Control of a Four-Floor Smart Elevator

Prepared For

Pavlos Paleologou, Mechatronics and Robotics Program Head - BCIT

Prepared by

Taeyoon Rim (A01160214)

Submitted on May 9, 2021

Acknowledgements

I would like to thank everyone who have supported the development of this project. I would like to send my sincere appreciation toward Pavlos Paleologou for teaching and guiding us through the course, preparing this project, giving us useful tips and helps and making the physical lab access possible during unprecedented times. I would also like to thank Isaiah Regacho and Brian Gaensbauer for instructing and supervising us, and providing support during labs and the project. Additionally, I would like to acknowledge Greg Scutt and BCIT Safety, Security & Emergency Management for successfully complying with the COVID-19 safety plan and for making physical lab access possible. Last but not least, I would like to thank Matthew Rockall for his guidance through the course and technical documents for this project.

Executive Summary

All parts of the project including multiple calls, door control using AOI, emergency stop, statistical analysis, four floors, fault detection, networking and HMI have been completed, tested and are ready to be demonstrated.

The elevator program was developed for a pre-assembled elevator board. A full system for this project consists as follows:

- 1 PLC and 1 elevator board (from part #1 to part #8),
- 2 PLCs and 2 elevator boards (part #9)
- 2 PLCs, 2 elevator boards and 2 HMI controllers (part #10)

Please see Appendix E to learn how to use the HMI software.

The main elevator logic and door control logic uses state method #3. For details, please see Appendix A for state diagrams.

Table of Contents

Table of Figures

Table of Tables

Introduction

This project involved the implementation of an elevator algorithm with statistical analysis, fault detection, and Ethernet/IP communication for a four-floor smart elevator and was prepared for Pavlos Paleologou. Mechatronics and Robotics Program Head, at British Columbia Institute of Technology.

Written in 10 parts, the Ladder logic programs for the control of a four-floor smart elevator were developed using Studio 5000 Logix Designer® on Allen-Bradley 1769-L30ER CompactLogix™ PLCs. It was programmed with general instructions, subroutines, tasks, UDTs and AOIs to process the inputs captured from limit switches and pushbuttons on a pre-built workbench to actuate indicator lights, solenoid, motor, and relay, based on its algorithm.

The purpose of this project was to get familiarized with the reference manual and to apply lecture materials in a real-world industrial setting by writing programs that are clear and easy to expand.

Table 1 - List of completed projects and the date completed

Part #	Description	Completed	Date Completed
1	Three floors, one floor call at a time, and no door	Y	8-4-2020
2	Three floors, inside panel	Y	15-4-2020
3	Three floors, multiple calls	Y	17-4-2020
4	Three floors, add door control	Y	22-4-2020
5	Three floors, emergency button	Y	22-4-2020
6	Three floors, add statistical analysis	Y	24-4-2020
7	Four floors	Y	24-4-2020
8	Four floors, fault detection	Y	26-4-2020
9	Produce and consume a tag using the Ethernet/IP protocol	Y	6-5-2020
10	Four floors, HMI implementation	Y	8-5-2020

Project Description

Process Definition

When the elevator is first started, the elevator will move to the 1st floor and wait for user interaction.

When a floor button or a inside panel button for any floor is pressed, elevator will service the floor request. The user must be able to request multiple floor calls while the elevator is still servicing the floor or is in idle state. When the buttons are pressed, the indicator light corresponding to the pushbutton will be latched until the floor is serviced.

When a user presses PBE5, elevator goes into emergency state and the user can only open DOOR (the DOOR cannot be closed in emergency state). To recover from the emergency state, the administrator must press Restart button located in Administration tab of the HMI screen.

The elevator will record the number of floor calls and the average service time for each floor. The data gathered in this step is used to determine whether a floor of the elevator needs a maintenance. It will notify the operator through HMI screen or ILTS0.

The elevator will determine faults while it is in normal operation (Current_State of the main elevator logic is in between 0 and 5, inclusive). Please refer to Appendix D for the list of faults, conditions, and possible causes.

The elevator uses Ethernet/IP protocol to communicate with another PLC. Two PLC then exchange information and perform statistical analysis to notify that they need a maintenance.

The elevator is monitored and controlled using HMI by operators and administrators.

Please see Appendix A for the state diagrams of the main elevator logic and Door_Control logic.

Implementation

Multiple floor calls

The core components for achieving multiple floor calls are subroutines UpdateFloorRequest, UpdateDirection, and UpdateNextFloor. During normal operation, they automatically update all required variable for continuous servicing.

Door control

Using Door_Control, Door can only be controlled at State #2 of the main elevator logic. It implements the state method #3 in a environment that is separated from the main elevator logic to perform state changes and energize the actuator.

Statistical Analysis

Statistical analysis is done in subroutine UpdateAnalytics. When the door is fully closed after a service, It records the floor calls and average service time into variable Analytics. Which is an array created from UDT Analysis. This UDT is composed of DINT Floor_Calls, DINT Average_Service_Time and TIMER Service_Timer.

Fault Detection

Subroutine FaultDetection is run on FaultDetectionTask, which is a periodic task that is run every 10 milliseconds. It constantly looks for active fault condition and records the number of occuerence and the timestamp of when it occurred in a variable Logs. Logs is an array created from UDT FaultLog, which contains DINT Occurrence, LINT[30] Timestamp and BOOL Status.

Ethernet/IP protocol

The produced and consumed of type UDT SharedData are created to exchange data at set interval. UDT SharedData consists of UDT Errors[3] Errors, UDT Stats Stats, BOOL Is_Maintenance_Required, BOOL Maintenance_Completed. Where UDT Stats is composed of DINT[4] Floor_Calls and DINT[4] Average_Service_Time and UDT Errors consists of DINT Error_Code and BOOL Status. Another UDTs are defined inside SharedData to organize the data in a logical way and to make it easier to monitor tags. Subroutine SyncSharedData is used to run required logic.

HMI

HMI software was developed for X2 Pro 7 B2 HMI screen (made by Beijer Electronics) using iX Developer. The software is composed of 3 tabs and 2 pop-up dialog for log-in and display settings, and uses Text Library and controller tags to exchange and display information in and out of PLC. Subroutine SyncHMI is used to run required logic to upload or utilize the data. For more detail, please see Appendix E.

Program Description

MainRoutine

Lung 0: Jumps to MapInputsAndIndicators subroutine to capture pushbuttons, limit switches and indicator lights status.
Lungs 2-3: Processes emergency request.
Lungs 5-6: Processes restart request.
Lungs 8-15: Sets the corresponding bits in Floor_request.
Lung 17: Jumps to SyncSharedData subroutine.
Lung 18: Jumps to SyncHMI subroutine.
Lung 20: Initializes Current_State and Next_State on the first scan.
Lung 21: Jumps to InitializeElevator subroutine on the first scan.
Lung 22: Update Current_Floor when available.
Lungs 24-34: Moves from a state to a state when conditions are met.
Lungs 36-40: Internal actions; updates necessary variables to support multiple floor calls.
Lungs 42-45: Internal actions; updates necessary variables to control DOOR.
Lung 47: Internal actions; updates necessary variables to perform statistical analysis.
Lungs 49-52: Internal actions; updates necessary variables to handle emergency state.
Lungs 54-55: Energize outputs.

InitializeElevator

Lung 0: Initializes Direction and clears Floor_Request.
Lung 1: Initializes elevator after restart by resetting Restart_Request and int_indicatorLights, then closing DOOR.

InitializeFloorVariables

Lung 0: Sets Current_Floor and Next_Floor to 1 after arriving at the 1st floor.

InitializeStatistics

Lung 0: Resets timers used for measuring service time to prevent timing after elevator restart.
Lungs 1-2: Resets Floor_Calls and Average_Service_Time for all floors.

MapInputsAndIndicators

Lung 0: Maps int_pushbuttons from the input module.
Lungs 1-7: Overrides virtual HMI pushbuttons if available.
Lung 8: Maps int_limitSwitches from the input module.
Lungs 10-18: Latches indicator lights.
Lung 20: Maps int_indicatorLights to the output module.

UpdateFloorRequest

Lung 0: Converts Current_Floor into its binary counterpart (because Floor_Request is in binary format).
Lung 1: Updates (clears bits in) Floor_Request.

UpdateDirection

Lung 0: Converts Current_Floor into its binary counterpart (because Floor_Request is in binary format) and clears Available_Floor.
Lungs 2-9: Finds all floors that you could move to in binary format. For example, if elevator is moving in up direction and you are at 2nd floor, you can only go to 3rd or 4th floor.
Lung 10: Masks previous information with Floor_Request to find Available_Floor, which is the floor(s) that you could move to and is/are requested to go to.
Lung 12: Changes direction if there is no Available_Floor.

UpdateNextFloor

Lung 0: Converts Current_Floor into its binary counterpart (because Floor_Request is in binary format).
Lungs 1-8: Updates Next_Floor based on Direction, Floor_Request and Current_Floor.

Door_Control

Lung 0: Initializes timers.
Lung 1: Reset DoorTimer when PBO is pressed.
Lung 3: Initializes Current_State and Next_State on the first scan.
Lung 4: Initializes Current_State, Next_State and DoorTimer on OnService (just arrived at State #2 of the main elevator logic).
Lung 5: Update Current_Floor when available.
Lungs 7-10: Moves from a state to a state when conditions are met.
Lungs 12-14: Internal actions; updates necessary variables to control DOOR.
Lung 16: Energize outputs.

UpdateAnalysis

Lung 0: No actions.
Lung 1: Starts measuring time for floors with Floor_Request.
Lung 2: Stops measuring time when elevator arrives at the requested floor.
Lung 3: Statistical analysis result for maintenance bit set.

FaultDetection

Lungs 0-2: Detects fault corresponding to error code 1001; starts the timer when DOOR is active and DCLS is not, until DCLS detects the door. If the ACC of the timer is greater than 5000 milliseconds, fault will be recorded.
Lungs 4-5: Detects fault corresponding to error code 1002; uses a Boolean equation to detect if multiple floor limit switches are registered. If more than one switch is active at the same time, fault will be recorded. When this fault occurs, the elevator logic will go into emergency state.
Lungs 7-8: Detects fault corresponding to error code 1003; compares Occurrence of a fault log to its maximum length (30). If it is greater or equal to the maximum length, the fault will be recorded.

WriteLog

Lung 0: Records how many times a fault occurred and when it occurred. The time stamp is encoded as Unix Epoch time; it describes how many microseconds have elapsed since the Epoch (January 1, 1970 00:00:00 UTC).

SyncSharedData

Lungs 0-2: Receives maintenance alert from the other PLC then sets ILTS0 if a maintenance is needed. HMI_Elevator_Reset must be pressed to recover from maintenance alert(s).
Lungs 4-8: Determines if the other PLC requires a maintenance by analyzing their Floor_Calls and Average_Service_Time. For the testing purpose, Floor_Calls_Threshold was set to 2 (3 floor calls will set the Is_Maintenance_Required).
Lungs 10-13: Updates Error_Code and Status for any fault.
Lungs 14-15: Updates the data required for statistical analysis to be performed by the other PLC.

SyncHMI

Lungs 0-2: Determines HMI_Main_Status for HMI status bar. Priority is given to emergency state > maintenance needed > normal operation.
Lungs 4-7: Receives reset requests for Floor_Calls and Average_Service_Time for each floor (analytics).

Conclusions

All parts of the project have been successfully completed:

- Three floors, one floor call at a time, and no door
- Three floors, inside panel
- Three floors, multiple calls
- Three floors, add door control
- Three floors, emergency button
- Three floors, add statistical analysis
- Four floors
- Four floors, fault detection
- Produce and consume a tag using the Ethernet/IP protocol
- Four floors, HMI implementation

The program was tested during the development phase and all bugs that was found during the testing has been troubleshooted.

Recommendations

Because of the way the project is defined, the program was developed as how it was outlined. Here are few recommendations to improve the project:

- Better HMI organization
- Door Control AOI

HMI could be better organized by splitting Administration tab. Administration tab of the HMI software contains information and user interactions for both PLCs and may confuse the user. It is possible to enhance the software clarity by moving the Restart button to another screen.

Door, in this program, is a special output actuator that is outside the control of the main elevator logic. Which means, in the output energize section of the main elevator logic, the door cannot be fully controlled. For the program readability and expandability, it would be great idea to include the door into the main elevator logic.

Appendices

A. State Diagrams

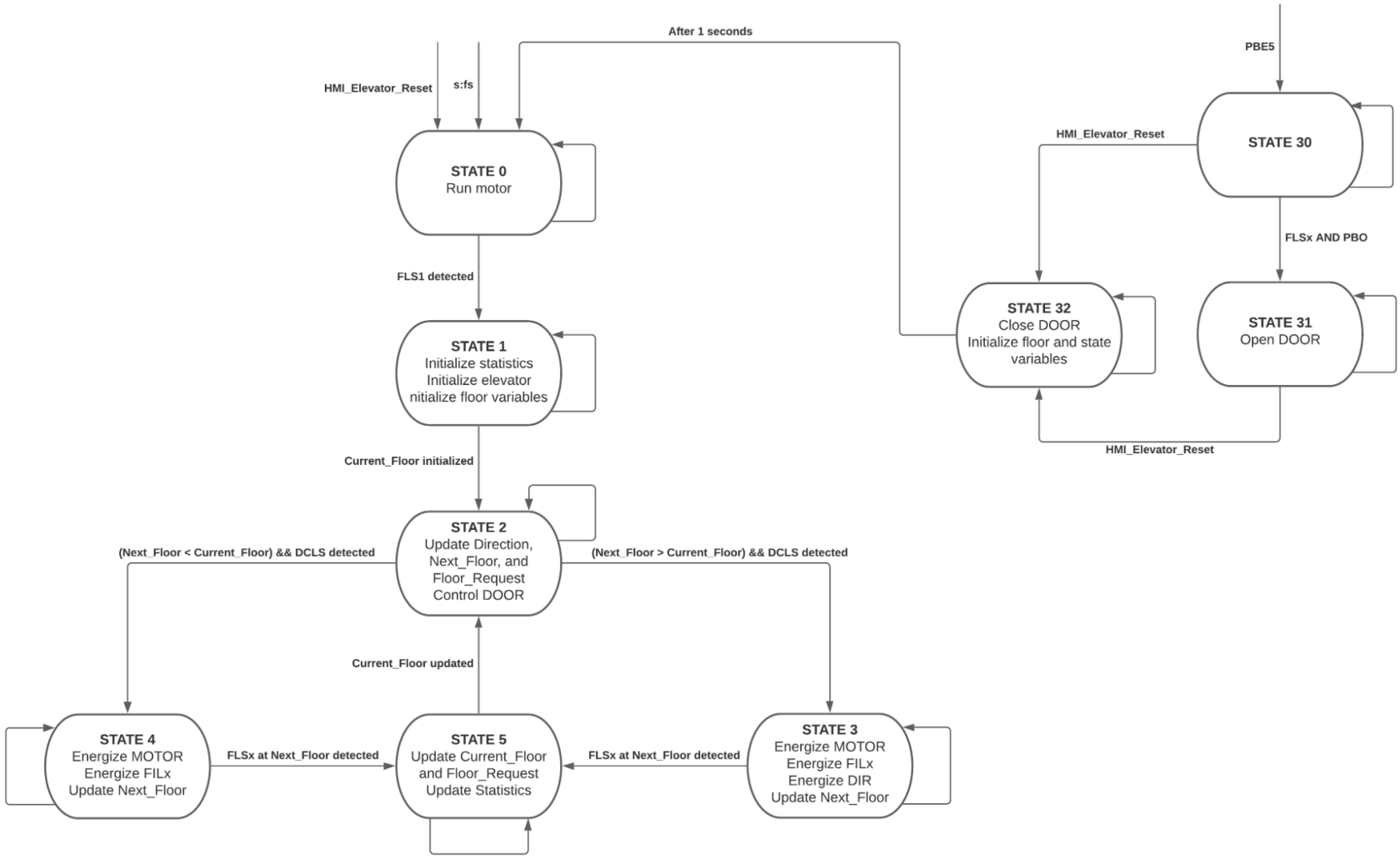

Figure 1 - State diagram for main elevator logic

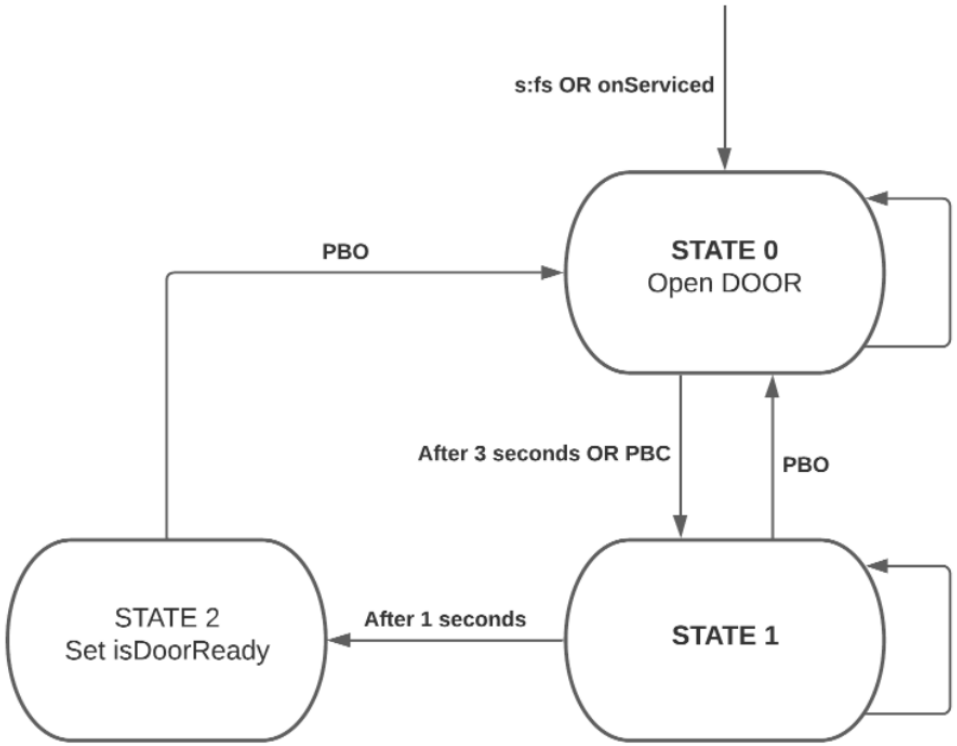

Figure 2 - State diagram for Door_Control

B. Input and Output Mapping Tables

Table 2 - Input mapping table for elevator board

Description	Switch Type	Function	Tag*	Alias
Inside panel pushbutton for 4th floor	N.O. Pushbutton	Floor request to 4th floor. Latches IL4	PB4	Local:2:I.Data.0
Inside panel pushbutton for 3rd floor	N.O. Pushbutton	Floor request to 3rd floor. Latches IL3	PB3	Local:2:I.Data.1
Inside panel pushbutton for 2nd floor	N.O. Pushbutton	Floor request to 2nd floor. Latches IL2	PB2	Local:2:I.Data.2
Inside panel pushbutton for 1st floor	N.O. Pushbutton	Floor request to 1st floor. Latches IL1	PB1	Local:2:I.Data.3
Pushbutton to open door	N.O. Pushbutton	Opens DOOR when main elevator logic is at state #2 or #30	PBO	Local:2:I.Data.4
Pushbutton to close door	N.O. Pushbutton	Closes DOOR when main elevator logic is at state #2	PBC	Local:2:I.Data.5
Pushbutton to emergency stop	N.O. Pushbutton	Emergency stop request	PBE5	Local:2:I.Data.6
Floor pushbutton for 4th floor	N.O. Pushbutton	Floor request to 4th floor. Latches FIL4	FPB4	Local:2:I.Data.8
Floor pushbutton for 3rd floor	N.O. Pushbutton	Floor request to 3rd floor. Latches FIL3	FPB3	Local:2:I.Data.9
Floor pushbutton for 2nd floor	N.O. Pushbutton	Floor request to 2nd floor. Latches FIL2	FPB2	Local:2:I.Data.10
Floor pushbutton for 1st floor	N.O. Pushbutton	Floor request to 1st floor. Latches FIL1	FPB1	Local:2:I.Data.11
Limit switch for 1st floor	N.O. Limit switch	Detect if the elevator cart is on the 1st floor	FLS1	Local:3:I.Data.0
Limit switch for 2nd floor	N.O. Limit switch	Detect if the elevator cart is on the 2nd floor	FLS2	Local:3:I.Data.1
Limit switch for 3rd floor	N.O. Limit switch	Detect if the elevator cart is on the 3rd floor	FLS3	Local:3:I.Data.2
Limit switch for 4th floor	N.O. Limit switch	Detect if the elevator cart is on the 4th floor	FLS4	Local:3:I.Data.3
Limit switch for door close	N.O. Limit switch	Detect if the DOOR is fully closed	DCLS	Local:3:I.Data.8
Limit switch for door open	N.O. Limit switch	Detect if the DOOR is fully open	DOLS	Local:3:I.Data.9

*Because the inputs are mapped to a DINT variable using MVM instruction, the tag listed in this table are not used in the program. It is listed only for reference.

