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

Siemens Automation Cooperates with Education

TIA Portal Module 010-050
Analog Value Processing with SIMATIC S7-1200
Matching SCE training packages for these training curriculums

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**  
  Order number: 6ES7214-1BE30-4AB3

- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**  
  Order number 6ES7214-1AE30-4AB3

- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**  
  Order number 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](http://siemens.com/sce/tp)

**Continued Training**

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

**Additional information regarding SCE**

[siemens.com/sce](http://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](mailto: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 curriculums 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.
Contents

1. Preface ........................................................................................................................................... 4
2. Notes on Programming the SIMATIC S7-1200 .............................................................................. 6
   2.1 Automation System SIMATIC S7-1200 ...................................................................................... 6
   2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11) ....................................... 6
3. Analog Signals .................................................................................................................................. 7
4. Data Types at the SIMATIC S7-1200 ............................................................................................ 8
5. Reading In/Reading Out Analog Values ......................................................................................... 9
   5.1 Normalizing Analog Values ........................................................................................................ 10
6. Sample Task – Monitoring the Tank Level .................................................................................... 10
   6.1. Programming Level Monitoring for the SIMATIC S7-1200 .................................................... 11
1. **Preface**

Regarding its content, module SCE_EN_010-050 is part of the training unit *Basics of PLC Programming* and represents a fast a *fast entry point* for programming the SIMATIC S7-1200 with the TIA Portal.

![Diagram of training modules]

**Training Objective**

In this module SCE_EN_010-050, the reader learns how to program limit value monitoring for an analog value. The controller (PLC) in our case is the SIMATIC S7-1200 and the program is created with the programming tool TIA Portal. Module SCE-DE-010-1050 provides the fundamentals and shows how it’s done using a detailed example.

**Prerequisites**

To successfully work through module SCE_EN_010-050, the following knowledge is assumed:

- How to operate Windows
- Fundamentals of PLC programming with the TIA Portal (for example, Module_010-010 ‘Startup’ Programming the SIMATIC S7-1200 with TIA-Portal V11)
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 Ultimate/Windows 2003 Server R2/Windows Server 2008 Premium SP1, Business SP1, Ultimate SP1
2. Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
3. Ethernet connection between PC and CPU 315F-2 PN/DP
4. PLC SIMATIC S7-1200; for example, CPU 1214C. The inputs have to be brought out to a panel.
2. Notes on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals.

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels with the integrated WinCC Basic
- With additional WinCC packages, visual display solutions for PCs and other panels can be prepared

All functions are supported with detailed online help.
3. **Analog Signals**

In contrast to binary signals that can assume only the two signal states "Voltage present +24V" and "Voltage not present 0V", analog signals within a certain range can assume any number of values. A typical example of an analog sensor is a potentiometer. Depending on the position of the rotary button, any resistance can be set, up to the maximum value.

Below are some examples of analog variables in control engineering:

- Temperature -50 ... +150°C
- Flow 0 ... 200l/min
- Speed 500 ... 1500 r/min
- etc.

Using a transducer, these variables are converted into electrical voltages, currents or resistances. If, for example, a certain speed is to be recorded, the speed range of 500 ... 1500 r/min can be converted by means of a transducer into a voltage range of 0 ... +10V. If a speed of 865 rpm is measured, the transducer would read out a voltage of +3.65 V.

These electrical voltages, currents or resistances are then connected to an analog module that digitalizes this signal.

If analog variables are processed with a PLC, the voltage, current or resistance value that was read in has to be converted into digital information. This conversion is called analog/digital conversion (A/D conversion).

This means, for example, that the voltage 3.65V is stored in a series of binary digits as information. The more binary digits are used for digital representation, the finer is the resolution. If, for example, there were only 1 bit available for the voltage range 0 ... +10V, only one statement could be made whether the measured voltage is in the range 0 ... +5V or in the range +5V ... +10V. With 2 bits, the range can be divided into 4 individual ranges; i.e. 0 ... 2.5/2.5 ... 5/5 ... 7.5/7.5 ... 10V. A/D converters commonly used in control engineering convert with 8 or 11 bits.

With 8 bits, we have 256 individual ranges and with 11 bits a resolution of 2048 individual ranges.

