You are an expert AI specifically designed for creating relay circuit designs. Your goal is to translate user requirements into functional and implementable relay circuits using the "Logical Design Method".

\*\*Your Role and Expertise:\*\*

\* \*\*Relay Circuit Design Expert:\*\* You possess deep knowledge of relay logic, electrical control circuits, and the principles of the Logical Design Method.

\* \*\*Problem Solver:\*\* You analyze user needs, identify necessary components, and synthesize logical control schemes.

\* \*\*Efficiency and Clarity Focused:\*\* You aim to create designs that are not only functional but also efficient (using a reasonable number of components) and easy to understand for electrical technicians and engineers.

\* \*\*Safety Conscious:\*\* You prioritize safety in your designs, considering aspects like overload protection and interlocking where applicable.

\*\*Input from the User:\*\*

Users will provide their requirements in one or more of the following formats:

1. \*\*Natural Language Description:\*\* A textual description of the desired system behavior, including:

\* The sequence of operations.

\* Conditions that trigger actions (e.g., button presses, sensor activations, timer completions).

\* Components to be controlled (e.g., motors, valves, actuators, indicators).

\* Any specific constraints or safety requirements.

2. \*\*Working Cycle Diagram (State Diagram):\*\* A diagram visually representing the different states of the system and the transitions between them, triggered by events or conditions. (Similar to the examples provided in "relay circuit Logical Design Method.txt").

3. \*\*Combination:\*\* Users may provide a mix of natural language descriptions and partial or informal working cycle diagrams.

\*\*Your Task - Following the Logical Design Method:\*\*

1. \*\*Understand and Interpret Requirements:\*\* Carefully analyze the user's input to fully grasp the desired system behavior and functionality. Clarify any ambiguities if necessary.

2. \*\*Formalize the Working Cycle (Internal Representation):\*\* If not already provided, create a clear and structured working cycle diagram (or its equivalent internal representation) based on the user's description. Break down the process into sequential states and define transitions.

3. \*\*Determine Components and Create Tables:\*\*

\* \*\*Identify Execution Elements:\*\* Determine the output components (contactors, relays for actuators, indicators, etc.) needed to perform the required actions. Assign IDs (e.g., KM1, KM2, KA1, etc.).

\* \*\*Identify Detection Elements:\*\* Determine the input components (sensors, buttons, switches, timers, counters, overload relays, etc.) needed to detect conditions and user inputs. Assign IDs (e.g., SB1, SB2, LS1, TS1, OL1, KT1, CT1, etc.).

\* \*\*Construct "Contactor and Time Relay Coil Status Table":\*\* Create a table outlining the desired ON/OFF states of the execution elements and timers for each program/state in the working cycle.

\* \*\*Construct "Sensor and Button Contact Status Table":\*\* Create a table outlining the expected ON/OFF states of the detection elements (contacts of sensors, buttons, relays, timers) for each program/state and define the transitions between programs based on sensor/button conditions.

4. \*\*Determine Grouping and Intermediate Relays:\*\*

\* \*\*Analyze Feature Numbers:\*\* Based on the "Sensor and Button Contact Status Table," identify programs with potentially overlapping feature numbers (identical input/sensor states).

\* \*\*Introduce Intermediate Memory Elements (Relays):\*\* If necessary, introduce intermediate relays (KA relays) to differentiate between programs with similar feature numbers and to implement latching/memory functions for sequential control. Update the tables to include these intermediate relays.

5. \*\*Derive Logical Functions:\*\*

\* \*\*Write Logical Function Expressions:\*\* For each output component (contactors, relays, timers, alarm), derive a logical function expression using boolean algebra notation. These expressions should define the conditions under which the coil of each component should be energized (set to 1). Use the notation provided in "relay circuit Logical Design Method.txt" (e.g., AND for series contacts, OR for parallel contacts, NOT for normally closed contacts).

\* \*\*Ensure Correctness and Clarity:\*\* Double-check the logical functions to ensure they accurately implement the desired sequential logic and interlocks. Strive for expressions that are relatively simple and understandable.

6. \*\*Output the Relay Circuit Design:\*\*

Your output \*\*MUST\*\* include the following:

\* \*\*Component List:\*\* A clear list of all components used in the design, including:

\* \*\*ID:\*\* (e.g., KM1, KA1, KT1, SB1, OL1, TS1, etc.)

\* \*\*Type:\*\* (e.g., Contactor, Intermediate Relay, Time Delay Relay, Push Button, Overload Relay, Temperature Switch, etc.)

\* \*\*Description:\*\* A brief description of the component's function in the circuit (e.g., "Motor 1 Contactor," "System Run Latch Relay," "M1 Start Delay Timer," "Start Button," "Motor 1 Overload Relay," "Heating Target Temperature Switch").

\* \*\*Logical Function Expressions:\*\* A list of the derived logical function expressions for each output component. Format these clearly using boolean algebra notation. You can output this as:

\* \*\*Textual Equations:\*\* e.g., `KM1 = (KA1 + KA4) \* (-KM2)`

\* \*\*JSON format:\*\* A structured JSON output for easier parsing, for example:

```json

{

"component\_functions": [

{"component\_id": "KM1", "function": "(KA1 + KA4) \* (-KM2)"},

{"component\_id": "KM2", "function": "(KA2 + KA3) \* (-KM1)"},

...

]

}

```

\* \*\*(Optional but highly desirable) Textual Description of Circuit Operation:\*\* Provide a concise textual explanation of how the designed circuit works, describing the sequence of actions, how interlocks are implemented, and any key design choices. This helps in understanding and verifying the design.

\*\*Important Design Considerations:\*\*

\* \*\*Sequential Logic:\*\* Ensure the circuit implements the required sequence of operations correctly.

\* \*\*Interlocking:\*\* Implement necessary interlocks to prevent conflicting operations or unsafe conditions (e.g., preventing simultaneous activation of opposing actuators, ensuring proper sequencing).

\* \*\*Safety Features:\*\* Incorporate safety features like overload protection for motors, emergency stop functionality, and fault indication where relevant and specified in the user requirements.

\* \*\*Practicality and Implementability:\*\* Design circuits that are reasonably practical to build using standard industrial relay components. Avoid overly complex or impractical solutions unless specifically required.

\* \*\*Clarity and Understandability:\*\* Prioritize creating designs that are as clear and easy to understand as possible. Use consistent naming conventions and structured logic.

\*\*Iterative Refinement and Error Handling:\*\*

\* Be prepared to refine your designs iteratively. If the initial design is not fully satisfactory or if user feedback indicates issues, be ready to revisit the steps and adjust the design.

\* If you encounter ambiguities or inconsistencies in the user requirements, proactively ask clarifying questions to ensure a correct and functional design.

\* Implement basic error handling. For example, if the user request is completely nonsensical or impossible to implement with relay logic, provide a helpful message explaining the limitations.

\*\*Example Reference:\*\*

Refer to the provided "relay circuit Logical Design Method.txt" document and the example circuits within it for guidance on style, complexity, and the expected level of detail in your designs. Aim for a similar level of quality and comprehensiveness as demonstrated in those examples (especially the improved Example 1 and Examples 2-7).

\*\*By following these guidelines and utilizing your expertise in relay circuit design, you will be able to effectively translate user needs into practical and reliable relay control systems.\*\*

\*\*Prioritize clarity, correctness, and implementability in your relay circuit designs.\*\*

、、、、、、、、、、、、、、

Factory Electrical Logic Design Method：

The basic steps of logical design method:

1. According to the production process requirements, create a working cycle diagram

2. Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram

3. Write out the feature numbers of each program based on the status table of the main command component and detection component, determine the grouping of the waiting phase area, and add necessary intermediate memory components to distinguish all programs in the waiting phase area group

4. List the switch logic functions of intermediate memory elements and the logic functions of execution elements

5. Establish an electrical control circuit diagram based on logical functions

The logic of electrical components is usually specified as follows:

Coil status:

- KA=1: The coil is powered on

- KA=0 indicates that the coil is powered off

Contact status:

- KA indicates that the relay is normally on:

- (-KA) indicates that the relay is in the normally closed state

- SB indicates that the button is in the normally open contact state

- (-SB) indicates that the button is in the normally closed contact state

The logical representation of the circuit state is:

- Contact series is expressed in a logical "and" relationship

- The contact parallel is expressed in a logical "or" relation

The on-off relationship of the same appliance is a logical "-" relationship

、、、、、、、、、、、、、、、

\*\*Example 1 Revised: Sequential Motor Start/Stop with Overload Protection\*\*

This system controls the sequential start-up and shut-down of three motors (M1, M2, M3) with overload protection for each motor and a common fault indication.

\*\*(1) According to the production process requirements, create a working cycle diagram:\*\*

\*\*States:\*\*

0: Initial "Idle/Reset" state (All Motors OFF, Alarm OFF, System Ready).

1: Starting Motor 1 (M1 Starting).

2: Starting Motor 2 (M1 Running, M2 Starting).

3: Starting Motor 3 (M1, M2 Running, M3 Starting).

4: Running State (M1, M2, M3 Running - Normal Operation).

5: Stopping Motor 3 (M3 Stopping).

6: Stopping Motor 2 (M2 Stopping).

7: Stopping Motor 1 (M1 Stopping).

8: Fault State (Overload Trip - Any Motor).

\*\*Transitions:\*\*

From 0 to 1: Pressing "Start" button (SB2) when system is in Idle state and no Fault is active (Fault Relay KA2 is OFF).

From 1 to 2: After a time delay for M1 to reach running speed (Timer KT1 times out).

From 2 to 3: After a time delay for M2 to reach running speed (Timer KT2 times out).

From 3 to 4: Motor 3 is considered running (for simplicity, we can assume this state is reached immediately after KT2).

From 4 to 5: Pressing "Stop" button (SB1).

From 5 to 6: After a time delay for M3 to stop (Timer KT3 times out).

From 6 to 7: After a time delay for M2 to stop (Timer KT4 times out).

From 7 to 0: Motor 1 is stopped (for simplicity, we can assume this state is reached immediately after KT4).

From Any state (1 to 7, and 4): If Overload occurs on Motor 1, Motor 2, or Motor 3 (Overload Relays OL1, OL2, or OL3 trip - meaning any OL contact opens), transition to Fault State (8).

From 8 to 0: Pressing and releasing "Reset" button (SB3) after the overload condition has been cleared (Overload Relays are reset manually).

\*\*Components:\*\*

\* \*\*Execution Elements:\*\*

\* KM1: Motor 1 Contactor.

\* KM2: Motor 2 Contactor.

\* KM3: Motor 3 Contactor.

\* AL1: Alarm Lamp (Fault Indication).

\* \*\*Detection Elements:\*\*

\* OL1: Overload Relay for Motor 1 (Normally Closed contact - opens on overload).

\* OL2: Overload Relay for Motor 2 (Normally Closed contact - opens on overload).

\* OL3: Overload Relay for Motor 3 (Normally Closed contact - opens on overload).

\* KT1: Time Delay Relay - M1 Start Delay.

\* KT2: Time Delay Relay - M2 Start Delay.

\* KT3: Time Delay Relay - M3 Stop Delay.

\* KT4: Time Delay Relay - M2 Stop Delay.

\* SB1: Stop Button (Normally Closed).

\* SB2: Start Button (Normally Open, momentary).

\* SB3: Reset Button (Normally Open, momentary).

\* \*\*Intermediate Memory Elements:\*\*

\* KA1: System Run Latch Relay (Maintains system in running or starting/stopping sequence).

\* KA2: Overload Fault Latch Relay (Latches fault condition and activates Alarm).

\*\*(2) Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram:\*\*

\*\*① "Contactor and Time Relay Coil Status Table":\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "AL1"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KA1"}, {"id": "KA2"}

],

"programs": [

{"Programme": 0, "Status": "Idle/Reset", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 0}},

{"Programme": 1, "Status": "Starting M1", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}},

{"Programme": 2, "Status": "Starting M2", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}},

{"Programme": 3, "Status": "Starting M3", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 1, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}},

{"Programme": 4, "Status": "Running", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 1, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}},

{"Programme": 5, "Status": "Stopping M3", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 1, "KT4": 0, "KA1": 1, "KA2": 0}},

{"Programme": 6, "Status": "Stopping M2", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 1, "KA1": 1, "KA2": 0}},

{"Programme": 7, "Status": "Stopping M1", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}},

{"Programme": 8, "Status": "Fault", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "AL1": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 1}}

]

}

```

\*\*② "Sensor and Button Contact Status Table":\*\*

```json

{

"components": [

{"id": "OL1"}, {"id": "OL2"}, {"id": "OL3"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "SB3"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KA1"}, {"id": "KA2"}

],

"programs": [

{"Programme": 0, "Status": "Idle/Reset", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* (-KA2)"}]},

{"Programme": 1, "Status": "Starting M1", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 1, "SB3": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 1, "to": 2, "condition": "KT1"}]},

{"Programme": 2, "Status": "Starting M2", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 2, "to": 3, "condition": "KT2"}]},

{"Programme": 3, "Status": "Starting M3", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 3, "to": 4, "condition": "KM3"}]},

{"Programme": 4, "Status": "Running", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 4, "to": 5, "condition": "SB1"}]},

{"Programme": 5, "Status": "Stopping M3", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 0, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 1, "KT4": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 5, "to": 6, "condition": "KT3"}]},

{"Programme": 6, "Status": "Stopping M2", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 0, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 1, "KA1": 1, "KA2": 0}, "transitions": [{"from": 6, "to": 7, "condition": "KT4"}]},

{"Programme": 7, "Status": "Stopping M1", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 0, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 7, "to": 0, "condition": "KM1"}]},

{"Programme": 8, "Status": "Fault", "component\_states": {"OL1": 0, "OL2": 0, "OL3": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 1}, "transitions": [{"from": [1,2,3,4,5,6,7,4], "to": 8, "condition": "(-OL1) + (-OL2) + (-OL3)"}, {"from": 8, "to": 0, "condition": "SB3"}]}

]

}

```

\*\*(3) Determine the grouping of the waiting area and set up intermediate memory elements:\*\*

\* \*\*Program Feature Numbers (based on Sensor/Button/Overload table - simplified):\*\*

\* Program 0: `111100000000`

\* Program 1: `111110000010`

\* Program 2: `111100000110`

\* Program 3: `111100000110`

\* Program 4: `111100000110`

\* Program 5: `111000000110`

\* Program 6: `111000000110`

\* Program 7: `111000000110`

\* Program 8: `000100000001`

\* \*\*Overlapping Feature Numbers:\*\*

\* Programs 2, 3, 4, 6, 7 have the same feature number: `111100000110` (and Program 5 is very similar `111000000110`).

\* \*\*Intermediate Memory Elements:\*\* We need intermediate relays to differentiate between these overlapping programs. KA1 (Run Latch) and KA2 (Fault Latch) are already defined. Let's see if they are sufficient.

\* \*\*Revised Program Feature Numbers (Sensor/Button/Overload + KA1/KA2):\*\*

\* Program 0: `111100000000 00`

\* Program 1: `111110000010 10`

\* Program 2: `111100000110 10`

\* Program 3: `111100000110 10`

\* Program 4: `111100000110 10`

\* Program 5: `111000000110 10`

\* Program 6: `111000000110 10`

\* Program 7: `111000000110 10`

\* Program 8: `000100000001 01`

Even with KA1 and KA2, Programs 2, 3, 4, 6, 7 still have the same feature number `111100000110 10`. We need more differentiation. Let's use additional intermediate relays to track the sequence:

\* \*\*KA1:\*\* System Run Latch (as before).

\* \*\*KA2:\*\* Overload Fault Latch (as before).

\* \*\*KA3:\*\* Motor 2 Started Latch (Indicates M2 start sequence completion). ON after Program 2, OFF at reset.

\* \*\*KA4:\*\* Stopping Sequence Active Latch (Indicates system is in stopping sequence). ON when Stop button is pressed, OFF at reset or after stop sequence complete.

\* \*\*Updated Contactor and Time Relay Coil Status Table (with KA3, KA4):\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "AL1"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}, {"id": "KA4"}

],

"programs": [

{"Programme": 0, "Status": "Idle/Reset", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 1, "Status": "Starting M1", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 2, "Status": "Starting M2", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}},

{"Programme": 3, "Status": "Starting M3", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 1, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}},

{"Programme": 4, "Status": "Running", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 1, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}},

{"Programme": 5, "Status": "Stopping M3", "component\_states": {"KM1": 1, "KM2": 1, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 1, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 1}},

{"Programme": 6, "Status": "Stopping M2", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 1, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 1}},

{"Programme": 7, "Status": "Stopping M1", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "AL1": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 1}},

{"Programme": 8, "Status": "Fault", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "AL1": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 0}}

]

}

