Fundamentals of Ladder Logic Industrial Programming

An Online Continuing Education Course for Engineers

Course Number: IC-2013

Credit: 2 Hours / 2 PDH / 2 CPD

Fundamentals of Ladder Logic Industrial Programming

Robert Noriega, P.E.

Introduction

Programmable logic computers, or PLCs, are industrial computers that control automation for industrial processes. The most common programming language for PLCs is ladder logic, also called ladder diagrams, but there are also four additional industrial programming languages that we will look at as well. Ladder logic is a visual representation of an automation process. It is designed to look like an electrical relay diagram so that electricians can easily read it and troubleshoot it. IEC 61131-3 is the international standard for programming languages for PLCs, and as a result, we will be referring to it often in this course [1].

We will cover everything from how to read ladder logic to the different parts of a PLC. I will summarize some of the most common symbols used in ladder logic and provide real-world scenarios in which they may be used. Once you have a solid foundation in reading ladder logic, we will cover the basics of troubleshooting automation processes. Throughout the course, I will emphasize the advantages and disadvantages of ladder logic programming and provide reasons why we may use some of the alternative programming languages that are also at our disposal.

In this course, we will explore a variety of ladder logic examples. I will provide screenshots and will walk through the logic of each in words, but I highly recommend you also try it out yourself. Even if you do not have access to an actual PLC or the required software, there are a number of free, online ladder logic platforms that allow you to build and test your own logic. You will get the most out of this course if you test out what you learn and build the examples yourself to test and troubleshoot.

Ladder Logic Basics

Ladder logic is called as such because the programming resembles a ladder. The ladder consists of two rails connected by a series of rungs that contain the automation logic. You can imagine the left rail as electrically energized and the right rail as an electrical ground. As you move from left to right on a rail meeting the conditions of the various commands, more and more of the circuit is energized until you reach the right rail, completing the circuit. Another way to think of ladder logic from more of a computer science background is that the entire ladder is a giant WHILE loop, and each rung is an IF statement. Instead of logical AND and OR statements, the rung will have commands in series or parallel. When ladder logic instructions are put in series, the input connection of an instruction will equal the output connection of the previous instruction. When ladder logic instructions are put in parallel, the input and output of both instructions are linked, and if either instruction's conditions are met, it will allow power to flow.

Ladder logic is entirely comprised of the fundamentals of how a relay works, so it is important to first understand what a relay is and how it functions. In simple terms, a relay is an electronically-operated switch. It typically has a normally-open contact and a normally-closed contact, and when voltage is applied to an electromagnetic coil, the normally-open side will close (completing a circuit) and the normally-closed side will open (creating an open circuit).

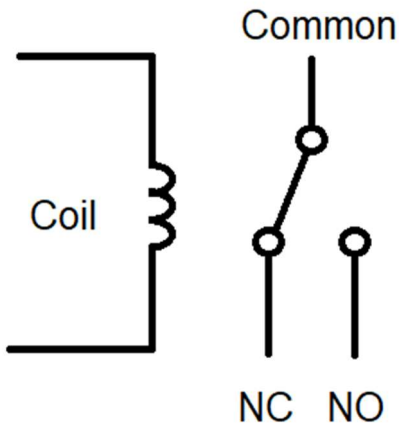

Figure 1 - Diagram of a relay showing the coil, common input, normally-closed contact, and normally-open contact. Source: Robert Noriega (self)

There are three main steps to how a PLC reads and processes ladder logic: read inputs, execute logic, and set outputs. During the input reading step, the PLC will poll each of the input devices connected to the PLC to determine their current state. It will then read the ladder logic stored in the processor of the PLC. The ladder logic will be read from top to bottom, evaluating each rung from left to right before proceeding to the next rung. The process of a PLC reading through the ladder logic is often referred to as a scan, and the speed at which the PLC can scan through the logic is directly proportional to the processing power of the PLC's CPU. Finally, based on the conditions of the ladder logic, the PLC will write new values to the output devices. The top-to-bottom nature of ladder logic can be significant for a few reasons. For instance, if you have multiple rungs setting an output to a particular value, the rung further to the bottom is going to set precedence because it is going to overwrite the value of the previous rung before the PLC has a chance to write it to the output device.

