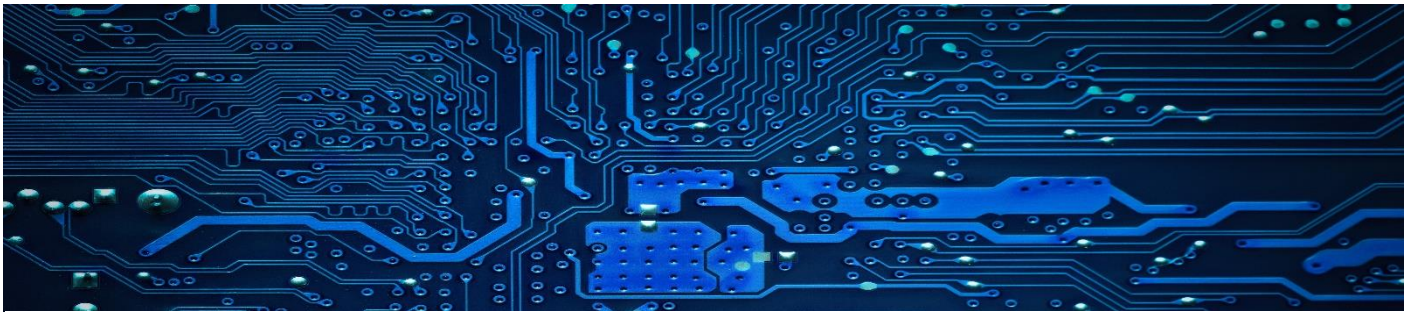


Development of Ladder Diagrams and SCADA Systems for the Festo Stations

Nimesha Jayatunge 20464631@student.curtin.edu.au
Edward Munns 18871773@student.curtin.edu.au

Mechatronics Automation Project 2023

Curtin University

Abstract:

In the first semester of 2023 a project was undertaken to implement PLC logic controllers and SCADA user interfaces for a three-station processing line consisting of a distribution, a testing and a sorting station designed to manipulate an assortment of input blocks.

The project required the station PLC software first be implemented on a station-by-station basis, then integrated together to form a processing line. SCADA user interfaces for the three stations were required to be implemented to allow system monitoring and control from a PC connected to the PLC network.

Sysmac Studio was used to develop the ladder logic for the PLCs. Each station's ladder logic consisted of function blocks for the actuators that performed certain actions based on the inputs received from various sensors on the station. Infrared transmitters and receivers allowed one way communication between the stations so that they could be integrated into a robust production line. The SCADA interface was developed using structured text in Ignition, and used animated sprites to make the user interface more intuitive.

The project was completed successfully, with a fully integrated robust production line developed by the deadline. Generalized and reusable function blocks were developed for each actuator, allowing for more efficient integration. The final production line is capable of receiving four types of workpieces, and sorting them into three trays based on colour. White pieces are rejected, black pieces are sorted into the leftmost tray, metal pieces into the middle tray, and red into the right tray.

In undertaking this project, a great deal was learned, specifically regarding the programming of PLCs using ladder logic. Methods to implement sequential operations, communications protocols, and sensor-based logic were developed. Furthermore, the importance of modularization and reusability of functions was realized. This project provided a strong foundation and understanding of PLC programming that will be vital in future projects.

Contents

Table of Figures:

1. Introduction and Background:

This report details the design and implementation of a three-station automated production line apparatus consisting of pneumatic and electrical sensors and actuators with process control implemented via individual station programmable logic controllers (PLCs). The three stations are designed to manipulate and process cylindrical blocks into classes based on color and height variations with the set of block colors consisting of white, black, red, and metallic and the set of block heights consisting of acceptable-height and over-height. The three stations are designed to be integrated into a serially chained processing line consisting of a distribution, a testing, and a sorting component. This report will first establish the project background by describing the mechanical, sensor and actuator configurations of the three stations. Next the approaches used for control implementation will be described on both an individual station and integrated processing line basis. Results of the described implementation will then be analyzed and discussed. The report will conclude with a reflection of the benefits of using the discussed design paradigms on final project outcomes.

1.1 Distribution Station:

This station forms the entry point of the system and is responsible for passing the input blocks from an input magazine to an output position on the downstream (testing) station. The station accomplishes this functionality by way of 2 actuators, various sensors, and a vacuum operated end effector. When a block is detected at the magazine, first the block is moved forward using the ejector actuator, then the swingarm actuator is moved to the ejected block location, the vacuum at the end effector is turned on and the block is picked. The swing arm is then transitioned to the presentation location at the subsequent (testing) station, and finally the vacuum is turned off placing the block. This process is repeated until the magazine is empty or the station state is changed. The station also had to be capable of identifying error conditions of ejector blocked, swing arm timeout and block dropped errors and entering an error state if an error occurs. The input/output signal bit description table, mechanical sub-components and system diagram of the distribution station can be found below.

1. PLC control bit Description:

Description	Data Direction	Variable Name	PLC Bit Address
Ejection cylinder extended	Input	Station_1B2	0.01
Ejection cylinder retracted	Input	Station_1B1	0.02
End effector vacuum present	Input	Station_2B1	0.03
Swing arm at magazine	Input	Station_3S1	0.04
Swing arm at downstream	Input	Station_3S2	0.05
Magazine empty	Input	Station_B4	0.06
Downstream IR input (Receiver)	Input	Station_IP_FI	0.07
Start button	Input	Panel_S1	0.08
Stop button	Input	Panel_S2	0.09
Reset button	Input	Panel_S4	0.11
Magazine eject	Output	Station_1Y1	1.00
End effector vacuum on	Output	Station_2Y1	1.01
Ejector pulse on	Output	Station_2Y2	1.02
Swing arm to magazine	Output	Station_3Y1	1.03
Swing arm to downstream	Output	Station_3Y2	1.04
Run mode indicator light	Output	Station_H1	1.08
Reset mode indicator light	Output	Station_H2	1.09
Magazine empty light (Q1)	Output	Panel_H3	1.10
Station error light (Q2)	Output	Panel_H4	1.11

Table 1:Distribution station bit description table (Curtin University, 2023).

3. Mechanical components:

Description
Ejector
Swing arm
End effector
Magazine

4. Station diagram:

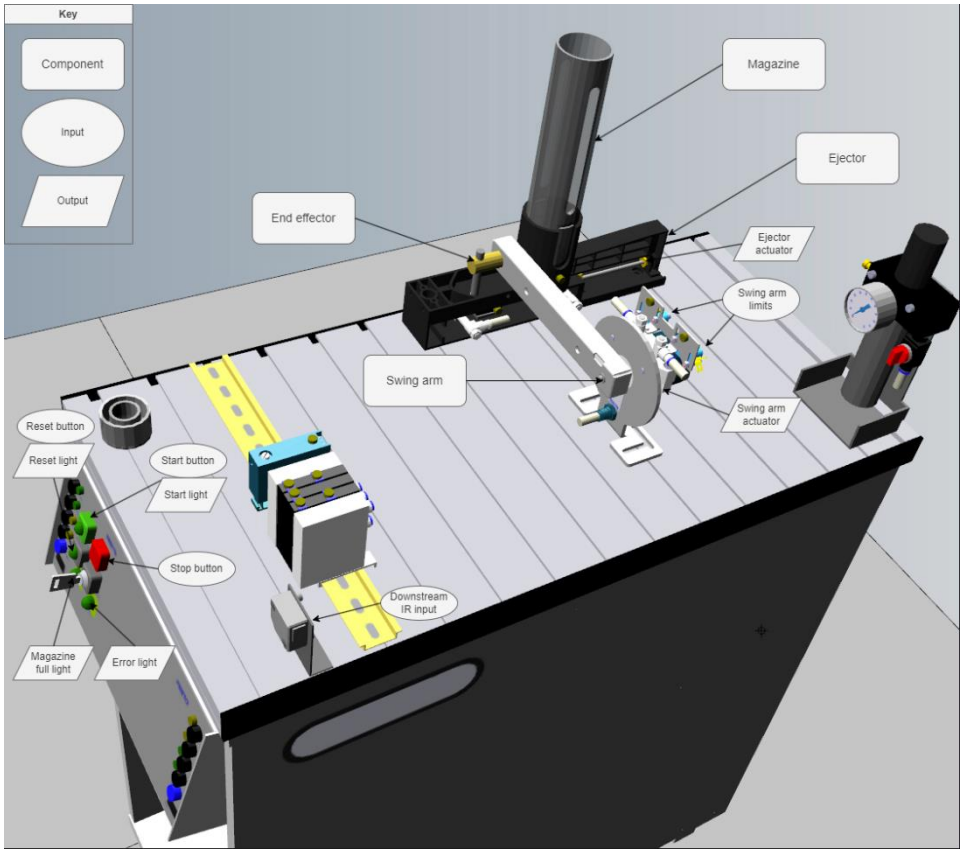


Figure 1: Distribution station diagram (CIROS Mechatronics).

1.2 Testing Station:

This station receives incoming blocks from the distribution station and tests the properties of the incoming blocks on height and color characteristics. To achieve this functionality the station is equipped with 2 actuators, a height sensor, a not-black sensor, and a part available sensor. When a part becomes available, first the part is assessed on color properties using the not black sensor, if the part is black and the rejection tray is not full the block is immediately ejected into the rejection tray. Otherwise, the block is translated vertically using the elevator actuator and the height of the block is evaluated with the height sensor. If the block is over-height it is a white block and is lowered down to the home position with the elevator actuator before being ejected into the reject tray, otherwise the block is ejected at the top elevator position and slides down the accepted tray to the subsequent (sorting) station. This process is repeated each time a new block is presented to the

station, or the station state is changed. The station also had to be capable of identifying error conditions of ejector blocked, elevator timeout, rejected tray full and safety beam interrupted errors and entering an error state if any error occurs. The input/output signal bit description table, mechanical sub-components and system diagram of the testing station can be found below.

1. PLC control bit Description:

Description	Data Direction	Variable Name	PLC Bit Address
Part available	Input	Part_AV	0.00
Part not black	Input	Station_B2	0.01
Safety through beam	Input	Station_B4	0.02
Part height correct	Input	Station_B5	0.03
Elevator at top	Input	Station_1B1	0.04
Elevator at bottom	Input	Station_1B2	0.05
Ejector retracted	Input	Station2B1	0.06
Downstream IR input (Receiver)	Input	Station_IP_FI	0.07
Start button	Input	Panel_S1	0.08
Stop button	Input	Panel_S2	0.09
Reset button	Input	Panel_S4	0.11
Elevator down	Output	Station_1Y1	1.00
Elevator up	Output	Station_1Y2	1.01
Advance ejector	Output	Station_2Y1	1.02
Air cushion on	Output	Station_3Y1	1.03
Upstream IR output (Transmitter)	Output	Station_IP_N_FO	1.07
Run mode indicator light	Output	Station_H1	1.08
Reset mode indicator light	Output	Station_H2	1.09
Rejected queue full light (Q1)	Output	Station_H3	1.10
Error light (Q2)	Output	Station_H4	1.11

Table 2: Testing station bit description table (Curtin University, 2023).

3. Mechanical components:

Description
Elevator
Rejected queue
Ejector
Output tray

4. Station diagram:

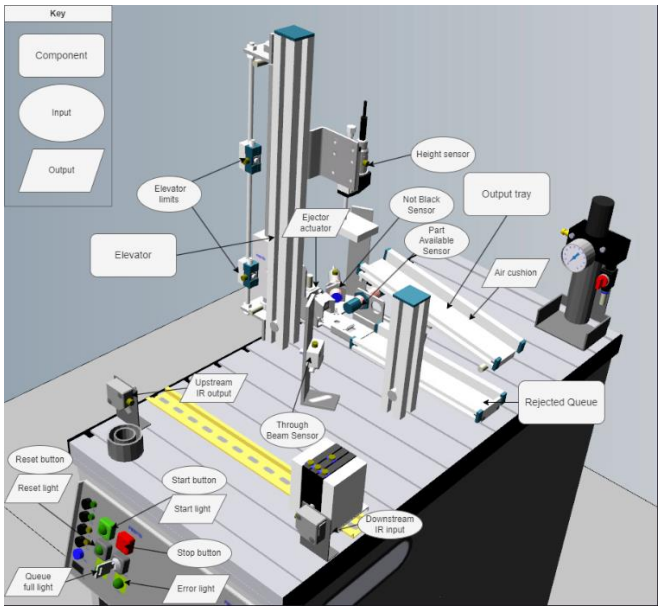


Figure 2: Testing station diagram (CIROS Mechatronics).

1.3 Sorting Station:

This station receives incoming blocks from the previous (testing) station, evaluates them on color properties and sorts them into color classes of black, metallic and red. To achieve this functionality the station is equipped with a stopper actuator to halt block progression while the previous block is handled, a horizontal conveyor belt system powered by a DC motor, two branch actuators that control branch intercept levers for the first two color class trays, and a part available, a not-black and an is-metallic sensor. When a block is presented to the station, first the color attributes of the block are evaluated with the not-black and is-metallic sensors, the appropriate intercept arm actuator is moved forward, the stopper actuator is momentarily moved to the back position and the conveyor is powered on. The stopper is then returned to the forward position to prevent further blocks from progressing until the current block has been deposited into the appropriate tray. Once the current block has been deposited the interception arms are returned to the home position and the process is repeated until no block is present at the presentation location or the station state is changed. The station also had to be capable of identifying error conditions of sorting tray overloaded, branch 1 and branch 2 actuator timeout and conveyor timeout errors and entering an error state if an error occurs. The input/output signal bit description table, mechanical sub-components and system diagram of the sorting station can be found below.

1. PLC control bit Description:

Description	Data Direction	Variable Name	PLC Bit Address
Part available	Input	Part_AV	0.00
Metallic block	Input	Station_B2	0.01
Not black block	Input	Station_B3	0.02
Rejected slide full	Input	Station_B4	0.03
Branch 1 retracted	Input	Station_1B1	0.04
Branch 1 extended	Input	Station_1B2	0.05
Branch 2 retracted	Input	Station_2B1	0.06
Branch 2 extended	Input	Station_2B2	0.07
Start button	Input	Panel_S1	0.08
Stop button	Input	Panel_S2	0.09
Reset button	Input	Panel_S4	0.11
Conveyor motor on	Output	Station_K1	1.00
Branch 1 to extended	Output	Station_1Y1	1.01
Branch 2 to extended	Output	Station_2Y1	1.02
Stopper to retracted	Output	Station_3Y1	1.03
Downstream IR output (Transmitter)	Output	Station_IP_N_FO	1.07
Run mode indicator light	Output	Station_H1	1.08
Reset mode indicator light	Output	Station_H2	1.09
Magazine empty light (Q1)	Output	Panel_H3	1.10
Station error light (Q2)	Output	Panel_H4	1.11

Table 3:Table 2: Sorting station bit description table (Curtin University, 2023).

3. Mechanical sub-systems:

Description
Stopper
Branch arm 1
Branch arm 2
Conveyor belt
Black output tray
Metallic output tray
Red output tray

4. Station diagram:

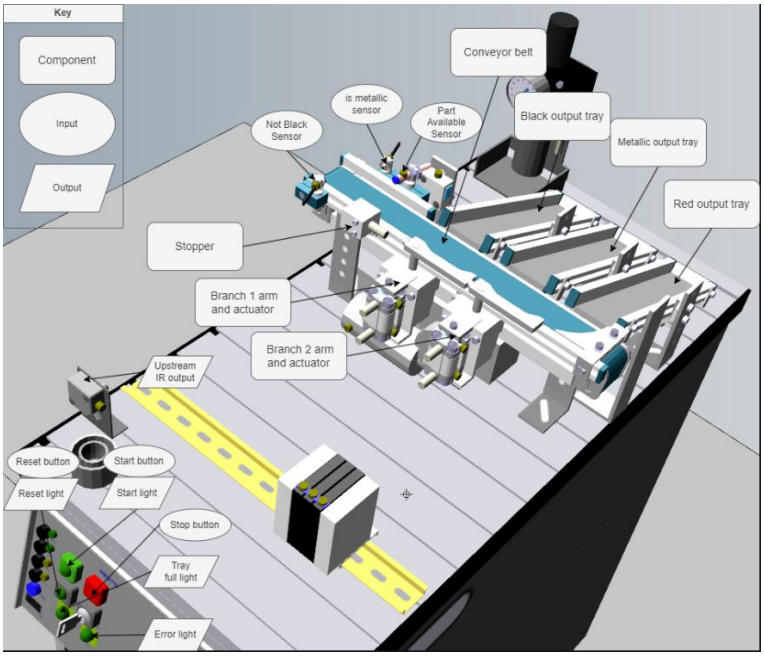


Figure 3: Sorting station diagram (CIROS Mechatronics).

2. Approaches Used:

This section will explain the approaches used in the planning, design, and implementation stages of the project software development. Both the PLC logic and SCADA user interface methodologies will be discussed along with the methods used for software testing.

2.1 Project planning:

This project began with the analysis of the major project milestones and their associated deadlines. Results of this analysis were then collated into a Gantt chart allocating time frames to all tasks. As the project progressed, this Gantt chart was re-examined at the completion of each significant project milestone and adjusted if required. This monitoring of progress was essential for ensuring group members were constantly aware of time constraints at all points of the development process.

