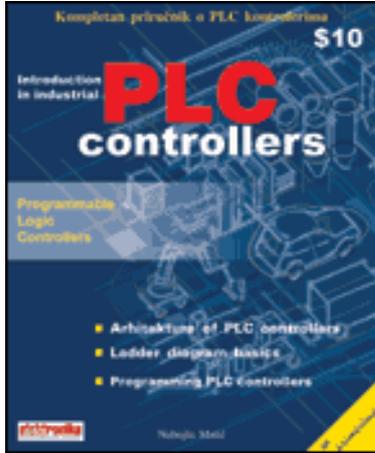


Introduction to PLC controllers, *for begginers too!*

Author: Nebojsa Matic



Paperback - 252 pages (June 10, 2001)

Dimensions (in inches): 0.62 x 9.13 x 7.28

Content: What are they? How to connect a simple sensor. How to program in ladder diagram. In this book you will find answers to these questions and more...

C o n t e n t s

Chapter I [Operating system](#)

[Introduction](#)

[1.1 Conventional control panel](#)

[1.2 Control panel with a PLC controller](#)

[1.3 Systematic approach to designing a process control system](#)

Chapter II [Introduction to PLC controllers](#)

[Introduction](#)

[2.1 First programmed controllers](#)

[2.2 PLC controller parts](#)

[2.3 Central processing Unit –CPU](#)

[2.4 Memory](#)

[2.5 PLC controller programming](#)

[2.6 Power supply](#)

[2.7 Input to PLC controller](#)

[2.8 Input adjustment interface](#)

[2.9 PLC controller output](#)

[2.10 Output adjustment interface](#)

[2.11 Extension lines](#)

Chapter III [Connecting sensors and output devices](#)

[3.1 Sinking-Sourcing concept](#)

[3.2 Input lines](#)

[3.3 Output lines](#)

Chapter IV [Architecture of a specific PLC controller](#)

[Introduction](#)

- [4.1 Why OMRON?](#)
- [4.2 CPM1A PLC controller](#)
- [4.3 PLC controller output lines](#)
- [4.4 PLC controller input lines](#)
- [4.5 Memory map for CPM1A PLC controller](#)

Chapter V [Relay diagram](#)

[Introduction](#)

- [5.1 Relay diagram](#)
- [5.2 Normally open and Normally closed contacts](#)
- [5.3 Short example](#)

Chapter VI [SYSWIN, program for PLC controller programming](#)

[Introduction](#)

- [6.1 How to connect a PLC controller to a PC](#)
- [6.2 SYSWIN program installation](#)
- [6.3 Writing a first program](#)
- [6.4 Saving a project](#)
- [6.5 Program transfer to PLC controller](#)
- [6.6 Checkup of program function](#)
- [6.7 Meaning of tool-bar icons](#)
- [6.8 PLC controller function modes](#)
- [6.9 RUN mode](#)
- [6.10 MONITOR mode](#)
- [6.11 PROGRAM-STOP mode](#)
- [6.12 Program execution and monitoring](#)
- [6.13 Program checkup during monitoring](#)
- [6.14 Graphic display of dimension changes in a program](#)

Chapter VII [Examples](#)

[Introduction](#)

- [7.1 Self-maintenance](#)
- [7.2 Making large time intervals](#)
- [7.3 Counter over 9999](#)
- [7.4 Delays of ON and OFF status](#)
- [7.5 Alternate ON-OFF output](#)
- [7.6 Automation of parking garage for 100 vehicles](#)
- [7.7 Operating a charge and discharge process](#)
- [7.8 Automation of product packaging](#)
- [7.9 Automation a storage door](#)

Addition A [Extending a number of U/I lines](#)

[Introduction](#)

- [A.1 Differences and similarities](#)
- [A.2 Marking a PLC controller](#)
- [A.3 Specific case](#)

Addition B Detailed memory map for PLC controller

Introduction

- B.1 General explanation of memory regions
- B.2 IR memory region
- B.3 SR memory region
- B.4 AR memory region
- B.5 PC memory region

Addition C PLC diagnostics

Introduction

- C.1 Diagnostic functions of a PLC controller
- C.2 Errors
- C.3 Fatal errors
- C.4 User defined errors
- C.5 Failure Alarm –FAL(06)
- C.6 Severe Failure Alarm –FALS(07)
- C.7 MESSAGE –MSG(46)
- C.8 Syntax errors
- C.9 Algorithm for finding errors in a program

Addition D Numeric systems

Introduction

- D.1 Decimal numeric system
- D.2 Binary numeric system
- D.3 Hexadecimal numeric system

Addition E Detailed set of instructions

Introduction

- E.1 Order of input lines
- E.2 Order of output lines
- E.3 Order of operating instructions
- E.4 Timer/counter instructions
- E.5 Instructions for data comparison
- E.6 Instructions for data transfer
- E.7 Transfer instructions
- E.8 Instructions for reduction/enlargement
- E.9 Instructions for BCD / binary calculations
- E.10 Instructions for data conversion
- E.11 Logic instructions
- E.12 Special instructions for calculations
- E.13 Subprogram instructions
- E.14 Instructions for operating interrupts
- E.15 U/I instructions
- E.16 Display instructions
- E.17 Instructions for control of fast counter
- E.18 Diagnostic functions
- E.19 Special system instructions

Send us a comment about a book

Subject :

Name :

State :

E-mail :

Your message:

CHAPTER 1 Process control system

[Introduction](#)

[1.1 Conventional control panel](#)

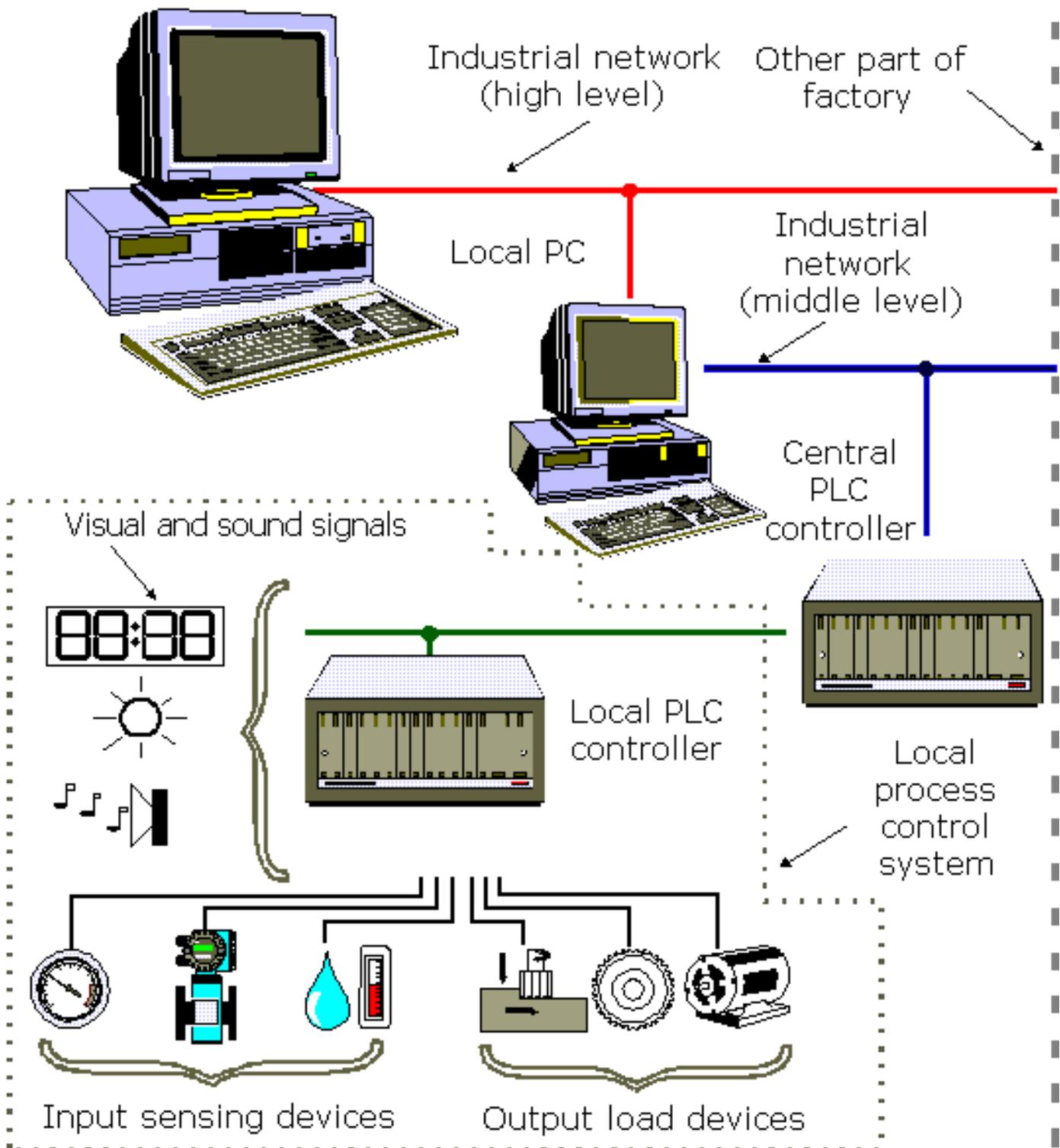
[1.2 Control panel with a PLC controller](#)

[1.3 Systematic approach to designing a process control system](#)

Introduction

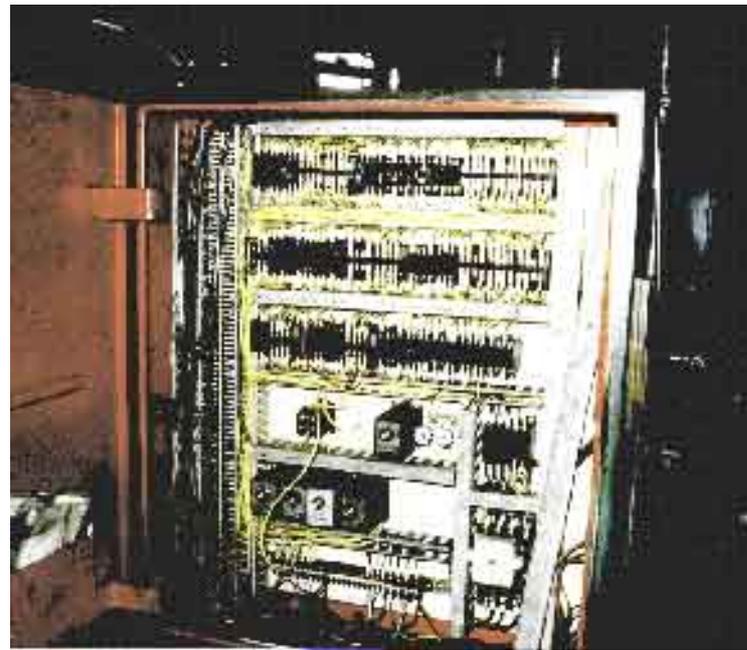
Generally speaking, process control system is made up of a group of electronic devices and equipment that provide stability, accuracy and eliminate harmful transition statuses in production processes. Operating system can have different form and implementation, from energy supply units to machines. As a result of fast progress in technology, many complex operational tasks have been solved by connecting programmable logic controllers and possibly a central computer. Beside connections with instruments like operating panels, motors, sensors, switches, valves and such, possibilities for communication among instruments are so great that they allow high level of exploitation and process coordination, as well as greater flexibility in realizing an process control system. Each component of an process control system plays an important role, regardless of its size. For example, without a sensor, PLC wouldn't know what exactly goes on in the process. In automated system, PLC controller is usually the central part of an process control system. With execution of a program stored in program memory, PLC continuously monitors status of the system through signals from input devices. Based on the logic implemented in the program, PLC determines which actions need to be executed with output instruments. To run more complex processes it is possible to connect more PLC controllers to a central computer. A real system could look like the one pictured below:

SCADA system



1.1 Conventional control panel

At the outset of industrial revolution, especially during sixties and seventies, relays were used to operate automated machines, and these were interconnected using wires inside the control panel. In some cases a control panel covered an entire wall. To discover an error in the system much time was needed especially with more complex process control systems. On top of everything, a lifetime of relay contacts was limited, so some relays had to be replaced. If replacement was required, machine had to be stopped and production too. Also, it could happen that there was not enough room for necessary changes. control panel was used only for one particular process, and it wasn't easy to adapt to the requirements of a new system. As far as maintenance, electricians had to be very skillful in finding errors. In short, conventional control panels proved to be very inflexible. Typical example of conventional control panel is given in the following picture.



In this photo you can notice a large number of electrical wires, time relays, timers and other elements of automation typical for that period. Pictured control panel is not one of the more “complicated” ones, so you can imagine what complex ones looked like.

Most frequently mentioned disadvantages of a classic control panel are:

- Too much work required in connecting wires
- Difficulty with changes or replacements
- Difficulty in finding errors; requiring skillful work force
- When a problem occurs, hold-up time is indefinite, usually long.

1.2 Control panel with a PLC controller

With invention of programmable controllers, much has changed in how an process control system is designed. Many advantages appeared. Typical example of control panel with a PLC controller is given in the following picture.



Advantages of control panel that is based on a PLC controller can be presented in few basic points:

- 1.** Compared to a conventional process control system, number of wires needed for connections is reduced by 80%
- 2.** Consumption is greatly reduced because a PLC consumes less than a bunch of relays
- 3.** Diagnostic functions of a PLC controller allow for fast and easy error detection.
- 4.** Change in operating sequence or application of a PLC controller to a different operating process can easily be accomplished by replacing a program through a console or using a PC software (not requiring changes in wiring, unless addition of some input or output device is required).
- 5.** Needs fewer spare parts
- 6.** It is much cheaper compared to a conventional system, especially in cases where a large number of I/O instruments are needed and when operational functions are complex.
- 7.** Reliability of a PLC is greater than that of an electro-mechanical relay or a timer.

1.3 Systematic approach in designing an process control system

First, you need to select an instrument or a system that you wish to control. Automated system can be a machine or a process and can also be called an process control system. Function of an process control system is constantly watched by input devices (sensors) that give signals to a PLC controller. In response to this, PLC controller sends a signal to external output devices (operative instruments) that actually control how system functions in an assigned manner (for simplification it is recommended that you draw a block diagram of operations' flow).

Secondly, you need to specify all input and output instruments that will be connected to a PLC controller. Input devices are various switches, sensors and such. Output devices can be solenoids, electromagnetic valves, motors, relays, magnetic starters as well as instruments for sound and light signalization.

Following an identification of all input and output instruments, corresponding designations are assigned to input and output lines of a PLC controller. Allotment of these designations is in fact an allocation of inputs and outputs on a PLC controller which correspond to inputs and outputs of a system being designed.

Third, make a ladder diagram for a program by following the sequence of operations that was determined in the first step.

Finally, program is entered into the PLC controller memory. When finished with programming, checkup is done for any existing errors in a program code (using functions for diagnostics) and, if possible, an entire operation is simulated. Before this system is started, you need to check once again whether all input and output instruments are connected to correct inputs or outputs. By bringing supply in, system starts working.

CHAPTER 2 Introduction to PLC controllers

[Introduction](#)

[2.1 First programmed controllers](#)

[2.2 PLC controller parts](#)

[2.3 Central Processing unit -CPU](#)

[2.4 Memory](#)

[2.5 How to program a PLC controller](#)

[2.6 Power supply](#)

[2.7 Input to a PLC controller](#)

[2.8 Input adjustable interface](#)

[2.9 Output from a PLC controller](#)

[2.10 Output adjustable interface](#)

[2.11 Extension lines](#)

Introduction

Industry has begun to recognize the need for quality improvement and increase in productivity in the sixties and seventies. Flexibility also became a major concern (ability to change a process quickly became very important in order to satisfy consumer needs).

Try to imagine automated industrial production line in the sixties and seventies. There was always a huge electrical board for system controls, and not infrequently it covered an entire wall! Within this board there was a great number of interconnected electromechanical relays to make the whole system work. By word "connected" it was understood that electrician had to connect all relays manually using wires! An engineer would design logic for a system, and electricians would receive a schematic outline of logic that they had to implement with relays. These relay schemas often contained hundreds of relays. The plan that electrician was given was called "ladder schematic". Ladder displayed all switches, sensors, motors, valves, relays, etc. found in the system. Electrician's job was to connect them all together. One of the problems with this type of control was that it was based on mechanical relays. Mechanical instruments were usually the weakest connection in the system due to their moveable parts that could wear out. If one relay stopped working, electrician would have to examine an entire system (system would be out until a cause of the problem was found and corrected).

The other problem with this type of control was in the system's break period when a system had to be turned off, so connections could be made on the electrical board. If a firm decided to change the order of operations (make even a small change), it would turn out to be a major expense and a loss of production time until a system was functional again.

It's not hard to imagine an engineer who makes a few small errors during his project. It is also conceivable that electrician has made a few mistakes in connecting the system. Finally, you can also imagine having a few bad components. The only way to see if everything is all right is to run the system. As systems are usually not perfect with a first try, finding errors was an arduous process. You should also keep in mind that a product could not be made during these corrections and changes in connections. System had to be literally disabled before changes were to be performed. That meant that the entire production staff in that line of production was out of work until the system was fixed up again. Only when electrician was done finding errors and repairing,, the system was ready for production. Expenditures for this kind of work were too great even for well-to-do companies.

2.1 First programmable controllers

"General Motors" is among the first who recognized a need to replace the system's "wired" control board. Increased competition forced auto-makers to improve production quality and productivity. Flexibility and fast and easy change of automated lines of production became crucial! General Motors' idea was to use for system logic one of the microcomputers (these microcomputers were as far as their strength beneath today's eight-bit microcontrollers) instead of wired relays. Computer could take place of huge, expensive, inflexible wired control boards. If changes were needed in system logic or in order of operations, program in a microcomputer could be changed instead of rewiring of relays. Imagine only what elimination of the entire period needed for changes in wiring meant then. Today, such thinking is but common, then it was revolutionary!

Everything was well thought out, but then a new problem came up of how to make electricians accept and use a new device. Systems are often quite complex and require complex programming. It was out of question to ask electricians to learn and use computer language in addition to other job duties. General Motors Hydromatic Division of this big company recognized a need and wrote out project criteria for first programmable logic controller (there were companies which sold instruments that performed industrial control, but those were simple sequential controllers ù not PLC controllers as we know them today). Specifications required that a new device be based on electronic instead of mechanical parts, to have flexibility of a computer, to function in industrial environment (vibrations, heat, dust, etc.) and have a capability of being reprogrammed and used for other tasks. The last criteria was also the most important, and a new device had to be programmed easily and maintained by electricians and technicians. When the specification was done, General Motors looked for interested companies, and encouraged them to develop a device that would meet the specifications for this project.

"Gould Modicon" developed a first device which met these specifications. The key to success with a new device was that for its programming you didn't have to learn a new programming language. It was programmed so that same language ùa ladder diagram, already known to technicians was used. Electricians and technicians could very easily understand these new devices because the logic looked similar to old logic that they were used to working with. Thus they didn't have to learn a new programming language which (obviously) proved to be a good move. PLC controllers were initially called PC controllers (programmable controllers). This caused a small confusion when Personal Computers appeared. To avoid confusion, a designation PC was left to computers, and programmable controllers became programmable logic controllers. First PLC controllers were simple devices. They connected inputs such as switches, digital sensors, etc., and based on internal logic they turned output devices on or off. When they first came up, they were not quite suitable for complicated controls such as temperature, position, pressure, etc. However, throughout years, makers of PLC controllers added numerous features and improvements. Today's PLC controller can handle highly complex tasks such as position control, various regulations and other complex applications. The speed of work and easiness of programming were also improved. Also, modules for special purposes were developed, like communication modules for connecting several PLC controllers to the net. Today it is difficult to imagine a task that could not be handled by a PLC.

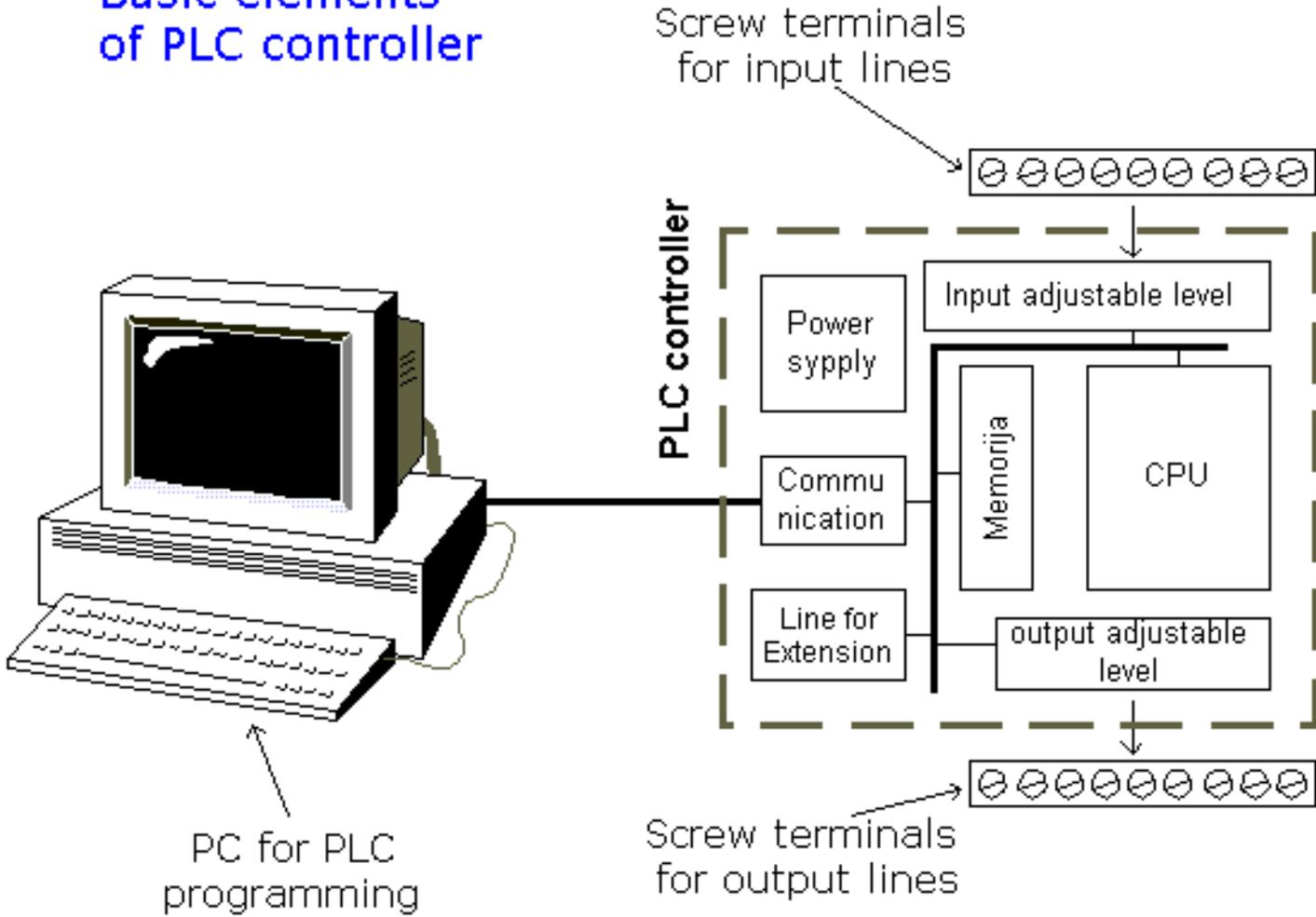
2.2 PLC controller components

PLC is actually an industrial microcontroller system (in more recent times we meet processors instead of microcontrollers) where you have hardware and software specifically adapted to industrial environment. Block schema with typical components which PLC consists of is found in the following picture. Special attention needs to be given to input and output, because in these blocks you find protection needed in isolating a CPU blocks from damaging influences that industrial environment can bring to a CPU via input lines. Program unit is usually a computer used for writing a program (often in ladder diagram).

2.3 Central Processing Unit - CPU

Central Processing Unit (CPU) is the brain of a PLC controller. CPU itself is usually one of the microcontrollers. Aforetime these were 8-bit microcontrollers such as 8051, and now these are 16- and 32-bit microcontrollers. Unspoken rule is that you'll find mostly Hitachi and Fujicu microcontrollers in PLC controllers by Japanese makers, Siemens in European controllers, and Motorola microcontrollers in American ones. CPU also takes care of communication, interconnectedness among other parts of PLC controller, program execution, memory operation, overseeing input and setting up of an output. PLC controllers have complex routines for memory checkup in order to ensure that PLC memory was not damaged (memory checkup is done for safety reasons). Generally speaking, CPU unit makes a great number of check-ups of the PLC controller itself so eventual errors would be discovered early. You can simply look at any PLC controller and see that there are several indicators in the form of light diodes for error signalization.

Basic elements of PLC controller



2.4 Memory

System memory (today mostly implemented in FLASH technology) is used by a PLC for an process control system. Aside from this operating system it also contains a user program translated from a ladder diagram to a binary form. FLASH memory contents can be changed only in case where user program is being changed. PLC controllers were used earlier instead of FLASH memory and have had EPROM memory instead of FLASH memory which had to be erased with UV lamp and programmed on programmers. With the use of FLASH technology this process was greatly shortened. Reprogramming a program memory is done through a serial cable in a program for application development.

User memory is divided into blocks having special functions. Some parts of a memory are used for

storing input and output status. The real status of an input is stored either as "1" or as "0" in a specific memory bit. Each input or output has one corresponding bit in memory. Other parts of memory are used to store variable contents for variables used in user program. For example, timer value, or counter value would be stored in this part of the memory.

2.5 Programming a PLC controller.

PLC controller can be reprogrammed through a computer (usual way), but also through manual programmers (consoles). This practically means that each PLC controller can be programmed through a computer if you have the software needed for programming. Today's transmission computers are ideal for reprogramming a PLC controller in factory itself. This is of great importance to industry. Once the system is corrected, it is also important to read the right program into a PLC again. It is also good to check from time to time whether program in a PLC has not changed. This helps to avoid hazardous situations in factory rooms (some automakers have established communication networks which regularly check programs in PLC controllers to ensure execution only of good programs).

Almost every program for programming a PLC controller possesses various useful options such as: forced switching on and off of the system inputs/outputs (I/O lines), program follow up in real time as well as documenting a diagram. This documenting is necessary to understand and define failures and malfunctions. Programmer can add remarks, names of input or output devices, and comments that can be useful when finding errors, or with system maintenance. Adding comments and remarks enables any technician (and not just a person who developed the system) to understand a ladder diagram right away. Comments and remarks can even quote precisely part numbers if replacements would be needed. This would speed up a repair of any problems that come up due to bad parts. The old way was such that a person who developed a system had protection on the program, so nobody aside from this person could understand how it was done. Correctly documented ladder diagram allows any technician to understand thoroughly how system functions.

2.6. Power supply

Electrical supply is used in bringing electrical energy to central processing unit. Most PLC controllers work either at 24 VDC or 220 VAC. On some PLC controllers you'll find electrical supply as a separate module. Those are usually bigger PLC controllers, while small and medium series already contain the supply module. User has to determine how much current to take from I/O module to ensure that electrical supply provides appropriate amount of current. Different types of modules use different amounts of electrical current.

This electrical supply is usually not used to start external inputs or outputs. User has to provide separate supplies in starting PLC controller inputs or outputs because then you can ensure so called "pure" supply for the PLC controller. With pure supply we mean supply where industrial environment can not affect it damagingly. Some of the smaller PLC controllers supply their inputs with voltage from a small supply source already incorporated into a PLC.

2.7 PLC controller inputs

Intelligence of an automated system depends largely on the ability of a PLC controller to read signals from different types of sensors and input devices. Keys, keyboards and by functional switches are a basis for man versus machine relationship. On the other hand, in order to detect a working piece, view a mechanism in motion, check pressure or fluid level you need specific automatic devices such as proximity sensors, marginal switches, photoelectric sensors, level sensors, etc. Thus, input signals can be logical (on/off) or analogue. Smaller PLC controllers usually have only digital input lines while larger also accept analogue inputs through special units attached to PLC controller. One of the most frequent analogue signals are a current signal of 4 to 20 mA and millivolt voltage signal generated by various sensors. Sensors are usually used as inputs for PLCs. You can obtain sensors for different purposes. They can sense presence of some parts, measure temperature, pressure, or some other physical dimension, etc. (ex. inductive

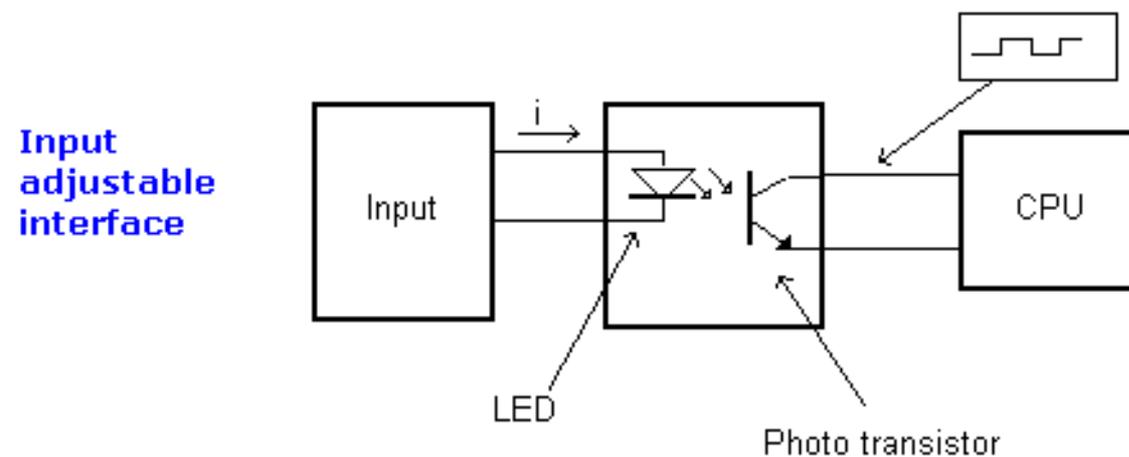
sensors can register metal objects).

