Rafik Hariri University
جامعة رفيق الحريري

College of Engineering-Department of Mechanical and Mechatronics

MECA 542-Industrial and Manufacturing Control

Final Project: Automated Car Parking System

By:

Ahmad Achaji – 20210313

Ahmad Zouhair Hammoudeh –20210482

Bahaa Doucmak - 20210199

Submitted to:

Dr. Ahmad Koubeissi

Table of Contents

List of figures

1. Introduction

The Automated Car Parking System (ACPS) project aims to design and simulate a control system using a Programmable Logic Controller (PLC).

The system's goal is to automate the entire process of vehicle entry, parking allocation, car retrieval, and exit, integrating key features such as safety mechanisms and payment functionalities.

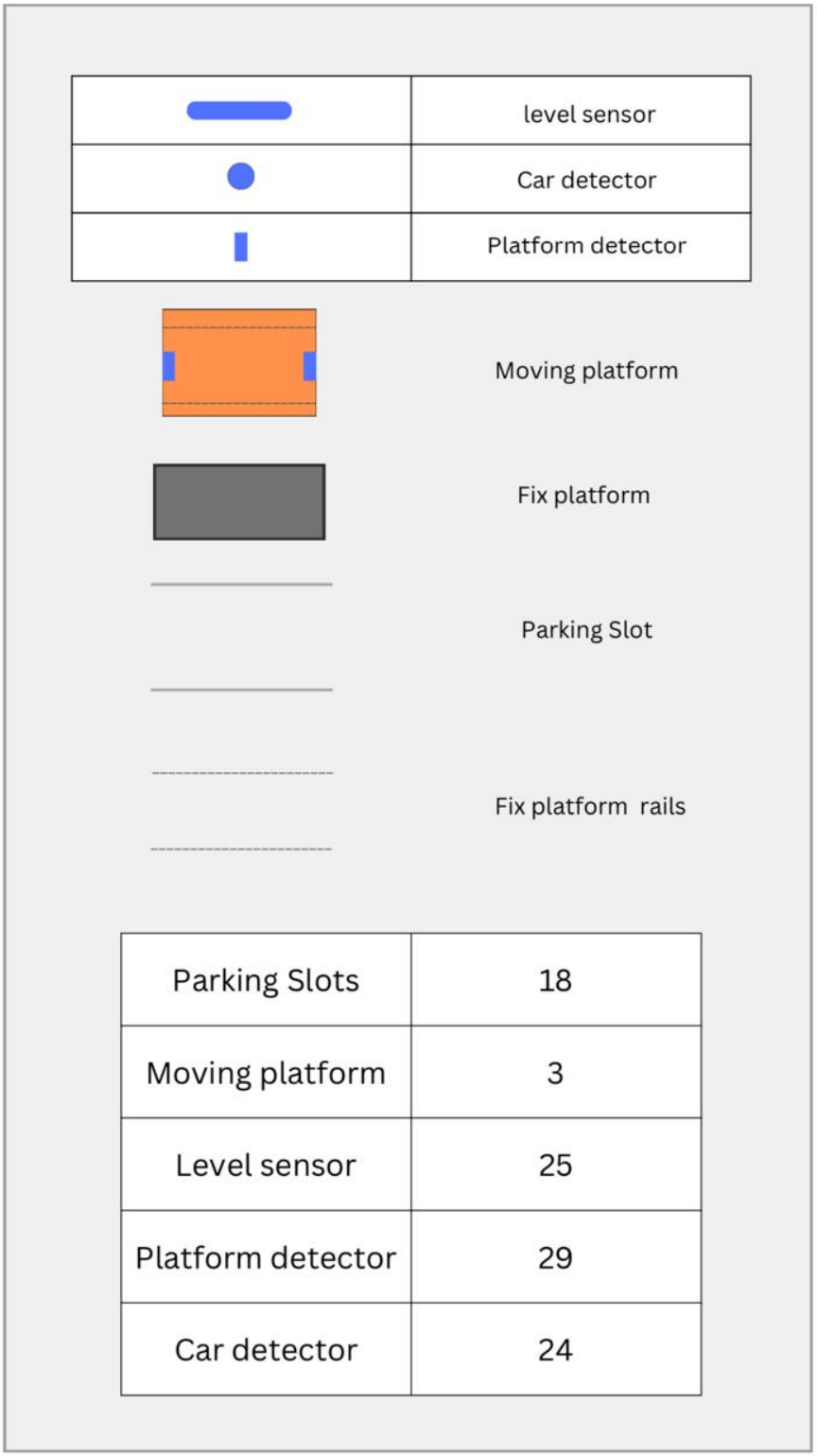
2. Project Objectives

- Design and develop ladder logic for PLC control of the parking system.
- Simulate the operation of the automated parking system.
- Integrate safety mechanisms within the system.
- Document system design, ladder logic, simulation process, and troubleshooting.

3. System Design

3.1 System Overview

- Multi-Story Parking Structure: A minimum of 3 levels.
  - Components: Entry and exit gates, ticket dispensers, car transfer mechanisms (elevator/conveyor system), parking slot sensors,
  level sensors, payment terminal, and emergency stop buttons.
  - Parking Slots and Detection: Parking slots on each level with sensors to detect the presence of a vehicle. Additionally, the system includes floor-level sensors for proper slot allocation.



Parking Slots	18
Moving platform	3
Level sensor	25
Platform detector	29
Car detector	24

Figure 1 System description and schematic label

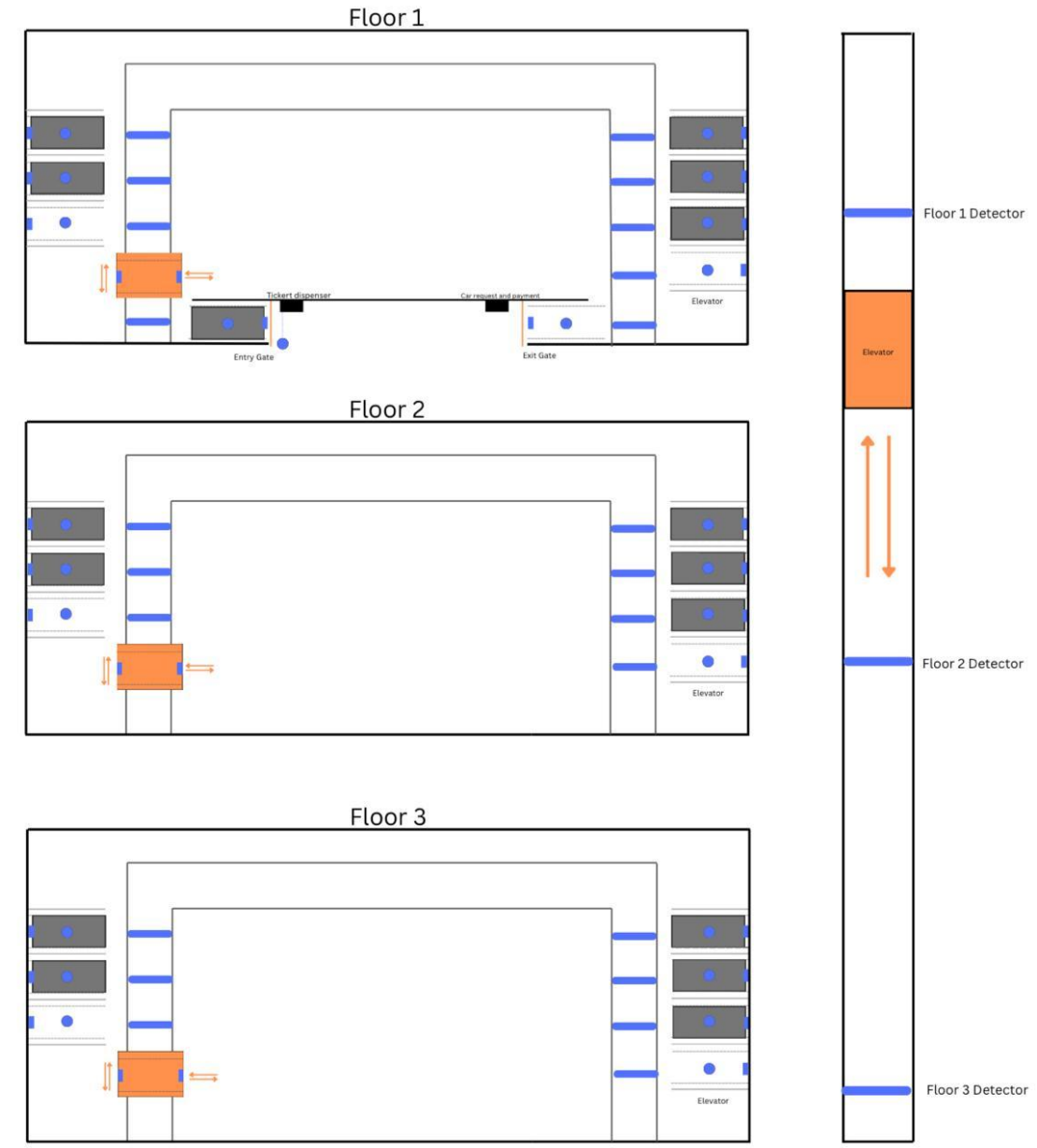


Figure 2 System schematic

3.2 System Components

- Entry Gate: A controlled barrier that allows cars to enter after ticket issuance.

- Exit Gate: A controlled barrier that ensures safe vehicle exit after successful payment.

- Ticket Dispenser: Issues unique tickets to cars entering the system.

-Car Transfer Mechanism: Uses motors, conveyors, or an elevator to move cars between levels and slots.

- Sensors: Detect the presence of cars in slots, platform position, and the entry/exit process.

- Payment Terminal: Handles payments before car exit

4. Data blocks and data management

To make the system robust, easy to use, and easy to edit, we created multiple data blocks to store the sensor values as arrays. We created several boolean arrays (0..8) to store the values of the platform detectors, car detectors, and position detectors for each floor. These values are saved in a specific order, meaning the index of the array is important. Each index represents the slot number: the entry gate has index 0, the exit gate has index 8, and the elevator is assigned index 7. All other slots are numbered based on their order, starting from the left and moving clockwise. We also added a function that runs continuously to populate these arrays with the input values.

