A Beginner's Guide to the Programmable Logic Controller

By Max Ramer, ECE '22

Introduction

In this paper, I hope to provide an easy-to-understand tutorial for beginners trying to navigate their way through the world of the Programmable Logic Controller (PLC). PLCs have become the focal point of the minimum viable product that my team will deliver to our sponsor come May. As a senior Computer Engineer in the Tufts ECE department, I've never had the opportunity to take a class on PLC hardware or software of any kind. That being said, it was not difficult to pick up the basics because classic ECE concepts are at the heart of PLCs. Those who have basic circuit analysis and computer science experience should have no problem getting started. I hope this guide serves as an introductory resource for anyone starting out with PLCs like myself.

Brief History

PLCs were developed around the same time as computers in the 1960s; however, their impact was, and continues to be, less apparent to the public eye. The first PLC was developed in 1968 by Richard Morley of Bedford Associates as a solution for General Motors' desire for a Standard Machine Controller (Boyes, 2010). General Motors required a device that was modular, extensible, durable, and cheap. Most importantly, they sought a device that could replace hardwired relays and messy circuitry in their automobile factories (Hughes, 2005). Morley and his team exceeded General Motors' expectations with the design of the first ever PLC. Shortly thereafter, they created a new company to focus strictly on PLC development. About the size of a suitcase, a PLC could replace cabinets with relays and electrical wiring that previously took up walls of space in factories (Greeff, 2004). The company that Morley and his team founded was bought out by the company that is known today as Schneider Electric.

Basic Functionality

While PLCs have evolved into powerful and complex computers, the basic functionality has remained the same since the '60s. At a high level, a PLC performs logic on a given set of inputs to change its output states. You can think of the PLC as a black box that takes input and spits out output.

As the programmer, you program the logic that determines how the input changes the output. A PLC can do simple tasks very quickly and efficiently. The basic functionality of a PLC is as follows:

1. The CPU reads the state of its inputs (i.e., a switch is off, valve is open, etc.)
2. The CPU performs logic on the input states as designed by the programmer
3. The CPU changes the state of its outputs (i.e., switch turns off, valve closes, etc.)

(Reddy, 2015)

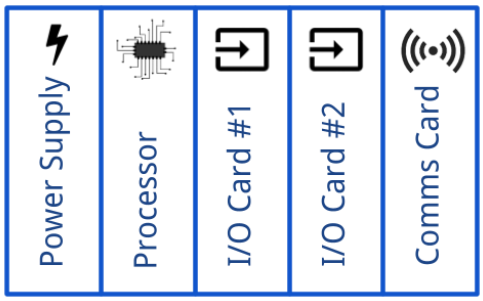

Figure 1. Component stack diagram of a standard PLC (Based off of Instrumentation Tools, 2015).

The Anatomy of PLCs

A PLC can be broken down into four essential components illustrated by Figure 1 above: the power supply, processor, input/output system, and communications card (Hughes, 2005). Similar to standard computers, the processor, or CPU, is the brain of the device and does the bulk of the work. The CPU performs the control logic designed by the programmer and contains the memory required to store code and data. The input/output system provides a means of connecting the PLC to external devices such as motors, switches, and sensors. Notice that there can be several I/O systems on the device. The communication module enables the use of comms networks in PLC programming.

Software & Programming

PLC software involves very low-level programming that is not too hard to pick up. The two most common programming languages are Ladder Logic and Functional Block Diagram (Austin, 2015).

Ladder Logic

PLC Academy describes Ladder Logic as a language which "expresses logic operations with symbolic notation … that is made out of rungs of logic, forming what looks like a ladder - hence the name Ladder Logic" (Peter, 2017). It was originally developed for electrical engineers and others who would find programming in a visual/schematic nature easier than typing out text. The core instructions of Ladder Logic are known as examine if closed (XIC), examine if opened (XIO), and output energized (OTE). The first two instructions can be thought of as normally closed and normally open switches, while the last instruction can be thought of as a relay that passes on the current result. Figure 2 below is an example of a Ladder Logic program.

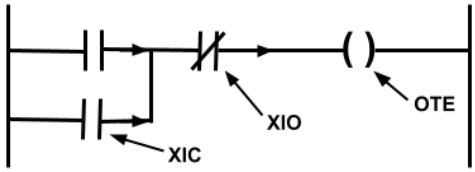

Figure 2. Sample Ladder Logic program. The XIO and XIC symbols are the 'rungs' of the ladder.

