CHAPTER 11

Ladder and Functional Block Programming

W. Bolton

This (and the following) chapter comes from the book Programmable Logic Controllers *by W. Bolton, ISBN: 9780750681124. The first edition of the book was published in 1996, which explains why the author commences the preface by saying: "Technological advances in recent years have resulted in the development of the programmable logic controller (PLC) and a consequential revolution of control engineering." (Don't worry; the chapter presented here is from the fourth edition in 2006.)*

The thing is that this is something of a timeless subject, so this book provides a very useful introduction to programmable logic controllers and aims to ease the tasks of practicing engineers coming first into contact with programmable logic controllers. It addresses the problem of different programmable control manufacturers using different nomenclature and program forms by describing the principles involved and illustrating them with examples from a range of manufacturers.

To assist the reader to develop the skills necessary to write programs for programmable logic controllers, many worked examples, multi-choice questions and problems are included in the book with answers to all multi-choice questions and problems given at the end of the book.

Now, Programmable Logic Controllers is not related to FPGAs per se, so you may be wondering why this chapter is included here. The answer is simple—I personally have almost invariably found that anything I learned came in useful at some stage in my career. In order to facilitate PLCs being used by engineers without any great knowledge of conventional programming languages and techniques, an approach called "ladder programming" was developed. This is a means for nonprogrammers to create programs *that can then be converted into machine code for use by the PLC microprocessor. This method of capturing programs has been adopted by most PLC manufacturers.*

I found this to be very interesting and I hope you will agree with me (see also the next chapter in which we see examples of this technique being used in real-world applications).

—Clive "Max" Maxfield

Programs for microprocessor-based systems have to be loaded into them in *machine code*, this being a sequence of binary code numbers to represent the program instructions. However, *assembly language* based on the use of mnemonics can be used; for example, LD is used to indicate the operation required to load the data that follows the LD, and a computer program called an assembler is used to translate the mnemonics into machine code. Programming can be made even easier by the use of the so-called *high-level languages*, such as C, BASIC, PASCAL, FORTRAN, COBOL. These use prepackaged functions, represented by simple words or symbols descriptive of the function concerned. For example, with C language the symbol & is used for the logic AND operation. However, the use of these methods to write programs requires some skill in programming and PLCs are intended to be used by engineers without any great knowledge of programming. As a consequence, *ladder programming* was developed. This is a means of writing programs which can then be converted into machine code by some software for use by the PLC microprocessor.

This method of writing programs became adopted by most PLC manufacturers, however each tended to have developed their own versions and so an international standard has been adopted for ladder programming and indeed all the methods used for programming PLCs. The standard, published in 1993, is IEC 1131-3 (International Electrotechnical Commission). The IEC 1131-3 programming languages are ladder diagrams (LAD), instruction list (IL), sequential function charts (SFC), structured text (ST), and function block diagrams (FBD).

This chapter is an introduction to the programming of a PLC using ladder diagrams and functional block diagrams, with discussion of the other techniques in the next chapter. Here we are concerned with the basic techniques involved in developing ladder and function block programs to represent basic switching operations, involving the logic functions of AND, OR, Exclusive OR, NAND and NOR, and latching.

11.1 Ladder Diagrams

As an introduction to ladder diagrams, consider the simple wiring diagram for an electrical circuit in Figure 11.1a. The diagram shows the circuit for switching on or off an electric motor. We can redraw this diagram in a different way, using two vertical lines to represent the input power rails and stringing the rest of the circuit between them. Figure 11.1b shows the result. Both circuits have the switch in series with the motor and supplied with electrical power when the switch is closed. The circuit shown in Figure 11.1b is termed a *ladder diagram.*

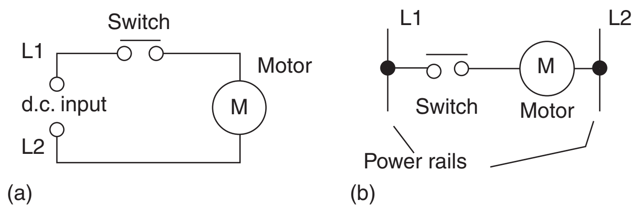


Figure 11.1: Ways of drawing the same electrical circuit

With such a diagram the power supply for the circuits is always shown as two vertical lines with the rest of the circuit as horizontal lines. The power lines, or rails as they are often termed, are like the vertical sides of a ladder with the horizontal circuit lines like the rungs of the ladder. The horizontal rungs show only the control portion of the circuit; in the case of Figure 11.1 it is just the switch in series with the motor. Circuit diagrams often show the relative physical location of the circuit components and how they are actually wired. With ladder diagrams no attempt is made to show the actual physical locations and the emphasis is on clearly showing how the control is exercised.

Figure 11.2 shows an example of a ladder diagram for a circuit that is used to start and stop a motor using push buttons. In the normal state, push button 1 is open and push button 2 closed. When button 1 is pressed, the motor circuit is completed and the motor starts. Also, the holding contacts wired in parallel with the motor close and remain closed as long as the motor is running. Thus when the push button 1 is released, the holding contacts maintain the circuit and hence the power to the motor. To stop the motor, button 2 is pressed. This disconnects the power to the motor and the holding contacts open. Thus when push button 2 is released, there is still no power to the motor. Thus we have a motor which is started by pressing button I and stopped by pressing button 2.

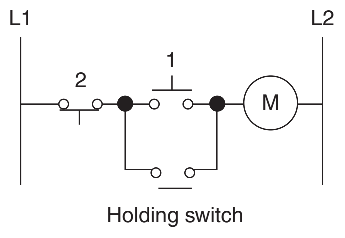


Figure 11.2: Stop-start switch

11.1.1 PLC Ladder Programming

A very commonly used method of programming PLCs is based on the use of *ladder diagrams*. Writing a program is then equivalent to drawing a switching circuit.
The ladder diagram consists of two vertical lines representing the power rails. Circuits are connected as horizontal lines, i.e., the rungs of the ladder, between these two verticals.

In drawing a ladder diagram, certain conventions are adopted:

1. The vertical lines of the diagram represent the power rails between which circuits are connected. The power flow is taken to be from the left-hand vertical across a rung.
2. Each rung on the ladder defines one operation in the control process.
3. A ladder diagram is read from left to right and from top to bottom, Figure 11.3 showing the scanning motion employed by the PLC. The top rung is read from left to right. Then the second rung down is read from left to right and so on.

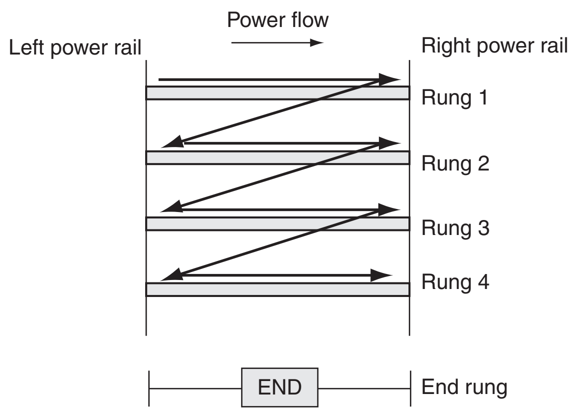


Figure 11.3: Scanning the ladder program

When the PLC is in its run mode, it goes through the entire ladder program to the end, the end rung of the program being clearly denoted, and then promptly resumes at the start. This procedure of going through all the rungs of the program is termed a *cycle*. The end rung might be indicated by a block with the word END or RET for return, since the program promptly returns to its beginning.

