© 2018 UniCourse Ltd. All Rights Reserved.

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

Sample

edexcel Approved Centre

© 2018 UniCourse Ltd. All Rights Reserved.

Contents

Sample

edexcel Approved Centre

© 2018 UniCourse Ltd. All Rights Reserved.

INTRODUCTION

Design a simple PLC program by considering PLC information, programming and communication techniques

Programming language:

Signal types

Number systems (binary, octal, hexadecimal)

Allocation lists of inputs and outputs

Communication techniques

Network methods

Logic functions (AND, OR, XOR)

Associated elements (timers, counters, latches)

Test and debug methods:

Systematic testing and debugging methods

Proper application of appropriate testing and debugging methods

Sample

edexcel Approved Centre

© 2018 UniCourse Ltd. All Rights Reserved.

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.

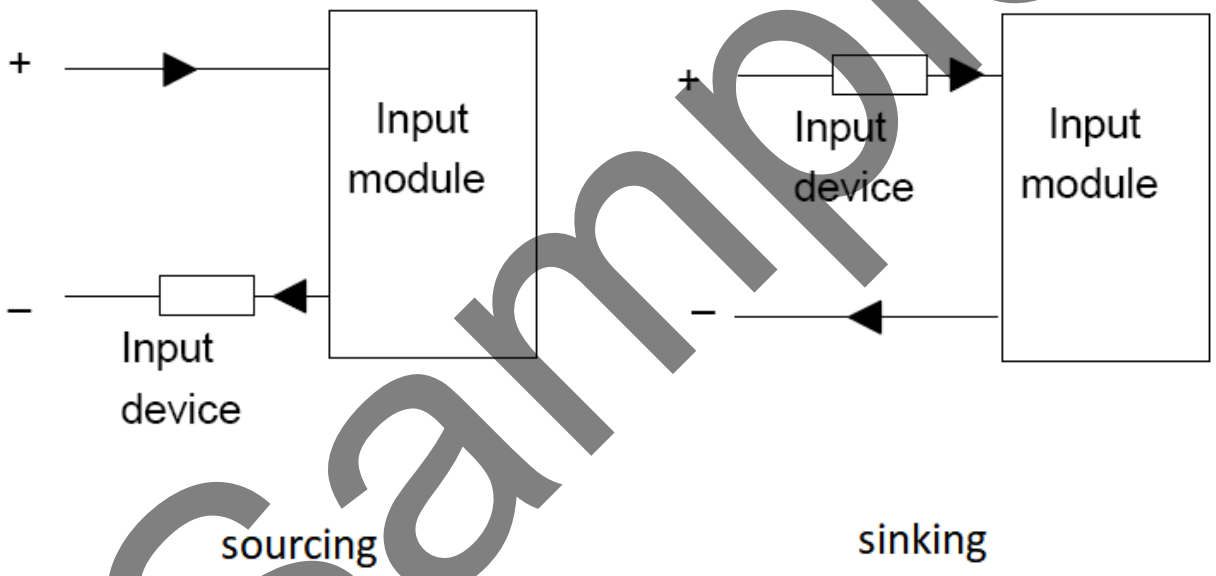

Figure 1: *Current sourcing and current sinking*

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).

Sample

© 2018 UniCourse Ltd. All Rights Reserved.

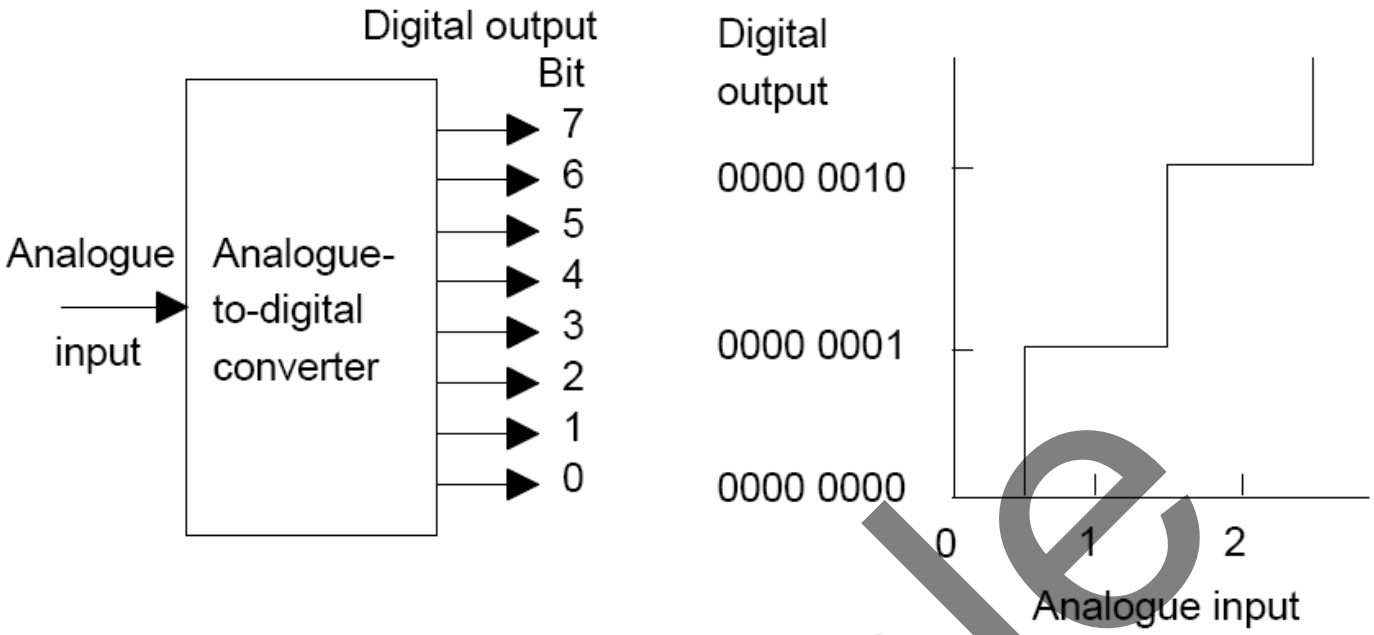

Figure 2: *Analogue-to-digital conversion*

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, is the input signal increased by 3 mV there will be no change in the digital output code. If the input signal 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.

Sample

edexcel Approved Centre

© 2018 UniCourse Ltd. All Rights Reserved.

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.

Sample

Binary System (Base 2)

The Binary system only uses two digits, 0 and 1 (hence the term **Bi**nary). 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 **Oct**al). 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