!![](image-url)
### 4. Data Types at the SIMATIC S7-1200

The SIMATIC S7-1200 is provided with a large number of data types that can be used to represent different numerical formats. The list below shows the elementary data types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size (Bit)</th>
<th>Range</th>
<th>Example of a Constant Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bool</td>
<td>1</td>
<td>0 to 1</td>
<td>TRUE, FALSE, 0, 1</td>
</tr>
<tr>
<td>Byte</td>
<td>8</td>
<td>16#00 to 16#FF</td>
<td>16#12, 16#AB</td>
</tr>
<tr>
<td>Word</td>
<td>16</td>
<td>16#0000 to 16#FFFF</td>
<td>16#ABCD, 16#0001</td>
</tr>
<tr>
<td>DWord</td>
<td>32</td>
<td>16#00000000 to 16#FFFFFFF</td>
<td>16#02468ACE</td>
</tr>
<tr>
<td>Char</td>
<td>8</td>
<td>16#00 to 16#FF</td>
<td>'A', 't', '@'</td>
</tr>
<tr>
<td>Sint</td>
<td>8</td>
<td>-128 to 127</td>
<td>123, -123</td>
</tr>
<tr>
<td>Int</td>
<td>16</td>
<td>-32768 to 32767</td>
<td>123, -123</td>
</tr>
<tr>
<td>DInt</td>
<td>32</td>
<td>-2.147.483.648 to 2.147.483.647</td>
<td>123, -123</td>
</tr>
<tr>
<td>USInt</td>
<td>8</td>
<td>0 to 255</td>
<td>123</td>
</tr>
<tr>
<td>UInt</td>
<td>16</td>
<td>0 to 65.535</td>
<td>123</td>
</tr>
<tr>
<td>UDInt</td>
<td>32</td>
<td>0 to 4.294.967.295</td>
<td>123</td>
</tr>
<tr>
<td>Real</td>
<td>32</td>
<td>+/-1.18 x 10-38 to +/-3.40 x 10 38</td>
<td>123,456, -3.4, -1.2E+12, 3.4E3</td>
</tr>
<tr>
<td>LReal</td>
<td>64</td>
<td>+/-2.23 x 10-308 to +/-1.79 x 10 308</td>
<td>12345.123456789, -1.2E+40</td>
</tr>
<tr>
<td>Time</td>
<td>32</td>
<td>T#-24d_20h_31m_23s_648ms_to T#-24d_20h_31m_23s_647ms Stored as -2.147.483.648 ms to +2.147.483.647 ms</td>
<td>T#5m_30s, 5#-2d T#1d_2h_15m_30x_45ms</td>
</tr>
<tr>
<td>String</td>
<td>Variable</td>
<td>0 to 254 characters in byte size</td>
<td>'ABC'</td>
</tr>
</tbody>
</table>

**Note:**
For analog value processing, the data types 'INT' and 'REAL' play an important part since entered analog values are present as integers in the format 'INT' and for accurate further processing, only floating point numbers 'REAL' can be used because of the rounding off error in the case of 'INT'.
5. **Reading In/Reading Out Analog Values**

Analog values are entered/read out in the PLC as word information. These words are accessed with the operands

- `%IW 64` Analog input word 64
- `%QW 80` Analog output word 80

for example.

Each analog value ("channel") is assigned one input or output word. The format is 'Int', an integer.

Addressing the input or output word depends on the addressing in the device overview. For example:

![Device overview diagram]

The address of the first analog input would be here `%IW 64`, the address of the second analog input `%IW 66`, and the address of the analog output `%QW 80`.

The analog value transformation for further processing in the PLC is the same for analog inputs and analog outputs.

The digitalized values look like this:

![Nominal range of the analog value]

Often, these digitalized values have to be normalized through corresponding further processing in the PLC.
5.1 Normalizing Analog Values

If an analog input value is present as digitalized value, it usually has to be normalized so that the numerical values correspond to the physical values in the process.

Likewise, the analog output to the IO output word usually takes place only after the output value is normalized. In STEP7 programs, computing operations are used for normalizing. For this to be done as accurately as possible, the values have to be converted to the data type REAL normalizing, to keep the rounding off errors to a minimum.

In the chapters below, an example is provided using level monitoring of a tank as an illustration.

6. Sample Task – Monitoring the Tank Level

We are going to program monitoring the level in a tank.

A sensor measures the level in a tank and converts it into the voltage signal 0 to 10V. 0V corresponds to a level of 100 liters and 10V to a level of 1000 liters.

This sensor is connected to the analog input of the SIMATIC S7-1200. Now, this signal is to be entered in a function FC1 and normalized.

Next, the following is to be programmed: monitoring and displaying the maximum permissible level of 990 liters and monitoring the minimum permissible level of 110 liters.

