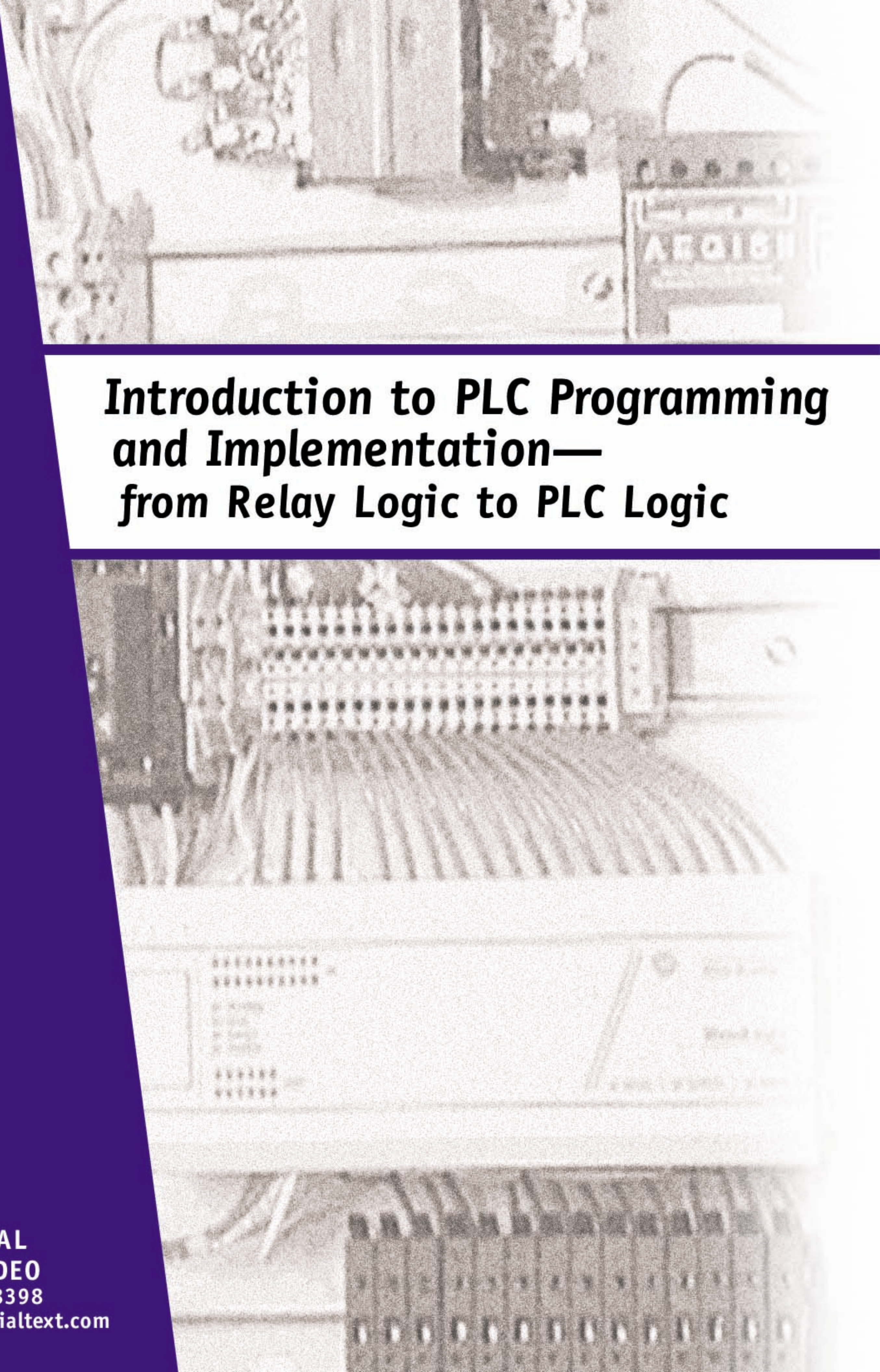


Introduction to PLC Programming and Implementation—from Relay Logic to PLC Logic

INDUSTRIAL
TEXT & VIDEO
1-800-752-8398
www.industrialtext.com

A Special Note To Our Customers

*Here's a valuable PLC reference that you can use right now. This particular reference is taken from our award-winning textbook—**Programmable Controllers: Theory and Implementation, 2nd Edition**.*

In it, you'll get an overview of how relay logic can be converted into PLC logic. There's also lots of examples, tables, and ladder diagrams to help explain the topics.

Best yet, we've included the corresponding chapter from the companion workbook. Here you can look over the key points as well as see how much you learned by answering the review questions. And, yes, the answers are also included.

This PLC reference is just a sample of what the textbook and workbook have to offer. If you like it, we've included the product literature page with the order number.

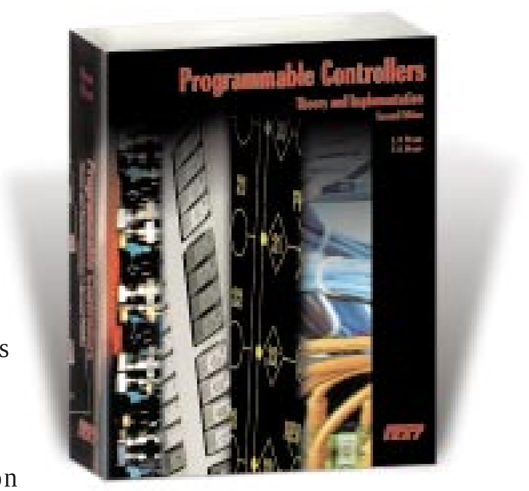
Industrial Text & Video Company

1-800-752-8398

www.industrialtext.com

PLC Reference Book

"You covered a huge amount of detail very well. It was very easy to understand."
—Jeff Camp, United Control Corp.



The biggest book on PLCs. Written by industry experts, this book covers important, up-to-date, real-world programmable controller topics and applications. This new edition is completely revised and updated to give you the latest developments and insights from the field. At 5 pounds and 1,035 pages, it puts all the PLC information you need at your fingertips. And, since this is a generic PLC reference, it will help you with all of the different makes and models of PLCs in your facility.

But, this book is about more than just PLCs—it also thoroughly explains process control, instrumentation, and plant networks. Whether you're already an expert on PLCs or just starting out, our problem-solving approach is guaranteed to help you succeed.

Catalog# ABT-ITV206BOOK **$88**

21 Chapters of PLC Know-How

TABLE OF CONTENTS

• Valuable Maintenance Tips •

SELECTION, INSTALLATION & SAFETY

✔ Follow our 11 major steps in selecting a PLC for an application and avoid using the wrong controller
✔ Install sinking and sourcing inputs and outputs properly—one wrong wire and it won't work
✔ Implement safety circuits correctly in PLC applications to protect people and equipment
✔ Prevent noise, heat, and voltage variations from ruining your PLC system
✔ Implement a step-by-step static and dynamic start-up checkout to guarantee smooth PLC system operation
✔ Design preventive safety and maintenance into your total control system

TROUBLESHOOTING & MAINTENANCE

✔ Learn no-nonsense troubleshooting procedures to reduce downtime
✔ Troubleshoot analog I/O and avoid undesirable count jumps
✔ Learn 6 preventive maintenance procedures to keep your PLC system running fault free
✔ Learn a step-by-step procedure for finding hidden ground loops
✔ Learn how to deal with leaky inputs
✔ Identify vibration problems and use them for preventive engineering control
✔ Control excessive line voltage and avoid intermittent shutdowns

PROGRAMMING

✔ Learn the number systems and codes used in PLC addressing
✔ Eliminate the confusion of ladder logic programming
✔ Master all types of timers and counters used in real-life applications
✔ Avoid ladder scan evaluation problems
✔ Implement a safe circuit with hardware and software interlocking

Programmable Controllers: Workbook/Study Guide

"Sometimes you think you know it all, but after reading the questions, I often times had to refer back to the theory book."
—Ernest Presto, Electrical Engineer, Polyclad Laminates, Inc.

Imagine having the answers to over 800 PLC problems at your fingertips. That's what you get with *Programmable Controllers: Workbook and Study Guide*. At 334 pages, it's the perfect companion to *Programmable Controllers: Theory and Implementation*, 2nd Edition.

This workbook provides not only valuable summaries of each of the textbook's twenty-one chapters, but also over 800 review questions. And each of the review questions includes a detailed answer and explanation. Use it on the job to brush up on the essentials and to solve any PLC problem.

Whether you're an expert or just learning about PLCs, you'll find plenty to put your skills to the test.

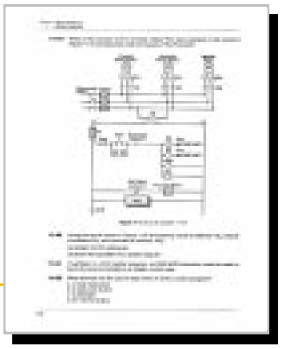
You Will Learn:

Catalog #ABT-ITV206WKBK **$28**

- Proper address assignment and interfacing
- Basic PLC ladder program implementation
- Data measurement
- Internal coil assignments
- Proper digital and analog interfacing procedures
- Advanced function block programming
- Network protocols
- Analog input and output data handling
- Correct PLC installation

Perfect textbook companion:

- 800 answers to common PLC problems at your fingertips
- Makes a great review tool
- Practice PLC addressing and programming
- Great on-the-job quick-reference guide
- Separate answer section makes quizzing easy
- Valuable chapter summaries

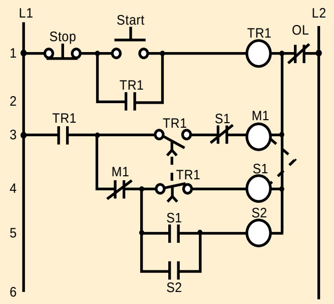


Sample pages from the workbook

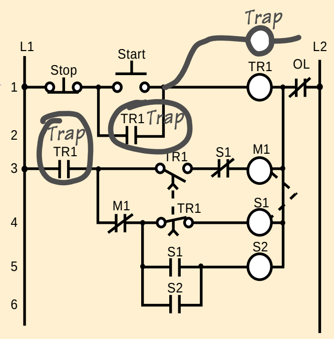
Sample Problem

A sample problem from
Chapter 11 of the workbook:
System Programming and Implementation

Q. Circle the locations where timer traps will be used in the PLC implementation of this reduced-voltage start motor circuit.



A.



Introduction to PLC Programming and Implementation— from relay logic to PLC logic

He that invents a machine augments the power of man and the well-being of mankind.

—Henry Ward Beecher

Key Terms

Control strategy—the sequence of steps that must occur during a process or PLC program to produce the desired output control.

Control task—the desired results of a control program.

Flowcharting—a method of pictorially representing the operation of a process in a sequential manner.

Program coding—the process of translating a logic or relay diagram into PLC ladder program form.

Contents

HIGHLIGHTS

The implementation of a control program requires complex organizational and analytical skills, which change depending on the application. Because they are so varied, we cannot explain how to solve every specific control task. Nevertheless, we can provide you with techniques and guidelines for completing this problem-solving process. In this handbook, we will introduce a strategy for implementing a control program, which includes program organization, system configuration, and I/O programming. These strategies also apply to PLCs with the IEC 1131-3 programming standard. Additionally, we will present both simple and complex PLC programming examples. After you finish, you will be ready to learn how to document the PLC system—the last step in implementing the control program.

1 Control Task Definition

A user should begin the problem-solving process by defining the **control task**, that is, determining what needs to be done. This information provides the foundation for the control program. To help minimize errors, the control task should be defined by those who are familiar with the operation of the machine or process. Proper definition of the task is directly related to the success of the control program.

Control task definition occurs at many levels. All of the departments involved must work together to determine what inputs are required, so that everyone understands the purpose and scope of the project. For example, if a project involves the automation of a manufacturing plant in which materials will be retrieved from the warehouse and sent to the automatic packaging area, personnel from both the warehouse and packaging areas must collaborate with the engineering group during the system definition. Management should also be involved if the project requires data reporting.

If the control task is currently done manually or through relay logic, the user should review the steps of the manual procedure to determine what improvements, if any, can be made. Although relay logic can be directly implemented in a PLC, the procedure should be redesigned, when possible, to meet current project needs and to capitalize on the capabilities of programmable controllers.

2 Control Strategy

After the control task has been defined, the planning of its solution can begin. This procedure commonly involves determining a **control strategy**, the sequence of steps that must occur within the program to produce the desired output control. This part of the program development is known as the development of an algorithm. The term *algorithm* may be new or strange to some readers, but it need not be. Each of us follows algorithms to accomplish certain tasks in our daily lives. The procedure that a person follows to go from home to either school or work is an algorithm—the person exits the house, gets into the car, starts the engine, and so on. In the last of a finite number of steps, he or she reaches the destination.

The PLC strategy implementation for a control task closely follows the development of an algorithm. The user must implement the control from a given set of basic instructions and produce the solution in a finite number of steps. If developing an algorithm to solve the problem becomes difficult, he or she may need to return to the control task definition to redefine the problem. For example, we cannot explain how to get from where we are to Bullfrog County, Nevada unless we know both where we are and where Bullfrog County is. As part of the problem definition, we need to know if a particular method of transportation is required. If there is a time constraint, we need to know that too. We cannot develop a control strategy until we have all of this problem definition information.

The fundamental rule for defining the program strategy is *think first, program later*. Consider alternative approaches to solving the problem and allow time to polish the solution algorithm before trying to program the control function. Adopting this philosophy will shorten programming time, reduce debugging time, accelerate start-up, and focus attention where it is needed—on design when designing and on programming when programming.

Strategy formulation challenges the system designer, regardless of whether it is a new application or the modernization of an existing process. In either case, the designer must review the sequence of events and optimize control through the addition or deletion of steps. This requires a knowledge of the PLC-controlled field devices, as well as input and output considerations.

3 Implementation Guidelines

A programmable controller is a powerful machine, but it can only do what it is told to do. It receives all of its directions from the control program, the set of instructions or solution algorithms created by the programmer. Therefore, the success of a PLC control program depends on how organized the user is. There are many ways to approach a problem; but if the application is approached in a systematic manner, the probability of mistakes is less.

The techniques used to implement the control program vary according to the programmer. Nevertheless, the programmer should follow certain guidelines. Table1 lists programming guidelines for new applications and modernizations. New applications are new systems, while modernizations are upgraded existing control systems that have functioned previously without a PLC (i.e., through electromechanical control or individual, analog, loop controllers).

New Applications	Modernizations
• Understand the desired function of the system.	• Understand the actual process or machine function.
• Review possible control methods and optimize the process operation.	• Review machine logic of operation and optimize when possible.
• Flowchart the process operation.	• Assign real I/O and internal addresses to inputs and outputs.
• Implement the flowchart by using logic diagrams or relay logic symbology.	• Translate relay ladder diagram into PLC coding.
• Assign real I/O addresses and internal addresses to inputs and outputs.	
• Translate the logic implementation into PLC coding.	

Table 1. Programming guidelines.

As mentioned previously, understanding the process or machine operation is the first step in a systematic approach to solving the control problem. For new applications, the strategy should follow the problem definition. Reviewing strategies for new applications, as well as revising the actual method of control for a modernization project, will help detect errors that were introduced during the planning stages.

The programming stage reveals the difference between new and modernization projects. In a modernization project, the user already understands the operation of the machine or process, along with the control task. An existing relay ladder diagram, like the one shown in Figure1, usually defines the sequence of events in the control program. This ladder diagram can be almost directly translated into PLC ladder diagrams.

New applications usually begin with specifications given to the person who will design and install the control system. The designer translates these specifications into a written description that explains the possible control strategies. The written explanation should be simple to avoid confusion. The designer then uses this explanation to develop the control program.

4 Program Organization and Implementation

Organization is a key word when programming and implementing a control solution. The larger the project, the more organization is needed, especially when a group of people is involved.

In addition to organization, a successful control solution also depends on the ability to implement it. The programmer must understand the PLC control task and controlled devices, choose the correct equipment for the job

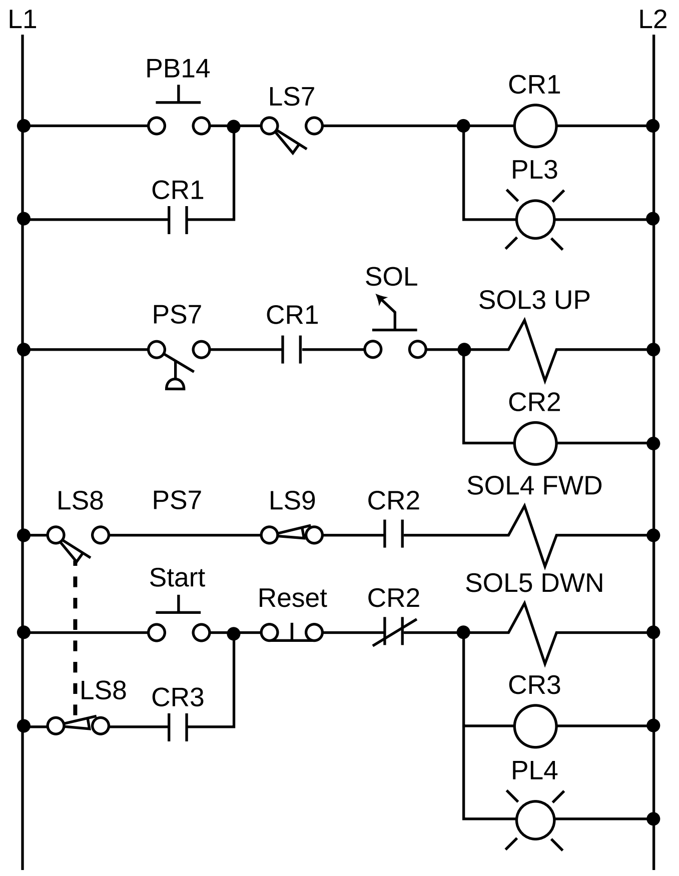


Figure 1. Electromechanical relay circuit diagram.

(hardware and software), and understand the PLC system. Once these preliminary details are understood, the programmer can begin sketching the control program solution. The work performed during this time forms an important part of the system or project documentation. Documenting a system once it is installed and working is difficult, especially if you do not remember how you got it to work in the first place. Therefore, documenting the system throughout its development will pay off in the end.

Creating Flowcharts and Output Sequences

Flowcharting is a technique often used when planning a program after a written description has been developed. A flowchart is a pictorial representation that records, analyzes, and communicates information, as well as describes the operational process in a sequential manner. Figure 2 illustrates a simple flowchart. Each step in the chart performs an operation, whether it is an input/output, decision, or data process.

In a flowchart, broad concepts and minor details, along with their relationship to each other, are readily apparent. Sequences and relationships that are hard to extract from general descriptions also become obvious when expressed through a flowchart. Even the flowchart symbols themselves have specific meanings, which aid in the interpretation of the solution algorithm. Figure 3 illustrates the most common flowchart symbols and their meanings.

The main flowchart itself should not be long and complex; instead, it should point out the major functions to be performed (e.g., compute engineering units from analog input counts). Several smaller flowcharts can be used to further describe the functions specified in the main flowchart.

Once the flowchart is completed, the user can employ either logic gates or contact symbology to implement the logic sequences. Logic gates implement a logical output sequence given specific real and/or internal input conditions,

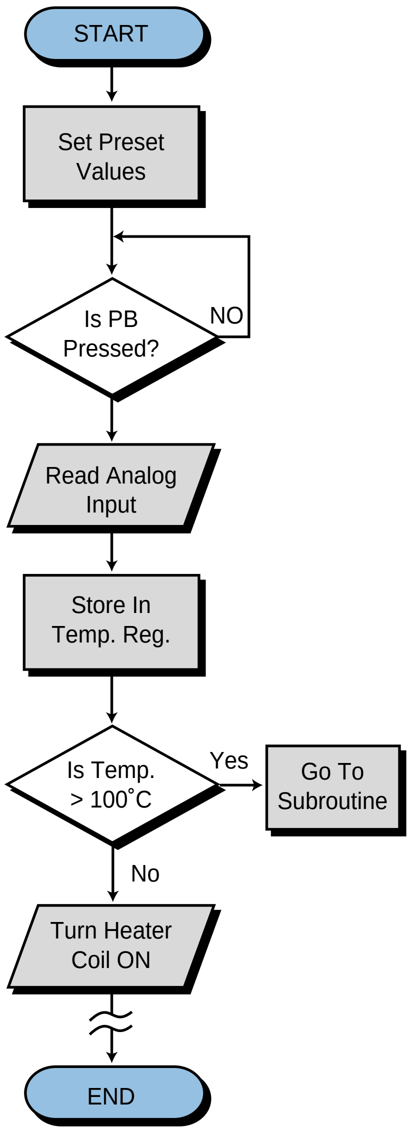
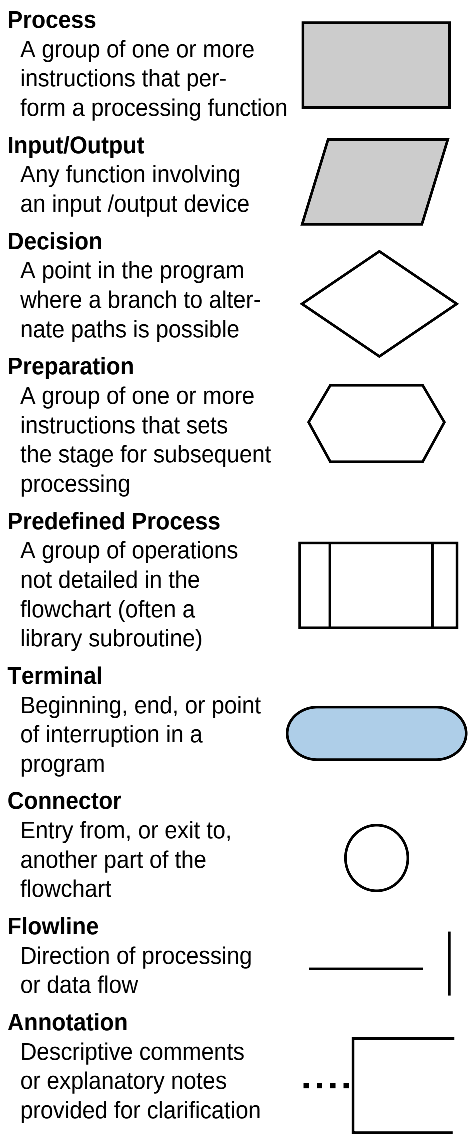


Figure 2. Simple flowchart.



Process
A group of one or more instructions that perform a processing function

Input/Output
Any function involving an input /output device

Decision
A point in the program where a branch to alternate paths is possible

Preparation
A group of one or more instructions that sets the stage for subsequent processing

Predefined Process
A group of operations not detailed in the flowchart (often a library subroutine)

Terminal
Beginning, end, or point of interruption in a program

Connector
Entry from, or exit to, another part of the flowchart

Flowline
Direction of processing or data flow

Annotation
Descriptive comments or explanatory notes provided for clarification

Figure 3. Flowchart symbols.

while PLC contact symbology directly implements the logic necessary to program an output rung. Figure 4 illustrates both of these programming methods. Users should employ whichever method they feel most comfortable with or, perhaps, a combination of both (see Figure 5). Logic gate diagrams, however, may be more appropriate in controllers that use Boolean instruction sets.

Inputs and outputs marked with an X on a logic gate diagram, as in Figure 4b, represent real I/O in the system. If no mark is present, an I/O point is an internal. The labels used for actual input signals can be either the actual device names (e.g., LS1, PB10, AUTO, etc.) or symbolic letters and numbers that are associated with each of the field elements. During this stage, the user should prepare a short description of the logic sequence.

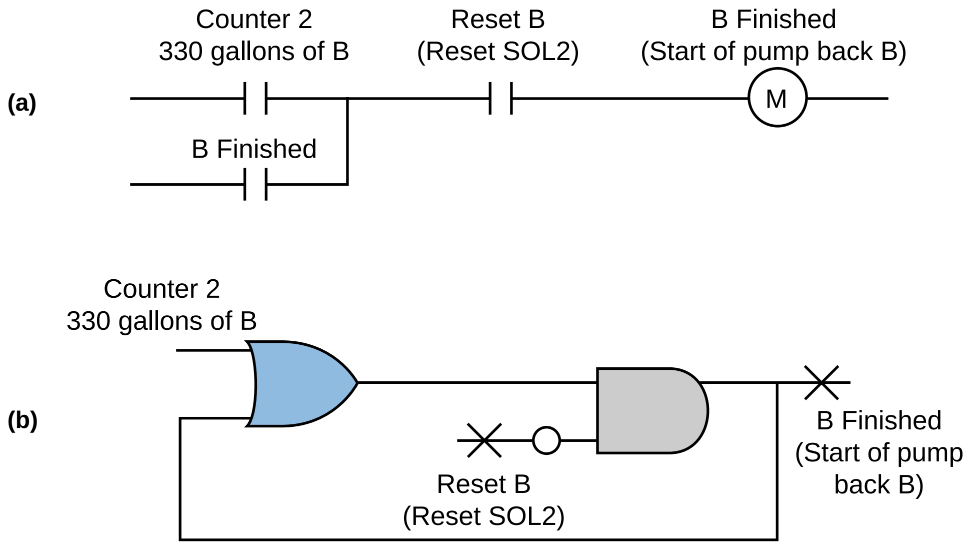


Figure 4. (a) PLC contact symbology and **(b)** logic gate representation of a logic sequence.

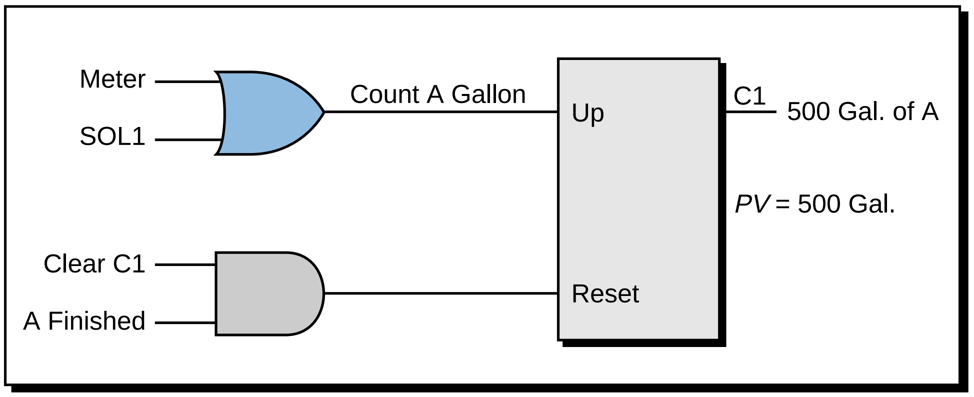


Figure 5. A combination of logic gates and contact symbology.

Configuring the PLC System

PLC configuration should be considered during flowcharting and logic sequencing. The PLC's configuration defines which I/O modules will be used with which types of I/O signals, as well as where the modules will be located in the local or remote rack enclosures. The modules' locations determine the I/O addresses that will be used in the control program.

During system configuration, the user should consider the following: possible future expansions; special I/O modules, such as fast-response or wire fault inputs; and the placement of interfaces within a rack (all AC I/O together, all DC and low-level analog I/O together, etc.). Consideration of these details, along with system configuration documentation, will result in a better system design.