Table 3 - Input mapping table for HMI virtual inputs

Location (Tab)	Switch Type	Function	Tag
Overview	Virtual N.O. Pushbutton	If clicked, override PBO	HMI_Elevator_HPBO
Overview	Virtual N.O. Pushbutton	If clicked, override PBC	HMI_Elevator_HPBC
Overview	Virtual N.O. Pushbutton	If clicked, override PBE5	HMI_Elevator_HPBE5
Overview	Virtual N.O. Pushbutton	If clicked, override FPB4	HMI_Elevator_HPB4
Overview	Virtual N.O. Pushbutton	If clicked, override FPB3	HMI_Elevator_HPB3
Overview	Virtual N.O. Pushbutton	If clicked, override FPB2	HMI_Elevator_HPB2
Overview	Virtual N.O. Pushbutton	If clicked, override FPB1	HMI_Elevator_HPB1
Administration	Virtual N.O. Pushbutton	Restart elevator	HMI_Elevator_Reset
Diagnostics	Virtual N.O. Pushbutton	Reset Floor_Calls and Average_Service_Time for 1st floor	HMI_Elevator_ResetFloor1
Diagnostics	Virtual N.O. Pushbutton	Reset Floor_Calls and Average_Service_Time for 2nd floor	HMI_Elevator_ResetFloor2
Diagnostics	Virtual N.O. Pushbutton	Reset Floor_Calls and Average_Service_Time for 3rd floor	HMI_Elevator_ResetFloor3
Diagnostics	Virtual N.O. Pushbutton	Reset Floor_Calls and Average_Service_Time for 4th floor	HMI_Elevator_ResetFloor4

Table 4 - Output mapping table for elevator board

Description	Function	Tag*	Alias
Indicator lights for 4th floor inside panel pushbutton	Indicate request is being serviced, after PB4 is pressed	IL4	Local:4:O.Data.0
Indicator lights for 3rd floor inside panel pushbutton	Indicate request is being serviced, after PB3 is pressed	IL3	Local:4:O.Data.1
Indicator lights for 2nd floor inside panel pushbutton	Indicate request is being serviced, after PB2 is pressed	IL2	Local:4:O.Data.2
Indicator lights for 1st floor inside panel pushbutton	Indicate request is being serviced, after PB1 is pressed	IL1	Local:4:O.Data.3
Indicator light for emergency stop	Indicate elevator is in emergency state	ILE5	Local:4:O.Data.6
Indicator lights for 4th floor pushbutton	Indicate request is being serviced, after FPB4 is pressed	FIL4	Local:4:O.Data.8
Indicator lights for 3rd floor pushbutton	Indicate request is being serviced, after FPB3 is pressed	FIL3	Local:4:O.Data.9
Indicator lights for 2nd floor pushbutton	Indicate request is being serviced, after FPB2 is pressed	FIL2	Local:4:O.Data.10
Indicator lights for 1st floor pushbutton	Indicate request is being serviced, after FPB1 is pressed	FIL1	Local:4:O.Data.11
Indicator light for maintenance needed	Indicate elevator requires a maintenance	ILTS0	Local:4:O.Data.14
Elevator motor	Control the elevator cart	MOTOR	Local:5:O.Data.0
Door solenoid	Control the door of the elevator cart	DOOR	Local:5:O.Data.2
Direction relay	Control the direction of the elevator motor	DIR	Local:5:O.Data.4

*Because the indicator lights are mapped from a DINT variable using MVM instruction, the tag for indicator lights listed in this table are not used in the program. It is listed only for reference.

C. Ladder Logic Programs

MainRoutine

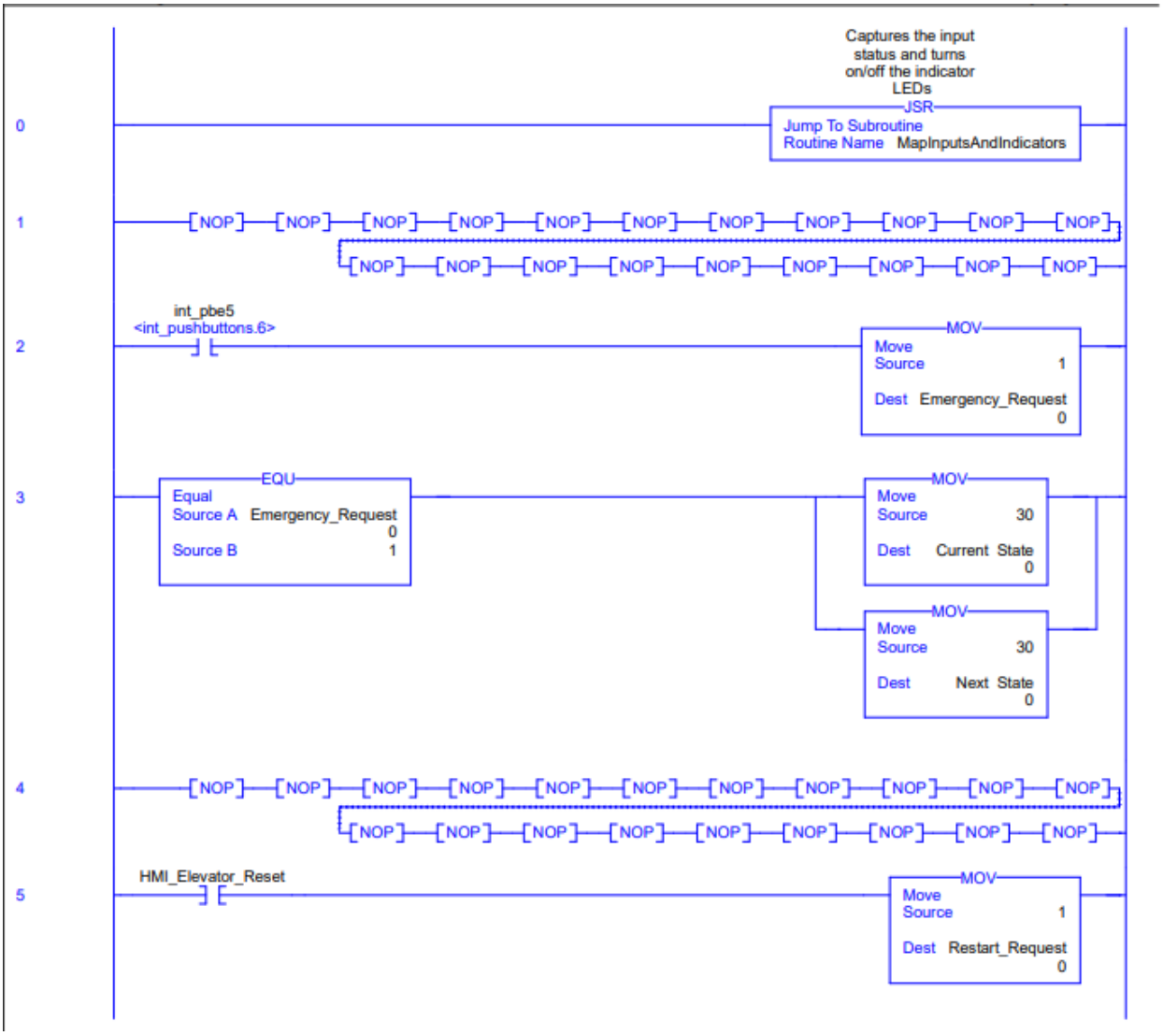

Figure 3 - Ladder Logic program of MainRoutine (page 1 of 10)

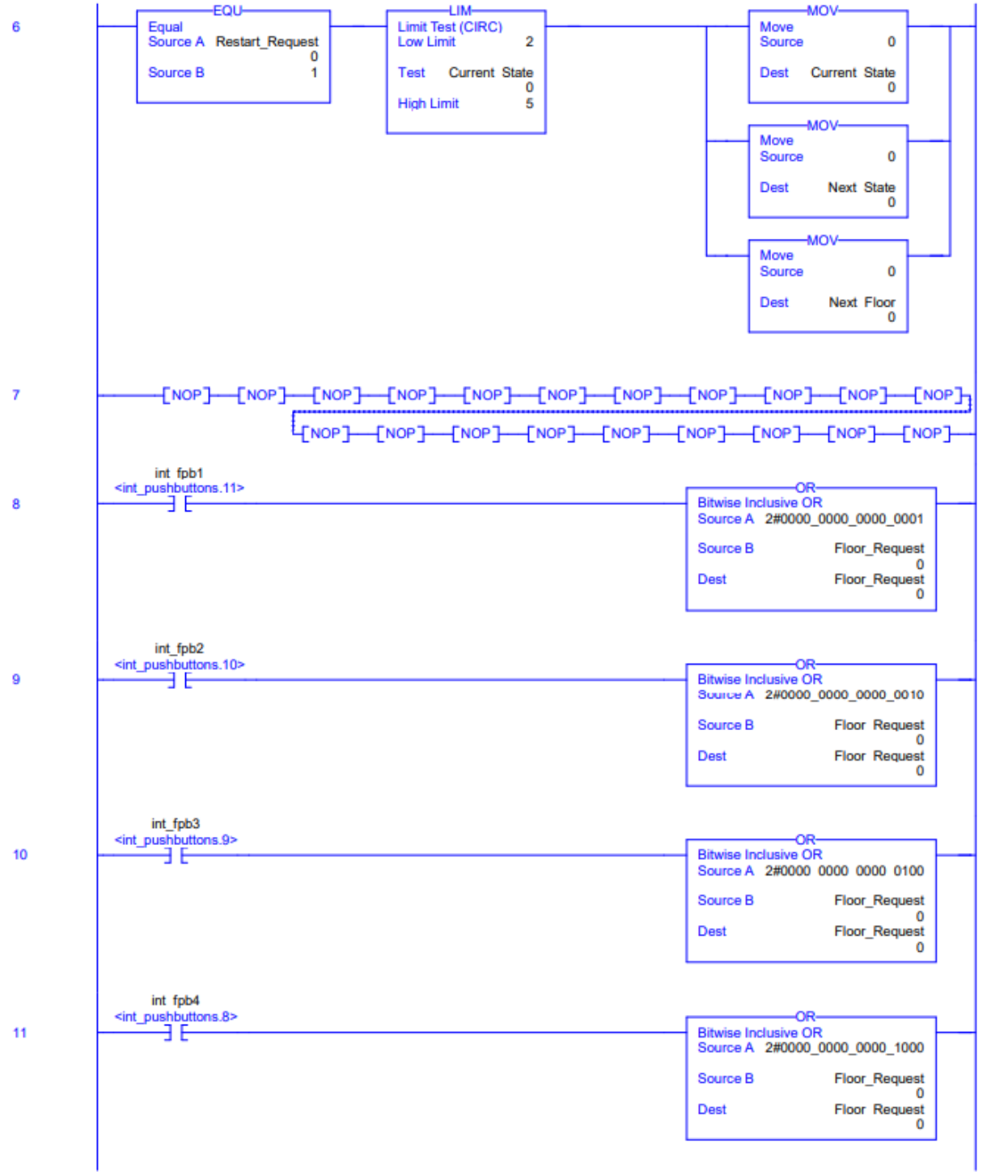

Figure 4 - Ladder Logic program of MainRoutine (page 2 of 10)

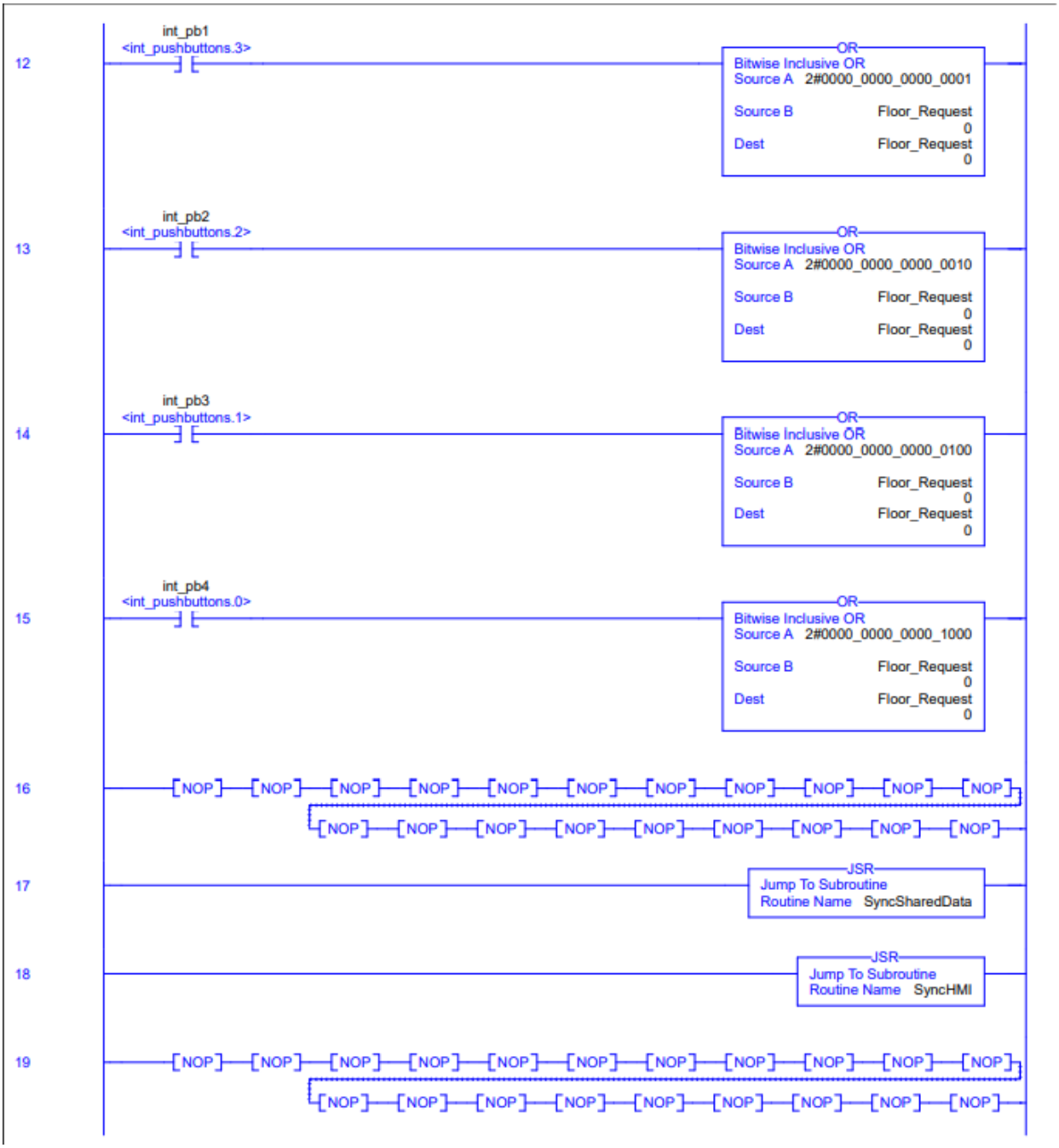

Figure 5 - Ladder Logic program of MainRoutine (page 3 of 10)

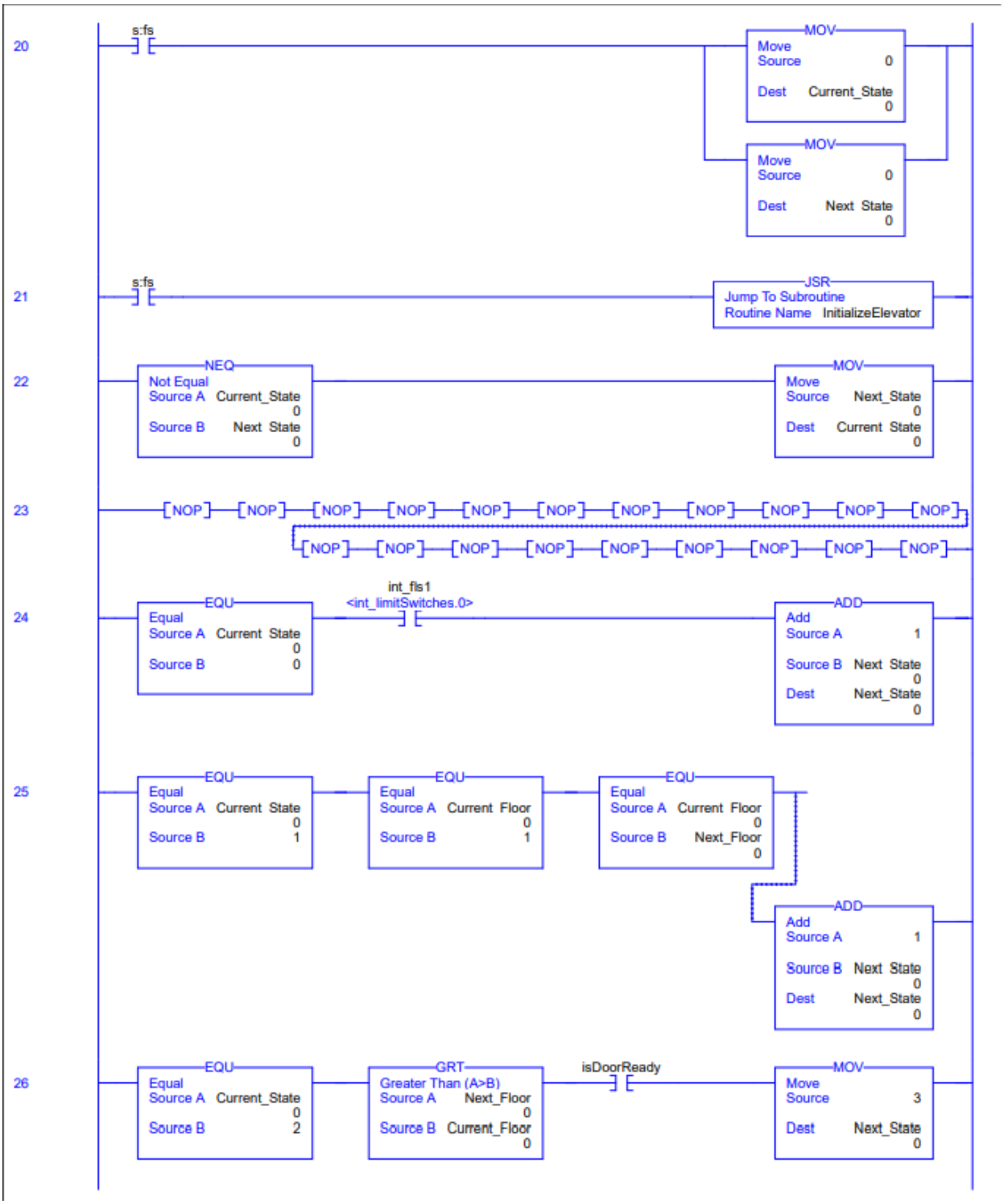

Figure 6 - Ladder Logic program of MainRoutine (page 4 of 10)

