First Steps with STEP 7 Lite V3.0

SIMATIC
STEP 7 Lite V3.0
Edition 04/2004

simatic
STEP 7 Lite

SIEMENS

SIEMENS

SIMATIC Software

First Steps with STEP 7 Lite V3.0

Getting Started

04/2004
A5E00293886-01

Safety Guidelines

This manual contains notices intended to ensure personal safety, as well as to protect the products and connected equipment against damage. These notices are highlighted by the symbols shown below and graded according to severity by the following texts:

Danger
indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.

Warning
indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.

Caution
indicates that minor personal injury can result if proper precautions are not taken.

Caution
indicates that property damage can result if proper precautions are not taken.

Notice
draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only qualified personnel should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:

Warning
This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.
This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Trademarks

SIMATIC®, SIMATIC HMI® and SIMATIC NET® are registered trademarks of SIEMENS AG.
Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

Siemens AG
Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nuernberg

Siemens Aktiengesellschaft

A5E00293886

Welcome to STEP 7 Lite

... the SIMATIC software for generating PLC programs in LAD, FBD or STL for SIMATIC S7-300 (including SIMATIC C7), ET 200S and ET 200X. STEP 7 Lite is designed for the newcomer to SIMATIC as well as for the user editing projects offline.
You will need basic STEP 7 software or STEP 7 Professional if you want to implement a SIMATIC S7-400 PLC, distributed I/O, CP communication modules, FM function modules, or systems consisting of more than one CPU.

Information on STEP 7 Lite

STEP 7 Lite is a software not only designed for newcomers, but also for the expert who primarily programs medium performance systems. With STEP 7, programs created in STEP 7 Lite can be imported/exported for further use. Compared to STEP 7, we went new ways in designing the user interface. Enhanced Explorer functions, transparent project overviews and the usual Windows operating philosophy, all of which will offer optimal support to you for getting started and working with our SIMATIC Software.

Information on this Getting Started

Here you will get to know the basics of STEP 7 Lite. We shall guide you through practical exercises introducing you to essential on-screen dialogs and operating procedures, prepared in such a way that you can start at almost any chapter. Descriptions and operating procedures you should refer or which you must follow are highlighted in red color. Brief excursions to associated topics are referenced in blue color.

Prerequisites for working with this Getting Started

What you need to work through the practical STEP 7 Lite exercises in this Getting Started:
- a SIMATIC PG or a PC,
- the STEP 7 Lite software package and the authorization disk,
- a SIMATIC S7-300 PLC.

Please note the Order No. table in Chapter 1.

Further Documentation

- After installation of STEP 7 Lite, select **Start > Simatic > Documentation** on your CD to open and print the electronic manual "Programming with STEP 7 Lite".

Have lots of fun and success!

SIEMENS AG

Overview of the Getting Started Sample Projects

After installing your STEP 7 Lite software, unless you have selected another directory, go to <Drive>:\Siemens\S7lite\Examples\English ... to find the programming samples to follow.

This Getting Started refers to these samples:

- **first_stepd_stl.k7p**
- **first_steps_fdb.k7p**
- **first_steps_lad.k7p**

All sample programs are identical, differing only in the programming language you choose to work with.

Contents

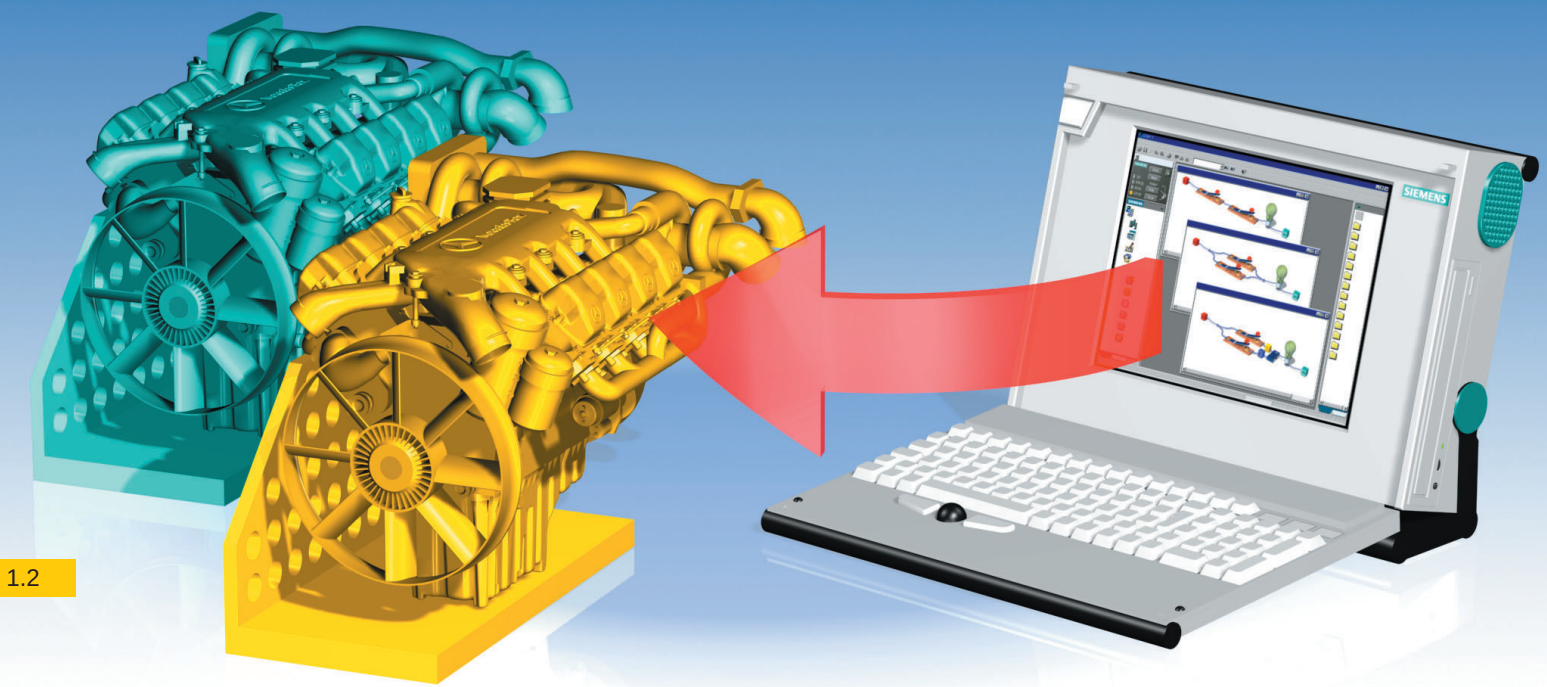
Part 1: Getting started with STEP 7 Lite - Essential basics

Part 2: How to develop an automation solution with STEP 7 Lite

Part 3: Downloading, Testing and Diagnosing

1 Overview and installation

What are you going to learn?



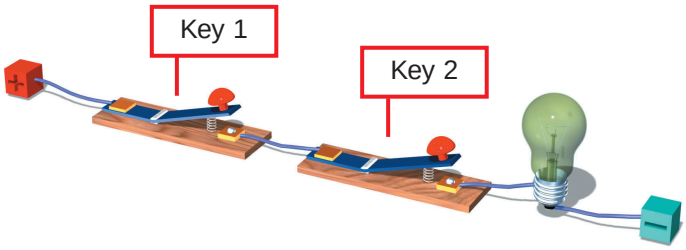
Practical exercises in this manual will show you how easy it is to handle the programming languages LAD (Ladder logic), FBD (Function block diagram) and STL (Statement list) in STEP 7 Lite.

Start by creating a project. Name it “Getting Started”.
Next, you will create a PLC program in this project, using the simple binary logical operations AND, OR, MEMORY CIRCUIT.

You are then going to enhance this PLC program to operate a motor testing bench.

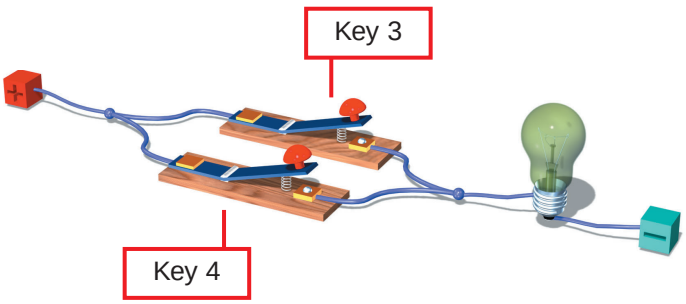
Basic know-how

Our programming examples are based on three fundamental, binary logical operations:



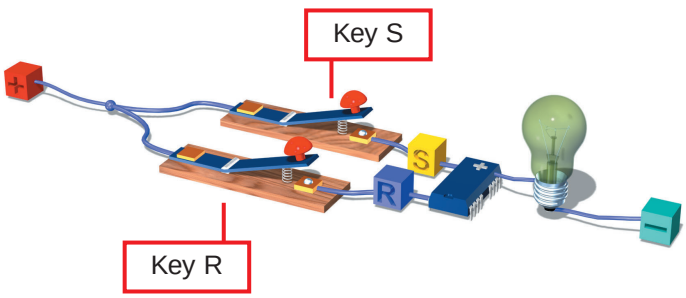
Series circuit
The first binary logical operation you are then going to program is an AND function. The AND function can be demonstrated by an electrical circuit that is equipped with two pushbuttons.

The lamp is lit when pushbutton 1 **AND** 2 are pressed.



Parallel circuit
The second binary logical link is the OR function which can also be demonstrated in an electrical circuit.

The lamp is lit when pushbutton 3 **OR** 4 is pressed.

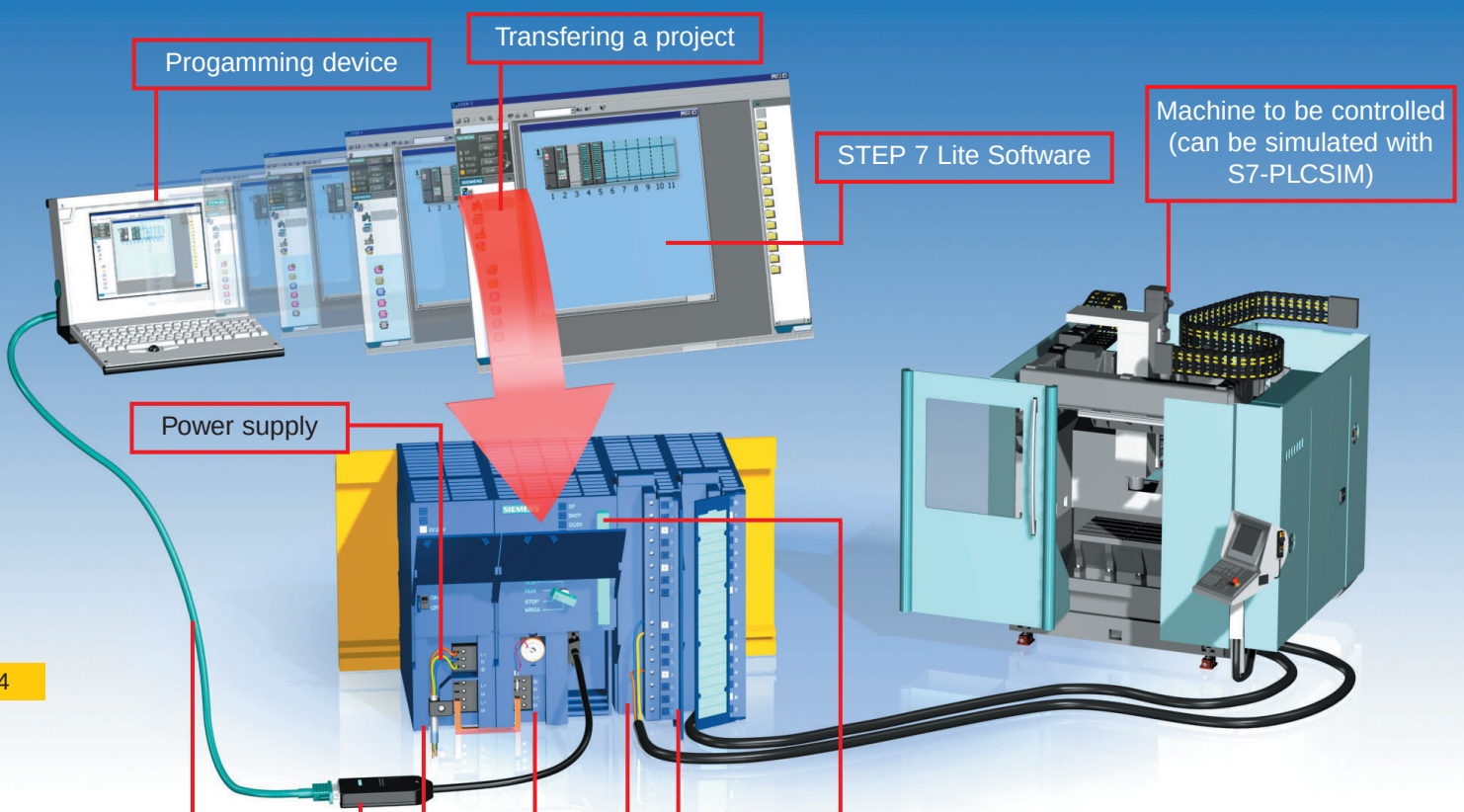


Memory circuit
(Set/reset flipflop)
The third binary logical operation is the memory circuit. In an electrical circuit it responds to certain voltage levels and outputs these accordingly.

Press pushbutton S. The lamp remains lit **until** pushbutton R is pressed.

You are going to program all three binary logical operations to form a practical sample circuit – a motor bench. You will learn how to handle following STEP 7 Lite program elements: Organization blocks, function blocks, instance data blocks, functions, global data blocks.

Interaction between hardware and software



Using the STEP 7 Lite software, you are going to create your user program in a **project**.

The programmable controller (PLC) consists of a **rack**, a **power supply**, a **CPU** and **input** or **output modules**.

The PLC monitors and controls your machine with the help of the user program.
I/O modules are addressed in the user program via I/O addresses.

Component Checklist

You require the following components to create the sample project described above.

PLC station (1)

	Order numbers:
Power supply (PS 307 2A)	6ES7307-1BA00-0AA0
CPU (CPU 315)	6ES7315-1AF03-0AB0
Digit input (SM32DI 16xDC24V)	6ES7321-1BH02-0AA0
Digital output (SM322 DO 16xDC24V/0,5A)	6ES7322-1BH01-0AA0
Backup battery (Li) 3,4V	6ES7971-1AA00-0AA0
Profile rail 480 mm	6ES7390-1AE80-0AA0

STEP 7 Lite lets you program components of the S7-300, ET 200S and ET 200X series. The modules used in the sample project are listed in brackets. Of course, you are free to use other modules of these series.

Computer (2)

SIMATIC PG Power PG, Field PG or Commonly available PC with CP 5611	www.ad.siemens.de/ simatic-pg
Operating system Windows 2000 Windows XP Home or Professional Edition Internet Explorer as of V6.0	or

We recommend you use our SIMATIC PGs. These units can withstand harsh industrial environments.
You will need an additional interface cable if you decide to use a commonly available PC. This interface is already integrated in SIMATIC PGs.

Software (3)

Software STEP 7 Lite (Floating License)	6ES7810-3CC07-0YA5

Notes on installation are found on the CD, in STEP7Lite\Disk1\README.WRI.

Documentation (4)

First Steps with STEP 7 Lite V3.0

This ”First Steps with STEP 7 Lite” manual is supplied with a software CD that also contains the electronic manual ”Programming with STEP 7 Lite” and the Online Help.

Options package (5)

Simulation software S7-PLCSIM (Floating License)	6ES7841-0CC04-0YA5
Simulation software S7-PLCSIM (Upgrade)	6ES7841-0CC04-0YE5

S7-PLCSIM simulates a connected PLC. S7-PLCSIM is helpful if you want to run a program test without having local access to hardware.

STEP 7 Lite

Configuring

Designing the solution of an automation task
chapter 3

Creating a project

chapter 4

Configuring the hardware

chapter 4

Creating a program

chapter 5 – 9

Transfering program to CPU

chapter 10

Testing the program

chapter 11

A project represents the central element in STEP 7 Lite. Within this project you solve all your automation tasks – starting at the hardware configuration and working your way to the program test run.

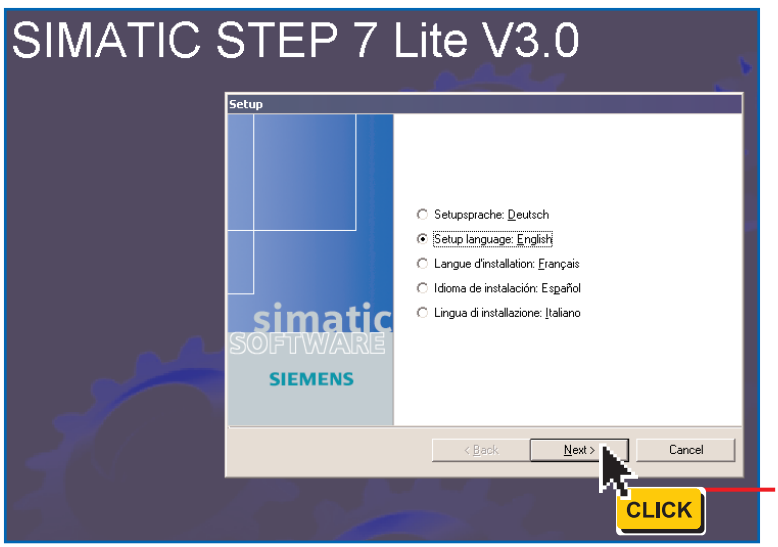
We recommend you configure your hardware first before you run large programs with many I/Os. In this case, you will have the advantage that STEP 7 Lite displays available addresses in Hardware Configuration.

When you choose to start by writing the program, you would rather have to determine available addresses by yourself according to the selected component, as in this case you could not call them via STEP 7 Lite.

Hardware Configuration not only lets you specify addresses, but also allows you to edit module parameters and characteristics.

If you prefer to start programming right away you can skip hardware configuration, as this ”Getting started” project requires only very few I/Os.

Installing STEP 7 Lite



For the installation you need:

– the STEP 7 Lite CD containing the installation instructions in **STEP7Lite\Disk1\Readme.WRI,**

and

– the corresponding license key (user authorization).

Insert the STEP 7 Lite CD.
The installation programm is started automatically or via **drive>:\setup.exe.**

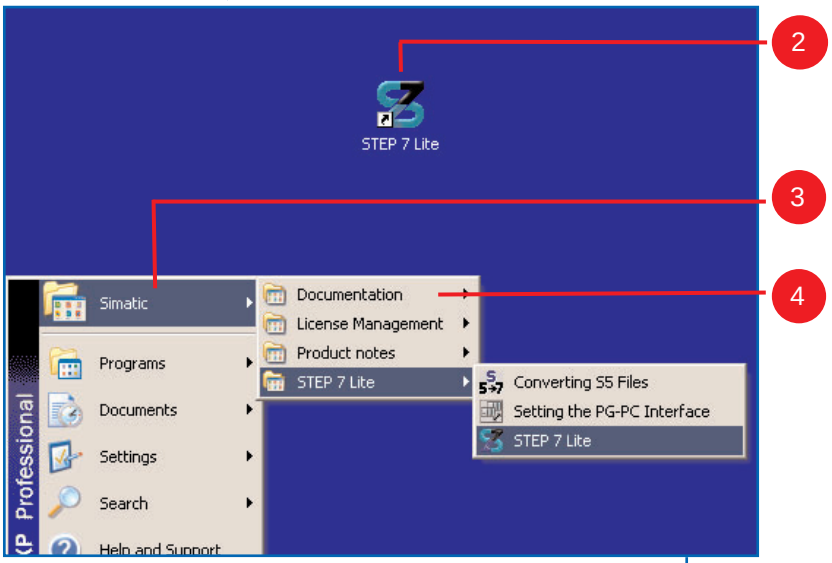
Follow the installation instructions.

When prompted to do so, insert the data carrier containing the license key.

Follow the on-screen instructions for installing the license key.

Remove the data carrier before restarting the computer.