Other devices also can serve as inputs to PLC controller. Intelligent devices such as robots, video systems, etc. often are capable of sending signals to PLC controller input modules (robot, for instance, can send a signal to PLC controller input as information when it has finished moving an object from one place to the other.)

2.8 Input adjustment interface

Adjustment interface also called an interface is placed between input lines and a CPU unit. The purpose of adjustment interface to protect a CPU from disproportionate signals from an outside world. Input adjustment module turns a level of real logic to a level that suits CPU unit (ex. input from a sensor which works on 24 VDC must be converted to a signal of 5 VDC in order for a CPU to be able to process it). This is typically done through opto-isolation, and this function you can view in the following picture.

Opto-isolation means that there is no electrical connection between external world and CPU unit. They are "optically" separated, or in other words, signal is transmitted through light. The way this works is simple. External device brings a signal which turns LED on, whose light in turn incites photo transistor which in turn starts conducting, and a CPU sees this as logic zero (supply between collector and transmitter falls under 1V). When input signal stops LED diode turns off, transistor stops conducting, collector voltage increases, and CPU receives logic 1 as information.



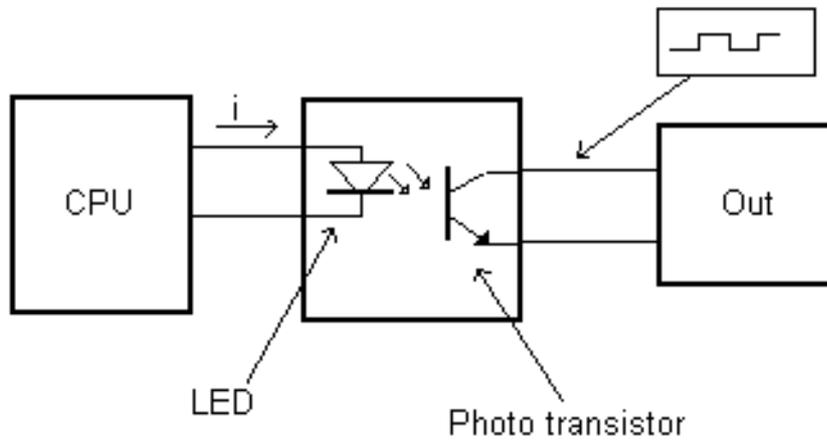
2.9 PLC controller output

Automated system is incomplete if it is not connected with some output devices. Some of the most frequently used devices are motors, solenoids, relays, indicators, sound signalization and similar. By starting a motor, or a relay, PLC can manage or control a simple system such as system for sorting products all the way up to complex systems such as service system for positioning head of CNC machine. Output can be of analogue or digital type. Digital output signal works as a switch; it connects and disconnects line. Analogue output is used to generate the analogue signal (ex. motor whose speed is controlled by a voltage that corresponds to a desired speed).

2.10 Output adjustment interface

Output interface is similar to input interface. CPU brings a signal to LED diode and turns it on. Light incites a photo transistor which begins to conduct electricity, and thus the voltage between collector and emitter falls to 0.7V, and a device attached to this output sees this as a logic zero. Inversely it means that a signal at the output exists and is interpreted as logic one. Photo transistor is not directly connected to a PLC controller output. Between photo transistor and an output usually there is a relay or a stronger transistor capable of interrupting stronger signals.

Output adjustable interface



2.11 Extension lines

Every PLC controller has a limited number of input/output lines. If needed this number can be increased through certain additional modules by system extension through extension lines. Each module can contain extension both of input and output lines. Also, extension modules can have inputs and outputs of a different nature from those on the PLC controller (ex. in case relay outputs are on a controller, transistor outputs can be on an extension module).

CHAPTER 3 Connecting sensors and execution devices

[Introduction](#)

[3.1 Sinking-sourcing concept](#)

[3.2 Input lines](#)

[3.3 Output lines](#)

Introduction

Connecting external devices to a PLC controller regardless whether they are input or output is a special subject matter for industry. If it stands alone, PLC controller itself is nothing. In order to function it needs sensors to obtain information from environment, and it also needs execution devices so it could turn the programmed change into a reality. Similar concept is seen in how human being functions. Having a brain is simply not enough. Humans achieve full activity only with processing of information from a sensor (eyes, ears, touch, smell) and by taking action through hands, legs or some tools. Unlike human being who receives his sensors automatically, when dealing with controllers, sensors have to be subsequently connected to a PLC. How to connect input and output parts is the topic of this chapter.

3.1 Sinking-Sourcing Concept

PLC has input and output lines through which it is connected to a system it directs. Input can be keys, switches, sensors while outputs are led to different devices from simple signalization lights to complex communication modules.

This is a very important part of the story about PLC controllers because it directly influences what can be connected and how it can be connected to controller inputs or outputs. Two terms most frequently mentioned when discussing connections to inputs or outputs are "*sinking*" and "*sourcing*". These two concepts are very important in connecting a PLC correctly with external environment. The most brief definition of these two concepts would be:

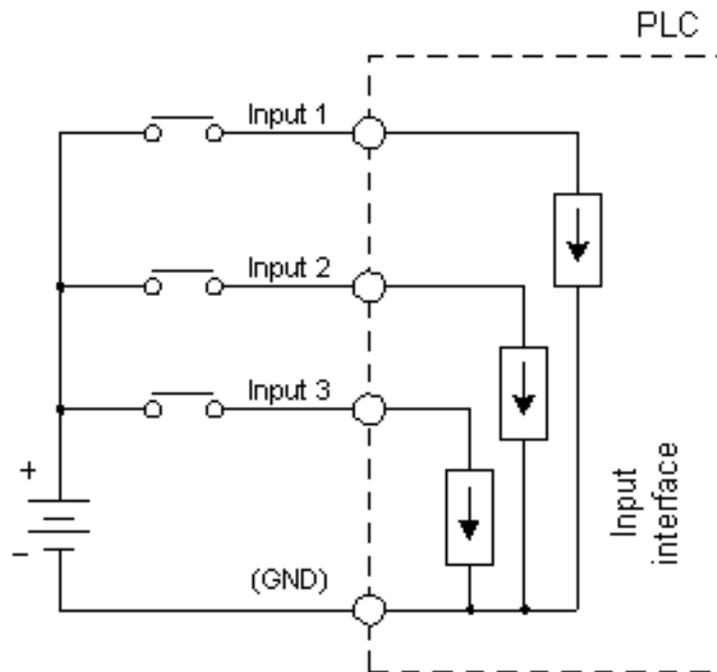
SINKING = Common GND line (-)

SOURCING = Common VCC line (+)

First thing that catches one's eye are "+" and "-" supply, DC supply. Inputs and outputs which are either sinking or sourcing can conduct electricity only in one direction, so they are only supplied with direct current.

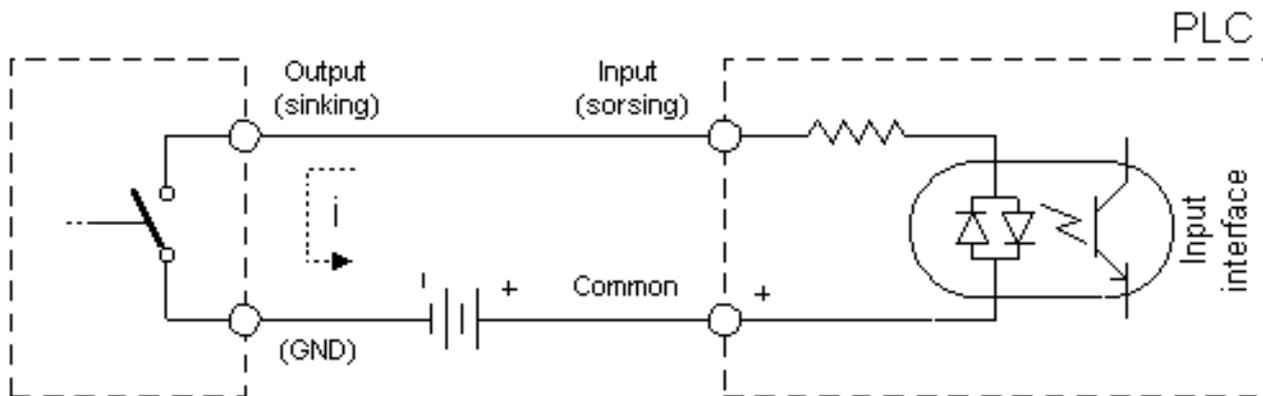
According to what we've said thus far, each input or output has its own return line, so 5 inputs would need 10 screw terminals on PLC controller housing. Instead, we use a system of connecting several inputs to one return line as in the following picture. These common lines are usually marked "COMM" on the PLC controller housing.

Connecting several inputs to a common line

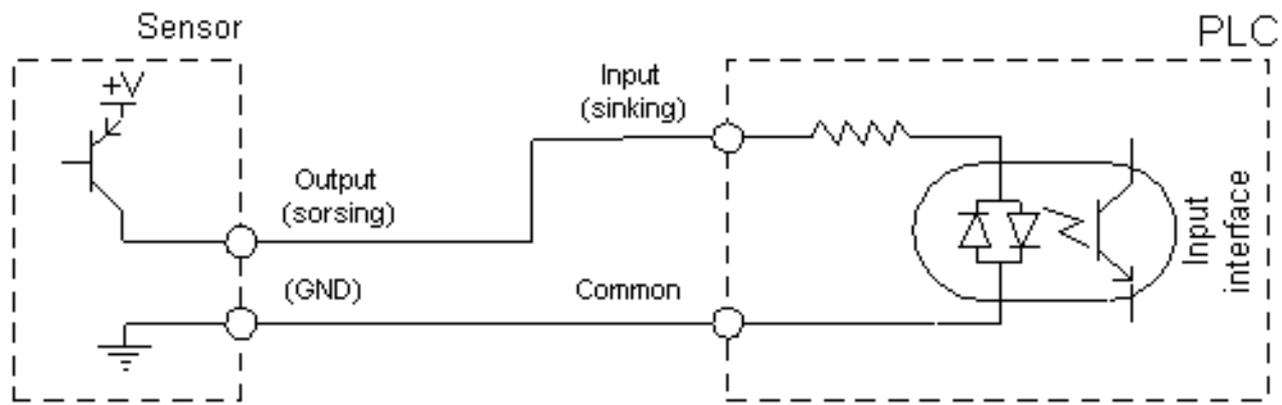


3.2 Input lines

Explanation of PLC controller input and output lines has up to now been given only theoretically. In order to apply this knowledge, we need to make it a little more specific. Example can be connection of external device such as proximity sensor. Sensor outputs can be different depending on a sensor itself and also on a particular application. Following pictures display some examples of sensor outputs and their connection with a PLC controller. Sensor output actually marks the size of a signal given by a sensor at its output when this sensor is active. In one case this is +V (supply voltage, usually 12 or 24V) and in other case a GND (0V). Another thing worth mentioning is that sinking-sourcing and sourcing - sinking pairing is always used, and not sourcing-sourcing or sinking-sinking pairing.

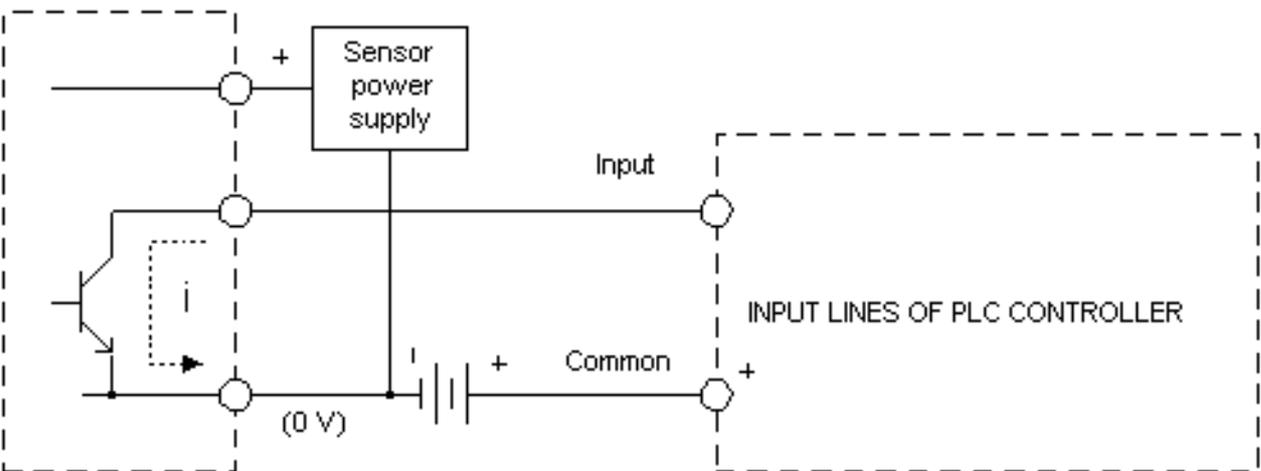


Connecting sensors with sinking output to a PLC controller sourcing input

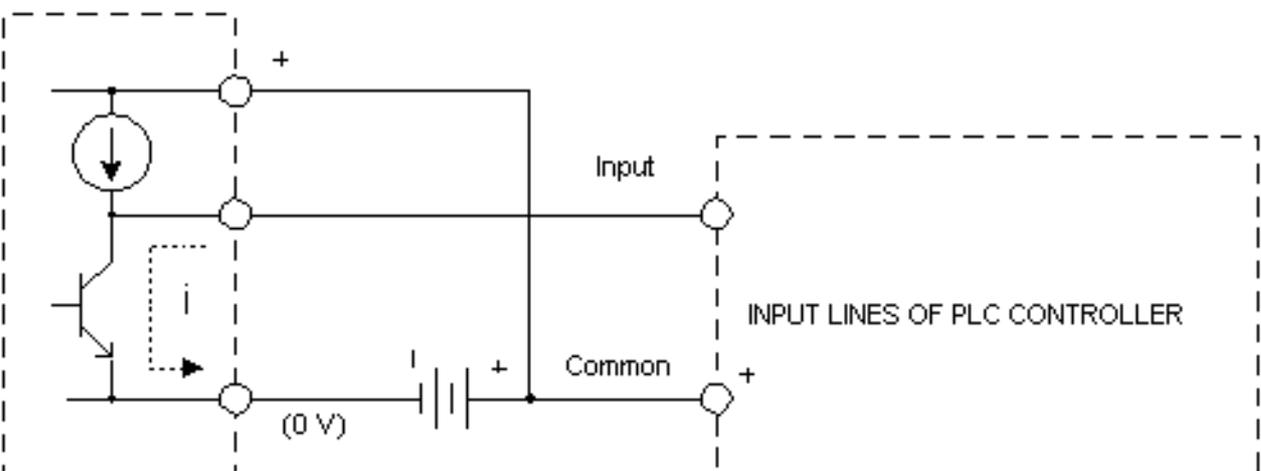


Connecting sensors with a sourcing output to a PLC controller sinking input

If we were to make type of connection more specific, we'd get combinations as in following pictures (for more specific connection schemas we need to know the exact sensor model and a PLC controller model).

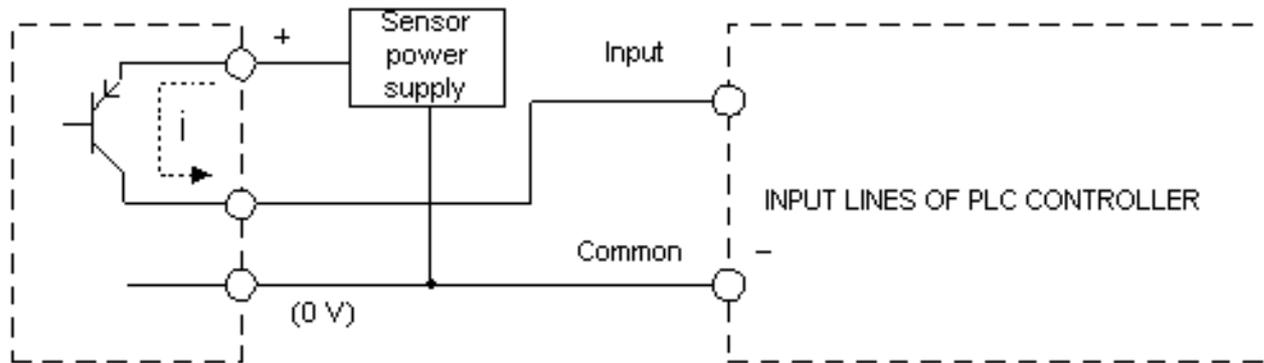


Connecting sensor with a NPN current output to a PLC controller input

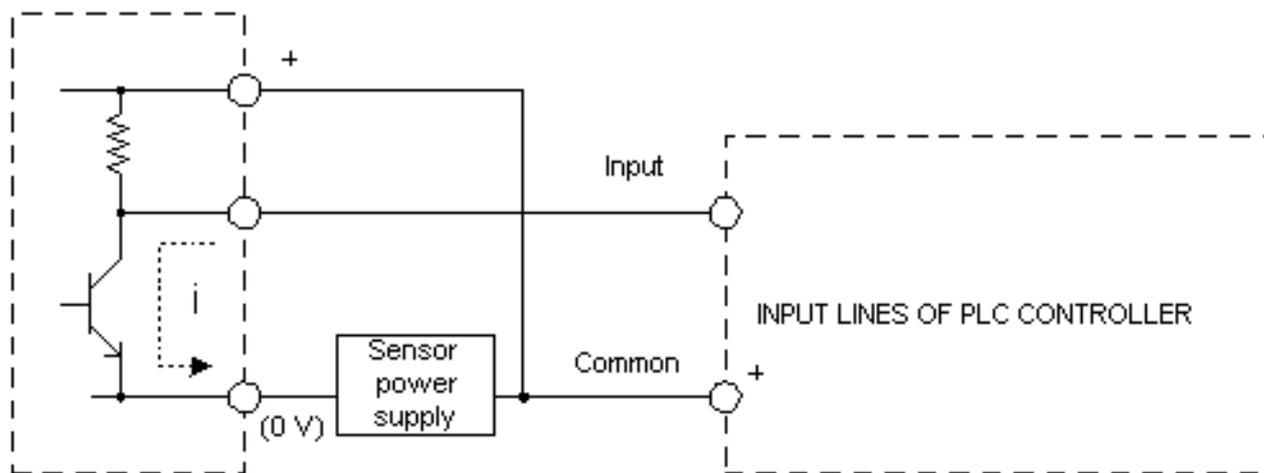


Connecting sensor with NPN open collector to a PLC controller input

Connecting sensor with NPN open collector to a PLC controller input



Connecting sensor with PNP current output to a PLC controller input



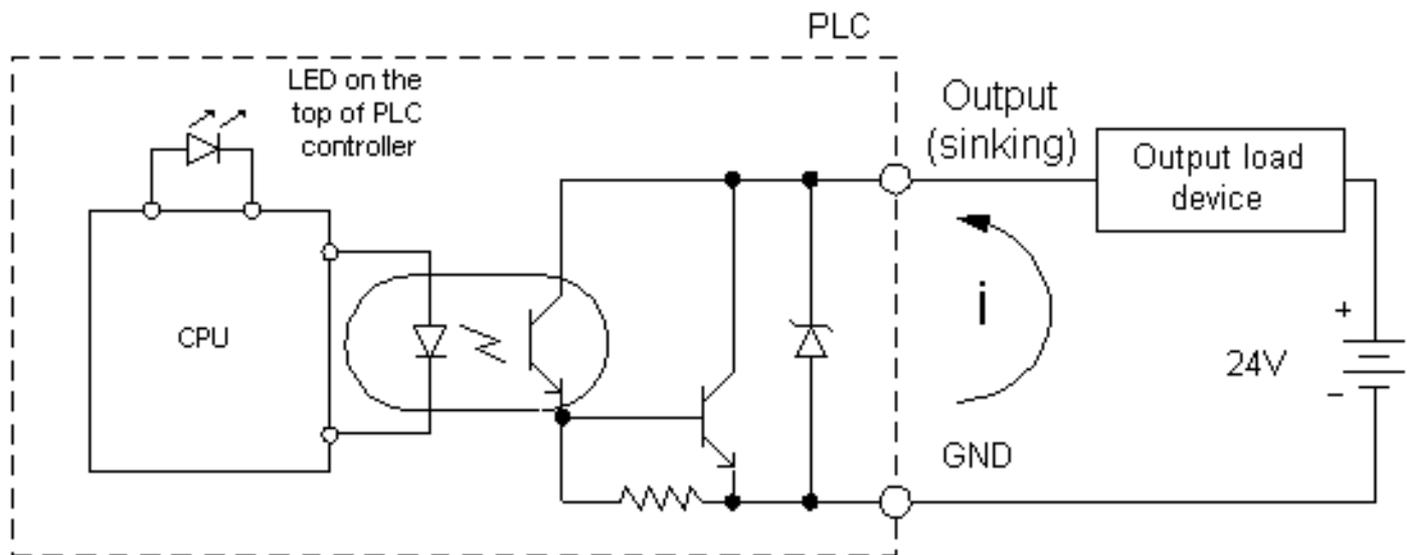
Connecting sensor with voltage output to a PLC controller input

3.3 Output lines

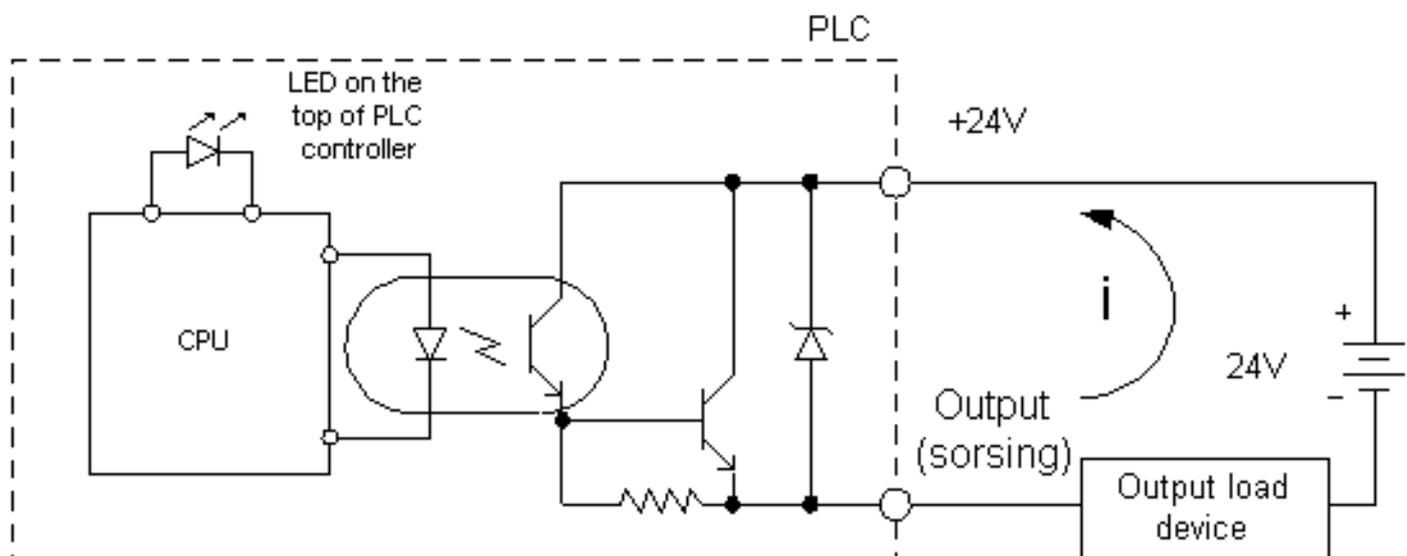
PLC controller output lines usually can be:

- transistors in PNP connection
- transistors in NPN connection
- relays

The following two pictures display a realistic way how a PLC manages external devices. It ought to be noted that a main difference between these two pictures is a position of "output load device". By "output load device" we mean some relay, signalization light or similar.



Connecting output load device to a sinking PLC controller output



Connecting output load device to a sourcing PLC controller output

How something is connected with a PLC output depends on the element being connected. In short, it depends on whether this element of output load device is activated by a positive supply pole or a negative supply pole.

CHAPTER 4 Architecture of specific PLC controller

[Introduction](#)

[4.1 Why OMRON?](#)

[4.2 CPM1A PLC controller](#)

[4.3 PLC controller input lines](#)

[4.4 PLC controller output lines](#)

[4.5 How a PLC controller works](#)

[4.6 CPM1A PLC controller memory map](#)

[4.7 Timers and counters](#)

Introduction

This book could deal with a general overview of some supposed PLC controller. Author has had an opportunity to look over plenty of books published up till now, and this approach is not the most suitable to the purposes of this book in his opinion. Idea of this book is to work through one specific PLC controller where someone can get a real feeling on this subject and its weight. Our desire was to write a book based on whose reading you can earn some money. After all, money is the end goal of every business!

4.1 Why OMRON?

Why not? That is one huge firm which has high quality and by our standards inexpensive controllers. Today we can say almost with surety that PLC controllers by manufacturers round the world are excellent devices, and altogether similar. Nevertheless, for specific application we need to know specific information about a PLC controller being used. Therefore, the choice fell on OMRON company and its PLC of micro class CPM1A. Adjective "micro" itself implies smallest models from the viewpoint of a number of attached lines or possible options. Still, this PLC controller is ideal for the purposes of this book, and that is to introduce a PLC controller philosophy to its readers.

4.2 CPM1A PLC controller

Each PLC is basically a microcontroller system (CPU of PLC controller is based on one of the microcontrollers, and in more recent times on one of the PC processors) with peripherals that can be digital inputs, digital outputs or relays as in our case. However, this is not an "ordinary" microcontroller system. Large teams have worked on it, and a checkup of its function has been performed in real world under all possible circumstances. Software itself is entirely different from assemblers used thus far, such as BASIC or C. This specialized software is called "ladder" (name came about by an association of program's configuration which resembles a ladder, and from the way program is written out).

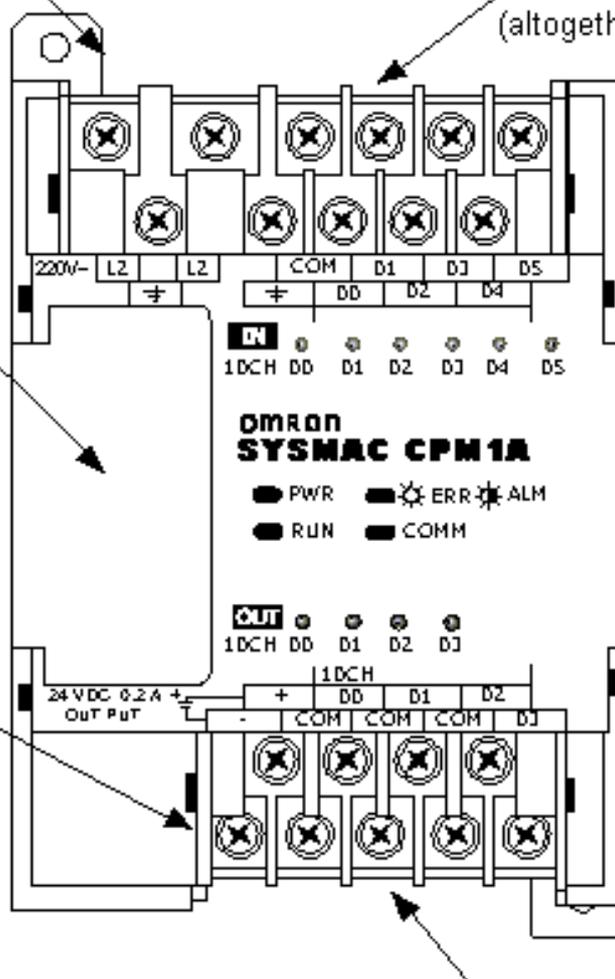
Specific look of CPM1A PLC controller can be seen in the following picture. On the upper surface, there are 4 LED indicators and a connection port with an RS232 module which is interface to a PC computer. Aside from this, screw terminals and light indicators of activity of each input or output are visible on upper and lower sides. Screw terminals serve to manually connect to a real system. Hookups L1 and L2 serve as supply which is 220V~ in this case. PLC controllers that work on power grid voltage usually have a source of direct supply of 24 VDC for supplying sensors and such (with a CPM1A source of direct supply is found on the bottom left hand side and is represented with two screw terminals. Controller can be mounted to industrial "track" along with other elements of automatization, but also by a screw to the machine wall or control panel.

Three screw terminal for a hookup of 220VAC

Inputs to a PLC controller from 00 to 05 (altogether six)

Beneath a small plastic cover is found a connector to hookup an RS232 interface for connection with a PC computer.

Supply of 24VDC to incite input or a sensor.



Programmable logic controller CPM1A

PLC controller outputs, from 00 to 03 (altogether 4)

Controller is 8cm high and divided vertically into two areas: a lower one with a converter of 220V~ at 24VDC and other voltages needed for running a CPU unit; and, upper area with a CPU and memory, relays and digital inputs.

When you lift the small plastic cover you'll see a connector to which an RS232 module is hooked up for serial interface with a computer. This module is used when programming a PLC controller to change programs or execution follow-up. When installing a PLC it isn't necessary to install this module, but it is recommended because of possible changes in software during operation.

