SCE Training Curriculum
Siemens Automation Cooperates with Education | 09/2017

TIA Portal Module 052-201
High-Level Language Programming
with SCL and SIMATIC S7-1500
Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7 CPU 1516F-3 PN/DP**
  Order no.: 6ES7516-3FN00-4AB2

- **SIMATIC STEP 7 Professional V14 SP1 - Single license**
  Order no.: 6ES7822-1AA04-4YA5

- **SIMATIC STEP 7 Professional V14 SP1 - Classroom license (set of 6)**
  Order no.: 6ES7822-1BA04-4YA5

- **SIMATIC STEP 7 Professional V14 SP1 - Upgrade license (set of 6)**
  Order no.: 6ES7822-1AA04-4YE5

- **SIMATIC STEP 7 Professional V14 SP1 - Student license (set of 20)**
  Order no.: 6ES7822-1AC04-4YA5

Please note that these trainer packages are replaced with successor packages when necessary. An overview of the currently available SCE packages is provided at: [siemens.com/sce/tp](http://siemens.com/sce/tp)

**Continued training**
For regional Siemens SCE continued training, contact your regional SCE representative: [siemens.com/sce/contact](http://siemens.com/sce/contact)

**Additional information regarding SCE**
[siemens.com/sce](http://siemens.com/sce)

**Notes on use**
This SCE training curriculum for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational and R&D institutions. Siemens AG assumes no responsibility for the content.

This curriculum may be used only for initial education with respect to Siemens products/systems. That is, it may be copied in part or in whole and handed out to trainees for use within the framework of their education. Transmission and reproduction of this curriculum as well as communication of its content is permitted within public educational institutions for educational purposes. Any exceptions require written consent from Siemens AG. Contact: Roland Scheuerer [roland.scheuerer@siemens.com](mailto:roland.scheuerer@siemens.com).

Parties breaching this provision shall be liable for damages. All rights reserved, including those relating to translation and in particular those rights created as a result of a patent being granted or utility model being registered.

Use for industry customers is expressly prohibited. Commercial use of the curriculum is not permitted.

We wish to thank the TU Dresden, especially Prof. Dr.-Ing. Leon Urbas, the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this training curriculum.
Table of contents

1. Objective .......................................................................................................................................... 4
2. Requirement .................................................................................................................................... 4
3. Required hardware and software ...................................................................................................... 5
4. Theory ............................................................................................................................................. 6
   4.1 The S7-SCL programming language .........................................................................................6
   4.2 The S7-SCL development environment .....................................................................................6
5. Task ................................................................................................................................................. 9
   5.1 Example task tank volume .........................................................................................................9
   5.2 Expansion of the sample task ....................................................................................................9
6. Planning ........................................................................................................................................... 9
   6.1 Global data block “Data_Tank” ..................................................................................................9
   6.2 “Calculate_Volume” function ...................................................................................................10
   6.3 Expansion of the “Calculate_Volume” function .........................................................................10
7. Structured step-by-step instructions ................................................................................................ 11
   7.1 Retrieving an existing project ...................................................................................................11
   7.2 Saving the project under a new name ......................................................................................12
   7.3 Creating the “Data_Tank” data block .......................................................................................12
   7.4 Creating the “Calculate_Volume” function ................................................................................14
   7.5 Specifying the interface of the “Calculate_Volume” function .....................................................15
   7.6 Programming the “Calculate_Volume” function ........................................................................16
   7.7 Programming the "Main [OB1]" organization block .................................................................17
   7.8 Compiling and downloading the program ...............................................................................19
   7.9 Monitoring and testing the organization block .........................................................................20
   7.10 Expansion of the “Calculate_Volume” function .........................................................................22
   7.11 Customizing the organization block .........................................................................................27
   7.12 Compiling and downloading the program ...............................................................................28
   7.13 Monitoring and testing the organization block .........................................................................29
   7.14 Monitoring and testing the “Calculate_Volume” function .........................................................31
   7.15 Archiving the project ................................................................................................................34
8. Checklist ........................................................................................................................................ 34
9. Exercise ......................................................................................................................................... 35
   9.1 Task description – Exercise .....................................................................................................35
   9.2 Planning ..................................................................................................................................35
   9.3 Checklist – Exercise ................................................................................................................36
10. Additional information .....................................................................................................................37
HIGH-LEVEL LANGUAGE PROGRAMMING WITH S7-SCL

1. Objective

In this section, you will learn more about the basic functions of the S7-SCL high-level language. Test functions for eliminating logical programming errors will also be presented.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2. Requirement

