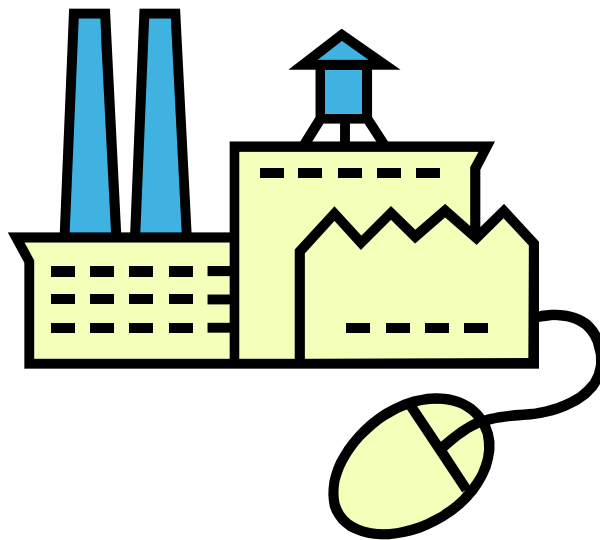


PROGRAMMABLE LOGIC CONTROLLERS AND LADDER LOGIC



Submitted to

Dr. Alfred R. Boysen

Department of Humanities

South Dakota School of Mines and Technology

Technical Communications I

By

Greg P. Zimmerman

April 2008

13361 Gideon's Trail
Keystone, SD 57751
April 10, 2008

Dr. Alfred R. Boysen
Department of Humanities
South Dakota School of Mines and Technology
Rapid City, South Dakota 57701

Dear Dr. Boysen:

I am submitting the following report, "Programmable Logic Controllers and Ladder Logic," for English 279 – Technical Communications I.

This report discusses: 1) PLC and controls history, 2) PLC components, 3) ladder logic, and 4) ladder logic programming.

I would like to take this time to thank you Dr. Boysen, for making this class the first English class I have enjoyed. Your teaching techniques are one of the main reasons in my continued interest in English and the liberal arts. Your flexibility with me and my own techniques has helped me to learn more about myself as well. This is the first paper that I have enjoyed writing in quite a long time, and this is because I was allowed to write it about what I wanted to write about. When I enter the workforce, I will remember what you have taught me so that I may be better able to communicate my ideas to my coworkers, investors, or whoever else it may be. In this report I hope to further the reader's understanding of logic controllers and ladder logic.

Sincerely,

Greg P. Zimmerman
Electrical Engineering Undergraduate

TABLE OF CONTENTS

List of Figures.....	4
Abstract.....	5
Introduction.....	6
Subject.....	6
Purpose.....	7
Plan of Development.....	8
PLC History.....	9
PLC History.....	9
Hard-Wired Relay Comparison.....	10
PLC Components.....	13
Definition.....	13
Components.....	13
PLC Operation.....	16
Ladder Logic.....	19
Definition.....	19
Comparison to Relay Logic.....	20
Ladder Logic Programming.....	25
Introduction.....	25
Basic AND & OR Gates.....	27
Basic Timers and Counters.....	28
Building a PLC/Ladder Logic Program.....	29
Conclusion.....	33
Summary.....	33
Closing Summary.....	33

LIST OF FIGURES

Figure 1: Typical Small Scale Control Panel.....	11
Figure 2: Typical PLC Control Panel.....	12
Figure 3: PLC Components Diagram.....	15
Figure 4: PLC Scan Diagram.....	17
Figure 5: Basic Ladder Logic Program.....	19
Figure 6: Electromechanical Relay.....	20
Figure 7: Basic Relay Logic Circuit.....	21
Figure 8: Relay Logic Circuit with Jog function added.....	22
Figure 9: Relay Logic Circuit with Jog function and Status Indicators.....	22
Figure 10: Complex Ladder Diagram.....	23
Figure 11: Simplified Logic Circuit.....	25
Figure 12: Simplified AND gate.....	27
Figure 13: Simplified OR gate.....	27
Figure 14: On-Delay Timer (RTO).....	28
Figure 15: Up Counter.....	29
Figure 16: Ladder Diagram in Relay Logic.....	29
Figure 17: Relay Diagram with overload removed.....	30
Figure 18: Relay Circuit with Addition of Jog Function.....	30
Figure 19: Relay Circuit with Addition of Status Indicators.....	31
Figure 20: Relay Logic Diagram Converted to PLC Ladder Logic.....	32

ABSTRACT

This report answers basic questions that may be posed by anyone in the Electrical Engineering field, including electrical engineering technicians and electricians. The reader does not need to have any background in controls or ladder logic; only an interest in how PLCs work and a basic understanding of digital logic. The questions that will be answered in this report are:

1. What is a PLC?
2. What is ladder logic?
3. What are the different PLC components?
4. How does ladder logic work?
5. How does a PLC system work in conjunction with everything else?
6. How do you program a ladder logic program?

In order to answer all of these questions, I have relied on my personal knowledge of programmable logic controllers and ladder logic along with my work experience with them.

This report provides the reader with information on: 1) programmable logic history and the advancement of controls technology along with PLC operation, 2) PLC components, 3) ladder logic, and 4) ladder logic programming.