To make use of these arrays, we utilized several SCL functions and a function loop to easily access any of the sensors with minimal code, making the system easier to edit and reuse. This approach helps make the system more compact. Some of the SCL functions are responsible for tasks such as searching for the moving platform's position, finding the nearest slot (whether it contains a platform or not), checking if a car is available in a specific slot, and determining the position of the moving platform.

Program blocks

Sensor Data [DB1]

Sensor Data Properties							
General							
Name	Sensor Data	**Number**	1	**Type**	DB	**Language**	DB
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Start value	Retain
Static			
ParkingSlot_Car_Detector 1	Array[0..8] of Bool		False
ParkingSlot_Car_Detector 2	Array[0..8] of Bool		False
ParkingSlot_Car_Detector 3	Array[0..8] of Bool		False
ParkingSlot_Car_Detector_OLD1	Array[0..8] of Bool		False
ParkingSlot_Car_Detector_OLD2	Array[0..8] of Bool		False
ParkingSlot_Car_Detector_OLD3	Array[0..8] of Bool		False
ParkingSlop_Platform_Detector 1	Array[0..8] of Bool		False
ParkingSlop_Platform_Detector 2	Array[0..8] of Bool		False
ParkingSlop_Platform_Detector 3	Array[0..8] of Bool		False
Platform 1 LevelSensor	Array[0..8] of Bool		False
Platform 2 LevelSensor	Array[0..8] of Bool		False
Platform 3 LevelSensor	Array[0..8] of Bool		False
Entry Gate Number	Int	0	False
Elevator Numer	Int	7	False
Exit Gate Number	Int	8	False

Figure 3 Data block- Sensory data

Program blocks

Saving Sensing values to arrays [FC1]

Saving Sensing values to arrays Properties							
General							
Name	Saving Sensing values to arrays	**Number**	1	**Type**	FC	**Language**	LAD
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Default value
Input		
Output		
InOut		
Temp		
Constant		
▼ Return		
Saving Sensing values to arrays	Void	

Network 1: Saving values from the Car detection sensors floor1 into arrays

Figure 4 Saving sensing values to array – Function (part of it)

Program blocks

Calculate the position of the Moving platform [FC5]

Calculate the position of the Moving platform Properties							
General							
Name	Calculate the position of the Moving platform	**Number**	5	**Type**	FC	**Language**	SCL
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Default value
▼ Input		
Platform level	Array[0..8] of Bool	
▼ Output		
Current_Position	Int	
InOut		
▼ Temp		
col	Int	
Constant		
▼ Return		
Calculate the position of the Moving platform	Void	

```
0001
0002 #"Current_Position" := -1;
0003
0004 // Loop through the 1D array
0005 FOR #col := 0 TO 8 DO
0006   IF #"Platform level"[#col] = TRUE THEN
0007     #"Current_Position" := #col;  // Store the first free slot (column index)
0008     EXIT;
0009   END_IF;
0010 END_FOR;
0011
0012
```

Figure 5 SCL code - example of the advantage of using the arrays

The same concept is applied to car drivers too.

Program blocks

Timers_variables [DB8]

Timers_variables Properties							
General							
Name	Timers_variables	**Number**	8	**Type**	DB	**Language**	DB
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Start value	Retain
Static			
CarTime1	Array[1..6] of Time		False
CarTime2	Array[1..6] of Time		False
CarTime3	Array[1..6] of Time		False

Figure 6 timers Data block

Bring Timer value [FC20]

Bring Timer value Properties							
General							
Name	Bring Timer value	**Number**	20	**Type**	FC	**Language**	SCL
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Default value
Input		
CarNumber	Int	
Floor	Int	
CarTime1	Array[1..6] of Time	
CarTime2	Array[1..6] of Time	
CarTime3	Array[1..6] of Time	
Output		
TimerValue	Time	
InOut		
Temp		
Constant		
Return		
Bring Timer value	Void	

```
0001 #TimerValue := T#0S;
0002 CASE #"Floor" OF
0003   1:  // Statement section case 1
0004     #TimerValue := #CarTime1[#CarNumber]
0005     ;
0006   2:  // Statement section case 2
0007     #TimerValue := #CarTime2[#CarNumber]
0008     ;
0009   3:
0010     #TimerValue := #CarTime3[#CarNumber]
0011     ;
0012 END_CASE;
```

Figure 7 SCL Code – Checking the value of a specific timer with minimal code

5. System algorithm

The system follows the Below sequence:

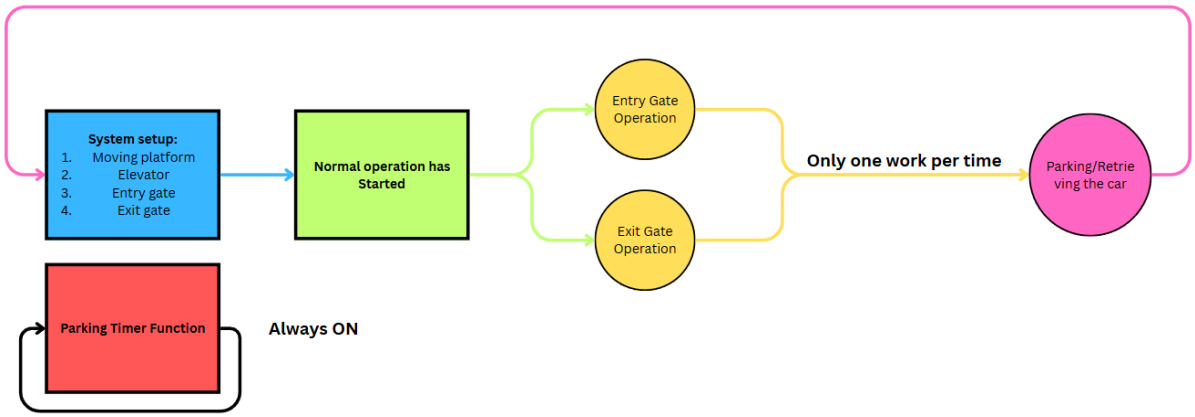


Figure 8 System sequence operation

5.1 System Set up

The system won't start working if the auto mode button is not pressed. After pressing the auto mode button, the setup mode will be activated. This mode ensures that everything is ideal to start the normal operation; if not, some actions will be taken to reach the ideal case to start normal operation.

The ideal case is when:

1. All the moving platforms and the elevator do not contain any fixed platforms.
2. There is a Fix platform at the entry gate and none at the exit.

5.1.1 Moving Platform Set up

The system will check sequentially each one of the moving platforms if they have scenario 3 or 2. To do so, a counter is used to count up each time the system reads one of the scenarios. And the reading will keep going while none of the scenarios is activated, and if some scenario has worked, the system will wait until it finishes and keep reading. This is represented in Figure below.

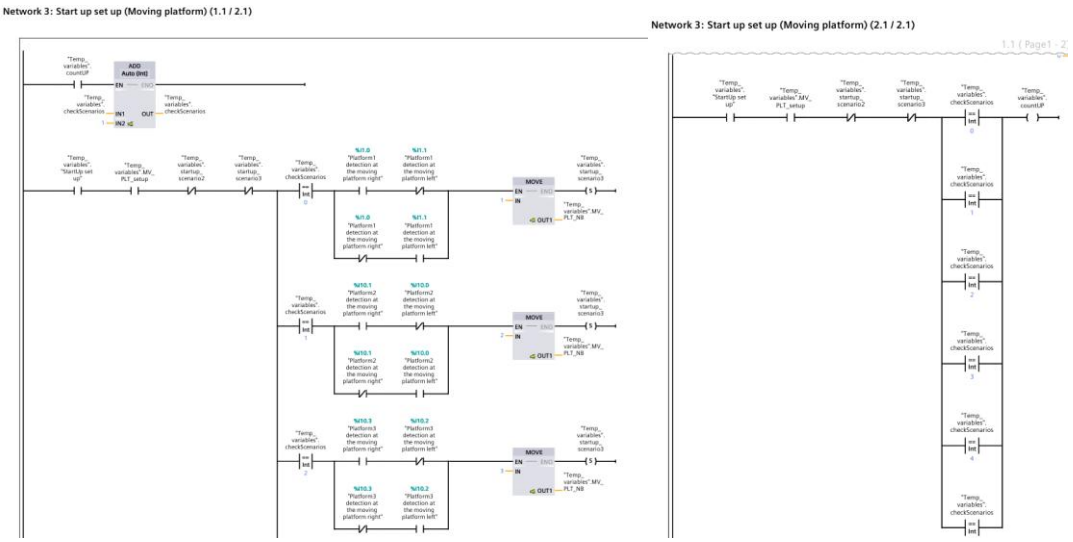


Figure 9 Moving platform setup—the logic of checking each scenario—checking for scenario 3

Scenario 3

is when the fixed platform is not fully attached to the moving platform.

Figure 4 shows the checking of the counter used and the checking for scenario 3 for each platform.

When Scenario 3 is activated, the system will check the position of the moving platform, and then the moving platform will swipe the fixed platform to the parking slot. The sweep will stay working until the two-platform sensor at the moving platform is low and the platform sensor in the parking slot is high. The Figure below Shows the entire logic of this operation.

