Exploration of IEC 1131-3’s LAD and SFC Languages in PLC Programming

by

Yew Jia-Ming

15952

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
Electrical and Electronics

JANUARY 2016

Universiti Teknologi PETRONAS
Bandar Seri Iskandar,
32610 Tronoh,
Perak Darul Ridzuan,
Malaysia

CERTIFICATION OF APPROVAL

Exploration of IEC 1131-3's LAD and SFC Languages in PLC Programming

by

Yew Jia-Ming

15952

A project dissertation submitted to the
Electrical and Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
Electrical and Electronics

Approved by,

(Dr. Nordin Saad)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2016

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

YEW JIA-MING

Abstract

With the vast utilization in industrial applications, Programmable Logic Controller has built a strong foundation in the industrial sectors. Having five programming languages being recognized by IEC61131-3, program developers have the freedom to opt for the languages that suit themselves as well as base on their prior basic knowledge about that language. However, the selection of languages will affects the effectiveness of a project or application and programmer should choose the language that suits the application the best. Having a PLC in controlling an electro-pneumatic actuating robotic mechanical arm, which programmed by LAD and SFC, this project aims to evaluate and study the use of these two languages in approaching industrial automation. With performance of the mechanical arm being analyzed as well as the program structures, conclusions are being made on the aspect of the suitability of the languages in approaching industrial applications, ease of use and shortcomings. This project explicit the steps in construction and transformation of a movement diagram and sequential chart into ladder logic and the simulation of the logic diagram.

Acknowledgement

Firstly, I would like to acknowledge with gratitude to Associate Professor, Dr. Nordin bin Sa'ad, my respective supervisor, for giving me the opportunity to be under his guidance in completing this research. It's a great opportunity for me to write about subject like "Exploration of IEC 1131-3's LAD and SFC Languages in PLC Programming", as I am always interested in the field of automation control.

I am immensely grateful to Dr. Nordin bin Sa'ad, who has always been sincere and helpful in making me understand the concepts in the automation world. At the time of preparing this paper, I have gone through different research papers and websites, which enlightened me with the basic of automation standards, languages and PLC working principles. Besides, with the subject "Industrial Automation and Control Systems", I am able to understand and build my foundation on the subjects of LAD and SFC constructions.

I would like to take this opportunity to express my sincere appreciation to Mr. Azhar, who has provided insight and expertise that greatly assisted this research. With his advices on the automation simulation aspects, I am able to get better results and in a shorter period of time.

I must also thank Mr. Isnani for helping me in setting up the electro-pneumatic actuating robotic mechanical arms for this project. Without his assistances, I would not be able to complete the cabling connections and interfacing between the actuators and programmable logic controller in limited time.

This research paper is made possible through the helps and supports from everyone, and I sincerely thank you for their insights and comments. Any mistakes or imperfect results are my own and should not tarnish the reputations of these esteemed persons.

Table of Content

List of Figures

List of Tables

Abbreviations and Nomenclatures

LAD	Ladder Diagram
SFC	Sequential Function Chart
IEC	International Electrotechnical Commission

Chapter 1: Introduction

1.1 Background

With the employment of automation and control system in broad range of industrial applications, a lot of sectors such as manufacturing, packaging, automobile as well as petro-chemical are able to attain process outcomes with higher speed, accuracy and repetitiveness. Reliability, endurance, assurance of products and services quality are guarantees with the utilization of automation.

Since the introduction of Programmable Logic Controller in the 1960's, most of the automation and instrumentation control systems are being responsible by PLC. PLC are microprocessor-based computers with the purpose of implementing control algorithm in industrial automation. [6] PLC is able to provide a reliable and long service lifespan, making it remains as the backbone of most automation projects in the sector of process and manufacturing control. [1]

With the wide application of PLC, it is important that programs, and subsequently the behaviour of the controlled application can be understood by industrial personnel. Since PLC was first introduced to replace hard-wired relay control systems, and in order for electricians who had been dealing with hard-wired control systems to easily understand the working principles of PLC, a relay logical based graphical programming language called Ladder Diagram (LAD) was developed.

Due to the increasing controlling of sequential based application, another programming language called Sequential Function Chart (SFC) was introduced. SFC is an event or time driven programming languages based on a French national standard, depicting sequential behavior of a control system. The resemblance of this language to computer flow chart with its simple concept, travelling from top to bottom, executing every actions, provided with certain conditions, making it receiving welcoming adoption from a lot of vendors.

However, these two languages, together with one graphical and other two textual programming languages, which are Function Block Diagram (FBD), Structured Text (ST), and Instruction List (IL) possess their own benefits as well as shortcomings. They are not able to completely replace one another, and due to this (major factor), IEC1131-3 (IEC61131-3) is established. IEC61131-3 aims to address the method in approaching control problems. Therefore, in this project that we will explore these two programming languages (SFC and LAD) and compare their benefits and shortcomings.

1.2 Problem Statement

Until now, despite having several other programming languages being available and IEC61131-3 recognized five of the mostly used, LAD remains the dominant language being used in developing the PLC programs. It is undoubtedly that LAD is overwhelming due to its adoptability from the earlier relay logic diagram, and hard-wired like characteristics, but over the years, as the complexity of the applications tends to increase, it is obvious that the result tends to put greater weight on the formalized programming languages.

The ultimatum for lesser development time, and possibility of re-using existing software modules result in the need for formal approach in PLC programming. [7] & [8] However, [6] showed that an investigation among skilled PLC users on the aspect of programming languages preferences, 25% of the participants are selecting a tool based on their prior knowledge rather than performances. This explains the slow adoption of SFC in North America and rest of the world.

The choice of PLC software structure used in a project has an impact on efficiency and process flexibility. [6] demonstrated that with an appropriate choice, will bring about significant cost savings in development time. In this project, we will program an electro-pneumatic actuating robotic mechanical arm controlled by a PLC with LAD and SFC. The study of the performance of an electro-pneumatic actuating robotic mechanical arm in performing pick and place action is being carried out in this project. Comparison and analysis were also being done on the program structures of these programming methods.

In particular, this project work is attempting to answer the following questions:

- How to develop a programming routine in SFC for a PLC to control an industrial process (pick and place robot)
- How to implement the SFC approach on a PLC that use a software that does not support SFC. (Older available PLC software tends to only support LAD)

1.3 Objectives

As justified from the title of this project, which is Exploration of IEC 1131-3's LAD and SFC Languages in PLC Programming, this project is aimed to

- Evaluate through 'study-by-doing', of the programming process necessary when using LAD, and SFC programming languages in approaching industrial automation problems.
- Compare two standard programming methods, LAD and SFC, in terms of the approach of solving a problem, programming steps, limitations of the project, documentations, and similarities.

Based on these objectives, the expected outcomes would be a

- Guide on the appropriate way to approach a problem using LAD and SFC languages implemented on an industrial analogical five electro-pneumatic actuator robotic system.
- Workable robotic systems programmed using both programming languages with complete documentations.
- Performance comparison for the mechanical arm as a result of the two programming languages, as well as the ease of programming.

1.4 Scope of Study

Followed up from the objective section, the scope of study of this project focuses on:

- Two IEC 1131-3's Standard Programming Languages: LAD and SFC
- Programmable Logic Controller and software
- LAD Simulation software
- Electro-pneumatic actuator robotic mechanical arm

Sequential programming is used in this project to demonstrate the usage of two programming methods. The pneumatic actuator movements are being studied in order to obtain a rough idea or sequence on how those actuators should be moved to perform a pick and place operation. A movement diagram is thus constructed and is being shown in the result section.

Besides, testing was conducted on those five actuators, as to clear out doubts or any physical instrumentation errors occurring on the actuators during the execution of the final revised movement. A series of tests covering part of the actuators are being conducted and the resultant LAD program is attached in the appendices section as well.

1.5 Relevancy and Feasibility

Although much complicated automation systems can be controlled by PLC, however, in this project, a simple "pick and place" application is being used, as the main focus of this project is to provide a proper way to program a PLC using ladder logic and SFC, and to compare the performance.

Although a more complicated application is able to explicit the necessity of subroutine repeatability, but due to the unsupportive of the PLC to SFC, a conversion of SFC to LAD is a need and it is anticipated that the result would not indicate major differences. The project is pertinent in the sense that comparisons were being done on the performance of which the suitability and applicability of programming languages on sequential applications are being analyzed.

This project is able to be applied in manufacturing or automotive industries where this operation is usually being used in the product transition section from each station. The usage of pneumatic system is able to achieve higher number of operations and improve operational costs.

Chapter 2: Literature Review and/or Theory

IEC 1131-3 (IEC61131-3) standards was developed concerning the blooming of the number of automation vendors, complexity of applications, and the methods of addressing control functions. IEC 1131-3 aims to address many of the limitations of conventional PLC languages by defining a coherent suite of languages and concepts. It encourages well-structured 'top-down' and 'bottom-up' program of development, strong data typing, full execution control support for the realization of complex sequential behaviour, support for data structures, flexible languages selection and vendor independent software elements [1] and [6]

A selection of programming languages are being recognized and supported by this standard. These languages include Instruction List (IL), Structure Text (ST), Function Block Diagram (FBD), Ladder Diagram (LAD) and Sequential Function Chart (SFC). Every language possesses their edges and shortcomings. ST has a better end on the aspect of execution speed, complex mathematics operation implementation and ease of use for newer engineer. Similar with IL, ST also has a greater impact on the acceptance in Europe. LAD on the other hand has the universal acceptance, and it was a solace in code changing. While focusing on the ease of maintenance for end user, processes interlocking and concurrent operations, SFC is a better selection, but LAD and FBD are better for applications that utilize mainly digital I/O and basic processing.

LAD is a graphical representation of the hard-wired electrical wiring diagram. It uses the relay logic to implement Boolean functions. [9] LAD was originated from the automotive industry, where electrical wiring diagrams are used to describe relay control schemes. [1] Due to its easily understandable characteristics, LAD is widely used in conventional as well as modern PLC programs development. LAD is regarded as ladder because of its power lines, or rails, which resemble the vertical sides of a ladder, with the horizontal circuit lines looks like the rung of the ladder, as illustrated in the diagram below.

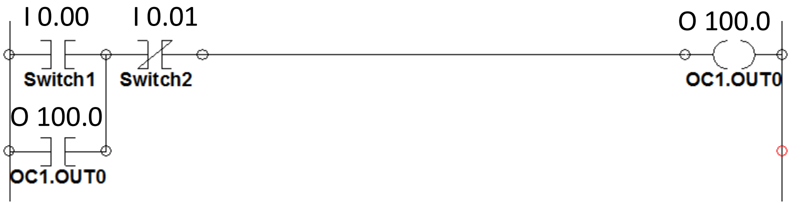


Figure 2. 1 Ladder Diagram

“I” in the figure above represents input and “O” represents output, while “0.00”, “100.0” represent the memory addresses. This mean that memory addresses from 0 to 99 are allocated for input while 100 to 199 are being allocated for output and you will see further on in this report, 200 to 299 are allocated for virtual relays and holding relays.

However, as complexity of PLC functionality has grown, many control applications involve PID, trigonometry, and data analysis. In order to achieve these advancement, LAD program tends to be more complicated and difficult to interpret. Besides, involving hundreds of inputs and outputs in a program eventually caused the program difficult to follow. It is hard to isolate and troubleshoot, unless with extensive documentations.

When a program takes in a lot of counters and timers, LAD tends to get more complex easily. Every timers or counters require a memory bits or holding relays to handle it. Latching structure is a need whenever continuity of a process or stage is to be maintained. Besides, LAD does not support application that involves a lot of subroutines or program blocks. Some logic blocks might be used over and over again. SFC while on the other hand is able to achieve that, providing a high reusability program structure.

SFC or formerly known as GRAFCET, [10] is a graphical method of structuring programs and function blocks, with other four programming languages being recognized by IEC 1131-3 enclosed inside. [11] It consists of three major components: steps, actions and transitions. Steps consists of a bundle of programming logic and it is connected to one or more action blocks which each action block is associated with an action. [9] Transitions can be regarded as a gate or a custom, allowing the program to execute from one step to another. This gate only actives when the steps before it is active; and when active, the transition deactivates the step before it and activates the step after it. Action on the other hand is the unit associated with the action block which connected to the step. Every action is controlled by the action block through action qualifier, with every single qualifier brings about different meanings. A general SFC with feedback is shown as below:

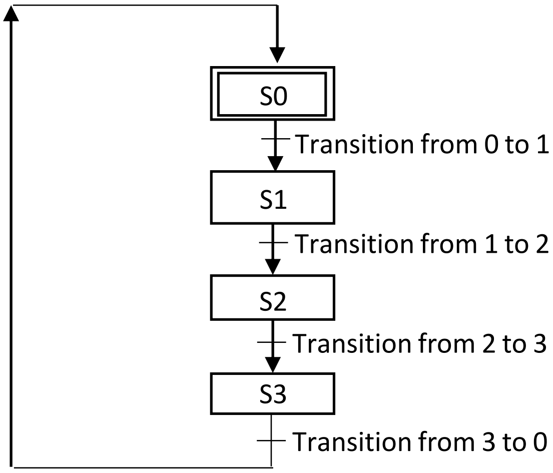


Figure 2. 2 SFC General Structures

SFC is the simplest programming method to implement if the application involves series of repeatable process. For a normal pick and place mechanism, the process is usually in a sequential form. Since there will only be one active piece of code and one transition to be concerned with at a time, condition checking and control of the process should be achievable without large rungs. Taking pick and place mechanism as example, if the arm moved to the object but not picked it, in SFC, attention can be focused on the transition between “move to product” and “pick product”.

SFC is able to perform selection structure or simultaneous configuration, besides sequential, allowing isolating analysis of a program being done conveniently. Furthermore, with a simple action box and all the relevant coding being written inside it literally improve readability. Every step maintains its own step timer, with no duty of starting a specific timer. Therefore, every action is allowed to be running in its own pace, without getting the effect of the coding external of the action box.

There do have the downside for SFC, as not every application possesses sequential behaviour. This type of structure format could added unnecessary complexity.

However, rather than being languages by itself, [6] SFC can be seen as a method for organizing programs, allowing separation of a large program into smaller, more understandable sections. Although some SFC is eventually being converted into Ladder Logic, due to the PLC itself not supporting SFC, it still can be a good way to analyse a problem.

Pneumatic actuators use compressed air to transmit energy in order to perform some mechanical motion tasks. Pneumatic systems are widely used in nowadays industrial automation as it is fast, thus achieving shorter cycle time (higher number of operations) compared to hydraulic systems. Pneumatics is different from hydraulics as hydraulics converts pressurized fluid to mechanical energy. Hydraulics has a greater force and is capable of moving heavier loads. Both use the fluid dynamics concept of pressure.

As pneumatic system uses normal air for compression, it does not have return lines and gases are exhausted into the atmosphere through the pressure relieve valve or the exhaust port of the five ports two ways or three ports two ways directional control valves. Other components for pneumatic equipment set includes cylinder with rod, air compressor, air tank, transition lines, solenoid valves, and some passive components.