4. Each rung must start with an input or inputs and must end with at least one output. The term input is used for a control action, such as closing the contacts of a switch, used as an input to the PLC. The term output is used for a device connected to the output of a PLC, e.g., a motor.
5. Electrical devices are shown in their normal condition. Thus a switch, which is normally open until some object closes it, is shown as open on the ladder diagram. A switch that is normally closed is shown closed.
6. A particular device can appear in more than one rung of a ladder. For example, we might have a relay that switches on one or more devices. The same letters and/or numbers are used to label the device in each situation.
7. The inputs and outputs are all identified by their addresses, the notation used depending on the PLC manufacturer. This is the address of the input or output in the memory of the PLC.

Figure 11.4 shows standard IEC 1131-3 symbols that are used for input and output devices. Some slight variations occur between the symbols when used in semi-graphic form and when in full graphic. Note that inputs are represented by different symbols representing normally open or normally closed contacts. The action of the input is equivalent to opening or closing a switch. Output coils are represented by just one form of symbol.

To illustrate the drawing of the rung of a ladder diagram, consider a situation where the energizing of an output device, such as a motor, depends on a normally open start switch being activated by being closed. The input is thus the switch and the output the motor. Figure 11.5a shows the ladder diagram.

Starting with the input, we have the normally open symbol | | for the input contacts. There are no other input devices and the line terminates with the output, denoted by the symbol ( ). When the switch is closed, i.e., there is an input, the output of the motor is activated. Only while there is an input to the contacts is there an output. If there had been a normally closed switch |/| with the output (Figure 11.5b), then there would have been an output until that switch was opened. Only while there is no input to the contacts is there an output.

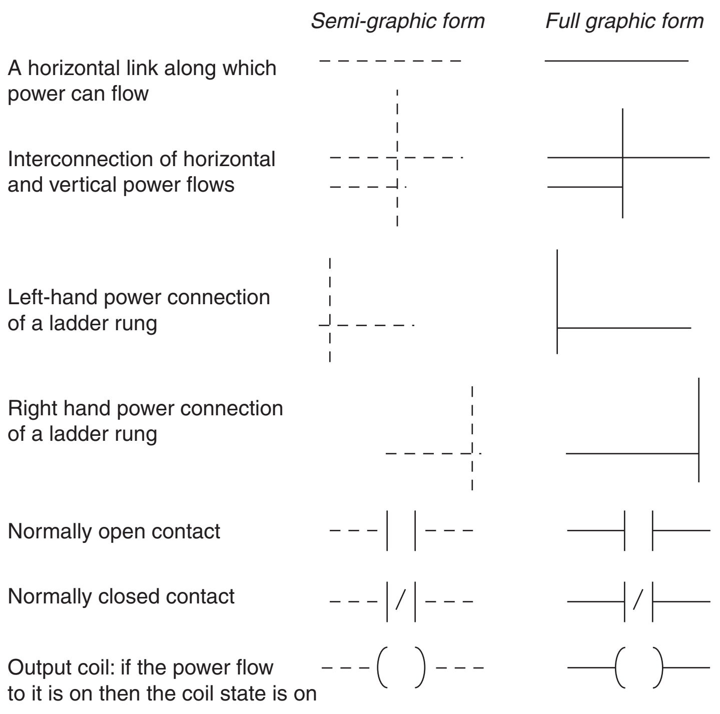

Semi-graphic form
Full graphic form
A horizontal link along which power can flow
Interconnection of horizontal and vertical power flows
Left-hand power connection of a ladder rung
Right hand power connection of a ladder rung
Normally open contact
Normally closed contact
Output coil: if the power flow to it is on then the coil state is on

Figure 11.4: Basic symbols

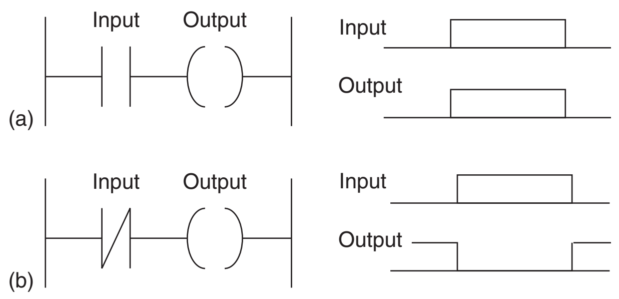


Figure 11.5: A ladder rung

In drawing ladder diagrams the names of the associated variable or addresses of each element are appended to its symbol. Thus Figure 11.6 shows how the ladder diagram of Figure 11.5a would appear using (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique notations for the addresses. Thus, Figure 11.6a indicates that this rung of the ladder program has an input from address X400 and an output to address Y430. When wiring up the inputs and outputs to the PLC, the relevant ones must be connected to the input and output terminals with these addresses.

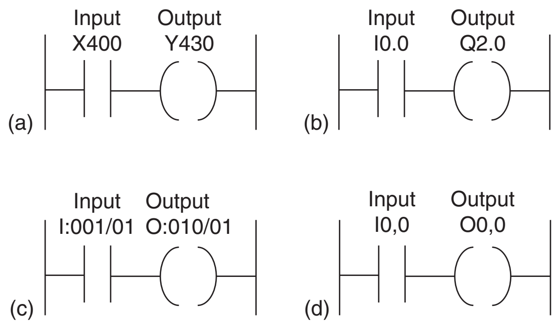


Figure 11.6: Notation: (a) Mitsubishi (b) Siemens (c) Allen-Bradley (d) Telemecanique

11.2 Logic Functions

There are many control situations requiring actions to be initiated when a certain combination of conditions is realized. Thus, for an automatic drilling machine, there might be the condition that the drill motor is to be activated when the limit switches are activated that indicate the presence of the workpiece and the drill position as being at the surface of the workpiece. Such a situation involves the AND logic function, condition A and condition B having both to be realized for an output to occur. This section is a consideration of such logic functions.

11.2.1 AND

Figure 11.7a shows a situation where an output is not energized unless two, normally open, switches are both closed. Switch A *and* switch B have both to be closed, which thus gives an AND logic situation. We can think of this as representing a control system with two inputs A and B (Figure 11.7b). Only when A and B are both on is there an output. Thus if we use 1 to indicate an on signal and 0 to represent an off signal, then for there to be a 1 output we must have A *and* B both 1. Such an operation is said to be controlled by a logic gate and the relationship between the inputs to a *logic gate* and the outputs is tabulated in a form known as a truth table. Thus for the AND gate we have:

Inputs		Output
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

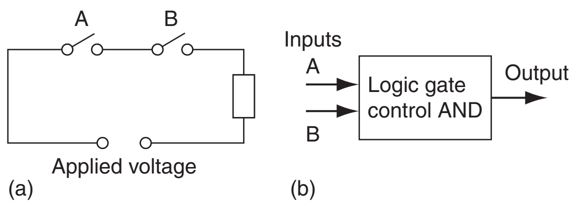


Figure 11.7: (a) AND circuit (b) AND logic gate

An example of an AND gate is an interlock control system for a machine tool so that it can only be operated when the safety guard is in position and the power switched on.

Figure 11.8a shows an AND gate system on a ladder diagram. The ladder diagram starts with | |, a normally open set of contacts labeled input A, to represent switch A and in series with it | |, another normally open set of contacts labeled input B, to represent switch B. The line then terminates with O to represent the output. For there to be an output, both input A and input B have to occur, i.e., input A and input B contacts have to be closed (Figure 11.8b). In general:

On a ladder diagram contacts in a horizontal rung, i.e., contacts in series, represent the logical AND operations.