Desktop after installation



After installation is completed, STEP 7 Lite will be displayed on the desktop and in the Start menu.

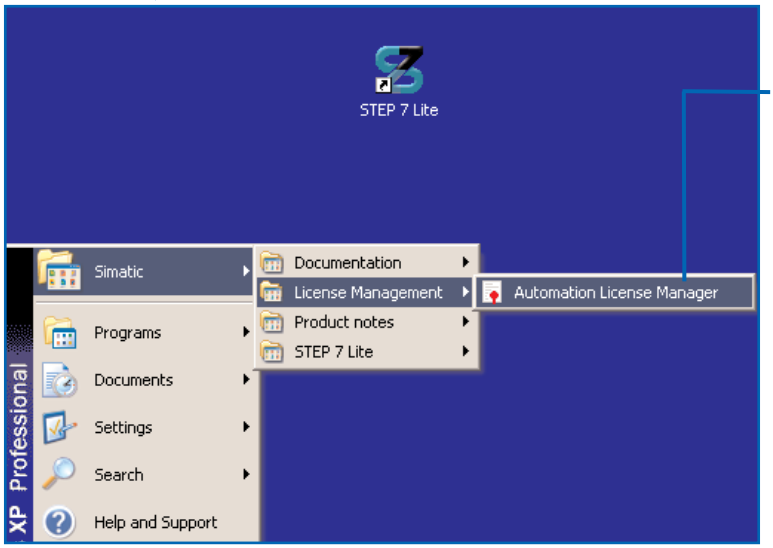
Any additional SIMATIC software you install can be called via this SIMATIC directory.

You can find the printable STEP 7 Lite documentation under **Simatic > Documentation**.

Transfer license keys

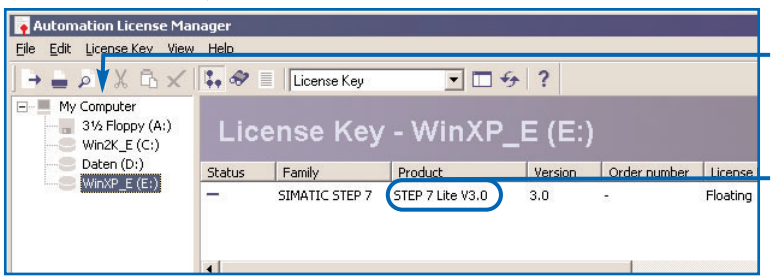
If no valid license key is installed for STEP 7 Lite, a trial license key is used, which is supplied and installed by default together with STEP 7 Lite. However, STEP 7 Lite can only be used for 14 days with this license key. When STEP 7 Lite is started the first time without a valid license key, the trial license is activated.

To transfer the license key from one computer to another, proceed as follows:



5 Start the Automation License Manager.

User Interface for the Automation License Manager



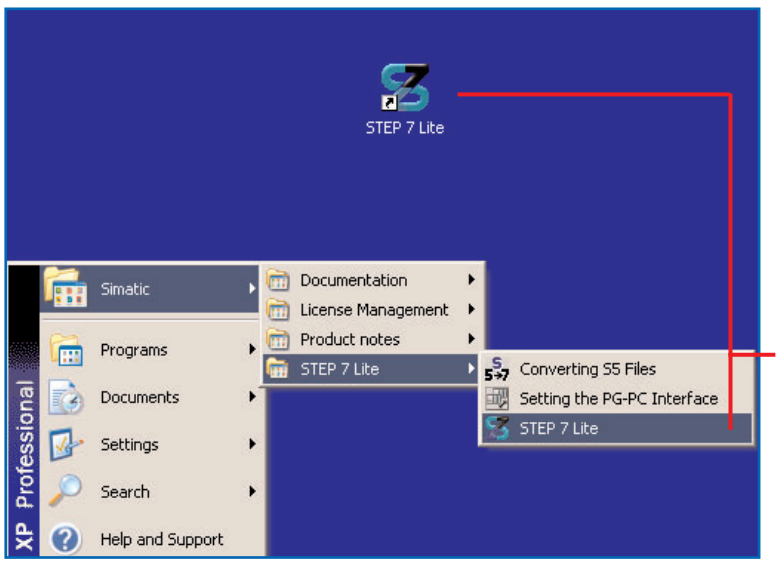
6 Access the drive on which the license key to be transferred is located.

7 Select the license key, and then select the **License Key > Transfer** menu command.

In the dialog that is then displayed, on the target computer select the drive to which you want to transfer the license key.

2 Starting and operating

Opening the sample project

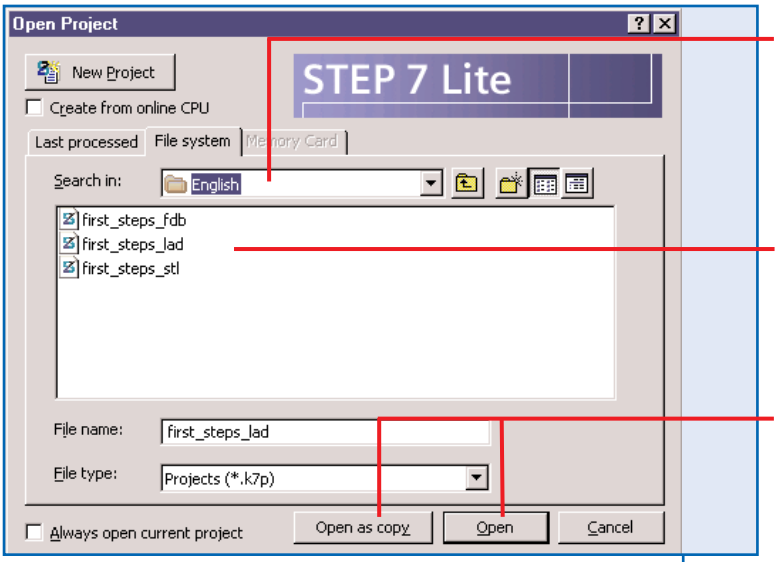


How to open the sample project in LAD

STEP 7 Lite is installed on your computer.

This chapter contains the most important information relating to the user interface.

Start STEP 7 Lite via Start menu or desktop icon.



2 Go to the sample program directory

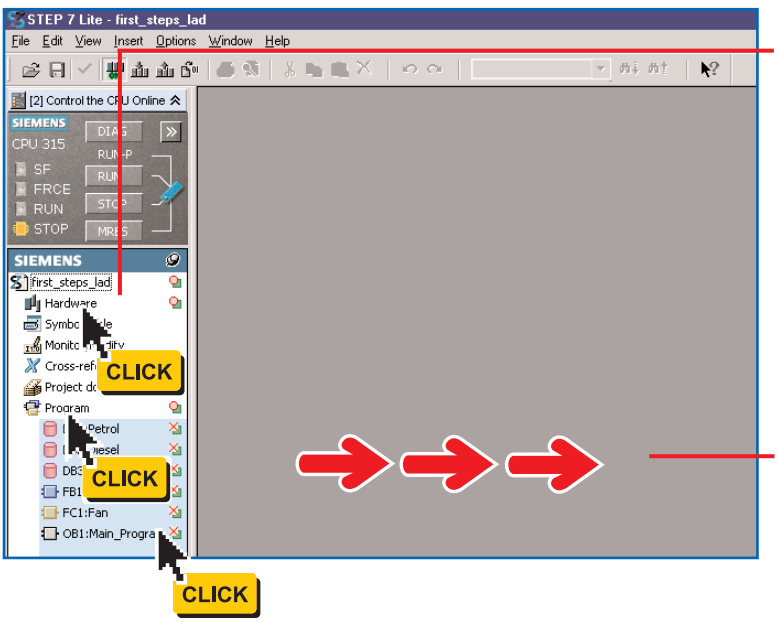
<Drive>:\Siemens\S7Lite \Examples\English ...

3 Select the LAD sample file.

first_steps_lad.k7p

4 Open the project file.
Open the project as a copy to avoid overwriting the sample project supplied by mistake.

The selected project is opened

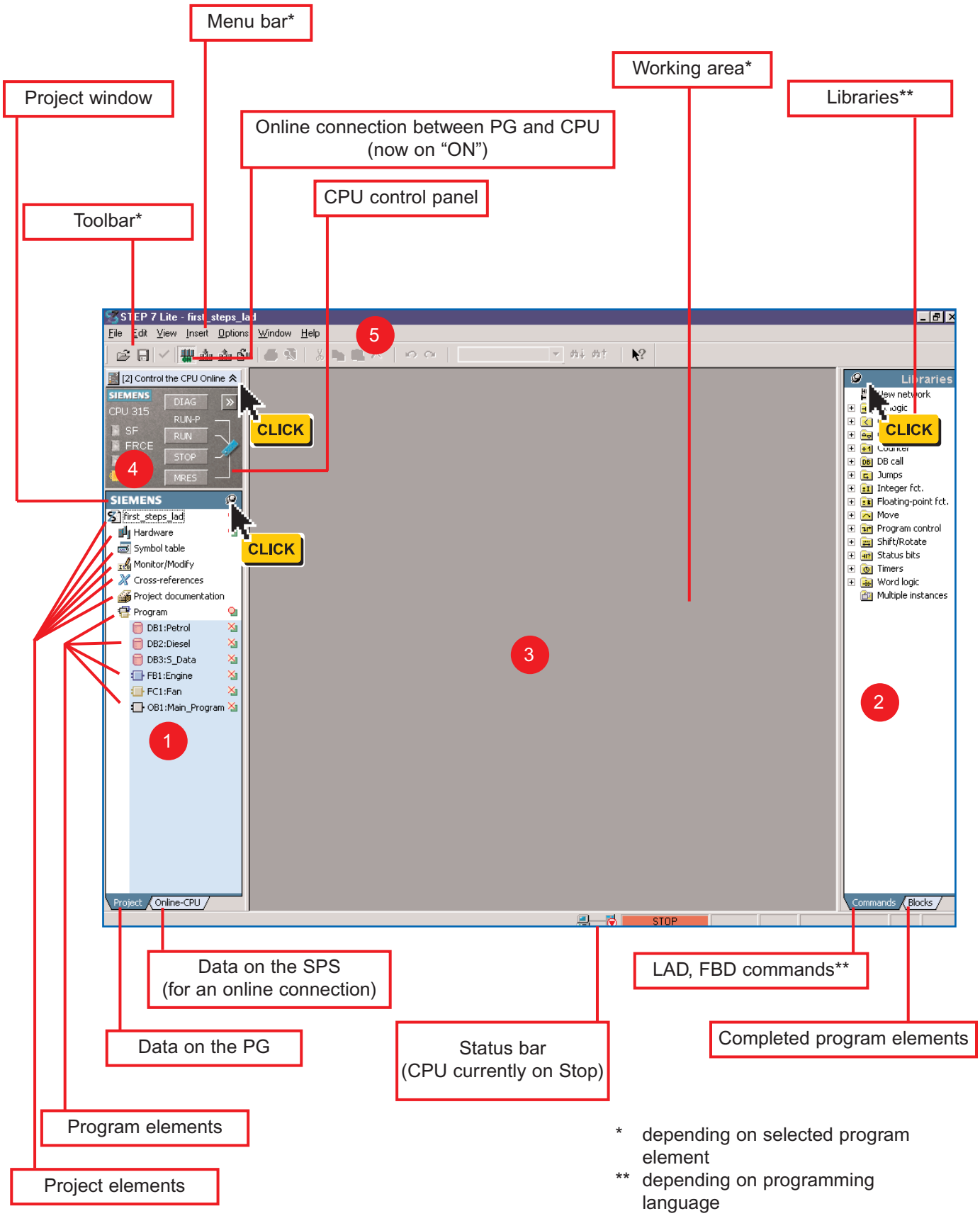


5 The project window displays "first_steps_lad".

The left project window represents the central navigation tool of STEP 7 Lite. From here, you can open all STEP 7 Lite views via the project elements "Hardware", "Symbol table" etc.

6 Double-click on each elements. The views are opened in the gray working area, while the menu bar on top is adapted to the respective view.

Double-click on the project elements to open all STEP 7 Lite views one after the other. Close any windows not required anymore in order to maintain your overview.



* depending on selected program element

** depending on programming language

The user interface

The user interface is split into five areas:

1. Project window
   All project elements you require are already created when you generate a new project.

2. Libraries
   The included blocks are found under ”Libraries”. LAD and FBD block instructions are found under ”Commands”.

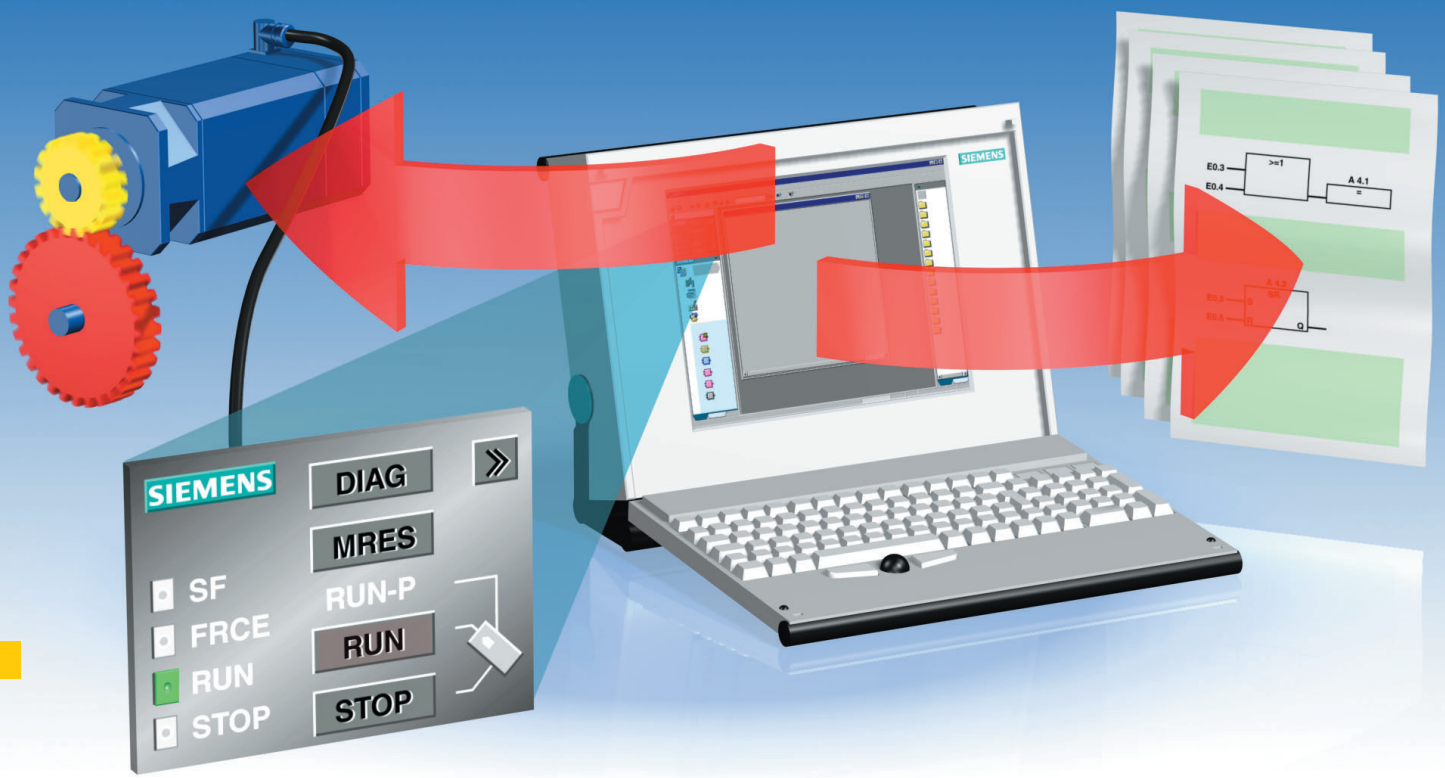
3. Working area
   The views in which you can edit your project can be opened here.

4. CPU operator panel
   Represents the CPU front panel with its displaying and operating elements. Lets you change operating states.

5. Menu bar
   Contains all menus available in STEP 7 Lite – e.g. with opened block, menu command **View > LAD** for changing the programming language.

Click on the expansion icon to show or hide the CPU operator panel. Click on the pin needle to lock or unlock the view of the project window and libraries. When unlocked, you can increase or reduce the size of the working area by dragging it with the mouse pointer towards the edge.

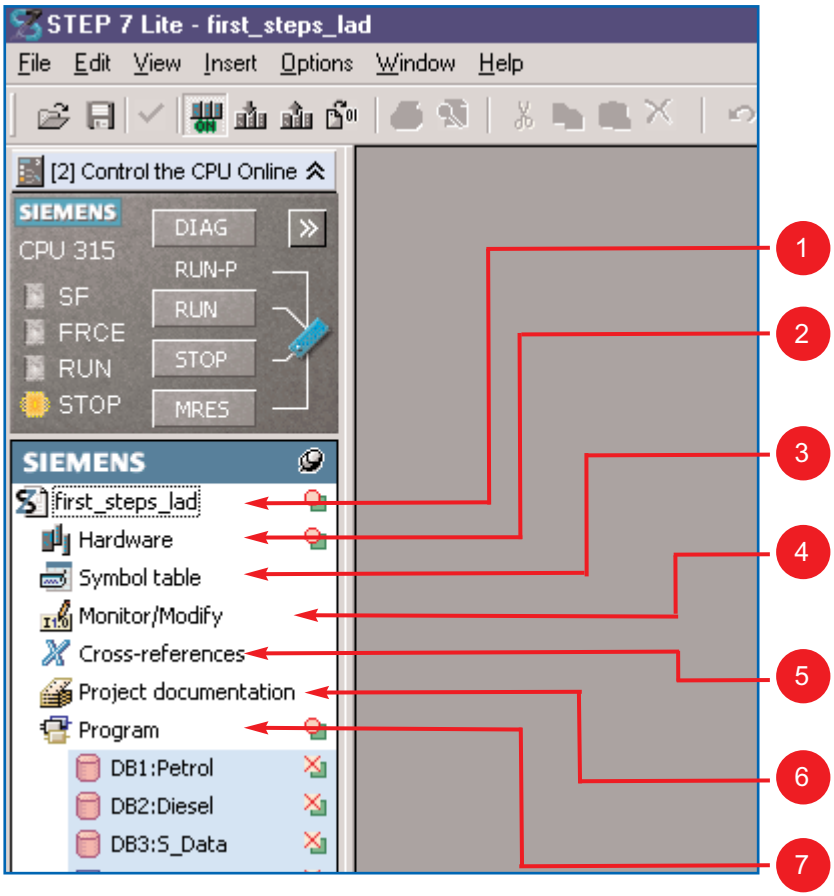
Project handling



What is a project?

The project data of a STEP 7 Lite project includes all data of a SIMATIC S7-300, C7 or of a modular Distributed I/O System ET 200X or ET 200S (stand-alone).

Projects serve the purpose of saving all data acquired during the creation of an automation solution in a managed file system.



Project handling

Project elements are linked to the following tasks:

1. Creating and saving a project
2. Hardware configuration, module parameter assignment and hardware error diagnostics
3. Specifying symbols for symbolic programming
4. Running program tests, monitoring, controlling and forcing addresses in the CPU
5. Evaluation of the program structure and addresses used
6. Individual arrangement of program documentation
7. Using blocks to create an SPS user program.

