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.
© 2015 John Wiley & Sons, Ltd. Published 2015 by John Wiley & Sons, Ltd.

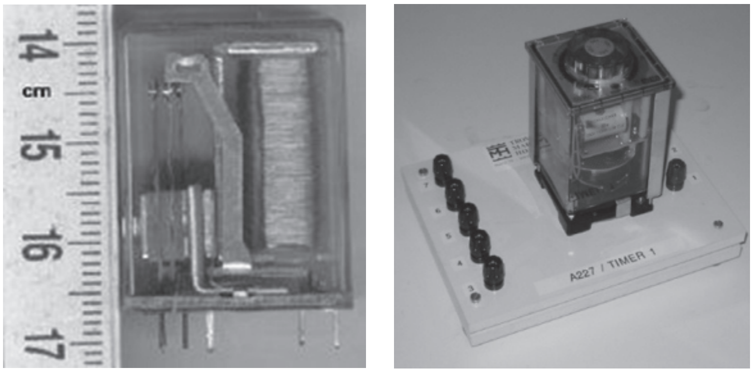

Figure 1.1 Example of a relay and a timer (mounted on a connector board)

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

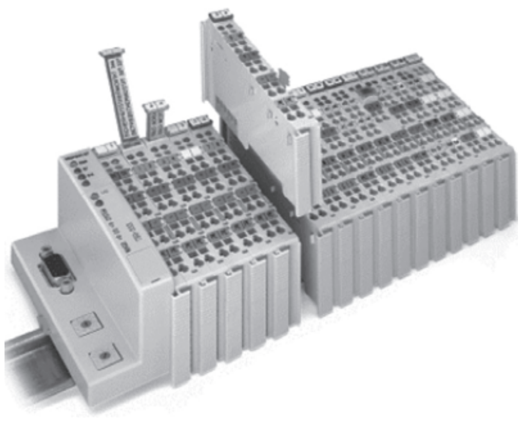

Figure 1.4 Newer generation PLC from Wago with Profibus coupler and I/O

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.)

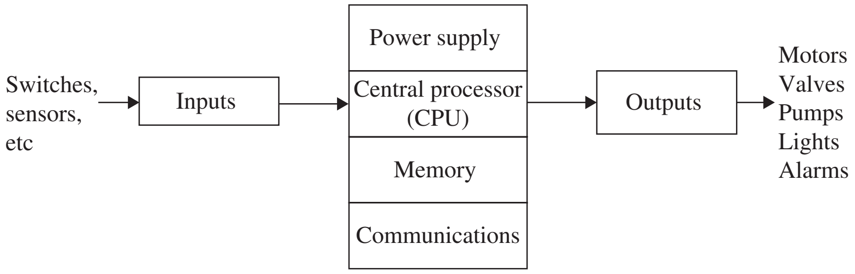

Figure 1.5 Block schematic representation of a PLC

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.

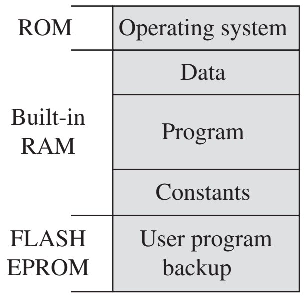

Figure 1.6 Typical memory board

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.