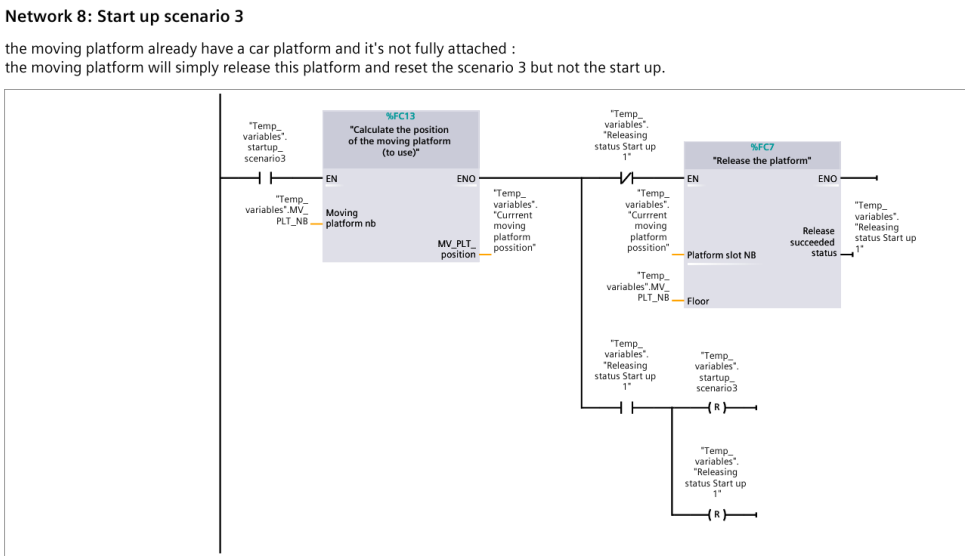
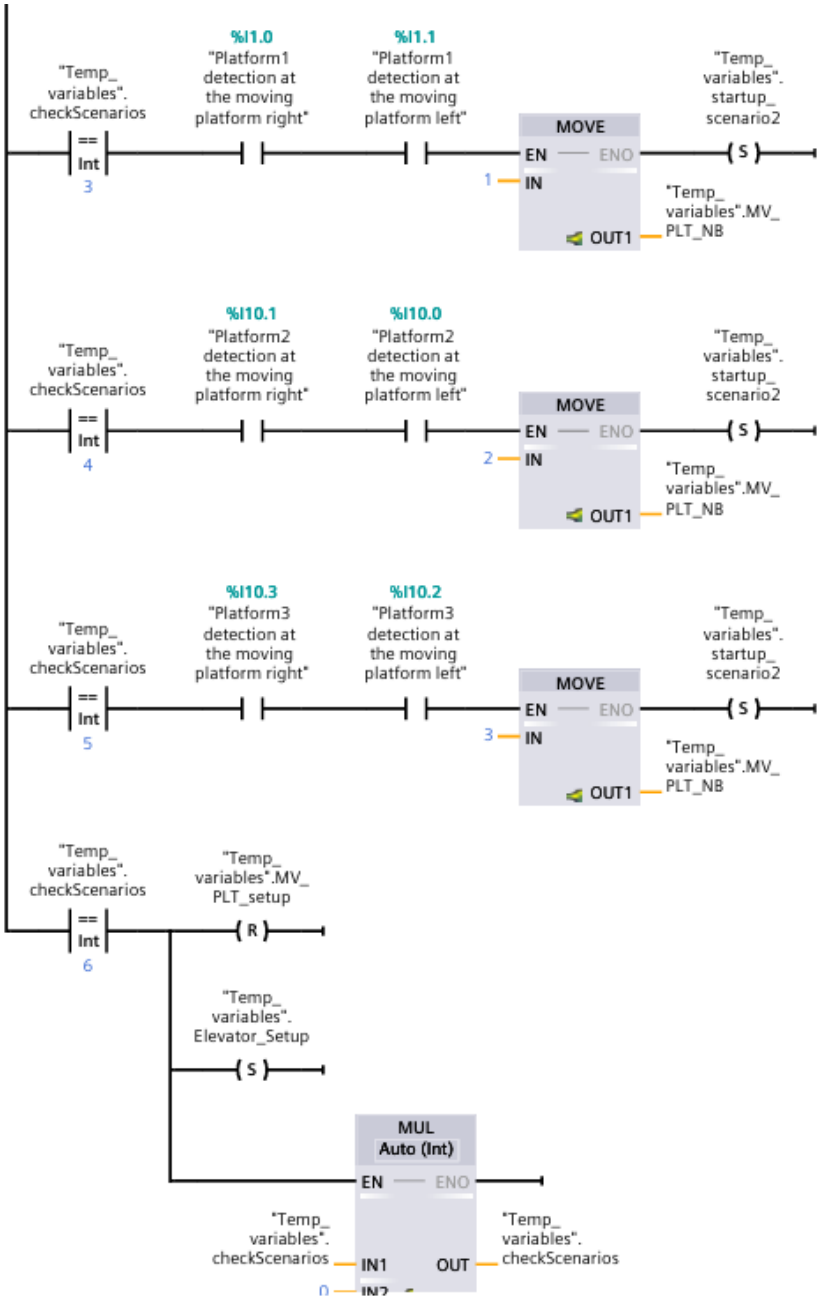


Figure 10 Scenario 3 Ladder code

Scenario 2

A scenario is when the fixed platform is fully attached to the moving one. The Figure below shows the logic of checking the scenario and the restart of the counter after finishing checking all the scenarios successfully.



2.1 (Page1 - 3)

Figure 11 Moving platform setup—Checking scenario 2—restart counter

When scenario 2 is activated, the system will check the nearest free slot that does not contain a fixed platform on the floor of the moving platform. If it doesn't find any, it will search in all the other floors.

To calculate the nearest slot that contains or does not contain a fixed platform, the system starts moving from the left (e.g., in the first floor, it will start checking from the entry gate position) and moving clockwise; the same applies at all the floors.

The Figure below shows the ladder logic of scenario 2 and the logic used to check if the nearest free fix slot is on the same floor or not.

- **Scenario 2.1:** The nearest slot that does not have a fixed platform is located on the same floor as the moving platform.
- **Scenario 2.2:** The nearest slot that does not have a fixed platform is situated on the other floor of the moving platform.

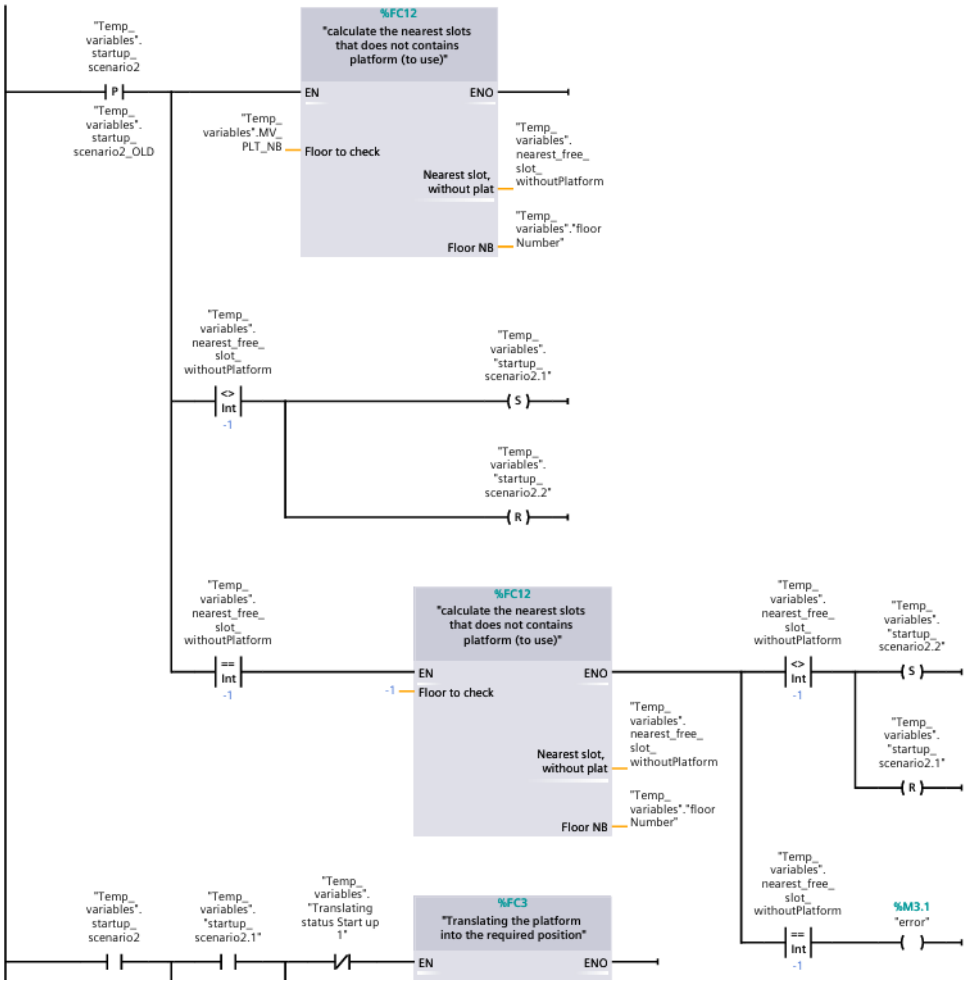


Figure 12 Moving Platform Setup—Scenario 2—Checking the position of the nearest slot without a fixed platform

Scenario 2.1

If this scenario is active, the system will:

1. Move the moving platform into the nearest slot that does not contain a fixed platform.
2. The moving platform will release the fixed platform.
3. Reset scenarios 2.1 and 2.

