



Write a self-contained IEC 61131-3 Structured Text program to implement PID feedback control for the temperature inside a gas turbine by providing a setpoint for the opening of an inlet valve.



The following Structured Text (ST) program is designed to implement a PID feedback control loop for the temperature inside a gas turbine by providing a setpoint for the opening of an inlet valve according to the IEC 61131-3 standard. This program uses the TON (timer on-delay) and PID (Proportional-Integral-Derivative) function blocks. The TON function block is used to implement a fixed-time interval for the PID control loop execution.

Note that you might need to adapt the variable names and data types to fit your specific PLC programming environment.

iechstCopy code

```
PROGRAM PIDControl;

VAR
    // Inputs
    TurbineTemperature: REAL;
    TemperatureSetpoint: REAL;

    // Outputs
    InletValveOpening: REAL;

    // Internal variables
    LoopInterval: TIME := T#100ms;
    LoopTimer: TON;
    PIDControl: PID;
    PIDOutput: REAL;
    PIDParameters: PID_PARAM := (Kp := 10, Ki := 0.01, Kd := 1, dt := 0.1, Eps := 1, Min := 0, M

END_VAR;

METHOD RunControlLoop: BOOL;
BEGIN
    LoopTimer(IN := NOT LoopTimer.Q, PT := LoopInterval);
    IF LoopTimer.Q THEN
        PIDControl
        (
            SET_PARAMS := TRUE,
            PARAMS := PIDParameters,
            SP := TemperatureSetpoint,
            PV := TurbineTemperature,
            Y => PIDOutput
        );
        InletValveOpening := PIDOutput;
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END_IF;
END_METHOD;

END_PROGRAM;
```

In this example, the inputs are `TurbineTemperature` and `TemperatureSetpoint`. The output is `InletValveOpening`. The PID control loop is executed every 100 milliseconds (defined by `LoopInterval`). The PID parameters (Kp, Ki, Kd, and other settings) are configured in the `PIDParameters` variable.

The `RunControlLoop` method checks if the `LoopTimer` has elapsed and, if so, runs the PID control loop, updating the output `InletValveOpening`. The method returns TRUE if the control loop has been executed and FALSE otherwise.



Revise the code to not use the PIDControl function block and to use VAR\_INPUT and VAR\_OUTPUT.



The following revised Structured Text (ST) program implements a PID feedback control loop for the temperature inside a gas turbine without using the PIDControl function block, and it uses VAR\_INPUT and VAR\_OUTPUT for defining input and output variables according to the IEC 61131-3 standard.

iechstCopy code

```
PROGRAM PIDControl;

VAR_INPUT
    TurbineTemperature: REAL;
    TemperatureSetpoint: REAL;
END_VAR;

VAR_OUTPUT
    InletValveOpening: REAL;
END_VAR;

VAR
    // Internal variables
    LoopInterval: TIME := T#100ms;
    LoopTimer: TON;
    PIDOutput: REAL;

    // PID parameters
    Kp: REAL := 10;
    Ki: REAL := 0.01;
    Kd: REAL := 1;
    dt: TIME := T#100ms;

    // PID internal variables
    previousError: REAL := 0;
    integral: REAL := 0;
    derivative: REAL := 0;
    error: REAL;
    deltaTime: REAL;
END_VAR;

METHOD RunControlLoop: BOOL;
BEGIN
    LoopTimer(IN := NOT LoopTimer.Q, PT := LoopInterval);
    IF LoopTimer.Q THEN
        deltaTime := TIME_TO_REAL(dt);
        error := TemperatureSetpoint - TurbineTemperature;

        integral := integral + error * deltaTime;
        derivative := (error - previousError) / deltaTime;

        PIDOutput := Kp * error + Ki * integral + Kd * derivative;

        InletValveOpening := LIMIT(0, PIDOutput, 100);

        previousError := error;
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END_IF;
END_METHOD;

END_PROGRAM;
```

In this version of the program, the PID control loop is implemented manually within the `RunControlLoop` method. The PID algorithm uses the `Kp`, `Ki`, and `Kd` parameters, along with the internal `integral`, `derivative`, and `previousError` variables. The `RunControlLoop` method calculates the error, updates the integral and derivative values, calculates the PID output, and limits the `InletValveOpening` value between 0 and 100. The method returns TRUE if the control loop has been executed and FALSE otherwise.

