

PROGRAMMABLE LOGIC CONTROLLERS

TUTORIAL – OUTCOME 3 PART 1

This work covers part of outcome 3 of the Edexcel standard module:

UNIT 21799P PROGRAMMABLE LOGIC CONTROLLERS

Outcome 3 is the most demanding of the outcomes and can only be affectively studied with the use of suitable hardware and/or simulation software such as PneusimPro™ or Bytronics™ simulation software. An industrial background will also be of great benefit to students.

SYLLABUS

Methods of programming: ladder and logic diagrams, statement lists, Boolean algebra, function diagrams, BASIC; `C' and Assembler; Graphical Programming language.

Advanced function: less than, greater than, Binary to BCD, calculations, PID control

Producing and storing text : contact labels, rung labels, programming lists, cross referencing.

Testing and debugging: forcing inputs, forcing outputs, changing data, comparing files (tapes, EPROM, disc), displayed error analysis.

Associated elements: contacts, coils, timers, counters, override facilities, flip-flops, shift registers, sequences.

Outcome 3 Programming techniques	Investigates methods of programming PLCs
	Investigates the range and type of advanced functions of PLCs
	Describes the advantages of offline programming.
	Investigates methods of producing and storing text and documentation.
	Investigates methods of testing and debugging hardware and software.
	Identify elements associated with the preparation of a PLC programme.
	Produce and demonstrate a PLC programme of at least 50 instructions for an engineering application.

The work is continued in part 2 where more advanced programming is covered.

1. INTRODUCTION

Let's first recall what PLC's are about. They are used to control automated systems such as manufacturing cells and plant processes. Their origins are in the use of relays to perform automatic functions and the wiring diagrams for these relay logic circuits resembled a ladder. PLCs replace the physical relays with imaginary ones that are part of the programme. They are physically connected to a set of input sensors/switches and control the on/off status of the output contacts. The programme within the PLC determines the way the inputs control the outputs. There are many methods of programming a PLC and this outcome is mainly about these.

The basic programming methods are the same for each manufacturer but the PLCs that they manufacture have wide differences in their capabilities and protocols. Each manufacturer produces their own system for programming their PLCs and will often supply the hardware and software to do it. There is no universal programming hardware/software that will allow you to programme any type of PLC, although some progress has been made towards developing a common method.

2. REVISION and SUMMARY OF PROGRAMMING TERMS

Tags and Labels and identifiers

Tags, labels and identifiers are the names and symbols given to hardware components to identify them in the PLC programme.

Flip flops and Darlington Pairs

These are methods for converting two data lines into one and vice versa.

Flow Charts

These are more a programming aid than a method of programming and it is explained later. A flow chart is also known as an Algorithm.

Step Functions

This is a simpler method of programming where the control action is a set of sequential steps. The programme can be produced as a ladder diagram with the special condition that each rung is activated in sequence and it is then called a STEP LADDER PROGRAMME. PLCs that are specifically designed for step functions are explained later.

Function Diagrams

This is a system that was produced in France in an effort to produce a standard graphical method that would work with any PLC (so long as the manufacturer designed it to accept the method). It is popularly known as Grafcet.

Ladder Logic Diagrams

This is a popular method that allows you to produce the logic control circuits as rungs on a ladder as described briefly in Outcome 2. Programmes may be designed on a computer in which the ladder diagram is constructed. It is important to remember that each rung of the ladder is an independent circuit that can be activated at any time and not in any specific order.

Statement List

This is a more basic and more difficult method. The programme is produced as a series of statements using mnemonics. Most ladder diagrams can be automatically converted into a statement list and vice versa.

Advanced Graphics

Some manufacturers produce state of the art graphical programmes that allow you to construct a virtual system and simulate it. The programme is automatically generated.

Truth Tables and Boolean Algebra

These were explained in outcome 2 and more fully later. This is a programming tool rather than a method.

Timing Diagrams

This is explained later as a useful tool to help produce programmes.

Let's now look at these in more detail.

3. TAGS , LABELS and IDENTIFIERS

Consider the simple arrangement shown with four input elements connected to the PLC terminals and four output elements.

