

# CHAPTER 10: PROGRAMMABLE LOGIC CONTROLLERS

## 10.1 Introduction

The National Electrical Manufacturers Association (NEMA) as defines a programmable logic controller:

*A digitally operating electronic apparatus which uses a programmable memory for the internal storage of instructions for implementing specific functions such as logic, sequencing, timing, counting, and arithmetic to control, through digital or analog input/output modules, various types of machines or processes.*

In essence, the programmable logic controller consists of computer hardware, which is programmed to simulate the operation of the individual logic and sequence elements that might be contained in a bank of relays, timers, counters, and other hard-wired components.

We will adopt the initials PLC as an abbreviation for the programmable logic controller. PC is widely used in industry for the programmable controller, but the increasingly popular personal computer is also abbreviated PC. To avoid confusion in our book, we will use PLC exclusively for the programmable controller and PC for the personal computer.

The PLC was introduced around 1969 largely as a result of specifications written by the General Motors Corporation. The automotive industry had traditionally been a large buyer and user of electromechanical relays to control transfer lines, mechanized production lines, and other automated systems. In an effort to reduce the cost of new relays purchased each year, GM prepared the specifications for a “programmable logic controller” in 1968. The requirements included:

- The device must be programmable and re-programmable.
- It must be designed to operate in an industrial environment.
- It must accept 120-V ac signals from standard pushbuttons and limit switches.
- Its outputs must be designed to switch and continuously operate loads such as motors and relays of 2-A rating.
- Its price and installation cost must be competitive with relay and solid —state logic devices then in use.

Several companies saw a commercial opportunity in the GM initiative and developed various versions of a special-purpose computer we now refer to as the PLC.

There are significant advantages in using a programmable logic controller rather than conventional relays, timers, counters, and other hardware elements. These advantages include:

- Programming the PLC is easier than wiring the relay control panel. We discuss PLC programming in one of the following subsections.
- The PLC can be reprogrammed. Conventional controls must be rewired and are often scrapped instead.
- PLCs take less floor space than relay control panels.
- Maintenance of the OPLC is easier, and reliability is greater.
- The PLC can be connected to the plant computer systems more easily than relays can.

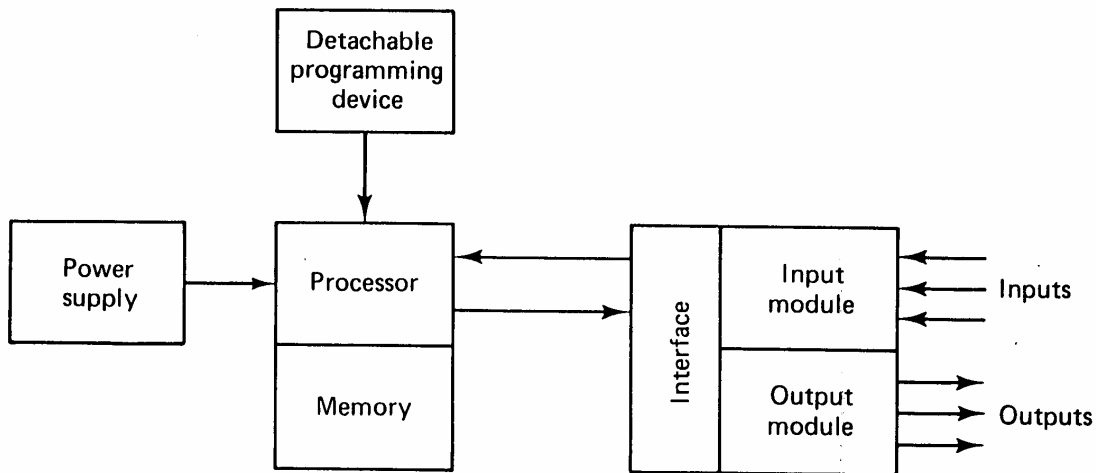
The following subsections describe the components, programming, and operation of the PLC.

We also survey some of its additional capabilities beyond logic control and sequencing.

### 10.1.1 Components of the PLC

A schematic diagram of a programmable logic controller is presented. The basic components of the OPLC are the following

- Input module
- Output module
- Processor
- Memory
- Power supply
- Programming device



**Figure 10.1: Diagram of the programmable logic controller**

The components are housed in a suitable cabinet designed for the industrial environment. A commercially available PLC is shown in figure.

