Birmingham Metropolitan College

Totally Integrated Automation (TIA)

Automation System S7-1200
Programmable Logic Controllers
CONTENTS

S7-1200 PLC HARDWARE OVERVIEW 3
OPERATING MODES OF THE CPU 5
PROGRAMMING THE S7-1200 6
FIRST PROGRAM 15
USING DIGITAL INPUTS/OUTPUTS 23
PROGRAMMING EXERCISES 29
SET/RESET 33
TIMERS 36
COUNTERS 42
S7-1200 HARDWARE OVERVIEW

Configuring and Operating the SIMATIC S7-1200

Module Spectrum:
The SIMATIC S7-1200 is a modular automation system and offers the following module spectrum:
- Central processing unit (CPU) with different capacity, integrated inputs/outputs and PROFINET interface (for example, CPU1214C)

Power supply PM with input AC 120/230V, 50Hz/60Hz, 1.2A/0.7A, and output DC 24V/2.5A

Signal boards SB for adding analogue or digital inputs/outputs; whereby the size of the CPU does not change (signal boards can be used with the CPUs 1211C/1212C and 1214C).
Programmable Logic Controllers

**Important CPU elements:**

With an integrated voltage supply (24V connection) and integrated inputs and outputs, the S7-1200 CPU is ready, without additional components.

To communicate with a programming device, the CPU is equipped with an integrated TCP/IP port. By means of an ETHERNET network, the CPU is able to communicate with HMI operator devices or other CPUs.

1. Connection 24V
2. Plug-in terminal block for user wiring (behind the cover plates)
3. Status LEDs for the integrated I/O and the CPU’s operating mode
4. TCP/IP connection (on the lower side of the CPU)

The **SIMATIC Memory Card (MC)** stores the program, data, system data, files and projects. It can be used for the following:

- Transferring a program to several CPUs
- Firmware update of CPUs, signal modules SM and communication modules CM
Operating Modes of the CPU

The CPU has the following operating modes:

- In the operating mode **STOP**, the CPU does not execute the program, and you can load a project.
- In the operating mode **STARTUP**, the CPU performs a start-up.
- In the operating mode **RUN**, the program is executed cyclically. Projects cannot be loaded in the CPU’s RUN mode.

The CPU does not have a physical switch for changing the operating mode. The operating mode (**STOP** or **RUN**) is changed by using the button on the operator panel of the software STEP7 Basic.

In addition, the operator panel is provided with the button **MRES** to perform a general memory reset and displays the status LEDs of the CPU.

The colour of the **status LED RUN/STOP** on the front of the CPU indicates its current operating mode.

- **Yellow** light indicates the **STOP** mode.
- **Green** light indicates the **RUN** mode.
- **Flashing** light indicates the **STARTUP** mode.

In addition, there are LEDs for **ERROR** to indicate errors and **MAINT** to indicate that maintenance is required.
Programmable Logic Controllers

Programming 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, visualisation and networking the automation solution are set up, parameterised and programmed.

Online tools are provided for error diagnosis.

The software 'Totally Integrated Automation Portal' has two different views:

- the portal view
- the project view.

Portal View

The portal view provides a task oriented view of the tools for processing the project. Here, you can quickly decide what you want to do, and call the tool for the respective task. If necessary, a change to the project view takes place automatically for the selected task. Primarily, getting started and the first steps are to be facilitated here.

You will need to select 'Create new project' first.

Put in a project name for example ‘Project 1’ then ‘Create’
Then the software will start creating a project.

Then we get a Portal View as shown below.
Programmable Logic Controllers

Select Configure a device and we will see the screen shown below

Add new device
Now open SIMATIC S7-1200 and then CPU

You will now need to select the type and model of your PLC as shown above. Once you have made your selection just click on 'add'
We now need to setup the communications link between the PLC and the PC.

The PLC **must** have power connected to the module and an Ethernet cable connected between the PLC and the computer.

