Exercise 2

Contents

Introduction

In this exercise you will learn about the different POUs (Programmable Organization Units) and how to use ENUMs.

Exercises

1. Connect PC and PLC

Follow the same steps as last time of the "TwinCAT Guide":

1. Power the PLC
2. Establish connection
3. Open System Manager
4. Select Target System
5. Run Mode

2. Read the ST Reference

1. Read about the different POUs (Functions, Programs, Function Blocks) and their characteristics.
2. Read about ENUMs.

3. Function exercise

1. Create a new project with a MAIN program.
2. Create the function "Addition" that takes the REAL inputs "a" and "b".
3. Inside "Addition" add the 2 inputs and store them in a local variable called temp.
4. Return the value of temp.
5. Call this function from your MAIN program with different input values.
6. Run the program and use the debugger to observe, that ...
    a. the values from MAIN are copied into the function
    b. the temp variable inside the function does not remember it's value from the last time it was assigned

4. Program exercise

1. Create the program "DoSomeMath" and call it from MAIN
2. Create a counter inside "DoSomeMath" that remembers how many times the program has been executed
3. Use the debugger to read the value of the counter.

5. Function Block exercise

1. Create the function block "SimpleCalculator" with REAL inputs "a" and "b", REAL outputs "resultAdd", "resultSub", "resultDiv" and "resultMul".
2. Program the logic to achieve the math operations: addition, subtraction, multiplication and division
3. Create two instances of SimpleCalculator in DoSomeMath and call them with different input values for a and b.
4. Observe the execution with the debugger.
5. In Tab Data types create a new ENUM "MathOp" with the entries: MathOpAdd, MathOpSub, MathOpMul, MathOpDiv
6. Create a function block "AdvancedCalculator" with REAL inputs "a" and "b", BOOL input useLastResult, MathOp "op" and REAL output "result"
7. A local variable is needed to remember the last result of the previous calculation
8. In the AdvancedCalculator use a CASE switch to do the requested mathematical operation. For instance if op is MathOpAdd then a is added to b.
9. Store the result of the operation inside the AdvancedCalculator. If input useLastResult is TRUE use the stored last result instead of input b.
10. Return the result.
11. Create two instances of AdvancedCalculator in DoSomeMath and call them with different input values for a, b, useLastResult and op. Observe the execution with the debugger.

6. Extended Calculator

Extend the AdvancedCalculator such that:

1. It can store up to 3 results.
2. The user can decide which of the 3 results will be overwritten. Use an ENUM for this.
3. The user can choose any of the 3 results to be used in the calculation instead of variable b when useLastResult is TRUE.