

**ASE-9516**

**Special Assignment in Factory Automation**

**LAD-GEN (Ladder Generator)**

****

BALAJI GOPALAKRISHNAN

Student ID: 245013

Email ID: [balaji.gopalakrishnan@student.tut.fi](mailto:balaji.gopalakrishnan@student.tut.fi)

Contents

[1 Objective 3](#_Toc488601818)

[2 Introduction 3](#_Toc488601819)

[3 Flowchart Methodology 4](#_Toc488601820)

[3.1 Current Methodology 4](#_Toc488601821)

[3.2 Drawback 7](#_Toc488601822)

[3.3 Solution 8](#_Toc488601823)

[4 Methodology 8](#_Toc488601824)

[4.1 Model 8](#_Toc488601825)

[4.2 Technique 9](#_Toc488601826)

[4.3 Tools 9](#_Toc488601827)

[5 Implementation and Validation 11](#_Toc488601828)

[6 Conclusion and Future works 16](#_Toc488601829)

[7 REFERENCES: 17](#_Toc488601830)

**LIST OF FIGURES**

[Figure 1 Lad-gen overall architecture 3](#_Toc488601802)

[Figure 2 Example flowchart 4](#_Toc488601803)

[Figure 3 Transition Logic (Flowchart methodology) 5](#_Toc488601804)

[Figure 4 Function Logic (Flowchart methodology) 6](#_Toc488601805)

[Figure 5 Output Logic (Flowchart methodology) 6](#_Toc488601806)

[Figure 6 Decision flowchart 7](#_Toc488601807)

[Figure 7 Decision flowchart with transitions and functions 7](#_Toc488601808)

[Figure 8 Output logic (Just Operation) 8](#_Toc488601809)

[Figure 9 Output Logic (Both operation and decision) 8](#_Toc488601810)

[Figure 10 Output logic with set reset coils 8](#_Toc488601811)

[Figure 11 FESTO Distribution station 11](#_Toc488601812)

[Figure 12 Flowchart for distribution logic 12](#_Toc488601813)

[Figure 13 Lad-gen transition logic 13](#_Toc488601814)

[Figure 14 Lad-gen function logic 14](#_Toc488601815)

[Figure 15 Lad-gen output logic 15](#_Toc488601816)

[Figure 16 Ladder diagram in TIA Portal 16](#_Toc488601817)

# Objective

The main objective of the assignment was to develop a web based application for generating ladder diagrams from flowchart. The logic in flowchart is to be converted in to ladder diagram using the flowchart methodology. Then the generated ladder diagram is to be tested and validated using the FESTO MPS line.

# Introduction

Machines have become an integral part of our world from the 19th century. Initially, they were just controlled mechanically, later the relays were introduces to control them electrically. One step going further, relays were grouped and it was controlled electronically by the means of PLC (Programmable Logic Controller). That is the pace were the rapid development of machinery in the industry came to place. While designing the PLC, it was planned to be programmed easily. While coming across the previously defined logic ways like relay, wirings and switches in the ladder form, Ladder logic fashion is formed[1].

From the day it has been introduced, ladders have been one of the easiest and most commonly used languages for PLC programming. The only toughest part will be the conversion of a logic form the mind/paper to a ladder logic format. There have been many methods used for the transformation like, state chart methodology[2] and flowchart methodology[3]. State chart methodology is where a logic defined with the machine states is converted into ladder logic and flowchart methodology is the method where a logic defined in the form of flowchart is converted to a ladder logic.

The complete assignment is based on the flowchart methodology, where an application has been built on it to make the conversion of flowchart to ladder logic automatically. The application is web based, where the user interacts with a drag drop type interface to build the flowchart. Once the flowchart is built, with the click of a button it is translated to a ladder logic. The user can then use the generated ladder logic diagram for programming or validation.

This is a full stack NodeJS application, having frontend (express.js), backend (NodeJS) and a database (mongoDB). The frontend provides the interface for communication with the user, while the backend has the logics to convert flowchart, manage users and programs; while the database is used to store the users and their respective programs. The complete architecture of the application can be found in the below Figure 1.

A screenshot of a cell phone

Description generated with high confidence

Figure Lad-gen overall architecture

# Flowchart Methodology

Flowchart methodology is the method used to convert flowchart into ladder logic. it goes step by step. Initially the flowchart is constructed for the required process, then the functions and transitions are mapped. Later, the transition ladder is formed, then the function ladder is formed and later the output ladder is formed. The methodology has been explained with an example below.