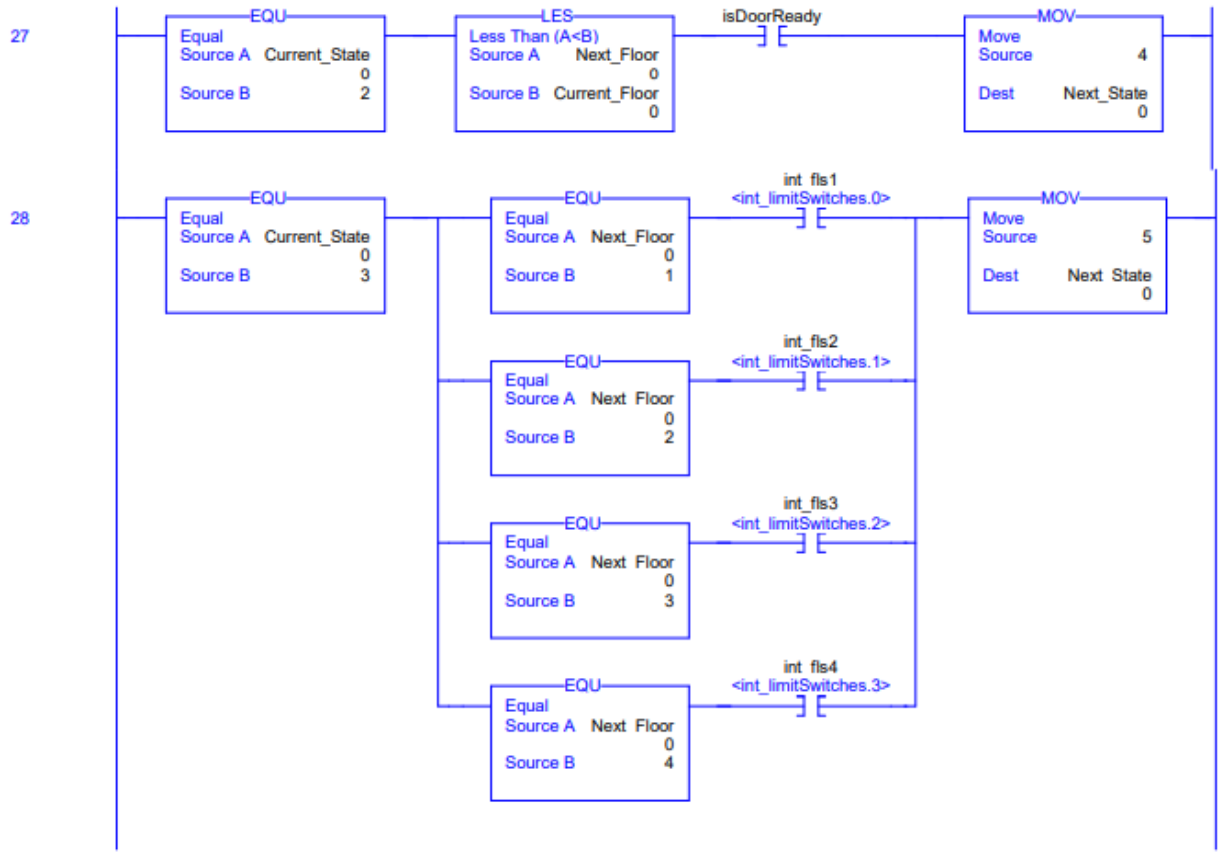

Figure 7 - Ladder Logic program of MainRoutine (page 5 of 10)

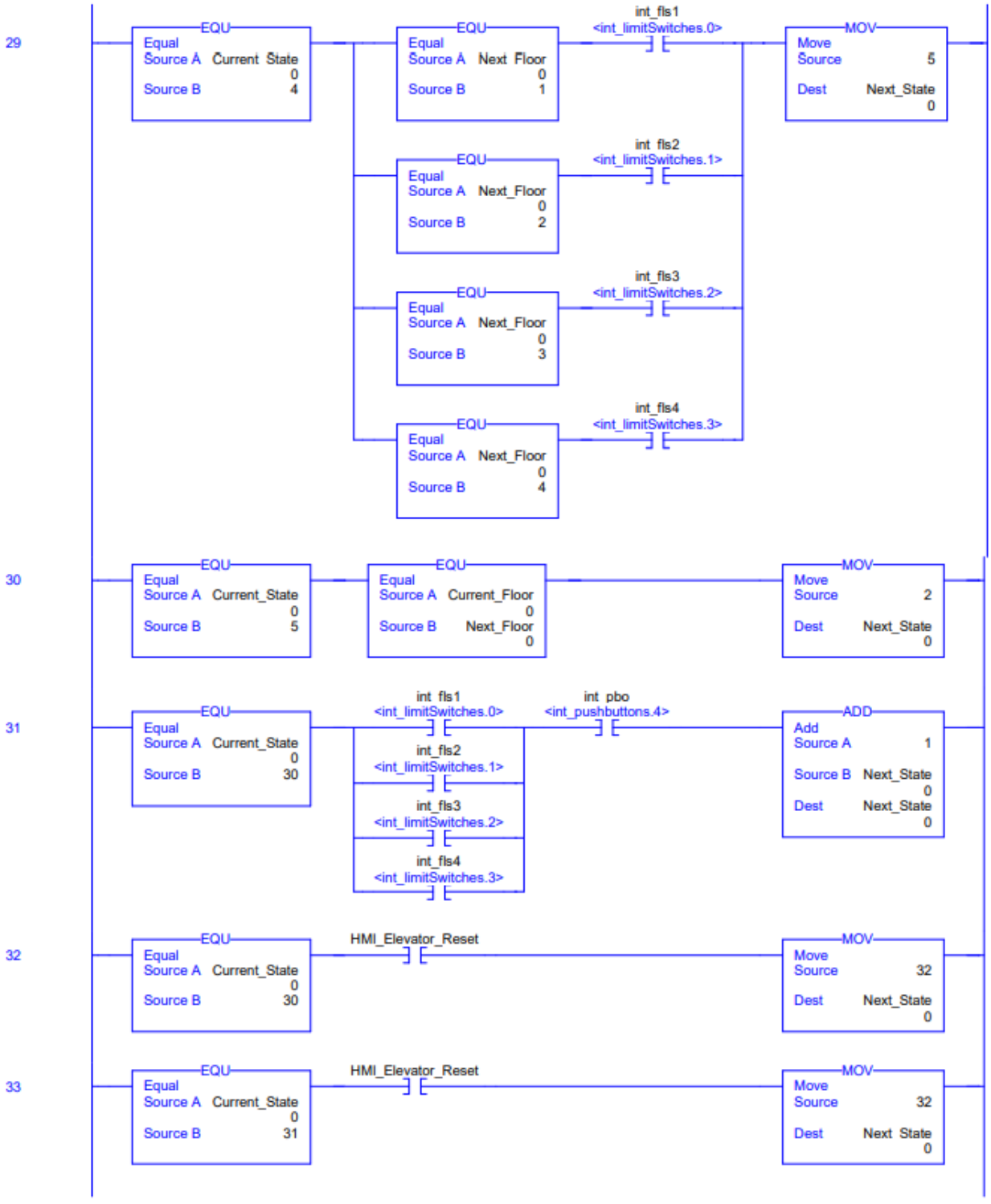

Figure 8 - Ladder Logic program of MainRoutine (page 6 of 10)

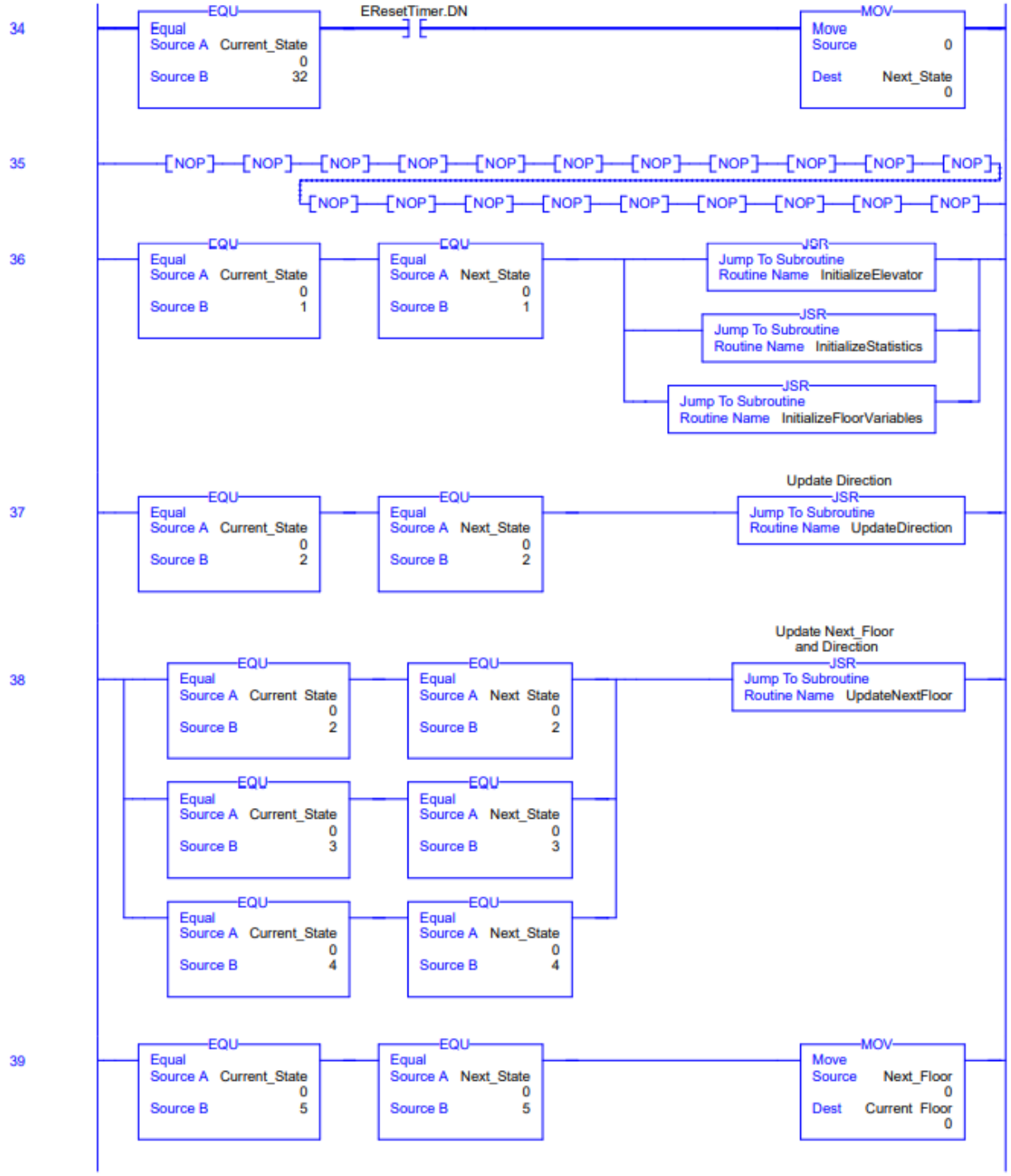

Figure 9 - Ladder Logic program of MainRoutine (page 7 of 10)

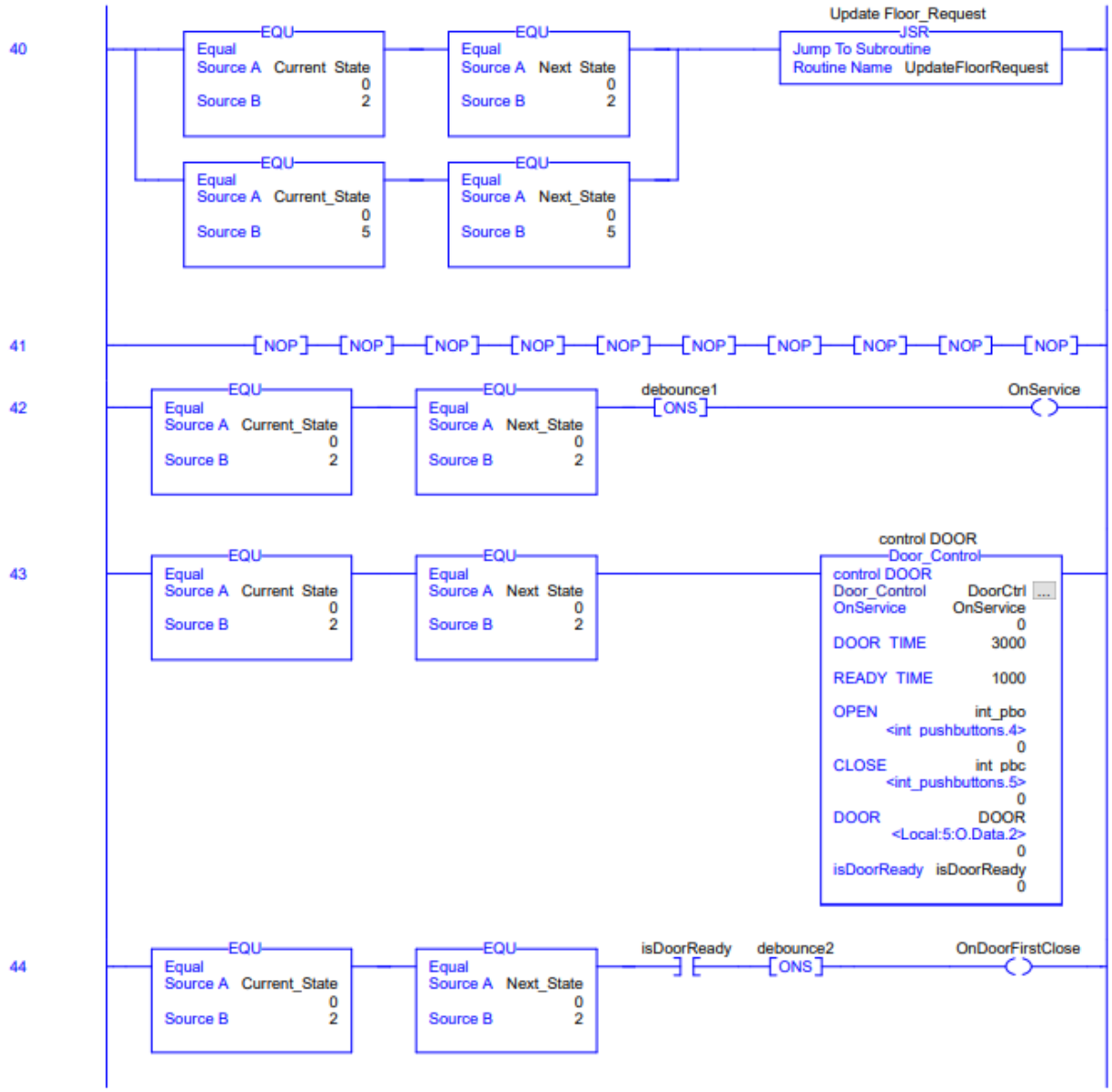

Figure 10 - Ladder Logic program of MainRoutine (page 8 of 10)

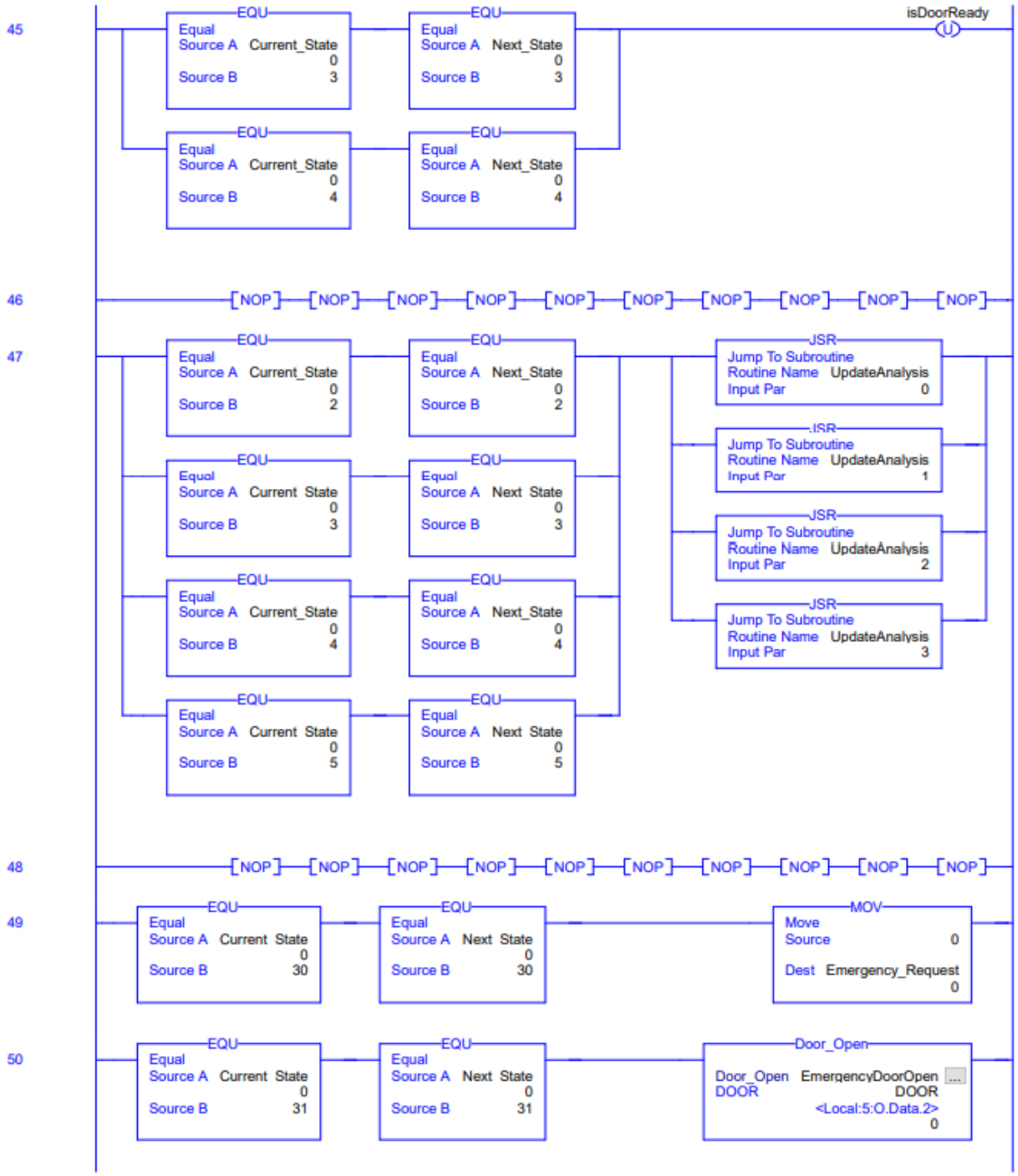

Figure 11 - Ladder Logic program of MainRoutine (page 9 of 10)