File handling

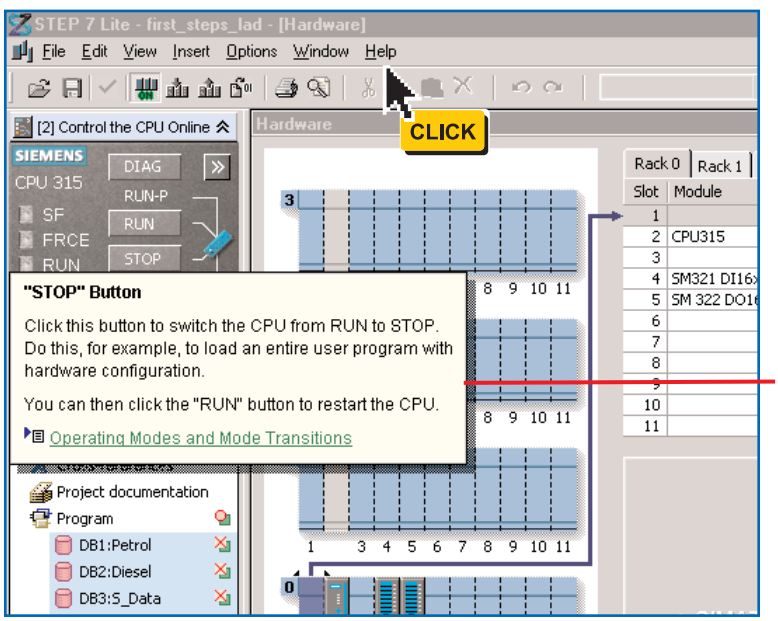
Save the project under its name and file format **... .k7p**.

STEP 7 Lite lets you open only one instance of a **.k7p file**.

Risk of accident – When there is an online connection between the PG and CPU, you can use the CPU operator panel to trigger motions in a plant, for example.

Thus, never select ”RUN” if you cannot entirely exclude personal risk.

Calling help functions



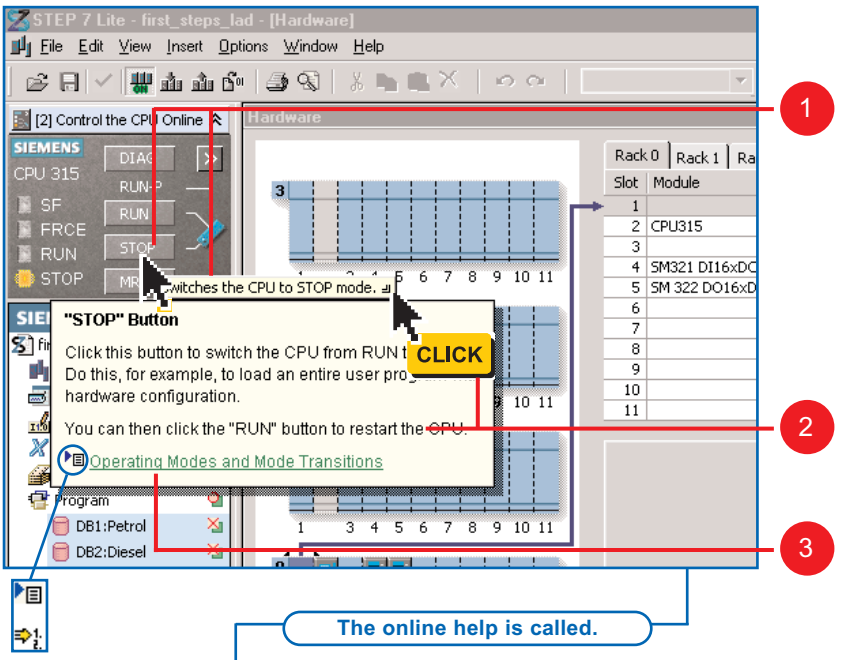
Windows Help

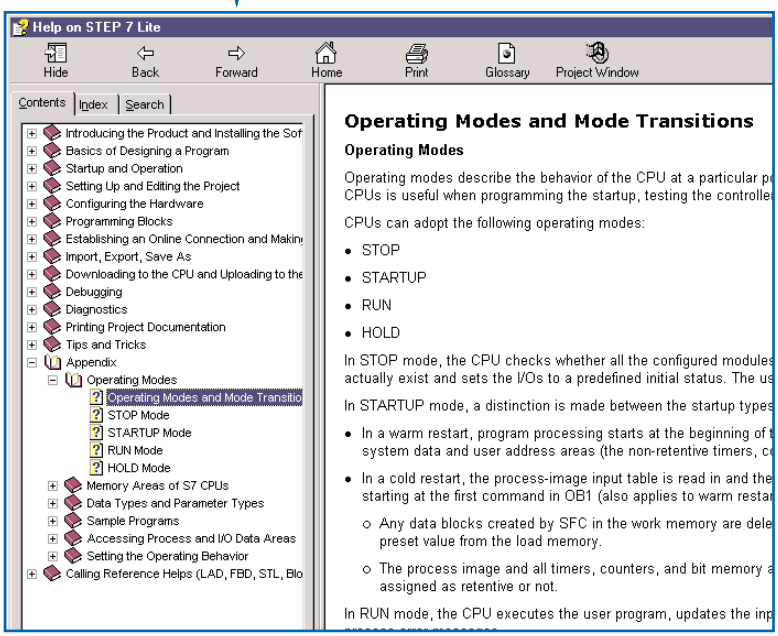
You will find it easy to handle the STEP 7 Lite Help system if you have previously worked with Microsoft programs.

1 Press **F1**:
The detailed basic help pops up.

2 Press **Shift + F1**:
Then, position the question mark cursor on a button and click it to open direct help on this button.

You can also choose to access these two help systems via **Help** in the menu bar.





The three Help sections

Quick help

Without clicking, position the cursor on the **STOP** button, for example.

A quick help on the button is displayed when you position the cursor on the button and leave it there for a moment.

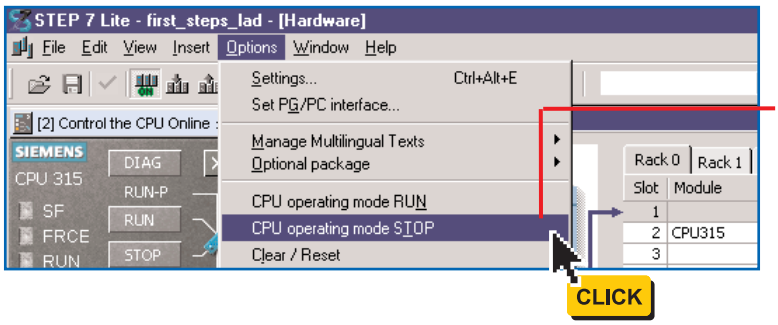
Direct help

Click on the small arrows to open direct help as well.

Basic help

Click on the link. Detailed basic help on the selected topic pops up in a separate window.

Note:
The different link icons identify the type of help called in the basic help.
Leaf = Background information
List = Handling instructions

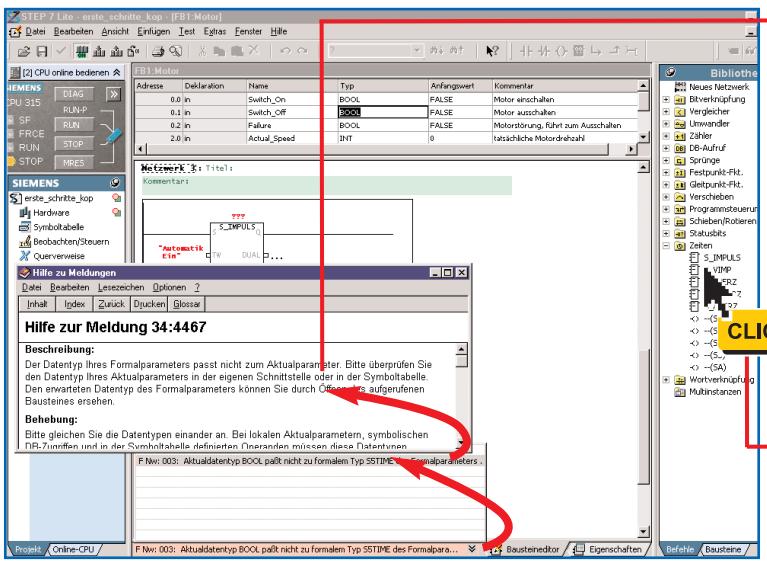


1

Further help

Help on menu commands

Press **SHIFT** + **F1** to open a pull-down menu. Click on a menu command. The help on this command is displayed.



2

Help on error messages

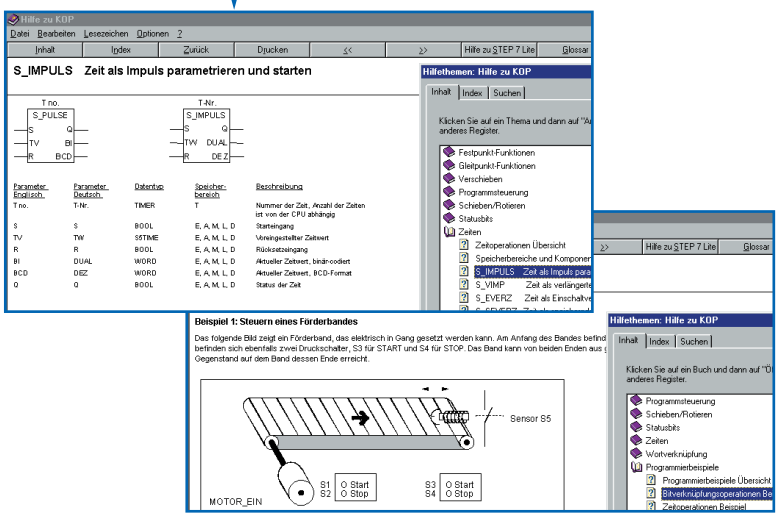
You can also call help on error messages which are displayed in the status bar. Right-click on the error message to call **Help**.

Reference help

Press **SHIFT** + **F1** and then click on **S_IMPULS**, for example.
The reference help pops up in a separate window.

3

The reference help opens.



The reference help provides help on the selected instruction or block.

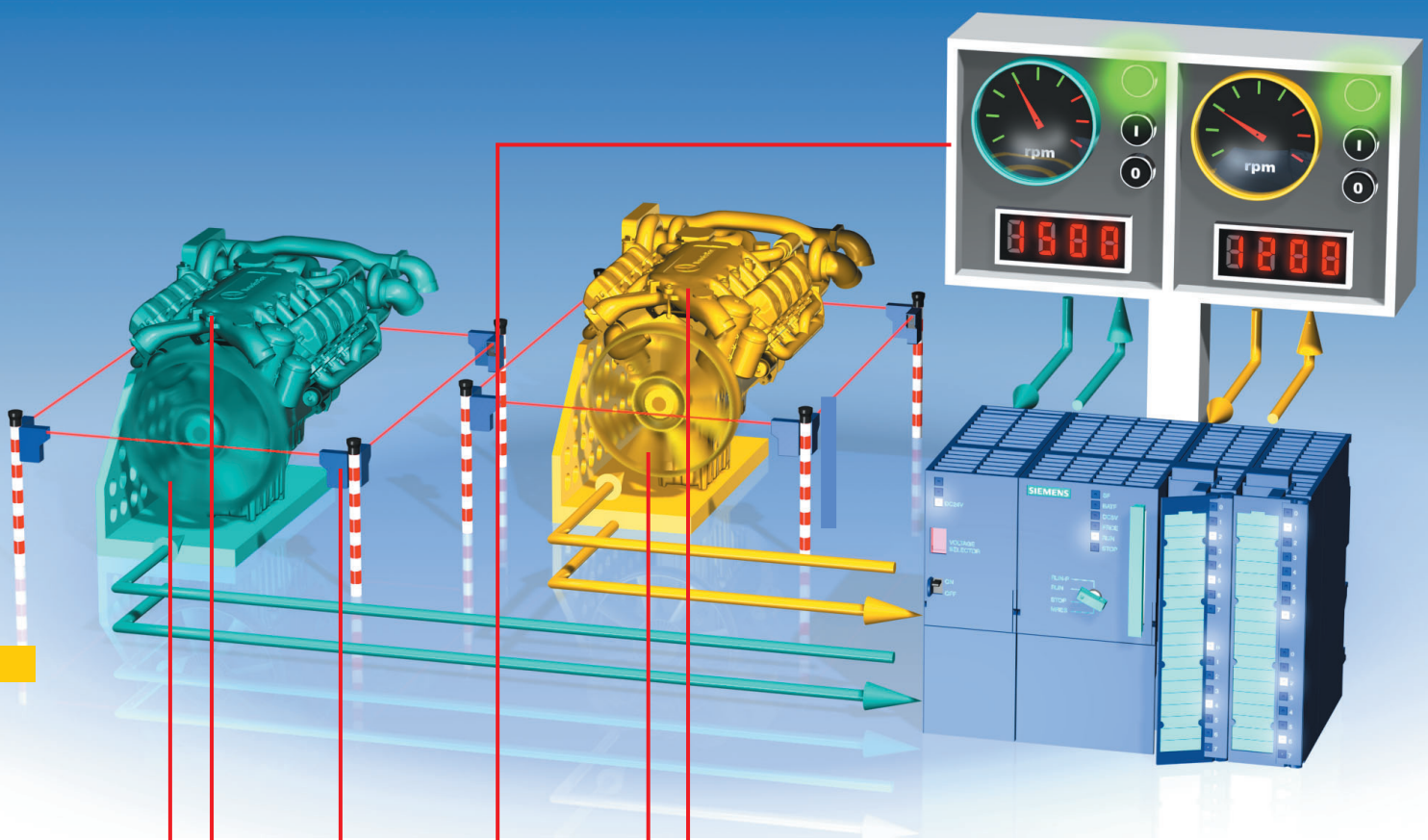
Here you can also call programming samples, for example.

4

Close the copy of the sample project after you have familiarized yourself with the operation of STEP 7 Lite.

3 Implementing the task

Task – Motor bench



Light barriers lead to OFF

Operator station

Petrol engine with ON/OFF switch and tachometer

Diesel engine with ON/OFF switch and tachometer

Fan with ON/OFF switch

Fan with ON/OFF switch

Using our "Motor bench" example, we are going to show you the most important steps to implementing this task in an automation solution.

Task

A controller for a motor bench is to be developed. In the first step, we will only develop the global circuit for the actual testing circuits:

1. A **petrol engine** and a **Diesel engine** on a bench is to be switched on and off individually.

2. A light barrier on each engine secures the area of hazard. This barrier triggers an EMERGENCY-OFF circuit, independent of the sample program.

3. An electrically driven **fan** is also switched on or off with the engine.

4. The fan's off delay is four seconds.

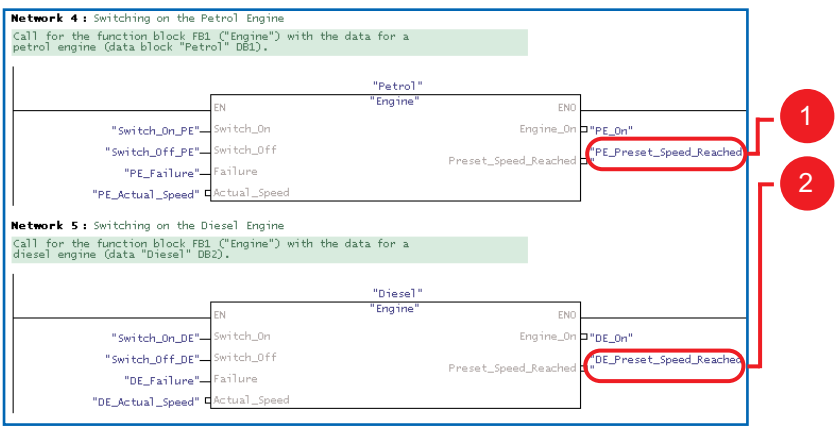
5. The operator will receive a signal indicating that the engines have reached their speed setpoint:

   Petrol engine = 1.500 U/min
   Diesel engine = 1.200 U/min

Solution

Here the solution beforehand: **OB1** in the sample programs contains the signal "Preset_Speed_Reached", realized in

- **Network 4** for the Petrol engine. and in
- **Network 5** for the Diesel engine.

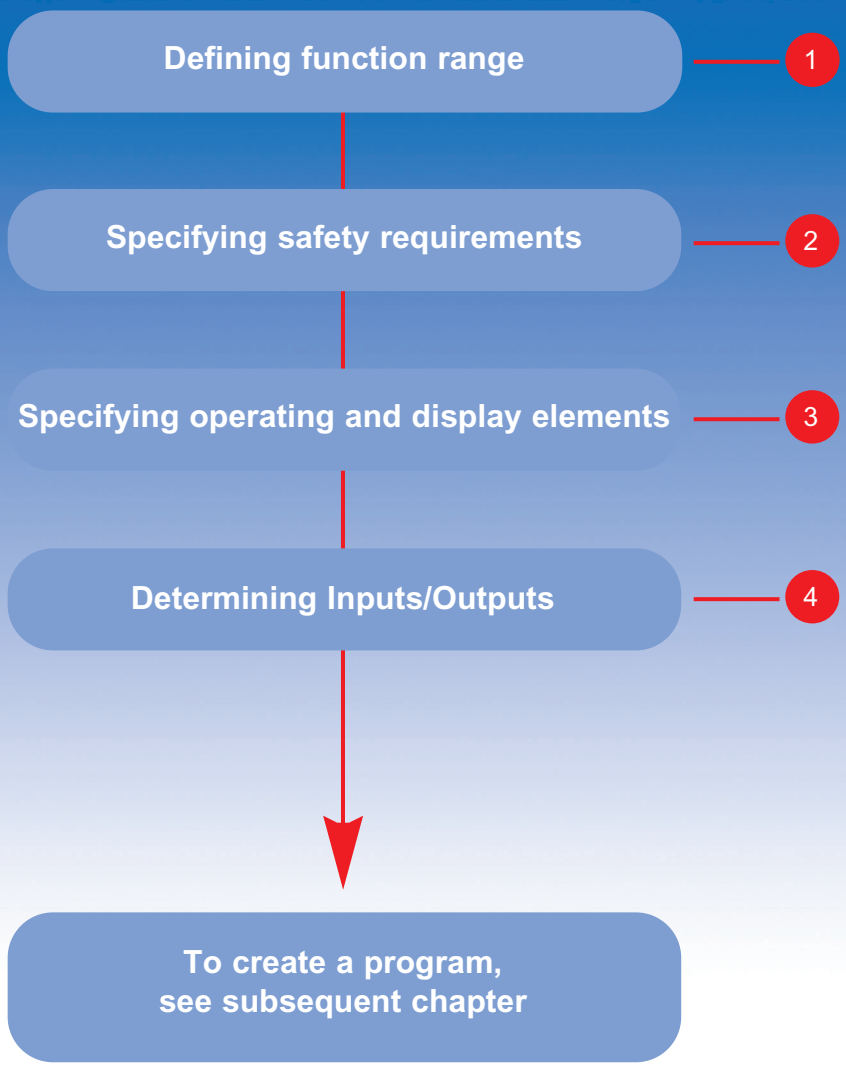


Further information is found in Chapter 7, section "Programming block calls".

You could now use the "Preset_Speed_Reached" signal to initiate a testing process, e.g.:

- Start of a exhaust gas comparison test
- Start of a speed stability measurement.
  However, this is not part of our sample program.

Splitting the process



Split the process before you start programming.

A basic procedure you can use in any configuration is shown above.

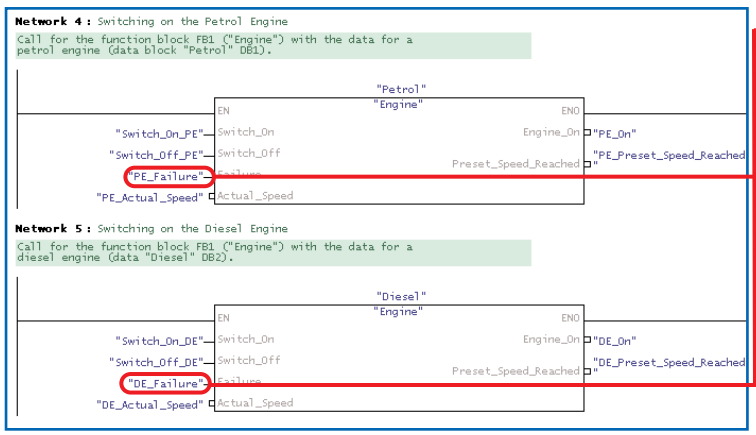
Every step can be split into subsections.
Higher granularity of the split process considerably simplifies the user program structure.

Function range	Related devices
Function range A	= Petrol engine = Tachometer = Fan
Function range B	= Diesel engine = Tachometer = Fan

1

Describing function areas:

- Split the process into related groups.
- Specify the elements controlling this area.
- Specify electrical, mechanical and logical I/Os for all tasks.
- Specify locks and dependencies between the tasks.



