UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Master's Degree course in Automation Engineering

Academic Year 2018/2019

INDUSTRIAL AUTOMATION SYSTEMS

Project Report

Student

Ivonne

Index of Contents

Introduction

The aim of this report is to explain the work that has been done to realize a project about industrial automation application.

It deals with a mechanical system used for loading and unloading of goods has been implemented. It has composed by two carts that are able to move themselves back and forth on the rails. This structure has been modelled by Sequential Function Chart, a graphical programming language used for programmable logic controllers, named PLCs.

Chapter 1: Sequential Function Chart diagram

The second part of this report has been focused on the implementation of the plant for loading and unloading of goods by self-propelled carts.

First, an overview on PLCs and their several programming languages has been done.

1.1. Programmable Logic Controller

The term “Programmable Logic Controller”, often in abbreviation “PLC”, refers to an industry device used in the management or control of several industrial processes. This kind of computer works executing a program and processing digital and analogue signals that have been generated by sensors and then received by the actuators which belong to the industrial plant in question. Nowadays, these devices have been also utilized into domestic environment, thanks to the progressive miniaturization of their electronic components and their low cost.

A PLC is a modular hardware object, characterized by robustness to interference, high temperature and great humidity and it’s capable to work, for a long time, on system that can never stop.

PLC structure depends on the process that needs to be automated, but it’s generally composed of a power supply unit, of the CPU that can have internal or external RAM, ROM, EPROM or EEPROM memory, of several digital input and outputs cards, and if it’s necessary also the analogue ones. If the PLC operates in a network with other PLCs, communication cards are needed too.

Generally, a logic controller can be defined as a device that relates input logical variables to output variables, using a set of combinational and / or sequential algorithms. The logic controller is defined as static, if the outputs depend on the values of the inputs at the same time, while it’s said dynamic if equations that bind inputs and

outputs are of the sequential type, so the outputs of the system also depend on the past values of the inputs.

This kind of controllers can be implemented by means of a set of physical devices that create logic gates, such as AND, OR, NOT and delay elements that define the sequential algorithm through their interconnection or by a programmable electronic system and a suitable stored control program. So, in the first case, it’s called wired logic controller and in the other case it’s named as programmable logic controller.

A logic or sequential controller is characterised to be easily programmable and reprogrammable in minimum interruption times, robust and therefore made with components and materials suitable for operation in an industrial environment, modular to permit maintenance and repairing actions. Moreover, it should have an easily expandable configuration, small dimensions and low energy consumption, the capability to interface with sensors, actuators and centralized systems in a simple way and an expandable internal memory for programs and data. Generally, the PLC is equipped with an operating system and several user-oriented programming modes, such as a graphic programming language that represents schematics of electric circuits.

In 1993, the international electrotechnical committee (IEC) has issued a standard about the hardware and software structure of this component, named as IEC 61131. In particular, this document defines the PLC as "an electronically operated digital system, intended for use in the industrial environment, which uses a programmable memory for the internal storage of user-oriented instructions for the implementation of specific functions, such as logic, sequencing, timing, counting and arithmetic calculation, and to control various types of machines and processes through both digital and analogue inputs and outputs ".

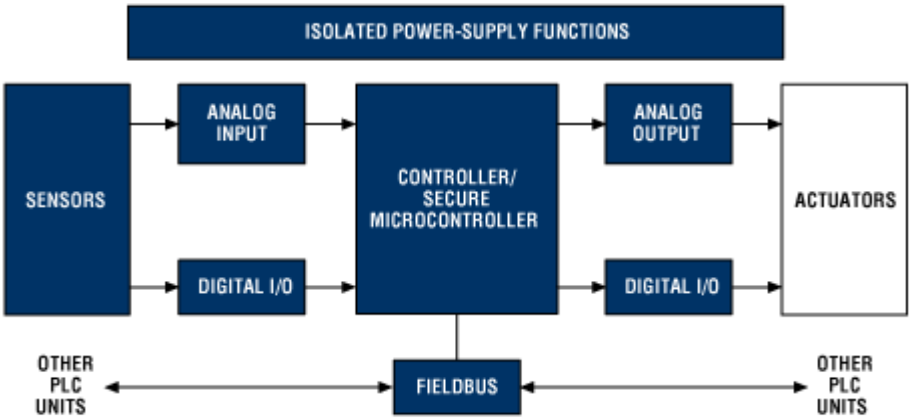


Figure 1: PLC scheme.

