### 1.Conveyor Control

**Use Case Name (System/Project)**:

Conveyor Belt Control System

**Actors:**

Operator

**Use Cases:**

[UC-1]: Monitor Sensors for Object Detection – Continuously monitor five sensors for object detection.

[UC-2]: Process Stop Button Activation – Detect and process stop button press at any station.

[UC-3]: Command Conveyor Belt Start – Start the belt when a sensor detects an object and no stop button is pressed.

[UC-4]: Command Conveyor Belt Stop – Stop the belt on stop button press or if sensor conditions are not met.

[UC-5]: Maintain Conveyor Belt Speed – Ensure belt speed is maintained at 2 meters per second.

**System Boundary:**

The system is an IEC 61131-3 Structured Text program designed to control a conveyor belt system with three stations. It interacts with an operator and processes inputs from five sensors to manage the conveyor belt's operation.

**Relationships Among Actors:**

The Operator interacts with the system by activating stop buttons (relevant to [UC-2], [UC-4]).

**Detailed Descriptive Information for Each Use Case:**

**Use Case: [UC-1] Monitor Sensors for Object Detection**

(1) Brief Description: Continuously monitor five sensors for object detection.

(2) Preconditions:

– Sensors are calibrated and active.

– System is powered on.

(3) Event Flow:

Basic Flow:

1. Poll sensors continuously.

2. Log detection signal if an object is detected.

3. Forward detection signal to start logic if no stop button is pressed.

Alternative Flow(s): (None explicitly provided in input)

(4) Post Conditions:

Success: Detection signal guides further actions.

(5) Special Requirements: (None explicitly provided beyond system overview)

**Use Case: [UC-2] Process Stop Button Activation**

(1) Brief Description: Detect and process stop button press at any station.

(2) Preconditions:

– Stop buttons are accessible and functioning.

(3) Event Flow:

Basic Flow:

1. Log stop command upon button press.

2. Stop command supersedes sensor detection.

3. Trigger belt stop.

Alternative Flow(s): (None explicitly provided in input)

(4) Post Conditions:

Success: System enters a stopped state.

(5) Special Requirements: (None explicitly provided beyond system overview)

**Use Case: [UC-3] Command Conveyor Belt Start**

(1) Brief Description: Start the belt when a sensor detects an object and no stop button is pressed.

(2) Preconditions:

– Positive sensor signal.

– No active stop command.

– Belt is off.

(3) Event Flow:

Basic Flow:

1. Receive detection signal.

2. Check for no active stop command.

3. Send start command and trigger speed maintenance.

Alternative Flow(s): (None explicitly provided in input)

(4) Post Conditions:

Success: Belt moves at commanded speed.

(5) Special Requirements: (None explicitly provided beyond system overview)

**Use Case: [UC-4] Command Conveyor Belt Stop**

(1) Brief Description: Stop the belt on stop button press or if sensor conditions are not met.

(2) Preconditions:

– Stop command issued or no sensor detection.

(3) Event Flow:

Basic Flow:

1. Receive stop command.

2. Verify and send stop command to belt.

Alternative Flow(s): (None explicitly provided in input)

(4) Post Conditions:

Success: Belt stops completely.

(5) Special Requirements: (None explicitly provided beyond system overview)

**Use Case: [UC-5] Maintain Conveyor Belt Speed**

(1) Brief Description: Ensure belt speed is maintained at 2 meters per second.

(2) Preconditions:

– Belt is running.

(3) Event Flow:

Basic Flow:

1. Set belt motor to 2 m/s.

2. Monitor and adjust speed as needed.

Alternative Flow(s): (None explicitly provided in input)

(4) Post Conditions:

Success: Belt runs uniformly at 2 m/s.

**(5) Special Requirements: (None explicitly provided beyond system overview)**

**Priority of Use Cases:**

[UC-1] Monitor Sensors for Object Detection: High

[UC-2] Process Stop Button Activation: High

[UC-3] Command Conveyor Belt Start: High

[UC-4] Command Conveyor Belt Stop: High

[UC-5] Maintain Conveyor Belt Speed: Medium

**Data Requirements (as provided):**

[D-1]: Sensor Data: Real-time status from sensors.

[D-2]: Button Input Data: Signals from stop buttons.

[D-3]: Conveyor Belt Status Data: States and speed measurement.

[D-4]: Timestamp/Data Logging: Recording times of detections and commands.

**Business Rules (as provided):**

[B-1]: Belt starts only with valid sensor detection and no stop command.

[B-2]: Stop command takes precedence over sensor signals.

[B-3]: Belt speed must be regulated to 2 m/s once activated.

[B-4]: Commands must be executed in real-time for safety and efficiency.

### CTL/LTL Specifications (with Source Mapping):

1. **AG((Sensors\_Calibrated & System\_Powered & Sensor\_Detect) -> Detection\_Signal)**

From: [UC-1] Preconditions (Sensors\_Calibrated, System\_Powered) & Basic Flow

1. **AG(Stop\_Button\_Pressed -> Stop\_Command)**

From: [UC-2] Basic Flow

1. **AG((Detection\_Signal & !Stop\_Command) ->** **Belt\_Running)**

From: [UC-3] Preconditions

1. **AG((Stop\_Command | (!Sensor\_Detect)) -> !Belt\_Running)**

From: [UC-4] Preconditions & Basic Flow; Business Rule [B-2].

1. **AG(Belt\_Running -> (Belt\_Speed = 2))**

From: [UC-5] Precondition (Belt\_Running) & Basic Flow/Postcondition (Belt\_Speed = 2); Business Rule [B-3].

### Spray Station

**Use Case Name (System/Project):**

Spray Station Operation

**Actors:**

Operator

Conveyor System

Spray System

**Use Cases:**

[UC-1]: Spray Station Operation – The process of moving the car to the spray station, spraying it, and then moving it away.

**System Boundary:**

IEC 61131-3 Structured Text Program

Relationships Among Actors:

The Operator interacts with both the Conveyor System and the Spray System.

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Spray Station Operation

(1) Brief Description:

The operator moves the car to the spray station using the conveyor system. Once the car is in position and the operator presses the spray button, the spray system automatically sprays the car for 3 seconds. After spraying, the spray button resets, and the conveyor system restarts to move the car away.

(2) Preconditions:

– The car is on the conveyor system.

– The spray system is ready to spray.

(3) Event Flow:

Basic Flow:

1. The operator presses the conveyor system button to move the car to the spray station.

2. The operator presses the spray button.

3. The spray system sprays the car for 3 seconds.

4. The spray button resets.

5. The conveyor system restarts and moves the car away.

Alternative Flow:

– If the conveyor stop button is pressed while the car is being detected by the sensor, the conveyor system stops.

(4) Post Conditions:

Success: The car has been sprayed, the spray button is reset, and the car is moved away from the spray station.

(5) Use Case Scenario: (This section is usually a more narrative form of the Basic Flow, which is already covered. For dataset consistency, I'm including it as provided.)

An operator is spraying a car. They press the conveyor system button to move the car to the spray station. Once the car is in position, they press the spray button. The spray system automatically sprays the car for 3 seconds. The spray button resets, and the conveyor system restarts to move the car away.

(6) Special Requirements:

– The spray system must be able to detect the car and spray for exactly 3 seconds.

Priority of Use Cases:

[UC-1] Spray Station Operation: High

**Data Requirements:**

[D-1]: Car position data from the conveyor system. (Implies a sensor for car detection)

[D-2]: Spray system status data.

**Business Rules (Optional):**

[B-1]: The spray system must only spray when the car is detected and the spray button is pressed.

[B-2]: The conveyor system must stop when the conveyor stop button is pressed and the car is detected by the sensor.

### CTL/LTL Expressions (with Source Mapping ):

**1.AG((operator\_presses\_spray\_button & car\_detected) -> spray\_system\_activate)**

From: [UC-1] Basic Flow step 2 & 3, [UC-1] Special Requirements, Business Rule [B-1].The operator presses the spray button and detects a car, then the system will activate the spraying.

