speed timer), TMS (super high speed timer) or TMRAF (accumulating fast timer)

The figure shown at the following page (figure 2.34) is a typical timer instruction symbol (depending on which manufacturer is chosen). Remember that while they may look different they are all used basically the same way.



Figue 2.34 On delay timer

The timer above (figure 2.34) 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 will be used later in the program. Remember that the duration of a tick (increment) varies with the vendor and the timebase used. (i.e. a tick might be 1ms or 1 second or...)

Below is the symbol shown on a ladder diagram:

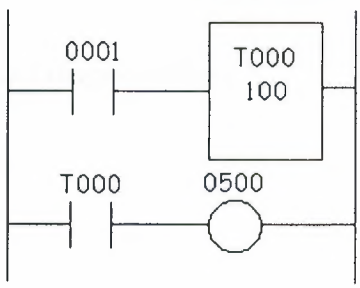


Figure 2.35 On delay timer sample ladder diagram

In this diagram when the input 0001 turns on, 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. 100ticks X 100ms = 10,000ms. 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.

An accumulating timer would look similar to this:

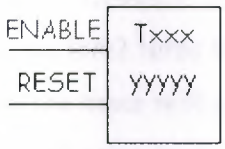


Figure 2.36 Accumulating (Retentive) timer

This timer 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 will be used later in the program. Remember that the duration of a tick (increment) varies with the vendor and the timebase used. (i.e. a tick might be 1ms or 1 second or...) If however, the enable input turns off before the timer has completed, the current value will be retained. When the input turns back on, the timer will continue from where it left off. The only way to force the timer back to its preset value to start again is to turn on the reset input.

The symbol is shown in the ladder diagram below.

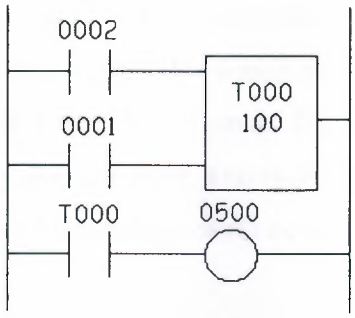


Figure 2.37 Accumulating (Retentive) timer sample ladder diagram

In this diagram when the input 0002 turns on, timer T000 (a 10ms increment timer) starts ticking. It will tick 100 times. Each tick (increment) is 10ms so the timer will be a 1000ms (i.e. 1 second) timer. 100ticks X 10ms = 1,000ms. When 1 second has elapsed, the T000 contacts close and 500 turns on. If input 0002 turns back off the current elapsed time will be retained. When 0002 turns back on the timer will continue where it left off. When input 0001 turns on (true) the timer T000 will reset back to 0 causing its contacts to turn off (become false) thereby making output 500 turn back off.

One important thing to note is that counters and timers can't have the same name (in most PLCs). This is because they typically use the same registers.

Always remember that although the symbols may look different they all operate the same way. Typically the major differences are in the duration of the ticks increments.

2.11.1 Timer Accuracy

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 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, obviously there will be an error. Further should 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. Below is a diagram illustrating the worst possible input error. Note from it that the worst possible input error would be 1 complete scan time + 1 program execution time. Remember that a program execution time varies from program to program. (depends how many instructions are in the program!)

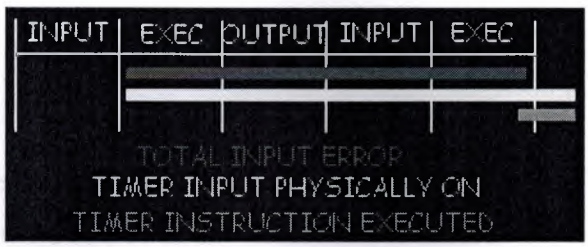


Figure 2.38 The worst possible input error

Shown below is a diagram illustrating the worst possible output error. Easily can be seen from it that the worst possible output error would be 1 complete scan time.

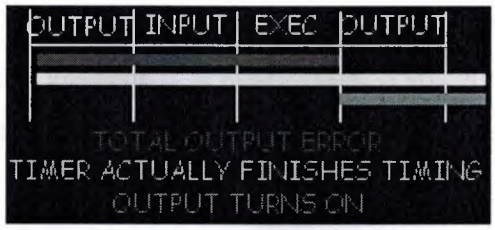


Figure 2.39 The worst possible output error

Based upon the above information can be seen that the total worst possible timer error would be equal to:

1 scan time + 1 program execution time + 1 scan time
= 2 scan times + 1 program execution time.

It means that even though most manufacturers currently have timers with 1ms increments they really shouldn't be used for durations less than a few milliseconds. This assumes that scan time is 1ms. If scan time is 5ms it is better not use a timer with duration less than about 15ms. The point is however, if what error is known to expect, then can be thought about whether this amount of error is acceptable for the application. In most applications this error is insignificant but in some high speed or very precise applications this error can be very significant.

Note that the above errors are only the "software errors". There is also a hardware input error as well as a hardware output error.

The hardware input error is caused by the time it takes for the plc to actually realize that the input is on when it scans its inputs. Typically this duration is about 10ms. This is because many PLCs require that an input should be physically on for a few scans before it determines it is physically on. (to eliminate noise or "bouncing" inputs)

The hardware output error is caused by the time it takes from when the plc tells its output to physically turn on until the moment it actually does. Typically a transistor takes about 0.5ms whereas a mechanical relay takes about 10ms.

2.12 One-shots

A one-shot is used to make something happen for only one scan. Most manufacturers have one-shots that react to an off to on transition and a different type that reacts to an on to off transition. Some names for the instructions could be difu/difd (differentiate up/down), sotu/sotd (single output up/down), osr (one-shot rising) and others. They all, however, end up with the same result regardless of the name.

Figure 2.40 One-shot Instruction

Above is the symbol for a difu (one-shot) instruction. A difd looks the same but inside the symbol it says "difd". Some of the manufacturers have it in the shape of a box but, regardless of the symbol, they all function the same way.

Let's now setup an application to see how this instruction actually functions in a ladder. This instruction is most often used with some of the advanced instructions where some things that must happen only once are done. Let's set up a flip/flop circuit. In simple terms, a flip/flop turns something around each time an action happens. Here a single pushbutton switch is used. The first time the operator pushes it an output is wanted to turn on. It will remain "latched" on until the next time the operator pushes the button. When he does, the output turns off.

Here's the ladder diagram that does just that:

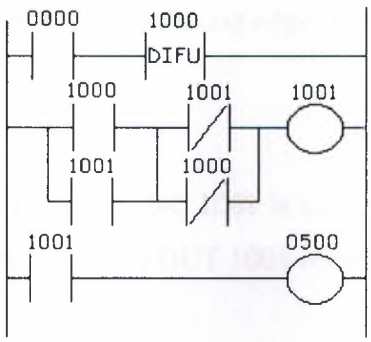


Figure 2.41 A flip/flop circuit

It is taken one step at a time:

Rung 1:

When NO (normally open) input 0000 becomes true DIFU 1000 becomes true.

Rung 2:

NO 1000 is true, NO 1001 remains false, NC 1001 remains true, NC 1000 turns false. Since there is a true path, (NO 1000 & NC 1001) OUT 1001 becomes true.

Rung 3:

NO 1001 is true therefore OUT 500 turns true.

Next Scan;

Rung 1:

NO 0000 remains true. DIFU 1000 now becomes false. This is because the DIFU instruction is only true for one scan. (i.e. the rising edge of the logic before it on the rung)

Rung 2:

NO 1000 is false, NO 1001 remains true, NC 1001 is false, NC 1000 turns true. Since still there is a true path, (NO 1001 & NC 1000) OUT 1001 remains true.

Rung 3:

NO 1001 is true therefore OUT 500 remains true.

After 100 scans, NO 0000 turns off (becomes false). The logic remains in the same state as "next scan" shown above. (difu doesn't react therefore the logic stays the same on rungs 2 and 3)

On scan 101 NO 0000 turns back on. (becomes true);

Rung 1:

When NO (normally open) input 0000 becomes true DIFU 1000 becomes true.

Rung 2:

NO 1000 is true, NO 1001 remains true, NC 1001 becomes false, NC 1000 also becomes false. Since there is no longer a true path, OUT 1001 becomes false.

Rung 3:

NO 1001 is false therefore OUT 500 becomes false.

Executing the program 1 instruction at a time makes this and any program easy to follow. Actually a larger program that jumps around might be difficult to follow but a pencil drawing of the registers sure does help!

2.13 Shift Registers

In many applications it is necessary to store the status of an event that has previously happened. If we must store many previous events and act upon the later, the shift register instruction is needed.

A register or group of registers is used 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.

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. Note that not all manufacturers have shift registers that shift data to the right but most all do have left shifting registers.

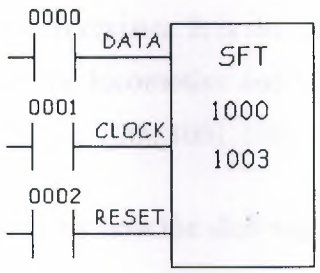


Figure 2.42 Shift register

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.

The 1000 inside the shift register symbol is the location of the first bit of our shift register. If the shift register is thought as a train then this bit is the locomotive. The 1003 inside the symbol above is the last bit of the 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)

Let's examine an application to see why/how the shift register can be used.

Imagine an ice-cream cone machine. There are 4 steps. First it is verified that the cone is not broken. Next ice cream is put inside the cone. (turn on output 500) Next peanuts are added. (turn on output 501) And finally sprinkles are added.(turn on output 502) A sensor is used to look at the bottom of the cone. (input 0000) If it is 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)

Here's what the ladder would look like:

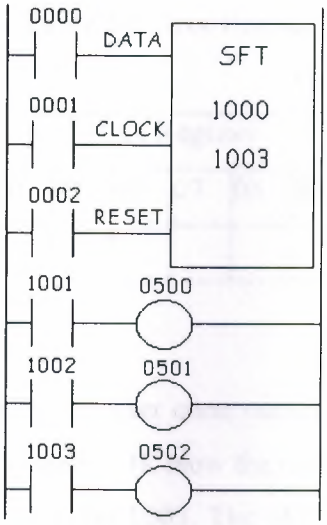


Figure 2.43 Shift Register sample ladder diagram

Let's now follow the shift register as the operation takes place. Here's what the 1000 series register (the register that is being shifted) looks like initially:

Table 2.8 10xx Register initially

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

A good cone comes in front of the sensor (input 0000). The sensor (data input) turns on. 1000 will not turn on until the rising edge of the encoder (input 0001). Finally the encoder now generates a pulse and the status of the data input (cone sensor input 0000) is transferred to bit 1000. The register now looks like:

Table 2.9 10xx Register

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

As the conveying system moves on, another cone comes in front of the sensor. This time it's a broken cone and the sensor remains off. Now the encoder generates another pulse. The old status of bit 1000 is transferred to bit 1001. The old status of 1001 shifts to 1002. The old status of 1002 shifts to 1003. And the new status of the data input (cone sensor) is transferred to bit 1000. The register now looks like:

Table 2.10 10xx Register

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

Since the register shows that 1001 is now on, the ladder says that output 0500 will turn on and ice cream is put in the cone.