Throughout this course, you will encounter several different terms, such as Boolean variable, bit, or discrete input/output, that all refer to the same concept: a two-state signal. A Boolean variable is a data type that can have a value of either TRUE or FALSE. A bit is a binary digit that can have a value of 0 or 1. A discrete signal is a signal that can either be ON or OFF. In ladder logic and PLC programming in general, these terms are often used interchangeably. A signal may be described as 1, ON, or TRUE when

energized, and 0, OFF, or FALSE when de-energized. These terms all represent the same thing, which is whether a device or logical condition is active or inactive.

Now that you know the basic principles of ladder logic, let's look at some of the hardware components that make up a PLC before we begin looking at actually code.

PLC Basic Overview

PLCs come in a wide array of makes and models, but they all have the same basic components: the processor, input/output (I/O) modules, the communication module, and a power supply. Each component is typically connected via the backplane or chassis, which contains slots into which each module is inserted.

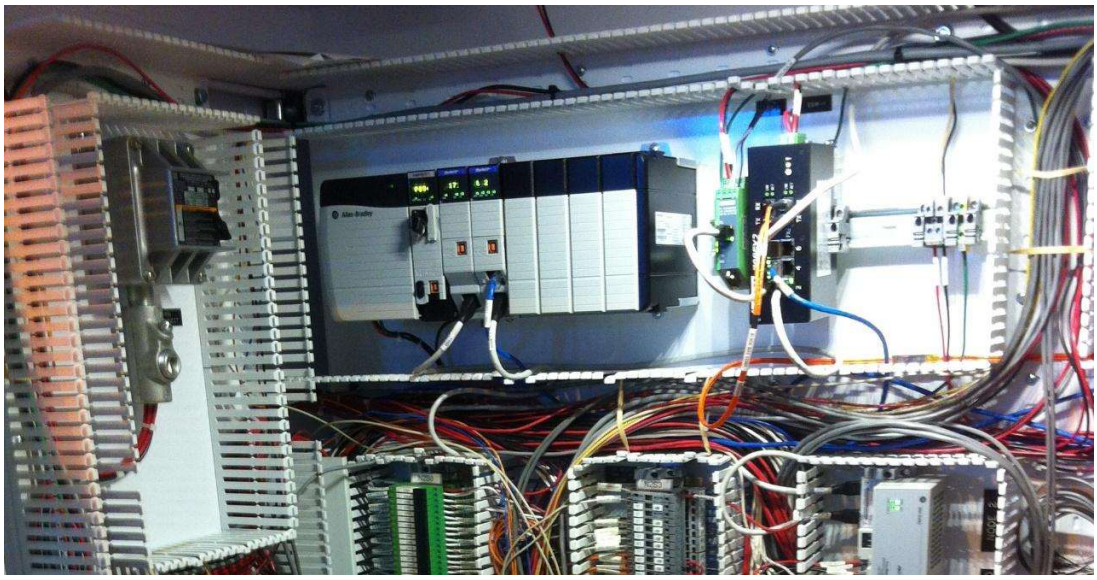

Figure 2 – A PLC panel. Source: Robert Noriega (self)

The processor or CPU (central processing unit) is where the program is stored. It also has data tables to relate the program to the I/O devices and periodically performs housekeeping and diagnostic tasks to keep things running smoothly. On most PLCs, the processor has a key switch that can switch the processor between run, program, and remote modes. A common issue people run into if a PLC is not letting them upload new logic is that the key switch is in the wrong setting. Your company may also have requirements on the default state to leave the key switch in to prevent cybersecurity attacks. Another important component of the processor is the RAM battery. This is typically a coin-style, lithium, 3V battery that keeps the RAM and internal clock from being wiped when the PLC experiences power interruptions. Depending on how often your PLC is without power, its lifespan could range from several

months to several years. PLCs typically have a battery indicator light that will notify when it should be replaced.