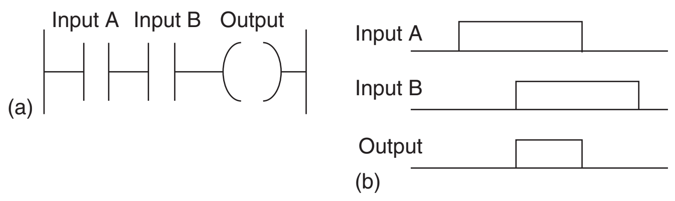


Figure 11.8: AND gate with a ladder diagram rung

11.2.2 OR

Figure 11.9a shows an electrical circuit where an output is energized when switch A *or* B, both normally open, are closed. This describes an OR logic gate (Figure 11.9b) in that input A *or* input B must be on for there to be an output. The truth table is:

Inputs		**Output**
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	1

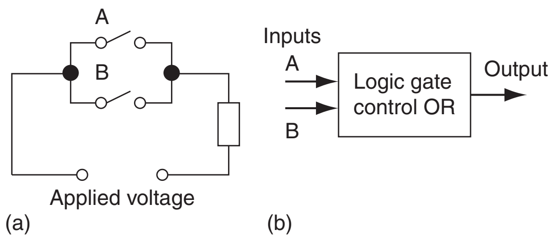


Figure 11.9: (a) OR electrical circuit (b) OR logic gate

Figure 11.10a shows an OR logic gate system on a ladder diagram, Figure 11.10b showing an equivalent alternative way of drawing the same diagram. The ladder diagram starts with | |, normally open contacts labeled input A, to represent switch A and in parallel with it | |, normally open contacts labeled input B, to represent switch B. Either input A *or* input B have to be closed for the output to be energized (Figure 11.10c). The line then terminates with O to represent the output. In general:

Alternative paths provided by vertical paths from the main rung of a ladder diagram, i.e., paths in parallel represent logical OR operations.

An example of an OR gate control system is a conveyor belt transporting bottled products to packaging where a deflector plate is activated to deflect bottles into a reject bin if either the weight is not within certain tolerances or there is no cap on the bottle.

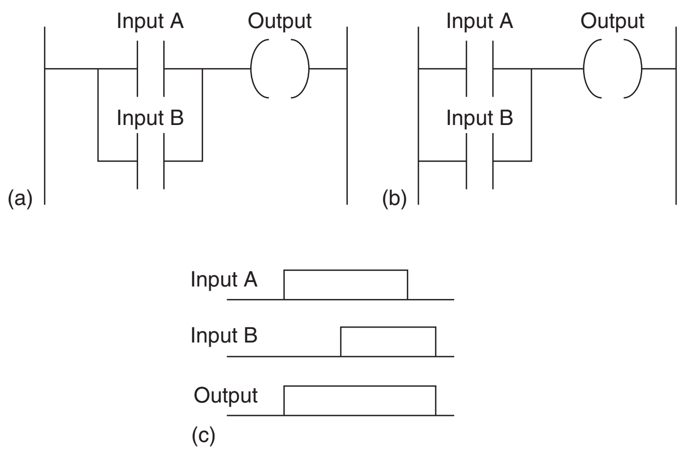


Figure 11.10: OR gate

11.2.3 NOT

Figure 11.11a shows an electrical circuit controlled by a switch that is normally closed. When there is an input to the switch, it opens and there is then no current in the circuit. This illustrates a NOT gate in that there is an output when there is no input and no output when there is an input (Figure 11.11c). The gate is sometimes referred to as an *inverter*. The truth table is:

Input A	**Output**
0	1
1	0

Figure 11.11b shows a NOT gate system on a ladder diagram. The input A contacts are shown as being normally closed. This is in series with the output ( ). With no input to input A, the contacts are closed and so there is an output. When there is an input to input A, it opens and there is then no output.

An example of a NOT gate control system is a light that comes on when it becomes dark, i.e., when there is no light input to the light sensor there is an output.

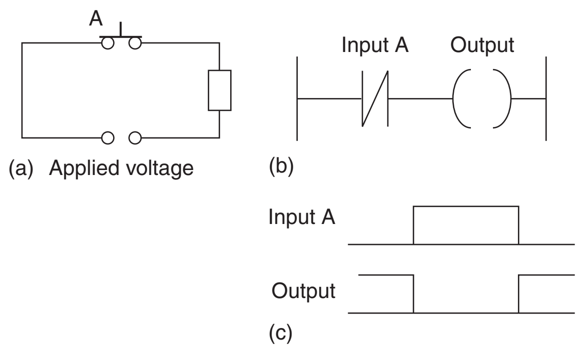


Figure 11.11: (a) NOT circuit (b) NOT logic with a ladder rung (c) high output when no input to A

11.2.4 NAND

Suppose we follow an AND gate with a NOT gate (Figure 11.12a). The consequence of having the NOT gate is to invert all the outputs from the AND gate. An alternative, which gives exactly the same results, is to put a NOT gate on each input and then follow that with OR (Figure 11.12b). The same truth table occurs, namely:

Inputs		**Output**
A	**B**	
0	0	1
0	1	1
1	0	1
1	1	0

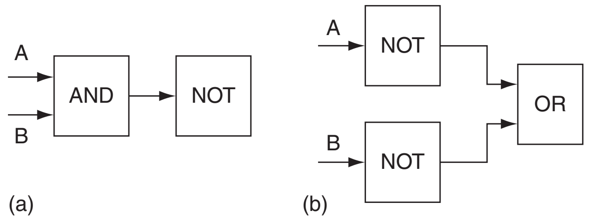


Figure 11.12: NAND gate

Both the inputs A and B have to be 0 for there to be a 1 output. There is an output when input A and input B are not 1. The combination of these gates is termed a NAND gate (Figure 11.13).

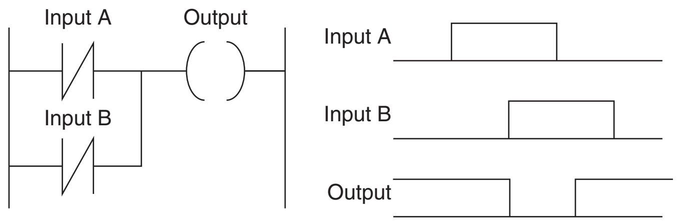


Figure 11.13: A NAND gate

An example of a NAND gate control system is a warning light that comes on if, with a machine tool, the safety guard switch has not been activated and the limit switch signalling the presence of the workpiece has not been activated.

11.2.5 NOR

Suppose we follow an OR gate by a NOT gate (Figure 11.14a). The consequence of having the NOT gate is to invert the outputs of the OR gate. An alternative, which gives exactly the same results, is to put a NOT gate on each input and then an AND gate for the resulting inverted inputs (Figure 11.14b). The following is the resulting truth table:

Inputs		**Output**
A	**B**	
0	0	1
0	1	0
1	0	0
1	1	0

The combination of OR and NOT gates is termed a NOR gate. There is an output when neither input A or input B is 1.

Figure 11.15 shows a ladder diagram of a NOR system. When input A and input B are both not activated, there is a 1 output. When either X400 or X401 are 1 there is a 0 output.