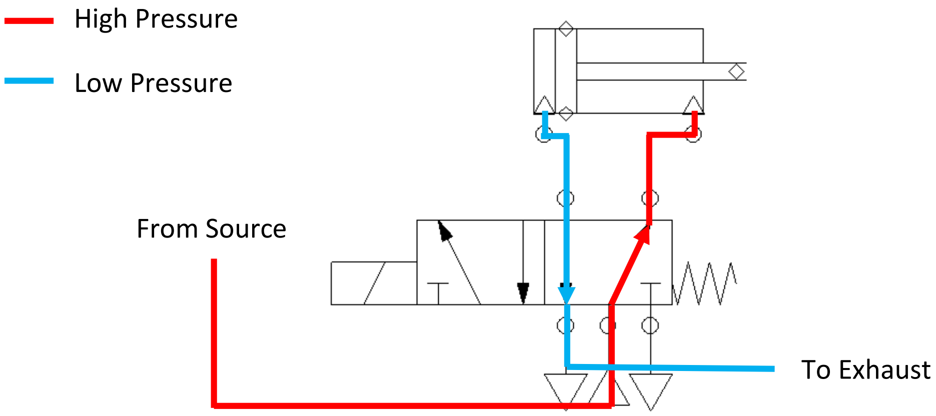


Figure 2. 3 Pneumatic Cylinder Schematics Diagram

A number of cylinders from the pneumatic system are integrated to form a physical arm, as shown below, operating mechanically, to perform certain specific task, thus knowing as pneumatic system mechanical arm. The capability of mechanical arm to perform tasks with high accuracy, and precision has dramatically improve product quality. Utilizing automated mechanical arm is able to speed up the production rate, or maintain the optimum speed without breaks. Furthermore, by replacing those tasks that were normally done by human, is able to create a safer working condition, as the roles of workers had changed from practical to supervisory. By operating the solenoid valves or the directional valves with electricity, the system is addressed as electro-pneumatic actuating robotic mechanical arm.

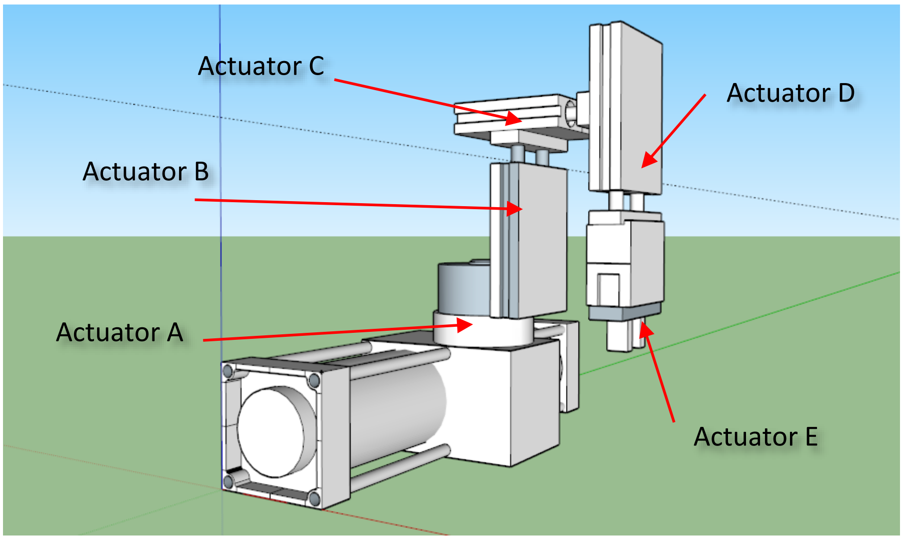


Figure 2. 4 Electro-pneumatic Sketch Diagram (without transition lines)

In this project, electro-pneumatic actuating robotic arm systems were being used to demonstrate the performance of a PLC being developed or coded by programming languages recognized by IEC 1131-3 standard. The robotic arm is expected to work smoothly and is able to achieve the objective of this project. The diagrams of the robotic arm are attached in the appendices section.

This project includes the development of a task-level autonomy system upon activation of the start button, instead of teleportation or supervisory. Since the system only perform simple load transferring task, there is no need for high levels of autonomy. Although much complicated automation systems like welding, painting or components assembly based on coordination could be used as example, this project focuses on the selection of programming methods.

Chapter 3: Methodology / Project Work

3.1 Research Methodology

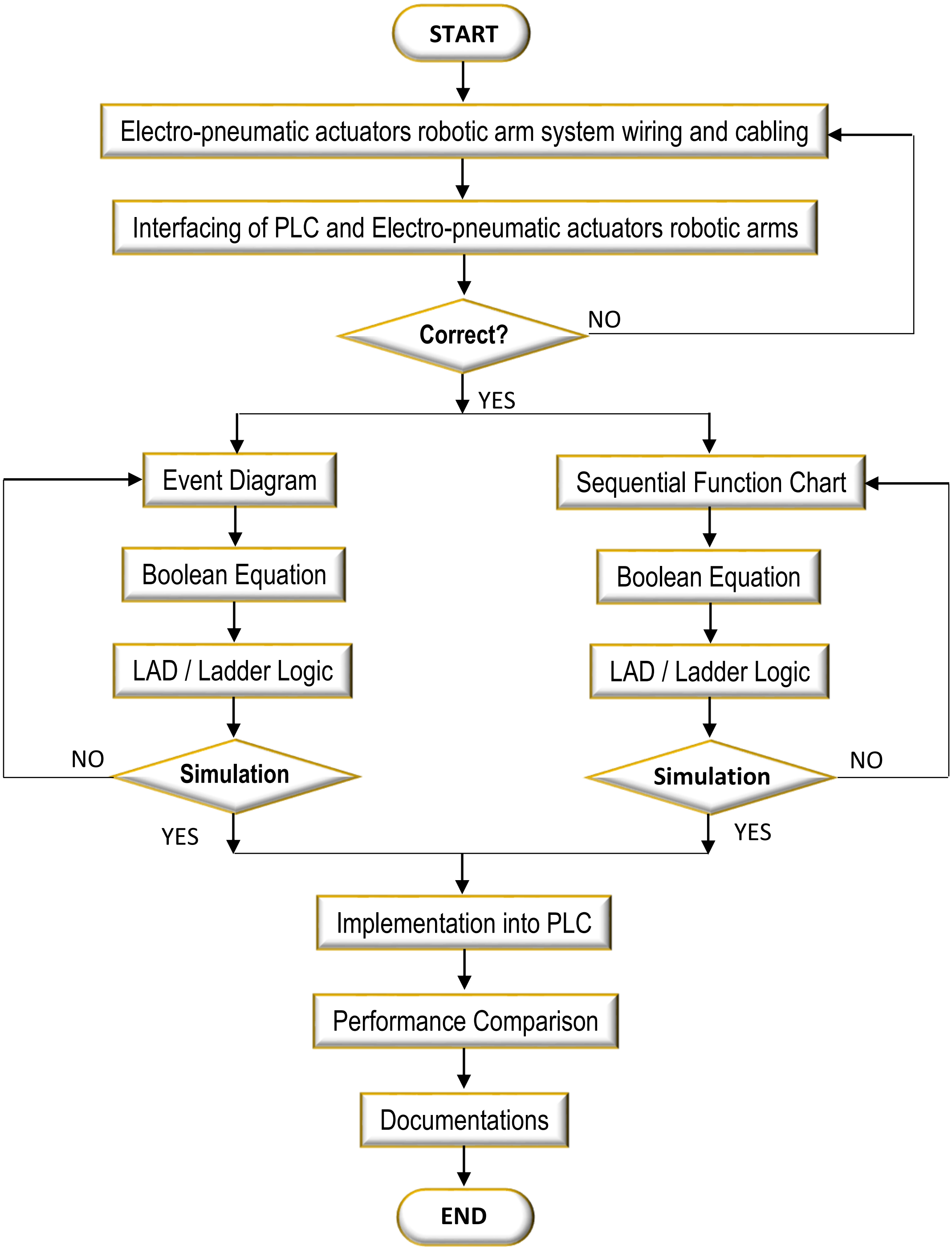


Table 3. 1 Methodology Flow Diagram

In this project, the workstation is to use a mechanical arms operated by compressed air, to pick up an object and moves towards to the other side, putting it down and returning into the original position autonomously. Understanding of the number of actuators used in the automation and how those actuators is to be arranged in order to achieve the specified task is the first most stage in approaching this project.

With the provision of input, output and memory addresses used in the LAD, interfacing between the controller and the mechanical arms pneumatic system is started. Wiring connections are linked from the controller to the solenoid valves and from there cabling between pneumatic components such as air compressor, air tank, transition lines, solenoid valves and the actuators are secured. The set ups are as shown in the picture below. Troubleshooting or reviewing back the cabling and wiring connections are carried out if the output of the mechanical arm falls out of expectation and with that succeed, we proceed to the construction of ladder logic for the PLC.

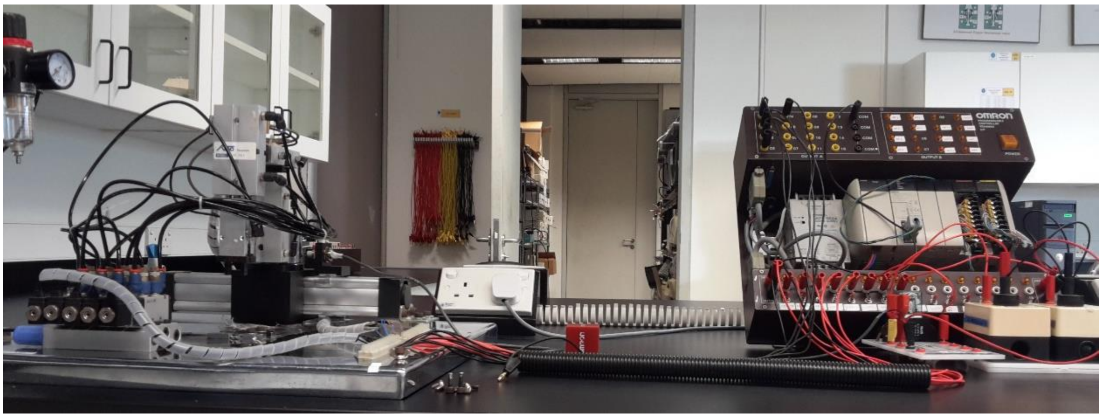


Figure 3. 1 Project Set-up (controller and pneumatic actuating mechanical arm)

An event diagram or movement diagram and a flow chart are being constructed respectively for LAD and SFC. We will first proceed with LAD. Using equations as below:

$$phase\ I = [SET\ \cup\ phase\ I] \cap \overline{RESET} \qquad (1)$$

$$Y_A = [\,SET\ \cup\ Y_A\,]\ \cap\ [\overline{RESET}] \qquad (2)$$

where,

Y_A *represents the actuator*

SET represents the conditions that activate the phase

RESET represents the conditions that deactivate the phase

Boolean Equations for one cycle of the operation can be obtained from the movement diagram for every single phase and actuators and LAD is constructed from these equations. Additional components are added in order for the cycle to repeat itself unlimited until the stop button is pressed. The ladder logic is being simulated using Automation Studio software before loading into the programmable logic controller.

The same methodology is used in approaching the SFC. In this project, a SFC chart is being constructed and converted into Boolean Equations using formulas as below:

$$E_n = (E_{n-1} \cap R_{n-1}) \cup (E_n \cap \overline{E_{n+1}}) \quad (3)$$

where,

E_n *represents the current stage*

E_{n-1} *represents the previous stage*

E_{n+1} *represents the following stage*

R_{n-1} *represents the previous transition*

Ladder logic is constructed from these equations.

With the completion of simulation, the program is loaded into the PLC. Operational performance of the arm as a result of Event Diagram is being compared with the one programmed using SFC method. The performance evaluation can be subjective, branching from the requirement of virtual relays, arrangement of ladder components, smoothness of those movement, to ease of troubleshooting or debugging. Documentation marks the end of the project with all the procedures for the conversion of event diagram and SFC to LAD, Boolean Equation derivations, performance comparison being recorded.

3.2 Project Key Milestone

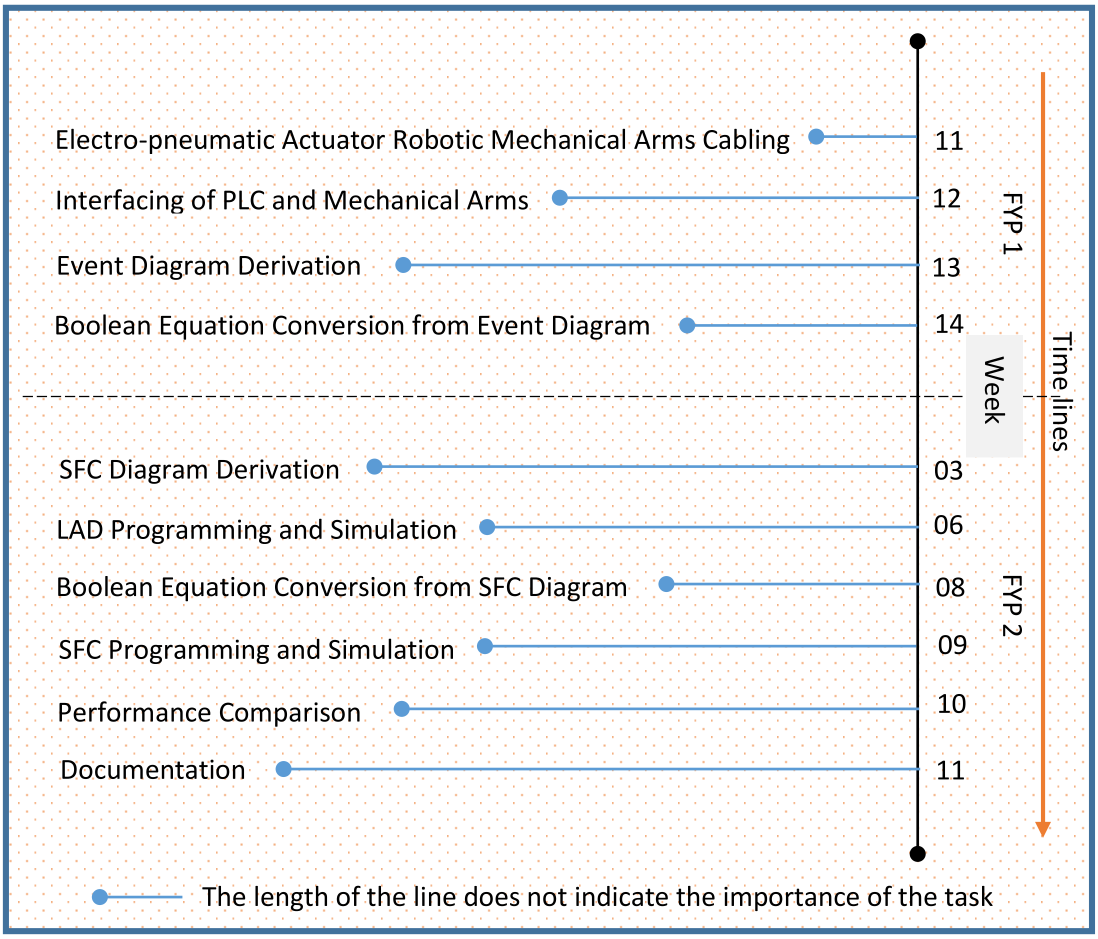

Electro-pneumatic Actuator Robotic Mechanical Arms Cabling
11
Interfacing of PLC and Mechanical Arms
12
Event Diagram Derivation
13
Boolean Equation Conversion from Event Diagram
14
FYP 1
Week
Time lines
SFC Diagram Derivation
03
LAD Programming and Simulation
06
Boolean Equation Conversion from SFC Diagram
08
SFC Programming and Simulation
09
Performance Comparison
10
Documentation
11
FYP 2
The length of the line does not indicate the importance of the task

Figure 3. 2 Project Key Milestone

3.3 Project timeline (Gantt-Chart)

Subject		1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Problem Identification Phase	Selection of Project Topic	🟪	🟪																											
	Understand Title Requirement	🟪	🟪	🟪	🟪																									
	Understand Scope of the Title		🟪	🟪	🟪																									
Preliminary Research Work	IEC 1131-3 Standard		🟪	🟪	🟪	🟪																								
	Electro-pneumatic Robotic Arm Systems Cabling and Wiring						🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪												
	Ultrasonic Sensor Installation					🟪	🟪	🟪	🟪	🟪																				
	Extended Proposal Submission						🟨																							
	Proposal Defence								🟪	🟪																				
Project Development Phase	Movement Diagram and LAD Development										🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪			
	Interim Draft Report Submission													🟨																
	Interim Report Submission														🟨															
Project Implementation Phase	SFC Chart Derivation and LAD Simulation															🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪			
	SFC Simulation															🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪			
	Transfer of program into PLC															🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪			
	Progress Report Submission																					🟨								
Project Evaluation Phase	Performance Evaluation																		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪			
	Feedback																						🟪	🟪	🟪	🟪	🟪	🟪	🟦	
	Pre-SEDEX																								🟨					
	Viva																												🟨	
Documentation Phase	Draft Report Submission																										🟨			
	Dissertation Submission (soft bound)																											🟨		
	Technical Paper Submission																											🟨		
	Project Dissertation Submission (hard bound)																													🟨

Table 3. 2 Gantt chart

🟨 Deadline 🟦 Uncompleted 🟪 Completed

Chapter 4: Result and Discussion

As mentioned in the former section, analysis on the mechanical arm is being done at first. Envisioning how the mechanical arm is picking and placing the objects helps in planning the sequence of the actuators. Figure 2.4 shows the five actuators used in this project. Besides, availability of sensors in detecting the position of the rod or piston will affect the decision in using timers in replace of the missing sensors, as we need a triggering signal to activate the next secondary variable for ladder logic or transition for SFC.