The input module and output module are the connections to the industrial process that in to be controlled. The inputs to the controller are signals from limit switches, pushbuttons, sensors, and other on/off devices. In addition, as we will describe later, larger PLCs are capable of accepting signals from analog devices of the type modeled. The outputs from the controller are on/off signals to operate motors, valves, and other devices required to actuate the process.

The processor is the central processing unit (CPU) of the programmable controller. It executes the various logic and sequencing functions described in previous Sections by operating on the PLC INPUTS TO DETERMINE THE APPROPRIATE OUTPUT SIGNALS. The processor is microprocessor very similar in its construction to those used in personal computers and other data-processing equipment. Tied to the CPU is the PLC memory, which contains the program of logic, sequencing, and other input/output operations. The memory for a

programmable logic controller is specified in the same way as for a computer, and may range from 1k to over 48 k of storage capacity. A power supply of 115 V ac is specially used to drive the PLC even though the components of the industrial process that are regulated may have a higher voltage and power rating than the controller itself.

The PLC is programmed by means of a programming device. The programming device (sometimes referred to as a programmer) is usually detachable from the PLC cabinet so that it can be shared between different controllers. Different PLC manufactures provide different devices, ranging from simple teach pendant-type devices, similar to those used in robotics, to special PLC programming keyboards and CRT displays.

### **10.1.2 Programming the PLC**

Most of the programming methods in use today for PLCs are based on the ladder logic diagram. This diagram has been found to be very convenient for shop personnel who are familiar with circuit diagrams because it does not require them to learn an entirely new programming language. What is required is a means of inputting the program into the OPLC memory:

There are various approaches for entering and interconnecting the individual logic elements.

These include:

1. Entry of the ladder logic diagram
2. Low-level computer-type languages
3. High-level computer-type languages
4. Functional blocks
5. Sequential function chart

The first method involves direct entry of the ladder logic diagram into the PLC memory. This method requires the use of a keyboard and CRT with limited graphics capability to display symbols representing the components and their interrelationships in the ladder logic diagram. The PLC keyboard device is often designed with keys for each of the individual symbols. Programming is accomplished by inserting the appropriate components into the rungs of the ladder diagram. The components are of two basic types. Contacts and coils. Contacts are used to represent loads such as motors, Solenoids, relays, timers, counters, etc. in effect; the programmer inputs the ladder logic circuit diagram rung by rung into the PLC memory with the CRT displaying the results for verification.

The second method makes use of a low-level computer-type language that parallels the ladder logic diagram. Using the language instructions, the programmer contracts the ladder diagram by specifying the various components and their relationships for each rung. Let us explain this approach by developing an elementary PLC instruction set. Our PLC “language” will be a composite of various manufacturers’ languages, containing perhaps fewer features than most commercially available PLCs. We will assume that the programming device consists of a suitable keyboard for entering the individual components on each rung of the ladder logic diagram. A CRT capable of displaying each ladder rung, and perhaps several rungs that precede it, is useful to verify the program.

The low-level languages are generally limited to the types of logic and sequencing functions that can be defined in a ladder logic diagram. Although timers and counters have not been

illustrated in the two preceding examples, some of the problems at the end of the chapter require the reader to make use of them.

High-level computer-type languages are likely to become more common in the future to program the PLC. There are several of these languages that are beginning to be offered commercially, including [ SYBIL (GTE Sylvania), MCL model APC-2 (Cincinnati Milacron), and Control Statements (Reliance Electric). Most of the available languages use an instruction set that is similar to the BASIC computer language for personal computers. There are additional statements available beyond the normal BASIC set to accomplish the control functions.

The principal advantage offered by the high-level languages for programming the PLC is their capability to perform data processing and calculations on values other than binary.. Ladder logic diagrams and low- level OPLC languages are usually quite limited in their ability to operate on signals that are other than ON/OFF types. The capability to perform data processing and computation permits the use of more complex control algorithms, communications with other computer-based systems display of data on a CRT console, and input of data by a human operator. Another advantage of the higher-level languages is the relative ease with which a user can interpret a printout of a complicated control program. Explanatory comments can be inserted into the program to facilitate the interpretation.

## 10.2 Binary Control Logic

In some application of control systems, the variables are binary — they can be either of two possible values, 1 or 0. These values can be interpreted to mean ON or OFF, true or false, object present or not present, high voltage value or low voltage value, and so on.