We now need to follow this procedure.

1. Select ‘Control Panel’ from the start menu

2. Select ‘Setting the PG-PC Interface’
3. Then select ‘ISO Ind. Ethernet’ and select ‘OK’.

**Beware** there are two ISO Ind. Ethernet, select the family PCI and not the wireless.
4. Now go back to the project view and select ‘Online access’ from the project tree.


6. The accessible device should now be shown with its own ID.

7. We now need to make that connection between the PLC and the PC. Now select ‘Portal View’ and open Online and Diagnostics.
Programmable Logic Controllers

8. Select ‘Accessible devices’

9. Now we can connect our PLC to the PC.

10. Now select the PN/IE from the type of the PG/PC interface window as shown below
Programmable Logic Controllers

11. Then select the PG/PC interface as shown

12. Then the software will try to connect the two devices

13. Once it’s found the accessible device and has placed the necessary information in the column you can test the connection by Flashing the LED’s at the front of the module

14. Now select ‘Show’ and you can now start programming.
Programmable Logic Controllers

First Program
Open from the Project Tree ‘PLC_1’
then open Program blocks
then Main[OB1] Organizational Block

This will now open what’s called a Network 1. This is where your instructions will sit.
These instructions will tell the PLC what to do, to control what to turn ‘ON’ and what to turn ‘OFF’
Programmable Logic Controllers

Now move your cursor from the mouse and hover over each symbol and write what each icon means below:

Now your first program will use the following symbols

For the program to work first me must give them an address. This is so the PLC will now which device to turn ‘On’ or ‘OFF’.

Address as I0.0 (the letter I and the number zero, dot, zero), and
We can associate each symbol with a Tag.

Open PLC tags and then double click on ‘Show all tags’ the above window opens.

We can add the tag names as shown below. Each tag will have a Data Type and an Address.

Once the tag name is given we can select the Data type, in this instance we have chosen Bool (short for Boolean). Then we allocate the correct address, whether it’s an input or an output.
You can set the tags before you start programming if you want. Then all you will need to do is select the correct tag for the symbol.
Programmable Logic Controllers

Now you have completed your first program, you will now have to download the program from the PC to the PLC.

Select 'Download to device'

Then you will have to remind the PLC what PG/PC interface you have selected before.
Programmable Logic Controllers

If no accessible device is shown once you have set up the interface then you will have to select ‘Show all accessible devices’

Once the scan is complete then select ‘Load’

It will now ask you to assign IP address select ‘Yes’
Programmable Logic Controllers

Now you will get a new screen select ‘Load’
Select ‘Start all’ then ‘Finish’

Now you have successfully downloaded the program into the PLC. You will now notice that the menu bar is now orange and the Network line has changed green. You can now turn the Start switch on and observe what happens next.
Programmable Logic Controllers

Exercise 2

Place an input after the output component, address it then verify the program. Record your findings below.

Laboratory Exercise 2 Using Digital Inputs/outputs

What You Will Learn

After completing this exercise, you will be able to . . .

- Write a program using two different methods for a Start / Stop Circuit
Before You Begin

**BIT**: A Digital 0 or 1 in Binary. A bit may represent the state of, ON or OFF, of a discrete I/O device

**Examine If Closed**: Examine a bit for an ON (1) condition.

If the Bit is ON, the instruction is true.

**Examine If Open**: Examine a bit for an OFF (0) condition.

If the Bit is OFF, the instruction is true.

**Output Energise**: If the rung is TRUE, turns ON a bit.

When the rung goes False the bit turns OFF.

**Output Latch**: If the rung is TRUE, turns ON a bit.

When the rung goes False the bit remains ON.

**Output Unlatch**: If the rung is TRUE turns OFF a bit.
Here’s How

Using the following examples your instructor will demonstrate...

