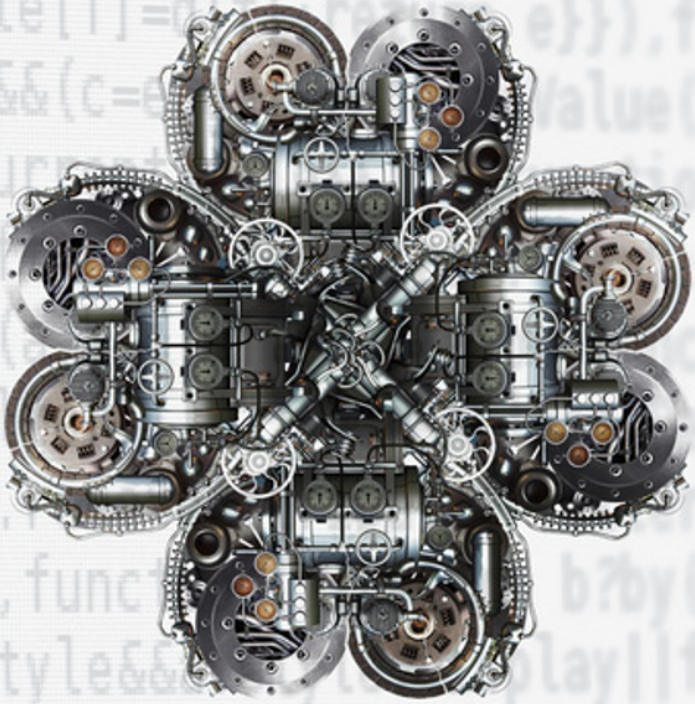
DAG H. HANSSEN



PROGRAMMABLE LOGIC CONTROLLERS

A PRACTICAL APPROACH TO IEC 61131-3 USING CODESYS®

WILEY

PROGRAMMABLE LOGIC CONTROLLERS

PROGRAMMABLE LOGIC CONTROLLERS

A PRACTICAL APPROACH TO IEC 61131-3 USING CODESYS

Dag H. Hanssen
Institute of Engineering and Safety, University of Tromsø, Norway

Translated by
Dan Lufkin

WILEY

This edition first published 2015

Registered Office
John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

Authorised Translation from the Norwegian language edition published by Akademika forlag, Programmerbare Logiske Styringer – basert på IEC 61131-3, 4. Utgave. This translation has been published with the financial support of NORLA.

Library of Congress Cataloging-in-Publication Data

Hanssen, Dag Håkon, author.
Programmable Logic Controllers: A Practical Approach to IEC 61131-3 using CODESYS / Dag Hakon Hanssen.
pages cm
Includes bibliographical references and index.
ISBN 978-1-118-94924-5 (pbk.)
1. Sequence controllers, Programmable. 2. Programmable logic devices. I. Title.
TJ223.P76H36 2015
621.39′5–dc23
2015018742

A catalogue record for this book is available from the British Library.