This section is based on the hardware configuration of SIMATIC S7 CPU1516F-3 PN/DP but can also be implemented with other hardware configurations that have digital and analog input and output cards. To implement this section, you can use the following project, for example:

SCE_EN_012_101_Hardwarekonfiguration_CPU1516F.zap13

You should also have basic knowledge of high-language programming, for example, using Pascal.
3. Required hardware and software

1. Engineering station: requirements include hardware and operating system
   (for additional information, see Readme on the TIA Portal Installation DVDs)

2. SIMATIC STEP 7 Professional software in TIA Portal – as of V13

3. SIMATIC S7-1500/S7-1200/S7-300 controller, e.g. CPU 1516F-3 PN/DP –
   Firmware as of V1.6 with memory card and 16DI/16DO and 2AI/1AO

4. Ethernet connection between engineering station and controller
4. Theory

4.1 The S7-SCL programming language

S7-SCL (Structured Control Language) is a high-level, Pascal-based programming language that allows for structured programming. The language corresponds to the “Sequential Function Chart” (SFC) language specified in the standard DIN EN-61131-3 (IEC 61131-3). In addition to high-level language elements, S7-SCL also includes typical PLC elements as language elements, such as inputs, outputs, timers, bit memories, block calls, etc. S7-SCL particularly supports the STEP 7 block concept and, in addition to the statement list (STL), ladder logic (LAD) and function block diagram (FBD), S7-SCL makes it possible to program blocks that conform to the standards. This means S7-SCL supplements and expands the STEP 7 programming software with its programming languages LAD, FBD and STL.

You do not have to create every function yourself but can use precompiled blocks, such as system functions and system function blocks that are present in the CPU’s operating system.

Blocks that are programmed with S7-SCL can be mixed with STL, LAD and FBD blocks. This means that a block programmed with S7-SCL can call another block that is programmed in STL, LAD or FBD. Accordingly, S7-SCL blocks can also be called in STL, LAD and FBD programs.

The S7-SCL test functions make it possible to search for logical programming errors in an error-free compilation.

4.2 The S7-SCL development environment

With regard to the use of S7-SCL, there is a development environment that is tailored to the specific properties of both S7-SCL and STEP 7. This development environment consists of an editor/compiler and a debugger.
Editor/Compiler

The S7-SCL editor is a text editor that can be used to edit any kind of text. The central task of the S7-SCL editor is creating and editing blocks for STEP 7 programs. A basic syntax check is performed during the input which makes it easier to avoid errors during programming. Syntax errors are displayed in different colors.

The editor offers the following options:

- Programming of an S7 block in the language S7-SCL.
- Convenient adding of language elements and block calls with drag & drop.
- Direct syntax check during programming.
- Customization of the editor to meet your needs, e.g. colors for the different language elements according to syntax.
- Checking of the finished block through compiling.
- Display of all errors and warnings that occur during compiling.
- Localization of error locations in the block, optionally with error description and information on troubleshooting.
Debugger

The S7-SCL debugger enables you to check a program while it is running in the automation system (AS) and thus find any potential logical errors.

S7-SCL provides two different test modes:
- Continuous monitoring
- Step-by-step monitoring

With "Continuous monitoring" you can test a group of instructions within a block. During the test, the values of the tags and parameters are displayed in chronological order and—if possible—updated cyclically.

With "Step-by-step monitoring" the logical program sequence is followed. You can execute the program algorithm instruction-by-instruction and observe how the contents of the processed tags change in a result window.

The type of CPU you are using determines whether or not you can use "Step-by-step monitoring". The CPU must support the use of breakpoints. The CPU used in this document does not support breakpoints.
5. Task

5.1 Example task tank volume
In the first part, you are to program the calculation of the tank volume.

5.2 Expansion of the sample task
In the second part, the task is expanded and you are programming an error evaluation.

6. Planning

The tank is in the shape of a vertical cylinder. The filling level of the volume is measured with an analog sensor. For the first test, the filling level value shall be available in standardized form (in meters).

Global parameters, such as the diameter and the height of the tank, shall be stored in a "Data_Tank" data block.

The program for calculation of the content shall be written in a "Calculate_Volume" function and the parameters shall use the unit 'meter' or 'liter'.