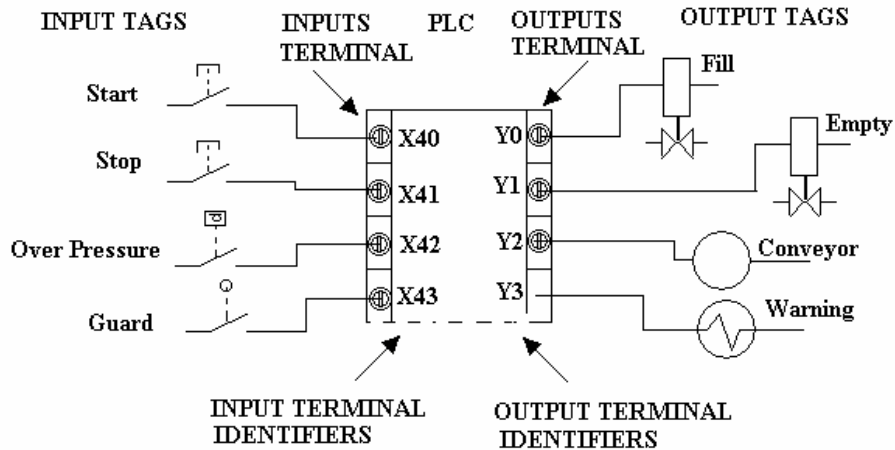


Figure 1

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 help you remember what they represent.

4. FLIP FLOPS and DARLINGTON PAIRS

Consider the circuit shown.

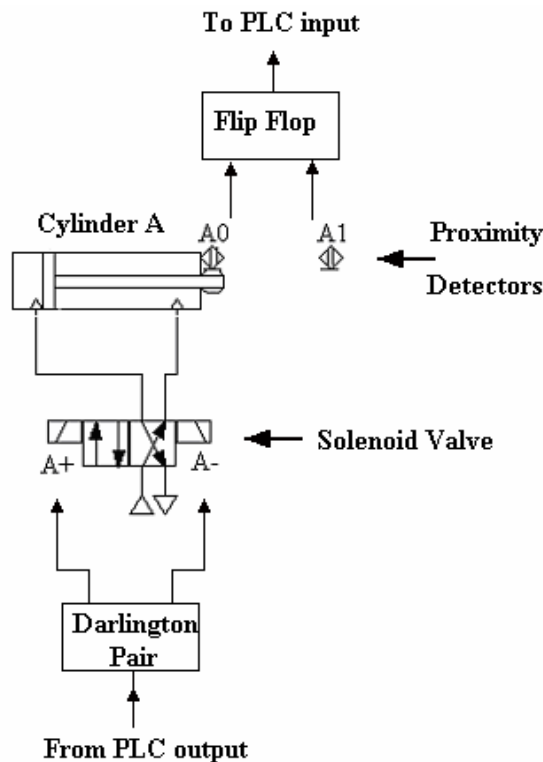


Figure 2

The pneumatic cylinder identifier is A. A Darlington Pair has two outputs connected to the two solenoids. The single input is connected to the PLC terminal and tagged A. If the PLC puts a high signal onto this terminal, the solenoid A+ is switched on and the cylinder extends. If the PLC puts a low signal on the terminal the solenoid A- is energised and the A+ de-energised and the cylinder retracts. In this way only one terminal on the PLC is needed to control the cylinder as either on (high) or off (low). (Note that the same object may be achieved without a Darlington pair by using one solenoid and a spring at the other end).

The cylinder has two proximity switches tagged A1 and A0. The problem is that when the piston is between the sensors, neither sensor is activated and this makes programming difficult. Ideally we wish to connect to a single input terminal on the PLC to indicate whether the cylinder is out (on) or in (off). This is done with a FLIP FLOP. A flip flop has two inputs. The status of the output remains unchanged until the other input changes. The status of the feedback signal remains unchanged when the piston is in between the sensors. Hence the single signal line to the PLC is high (on) when the cylinder is out and low (off) when the cylinder is in and does not change status while moving from one to the other.

5. 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. The main symbols for flow diagrams are shown here.

