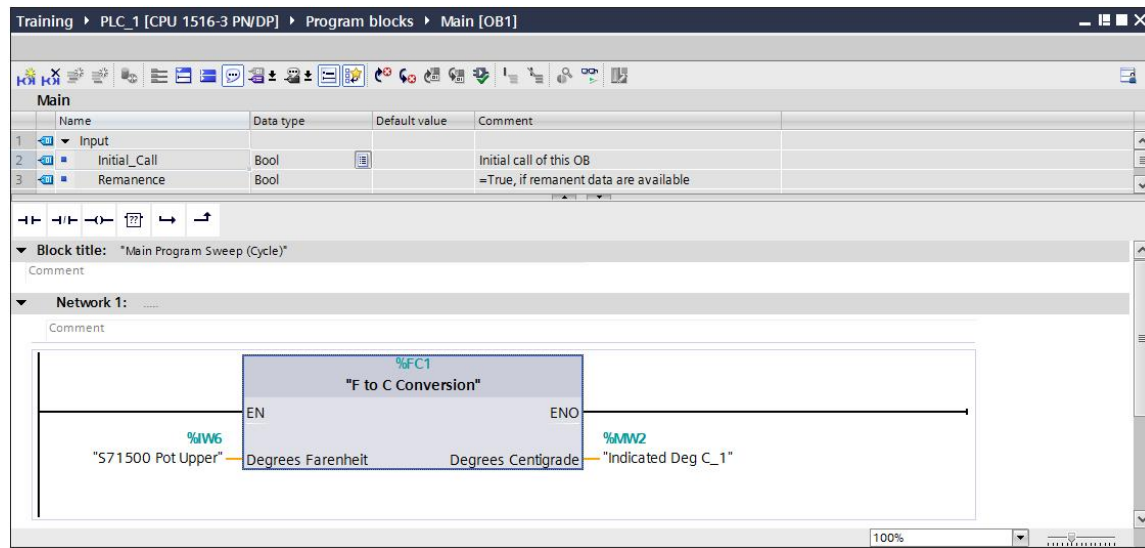


Chapter 3 Exercises

3.1.1	Create a new FC.....	3
1)	Create a new FC.....	3
2)	Add Parameters to the New FC for reusability	3
3)	Add necessary code (in LAD) to perform scaling and conversion.....	4
4)	Call the newly created Function in the Main [OB1] routine.	5
5)	Download the program changes to the PLC.....	6
6)	Add a second call of the new function.	6
7)	Monitor a specific instance of a reusable block	7
3.2.1	Create a New Global Data Block.....	9
1)	Create a new Global Data Block.	9
2)	Add Data elements to the new data block.	9
3)	Create a Watch Table for changing data block values	10
4)	Manage Data Block values and operations.	11
3.2.2	Insert and Parameterize FB Motor Starter.....	15
1)	Add a new FB to the program.	15
2)	Add the Local Input, Output and Static Parameters to the declaration table.....	16
3)	Program the FB (in LAD) for reusable toggle functionally.....	17
3.3.1	Test Functionality.....	18
1)	Test the new FB functionality in Main [OB1].	18
3.4.1	Modify “FB Motor Starter” for Motor Starts Counter Control.....	21
1)	Modify Local Input, Output and Static to the Parameters in the declaration area. 21	
2)	Instantiate CTU for Count Up control.	22
3)	Program Network 2 for Reusability by using block parameters and local variables.....	23
3.4.2	Test Functionality.....	24
1)	Test Functionality in OB1.....	24
3.5.1	Create FB Motor Controls to Control Multiple Motor Starters.....	26
1)	Create a new FB.....	26
2)	Parameterize FB Motor Controls Input and Output declarations to control the Logic. 27	
3)	Instantiate FB Motor Starter into FB Motor Controls for Motor Starter 1 Control.	28
4)	Repeat the previous steps for Network 2 for a call to Motor 2.	29
5)	Observe the Static area in FB Motor Controls.	30
3.5.2	Complete the calls of FB Motor Controls.....	31
1)	Call FB Motor Controls in OB1.	31
3.5.3	Testing and Monitoring the Motor Starters	32
1)	Monitor the Motor Starter Status from OB1.....	32
3.6.1	Create a new PLC Data Type	35
1)	Create PLC Data Type “Motor Control”	35

3.1 Create and Call a parameterized FC



Description

Functions are used to program frequently recurring or complex automation functions. They can be parameterized and return a value (called the function value) to the calling block. The function value is optional. In addition to the function value, functions may also have other input and output parameters. Functions do not store information, and have no assigned data block.

In this exercise, parameters will be added to an FC to gain reusability of the FC. This will allow the local logic in the block to be parameterized generically; enabling different "Real World" addresses to be passed into the call for each instance the FC is used.

Objectives

Upon completion of this exercise, the student shall be able to:

- Configure a Reusable Function
- Use Formal Parameters for block reusability
- Use LAD to program the Function
- Call and Test the Function in OB1 twice

Prerequisites

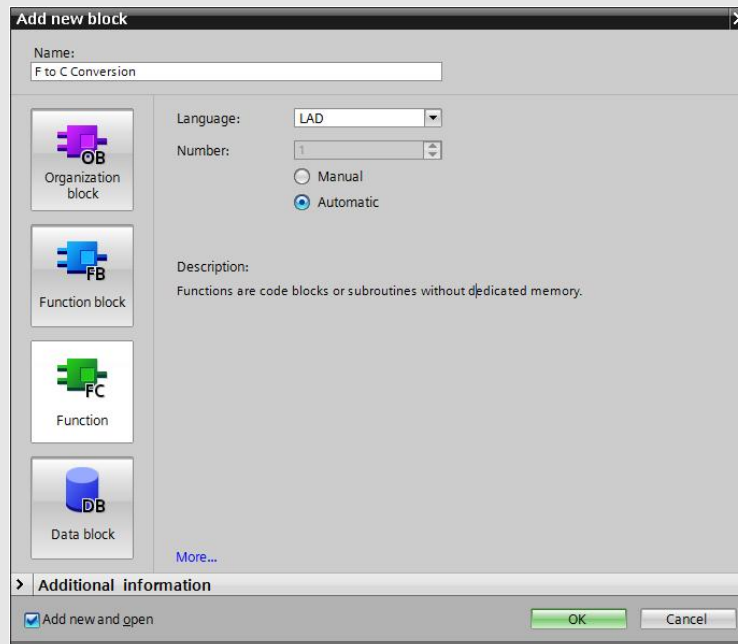
The following prerequisites must be completed before this exercise is started:

- Chapter 2 Exercise completed, or the Chapter 3 Seed Project has been retrieved and opened for editing.

3.1.1 Create a new FC

1) Create a new FC.

1. From the Project Tree, expand the branch under the S7-1500 “PLC_1” labeled “Program blocks”. Double click on “Add new block” to open the Add new block dialog.
2. Select the “Function” button. Assign the Name “F to C Conversion”. Make sure “Add new and open” is selected. Click “OK” to create and open the block.



2) Add Parameters to the new FC for reusability

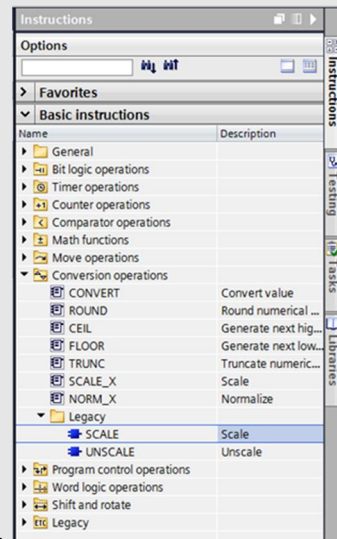
1. Access the FC declaration area at the top of the editor. If it isn't visible, drag the divider down to expose the parameter editor.
2. Define the block parameters as shown below. Pay close attention to data types and default values.

	Name	Data type	Default value
1	Input		
2	Degress Farenheit	Int	
3	Output		
4	Degrees Centigrade	Int	
5	InOut		
6	<Add new>		
7	Temp		
8	Scaled Degrees Farenheit	Int	
9	Scale_RET_VAL	Int	
10	<Add new>		
11	Constant		
12	<Add new>		
13	Correction - Bias	Int	32
14	Correction - Gain Numerator	Int	5
15	Correction - Gain Denominator	Int	9
16	SCALE_HI_LIM	Real	500.0
17	SCALE_LO_LIM	Real	0.0
18	UNIPOLAR	Bool	false
19	<Add new>		
20	Return		
21	F to C Conversion	Void	

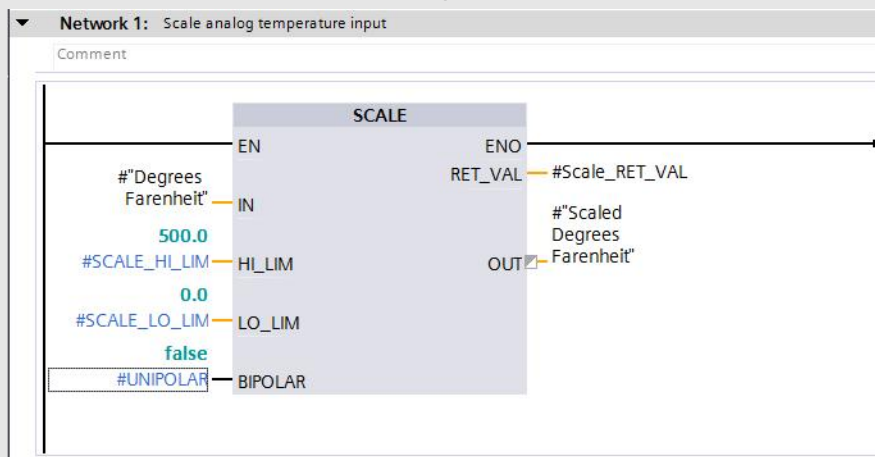
- 3) Add necessary code (in LAD) to perform scaling and conversion.

NOTE: The analog input value from the potentiometer is unscaled and considered “raw”. Before we use this value, we need to “scale” the value into a meaningful analog value.

1. In Network 1 of the work area, add a SCALE instruction (NOT Scale_X). This can be accomplished by selecting it from the Instruction browser – “Basic Instructions”, “Conversion operations”, “Legacy”, “SCALE”




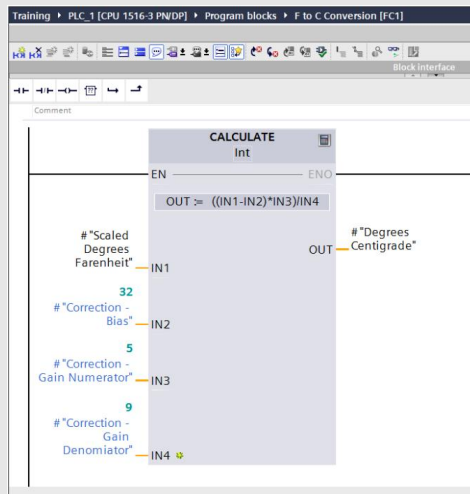
2. Our input signal is 0-10VDC. Parameterize the new block as shown below. This will provide a value between 0-500 for the input raw value of 0-27648 units.



3. The next network will contain the equation for converting Degrees F to Degrees C. Rather than using a series of simple math instructions or using a pure text-based language, let's use the CALCULATE instruction.

