CHAPTER

6

Introduction to Logic

OBJECTIVES

After completing this chapter, you should be able to:

- explain AND logic
- explain OR logic
- explain NOT logic
- develop and fill in a truth table for specified logic
- determine if a PLC ladder rung is true or false under specified conditions
- identify ControlLogix Boolean function blocks

INTRODUCTION

This chapter introduces the concepts of how logic functions are executed when solving a PLC program. We will introduce how a PLC solves a user program using AND, OR, and NOT logic.

Conventional hardwired relay ladder diagrams represent actual hardwired control circuits. In a hardwired circuit, there must be electrical continuity before the load will energize. Even though PLC ladder logic was modeled after the conventional relay ladder, there is no electrical continuity in PLC ladder logic. PLC ladder rungs must have logical continuity before the output will be directed to energize.

Function block diagram programming is a relatively new programming language for American PLCs. Rockwell Automation's ControlLogix family of PLCs is the first Rockwell Automation/Allen-Bradley PLC that supports function block diagram programming. Function block, as it is called, is an additional programming language option where boxes called function blocks replace traditional relay ladder instructions. This chapter will

introduce Boolean function blocks. Boolean function blocks replace the traditional normally open and normally closed instructions.

CONVENTIONAL LADDERS VERSUS PLC LADDER LOGIC

The familiar electrical ladder diagram is the traditional method of representing an electrical sequence of operation in hardwired relay circuits. Ladder diagrams are used to represent the interconnection of field devices. Each rung of the ladder clearly illustrates the relationship of turning on one field device and shows how it interacts with the next field device. Due to industrywide acceptance, ladder diagrams became the standard method of providing control information to users and designers of electrical equipment. With the advent of the programmable controller, one of the specifications for this new control device was that it had to be easily programmed. As a result, programming fundamentals were developed directly from the old familiar ladder diagram format, with which electrical maintenance personnel were already familiar.

The difference between a PLC ladder program and relay ladder rungs involves continuity. An electrical schematic rung has electrical continuity when current flows uninterrupted from the left power rail to the right power rail. Electrical continuity, as illustrated in Figure 6-1, is required to energize the load. An electrical current flows from L-1 through SW 1 and onto the load, returning by way of L-2. When current flows, there is electrical continuity to the status table, where it is stored.

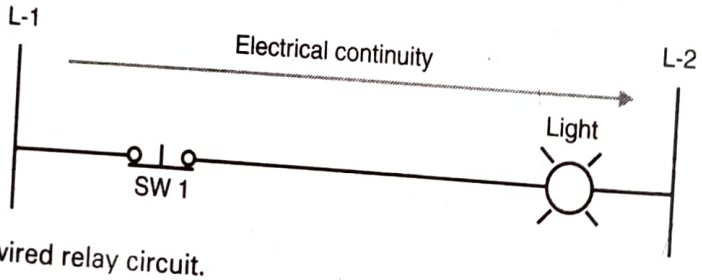

Figure 6-1 Hardwired relay circuit.

Even though a PLC ladder program closely resembles an electrical schematic, PLC ladder symbols represent ladder program instructions. A PLC program is a set of instructions that is stored in memory. These instructions tell the PLC what to do with input signals and then, as a result of following the instructions, where to send the signals.

A PLC relay ladder program uses an electrician's ladder schematic as a model. Even though a PLC ladder program employs familiar terms like "rungs" and "normally open" and "normally closed" contacts, relay ladder logic has no electrical continuity between an input and the controlled output. There is no physical conductor that carries the input signal through to the output. As introduced in Chapter 5, a PLC input signal follows these six steps:

1. The input signal is seen by the input module.
2. The input module isolates and converts the input signal to a low-voltage signal with which the PLC can work.

3. The ON or OFF signal from the input section is sent via the backplane to the input status file, where it is stored.
4. The processor will look at each input's ON or OFF level as it solves the user program.
5. The resulting ON or OFF action, as a result of solving each rung, is sent to the output status file for storage.
6. During the output update portion of the scan, the processor will send the ON or OFF signal from each bit in the output status file to the associated output screw terminal by way of the output module.

Individual ladder-programming symbols are represented as instructions in the CPU section of Figure 6-2. The notations I:01, I:02, and O:01 represent the instructions and their addresses. When programming the PLC, these instructions are entered one by one and stored sequentially in the user program portion of the processor's memory. When the PLC is in run mode, these instructions are combined to arrive at the resulting ON or OFF state of each rung's output.

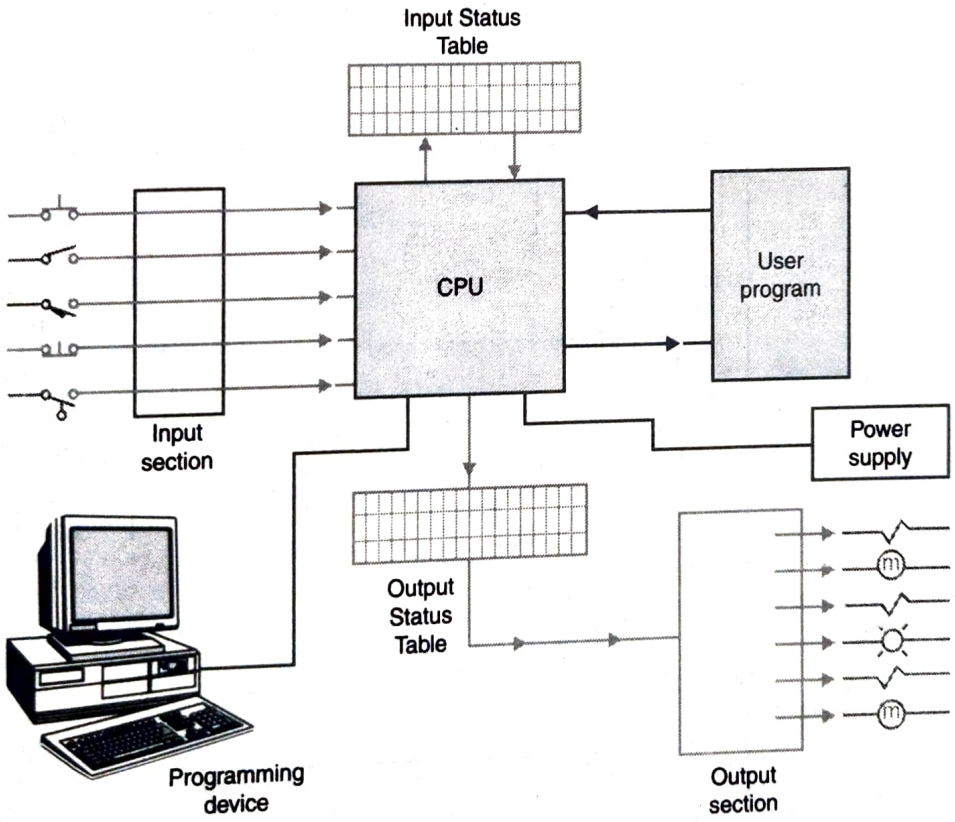

Figure 6-2 Signal flow into and out of a PLC. Notice that there is no electrical continuity between the inputs and the controlled output.

The PLC follows, or executes, the instructions stored in its memory the same way you might follow instructions to make, say, packaged grape drink. The package instructions instruct you to do the following:

1. Get one cup of sugar.
2. Put two quarts of water and the sugar into a container.
3. Add the contents of the package.
4. Mix until uniform.
5. Grape drink is ready to serve.

Following this procedure will provide an end product (grape drink). Likewise, the PLC follows the instructions programmed in its memory to achieve an end product.

WHAT IS LOGIC?

Devices in an electrical schematic diagram are described as being open or closed. PLC ladder instructions are typically referred to as either true or false. When a PLC solves the user program, it is said to be solving the ladder logic.

See Figure 6-3 for a look at a PLC ladder program printout.

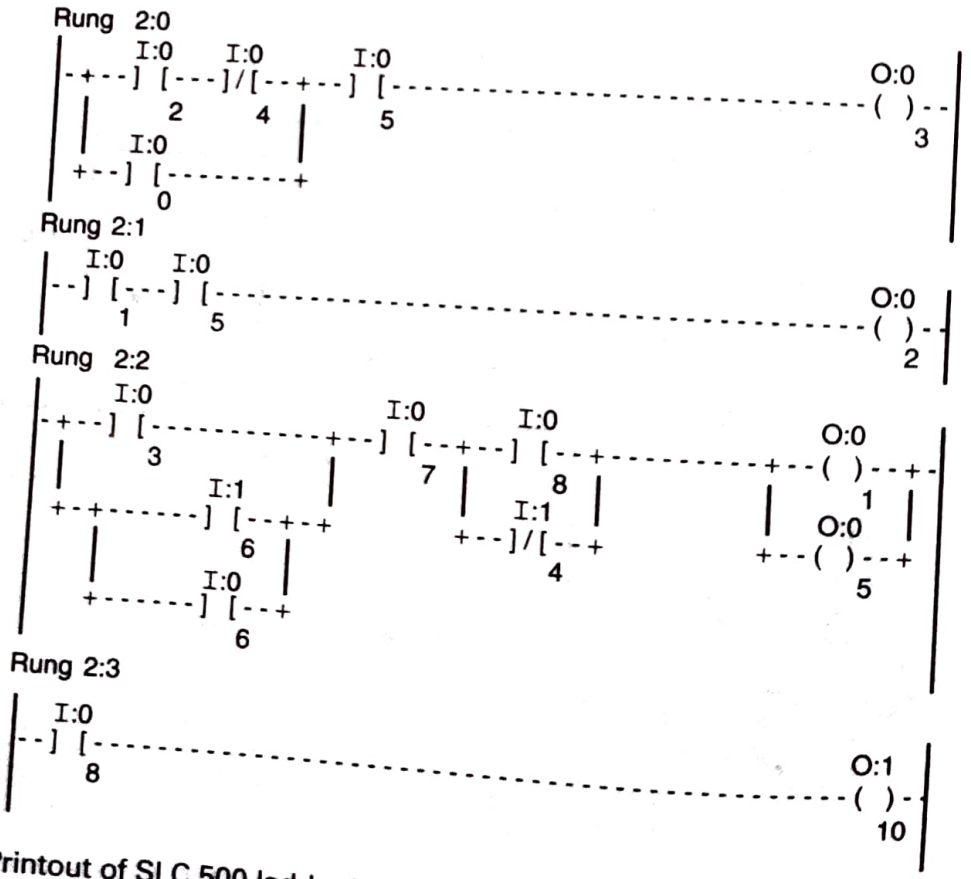

Figure 6-3 Printout of SLC 500 ladder logic developed on DOS-based Rockwell Software A.I. Series PLC 500 Ladder Logistics software.