Followings are some binary input and output devices. Table I .1 Binary Input and Output devices

**TABLE 1.1 Binary Input and Output Devices**

<i>Device</i>	<i>One/zero interpretation</i>
<b>Input</b>	
Limit switch	Contact/no contact
Photodetector	Contact/no contact
Pushbutton switch	On/off
Timer	On/off
Control relay	Contact/no contact
Circuit breaker	Contact/no contact
<b>Output</b>	
Motor	On/off
Alarm buzzer	On/off
Control relay	Contact/no contact
Lights	On/off
Valves	Closed/open
Clutch	Engaged/not engaged
Solenoid	Energized/not energized

### 10.3 Logic Control And Sequencing

A logic control system is a switching system whose output at any moment is determined exclusively by the value of inputs.

A logic control system has no memory and does not consider any previous values of the input signals in determining the output signal. Neither does it have any operating characteristics that perform as a function of time.

#### **An example from Robotics.**

A *sequencing system* is one that uses internal timing devices to determine when to initiate changes in output variables.

#### 10.3.1 Logic Control Elements

There are three basic elements of Logic Control, which are also called Logic Gates:

##### **1. AND**

##### **2. OR**

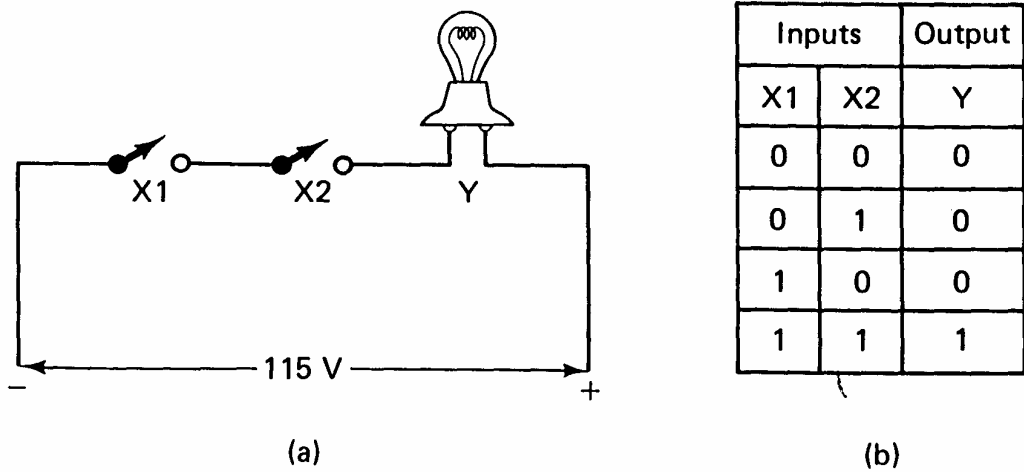
##### **3. NOT**

There are other elements which are derived from these three basic elements above like NOR, and NAND, etc.

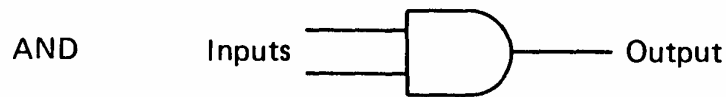
In each case, the logic gate is designed to provide a specified output value based on the values of input(s). For both inputs and outputs, the values can be either of the two levels, the binary values 0 or 1. For purpose of industrial control, we will define 0 (zero) to mean OFF and 1 (one) to mean ON,

##### 10.3.1.1 Logic Gate AND

The logic gate AND outputs a value of 1 if all of the inputs are 1, and 0 otherwise. Figure 10.2 (a) illustrates the operation of a logical AND gate. If both of the switches, X1 and X2 (representing inputs), in the circuit are closed, the lamp Y (representing the output) is on. The truth table for the AND gate is shown in Figure 10.2(b).



**Figure 10.2: Logical AND gate: (a) Circuit illustrating the operation, (b) Its truth table**



### 10.3.1.2 Logic Gate OR

The logic gate OR outputs a value of 1 if either of the inputs have a value of 1, and 0 otherwise. Figure 10.3 (a) illustrates the operation of a logical OR gate. In this case, X1 and X2 (representing inputs) are arranged in a parallel circuit, so that if either of the switches is closed, the lamp Y (representing the output) will be on. The truth table for the OR gate is shown in Figure 10.3 (b).