1.2. Programming languages

The PLCs should be programmed using a specialized software that runs on a PC, which allows to create programs to download in the PLC CPU memory.

In the IEC 61131 document, five programming languages have been standardized, which are three graphical programming languages, that are *Ladder diagram*, *Sequential function* chart and *Function Block Diagram* and two textual one, namely *Instruction List* and *Structured Text*.

1.2.1. Ladder Diagram

The Ladder Diagram, also called “contact language”, is the computer transposition of electrical circuits that have been used in the definition of sequential wire controllers. So, logic symbols, such as AND, OR and NOT, corresponding to input and output signals are used to implement the control logic by drawing the electrical diagrams on the programming software and no longer by wiring the relays.

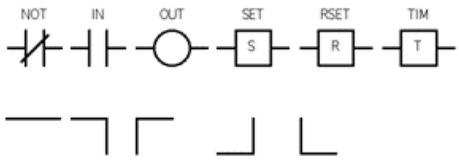


Figure 2: Ladder Diagram symbols.

So, this language is a graphical representation of electronic devices that realize logical expressions in which the operands are open or closed relays and their result is stored inside a coil. In fact, two vertical lines represent the power supply and delimit the electrical components of the circuit and so the logical operations. Then, these sequential expressions are represented by horizontal lines that power the coils allowing the flow of the current, from the left line to the right one. Contacts can allow or deny the passage of current from left to right, depending on their state, realizing the behaviour of a Boolean variable. In the same way, coils can be powered or not, thus storing the true or false value of the circuit connected to them.

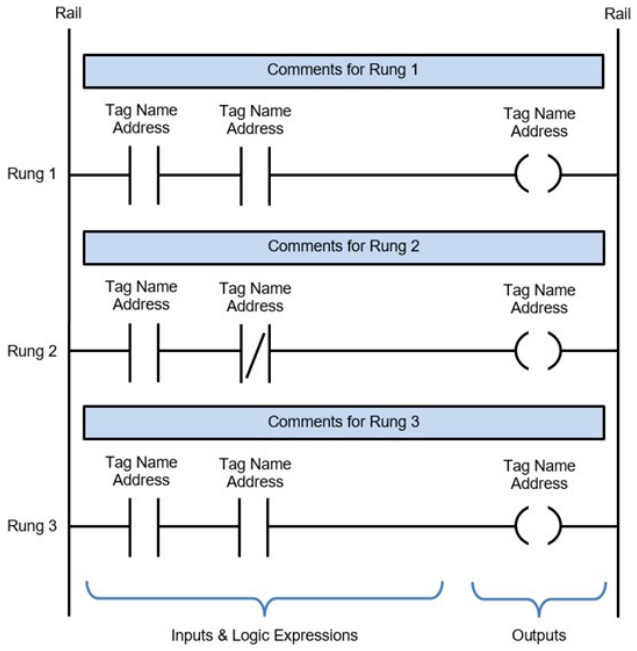


Figure 3: Example of a LD structure.

Ladder programming follows two fundamental principles, which are that the energy flow moves from left to right and the execution of instructions always takes place from top to bottom. Moreover, contacts let the current flow only in the presence of a rising edge of the variable associated with them, in fact, the current can flow though one of

them only if its associated variable takes the true value, after that it's has been characterized by the false value in the previous cycle. These contacts are graphically characterized by the letter P and there are also some contacts that let the current flow in the presence of a falling edge of the associated variable, which are indicated by the letter N.

1.2.2. Sequential function chart

The sequential function chart is a graphic programming language used for modelling the sequential evolution of an automation system, by implementing a finite state automaton, whose machine cycle is represented as a series of sequential control actions. As it's a high-level programming language, it can be considered a graphic formalism for the description of logical actions, in fact, in the implementation of a sequential control program, actions and conditions should be better specified, and so other languages, such as Ladder Diagram, have been necessary. The SFC is composed of three fundamental graphical elements, that are state, transition and directed arc.

The *state* or step is a situation of the system that can change or be modified only at the occurrence of a certain event. So, it's a specific operating condition of the system in question which refers to determined actions that should be accomplished. Then, the evolution of the system from one state to another occurs when the corresponding events happen. So, a state can be active, or inactive, and to it has been associated several control instructions that will be executed only when the status is active.

Generally, states are represented by rectangles in which the representative and unique name of the state is inserted, such as numerical element. Moreover, the actions which are associated to each state are usually figured as a rectangle too, in which these have been described and it's connected to the state which it refers.