Before proceeding to the construction of the final program, a couple of initial programs are being conducted to test out the actuators. The same procedures applied to these initial programs, which were kick-started with the construction of event diagram and sequential chart, then conversion into Boolean Equations and then construction of ladder logic from these equations. All these are being documented under the appendices section.

4.1 Real and Final Program

Case study: All five actuators are to extend and retract in a sequence that is able to transfer an object from one position to another position. A sequence of movement as shown below are proposed:

C+D+E+ | delay | D-C- | B+A+ | delay | B-C+D+E- | delay | D-C-A-

Pressing the start push button (PB Start) causes the cycle to execute and pressing the stop push button (PB Stop) causes the operation or cycle to stop.

4.1.1 Movement Diagram

One cycle of the actuator movements are being constructed in the movement diagram below and are divided into specific secondary variable or phases. From there, table of secondary variables and outputs are being constructed.

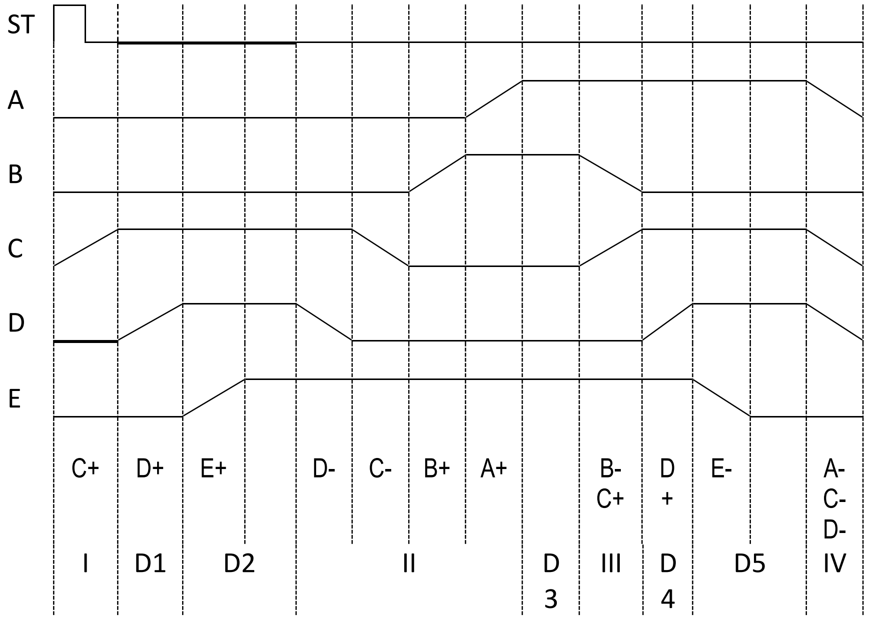


Figure 4. 1 Movement or Event Diagram

Secondary variables	SET	RESET
I	PB	C+
HRT1	C+ and D- and ~~(E+)~~	tim1
HRT2	tim1	tim2
II	tim2	A+
HRT3	A+ and B+ and ~~(HR3)~~	tim3
HR3	tim3	C+ and D-
HRT4	C+ and D- and A+	tim4
HRT5	tim4	tim5
HR4	tim5	C- and D-

Actuators	SET	RESET
Y(A)	II and B+	tim5
Y(B)	II and C-	tim3
Y(C)	I or (D- and III)	(D- and II) or tim5
Y(D)	HRT1 or HRT4	tim2 or tim5
Y(E)	tim1 and HRT2	tim4

Table 4. 1 Table of Secondary variables and outputs

4.1.2 Boolean Equations

From the table of secondary variables and outputs, Boolean Equations are derived using these equations:

$$phase\ I = [SET\ \cup\ phase\ I] \cap \overline{RESET}$$

$$Y_A = [\,SET\ \cup\ Y_A\,]\ \cap\ [\overline{RESET}]$$

$$HR1 = (PB\ \vee HR1) \wedge \overline{(C+)} \tag{4}$$

$$HRT1 = ((C + \wedge D - \wedge\ \overline{E+})\ \vee HRT1) \wedge\ \overline{(tim1)} \tag{5}$$

$$tim1 = HRT1(2secs) \tag{6}$$

$$HRT2 = (tim1\ \vee HRT2) \wedge\ \overline{(tim2)} \tag{7}$$

$$tim2 = HRT2(3secs) \tag{8}$$

$$HR2 = (tim2\ \vee HR2) \wedge\ \overline{(A+)} \tag{9}$$

$$HRT3 = ((A + \wedge B + \wedge\ \overline{HR3})\ \vee HRT3) \wedge\ \overline{(tim3)} \tag{10}$$

$$tim3 = HRT3(1sec) \tag{11}$$

$$HR3 = (tim3\ \vee HR3) \wedge\ \overline{(C + \wedge D -)} \tag{12}$$

$$HRT4 = ((A + \wedge C + \wedge D-)\ \vee HRT4) \wedge\ \overline{(tim4)} \tag{13}$$

$$tim4 = HRT4(2secs) \tag{14}$$

$$HRT5 = (tim4\ \vee HRT5) \wedge\ \overline{(tim5)} \tag{15}$$

$$tim5 = HRT5(3secs) \tag{16}$$

$$HR4 = (tim5\ \vee HR4) \wedge\ \overline{(C - \wedge D -)} \tag{17}$$

$$Y(A) = \big((HR2\ \wedge B+) \vee Y(A)\big) \wedge \overline{(tim5)} \tag{18}$$

$$Y(B) = ((HR2\ \wedge C-)\ \vee Y(B)) \wedge \overline{(tim3)} \tag{19}$$

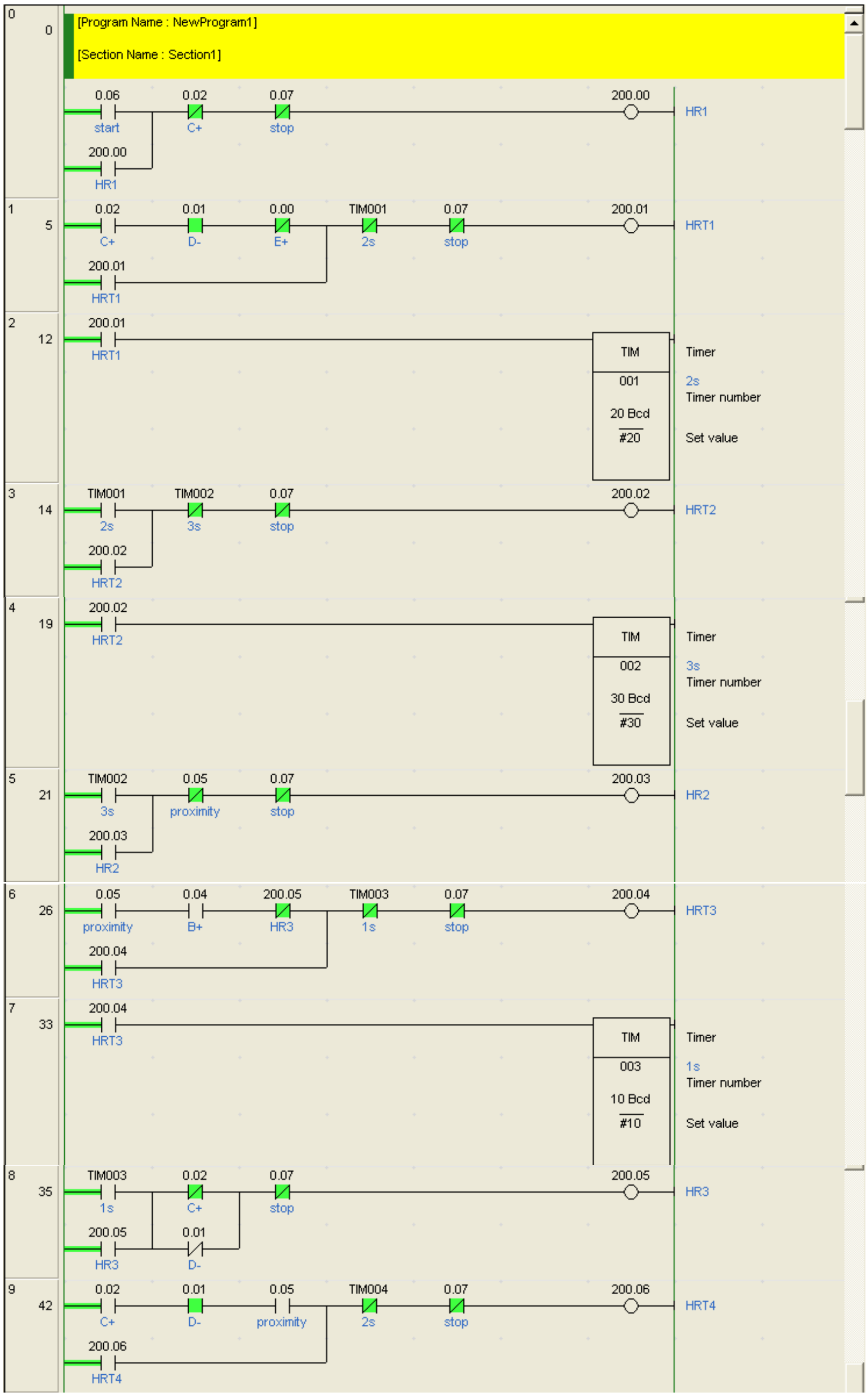
$$Y(C) = (HR1\ \vee (D - \wedge HR3) \vee Y(C)) \wedge \overline{\big((D - \wedge HR2)\ \vee tim5\big)} \tag{20}$$

$$Y(D) = (HRT1\ \vee HRT4\ \vee Y(D)) \wedge \overline{(tim2\ \vee tim5)} \tag{21}$$

$$Y(E) = \big((tim1\ \wedge HRT2) \vee Y(E)\big) \wedge \overline{(tim4)} \tag{22}$$

4.1.3 Ladder Diagram

From the Boolean Equations derived in the previous section, ladder logic is constructed and is shown below:


[Program Name : NewProgram1]

[Section Name : Section1]

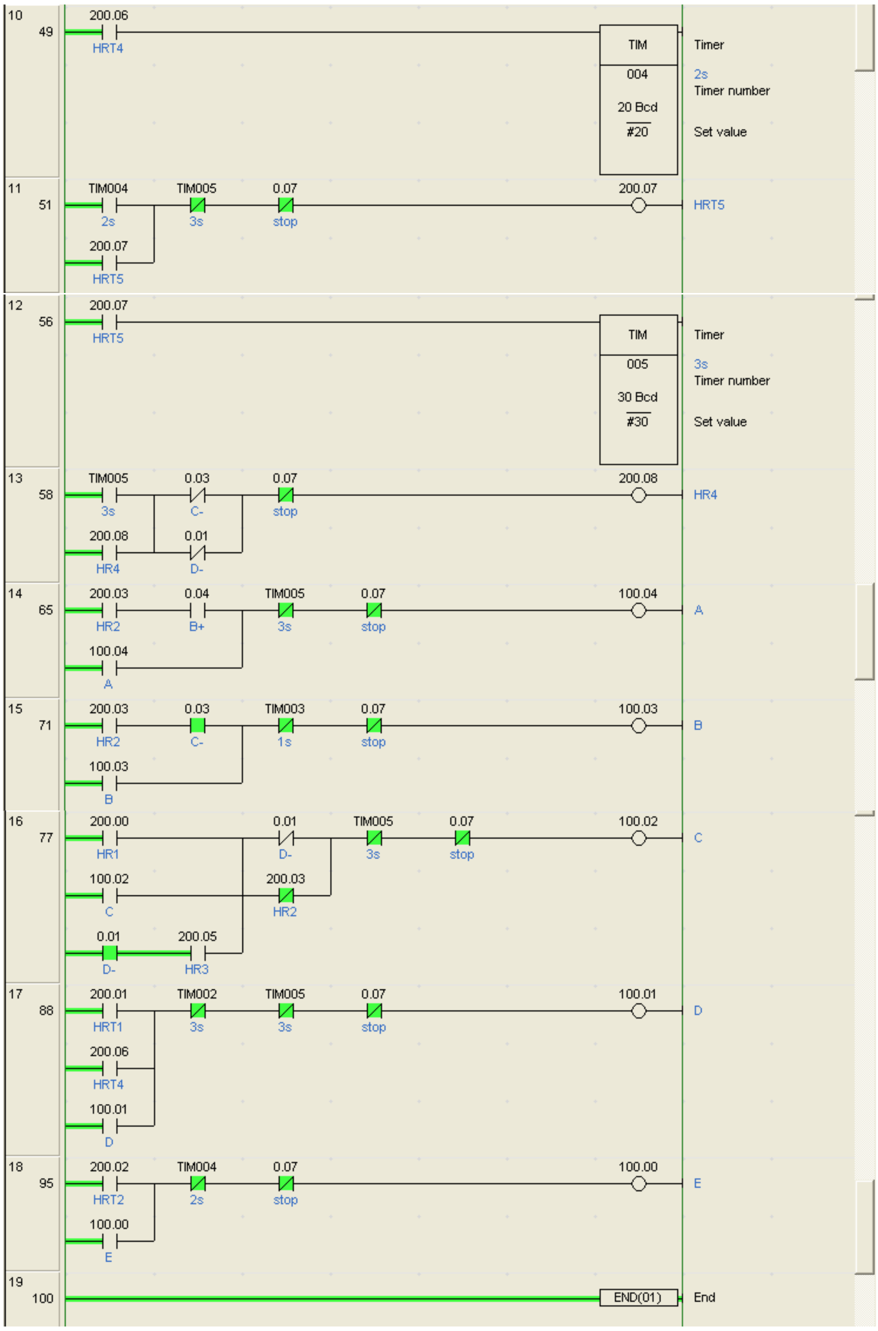


Figure 4. 2 Ladder Logic

4.1.4 Sequential Function Chart

The actuator movements are being planned in the SFC, with the transitions only turn on when the respective conditions are fulfilled. The last transition is feed-backed to the initial step.

