

Pearson BTEC Levels 4 Higher Nationals in Engineering (RQF)

# Unit 15: Automation, Robotics and Programmable Logic Controllers (PLCs)

## Unit Workbook 2

in a series of 2 for this unit

Learning Outcome 2

# PLC Program Design

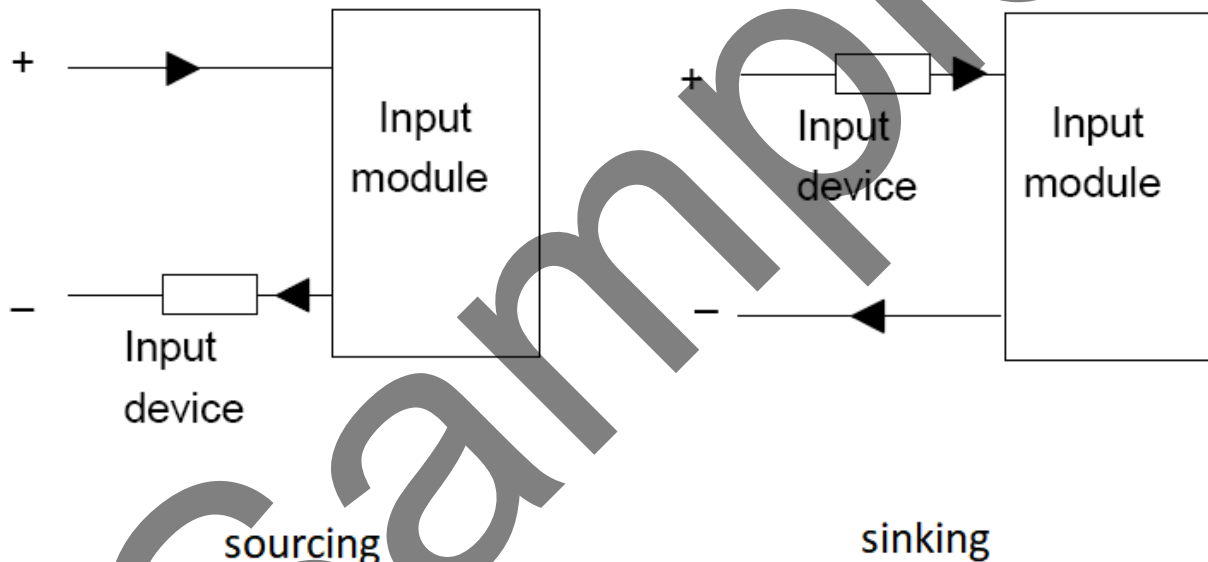
## Programming Language

### Signal Types

Signals from sensors, and to output devices, can be in one of three forms;

- Analogue – a continuously varying signal amplitude with time.
- Discrete – basically just on and off
- Digital – pulses with flat tops and bottoms

