BCIT SCHOOL OF ENERGY

Mechatronics and Robotics Engineering Technology

ROBT 4456:
PLC Applications

PROJECT REPORT:
ADVANCED FOUR-FLOOR ELEVATOR

Author:	Wendel Pena
Date:	April 14, 2024

ACKNOWLEDGMENTS

I must express my thanks to the Mechatronics and Robotics faculty for their support and encouragement throughout the project development: Isaiah Regacho, Andrew Friesen, and Edwin Cheung. Completion of the project would have not been accomplished without the guidance from my lab instructor, Isaiah Regacho. I would like to show my appreciation for Isaiah who was available to discuss PLC programming techniques outside of scheduled lab periods.

ABSTRACT

This PLC project demonstrates the application of various PLC programming techniques. This project programs the lab bench mock elevator to behave as a real elevator. To accomplish this, the project utilizes a finite state machine to control the process. Additional unique features were implemented which required advanced PLC programming topics such as Add-On Instructions (AOI) and User Defined Data Types (UDT).

PREFACE

The assignment of this project was provided in 8 distinct parts. Each part asks for additional functionality and must be completed sequentially. All 8 parts are complete and documented.

1) Three Floor Elevator
2) Inside Panel
3) Multiple Calls/Requests
4) Elevator Door
5) Emergency Stop
6) Four Floors
7) Statistical Analysis
8) Fault Detection

CONTENTS

FIGURES

TABLES

DEFINITIONS

Abbreviation	Definition
PLC	Programmable Logic Controller
AOI	Add-On Instruction
UDT	User-Defined Data Type

Table 1: List of Abbreviations

<table>
<tr><th>Symbol</th><th>Definition</th></tr>
<tr><td>V:</td><td>Virtual: For buffering inputs/outputs</td></tr>
<tr><td>!</td><td>Boolean operator: Logical NOT</td></tr>
<tr><td>&</td><td>Boolean operator: Logical AND</td></tr>
<tr><td>|</td><td>Boolean operator: Logical OR</td></tr>
<tr><td>=</td><td>Assignment operator. Sets variable equal to a value</td></tr>
<tr><td>==</td><td>Equality operator. Compares 2 values for equality</td></tr>
</table>

Table 2: List of Symbols

1 INTRODUCTION

This document discusses the design and implementation of an Advanced Four-Floor Elevator PLC controller. As a PLC programming project, the hardware is provided by BCIT Mechatronics and Robotics.

1.1 PROJECT DESCRIPTION

The elevator project is defined in 8 parts. Each part demands additional features on top of the previous parts. This report reflects the most advanced implementation.

1.2 PROJECT HARDWARE

The hardware design and implementation are provided at the start of the project. The main PLC controller is an Allen-Bradley 1769 CompactLogix PLC and was programmed in RSLogix Studio 5000.

