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

TIA Portal Module 030-020
IEC Timers and IEC Counters of 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](mailto: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. Instances and multi-instances for programming of SIMATIC S7-1200 .................................................7
   3.1 Instance data blocks/single instances ...............................................................................................7
   3.2 Multi-instances ..................................................................................................................................9
4. Example task for press control with time delay and a single instance .............................................11
5. Programming of the press with SIMATIC S7-300 .............................................................................12
6. Example task for belt control with a counter and a multi-instance ...................................................25
7. Programming the belt with SIMATIC S7-300 .......................................................................................26
1. Preface

The SCE_EN_030-020 module contents form part of the ‘Advanced functions for PLC programming’ training unit and explain how IEC Timers, IEC Counters, single instances and multi-instances are used for SIMATIC S7 programming.

Learning objective:

In this module, the reader will become acquainted with the use of single instances and multi-instances for SIMATIC S7 programming of IEC Timers and IEC Counters with the TIA Portal programming tool. The module explains the various types of instances and shows step-by-step how to expand two program blocks to include an IEC Timer and IEC Counter.

- Use of single instances
- Adding an IEC Timer to a function in a program
- Use of multi-instances
- Adding an IEC counter to a function block in a program

Requirements:

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

- Proficiency in working with Windows
- Basics of PLC programming with the TIA Portal and knowledge of block types of the S7-300 (e.g., module SCE_EN_020-010_R1201_Startup programming with SIMATIC S7-300 and module SCE_EN_030-010__R1201__Block types of 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.
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. **Instances and multi-instances for programming of SIMATIC S7-1200**

The call of a function block is referred to as an instance. An instance data block is assigned to every call of a function block and serves as a data memory. It stores the actual parameters and the static data of the function block.

The tags declared in the function block determine the structure of the instance data block.

**Use of single instances and multi-instances**

You can assign instance data blocks as follows:

- Call as a **single instance**:
  - A separate instance block per instance of a function block

- Call as a **multi-instance**:
  - One instance data block for several instances of one or more function blocks

3.1 **Instance data blocks/single instances**

The call of a function block that is assigned its own instance data block is called a single instance.

If the function block was created according to the rules for standard blocks (see SCE_EN_030-010 module), it can also be called multiple times.

However, you must assign another instance data block for each call as a single instance.
Example for single instances:

The following figure shows the control of two motors using one function block FB10 and two different data blocks:

The different data for the individual motors, such as speed, ramp-up time, and total operating time, are saved in the different instance data blocks DB 10 and DB 11.

Note: Some commands, such as timers and counters, behave like function blocks. When these are called, they also represent instances and require an assigned memory area, e.g., in the form of an instance data block.
3.2 Multi-instances

You may want to limit the number of data blocks used for instances, or this may be necessary due to lack of memory space.

If other function blocks, timers, counters, etc. that already exist will be called in a function block in your user program, you can call these other function blocks without separate (i.e., additional) instance DBs. Simply select the 'Multi-instance' call option:

Notes:

Multi-instances enable a called function block to store its data in the instance data block of the calling function block.

In this case, the calling block must always be a function block.

This allows you to concentrate the instance data in one instance data block and thus make better use of the number of DBs available.

Incidentally, this is always required when the calling block is to remain available for reuse as a standard block.
Example for multi-instances:

The following figure shows the call of a CTUD-type counter (Counter up/down) that is called twice. The different data for the two counters is stored as different multi-instances in the instance data block DB1 of the calling function block FB1.
4. Example task for press control with time delay and a single instance

For our program, the press control from the SCE_EN_030-010 module will be expanded to include a timer.

The task can be described as follows:
A press with protective equipment will be activated with START button S3 only if the protective grid has already been closed for at least 1 second. This state is monitored with a “Protective grid closed” sensor B1.
If this state exists, a 5/2-way valve M0 for the press cylinder is actuated so that a plastic shape can be pressed.
The press will raise again when the EMERGENCY STOP button (NC) is actuated, or the protective grid sensor B1 is no longer activated.
An instance DB will be used here as memory for the timer.

Assignment list:

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%I 0.1</td>
<td>EMERGENCY STOP</td>
<td>EMERGENCY STOP button (NC)</td>
</tr>
<tr>
<td>%I 0.3</td>
<td>S3</td>
<td>Start button S3 (NO)</td>
</tr>
<tr>
<td>%I 0.4</td>
<td>B1</td>
<td>“Protective grid closed” sensor (NO)</td>
</tr>
<tr>
<td>%Q 0.0</td>
<td>M0</td>
<td>Extend cylinder A</td>
</tr>
</tbody>
</table>

EMERGENCY STOP
5. Programming of the press with 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 open a project for the SIMATIC S7-300, save it under a new name, and adapt it to the new requirements:

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

2. Open the “startup” project from the SCE_EN_020_010 module in the portal view as a template for this program. (→ Open existing project → startup → Open)
3. ‘First steps’ for configuring are now suggested. We want to ‘Open the project view’. (→ Open the project view)

