

# PROGRAMMABLE LOGIC CONTROLLERS

## NQF LEVEL 3

### OUTCOME 3 PROGRAMMING METHODS

This is set at the British Edexcel National level NQF 3

On completion of this tutorial you should be able to use programming techniques to produce a program for a modern programmable controller. You will:

- Learn the programming methods of ladder and logic diagrams, statement listing, functional diagrams, graphical programming languages, mimic diagrams, sequential function charts (SFCs)
- Explain how to produce, store and present programs using a human computer interface (HCI)
- Explain the use within the program of, bit, branch, timer/counter, comparison, logical, arithmetic instructions

#### Contents

1. *Introduction*
2. *Human Computer Interface*  
*Using a Computer*  
*Programming Panels*
3. *Monitoring and Controlling*  
*Mimic Diagrams*
4. *Programming Methods*  
*Tags, Labels and Identifiers*  
*Flow Charts*  
*Function Diagrams*  
*Sequential Function Charts (SFC) or Grafset*  
*Ladder Logic Diagrams*  
*Latches*  
*Logic Functions*  
*Flip Flops*  
*Timers and Counters*  
*Timing Diagrams*  
*Instruction Sets*  
*Truth Tables*  
*Boolean Algebra*
6. *Testing and Debugging*
7. *Installation and Commissioning*
8. *Safety Notes*

## 1. Introduction

There are many ways to programme a PLC and the methods are quite varied. Typically the programme is created on a computer and transferred to the PLC. Usually the programme is tested with a simulation on the computer or on a PLC itself. Once the PLC is controlling a given process, it may be necessary to obtain or put in data at various locations and this is done with some kind of human computer interface (HCI) and this is a good point to start this tutorial.

## 2. Human Computer Interface

In order to programme and interact with the PLC we must have some kind of interface with the human programmer. This is often called Human Computer Interface (or Interaction) and abbreviated to HCI. These should enable the programmer to examine the programme stored in the PLC or to download a new programme. Other functions might be monitoring, diagnostics, testing and data gathering.

With the rapid advances in computer technology it is likely that modern PLC equipment will be supplied with software and interfaces to enable all this to be done with a laptop computer. Older installations may have a variety of dedicated programming hardware with different capabilities such as limited memory and storage. The following describes the types of interface that you may come across. You may see some words and terms that you don't understand yet.

### *Using a Computer*

The entire programme may be constructed on a computer and then downloaded to the PLC. An existing programme may be retrieved from the plc to the computer and changed. The computer may monitor the PLC when working and mimic the actual controlled processes on the screen.



The connection between the two may be through a cable, a wireless link or a network cable. A laptop is a useful tool as it can be taken to the PLC and plugged in to perform programming, diagnostics and data gathering.

The software used by the computer will be available from the PLC manufacturer and will vary from basic level to advanced level. The best software will make the programming easier for the operator to understand and use (i.e. a good HCI).

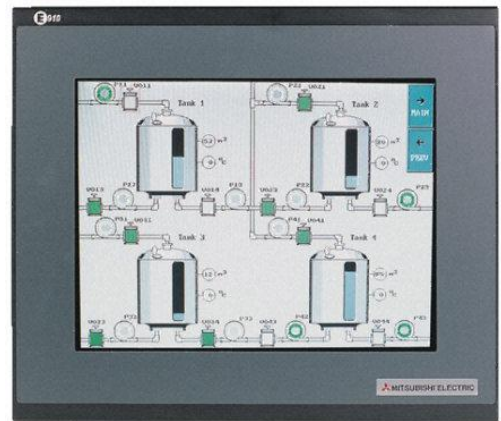
## Programming Panels

Programming panels are plugged into the PLC and will allow the programme to be downloaded or uploaded. Clearly this will be more limited in scope and not so easy for the operator as a well designed computer software programme. Basic panels only have keyboard entry to enter instructions sets. Advanced panels have screens to present the programme in various graphical or text formats and may have keyboard entry or touch screen. It should be possible to download programmes from a computer to the panel and then take the panel to the site of the PLC installation and download it to the PLC.



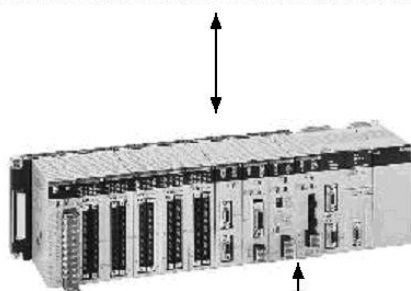
### 3. Monitoring and Controlling

Here is a top of the range Mitsubishi panel (cost about £2000) designed for the highest level of human machine interface (HMI). It is capable of displaying an entire graphic of the process being controlled (mimic) and monitoring all the pressures, temperatures, flow rates and so on. It is designed to work with a wide range of PLCs other than Mitsubishi. A very desirable safety feature is a password to stop anyone from interfering with the process. This panel would enable an authorised person to change settings in the controlled process by touching the screen. The original programming would probably be done on a computer but possibly with the panel.

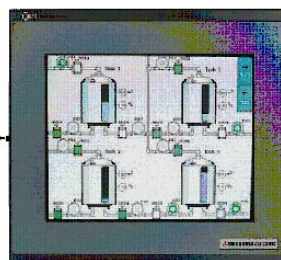


A simplified set up might be like this. It allows supervisors to monitor, adjust or control the process and acquire data about the process. This capability is called Supervisor Control and Data Acquisition (SCADA).

CONNECTIONS TO THE CONTROLLED PROCESS



PROGRAMMING SYSTEM



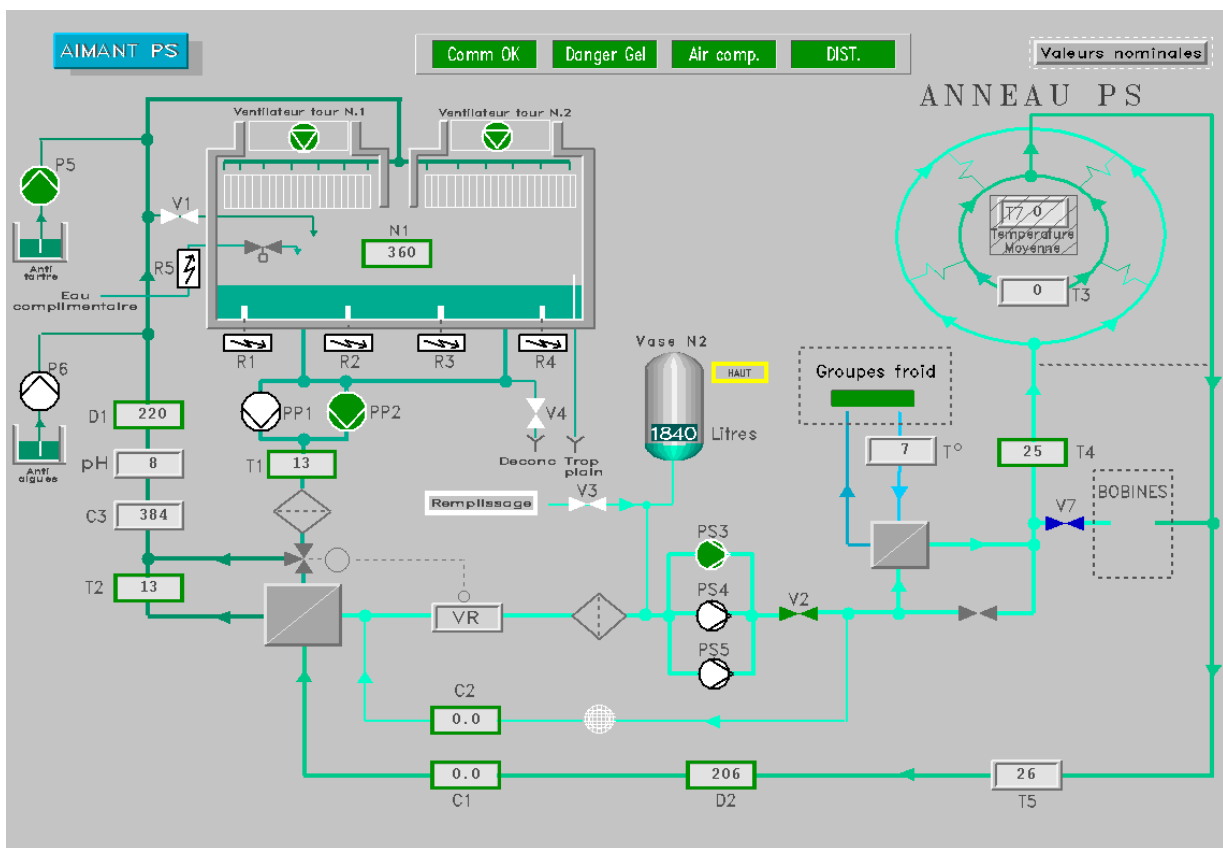
MONITORING AND CONTROL STATION

An arrangement like this will allow a lot of other things to be done like:

- recall records of any given process (e.g. temperature - time data)
- print out the data in text or graphical form for report purposes
- conduct statistical analysis of the stored data and make corrections to the system based on the results
- bring attention to faults and reduction in performance (e.g. set off alarm)
- automatically check instruments and recalibrate or adjust them (smart equipment).
- back up the programme and reinstate it at some point if required
- conduct regular diagnostics on the system (smart equipment)

### Mimic Diagrams

A mimic diagram is simply a graphical representation of the controlled process. This may take the form of a large illuminated panel in a control room as shown right or a monitor screen as shown below. It is a schematic diagram of the system being controlled with all the tags and labels shown. The screen is linked live to the process and mimics the process showing the values of the process variables and status of the switches and valves. For example it will show when a valve is open and a pump is running and what the flow rate and pressure is. Other examples are the use to monitor the status of a water supply network or electricity grid or the status of the railway points in a rail network.

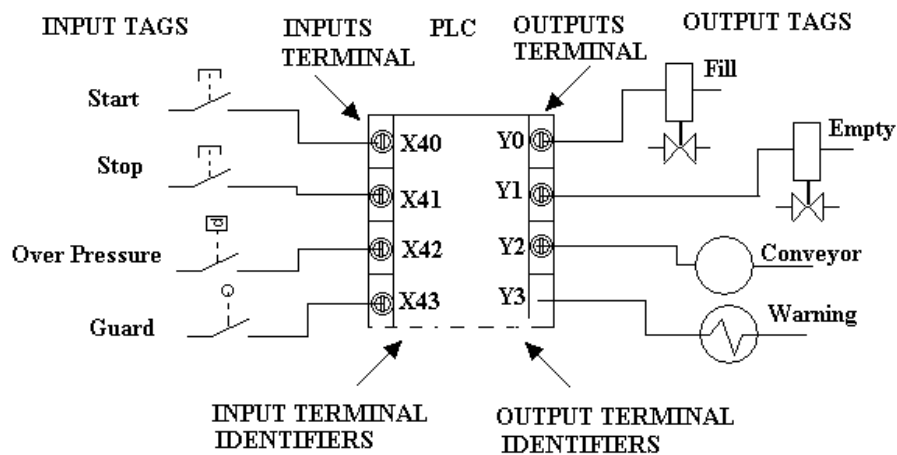


## 4. Programming Methods

Now you have looked at programming hardware, you need to study the actual programming methods. At this level it is impractical to expect a student to learn and use all the programming methods available and your course tutor may decide which one you should give priority to. Under the IEC 61131-3 standard, PLCs can be programmed using five standards-based programming languages.

### Tags, Labels and Identifiers

These are used to make programming simpler by allocating meaningful names to hardware connected to the PLC and internal elements such as counters, timers, registers and relays. Consider a simple PLC with the external connections as shown. There are four input elements connected to the PLC terminals and four output elements.



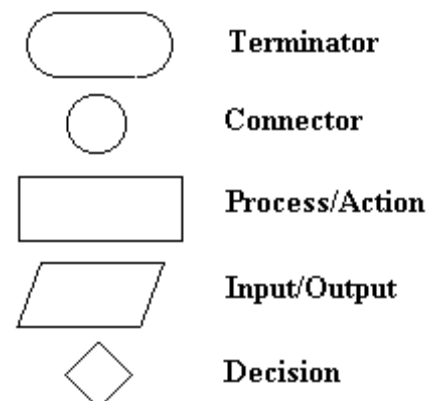
The input and output terminals and other internal relays used inside a PLC are identified by the manufacturer. For example on the Mitsubishi range of PLCs, input terminals are numbered with an X such as X40, X41 and so on. Output terminals start with a Y (e.g.Y40). Other letters are used for internal functions such as T for timers and C for counters. The input and output devices such as switches, sensors motors and actuators and many other items are more easily recognised with labels and tags such as Start, Stop, Guard, Fill, Conveyor, and so on.

When programming the PLC, you have to set up the labels and tags first so that if you allocate Start to terminal X40, then whenever you enter Start, X40 is automatically identified. How this is done depends upon the programming method being used but it makes it easier to programme. Programming software also allows you to add comments with the labels to provide more information about them. The graphics may use these labels on the screen to make it easy to identify the items displayed.

### Flow Charts

Flow diagrams (also called algorithms) are widely used to explain decision making processes that arrive at a logical answer. They are particularly useful for computer programmers because most programmes can be reduced to a series of YES or NO answers to each decision that must be made.

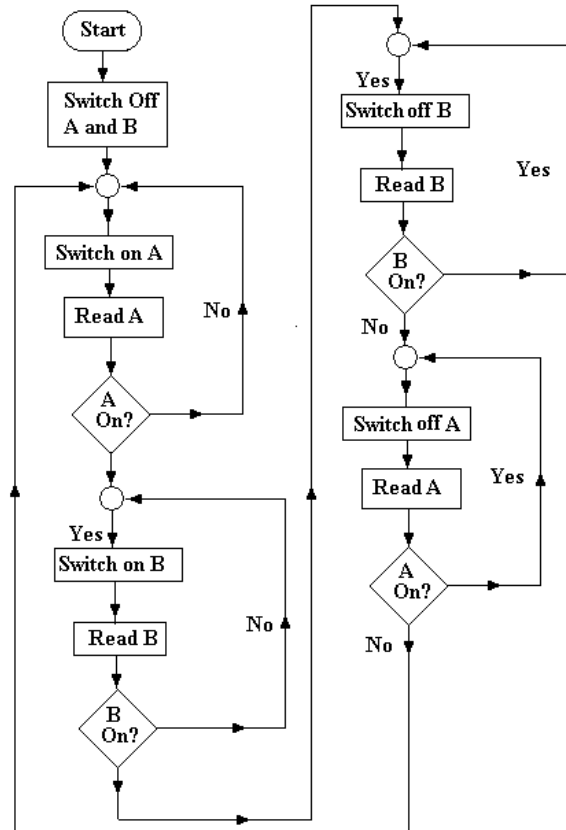
In PLC work they can be a useful tool to help produce a ladder logic diagram (following). The main symbols for flow diagrams are shown here.



## WORKED EXAMPLE No. 1

A machine has 2 actuators A and B that must perform the sequence A+(on) B+ (on) B- (off) A- (off) . Each actuator has a sensors and flip flops so that a single signal indicates when they are on or off. Each actuator is operated by a single on/off signal. Produce a flow chart for the sequence.

### SOLUTION



Step 1 – When we start we must make sure both actuators are off by switching them off.

Step 2 – Start the cycle by switching A on.

Step 3 – Check that A is on and if it is not, then keep looping back until it is.

Step 4 – When A is on switch on B.