Each rung is a program statement. A program statement consists of a condition, or conditions, along with some type of action. Inputs are the conditions, and the action, or output, is the result of the conditions. Each PLC ladder rung can be looked at as a problem the processor has to solve. The PLC combines ladder program instructions similar to the physical wiring hardware devices in series or parallel. However, rather than working in series or parallel, the PLC combines instructions logically using logical operators. Logical operations performed by a PLC are based on the fundamental logic operators: AND, OR, and NOT. These operators are used to combine the instructions on a PLC rung so as to make the outcome of each rung either true or false. The symbol that represents the result of solving the input logic on a particular rung is the output.

OVERVIEW OF LOGIC FUNCTIONS

To understand and program programmable controllers, we must understand basic logic. Three logic functions will be introduced here.

One Instruction Combined in Series with Another Is "AND"

You performed a logical operation when you mixed the grape drink. Although you might not realize it, you performed AND logic. Let's rewrite our drink-mixing task to see how it relates to AND logic:

1. Get one cup of sugar.
2. AND put two quarts of water AND the sugar into a container.
3. AND add contents of package.
4. AND mix until uniform.
5. Grape drink is ready to serve.

AND logic is similar to placement in series, as all series devices must pass continuity before the outcome will be allowed to happen. Likewise, when mixing the grape drink, all steps must be performed before a satisfactory beverage is produced.

One Instruction Combined in Parallel with Another Is OR

Our drink-mixing example can be made into OR logic (a parallel operation). When mixing our drink, there may be a choice whereby grape flavoring OR orange flavoring can be used. By following the same instructions but then choosing either grape flavor or orange flavor, OR logic is carried out.

The Opposite of a Normally Open Instruction Is a Normally Closed Instruction

If a set of normally open contacts is represented by a normally open instruction and a set of normally closed contacts is represented by a normally closed instruction, the normally open instruction must be energized to become true, while the normally closed instruction must not be energized to be true. Since there are only two states in digital logic, the normally closed instruction represents the opposite of the normally open instruction.

The term NOT is used for the normally closed instruction, as the input is not energized for the normally closed instruction to be true. NOT logic also refers to the normally closed instruction because the output is the inverse of the input. If the input is true, the instruction will be evaluated as NOT true. If the input is NOT true, the instruction will be evaluated as true. As a result of the input instruction being the opposite, or inverted, in relation to the instruction's status, NOT logic is also referred to as an inverter. Different manufacturers will refer to the normally closed instruction and its logical function with different terminology; however, all terms work in the same manner.

Let's explore these logic functions and see how they relate to PLC ladder programs.

SERIES—THE *AND* LOGIC FUNCTION

The old familiar series circuit can also be referred to as an AND logic function. In the series circuit (Figures 6-4 and 6-5), switch 1 AND switch 2 must be closed to have electrical continuity. When there is electrical continuity, output (light 1) will energize. The key word here is AND.

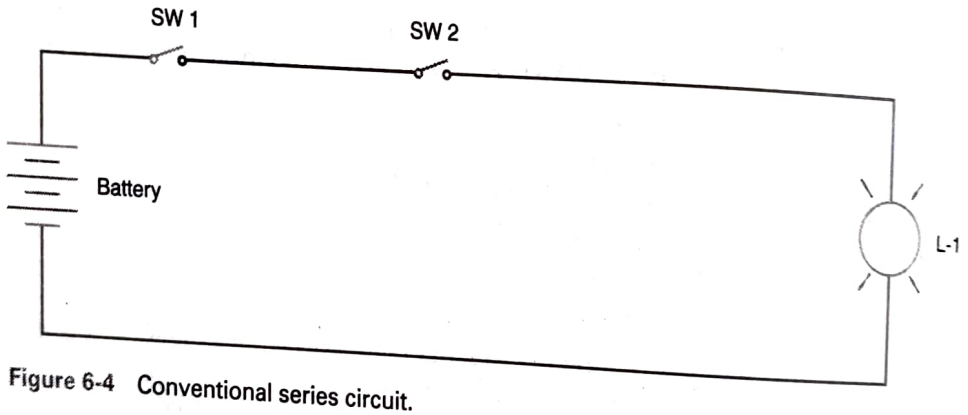

Figure 6-4 Conventional series circuit.

The circuit in Figure 6-4 is represented as a schematic diagram ladder rung in Figure 6-5.

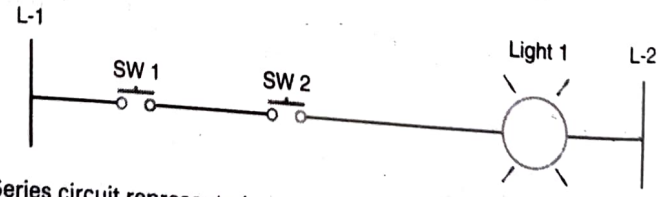

Figure 6-5 Series circuit represented as a conventional ladder rung.

Closing switch 1 and switch 2 will provide power, or electrical continuity, to L-1. This is illustrated in Figure 6-6.

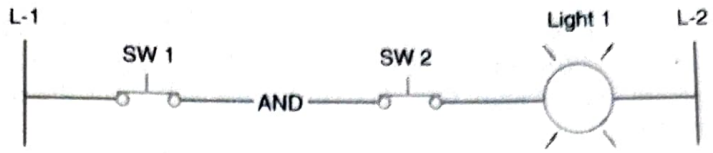

Figure 6-6 Switch 1 AND Switch 2 need to be closed to energize L-1.

Let us look at all the possible combinations that switch 1 (SW 1) and switch 2 (SW 2) can have, and the resulting output signals (Figure 6-7).

POSSIBLE SWITCH CONDITIONS AND RESULTING OUTPUT		
SW 1	SW 2	Light 1
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Figure 6-7 Truth table for AND logic.

Figure 6-7 is called a **truth table.** All possible input configurations for switches 1 and 2 are listed in the two left-hand columns. The expected output signal for light 1 is listed in the right-hand column. From the truth table, you can see that only when switch 1 AND switch 2 are ON will the output (light 1) energize.

Figure 6-8 is an example of the ladder program instructions that would be entered using a handheld programmer and an Allen-Bradley MicroLogix 1000. The ladder rung in Figure 6-8 is identical to that in Figure 6-6 except that the symbols have been changed to PLC ladder format.

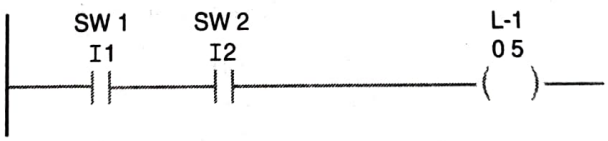

Figure 6-8 PLC representation of the rung represented in Figure 6-6.

Here is the program listing for the MicroLogix 1000 if you are entering the program with a handheld programmer:

LOAD I1
AND I2
OUT O5

The instructions tell the processor to load input 1 (I1) into memory, AND it with input 2 (I2), and then output the result to output 5 (O5). The resulting output will be determined by the truth table (Figure 6-7). The truth table represents the rules for ANDing two inputs together.

Before we look at the truth table in Figure 6-9, let's update our input status from OFF/ON to a more commonly accepted form. OFF is the same as no power or power not ON, and can be represented by the symbol 0. ON is the same as power ON, which can be represented by the symbol 1. We will use the commonly accepted symbol 1 to represent the presence of a valid signal and the commonly accepted symbol 0 to represent the absence of a valid signal.

Our truth table for the previous example would look as follows (Figure 6-9):

TWO-INPUT *AND* TRUTH TABLE		
SW 1	SW 2	Light 1
0	0	0
0	1	0
1	0	0
1	1	1

Figure 6-9 Truth table for two-input AND logic. This truth table says the same thing as the one in Figure 6-7.

The truth table in Figure 6-9 can also be represented as shown in Figure 6-10.

First condition
FF = ?
False AND False = False
or
0 AND 0 = 0

Second condition
FT = ?
False AND True = False
or
0 AND 1 = 0

Third condition
TF = ?
True AND False = False
or
1 AND 0 = 0

Fourth condition
TT = ?
True AND True = True
or
1 AND 1 = 1

Figure 6-10 Explanation of Figure 6-9 truth table.

THREE-INPUT *AND* LOGIC

Figure 6-11 has three switches in series controlling the load L-1. The conventional series circuit is shown in Figure 6-12. Figures 6-11 and 6-12 state that switch 1 AND switch 2 AND switch 3 must be energized before output L-1 will occur.