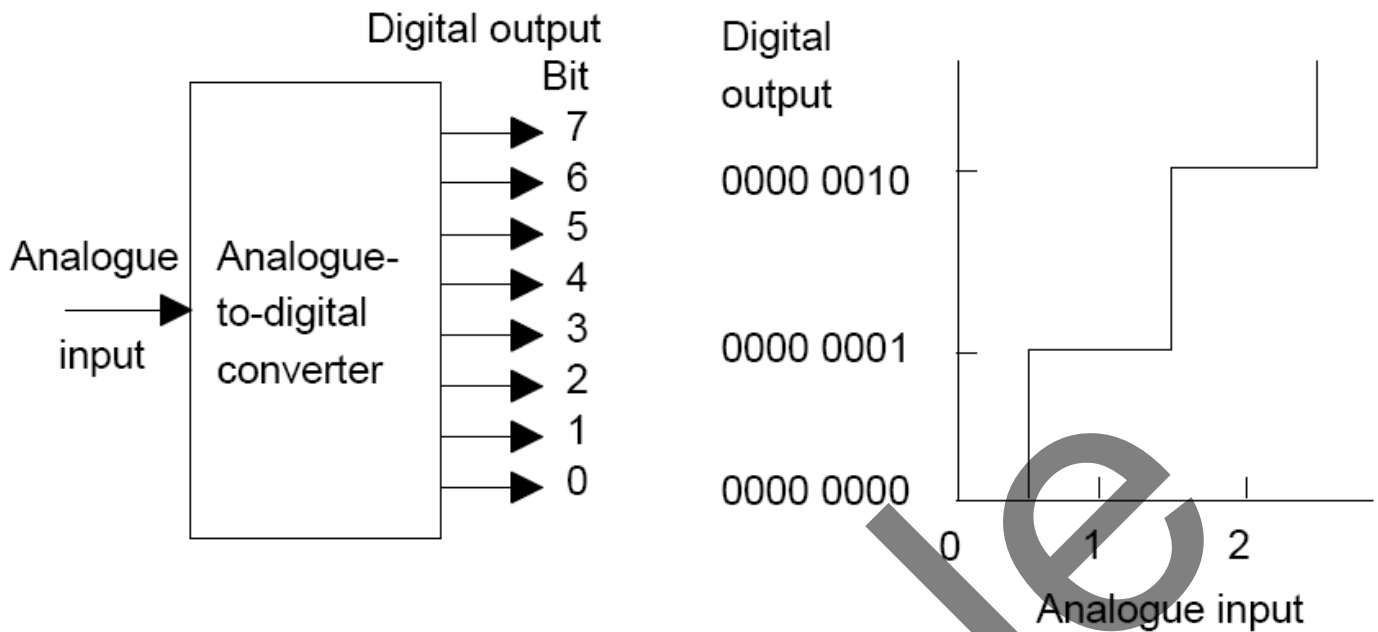
The CPU inside the PLC will, however, only respond to digital signals of a particular amplitude. Usually 0 volts to represent logic 0, and +5 volts to represent logic 1. This means that input and output signals must be manipulated so that they are presented in the correct form for the CPU. This is a job for the PLC input/output units.



**Figure 1:** Current sourcing and current sinking

When the PLC input unit supplies current to the input device, we term this *current sourcing*. When it is the input device itself that supplies the current to the input unit, we term this *current sinking*, as in Figure 1.

To convert an analogue signal into a digital signal we use an ADC (analogue-to-digital converter).



**Figure 2:** Analogue-to-digital conversion

Figure 2 shows an 8-bit ADC. The analogue signal is input to the ADC, which then converts the signal amplitude at each moment in time into a digital code. For an 8-bit ADC there are  $2^8 = 256$  possible digital codes to represent the input signal. These will range from 00000000 (zero) to 11111111 (255).

Should the analogue input signal vary between 0 V and 10 V then this will give us a resolution of  $10/255$ , which is around 4 mV. Therefore, if the input signal is increased by 3 mV there will be no change in the digital output code. If the input signal is increased by 8 mV then this will cause the digital output to increase by two steps. Should we use an ADC with 16 bits then there will be  $2^{16}$  possible digital codes i.e. 65,536 of them. In this case, the resolution will be  $10/65,535$ , which is around 0.15 mV. Therefore, the resolution will improve (get smaller) if we use an ADC with a greater number of bits.

## Number Systems

The basic types of number we deal with regularly in engineering are:

- **Natural** – Everyday positive whole counting numbers (i.e. 0, 1, 2, 3, 4, 5, ...)
- **Integer** – Similar to Naturals, but negatives are allowed (i.e. ...-3, -2, -1, 0, 1, 2, 3, ...)
- **Rational** – Can be expressed as a fraction of two integers (0 not allowed in denominator, i.e.  $-9/5$ )
- **Irrational** - Can NOT be expressed as a fraction of two integers (i.e.  $\sqrt{2}$  )
- **Real** – ANY quantity along a continuous line (i.e. -16.76449)
- **Complex** – Have real and imaginary components. Read more on these later

Common number systems encountered in engineering are:

- **Binary** – Base 2
- **Octal** – Base 8
- **Denary** – Base 10 (what you're most used to)
- **Duodecimal** (also known as **Dozenal**) – Base 12
- **Hexadecimal** – Base 16

To work efficiently in modern engineering, you need to be able to not only appreciate each of these number systems but also convert between them. You can always check your answers by using the Windows Calculator (click on View then Programmer). Using the calculator is fine, but Assignment 1 asks that you demonstrate knowledge of the mechanics of these conversions. Let's look at each system first and then work out how to go about the conversions in detail.

