Training Document for Integrated Automation Solutions
Totally Integrated Automation (TIA)

MODULE M3
IEC Timers and IEC Counters
for the SIMATIC S7-1200
This document was prepared by the Siemens AG for training purposes for the project: Siemens Automation Cooperates with Education (SCE). Siemens AG does not guarantee the contents.

Passing on this document or copying it and using and communicating its content within public training and continued education establishments is permitted. Exceptions require a written authorization by Siemens AG. (Michael Knust michael.knust@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.

We wish to thank the Michael Dziallas Engineering corporation and the instructors of vocational/professional schools as well as all other persons for their support during the preparation of this document.
1. Preface .......................................................................................................................................................5
2. Notes on Programming the SIMATIC S7-1200 .......................................................................................7
   2.1 Automation System SIMATIC S7-1200 .................................................................................................7
   2.2 Programming Software STEP 7 Basic V10.5 (TIA Portal V10.5) .......................................................7
3. Instances and Multi-Instances when Programming the SIMATIC S7-1200 ................................................8
   3.1 Instance Data Blocks/Single Instances ..............................................................................................8
   3.2 Multi-Instances ................................................................................................................................10
4. Sample Task Press Control with Time and Instance DB .........................................................................12
5. Programming the Press with a Time Delay with the SIMATIC S7-1200 ................................................13
6. Sample Task for Belt Control with Counter and Multi-Instance ............................................................29
7. Programming the Belt with the SIMATIC S7-1200 ...................................................................................30
The following symbols guide you through this module:

- Information
- Installation
- Programming
- Sample Task
- Notes
1. Preface

Regarding its content, module M3 is part of the training unit 'SIMATIC S7-1200 and TIA Portal' and explains the use of timers and counters when programming the SIMATIC S7 1200.

**Training Objective:**

In this module M3, the reader learns how to use the special instructions for timers and counters when programming the SIMATIC S7-1200. The module shows how to program a timer in a function with an instance DB/single instance as assigned memory. In addition, how to program a counter in a function block is described. In this case, the counter is not assigned a separate instance DB as memory, but a multi-instance.

**Prerequisites:**

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

- How to operate Windows
- Basics of PLC programming with the TIA Portal (for example, Module M1 – 'Startup' Programming of the SIMATIC S7-1200 with TIA Portal V10)
- Blocks for the SIMATIC S7-1200 (for example, Module M2 – Block Types for the SIMATIC S7-1200)
Hardware and software required

1. PC Pentium 4, 1.7 GHz (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
   Operating system Windows XP (Home SP3, Professional SP3)/Windows Vista (Home Premium SP1, Business SP1, Ultimate SP1)
2. Software STEP7 Basic V10.5 SP1 (Totally Integrated Automation (TIA) Portal V10.5)
3. Ethernet connection between PC and CPU 1214C
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 SIMATIC S7-1200 automation system is a modular mini-control system for the lower performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.
The S7 controller consists of a CPU that is already equipped with inputs and outputs for digital and analog signals.

Additional input and output modules (IO modules) can be installed if the integrated inputs and outputs are not sufficient for the desired application.

If necessary, communication processors for RS 232 or RS 485 are added.
An integrated TCP/IP interface is obligatory for all CPUs.

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 Basic V10.5.

2.2 Programming Software STEP 7 Basic V10.5 (TIA Portal V10.5)

The software STEP 7 Basic V10.5 is the programming tool for the automation system
- SIMATIC S7-1200

With STEP 7 Basic V10.5, the following functions can be utilized to automate a plant:
- Configuring and parameterizing the hardware
- Defining communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating visual displays for the SIMATIC Basic panels

All functions are supported with detailed online help.
3. Instances and Multi-Instances when Programming the SIMATIC S7-1200

Calling a function block is referred to as instance. To each call of a function block, an instance data block is assigned that is used for data storage. Here, the actual parameters and the static data is stored.