4. Start by saving the project under a different name. (→ Project → Save as)
5. ‘Save’ the project under the new name ‘press_timer’. (press_timer → Save)

6. To make the changes, double-click the ‘Program Press [FC1]’ block to open it. (Program Press [FC1])
7. Now, you can start to make changes to the program. Our solution requires a ‘TON’ ON delay. You will find this in the ‘Timers’ folder under ‘Instructions’. If you hover the mouse over an object, such as the TON timer, you will receive detailed information about this object. Click the detailed information to view the online help for this object. (→ Instructions → Timers → TON → Generate on-delay)
8. Take this opportunity to thoroughly review the online help information on all timer functions.
9. Next, insert a new network at the start of the block, and then drag the ‘TON’ timer into this network using the mouse. (→ TON)

10. The timer function requires a memory, which in this case requires you to create a new instance data block in the form of a ‘single instance’. (→ OK)

Note: A multi-instance can only be used for programming within a function block. This will be shown in the example for the IEC Counter below.
11. For declaration of the local tags, a Temp (temporary) tag will now also be added. 

\[ \rightarrow \text{q\_iec\_timer\_0} \]

\textbf{Temp:} 
\texttt{q\_iec\_timer\_0}
12. The binary state (Q) of the TON ON delay will be assigned to temporary tag q_iec_timer_0. For this reason, drag the `+1` instruction onto output Q of the TON on-delay. (→ `+1`)
13. Now, connect the ‘TON’ on-delay to time base ‘T#1s’ for 1 second and the ‘IN’ input to input tag ‘#sensor_steel mesh guard’. Enter temporary tag ‘q_iec_timer_0’ twice as shown here in order to scan the state of the ‘TON’ on-delay from Network 1 to Network 2 for control of the press cylinder. Then, click [Save project] to save the project. (→ T#5s → #sensor_steel mesh guard → q_iec_timer_0 → [Save project])
14. To download your entire program to the CPU, first select the 'Control press' folder and then click the Download to device icon. (→ Control press → Download to device)
15. If you forgot to specify the PG/PC interface beforehand, a window appears where it can be specified. (→ PG/PC interface for loading → Load)
16. Then, click ‘Load’ again. The status of the load operation will be displayed in a window.  
(→ Load)

<table>
<thead>
<tr>
<th>Status</th>
<th>Target</th>
<th>Message</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Control press</td>
<td>Ready for loading</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device configuration</td>
<td>Delete and replace system data in target</td>
<td>Download to device</td>
</tr>
<tr>
<td></td>
<td>Software</td>
<td>Download software to device</td>
<td>Consistent download</td>
</tr>
</tbody>
</table>

17. The successful load result is now displayed in a window. Now click ‘Finish’. This will also restart the CPU. (→ Finish)
18. Click the Monitoring on/off icon to monitor the state of the timer when testing the program.
6. Example task for belt control with a counter and a multi-instance

When blocks that will function as a sort of “black box” in any program are to be created, they must be programmed using tags. The following rule applies: absolutely addressed inputs/outputs, bit memory, etc. are not permitted to be used in these blocks. Only tags and constants may be used within the block.

If this type of multi-use block calls lower-level function blocks or a timer or counter, a separate data block must not be assigned to it.

The necessary memory space is provided as a multi-instance within the instance DB assigned to the calling function block.

In the following example, the function block containing a mode-dependent belt control system will be expanded to include a bottle counter.

The belt will then always transfer 20 bottles into a box. When the box is full, the belt will be stopped and the box must be replaced.

In this case, the ‘S1’ button will enable selection of ‘manual’ mode and the ‘S2’ button will enable selection of ‘automatic’ mode.

In ‘manual’ mode, the motor will be switched on as long as the ‘S3’ button is actuated, during which time the ‘S4’ button must not be actuated.

In ‘Automatic’ mode, the ‘S3’ button will switch on the belt motor and the ‘S4’ button (NC) will switch off the belt motor.

In addition, there is a sensor ‘B0’ that counts the bottles transferred to a box. When 20 bottles are counted, the belt is stopped.

When the first box or a new box is put in place, this must be confirmed with the ‘S5’ button.

Assignment list:

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%I 0.0</td>
<td>S1</td>
<td>Button operation mode manual S1 (NO)</td>
</tr>
<tr>
<td>%I 0.1</td>
<td>S2</td>
<td>Button operation mode automatic S2 (NO)</td>
</tr>
<tr>
<td>%I 0.2</td>
<td>S3</td>
<td>ON button S3 (NO)</td>
</tr>
<tr>
<td>%I 0.3</td>
<td>S4</td>
<td>OFF button S4 (NC)</td>
</tr>
<tr>
<td>%I 0.6</td>
<td>S5</td>
<td>Button S5 (NO) reset counter/new box</td>
</tr>
<tr>
<td>%I 0.7</td>
<td>B0</td>
<td>Sensor B0 (NO) bottle counter</td>
</tr>
<tr>
<td>%Q 0.2</td>
<td>M01</td>
<td>Belt motor M01</td>
</tr>
</tbody>
</table>
7. Programming the belt with 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 open a project for the SIMATIC S7-300, save it under a new name, and adapt it to the new requirements:

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

2. Open the “FB_Band” project from the M2 module in the portal view as a template for this program. (→ Open existing project → FB_Band → Open)
3. ‘First steps’ for configuring are now suggested. We want to ‘Open the project view’. (→ Open the project view)

4. Start by saving the project under a different name. (→ Project → Save as)
5. ‘Save’ the project under the new name ‘FB_Band_Counter’.