2.2 PLC hardware logic control development methodologies:

PLC logic software was developed using the Sysmac Studio PLC integrated development environment (IDE) and the ladder network programming method. This programming method is a graphical programming technique in which logic is programmed by inserting inputs and outputs represented by graphical switches into “rungs” that make up ladder networks. The IDE has a pre-built library of common function blocks and additionally allows for the creation of user defined function blocks. These function blocks contain an internal ladder network for block logic and inputs/outputs from

these blocks can be configured as per user requirements. This allows for function blocks to be used in the same way as a class in a traditional Object-Oriented Programming (OOP) language. Using this OOP approach to development makes it possible to implement multiple instances of the same function block for control of any hardware sub-system of identical or similar functionality.

Each PLC program required for the project began by separating the system into smaller sub-systems with each sub-system requiring a function block instance within the main program. This sub-system separation was carried out using function block diagrams to help in assessing block input and output requirements. The internal logic of function blocks was then designed using state diagrams before being implemented with a ladder network.

2.3 SCADA user interface development methodologies:

The Supervisory Control And Data Acquisition (SCADA) User Interface (UI) software was developed using the Ignition IDE. This IDE is designed specifically for graphical UI design of PLC systems with the ability to connect to a network attached PLC and read/write PLC software variables in real time. This process began with the creation of a UI layout diagram showing all required UI elements and possible block positions for all station program states. Required variables were then added to the PLC ladder network if not yet implemented before being exported from the PLC software and imported into the SCADA software. These variables can be accessed from within the IDE using tags when the PLC is connected. These tags can then be used for UI logic development by binding them to UI element properties/states using either the direct binding method or the binding through expressions method (Cui, 2023).

2.4 Software testing methodologies:

Testing of the PLC logic software was initially performed virtually using the CIROS Mechatronics PLC simulation software and the appropriate station 3D model. After virtual testing of software was successful programs were then hardware tested by uploading to the relevant physical station PLC and performing real-world hardware tests. Once hardware testing of the PLC logic was successful the SCADA UI software was tested by running it simultaneously with the PLC software on the physical hardware.

3. Results and Discussion:

This section will discuss the results of the project implementation, first on a stand-alone station basis across the three stations, then on an integrated production line basis. This section will also cover the development of an inter-station communications protocol implemented during the integration phase of the project and SCADA based human machine interface (HMI) control and monitoring software built for each station. The software development paradigms employed to improve abstraction, modularization, and separation of concerns of control logic function blocks used during development of the project is described in detail in order to demonstrate why these methods of

development were selected for use in the project and to highlight the benefits of using these methods for PLC based system design.

3.1 Distribution station:

The development of the distribution station control logic began with the separation of sub-systems into function blocks as a means of breaking the task down into smaller sub-tasks. The function blocks used for the distribution station during this preliminary development phase were:

- The control panel
- The swing arm actuator
- The ejector actuator
- The end effector vacuum

The control panel function block was implemented with hardware inputs from the start, stop and reset buttons and hardware outputs for the start, reset, magazine empty (Q1) and system error (Q2) lights. This function block implemented a station wide master state machine with states of run, pause, reset, and error modes which were output to other function blocks in the system using an internal software Boolean variable for each state. Error signals from other sub-systems were also input into this function block to signal an error had occurred and put the station into error mode. As this function block is identical across all other stations covered in this report, the block will be omitted from subsequent station implementation descriptions. A state diagram of this sub-system can be seen in figure 4.

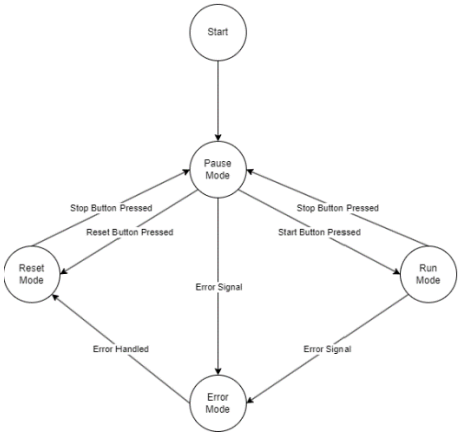


Figure 4: Control panel state diagram.

The swing arm function block was implemented using hardware input signals from the at_home and at_magazine actuator limit switches and hardware outputs to the to_home and to_magazine actuator control solenoids. Station state Boolean variables from the control panel function block were input to this block to allow the sub-system to respond to state changes of the station wide master state machine, and output variables for the swing-arm actuator states of at_home, to_home, at_end, to_end, to_resting and at_resting. An internal timer was used to time the transition period the swing-arm took to reach the mid-point of travel with this timer timeout signaling the at_resting state had been reached. The block sends an output signal to the ejector function block to signal a block should be ejected based on the current swing arm state and the magazine empty signal. An

internal timer timeout was used to identify a swing arm timeout error and forward this signal to the control panel block. The state diagram of the swing arm actuator can be seen in figure 5.

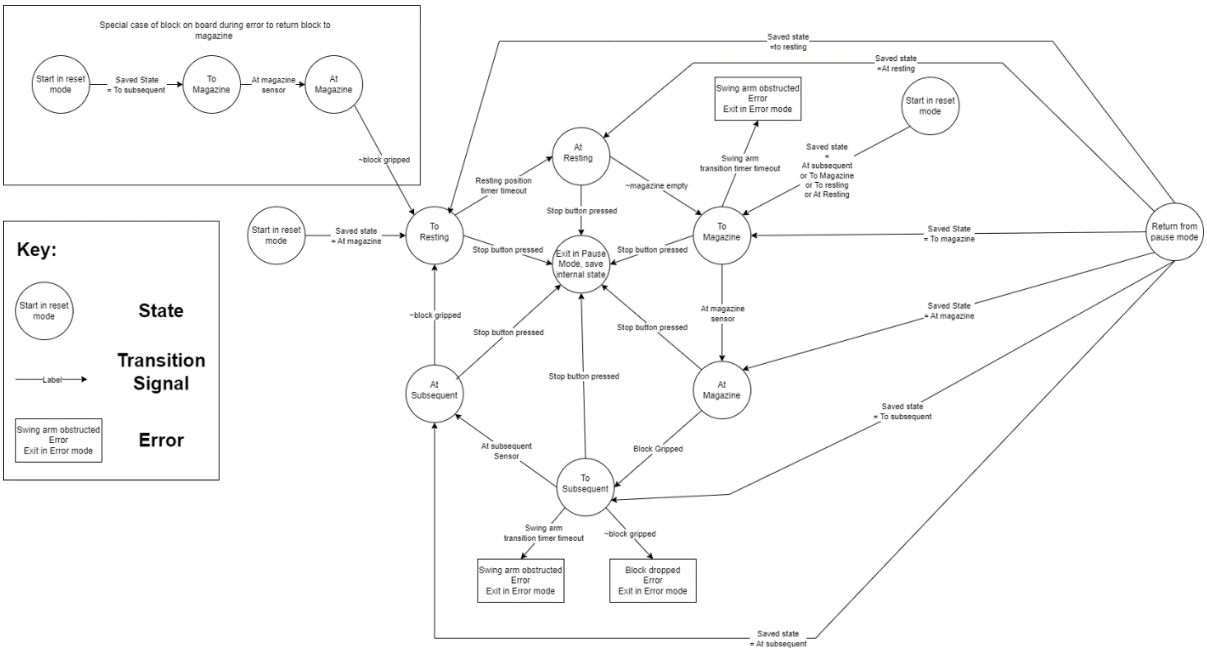


Figure 5: Distribution station swing arm actuator state diagram.

The ejector actuator function block was implemented using the input from the magazine empty sensor to evaluate if a block was available, two inputs for the ejector at-home and at-end limit switches, an output to the actuator control solenoid to enable control of the actuator, software inputs for the four master station states, a software input from the swing-arm function block used to signal a block should be ejected and a software output of the current magazine status (block available, not available). The function block used an internal timer to detect the occurrence of an ejector blocked error and forward this signal to the control panel function block.

The vacuum function block was used for the end effector vacuum control logic and received an input from the vacuum present sensor to establish if a block was currently gripped, a vacuum-on control signal from the swing-arm actuator block, station master state inputs and a hardware output to the vacuum control solenoid to enable hardware control of the end effector vacuum. The function block can identify if a block has been dropped with a combination of the block-gripped and vacuum-on signals and forwards this error condition to the master state machine if it occurs.

The function block diagram of this system can be seen in figure 6.

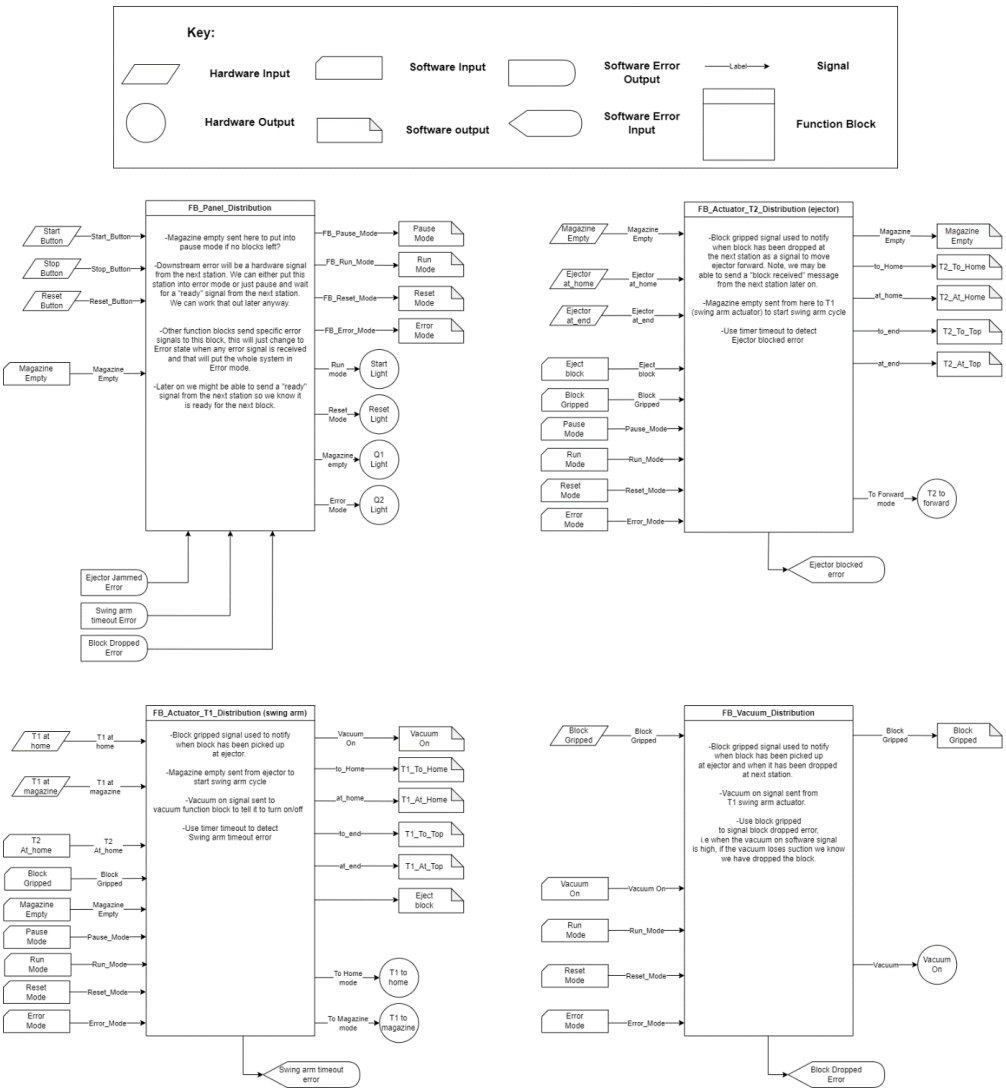


Figure 6: Distribution station function block diagram.

3.2 Testing station:

Initial development of the stand-alone testing station was implemented by first dividing the system into sub-systems and creating a function block for each sub-system. The initial sub-systems were:

- The control panel
- The sensor block
- The elevator actuator
- The ejector actuator

The sensor block was responsible for evaluation of the block properties via the not-black and height hardware sensor inputs, part availability via the part-available hardware sensor input, detection of safety through beam errors using the safety through beam hardware sensor input, detection of rejected tray full and overloaded error status by keeping a count of rejected blocks throughout the testing process with an integer variable. Master station state output is monitored via four station master state software inputs. The sensor block is also responsible for outputting control signals to both the elevator and ejector actuator function blocks and controlling the output tray air cushion

solenoid for accepted block passing to the sorting station. A state diagram of this system can be seen in figure 7.

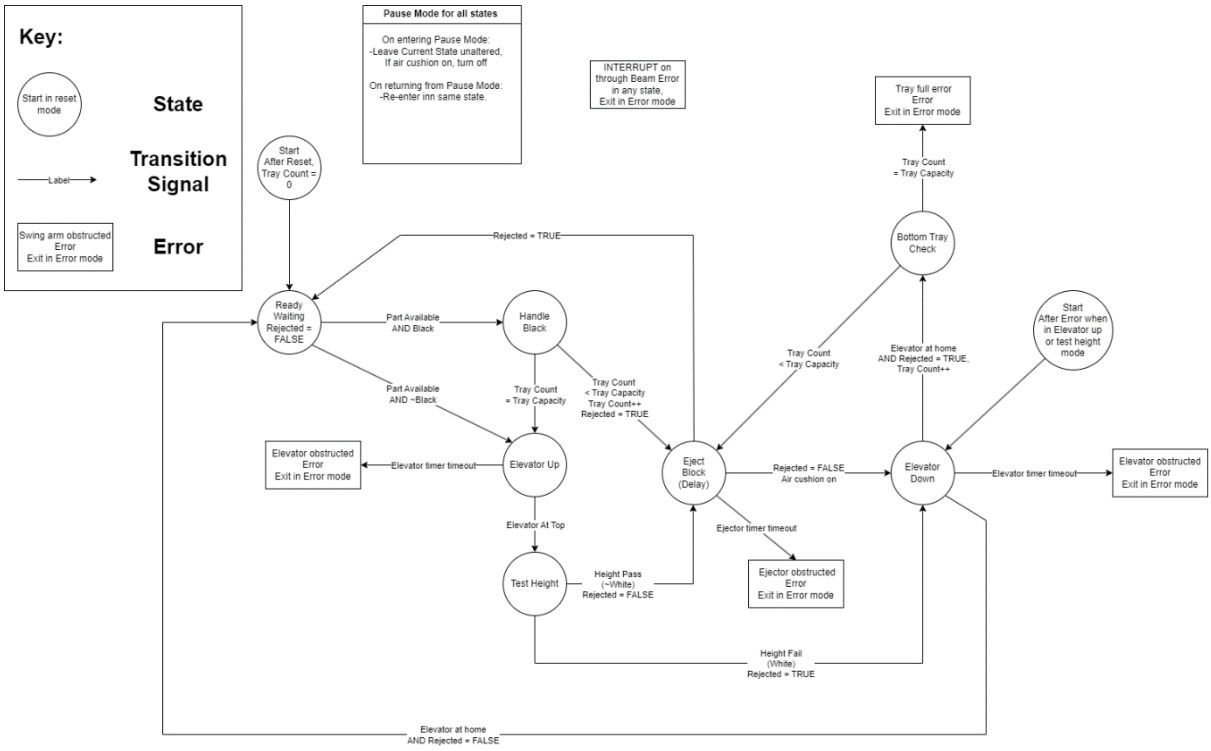


Figure 7: Sensor function block state diagram.

The elevator function block evaluates hardware inputs for the at-home and at-end limit switches of this actuator and control signal inputs from the Sensor block to establish the currently required state the actuator should be in. The block includes the two control solenoid hardware outputs to control the elevator position and outputs the current internal elevator block state via four software outputs of to-home, at-home, to-end, and at-end which are used as inputs to the Sensor block. An elevator timeout error can be detected with an internal timer and this error signal is passed to the control panel function block to signal an error has occurred.

The ejector actuator function block is responsible for monitoring the hardware limit switch input at the home position and outputting the ejector to end hardware control output signal. The block receives control signals from the sensor block signaling when the actuator should transition to the end position or the home position. The block also outputs the current internal state via the at-home, to-home, at-end, and to-end variables which are used as input to the sensor block for station logic control. The block is equipped with an internal timer used to identify an ejector blocked error and forward this signal to the control panel function block. The function block diagram of the testing station system can be seen in figure 8.