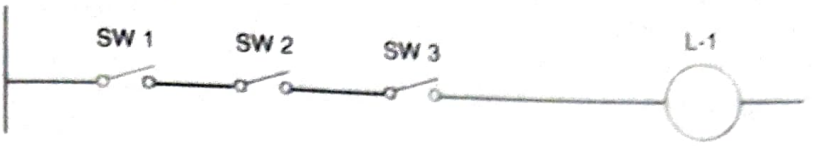

Figure 6-11 Three-ihput series circuit.

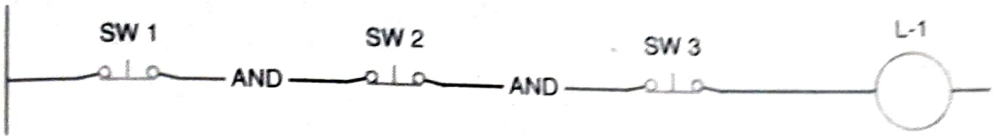

Figure 6-12 Three-input AND circuit.

Figure 6-13 illustrates the converted PLC ladder rungs for the SLC 500 PLC as the top rung. The center rung is for a PLC 5, and the bottom rung is for the ControlLogix PLC. Each rung has its respective input and output addresses.

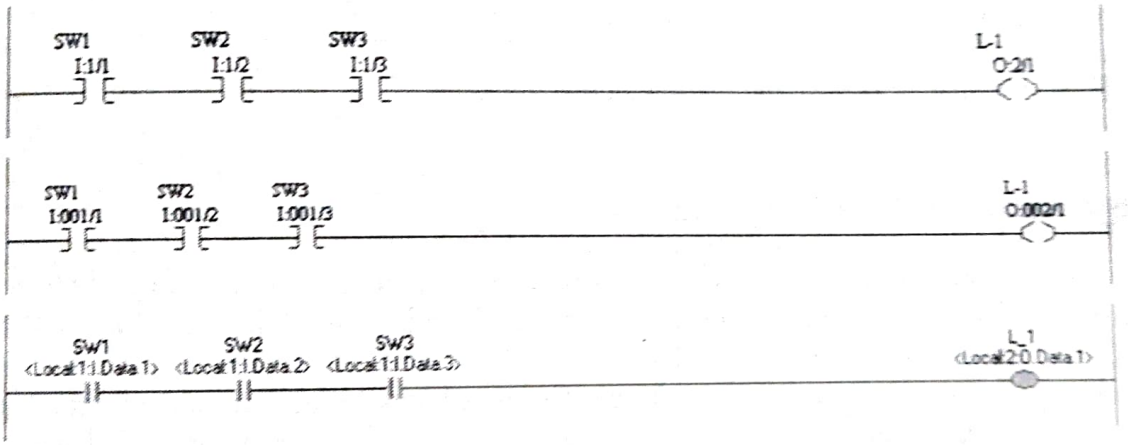

Figure 6-13 PLC three-input AND logic.

The following is the program listing for Figure 6-13 if you are programming with the Allen-Bradley MicroLogix 1000 and entering the program with a handheld programmer:

LOAD I1
AND I2
AND I3
OUT O1

The instructions tell the processor to load input 1 (I1) into memory, AND it with input 2 (I2), AND the result of the previous logical operation with input 3 (I3), and OUTPUT the result to output 1 (O1).

Let's assume that all three inputs are false. If I1, which is false, is loaded into memory and then ANDed with I2, which is also false, the result of this logical AND operation is false. Now we AND I3, which is false, to the result of I1 and I2, which was false. ANDed with I3, the output is also false.

Figure 6-14 is a truth table illustrating the expected outputs for three-input AND logic.

THREE-INPUT *AND* LOGIC			
Switch 1	Switch 2	Switch 3	Light 1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Figure 6-14 Three-input AND logic truth table.

FUNCTION BLOCK DIAGRAM *AND* LOGIC

Rockwell Automation's ControlLogix PLC family can be programmed in function block diagram language in addition to standard ladder logic. The PLC 5 and SLC 500 families do not support function block.

The principles of combining function block inputs are basically the same as ladder logic. Instead of normally open ladder logic symbols, boxes are used and referred to as function blocks. Figure 6-13 illustrates three rungs, three-input AND ladder logic. A function block diagram will represent the same logic using a Boolean AND (BAND) function block. Figure 6-15 illustrates a BAND function block. Notice the three blocks to the left of the BAND function block; they have Data.1, Data.2, and Data.3 as part of the information inside. These are function block input references. Refer back to Chapter 3 to review basic function block diagram elements. Input references represent the input address where the information is coming from. ControlLogix I/O addressing is a little different than traditional PLC addressing. Input and output addresses for the ControlLogix platform are referred to as tags. The ControlLogix PLC identifies physical inputs and outputs

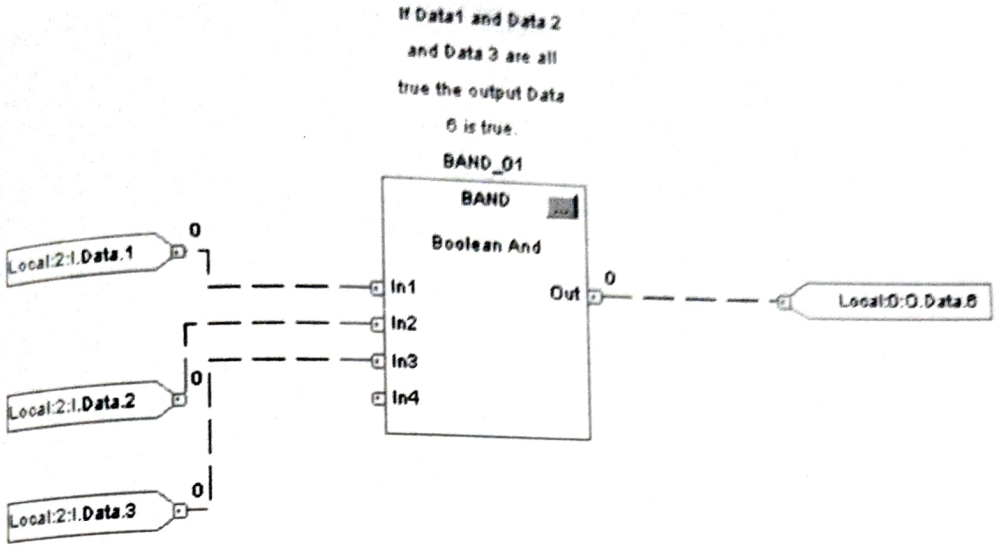

Figure 6-15 Function block BAND.

as data. In this example we are going to AND the three input references together. Here we are looking to see if Data.1 AND Data.2 AND Data.3 are all true. That is the job of the BAND block.

On the left side of the function block are the input points to the block. These are called pins. In 1 through In 4 are shown. This function block can contain up to eight input pins. In this example, since there are only three input references, only three input pins are used. When the inputs represented by Data.1 AND Data.2 AND Data.3 are true the BAND function block will be true. Being true, the output (out pin) of the block will be true. The dotted line or wire connected between the output pin and the output reference symbol identified as Data.6 represents the output data as a bit. Notice the 0 just to the right of the input reference and out pin. This identifies the logical state of the input reference or output pin for the instruction. A zero identifies the reference tag as false, while a one signifies the reference tag is true. In this example each input reference has a zero representing its input state as false. Since we do not have logical continuity, when executed, the BAND block will mark its output as false, a zero. As a result, the associated output reference and its tag will also be false. This function block diagram is equivalent to the ControlLogix rung from Figure 6-13.

Clicking on the View Properties box in the upper right-hand corner of the BAND function block reveals the BAND Properties view as illustrated in Figure 6-16. Notice that there are eight input pins for this function block. They are named In1 through In8. Checking the boxes in the viability (Vis) column turns on the four input pins and displays them on the function block. Since In5 through In8 are not checked, these pins will not be displayed on the function block. This illustrates how the programmer can select the function block options that are specifically needed for this application. Notice that the Out pin is also checked. Other properties and their associated pins that are not checked will not be displayed on the function block.