Figure 1: Mimic Elevator Setup

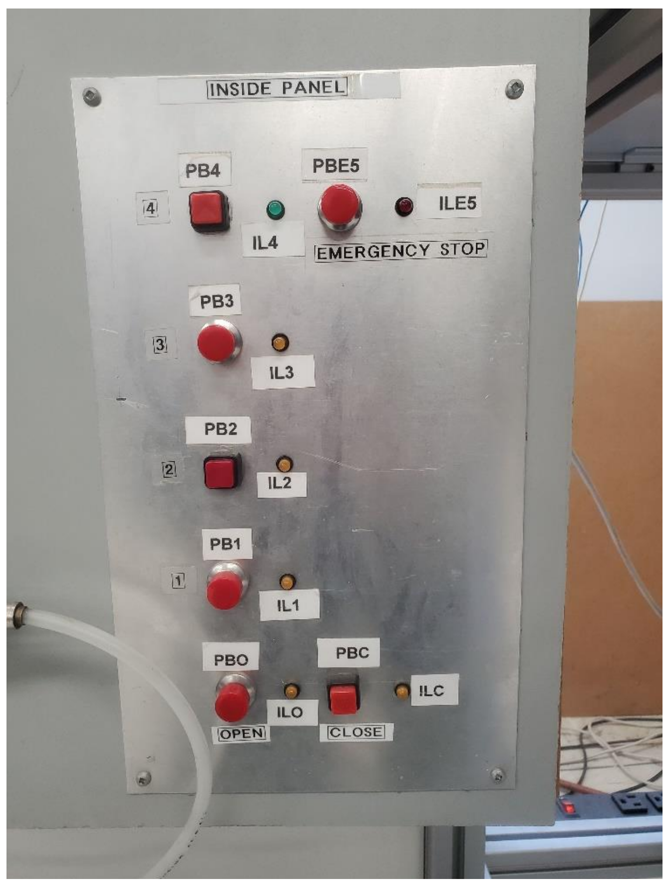

Figure 2: Elevator Inside Panel

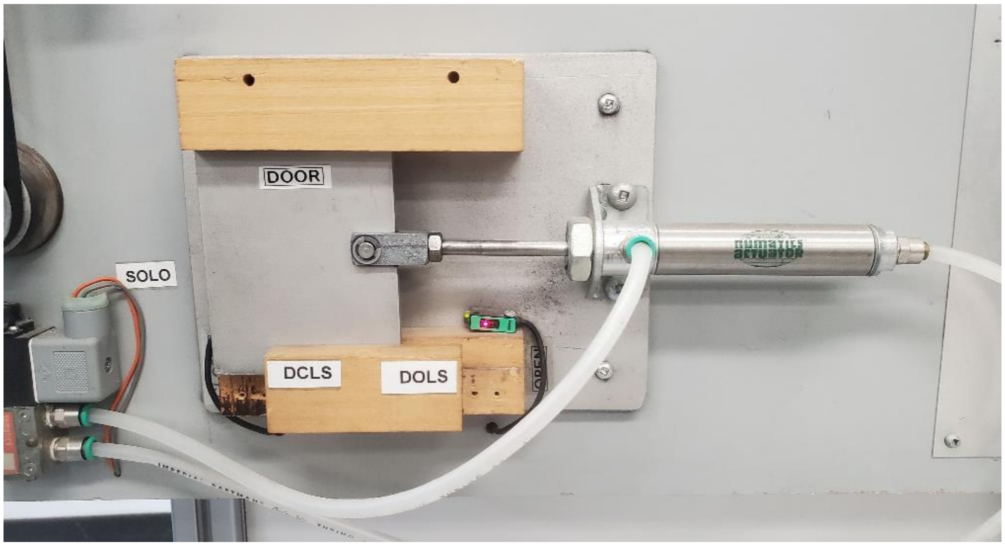

Figure 3: Door Mechanism

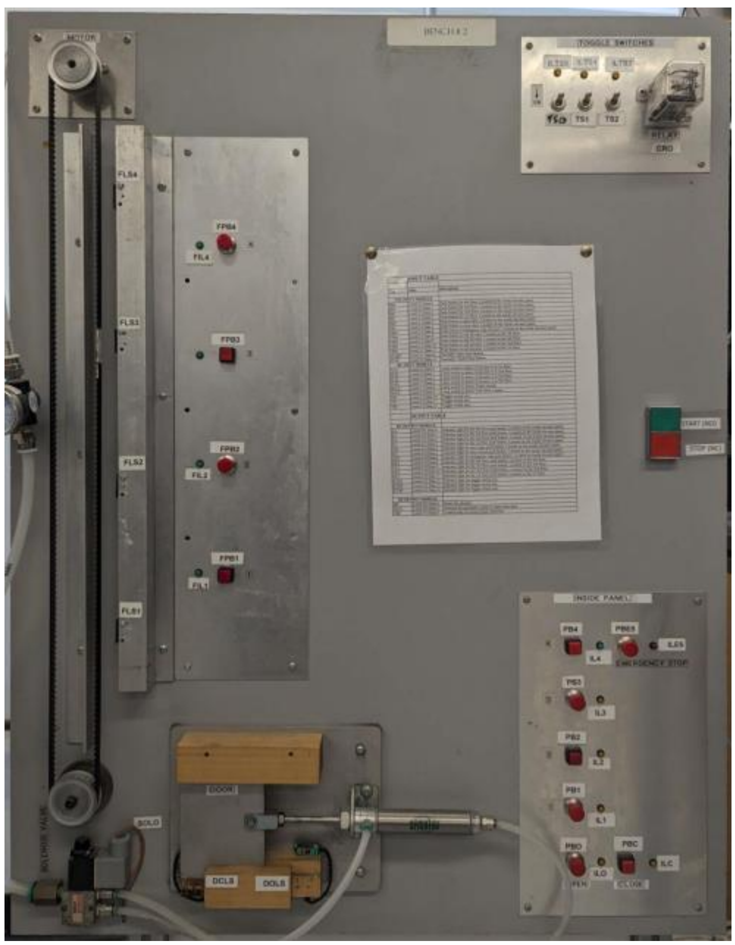

Figure 4: Entire Physical System

2 Project Overview

The function of the project is to mimic a 4-floor elevator that stops and services floors by opening and closing the door. The program manages a variety of faults, data logging, and multi-calls to different floors.