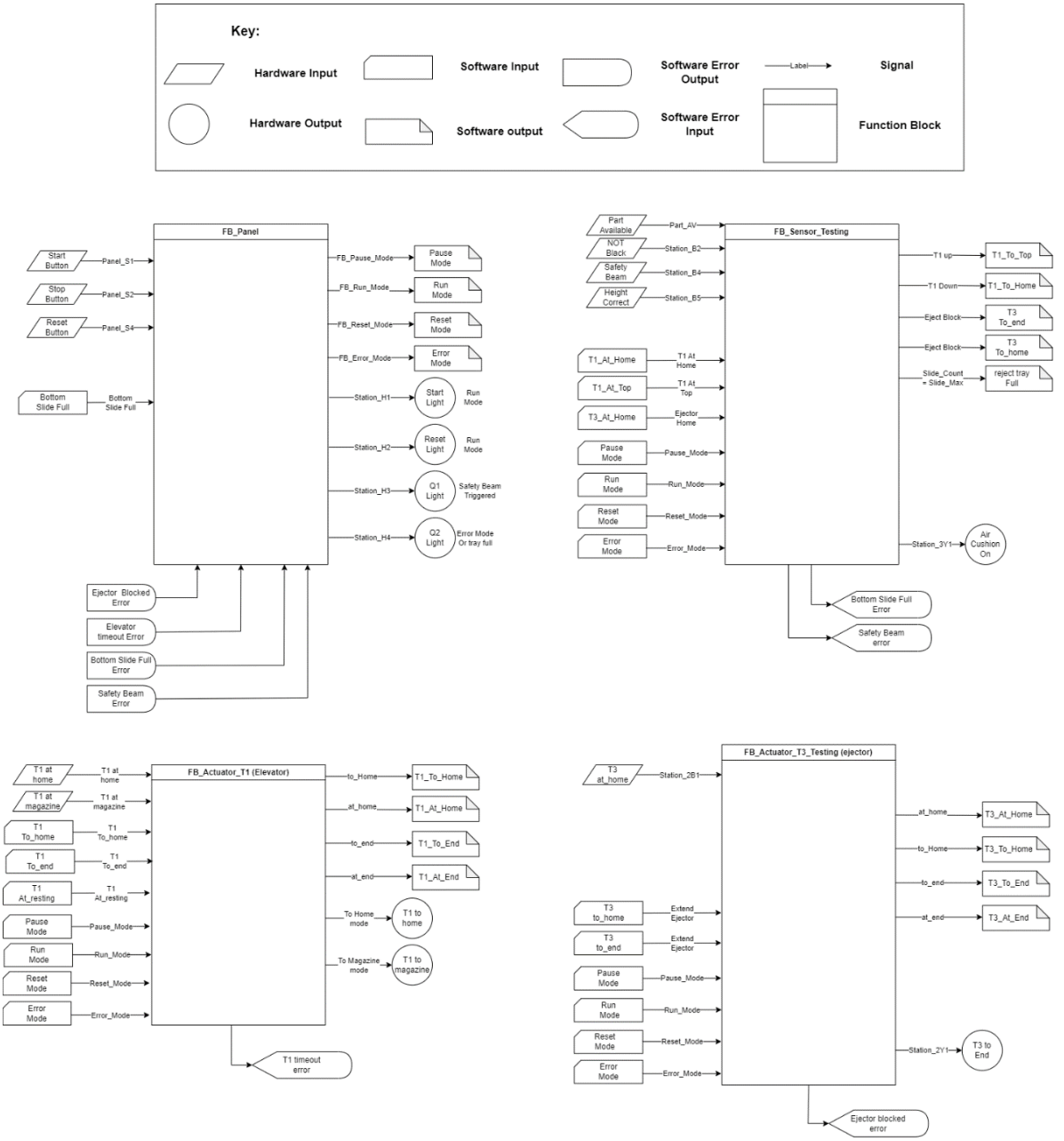


Figure 8: Testing station function block diagram.

3.3 Sorting station:

The stand-alone development of the sorting station control system began with the separation of the sub-systems into the following function blocks:

- The control panel
- The Logic block
- The branch 1 actuator
- The branch 2 actuator
- The Stopper actuator

The Logic function block was designed to handle all block sorting logic, send command signals to the branch 1, branch 2 and Stopper actuators, control conveyor belt on/off state and keep count of the

processed blocks in the 3 color output trays. To accomplish this the Logic block had hardware input signals for the Part-available, the not-black, the is-metallic and the slide overflow sensors. To enable actuator control, software output signals for branch-1 and branch-2 extend and stopper retract were implemented and the branch 1 and 2 actuator states were monitored with software input signals from these blocks. The block was also equipped with an internal timer to enable detection of block-lost errors via this timer timeout and would signal a slide overload error if the slide overload sensor was triggered, relaying these signals back to the station master state machine if an error was detected. A state diagram of this Logic block state machine can be seen in figure 9.

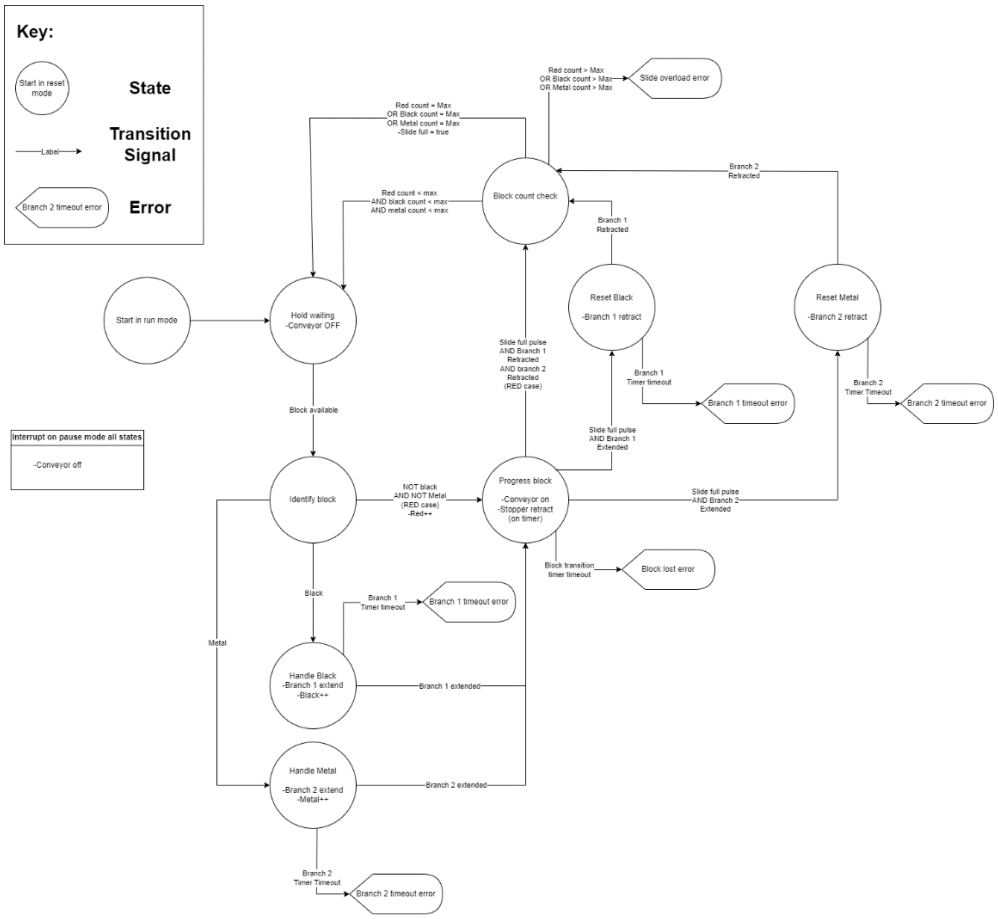


Figure 9: Sorting station Logic block state diagram.

The branch 1 and 2 function blocks were both implemented as instances of the now generic Actuator_T2 function block. These had hardware inputs for the at-home and at-end limit switches and a hardware output for the to-end actuator control solenoid. These blocks functioned as “slaves”, taking input control signals and outputting internal state signals to and from the Logic “master” block. They were equipped with internal timers for home-to-end transition timeout error detection, relaying error signals back to the station master state machine.

The Stopper actuator function block was implemented with an instance of the Actuator_T3 function block, with a hardware input for the at-home limit switch, hardware output for the actuator control solenoid and control input and state output to the Logic “master” block.

A function block diagram of this system can be seen in figure 10.

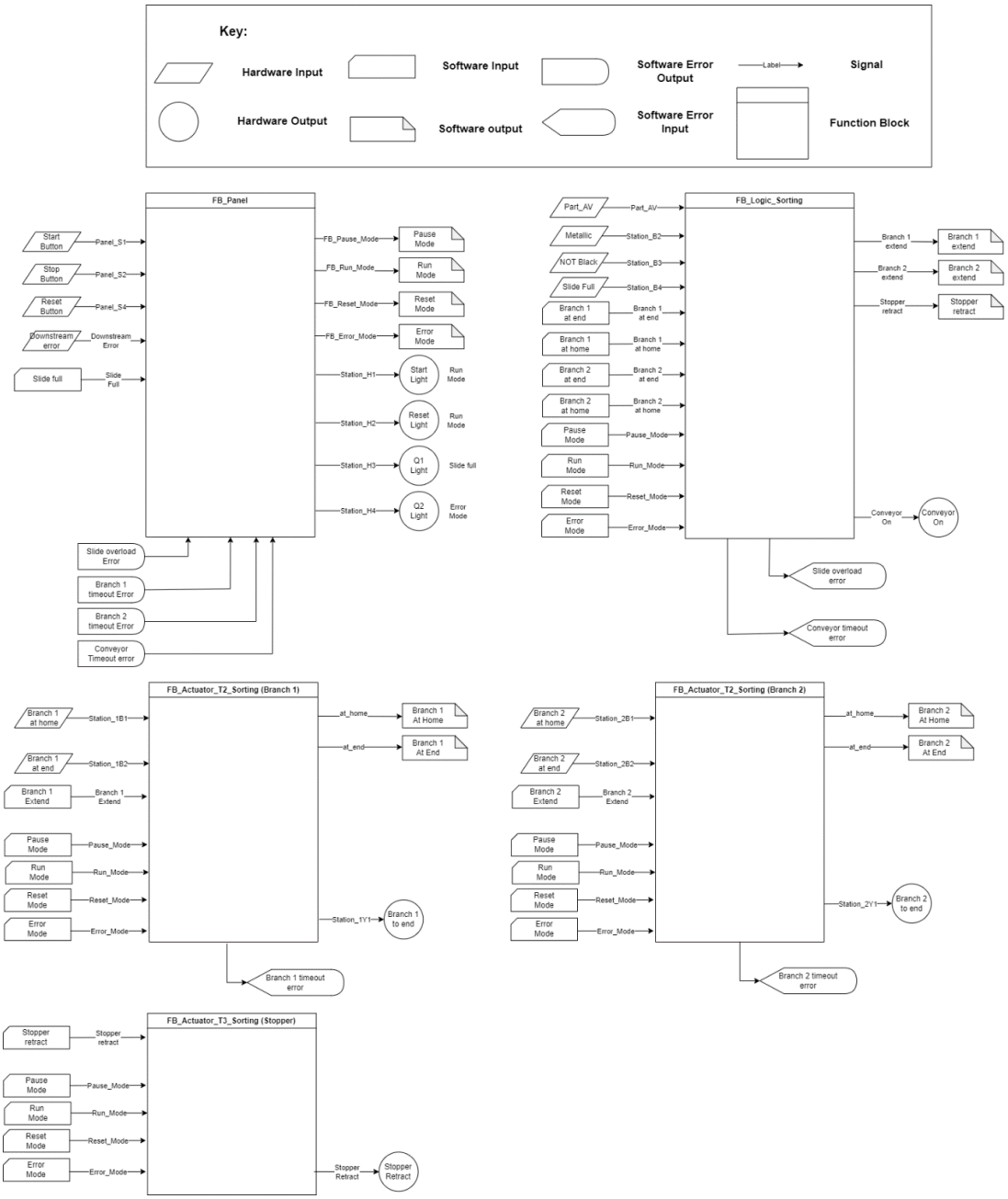


Figure 10: Sorting station function block diagram.

3.4 Station integration:

The next stage of the project required the integration of the previously developed individual station control logic into a serially chained production line super-system. Analysis and discussion of the previous station implementations highlighted certain aspects of station requirements that could be taken advantage of throughout the integration stage to help simplify the development process. It was first noticed that the control panel logic across all three stations was generic, allowing the same function block to be re-used across all three stations. It was recognized that applying this object-oriented approach to other sub-system components which exhibited similar functional and IO requirements could significantly streamline the integration process by allowing re-use of instances of

function blocks across multiple stations. It was noticed that the actuator function blocks in particular could benefit from the increased modularity this design approach enables, so the functionality of the actuators from all three stations was examined and actuators were grouped into the following three categories based on control signal requirements and functional compatibility:

- **Actuator_T1:**
    1. Distribution station swing-arm
    2. Testing station elevator

This actuator state machine was redesigned for this new implementation of the actuator_T1 logic. As the distribution station required the addition of the at-resting actuator location and related states, this logic was relocated to the distribution station Logic block and handled via the addition of the at-resting control input. A state diagram of this state machine can be seen in figure 11.

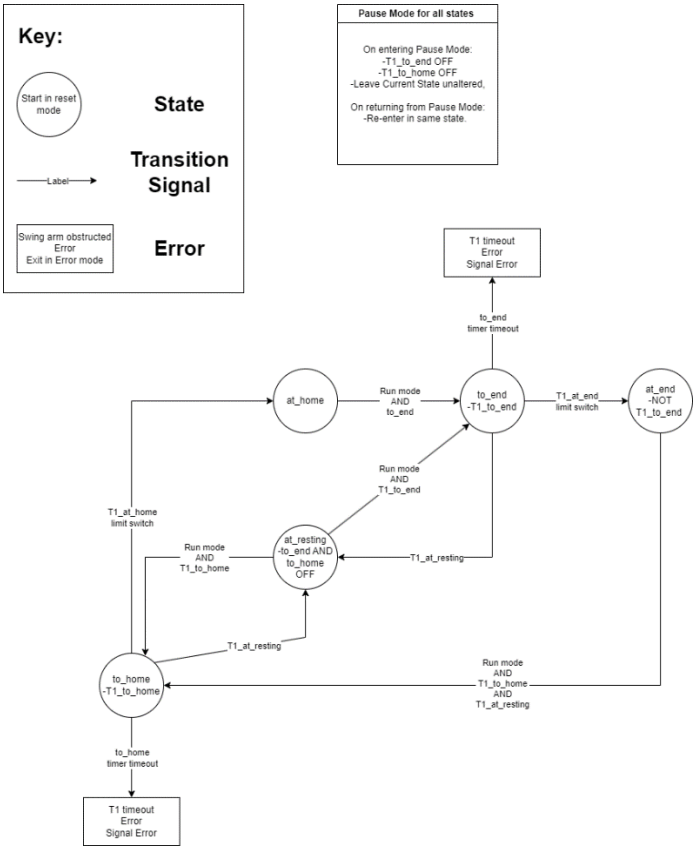


Figure 11: Actuator_T1 block state diagram.

- **Actuator_T2:**
    1. Distribution station ejector
    2. Sorting station branch 1
    3. Sorting station branch 2

An updated state diagram of this actuator_T2 state machine can be seen in figure 12.

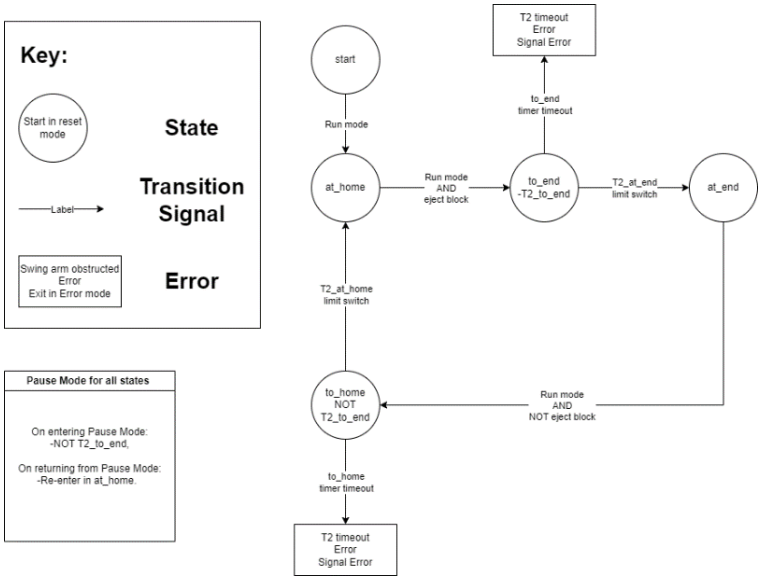


Figure 12: Actuator_T2 block state diagram.

- **Actuator_T3:**
    1. Testing station ejector
    2. Sorting station stopper

The updated state diagram for this actuator_T3 state machine can be seen in figure 13.

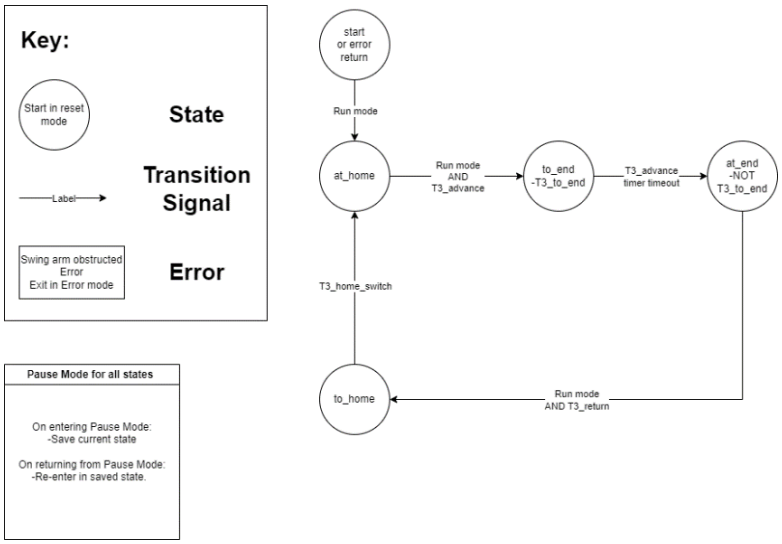


Figure 13: Actuator_T3 block state diagram.