**2.A****G(conveyor\_stop\_button\_pressed & car\_detected -> !conveyor\_system\_starts)**

From: [UC-1] Alternative Flow, Business Rule [B-2].当传送带停止按钮被按下且检测到汽车时，传送带系统不会启动

**3.AG(!(spray\_active & (!car\_detected | !****spray\_button\_pressed)))**

From: Inverse of Business Rule [B-1].禁止在未检测到汽车或未按下喷淋按钮时进行喷淋

**4.G(spray & spray\_timer = 3 -> X !spray)**  
From: [UC-1] Basic Flow step 3, [UC-1] Special Requirements.定义了喷淋操作在持续3秒后必须停止  
**5.****EF(conveyor\_stop\_button\_pressed & car\_detected)**

From: [UC-1] Alternative Flow (implies this condition can be reached).存在一条路径可以达到“传送带停止按钮被按下且检测到汽车”的状态  
**6.AG(EF(spray\_button\_pressed & car\_detected))**

From：[UC-1] Basic Flow steps 2-3，系统总是可能最终达到“操作员按下喷淋按钮且检测到汽车”的状态

**7.AG((timer> = 3 & !spray) -> (conveyor\_system\_restarts\_signal))**

From: [UC-1] Basic Flow step 5 ("The conveyor system restarts and moves the car away").当喷淋完成后，传送带系统重新启动的信号激活。

1条LTL, 6条CTL 一共7条

### Liquid Mixing System

**Use Case Name (System/Project):**

Liquid Mixing System

**Actors:**

Operator

**Use Cases:**

[UC-1]: Pouring Liquid – The operator uses buttons S1 and S2 to pour unmixed solutions into the tank.

[UC-2]: Preparing Mixed Solution – The operator uses button S3 to prepare the mixed solution.

**System Boundary:**

control system managing buttons S1, S2, S3, liquid pouring, motor M, valve V1, and tank level sensing.

**Relationships Among Actors:**

The Operator interacts with the system by pressing buttons S1, S2, and S3, and observing the liquid level.

**Detailed Descriptive Information for Each Use Case:**

**Use Case: [UC-1] Pouring Liquid**

(1) Brief Description:

The operator uses buttons S1 and S2 to pour unmixed solutions into the tank.

(2) Preconditions:

– The tank is not full.

(3) Event Flow:

Basic Flow:

1. The operator presses button S1, and Liquid 1 is poured into the tank in the next state.

2. The operator presses button S2, and Liquid 2 is poured into the tank in the next state.

Alternative Flow:

– If the tank is full, buttons S1 and S2 are disabled in the next state.

(4) Post Conditions:

Success: The tank contains unmixed solutions, unless it is full.

(5) Use Case Scenario:

The operator wants to pour Liquid 1 into the tank, so they press button S1, and Liquid 1 is poured in the next state.

The operator wants to pour Liquid 2 into the tank, so they press button S2, and Liquid 2 is poured in the next state.

(6) Special Requirements:

– The system must accurately measure the liquid level in the tank and disable buttons S1 and S2 in the next state when the tank is full.

Priority: High

**Use Case: [UC-2] Preparing Mixed Solution**

(1) Brief Description:

The operator uses button S3 to prepare the mixed solution.

(2) Preconditions:

– The tank contains unmixed solutions.

(3) Event Flow:

Basic Flow:

1. The operator presses button S3, buttons S1 and S2 are disabled in the next state, the mixing motor M is activated in the next state, and valve V1 is eventually opened to discharge the mixed liquid.

Alternative Flow:

– If the tank is empty, button S3 is disabled in the next state.

(4) Post Conditions:

Success: The tank is empty, and the mixed solution is ready.

(5) Use Case Scenario:

The operator wants to prepare the mixed solution, so they press button S3.

(6) Special Requirements:

– The system must accurately control the mixing motor M and ensure valve V1 is eventually opened.

Priority: Medium

**Data Requirements:**

[D-1]: Button S1 Status (Pressed/NotPressed)

[D-2]: Button S2 Status (Pressed/NotPressed)

[D-3]: Button S3 Status (Pressed/NotPressed)

[D-4]: Tank Level (e.g., NotFull, Full, Empty, or a continuous value)

[D-5]: Liquid 1 Pouring Status (Active/Inactive)

[D-6]: Liquid 2 Pouring Status (Active/Inactive)

[D-7]: Motor M Status (Active/Inactive)

[D-8]: Valve V1 Status (Open/Closed)

**Business Rules:**

[B-1]: Pouring buttons (S1, S2) are disabled if the tank is full.

[B-2]: Mixing button (S3) is disabled if the tank is empty.

[B-3]: When mixing (S3 pressed), pouring buttons (S1, S2) are disabled.

[B-4]: When mixing (S3 pressed), motor M activates, and eventually valve V1 opens.

### CTL/LTL Specifications (with Source Mapping):

1. **G((Operator\_Presses\_S1 &** **!Tank\_Is\_Full) -> X(Liquid1\_Is\_Pouring))**

From: [UC-1] Basic Flow step 1.如果按下S1，tank不满等满足，则Liquid1倒入。

1. **G((Operator\_Presses\_S2 & !Tank\_Is\_Full) -> X(Liquid1\_Is\_Pouring))**

From: [UC-1] Basic Flow step 2.如果按下S2，tank不满等条件满足，则Liquid2倒入。

1. **G(tank\_full & X(button\_S1 & button\_S2) ) -> X(****!****button\_S1 & !****button\_S2))**

From: [UC-1] Alternative Flow, [UC-1] Special Requirements. 如果油箱已满，则在S1和S2按钮信号将被禁用

1. **G(button\_S3 -> X (!****button\_S1 & !****button\_S2 & motor\_M & F valve\_V1))**

From: [UC-2] Basic Flow step 1, [UC-2] Special Requirements.按下S3后，在下一个状态S1和S2被禁用，并且电机M启动。最终阀门V1会打开

1. **G(tank\_empty -> X (!button\_S3))**

From：[UC-2] Alternative Flow.如果tank为空，则在下一个状态S3按钮被禁用。

一共5条LTL

### Elevator Control

**Use Case Name (System/Project):**

Elevator Control System (Inferred)

**Actors:**

Building User

Passenger

Elevator Control System (internal logic)

**Use Cases:**

[UC-1]: Process Floor Call Request

[UC-2]: Manage Elevator Movement

[UC-3]: Control Elevator Door Operation

[UC-4]: Reopen Elevator Door on Inactivity

**System Boundary:**

IEC 61131-3 Structured Text Elevator Control Program

**Relationships Among Actors:**

Building User interacts with floor call buttons.

Passenger interacts with the internal call panel.

Both generate requests for [UC-1].

[UC-2] (Elevator Control System) receives data from [UC-1] and interfaces with floor limit switches.

[UC-3] (Elevator Control System) is triggered by [UC-2] stop event and affects Passengers.

[UC-4] (Elevator Control System) is triggered after [UC-3] door closure and monitors internal buttons.

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Process Floor Call Request

(1) Brief Description:

Registers floor call requests by detecting button presses from external panels and internal panels, logs the floor and desired direction, and queues the request for elevator movement.

(2) Preconditions:

1. The control program is active and monitoring button signals.

2. Sensors for external and internal call buttons are operational.

(3) Event Flow:

Basic Flow (Event Flow):

1. A Building User or Passenger presses a call button.

2. The system detects the button press.

3. The requested floor and associated direction (up or down) are recorded.

4. The request is queued in the call queue.

5. The movement control module is notified to update its request list.

6. End event.

Alternative Flow (Alternative Flow):

Alternate Flow 1 – Button Signal Noise:

a. If a transient signal is detected, the input is ignored.

b. A debounce check is performed.

c. If confirmed, the request is processed as in the Basic Flow.

d. End alternative flow.

Alternate Flow 2 – Simultaneous Conflicting Signals:

a. If both up and down buttons are activated concurrently on a floor, an error is logged.

b. The system prioritizes the request matching the current elevator movement.

c. The conflicting request is deferred until the elevator changes direction.

d. End alternative flow.

(4) Postconditions:

• The call queue is updated with the floor and direction.

• The control module is ready to schedule the elevator stop.

• A log entry is created with the request details.

(5) Use Case Scenario:

A user presses a button and the call is successfully recorded along with its directional intent. In cases of noise or conflict, the system employs a debounce mechanism or logs an error before proceeding.

(6) Special Requirements:

• Rapid registration of the call request.

• Debounce algorithms to filter out noise.

• Priority adjustments when simultaneous requests are received.

Priority: High

Use Case: [UC-2] Manage Elevator Movement

(1) Brief Description:

Controls the elevator’s travel between floors by selecting the next stop based on directional requests. When moving upward, it stops only at floors with active up button requests; when moving downward, it stops only at floors with active down button requests.