Functional Block Diagram

Functional Block Diagram (FBD) involves more systems engineering and block diagrams than Ladder Logic - hence the name Functional Block Diagram. It replaces lines of code with functional boxes with one or more inputs and outputs, similar to the black box method discussed earlier. The basic functional blocks of FBD share similar functionality as the basic logic gates and latches in digital design: AND, OR, NOT, XOR, assignment operation, bit-stable, and edge detection. Assignment operation is equivalent to the OTE instruction in Ladder Logic, while bit-stable and edge detection are equivalent to latches and clock-controlled devices, respectively. My team has found FBD to be easier to learn than Ladder Logic however both can be picked up fairly quickly.

Exercise: PLCs in Practice

Now that you're familiar with PLCs, I'm going to give you a brief description of the problem that my team solved in senior design and let you decide if using PLCs was the right call.

My team and I are designing a smart water distribution system for a school in Africa. Currently, the school relies on manual transportation to get water from a main tank on the outskirts of the school to various sub-tanks inside the campus. The school requires a water distribution system that is:

1. Digitally controlled and automated so trucks don't need to manually fill up water tanks
2. Durable and reliable; capable of lasting at least 10 years
3. Easy to control and monitor when something goes wrong

See Figure 3 below for our proposed solution.

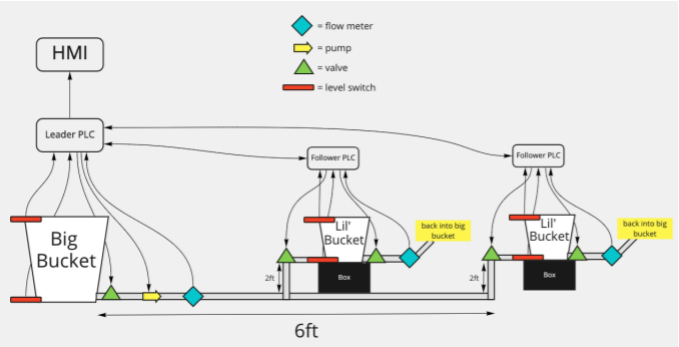

Figure 3. My team's proposed solution to our senior design problem.

Resources & Wrap-Up

There are countless online resources that can provide a much more in-depth explanation of how PLCs work and their applications in industry. A good place to start is myplctraining.com, which has a four-part tutorial that walks readers through PLC functionality using intuitive graphics. For readers looking for a more academic approach, I strongly suggest *Programmable Logic Controllers, 5th Edition* by Hughes. The text is beginner-level and very descriptive with simple examples. It is the first citation in this guide. You also can't go wrong with a simple Google or YouTube search. I found several instructional and reputable sources such as PLC Academy and Instrumentation Tools with a quick search.

I hope this guide was informative and as non-technical as possible. PLCs are a true engineering marvel that have forever changed industrial automation. They will continue to be improved upon as our understanding of nanotechnology increases. PLCs will likely be at the heart of industrial automation for years to come.

References

1. Hughes, Thomas, A.. (2005). Programmable Controllers (4th Edition) - 1.1.1 Brief History of PLCs. (pp. 2). International Society of Automation (ISA). Retrieved from https://app.knovel.com/hotlink/pdf/id:kt007UN2P2/programmable-controllers/brief-history-plcs

2. Greeff, Gerhard Ghoshal, Ranjan. (2004). Practical E-Manufacturing and Supply Chain Management - 2.3.6 Advanced Process Control. (pp. 12,13). Elsevier. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt009X8OC2/practical-e-manufacturing/advanced-process-control

3. Reddy, Y. Jaganmohan. (2015). Industrial Process Automation Systems - Design and Implementation - 2.1 Introduction to the Programmable Logic Controller. Elsevier. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt00U9CUB1/industrial-process-automation/introduction-programmable

4. Scott, Austin. (2015). Learning RSLogix 5000 Programming - 6.1.2.1 Function Block versus Ladder Logic. (pp. 98). Packt Publishing. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt011DLY91/learning-rslogix-5000/function-block-versus

5. Staff, E. (2021, March 4). PLC components - programmable logic controllers. Instrumentation Tools. Retrieved January 19, 2022, from https://instrumentationtools.com/components-of-plc/

6. Michael, S. (2019, November 5). An overview of software languages for programmable logic controllers (plcs) - technical articles. Control. Retrieved January 19, 2022, from https://control.com/technical-articles/an-overview-of-software-languages-for-programmable-logic-controllers-plcs/

7. Peter. (2017, September 4). PLC ladder logic programming tutorial (basics). PLC Academy. Retrieved January 19, 2022, from https://www.plcacademy.com/ladder-logic-tutorial/

8. Boyes, Walt. (2010). Instrumentation Reference Book (4th Edition). (pp. 617,). Elsevier. Retrieved from https://app.knovel.com/hotlink/toc/id:kpIRBE0016/instrumentation-reference/instrumentation-reference