PLCs can have a wide variety of I/O modules to accommodate a wide array of devices. The most common modules are discrete input, discrete output, analog input, and analog output modules. Discrete input devices include limit switches and pushbuttons. Discrete outputs could be something as simple as an indicator light or the run signal to a contactor that controls a motor starting. Analog inputs include signals such as valve position indication, pressure transmitters, and temperature transmitters. Analog outputs include signals such as valve position command or a speed command to a variable frequency drive on a motor. The I/O modules allow the PLC to interface with all four of those main types of I/O devices. Still, there are other, more specialized modules, such as high-speed counter cards that take frequency inputs and temperature modules that receive raw data from thermocouples or RTDs and convert it to temperature readings for the PLC to use.

The communication module interfaces between other systems such as the SCADA (supervisory control and data acquisition) system, HMIs (human-machine interfaces), smart i/o devices, and any remote I/O panels not locally connected to the PLC. SCADA systems are commonly used in environments with multiple PLC systems that require monitoring by a central control room. The communication module allows the control room to send commands remotely to the PLC and receive data back in turn. PLCs will always have some sort of HMI to enable the technician to observe the status of various processes, see if there are any alarm statuses, trend data, and perform local manual control of the processes. The communication module allows for the HMI to poll data and send command signals to the PLC. Finally, if your process is big enough, you may have remote I/O panels to simplify wiring. Rather than have 50 process signals routed from end devices to your PLC 200 feet away, you can have all of those process signals wired to a much closer remote I/O (RIO) panel and then have one fiber optic line that goes from the RIO panel to the PLC. In this configuration, the RIO panel serves as an I/O module that the PLC's communication module can interact with to poll devices and send commands, even though the I/O is not located in a local I/O module.

The power supply powers all of the other modules in the PLC. The most common power supply voltage is 24 VDC. One thing to keep in mind is that this power supply generally does not power the field devices. Field devices will have their own power supplies. While this may be obvious for higher voltage devices like pumps and motors, it's also true for low-voltage sensors and transmitters. The PLC power supply's current capacity is usually limited to powering only the modules directly connected to the chassis.

All PLCs are a little different, so I encourage you to familiarize yourself with the specific hardware used by your company. The basic components that we've discussed function similarly across brands, but they are all going to look different and have slightly different features. If you can understand the differences, you will be able to troubleshoot more effectively, select the right modules for your automation process, and make the most of your PLCs' capabilities. Whether you're working with Allen-Bradley, Siemens, Schneider, Mitsubishi, or another manufacturer, explore technical manuals, wiring diagrams, and the

software tools specific to your application. The more comfortable you are with the specifics of your PLC hardware, the more confident and efficient you'll be in the field.

Basic Ladder Logic Symbols

There are ten basic ladder logic symbols, of which you need to be familiar. Each symbol will have a variable associated with it, in addition to an input (left connection) and an output (right connection). For contacts, the output connection of the contact will be set accordingly, depending on the state of the variable. For coils, the variable will be set accordingly, depending on the state of the input connection. In this section, we will look at the specific logic each of the basic ladder logics uses to translate an input condition to the associated output condition.

The two most basic types of contacts are normally-open and normally-closed contacts. For a normally-open contact, when the left connection is energized, the right connection will be energized if the associated variable is TRUE. This can be useful for performing a task when a condition is met. If you want to turn on a pump when a low-level switch is activated, this would be the type of contact to use. For a normally-closed contact, when the associated variable is energized, the right connection will be de-energized, and the output connection will be energized when the variable is de-energized. You would use this contact to perform a task when a condition is no longer met or to stop a task when a condition is met. Going back to the pump example. Assume, in addition to a low-level switch, you have a high-level switch. You could use a normally-closed contact such that when the high-level switch is activated, it will cut off the pump. The last two types of contacts are transition-sensing or edge-sensing contacts.