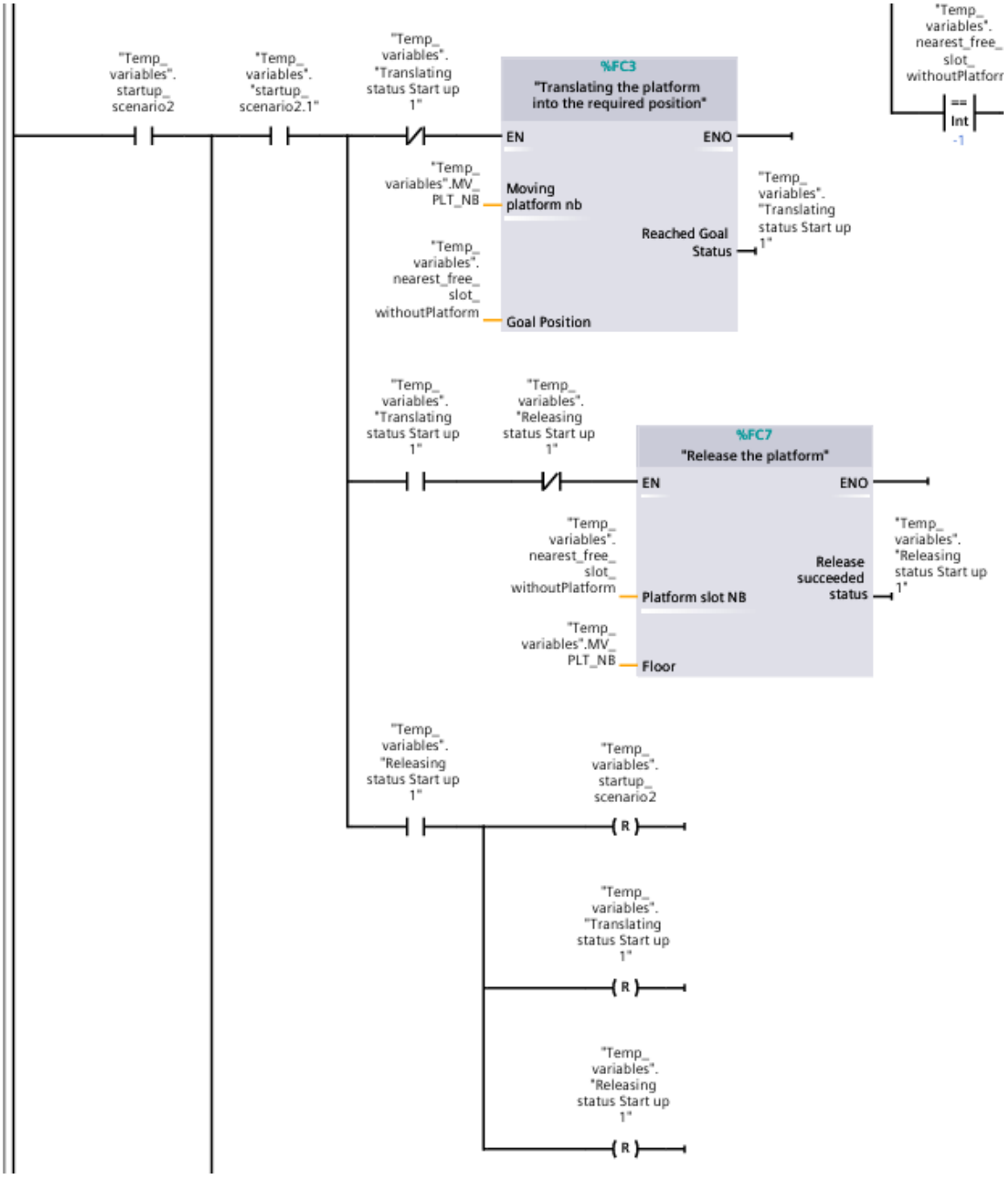


Figure 13 Moving Platform Setup—Scenario 2.1—Release the fixed platform in the nearest slot that does not contain a fixed platform on the same floor.

Scenario 2.2

If this scenario is activated, the system will:

1. Translate the elevator into the first position.
2. Move the platform into the elevator and release it.
3. Translate the elevator to the floor that contains the slot without a fixed platform.
4. The moving platform of this new floor will attach to the fixed platform, then move it to the new free float and release it.
5. Reset the scenario 2 and 2.2.

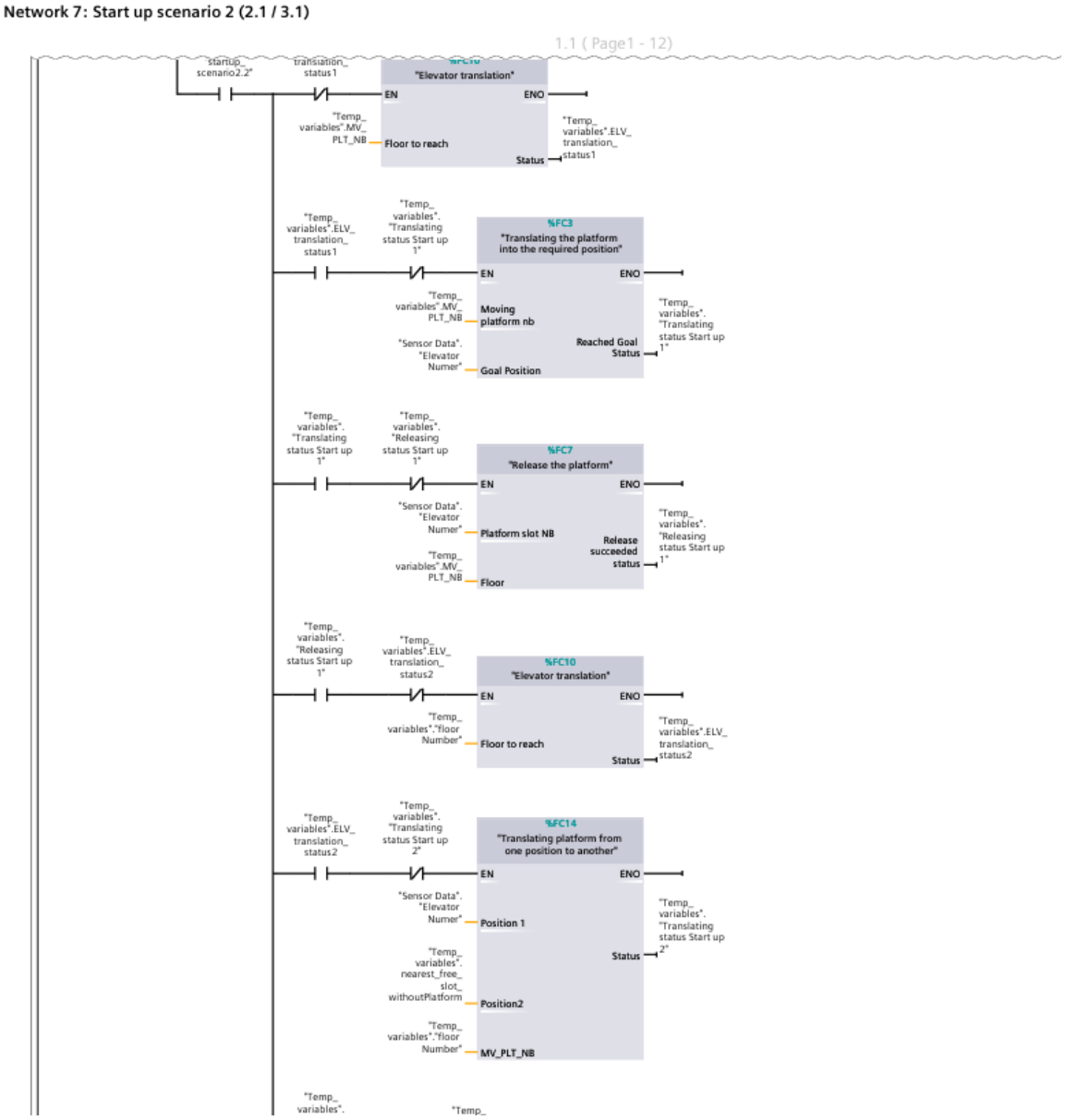


Figure 14 Moving Platform Setup—Scenario 2.2—Release the fixed platform in the nearest slot that does not contain a fixed platform on a different floor.

5.1.2 Elevator Set up

The system will verify whether a fixed platform is present in the elevator. If the elevator does not have a fixed platform, this process will conclude, and the system will proceed to check other configurations. The figure below shows the ladder logic of the elevator when there is no fixed platform.

Network 4: Start up set up (Elevator)

check if there is a platform in the elevator,
if there is one the system will search for the nearest slot that does not cointains a platform
the elevator will go to that floor
the moving platform will take this platform from the elevator to the free slot

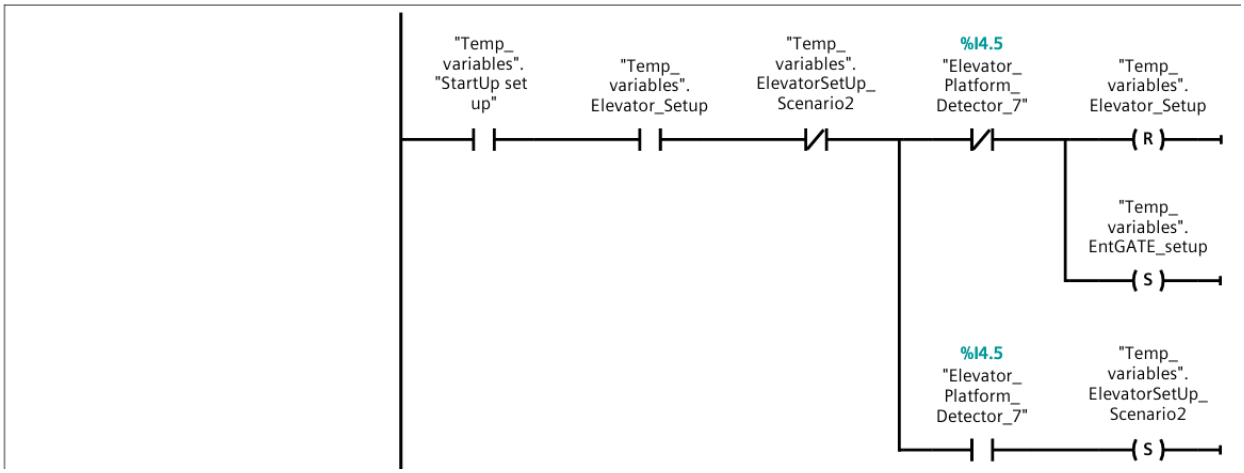


Figure 15 Elevator Setup: fixed platform in the elevator

If the elevator contains a fixed platform, the system will:

1. Calculate the nearest slot that does not contain a fixed platform. The calculation will start from floor 1 until floor 3, stopping when a free slot is detected.
2. The Elevator will translate to the floor that does not contain a fixed platform
3. The moving platform will take the fixed platform from the elevator and translate it to the required slot, then release it.
4. Reset the required memories.

The figure below shows this logic inside the TIA Portal environment.