Transitions represent the possibility of evolution from a state to another, when a condition that has been associated to the transition in question has been verified. These transitions, usually expressed through Boolean functions and represented as a

horizontal line that cuts the arc that unites two states, are used to describe the events that could change the operating status of the system. Finally, the evolution from one state to another is univocally determined by the directed arc that unites them and by the transition, in addition to the relative condition, which determines the possibility of this evolution to happen.

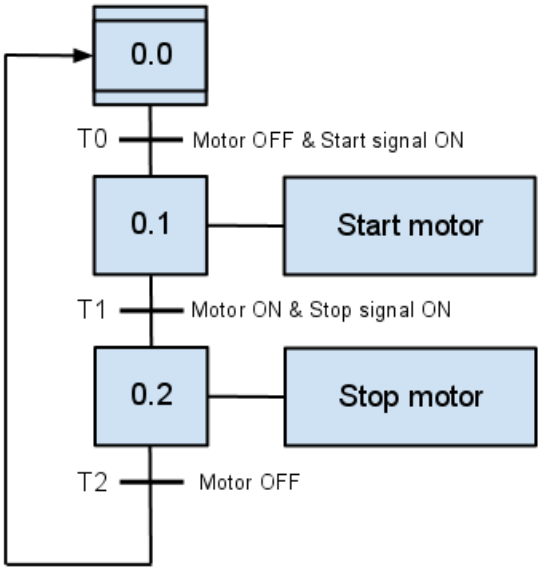


Figure 4: Example of SFC program.

So, the sequential control via an SFC diagram gives the possibility to describe, in a simple and unambiguous way, the specifications of sequential operation of an automatic system, to overcome the limits of finite state automata, for example by allowing the parallelism.

1.3. Self-propelled Carts Project

The automation process that should be implemented is a system for loading and unloading of goods by means of two self-propelled carts. Their movement takes place along two rail lines, that join into one where two carts share the same area. These zones are the rest area and the unloading one. On the contrary, the loading area and the

waiting one are separated, in fact they consist of two different tracks, on which each cart can move itself back and forth, according to the tasks that it should accomplish. The access on the shared areas has been managed by a semaphore that guarantees the mutual exclusion of the shared track. So, it has the aim to give a signal to an actuator which should activate the rail switch of this common resource in favour of A cart or B cart, so any collision can occur. Moreover, both in proximity of the loading area and the unloading one, some actuators are present to realize loading and unloading operations of goods.

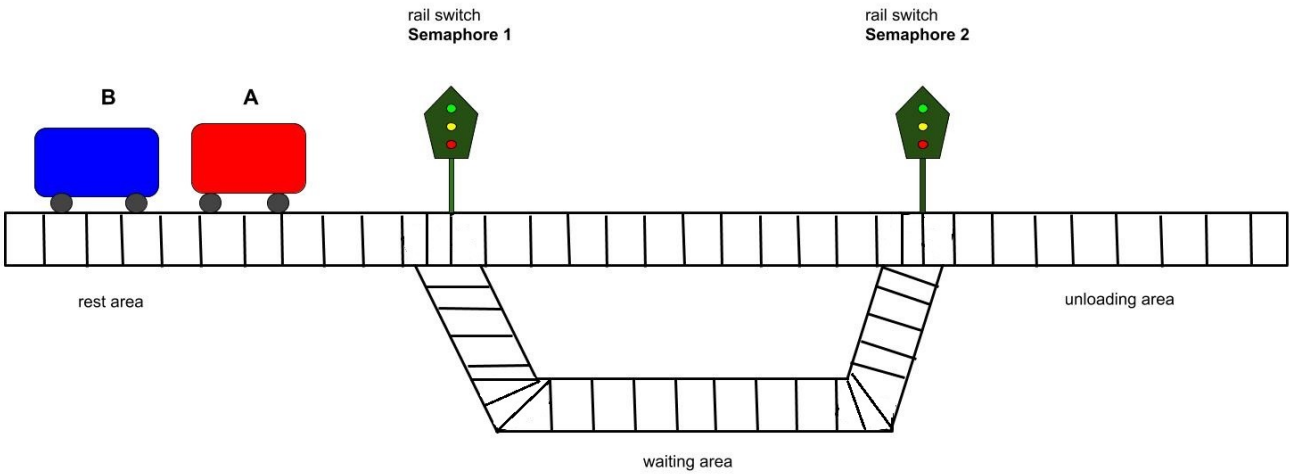


Figure 5: Self-propelled cart system.