CPM1A

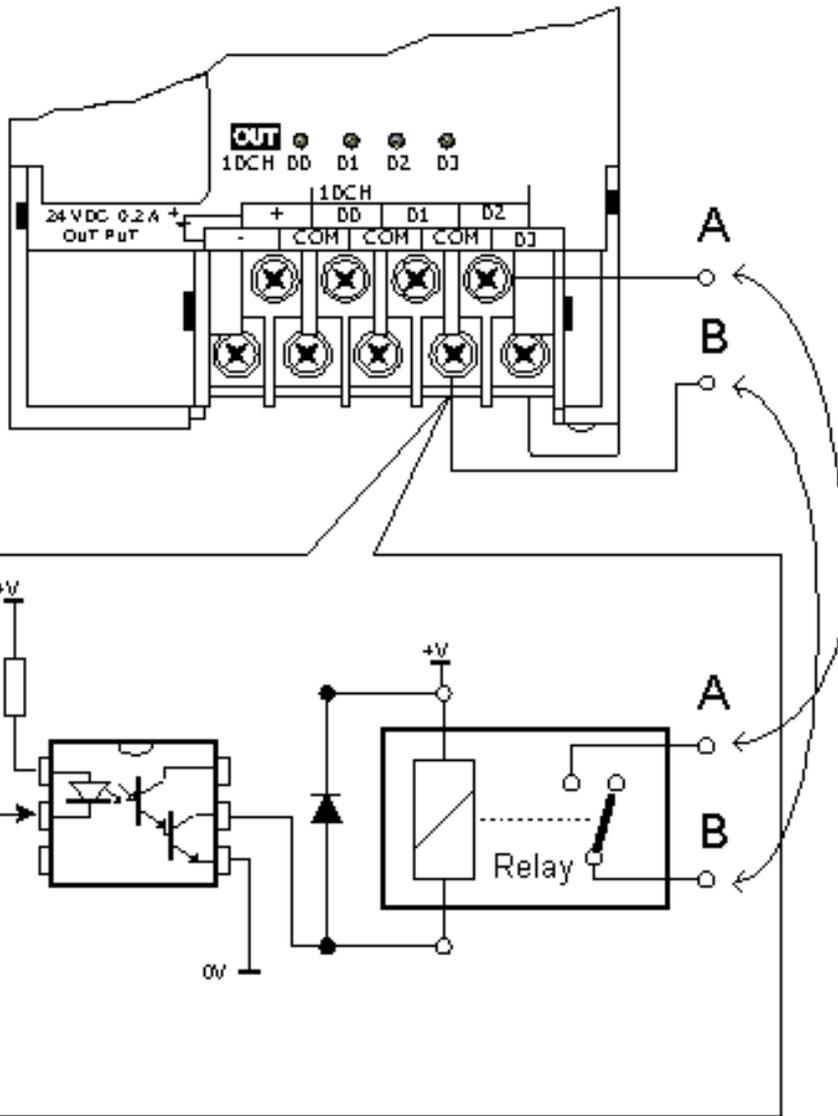


To better inform programmers on PLC controller status, maker has provided for four light indicators in the form of LED's. Beside these indicators, there are status indicators for each individual input and output. These LED's are found by the screw terminals and with their status are showing input or output state. If input/output is active, diode is lit and vice versa.

4.3 PLC controller output lines

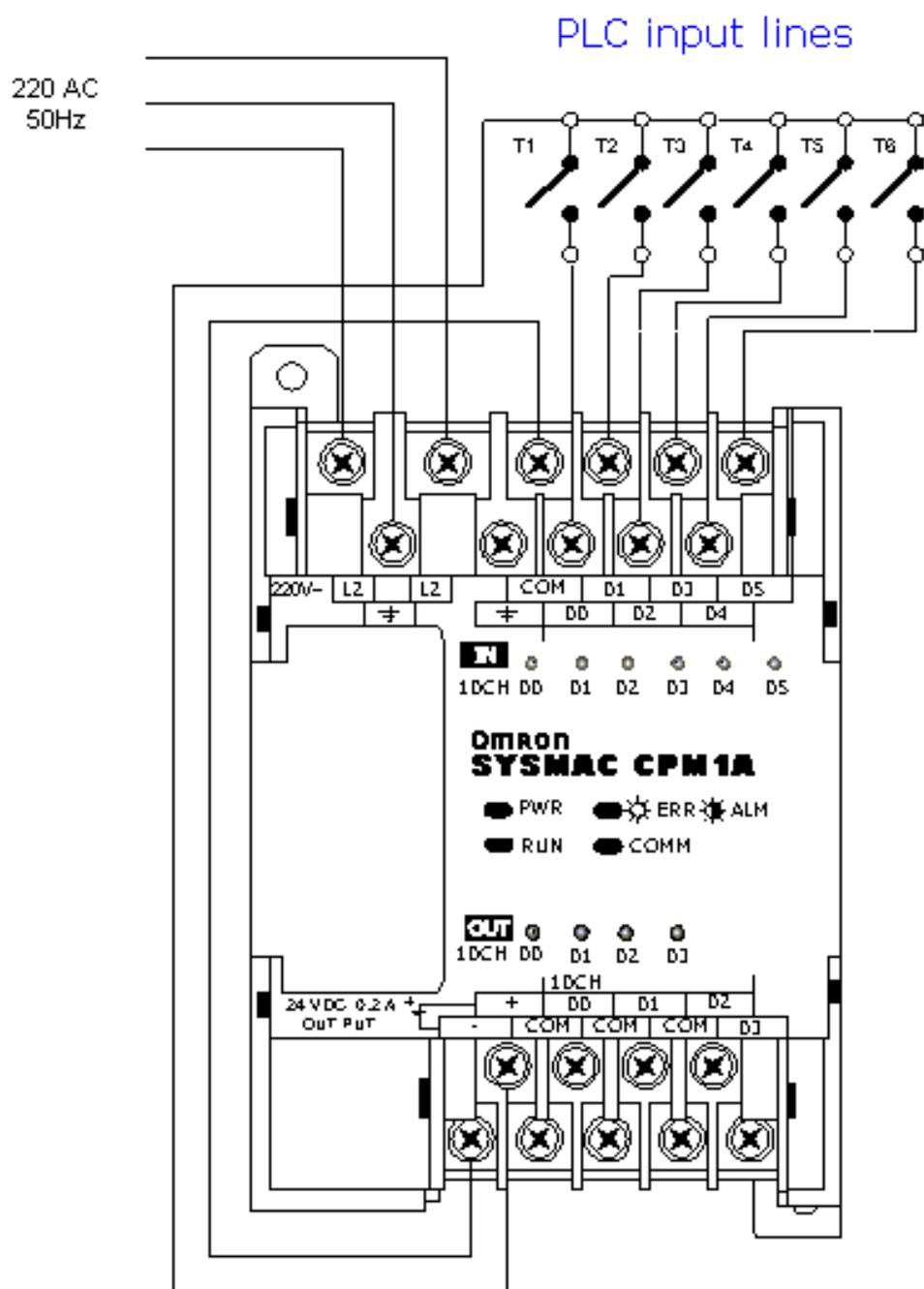
Aside from transistor outputs in PNP and NPN connections, PLC can also have relays as outputs. Existence of relays as outputs makes it easier to connect with external devices. Model CPM1A contains exactly these relays as outputs. There a 4 relays whose functional contacts are taken out on a PLC controller housing in the form of screw terminals. In reality this looks as in picture below.

Relay output



With activation of phototransistor, relay comes under voltage and activates a contact between points A and B. Contacts A and B can in our case be either in connection or interrupted. What state these contacts are in is determined by a CPU through appropriate bits in memory location IR010. One example of relay status is shown in a picture below. A true state of devices attached to these relays is displayed.

source of direct voltage of 24V, the same source can be used to incite input (problem with this source is its maximum current which it can give continually and which in our case amounts to 0.2A). Since inputs to a PLC are not big consumers (unlike some sensor where a stronger external supply must be used) it is possible to take advantage of the existing source of direct supply to incite all six keys.



4.5 How a PLC controller functions

Basis of a PLC function is continual scanning of a program. Under scanning we mean running through all conditions within a guaranteed period. Scanning process has three basic steps:

Step 1.

Testing input status. First, a PLC checks each of the inputs with intention to see which one of them has status ON or OFF. In other words, it checks whether a sensor, or a switch etc. connected with an input is activated or not. Information that processor thus obtains through this step is stored in memory in order to be used in the following step.

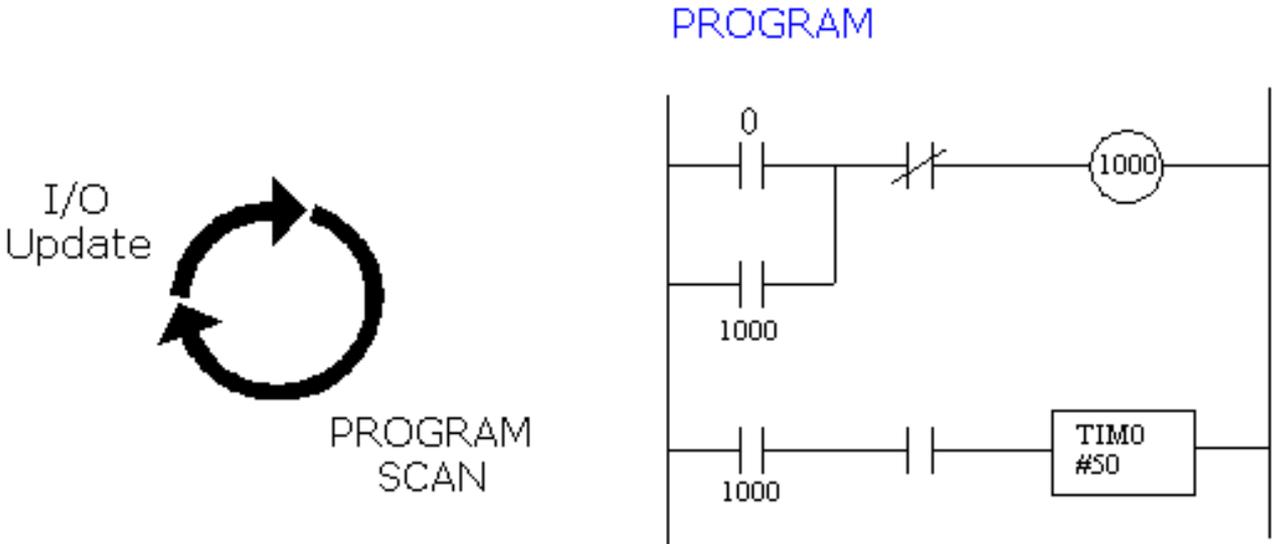
Step 2.

Program execution. Here a PLC executes a program, instruction by instruction. Based on a program

and based on the status of that input as obtained in the preceding step, an appropriate action is taken. This reaction can be defined as activation of a certain output, or results can be put off and stored in memory to be retrieved later in the following step.

Step 3.

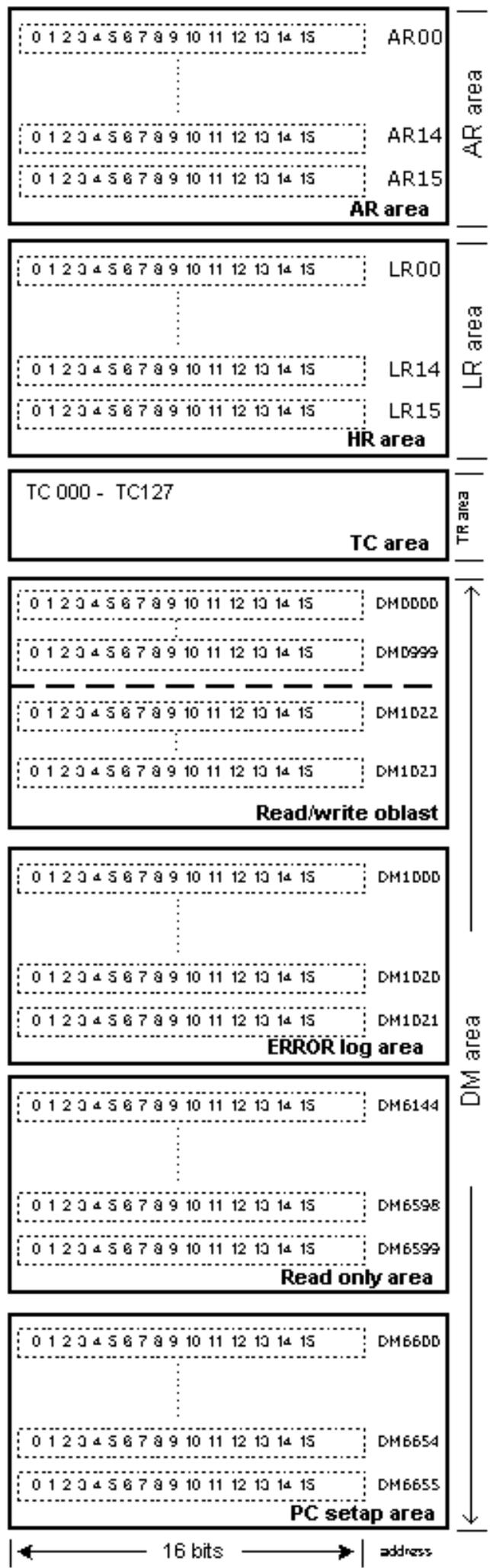
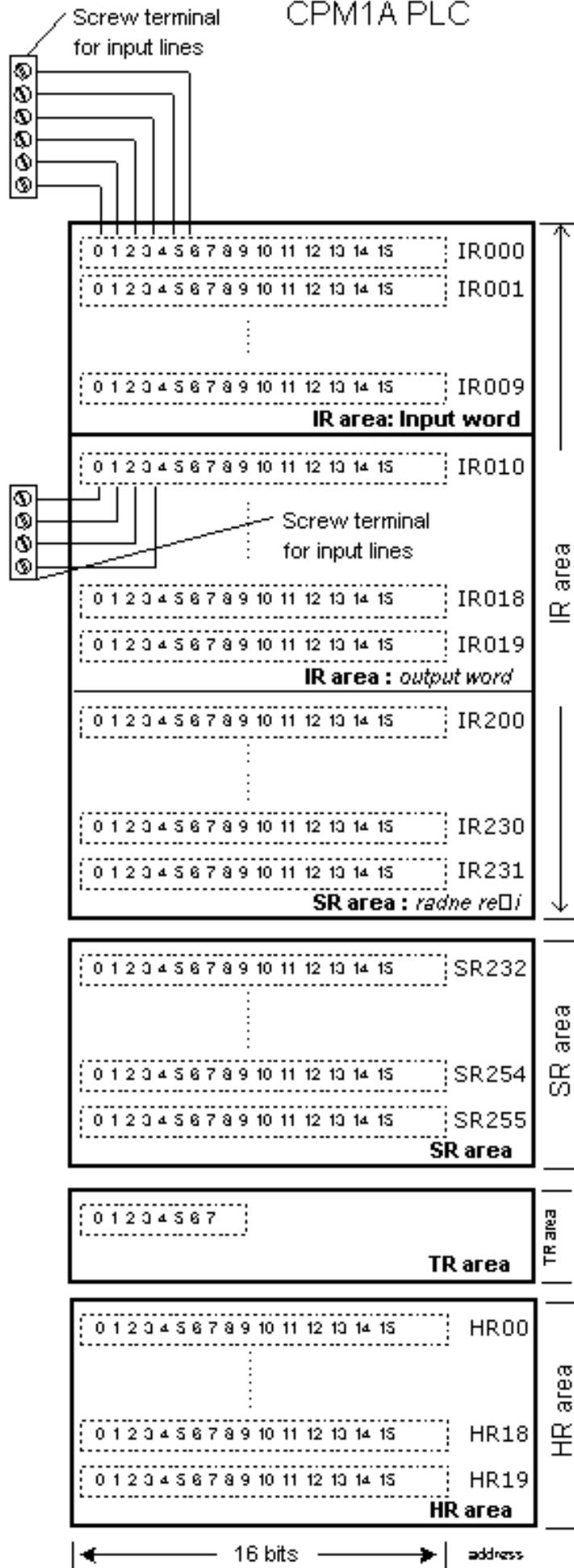
Checkup and correction of output status. Finally, a PLC checks up output status and adjusts it as needed. Change is performed based on the input status that had been read during the first step, and based on the results of program execution in step two. Following the execution of step 3 PLC returns to the beginning of this cycle and continually repeats these steps. Scanning time is defined by the time needed to perform these three steps, and sometimes it is an important program feature.



4.6 CPM1A PLC controller memory map

By memory map we mean memory structure for a PLC controller. Simply said, certain parts of memory have specific roles. If you look at the picture below, you can see that memory for CPM1A is structured into 16-bit words. A cluster of several such words makes up a region. All the regions make up the memory for a PLC controller.

MEMORU MAP of CPM1A PLC



Unlike microcontroller systems where only some memory locations have had their purpose clearly defined (ex. register that contains counter value), a memory of PLC controller is completely defined, and more importantly almost entire memory is addressable in bits. Addressability in bits means that it is enough to write the address of the memory location and a number of bits after it in order to manipulate with it. In short, that would mean that something like this could be written: "201.7=1" which would clearly indicate a word 201 and its bit 7 which is set to one.

IR region

Memory locations intended for PLC input and output. Some bits are directly connected to PLC controller inputs and outputs (screw terminal). In our case, we have 6 input lines at address IR000. One bit corresponds to each line, so the first line has the address IR000.0, and the sixth IR000.5. When you obtain a signal at the input, this immediately affects the status of a corresponding bit. There are also words with work bits in this region, and these work bits are used in a program as flags or certain conditional bits.

SR region

Special memory region for control bits and flags. It is intended first and foremost for counters and interrupts. For example, SR250 is memory location which contains an adjustable value, adjusted by potentiometer no.0 (in other words, value of this location can be adjusted manually by turning a potentiometer no.0).

TR region

When you move to a subprogram during program execution, all relevant data is stored in this region up to the return from a subprogram.

HR region

It is of great importance to keep certain information even when supply stops. This part of the memory is battery supported, so even when supply has stopped it will keep all data found therein before supply stopped.

AR region

This is one more region with control bits and flags. This region contains information on PLC status, errors, system time, and the like. Like HR region, this one is also battery supported.

LR region

In case of connection with another PLC, this region is used for exchange of data.

Timer and counter region

This region contains timer and counter values. There are 128 values. Since we will consider examples with timers and counters, we will discuss this region more later on.

DM region

Contains data related to setting up communication with a PC computer, and data on errors.

Each region can be broken down to single words and meanings of its bits. In order to keep the clarity of the book, this part is dealt with in Attachments and we will deal with those regions here whose bits are mostly used for writing.

Data area		Words	Bits	Function
IR area ¹	Input area	IR 000 to IR 009 (10 words)	IR 00000 to IR 00915 (160 bits)	These bits can be allocated to the external I/O terminals.
	Output area	IR 010 to IR 019 (10 words)	IR 01000 to IR 01915 (160 bits)	
	Work area	IR 200 to IR 231 (32 words)	IR 20000 to IR 23115 (512 bits)	Work bits can be freely used within the program.
SR area		SR 232 to SR 255 (24 words)	SR 23200 to SR 25515 (384 bits)	These bits serve specific functions such as flags and control bits.
TR area		---	TR 0 to TR 7 (8 bits)	These bits are used to temporarily store ON/OFF status at program branches.
HR area ²		HR 00 to HR 19 (20 words)	HR 0000 to HR 1915 (320 bits)	These bits store data and retain their ON/OFF status when power is turned off.
AR area ²		AR 00 to HR 15 (16 words)	AR 0000 to HR 1515 (256 bits)	These bits serve specific functions such as flags and control bits.
LR area ¹		LR 00 to LR 15 (16 words)	LR 00000 to LR1515 (256 bits)	Used for a 1:1 data link with another PC.
Timer/Counter area ²		TC 000 to TC 127 (timer/counter numbers) ³		The same numbers are used for both timers and counters.
DM area	Read/write ²	DM 0000 to DM 0999 DM 1022 to DM 1023 (1,002 words)	---	DM area data can be accessed in word units only. Word values are required when the power is turned off.
	Error log ⁴	DM 1000 to DM 1021 (22 words)	---	Used to store the timer of occurrence and error code of errors that occur. These words can be used as ordinary read/write DM when the error log function isn't being used.
	Read-only ⁴	DM 6144 to DM 6599 (456 words)	---	Cannot be overwritten from program.
	PC Setup ⁴	DM 6600 to DM 6655 (56 words)	---	Used to store various parameters that control PC operation.

Note:

1. IR and LR bits that are not used for their allocated functions can be used as work bits.
2. The contents of the HR area, LR area, Counter area, and read/write DM area are backed up by a capacitor. At 25 oC, the capacitor will back up memory for 20 days.
3. When accessing a PV, TC numbers are used as word data; when accessing Completing Flags, they are used as bit data.
4. Data in DM6144 to DM6655 cannot be overwritten from the program, but they can be changed from a Peripheral Device

4.7 Timers and counters

Timers and counters are indispensable in PLC programming. Industry has to number its products, determine a needed action in time, etc. Timing functions is very important, and cycle periods are critical in many processes.

There are two types of timers delay-off and delay-on. First is late with turn off and the other runs late in turning on in relation to a signal that activated timers. Example of a delay-off timer would be staircase lighting. Following its activation, it simply turns off after few minutes.

Each timer has a time basis, or more precisely has several timer basis. Typical values are: 1 second, 0.1 second, and 0,01 second. If programmer has entered .1 as time basis and 50 as a number for delay increase, timer will have a delay of 5 seconds ($50 \times 0.1 \text{ second} = 5 \text{ seconds}$).

Timers also have to have value SV set in advance. Value set in advance or ahead of time is a number of increments that timer has to calculate before it changes the output status. Values set in advance can be constants or variables. If a variable is used, timer will use a real time value of the variable to determine a delay. This enables delays to vary depending on the conditions during function. Example is a system that has produced two different products, each requiring different timing during process itself. Product A requires a period of 10 seconds, so number 10 would be assigned to the variable. When product B appears, a variable can change value to what is required by product B.

Typically, timers have two inputs. First is timer enable, or conditional input (when this input is activated, timer will start counting). Second input is a reset input. This input has to be in OFF status in order for a timer to be active, or the whole function would be repeated over again. Some PLC models require this input to be low for a timer to be active, other makers require high status (all of them function in the same way basically). However, if reset line changes status, timer erases accumulated value.

With a PLC controller by Omron there are two types of timers: TIM and TIMH. TIM timer measures in increments of 0.1 seconds. It can measure from 0 to 999.9 seconds with precision of 0.1 seconds more or less.

Quick timer (TIMH) measures in increments of 0.01 seconds. Both timers are "delay-on" timers of a lessening-style. They require assignment of a timer number and a set value (SV). When SV runs out, timer output turns on. Numbers of a timing counter refer to specific address in memory and must not be duplicated (same number can not be used for a timer and a counter).

CHAPTER 5 Ladder diagram

[Introduction](#)

[5.1 Ladder diagram](#)

[5.2 Normally open and normally closed contacts](#)

[5.3 Brief example](#)

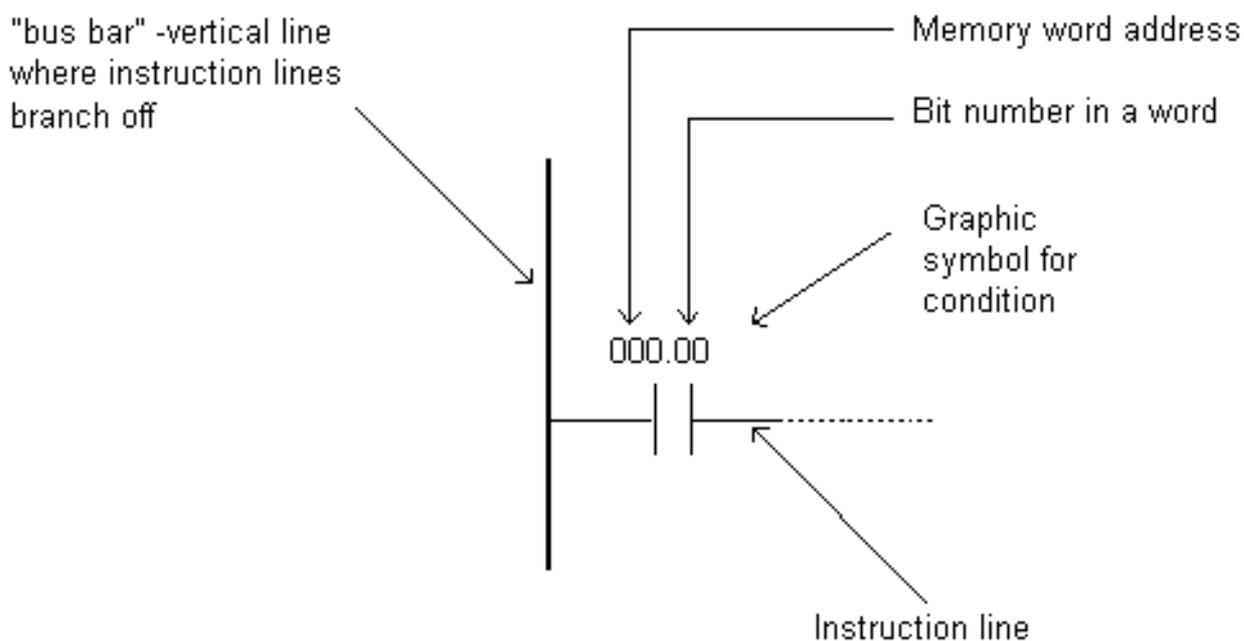
Introduction

Programmable controllers are generally programmed in ladder diagram (or "relay diagram") which is nothing but a symbolic representation of electric circuits. Symbols were selected that actually looked similar to schematic symbols of electric devices, and this has made it much easier for electricians to switch to programming PLC controllers. Electrician who has never seen a PLC can understand a ladder diagram.

5.1 Ladder diagram

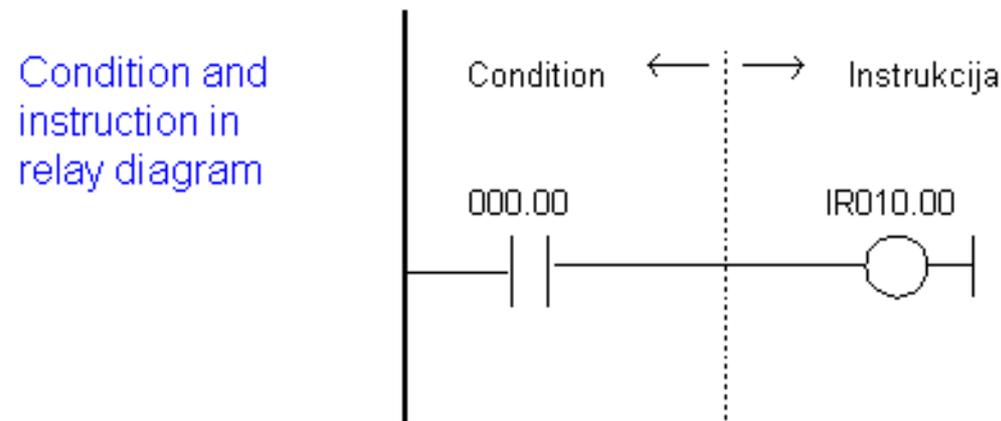
There are several languages designed for user communication with a PLC, among which ladder diagram is the most popular. Ladder diagram consists of one vertical line found on the left hand side, and lines which branch off to the right. Line on the left is called a "bus bar", and lines that branch off to the right are instruction lines. Conditions which lead to instructions positioned at the right edge of a diagram are stored along instruction lines. Logical combination of these conditions determines when and in what way instruction on the right will execute. Basic elements of a relay diagram can be seen in the following picture.

Basic elements of a relay diagram



Most instructions require at least one operand, and often more than one. Operand can be some memory location, one memory location bit, or some numeric value -number. In the example above, operand is bit 0 of memory location IR000. In a case when we wish to proclaim a constant as an operand, designation # is used beneath the numeric writing (for a compiler to know it is a constant and not an address.)

Based on the picture above, one should note that a ladder diagram consists of two basic parts: left section also called conditional, and a right section which has instructions. When a condition is fulfilled, instruction is executed, and that's all!



Picture above represents an example of a ladder diagram where a relay is activated in a PLC controller when a signal appears at input line 00. Vertical line pairs are called conditions. Each condition in a ladder diagram has a value ON or OFF, depending on a bit status assigned to it. In this case, this bit is also physically present as an input line (screw terminal) to a PLC controller. If a key is attached to a corresponding screw terminal, you can change bit status from a logic one status to a logic zero status, and vice versa. Status of logic one is usually designated as "ON", and status of logic zero as "OFF".

Right section of a ladder diagram is an instruction which is executed if the left condition is fulfilled. There are several types of instructions that could easily be divided into simple and complex. Example of a simple instruction is activation of some bit in memory location. In the example above, this bit has physical connotation because it is connected with a relay inside a PLC controller. When a CPU activates one of the leading four bits in a word IR010, relay contacts move and connect lines attached to it. In this case, these are the lines connected to a screw terminal marked as 00 and to one of COM screw terminals.

5.2 Normally open and normally closed contacts

Since we frequently meet with concepts "normally open" and "normally closed" in an industrial environment, it's important to know them. Both terms apply to words such as contacts, input, output, etc. (all combinations have the same meaning whether we are talking about input, output, contact or something else).

Principle is quite simple, normally open switch won't conduct electricity until it is pressed down, and normally closed switch will conduct electricity until it is pressed. Good examples for both situations are the doorbell and a house alarm.