Set in 10/12pt Times by SPi Global, Pondicherry, India

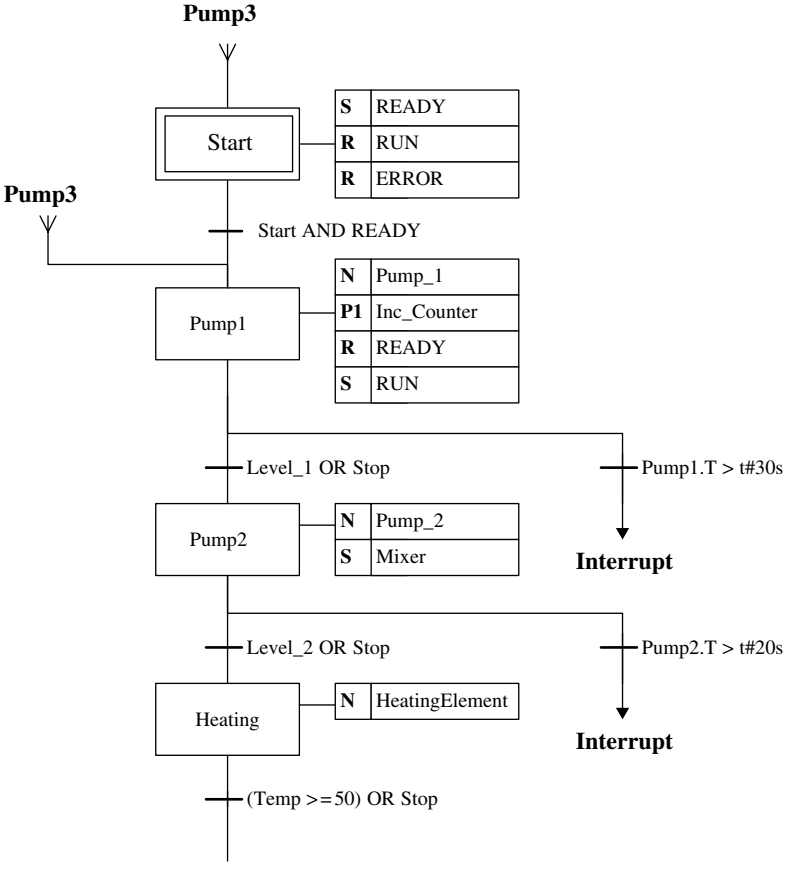
1 2015

Contents

Programmable Logic Controllers

A Practical Approach to IEC 61131-3 Using CODESYS

First edition



Dag H. Hanssen

Preface

As long as there have been competing producers of PLCs on the market, there have been different programming languages from one PLC brand to another. Even though the same languages, beginning with Instruction Lists (IL) and Ladder diagram (LD), have been used by most of the producers, all of them added their own "dialects" to the languages. When physical programming terminals replaced software-based programming tools, the differences between languages of the various producers escalated. Several programming languages also saw the light of day. This development was the natural result of the attempt by the producers to make themselves stand out among increasing competition by developing the most user-friendly languages and tools.

When the IEC[1] 61131-3 standard came out in 1993, the situation started to improve. This standard was the result of the work that had been ongoing for several years in which the best from the various languages and dialects from different producers was assembled into a single document. This is not a rigid standard in the sense that the producers *must* follow all requirements and specifications, but more a set of guidelines that the producers could choose to follow to a certain extent. Today, most of the equipment producers have come to realize the advantages of organizing themselves in accordance with the standard. All of the major producers of PLCs, such as Telemecanique, Wago, Mitsubishi, Klockner Moeller, Allen-Bradley, Omron, Siemens, and so on, have therefore, to a greater or lesser extent, adapted their programming tools to IEC 61131-3.

This book covers close to 100% of the specifications and guidelines that are given in Standard (International Electrotechnical Commission, 2013).[2] The book will therefore be interested to everyone who works with, or wants to learn about programming PLCs, no matter which PLC brand they use.

---

[1] IEC—International Electrotechnical Commission. This edition of the book was updated in conformity with the 3rd edition of IEC 61131-3, issued February 2013.

[2] The Standard IEC 61131-3 is introduced in Chapter 5.

The book does not assume any previous knowledge of programming.
Comments and suggestions for contents will be gratefully received.
The book is divided into five main parts:

- Part 1: Hardware Chapters 1–3
- Part 2: Methodic Chapter 4
- Part 3: IEC 61131-3 Chapters 5–8
- Part 4: Programming Chapters 9–13
- Part 5: Implementation Chapters 14–15

Chapter 1 contains a brief history and a short description of the design and operation of PLCs in general. Chapters 2 and 3 give a basic introduction to digital and analog signals and equipment for detection, measurement, and manipulation of discrete and continuous quantities.

Chapter 4 focuses on methods for planning and design of structurally efficient programs. It also provides an introduction into Boolean algebra. Chapters 5 and 6 introduce the IEC standard elements such as literals, keywords, data types, variables, and addressing. Chapters 7 and 8 cover standardized functions and functional blocks.