2.1 Studio 5000 Project Organization

The project is organized into four different routines, each having their own specific operations, while still being able to share variables/tags with each other. The image below shows how the different routines interact and share parameters and tags.

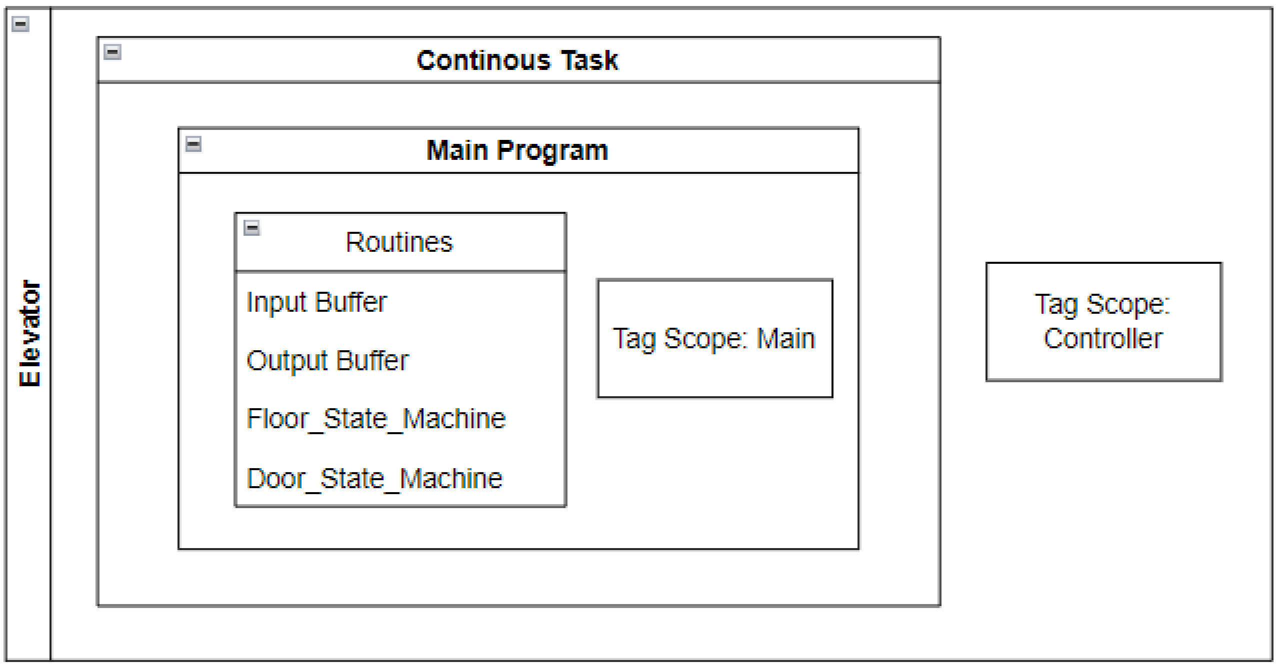

Figure 5: Visualization of Project Organization

2.2 Controller Tags

This table contains the controller tags starting with the input and output modules.

Tag	Type	Task
Local:2:I.Data	DINT	InputBuffer
Local:3:I.Data	DINT	InputBuffer
Local:4:O.Data	DINT	OutputBuffer
Local:5:O.Data	DINT	OutputBuffer

Table 3: Controller Tags

2.3 Ladder Logic Structure

Each program adheres to an Input Buffer, Logic, and Output Buffer structure. The input buffer routine of a program ensures logical inputs are not change during a single scan. The output buffer only updates after the rest of the program has been scanned.

Each subroutine is responsible for a dedicated set of tags which may not be written to in other subroutines.

Figure 6: Main Program Routine

2.4 Add-On Instructions

This project uses 1 add-on instruction to log relevant data on floor calls such as date of service, time to service, etc.

