SyntaxShift — Understand PLC Programs. Document. Modernize.

AI-powered analysis and documentation of existing PLC software for safe retrofit projects

1 **Understand** – Analyze program flows, explain function blocks, describe variables

Source

Source Code: + Upload

```
```
// IMPORTANT NOTE FOR SYNTAXSHIFT:
// Please translate and generate everything in English.
// This includes code comments, variable descriptions, explanations,
// documentation, labels, and any demo video narration.

// Inputs
I 0.0   Manual_On   // Manual mode ON
MD 100   Temperature_C   // Temperature in °C (REAL)
MW 200   CO2_ppm   // CO₂ concentration in ppm (INT)

// Outputs
Q 0.0   Ventilation   // Main fan ON/OFF
Q 0.1   Stage1   // Fan speed level 1
Q 0.2   Stage2   // Fan speed level 2
Q 0.3   Alarm   // Alarm indicator

// Internal flags
M 10.0   vent_on_req   // Ventilation request
M 10.1   stage1_req   // Stage 1 request
M 10.2   stage2_req   // Stage 2 request

M 11.0   Temp_Alarm   // Temperature alarm
M 11.1   CO2_Alarm   // CO₂ alarm
```

Explain

### Brief Description

The AWL code implements a simple ventilation control with two automatic criteria:

- manual enable via `I 0.0`
- $CO_2$-based switching with two thresholds
- temperature-based switching with two thresholds
- alarm generation for high temperature or high $CO_2$

Technically, the program clears internal request bits at the beginning of each scan, then evaluates manual mode first, otherwise $CO_2$ logic, then temperature logic, writes the outputs, and finally updates the alarm bits and alarm output.

### Program Structure

The program is structured linearly with labels and jumps rather than formal network separators.

Recognizable sections:

- **Reset all requests**

2 **Document** – Automatic function and variable descriptions

Edit Output:

```
edit co2 ppm to 2000
yes
```

```
PROGRAM VentilationControl
VAR
    Manual_On : BOOL;
    Temperature_C : REAL;
    CO2_ppm : INT;
    Ventilation : BOOL;
    Stage1 : BOOL;
    Stage2 : BOOL;
    Alarm : BOOL;
    ventOnReq : BOOL;
    stage1Req : BOOL;
    stage2Req : BOOL;
    tempAlarm : BOOL;
    co2Alarm : BOOL;
END_VAR

ventOnReq := FALSE;
stage1Req := FALSE;
stage2Req := FALSE;

IF Manual_On THEN
    ventOnReq := TRUE;
    stage1Req := TRUE;
ELSE
    IF CO2_ppm > 2000 THEN
        ventOnReq := TRUE;
        stage2Req := TRUE;
```

3 **Engineering Assistant** – Get instant answers about your code

block that contains both code and data, capable of storing its input, output, and internal variables over multiple execution cycles.

Explanation
Function Blocks are used for tasks that require state retention, such as complex calculations or logic involving timers and counters. Each call to a Function Block requires an Instance Data Block to store its data. This allows multiple instances of the same Function Block to operate independently.

Example
When you call a Function Block, you specify parameters for operation. For example, calling a motor control FB might involve parameters for speed and torque, which are stored in the Instance Data Block for that particular instance.

Common Pitfalls
A common challenge is ensuring that Instance Data Blocks are properly managed, especially when the same FB is used multiple places within a program, as this can lead to incorrect data management if not handled correctly.

Relevance for Conversion
When converting code, awareness of how the FB

More features: IL to Structured Text | IL to Siemens SCL | Version control | Project upload

The biggest challenge in retrofit is not the translation – it's **understanding existing PLC programs**.

SyntaxShift analyzes, explains, and documents automatically – the foundation for safe modernization.
```