Chapters 9 to 13 deal with programming: Chapter 9 covers programming with LD. Chapter 10 covers functional block diagrams (FBD). Chapter 11 covers the structured text (ST) language. The last language covered in the book is actually not a programming language as such, but rather a tool for structuring program code. This is called a Sequential Function Chart (SFC) and is described in Chapter 12.

Chapter 13 contains some larger practical programming examples.

The last two chapters in the book cover programming tools. Here, I have chosen to focus on CODESYS. There are several reasons for this; first, CODESYS follows the standard almost 100%. Furthermore, CODESYS is a hardware-independent programming tool that is currently used by well over 250 hardware suppliers. Finally yet importantly, the program can be downloaded free and it contains a simulator. Most of the program code in the book was written and tested with this tool.

I would like to thank the following persons and companies:

- Associate Professor Tormod Drengstig, University of Stavanger, for much good feedback, suggestions for improvements, and the contribution of several examples
- Assistant Professor Inge Vivås, Bergen University College, for giving his permission to reuse some problems (Section 4.6.4 and Problems 4.10 and 10.5)
- Assistant Professor Veslemøy Tyssø, Oslo and Akershus University College of Applied Science, for having read an earlier edition of the book and having provided expert contributions
- Colleagues and management at the University of Tromsø, Department of Engineering and Safety, for the support and patience
- Schneider Electric for granting me permission to use material from their “Automation Solution Guide” when writing about sensors in Chapter 2

Dag H. Hanssen

Part One
Hardware

1

About PLCs

The programmable logic controller (PLC) has its origin in relay-based control systems, also called hard-wired logic.[1]

Before PLCs became common in industry, all automatic control was handled by circuits composed of relays,[2] switches, clocks and counters, etc (Figure 1.1). Such controls required a lot of wiring and usually filled large cabinets full of electromagnetic relays. Electricians had to assemble controls or use a prepared relay wiring diagram. The relay wiring diagrams showed how all the switches, sensors, motors, valves, relays, etc. were connected. Such relay wiring diagrams are the forerunners for the ladder diagram (LD) programming language, which is still a common programming language used in programming PLCs.

There were many disadvantages with these mechanical controls. In addition to taking up a lot of room, they demand time and labor to implement them and to make any changes in such equipment. A relay control usually consists of hundreds of relays connected together with wires running in every direction. If the logical function needs to be changed or expanded, the entire physical unit must be rewired, something that is obviously expensive in terms of working time. Since the relays are electromechanical devices, they also had a limited service life, something that led to frequent operational interruptions with subsequent disruption.

There also was no way of testing before the control was wired up. Testing therefore had to take place by running the unit. If there was a small failure in the schematic diagram or if an electrician had connected a wire wrong, this could result in dramatic events.

[1] Originally, the designation PC—Programmable Controller—was used. This naturally caused some confusion when Personal Computer became a well-known concept.

[2] A relay is an electromechanical component that functions like an electrical switch. A weak current (so-called control current) activates the switch so that a stronger current can be switched on or off.

Programmable Logic Controllers: A Practical Approach to IEC 61131-3 Using CODESYS, First Edition. Dag H. Hanssen.

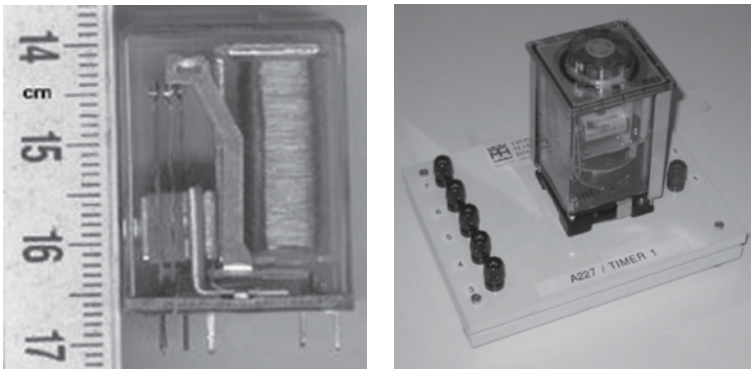


Figure 1.1 Example of a relay and a timer (mounted on a connector board)