Real and Internal I/O Assignment

The assignment of inputs and outputs is one of the most important procedures that occurs during the programming organization and implementation stages. The I/O assignment table documents and organizes what has been done thus far. It indicates which PLC inputs are connected to which input devices and which PLC outputs drive which output devices. The assignment of internals, including timers, counters, and MCRs, also takes place here. These assignments are the actual contact and coil representations that are used in the ladder diagram program. In applications where electromechanical relay diagrams are available (e.g., modernization of a machine or process), identification of real I/O can be done by circling the devices and then assigning them I/O addresses (see Example 1).

Table 2 shows an I/O address assignment table for real inputs and outputs, while Table 3 shows an I/O address assignment table for internals. These assignments can be extracted from the logic gate diagrams or ladder symbols

	I/O Address			
Module Type	**Rack**	**Group**	**Terminal**	**Description**
Input	0	0	0	LS1—Position
	0	0	1	LS2—Detect
	0	0	2	Sel Switch—Select 1
	0	0	3	PB1—Start
Output	0	0	4	SOL1
	0	0	5	PL1
	0	0	6	PL2
	0	0	7	Motor M1
Output	0	1	0	SOL2
	0	1	1	PL3

Table 2. I/O address assignment table for real inputs and outputs.

Device	Internal	Description
CR7	1010	CR7 replacement
TDR10	T200	ON-delay timer 12 sec
CR10	1011	CR10 replacement
CR14	1012	CR14 replacement
—	1013	Setup interlock

Table 3. I/O address assignment table for internal outputs.

that were used to describe the logic sequences. They can also come from the circled elements on an electromechanical diagram. The numbers used for the I/O addresses depend on the PLC model used. These addresses can be represented in octal, decimal, or hexadecimal. The description section of the table specifies the field devices that correspond to each address.

The table of address assignments should closely follow the input/output connection diagram (see Figure 6). Although industry standards for I/O representations vary among users, inputs and outputs are typically represented by squares and diamonds, respectively. The I/O connection diagram forms part of the documentation package.

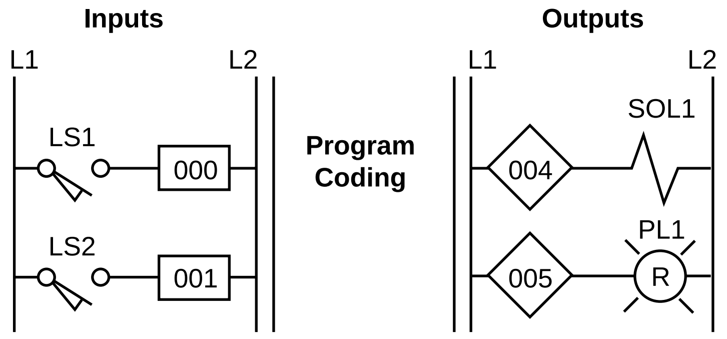


Figure 6. Partial connection diagram for the I/O address assignment in Table 2.

During the I/O assignment, the user should group associated inputs and outputs. This grouping will allow the monitoring and manipulation of a group of I/O simultaneously. For instance, if 16 motors will be started sequentially, they should be grouped together, so that monitoring the I/O registers associated with the 16 grouped I/O points will reveal the motors' starting sequence. Due to the modularity of an I/O system, all the inputs and all the outputs should be assigned at the same time. This practice will prevent the assignment of an input address to an output module and vice versa.

EXAMPLE 1

For the circuit shown in Figure 7, **(a)** identify the real inputs and outputs by circling each, **(b)** assign the I/O addresses, **(c)** assign the internal addresses (if required), and **(d)** draw the I/O connection diagram.

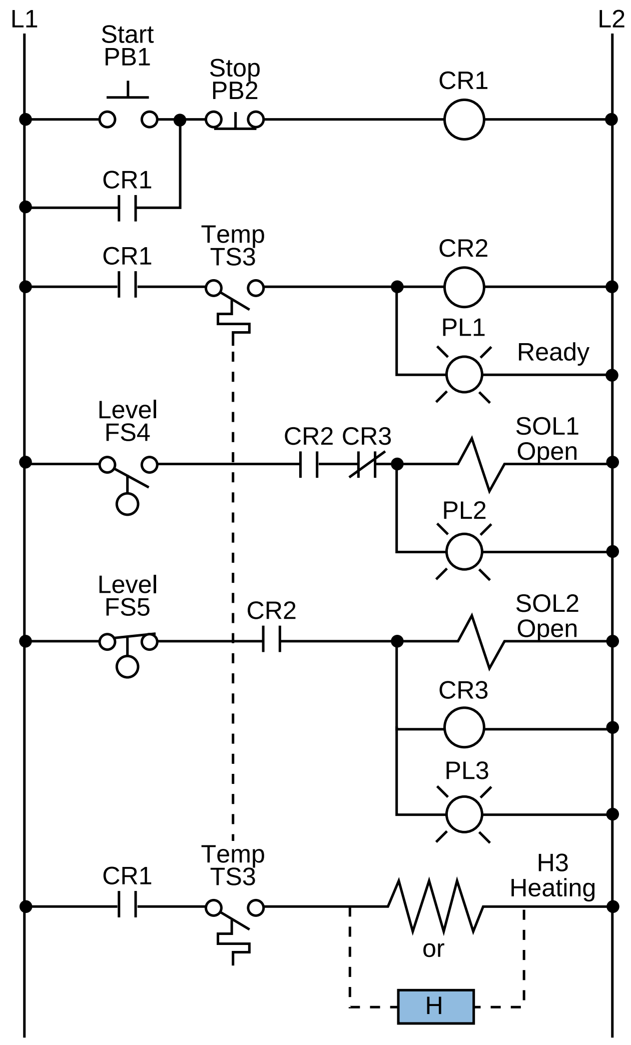


Figure 7. Electromechanical relay circuit.

Assume that the PLC used has a modularity of 8 points per module. Each rack has 8 module slots, and the master rack is number 0. Inputs and outputs can have any address as long as the correct module is used. The PLC determines whether an input or output module is connected in a slot. The number system is octal, and internals start at address 1000_8.

Solution

(a) Figure 8 shows the circled real input and output connections. Note that temperature switch TS3 is circled *twice* even though it is only *one* device. In the address assignment, only one of them is referenced, and only one of them is wired to an input module.

(b) Table 4 illustrates the assignment of inputs and outputs. It assigns all inputs and all outputs, leaving spare I/O locations for future use.

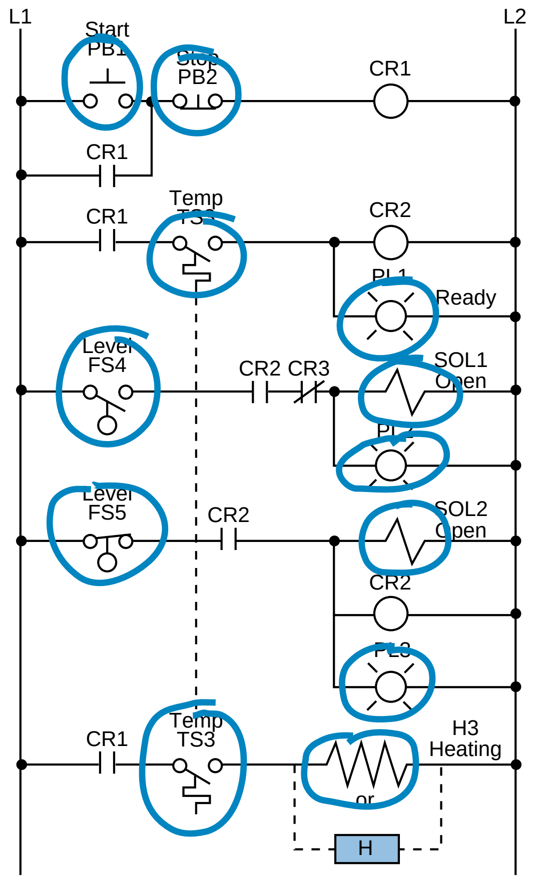


Figure 8. Identification of real I/O (circled).

Module Type	I/O Address Rack	Group	Terminal	Description
Input	0	0	0	Start PB1
	0	0	1	Stop PB2
	0	0	2	Temp TS3
	0	0	3	Level FS4
	0	0	4	Level FS5
	0	0	5	—
	0	0	6	—
	0	0	7	—
Spare	0 · · · 0	1 · · · 1	0 · · · 7	Not used
Output	0	2	0	PL1 Ready
	0	2	1	SOL1 Open
	0	2	2	PL2
	0	2	3	SOL2 Open
	0	2	4	PL3
	0	2	5	H3 Heating
	0	2	6	—
	0	2	7	—

Table 4. I/O address assignment.

(c) Table 5 presents the output assignments, including a description of each internal. Note that control relay CR2 is not assigned as an internal since it is the same as the output rung corresponding to PL1. When the control program is implemented, every contact associated with CR2 will be replaced by contacts with address 020 (the address of PL1).

Device	Internal	Description
CR1	1000	Control relay CR1
CR2	—	Same as PL1 Ready
CR3	—	Same as SOL2 Open

Table 5. Internal output assignment.

(d) Figure 9 illustrates the I/O connection diagram for the circuit in Figure 7. This diagram is based on the I/O assignment from part (b). Note that only one of the temperature switches, the normally open TS3 switch, is a connected input. The logic programming of each switch should be based on a normally open condition.

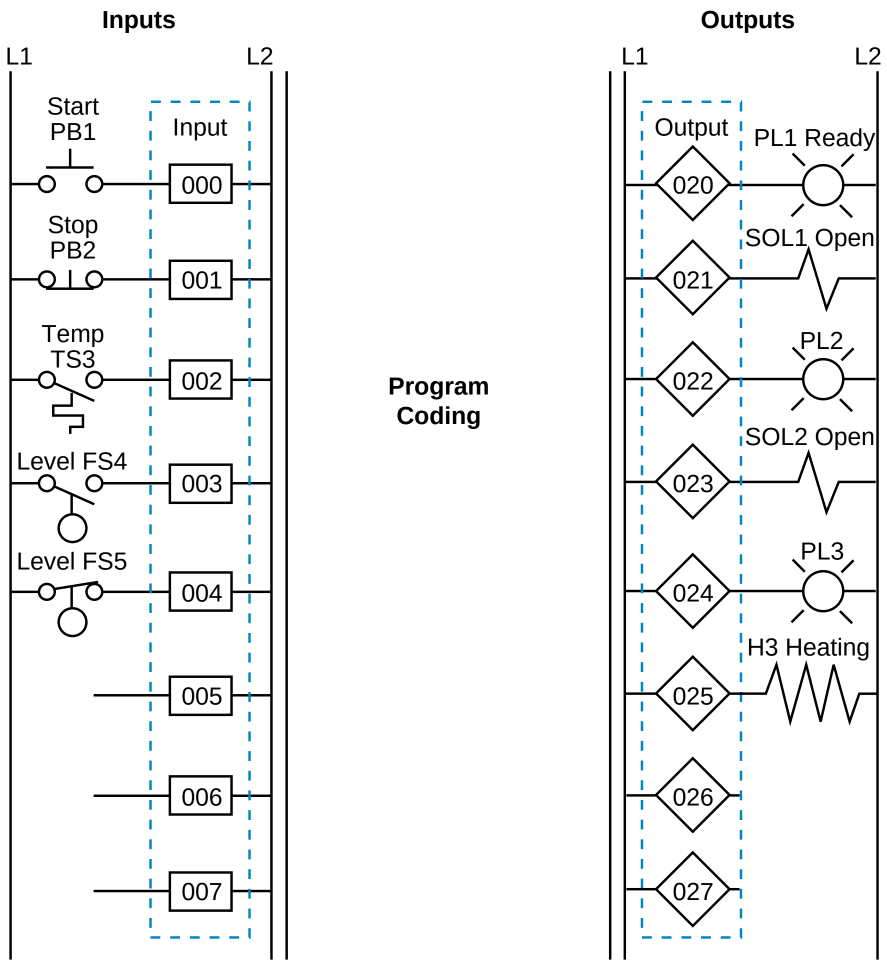


Figure 9. I/O connection diagram.

Register Address Assignment

The assignment of addresses to the registers used in the control program is another important aspect of PLC organization. The easiest way to assign registers is to list all of the available PLC registers. Then, as they are used, describe each register's contents, description, and function in a register assignment table. Table 6 shows a register assignment table for the first 15 registers in a PLC system, ranging from address 2000_8 to address 2016_8.

Register	Contents	Description
2000	Analog input	Temperature input temp 3 (inside)
2001	Analog input	Temperature input temp 4 (outside)
2002	Spare	–
2003	Spare	–
2004	TWS input	Set point (SP1) input from TWS panel 1
2005	TWS input	Set point volume (V1) from TWS panel 2
2006	Constant 2350	Timer constant of 23.5 sec (0.01 sec TB)
2007	Accumulated	Accumulated value for counter R2010
2010	Spare	–
2011	Spare	–
2012	Constant 1000	Beginning of look-up table (value #1)
2013	Constant 1010	Look-up value #2
2014	Constant 1023	Look-up value #3
2015	Constant 1089	Look-up value #4
2016	Constant 1100	Look-up value #5

Table 6. Register assignment table.

Elements to Leave Hardwired

During the assignment of inputs and outputs, the user should decide which devices will not be wired to the controller. These elements will remain part of the electromechanical control logic. These elements usually include devices that are not frequently switched off after start, such as compressors and hydraulic pumps. Components like emergency stops and master start push buttons should also remain hardwired, principally for safety purposes. This way, if the controller is faulty and an emergency occurs, the user can shut down the system without PLC intervention.

Figure 10 provides an example of system components that are typically left hardwired. Note that the normally open PLC Fault Contact 1 (or watchdog timer contact) is wired in series with other emergency conditions. This contact stays closed when the controller is operating correctly, but opens when a fault occurs. The system designer can also use this contact if an emergency occurs to disable the PLC system's operation.

PLC fault contacts are safety contacts that are available to the user when implementing or enhancing a safety circuit. When a PLC is operating correctly, the normally open fault contact closes and the normally closed one opens when the PLC is first turned on. As shown in Figure 10, these contacts are connected in series with the hardwired circuit, so that if the PLC fails during standard operation, the normally open contacts will open. This will shut down the hardwired circuit at the point where the PLC becomes the controlling element. This circuit also uses a safety control relay (SCR) to control power to the rest of the control components. The normally closed fault contacts are used to indicate an alarm condition.

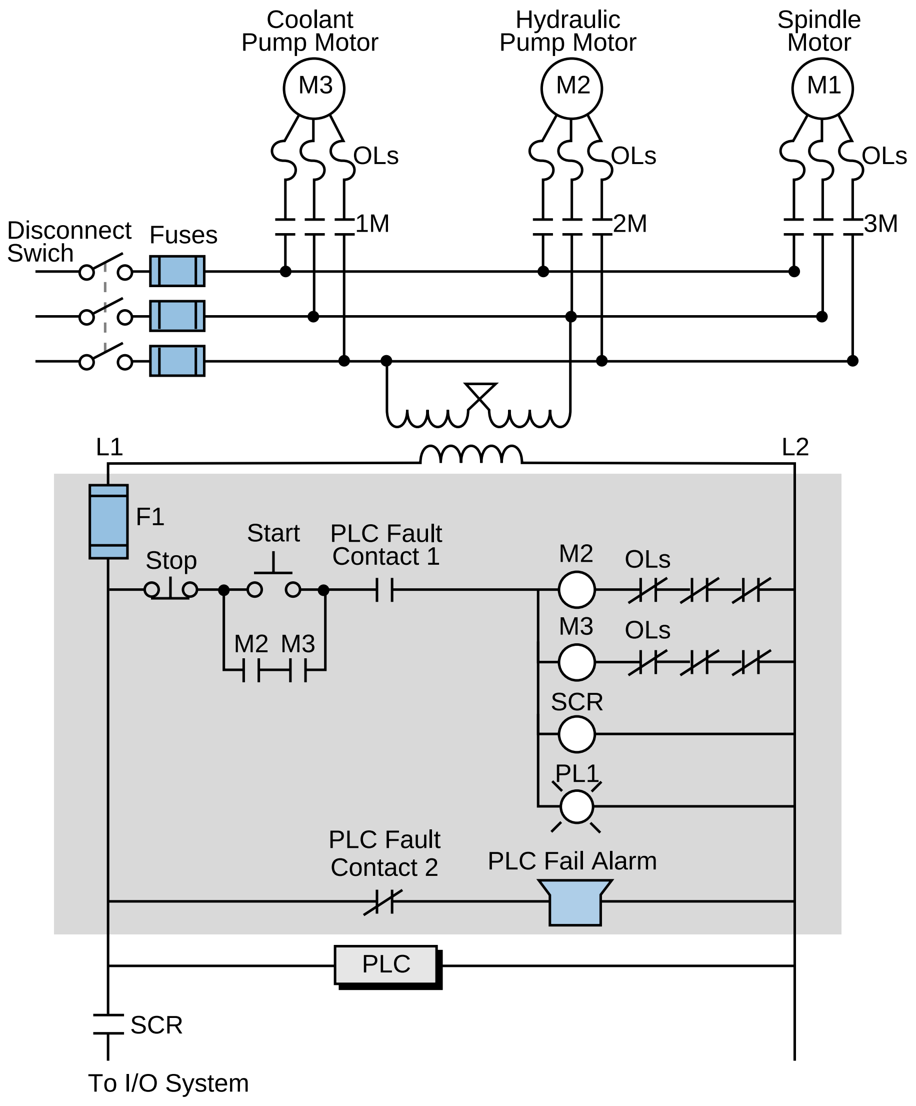


Figure 10. Hardwired components in a PLC system.

In the diagram shown in Figure 10, an emergency situation (including a PLC malfunction) will remove power (L1) to the I/O modules. The turning OFF of the safety control relay (SCR) will open the SCR contact, stopping the flow of power to the system. Furthermore, the normally closed PLC fault contact (PLC Fault Contact 2) in the hardwired section will alert personnel of a system failure due to a PLC malfunction. The designer should implement this type of alarm in the main PLC rack, as well as in each remote I/O rack location, since remote systems also have fault contacts incorporated into the remote controllers. This allows subsystem failures to be signaled promptly, so that the problem can be fixed without endangering personnel.

SPECIAL INPUT DEVICE PROGRAMMING

Some PLC circuits and input connections require special programming. One example is the programming of normally closed input devices. Remember that the programming of a device is closely related to how that device should behave in the control program.

Normally Closed Devices. An input device that is wired as a normally open input can be programmed to act as either a normally open or a normally closed device. The same rule applies for normally closed inputs. Generally, if a device is wired as a normally closed input and it must act as a normally closed input, its reference address is programmed as normally open. As the following example illustrates, however, a normally closed device in a hardwired circuit is programmed as normally closed when it is replaced in the PLC control program. Since it is not referenced as an input, the program does not evaluate the device as a real input.

EXAMPLE 2

For the circuit in Figure 11, draw the PLC ladder program and create an I/O address assignment table. For inputs, use addresses 10_8 through 47_8. Start outputs at address 50_8 and internals at address 100_8.

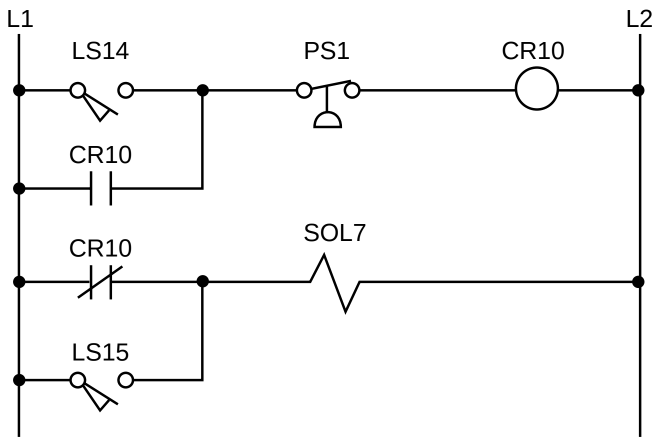


Figure 11. Electromechanical relay circuit.

SOLUTION

Figure 12 shows the equivalent PLC ladder diagram for the circuit in Figure 11. Table 7 shows the I/O address assignment table for this example. The normally closed contact (CR10) is programmed as normally closed because internal coil 100 references it and requires it to operate as a normally closed contact.

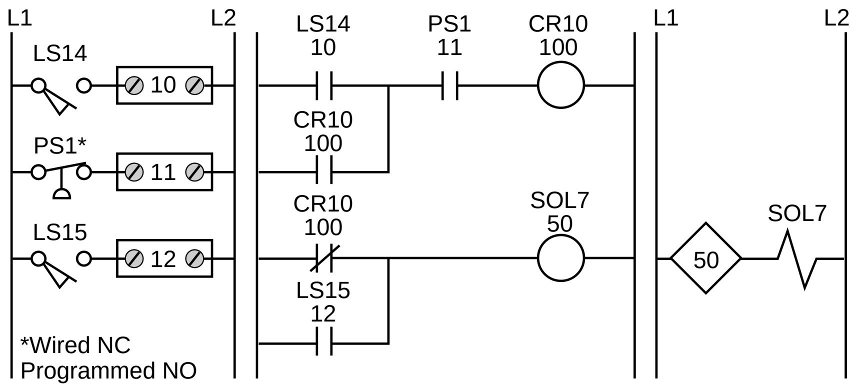


Figure 12. PLC ladder diagram of the circuit in Figure 11.

I/O Address	Device	Type
10	LS14	Input
11	PS1	Input
12	LS15	Input
50	SOL7	Output
100	CR10	Internal

Table 7. I/O address assignment table.

Master Control Relays. Another circuit the programmer should be aware of is a master control relay (MCR). In electromechanical circuit diagrams, an MCR coil controls several rungs in a circuit by switching ON or OFF the power to those rungs. In a hardwired circuit, there is no definite end to an MCR except when the circuit is followed all the way through. For example, in Figure13, the MCR output in line 1 controls the power to the hardwired

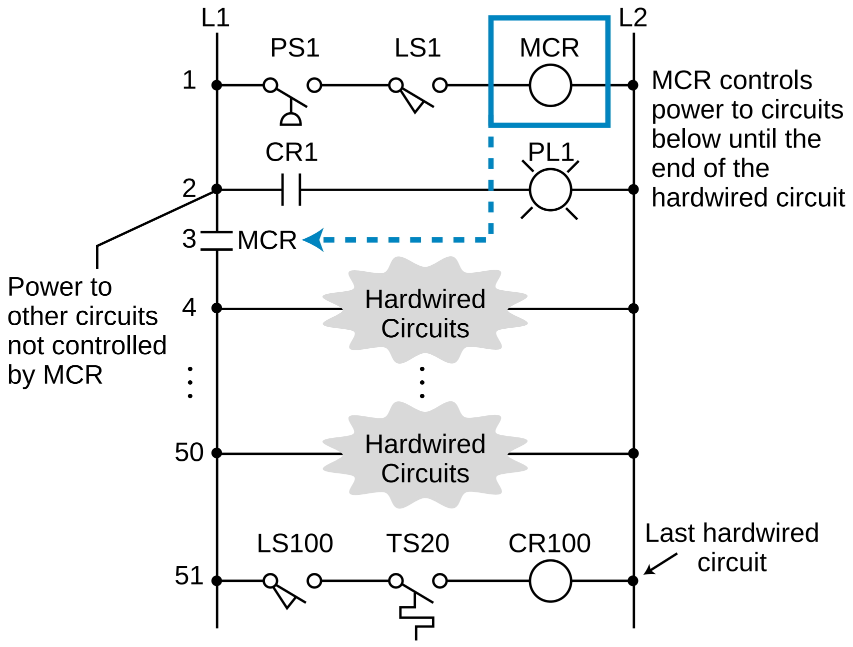

L1
L2
1
PS1
LS1
MCR
MCR controls power to circuits below until the end of the hardwired circuit
2
CR1
PL1
3
MCR
Power to other circuits not controlled by MCR
4
Hardwired Circuits
50
Hardwired Circuits
51
LS100
TS20
CR100
Last hardwired circuit

Figure 13. Electromechanical relay circuit with a master control relay.

elements from line 3, where the MCR contact is located, to the last element in line 51. If the master control relay is ON, power will flow to these rungs (lines 4 through 51). If the master control relay is OFF, power will not flow and these devices will not implement the control action. This configuration is equivalent to a hardwired subprogram or subroutine—if the MCR is ON, the rungs are executed; if it is OFF, the rungs are not executed. At line 2 in the circuit, power branches to other circuits that are not affected by the MCR's action. These circuits are the regular hardwired program.

During the translation from a hardwired ladder circuit to PLC symbology, the programmer must place an END MCR instruction after the last rung the MCR should control. Figure14 illustrates the placement of the MCR instruction for the circuit in Figure 13. To provide proper fencing for the program's MCR control section, internal output coil 1000, labeled CR1 (line 1 of PLC program), was inserted so that PL1 would not be inside the fenced MCR area. This is the way the hardwired circuit operates. The END1 instruction

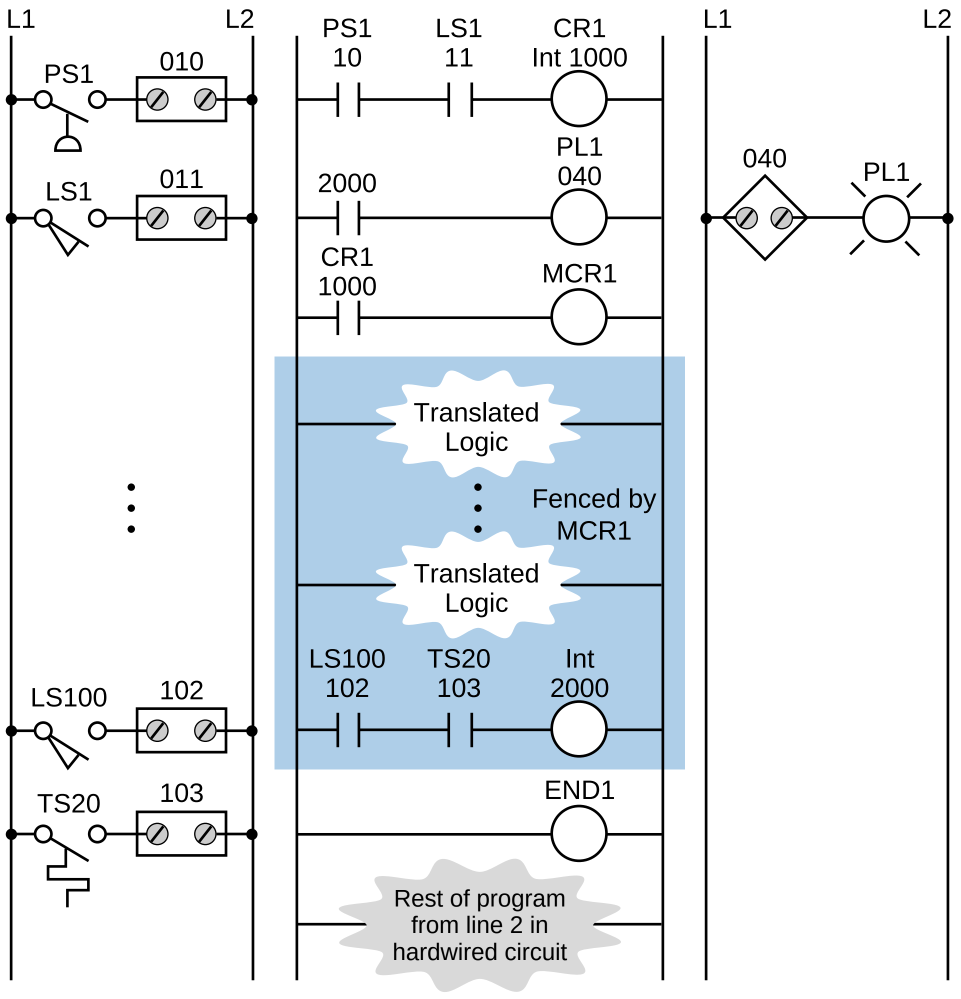


Figure 14. PLC ladder diagram with MCR fence.

ends the MCR fence. The instructions corresponding to the hardwired circuits that branch from line 2 in the electromechanical diagram of Figure 13 are located after the END1 instruction. Figure15 illustrates a partial ladder rung of a more elaborate circuit with this type of MCR condition. The corresponding PLC program should have an END MCR after the rung containing the PL3 output.