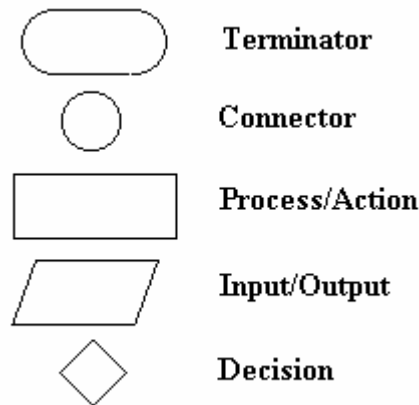


Figure 3

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

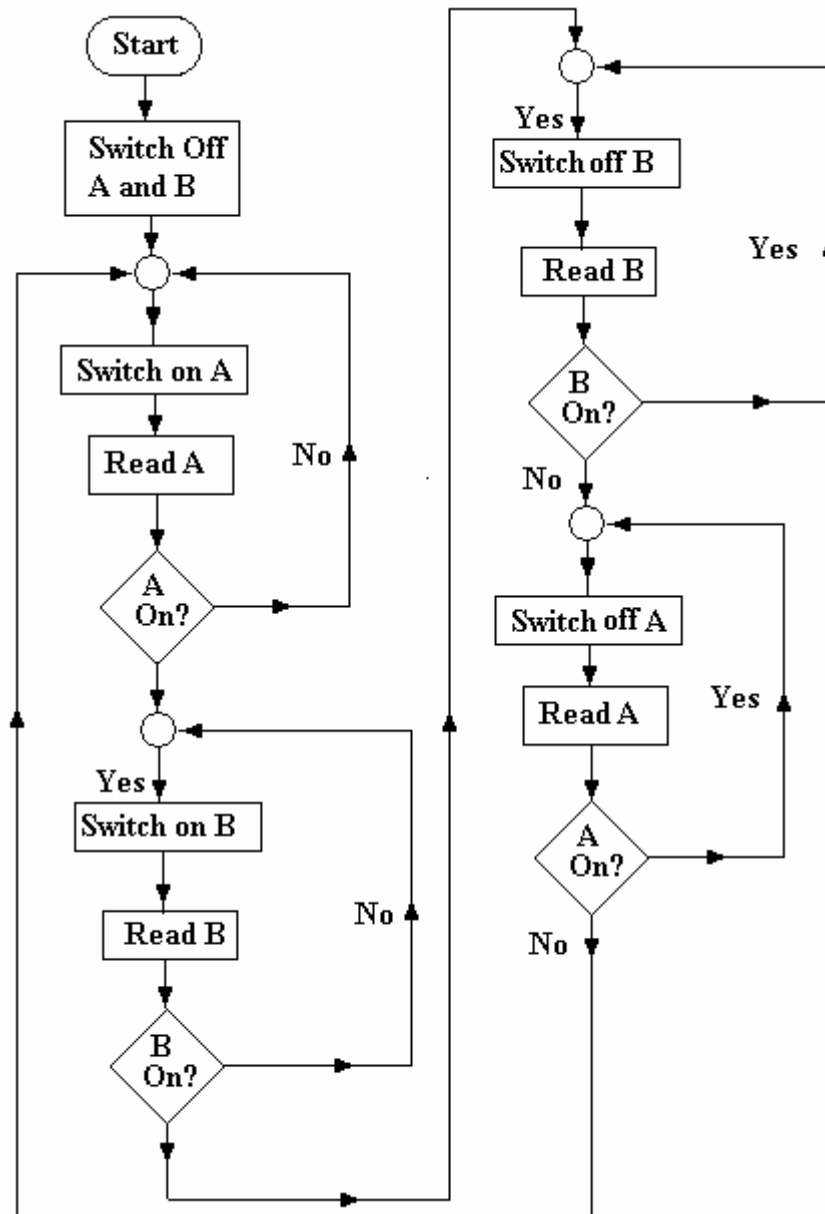


Figure 4

- 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.

6. STEP CONTROLLERS

For a sequence like the one in the last section, the best form of PLC is a dedicated STEP controller. These have input and output terminals in the normal way. They are programmed through a built in panel with programming buttons and a display to show the programme and the state of the inputs and outputs. These conduct the steps one at a time waiting for the correct feedback data before executing the next step. A typical display from a sequential controller is like this.

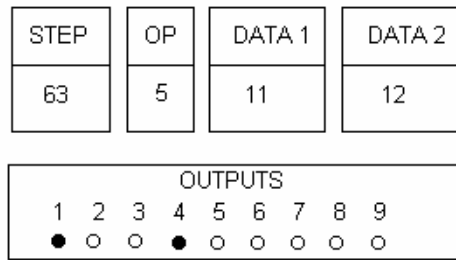


Figure 5

The display informs us that at step 63 in the programme, OUTPUTS 1 and 4 are already switched ON and before it will advance to step 64, the sensors attached to terminals 11 OR 12 must be activated. The OP numbers are codes for logical commands and in this case OP5 is an OR.

Ordinary PLCs can be programmed in this way also if they are designed for it.

7. FUNCTION DIAGRAMS - GRAFCET-SFC

Grafcet (Sequential Functional Chart) is a method of programming PLC's that is gaining in popularity but few PLC manufacturers have produced software to enable it to be done this way. The method is easiest to use with machines that go through a fixed sequence of operations with feed back after the completion of each step to confirm that it has happened (like the previous example) but it is possible to do more advanced things as well. 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/grafcet.html>.

The chart starts with an initial step shown as a double 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+.