6.1 Global data block "Data_Tank"

The global parameters are stored in multiple structures in a global data block.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Start value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimensions</td>
<td>STRUCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>REAL</td>
<td>12.0</td>
<td>in meter</td>
</tr>
<tr>
<td>diameter</td>
<td>REAL</td>
<td>3.5</td>
<td>in meter</td>
</tr>
<tr>
<td>measured_data</td>
<td>STRUCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>filling_level_per</td>
<td>INT</td>
<td>0</td>
<td>range 0...27648</td>
</tr>
<tr>
<td>filling_level_scal</td>
<td>REAL</td>
<td>0.0</td>
<td>range 0...12.0</td>
</tr>
<tr>
<td>volume_liquid</td>
<td>REAL</td>
<td>0.0</td>
<td>in liter</td>
</tr>
<tr>
<td>fault_flags</td>
<td>STRUCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>calculate_volume</td>
<td>BOOL</td>
<td></td>
<td>fault == true</td>
</tr>
</tbody>
</table>

Table 1: Parameters in the "Data_Tank" data block
6.2 "Calculate_Volume" function

This block calculates the content of the tank in liters.

In the first step, the transferred parameters are not to be checked for reasonableness.

The following parameters are required for this step:

<table>
<thead>
<tr>
<th>Input</th>
<th>Data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter</td>
<td>REAL</td>
<td>diameter cylindric tank in meter</td>
</tr>
<tr>
<td>Filling_level</td>
<td>REAL</td>
<td>filling level of liquid in meter</td>
</tr>
</tbody>
</table>

Output

| Volume         | REAL      | volume of liquid in the tank in liter|

Table 2: Parameters for "Calculate_Volume" function in the first step

The formula for calculating the volume of a vertical cylinder is used to solve the task. The conversion factor 1000 is used to calculate the result in liters.

\[ V = \frac{d^2 \pi \cdot h}{4} \Rightarrow V = \frac{\text{Diameter}^2 \cdot 3.14159 \cdot \text{Filling_level} \cdot 1000}{4} \]

6.3 Expansion of the "Calculate_Volume" function

The second step checks whether the diameter is greater than zero. You also want to check whether the filling level is greater than or equal to zero or less than or greater than the height of the tank. In case of an error, the new parameter "er" is set to TRUE, and the parameter "Volume" is set to the value -1.

To do so, expand the interface by the parameters "er" and "Height".

<table>
<thead>
<tr>
<th>Input</th>
<th>Data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>REAL</td>
<td>height cylindric tank in meter</td>
</tr>
<tr>
<td>Diameter</td>
<td>REAL</td>
<td>diameter cylindric tank in meter</td>
</tr>
<tr>
<td>Filling_level</td>
<td>REAL</td>
<td>filling level of liquid in meter</td>
</tr>
</tbody>
</table>

Output

| er             | BOOL      | fault flag; fault == true           |
| Volume         | REAL      | volume of liquid in the tank in liter|

Table 3: Parameters for "Calculate_Volume" function in the second step
7. Structured step-by-step instructions

You can find instructions on how to implement the planning below. If you already have a good understanding of everything, it is sufficient to focus on the numbered steps. Otherwise, simply follow the steps of the instructions explained below.

7.1 Retrieving an existing project

→ Before you can start programming, you need a project with a hardware configuration. (e.g. SCE_EN_012-101_Hardware_configuration_S7-1516F_....zap).

To retrieve an existing project, you must select the respective archive from the Project view under →Project →Retrieve. Confirm your selection with "Open".

( → Project → Retrieve → Selection of a .zap archive → Open)

→ Next you can select the target directory to which you want to save the retrieved project. Confirm your selection with "OK".

( → Project → Save as... → OK )
7.2 Saving the project under a new name

You save the opened project under the name 052-201_Startup_SCL.

( → Project → Save as → 052-201_Startup_SCL → Save )

7.3 Creating the "Data_Tank" data block

In the Project view, go to → Program blocks and create a new block with a double-click on → Add new block.
Now select a data block and enter the name.

(→ → "Data_Tank" → OK)
Now you enter the names of the tags listed below with data type, start value and comment.

7.4 Creating the "Calculate_Volume" function

Now you add a function, enter the name and select the language.

( → Add new block → "Calculate_Volume" → SCL → OK )
7.5 Specifying the interface of the "Calculate_Volume" function

→ The top section of your programming view shows the interface description of your function.

→ Create the following input and output parameters.

( → Name → Data type → Comment )
7.6 Programming the "Calculate_Volume" function

→ Enter the program shown below.