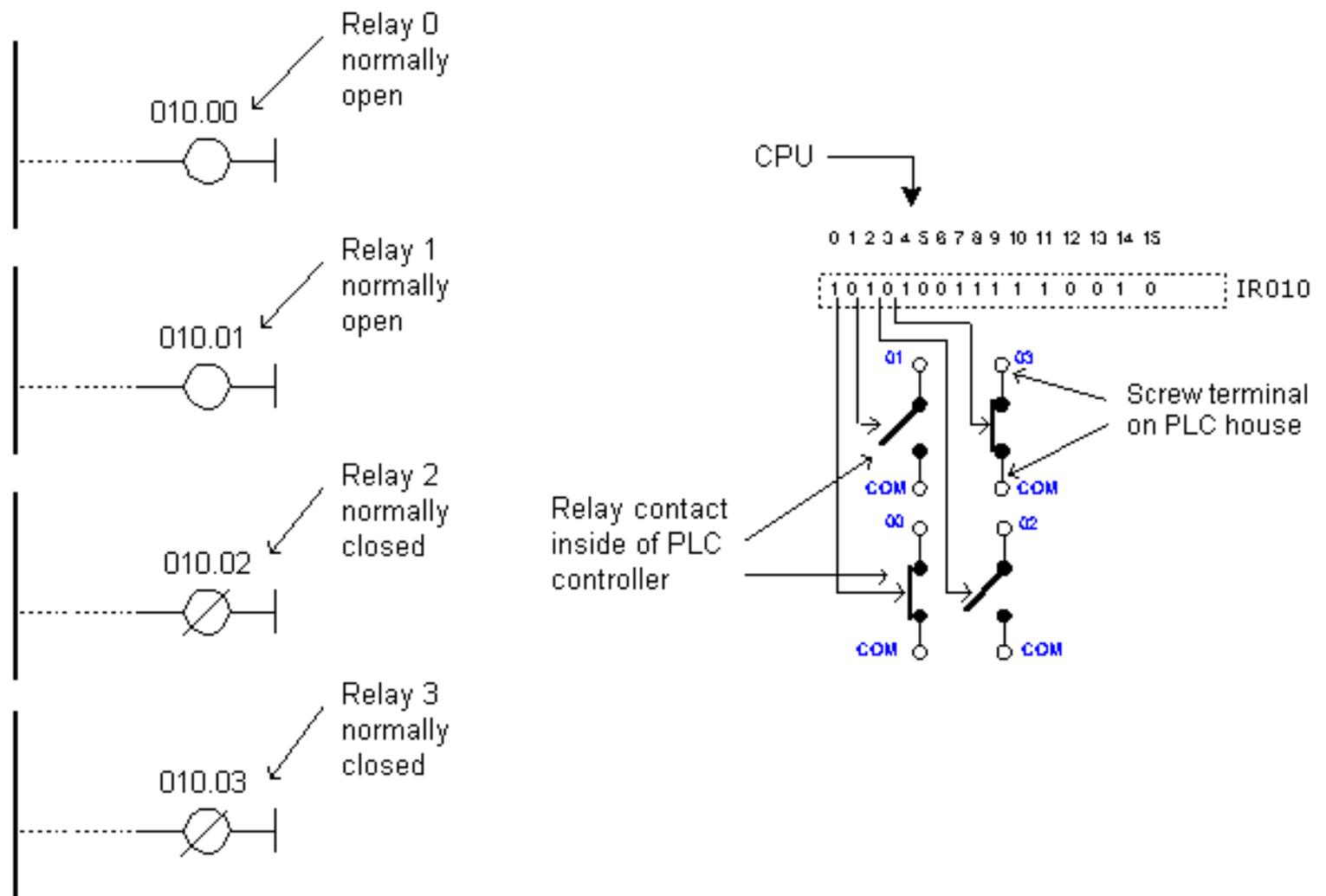
If a normally closed switch is selected, bell will work continually until someone pushes the switch. By pushing a switch, contacts are opened and the flow of electricity towards the bell is interrupted. Of course, a system so designed would not in any case suit the owner of the house. A better choice would certainly be a normally open switch. This way bell wouldn't work until someone pushed the switch button and thus informed of his or her presence at the entrance.

Home alarm system is an example of an application of a normally closed switch. Let's suppose

that alarm system is intended for surveillance of the front door to the house. One of the ways to "wire" the house would be to install a normally open switch from each door to the alarm itself (precisely as with a bell switch). Then, if the door was opened, this would close the switch, and an alarm would be activated. This system could work, but there would be some problems with this, too. Let's suppose that switch is not working, that a wire is somehow disconnected, or a switch is broken, etc. (there are many ways in which this system could become dysfunctional). The real trouble is that a homeowner would not know that a system was out of order. A burglar could open the door, a switch would not work, and the alarm would not be activated. Obviously, this isn't a good way to set up this system. System should be set up in such a way so the alarm is activated by a burglar, but also by its own dysfunction, or if any of the components stopped working. (A homeowner would certainly want to know if a system was dysfunctional). Having these things in mind, it is far better to use a switch with normally closed contacts which will detect an unauthorized entrance (opened door interrupts the flow of electricity, and this signal is used to activate a sound signal), or a failure on the system such as a disconnected wire. These considerations are even more important in industrial environment where a failure could cause injury at work. One such example where outputs with normally closed contacts are used is a safety wall with trimming machines. If the wall doors open, switch affects the output with normally closed contacts and interrupts a supply circuit. This stops the machine and prevents an injury.

Concepts normally open and normally closed can apply to sensors as well. Sensors are used to sense the presence of physical objects, measure some dimension or some amount. For instance, one type of sensors can be used to detect presence of a box on an industry transfer belt. Other types can be used to measure physical dimensions such as heat, etc. Still, most sensors are of a switch type. Their output is in status ON or OFF depending on what the sensor "feels". Let's take for instance a sensor made to feel metal when a metal object passes by the sensor. For this purpose, a sensor with a normally open or a normally closed contact at the output could be used. If it were necessary to inform a PLC each time an object passed by the sensor, a sensor with a normally open output should be selected. Sensor output would set off only if a metal object were placed right before the sensor. A sensor would turn off after the object has passed. PLC could then calculate how many times a normally open contact was set off at the sensor output, and would thus know how many metal objects passed by the sensor.

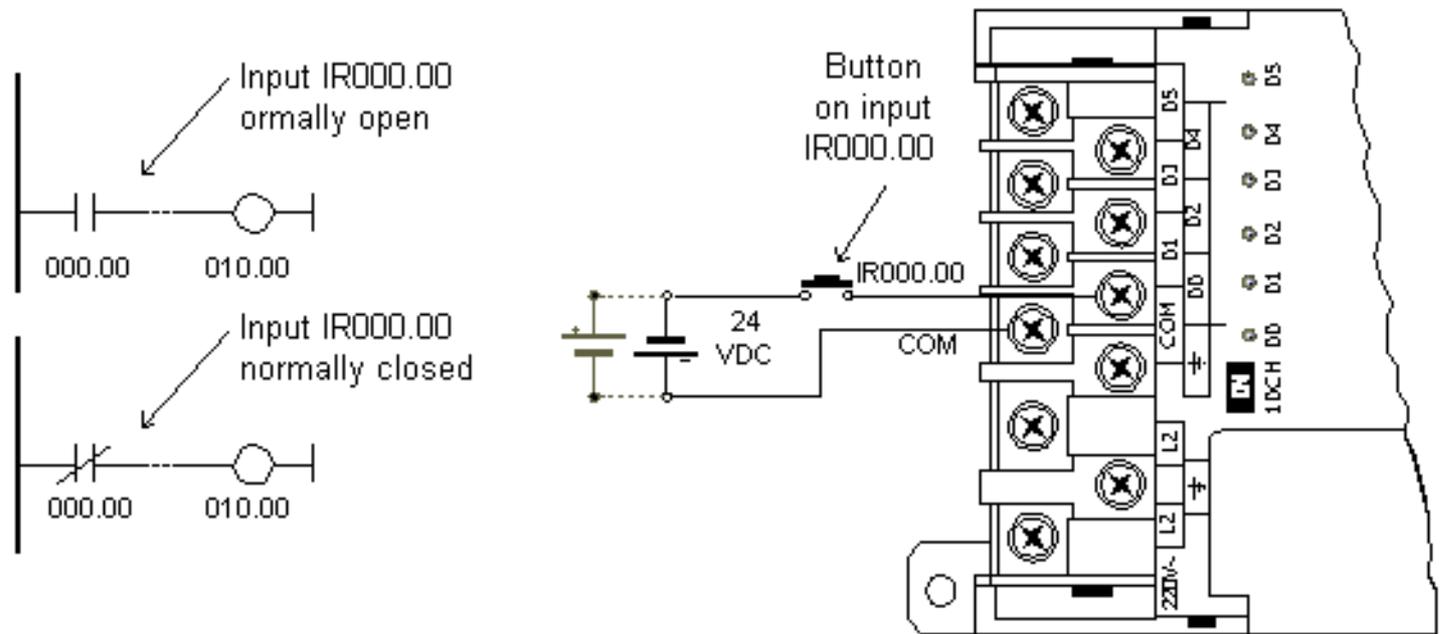
Concepts normally open and normally closed contact ought to be clarified and explained in detail in the example of a PLC controller input and output. The easiest way to explain them is in the example of a relay.



Normally open contacts would represent relay contacts that would perform a connection upon receipt of a signal. Unlike open contacts, with normally closed contacts signal will interrupt a contact, or turn a relay off. Previous picture shows what this looks like in practice. First two relays are defined as normally open, and the other two as normally closed. All relays react to a signal! First relay (00) has a signal and closes its contacts. Second relay (01) does not have a signal and remains opened. Third relay (02) has a signal and opens its contacts considering it is defined as a closed contact. Fourth relay (03) does not have a signal and remains closed because it is so defined.

Concepts "normally open" and "normally closed" can also refer to inputs of a PLC controller. Let's use a key as an example of an input to a PLC controller. Input where a key is connected can be defined as an input with open or closed contacts. If it is defined as an input with normally open contact, pushing a key will set off an instruction found after the condition. In this case it will be an activation of a relay 00.

If input is defined as an input with normally closed contact, pushing the key will interrupt instruction found after the condition. In this case, this will cause deactivation of relay 00 (relay is active until the key is pressed). You can see in picture below how keys are connected, and view the relay diagrams in both cases.

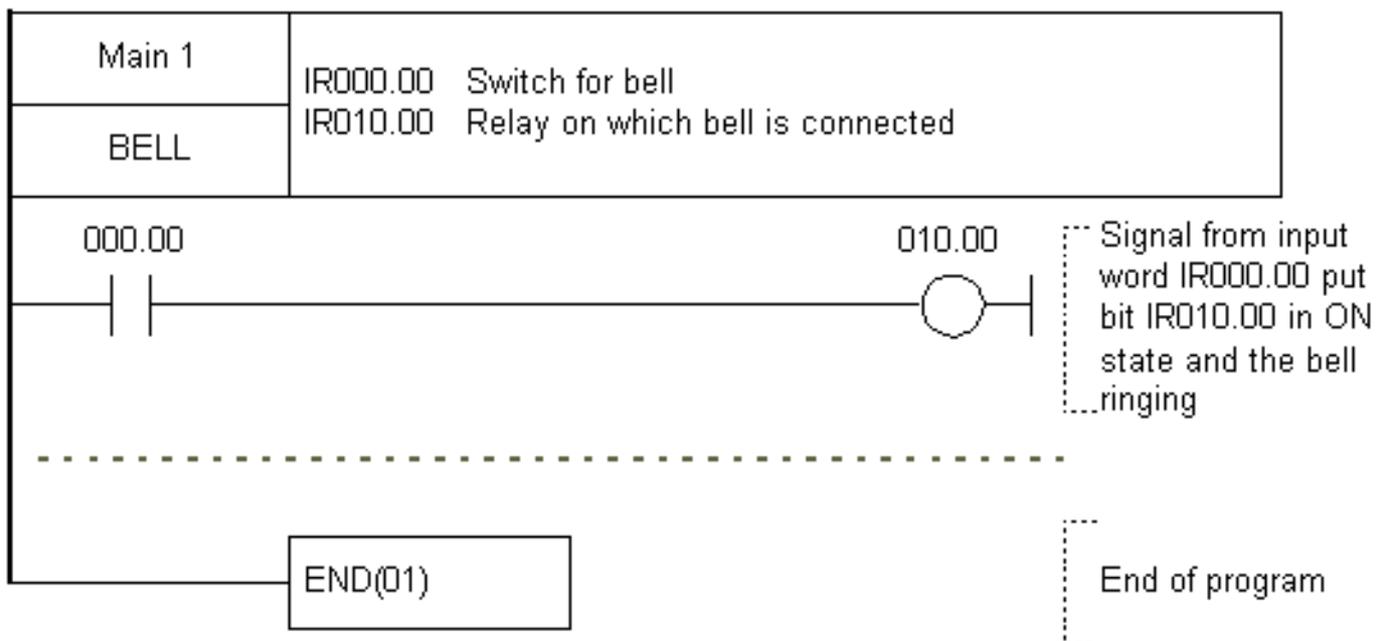


Normally open/closed conditions differ in a ladder diagram by a diagonal line across a symbol. What determines an execution condition for instruction is a bit status marked beneath each condition on instruction line. Normally open condition is ON if its operand bit has ON status, or its status is OFF if that is the status of its operand bit. Normally closed condition is ON when its operand bit is OFF, or it has OFF status when the status of its operand bit is ON.

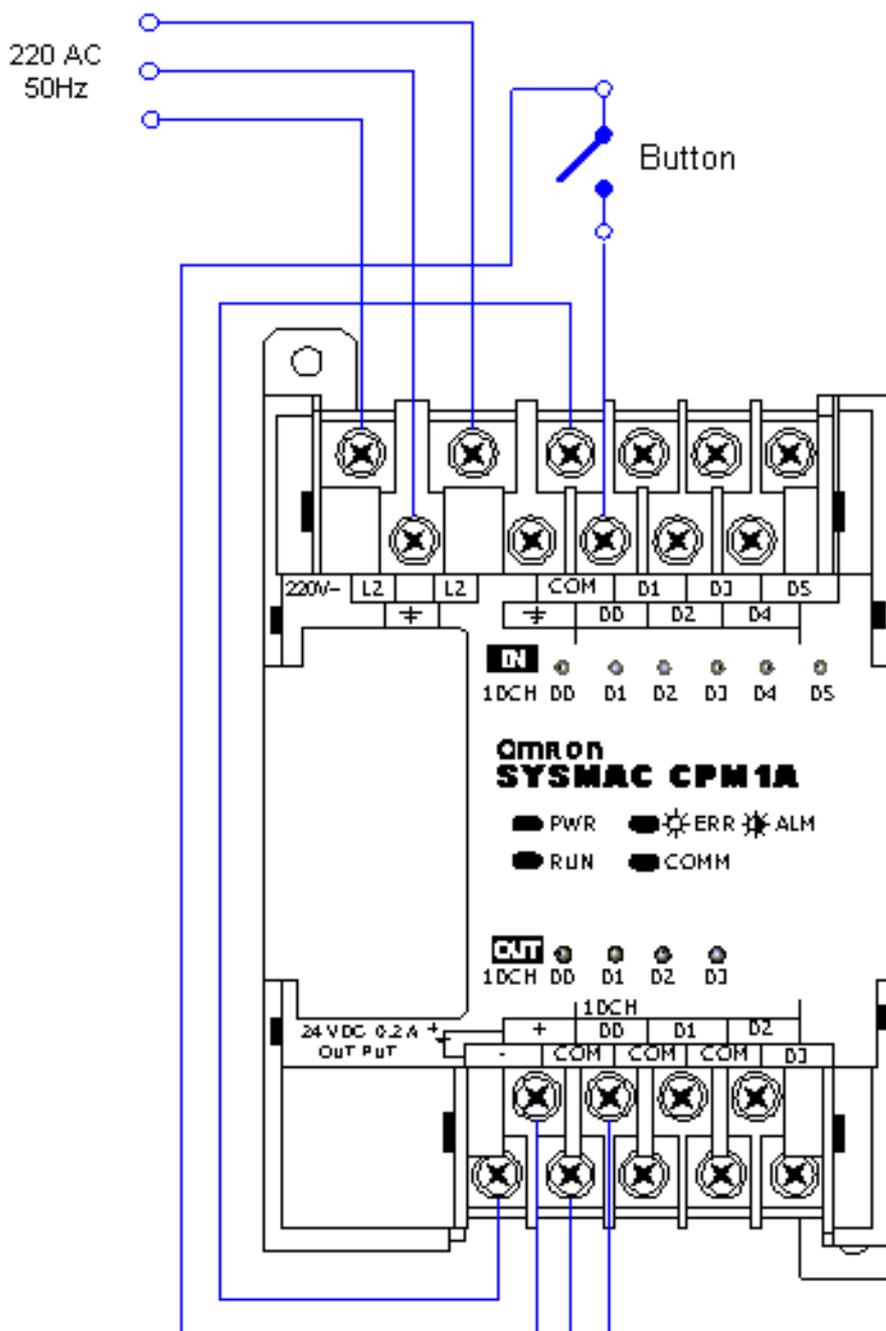
When programming with a ladder diagram, logical combination of ON and OFF conditions set before the instruction determines the eventual condition under which the instruction will be, or will not be executed. This condition, which can have only ON or OFF values is called instruction execution condition. Operand assigned to any instruction in a relay diagram can be any bit from IR, SR, HR, AR, LR or TC sector. This means that conditions in a relay diagram can be determined by a status of I/O bits, or of flags, operational bits, timers/counters, etc.

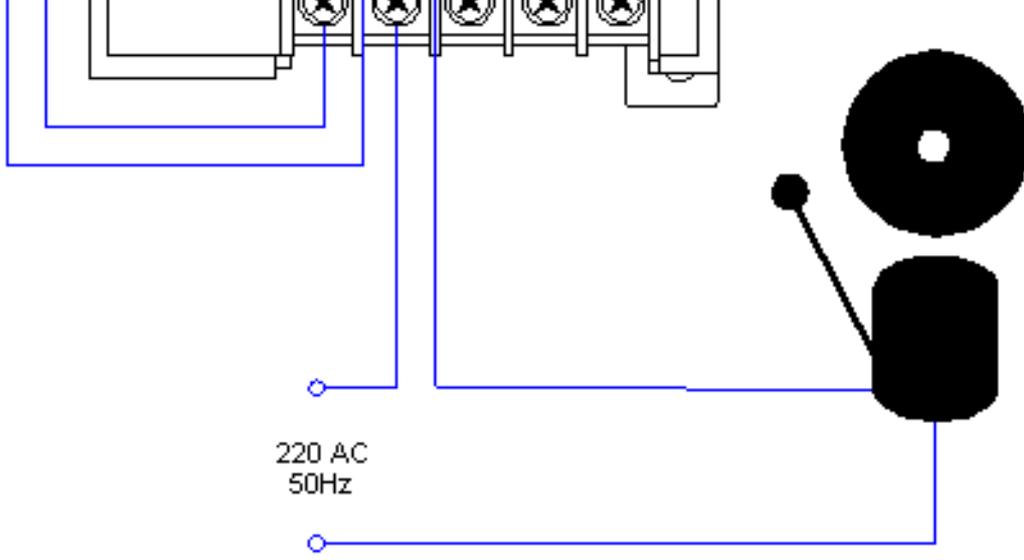
5.3 Brief example

Example below represents a basic program. Example consists of one input device and one output device linked to the PLC controller output. Key is an input device, and a bell is an output supplied through a relay 00 contact at the PLC controller output. Input 000.00 represents a condition in executing an instruction over 010.00 bit. Pushing the key sets off a 000.00 bit and satisfies a condition for activation of a 010.00 bit which in turn activates the bell. For correct program function another line of program is needed with END instruction, and this ends the program.



The following picture depicts the connection scheme for this example.





CHAPTER 6 SYSWIN program for programming a PLC controller

[Introduction](#)

[6.1 Connecting a PLC controller with a PC computer](#)

[6.2 SYSWIN program installation](#)

[6.3 Writing your first program](#)

[6.4 Saving a project](#)

[6.5 Program transfer to PLC controller](#)

[6.6 Testing program function](#)

[6.7 Interpretation of "Tools" icons](#)

[6.8 PLC controller working modes](#)

[6.9 Run mode](#)

[6.10 Monitor mode](#)

[6.11 Program-Stop mode](#)

[6.12 Program execution and monitoring](#)

[6.13 Impact on the program during monitoring](#)

[6.14 Graphic representation of dimension changes in a program](#)

Introduction

SYSWIN is a software designed for OMRON programmable controllers class C and CV. It is designed for creating and maintaining a program, as well as for testing PLC controller function, in off-line and controller's operational regime.

Necessary conditions for starting SYSWIN are Microsoft Windows environment on a standard IBM or 386/486 compatible or Pentium computer, with 8MB RAM at least, and 10MB free disc space.

6.1 Connecting a PLC controller with a PC computer

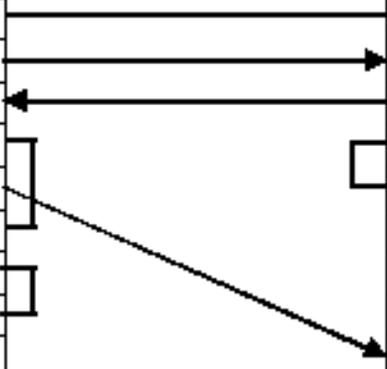
PLC controller is linked with a PC computer through an RS-232 cable. One end of the cable is connected to a serial PC port (9-pin or 25-pin connector), while the other end is connected to an RS-232C connector on RS232 module of a CPM1A controller. In order to establish a connection with a PC, DIP switch on the connector must be set in "Host" position.

IBM PC/AT or Compatible RS-232

Signal	Pin
FG	1
RD	2
SD	3
DIR	4
SG	5
DSR	6
RS	7
CS	8
---	9

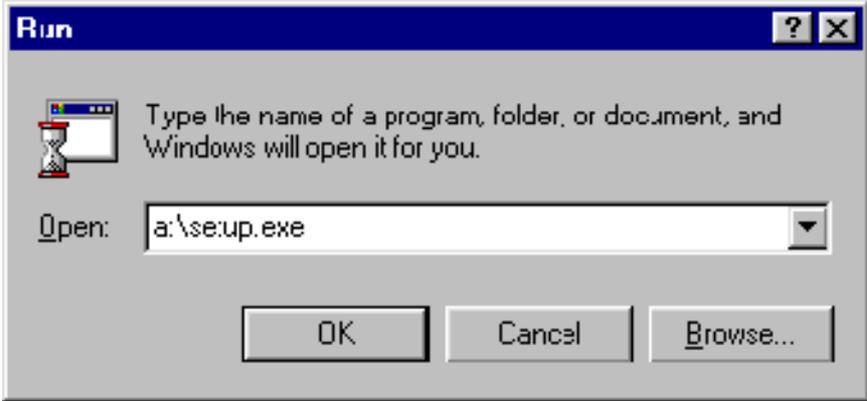
PLC CPM1A

Pin	Signal
1	FG
2	SD
3	RD
4	RS
5	CS
6	---
7	---
8	---
9	SG

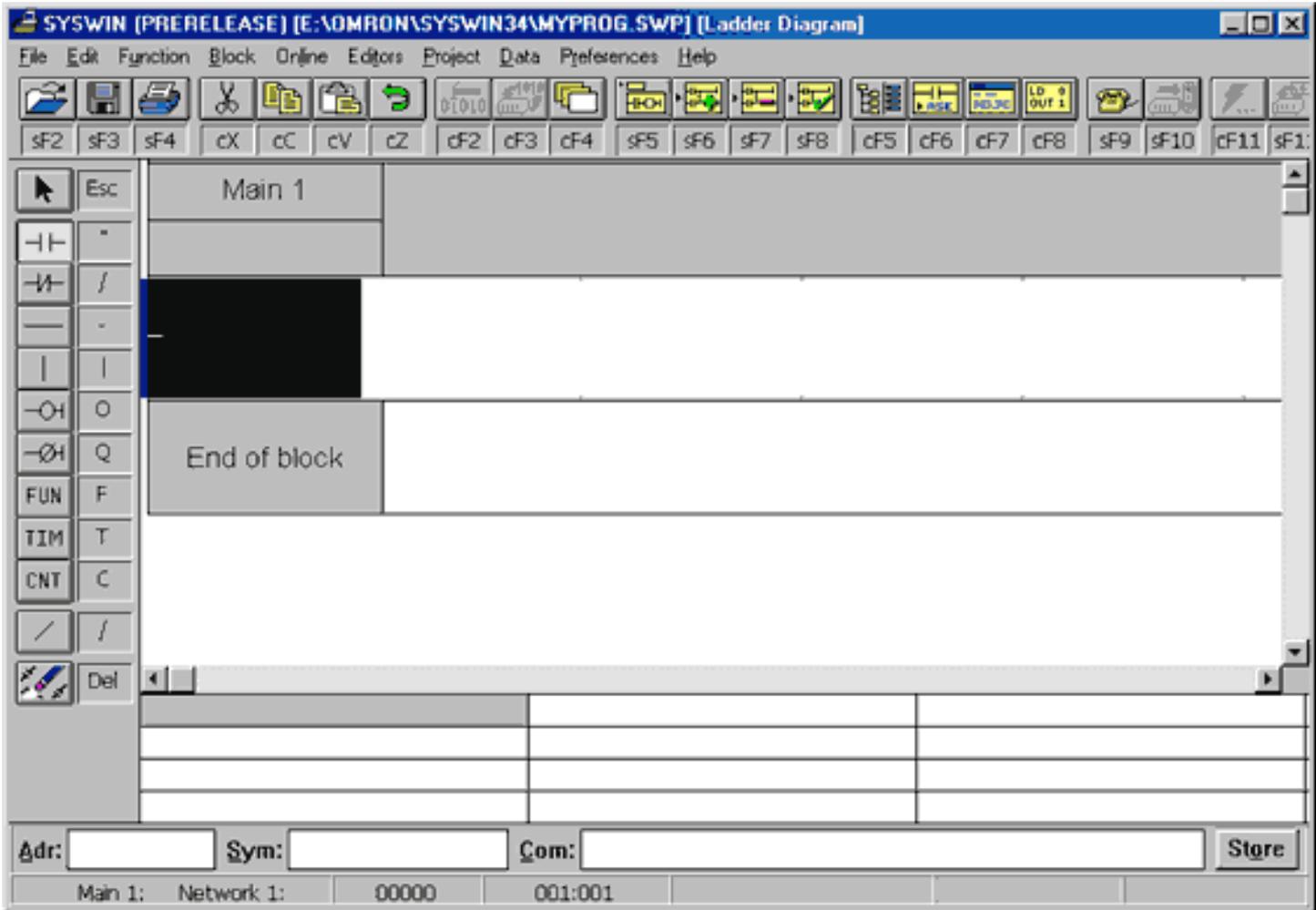


6.2 SYSWIN program installation

Instruction package for CPM1A is covered by three SYSWIN installation diskettes. It can be installed in Windows 3.1, 3.11, 95, 98 or NT 4.0. In order to start the installation you need to select RUN option from a START menu.

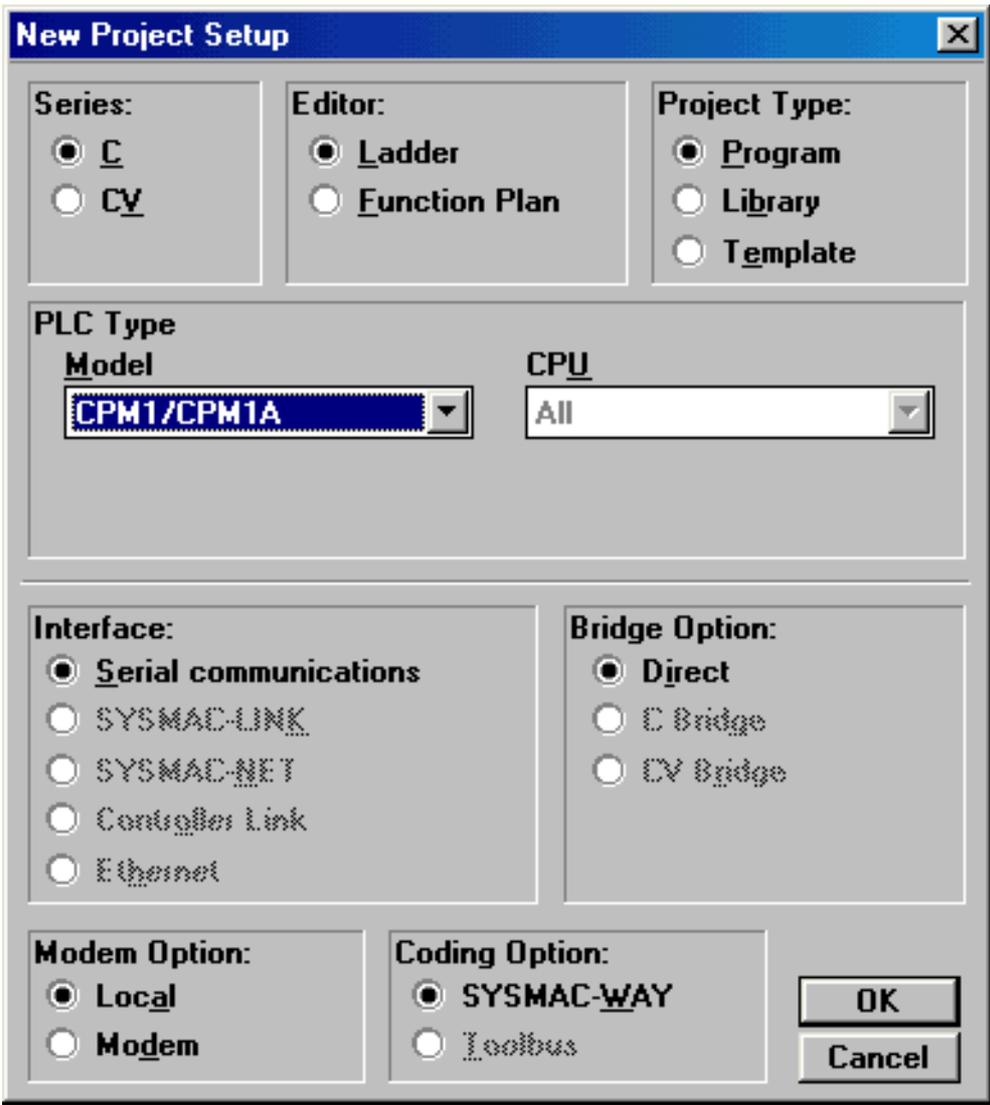


A window will come up like the one below where you need to write in the file command "setup.exe". Mentioned file can be found in the installation directory of Syswin program. Following a brief installation procedure you will get a program group Syswin 3.4. Double-click on Syswin icon starts a Syswin program which opens as in the following picture.