As the conveying system continues to move on, another cone comes in front of the sensor. This time it's a good cone and the sensor turns on. Now the encoder generates another pulse. The old status of bit 1000 is transferred to bit 1001. The old status of 1001 shifts to 1002. The old status of 1002 shifts to 1003. And the new status of the data input (cone sensor) is transferred to bit 1000. The register now looks like:

Table 2.11 10xx Register

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

Since the register shows that 1002 is now on the ladder says that output 0501 will turn on and peanuts are put on the cone. Since 1001 now holds the status of a broken cone, 500 remains off in the ladder above and no ice-cream is inserted into this cone. As the conveying system continues to move on, another cone comes in front of the sensor. This time it's also a good cone and the sensor turns on. Now the encoder generates another pulse. The old status of bit 1000 is transferred to bit 1001. The old status of 1001 shifts to 1002. The old status of 1002 shifts to 1003. And the new status of the data input (cone sensor) is transferred to bit 1000. The register now looks like:

Table 2.12 10xx Register

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

Since the register shows that 1003 is now on the ladder says that output 0502 will turn on and sprinkles are put on the cone. Since 1002 now holds the status of a broken cone, 501 remains off in the ladder above and no peanuts are put onto this cone. Since the register

shows that 1001 is now on the ladder says that output 0500 will turn on and ice cream is put in that cone.

As the conveying system continues to move on, another cone comes in front of the sensor. This time it's another broken cone and the sensor turns off. Now the encoder generates another pulse. The old status of bit 1000 is transferred to bit 1001. The old status of 1001 shifts to 1002. The old status of 1002 shifts to 1003. And the new status of the data input (cone sensor) is transferred to bit 1000. The register now looks like:

Table 2.13 10xx Register

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

Notice that the status of our first cone has disappeared. In reality its sitting in location 1004 but it's useless to draw an application with 16 processes here. Suffice it to say that after the bit is shifted all the way to the left it disappears and is never seen again. In other words, it has been shifted out of the register and is erased from memory. Although it's not drawn, the operation above would continue on with each bit shifting on the rising edge of the encoder signal.

The shift register is most commonly used in conveyor systems, labeling or bottling applications, etc. Sometimes it's also conveniently used when the operation must be delayed in a fast moving bottling line. For example, a solenoid can't immediately kick out a bad can of beer when the sensor says its bad. By the time the solenoid would react the can would have already passed by. So typically the solenoid is located further down the conveyor line and a shift register tracks the can to be kicked out later when it's more convenient.

2.14 Getting and Moving Data

It is the turn of working with some data. This is what can be considered to be getting into the "advanced" functions of a plc. On the lines that follow two of the most popular ways are explored to get and manipulate data.

The single instruction is commonly called MOV (move). Some vendors also include a MOVN (move not). It has the same function of MOV but it transfers the data in inverted form. (i.e. if the bit was a 1, a 0 is stored/moved or if the bit was a 0, a 1 is stored/moved). The MOV typically looks like that shown on the next page.

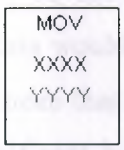


Figure 2.44 MOV instruction symbol

The paired instruction typically is called LDA (Load Accumulator) and STA (Store Accumulator). The accumulator is simply a register inside the CPU where the plc stores data temporarily while it is working. The LDA instruction typically looks like that shown below, while the STA instruction looks like that shown below to the right.

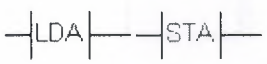


Figure 2.45 LDA and STA instructions

Regardless of whether the one symbol or two symbol instruction set that used (there is no choice as it depends on whose plc is used) they work the same way.

Let's see the single instruction first. The MOV instruction needs to know 2 things from user.

Source (xxxx):

This is where the data wanted to move is located. A constant could be written here (2222 for example). This would mean the source data is the number 2222. Also a location or address of where the data wanted to move could be written. If DM100 was written this would move the data that is located in data memory 100.

Destination (yyyy):

This is the location where the data will be moved to. An address is written here. For example if DM201 is written here the data would be moved into data memory 201. Also 0500 could be written here. This would mean that the data would be moved to the physical outputs. 0500 would have the least significant bit, 0501 would have the next bit... 0515 would have the most significant bit. This would be useful if a binary display, connected to the outputs, be had and wanted to display the value inside a counter for the machine operator at all times (for example).

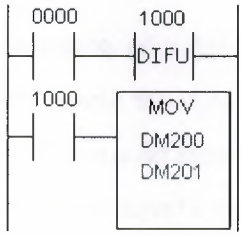


Figure 2.46 MOV instruction sample ladder diagram

The ladder diagram to do this would look similar to that shown above.

Notice that a "difu" instruction is being used here. The reason is simply because if the data wasn't would be moved during each and every scan. Sometimes this is a good thing (for

example if data is being acquired from an A/D module) but other times it's not (for example an external display would be unreadable because the data changes too much).

The ladder shows that each time real world input 0000 becomes true, difu will become true for only one scan. At this time Load 1000 will be true and the plc will move the data from data memory 200 and put it into data memory 201.Simple but effective. If, instead of DM200, 2222 had been written in the symbol, the number (constant) 2222 would have been written into DM201.

The two symbol instruction works in the same method but looks different. To use them two things must be supplied, one for each instruction:

LDA Instruction:

This instruction is similar to the source of a MOV instruction. This is where the data wanted to move is located. A constant could be written here (2222 for example). This would mean the source data is the number 2222. Also a location or address, of where the data we want to move is located, could be written. If DM100 was written this would move the data that is located in data memory 100.

STA Instruction:

This instruction is similar to the destination of a MOV instruction. An address is written here. For example if DM201 is written here the data would be moved into data memory 201. 0500 also could be written here. This would mean that the data would be moved to the physical outputs. 0500 would have the least significant bit, 0501 would have the next bit... 0515 would have the most significant bit. This would be useful if a there was a binary display connected to the outputs and the value is wanted to display inside a counter for the machine operator at all times (for example).

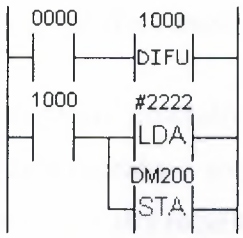


Figure 2.47 LDA and STA instructions sample ladder diagram

The ladder diagram to do this would look similar to that shown above. Here again notice that a one-shot is being used so that the move only occurs once for each time input 0000 becomes true. In this ladder the constant 2222 is being moved into data memory 200. The "#" is used by some manufactures to symbolize a decimal number. If 2222 was used this plc would think it meant address 2222. PLCs are all the same... but they are all different.

2.15 Math Instructions

In general, PLCs almost always include these math functions:

Addition:

The capability is to add one piece of data to another. It is commonly called ADD.

Subtraction:

The capability is to subtract one piece of data from another. It is commonly called SUB.

Multiplication:

The capability is to multiply one piece of data by another. It is commonly called MUL.

Division:

The capability is to divide one piece of data from another. It is commonly called DIV.

As seen with the MOV instruction there are generally two common methods used by the majority of plc makers. The first method includes a single instruction that asks us for a few key pieces of information. This method typically requires:

Source A:

This is the address of the first piece of data will be used in the formula. In other words it's the location in memory of where the first "number" is that used in the formula.

Source B:

This is the address of the second piece of data will be used in our formula. In other words it's the location in memory of where the second "number" is that used in the formula.

NOTE: typically with 2 pieces of data can be worked at a time. In other words directly a formula like 1+2+3 can't be worked with. It has to be broken up into pieces. Like 1+2=X then X+3= result.

Destination:

This is the address where the result of the formula will be put. For example, if 1+2=3, the 3 would automatically be put into this destination memory location.

Figure 2.48 ADD symbol

The instructions above typically have a symbol that looks like that shown above. The word ADD would be replaced by SUB, MUL, DIV, etc. In this symbol, the source A is DM100, the source B is DM101 and the destination is DM102. Therefore, the formula is simply whatever value is in DM100 + whatever value is in DM101. The result is automatically stored into DM102.

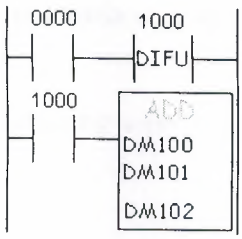


Figure 2.49 Math functions sample ladder diagram

Shown above is how to use math functions on a ladder diagram. Please note that a one-shot instruction is being used. As seen before, this is because if it wasn't used the formula would be executed on every scan. Odds are good that the function is only wanted to execute one time when input 0000 becomes true. If number 100 had been put into DM100 and 200 into DM101, the number 300 would be stored in DM102.(i.e. 100+200=300, right??)

Figure 2.50 ADD symbol (dual method)

The dual instruction method would use a symbol similar to that shown above. In this method, this symbol is given only the Source B location. The Source A location is given by the LDA instruction. The Destination would be included in the STA instruction.

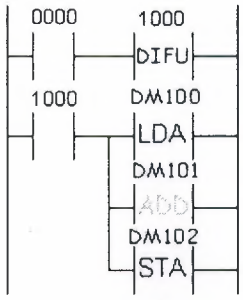


Figure 2.51 Dual method sample ladder diagram

Shown above is a ladder diagram showing what is meant. The results are the same as the single instruction method.

What would happen if the result had been greater than the value that could be stored in a memory location?

Typically the memory locations are 16-bit locations. In plain words this means that if the number is greater than 65535 (2^16=65536) it is too big to fit. Then an overflow is gotten. Typically the plc turns on an internal relay that tells an overflow has happened. Depending on the plc, different data would be in the destination location. (DM102 from example) Most PLCs put the remainder here.

Some use 32-bit math which solves the problem. (except for really big numbers!)

Many PLCs also include other math capabilities. Some of these functions could include:

1 Square roots
2 Scaling
3 Absolute value
4 Sine

5 Cosine

6 Tangent

7 Natural logarithm

8 Base 10 logarithm

9 X^Y (X to the power of Y)

10 Arcsine (tan, cos)

Some PLCs can use floating point math as well. Floating point math is simply using decimal points. In other words, can be said that 10 divided by 3 is 3.333333 (floating point). Or 10 divided by 3 is 3 with a remainder of 1(long division). Many micro/mini PLCs don't include floating point math. Most larger systems typically do.

2.16 DC Inputs

Typically, dc input modules are available that will work with 5, 12, 24, and 48 volts.

DC input modules allow to connect either PNP (sourcing) or NPN (sinking) transistor type devices to them. If a regular switch (i.e. toggle or pushbutton, etc.) is used typically don't have to worry about whether we wire it as NPN or PNP. Note that most PLCs won't let us mix NPN and PNP devices on the same module. When using a sensor (photo-eye, etc.), however, have to worry about its output configuration. Always verify whether it's PNP or NPN.

The difference between the two types is whether 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.

The figure 2.52 is what the outputs look like for NPN and PNP sensors.

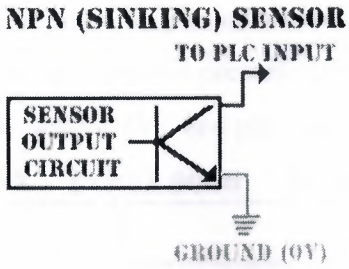


Figure 2.52 NPN sensor

On the NPN sensor one output is connected to the PLC's input and the other output to the power supply ground. If the sensor is not powered from the same supply as the plc, both grounds should be connected 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.