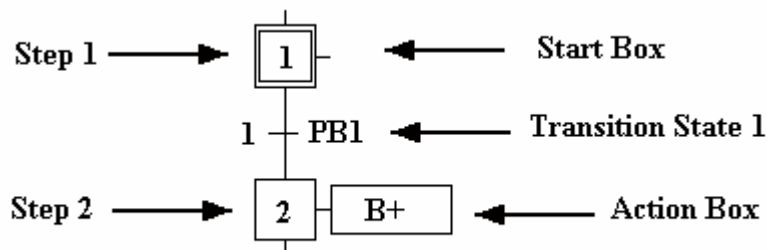


Figure 6

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.

8. OTHER COMPUTER PROGRAMMING LANGUAGES

Some PLCs may be programmed using computer programming language such as C++. Other programming software allows you to use statements like this:

IF A>B AND B<C THEN SET D ELSE RESET D.

This means if A is larger than B and B is smaller than C then set D otherwise Reset D.

Some people might recognise this as similar to basic computer programming language such as BBC Basic and other similar ones used on PCs but now falling into disuse. Elements of this are incorporated in the SFC programming method where statements such as these may be used at the transition state point.

PLCs like the Festo/Beck range use this form of programming.

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.

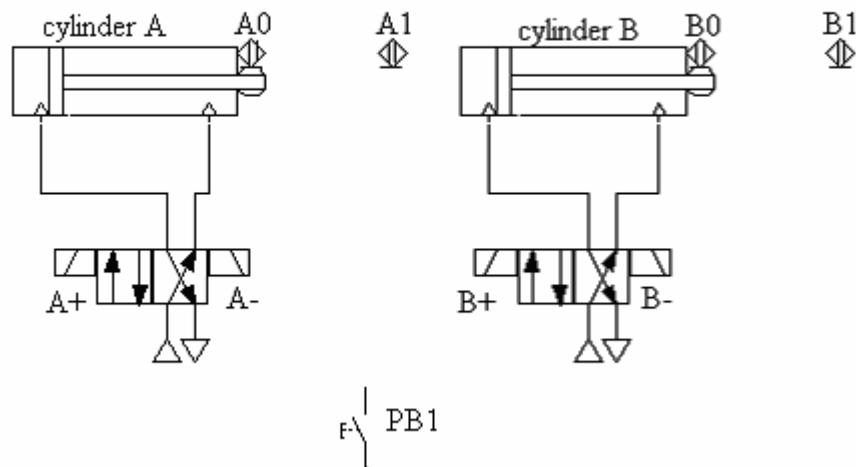


Figure 7

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.