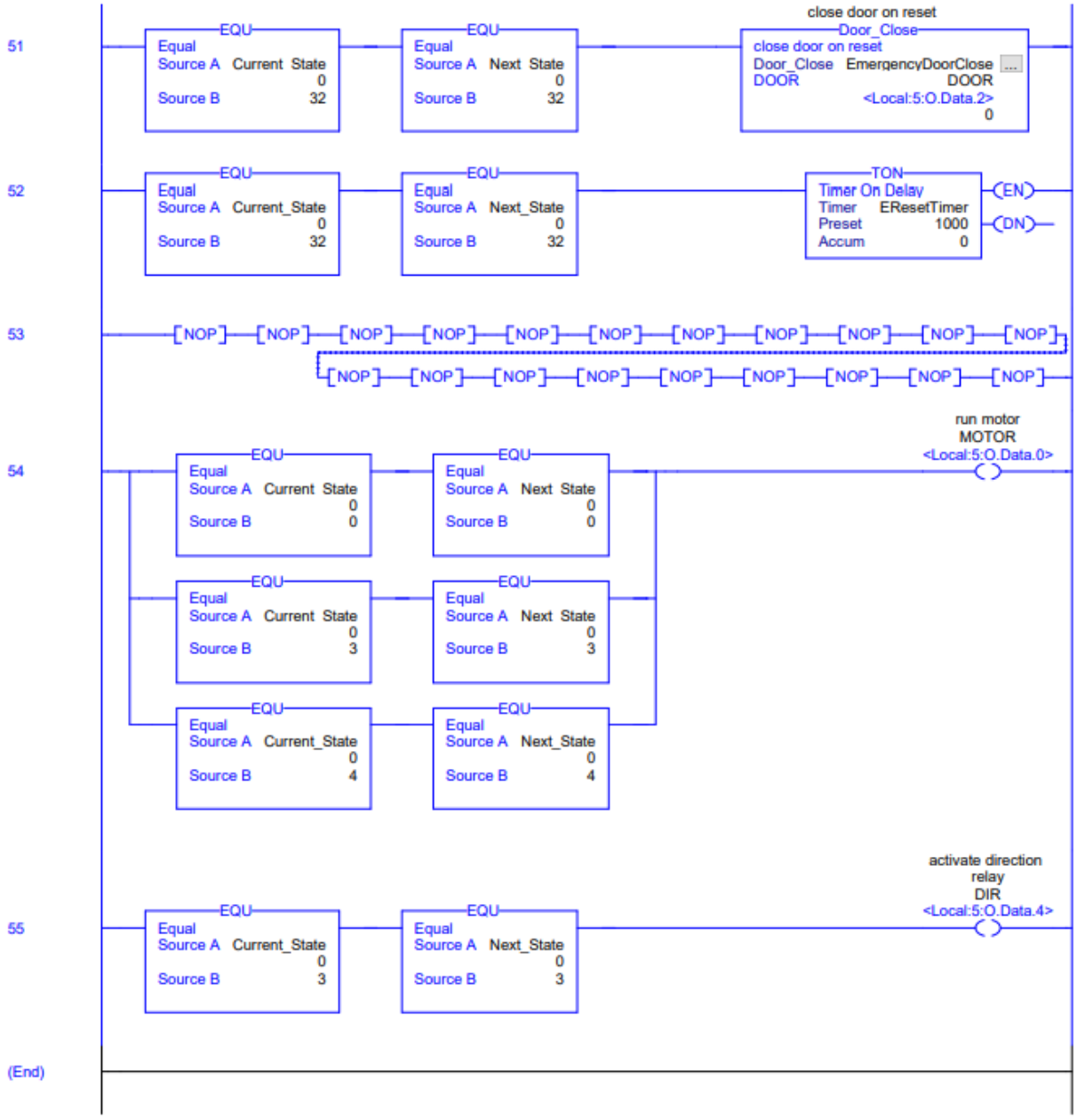

Figure 12 - Ladder Logic program of MainRoutine (page 10 of 10)

InitializeElevator

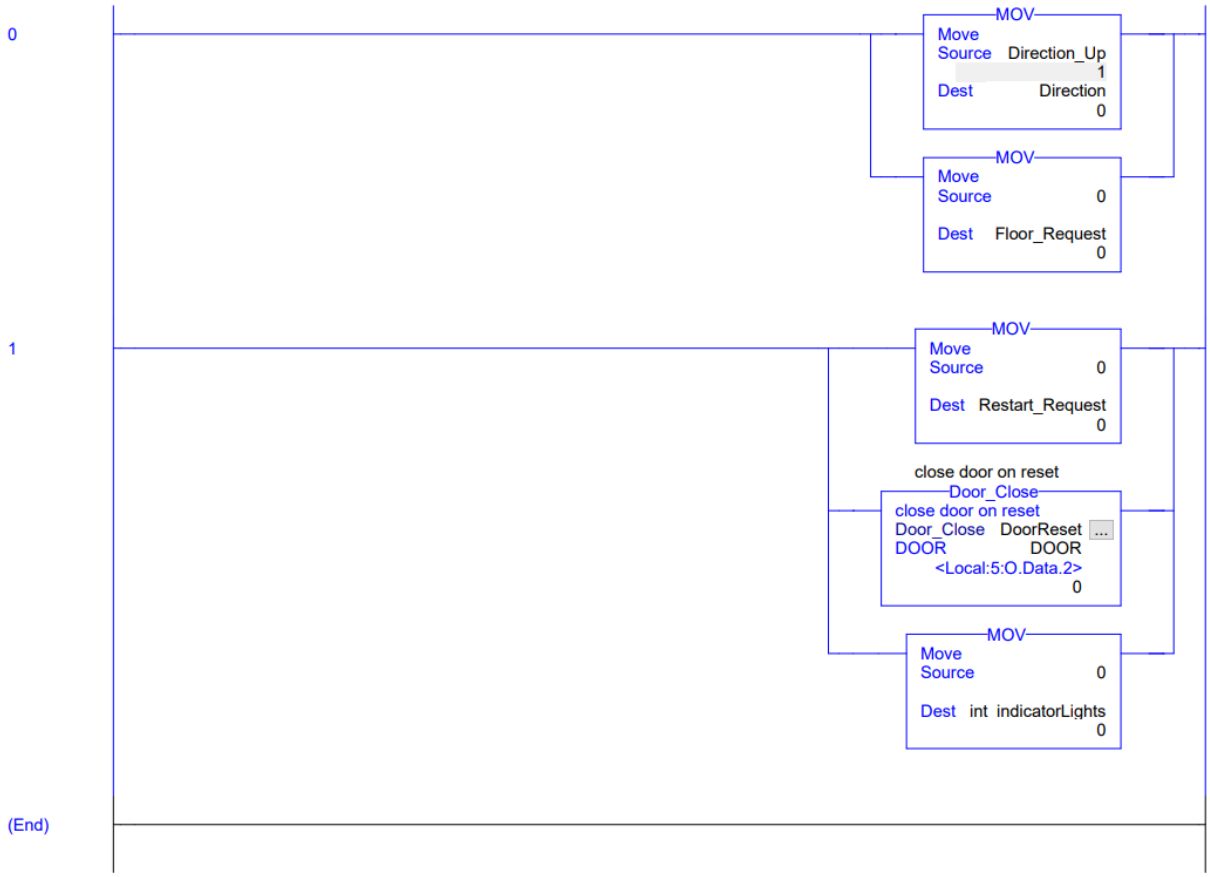

Figure 13 - Ladder Logic program of InitializeElevator

InitializeFloorVariables

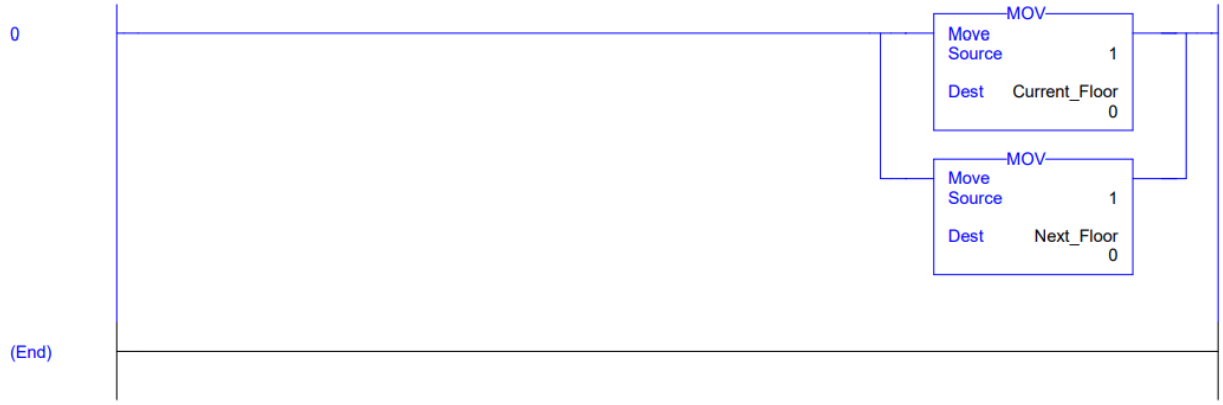

Figure 14 - Ladder Logic program of InitializeFloorVariables

InitializeStatistics

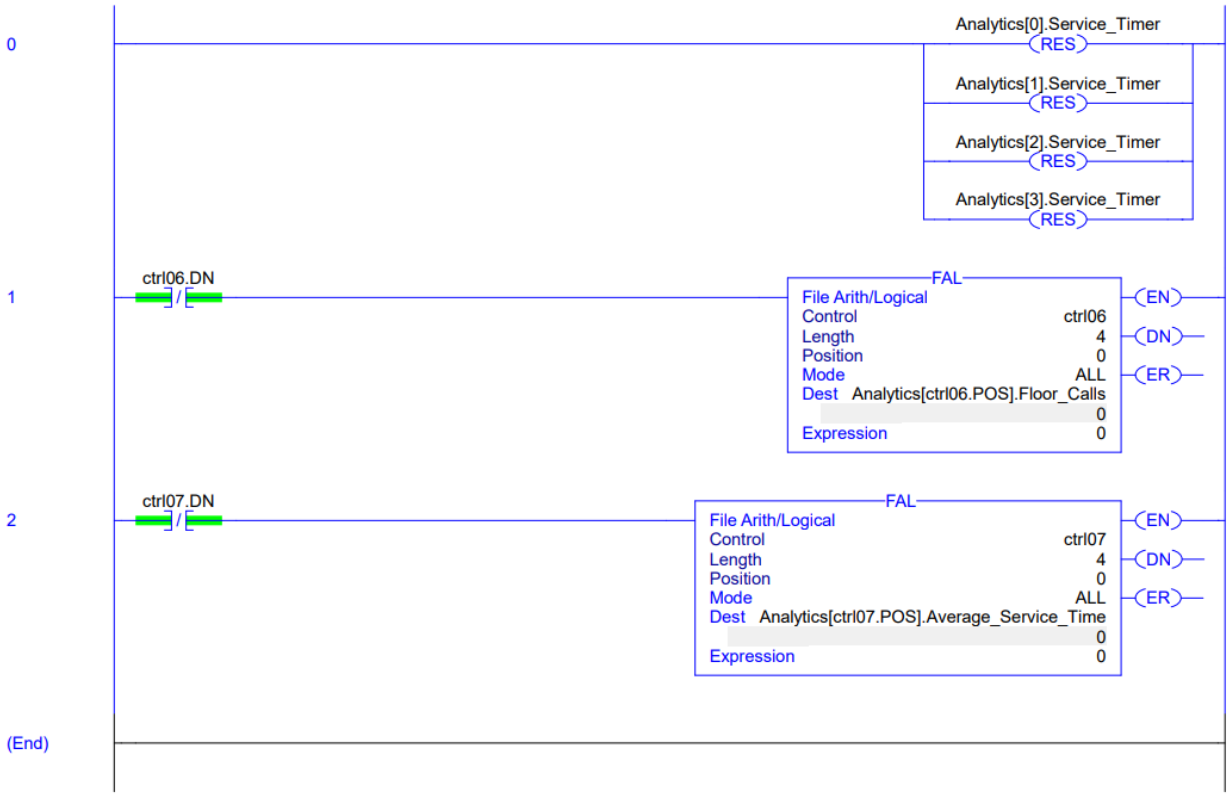

Figure 15 - Ladder Logic program of InitializeStatistics

MapInputsAndIndicators

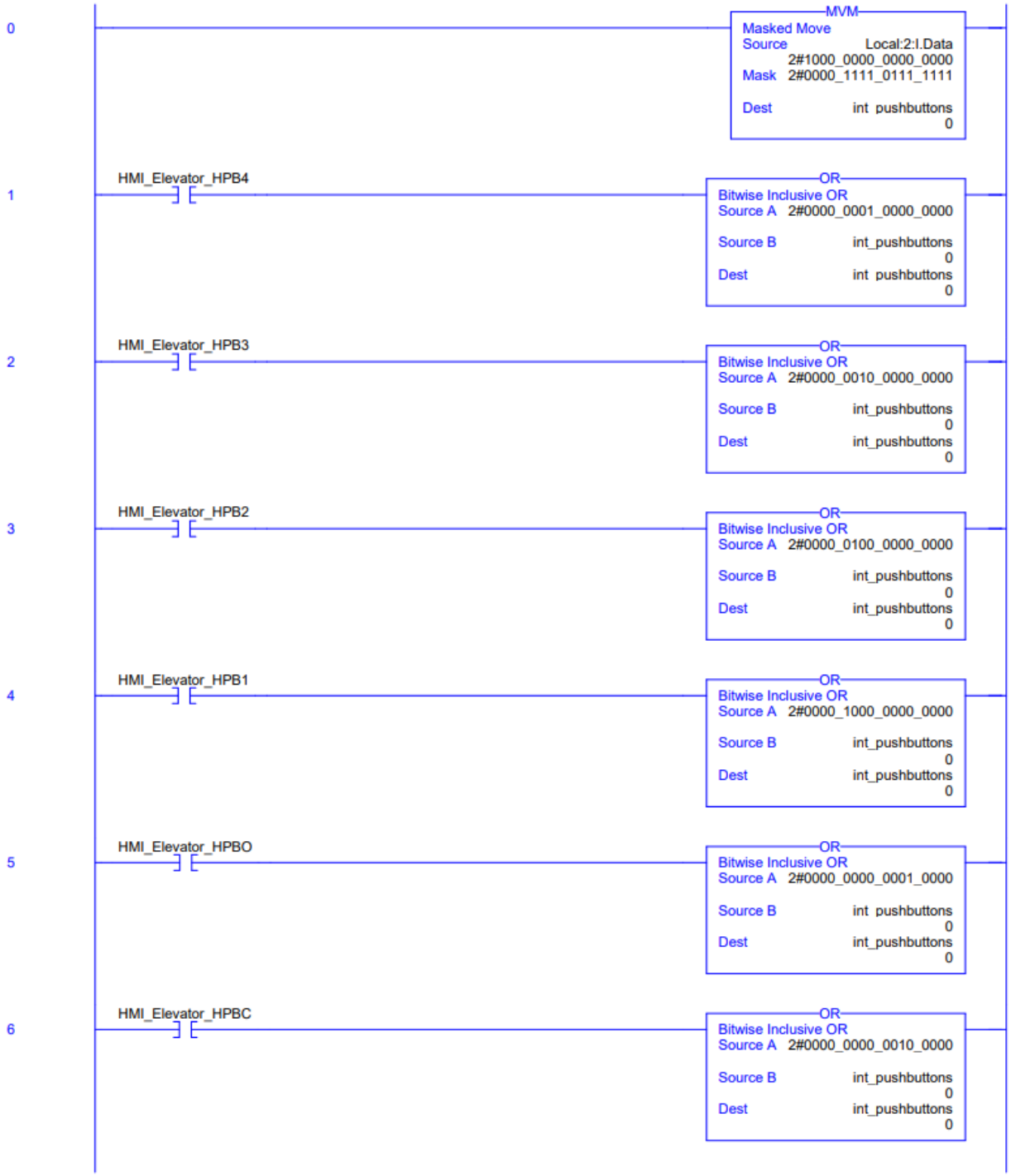

Figure 16 - Ladder Logic program of MapInputsAndIndicators (page 1 of 3)

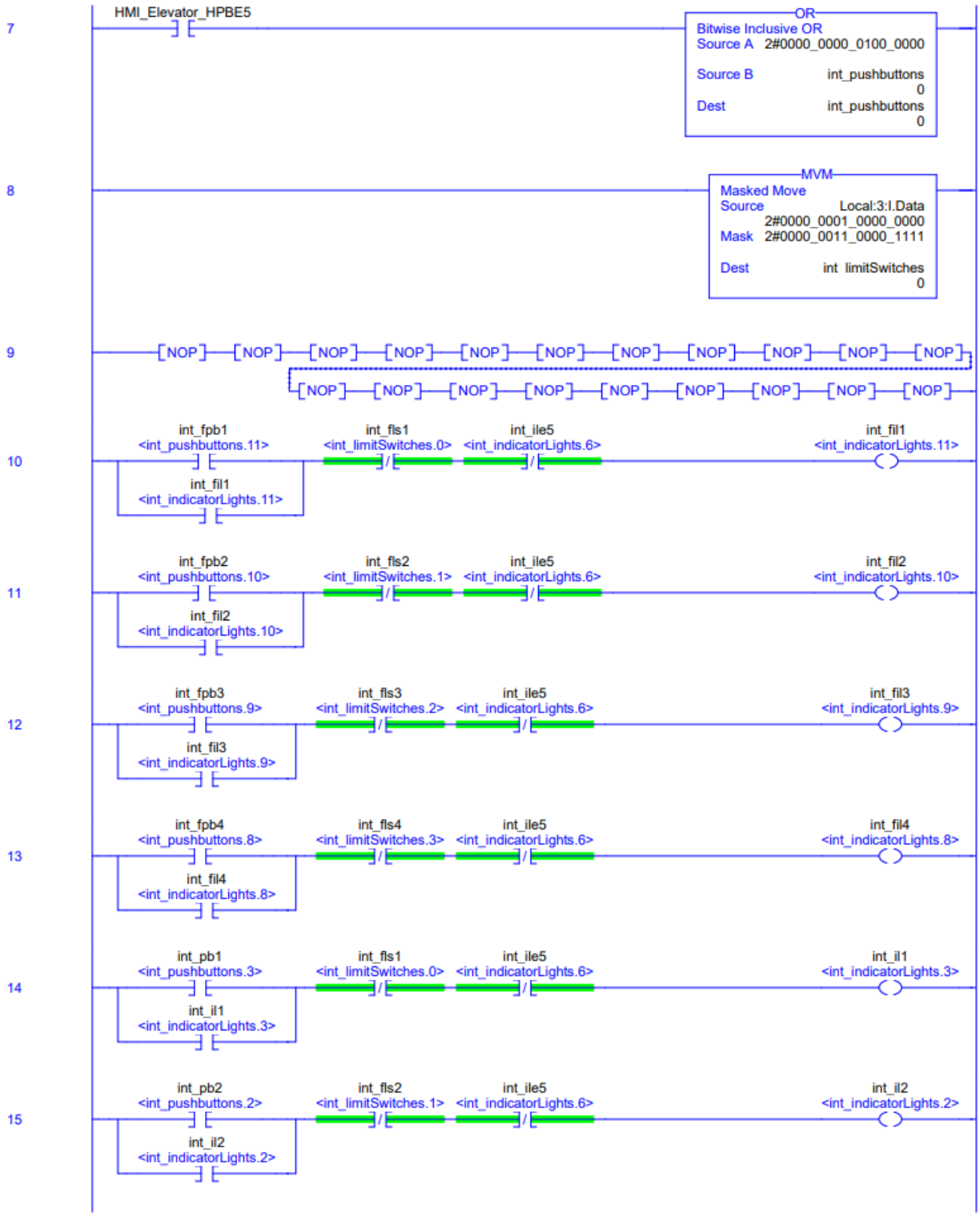

Figure 17 - Ladder Logic program of MapInputsAndIndicators (page 2 of 3)