### Binary System (Base 2)

The Binary system only uses two digits, 0 and 1 (hence the term **Binary**). Let's look at a template for the Binary system...

$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
How many 128's	How many 64's	How many 32's	How many 16's	How many 8's	How many 4's	How many 2's	How many 1's

### Octal System (Base 8)

The Octal system uses eight digits, 0 through to 7 (hence the term **Octal**). Let's look at a template for the Octal system...

$8^7$	$8^6$	$8^5$	$8^4$	$8^3$	$8^2$	$8^1$	$8^0$
How many 2,097,152's	How many 262,144's	How many 32,768's	How many 4,096's	How many 512's	How many 64's	How many 8's	How many 1's

### Denary System (Base 10)

The Denary system uses ten digits, 0 through to 9 (hence the term **Denary**). Let's look at a template for the Denary system (which you got used to as a child) ...

$10^7$	$10^6$	$10^5$	$10^4$	$10^3$	$10^2$	$10^1$	$10^0$
How many 10 millions	How many millions	How many 100,000's	How many 10,000's	How many 1,000's	How many 100's	How many 10's	How many 1's

### Duodecimal System (Base 12)

The Duodecimal system uses twelve digits, 0 through to 9, then 'T' for ten and 'E' for eleven (hence the alternative term **Dozenal**, meaning dozen). Let's look at a template for the Duodecimal system ...

$12^7$	$12^6$	$12^5$	$12^4$	$12^3$	$12^2$	$12^1$	$12^0$
How many 35,831,808's	How many 2,985,984's	How many 248,832's	How many 20,736's	How many 1,728's	How many 144's	How many 12's	How many 1's

### Hexadecimal System (Base 16)

The Hexadecimal system uses sixteen digits, 0 through to 9 plus A through to F (hence the term **Hexadecimal**). To employ 16 digits, we have used the letters A to F to represent the numbers 10 to through to 15 respectively. Let's look at a template for the Hexadecimal system...

$16^7$	$16^6$	$16^5$	$16^4$	$16^3$	$16^2$	$16^1$	$16^0$
How many etc.	How many etc.	How many etc.	How many 65,536's	How many 4,096's	How many 256's	How many 16's	How many 1's

### Representing Numbers in Each System

If we wanted to represent the everyday Denary (base 10) number 12 in Binary format then we would have to work from left to right in the Binary table...

How many 128's in 12?      Answer: 0      so put 0 in that cell

How many 64's in 12?      Answer: 0      so put 0 in that cell

How many 32's in 12?      Answer: 0      so put 0 in that cell

How many 16's in 12?      Answer: 0      so put 0 in that cell

How many 8's in 12?      Answer: 1      so put 1 in that cell and subtract the 8 from the 12, leaving 4

How many 4's in 4?      Answer: 1      so put 1 in that cell and subtract the 4 from the 4, leaving 0

There is nothing left over here, so we've finished. We may now write  $12_{10} = 00001100_2$ .

Some examples of starting with a Denary number and representing it in the Binary, Octal and Hexadecimal formats now follow...

### Binary System (Base 2)

Denary Examples	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
12	0	0	0	0	1	1	0	0
17	0	0	0	1	0	0	0	1
57	0	0	1	1	1	0	0	1
129	1	0	0	0	0	0	0	1

### Octal System (Base 8)

Denary Examples	$8^7$	$8^6$	$8^5$	$8^4$	$8^3$	$8^2$	$8^1$	$8^0$
7	0	0	0	0	0	0	0	7
17	0	0	0	0	0	0	2	1
855	0	0	0	0	1	5	2	7
1,067,003	0	4	0	4	3	7	7	3

### Hexadecimal System (Base 16)

Denary Examples	$16^7$	$16^6$	$16^5$	$16^4$	$16^3$	$16^2$	$16^1$	$16^0$
15	0	0	0	0	0	0	0	F
33	0	0	0	0	0	0	2	1
99,999	0	0	0	1	8	6	9	F
200,000,000	0	B	E	B	C	2	0	0

### Conversions between bases:

Denary to Binary, Denary to Octal and Denary to Hexadecimal conversions are all performed in the manner described in the above tables.