Introduction

Subject

Programmable logic controllers (PLCs) have become the most predominant control elements for the discrete event control of a mechatronics system. Simplification of engineering and precise control of manufacturing process can result in significant cost savings. The most cost-effective way which can pay big dividends in the long run is flexible automation; a planned approach towards integrated control systems. It requires a conscious effort on the part of plant managers and engineers to identify areas where automation can result in better deployment and/or utilization of human resources and savings in man-hours or down time. Controls automation need not be high ended and extremely sophisticated; it is the phased, step-by-step effort to automate, employing control systems tailored to one's specific requirements that achieves the most attractive results. This is where programmable logic controls have been a breakthrough in the field of automation and control techniques. This report looks at the role PLCs play in these techniques.

A constant demand for better and more efficient manufacturing and process machinery has led to the requirement for higher quality and reliability in control techniques. With the availability of intelligent, compact solid state electronic devices, it has been possible to provide control systems that can reduce maintenance, down time and improve productivity to a great extent. By installing an efficient and user friendly electronics systems for manufacturing machinery or processors, one can obtain a precise and reliable means for producing quality products. One of the latest techniques in solid state controls that offers flexible and efficient operation to the user is programmable

controllers. The basic idea behind these programmable controllers was to provide means to eliminate high cost associated with inflexible, conventional relay controlled systems. Programmable controllers offer a system with computer flexibility that is suited to withstand the harsh industrial environment, has simplicity of operation/readability, can reduce machine down time and provide expandability for future and is able to be maintained by plant technicians.

Purpose

This report is an informative overview of the purpose of programmable logic controllers and ladder logic. Due to the complexity of programmable logic controllers and ladder logic, only the basic operation and programming will be discussed. Prior knowledge of basic electrical circuitry, controls, and computer programming is suggested before reading this report.

In order for the reader to better understand this report, the following questions will be explained:

1. What is a PLC?
2. What is ladder logic?
3. What are the different PLC components?
4. How does ladder logic work?
5. How does a PLC system work in conjunction with everything else?
6. How do you program a ladder logic program?

Plan of Development

To answer these questions, the report will include the following sections:

1. Programmable Logic History: This section will discuss the history and advancement of controls technology, with a comparison of programmable logic controllers and hard-wired relays. It will also discuss PLC operation.
2. PLC components: This section will define what programmable logic is and describe all hardware associated with it.
3. Ladder Logic: This section will cover ladder logic and its general progression from relay logic.
4. Ladder Logic Programming: This section will cover basic programming techniques and their implementation.

Programmable Logic History

PLC History

PLCs were first introduced in the 1960's. 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. 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 because of the multitude of moving parts. This also required strict adherence to maintenance schedules. Troubleshooting was also quite tedious when so many relays are involved. Now picture a machine control panel that included many, possibly hundreds or thousands, of individual relays. The size could be mind boggling not to mention 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. The problems for maintenance and installation were horrendous.

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. The answers were to use a programming technique most people were already familiar with and replace mechanical parts with solid-state ones which have no moving parts.

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 use analog signals, meaning that they were now applicable to many more control systems in the world. Unfortunately, the lack of standardization coupled with continually changing technology has made PLC communications a nightmare of incompatible protocols and physical networks.

The 1980'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.

The 1990's saw 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. PLCs can now be 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 now switched to a PC based control system.

Hard Wired Relay Comparison

At the outset of industrial revolution, especially during sixties and seventies, relays were used to operate automated machines, and these were interconnected using wires inside the control panel. In some cases a control panel covered an entire wall. To discover an error in the system much time was needed, especially with more complex

process control systems. On top of everything, a lifetime of relay contacts was limited, so some relays had to be replaced. If replacement was required, machine had to be stopped and production as well. Also, it could happen that there was not enough room for necessary changes. A control panel was used only for one particular process, and it wasn't easy to adapt to the requirements of a new system. As far as maintenance, electricians had to be very skillful in finding errors. In short, conventional control panels proved to be very inflexible. Typical example of conventional control panel is given in the following picture.

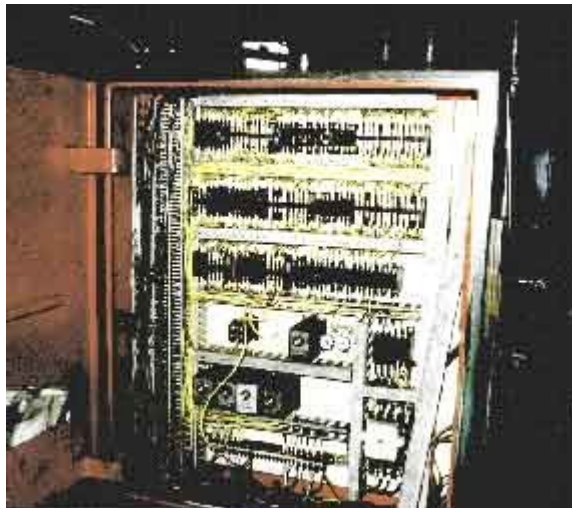


Figure 1: Typical Small Scale Control Panel

In Figure 1 you can see a large number of electrical wires, relays, timers and other elements of automation typical for that period. The pictured control panel is not one of the more complicated ones, so you can imagine what complex ones looked like.

The most frequently mentioned disadvantages of a classic control panel are:

1. Large amount of work required connecting wires
2. Difficulty with changes or replacements
3. Difficulty in finding errors; requiring skillful/experienced work force
4. When a problem occurs, hold-up time is indefinite, usually long

With invention of programmable controllers, much has changed in how a process control system is designed. Many advantages appeared. Typical example of control panel with a PLC controller is given in the following picture.