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

Applying single and multi-instances

Instance data blocks can be assigned as follows:

- Call as single instance:
  - A separate instance data block for each instance of a function block

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

3.1 Instance Data Blocks/Single Instances

The call of a function block to which its own instance data block is assigned is referred to as single instance.

If the function block was generated according to the rules for standard blocks (refer to Module M2), it can be called multiple times.

However, for each call as single instance, you have to assign a different instance data block.
Example of single instances:

The figure below shows two motors being controlled with a function block FB10 and two different data blocks:

The different data for the individual motors -for example, speed, power-up time, total operating time- is stored in the different instance data blocks DB10 and DB11.

Note: Some instructions such as timers and counters behave like function blocks. If they are called, they also represent instances and need an assigned memory area; in the form of an instance data block, for example.
3.2 Multi-Instances

Because of the memory capacity of the CPUs used, it is possible that you want to or you can allocate only a limited number of data blocks for instance data.

If in your user program, additional already existing function blocks, timers, counters, etc. are called in a function block, it is possible to call these additional function blocks without their own (that is, additional) instance DBs.

Simply select the call options 'Multi-Instance':

Notes:

For a function block that was called, multi-instances make it possible to place its data in the instance data block of the function block that is calling.

The block that is calling always has to be a function block in this case.

In this way, you concentrate the instance data in one instance data block; i.e., you can utilize the available number of DBs more efficiently.

This, by the way, always has to be done if the block that is calling is to be reusable as a standard block.
Example for Multi-Instances:

The figure below shows the control of two motors with one function block FB10 that is called twice. The different data for the two motors -for example, speed, power-up time, total operating time- is stored as different multi-instances in instance data block DB1 of the calling function block FB1.

Note: Some instructions such as timers and counters behave like function blocks. If they are called, they also represent instances and need an assigned memory area. These also can be provided as multi-instances.
4. Sample Task: Press Control with Timer and Instance DB

For our program, a time will be added to the press control in Module M1.

The task to be performed is as follows:
A press with a protective guard is to be started with a START button S3 only if the protective grid is closed. This state is monitored with a sensor Protective Grid closed B1. If this is the case, a 5/2 way valve M0 for the press cylinder is activated so that a plastic shape can be pressed.
The press is to retract again when the EMERGENCY OFF button (break contact) EMERGENCY OFF is operated, or the sensor Protective Grid B1 no longer responds. If the sensor Cylinder Extended B2 responds, the press is to retract again after a press time of 5 seconds.
An instance DB is used as the 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 OFF</td>
<td>EMERGENCY OFF button (break contact)</td>
</tr>
<tr>
<td>%I 0.3</td>
<td>S3</td>
<td>Start button S3 (make contact)</td>
</tr>
<tr>
<td>%I 0.4</td>
<td>B1</td>
<td>Sensor Protective Grid closed (make contact)</td>
</tr>
<tr>
<td>%I 0.5</td>
<td>B2</td>
<td>Sensor Cylinder extended (make contact)</td>
</tr>
<tr>
<td>%Q 0.0</td>
<td>M0</td>
<td>Extend Cylinder A</td>
</tr>
</tbody>
</table>
5. Programming the Press with a Time Delay with the SIMATIC S7-1200

The software 'Totally Integrated Automation Portal' manages the project and does the programming.

Here, under a uniform interface, the components such as the controller, visualization and networking the automation solution are set up, parameterized and programmed. Online tools are provided for error diagnosis.

In the steps that follow, a project can be opened for the SIMATIC S7-1200, it can be stored under a different name and adapted to the new requirement.