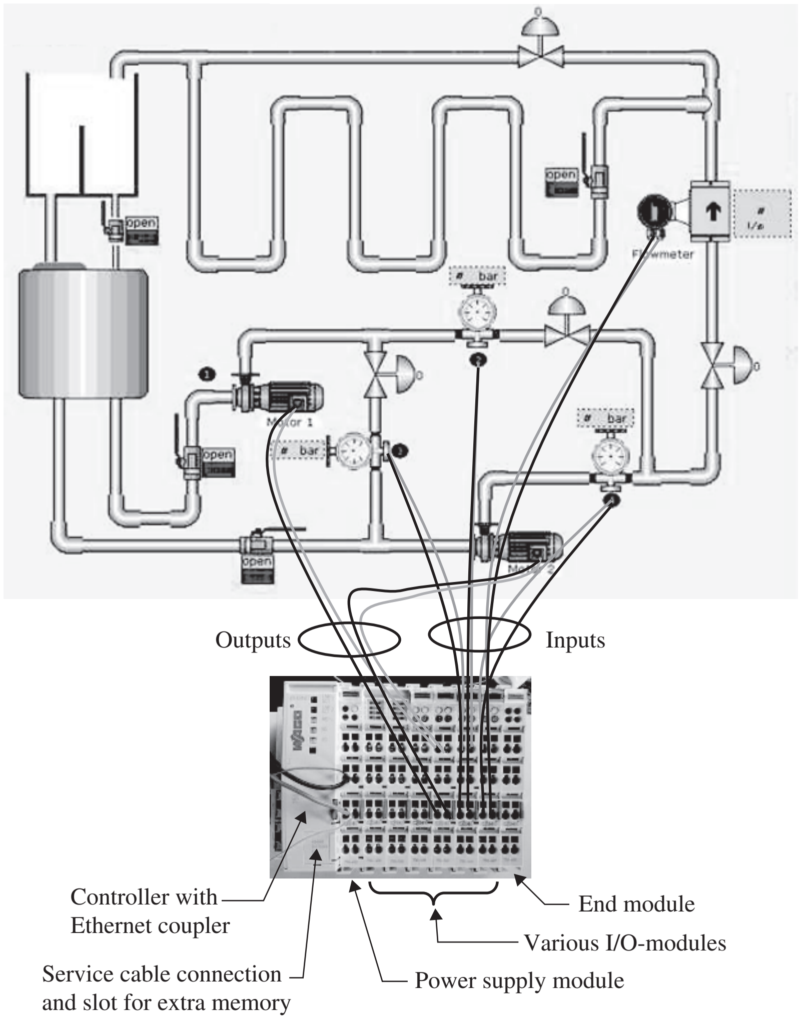

Figure 1.7 Illustration of a process section that is controlled by a PLC

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 5V.

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 5V internally, it is possible to use voltage levels at the input from 5 up to 230V.

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

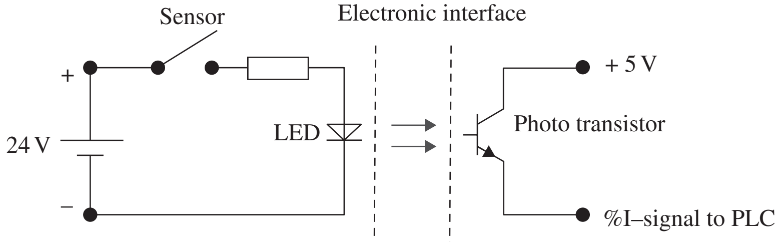

Figure 1.8 Principle of an optical coupler

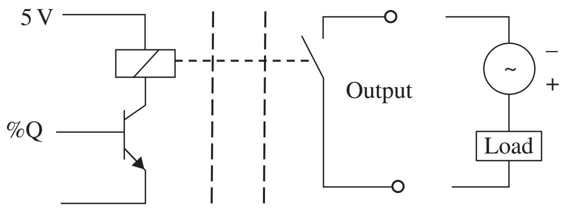

Figure 1.9 Principle of a relay output

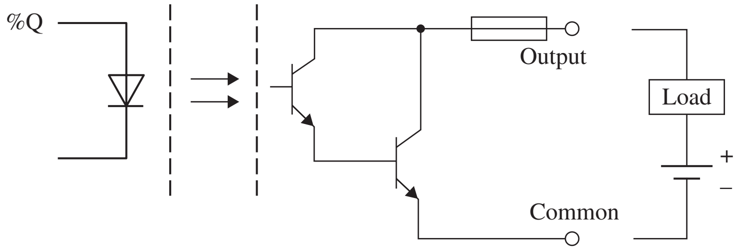

Figure 1.10 Principle of a transistor output

PLC to activate loads in which currents up to several Amperes can pass. In addition, the relay provides isolation between the PLC and external circuits.

Transistor Outputs

Compared to transistor outputs, relay outputs are relatively slow. Another advantage that makes transistor outputs popular is that they are cheaper than relay output modules. As the name indicates, such modules use transistors to complete the external circuits. It is this electronic switching that makes such modules significantly faster than relay modules, which switch with mechanical relays.[9] The disadvantage of transistor outputs is that, unless one uses an additional external relay, they can only be used for switching DC. They also cannot handle wrong polarity and are particularly sensitive to overload. Fuses with built-in electronics are therefore used in order to protect these outputs. Optical couplers are also used for electrical isolation (see Figure 1.10).

The operation of the circuit in the figure is as follows: When the output address is set logically high by the program, the phototransistor conducts. This triggers the next transistor and this completes the external circuit. Series connection of transistors (often called *Darlington* circuits) is used to increase the current capacity of the output stage.