( → Enter program )

```
052-201_Startup_SCL_S7-1500 → CPU_1516F [CPU 1516F-3 PN/DP] → Program blocks → Calculate_Volume [FC1]

Calculate_Volume

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Default value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter</td>
<td>Real</td>
<td></td>
<td>diameter of cylindrical tank in meter</td>
</tr>
<tr>
<td>Filling_level</td>
<td>Real</td>
<td></td>
<td>filling level of liquid in meter</td>
</tr>
<tr>
<td>Volume</td>
<td>Real</td>
<td></td>
<td>volume of liquid in the tank in liter</td>
</tr>
</tbody>
</table>

IF

1. $Volume := \pi \times \frac{Diameter}{4} \times 3.14159 \times Filling_level \times 1000;
```

→ Now compile your program and check it for syntax errors. These are displayed in the Inspector window below the programming. Correct any errors and compile the program again. Then save your program.

( → → Eliminate errors → Save project )
7.7 Programming the "Main [OB1]" organization block

→ Before programming the "Main [OB1]" organization block, switch the programming language to FBD. To do so, left-click "Main [OB1]" in the "Program blocks" folder.

( → CPU_1516F[CPU 1516F-3 PN/DP] → Program blocks → Main [OB1] → Switch programming language → FBD )

→ Now double-click the "Main [OB1]" organization block to open it.
→ Call the "Calculate_Volume" function in the first network. Assign network title, comment and connect the parameters. Then save your project.

( → Call "Calculate_Volume" → Assign network title → Write network comment → Connect parameters → Save project )
7.8 Compiling and downloading the program

→ Click the "Program blocks" folder and compile the entire program. After successful compilation, download the project to the PLC.

( ▶ ▹ ▶ )

→ Select PG/PC interface → Select subnet → Start search → Load
→ Make selection, if necessary → Load

→ Finish

7.9 Monitoring and testing the organization block

→ In the open OB1 click the icon to monitor the block.
Test your program by writing a value to the "Filling_level_scal" tag in the data block.

( → Right-click on "Filling_level_scal" → "Modify" menu → Modify operand )

→ Enter value 6.0 → OK
Check the result for correctness.

7.10 Expansion of the "Calculate_Volume" function

Open the "Calculate_Volume" function, and insert a row in the output parameters with a right-click on the row in the interface.

(→ Open "Calculate_Volume" → Right-click on row 5 → Insert row )
→ Enter the parameter "er" with data type BOOL and comment.

→ Follow the same steps to add the "Height" tag with data type Real and comment.

→ Then go to the "IF...THEN...ELSE" control statement from the "Program control operations" of basic instructions.

( → Instructions → Basic instructions → Program control operations → "IF...THEN...ELSE")
→ Then drag the "IF...THEN...ELSE" control statement to the second row of the program.

("IF...THEN...ELSE" → drag & drop)
→ Highlight the mathematical formula and drag it to the semicolon in front of the ELSE.

(→ highlight → drag & drop )

→ Complete the function and check your program by compiling it.

(→ Complete program → )
Comments can be added with "(**)" as block comment and with "//" as row comment.

You can now complete your program with comments.

( Add block comment starting with row 1
→ Add row comments in rows 12 and 16 )
7.11 Customizing the organization block

→ Open OB1 and update the inconsistent block calls by clicking 

( → Open OB1 → )
7.12 Compiling and downloading the program

To do so, add the parameters "er" and "Height".

Click the "Program blocks" folder and compile the entire program. After successful compilation, download the project to the PLC. Then save your project.
7.13 Monitoring and testing the organization block

→ In the open OB1 click the icon to monitor the block.

→ Test your program by writing a value to the "Filling_level_scal" tag in the data block.

( → Right-click on "Filling_level_scal" → "Modify" menu → Modify operand → Enter value 6.0 → OK → Check )
Now test if an error is output by setting the diameter to zero.
( → Right-click on “Diameter” → “Modify” menu → Modify operand → Enter value 0.0 → OK → Check )
7.14 Monitoring and testing the "Calculate_Volume" function

Finally, open and monitor the "Calculate_Volume" function with a right-click on the function and selection of the "Open and monitor" menu command.

(→ Right-click on function → Open und monitor)
You can show the values of the individual tags of the IF query with a click on the black arrow:

<table>
<thead>
<tr>
<th>Result</th>
<th>FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Diameter</td>
<td>0.000000E+000</td>
</tr>
<tr>
<td>#Filling_level</td>
<td>6.000000E+000</td>
</tr>
<tr>
<td>#Filling_level</td>
<td>6.000000E+000</td>
</tr>
<tr>
<td>#Height</td>
<td>1.200000E+001</td>
</tr>
<tr>
<td>#or</td>
<td>TRUE</td>
</tr>
<tr>
<td>#Volume</td>
<td>-1.000000E+000</td>
</tr>
</tbody>
</table>
→ Right-click the tag to adjust the display format.
( → Right-click tag → Display format → Floating point )