1.1 History

The first PLC came into commercial production when General Motors was looking for a replacement for relay controls. Increased competition and expanded demands on the part of customers meant a demand for higher efficiency, and the natural step was to design a software-based system that could replace the relays. The requirement was that the new system should be able to:

- Compete on price with traditional relay controls
- Be flexible
- Withstand a harsh environment
- Be modular with respect to the number of inputs and outputs[3]
- Be easy to program and reprogram

Several corporations started work on providing a solution to the problem. Bedford Associates, Inc. from Bedford, Massachusetts, suggested something they called a "modular digital controller" (MODICON). MODICON 084[4] was the first PLC that went into commercial production. The key to its success was probably the programming language, LD, which was based on the relay diagrams that electricians were familiar with. Today there is no question about the use of programmable controls; the question is rather what type to use.

The first PLCs were relatively simple in the sense that their function was to replace relay logic and nothing else. Gradually, the capabilities improved more and more and functions such as counters and time delays were added. The next step in development was analog input/output and arithmetic functions such as comparators and adders.

With the development of semiconductor technology and integrated circuits, programmable controls became widely used in industry. Particularly when microprocessors came on the market in the beginning of the 1970s, development proceeded at a rapid pace.

[3] This means that it must be possible to increase the number of inputs and outputs by inserting extra modules/boards/blocks. In order to offer cheaper hardware, there are also many PLCs that are not modular.

[4] 084 indicates that it was the 84th project for the company. After that, the corporation established a new company, (MODICON), which focused on producing PLCs.

The PLCs of today come with development tools in the form of software with every imaginable ready-to-use function. Examples are program codes for managing communications as well as processing functions such as proportional integrator/derivative regulators, servo controls, axial control, etc. In other words, there is the same pace of development as with the PC (Figures 1.2, 1.3, and 1.4).

The communications side also experienced rapid development. Demand grew quickly for PLCs that could talk to one another and that could be placed away from the actual production lines. Around 1973 Modicon developed a communications protocol that they called Modbus. This made it possible to set up communications between PLCs, and the PLCs could therefore be located away from production. Modicon's Modbus also provided for management of analog signals. As there became more and more manufacturers of PLCs and

Figure 1.2 Omron Sysmac C20—Nonmodular PLC with digital I/O and programming terminal

Figure 1.3 PLCs from Telemecanique come in different sizes

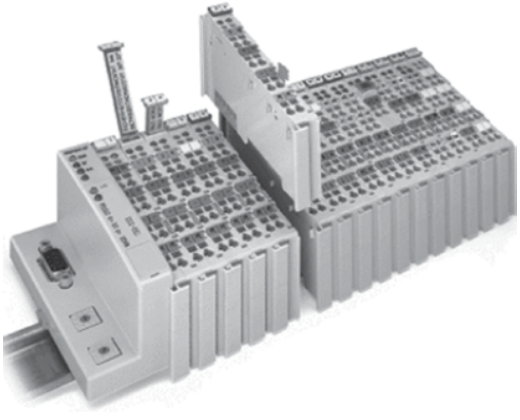


Figure 1.4 Newer generation PLC from Wago with Profibus coupler and I/O

associated equipment, there also developed more proprietary[5] and nonproprietary communications protocols. The lack of standardization, together with continual technological development, meant that PLC communication became a nightmare of incompatible protocols and various physical networks. Even today, there are problems, although manufacturers now offer solutions for communications over a selection of known and standardized protocols.

Several programming languages also came into use. Earlier LD, as we mentioned, was synonymous with PLC programming. *Instruction List* (IL) was also an early language that had many similarities with the assembly language that used for programming microprocessors. Later the graphical language *Sequential Function Chart* (SFC) was added. This was specially developed for implementation of sequential controls.