On the PNP sensor one output is connected to positive voltage and the other output to the PLC's 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.

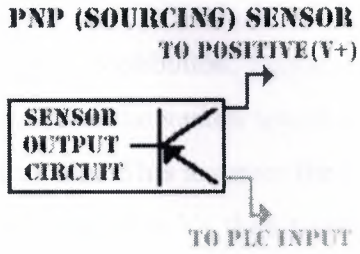


Figure 2.53 PNP sensor

Inside the sensor, the transistor is just acting as a switch. The sensors 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).

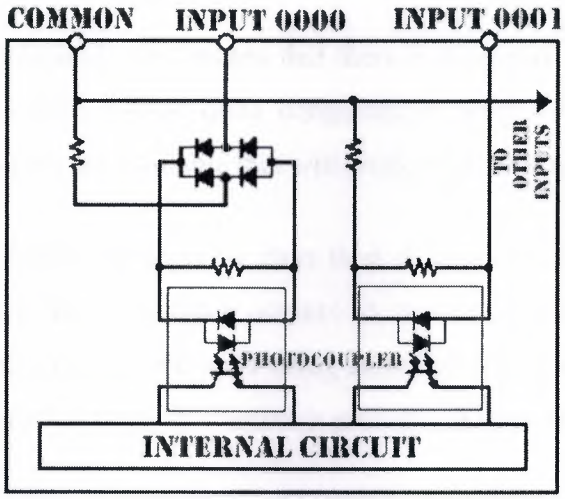


Figure 2.54 DC input module circuit

The only things accessible to the user are the terminals labeled COMMON, INPUT 0000, INPUT 0001, INPUT xxxx. 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 photocouplers 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.

2.17 AC Inputs

An ac voltage is non-polarized. This means that there is no positive or negative to "worry about". However, ac voltage can be quite dangerous to work with if working careless. Typically, ac input modules are available that will work with 24, 48, 110, and 220 volts.

AC input modules are less common these days than dc input modules. The reason is 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 the application is using a sensor it probably is operating on a dc voltage.

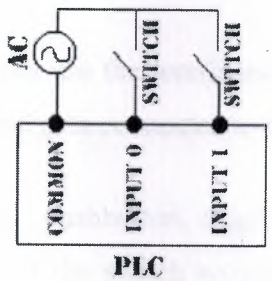


Figure 2.55 AC device connected to PLC

Typically an ac device is connected to the 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 Turkey is commonly black (neutral), red (hot) and green with yellow (3rd wire ground when applicable).

The PLC's ac input module circuit typically looks like this:

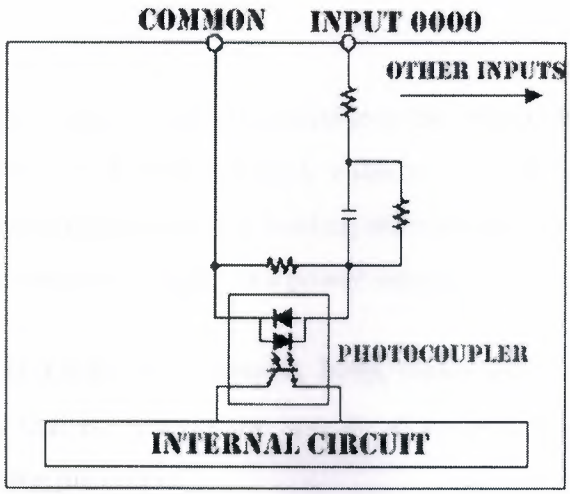


Figure 2.56 AC input module circuit

The only things accessible to the user are the terminals labeled COMMON, INPUT 0000, INPUT xxxx... 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 XXXX. The other end goes to the ac hot wire. This assumes the common terminal is connected to neutral.

The photocouplers are used to isolate the PLC's 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.

2.18 Relay Outputs

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 outputs. It is called load because output is loading with something. If no load is connected to the output (i.e. just connected directly to a power supply) the outputs would be damaged.

Some common forms of a load are a solenoid, lamp, motor, etc. These "loads" come in all sizes. Electrical sizes, that is. Always the specifications of the load should be checked before connecting it to the plc output.

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 is an air conditioning unit. 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 PLC's 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.

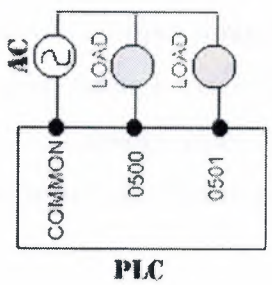


Figure 2.57 Output-relay connection

Shown above is a typical method of connecting the outputs to the plc relays. Although the 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 the power supply and the other end of the supply is connected to the load. The other half of the load gets connected to the actual plc output that is designated within the ladder program.

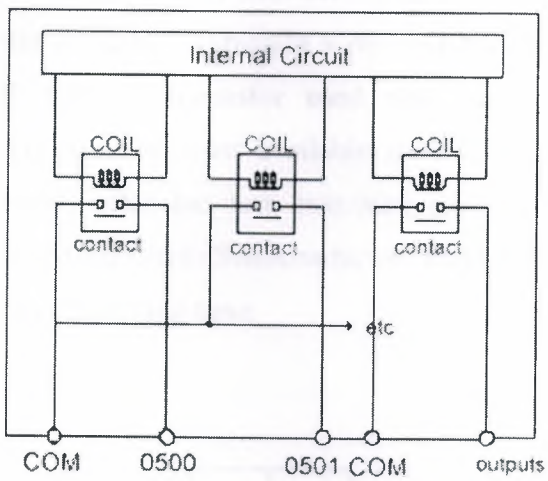


Figure 2.58 Relay circuit diagram

The relay is internal to the plc. Its circuit diagram typically looks like that shown above. When 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 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. The load will than have an open circuit and will therefore be off.

2.19 Transistor Outputs

The next type of output 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.

A transistor can be thought as a solid-state switch. Or more simply an electrical switch. A small current applied to the transistors "base" (i.e. input) lets 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.

Should also be kept in mind that as seen 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.

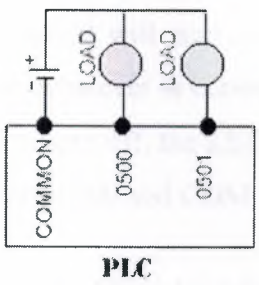


Figure 2.59 Transistor output-Output device connection

Shown above is how the output device is connected to the transistor output. 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 proper polarity must be always observed for the output. One end of the load is connected directly to V+ as shown above.

The figure 2.60 is a typical output circuit diagram for an NPN type output.

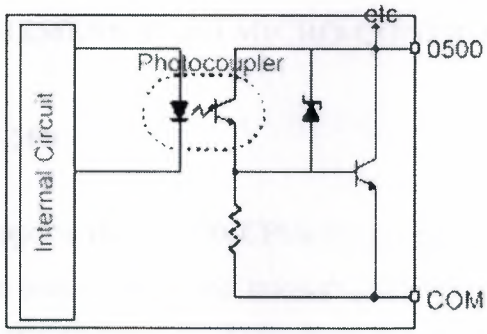


Figure 2.60 NPN type output circuit diagram

Notice that as seen with the transistor type inputs, there is a photocoupler isolating the "real world" from the internal circuit. When the ladder diagram calls for it, the internal circuit turns on the photocoupler by applying a small voltage to the LED side of the photocoupler. This makes the LED emit light and the receiving part of the photocoupler 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.

One other important thing to note is that a transistor typically cannot switch as large a load as a relay. 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.

Chapter 3

SIEMENS S7-200 MICRO-CONTROLLER

3.1 Overview of an S7-200

STEP 7-Micro/WIN supports the S7-200 CPUs by giving you the features to set up and manage your application project. A project consists of the program you enter with STEP 7-Micro/WIN, along with the documentation you write for the program and the configuration you set up for the CPU.

You have the option of selecting either Ladder or Statement List as your programming language. With the S7-200 CPUs, you have a basic program structure that gives you flexibility in setting up any subroutines or interrupts that you program.

3.2 Introduction to the Simatic S7-200 Micro PLC

The Simatic S7-200 series is a line of micro-programmable logic controllers (Micro PLCs) that can control a variety of automation applications. Figure 3.1 shows an S7-200 Micro PLC. The compact design, expandability, low cost, and powerful instruction set of the S7-200 Micro PLC make a perfect solution for controlling small applications. In addition, the wide variety of CPU sizes and voltages provides you with the flexibility you need to solve your automation problems.

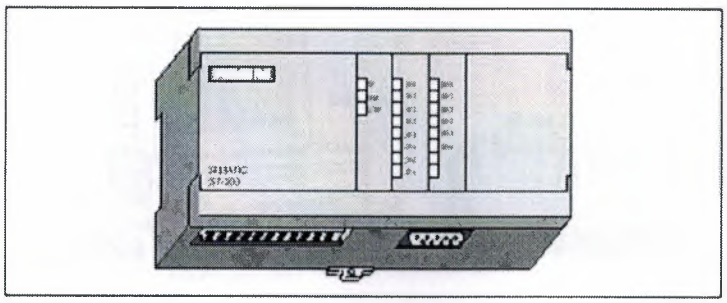


Figure 3.1 S7-200 Micro PLC

3.3 Comparing the Features of the S7-200 Micro PLCs

3.3.1 Equipment Requirements

Figure 3.2 shows the basic S7-200 Micro PLC system, which includes an S7-200 CPU module, a personal computer, STEP 7-Micro/WIN programming software, and a communications cable.

In order to use a personal computer (PC), you must have one of the following sets of equipment:

1 A PC/PPI cable

2 A communications processor (CP) card and multipoint interface (MPI) cable

3 A multipoint interface (MPI) card. A communications cable is provided with the MPI card.

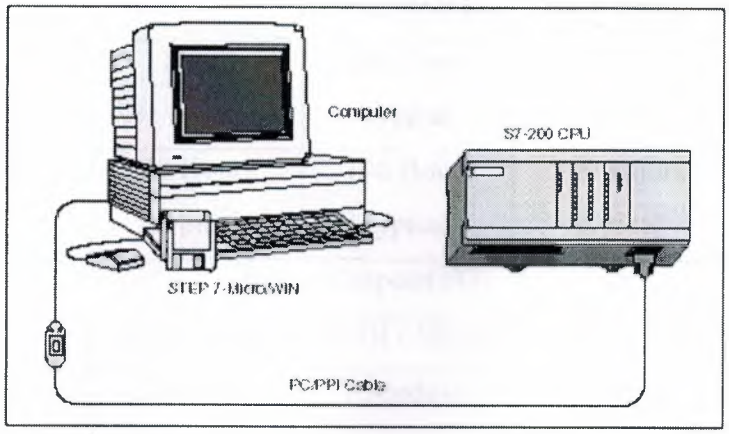


Figure 3.2 Components of an S7-200 Micro PLC System

3.3.2 Capabilities of the S7-200 CPUs

The S7-200 family includes a wide variety of CPUs. This variety provides a range of features to aid in designing a cost-effective automation solution. Table 3.1 provides a summary of the major features of each S7-200 CPU.