**Normally Open Contact**

\[ I \ 0.0 \]

**Normally Closed Contact**

\[ I \ 0.1 \]

**Output**

\[ Q \ 0.0 \]

**Output Reset**

\[ Q \ 0.1 \]

**Output Set**

\[ Q \ 0.2 \]
In this exercise you will . . .

**Exercise**

*Directions:*

**Start a New Program**

**Define the I/O Yourself**

1. Enter the program in Fig 1 and test that it works.
2. Modify the program to include the lines shown in figure 2.
3. Test the modification.
4. Modify the program to include the lines shown in figure 3.
5. Test the modification
6. Write program to switch on and off output 0 using input 0.

   Record your diagram and findings
Programmable Logic Controllers

Figure 1
Modify the program in figure 1 with the component shown in figure 2.

Figure 2
Verify program then switch input on and off and record the results below.

What happens when you switch I 0.0 off? ..............................................

Modify the circuit as shown below in figure 3.

Figure 3
Exercise 3

What You Will Learn

After completing this lesson, you will be able to . . .

- Write a program using switches to operate as logic gates

Before You Begin

Logic Gates Logic gates are AND, OR, NAND, NOR and EX OR gates.

Exercise 3

In this exercise you will . . .

Directions:

Start a New Program

1. Write programs to answer the questions
2. Draw the circuit diagram into your booklet
Write a program to perform the following:

Use the notes provided to answer the following. In each case draw up a truth table to assist you.

1. Design a ladder logic circuit that acts as a simple AND gate.

2. Design a ladder logic circuit that acts as a simple OR gate.
Programmable Logic Controllers

3. AND gate circuit #2. Using two switches, turn on a RED light if BOTH are 0, and a green light if BOTH are 1. Draw the ladder logic circuit.

4. OR gate circuit #2. Using two switches turn on a red light if BOTH are 0, and green light if EITHER is 1. Draw the ladder logic circuit.
Programmable Logic Controllers

5. NAND Gate. To turn OFF a light if both switches are 1, otherwise the light is ON. Draw the ladder logic circuit below. There are two ways to solve this – one uses an AND circuit and the other uses an OR circuit. Show both circuits.

6. NOR Gate. To turn on a light if both switches are 0. There are again two ways to solve this problem using AND and OR circuits – show both.
Programmable Logic Controllers

7. EX OR Gate. To turn on red light if 0, and green light if 1. Draw the ladder logic circuit below.

8. STRETCH QUESTION Write a ladder logic program to complete the truth table for each light below

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>RED</th>
<th>AMBER</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Set and Reset Outputs

---(R)---
---(S)---

Set and Reset are retentive output instructions. Set can only turn ‘on’ a bit, while Reset can only turn off a bit. These instructions are usually used in pairs, with both instructions addressing the same bit.

Your program can examine a bit controlled by Set and Reset instructions as often as necessary.

**Warning**

**Under fatal error conditions, physical outputs are turned off. Once the error conditions are cleared, the controller resumes operation using the data table value of the operand.**

Using Set instruction

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

Using Reset instruction

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

**Exercise 3**
Programmable Logic Controllers

1. Write a program that will first turn on the RED light using Set instruction. Then second turn on the Green light and turn off the RED light using Set and Reset instructions. You will use two inputs to carry out the task. Draw your circuit below.

2. Adjust your program in Q1 where you will use only ONE input to do the operation. Draw your circuit diagram below.
3. Redesign the AND Gate circuit to turn on red light if 0, and green light if 1 using Set and Reset instructions. Draw the ladder logic circuit below.