1.1.1 More Recent Developments

All of the aforementioned languages were incorporated into the international standard IEC 61131-3 (International Electrotechnical Commission, 2013). The standard also defines the *function block diagram* (FBD) graphic language and the *structured text* (ST) language. FBD has a symbol palette that is based on recognized symbols and functions from digital technology. ST is a high-level language that provides associations with Pascal and C.

Before the IEC 61131-3 standard appeared, and for many years thereafter, there were relatively large differences between PLCs from various manufacturers. This was particularly true of capabilities for selection of programming language and how the language that was implemented in the PLCs was designed. Recently, to the delight of users, manufacturers began

[5] A proprietary protocol is owned by the manufacturer who developed it. The source code is not freely accessible. A non-proprietary protocol is either a standard protocol or an open protocol that is distributed by many manufacturers who make equipment for communication over such a protocol.

to follow IEC 61131-3 to a greater and greater extent. This made it easier to go from one brand of PLC to another as well as making it easier, to a certain extent, for customers to know what they were getting.

There are also a number of "software-based PLCs" on the market. As the name indicates, this software is designed to control processes directly from a PC. The challenge has been to build systems that are sufficiently reliable and robust. Industry is generally critical of such solutions, mostly based on experience with many a computer crash.

Another amphibious solution is the possibility of buying a circuit board for a computer onto which the program code can be loaded. The board is made so that it is capable of carrying on with the job independently even if the computer should crash.

In recent years, manufacturers have devoted considerable resources to developing solutions for connecting instruments and actuators into a network. Such a communication bus is called a *fieldbus*, referring to the fact that there is communication between field instruments, in other words, instruments below the process level. Other standards and *de facto*[6] standards are also on the market.

Work on an IEC standard for the fieldbus started as early as 1984/1985. The requirement was naturally that the standard should be an open fieldbus solution for industrial automation. It should include units such as motor controls, decentralized I/O, and PLCs, in addition to the distributed control systems (DCS) and field instruments used in the processing industry. The goal was also that the standard should cover all pertinent areas such as building automation, process automation, and general industrial automation.

It was not until the end of 1999 that those involved came to an agreement. The result was that a total of eight (partially dissimilar) systems were incorporated into a standard called *IEC 61158*. In other words, this was not an open solution. Even though manufacturers and suppliers argued that it was good for users to have plenty of choices, this unity did not make things much easier for engineers and others working on automation.

Several of the major manufacturers currently offer integrated solutions with I/O modules for all of the major fieldbus standards where a controller (PLC) or a gateway manages communication among the various standards simultaneously.

Another trend is that manufacturers of hardware and communication solutions offer more equipment for wireless communication (Ethernet). What is new here is that these also include individual sensors and individual instruments. In this way, it is possible to implement wireless systems right out to the sensor level.

1.2 Structure

As we said, there are a great many types of PLCs on the market. Hundreds of suppliers include PLCs of various sizes in their stock. The smallest PLCs have relatively small memory capacity and calculating capability and usually limited or no capability for expansion of the number of I/Os. The largest have processor power equivalent to powerful computers,

[6] *De facto* is a Latin expression that means "actually" or "in reality." De facto is the opposite of *de jure,* which means "according to law." If something is *de facto*, it means something that is generally recognized. A de facto standard is thus a standard that is so widely used that that everyone follows it as though it were an authorized standard. (*Source:* Wikipedia.)

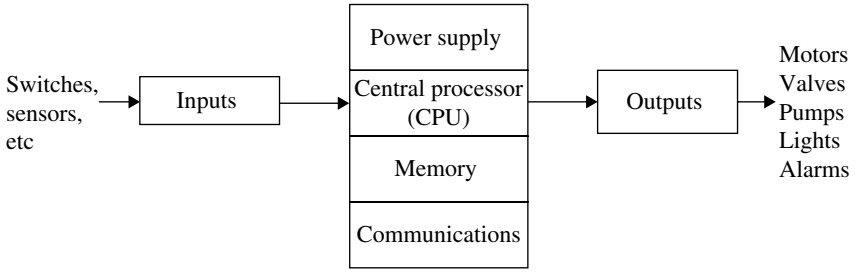


Figure 1.5 Block schematic representation of a PLC

have a large number of I/Os, and handle multitasking.[7] Such PLCs usually have a supervisory function (master) in an industrial data network where smaller PLC types can be incorporated as slaves.