Table 3.1 Summary of the S7-200 CPUs

Feature	CPU 212	CPU 214	CPU 215	CPU 216
Physical Size of unit	160mm x 80mm x 62mm	197mm x 80mm x 62mm	218mm x 80mm x 62mm	218mm x 80mm x 62 mm
Memory				
Program (EEPROM)	512 words	2 k words	4 k words	4 k words
User Data	512 words	2 k words	2.5 k words	2.5 k words
Internal Memory Bits	128	256	256	256
Memory Cartridge	None	Yes	Yes	Yes

		(EEPROM)	(EEPROM)	(EEPROM)
Optional Battery Cartridge	None	200 Days typical	200 Days typical	200 Days typical
Backup (super capacitor)	50 Hours typical	190 Hours typical	190 Hours typical	190 Hours typical
Inputs/Outputs(I/O)				
Local I/O	8 DI / 6 DQ	14 DI / 10 DQ	14 DI / 10 DQ	24 DI / 16 DQ
Expansion Modules (max.)	2 Modules	7 Modules	7 Modules	7 Modules
Process-Image I/O Register	64 DI / 64 DQ	64 DI / 64 DQ	64 DI / 64 DQ	64 DI / 64 DQ
Analog I/O (expansion)	16 AI / 16 AQ	16 AI / 16 AQ	16 AI / 16 AQ	16 AI / 16 AQ
Selectable Input Filter	No	Yes	Yes	Yes

3.4 Major Components of the S7-200 Micro PLC

An S7-200 Micro PLC consists of an S7-200 CPU module alone or with a variety of optional expansion modules.

3.4.1 CPU Module

The S7-200 CPU module combines a central processing unit (CPU), power supply, and discrete I/O points into a compact, stand-alone device.

1 The CPU executes the program and stores the data for controlling the automation task or process.
2 The power supply provides electrical power for the base unit and for any expansion module that is connected.

3 The inputs and outputs are the system control points: the inputs monitor the signals from the field devices (such as sensors and switches), and the outputs control pumps, motors, or other devices in your process.

4 The communications port allows you to connect the CPU to a programming device or to other devices. Some S7-200 CPUs have two communications ports.

5 Status Lights provide visual information about the CPU mode (RUN or STOP), the current state of the local I/O, and whether a system fault has been detected.

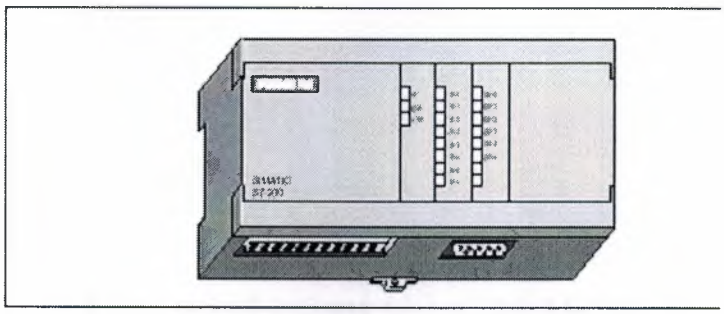


Figure 3.3 S7-212 CPU Module

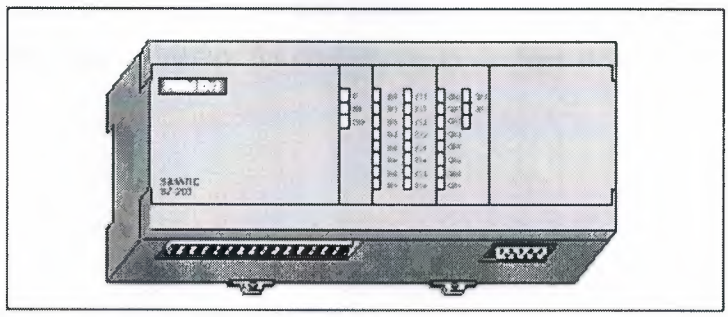


Figure 3.4 S7-214 CPU Module

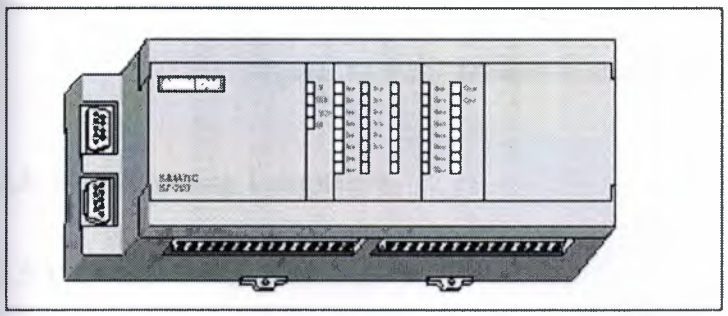


Figure 3.5 S7-215 and S7-216 CPU Module

3.4.2 Expansion Modules

The 7-200 CPU module provides a certain number of local I/O. Adding an expansion module provides additional input or output points. As shown in Figure 3.6, the expansion module comes with a bus connector for connecting to the base unit.

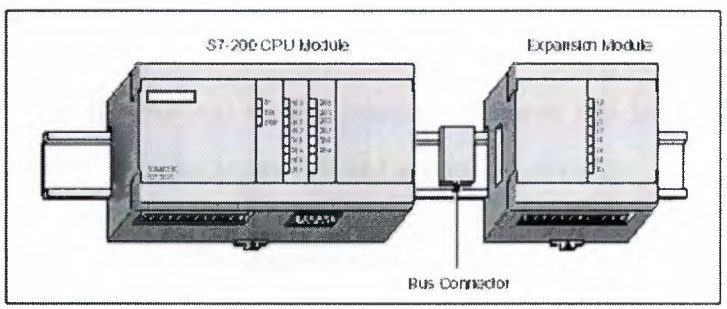


Figure 3.6 CPU Module with an Expansion Module

3.5 Programming Languages

3.5.1 Ladder Programs

In Ladder programs, the basic elements of logic are represented with contacts, coils, and boxes. A set of interconnected elements that make a complete circuit is called a network.

A hard-wired input is represented by a symbol called a contact. A normally open contact enables power flow when closed. A contact can also be normally closed. In this case, power flow occurs when the contact is opened.

A hard-wired output is represented by a symbol called a coil. When a coil has power flow, the output is turned on.

A box is a symbol for a complex operation performed within the CPU. The box simplifies programming of the operation. For example, boxes represent timers, counters, and math operations.

3.5.2 STL Programs

STL program elements are represented by a set of instructions for performing the desired functions. Instead of using the graphic display as shown by ladder programs, the STL program is shown in text format.

3.6 CPU Memory

The user memory in the S7-200 CPUs consists of three blocks: program, data, and configurable parameters. The blocks are defined according to usage:

1 Program memory stores the user program.

2 Data memory includes a temporary area for the program and storage of data. The temporary storage, calculations, and constants reside in data memory. Additionally, data for timers, counters, high-speed counters, and analog inputs and outputs are stored in data memory.

3 Configurable Parameter memory stores either the default or the modified parameters of the program setup. The configurable parameters include items such as protection level, password, station address, and retentive range information.

3.7 Simatic S7-200 Application Areas

The SIMATIC S7-200 series is a line of micro-programmable logic controllers (Micro PLCs) that can control a variety of automation applications. Compact design, low cost, and a powerful instruction set make the S7-200 controllers a perfect solution for controlling small applications. The wide variety of CPU sizes and voltages, and the windows-based programming tool, give you the flexibility you need to solve your automation problems.

3.7.1 The S7-200 characteristics

1 Easy entry
2 Uncomplicated operation
3 Peerless real-time characteristics
4 Powerful communications capabilities

The S7-200 also covers areas where previously special electronics have been developed for cost reasons. Application areas include:

1 Baling processes
2 Plaster & Cement mixers
3 Suction Plants
4 Centralized lubricating systems/flange lubricating systems
5 Woodworking machinery
6 Gate controls

7 Hydraulic lifts
8 Conveyor systems
9 Food & Drink Industry
10 Laboratories
11 Modem applications via dial-up, leased-line, or radio remote monitoring (SCADA)
12 Electrical Installations

3.7.2 Mechanical features

1 Rugged, compact plastic housing using SIMATIC's prize-winning design
2 Easily accessible wiring and operator control and display elements protected by front covers
3 Installs on standard horizontal or vertical DIN rail or direct cabinet mounting with built-in mounting
4 Terminal block as permanent wiring assembly (optional)

3.7.3 Design features

1 International standards; Meets the requirements through compliance with VDE, UL, CSA and FM standards.
2 The quality management system used during manufacturing has ISO 9001 certification; and Data back up; the user program and the most important parameter settings are stored in the internal EEPROM. A heavy-duty capacitor provides additional back up for all data over longer periods (typically up to 50 or 190 hours). An optional battery module ensures that the data remain stored for 200 days (typically) after power failure.

3.7.4 Benefits of the S7-200

The SIMATIC S7-200 Micro PLC is a full-featured programmable logic control system offering stand-alone CPUs, micro-modular expansion capability, and operator interface solutions. Almost any application that requires automation, from basic discrete or analog

control, to intelligent networked solutions, can benefit by using the powerful S7-200 family of products.

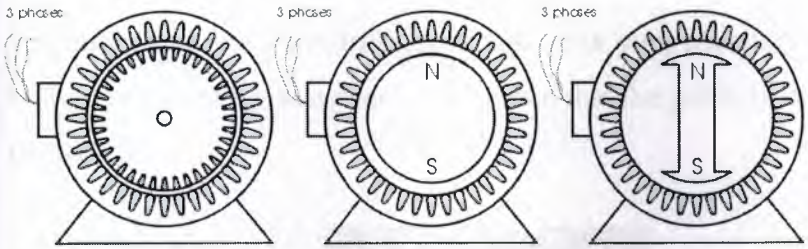
The SIMATIC S7-200 offers real-time control with Boolean processing speeds of 0.37μs per instruction. This fast execution speed, combined with our 20Khz high-speed counters, interrupts, and 20KHz pulse outputs, provide quick responses in demanding real-time applications. The S7-200 has over 200 instructions, including math, PID, For/Next loops, subroutines, sequence control, and more.

All S7-200 CPUs offer at least one RS485 communication port with speeds up to 187.5Kbaud. This not only provides fast access for programming and maintenance, but also allows you to build master/slave networks with up to 31 stations.

Using our Freeport capability can also connect non-S7-200 devices, such as bar code readers, intelligent machines, etc.. With Freeport, you can easily adapt the S7-200 CPU to virtually any serial ASCII protocol.

CHAPTER 4
EQUIPMENTS OF THE PROJECT

4.1 AC MOTORS



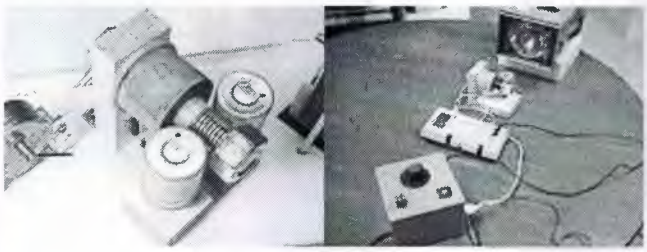
AC motors are also fairly simple to understand. They are a little trickier to make but will need single-phase or three-phase AC power to make them work. In the little diagrams above, we have a squirrel cage ac induction motor, a permanent magnet synchronous machine, and a synchronous motor. The inventor of the three-phase AC motor was Nikole tesla, a pioneer in electromagnetism.