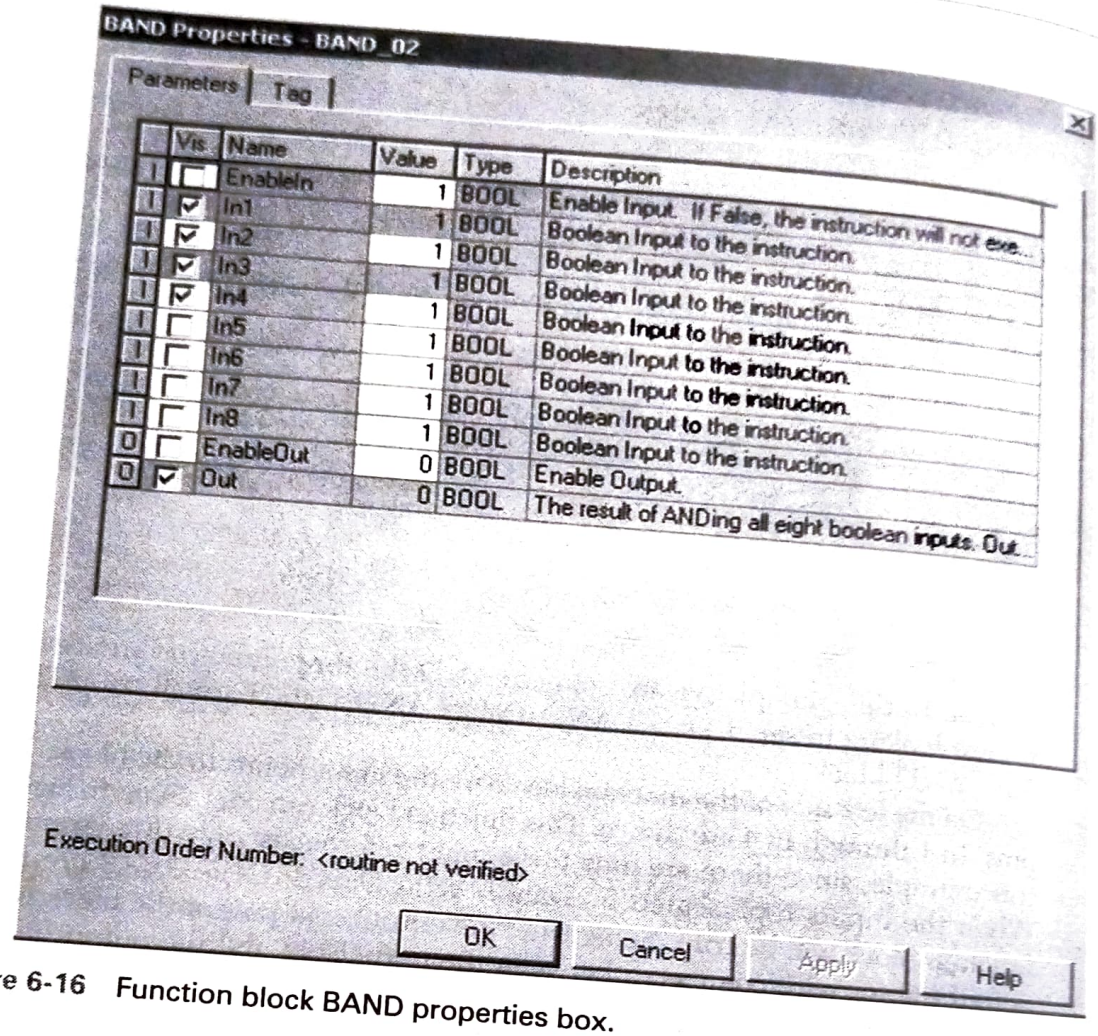

Figure 6-16 Function block BAND properties box.

PARALLEL CIRCUITS—THE *OR* LOGIC FUNCTION

The familiar parallel circuit can also be referred to as the OR logic function. The rule of OR logic is that if any input is true, the output will also be true. OR logic also states that if all inputs are true, the output will be true. In Figure 6-17, if switch 1 OR switch 2 is energized, light 1 will energize. If both SW 1 and SW 2 are true, the output will also be true.

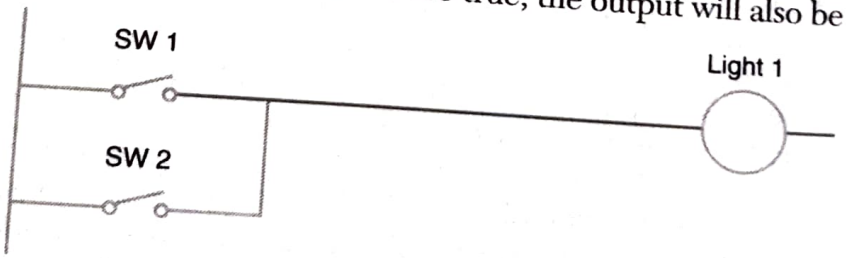

Figure 6-17 Conventional parallel circuit where switch 1 or switch 2 can energize the load, light 1.

Figure 6-18 illustrates Figure 6-17 converted to a PLC ladder rung. Remember, when drawing programmable controller ladder diagrams, do not use the conventional switch symbols such as we employed in the previous examples. A PLC rung of logic will have normally open or normally closed contacts instead of normally open or closed switch symbols. Addresses and instructions are included. Text information in addition to each contact and its address, such as SW 1, SW 2, and L-1, is referred to as instruction comments. Instruction comments can be added from programming software.

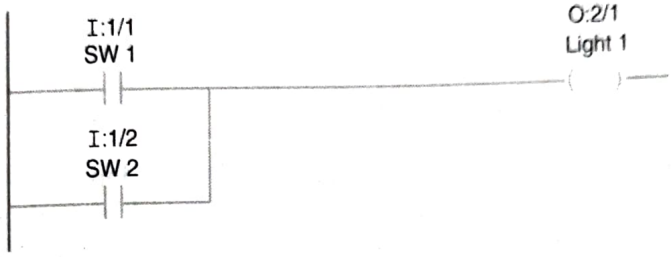

Figure 6-18 Programmable controller ladder diagram symbology. Addressing format is from the Allen-Bradley SLC 500 family of PLCs. Both ladder diagrams, Figures 6-17 and 6-18, are the same circuit, but with different symbols.

A two-input OR truth table representing Figures 6-17 and 6-18 is illustrated in Figure 6-19.

TWO-INPUT *OR* TRUTH TABLE		
SW 1	SW 2	Light 1
0	0	0
0	1	1
1	0	1
1	1	1

Figure 6-19 Two-input OR truth table.

Notice that in the case of the OR circuit, if either switch is ON, the output will be true. In addition, if both switches are ON, the output will be true.

Figure 6-20 shows a three-input parallel circuit, using three-input OR logic.

The following is a PLC program listing from Figure 6-20 for the Allen-Bradley MicroLogix 1000 if you are entering the program with a handheld programmer:

LOAD I1
OR I2
OR I3
OUT O2

The instructions tell the processor to load input 1 (I1) into memory, OR it with input 2 (I2), then OR the resultant of the previous logic operation with input 3 (I3), and then output the result to output 2 (O2).

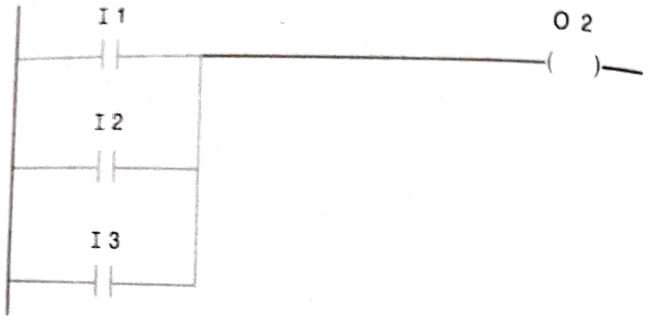

Figure 6-20 Three-input OR logic.

Let's assume that inputs I1 and I3 are false, and input I2 is true. If I1, which is false, is loaded into memory and then ORed with I2, which is true, the result of this logical OR operation is true. Now we OR I3, which is false, with the result of I1 and I2. That resultant was true; ORed with I3, which is false, its output is true. Remember the rule of OR logic: If any or all inputs are true, the output is true.

What are the expected outputs from three-input OR logic for Figure 6-21? Remember that if one or more inputs are true, the output will be true.

THREE-INPUT *OR* LOGIC			
Switch 1	Switch 2	Switch 3	Light 1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Figure 6-21 Three-input OR logic truth table.

To this point we have been working with normally open inputs. For a normally open input to be true, there must be power to the PLC input screw terminal. This ON signal to the input module will result as a 1 in the associated input status file bit position. A 1 in the input status file will cause the associated normally open PLC ladder instruction to close, or become true.

Typically, when a logical 1 in the input status table is associated with a normally open instruction, something is expected to operate. Likewise, if a logical 0 from the input status table is associated with a normally open instruction, something is expected to become false, or turn off.

The function block Boolean OR (BOR) is illustrated in Figure 6-22. This function block has the same functional components as the BAND. For this example if Data.1 OR Data.2 OR Data.3 are true, the output of the BOR function block will be true, or a 1. As with ladder OR logic, if any combination of input references is true the function block will be true. With the function block true, the output reference tag Data.6 will be true.

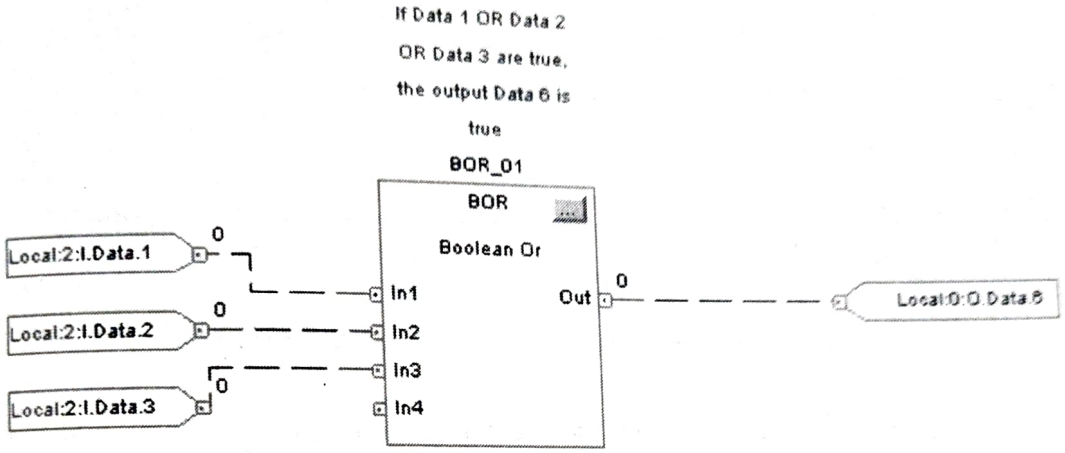

Figure 6-22 Function block BOR.

The NOT logic operator works in a manner opposite from the AND and OR logic with which we have been working. The next section will introduce the normally closed instruction and NOT logic.