## Current Methodology

Below (Figure 2) is a water tank example to demonstrate flowchart methodology. The logic is very simple,

* Initially open the outlet valve and close the inlet valve.
* Once the start button is pressed, open the inlet valve and close the outlet valve.
* If the tank is full or if the stop button is pushed, open the outlet valve and close the inlet valve.
* Repeat the process.

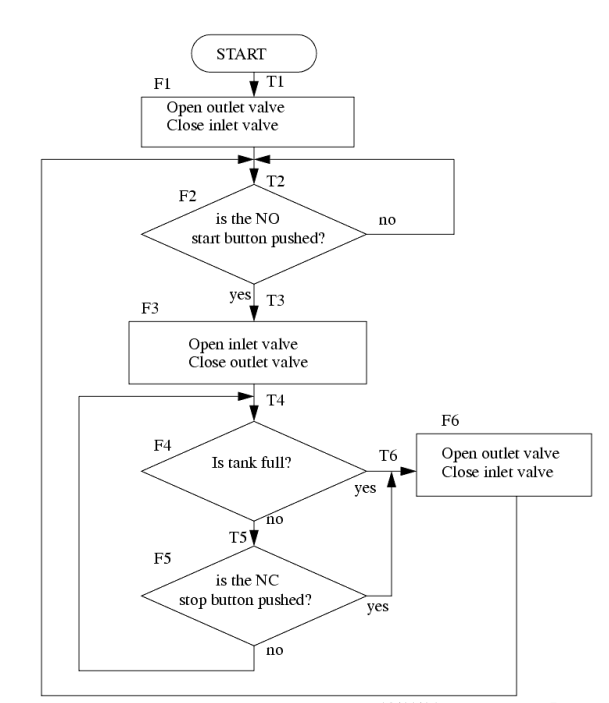


Figure Example flowchart

Once the flowchart has been formed, transition and functions are mapped. First map the functions, each model is mapped as functions. Later, the transitions are mapped, which is the flow from one model to the another. In transitions, if the flow is coming back to the same mode or if the flow is from the last model to the start modes, the transitions are ignored.

Once the transitions and the functions are mapped, the transition ladder logic is formed. It is derived in such a way that; the left side of the rung constitutes to the function blocks and switches which activate the transitions in form of inputs. The right side of the rung is the transition which is the outputs in form of normal coil as shown in Figure 3.

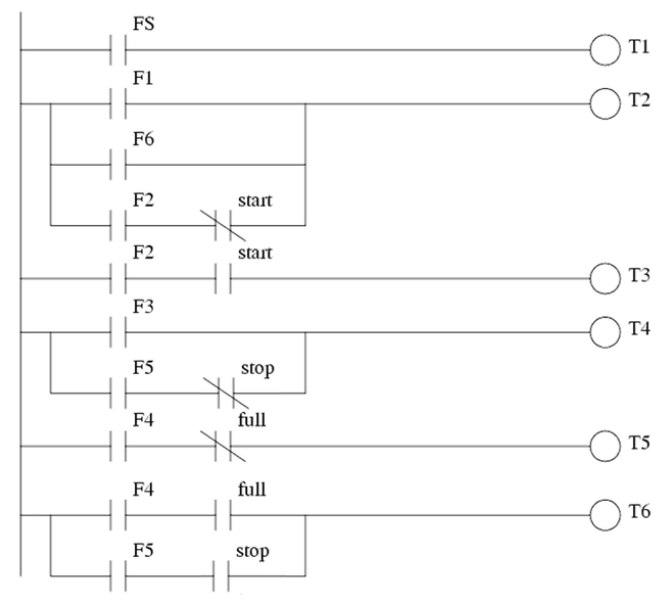


Figure Transition Logic (Flowchart methodology)

Once the transition ladder logic is mapped, the function ladder logic is formed. It is derived in such a way that; the left side of the rung constitutes to the transition blocks which activate the functions along with the functions itself in form of inputs. The right side of the rung is the functions which is the outputs in form of normal coil as shown in Figure 4.