The other major insight uncovered during this analysis was recognized during development of the testing station control logic. The approach of moving system sensor and state input data to the sensor function block and using outputs from this block as control command signals to actuator blocks allowed station specific signals to be abstracted out of the actuator blocks resulting in them becoming station non-specific (generic). It was decided to implement this approach across the stations by creating a Logic function block for each station to handle station specific signals and logic and to use this block as a “master” that handles this logic and outputs simple command signals to

the three actuator “slave” blocks and other station sub-system component blocks. Aspects of this development method began to be implemented during the sorting station development and can be seen in the function block implementation of the sorting station in figure 10 above.

It was also realized that integration of the three stations created inter-station process flow dependencies. These dependencies included:

1. The distribution station would need to halt block distribution until the testing station had handled the previous block and returned to the home position.

2. All previous stations would need to halt in the event of a station error until the error had been corrected and the station had been reset.

3. All previous stations would need to halt in the event a station was put into pause mode by the user.

4. All previous stations would need to halt if any of the block receival trays of a station reached maximum block capacity.

To deal with these dependencies, a communications protocol to allow station-to-station message passing would need to be developed. The three stations were equipped with infra-red transmitter and receiver hardware components to allow this communications system to be implemented. As these dependencies propagated from the process end point to the start point in only the reverse direction, only backwards propagating communications would be necessary as in the diagram below:

$$\|Distribution\ station\| \leftarrow data - \|Testing\ station\| \leftarrow data - \|Sorting\ station\|$$

3.4.1 Communication protocol development:

The inter-station communications protocol was developed using a binary pulse signaling system. The system uses a 2-bit wide HIGH signal to indicate the start of a message, with each required station-to-station message being defined by a unique rising edge pulse count over the constant message period. Two communication connections were required, one for testing station output to distribution station input and one for sorting station output to testing station input. The messaging system was designed to allow the end point sorting station to act as a master with user input to the control panel of this station propagating the relevant messages to the testing station, which in turn passes the message onto to the distribution station. Signal definitions of these two message protocols can be seen in the tables below.

Testing station output to distribution station input communications protocol table

Message description	Binary signal [2 start bits, 11 data bits]	Rising edge Pulse count	Reason for Implementation
Downstream station start button pressed	[1,1,0,1,0,0,0,0,0,0,0,0,0]	1	Signals downstream station start button pressed, distribution station to enter run mode
Downstream station stop button pressed	[1,1,0,1,0,1,0,0,0,0,0,0,0]	2	Signals downstream station stop button pressed, distribution station to enter pause mode
Downstream station reset button pressed	[1,1,0,1,0,1,0,1,0,0,0,0,0]	3	Signals downstream station reset button pressed, distribution station to enter pause mode
Downstream station error	[1,1,0,1,0,1,0,1,0,1,0,0,0]	4	Signals an error has occurred on a downstream station, distribution station to enter pause mode
Testing station ready to receive part	[1,1,0,1,0,1,0,1,0,1,0,1,0]	5	Signals testing station is ready for next block, distribution station can continue operation.

Table 4: Testing to distribution station communications protocol.

Sorting station output to testing station input communications protocol table

Message description	Binary signal [2 start bits, 11 data bits]	Rising edge Pulse count	Reason for Implementation
Sorting station start button pressed	[1,1,0,1,0,0,0,0,0,0,0,0,0]	1	Signals sorting station start button pressed, testing station to enter run mode, transmit message on to distribution station
Sorting station stop button pressed	[1,1,0,1,0,1,0,0,0,0,0,0,0]	2	Signals sorting station stop button pressed, testing station to enter pause mode, transmit message on to distribution station
Sorting station reset button pressed	[1,1,0,1,0,1,0,1,0,0,0,0,0]	3	Signals sorting station reset button pressed, testing station to enter pause mode, transmit message on to distribution station
Sorting station error	[1,1,0,1,0,1,0,1,0,1,0,0,0]	4	Signals an error has occurred on sorting station, testing station to enter pause mode, transmit message on to distribution station

Table 5: Sorting to testing station communications protocol.

3.4.2 Distribution station integration:

Integration of the distribution station required the addition of a communications Receiver function block to decode and process input messages received from the testing station transmitter. A Logic function block was then added to handle station control logic and output control signals to the updated Actuator_T1 (swing-arm), and Actuator_T2 (ejector) generic function blocks, and the end effector vacuum function block.

The updated distribution station function block diagram after this integration stage was implemented can be seen in figure 14.

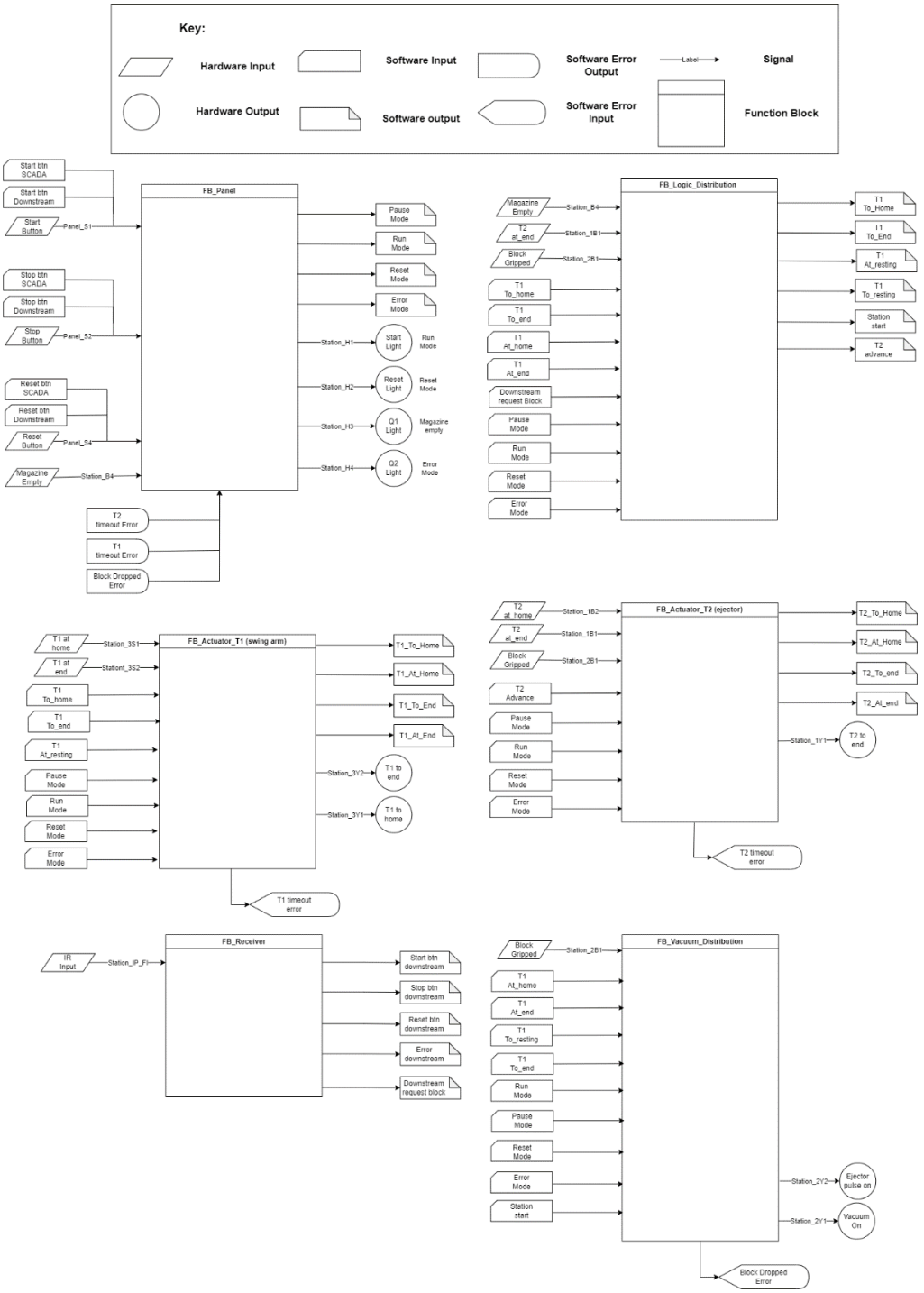


Figure 14: Distribution station integrated function block diagram.

3.4.3 Testing station integration:

Integration of the testing station required the addition of two communications blocks with both a receiver (from sorting station) and transmitter (to distributing station) function block required. This station must act locally on incoming messages as well as relay some of these messages further down the chain to the distribution station. To enable this functionality, messages must be able to be passed from the receiver function block to the transmitter function block. The original Sensor block

was replaced with a Logic block which retained the original functionality of the Sensor block with the addition of incoming and outgoing message handling between the receive and transmit blocks and control output signals for the updated Actuator_T1 (elevator) and Actuator_T3 (ejector) generic function blocks.

The updated testing station function block diagram after this integration stage was implemented can be seen in figure 15.

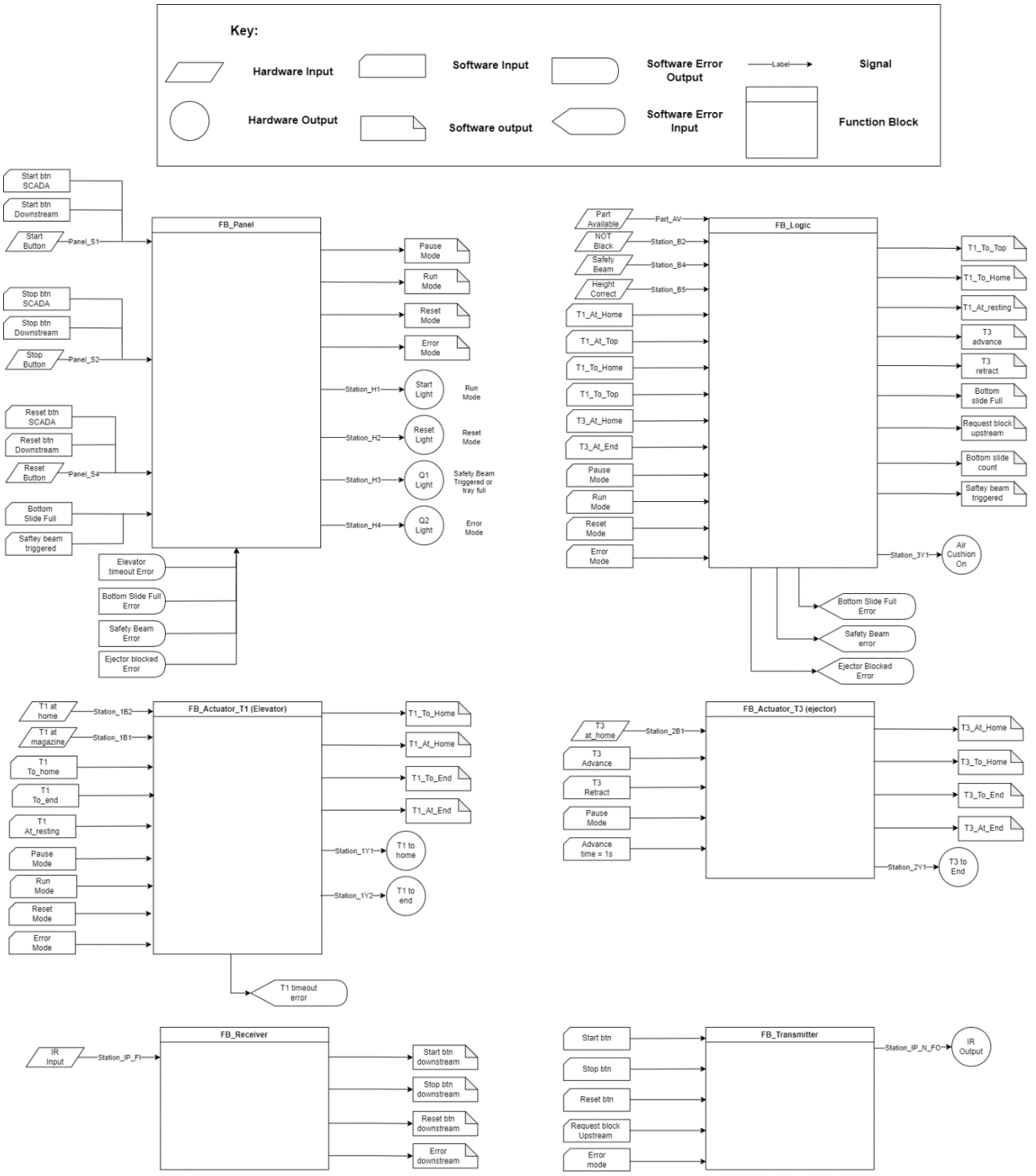


Figure 15: Testing station integrated function block diagram.

3.4.4 Sorting station integration:

The sorting station integration was again implemented with the replacement of the previous branch 1 and 2 actuator blocks with instances of the updated Actuator_T2 generic block and the replacement of the stopper actuator block with an instance of the updated Actuator_T3 generic block. A transmitter block was again added to enable message passing to the testing station. As the initial sorting station implementation already incorporated the master Logic block design methodology for station control, this block required only minimal alterations be put in place for the message transmitting system logic. A function block diagram of this system can be seen in figure 16.

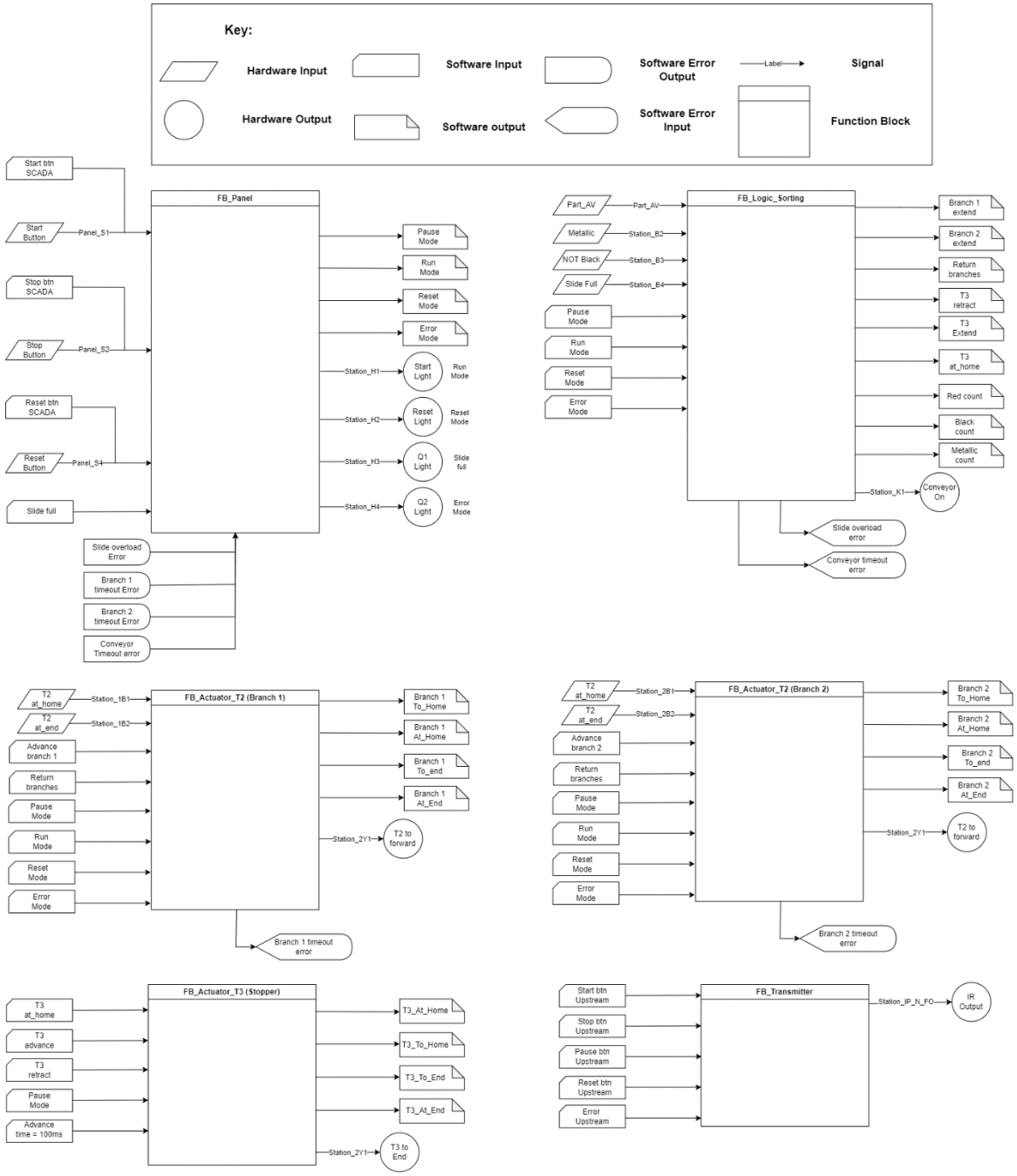


Figure 16: Sorting station integrated function block diagram.