Figure 7: getFloorData Logic

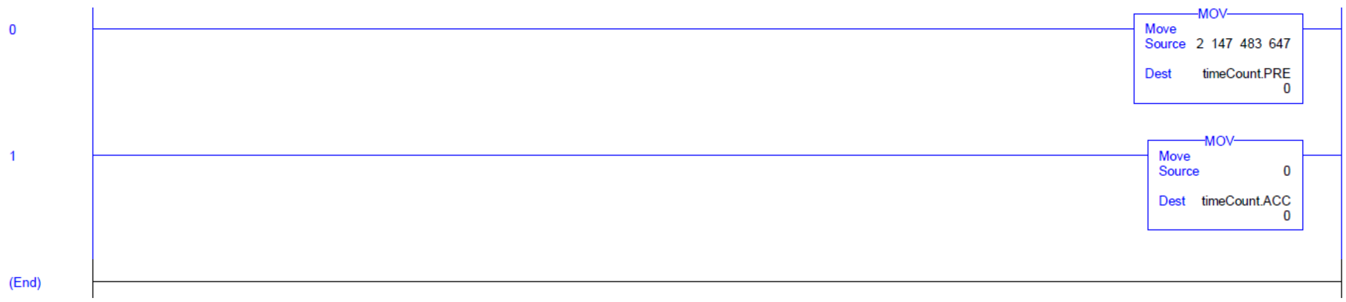

Figure 8: getFloorData Prescan

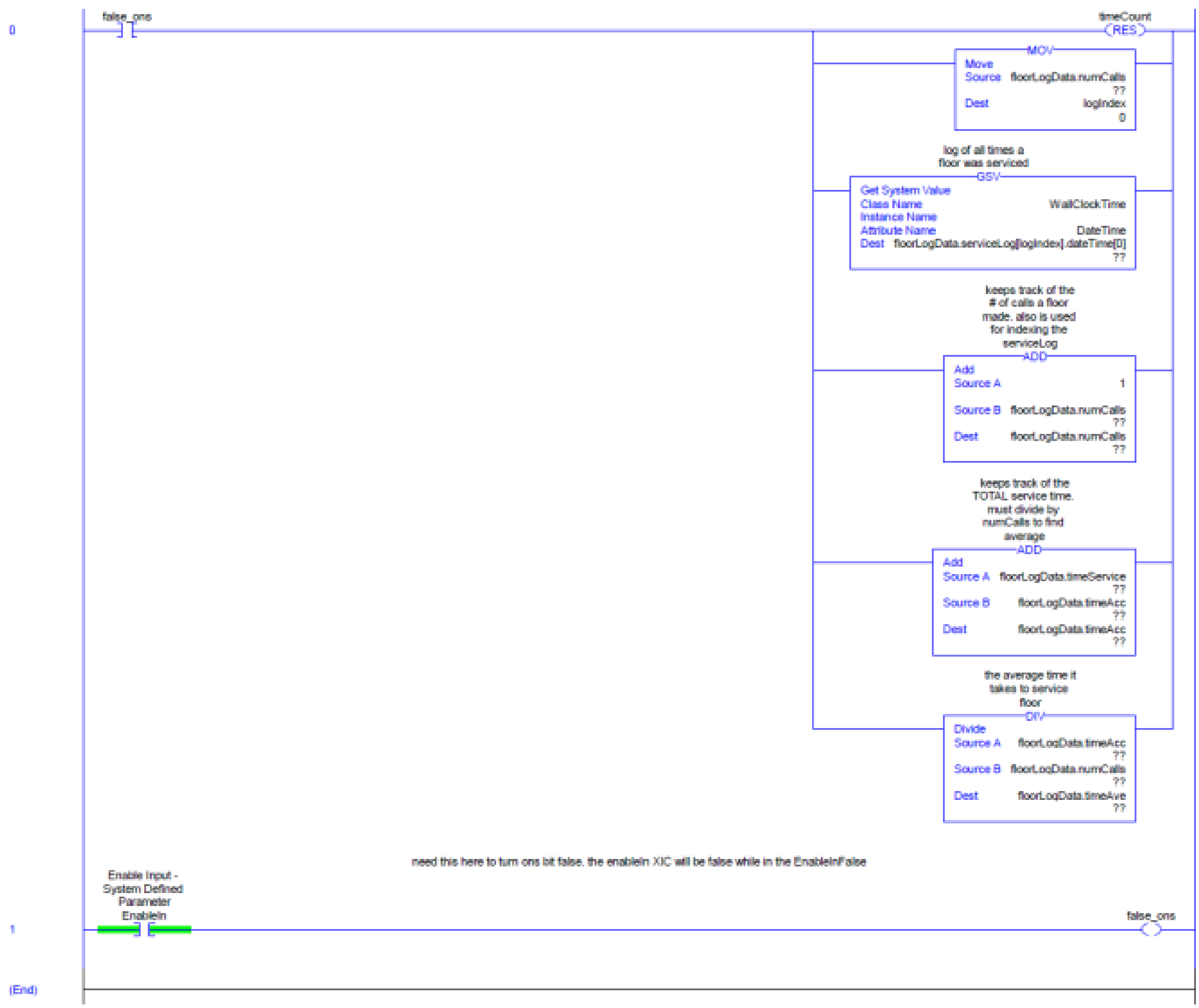

Figure 9: getFloorData EnableInFalse

2.5 User Defined Types

This project uses 3 user-defined types which gather all relevant data for elevator call log data, fault event logging, and datetime.