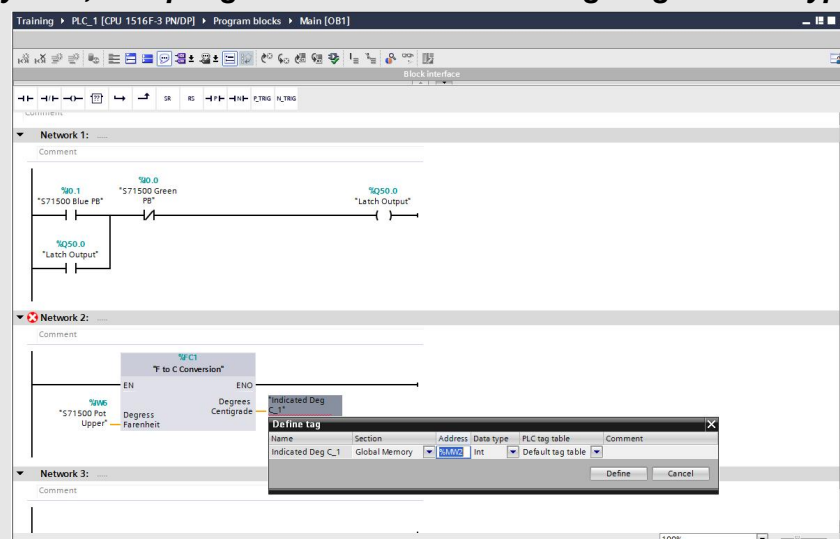
$$\text{DEG C} = (\text{DEG F} - 32) * (5/9)$$

4. Add the CALCULATE instruction to Network 2. It's found under "Basic instructions, Math functions, CALCULATE".
5. Add two inputs to the CALCULATE instruction by clicking the "starburst"  next to IN2. Each input requires a click to add.
6. Set the instruction data type to INT
7. Parameterize the block as shown below.



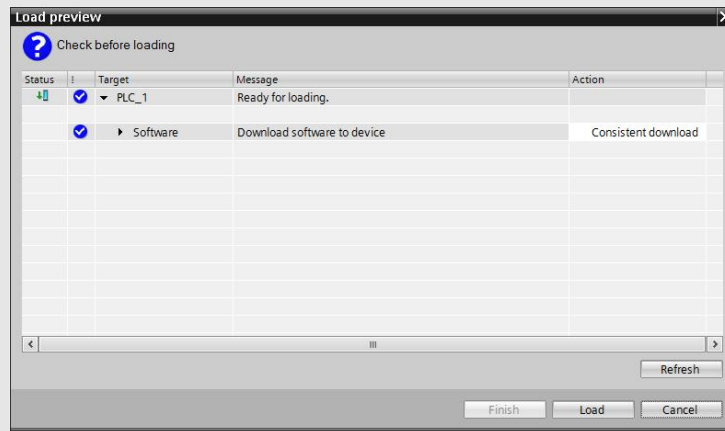
8. Enter the equation in the CALCULATE box field as " $((IN1-IN2)*IN3)/IN4$ ".
9. Save your project.

- 4) Call the newly created Function in the Main [OB1] routine.
 1. From the Project tree, double-click on Main [OB1] to open it for editing. Delete the latch in Network 1.
 2. Drag the newly created Function from the project tree and drop it on Network 2.
 3. Parameterize the new Function call as shown. **Note the tag "Indicated Deg C_1" must be added. Create the tag in the Default tag table, assign the tag in the Global Memory area, accepting the default address and giving it a data type of INT.**



5) Download the program changes to the PLC.

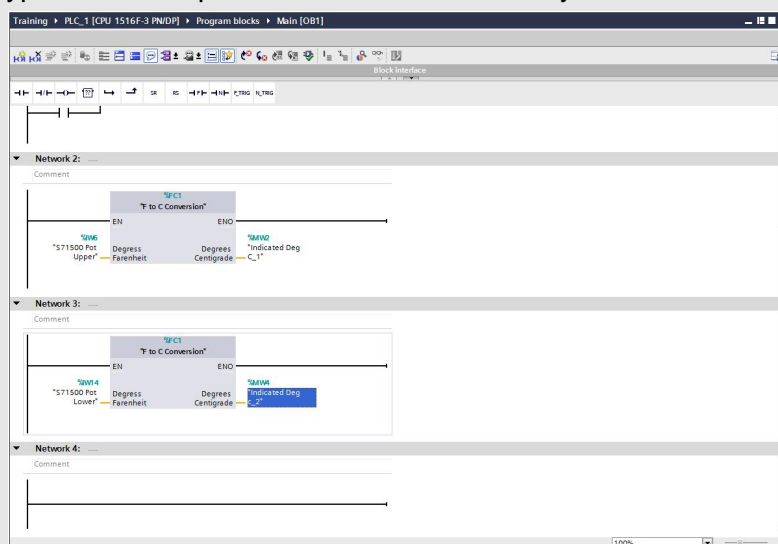
1. From the Project tree, select the Program blocks folder of PLC_1. Click the “Download” button to download the changes.
2. The Load preview dialog box will appear, informing you of what actions are about to be executed.



3. Open the Main [OB1] routine, then select the “Monitor” button. Adjust the potentiometer as desired and observe the operation.

6) Add a second call of the new function.

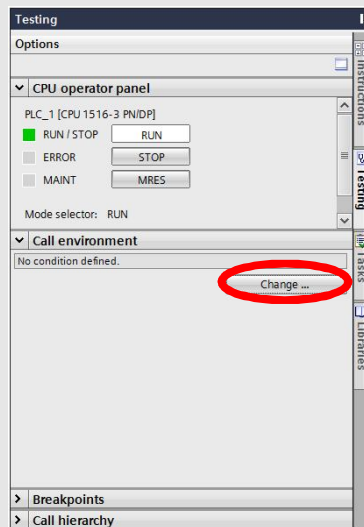
1. Stop the Monitor operation in progress to allow for program editing.
2. From the Project tree, add a second call of F to C Conversion to Network 3.
3. Parameterize the new call as shown below. Note you will need to add a second indication tag as before with the name as shown below. Add it to the Default tag table with data type INT. Accept the default Global memory address.



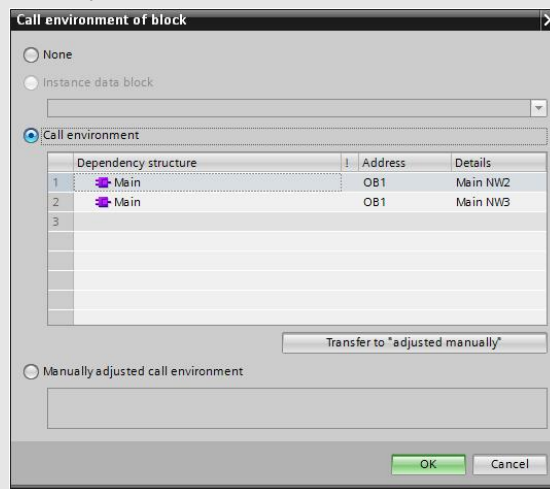
4. Download your changes and test.

7) Monitor a specific instance of a reusable block

1. Now that we're using a block multiple times, it is necessary to specify which instance you'd like to monitor. **From the Project tree**, open F to C Conversion [FC1] for editing.
2. Enable the monitor function. Change the values of both potentiometers. Note the data being displayed only changes with one potentiometer, and we're not really sure which one is changing.
3. To define which call you'd like to monitor, the "Call environment" must be defined. This is done using the "Call environment" tab of the Task cards. Click the "Change" button.



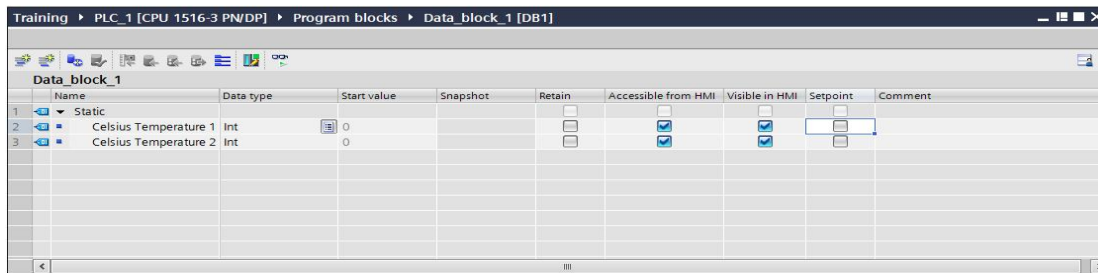
4. In the dialog that appears, each call of the function is listed. Select "Call environment", then click which instance you'd like to monitor.



5. Verify that each potentiometer's operation can be observed.
6. In the S7-1500, you can also simply open the instance you'd like to view from the calling block.
7. Save your project.

This completes exercise 3.1

3.2 Create and Use a Global Data Block



Description

Data storage and management is a key part of application program development. STEP7 provides the user with the ability to create and manage data storage containers called Data Blocks to assist in data management.

In this exercise, we will create a global data block to hold data values currently being held by values in the Global Memory area. Use of the Global Memory area, while not incorrect in many cases, should be avoided when possible to better prepare programs for portability across platforms. Using Global Data Blocks helps meet this criterion.

Objectives

Upon completion of this exercise, the student shall be able to:

- Create and use a Global Data Block.
- Add new data elements.
- Create a Watch table to be able to change the data block values.
- Manage Data Block values and operations.

Prerequisites

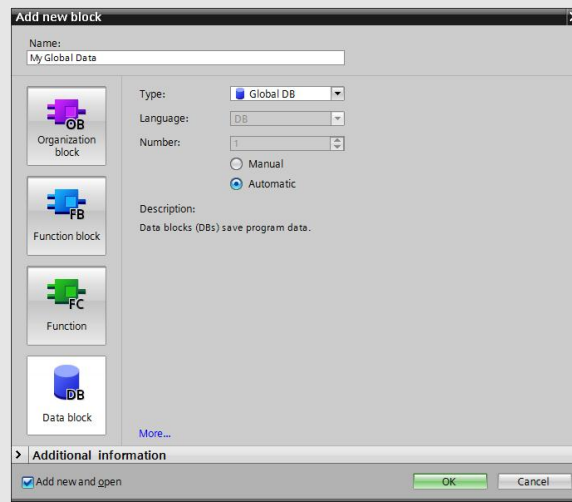
The following prerequisites must be completed before this exercise is started:

- Exercise 4.1 has been completed.

3.2.1 Create a New Global Data Block

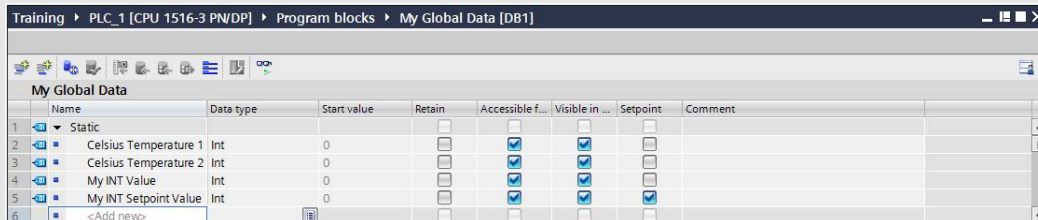
1) Create a new Global Data Block.

1. From the Project Tree, expand the branch under the S7-1500 “PLC_1” labeled “Program blocks”. Double click on “Add new block” to open the Add new block dialog.
2. Select the “Data block” button. Assign the Name “My Global Data”. Make sure the “Type” is Global DB and “Add new and open” is selected. Click “OK” to create and open the block.



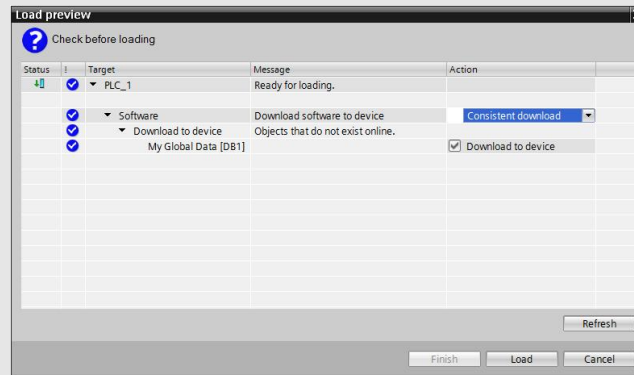
2) Add Data elements to the new data block.