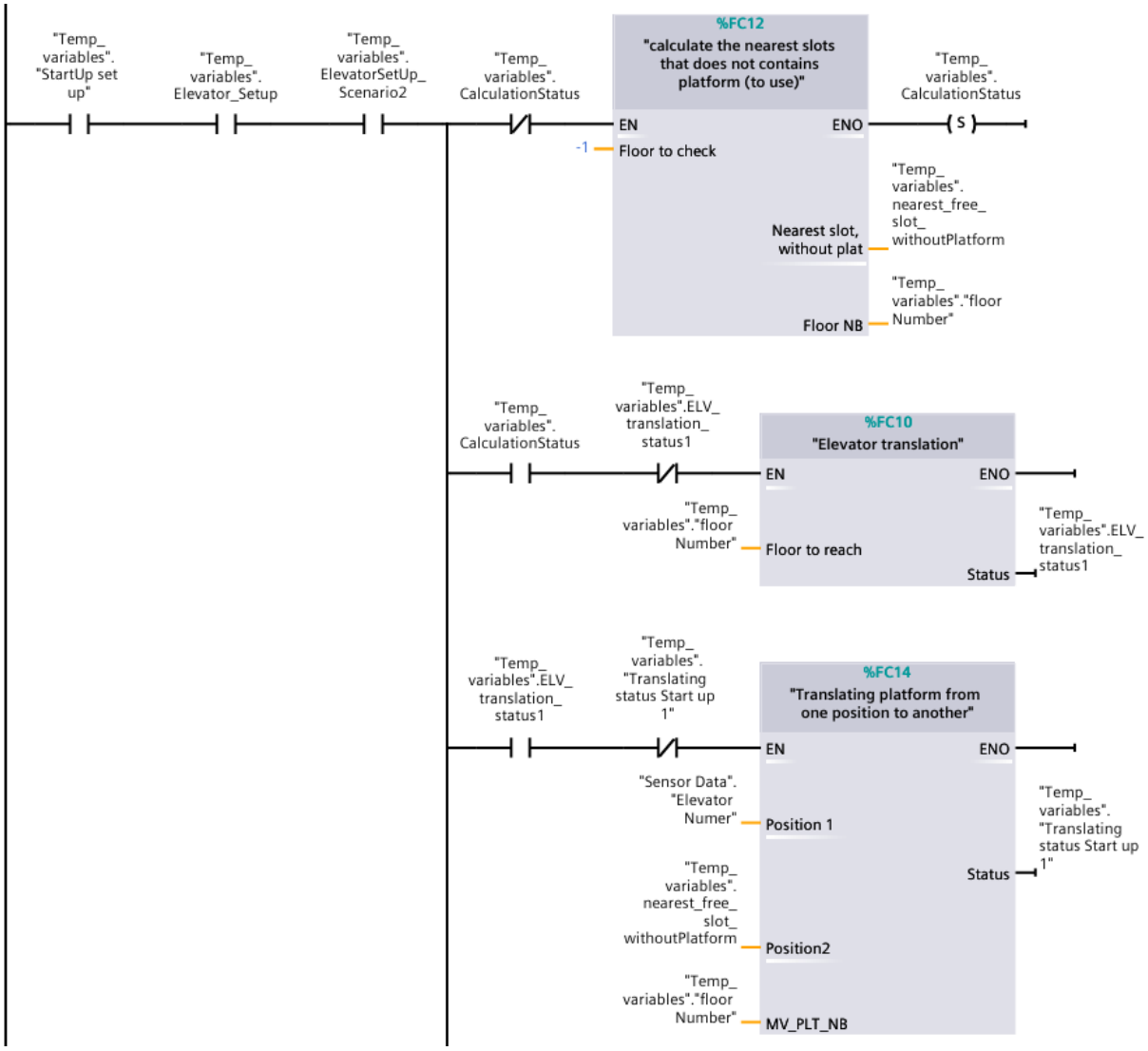


Figure 16 Elevator Setup- There is a fixed platform in the elevator

5.1.3 Entry Gate Set up

The system will calculate the nearest slot that contains a free platform and does not have a car.

After that, the system will check the floor of this free slot and act based on it.

1. If the free slot is on the first floor, **Scenario 1** will be activated
2. If the free slot is on the other floors, **Scenario 2** will be activated

The figure below shows the logic of this operation in the Tia portal.

Network 5: Start up set up (Entry gate) (1.1 / 2.1)

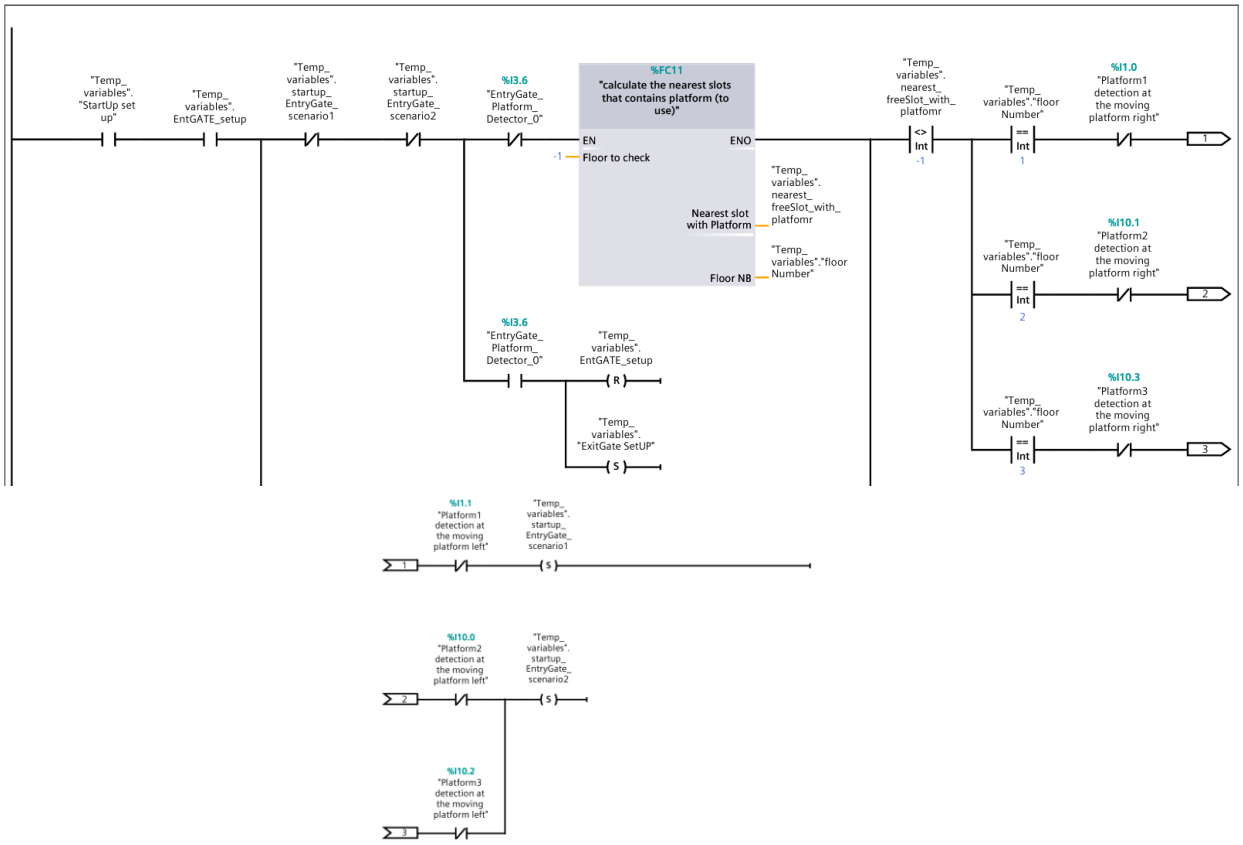


Figure 17 Entry gate setup - choosing the scenario

If **Scenario 1** is activated (the free slot is on the first floor), the system will bring the platform from the free slot and attach it to the entry gate. This entire process is available in a function called “Translating platform from one position to another,” and it’s represented in the figure below.

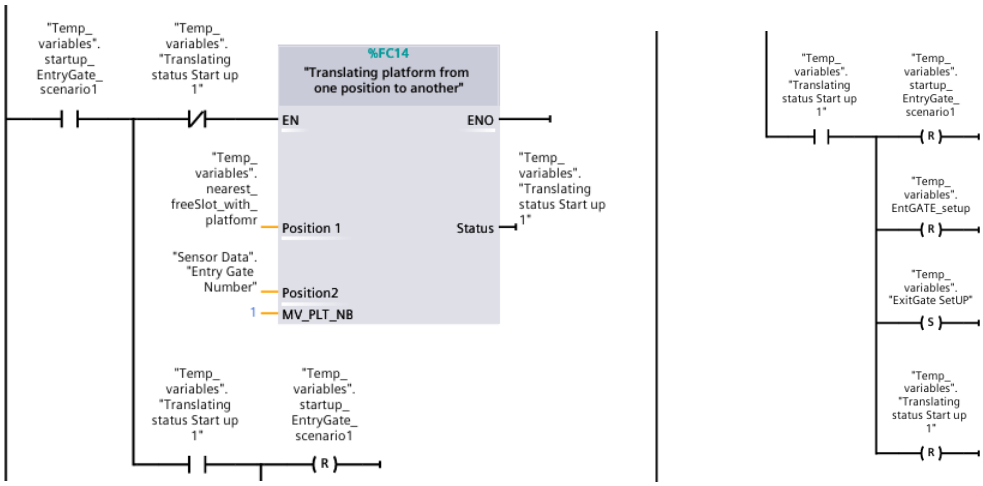


Figure 18 Entry gate setup- Scenario 1- Translating the platform from floor 1 to the entry gate

If **Scenario 2** is activated (the free slot is on the second or third floor), the system will bring the platform from the free slot and attach it to the entry gate, but the process will be more complex and will contain an additional process. This entire process is available in a function called “Translating platform from one Floor to another,” and it’s represented in the figure below.