If we make a simplification, we can say that a PLC functions in the same way that a computer does. Schematically, we can break a PLC down into six major units as shown in Figure 1.5.

The main parts thus consist of a central processing unit (CPU), memory, power supply, circuit modules to receive and transmit data (I/O units), and communications modules. We can perhaps also add displays/indicator lights since most of the PLCs incorporate LEDs that indicate the state of the PLC and/or the digital I/Os. Some also have displays that can furnish other information. In order for us to understand how a PLC operates and functions, it is necessary to look a little closer at the main components.

The main units are connected together with wires or copper strips called buses. All communications between the main parts of the PLC take place via these buses. A bus is a collection of a number of wires, for instance, eight, where information is transferred in binary form (one bit per wire in parallel). Typically, a PLC will have four buses: address bus, data bus, control bus, and system bus:

1. The data bus is used for transfer of data between the CPU, memory, and I/O.
2. The address bus is used to transfer the memory addresses from which data will be fetched or to which data will be sent. An address can indicate, for instance, a location down to a word in a particular register. A 16-line address bus can thus transfer $2^{16} = 65\,536$ different addresses.
3. The control bus is used for synchronizing and controlling traffic circuits.
4. The system bus is used for I/O communication.

Central Processing Unit

This is the brain of the PLC. Here are performed all of the instructions and calculations, and it controls the flow of information and how the program operates. Normally the CPU is a part of the physical block and contains the memory, communications ports, status indicator lights, and sometimes the power supply.

[7] Can run several parallel program sessions simultaneously.

Memory

The size of the memory varies from one brand of PLC to another, but the memory can often be expanded by installing an extra memory card, for instance an SD card. A PLC will commonly have the following memory units:

- Read-only memory (ROM) for permanent storage of operating system and system data. Since the information stored in a ROM cannot be deleted, an erasable programmable ROM (EPROM) is used for this purpose. In this way, it is possible to update a PLC operating system.
- Random access memory (RAM) for storage of programs. This is because a RAM is very fast. Since the information in a RAM cannot be maintained without current, PLCs have a battery so that the program code will not be lost in the event of a power failure. Some PLCs also have the capability of program storage in an EPROM. RAMs are also used when the program code is running. This is used, for instance, for I/O values and the states of timers and counters.
- Some PLCs offer the capability of inserting extra memory.

Figure 1.6 shows a typical memory board for a PLC that has an EPROM for a backup copy of the program.

Communications Unit

This unit incorporates one or more protocols for handling communications. All PLCs have a connection for a programming cable and often for an operator panel, printer, or network. Various physical standards are used, for both the programming port and for the ports for connections to other equipment. Current PLCs are usually programmed from an ordinary PC with a programming tool developed for that particular type of PLC.

It is not always necessary to have a direct connection between the PLC and the PC in order to transfer the program code to the PLC. However, it is currently the most common approach—at least for smaller systems. Sometimes, the programming can be performed via a network consisting of several PLCs and other equipment or via Ethernet. Some PLCs also have a built-in web server.

The development of instrumentation buses has enabled PLC manufacturers to supply built-in, or modular, solutions for communications via a large number of various protocols. Examples of such are the AS-i bus, PROFIBUS, Modbus, and CANbus.

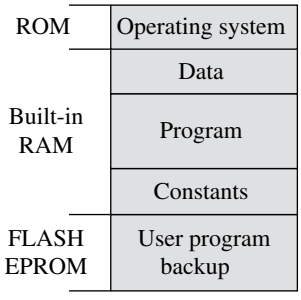


Figure 1.6 Typical memory board

Current developments are toward expanded use of Ethernet as a protocol for high-speed communications. Most manufacturers are offering solutions for this.