Step 4 – Check that B is on and if it is not, then keep looping back until it is.

Step 5 – When B is on, switch B off.

Step 6 – Check if B is off and if it isn't keep looping back until it is.

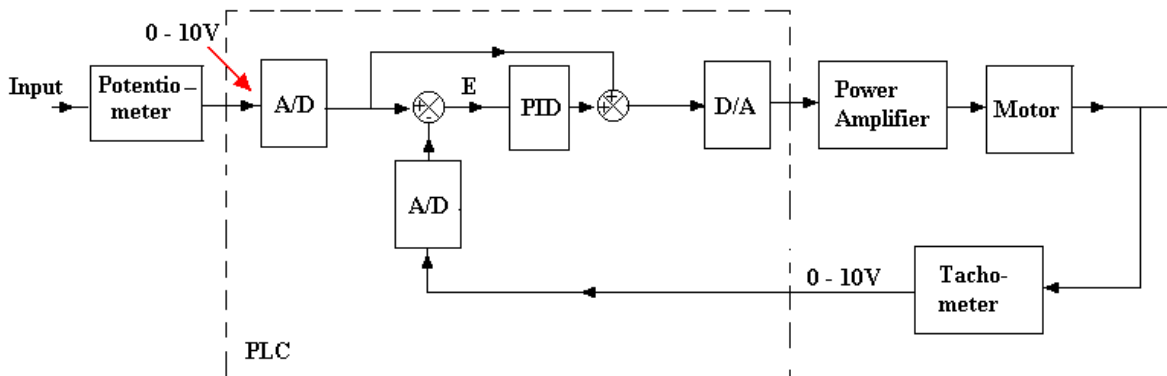
Step 7 – When B is off, switch A off.

Step 8 – Check if A is off and if it isn't keep looping back until it is.

Step 9 – Loop back to the start.

## Function Diagrams

Function block diagram is one of five languages for PLCs supported by [standard IEC 61131-3](#). You have probably seen many of these without knowing that they are called functional diagrams. Most people call them block diagrams. They are widely used to explain how systems and sub-systems connect to each other. Here is a function diagram showing how a PLC may be used to control the speed of an electric motor.



Note that a PLC may contain analogue - digital converters and PID Functions.

A/D - Analogue to Digital converter      D/A - Digital to Analogue converter.

PID - Proportional, Integral and Digital control function (3 term control)

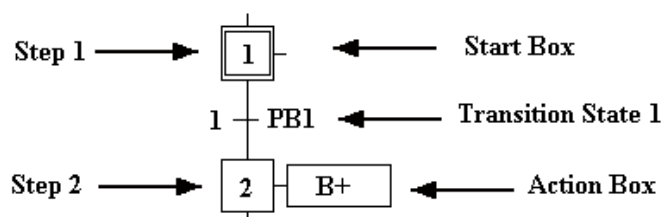
You may or may not know what the above means. That is the problem with this unit; you need to know a lot of other things about control systems. It is way beyond the scope of this unit to explain three term control but you will find it elsewhere on the website [www.freestudy.co.uk](http://www.freestudy.co.uk). Basically it is a form of signal processing that involves setting three constants in order to obtain optimal performance from the system.

## Sequential Function Charts (SFC) or Grafset

Grafset (Sequential Functional Chart) is a graphical method of programming PLC's that is gaining in popularity. **It is best suited to controlling processes that go through a fixed sequence** (but it can be made more flexible). Typically the circuit is constructed with a computer and the graphic symbols are selected and dragged into place. When the circuit is completed and tested the computer converts the circuit into a digital programme and downloads it to the PLC. Unfortunately PLC manufacturers do not always use the same protocol and it is usual to obtain the software from the PLC manufacturer and it will only work with those companies products. This may be changing and standardisation may be taking place to allow software to work with a range of PLCs. There is an international body that is setting standards so that all PLC will respond in the same way to a programme. A useful website to visit is <http://www.lurpa.ens-cachan.fr/grafset.html>.

Each action is followed by feed back to confirm that the action has happened.

The chart starts with an initial step shown as a double box (start box). This is followed by a transition state and here you must enter the tag (PB1) of the input switches that must be activated in order to proceed. In this case PB1 (Push Button 1) must be pressed before you can proceed to step 2. This is followed by the next step (2) and to this is attached an action box. In the action box you enter the tag name for the output element. In this case it switches on actuator B and the tag is B+.



If you want two things to happen together you attach a second action box. At the end of the programme you may loop back to the beginning to complete the sequence. You may also use logic statements such as AND, OR, NAND and so on at the transition points. The next step is not performed until the logical condition is met. This makes it a versatile system. It is also possible to branch and jump to other routines.

### WORKED EXAMPLE No. 2

Produce a sequential function diagram to control the system shown so that the actuators perform the following sequence.

It starts with everything off (retracted). Closing the push button switch PB1 starts the cycle.

Next B goes on (B+) and then when reaching full stroke it goes off again (B-)

When B is fully retracted (off) it is switched on again.

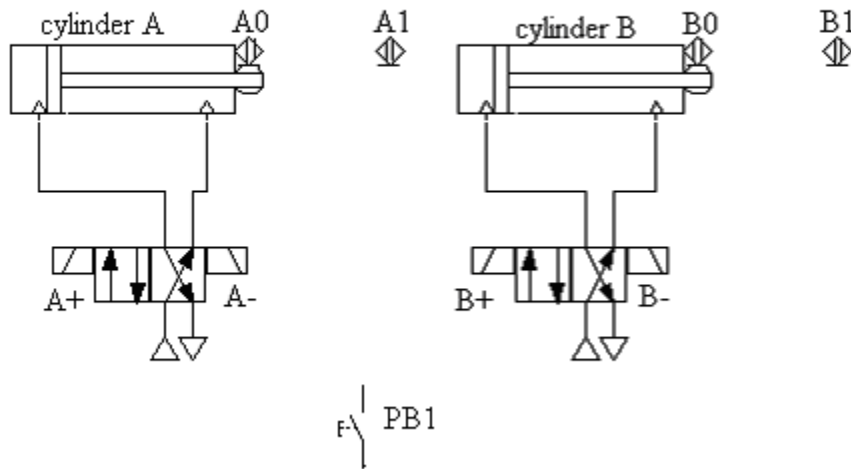
When B reaches its full stroke for the second time, A is switched on.

When A reaches full stroke both cylinders are switched off together and retract together.

The cycle stops until PB1 is pushed again.