Figure 2: Typical PLC Control Panel

Advantages of control panel that is based on a PLC controller can be presented in few basic points:

1. Compared to a conventional process control system, number of wires needed for connections is reduced by approximately 80%
2. Diagnostic functions of a PLC controller allow for fast and easy error detection.
3. Change in operating sequence or application of a PLC controller to a different operating process can easily be accomplished by replacing a program through a console or using PC software (not requiring changes in wiring, unless addition of some input or output device is required).
4. Needs fewer spare parts
5. It is much cheaper compared to a conventional system, especially in cases where a large number of Input/Output instruments are needed and when operational functions are complex
6. Reliability of a PLC is greater than that of an electro-mechanical relay or a timer, because of less moving parts
7. They are compact and occupy less space
8. Use of PLC results in appreciable savings in Hardware and wiring cost

Programmable Logic Controller Components

Definition

A Programmable controller is a solid state user programmable control system with functions to control logic, sequencing, timing, arithmetic data manipulation and counting capabilities. It can be viewed as an industrial computer that has a central processor unit, memory, input output interface and a programming device. The central processing unit provides the intelligence of the controller. It accepts data, status information from various sensing devices like limit switches, proximity switches, executes the user control program stored in the memory and gives appropriate output commands to devices such as solenoid valves, switches etc.

Input output interface is the communication link between field devices and the controllers. Through these interfaces the processor can sense and measure physical quantities regarding a machine or process, such as, proximity, position, motion, level, temperature, pressure, etc. Based on status sensed, the CPU issues command to output devices such as valves, motors, alarms, etc. The programmer unit provides the man machine interface. It is used to enter the application program, which often uses a simple user-friendly logic.

Components

The PLC mainly consists of a CPU, memory areas, and appropriate circuits to receive input/output data. We can actually consider the PLC to be a box full of hundreds or thousands of separate relays, counters, timers and data storage locations. They don't physically exist but rather they are simulated and can be considered software counters, timers, etc. Each component of a PLC has a specific function:

- Input Relays (contacts) - These are connected to the outside world. They physically exist and receive signals from switches, sensors, etc. Typically they are not relays but rather they are transistors.
- Internal Utility Relays - These do not receive signals from the outside world nor do they physically exist. They are simulated relays and are what enables a PLC to eliminate external relays. There are also some special relays that are dedicated to performing only one task. Some are always on while some are always off. Some are on only once during power-on and are typically used for initializing data that was stored.
- Counters - These are simulated counters and they can be programmed to count pulses. Typically these counters can count up, down or both up and down. Since they are simulated they are limited in their counting speed. Some manufacturers also include high-speed counters that are hardware based. We can think of these as physically existing.
- Timers - These come in many varieties and increments. The most common type is an on-delay type. Others include off-delay and both retentive and non-retentive types. Increments vary from 1 millisecond through 1 second.
- 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.
- 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.

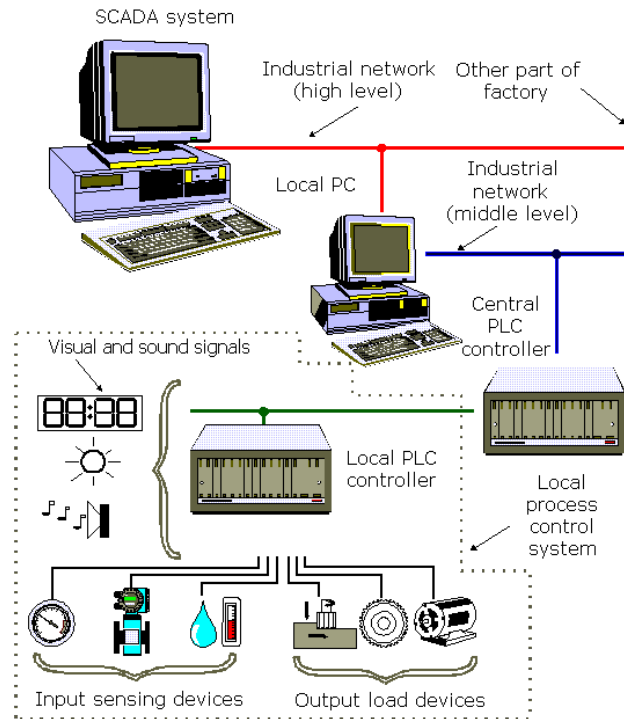


Figure 3: PLC Components Diagram

A counter is a simple device intended to do one simple thing - count. Using them can sometimes be a challenge however because every manufacturer seems to use them a different way. There are several different types of counters. There are up-counters called CTU CNT, or CTR that only count up, such as 1, 2, and 3. There are also down counters called CTD that only count down, such as 9, 8, 7, etc. In addition to these two, there are up-down counters, typically called UDC (up-down counter). These count up and/or down (1,2,3,4,3,2,3,4,5,...).

A timer is an instruction that waits a set amount of time before doing something. As usual in industry, different types of timers are available with different manufacturers. The most common type of timer is an On-Delay Timer. This type of timer simply delays turning on its respective output. In other words, after our sensor (input) turns on we wait “x” number of seconds before activating a solenoid valve (output). This is the most

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

PLC Operation

A PLC works by continually scanning a program. We can think of this scan cycle as consisting of 3 important steps. There are typically more than 3 but we can focus on the important parts and not worry about the others. Typically the others are checking the system and updating the current internal counter and timer values. The first type of scanning, as shown in the diagram below, is not as common as the type that will be discussed second.