Binary to Denary, Octal to Denary and Hexadecimal to Denary conversions are all performed in the reverse manner to those conversions already stated.

That leaves us with six types of conversion to deal with:

- Binary to Octal
- Octal to Binary
- Binary to Hexadecimal

## Hexadecimal Addition

To add hexadecimal numbers together we need to think of the hex number system (0 to F) and start at the first digit to be added to. We then count to the right. Once we move right and pass F we restart (loop) to the left at 0 until we have counted the number of digits for the number we wish to add.

Let's clarify this process by first of all listing the digits in the hex system ...

**0 1 2 3 4 5 6 7 8 9 A B C D E F**

Now assume we wish to add the hex numbers 8 and A. We start at 8 and then move right by A places (i.e. 10 actual places). Once we have moved right by 7 places we reach the end, so need to go back to the start and move right by the remaining 3 places, which will take us to 2. Since we have looped once we note down that 1 loop and then append where we ended up, i.e. that 2. The answer in hex is therefore 12 (one lot of 16 plus 2 is 18, of course, in decimal).

Here is a useful table for hex addition ...

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

To use the table, locate the first digit to be added in the orange row. Then, locate the second digit to be added in the green column. Where your selections meet is the answer to the addition. For example, to add together hex 9 with hex A we locate the 9 in the orange row. Then, locate the A in the green column. Where these two cells meet is the answer i.e. hex 13 in this case. Here is an example use of the table for that particular problem ...

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

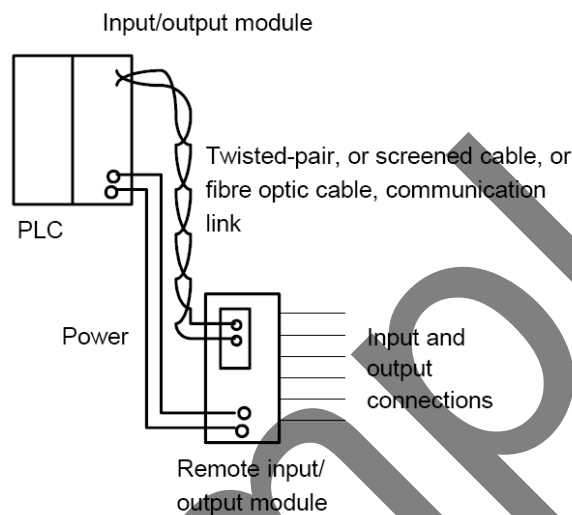
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Sample