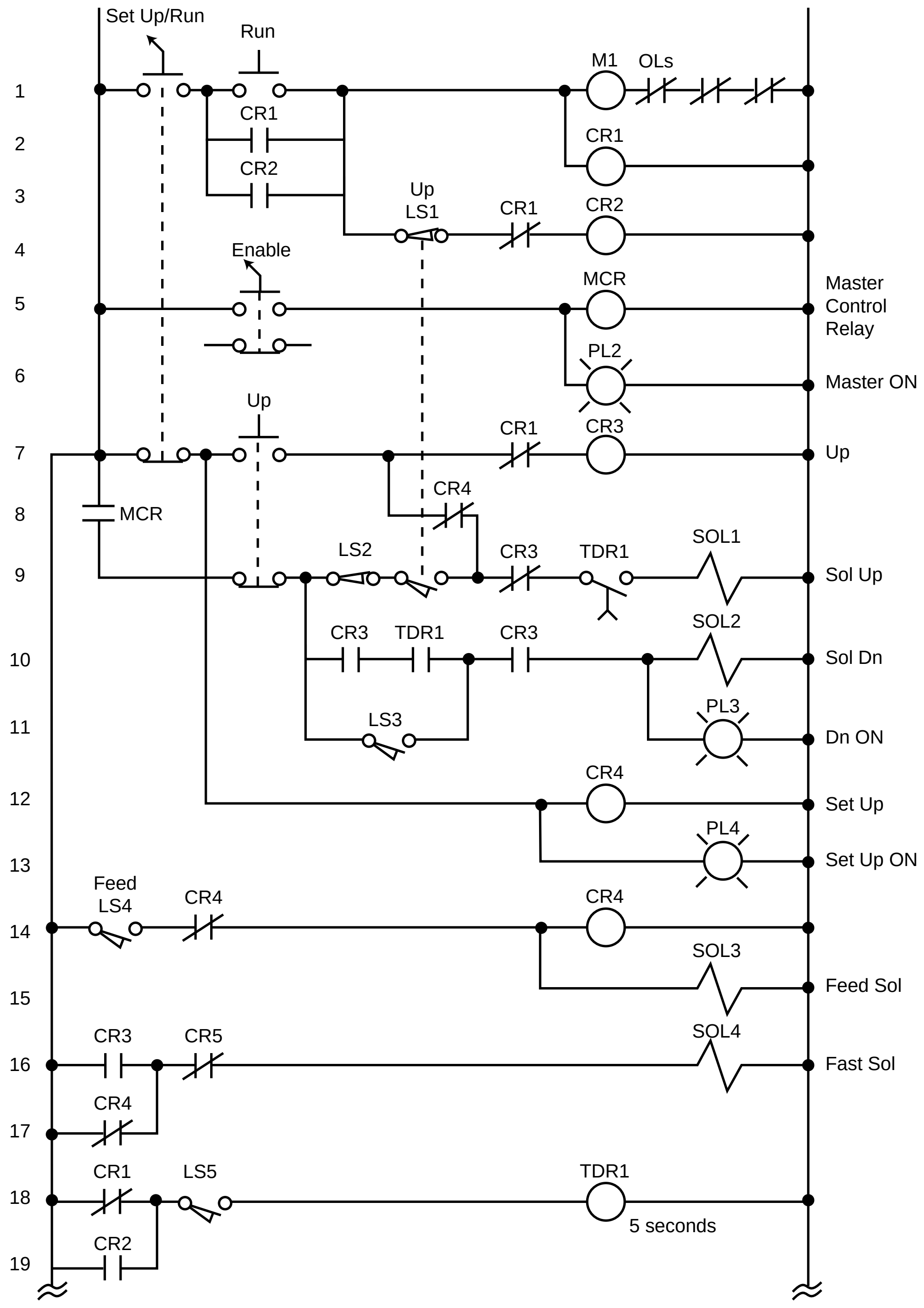


Figure 15. Electromechanical relay circuit with an MCR.

Example 3

Highlight the sections of the circuit in Figure 15 that will be under the control of a PLC MCR. What additional measures must be taken to include or bypass other hardwired circuits within the MCR fence?

Solution

Figure 16 highlights the circuits that must be fenced under the MCR instruction. Note that solenoid SOL1 and part of its driving logic are not included in the MCR fencing because SOL1, CR3, and TDR1 can also be turned ON by logic prior to the MCR fence (see Figure 17). For the MCR fence to be properly programmed, the PLC program must

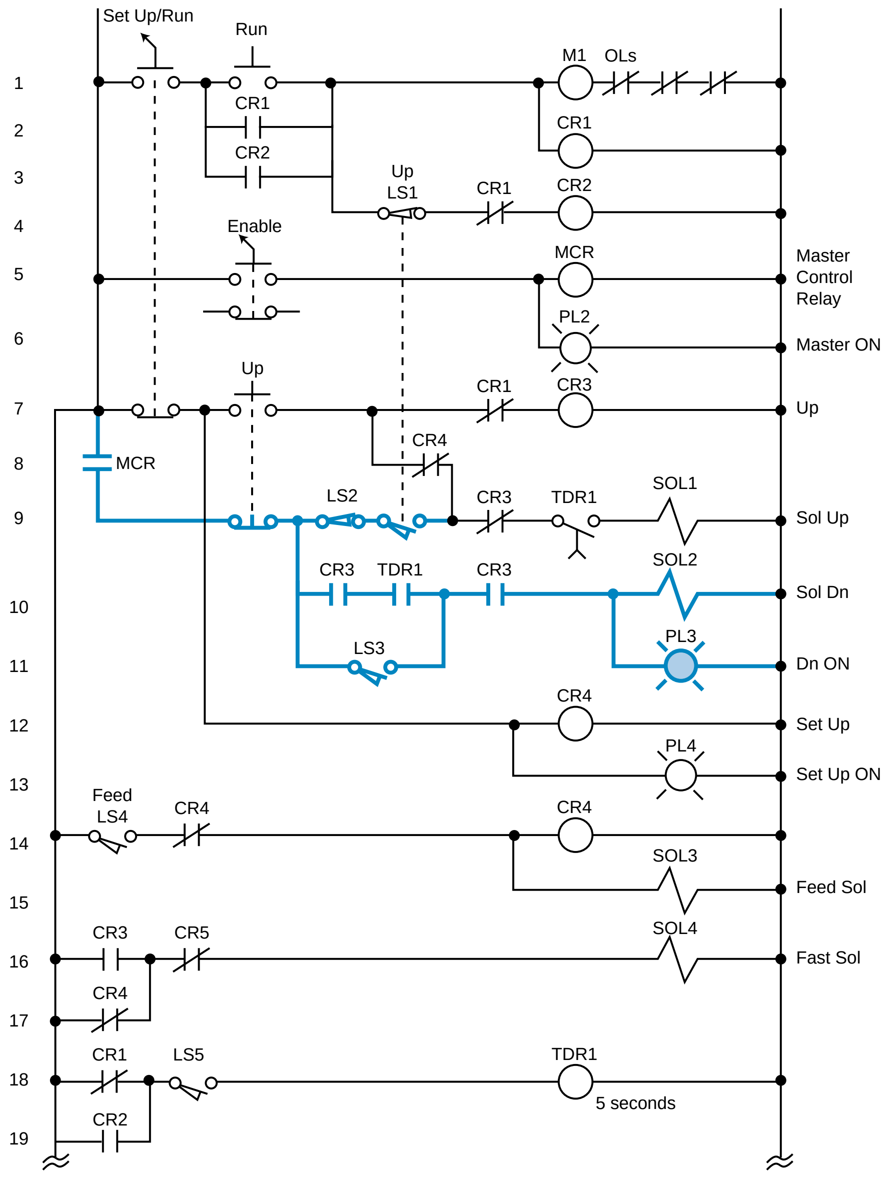


Figure 16. MCR-controlled program elements.

include two internal control relays that take SOL1 out of the fence. Figure 18 illustrates the fenced circuit with the additional internals (CR1000 and CR1001). Note that the instructions in this diagram have the same names as in the hardwired circuit. The solenoid SOL1 will be outside of the MCR fence because it can be turned ON by either the outside logic (highlighted section in Figure 17) or the logic inside the MCR fence (highlighted section in Figure 18).

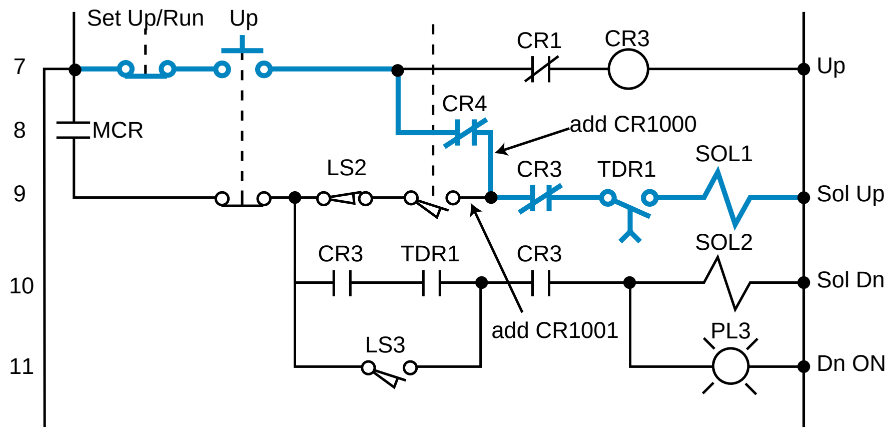


Figure 17. SOL1 activated by logic outside of the MCR fence.

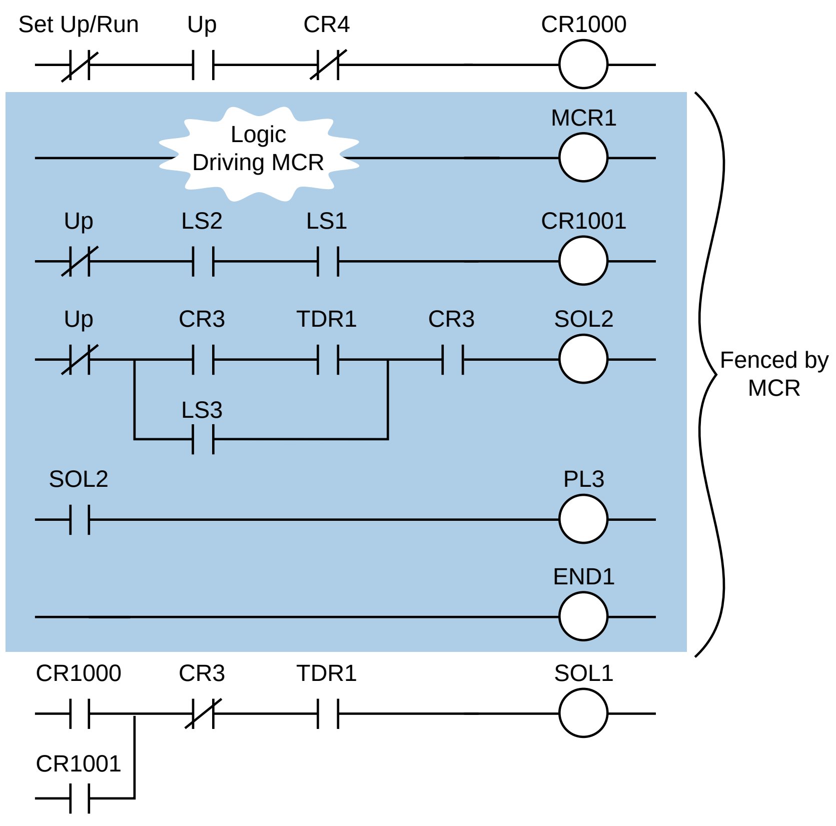
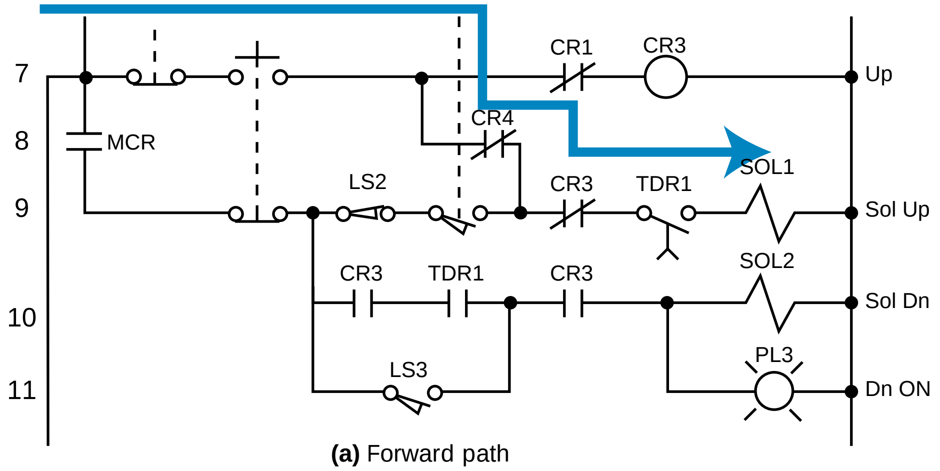
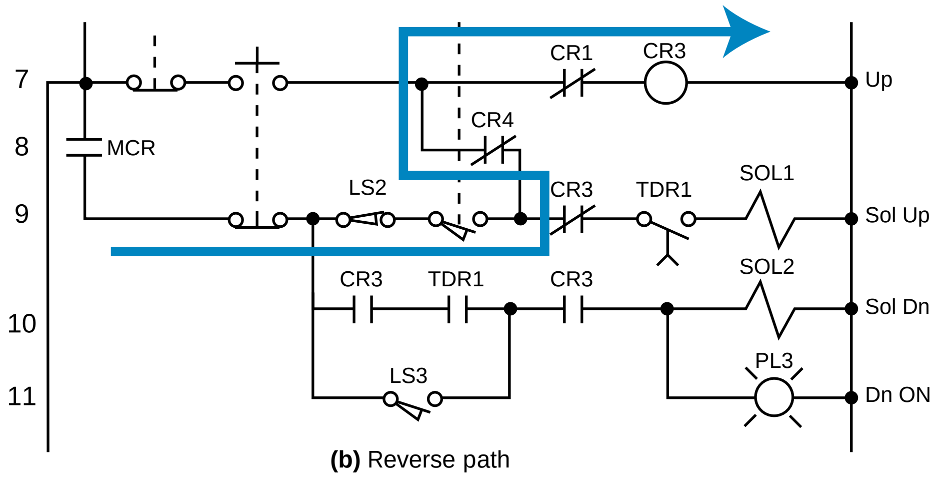


Figure 18. MCR fence.

Bidirectional Power Flow. The circuit in Figure 19 illustrates another condition that can cause programming problems: the possibility of bidirectional power flow through the normally closed CR4 contact in line 8. To solve the bidirectional flow problem, the programmer must know whether or not CR4 influences the two output rungs to which it is connected. These rungs are the CR3 control relay output and the solenoid SOL1 output (rungs 7 and 9, respectively). Figure 19 illustrates the two paths that can occur in the hardwired circuit. PLCs only allow forward paths; therefore, if a reverse path is necessary for this circuit's logic, the CR4 contact must be included in the logic driving the CR3 output (see Figure 9b).



(a) Forward path



(b) Reverse path

Figure 19. (a) Forward and **(b)** reverse power flow in a hardwired circuit.

Instantaneous Timer Contacts. The electromechanical circuit shown in Figure 15 specifies an instantaneous timer contact (the normally open TDR1 contact in line 10). This type of contact, however, is usually unavailable in PLCs. To implement an instantaneous timer contact (i.e., a contact that closes or opens once the timer is enabled), the programmer must use an internal output to trap the timer, then use the internal's contact as an instantaneous contact to drive the timer's logic.

In the electromechanical circuit in Figure 20a, if PB1 and LS1 both close, the timer will start timing and the instantaneous contact (TMR1-1) will close, thus sealing PB1. If PB1 is released (OFF), the timer will continue to time because the circuit is sealed. Figure 20b illustrates the technique for trapping a timer. In this PLC program, an internal output traps the instantaneous contact from the circuit's electromechanical timer. Thus, the contacts from this internal drive the timer. If a trap does not exist, the timer will start timing when PB1 and LS1 both close, but will stop timing as soon as PB1 is released.

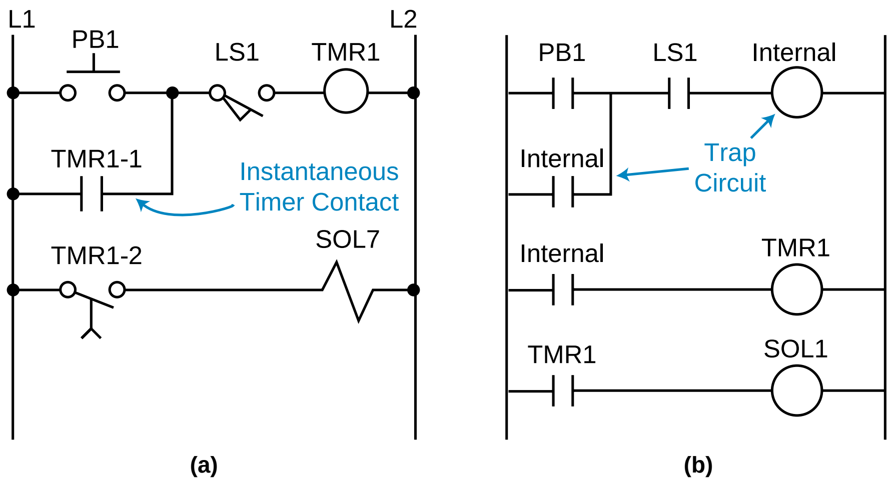


Figure 20. (a) An instantaneous timer contact in a hardwired circuit and **(b)** a trapped timer in a PLC circuit.

Complicated Logic Rungs. When a logic rung is very confusing, the best programming procedure is to isolate it from the other rungs. Then, reconstruct all of the possible logic paths from right to left, starting at the output and ending at the beginning of the rung. If a section of a rung, like the one discussed in Example 3, directly connects or interacts with another rung, it may be easier to create an internal output at the point where the two rungs cross. Then, use the internal output to drive the rest of the logic. For the circuit shown in Figure 15, this cross point is in line 9 at the normally closed contact CR4 between normally open LS1 and normally closed CR3.

Program Coding/Translation

Program coding is the process of translating a logic or relay diagram into PLC ladder program form. This ladder program, which is stored in the application memory, is the actual logic that will implement the control of the machine or process. Ease of program coding is directly related to how orderly the previous stages (control task definition, I/O assignment, etc.) have been done. Figure 21 shows a sample program code generated from logic gates and electromechanical relay diagrams (internal coil 1000 replaces the control relay). Note that the coding is a PLC representation of the logic, whether it is a new application or a modernization. The next sections examine this coding process closer and present several programming examples.

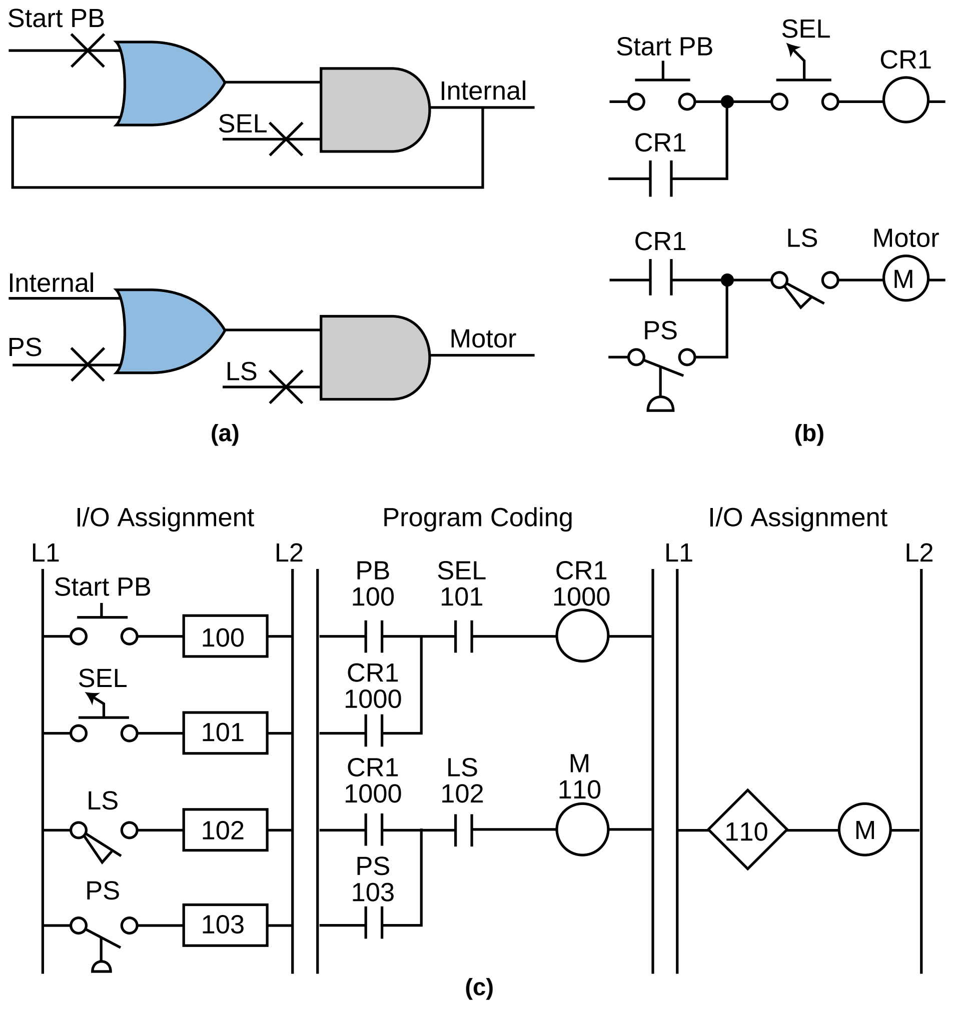


Figure 21. Translation from **(a)** logic gates and **(b)** an electromechanical relay diagram into **(c)** PLC program coding.

5 Discrete I/O Control Programming

In this section, we will present several programming examples that illustrate the modernization of relay systems. We will also present examples relating to new PLC control implementations. These examples will deal primarily with discrete controls. The next section will explain more about analog I/O interaction and programming.

Control Programming and PLC Descriptions

Modernization applications involve the transfer of a machine or process's control from conventional relay logic to a programmable controller. Conventional hardwired relay panels, which house the control logic, usually present maintenance problems, such as contact chatter, contact welding, and other electromechanical problems. Switching to a PLC can improve the performance of the machine, as well as optimize its control. The machine's "new" programmable controller program is actually based on the instructions and control requirements of the original hardwired system.

Throughout this section, we will use the example of a midsized PLC capable of handling up to 512 I/O points (000 to 777 octal) to explain how to implement and configure a PLC program. The I/O structure of the controller has 4 I/O points per module. The PLC has eight racks (0 through 7), each one with eight slots, or groups, where modules can be inserted. Figure 22 illustrates this configuration.

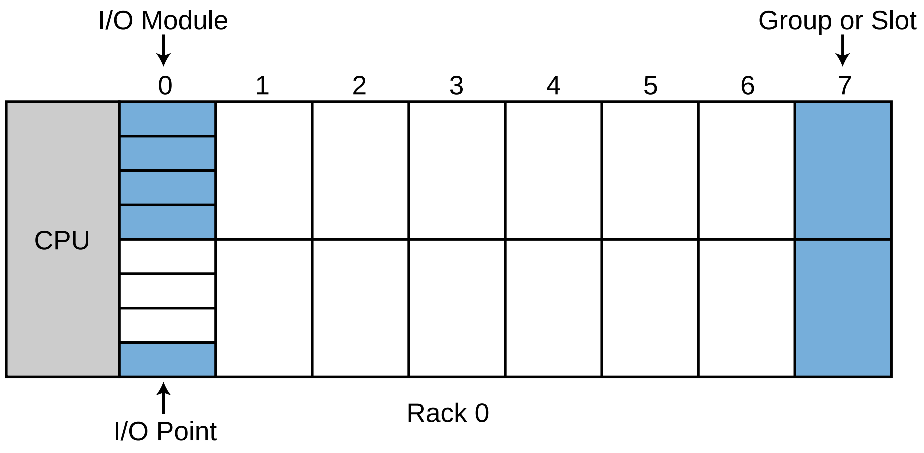


Figure 22. Example PLC configuration.

The PLC can accept four-channel analog input modules, which can be placed in any slot location. When analog I/O modules are used, discrete I/O cannot be used in the same slot. The PLC can also accept multiplexed register I/O. These multiplexed modules require two slot positions and provide the enable (select) lines for the I/O devices.

Addresses 000 through 777 octal represent input and output device connections mapped to the I/O table. The first digit of the address represents the rack number, the second digit represents the slot, and the third digit specifies the terminal connection in the slot. The PLC detects whether the slot holds an input or an output.

Point addresses 1000_8 to 2777_8 may be used for internal outputs, and register storage starts at register 3000_8 and ends at register 4777_8. Two types of timer and counter formats can be used—ladder format and block format—but all timers require an internal output to specify the ON-delay output. Ladder format timers place a “T” in front of the internal output address, while block format timers specify the internal output address in the block’s output coil.

Throughout the examples presented in this section and the next, we will use addresses 000_8 through 027_8 for discrete inputs and addresses 030_8 through 047_8 for discrete outputs. Analog I/O will be placed in the last slot of the master rack (0) whenever possible. During the development of these examples, you will discover that sometimes the assignment of internals and registers is performed parallel to the programming stages.

