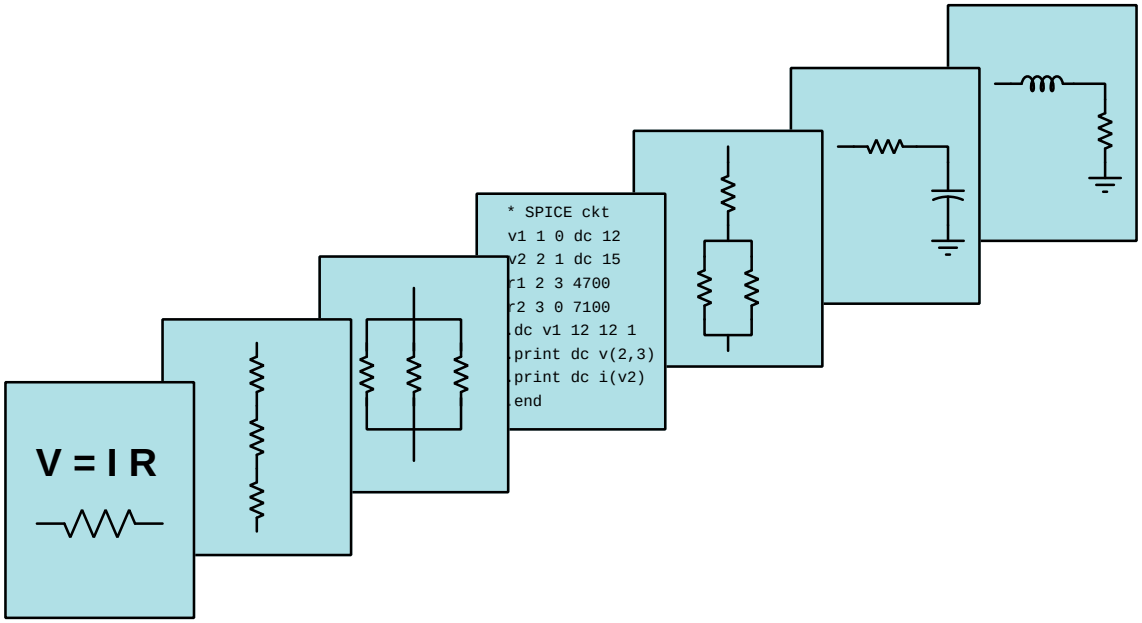
Modular Electronics Learning (ModEL) project



Relay Ladder Logic

Last update = 18 April 2025

Contents

Chapter 1

Introduction

1.1 Recommendations for students

Electromechanical relays are electrical switches actuated by the magnetic force of an electromagnet coil. When a relay's coil is energized with a suitable amount of electric current, the magnetic field produced by that coil acts upon a movable iron *armature* which in turn causes switch contacts to open and/or close. Relays are incredibly useful devices even today, and in some applications are even preferable to semiconductor-based switching circuits.

A common form of schematic diagram used to document relay-based circuits is the so-called *ladder diagram*. Not only does this diagram type lend itself well to documenting connections between the coils and contacts of relays, but it is actually used as a form of *graphical programming language* for a class of industrial control computers called *Programmable Logic Controllers*, or *PLCs*.

Important concepts related to digital logic and digital signals include **logic states**, **logic functions**, **truth tables**, the **normal** state of a switch, **logic states**, **hot** and **neutral** AC line conductors, and **three-phase** electric power.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to determine the "normal" switch contact states for an unlabeled relay? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How might an experiment be designed and conducted to measure the pick-up voltage for a relay? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How might an experiment be designed and conducted to measure the drop-out voltage for a relay? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- What are some practical applications of electromechanical relays?
- What is the fundamental characteristic of an OR logic function?

- What is the fundamental characteristic of an AND logic function?
- What is the fundamental characteristic of a NOT logic function?
- What is the fundamental characteristic of a NOR logic function?
- What is the fundamental characteristic of a NAND logic function?
- What type of switch contact interconnections form an AND function?
- What type of switch contact interconnections form an OR function?
- How do ladder diagrams differ from normal schematic diagrams?
- What does it mean to say that a switch contact is either *normally-open* or *normally-closed*?
- How is the concept of "normal" switch contact state often misunderstood or misapplied?
- Why is it important to never change a NO contact to NC (or vice-versa) as a way to indicate its present state?
- What is the significance of identically-numbered wires in a ladder diagram?
- What is the purpose of a programmable logic controller?

1.2 Challenging concepts related to relay ladder logic

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Normal status of a switch** – to say that a switch is "normally open" or "normally closed" refers to its electrical state *when at rest*, and not necessarily the state you will typically find the switch in. The root of the confusion here is the word *normal*, which most people take to mean "typical" or "ordinary". In the case of switches, though, it refers to the zero-stimulation status of the switch as it has been manufactured.

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

  Assessment – Interpret elements of a real ladder-logic diagram such as the one shown in the Tutorial's "Ladder diagram example" section. Relevant details to interpret include wire numbers, locations of "Motor Forward" (MF) and "Motor Reverse" (MR) contacts and coils, "Master Control Relay" (MCR) location and function, "seal-in" contacts for making the motor continually run after momentarily pressing the Start pushbutton switch, etc.

- **Outcome** – Apply the concept of "normal" contact status to relay circuits

  Assessment – Predict the status of every relay contact and every relay coil in a ladder-style diagram for various input switch state combinations. A good starting point would be the circuits shown in the Tutorial for basic logic functions with truth tables provided (to check the final result).

- **Outcome** – Diagnose a faulted relay-logic circuit

  Assessment – Predict the effect(s) of a single component failing either open or shorted in a relay-logic circuit; e.g. pose problems in the form of the "Effects of a ground fault in a relay control circuit" Diagnostic Reasoning question.

  Assessment – Determine the probability of various component faults in a relay-logic circuit given symptoms and measured values; e.g. pose problems in the form of the "Identifying possible faults in a relay circuit" Diagnostic Reasoning question.

- **Outcome** – Independent research

  Assessment – Locate relay datasheets and properly interpret some of the information contained in those documents including coil ratings, contact ratings, cycle life, etc.

Chapter 2

Case Tutorial

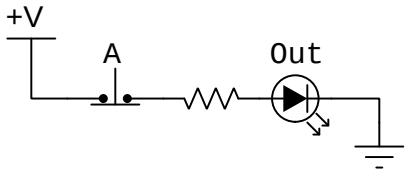
The idea behind a *Case Tutorial* is to explore new concepts by way of example. In this chapter you will read less presentation of theory compared to other Tutorial chapters, but by close observation and comparison of the given examples be able to discern patterns and principles much the same way as a scientific experimenter. Hopefully you will find these cases illuminating, and a good supplement to text-based tutorials.

These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

2.1 Example: simple logic functions using switches

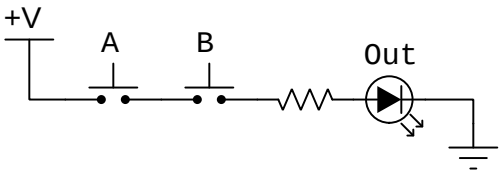
In each of the following circuits, the resistor serves to limit LED current to a safe value. A "1" state refers to a *pressed* pushbutton switch in the context of the logic function's input, and to an *energized* LED in the context of a logic function's output:

NOT function



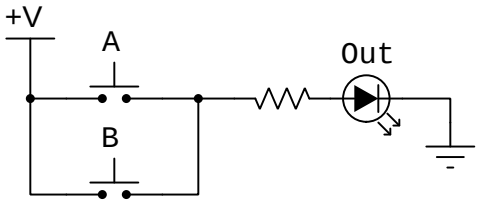
A	Out
0	1
1	0

AND function



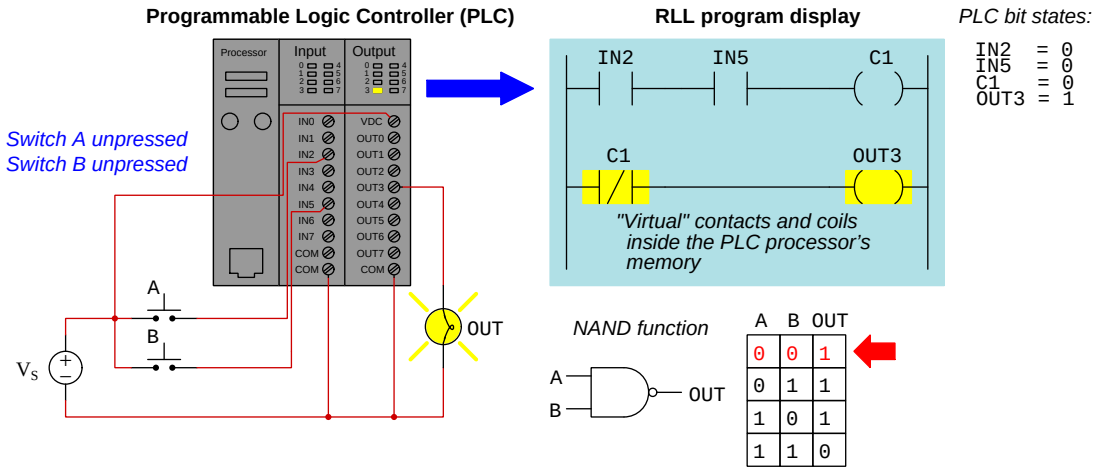
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

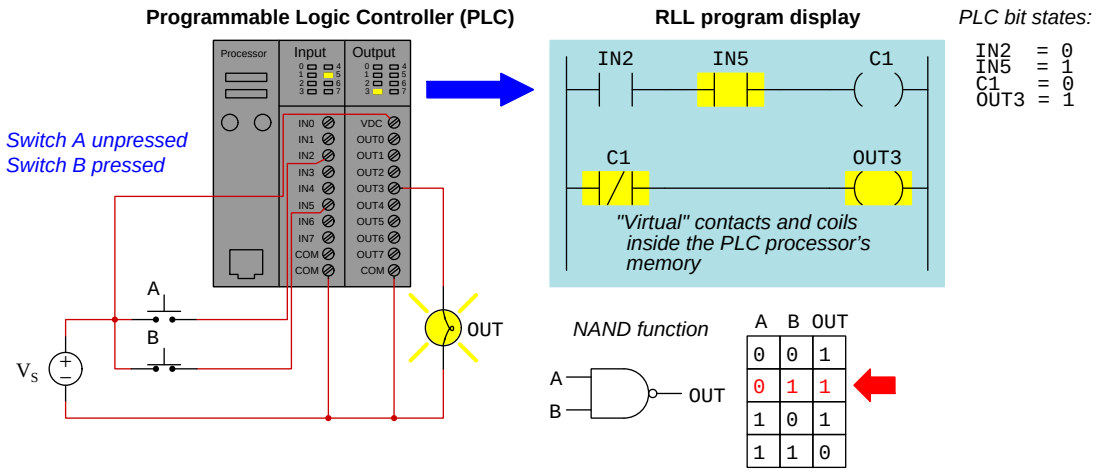
OR function

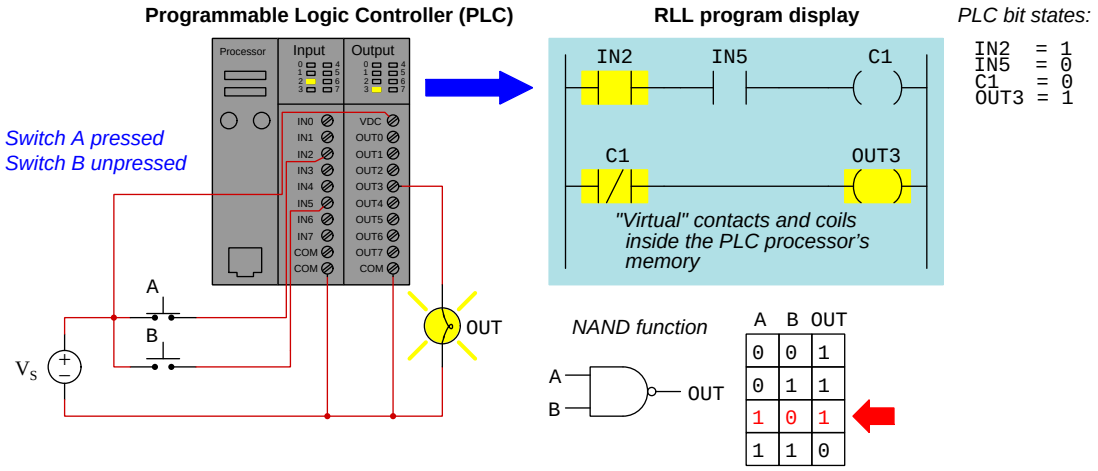


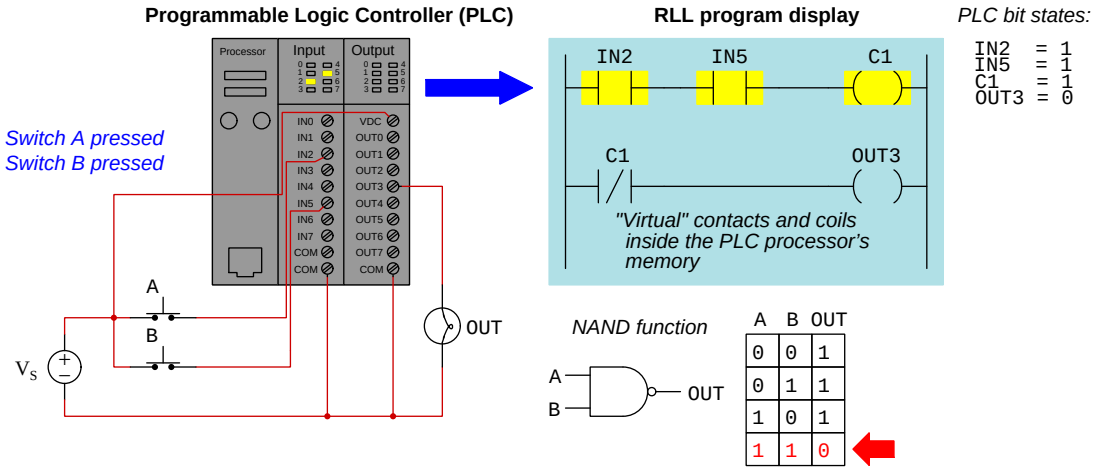
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