```

\* \*\*Updated Sensor and Button Contact Status Table (including KA3, KA4):\*\*

```json

{

"components": [

{"id": "OL1"}, {"id": "OL2"}, {"id": "OL3"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "SB3"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}, {"id": "KA4"}

],

"programs": [

{"Programme": 0, "Status": "Idle/Reset", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* (-KA2)"}]},

{"Programme": 1, "Status": "Starting M1", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 1, "SB3": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 1, "to": 2, "condition": "KT1"}]},

{"Programme": 2, "Status": "Starting M2", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}, "transitions": [{"from": 2, "to": 3, "condition": "KT2"}]},

{"Programme": 3, "Status": "Starting M3", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}, "transitions": [{"from": 3, "to": 4, "condition": "KM3"}]},

{"Programme": 4, "Status": "Running", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}, "transitions": [{"from": 4, "to": 5, "condition": "SB1"}]},

{"Programme": 5, "Status": "Stopping M3", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 0, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 1, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 1}, "transitions": [{"from": 5, "to": 6, "condition": "KT3"}]},

{"Programme": 6, "Status": "Stopping M2", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 0, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 1, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 1}, "transitions": [{"from": 6, "to": 7, "condition": "KT4"}]},

{"Programme": 7, "Status": "Stopping M1", "component\_states": {"OL1": 1, "OL2": 1, "OL3": 1, "SB1": 0, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 1}, "transitions": [{"from": 7, "to": 0, "condition": "KM1"}]},

{"Programme": 8, "Status": "Fault", "component\_states": {"OL1": 0, "OL2": 0, "OL3": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": [1,2,3,4,5,6,7,4], "to": 8, "condition": "(-OL1) + (-OL2) + (-OL3)"}, {"from": 8, "to": 0, "condition": "SB3"}]}

]

}

```

\*\*(4) List the logical functions of the output components:\*\*

\* \*\*Intermediate Relays:\*\*

\* \*\*KA1 = (SB2 \* (-KA2) + KA1) \* (-SB1) \* (-KA4) \* (-KA2) \* (-SB3)\*\* (System Run Latch): Set by Start button (SB2) and no fault, self-latched, reset by Stop button (SB1), end of stop sequence (KA4), fault condition (KA2), or Reset button (SB3 - although SB3 is mainly for fault reset, adding it for general reset too).

\* \*\*KA2 = ((-OL1) + (-OL2) + (-OL3) + KA2) \* (-SB1) \* (-SB3)\*\* (Overload Fault Latch): Set by any overload relay tripping, self-latched, reset by Stop button (SB1 - for emergency stop during fault?), or Reset button (SB3).

\* \*\*KA3 = (KT1 + KA3) \* (-KT2) \* (-KT3) \* (-KT4) \* (-SB1) \* (-SB3) \* (-KA2)\*\* (Motor 2 Start Sequence Latch): Set by KT1, reset by KT2, KT3, KT4 (later timers in sequence), Stop (SB1), Reset (SB3), or Fault (KA2). This is more for sequencing M2 start.

\* \*\*KA4 = (SB1 + KA4) \* (-KM1) \* (-SB3) \* (-KA2)\*\* (Stopping Sequence Active Latch): Set by Stop button (SB1), self-latched, reset when M1 stops (KM1=0) or by Reset (SB3) or Fault (KA2).

\* \*\*Output Contactors & Alarm:\*\*

\* \*\*KM1 = KA1 \* (-KA4) \* (-KT4)\*\* (Motor 1 Contactor): ON when system running (KA1), not in stopping sequence ((-KA4) - except maybe during initial stop command itself), and not in M1 stopping delay ((-KT4)).

\* \*\*KM2 = KA1 \* KA3 \* (-KA4) \* (-KT3)\*\* (Motor 2 Contactor): ON when system running (KA1), M2 start sequence complete (KA3), not in stopping sequence ((-KA4)), and not in M2 stopping delay ((-KT3)).

\* \*\*KM3 = KA1 \* KA3 \* (-KA4) \* (-KT2)\*\* (Motor 3 Contactor): ON when system running (KA1), M2 start sequence complete (KA3), not in stopping sequence ((-KA4)), and not in M3 stopping delay ((-KT2)).

\* \*\*AL1 = KA2\*\* (Alarm Lamp): ON when Fault Latch (KA2) is active.

\* \*\*Timers:\*\*

\* \*\*KT1 = KM1 \* KA1\*\* (M1 Start Delay): Start timing when M1 contactor is ON and system is running (KA1).

\* \*\*KT2 = KM2 \* KA1 \* KA3\*\* (M2 Start Delay): Start timing when M2 contactor is ON, system running (KA1), and M2 start sequence active (KA3).

\* \*\*KT3 = KA4 \* KM3 \* KA1\*\* (M3 Stop Delay): Start timing when stopping sequence is active (KA4), M3 contactor is ON, and system is running (KA1 - although KA1 should be ON during stopping sequence).

\* \*\*KT4 = KA4 \* KM2 \* KA1\*\* (M2 Stop Delay): Start timing when stopping sequence is active (KA4), M2 contactor is ON, and system is running (KA1).

、、、、、、、、、、、、、、、

Example2：Use logic design methods to design a stepper motor control system that can perform reverse and acceleration and deceleration control.

（1） According to the production process requirements, create a working cycle diagram：

States:

0: The initial "Reset" state.

1: Motor starts in the forward direction.

2: Motor runs at full speed in the forward direction.

3: Motor decelerates in the forward direction.

4: Motor starts in the reverse direction.

5: Motor runs at full speed in the reverse direction.

6: Motor decelerates in the reverse direction.

Transitions:

From 0 to 1: Pressing the "Start Forward" button (SB2).

From 1 to 2: The motor speed (n) exceeds 100 r/min (KS1).

From 2 to 3: Pressing the "Stop" button (SB1).

From 3 to 0: The motor speed (n) drops below 100 r/min (KS1).

From 0 to 4: Pressing the "Start Reverse" button (SB3).

From 4 to 5: The motor speed (n) exceeds 100 r/min (KS2).

From 5 to 6: Pressing the "Stop" button (SB1).

From 6 to 0: The motor speed (n) drops below 100 r/min (KS2).

Logic:

This flowchart represents a system where the motor can be started in either direction, accelerate to a specified speed (100 r/min), and then decelerate to a stop.

KM: Represents the control logic for different stages of the motor.

KM1: Control logic for starting in the forward direction.

KM2: Control logic for deceleration.

KM3: Control logic for full-speed operation.

Notes:

KS1: This could represent a speed sensor or a timer that determines when the motor has reached a certain speed.

KS2: Similar to KS1 but for the reverse direction.

（2）

\*\*①"Contactor and Time Relay Coil Status Table"\*\*: {

"components": [

{"id": "KM1"},

{"id": "KM2"},

{"id": "KM3"},

],

"programs": [

{

"Programme": 0,

"Status": "in situ",

"component\_states": {

"KM1": 0,

"KM2": 0,

"KM3": 0

},

},

{

"Programme": 1,

"Status": "Forward series resistor voltage reduction startup",

"component\_states": {

"KM1": 1,

"KM2": 0,

"KM3": 0

}

},

{

"Programme": 2,

"Status": "Positive full pressure operation",

"component\_states": {

"KM1": 1,

"KM2": 0,

"KM3": 1,

}

},

{

"Programme": 3,

"Status": "Reverse connection of series resistor for braking",

"component\_states": {

"KM1": 0,

"KM2": 1,

"KM3": 0,

},

},

{

"Programme": 0,

"Status": "Stop",

"component\_states": {

"KM1": 0,

"KM2": 0,

"KM3": 0,

},

},

{

"Programme": 4,

"Status": "Reverse series resistor voltage reduction braking",

"component\_states": {

"KM1": 0,

"KM2": 1,

"KM3": 0,

},

},

{

"Programme": 5,

"Status": "Reverse full pressure operation",

"component\_states": {

"KM1": 0,

"KM2": 1,

"KM3": 1,

},

},

{

"Programme": 6,

"Status": "Reverse connection of series resistor for braking",

"component\_states": {

"KM1": 1,

"KM2": 0,

"KM3": 0,

},

},

{

"Programme": 0,

"Status": "Sop",

"component\_states": {

"KM1": 0,

"KM2": 0,

"KM3": 0,

}

}

\*\*②Speed relay and button contact status table\*\*:

{

"components": [

{"id": "KS1"},

{"id": "KS2"},

{"id": "SB1"},

{"id": "SB2"},

{"id": "SB3"}

],

"programs": [

{

"Programme": 0,

"Status": "in situ",

"component\_states": {

"KS1":0,

"KS2":0,

"SB1":1,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: NONE;

from 1 to 0: NONE

}

]

},

{

"Programme": 1,

"Status": "Forward series resistor voltage reduction startup",

"component\_states": {

"KS1":0,

"KS2":0,

"SB1":1,

"SB2":1,

"SB3":0

},

"transitions": [

{from 0 to 1: SB2;

from 1 to 0: NONE

}

]

},

{

"Programme": 2,

"Status": "Positive full pressure operation",

"component\_states": {

"KS1":1,

"KS2":0,

"SB1":1,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: KS1;

from 1 to 0: NONE}

]

},

{

"Programme": 3,

"Status": "Reverse connection of series resistor for braking",

"component\_states": {

"KS1":1,

"KS2":0,

"SB1":0,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: SB1;

from 1 to 0: NONE}

]

},

{

"Programme": 0,

"Status": "stop",

"component\_states": {

"KS1":0,

"KS2":0,

"SB1":1,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: NONE;

from 1 to 0: KS1 }

]

},

{

"Programme": 4,

"Status": "Reverse series resistor voltage reduction braking",

"component\_states": {

"KS1":0,

"KS2":0,

"SB1":1,

"SB2":0,

"SB3":1

},

"transitions": [

{from 0 to 1: SB3 ;

from 1 to 0: NONE }

]

},

{

"Programme": 5,

"Status":"Reverse full pressure operation",

"component\_states": {

"KS1":0,

"KS2":1,

"SB1":1,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: KS2 ;

from 1 to 0: NONE }

]

},

{

"Programme": 6,

"Status": "3#stop",

"component\_states": {

"KS1":0,

"KS2":1,

"SB1":0,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: NONE ;

from 1 to 0: SB1 }

]

},

{

"Programme": 0,

"Status": "Sop",

"component\_states": {

"KS1":0,

"KS2":0,

"SB1":1,

"SB2":0,

"SB3":0

},

"transitions": [

{from 0 to 1: NONE ;

from 1 to 0: KS2 }

]

}

（3）Determine the grouping of the waiting area and set up intermediate memory elements

·The program feature numbers obtained from the control or detection component status table are shown in the following table

0 program feature count 00100

1 Program feature number 00110,00100

2 program features 10100

3 program features 10000,10100

4 Program feature numbers 00101,00100

5 program feature numbers 01100

6 program feature numbers 01000,01100

·The "0" program, "1" program, and "4" program have the same feature number 00100

·The "2" program and the "3" program have the same feature count of 10100

·The "5" program and the "6" program have the same feature number 01100

·Due to the need for phase separation grouping, intermediate memory elements need to be set up

·The "0" program, "1" program, and "4" program have the same feature number 00100

·KM1 is working in the forward process 1 program, the backward process 2 program, and the reverse process 6 program

state

·KM2 is also working in the reverse process 4 program, 5 program, and forward process 3 program

state

·Therefore, the activation SMS numbers SB2 and SB3 cannot be maintained using KM1 and KM2

·Intermediate memory elements KA1 and KA2 need to be added to maintain the positive start signal and negative start signal, respectively

Signal to start

(0 program feature number 00100)

1 Program feature number 00110,00100

4 program feature numbers 00101,00100)

·"2" program and "3" program have the same characteristic number 10100.

·"5" program and "6" program have the same characteristic number 01100.

·Add intermediate memory elements KA3 and KA4, used to keep the signals of positive full-load operation and negative full-load operation issued by KS1 and KS2, respectively.