2

Specifying safety requirements:

In our sample this is the emergency off circuit. In the real world, however, programming this task is much more complex.

3.5

3

Defining operator and display elements:

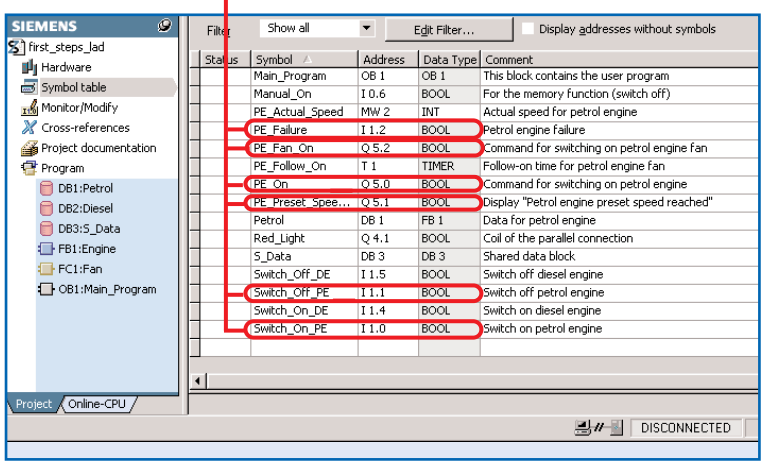
Every process requires an operator and monitoring system that enables human control of the process.

4

Specifying I/Os:

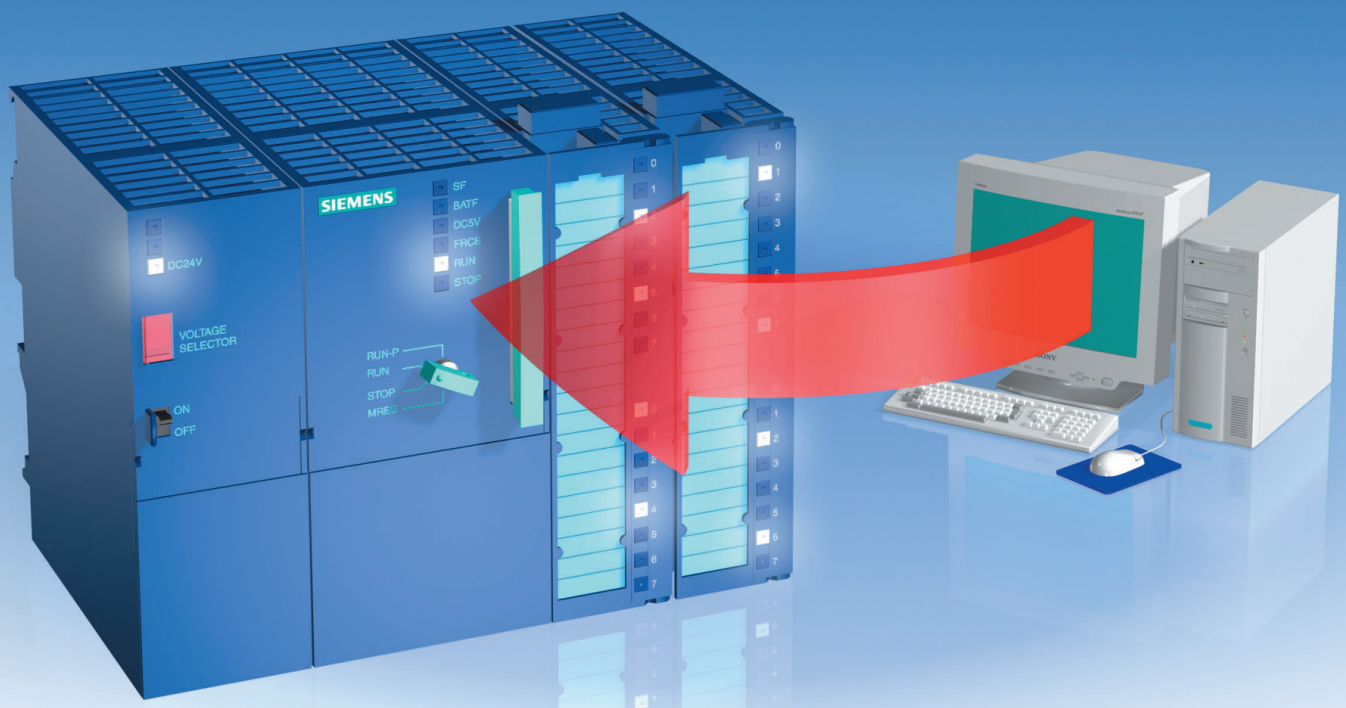
Even for our small sample project, you need three physical **I**nputs and **O**utputs for the petrol engine **PE**

The symbol table in Chapter 5 offers you a good overview of all I/Os.

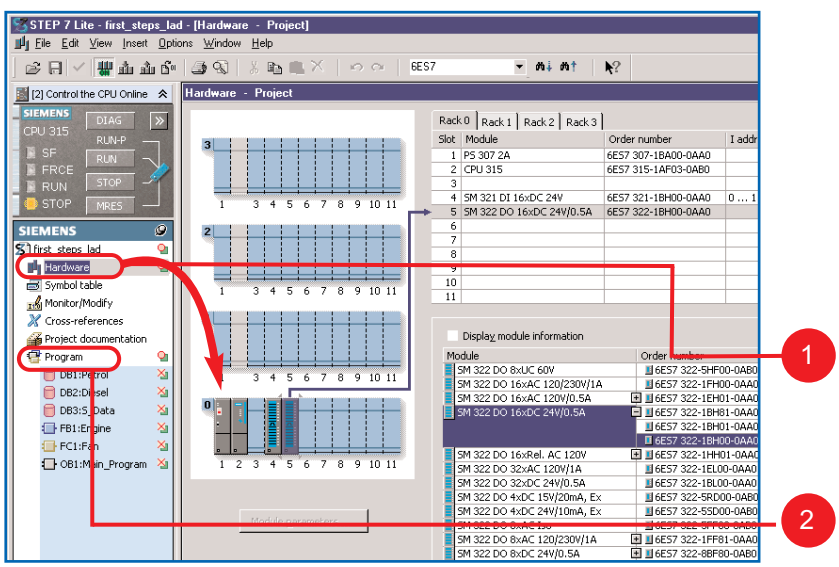


We invite newcomers to participate in the SIEMENS training courses. Here, they are shown practical examples on how to automate processes using a SIMATIC system.

4 Module configuration



What happens during configuration?

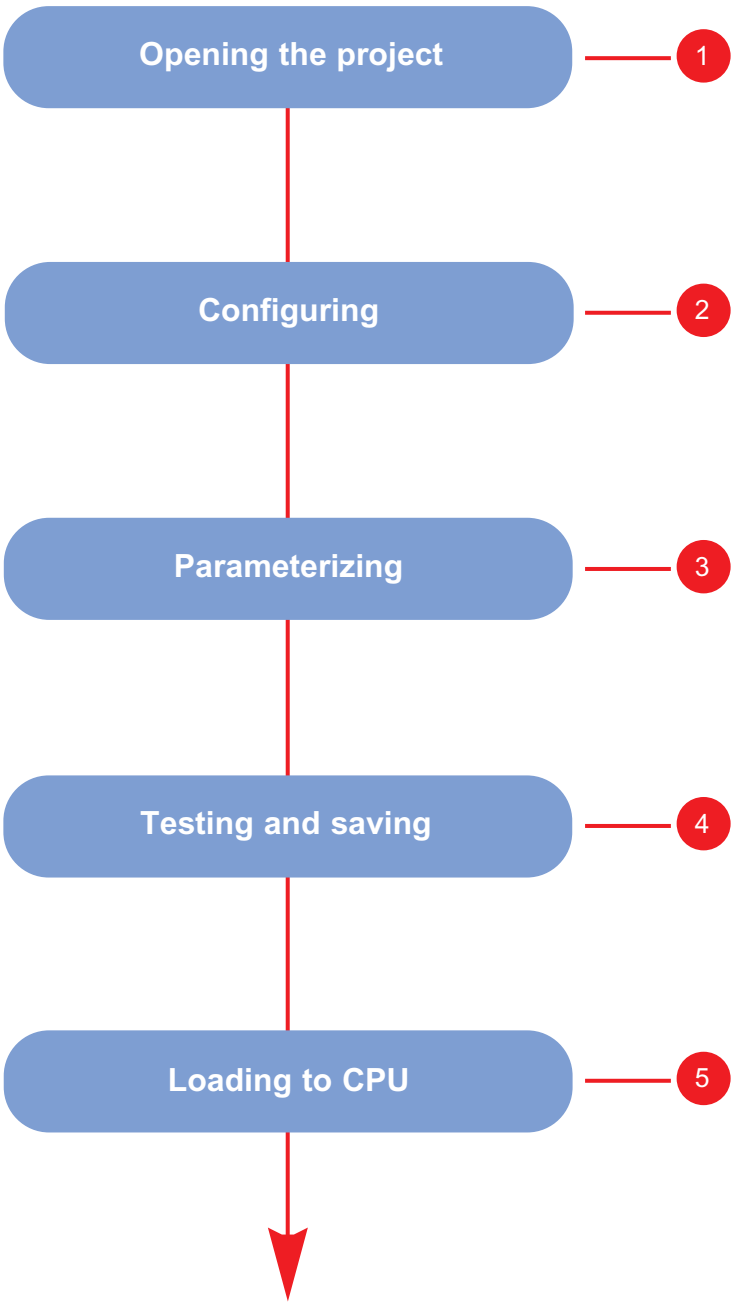


During hardware configuration you develop an image of the PLC station on your programming device. You can copy this configuration to other STEP 7 Lite projects and, if required, modify and download it to other existing stations. During the PLC startup routine, the CPU compares the default configuration created in STEP 7 Lite with the actual configuration of the system. This way any existing errors can then be detected and reported immediately.

1 The hardware element in your project window shows a graphic representation of a rack. Here, specify all modules you have integrated in the PLC station.

2 At a later point, you are going to edit the user program in the program element for precisely this hardware configuration.

Hardware configuration overview



Create your hardware configuration within a project. Projects and programs always belong together, since you ...

... also assign the I/O address areas in your hardware configuration. And, you subsequently specify these addresses in your program.

You can modify some of the modules for the Siemens factory settings. However, this is not required in most cases.

You can then you perform a consistency check for your hardware configuration and save your entries.

Download the hardware configuration from your PG to the CPU.
So that,

- the CPU can recognize all existing modules and their addresses,
- module parameters can be enabled.

Creating a new project

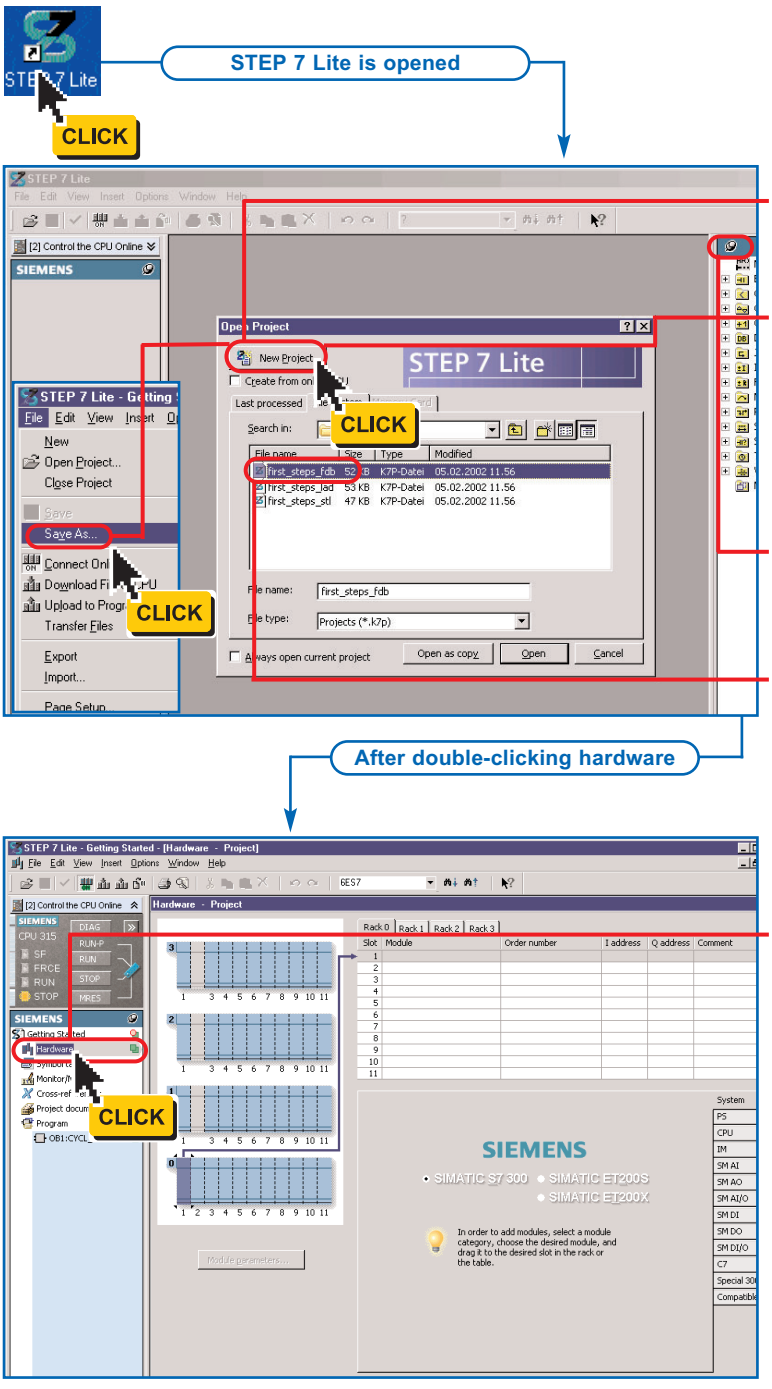
1 In this chapter you are going to create a new ”Getting Started” project, shown here at top entry level.
In the subsequent chapters you will continue to develop this project.

Your “Getting Started” project must have your existing hardware configuration and not the one in our samples.

You can follow the sample projects also installed when configuring your hardware. The sample projects are under :
Drive:\Siemens\S7Lite\Examples\English...

The hardware configuration is the same in all three sample projects.

How to create a new project

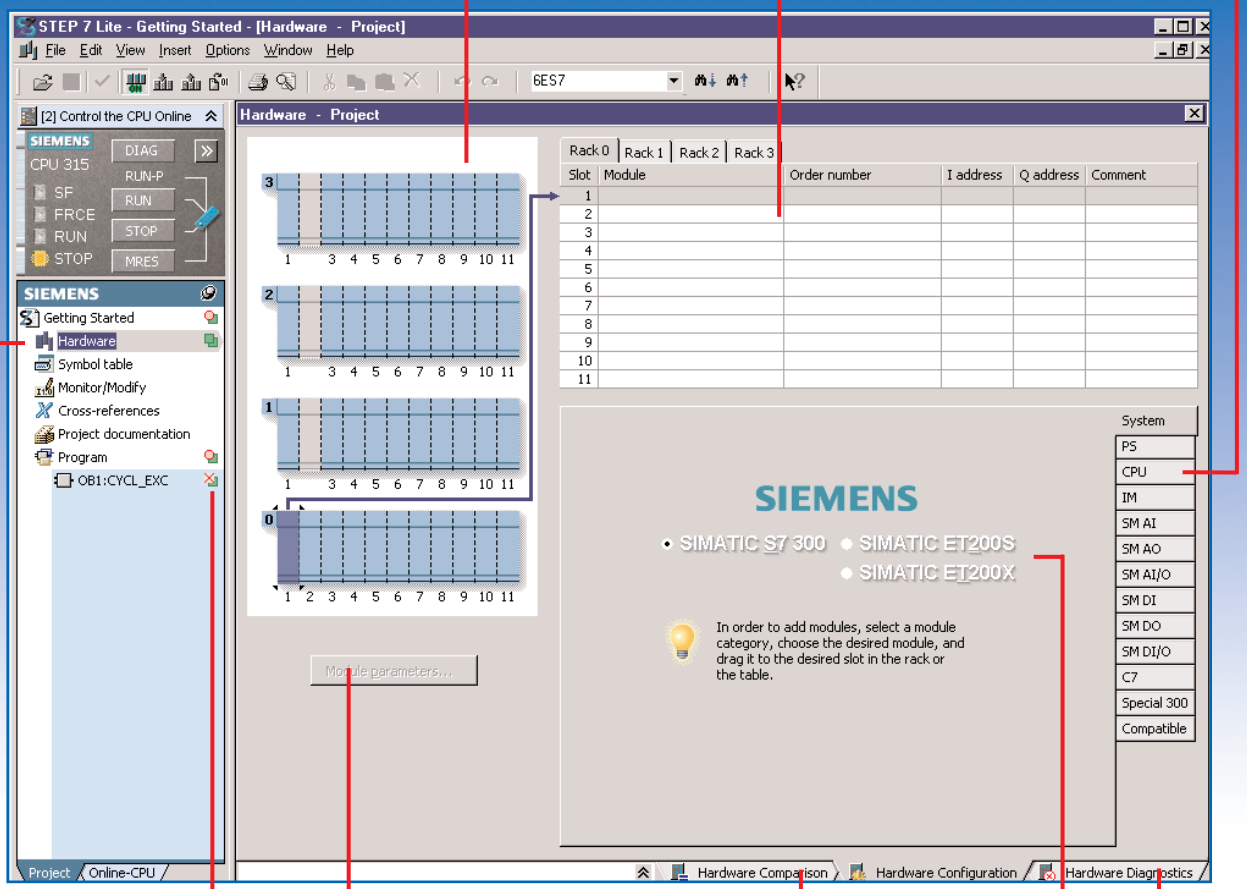


1. Start STEP 7 Lite with a double-click.
2. Open a new project.
3. Save the project as

   ”Getting Started”
4. Click on the pin needle to hide the libraries.
5. If you want to use one of our configurations as a guideline, also open one of the included sample projects in a second instance of STEP 7 Lite.
6. Double-click on Hardware.
   The hardware configuration windows opens in the working area.

You can copy hardware configurations from one project to another. For example, if you want to continue immediately with Chapter 5, open one of the included sample projects and copy the **Hardware** element to your ”Getting Started” project (see Step 5).

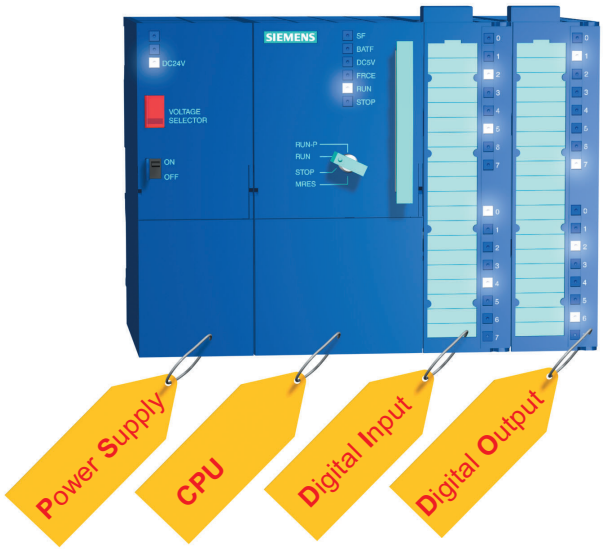
Working in the hardware configuration view



Overview

After opening the hardware element with a double-click in your project window, the ”Hardware” view is shown at the right side in the working area.

Select the modules of your PLC station from the hardware catalog here.