2.2 Example: NAND function in a PLC






Programmable Logic Controller (PLC)
RLL program display
PLC bit states:
Processor
Input
Output
Switch A pressed
Switch B unpressed
A
B
OUT
IN2
IN5
C1
C1
OUT3
"Virtual" contacts and coils inside the PLC processor's memory
IN2 = 1
IN5 = 0
C1 = 0
OUT3 = 1
NAND function
A B OUT
0 0 1
0 1 1
1 0 1
1 1 0


Programmable Logic Controller (PLC)
RLL program display
PLC bit states:
Processor
Input
Output
Switch A pressed
Switch B pressed
A
B
OUT
IN2
IN5
C1
C1
OUT3
"Virtual" contacts and coils inside the PLC processor's memory
IN2 = 1
IN5 = 1
C1 = 1
OUT3 = 0
NAND function
A B OUT
0 0 1
0 1 1
1 0 1
1 1 0

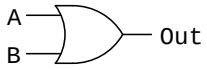
Chapter 3

Tutorial

3.1 Logic function review

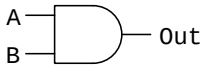
Electromechanical relays have many practical purposes, but in this tutorial we will focus on the use of relays to construct *logic circuits*: electrical circuits intended to fulfill some discrete-control function. The fundamental building-blocks of logic systems are the *logic functions* of *OR*, *AND*, and *NOT*, from which we may form logical systems of great complexity. To begin, let us review each of these functions along with their respective *truth tables*:

OR function



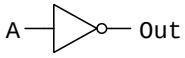
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

AND function



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

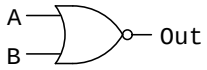
NOT function



A	Out
0	1
1	0

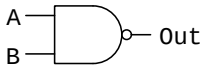
By combining OR and AND functions with inversion (the NOT function), we may form two more common "building-block" logical functions, *NOR* and *NAND*:

NOR function



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

NAND function

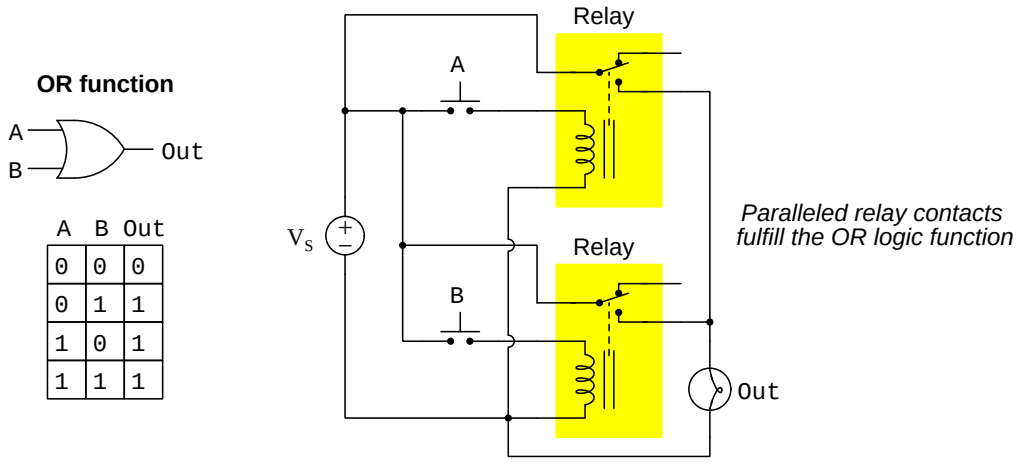


A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

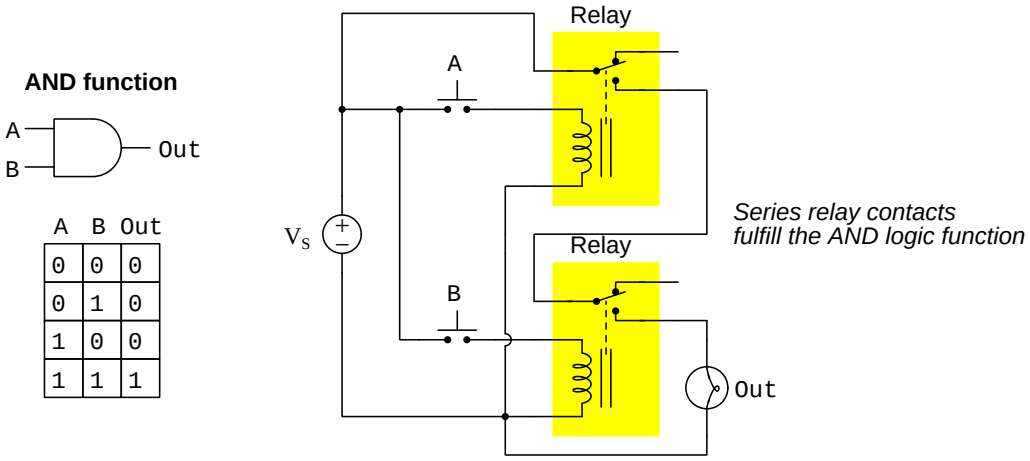
Next we will examine equivalent relay circuits in schematic form fulfilling each of these logic functions, interpreting a "1" state as *energized* and a "0" state as *de-energized*. Normally-open pushbutton switches will provide input signals to our logic function, while a lamp provides visual output signal confirmation.

3.2 Relay logic

Our first example will be an OR function, energizing the output if either *or* both of the inputs are energized (i.e. pushbutton switches closed). The two relays' normally-open contacts are connected in parallel with each other to fulfill the OR functionality, allowing the lamp to energize if either contact actuates:

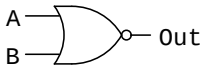


Next, we will see an example of the AND function, energizing the output only if both input A *and* input B energize. The two relays' normally-open contacts are connected in series with each other to fulfill the AND functionality, allowing the lamp to energize only if both contacts actuate:

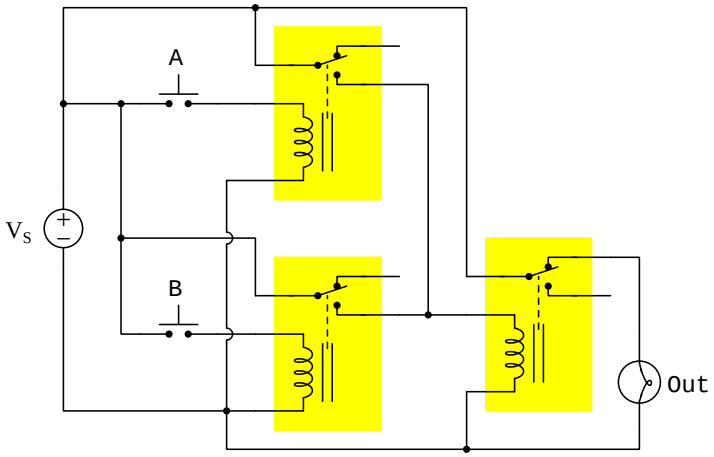


The NOT function may be implemented in relay form by using the normally-closed (NC) relay contact, so that it conducts electricity when the coil is de-energized and breaks the circuit when the coil is energized. Attaching such a logical "inverter" to the output of the OR and AND relay circuits creates NOR and NAND functions, respectively:

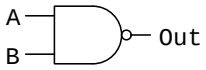
NOR function



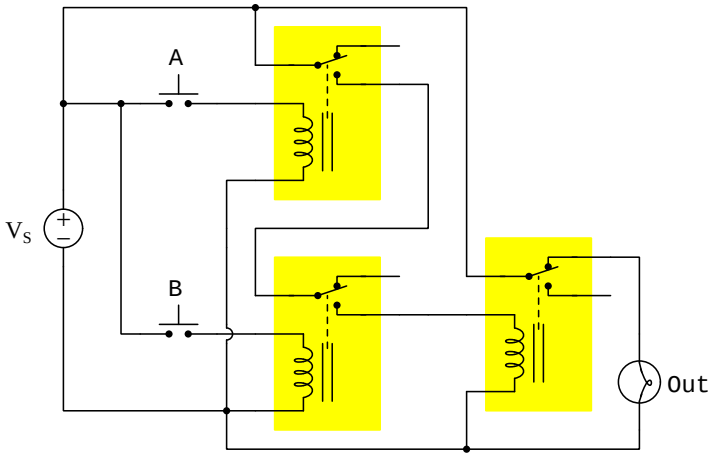
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0



NAND function



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

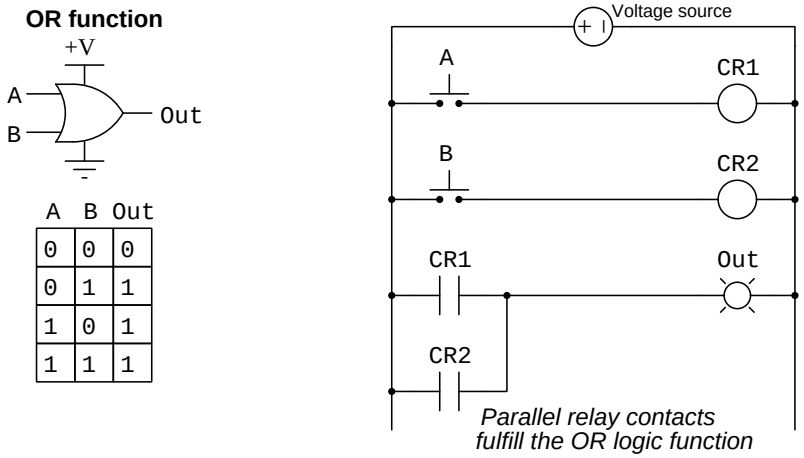


3.3 Ladder diagrams

It should be evident at this point that schematic diagrams are a clumsy form of documentation for circuits having more than a couple of electromechanical relays. The chief reason for this is the way in which schematic diagrams link relay coils with relay contacts: by dashed lines joining those two sets of symbols in close proximity. If we can find a different way to show the magnetic links between relay contacts and their respective coils, our diagrams could be significantly rearranged for neater appearance.

This is the basic premise of a *ladder diagram*. In this form of electrical diagram, relay coils and contacts each bear identifying *names*[1], the association between contact(s) and coil being made by name rather than by connecting dashed lines. To further simplify this diagram form, the two poles of the voltage source powering the relay circuit are shown as two vertical lines[2], one at each edge of the page. Furthermore, relay coils are represented as circles, and relay contacts as separated lines, in ladder diagrams.

To illustrate, the first logic function previously shown in schematic diagram form (OR) will now be re-drawn in ladder diagram form:

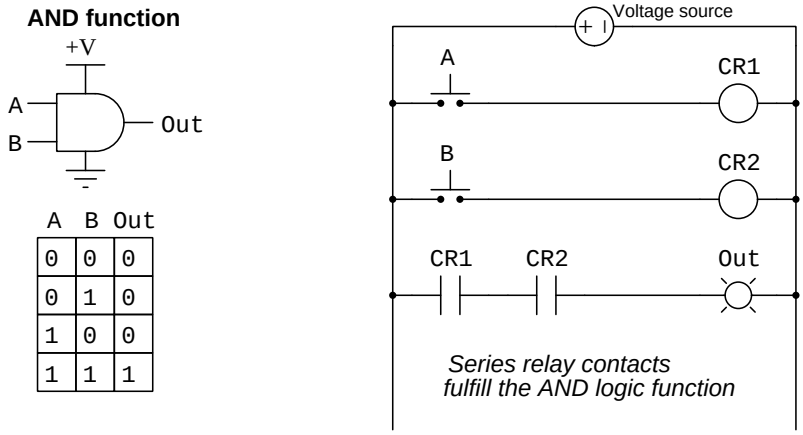


This ladder-style diagram is so much clearer than the schematic-style diagram previously shown, owing to the complete absence of any crossing lines. The parallel connection between the two relays' normally-open contacts is quite obvious in the ladder diagram, and the lack of dashed lines connecting relay contacts to their respective coils unclutters the illustration.

[1] Any name is permitted, but it is commonplace to find relays identified by *CR* ("Control Relay") labels such as CR1, CR2, CR3, etc.

[2] Admittedly, this symbol does look a like lot the one for a non-polarized *capacitor*. When capacitors are represented in ladder diagrams (which happens to be rare), the polarized-style symbol (with one curved plate) is commonly used to distinguish it from a normally-open contact.

