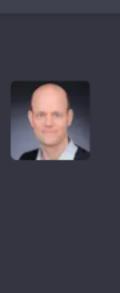
Model: GPT-4



Create a P&I diagram in textual notation for a distillation column including process equipment, instrumentation, control functions, safety interlocks, and piping, Use

typical tagnames.

In this textual representation of a P&I diagram for a distillation column, we'll use typical tagnames to describe the various components, instrumentation, control functions, safety interlocks, and piping involved in the process. 1. Process Equipment

- R-101: Reflux Drum R-102: Reboiler
- P-101A/B: Feed Pump (A and B)
 - HX-101: Feed Preheater HX-102: Condenser

DC-101: Distillation Column

2. Instrumentation

FT-101: Feed Flow Transmitter

LT-102: Reboiler Level Transmitter

- LT-101: Reflux Drum Level Transmitter
- TT-101: Feed Temperature Transmitter
- TT-102: Distillate Temperature Transmitter PT-101: Distillation Column Pressure Transmitter
- 3. Control Functions
- FIC-101: Feed Flow Control
- LIC-101: Reflux Drum Level Control

LIC-102: Reboiler Level Control

LSH-101: Reflux Drum High-Level Switch

LSL-102: Reboiler Low-Level Switch

PI-106: Steam Supply Piping

The above textual representation provides a clear overview of the different

distillation column. Provide concrete values for the high limits and low limits.

control the safety interlocks of the distillation column described earlier. We'll use

Copy code

the following concrete values for the high limits and low limits:

- TIC-101: Feed Temperature Control TIC-102: Distillate Temperature Control
- PIC-101: Distillation Column Pressure Control
- 4. Safety Interlocks
- LSL-101: Reflux Drum Low-Level Switch LSH-102: Reboiler High-Level Switch
- PSH-101: Distillation Column High-Pressure Switch PSL-101: Distillation Column Low-Pressure Switch
- 5. Piping
- PI-101: Feed Inlet Piping
- PI-102: Distillate Outlet Piping PI-103: Bottoms Outlet Piping
- PI-104: Reflux Piping PI-105: Condensate Piping
- PI-107: Cooling Water Inlet Piping PI-108: Cooling Water Outlet Piping
- components and their functions within a distillation column process.

ம

- Write a 61131-3 structured text program to implement the interlocks of this
- In this example, we'll implement a simple 61131-3 structured text program to

 Distillation Column Low Pressure: 300 kPa Here is the structured text program:

Reflux Drum High Level: 90%

Reflux Drum Low Level: 10%

Reboiler High Level: 90%

Reboiler Low Level: 10%

- Distillation Column High Pressure: 1200 kPa

- plaintext PROGRAM DistillationColumnInterlocks

VAR

LT_101: REAL := 0; (* Reflux Drum Level *) LT_102: REAL := 0; (* Reboiler Level *)

```
LSH_101: BOOL := FALSE; (* Reflux Drum High-Level Switch *)
      LSL_101: BOOL := FALSE; (* Reflux Drum Low-Level Switch *)
      LSH_102: BOOL := FALSE; (* Reboiler High-Level Switch *)
      LSL_102: BOOL := FALSE; (* Reboiler Low-Level Switch *)
      PSH_101: BOOL := FALSE; (* Distillation Column High-Pressure Switch *)
      PSL_101: BOOL := FALSE; (* Distillation Column Low-Pressure Switch *)
  END_VAR
  METHOD UpdateInterlocks()
      (* Reflux Drum Level Interlocks *)
      LSH_101 := LT_101 >= 90;
      LSL_101 := LT_101 <= 10;
      (* Reboiler Level Interlocks *)
      LSH_102 := LT_102 >= 90;
      LSL_102 := LT_102 <= 10;
      (* Distillation Column Pressure Interlocks *)
      PSH_101 := PT_101 >= 1200;
      PSL_101 := PT_101 <= 300;
  END_METHOD
  END_PROGRAM
This structured text program defines variables for the level and pressure
transmitters (LT_101, LT_102, and PT_101) and the interlock switches (LSH_101,
LSL_101, LSH_102, LSL_102, PSH_101, and PSL_101). The UpdateInterlocks() method
updates the interlock switch states based on the current level and pressure
readings.
```

PT_101: REAL := 0; (* Distillation Column Pressure *)

Extend the program with the actions that are performed when the interlocks are activated.

In this extension of the structured text program, we'll add actions to be performed

when the interlocks are activated. These actions can include shutting down certain

Copy code

In a real-world implementation, you would periodically call the UpdateInterlocks()

method and integrate it with your control system to act on the interlock switch

states, such as shutting down the process or triggering alarms.

equipment, triggering alarms, or engaging emergency systems.



plaintext

LT_101: REAL := 0; (* Reflux Drum Level *) LT_102: REAL := 0; (* Reboiler Level *) PT_101: REAL := 0; (* Distillation Column Pressure *)

```
PROGRAM DistillationColumnInterlocks
  VAR
      LSH_101: BOOL := FALSE; (* Reflux Drum High-Level Switch *)
      LSL_101: BOOL := FALSE; (* Reflux Drum Low-Level Switch *)
      LSH_102: BOOL := FALSE; (* Reboiler High-Level Switch *)
      LSL_102: BOOL := FALSE; (* Reboiler Low-Level Switch *)
     PSH_101: BOOL := FALSE; (* Distillation Column High-Pressure Switch *)
     PSL_101: BOOL := FALSE; (* Distillation Column Low-Pressure Switch *)
     Shutdown: BOOL := FALSE; (* Shutdown flag *)
     Alarm: BOOL := FALSE; (* Alarm flag *)
  END VAR
 METHOD UpdateInterlocks()
      (* Reflux Drum Level Interlocks *)
      LSH_101 := LT_101 >= 90;
      LSL_101 := LT_101 <= 10;
      (* Reboiler Level Interlocks *)
      LSH_102 := LT_102 >= 90;
      LSL_102 := LT_102 <= 10;
      (* Distillation Column Pressure Interlocks *)
      PSH_101 := PT_101 >= 1200;
      PSL_101 := PT_101 <= 300;
     IF LSH_101 OR LSL_101 OR LSH_102 OR LSL_102 OR PSH_101 OR PSL_101 THEN
          Alarm := TRUE;
      ELSE
          Alarm := FALSE;
      END_IF;
      IF LSH_101 OR LSH_102 OR PSH_101 THEN
          Shutdown := TRUE;
      ELSE
          Shutdown := FALSE;
      END_IF;
  END_METHOD
 METHOD PerformActions()
      IF Alarm THEN
          (* Implement Alarm logic, such as notifying operators or logging ev
      END_IF;
      IF Shutdown THEN
          (* Implement Shutdown logic, such as stopping equipment and engagir
      END_IF;
  END_METHOD
  END_PROGRAM
In this updated program, we've added a Shutdown flag and an Alarm flag. The
UpdateInterlocks() method now sets these flags based on the state of the interlock
switches. The PerformActions() method has been introduced to implement the
```

actions that are performed when the interlocks are activated. In a real-world implementation, you would replace the comments in the PerformActions() method with the appropriate logic to handle alarms and shutdowns. Periodically, you would call both the UpdateInterlocks() and PerformActions() methods to ensure that the interlocks are up-to-date, and the necessary actions are performed when the interlocks are activated.