There are a couple types of basic AC motors you can build. They also make super science fair projects.

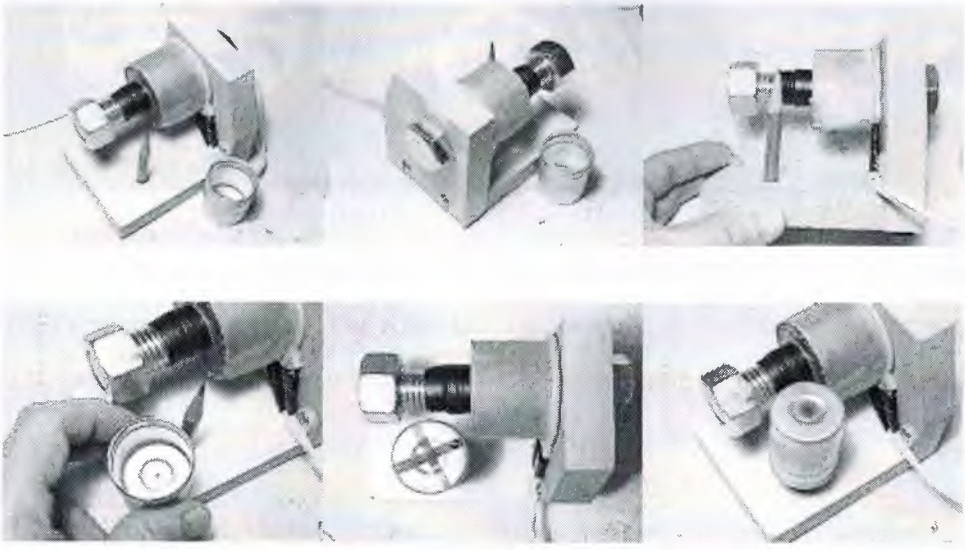
A Very Simple AC Motor

Here is a photo of a very simple eddy current AC motor I put together. I think this one wins the prize for the Simplest AC Motor you can make. It works great and is very easy to build. I found the original plans in a book titled: "Physics Demonstration Experiments" by Harry F. Meiners, Vol 2, Ronald Press Co., NY, 1970, LCCC #69-14674. With some experimentation, I found that the can spins faster when the nut is on the end of the bolt than when the nut is removed. What do you think will happen if the

rotor is moved to the other side of the bolt? It consists of a coil mounted onto a 3/4" bolt. The coil is about 100' of 20AWG wire, on a form about 1.5" long, with a dc resistance of about 1.2 ohms, and an inductance of about 2.4mH as an air-core inductor. The voltage supplied to the coil is 19Vac from a plug-in transformer and supplies about 2.5Aac to the coil. The rotor is an aluminum film canister (today they use plastic, but you might still find a few of these around - ask your friends) with a dimple in the bottom of it, resting on a pencil. (I figured that the graphite in the pencil will lubricate the rotor.)



The eddy current motor on the left has two rotors, they spin in opposite directions. The set-up on the right shows a variac, multimeter, eddy current motor, and a calibrated strobe. With this, we could plot speed vs. voltage. We found that the rotor would spin about 1000 rpm with 120V applied to it. Can't keep it there for long, since the coil and bolt get real hot. On these two coils, a smaller diameter wire was used, so the dc resistance was about 11.2 ohms, and 24mH as an air-core inductor. With this, we could apply 120Vac to it and only 2 amps would be drawn.

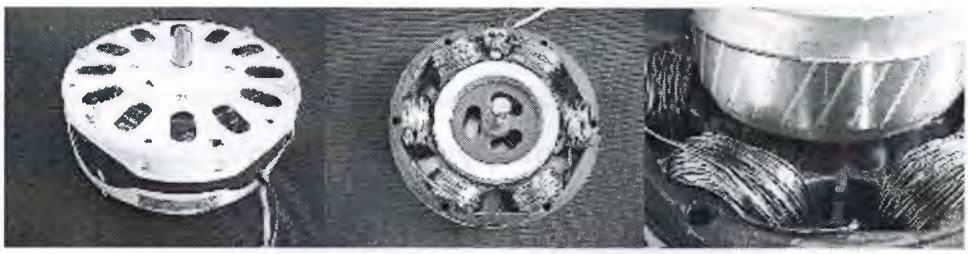


This shows the basic construction. The bolt is a 4" long 3/4-13 bolt, the wood is 3/4" thick. I put a small dimple into the bottom of the aluminum film canister so it would sit onto the pencil point. The red strips of tape helped with the strobe and looks cool as it spins. I found that the nut on the end of the bolt makes it go faster.

4.1.1 A Shaded Pole AC Motor

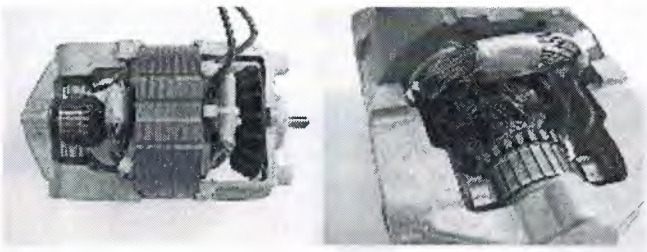
Here is a photo of a typical shaded pole motor. See the close-up of the notch in the laminations and the extra heavy winding of two turns creating the phase difference between the two sections of the laminations, giving the magnetic field a directional motion. The rotor spins CW as seen from the end with the screw on the shaft. Motors like this are used in thousands of applications.

Another Shaded Pole AC Motor



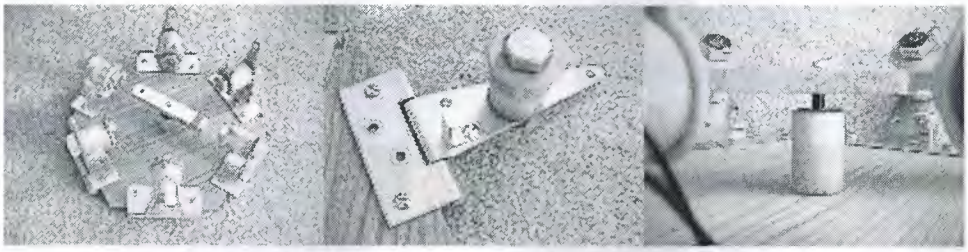
Here is a photo of a ceiling fan motor, also shaded pole, but with six windings instead of only one as seen above. The rotor laminations are skewed to provide smoother torque. The pole pieces with the windings have a slot in them to create a delayed flux, creating a direction for rotation.

4.2 A Universal Motor



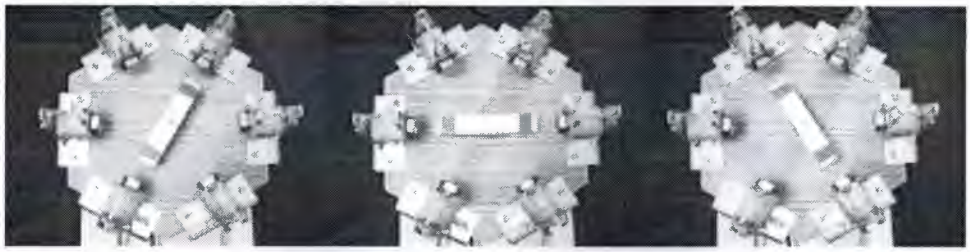
And here is a photo of a universal motor. It has brushes like a DC motor, but will operate on AC or DC.

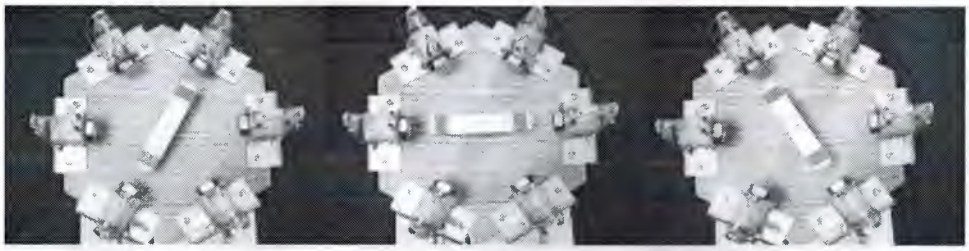
4.3 A 3-Phase AC Motor Demonstrator



Here is a project my daughter is working on. It shows how a 3-phase AC motor works with a rotating magnetic field and a permanent magnet rotor, making it a synchronous AC motor. We have pushbuttons which allow the user to turn on any one of the pairs of opposite coils, in either a N-S or a S-N orientation. For example, the green button turns on the horizontal pair of coils in a S-N orientation. The yellow button turns on the horizontal pair of coils in a N-S orientation. On each coil is a bi-color LED to indicate the magnetic polarity of the coil when it is turned on. The power to the coils (each pair connected in parallel) is supplied by a 5v computer power supply. The coils draw about 4amps at 5Vdc each, so a supply with 23amps available is a great match. Each coil is mounted on a 3/4" bolt, attached to a hinge. This way, sets of coils can be folded down out of the way to show how a shaded pole motor works. The rotor is a bar of steel with a NIB magnet on each end. The rotor does oscillate a bit when going from coil to coil.

Here's more photos:

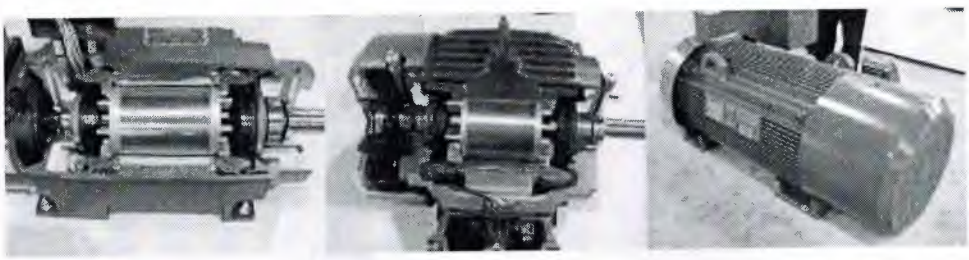




By pressing the colored buttons in the correct sequence, the rotor will follow the magnetic field in a clockwise fashion. The faster you go through the sequence, the faster the rotor will rotate. This shows that the speed of this motor is dependant on the frequency of the power applied to it. The higher the frequency, the faster it goes. At 60Hz, it would rotate at 1 revolution/cycle * 60 cycles/sec * 60 sec/min = 3600 revolutions per minute or rpm.

4.3.1 Three Phase AC Motor Stator

Industrial AC Motors



These are cut-aways of actual industrial three phase AC motors. They have different HP ratings, from 5hp, 2hp, 900hp. They are manufactured by Reliance Electric, part of Rockwell Automation.

4.4 Linear motors

A linear motor is like an ac motor, but it is unwrapped and laid out flat. The photos show parts of linear motors. Some have flat coils and magnet sections, others are "T" shaped.