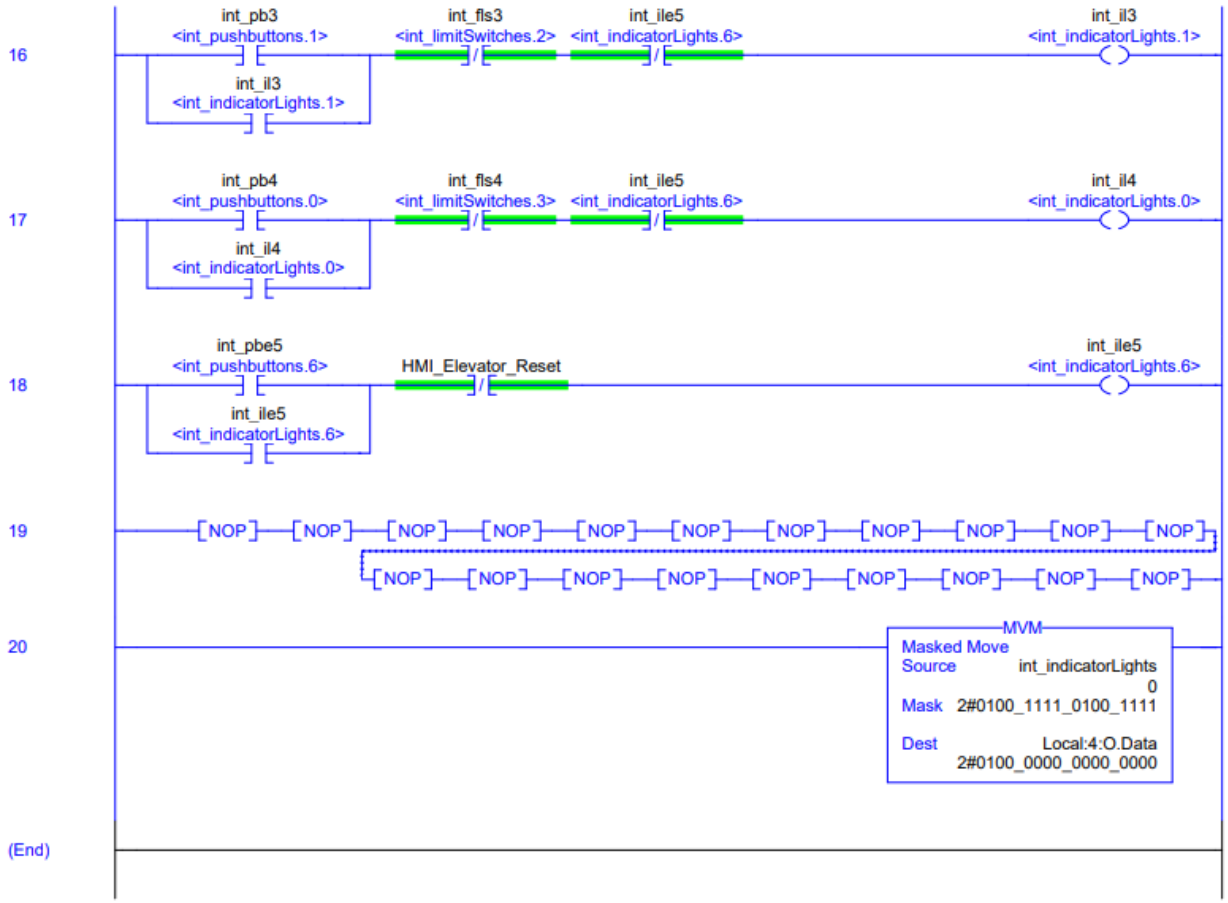

Figure 18 - Ladder Logic program of MapInputsAndIndicators (page 3 of 3)

UpdateFloorRequest

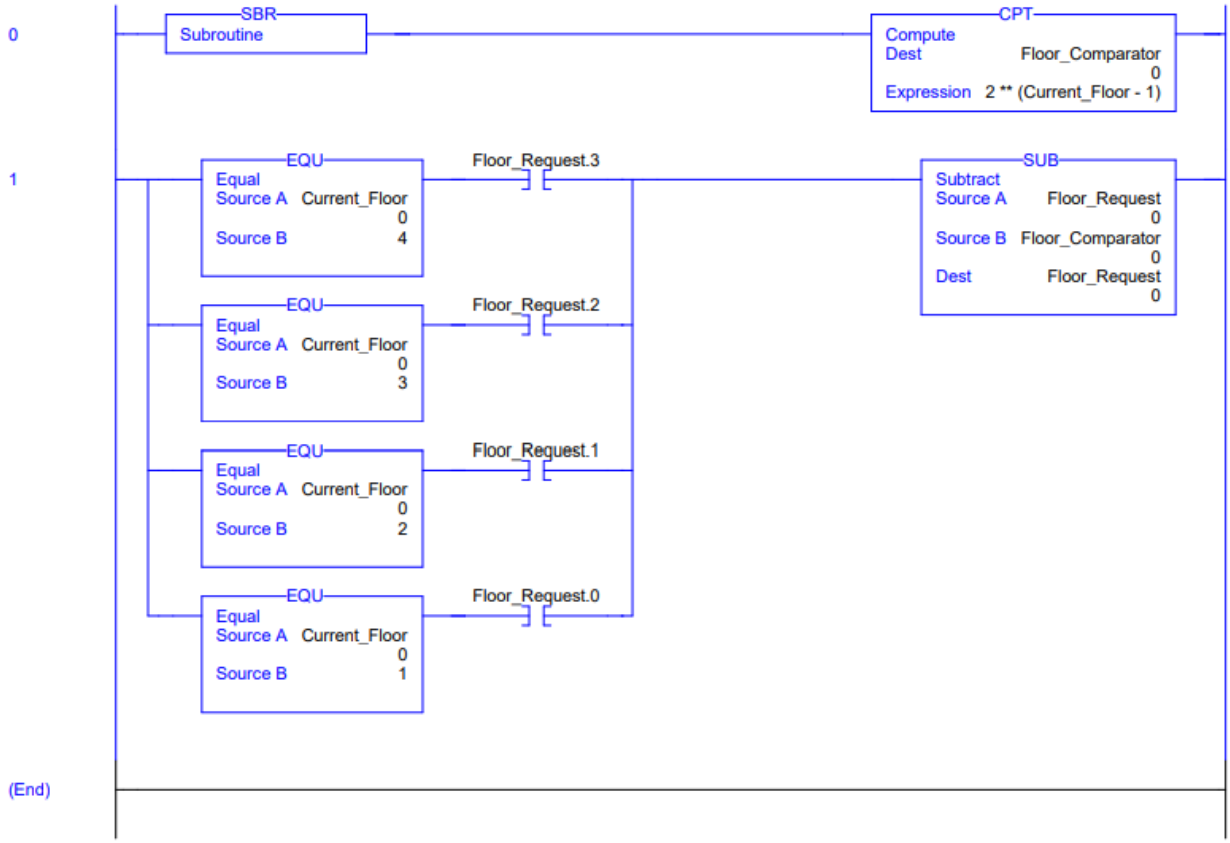

Figure 19 - Ladder Logic program of UpdateFloorRequest

UpdateDirection

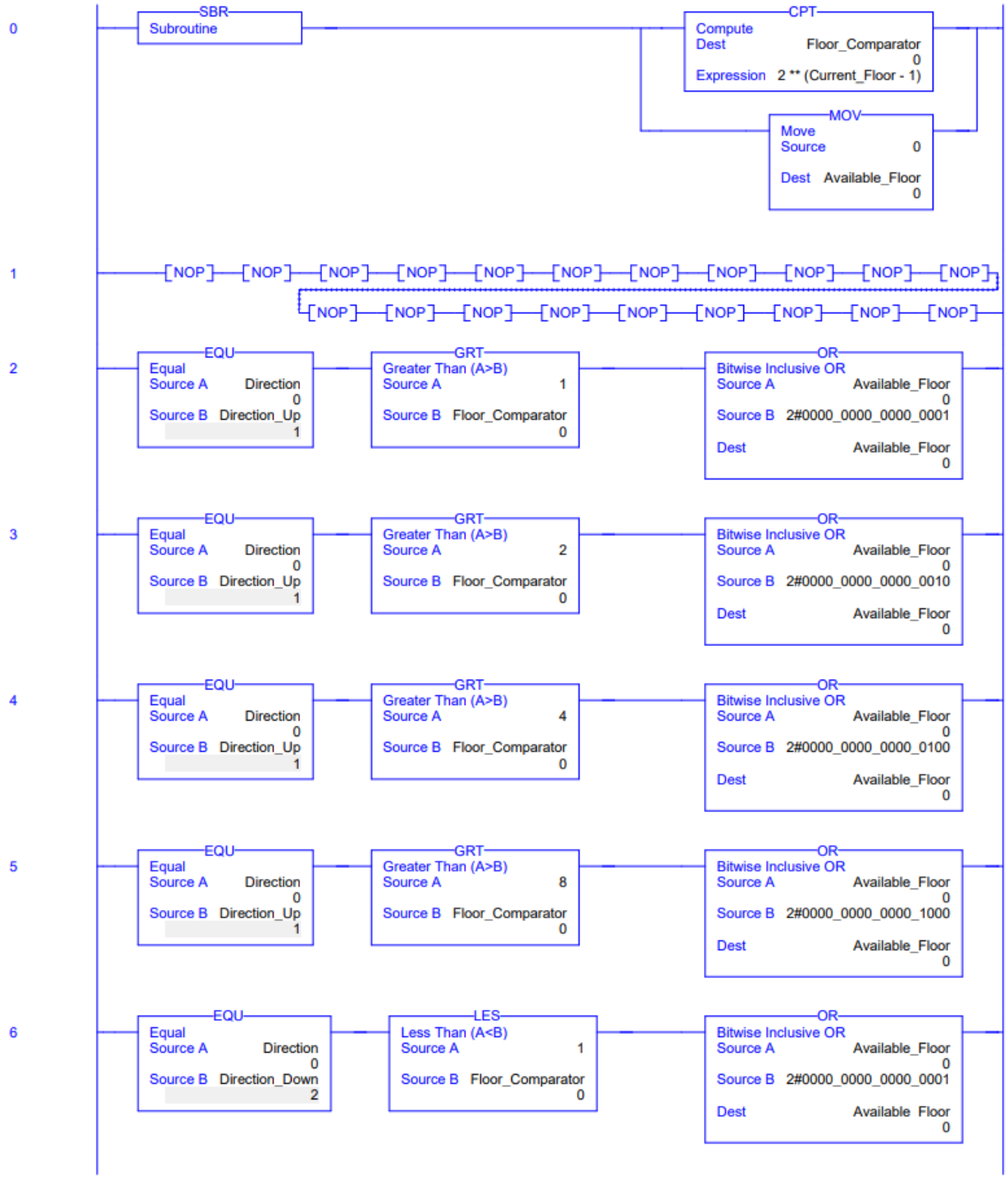

Figure 20 - Ladder Logic program of UpdateDirection (page 1 of 2)

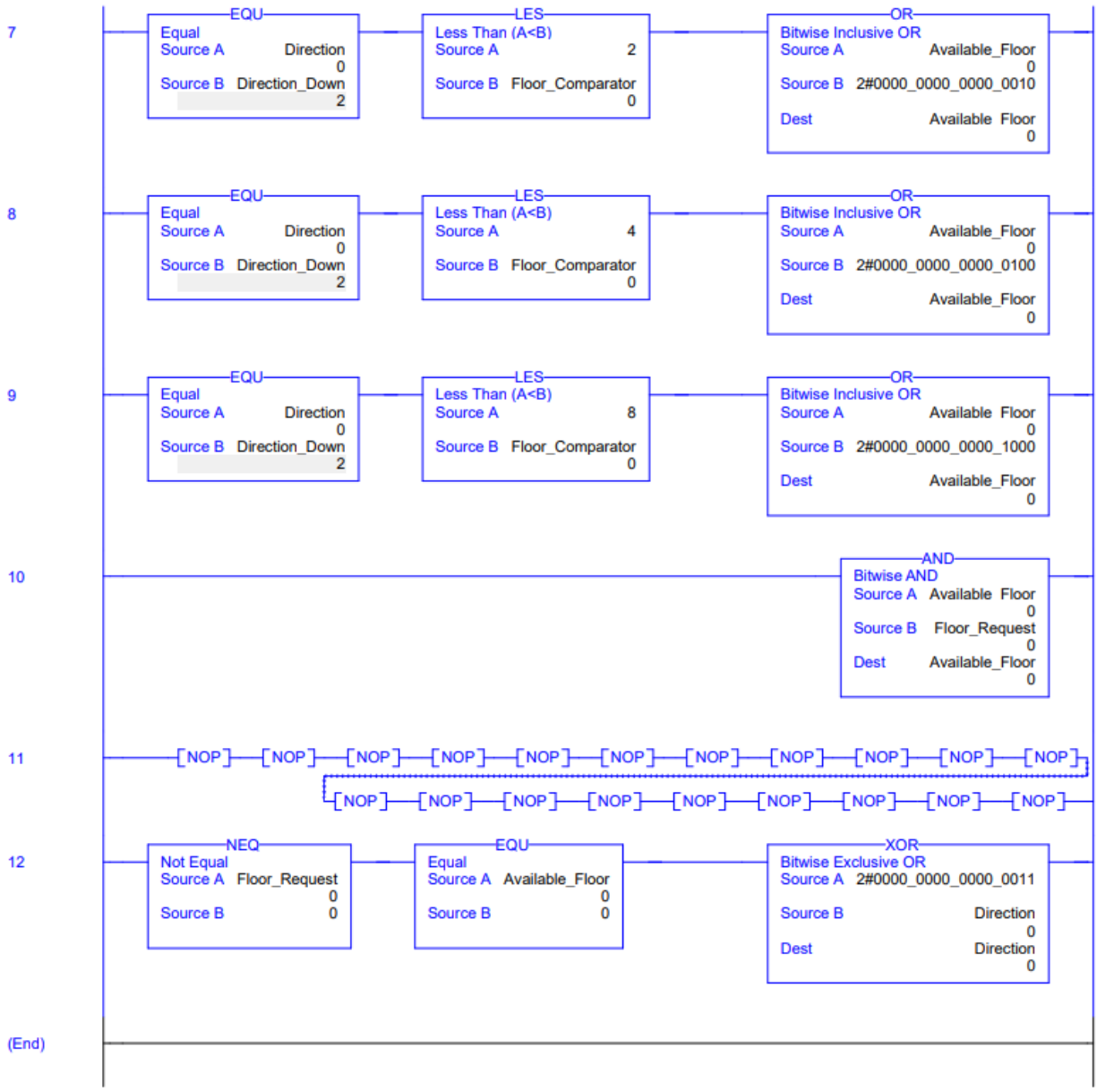

Figure 21 - Ladder Logic program of UpdateDirection (page 2 of 2)

UpdateNextFloor

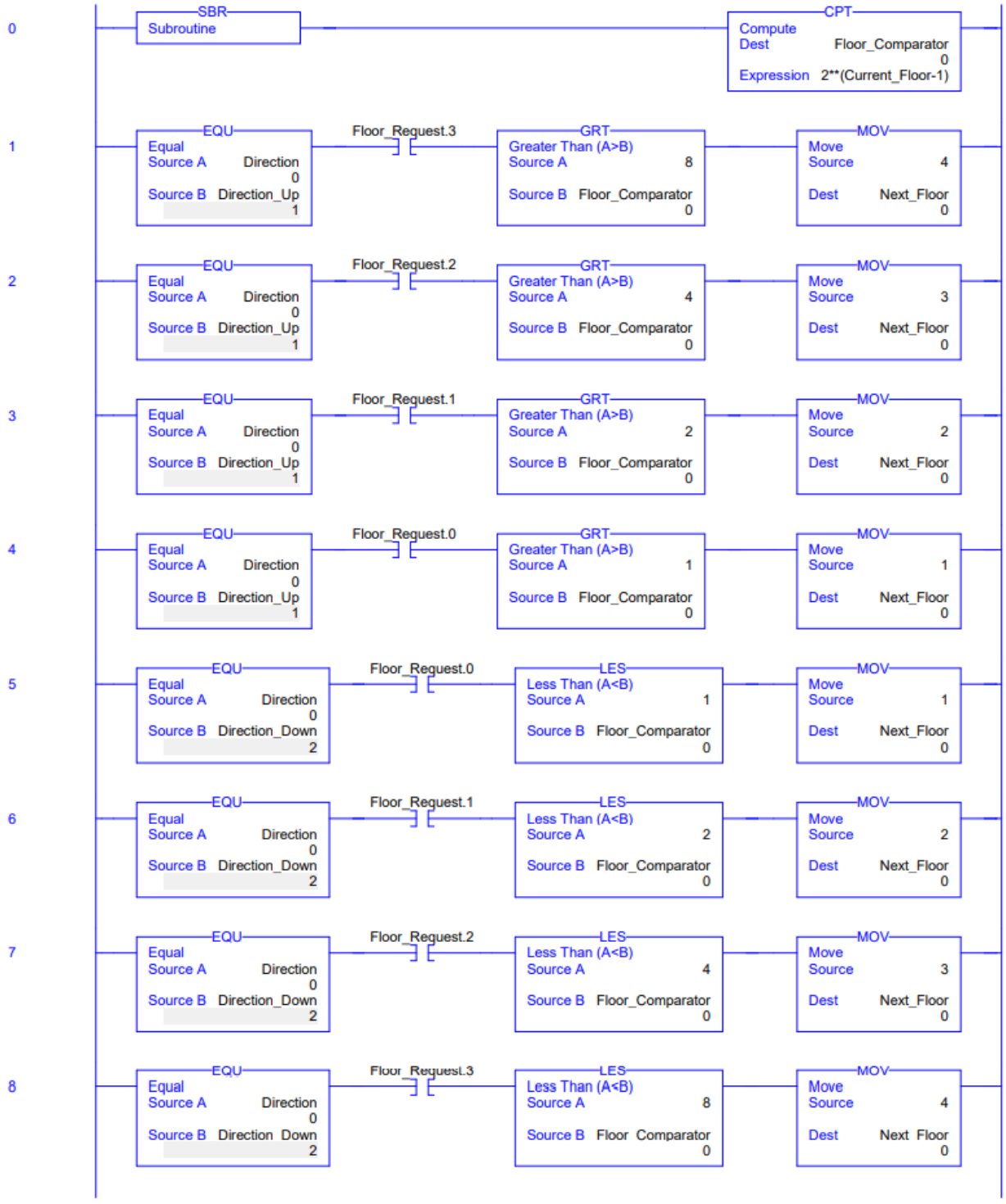

Figure 22 - Ladder Logic program of UpdateNextFloor (page 1 of 2)

Figure 23 - Ladder Logic program of UpdateNextFloor (page 2 of 2)

Door_Control

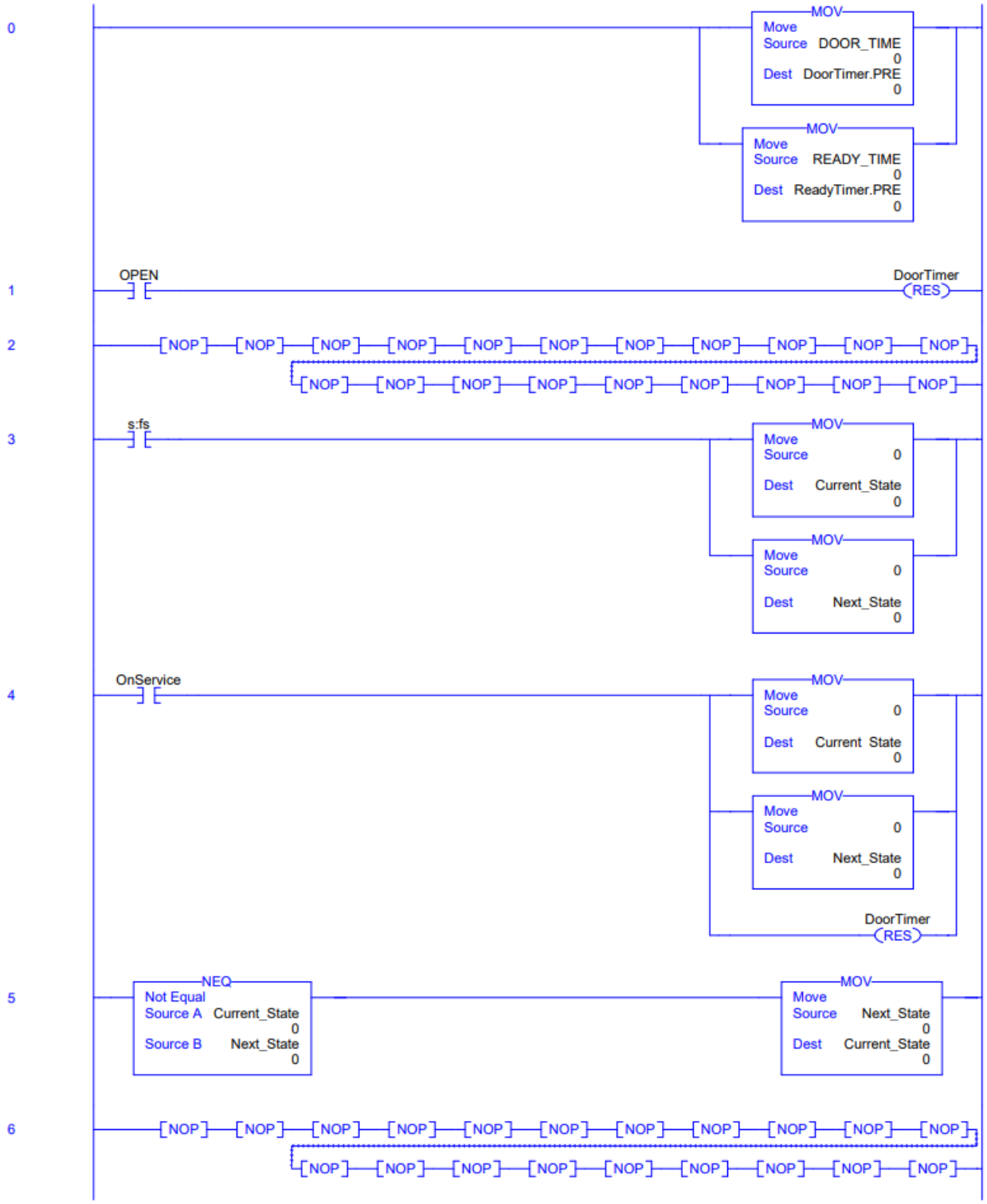

Figure 24 - Ladder Logic program of Door_Control (page 1 of 3)