The ladder diagram for the AND logic function is just as clean and easy to interpret:

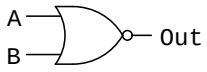


Note how voltage source connections appear in both the semiconductor logic gate and the relay logic circuit diagrams. This is not typically the case, it being conventional to omit such power source connections from both relay ladder diagrams as well as semiconductor logic gate diagrams for the sake of simplicity. Often we will see the left- and right-hand vertical lines in a relay ladder diagram simply unconnected, the assumption being that these lines will connect to an appropriate voltage source for the circuit to operate. In AC-powered relay circuits, the left-hand vertical line of the ladder diagram is typically labeled "L1" (line 1), and the right-hand vertical line "L2" (line 2). By convention, the right-hand line connection is usually connected to Earth ground, which makes it the *neutral* conductor, while the left-hand line connection is ungrounded and therefore considered "hot". These details are unimportant for the logical function of the circuit, but they are very important for considerations such as fusing (overcurrent protection)[3] and general safety.

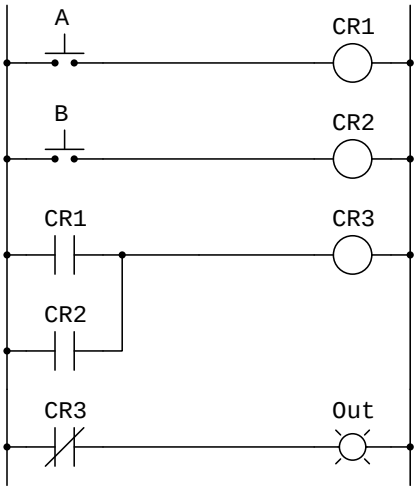
[3]Only ungrounded power conductors should be interrupted by overcurrent protection devices such as fuses and/or circuit breakers, the grounded (neutral) conductor(s) always maintaining electrical commonality with Earth. This maintains as many conductors at Earth potential in a line-powered circuit as possible in all operating conditions, and ensures "hot" conductors are disconnected following a high-current fault.

Inversion (NOT) functions require normally-closed relay contacts, and the ladder diagram symbol for this is a pair of separated lines with a diagonal line drawn through it to denote its normally-closed (NC) function. This will be illustrated in the following ladder diagrams for NOR and NAND functions, respectively:

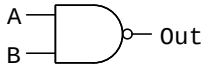
NOR function



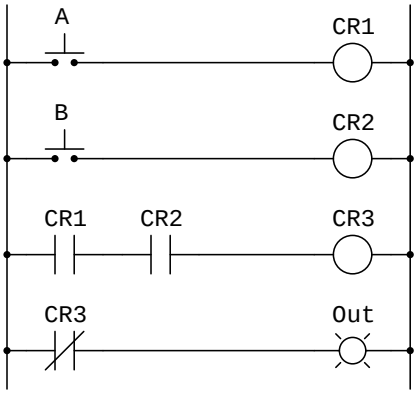
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0



NAND function



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

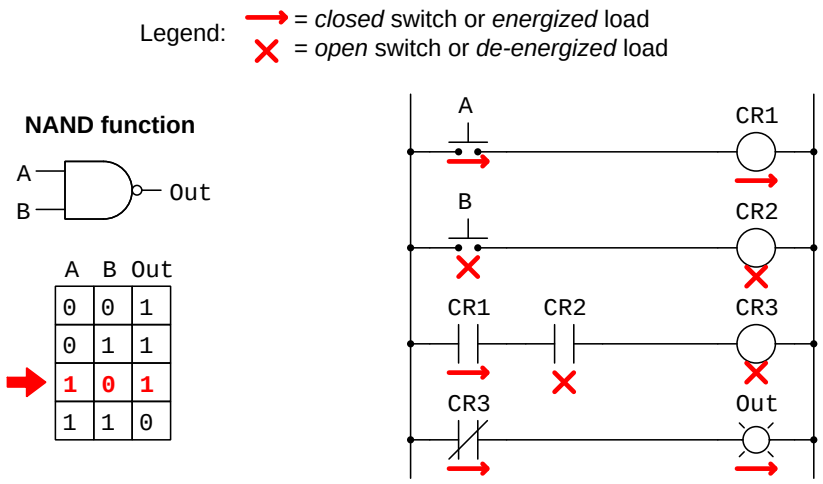


It is very important to realize that the diagonal line drawn through a relay contact represents its *normal state* and not necessarily its *present state* or *typical state*. In fact, this is one of the most common misconceptions for students first encountering ladder diagrams.

3.4 Annotating ladder logic circuits

A helpful analytical strategy for all kinds of circuit analysis is to *annotate* electrical states and conditions on the schematic diagram. Relay ladder logic circuits are no exception to this, and so here we will explore ways to denote the live state of each and every contact and load in the circuit in order to follow the logical relationships between them. As previously mentioned, the presence or absence of a diagonal line on a relay contact symbol tells us nothing about its state at any given time, but only its state *when that relay is at rest* (i.e. relay coil de-energized). A convention I have grown to use for the purpose of annotating a ladder diagram with live status is to draw a colored arrow or line just below any contact that is closed (to show its electrical conductivity) and to draw a colored "X" symbol just below any contact that is open (to show its lack of conductivity). The same arrow vs. "X" symbology is useful for denoting the state of loads (e.g. relay coils, lamps) in the circuit.

Let's annotate this NAND function relay circuit for a condition where A = 1 and B = 0:



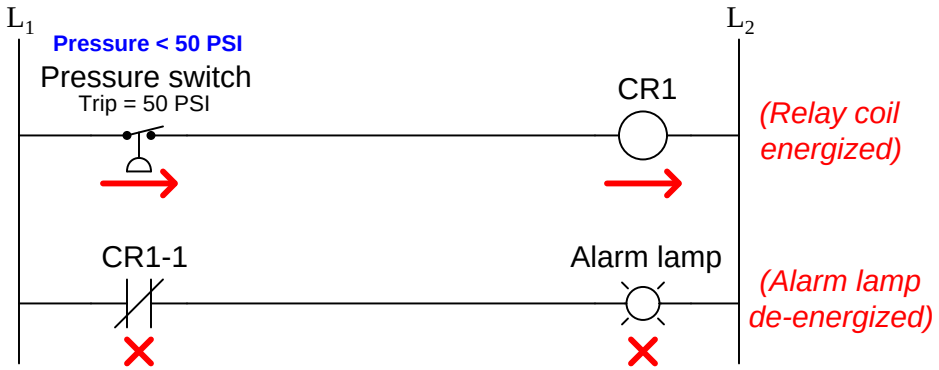
Note how each normally-open relay contact follows the status of its respective coil: a de-energized relay coil leaves its NO contact in the open state (e.g. CR2); an energized relay coil actuates its NO contact to be closed (e.g. CR1). The one and only normally-closed relay contact in this circuit exhibits a state opposite that of its coil, as expected for an NC contact: a de-energized CR3 coil leaves NC contact CR3 in its resting (closed) state; if coil CR3 were to energize, it would actuate contact CR3 to become open and turn off the lamp.

Again, it is very important to leave the relay contact symbols as they were originally drawn – NC contacts with diagonal lines, and NO contacts with none – because the presence or absence of a diagonal line indicates the *normal* state of that contact, but not its *actual*[4] state in any given condition.

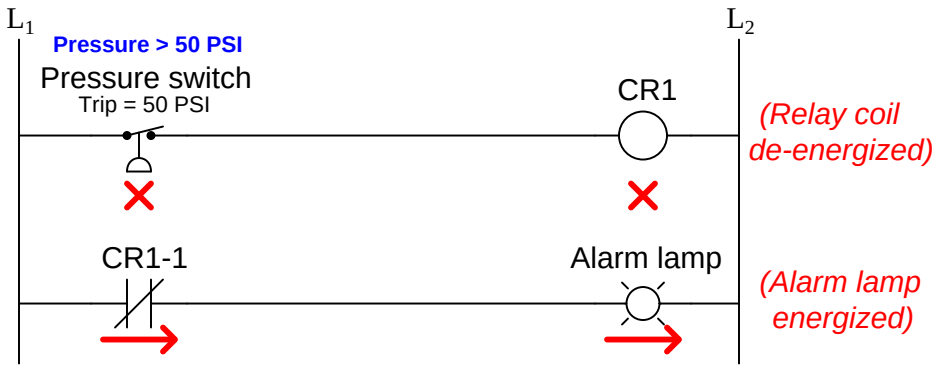
[4] An all-too-common tendency among students and working professionals alike is to draw diagonal lines to show when a contact *closes*, but this erroneously conflates normal status with present status. I have met many people who do this, and I have also met many people who struggle to analyze complex ladder-diagram systems *for precisely this reason*. I urge you to avoid this bad habit as you learn to drawn and analyze ladder diagrams!

More examples are always helpful, so let's consider another one. Here we see a ladder diagram of a circuit containing a *pressure switch*, that switch actuated by some fluid pressure rather than by someone's hand. This pressure switch happens to have *normally-closed* contacts, which means those contacts will be electrically closed when there is no fluid pressure applied to the pressure switch (i.e. when the pressure switch is at rest). The relay it's wired to also happens to have normally-closed contacts, which means the relay's contacts will be closed when the relay coil is de-energized (i.e. when the relay is at rest).

When less than 50 PSI of fluid pressure exists at the switch, the circuit is in the state shown below (by the arrow and "X" symbols):



Using arrow and "X" symbols again to represent the presence or absence of power in this circuit, we will now analyze its status with an applied switch pressure greater than 50 PSI:



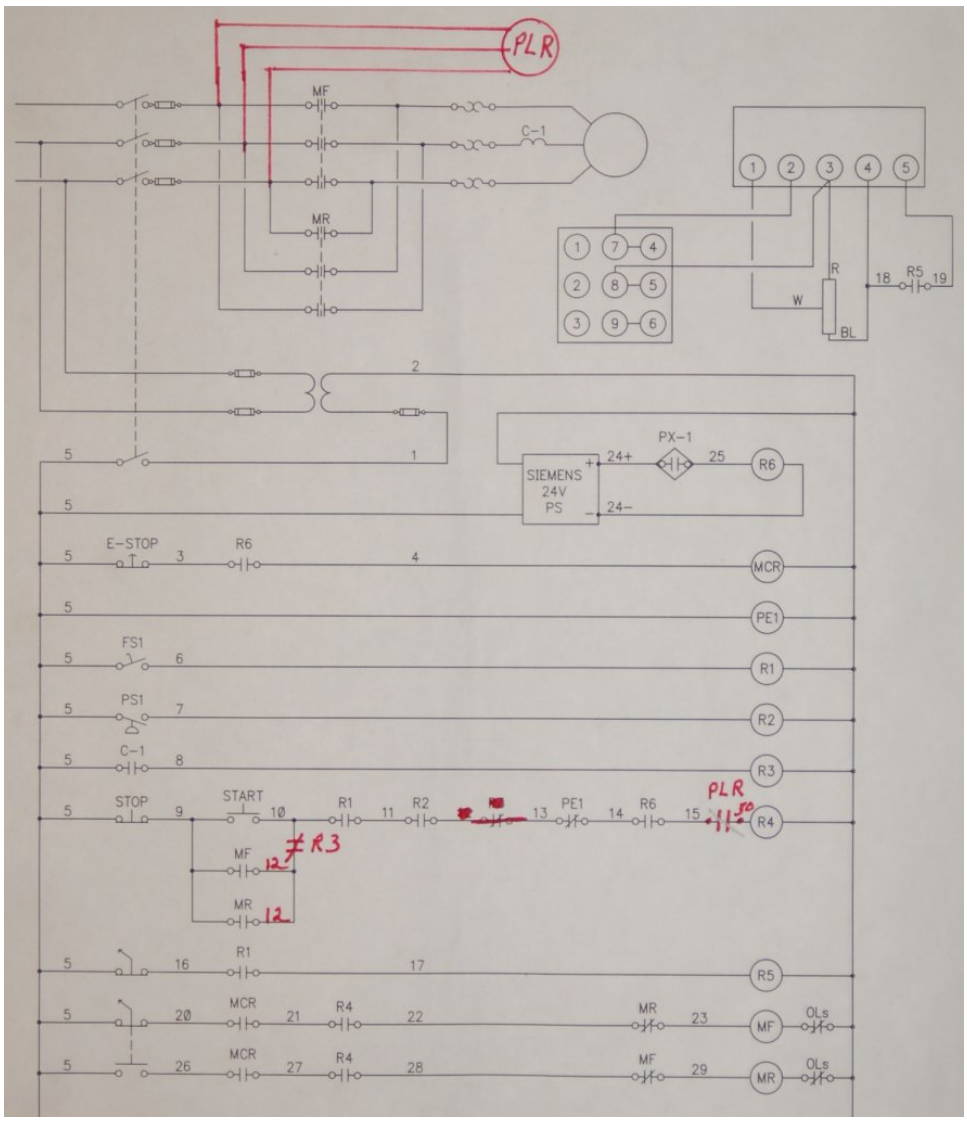
Now that there is sufficient fluid pressure applied to the switch to actuate it, its contact is forced into the actuated state which for this "normally-closed" switch is open. This open condition de-energizes relay coil CR1, allowing relay contact CR1-1 to spring-return to its normal status (closed), thus sending power to the alarm lamp. From this analysis we see that the lamp fulfills the function of a *high pressure alarm*, energizing when the applied pressure exceeds the trip point.