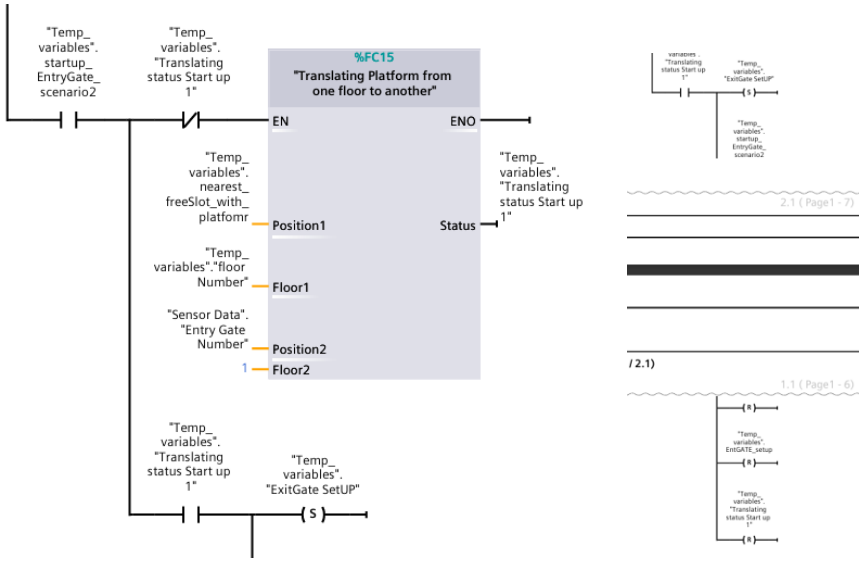


Figure 19 Entry gate setup- Scenario 2- Translating the platform from floor 2 or 3 to the entry gate

5.1.4 Exit Gate Set up

The exit gate setup works using the same logic as the entry gate. However, the key difference is that the system checks for the nearest slot without a fixed platform, rather than one with a fixed platform and no car. Once identified, the system moves the fixed platform from the exit gate to the target slot, which may be located on floor 1 (scenario 1) or on floor 2 or 3 (scenario 2). The same functions used previously—“Translating platform from one position to another” and “Translating platform from one floor to another”—are reused here, with the only change being the parameters passed to these functions.

5.2 Normal Operation

After the setup is completed, the system enters the ideal state and is ready for operation. An LED at both the entry and exit gates indicates the system’s status: green means the system is ready and the user can park or request their car, while red indicates that an operation is currently in progress.

From here, the user might start the entry or exit gate operation (only once per time).

5.2.1 Entry Gate

To start the operation, the car detector at the entry gate (located near the ticket dispenser) must be activated (high), and the ticket dispenser should also be triggered (high).

Once the system starts, it calculates the nearest available slot that does not contain a fixed platform and ensures that at least one free slot is available. This process is illustrated in the figure below.

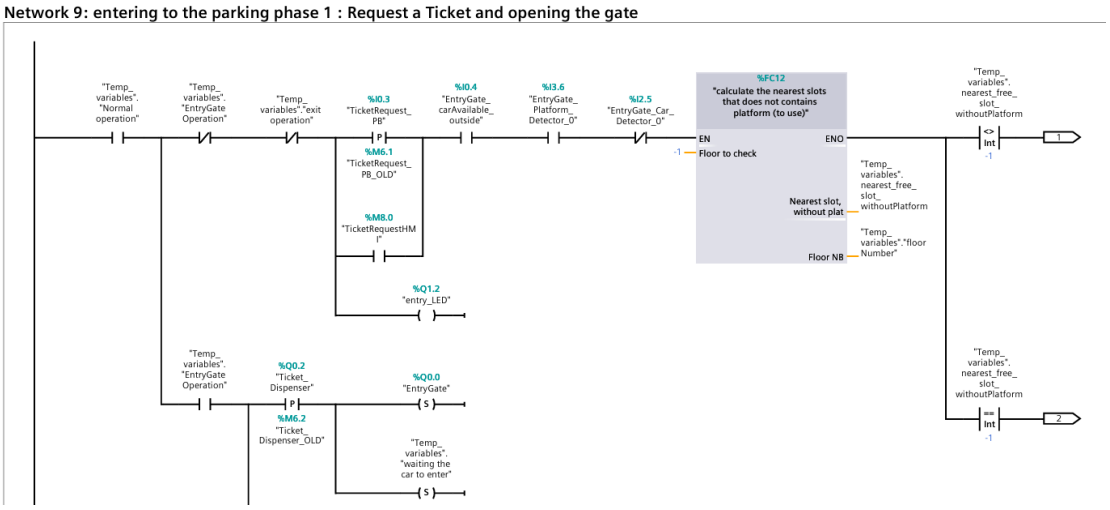


Figure 20 Normal Operation - Entry Gate - Ticket Request

After finding the nearest slot, the system assigns an ID number to the car. This ID consists of two digits: the first represents the floor number, and the second represents the slot number. The ticket dispenser then issued the ID card to the user. This process is illustrated in the figure below.

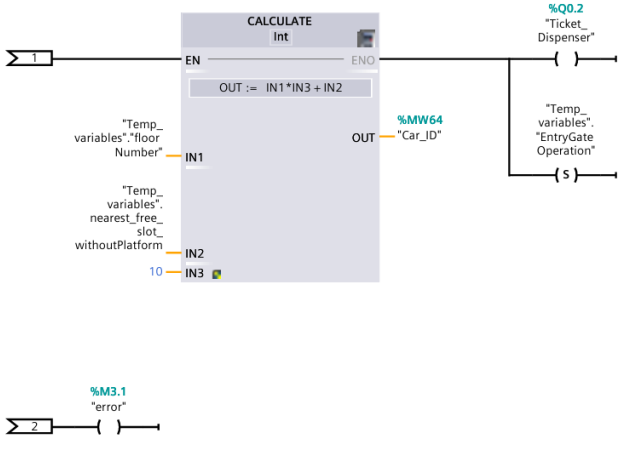


Figure 21 Normal Operation - Entry Gate - ID preparation

After activating the ticket dispenser, the entry gate becomes active. When the system detects that the car has entered—by receiving a high signal from the car detector at slot 0—it waits for 10 seconds before closing the gate and starting the translation operation, as shown in the figure below.

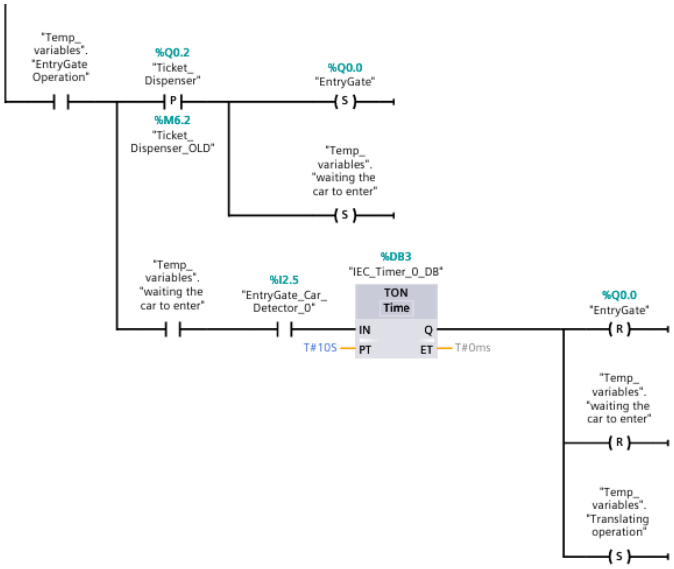


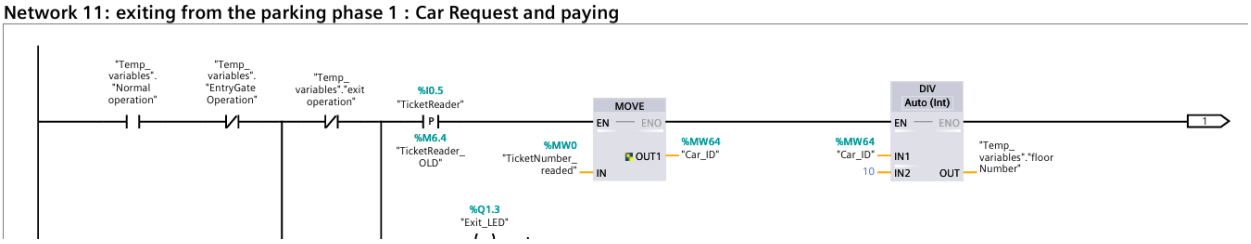
Figure 22 Normal Operation - Entry Gate – Opening and closing the gate

Now, the system will translate the platform to the required slot by using the previously mentioned functions—"Translating platform from one position to another" and "Translating platform from one floor to another"—with the appropriate parameters set accordingly.

After completing this step, the setup will be Set again, and the system will return to the ideal state, ready to process the next operation.

5.2.2 Exit Gate

To start the exit operation, the user shall put his ticket in the ticket reader. After that, the system will apply the required calculation to separate the two digits and save their values in memory. The system will apply a function to check if the car is available at this destination. If yes, the system will proceed to the billing operation. Below is the figure that represents the previous operation.



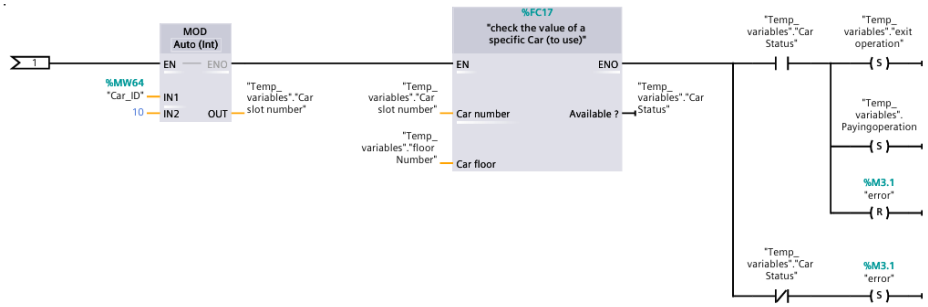


Figure 23 Normal Operation - Exit Gate – Reading the ticket value

The billing operation will start now, but before that, let's check the car timer function, which is always active in our system. For each slot in the system, there is a timer that starts when the car detector at that slot goes high and resets when the car detector reads a falling edge. The preset time (Pt) is set to 24 hours but can be adjusted to any suitable value. Below is a figure that shows this logic in the code for slot number 1.