6. To create new global tags, double-click ‘default tag table’ under ‘PLC tags’ in ‘Control_belt conveyor’. 
7. Change the tag table as specified. Create the two global tags ‘B0’ and ‘S5’. (→ B0 / Bool / %I0.7 / Bottle counter sensor → S5 / Bool / %I0.6 / Reset counter/new box)

8. To make the changes in the program, double-click the ‘Band [FB1]’ block to open it. (→ Band [FB1])
9. Start by adding 2 rows to the interface for the input tags. (→ Interface → Input → Add row)

10. In addition add the following tags for the declaration of the local tags.

   **Input:**
   - `sensor_bottle`  
   - `reset_counter`  

   **Temp:**
   - `q_iec_counter_0`

![Image of interface showing added input tags](image-url)
11. After that, you can start to make changes to the program. We require a ‘CTD’ count down counter for our solution with the counter. You will find this in the ‘Counters’ folder under ‘Instructions’. If you hover the mouse over an object, such as the CTD counter, you will receive detailed information about this object. Click the detailed information to view the online help for this object. (Instructions → Counters → CTD → CTD Count down)
12. Take this opportunity to thoroughly review the online help information on all counters.
13. Next, insert a new network at the start of the block, and then drag the ‘CTD’ counter into this network using the mouse. (→ CTD)

14. The counter function requires a memory. Here, this memory is provided as a ‘multi-instance’ within the instance data block by the function block without the creation of a new instance data block. (→ Multi-instance → OK)

Note: A multi-instance can only be used for programming within a function block.
15. Now, connect the ‘CTD’ count down counter to the value specification ‘20’ for the 20 bottles and interconnect the ‘CD’ input with ‘#sensor_bottle’ and the ‘LD’ input with ‘#reset_counter’. Then, drag the instruction ‘<1’ onto output Q of the TON on-delay in order to assign the binary state of the CTD counter to temporary tag ‘#q_iec_counter_0’. (→ 20 → #sensor_bottle → #reset_counter → <1 → #q_iec_counter_0)
16. Now insert another AND instruction in the fourth network as shown here. Then, negate the second contact of the AND function and connect this to the local tag ‘#q_iec_counter_0’.

\[(\rightarrow \& \rightarrow \neg \rightarrow #q\_iec\_counter\_0)\]
17. Now open the ‘Main [OB1]’ block in order to update the call of the ‘Band [FB1]’ block there. (→ Main [OB1])

18. In the ‘Main [OB1]’ block, right-click “Band” and then ‘Update block call’. (→ Main [OB1] → Update block call)
19. Then select the ‘New interface’ and confirm with ‘OK’. (→ New interface → OK)

20. Connect the two new input tags to the PLC tags “B0” and “S5” shown here. Next, click to save the project. (→ “B0” → “S5” → Save project)
21. To download your entire program to the CPU, first select the ‘Control belt conveyor’ folder and click the Download to device icon. (→ Control belt conveyor →)

![Image of TIA Portal Module 030-020, Edition 03/2013](image-url)
22. If you forgot to specify the PG/PC interface beforehand, a window appears where it can be specified. (→ PG/PC interface for loading → Load)

<table>
<thead>
<tr>
<th>Device</th>
<th>Device type</th>
<th>Type</th>
<th>Address</th>
<th>Subnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control belt conveyor</td>
<td>CPU 315F-2 PN/DP</td>
<td>PPIE</td>
<td>192.168.0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU 315F-2 PN/DP</td>
<td>MPI</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Type of the PG/PC interface: PPIE
PG/PC interface: 230.MC 92566MC Gigabit
Connection to subnet: PLC, PPIE
1st gateway: 

Accessible devices in target subnet:

<table>
<thead>
<tr>
<th>Device</th>
<th>Device type</th>
<th>Type</th>
<th>Address</th>
<th>Target device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control press</td>
<td>CPU 315F-2 PN/DP</td>
<td>PPIE</td>
<td>192.168.0.1</td>
<td>Control press</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>PPIE</td>
<td>Access address</td>
</tr>
</tbody>
</table>

Define status information:
- Connected to address 192.168.0.1
- Scanning ended
23. Then, click ‘Load’ again. The status of the load operation will be displayed in a window. (
→ Load)

24. The successful load result is now displayed in a window. Now click ‘Finish’. This will also restart the CPU. (→ Finish)
25. Click the Monitoring on/off icon \( \text{\includegraphics[height=1cm]{monitoring_icon.png}} \) to monitor the state of the counter when testing the program. \( \text{\includegraphics[height=1cm]{monitoring_icon.png}} \)