SCE Training Curriculum
for Integrated Automation Solutions
Totally Integrated Automation (TIA)

Siemens Automation Cooperates with Education

TIA Portal Module 010-090
Startup – High Level Language Programming
with S7-SCL and SIMATIC S7-1200
Matching SCE Training Packages for these Manuals

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
  Order No.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
  Order No.: 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
  Order No.: 6ES7822-0AA01-4YE0

Please note that these training packages are replaced with successor packages when necessary. An overview of the currently available SCE packages is provided under: siemens.com/sce/tp

Continued Training
For regional Siemens SCE continued training, please contact your regional SCE contact person siemens.com/sce/contact

Additional information regarding SCE
siemens.com/sce

Information regarding Usage

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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this training curriculum and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the training curriculum being used commercially.

We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this training curriculum.
1. Preface ................................................................................................................................................. 4
2. Instructions regarding the Programming Language S7 SCL ................................................................. 6
3. S7-SCL Development Environment ........................................................................................................ 7
4. Sample Task Tank Content ...................................................................................................................... 8
   4.1 Task Description ................................................................................................................................ 8
   4.2 Assignment List/Tag Table ................................................................................................................... 8
   4.3 Program Structure ............................................................................................................................... 9
   4.4 Interface of the Block calculate_volumeTank [FC140] ....................................................................... 10
   4.5 Note on the solution ........................................................................................................................... 11
5. Programming the Tank Content Calculation for the SIMATIC S7-1200 in S7-SCL ............................. 12
   5.1 Setting Up the Project and Configuring the Hardware ........................................................................ 12
   5.2 Setting Up the Program ..................................................................................................................... 15
   5.3 Testing the Program .......................................................................................................................... 23
   5.4 Expanding the Program ...................................................................................................................... 26
1. **Preface**

Regarding its content, module 010-090 is part of the training unit 'Basics of PLC Programming' and provides a quick entry point for programming the SIMATIC S7-1200 using the programming language S7 SCL with the TIA portal.

**Training Objective**

In this module 010-090, the reader is introduced to the basic functions of the S7 SCL development environment. In addition, test functions for the removal of logical programming errors are shown.

**Prerequisites**

To successfully work through this module 010-090, the following knowledge is assumed:

- How to handle Windows
- Basics of PLC programming with STEP7 Professional V11
  (for example, Modules 010-010 to 010-040)
- Basic knowledge of high level language programming such as Pascal
Required hardware and software

1 PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
   Operating system Windows XP Professional SP3/Windows 7 Professional/Windows 7 Enterprise/
   Premium SP1, Business SP1, Ultimate SP1
2 Software STEP7 Professional V11 SP2 (Totally Integrated Automation (TIA) Portal V11)
3 Ethernet connection between PC and CPU 1214C
4 PLC SIMATIC S7-1200 starting with Firmware V2.2; for example, CPU 1214C with 14DI/10DO
   signal module.
   The inputs have to be brought out to a panel.
2. **Instructions regarding the Programming Language S7 SCL**

S7 SCL (Structured Control Language) is a higher level programming language that is based on PASCAL and makes structured programming possible. The language corresponds to the sequential language SFC “Sequential Function Chart” 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 such as inputs, outputs, timers, flags, block calls, etc. as language elements. It supports the STEP 7 block concept and in addition to STL, LAD and FBD- allows for standard-conforming programming of blocks. That means, S7-SCL supplements and expands the programming software STEP 7 with its programming languages LAD, FBD and STL.

You don’t have to generate each function yourself; preassembled blocks such as system functions or system function blocks are provided that are available in the CPU’s operating system.

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

The S7-SCL test functions make possible the search for logical programming errors in a faultless compilation.
3. S7-SCL Development Environment

To use S7-SCL, a development environment is provided that is harmonized with specific S7-SCL characteristics as well as with STEP 7. This development environment consists of an Editor/Compiler and a Debugger.

![Diagram showing Editor/Compiler and Debugger](image)

**Editor/Compiler**

The S7-SCL editor is a text editor that can be used to edit any text. The central task that you will be performing is the generation and editing of blocks for STEP 7 programs. During the input, a basic syntax check is performed which simplifies faultless programming. Syntax errors are shown in different colors.