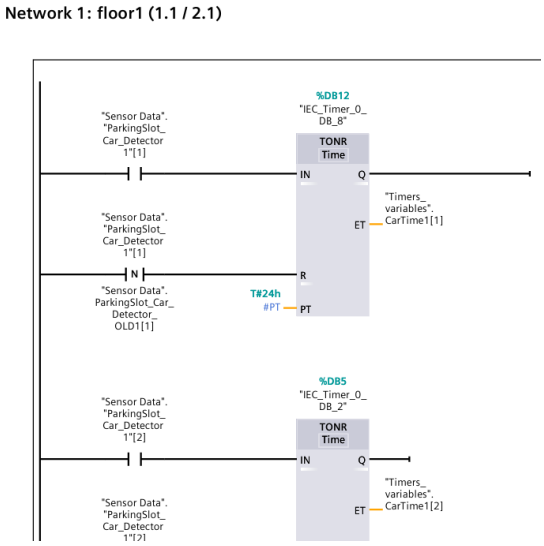


Figure 24 Car Timer Function

After understanding how this function works, we will now check the billing method in our system at the exit gate. A display will be added later to show the user the amount of money they need to pay, but for now, let's assume this feature is available. After the payment is made, the system will calculate the required amount using the billing function, as shown in the figure below. The system will then check if the paid amount is equal to or greater than the required amount and, if so, will start the exit operation.

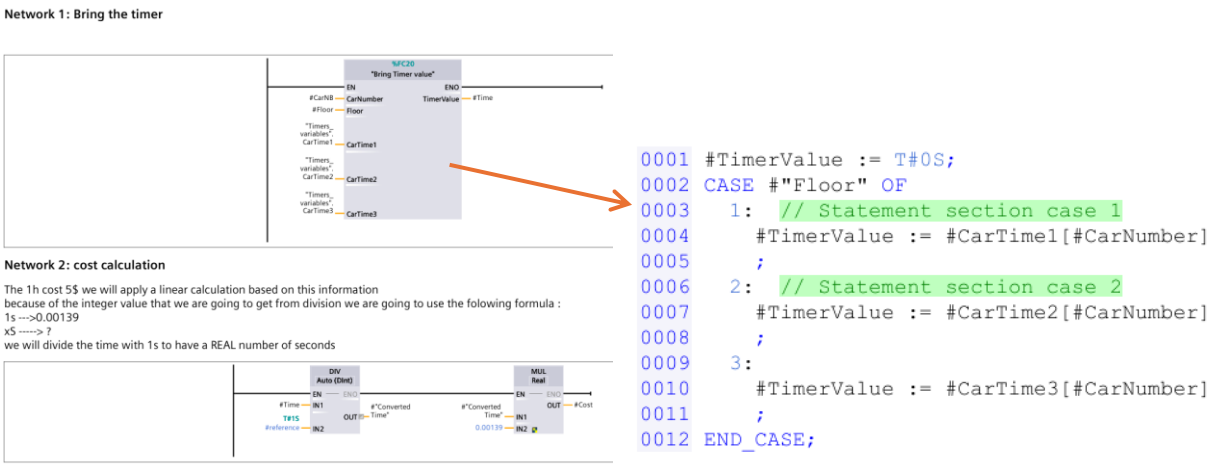


Figure 25 Preparing the car bill

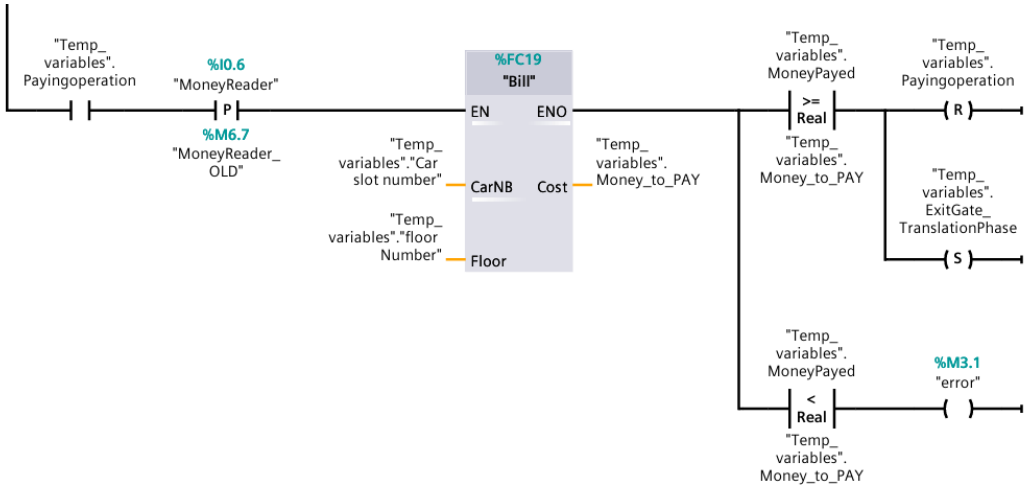


Figure 26 Normal Operation - Exit Gate – Paying operation

When the exit translation operation starts, the system will bring the specific car from its position to the exit gate using the two functions— “Translating platform from one position to another” and “Translating platform from one floor to another”—with the appropriate parameters set accordingly.

After bringing the car to the exit gate, the gate will open when the car detector reads a high signal. It will then close 10 seconds after detecting a falling edge from the car detector sensor at the exit gate. This process is like the one shown in Figure 17, but in this case, the “entryGate_carDetector” should be a normally closed contact.

After completing this step, the setup will be Set again, and the system will return to the ideal state, ready to process the next operation.

6. Simulation

For the simulation, we used PLC Sim to change the values of the inputs and visualize the system's outputs. Moreover, an HMI was designed to facilitate the visualization process. The HMI contains three screens—one for each floor—each displaying the values of the sensors and showing the exact operation of the moving platform, the elevator, and their positions.

We simulated the system by imagining how the inputs would change in the real world and manually modifying them using PLC Sim. For memory values, such as the ticket ID, we followed the same approach—manually changing the values either from TIA Portal itself or through PLC Sim.

We were not able to show the simulation results due to an error in our TIA Portal (V16) (something relating to license). We attempted to open the file in a different location, but the software version there was older than ours, so the file could not be opened.

Below is a figure that shows the HMI design for Floor 1.

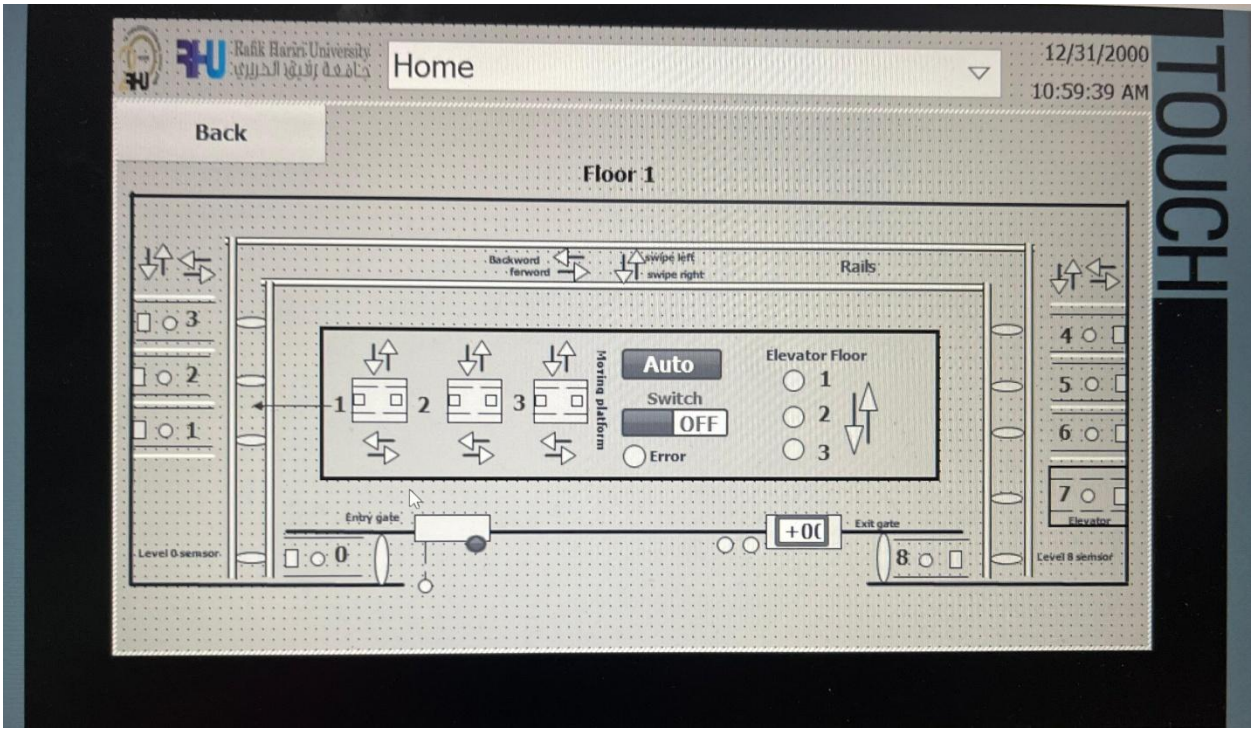


Figure 27 HMI Floor 1

Based on the previous figure, we can see that it is possible to start and stop the system, apply for a ticket, and visualize all the sensors in the system, as well as the operation of all actuators on every floor.

7. Conclusion

The Automated Car Parking System was successfully designed, simulated, and validated. The system integrates all necessary features, including entry/exit, parking slot allocation, car transfer, payment handling, and safety measures. The PLC-based control system operates efficiently, and troubleshooting efforts have resolved key issues

8. Future Improvements

- Adding error handling for different problems and scenarios (e.g., full parking, wrong ticket used). (Note: we used an error memory in every place where we anticipated a problem, but we did not specify the issue or define any corrective actions.)
- Fixing the cashier system at the exit gate to make it more realistic.
- Adding an alarm and notification screen to the HMI.
- Adding more customization options to the HMI, including a private page for modifying these parameters.
- Implementing priority parking for electric vehicles or VIP customers.
- Introducing reservation-based parking management through mobile applications.