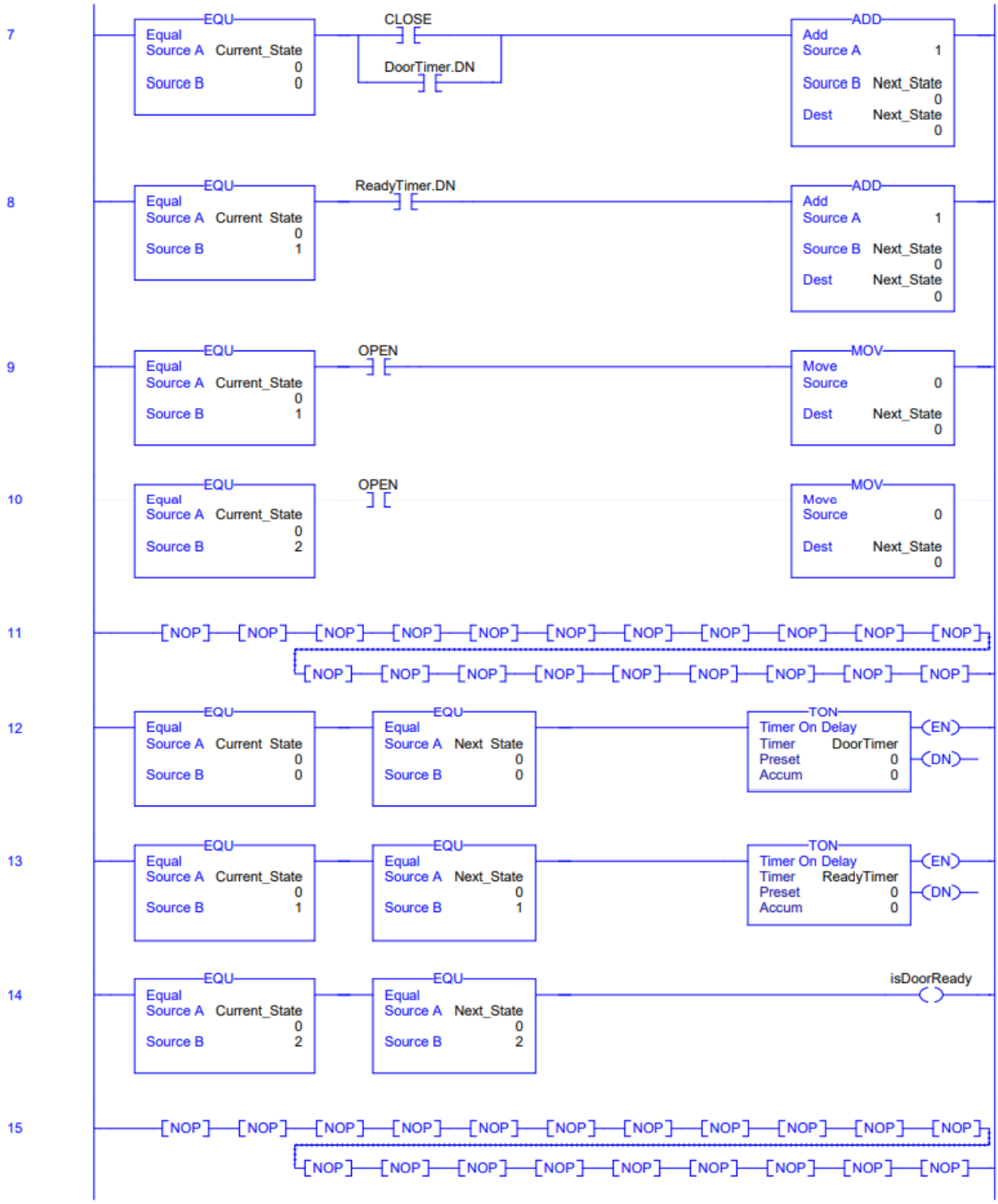

Figure 25 - Ladder Logic program of Door_Control (page 2 of 3)

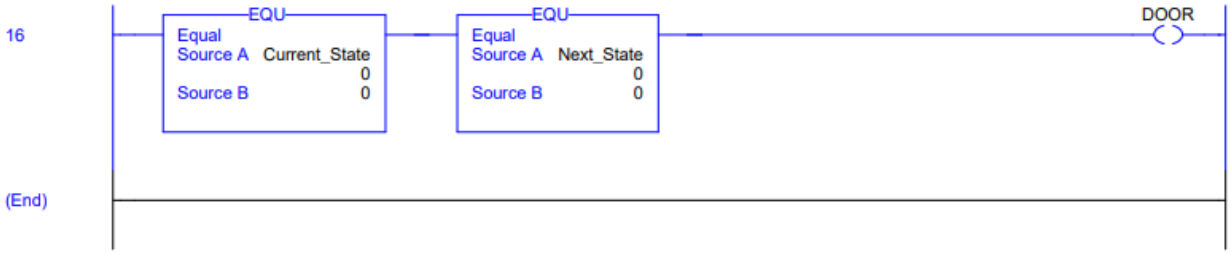

Figure 26 - Ladder Logic program of Door_Control (page 3 of 3)

UpdateAnalysis

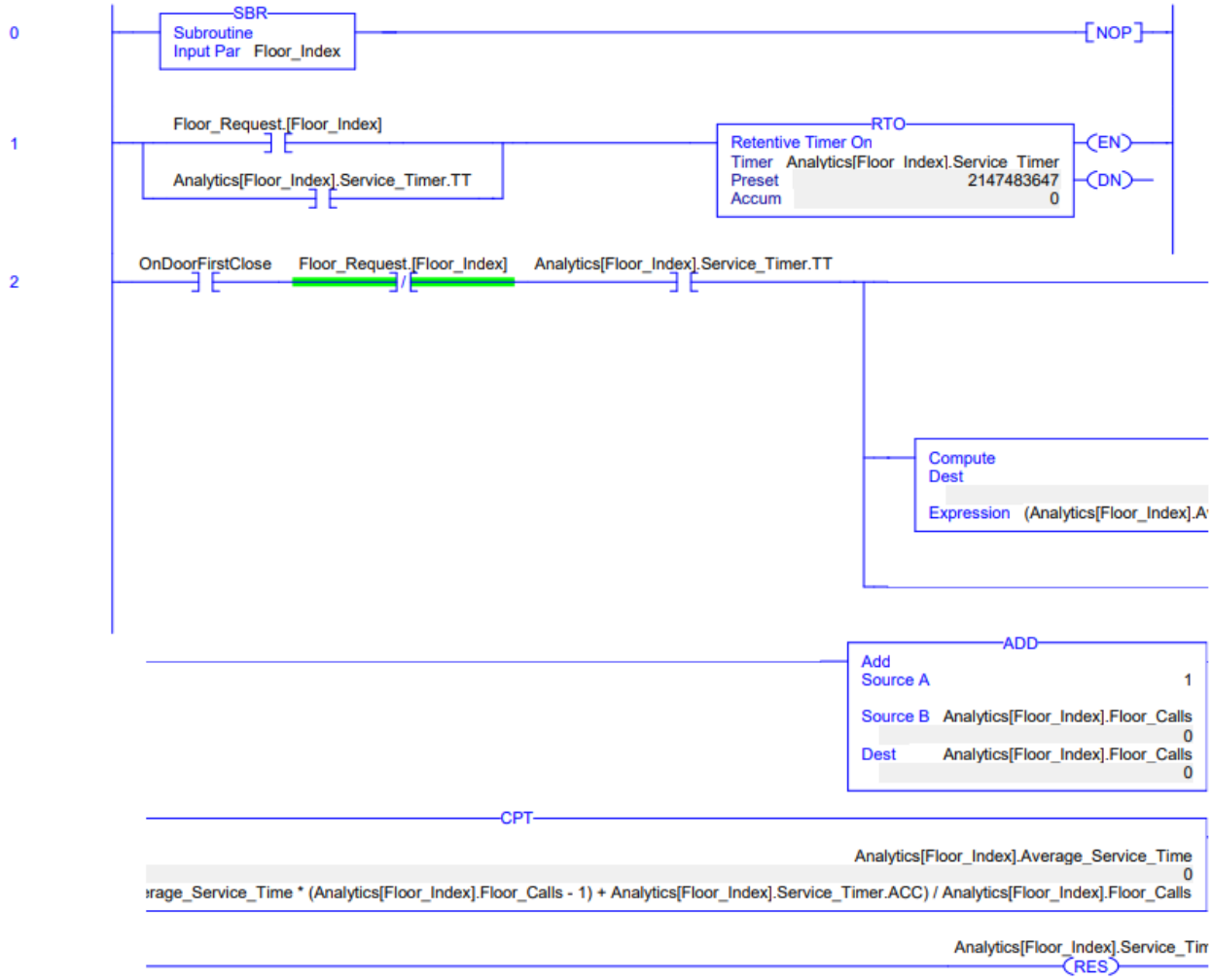

Figure 27 - Ladder Logic program of UpdateAnalysis (page 1 of 2)

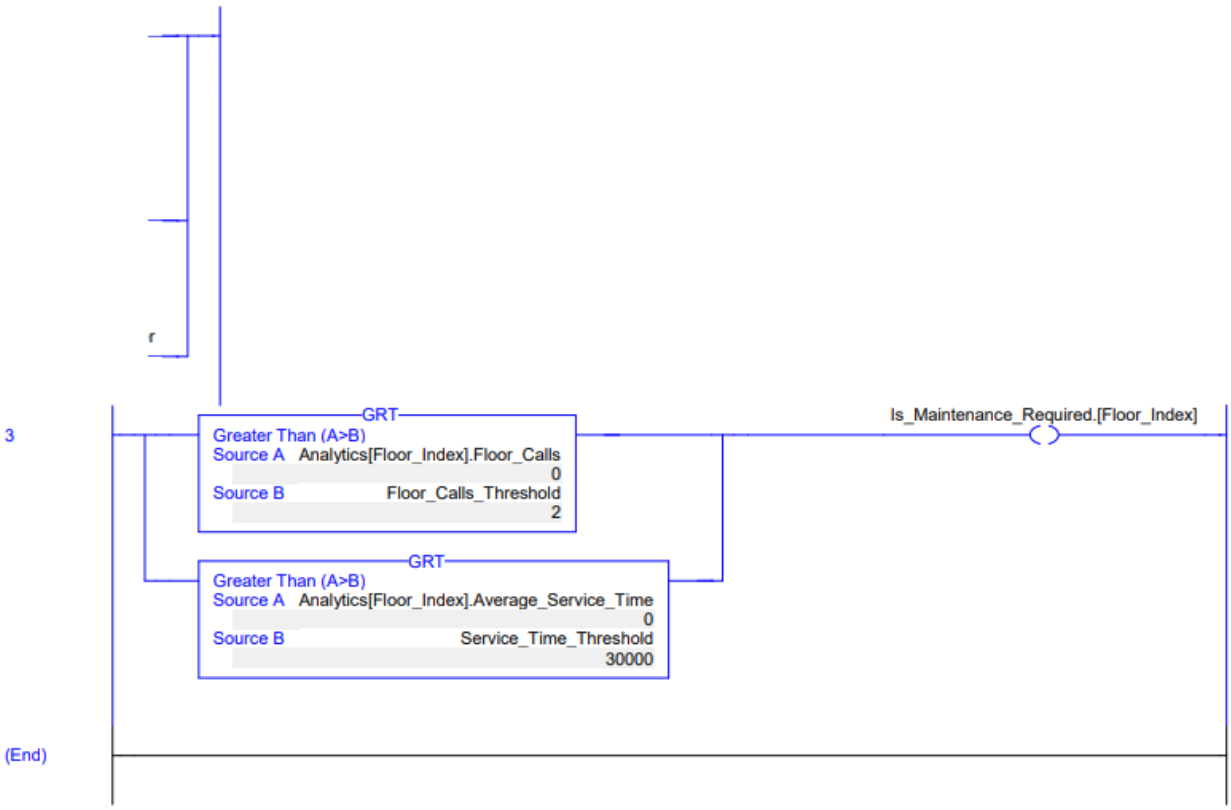

Figure 28 - Ladder Logic program of UpdateAnalysis (page 2 of 2)

FaultDetection

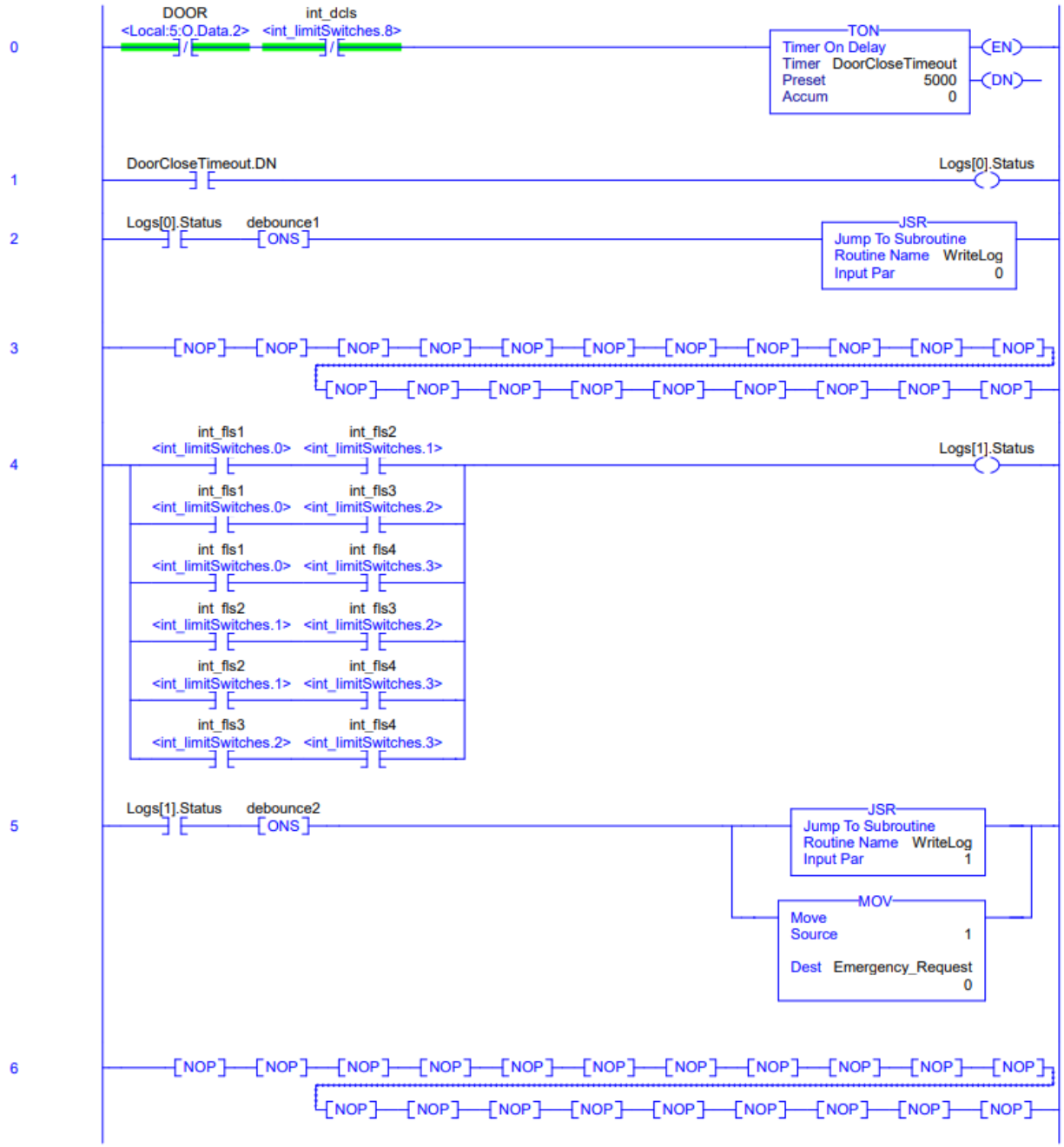

Figure 29 - Ladder Logic program of FaultDetection (page 1 of 2)

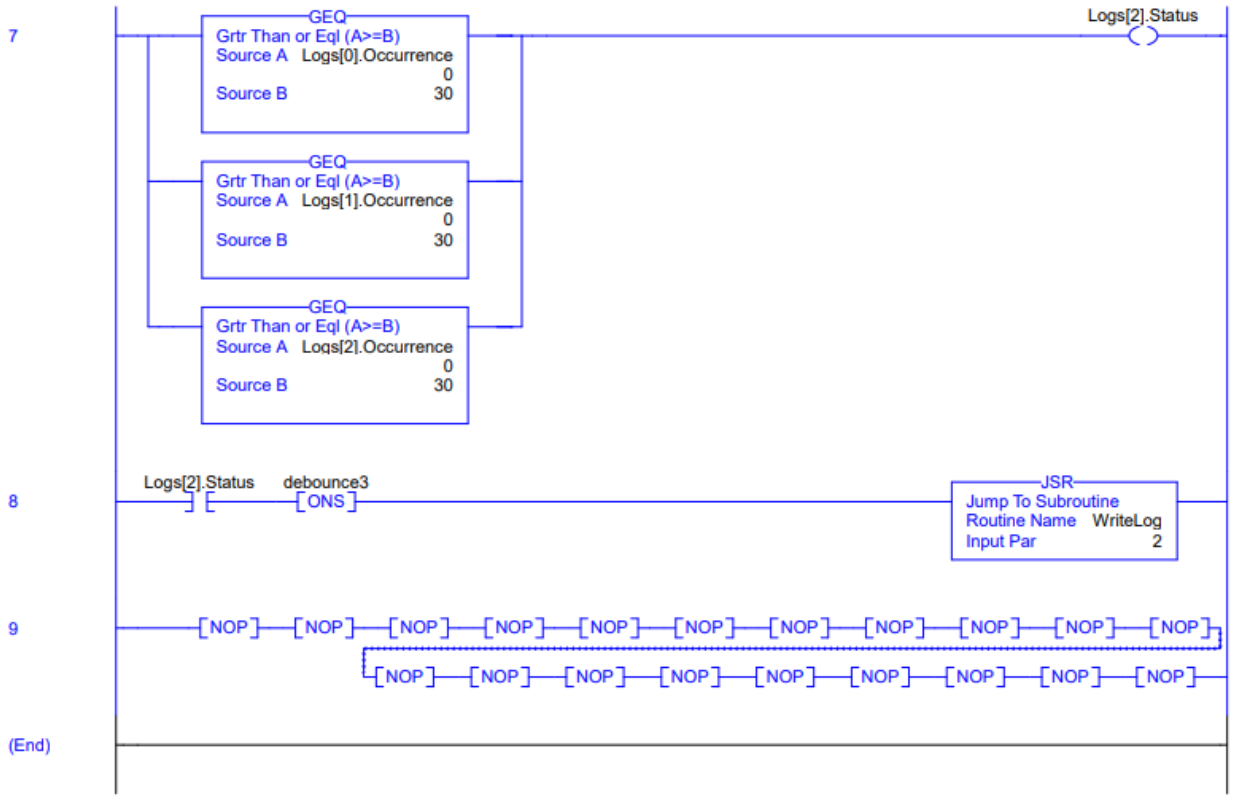

Figure 30 - Ladder Logic program of FaultDetection (page 2 of 2)