B+ B- B+ A+ (A- B-) simultaneously,

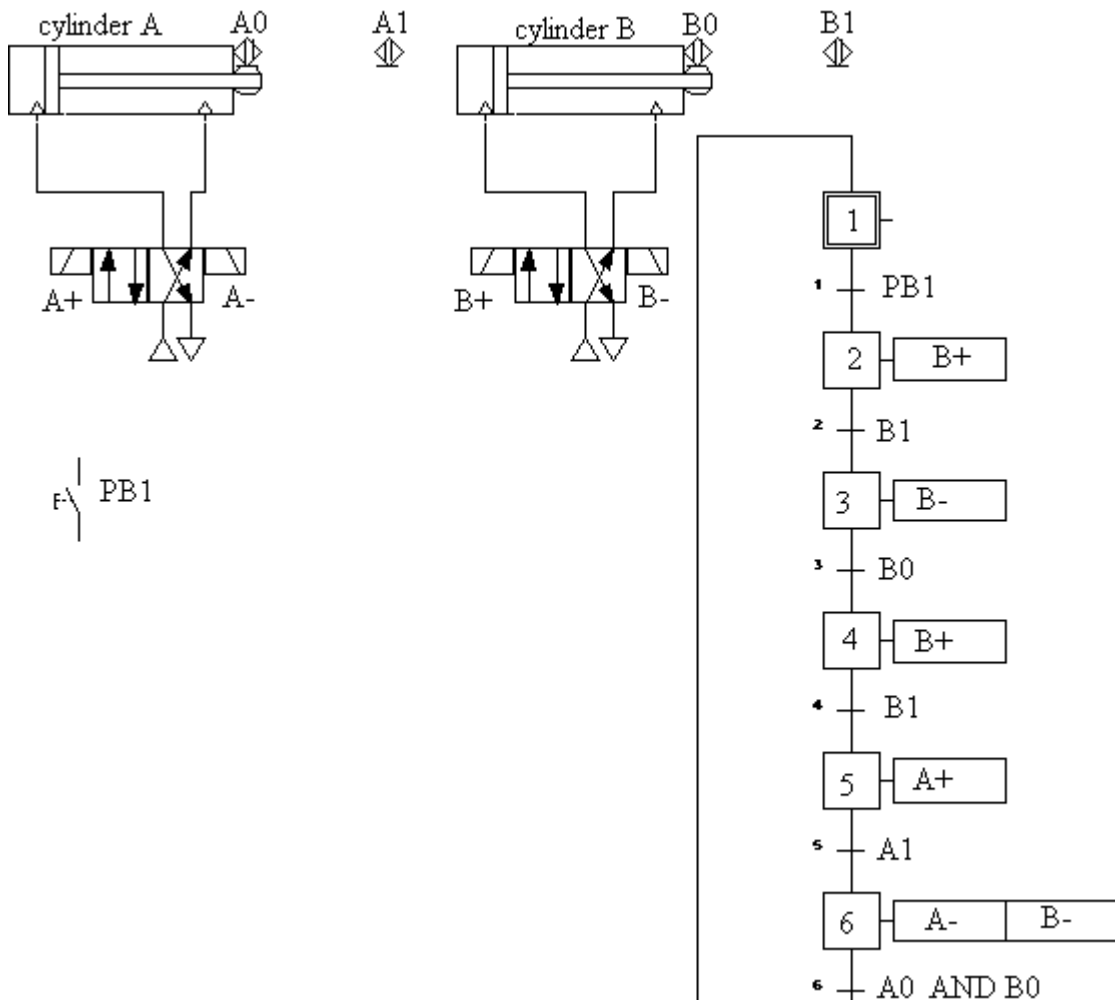
If you have access to PneusimPro simulation software, you should construct the circuit and Grafcet programme using appropriate tags and simulate the result to see if it works.





## SOLUTION

In this kind of circuit it is normal to tag the actuators A, B, C etc. Plus (+) means extend, minus (-) means retract. These tags are allocated to the solenoids that produce the required action. The sensors are located at the two positions of each cylinder and are tagged A0/B0 for the retracted position and A1/B1 for the extended position and so on for each cylinder.

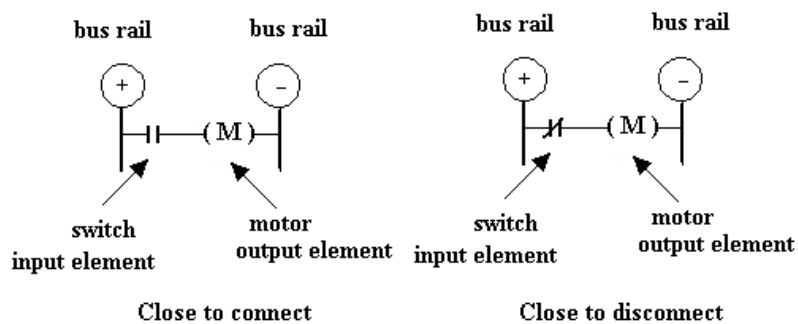


The above is the mimic screen on which the actuators and valves actually move as the programme is executed. The progress of the programme is indicated by colour changes and this helps to sort out problems (debugging).

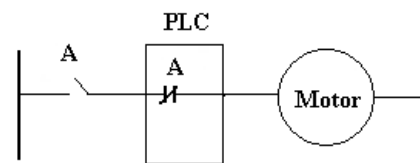
## Ladder Logic Diagrams

This is basically an electric circuit diagram consisting of many parallel circuits all connected between a live rail and a neutral rail. The result is a circuit diagram resembling a ladder and each rung of the ladder is a separate circuit. Each circuit is always active so the system is very flexible. **It is important to realise that the circuit does not operate in sequence but any rung may be activated at any time.** This makes it particularly useful for flexible systems where intervention may occur at any time (e.g. someone presses an emergency stop switch or a magazine becomes emptied or jammed or a variable gets out of limits and sets off an alarm).

The graphical programming method is easiest with drag and drop symbols which may be American or European. (**American symbols are used here**). Each rung of the ladder is an imaginary circuit linking the Plus and Minus bus rails. The rung contains input elements and output elements that must be tagged. When the input elements create a state where the circuit is made, current flows and the output element is activated. Consider the single rung (circuit) below. The left diagram shows a motor (M) connected through an open contact. When the contact is closed the motor is switched on. The right circuit shows the motor connected through a closed contact and will normally be on. Operating the switch will disconnect the motor and turn it off.



It is of great importance to note that the normally closed symbol should be regarded as a NOT gate. The actual switch connected to the PLC may be normally open but the PLC programme reverses it and makes it normally closed. It should be seen like this.

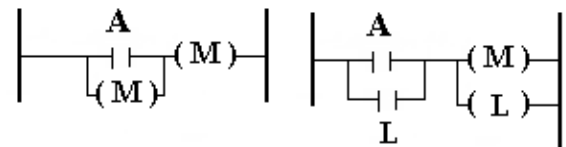


Closing A turns the motor off. It can be very confusing working with real normally closed switches as the not gate converts them into normally open.

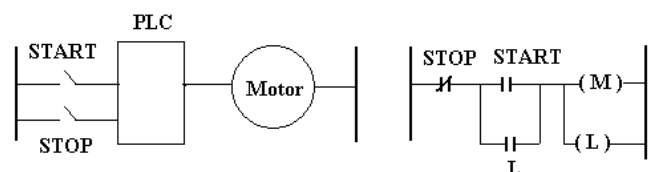
## Latches

On many PLCs, the programming allows you to use an imaginary switch with the same tag as the output. The left circuit below shows that when A is operated the motor comes on and the imaginary contact tagged M will close so that if A is opened again, the motor stays on. Another way to do this is to use another imaginary output (often called internal relay) which carry identifiers such as L for Latch and use them as shown in the right diagram.

The origins of this are in the equipment that used actual relays which when turned on, closed a set of contacts connected across the switch so that the switch became latched.



The next diagram shows how two normally open switches are used to start and stop a motor. When the start switch is operated, the programme turns the motor and the Latch on.



The latch keeps the motor running even when the start switch is released. The stop switch is seen by the programme as a normally closed switch. When operated, the programme breaks the circuit, the motor stops and the latch is broken.

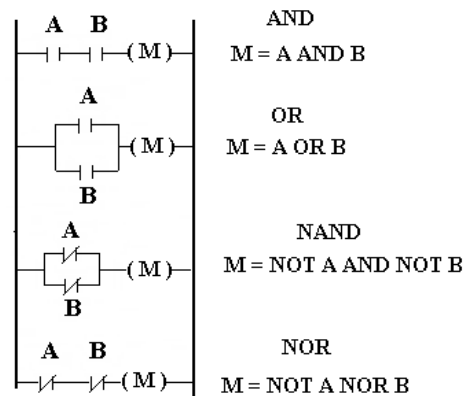
Some PLCs have a function called SET. When an output is set, it stays on no matter what else happens until the command RESET is used. This is simpler than using latches.

### **SELF ASSESSMENT EXERCISE No. 1**

1. Which form of PLC programming would you recommend for the following industrial applications? State your reasons.
  - a) A pneumatic machine following a fixed sequence of operation.
  - b) A robotic packing device that picks up objects when presented at irregular intervals then checks their weight and places them into a box if it is correct and into a rejection bin if incorrect. It also counts them and replaces the full box with an empty box when full.
  - c) A complex process plant system with multiple control functions for pressure, flow, temperature and so on.
2. Explain the importance of a mimic panel at the control centre of a railway network.