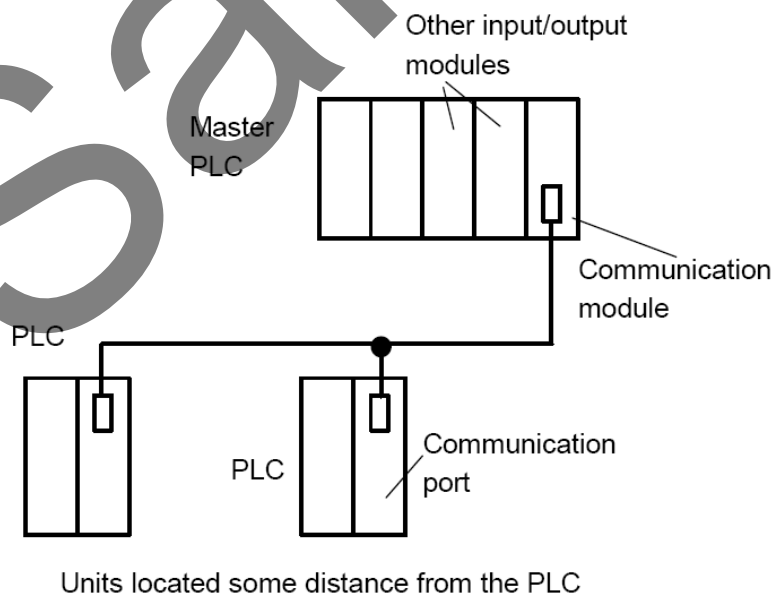
## Communication Techniques

When there are many inputs and outputs located at some distance from the main PLC it is not practical or wise to run many cables from these locations to the PLC. Instead we use input/output modules in the remote areas and then just run a twisted pair, screened coaxial or fibre cable to the main PLC, as in figure 3.



**Figure 3:** *Communicating with remote inputs/outputs*

Some situations might call for a number of slave PLCs communicating with the main PLC, as in figure 4.

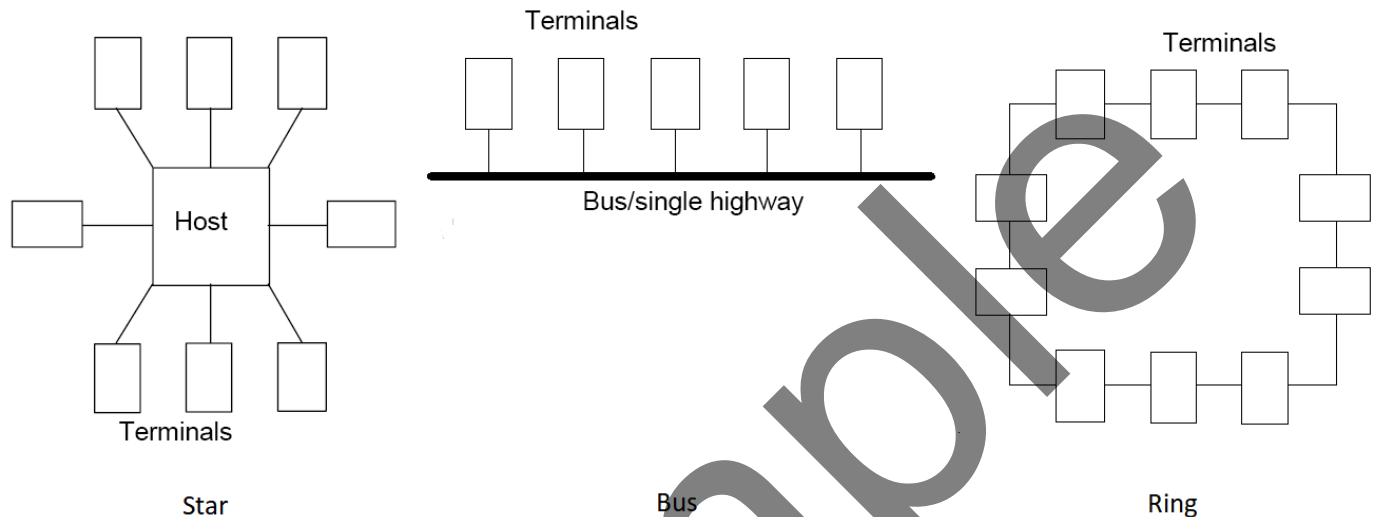


**Figure 4:** *Using a Master PLC to communicate with Slave PLCs*

## Network Methods

Modern industrial plants these days need interconnections between PLCs, PCs, robots, tablets, mobile phones and machinery. To achieve this communication need a LAN (local area network) is implemented within or between buildings.

Such networks may take three basic forms; star, bus and ring, as depicted in figure 5.



**Figure 5:** *Star, Bus and Ring LAN topologies*

In the star form the master, or host, is at the centre, whereas the slave terminals are located at the perimeter of the master. The master polls the terminals, in turn, to see whether they wish to communicate with the master.

With the bus topology, all the terminals are on a single cable, so a communications protocol is established to ensure that only one terminal may communicate with the master at any one time.