Where students typically find themselves confused is assuming the switch contact will be in the same state it is drawn in. This is not necessarily true. The way switch contacts are drawn merely

reflects their *normal* status as defined by the device manufacturer, which means the electrical status of the device's contacts when there is no (or insufficient) actuating stimulus present. Whether or not the switch will actually be in its normal state at any given time is a question of whether or not a sufficient stimulus is present to actuate that switch. Just because a switch is drawn normally-closed does not necessarily mean it *will* be closed when you go to analyze it. All it means is that the switch will be closed *when nothing actuates it.*

3.5 Ladder diagram example

An actual ladder diagram of a relay-based motor control system is shown here, complete with *red-line* edits showing modifications to the circuit made by an industrial electrician:



The red-lined edits made to this diagram show a *power loss relay* (PLR) added to the motor circuit, designed to interrupt power to control relay `R4` in the event of a loss of three-phase electrical power. Also changed in this diagram is the location of the normally-closed contact `R3`, which is actuated by overload relay C-1.

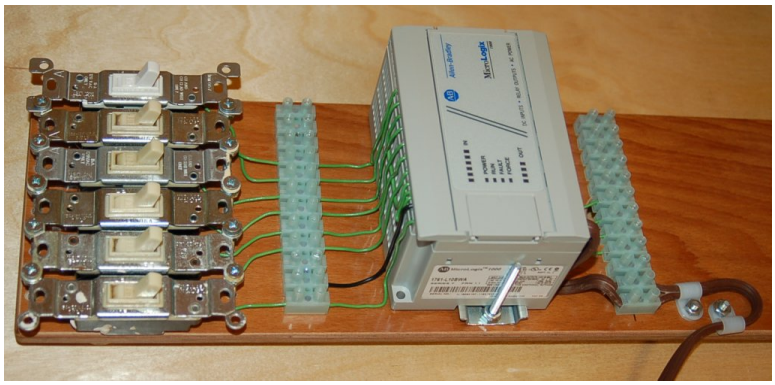
Note how every electrically-common conductor in this motor control circuit bears the same wire

number[5]. This is most evident in wire #5 (on the left-hand side of the diagram), but also in wire #12 which is a new label with the relocation of relay contact R3, and also in wire #50 which is a new label with the addition of the PLR contact[6]. Note how all electrically distinct wires (i.e. separated from each other by one or more components) have differing wire numbers.

3.6 Ladder logic PLC programming

A special type of industrial computer called a *Programmable Logic Controller* or *PLC* uses ladder diagrams as a programming language, instructing the PLC how it should respond to input signals from devices such as switches. PLCs offer all the capabilities of relay-based logical circuits, but with the added benefits of flexibility (in that they need only be re-programmed, not re-wired, to behave differently). PLCs also offer a host of features either impossible or inconvenient to implement using electromechanical relays, which is why they are so popular as industrial control devices. The reason for ladder diagrams as a programming language is to make PLC programming more accessible to electricians and other technical personnel already familiar with relay circuit ladder diagrams.

Here we see a MicroLogix 1000 PLC manufactured by Rockwell Automation under the "Allen-Bradley" name, attached to a wooden board and six toggle switches to be used as a "trainer" for students to learn PLC programming:



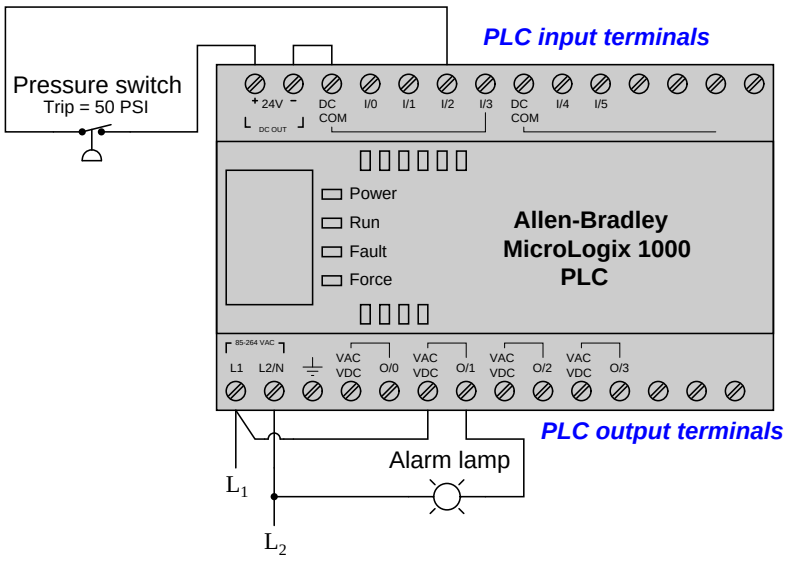
PLCs are complex enough to deserve their own tutorial, but we will explore basic PLC functionality here because it relates so closely to the ladder diagrams used to document hard-wired relay control systems.

[5] For those familiar with SPICE circuit simulation software, wire numbers in a control relay circuit are analogous to *node numbers* in a SPICE netlist: identical numbers denote electrical commonality.

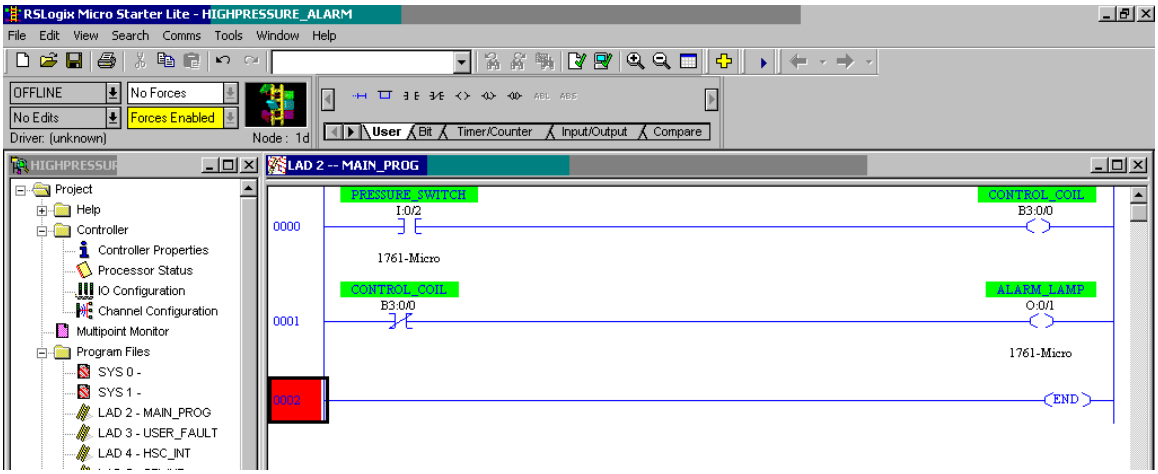
[6] Close inspection of this hand-edited diagram reveals a penciled diagonal line drawn across the PLR contact, opposite to the angle that is conventional. It is unclear whether this diagonal line represents a truly normally-closed contact or whether this is an example of someone's bad habit of inserting diagonal lines to represent the *temporary* state of the contact. The PLR relay is designed to open this contact in the event of a *past* loss of power, which means this relay "latches" open when de-energized and must be reset by some external means, most likely a manually-operated pushbutton. What is considered the "normal" or "resting" state of a latching relay is sometimes a matter of interpretation and debate, so more information would be necessary on the operation of the PLR relay before we could render a judgment as to whether the PLR contact symbol deserves to be drawn as NO or NC.

Here, we will emulate the exact same high-pressure alarm circuit seen in an earlier section using an Allen-Bradley MicroLogix 1000 PLC instead of a relay coil:

Wiring diagram:



Ladder-logic program:

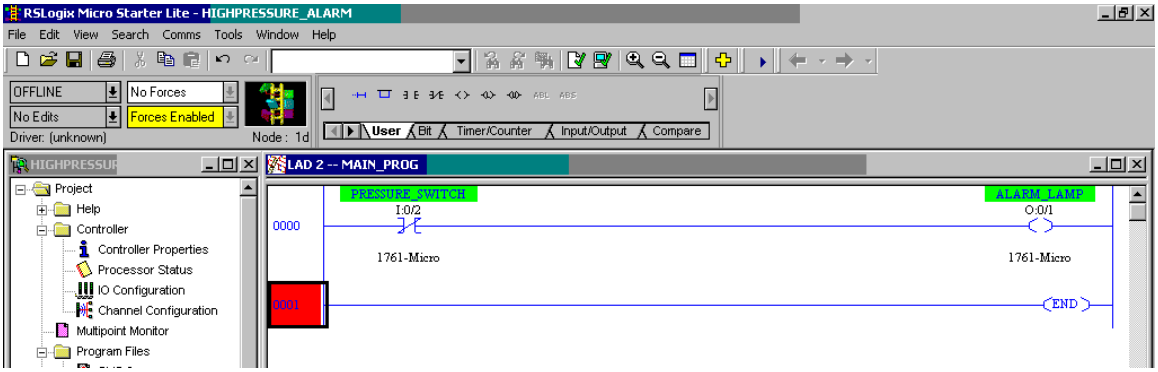


Suppose a fluid pressure of 36 PSI is applied to the pressure switch. This is less than the switch's trip setting of 50 PSI, leaving the switch in its "normal" (closed) state. This sends power to input `I:0/2` of the PLC. The contact labeled `I:0/2` drawn in the ladder-logic program of the PLC acts like a relay contact driven by a coil energized by input terminal `I:0/2`. Thus, the closed pressure switch

contact energizes input terminal I:0/2, which in turn "closes" the normally-open contact symbol I:0/2 drawn in the ladder-logic program. This "virtual" contact sends virtual power to a virtual coil labeled B3:0/0, which is nothing more than a single bit of data in the PLC's microprocessor memory. "Energizing" this virtual coil has the effect of "actuating" any contact drawn in the program bearing the same label. This means the normally-closed contact B3:0/0 will now be "actuated" and thus in the open state, not sending virtual power to the output coil O:0/1. With virtual coil O:0/1 "unpowered," the real-life output O:0/1 on the PLC will be electrically open, and the alarm lamp will be unpowered (off).

If we apply a fluid pressure of 61 PSI to the pressure switch, the normally-closed pressure switch contact will be actuated (forced) into the open state. This will have the effect of de-energizing PLC input I:0/2, thus "opening" the normally-open virtual contact in the PLC program bearing the same label. This "open" virtual contact interrupts virtual power to the virtual coil B3:0/0, causing the normally-closed virtual contact B3:0/0 to "close," sending virtual power to virtual coil O:0/1. When this virtual output coil "energizes," the real-life output channel of the PLC activates, sending real power to the alarm light to turn it on, signaling a high-pressure alarm condition.

We may simplify this PLC program further by eliminating the virtual control relay B3:0/0 and simply having input I:0/2 activate output O:0/1 through a "normally-closed" virtual contact:



The effect is the same: the PLC output O:0/1 will activate whenever input I:0/2 de-energizes (whenever the pressure switch is opened by a high pressure), turning on the alarm lamp in a high-pressure condition. In a low-pressure condition, the energized input I:0/2 forces the virtual normally-closed contact I:0/2 to open, thus de-energizing the PLC's output O:0/1 and turning the alarm lamp off.

Programmable Logic Controllers have not only greatly simplified the wiring of industrial logic controls by replacing multitudes of electromechanical relays with a microprocessor, but they have also added advanced capabilities such as counters, timers, sequencers, mathematical functions, communications, and of course the ability to easily modify the control logic through programming rather than by re-wiring relays. The beauty of ladder-logic programming is that it translates the technician's understanding of traditional relay control circuits into a virtual form where contacts and coils interact to perform practical control functions. A key concept to master, however, is the association of real-life conditions to switch status based on the "normal" representation of those

switch contacts, whether the switches be real (relay) or virtual (PLC). With this vital concept mastered, both hard-wired relay control circuits and PLC programs become possible to understand. Without mastering this vital concept, neither relay control circuits nor PLC programs make sense.

Chapter 4

Historical References

This chapter is where you will find references to historical texts and technologies related to the module's topic.

Readers may wonder why historical references might be included in any modern lesson on a subject. Why dwell on old ideas and obsolete technologies? One answer to this question is that the initial discoveries and early applications of scientific principles typically present those principles in forms that are unusually easy to grasp. Anyone who first discovers a new principle must necessarily do so from a perspective of ignorance (i.e. if you truly *discover* something yourself, it means you must have come to that discovery with no prior knowledge of it and no hints from others knowledgeable in it), and in so doing the discoverer lacks any hindsight or advantage that might have otherwise come from a more advanced perspective. Thus, discoverers are forced to think and express themselves in less-advanced terms, and this often makes their explanations more readily accessible to others who, like the discoverer, comes to this idea with no prior knowledge. Furthermore, early discoverers often faced the daunting challenge of explaining their new and complex ideas to a naturally skeptical scientific community, and this pressure incentivized clear and compelling communication. As James Clerk Maxwell eloquently stated in the Preface to his book *A Treatise on Electricity and Magnetism* written in 1873,

> It is of great advantage to the student of any subject to read the original memoirs on that subject, for science is always most completely assimilated when it is in its nascent state . . . `[page xi]`

Furthermore, grasping the historical context of technological discoveries is important for understanding how science intersects with culture and civilization, which is ever important because new discoveries and new applications of existing discoveries will always continue to impact our lives. One will often find themselves impressed by the ingenuity of previous generations, and by the high degree of refinement to which now-obsolete technologies were once raised. There is much to learn and much inspiration to be drawn from the technological past, and to the inquisitive mind these historical references are treasures waiting to be (re)-discovered.

