Write a self-contained function block in 61131-3 structured text that implements a PID controller.

A PID controller is a control loop mechanism that calculates an error value as the difference between a measured process variable and a desired setpoint. The PID controller attempts to minimize the error by adjusting the process control inputs. The PID controller algorithm involves three separate parameters: the proportional, the integral, and the derivative values, which are denoted as P, I, and D, respectively.

Here's a self-contained function block in IEC 61131-3 Structured Text that implements a PID controller:

iecst Copy code

```
FUNCTION_BLOCK PID_Controller
VAR_INPUT
    SP : REAL; // Setpoint
    PV : REAL; // Process Variable
    Ts : REAL; // Sampling Time (s)
    Kp : REAL; // Proportional Gain
    Ki : REAL; // Integral Gain
    Kd : REAL; // Derivative Gain
END_VAR

VAR_OUTPUT
    CV : REAL; // Control Variable
END_VAR

VAR
    e_prev : REAL; // Previous Error
    I_sum : REAL; // Sum of Integral Errors
    D_diff : REAL; // Derivative Error Difference
END_VAR

METHOD RunController : REAL
    VAR
        error : REAL;
        P_term : REAL;
        I_term : REAL;
        D_term : REAL;
    END_VAR

    // Calculate Error
    error := SP - PV;

    // Calculate Proportional Term
    P_term := Kp * error;

    // Calculate Integral Term
    I_sum := I_sum + (Ki * error * Ts);
    I_term := I_sum;

    // Calculate Derivative Term
    D_diff := (error - e_prev) / Ts;
    D_term := Kd * D_diff;

    // Update Previous Error
    e_prev := error;

    // Calculate Control Variable
    CV := P_term + I_term + D_term;

    RETURN CV;
END_METHOD
END_FUNCTION_BLOCK
```

To use this function block, you need to create an instance of it and call the **`RunController`** method with appropriate inputs. This function block uses the standard form of the PID algorithm, and you may need to modify it according to your specific application or control requirements.