Simple Relay Replacement

This relay replacement example involves the PLC implementation of the electromechanical circuit shown in Figure 23. The hardware timer TMR1 requires instantaneous contacts in the first rung, which are used to latch the

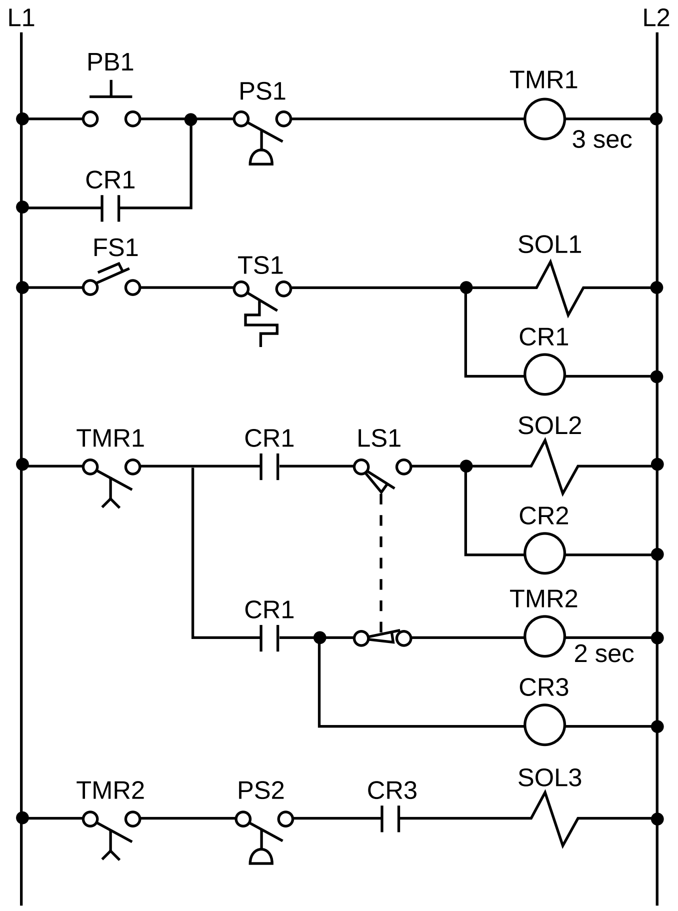


Figure 23. Electromechanical relay circuit.

rung. If the instantaneous TMR1 contacts are implemented using a PLC time-delay contact, then PB1 must be pushed for the timer's required time preset to latch the rung. This instantaneous contact will be implemented by trapping the timer with an internal output.

Tables 8 and 9 show the I/O address and internal output assignments for the electromechanical circuit's real I/O. Table 10 presents the register assignment table. Note that internals do not replace control relays CR1 and CR2 since the output addresses 030 and 031 corresponding to solenoids SOL1 and SOL2 are available. Therefore, addresses 030 and 031 can replace the CR1 and CR2 contacts, respectively, everywhere they occur in the program. The normally open contact LS1 connects limit switch LS1 to the PLC input interface; and the normally open LS1 reference, programmed with an examine-OFF instruction, implements the normally closed LS1 in the program. Figure 24 illustrates the PLC program coding solution.

	I/O Address			
Module Type	**Rack**	**Group**	**Terminal**	**Description**
Input	0	0	0	PB1
	0	0	1	PS1
	0	0	2	FS1
	0	0	3	TS1
Input	0	0	4	LS1
	0	0	5	PS2
	0	0	6	—
	0	0	7	—
Output	0	3	0	SOL1
	0	3	1	SOL2
	0	3	2	SOL3
	0	3	3	—

Table 8. I/O address assignment.

Device	Internal	Description
TMR1	1000	Used to trap TMR1
CR1	—	Same as SOL1 (030)
CR2	—	Same as SOL2 (031)
TMR1	1001	Timer TMR1
TMR2	1002	Timer TMR2
CR3	1003	Replace CR3

Table 9. Internal address assignment.

Register	Description
4000	Preset timer count for 3 sec
4001	Accumulated count timer 1001
4002	Preset timer count for 2 sec
4003	Accumulated count timer 1002

Table 10. Register assignment.

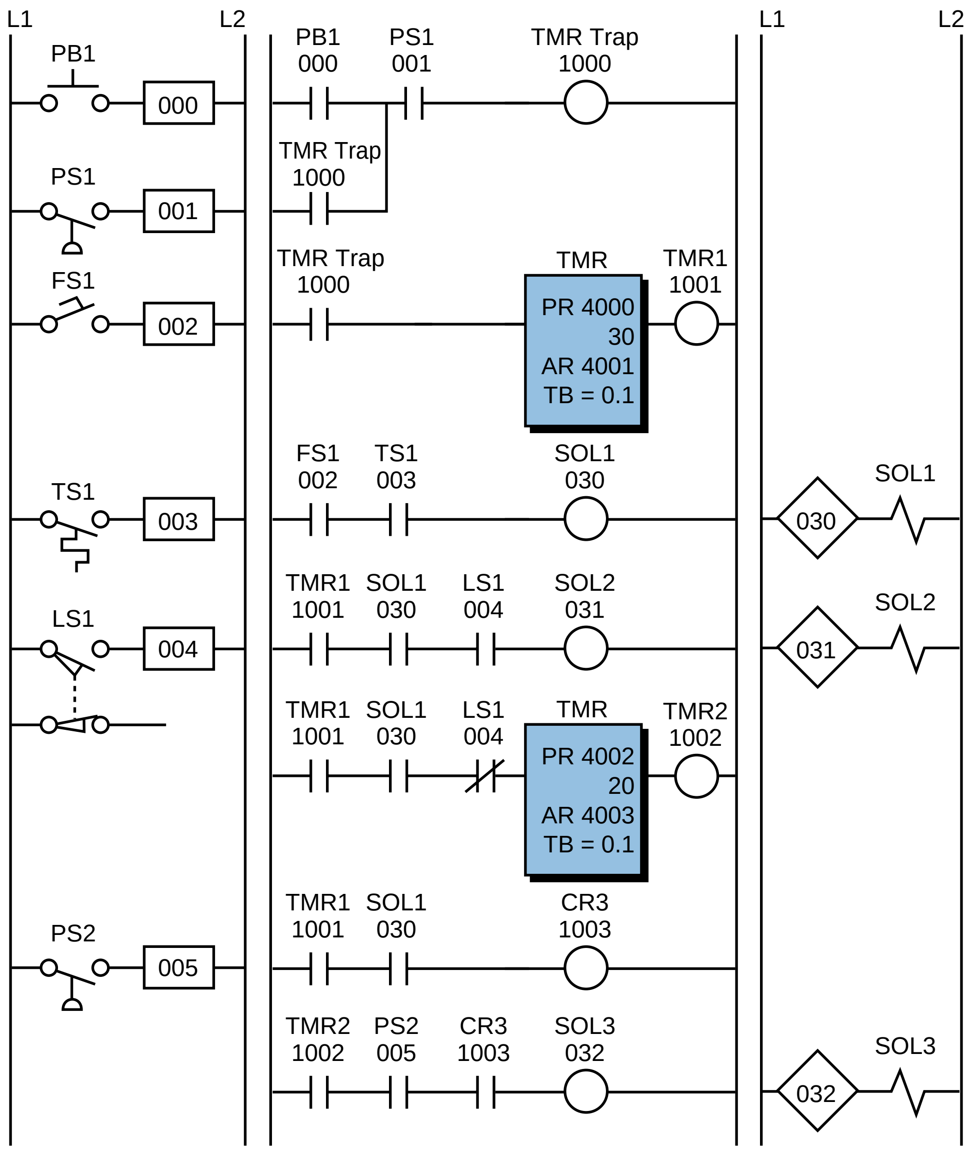
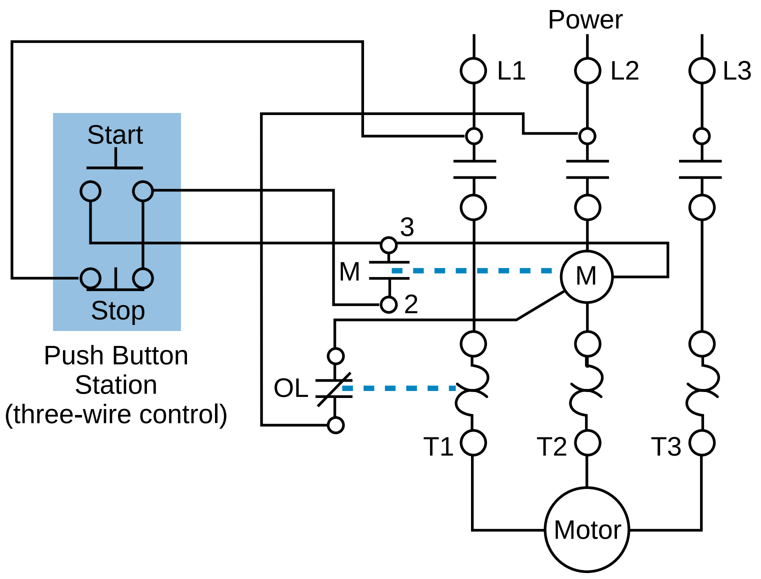


Figure 24. PLC implementation of the circuit in Figure 23.

Simple Start/Stop Motor Circuit

Figure 25 shows the wiring diagram for a three-phase motor and its corresponding three-wire control circuit, where the auxiliary contacts of the starter seal the start push button. To convert this circuit into a PLC program, first determine which control devices will be part of the PLC I/O system; these are the circled items in Figure 26. In this circuit, the start and stop push buttons (inputs) and the starter coil (output) will be part of the PLC system. The starter coil's auxiliary contacts will not be part of the system because an internal will be used to seal the coil, resulting in less wiring and fewer connections.

(a)



(b)

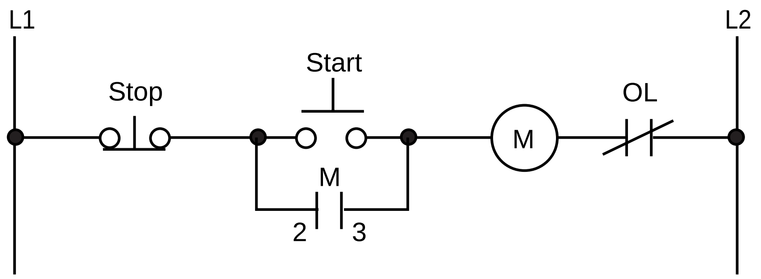


Figure 25. (a) Wiring diagram and **(b)** relay control circuit for a three-phase motor.

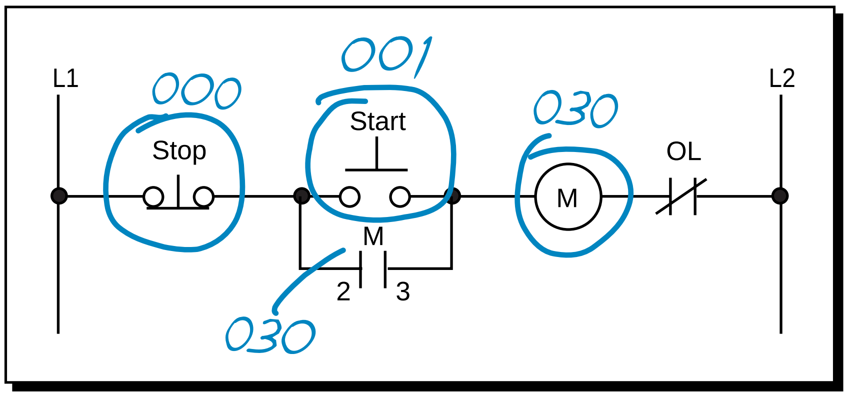


Figure 26. Real inputs and outputs to the PLC.

Table 11 shows the I/O address assignment, which uses the same addressing scheme as the circuit diagram (i.e., inputs: addresses 000 and 001, output: address 030).

To program the PLC, the devices must be programmed in the same logic sequence as they are in the hardwired circuit (see Figure 27). Therefore, the stop push button will be programmed as an examine-ON instruction

	I/O Address			
Module Type	**Rack**	**Group**	**Terminal**	**Description**
Input	0	0	0	Stop PB (NC)
	0	0	1	Start PB
	0	0	2	—
	0	0	3	—
Output	0	3	0	Motor M1
	0	3	1	—
	0	3	2	—
	0	3	3	—

Table 11. I/O address assignment.

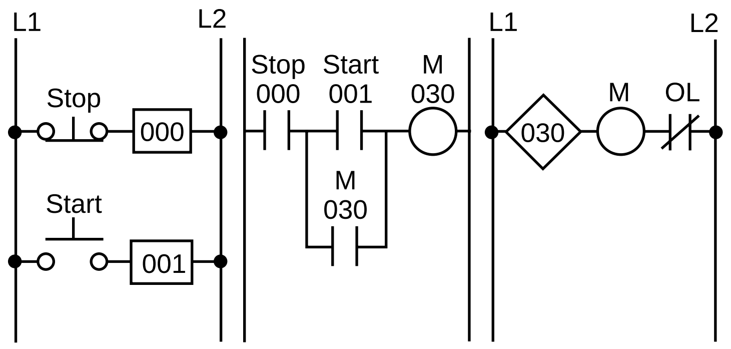


Figure 27. PLC implementation of the circuit in Figure 25.

(a normally open PLC contact) in series with the start push button, which is also programmed as an examine-ON instruction. This circuit will drive output 030, which controls the starter. If the start push button is pressed, output 030 will turn ON, sealing the start push button and turning the motor ON through the starter. If the stop push button is pressed, the motor will turn OFF. Note that the stop push button is wired as normally closed to the input module. Also, the starter coil's overloads are wired in series with the coil.

In a PLC wiring diagram, the PLC is connected to power lines L1 and L2 (see Figure 28). The field inputs are connected to L1 on one side and to the module on the other. The common, or return, connection from the input module goes to L2. The output module receives its power for switching the load from L1. Output terminal 030 is connected in series with the starter coil and its overloads, which go to L2. The output module also directly connects to L2 for proper operation. Note that, in the motor control circuit's wiring diagram (see Figure 29), the PLC output module is wired directly to the starter coil.

Although the three-phase motor has a three-wire control circuit, its corresponding PLC control circuit has only two wires. This two-wire configuration is similar to a three-wire configuration because it provides low-voltage release; however, it does not provide low-voltage protection. Referring to

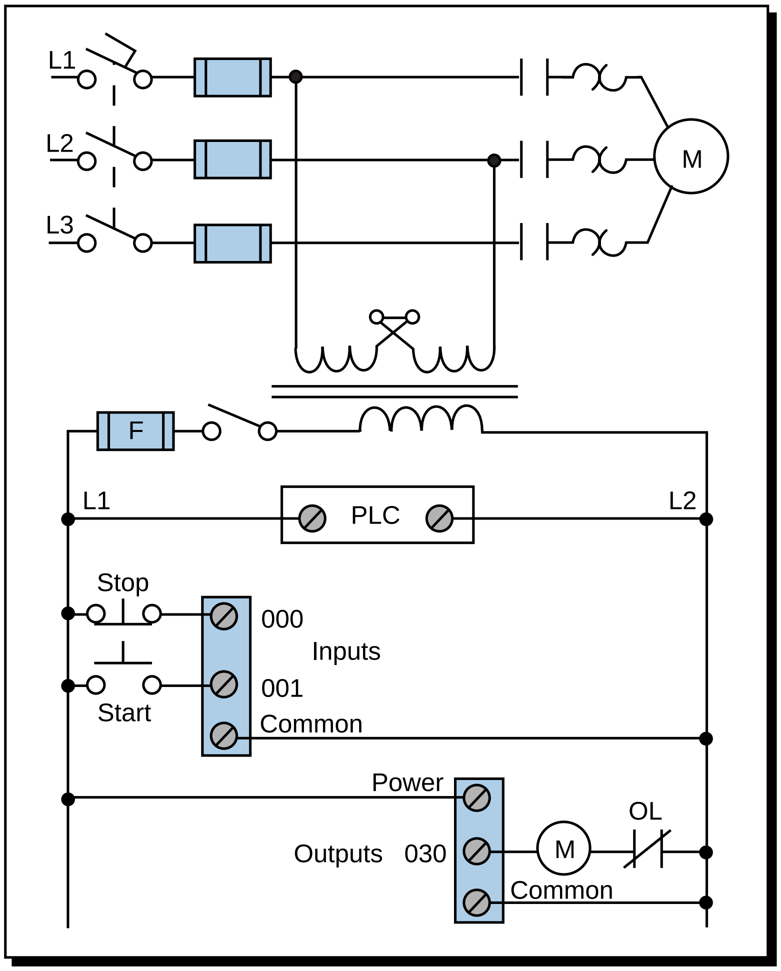


Figure 28. PLC wiring diagram of a three-phase motor.

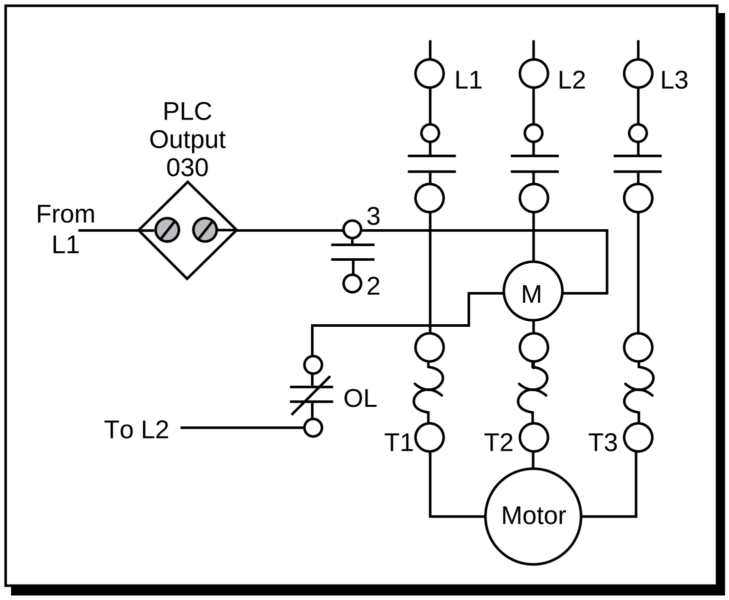


Figure 29. Motor control circuit's wiring diagram.

Figure 29, the starter's seal-in contacts (labeled as 3—||—2) are not used and are shown as unconnected. If the motor is running and the overloads open, the motor will stop, but the circuit will still be ON. Once the overloads cool off and the overload contacts close, the motor will start again immediately. Depending on the application, this situation may not be desirable. For example, someone may be troubleshooting the motor stoppage and the motor may suddenly restart. Making the auxiliary contact an input and using its address to seal the start push button can avoid this situation by making the two-wire circuit act as a three-wire circuit (see Figure 30). In this configuration, if the overloads open while the motor is running, the coil will turn off and their auxiliary contacts will break the circuit in the PLC.

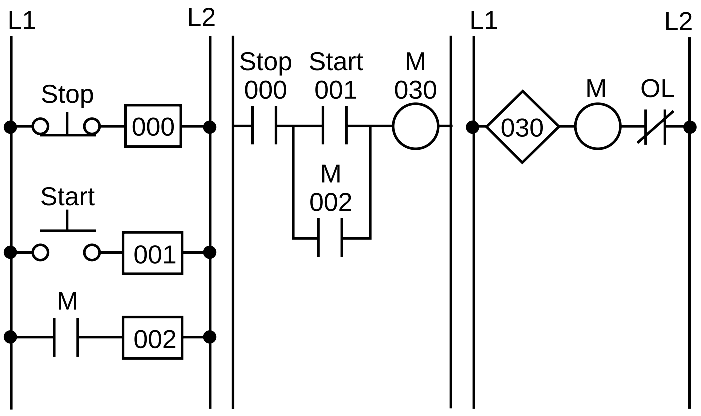


Figure 30. Two-wire circuit configured as a three-wire circuit.

Forward/Reverse Motor Interlocking

Figure 31 illustrates a hardwired forward/reverse motor circuit with electrical and push button interlockings. Figure 32 shows the simplified wiring diagram for this motor. The PLC implementation of this circuit should

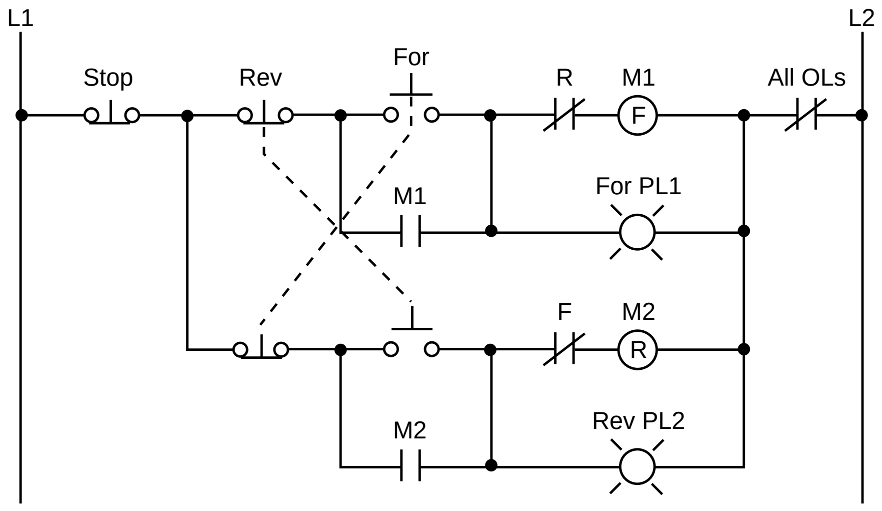


Figure 31. Hardwired forward/reverse motor circuit.

include the use of the overload contacts to monitor the occurrence of an overload condition. The auxiliary starter contacts (M1 and M2) are not required in the PLC program because the sealing circuits can be programmed using the internal contacts from the motor outputs. Low-voltage protection can be implemented using the overload contact input so that, if an overload occurs, the motor circuit will turn off. However, after the overload condition passes, the operator must push the forward or reverse push button again to restart the motor.

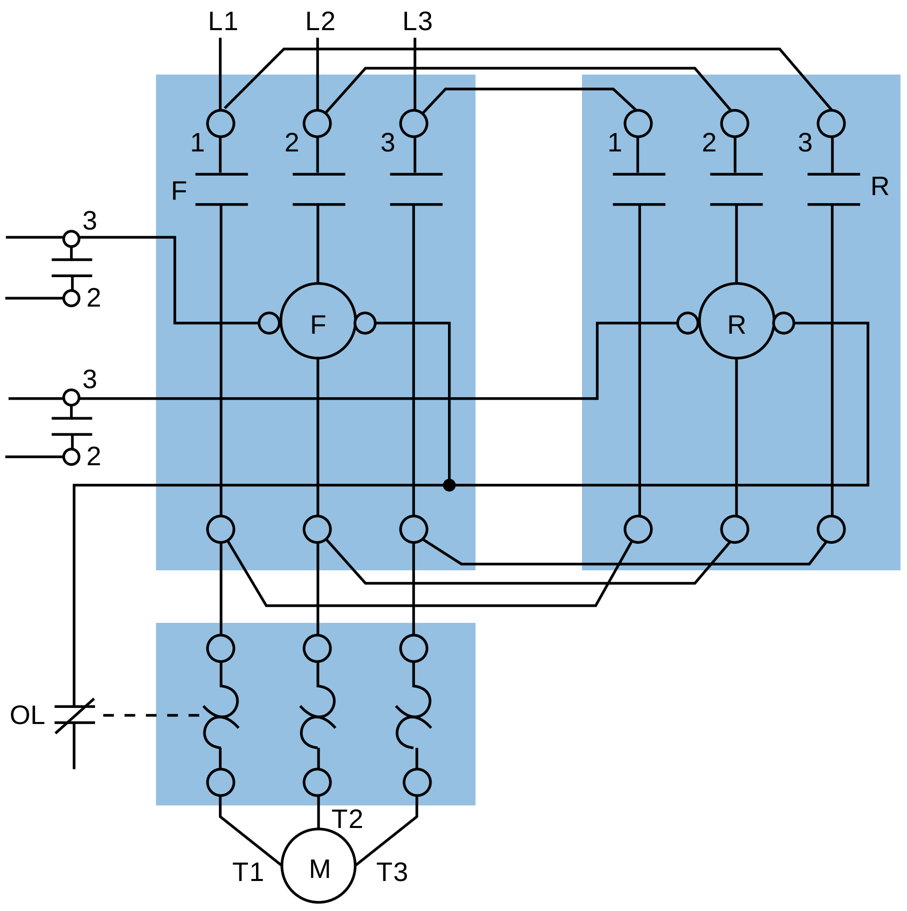


Figure 32. Forward/reverse motor wiring diagram.

For simplicity, the PLC implementation of the circuit in Figure 31 includes all of the elements in the hardwired diagram, even though the additional starter contacts (normally closed R and F in the hardwired circuit) are not required, since the push button interlocking accomplishes the same task. In the hardwired circuit, this redundant interlock is performed as a backup interlocking procedure.

Figure 33 shows the field devices that will be connected to the PLC. The stop push button has address 000, while the normally open sides of the forward and reverse push buttons have addresses 001 and 002, respectively. The overload contacts are connected to the input module at address 003. The output

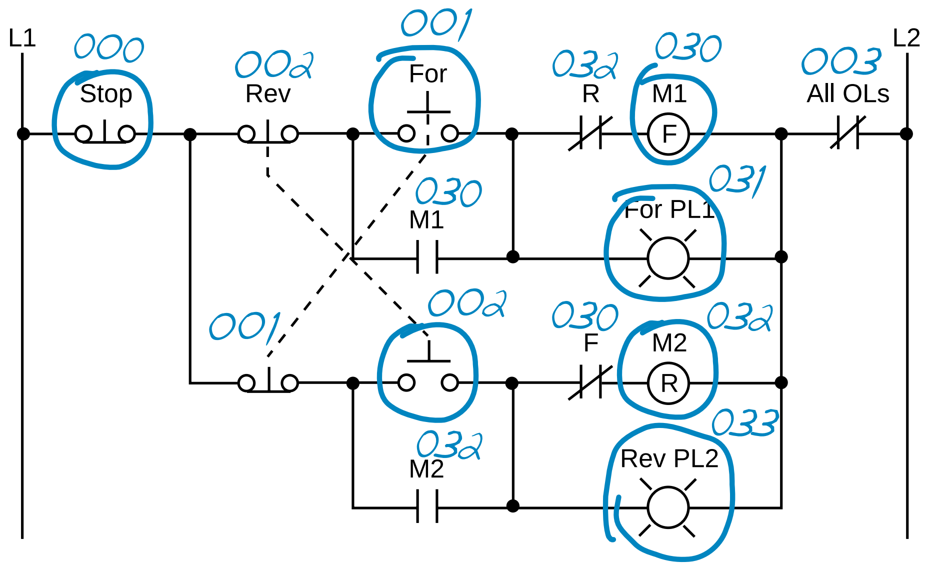


Figure 33. Real inputs and outputs to the PLC.

devices—the forward and reverse starters and their respective interlocking auxiliary contacts—have addresses 030 and 032. The forward and reverse pilot light indicators have address 031 and 033, respectively. Additionally, the overload light indicators have addresses 034 and 035, indicating that the overload condition occurred during either forward or reverse motor operation. The addresses for the auxiliary contact interlocking using the R and F contacts are the output addresses of the forward and reverse starters (030 and 032). The ladder circuit that latches the overload condition (forward or reverse) must be programmed before the circuits that drive the forward and reverse starters as we will explain shortly. Otherwise, the PLC program will never recognize the overload signal because the starter will be turned off in the circuit during the same scan when the overload occurs. If the latching circuit is after the motor starter circuit, the latch will never occur because the starter contacts will be open and continuity will not exist.