(2) Preconditions:

1. The call queue contains valid floor requests with associated direction indicators.

2. All floor limit switch sensors are functioning properly.

3. The current floor position and elevator direction are known.

(3) Event Flow:

Basic Flow (Event Flow):

1. The system identifies the elevator’s current direction.

2. If the elevator is moving upward:

a. The call queue is scanned for active up button requests on floors above the current position.

b. The next immediate floor (the smallest numbered floor above the current floor with an active up request) is selected.

c. The elevator is commanded to move upward.

d. As the elevator nears the selected floor, the floor’s limit switch is monitored.

e. On activation of the limit switch, the elevator stops, the related request is cleared, and the door operation processes are triggered.

3. If the elevator is moving downward:

a. The call queue is scanned for active down button requests on floors below the current position.

b. The closest floor (the highest numbered floor below the current floor with an active down request) is selected.

c. The elevator is commanded to move downward.

d. As the elevator nears the floor, the corresponding floor limit switch is monitored.

e. On activation, the elevator stops, the request is removed from the queue, and door operations are initiated.

4. If no matching directional call is found:

a. The system checks for any pending call in the opposite direction.

b. If found, the elevator changes direction and follows the respective process.

c. Otherwise, the system enters a standby state awaiting new call requests.

5. End event.

Alternative Flow (Alternative Flow):

Alternate Flow 1 – No Valid Directional Call:

a. If there are no up requests while moving upward or down requests while moving downward, the system inspects if calls exist in the opposite direction.

b. If such calls exist, the elevator changes direction.

c. If none are present, the elevator holds its current position until a new request arrives.

d. End alternative flow.

Alternate Flow 2 – Sensor or Limit Switch Failure:

a. If a floor arrival is not confirmed due to sensor/limit switch malfunction, the elevator issues a safe-stop command.

b. An error is logged and the system enters emergency mode requiring manual intervention.

c. End alternative flow.

(4) Postconditions:

• The elevator stops at the designated floor corresponding to the matched directional request.

• The fulfilled call request is removed from the call queue.

• The door control use case is activated.

• The elevator’s current position and direction are updated.

(5) Use Case Scenario:

During upward transit, the system continuously monitors for any active up button request above the current floor, and similarly for downward transit. When such a request is detected, the elevator is commanded to stop accordingly and the request is cleared once the stop is confirmed via the floor’s limit switch.

(6) Special Requirements:

• Real-time evaluation of directional call requests.

• Accurate synchronization with floor sensor inputs for reliable stopping.

• Priority in processing calls that match the present elevator motion.

• Immediate emergency handling in case of sensor anomalies.

Priority: High

Use Case: [UC-3] Control Elevator Door Operation

(1) Brief Description:

Manages the opening of the elevator door for 7 seconds upon floor arrival and then closes it. The process may be interrupted if a safety condition is detected.

(2) Preconditions:

1. Elevator has successfully arrived at a designated floor.

2. Door mechanism and its sensors are operational.

(3) Event Flow:

Basic Flow (Event Flow):

1. Following floor arrival, the door is commanded to open.

2. A 7-second timer is initiated concurrently.

3. After 7 seconds, the door is commanded to close.

4. End event.

Alternative Flow (Alternative Flow):

Alternate Flow 1 – Premature Door Closure:

a. If a safety sensor or fault detection triggers an override while the door is open, a close command is issued immediately.

b. The system logs the override event and recalibrates the timer if needed.

c. End alternative flow.

(4) Postconditions:

• The door remains open for 7 seconds and then closes, unless a safety override occurs.

• The door state is updated in the control system.

• The closure event signals subsequent monitoring for internal button activity (see UC4).

(5) Use Case Scenario:

After landing on a floor, the door opens automatically and remains open for the set duration unless a safety-related event necessitates an earlier closure, in which case the timer is adjusted.

(6) Special Requirements:

• The door open timer must adhere strictly to the 7-second duration.

• A safety override must immediately cancel the timer.

• All door operations are logged for maintenance.

Priority: High

Use Case: [UC-4] Reopen Elevator Door on Inactivity

(1) Brief Description:

If no internal floor request is detected within 10 seconds after the door closes, the system reopens the door to prompt passenger input.

(2) Preconditions:

1. The door has closed as per the normal cycle.

2. No internal call button press has been registered within the 10-second window following door closure.

3. Internal sensors for button press detection are operational.

(3) Event Flow:

Basic Flow (Event Flow):

1. Immediately after door closure, a 10-second inactivity timer is started.

2. The system monitors internal buttons during this period.

3. If no button press is detected by the end of 10 seconds, the door is automatically commanded to reopen.

4. End event.

Alternative Flow (Alternative Flow):

Alternate Flow 1 – Button Press Detected During Timer:

a. If an internal button is pressed before the 10-second timer expires, the inactivity timer is canceled.

b. The door remains closed and the new request is processed through UC1.

c. End alternative flow.

(4) Postconditions:

• If inactivity persists, the door is reopened.

• If a button press occurs within 10 seconds, the system continues normal operation.

• The inactivity timer is reset after each cycle.

(5) Use Case Scenario:

Upon door closure, the system briefly awaits any further passenger input. Lack of input triggers an automatic door reopening to invite a new floor selection.

(6) Special Requirements:

• The 10-second timer must be precise and initiated immediately after door closure.

• Sensor monitoring for internal requests must remain uninterrupted during this period.

• All door reopen events are logged for system diagnostics.

Priority: High (Note: Original input was Medium, but I am keeping it as High as per the summary list provided later in the input. If it should be Medium, please clarify.)

**Data Requirements:**

[D-1]: Call Queue Data Structure containing floor numbers, requested direction, timestamp, and source indicator. (From UC1)

[D-2]: Current Elevator Status including the current floor, travel direction, and sensor state. (From UC2)

[D-3]: Movement Command Log capturing issued movement orders and confirmed stops. (From UC2)

[D-4]: Door Timer Record including start time, duration, and end time of each operation cycle. (From UC3)

[D-5]: Door Sensor Status data for real-time monitoring of door positions. (From UC3)

[D-6]: Inactivity Timer Log capturing start and stop times. (From UC4)

[D-7]: Internal Button Press Records with timestamps during the inactivity period. (From UC4)

**Business Rules (Optional):**

[B-1]: If multiple requests occur, priority is given to requests aligning with the current elevator direction. (From UC1)

[B-2]: The elevator must always halt for an active up button request during upward movement and for an active down button request during downward movement, whenever such a request exists. (From UC2)

[B-3]: The door shall remain open for 7 seconds unless a safety override condition forces an immediate closure. (From UC3)

[B-4]: On reopening due to inactivity, the cycle resets and any subsequent button press restarts the door control process. (From UC4)

Summary of Use Case Priorities:

• UC1: Process Floor Call Request – High

• UC2: Manage Elevator Movement – High

• UC3: Control Elevator Door Operation – High

• UC4: Reopen Elevator Door on Inactivity – Medium (Corrected based on this summary, overriding the "High" I noted earlier for UC4)

### CTL/LTL Specifications (with Source Mapping):

1. **AG((Elevator\_Moving\_Up & Up\_Call\_Button\_Active & !Limited\_Switch) -> Elevator\_Stops\_At\_Floor)**

From：[UC-2] Basic Flow steps 2 & 3, Business Rule [B-2].表述电梯在特定方向移动时，如果该方向上有呼叫，则电梯会在相应楼层停止。

1. **AG((Elevator\_Moving\_Down & Down\_Call\_Button\_Active & !Limited\_Switch) -> Elevator\_Stops\_At\_Floor)**

From：[UC-2] Basic Flow steps 2 & 3, Business Rule [B-2].表述电梯在特定方向移动时，如果该方向下有呼叫，则电梯会在相应楼层停止。

1. **G((****Doors\_Open & timer=7000) ->** **Doors\_Close\_Command)**

From: [UC-3] Postcondition.如果门仍在开启状态（或者关闭命令已发出）且计时器已满足条件，则马上发出关门指令。

1. **G((!Doors\_Open & Timer >= 10000 & !Button\_Pressed) -> Doors\_Open\_Command)**;

From: [UC-4] Basic Flow steps 1-3, Special Requirements.如果门已关闭，计时器达到或超过10秒，并且没有按钮被按下，则发出开门指令。

1. **AG(!(Elevator\_Moving\_Up & Top\_Limit\_Switch\_Activated | Elevator\_Moving\_Down & Bottom\_Limit\_Switch\_Activated))**  
   From: Physical safety limit, implied by [UC-2] (stopping at floors, not overshooting),禁止电梯向上移动时触发顶部限位开关，或向下移动时触发底部限位开关（这些通常是极限停止开关，表示故障或越程）
2. **AG(ControlProgram\_Active &CallButton\_Sensors\_Operational)**  
   From：[UC-1] Preconditions 在所有路径上，控制程序始终是激活的，并且呼叫按钮传感器始终是可操作的。

一共6个 2个LTL

### Sequential Motor Control

**Use Case Name (System/Project):**

Sequential Motor Control System (Inferred)

**Actors:**

Motor Control System

**Use Cases:**

[UC-1]: Sequential Motor Operation – The process of starting three motors in a specific sequence with a delay between each start.