1. Enter new INT data elements to the data block named “Celsius Temperature 1”, “Celsius Temperature 2”, “My INT Value”, and “My INT Setpoint Value”. Set the properties of each element as shown.

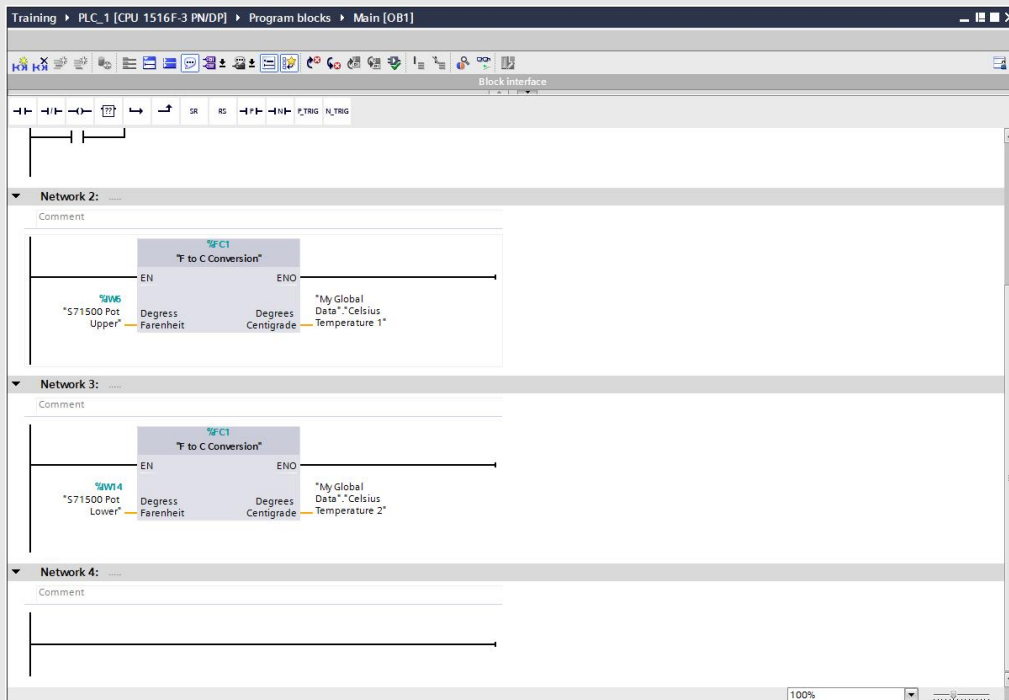


	Name	Data type	Start value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Static							
2	Celsius Temperature 1	Int	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	Celsius Temperature 2	Int	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4	My INT Value	Int	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
5	My INT Setpoint Value	Int	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	<Add new>							

2. Download the new data block into the controller.



3. Open the Main [OB1] block for editing. Change the current tags in use for the F to C Conversion functions to the new data block elements just created. See the figure below.



4. Download Main [OB1].
5. Open the new data block "My Global Data" and enable the Monitor function. Test the program changes for correct operation.

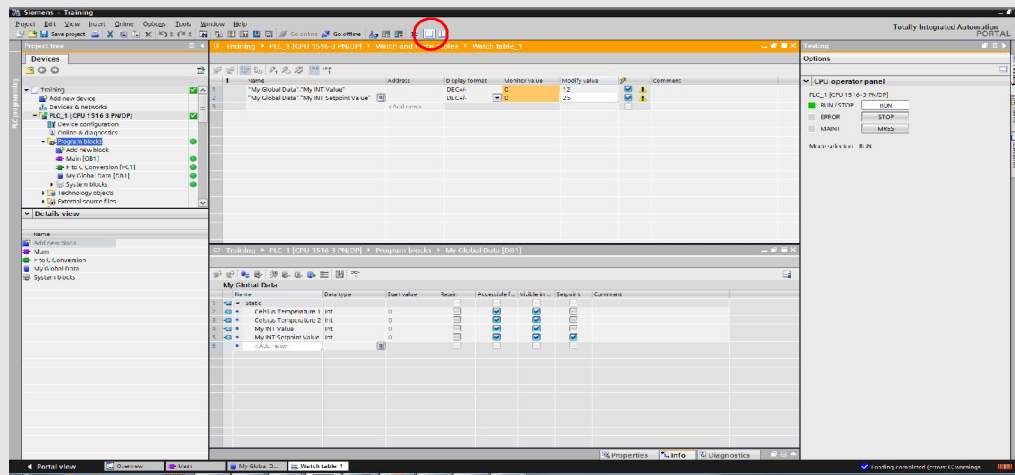
3) Create a Watch Table for changing data block values


1. From the Project tree, expand the branch labeled under PLC_1 labeled "Watch and force tables".
2. Double click on "Add new watch table" to create a new table. Accept the default name.
3. Using the "tag browse" function, add the tags "My Global Data", "My INT Value" and "My Global Data", "My INT Setpoint Value" to the watch table.

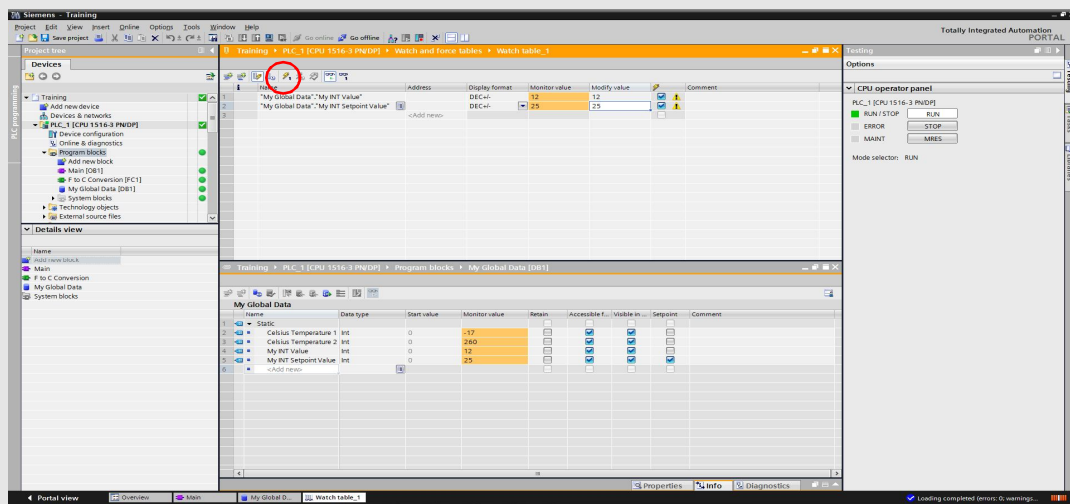
	Name	Address	Display format	Monitor value	Modify value	Comment
1	"My Global Data"					
2	None					
	"Celsius Temperature 1"	Int				
	"Celsius Temperature 2"	Int				
	"My INT Setpoint Value"	Int				
	"My INT Value"	Int				


4) Manage Data Block values and operations.

1. Click the “Split editor space horizontally”  so that the watch table and “My Global Data” can both be viewed. Enable “Monitor” in both editors.

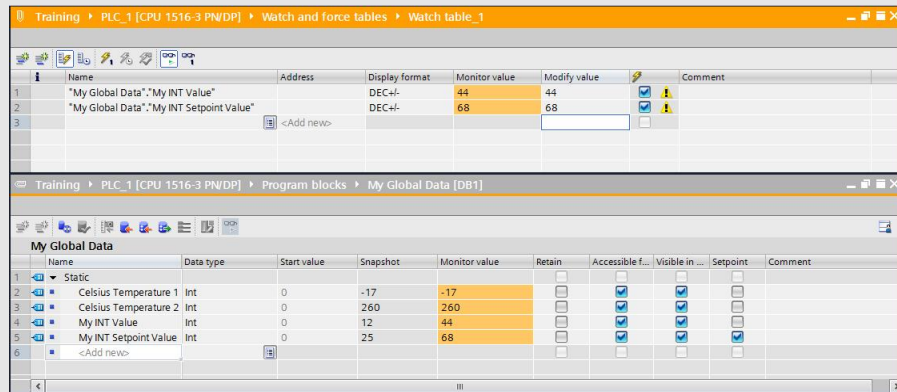


2. Enter new values in the watch table for “My INT Value” and “My INT Setpoint Value”. In the “Modify value” column. Note that the checkbox under the “lightning bolt” is checked, indicating that the “Modify” operation will affect these tags.
3. Click the “Modify all selected values once and now” button.  Note the change to the data values in both editors.

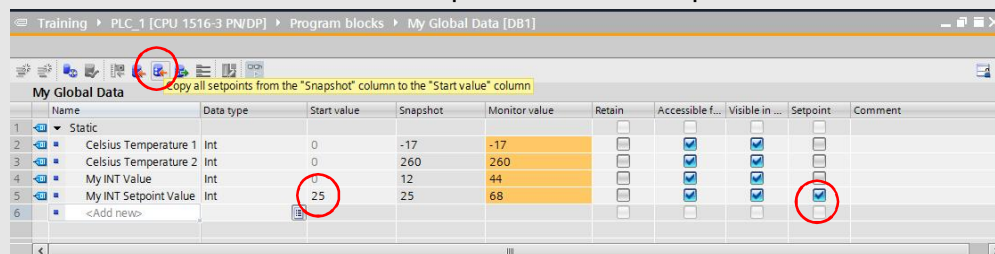


4. In the data block editor, click the Snapshot button.  Note the additional column that appears with the current values.

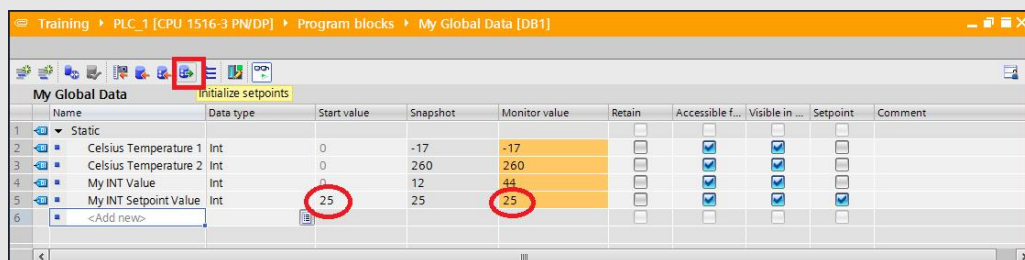
- Return to the watch table. Enter two new values for the tags and click “Modify”. Note that the values changed, but not the snapshots.



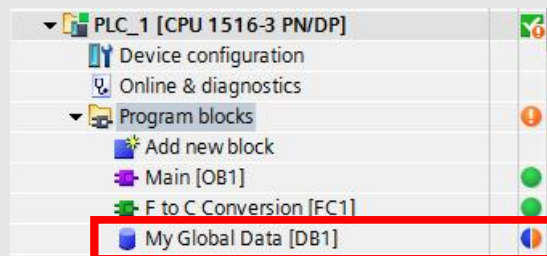
- Click the “Copy all setpoints from the “Snapshot” column to the “Start value” column” button. Note that ONLY the setpoint value was copied.



- Click the “Initialize setpoints” button. Note that the value specified as a “setpoint” now has the same Monitor value and Start value.



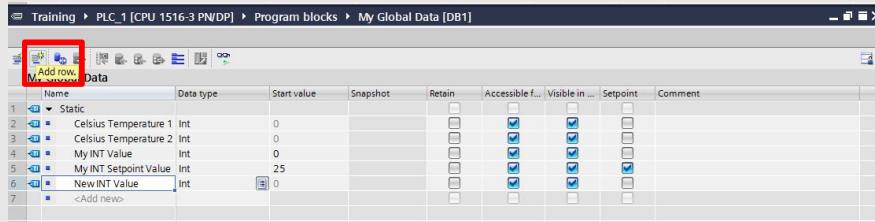
- Now that we have changed the Start value of one of our elements, the data block offline no longer matches the online version, and the software alerts you to that.