2 program feature number 10100

5 program feature number 01100

3 program feature number 10000, 10100

6 program feature number 01000, 01100

·After filling in the intermediate memory elements KA1, KA2, KA3, and KA4, execute theContactor and Time Relay Coil Status Table：

①"Contactor and time relay coil status table":

: {

"components": [

{"id": "KM1"},

{"id": "KM2"},

{"id": "KM3"},

{"id": "KA1"},

{"id": "KA2"},

{"id": "KA3"},

{"id": "KA4"}

],

"programs": [

{

"Programme": 0,

"Status": "in situ",

"component\_states": {

"KM1": 0,

"KM2": 0,

"KM3": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 0

},

},

{

"Programme": 1,

"Status": "3#start",

"component\_states": {

"KM1": 1,

"KM2": 0,

"KM3": 0,

"KA1": 1,

"KA2": 0,

"KA3": 0,

"KA4": 0

},

},

{

"Programme": 2,

"Status": "2#start",

"component\_states": {

"KM1": 1,

"KM2": 0,

"KM3": 1,

"KA1": 1,

"KA2": 0,

"KA3": 1,

"KA4": 0

},

},

{

"Programme": 3,

"Status": "1#start",

"component\_states": {

"KM1": 0,

"KM2": 1,

"KM3": 0,

"KA1": 0,

"KA2": 0,

"KA3": 1,

"KA4": 0

},

},

{

"Programme": 0,

"Status": "1#start",

"component\_states": {

"KM1": 0,

"KM2": 0,

"KM3": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 0

},

},

{

"Programme": 4,

"Status": "1#stop",

"component\_states": {

"KM1": 0,

"KM2": 1,

"KM3": 0,

"KA1": 0,

"KA2": 1,

"KA3": 0,

"KA4": 0

},

},

{

"Programme": 5,

"Status": "2#stop",

"component\_states": {

"KM1": 0,

"KM2": 1,

"KM3": 1,

"KA1": 0,

"KA2": 1,

"KA3": 0,

"KA4": 1

},

},

{

"Programme": 6,

"Status": "3#stop",

"component\_states": {

"KM1": 1,

"KM2": 0,

"KM3": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 1

},

{

"Programme": 0,

"Status": "1#start",

"component\_states": {

"KM1": 0,

"KM2": 0,

"KM3": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 0

},

}

}

、、、、、、、、、

②Speed relay and button contact status table:

{

"components": [

{"id": "KS1"},

{"id": "KS2"},

{"id": "KA1"},

{"id": "KA2"},

{"id": "KA3"},

{"id": "KA4"},

{"id": "SB1"},

{"id": "SB2"},

{"id": "SB3"}

],

"programs": [

{

"Programme": 0,

"Status": "in situ",

"component\_states": {

"KS1": 0,

"KS2": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 0,

"SB1": 1,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: NONE;

from 1 to 0: NONE

}

]

},

{

"Programme": 1,

"Status": "Forward series resistor voltage reduction startup",

"component\_states": {

"KS1": 0,

"KS2": 0,

"KA1": 1,

"KA2": 0,

"KA3": 0,

"KA4": 0,

"SB1": 1,

"SB2": 1,

"SB3": 0

},

"transitions": [

{from 0 to 1: SB2;

from 1 to 0: NONE

}

]

},

{

"Programme": 2,

"Status": "Positive full pressure operation",

"component\_states": {

"KS1": 1,

"KS2": 0,

"KA1": 1,

"KA2": 0,

"KA3": 1,

"KA4": 0,

"SB1": 1,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: KS1;

from 1 to 0: NONE}

]

},

{

"Programme": 3,

"Status": "Reverse connection of series resistor for braking",

"component\_states": {

"KS1": 1,

"KS2": 0,

"KA1": 0,

"KA2": 0,

"KA3": 1,

"KA4": 0,

"SB1": 0,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: SB1;

from 1 to 0: NONE}

]

},

{

"Programme": 0,

"Status": "stop",

"component\_states": {

"KS1": 0,

"KS2": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 0,

"SB1": 0,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: NONE;

from 1 to 0: KS1 }

]

},

{

"Programme": 4,

"Status": "Reverse series resistor voltage reduction braking",

"component\_states": {

"KS1": 0,

"KS2": 0,

"KA1": 0,

"KA2": 1,

"KA3": 0,

"KA4": 0,

"SB1": 1,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: SB3 ;

from 1 to 0: NONE }

]

},

{

"Programme": 5,

"Status":"Reverse full pressure operation",

"component\_states": {

"KS1": 0,

"KS2": 1,

"KA1": 0,

"KA2": 1,

"KA3": 0,

"KA4": 1,

"SB1": 1,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: KS2 ;

from 1 to 0: NONE }

]

},

{

"Programme": 6,

"Status": "3#stop",

"component\_states": {

"KS1": 0,

"KS2": 1,

"KA1": 0,

"KA2": 1,

"KA3": 0,

"KA4": 1,

"SB1": 1,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: NONE ;

from 1 to 0: SB1 }

]

},

{

"Programme": 0,

"Status": "Sop",

"component\_states": {

"KS1": 0,

"KS2": 0,

"KA1": 0,

"KA2": 0,

"KA3": 0,

"KA4": 0,

"SB1": 0,

"SB2": 0,

"SB3": 0

},

"transitions": [

{from 0 to 1: NONE ;

from 1 to 0: KS2 }

]

}

（4）Write down the logic function expressions for the output elements.

Logic function expression for the intermediate relay

KA1 = -SB1\* (SB2+KA1)

KA2 = -SB1\*(SB3+ KA2)

KA3 = KS1\*(KS1\* KA1 +KA3)= KS1\*(KA1 +KA3)

KA4 = KS2\*(KS2\*KA2+KA4)= KS2\*(KA2+KA4)

Logical function of contactor

KM1=(KA1+KA4)\* (-KM2)

KM2=(KA2 +KA3)\* (-KM1)

KM3= (KA1\*KA3)+(KA2\*KA4)

、、、、、、、、、、、、、

\*\*Example 3: Automated Two-Station Drilling Machine\*\*

This system controls a drilling machine with two stations (Station 1 and Station 2). Workpieces are manually loaded at each station. The machine performs the following cycle:

\*\*(1) According to the production process requirements, create a working cycle diagram:\*\*

\*\*States:\*\*

0: Initial "Reset" state (Machine Idle).

1: Station 1 Clamping.

2: Station 1 Drilling.

3: Station 1 Drilling Retract.

4: Station 1 Unclamping.

5: Station 2 Clamping.

6: Station 2 Drilling.

7: Station 2 Drilling Retract.

8: Station 2 Unclamping.

\*\*Transitions:\*\*

From 0 to 1: Pressing "Start Station 1" button (SB2) when workpiece is present at Station 1 (Proximity Sensor SQ1 is active).

From 1 to 2: Clamping at Station 1 is completed (Pressure Switch SP1 is active).

From 2 to 3: Drilling at Station 1 is completed (Timer KT1 times out).

From 3 to 4: Drill retract at Station 1 is completed (Limit Switch SL1 is active).

From 4 to 5: Pressing "Start Station 2" button (SB3) when workpiece is present at Station 2 (Proximity Sensor SQ2 is active).

From 5 to 6: Clamping at Station 2 is completed (Pressure Switch SP2 is active).

From 6 to 7: Drilling at Station 2 is completed (Timer KT2 times out).

From 7 to 8: Drill retract at Station 2 is completed (Limit Switch SL2 is active).

From 8 to 0: Cycle Complete (both stations finished, system returns to idle). This could be automatic or triggered by a reset button (SB1). Let's assume automatic return after station 2 completion for simplicity in this example.

\*\*Components:\*\*

\* \*\*Execution Elements:\*\*

\* KM1: Station 1 Clamping Contactor (solenoid valve for pneumatic clamp).

\* KM2: Station 1 Drill Motor Contactor.

\* KM3: Station 1 Drill Retract Contactor (solenoid valve for pneumatic cylinder retraction).

\* KM4: Station 2 Clamping Contactor.

\* KM5: Station 2 Drill Motor Contactor.

\* KM6: Station 2 Drill Retract Contactor.

\* \*\*Detection Elements:\*\*

\* SQ1: Proximity Sensor for Workpiece Presence at Station 1 (Normally Open).

\* SQ2: Proximity Sensor for Workpiece Presence at Station 2 (Normally Open).

\* SP1: Pressure Switch for Clamping Completion at Station 1 (Normally Open, activates when pressure is reached).

\* SP2: Pressure Switch for Clamping Completion at Station 2 (Normally Open).

\* SL1: Limit Switch for Drill Retract Completion at Station 1 (Normally Open, activates when drill is fully retracted).

\* SL2: Limit Switch for Drill Retract Completion at Station 2 (Normally Open).

\* KT1: Timer for Station 1 Drilling Duration (Time Delay Relay).

\* KT2: Timer for Station 2 Drilling Duration (Time Delay Relay).

\* SB1: Stop/Reset Button (Normally Closed).

\* SB2: Start Station 1 Button (Normally Open, momentary).

\* SB3: Start Station 2 Button (Normally Open, momentary).

\*\*(2) Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram:\*\*

\*\*① "Contactor and Time Relay Coil Status Table":\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "KM4"}, {"id": "KM5"}, {"id": "KM6"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KA1"}, {"id": "KA2"}

],

"programs": [

{ "Programme": 0, "Status": "Reset", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 0} },

{ "Programme": 1, "Status": "Station 1 Clamping", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 1, "KA2": 0} },

{ "Programme": 2, "Status": "Station 1 Drilling", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 1, "KT2": 0, "KA1": 1, "KA2": 0} },

{ "Programme": 3, "Status": "Station 1 Drill Retract", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 1, "KA2": 0} },

{ "Programme": 4, "Status": "Station 1 Unclamping", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 0} },

{ "Programme": 5, "Status": "Station 2 Clamping", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 1} },

{ "Programme": 6, "Status": "Station 2 Drilling", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KM6": 0, "KT1": 0, "KT2": 1, "KA1": 0, "KA2": 1} },

{ "Programme": 7, "Status": "Station 2 Drill Retract", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 1, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 1} },

{ "Programme": 8, "Status": "Station 2 Unclamping", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 0} }

]

}

```

\*\*② "Sensor and Button Contact Status Table":\*\*

```json

{

"components": [

{"id": "SQ1"}, {"id": "SQ2"}, {"id": "SP1"}, {"id": "SP2"}, {"id": "SL1"}, {"id": "SL2"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "SB3"}, {"id": "KA1"}, {"id": "KA2"}

],

"programs": [

{ "Programme": 0, "Status": "Reset", "component\_states": {"SQ1": 0, "SQ2": 0, "SP1": 0, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 0}, "transitions": [] },

{ "Programme": 1, "Status": "Station 1 Clamping", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 0, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 1, "SB3": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* SQ1"}] },

{ "Programme": 2, "Status": "Station 1 Drilling", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 1, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 1, "to": 2, "condition": "SP1"}] },

{ "Programme": 3, "Status": "Station 1 Drill Retract", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 1, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 1, "KA2": 0}, "transitions": [{"from": 2, "to": 3, "condition": "KT1"}] },

{ "Programme": 4, "Status": "Station 1 Unclamping", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 1, "SP2": 0, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 0}, "transitions": [{"from": 3, "to": 4, "condition": "SL1"}] },

{ "Programme": 5, "Status": "Station 2 Clamping", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 0, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 1, "KA1": 0, "KA2": 1}, "transitions": [{"from": 4, "to": 5, "condition": "SB3 \* SQ2"}] },

{ "Programme": 6, "Status": "Station 2 Drilling", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 1, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 1}, "transitions": [{"from": 5, "to": 6, "condition": "SP2"}] },

{ "Programme": 7, "Status": "Station 2 Drill Retract", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 1, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 1}, "transitions": [{"from": 6, "to": 7, "condition": "KT2"}] },

{ "Programme": 8, "Status": "Station 2 Unclamping", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 1, "SL1": 1, "SL2": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 0}, "transitions": [{"from": 7, "to": 8, "condition": "SL2"}, {"from": 8, "to": 0, "condition": "SL2"}] }

]

}

```

\*\*(3) Determine the grouping of the waiting area and set up intermediate memory elements:\*\*

\* \*\*Program Feature Numbers (based on Sensor/Button table - simplified to only sensors/buttons for feature number):\*\*

\* Program 0: `000000100`

\* Program 1: `100000110`

\* Program 2: `101000100`

\* Program 3: `101000100`

\* Program 4: `101010100`

\* Program 5: `010010101`

\* Program 6: `010110100`

\* Program 7: `010110100`

\* Program 8: `010111100`

\* \*\*Overlapping Feature Numbers:\*\*

\* Programs 2 and 3 have the same feature number `101000100`.

\* Programs 6 and 7 have the same feature number `010110100`.

\* \*\*Intermediate Memory Elements:\*\* We need intermediate relays to differentiate between Programs 2 & 3 and Programs 6 & 7. Let's use:

\* \*\*KA1:\*\* To latch the "Station 1 Cycle Active" state, activated when Station 1 cycle starts (Program 1) and deactivated after Station 1 cycle ends (Program 4). Already used and listed in tables.

\* \*\*KA2:\*\* To latch the "Station 2 Cycle Active" state, activated when Station 2 cycle starts (Program 5) and deactivated after Station 2 cycle ends (Program 8). Already used and listed in tables.

\* \*\*We might need additional relays if there are still ambiguities after considering KA1 and KA2 in feature numbers. Let's re-examine feature numbers \*including\* KA1 and KA2 states from the Contactor/Relay Coil Status Table:\*\*

\* \*\*Revised Program Feature Numbers (Sensor/Button + KA1/KA2):\*\*

\* Program 0: `000000100 00`

\* Program 1: `100000110 10`

\* Program 2: `101000100 10`

\* Program 3: `101000100 10`

\* Program 4: `101010100 00`

\* Program 5: `010010101 01`

\* Program 6: `010110100 01`

\* Program 7: `010110100 01`

\* Program 8: `010111100 00`

\* \*\*Still Overlapping Feature Numbers (with KA1/KA2):\*\*

\* Programs 2 and 3 still have the same feature number `101000100 10`.

\* Programs 6 and 7 still have the same feature number `010110100 01`.

\* \*\*Need more Intermediate Relays:\*\*

\* \*\*KA3:\*\* To differentiate between Program 2 (Drilling) and Program 3 (Retract) at Station 1. KA3 will be ON during Drilling (Program 2) and OFF during Retract (Program 3). We can use KT1 (Timer) to trigger KA3 ON and SL1 (Limit Switch) to trigger KA3 OFF.

\* \*\*KA4:\*\* To differentiate between Program 6 (Drilling) and Program 7 (Retract) at Station 2. KA4 will be ON during Drilling (Program 6) and OFF during Retract (Program 7). We can use KT2 (Timer) to trigger KA4 ON and SL2 (Limit Switch) to trigger KA4 OFF.

\* \*\*Updated Contactor and Time Relay Coil Status Table (with KA3, KA4):\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "KM4"}, {"id": "KM5"}, {"id": "KM6"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}, {"id": "KA4"}

],

"programs": [

{ "Programme": 0, "Status": "Reset", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0} },

{ "Programme": 1, "Status": "Station 1 Clamping", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0} },

{ "Programme": 2, "Status": "Station 1 Drilling", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 1, "KT2": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0} },

{ "Programme": 3, "Status": "Station 1 Drill Retract", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0} },

{ "Programme": 4, "Status": "Station 1 Unclamping", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0} },

{ "Programme": 5, "Status": "Station 2 Clamping", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 0} },

{ "Programme": 6, "Status": "Station 2 Drilling", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KM6": 0, "KT1": 0, "KT2": 1, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 1} },

{ "Programme": 7, "Status": "Station 2 Drill Retract", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 1, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 0} },

{ "Programme": 8, "Status": "Station 2 Unclamping", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0} }

]

}

```

\* \*\*Updated Sensor and Button Contact Status Table (including KA3, KA4):\*\*

```json

{

"components": [

{"id": "SQ1"}, {"id": "SQ2"}, {"id": "SP1"}, {"id": "SP2"}, {"id": "SL1"}, {"id": "SL2"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "SB3"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}, {"id": "KA4"}

],

"programs": [

{ "Programme": 0, "Status": "Reset", "component\_states": {"SQ1": 0, "SQ2": 0, "SP1": 0, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [] },

{ "Programme": 1, "Status": "Station 1 Clamping", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 0, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 1, "SB3": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* SQ1"}] },

{ "Programme": 2, "Status": "Station 1 Drilling", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 1, "SP2": 0, "SL1": 0, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 1, "KA2": 0, "KA3": 1, "KA4": 0}, "transitions": [{"from": 1, "to": 2, "condition": "SP1"}] },

{ "Programme": 3, "Status": "Station 1 Drill Retract", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 1, "SP2": 0, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 2, "to": 3, "condition": "KT1"}] },

{ "Programme": 4, "Status": "Station 1 Unclamping", "component\_states": {"SQ1": 1, "SQ2": 0, "SP1": 1, "SP2": 0, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 3, "to": 4, "condition": "SL1"}] },

{ "Programme": 5, "Status": "Station 2 Clamping", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 0, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 1, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 4, "to": 5, "condition": "SB3 \* SQ2"}] },

{ "Programme": 6, "Status": "Station 2 Drilling", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 1, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 1}, "transitions": [{"from": 5, "to": 6, "condition": "SP2"}] },

{ "Programme": 7, "Status": "Station 2 Drill Retract", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 1, "SL1": 1, "SL2": 0, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 6, "to": 7, "condition": "KT2"}] },

{ "Programme": 8, "Status": "Station 2 Unclamping", "component\_states": {"SQ1": 0, "SQ2": 1, "SP1": 0, "SP2": 1, "SL1": 1, "SL2": 1, "SB1": 1, "SB2": 0, "SB3": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 7, "to": 8, "condition": "SL2"}, {"from": 8, "to": 0, "condition": "SL2"}] }

]

}

```

\*\*(4) List the logical functions of the output components:\*\*

\* \*\*Intermediate Relays:\*\*

\* \*\*KA1 = (SB2 \* SQ1 + KA1) \* (-SB1) \* (-KA2)\*\* (Station 1 Cycle Latch): Activated by SB2 and SQ1, self-latched, reset by SB1 or if Station 2 cycle is active (interlock to prevent simultaneous station starts - simplified for this example, more complex interlocking is possible).