4.1 New York City's relay-based subway signal system

A very interesting historical example of a control system based on electromechanical relays is one that is still in operation with portions of it *over seventy years old*[1]. This is the "interlocking" signal system employed by the New York City subway system, the purpose of which is to signal to human train operators when it is safe to move their train into a section of shared track. A report issued by the Regional Plan Association in 2014 describes certain aspects of this signaling system:

> Today, the New York City subway relies on a central nervous system made up of 15,000 signal blocks, 3,500 mainline switches and 339,000 signal relays. These components, which have hardly changed since the subway opened in 1904, let train operators know when it is safe for them to move trains forward. [page 5]

Another section of the RPA report describes how the subway tracks are divided into "blocks" of fixed length, and how the presence of a subway car on the track is electrically detected by a relay circuit:

> The subway operates using a conventional fixed-block regime, meaning that its tracks are divided up into segments or blocks that average 1,000 feet in length. An insulator is placed between the rails on both ends of the track segment to create a block. An electrical current is then run through the block to a relay creating an electrical circuit. As long as the circuit is closed, meaning that the current is able to travel unimpeded from one end of the circuit through the relay to the other, the block is deemed open and not occupied by a train. As soon as a train enters a block its steel wheels break or "short" the circuit causing the relay to discharge and the block to register as being occupied. The state of the blocks ahead dictates if or how fast a train may proceed along its route. An open circuit can also indicate a broken rail or a signal malfunction. [page 13]

The authors of this report were not necessarily writing for an audience of technicians or engineers, and so the terminology used to describe the electrical status of a track block is somewhat imprecise. However, it is clear that the metal wheels and axles of the subway cars act as electrical connections bridging one rail of an occupied "block" to the other rail, and that electrical connection diverts current that would otherwise pass through the coil of an electromechanical relay. The relay's contacts then cause colored signaling lamps to change state, much like the colored lamps of a traffic-control light in a city street.

According to an article written by Dan Rivoli for the New York Daily News on 22 May 2017, the age of certain system components is impressive. Control relays manufactured by the General Railway Signal Company (no longer in business) have labels with service dates from World War Two. Even many of the cables still used in this signaling system are old, dating back about 70 years. These cables use *cloth fabric* insulation rather than rubber or plastic, and are prone to catching fire. One upgrade strategy is to replace the older 120 Volt relays with 16 Volt relays, the reduced voltage posing less threat to the integrity of old cloth-insulated cables.

[1] Table 11 on 53 of RPA's "Moving Forward" report shows three different lines on the New York City subway system with relay-based interlocking signals that have not been completely renewed in over 70 years, the oldest estimated to be 78 years old. Twenty of the subway lines operate using relay-based controls at least 50 years old!

In another newspaper article written by Jeremy Smerd for the New York Sun in February 2005, the condition of rooms containing control relays for the subway signaling system was described in terms revealing how antiquated the system is. Smerd's article describes some of these rooms as lacking fire detection or suppression equipment, as well as lacking disconnect switches allowing firefighters arriving on the scene to shut off electrical power before taking further action. A fire that destroyed control relays at the Chambers Street subway station in January 2005 was apparently triggered by something burning inside a shopping cart that was abandoned adjacent to the relay room in a tunnel near the subway platform. Cloth-insulated cables passing from the relay room to the tunnel caught fire, and the flammable cloth continued to burn into the relay room itself where it destroyed the panel full of control relays.

Chapter 5

Derivations and Technical References

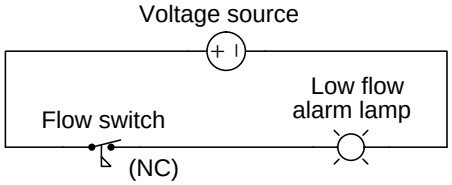
This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

5.1 Normal status of a switch contact

An perennial source of confusion among students new to electric switching circuits is the use of the word *normal* to refer to an electrical switch's default state. Switches, of course, are discrete devices capably only of two definite states: *open* (i.e. no electrical continuity) and *closed* (i.e. electrical continuity). *Toggle* switches are constructed in such a way that they may latch in either of these two states, which means they have no default condition. A great many switch types, however, are designed with a spring-return mechanism or equivalent functionality to make the switch return to a certain default state in the absence of any external stimulus. **This is called the "normal" state of the switch: its electrical state when at rest.**

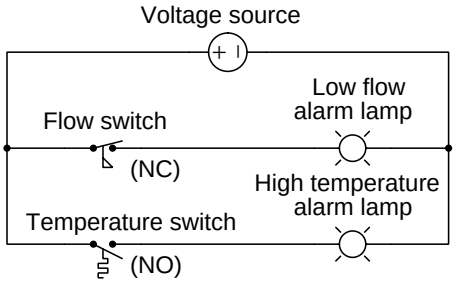
Where this becomes confusing is in applications where such a switch is *typically* found in an actuated state, such that ordinary operating conditions for the circuit maintain that switch in its *non-normal* state. Colloquial use of the word "normal" is synonymous with "typical" which makes it possible for someone to see a switch's "normal" status and mistakenly think this refers to its state in the circuit's normal operation rather than meaning its "normal" status as defined by its manufacturer.

A brief illustration is helpful here. Consider a *flow switch* used to detect the presence of liquid coolant flow through a pipe, carrying coolant to an operating engine. Engines, of course, dissipate heat as they run, and so a continuous flow of coolant to the engine is critical for maintaining safe operating temperature. A simple diagram shows how this flow switch would be connected to a warning lamp to alert personnel of any interruption in coolant flow to the engine:



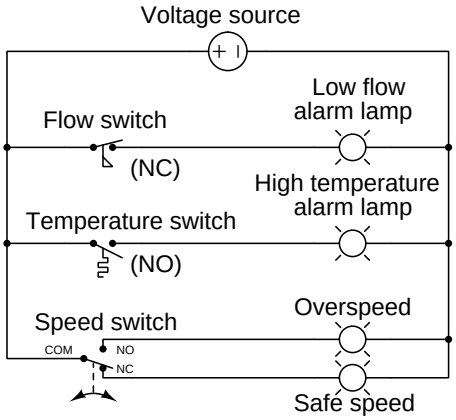
Since the purpose of this circuit is to energize the warning lamp in the event of *no coolant flow*, the flow switch's spring-return mechanism must be configured in such a way to *close* the switch contact in the absence of flow. In other words, this flow switch's contact will be closed when at rest – i.e. it will be a *normally-closed* flow switch. However, during typical operation when adequate coolant flow is present in the pipe, this switch will be held in its open state and the alarm lamp will be de-energized. Even though the flow switch is *normally-closed* (NC), in this application it will be *typically open* – the "normal" and "typical" states for this switch in this application are opposite.

Now consider the addition of a different type of switch and alarm lamp to the circuit, with the new switch installed on the same heat-dissipating engine serving to warn personnel if the engine becomes too hot:



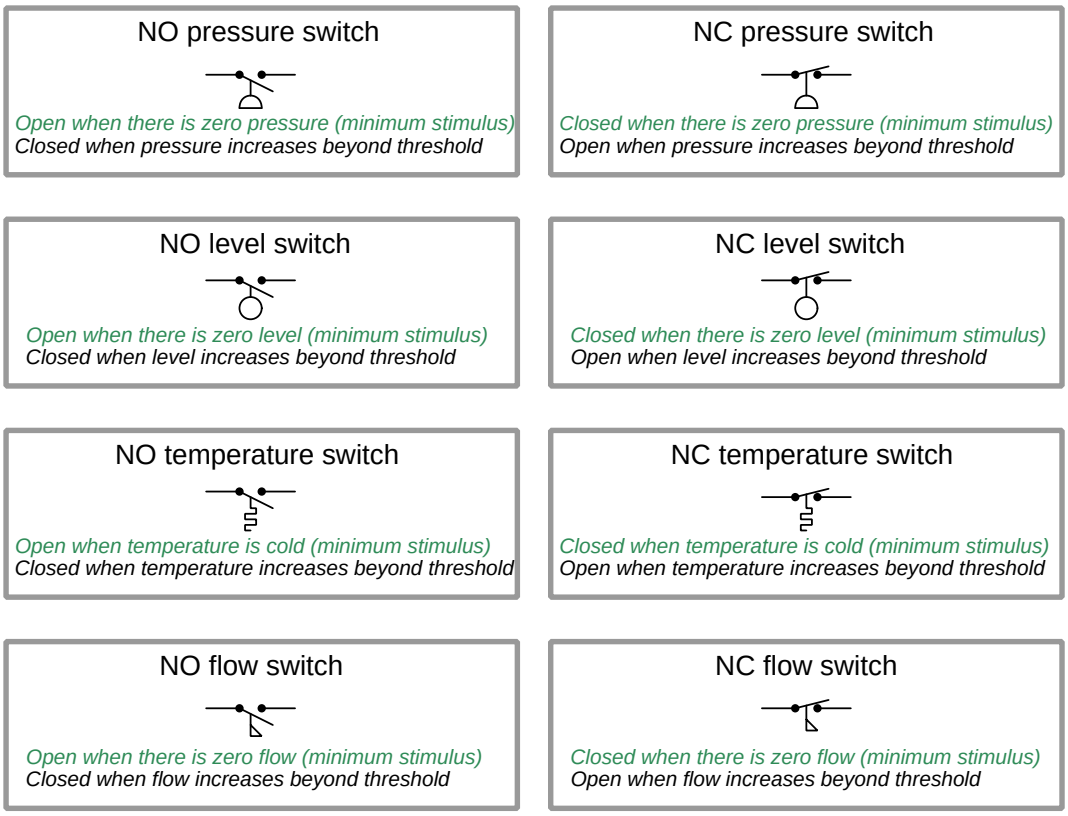
This new switch's purpose is to energize its warning lamp in the event the engine overheats, and its mechanism must be configured to *close* the switch in the presence of high temperature. This means the temperature switch's spring-return will force it open at rest, making it a *normally-open* temperature switch. During typical operation when the engine's temperature is within reasonable bounds, this switch will still be in its resting state, and so this *normally-open* (NO) temperature switch will also be *typically open* – a case where "normal" and "typical" states happen to be identical.

Let us consider one more switch application for this hypothetical engine, this time using a *single-pole, double-throw* (*SPDT*) speed switch to monitor the engine's shaft speed and trigger energization of two indicator lamps, one for "safe speed" and another for "overspeed":



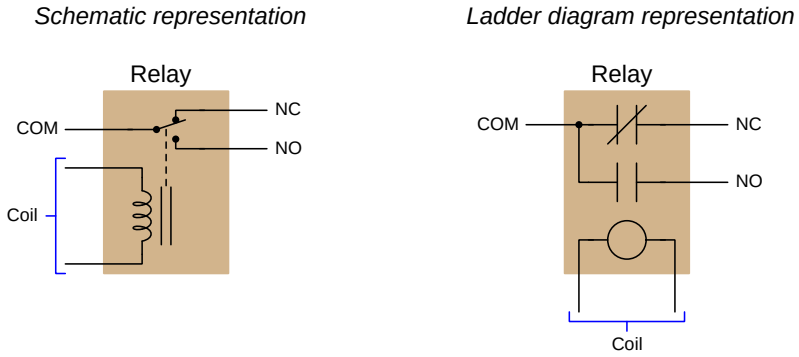
Note the COM, NO, and NC labeling of this switch's three terminals, denoting "Common", "Normally-Open", and "Normally-Closed", respectively. As with the other two switches, these contact labels as well as the switch symbol itself as drawn in the diagram represent the switch's state *when at rest*. This is strict convention in electrical switching circuits: the "normal" state of any switch is defined by a condition of minimal stimulus, and this is always how it is drawn.

A helpful tip to remember about sensing switches and their respective symbols is that the symbols are conventionally drawn in such a way that an *upward* motion of the movable switch element represents *increasing stimulus*. Here are some examples of this, showing various switch types and NO/NC contact configurations, comparing their states with no stimulus versus when the stimulus exceeds the each switch's threshold or "trip" setting. The *normal* status of each switch as defined by the manufacturer is labeled in green text:



Interestingly, the convention of upward motion representing the direction of stimulus is not maintained for hand-operated switches.

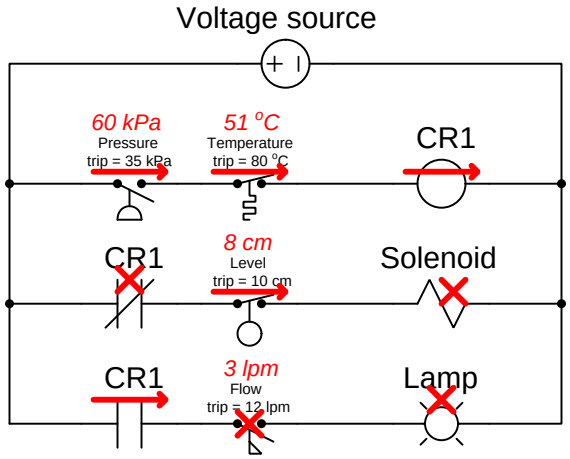
Switch contacts within electromechanical relays are also characterized as being either normally-open (NO) or normally-closed (NC), and in this case the stimulus in question is the energization of the relay's electromagnet coil. When the coil is de-energized, the contacts will all be in their resting (i.e. "normal") states which is also how the relay's contacts are drawn in diagrams. When the coil is energized, though, all contacts within the relay flip to their opposite states: all NO contacts *close* and all NC contacts *open*. The specific symbols used to represent relay coils and contacts differ according to the type of diagram, but their meaning is the same:



A normally-closed (NC) relay contact is one which will be in its closed state when the coil is de-energized, represented in diagram form by touching lines or by a slash mark between the two contact plates. A normally-open (NO) relay contact is one which will be in its open state when the coil is de-energized, represented in diagram form by an air gap between the contacting surfaces. Upon energization of the relay coil, all the contacts within that relay change state, *but their written symbols remain the same*[1] in order to represent their *resting* states.