You can read more about relay and transistor outputs in Section 2.7.3.

Triac Outputs

Triac outputs are not very common. They are used in situations that require fast switching of AC. Such outputs are also extremely sensitive to overcurrent and are protected with fuses.

1.3 PLC Operation

As discussed earlier, a PLC operates, in principle, in the same way as a PC. This means that a PLC must be programmed in order for it to perform its tasks. For a PLC, this usually means controlling and monitoring a *process*. This somewhat diffuse concept is used as a generalized word to describe a limited physical environment:

- A process can for instance be a room in a building with heating ovens, light, and ventilation. Then it can be the task of the PLC to control physical quantities such as temperature, CO_2 content, and humidity in the space.

[9] So-called solid-state relays are now available which switch electronically. I do not know to what extent these will be adopted by manufacturers in production of output modules with relay outputs.

- A process can also consist of a conveyor belt with goods, sensors, pneumatic pistons, and a labeling machine. The task of a PLC would then typically be to control labeling of goods, count goods, sort them, and group them.

The word *state* is often used to describe the various operating modes of a process. What states a process has depends on the nature of the process (process type). A process can have, for instance, the following states: *Fill tank—agitate—heat—Drain*.

The word *state* can also be used more specifically, for instance, for a temperature that has reached a certain value.

It is also the nature of the process that dictates what sensors and actuators are needed. Physical quantities that must be sensed can be distance, proximity, level, pressure, temperature, flow, velocity, rpm, etc. When the sensors that sense the physical quantities are connected to a PLC input module, the PLC has all of the information that is required in order to control and monitor the process. What is missing then are *actuators* and a *user program*.

- The function of actuators is to operate upon and change the states of the process. The type of process therefore determines what activators are required. These can be pumps, valves, switches, or motors.
- The user program employs available information from sensors, internally stored data, and the state of outputs to make decisions and calculate new output signals to the actuators.

Software

In almost all cases, there exist dedicated data tools for development of programs. Users can sit in the office and work on program code until they are sure that it is going to function in the PLC. Many of these programming tools have built-in functions for error detection and simulation, something that makes the job significantly simpler. When the user has finished programming, the PLC can be connected to the PC via a dedicated programming cable, and the program can be transferred from the PC to the PLC. When this has been done, the PC can be disconnected and the PLC is ready to perform its control tasks.

1.3.1 Process Knowledge

Before a PLC can be programmed, it is necessary to have good knowledge about the process (the part of the facility) that the PLC is going to control. Good understanding of the process is important in order to obtain good results, and sometimes it is necessary in order to get the control to function at all. This can be a time-consuming part of the job and often implies access to the understanding that operating personnel are familiar with. Remember that no one knows a process better than the people do whose daily job it is to make it work. Having said this, remember that operating personnel often have strong opinions about the process and how it should function and be controlled and that this is not necessarily the optimum way of doing things. It can be difficult to think in a new direction and to see other possibilities when things have been done in the same way for a long time.

Try to obtain available documentation such as engineering data, wiring diagrams, reporting forms, troubleshooting guides, maintenance SOPs, and the like.

I/O-Lists

An I/O-list is a basic part of choosing the correct PLC and the right accessories. The I/O list must therefore contain an overview of all of the required input and output signals. It is natural to group these by type, such as digital and analog. The analog can be further grouped according to standard signal formats, such as 4–20 mA, 0–10 V, type of temperature sensor, etc. Sometimes, the special signal formats impose extra requirements on hardware.

The digital signals can also be grouped. For instance, there are counter inputs and integral counter modules that measure pulses with higher frequencies than a normal digital input can tackle. An example of such a rapidly changing signal is a signal from an *encoder*,[10] which is a type of equipment that can be used for counting rpm and positional control.

Should the input blocks be of the *sink* or *source*[11] type? The type of actuator affects the selection of the proper type of output blocks: Should you use relay outputs or transistor outputs? How should the various actuators be supplied?

In addition to the flow of the process, desired performance and requirement for sensors, actuators, and I/O modules, it is also necessary to evaluate other aspects in and around the operation:

- Safety of personnel
- Any danger of fire or explosion
- Provision of alarms

Safety

This is a comprehensive theme that I am only going to touch on here. There are naturally laws and regulations concerning safety and I merely refer to those. At a minimum, you can try to describe what should happen in the event of a power failure, communications breakdown, activation of stops and emergency stops, etc.