You may also see them referred to as one-shots because when activated, their output is energized for exactly one scan cycle of the PLC. For a positive transition-sensing coil, when the variable goes from de-energized to energized, the output connection will energize for exactly one scan of the PLC. This is commonly used for push buttons to make sure that when you press the button, it does not initiate a task multiple times. For a negative transition-sensing coil, when the variable goes from energized to de-energized, the output connection will energize for exactly one scan of the PLC. Imagine a dead-man switch for this type of contact. When the button is released, the output task will occur exactly once. See Figure 3 for a breakdown of how the variable state can affect the output connection of the four types of basic contacts.

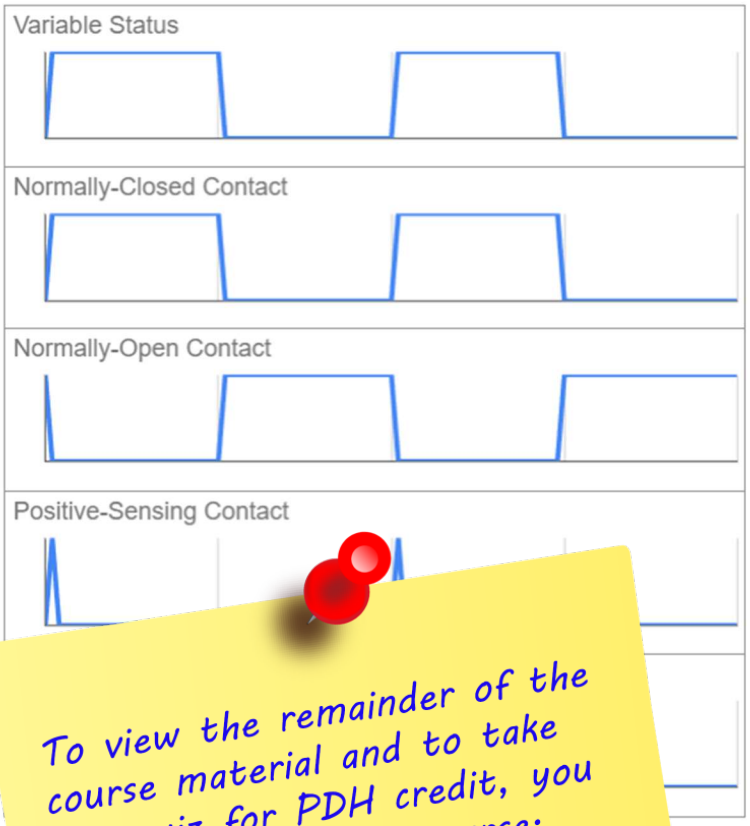

Figure 3 - Conta *Robert Noriega (self)*

Similar to contacts, the t ne normally-closed coil. For a normally-open coil, left connection. Consider a light that turn closed coil, the associated variable will be Considering indicator lights again, you might wa ime, indicating it's available. The next two go together in a pair, a as the set coil and the reset coil (also known as the latch and unlatch coils). When the input connection of a set coil is energized, the variable will be set to an energized state and will stay energized until the reset coil is energized. You use these whenever you want a condition to persist until you specifically tell it to stop. Perhaps you have a local manual bypass state that you want to enable to prevent remote starts during troubleshooting. You could have a local button that you press to enable the status (latch the local manual bypass). Once you are done, you will have another button to return to remote automatic status (unlatch the local manual bypass). When we look at examples, we will discuss where it is and isn't appropriate to use set/reset coils. However, before we get there, keep in mind the idea that ladder logic scans from top to bottom before setting output states because this can drastically affect set/reset coils.

You always want your reset rung to go after the set rung in your ladder logic. That is because if both the conditions for the set rung and the reset rung happened to be energized at the same time, if the set