UDT #1: LOG_DATA[4]

Name	Type	Description
numCalls	DINT	The # of times a floor is serviced
timeAve	DINT	Average time to service a floor
timeAcc	DINT	Tracks TOTAL service time. To be divided by numCalls to find average time.
timeService	DINT	Time it takes to service a floor for one instance
serviceLog	DATA_TIME[50]	Log of all times a floor was serviced

Table 4: LOG_DATA UDT values

UDT #2: DATE_TIME

Name	Type	Description
dataTime	DINT[7]	Array to hold datetime of when floors are serviced. (In format: Year, month, day, hour, minute, second, millisecond)

Table 5: DATE_TIME UDT values

UDT #3: FAULTS[4]

Name	Type	Description
faultCode	DINT	Code for each unique fault
faultType	DINT	Minor (1) or major (2) fault
systemState	DINT	Logs the current elevator floor state when the fault occurred
timeFaulted	DINT[7]	Datetime the fault occurred

Table 6: FAULTS UDT values

3 Main Program

The elevator operation is managed by a state machine. This state machine determines logical movement of the elevator by driving the motor and its direction. The state machine main inputs are received by the request manager and the call manager. Faults also affect the state machine. The image below shows how the different subsystems relate to each other.

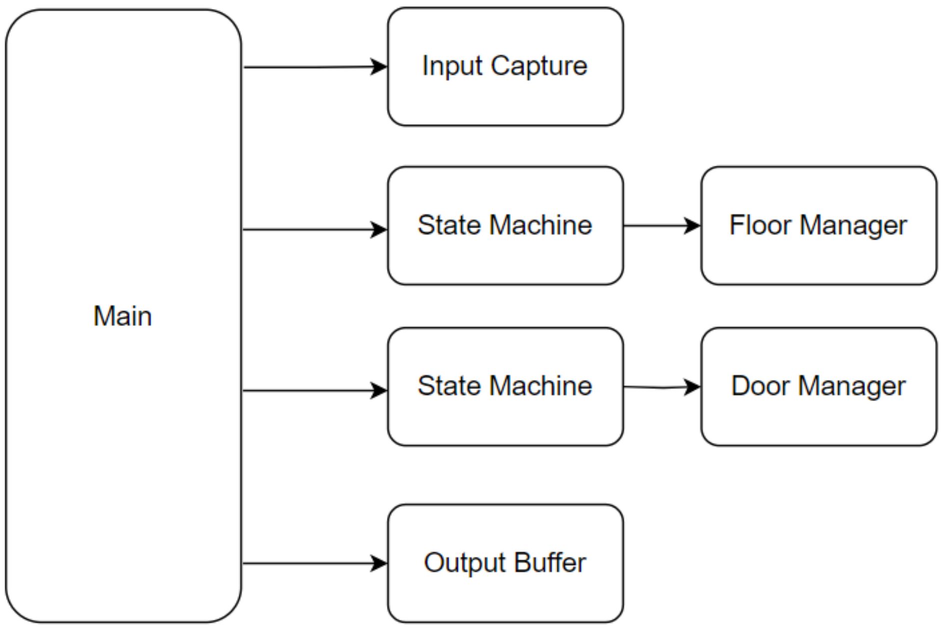

Figure 10: Block diagram of where each subroutine is called

3.1 Program Tags

These tags are external to the Main program. Access to these tags is managed by the input and output buffer subroutines.