[1] A bad habit some people adopt is to draw a slash mark through a relay contact symbol in order to annotate that relay contact's *closure* when analyzing the diagram for a relay-based circuit. This habit should avoided, as the symbols used to represent *normal* status should never be used to represent *present* status. There is enough confusion as it is surrounding the term "normal" without any more being added, so please do not contribute to the chaos!

When analyzing electrical switching circuits, a helpful problem-solving strategy is to annotate the diagram with symbols denoting the *actual* status of each switch contact in any given circuit condition, and not the *normal* status. Such annotations make it easier to determine which loads in a circuit will be energized, and which will not, for any given circuit condition. For this I recommend sketching an arrow or a line nearby a contact to show a closed state, and an "X" nearby a contact to show an open state. These annotations demonstrate real contact status without obscuring normal status. Consider these annotations used in the following example diagram:



In the upper "rung" of this ladder-style diagram we see the normally-open pressure switch is actuated (i.e. closed) because the applied pressure of 60 kPa exceeds the switch's trip setting of 35 kPa. The normally-closed temperature switch is unactuated (i.e. closed) because the applied temperature of 51 ^{o}C is less than the trip threshold of 80 ^{o}C. The red arrows annotating both switches show their closed statuses. Wired in series, these two closed switch contacts permit energizing current to the coil of relay CR1, and so another red arrow drawn there indicates that coil's energized status.

In the second and third rungs we see the present status of each CR1 contact. Since the CR1 relay coil is energized it places each CR1 relay contact into a state opposite of its "resting" or "normal" condition, therefore the normally-closed CR1 contact in rung 2 is open (shown with a red "X" annotation) and the normally-open CR1 contact in rung 3 is closed (shown with a red arrow annotation). The level switch's stimulus is less than its trip setting, and so that normally-closed contact remains closed and gets a red arrow. The flow switch's stimulus is also less than its trip setting, and so that normally-open switch remains open and gets a red "X" annotation. Neither rung 2 nor rung 3 is completed because one of the series-connected contacts in each rung is open thus preventing energization of its load. Therefore, both the solenoid coil and the lamp are de-energized, shown with red "X" annotations.

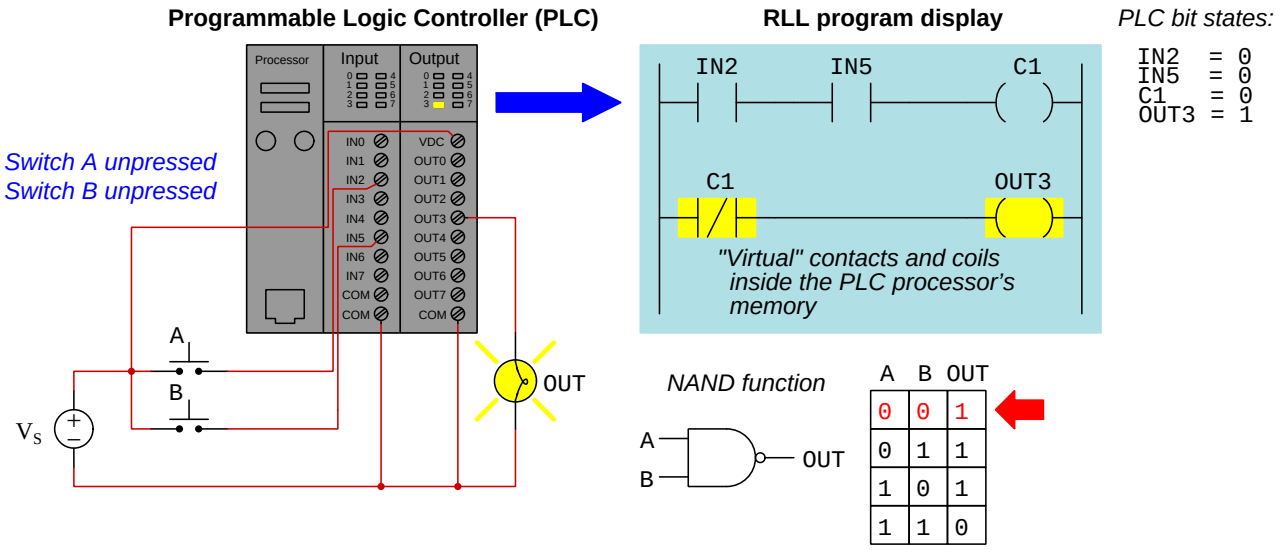
5.2 Ladder Logic Programming of PLCs

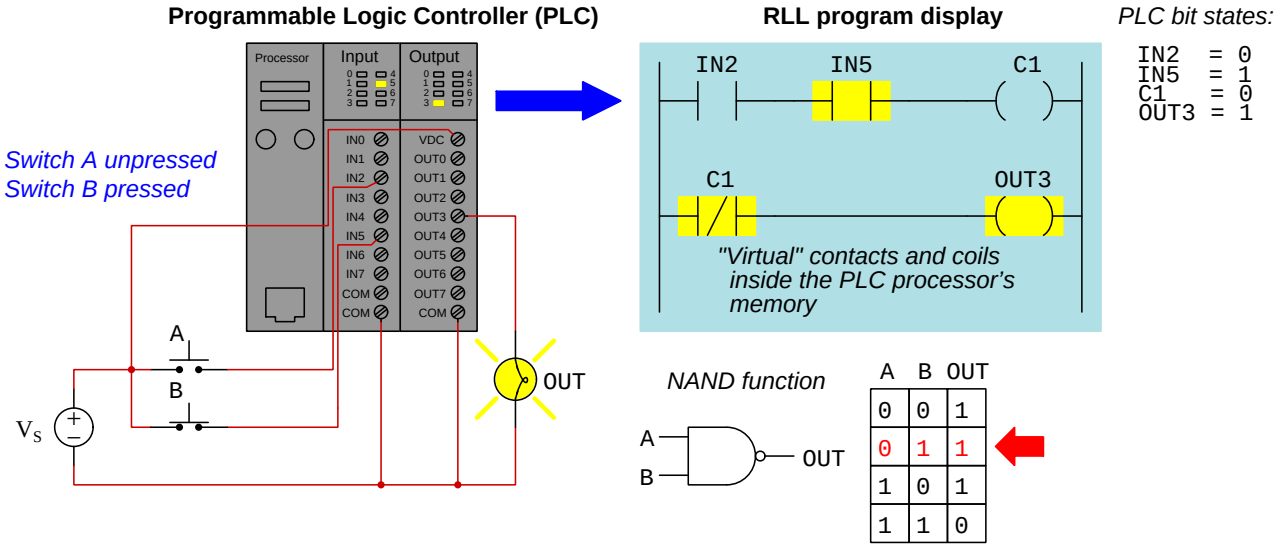
Ladder diagrams are such useful conventions for expressing discrete logic functions that they are even used as a form of *programming language* for computer-based industrial control devices called *Programmable Logic Controllers*, or *PLCs*. Programmable logic controllers were developed to replace complex electromechanical relay logic circuits with something faster, more reliable, and more easily modified. The basic idea of a PLC is that it receives discrete electrical signal inputs from switches, processes those input states according to a logic function drawn as a ladder diagram[2], and then drive output switching elements to send electricity to loads such as lamps, solenoids, and relay coils.

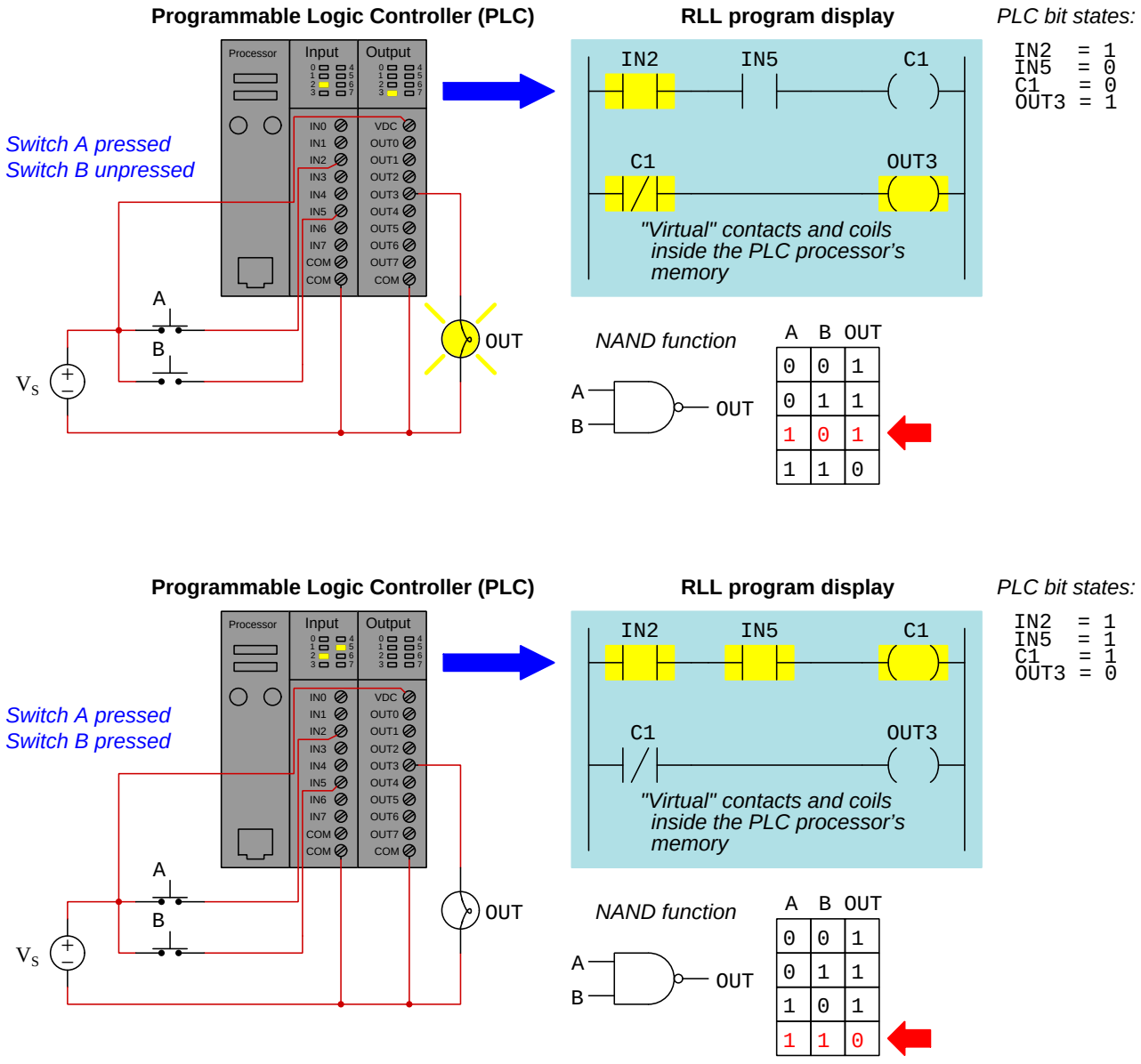
In order to demonstrate basic PLC functionality, I will present several illustrations of a PLC connected to a pair of pushbutton switches, the PLC programmed using *Relay Ladder Logic* (RLL)[3]. The program is shown on a blue screen, as viewed from a personal computer connected to the PLC's processor via a communication cable. Like practically all PLCs, this one reveals the current status of its virtual "contacts" and "coils" by means of color highlighting on the personal computer screen, a colored contact or coil having a "true" state and an uncolored contact or coil having a "false" state. The particular program I will show in these examples implements a simple NAND logic function, and each illustration will show this NAND function in a different combinational state.

Close examination of each illustration is strongly recommended. Compare the truth table states against the LED indicators on the PLC's input and output modules, and also with the virtual "contacts" and "coils" visible within the RLL program screen.

[2] As though the PLC contained an array of easily-reconfigured relays inside.
[3] Sometimes called *Ladder Diagram*, or LD.


Programmable Logic Controller (PLC)
RLL program display
PLC bit states:
Processor
Input
Output
IN0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
COM
COM
VDC
OUT0
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
COM
Switch A unpressed
Switch B unpressed
A
B
V_S
OUT
IN2
IN5
C1
C1
OUT3
"Virtual" contacts and coils inside the PLC processor's memory
IN2 = 0
IN5 = 0
C1 = 0
OUT3 = 1
NAND function
A
B
OUT
A B OUT
0 0 1
0 1 1
1 0 1
1 1 0


Programmable Logic Controller (PLC)
RLL program display
PLC bit states:
Processor
Input
Output
IN0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
COM
COM
VDC
OUT0
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
COM
Switch A unpressed
Switch B pressed
A
B
V_S
OUT
IN2
IN5
C1
C1
OUT3
"Virtual" contacts and coils inside the PLC processor's memory
IN2 = 0
IN5 = 1
C1 = 0
OUT3 = 1
NAND function
A
B
OUT
A B OUT
0 0 1
0 1 1
1 0 1
1 1 0



"Ladder-logic" programming is limited compared to text-based computer programming languages, but it is far more intuitive and resembles the diagrams of relay control systems PLCs were designed to replace. Some modern PLCs support text-based and function-block-based programming languages in addition to RLL, in order to give the end-user an array of programming options.