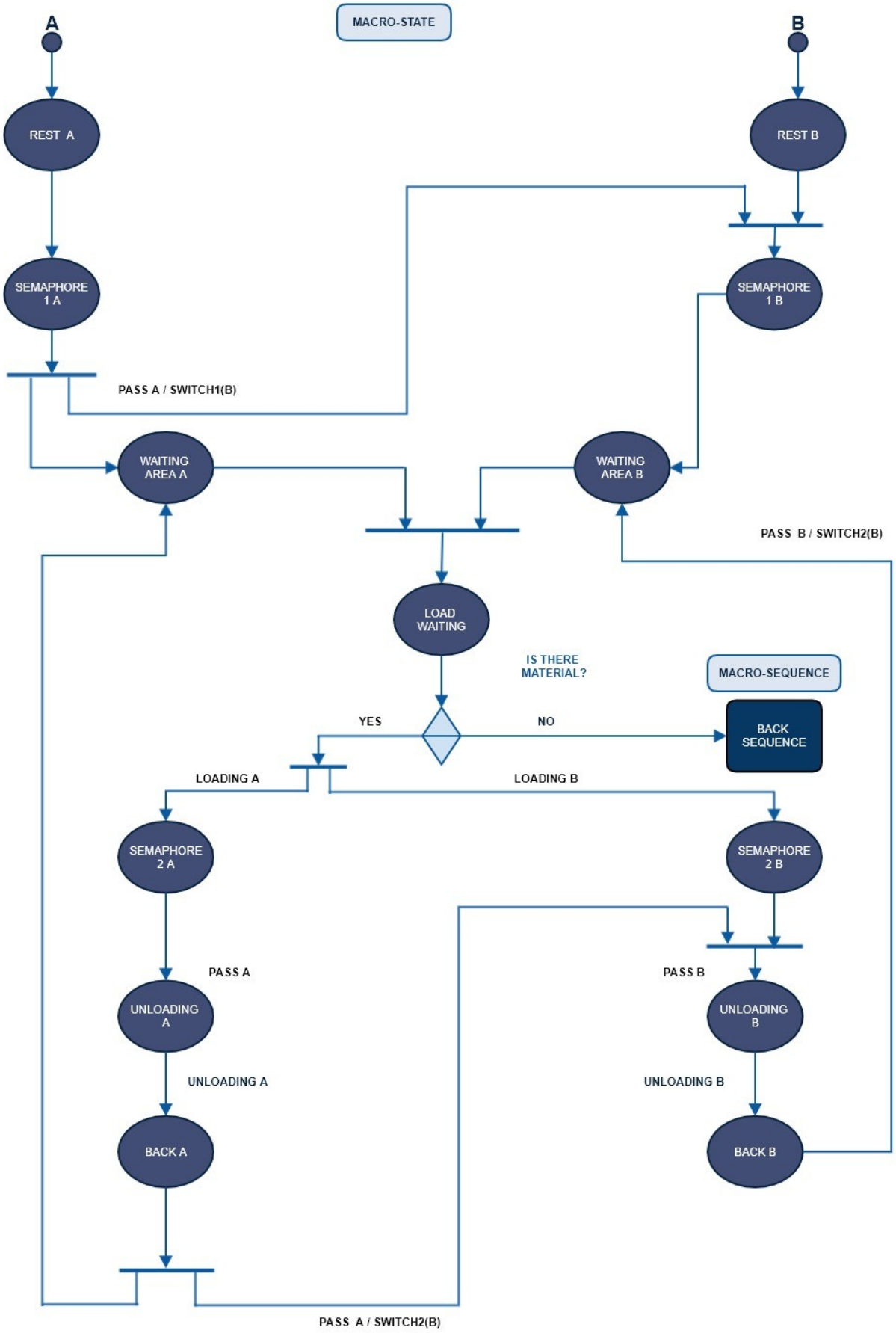
The project purpose is to solve this problem making the carts move themselves independently from each other, ensuring the absence of crashes in the shared zone.

The initial position of the carts consists in the rest area and it's important that their disposition on it respects that A cart follows B cart. So, A cart is always the first to start, when the whole automation system has been activated by an appropriate signal. Then, the presence of the two carts on the loading and unloading area will be verified, before starting the related operations, thanks to the use of specified sensors.