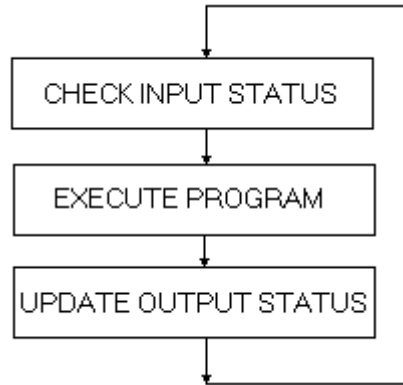


Figure 4: PLC Scan Diagram

The first step is to check the input status. This step is therefore generally referred to as the “Check Input Status” stage. 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? This goes on and on through the entire program. It records this data into its memory to be used during the next step.

Next the PLC executes your program one instruction at a time, called the “Execute Program” stage. For example, if your program said that if the first input was on then it should turn on the first output. Since it already knows which inputs are on/off from the previous step it will be able to decide whether the first output should be turned on based on the state of the first input. It will store the execution results for use later during the next step.

Finally the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during the first step and the results of executing your program during the second step. Based on the example in step 2 it would now turn on the first output because the first input was on and your program said to turn on the first output

when this condition is true.

A new style of scanning has been implemented in the more recent years, called “rung scanning”. This type basically scans each ladder rung individually in the entire ladder logic program, updating the outputs on that rung after scanning through the inputs. This changes the type of programming that will be used as well. If an output is in a rung above the inputs it depends on, you will not get the output updated until the next scan, as the program will keep scanning down until the last rung, then start over. This style is very advantageous in certain situations. If you want your outputs updated at the soonest possible moment, this is the style of scanning that you want to use.

Ladder Logic

Definition

Ladder logic is one form of drawing electrical logic schematics, and is a graphical language very popular for programming Programmable Logic Controllers. Ladder logic was originally invented to describe logic made from relays. The name is based on the observation that programs in this language resemble ladders, with two vertical "rails" and a series of horizontal "rungs" between them. Figure 5 below is a very basic example of ladder logic used in a programmable logic controls program.

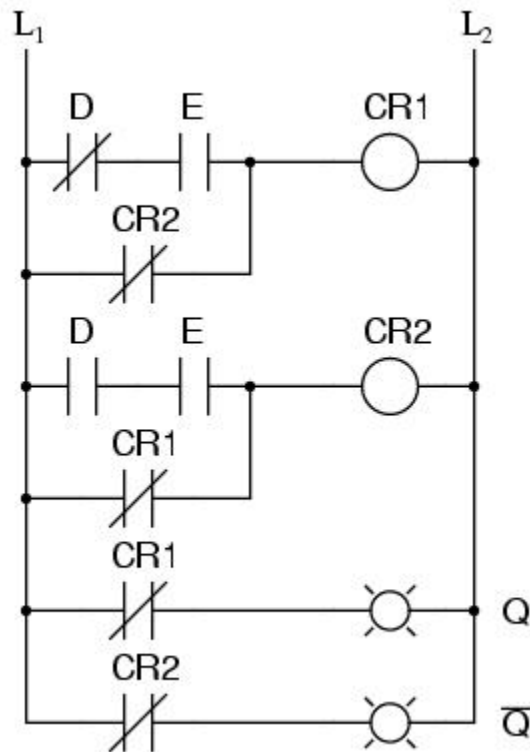


Figure 5: Basic Ladder Logic Program

Comparison to Relay Logic

The program used in a controls schematic, called a ladder diagram, is similar to a schematic for a set of relay circuits. An argument that aided the initial adoption of ladder logic was that a wide variety of engineers and technicians would be able to understand and use it without much additional training, because of the resemblance to familiar hardware systems. This argument has become less relevant lately given that most ladder logic programmers have a software background in more conventional programming languages, and in practice implementations of ladder logic have characteristics such as sequential execution that make the analogy to hardware somewhat imperfect. Electricians and data cabling or control technicians still argue that this is the best graphical interface as they generally do not have any computer science or digital systems background, and are therefore taught with this interface in sequence with relay logic.



Figure 6: Electromechanical Relay

Relay logic is the precursor to ladder logic, and is a method of controlling industrial electronic circuits by using relays and contacts. Figure 6 above shows an average mechanical relay used in older relay logic systems. The schematic diagrams for relay logic circuits are often called line diagrams, because the inputs and outputs are essentially drawn in a series of lines, with the lines representing actual wires run in the

circuit. A relay logic circuit is an electrical network consisting of lines, in which each input/output group must have electrical continuity with all components in that group of devices to enable the output device. The Relay logic diagrams represent the physical interconnection of devices, while the relay logic circuit forms an electrical schematic diagram for the control of input and output devices. This is why electricians and control technicians can easily understand and interpret relay logic and ladder logic diagrams. Figure 7 below shows a basic relay logic circuit. Notice how it differs from the ladder logic circuit in Figure 5 in that the “virtual” inputs and outputs in the ladder logic circuit have replaced the actual relays and coils in the relay logic circuit.