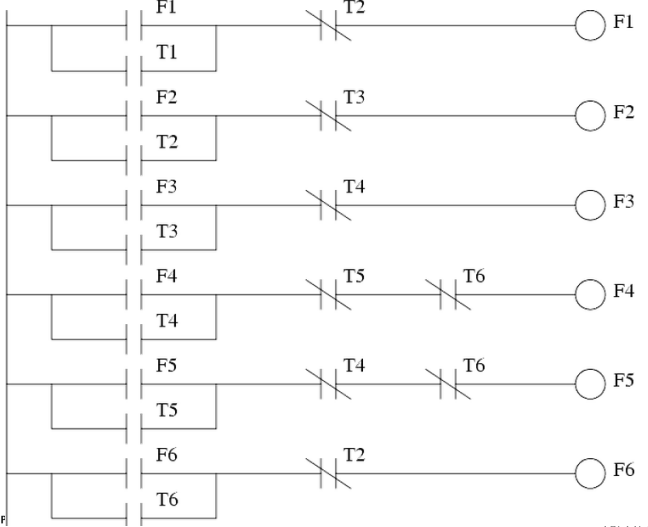


Figure Function Logic (Flowchart methodology)

Once the transition and function ladder logic is mapped, the output ladder logic is formed. It is derived in such a way that; the left side of the rung constitutes to the functions in which the output is on, those functions form the inputs. The right side of the rung is the output in form of normal coil as shown in Figure 5.

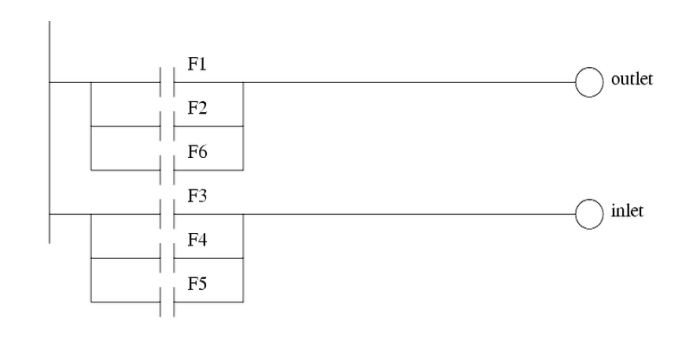


Figure Output Logic (Flowchart methodology)

## Drawback

While the application was being constructed, there was an important drawback with the methodology was found. It relates to the way the output logic can be formed. Basically, the issue was with mapping the functions to the output ladder logic. the example can be seen step by step in the below Figure 6, Figure 7, Figure 8 & Figure 9.

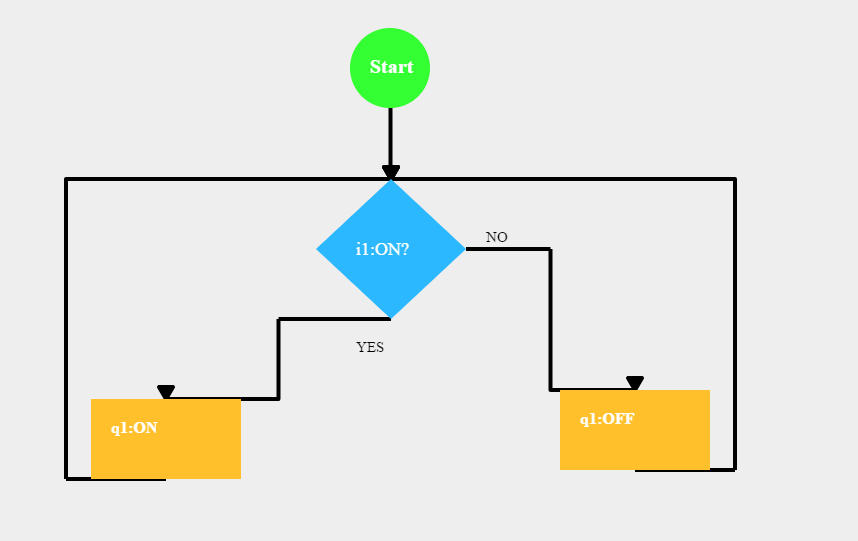


Figure Decision flowchart

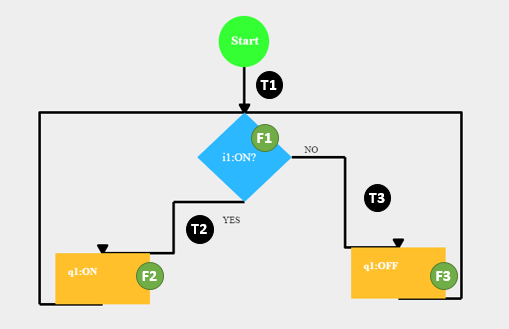


Figure Decision flowchart with transitions and functions

From the flowchart, it can be seen that the output logic can be formed in two ways. Initially, if it is just considering the operation model (Figure 8), then it can be seen that the output will not be activated, while the decision model of flow chart is being executed. In the second state (Figure 9), if it is considering both operation and decision model, then the output will be activated even if it should be off as per the previous stance. Hence it can be seen that both the ways, it is not correct.