Power Supply
All PLCs must be connected to a power supply. Usually the power supply is an interchangeable module, but some smaller PLCs have the power supply as an integrated part of the processor and communications module. Even though the electronics in the PLC operate at 5 V, it is impractical to use this as an operating voltage. Most manufacturers therefore provide power supplies in several versions: 220 V AC, 120 V AC, and 24 V DC. If there is no access to power-line voltage, a variant with 24 V DC can be the solution. Usually there is access to 24 V out in the facility since this voltage level is standard for most sensors and transmitters. The advantage of being able to use a power supply that connects to the power line is that there is often a 24V output on the unit that can be used for powering sensors.

It is practical to have the power supply as a replaceable module. Then the PLC can be used in other physical locations in processing where there is not access to the same voltage level.

1.2.1 Inputs and Outputs

This is the contact between a PLC and the outside world. In a modular PLC, all inputs and outputs take place in blocks or modules that are designed to receive various types of signals and to transmit signals in various formats. There are input blocks for digital signals, analog signals, thermal elements and thermocouples, encoders, etc. There are also output blocks for digital and analog signals as well as blocks for special purposes.

Every input and output has a unique address that can be utilized in the program code. The I/O modules take care of electric isolation to protect the PLC and often have built-in functions for signal processing. This means that input and output signals can be connected directly without needing to use any extra electronic circuitry.

Chapter 2 deals with digital signals, sensors, and actuators, in Chapter 3 the theme is analog signals, and standard signal formats. On the next few pages, there follows only a general introduction to the inputs and outputs of a PLC.

Figure 1.7 on the next page shows a sketch of a process section that is controlled by a PLC. Various signal cables are drawn in the figure for the sake of illustration.

The process is equipped with three pressure transmitters and one flow transmitter. These constitute the input signals to the PLC in the figure.

Based on these measurements, among others, the PLC is programmed to control two pumps. The signals to the pumps thus constitute the output signals from the PLC.

The figure also shows an example of how a PLC rack can be assembled. From left to right, we see the following:

- The controller itself (CPU, memory, status lights, etc.) with built-in Ethernet (the unit in this case also has a built-in web server).
- A power supply (can supply sensors and other small equipment).
- I/O-modules (digital outputs, tele-modules, analog inputs and outputs).
- End modules that terminate the internal communications bus.