The Editor offers the following:
- Programming an S7 block in the language S7-SCL
- Convenient insertion of language elements and block calls with drag&drop
- Direct syntax check during programming.
- Setting the editor according to your requirements; for example, through syntax oriented coloring of the different language elements
- Checking the completed block through compilation
- Display of all errors and warnings that occur during compilation
- Identifying the faulty location in the block, optionally with error description and information regarding error removal

**Debugger**

The S7-SCL debugger allows for testing a program during its run in the AS and thus locates possible logical errors.

To this end, the S7-SCL provides two different test modes:
- Step by step monitoring
- Continuous monitoring

In the case of “step by step monitoring”, the logical program sequence is traced. You can execute the program algorithm instruction by instruction, and monitor in a result window how the tag contents that are being edited change.

With “continuous monitoring“ we test a group of instructions within a block. During the test run, the values of the tags and the parameters are shown in a chronological sequence and -to the extent possible- updated cyclically.
4. Sample Task Tank Content

4.1 Task Description

For our first program, we are programming the calculation of the contents of a tank.

The tank has the shape of an upright cylinder. The level of the content is measured with an analog sensor. In the task, the value of the level is already available, standardized in the unit 'Meter'.

The program is to be programmed in a function FC140 'calculate_volume_tank'. Transfer parameters are the diameter and the level in the unit Meter. The result is the content of the tank in the unit Liter.

4.2 Assignment List/Tag Table

Since for modern programming, tags are used rather than absolute addresses, first the global PLC tags have to be specified.

These global PLC tags are descriptive names with comments for those inputs and outputs that are used in the program. Later, during programming, we can access the global PLC tags by means of this name. These global tags can be used in the entire program in all blocks.

Default Tag Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>filling_level_tank1</td>
<td>REAL</td>
<td>%MD40</td>
<td>filling level of tank1 (meter)</td>
</tr>
<tr>
<td>diameter_tank1</td>
<td>REAL</td>
<td>%MD44</td>
<td>diameter tank1 (meter)</td>
</tr>
<tr>
<td>volume_tank1</td>
<td>REAL</td>
<td>%MD48</td>
<td>volume tank1 (liter)</td>
</tr>
</tbody>
</table>
4.3 Program Structure

The program structure is written in so-called blocks. As standard, the organization block Main [OB1] is already there. It represents the interface to the CPU’s operating system which calls this block automatically and processes it cyclically.

From this organization block, additional blocks can be called in turn for structured programming, such as the function calculate_volume_tank [FC140]. The purpose is breaking down an overall task into partial tasks; they can then be solved and tested more easily in their functionality.

Structure of the Sample Task Tank Content

- Organization block Main [OB1]
  - Block called cyclically by the operating system. Here, the call of the function calculate_volume_tank is shown [FC140]

- Function: calculate_volume_tank [FC140]
  - In this example, contains the actual program. Is called by [OB1].
4.4 Interface of the Block calculate_volume_tank [FC140]

Before the program can be written, the interface of the block has to be declared. When the interface is declared, only the local tags that are known in this block are specified.

The tags or interface parameters consist of two groups:

- **Block parameters that generate the interface of the block for the call in the program.**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Function</th>
<th>Available in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input parameter</td>
<td>Input</td>
<td>Parameters whose values the block reads.</td>
<td>Functions, function blocks and some types of organization blocks</td>
</tr>
<tr>
<td>Output parameter</td>
<td>Output / Return</td>
<td>Parameters whose values the block writes.</td>
<td>Functions and function blocks</td>
</tr>
<tr>
<td>In/out parameter</td>
<td>InOut</td>
<td>Parameters whose value the block reads when called, and after processing writes to the same parameter.</td>
<td>Functions and function blocks</td>
</tr>
</tbody>
</table>

- **Local data used for storing intermediate results.**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Function</th>
<th>Available in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporary local data</td>
<td>Temp</td>
<td>Tags that are used to store temporary intermediate results. Temporary data is maintained for one cycle only.</td>
<td>Functions, function blocks and organization blocks</td>
</tr>
<tr>
<td>Static local data</td>
<td>Static</td>
<td>Tags that are used for storing static intermediate results in the instance data block. Static data is retained -also over several cycles- until it is written again.</td>
<td>Function blocks</td>
</tr>
</tbody>
</table>
The interface parameters used in our sample program for the block 'calculate_volume_tank [FC140]' are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>filling_level</td>
<td>REAL</td>
<td>filling level tank (meter)</td>
</tr>
<tr>
<td>IN</td>
<td>diameter</td>
<td>REAL</td>
<td>diameter tank (meter)</td>
</tr>
<tr>
<td>OUT</td>
<td>volume</td>
<td>REAL</td>
<td>volume of liquid in the tank (liter)/ -1 if error</td>
</tr>
</tbody>
</table>

