SCE Training Curriculum for the end-to-end automation solution Totally Integrated Automation (TIA)

TIA Portal Module 030-050
Analog Value Processing with SIMATIC S7-300
Suitable SCE trainer packages for these documents

**SIMATIC controllers**
- **SIMATIC S7-300 with CPU 314C-2PN/DP**
  Order no.: 6ES7314-6EH04-4AB3
- **SIMATIC S7-300 with CPU 314C-2PN/DP (upgrade)**
  Order no.: 6ES7314-6EH04-4AB4
- **SIMATIC S7-300 with CPU 315F-2PN/DP**
  Order no.: ES7315-2FH14-4AB1
- **SIMATIC ET 200S with CPU IM151-8 F PN/DP**
  Order no.: 6ES7151-8FB00-4AB1

**SIMATIC STEP 7 software for training**
- **SIMATIC STEP 7 Professional V11 - Single license**
  Order no.: 6ES7822-1CC01-4YA5
- **SIMATIC STEP 7 Professional V11 - Classroom license (up to 12 users)**
  Order no.: 6ES7822-1AA01-4YA5
- **SIMATIC STEP 7 Professional V11 - Upgrade license (up to 12 users)**
  Order no.: 6ES7822-1AA01-4YE5
- **SIMATIC STEP 7 Professional V11 - Student license (up to 20 users)**
  Order no.: 6ES7822-1AC01-4YA5

Please note that these trainer packages may be replaced by successor trainer packages. An overview of the currently available SCE packages is provided under: [siemens.com/sce/tp](http://siemens.com/sce/tp)

**Advanced training**
Please get in touch with your regional SCE contact for information on regional Siemens SCE advanced training [siemens.com/sce/contact](http://siemens.com/sce/contact)

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

**Information regarding usage**

This training curriculum for the end-to-end automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D facilities. Siemens AG does not make any guarantee regarding its contents.

This training curriculum may only be used for initial training on Siemens products/systems. That is, it may be copied in whole or in part and handed out to trainees for use within the context of their training. Distribution and reproduction of this document and disclosure of its contents are permitted within public education and further education facilities for educational purposes.

Any exceptions require written consent from the Siemens AG contact person: Mr. Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will 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. We do not consent to any commercial use of the training curriculum.

We would like to thank Michael Dziallas Engineering and all those involved for their support in creating this training curriculum.
1. Preface.................................................................................................................................4
2. Notes on programming for SIMATIC S7-300........................................................................6
   2.1 SIMATIC S7-300 automation system ...............................................................................6
   2.2 STEP 7 Professional V11 (TIA Portal V11) programming software................................6
3. Analog signals .....................................................................................................................7
4. Data types for SIMATIC S7-300 ........................................................................................9
5. Reading in/outputting analog values ..................................................................................10
   5.1 Scaling analog values .....................................................................................................11
6. Example task for fill level monitoring in a tank....................................................................12
7. Programming the fill level monitoring for SIMATIC S7-300 .............................................13
1. Preface

The SCE_EN_030-050 module contents form part of the ‘Advanced functions for PLC programming’ training unit and explain analog value processing with SIMATIC S7.

Learning objective:

In this module, the reader will learn how to program limit monitoring for the fill level of a tank. A sensor will provide the fill level of the tank as an analog value. The module explains the basic principles and outlines the procedure using a detailed example.

Requirements:

To successfully work through this module, the following knowledge is required:

- Basics of PLC programming with the TIA Portal
  (e.g., module SCE_EN_020-010_R1110_Startup programming with SIMATIC S7-300)
**Required hardware and software**

1. PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, approx. 2 GB of free hard disk space
   - Operating system Windows XP Professional SP3 / Windows 7 Professional / Windows 7 Enterprise / Windows 7 Ultimate / Windows 2003 Server R2 / Windows Server 2008 Premium SP1, Business SP1, Ultimate SP1
3. Ethernet connection between the PC and CPU 315F-2 PN/DP
4. SIMATIC S7-300 PLC, e.g., CPU 315F-2PN/DP with 16DI/16DO signal module. The inputs must be fed out to a control panel.
5. AI4/AO2 x 8Bit signal module for analog value acquisition
2. Notes on programming for SIMATIC S7-300

2.1 SIMATIC S7-300 automation system

The SIMATIC S7-300 automation system is a modular microcontroller system for the low and medium performance range. A comprehensive range of modules is available to optimally adapt the system to the automation task. The S7 controller consists of a power supply, a CPU, and input and output modules for digital and analog signals. If necessary, communication processors and function modules are also used for special tasks such as stepper motor control.