Safety for humans and animals is something that must be taken extremely seriously. For instance, if a person is caught in a drive mechanism, then the actuator in question must be deactivated and an alarm sounded, at a minimum. In a power failure, you must decide whether the control should start again from where it stopped when the power failure happened, or whether it should be started anew. The same is true of a communications breakdown. There should be built-in monitoring of communications between the PLC and the HMI/SCADA[12]/operator panel so that the control is not cut off from manual override.

Almost all process facilities will also have one or more emergency stops, motor monitors, and the like. How to handle such events must also be described for later programming.

This also applies to switching over between automatic and manual control in a regulator, even though this is not so critical. This is often a natural portion of the process control.

[10] See Section 2.4.5.
[11] See Section 2.5.1 on page 48.
[12] Human–Machine Interface/Supervisory Control And Data Acquisition.

Provision of Alarms

Even though it is important to provide alarms in order to indicate when something has happened, it often turns out that an operator is drowning in alarms. Make a careful evaluation of which signals need to be specially monitored, what breakdowns must be handled by the PLC, and what the human–machine interface should handle, for instance.

The provision of alarms and safety will be significant considerations in how the program is organized (split up and grouped into critical and less-critical sections of programming).

Programming Situations

In order to simplify the programming, even at this stage it can be useful to formulate something about the flow in the process. It is easy to identify the flow when processes proceed from one state to the next in a particular order (e.g., filling—heating—agitating—draining).

Sometimes, several things happen simultaneously, or in parallel, and sometimes manual activities or process-controlled events decide what should be the next step in the sequence. It is often good to describe these sequences in words and/or by the use of a flow chart, state diagram, or the like. Which signals and events affect the transition from one step to the next in sequence? How should the various steps be performed? Which actuators should be activated and when should they be activated?

Sometimes, the process to be controlled is of such a nature that the transition from one state to another does not follow a predetermined pattern, but rather proceeds in a more random pattern. Such systems are referred to as *combinatoric* (despite the fact that the output states may well be a function of time as well). For such systems, it may be advantageous to use a slightly different procedure to develop the algorithms for control. There are methods that can be used to determine these algorithms in a systematic way and one of these will be described in Chapter 4.

1.3.2 Standard Operations

A process is in continual change. Even though the process has approached a stationary state, for instance, the fluid level in the tank has reached 80%, there will be disturbances in the form of pressure-drop in tubing, changed withdrawal of fluid and the like, that require that the PLC continually monitor the state of the input signals and correct the output signals. All PLCs in normal operation[13] therefore perform the same four operations[14] in a repeating cycle (Figure 1.11):

1. Internal processing
2. Read inputs
3. Program execution
4. Update outputs

[13] By this we mean a PLC that is in RUN mode. Other typical modes are *programming*, *stop*, *error*, and *diagnostic* (troubleshooting).

[14] This is a somewhat broad-brush treatment since the CPU is performing minor operations in addition to those mentioned. For instance, the CPU checks to what extent any of the inputs or outputs are forced to particular states by the user (the programmer). In addition, communications tasks are performed as mentioned under *Internal processing*.

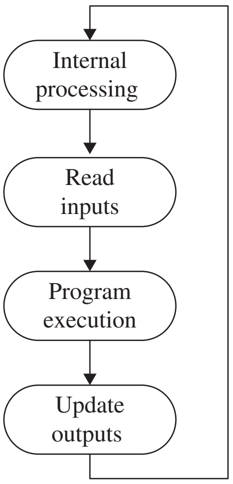

Figure 1.11 Operations that are performed in RUN mode

Internal Processing

A PLC always checks its own state before it performs user-related operations. If a response from hardware such as I/O or communications modules is lacking, the PLC gives notice of this by setting one or more *flags*. A flag is an internal Boolean address that can be checked by the user and/or an associated display or indicator light that gives a notification to the operator of an error state. For serious errors, normal operation is interrupted and the system goes over into an error state. Errors on individual inputs and individual outputs are also reported by setting flags. An example is when a system measures 0 current at an input configured for standard 4–20 mA signal or when an output is overloaded.

Software-related events that are performed in the internal operation are updating clocks, changing modes between run, stop, and program and setting *watchdog*[15] times to zero, among others.

Read Inputs

In this operation, input status is copied over to memory. How long this takes depends on the number of input modules and the number of inputs at each module that is in use. Analog inputs take significantly longer to read since this involves digitization of the analog values. In order not to reduce the update frequency of outputs, the PLC does not wait for new values to be available, but rather continues with the next operation in the cycle. This means that even though the physical values change continually, the same measured value can be used many times before an updated value is accessible in the memory.