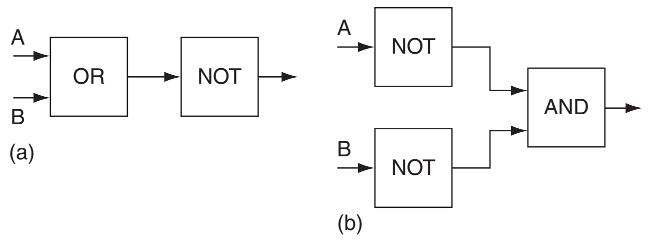


Figure 11.14: NOR gate

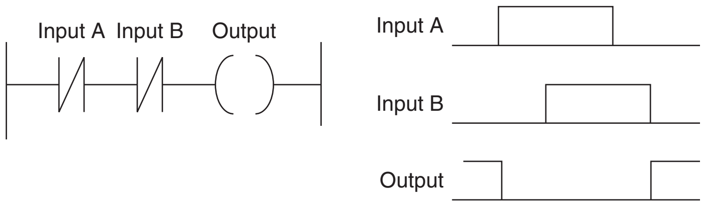


Figure 11.15: NOR gate

11.2.6 Exclusive OR (XOR)

The OR gate gives an output when either or both of the inputs are 1. Sometimes there is, however, a need for a gate that gives an output when either of the inputs is 1 but not when both are 1, i.e., has the truth table:

Inputs		**Output**
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	0

Such a gate is called an *Exclusive* OR or XOR gate. One way of obtaining such a gate is by using NOT, AND and OR gates as shown in Figure 11.16.

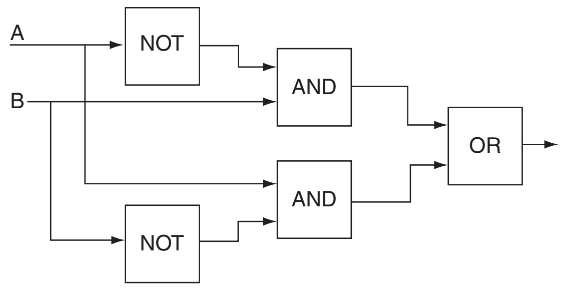


Figure 11.16: XOR gate

Figure 11.17 shows a ladder diagram for an XOR gate system. When input A and input B are not activated then there is 0 output. When just input A is activated, then the upper branch results in the output being 1. When just input B is activated, then the lower branch results in the output being 1. When both input A and input B are activated, there is no output. In this example of a logic gate, input A and input B have two sets of contacts in the circuits, one set being normally open and the other normally closed. With PLC programming, each input may have as many sets of contacts as necessary.

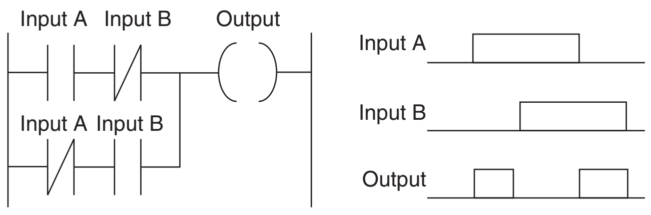


Figure 11.17: XOR gate

11.3 Latching

There are often situations where it is necessary to hold an output energized, even when the input ceases. A simple example of such a situation is a motor, which is started by pressing a push button switch. Though the switch contacts do not remain closed, the motor is required to continue running until a stop push button switch is pressed. The term *latch circuit* is used for the circuit used to carry out such an operation. It is a self-maintaining circuit in that, after being energized, it maintains that state until another input is received.

An example of a latch circuit is shown in Figure 11.18. When the input A contacts close, there is an output. However, when there is an output, another set of contacts associated

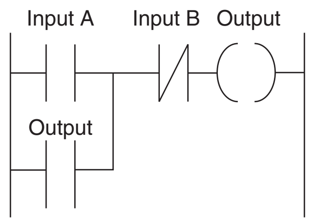


Figure 11.18: Latched circuit

with the output closes. These contacts form an OR logic gate system with the input contacts. Thus, even if the input A opens, the circuit will still maintain the output energized. The only way to release the output is by operating the normally closed contact B.

As an illustration of the application of a latching circuit, consider a motor controlled by stop and start push button switches and for which one signal light must be illuminated when the power is applied to the motor and another when it is not applied. Figure 11.19 shows the ladder diagram with Mitsubishi notation for the addresses.

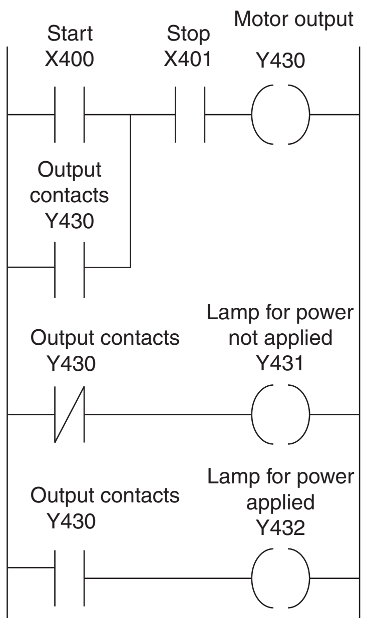


Figure 11.19: Motor on-off, with signal lamps, ladder diagram. Note that the stop contacts X401 are shown as being programmed as open. If the stop switch used is normally closed then X401 receives a start-up signal to close. This gives a safer operation than programming X401 as normally closed.

X401 is closed when the program is started. When X400 is momentarily closed, Y430 is energized and its contacts close. This results in latching and also the switching off of Y431 and the switching on of Y432. To switch the motor off, X401 is pressed and opens. Y430 contacts open in the top rung and third rung, but close in the second rung. Thus Y431 comes on and Y432 off.

Latching is widely used with start-ups so that the initial switch on of an application becomes latched on.

11.4 Multiple Outputs

With ladder diagrams, there can be more than one output connected to a contact. Figure 11.20 shows a ladder program with two output coils. When the input contacts close, both the coils give outputs.

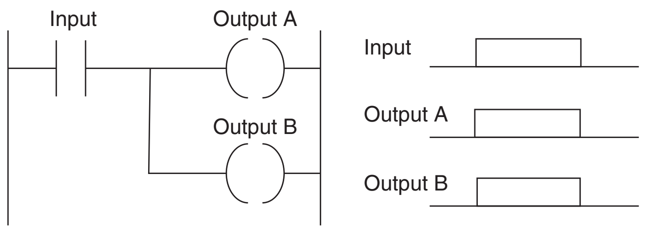


Figure 11.20: Ladder rung with two outputs

For the ladder rung shown in Figure 11.21, output A occurs when input A occurs. Output B only occurs when both input A and input B occur.