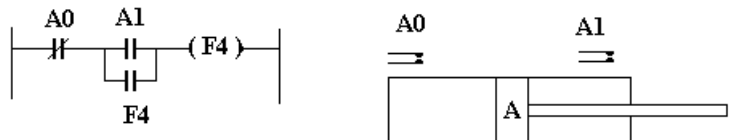
## Logic Functions

The rungs of the ladder can be designed to produce many logical functions such as AND, OR, NAND, NOR and so on.



## Flip Flops

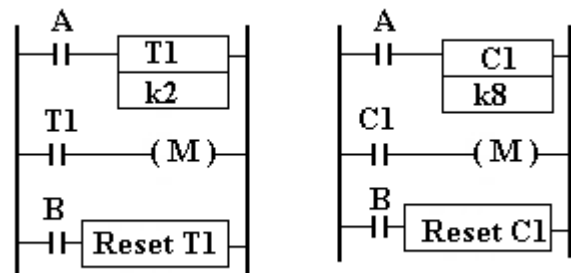
It is possible to make the PLC perform the flip flop function. On the downside, this means that two PLC input terminals are used instead of 1. Consider a cylinder with two proximity detectors A1 and A0 which are both normally open switches. The programme treats A0 as if it were normally closed.



When the cylinder is in between the two positions the contact A1 is open. The circuit is not made and F4 is off. If the cylinder moves out and operates A1, then the circuit is made and F4 is turned on. F4 is latched across A1 so if the piston retracts and A1 becomes disconnected, the circuit is still complete and F4 stays on. When fully retracted, the sensor A0 is activated and breaks the circuit. F4 is then switched off. If the piston moves forward again, F4 remains off until the contact A1 is again closed. In this way the status of the cylinder is indicated by F4. The contact F4 should now be used instead of the contact A in the programme.

## Timers and Counters

The way that timers and counters work varies from one type of PLC to another. The right diagram shows a timer. When switch A is closed, the timer starts running and after 2 seconds (indicated by k2) the timer contact closes and switches on the motor. The right hand diagram shows a counter. The counter is set to 8 (the k8 term). Each time switch A is closed the counter decrements. If the switch is closed and opened repeatedly, after the eighth closure, the counter contact closes and turns on the motor.



There are many variations on this such as counting up rather than down. Once a counter or timer has operated, they need to be reset again before they can be reused. On some PLCs the timer automatically resets when the switch A is opened. The diagrams above show that switch B is used to reset the counter and timer. Reset is an example of a command being used as an output element.

## Timing Diagrams

This is not in any standard but you might find it useful to help work out solutions that involve things going through a cycle entirely controlled by timers (like traffic lights). The example below shows how useful it is.

### WORKED EXAMPLE No. 3

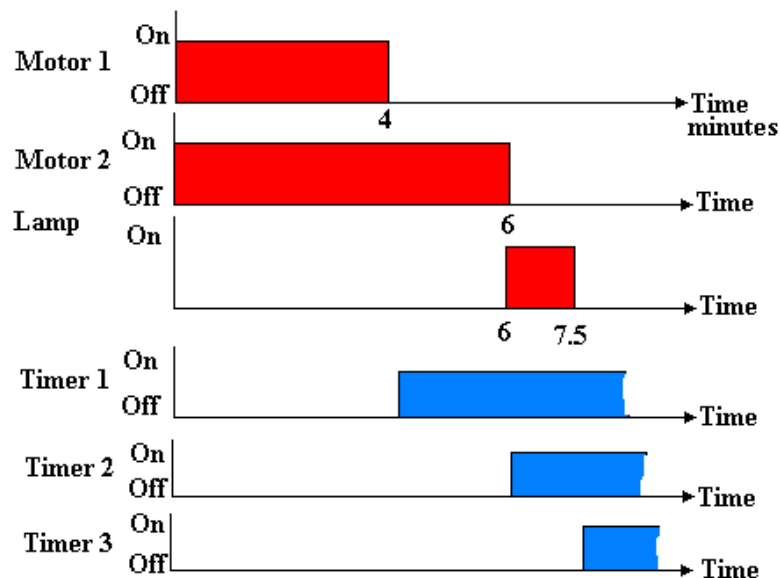
Two motors (outputs M1 and M2) are to be controlled as follows.

- When the run switch is operated both motors must run.
- After 4 minutes motor 1 must stop.
- Motor 2 continues running for another 2 minutes and stops.
- At this point a lamp is switched on.
- After a further 90 seconds, the lamp goes off and the cycle restarts.
- If a stop switch is operated, the system will continue to the end of the cycle and stop.

Produce a PLC programme to make it work.

### SOLUTION

There are many solutions to this problem and this is just one. Timers may run in parallel or series or both. Here is the timing cycle diagram.



In this solution we shall need 3 timers T1, T2 and T3. T1 and T2 are started together. Timer 1 is set to 4 minutes, timer 2 to 6 minutes. Timer 3 is started after 6 minutes and runs for 90 seconds (1.5 minutes).

There are a wide variety of PLC commands for ladder logic. For example, in the Mitsubishi system timers and counters are automatically reset when the logic which starts them running becomes untrue. In other types such as Bytronics - LADSIM programmes, you need to use a reset command to reset timers and counters.

The following is the solution for a Mitsubishi using MEDOC programming. When the run switch is activated all timers are off. If the cycle is turned into ladder logic, this is the result.

**RUNG 1** Timers 1 and 2 are set running by timer 3 being off (i.e. as soon as the run switch is activated).

**RUNG 2** Motor 1 runs as long as timer 1 is off.

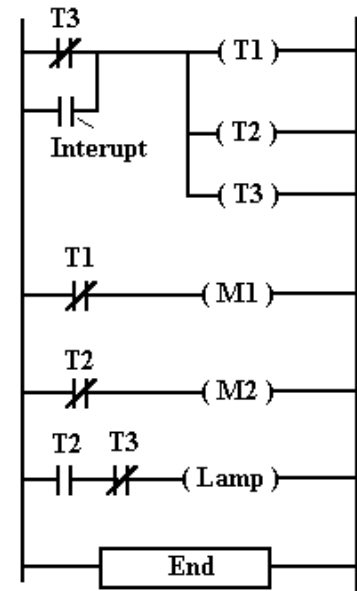
After 4 minutes timer 1 comes on and motor 1 stops.

**RUNG 3** Motor 2 runs as long as timer 2 is off. Timer 2 continues running and after 6 minutes from start it comes on and motor 2 stops.

**RUNG 4** The lamp is on only if timer 2 is on (Motor 2 stopped) and timer 3 is off. Hence the lamp comes on as soon as motor 2 stops and goes off after 90 seconds when timer 3 comes on.

Since timer 3 OFF is in rung 1, timers 1 and 2 are automatically reset and go off causing rung 1 to be reactivated.

**STOP** If the STOP switch is put on, timers 1 and 2 will not reset and the cycle stops at the end.



#### WORKED EXAMPLE No. 4

Components pass along a chute and interrupt a light switch which goes low (off) each time it is interrupted. Every time 6 components have been counted, an eject operation is used to remove the batch and the then it all starts again. Produce a ladder logic diagram to do this operation. The counter is designated C460.

#### SOLUTION

Again, the solution depends upon the type of PLC and programming facilities. This is a solution for a Mitsubishi.

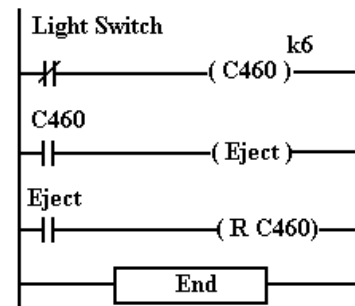
#### EXPLANATION

**RUNG 1** Each time the light switch is interrupted, the counter is decremented by 1. The counter is set up for 6 counts (the k6).