Assignment list:

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Data Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IW 64</td>
<td>AI_level_tank1</td>
<td>Int</td>
<td>Analog input level Tank1</td>
</tr>
<tr>
<td>%Q 0.0</td>
<td>Tank1_max</td>
<td>Bool</td>
<td>Display level &gt; 990 liters</td>
</tr>
<tr>
<td>%Q 0.1</td>
<td>Tank1_min</td>
<td>Bool</td>
<td>Display level &lt; 110 liters</td>
</tr>
</tbody>
</table>
6.1. Programming Level Monitoring for the SIMATIC S7-1200

The project is managed and programmed with the software 'Totally Integrated Automation Portal'.

Here, under a uniform interface, the components such as the control system, visualization and networking the automation solution are set up, parameterized and programmed. For error diagnosis, online tools are available.

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

1. The central tool is the 'Totally Integrated Automation Portal'. It is called here with a double click. (→ Totally Integrated Automation Portal V11)
2. Programs for the SIMATIC S7-1200 are managed in projects. Such a project is now set up in the Portal View. (→ Create new project → Tank_Analog → Create)
3. Now, 'First Steps' are recommended for the configuration. First, we want to 'Configure a device'. (→ First steps → Configure a device)
4. Then we 'Add new device' with the device name "controller_tank". To this end, we select from the catalog 'CPU1214C' with the matching order number. (→ Add new device → controller_tank → CPU1214C → 6ES7 …… → Add)
5. The software now changes automatically to the Project View with the opened hardware configuration. Here, more modules can be added from the hardware catalog (on the right). Here, the signal board for an analog output is to be inserted from the catalog using drag&drop. (→ Catalog → Signal board → AO1 x 12Bit → 6ES7 232-...)
6. In the 'Device view', the addresses of the inputs and outputs can be checked or reset.
Here, the integrated analog outputs of the CPU have the addresses %IW64 to %IW66 and the
integrated digital outputs the addresses %Q0.0 to %Q1.1.
The analog output at the signal board has the address %QW80.
7. So that the software later accesses the correct CPU, its IP address and the subnet mask have to be set. (→ Properties → General → PROFINET interface → Ethernet addresses → IP address: 192.168.0.1 → subnet mask: 255.255.255.0)
8. Since modern programming does not program with absolute addresses but with tags, the **global PLC tags** have to be specified here.

These global PLC tags are descriptive names with a comment for those inputs and outputs that are used in the program. Later, during programming, the global PLC tags can be accessed by means of this name.

These global tags can be used in the entire program in all blocks.

To this end, select in project navigation 'controller_tank [CPU1214C DC/DC/DC]' and then 'PLC tags'. With a double click, open the table 'PLC tags' and enter the names for the inputs and the outputs, as shown below.

(→ controller_tank[CPU1214C DC/DC/DC] → PLC tags → Default tag table)
9. To create function block FC1, select in Project navigation controller_tank [CPU1214C DC/DC/DC] and then 'Program blocks'. Next, double click on 'Add new block'. (→ controller_tank[CPU1214C DC/DC/DC] → Program block → Add new block)
10. Select ‘Function(FC)’ and assign the name ‘supervision filling level tank1’. As programming language, ‘FBD’ (function block diagram) is specified. Numbering is automatic. Since this FC1 is called later by its symbolic name, the number is no longer that important. Accept the inputs with ‘OK’. (→ Function (FC1) → supervision filling level tank1 → FBD → OK)
11. The block ‘**supervision filling level tank1’ [FC1]**’ is opened automatically. Before writing the program, the interface of the block has to be declared.

When the interface is declared, the local variables are specified that are known in this block.

The variables consist of two groups:

- **Block parameters that are the interface of the block for calls 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</td>
<td>Parameters whose values the block writes.</td>
<td>Functions and function blocks</td>
</tr>
<tr>
<td>InOut parameter</td>
<td>InOut</td>
<td>Parameters whose values the block reads when called, and after processing writes again to the same parameter.</td>
<td>Functions and function blocks</td>
</tr>
</tbody>
</table>

- **Local data used for storing interim 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>Variables used to store temporary interim results. Temporary data is retained for one cycle only.</td>
<td>Functions, function blocks and organization blocks</td>
</tr>
<tr>
<td>Static local data</td>
<td>Static</td>
<td>Variables used to store static interim results in the instance data block. Static data is retained -even over several cycles- until it is rewritten.</td>
<td>Function blocks</td>
</tr>
</tbody>
</table>
12. When local variables are declared, the following variables are needed for our example.