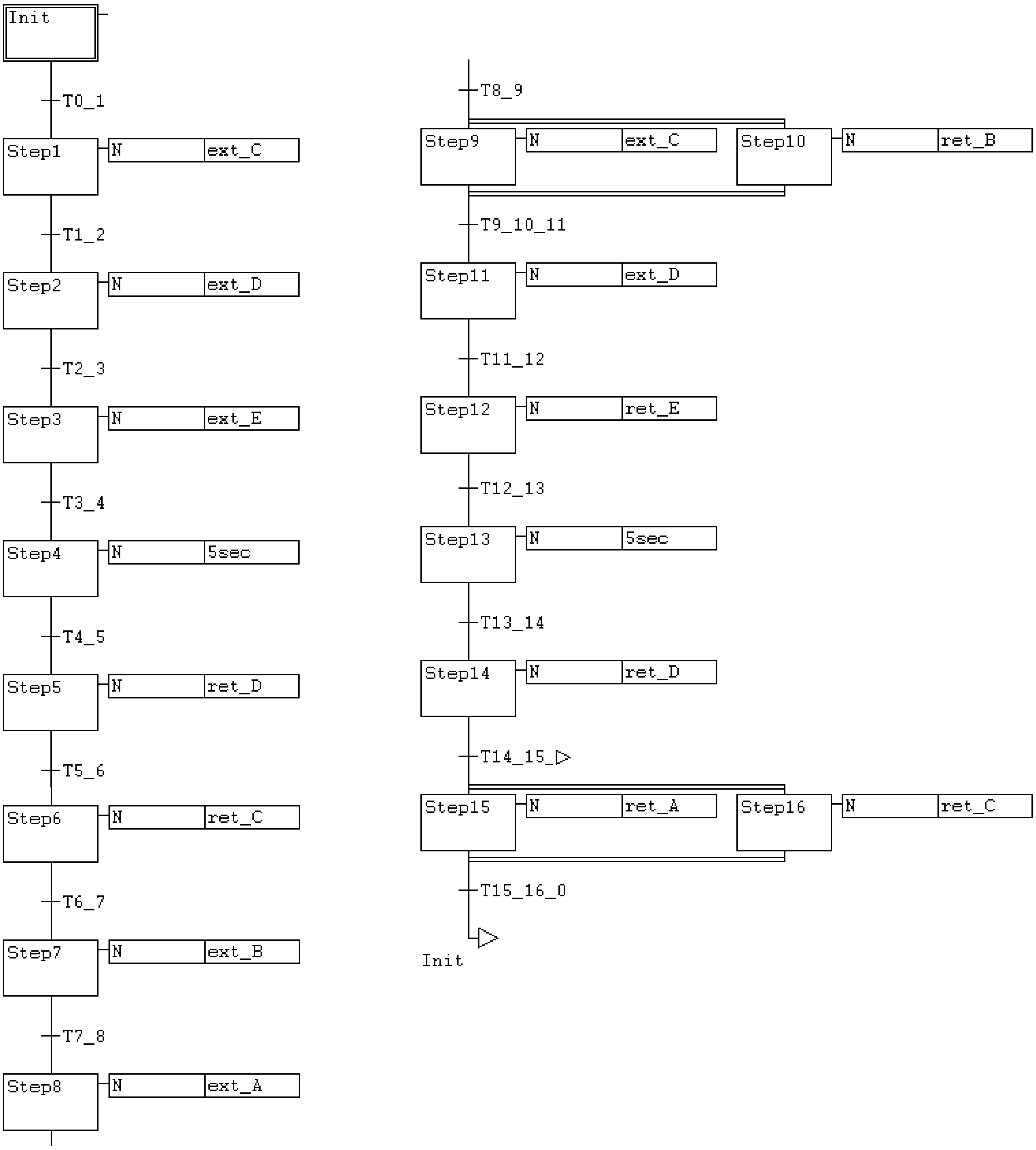


Figure 4. 3 Sequential Function Chart (SFC)

4.1.5 SFC equivalent LAD

With that, the completed program is shown as shown as below, and with the simulation diagram attached in the appendices section. With the start button to initialize the *start_enable_relay*, the program is being latched on using this holding relay.

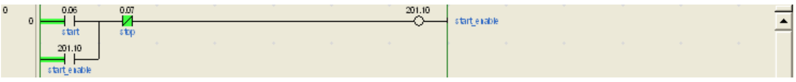
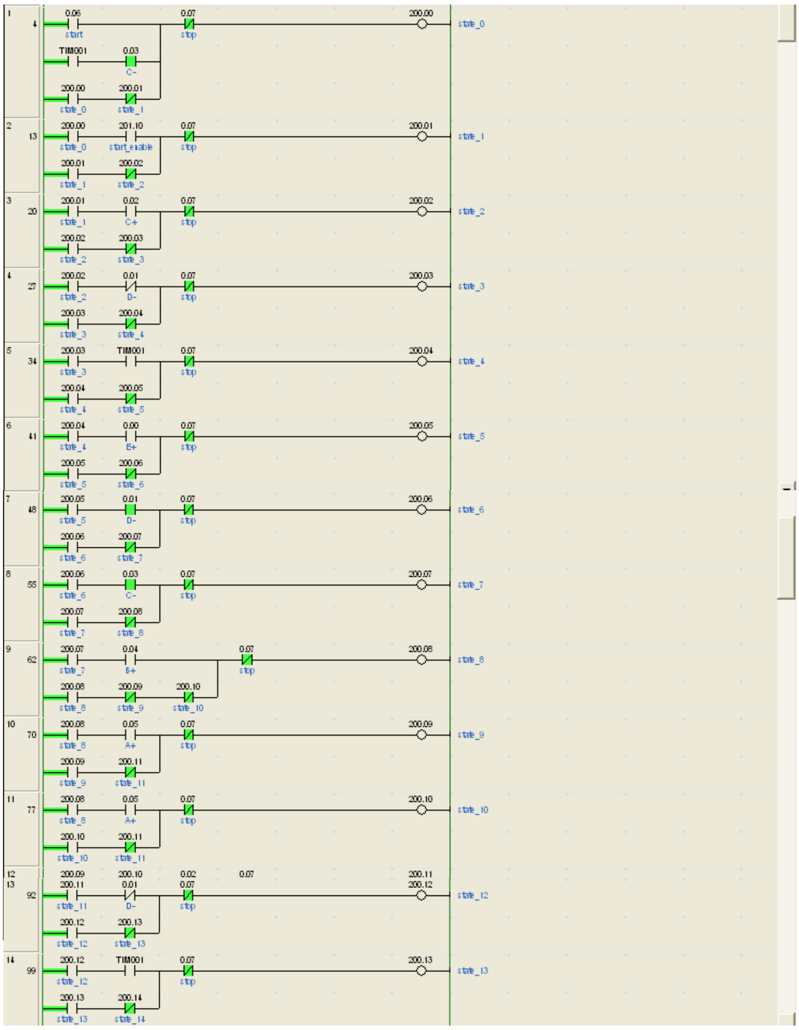


Figure 4. 4 initial condition

For the state conditions section, with each *state* and its respective *actuator in position* or with the *timer finished* or *timer disable signal* as the *normally open* contacts, the succeeding state is being turned on. Each state is being latched by itself and it turns off the preceding state.



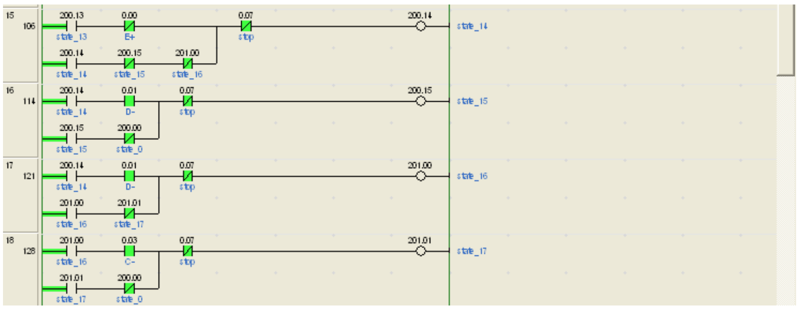
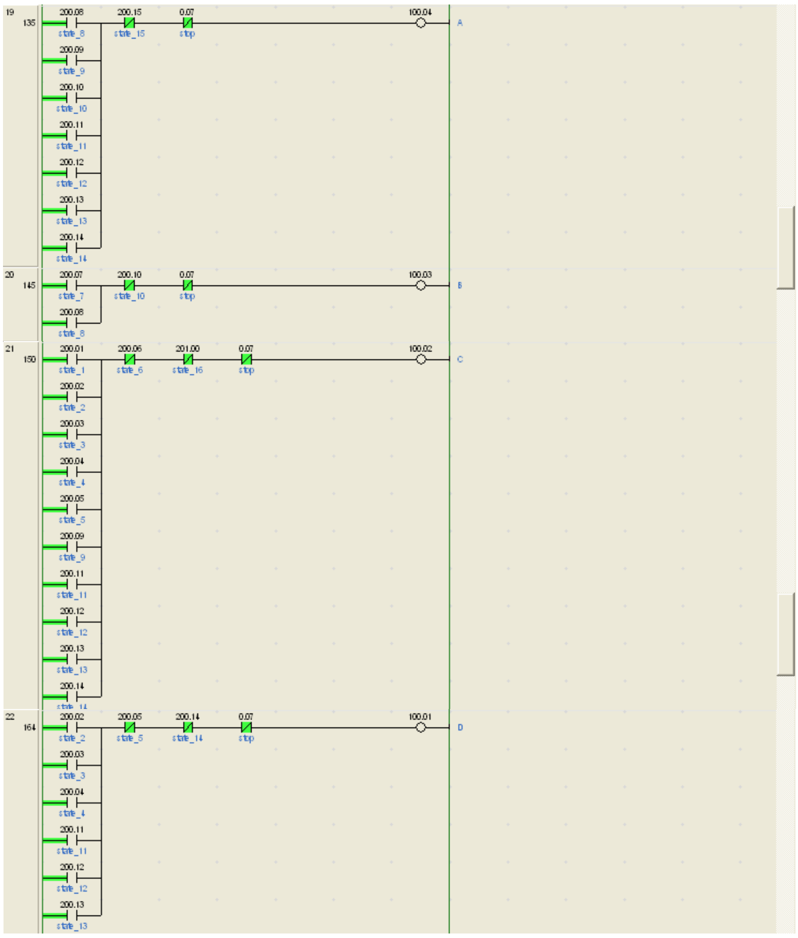


Figure 4. 5 State Conditions

For the output section, with the *states* that is responsible for firing up a specific actuator being connected to the *output coil*, the coil is initialized. However, in order to maintain the active mode of the outputs, the following *states* after that respective *state* which the actuator is required to be continually extending are needed to be connected to the *output* as well.



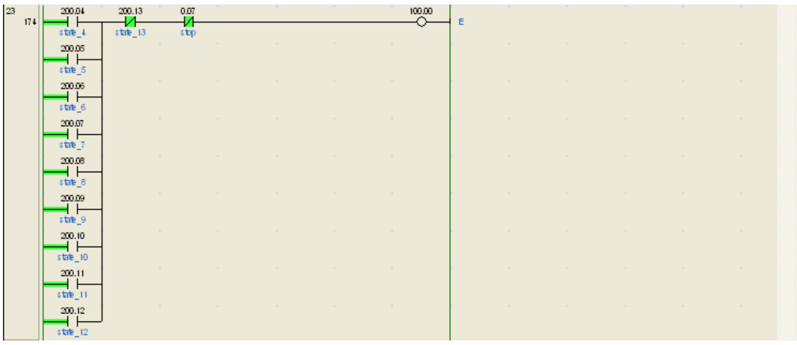


Figure 4. 6 Outputs

Whatever timers and counters used in the program are being categorized in this following section. Just like the output, which states are used to turn on the timer are being connected to the timer block. And in SFC, whichever states that require the same duration of delay can be connected together in OR configuration to the timer block.

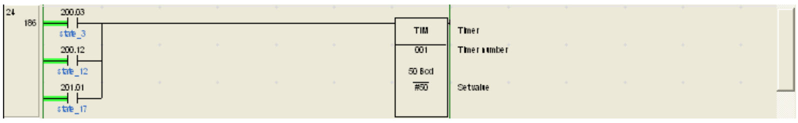


Figure 4. 7 Timer delay action block

4.2 Performance Comparison

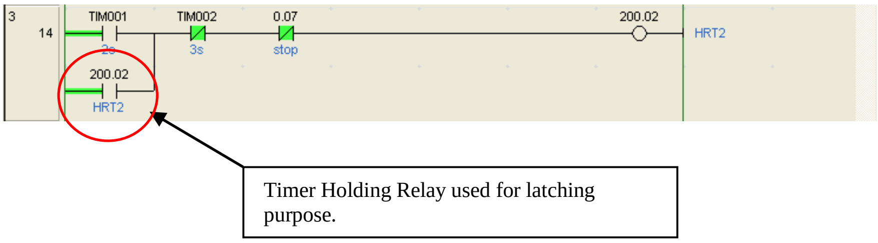
Based on the observation of the resultant movement of the electro-pneumatic actuating robotic mechanical arm, under the same actuator sequences, both languages are able to achieve the same performance. Although base on personal evaluation, SFC seems to have a smoother performance, however, this unproven standard should be put aside in order to have a fair comparison for both languages. With the draw match between both languages, attention is being focused on next section, which is the program structures.

4.2 Program Structure Comparison

In this project, since there are two version of the LADs, which are the one created by conventional LAD and the one derived from SFC, comparisons were done based on these two version of LADs.

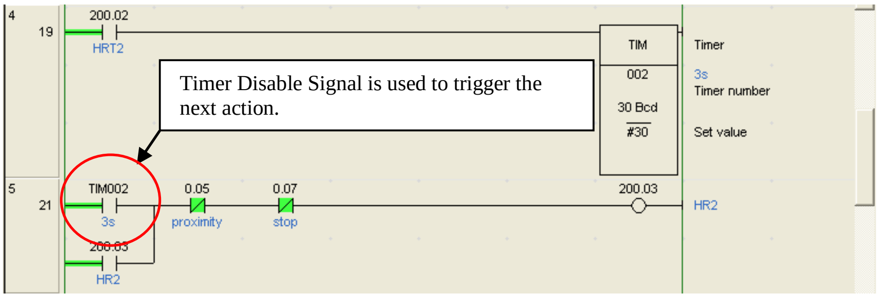
4.2.1 Re-usability of Timer or Counter Action Block

From the figure above, which illustrate the timer program structure of LAD, we can realize that every timer in LAD needs a holding relay for latching purpose. Since the triggering signal will stopped in certain period after it has initialized the *coil*, we need a constant signal to power up the timer, thus, we need the *holding relay*. Besides, in LAD, the *timer disable signal*, which is the signal that turned on when timer finished its timing, is used to trigger the next action, so if we are using the same timer for every same duration of delay, we might trigger other *virtual relays* which link to actuator outputs or cause the *timer-disable-signal-triggering-states* to repeat again.



Timer Holding Relay used for latching purpose.

Figure 4. 8 Timer Holding Relay for Latching

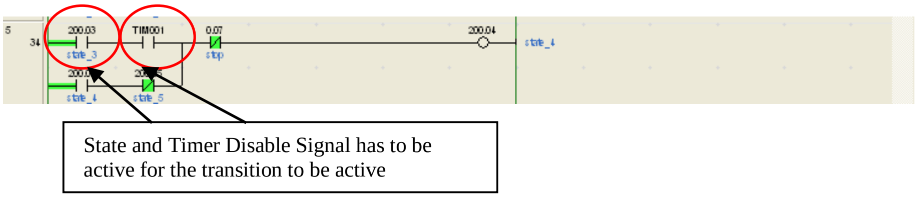


Timer Disable Signal is used to trigger the next action.

Figure 4. 9 Utilization of timer holding relay

Therefore, conclusion were made on this section that conventional LAD does not allow the reusability of timer action block.

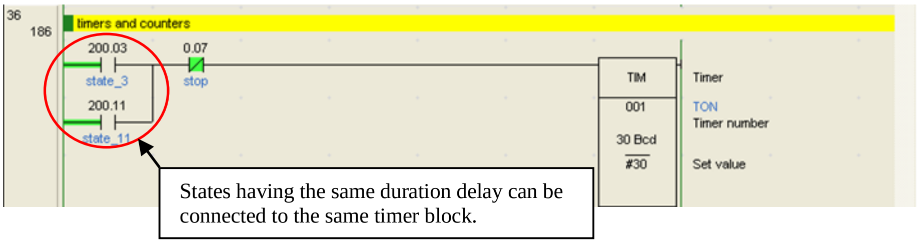
However, for SFC, there is a *transition* separating every *state*, and when being converted into equivalent LAD, the program needs the *state* and the *timer disable signal* to be active in order for the proceeding *state* to be active.