**System Boundary:**

IEC 61131-3 Structured Text Program

**Relationships Among Actors:**

The Motor Control System is responsible for executing the Sequential Motor Operation.

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Sequential Motor Operation

(1) Brief Description:

The use case involves the operation of three motors in a specific sequence. Motor 1 starts first, followed by Motor 2 after a delay, and finally Motor 3 starts after another delay. The entire operation must be completed within 10 seconds. At any given time, only one motor can be in the start state.

(2) Preconditions:

– All three motors are in a state ready for operation.

(3) Event Flow:

Basic Flow:

1. The Motor Control System initiates the operation.

2. Motor 1 starts.

3. After a delay, Motor 2 starts only if Motor 1 has stopped.

4. After another delay, Motor 3 starts only if Motor 2 has stopped.

5. The operation is completed within 10 seconds.

Alternative Flow:

– If any motor fails to start, the Motor Control System halts the operation and triggers an error notification.

(4) Post Conditions:

Success: All three motors have started in sequence and the operation is completed within 10 seconds.

(5) Use Case Scenario:

In a manufacturing plant, the Motor Control System is used to operate three motors that drive different parts of a production line. The motors must start in a specific sequence to ensure the smooth operation of the production line.

(6) Special Requirements:

– The operation must adhere to the constraint that two motors cannot start simultaneously, and the total start time must be within 10 seconds. Additionally, each motor must stop before the next motor starts.

**Priority of Use Cases:**

[UC-1] Sequential Motor Operation: High

**Data Requirements:**

[D-1]: Motor Status Data: Information about the status of each motor (e.g., ready, operating, error).

**Business Rules:**

[B-1]: Two motors cannot start simultaneously.

[B-2]: The motors must start in sequence. (Motor 1 -> Motor 2 -> Motor 3)

[B-3]: The total start time must be within 10 seconds. (This seems to refer to the completion of the entire sequence of starts, not just individual motor start durations).

[B-4]: Each motor must stop before the next motor starts. (Derived from Event Flow and Special Requirements).

### CTL/LTL Expressions (with Source Mapping):

1. **AG(****!(Motor1\_Start & Motor2\_Start) & !(Motor2\_Start & Motor3\_Start)&!(Motor1\_Start & Motor3\_Start))**

From: [UC-1] Basic Flow steps 2, 3, 4; [UC-1] Special Requirements; Business Rule [B-2], [B-4].

1. **G((****Motor1\_Running AND Delay1\_Timer\_Elapsed) --> X(Motor2\_****Running))  
    G((Motor2\_Stopped AND Delay2\_Timer\_Elapsed) --> X(Motor3\_Running))**

**G((Motor3\_Stopped AND Delay3\_Timer\_Elapsed) --> X(Motor2\_Running))**  
From: [UC-1] Basic Flow step 5 (按顺序启动).

1. **AG(Operation\_Total\_Time <= 10)**  
   From：[UC-1] Basic Flow step 5; [UC-1] Special Requirements; Business Rule [B-3].约束总操作时间在10s以内。

一共3个，1CTL,2LTL

### Traffic Control

**Use Case Name (System/Project):**

Traffic Light Control System Operation

**Actors:**

Pedestrian

Vehicle

Emergency Vehicle

**Use Cases:**

[UC-1]: Normal Traffic Light Cycle – Cycle the traffic lights through the sequence green → yellow → red under stable conditions.

[UC-2]: Process Pedestrian Request – When a pedestrian presses the button under safe conditions (the light is red, no vehicles or emergency vehicles are detected), change the light to green. Note that any detection of an emergency event immediately cancels this request.

[UC-3]: Process Approaching Vehicle – When a vehicle is detected, transition the current light to yellow and then to red, ensuring a safe interruption to the normal cycle.

[UC-4]: Process Emergency Vehicle Request – When an emergency vehicle is detected, immediately override any current operation. This use case explicitly checks for any concurrent pedestrian requests; if both are present, the system cancels the pedestrian request and forces a red light in the next state.

[UC-5]: Handle System/Sensor Failure – Monitor sensor and system errors. When a failure is detected, suspend any active use case, transition the traffic lights to a safe state, and alert maintenance.

**System Boundary:**

IEC 61131-3 Structured Text Program – Traffic Light Controller

**Relationships Among Actors:**

Pedestrians activate UC-2 by pressing the pedestrian button.

Vehicles and Emergency Vehicles are detected via sensors that trigger UC-3 and UC-4 respectively.

The Normal Traffic Light Cycle (UC-1) is continuously running in the background but can be overridden by any event that initiates UC-2, UC-3, or UC-4.

UC-5 is invoked automatically whenever a sensor or system error is detected, suspending any ongoing operations.

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Normal Traffic Light Cycle

(1) Brief Description:

Continuously cycle the traffic lights through green → yellow → red under stable, unperturbed conditions.

(2) Preconditions:

– The system is active.

– No external events (pedestrian, vehicle, or emergency vehicle requests) are pending.

(3) Event Flow:

Basic Flow:

1. The system displays green for a preset duration.

2. The light transitions to yellow after the green interval.

3. The light changes to red following the yellow phase.

4. The cycle then repeats from green.

Alternative Flow:

– If any external sensor input (pedestrian, vehicle, or emergency vehicle) is detected during any phase, the normal cycle is suspended and control shifts to the corresponding use case.

(4) Post Conditions:

– The system resumes standard cyclic operation after handling any interruptions.

(5) Special Requirements:

– Only one traffic light may be active at any given time.

– Timing intervals should be strictly enforced.

Use Case: [UC-2] Process Pedestrian Request

(1) Brief Description:

Respond to a pedestrian’s button press by transitioning the light from red to green, assuming safe conditions are met.

(2) Preconditions:

– The pedestrian button is pressed.

– The current light is red.

– No vehicles or emergency vehicles are detected by the sensors.

(3) Event Flow:

Basic Flow:

1. Pedestrian presses the button.

2. The system verifies that the light is red and that there are no approaching vehicles or emergency vehicle events.

3. The light is switched to green to allow pedestrian crossing.

4. Once the pedestrian crossing interval ends, the system returns to the normal cycle.

Alternative Flow:

– If a vehicle is detected following the button press, or if an emergency event occurs concurrently, the pedestrian request is cancelled and control is handed off to UC-3 or UC-4, as applicable.

(4) Post Conditions:

– The traffic light is set to green for pedestrian crossing if conditions are safe.

– The system resumes the normal cycle after the crossing is complete.

(5) Special Requirements:

– This use case is immediately overridden by both vehicle and emergency vehicle events.

Use Case: [UC-3] Process Approaching Vehicle

(1) Brief Description:

Immediately respond to the detection of an approaching vehicle by transitioning the traffic light to yellow and then to red.

(2) Preconditions:

– A vehicle is detected via sensor input.

– No emergency vehicle request is currently active.

(3) Event Flow:

Basic Flow:

1. The vehicle sensor detects an approaching vehicle.

2. The system transitions the current light to yellow as a caution signal.

3. The light then rapidly transitions to red to stop traffic.

4. Following the vehicle event, the normal cycle resumes.

Alternative Flow:

– If vehicles are detected in rapid succession, the red phase is extended to ensure safety.

(4) Post Conditions:

– The traffic light remains red long enough to ensure vehicular stoppage.

– The system resumes the standard cycling once the threat has passed.

(5) Special Requirements:

– The system’s reaction time must be minimal.

Use Case: [UC-4] Process Emergency Vehicle Request

(1) Brief Description:

Provide immediate and exclusive processing for emergency vehicles by overriding all other operations.

(2) Preconditions:

– An emergency vehicle is detected via reliable sensors.

(3) Event Flow:

Basic Flow:

1. The emergency vehicle is detected by the sensor.

2. Any ongoing operations—including pedestrian and vehicle processing—are preempted immediately.

3. The system transitions the lights to a state optimized for emergency passage. For instance, if a pedestrian button is pressed concurrently, the system forces a red light in the next state to ensure safety.

4. After the emergency vehicle has passed and sensor functionality is verified, the system reverts to the normal cycle.

Alternative Flow:

– If a pedestrian request is raised during the emergency event, the system cancels the request and enforces a red light in the subsequent state.