Tag	**Mapping**	**Type**	**Subroutine**
V_2I_Data	Local:2:I.Data	DINT	InputBuffer
V_3I_Data	Local:3:I.Data	DINT	InputBuffer
V_4O_Data	Local:4:O.Data	DINT	OutputBuffer
V_5O_Data	Local:5:O.Data	DINT	OutputBuffer

Table 7: Controller Tag Buffering

Tag	Alias	Type	Description
V_PB4	V_2I_Data.0	BOOL	V: Pushbutton 4
V_PB3	V_2I_Data.1	BOOL	V: Pushbutton 3
V_PB2	V_2I_Data.2	BOOL	V: Pushbutton 2
V_PB1	V_2I_Data.3	BOOL	V: Pushbutton 1
V_PBO	V_2I_Data.4	BOOL	V: Pushbutton Open
V_PBC	V_2I_Data.5	BOOL	V: Pushbutton Close
V_PBE5	V_2I_Data.6	BOOL	V: Pushbutton Emergency 5
V_FPB4	V_2I_Data.8	BOOL	V: Floor Pushbutton 4
V_FPB3	V_2I_Data.9	BOOL	V: Floor Pushbutton 3
V_FPB2	V_2I_Data.10	BOOL	V: Floor Pushbutton 2
V_FPB1	V_2I_Data.11	BOOL	V: Floor Pushbutton 1
V_START	V_2I_Data.14	BOOL	V: START (N.O.)
V_STOP	V_2I_Data.15	BOOL	V: STOP (N.C.)

Table 8: Input Module 2 - Input Buffer Tags

Tag	Alias	Type	Description
V_FLS1	V_3I_Data.0	BOOL	V: Floor Limit Switch 1
V_FLS2	V_3I_Data.1	BOOL	V: Floor Limit Switch 2
V_FLS3	V_3I_Data.2	BOOL	V: Floor Limit Switch 3
V_FLS4	V_3I_Data.3	BOOL	V: Floor Limit Switch 4
V_DCLS	V_3I_Data.8	BOOL	V: Door Close Limit Switch
V_DOLS	V_3I_Data.9	BOOL	V: Door Open Limit Switch
V_TS2	V_3I_Data.10	BOOL	V: Toggle Switch 2
V_TS1	V_3I_Data.11	BOOL	V: Toggle Switch 1
V_TS0	V_3I_Data.12	BOOL	V: Toggle Switch 0

Table 9: Input Module 3 - Input Buffer Tags

Tag	Alias	Type	Description
V_IL4	V_4O_Data.0	BOOL	V: Indicator Light 4
V_IL3	V_4O_Data.1	BOOL	V: Indicator Light 3
V_IL2	V_4O_Data.2	BOOL	V: Indicator Light 2
V_IL1	V_4O_Data.3	BOOL	V: Indicator Light 1
V_ILO	V_4O_Data.4	BOOL	V: Indicator Light Open
V_ILC	V_4O_Data.5	BOOL	V: Indicator Light Close
V_ILE5	V_4O_Data.6	BOOL	V: Indicator Light Emergency 5
V_FIL4	V_4O_Data.8	BOOL	V: Floor Indicator Light 4
V_FIL3	V_4O_Data.9	BOOL	V: Floor Indicator Light 3
V_FIL2	V_4O_Data.10	BOOL	V: Floor Indicator Light 2
V_FIL1	V_4O_Data.11	BOOL	V: Floor Indicator

Table 10: Output Module 4 - Output Buffer Tags

Tag	Alias	Type	Description
V_M0	V_5O_Data.0	BOOL	V: Motor 0
V_CR0	V_5O_Data.2	BOOL	V: Control Relay 0

Table 11: Output Module 5 - Output Buffer Tags

3.2 Elevator Floor State Machine

This subroutine is responsible for the main elevator operation. The elevator actuators are only controlled by the logic within this subroutine and is also the only subroutine which may call the door state machine subroutine. The logic within this subroutine implements a finite state machine. Below are the tags associated with the floor state machine.

Tag	Type	Description
Current_State	DINT	Current State of the FSM
Next_State	DINT	Accepted Next State of the FSM
callArray	DINT[6]	Array to hold current calls for floors.
currentFloor	DINT	The floor the elevator is at currently
nextFloor	DINT	The floor that is currently being called to
checkedBothDir	DINT	Variable to determine if elevator looked for called floors both up/down
doorOpenFunc	DINT	Variable to determine when door state machine can run (floor and door state machine common variable)
down_ons	BOOL	One shot bit for indexing down callArray
up_ons	BOOL	One shot bit for indexing up callArray
elevatorDirection	DINT	Variable affecting CR0 based on what direction elevator should move in
floor1	DINT	Constant used for array indexing
floor2	DINT	Constant used for array indexing
floor3	DINT	Constant used for array indexing
floor4	DINT	Constant used for array indexing
floor1_ons	BOOL	One shot bit for reading user button presses
floor2_ons	BOOL	One shot bit for reading user button presses
floor3_ons	BOOL	One shot bit for reading user button presses
floor4_ons	BOOL	One shot bit for reading user button presses
index	DINT	Variable for indexing through callArray

Table 12: Floor_State_Machine Tags

The state machine forces the elevator to behave a specific way. Modifying the transitions can modify the overall behaviour of the system. Below is a state diagram for the floor state machine.