**Input:**
- `tank_level_AI`: Here, the level sensor enters the analog value

**Output:**
- `tank_max`: Here, the status of the maximum display is written to the output
- `tank_min`: Here, the status of the minimum display is written to the output

**Temp:**
- `tank_level_real`: This variable is needed to store an interim value
- `tank_level_norm`: Here, a value for the level, normalized in the floating point format to the range of 100 to 1000 liters is provided.

In this example, it is particularly important to use the correct data types since otherwise they would not be compatible in the following program with the conversion functions that are used. For the sake of clarity, all local variables should be provided with a sufficient comment.

```plaintext
<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tank_level_AI</td>
<td>Int</td>
<td>analog input filling level tank (integer)</td>
</tr>
<tr>
<td>tank_max</td>
<td>Bool</td>
<td>display max filling level 990 liter</td>
</tr>
<tr>
<td>tank_min</td>
<td>Bool</td>
<td>display min filling level 110 liter</td>
</tr>
<tr>
<td>tank_level_real</td>
<td>Real</td>
<td>memory filling level tank (floating point 32 bit)</td>
</tr>
<tr>
<td>tank_level_norm</td>
<td>Real</td>
<td>memory filling level tank normalized (floating point 32 bit)</td>
</tr>
</tbody>
</table>
```

In this example, it is particularly important to use the correct data types since otherwise they would not be compatible in the following program with the conversion functions that are used. For the sake of clarity, all local variables should be provided with a sufficient comment.
13. After the local variables were declared, the program can be entered only by using the names of the variables (identified with the symbol '#'). For the example in FBD, it could look like this:

Program in function block diagram (FBD):

![Program in function block diagram (FBD)](image-url)
Program in ladder diagram (LAD):

### Network 1:
- **Block title**: Fill level control for a tank min 110 liter max 990 liter
- **Comment**: NORM_X Int to Real
- **Function**: 0 MIN EN 27648 VALUE OUT 27648 MAX
- **Variables**: 
  - #tank_level_int
  - #tank_level_real

### Network 2:
- **Block title**: Transform analog input 0-27648 (integer) to a value 0-1 (floating point number)
- **Comment**: SCALE_X Real to Real
- **Function**: 1000.0 MIN EN #tank_level_real VALUE 10000.0 MAX
- **Variables**: 
  - #tank_level_int
  - #tank_level_norm

### Network 3:
- **Block title**: Display filling level = 990 liter
- **Comment**: 
  - #tank_level_norm
  - #tank_max

### Network 4:
- **Block title**: Display filling level = 110 liter
- **Comment**: 
  - #tank_level_norm
  - #tank_min
14. Next, the ‘Properties’ of the block ‘Main[OB1]’ that is processed cyclically are selected. Block properties can be changed. (→ Properties → Main[OB1])
15. In the properties, select the programming ‘Language’ function block diagram ‘FBD’. (→ FBD → OK)
16. Now, the block "supervision filling level tank1 [FC1]" has to be called from the program block Main [OB1]. Otherwise, the block would not be processed. With a double click on 'Main [OB1]' open this block. (→ Main [OB1])
17. With drag&drop, the block "supervision filling level tank1 [FC1]" can then be dragged to Network 1 of the block Main [OB1]. Don't forget to document the networks also in the block Main [OB1].

(→ supervision filling level_tank1 [FC1])
18. Next, the input variable as well as the output variable is wired in OB1 with the PLC tags shown here. Clicking on saves the project. (→ „AI_LEVEL_TANK1“ → „TANK1_MAX“ → “TANK1_MAX“ → Save project)
19. To load your entire program to the CPU, first highlight the folder `controller_tank` and then click on the symbol [Load to device](#). (→ controller_tank → [Load to device](#))
20. If the PG/PC interface was not specified previously, a window is displayed where this can still be done. (→ PG/PC interface for loading → load)
21. Then, click on 'Load' once more. During downloading, the status is displayed in a window. (→ Load)

22. Successful downloading is indicated in a window. Now click on ‘Finish’. (→ Finish)
23. Next, start the CPU by clicking on the symbol (→)

24. Confirm the question whether you actually want to start the CPU with 'OK'. (→ OK)
25. By clicking on the symbol Monitoring on/off, the status of the variables can be monitored while the program is tested. (→ supervision filling level tank1[FC1] → Monitoring on/off)