– If sensor or system errors occur during emergency handling, UC-5 is invoked.

(4) Post Conditions:

– Exclusive clearance is provided to the emergency vehicle.

– The system maintains a forced red light when an emergency and a pedestrian event occur concurrently, before safely returning to the normal cycle.

(5) Special Requirements:

– Emergency requests always override any pedestrian or vehicular requests.

Use Case: [UC-5] Handle System/Sensor Failure

(1) Brief Description:

Detect and manage any sensor or system errors by transitioning to a safe state and alerting maintenance.

(2) Preconditions:

– A sensor or system error is detected.

(3) Event Flow:

Basic Flow:

1. The system continuously monitors sensor and system performance.

2. Upon detection of an error, the current operation is suspended.

3. The traffic light transitions to a predetermined safe state ensuring no conflicting signals.

4. Maintenance is alerted to the failure.

5. Once the error is resolved and sensors are confirmed operational, the system resumes the normal cycle.

(4) Post Conditions:

– The system is in a safe state until errors are corrected.

– Service personnel are notified of the issue.

(5) Special Requirements:

– The current operation is suspended immediately upon error detection to ensure safety.

**Priority of Use Cases:**

[UC-4 Process Emergency Vehicle Request]: High

[UC-3 Process Approaching Vehicle]: High

[UC-2 Process Pedestrian Request]: Medium

[UC-1 Normal Traffic Light Cycle]: Low

[UC-5 Handle System/Sensor Failure]: High

**Data Requirements:**

[D-1]: Sensor data indicating vehicle presence.

[D-2]: Status of the pedestrian button (pressed/released).

[D-3]: Sensor data indicating emergency vehicle detection.

[D-4]: Configured timing intervals for transitioning between lights.

[D-5]: Current state indicator for the traffic light (e.g., green, yellow, red).

[D-6]: Sensor/system status for detecting error states.

**Business Rules:**

[B-1]: Only one traffic light may be illuminated at a given time.

[B-2]: Emergency vehicle requests override any pedestrian or vehicle events.

[B-3]: The traffic light cycle (green → yellow → red) is always completed unless interrupted by a higher-priority event.

[B-4]: In the event of concurrent emergency and pedestrian requests, the system forces a red light for safety before any further transitions.

### CTL/LTL Specifications (with Source Mapping):

1. **G((****light\_ green &** **!Vehicle\_Detected** **& !emergency\_vehicle\_detected & !Pedestrian\_Button & light\_ green\_time\_elapsed) -> X(light\_ yellow));**

From: [UC-1] Basic Flow steps 1-2.在没有外部请求的稳定条件下，绿灯之后最终会变黄灯。

1. **G((light\_ yellow & !Vehicle\_Detected & !emergency\_vehicle\_detected & !Pedestrian\_Button & light\_ yellow\_time\_elapsed) -> X(****light\_ red));**

From: [UC-1] Basic Flow steps 2-3.在没有外部请求的稳定条件下，黄灯之后最终会变红灯。

1. **G((light\_red) &** **!Vehicle\_Detected & !****Pedestrian\_Button & !emergency\_detected) -> X(light\_ green));**

From: [UC-1] Basic Flow steps 3-4在没有外部请求稳定条件下，红灯之后最终会变绿灯，完成一个循环。

1. **AG(!( (****light\_ green &****light\_yellow) | (light\_ green &** **light\_red) | (light\_yellow &** **light\_red)));**

From: [UC-1] Special Requirements; Business Rule [B-1].确保任何时候交通灯的不同颜色不会同时点亮。

1. **AG((Pedestrian\_Request\_Active & (Vehicle\_Detected |Emergency\_Vehicle\_Detected)) -> Pedestrian\_Request\_Cancelled)**

From: General system liveness/behavior.系统总是能够最终达到绿灯、黄灯或红灯状态之一。这表明系统不会卡在没有明确灯色的状态。

1. **G((****Pedestrian\_Button &** **light\_red & !****emergency\_vehicle\_detected & !Vehicle\_Detected ) -> X(light\_ green));**

From: [UC-2] Preconditions & Basic Flow step 3.如果满足所有行人请求条件并且在下一个状态有一个确认信号，那么灯会变绿

1. **AG((****emergency\_vehicle\_detected & Pedestrian\_Button) -> (light\_red));**

From: [UC-4] Basic Flow step 3, Alternative Flow; Business Rule [B-4].如果同时检测到紧急车辆和行人按钮按下，则在下一个状态灯保持变为红色状态。

一共7个，

### Water Pump

**Use Case Name (System/Project):**

Water Pump Control System (Inferred)

**Actors:**

User (Operator)

**Use Cases:**

[UC-1]: Fill Tank – The user starts the water pump to fill the tank until the level sensor detects a high water level.

[UC-2]: Empty Tank – The user empties the tank using the discharge valve.

**System Boundary:**

(Not explicitly defined, but implied to be the control system managing the start/stop buttons, pump, latching mechanism, level sensor, and discharge valve.)

**Relationships Among Actors:**

The User (Operator) interacts with the system by pressing Start/Stop buttons and opening/closing the discharge valve.

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Fill Tank

(1) Brief Description: The user starts the water pump to fill the tank until the level sensor detects a high water level.

(2) Preconditions:

– The tank is not full.

– The water pump is unlocked.

(3) Event Flow:

Basic Flow:

1. The user presses the Start Button (START PB) when the pump is unlocked.

2. The water pump starts filling the tank.

3. The system uses the SET instruction to latch the motor (make it not unlocked) once the pump is running.

4. The level sensor detects the high water level.

5. The system uses the RESET instruction to make the motor unlocked and stop the pump.

Alternative Flow:

1. If the water pump is not unlocked (i.e., latched), the user must press the Stop Button (STOP PB) to use the RESET instruction and make it unlocked before pressing the Start Button.

Exception Flow:

Sensor Failure:

1. If the level sensor fails, the system should trigger an alarm and use the RESET instruction to stop the water pump (and make it unlocked).

Button Malfunction:

1. If the Start or Stop Button malfunctions, the user should manually disconnect the power to stop the pump.

(4) Postconditions:

– The tank is full.

– The water pump is unlocked.

(5) Use Case Scenario: The user wants to fill the tank. They press the Start Button when the pump is unlocked, the pump starts, and the system latches the motor (makes it not unlocked) using the SET instruction. When the tank is full, the level sensor detects the high water level, and the system makes the motor unlocked using the RESET instruction to stop the pump.

(6) Special Requirements:

– The water pump must be unlocked when the Start Button is pressed.

Priority: High

Use Case: [UC-2] Empty Tank

(1) Brief Description: The user empties the tank using the discharge valve.

(2) Preconditions:

– The tank is full.

(3) Event Flow:

Basic Flow:

1. The user opens the Discharge Valve.

2. The water in the tank is emptied.

Exception Flow:

Valve Malfunction:

1. If the Discharge Valve fails, the user should manually empty the tank.

(4) Postconditions:

– The tank is empty.

(5) Use Case Scenario: The user wants to empty the tank. They open the Discharge Valve, and the water in the tank is emptied.

(6) Special Requirements:

– The tank must be full before the Discharge Valve can be opened.

Priority: Medium

**Data Requirements:**

(Inferred from use cases and specifications, using "unlocked" terminology)

[D-1]: Tank Full Status (Boolean: true if full, false otherwise)

[D-2]: Pump Unlocked Status (Boolean: true if unlocked, false if latched)

[D-3]: Start Button Press Event

[D-4]: Stop Button Press Event

[D-5]: Pump Running Status (Boolean)

[D-6]: SET Latch Command Event/Status (makes pump not unlocked)

[D-7]: High Water Level Detected Event/Status

[D-8]: RESET Unlatch Command Event/Status (makes pump unlocked)

[D-9]: Sensor Failure Status (Boolean)

[D-10]: Alarm Triggered Status (Boolean)

[D-11]: Button Malfunction Status (Boolean)

[D-12]: Manual Power Disconnect Action (External)

[D-13]: Discharge Valve Open Status (Boolean)

[D-14]: Tank Empty Status (Boolean)

[D-15]: Valve Malfunction Status (Boolean)

[D-16]: Manual Empty Tank Action (External)

**Business Rules:**

(Inferred from use cases and specifications, using "unlocked" terminology)

[B-1]: Pump cannot start if not unlocked (i.e., latched) or tank is full.

[B-2]: Pump becomes latched (not unlocked) after starting.

[B-3]: Pump stops and becomes unlocked when high water level is detected or sensor fails.

[B-4]: Stop button makes the pump unlocked if it was latched (not unlocked).