Table 12 shows the real I/O address assignment for this circuit. Figure 34 shows the PLC implementation, which follows the same logic as the hardwired circuit and adds additional overload contact interlockings. Note that the motor circuit also uses the overload input, which will shut down the motor. The normally closed overload contacts are programmed as normally open in the logic driving the motor starter outputs. The forward and reverse motor commands will operate normally if no overload condition exists because the overload contacts will provide continuity. However, if an overload occurs, the contacts in the PLC program will open and the motor circuit will turn OFF. The overload indicator pilot lights (OL Fault Fwd and OL Fault Rev) use latch/unlatch instructions to latch whether the overload occurred in the forward or reverse operation. Again, the latching occurs before the forward and reverse motor starter circuits, which will turn off due

Module Type	I/O Address: Rack	I/O Address: Group	I/O Address: Terminal	Description
Input	0	0	0	Stop PB (wired NC)
	0	0	1	Forward PB (wired NO)
	0	0	2	Reverse PB (wired NO)
	0	0	3	Overload contacts
Input	0	0	4	Acknowledge OL/Reset PB
	•	•	•	
	•	•	•	
	•	•	•	
Output	0	3	0	Motor starter M1 (FWD)
	0	3	1	Forward PL1
	0	3	2	Motor starter M2 (REV)
	0	3	3	Reverse PL2
Output	0	3	4	Overload condition FWD
	0	3	5	Overload condition REV
	0	3	6	—
	0	3	7	—

Table 12. I/O address assignment.

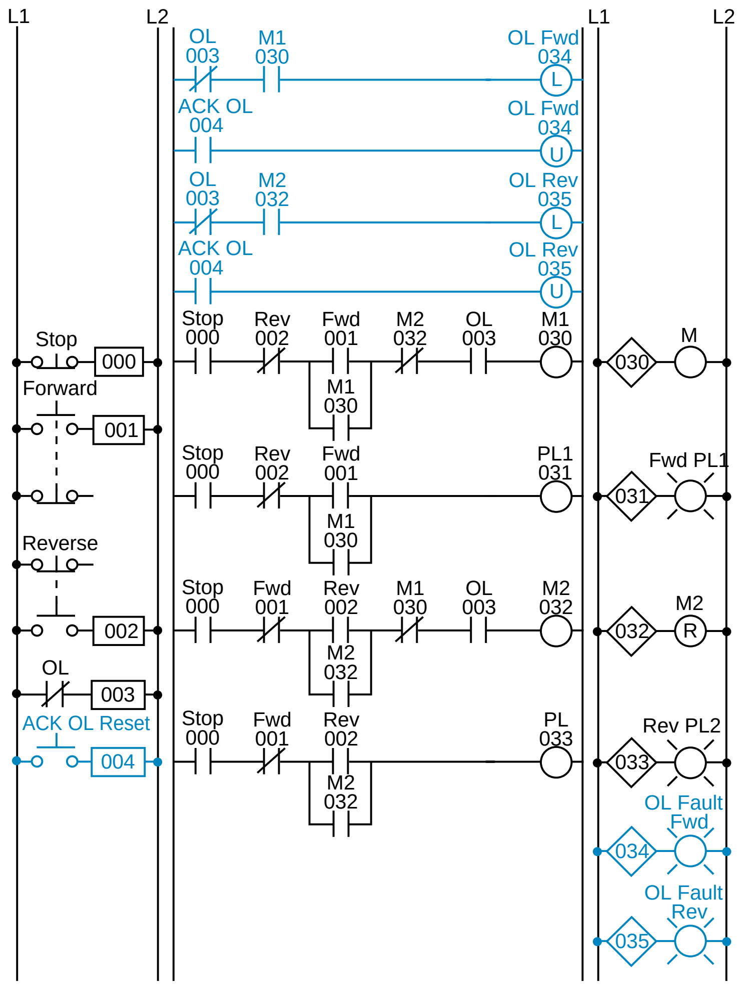


Figure 34. PLC implementation of the circuit in Figure 31.

to the overload. An additional normally open acknowledge overload reset push button, which is connected to the input module, allows the operator to reset the overload indicators. Thus, the overload indicators will remain latched, even if the physical overloads cool off and return to their normally closed states, until the operator acknowledges the condition and resets it.

Figure 35 illustrates the motor wiring diagram of the forward/reverse motor circuit and the output connections from the PLC. Note that the auxiliary contacts M1 and M2 are not connected. In this wiring diagram, both the forward and reverse coils have their returns connected to L2 and not to the overload contacts. The overload contacts are connected to L1 on one side and to the PLC's input module on the other (input 003). In the event of an overload, both motor starter output coils will be dropped from the circuit because the PLC's output to both starters will be OFF.

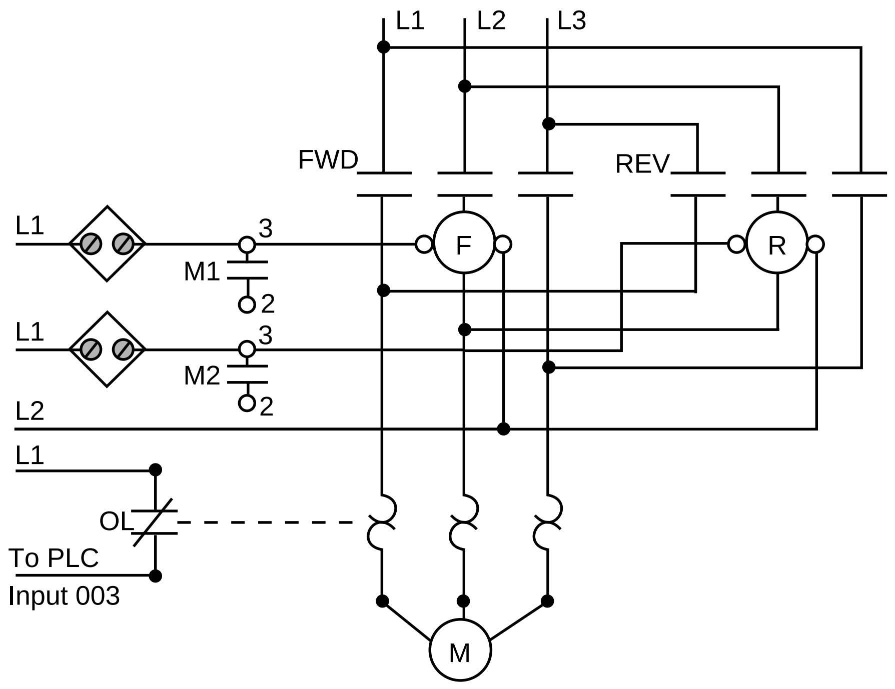
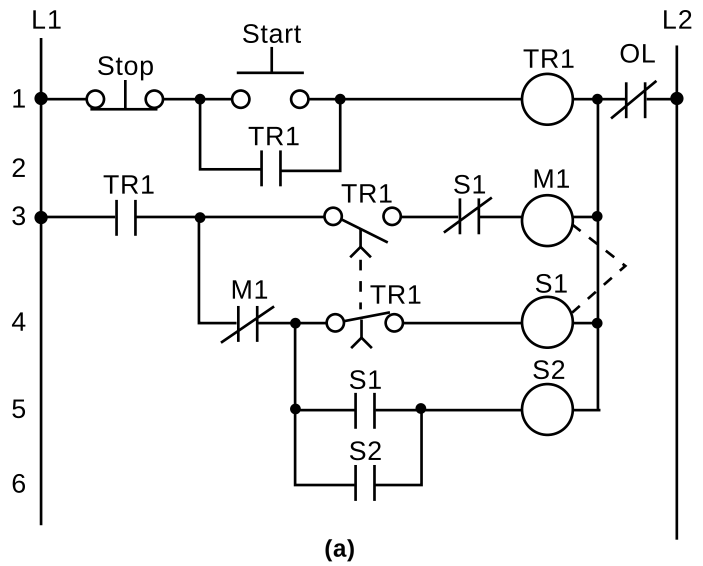


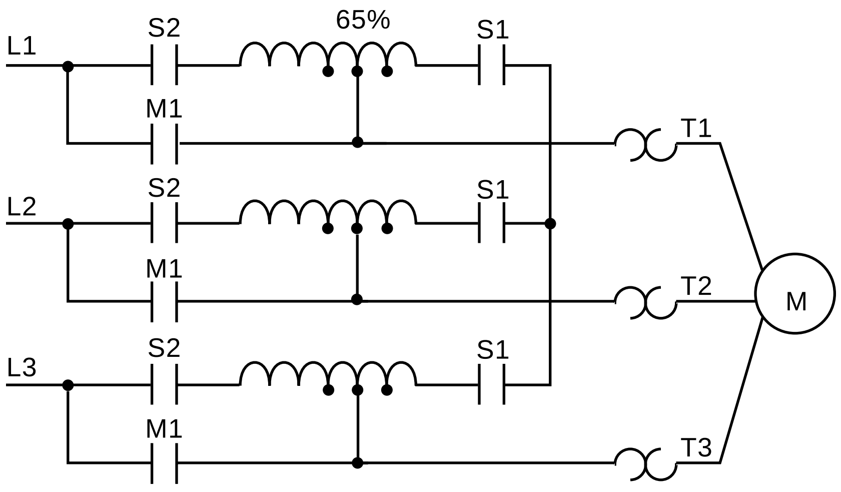
Figure 35. Forward/reverse motor wiring diagram.

Reduced-Voltage-Start Motor Control

Figure 36 illustrates the control circuit and wiring diagram of a 65% tapped, autotransformer, reduced-voltage-start motor control circuit. This reduced-voltage start minimizes the inrush current at the start of the motor (locked-rotor current) to 42% of that at full speed. In this example, the timer must be set to 5.3 seconds. Also, the instantaneous contacts from the timer in lines 2 and 3 must be trapped.



(a)



(b)

Figure 36. (a) Hardwired relay circuit and **(b)** wiring diagram of a reduced-voltage-start motor.

Figure 37 illustrates the hardwired circuit with the real inputs and outputs circled. The devices that are not circled are implemented inside the PLC through the programming of internal instructions. Tables 13, 14, and 15 show the I/O assignment, internal assignment, and register assignment, respectively. Figure 38 illustrates the PLC implementation of the reduced-voltage-start circuit. The first line of the PLC program traps the timer with internal output 1000. Contacts from this internal replace the instantaneous timer contacts specified in the hardwired control circuit. This PLC circuit implementation does not provide low-voltage protection, since the interlocking does not use the physical inputs of M1, S1, and S2. If low-voltage protection is required, then the starter's auxiliary contacts or the overload contacts can be programmed as described in the previous examples. If the auxiliary contacts or the overloads are used as inputs, they must be programmed as

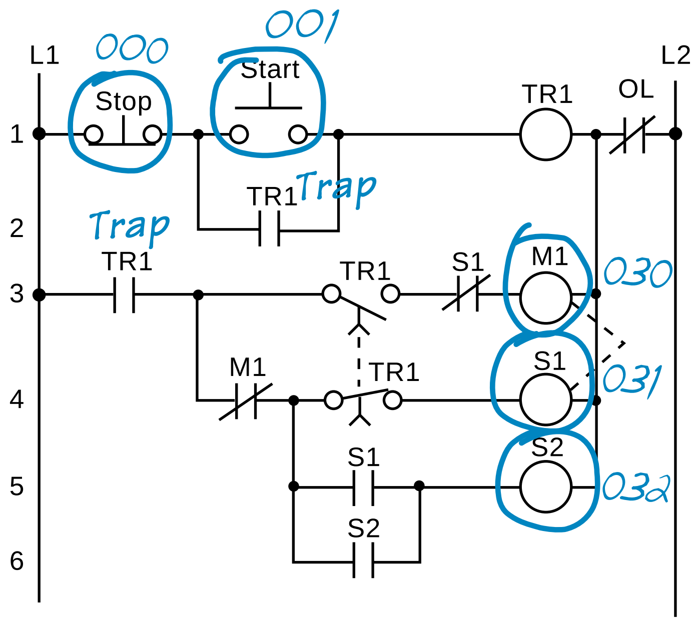


Figure 37. Real inputs and outputs to the PLC.

	I/O Address			
Module Type	Rack	Group	Terminal	Description
Input	0 0	0 0	0 1	Stop PB (NC) Start PB (NO)
Output	0 0 0	3 3 3	0 1 2	Motor Starter M1 S1 S2

Table 13. I/O address assignment.

Device	Internal	Description
—	1000	Trap timer circuit
Timer	1001	Timer

Table 14. Internal address assignment.

Register	Description
4000	Preset register value 53, time base 0.1 sec for 5.3 sec (timer output is 1001)
4001	Accumulated register for timer output 1001

Table 15. Register assignment.

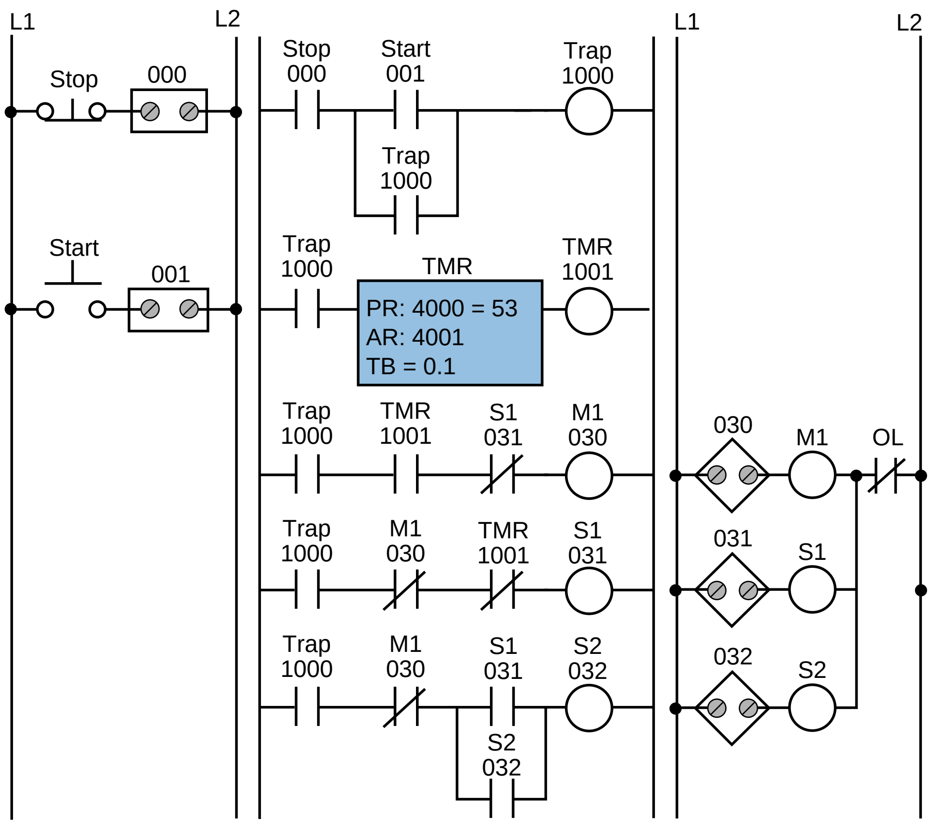


Figure 38. PLC implementation of the circuit in Figure 36.

normally open (closed when the overloads are closed and the motor is running) and placed in series with contact 1000 in line 3 of the PLC program. If the overloads open, the circuit will lose continuity and M1 will turn OFF.

AC Motor Drive Interface

A common PLC application is the speed control of AC motors with variable speed (VS) drives. The diagram in Figure 39 shows an operator station used to manually control a VS drive. The programmable controller implementation of this station will provide automatic motor speed control through an analog interface by varying the analog output voltage (0 to 10 VDC) to the drive.

The operator station consists of a speed potentiometer (speed regulator), a forward/reverse direction selector, a run/jog switch, and start and stop push buttons. The PLC program will contain all of these inputs except the potentiometer, which will be replaced by an analog output. The required input field devices (i.e., start push button, stop push button, jog/run, and forward/reverse) will be added to the application and connected to input modules, rather than using the operator station's components. The PLC program will contain the logic to start, stop, and interlock the forward/reverse commands.

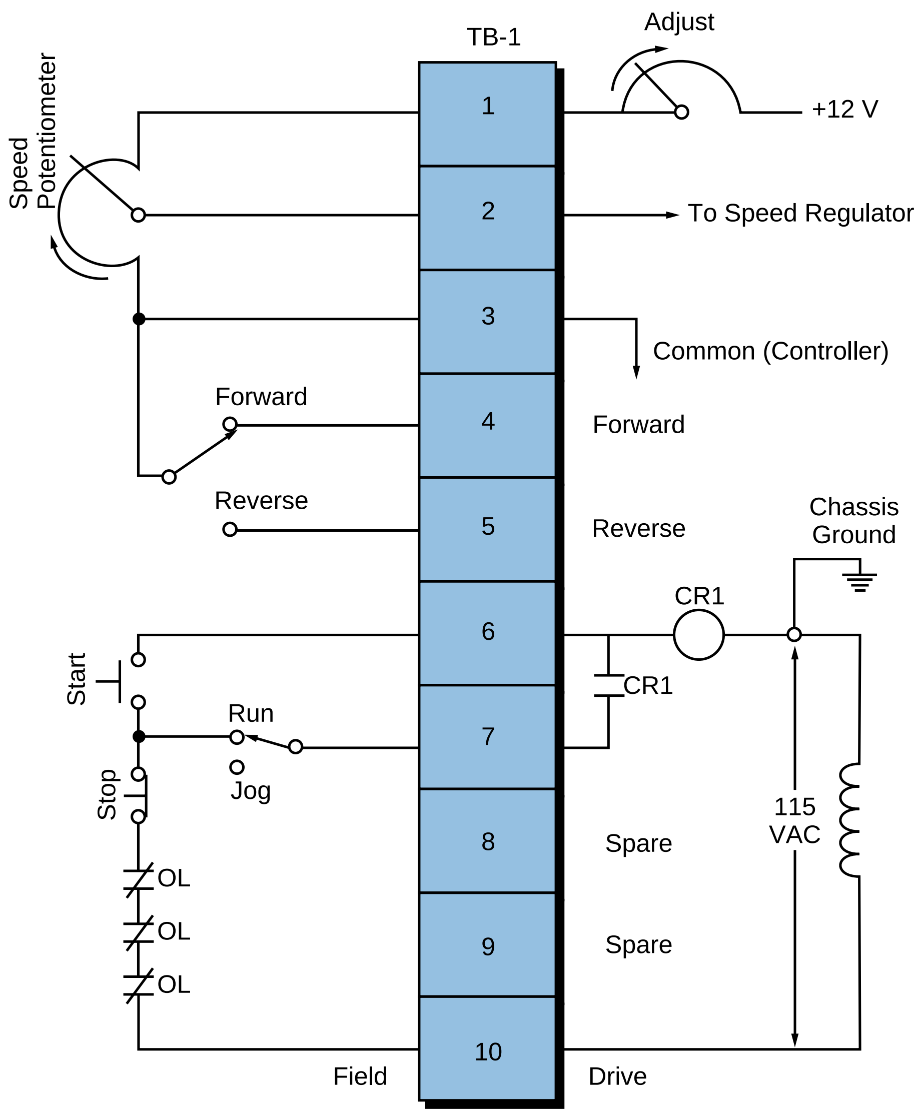


Figure 39. Operator station for a variable speed drive.

Table 16 shows the I/O address assignment table for this example, while Figure 40 illustrates the connection diagram from the PLC to the VS drive's terminal block (TB-1). The connection uses a contact output interface to switch the forward/reverse signal, since the common must be switched. To activate the drive, terminal TB-1-6 must receive 115 VAC to turn ON the internal relay CR1. The drive terminal block TB-1-8 supplies power to the PLC's L1 connection to turn the drive ON. The output of the module (CR1) is connected to terminal TB-1-6. The drive's 115 VAC signal is used to control the motor speed so that the signal is in the same circuit as the drive, avoiding the possibility of having different commons (L2) in the drive (the start/stop common is not the same as the controller's common). In this configuration, the motor's overload contacts are wired to terminals TB-1-9 and TB-1-10, which are the drive's power (L1) connection and the output interface's L1 connection. If an overload occurs, the drive will turn OFF

Module Type	I/O Address: Rack	I/O Address: Group	I/O Address: Terminal	Description
Input	0	0	0	Start
	0	0	1	Stop
	0	0	2	Forward/reverse selector
	0	0	3	Run/jog selector
Output 115 VAC	0	3	0	Drive enable (L1 from drive)
	0	3	1	
	0	3	2	
	0	3	3	
Output Contact	0	3	4	Forward
	0	3	5	Reverse
	0	3	6	
	0	3	7	
Analog Output	0	7	0	Analog speed reference 0–10 VDC
	0	7	1	
	0	7	2	
	0	7	3	

Table 16. I/O address assignment.

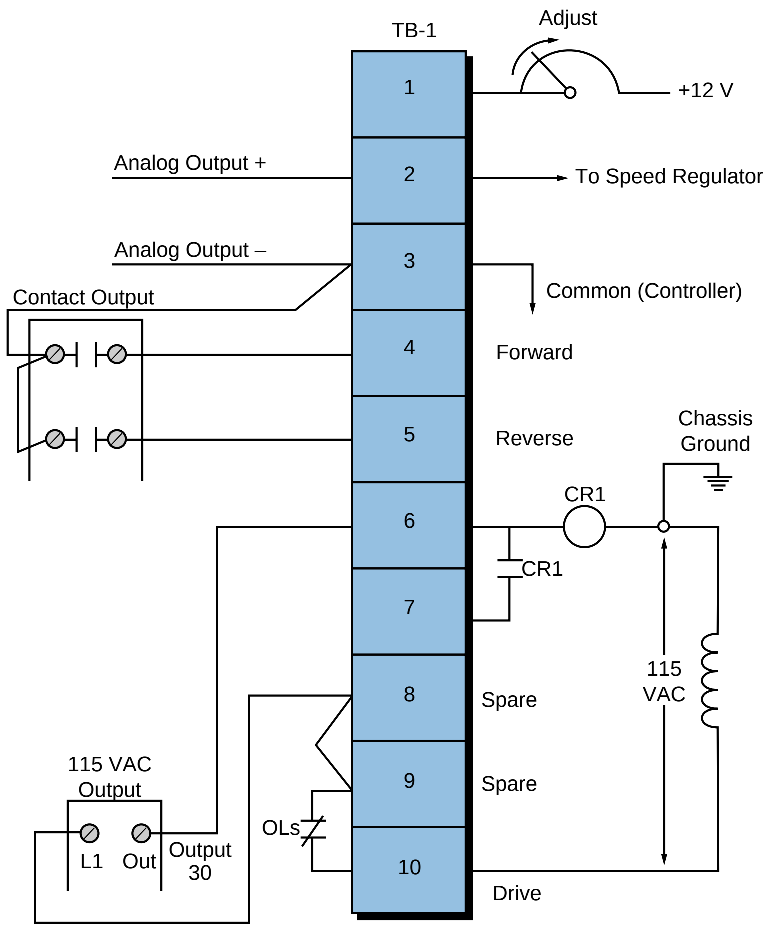


Figure 40. Connection diagram from the PLC to the VS drive's terminal block.

because the drive's CR1 contact will not receive power from the output module. This configuration, however, does not provide low-voltage protection, since the drive and motor will start immediately after the overloads cool off and reclose. To have low-voltage protection, the auxiliary contact from the drive, CR1 in terminal TB-1-7, must be used as an input in the PLC, so that it seals the start/stop circuit.

Figure 41 shows the PLC ladder program that will replace the manual operator station. The forward and reverse inputs are interlocked, so only one of them can be ON at any given time (i.e., they are mutually exclusive). If the jog setting is selected, the motor will run at the speed set by the analog output when the start push button is depressed. The analog output connection simply allows the output to be enabled when the drive starts. Register 4000 holds the value in counts for the analog output to the drive. Internal 1000, which is used in the block transfer, indicates the completion of the instruction.

Sometimes, a VS drive requires the ability to run under automatic or manual control (AUTO/MAN). Several additional hardwired connections must be made to implement this dual control. The simplest and least expensive way to do this is with a selector switch (e.g., a four-pole, single-throw, single-break selector switch). With this switch, the user can select either the automatic or manual option. Figure 42 illustrates this connection. Note that

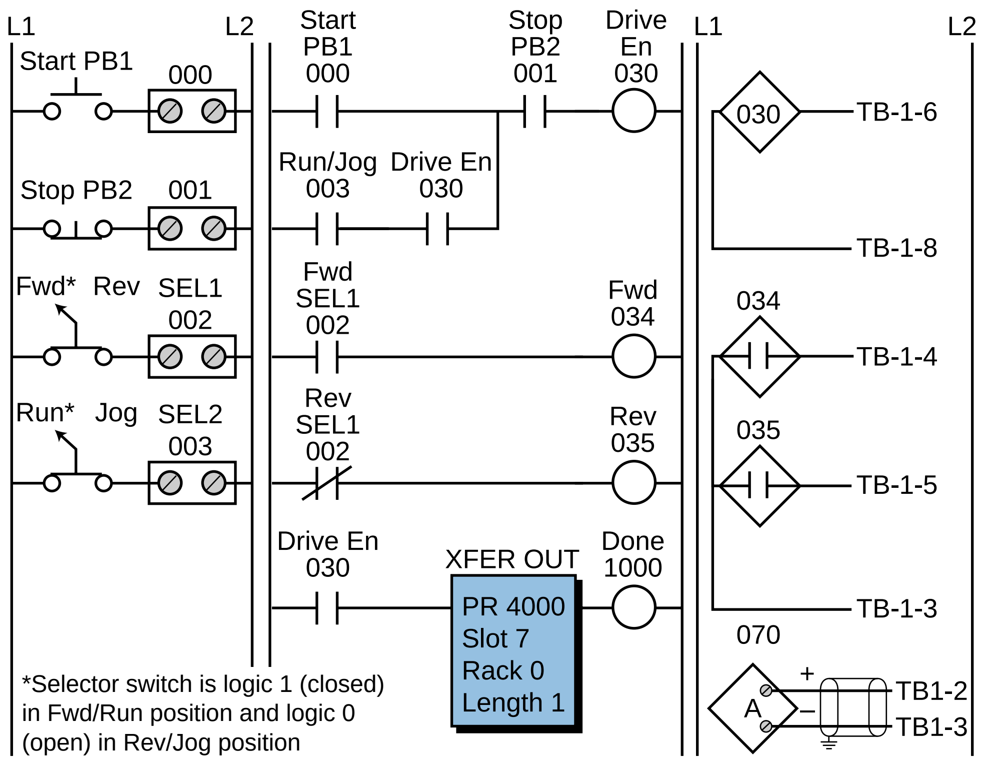


Figure 41. PLC implementation of the VS drive.