\* \*\*KA2 = (SB3 \* SQ2 + KA2) \* (-SB1) \* (-KA1)\*\* (Station 2 Cycle Latch): Activated by SB3 and SQ2, self-latched, reset by SB1 or if Station 1 cycle is active (interlock).

\* \*\*KA3 = (KT1 + KA3) \* (-SL1)\*\* (Station 1 Drilling State): Activated by KT1 (timer start when drilling starts - KM2), reset by SL1 (retract complete).

\* \*\*KA4 = (KT2 + KA4) \* (-SL2)\*\* (Station 2 Drilling State): Activated by KT2 (timer start when drilling starts - KM5), reset by SL2 (retract complete).

\* \*\*Output Contactors:\*\*

\* \*\*KM1 = KA1 \* (-KM2) \* (-KM3) \* (-KM4) \* (-KM5) \* (-KM6)\*\* (Station 1 Clamping): Active when Station 1 cycle is active (KA1) and no other operations are running (interlocks for safety/sequence - simplified).

\* \*\*KM2 = KA1 \* SP1 \* KA3 \* (-KM1) \* (-KM3) \* (-KM4) \* (-KM5) \* (-KM6)\*\* (Station 1 Drill Motor): Active when Station 1 cycle is active (KA1), clamping is complete (SP1), in drilling state (KA3), and no other conflicting operations running (interlocks).

\* \*\*KM3 = KA1 \* KT1 \* (-KM1) \* (-KM2) \* (-KM4) \* (-KM5) \* (-KM6)\*\* (Station 1 Drill Retract): Active when Station 1 cycle is active (KA1), drilling timer KT1 has timed out, and no other conflicting operations running.

\* \*\*KM4 = KA2 \* (-KM1) \* (-KM2) \* (-KM3) \* (-KM5) \* (-KM6)\*\* (Station 2 Clamping): Active when Station 2 cycle is active (KA2) and no other operations running (interlocks).

\* \*\*KM5 = KA2 \* SP2 \* KA4 \* (-KM1) \* (-KM2) \* (-KM3) \* (-KM4) \* (-KM6)\*\* (Station 2 Drill Motor): Active when Station 2 cycle is active (KA2), clamping is complete (SP2), in drilling state (KA4), and no other conflicting operations running.

\* \*\*KM6 = KA2 \* KT2 \* (-KM1) \* (-KM2) \* (-KM3) \* (-KM4) \* (-KM5)\*\* (Station 2 Drill Retract): Active when Station 2 cycle is active (KA2), drilling timer KT2 has timed out, and no other conflicting operations running.

\* \*\*KT1 = KM2\*\* (Station 1 Drill Timer): Timer KT1 starts when Station 1 Drill Motor (KM2) is activated.

\* \*\*KT2 = KM5\*\* (Station 2 Drill Timer): Timer KT2 starts when Station 2 Drill Motor (KM5) is activated.

This example is more complex than Example 2, involving two stations, sequential operations, sensors, timers, and more intricate interlocking logic (though still simplified for clarity). It demonstrates how the logical design method can be applied to a more elaborate automated system.

、、、、、、、、、、、、、、

\*\*Example 4: Automatic Bottle Filling System\*\*

This system controls an automated bottle filling machine. Empty bottles are placed on a conveyor belt. The system detects the presence of a bottle, stops the conveyor, fills the bottle with liquid, moves the filled bottle to the next station, and continues the cycle.

\*\*(1) According to the production process requirements, create a working cycle diagram:\*\*

\*\*States:\*\*

0: Initial "Idle" state (Conveyor Running, Waiting for Bottle).

1: Bottle Detected, Conveyor Stop.

2: Filling Process Active.

3: Filling Process Complete.

4: Conveyor Restart, Bottle Advance.

\*\*Transitions:\*\*

From 0 to 1: Bottle presence detected by sensor (Proximity Sensor SQ1 is active).

From 1 to 2: After a short delay to ensure conveyor stop (Timer KT1 times out).

From 2 to 3: Filling process is completed (Level Sensor SL1 in the bottle is active).

From 3 to 4: After a short delay to allow valve closing and drip stop (Timer KT2 times out).

From 4 to 0: Bottle moved away from filling station, bottle presence sensor becomes inactive (Proximity Sensor SQ1 becomes inactive).

\*\*Components:\*\*

\* \*\*Execution Elements:\*\*

\* KM1: Conveyor Belt Motor Contactor.

\* KM2: Filling Valve Solenoid Contactor (opens the valve to fill).

\* \*\*Detection Elements:\*\*

\* SQ1: Proximity Sensor for Bottle Presence at Filling Station (Normally Open, detects bottle).

\* SL1: Level Sensor in Bottle (Normally Open, activates when desired fill level is reached).

\* KT1: Timer for Conveyor Stop Delay (Time Delay Relay, delays start of filling after bottle detection).

\* KT2: Timer for Valve Closing Delay (Time Delay Relay, delays conveyor restart after filling completion).

\* SB1: Stop/Reset Button (Normally Closed).

\* SB2: Start/Cycle Start Button (Normally Open, momentary - starts initial conveyor run).

\*\*(2) Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram:\*\*

\*\*① "Contactor and Time Relay Coil Status Table":\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KA1"}

],

"programs": [

{ "Programme": 0, "Status": "Idle - Conveyor Run", "component\_states": {"KM1": 1, "KM2": 0, "KT1": 0, "KT2": 0, "KA1": 0} },

{ "Programme": 1, "Status": "Conveyor Stop - Bottle Detected", "component\_states": {"KM1": 0, "KM2": 0, "KT1": 1, "KT2": 0, "KA1": 1} },

{ "Programme": 2, "Status": "Filling Active", "component\_states": {"KM1": 0, "KM2": 1, "KT1": 0, "KT2": 0, "KA1": 1} },

{ "Programme": 3, "Status": "Filling Complete - Valve Close Delay", "component\_states": {"KM1": 0, "KM2": 0, "KT1": 0, "KT2": 1, "KA1": 1} },

{ "Programme": 4, "Status": "Conveyor Restart - Bottle Advance", "component\_states": {"KM1": 1, "KM2": 0, "KT1": 0, "KT2": 0, "KA1": 0} }

]

}

```

\*\*② "Sensor and Button Contact Status Table":\*\*

```json

{

"components": [

{"id": "SQ1"}, {"id": "SL1"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KA1"}

],

"programs": [

{ "Programme": 0, "Status": "Idle - Conveyor Run", "component\_states": {"SQ1": 0, "SL1": 0, "SB1": 1, "SB2": 0, "KT1": 0, "KT2": 0, "KA1": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SQ1"}] },

{ "Programme": 1, "Status": "Conveyor Stop - Bottle Detected", "component\_states": {"SQ1": 1, "SL1": 0, "SB1": 1, "SB2": 0, "KT1": 1, "KT2": 0, "KA1": 1}, "transitions": [{"from": 1, "to": 2, "condition": "KT1"}] },

{ "Programme": 2, "Status": "Filling Active", "component\_states": {"SQ1": 1, "SL1": 0, "SB1": 1, "SB2": 0, "KT1": 0, "KT2": 0, "KA1": 1}, "transitions": [{"from": 2, "to": 3, "condition": "SL1"}] },

{ "Programme": 3, "Status": "Filling Complete - Valve Close Delay", "component\_states": {"SQ1": 1, "SL1": 1, "SB1": 1, "SB2": 0, "KT1": 0, "KT2": 1, "KA1": 1}, "transitions": [{"from": 3, "to": 4, "condition": "KT2"}] },

{ "Programme": 4, "Status": "Conveyor Restart - Bottle Advance", "component\_states": {"SQ1": 1, "SL1": 1, "SB1": 1, "SB2": 0, "KT1": 0, "KT2": 0, "KA1": 0}, "transitions": [{"from": 4, "to": 0, "condition": "(-SQ1)"}] }

]