Chapter 6

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

6.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

6.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

6.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Discrete signal

Logic function

Truth table

OR function

AND function

NOT function

NOR function

NAND function

Parallel contact logic

Series contact logic

Normally-closed contact logic

Normal state of a switch

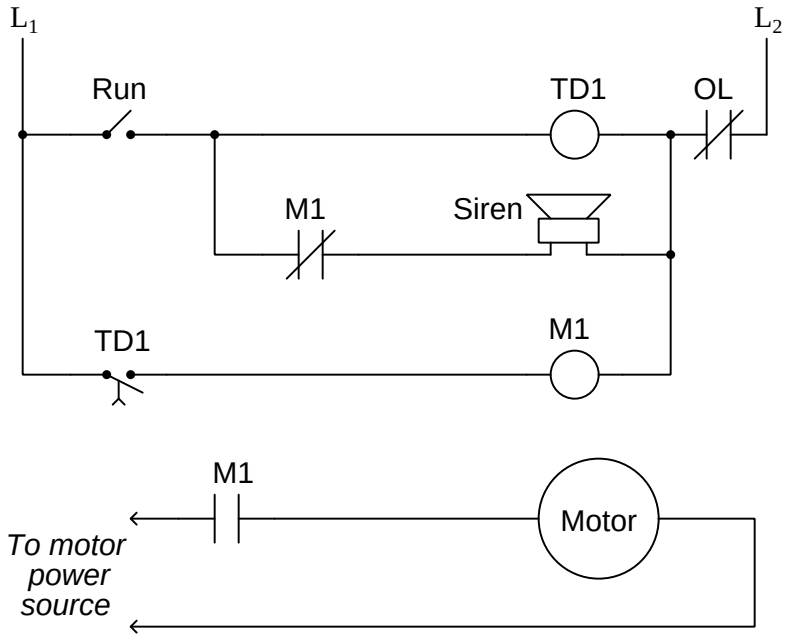
Relay ladder diagram

Annotating relay states

Wire numbering

6.1.3 Conveyor warning siren

An electric motor is used to power a large conveyor belt. Before the motor actually starts, a warning siren activates to alert workers of the conveyor's forthcoming action. The following relay circuit accomplishes both tasks (motor control plus siren alert). M1 is a "motor contactor" which is nothing more than a large relay with high-current-rated contacts (not shown here) used to make and break power to the electric motor moving the conveyor belt:



Study this ladder logic diagram, then explain how the system works.

Challenges

- Supposing the electric motor was powered by the same line connections (L1 and L2) as this relay control circuit, where would it be inserted into this diagram.
- Suppose contactor M1 only had one electrical switch contact, and that was the one for motor power. In the absence of a normally-closed "auxiliary" contact used here to control the siren, how could this same warning siren be controlled? What else would you have to add to this circuit, and where exactly would it go?

6.1.4 Active reading exercise: motor control circuit diagram

A good habit when reading technical documents is to read them *actively*, which means among other things analyzing the information presented to you rather than just trying to absorb it. An application of this is the image of a motor control ladder diagram in the Tutorial, where someone had made "red-line" edits to the diagram after installing a Power Loss Relay (PLR). A reader could merely examine the diagram, read the accompanying text, and move on, but they will learn a lot more if they try to analyze that diagram to see how the motor control system is supposed to function.

Examine this diagram closely, and then answer the following questions about it:

- Identify the relay coil which needs to be energized in order to make the three-phase AC motor spin in the "forward" direction.
- Identify the relay coil which needs to be energized in order to make the three-phase AC motor spin in the "reverse" direction.
- Explain how pressing the Emergency Stop ("E-Stop") latching pushbutton forces the motor to stop.
- Pressing the momentary-contact Start pushbutton causes the motor to run and remain running even after that Start pushbutton is released. How does this circuit achieve a "latching" function to keep the motor running even after the Start pushbutton switch returns to its normal (open) state?
- Where did new wire numbers have to be inserted into this circuit, and why were new numbers necessary?
- Proximity switch PX-1 is an example of a *permissive* contact, because it needs to sense the presence of a machine part before permitting the motor to run in either direction. Note the unusual way in which PX-1 is wired into the circuit, which is unlike how any of the other switches (e.g. PS1, FS1. E-Stop, etc.) are wired. Explain why this is.

Challenges

- Identify the error in this diagram regarding PE1.
- What purpose do you suppose the Power Loss Relay serves?

6.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

6.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

6.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

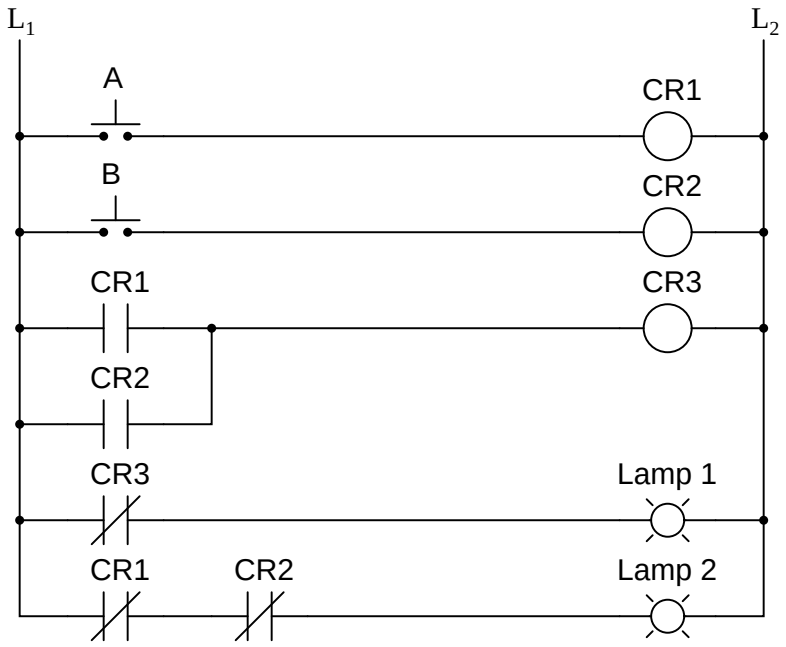
Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

6.2.3 Associating logic functions with relay circuit

Write a truth table for each of the indicator lamps in the following ladder diagram, and determine which logic function (AND, OR, NAND, NOR, or NOT) best describes each lamp's behavior with respect to the status of the input switches.

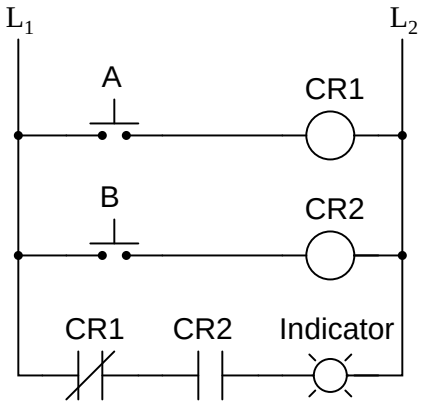


Challenges

- Without moving any wire connections, but only substituting components, identify how to create an AND function in this relay circuit.

6.2.4 Truth table for a relay circuit

Analyze the following relay logic circuit, completing the truth table accordingly:



Truth table

A	B	Output
0	0	
0	1	
1	0	
1	1	

Also, demonstrate how to apply the annotation techniques described in the Tutorial to this circuit as you determine what its output will be for each input condition.

Challenges

- Sketch a logic function symbol representing this relay circuit's truth table.
- Modify this circuit to perform the same logical function (i.e. manifest the same truth table) but without using a relay.
- Modify this circuit to form an AND function.
- Modify this circuit to form a OR function.
- Modify this circuit to form a NAND function.
- Modify this circuit to form a NOR function.

6.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

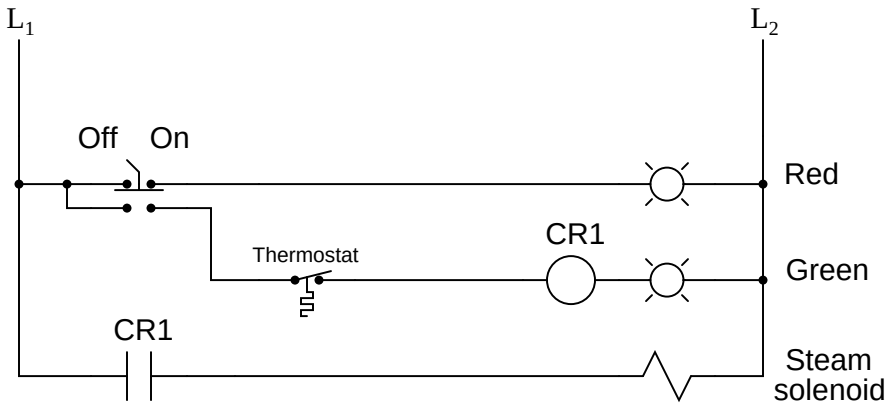
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

6.3.1 Mistaken wiring in a steam control circuit

The following ladder logic diagram (for a steam heater control) contains a serious mistake. A thermostat senses the temperature of a steam-heated machine, causing an electrically-actuated "solenoid" valve to pass steam to the machine if ever it becomes too cool, and shutting off that solenoid valve when the temperature is high enough:



This mistake is very common among students new to relay ladder-logic diagrams. Explain what the mistake is, and draw a corrected version of this relay circuit.

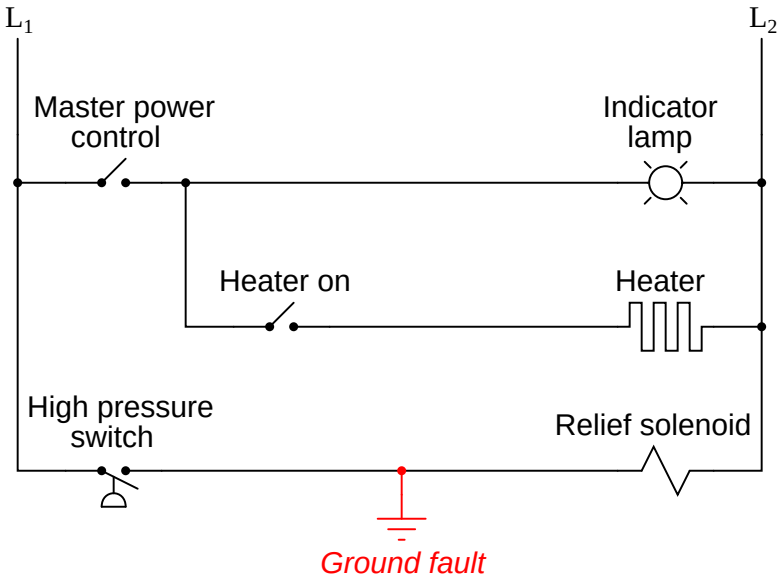
Challenges

- Explain what this system *would* do if wired as shown.
- Explain how this thermostatically-controlled system is supposed to operate.
- Explain the operation of the on/off switch: how to interpret the symbol within the diagram.

6.3.2 Effects of a ground fault in a relay control circuit

Safety is a paramount concern in electrical systems. Generally, we try to design electrical circuits so that if and when they fail, they will do so in the manner safest to those people working around them, and to the equipment and process(es) controlled by the circuit.

One of the more common failure modes of circuits having wires strung through metal conduit is the *accidental ground* – otherwise known as a *ground fault* – where the electrical insulation surrounding a wire fails, resulting in contact between that wire and a grounded metal surface. Suppose a ground fault were to occur at the point shown in this ladder diagram:



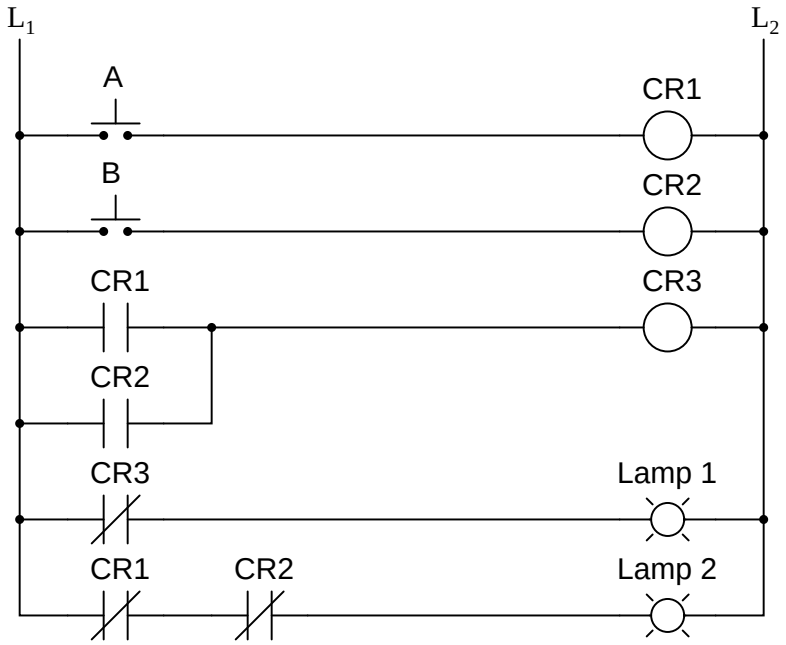
What would be the result of this ground fault? Would it matter if the L1/L2 power connections were reversed?