1. The central tool is the 'Totally Integrated Automation Portal'. Here, we call it with a double click (→ Totally Integrated Automation Portal V10)
2. The project "startup" from Module M1 is now opened in the portal view as the basis for this program (→ Open existing project → startup → open)
3. Next, 'First Steps' for the configuration are suggested. We want to 'Open project view' (→ Open project view)
4. First, we want to save the project under another name (→ Project → Save As)
5. Now, 'Save' the project under the new name 'press_timer' (→ press_timer → Save)
6. To make the changes, open the block 'Program Press[FC1]' with a double click (→ Program Press[FC1])
7. Now we can start changing the program. When generating our solution with the delay, we need an ON delay 'TON'. It is located under 'Instructions' in the folder 'Timers'. If you point with the mouse to an object such as the timer of TON, detail info about this object will be provided (→ Instructions → Timers → TON)
8. If you highlight an object and then press the button 'F1' on your PC, online help regarding this object is displayed in a window to the right (→ F1)

---

Note: Here, go to online help and inform yourself in detail about all timing functions.
9. Next, drag the timer ‘TON’ with the mouse to the third contact of the OR function behind the variable ‘#B_Cylinder’ (→ TON → #B_Cylinder)
10. For the timing function we need memory. Here, it can be made available only by generating a new instance data block as a 'single instance' (→ OK)

Note: Multi-instances can be used only when programming within a function block. This will be shown below in the example for the IEC counter.
11. Now, connect the time delay 'TON' with the time base 't#5s' for 5 seconds. By clicking on the project is saved (→ t#5s → Projekt speichern)
12. To load your entire program into the CPU, highlight the folder 'Control Press' and then click on the symbol \(\rightarrow\) Load to device (→ Control Press →).
13. If the CPU is in the 'RUN' mode, you will be asked whether you want to take it to the 'STOP' mode. Confirm with 'OK' (→ OK)

14. If you omitted to specify the PG/PC interface beforehand (refer to module M1, Chapter 4), a window is displayed where you can do this now (→ PG/PC interface for loading → Load)
15. Confirm 'Load' once more. During loading, the status is shown in a window (→ Load)

16. If loading was successful, it is displayed in a window. Now click on 'Complete' (→ Complete)
17. Next, start the CPU by clicking on the symbol $\rightarrow$.

18. Confirm the question whether you actually want to start the CPU with 'OK' $\rightarrow$ OK.
19. By clicking on the symbol Monitoring on/off you can, while the program is tested, observe the status of the timer as well as the time that expired (→).
6. Sample Task for Belt Control with Counter and Multi-Instance

When blocks are to be generated that work in any program like a "Black Box" as it were, they have to be programmed by using variables. In this case, the following rule applies: in these blocks, no absolute-addressed inputs/outputs, flags etc. must be used. Within the block, only variables and constants are used.

If secondary function blocks -or timers/counters- are called from a block that can be used multiple times, they must not be assigned their own data block. The required memory is provided as multi-instance within the instance DB that is assigned to the function block doing the calling.

In the example below, we add a bottle counter to the function block that already contains a belt control dependent on the operating mode.

With this belt, 20 bottles are to be transported to a case. When the case is full, the belt is stopped and the case has to be exchanged.

With the button 'S1', we want to select the operating mode 'Manual', and with the button 'S2' the operating mode 'Automatic'.

In the operating mode 'Manual', the motor is switched on as long as the button 'S3' is operated, whereby button 'S4' must not be operated.

In the operating mode 'Automatic', the belt motor is switched on with button 'S3' and switched off with button 'S4' (break contact).

In addition, there is a sensor 'B0' that counts the bottles into the case. After counting 20 bottles, the belt is stopped.

When a new case is put in place, this has to be confirmed with the button 'S5'.

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 operating mode Manual S1 (make contact)</td>
</tr>
<tr>
<td>%I 0.1</td>
<td>S2</td>
<td>Button operating mode Automatic S2 (make contact)</td>
</tr>
<tr>
<td>%I 0.2</td>
<td>S3</td>
<td>On button S3 (make contact)</td>
</tr>
<tr>
<td>%I 0.3</td>
<td>S4</td>
<td>Off button S4 (break contact)</td>
</tr>
<tr>
<td>%I 0.6</td>
<td>S5</td>
<td>Button S5 (make contact) Reset counter/new case</td>
</tr>
<tr>
<td>%I 0.7</td>
<td>B0</td>
<td>Sensor B0 (make contact) 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 the SIMATIC S7-1200**

The software ‘**Totally Integrated Automation Portal**’ manages the project and does the programming.