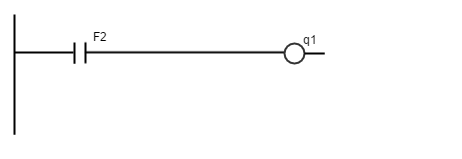


Figure Output logic (Just Operation)

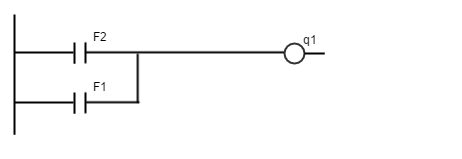


Figure Output Logic (Both operation and decision)

## Solution

In order to solve the drawback, there is a new solution has been derived as part of this assignment. That is to change the outputs from normal coil to set reset coil. This will solve the issue of output memory, whether if it is off state or on state.

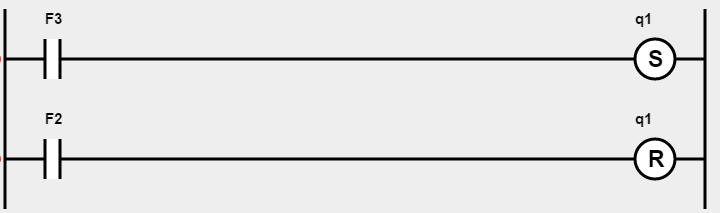


Figure Output logic with set reset coils

# Methodology

The methodology with which the application has been formed is detailed below.

## Model

The complete application was based on the NodeJS full stack, which has a server, frontend and a database. The complete model of the application is shown in the Figure 11.

A picture containing screenshot

Description generated with very high confidence

Figure Lad-gen architecture

The individual components in the architecture are explained below.

### Frontend

Frontend is the part were the user interacts with the application. This is the place where the user drags and drops objects to create the Flowchart and also views the ladder logic diagram once it has been generated. The frontend is programmed with different components to achieve the tasks of generating a flowchart and displaying a ladder diagram. Software components like HTML, Canvas, jQuery, JavaScript and Angular JS are used for individual purposes like,

HTML – Building the webpages

Canvas – a component of HTML used to create drag and drop stuffs for flow chart and display custom diagrams like the ladder rungs with inputs and outputs.

jQuery – Used for dynamically updating components in the webpage form the server and the angular JS ends.

JavaScript and AngularJS - manages frontend’s logic and mathematics. Also helps in communication between frontend and backend.

### Backend

Backend is the part which is used to validate user, save or render flowcharts and convert flowchart to ladder logic. The backend contacts with the frontend via http protocol and json messages. While it contacts the database via SQL. The communication between them are clearly explained in the . The complete backend is developed with NodeJS.

### Database

Database part in the application holds the user details and the saved flowchart details. The database is mongoDB based and uses SQL query and update to contact with the backend.

## Technique

The communications that happen between the models with the user can has been listed as a sequence diagram as shown in Figure 12.

## Tools

The software tools used for building the application are,

* HTML, Canvas, jQuery, JavaScript and AngularJS as part of front end.
* NodeJs as part of back end.