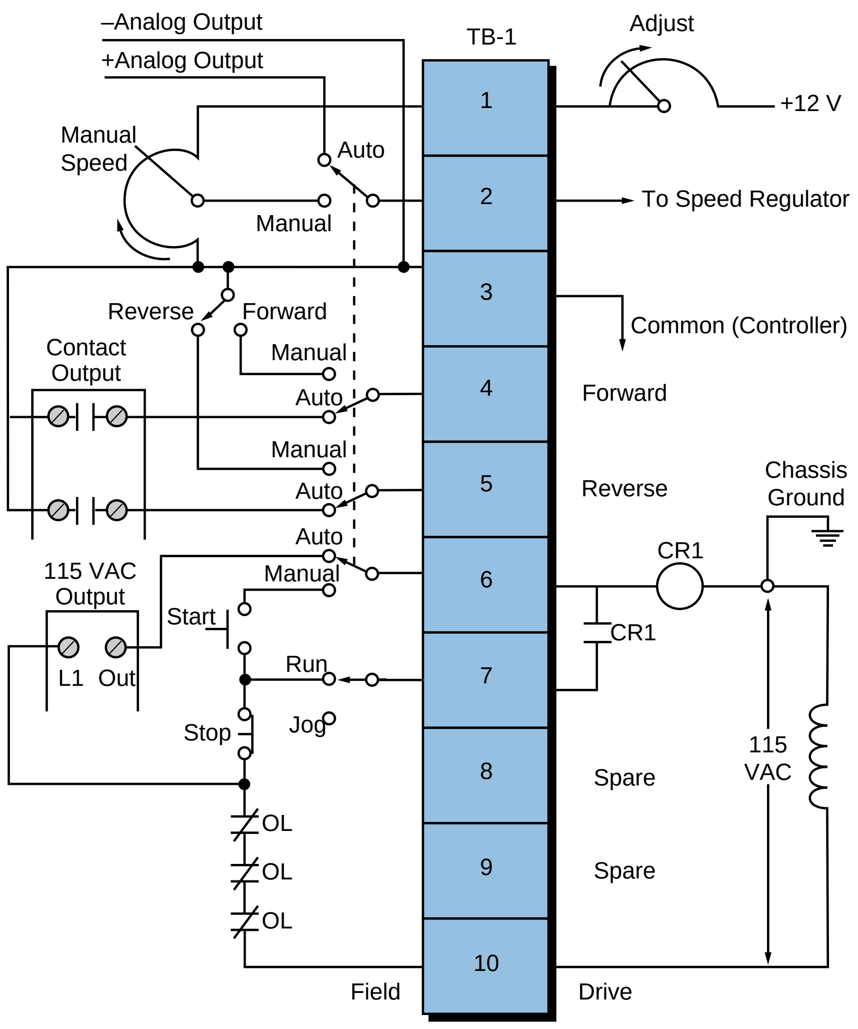


Figure 42. VS drive with AUTO/MAN capability.

the start, stop, run/jog, potentiometer, and forward/reverse field devices shown are from the operator station. These devices are connected to the PLC interface under the same names that are used in the control program (refer to Figure 41). If the AUTO/MAN switch is set to automatic, the PLC will control the drive; if the switch is set to manual, the manual station will control the drive.

Continuous Bottle-Filling Control

In this example (see Figure 43), we will implement a control program that detects the position of a bottle via a limit switch, waits 0.5 seconds, and then fills the bottle until a photosensor detects a filled condition. After the bottle is filled, the control program will wait 0.7 seconds before moving to the next bottle. The program will include start and stop circuits for the outfeed motor and the start of the process. Table 17 shows the I/O address assignment, while Tables 18 and 19 present the internal and register assignments, respectively. These assignments include the start and stop process signals.

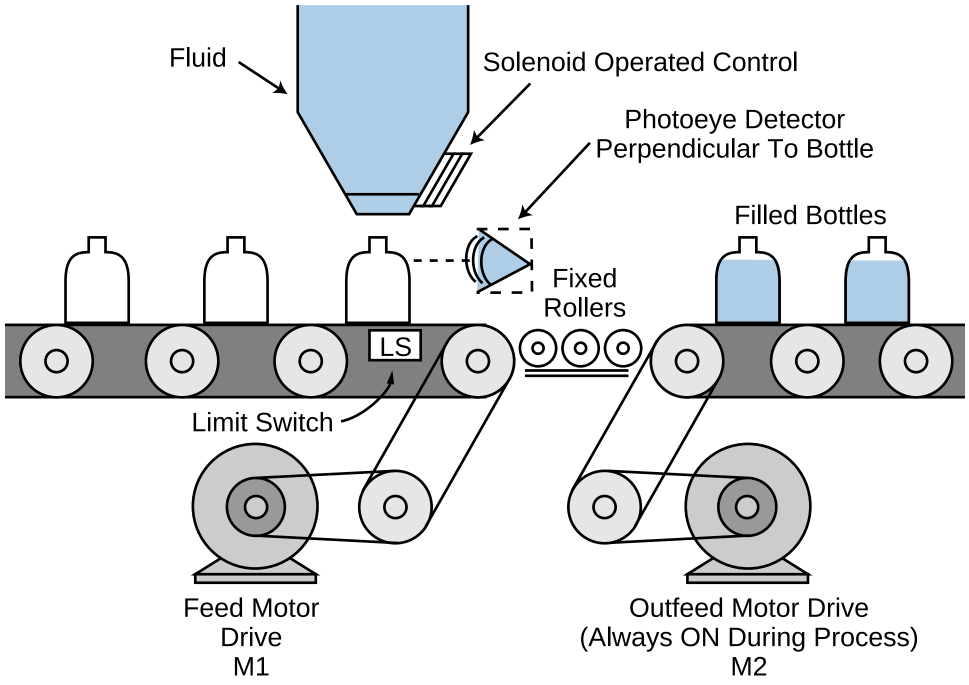

Fluid
Solenoid Operated Control
Photoeye Detector Perpendicular To Bottle
Filled Bottles
Fixed Rollers
LS
Limit Switch
Feed Motor Drive M1
Outfeed Motor Drive (Always ON During Process) M2

Figure 43. Bottle-filling system.

Module Type	I/O Address Rack	Group	Terminal	Description
Input	0	0	0	Start process PB1
	0	0	1	Stop process PB2 (NC)
	0	0	2	Limit switch (position detect)
	0	0	3	Photoeye (level detect)
Output	0	3	0	Feed motor M1
	0	3	1	Outfeed motor M2 (system ON)
	0	3	2	Solenoid control
	0	3	3	—

Table 17. I/O address assignment.

Device	Internal	Description
Timer	1001	Timer for 0.5 sec delay after position detect
Timer	1002	Timer for 0.7 sec delay after level detect
—	1003	Bottle filled, timed out, feed motor M1

Table 18. Internal output assignment.

Register	Description
4000	Preset value 5, time base 0.1 sec (1001)
4001	Accumulated value for 1001
4002	Preset value 7, time base 0.1 sec (1002)
4003	Accumulated value for 1002

Table 19. Register assignment.

Figure 44 illustrates the PLC ladder implementation of the bottle-filling application. Once the start push button is pushed, the outfeed motor (output 031) will turn ON until the stop push button is pushed. The feed motor M1 will be energized once the system starts (M2 ON); it will stop when the limit switch detects a correct bottle position. When the bottle is in position and 0.5 seconds have elapsed, the solenoid (032) will open the filling valve and remain ON until the photoeye (PE) detects a proper level. The bottle will remain in position for 0.7 seconds, then the energized internal 1003 will start the feed motor. The feed motor will remain ON until the limit switch detects another bottle.

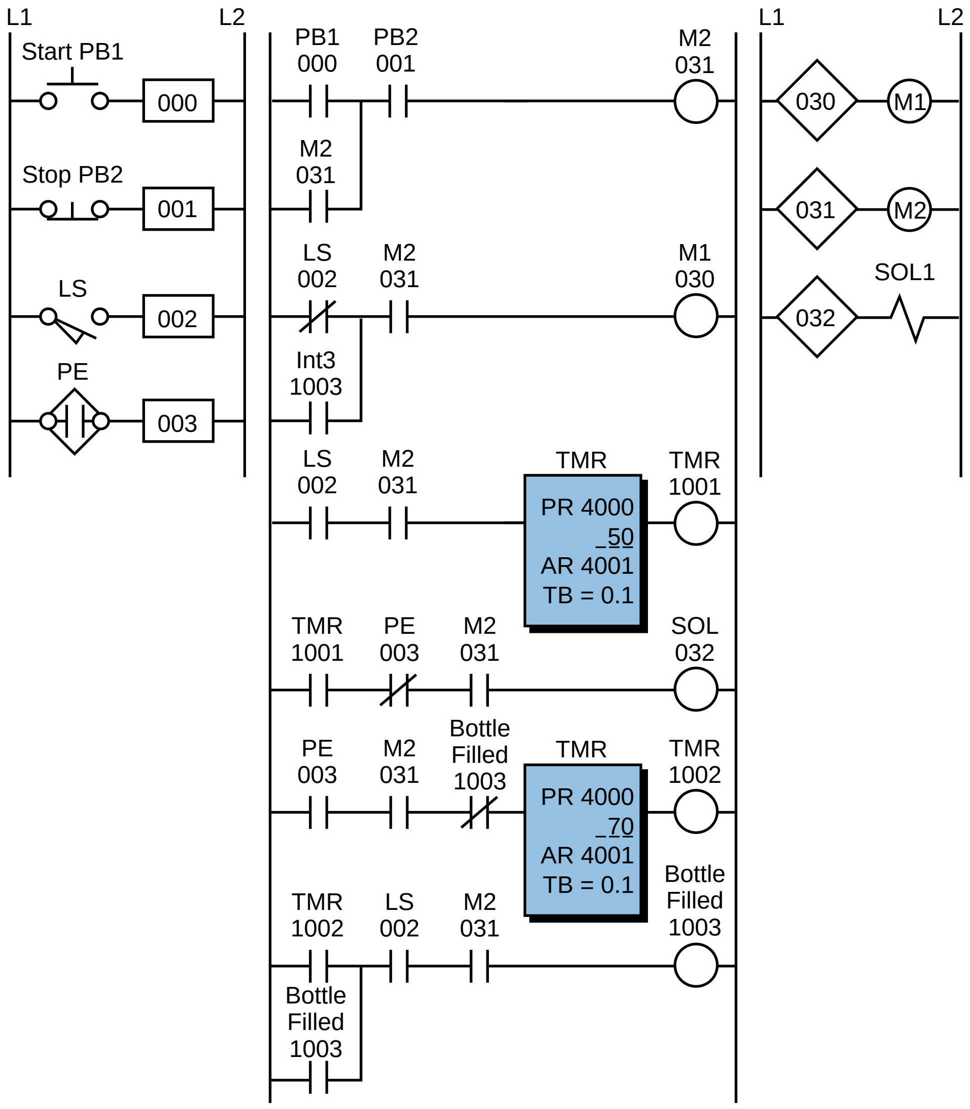


Figure 44. PLC implementation of the bottle-filling application.

Large Relay System Modernization

This example presents the modernization of a machine control system that will be changed from hardwired relay logic to PLC programmed logic. The field devices to be used will remain the same, with the exception of those that the controller can implement (e.g., timers, control relays, interlocks, etc.). The benefits of modernizing the control of this machine are:

- a more reliable control system
- less energy consumption
- less space required for the control panel
- a flexible system that can accommodate future expansion

Figure 45 illustrates the relay ladder diagram that presently controls the logic sequence for this particular machine. For the sake of simplicity, the diagram shows only part of the total relay ladder logic.

An initial review of the relay ladder diagram indicates that certain portions of the logic should be left hardwired—lines 1, 2, and 3. This will keep all emergency stop conditions independent of the controller. The hydraulic pump motor (M1), which is energized only when the master start push button is pushed (PB1), should also be left hardwired. Figure 46 illustrates these hardwired elements. Note that the safety control relay (SCR) will provide power to the rest of the system if M1 is operating properly and no emergency push button is depressed. Furthermore, the PLC fault contact can be placed in series with the emergency push buttons and also connected to a PLC failure alarm. During proper operation, the PLC will energize the fault coil, thus closing PLC Fault Contact 1 and opening PLC Fault Contact 2.

Continuing the example, we can now start assigning the real inputs and outputs to the I/O assignment document. We will assign internal output addresses to all control relays, as well as timers and interlocks from control relays. Tables 20 and 21 present the assignment and description of the inputs and outputs, as well as the internals. Note that inputs with multiple contacts, such as LS4 and SS3, have only one connection to the controller.

Figure 47 shows the PLC program coding (hardwired relay translation) for this example. This ladder program illustrates several special coding techniques that must be used to implement the PLC logic. Among these techniques are the software MCR function, instantaneous contacts from timers, OFF-delay timers, and the separation of rungs with multiple outputs.

An MCR internal output, specified through the program software, performs a function similar to a hardwired MCR. Referring to the relay logic diagram in Figure 45, if the MCR is energized, its contacts will close, allowing power to flow to the rest of the system. In the PLC software, the internal MCR

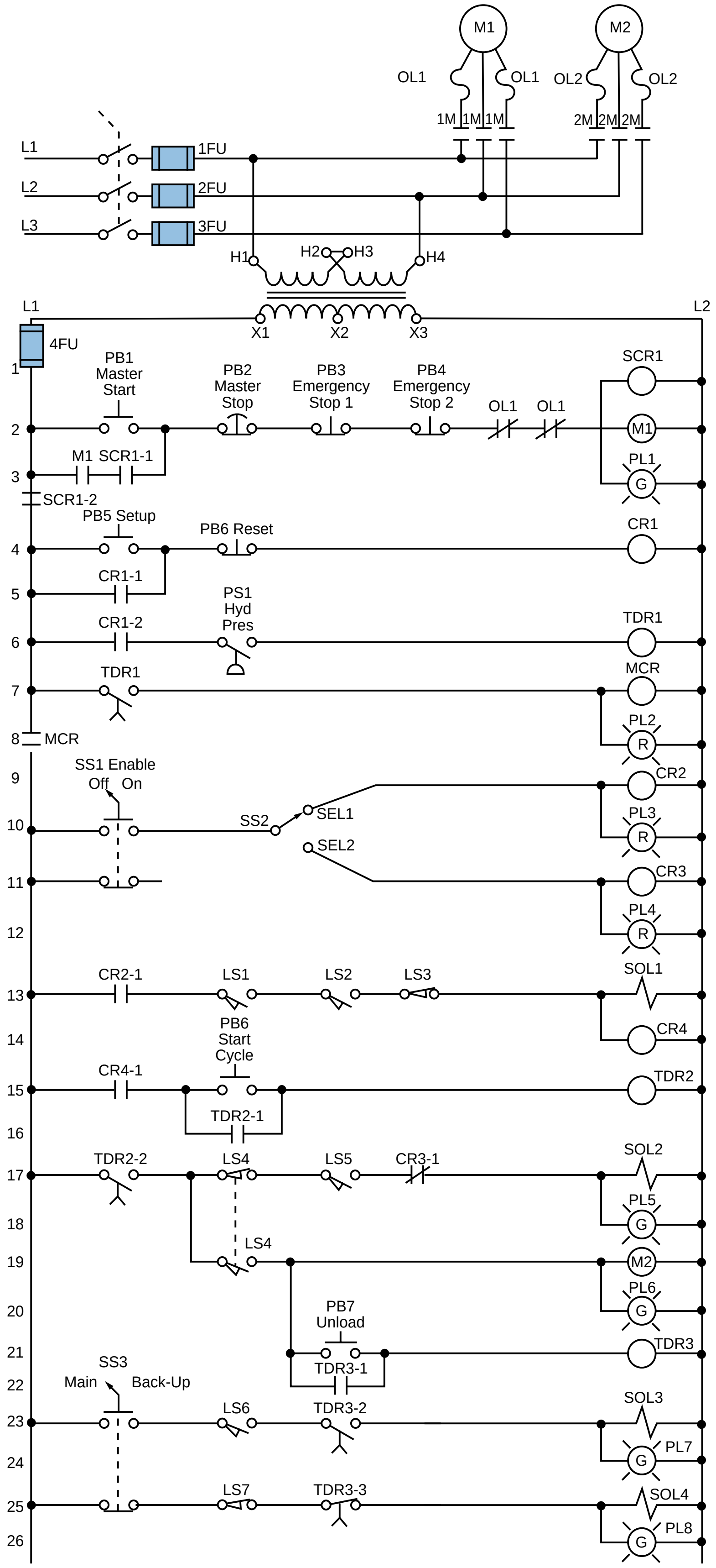


Figure 45. Electromechanical relay diagram.

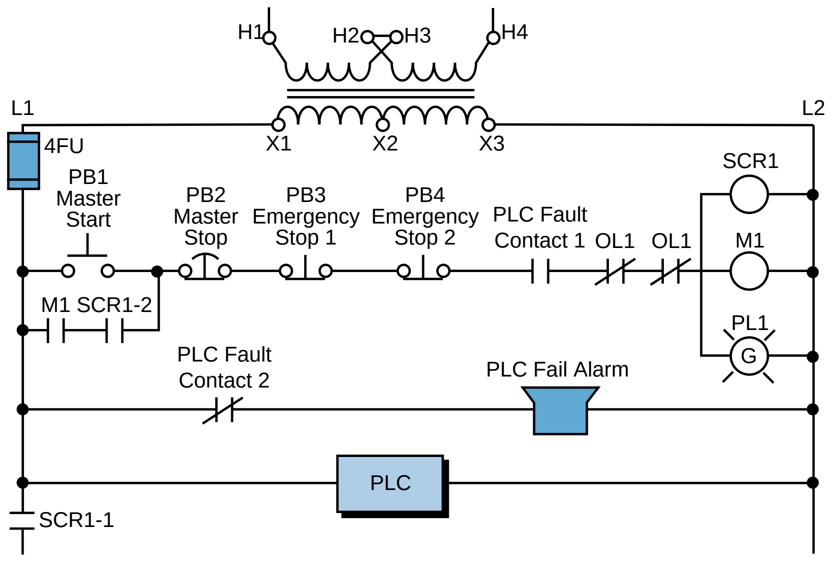


Figure 46. Elements of the moderization example system to be left hardwired.

Module Type	I/O Address Rack	Group	Terminal	Description
Input	0 0 0 0	0 0 0 0	0 1 2 3	PB5—Setup PB PB6—Reset (wired NC) PS1—Hydraulic pressure switch SS1—Enable selector switch (NC contact left unconnected)
Input	0 0 0 0	0 0 0 0	4 5 6 7	SEL1—Select 1 position SEL2—Select 2 position LS1—Limit switch up (position 1) LS2—Limit switch up (position 2)
Input	0 0 0 0	1 1 1 1	0 1 2 3	LS3—Location set PB6—Start load cycle LS4—Trap (wired NC) LS5—Position switch
Input	0 0 0 0	1 1 1 1	4 5 6 7	PB7—Unload PB SS3—Main/backup (wired NO) LS6—Maximum length detect LS7—Minimum length backup
Output	0 0 0 0	3 3 3 3	0 1 2 3	PL2—Setup OK PL3—Select 1 PL4—Select 2 SOL1—Advance forward
Output	0 0 0 0	3 3 3 3	4 5 6 7	SOL2—Engage PL5—Engage ON M2—Run motor PL6—Motor run ON
Output	0 0 0 0	4 4 4 4	0 1 2 3	SOL3—Fast stop PL7—Fast stop ON SOL4—Unload with backup PL8—Backup ON

Table 20. I/O address assignment.

Device	Internal	Description
CR1	1000	CR1 (Setup Rdy)
TDR1	2000	Timer preset 10 sec register 3000 (accumulated register 3001)
MCR	MCR1700	First MCR address
CR2	—	Same as PL3 address
CR3	—	Same as PL4 address
CR4	—	Same as SOL1
—	1001	To set up internal for instantaneous contact of TDR2
TDR2	2001	Timer preset 5 sec register 4002 (accumulated register 4003)
—	1002	To set up internal for instantaneous contact of TDR3
TDR3	2002	Timer preset 12 sec register 4004 (accumulated register 4005)

Table 21. Internal address assignment.

1700 accomplishes this same function (for this example, MCR1700 is the first available address for MCRs). If the MCR coil is not energized, the PLC will not execute the ladder logic that is fenced between the MCR coil and the END MCR instruction.

An internal will not replace the control relay CR2 in line 9 since the PL3 contacts in line 10 can be used instead. This technique can be used whenever a control relay is in parallel with a real output device. Moreover, we do not need to separate the coils in lines 17 and 18 of the hardwired logic. This has already been done, since the PLC used here does not allow rungs with multiple outputs. Using separate rungs for each output is always a good practice.

The normally closed inputs that are connected to the input modules are programmed as normally open, as explained in the previous sections. The limit switch LS4 has two contacts—a normally open one and a normally closed one in lines 17 and 19, respectively, of Figure 45. However, only one set of contacts needs to be connected to the controller. In this example, we have selected the normally closed contact LS4. Although the normally open contact is not connected to the controller, its hardwired function can still be achieved by programming LS4 as a normally closed ladder contact.

Applications such as this one also require timers with instantaneous contacts, which are not available in most PLCs. An instantaneous contact is one that opens or closes when the timer is enabled. In most PLCs, an internal coil is used as a substitute for an instantaneous contact. Line 15 in the hardwired logic shows that, if PB6 is pressed and CR4 is closed, the timer TDR2 will start timing and contact TDR2-1 will seal PB6. This arrangement requires special PLC implementation. If we use software timer contacts, the

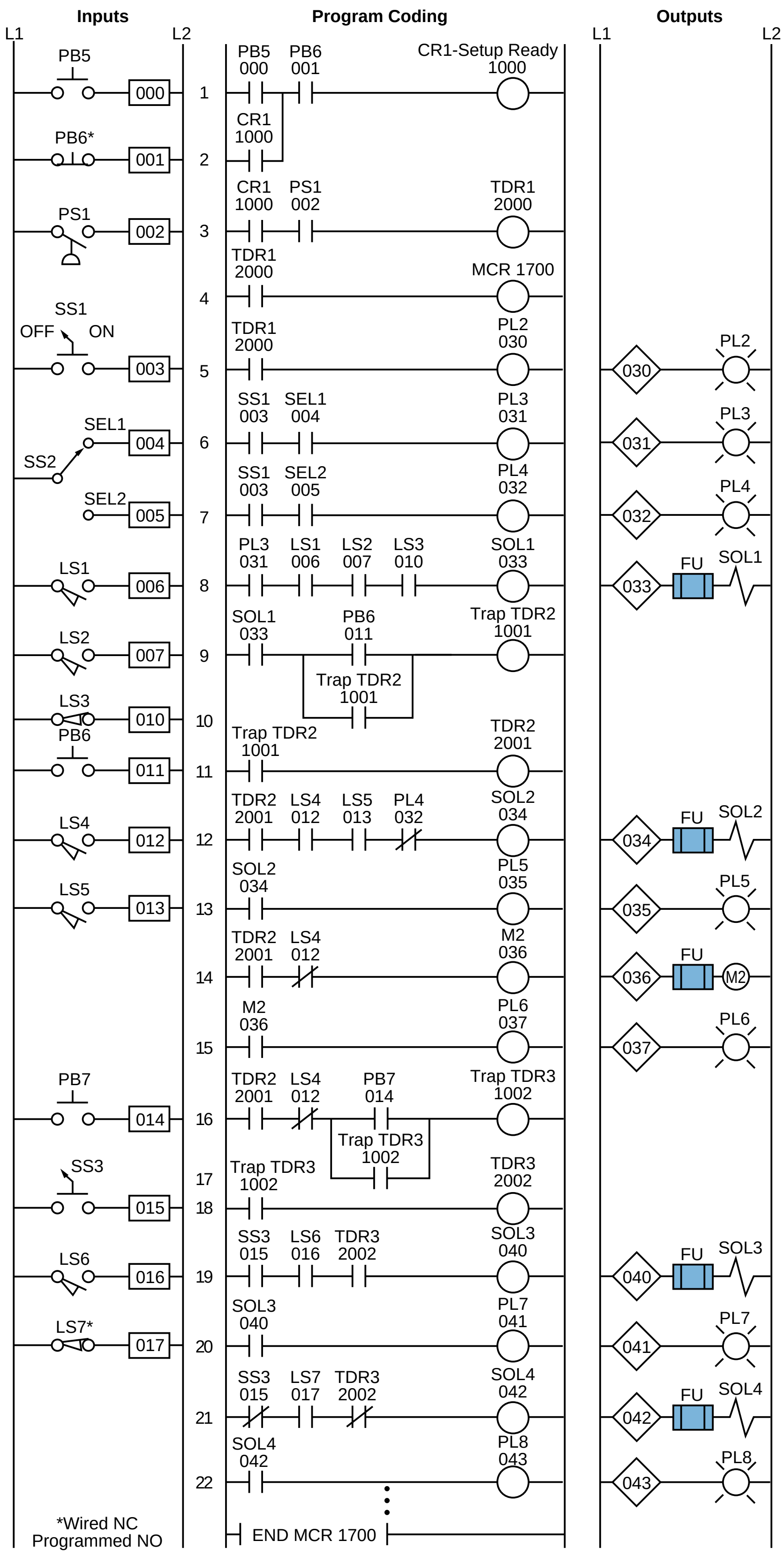


Figure 47. PLC implementation of the circuit in Figure 45.

timer will not seal until it has timed out. If PB6 is released, the timer will reset because PB6 is not sealed. To solve this problem, we can use internal coil 1001 to seal PB6 and start timing timer 2001 (TDR2). Lines 9, 10, and 11 of the PLC program coding show this technique. The time delay contacts (2001) are used for ON delays.

Introduction to PLC Programming and Implementation— from relay logic to PLC logic

Study Guide and Review Questions

PLC Skills

- **Review**
- **Reinforce**
- **Test**
- **Sharpen**

Industrial Text & Video Company www.industrialtext.com 1-800-752-8398

Study Guide

- The first step in developing a control program is the definition of the *control task*. The control task specifies what needs to be done and is defined by those who are involved in the operation of the machine or process.

- The second step in control program development is to determine a *control strategy*, the sequence of processing steps that must occur within a program to produce the desired output control. This is also known as the development of an algorithm.

- A set of guidelines should be followed during program organization and implementation in order to develop an organized system. Approach guidelines apply to two major types of projects: new applications and modernizations of existing equipment.

- *Flowcharting* can be used to plan a program after a written description has been developed. A flowchart is a pictorial representation of the process that records, analyzes, and communicates information, as well as defines the sequence of the process.

- Logic gates or contact symbology are used to implement the logic sequences in a control program. Inputs and outputs marked with an “X” on a logic gate diagram represent real I/O.

- Three important documents that provide information about the arrangement of the PLC system are the I/O assignment table, the internal address assignment table, and the register address assignment table.
  - The *I/O assignment table* documents the names, locations, and descriptions of the real inputs and outputs.
  - The *internal address assignment table* records the locations and descriptions of internal outputs, registers, timers, counters, and MCRs.
  - The *register address assignment table* lists all of the available PLC registers.

- Certain parts of the system should be left hardwired for safety reasons. Elements such as emergency stops and master start push buttons should be left hardwired so that the system can be disabled without PLC intervention.

- Special cases of input device programming include the program translation of normally closed input devices, fenced MCR circuits, circuits that allow bidirectional power flow, instantaneous timer contacts, and complicated logic rungs.
  - The programming of contacts as normally open or normally closed depends on how they are required to operate in the logic program. In most cases, if a normally closed input device is required to act as a normally closed input, its reference address is programmed as normally open.