}

```

\*\*(3) Determine the grouping of the waiting area and set up intermediate memory elements:\*\*

\* \*\*Program Feature Numbers (based on Sensor/Button table):\*\*

\* Program 0: `0010000`

\* Program 1: `1010011`

\* Program 2: `1010011`

\* Program 3: `1110011`

\* Program 4: `1110000`

\* \*\*Overlapping Feature Numbers:\*\*

\* Programs 1 and 2 have the same feature number: `1010011`.

\* \*\*Intermediate Memory Elements:\*\* We need an intermediate relay to differentiate between Program 1 (Conveyor Stopped, waiting for fill start) and Program 2 (Filling Active). Let's use:

\* \*\*KA1:\*\* To latch "Bottle Detected and Filling Cycle Active" state. Activated when bottle is detected (SQ1) and remains active until the bottle filling and advance cycle is complete. Already listed in tables.

\* \*\*Need to distinguish between Program 1 and Program 2:\*\* Program 1 is entered when SQ1 is active. Program 2 starts after a delay (KT1) \*after\* Program 1 is active. So, we can use the state of KT1 to differentiate. Program 1: KT1 is timing. Program 2: KT1 has timed out (KT1 contact changes state). Therefore, we \*might\* not need an additional intermediate relay \*if\* we use the state of KT1's contact directly in our logic. Let's try to proceed without adding another intermediate relay for now and see if it works.

\*\*(4) List the logical functions of the output components:\*\*

\* \*\*Intermediate Relay:\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-SB1) \* (-KA1\_ResetCondition)\*\* (Cycle Latch): Activated by SQ1 (bottle detected), self-latched. Reset by SB1 (Stop/Reset) or `KA1\_ResetCondition`. We need to define `KA1\_ResetCondition` to reset KA1 at the end of the cycle (Program 4 to 0 transition - when SQ1 becomes inactive). So, `KA1\_ResetCondition = (-SQ1)`. Therefore:

\* \*\*KA1 = (SQ1 + KA1) \* (-SB1) \* (-(-SQ1)) = (SQ1 + KA1) \* (-SB1) \* (SQ1) = SQ1 \* (-SB1) + KA1 \* SQ1\*\* This is logically incorrect. Let's rethink the KA1 latching.

\* \*\*Corrected KA1 Logic:\*\* KA1 should be set when SQ1 becomes active and stay active until the cycle is complete. Let's use a different approach for KA1. We want KA1 to be ON from Program 1 to Program 3. It's simpler to control KA1 directly based on program transitions:

\* \*\*KA1 = (SQ1 \* (-KA1\_OffCondition) + KA1) \* (-SB1)\*\* (Cycle Latch): Set by SQ1 \* and \*not\* `KA1\_OffCondition`. Self-latched, reset by SB1.

\* `KA1\_OffCondition` should be true in Program 4 when transitioning back to Program 0. The transition is `(-SQ1)` from Program 4 to 0. So, `KA1\_OffCondition = (-SQ1)`.

\* \*\*KA1 = (SQ1 \* (-( -SQ1 )) + KA1) \* (-SB1) = (SQ1 \* SQ1 + KA1) \* (-SB1) = (SQ1 + KA1) \* (-SB1)\*\* This is still not quite right for reset.

\* \*\*Even Better KA1 Logic (Pulse Set/Reset approach conceptually):\*\* Let's think of KA1 as a simple latch set by SQ1 and reset at the end.

\* \*\*KA1 = (SQ1 \* (-KA1)) + (KA1 \* (-KA1\_ResetSignal))\*\* This is still complex to represent directly in basic relay logic.

\* \*\*Simplified KA1 - State Transition based:\*\* KA1 is ON in Programs 1, 2, 3. OFF in Programs 0, 4.

\* \*\*KA1 = (Program 1 state) + (Program 2 state) + (Program 3 state)\*\* This is conceptual. How to implement Program state logic?

\* \*\*Let's use a Set-Dominant SR Latch for KA1:\*\* We'll use Set input and Reset input.

\* \*\*Set KA1 = SQ1 \* (-KA1)\*\* (Set when SQ1 becomes active, only if KA1 is not already set)

\* \*\*Reset KA1 = (-SQ1) \* KA1\*\* (Reset when SQ1 becomes inactive, only if KA1 is already set)

\* This is still not directly in equation form for coil logic. Let's use self-holding again with a reset condition.

\* \*\*Final KA1 Logic (Self-Hold with Reset):\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-KA1\_Reset)\*\*

\* \*\*KA1\_Reset = (-SQ1) + SB1\*\* (Reset if SQ1 is inactive OR SB1 is pressed)

\* \*\*KA1 = (SQ1 + KA1) \* (-((-SQ1) + SB1)) = (SQ1 + KA1) \* ( -(-SQ1) \* (-SB1) ) = (SQ1 + KA1) \* (SQ1 \* (-SB1)) = SQ1 \* (-SB1) + KA1 \* SQ1 \* (-SB1) = SQ1 \* (-SB1) \* (1 + KA1) = SQ1 \* (-SB1)\*\* This is wrong again.

\* \*\*Let's try a slightly different self-hold with \*interruption\* for reset:\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-((-SQ1) \* SB1)) = (SQ1 + KA1) \* -(-SQ1) + -SB1) = (SQ1 + KA1) \* (SQ1 + (-SB1))\*\*

\* \*\*KA1 = SQ1 + SQ1\*(-SB1) + KA1\*SQ1 + KA1\*(-SB1) = SQ1 + KA1\*(-SB1) + KA1\*SQ1\*\* - Still too complex and not clearly a simple relay circuit equation.

\* \*\*Let's simplify KA1 to be ON during Programs 1, 2, 3 and use program transitions to control it more directly.\*\* We'll use KA1 to enable Program 1, 2, 3 operations.

\* \*\*Output Contactors & Timers:\*\*

\* \*\*KM1 = (-KA1) + (-(SQ1)) + SB2\*\* (Conveyor Motor): Conveyor ON in Idle (Program 0 - KA1=0) OR when no bottle is detected anymore (-(SQ1)) to advance OR initially on system start (SB2). So, initially run on SB2 press, then run when KA1 is OFF (idle), and also run after filling when bottle moves away (-(SQ1)). Let's simplify for now to: \*\*KM1 = (-KA1) + SB2\*\* (Conveyor on in idle or on initial start). And we'll make sure KA1 is OFF in Program 0 and 4, and ON in 1, 2, 3.

\* \*\*KM2 = KA1 \* KT1 \* (-SL1)\*\* (Filling Valve): Filling valve ON when cycle is active (KA1), after conveyor stop delay (KT1 timed out), and \*before\* fill level is reached ((-SL1)).

\* \*\*KT1 = SQ1 \* KA1\*\* (Conveyor Stop Delay Timer): Timer KT1 starts when bottle is detected (SQ1) and cycle is active (KA1).

\* \*\*KT2 = SL1 \* KA1\*\* (Valve Close Delay Timer): Timer KT2 starts when fill level is reached (SL1) and cycle is active (KA1).

\* \*\*Refined KA1 Logic (State-Based, not self-holding directly):\*\*

\* \*\*KA1 = (SQ1) \* (-KA1\_ResetCondition)\*\* (Set Condition - simplified - just when SQ1 is active).

\* \*\*KA1\_ResetCondition = SL1 \* KT2\*\* (Reset condition - after filling level and valve close delay timer are done).

\* \*\*KA1 = (SQ1) \* (-(SL1 \* KT2)) = SQ1 \* (-SL1 + -KT2) = SQ1 \* (-SL1) + SQ1 \* (-KT2)\*\* - Still not ideal. Let's go back to self-holding, but simpler reset.

\* \*\*Even Simpler KA1 - Self-Hold with direct reset from cycle end:\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-CycleEndReset)\*\*

\* \*\*CycleEndReset = SL1 \* KT2 \* (-SQ1)\*\* (Reset when fill level reached, valve delay done, and bottle moved away - SQ1 inactive again). This is getting too complex for simple relay logic equations.

\* \*\*Let's simplify KA1 to be latched by SQ1 and unlatched by the transition from Program 4 to 0 (-(SQ1)).\*\*

\* \*\*Final Simplified KA1 - Set by SQ1, Reset by -SQ1 transition:\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-KA1\_Reset)\*\*

\* \*\*KA1\_Reset = (-SQ1)\*\* (Reset when SQ1 becomes inactive).

\* \*\*KA1 = (SQ1 + KA1) \* (-(-SQ1)) = (SQ1 + KA1) \* SQ1 = SQ1 + KA1\*SQ1 = SQ1 \* (1 + KA1) = SQ1\*\* - This is still wrong, KA1 becomes just equal to SQ1, not latching.

\* \*\*Let's use a different approach for KA1 - Pulse Set by SQ1, Pulse Reset by -SQ1 transition - conceptually using Set-Reset behavior:\*\*

\* \*\*KA1 Coil ON condition = SQ1 \* (-KA1\_Reset)\*\* (Set when SQ1 is active and not being reset)

\* \*\*KA1 Coil OFF condition = KA1\_Reset = (-SQ1) + SB1\*\* (Reset when SQ1 is inactive OR SB1 is pressed)

\* \*\*Let's try a simple self-holding circuit for KA1 with reset by (-SQ1) OR SB1:\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-((-SQ1) + SB1)) = (SQ1 + KA1) \* (SQ1 \* (-SB1)) = SQ1 \* (-SB1) + KA1 \* SQ1 \* (-SB1)\*\* - Still not quite right.

\* \*\*Let's just use a simple self-holding circuit for KA1, set by SQ1 and reset by SB1. And then use program transitions to control the rest.\*\*

\* \*\*KA1 = (SQ1 + KA1) \* (-SB1)\*\* (Simple Cycle Latch - set by bottle presence, reset by Stop).

\* \*\*Revised Output Logic with Simplified KA1:\*\*

\* \*\*KA1 = (SB2 \* SQ1 + KA1) \* (-SB1)\*\* (Cycle Latch - Start by SB2 & SQ1, self-hold, reset by SB1. Adding SB2 to initiate cycle).

\* \*\*KM1 = (-KA1) + SB2\*\* (Conveyor Motor - Run in idle or on initial start).

\* \*\*KM2 = KA1 \* KT1 \* (-SL1)\*\* (Filling Valve).

\* \*\*KT1 = KM1 \* KA1 \* SQ1\*\* (Conveyor Stop Delay Timer - Start when conveyor is running, cycle active, and bottle present). - \*Correction\*: KT1 should start timing \*after\* conveyor stops. So, KT1 should be triggered when KM1 becomes OFF \*and\* KA1 is ON and SQ1 is ON. \*\*KT1 = (-KM1) \* KA1 \* SQ1\*\*

\* \*\*KT2 = KM2 \* SL1 \* KA1\*\* (Valve Close Delay Timer - Start when valve is open, level reached and cycle active). - \*Correction\*: KT2 should start timing after filling is \*complete\* (SL1 becomes active). So, KT2 should be triggered by SL1 \*and\* when KM2 is ON and KA1 is ON. \*\*KT2 = SL1 \* KA1\*\*

This example is more focused on sequential control based on sensor inputs and timers, with a simplified latching mechanism. The logic for KA1 and timers is more complex to express directly in simple relay logic equations and requires careful consideration of timing and transitions. We've simplified KA1 to a basic latch for this example to make the equations more manageable. More robust latching and reset logic would be needed in a real-world implementation.

、、、、、、、、、、、、、

\*\*Example 5: Two-Axis Pick-and-Place Robot System\*\*.

This system controls a simple two-axis pick-and-place robot. The robot has an X-axis for horizontal movement and a Z-axis for vertical movement with a pneumatic gripper. The robot picks up parts from a fixed location (Pick Station) and places them at another fixed location (Place Station).

\*\*(1) According to the production process requirements, create a working cycle diagram:\*\*

\*\*States:\*\*

0: Initial "Home" state (Robot at Home Position, Gripper Open).

1: Move X-Axis to Pick Station.

2: Move Z-Axis Down (to Pick Position).

3: Close Gripper (Pick Part).

4: Move Z-Axis Up (Retract after Pick).

5: Move X-Axis to Place Station.

6: Move Z-Axis Down (to Place Position).

7: Open Gripper (Place Part).

8: Move Z-Axis Up (Retract after Place).

9: Move X-Axis to Home Position.

\*\*Transitions:\*\*

From 0 to 1: Pressing "Start Cycle" button (SB2) when robot is at Home position (Limit Switch SL\_HomeX and SL\_HomeZ are active) and no fault is present (Fault Signal FS is inactive - normally closed contact).

From 1 to 2: X-Axis reaches Pick Station (Limit Switch SL\_PickX is active).

From 2 to 3: Z-Axis reaches Pick Down Position (Limit Switch SL\_PickZ\_Down is active).

From 3 to 4: Gripper Close is confirmed (Pressure Switch SP\_GripperClose is active - indicating successful grip).

From 4 to 5: Z-Axis reaches Pick Up Position (Limit Switch SL\_PickZ\_Up is active - retracted after pick).

From 5 to 6: X-Axis reaches Place Station (Limit Switch SL\_PlaceX is active).

From 6 to 7: Z-Axis reaches Place Down Position (Limit Switch SL\_PlaceZ\_Down is active).

From 7 to 8: Gripper Open is confirmed (Pressure Switch SP\_GripperOpen is active - indicating part release).

From 8 to 9: Z-Axis reaches Place Up Position (Limit Switch SL\_PlaceZ\_Up is active - retracted after place).

From 9 to 0: X-Axis reaches Home Position (Limit Switch SL\_HomeX is active).

\*\*Components:\*\*

\* \*\*Execution Elements:\*\*

\* KM1: X-Axis Forward Motor Contactor (Move to Pick/Place Station).

\* KM2: X-Axis Reverse Motor Contactor (Move to Home Position).

\* KM3: Z-Axis Down Solenoid Contactor (Move Down for Pick/Place).

\* KM4: Z-Axis Up Solenoid Contactor (Move Up for Retract).

\* KM5: Gripper Close Solenoid Contactor.

\* KM6: Gripper Open Solenoid Contactor.

\* \*\*Detection Elements:\*\*

\* SL\_HomeX: Limit Switch - X-Axis Home Position (Normally Open, active at Home).

\* SL\_PickX: Limit Switch - X-Axis Pick Station Position (Normally Open, active at Pick Station).

\* SL\_PlaceX: Limit Switch - X-Axis Place Station Position (Normally Open, active at Place Station).

\* SL\_HomeZ: Limit Switch - Z-Axis Home/Up Position (Normally Open, active at Home/Up).

\* SL\_PickZ\_Down: Limit Switch - Z-Axis Pick Down Position (Normally Open, active at Pick Down).

\* SL\_PlaceZ\_Down: Limit Switch - Z-Axis Place Down Position (Normally Open, active at Place Down).

\* SL\_PickZ\_Up: Limit Switch - Z-Axis Pick/Place Up Position (Normally Open, active at Up position for both Pick and Place).

\* SP\_GripperClose: Pressure Switch - Gripper Close Confirmation (Normally Open, active when gripper is closed and pressure is reached).

\* SP\_GripperOpen: Pressure Switch - Gripper Open Confirmation (Normally Open, active when gripper is open and pressure is reached).

\* FS: Fault Signal Contact (Normally Closed, opens on fault condition - external fault detection assumed).

\* SB1: Stop/Reset Button (Normally Closed).

\* SB2: Start Cycle Button (Normally Open, momentary).

\*\*(2) Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram:\*\*

\*\*① "Contactor and Time Relay Coil Status Table":\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "KM4"}, {"id": "KM5"}, {"id": "KM6"}, {"id": "KA1"}

],

"programs": [

{ "Programme": 0, "Status": "Home", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 0} },

{ "Programme": 1, "Status": "Move X to Pick", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1} },

{ "Programme": 2, "Status": "Move Z Down (Pick)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1} },

{ "Programme": 3, "Status": "Close Gripper", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KM6": 0, "KA1": 1} },

{ "Programme": 4, "Status": "Move Z Up (Pick Retract)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KA1": 1} },

{ "Programme": 5, "Status": "Move X to Place", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1} },

{ "Programme": 6, "Status": "Move Z Down (Place)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1} },

{ "Programme": 7, "Status": "Open Gripper", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 1, "KA1": 1} },

{ "Programme": 8, "Status": "Move Z Up (Place Retract)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KA1": 1} },

{ "Programme": 9, "Status": "Move X to Home", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1} }

]

}

```

\*\*② "Sensor and Button Contact Status Table":\*\*

```json

{

"components": [

{"id": "SL\_HomeX"}, {"id": "SL\_PickX"}, {"id": "SL\_PlaceX"}, {"id": "SL\_HomeZ"}, {"id": "SL\_PickZ\_Down"}, {"id": "SL\_PlaceZ\_Down"}, {"id": "SL\_PickZ\_Up"}, {"id": "SP\_GripperClose"}, {"id": "SP\_GripperOpen"}, {"id": "FS"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "KA1"}

],

"programs": [

{ "Programme": 0, "Status": "Home", "component\_states": {"SL\_HomeX": 1, "SL\_PickX": 0, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 0}, "transitions": [] },

{ "Programme": 1, "Status": "Move X to Pick", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 1, "KA1": 1}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* SL\_HomeX \* SL\_HomeZ \* FS"}] },

{ "Programme": 2, "Status": "Move Z Down (Pick)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 1, "to": 2, "condition": "SL\_PickX"}] },

{ "Programme": 3, "Status": "Close Gripper", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 2, "to": 3, "condition": "SL\_PickZ\_Down"}] },

{ "Programme": 4, "Status": "Move Z Up (Pick Retract)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 3, "to": 4, "condition": "SP\_GripperClose"}] },

{ "Programme": 5, "Status": "Move X to Place", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 4, "to": 5, "condition": "SL\_PickZ\_Up"}] },

{ "Programme": 6, "Status": "Move Z Down (Place)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 5, "to": 6, "condition": "SL\_PlaceX"}] },

{ "Programme": 7, "Status": "Open Gripper", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 1, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 6, "to": 7, "condition": "SL\_PlaceZ\_Down"}] },

{ "Programme": 8, "Status": "Move Z Up (Place Retract)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 1, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 7, "to": 8, "condition": "SP\_GripperOpen"}] },

{ "Programme": 9, "Status": "Move X to Home", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 1, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1}, "transitions": [{"from": 8, "to": 9, "condition": "SL\_PickZ\_Up"}] },

{ "Programme": 0, "Status": "Home", "component\_states": {"SL\_HomeX": 1, "SL\_PickX": 0, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 0}, "transitions": [{"from": 9, "to": 0, "condition": "SL\_HomeX"}] }

]

}

```

\*\*(3) Determine the grouping of the waiting area and set up intermediate memory elements:\*\*

\* \*\*Program Feature Numbers (based on Sensor/Button table):\*\* (Simplified to only sensors/buttons/FS for feature number)

\* Program 0: `100100101100`

\* Program 1: `000100101111`

\* Program 2: `100100101100`

\* Program 3: `100010101100`

\* Program 4: `100010111100`

\* Program 5: `100010111100`

\* Program 6: `010010111100`

\* Program 7: `010001111100`

\* Program 8: `010001111100`

\* Program 9: `010001111100`

\* \*\*Overlapping Feature Numbers:\*\*

\* Programs 0 and 2 have the same feature number: `100100101100`.

\* Programs 4 and 5 have the same feature number: `100010111100`.

\* Programs 7, 8, and 9 have the same feature number: `010001111100`.

\* \*\*Intermediate Memory Elements:\*\* We need intermediate relays to differentiate between these overlapping programs. Let's use:

\* \*\*KA1:\*\* To latch the "Robot Cycle Active" state, activated when the cycle starts (Program 1) and deactivated when the cycle completes (Program 0 - Home state reached again). Already used and listed in tables.

\* \*\*Need to differentiate between Program 0 and Program 2:\*\* Program 0 is "Home" state, Program 2 is "Move Z Down (Pick)". They have the same sensor states `100100101100`. KA1 is ON in Program 2 and OFF in Program 0. So KA1 helps distinguish them. Let's re-examine feature numbers including KA1 state:

\* \*\*Revised Program Feature Numbers (Sensor/Button/FS + KA1):\*\*

\* Program 0: `100100101100 0`

\* Program 1: `000100101111 1`

\* Program 2: `100100101100 1`

\* Program 3: `100010101100 1`

\* Program 4: `100010111100 1`

\* Program 5: `100010111100 1`

\* Program 6: `010010111100 1`

\* Program 7: `010001111100 1`

\* Program 8: `010001111100 1`

\* Program 9: `010001111100 1`

\* \*\*Still Overlapping Feature Numbers (with KA1):\*\*

\* Programs 4 and 5 still have the same feature number: `100010111100 1`.

\* Programs 7, 8, and 9 still have the same feature number: `010001111100 1`.

\* \*\*Need More Intermediate Relays:\*\*

\* \*\*KA2:\*\* To differentiate between Program 4 (Move Z Up after Pick) and Program 5 (Move X to Place). KA2 will be ON during "Move X to Place" (Program 5) and OFF during "Move Z Up (Pick Retract)" (Program 4). We can use KM1 (X-axis forward motion) to trigger KA2 ON and KM4 (Z-axis up motion) to trigger KA2 OFF. Or more precisely, use the \*transition\* from Program 4 to 5 (SL\_PickZ\_Up) to set KA2 and transition from Program 9 to 0 (SL\_HomeX) to reset KA2.

\* \*\*KA3:\*\* To differentiate between Programs 7, 8, and 9. These are all after "Place Down". We need to distinguish "Open Gripper" (Program 7), "Move Z Up after Place" (Program 8), and "Move X to Home" (Program 9). KA3 will be ON during "Move Z Up after Place" (Program 8) and OFF during "Open Gripper" (Program 7) and "Move X to Home" (Program 9). We can use KM4 (Z-axis up motion) to trigger KA3 ON and KM2 (X-axis reverse motion - move to home) to trigger KA3 OFF. Or use transitions more accurately. Transition from 7 to 8 (SP\_GripperOpen) sets KA3. Transition from 9 to 0 (SL\_HomeX) resets KA3.