6.3 Writing your first program

Writing a program begins with New Project option from a File menu. In a message window that appears you need to select options as in picture below.

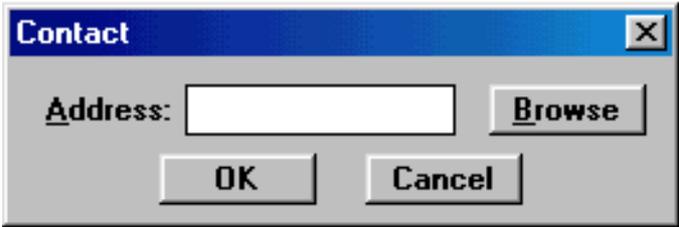


Select a PLC controller by clicking on OK, and a program is ready to be used. It is recommended when you begin working that you write in a header a title of a program, author's name and inputs/outputs used. This may seem as a waste of time, but really isn't because this habit of writing comments will pay off in the future.

Program written here is just a basic program made for learning Syswin. Program can detect when a key has been pressed and can activate a relay at the PLC controller output. As long as the key is pressed down, a relay is active. Operation of a relay and a key can be followed via LED diodes on PLC controller housing. Writing a program begins with a click on the first icon to the left, recognized by two vertical lines. Icon beneath this one is similar to the first but for a slash. These two icons correspond with concepts normally open and normally closed contact which all instruction lines start with. You can select an option with an open contact by clicking on the first icon. When you click on the black rectangle to the right, a small window will appear where you need to write in the address of a bit a contact relates to.

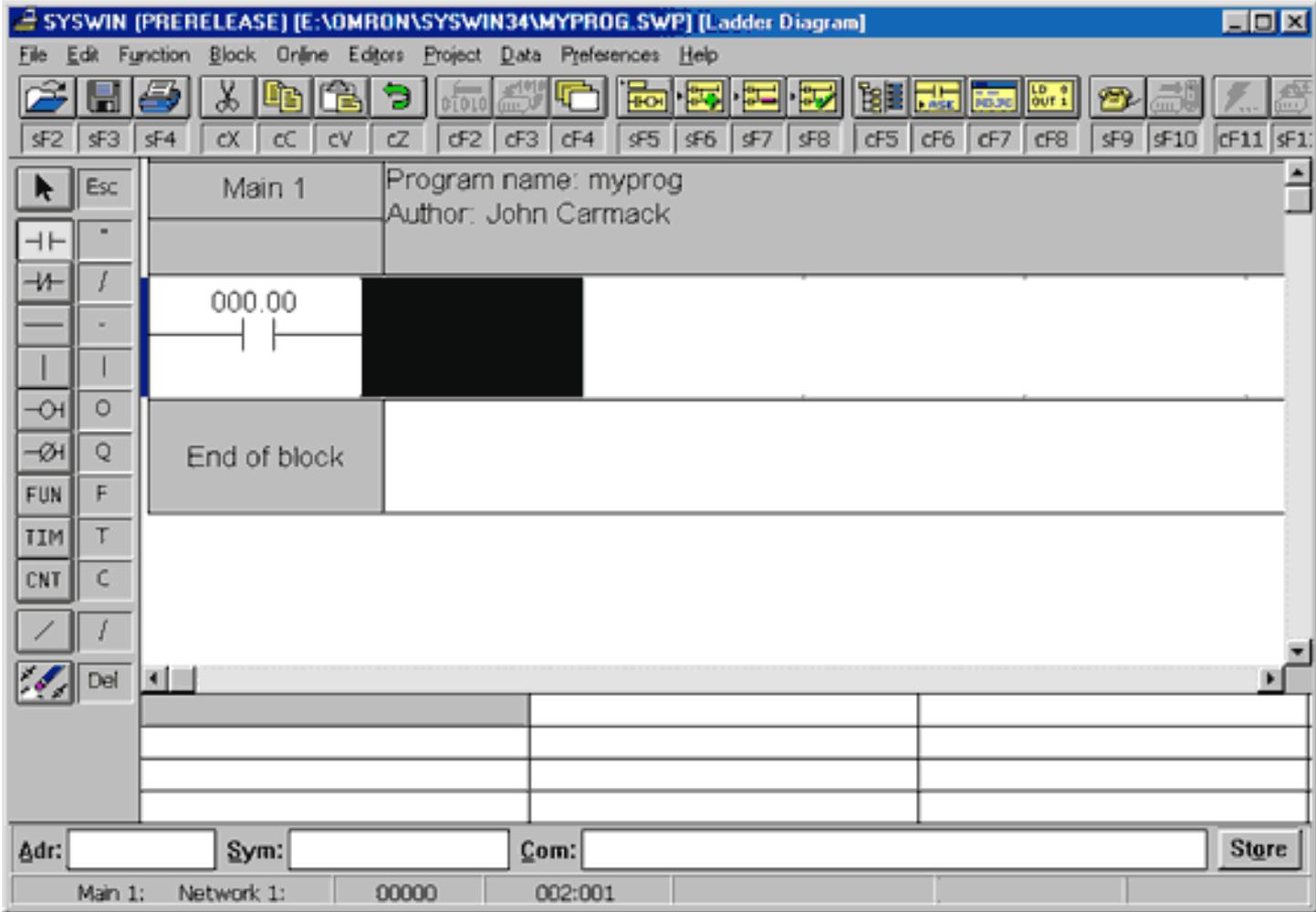
It is very important to use addresses in a regular way when programming with SYSWIN. Addresses can have two parts, first refers to the word address, and the second to bit address in that word (both numbers must be separated by a period). For example, if address 200 is used, SYSWIN will interpret this as 2.00, and a zero bit whose word address is 2 will be called for. If you wish to access word 200 or its zero bit, you must use a call 20000, or better even 200.00. In this example address 000.00 is assigned for input address (key). This address represents a zero bit for word 000 from memory region IR. Simply said, it is an input screw terminal designated as 00 input. By connecting a key to it, and to one of the COMM terminal screws, a needed connection

between PLC controller and keys is established.



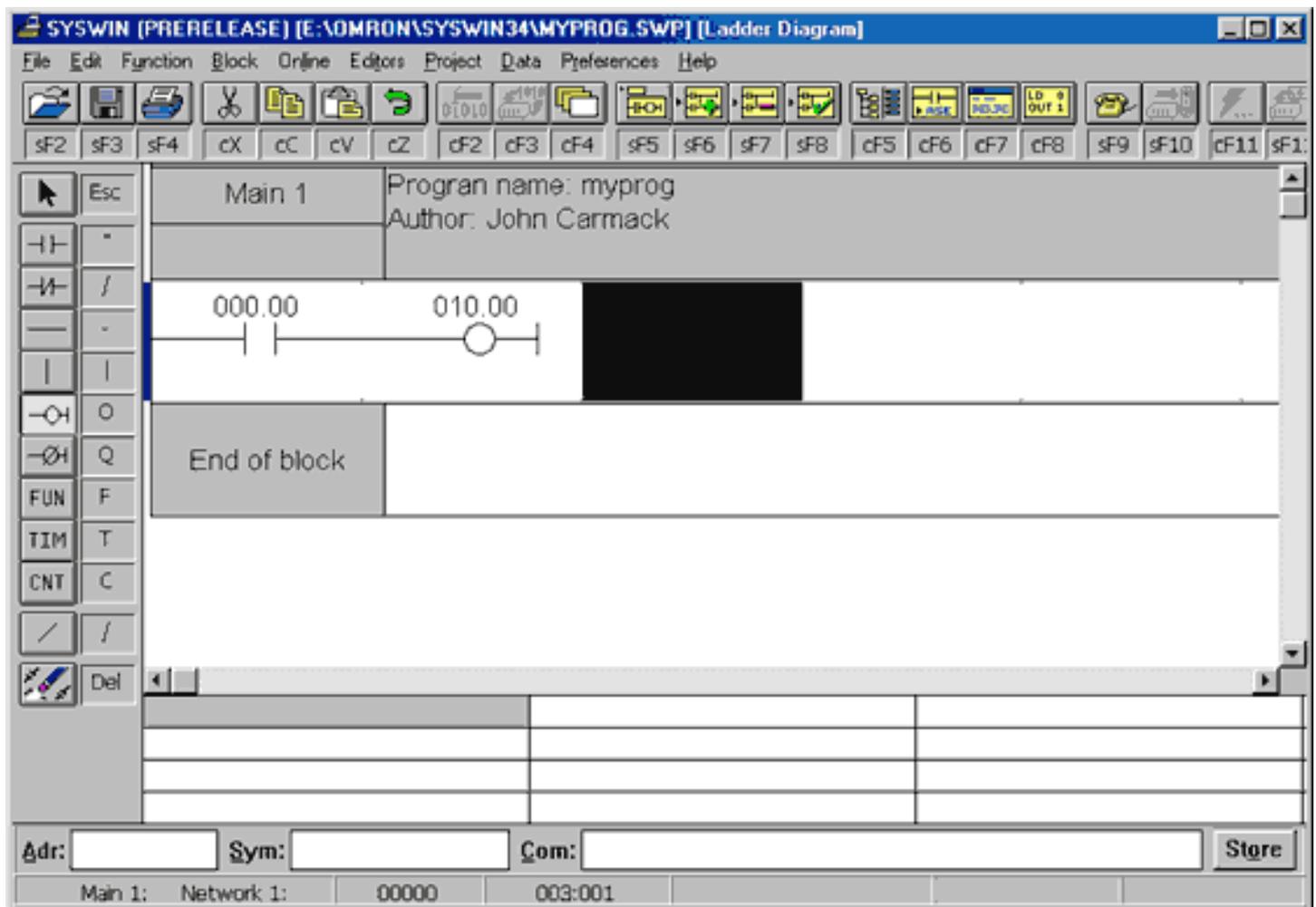
Address dialogue box for a bit that contact refers to

When you have written in 000.00, select OK, and first segment of the program will come up. Bit address will appear above the symbol with two vertical lines which refers to this bit, and a black rectangle will move one space to the right.



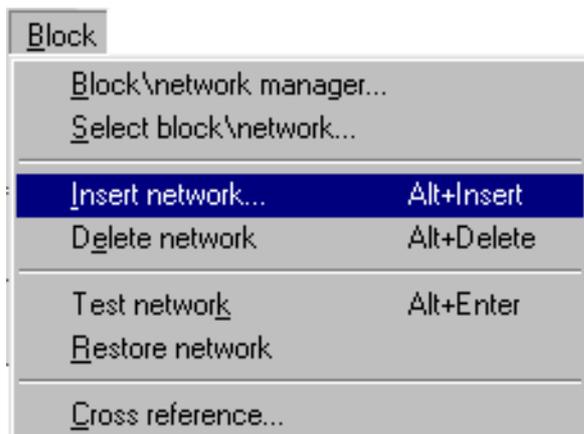
First element of a program myprog.swp

First instructions up to the bus bar are called conditions because their execution activates instructions found to the right of the condition instructions. When a condition is entered, you also need to enter a corresponding instruction that is set off by an execution of the condition. In this example it is a relay controlled by a 00 bit in a word 010 of memory region IR. Output instructions are represented by a circle, or a circle and a line if we are dealing with a normally closed contact. By clicking on the icon with a circle, you select an output option with normally open contacts. Click on a black rectangle, and a contact window will come up where you need to write in the address for the output bit 010.00. Output of the IR region is found at address IR010, and first four bits of this word represent a relay within a PLC controller (if we are talking about a model CPM1A with relay outputs). Program done so far looks as in picture below.

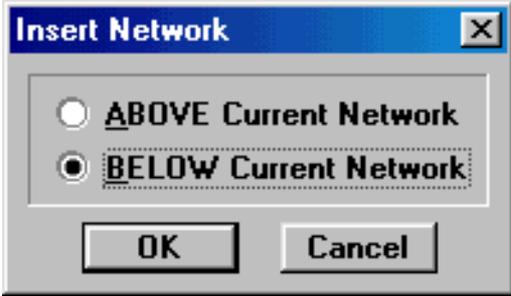


Second element of myprog.swp program

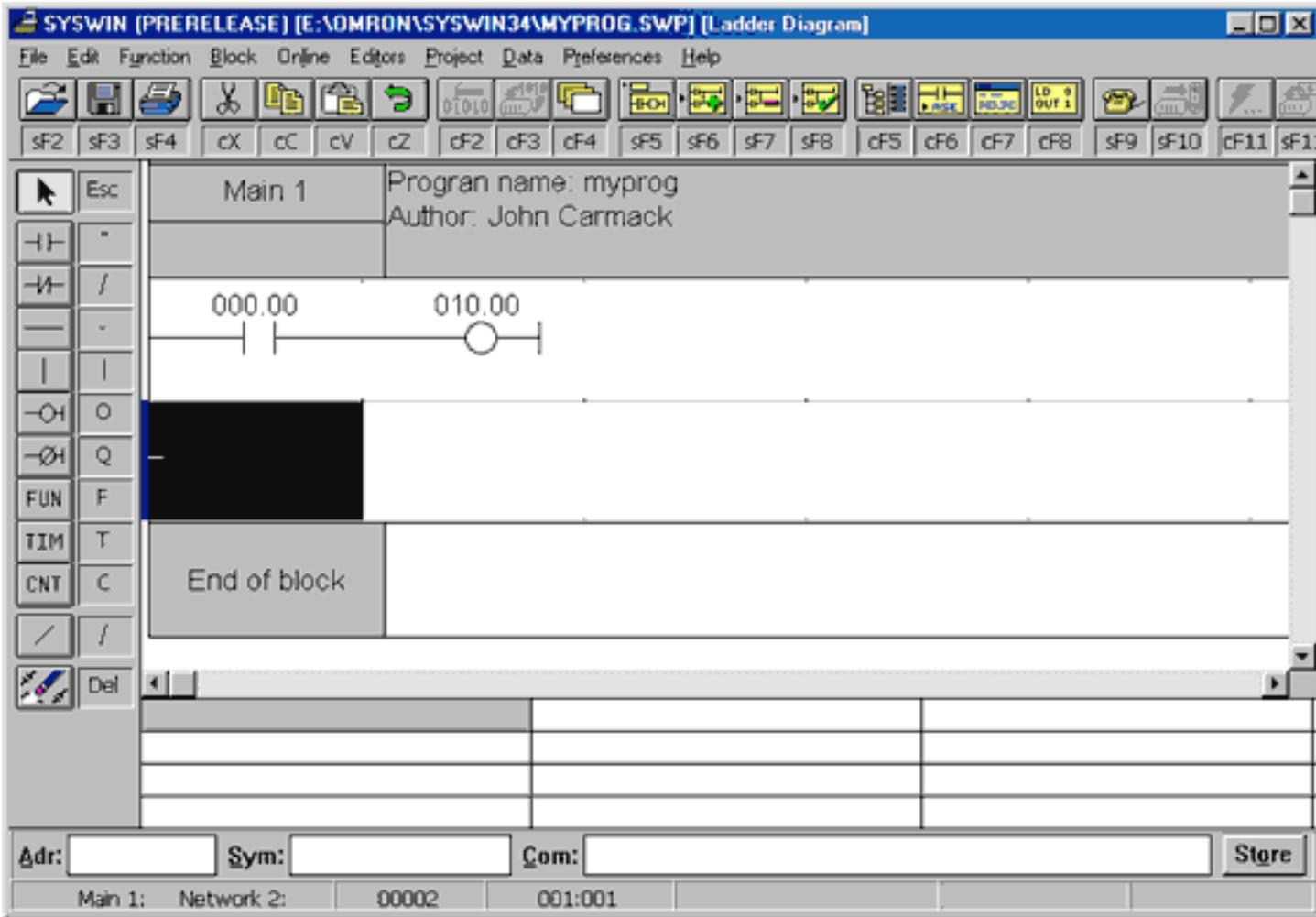
The basic functional entirety of some program is *Network*. Program consists of several networks found one below the other. Operations with these are found in Block option of the menu. Of all options, two basic ones, Insert network and Delete network are used the most. Other makers for PLC controllers use different concepts such as Rung instead of the term Network. Simply said, we are talking about a PLC program sequence which has one or more executing instructions, and along with END instruction can make up one correct PLC program. As the first network in a program is already in use, the next one has to be added. Adding a Network is done with Insert network command from a Block menu.



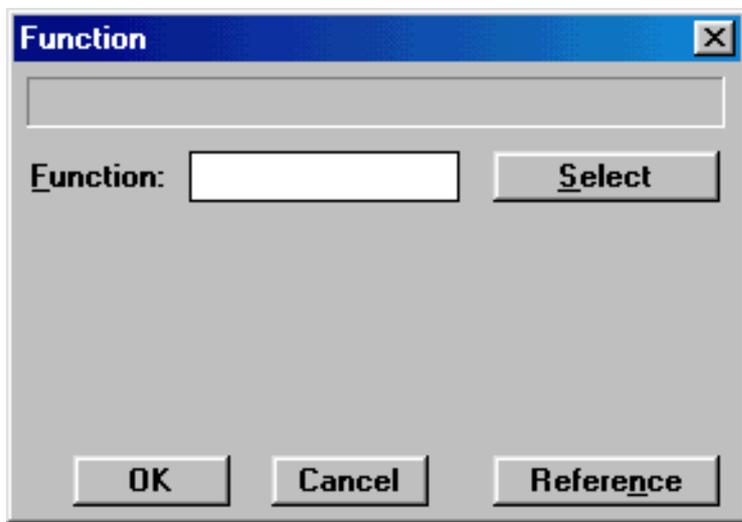
When selecting this option, a small window appears where you need to select whether a new network will appear above or below the existing one.



In our case you should choose the second option and click on OK. Following this, a new network appears as in picture below.

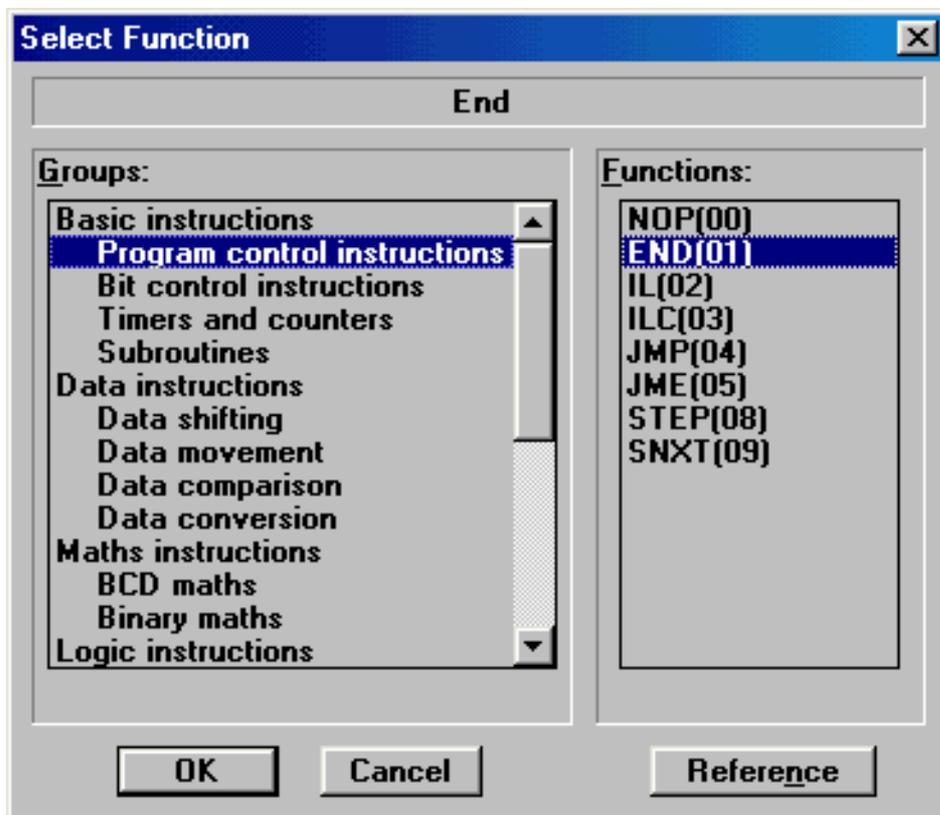


Last network in every program must contain END instruction. Since this is a simple example, second network is also the last. End instruction is found among the functions. In order to come to it, you need to click on FUN icon following which a window as in picture below will come up.



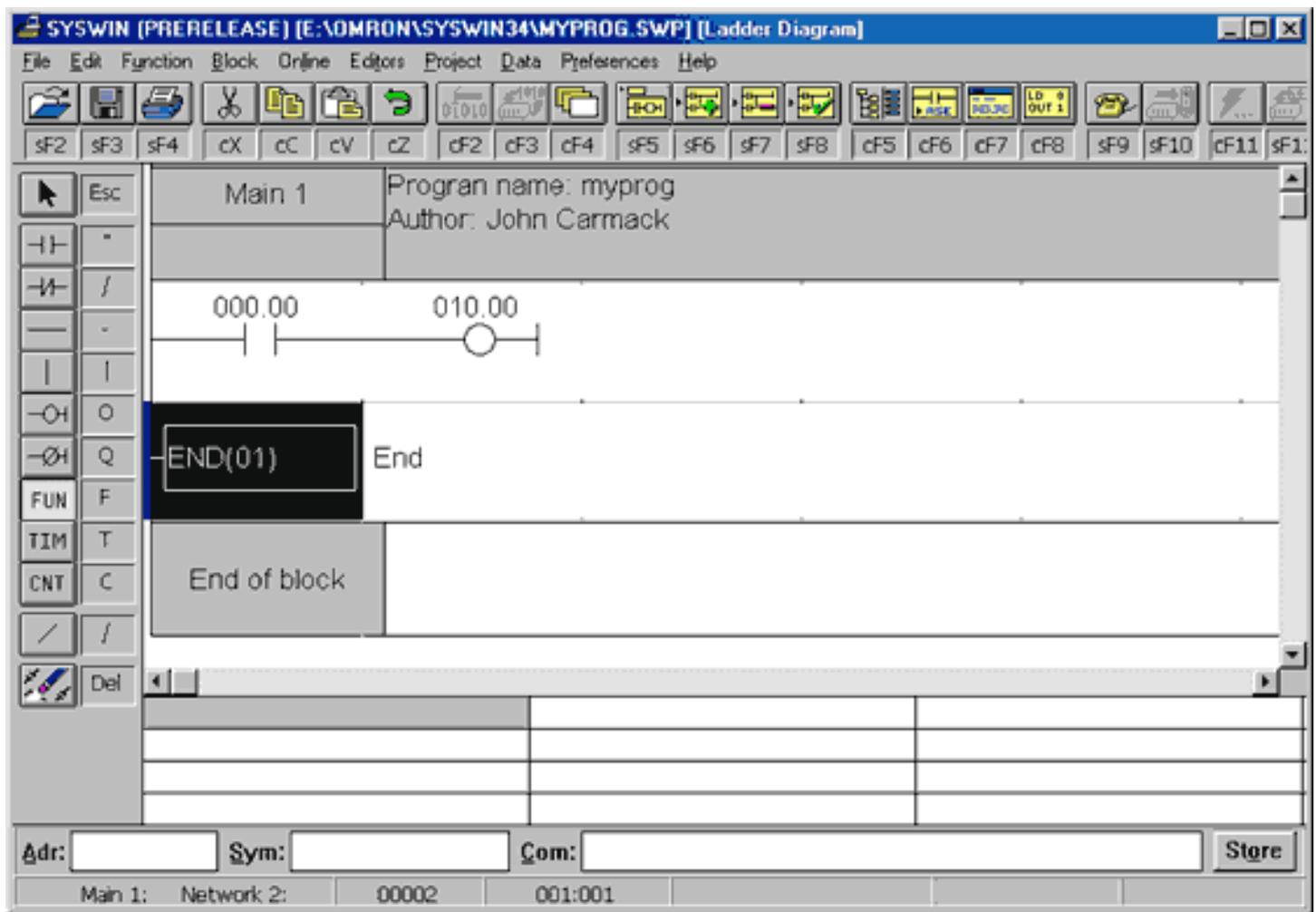
Selecting a function by clicking on FUN icon.

END instruction can be obtained either by writing in "END" in newly obtained window or by clicking on Select which gives all PLC controller instructions sorted by the regions as in the following picture.



Selecting END instruction from a set of instructions sorted in regions.

By entering the END instruction your writing of a program is finished. Finished program looks as in the following picture.



Finished myprog.swp program

6.4 Saving a project

Since you've finished writing a program, you need to save a project. Select Save Project option from a File menu, and write in the file name in a message window (myprog.swp in this case). After you click on OK, project will be saved. You can access SYSWIN file contents only from SYSWIN; file type is identified by extension:

Project.swp - SYSWIN program

Project.swl- SYSWIN library

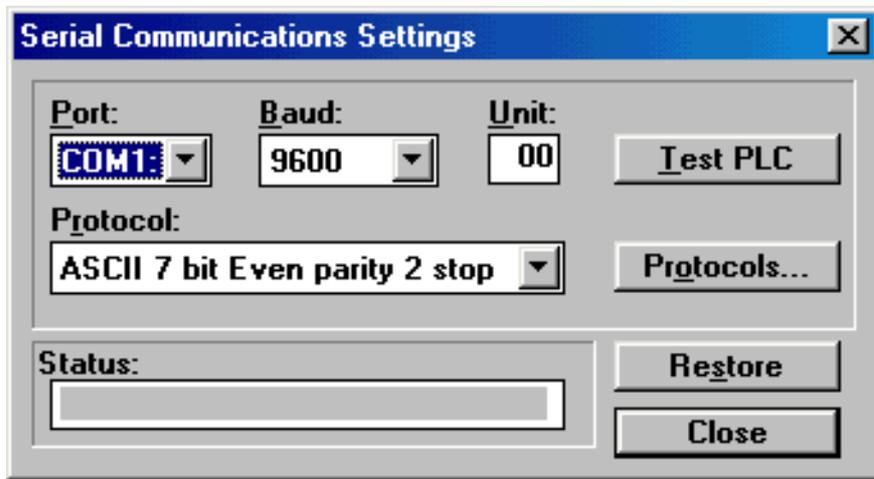
Project.swt - SYSWIN pattern

Project.swb - SYSWIN back-up file

Project.prg - PMD program

6.5 Program transfer to PLC controller

First you need to check whether PLC is connected with a PC correctly, and you'll do this by checking physical connection through a serial cable. Following this you need to select a Communication option from Project menu in order to set parameters for serial communication. Of all the parameters, the most important one to be selected is a serial port of a computer that PLC is connected to. Default settings for CPM1A are: COM1, 9600 Baud, Unit 00, protocol ASCII 7 bit Even Parity 2 stop and they need to be left so. To check how communication functions, you can click on Test PLC to test link with a PLC controller.



When a connection has been established, program transfer begins with a click on download from Online menu. Select expansion function or memory allocation. Before you program a PLC, it's good to erase program's memory contents. Finally, after a successful program transfer to a PLC, a message window will come up to inform us of this.

6.6 Testing program function

Program check option from a Project menu allows testing of program function. Message that appears following a command has several options that can be selected before you run a test. Once these options have been selected, click on Execute, and a report on testing and errors will be displayed. You can further check for errors, and there is also a 'Go to Network' command which transfers you to a segment where the error was found.

SYSWIN has classic editorial capabilities, such as Edit/Find or Edit/Replace commands. Searching through a program for assigned values or symbols is quick and offers a large number of optional filters. We can search through an entire program or its segments, and this is defined with option call. Also, there are possibilities for defining a search path, as well as for different actions when looking for a desired element.

Beside this, SYSWIN provides various advantages in situations where we need permanent archiving of user program. It is especially important to periodically print projects that are made quick and easy by SYSWIN. Projects can be printed in many different formats, and printing can include specific sections of a project.

6.7 Meaning of "Tools" icons

SYSWIN has several types of editors among whom a relay diagram also known as relay editor, or first editor that awaits us upon starting a SYSWIN program is the most frequently used editor.

First we need to explain tools palette (Drawing Tools) and the meaning of each icon. Aside through the usual mouse click, you can access the specific elements of this palette from a keyboard. You'll find a corresponding key of the keyboard by each icon, and you can accomplish the same action with it as you would using a mouse.



By clicking on the icon, we have selected a desired tool, and with a click on network section this symbol will be stored in a program. Explanation for each of the icons is given as follows:



Open contact icon. By clicking on this icon (or using a key '"') we enter an open contact into Network. We need to position the element we have entered at a specified place (black space). Following this, a message window where data can be entered (open contact address-number of words, bit position) is activated automatically.



Closed contact icon. By clicking on this icon (or '/' on keyboard) we enter a closed contact or inverted condition into network.