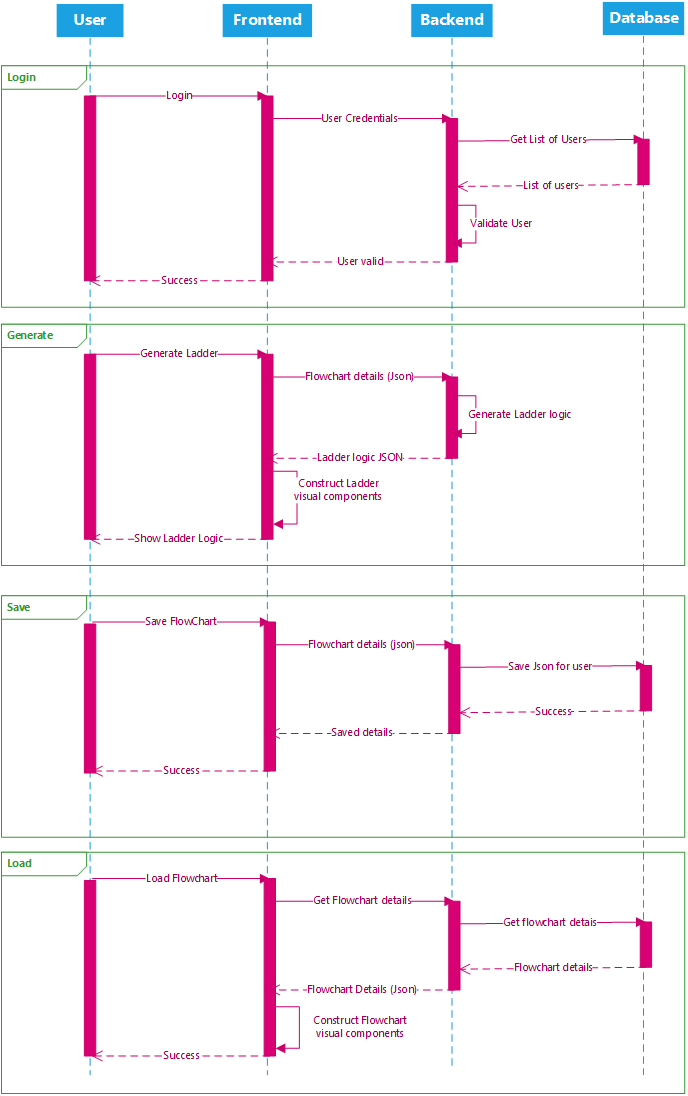


Figure Application Sequence diagram

# Implementation and Validation

The application is implemented and validated using the FESTO MPS Delivery station. The delivery station is shown in the below Figure 13.

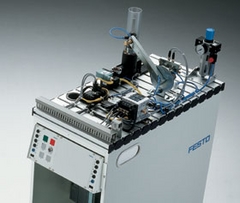


Figure FESTO Distribution station

## Implementation

The ladder diagram in the application can be generated by the following steps as shown in the Figure 14.

Figure Lad-gen Flowchart generation (Implementation)

### Register I/O

First step is to register the inputs and outputs for the process to which the ladder logic is to be generated. The name of the I/O along with its tag are registered as shown in the Figure 15



Figure I/O list Lad-gen

## Creating flowchart objects

The next step is to generate the objects/models required for the flowchart. Mostly the processing and decision boxes.

## Assigning inputs and outputs to objects

The inputs and the outputs registered will appear in the objects (inputs in processing box and outputs in decision box). They are assigned to the respective objects as per the flowchart logic.

## Arrange the object and connect the flowchart

Once all the I/O’s have been assigned, the object are dragged to the position as per their logic and connected via lines.

## Verify and click generate

Once the flowchart has been built, it looks like as shown in the Figure 16. Verify the flowchart and click on generate to generate the ladder logic.