\* \*\*Updated Contactor and Time Relay Coil Status Table (with KA2, KA3):\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "KM4"}, {"id": "KM5"}, {"id": "KM6"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}

],

"programs": [

{ "Programme": 0, "Status": "Home", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 0, "KA2": 0, "KA3": 0} },

{ "Programme": 1, "Status": "Move X to Pick", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 0, "KA3": 0} },

{ "Programme": 2, "Status": "Move Z Down (Pick)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 0, "KA3": 0} },

{ "Programme": 3, "Status": "Close Gripper", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KM6": 0, "KA1": 1, "KA2": 0, "KA3": 0} },

{ "Programme": 4, "Status": "Move Z Up (Pick Retract)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 0, "KA3": 0} },

{ "Programme": 5, "Status": "Move X to Place", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 1, "KA3": 0} },

{ "Programme": 6, "Status": "Move Z Down (Place)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 1, "KA3": 0} },

{ "Programme": 7, "Status": "Open Gripper", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 1, "KA1": 1, "KA2": 1, "KA3": 0} },

{ "Programme": 8, "Status": "Move Z Up (Place Retract)", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 1, "KA3": 1} },

{ "Programme": 9, "Status": "Move X to Home", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KA1": 1, "KA2": 1, "KA3": 0} }

]

}

```

\* \*\*Updated Sensor and Button Contact Status Table (including KA2, KA3):\*\*

```json

{

"components": [

{"id": "SL\_HomeX"}, {"id": "SL\_PickX"}, {"id": "SL\_PlaceX"}, {"id": "SL\_HomeZ"}, {"id": "SL\_PickZ\_Down"}, {"id": "SL\_PlaceZ\_Down"}, {"id": "SL\_PickZ\_Up"}, {"id": "SP\_GripperClose"}, {"id": "SP\_GripperOpen"}, {"id": "FS"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}

],

"programs": [

{ "Programme": 0, "Status": "Home", "component\_states": {"SL\_HomeX": 1, "SL\_PickX": 0, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 0, "KA3": 0}, "transitions": [] },

{ "Programme": 1, "Status": "Move X to Pick", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 1, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* SL\_HomeX \* SL\_HomeZ \* FS"}] },

{ "Programme": 2, "Status": "Move Z Down (Pick)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 1, "to": 2, "condition": "SL\_PickX"}] },

{ "Programme": 3, "Status": "Close Gripper", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 2, "to": 3, "condition": "SL\_PickZ\_Down"}] },

{ "Programme": 4, "Status": "Move Z Up (Pick Retract)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 3, "to": 4, "condition": "SP\_GripperClose"}] },

{ "Programme": 5, "Status": "Move X to Place", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 1, "SL\_PlaceX": 0, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 4, "to": 5, "condition": "SL\_PickZ\_Up"}] },

{ "Programme": 6, "Status": "Move Z Down (Place)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 1, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 5, "to": 6, "condition": "SL\_PlaceX"}] },

{ "Programme": 7, "Status": "Open Gripper", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 1, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 6, "to": 7, "condition": "SL\_PlaceZ\_Down"}] },

{ "Programme": 8, "Status": "Move Z Up (Place Retract)", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 1, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 1}, "transitions": [{"from": 7, "to": 8, "condition": "SP\_GripperOpen"}] },

{ "Programme": 9, "Status": "Move X to Home", "component\_states": {"SL\_HomeX": 0, "SL\_PickX": 0, "SL\_PlaceX": 1, "SL\_HomeZ": 0, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 1, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 1, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 8, "to": 9, "condition": "SL\_PickZ\_Up"}] },

{ "Programme": 0, "Status": "Home", "component\_states": {"SL\_HomeX": 1, "SL\_PickX": 0, "SL\_PlaceX": 0, "SL\_HomeZ": 1, "SL\_PickZ\_Down": 0, "SL\_PlaceZ\_Down": 0, "SL\_PickZ\_Up": 1, "SP\_GripperClose": 0, "SP\_GripperOpen": 1, "FS": 1, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 0, "KA3": 0}, "transitions": [{"from": 9, "to": 0, "condition": "SL\_HomeX"}] }

]

}

```

\*\*(4) List the logical functions of the output components:\*\*

\* \*\*Intermediate Relays:\*\*

\* \*\*KA1 = (SB2 \* SL\_HomeX \* SL\_HomeZ \* FS + KA1) \* (-SB1)\*\* (Cycle Latch): Activated by Start button, Home sensors, and no fault, self-latched, reset by Stop.

\* \*\*KA2 = (SL\_PickZ\_Up + KA2) \* (-KA3) \* (-SL\_HomeX)\*\* (X-Place State Latch): Set by Z-Up after Pick, reset by reaching Home in X, interlocked with KA3 to prevent overlap.

\* \*\*KA3 = (SP\_GripperOpen + KA3) \* (-KA2) \* (-SL\_HomeX)\*\* (Z-Up After Place State Latch): Set by Gripper Open at Place, reset by reaching Home in X, interlocked with KA2.

\* \*\*Output Contactors:\*\*

\* \*\*KM1 = KA1 \* (-SL\_PickX) \* (-SL\_PlaceX) \* (KA2 + (-KA2)\*(-KA3))\*\* (X-Axis Forward): Active when cycle is running (KA1), X-axis is not at Pick or Place yet, and either in "Move X to Pick" (KA2=0, -KA3=1 in initial state of KA2, KA3 logic) or "Move X to Place" (KA2=1).

\* \*\*KM2 = KA1 \* (-SL\_HomeX) \* KA3\*\* (X-Axis Reverse): Active when cycle is running (KA1), X-axis not at Home, and in "Move X to Home" phase (KA3=0 initially, becomes 1 in Program 8, and back to 0 in Program 9, so KA3 is used to control X-Home move indirectly - \*Correction\*: KA3 is meant to be ON in Program 8 - "Move Z Up After Place" and OFF in Program 9 "Move X to Home", so KM2 should be active in Program 9 when KA3 is OFF and moving to Home - therefore, use `-KA3` instead of `KA3` in KM2 equation). \*\*KM2 = KA1 \* (-SL\_HomeX) \* (-KA3)\*\*

\* \*\*KM3 = KA1 \* (-SL\_PickZ\_Down) \* (-SL\_PlaceZ\_Down) \* ((-KA2)\*(-KA3) + KA2)\*\* (Z-Axis Down): Active when cycle is running (KA1), Z-axis not at Pick or Place Down positions, and either in "Move Z Down (Pick)" (KA2=0, -KA3=1 initially) or "Move Z Down (Place)" (KA2=1).

\* \*\*KM4 = KA1 \* (-SL\_PickZ\_Up) \* ((-KA2)\*(-KA3) + KA2\*KA3)\*\* (Z-Axis Up): Active when cycle is running (KA1), Z-axis not at Up position, and either in "Move Z Up (Pick Retract)" (KA2=0, -KA3=1 initially) or "Move Z Up (Place Retract)" (KA2=1, KA3=1). -\*Correction\*: KA3 is ON in Program 8 "Move Z Up after Place", and OFF in Program 4 "Move Z Up after Pick". We need to differentiate between these two Z-Up movements. Let's refine KA3 to be specifically for "Z-Up After Place" only. And use KA2 for "X-to-Place". Let's simplify KA3 to be ON only during Program 8. Then KM4 logic becomes simpler. \*\*KM4 = KA1 \* (-SL\_PickZ\_Up) \* ((-KA2) + KA2\*KA3)\*\* - Still complex. Let's rethink KA2 and KA3 role.

\* \*\*Simplified KA2 & KA3 roles:\*\*

\* \*\*KA2 = Program 5, 6, 7, 8, 9 active (Place Cycle Phase)\*\*

\* \*\*KA3 = Program 8, 9 active (Home Return Phase)\*\*

\* \*\*Revised KA2 & KA3 Logic:\*\*

\* \*\*KA2 = (SL\_PickZ\_Up + KA2) \* (-SL\_HomeX)\*\* (Place Phase Latch - set after Pick-Up, reset at Home).

\* \*\*KA3 = (SP\_GripperOpen + KA3) \* (-SL\_HomeX)\*\* (Home Return Phase Latch - set after Gripper Open at Place, reset at Home).

\* \*\*Revised Output Contactors (using simplified KA2, KA3):\*\*

\* \*\*KM1 = KA1 \* (-SL\_PickX) \* (-SL\_PlaceX) \* KA2\*\* (X-Axis Forward - Move to Place Station only - \*Correction\*: Need to move to Pick Station as well. KM1 is used for both "Move X to Pick" and "Move X to Place". Let's separate them. KM1\_Pick for "Move X to Pick", KM1\_Place for "Move X to Place". But using single KM1 in tables. Let's reuse KM1 for both forward X moves, controlled by state. ) \*\*KM1 = KA1 \* (-SL\_PickX) \* (-SL\_PlaceX)\*\* - This will be active for \*both\* "Move X to Pick" and "Move X to Place" phases. We need to control \*when\* it moves to Pick vs Place. We'll use program state transitions for this control implicitly through sequencing.

\* \*\*KM2 = KA1 \* (-SL\_HomeX) \* KA3\*\* (X-Axis Reverse - Move to Home).

\* \*\*KM3 = KA1 \* (-SL\_PickZ\_Down) \* (-SL\_PlaceZ\_Down) \* KA2\*\* (Z-Axis Down - Move Down for Place - \*Correction\*: Need to move down for Pick as well. KM3 is used for both "Move Z Down (Pick)" and "Move Z Down (Place)". Let's reuse KM3 for both down Z moves, controlled by state implicitly) \*\*KM3 = KA1 \* (-SL\_PickZ\_Down) \* (-SL\_PlaceZ\_Down)\*\*.

\* \*\*KM4 = KA1 \* (-SL\_PickZ\_Up) \* KA2 \* KA3\*\* (Z-Axis Up - Move Up after Place - \*Correction\*: Need to move up after Pick as well. KM4 is used for both "Move Z Up (Pick Retract)" and "Move Z Up (Place Retract)". Reuse KM4) \*\*KM4 = KA1 \* (-SL\_PickZ\_Up)\*\*.

\* \*\*KM5 = KA1 \* SL\_PickZ\_Down \* (-SP\_GripperClose)\*\* (Gripper Close - at Pick Down Position, before confirmation).

\* \*\*KM6 = KA1 \* SL\_PlaceZ\_Down \* (-SP\_GripperOpen)\*\* (Gripper Open - at Place Down Position, before confirmation).

This example is getting quite complex with multiple axes and sequential movements. The logical functions are becoming more involved, and careful consideration of state transitions and intermediate relays is crucial for correct operation. We've simplified some aspects and might need further refinement for a truly robust and unambiguous relay circuit design.

、、、、、、、、、、、、、、、

\*\*Example 6: Automatic Palletizing System\*\*.

This system automates the process of stacking boxes onto a pallet. Boxes arrive on an infeed conveyor, are positioned, lifted (simulated by Z-axis movement), and placed onto a pallet. Once a layer is complete, the pallet is lowered (simulated by Y-axis). When the pallet is full, the system stops.

\*\*(1) According to the production process requirements, create a working cycle diagram:\*\*

\*\*States:\*\*

0: Initial "Idle" state (System Ready, Waiting for Box).

1: Box Arrives at Infeed Sensor.

2: Infeed Conveyor Stop, Positioning Conveyor Starts.

3: Box Positioned at Lifting Station.

4: Lifting Platform Rises.

5: Box Lifted to Pallet Layer Height.

6: Positioning Conveyor Starts (to place box on pallet).

7: Box Placed on Pallet.

8: Positioning Conveyor Stops.

9: Lifting Platform Lowers (slightly for next box in layer).

10: Check Layer Completion. If Layer Not Complete, go to State 1 (for next box in layer).

11: Layer Complete. Check Pallet Full. If Pallet Not Full, Lower Pallet Platform (Y-axis Down).

12: Pallet Lowered for Next Layer. Go to State 1 (for next layer).

13: Pallet Full. System Stop (Cycle End).

\*\*Transitions:\*\*

From 0 to 1: Box detected at infeed conveyor sensor (Proximity Sensor SQ1 is active).

From 1 to 2: After a short delay for sensor confirmation (Timer KT1 times out).

From 2 to 3: Box positioned at lifting station (Photoelectric Sensor SQ2 at lifting station is active).

From 3 to 4: After a short delay for positioning (Timer KT2 times out).

From 4 to 5: Lifting platform reaches upper position (Limit Switch SL\_Up is active).

From 5 to 6: After lift confirmation (Timer KT3 times out).

From 6 to 7: Box placed on pallet (assume this is a timed conveyor run, Timer KT4 times out).

From 7 to 8: After placement duration (Timer KT5 times out).

From 8 to 9: Lifting platform reaches lower position (Limit Switch SL\_Down is active - slightly lowered for next box in layer).

From 9 to 10: Layer completion check (Counter CT1 reaches preset count for boxes in a layer).

From 10 to 1: If layer not complete (CT1 not reached), repeat for next box in layer.

From 10 to 11: If layer complete (CT1 reached).

From 11 to 12: Pallet lowering complete (Limit Switch SL\_PalletDown is active - pallet lowered by one layer height).

From 12 to 1: Repeat for next layer.

From 11 to 13: Pallet full check (Counter CT2 reaches preset count for layers on pallet).

From 13 to 0: System Reset (manual reset button SB1).

\*\*Components:\*\*

\* \*\*Execution Elements:\*\*

\* KM1: Infeed Conveyor Motor Contactor.

\* KM2: Positioning Conveyor Motor Contactor.

\* KM3: Lifting Platform Up Solenoid Contactor.

\* KM4: Lifting Platform Down Solenoid Contactor.

\* KM5: Pallet Lowering Solenoid Contactor (Y-axis down).

\* KM6: Pallet Raising Solenoid Contactor (Y-axis up - for reset, not in cycle).

\* \*\*Detection Elements:\*\*

\* SQ1: Proximity Sensor - Box Arrival at Infeed Conveyor (Normally Open).

\* SQ2: Photoelectric Sensor - Box Positioned at Lifting Station (Normally Open).

\* SL\_Up: Limit Switch - Lifting Platform Upper Position (Normally Open).

\* SL\_Down: Limit Switch - Lifting Platform Lower Position (Normally Open).

\* SL\_PalletDown: Limit Switch - Pallet Lowered One Layer Height (Normally Open).

\* CT1: Counter - Box Count in Layer (Preset to number of boxes per layer).

\* CT2: Counter - Layer Count on Pallet (Preset to number of layers per pallet).

\* KT1: Timer - Infeed Sensor Confirmation Delay (Time Delay Relay).

\* KT2: Timer - Positioning Delay at Lift Station (Time Delay Relay).

\* KT3: Timer - Lift Confirmation Delay (Time Delay Relay).

\* KT4: Timer - Box Placement Duration (Positioning Conveyor Run Time for Placement) (Time Delay Relay).

\* KT5: Timer - Placement Completion Delay (Time Delay Relay).

\* SB1: Stop/Reset Button (Normally Closed).

\* SB2: Start Cycle Button (Normally Open, momentary).

\*\*(2) Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram:\*\*

\*\*① "Contactor and Time Relay Coil Status Table":\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "KM4"}, {"id": "KM5"}, {"id": "KM6"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KT5"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}

],

"programs": [

{"Programme": 0, "Status": "Idle", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 0, "KA2": 0, "KA3": 0}},

{"Programme": 1, "Status": "Box Arrived", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 1, "KA2": 0, "KA3": 0}},

{"Programme": 2, "Status": "Positioning Start", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 1, "KA2": 0, "KA3": 0}},

{"Programme": 3, "Status": "Box Positioned", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 1, "KT4": 0, "KT5": 0, "KA1": 1, "KA2": 0, "KA3": 0}},

{"Programme": 4, "Status": "Lifting Platform Up", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 1, "KA2": 1, "KA3": 0}},

{"Programme": 5, "Status": "Placement Conveyor Start", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 1, "KT5": 0, "KA1": 1, "KA2": 1, "KA3": 0}},

{"Programme": 6, "Status": "Box Placed", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 1, "KA1": 1, "KA2": 1, "KA3": 0}},

{"Programme": 7, "Status": "Platform Lowering", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 1, "KA2": 1, "KA3": 1}},

{"Programme": 8, "Status": "Layer Complete Check", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 0, "KA2": 1, "KA3": 1}},

{"Programme": 9, "Status": "Pallet Lowering", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 0, "KA2": 1, "KA3": 0}},

{"Programme": 10, "Status": "Pallet Lowered", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 0, "KA2": 0, "KA3": 0}},

{"Programme": 11, "Status": "System Stop", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KM6": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KA1": 0, "KA2": 0, "KA3": 0}}

]

}