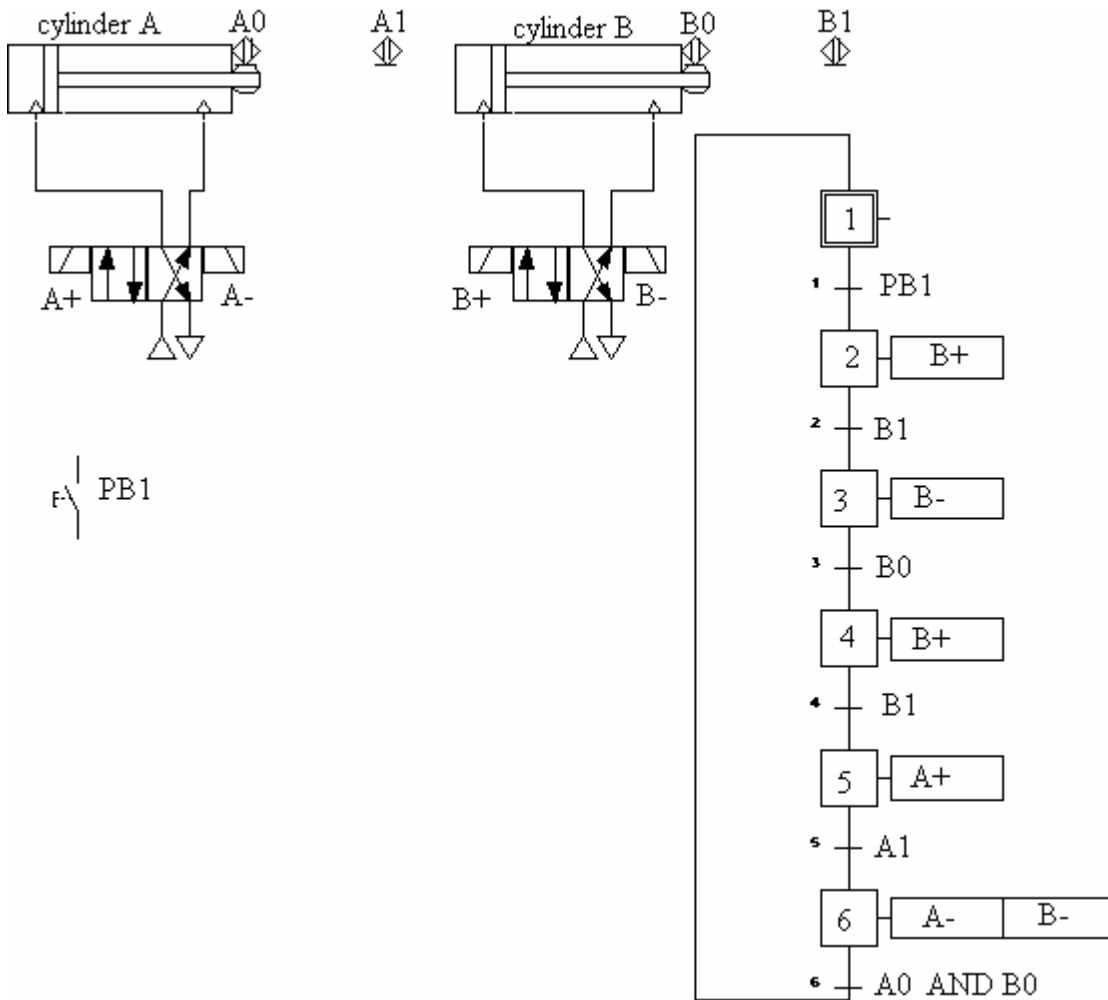


Figure 8

8. LADDER LOGIC DIAGRAMS

INPUT AND OUTPUT ELEMENTS

This was briefly covered in outcome 2. Here we will revise it and take it further. Each rung of the ladder is an imaginary circuit linking the Plus and Minus bus rails. When the circuit is made, current flows and the output element is on. We shall only use American symbols here.

Consider the circuit below. The left diagram shows a motor 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.

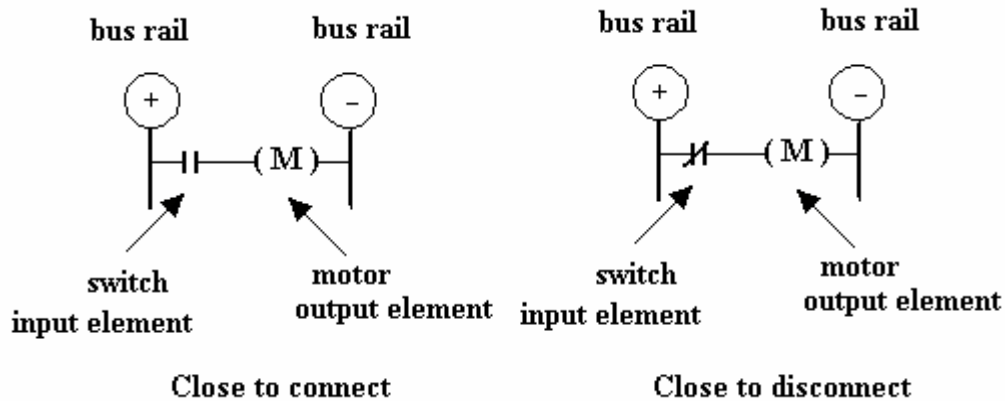


Figure 9

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.

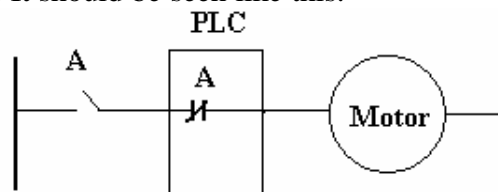


Figure 10

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.

Remember that output elements may be a wide range of things such as heating elements, solenoids, valves and so on. They may also be imaginary relays used for various purposes such as latches and flags. They are commonly tagged with L, M or F.

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.

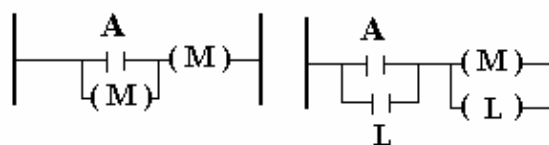


Figure 11

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.

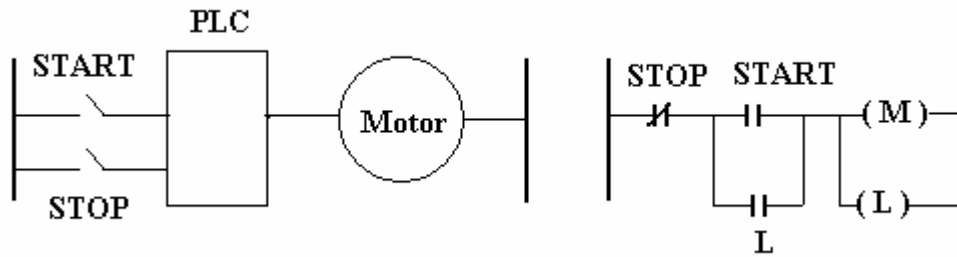


Figure 12

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.