4.5 Contactor

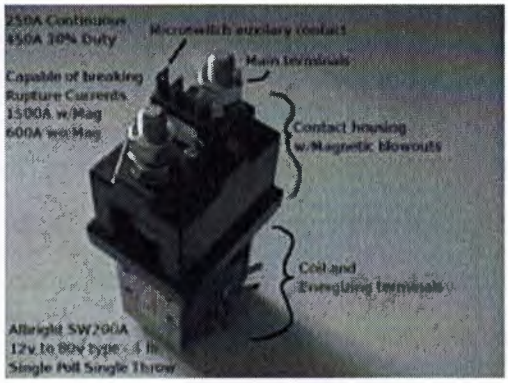
A contactor previously fitted in an elevator control system

A contactor is an electrical device used for switching a power circuit. A contactor is activated by a control input which is a lower voltage / current than that which the contactor is switching. Contactors come in many forms with varying capacities and features. Unlike a circuit breaker a contactor is not intended to interrupt a short circuit current.

Contactors range from having a breaking current of several amps and 110 volts to thousands of amps and many kilovolts. The physical size of a contactor ranges from those which are as large as a small car and those which are small enough to fit inside electrical equipment.

Contactors are used to control electric motors, lighting, heating, capacitor banks, and other electrical loads.

Construction



A contactor is composed of three different systems. The contact system is the current carrying part of the contactor. This includes Power contacts, Auxiliary contacts and contact springs. An electromagnet system provides the driving force to close the contacts. The enclosure system, is a frame housing the contact and the electromagnet. Enclosures are made of insulating materials like Bakelite, Nylon 6, and thermosetting plastics to protect and insulate the contacts and to provide some measure of , protection to personnel coming in contact. Open-frame contactors may have a further enclosure to protect against dust, oil, explosion hazards and weather.

Contactors used for starting electric motors are commonly fitted with overload protection to prevent damage to their loads. When an overload is detected the contactor is tripped removing power downstream from the contactor.

Some contactors are motor driven rather than relay driven and high voltage contactors (greater than 1000 volts) often have arc suppression systems fitted (such as a vacuum or an inert gas surrounding the contacts).

Magnetic Blowouts are sometimes used to increase the amount of current a contactor can successfully break. The field produced by the magnets in proximity to the contact forces the arc produced while breaking current to flow through the field which is curved and a greater distance than the straight path between the contacts. The magnetic blowouts in the pictured Albright contactor more than double the current it can break from 600 Amps to 1500 Amps.

Sometimes an Economizer circuit is also installed to reduce the power required to keep a contactor closed. A somewhat greater amount of power is required to initially close a contactor than is required to keep it closed thereafter. Such a circuit can save a substantial amount of power and allow the energized coil to stay cooler. Economizer

circuits are nearly always applied on direct-current contactor coils and on large alternating current contactor coils.

Contactors are often used to provide central control of large lighting installations, such as an office building or retail building. To reduce power consumption in the contactor coils, latching contactors are used, with two coils. One coil, momentarily energized, closes the power circuit contacts; the second opens the contacts.

A basic contactor will have a coil input (which may be driven by either an AC or DC supply depending on the contactor design) and generally a minimum of two poles which are controlled.

4.5.1 Operating Principle

A contactor is a type of electrical relay. Unlike general-purpose relays, contactors are designed to be directly connected to high-current load devices, not other control devices. Relays tend to be of much lower capacity and are usually designed for both Normally Closed and Normally Open applications. Devices switching more than 15 amperes or in circuits rated more than a few kilowatts are usually called contactors. Apart from optional auxiliary low current contacts, a contactor normally only has Normally Open contacts fitted.

When current passes through the electromagnet, a magnetic field is produced which attracts ferrous objects, in this case the moving core of the contactor is attracted to the stationary core. Since there is an air gap initially, the electromagnet coil draws more current initially until the cores meet and reduct the gap, increasing the inductive impedance of the circuit.

For contactors energized with alternating current, a small part of the core is surrounded with a shading coil, which slightly delays the magnetic flux in the core. The effect is to average out the alternating pull of the magnetic field and so prevent the core from buzzing at twice line frequency.

Most motor control contactors at low voltages (600 volts and less) are "air break" contactors, since ordinary air surrounds the contacts and extinguishes the arc when

interrupting the circuit. Modern medium-voltage motor controllers use vacuum contactors.

Motor control contactors can be fitted with short-circuit protection (fuses or circuit breakers), disconnecting means, overload relays and an enclosure to make a combination starter. In large industrial plants many contactors may be assembled in motor control centers.

4.5.2 Ratings

Contactors are rated by designed load current, maximum fault withstand current, duty cycle, voltage, and coil voltage. A general purpose motor control contactor may be suitable for heavy starting duty on large motors; so-called "definite purpose" contactors are carefully adapted to such applications as air-conditioning compressor motor starting. North American and European ratings for contactors follow different philosophies, with North American contactors generally emphasizing simplicity of application while European rating philosophy emphasizes design for the intended life cycle of the application.

Chapter 5

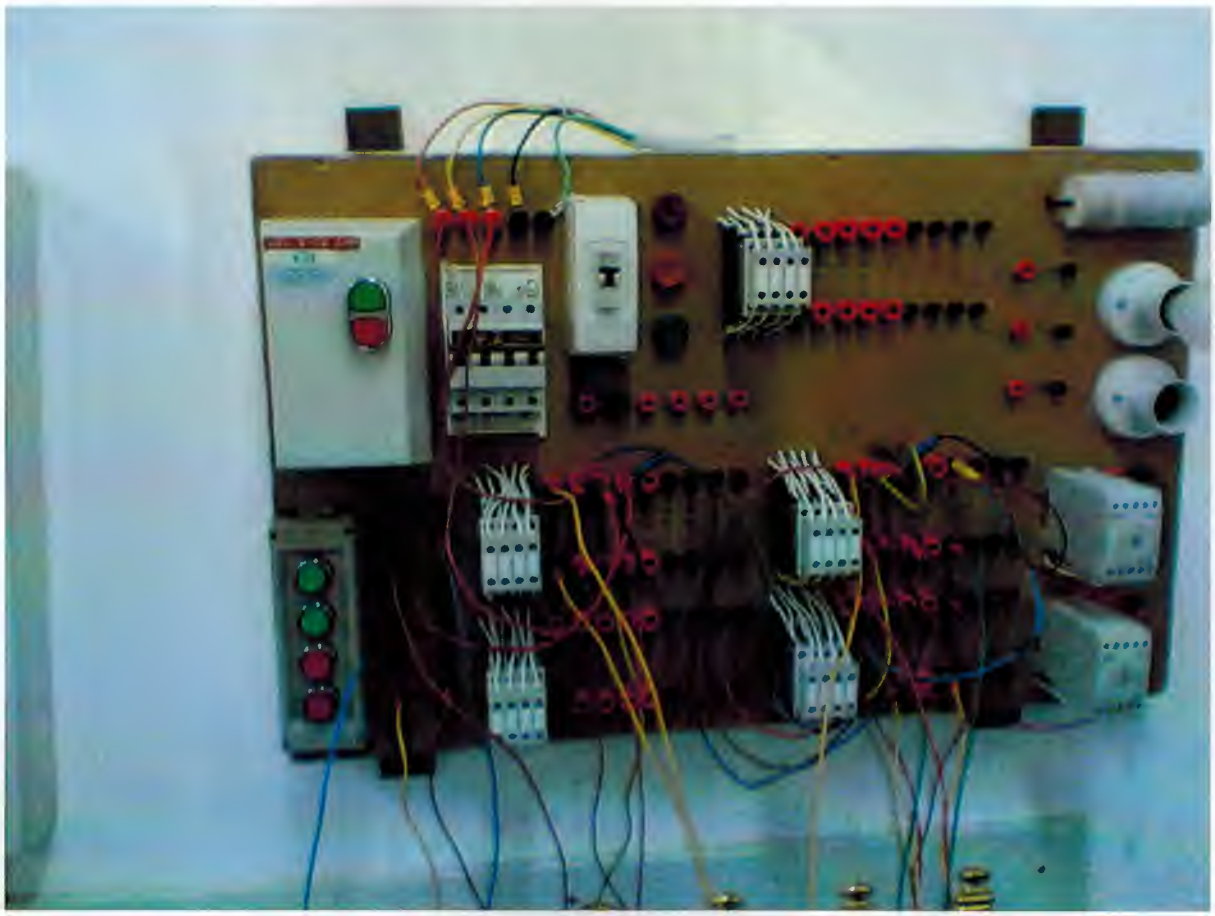
DELTA WYE START OF THE MOTOR WITH PLC

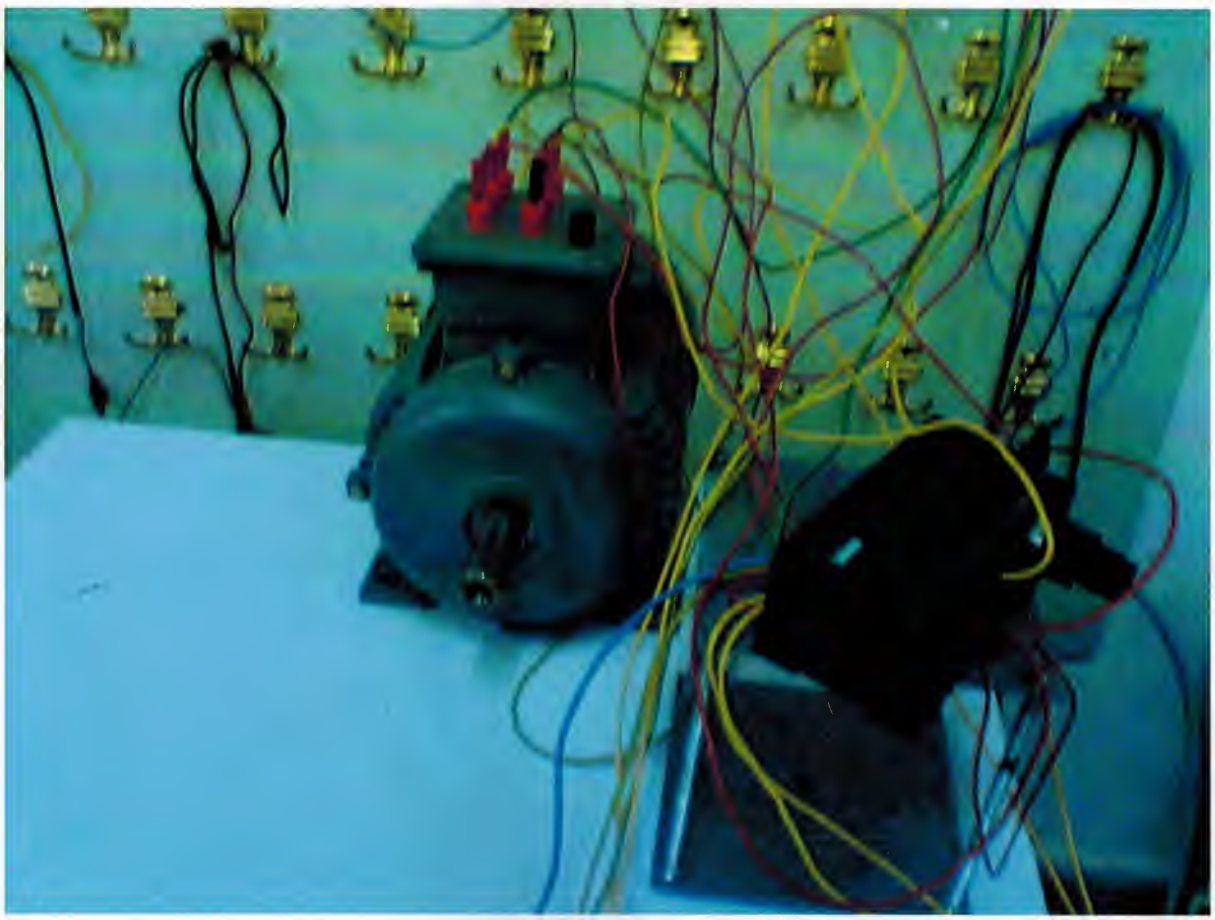
Motors which they are working more than normal conditions while they are taking the circuit wye and delta connections are prevalent uses. A motor which it has 380 volt,if we want to connect in wye and delta connection firstly we should connect the motor in wye connection than we connect it in delta connection. So while the motor is working apply 220 volt by this instantenously motor activates by the less current.

