Write a self-contained function block in IEC 61131-3 Structured Text to compute a sine function.

Here's a self-contained function block in IEC 61131-3 Structured Text for computing the sine function:

iecst Copy code

```
FUNCTION_BLOCK FB_SineFunction
VAR_INPUT
    AngleRad: REAL; // Input angle in radians
END_VAR

VAR_OUTPUT
    SineValue: REAL; // Output sine value
END_VAR

METHOD ComputeSine: REAL
VAR
    n: INT;
    x, term, sum: REAL;
END_VAR

    sum := 0;
    x := AngleRad;

    // Using Taylor Series for sine function (up to 10 iterations)
    FOR n := 0 TO 9 DO
        term := (((-1) ** n) * (x ** (2 * n + 1))) / (FACTORIAL(2 * n + 1));
        sum := sum + term;
    END_FOR;

    ComputeSine := sum;
END_METHOD

METHOD FACTORIAL: REAL
VAR_INPUT
    n: INT; // Integer input for factorial calculation
END_VAR
VAR
    i: INT;
    result: REAL;
END_VAR

    result := 1;

    IF n >= 0 THEN
        FOR i := 1 TO n DO
            result := result * i;
        END_FOR;
    END_IF;

    FACTORIAL := result;
END_METHOD

END_FUNCTION_BLOCK
```

This function block, `FB_SineFunction`, contains two methods:

1. `ComputeSine`: Computes the sine value using the Taylor series approximation (up to 10 iterations).
2. `FACTORIAL`: Computes the factorial of a given integer value.

To use this function block in your code, create an instance of the `FB_SineFunction` and call the `ComputeSine` method with the desired angle in radians. For example:

iecst Copy code

```
PROGRAM Main
VAR
    SineCalc: FB_SineFunction;
    InputAngle: REAL := 1.57; // Angle in radians
    SineResult: REAL;
END_VAR

SineCalc.AngleRad := InputAngle;
SineResult := SineCalc.ComputeSine();
END_PROGRAM
```

This example sets the `InputAngle` variable to 1.57 radians (approximately equal to pi/2) and computes the sine value using the `ComputeSine` method. The result is stored in the `SineResult` variable.