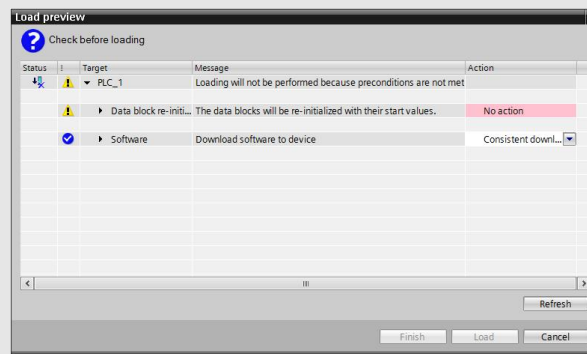
- Since no STRUCTURAL changes were made to the data block, the data block can be downloaded to the PLC without initialization. Download the data block to the PLC.

10. Exit monitor mode in the data block editor.

11. Select the last element in the data block. Add a new element to the data block using the “Add row” button. Name the new element New INT Value with a data type of INT.

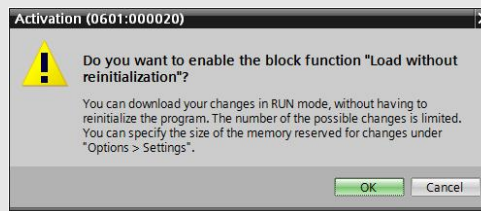


12. Download the data block. Note that since we didn't enable download without initialization and the new element was added, the data block MUST be reinitialized prior to download.

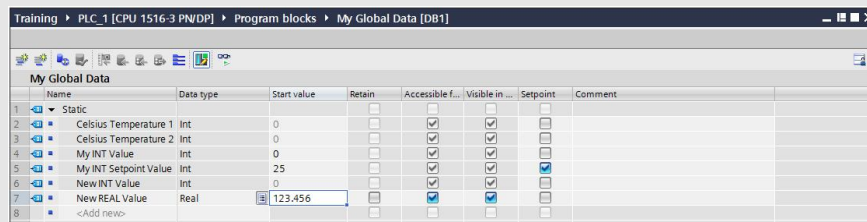


13. Accept the re-initialization and download the block. Select “Go Offline”.

14. In the data block editor, click the “Download without reinitialization” button. Acknowledge the message that appears.



15. Add another new element to the data block. Name the element “New REAL Value” and set the data type as REAL. Enter a value of 123.456 as the start value.

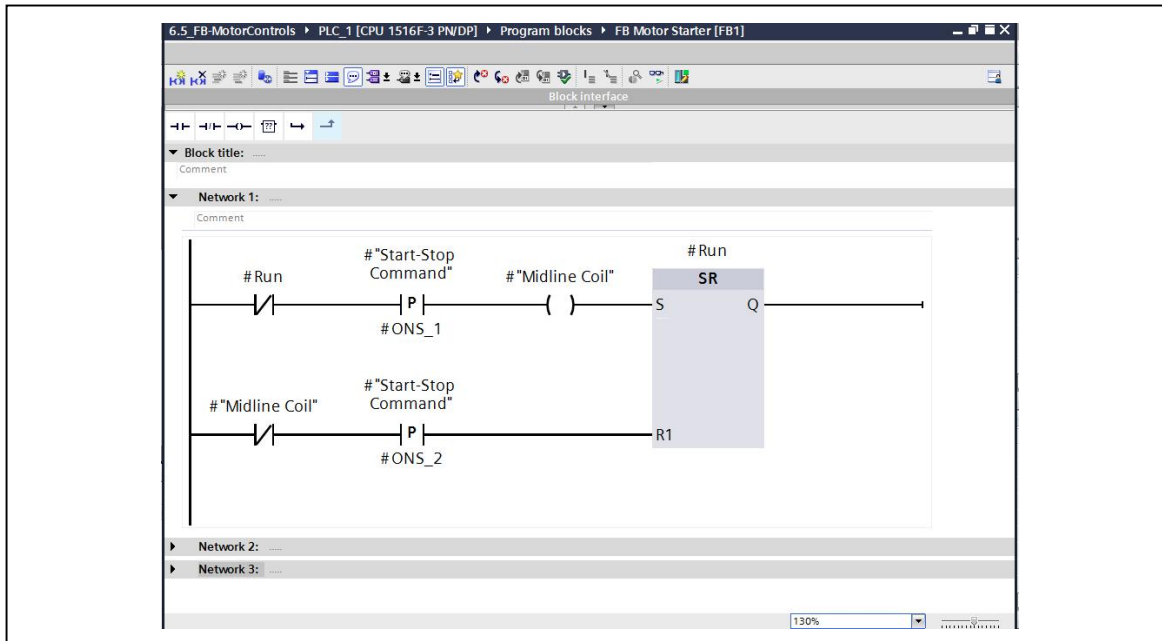


16. Download your changes. Note that the data block can now be downloaded without initialization.

17. Save your project.

This completes Exercise 3.2.

3.3 Use a Function Block and Instance Data Block



Description

Function Blocks are parts of the program whose calls can be programmed via block parameters. They have a variable memory which is located in a data block. This data block is permanently allocated to the function block, or, to be more precise, to the function block call. It is even possible to assign a different data block (with the same data structure but containing different values) to each function block call. Such a permanently assigned data block is called an instance data block, and the combination of function block call and instance data block is referred to as an instance. Function blocks can also save their variables in the instance data block of the calling function block; this is referred to as a "local instance".

In this exercise, a reusable Function Block will be created and tested to control an ON/OFF Toggle using a normally open pushbutton. Static variables within the FB will be used for edge operations.

Objectives

Upon completion of this exercise, the student shall be able to:

- Configure a Reusable Function Block.
- Use Formal Parameters for block reusability.
- Use LAD to program the Function Block.
- Call and Test the Function Block in Main [OB1]

Prerequisites

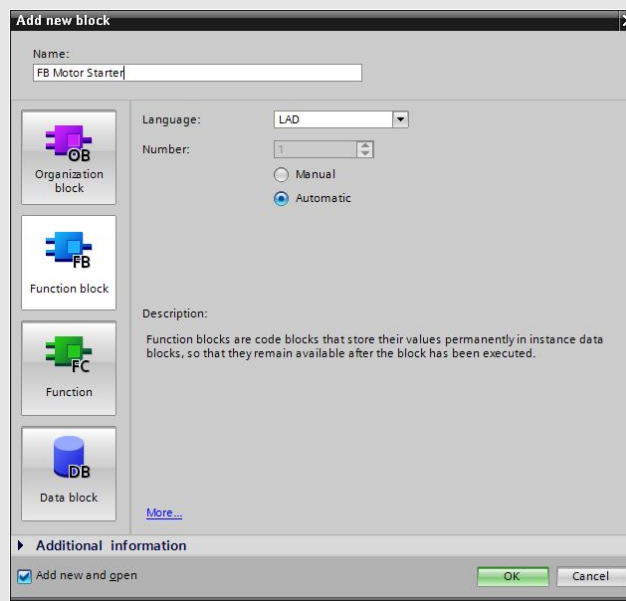
The following prerequisites must be completed before this exercise is started:

- Exercise 3.2 has been completed.

3.3.1 Insert and Parameterize FB Motor Starter.

1) Add a new FB to the program.

1. From the Project tree, double click on the “Add new block” object.
2. Select “Function Block”
3. Assign the Name “FB Motor Starter”
4. Select “LAD” for the Language.
5. Select “Add new and open”.
6. Select OK.



2) Add the Local Input, Output and Static Parameters to the declaration table.

1. Enter the following parameters to control an Output ON/OFF state with a PB wired ON.
2. Add the following elements to the FB interface:
 - Input: Start-Stop Command Data Type BOOL
 - InOut: Run Data Type BOOL
 - Static: Midline Coil Data Type BOOL
 - Static: ONS_1 Data type BOOL
 - Static: ONS_2 Data type BOOL

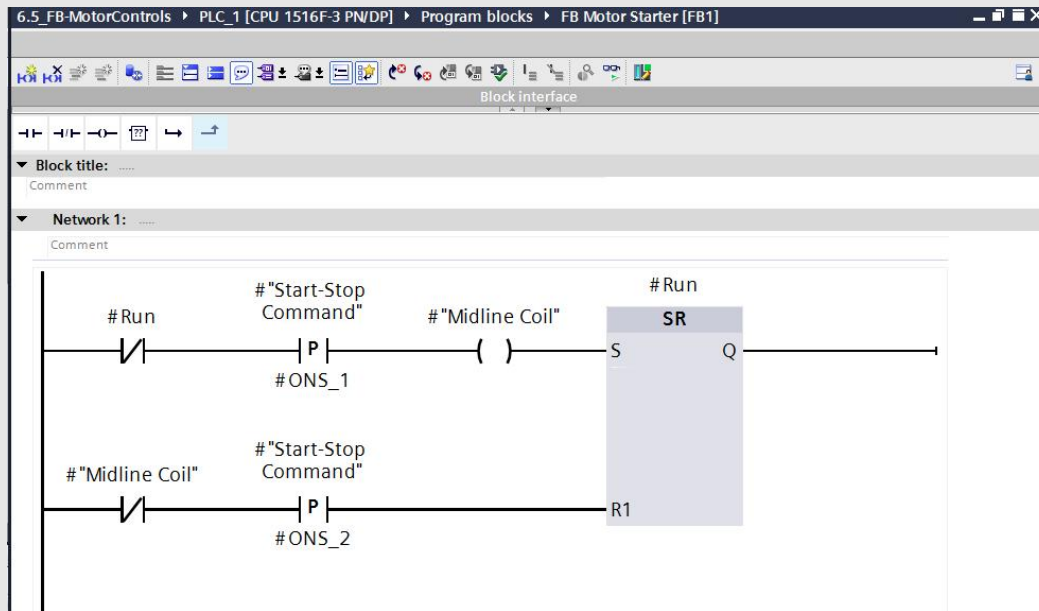
6.2_CreateGlobalDB ▸ PLC_1 [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ FB Motor Starter [FB1]

	Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Input							
2	Start-Stop Command	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4	Output							
5	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6	InOut							
7	Run	Bool	false	Non-ret...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
8	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
9	Static							
10	Midline Coil	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
11	ONS_1	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
12	ONS_2	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
13	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
14	Temp				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
15	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
16	Constant				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

✓ **NOTE:** This completes the declaration table.

3) Program the FB (in LAD) for reusable toggle functionally.

1. Program the following code to control an Output ON/OFF state with a PB wired ON.
2. Enter the following code to Network 1.

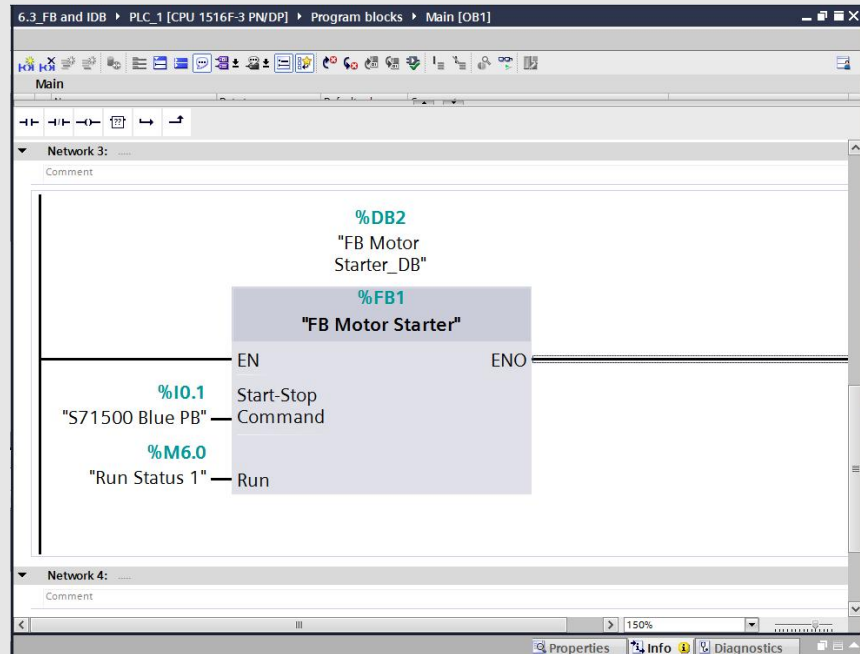


✓ **NOTE: This completes the Instruction Code area.**

3.3.2 Test Functionality.

1) Test the new FB functionality in Main [OB1].