State and Timer Disable Signal has to be active for the transition to be active

Figure 4. 10 Reusing Timer Action Block

Therefore, utilizing the same *timer action block* will only trigger the *time disable signal*, but without the active state, the condition of the transition is not fulfilled, thus, the program will not proceed to the next state. Therefore, from here, we know SFC equivalent LAD enables us to reuse the timer action block. We can connect those states that require the same duration of time delay to one timer action block.

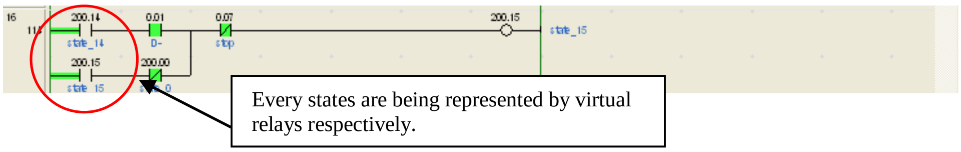


States having the same duration delay can be connected to the same timer block.

Figure 4. 11 Re-usability of Timer Action Block

4.2.2 Usage of Virtual Relays

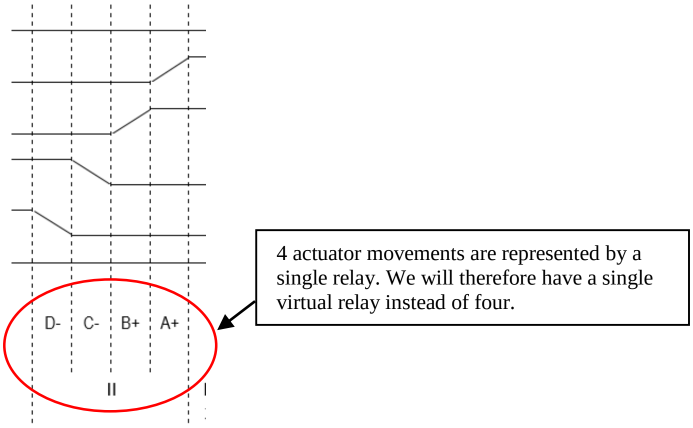
When being converted into the equivalent LAD, every step in the SFC program is being treated as single phase and every phase is being represented by a virtual relay.



Every states are being represented by virtual relays respectively.

Figure 4. 12 Virtual Relay representation for transition and state

While for conventional LAD, as illustrated by the *movement* or *event diagram* in *Figure 4.1*, a sequence of different actuator movements are being represented as a single phase. Therefore, comparatively, conventional LAD has a lesser number of virtual relays used and if a program or an application tends to increase in complexity, SFC equivalent LAD will consumes more memories.



4 actuator movements are represented by a single relay. We will therefore have a single virtual relay instead of four.

Figure 4. 13 Virtual relay presentation for conventional LAD

4.2.3 Program Complexity

Most of the industrial applications can be categorized into or contain the following three configurations:

- sequential,
- parallel, and
- selective

In this project, both languages are able to achieve the first two configurations, which are the sequential and parallel. Appendices III demonstrates the selective configuration, which an alarm indicator used will sound when the actuator E extended but not fully extended. From the result indicated, it shown that both languages are able to perform the selective configuration also. Therefore, with the capability of both languages to tackle these three configurations, complexity became the attention of this section.

Judging from the visual aspect of the program structures, SFC equivalent LAD is much more complex compared to conventional LAD. However, with detailed attention paid on the program, one can realize that the program structure is basically divided only into states condition and outputs, and every state is rigidly being triggered and stopped by its preceding state and succeeding state respectively, thus it is all the same for every problem, depending on the number of states used. While due to the phase in conventional LAD representing a sequence of different actuator movements, there are three layering in the program structure, which are phases, state conditions and outputs. Therefore, troubleshooting and debugging is more troublesome for conventional LAD compared to SFC equivalent LAD.

4.3 Summary

Summing up for the result and discussion, the most important advantage about this research is that using conversion of SFC into LAD, we are able to use SFC in those controllers and software compilers which do not support SFC method. The flowchart resemblance SFC gives us the gist of the process flow in a single glance. Complex program logic can be modelled effectively using a flowchart. [12] Although from the comparison described in the previous section indicates that the equivalent LAD is much more complex than conventional LAD, but with the flow chart resemblance characteristics of SFC, troubleshooting can be done on the SFC layer instead of the converted LAD. Diagramming the user's experience as they navigate through the program is a valuable prerequisite. [12]

With the encapsulation capability of SFC, which enable user to hide or bundle certain number of their programming components or information within the program blocks, program structure can be further simplified. This directly make troubleshooting easier or debugging easier. With this, conclusion were made that SFC is a better selection when dealing with sequential programming and sequential based type of applications.

Chapter 5: Conclusion and Recommendation

In conclusion, this project demonstrates the usage of two programming languages being recognized by IEC 1131-3 (IEC61131-3) in programming a PLC in controlling an electro-pneumatic actuator robotic mechanical arm. The performance of application as a result of these languages as well as the programming structures are being compared.

SFC was designed aims in tackling sequential problems and the flowcharts resemblance features of SFC were a mainstay of procedural programming. [12] However, the result cannot be taken as it represents the whole, as in this paper that the SFC is being converted into LAD. This is because the programming software available for this project does not support SFC.

For the part of this project in which sequential programming is being planned using SFC, it is then being entered into the PLC in the form of ladder logic. By one way, the program can be highly structured, standardised and easy to debug and modify, while the familiarity of ladder logic is preserved. [11] By another way, the non-supportive of SFC in older version controllers are still possible to be programmed using SFC, with the utilization of SFC equivalent LAD.

The choice of selecting either of the programming languages depends on programmers' own preferences. Strong fundamental knowledge about a specific languages and years of experience using that languages will actually produce a more effective software structure, with lesser bugs. Although continuous learning new things is good as it transforms one into a more competent person, but factor like PLC platform, memory capacity or processor speed of a PLC will influence the choice of languages.

Entitlement to decide which languages work best for the application should be given to programmers. The selection of hardware and software according to the application should not be constrained to company available resources as well. This will eventually ease maintenance and problem troubleshooting, as well as technological migration.
For future recommendation, a more complex or sophisticated system can be the focus of this project, which involves greater amount of automation controlling, and different

end effector, such as spinning, welding, or vacuum-based gripping, instead of conventional vacuum gripping. Human interfacing through user interface can be enrolled into the system design, which allow the users to manipulate the system.

Chapter 6 References

[1] R.W. Lewis, Programming Industrial Control Systems using IEC 1131-3. IEEE Control Engineering Series. 1998.

[2] K-H. John, and M. Tiegelkamp, IEC 1131-3: Programming Industrial Automation Systems. Springer. 2001.

[3] E, Estevez, M. Marcos, Member, IEEE, N. Iriondo, D. Orive. Graphical Modeling of PLC-based Industrial Control Applications.

[4] A.P. Kalogeras, C. Diedrich, P. Neumann, G.D. Papadopoulos, Function Block definition based on the IEC 1499 Language.

[5] M. Maslar, PLC Standard Programming Languages: IEC 1131-3: Rockwell Software Inc.

[6] Hajarnavis, Vivek: An evaluation and comparison of PLC Programming Techniques: Innovation Report. University of Warwick. 2006.

[7] G. Frey and L. Litz. Formal Methods in PLC Programming, Nashville, Oct 8-11, 2000.

[8] I.A. Antoniadis and V.I.N. Leopoulus, A concept for the integrated process description, PLC programming & simulation using Petri-Nets: Application in a production process. Proc. of the IEEE, SMC, 2000.

[9] S. Johannsson, M. Ohman, Karl-Erik Arzen, Implementation Aspects of the PLC Standard IEC 1131-3: p. 3, 1998.

[10] R. David, Alla, H., Petro Nets and Grafcet: Tools for modelling discrete events systems: 1992.

[11] K. Collins, PLC Programming for Industrial Automation.

[12] H. Nicholas, The Top 5 Reasons to Use Flowcharts. BreezeTree Software. Retrieved from: http://www.breezetree.com/articles/top-reasons-to-flowchart.htm

Chapter 7 Appendices

7.1 Appendix I: Testing of Actuators A and B

Case study: actuator A and B are to extend and retract in a sequence of A+B+B-A-. Pressing the start push button (PB Start) causes the cycle to execute and pressing the stop push button (PB Stop) causes the operation or cycle to stop.

7.1.1 Movement Diagram

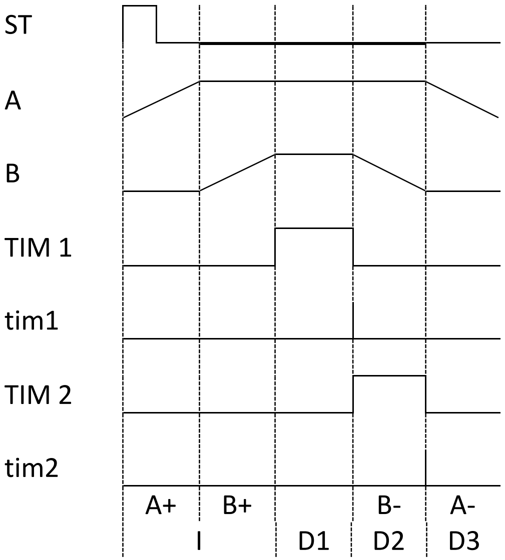


Figure 7. 1 Movement Diagram for AB Testing

Second variables	SET	RESET
HR1	PB	B+
HRT1	B+	tim1
HRT2	tim1	tim2
HRT3	tim2	tim3

Actuators	SET	RESET
Y(A)	HR1	TIM3
Y(B)	A+ and HR1	tim1

Table 7. 1 Table of Secondary Variables and Outputs

7.1.2 Boolean Equations

$$\boldsymbol{HR1} = (\boldsymbol{PB} \vee \boldsymbol{HR1}) \wedge \overline{(\boldsymbol{B}\,+)}$$

$$\boldsymbol{HRT1} = (\boldsymbol{B} + \vee \boldsymbol{HRT1}) \wedge \overline{(\boldsymbol{tim1})}$$

$$\boldsymbol{tim1} = \boldsymbol{HRT1}(\boldsymbol{5s})$$

$$\boldsymbol{HRT2} = (\boldsymbol{tim1}\ \vee \boldsymbol{HRT2}) \wedge \overline{(\boldsymbol{tim2})}$$

$$\boldsymbol{tim2} = \boldsymbol{HRT2}(\boldsymbol{2s})$$

$$\boldsymbol{HRT3} = (\boldsymbol{tim2}\ \vee \boldsymbol{HRT3}) \wedge \overline{(\boldsymbol{tim3})}$$

$$\boldsymbol{tim3} = \boldsymbol{HRT3}(\boldsymbol{1s})$$

$$\boldsymbol{Y}(\boldsymbol{A}) = (\boldsymbol{HR1} \vee \boldsymbol{Y}(\boldsymbol{A})) \wedge \overline{(\boldsymbol{TIM3})}$$

$$\boldsymbol{Y}(\boldsymbol{B}) = ((\boldsymbol{A} + \wedge \boldsymbol{HR1})\ \vee \boldsymbol{Y}(\boldsymbol{B})) \wedge \overline{(\boldsymbol{tim1})}$$