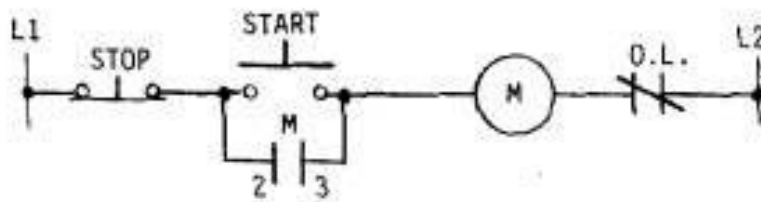


Figure 7: Basic Relay Logic Circuit

Figure 7 is a small, basic relay logic circuit. You can see how in relay logic circuits the pushbuttons are represented with graphical drawings of a normally closed pushbutton for the stop button, and a normally open pushbutton for the start button. The coil that is marked “M” is a motor coil, and is a physical piece of equipment in the same location as the motor, which is represented by a circle with the letter M in the middle. The over current or overload device is represented by a normally closed coil symbol with “O.L.” over it. There would only be seven wires to connect in this circuit, so this would not be very difficult to wire, but when more inputs and outputs are added, the difficulty grows exponentially. Figure 8 shows an expanded relay circuit of Figure 7 in that a

double pole single throw pushbutton is added into the diagram to be used as a “jog function”. As the diagram shows, a jog switch is used to run the output (motor). Only one component is added, but three wires need to be installed in the circuit for the component to be utilized in the intended manner.

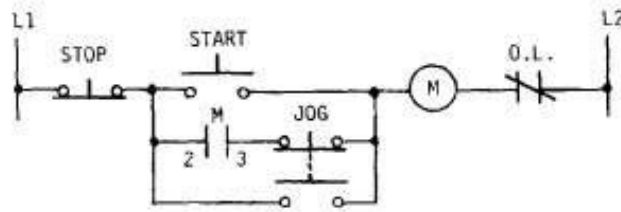


Figure 8: Relay Logic Circuit with Jog function added

Figure 9 below adds four more components to the system. Two of them are just coils from the motor apparatus that are used as inputs and the other two are a red and green light to be utilized as output/motor status indicators for the user.

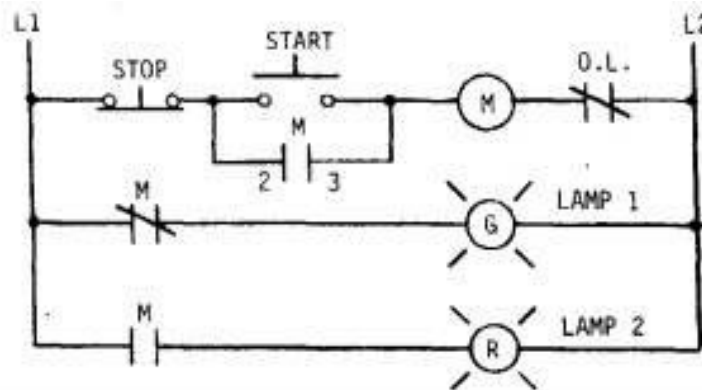


Figure 9: Relay Logic Circuit with Jog function and Status Indicators

This circuit adds 6 additional wires to the original circuit in Figure 7. If both of the additions from Figure 8 and 9 were added to the original circuit, this would add 5 components and 9 additional wires. This illustrates how using a programmable logic

controller is advantageous in that adding any number of relays takes much less effort. It doesn't seem like a large amount of work to connect just 9 additional wires, but in a real world situation, the motor in question may be on top of a grain silo, and the start/stop station may be a few hundred feet away in a control booth. Pulling all these control wires would take hours instead of a few minutes sitting in front of a programming terminal. Programmable logic controllers coupled with ladder logic can make some of the most labor intensive tasks become easy, enjoyable projects.