[B-5]: Discharge valve can only be opened if tank is full.

### CTL/LTL Expressions (with Source Mapping):

1. **AG (****pump\_on -> locked)**

From：UC-1 BF3, [B-2]

1. **G ((start\_pb\_pressed\_event & !locked & !high\_level) -> X** **pump\_on &** **locked)**  
   From: [UC-1] Preconditions
2. **G ((stop\_pb\_pressed\_event &** **locked &** **pump\_on) -> X !pump\_on &** **!locked)**

From: [UC-2] Event Flow Scenario A, step 1 & 2

1. **G((sensor\_failure\_detected &** **pump\_on) -> X (alarm\_triggered & !pump\_on & !locked))**

From: [UC-1] Exception (Sensor Failure), [B-3]

1. **AG (discharge\_valve\_opened -> tank\_Is\_full)**

From: [UC-2] Precondition, BF1, [B-5]全局来看，在所有路径上，如果排放阀被打开，那么（在打开时）储罐必须是满的。

1. **G ((pump\_on & locked & high\_level) -> X (!locked & !pump\_on))**

From: [UC-1] Basic Flow steps 4, 5, 6, 7

1. **EF(Tank\_Is\_Empty)**

From：[UC-2] Postcondition。

一共7条

### Coffee Maker

**Use Case Name:**

Beverage Coffee Machine Control System

**Actors:**

Operator

**Use Cases:**

[UC-1]: Perform Emergency Stop – Immediately halts all operations, closes all valves, and stops the mixer upon operator command or system-detected critical fault.

[UC-2]: Start Machine – Initiates the system, performs self-checks, and transitions the machine to an idle state.

[UC-3]: Prepare Coffee and Milk Beverage – Fills with coffee and milk to 130ml, closes input valves, mixes for 4 seconds, then dispenses.

[UC-4]: Prepare Coffee Only Beverage – Fills with coffee to 130ml, closes coffee valve, then dispenses (no mixing).

**System Boundary**:

IEC 61131-3 Structured Text Program controlling the coffee machine hardware (buttons, valves, mixer, level sensor).

Relationships Among Actors:

The Operator interacts with the system by pressing the Emergency Stop Button ([UC-1]), Start Button ([UC-2]), Coffee and Milk Mode Button ([UC-3]), and Coffee Only Mode Button ([UC-4]).

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Perform Emergency Stop

(1) Brief Description:

Immediately halts any running functions, closes all valves (coffee, milk, output), and stops the mixer when the emergency stop button is pressed or a critical abnormality is detected.

(2) Preconditions:

– The machine is powered on.

(3) Event Flow:

Basic Flow (Operator Initiated):

1. The Operator presses the Emergency Stop Button.

2. The system immediately commands the coffee valve to close.

3. The system immediately commands the milk valve to close.

4. The system immediately commands the output valve to close.

5. The system immediately commands the mixer to stop.

6. The action is logged or displayed.

Alternative Flow(s):

– AF-1: System Initiated Emergency Stop

1. System detects a critical fault condition (e.g., valve stuck, level sensor error during operation, mixer malfunction).

2. Proceed with steps 2-6 of the Basic Flow.

– AF-2: Actuator Non-Response

1. If any actuator (mixer or valves) does not confirm closure/stop after command:

2. System logs a fault.

3. System reissues stop/close commands until confirmed off, or enters a deeper fault state.

(4) Post Conditions:

Success:

– The machine is in a safe, halted state with all specified actuators deactivated.

(5) Special Requirements:

– Response to emergency stop must be immediate.

Use Case: [UC-2] Start Machine

(1) Brief Description:

Initiates the system from standby, performs necessary self-checks, and transitions the machine to an idle state, ready for beverage selection.

(2) Preconditions:

– The system is in standby mode.

– No un-cleared critical fault conditions exist.

– Operator presses the Start Button.

(3) Event Flow:

Basic Flow:

1. System detects the Start Button press.

2. The system performs a self-test (e.g., checks sensors, valve positions, mixer status).

3. Upon a successful self-test:

4. The machine transitions to an idle state.

Alternative Flow(s):

– AF-1: Self-Test Failure

1. If any component fails during the self-test:

2. An error message is generated/logged.

3. Machine does not transition to idle state.

(4) Post Conditions:

Success:

– The machine is in idle state, ready to accept beverage mode commands.

Failure (if AF-1 occurs):

– An error is indicated, and the machine is not ready for operation.

(5) Special Requirements:

– Self-test must cover critical components.

Use Case: [UC-3] Prepare Coffee and Milk Beverage

(1) Brief Description:

Prepares a beverage combining coffee and milk. Fills mixer tank to 130ml, closes input valves, mixes for 4 seconds, then opens output valve to dispense.

(2) Preconditions:

– Machine is in idle state.

– Operator selects Coffee and Milk mode.

– All associated hardware is functional.

(3) Event Flow:

Basic Flow:

1. System detects Coffee and Milk mode selection.

2. System commands coffee valve to open.

3. System commands milk valve to open.

4. Mixer tank begins filling; sensors monitor liquid level.

5. When liquid level reaches 130 ml:

a. System commands coffee valve to close.

b. System commands milk valve to close.

c. System commands mixer to start.

d. A 4-second mixing timer is started.

6. Mixer runs.

7. When mixing timer reaches 4 seconds:

a. System commands mixer to stop.

8. After mixer stops, system commands output valve to open to dispense beverage.

9. After dispensing is complete (e.g., based on timer or sensor), output valve closes, and the system returns to idle state.

Alternative Flow(s):

– AF-1: Fill Level Not Reached

1. If the 130 ml level is not detected within an expected fill period:

2. An alarm is triggered, and the operation is halted.

3. Operator intervention is requested.

– AF-2: Mixer Malfunction

1. If the mixer stops prematurely or fails to start/run correctly:

2. An error is logged, and a safe recovery routine (possibly [UC-1]) is initiated.

(4) Post Conditions:

Success:

– Beverage is dispensed.

– System returns to idle state.

(5) Special Requirements:

– Mixing duration must be exactly 4 seconds.

– Liquid level for filling must be 130 ml.

Use Case: [UC-4] Prepare Coffee Only Beverage

(1) Brief Description:

Prepares a coffee-only beverage. Fills mixer tank with coffee to 130ml, closes coffee valve, then opens output valve to dispense (no mixing).

(2) Preconditions:

– Machine is in idle state.

– Operator selects Coffee Only mode.

– Coffee valve and associated sensors are functional.

(3) Event Flow:

Basic Flow:

1. System detects Coffee Only mode selection.

2. System commands coffee valve to open.

3. Mixer tank begins filling; sensors monitor liquid level.

4. When liquid level reaches 130 ml:

a. System commands coffee valve to close.

b. System commands output valve to open to dispense beverage.

5. After dispensing is complete, output valve closes, and the system returns to idle state.

Alternative Flow(s):

– AF-1: Fill Level Not Reached (Coffee Only)

1. If the 130 ml level is not detected within an expected fill period:

2. System performs brief additional rechecks.

3. If still unsuccessful, an alarm is raised, and operator intervention is requested.

– AF-2: Coffee Valve Malfunction

1. If the coffee valve fails to close properly:

2. An error is logged, and an emergency routine (possibly [UC-1]) may be triggered.

(4) Post Conditions:

Success:

– Beverage is dispensed.

– System returns to idle state.

(5) Special Requirements:

– Liquid level for filling must be 130 ml.

– No mixing cycle is performed in this mode.

**Priority of Use Cases:**

[UC-1] Perform Emergency Stop: High (Critical for safety)

[UC-2] Start Machine: High

[UC-3] Prepare Coffee and Milk Beverage: High

[UC-4] Prepare Coffee Only Beverage: Medium

**Data Requirements:**

[D-1]: Emergency Button Pressed Status (Event or persistent state until reset)

[D-2]: Start Machine Button Pressed Status (Event)

[D-3]: Coffee and Milk Mode Selected Status (Event or state indicating current mode)

[D-4]: Coffee Only Mode Selected Status (Event or state indicating current mode)

[D-5]: Max Level Reached Sensor Status (Boolean state, true when liquid at 130ml)

[D-6]: Coffee Valve State (Open/Closed)

[D-7]: Milk Valve State (Open/Closed)

[D-8]: Output Valve State (Open/Closed - also referred to as mixer\_valve in original constraints)

[D-9]: Mixer State (On/Off)

[D-10]: Mix Timer Value (Counts from 0 to 4 seconds during mixing)

[D-11]: Self-Test Passed Status (Boolean result of self-test)

[D-12]: System Operational State (Enum: Standby, Idle, Filling, Mixing, Dispensing, Fault, EmergencyStop)

[D-13]: Fault/Error Log Data

**Business Rules**:

[B-1]: Upon selection of Coffee and Milk mode and after the mixer tank's liquid level reaches the maximum level, both the milk valve and the coffee valve must be commanded to close in the next operational state.