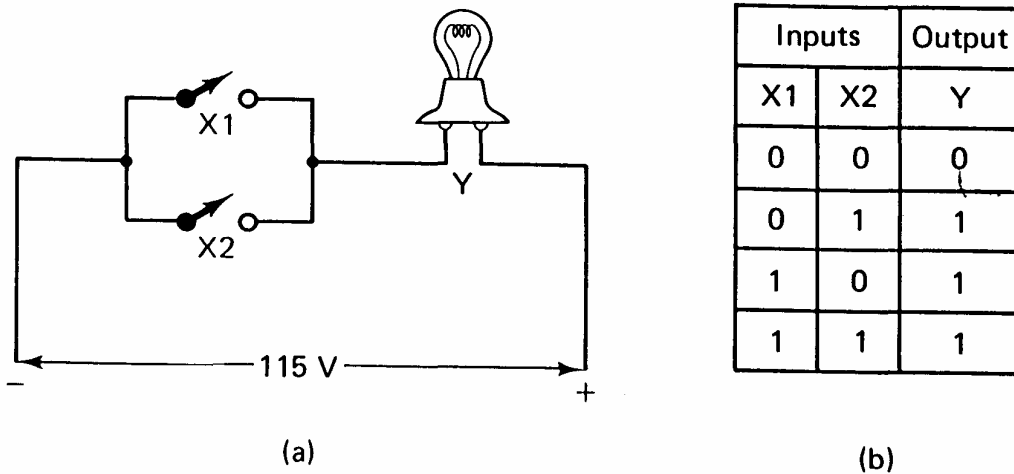
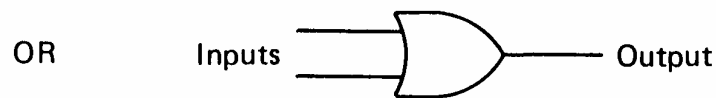


Figure 10.3: Logical OR gate: (a) Circuit illustrating the operation, (b) Its truth table



### 10.3.1.3 Logic Gate NOT

Unlike the AND and OR gate, logic gate NOT has a single input and a single output too. If the input is 1, the output is 0; if the input is 0, the output is 1. Figure 10.4 (a) shows a circuit in which the input switch X1 is arranged in parallel with the outputs so that the voltage flows through the lower path when the switch is closed (thus  $Y = 0$ ), and the upper path when the circuit is open (thus  $Y = 1$ ). The truth table for the NOT gate is shown in Figure 10.4 (b).

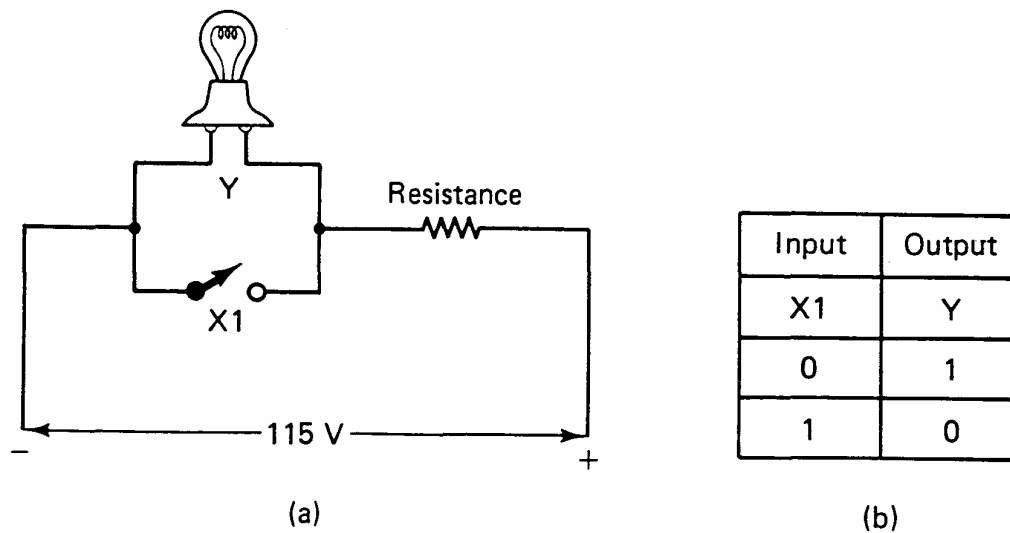
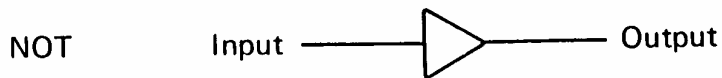