- Master control relays turn ON and OFF power to certain logic rungs. In a PLC program, an END MCR instruction must be placed after the last rung an MCR will control.
- PLCs do not allow bidirectional power flow, so all PLC rungs must be programmed to operate only in a forward path.
- PLCs do not provide instantaneous contacts; therefore, an internal output must be used to trap a timer that requires these contacts.
- Complicated logic rungs should be isolated from the other rungs during programming.

• Program coding is the process of translating a logic or relay diagram into PLC ladder program form.

• The benefits of modernizing a relay control system include greater reliability, less energy consumption, less space utilization, and greater flexibility.

REVIEW QUESTIONS

1 What is the first step in designing an effective PLC control system?
a–approach the system in a systematic manner
b–flowchart the process
c–define the control task
d–define the control strategy

2 A(n) ________________ is a procedure that uses a finite number of steps to achieve a desired outcome.

3 List four guidelines that are recommended as an approach to modernizing a control system.

4 In a modernization project, an existing ________________ often defines the sequence of events in the control program.

5 System operation for new applications usually begins with:
a–sample diagrams
b–specifications
c–the control strategy
d–logic diagrams

6 A(n) ________________ is a graphical representation of a solution's algorithm.

7 Logic sequences for a control program can be created using:
a–logic gates
b–relay ladder symbology
c–PLC contact symbology
d–all of the above

8 Draw the equivalent logic gate diagram for the circuit shown in Figure 1.

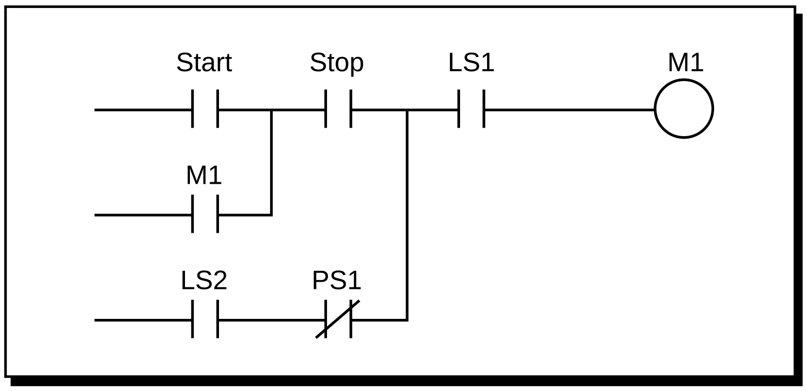


Figure 1. Circuit for problem 8.

9 Draw the equivalent contact symbology diagram for the logic gates shown in Figure 2.

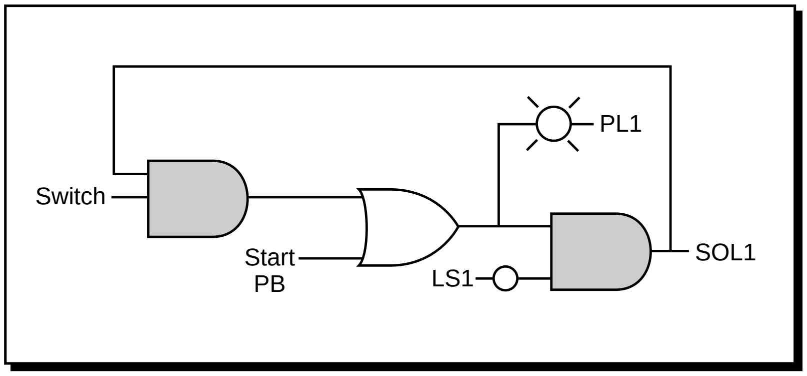


Figure 2. Logic gates for problem 9.

10 *True/False*. Only real inputs and outputs are documented during address assignment.

11 I/O address assignments are typically represented in one of three number systems: ________________, ________________, or ________________.

12 The I/O address assignment table should closely follow the ________________.

13 Using the circuit shown in Figure 3 and assuming that the PLC has a modularity of 8 points per module, there are eight modules per rack, the master rack is numbered 0, and the number system is octal:

(a) circle all real inputs and outputs

(b) assign the I/O addresses

(c) draw the I/O connection diagram

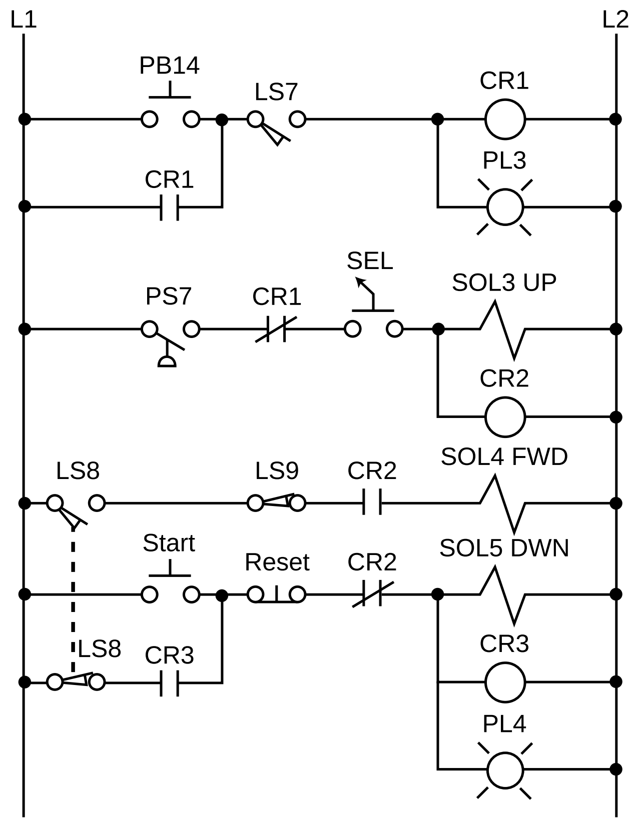


Figure 3. Circuit for problem 13.

14 The principle reason for leaving certain portions of the control circuit hardwired is to:

a–minimize wiring
b–avoid failure of main magnetic elements
c–ensure safety
d–keep some devices running at all times

15 The PLC fault contacts are wired to other hardwired emergency circuit elements:

a–in parallel
b–in series
c–normally open
d–normally closed

16 The main reason the PLC fault contacts are included in the hardwired circuit is:

a–to prevent system shut down
b–to detect I/O failures
c–to include the PLC as an emergency stop condition
d–to shut down the system if there is a PLC failure

17 Describe the purpose and operation of a safety control relay (SCR).

18 *True/False*. Normally closed input devices are always programmed normally open.

19 What is the purpose of the normally closed PLC fault contacts in the circuit in Figure 4 and describe what will happen if the PLC fails?

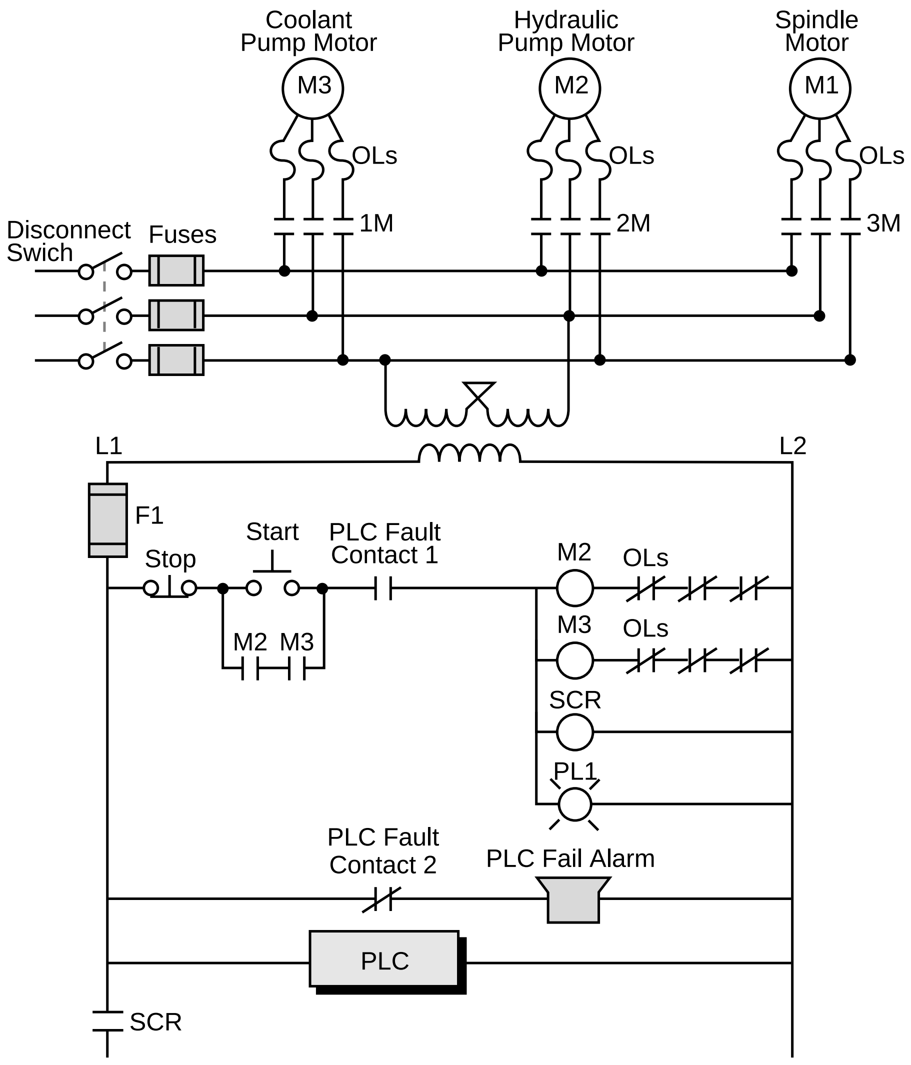


Figure 4. Circuit for problem 19.

20 Using the circuit shown in Figure 5 and starting inputs at address 10_8, outputs at address 50_8, and internals at address 100_8:

(a) assign the I/O addresses

(b) draw the equivalent PLC ladder diagram

21 *True/False.* In a PLC ladder program, an END MCR instruction must be used to fence the area controlled by a master control relay.

22 What element can be used to trap timers in a PLC control program?

a–a reset instruction
b–a start push button
c–a pilot light
d–an internal output

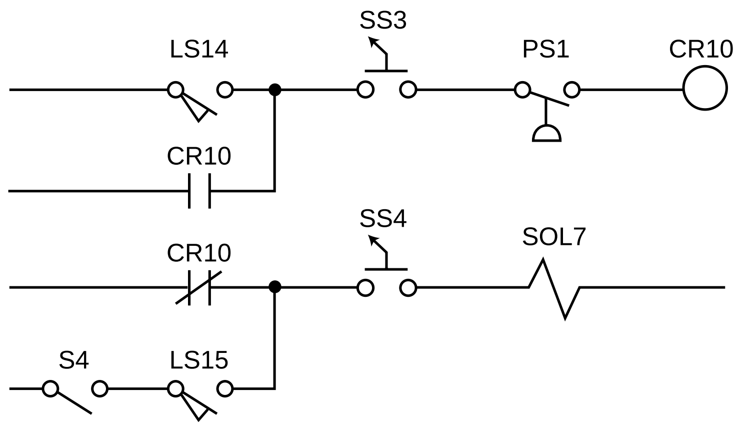


Figure 5. Circuit for problem 20.

23 Explain why the hardwired circuit in Figure 6 must be reconfigured when it is translated into a PLC ladder diagram.

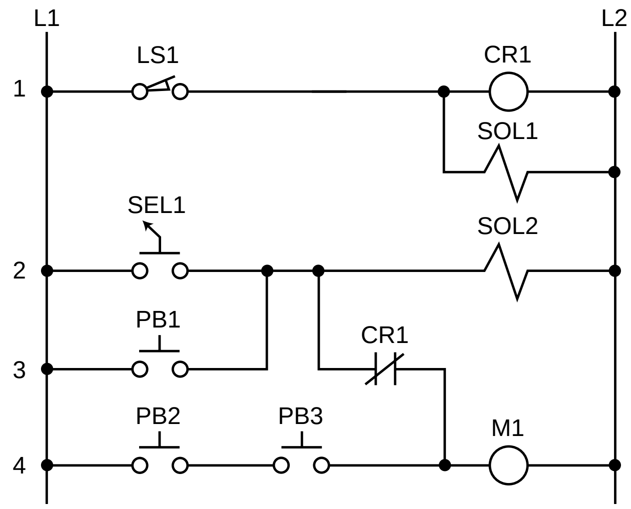


Figure 6. Circuit for problem 23.

24 Program ___________________ is the process of translating logic or relay contact diagrams into PLC ladder form.

25 Assuming that inputs use addresses 000–027, outputs use addresses 030–047, internals start at address $100_{8,}$ timers start at address $200_{8,}$ and an internal output is used to trap the instantaneous timer contacts, use the circuit shown in Figure 7 to:

(a) assign the internal addresses

(b) assign the I/O addresses

(c) draw the I/O connection diagram

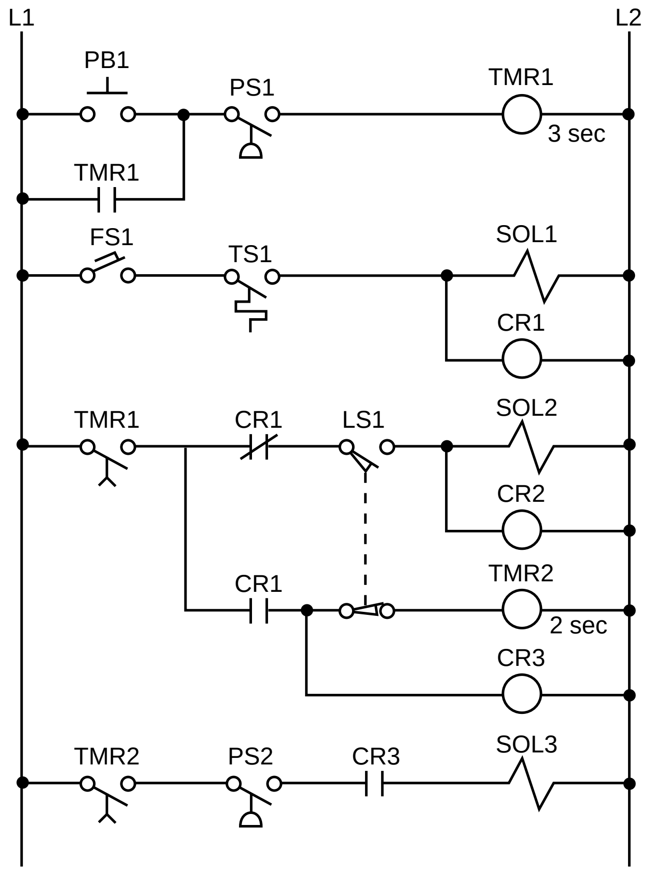


Figure 7. Circuit for problem 25.

26 Given that the stop push button will be wired as normally open, use the circuit in Figure 8 to:

(a) assign the I/O addresses

(b) draw the I/O connection diagram

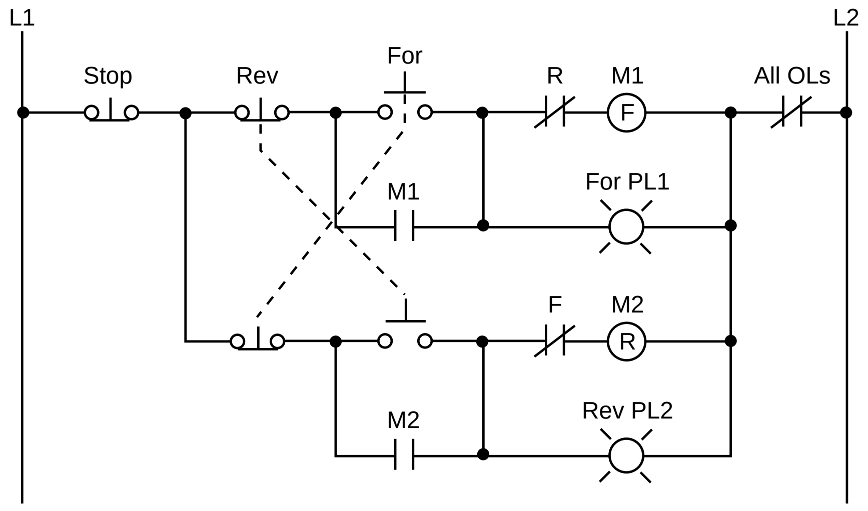


Figure 8. Circuit for problem 26.

27 Circle the locations where timer traps will be used in the PLC implementation of the circuit in Figure 9.

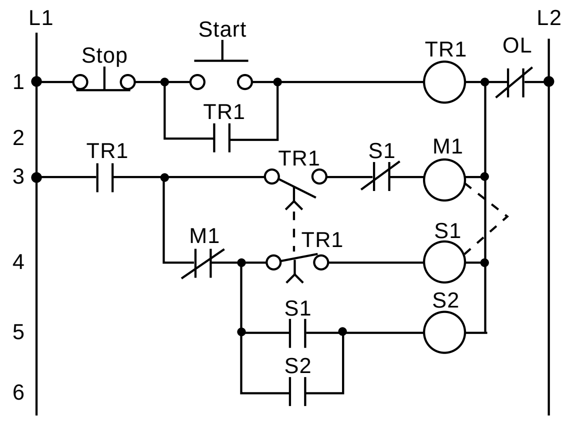


Figure 9. Circuit for problem 27.

28 Figure 10 shows a variable speed drive that is manually controlled by an operator station. What input field devices are required for the PLC implementation of this station?

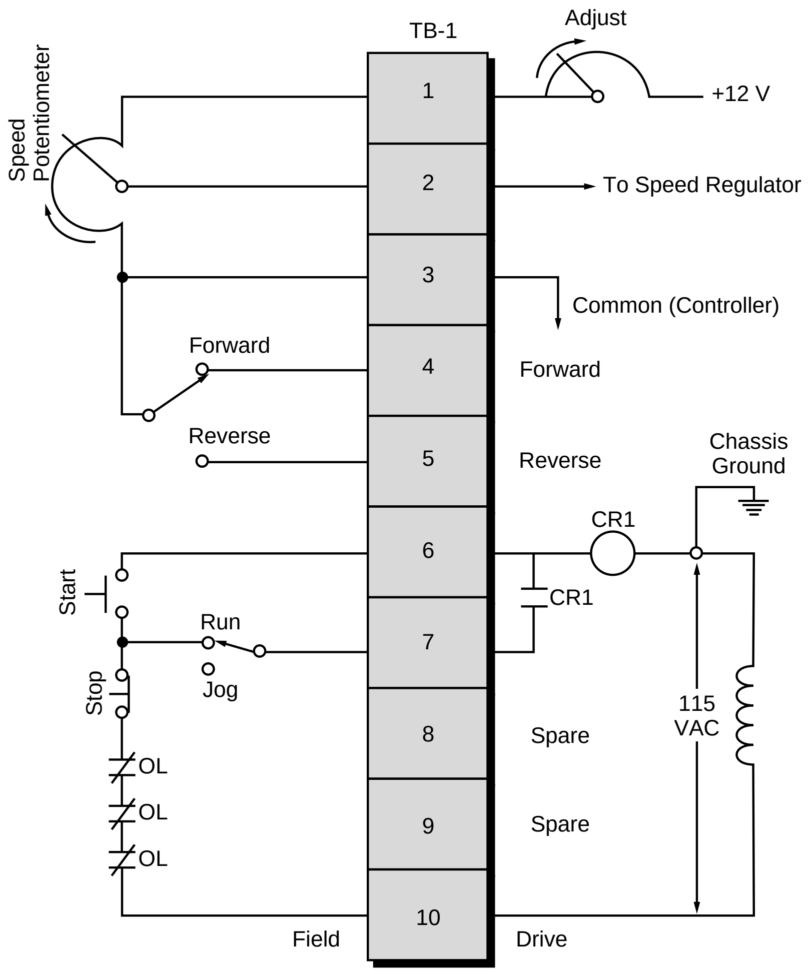


Figure 10. Variable speed drive.

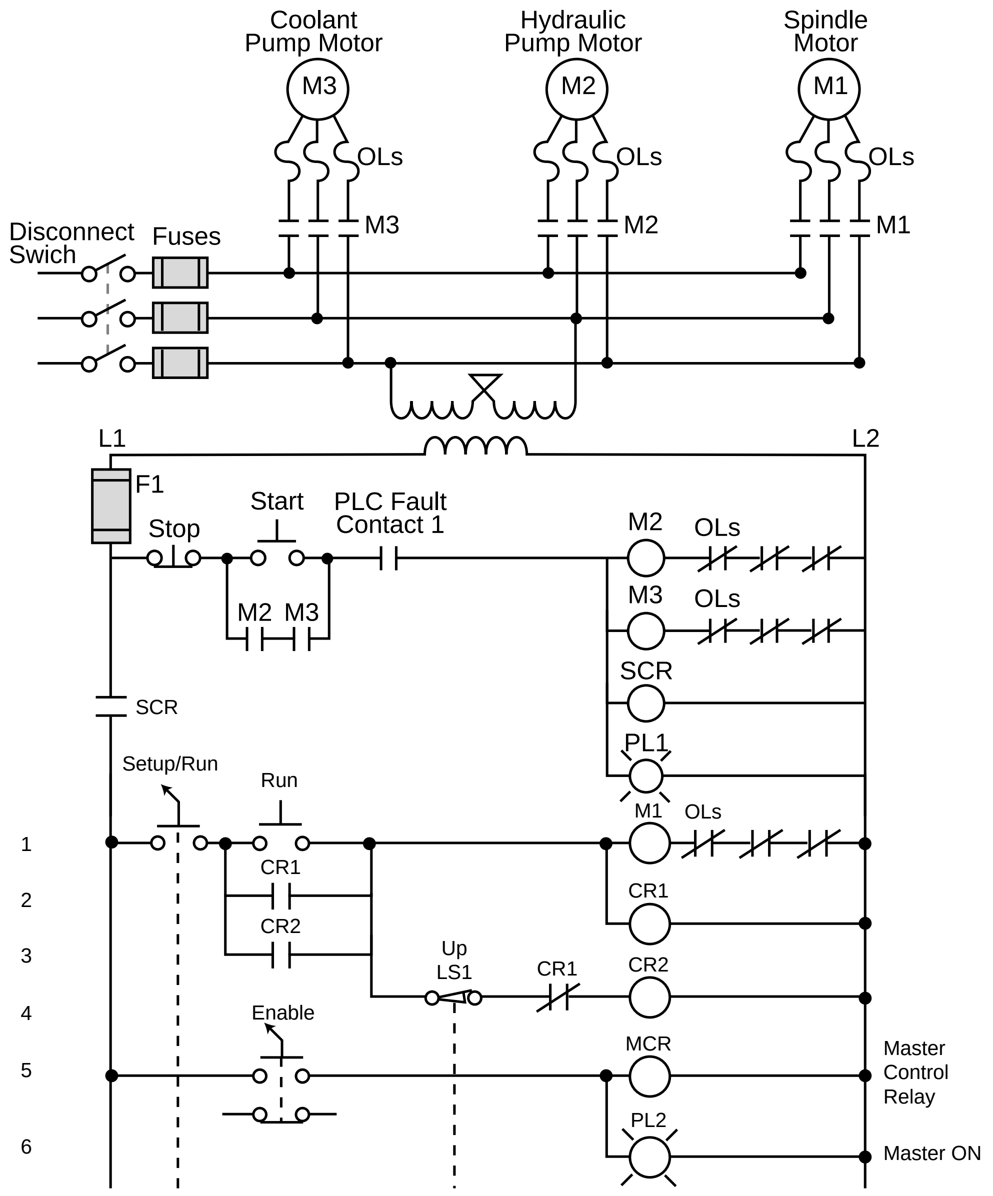
29 The PLC implementation of the large relay circuit in Figure 11 should include a normally open PLC fault contact and use internals to trap the timers. The PLC system has capacity for 512 I/O (000 to 777 octal). Inputs should start at address 000_8 and outputs should start at address 030_8. Internals should have addresses 1000–1777, MCRs should have addresses 2000–2037, and timers should have addresses 2040–2137. Using this large relay circuit:

(a) indicate the portions to be left hardwired

(b) assign the I/O addresses

(c) assign the internal addresses

(d) draw the I/O connection diagram



(continued on next page)

Figure 11. Large relay circuit.

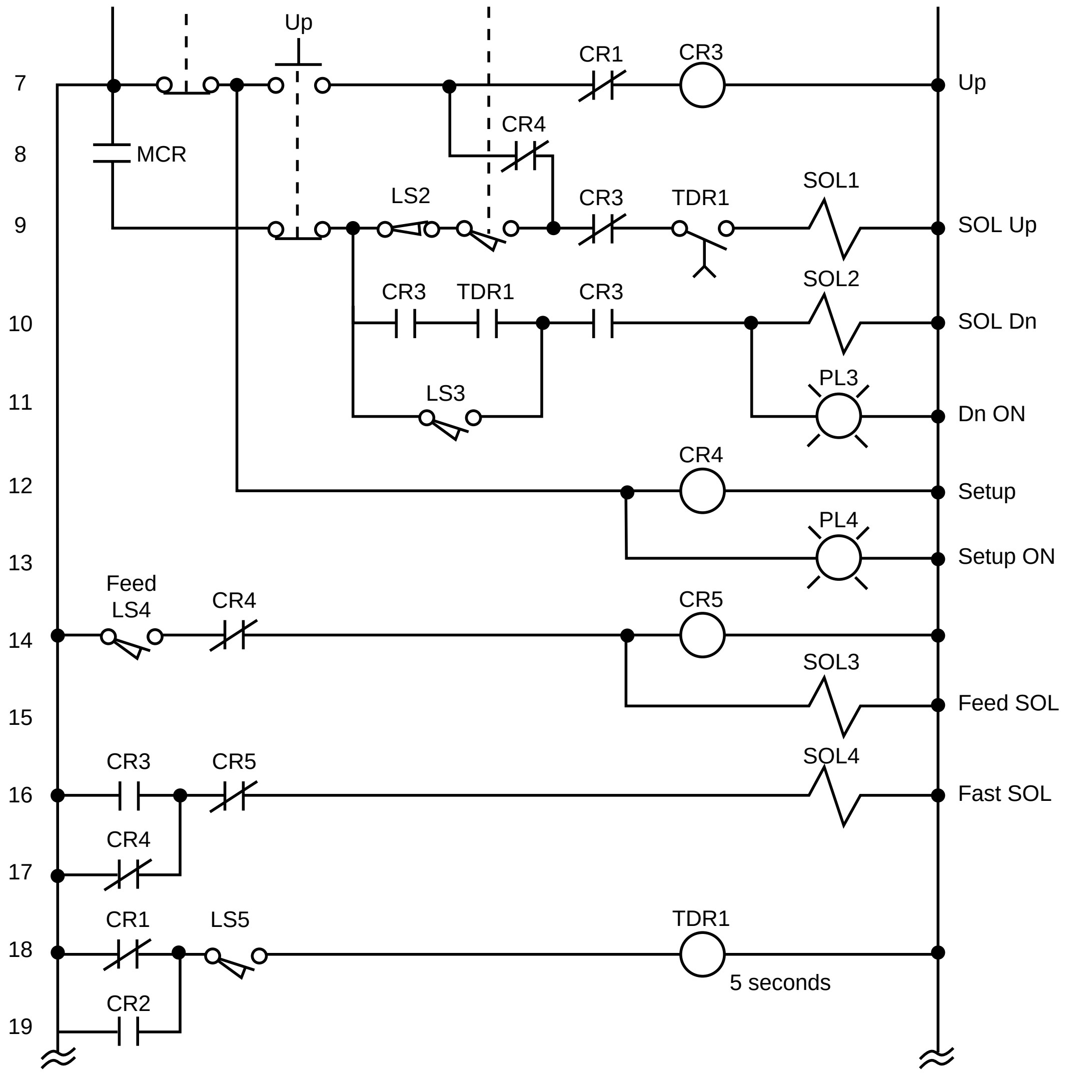


Figure 11 continued.