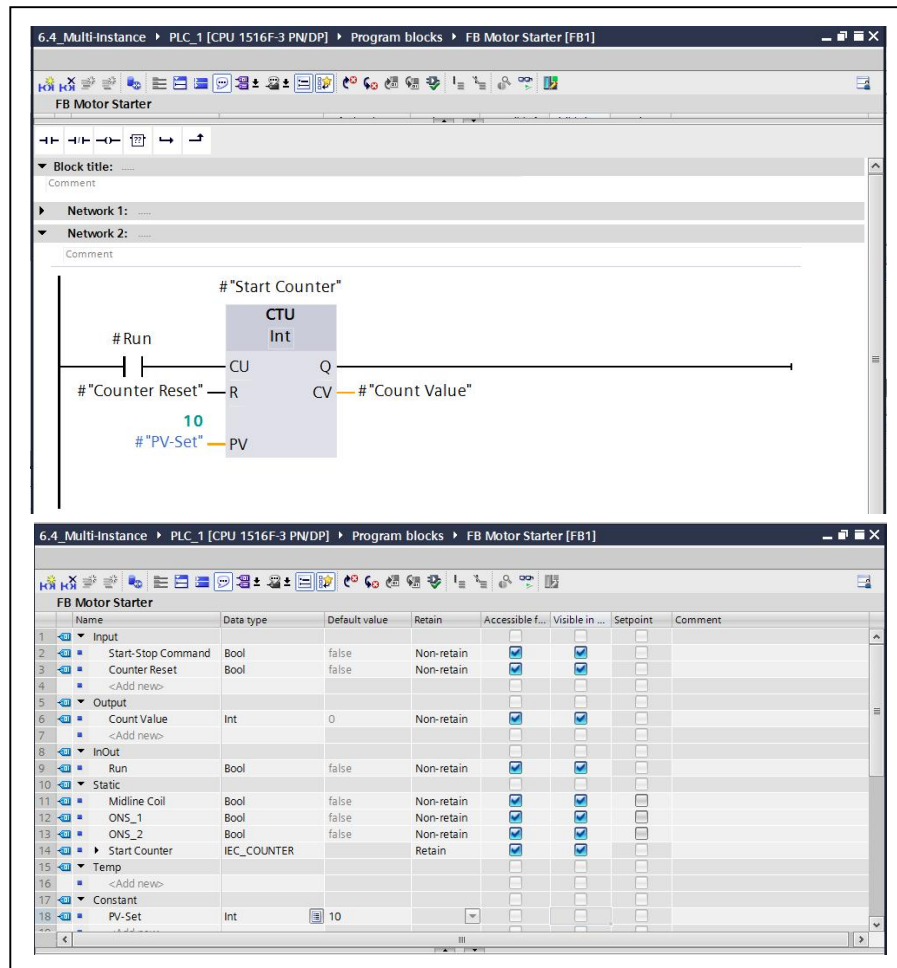
1. Open Main [OB1] for editing.
2. Insert a new Network 4 if it's not already there.
3. Call the new FB and accept the default name of the instance DB.



4. For the Start-Stop Command, use the previously created tag “S71500 Blue PB”.
 5. For the Run output, create a new Boolean tag “Run Status 1” and add it to the Global memory area in the Default tag table.
 6. From the Project tree, select the Program blocks folder, then click Compile to compile the entire program.
 7. Download the Program blocks folder and test the code.
- ✓ **NOTE: Monitor OB1, The new FB and instance DB during the test procedure for correct functionality.**

This completes Exercise 3.3.

3.4 Use an IEC Counter SFB as a Multi-Instance FB



Description

IEC Counter Functions are integrated into the operating system of the CPU as system function blocks, SFBs. Three counters are available: "CTU" Up Counter, "CTD" Down Counter, and "CTUD" Up-down Counter. You can find the interface description for offline programming in the Help file.

You can call these SFBs with an instance data block or you can use these SFBs as local instances in a function block.

In this exercise, a "CTU" from the Counting functions will be instantiated to count the number of motor starts, giving the FB "FB Motor Starter" the same functionality as the FC "Motor Start Latch". The main difference is the FB has its own dedicated memory associated with it when called in OB1.

Objectives

Upon completion of this exercise, the student shall be able to:

- Instantiate and Configure SFB Counters.
- Use LAD to program the Function Block.
- Use Formal Parameters for block reusability.
- Call and Test the Function Block in OB1.

Prerequisites

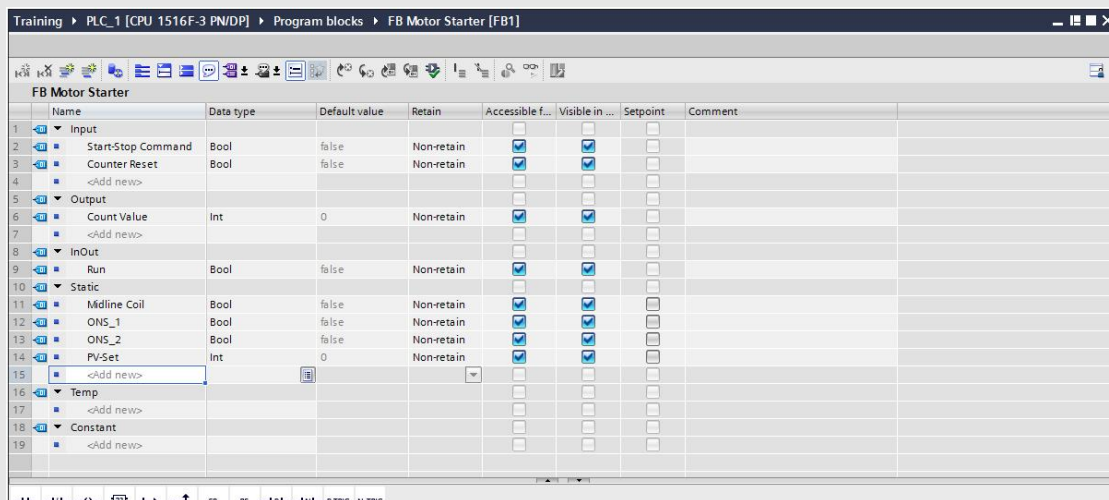
The following prerequisites must be completed before this exercise is started:

- Exercise 3.3 has been completed.

3.4.1 Modify “FB Motor Starter” for Motor Starts Counter Control

1) Modify Local Input, Output and Static to the Parameters in the declaration area.

1. Open “FB Motor Starter”.
2. Enter the following declarations to control system functionality.
 - Input: Counter Reset Data Type BOOL
 - Output: Count Value Data Type INT
 - Constant: PV-Set Data Type INT, DEFAULT value = 10

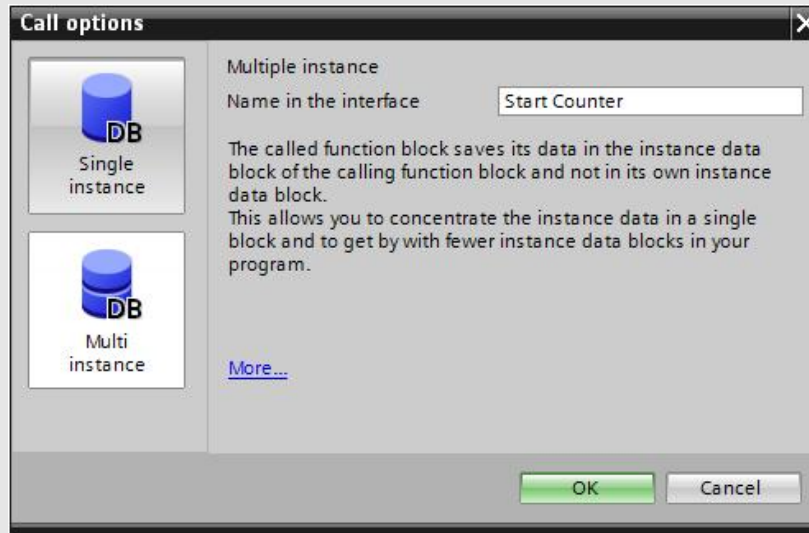


Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
Input							
Start-Stop Command	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Counter Reset	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<Add new>							
Output							
Count Value	Int	0	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<Add new>							
InOut							
Run	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Static							
Midline Coil	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
ONS_1	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
ONS_2	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
PV-Set	Int	0	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<Add new>							
Temp							
<Add new>							
Constant							
<Add new>							

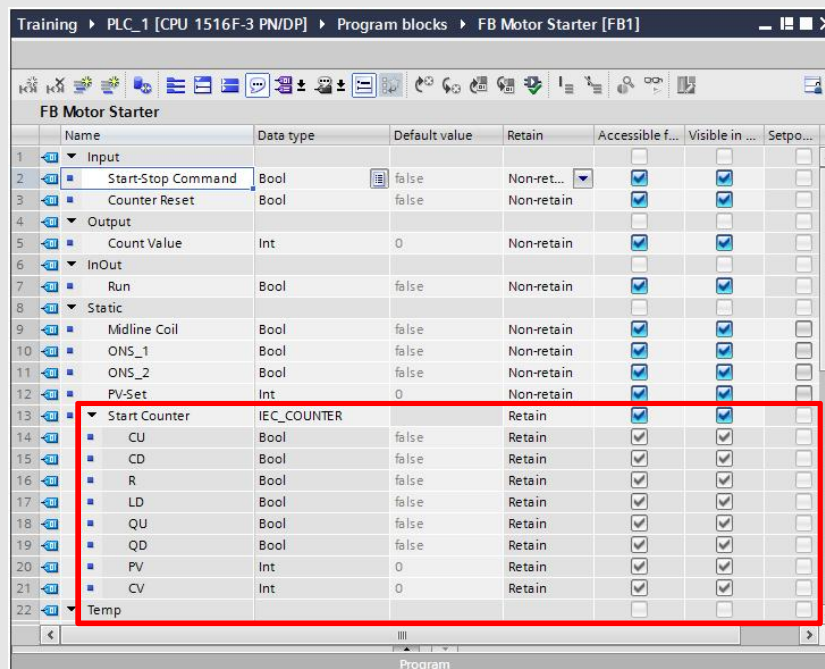
✓ **NOTE:** This completes the addition to the declaration area.

2) Instantiate CTU for Count Up control.

1. From the Instructions pane, select Basic instructions, then Counter Operations.
2. Select the instruction CTU and add it to the new Network 2.
3. When the Call options dialog appears, select "Multi-instance" and enter the name "Start Counter".