The function call in a block programmed in FBD looks like this:

**Function call: Representation in FBD**

```
FC140
  calculate_volume_tank
    filling_level: REAL
    diameter: REAL
    REAL : volume
```

### 4.5 Note on the solution

To solve the task, the formula for volume calculation of an upright cylinder is used. The conversion factor 1000 is used to calculate the result in liters.

\[
V = \frac{d^2}{4} \pi h \\
\Rightarrow \quad \text{volume} = \frac{\text{diameter}^2}{4} \times 3.14159 \times \text{filling_level} \times 1000
\]
5. **Programming the Tank Content Calculation for the SIMATIC S7-1200 in S7-SCL**

In the steps below, a project can be set up for the SIMATIC S7-1200 and the solution to the task can be programmed.

5.1 **Setting Up the Project and Configuring the Hardware**

1. The central tool is the *Totally Integrated Automation Portal*. It is called here with a double click. (→ TIA Portal V11)

![TIA Portal V11]

2. Programs for the SIMATIC S7-1200 are managed in projects. We are now setting up such a project in the portal view. (→ Create new project → scl_startup → Create)

![Creating a new project in TIA Portal]

---

*SCE Training Curriculum*  
TIA Portal Module 010-090, Edition 09/2012  
SCE_EN_010-090_R1209_Startup - High Level Language Programming with S7-SCL and SIMATIC S7-1200  
To be used only in Educational/R&D Facilities  
Unrestricted / © Siemens AG 2012. All Rights Reserved
3. Now, 'First steps' are suggested for configuration. We first want to 'Configure a device'.

4. Then we 'Add new device' with the 'Device name controller 001'. To this end, we select from the catalog 'CPU1214C AC/DCRly' with the matching order number. ('Add new device' → controller 001 → PLC → SIMATIC S7-1200 → CPU → CPU1214 AC/DC/Rly → 6ES7 214-1BE30-0XB0 → V2.2 → Add)
5. The software now changes automatically to the project view, with the opened hardware configuration in the device view. For the CPU to later access the correct CPU, its 'ETHERNET address' has to be set.

(→ Properties → General → ETHERNET address → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0)
5.2 Setting Up the Program

1. Open the 'Default tag table' and enter the 'Operands' with their names and data types.

   (→ controller 001[CPU1214 AC/DC/Rly] → PLC tags → Default tag table → Add new)
2. In order to generate the function calculate_volume_tank [FC140], in project navigation select 'controller 001[CPU1214 AC/DC/Rly]' and then 'Program blocks'. Next, double click on 'Add new block'. In the selection, select 'Function (FC)' and assign the name 'calculate_volume_tank'. Change the programming language to 'SCL'. The numbering can be changed by switching from automatic to manual. Enter the number 140. Accept the inputs with 'OK'.

(→ controller 001[CPU1214 AC/DC/Rly] → Program blocks → Add new block → Function (FC) → calculate_volume_tank → SCL → manual → 140 → OK)
3. The block 'calculate_volume-tank [FC140]' is opened automatically. Now, enter 'Input and Output parameter' of the block as specified. All local tags should be provided with 'sufficient comment' for better understanding.

(→ Enlarge interface of block FC140 → enter interface parameters)

Note:
To avoid a confusion with the PLC tags, it is helpful to write the local tags in lowercase letters.
4. After the local tags were declared, we can start creating the program shown here. The square function 'SQR' integrated in SCL is used to square a function. The number to be squared is in parentheses. To insert the function, drag it to the location in the program where it is to be used (→ Instructions (menu to the right!) → Simple instructions → Math functions → SQR).

5. Supplement the program as shown below.
6. The program can now be saved and compiled.

(save project → Compile)

7. Syntax errors that occurred are detected during compilation and displayed in 'Menu info/Compile'.

(info → Compile)
8. Now, the function can be called in the 'Main[OB1]'. Before we open the block 'Main[OB1]' with a double click, we change its programming language to 'FBD'.