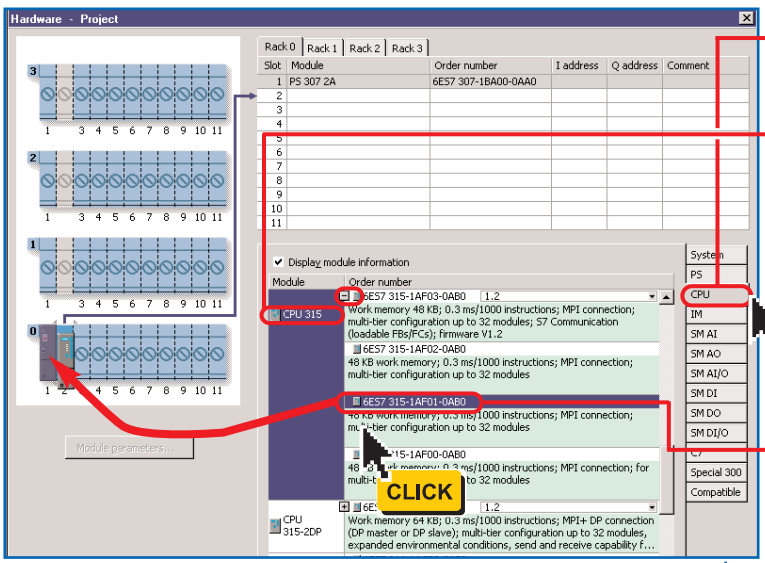
How to configure the hardware

The following modules are stored in the sample programs:

1 **P**ower **S**upply = Power supply module
2 **CPU** = SPS module
3 **D**igital **I**nput = Digit input module
4 **D**igital **O**utput = Digital output module

Order numbers are imprinted on the module front panels.

Configure your module as described below.



Another window opens

Click on **CPU**.

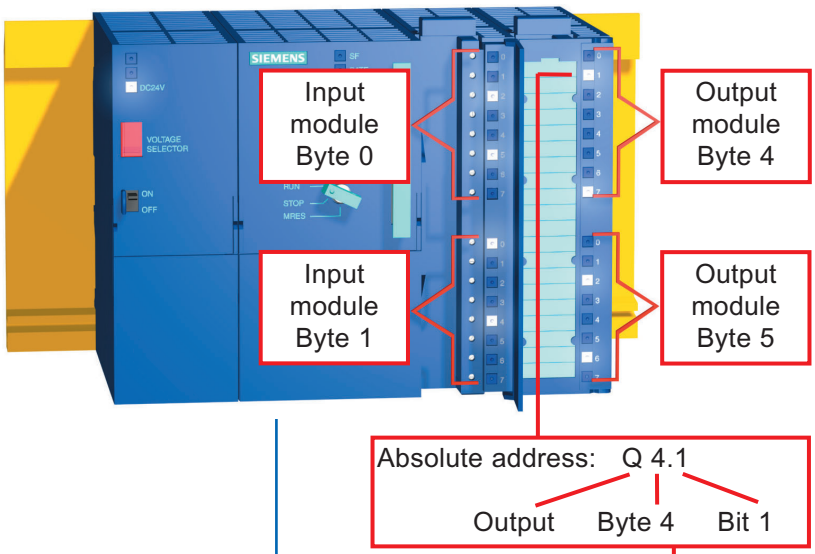
Go to **CPU 315** and click on the "**+**" icon to view all earlier versions of CPU 315.

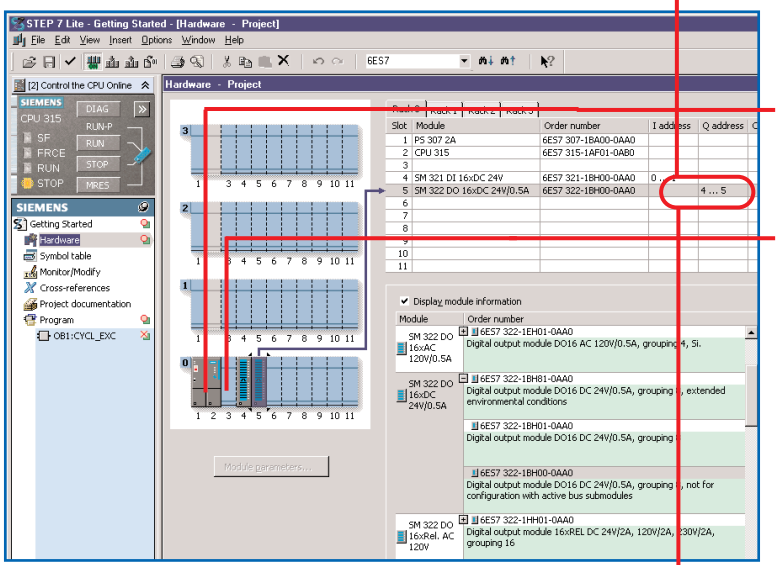
Drag the CPU to the rack using a drag-and-drop. A lock sign indicates slots which are not permitted according to the slot rules.

Proceed in the same way with all modules.

We shall continue with Chapter "Module parameter assignment" on Page 4.12. Refer to the next page for more details.

Of course, if you later want to download the hardware configuration to your CPU you must configure your hardware, and not necessarily the one in our sample project.





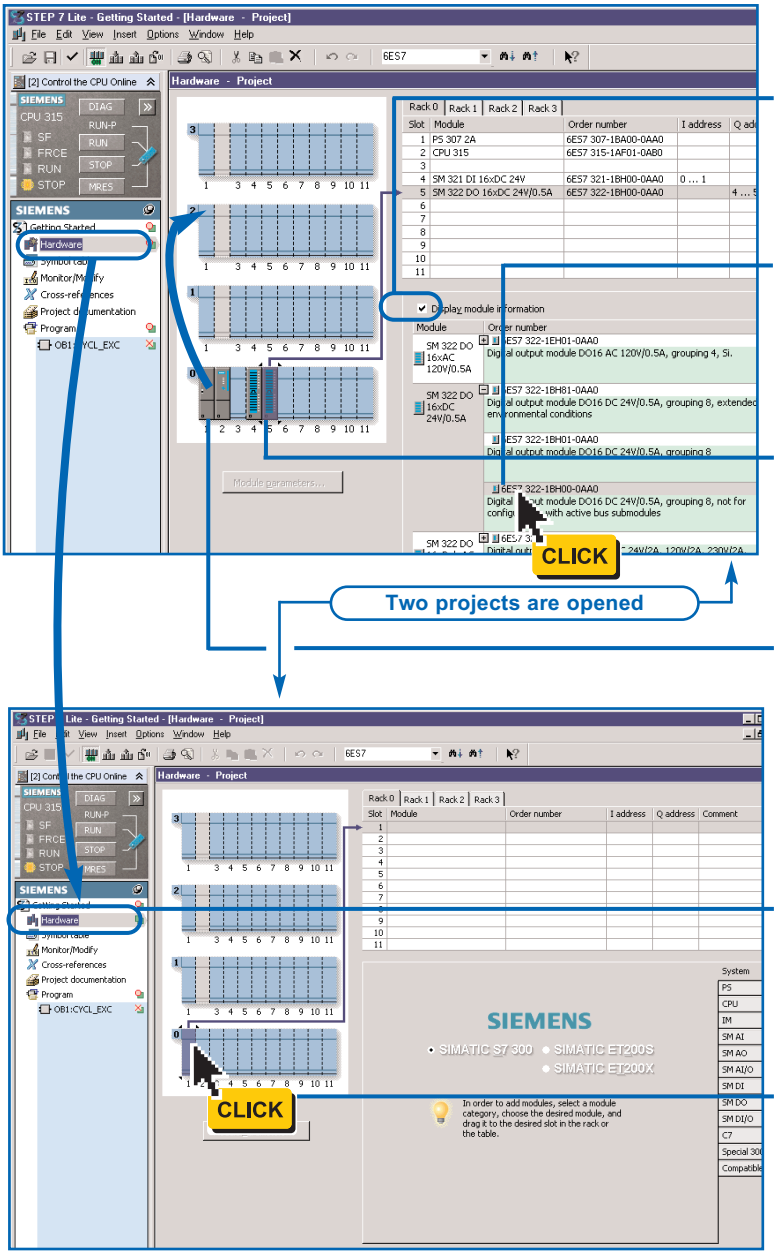
Configuration results

Working from left to right, insert the power supply module, CPU, input module, output module.

Modules must be inserted without leaving empty slots between them. Otherwise, they cannot be supplied with power via backplane bus.

Exception in STEP 7 Lite: Slot 3 is reserved for the interface module (IM) you can use to connect racks stacked on top. If modules are inserted only in the lowest rack, you can leave a space in the configuration.

The address specification bytes have been set automatically in the **I/O address** columns of the **configuration table**. They are a major component of address specification for programming.



Tips – Inserting modules

1 Improve your screen overview by hiding **module information**, e.g. if you want to perform a directed search for order numbers.

2 Insert modules, for example, by right-clicking the module to open the pop-up menu and select **Insert module**.

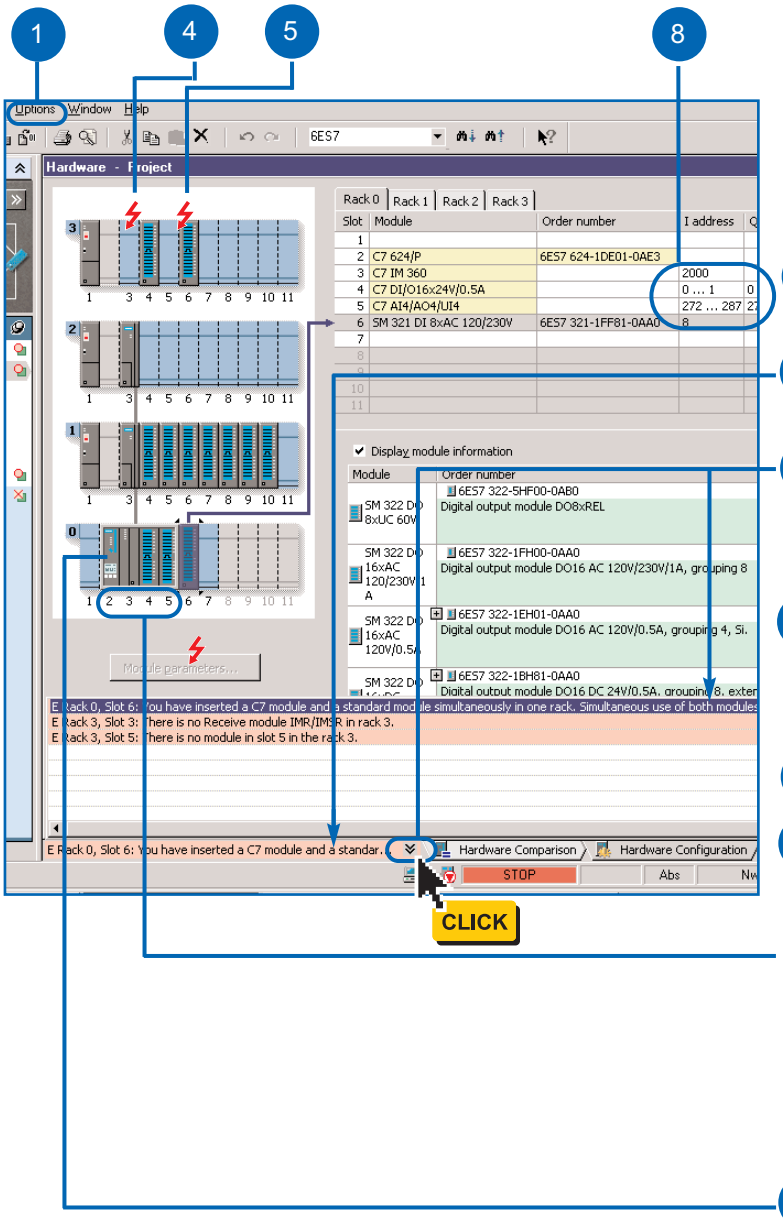
3 You can insert new modules between two existing modules. The modules are shifted to available slots on the right side.

4 Using the Shift key, you can highlight any number of modules in the rack and then copy them or move them by a drag-and-drop operation.

5 Open another project in a second instance of STEP 7 Lite. Then, for example, drag the complete hardware configuration from one project to another by a drag-and-drop operation.

6 To delete a module, right-click on the module to open the context-sensitive menu and select delete.

Feel free to try out all functions you have used in other Windows applications.
We have implemented many Windows functions in STEP 7 Lite, e.g. pop-up menus, drag-and-drop operation, working with shortcut key etc.



Example of a maximum configuration

For demonstration purposes we have created a large configuration with some errors.

1 Call the troubleshooting routine via: **Options > Check Consistency**.

2 An existing configuration error is displayed in this view.

3 Left-click on the expansion icon to display all errors.

Existing errors:

4 An interface module (IM) is missing in Rack 3. As a result, there is no connection to Rack 3. Racks 1 and 2 are equipped with interface modules.

5 Free slots are not allowed.

6 This is a C7 compact system (highlighted in yellow background color in the configuration table). The module on the right side is incompatible.

After you have eliminated the errors, perform another consistency check.

And otherwise:

7 STEP 7 Lite allows only one CPU per project. The CPU is always inserted in Rack 0. The top slots are not used.

8 Please note how the address areas are incremented in the configuration table.

Online Help: F1

– Under **Content > Configuring the hardware > Configuring modules** in the Help on STEP 7 Lite you can find global configuration rules.

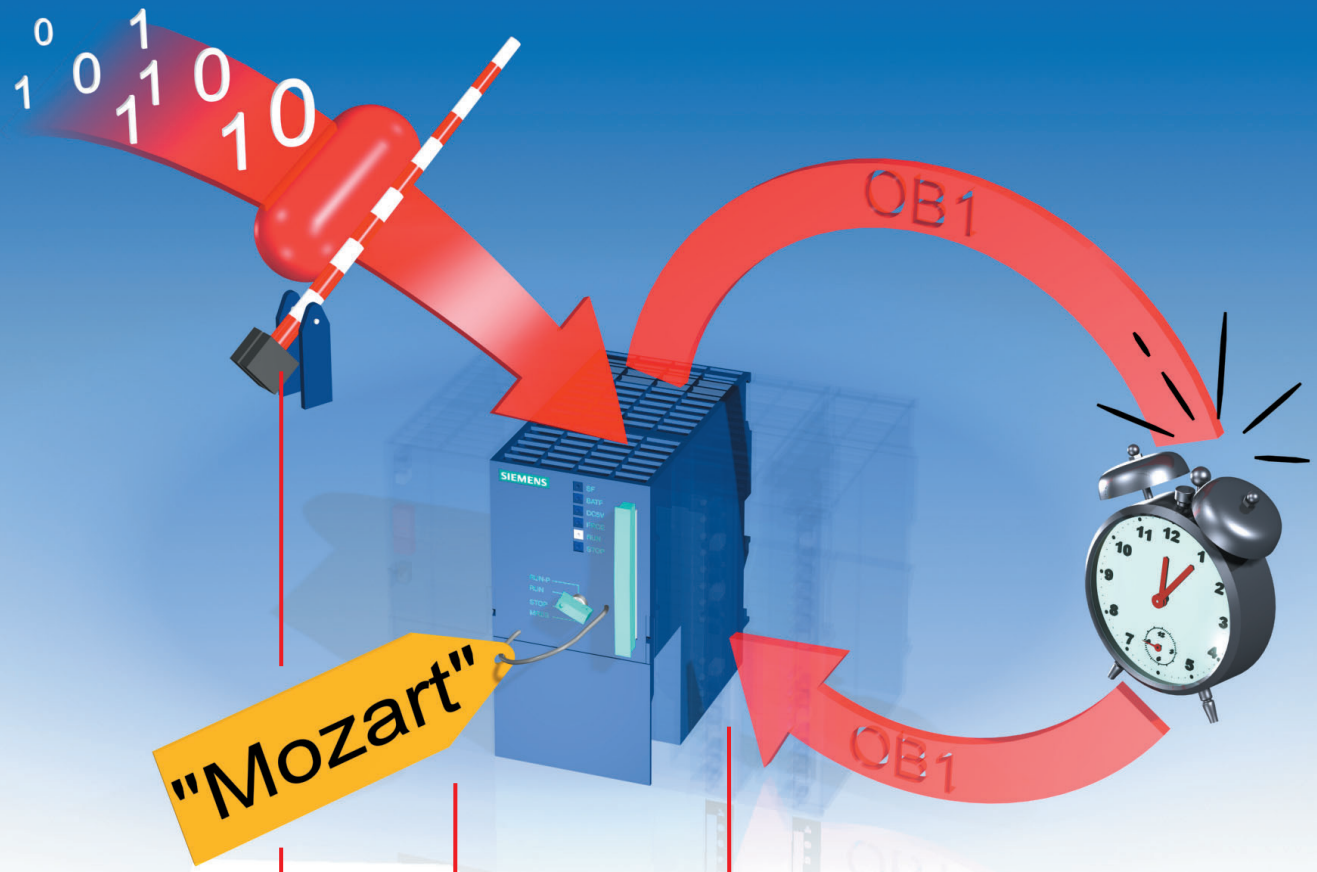
– Under **Index > Slot rules**, you can find the most important rules on insertion.

Browsing the hardware catalog with MLFB

The MLFB number represents the Siemens order number.

If you know the MLFB number of a module you want to look up in the selected hardware catalog, you can enter this MLFB via ”Find text” dialog box in the toolbar. Then, press **Return** to display the module.

Module parameter assignment



What is parameter assignment?

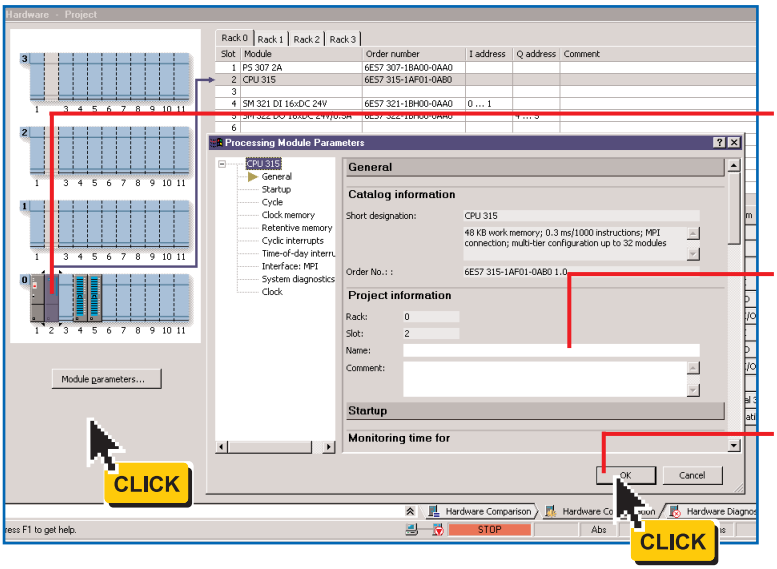
You can customize the operating characteristics of some of the analog and digital modules, as well as those of the CPU: we refer to this as ”Parameter assignment”.

Examples of parameter assignment to a CPU:

1 – You can interrupt the CPU’s program cycle via watchdog interrupt.

2 – Specify a name for the CPU. In this case, it is ”Mozart”.

3 – You can also password protect your CPU against MPI read/write access.



Assign parameters to CPU 315

1. Highlight CPU 315. Click on the **module parameters** button.
2. In the **Processing module parameters** dialog, enter ”Mozart” in the **Name** box.
3. Confirm your entries with **OK**. The window is closed.

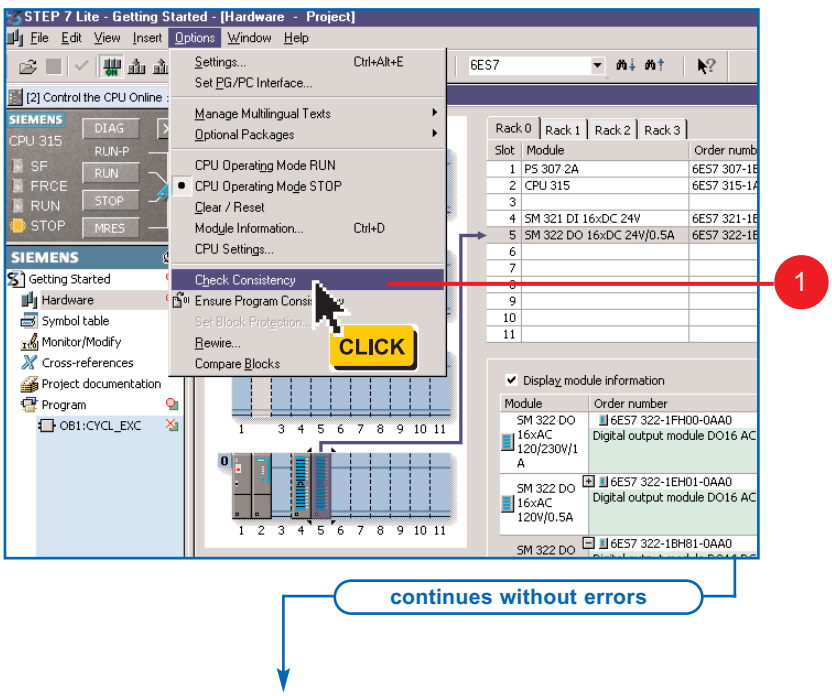
All basic parameters are factory set and match almost any standard functions.