Horizontal line. By clicking on this icon (or using '-' on a keyboard), horizontal line is lengthened out from left to right. SYSWIN, however, retains a right to make drawn lines optimal in terms of length, or to point out possible errors. This option is used when you need to add another condition before an instruction contingent upon this condition, or when something simply can not fit.



Vertical line. With a click on this icon, or use of '?', we draw vertical lines from top to bottom. This option is necessary to realize parallel connections between contacts.



Output instruction. This represents an instruction that is executed if condition instruction preceding it is executed. With the help of this instruction we advance a result of logical expression with output variables (bits). We can arrive to this instruction with the help of keyboard ('O' key).

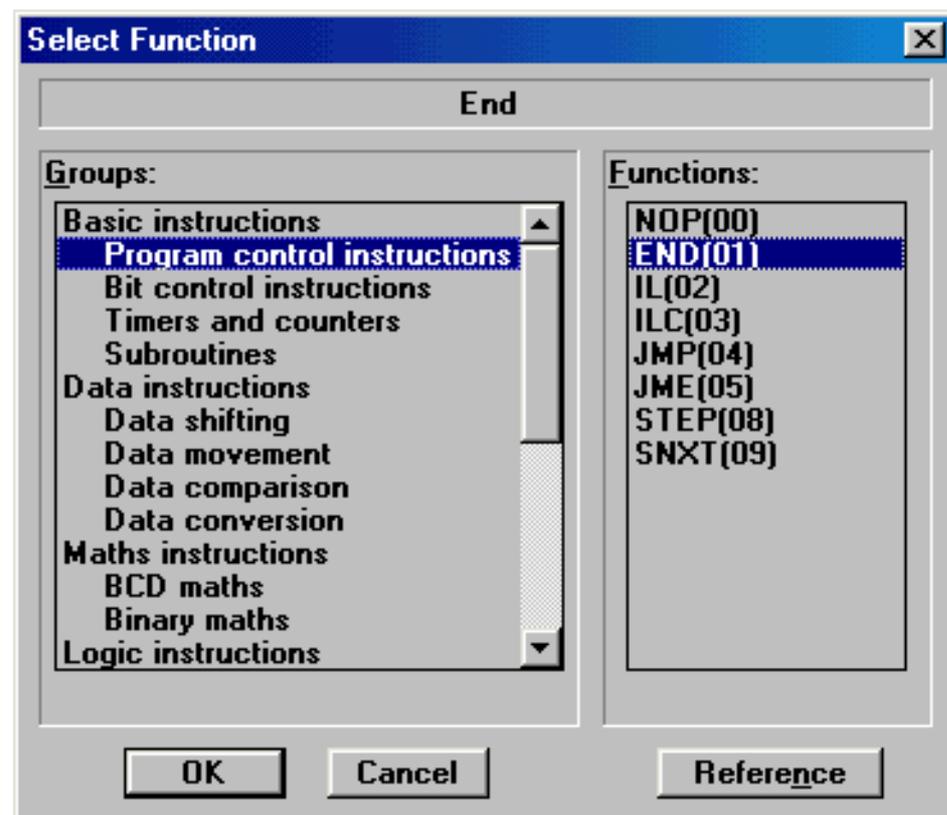


Inverted output instruction (shortcut-key 'Q'). Similarly to the previous case, with this executing instruction we advance a result of logical expression to an output bit, and the only difference is that this bit is turned on if a condition is not executed and vice versa.



PLC functions (shortcut-key 'F'). Click on this icon accomplishes possibility of installment of complex PLC instructions into a program. Window that appears following a click on the icon contains all instructions sorted by sections. Some of these instructions are given separately as

icons, and some can be accessed only through this option. One such instruction is END instruction which is used in each program. Window that comes up is displayed in the following picture.



When this window pops up, select an instruction and click on OK.



Click on this icon (or using 'T' key) will give you an option to enter a timer into the program. Using a mouse, click on the bright area of the monitor, and a message window comes up where you can enter needed information relating to a timer (timer designation and duration in milliseconds). This way, we get a classic timer or timer with a delay when turned on. If some other version of a timer is needed, preceding FUN icon should be used, and option Timers and counters (see picture above) selected.



Counter icon. Click on this icon (or 'C' key), and this will install a classic counter into a PLC program. Prior to this we enter needed information in message window: designation of the counter (CNT001 for instance) and counter value. Change of counter status (decrementing by 1) is done when an input signal (CP) changes from OFF to ON status.



With this icon we can invert previously entered contact, output or input. Inversion is done so that we first click on this icon, and then on a variable whose inversion we wish to perform.



Erase icon. Click on this icon and a shaded area of network erases the shaded part of the program.

Mouse plays an important part in the SYSWIN program. Each double-click on any PLC instruction results in a corresponding editor where necessary changes can be entered. This principle is accordingly installed into SYSWIN, so double-click on block or network heading (BLOCK HEADER BAR or NETWORK HEADER BAR) gives the same results.

6.8 PLC controller working modes

There are several ways to find out the present working mode, for example from an Online Mode menu or its display in a Toolbar. This option is accessible if communication was successfully established with a PLC controller.

If we choose a mode that differs from a present one, change of mode will be momentary. In order to avoid an accidental change of PLC controller mode, there is an option that obliges a computer to ask a question before each mode change, whether that is the option a user really wants (this option is included as default). PLC controller has three modes in class C, MONITOR, RUN and PROGRAM/STOP mode.

6.9 RUN MODE

This PLC mode enables program to be executed as basic operation. It is used in final testing, after a program has been tested in detail, and errors have been eliminated. SYSWIN can not change memory contents of PLC controller in this mode, neither is the change of a program being executed possible. Of course, when program is finished and tested, PLC begins its new life in command closet, being first set to RUN mode.

6.10 Monitor mode

In this mode, program execution is possible, as well as editing and monitoring during operation. This is the most frequently used mode in program development. When this mode has been selected, controller has an obligation to supply a PC with information which relates to program itself, or more precisely to status of variables in the program. If we additionally confirm Monitoring option from an Online menu, we can follow current values of variables on the monitor itself, in real time.

All changes in inputs and outputs can be viewed on the monitor, and status of variables and program locations used in the program are registered and memorized.

6.11 Program-Stop mode

Choosing this mode simply stops a PLC controller if PLC was in RUN or MONITOR mode. It is used for data and program transfer to PLC controller.

6.12 Program execution and monitoring

Program transferred from a PC to a PLC starts executing at the moment when you move from a Stop/Program mode to a Monitor or Run mode. When Monitoring function starts executing, some sections of the monitor will be shaded (see picture above), and this way you can follow program execution. Monitoring is active during editing of some program segment, and is stopped at the moment when a changed section of the program is transferred into a PLC controller.

6.13 Impact on the program during monitoring

During monitoring, you can use the right button on the mouse to call up a menu of some elements of ladder diagram. Menu that appears when we click on location where address of some bit is positioned, contains the following elements:

Force Set - used for permanent forced set up of bit status to ON

Force Reset - used for permanent forced set up of bit status to OFF

Cancel - cancels out the forced status

Set (1) - used for a brief change of bit status from OFF to ON status

Reset (0) - used for a brief change of bit status from ON to OFF status

Cancel All - cancels out forced status of all bits

With the help of these options, status of bits can be changed, word contents in controller memory can also be changed, and all or some of the earlier forced settings can be cancelled out. The concept of forcing entails forced set up of some input/output to ON or OFF status for program

reevaluation. At the moment when PLC leaves monitoring regime, data on forced bits and words is lost. Simultaneous forcing and evaluation of contents of a greater number of dimensions, and Data Set Bar is used for this, usually found at the bottom of the monitor (see previous picture). Editing as well as defining an area in Data Set Bar is accomplished with a double click following which a corresponding message window appears, and we write here address for the bit whose status we are following.

6.14 Graphic representation of dimension changes in a program

SYSWIN allows graphic representation of dimensions with time as abscissa. When a monitoring mode is in use, monitor display changes through time, showing changes in values of monitored dimensions. Monitor refreshment is done after a reception of each sample where sample intervals are 0,1-65.5 seconds. Graphics saved in this way can be stored for later analysis as a file, or read in an already saved file.

Procedure for starting graphic monitoring is following:

- from Editors menu select Time chart monitoring option
- from a new tools palette select Trace Configure (pictured as a key).
- Fill out message box Configure Time Chart Monitor (see next picture)
- From Online menu choose Tracing.

With Trace/Configure command adjust parameters for monitoring. Necessary parameters are Trigger or an event where saving will begin, sampling period and bits and/or words whose values we are monitoring. Mapping of a time diagram for dimensions previously specified begins after the last command.

Quitting is done with a click on a black square icon, and restarting is performed by clicking on an a red circle icon. Return to the editor with a ladder diagram by clicking on Editors menu and Program editor submenu.

CHAPTER 7 EXAMPLES

[Introduction](#)

[7.1 Self-maintenance](#)

[7.2 Making large time intervals](#)

[7.3 Delays of ON and OFF status](#)

[7.4 Counter over 9999](#)

[7.5 Alternate ON-OFF output](#)

[7.6 Automation of parking garage](#)

[7.7 Operating a charge and discharge process](#)

[7.8 Automation of product packaging](#)

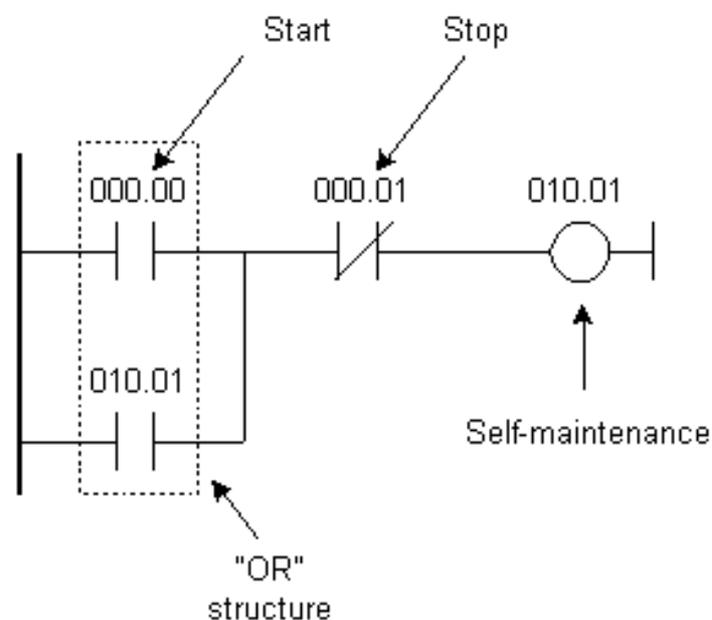
[7.9 Automation of storage door](#)

Introduction

Programming only related examples make up the first group of examples. They are given as separate small programs that can later be incorporated into larger ones. Second group consists of examples which can be applied to some real problems.

7.1 Self-maintenance

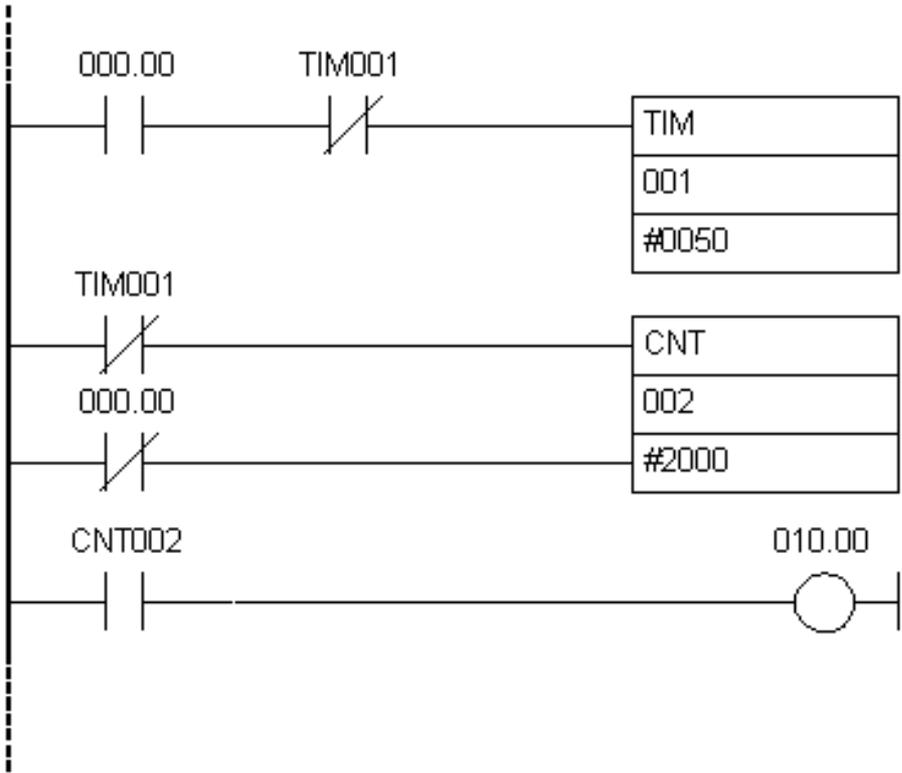
Program allows input to remain at ON status even when the condition that brought it to that status stops. Example in picture below illustrates how use of a key connected to the input IR000.00 changes IR010.01 output status to ON. By letting the key go, output IR010.01 is not reset. This is because IR010.01 output keeps itself at status ON through OR circuit (having IR000.00), and it stays in this status until key at input IR000.01 is pressed. Input IR000.01 is in I connection with the output pin IR010.01 which cancels out a condition, and resets an IR010.01 bit. Example of self-maintenance is quite frequent in specific applications. If a user was connected to IR010.01 output, START and STOP functions could be realized from two keys (without the use of switches). Specifically, input IR000.00 would be a START key, and IR000.01 would be a STOP key.



7.2 Making large time intervals

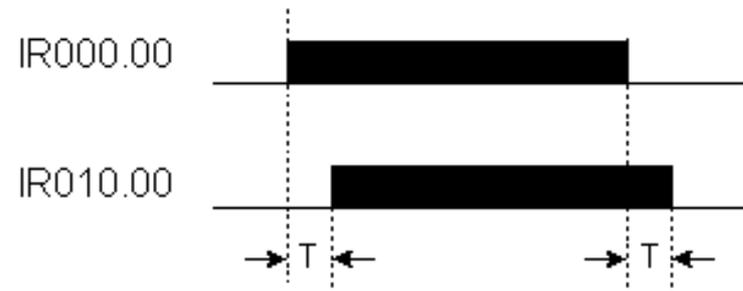
If it's necessary to make a bigger time interval of 999.9 seconds (9999x0.1s) two linked timers, or a timer and a counter can be used as in this example. Counter is set to count to 2000, and timer is set to 5 seconds which gives a time interval of 10.000 seconds or 2.77 hours. By executing a condition at IR000.00 input, timer begins to count. When it reaches the limit, it sets a flag TIM001 which interrupts the link and simultaneously resets a timer. Once 5 seconds have run out, flag TIM001 changes its status to ON and executes a condition at the counter input CNT002. When a counter numbers 2000 such changes in timer flag status, TIM001 sets its flag CNT002 which in turn executes a condition for IR010.00 to change status to ON. Time that has elapsed from the change of IR000.00 input status to ON and a change of IR010.00 input status to ON comes to 10.000 seconds.

Ladder Diagram:

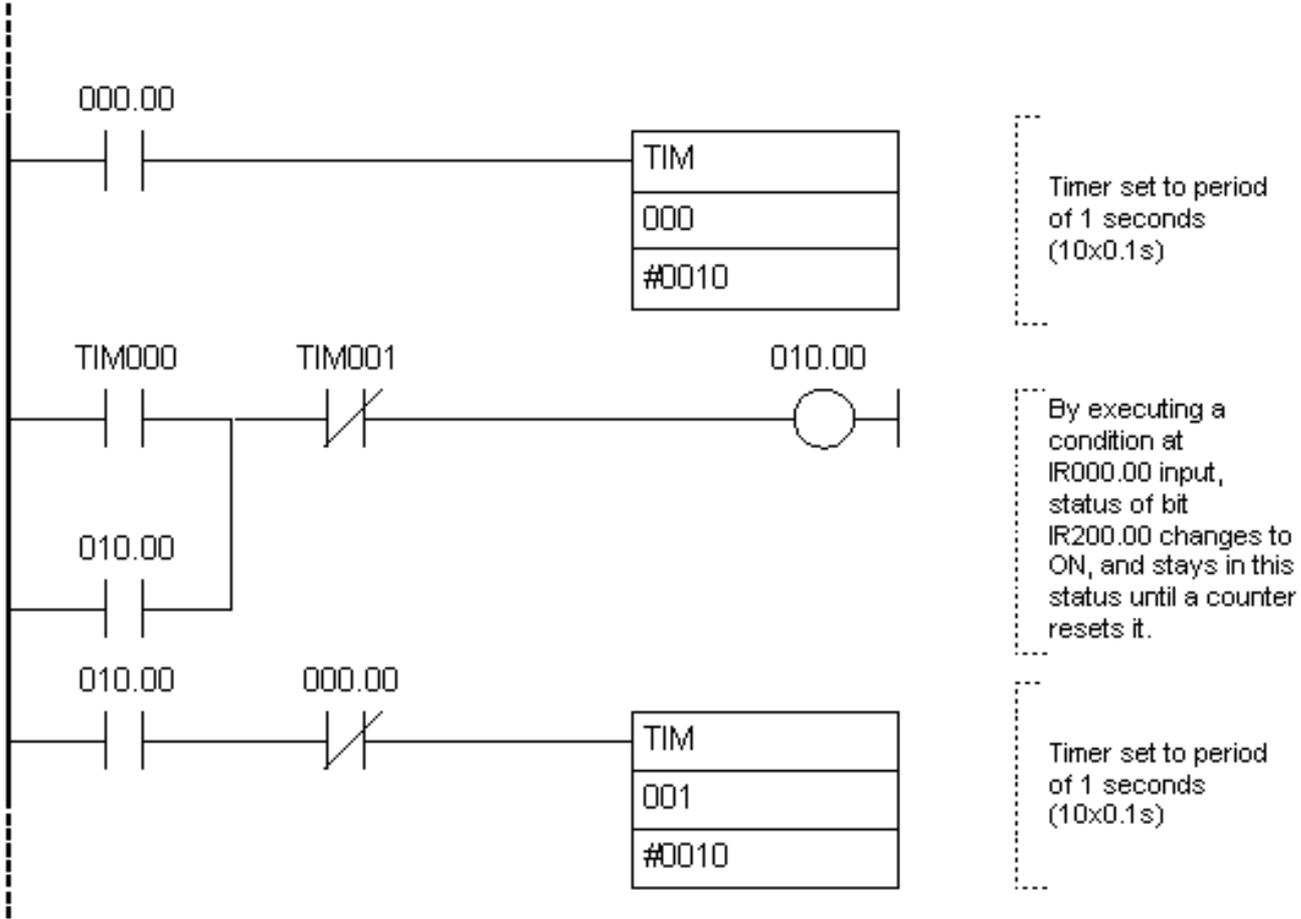


7.3 Delays of ON and OFF status

Example shows how to make output (IR010.00) delay as opposed to (in relation to ?? unclear meaning) input (IR000.00). By executing a condition at IR000.00 input, timer TIM000 begins counting a set value 10 in steps of 0.1 seconds each. After one second has elapsed, it set its flag TIM000 which is a condition in changing output status IR010.00 to ON. Thus we accomplish a delay of one second between ON status of IR000.00 input and ON status IR010.00 input. By changing IR010.00 output status to ON, half of the condition for activation of the second timer is executed. Second half of the timer is executed when IR000.00 input changes status to OFF (normally closed contact). Timer TIM001 sets its flag TIM001 after one second, and interrupts a condition for keeping an output in ON status.



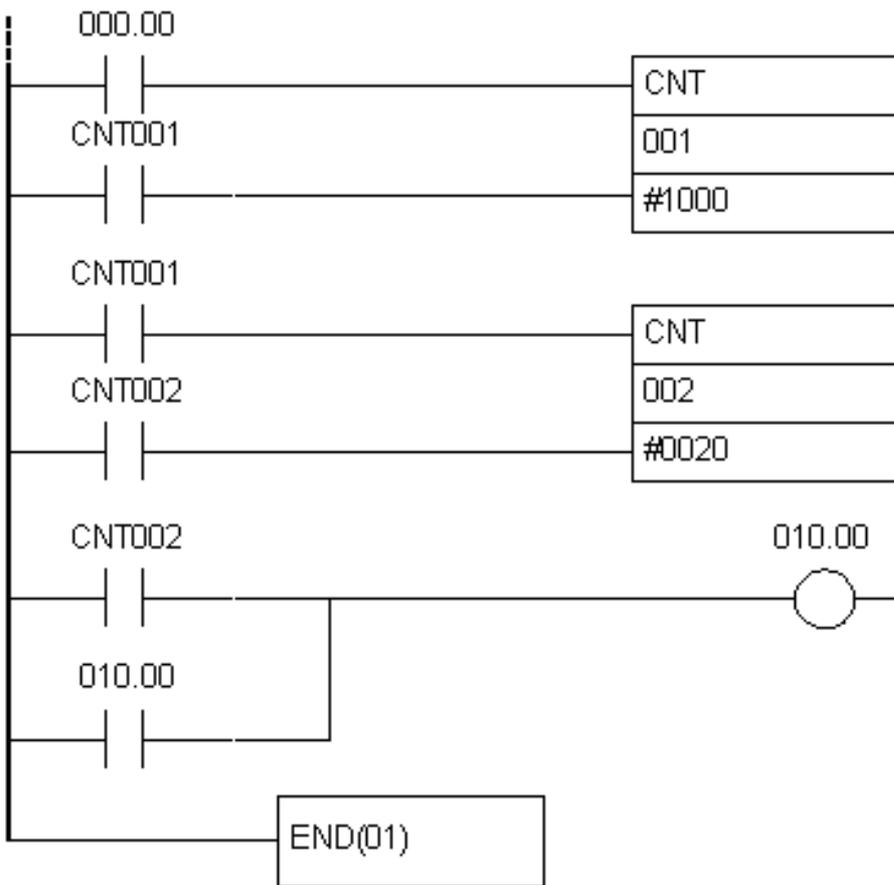
Ladder Diagram:



7.4 Counter over 9999

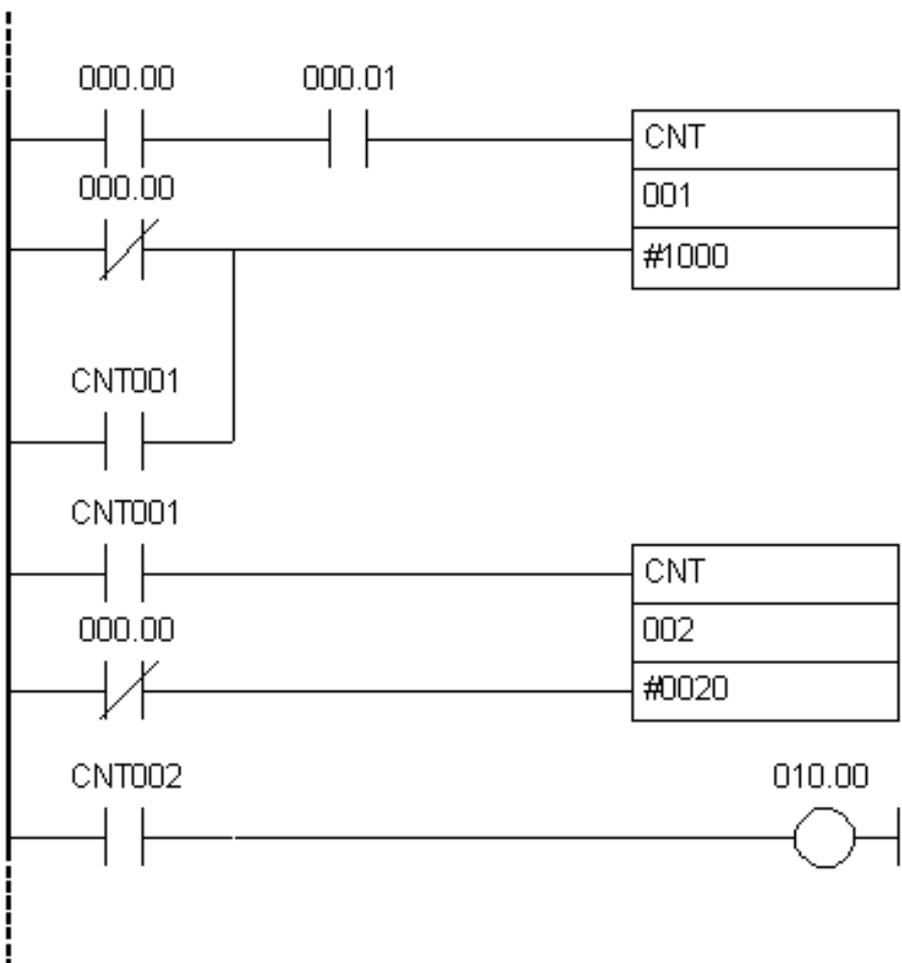
If you need to count over 9999 (maximum value for a counter), you can use two connected timers. First counter counts up to certain value, and the other one counts flag status changes of the first counter. Thus you get the possibility of counting up to a value which is a result of set values of the first and second counter. In an example at the bottom, first counter counts up to 1000, and second up to 20 which allows you to count to 20000. By executing a condition at IR000.00 input (line whose changes are followed is brought to it), first counter decreases its value by one. This is repeated until counter arrives at zero when it sets its flag CNT001 and simultaneously resets itself (is made ready for a new cycle of counting from 1000 to 0). Each setting of CNT001 influences the other counter which sets its flag after twenty settings of the first counter's flag. By setting CNT002 flag of the second counter, a condition is executed for an IR010.00 output to be activated and to stay in that status through self-maintenance.

Ladder Diagram:



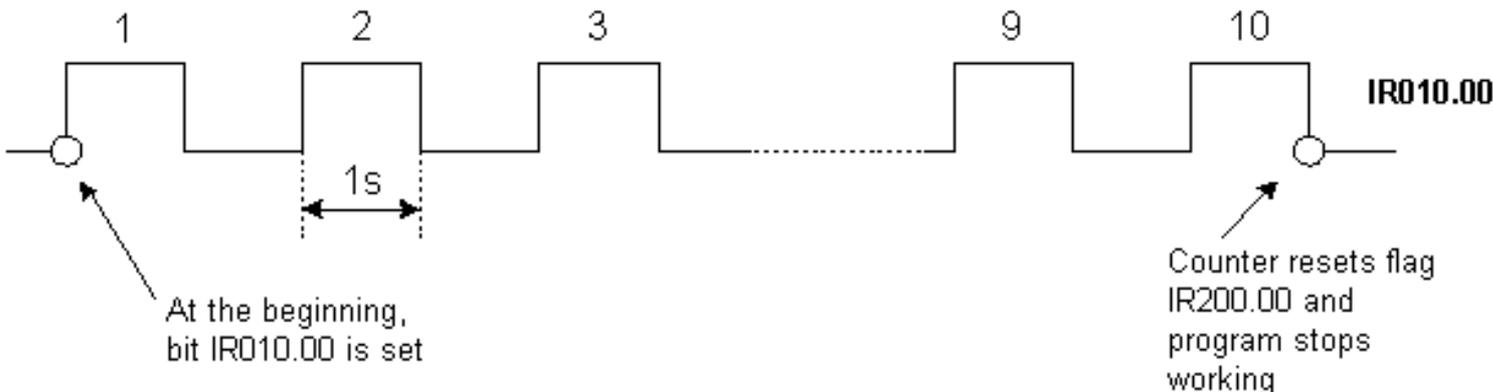
Same effect can be achieved with a modified program below. First change is that there is a "switch" for the whole program, and this is IR000.00 input (program can accomplish its function only while this switch is active). Second change is that the line whose status is followed is brought to IR000.01 input. The rest is the same as in the previous version of the program. Counter CNT002 counts status changes of the CNT001 counter flag. When it numbers them, it changes the status of its flag CNT002 which executes the condition for status change of IR010.00 output. This changes IR010.00 output status after 20000 changes of input IR000.01.

Ladder Diagram:



7.5 Alternate ON-OFF output

Example makes a certain number of impulses of desired duration at PLC controller IR010.00 output. Number of impulses is given in instruction of the counter (here it is a constant #0010 or ten impulses) impulse duration in two timer instructions. First timer defines duration of ON status, and second one duration of OFF status of IR010.00 output bit. In the example these two durations are the same, but through assigning them different parameters they can differ so that duration of ON status can be different from duration of OFF status.



Program starts executing a condition at $IR000.00$ bit. Since a normally closed contact which refers to counter flag (that isn't set) is linked with this $IR000.00$ bit in "I" circuit, this status of $IR200.00$ bit will change to ON. Bit $IR200.00$ keeps its status through self-maintenance until counter flag is not set and a condition interrupted.