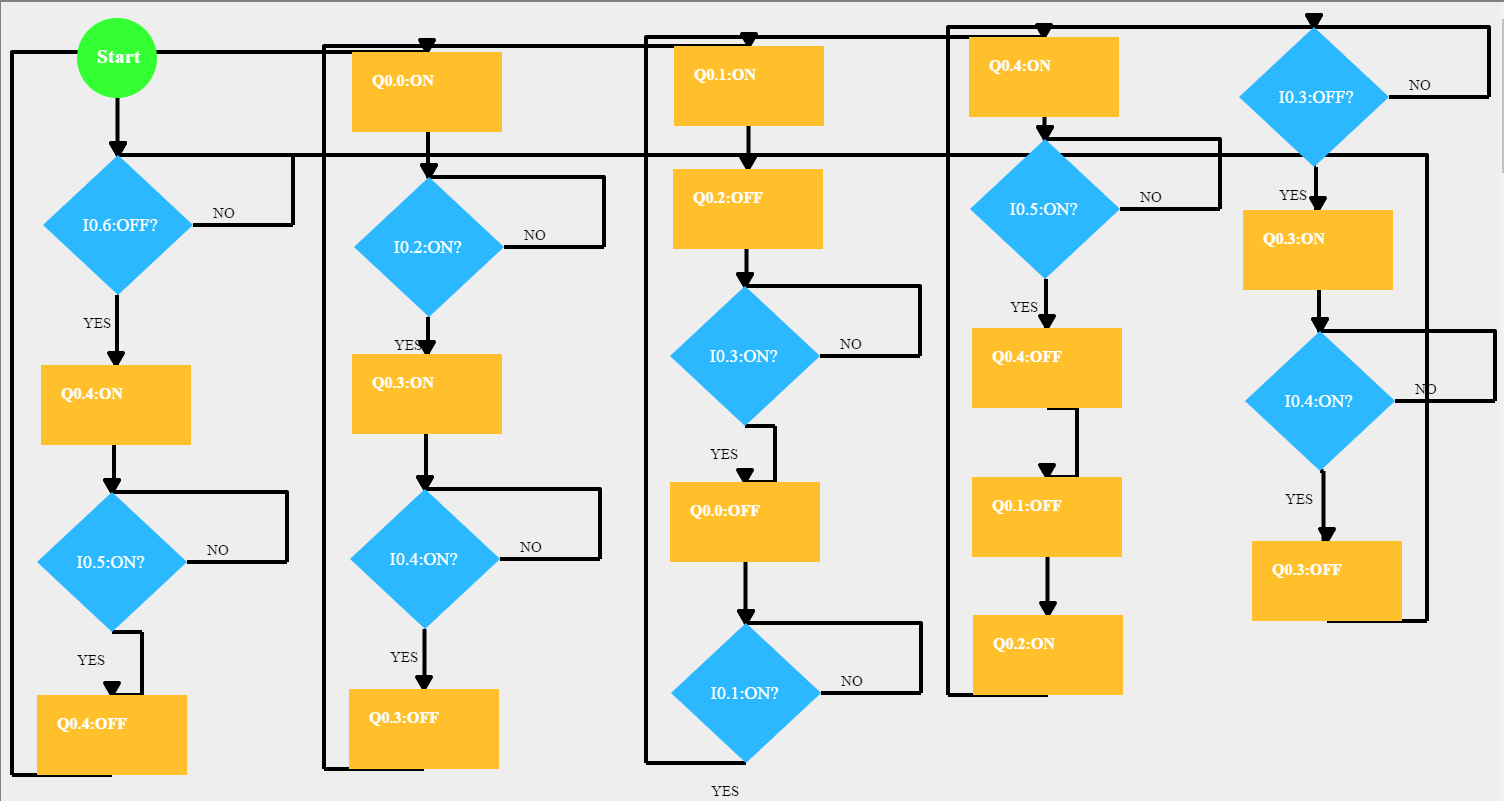


Figure Flowchart for distribution logic

The generated ladder logic will be shown in the application by separating it into transitions, functions and outputs as shown in Figure 17, Figure 18 and Figure 19.

A screenshot of a cell phone

Description generated with high confidence

Figure Lad-gen transition logic

A screenshot of a cell phone

Description generated with high confidence

Figure Lad-gen function logic

A screenshot of a cell phone

Description generated with very high confidence

Figure Lad-gen output logic

## Validation:

The generated logic diagrams were manually copied to TIA portal and then it was tested in FESTO MPS station by deploying it via TIA portal. The codes from TIA portal are available in the GitHub link mentioned in Section 6. A demo was also presented regarding the same.