If anything goes wrong after you have made changes – do not worry – the hardware catalog still contains the basic settings for all modules.

Online Help: F1
CPU parameters are often related to organization blocks.

In the **Index** under **Cyclic interrupt**, you can therefore find the description of **Organization blocks for cyclic interrupts (OB30 to OB38)**.

Saving configuration data

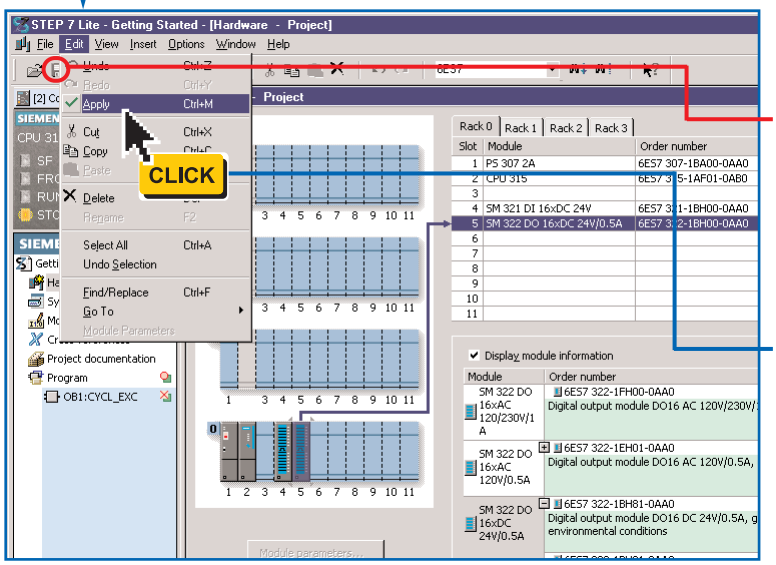


How to check configuration data

Before you save your configuration, you should always perform a consistency check.

Call menu item **Options > Check consistency**.
This is to check whether your configuration data can be generated using your entries.

Confirm the message ”The configuration is error-free.” with **OK**.



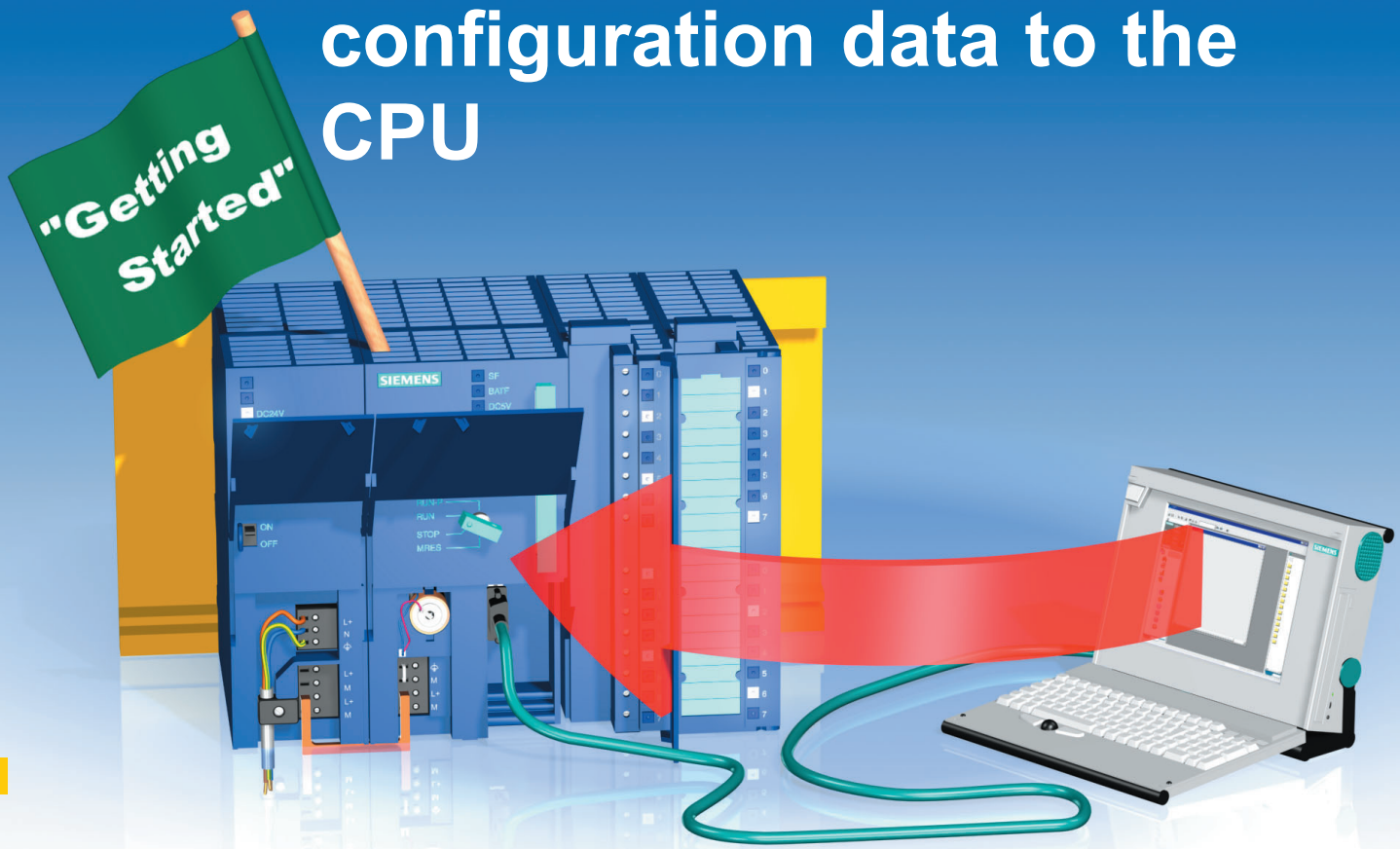
How to save configuration data

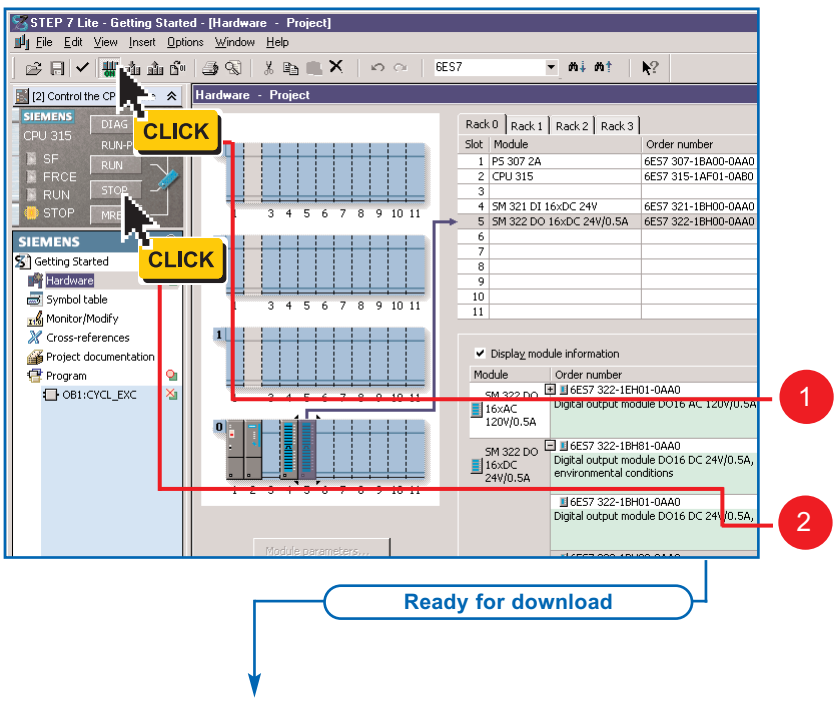
Select **File > Save**, or click on the disk icon in the tool bar.

This does not only save your hardware configuration, but rather all project elements.

When you select the menu command **Edit > Apply**, your configuration data (always the content of the active window) is saved to a temporary file. This file saving method is recommended if you intend you to undo modifications later.
After only applying data, you are prompted to save your changes when you close the project.

Downloading hardware configuration data to the CPU





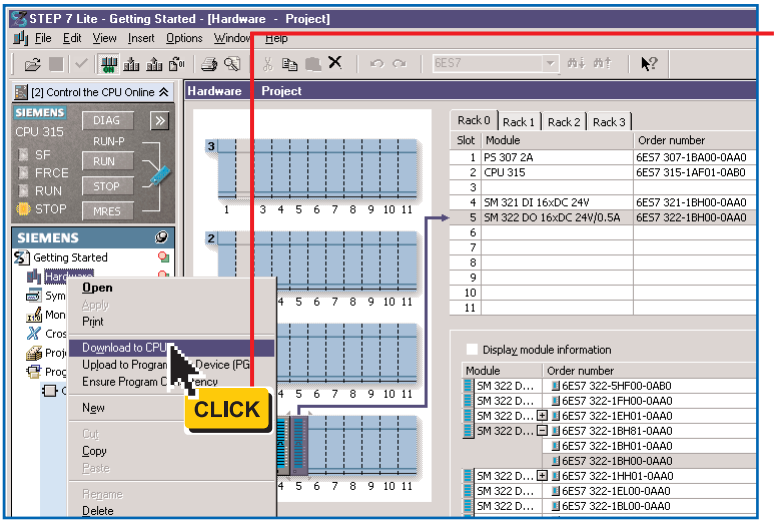
How to prepare the download

With this download you transfer all configuration data to the CPU.
Note that you must first establish an "Online connection" between the CPU and the PG. Details on this topic are found in Chapter 10.

"Connect Online" essentials:

After you have connected the cables and performed a CPU memory reset, click on **Connect Online**.

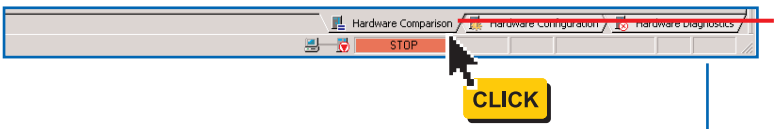
In the CPU operator panel, set the CPU to **STOP** mode. The footer displays a red STOP icon.



3 Right-click on **Hardware** to select the **Download to CPU** function.

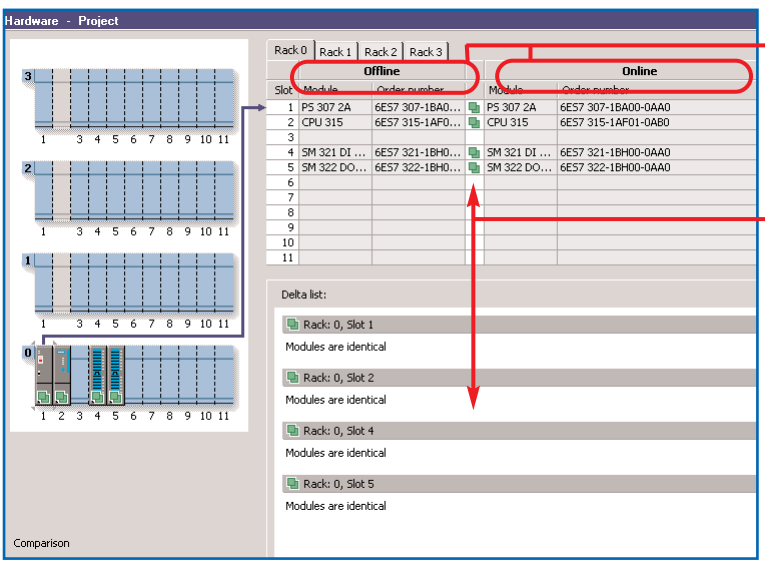
Hardware configuration data is now downloaded to the CPU. Parameters are assigned to the modules.

Online/Offline comparison



4 Click on the **Hardware Comparison** tab.

The view "Hardware Comparison " is opened



5 Here you can verify consistency between configuration data on your PG (Offline) to data downloaded to the CPU (Online).

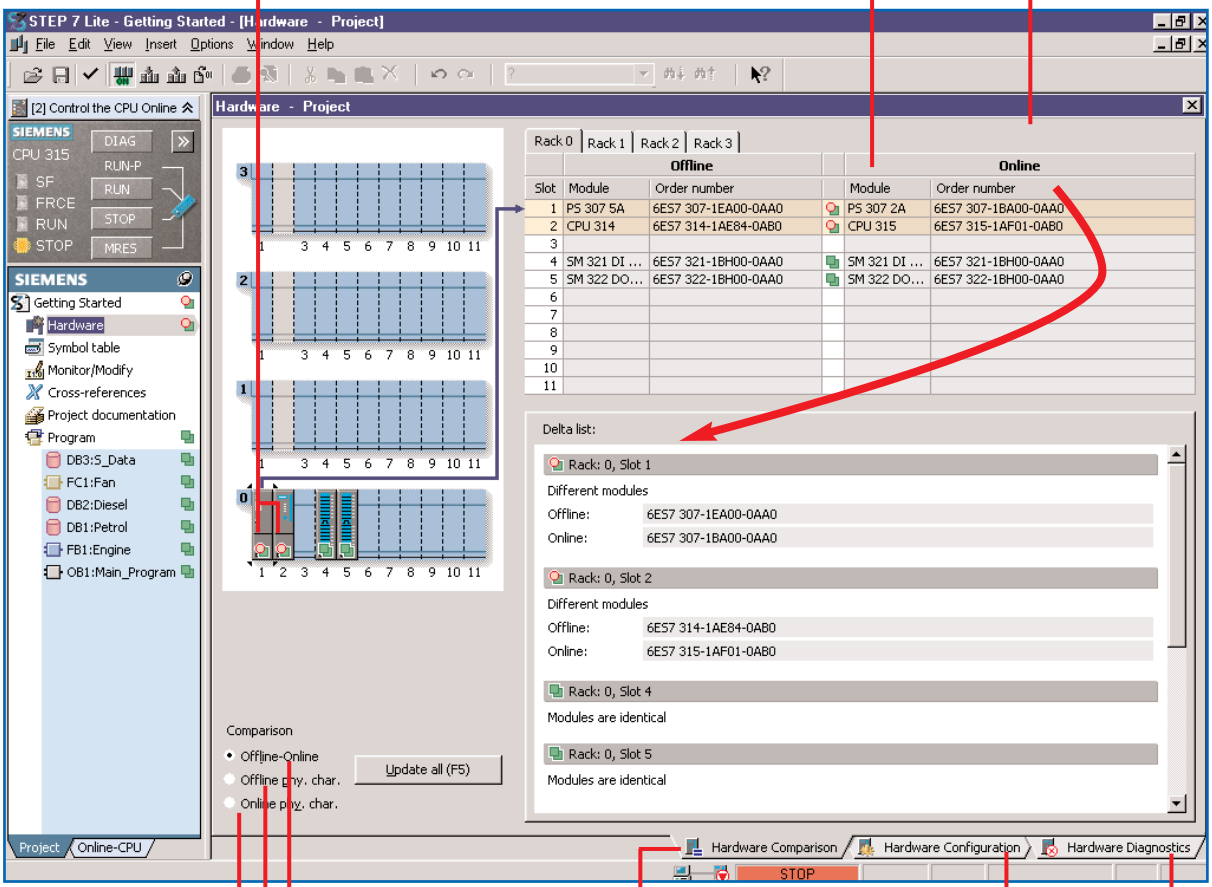
6 Existing data inconsistencies are indicated in pictograms and details are returned in the delta list.

For more details on screen

Using the instruction **Upload to Programming Device (PG)**, you can upload a hardware configuration from the CPU to the PG. This is a typical service action if you want to access a switching cabinet with a PG in order to analyze an error.

Comparison table:
comparison of online/offline/physics

Module-related differential icons



...displays the difference between the configuration entered on the PG in the Project tab (offline) and the configuration downloaded to the CPU.

... displays the difference between the configuration entered and the physical hardware inserted.

...displays the difference between the configuration loaded and the physical hardware inserted.

Overview

1. In the **Hardware Configuration** tab, enter your hardware as described above.
2. If errors have occurred, perform a consistency check of configuration data via **Hardware Comparison** tab. Details on this topic will follow.
3. If problems persist, it could well be that modules are defective. Perform a check via **Hardware Diagnostics** tab. Details are found in Chapter 12 under ”Error diagnostics”.

How to detect errors:

You have downloaded the configuration to the CPU and called the **Hardware Comparison** tab.

In your project window, **Hardware** is marked with a **collective pictogram**. This indicates that one or several modules do not match.

Pictograms on the **Rack** modules identify these modules.

Compare: Offline - Online

Online: Configuration which was downloaded to the CPU.

Offline: Configuration on the PG.

When you click on **Hardware Comparison,** the **Comparison: Offline-Online** button is selected by default. The **Delta list** displays the differences in your configuration and parameter assignment.

Compare: Offline - Physics
Compare: Online - Physics

Physics: Refers to the configuration a CPU will recognize automatically, without prior configuration download.

Click on the corresponding button to compare the Online/Offline configuration with the physics.

Symbols

These are the essential icons of the hardware configuration.

The configured module does not match the module of the Online CPU.

The physically inserted module matches the configured module, however, it has been assigned different module parameters.

The module is configured, but does not exist online.

Symbolizes a ”Possibly identical module”. The type of the physically inserted module matches the configured module. It cannot be determined whether the order numbers also match.

Operating mode RUN

Operating mode STOP

Operating mode HALT

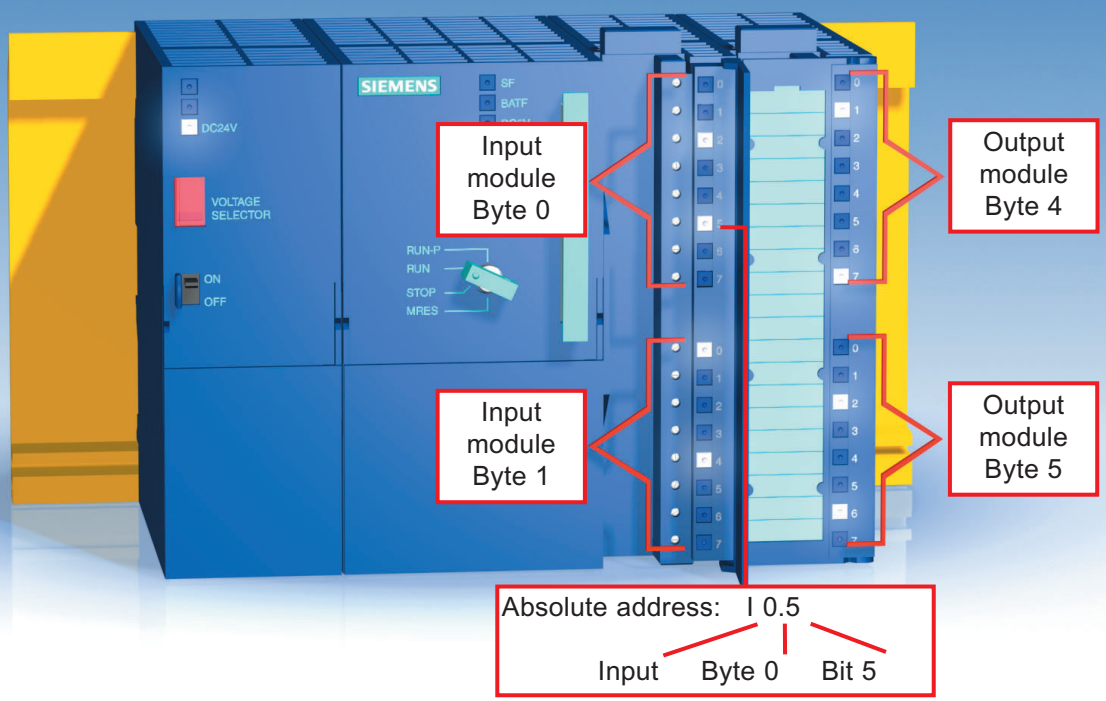
Error

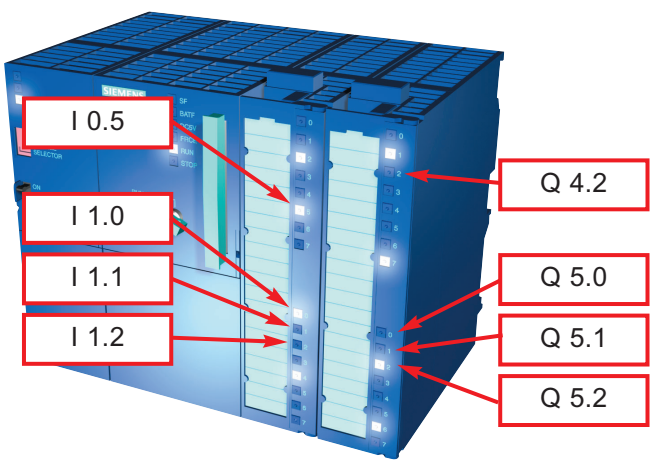
Symbols (Icons)
For more information on symbols use the quick info icon.