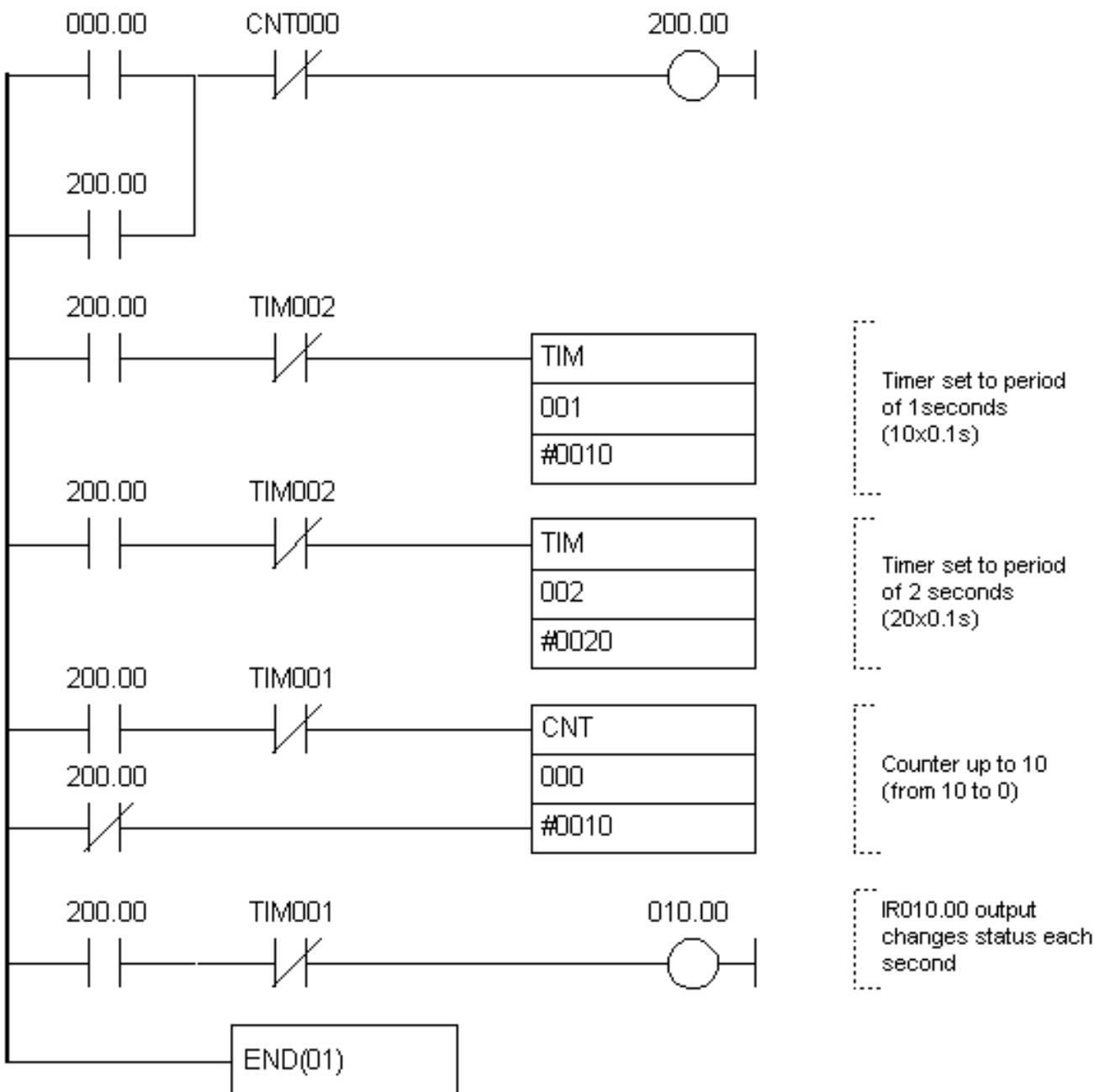
When an $IR200.00$ bit is set, timers $TIM001$ and $TIM002$ start counting a set interval number at 0.1 s (

in the example, this number is 10 for the first timer, or 20 for the second timer, and this sets the period of one or two seconds). With both timers, a normally closed contact which refers to TIM002 timer flag is connected with IR200.00 bit. When this flag is set which happens every two seconds, both timers are reset. Timer TIM002 resets timer TIM001 and itself, and this starts a new cycle.

At the start of a program, IR010.00 output bit changes status to ON and stays in this status until TIM001 flag changes status to ON (after one second). By changing TIM001 flag status to ON, condition is broken (because it is represented as normally closed contact) and IR010.00 bit changes status to OFF.

IR010.00 output status changes to ON again when time has run out on TIM002 timer. This resets TIM001 timer and its flag which in turn executes a condition for status change of the IR010.00 output. Cycle is thus repeated until a counter numbers 10 changes of TIM001 flag status. With the change of status of CNT000 counter flag, a condition for an assisting bit IR200.00 is broken, and program stops working.

Ladder Diagram:



CHAPTER 7 EXAMPLES

[Introduction](#)

[7.1 Self-maintenance](#)

[7.2 Making large time intervals](#)

[7.3 Delays of ON and OFF status](#)

[7.4 Counter over 9999](#)

[7.5 Alternate ON-OFF output](#)

[7.6 Automation of parking garage](#)

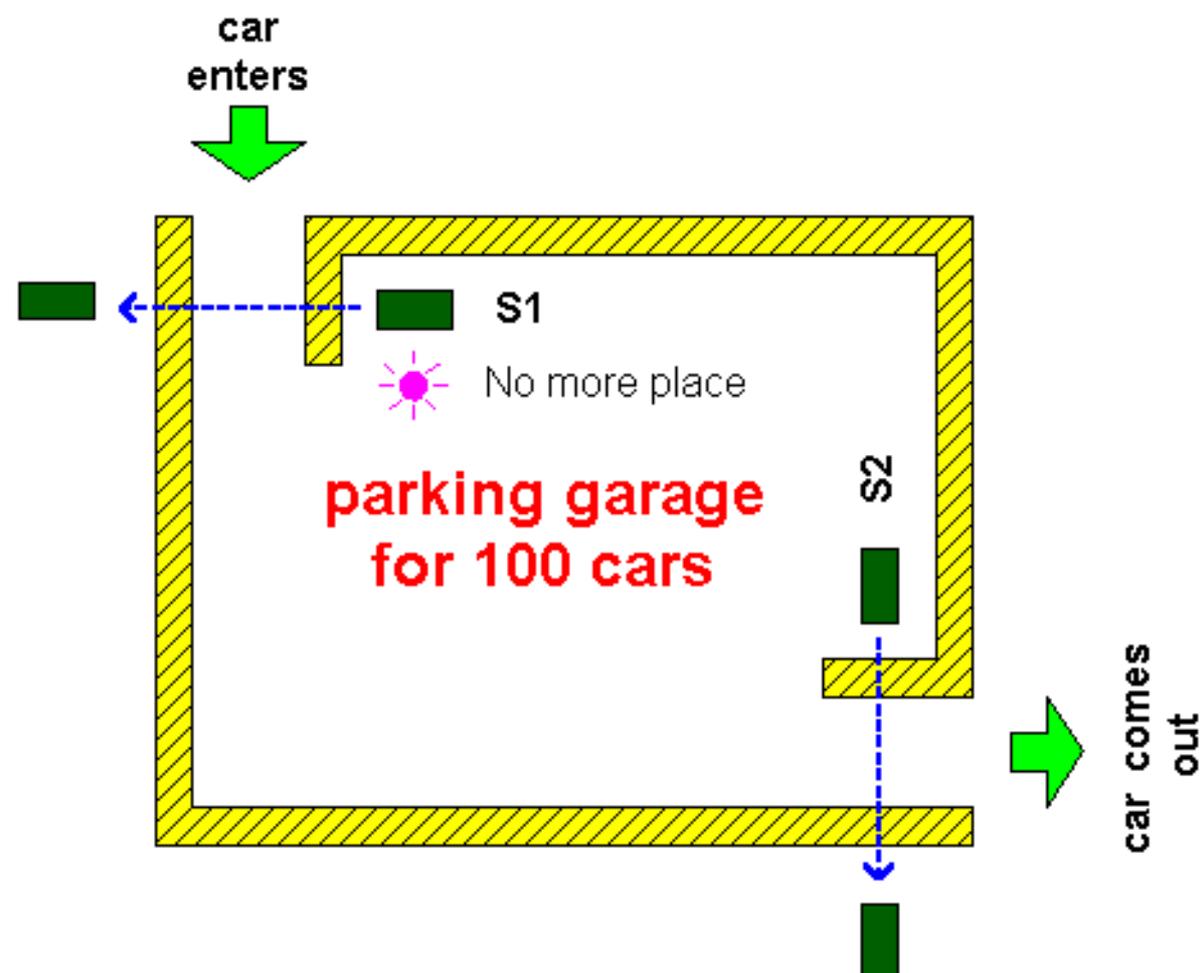
[7.7 Operating a charge and discharge process](#)

[7.8 Automation of product packaging](#)

[7.9 Automation of storage door](#)

7.6 Automation of parking garage

We are dealing with a simple system that can control 100 car at the maximum. Each time a car enters, PLC automatically adds it to a total sum of other cars found in the garage. Each car that comes out will automatically be taken off. When 100 cars park, a signal will turn on signaling that a garage is full and notifying other drivers not to enter because there is no space available.



Signal from a sensor at the garage entrance sets bit IR200.00. This bit is a condition for execution

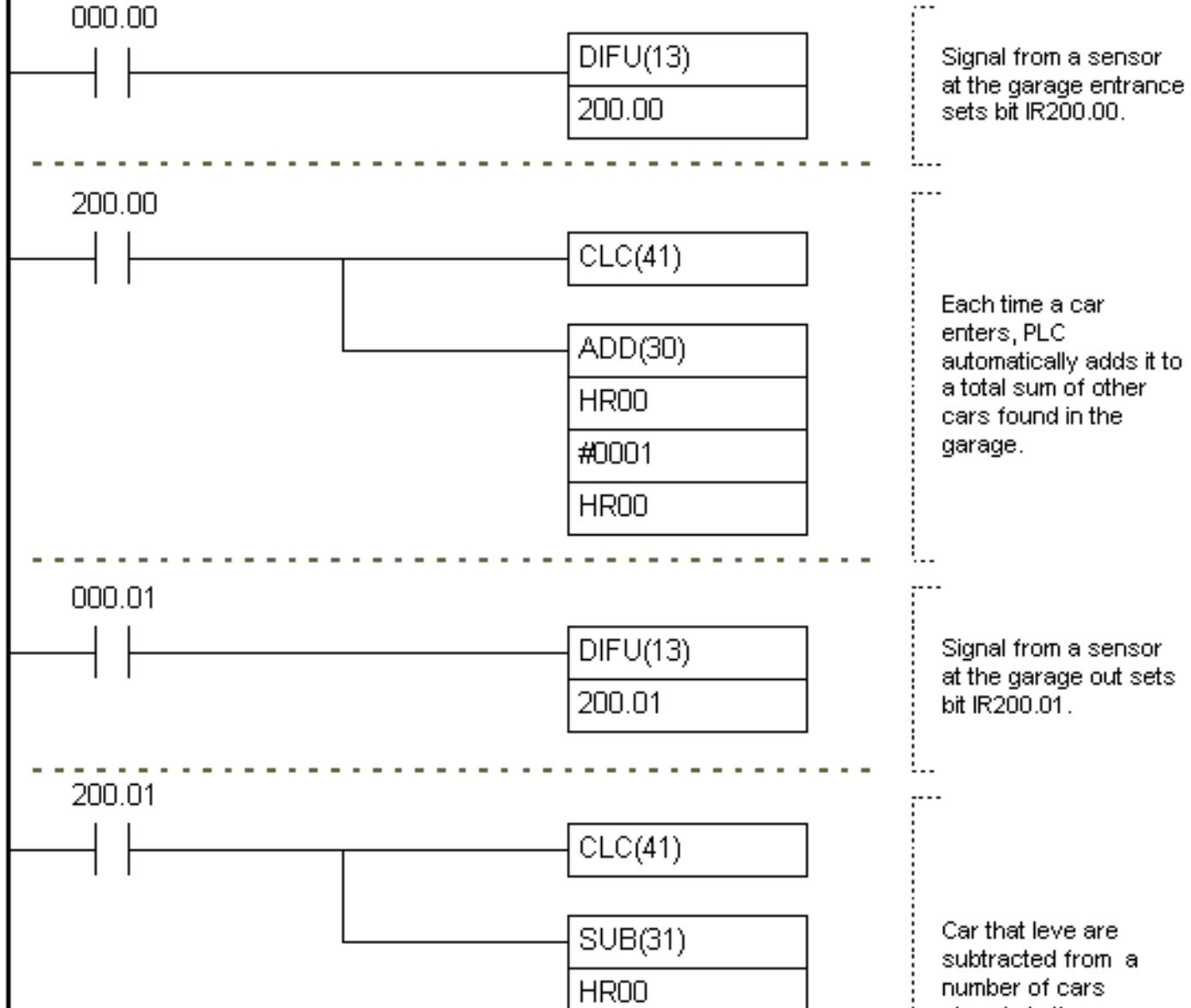
of the following two instructions in a program. First instruction resets carry bit CY (it is always done before some other calculation that would influence it), and the other instruction adds one to a number of cars in word HR00, and a sum total is again stored in HR00. HR memory space is selected for storing a total number of cars because this keeps the status even after supply stops.

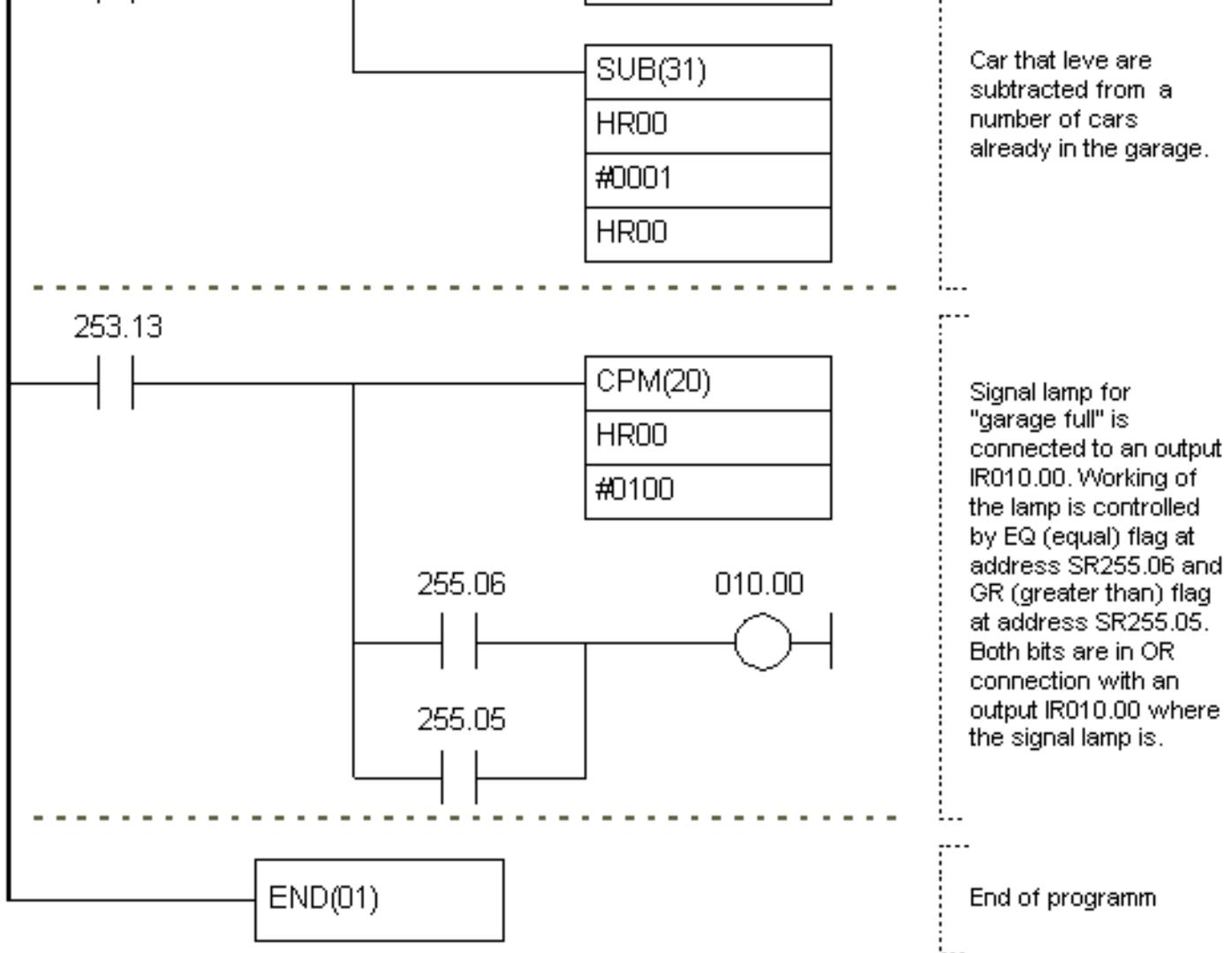
Symbol "#" in addition and subtraction instructions defines decimal constant that is being added or subtracted from a number of cars already in the garage. Condition for executing comparison instruction CPM is always executed because bit SR253.13 is always set; this practically means that comparison will be done in each cycle regardless whether car has entered or left the garage.

Signal lamp for "garage full" is connected to an output IR010.00. Working of the lamp is controlled by EQ (equal) flag at address SR255.06 and GR (greater than) flag at address SR255.05. Both bits are in OR connection with an output IR010.00 where the signal lamp is. This way lamp will emit light when a number of cars is greater than or equal to 100. Number of cars in a real setting can really be greater than 100 because some untrusting driver may decide to check whether there is any space left, and so a current number of cars can increase from a 100 to 101. When he leaves the garage, a number of cars goes down to 100 which is how many parking spots there are in fact.

Ladder diagram:

Main 1	IR000.00 Signal from a sensor No1 at the garage entrance
PARKING GARAGE	IR000.01 Signal from a sensor No2 at the garage out
	IR010.00 Signal lamp for "garage full"

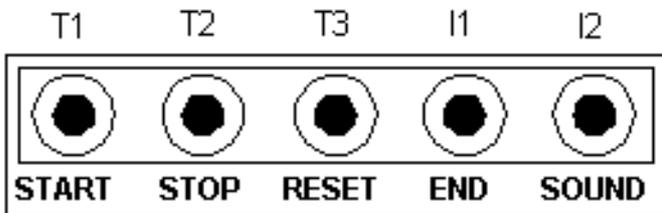
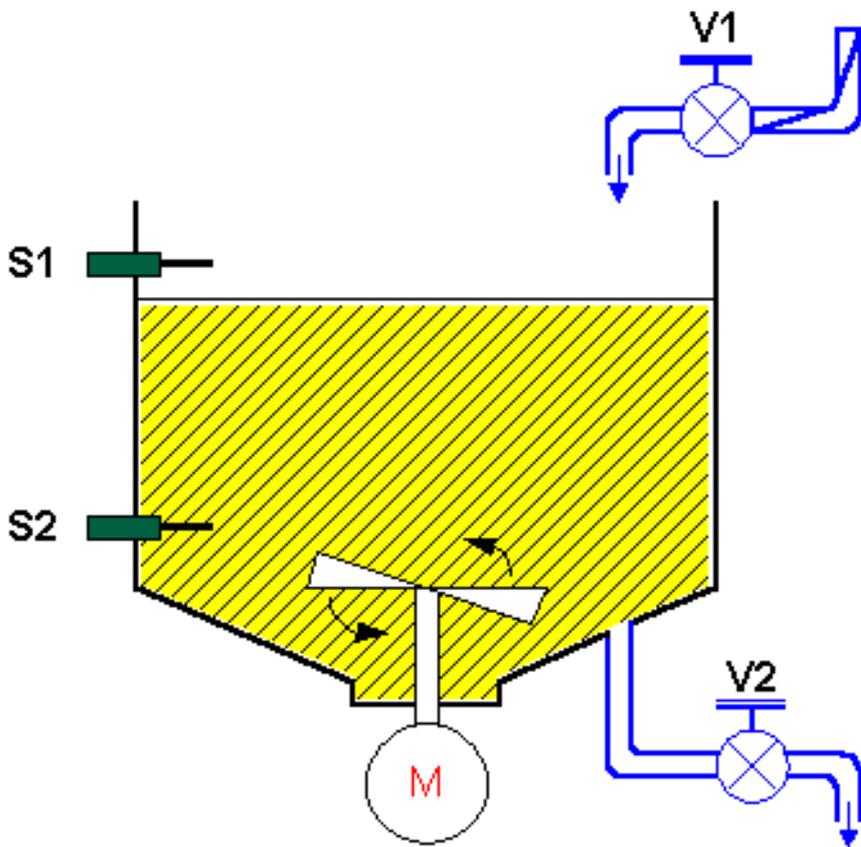




7.7 Operating a charge and discharge process

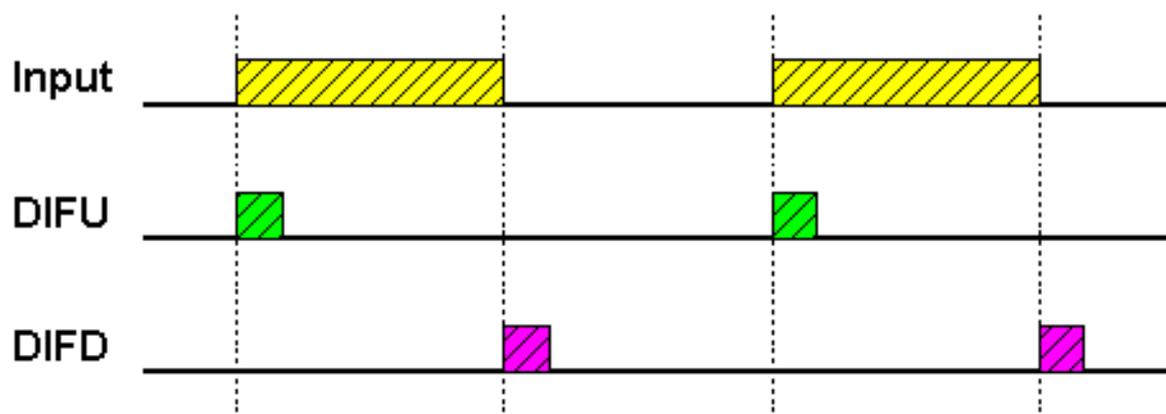
Charge and discharge of a reservoir is a common process in industry as well as a need for mixing two or more substances. By using automated valves this process can be completely automated. Let's say that fluid used in the example is water, and that a reservoir has to be filled up and emptied four times.

When you push T1 on the operating panel, valve V1 opens and a reservoir starts filling up with water. At the same time, motor M of the mixer starts working. When the reservoir fills up, water level goes up and reaches the level set by a sensor S1. V1 valve closes and motor of the mixer stops. Valve V2 opens then, and a reservoir starts emptying. When water level falls below the level set by a sensor S2, valve V2 closes. By repeating the same cycle four times, lamp that indicates end of a cycle is activated. Pressing T1 key will start a new cycle.



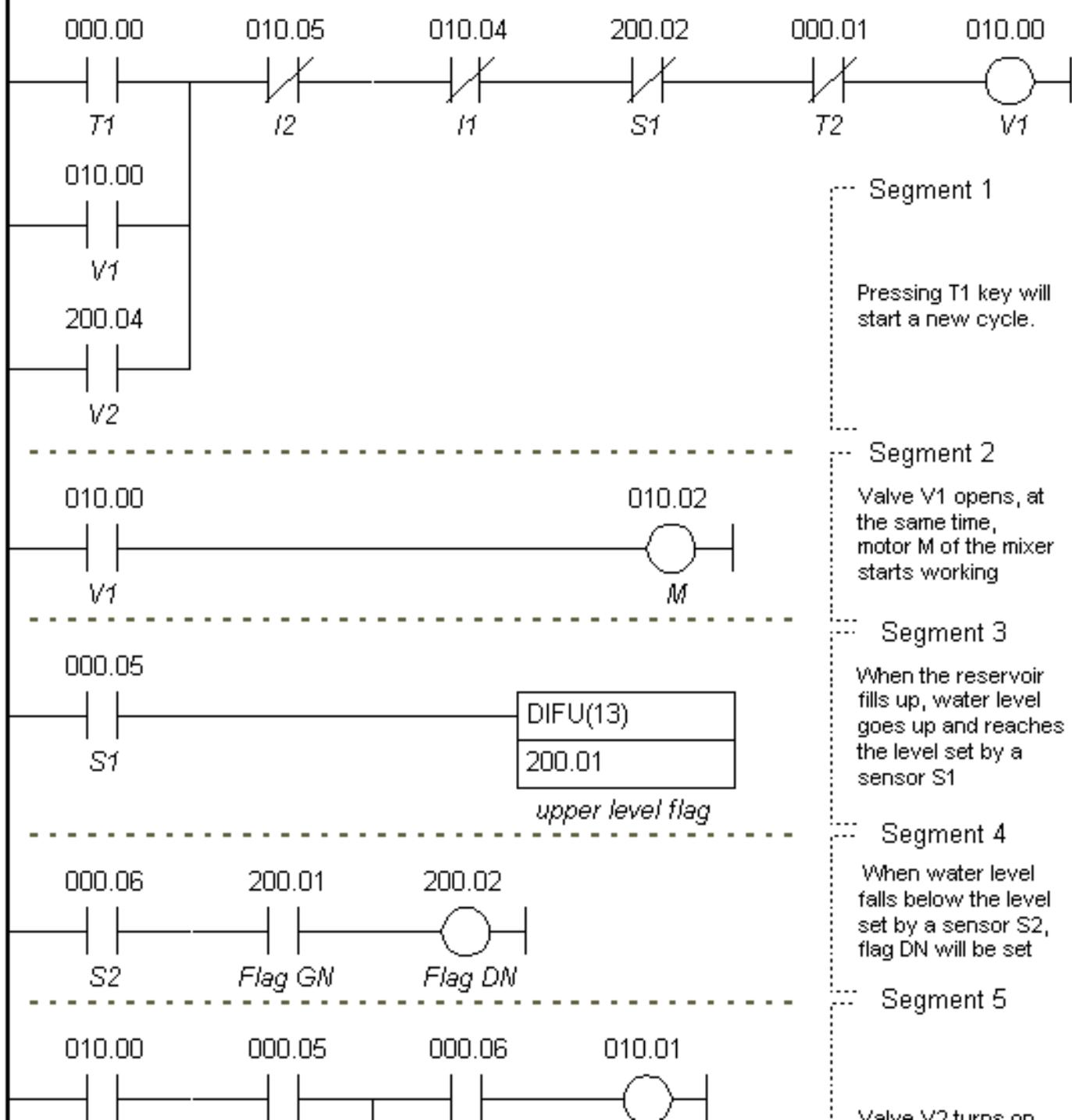
OPERATING PANEL

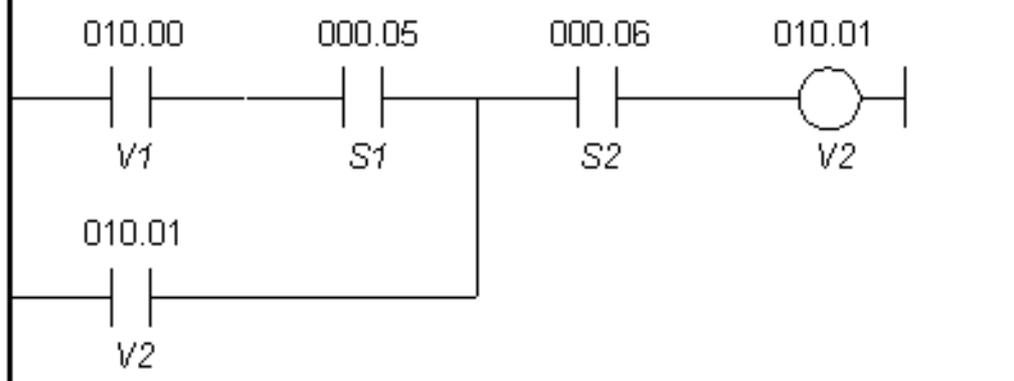
Both types of differentiators are used in this example. You can get an idea of what their role is from picture below. Level S1 and S2 sensors provide information on whether fluid level goes beyond a specified value. This type of information is not important when you wish to know whether fluid level goes up or down in a certain sequence. Mainly, event of approaching the upper level, or a moment when fluid that fills up a reservoir goes beyond upper level and activates sensor S1 is detected in segment 3 of a ladder diagram. Brief activation of IR200.02 output has as a consequence a turn off of an output V1 (valve for water, prevents further flow of water but also motor operation in the mixer). Moment prior to this (segment 5) valve V2 turns on which marks a beginning of fluid outflow. Other two differentiators (in segments 6 and 7) have a task of registering events such as closing a valve MV2 and drop in fluid level below allowed minimum.



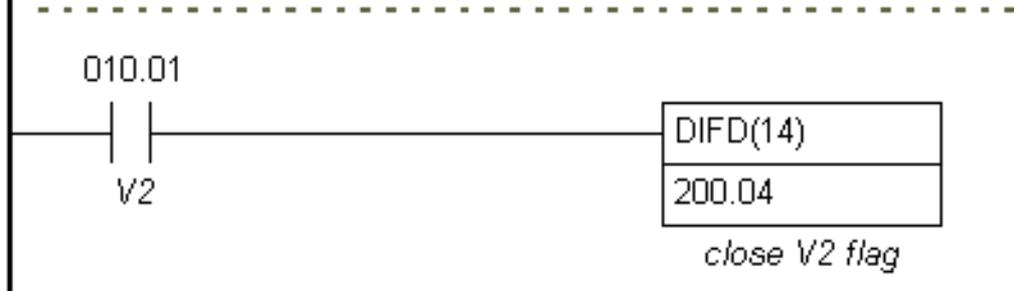
Ladder diagram:

Main 1	When you push T1 on the operating panel, valve V1 opens and a reservoir starts filling up with water. At the same time, motor M of the mixer starts working. When the reservoir fills up, water level goes up and reaches the level set by a sensor S1. V1 valve closes and motor of the mixer stops. Valve V2 opens then, and a reservoir starts emptying. When water level falls below the level set by a sensor S2, valve V2 closes.
RESERVOIR	





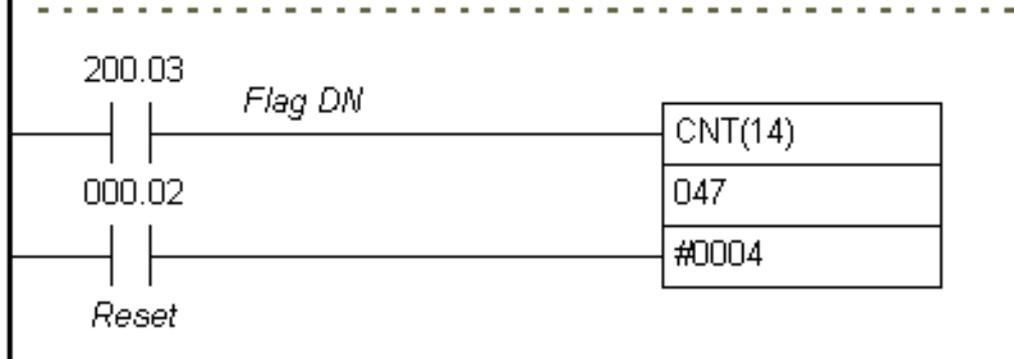
Segment 5
Valve V2 turns on which marks a beginning of fluid outflow.