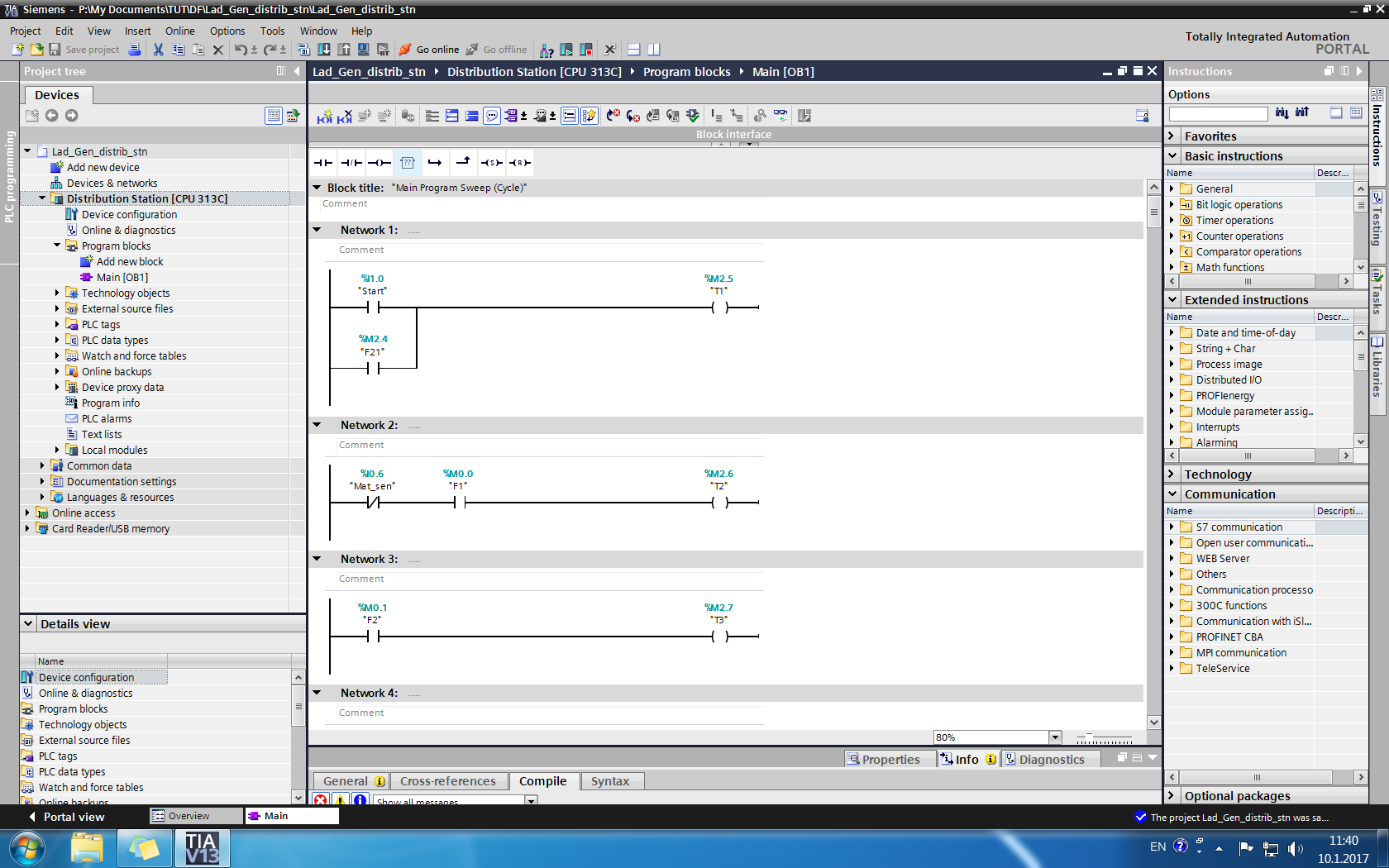


Figure Ladder diagram in TIA Portal

# Conclusion and Future works

The application was developed successfully and validated by testing the ladder logic generated by the application in a FESTO MPS station. The overall objective of the assignment was achieved successfully. This application will be helpful to the students in the following courses Demo Factory (ASE-9506), Discrete Automation Systems (ASE-9416) and Production Automation Exercises (MEI-56206).

Currently the application can be used to just generate the ladder diagrams and show them on the screen. Hence it needs further improvements to export the ladder diagram and import them to PLC programming toolkits like TIA portal. Other than this, there also needs to be some more improvements with respect to timers and counters. Also, the optimization of flowchart must be taken care. These and more will be considered as part of future works.

Links to the Code and online version of the tool can be found in the below links,

Online Version: <http://ladgene-baltor.rhcloud.com/>

Github Link: <https://github.com/Baltor12/LAD_Gene.git>

# REFERENCES:

[1] ‘History of the PLC | AutomationDirect’. [Online]. Available: http://library.automationdirect.com/history-of-the-plc/. [Accessed: 23-Jul-2017].

[2] ‘State Diagrams: A New Visual Language For Programmable Logic Controllers - fulltext.pdf’. [Online]. Available: https://macsphere.mcmaster.ca/bitstream/11375/9383/1/fulltext.pdf. [Accessed: 23-Jul-2017].

[3] ‘plc\_flowchart.fm - plc\_flowchart.pdf’. [Online]. Available: http://www.theautomationstore.com/product\_images/uploaded\_images/plc\_flowchart.pdf. [Accessed: 23-Jul-2017].