[B-2]: Upon selection of Coffee and Milk mode and after the mixer tank's liquid level reaches the maximum level, the mixer must be commanded to start in the next operational state.

[B-3]: Once the mixer is on and has been running for 4 seconds (as indicated by the mix timer), it must be commanded to stop in the next operational state.

[B-4]: After the mixer has been on and the mix timer indicates 4 seconds of operation (implying mixing is complete and mixer is about to stop or has just stopped), the output valve (mixer\_valve) must eventually be commanded to open.

[B-5]: Upon selection of Coffee Only mode and after the mixer tank's liquid level reaches the maximum level, the coffee valve must be commanded to close, and the output valve (mixer\_valve) must be commanded to open in the next operational state.

[B-6]: If the emergency stop button is pressed, then in the next operational state, the coffee valve, milk valve, and output valve must all be commanded to close, and the mixer must be commanded to stop.

[B-7]: (Implied by UC-3 Postcondition/Basic Flow step 7) After the 4-second mixing cycle completes and the mixer is stopped, the mixer must remain off until a system reset condition for starting a new beverage cycle is met.

### CTL/LTL Expressions (with Source Mapping)

1. **G((coffee\_and\_milk\_mode & max\_level\_is\_reached) -> X(!milk\_valve\_is\_open & !coffee\_valve\_is\_open))**

From：[UC-3] Prepare Coffee and Milk Beverage, Event Flow, Basic Flow,

1. **G((coffee\_and\_milk\_mode & max\_level\_is\_reached) -> X** **mixer\_is\_on)**

From：[UC-3] Prepare Coffee and Milk Beverage, Event Flow, Basic Flow,

1. **G(****mixer\_is\_on & mix\_timer\_value = 4 -> mixer\_stop\_cmd)**

From：[UC-3] Prepare Coffee and Milk Beverage, Event Flow, Basic Flow, step 7a:Corresponds to Business Rule

1. **G(mixer\_is\_on & mix\_timer\_value = 4 -> X output\_valve\_open\_cmd)**

From：[UC-3] Prepare Coffee and Milk Beverage, Event Flow, Basic Flow, step 8  
**5. G((****coffee\_only\_mode & max\_level\_is\_reached) -> X(!coffee\_valve\_is\_open & output\_valve\_is\_open))**  
From：[UC-4] Prepare Coffee Only Beverage, Event Flow, Basic Flow  
**6.** **AG(emergency\_button\_is\_pressed -> (****coffee\_valve\_close\_cmd & milk\_valve\_close\_cmd & output\_valve\_close\_cmd & mixer\_stop\_cmd ))**  
From：[UC-1] Perform Emergency Stop, Event Flow, Basic Flow, step 1: "The Operator presses the Emergency Stop Button"  
**7. AG(!(coffee\_only\_mode & mixer\_is\_on))**  
From：[UC-4] 在所有路径上，当机器处于纯咖啡模式激活状态时，搅拌器绝不会运行。

一共7条，其中5条LTL, 2条CTL

### 9.Absolute Number

**Use Case Name:**

Calculate Absolute Value Function Block (Inferred)

**Actors:**

User (e.g., a technician or engineer who inputs the value for which the absolute value is to be calculated)

**Use Cases:**

[UC-1]: Calculate Absolute Value – Processes a numerical input to calculate and return its absolute value.

**System Boundary:**

The system boundary includes the function block within the 61131-3 environment that processes the input value and returns the absolute value.

**Relationships Among Actors**:

There is no direct relationship among actors since there is only one actor, the User. However, the User interacts with the boundary object, which is the function block.

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Calculate Absolute Value

(1) Brief Description:

This use case describes the process by which a User inputs a numerical value into the function block, and the system calculates and returns the absolute value of the input, ensuring that the result is always greater than or equal to zero.

(2) Preconditions: (Note: Original text had preconditions listed under Basic Flow and also separately. I'm consolidating them here as per template.)

– The function block is loaded and operational within the 61131-3 environment.

– The User must have access to the 61131-3 environment and the function block.

– The function block must be correctly integrated into the user's system.

(3) Event Flow:

Basic Flow:

1. User inputs a numerical value into the function block.

2. The system calculates the absolute value of the input.

3. The system returns the calculated absolute value.

Alternative Flow(s):

– AF-1: Invalid Numerical Input

1. (Precondition for this flow) The function block encounters an input value that is not a valid numerical type.

2. The system provides an error message indicating invalid input.

3. The system does not perform the absolute value calculation.

(4) Post Conditions:

Success (Basic Flow):

– The system returns the absolute value of the input value, which is greater than or equal to zero.

– The system has returned a non-negative value as the absolute value of the input.

Failure (Alternative Flow AF-1):

– The system provides an error message indicating invalid input.

– If an error occurred, the system is ready to accept a new input after the User acknowledges the error message.

(5) Special Requirements:

– The function block must perform calculations with high reliability and accuracy.

– The system should respond within a predefined time constraint to ensure performance.

– The function block should be usable by technicians with basic training, ensuring usability.

– The system must be scalable to handle a large number of requests if necessary.

**Priority of Use Cases:**

[UC-1] Calculate Absolute Value: High (Since this use case involves a core functionality of the function block)

**Data Requirements**:

[D-1]: Input Value (Numerical type, e.g., REAL, INT, LREAL, DINT)

[D-2]: Output Absolute Value (Same numerical type as input, non-negative)

[D-3]: Error Message (String or Enum, indicating invalid input type)

[D-4]: Input Type Validity Status (Boolean: true if valid numeric, false if non-numeric)

**Business Rules:**

[B-1]: The output of the absolute value calculation must always be greater than or equal to zero.

[B-2]: If the input value is not a valid numerical type, an error message must be provided, and no absolute value calculation should be performed.

[B-3]: The system must not attempt to calculate an absolute value if the input is non-numeric.

### CTL/LTL Expressions (with Source Mapping)

1. **AG(Input\_Valid -> AF(Absolute\_Value\_Output >= 0))**  
   From：[UC-1] Calculate Absolute Value
2. **AG(!Non\_Numeric\_Input)**

From：[UC-1] Calculate Absolute Value, Alternative Flow AF-1, step 3系统永远不能处于“输入是非数字”的状态

1. **AG((Input\_Is\_Non\_Numeric) -> AF (Error\_Message\_Generated))**

From：[UC-1] Calculate Absolute Value, Alternative Flow AF-1, step 2

一共3条CTL

### 10.Car Wash Station

**Use Case Name:**

Automatic Car Wash System

**Actors:**

Car

Person

Sensor System

Alarm System

Car Wash Control System

**Use Cases:**

[UC-1]: Initiate and Execute Normal Wash Cycle – Manages the standard car wash process (start, spray, scrub, dry) from vehicle entry to exit when no safety issues are present.

[UC-2]: Monitor for Person Presence – Continuously checks for people in the wash area during active wash operations.

[UC-3]: Handle Emergency Stop (Person Detected) – Stops all wash operations and activates an alarm if a person is detected in the wash area during a cycle.

**System Boundary:**

Car Wash Station, specifically the IEC 61131-3 Structured Text Program and its interface to sensors and actuators.

Relationships Among Actors:

The Sensor System provides input (Car presence, Person presence) to the Car Wash Control System.

The Car Wash Control System executes [UC-1] based on initial sensor inputs.

The Car Wash Control System executes [UC-2] concurrently with [UC-1]'s active wash phases (spray, scrub, dry).

If [UC-2] detects a person, it triggers [UC-3].

[UC-3] commands the Car Wash Control System to stop operations and activate the Alarm System.

The Car is the subject of [UC-1]. The Person is the subject of detection for [UC-2] and [UC-3].

**Detailed Descriptive Information for Each Use Case:**

Use Case: [UC-1] Initiate and Execute Normal Wash Cycle

(1) Brief Description:

This use case describes the standard, uninterrupted car wash process. It starts when a car is detected and no person is present, proceeds through the wash stages (start, spray, scrub, dry), and concludes with the car leaving and the system resetting.

(2) Preconditions:

– A car is detected by the Sensor System (car\_detected is true).

– No people are detected in the wash area by the Sensor System (!person\_detected is true).

– The Car Wash Control System is in an idle\_state.