4. Write a ladder logic program to complete the truth table for each light below using the Set and Reset instructions. For each line only the stated colours must be on.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>RED</th>
<th>AMBER</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Programmable Logic Controllers

Timers (TON)

Description

The "Generate on delay" instruction is used to delay the setting of the Q output for the configured time duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). The programmed time duration PT begins when the instruction starts. When the time duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new rising edge is detected at the start input.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TON that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIME)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruction to the next call of the instruction.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:
Programmable Logic Controllers

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Memory area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Start input</td>
</tr>
<tr>
<td>PT</td>
<td>Input</td>
<td>TIME</td>
<td>I, Q, M, D, L or constant</td>
<td>Duration of the on delay. The value of the PT parameter must be positive.</td>
</tr>
<tr>
<td>Q</td>
<td>Output</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Output that is set when the time PT expires.</td>
</tr>
<tr>
<td>ET</td>
<td>Output</td>
<td>TIME</td>
<td>I, Q, M, D, L</td>
<td>Current time value</td>
</tr>
</tbody>
</table>

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:

![Pulse diagram image]
Programmable Logic Controllers

To use the Timer instruction, you will need to select from Basic instruction, Timer operations and select/drag TON onto the network as shown below.

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)
Programmable Logic Controllers

As from the table above PT is where we program the time required for the time delay.

An output coil can now be added at the end of the network.

Exercise

Write a program using I 0.0 as an input, TON with a 10S delay to turn on output Q 0.0.

Observe what happens.
Rewrite your program and add Timer TON to each output and write program below.

Design a traffic light program using Timer TON where Q0.0 controls the red light, Q0.1 controls the amber light and Q0.2 controls the green light.
Programmable Logic Controllers

Add to your traffic light program the sequence Red, Red and Amber, Green, Amber and back to Red for the circuit to work continuously.
Programmable Logic Controllers

Counters

CTU Count Up

The instruction "Count up" counts up the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction executes and the current count value at the CV output is incremented by one. When the instruction executes for the first time, the current count value at the CV output is set to zero. The count value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is decided by the parameter PV. If the current count value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Parameters

The following table shows the parameters of the "Count up" instruction:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Memory area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Count input</td>
</tr>
<tr>
<td>R</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Reset input</td>
</tr>
<tr>
<td>PV</td>
<td>Input</td>
<td>Integers</td>
<td>I, Q, M, D, L or constant</td>
<td>Value at which the output Q is set.</td>
</tr>
<tr>
<td>Q</td>
<td>Output</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Counter status</td>
</tr>
<tr>
<td>CV</td>
<td>Output</td>
<td>Integers</td>
<td>I, Q, M, D, L</td>
<td>Current count value</td>
</tr>
</tbody>
</table>
Programmable Logic Controllers

The following example shows how the instruction works:

To see how the up counter operates copy the diagram below:

Address the inputs and outputs with the correct addresses and type in 5 at PV input.

Turn on and off the switch connected to CU and observe what happens when you switched it on and off for five times.

Now design a program that will reset the counter 10 seconds after CTU becomes true

Draw circuit below.
Programmable Logic Controllers

CTD Counter down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction executes and the current count value at the CV output is decremented by one. When the instruction executes the first time, the count value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the count value is decremented until it reaches the low limit value of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current count value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Parameters

The following table shows the parameters of the "Count down" instruction:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Memory area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Count input</td>
</tr>
<tr>
<td>LD</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Load input</td>
</tr>
<tr>
<td>PV</td>
<td>Input</td>
<td>Integers</td>
<td>I, Q, M, D, L or constant</td>
<td>Value at which the output Q is set.</td>
</tr>
<tr>
<td>Q</td>
<td>Output</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Counter status</td>
</tr>
<tr>
<td>CV</td>
<td>Output</td>
<td>Integers</td>
<td>I, Q, M, D, L</td>
<td>Current count value</td>
</tr>
</tbody>
</table>
Programmable Logic Controllers

The following example shows how the instruction works.

To see how the down counter operates copy the diagram below:

Address the inputs and outputs with the correct addresses and type in 5 at PV input.

Turn on and off the switch connected to CU and observe what happens when you switched it on and off for five times.

Now design a program that will reset the counter 10 seconds after CTD becomes true.

Draw circuit below.