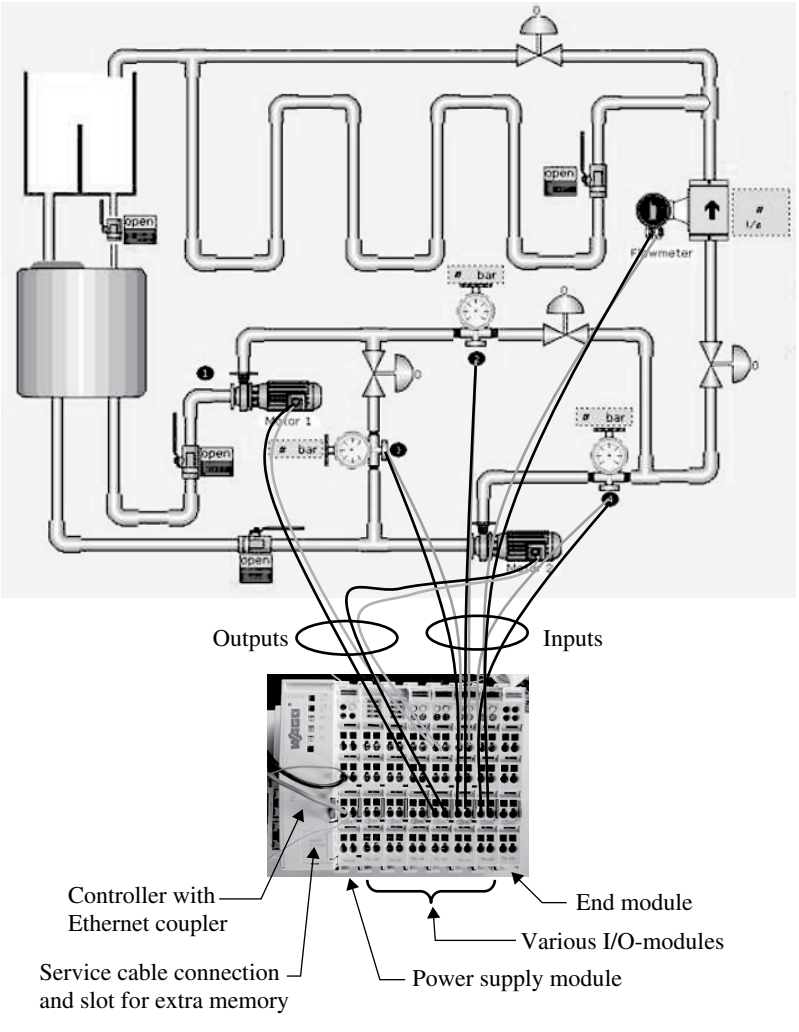


Figure 1.7 Illustration of a process section that is controlled by a PLC

1.2.1.1 Inputs

Digital input signals generally have a potential of 24 V DC, while the internal voltage in the PLC is 5 V. In order to protect the electronics in the PLC, the input modules generally use *optical couplers* (optical isolators). An optical coupler consists primarily of a light-emitting diode (LED) and a phototransistor.[8] Figure 1.8 illustrates the principle.

The diode and the transistor are electronically separated, but light can pass between them. When the signal at the input clamping circuit is logically high, the LED will emit an (infrared)

[8] A phototransistor is a type of bipolar transistor with transparent encapsulation. When the base–collector junction is sufficiently illuminated, the junction is biased and the transistor becomes conductive.

light. This light then triggers the transistor and results in a logically high signal in the electronic circuits in the module, where the potential is 5 V.

The gap between the LED and the phototransistor separates the external circuit from the internal electronics in the module. The internal electronics are thereby protected so that even though the PLC operates at 5 V internally, it is possible to use voltage levels at the input from 5 up to 230 V.

How much current an individual input can handle depends upon the engineering specifications of the input module in question. However, it is seldom that this is significant because most sensors have a low operating current.

Analog signals are fed into a PLC via analog-to-digital (A/D) converters. Converters are built into the analog input modules/cards. An analog signal is thus continually sampled and converted into binary values. Although in principle this requires only 1 bit for a low state input, often 16 bits are used to store values to an analog input.

1.2.1.2 Outputs

Standard digital output modules are often found in three different main types:

1. Relay outputs
2. Transistor outputs
3. Triac outputs

Relay Outputs

This type of output has the advantage that it can handle heavy loads and can be connected to both DC and AC loads at different voltages. When the CPU sets an output logically high, the associated output relay in the module in question closes and the external circuit to which the load is connected is completed (see Figure 1.9). The relay makes it possible for weak currents in the

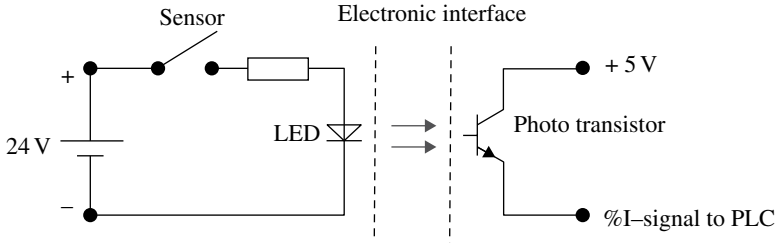


Figure 1.8 Principle of an optical coupler

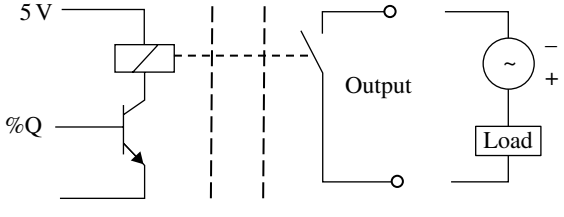


Figure 1.9 Principle of a relay output