ANSWERS

1 c–define the control task

2 algorithm

3 Guidelines for modernizing a control system include:
- understanding the actual process or machine function
- reviewing the machine logic and optimizing it when possible
- assigning real I/O addresses and internal addresses to inputs and outputs
- translating relay ladder diagrams into PLC coding

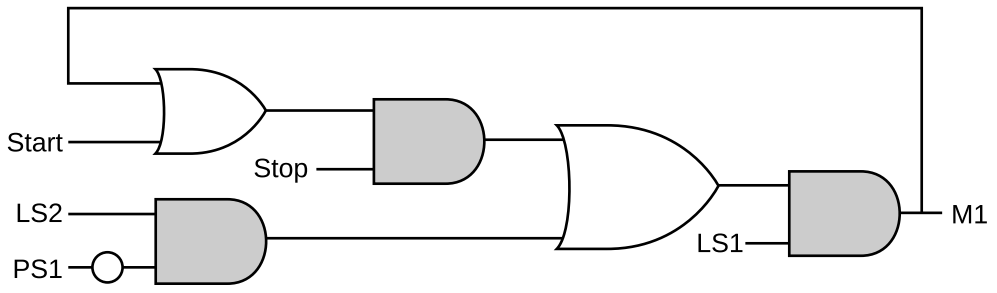
4 relay ladder diagram

5 b–specifications

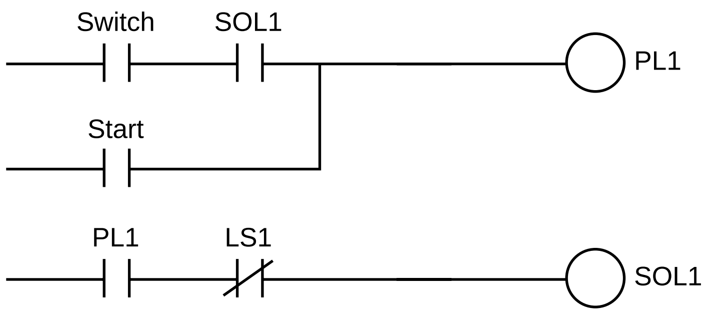
6 flowchart

7 d–all of the above

8



9

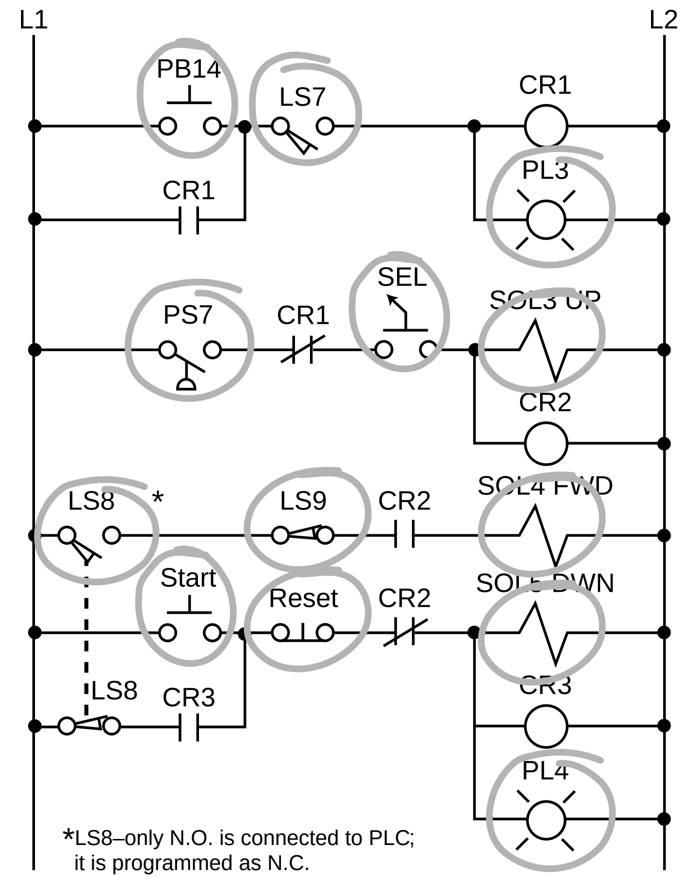


10 false; internal outputs are also documented during address assignment

11 octal, decimal, hexadecimal

12 I/O connection diagram

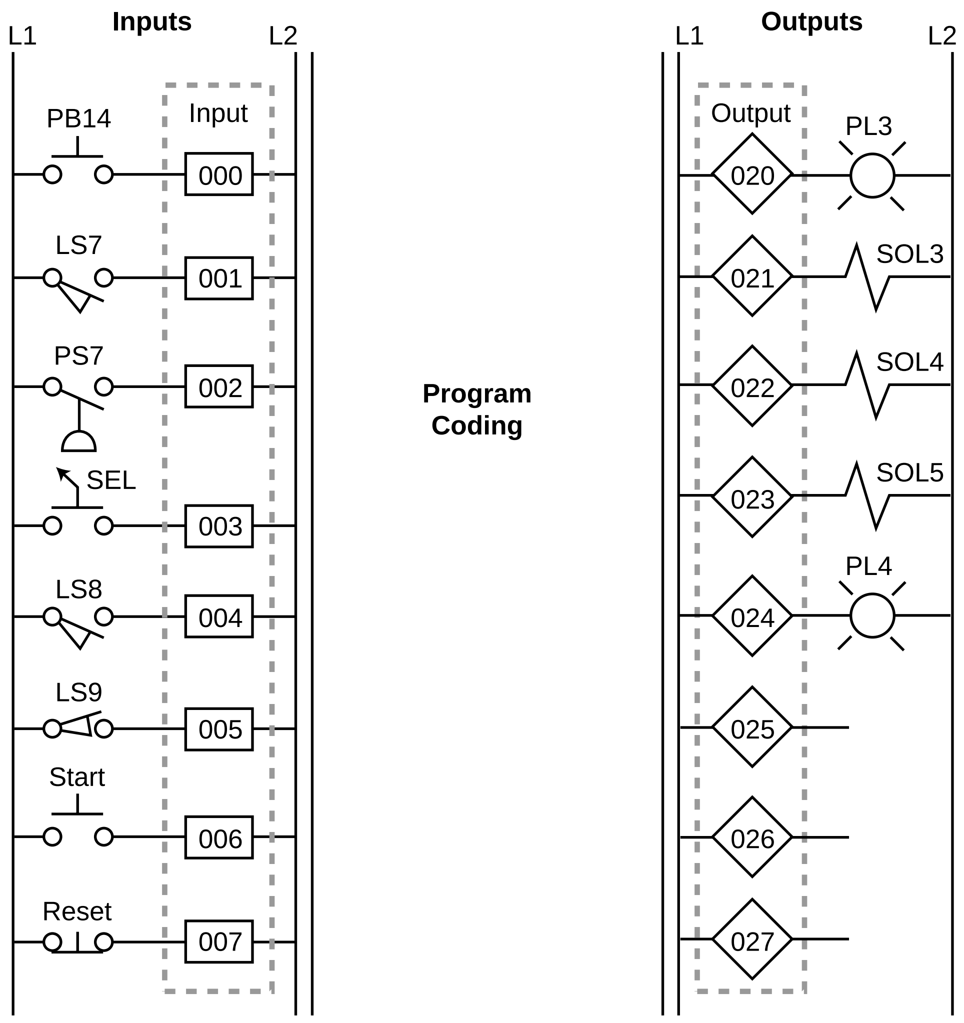
13 *(a)*



(b)

Module Type	I/O Address Rack	I/O Address Group	I/O Address Terminal	Description
Input	0	0	0	PB14
	0	0	1	LS7
	0	0	2	PS7
	0	0	3	SEL
	0	0	4	LS8
	0	0	5	LS9
	0	0	6	Start PB
	0	0	7	Reset PB
Spare	0	1	0	Not used
	·	·	·	
	·	·	·	
	·	·	·	
	0	1	7	
Output	0	2	0	PL3
	0	2	1	SOL3 Up
	0	2	2	SOL4 Forward
	0	2	3	SOL5 Down
	0	2	4	PL4
	0	2	5	—
	0	2	6	—
	0	2	7	—

(c)



14 c–ensure safety

15 b–in series

16 d–to shut down the system if there is a PLC failure

17 A safety control relay is used to remove power from the I/O modules during a system error. When a malfunction occurs, the safety control relay will turn off, opening its SCR contact to stop the flow of power to the connected devices.

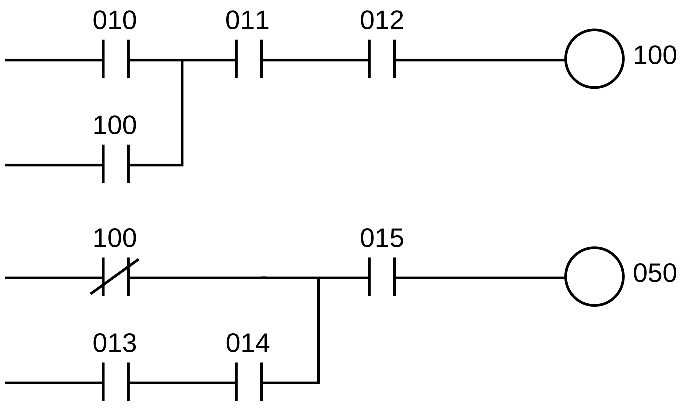
18 false; most of the time a normally closed input device is programmed as normally open; however, the programming of the input will depend on its function in the program

19 The normally closed PLC fault contacts are used to energize the PLC failure alarm. If the PLC fails, the PLC fault coil will not energize. Therefore, the normally open PLC fault contacts will not close to provide power to the connected devices. Instead, the normally closed PLC fault contacts will remain closed, sounding the PLC failure alarm.

20 *(a)*

Module Type	I/O Address Rack	Group	Terminal	Description
Input	0	1	0	LS14
	0	1	1	SS3
	0	1	2	PS1
	0	1	3	S4
	0	1	4	LS15
	0	1	5	SS4
	0	1	6	—
	0	1	7	—
Output	0	5	0	SOL7
	⋮	⋮	⋮	
Internal	1	0	0	CR10

(b)



21 true

22 d–an internal output

23 There is a possibility of bidirectional power flow through the normally closed contact CR1 in line 3. A PLC will only allow power to flow in a forward path. Therefore, if the reverse path from line 4 to line 2 is meant to be followed, the circuit would have to be reconfigured so that the CR1 contacts are included in both lines 2 and 4.

24 coding

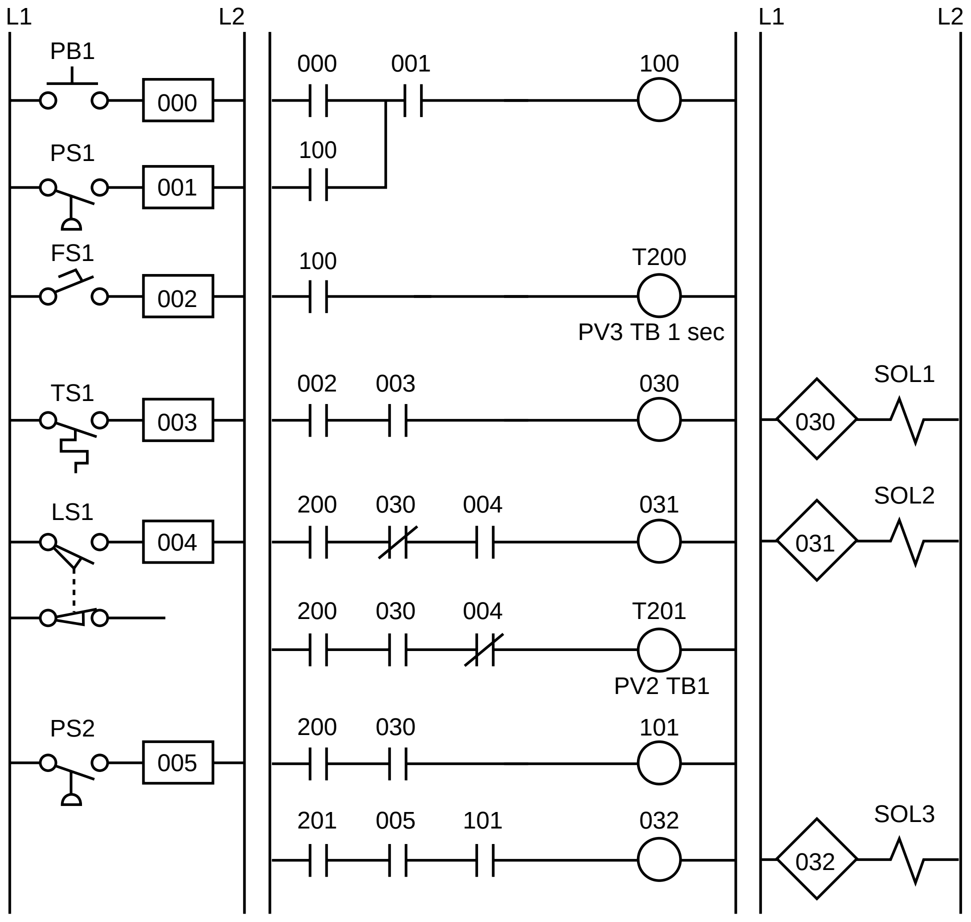
25 *(a)*

Device	Internal	Description
TMR1	100	Used to trap TMR1
CR1	—	Same as SOL1
CR2	—	Same as SOL2
CR3	101	Replace CR3
TMR1	200	Timer 1
TMR2	201	Timer 2

(b)

	I/O Address			
Module Type	**Rack**	**Group**	**Terminal**	**Description**
Input	0	0	0	PB1
	0	0	1	PS1
	0	0	2	FS1
	0	0	3	TS1
	0	0	4	LS1
	0	0	5	PS2
	0	0	6	—
	0	0	7	—
Output	0	3	0	SOL1
	0	3	1	SOL2
	0	3	2	SOL3
	⋮	⋮	⋮	

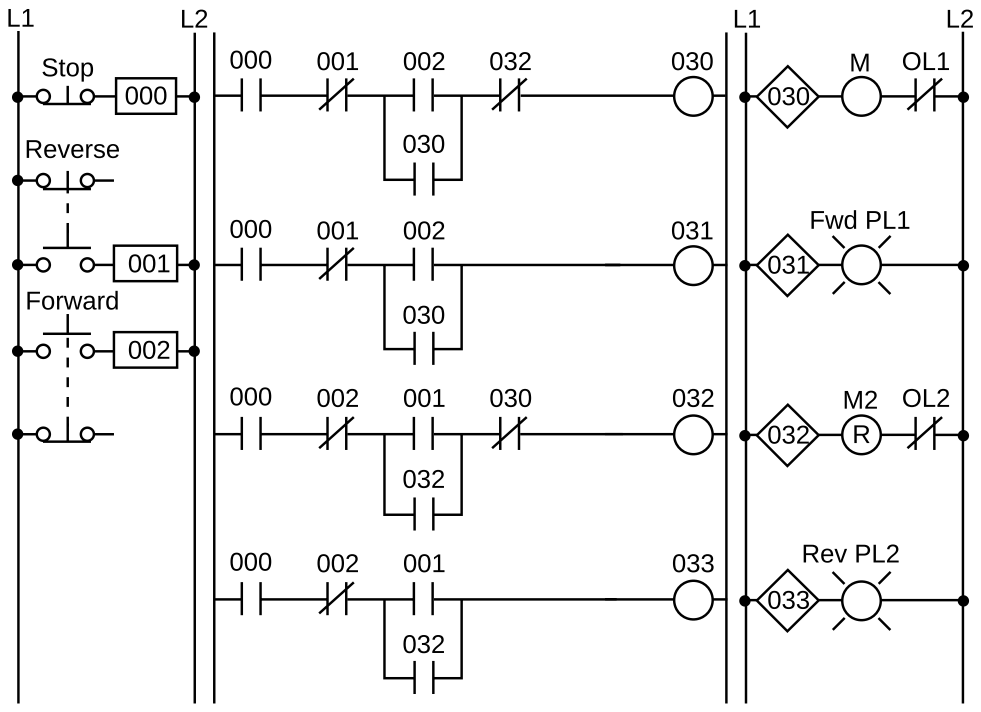
(c)



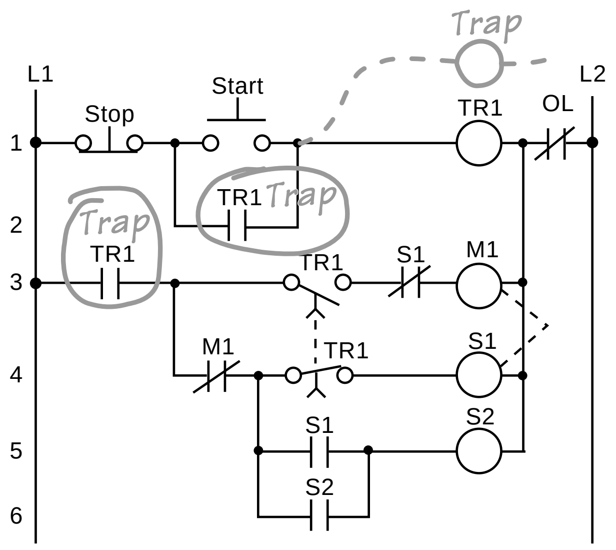
26 *(a)*

	I/O Address			
Module Type	**Rack**	**Group**	**Terminal**	**Description**
Input	0	0	0	Stop PB (wired NC)
	0	0	1	Forward PB (wired NO)
	0	0	2	Reverse PB (wired NO)
	0	0	3	Overload contacts
Input	0	0	4	Acknowledge OL/Reset PB
	•	•	•	
	•	•	•	
	•	•	•	
Output	0	3	0	Motor starter M1 (FWD)
	0	3	1	Forward PL1
	0	3	2	Motor starter M2 (REV)
	0	3	3	Reverse PL2

(b)

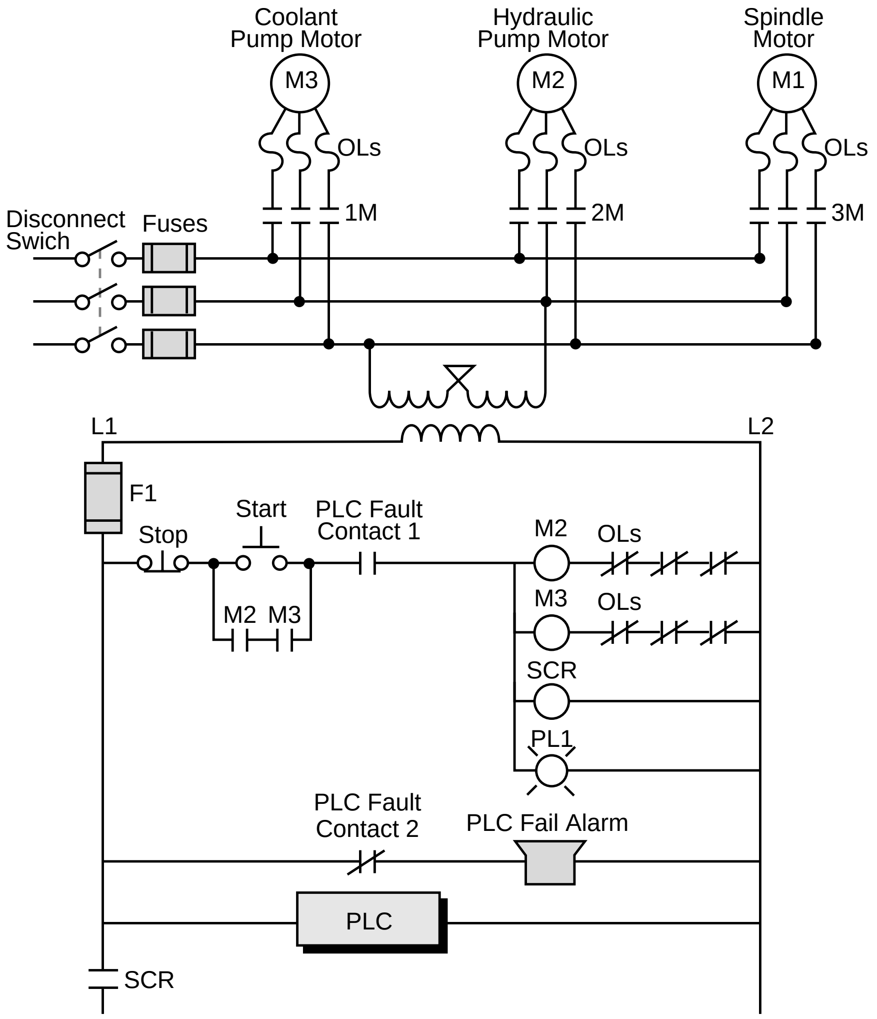


27



28 The field input devices are the start push button, stop push button, jog/run selector switch, and forward/reverse selector switch. The speed potentiometer will be replaced by an analog output in the PLC implementation.

29 *(a)*



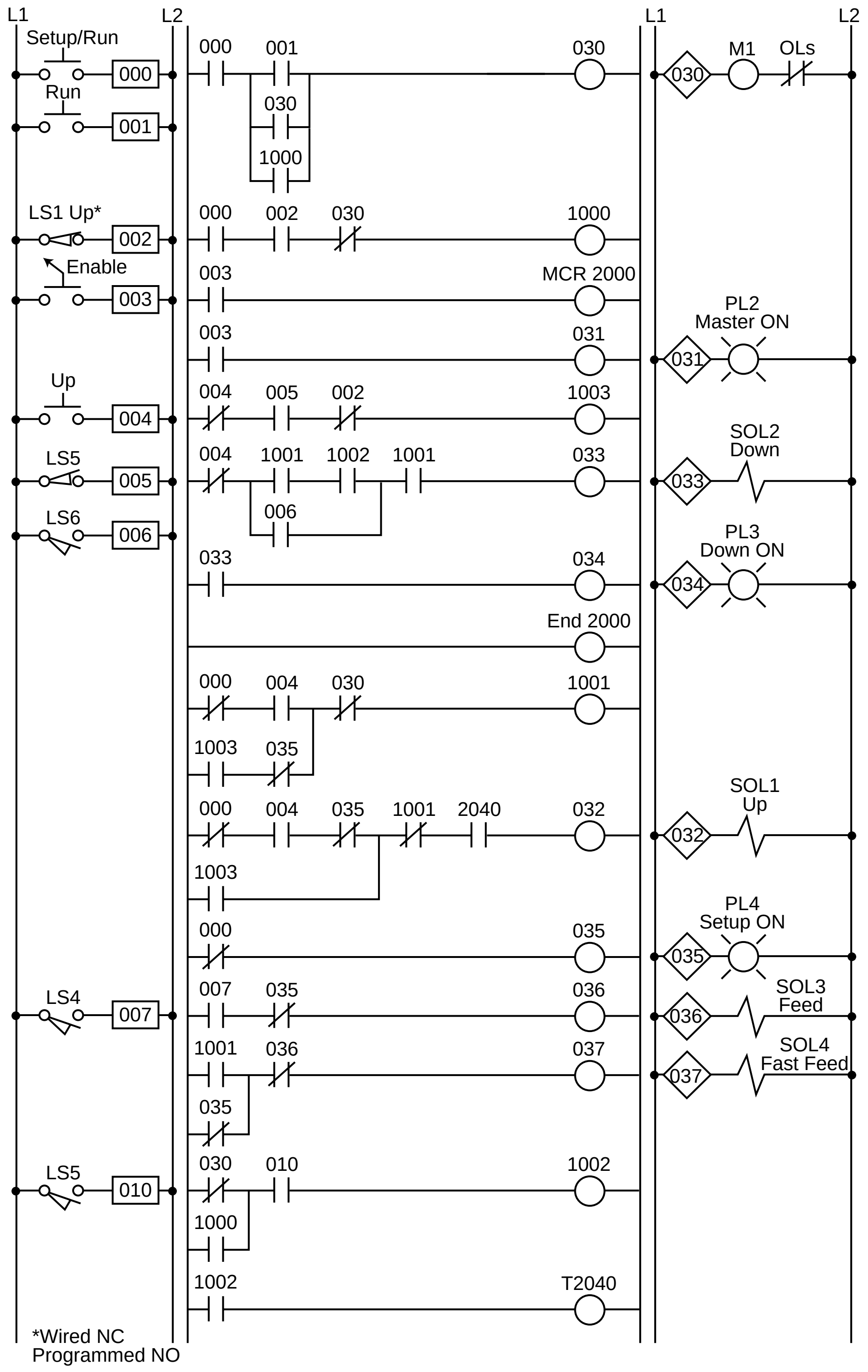
(b)

	I/O Address			
Module Type	**Rack**	**Group**	**Terminal**	**Description**
Input	0	0	0	Setup/Run
	0	0	1	Run PB
	0	0	2	Up LS1
	0	0	3	Enable SS
Input	0	0	4	Up PB
	0	0	5	LS2
	0	0	6	LS3
	0	0	7	Feed LS4
Input	0	1	0	LS5
	0	1	1	Not used
	⋮	⋮	⋮	⋮
	0	1	7	
Output	0	3	0	M1 Starter
	0	3	1	PL2 Master On
	0	3	2	SOL1 Up
	0	3	3	SOL2 Down
Output	0	3	4	PL3 Down On
	0	3	5	PL4 Setup On
	0	3	6	SOL3 Feed
	0	3	7	SOL Fast Feed

(c)

Device	Internal	Description
CR1	—	Same as M1
CR2	1000	Replace CR2
MCR	MCR 2000	First MCR address—replace MCR
CR3	1001	Replace CR3
CR4	—	Same as PL4
CR5	—	Same as SOL3
—	1002	Trap timer
TDR1	T 2040	First timer address—replace TDR1
MCR	END 2000	END MCR logic section
—	1003	Same as SOL1 in MCR fence

(d)


L1
L2
Setup/Run
000
Run
001
LS1 Up*
002
Enable
003
Up
004
LS5
005
LS6
006
LS4
007
LS5
010
*Wired NC
Programmed NO
000
001
030
1000
002
MCR 2000
003
031
004
005
1003
1001
1002
006
033
033
034
End 2000
035
032
2040
036
037
T2040
M1
OLs
PL2
Master ON
SOL2
Down
PL3
Down ON
SOL1
Up
PL4
Setup ON
SOL3
Feed
SOL4
Fast Feed