The programmable logic controller (PLC) uses the S7 program to monitor and control a machine or a process. The S7 program scans the I/O modules via input addresses (%I) and addresses their output addresses (%Q).

The system is programmed with the STEP 7 software.

2.2 STEP 7 Professional V11 (TIA Portal V11) programming software

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

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

STEP 7 Professional V11 provides the following functions for plant automation:

- Configuration and parameter assignment of the hardware
- Specification of the communication
- Programming
- Testing, commissioning, and servicing with operational/diagnostic functions
- Documentation
- Creation of visualizations for the SIMATIC Basic Panels using the integrated WinCC Basic software
- Visualization solutions for PCs and other panels can also be created with other WinCC software packages

Support is provided for all functions in a comprehensive online help system.
3. **Analog signals**

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

Examples of analog variables in control engineering:
- Temperature -50 to +150 °C
- Flow rate 0 to 200 l/min
- Speed 500 to 1500 rpm
- etc.

These variables are converted to electrical voltages, currents, or resistances using a measuring transducer. If, for example, a speed is to be recorded, the speed range of 500 to 1500 rpm can be converted to a voltage range of 0 to +10 V using a transducer. At a measured speed of 865 rpm, the measuring transducer then outputs a voltage value of +3.65 V.

![Diagram of analog signal conversion](image)

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

**Note:** Some analog modules can process a variety of signal types. This must be set in the device overview. Please refer to the information in the device manuals.
If analog variables will be processed with a PLC, the read-in voltage, current, or resistance value must be converted to digital information. This conversion is referred to as analog-digital conversion (A/D conversion). This means, for example, that the voltage value of 3.65 V is stored as information in a series of binary digits. The more binary digits the digital representation uses, the finer the resolution is. For example, if only 1 bit was available for the voltage range of 0 to +10 V, it would only be possible to indicate that the measured voltage falls within the range of 0 to +5 V or the range of +5 V to +10 V. With 2 bits, the range can be divided into 4 individual ranges, i.e., 0 to 2.5 / 2.5 to 5 / 5 to 7.5 / 7.5 to 10 V. Conventional A/D converters in control engineering use 8 or 11 bits for converting.

8 bits provide 256 individual ranges, while 11 bits provide a resolution of 2048 individual ranges.

8 bits: 256 ranges
11 bits: 2048 ranges

\[ 10 \text{ V: } 2048 = 0.0048828 \]

\[ \rightarrow \text{Voltage differences } < 5 \text{ mV can be detected} \]
4. Data types for SIMATIC S7-300

The SIMATIC S7-300 has many different data types for representing different numerical formats. A list of elementary data types is given below.