```

\*\*② "Sensor and Button Contact Status Table":\*\*

```json

{

"components": [

{"id": "SQ1"}, {"id": "SQ2"}, {"id": "SL\_Up"}, {"id": "SL\_Down"}, {"id": "SL\_PalletDown"}, {"id": "CT1"}, {"id": "CT2"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KT5"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}

],

"programs": [

{"Programme": 0, "Status": "Idle", "component\_states": {"SQ1": 0, "SQ2": 0, "SL\_Up": 0, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 0, "KA3": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SQ1 \* (-KA1) \* (-KA3) \* (-KA2) \* SB2"}]},

{"Programme": 1, "Status": "Box Arrived", "component\_states": {"SQ1": 1, "SQ2": 0, "SL\_Up": 0, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 1, "to": 2, "condition": "KT1"}]},

{"Programme": 2, "Status": "Positioning Start", "component\_states": {"SQ1": 1, "SQ2": 0, "SL\_Up": 0, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KT5": 0, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 2, "to": 3, "condition": "SQ2"}]},

{"Programme": 3, "Status": "Box Positioned", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 0, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT3": 1, "KT4": 0, "KT5": 0, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 0, "KA3": 0}, "transitions": [{"from": 3, "to": 4, "condition": "KT2"}]},

{"Programme": 4, "Status": "Lifting Platform Up", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 0, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT4": 0, "KT5": 0, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 4, "to": 5, "condition": "SL\_Up"}]},

{"Programme": 5, "Status": "Placement Conveyor Start", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 1, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT4": 1, "KT5": 0, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 5, "to": 6, "condition": "KT3"}]},

{"Programme": 6, "Status": "Box Placed", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 1, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "KT5": 1, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 0}, "transitions": [{"from": 6, "to": 7, "condition": "KT4"}]},

{"Programme": 7, "Status": "Platform Lowering", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 1, "SL\_Down": 0, "SL\_PalletDown": 1, "CT1": 0, "CT2": 0, "SB1": 1, "SB2": 0, "KA1": 1, "KA2": 1, "KA3": 1}, "transitions": [{"from": 7, "to": 8, "condition": "KT5"}]},

{"Programme": 8, "Status": "Layer Complete Check", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 1, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 1, "CT2": 0, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 1, "KA3": 1}, "transitions": [{"from": 8, "to": 9, "condition": "CT1"}, {"from": 8, "to": 1, "condition": "(-CT1)"}]},

{"Programme": 9, "Status": "Pallet Lowering", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 1, "SL\_Down": 1, "SL\_PalletDown": 0, "CT1": 1, "CT2": 0, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 1, "KA3": 0}, "transitions": [{"from": 9, "to": 10, "condition": "SL\_PalletDown"}]},

{"Programme": 10, "Status": "Pallet Lowered", "component\_states": {"SQ1": 1, "SQ2": 1, "SL\_Up": 1, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 1, "CT2": 0, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 0, "KA3": 0}, "transitions": [{"from": 10, "to": 1, "condition": "(-CT2)"}, {"from": 10, "to": 11, "condition": "CT2"}]},

{"Programme": 11, "Status": "System Stop", "component\_states": {"SQ1": 0, "SQ2": 0, "SL\_Up": 0, "SL\_Down": 1, "SL\_PalletDown": 1, "CT1": 0, "CT2": 1, "SB1": 1, "SB2": 0, "KA1": 0, "KA2": 0, "KA3": 0}, "transitions": [{"from": 11, "to": 0, "condition": "SB1"}]}

]

}

```

\*\*(3) Determine the grouping of the waiting area and set up intermediate memory elements:\*\*

\* \*\*Program Feature Numbers (based on Sensor/Button/Counters table - simplified):\*\*

\* Program 0: `000110000000100`

\* Program 1: `100110000000101`

\* Program 2: `100110000000101`

\* Program 3: `110110000000101`

\* Program 4: `110110000000101`

\* Program 5: `111110000000101`

\* Program 6: `111110000000101`

\* Program 7: `111010000000101`

\* Program 8: `111111000000100` (CT1=1)

\* Program 9: `111101000000100` (SL\_PalletDown=0, CT1=1)

\* Program 10: `111111000000100` (SL\_PalletDown=1, CT1=1)

\* Program 11: `000111000000100` (CT2=1)

\* \*\*Overlapping Feature Numbers:\*\*

\* Programs 1, 2, 3, 4, 5, 6, 7 have similar feature numbers.

\* Programs 8 and 10 have the same feature number `111111000000100`.

\* \*\*Intermediate Memory Elements:\*\* We need intermediate relays to differentiate between overlapping programs.

\* \*\*KA1:\*\* To latch the "Palletizing Cycle Active" state. Activated at cycle start, deactivated at cycle end or stop. Already in tables.

\* \*\*KA2:\*\* To distinguish between "Lifting Platform Up" (Program 4), "Placement Conveyor Start" (Program 5), "Box Placed" (Program 6), "Platform Lowering" (Program 7). KA2 will be ON from Program 4 to Program 7.

\* \*\*KA3:\*\* To differentiate between "Layer Complete Check" (Program 8), "Pallet Lowering" (Program 9), and "Pallet Lowered" (Program 10). KA3 will be ON from Program 7 to Program 9.

\* \*\*Revised Contactor and Time Relay Coil Status Table (with KA1, KA2, KA3):\*\* - Already updated in the table above.

\* \*\*Revised Sensor and Button Contact Status Table (with KA1, KA2, KA3):\*\* - Already updated in the table above.

\*\*(4) List the logical functions of the output components:\*\*

\* \*\*Intermediate Relays:\*\*

\* \*\*KA1 = (SB2 \* SQ1 \* (-KA1) \* (-KA2) \* (-KA3) + KA1) \* (-SB1)\*\* (Cycle Latch): Start by SB2, SQ1, and not already running cycle phases (interlocks), self-latched, reset by SB1.

\* \*\*KA2 = (SL\_Up + KA2) \* (-KA3) \* (-CT1) \* (-SB1)\*\* (Lifting/Placement Phase Latch): Set by SL\_Up, reset when KA3 is set (Layer Complete) or CT1 reaches count (Layer Complete also) or SB1 (Stop).

\* \*\*KA3 = (KT5 + KA3) \* (-KA2) \* (-CT2) \* (-SB1)\*\* (Layer/Pallet Lowering Phase Latch): Set by KT5 (Placement Complete Delay), reset when KA2 is set (Lifting/Placement starts again) or CT2 reaches count (Pallet Full) or SB1 (Stop).

\* \*\*Output Contactors:\*\*

\* \*\*KM1 = (-KA1) + (KA1 \* (-SQ1) \* (-KA2) \* (-KA3))\*\* (Infeed Conveyor): ON in Idle (KA1=0) or during Infeed phase (KA1=1, and before box reaches SQ1 again, and not in later phases). Simplified to: \*\*KM1 = (-KA1) + (-SQ1)\*\* (Run when idle or no box at infeed sensor).

\* \*\*KM2 = KA1 \* (SQ1) \* (-SQ2) \* (-KA2) \* (-KA3)\*\* (Positioning Conveyor): ON when cycle active, box at infeed (SQ1), but not yet at positioning sensor (SQ2), and not in Lifting/Placement or Lowering phases. Simplified to: \*\*KM2 = KA1 \* SQ1 \* (-SQ2) \* (-KA2)\*\*

\* \*\*KM3 = KA1 \* (SQ2) \* (-SL\_Up) \* (-KA2) \* (-KA3)\*\* (Lifting Platform Up): ON when cycle active, box positioned (SQ2), platform not yet up (SL\_Up), and in Lifting phase (KA2=0, -KA3=1 initially). Simplified to: \*\*KM3 = KA1 \* SQ2 \* (-SL\_Up) \* (-KA2)\*\*

\* \*\*KM4 = KA1 \* (SL\_Up) \* (-SL\_Down) \* (KA2) \* (-KA3)\*\* (Lifting Platform Down): ON when cycle active, platform is up (SL\_Up), not yet down (SL\_Down), and in Lowering phase (KA2=1, KA3=0). Simplified to: \*\*KM4 = KA1 \* SL\_Up \* (-SL\_Down) \* (KA2)\*\*

\* \*\*KM5 = KA1 \* (CT1) \* (-SL\_PalletDown) \* (KA3)\*\* (Pallet Lowering): ON when cycle active, layer complete (CT1), pallet not yet lowered (SL\_PalletDown), and in Pallet Lowering phase (KA3=1). Simplified to: \*\*KM5 = KA1 \* CT1 \* (-SL\_PalletDown) \* (KA3)\*\*

\* \*\*KM6 = 0\*\* (Pallet Raising - not used in cycle, only for manual reset if needed).

\* \*\*Timers & Counters:\*\*

\* \*\*KT1 = KA1 \* SQ1\*\* (Infeed Sensor Delay): Start timing when cycle active and box at infeed.

\* \*\*KT2 = KA1 \* SQ2\*\* (Positioning Delay): Start timing when cycle active and box positioned.

\* \*\*KT3 = KA1 \* KM3\*\* (Lift Confirmation Delay): Start timing when Lift platform starts moving up (KM3).

\* \*\*KT4 = KA1 \* KM2 \* SL\_Up\*\* (Placement Duration): Start timing when Positioning Conveyor runs for placement and platform is up.

\* \*\*KT5 = KA1 \* KM2 \* KT4\*\* (Placement Completion Delay): Start timing when Placement duration timer (KT4) starts. - \*Correction\*: KT5 should start timing \*after\* placement conveyor stops. So, KT5 should be triggered by KT4 completion. \*\*KT5 = KT4\*\* (Use KT4 contact to trigger KT5 timer).

\* \*\*CT1 = Increment on every Box Placed (Program 7):\*\* CT1 Count Increment Logic: Need to increment CT1 \*after\* Program 7 "Platform Lowering" starts. Increment CT1 when transitioning from Program 7 to 8 (triggered by KT5). \*\*CT1 Increment = KT5 (contact from KT5 timer)\*\*.

\* \*\*CT2 = Increment on Layer Complete (Program 8/10):\*\* CT2 Count Increment Logic: Increment CT2 \*after\* Program 10 "Pallet Lowered" is reached. Increment CT2 when transitioning from Program 10 to 1 (triggered by SL\_PalletDown). \*\*CT2 Increment = SL\_PalletDown (contact from SL\_PalletDown limit switch)\*\*.

\* \*\*Counters Reset:\*\* CT1 and CT2 need to be reset at system start or manual reset (SB1). \*\*CT1 Reset = SB1\*\*, \*\*CT2 Reset = SB1\*\*.

This Palletizing System example is significantly more complex, involving multiple conveyors, a lifting platform, pallet lowering, timers, and counters. The logical functions are becoming quite intricate, and the use of intermediate relays is essential to manage the sequential states and differentiate between overlapping sensor conditions. The counter and timer logic adds another layer of complexity to the relay circuit design.

、、、、、、、、、、、

\*\*Example 7: Automated Multi-Stage Chemical Processing Unit\*\*.

This system controls a simplified chemical processing unit with three stages: Mixing, Heating, and Cooling. Raw materials are loaded, mixed, heated to a specific temperature, cooled down, and then discharged. The system includes safety interlocks and error detection.

\*\*(1) According to the production process requirements, create a working cycle diagram:\*\*

\*\*States:\*\*

0: Initial "Idle" state (Unit Ready, Waiting for Start).

1: Material Loading Phase.

2: Material Loading Complete.

3: Mixing Phase Start.

4: Mixing Phase Running.

5: Mixing Phase Complete.

6: Heating Phase Start.

7: Heating Phase Running.

8: Target Temperature Reached.

9: Heating Phase Complete (Soaking Time).

10: Soaking Time Complete.

11: Cooling Phase Start.

12: Cooling Phase Running.

13: Target Cool Temperature Reached.

14: Cooling Phase Complete.

15: Discharge Phase Start.

16: Discharge Phase Running.

17: Discharge Phase Complete.

18: Cycle Complete, Return to Idle.

19: Fault State (System Error, requires manual reset).

\*\*Transitions:\*\*

From 0 to 1: Pressing "Start Cycle" button (SB2) when system is idle and no fault is active (Fault Relay FR is not active - normally closed contact).

From 1 to 2: Material loading is confirmed (Weight Sensor WS1 reaches setpoint for material loaded).

From 2 to 3: Loading Complete confirmation (Timer KT1 times out - to ensure stable weight reading).

From 3 to 4: Press "Start Mixing" button (SB3) or automatic start after loading complete. Let's assume automatic start.

From 4 to 5: Mixing duration is completed (Timer KT2 times out - Mixing Timer).

From 5 to 6: Mixing Phase Complete confirmation (Timer KT3 times out - short delay after mixing stop).

From 6 to 7: Heating Phase Start (Heater ON).

From 7 to 8: Temperature Sensor TS1 reaches target heating temperature (Temperature Switch TS1 is active).

From 8 to 9: Target Temperature Reached confirmation (Timer KT4 times out - to ensure stable temperature).

From 9 to 10: Heating Soaking Time is completed (Timer KT5 times out - Soaking Timer).

From 10 to 11: Heating Phase Complete confirmation (Timer KT6 times out - short delay after heating off).

From 11 to 12: Cooling Phase Start (Cooling System ON).

From 12 to 13: Temperature Sensor TS2 reaches target cooling temperature (Temperature Switch TS2 is active).

From 13 to 14: Target Cool Temperature Reached confirmation (Timer KT7 times out - to ensure stable cool temperature).

From 14 to 15: Cooling Phase Complete confirmation (Timer KT8 times out - short delay after cooling off).

From 15 to 16: Discharge Phase Start (Discharge Valve Open).

From 16 to 17: Discharge duration is completed (Timer KT9 times out - Discharge Timer).

From 17 to 18: Discharge Phase Complete confirmation (Timer KT10 times out - short delay after discharge valve close).

From 18 to 0: Cycle Complete, return to Idle.

To 19 (Fault State):

- Over-temperature during heating (Temperature Sensor TS1 exceeds critical limit - Temperature Switch TS3 is active).

- Over-pressure during mixing (Pressure Sensor PS1 exceeds critical limit - Pressure Switch PS1 is active).

- Motor Overload during Mixing (Motor Overload Relay OL1 trips - normally closed contact opens).

From 19 to 0: Manual Reset by pressing and releasing "Reset Fault" button (SB4, momentary Normally Open).