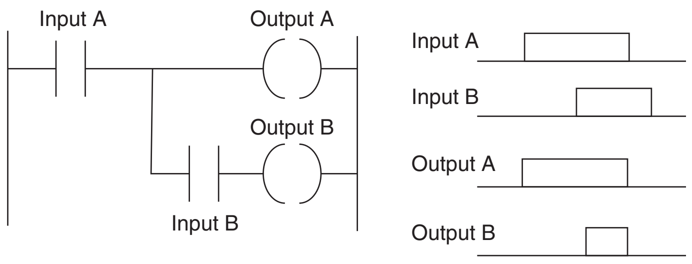


Figure 11.21: Ladder rung with two inputs and two outputs

Such an arrangement enables a sequence of outputs to be produced, the sequence being in the sequence with which contacts are closed. Figure 11.22 illustrates this with the same ladder program in Mitsubishi and Siemens notations. Outputs A, B and C are switched

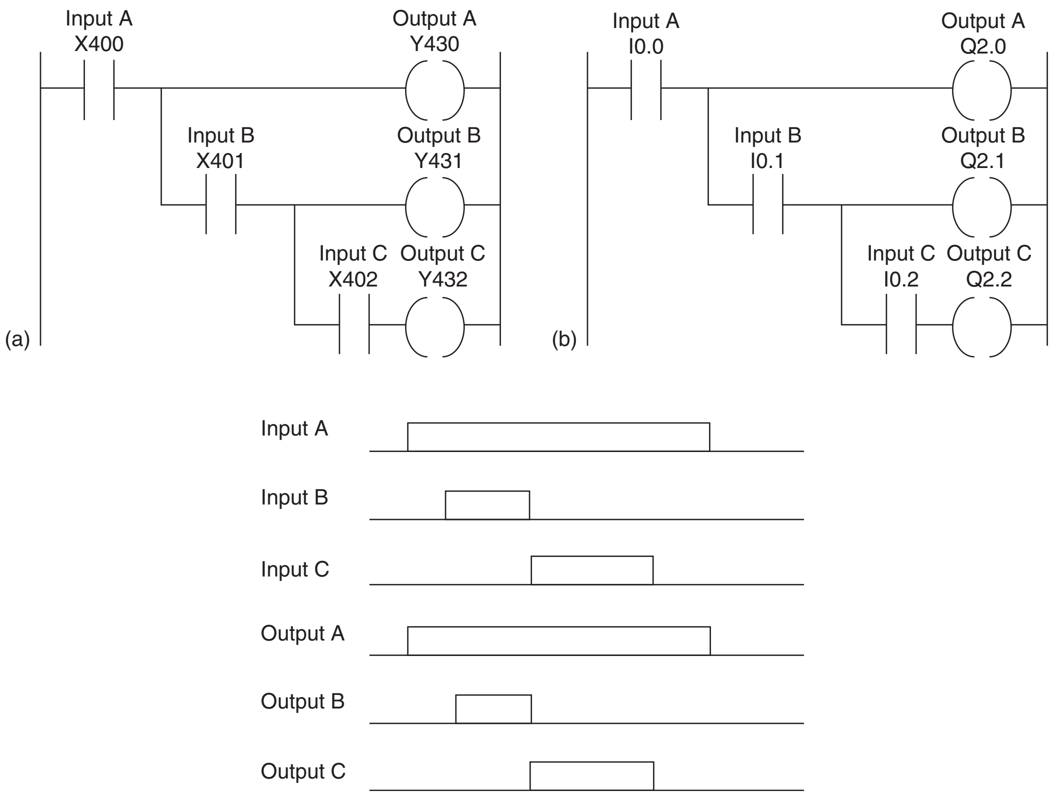


Figure 11.22: Sequenced outputs

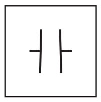
on as the contacts in the sequence given by the contacts A, B and C are being closed. Until input A is closed, none of the other outputs can be switched on. When input A is closed, output A is switched on. Then, when input B is closed, output B is switched on. Finally, when input C is closed, output C is switched on.

11.5 Entering Programs

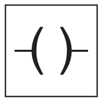
Each horizontal rung on the ladder represents an instruction in the program to be used by the PLC. The entire ladder gives the complete program. There are several methods that can be used for keying in the program into a programming terminal. Whatever method is used to enter the program into a programming terminal or computer, the output to the memory of the PLC has to be in a form that can be handled by the microprocessor. This is termed *machine language* and is just binary code, e.g., 0010100001110001.

11.5.1 Ladder Symbols

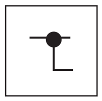
One method of entering the program into the programming terminal involves using a keypad having keys with symbols depicting the various elements of the ladder diagram and keying them in so that the ladder diagram appears on the screen of the programming terminal. For example, to enter a pair of contacts the key marked:



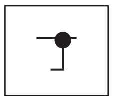
might be used, followed by its address being keyed in. To enter an output the key marked:



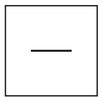
might be used, followed by its address. To indicate the start of a junction:



might be pressed; to indicate the end of a junction path:



To indicate horizontal circuit links, the following key might be used:



The terminal then translates the program drawn on the screen into machine language.

Computers can be used to draw up a ladder program. These involve loading the computer with the relevant software, e.g., RSLogix from Rockwell Automation Inc. for Allen-Bradley PLCs, MELSOFT – GX Developer for Mitsubishi PLCs, STEP 7 – Micro/WIN V4 for Siemens PLCs. The software operates on the Windows operating system and involves selecting items, in the usual Windows manner, from pull-down menus on the screen.

11.6 Function Blocks

The term *function block diagram* (FBD) is used for PLC programs described in terms of graphical blocks. It is described as being a graphical language for depicting signal and data flows through blocks, these being reusable software elements. A function block is a program instruction unit which, when executed, yields one or more output values. Thus, a block is represented in the manner shown in Figure 11.23 with the function name written in the box.

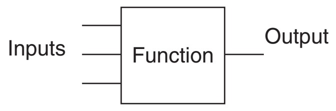


Figure 11.23: Function block

The IEC 113-3 standard for drawing such blocks is shown in Figure 11.24. A function block is depicted as a rectangular block with inputs entering from the left and outputs emerging from the fight. The function block type name is shown in the block, e.g., AND, with the name of the function block in the system shown above it, Timer1. Names of function block inputs are shown within the block at the appropriate input and output points. Cross diagram connectors are used to indicate where graphical lines would be difficult to draw without cluttering up or complicating a diagram and show where an output at one point is used as an input at another.

Function blocks can have standard functions, such as those of the logic gates or counters or times, or have functions defined by the user, e.g., a block to obtain an average value of inputs.

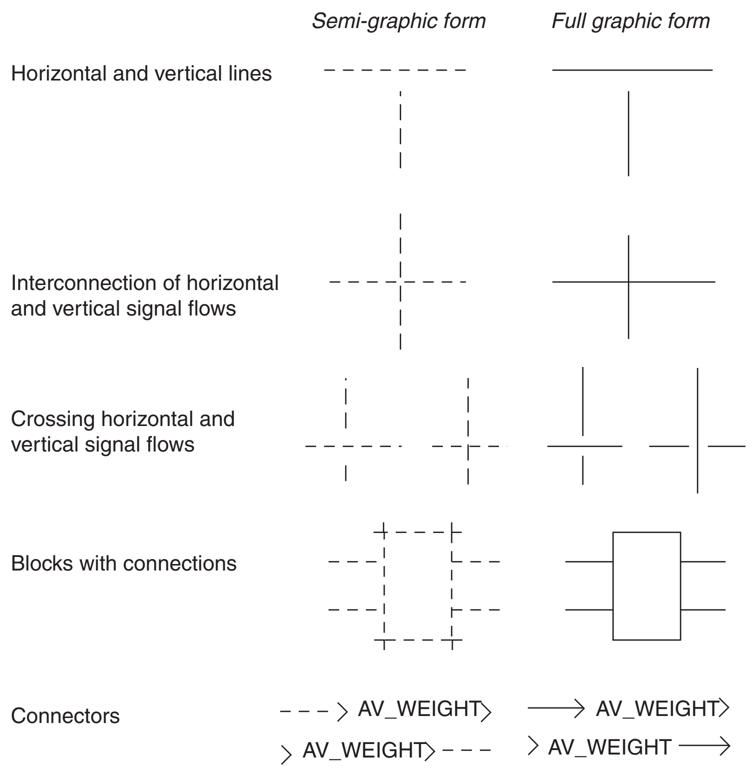

Semi-graphic form
Full graphic form
Horizontal and vertical lines
Interconnection of horizontal and vertical signal flows
Crossing horizontal and vertical signal flows
Blocks with connections
Connectors
AV_WEIGHT
AV_WEIGHT
AV_WEIGHT
AV_WEIGHT