3.4.5 Station Integration challenges and solutions:

One of the major challenges of developing the ladder logic was designing the communications protocol between stations. The proposed protocol was to encode the desired message in the number of pulses sent to the upstream station (a message of one pulse for START, two pulses for STOP, three pulses for RESET, four pulses for ERROR, and five pulses for REQUEST PIECE). This was achieved by first sending out a single pulse, no matter the message; then, if the desired message was not START, another pulse would be emitted; if the message was not STOP, a third pulse would be generated, and so on until a maximum of five pulses would be sent if none of the messages but REQUEST PIECE was desired.

There were two challenges in getting the communication system to function. The first was efficiently programming the sequence of pulses. Unlike structured text, ladder logic is not sequential in nature, hence all rungs in the ladder diagram are executed in parallel. This makes it difficult to program a sequence of operations. This obstacle was overcome by first modularizing the pulse generation with a custom function block that takes in pulse duty cycle and period, and then ordering these blocks in such a way that after each pulse was generated the rung would trigger a flag. The next rung containing a pulse generator would only activate if this flag was true. In this way, each rung would only execute its pulse if the previous rung had already finished generating its own.

The second challenge was in finding the right pulse length for the communications. The Infrared sensors that the stations are equipped with have a significant delay between the transmission from one to the reception in the next. This made it so that with smaller pulse durations, some information could be lost and the wrong message could be delivered. With larger pulse durations however, there was too high a delay between messages and so the stations would not be perfectly in sync. Through much experimentation and trial and error, the optimal pulse on time of 20ms (40ms period) was found, which resulted in the right balance between data preservation and communication speed.

One major oversight in the programming of the stations was that for the distributing and testing stations, the actuator function blocks were not initially programmed with modularization in mind. The ladder diagrams were very specialized, taking inputs from sensors which were specific to the particular station and implementing logic that was only relevant to that station. It was identified that this would make the integration of all three stations more difficult later down the line as function blocks wouldn't be reusable, and so before moving on to the sorting station, a great deal of time was spent in generalizing the logic for the first two stations. This was done by extracting station-specific logic from the actuator function blocks and placing them in a separate "Logic" function block, leaving behind only the essential and generic logic such as responding to action signals and error handling within the actuator block. The "Logic" block would make decisions for the actuators based on sensor inputs and send the required action to the relevant actuator function block. This process of generalizing the function blocks was very time consuming, and so for future projects it is recommended to begin development of the ladder diagrams with modularization in mind and to make the blocks as generic as possible from the beginning.

3.5 SCADA Human Machine Interface (HMI) implementation:

To allow for PC based system monitoring and control of the three stations a UI for each station was created using the Ignition SCADA development IDE. This process began with a UI layout diagram

being drawn up showing all required UI elements. All required PLC variables from the PLC control software for the three stations were exported from Sysmac studio and imported into Ignition for use in station signal monitoring. All error state signals were exported as individual variables to allow for fast diagnosis of system errors directly from the user interface, along with the block count variables for the testing station rejected tray and the three sorting station block color output trays. To enable PC control of the stations from the user interface, variables with read/write capabilities were added to both the PLC ladder network and SCADA interface for the start, stop and reset control panel buttons. Diagrams of the UI layout for the three stations are detailed below.

3.5.1 Distribution station SCADA UI implementation:

Once the additional SCADA button variables and type specific error variables were added to the distribution station ladder logic a plan for the UI was developed. This plan can be seen in figure 17.

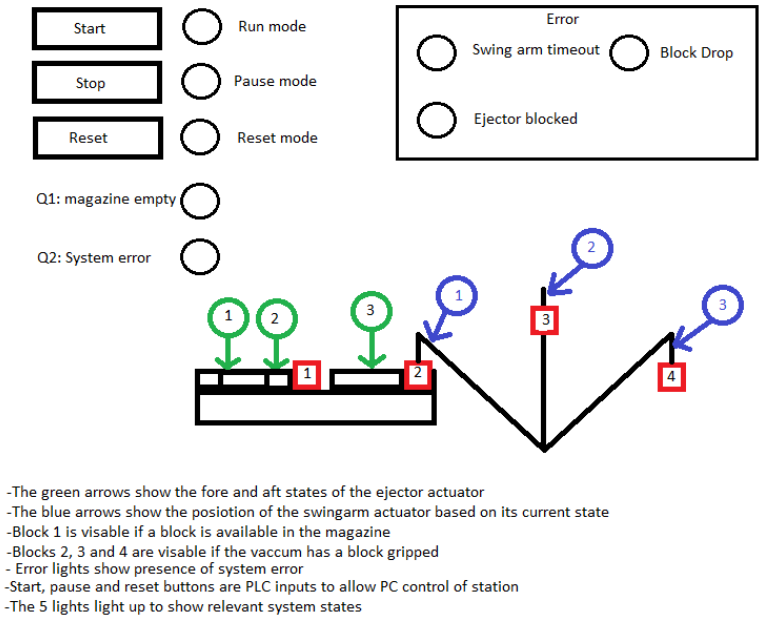


Figure 17: Distribution station SCADA diagram.

Next the plan was implemented in Ignition with all required UI elements being drawn up. The three button tags were set up with both read and write functionality and directly bound to the control data for the associated UI button. The station master states of run-mode, pause-mode, reset-mode, and error-mode, the three type-specific error tags, and the magazine-empty tag were directly bound to the fill color property of the associated indicator circles with appropriate colors set to indicate value changes of these variables. The station hardware was then animated through all actuator positions and block locations using values of the actuator and Logic function block output tags to set the visible property of each required animation element using expression bindings containing if statements with conditions based on the relevant tag/tags. Once this was completed the system was tested alongside the PLC software on the station hardware to ensure correct functionality.

3.5.2 Testing station SCADA UI implementation:

The required additional button variables, type specific error variables and the rejected tray count variable were added to the testing station ladder logic and a plan for the UI was once again developed. This plan can be seen in figure 18.

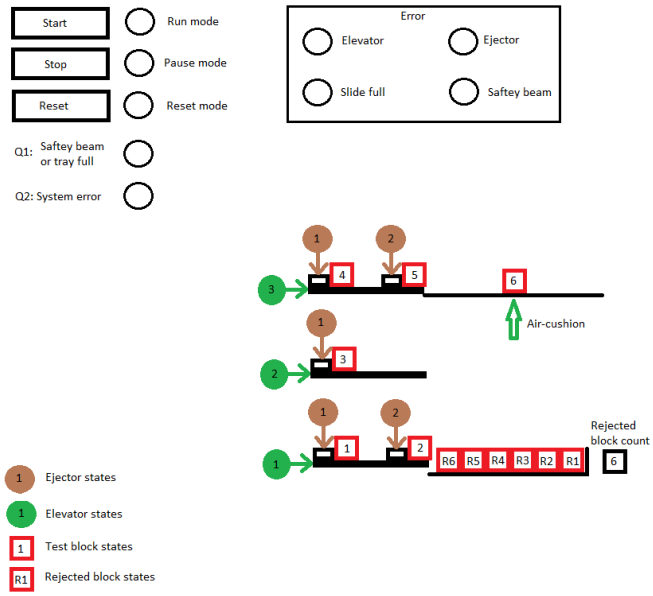


Figure 18: Testing station SCADA diagram.

The UI elements were drawn up in the Ignition IDE and the buttons, indicator lights, and animation elements were then implemented as in the previously discussed distribution station UI with the major difference being the dependance on the current elevator state of the ejector state. To take advantage of this dependency, only two ejector states were required with the current elevator state being used as an argument to the control binding if condition to find which ejector instance should be visible. The state of the air cushion at the output tray was indicated via the visibility of the green air-cushion arrow. The integer block count tag of the rejected tray was then used to as an argument to internal if statement conditions checked against integer constants in the visible property expression bindings for the rejected blocks R1 through R6 to animate the appropriate number of visible blocks in the tray and display this count in the rejected counter.

3.5.3 Sorting station SCADA UI implementation:

The sorting station required the same additional variables as the distribution station to be added along with additional integer variables for the red, black, and metallic output tray counts. A plan for the UI was then created. This plan can be seen in figure 19.

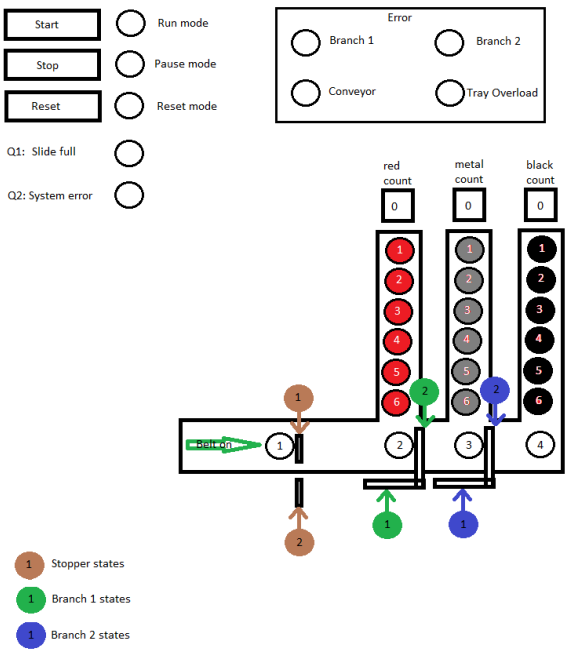


Figure 19: Sorting station SCADA diagram.

This UI was created in much the same way as the testing station UI with the addition of the two extra integer tags for the additional block output trays and output tray counters.

3.5.4 SCADA development challenges and solutions:

The greatest challenge faced during development of the SCADA UI software for the three stations was ensuring the latest SCADA and PLC software versions were compatible with each other as the parallel development of both systems progressed. PLC variables changed quite often during the integration phase of the project leading to SCADA tags becoming obsolete and new tags needing to be created. Every time this happened the SCADA side would require a fresh export/import of the PLC variables which lead to previously developed SCADA logic becoming non-functional due to missing or incorrectly named tags.

This issue was resolved through delaying the development of the SCADA control logic by prioritizing lab time to focus first on getting all three SCADA UIs drawn up in the Ignition IDE. Once this task was completed the latest versions of the PLC software for all three stations had the variable tables analyzed by the group members and a final list of required variables and variable names was decided upon to keep future software iterations compatible on both the SCADA and PLC sides. This allowed development of the SCADA logic and PLC integration stages to be completed in parallel with minimal disruptions to either aspect of the project.

4. Conclusion:

The project detailed in this report was completed successfully with all required functionalities being implemented, tested and ready for use by the project deadline. The creation and updating of the project Gantt chart at each task completion point allowed the group to constantly be aware of the time constraints the project was under at each important milestone of the project. The analysis of the first iteration of the station ladder logic programs undertaken before the integration stage of the project began was essential in deciding what methods should be preferred for the final tasks of the project. The use of OOP based generic function blocks to improve modularity wherever possible throughout the ladder network development combined with the abstraction provided by the creation of the master Logic function blocks for all stations greatly reduced the development time during the integration and SCADA development phases of the project. It would be highly recommended to begin the project with function block modularity in mind right from the beginning to assist in minimizing the required iterations of function block creation during the integration phase. Another method employed by the group that proved beneficial to timely task completion was the division of task responsibilities of group members between the station integration and SCADA development tasks towards the end of the project. This task division allowed these aspects of the project to be performed in parallel in the final weeks leading up to the deadline which helped achieve project completion within this development time window.

Bibliography

CIROS Mechatronics. (n.d.). FESTO MPS-C Distibution station user model.

CIROS Mechatronics. (n.d.). FESTO MPS-C Sorting station user model.

CIROS Mechatronics. (n.d.). FESTO MPS-C Testing station user model.

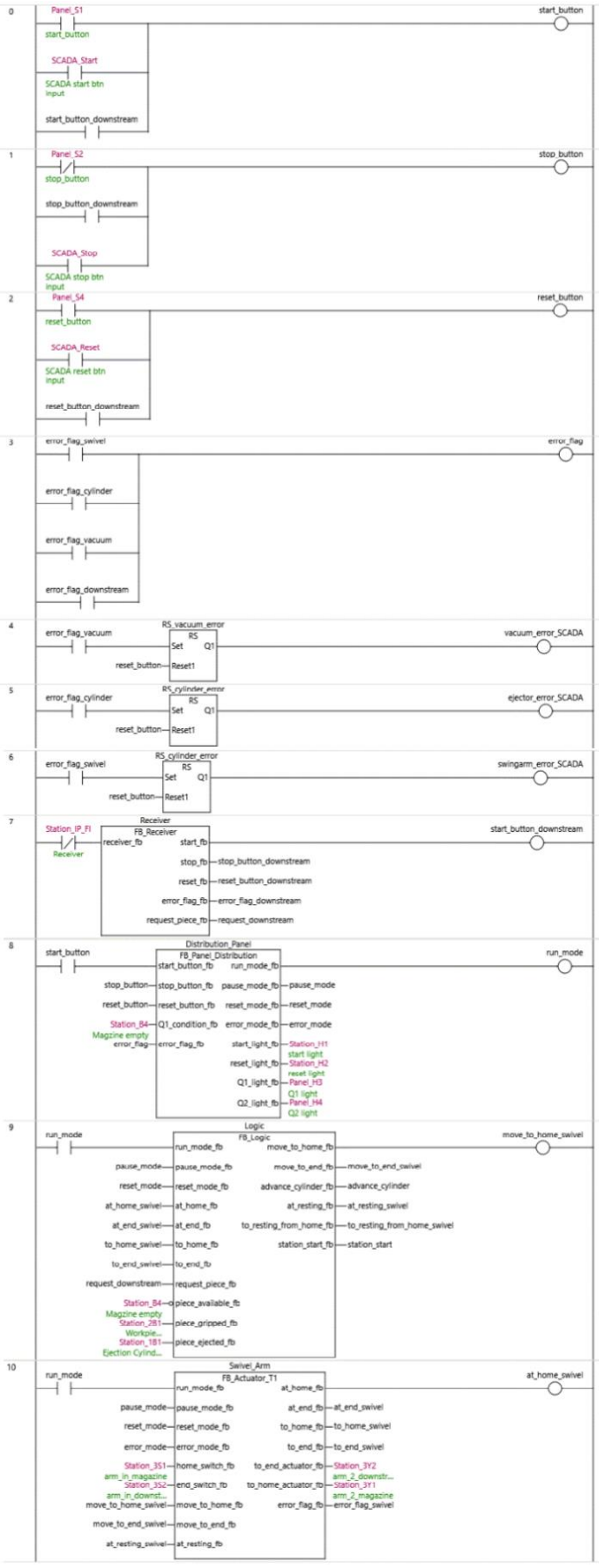
Cui, D. L. (2023). *Supervisory Control and Data Acquisition.* Curtin University, School of Civil & Mechanical Engineering.

Curtin University. (2023). Sysmac Studio Quick Start Guide.