Figure 11: Floor State Machine State Diagram

Pressing a button (FPBx or PBx) sends a signal into an array (callArrray) to keep track of pending calls for floors. Based on this array we can determine what floors need servicing.

To service a floor, State 5 turns on the motor and goes up/down depending on the elvatorDirection variable. Once the elevator hits a floors limit switch (FLSx), the program checks the callArray for a non 0 value. Finding a valid call to a floor stops the elevator and commences the door opening sequence explained in section 3.3.

3.2.1 Multi-Call Process

To handle multiple calls made to the elevator the program uses a "direction bias" process to check for pending calls. During the IDLE state, pressing a button sets the direction of the elevator to ONLY go either up or down. This is the start of the multi-call process. The elevator runs and services any calls to floors ONLY in the biased direction until 1 of 2 things occur, ordered by precedence: 1) There are no more pending calls, or 2) you are at the highest floor (4th floor). Once a direction is done being checked, the elevator checks for pending calls in the OPPOSITE direction that was previously checked using the same method previously mentioned.

This process goes on until all pending calls are serviced. If there are no more pending calls in BOTH directions, we can transition back to the IDLE state. Below is a visualization of the multi-call process.

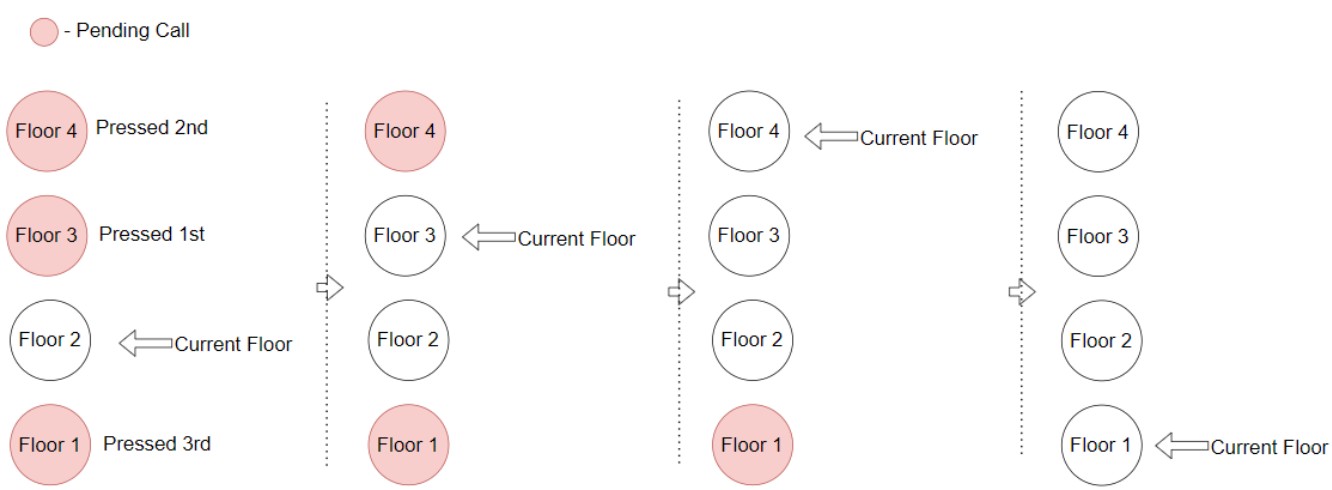

Figure 12: Visualization of Multi-call Routine

3.3 Door Routine

This subroutine manages the door controls and opens and closes while the elevator is servicing floors. A mutual variable “doorOpenFunc” can be modified by both the floor and door state machines. Setting doorOpenFunc to equal 1 triggers the door opening sequence, automatically opening the door for 5 seconds. The user can also close the door after 2 seconds using PBC. Additionally, the user can manually open the door for 3 seconds by pressing PB0. Below is the diagram for the state machine.