WriteLog

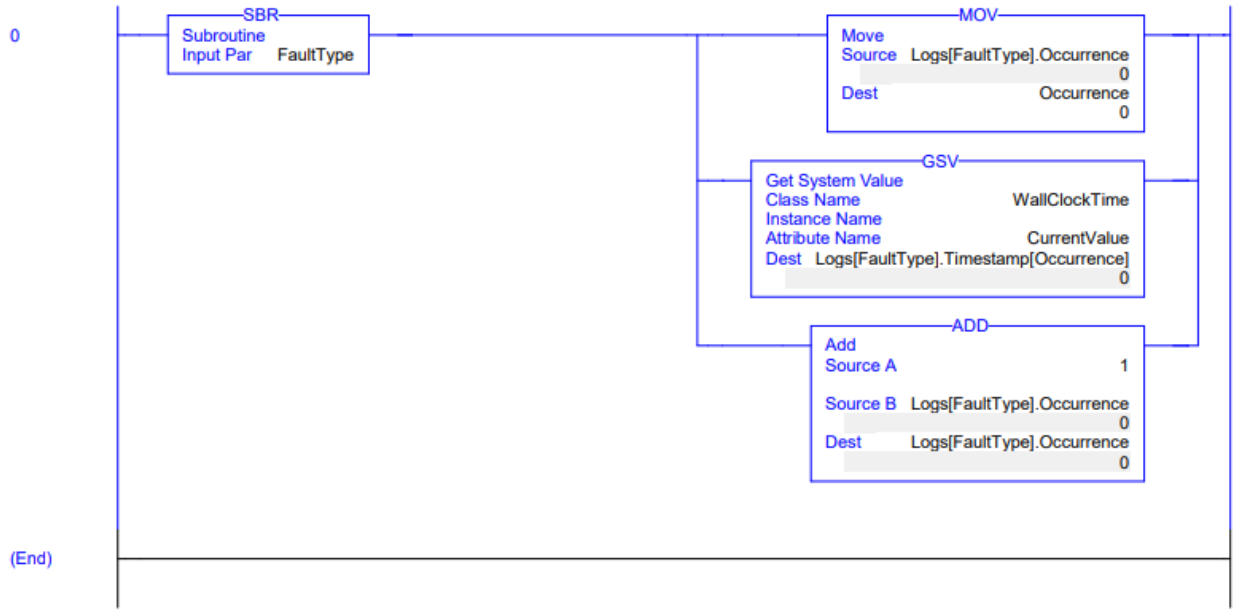

Figure 31 - Ladder Logic program of WriteLog

SyncSharedData

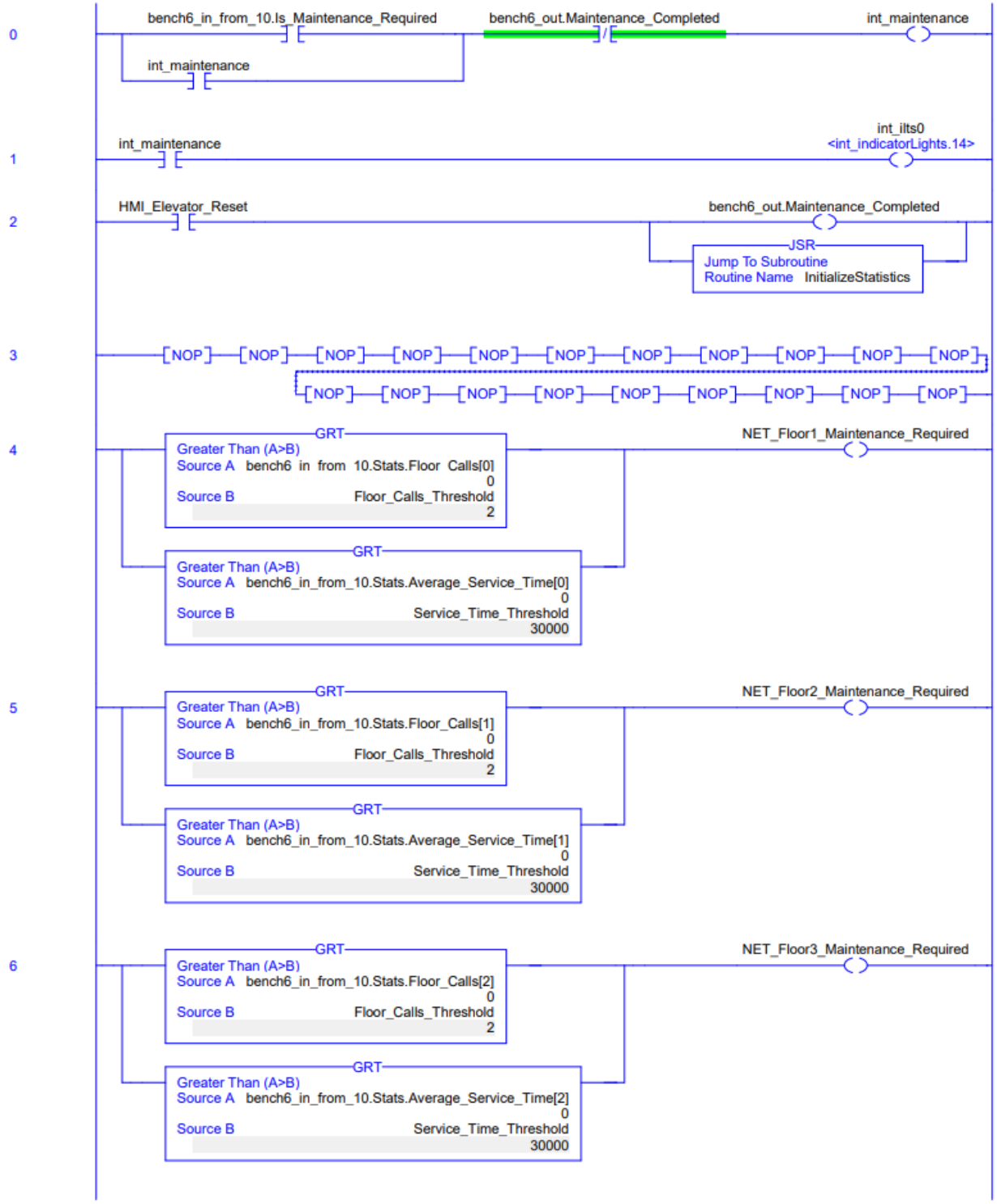

Figure 32 - Ladder Logic program of SyncSharedData (page 1 of 3)

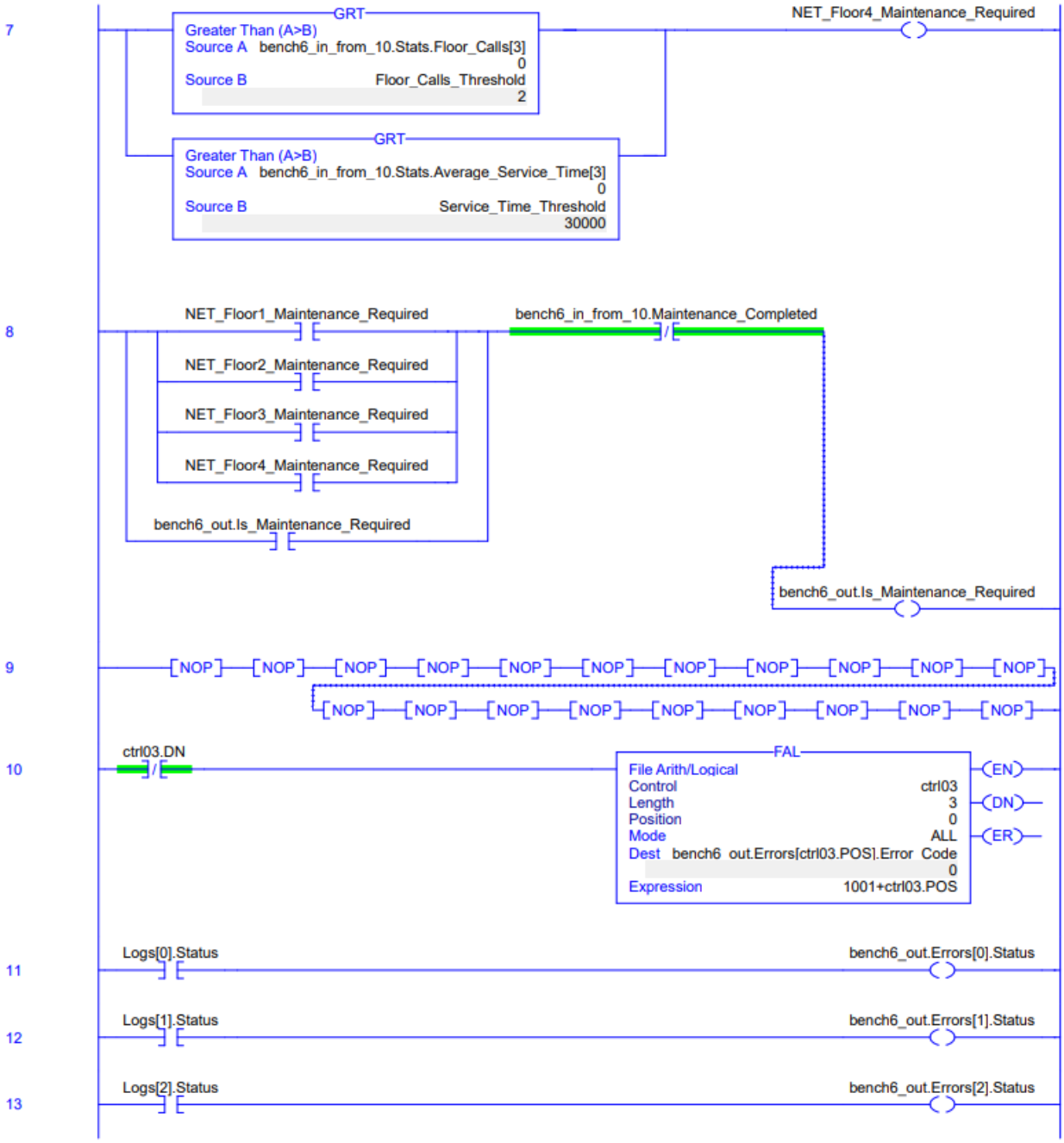

Figure 33 - Ladder Logic program of SyncSharedData (page 2 of 3)

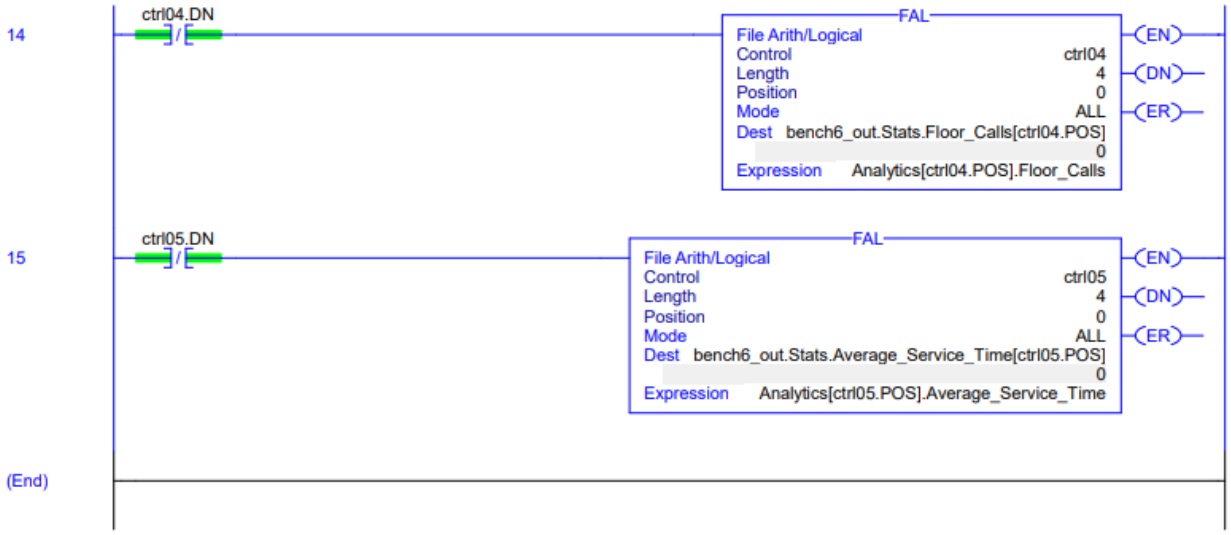

Figure 34 - Ladder Logic program of SyncSharedData (page 3 of 3)

SyncHMI

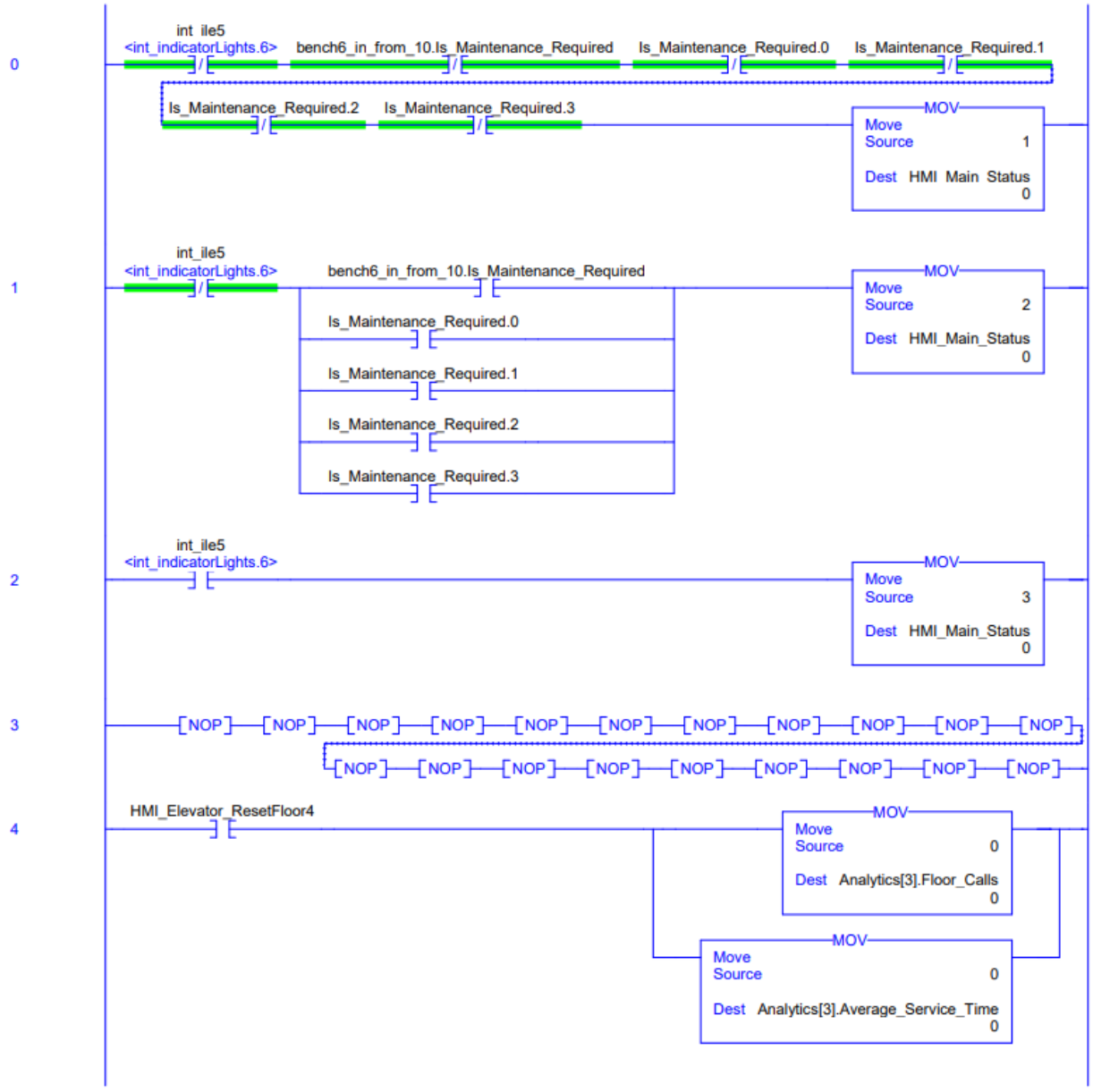

Figure 35 - Ladder Logic program of SyncHMI (page 1 of 2)

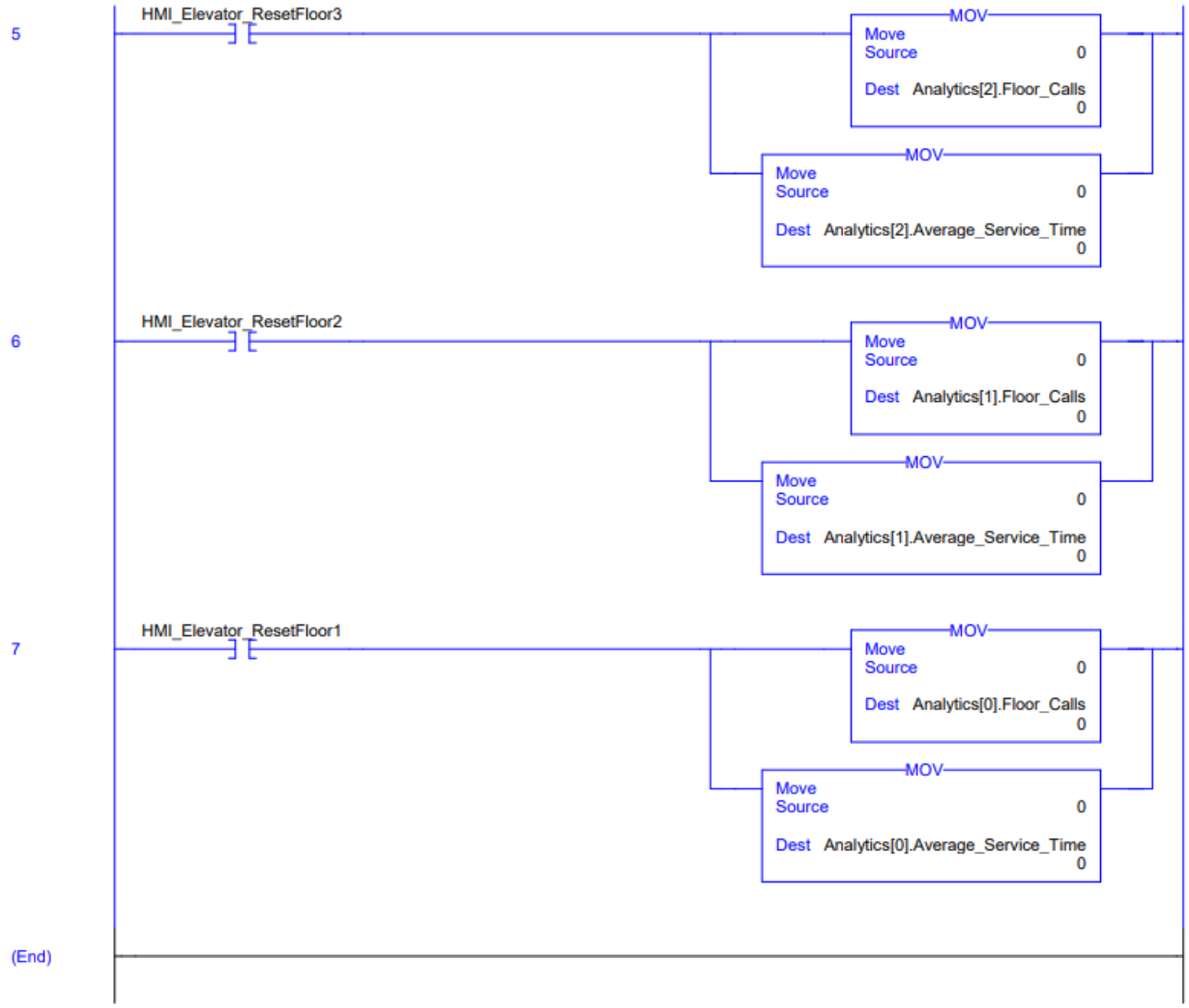

Figure 36 - Ladder Logic program of SyncHMI (page 2 of 2)