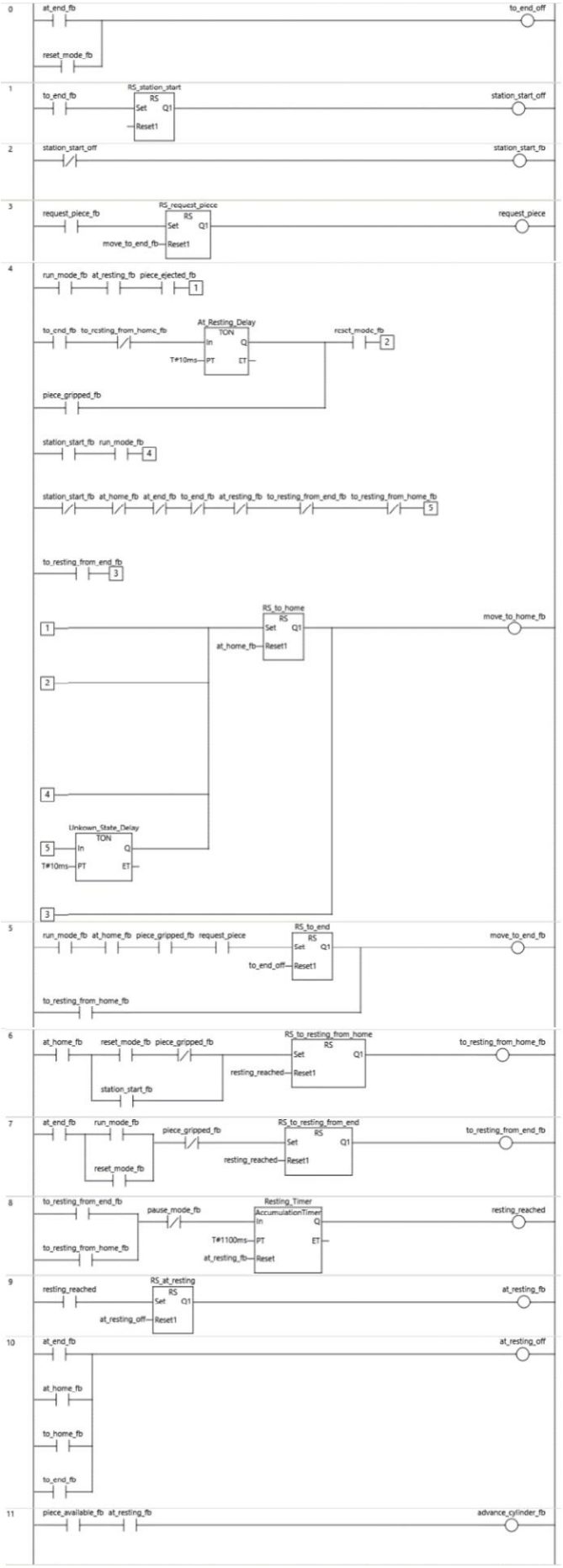
Appendix:

A1: Distribution station function block ladder networks:

A1.1: Program 0:

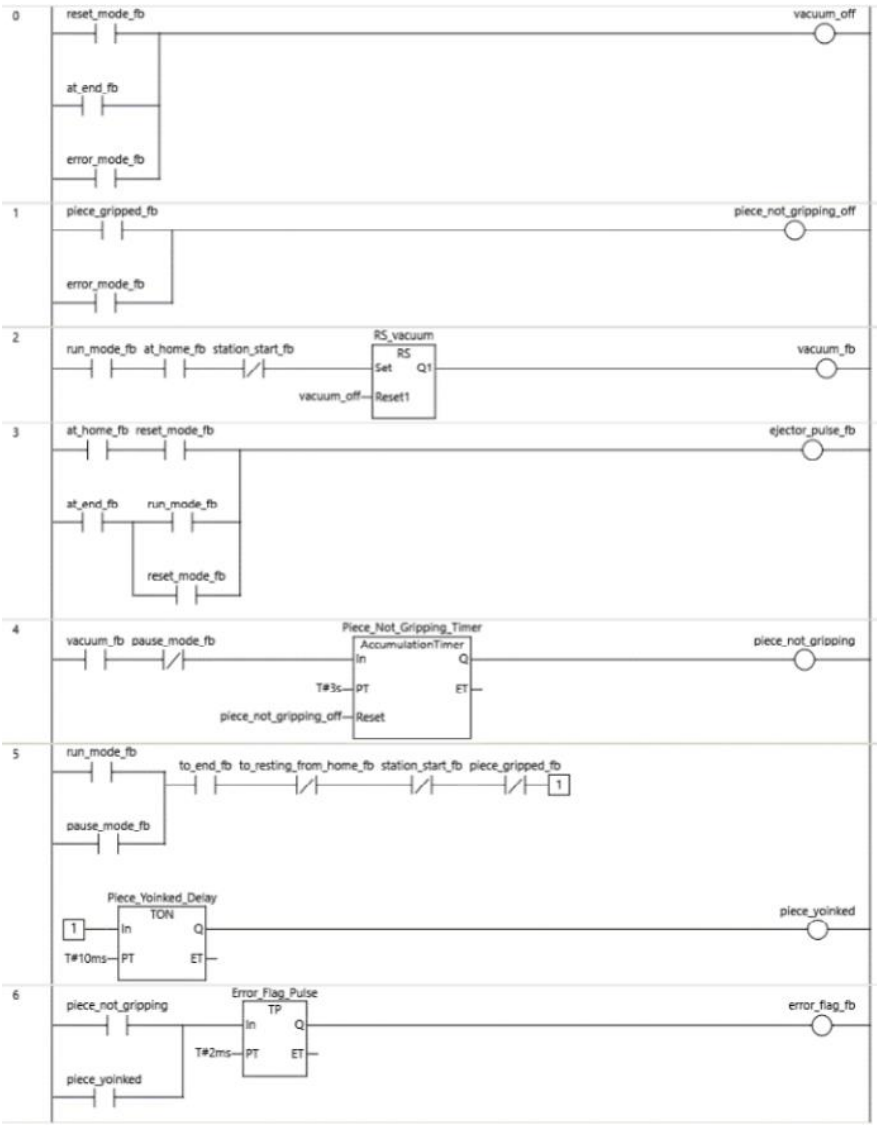


A1.2: Logic:


0
at_end_fb
reset_mode_fb
to_end_off
1
to_end_fb
RS_station_start
RS
Set
Q1
Reset1
station_start_off
2
station_start_off
station_start_fb
3
request_piece_fb
RS_request_piece
move_to_end_fb
request_piece
4
run_mode_fb at_resting_fb piece_ejected_fb
to_cnd_fb to_resting_from_home_fb
At_Resting_Delay
TON
In
Q
T#10ms
PT
ET
reset_mode_fb
piece_gripped_fb
station_start_fb run_mode_fb
station_start_fb at_home_fb at_end_fb to_end_fb at_resting_fb to_resting_from_end_fb to_resting_from_home_fb
to_resting_from_end_fb
5
RS_to_home
move_to_home_fb
at_home_fb
Unknown_State_Delay
run_mode_fb at_home_fb piece_gripped_fb request_piece
RS_to_end
move_to_end_fb
to_end_off
to_resting_from_home_fb
6
at_home_fb
reset_mode_fb piece_gripped_fb
RS_to_resting_from_home
to_resting_from_home_fb
resting_reached
station_start_fb
7
at_end_fb
run_mode_fb
piece_gripped_fb
RS_to_resting_from_end
to_resting_from_end_fb
resting_reached
reset_mode_fb
8
to_resting_from_end_fb
pause_mode_fb
Resting_Timer
AccumulationTimer
resting_reached
T#1100ms
at_resting_fb
Reset
to_resting_from_home_fb
9
resting_reached
RS_at_resting
at_resting_fb
at_resting_off
10
at_end_fb
at_resting_off
at_home_fb
to_home_fb
to_end_fb
11
piece_available_fb at_resting_fb
advance_cylinder_fb

B

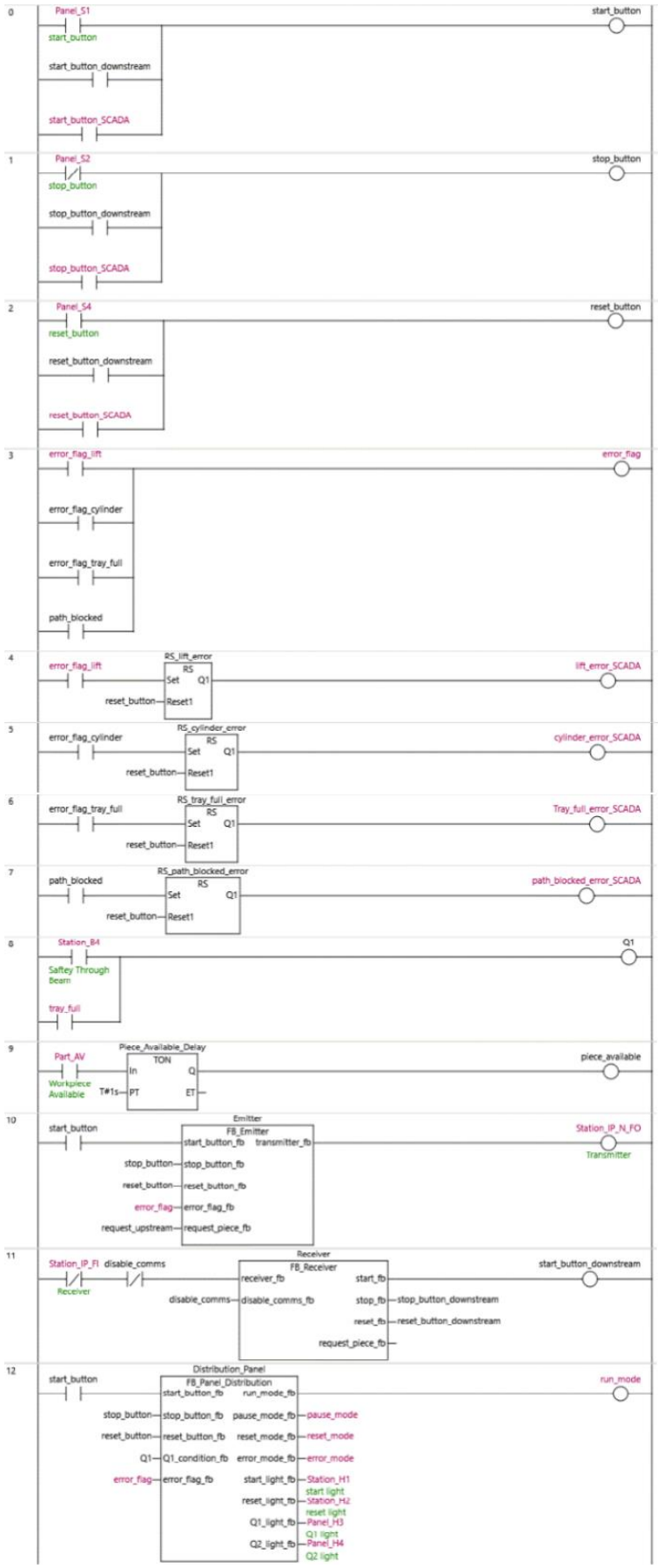
A1.3: Vacuum:


0
reset_mode_fb
at_end_fb
error_mode_fb
vacuum_off
1
piece_gripped_fb
error_mode_fb
piece_not_gripping_off
2
run_mode_fb at_home_fb station_start_fb
RS_vacuum
RS
Set
Q1
vacuum_off
Reset1
vacuum_fb
3
at_home_fb reset_mode_fb
at_end_fb
run_mode_fb
reset_mode_fb
ejector_pulse_fb
4
vacuum_fb pause_mode_fb
Piece_Not_Gripping_Timer
AccumulationTimer
In
Q
T#3s
PT
ET
piece_not_gripping_off
Reset
piece_not_gripping
5
run_mode_fb
to_end_fb to_resting_from_home_fb station_start_fb piece_gripped_fb
1
pause_mode_fb
Piece_Yoinked_Delay
TON
1
In
Q
T#10ms
PT
ET
piece_yoinked
6
piece_not_gripping
Error_Flag_Pulse
TP
In
Q
T#2ms
PT
ET
piece_yoinked
error_flag_fb

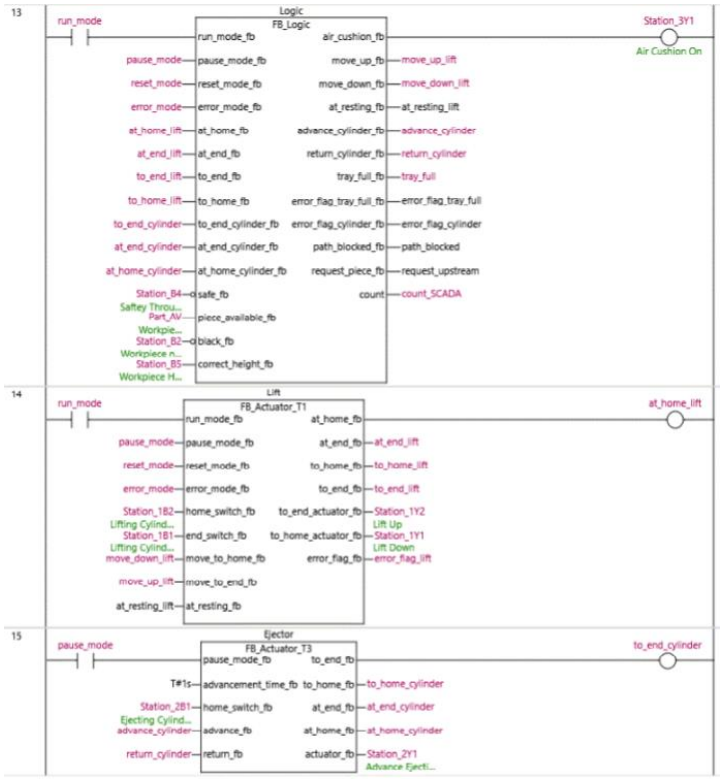
C

A2: Testing station function block ladder networks:

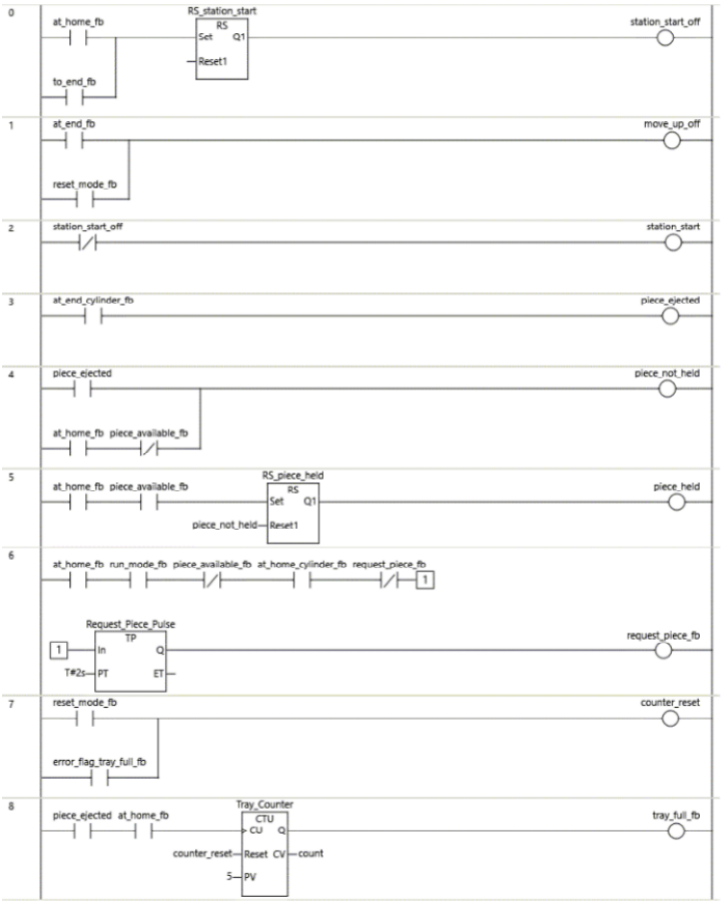
A2.1: Program 0:



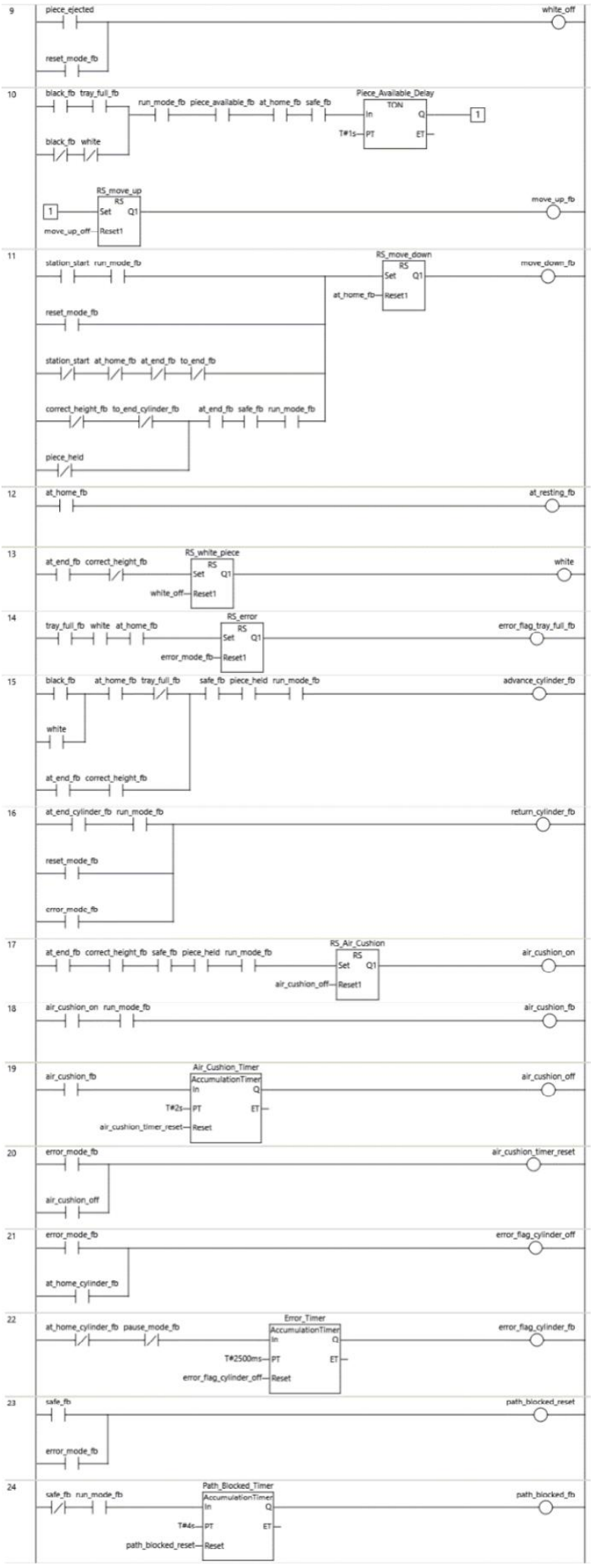
D


13
run_mode
Logic
FB_Logic
run_mode_fb
air_cushion_fb
Station_3Y1
Air Cushion On
pause_mode
pause_mode_fb
move_up_fb
move_up_lift
reset_mode
reset_mode_fb
move_down_fb
move_down_lift
error_mode
error_mode_fb
at_resting_fb
at_resting_lift
at_home_lift
at_home_fb
advance_cylinder_fb
advance_cylinder
at_end_lift
at_end_fb
return_cylinder_fb
return_cylinder
to_end_lift
to_end_fb
tray_full_fb
tray_full
to_home_lift
to_home_fb
error_flag_tray_full_fb
error_flag_tray_full
to_end_cylinder
to_end_cylinder_fb
error_flag_cylinder_fb
error_flag_cylinder
at_end_cylinder
at_end_cylinder_fb
path_blocked_fb
path_blocked
at_home_cylinder
at_home_cylinder_fb
request_piece_fb
request_upstream
Station_B4
safe_fb
count
count_SCADA
Saftey Throu...
Part_AV
piece_available_fb
Workpie...
Station_B2
black_fb
Workpiece n...
Station_B5
correct_height_fb
Workpiece H...
14
run_mode
Lift
FB_Actuator_T1
run_mode_fb
at_home_fb
at_home_lift
pause_mode
pause_mode_fb
at_end_fb
at_end_lift
reset_mode
reset_mode_fb
to_home_fb
to_home_lift
error_mode
error_mode_fb
to_end_fb
to_end_lift
Station_1B2
home_switch_fb
to_end_actuator_fb
Station_1Y2
Lifting Cylind...
Lift Up
Station_1B1
end_switch_fb
to_home_actuator_fb
Station_1Y1
Lifting Cylind...
Lift Down
move_down_lift
move_to_home_fb
error_flag_fb
error_flag_lift
move_up_lift
move_to_end_fb
at_resting_lift
at_resting_fb
15
pause_mode
Ejector
FB_Actuator_T3
pause_mode_fb
to_end_fb
to_end_cylinder
T#1s
advancement_time_fb
to_home_fb
to_home_cylinder
Station_2B1
home_switch_fb
at_end_fb
at_end_cylinder
Ejecting Cylind...
advance_cylinder
advance_fb
at_home_fb
at_home_cylinder
return_cylinder
return_fb
actuator_fb
Station_2Y1
Advance Ejecti...