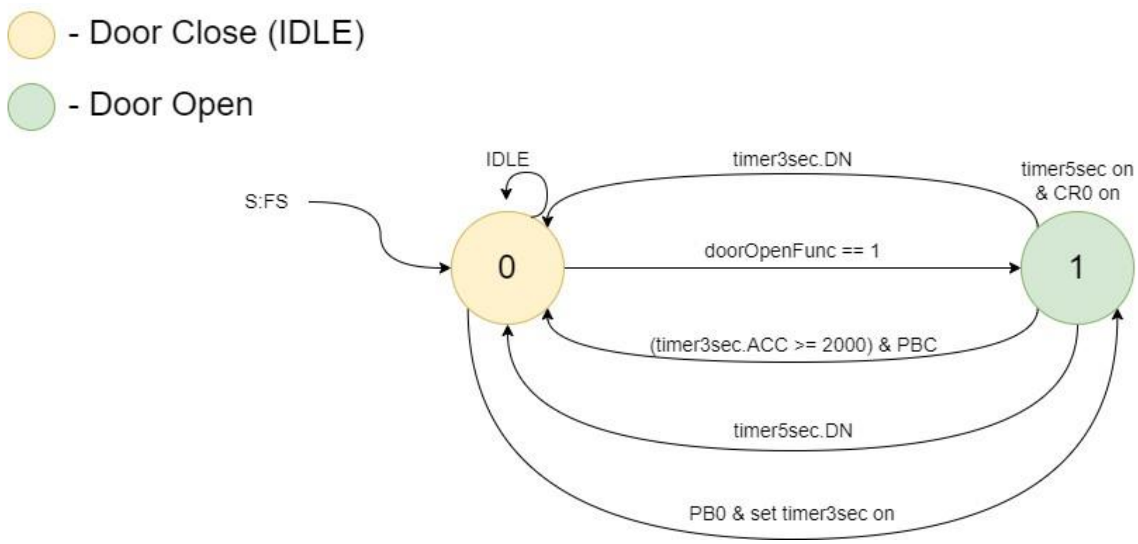

Figure 13: Door State Machine State Diagram

3.4 Emergency Stop

State 100 of the floor state machine sets the elevator into a major fault state, halting all processes in the floor state machine, and limits actions in the door state machine. During an emergency stop, when on a floor, the user can open the door and it will stay open. When in between floors, all door functions are turned off, preventing the door from opening while in an unsafe position. Pressing START resets the system back to State 0. Refer to Figure 11 for reference.

4 Other Features

4.1 Statistical Analysis

Part of the floor state machine. AOI getFloorData handles logging the data. Logs the datetime of when a floor was called (in the format: Year, Month, Day, Hour, Minute, Second, Millisecond), average time to service floor, number of times a floor was serviced, most recent time to service floor, and accumulate total time to service floor,

Using the accumulate time to service a floor and the number of calls, we can determine the average time to service a floor by dividing the two values, timed starting from a button press and ending when the elevator reaches the floor.

4.2 Fault Detection

Part of the floor state machine. The routine detects a total of 3 faults and logs information about each fault into the array masterFaultLog. The log stores data on unique fault codes, time faulted, fault type, and system state when faulted. The faults are as follows:

- Motor run time: faults when the motor runs for longer than 8 seconds.
- Door held open: faults when the door is held open for longer than 10 seconds
- Multiple sensors: faults when more than 1 floor limit switch sensor is activated

Figure 14: Example Fault Detection for Running Motor Fault

5 Conclusion

The elevator project has been a resounding success, characterized by numerous hours of experimentation and problem-solving. While encountering various challenges, particularly during the development of Part 6—Multi-call, each obstacle served as an opportunity for learning and growth.

Throughout my time, I have acquired invaluable skills in the programming of PLCs. Understanding and utilizing AOIs (Add-On Instructions) significantly enhanced the efficiency of my code while maintaining clarity and reducing clutter within the program. Additionally, the implementation of UDTs (User-Defined Types) proved essential in enhancing organization and minimizing confusion during coding processes.

To conclude, this project has not only deepened my understanding of PLCs but has also equipped me with practical skills that I am eager to apply in real-world scenarios. As I transition into the workforce, I am excited to leverage the knowledge and experiences gained from this project to further my professional development and contribute meaningfully to future endeavors.

6 Appendix: Ladder Logic PDF

List of program PDFs and where to find relevant ladder logic:

Main Routine: mainprogram-mainroutine.pdf

Floor State Machine: mainprogram-floor.pdf

Door State Machine: mainprogram-door.pdf

Input Buffer: mainprogram-bufferinput.pdf

Output Buffer: mainprogram-bufferoutput.pdf

Fault Detection: mainprogram-floor.pdf [page 7]

Statistical Analysis: getFloorData.pdf [page 14]