NOT LOGIC

A normally closed hardwire relay contact passes power any time the relay coil is not energized. Likewise, the normally closed PLC ladder logic instruction will pass power any time the input status file bit is not a 1. This means that the physical hardware input is not sending an input signal into the PLC's input module. NOT logic is the opposite of a normally open PLC instruction or contact. It can be used in conjunction with AND or OR logic when a logical 0 in the status file is expected to activate some output device. The NOT logic function is used when an input must not be energized for an output to be energized. Likewise, the NOT logic function is used when a logical 1, or true input, is necessary to make the instruction false or deactivate an output device.

The truth table in Figure 6-23 simply states that a normally closed instruction on a PLC ladder rung will be the inverse, or opposite, of the input status table bit associated with the specific instruction. If the input status table bit is a 1, or true, the normally closed instruction will be false. In comparison, when the input table status bit is false, or a 0, the associated normally closed instruction will be true.

The NOT logic function is somewhat difficult to grasp. Let's look a little closer at the relationship between the normally open contact and how it controls the output in comparison to the normally closed contact. Figure 6-24 illustrates two rungs, the first with a normally open instruction and the second with a normally closed instruction.

TWO-STATE LOGIC FUNDAMENTALS		
Input Signal to Input Module	Normally Open PLC Instruction	Normally Closed PLC Instruction
ON	TRUE	FALSE
OFF	FALSE	TRUE

Figure 6-23 Truth table for NOT logic.

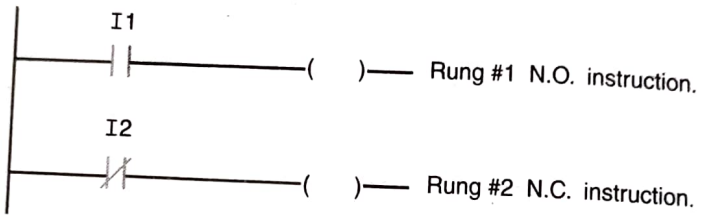

Figure 6-24 Conventional ladder diagram illustrating normally open and normally closed relay contacts controlling load, L-1.

ANALYSIS OF RUNG #1

Instruction I1 will energize the output only when there is a logical 1 in its associated input status file bit. A 1 in this bit position will cause the normally open instruction to become true and change state. In changing state, the instruction will allow logical continuity to pass on to the output instruction and make it true. Instruction I1 is considered true when it passes logical continuity. If there is no valid input signal from the field device attached to I1's screw terminal on the input module, a logical 0 will be placed in the input status file. A logical 0 in the input status file will result in this normally open input instruction becoming false. Being false, the instruction will not pass logical continuity.

ANALYSIS OF RUNG #2

The normally closed instruction works much like the normally closed contacts on a hardware relay. Being normally closed, instruction I2 will energize the output only when there is a logical 0 in its associated input status file bit. Even though there is a logical 0, or false input signal, in the status file, the normally closed instruction is true and passes logical continuity on to the output instruction. If there is a valid ON input signal from the field device attached to I2's input module screw terminal, a logical 1 will be placed in the input status file. A logical 1 in the input status file will cause the normally closed instruction to change state. The normally closed instruction will change from true (closed) to false (open). Being false, the normally closed instruction will not pass logical continuity to the output instruction. Without logical continuity, the output instruction will become false.

The function block Boolean NOT (BNOT) is illustrated in Figure 6-25. If the input reference representing input Data.1 is true, the output pin of the BNOT function block will be false, or a 0. See Figure 6-25. The output is the opposite of the input.

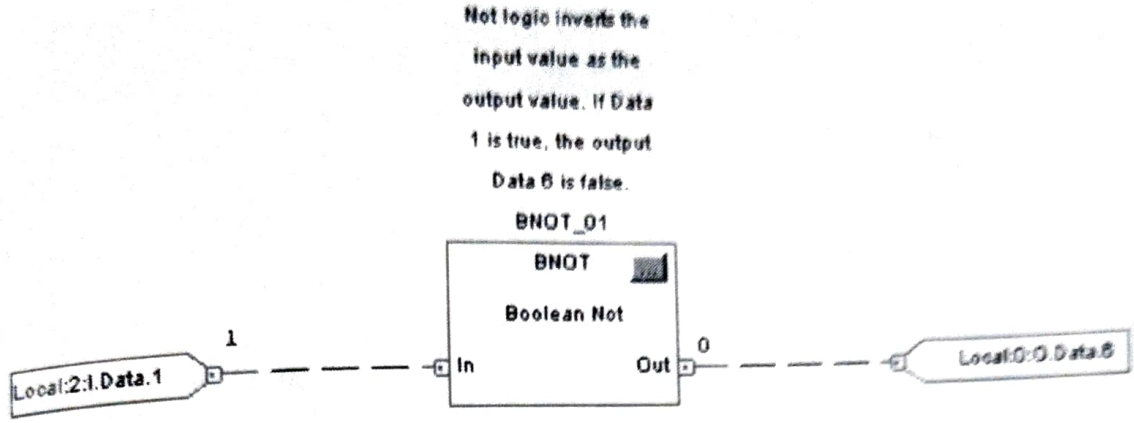

Figure 6-25 Function block NOT.

PARALLEL *NOT* LOGIC

A parallel ladder rung with normally closed inputs is a rung containing OR NOT logic.

Figure 6-26, a PLC ladder rung, has two input instructions, one normally open and one normally closed. This circuit contains parallel NOT logic. This conventional schematic rung will be true under the conditions shown in Figure 6-27. Input 1 must be true OR input 2 must NOT be true to make this rung true and energize output L-1.

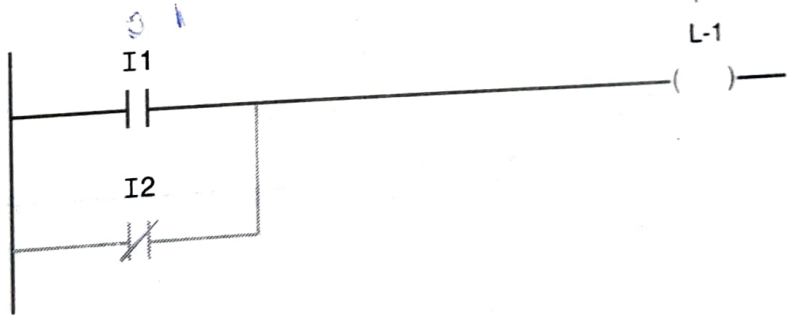

Figure 6-26 Parallel NOT logic.

INPUTS		INPUT STATUS FILE		OUTPUTS
I:1	I:2	I:1	I:2	O:0
0	0	0	1	1
0	1	0	0	0
1	0	1	1	1
1	1	1	0	1

NOT logic truth table.

Not all PLC manufacturers use the same terminology to identify the normally open and normally closed contact instruction. Figure 6-28 is a sample of terminology used with different PLCs.

DIFFERENT TERMINOLOGY USED FOR NORMALLY OPEN AND NORMALLY CLOSED INPUT INSTRUCTIONS	
—||—	Normally open Examine if closed AND
—|/|—	Normally closed Examine if open AND invert OR invert AND NOT

Figure 6-28 Normally open and normally closed instruction identification.

Let's look at the following ladder and program for an Allen-Bradley MicroLogix 1000 (see Figure 6-29). The program listing for Figure 6-25 is a MicroLogix 1000 ladder program when entering it with a handheld programmer. Notice that the normally closed instruction is referred to as "OR invert" (ORI).

LOAD I
OR I2
ORI I3
OUT O2

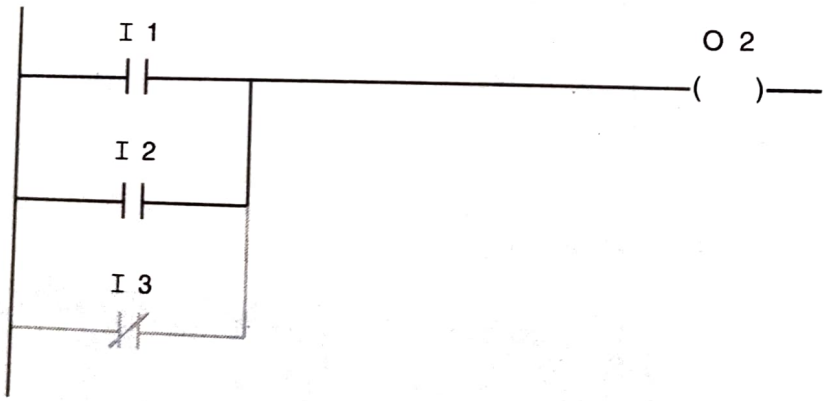

Figure 6-29 Three-input parallel NOT logic.

The instructions tell the processor to load input 1 (I1) into memory, OR it with input 2 (I2), then OR invert (OR a normally closed instruction), with the resultant of the previous logic with input 3 (I3) then being output to output 2 (O2).

OR logic states that when any or all inputs are true, the associated output will be true. Can we develop logic to give a true output if one or the other parallel inputs is true but not both? To solve this problem, we look at exclusive OR logic in the next section.

EXCLUSIVE *OR* LOGIC

Looking at a truth table for a two-input OR logic function, we see that there are three input conditions that will give us an output signal (see Figures 6-30 and 6-31):

1. If input 01 is off and input 02 is on.
2. If input 01 is on and input 02 is off.
3. If input 01 is on and input 02 is on.

Figure 6-30 Two-input OR logic.

TWO-INPUT *OR* TRUTH TABLE		
SW 1	SW 2	Light 1
0	0	0
0	1	1
1	0	1
1	1	1

Figure 6-31 Two-input OR truth table.

The exclusive OR logic function will allow either input 01 OR input 02, but not both together, to control the output (see Figure 6-32).