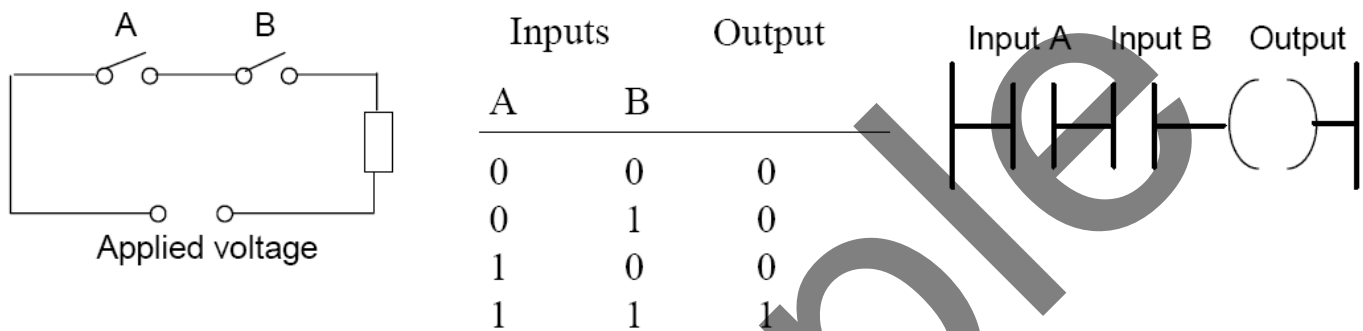
With the ring topology, all the terminals are again on a single cable, so a communications protocol is established to ensure that only one terminal may communicate with the master at any one time. A token is passed around the ring, so that any whichever terminal possesses the token is the only one allowed to communicate. That terminal will then release the token with the address its preferred terminal.

## Logic Functions

### AND function

Consider a simple electrical circuit consisting of an applied voltage source, a load (drawn as a rectangular block), and two switches, A and B, connected in series. The switches can be considered as inputs to the circuit and the load as the output. A truth table is a convenient method of describing the operation of the circuit (figure 6).

edit



**Figure 6:** AND circuit with its Truth Table and Ladder Diagram

As the load only receives current when *both* the A **and** B switches are closed, this circuit can be said to perform a 2-input **AND** function.

In the truth table an open switch is represented by a 0 and a closed switch by a 1. An energised load is represented by a 1 and an unenergised load by a 0.