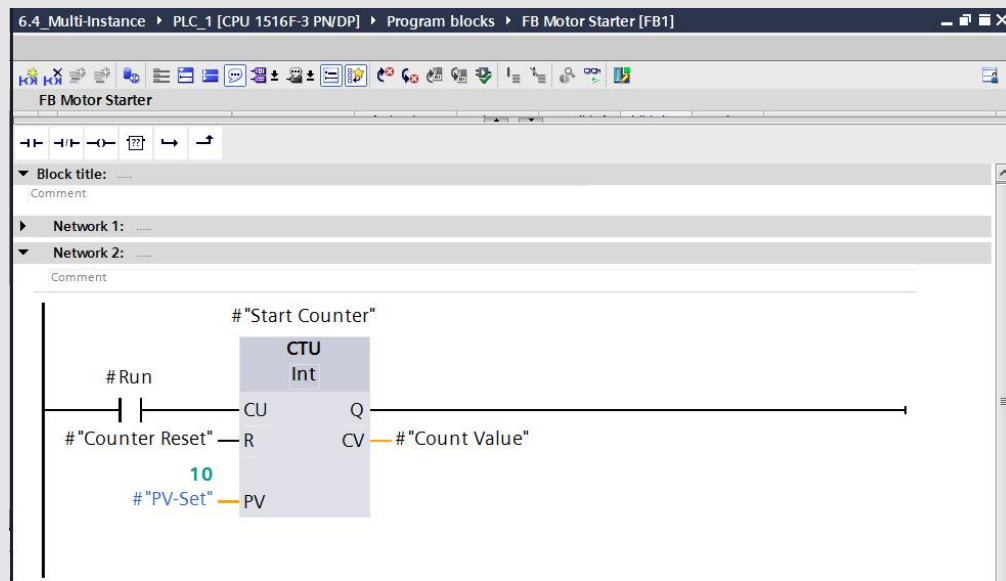


4. Select OK.
- Observe: The #Start Counter added to the Static declaration.



3) Program Network 2 for Reusability by using block parameters and local variables.

1. Add a “normally open” contact to the CU input of the counter.
2. Use #Run as the Control Bit.
3. Add #Counter Reset to the R input.
4. Add a #PV-Set to the PV input.
5. Add #Count Value to the CV Output.



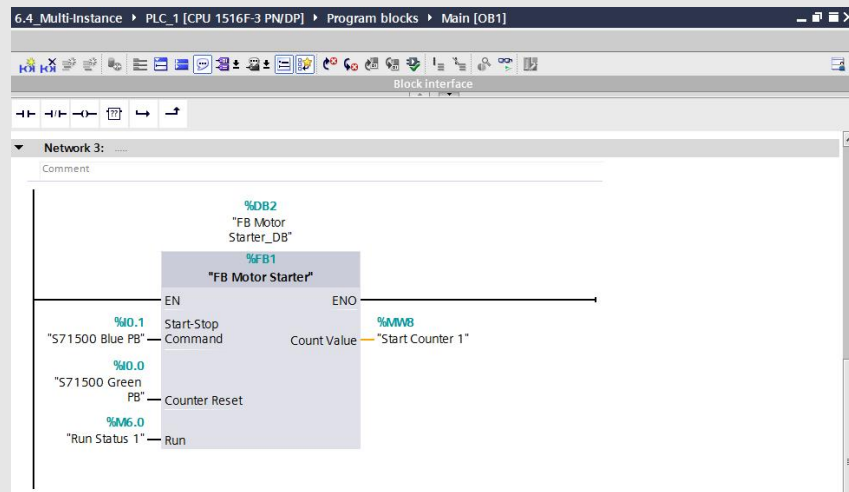
✓ **NOTE: This completes the Instruction Code area.**

6. Save the Project.
7. From the Project tree, select the “Program blocks” folder. Click the “Compile” button to recompile the entire program and update all blocks with new calls and code.

3.4.2 Test Functionality.

1) Test Functionality in OB1.

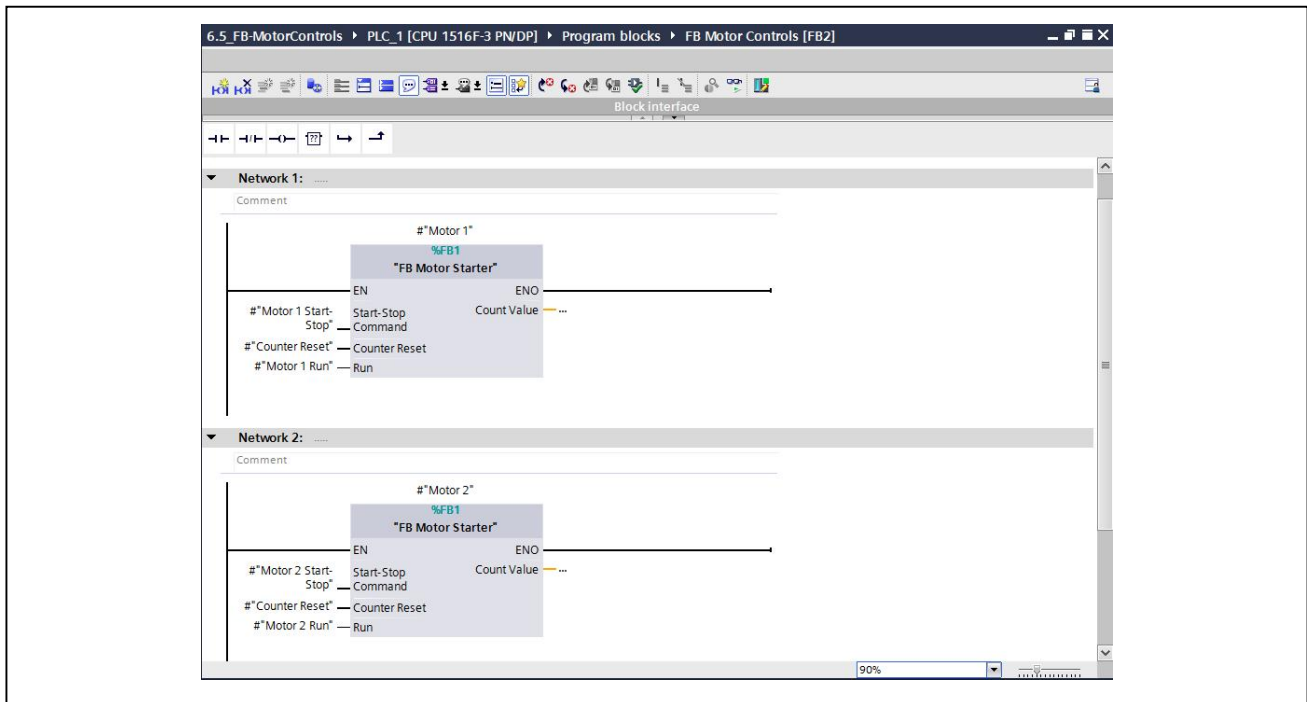
1. Open OB1 for editing
2. Select Network 4.
3. Note that the call of FB Motor Starter has been updated to reflect the changes.
4. Change Program to reflect the changes as shown below. Note the tag “Start Counter 1” will have to be created. Add the tag as an INT in the Global memory area and assign it to the Default tag table.



5. Select the Program blocks folder from the project tree, then click download.
 6. Select any blocks required to be overwritten to make the program be consistent, and then click “Load”.
 7. Test the program for functionality.
- ✓ **NOTE: Monitor Main, “FB Motor Starter”, and “FB Motor Starter_DB” during the test procedure for correct functionality.**

This completes Exercise 3.4.

3.5 Create a Multi Instance Function Block



Description

Instantiation of the previously created Function Blocks can be used to control multiple motors. Each will pass through different I/O for Motor control.

In this exercise, a new FB will be created to control Motor 1 and Motor 2. The original FB Motor Starter will be instantiated in the new "FB Motor Controls" FB. Each instance will be tested for functionality.

Objectives

Upon completion of this exercise, the student shall be able to:

- Instantiate multiple FB's into one FB.
- Test functionality while monitoring each instance of the FB Motor Starter call.

Prerequisites

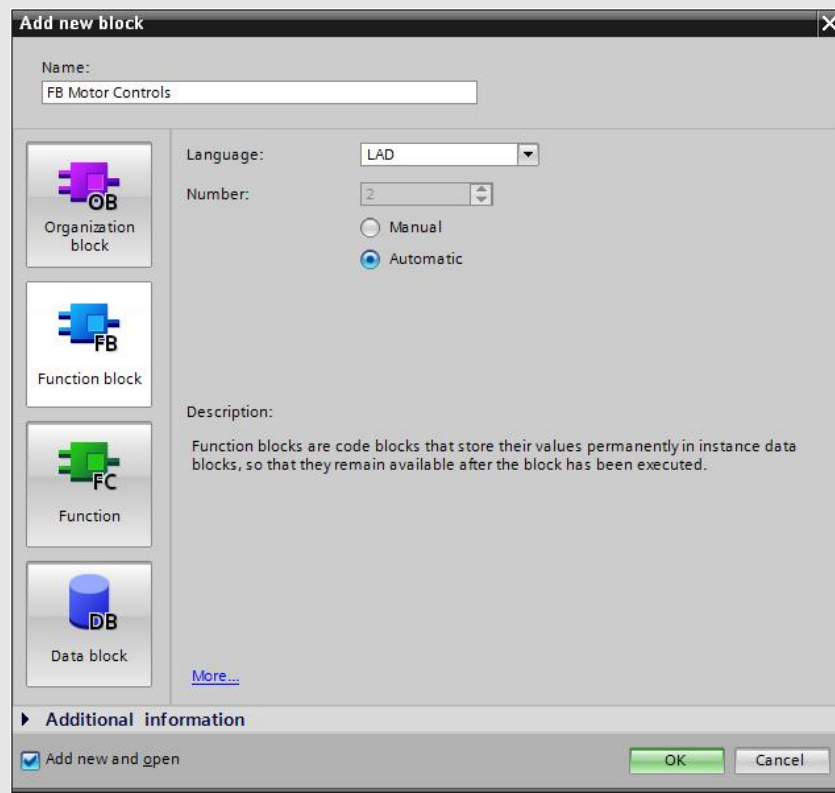
The following prerequisites must be completed before this exercise is started:

- Exercise 3.4 has been completed.

3.5.1 Create FB Motor Controls to Control Multiple Motor Starters

1) Create a new FB.

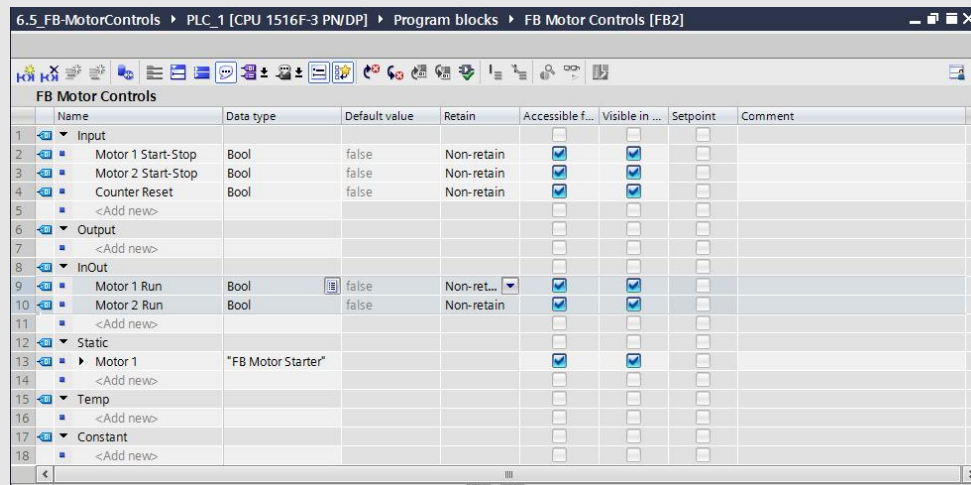
1. Double click on the “Add new block” element in the Project tree.
2. Select “Function Block”
3. Assign the Name “FB Motor Controls”
4. Select “LAD” for the Language.
5. Select “Add new and open”
6. Select OK.