Figure 10.4: Logical NOT gate: (a) Circuit illustrating the operation, (b) Its truth table



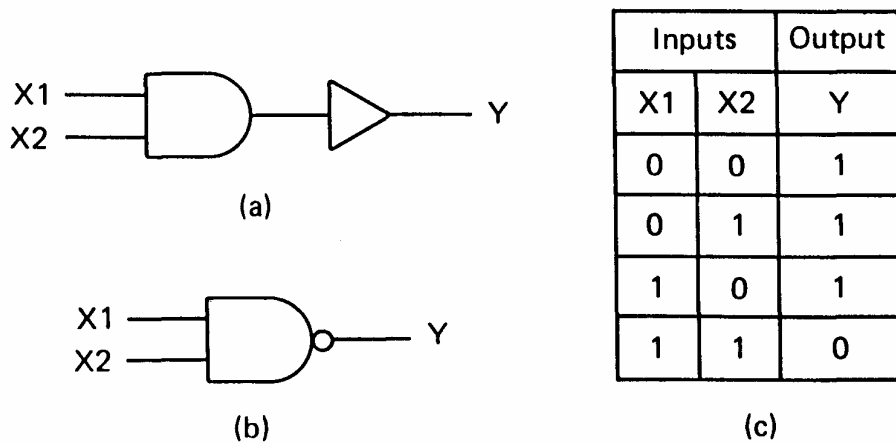
In addition to the three basic elements, there are two more elements that can be identified for use in combinational switching circuits. These are the NAND and NOR gates.

#### 10.3.1.4 Logic Gate NAND

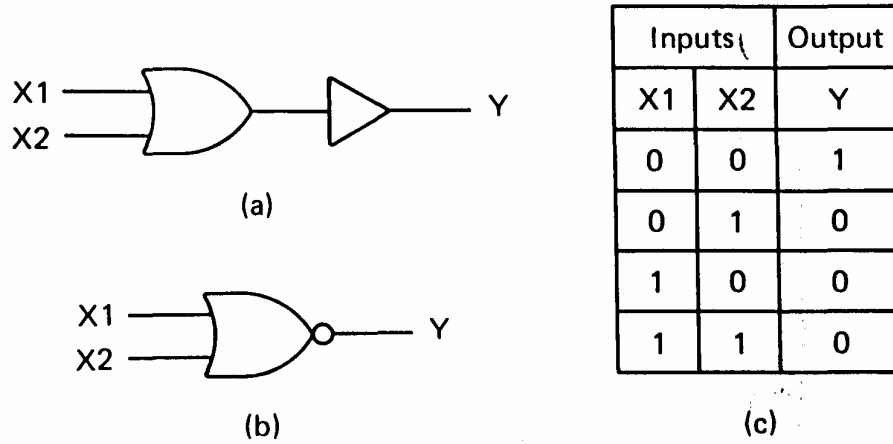
Logic gate NAND is formed by combining an AND and a NOT gate in sequence, as shown in Figure 10.5(a). The logical network symbol for the NAND gate and its truth table are shown in Figure 10.5 (b) and Figure 10.5(c).

#### 10.3.1.5 Logic Gate NOR

Logic gate NOR is formed by combining an OR gate followed by a NOT gate as illustrated in Figure 10.6 (a). The Logic Gate and truth table for the NOR gate are presented in Figure 10.6 (b) and Figure 10.6 (c).



**Figure 10.5: NAND gate: (a) combining AND and NOT gates to form NAND; (b) logic network symbol for NAND; (c) truth table for NAND.**



**Figure 10.6: NOR gate: (a) combining OR and NOT gates to form NOR; (b) logic network symbol for NOR; (c) truth table for NOR.**

### 10.3.2 Example

A motor has one start and one stop button. Being power to motor the output, construct the logic flow diagram and, the truth table.