**RUNG 2** When the counter value reaches zero, the counter contact (C460) closes and operates the eject mechanism.

**RUNG 3** The eject output is used as an input contact so that each time the eject command is executed, the counter is reset (R C460).

**RUNG 4** Most PLCs require an END command as shown to stop it running on into any other programme fragments left in the memory.

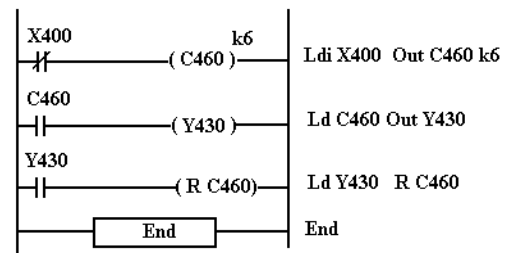


## Instruction Sets

This is a text based system. *It is much more difficult to write a programme this way* but the programme can be created with a simple keyboard connected to the PLC. Any of the programmes so far described can be created using statement lists. These use *mnemonics* to describe the action. The mnemonic is also known as the **OPCODE**. The OPERAND is the data to be executed by the opcode. For example the mnemonic Ld X400 is an instruction to load something (the opcode) and the something is the status of input X400 (the operand). Here are the mnemonics used by Mitsubishi.

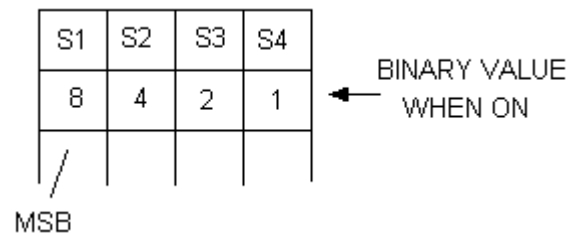
Instruction	Mnemonic	Symbol
Load	Ld	
Load Inverse	Ldi	
And	And	
Inverse And	Ani (Nand)	
Or	Or	
Inverse Or	Ori	
Out	Out	
And Branch	Anb	series branch
Or Branch	Orb	parallel branch
Conditional Jump	CJP	[CJP]
End Jump	EJP	[EJP]
Shift	SFT	[SFT]
Reset register	RST	[RST]
Set	S	[S .....]
Reset	R	[R .....]
Return	Ret	[RET ]

This is the statement list for the last ladder programme. Instructions sets can use labels and tags in just the same way as ladder diagrams and they can be automatically produced from a ladder diagram.



## Truth Tables

A truth table should be considered as a binary number or pattern covering every possible combination of input conditions. Suppose we have four input sensors. We need a table with 4 bits and we will put the most significant bit (MSB) on the left.



The maximum possible value is 15 so there are 16 possible combinations of input conditions. Show each starting with a binary value of 0 and ending with 15. When the truth table has been filled in as shown, you must decide which combinations switch on which outputs.

Suppose there are two outputs A and B and these are only to be switched on as follows.  
A is on when S3 OR S2 AND S1 is on.  
B is on when S4 or S1 is on.

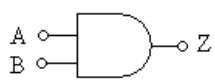
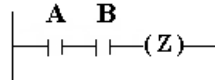
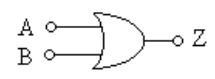
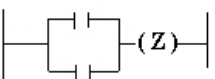
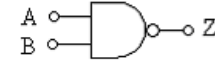
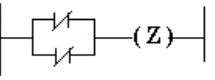
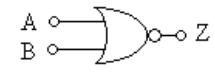
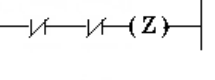
S1	S2	S3	S4	BINARY VALUE
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
1	1	1	1	15

The truth table below only shows the conditions that switch on A and B. If the outputs are being controlled by a computer, it is useful to know the binary value required to switch the outputs on. The number that switches on A is 2, 3 and 12. The number that switches on B are 1, 3 and 8.

S1	S2	S3	S4	A	B
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	0	1	0

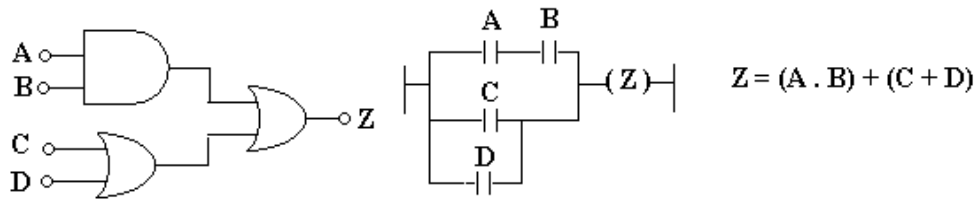
### Boolean Algebra

This is a useful tool for writing out and simplifying logic statements in short hand. A plus sign means OR and a dot means AND. Here are the four main logic gates shown as American Logic symbols along with the ladder logic diagram, truth table and Boolean expression. Remember a circle on the symbol at the input or output is a NOT gate and inverts the action.

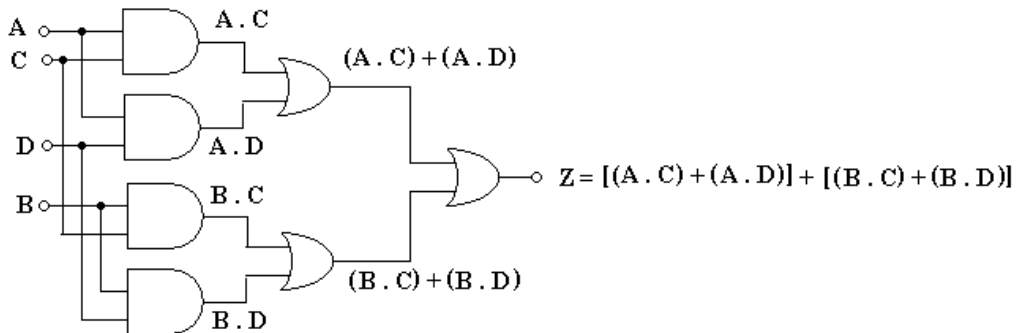
LOGIC	LADDER	TRUTH TABLE	BOOLEAN															
		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Z	0	0	0	0	1	0	1	0	0	1	1	1	$Z = A \cdot B$
A	B	Z																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Z	0	0	0	0	1	1	1	0	1	1	1	1	$Z = A + B$
A	B	Z																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Z	0	0	1	0	1	1	1	0	1	1	1	0	$\bar{Z} = A \cdot B$
A	B	Z																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Z	0	0	1	0	1	0	1	0	0	1	1	0	$\bar{Z} = A + B$
A	B	Z																
0	0	1																
0	1	0																
1	0	0																
1	1	0																



Here is another example.



Boolean algebra can be used to simplify logic circuits. Consider the following circuit.



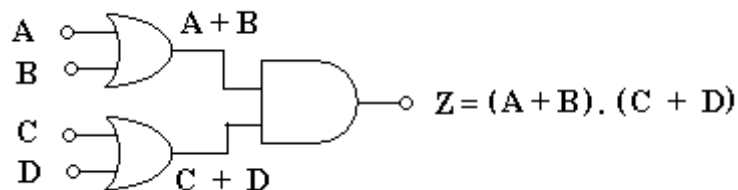
The output may be simplified first by removing the square brackets which makes no difference to the result.

$$Z = (A.C) + (A.D) + (B.C) + (B.D)$$

Next take out common A and B       $Z = A . (C + D) + B . (C + D)$

Next form a new bracket       $Z = (A + B) . (C + D)$

Now redraw the circuit.