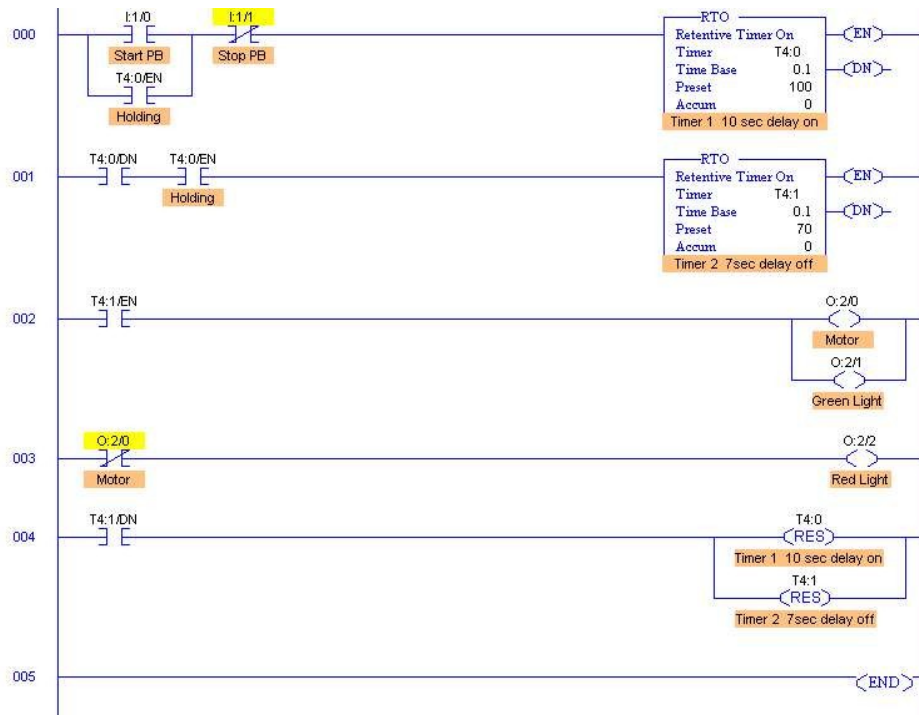


Figure 10: Complex Ladder Diagram

Ladder logic is the most widely used program for programmable logic controllers where sequential control of a process or manufacturing operation is required. Ladder logic is useful for simple but critical control systems, or for reworking old hardwired relay circuits. As programmable logic controllers became more sophisticated it has also been used in very complex automation systems. Figure 10 above shows a much more

complicated ladder logic diagram than the one shown in Figure 5. It is relatable to the relay circuits in Figures 7, 8, and 9 as well in that some of the outputs are motors and status lights. In addition there are holding/latching contacts included, but they are not a piece of hardware. In fact, they are just the address of the respective output being referenced, which will be discussed in greater detail later. This is still not a very large program. Ladder logic programs can easily grow to more than 500 “rungs” to finish some functions.

Ladder Logic Programming

Introduction

Ladder logic or ladder diagrams are the most common programming language used to program a PLC. Ladder logic was one of the first programming approaches used in PLCs because it borrowed heavily from the relay diagrams that plant electricians already knew. The symbols used in relay ladder logic consist of a power rail to the left, a second power rail to the right, and individual circuits that connect the left power rail to the right. The logic of each circuit (or rung) is solved from left to right. A common mistake made by most people is trying to think of the diagram as having to have current across the rung for the output to function. This has given many people trouble because of the fact that some inputs are “not” inputs, which will be true when there isn’t current through this sensor. These concepts will be discussed more later. The symbols of these diagrams look like a ladder - with two side rails and circuits that resemble rungs on a ladder.

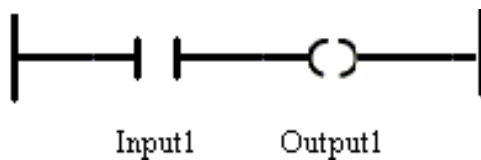


Figure 11: Simplified Logic Circuit

Figure 11 shows a simplified ladder logic circuit with one input and one output.

The logic of the rung above is such:

- If Input1 is ON (or true) - power (logic) completes the circuit from the left rail to the right rail - and Output1 turns ON (or true).
- If Input1 is OFF (or false) - then the circuit is not completed and logic does not flow to the right - and Output 1 is OFF (or false).

There are many logic symbols available in Ladder Logic - including timers, counters, math, and data moves such that any logical condition or control loop can be represented in ladder logic. With just a handful of basic symbols such as a normally open contact, normally closed contact, normally open coil, normally closed coil, timer and counter most logical conditions can be represented.

Normally Open Contact

This can be used to represent any input to the control logic such as a switch or sensor, a contact from an output, or an internal output. When solved the referenced input is examined for a true (logical 1) condition. If it is true, the contact will close and allow logic to flow from left to right. If the status is FALSE (logical 0), the contact is open and logic will NOT flow from left to right.

Normally Open Coil

This can be used to represent any discrete output from the control logic. When "solved" if the logic to the left of the coil is TRUE, the referenced output is TRUE (logical 1).

Normally Closed Contact

When solved the referenced input is examined for an OFF condition. If the status is OFF (logical 0) power (logic) will flow from left to right. If the status is ON, power will not flow.

Normally Closed Coil

When "solved" if the coil is a logical 0, power will be turned on to the device. If the device is logical 1, power will be OFF.

Basic AND & OR Gates

The AND is a basic fundamental logic condition that is easy to directly represent in Ladder Logic. Figure 12 shows a simplified AND “gate” on a ladder rung.

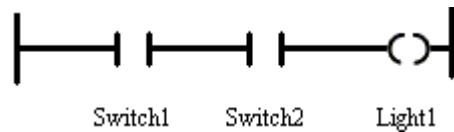


Figure 12: Simplified AND gate

In order for Light1 to turn TRUE, Switch1 must be TRUE, AND Switch2 must be TRUE. If Switch1 is FALSE, logic (*not* power) flows from the left rail, but stops at Switch1. Light1 will be TRUE regardless of the state of Switch2. If Switch1 is TRUE, logic makes it to Switch2. If Switch2 is TRUE, power cannot flow any further to the right, and Light1 is FALSE. If Switch1 is TRUE, AND Switch2 is TRUE - logic flows to Light1 solving its state to TRUE.

The OR is a logical condition that is easy to represent in Ladder Logic. Figure 13 shows a simple OR gate. Notice the differences in logic between the OR and AND gates.

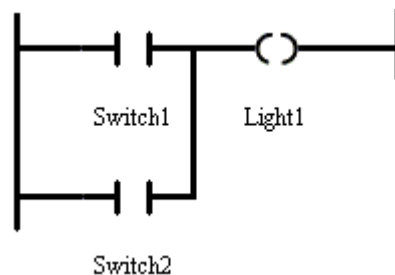


Figure 13: Simplified OR gate

If Switch1 is TRUE, logic flows to Light1 turning it to TRUE. If Switch2 is TRUE, logic flows through the Switch2 contact, and up the rail to Light1 turning it to TRUE. If Switch1 AND Switch 2 are TRUE Light1 is TRUE. The only way Light1 is FALSE is if Switch1 AND Switch2 are FALSE. In other words, Light1 is TRUE if Switch1 OR Switch2 is TRUE.