Figure 11.24: Function block diagrams representation

11.6.1 Logic Gates

Programs are often concerned with logic gates. Two forms of standard circuit symbols are used for logic gates, one having originated in the United States and the other being an international standard form (IEEE/ANSI) which uses a rectangle with the logic function written inside it. The 1 in a box indicates that there is an output when the input is 1. The OR function is given by ≥ 1; this is because there is an output if an input is greater than or equal to 1. A negated input is represented by a small circle on the input, and a negative output by a small circle on the output (Figure 11.25). Figure 11.26 shows the symbols. In FBD diagrams the notation used in the IEEE/ANSI form is often encountered.

Figure 11.27 shows the effect of such functional blocks in PLC programs.

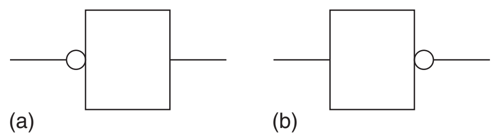


Figure 11.25: (a) Negated input (b) negated output

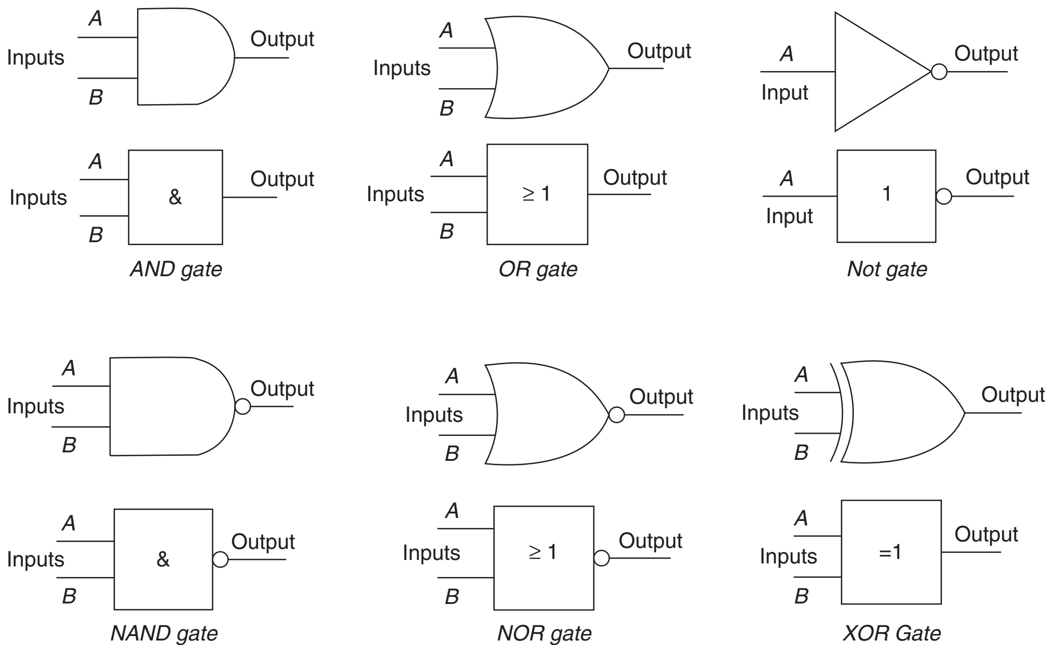


Figure 11.26: Logic gate symbols

To illustrate the form of such a diagram and its relationship to the ladder diagram, Figure 11.28 shows an OR gate. When A or B inputs are 1 then there is an output.

Figure 11.29 shows a ladder diagram and its function block equivalent in Siemens notation. The = block is used to indicate an output from the system.

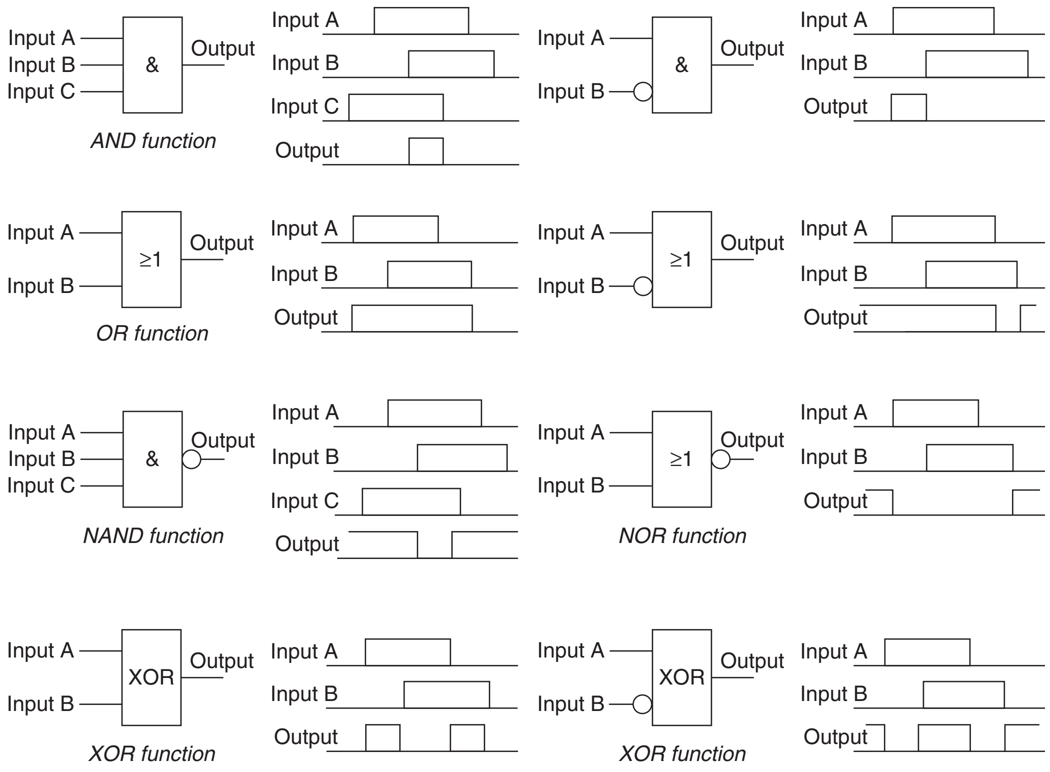


Figure 11.27: Functional blocks

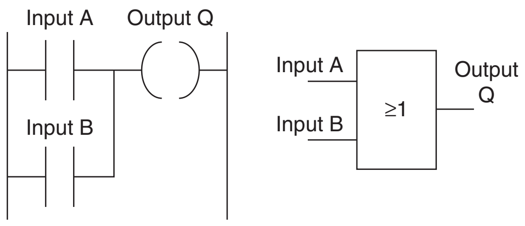


Figure 11.28: Ladder diagram and equivalent functional block diagram

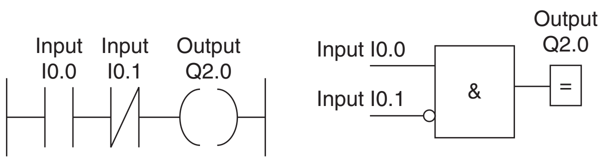


Figure 11.29: Ladder diagram and equivalent function block diagram

Figure 11.30 shows a ladder diagram involving the output having contacts acting as an input. The function block diagram equivalent can be shown as a feedback loop.