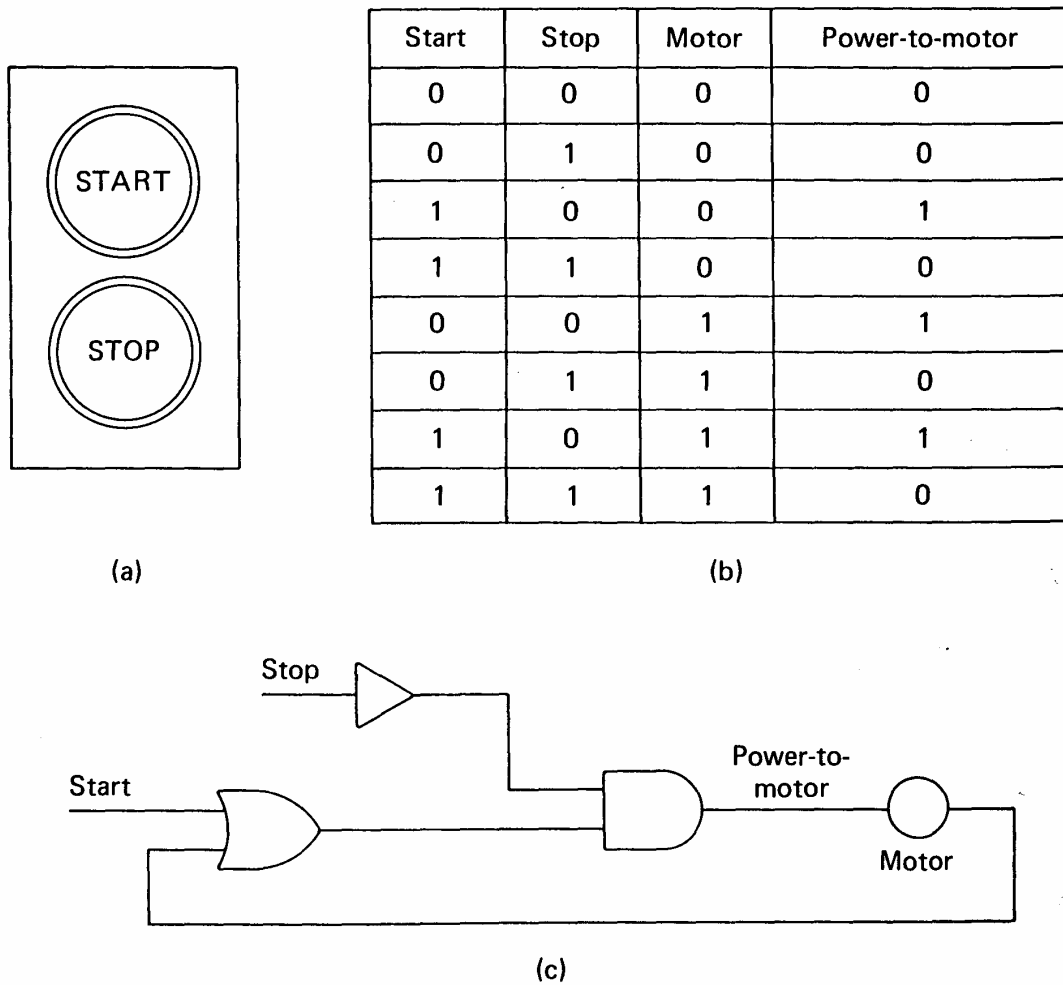


Figure 10.7: (a) Pushbutton switch; (b) its truth table; (c) its logic network diagram

### 10.3.3 Logic Ladder Diagram

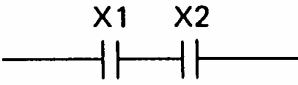
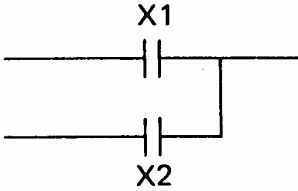
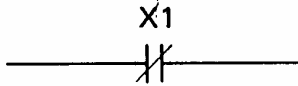
AND		X1 AND X2 ( $X1 \cdot X2$ )
OR		X1 OR X2 ( $X1 + X2$ )
NOT		NOT X1

FIGURE Logic elements AND, OR, and NOT.