A2.2: Logic:


0
at_home_fb
RS_station_start
RS
Set
Q1
Reset1
station_start_off
to_end_fb
1
at_end_fb
move_up_off
reset_mode_fb
2
station_start_off
station_start
3
at_end_cylinder_fb
piece_ejected
4
piece_ejected
piece_not_held
at_home_fb
piece_available_fb
5
at_home_fb
piece_available_fb
RS_piece_held
RS
Set
Q1
piece_held
piece_not_held
Reset1
6
at_home_fb
run_mode_fb
piece_available_fb
at_home_cylinder_fb
request_piece_fb
1
Request_Piece_Pulse
TP
1
In
Q
T#2s
PT
ET
request_piece_fb
7
reset_mode_fb
counter_reset
error_flag_tray_full_fb
8
piece_ejected
at_home_fb
Tray_Counter
CTU
CU
Q
counter_reset
Reset
CV
count
5
PV
tray_full_fb

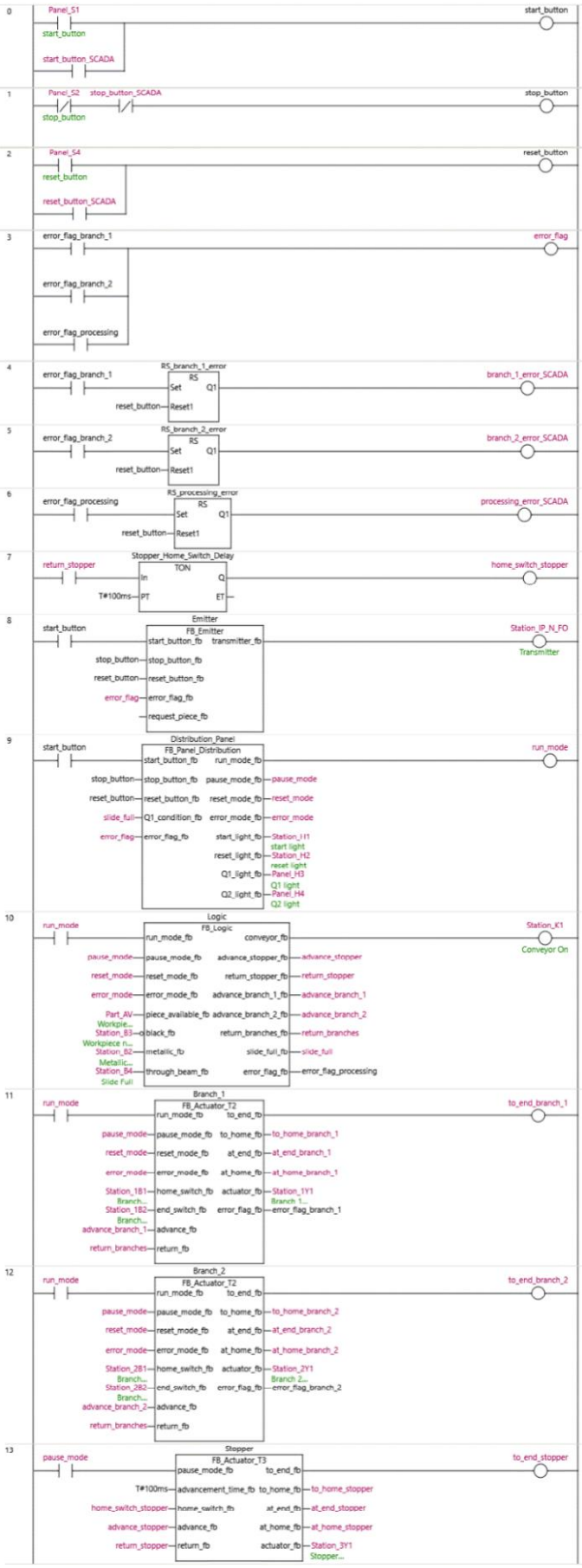
E


9
piece_ejected
white_off
reset_mode_fb
10
black_fb tray_full_fb
run_mode_fb piece_available_fb at_home_fb safe_fb
Piece_Available_Delay
TON
In Q
T#1s PT ET
1
black_fb white
RS_move_up
RS
1
Set Q1
move_up_off Reset1
move_up_fb
11
station_start run_mode_fb
RS_move_down
RS
Set Q1
at_home_fb Reset1
move_down_fb
reset_mode_fb
station_start at_home_fb at_end_fb to_end_fb
correct_height_fb to_end_cylinder_fb
at_end_fb safe_fb run_mode_fb
piece_held
12
at_home_fb
at_resting_fb
13
at_end_fb correct_height_fb
RS_white_piece
RS
Set Q1
white_off Reset1
white
14
tray_full_fb white at_home_fb
RS_error
RS
Set Q1
error_mode_fb Reset1
error_flag_tray_full_fb
15
black_fb
at_home_fb tray_full_fb
safe_fb piece_held run_mode_fb
advance_cylinder_fb
white
at_end_fb correct_height_fb
16
at_end_cylinder_fb run_mode_fb
return_cylinder_fb
reset_mode_fb
error_mode_fb
17
at_end_fb correct_height_fb safe_fb piece_held run_mode_fb
RS_Air_Cushion
RS
Set Q1
air_cushion_off Reset1
air_cushion_on
18
air_cushion_on run_mode_fb
air_cushion_fb
19
air_cushion_fb
Air_Cushion_Timer
AccumulationTimer
In Q
T#2s PT ET
air_cushion_timer_reset Reset
air_cushion_off
20
error_mode_fb
air_cushion_timer_reset
air_cushion_off
21
error_mode_fb
error_flag_cylinder_off
at_home_cylinder_fb
22
at_home_cylinder_fb pause_mode_fb
Error_Timer
AccumulationTimer
In Q
T#2500ms PT ET
error_flag_cylinder_off Reset
error_flag_cylinder_fb
23
safe_fb
path_blocked_reset
error_mode_fb
24
safe_fb run_mode_fb
Path_Blocked_Timer
AccumulationTimer
In Q
T#4s PT ET
path_blocked_reset Reset
path_blocked_fb

F

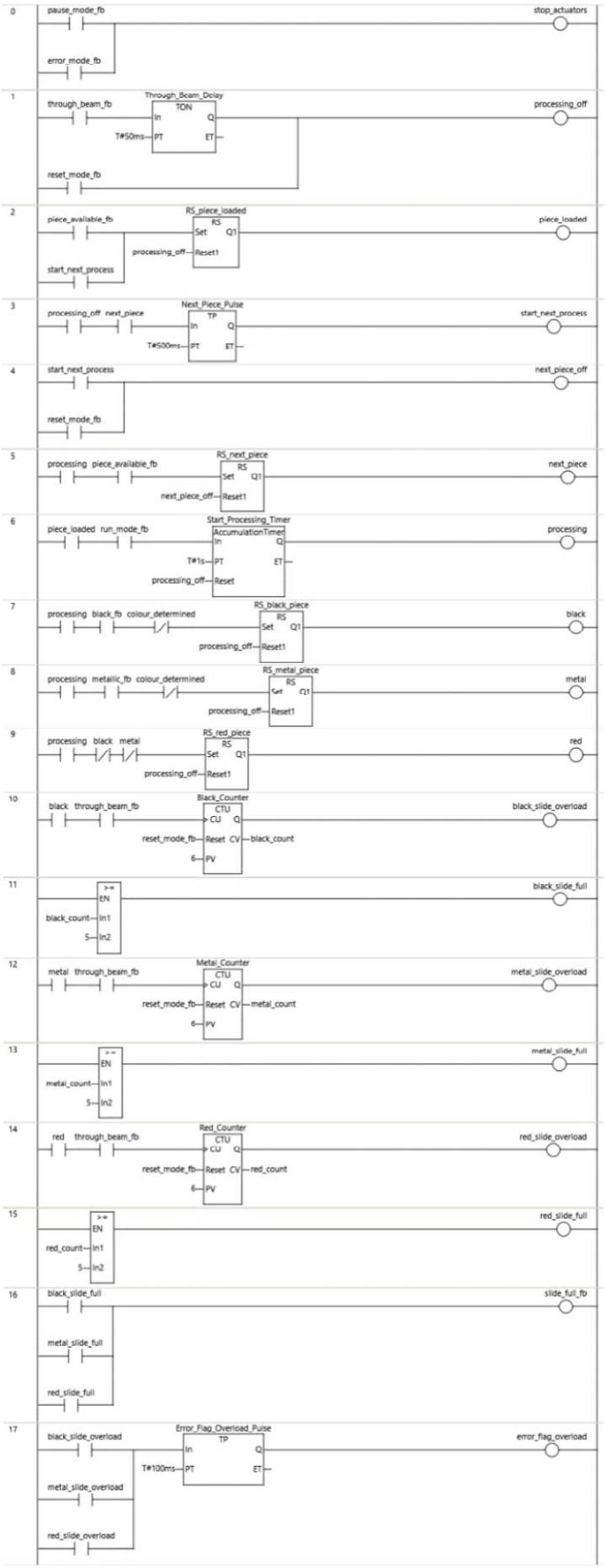
A3: Sorting station function block ladder networks:

A3.1: Program 0:

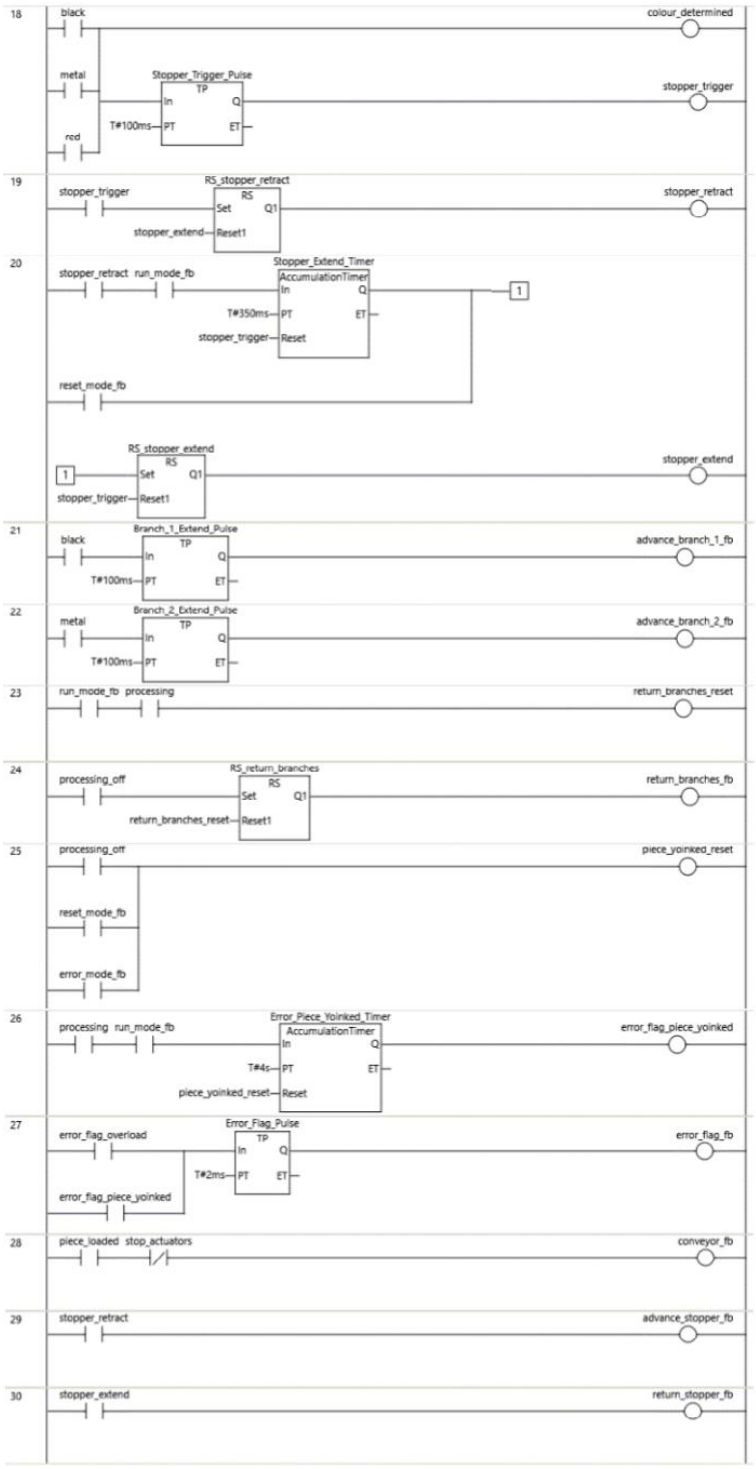


G

A3.2: Logic:

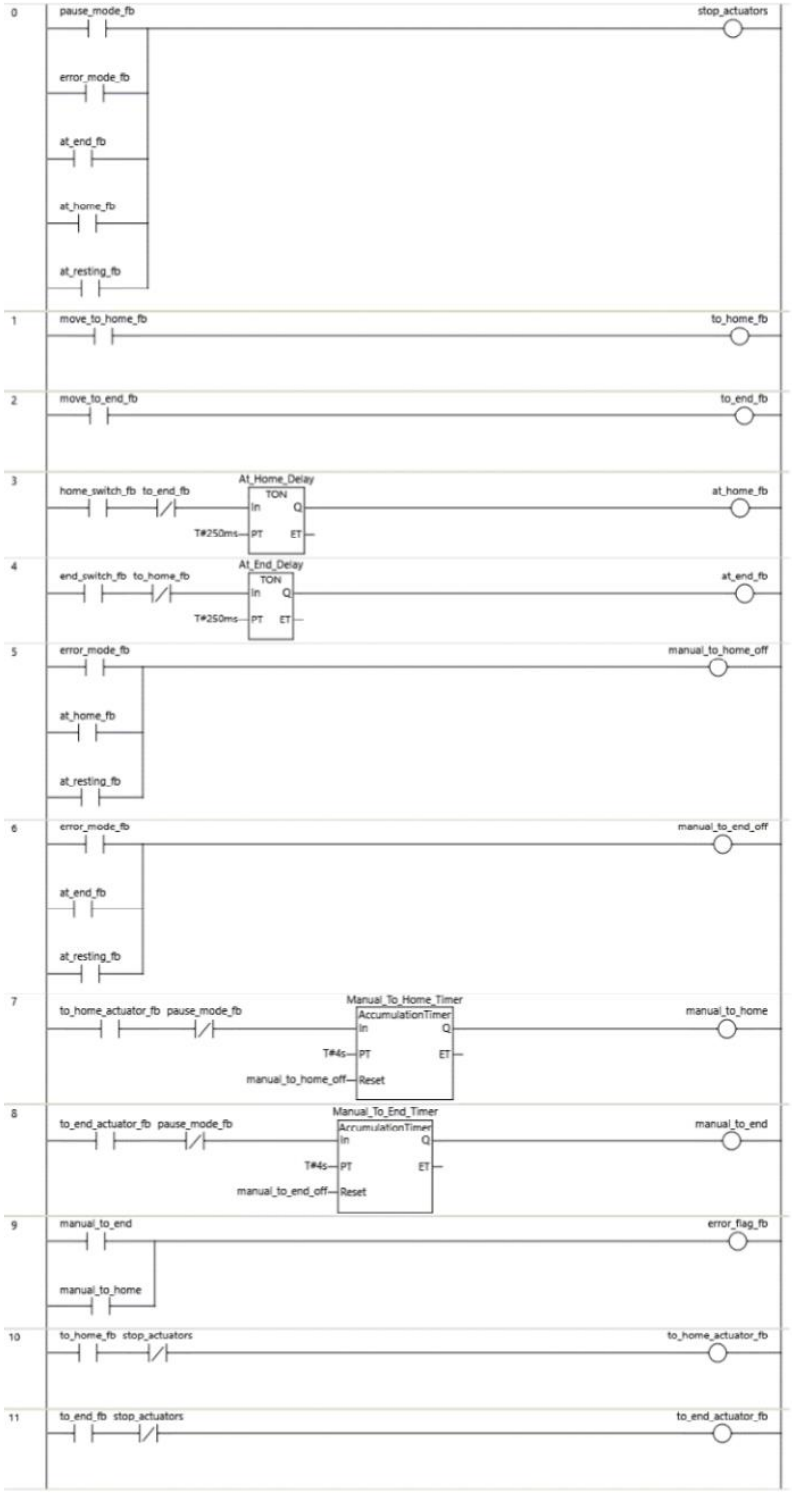

0
pause_mode_fb
error_mode_fb
stop_actuators
1
through_beam_fb
Through_Beam_Delay
TON
In
Q
T#50ms
PT
ET
reset_mode_fb
processing_off
2
piece_available_fb
RS_piece_loaded
RS
Set
Q1
Reset1
start_next_process
piece_loaded
3
processing_off
next_piece
Next_Piece_Pulse
TP
T#500ms
start_next_process
4
reset_mode_fb
next_piece_off
5
processing
piece_available_fb
RS_next_piece
next_piece
6
piece_loaded
run_mode_fb
Start_Processing_Timer
AccumulationTimer
T#1s
processing
7
black_fb
colour_determined
RS_black_piece
black
8
metallic_fb
RS_metal_piece
metal
9
black
metal
RS_red_piece
red
10
black_slide_overload
Black_Counter
CTU
CU
Reset
CV
black_count
6
PV
11
black_slide_full
>=
EN
In1
In2
5
12
metal_slide_overload
Metal_Counter
metal_count
13
metal_slide_full
14
red_slide_overload
Red_Counter
red_count
15
red_slide_full
16
slide_full_fb
17
error_flag_overload
Error_Flag_Overload_Pulse
T#100ms