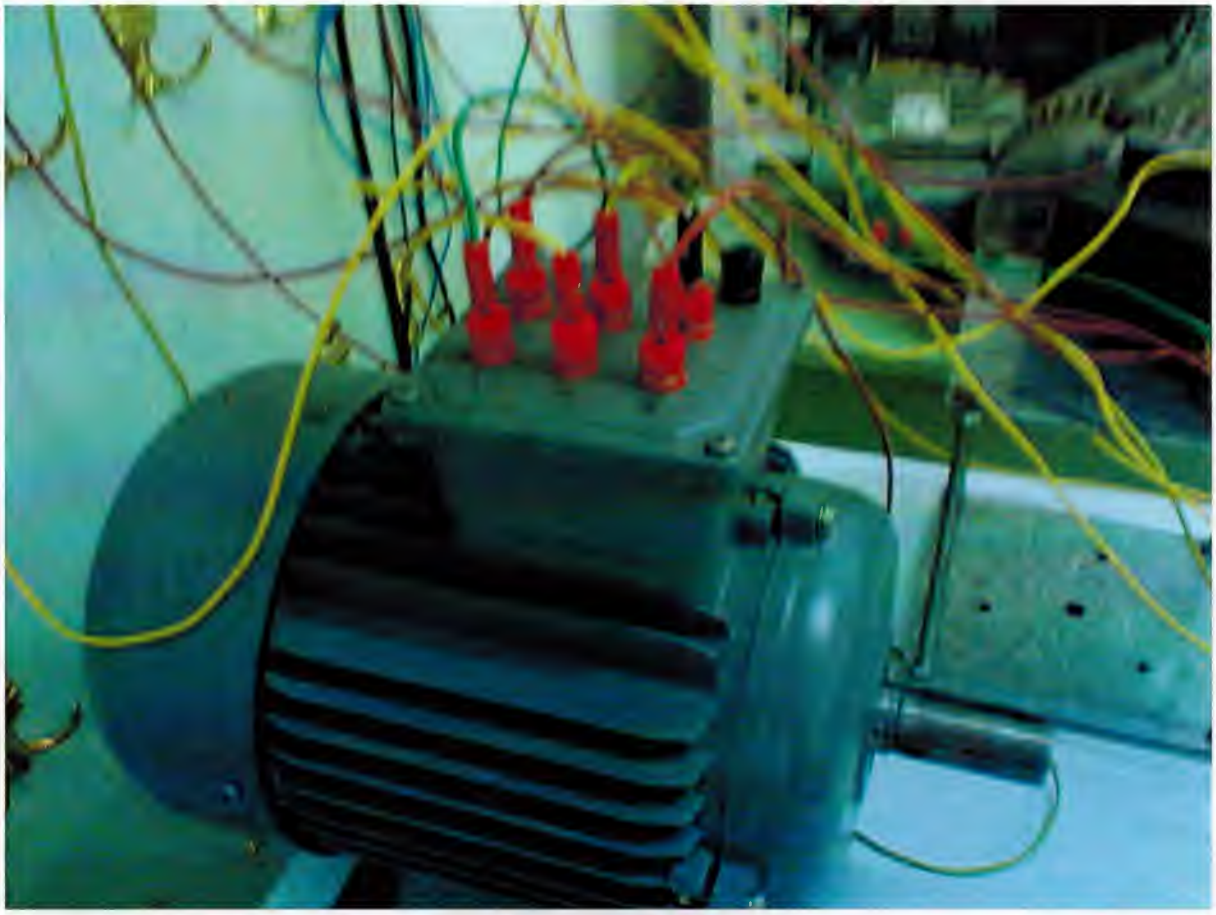
At wye-delta connection when pressed to start button firstly wye connection activates after a few seconds delta connection activates. While we are construct power circuit we choose three contactors these are main conductor wye condactor and delta condactor. R S T phases are connect to the input input of main contactor's than main contactor's outputs are connects to motor's wye input. Motor's delta outputs are connects to delta contactor. Delta contactor's outputs are connects to R S T phases and connects to wye contactor and wye contactor's output become short circuit.

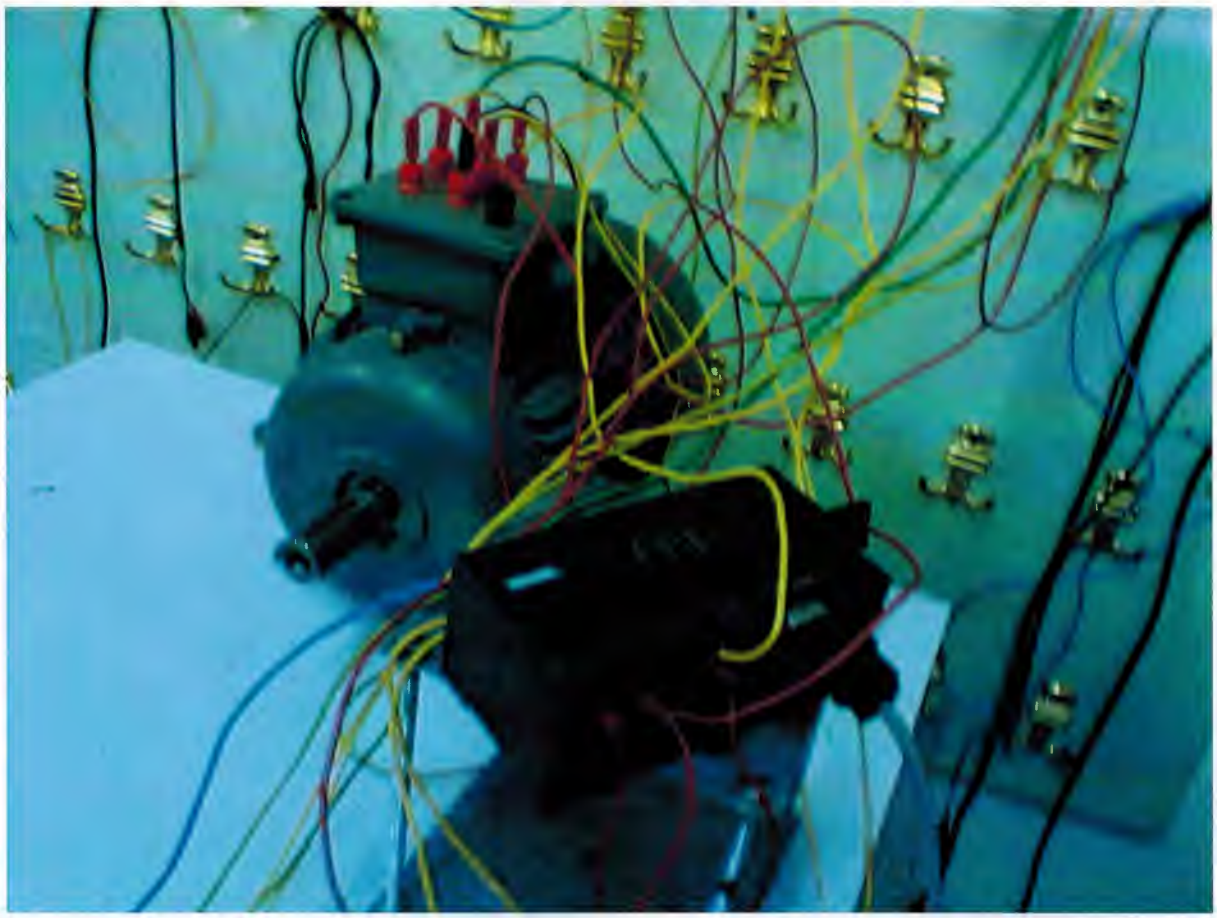
According to this power circuit we can write a PLC program. In any time when pressed the stop button motor stop. At a PLC circuit wye-delta shouldn't be on the circuit at the same time. So we connect front of wye contactors normally closed contact of delta contactor. Also for the same conditon we should connect front of delta contactors normaly closed contact of wye contactor in series.

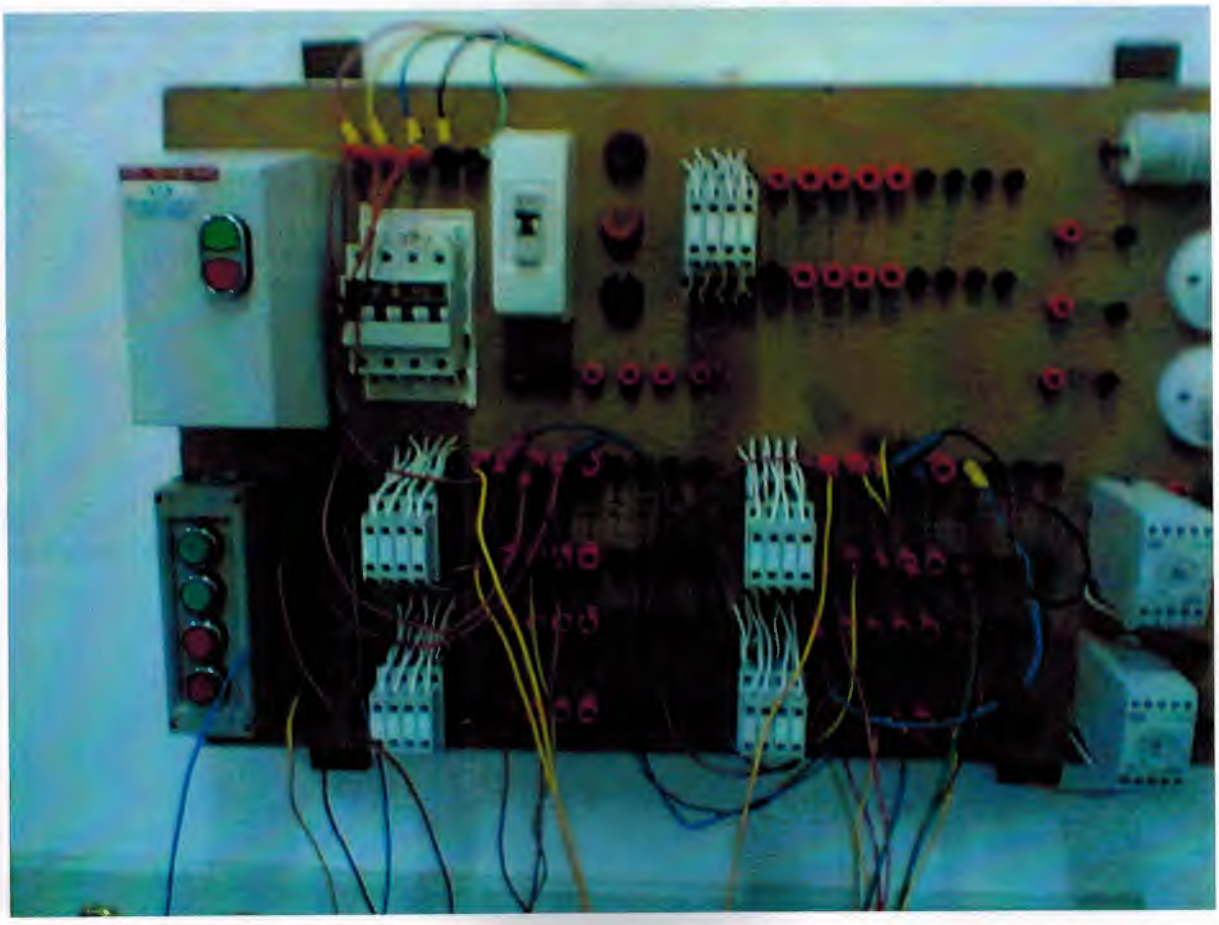
Stop button connects to PLC's normaly closed input also start button connects to PLC's normally open contact.