LOGIC FUNCTIONS

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

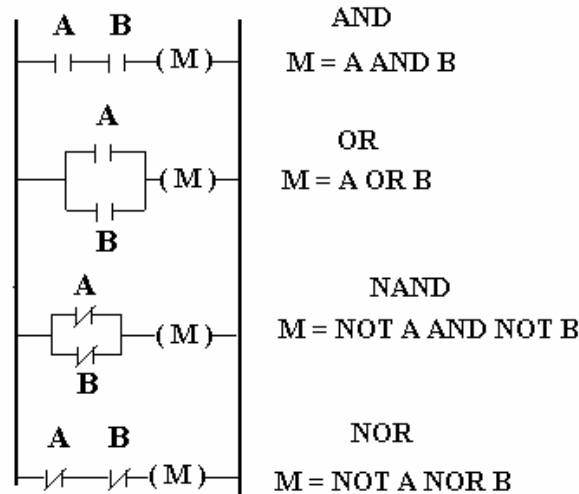


Figure 13

MORE 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 per cylinder 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.

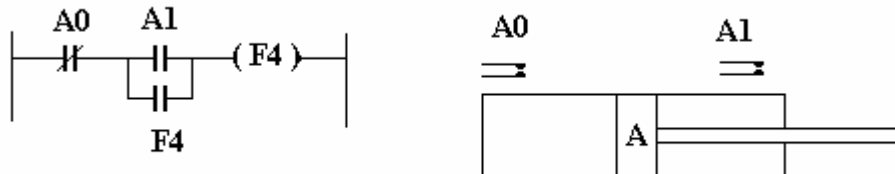


Figure 14

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.

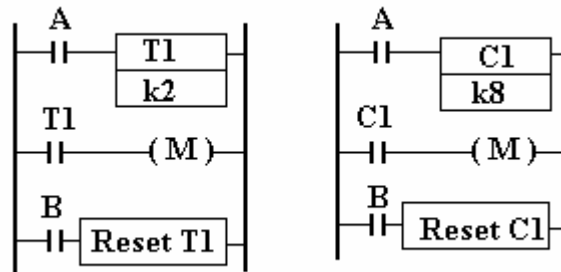


Figure 15

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

Many items of equipment such as traffic lights, may work entirely off timers. In order to help produce a PLC programme a timing diagram is very useful. These are useful when using timers to control a sequence and when the same set of conditions exist more than once in a cycle.

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 at any time, the system will continue to the end of the cycle and then 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.

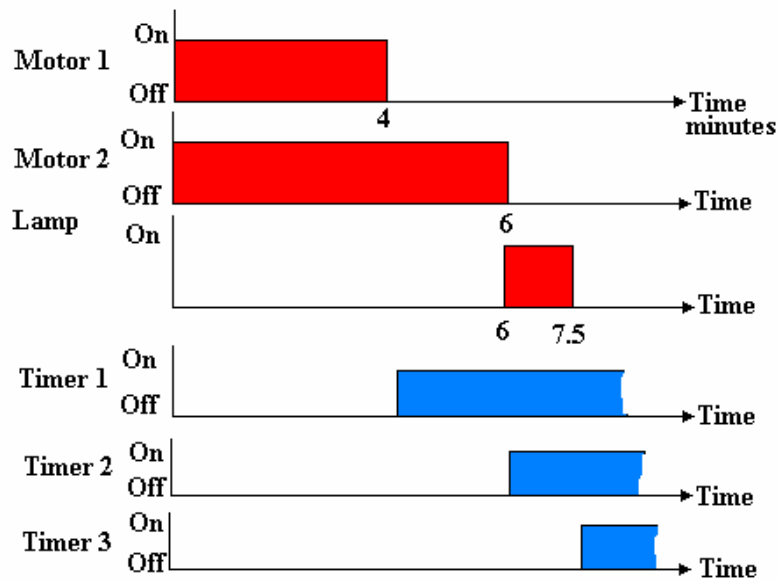


Figure 16

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.