(→ Main[OB1] → Switch programming language → FBD)

9. Next, we simply drag the function 'calculate_volume_tank[FC140]' into Network 1 of the Main[OB1]. The interface parameters of the function 'calculate_volume_tank[FC140]' have to be wired to the global PLC tags, as shown here. Remember to document the networks in the Main[OB1]. (→ Main[OB1] → Program blocks → calculate_volume_tank[FC140])
10. Save the project again with 

11. Now, in order to load the program blocks and the device configuration to the CPU, first select the folder `controller 001[CPU1214 AC/DC/Rly]` and then click on the symbol `Download to device`. 

Note: 
Through the loading process, the project is automatically compiled once more and errors are searched for.
12. Prior to the download, an overview is displayed to check the steps that are to be performed. Start these steps with 'Load', and start the CPU with 'Start all 'Finish'. (→ Load → Start all → Finish)
5.3 Testing the Program

1. By clicking on the symbol Monitoring on/off, you can observe the status of the input and output tags at the block `calculate_volume_tank` while the program is tested.
2. Since we don't have an analog sensor and therefore no corresponding process value available, we have to specify the values 'diameter_tank1' and 'filling_level_tank1' by using a monitoring table. Set up a 'new monitoring table' and enter the two values. Switch on the 'Monitoring mode' to view the current values.

(→ controller 001 → Monitoring and force table → New monitoring table → diameter_tank1, filling_level_tank1)

3. To specify the values, we now enter a control value in the column 'Control value'. With the button 'Write control values once and immediately', the values are transferred to the CPU.

(→ diameter_tank1 = 10.0 → filling_level_tank1 = 7.0)
4. Now, the program can be checked in OB1. (→)

5. We can monitor the values of the individual tags in the SCL editor. To this end, switch the 'Monitoring mode on'. (→)

Note:
If you click on this button, the current values of the tags programmed in this line are displayed.
5.4 Expanding the Program

Now, the block 'calculate_volume_tank' is to check whether faulty data was entered at the input parameters. In addition, the value 'height_max' is transferred to the block. It specifies the height of the tank.

The block is to evaluate whether the level of the tank is less than zero or more than the specified height of the tank. In addition, it is to be checked whether the diameter was specified as less than zero. If there is an error, a boolean output parameter is to return 'er' TRUE, and the value of the parameter is to be 'volume' -1.

Expanding the assignment list/tag table:

| Address %Q1.7 | Symbol error_bit | Data Type BOOL | Comment error, calculation not possible |

Expanding the interface of function FC140: calculate_volume_tank

| Type IN/OUT | Symbol height_max | Data Type REAL | Comment maximum height (meter) |
| Symbol er | Data Type BOOL | er = true -> error |

Function call: representation in FBD

```
FC140
  calculate_volume_tank
    Filling_level: REAL
    height_max: REAL
    diameter: REAL
    BOOL : er
    REAL: volume
```
1. Expand the block’s *tag table* as specified above.
   (→ controller 001[CPU1214 AC/DC/Rly] → PLC tags → default tag table → enter operands)

2. Expand the block’s *Interface parameters* as specified above.
   (→ controller 001[CPU1214 AC/DC/Rly] → Program blocks → calculate_volumeTank) → enter parameters)
3. Now, supplement the program as indicated below and search for syntax errors by compiling it. Save the program and load it to the controller.

(Write program → Save project → )
4. Since the block's parameters were changed, the call has to be updated in OB1. Open OB1 and scroll to the location of the block call. With the right mouse key, open the context menu and select 'Update'.

(→ controller 001[CPU1214 AC/DC/Rly] → Program blocks → Main [OB1] → right mouse key → Update)

5. Now, the old and the new interface are shown. Confirm with 'OK'.

(→ OK)
6. Supplement the values at the input parameter 'height_max' and at the output parameter 'er' as shown below. Compile, save and load the program to the controller.

(Supplement parameters → Save project →)

7. Check the changes in the 'Monitoring mode' of the block 'calculate_volume_tank'.

(→ controller 001[CPU1214 AC/DC/Rly] → Program block → calculate_volume_tank →)

Note:
The display format of the current values can be changed by right clicking the value and opening the context menu.