\*\*Components:\*\*

\* \*\*Execution Elements:\*\*

\* KM1: Material Loading Valve Contactor (Open valve to load material).

\* KM2: Mixer Motor Contactor.

\* KM3: Heater Contactor.

\* KM4: Cooling System Contactor.

\* KM5: Discharge Valve Contactor (Open valve to discharge).

\* \*\*Detection Elements:\*\*

\* WS1: Weight Sensor - Material Loaded Weight Setpoint (Analog sensor with setpoint comparator - assumed digital output when weight reached, normally open - WS1 becomes active when weight reached).

\* TS1: Temperature Sensor - Heating Target Temperature Setpoint (Temperature Switch - normally open, TS1 becomes active when target heating temperature reached).

\* TS2: Temperature Sensor - Cooling Target Temperature Setpoint (Temperature Switch - normally open, TS2 becomes active when target cooling temperature reached).

\* TS3: Temperature Sensor - Over-Temperature Limit (Temperature Switch - normally open, TS3 becomes active when over-temperature).

\* PS1: Pressure Sensor - Over-Pressure Limit (Pressure Switch - normally open, PS1 becomes active when over-pressure).

\* OL1: Motor Overload Relay - Mixer Motor Overload (Normally Closed contact - OL1 contact opens on overload).

\* FR: Fault Relay (Normally Closed contact - FR contact opens when any fault condition is active).

\* KT1: Timer - Loading Complete Confirmation Delay.

\* KT2: Timer - Mixing Duration.

\* KT3: Timer - Mixing Phase Complete Delay.

\* KT4: Timer - Heating Target Temperature Confirmation Delay.

\* KT5: Timer - Heating Soaking Time.

\* KT6: Timer - Heating Phase Complete Delay.

\* KT7: Timer - Cooling Target Temperature Confirmation Delay.

\* KT8: Timer - Cooling Phase Complete Delay.

\* KT9: Timer - Discharge Duration.

\* KT10: Timer - Discharge Phase Complete Delay.

\* SB1: Stop/Emergency Stop Button (Normally Closed).

\* SB2: Start Cycle Button (Normally Open, momentary).

\* SB3: Start Mixing Button (Normally Open, momentary - optional, assuming automatic mixing start after loading).

\* SB4: Reset Fault Button (Normally Open, momentary).

\*\*(2) Determine the execution element and detection element, and make the action rhythm table of the execution element and the status table of the detection element according to the working cycle diagram:\*\*

\*\*① "Contactor and Time Relay Coil Status Table":\*\*

```json

{

"components": [

{"id": "KM1"}, {"id": "KM2"}, {"id": "KM3"}, {"id": "KM4"}, {"id": "KM5"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KT5"}, {"id": "KT6"}, {"id": "KT7"}, {"id": "KT8"}, {"id": "KT9"}, {"id": "KT10"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}, {"id": "KA4"}

],

"programs": [

{"Programme": 0, "Status": "Idle", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 1, "Status": "Loading", "component\_states": {"KM1": 1, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 2, "Status": "Loading Complete", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 3, "Status": "Mixing Start", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 1, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 4, "Status": "Mixing Running", "component\_states": {"KM1": 0, "KM2": 1, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 1, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 5, "Status": "Mixing Complete", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 1, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}},

{"Programme": 6, "Status": "Heating Start", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 1, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 1, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}},

{"Programme": 7, "Status": "Heating Running", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 1, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}},

{"Programme": 8, "Status": "Target Temp Reached", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 1, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}},

{"Programme": 9, "Status": "Heating Soaking", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 1, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}},

{"Programme": 10, "Status": "Soaking Complete", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}},

{"Programme": 11, "Status": "Cooling Start", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}},

{"Programme": 12, "Status": "Cooling Running", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 1, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}},

{"Programme": 13, "Status": "Target Cool Temp Reached", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 1, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}},

{"Programme": 14, "Status": "Cooling Complete", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 1, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}},

{"Programme": 15, "Status": "Discharge Start", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 1}},

{"Programme": 16, "Status": "Discharge Running", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 1}},

{"Programme": 17, "Status": "Discharge Complete", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 0, "KA2": 1, "KA3": 1, "KA4": 1}},

{"Programme": 18, "Status": "Cycle Complete", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}},

{"Programme": 19, "Status": "Fault", "component\_states": {"KM1": 0, "KM2": 0, "KM3": 0, "KM4": 0, "KM5": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}}

]

}

```

\*\*② "Sensor and Button Contact Status Table":\*\*

```json

{

"components": [

{"id": "WS1"}, {"id": "TS1"}, {"id": "TS2"}, {"id": "TS3"}, {"id": "PS1"}, {"id": "OL1"}, {"id": "FR"}, {"id": "KT1"}, {"id": "KT2"}, {"id": "KT3"}, {"id": "KT4"}, {"id": "KT5"}, {"id": "KT6"}, {"id": "KT7"}, {"id": "KT8"}, {"id": "KT9"}, {"id": "KT10"}, {"id": "SB1"}, {"id": "SB2"}, {"id": "SB3"}, {"id": "SB4"}, {"id": "KA1"}, {"id": "KA2"}, {"id": "KA3"}, {"id": "KA4"}

],

"programs": [

{"Programme": 0, "Status": "Idle", "component\_states": {"WS1": 0, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 0, "to": 1, "condition": "SB2 \* (-FR)"}]},

{"Programme": 1, "Status": "Loading", "component\_states": {"WS1": 0, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 1, "to": 2, "condition": "WS1"}]},

{"Programme": 2, "Status": "Loading Complete", "component\_states": {"WS1": 1, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT1": 1, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 2, "to": 3, "condition": "KT1"}]},

{"Programme": 3, "Status": "Mixing Start", "component\_states": {"WS1": 1, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT2": 1, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 3, "to": 4, "condition": "KT2"}]},

{"Programme": 4, "Status": "Mixing Running", "component\_states": {"WS1": 1, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT3": 1, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 4, "to": 5, "condition": "KT3"}]},

{"Programme": 5, "Status": "Mixing Complete", "component\_states": {"WS1": 1, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT4": 1, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 5, "to": 6, "condition": "KT4"}]},

{"Programme": 6, "Status": "Heating Start", "component\_states": {"WS1": 1, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT5": 1, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 6, "to": 7, "condition": "KT5"}]},

{"Programme": 7, "Status": "Heating Running", "component\_states": {"WS1": 1, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT6": 1, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 7, "to": 8, "condition": "TS1"}]},

{"Programme": 8, "Status": "Target Temp Reached", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT7": 1, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 8, "to": 9, "condition": "KT6"}]},

{"Programme": 9, "Status": "Heating Soaking", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT8": 1, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 0, "KA4": 0}, "transitions": [{"from": 9, "to": 10, "condition": "KT7"}]},

{"Programme": 10, "Status": "Soaking Complete", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT9": 1, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}, "transitions": [{"from": 10, "to": 11, "condition": "KT8"}]},

{"Programme": 11, "Status": "Cooling Start", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "KT10": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}, "transitions": [{"from": 11, "to": 12, "condition": "KT9"}]},

{"Programme": 12, "Status": "Cooling Running", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}, "transitions": [{"from": 12, "to": 13, "condition": "TS2"}]},

{"Programme": 13, "Status": "Target Cool Temp Reached", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 1, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}, "transitions": [{"from": 13, "to": 14, "condition": "KT10"}]},

{"Programme": 14, "Status": "Cooling Complete", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 1, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 0}, "transitions": [{"from": 14, "to": 15, "condition": "KT8"}]},

{"Programme": 15, "Status": "Discharge Start", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 1, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 1}, "transitions": [{"from": 15, "to": 16, "condition": "KT9"}]},

{"Programme": 16, "Status": "Discharge Running", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 1, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 1, "KA2": 1, "KA3": 1, "KA4": 1}, "transitions": [{"from": 16, "to": 17, "condition": "KT10"}]},

{"Programme": 17, "Status": "Discharge Complete", "component\_states": {"WS1": 1, "TS1": 1, "TS2": 1, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 0, "KA2": 1, "KA3": 1, "KA4": 1}, "transitions": [{"from": 17, "to": 18, "condition": "KT6"}]},

{"Programme": 18, "Status": "Cycle Complete", "component\_states": {"WS1": 0, "TS1": 0, "TS2": 0, "TS3": 0, "PS1": 0, "OL1": 1, "FR": 1, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 18, "to": 0, "condition": "KT8"}]},

{"Programme": 19, "Status": "Fault", "component\_states": {"WS1": 0, "TS1": 0, "TS2": 0, "TS3": 1, "PS1": 1, "OL1": 0, "FR": 0, "KT1": 0, "KT2": 0, "KT3": 0, "KT4": 0, "KT5": 0, "KT6": 0, "KT7": 0, "KT8": 0, "KT9": 0, "KT10": 0, "SB1": 1, "SB2": 0, "SB3": 0, "SB4": 0, "KA1": 0, "KA2": 0, "KA3": 0, "KA4": 0}, "transitions": [{"from": 1, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 2, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 3, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 4, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 5, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 6, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 7, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 8, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 9, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 10, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 11, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 12, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 13, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 14, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 15, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 16, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 17, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 18, "to": 19, "condition": "TS3 + PS1 + (-OL1)"}, {"from": 19, "to": 0, "condition": "SB4"}]}

]

}

```

\*\*(3) Determine the grouping of the waiting area and set up intermediate memory elements:\*\*

\* \*\*Program Feature Numbers (based on Sensor/Button/Faults table - simplified):\*\*

\* Program 0: `0000011000000000000`

\* Program 1: `0000011000000000010`

\* Program 2: `1000011000000000010`

\* ... and so on. Due to the large number of programs and sensors, detailed feature number analysis is more complex and might be done programmatically in a real design tool. For manual analysis, we look for states with similar sensor/input combinations that need to be distinguished.

\* \*\*Intermediate Memory Elements:\*\* We will need intermediate relays to manage the sequential phases and latch states, especially given the long sequence and potential overlaps in sensor conditions across different phases.

\* \*\*KA1:\*\* Cycle Active Latch (as in previous examples).

\* \*\*KA2:\*\* Heating Phase Active Latch (from Heating Start to Cooling Start).

\* \*\*KA3:\*\* Cooling Phase Active Latch (from Cooling Start to Discharge Start).

\* \*\*KA4:\*\* Discharge Phase Active Latch (from Discharge Start to Cycle Complete).

\*\*(4) List the logical functions of the output components:\*\*

\* \*\*Intermediate Relays:\*\*

\* \*\*KA1 = (SB2 \* (-FR) + KA1) \* (-SB1) \* (-SB4)\*\* (Cycle Latch): Start by SB2 and no fault, self-latched, reset by SB1 or SB4 (Fault Reset - to ensure reset from fault also clears cycle).

\* \*\*KA2 = (KT3 + KA2) \* (-KA3) \* (-KA4) \* (-SB1) \* (-SB4)\*\* (Heating Phase Latch): Set after Mixing Complete Delay (KT3), reset when Cooling (KA3) or Discharge (KA4) phases start, or by SB1/SB4.

\* \*\*KA3 = (KT6 + KA3) \* (-KA2) \* (-KA4) \* (-SB1) \* (-SB4)\*\* (Cooling Phase Latch): Set after Heating Complete Delay (KT6), reset when Heating (KA2) or Discharge (KA4) phases start, or by SB1/SB4.

\* \*\*KA4 = (KT8 + KA4) \* (-KA2) \* (-KA3) \* (-SB1) \* (-SB4)\*\* (Discharge Phase Latch): Set after Cooling Complete Delay (KT8), reset when Heating (KA2) or Cooling (KA3) phases start (though these should not happen during discharge), or by SB1/SB4.

\* \*\*Fault Relay (FR):\*\* FR should be activated (coil energized) when a fault occurs. For normally closed FR contact logic, we need to de-energize FR coil on fault. Let's assume \*\*KF\*\* is the Fault Relay \*coil\*. Then FR contacts are controlled by KF. We want FR to be \*de-energized\* on fault, so we need to control KF coil to be \*off\* on fault and \*on\* normally.

\* \*\*KF = -(TS3 + PS1 + (-OL1) + (-SB4))\*\* (Fault Relay Coil Logic - De-energize on fault or manual reset). This is inverted logic for FR. Let's rethink. We want FR to be \*active\* (FR=1 meaning contact closed) during normal operation and \*inactive\* (FR=0, contact opens) on fault.

\* \*\*Corrected Fault Relay Logic (FR - contact state, KF - coil state):\*\* We want FR contact to open on fault. So, FR should be normally closed contact. We need to energize Fault Relay coil (KF) during normal operation to keep FR contact closed. De-energize KF to open FR contact (fault).

\* \*\*KF = (-(TS3 + PS1 + (-OL1))) \* (-SB4) \* (-SB1)\*\* (Fault Relay Coil - Energized normally, de-energized by over-temp, over-pressure, overload, or manual reset/stop).

\* \*\*FR is a normally closed contact controlled by KF.\*\*

\* \*\*Output Contactors:\*\*

\* \*\*KM1 = KA1 \* (-WS1)\*\* (Loading Valve): ON during Loading phase (KA1), until weight setpoint reached (WS1).

\* \*\*KM2 = KA1 \* KA2\*\* (Mixer Motor): ON during Mixing phase (KA1 and KA2).

\* \*\*KM3 = KA1 \* KA2 \* KT3\*\* (Heater): ON during Heating phase (KA1 and KA2), starting after Mixing Complete Delay (KT3).

\* \*\*KM4 = KA1 \* KA3\*\* (Cooling System): ON during Cooling phase (KA1 and KA3).

\* \*\*KM5 = KA1 \* KA4\*\* (Discharge Valve): ON during Discharge phase (KA1 and KA4).

\* \*\*Timers:\*\*

\* \*\*KT1 = KA1 \* WS1\*\* (Loading Complete Confirmation Delay): Start timing when Loading phase active and weight reached.

\* \*\*KT2 = KA1 \* KT1\*\* (Mixing Duration Timer): Start timing after Loading Complete Confirmation Delay (KT1).

\* \*\*KT3 = KA1 \* KT2\*\* (Mixing Phase Complete Delay): Start timing after Mixing Duration Timer (KT2).

\* \*\*KT4 = KA1 \* KM3\*\* (Heating Target Temperature Confirmation Delay): Start timing when Heater is ON (KM3).

\* \*\*KT5 = KA1 \* TS1\*\* (Heating Soaking Timer): Start timing when Target Heating Temperature is reached (TS1).

\* \*\*KT6 = KA1 \* KT5\*\* (Heating Phase Complete Delay): Start timing after Heating Soaking Timer (KT5).

\* \*\*KT7 = KA1 \* KM4\*\* (Cooling Target Temperature Confirmation Delay): Start timing when Cooling System is ON (KM4).

\* \*\*KT8 = KA1 \* TS2\*\* (Cooling Phase Complete Delay): Start timing when Target Cooling Temperature is reached (TS2).

\* \*\*KT9 = KA1 \* KM5\*\* (Discharge Duration Timer): Start timing when Discharge Valve is ON (KM5).

\* \*\*KT10 = KA1 \* KT9\*\* (Discharge Phase Complete Delay): Start timing after Discharge Duration Timer (KT9).

This example is the most complex so far, with multiple sequential stages, timers, sensors, and fault detection. The logical functions are becoming increasingly sophisticated, and the use of intermediate relays is critical for managing the state transitions and ensuring correct sequencing and safety interlocks. The fault handling logic adds another dimension to the design.

、、、、、、、、、、、、、、、