2) Parameterize FB Motor Controls Input and Output declarations to control the Logic.

1. Add the following elements to the Block interface

- Input: Motor 1 Start-Stop Data Type BOOL
- Input: Motor 2 Start-Stop Data Type BOOL
- Input: Counter Reset Data Type BOOL
- InOut: Motor 1 Run Data Type BOOL
- InOut: Motor 2 Run Data Type BOOL

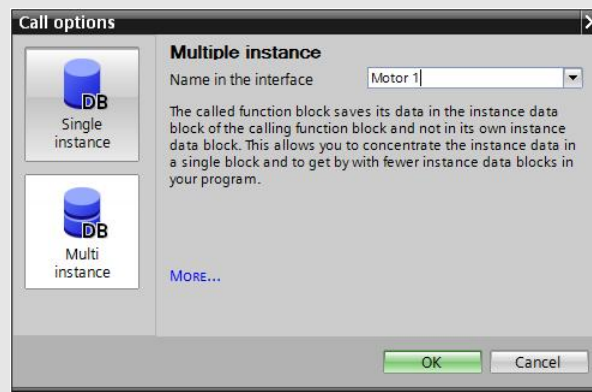


	Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Input							
2	Motor 1 Start-Stop	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	Motor 2 Start-Stop	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4	Counter Reset	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
5	<Add new>							
6	Output							
7	<Add new>							
8	InOut							
9	Motor 1 Run	Bool	false	Non-ret...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
10	Motor 2 Run	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
11	<Add new>							
12	Static							
13	Motor 1	"FB Motor Starter"			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	<Add new>							
15	Temp							
16	<Add new>							
17	Constant							
18	<Add new>							

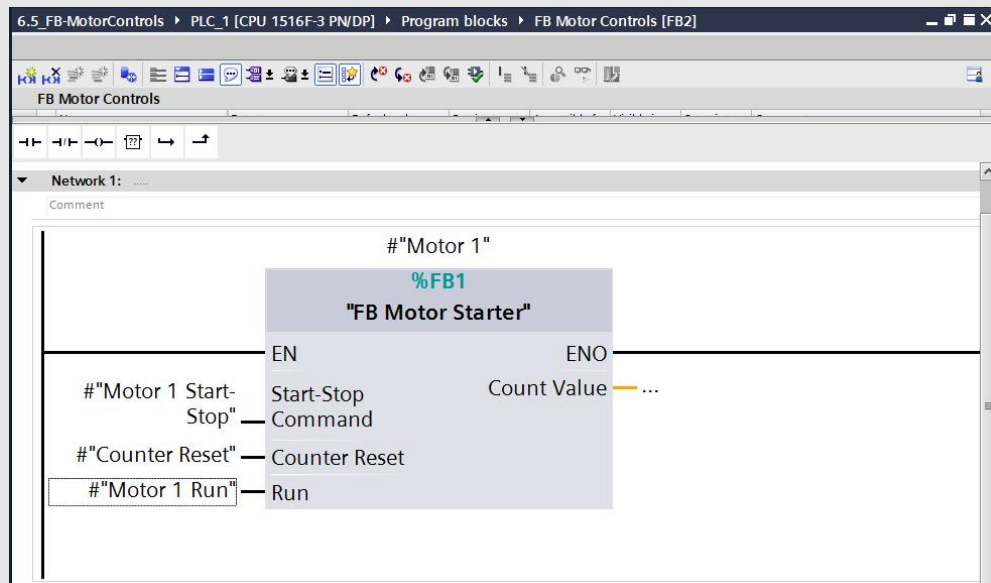
✓ **NOTE: This completes the declaration table.**

3) Instantiate FB Motor Starter into FB Motor Controls for Motor Starter 1 Control.

1. Open FB Motor Controls for editing if not already open.
2. Select Network 1.
3. Add a call to FB Motor Starter. When the Call Options dialog appears, select Multi-instance and name this instance "Motor 1" as follows:

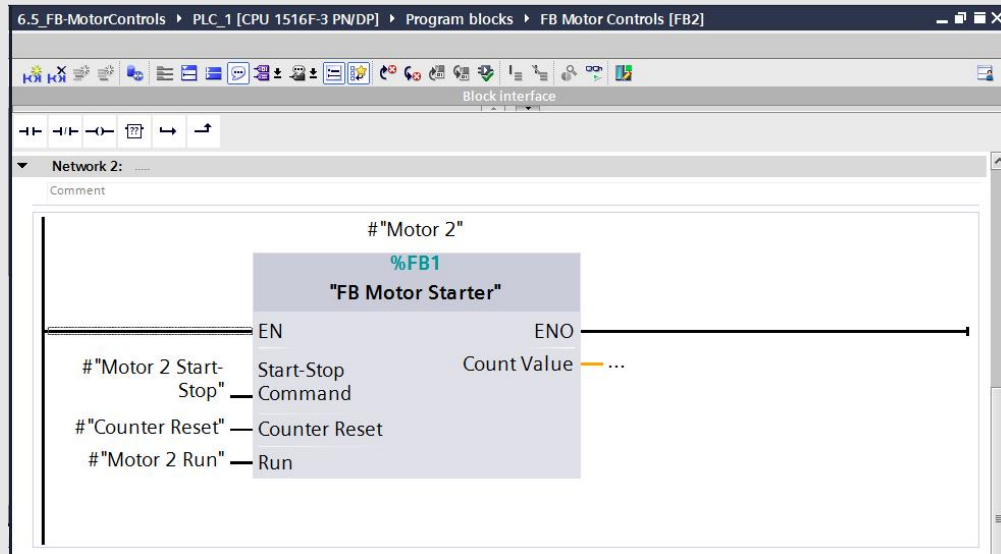


4. Program Network 1 as follows:



4) Repeat the previous steps for Network 2 for a call to Motor 2.

1. Add a new network, add the Multi-instance call of FB Motor Starter and address the block as shown below:



✓ **NOTE:** This completes the Instruction Code area.

5) Observe the Static area in FB Motor Controls.

1. Open the Block Interface for FB Motor Controls.
 2. Expand the Static Area.
- Observe: View the Static Declaration Area.

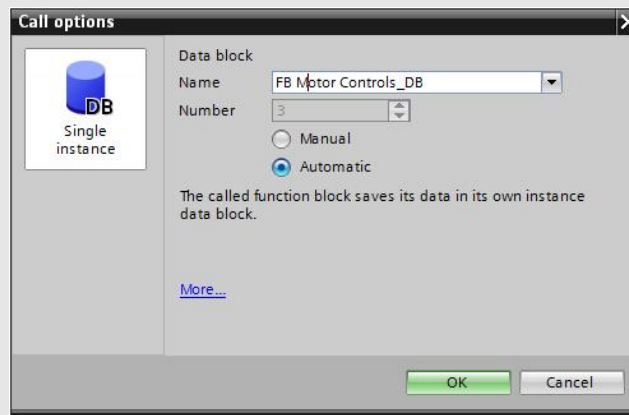
The screenshot shows the 'FB Motor Controls' block interface in the Siemens STEP 7 LAD editor. The 'Static' area is expanded, showing various inputs, outputs, and static declarations for Motor 1 and Motor 2.

	Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Input							
2	Motor 1 Start-Stop	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	Motor 2 Start-Stop	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4	Counter Reset	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
5	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6	Output							
7	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8	InOut							
9	Motor 1 Run	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
10	Motor 2 Run	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
11	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
12	Static							
13	Motor 1	"FB Motor Starter"			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	Input							
15	Start-Stop Com...	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
16	Counter Reset	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
17	Output							
18	Count Value	Int	0	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
19	InOut							
20	Run	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
21	Static							
22	Midline Coil	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
23	ONS_1	Bool	false	Non-ret...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
24	ONS_2	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
25	Start Counter	IEC_COUNTER		Retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
26	Motor 2	"FB Motor Starter"			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
27	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
28	Temp							

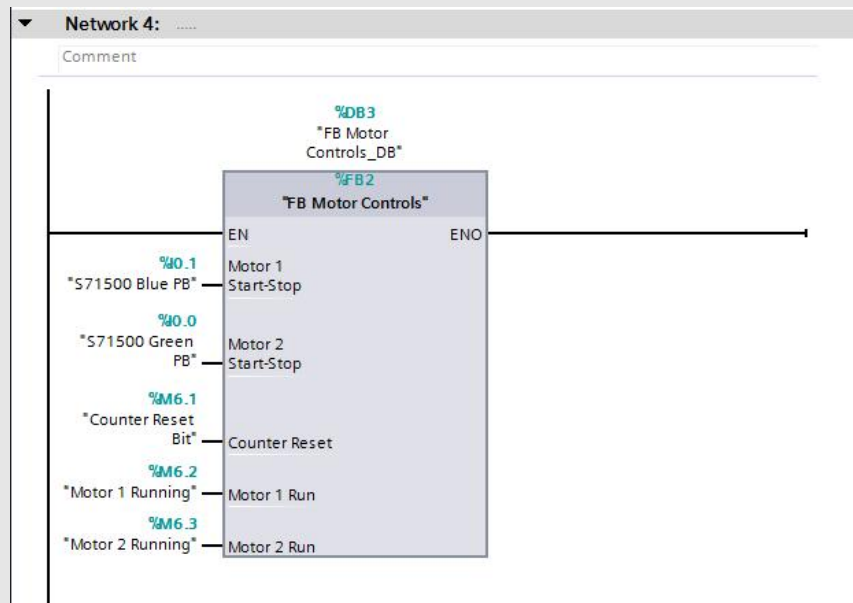
3.5.2 Complete the calls of FB Motor Controls

1) Call FB Motor Controls in OB1.

1. Open Main [OB1] for editing.
2. Delete the existing call of FB Motor Starter in Main [OB1] as we're now going to control two "motors" from a single call and the existing call is no longer required.
3. Call FB Motor Controls in Network 4. When the Call options dialog appears, accept the default name of the data block to be created and click "OK"



4. Program Network 4 as shown below. Note that the BOOL tags of "Counter Reset Bit", "Motor 1 Running", and "Motor 2 Running" are to be created as Global memory and added to the Default tag table.

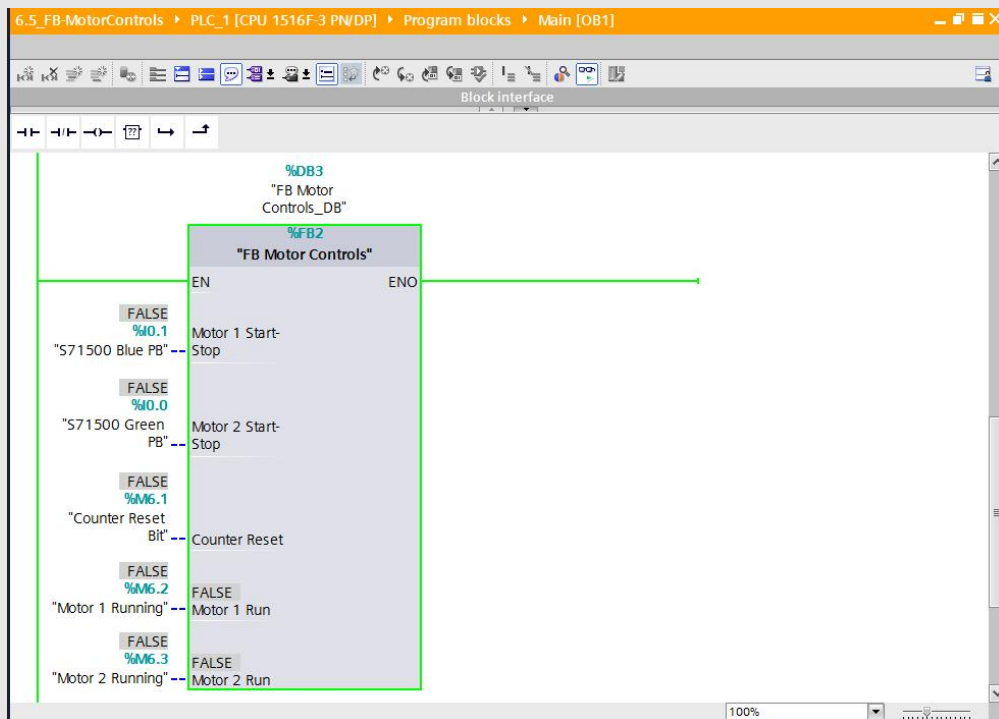


5. Select the Program blocks folder from the Project tree, then click compile.

3.5.3 Testing and Monitoring the Motor Starters

1) Monitor the Motor Starter Status from OB1.