Under **F1 > Index > Symbols (Icons)**, you can find an overview of icons which can be displayed in the project window, rack and comparison table.

5 Creating the symbol table

Absolute programming



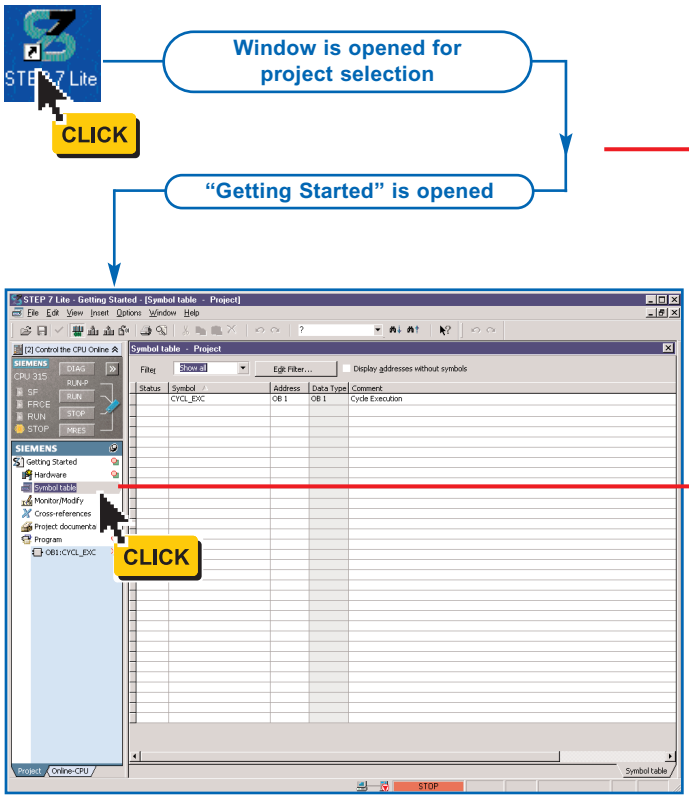


How to assign addresses

Chapter 4 describes how absolute addresses are assigned during hardware configuration. As a reminder:

Due to the hardware structure, every input and output is assigned a default absolute address.

The absolute address can be replaced by a freely selectable (symbolic) name (e.g. Q 4.2: Automatic_mode). Symbols are assigned independent of the programming language, that is, LAD, FBD or STL.



Symbol table and absolute addresses

Open STEP 7 Lite. In the **Open project** window, Click on the "Getting started.k7p" project you have created in Chapter 4.

Your project currently consists only of default project elements and of the program element **OB1**.

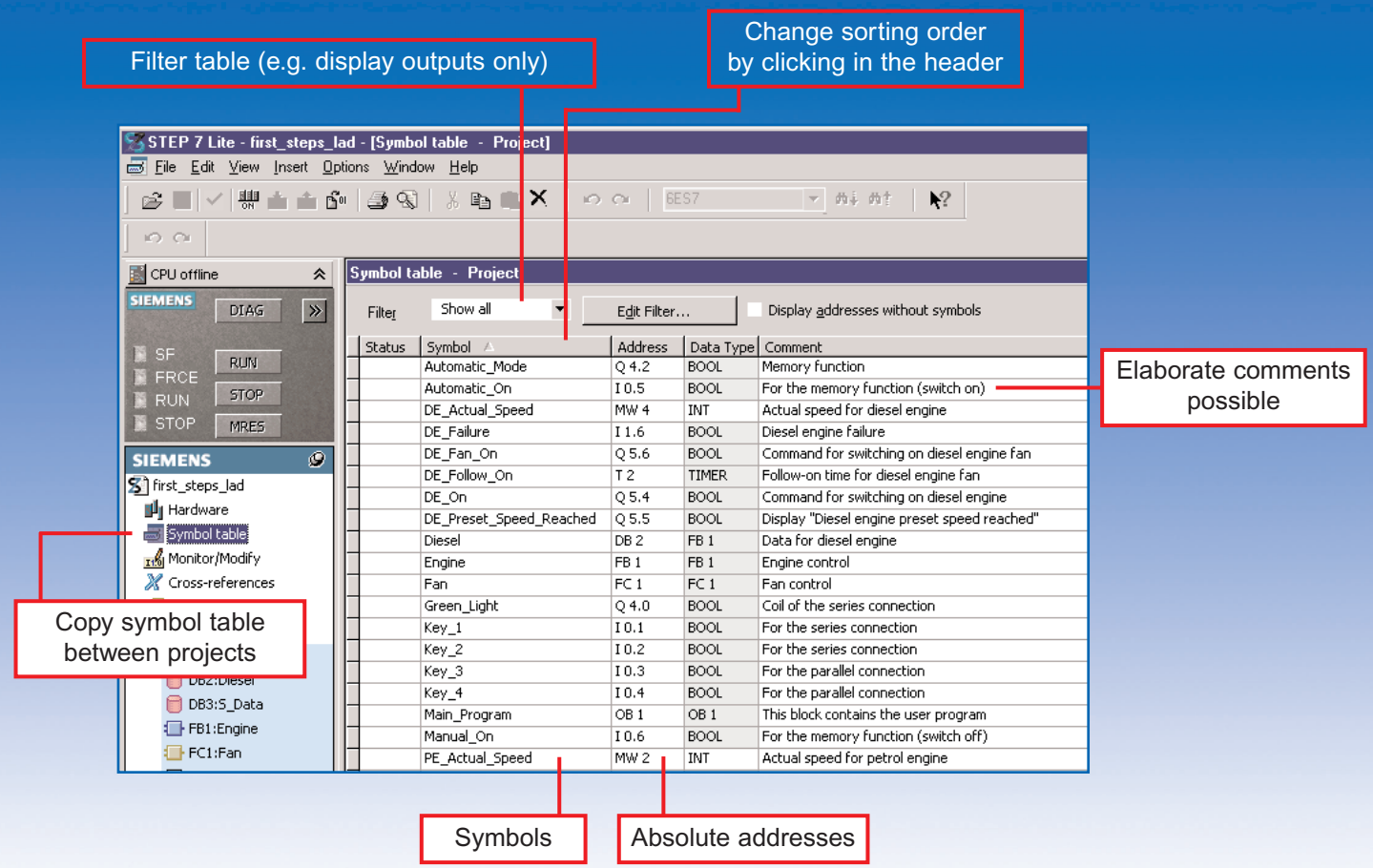
In the project window, double-click on the element **Symbol table**.

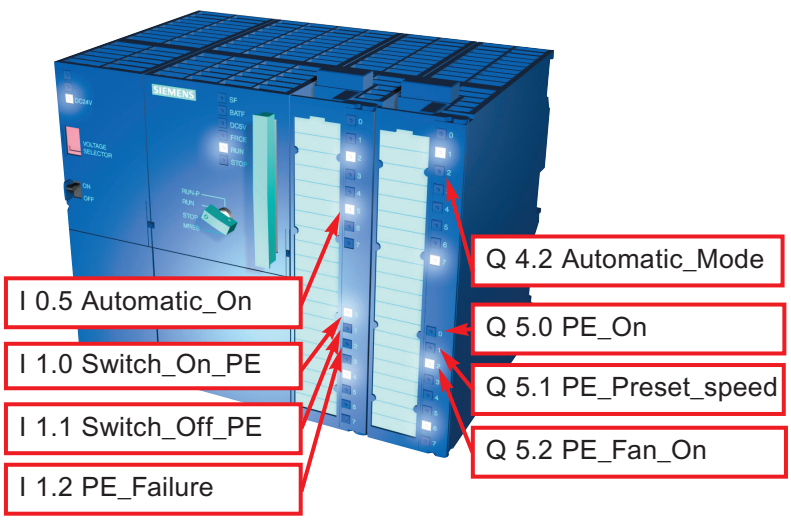
The symbol table currently consists only of the default organization block OB1.

Additional entries are not required if you choose to work with absolute addresses in your program. Simply close the window again.
Symbolic programming is used for the sample project. Proceed as described in the pages below.

You should only use absolute programming if you have to address only a small number of I/Os in your STEP 7 Lite program.

Symbolic programming



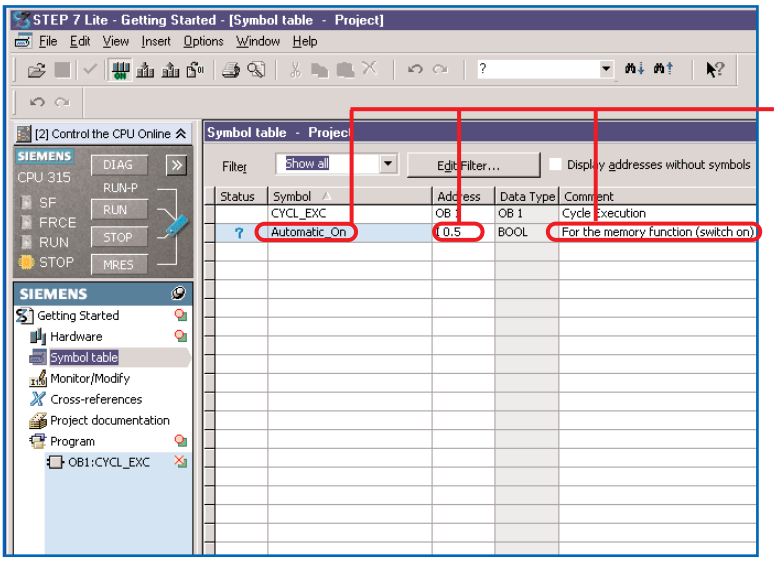


The symbol table

In the symbol table, assign a symbolic name and data type to all absolute addresses you want to address in your program, e.g. assign the "Automatic_On" symbol to input I0.5".

These names, referred to as global symbols, apply to the complete project.

Symbolic programming can considerably improve readability of your program.



Filling out the symbol table

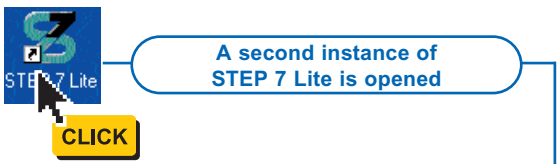
1 In the **Symbol** column, enter "Automatic_On" for address "I0.5". In the **Comment** column, enter the comment as shown to the left.

During input
Return = One row down
Ctrl + **z** = Undo

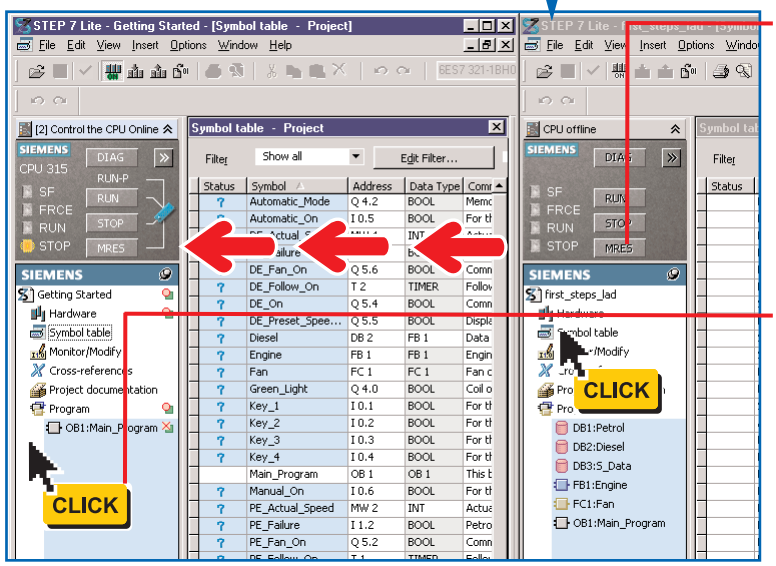
2 Save your entries via **File > Save**.

How to copy the symbol table

Since quite a large number of symbols are used in your "Getting Started" project, copy the symbol table from one of the included sample projects.



3 Also open the project "first_steps_lad.k7p" in a second instance of STEP 7 Lite.



4 In the "first_steps_lad" project, right-click on **Symbol table** to open the pop-up menu. Select **Copy**.

5 In the "Getting Started" project, right click in the project window to open the pop-up menu. Select **Paste**.
You are prompted to confirm over-writing. Confirm with "OK".

6 Close the project "first_steps_lad". Save your "Getting Started" project via **File > Save**.

Data types

Data types determine the type of signals a CPU has to process.

STEP 7 Lite uses, amongst others, the data types shown on the left.

BOOL
BYTE
WORD
DWORD

- Data of this type consists of bit combinations. 1 Bit (Type BOOL) up to 32 bit (DWORD).

CHAR

- Data of this type uses exactly one ASCII character.

INT
DINT
REAL

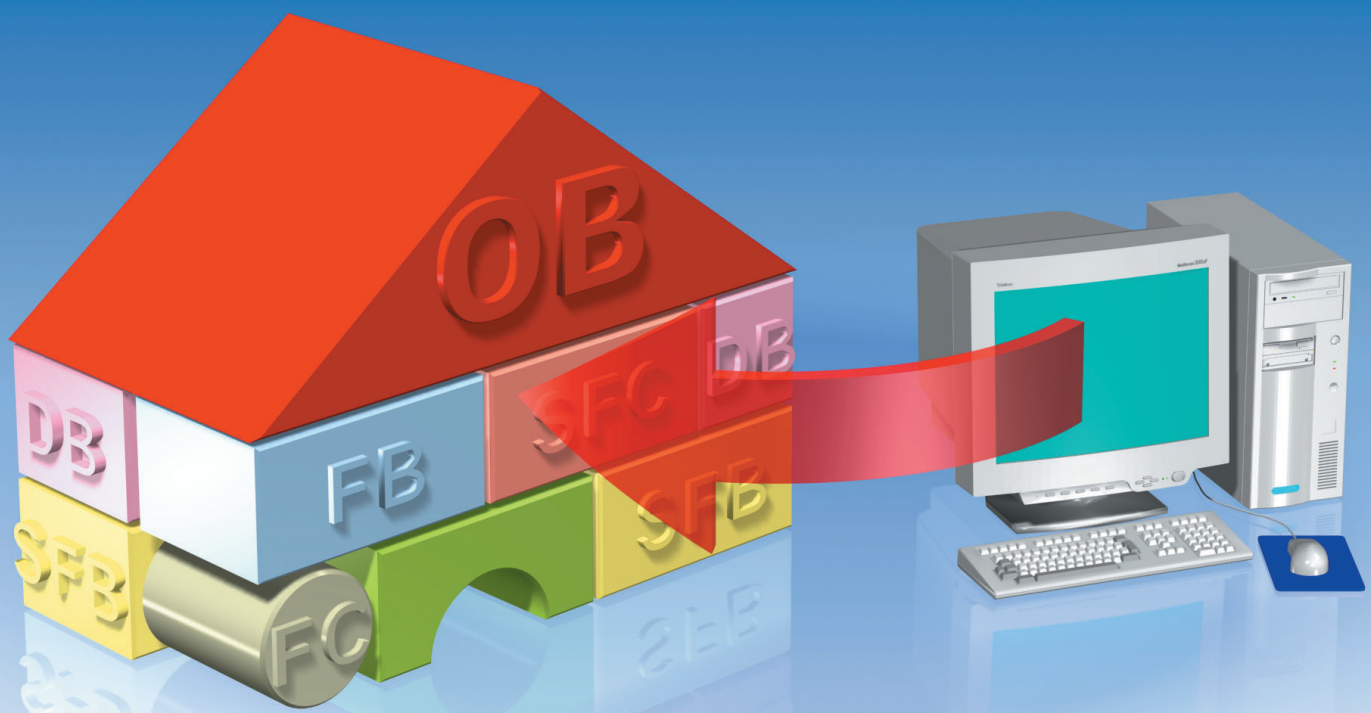
- Data of this type is available for processing numerical values (e.g. for calculating arithmetic expressions).

S5TIME
TIME
DATE
TIME_OF_DAY

- Data of this type represent different time and date values within STEP 7 Lite (e.g. for setting the date or input of the time value).

You will find more information on data types (e.g. permissible ranges of values and application samples) by clicking the help pointer on a data type and then jump to **Introduction to data types and parameter types**.

6 Getting started with programming



Choosing LAD, FBD or STL

6.2

Page 6.6 to 6.11
Ladder logic (LAD) is appropriate for users e.g. in the industrial electrical sector.

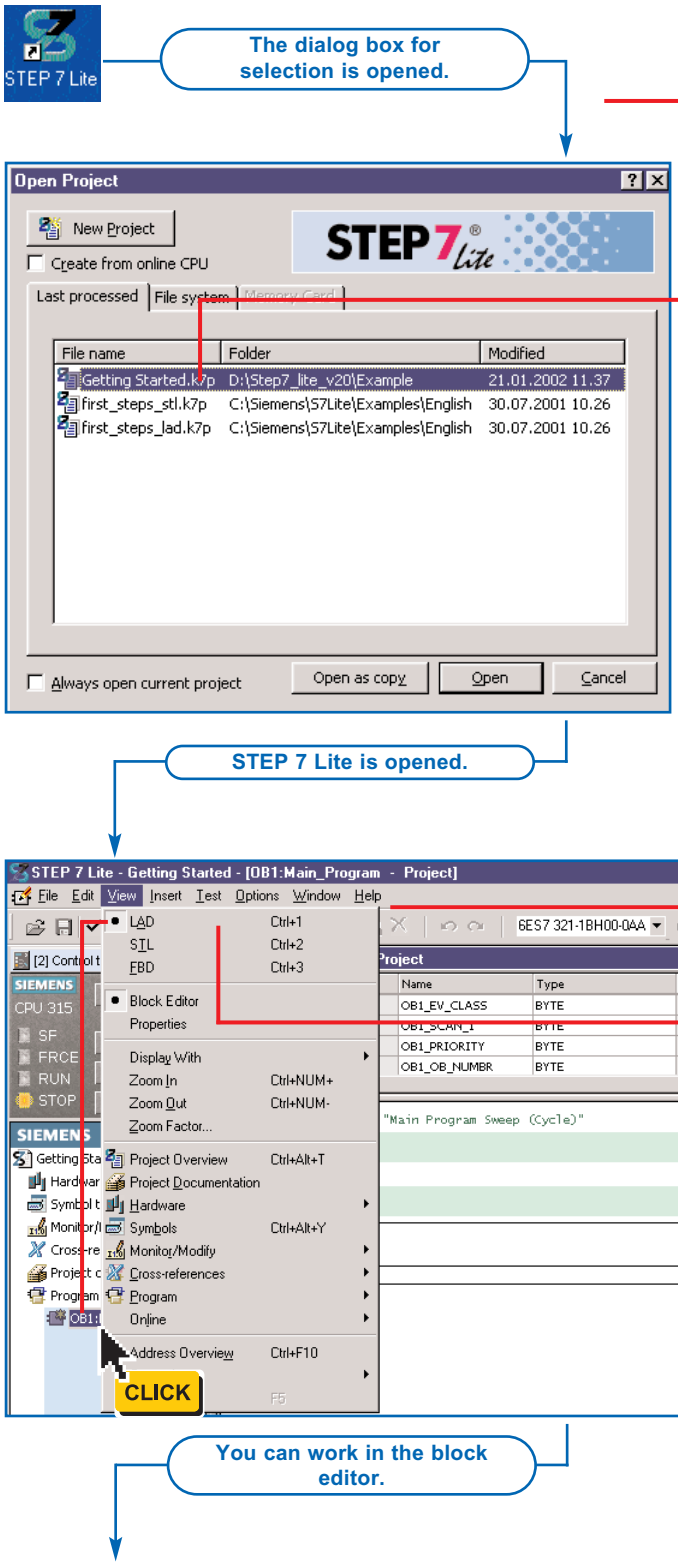
Page 6.12 to 6.17
Statement list (STL) is appropriate for users e.g. in the information industry.