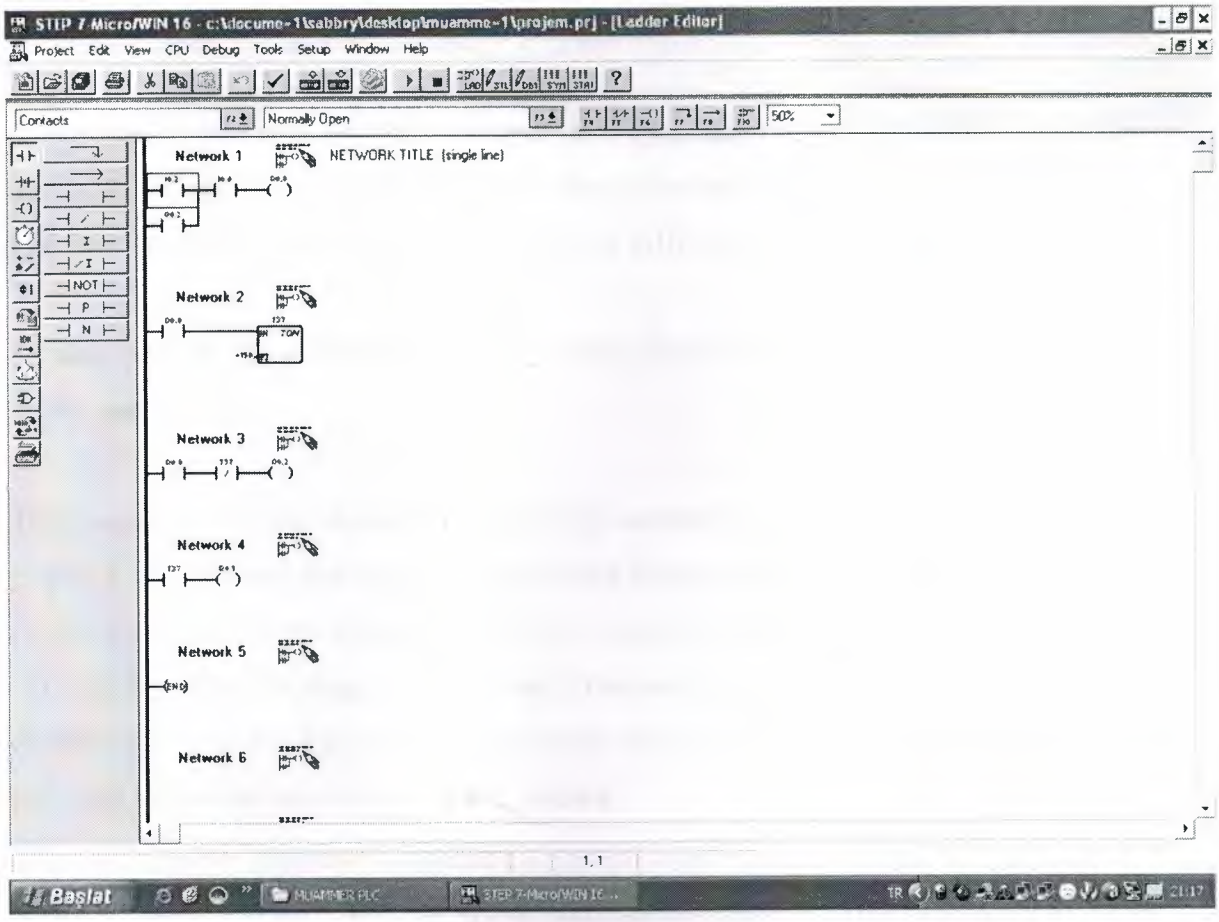


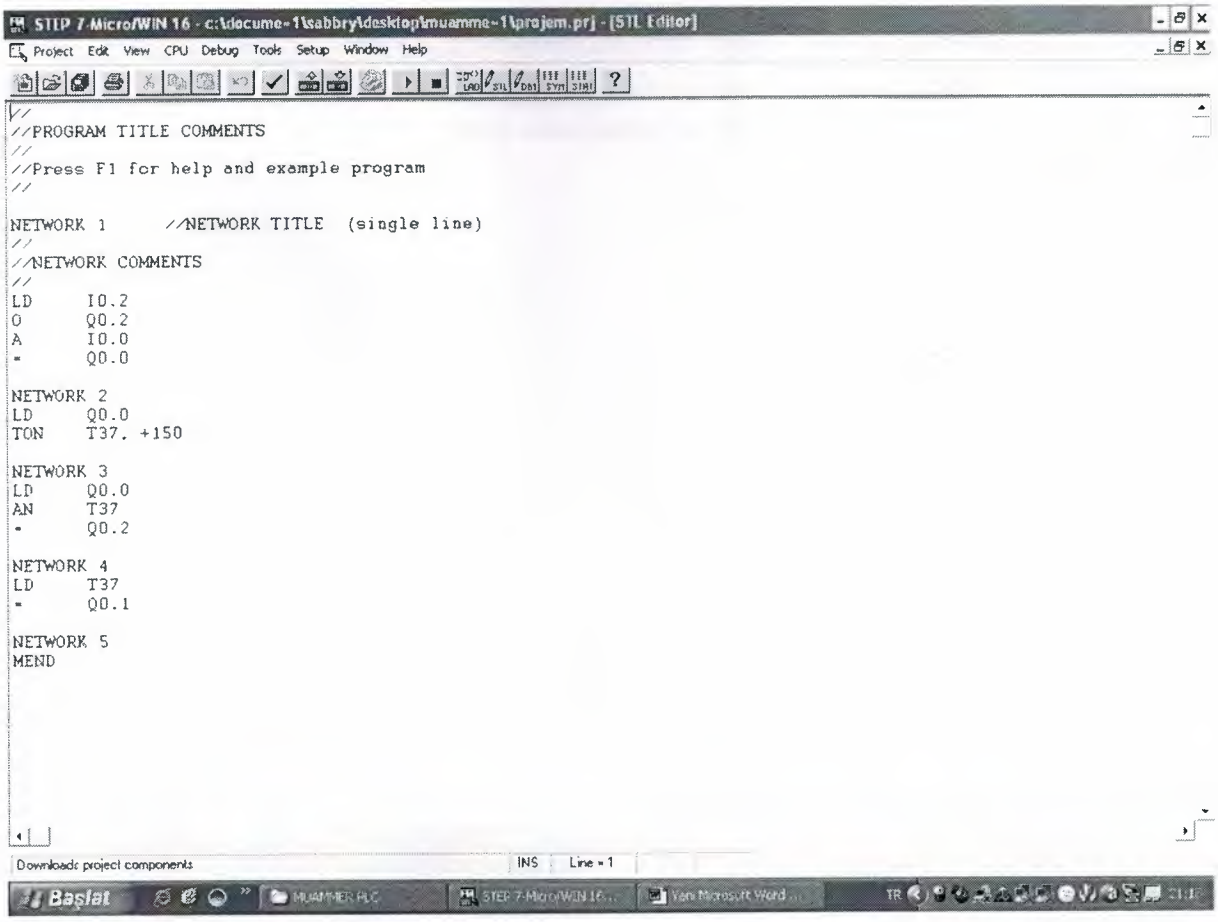









STEP 7-Micro/WIN 16 - c:\docume~1\sabbry\desktop\muamme~1\projem.prj - [Ladder Editor]
Project Edit View CPU Debug Tools Setup Window Help
Contacts
Normally Open
50%
Network 1
NETWORK TITLE (single line)
Network 2
Network 3
Network 4
Network 5
Network 6
1, 1
Başlat
MUAMMER PLC
STEP 7-Micro/WIN 16 ...


STEP 7-Micro/WIN 16 - c:\docume~1\sabbry\desktop\muamme~1\projem.prj - [STL Editor]
Project Edit View CPU Debug Tools Setup Window Help
//
//PROGRAM TITLE COMMENTS
//
//Press F1 for help and example program
//
NETWORK 1 //NETWORK TITLE (single line)
//
//NETWORK COMMENTS
//
LD I0.2
O Q0.2
A I0.0
= Q0.0
NETWORK 2
LD Q0.0
TON T37, +150
NETWORK 3
LD Q0.0
AN T37
= Q0.2
NETWORK 4
LD T37
= Q0.1
NETWORK 5
MEND
Downloads project components
INS
Line : 1
Başlat
MUAMMER PLC
STEP 7-Micro/WIN 16 ...
Yeni Microsoft Word ...

CONCLUSION

In this project Siemens S7- 200 PLC was used , for controllig of delta wye start of the motor. It has many advantageous in the industrial automation and building automation. The Siemens S7- 200 PLC can be used easily at different application areas.

In this project these main devices were used; Siemens S7- 200 PLC, contactor, AC motor, and cables

This project main topics is programmable logic controls'practical usage. İn this project,it was planned that wye-delta connected motor was controlled.Different kinds of control procedures were discussed and PLC controlled was decided.

At first, PLC's ladder diagram was made. After that input and output of the PLC was determined. After this STL diagram was made and by activating the circuit motors were provided to operate accordign to the procedure.

There are more complex real life applications systems similar. This model can be used as an experimental set for PLC education and some future applications

The system can be improved by adding some more control equipment.

REFERENCES

[1] SIMATIC S7-200 Programmable Controller, System Manual

[2] Özgür Cemal Özerdem, Programmable Logic Controller and Programming, Near East University Press, Lefkoşa 2002.

[3] http://www.sea.siemens.com/controls/product/s7200/CNs7200.htm

[4] http:// www.plcs.com

[5] E. C., Rusch R. J., "Electric Circuits And Machines", McGraw Hill, Seventh Edition, 1993,