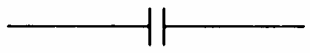
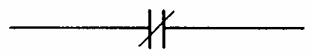
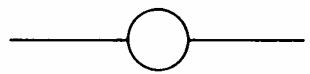
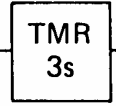
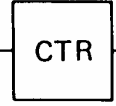
Ladder symbol	Typical hardware components
(a) 	Normally open contacts (switch, relay, other ON/OFF devices)
(b) 	Normally closed contacts (switch, relay, etc.)
(c) 	Output loads (motor, lamp solenoid, alarm, etc.)
(d) 	Timer
(e) 	Counter

Figure 10.8: Symbols for common logic and sequence components in a ladder logic diagram

### 10.3.3.1 An example of Logic Ladder Diagram

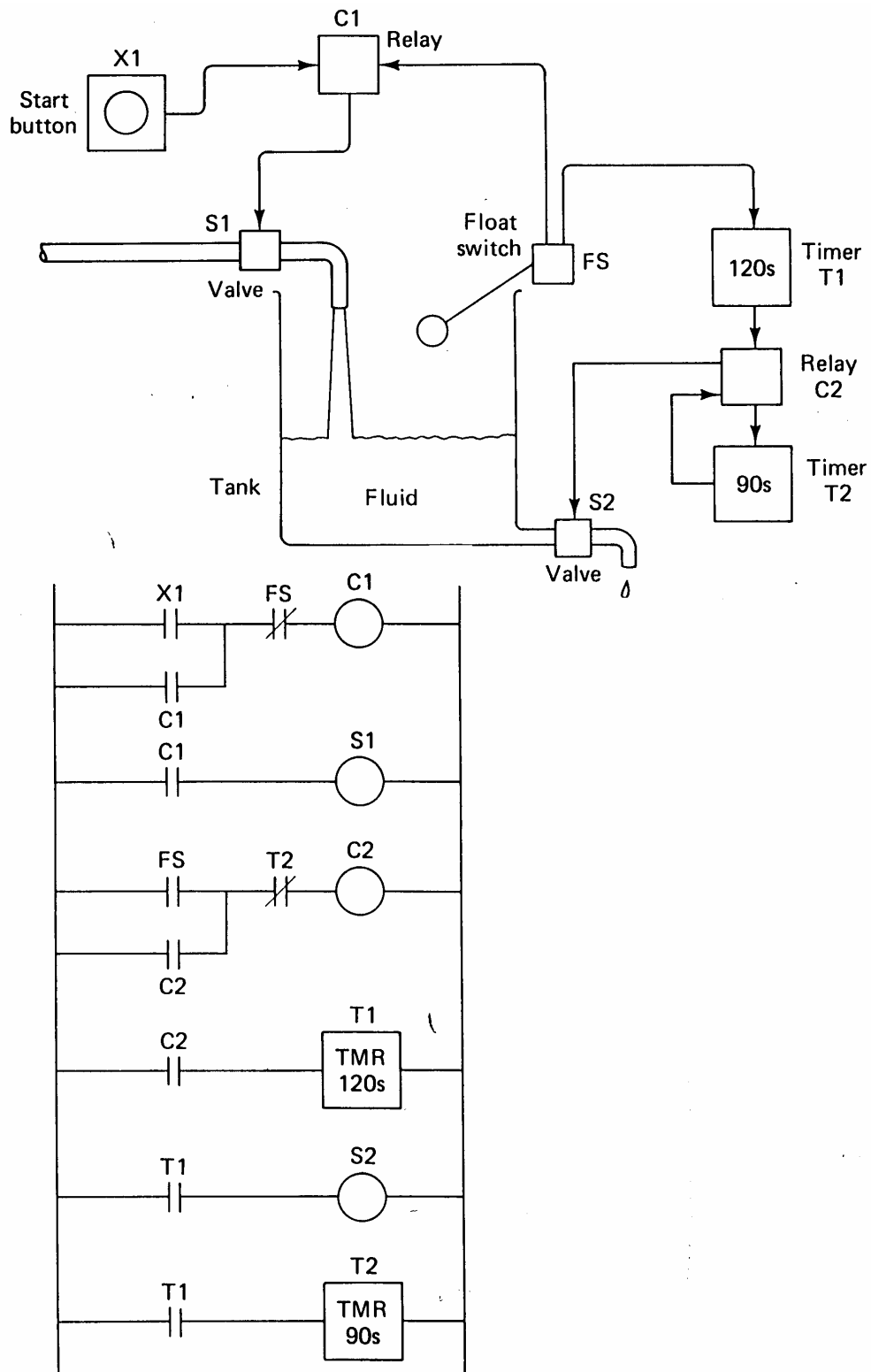


Figure 10.9: Logic ladder diagram