TRUTH TABLE FOR EXCLUSIVE *OR* LOGIC IN FIGURE 6-32		
I1	I2	O0
0	0	0
0	1	1
1	0	1
1	1	0

Figure 6-32 Exclusive OR logic truth table.

The logic for exclusive OR (sometimes referred to as X-OR) would look as follows (see Figure 6-33).

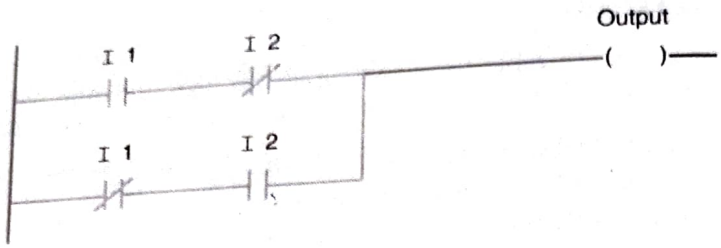

Figure 6-33 Exclusive OR logic.

ANALYSIS OF EXCLUSIVE *OR* LOGIC

If normally closed input I1 is true and input I2 is left as is, the logic on the main rung will become true, thus energizing the output. As for the state of the normally closed I1 instruction on the parallel branch, with the normally open input instruction true or closed, the normally closed contacts for input I1 on the parallel branch will open. With the normally closed contacts from input I1 open on the parallel branch, input I2 cannot control the output.

Input I2's logic will operate in the same manner. If I2's normally open instruction becomes true while I1's instructions remain in their normal state, the parallel branch will become true. With the parallel branch true, the rung will be true. The rung output will become true as there is logical continuity on the parallel branch. With the normally open I2 closed on the parallel branch, the normally closed I2 on the main rung will open and prevent I1 from controlling the output.

If, by chance, both input 1 and input 2 are energized (and therefore change from their normal state), their normally closed counterparts will both open. With an open on the main rung and the parallel branch, there is no way for the rung to become true. Recheck the truth table to verify this. Go through the logic yourself and check that if input 2 on the parallel branch is energized so that it is changed from its normal state and input 1 is left alone, the output will be energized as stated in the truth table.

Figure 6-34 illustrates the Function Block Boolean Exclusive Or (BXOR). If the input reference representing Data.1 OR the input reference representing Data.2 is true, but

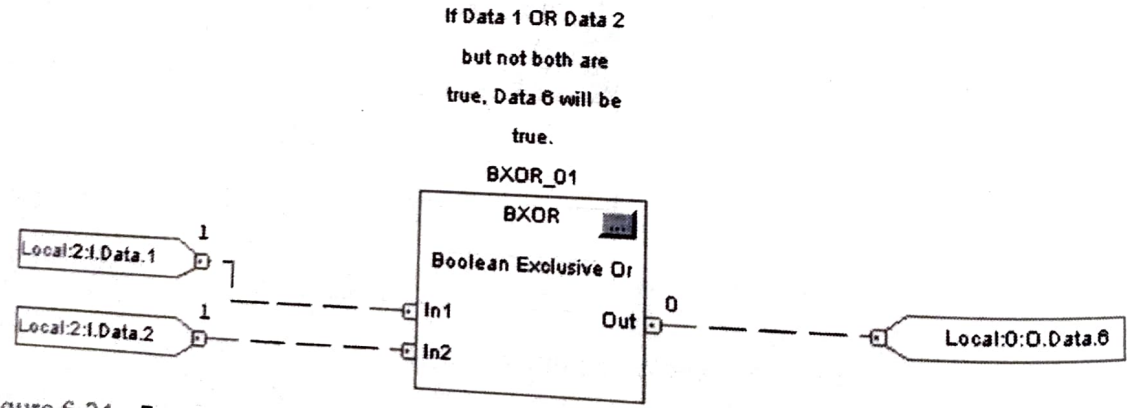

Figure 6-34 Function block XOR.

not both, the BXOR output will be true. The figure illustrates that both input references are true. As a result the output of the BXOR will be false. With the output false, output reference Data.6 will also be false.

COMBINATIONAL LOGIC

Most ladder rungs will include some combination of AND, OR, and NOT logic. No matter what the logic combination or how many logic elements or instructions are on a rung, there must be at least one path of true logical continuity before the output can be made true.

In Figure 6-35, the ladder rung has three logical paths by which it can be true:

1. If I1 AND I2 AND I4 are all true, output O2 will be true.
2. If I1 AND I3 AND I4 are all true, output O2 will be true.
3. If I1 AND I2 OR I3 AND I4 are all true, output O2 will be true.

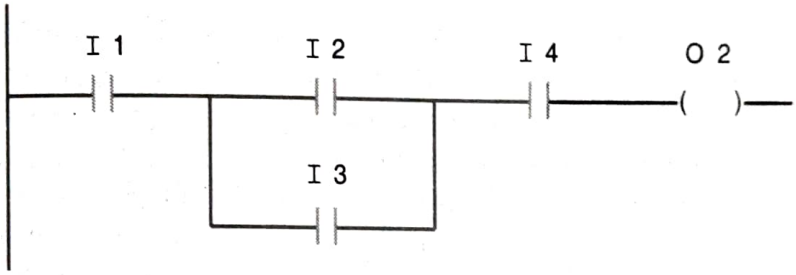

Figure 6-35 Combinational AND-OR logic.

In Figure 6-36, the ladder rung has four logical paths by which it can be true:

1. If I31 AND I10 AND I15 are all true, output O71 will be true.
2. If I4 AND I10 AND I15 are all true, output O71 will be true.
3. If I4 AND I10 AND I7 are all true, output O71 will be true.
4. If I31 AND I10 AND I7 are all true, output O71 will be true.

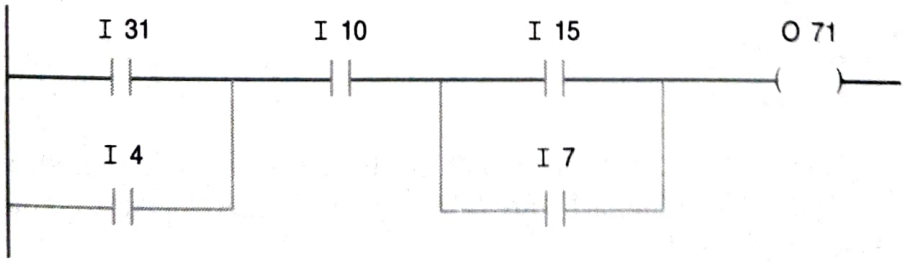

Figure 6-36 Combinational AND-OR logic. Four paths can make this rung logically true.

In Figure 6-37, the ladder rung has many logical paths by which it can be true. Study

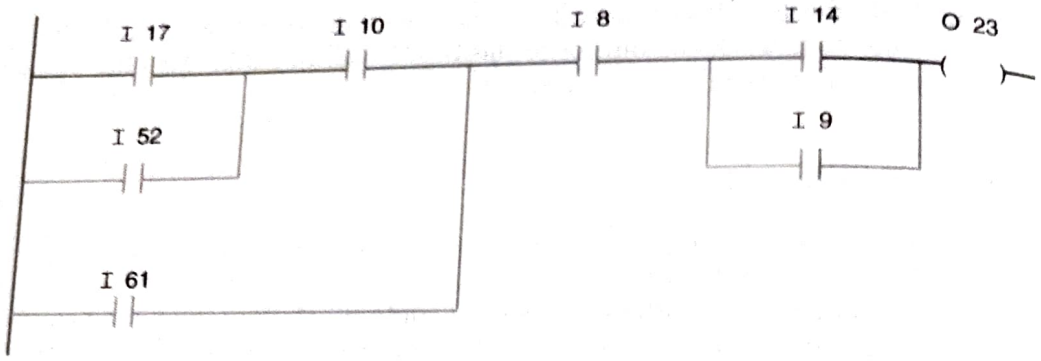

Figure 6-37 Combinational input logic.

PRIORITY OF LOGIC ELEMENTS

The priority of logic is very important in a program. This is especially important when entering a program with a handheld programmer. When using most handheld programmers, you actually enter the list of instructions representing the ladder logic for the current application. Some handheld programmers allow you to enter rungs of logic rather than a list of instructions. If you are able to enter the rungs of logic, the priority of logic concerns is taken care of by the programming device, which places the instructions in the proper position on the terminal's rungs. This is also true when programming PLC ladder rungs on a personal computer. When developing programs on a personal computer, the programmer actually develops the rungs and their associated instructions on the computer monitor's screen. Since the programmer is physically placing the instructions on the rung and in the correct position in relation to surrounding instructions, concerns about logic element priority are eliminated.

Evaluate the following program:

Load I1
AND I2
OR I3
AND I4
Out O7

The ladder rung in Figure 6-38 can be developed from the listed instructions.

The ladder rung in Figure 6-39 can also be developed from the listed instructions. Which is correct?

It should be evident that these two rungs of logic are not equivalent, even though they appear to have the same list of instructions. This is where logic priority becomes important. There need to be rules regarding instruction placement. The PLC was programmed to follow certain programming rules, and as a programmer, you must follow the same programming rules as the PLC. When working together with the same rules, human and

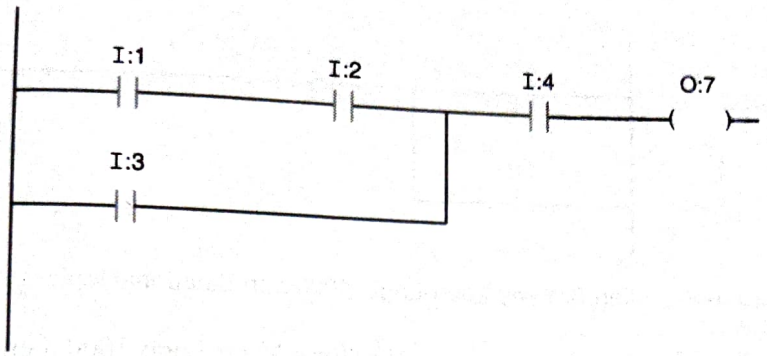

Figure 6-38 This ladder can be developed from the listed instructions.