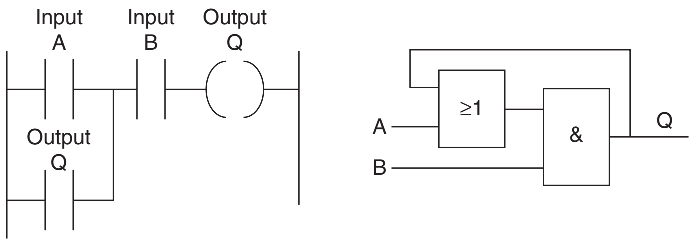


Figure 11.30: Ladder diagram and equivalent function block diagram

Consider the development of a function block diagram and ladder diagram for an application in which a pump is required to be activated and pump liquid into a tank when the start switch is closed, the level of liquid in the tank is below the required level and there is liquid in the reservoir from which it is to be pumped. What is required is an AND logic situation between the start switch input and a sensor input which is on when the liquid in the tank is below the required level. We might have a switch that is on until the liquid is at the required level. These two elements are then in an AND logic situation with a switch indicating that there is liquid in the reservoir. Suppose this switch gives an input when there is liquid. The function block diagram, and the equivalent ladder diagram, is then of the form shown in Figure 11.31.

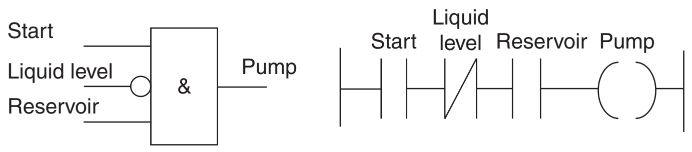


Figure 11.31: Pump application

11.6.2 Boolean Algebra

Ladder programs can be derived from Boolean expressions since we are concerned with a mathematical system of logic. In Boolean algebra there are just two digits, 0 and 1. When we have an AND operation for inputs A and B then we can write:

$$A \cdot B = Q$$

where Q is the output. Thus Q is equal to 1 only when A = 1 and B = 1. The OR operation for inputs A and B is written as:

$$A + B = Q$$

Thus Q is equal to 1 only when A = 1 or B = 1. The NOT operation for an input A is written as:

$$\overline{A} = Q$$

Thus when A is not 1 there is an output.

As an illustration of how we can relate Boolean expressions with ladder diagrams, consider the expression:

$$A + B \cdot C = Q$$

This tells us that we have A or the term B and C giving the output Q. Figure 11.32 shows the ladder and functional block diagrams. Written in terms of Mitsubishi notation, the above expression might be:

$$X400 + X401 \cdot X402 = Y430$$

In Siemens notation it might be:

$$I0.0 + I0.1 \cdot I0.2 = Q2.0$$

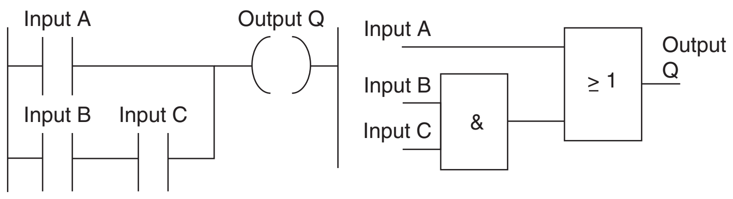


Figure 11.32: Ladder diagram

As a further illustration, consider the Boolean expression:

$$A + \overline{B} = Q$$

Figure 11.33 shows the ladder and functional block diagrams.

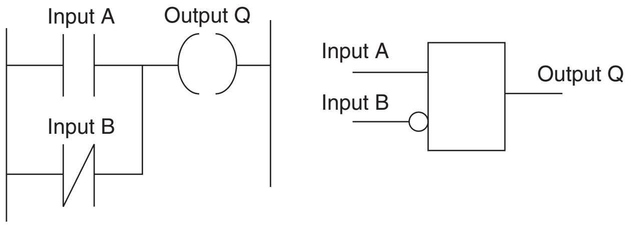


Figure 11.33: Ladder diagram

Written in terms of Mitsubishi notation, the expression might be:

$$\text{X400} + \overline{\text{X401}} = \text{Y430}$$

and in Siemens notation:

$$\text{I0.0} + \overline{\text{I0.1}} = \text{Q2.0}$$

Consider the exclusive-OR gate and its assembly from NOT, AND and OR gates, as shown in Figure 11.34.

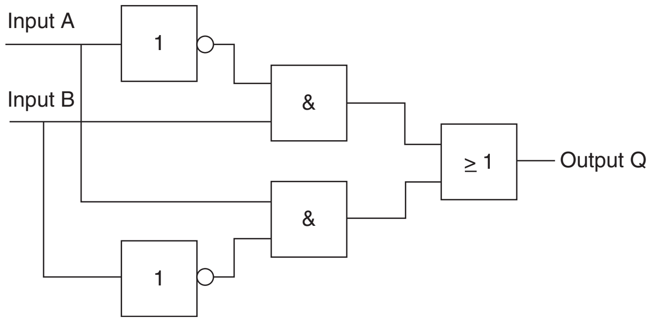


Figure 11.34: XOR gate

The input to the bottom AND gate is:

$$\text{A and } \overline{\text{B}}$$

and so its output is:

$$\text{A} \cdot \overline{\text{B}}$$

The input to the top AND gate is:

$$\overline{A} \text{ and } B$$

and so its output is:

$$\overline{A}{\cdot}B$$

Thus the Boolean expression for the output from the OR gate is:

$$A \cdot \overline{B} + \overline{A} \cdot B = Q$$

Consider a logic diagram with many inputs, as shown in Figure 11.35, and its representation by a Boolean expression and a ladder rung.

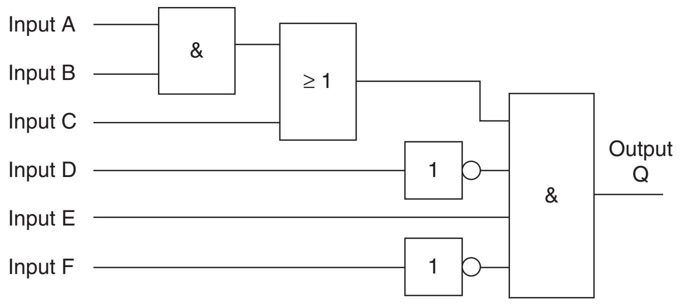


Figure 11.35: Logic diagram

For inputs A and B we obtain an output from the upper AND gate of A · B. From the OR gate we obtain an output of A · B + C. From the lower AND gate we obtain an output Q of:

$$(A \cdot B + C).\overline{D} \cdot E \cdot \overline{F} = Q$$

The ladder diagram to represent this is shown in Figure 11.36.

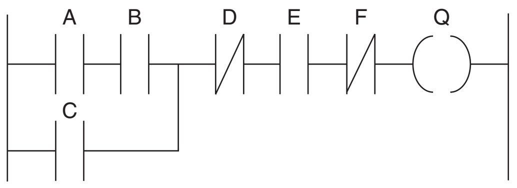


Figure 11.36: Ladder diagram for Figure 11.35

11.7 Program Examples

The following tasks are intended to illustrate the application of the programming techniques given in this chapter.