(3) Event Flow:

Basic Flow:

1. Car Wash Control System verifies preconditions (car detected, no person, system idle).

2. Car Wash Control System initiates the "Start" operation/phase (current\_stage\_is\_start\_op). This may involve initial positioning or checks.

3. After the "Start" operation completes (e.g., start\_op\_complete event), system commands the "Spray" operation (current\_stage\_is\_spray).

4. After spray duration is met (e.g., spray\_time\_met event), system commands the "Scrub" operation (current\_stage\_is\_scrub).

5. After scrub duration is met (e.g., scrub\_time\_met event), system commands the "Dry" operation (current\_stage\_is\_dry).

6. After dry duration is met (e.g., dry\_time\_met event), the dry operation completes.

7. Car Wash Control System signals car is ready to leave (car\_is\_ready\_to\_leave state/event).

8. Sensor System or Car Wash Control System detects car leaves station (car\_has\_left\_station\_event).

9. Car Wash Control System resets to an idle\_state.

Alternative Flow(s):

– AF-1: Person Detected During Cycle (Handled by UC-3)

1. If [UC-2] Monitor for Person Presence signals a person detection during any active wash stage (start, spray, scrub, dry - steps 2-6):

2. This use case ([UC-1]) is immediately interrupted.

3. Control is effectively transferred to [UC-3] Handle Emergency Stop.

(4) Post Conditions:

Success:

– The car is washed according to the defined cycle (start, spray, scrub, dry).

– The car has left the station.

– The Car Wash Control System is in idle\_state.

– No alarm was triggered during this use case.

Interrupted (by AF-1):

– The wash cycle is stopped prematurely.

– Further state is determined by [UC-3].

(5) Special Requirements:

– Each wash stage (start, spray, scrub, dry) must complete its defined duration/conditions unless interrupted.

– The sequence of stages must be strictly followed: Start -> Spray -> Scrub -> Dry.

Use Case: [UC-2] Monitor for Person Presence

(1) Brief Description:

This use case describes the continuous monitoring for any person within the car wash area specifically when a car wash cycle ([UC-1]) is actively performing its wash stages (start, spray, scrub, dry).

(2) Preconditions:

– [UC-1] Initiate and Execute Normal Wash Cycle is in an active wash stage (i.e., wash\_cycle\_active is true, meaning current stage is start, spray, scrub, or dry).

– The Sensor System (personnel detection part) is operational.

(3) Event Flow:

Basic Flow:

1. While wash\_cycle\_active is true:

2. The Sensor System continuously checks for the presence of a person in the wash area.

3. If no person is detected, monitoring continues.

Alternative Flow(s):

– AF-1: Person Detected

1. If the Sensor System detects a person (person\_detected becomes true):

2. This use case signals the Car Wash Control System about the detection.

3. This triggers the execution of [UC-3] Handle Emergency Stop.

(4) Post Conditions:

Success (no person detected during a check cycle):

– Monitoring continues. [UC-1] proceeds.

Person Detected (AF-1):

– A person detection event is flagged.

– [UC-3] is invoked.

(5) Special Requirements:

– High-frequency and reliable person detection is critical.

Use Case: [UC-3] Handle Emergency Stop (Person Detected)

(1) Brief Description:

This use case is triggered when a person is detected in the wash area during an active car wash cycle. It ensures all operations are immediately halted and an alarm is activated.

(2) Preconditions:

– A car wash cycle is in progress (wash\_cycle\_active is true).

– [UC-2] Monitor for Person Presence has signaled that a person has been detected (person\_detected is true).

(3) Event Flow:

Basic Flow:

1. Car Wash Control System receives person detection signal (while wash\_cycle\_active).

2. Car Wash Control System immediately interrupts any active stage of [UC-1].

3. Car Wash Control System commands all wash actuators (start, spray, scrub, dry mechanisms) to stop (operations\_halted state).

4. Car Wash Control System activates the Alarm System (alarm\_triggered state).

5. System logs the emergency event.

Alternative Flow(s): (Could include details from your first "精化后" UC3's alternative flow if desired, e.g., secondary shutdown attempt)

– AF-1: Primary Shutdown Fails

1. If the primary command to halt operations fails to achieve the operations\_halted state promptly:

2. System attempts a secondary shutdown mechanism.

3. Log failure and escalation.

(4) Post Conditions:

Success:

– All car wash operations are stopped.

– The alarm is active.

– The system is in a safe, halted state (e.g., emergency\_stop\_state).

(5) Special Requirements:

– Stoppage of operations must be immediate.

– Alarm activation must be clear.

**Priority of Use Cases:**

[UC-1] Initiate and Execute Normal Wash Cycle: medium

[UC-2] Monitor for Person Presence: High

[UC-3] Handle Emergency Stop (Person Detected):High

**Data Requirements:**

[D-1]: Car Detected Status (Boolean: car\_detected)

[D-2]: Person Detected Status (Boolean: person\_detected)

[D-3]: Wash Cycle Active Status (Boolean: wash\_cycle\_active - true if current\_stage is StartOp, Spray, Scrub, or Dry)

[D-4]: Current Wash Stage (Enum: Idle, StartOp, Spray, Scrub, Dry, DryComplete\_ReadyToLeave, EmergencyStop, SystemReset)

[D-5]: Operations Halted Status (Boolean: operations\_halted)

[D-6]: Alarm Triggered Status (Boolean: alarm\_triggered)

[D-7]: Car Has Left Station Event (Event: car\_has\_left\_station\_event)

[D-8]: Start Operation Complete Event (Event: start\_op\_complete)

[D-9]: Spray Time Met Event (Event: spray\_time\_met)

[D-10]: Scrub Time Met Event (Event: scrub\_time\_met)

[D-11]: Dry Time Met Event (Event: dry\_time\_met)

[D-12]: Personnel Sensor Check Event (Event: person\_sensor\_check\_event)

[D-13]: Primary Shutdown Failed Event (Event: primary\_shutdown\_failed)

[D-14]: Secondary Shutdown Attempted Event (Event: secondary\_shutdown\_attempted)

Business Rules:

[B-1]: The car wash cycle ([UC-1]) can only begin if a car is detected AND no person is detected AND the system is in an idle state.

[B-2]: If a person is detected (via [UC-2]) while a wash cycle ([UC-1]) is active, then [UC-3] must be invoked to immediately stop all wash operations AND trigger an alarm.

[B-3]: The normal wash cycle ([UC-1]) follows the sequence: StartOp -> Spray -> Scrub -> Dry.

[B-4]: During an active wash cycle ([UC-1]), continuous monitoring for person presence ([UC-2]) must occur.

[B-5]: Each wash stage (StartOp, Spray, Scrub, Dry) has a defined completion condition (e.g., timer elapsed or other conditions met) before transitioning to the next stage or completing the cycle.

### CTL/LTL Expressions (with Source Mapping)

1. **AG((car\_detected &** **!person\_detected & current\_stage\_is\_idle) ->** **car\_wash\_starts & wash\_cycle\_active)**

From: [UC-1] Preconditions; [UC-1] Basic Flow step 1 & 2; Business Rule [B-1].

1. **G((car\_wash\_starts & start\_op\_complete &** **!person\_detected) -> X current\_stage\_is\_spray)**

From：[UC-1] Basic Flow step 2 to 3; Business Rule [B-3], [B-5].

1. **G((current\_stage\_is\_spray & spray\_time\_met** **& !person\_detected) -> X current\_stage\_is\_scrub)**

From: [UC-1] Basic Flow step 3 to 4; Business Rule [B-3], [B-5].

1. **G((current\_stage\_is\_scrub & scrub\_time\_met & !person\_detected) -> X current\_stage\_is\_dry)**

From: [UC-1] Basic Flow step 4 to 5; Business Rule [B-3], [B-5].

1. **AG((current\_stage\_is\_dry & dry\_time\_met) -> car\_ready\_to\_leave)**

From: [UC-1] Basic Flow step 5 to 6 & 7; Business Rule [B-3], [B-5].

1. **AG((****wash\_cycle\_active & person\_detected) -> (operations\_halted & alarm\_triggered))**

From: [UC-3] Preconditions; [UC-3] Basic Flow steps 1-4; Business Rule [B-2].

1. **G(car\_has\_left\_station\_even -> X (System\_idle )**

From: [UC-1] Basic Flow steps 7, 8, 9.

1. **AG(wash\_cycle\_active -> person\_sensor\_check\_event)**

From: [UC-2] Basic Flow step 1 & 2; Business Rule [B-4].

一共8条，1条LTL, 7条CTL