### SELF ASSESSMENT EXERCISE No. 2

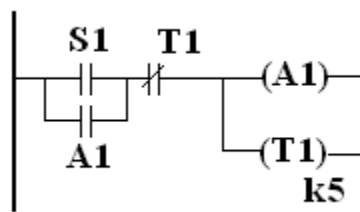
A machine is designed to clamp a component for 5 seconds and then release it. When the start switch (tagged S1) is pressed the actuator (tagged A1) is activated to clamp the component and the timer T1 is started. After the timing is completed the actuator opens the clamp and the timer automatically resets. The start switch is a push button switch that springs back to open when released. The process is started each time by a simple quick press of the switch.

Construct a ladder logic diagram that will control these functions and explain it.

Write out the mnemonic instruction list for the programme using the set described previously.

For simple applications like this, what would be the easiest way of programming the PLC and changing the timer setting occasionally?

The solution is on the next page but don't cheat by looking first.



LADDER DIAGRAM

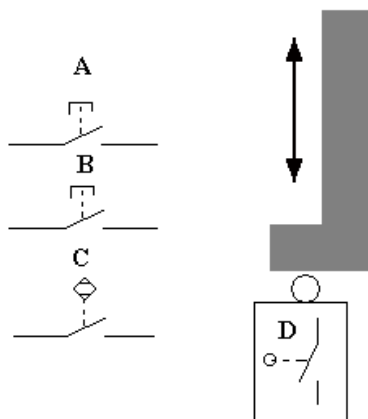
Ld S1 Or A1 Ani T1 Out A1 Out T1 k5

### EXPLANATION

When S1 is pressed momentarily, the circuit is completed and the actuator A1 clamps the component and the timer starts running. A1 latches across S1 so that the actuator remains closed when S1 is released. After the set time has elapsed the timer contact closes. This opens the normally closed contact thus switching off A1. This breaks the latch and resets the timer. This program could easily be done using mnemonics and a simple programming panel but a graphical ladder programmer would be easier. The timer constant k5 can be easily changed with the panel.

### SELF ASSESSMENT EXERCISE No. 3

A machine is switched on by either of two push buttons A or B. There is a safety barrier on the machine with a limit switch D at the closed position. In addition there is proximity switch C to detect if anyone is standing inside the barrier. If there is the machine must not start. All have normally open contacts.



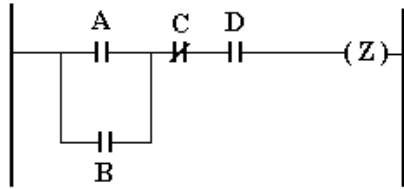
- Write out a suitable Boolean expression.
- Draw a ladder logic diagram for the Boolean expression
- Write out the instruction set.
- Construct the truth Table.

**Note that the switch symbols used above are European and these are more meaningful in this context.**

**The solution is on the next page but don't cheat by looking first.**

## ANSWERS

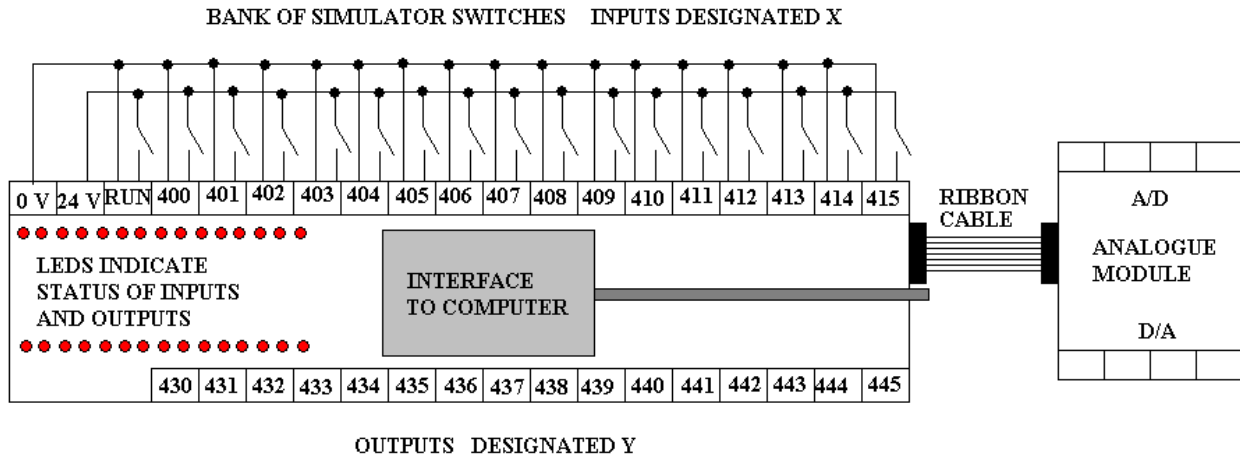
$$Z = (A + B) \cdot \bar{C} \cdot D \quad \text{Ld A Or B Ani C And D Out z}$$



	A	B	C	D	Z
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

## 6. Testing and Debugging

The diagram illustrates a basic system for testing a PLC programme. The Light Emitting Diodes are a useful feature to help monitor and debug the programme. The input terminals have a bank of switches attached to it to enable each to be set high or low. These are normally available as standard for simulation, testing and debugging. The programme is run and the output status observed. The inputs switches are closed and opened to simulate the feedback status and so each step in the programme can be followed to ensure that the appropriate output action occurs as required.



This system of monitoring is all very well for basic installations but more modern programming software should enable the programme to be tested and debugged on screen. For example, the on screen graphics would show where the programme has stopped and the status of all the inputs and outputs.

The main monitoring and debugging tools are in the software used to programme the PLC. If the PLC is connected to the computer with a suitable interface, the programmes may be moved either way between the PC and the PLC. The software may highlight the parts of the programme (e.g. ladder diagram) that are active and display the status of timers, counters and registers.

The most advanced software enables the controlled system to be produced graphically on screen and linked to the program. The system can be simulated and seen working. The simulation may well be the same graphics that will be seen at the operator's station on the human machine interface or mimic screen.

### *Bytronic Ladsim*

This is a very useful commercially available programme for students to construct ladder diagrams, test them and debug them. You will find a full description at this web address.