A signal lamp is required to be switched on if a pump is running and the pressure is satisfactory, or if the lamp test switch is closed. For the inputs from the pump and the pressure sensors we have an AND logic situation since both are required if there is to be an output from the lamp. We, however, have an OR logic situation with the test switch in that it is required to give an output of lamp on regardless of whether there is a signal from the AND system. The function block diagram and the ladder diagram are thus of the form shown in Figure 11.37. Note that with the ladder diagram we tell the PLC when it has reached the end of the program by the use of the END or RET instruction,

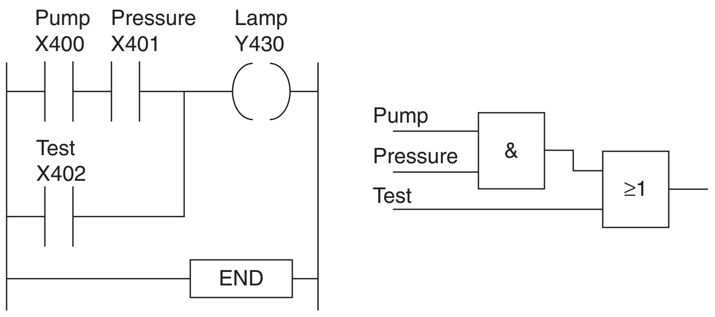


Figure 11.37: Signal lamp task

As another example, consider a valve which is to be operated to lift a load when a pump is running and either the lift switch is operated or a switch operated indicating that the load has not already been lifted and is at the bottom of its lilt channel. We have an OR situation for the two switches and an AND situation involving the two switches and the pump. Figure 11.38 shows a possible program.

As another example, consider a system where there has to be no output when any one of four sensors gives an output, otherwise there is to be an output. One way we could write a program for this is for each sensor to have contacts that are normally closed so there is an output. When there is an input to the sensor the contacts open and the output stops. We have an AND logic situation. Figure 11.39 shows the functional block diagram and the ladder diagram of a system that might be used.

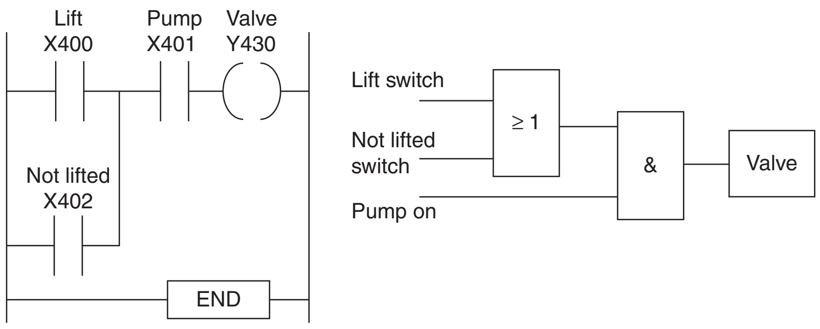


Figure 11.38: Valve operation program

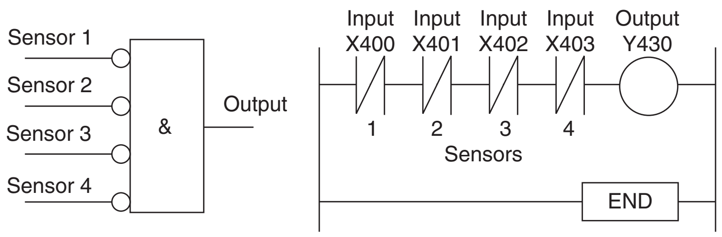


Figure 11.39: Output switched off by any one of four sensors being activated

11.7.1 Location of Stop Switches

The location of stop switches with many applications has to be very carefully considered in order to ensure a safe system. A stop switch is *not* safe if it is normally closed and has to be opened to give the stop action. If the switch malfunctions and remains closed then the system cannot be stopped. Figure 11.40a illustrates this.
A better arrangement is to program the stop switch in the ladder program as open in Figure 11.33b and use a stop switch that is normally closed and operating opens it. Thus there is an input signal to the system which closes the contacts in the program when it starts up.

Figure 11.41 shows where we can safely locate an emergency stop switch. If it is in the input to the PLC (Figure 11.41a) then if the PLC malfunctions it may not be possible to stop the motor. However, if the emergency stop switch is in the output, operating it will stop the motor and also cause the start switch to become unlatched if the arrangement shown in Figure 11.41b is being used. The motor will thus not restart when the emergency stop button is released.

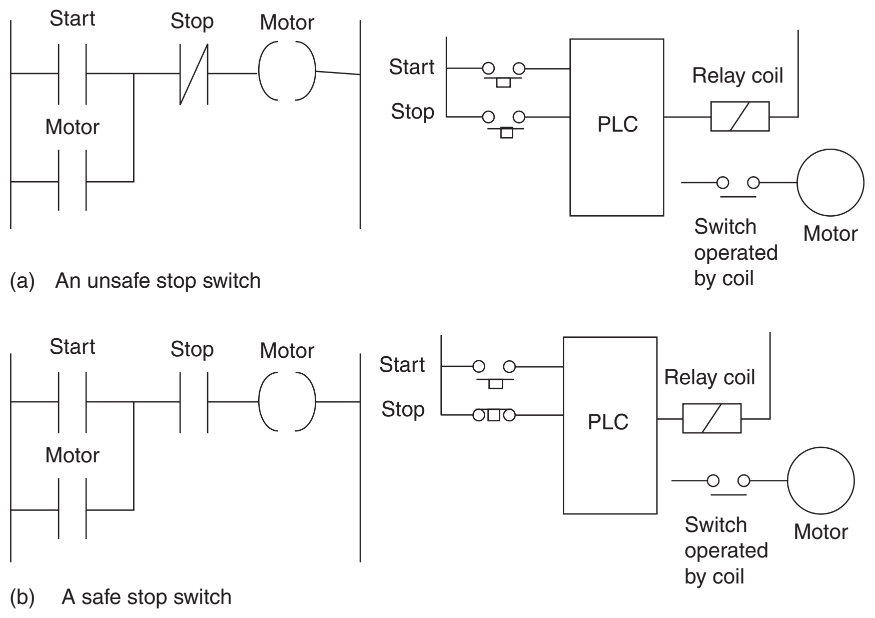


Figure 11.40: Motor stop switch location

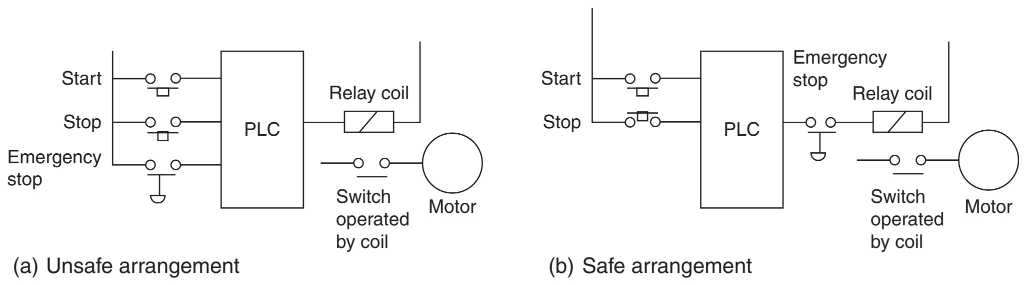


Figure 11.41: Location of emergency stop switch