Why copy the values to memory instead of reading the values where they are used in the program code? There are two reasons for this. First, it is quicker to read in all the values in the

[15] These are timers that the system uses to prevent the performance of an operation lasting longer than a determined maximum time. If the system is not ready to execute the user program within a half second, for instance, it may be that the program does not come out of a while-loop or the like. This results in an error state.

same operation. More importantly, this avoids possible problems when the same input address is used at several places in the program code. If the state of the input changes during the course of program execution, this can have serious consequences for the result of the program. A minor disadvantage of having the input values read simultaneously is that the system overlooks input pulses of a duration shorter than the total scan time (cycle time). See Section 1.3.2.

Program Execution

Execution of the program code takes place primarily in the order in which the code is written by the user. Smaller programs can be written in a coherent block of code, but larger programs require a different structuring of the code. The IEC 61131-3 standard defines guidelines for assignment of *priority*. Interruption routines and supervisory routines are assigned a higher priority than the main program. Activation of emergency stops is a common event that should cause interruption in the execution of the main program. Other reasons for changing the sequence of execution are conditional jumps or calls of subroutines, functions, and function blocks.

During the course of program execution, internal variables and output addresses are updated in the memory. The physical change in output values, however, is not changed until the final operation.

Update Outputs

In this last operation in the cycle, the output memory is read so that the state and values of the digital and analog outputs are updated. Later on in the book, we will see that the fact that the outputs are not updated until after the program code has been executed can be significant for how the program code is designed.

Note that a PLC in stop mode continues to perform in internal processing and read the inputs. The outputs are either set to a user-defined state/value (fallback) or maintained in their final state.

1.3.3 Cyclic, Freewheeling, or Event-Controlled Execution

The four operations described earlier are repeated continually, as discussed. Each cycle of operations is called a *scan,* and the time it takes the system to perform a scan is called the *scan time* or the *cycle time*. This is proportional to the size of the program, memory capacity, and type of processor. Usually this amounts to milliseconds and the cycle is repeated several times per second.

The effective scan time can vary from one scan to the next. In a scan, a new event may have suddenly appeared, one that activates a different part of the program code. In most types of PLCs, however, it is possible to configure where and how often a new scan is performed. Control of the manner in which the program is executed is achieved by associating the program to a *task*.[16] The three common modes in which a task is executed are as follows:

1. Cyclic
2. Freewheeling
3. Event-controlled

[16] The concept of *task* will be further described in Section 5.3.

Cyclic execution is based on having a fixed interval between the start of each scan. This interval must naturally be set long enough so that the execution of a single scan does not exceed the interval time. Such a control of execution speed can conveniently be used in the program code for counting and timing, for instance. A program block that includes a regulator is an example of code that should (must) be executed at fixed intervals.

With *freewheeling* execution, a new scan begins as soon as the previous one has completed. Because the code can contain many event-controlled events, the scan time can vary somewhat from one scan to the next. This is the fastest way of running a program.

Event-controlled execution is based on having the task (with associated program) be executed only if a (Boolean) condition is fulfilled. This can be useful in a program that normally is not to be performed and that is to be triggered by a particular event. An example of such a program could be emergency stop routines, startup routines and other extraordinary events.

One can also control how often the CPU scans a particular program unit by assigning a *priority* to the task(s). Tasks with higher priority will be monitored and scanned more often. Such tasks will typically contain important program units that manage critical events such as emergency stops or alarms.

Such control of executing and prioritizing program code makes it possible to build up multiapplications and/or a hierarchic structure of program units.

1.4 Test Problems

Problem 1.1

(a) What type of control was replaced by the PLC?
(b) What advantages are achieved by the use of a PLC?
(c) Name some differences between a newer PLC and the PLC from the 1970s.
(d) Most PLCs have a battery. Why do you think they have one?
(e) What is a CPU?
(f) Select a random PLC from a randomly selected manufacturer and do an Internet search for the various modules for the PLC. Make a list of at least 15 different modules that can be installed in the rack for the selected PLC.

Problem 1.2

(a) Name some advantages and disadvantages of transistor outputs and relay outputs.
(b) What is the purpose of an optical coupler and what two basic components does it contain?
(c) List the operations that a PLC performs during the course of a scan.
(d) What is the reason that a PLC checks the status of all inputs before each scan before the program code is executed and not during execution?
(e) What are cyclic execution and freewheeling execution?