<table>
<thead>
<tr>
<th>Type and description</th>
<th>Size in bits</th>
<th>Format option</th>
<th>Range and numerical representation (lowest to highest value)</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL (bit)</td>
<td>1</td>
<td>Boolean text</td>
<td>TRUE/FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>BYTE (byte)</td>
<td>8</td>
<td>Hexadecimal</td>
<td>B#16#0 to B#16#FF</td>
<td>B#16#10</td>
</tr>
<tr>
<td>WORD (word)</td>
<td>16</td>
<td>Binary number</td>
<td>2#0 to 2#1111_1111_1111_1111</td>
<td>2#0000_0000_0000_0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hexadecimal</td>
<td>W#16#0 to W#16#FFFF</td>
<td>W#16#1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BCD</td>
<td>C#(0.0) to C#99</td>
<td>C#998</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decimal number (unsigned)</td>
<td>B#(0.0) to B#(255.255)</td>
<td>B#(1.14.100.120)</td>
</tr>
<tr>
<td>DWORD (double word)</td>
<td>32</td>
<td>Binary number</td>
<td>2#0 to 2#1111_1111_1111_1111_1111_1111_1111</td>
<td>2#1000_0001_0001_1000_1011_0111_1111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hexadecimal</td>
<td>DW#16#0000_0000 to DW#16#FFFF_FFFF</td>
<td>DW#16#00A2_1234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decimal number (unsigned)</td>
<td>B#(0.0.0.0) to B#(255.255.255.255)</td>
<td>B#(1.14.100.120)</td>
</tr>
<tr>
<td>INT (integer)</td>
<td>16</td>
<td>Decimal number</td>
<td>-32768 to 32767</td>
<td>1</td>
</tr>
<tr>
<td>DINT (integer, 32 bit)</td>
<td>32</td>
<td>Decimal number</td>
<td>L#-2147483648 to L#2147483647</td>
<td>L#1</td>
</tr>
<tr>
<td>REAL (floating-point number)</td>
<td>32</td>
<td>IEEE floating-point number</td>
<td>High limit: +/-3.402823e+38 Low limit: +/-1.175495e-38</td>
<td>1.234567e+13</td>
</tr>
<tr>
<td>S5TIME (Simatic time)</td>
<td>16</td>
<td>S7 time in increments of 10 ms</td>
<td>SST#0H_0M_0S_10MS to SST#2H_46M_30S_0MS and SST#0H_0M_0S_0MS</td>
<td>SST#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS</td>
</tr>
<tr>
<td>TIME (IEC time)</td>
<td>32</td>
<td>IEC time in increments of 1 ms, signed integer</td>
<td>T#24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS</td>
<td>T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS</td>
</tr>
<tr>
<td>DATE (IEC date)</td>
<td>16</td>
<td>IEC date in increments of 1 day</td>
<td>D#1990-1-1 to D#2168-12-31</td>
<td>DATE#1994-3-15</td>
</tr>
<tr>
<td>TIME_OF_DAY</td>
<td>32</td>
<td>Time of day in increments of 1 ms</td>
<td>TOD#0:0:0.0 to TOD#23:59:59.999</td>
<td>TIME_OF_DAY#1:10:3.3</td>
</tr>
<tr>
<td>CHAR (character)</td>
<td>8</td>
<td>ASCII character</td>
<td>´A´, ´B´ etc.</td>
<td>´B´</td>
</tr>
</tbody>
</table>

Note: The ‘INT’ and ‘REAL’ data types play a major role in analog value processing since read-in analog values exist as integers in ‘INT’ format, and for accurate further processing only ‘REAL’ floating point numbers are a possibility due to the rounding errors with ‘INT’.
5. **Reading in/outputting analog values**

Analog values are read into the PLC or output from the PLC as word information. These words are accessed, for example, with the following operands:

- `%IW 272` Analog input word 272
- `%QW 272` Analog output word 272

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

The addressing of the input and output words is based on the addressing in the device overview. For example:

Here, the address of the first analog input would be `%IW 272`, the address of the second analog input would be `%IW 274`, the address of the analog output would be `%QW 272` … , and so on.
The analog value transformation for further processing in the PLC is the same for analog inputs and analog outputs.
The digitalized value ranges appear as follows:

![Diagram showing analog value transformation]

Often, these digitalized values must still be scaled by further processing them in the PLC in an appropriate manner.

5.1 **Scaling analog values**

If an analog input value exists as a digitalized value, it must usually still be scaled so that the numeric values correspond to the physical values in the process.

Likewise, the analog output to the I/O output word usually occurs only after the output value has been scaled.

In STEP 7 programs, math functions are used for the scaling. To enable scaling to be as accurate as possible, the values for the scaling must be converted to the REAL data type to minimize rounding errors.

The following sections will present an example based on the monitoring of the fill level in a tank.
6. **Example task for fill level monitoring in a tank**

For our example, fill level monitoring will be programmed.

A sensor measures the fill level in a tank and converts this to a voltage signal of 0 to 10 V. Here, 0 V corresponds to a fill level of 100 liters and 10 V to a fill level of 1000 liters.

This sensor is connected to the first analog input of the SIMATIC S7-300. This signal will then be read in and scaled in a function FC1.

Then, a monitoring and display function for the maximum permissible fill level of 990 liters and a monitoring function for the minimum permissible fill level of 110 litres will be programmed.

**Assignment list:**

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IW 272</td>
<td>AI_Fuel_Tank1</td>
<td>Analog input fill level Tank1</td>
</tr>
<tr>
<td>%Q 0.0</td>
<td>Tank1_max</td>
<td>Display fill level &gt; 990 liters</td>
</tr>
<tr>
<td>%Q 0.1</td>
<td>Tank1_min</td>
<td>Display fill level &lt; 110 liters</td>
</tr>
</tbody>
</table>
7. **Programming the fill level monitoring for SIMATIC S7-300**

   The ‘Totally Integrated Automation Portal’ software is used for project management and programming.