Here, under a uniform interface, the components such as controller, visual display and networking of the automation solution are set up, parameterized and programmed. Online tools are provided for error diagnosis.

In the steps below, for the SIMATIC S7-1200 a project can be opened, stored under a different name and adapted to the new requirement.

1. The central tool is the ‘**Totally Integrated Automation Portal**’. Here, we call it with a double click (→ Totally Integrated Automation Portal V10)
2. We now open the project "FB_Belt" from Module M2 in the portal view as the basis for this program (→ Open existing project → FB_Belt → open)
3. Next, 'First Steps' are suggested for the configuration. We want to 'Open project view' (→ Open project view)
4. Now, we first save the project under a different name (→ Project → Save As)
5. Next, 'Save' the project under the new name 'FB_Belt_Counter' (→ FB_Belt_Counter → Save)
6. To set up new global variables, open with a double click on 'PLC Variables(7)' in the 'Control Belt' under 'PLC variables' (open what? (→ Control Belt → PLC Variables → PLC Variables(7)))
7. Next, set up the two global variables 'B0' and 'S5' (→ B0/Bool/%I0.7/Sensor Bottle counter → S5/Bool/%I0.6/Reset counter/new case)
8. To make the changes, open the block 'Belt[FB1]' with a double click (→ Belt[FB1])
9. First, add two lines under Interface for the input variables (→ Interface → Input → Add line)
10. When declaring the local variables, we are adding the following variables.

**Input:**
- sensor_bottle   Sensor bottle counter
- reset_counter   Reset counter/new case
11. Now we can start changing the program. As we generate our solution with the counter, we need a down counter 'CTD'. It is located under 'Instructions' in the folder ‘Counters’. If you point with the mouse to an object such as the counter CTD, you will be provided with detailed information about this object (→ Instructions → Counters → CTD).
12. If you highlight an object and then press the 'F1' key on your PC, online help regarding this object is displayed in a window to the right (→ F1)

Note: Here, go to online help and inform yourself in detail about all counters
13. Now, first insert an AND between OR and assignment, and then drag the counter ‘CTD’ to the second contact of the AND function (→ & → CTD)
14. We need memory for the counter function. Here, the function block makes it available within the instance data block as 'Multi-Instance', without generating a new instance data block (→ Multi-Instance → OK)

**Note:** A multi-instance can be used only when programming within a function block.
15. Now, connect the down counter 'CTD' to the specified value '20' for the 20 bottles and connect the input 'CD' to '#sensor_bottle', and the input 'LOAD' to '#reset_counter'. Next, negate the second contact of the AND function. Click on and the project will be saved

Note: The down counter is most suitable for counting specified quantities, since simply the binary output 'Q' can be used for further connections. Otherwise, a comparator would have to be programmed.
16. Now, open the block 'Main[OB1]' to update the call of block 'Belt[FB1]' (→ Main[OB1])
17. In the block ‘Main[OB1]’, click with the right mouse key on "Belt“ and then on 'Update block call' (→ Main[OB1] → Update block call)
18. Next, select the 'New Interface' and confirm with 'OK' (→ New interface → OK)
19. Click on **Projekt speichern** and the project will be saved (→ **Projekt speichern**).
20. To load your entire program into the CPU, highlight the folder 'Control Belt', and then click on the symbol Load to device (→) Control Belt → )
21. If the CPU is in the 'RUN' mode, you will be asked whether you want to take it to the 'STOP' mode. Confirm with 'OK' (→ OK)

22. Confirm 'Load' once more. During loading, the status is displayed in a window (→ Load)
23. If loading was successful, it is displayed in a window. Click on 'Complete' (→ Complete)
24. Next, start the CPU by clicking on the symbol \( \rightarrow \) (→).

25. Conform the question whether you actually want to start the CPU with 'OK' (→ OK).
26. By clicking on the symbol □ Monitoring on/off, you can observe the counter status while the program is tested (→ □).