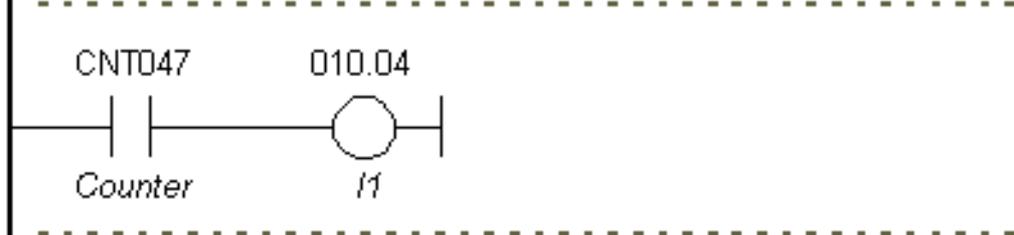
Segment 6
Valve V2 close V2 flag



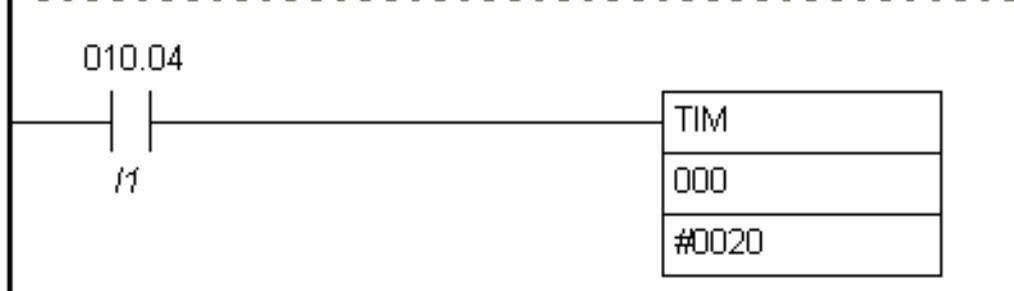
Segment 7
Activation of sensor S2 will setting low level flag.



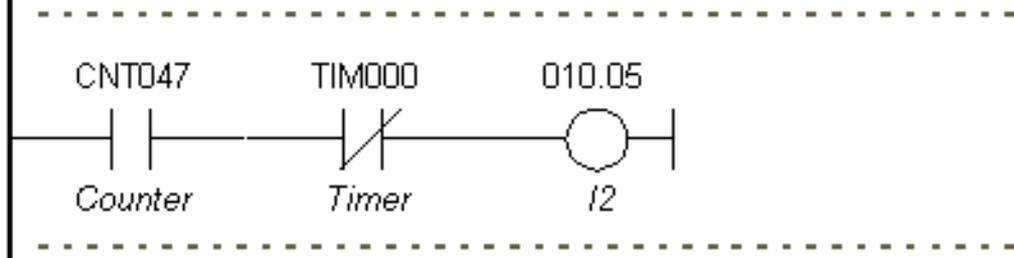
Segment 8
Compare number of low level flags with number of cycle.



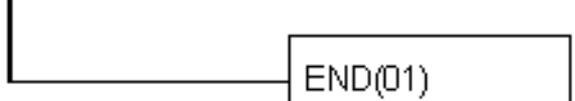
Segment 9
By repeating the same cycle four times, lamp that indicates end of a cycle is activated



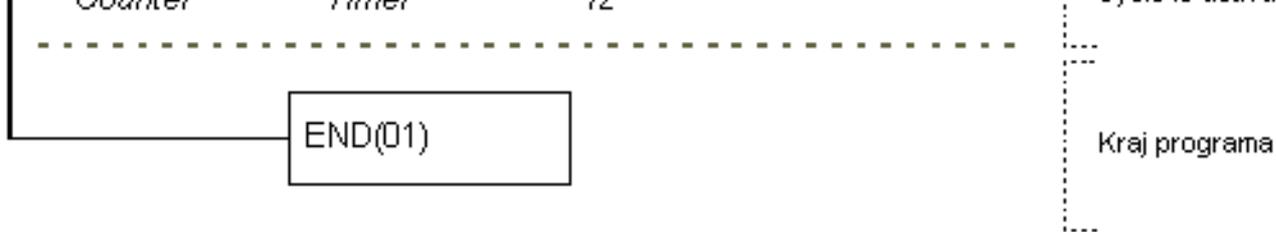
Segment 10
Set timer on 2 sec



Segment 11
By repeating the same cycle four times, lamp that indicates end of a cycle is activated.

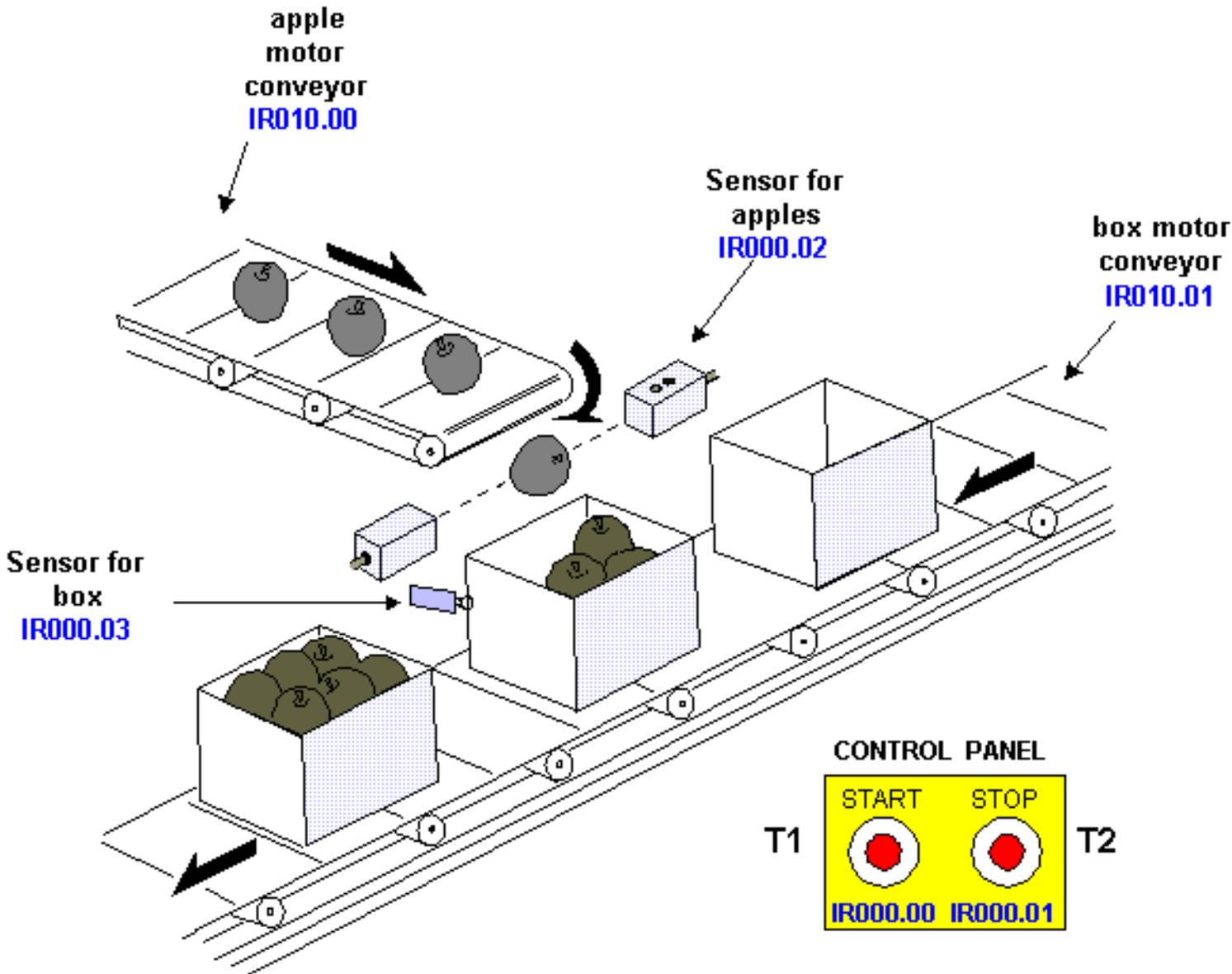


Kraj programa



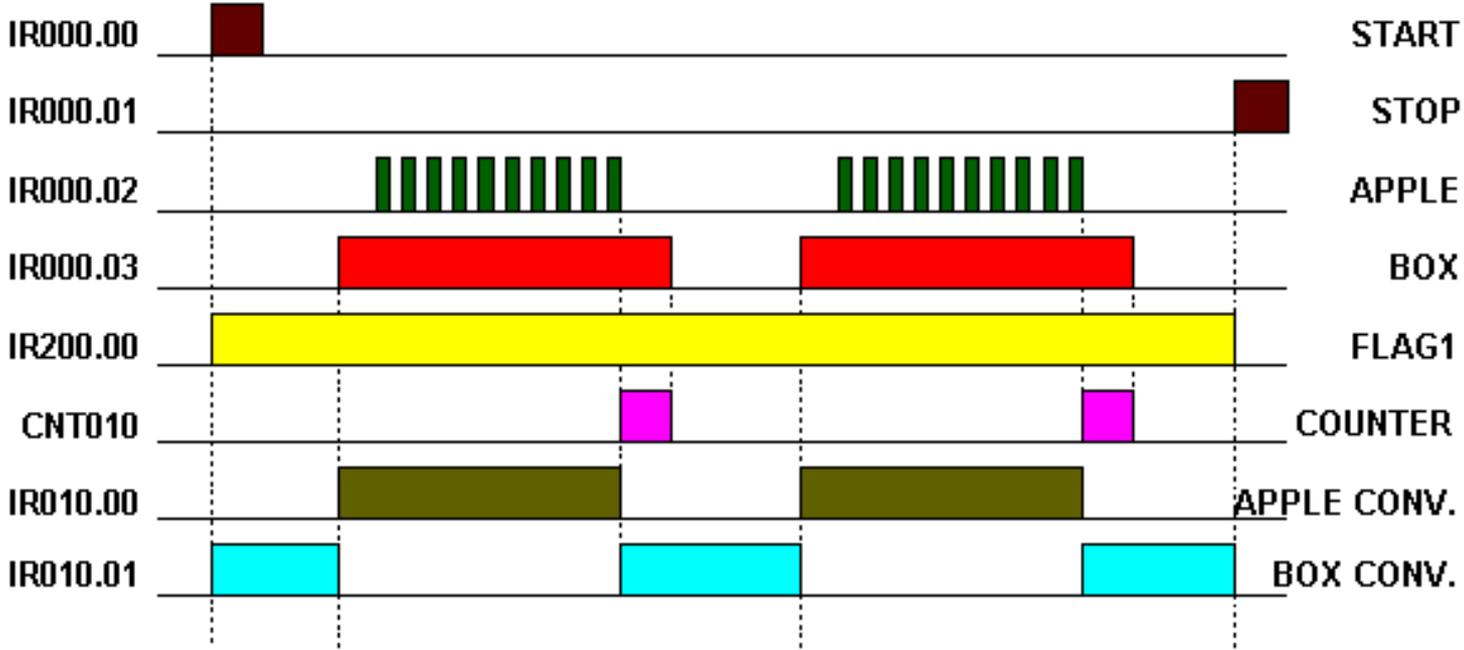
7.8 Automation of product packaging

Product packaging is one of the most frequent cases for automation in industry. It can be encountered with small machines (ex. packaging grain like food products) and large systems such as machines for packaging medications. Example we are showing here solves the classic packaging problem with few elements of automation. Small number of needed inputs and outputs provides for the use of CPM1A PLC controller which represents simple and economical solution.



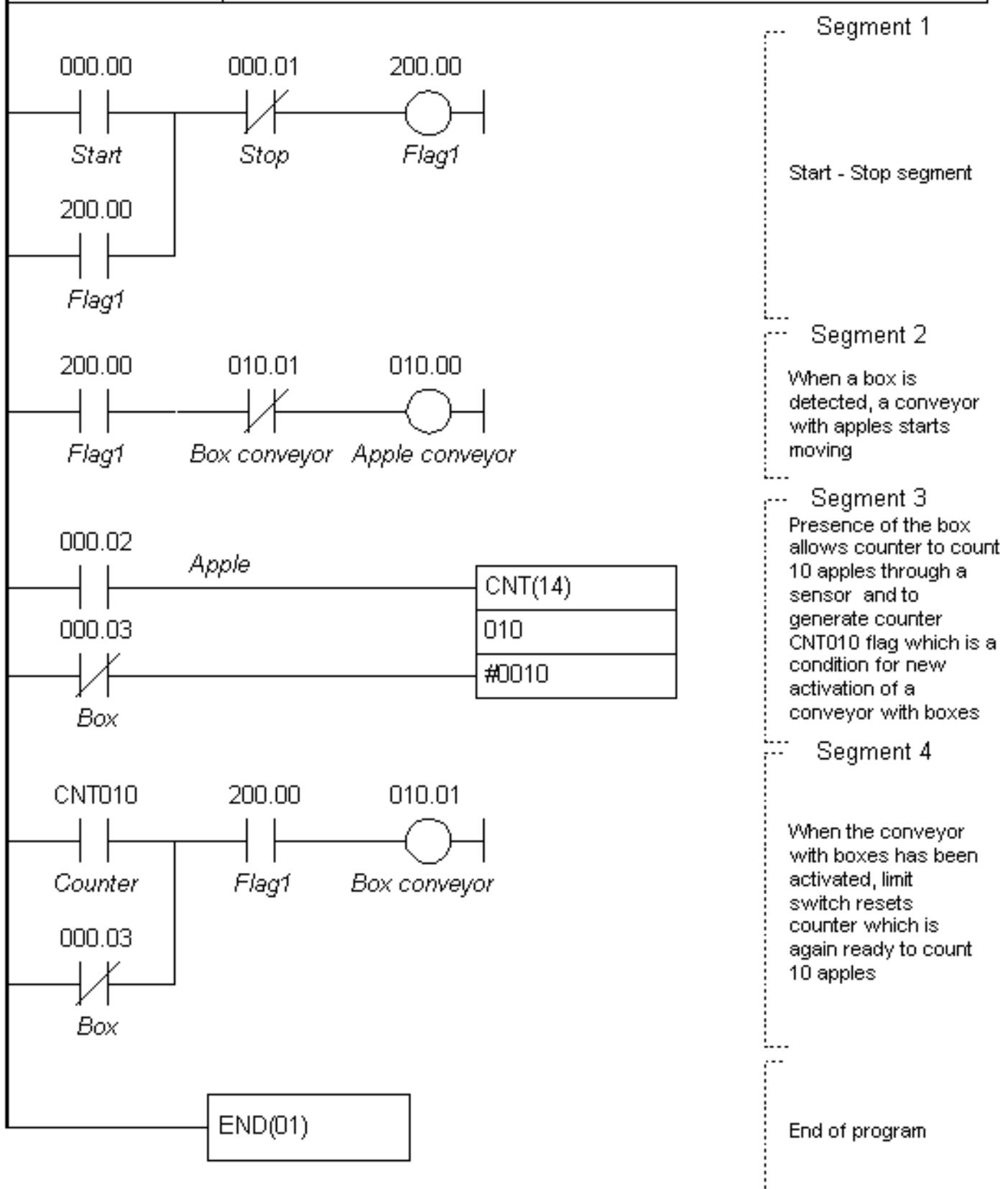
By pushing START key you activate Flag1 which represents an assisting flag (Segment 1) that comes up as a condition in further program (resetting depends only on a STOP key). When started, motor of an conveyor for boxes is activated. The conveyor takes a box up to the limit switch, and a motor stops then (Segment 4). Condition for starting a conveyor with apples is actually a limit switch for a box. When a box is detected, a conveyor with apples starts moving

(Segment 2). Presence of the box allows counter to count 10 apples through a sensor used for apples and to generate counter CNT010 flag which is a condition for new activation of a conveyor with boxes (Segment 3). When the conveyor with boxes has been activated, limit switch resets counter which is again ready to count 10 apples. Operations repeat until STOP key is pressed when condition for setting Flag1 is lost. Picture below gives a time diagram for a packaging line signal.



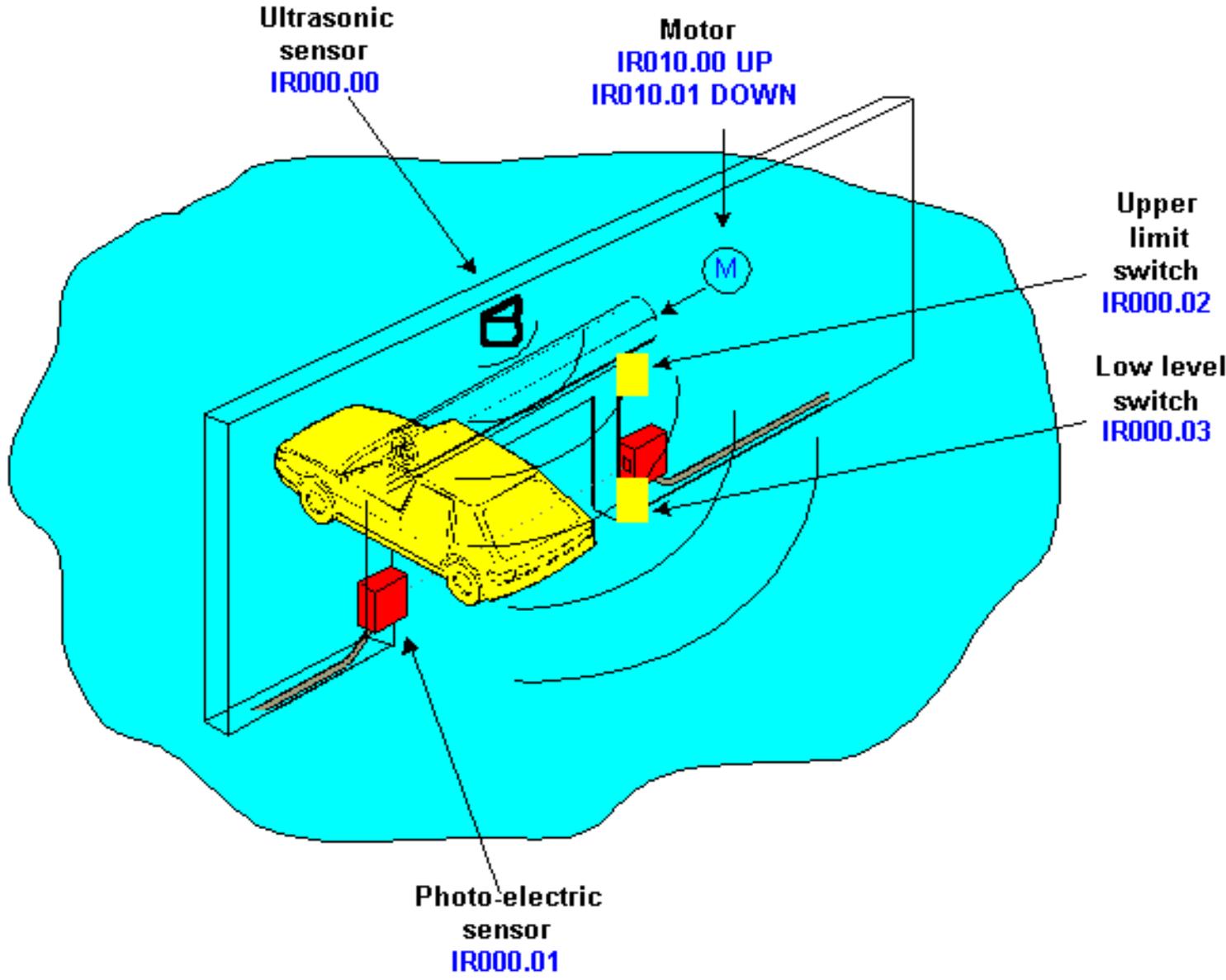
Ladder diagram:

Main 1	By pushing START key you activate Flag1 which represents an assisting flag (Segment 1) that comes up as a condition in further program (resetting depends only on a STOP key). When started, motor of an conveyor for boxes is activated. The conveyor takes a box up to the borderline switch, and a motor stops then (Segment 4).
product packaging	



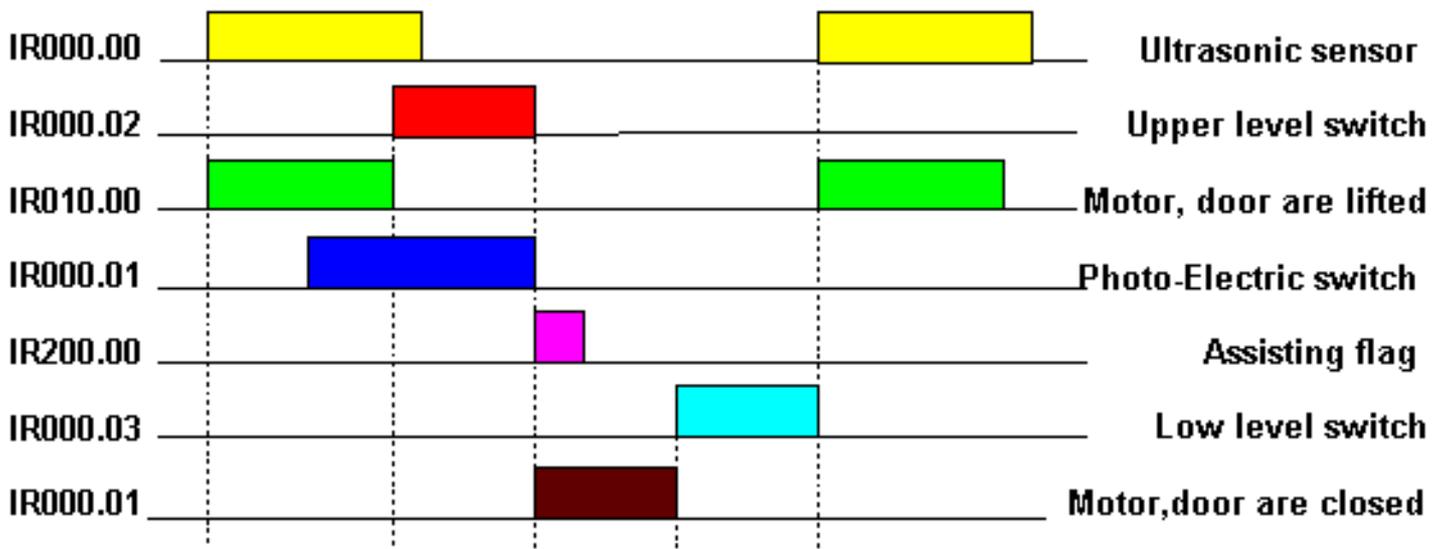
7.9 Automation of storage door

Storage door or any door for that matter can be automated, so that man does not have to be directly involved in their being opened or closed. By applying one three-phased motor where you can change direction of its movement, doors can be lifted up and lowered back down. Ultrasonic sensor is used in recognizing presence of a vehicle by the doors, and photo-electric sensor is used to register a passing vehicle. When a vehicle approaches, the doors move up, and when a vehicle passes through the door (a ray of light is interrupted on photo-electric sensor) they lower down.



By setting a bit IR000.00 at the PLC controller input where ultrasonic sensor is connected, output IR010.00 (a switch is attached to this output) is activated, so that a motor lifts the doors up. Aside from this condition, the power source for lifting the doors must not be active (IR010.01) and the doors must not be in upper position already (IR000.02). Condition for upper limit switch is given as normally closed, so change of its status from OFF to ON (when doors are lifted) will end a condition for bit IR010.00 where power source for lifting the doors is (Segment 1).

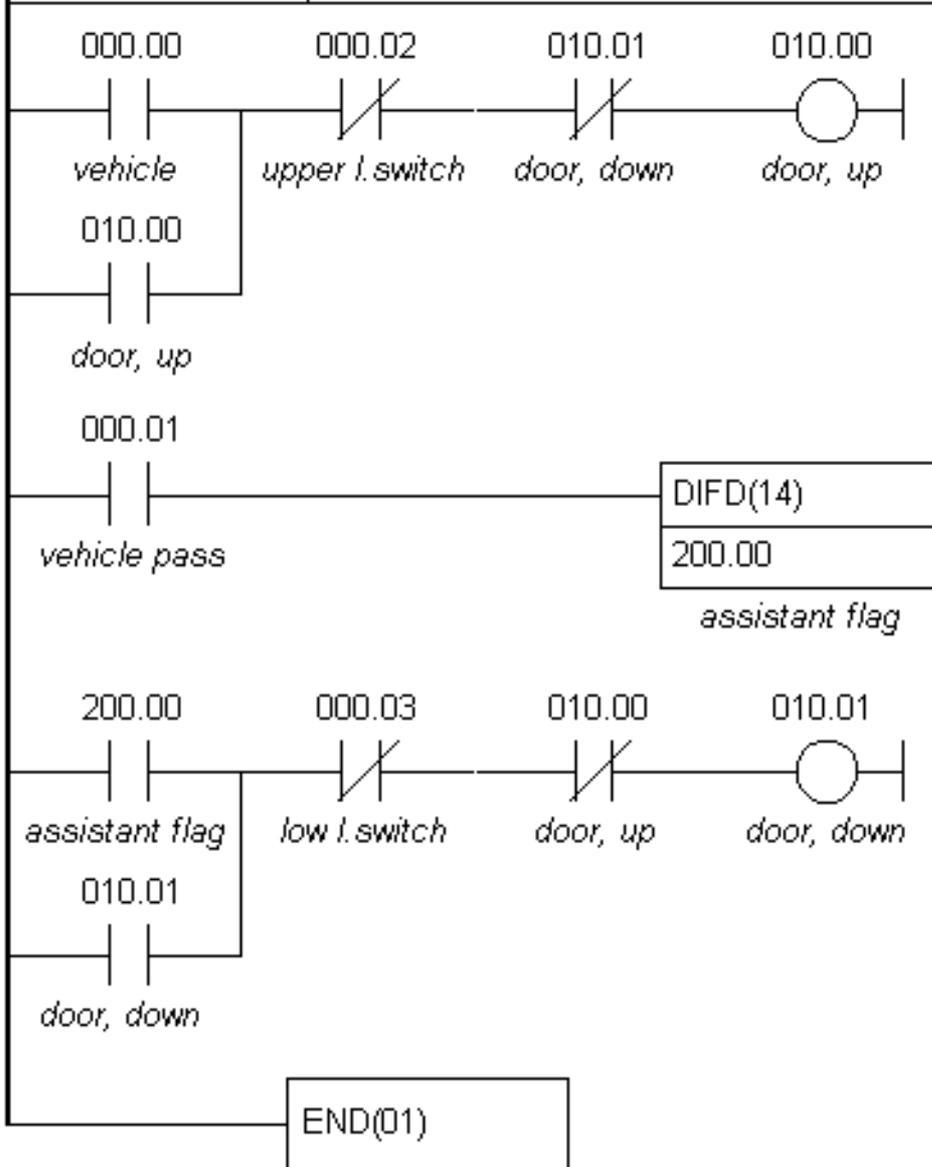
Photo-electric switch registers a vehicle that passes by, and sets flag IR200.00. DIFD instruction is used. This instruction is activated when a condition that precedes it changes status from ON to OFF. When a vehicle passes through a door, it interrupts a ray and bit IR000.01 status changes from ON to OFF (Segment 2).



By changing status of an assisting flag from OFF to ON a condition for lowering a door is executed (Segment 3). Aside from this condition, it is necessary that a unit power source for lifting a door is turned off, and that door is not in lower position already. Bit which operates this power source for lowering, IR010.01 is automatic, so doors are lowered until they come to the bottom limit switch which is represented in a condition as normally closed. Its status change from OFF to ON interrupts a condition of the power source for lowering doors. With oncoming new vehicle, cycle is repeated.

Ladder diagram:

Main 1	Ultrasonic sensor is used in recognizing presence of a vehicle by the doors, and photo-electric sensor is used to register a passing vehicle. When a vehicle approaches, the doors move up, and when a vehicle passes through the door (a ray of light is interrupted on photo-electric sensor) they lower down.
STORAGE DOOR	



Segment 1
By setting a bit IR000.00 at the PLC controller input where ultrasonic sensor is connected, output IR010.00 is activated, so that a motor lifts the doors up.

Segment 2
Photo-electric switch registers a vehicle that passes by, and sets flag IR200.00.

Segment 3
When a vehicle passes through a door, it interrupts a ray and bit IR000.01 status changes from ON to OFF

End of program