   Components such as control, visualization, and networking of the automation solution are created, assigned parameters, and programmed here using a standard interface. Online tools are available for the error diagnostics.

   The following steps enable you to create a project for the SIMATIC S7-300 and to program the solution for the task:

   1. The central tool is the ‘Totally Integrated Automation Portal’, which is opened here with a double-click. (→ TIA Portal V11)

   ![TIA Portal V11](image)

   2. Programs for the SIMATIC S7-300 are managed in projects. Start by creating a project in the portal view (→ Create new project → Tank_Analog → Create).
3. ‘First steps’ for configuring are now suggested. We want to start with ‘Configure a device’. (→ First steps → Configure a device)

4. Next, we will ‘Add a new device’. Choose the ‘CPU 315F-2 PN/DP’ with the appropriate order number from the catalog. (→ Add new device → Control_Tank → CPU 315F-2 PN/DP → 6ES7 …… → Add)
5. The software now switches automatically to the project view containing the opened hardware configuration. Additional modules can now be added from the hardware catalog (on the right!) and the addresses of the inputs/outputs can be set in the ‘Device overview’. Here, there is one I/O module with 16 inputs (addresses %I0.0 - %I1.7) and 16 outputs (addresses %Q0.0 - %Q1.7). (→ Device overview → DI/DO → DI16/DO16 x 24V/0.5A → 6ES7 323-1BL00-0AA0 → I address: 0…1 → Q address: 0…1)
6. After that, drag an AI/AO module with 4 analog inputs (addresses %IW272 / %IW274 / %IW276 / %IW278) and 2 analog outputs (addresses %QW272 / %QW274) onto slot 5. (→ Device overview → AI/AO → AI4/AO2 x 8Bit → 6ES7 334-0CE01-0AA0 → I address: 272…279 → Q address: 272…275)
7. To ensure that the software will access the correct CPU later, the IP address and the subnet mask of the CPU must be set. (→ Properties → General → PROFINET interface [X2] → Ethernet addresses → IP protocol → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0) (see also: SCE_EN_030-010 module on setting the programming interface.)
8. Because modern programming uses tags and not absolute addresses, the **global PLC tags** must be defined here.

These global PLC tags are descriptive names with a comment for each input and output used in the program. The global PLC tags can then be accessed later during programming via their names. These global tags can be used in all blocks anywhere in the program.

In the project tree, select ‘Control_Tank [CPU 315F-2 PN/DP]’ and then ‘PLC tags’. Double-click the ‘default tag table’ to open it, and enter the names for the inputs and outputs as shown below.

(→ Control_Tank [CPU 315F-2 PN/DP] → PLC tags → default tag table)

**Note:** Note also the data types of the tags.
9. Next, we want to create the function with the program. In the project tree, select ‘Control_Tank [CPU 315F-2 PN/DP]’ and then ‘Program blocks’. Then, double-click ‘Add new block’. 

(→ Control_Tank [CPU 315F-2 PN/DP] → Program blocks → Add new block)
10. Select ‘**Function (FC)**’ and assign the name ‘**Fill level monitoring**’. Specify the ‘**FBD**’ function block diagram as programming language. The numbering will be automatic. Since this FC1 will be called using its symbolic name later, the number does not play an important role. Click ‘**OK**’ to accept your entries. (→ Function (FC1) → Fill level monitoring → FBD → OK)
11. The ‘**Fill level monitoring [FC1]**’ block opens automatically. The interface of the block must be declared before the program can be written. In the interface declaration, the local tags known only in this block are defined.

The tags are divided into two groups:

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Designation</th>
<th>Function</th>
<th>Available in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input parameters</td>
<td>Input</td>
<td>Parameters whose values are read by the block.</td>
<td>Functions, function blocks, and some types of organization blocks</td>
</tr>
<tr>
<td>Output parameters</td>
<td>Output</td>
<td>Parameters whose values are written by the block.</td>
<td>Functions and function blocks</td>
</tr>
<tr>
<td>In/out parameters</td>
<td>InOut</td>
<td>A parameter whose value is read by the block when it is called and is written back by the block to the same parameter after it is processed.</td>
<td>Functions and function blocks</td>
</tr>
</tbody>
</table>

- **Local data** that is used for saving intermediate results.