7.1.3 Ladder Diagram

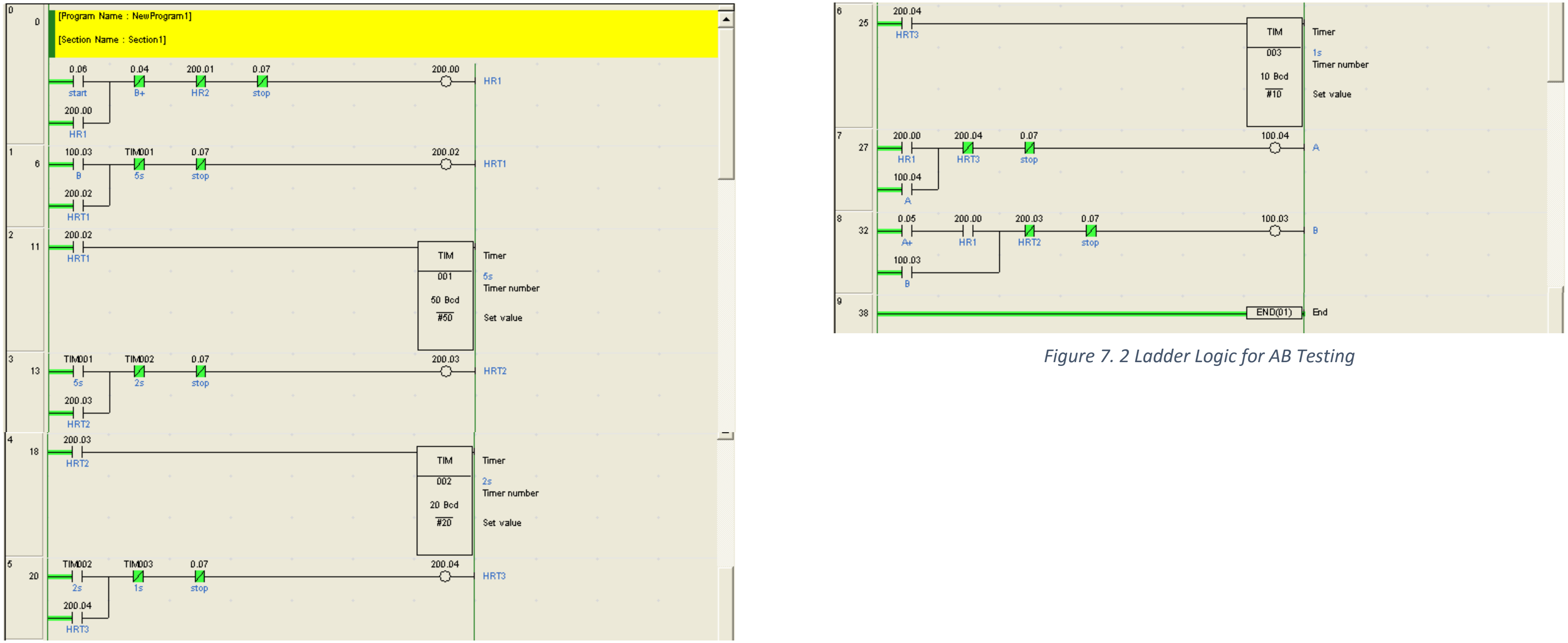


Figure 7. 2 Ladder Logic for AB Testing

7.1.4 Sequential Function Chart

Illustration of the Sequential Flow Chart for Actuators A and B

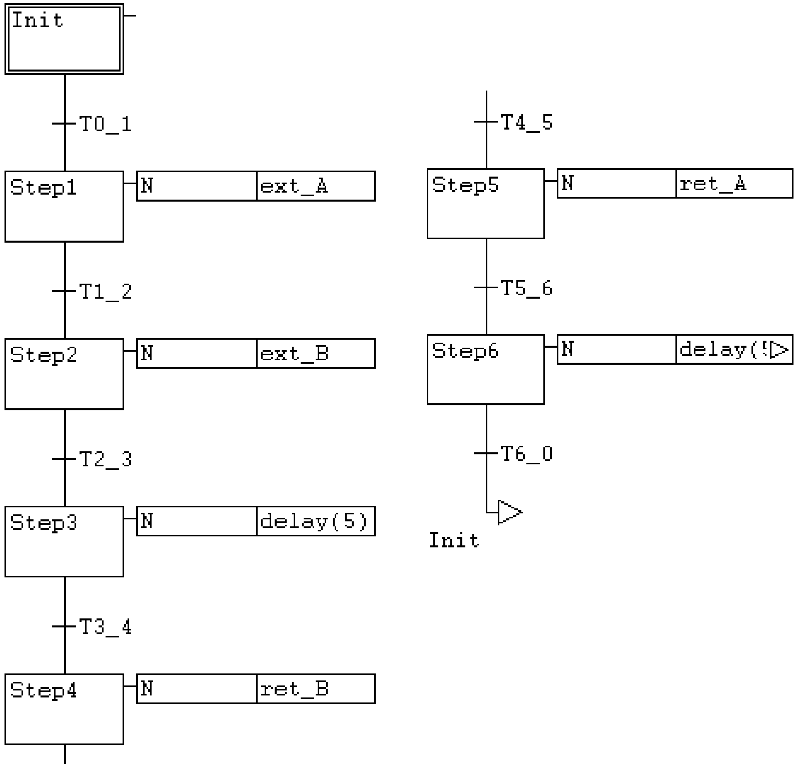
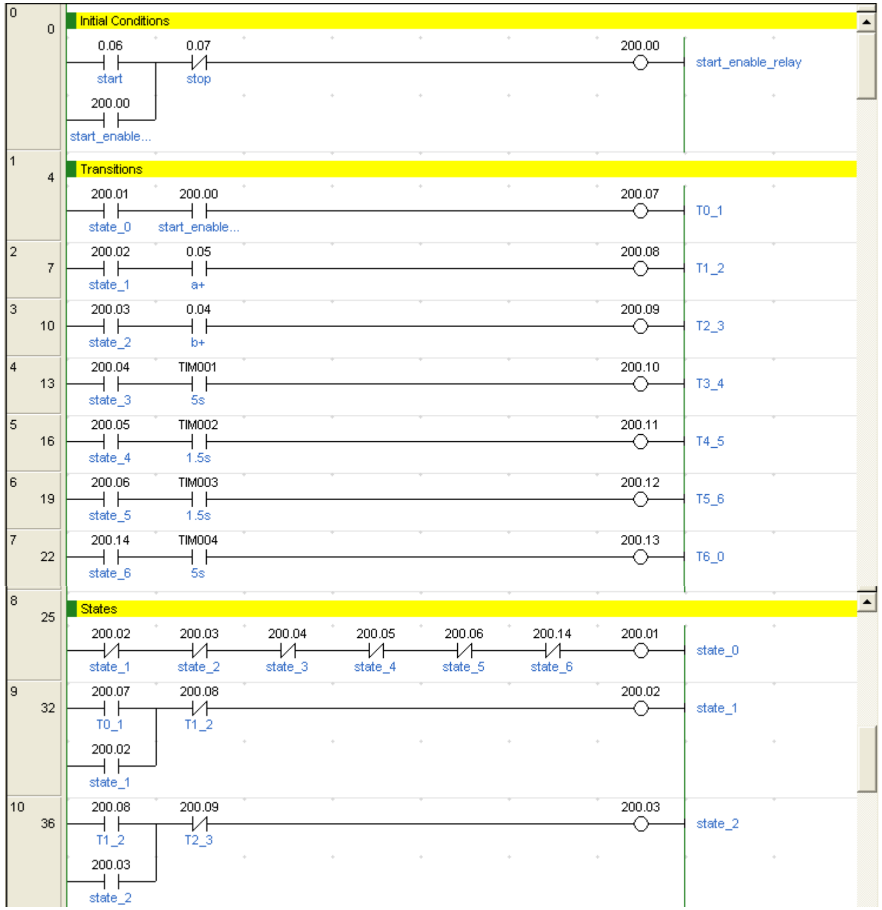


Figure 7. 3 Sequential Function Chart for Act. AB

7.1.5 SFC equivalent LAD (method 1)



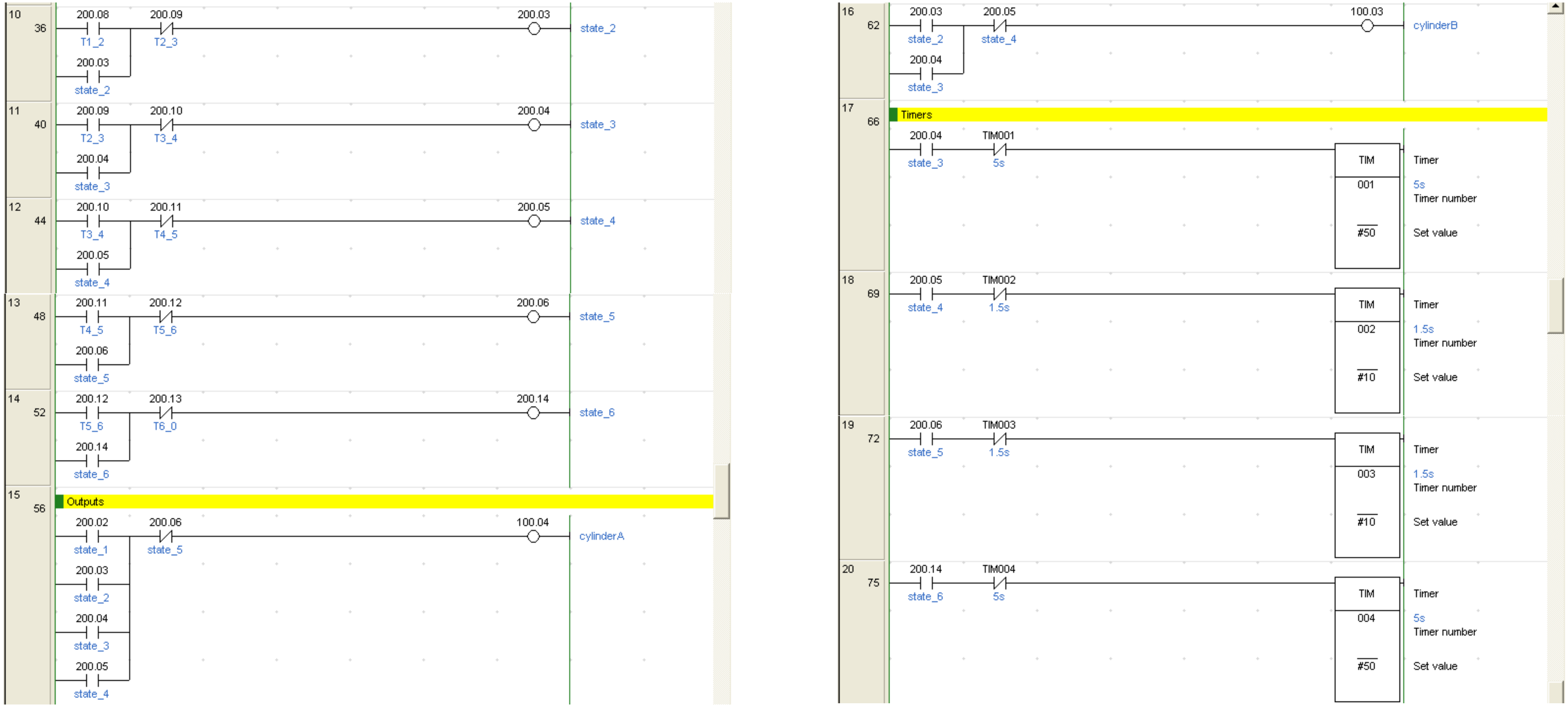


Figure 7. 4 SFC Equivalent LAD (method 1)

7.1.6 SFC equivalent LAD (method 2)

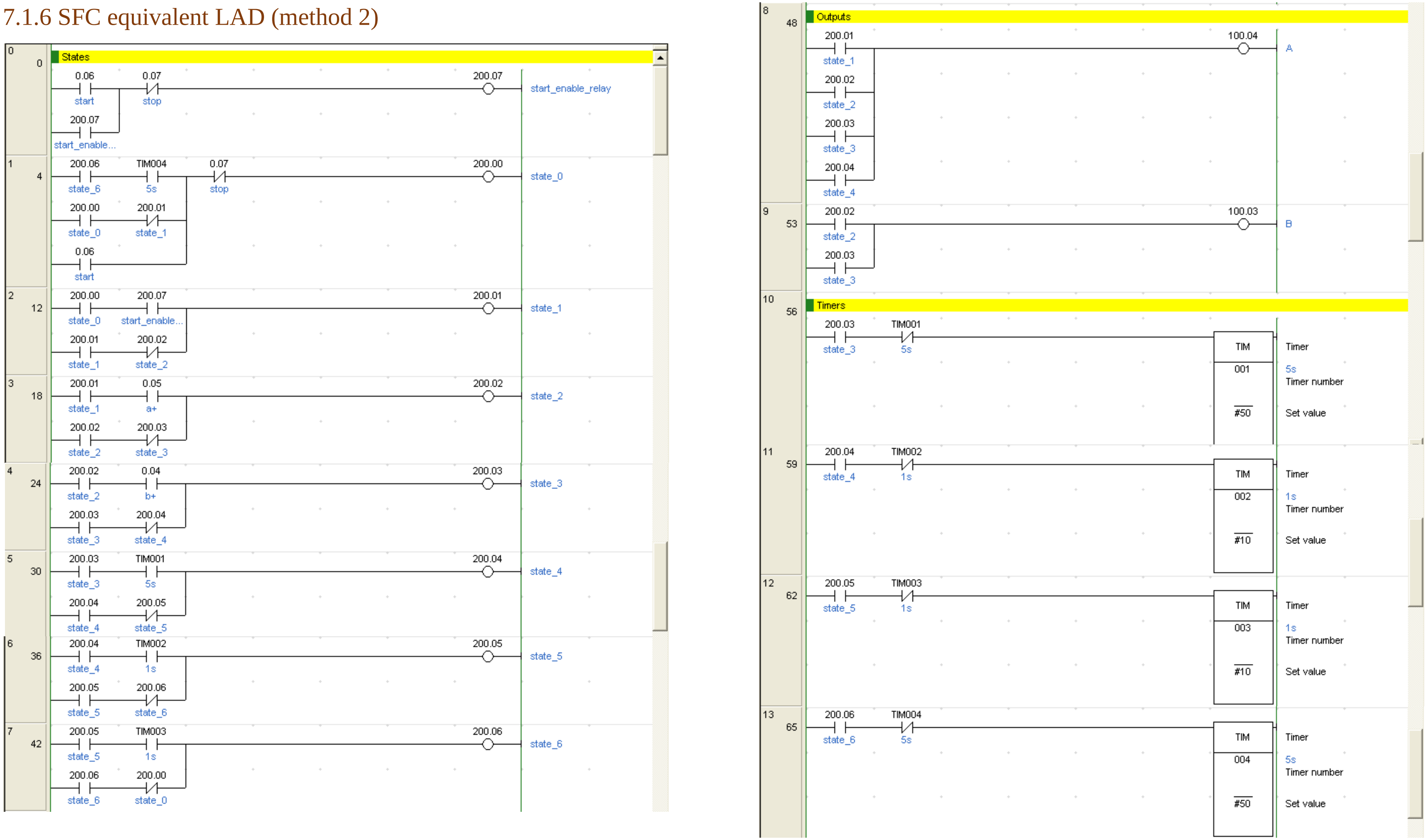


Figure 7. 5 SFC Equivalent LAD (method 2)

7.2 Appendix II: Testing of Actuators C, D and E

Case study: actuator C, D and E are to extend and retract in a sequence of

C+D+E+ | 5s | D-C- | D+E- | D-. Pressing the start push button (PB Start) causes the cycle to execute and pressing the stop push button (PB Stop) causes the operation or cycle to stop.

7.2.1 Movement Diagram

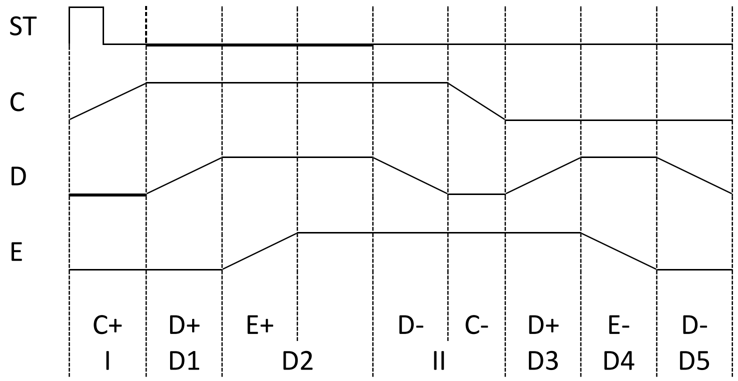


Figure 7. 6 Movement Diagram for CDE Testing

Second variables	SET	RESET
HR1	PB	C+
HRT1	C+ and D- and (~~E+)~~	tim1
HRT2	tim1	tim2
HR2	tim2	C-
HRT3	C- and D- and E+	tim3
HRT4	tim3	tim4
HRT5	tim4	tim5

Actuators	SET	RESET
Y(C)	HR1	D- and HR2
Y(D)	HRT1 or (C- and E+)	tim2 or tim4
Y(E)	tim1	tim3

Table 7. 2 Secondary Variables and Outputs

7.2.2 Boolean Equations

$$\boldsymbol{HR1 = (PB \vee HR1) \wedge \overline{(C+)}}$$

$$\boldsymbol{HRT1 = \big((C + \wedge D - \wedge \overline{E+}) \vee HRT1\big) \wedge \overline{(tim1)}}$$

$$\boldsymbol{tim1 = HRT1(2s)}$$

$$\boldsymbol{HRT2 = (tim1 \vee HRT2) \wedge \overline{(tim2)}}$$

$$\boldsymbol{tim2 = HRT2(5s)}$$

$$\boldsymbol{HR2 = (tim2 \vee HR2) \wedge \overline{(C-)}}$$

$$\boldsymbol{HRT3 = ((C- \wedge D- \wedge E+) \vee HRT3) \wedge \overline{(tim3)}}$$