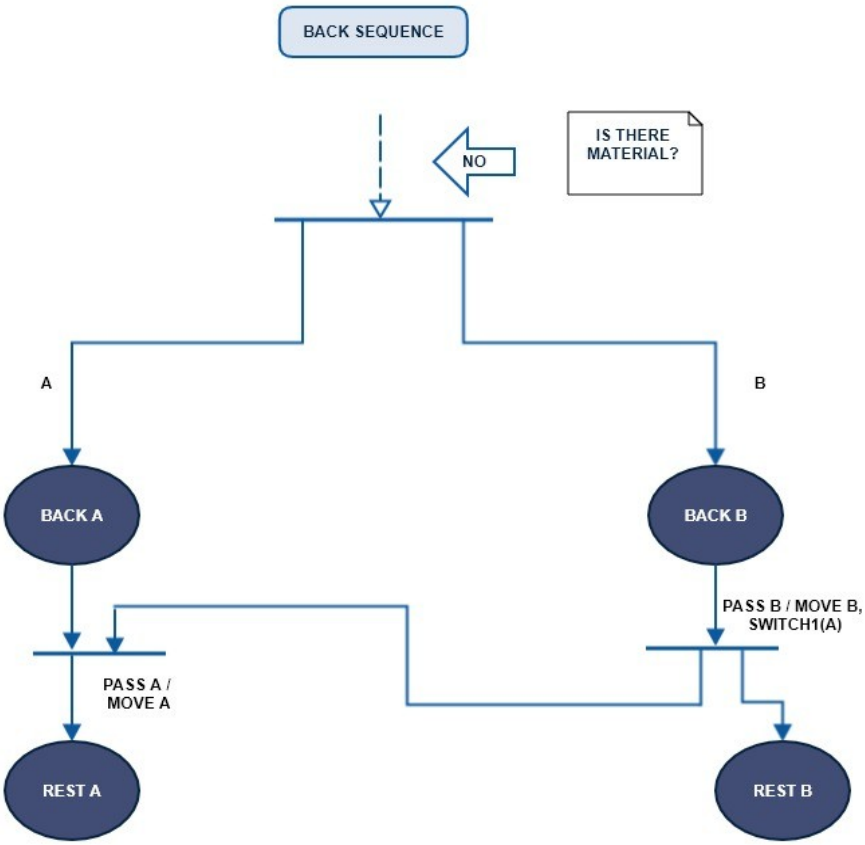
1.3.1. Finite-state machine

A finite-state machine or finite-state automaton is a mathematical model of computation, an abstract machine that is composed of a finite number of states. It can change from one state to another in response of some external inputs, that can verify or deny determined conditions that have been assigned to a specified transition. So, a finite-state machine is defined by a list of its states, its initial state and the condition for each transition.

Now, a representation of a finite-state machine, describing the automation system which has been reported in this work, has been figured as follows:


A
MACRO-STATE
B
REST A
REST B
SEMAPHORE 1 A
SEMAPHORE 1 B
PASS A / SWITCH1(B)
WAITING AREA A
WAITING AREA B
PASS B / SWITCH2(B)
LOAD WAITING
IS THERE MATERIAL?
MACRO-SEQUENCE
YES
NO
BACK SEQUENCE
LOADING A
LOADING B
SEMAPHORE 2 A
SEMAPHORE 2 B
PASS A
PASS B
UNLOADING A
UNLOADING B
UNLOADING A
UNLOADING B
BACK A
BACK B
PASS A / SWITCH2(B)