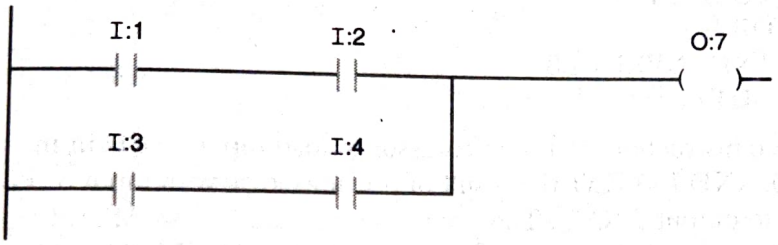

Figure 6-39 This ladder could also be created from the same listed instructions.

machine will understand each other and develop mutually acceptable programs. General programming rules are listed below:

1. Each rung begins at the left power rail.
2. Start each rung with the appropriate beginning instruction. This is typically a "Store" or "Load" instruction.
3. Program the next logic element closest to the one already programmed. In case a series and a parallel logic element are equidistant, always program the parallel logic function first. Continue following this rule until the output instruction is reached.
4. Typically, if two or more instructions are to be programmed on a parallel branch, such as in Figure 6-39, special instructions are used to connect, or group, parallel instructions on that branch. If no grouping instructions are included in the program, the default of one instruction per parallel branch, as illustrated in Figure 6-40, will be programmed.

Figure 6-40 shows a ladder and program for combinational logic programming with an Allen-Bradley MicroLogix 1000.

Rule three identifies the programming sequences when entering an instruction list program, such as in Figure 6-40. Instructions should be entered in the following order: I1, I2, I3, O2.

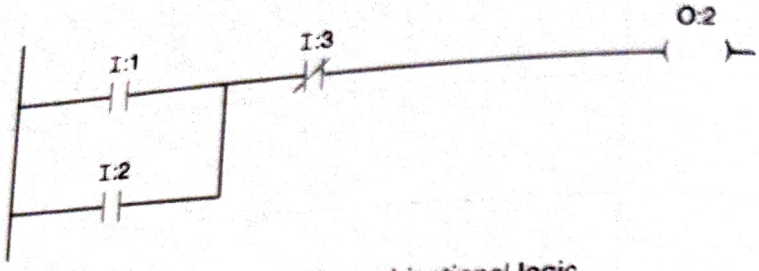

Figure 6-40 Allen-Bradley MicroLogix 1000 combinational logic.

The following is a program listing for a MicroLogix 1000 if entering the program with a handheld programmer. Notice that the normally closed instruction is referred to as an "AND invert" (ANI).

```
LOAD I:1
OR I:2
AND INVERT I:3
OUT O:2
```

These instructions tell the processor to load input 1 (I:1) in memory, OR it with input 2 (I:2), AND INVERT the result of previous logic with input 3 (I:3), and OUTPUT the result to output 2 (O:2). The programming rule for the MicroLogix 1000 states that if the programmer does not specify that the I2 AND INVERT I3 instructions are to be grouped together on the parallel branch, the instruction after OR I2 will default back to the main rung. This is illustrated in Figure 6-41.

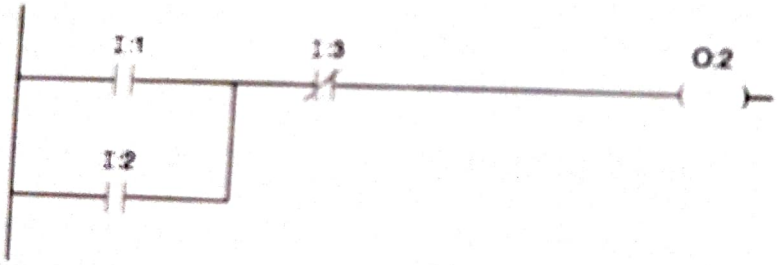

Figure 6-41 MicroLogix instruction list program containing a group of logic on the parallel branch.

Special grouping is accomplished through developing special groups of logic. Changing the programming instructions will direct the processor to evaluate the instructions in the desired manner. As an example, the following program will use the load instruction to separate the I:1 instruction from a second group starting with the LOAD I:2 instruction.

```
LOAD I:1
LOAD I:2          GROUP A
AND INVERT I:3    GROUP 2
OR Block
OUT O:2           Linking Instruction
```

The two groups of logic will be linked together using a special linking instruction, the "OR block" instruction. After the groups are linked together in parallel, the output instruction is evaluated (see Figure 6-42).

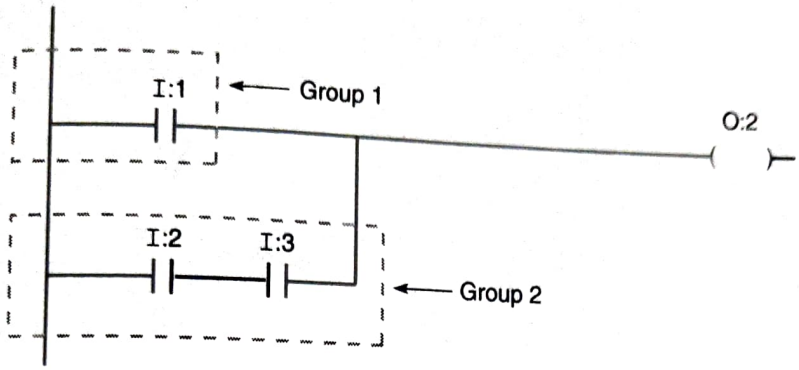

Figure 6-42 The OR block instruction links these two groups in parallel.

FUNCTION BLOCK ORDER OF EXECUTION

The order of function block is typically from input to output. Program your function blocks starting with one or more inputs and add additional blocks from left to right. Figure 6-43 illustrates two input references feeding into a BAND function block. The BAND function block feeds into a timer on-delay with reset (TONR). The TONR block's done output (DN) feeds into the output reference. Here the order of execution is from input to output, or left to right.

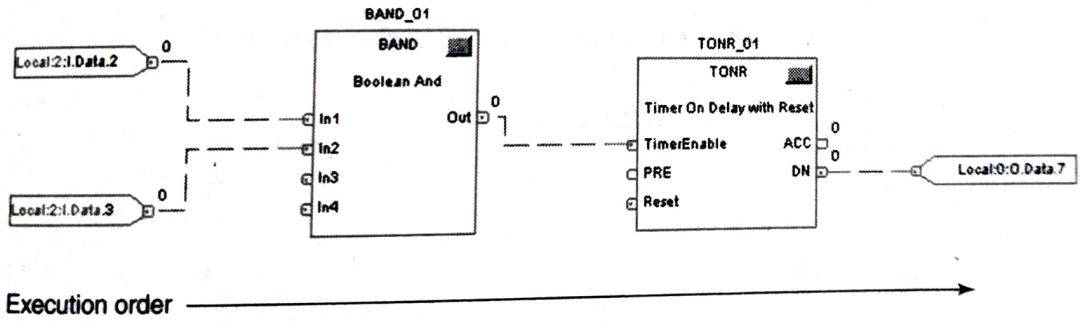

Figure 6-43 Execution order is from input to output in a simple function block diagram.

To view the execution order, go to the View Properties window. Figure 6-44 is a Properties view for the TONR block in our example. Note the execution order notation in the bottom left-hand corner. The execution order of this function block is 2.

Execution order is relative blocks that are wired together. Figure 6-45 illustrates two groups of blocks that are not wired together. The two groups of blocks are not related to each other. Each group of blocks will execute from right to left, or from input to output. Notice how the execution order alternates between groups of blocks.