$$\boldsymbol{tim3 = HRT3(5s)}$$

$$\boldsymbol{HRT4 = (tim3 \vee HRT4) \wedge \overline{(tim4)}}$$

$$\boldsymbol{tim4 = HRT4(1s)}$$

$$\boldsymbol{HRT5 = (tim4 \vee HRT5) \wedge \overline{(tim5)}}$$

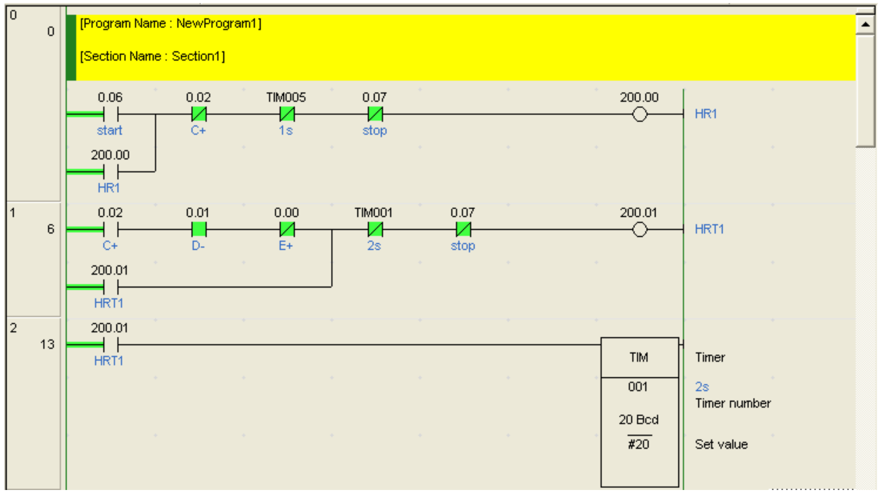
$$\boldsymbol{tim5 = HRT5(1s)}$$

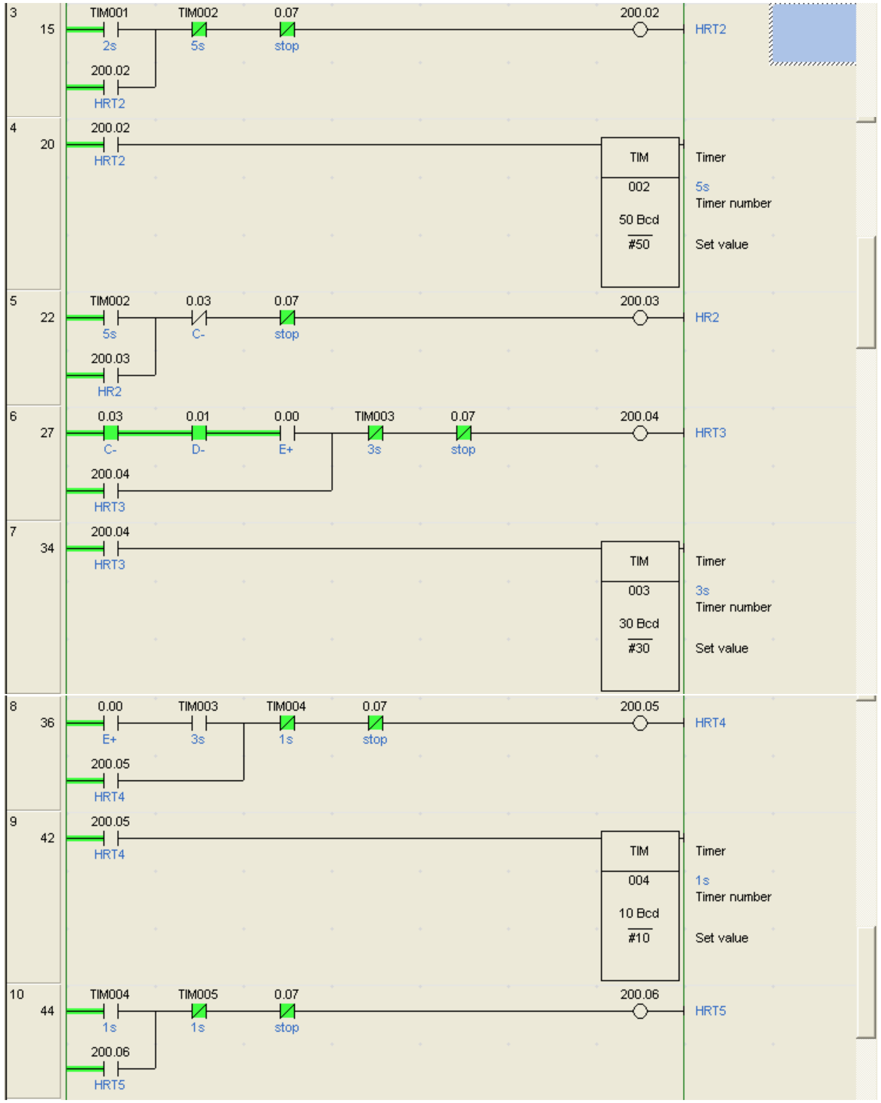
$$\boldsymbol{Y(C) = (HR1 \vee Y(C)) \wedge \overline{(D- \wedge HR2)}}$$

$$\boldsymbol{Y(D) = (HRT1 \vee (C- \wedge E+) \vee Y(D)) \wedge \overline{(tim2 \vee tim4)}}$$

$$\boldsymbol{Y(E) = \left(tim1 \wedge Y(E)\right) \wedge \overline{(tim3)}}$$

7.2.3 Ladder Diagram





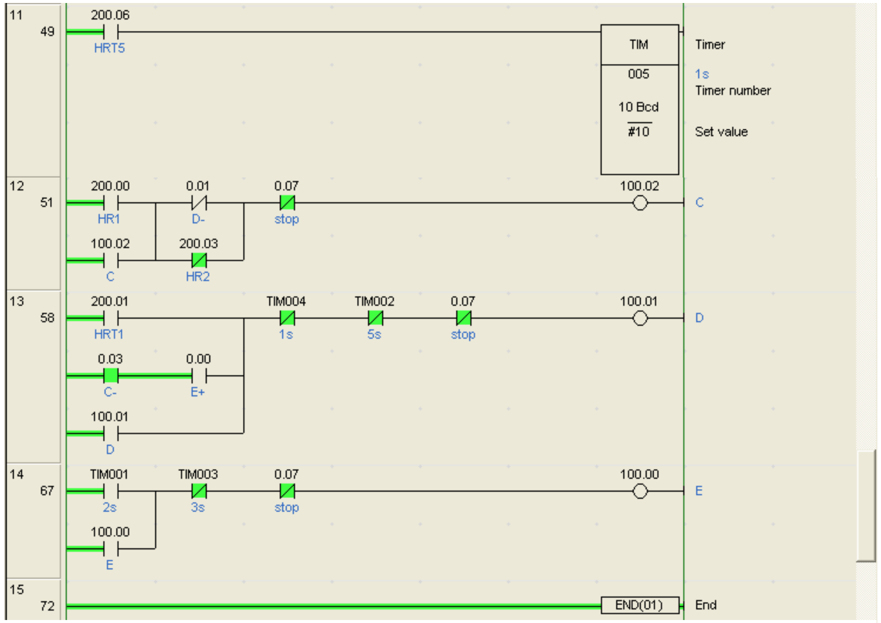


Figure 7. 7 Ladder Diagram

4.2.3 SFC Flow Chart

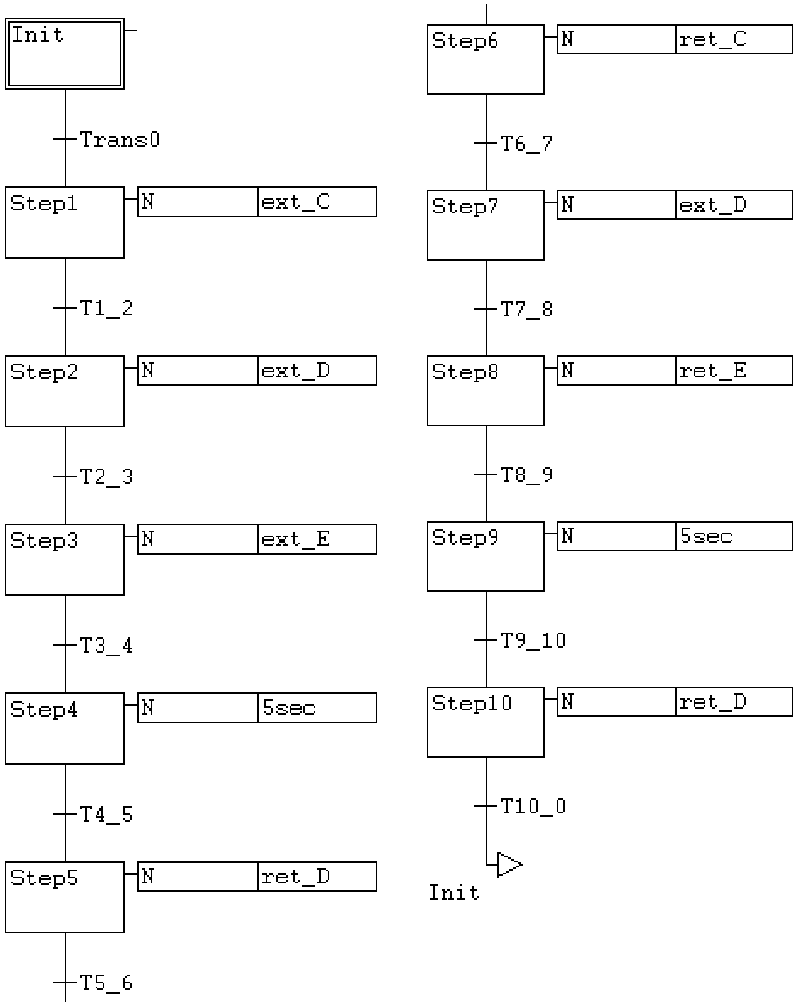
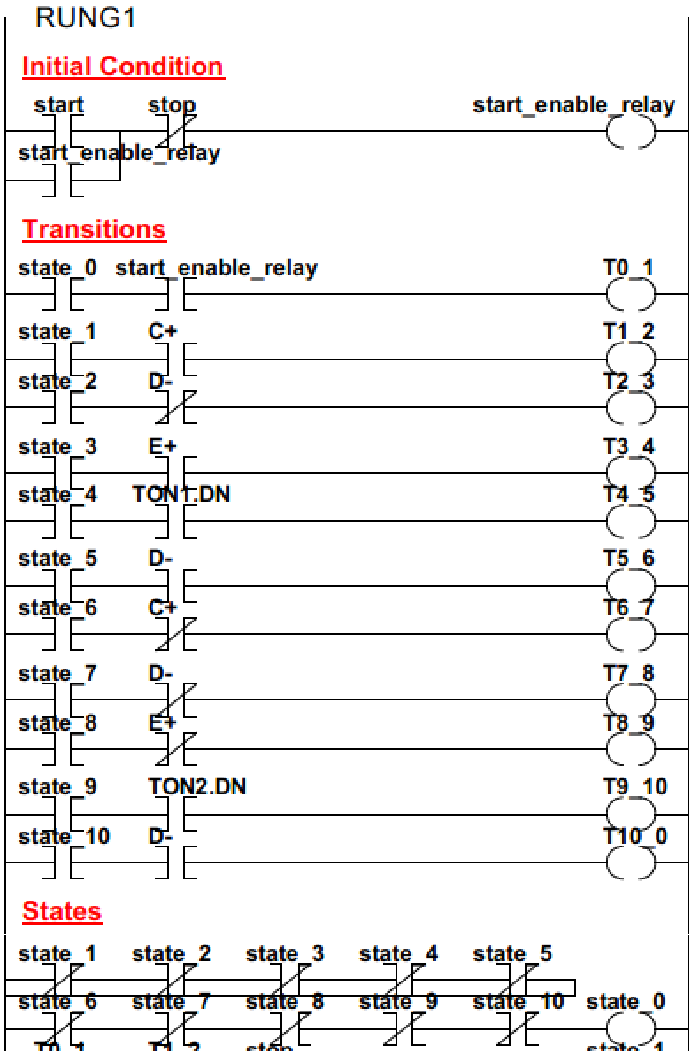
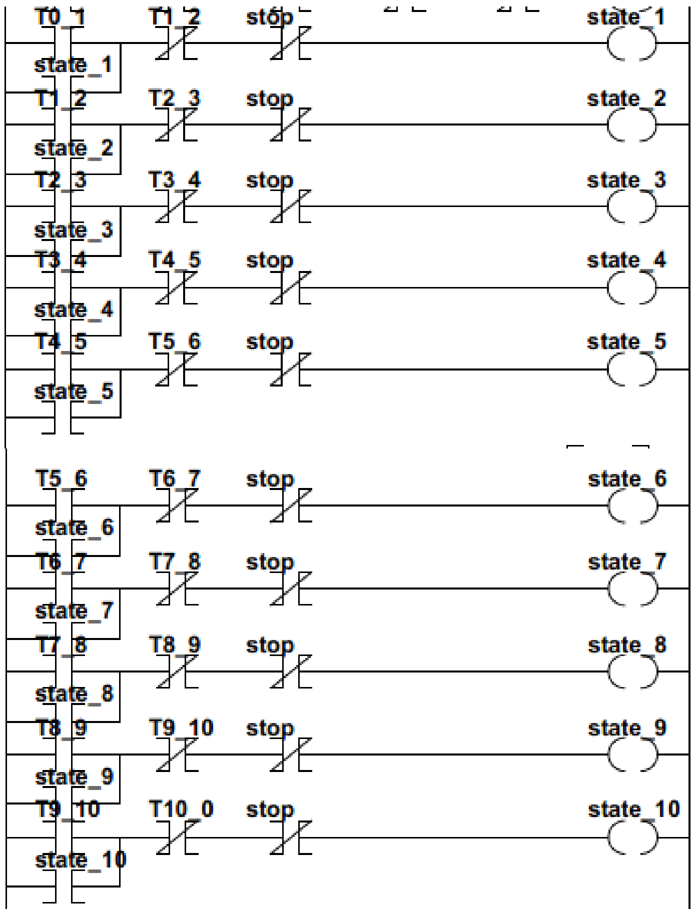


Figure 7. 8 Sequential Function Chart for Act. CDE

4.2.4 SFC Equivalent LAD (method 1)





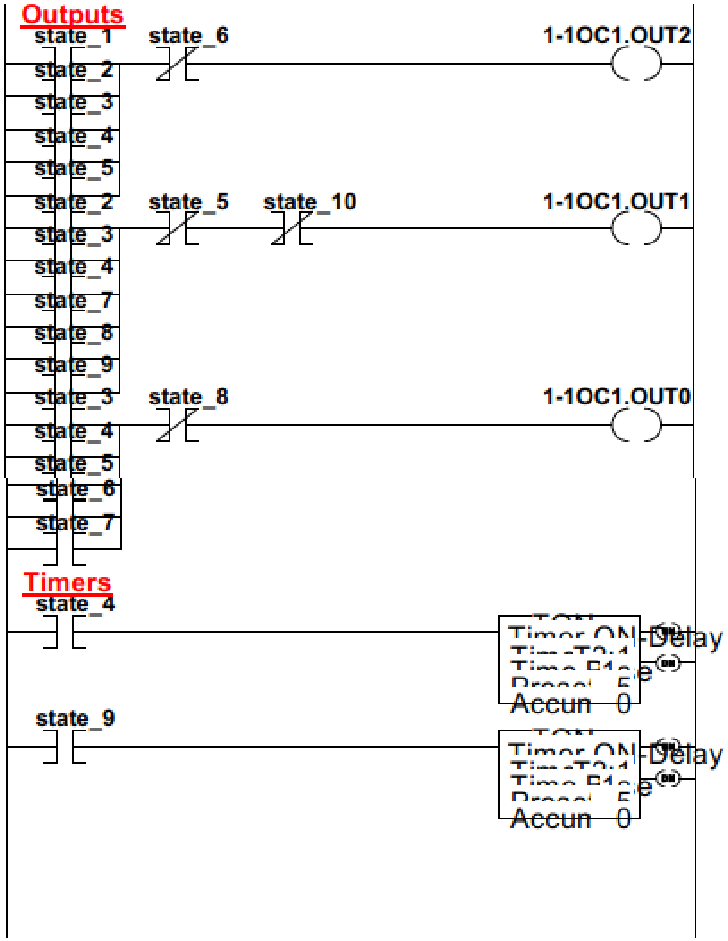


Figure 7. 9 SFC Equivalent LAD

7.3 Appendix III: Testing of Selective Function

Case study: actuator C, D and E are to extend and retract in a sequence of

D+E+ | D- C+ | D+E- | D-C-. Pressing the start push button (PB Start) causes the cycle to execute and pressing the stop push button (PB Stop) causes the operation or cycle to stop. To demonstrate the selective configuration, an alarm indicator is used which will sound when the actuator E extended but not fully extended.