H

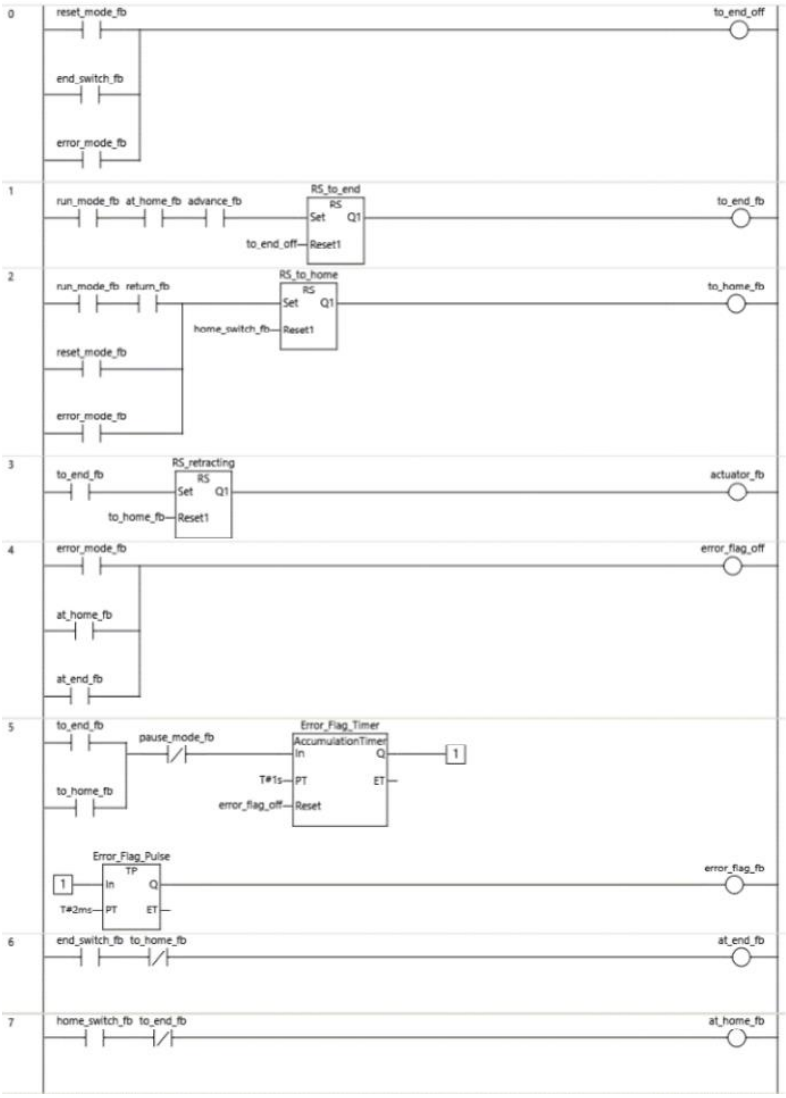

18
black
colour_determined
metal
Stopper_Trigger_Pulse
TP
In
Q
stopper_trigger
T#100ms
PT
ET
red
19
stopper_trigger
RS_stopper_retract
RS
Set
Q1
stopper_retract
stopper_extend
Reset1
20
stopper_retract
run_mode_fb
Stopper_Extend_Timer
AccumulationTimer
In
Q
1
T#350ms
PT
ET
stopper_trigger
Reset
reset_mode_fb
RS_stopper_extend
RS
1
Set
Q1
stopper_extend
stopper_trigger
Reset1
21
black
Branch_1_Extend_Pulse
TP
In
Q
advance_branch_1_fb
T#100ms
PT
ET
22
metal
Branch_2_Extend_Pulse
TP
In
Q
advance_branch_2_fb
T#100ms
PT
ET
23
run_mode_fb
processing
return_branches_reset
24
processing_off
RS_return_branches
RS
Set
Q1
return_branches_fb
return_branches_reset
Reset1
25
processing_off
piece_yoinked_reset
reset_mode_fb
error_mode_fb
26
processing
run_mode_fb
Error_Piece_Yoinked_Timer
AccumulationTimer
In
Q
error_flag_piece_yoinked
T#4s
PT
ET
piece_yoinked_reset
Reset
27
error_flag_overload
Error_Flag_Pulse
TP
In
Q
error_flag_fb
T#2ms
PT
ET
error_flag_piece_yoinked
28
piece_loaded
stop_actuators
conveyor_fb
29
stopper_retract
advance_stopper_fb
30
stopper_extend
return_stopper_fb

A4: Generic function block ladder networks:

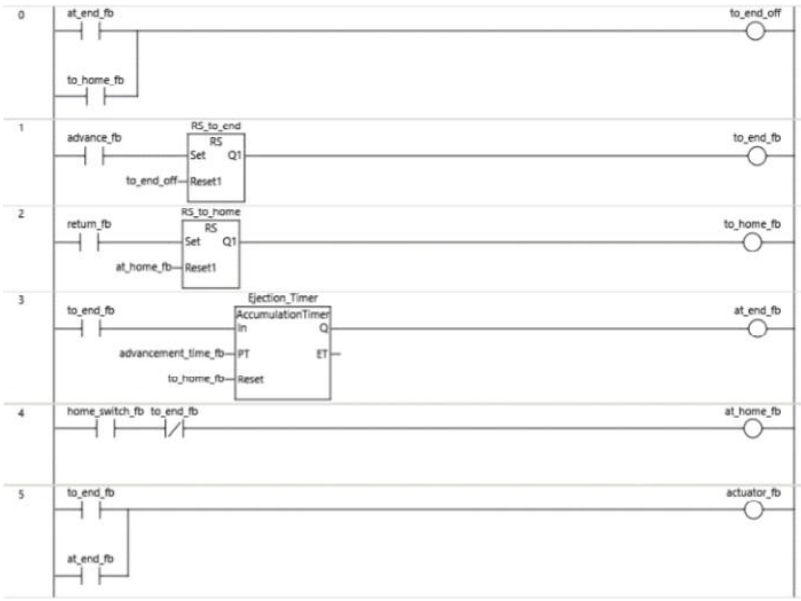
A4.1: Actuator_T1:



A4.2: Actuator_T2:

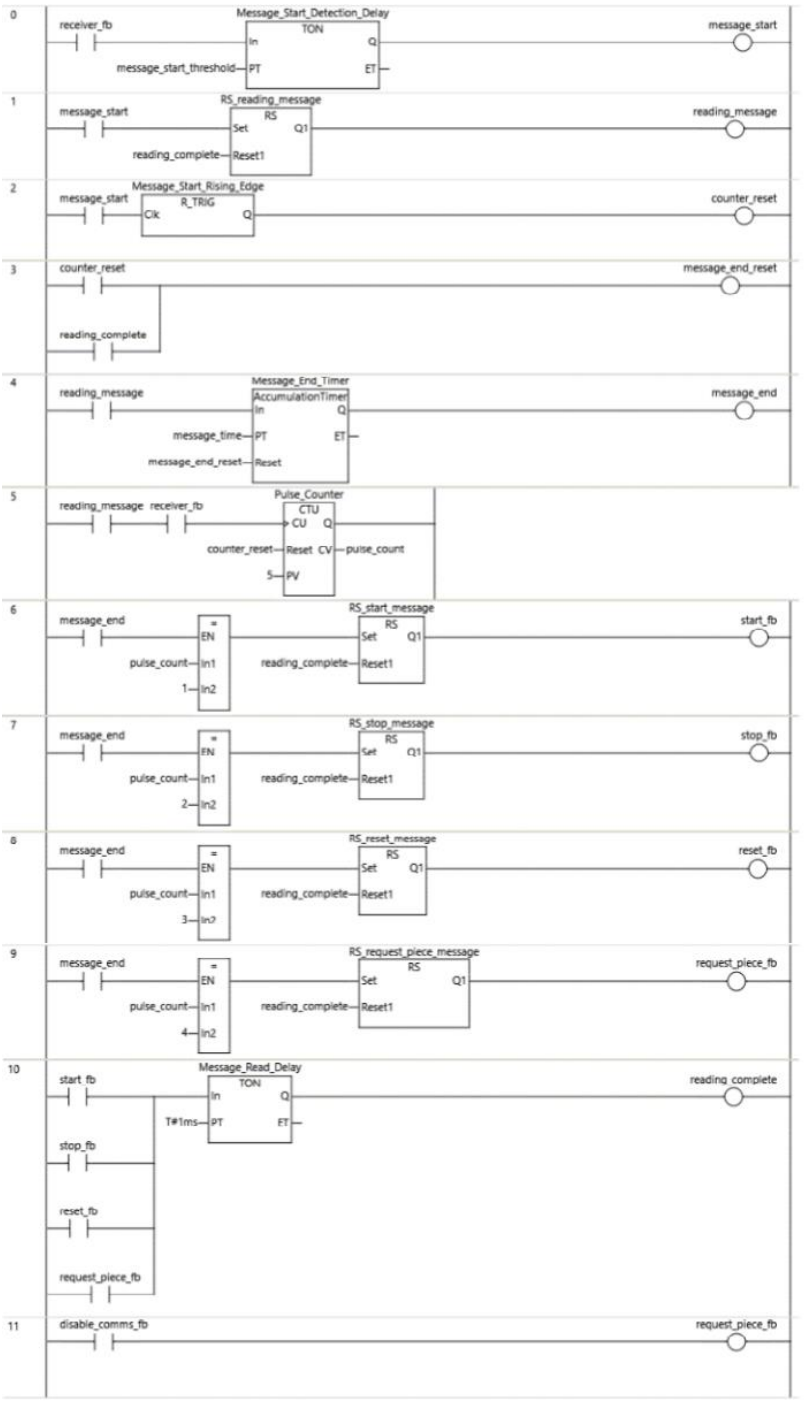


A4.3: Actuator_T3:



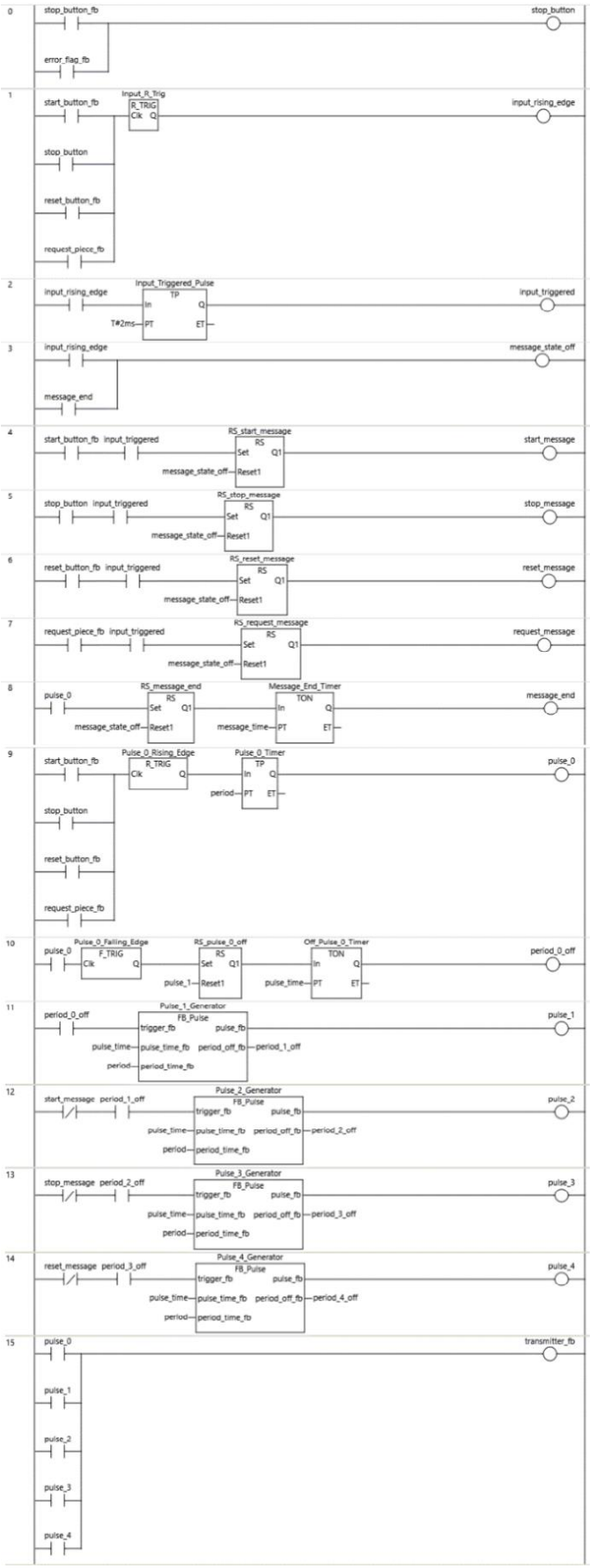
K

A4.4: Receiver:



L

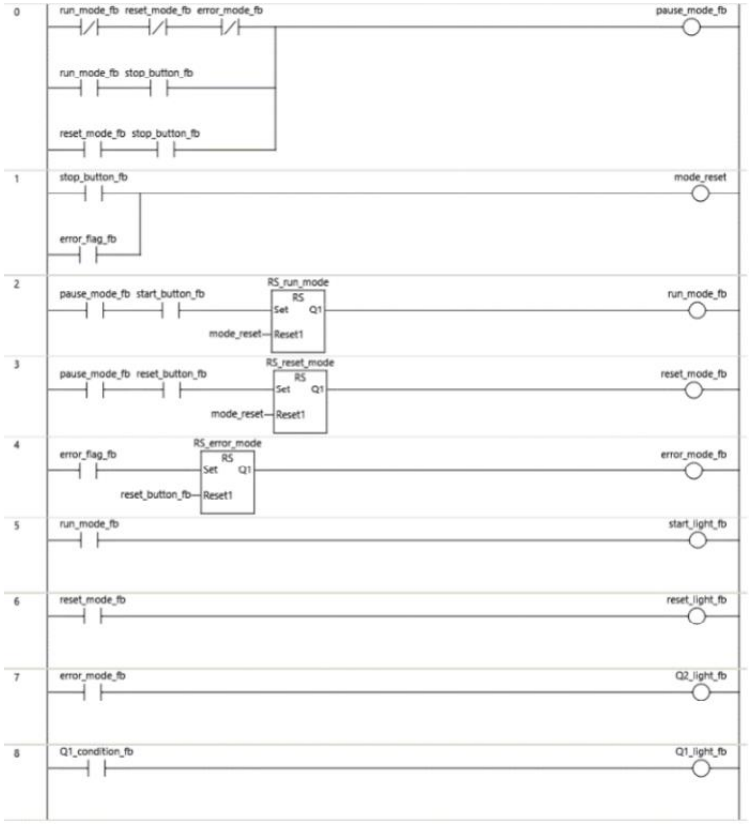
A4.5: Transmitter:



M

A4.6: Pulse:

A4.7: Control Panel:



N