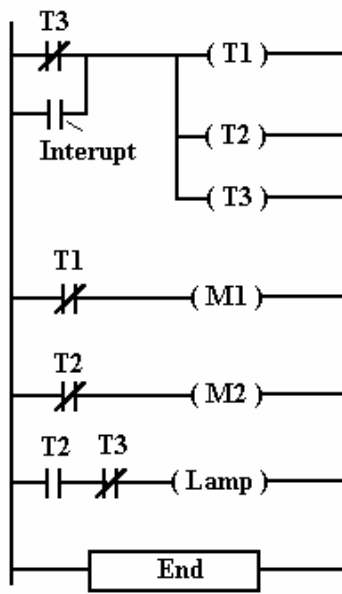


Figure 17

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.

An alternative for Mitsubishi PLC is to use special relay M77 to interrupt the cycle by disabling all outputs. This would be activated by the interrupt switch in series with T3 to stop the cycle when the lamp comes off.

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 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.

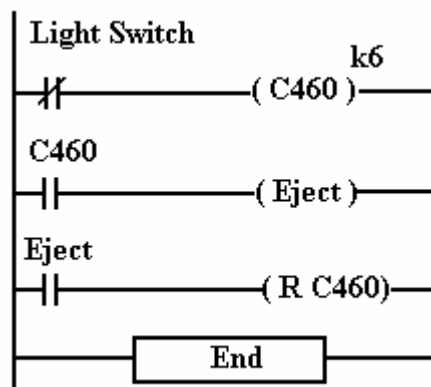


Figure 18

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.

9 INSTRUCTION SETS

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]

Figure 19

This is the statement list for the last ladder programme.

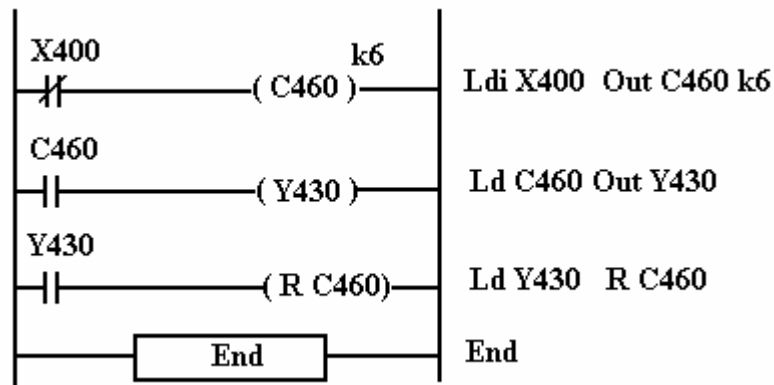


Figure 20

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.

10 TRUTH TABLES

This was explained in Outcome 2 but we need to examine them again. 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.

S1	S2	S3	S4
8	4	2	1
/			

← BINARY VALUE
WHEN ON

MSB

Figure 21

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.

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

Figure 22

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.

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

Figure 23

11. BOOLEAN ALGEBRA

This was partially covered in outcome 2 but we need to refresh ourselves on the subject. It 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.

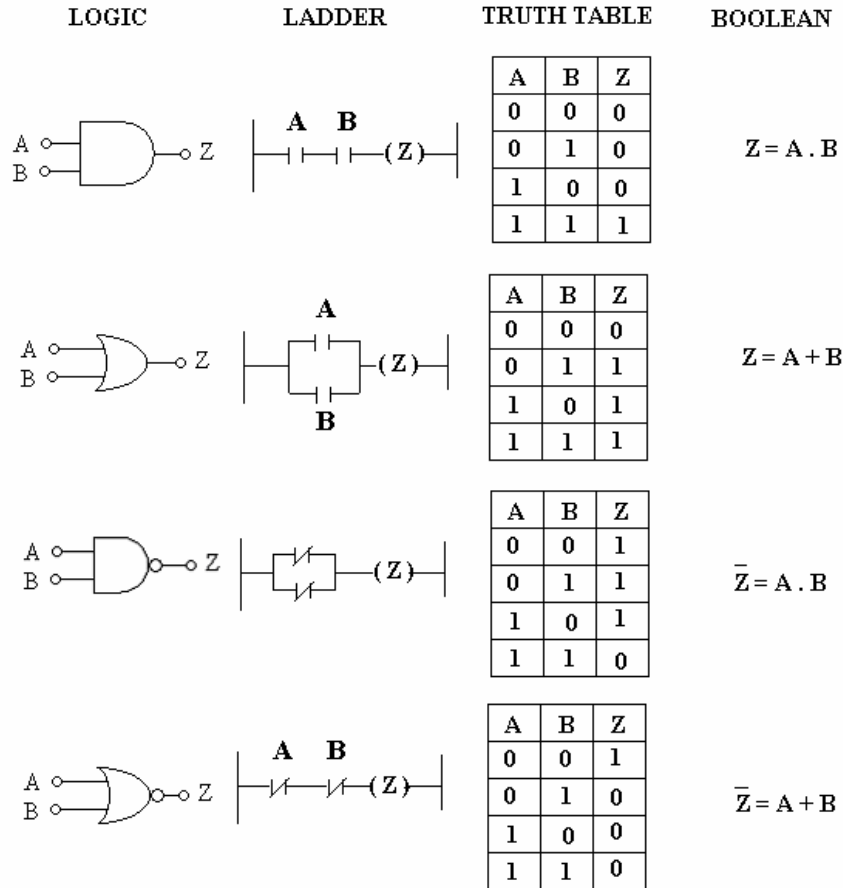


Figure 24

Here is another example.