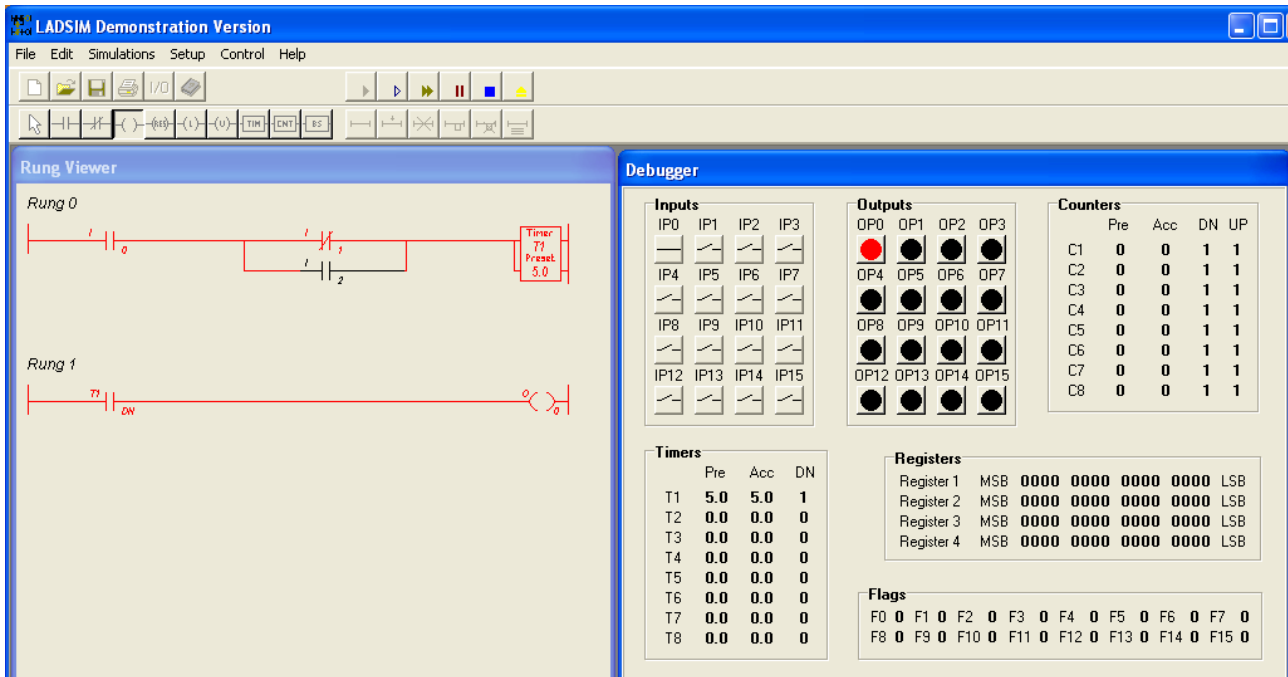
<http://www.edusoft.co.za/ladsim.htm>

The full version of this software includes some standard problems such as traffic lights and car park control and these have a great simulation to enable you to see it working. It is highly recommended that colleges use this software for training.

You can download a free evaluation version of this at this address.

<http://www.bytronic.net/html/downloads.html>

The picture below shows the screen you get when you run this free software. It has many of the best features of programming software. You can drag and drop symbols to form the ladder diagram. When you run the programme the inputs may be activated by clicking on the box (IP0 to IP15). In this example IP0 has been closed making the timer T1 run for 5 seconds. The timer has closed as indicated by the completion of 5 seconds and a '1' appearing under DN (for down). The second rung shows that the timer has closed and operated output OP0 as indicated by the red spot. The ladder diagram turns red as it is activated making it easy to see the effect of operating the various switches.



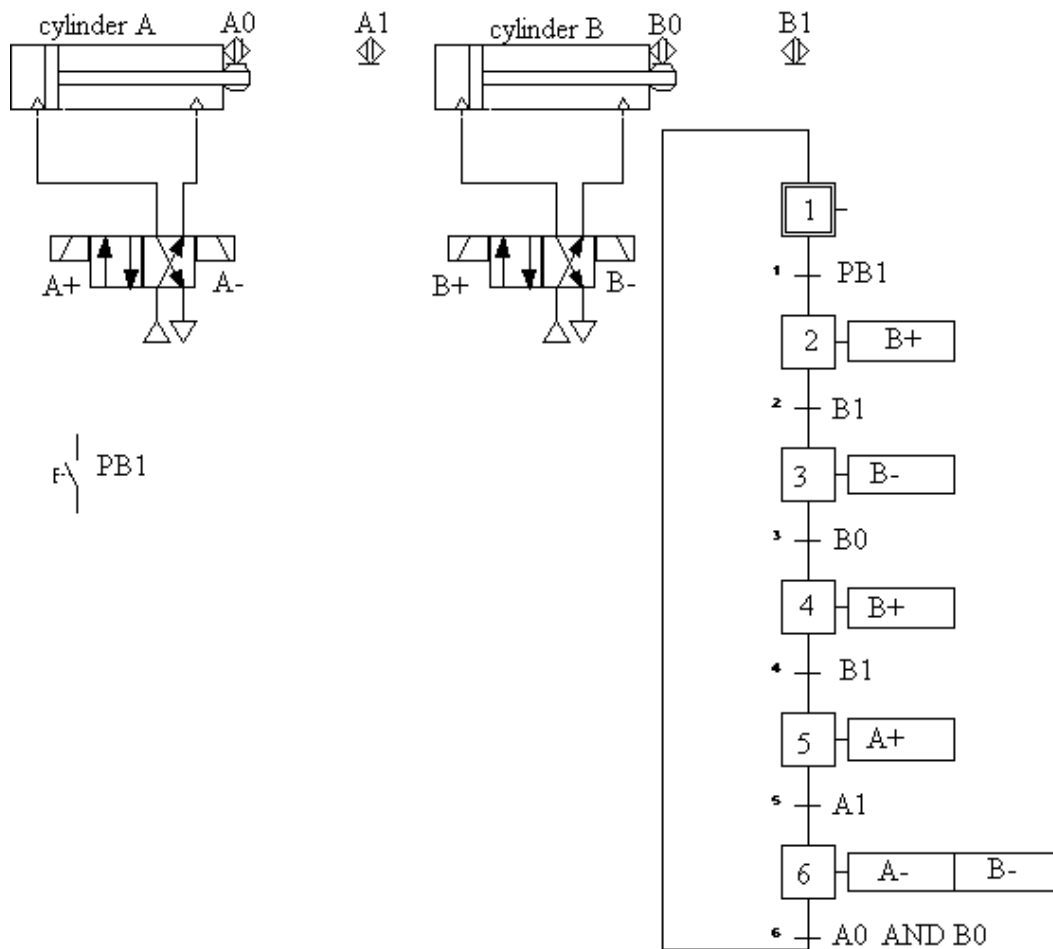
The assignments on the web site [www.freestudy.co.uk](http://www.freestudy.co.uk) include some Ladsim exercises.

## Automation Studio

This is an advanced suite of computer programmes available from this web address.

<http://www.automationstudio.com/EDUC/index.htm> You may also download movie demonstrations and this is recommended.

It has many features for constructing PLC programmes and mimic diagrams in various forms for hydraulic, pneumatic and electric systems. Interfaces are available to link the computer to real systems. A typical screen for use with Grafset programming is shown below. When PB1 is closed by clicking on it, the Grafset programme is activated after step 1 and each step progresses in parallel with the simulation on the right. If it sticks at any point then the problem is easily identified as the point in the diagram is highlighted and enables debugging to be used. The company also markets interfaces that enable the computer to be turned into a real PLC and operate real equipment that can be monitored in real time. Again, great care must be taken when forcing any action as unforeseen consequences can arise.



The assignments on the web site [www.freestudy.co.uk](http://www.freestudy.co.uk) include some exercises with this software.

## **7. *Installation and Commissioning***

The installation and commissioning or de-commissioning of systems controlled by a PLC must only be done by competent persons. When this is completed the system is handed over to the operators. Clearly the operators must be trained and qualified to operate the plant. It is a legal requirement for persons working on or around a plant to be trained and competent. For example, on commissioning, the competent persons would complete a certificate to the effect that the system is working satisfactorily and that the operators have demonstrated their ability to do their job.

## **8. *Safety Notes***

In complex applications of PLC control the consequences of errors in the programme must be seriously considered. It would be unacceptable for an error to produce an affect that would be harmful to people or plant. The following are things that should ideally be done.

- The programme should be thoroughly tested on a simulator to ensure it produces the expected results.
- The affects of all possible changes to the settings that can be made by the operator must also be thoroughly tested with 'what if' scenarios where the changes are put in at the simulator and the affects observed.
- Modern programming software should come with tools for debugging the programme as well as for testing and simulating the programme.

In many real applications, the software may be actually controlling the machine and operating a switch on the screen can override the real switch on the machine thus forcing something to happen. Such a system must be used with great care as the result can produce unwanted actions that might damage the machine or make the process do something unintended with danger to persons working on it.

### ***Safety Guards***

Machinery that produces a danger to humans in close proximity must be enclosed to prevent the human from entering the dangerous area. Guards, fences and perimeters should be used for this purpose. These should be fitted with interlocks so that any human entering the danger area will cause the machinery to stop. For example a robot could cause harm by striking or crushing a human near to it.

### ***Emergency Stop***

When a process is stopped during normal operation, such as interruption to the power supply or someone pressing the emergency stop button, they should always be designed stop so that the system is safe. For example actuators including robots should park themselves in a safe position.

The PLC programme must contain the elements that enable the above to take place.