<table>
<thead>
<tr>
<th>#Result</th>
<th>FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Diameter</td>
<td>0.000000E+000</td>
</tr>
<tr>
<td>#Filling_level</td>
<td>6.000000E+000</td>
</tr>
<tr>
<td>#Filling_level</td>
<td>6.000000E+000</td>
</tr>
<tr>
<td>#Height</td>
<td>1.200000E+001</td>
</tr>
<tr>
<td>#er</td>
<td>TRUE</td>
</tr>
<tr>
<td>#Volume</td>
<td>-1.000000E+000</td>
</tr>
</tbody>
</table>

→ Now test the other branch of the IF branch by modifying the diameter in OB1 back to 3.5 meters.
( → Open OB1 → Modify diameter to 3.5 → Open and monitor function )

![Image of OB1 code and display format settings](image-url)
7.15 Archiving the project

Finally, you want to archive the complete project. Select ‘Project’ → ‘Archive …’ in the menu. Open the folder in which you want to archive your project and save it as file type ‘TIA Portal Project archives’.

8. Checklist

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Successful compilation without error message</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Successful download without error message</td>
<td></td>
</tr>
</tbody>
</table>
| 3   | Modify operand (Diameter = 0.0)  
Result tag Volume = -1  
Result tag "er" = TRUE |        |
| 4   | Modify operand (Diameter = 3.5 and Filling_level_scal = 0)  
Result Volume = 0  
Result tag "er" = FALSE |        |
| 5   | Modify operand (Filling_level_scal = 6.0)  
Result Volume = 57726.72  
Result tag "er" = FALSE |        |
| 6   | Modify operand (Filling_level_scal = 12.0)  
Result Volume = 115453.4  
Result tag "er" = FALSE |        |
| 7   | Modify operand (Filling_level_scal = 14.0)  
Result Volume = -1  
Result tag "er" = TRUE |        |
| 8   | Project successfully archived |        |
9. Exercise

9.1 Task description – Exercise

In this exercise you are going to program a "Scaling" function. The program shall be generally applicable to any positive analog value. In our example task "Tank", the filling level is read by an analog sensor and stored as scaled value in the data block with this function.

In case of an error, the block shall set the error flag "er" to TRUE and set the parameter "Analog_scal" to zero as a result. An error exists when the "mx" parameter is less than or equal to "mn".

The function must contain the following parameters.

<table>
<thead>
<tr>
<th>Input</th>
<th>Data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog_per</td>
<td>INT</td>
<td>Analog value of the IO between 0..27648</td>
</tr>
<tr>
<td>mx</td>
<td>REAL</td>
<td>Maximum of the new scale</td>
</tr>
<tr>
<td>mn</td>
<td>REAL</td>
<td>Minimum of the new scale</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>er</td>
<td>BOOL</td>
<td>Error flag, no error = 0, error = 1</td>
</tr>
<tr>
<td>Analog_scal</td>
<td>REAL</td>
<td>Analog value scaled between mn..mx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In case of an error = 0</td>
</tr>
</tbody>
</table>

The following formula is used to solve the task:

\[
\text{Analog}_{\text{scal}} = \frac{\text{Analog}_{\text{per}}}{27648} \cdot (\text{mx} - \text{mn}) + \text{mn}
\]

An analog signal is required for this task. The operand used for this task must be entered in the PLC tag table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>INT</td>
<td>%IW64</td>
<td>Filling level between 0..27648</td>
</tr>
</tbody>
</table>

9.2 Planning

Now solve this task on your own.
### 9.3 Checklist – Exercise

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Operand added to PLC tag table</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Function FC: &quot;Scaling&quot; created</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Interface defined</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Function programmed</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>&quot;Scaling&quot; function added to network 1 of OB1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Input tags connected</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Output tags connected</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Successful compilation without error message</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Successful download without error message</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Analog value for filling level set to zero</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result Filling_level_scal = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result er = FALSE</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Analog value for filling level set to 27648</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result Filling_level_scal = 12.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result er = FALSE</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Analog value for filling level set to 13824</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result Filling_level_scal = 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result er = FALSE</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Modify operand (mx = 0.0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result Filling_level_scal = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result tag &quot;er&quot; = TRUE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Project successfully archived</td>
<td></td>
</tr>
</tbody>
</table>
10. Additional information

Additional information for more details is available as orientation help, such as Getting Started, Videos, Tutorials, Apps, Manuals, Programming Guidelines and Trial Software/Firmware, at the following link:

www.siemens.com/sce/s7-1500