Second variables	SET	RESET
HR1	PB	E+ or Stop
HR2	~~D-~~ and E+	C+ or Stop
HR3	C+ and E+	~~D-~~ and ~~E+~~ or Stop
HR4	~~E+~~ and ~~D-~~	C- or Stop

Actuators	SET	RESET
Y(C)	D- and HR2	D- and HR4 or Stop
Y(D)	PB or HR3	HR2 or HR4 or Stop
Y(E)	~~D-~~ and HR1	~~D-~~ and HR3 or Stop
Alarm	C- and E+ and ~~D-~~	Stop

Table 7. 3 Table of Secondary Variables and Outputs

7.3.1 Movement Diagram

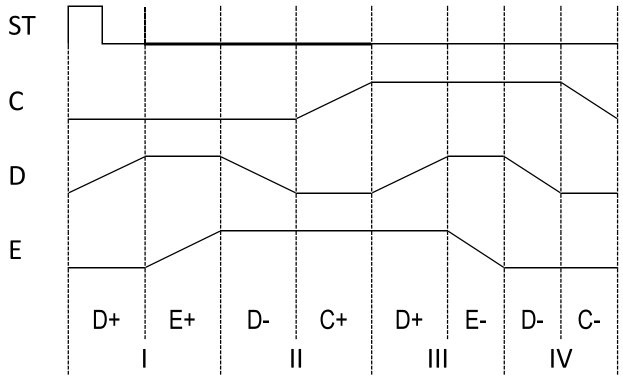
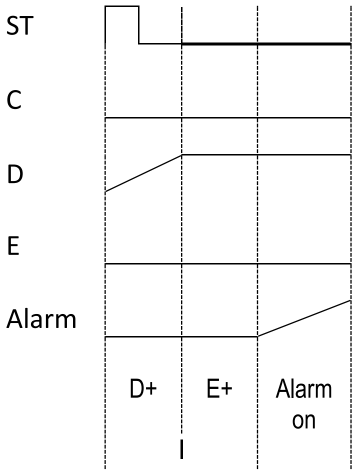


Figure 7. 10 Movement Diagram



7.3.2 Conventional Ladder Diagram

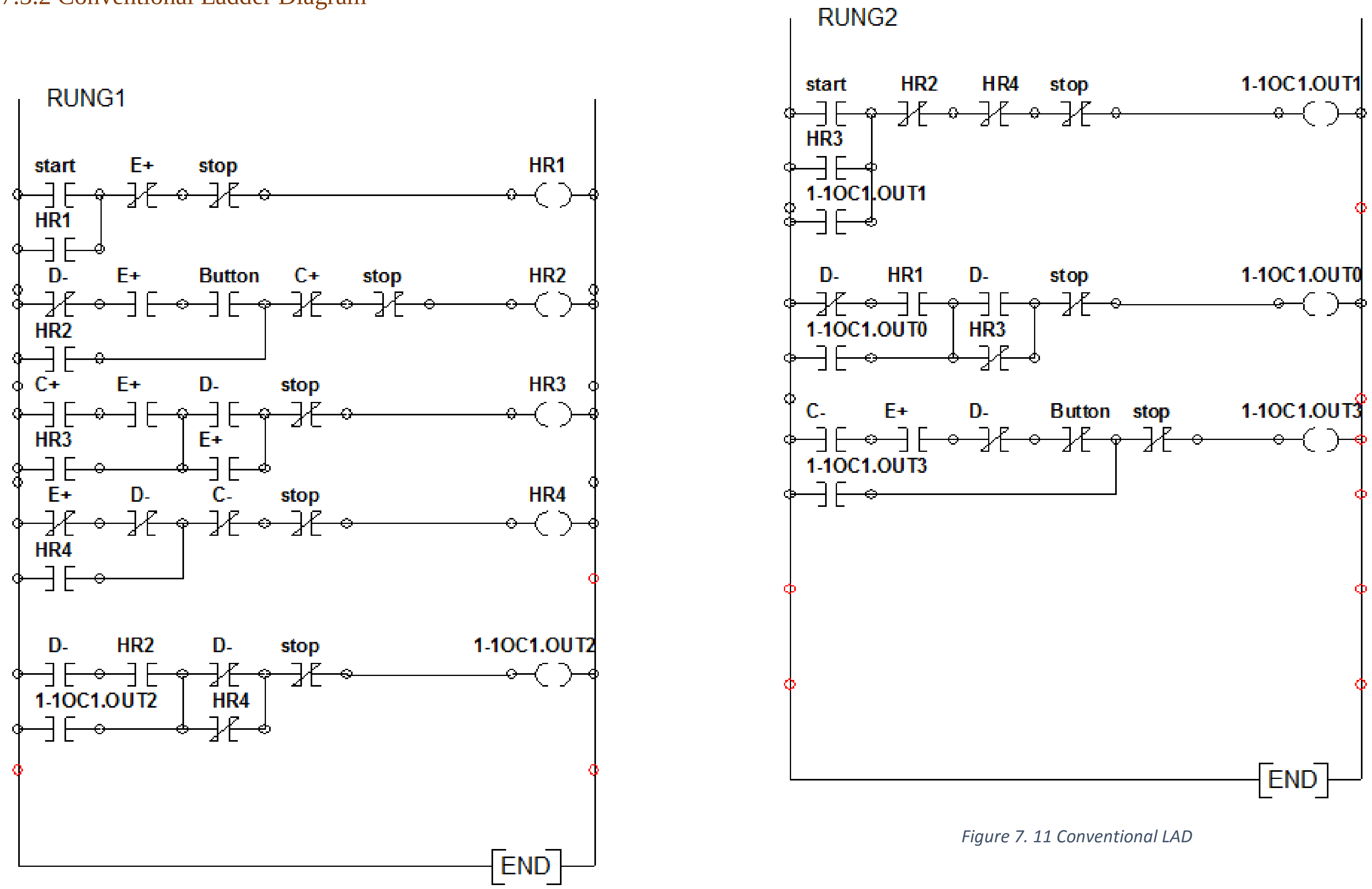


Figure 7. 11 Conventional LAD

7.3.3 Sequential Function Chart

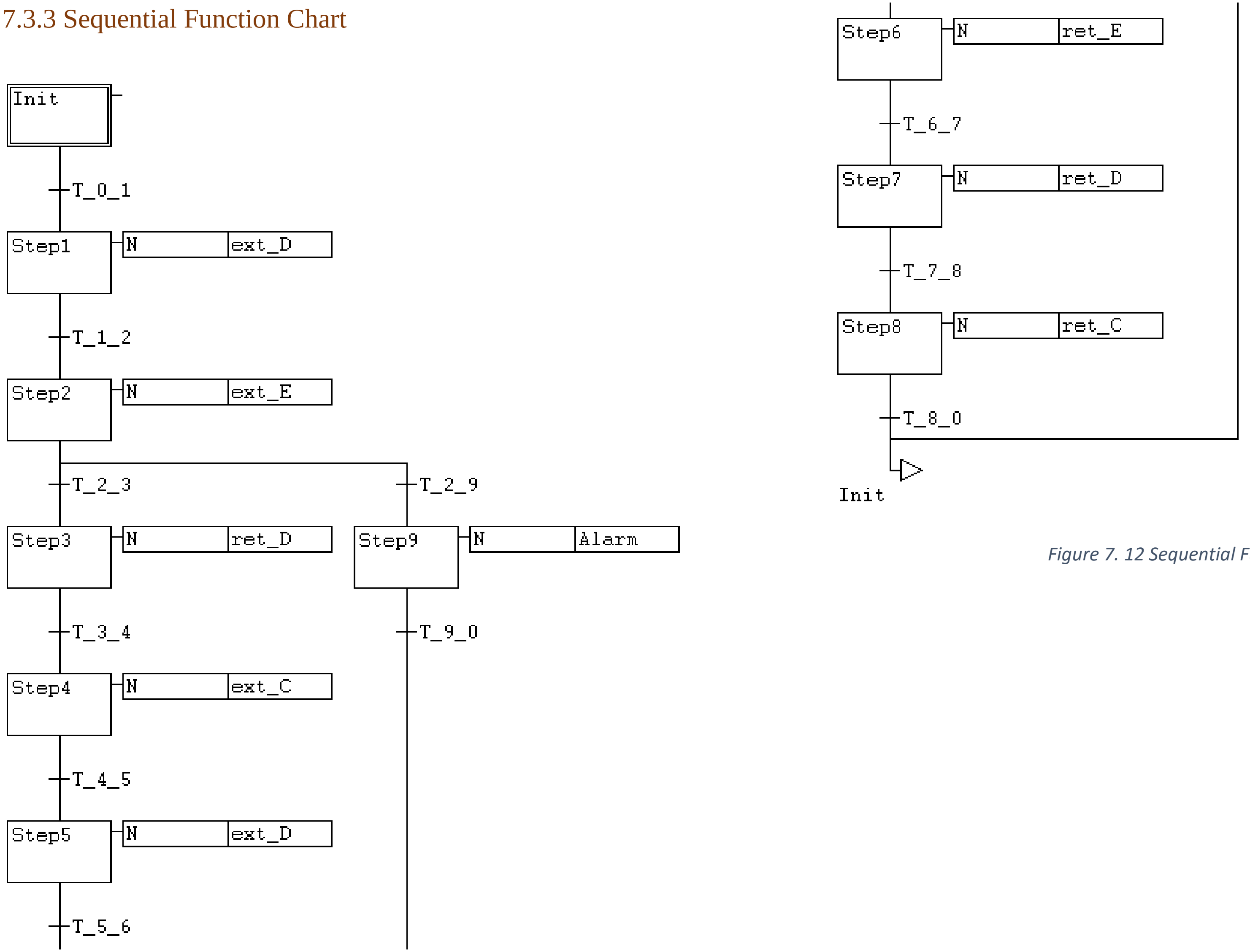
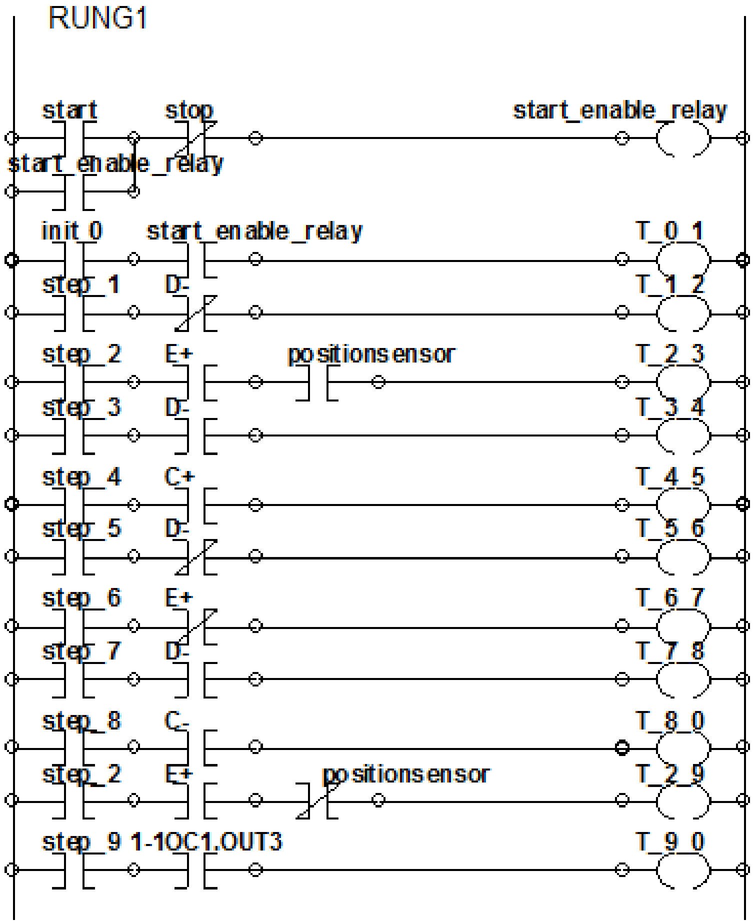
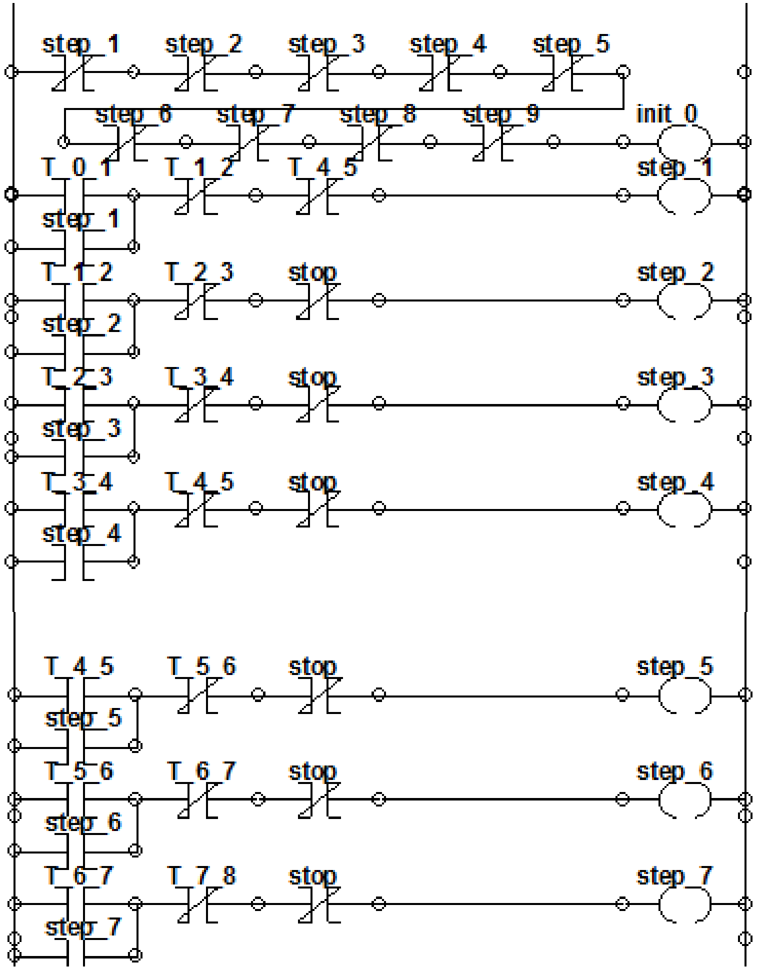


Figure 7. 12 Sequential Function Chart

7.3.4 SFC Equivalent LAD (Method 1)





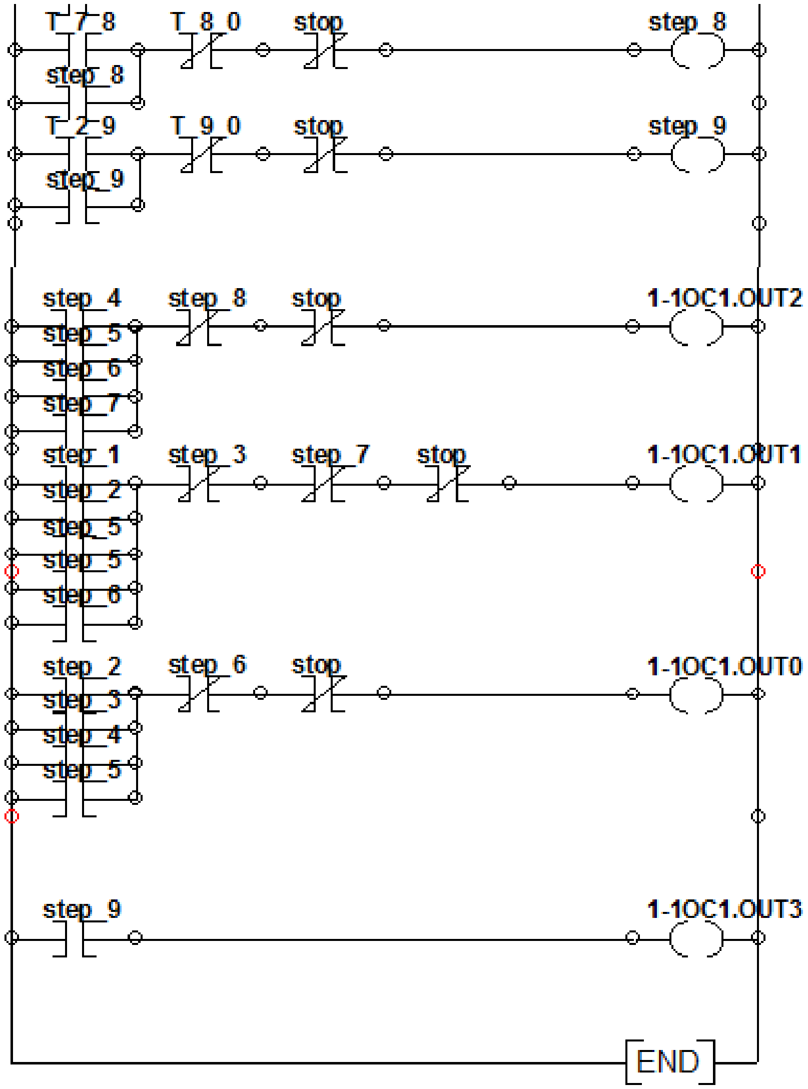


Figure 7. 13 SFC Equivalent LAD