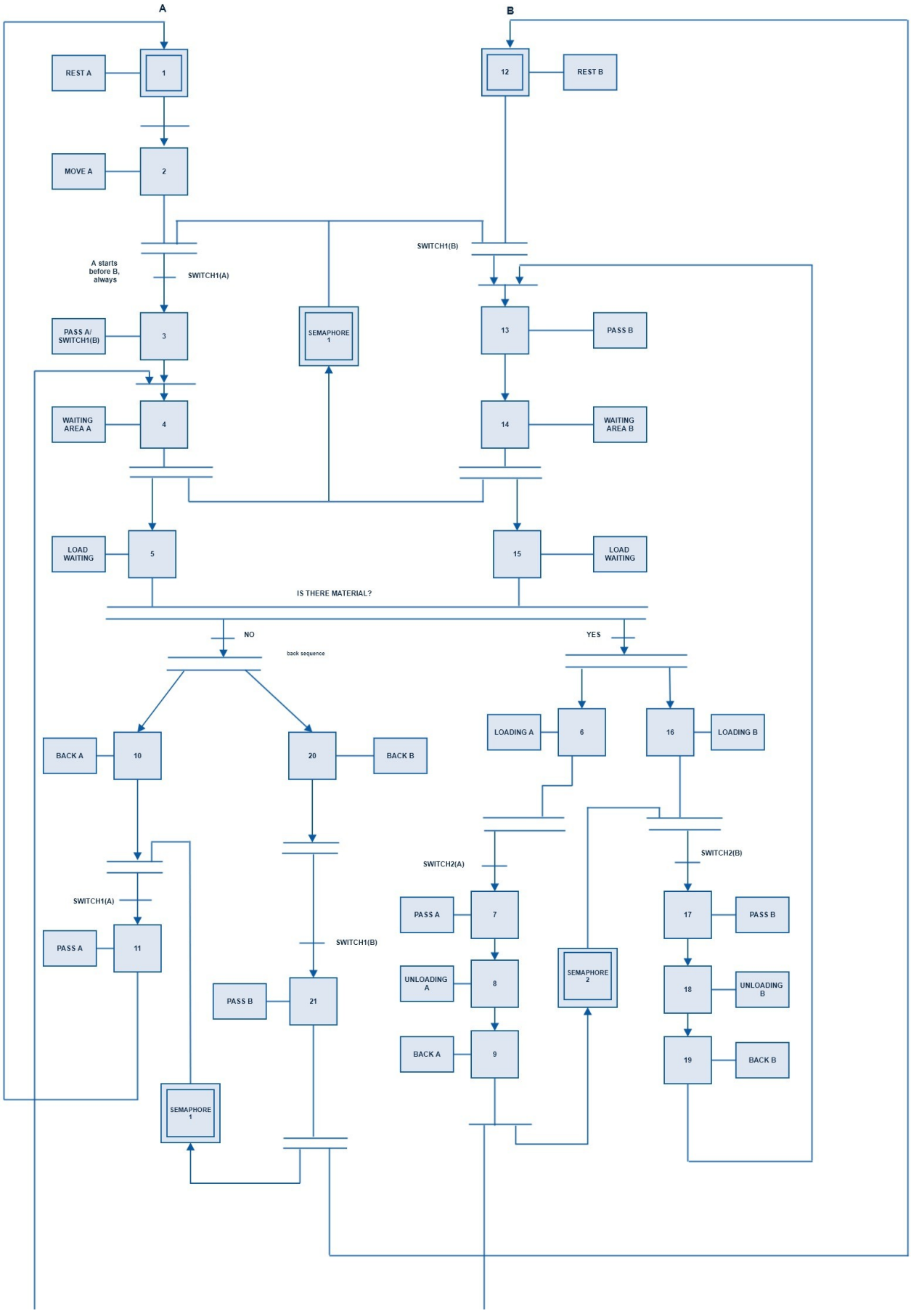
The explication of the macro-sequence called “Back sequence” has been shown below:



To do this finite-state automaton the Mealy machine has been used. It’s a finite-state automaton whose output values are determined by the current state and the current input, unlike the Moore machine, which instead works only as a function of the current state. In this case, an action has been associated to the arc transition and any possible transition can specify a different action.

1.3.2. SFC diagram implementation

Finally, as required by project specifications, the sequential control of the industrial process described above has been realized. For its implementation, the graphical programming language called SFC has been used to draw the following diagram:


A
B
REST A
1
12
REST B
MOVE A
2
A starts before B, always
SWITCH1(A)
SWITCH1(B)
PASS A/ SWITCH1(B)
3
SEMAPHORE 1
13
PASS B
WAITING AREA A
4
14
WAITING AREA B
LOAD WAITING
5
15
LOAD WAITING
IS THERE MATERIAL?
NO
back sequence
YES
BACK A
10
20
BACK B
LOADING A
6
16
LOADING B
SWITCH1(A)
SWITCH1(B)
SWITCH2(A)
SWITCH2(B)
PASS A
11
PASS B
21
PASS A
7
17
PASS B
UNLOADING A
8
SEMAPHORE 2
18
UNLOADING B
BACK A
9
19
BACK B
SEMAPHORE 1

Conclusion

In this report, a problem related to industrial automation systems has been considered, about an automated process of loading and unloading of goods, through two carts moving on rail tracks, has been considered. Generally, this kind of problems are related to PLC applications in an industrial environment. So, to formalize it, a graphical programming language that is Sequential Function Chart has been chosen and the corresponding diagram has been drawn.