Challenges

- Complete this ladder diagram by adding an AC power source to it.
- ???.

6.3.3 Identifying possible faults in a relay circuit

Suppose Lamp 1 in this circuit remained energized regardless of the states of switches A and B:



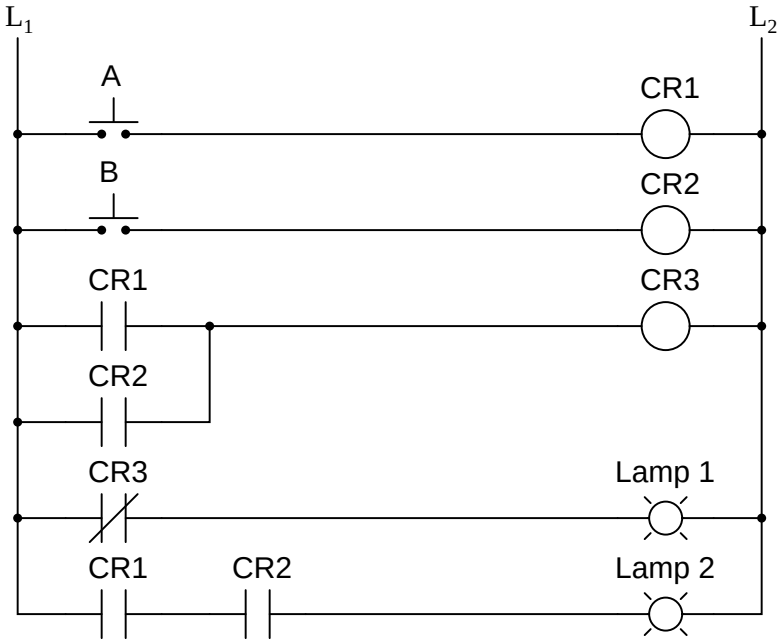
Identify at least two different faults that could cause this to happen.

Challenges

- Identify diagnostic tests useful in differentiating between the possible faults you identified.

6.3.4 Diagnosing relay circuit fault

There is a problem somewhere in this relay logic circuit. Lamp 2 operates exactly as it should, but lamp 1 never turns on. Identify all possible failures in the circuit that could cause this problem, and then explain how you would troubleshoot the problem as efficiently as possible (taking the least amount of electrical measurements to identify the specific problem).

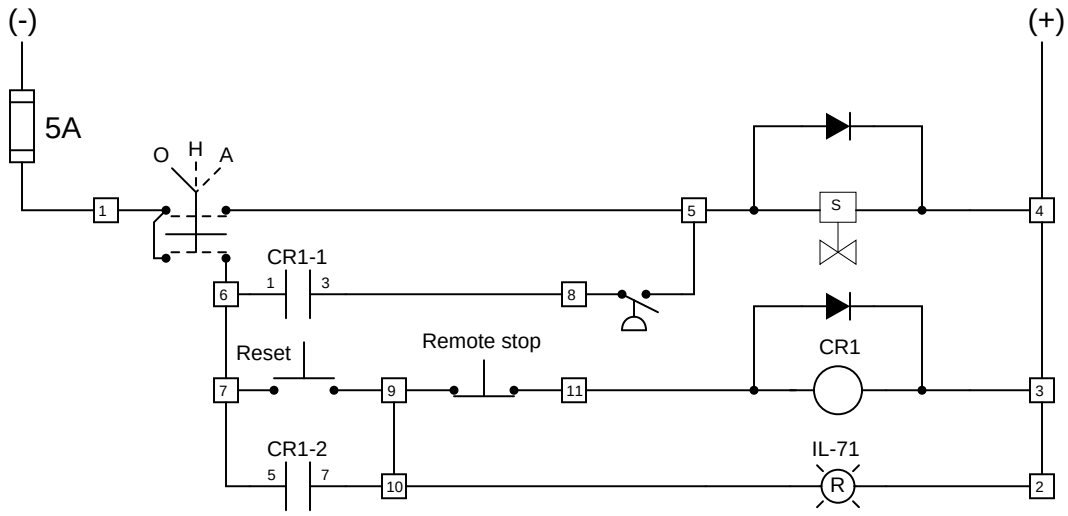


Challenges

- Identify the logical functions represented in this ladder logic circuit.

6.3.5 Faulted pressure-controlled solenoid valve

Examine this ladder diagram for a solenoid valve control circuit, where the status of a solenoid valve (either on or off) is controlled by hand switches and a pressure switch.



Suppose the solenoid valve refuses to energize when the Hand/Off/Auto switch has been placed in "Auto," the Reset pushbutton pressed, and a high-pressure condition exists. The solenoid can, however, be made to energize by placing the switch in the "Hand" position.

Beginning your troubleshooting steps, you first note that the red indicator light never comes on. Identify at least three possible faults that could (each one, individually) account for these symptoms. Also, identify at least three components you know to be fully functional in this circuit.

Possible faults:

-
-
-
-
-
-
-

Three known-good components:

-
-
-

Challenges

- Explain the purpose of the diodes in this ladder-logic circuit.

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits.

The following principles inform the instructional and assessment philosophies embodied in these learning modules:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[1] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[2] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[3] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[1] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[2] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[3] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another characteristic of these learning modules is a lack of step-by-step instructions in the Project and Experiment chapters. Unlike many modern workbooks and laboratory guides where step-by-step instructions are prescribed for each experiment, these modules take the approach that students must learn to closely read the tutorials and apply their own reasoning to identify the appropriate experimental steps. Sometimes these steps are plainly declared in the text, just not as a set of enumerated points. At other times certain steps are implied, an example being assumed competence in test equipment use where the student should not need to be told *again* how to use their multimeter because that was thoroughly explained in previous lessons. In some circumstances no steps are given at all, leaving the entire procedure up to the student.

This lack of prescription is not a flaw, but rather a feature. Close reading and clear thinking are foundational principles of this learning series, and in keeping with this philosophy all activities are designed to *require* those behaviors. Some students may find the lack of prescription frustrating, because it demands more from them than what their previous educational experiences required. This frustration should be interpreted as an unfamiliarity with autonomous thinking, a problem which must be corrected if the student is ever to become a self-directed learner and effective problem-solver. Ultimately, the need for students to read closely and think clearly is more important both in the near-term and far-term than any specific facet of the subject matter at hand. If a student takes longer than expected to complete a module because they are forced to outline, digest, and reason on their own, so be it. The future gains enjoyed by developing this mental discipline will be well worth the additional effort and delay.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

This methodology of assigning text-based modules to students for digestion and then using Socratic dialogue to assess progress and hone students' thinking was developed over a period of several years by the author with his Electronics and Instrumentation students at the two-year college level. While decidedly unconventional and sometimes even unsettling for students accustomed to a more passive lecture environment, this instructional philosophy has proven its ability to convey conceptual mastery, foster careful analysis, and enhance employability so much better than lecture that the author refuses to ever teach by lecture again.

Problems which often go undiagnosed in a lecture environment are laid bare in this "inverted" format where students must articulate and logically defend their reasoning. This, too, may be unsettling for students accustomed to lecture sessions where the instructor cannot tell for sure who comprehends and who does not, and this vulnerability necessitates sensitivity on the part of the "inverted" session instructor in order that students never feel discouraged by having their errors exposed. *Everyone* makes mistakes from time to time, and learning is a lifelong process! Part of the instructor's job is to build a culture of learning among the students where errors are not seen as shameful, but rather as opportunities for progress.

To this end, instructors managing courses based on these modules should adhere to the following principles:

- Student questions are always welcome and demand thorough, honest answers. The only type of question an instructor should refuse to answer is one the student should be able to easily answer on their own. Remember, *the fundamental goal of education is for each student to learn to think clearly and independently.* This requires hard work on the part of the student, which no instructor should ever circumvent. Anything done to bypass the student's responsibility to do that hard work ultimately limits that student's potential and thereby does real harm.

- It is not only permissible, but encouraged, to answer a student's question by asking questions in return, these follow-up questions designed to guide the student to reach a correct answer through their own reasoning.

- All student answers demand to be challenged by the instructor and/or by other students. This includes both correct and incorrect answers – the goal is to practice the articulation and defense of one's own reasoning.

- No reading assignment is deemed complete unless and until the student demonstrates their ability to accurately summarize the major points in their own terms. Recitation of the original text is unacceptable. This is why every module contains an "Outline and reflections" question as well as a "Foundational concepts" question in the Conceptual reasoning section, to prompt reflective reading.

- No assigned question is deemed answered unless and until the student demonstrates their ability to consistently and correctly apply the concepts to *variations* of that question. This is why module questions typically contain multiple "Challenges" suggesting different applications of the concept(s) as well as variations on the same theme(s). Instructors are encouraged to devise as many of their own "Challenges" as they are able, in order to have a multitude of ways ready to probe students' understanding.

- No assigned experiment or project is deemed complete unless and until the student demonstrates the task in action. If this cannot be done "live" before the instructor, video-recordings showing the demonstration are acceptable. All relevant safety precautions must be followed, all test equipment must be used correctly, and the student must be able to properly explain all results. The student must also successfully answer all Challenges presented by the instructor for that experiment or project.

Students learning from these modules would do well to abide by the following principles:

- No text should be considered fully and adequately read unless and until you can express every idea *in your own words, using your own examples.*
- You should always articulate your thoughts as you read the text, noting points of agreement, confusion, and epiphanies. Feel free to print the text on paper and then write your notes in the margins. Alternatively, keep a journal for your own reflections as you read. This is truly a helpful tool when digesting complicated concepts.
- Never take the easy path of highlighting or underlining important text. Instead, *summarize* and/or *comment* on the text using your own words. This actively engages your mind, allowing you to more clearly perceive points of confusion or misunderstanding on your own.
- A very helpful strategy when learning new concepts is to place yourself in the role of a teacher, if only as a mental exercise. Either explain what you have recently learned to someone else, or at least *imagine* yourself explaining what you have learned to someone else. The simple act of having to articulate new knowledge and skill forces you to take on a different perspective, and will help reveal weaknesses in your understanding.
- Perform each and every mathematical calculation and thought experiment shown in the text on your own, referring back to the text to see that your results agree. This may seem trivial and unnecessary, but it is critically important to ensuring you actually understand what is presented, especially when the concepts at hand are complicated and easy to misunderstand. Apply this same strategy to become proficient in the use of *circuit simulation software*, checking to see if your simulated results agree with the results shown in the text.
- Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[4] effort, and never give up!

Students interested in using these modules for self-study will also find them beneficial, although the onus of responsibility for thoroughly reading and answering questions will of course lie with that individual alone. If a qualified instructor is not available to challenge students, a workable alternative is for students to form study groups where they challenge[5] one another.

To high standards of education,

Tony R. Kuphaldt

[4] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

[5] Avoid the temptation to simply share answers with study partners, as this is really counter-productive to learning. Always bear in mind that the answer to any question is far less important in the long run than the method(s) used to obtain that answer. The goal of education is to empower one's life through the improvement of clear and independent thought, literacy, expression, and various practical skills.

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

"Moving Forward – Accelerating the Transition to Communications-Based Train Control for New York City's Subways", Regional Plan Association, New York, May 2014.

Rivoli, Dan, "NYC subway relies on decades-old, outmoded signals, switches and track equipment", New York Daily News, 22 May 2017.

Smerd, Jeremy, "MTA Assailed for Not Protecting Relay Rooms", New York Sun, 4 February 2005.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

18 April 2025 – expanded upon some instructor notes.

15 July 2024 – divided the Introduction chapter into two sections, one for students and one for instructors, and added content to the instructor section recommending learning outcomes and measures.

10 August 2023 – added a Case Tutorial chapter containing an example of simple logic circuits built using nothing but switches and LEDs.

28 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

13 July 2022 – moved text and images from the Historical References section on New York city's legacy subway control system into its own file for inclusion in other modules.

25 May 2022 – changed descriptions in the Tutorial regarding the explicit showing of DC power supplies for logic gate and relay ladder diagrams, to reflect changes made to image_1266 and image_1267.

11 May 2022 – added more sections to the Tutorial, borrowing content from my *Lessons In Industrial Instrumentation* textbook.

21 April 2022 – minor edits to the Tutorial, as well as an addition to the "Truth table for a relay circuit" Quantitative Reasoning question. Also added a Challenge question regarding the typographical error in the motor control diagram that was photographed for the Tutorial. This diagram wasn't my doing, so I don't feel bad about this error, but I still want to challenge students to identify why the error doesn't make sense!

29 November 2021 – divided Tutorial into sections.

9 May 2021 – commented out or deleted empty chapters.

19 November 2020 – added Challenge questions.

29 September 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions. Also replaced "trial-and-error" with "cut-and-try" as the name for the problem-solving strategy applied within the Tutorial.

15 April 2020 – commented on the PLR contact as shown in the example ladder-logic diagram.

23 March 2020 – added a Diagnostic Reasoning question.

16 April 2019 – added a Conceptual Reasoning question.

26 March 2019 – added questions and added reference to L1/L2 line connections to the Tutorial.

13 March 2019 – added an experiment, constructing a basic two-input logic function using toggle switches and at least one relay.

10 March 2019 – completed Foundational Concepts list.

12 January 2019 – wrote Tutorial chapter, and added a section in the Historical References chapter about the antiquated relay-based interlocking controls of the New York subway system.

28 December 2018 – added "Normal status of a switch" section to the Derivations and Technical References chapter.

24 December 2018 – document first created.

Index