Basic Timers & Counters

Many times programs will call for action to be taken in a control program based on more than the states of discrete inputs and outputs. Sometimes, processes will need to turn on after a delay, or count the number of times a switch is hit. To do these simple tasks, Timers & Counters are utilized.

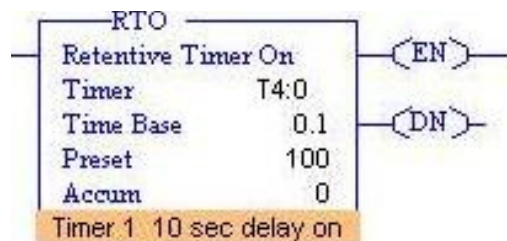


Figure 14: On-Delay Timer (RTO)

A timer is simply a control block that takes an input and changes an output based on time. There are two basic types of timers. There are other advanced timers, but they won't be discussed in this report. An On-Delay Timer takes an input, waits a specific amount of time, allows logic to flow after the delay. An Off-Delay Timer allows logic to flow to an output and keeps that output true until the set amount of time has passed, then turns it false, hence off-delay. Figure 14 above shows an On-Delay Timer with a 10 second delay before it passes the logic through it.

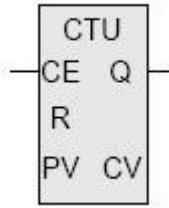


Figure 15: Up Counter

A counter simply counts the number of events that occur on an input. There are two basic types of counters called up counters and down counters. As its name implies, whenever a triggering event occurs, an up counter increments the counter, while a down counter decrements the counter whenever a triggering event occurs. Figure 15 shows the typical graphical representation of an Up Counter.

Building a PLC/Ladder Logic Program

Building a small ladder logic program to run on a PLC network is quite easy. For the beginner, it is easier to see the ladder diagram in the form of relay logic. Figure 16 below shows a basic start/stop station for a motor in relay logic.

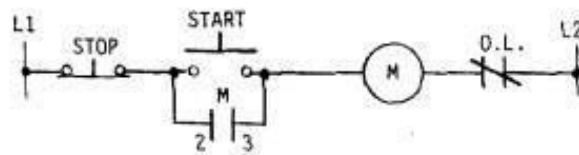


Figure 16: Ladder Diagram in Relay Logic

Just as in Figure 16 above, relay logic shows all components in the system. This is because relay logic is the same as the wiring diagrams that the electricians use, so all the wiring needs to be shown for the logic to work. Because of this, some components may not need to be included in the plc ladder logic diagram.

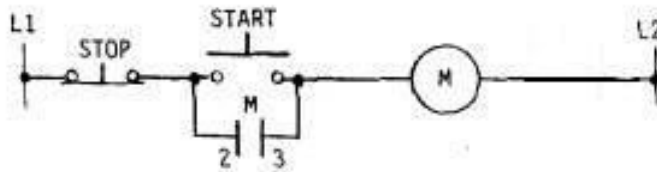


Figure 17: Relay Diagram with overload removed

Figure 17 above shows the same circuit as in Figure 16 with the overload removed. The overload is needed in relay logic because you have to have an overload device on any circuit; therefore it needs to be in the wiring diagram. This way, if you push too much current to the motor, the overload device will interrupt the circuit. Overloads are included internally in most any device anymore, but you will still see this in diagrams. There is still an overload device in a plc ladder logic circuit, but ladder logic shows only those components that have an input or output address, so you do not see it. In Figure 17, you can see that the start and stop buttons along with the motor relay will all be turned to inputs in the plc diagram and the motor, signified by a circle with an ‘M’ in the middle will be an output. The motor relay will not be a physical entity in the plc ladder diagram as it is in this relay logic. It will simply be an input that uses the same I/O address as the motor output. The stop button input can be located on either side of the start button/relay gate, as long as it is still in series with it.

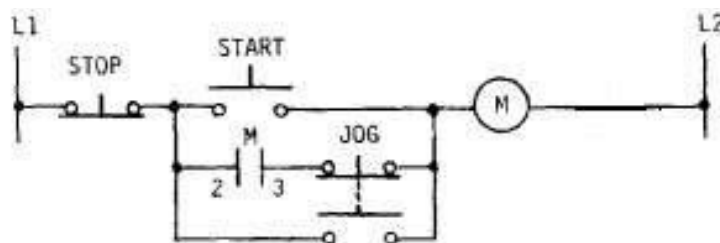


Figure 18: Relay Circuit with Addition of Jog Function

Figure 18 above shows the addition of a ‘Jog Function’ to the relay circuit. The jog function is generally added to any circuit for troubleshooting purposes only. Most jog functions are set up so that the only time the motor will run with the help of the jog

function is when the 'Jog Button' is pushed. In Figure 18 above, you can see this with the relay logic. As the circuit looks right now, when the Start Button is pressed, the motor will start, energizing the relay, and going across the Jog Button's normally closed contacts. The motor will stay running this way until the Stop Button is pressed. If instead the Jog Button is pressed, the current will travel across the normally open Jog contacts that are now closed. The motor will stay running until the Jog Button is no longer pressed.