9. PLC Tags

The PLC control system utilizes several types of tags organized into different floors and control rooms, as summarized below:

Main Tags Overview:

Entry and Exit System:

- TicketRequest_PB (Input)

- EntryGate (Output)

- ExitGate (Output)

- Ticket_Dispenser (Output)

- TicketReader (Input)

- MoneyReader (Input)

Platform Movement (Floor 1, 2, 3):

- PlatformX_Motor_forword, PlatformX_Motor_backword (Outputs)

- PlatformX_Motor_SwipeRight, PlatformX_Motor_SwipeLeft (Outputs)

- Floor sensors detect platform positions and car presence.

Elevator System:

- Elevator_Motor_Up, Elevator_Motor_Down (Outputs)

- Elevator_Detector_Floor_1, Elevator_Detector_Floor_2, Elevator_Detector_Floor_3 (Inputs)

Safety and Control Room Tags:

- Emergency_Sw (Input)

- AutoMode_PB, AutoMode_HMI (Manual and automatic mode switch)

- error, countUP, RESETcount (Memory control bits)

User Variables:

- Car_ID

- TicketNumber_readed

- TicketRequestHMI, Emergency_HMI (From HMI panel)

PLC tags / Floor1 [86]

PLC tags

PLC tags

Name	Data type	Address	Retain
TicketRequest_PB	Bool	%I0.3	False
EntryGate_carAvailable_outside	Bool	%I0.4	False
TicketReader	Bool	%I0.5	False
TicketNumber_readed	Int	%MW0	False
MoneyReader	Bool	%I0.6	False
EntryGate	Bool	%Q0.0	False
ExitGate	Bool	%Q0.1	False
Ticket_Dispenser	Bool	%Q0.2	False
Platform1 detection at the moving platform right	Bool	%I1.0	False
Platform1 detection at the moving platform left	Bool	%I1.1	False
Platform1_Motor_forword	Bool	%Q0.3	False
Platform1_Motor_Backword	Bool	%Q0.4	False
Platform1_Motor_SwipeRight	Bool	%Q0.5	False
Platform1_Motor_SwipeLeft	Bool	%Q0.6	False
Floor1_Platform_LevelSensor_ParkingSlot_1	Bool	%I1.3	False
Floor1_Platform_LevelSensor_ParkingSlot_2	Bool	%I1.4	False
Floor1_Platform_LevelSensor_ParkingSlot_0_Entrygate	Bool	%I1.2	False
Floor1_Platform_LevelSensor_ParkingSlot_3	Bool	%I1.5	False
Floor1_Platform_LevelSensor_ParkingSlot_5	Bool	%I2.1	False
Floor1_Platform_LevelSensor_ParkingSlot_6	Bool	%I2.2	False
Floor1_Platform_LevelSensor_ParkingSlot_7_elevator	Bool	%I2.3	False
Floor1_Platform_LevelSensor_ParkingSlot_8_exitGate	Bool	%I2.4	False
Floor1_Platform_LevelSensor_ParkingSlot_4	Bool	%I2.0	False
Floor1_ParkingSlot_Car_Detector_1	Bool	%I2.6	False
Floor1_ParkingSlot_Car_Detector_2	Bool	%I2.7	False
Floor1_ParkingSlot_Car_Detector_3	Bool	%I3.0	False
Floor1_ParkingSlot_Car_Detector_4	Bool	%I3.1	False
Floor1_ParkingSlot_Car_Detector_5	Bool	%I3.2	False
Floor1_ParkingSlot_Car_Detector_6	Bool	%I3.3	False
Floor1_ParkingPlatform_Detector_1	Bool	%I3.7	False
Floor1_ParkingPlatform_Detector_2	Bool	%I4.0	False
Floor1_ParkingPlatform_Detector_3	Bool	%I4.1	False
Floor1_ParkingPlatform_Detector_4	Bool	%I4.2	False
Floor1_ParkingPlatform_Detector_5	Bool	%I4.3	False
Floor1_ParkingPlatform_Detector_6	Bool	%I4.4	False
EntryGate_Car_Detector_0	Bool	%I2.5	False
EntryGate_Platform_Detector_0	Bool	%I3.6	False
ExitGate_Car_Detector_8	Bool	%I3.5	False
ExitGate_Platform_Detector_8	Bool	%I4.6	False
Elevator_Platform_Detector_7	Bool	%I4.5	False
Elevator_Car_Detector_7	Bool	%I3.4	False
Elevator_Motor_Up	Bool	%Q0.7	False
Elevator_Motor_Down	Bool	%Q1.0	False
Elevator_Detector_Floor_1	Bool	%I4.7	False
Elevator_Detector_Floor_2	Bool	%I5.0	False
Elevator_Detector_Floor_3	Bool	%I5.1	False
entry_LED	Bool	%Q1.2	False
Exit_LED	Bool	%Q1.3	False
Car_ID	Int	%MW64	False
TicketRequestHMI	Bool	%M8.0	False

PLC tags / ControlRoom [7]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	error	Bool	%M3.1	False
	AutoMode_PB	Bool	%I0.0	False
	Emergency_Sw	Bool	%I0.2	False
	countUP	Bool	%M6.5	False
	RESETcount	Bool	%M6.6	False
	AutoMode_HMI	Bool	%M7.0	False
	Emergency_HMI	Bool	%M7.2	False

PLC tags / Floor2 [25]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Platform2_Motor_forword	Bool	%Q2.0	False
	Platform2_Motor_backword	Bool	%Q2.1	False
	Floor2_Platform_LevelSensor_ParkingSlot_1	Bool	%I8.0	False
	Floor2_Platform_LevelSensor_ParkingSlot_2	Bool	%I8.1	False
	Floor2_Platform_LevelSensor_ParkingSlot_3	Bool	%I8.2	False
	Floor2_Platform_LevelSensor_ParkingSlot_4	Bool	%I8.3	False
	Floor2_Platform_LevelSensor_ParkingSlot_5	Bool	%I8.4	False
	Floor2_Platform_LevelSensor_ParkingSlot_6	Bool	%I8.5	False
	Floor2_Platform_LevelSensor_ParkingSlot_7_elevator	Bool	%I8.6	False
	Platform2 detection at the moving platform left	Bool	%I10.0	False
	Platform2 detection at the moving platform right	Bool	%I10.1	False
	Platform2_Motor_SwipeRight	Bool	%Q1.5	False
	Platform2_Motor_SwipeLeft	Bool	%Q1.4	False
	Floor2_ParkingSlot_Car_Detector_1	Bool	%I5.4	False
	Floor2_ParkingSlot_Car_Detector_2	Bool	%I5.5	False
	Floor2_ParkingSlot_Car_Detector_3	Bool	%I5.6	False
	Floor2_ParkingSlot_Car_Detector_4	Bool	%I5.7	False
	Floor2_ParkingSlot_Car_Detector_5	Bool	%I6.0	False
	Floor2_ParkingSlot_Car_Detector_6	Bool	%I6.1	False
	Floor2_ParkingPlatform_Detector_1	Bool	%I6.2	False
	Floor2_ParkingPlatform_Detector_2	Bool	%I6.3	False
	Floor2_ParkingPlatform_Detector_3	Bool	%I6.4	False
	Floor2_ParkingPlatform_Detector_4	Bool	%I6.5	False
	Floor2_ParkingPlatform_Detector_5	Bool	%I6.6	False
	Floor2_ParkingPlatform_Detector_6	Bool	%I6.7	False

PLC tags / floor 3 [25]

PLC tags

PLC tags

Name	Data type	Address	Retain
Platform3_Motor_forword	Bool	%Q3.0	False
Platform3_Motor_backword	Bool	%Q3.1	False
Floor3_Platform_LevelSensor_ParkingSlot_1	Bool	%I9.0	False
Floor3_Platform_LevelSensor_ParkingSlot_2	Bool	%I9.1	False
Floor3_Platform_LevelSensor_ParkingSlot_3	Bool	%I9.2	False
Floor3_Platform_LevelSensor_ParkingSlot_4	Bool	%I9.3	False
Floor3_Platform_LevelSensor_ParkingSlot_5	Bool	%I9.4	False
Floor3_Platform_LevelSensor_ParkingSlot_6	Bool	%I9.5	False
Floor3_Platform_LevelSensor_ParkingSlot_7_elevator	Bool	%I9.6	False
Platform3 detection at the moving platform left	Bool	%I10.2	False
Platform3 detection at the moving platform right	Bool	%I10.3	False
Platform3_Motor_SwipeRight	Bool	%Q1.6	False
Platform3_Motor_SwipeLeft	Bool	%Q2.2	False
Floor3_ParkingSlot_Car_Detector_1	Bool	%I7.0	False
Floor3_ParkingSlot_Car_Detector_2	Bool	%I7.1	False
Floor3_ParkingSlot_Car_Detector_3	Bool	%I7.2	False
Floor3_ParkingSlot_Car_Detector_4	Bool	%I7.3	False
Floor3_ParkingSlot_Car_Detector_5	Bool	%I7.4	False
Floor3_ParkingSlot_Car_Detector_6	Bool	%I7.5	False
Floor3_ParkingPlatform_Detector_1	Bool	%I7.6	False
Floor3_ParkingPlatform_Detector_2	Bool	%I7.7	False
Floor3_ParkingPlatform_Detector_3	Bool	%I10.4	False
Floor3_ParkingPlatform_Detector_4	Bool	%I10.5	False
Floor3_ParkingPlatform_Detector_5	Bool	%I10.6	False
Floor3_ParkingPlatform_Detector_6	Bool	%I10.7	False

PLC tags / OLD [6]

PLC tags

PLC tags

Name	Data type	Address	Retain
TicketRequest_PB_OLD	Bool	%M6.1	False
EntryGate_OLD	Bool	%M6.3	False
Ticket_Dispenser_OLD	Bool	%M6.2	False
AutoMode_PB_OLD	Bool	%M6.0	False
TicketReader_OLD	Bool	%M6.4	False
MoneyReader_OLD	Bool	%M6.7	False

10. References

- Petruzella, Frank D. *Programmable Logic Controllers*, 5th Edition. McGraw-Hill, 2016. [Link](https://www.mheducation.com/highered/product/programmable-logic-controllers-petruzella/M9780073510880.html)
- Siemens. *PLC Programming Manual – S7-1200/1500 Series*. [Link](https://support.industry.siemens.com/cs/products?pnid=14672&mlfb=6ES7211-1AE40-0XB0)