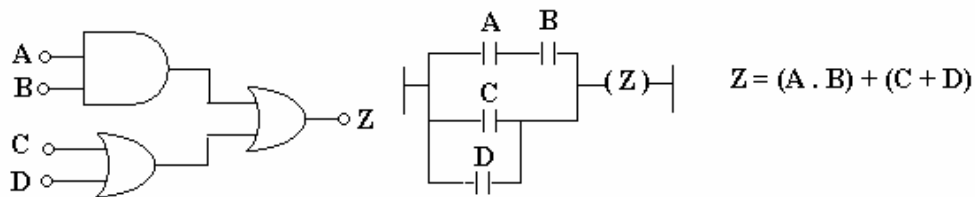


Figure 25

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

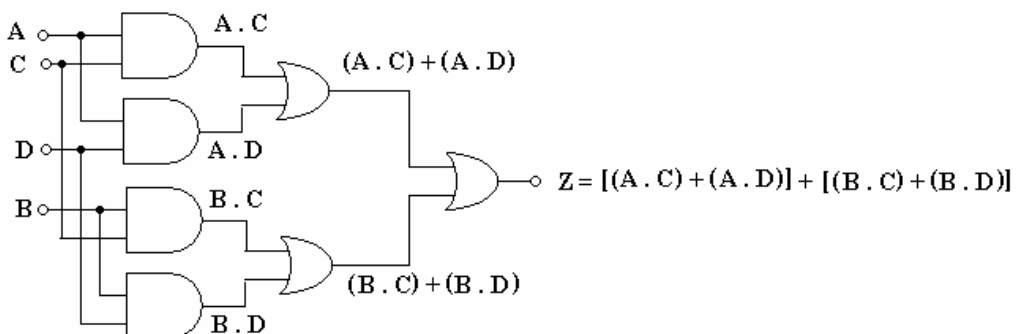


Figure 26

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.

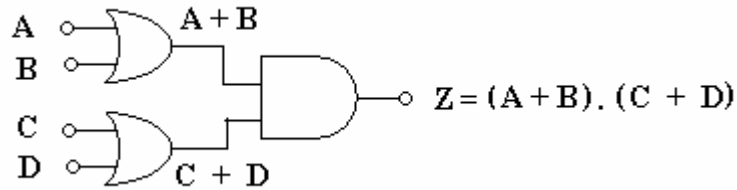
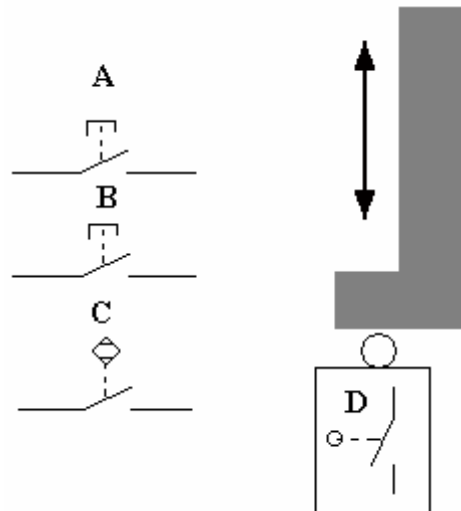


Figure 27

SELF ASSESSMENT EXERCISE

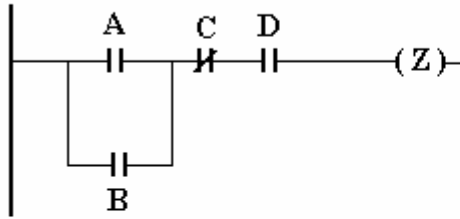
A machine is switched on by either of two push buttons A or B. There is a safety barrier on the machine with a limit switch D at the closed position. In addition there is proximity switch C to detect if anyone is standing inside the barrier. If there is the machine must not start. All have normally open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

ANSWERS

$$Z = (A + B) \cdot \bar{C} \cdot D$$



	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

Ld A Or B Ani C And D