D. Fault Chart

Table 5 - List of fault information that are monitored during normal operation

Error Code	Description	Possible Causes	Response
1001	Door could not be closed within 5 seconds of the request	Faulty solenoid; faulty door close limit switch	Record and set fault status
1002	Multiple floor limit switches are sensing an object	Faulty floor limit switch(es); object is blocking floor limit switch(es)	Trigger emergency Stop
1003	Fault log is full	Log needs to be cleared in 'Administration' tab; faulty sensors	Record and set fault status

E. Using HMI Software

Status and Menu Bars

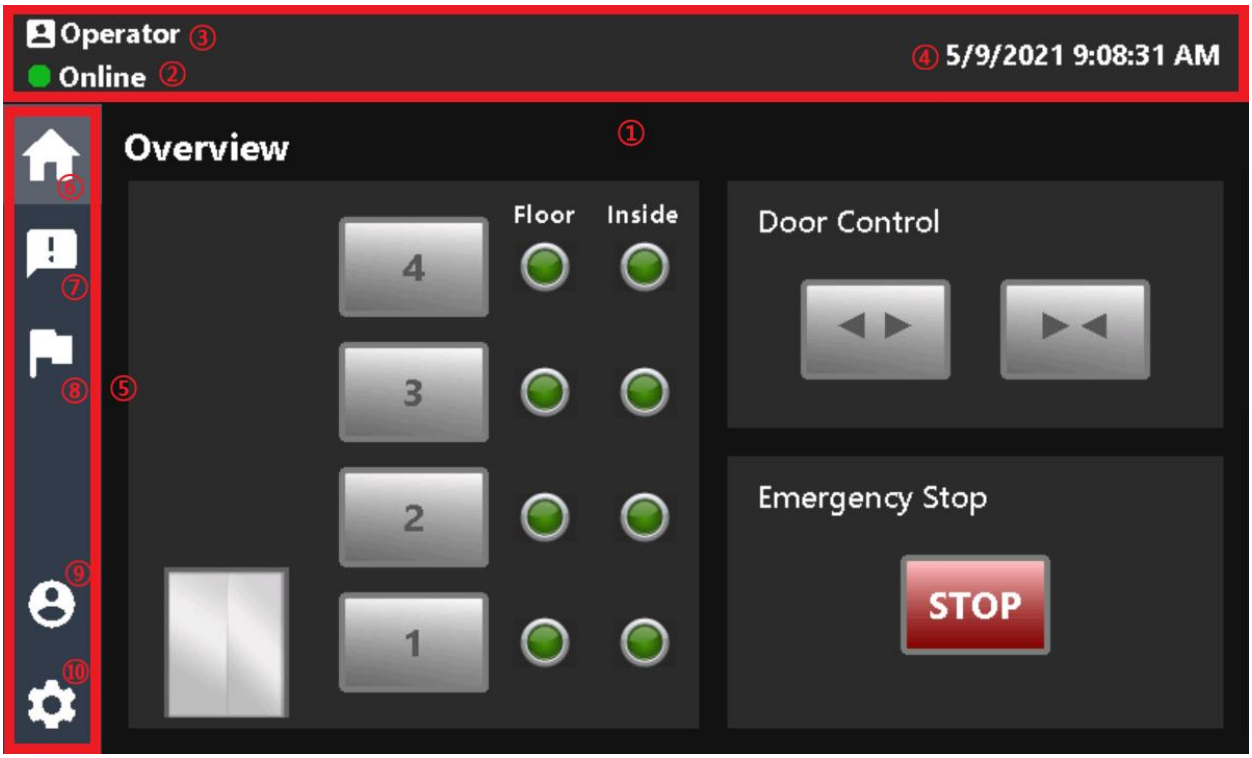

Figure 37 - HMI screenshot of status and menu bars

① Status Bar
② Status Indicator
③ Current User
④ Time
⑤ Menu Bar
⑥ Overview Tab
⑦ Diagnostics Tab
⑧ Administration Tab
⑨ Log In Pop-up
⑩ Display Settings Pop-up

Overview

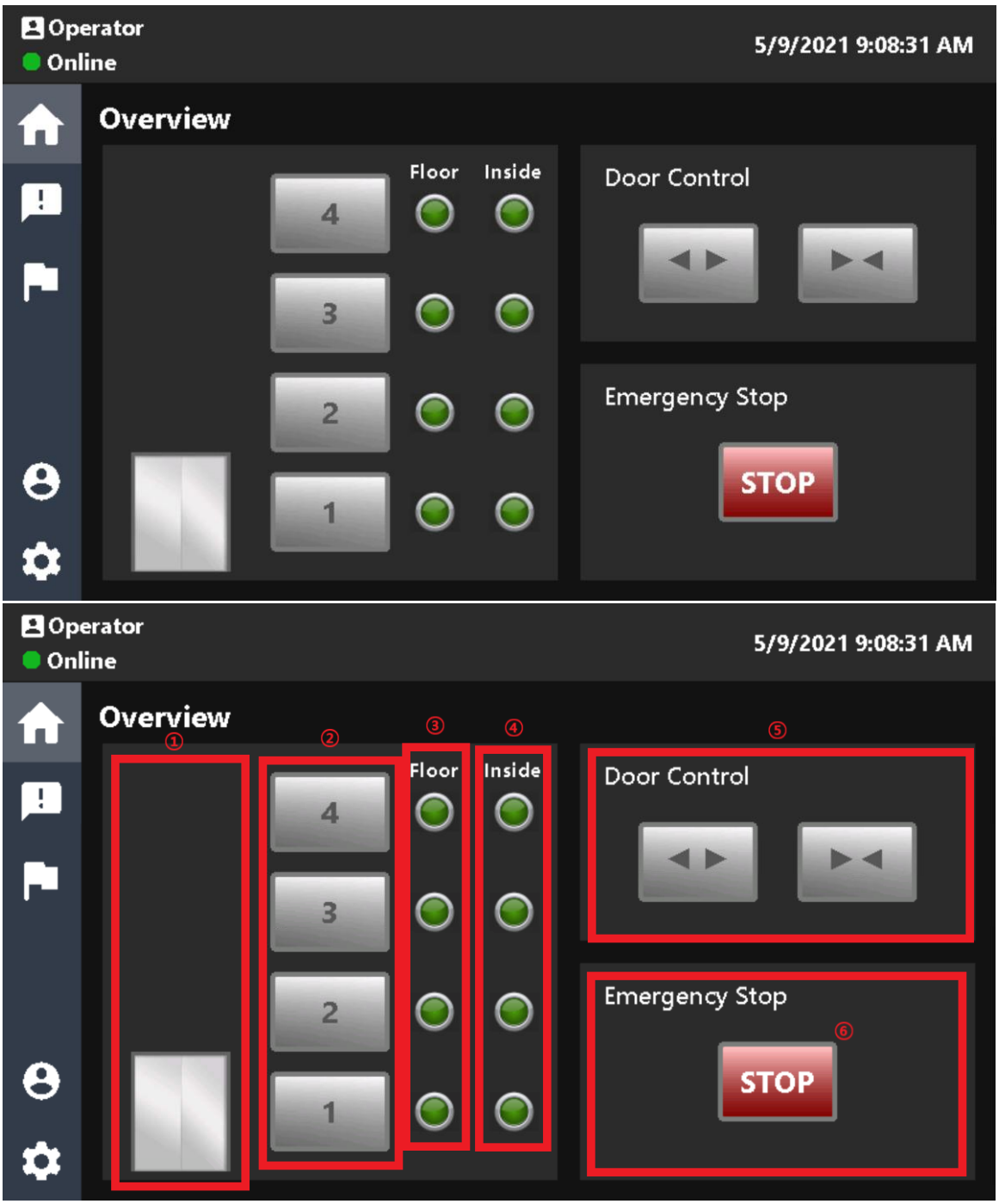

Figure 38 - HMI screenshot of Overview tab

① Elevator Status (Current Floor and Door)
② HMI Virtual Floor Pushbuttons
③ Floor Indicator Lights
④ Inside Panel Indicator Lights
⑤ Door Open and Close Pushbuttons

Emergency Stop

Figure 39 - HMI screenshot of emergency state alerts

① Status Indicator (Emergency)
② Blinking EMERGENCY text
③ Blinking Administration Tab alert

Diagnostics

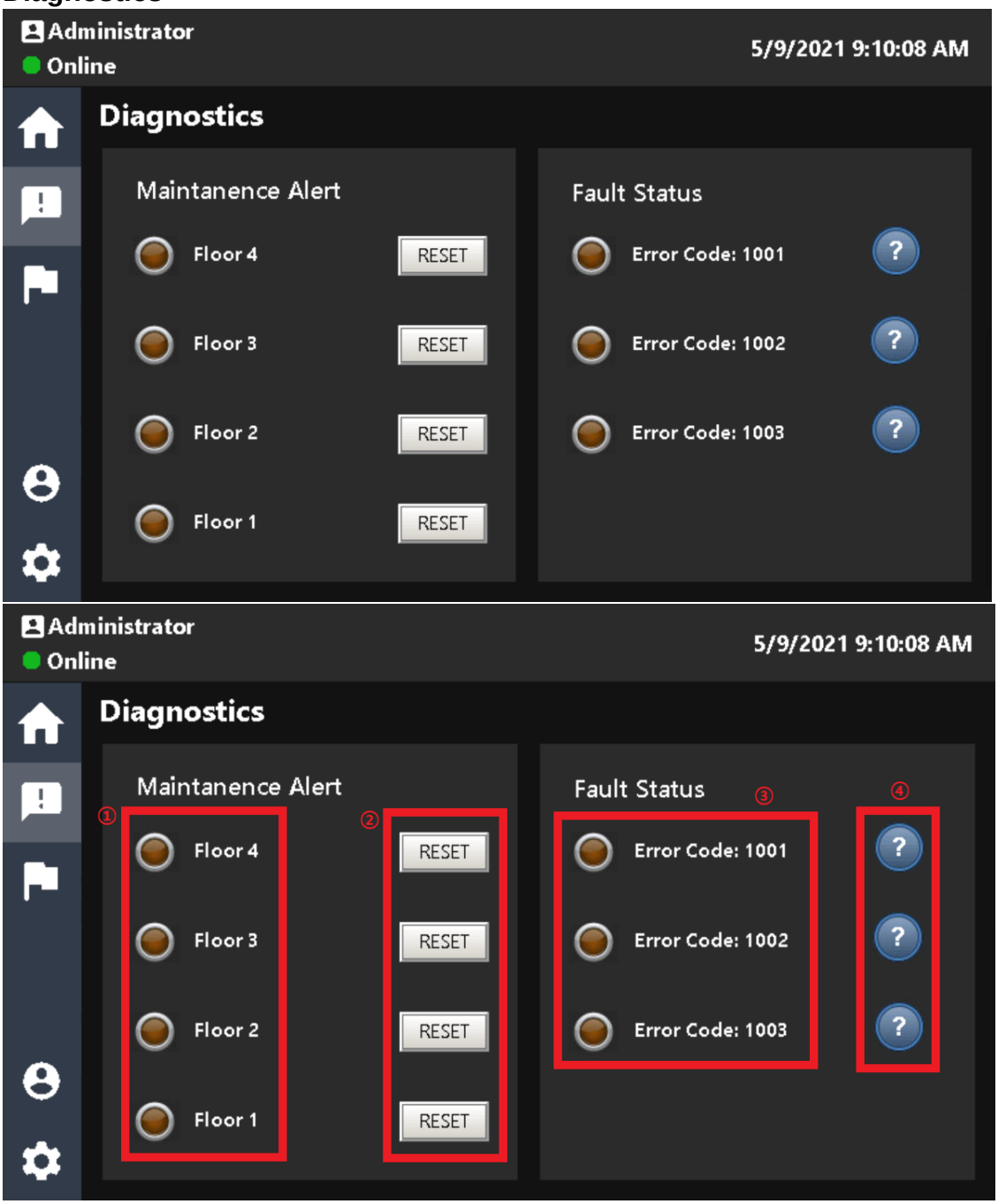

Figure 40 - HMI screenshot of Diagnostics tab

① Maintenance Alert Indicator Lights
② Analytics Counter Reset Pushbuttons
③ Fault Status Indicator Lights
④ Fault Help Menu Pushbuttons

Administration

Figure 41 - HMI screenshot of Administration tab

① Connected PLC Maintenance Alert Indicator Lights
② Connected PLC Floor Calls Data
③ Connected PLC Average Service Time Data
④ Connected PLC Fault Status Indicator Lights
⑤ Connected PLC Fault Help Menu Pushbuttons
⑥ Elevator Restart Pushbutton

Log In

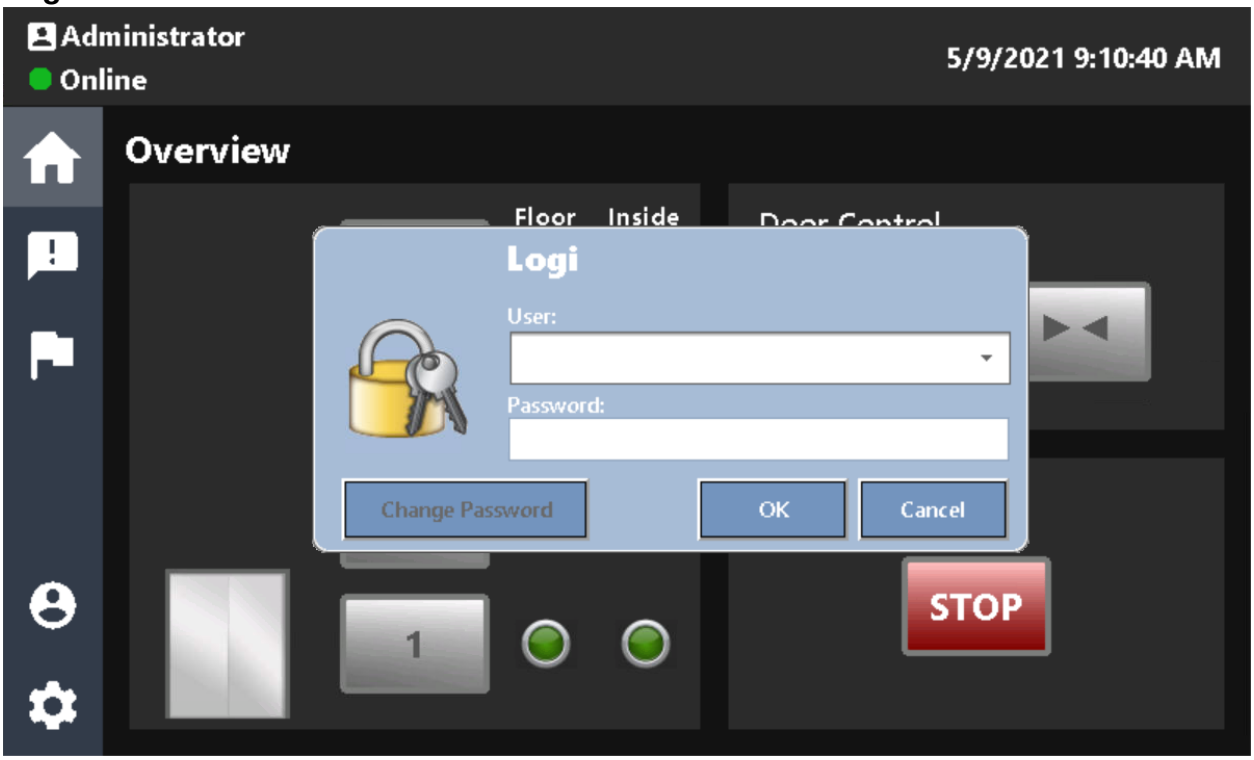

Figure 42 - HMI screenshot of Log-in Pop-up

Settings

Figure 43 - HMI screenshot of Display Settings Pop-up

Fault Help Menu

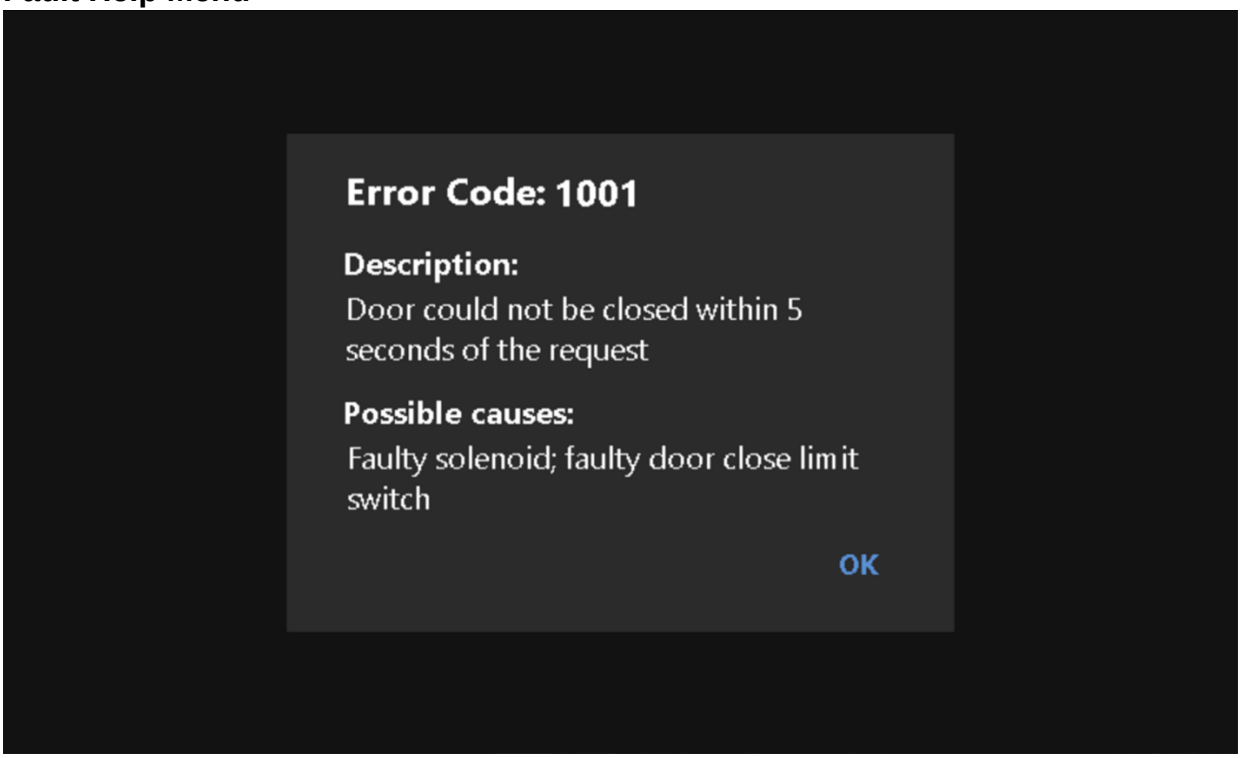

Figure 44 - HMI screenshot of fault help menu