1. Select the Program folder, then click "Download."
2. Open Main [OB1].
3. Select Network 3.
4. Monitor and Test each Motor Starter Control for correct functionality.



5. Deselect Monitor and close Main [OB1].
6. Save your project.

3.6 Create a PLC Data Type

Description

PLC Data Types allow the user to create a data template for commonly used data structures that can be used to create structures in data blocks, functions, and function blocks.

In this exercise, a new PLC Data type will be created and used to create parameters for use in the function blocks created earlier. This sets the stage for creating a similar HMI data type later, and leveraging the structure with an HMI Faceplate.

Objectives

Upon completion of this exercise, the student shall be able to:

- Create a PLC Data Type.

- Use the new Data Type in a code container.

- Test functionality while monitoring each instance of the FB Motor Starter call.

Prerequisites

The following prerequisites must be completed before this exercise is started:

- Exercise 3.5 has been completed.

3.6.1 Create a new PLC Data Type

1) Create PLC Data Type “Motor Control”

1. From the Project tree, expand the branch under PLC_1 labeled “PLC data types”.
2. Double click Add new data type
3. Right click on new data type and rename to “Motor Control”
4. Insert the required control parameters as shown below.

	Name	Data type	Default value	Accessible f...	Visible in ...	Setpoint	Co
1	Motor Start/Stop	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
2	Motor Run	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	Counter Reset	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4	Count Value	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

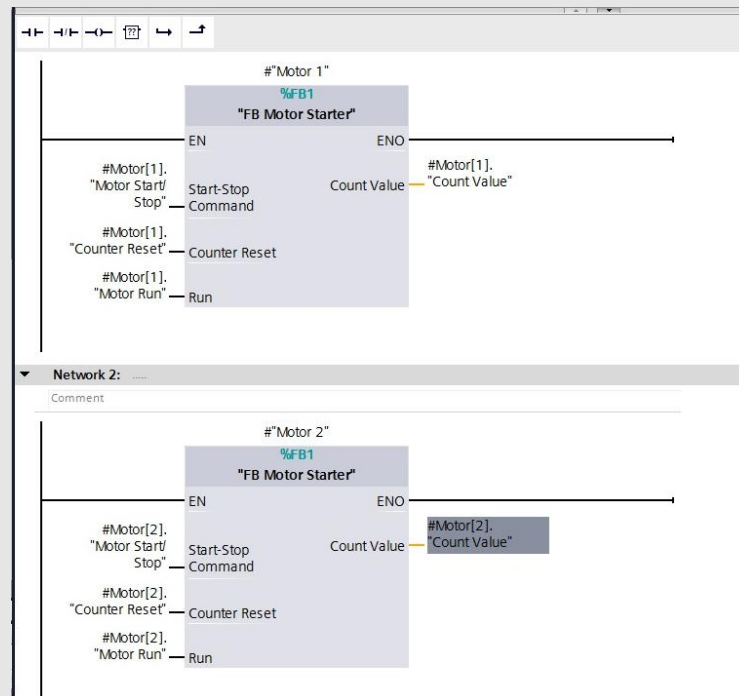
5. Save your project

2) Modify FB Motor Controls

1. Open FB Motor Controls [FB2] for editing.
2. We're going to replace the individual variables created for motor control and replace them with a PLC Data Type. DELETE the previously used Input and InOut parameters.
3. Add the new InOut parameter as an array named “Motors”. Dimension the array from 1..2 of data type “Motor Control”.

FB Motor Controls								
	Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	
1	▼ Input				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3	▼ Output				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5	▼ InOut				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6	▼ Motor	Array[1..2] of "Motor Control"			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7	▼ Motor[1]	"Motor Control"			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8	Motor Start/Stop	Bool			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
9	Motor Run	Bool			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
10	Counter Reset	Bool			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
11	Count Value	Int			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
12	▼ Motor[2]	"Motor Control"			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
13	Motor Start/Stop	Bool			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
14	Motor Run	Bool			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
15	Counter Reset	Bool			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
16	Count Value	Int			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
17	▼ Static				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
18	▶ Motor 1	"FB Motor Starter"			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
19	▶ Motor 2	"FB Motor Starter"			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
20	▼ Temp				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
21	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
22	▼ Constant				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
23	<Add new>				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

- Update the input parameters of the instances of "FB Motor Starter" as shown in the figure below.



- Save the Project

3) Create DB Motor Controls

1. Add a new global data block with the named "DB Motor Controls"
2. Add an array dimensioned from 1..2 of data type "Motor Control" See the figure below.

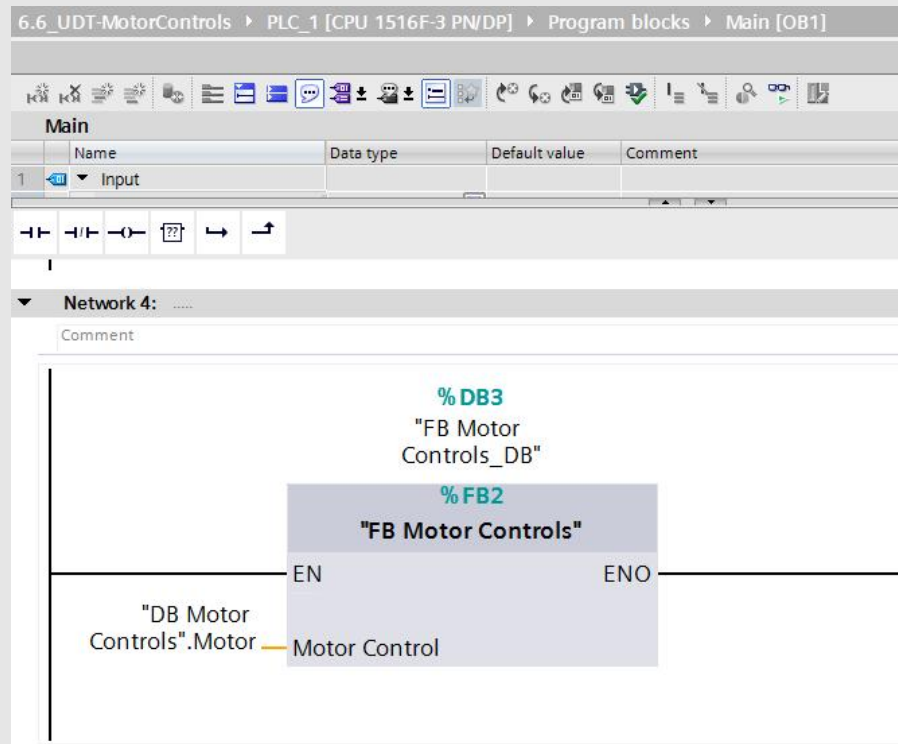
6.6_UDT-MotorControls ▸ PLC_1 [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ DB Motor Controls [DB4]

DB Motor Controls									
	Name	Data type	Start value	Retain	Accessible f...	Visible in ...	Setpoint	Co	
1	Static			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2	Motor	Array[1..2] of "Motor Control"		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Motor[1]	"Motor Control"		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
4	Motor Start/Stop	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
5	Motor Run	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
6	Counter Reset	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
7	Count Value	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
8	Motor[2]	"Motor Control"		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
9	Motor Start/Stop	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
10	Motor Run	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
11	Counter Reset	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
12	Count Value	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

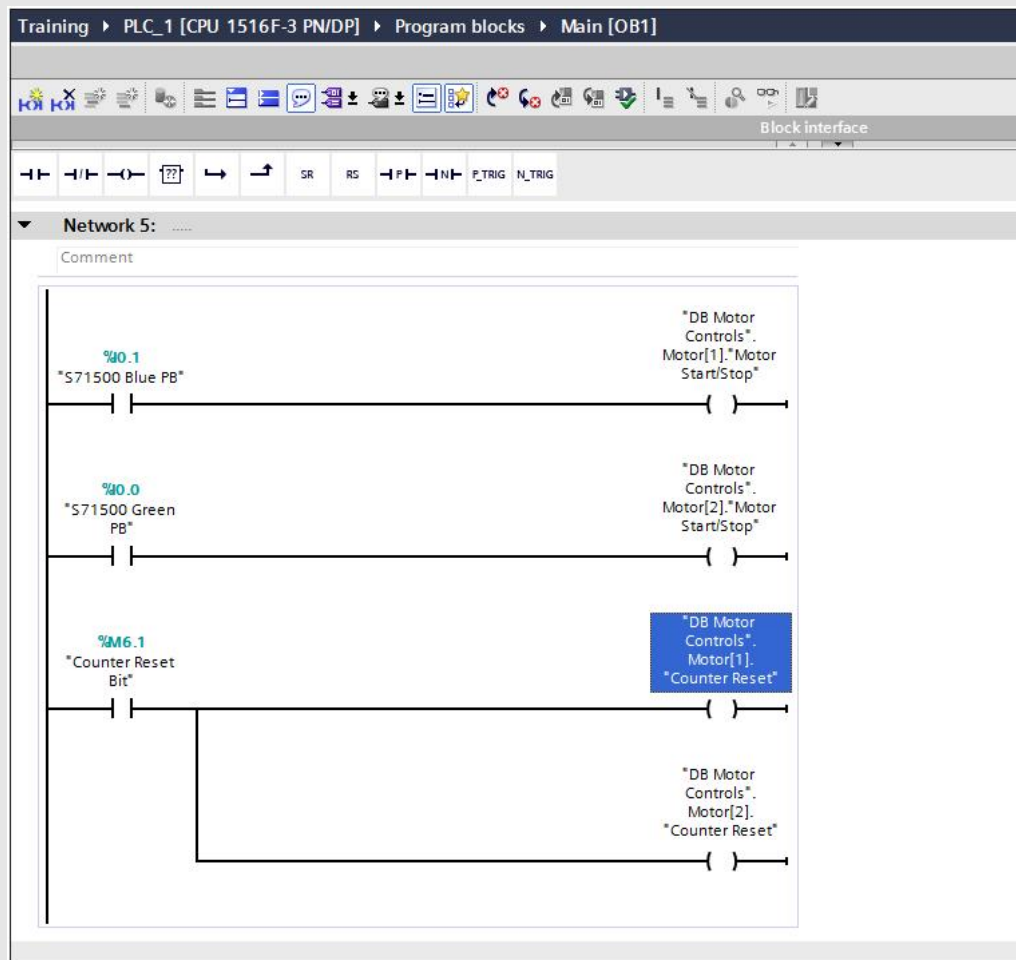
3. Update the call of FB Motor Controls inside OB1 to display the newly created parameters.
4. Attach DB Motor controls to InOut Parameters.

NOTE

When accessing the tag list for the input parameter, we want to use the entire array, not a single element. The software will try to anticipate using an individual element and will present you with square brackets. You have to edit the tag text and delete the bracket manually.



5. Insert a new network 5 in OB1. Create the logic shown below to attach addressing to DB Motor Controls.



6. Test the motor controls by monitoring DB Motor controls and pressing the two pushbuttons. Reset counter by modifying "Counter Reset Bit" Functionality should be the same as before.

This concludes the Chapter 3 Exercises.