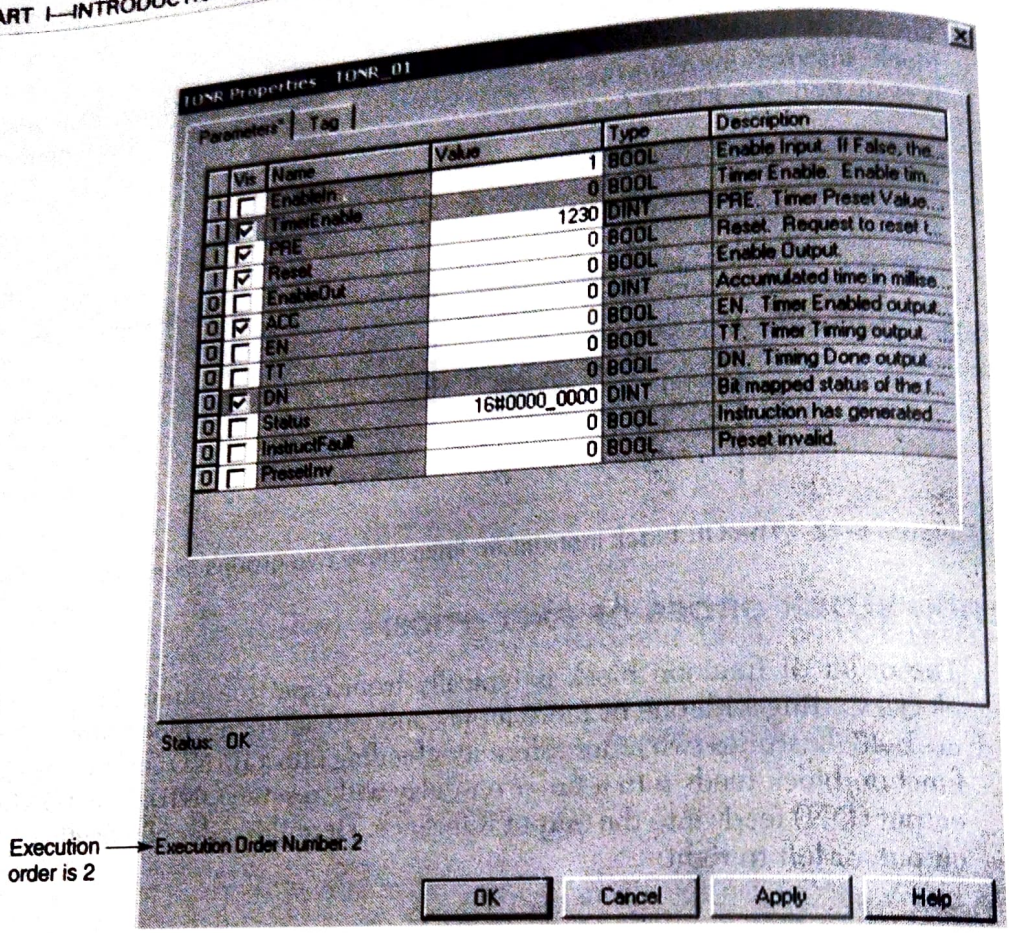

Figure 6-44 Execution order displayed in the TONR view properties.

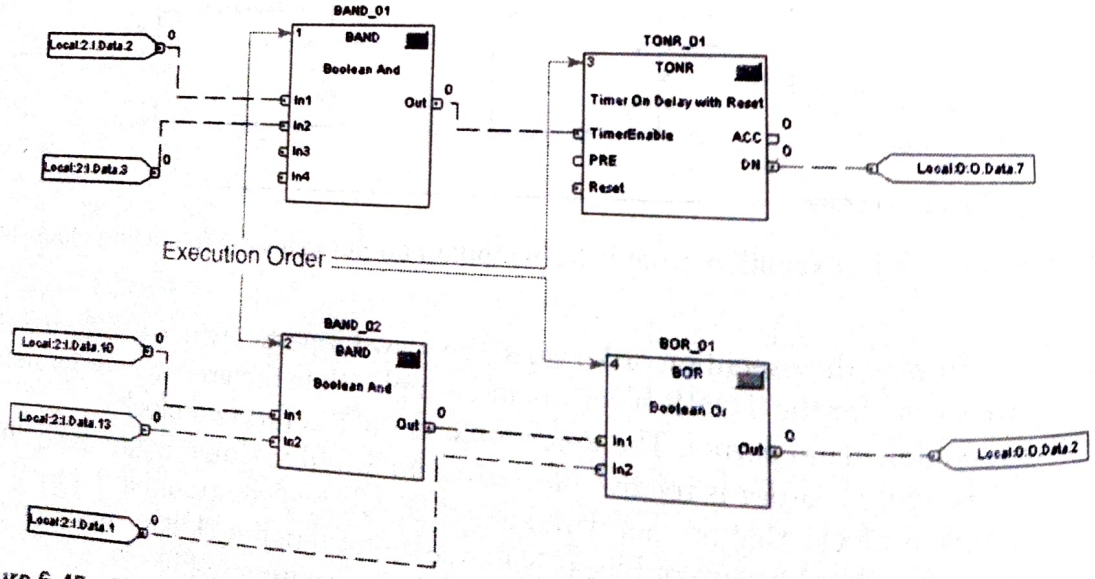

Figure 6-45 Execution order of unrelated groups of function blocks.

When function blocks are in a feedback loop, proper programming is required to identify which block is to be executed first. If not programmed properly, the processor cannot determine which block to execute first. To help the processor understand which block to execute first, place an assume data available indicator on the feedback input pin on the first block as illustrated in Figure 6-46 below.

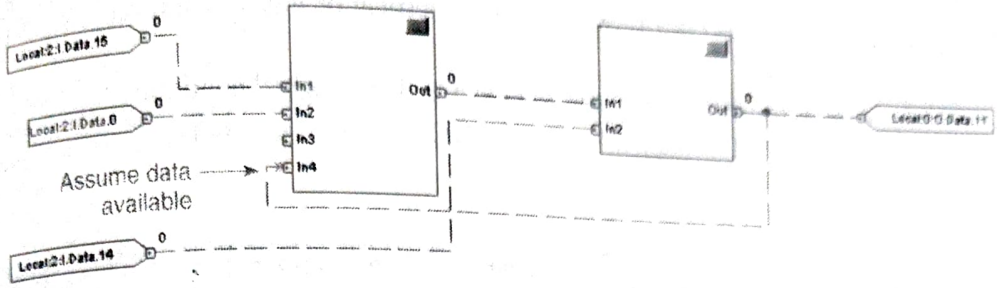

Figure 6-46 Use the assume data available indicator to identify which block the processor is to execute first.

SUMMARY

The main focus of this chapter was to introduce the concepts of how logic functions are executed when solving a PLC program. We looked at how a PLC solves its user program using AND, OR, and NOT logic. PLC ladder logic differs when compared to conventional relay logic in one important aspect: Relay logic has electrical continuity, whereas PLC logic has only logical continuity. Where there is actual current flow in a hardwired relay circuit, there is only logical continuity on a PLC rung.

Even though the PLC accepts the same signals and signal levels as relay circuits, the PLC input signal is transformed to a logical 1 or 0 as it is input to the input status file. The user program uses these logical 1, +5 VDC (ON or true), or logical 0, 0 VDC (OFF or false) signals from the input status file to solve the programmed instructions. The result of solving the user instructions using the basic logic operators, for a particular rung, is the output instruction becoming true or false. The resulting 0 (false) or 1 (true) signal is sent to the output status file, where it is stored. During the output update portion of the scan, the 0 or 1 is sent to the output module as either a 0 VDC level, for false, or +5 VDC level, for true.

REVIEW QUESTIONS

1. PLCs use which of the following logic operators?
   A. AND
   B. OR
   C. NOT
   D. all of the above
2. Each instruction on a ladder diagram has a reference number associated with it, which is:
   A. its instruction comment
   B. its position identifier on the ladder rung

C. a number referencing the instruction's associated input status table bit address
D. the instruction's input address.

3. True or false: Contacts shown on PLC ladder rungs can be in any one of the following states—ON, OFF, intermediate.
4. True or false: Programmable controller contacts and relay contacts operate in a similar manner, as both provide either electrical or logical continuity when the contacts are closed.
5. What is the difference between electrical and logical continuity?
6. Fill in the following truth table for NOT logic (Figure 6-47).

ON		
OFF		

Figure 6-47 NOT logic truth table.

7. The logical AND function is similar to:
   A. in parallel
   B. in series
   C. inverted logic
   D. both A and C
   E. depends on the application
8. The logical OR function is similar to:
   A. in parallel
   B. in series
   C. inverted logic
   D. both A and C
   E. depends on the application
9. The logical OR NOT function is similar to:
   A. in parallel
   B. in series
   C. inverted logic
   D. both A and C
   E. depends on the application
10. Illustrate a two-input AND logic PLC ladder rung.
11. Develop a truth table for the answer in 10.
12. Will the following rung be true or false (see Figure 6-48)?

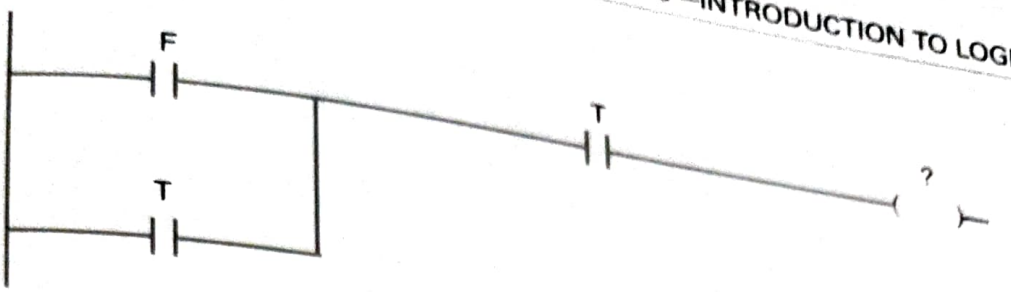

Figure 6-48 Rung for question 12.

13. The BAND function block is similar to:
    A. series ladder logic
    B. parallel ladder logic
    C. NOT ladder logic
    D. Boolean ladder logic
14. The BAND function block:
    A. ANDs up to twelve inputs
    B. ANDs up to eight input references
    C. logically combines up to eight input references
    D. performs a logical AND/OR function
15. A BXOR function block:
    A. is true when all inputs are true
    B. is true when neither input is true
    C. is true only when one or the other input pins is true
    D. is false when only one input is true
    E. is false when both input references are true
    F. C and E are correct
    G. B and D are true
16. A BNOT function block is true:
    A. when both inputs are true
    B. when the function block is true
    C. when the order of execution is true
    D. when the input is false
    E. both B and C
17. The order in which a series of function blocks are executed:
    A. is determined by the programmer
    B. is determined by the position of the block on the sheet
    C. typically flows from right to left
    D. typically flows from inputs to output
18. To resolve a loop:
    A. the instructions are executed in order from left to right
    B. the processor uses the assume data available indicator to know when to start
    C. the processor looks at the output block to determine the state of the feedback
    D. the processor waits until a start signal is received from the output block