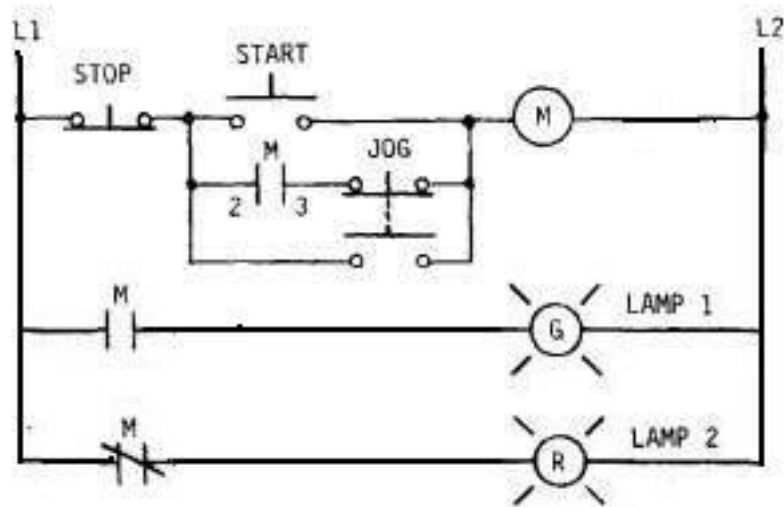


Figure 19: Relay Circuit with Addition of Status Indicators

Figure 19 above shows that same circuit with Status Indicators added. These are used in control rooms to inform users of the status of their motors or other moving parts. Green is the generally accepted color for a motor going, while red is stopped. The green light is energized when the normally open contact is energized by the moving motor, closing it. The red light is energized whenever a normally closed relay is closed, so it will turn off whenever the motor starts to run. From Figure 16 to Figure 19, one can see that with every component added, many wires need to be connected as well. Depending on how far away these components are away from each other, this can be very difficult and time consuming.

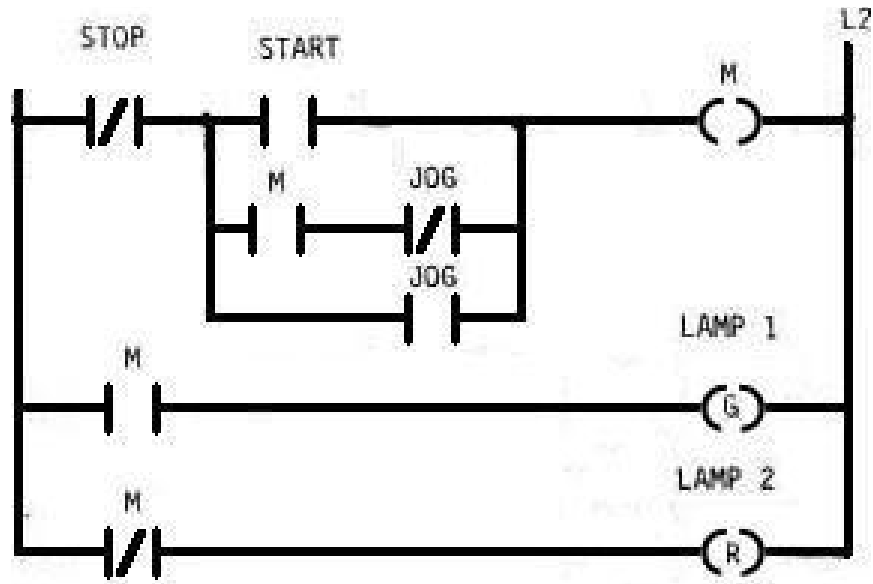


Figure 20: Relay Logic Diagram Converted to PLC Ladder Logic

Figure 20 above was converted from the relay logic in Figure 19 to the PLC

ladder logic seen here. If the PLC logic here was used in Figures 16-Figures 19, adding the various component would've taken much less time than physically wiring each component. PLC ladder logic can differ from relay logic in that different components are used as well. In the relay diagrams, a single button double pole switch was used so that it could perform two different functions. In PLC ladder logic, just a single pole button is needed, because the computer can be asked to look for a on or off state. For the status lights, instead of running wires to the motor relays the PLC diagram just looks for a true or false state of the motor output.

Conclusion

Summary

This report has discussed the role that programmable logic controllers have in the efficient design and control of mechanical processes. Also discussed was the understanding of ladder logic and the programming involved with it. Finally, the report has discussed relay logic and the evolution that ladder logic made from it. Four areas in regards to programmable logic controllers were addressed:

1. Programmable Logic History: This section discussed the history and advancement of controls technology, with a comparison of programmable logic controllers and hard-wired relays.
2. PLC components: This section defined what programmable logic is and described all hardware associated with it.
3. Ladder Logic: This section covered ladder logic and its general progression from relay logic.
4. Ladder Logic Programming: This section covered basic programming techniques and their implementation.

Closing Comments

With the speed of changing technology today it is easy to lose sight or knowledge of the basic theory or operation of programmable logic. Most people simply use the hardware to produce the results they desire. Hopefully, this report has given the reader a deeper insight into the inner workings of programmable logic and its role in mechanical operations. The idea of programmable logic is very simple to understand, but it is the complex programs that run in the ladder diagrams that make them difficult for the common user to fully understand. Hopefully this has alleviated some of that confusion.