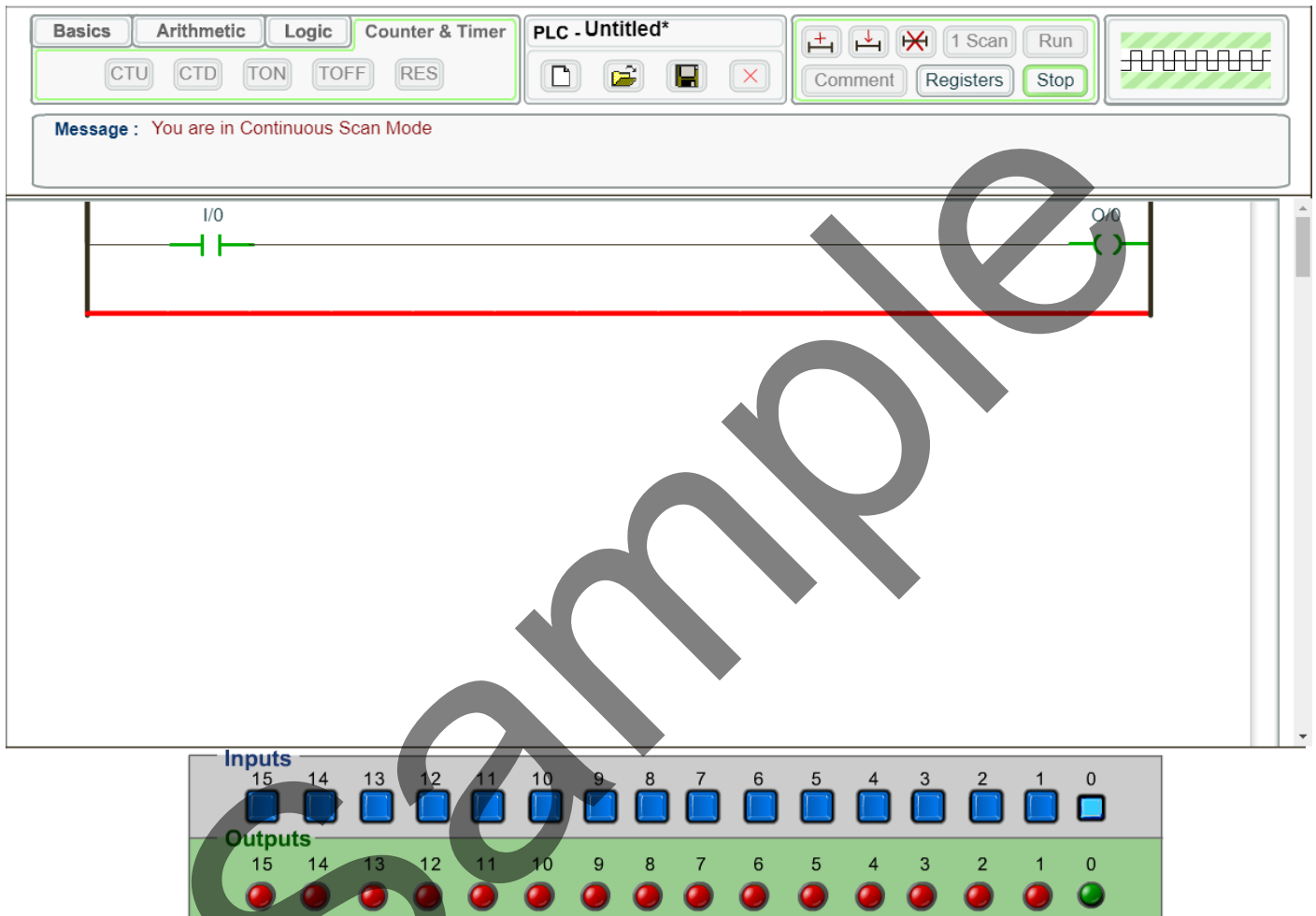
To the right of the truth table we see a Ladder Diagram, as it's known, for a PLC programmed to represent the AND function. The leftmost vertical bar is the positive power rail, and the rightmost vertical bar is the negative power rail (or ground). Between these power rails we see a Rung, drawn horizontally. The rung contains our input and output elements. Here there are three elements drawn. Input A is a normally open switch (yes, it looks like a capacitor, but in the PLC world this symbol is taken as a normally open switch). Input B is another normally open switch, and the Output is our load. In this case the output is a motor, denoted by the curly brackets in the PLC world. When both switch A AND switch B are closed the motor will energise.

You can produce your very first PLC program and replicate/simulate the ladder diagram in figure 6 by using a very useful [browser-based PLC simulator found here](https://plcsimulator.net) at plcsimulator.net (sign up is required, but it is safe). Go to its 'Help' section for useful videos on how to use it.

## Ladder Programs

### Example 1

Use [www.plcsimulator.net](http://www.plcsimulator.net) to design and simulate a ladder diagram for a simple system where a normally-open switch is pressed to drive a motor.



## Example 2

Use [www.plcsimulator.net](http://www.plcsimulator.net) to design and simulate a ladder diagram for a simple system where any one of three normally-open switches, when pressed, will drive a motor.

The screenshot displays a PLC simulator interface. At the top, there are tabs for 'Basics', 'Arithmetic', 'Logic', and 'Counter & Timer'. The 'Logic' tab is active, showing a ladder logic diagram with three normally open contacts labeled I/0, I/1, and I/2 connected in parallel to a coil labeled O/0. Below the diagram, a message box states 'Message : You are in Continuous Scan Mode'. The bottom of the interface shows a physical I/O rack with 16 inputs (blue buttons) and 16 outputs (red LEDs). Input 15 is pressed, and output 0 is lit green, indicating the motor is running.