<table>
<thead>
<tr>
<th>Type</th>
<th>Designation</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 retained for only one cycle.</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 saving static intermediate results in the instance data block. Static data is retained until it is overwritten, which may be after several cycles.</td>
<td>Function blocks</td>
</tr>
</tbody>
</table>
12. In this example, the tags shown here are specified in the declaration of the local tags. All local tags should also be provided with a sufficiently descriptive comment for better understanding.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Offset</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tank_fillLe</td>
<td>Int</td>
<td></td>
<td>Analog input fill level Tank in integer format</td>
</tr>
<tr>
<td>tank_max</td>
<td>Bool</td>
<td></td>
<td>Display maximum fill level 990 Liter exceeded</td>
</tr>
<tr>
<td>tank_min</td>
<td>Bool</td>
<td></td>
<td>Display minimum fill level 110 Liter underrun</td>
</tr>
<tr>
<td>scale_return</td>
<td>Word</td>
<td>0.0</td>
<td>Return value module Scale</td>
</tr>
<tr>
<td>tank_fillNorm</td>
<td>Real</td>
<td>2.0</td>
<td>Fill level Tank maxi in float format</td>
</tr>
<tr>
<td>bi_polar</td>
<td>Bool</td>
<td>6.0</td>
<td>Declaration if analog input is bipolar or not</td>
</tr>
<tr>
<td>Ret_val</td>
<td>Void</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Note also the data types of the tags.
13. For scaling, we need the ‘SCALE’ instruction from the ‘Conversion operations’ folder in the function block diagram (FBD). Create the program for reading in and scaling the analog value as shown in the 2 networks here. (→ Conversion operations → SCALE)

Note: Refer to the online help for information on the function and connection of the ‘SCALE’ block.
14. In Networks 3 and 4, program the two ‘Comparator operations’ ‘CMP>=’ and ‘CMP<=' as shown here in order to evaluate the scaled analog value and to control the binary fill level displays. Make sure to convert the comparator operations to the correct data type (‘Real’) for floating-point numbers as shown here. (Comparator operations → CMP>= → Real → CMP<=' → Real)
15. The next step is to select the ‘Properties’ of the cyclically processed ‘Main [OB1]’ block. 
(→ Main [OB1] → Properties)

16. For the ‘Language’, choose the ‘FBD’ function block diagram programming language. (→ FBD → OK)
17. The ‘Fill level monitoring’ block must now be called from the Main [OB1] program block. Double-click ‘Main [OB1]’ to open this block. (→ Main [OB1])

18. The ‘Fill level monitoring [FC1]’ block can then be moved into Network 1 of the Main [OB1] block using a drag-and-drop operation. Don’t forget to document the networks in the Main [OB1] block. (→ Fill level monitoring [FC1])
19. Now, interconnect the input tags and the output tag with the PLC tags shown here in OB1. Then, click \textbf{Save project} to save the project.
20. To load your entire program to the CPU, first select the ‘Control_Tank [CPU 315F-2 PN/DP]’ folder and then click the Download to device icon. (→ Control_Tank [CPU 315F-2 PN/DP]
21. In the following dialog, select ‘PN/IE’ as the PG/PC interface type and then a suitable network card as the PG/PC interface. After a ‘Refresh’ of the accessible devices, you should see your ‘CPU 315F-2 PN/DP’ with address 192.168.0.1 and be able to select this CPU as the target device. Then, click ‘Load’. (→ Type of the PG/PC interface: PN/IE → PG/PC interface: …… → Refresh → CPU 315F-2 PN/DP → Load)

Note: Details on setting the PG/PC interface can be found in the SCE_EN_020-010 module.
22. Then, click ‘Load’ again. The status of the load operation will be displayed in a window. (→ Load)

![Load preview](image)

23. The successful load result is now displayed in a window. To place the CPU315F-2 PN/DP back in RUN mode, click ‘Start all’ and then ‘Finish’. (→ Start all → Finish)

![Load results](image)
24. Click the Monitoring on/off icon \( \text{领会} \) to monitor the program in the blocks. (\( \rightarrow \text{领会} \))