Page 6.18 to 6.23
Function block diagram (FBD) is appropriate for users e.g. in circuit engineering.

In STEP 7 Lite you always create your program in the same programming interface, namely the block editor, regardless whether you choose LAD, FBD or STL. The user interface is adapted according to the programming language you have selected.

For example, if you choose to program in LAD, refer to the information on Page 6.6 to 6.11.

Opening OB1



1 Open STEP 7 Lite.

2 In the **Open project** dialog, click on the ”Getting Started.k7p” project.

You have already generated this project as described in Chapter 4 and filled out a symbol table for this project as described in Chapter 5.

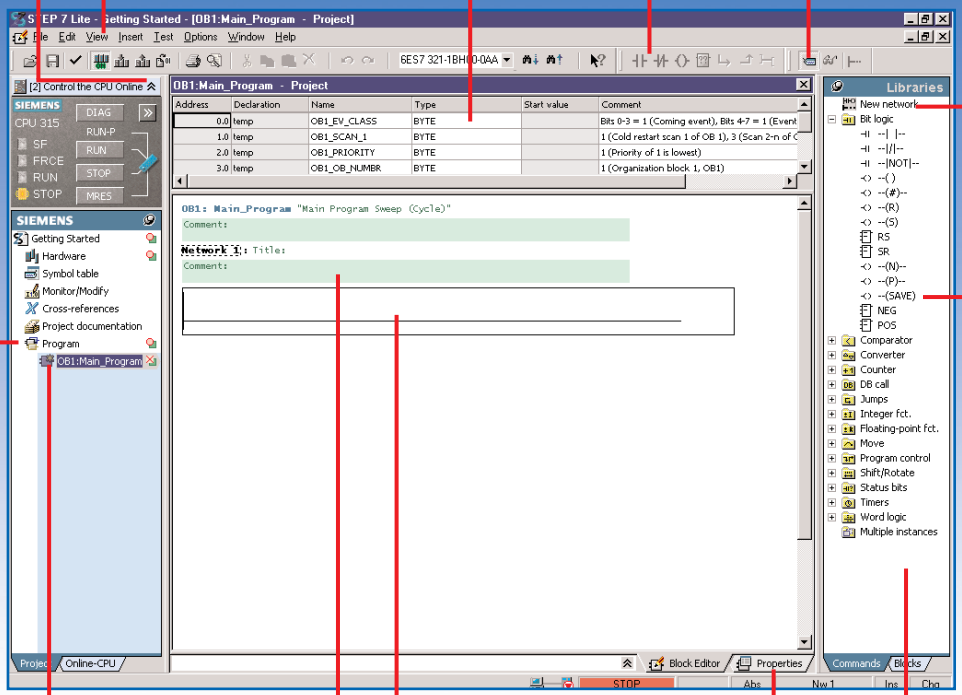
If this is not the case, simply open a ”New project” and copy the symbol table from one of the included sample projects.

3 Double-click on **OB 1** to open the block editor in your working area.

4 Click on **View**. This menu shows you the currently selected programming language; **LAD**, **FBD** or **STL**. Here you can also change the view.

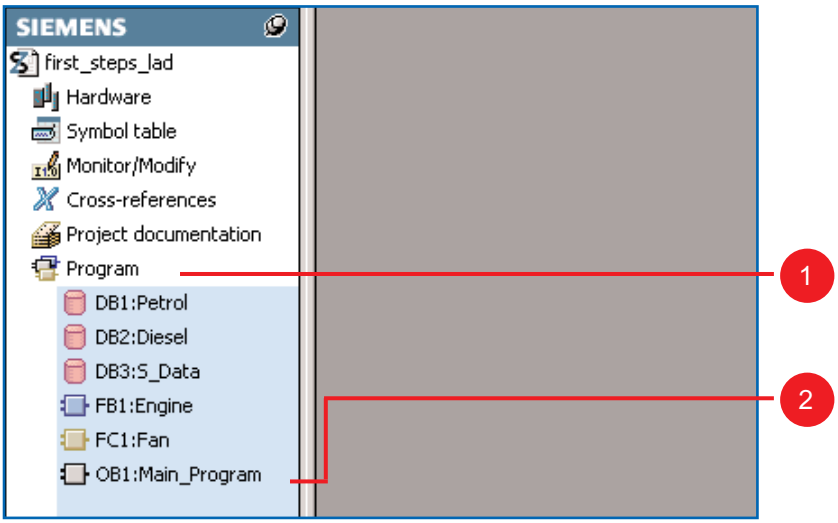
Note:
Some of the instructions cannot be displayed in all three programming languages. These are always displayed in STL.
You wil find information on the individual commands in LAD, FBD and STL via **F1 > Index > Language Descriptions**.

Working in the block editor



You always program blocks in the block editor.

We have inserted a view of LAD to represent the programming languages.



Project element “Program”

STEP 7 Lite user programs are split into blocks, thus allowing you easy management of large programming projects.

These blocks are displayed below the project element **Program**.

A new project only contains **OB1** that STEP 7 Lite generates automatically. Later on in your project, you are going to add other blocks, e.g.:

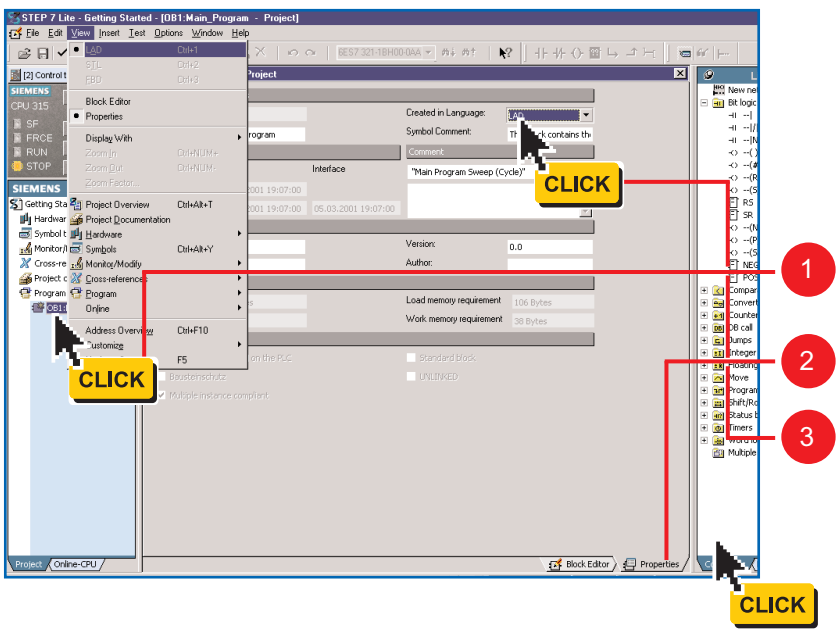
OB = Organization blocks
DB = Data blocks
FB = Function blocks
FC = Functions

The organization block OB 1 is the CPU’s operating system and it contains the main program. Additional blocks are mostly called in OB 1 and the specific parameters necessary for controlling the processs are assigned here too.

You can rename blocks using the **Rename** command of the pop-up menu.

Newcomers may require more information on working with blocks. Access this information with left-click on the project window and then press **F1 > Index > Blocks in the user program**.

Programming OB 1 in LAD

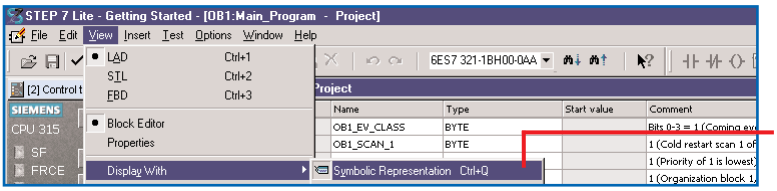


In the following section you will program a series circuit, a parallel circuit and the set/reset memory function in LAD (Ladder logic).

Specify the programming language to be used in programming and opening OB1 in future:

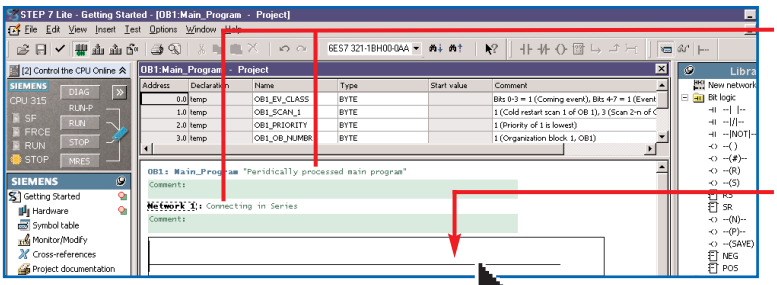
1. Double-click on **OB1**.
2. Click on **Properties**.
3. Select **LAD**. As of now, **OB1** will be opened in **LAD**.

Exit the **Properties** dialog box. LAD is now marked in the **View** menu, too.



5

Under **View**, select symbolic representation.



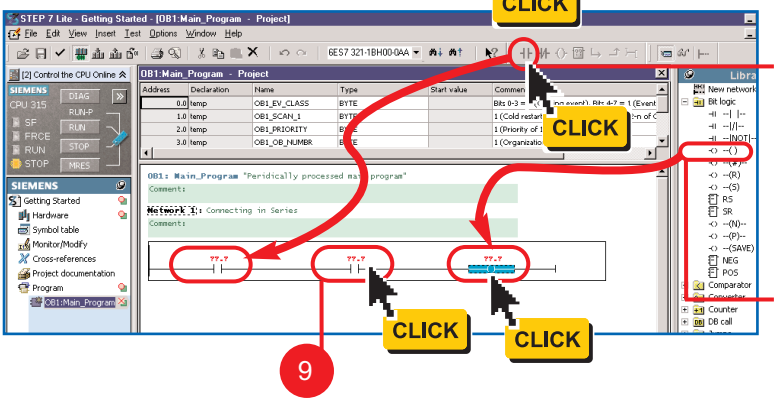
6

At **OB1**, enter “Periodically processed main program”. At **Network 1**, enter ”Connecting in Series”.

7

Click on the empty circuit to highlight it.

Insert three program elements, using different methods:



8

Click on the NO contact icon to insert it immediately.

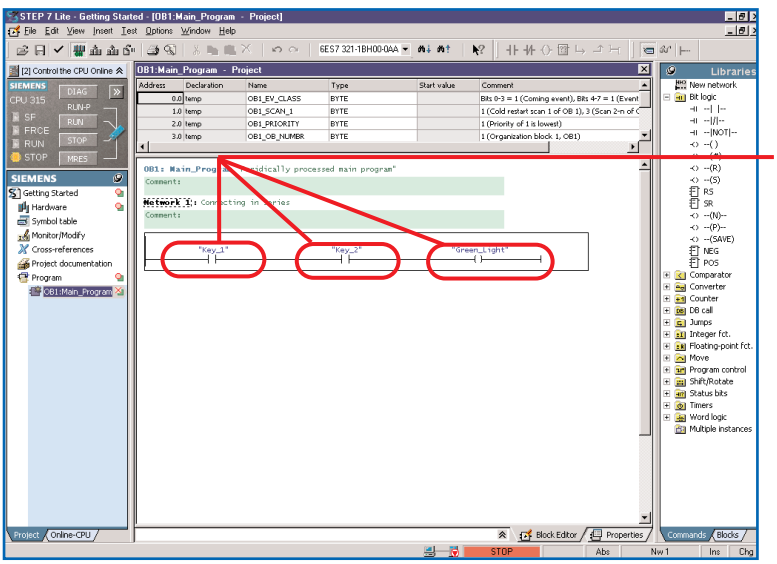
9

Right-click on the circuit to open the pop-up menu. Select NO contact.

10

Drag the coil to your circuit using the drag-and-drop operation.

The addressing of the NO contacts and the coil is still missing in the series circuit:



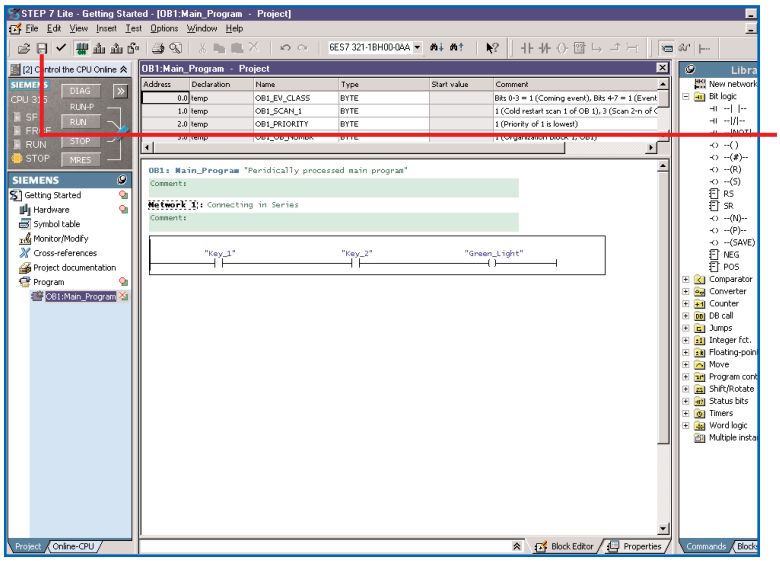
11

Click on **??.?**. Enter the symbolic name “Key_1” (with quotation marks). Or, click on ??.? to open the symbol selection list and select the name.

Confirm with **Return**.

Enter the symbolic name “Key_2” for your second NO contact.

Enter the coil name “Green_Light”.



12

The series circuit program is completed.

If no other icons are displayed in red color, click on the disk icon to save your entries.

Not only your entries in OB1 are saved, but rather all project elements.

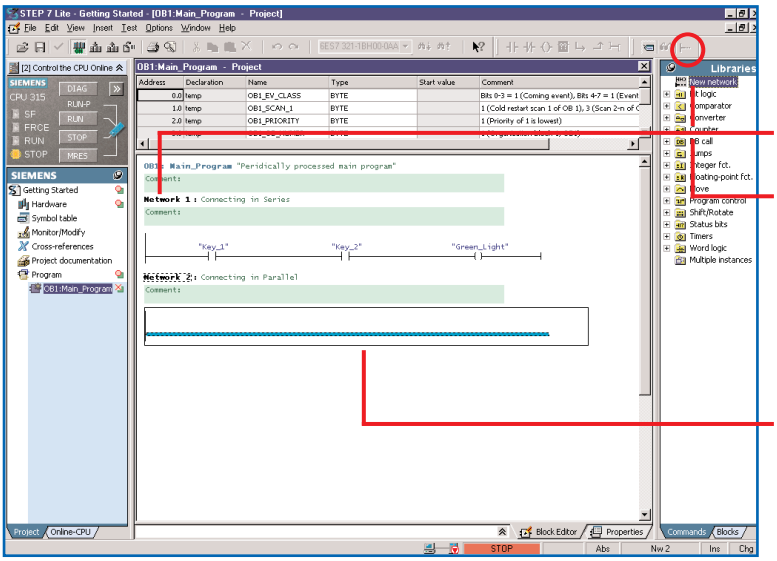
13

When you select **Edit > Apply**, your entries (always the content of the active window) are saved to a temporary file.
This file saving method is recommended in case you intend to undo changes later. After applying data, you are prompted to save your changes when you close the project.

Symbols are displayed in red color, for example, if not included in the symbol table or if a syntax error has occurred.

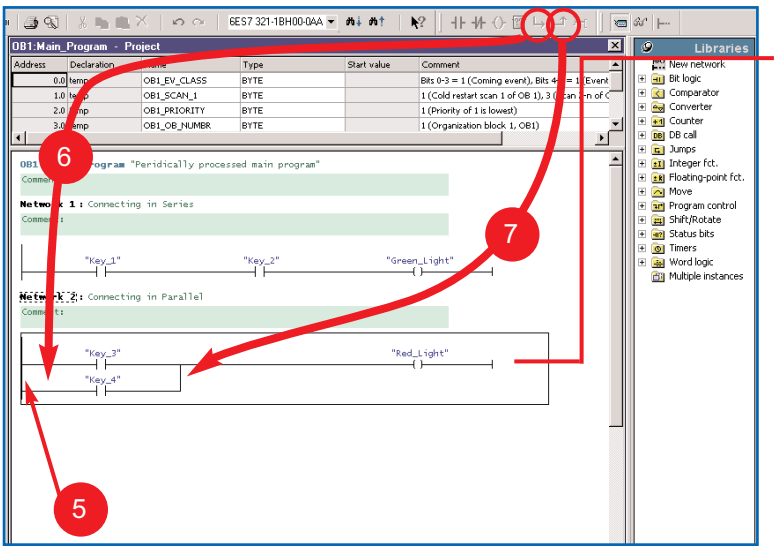
In this case you cannot save your entries, and the lower section of the editor displays a plain text message informing you of appropriate procedures.

How to program a parallel circuit in LAD



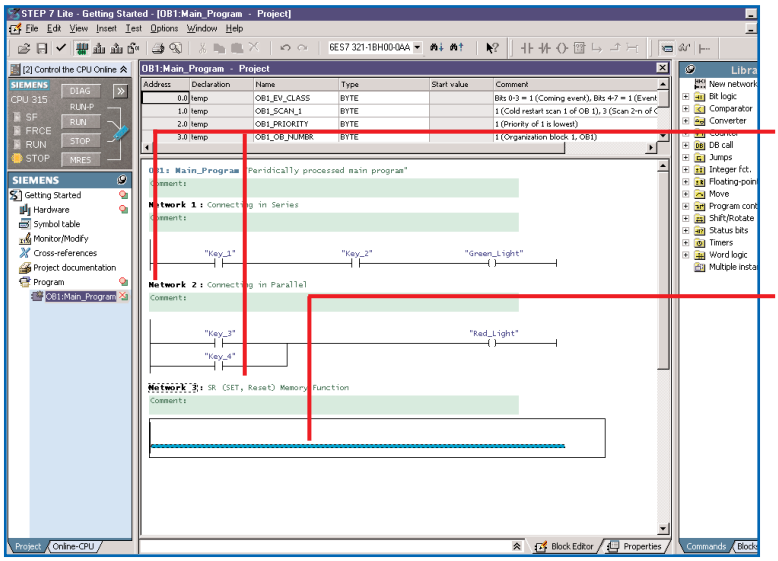
1. Highlight network 1.
2. Insert a new network.

   This action can also be performed via an icon in the toolbar, via context-sensitive menu or CTRL+R.
3. Again, highlight the circuit.



4. Insert a NO contact and a coil. Name them ”Key_3” and ”Red_Light”.
5. Highlight the left circuit.
6. Insert a parallel branch and add another NO contact to it.
7. Close the branch-off via icon or dragging the double arrow tip that is visible after you have inserted the NO contact.
8. The only thing our parallel branch is now still missing is addressing. Enter a name as shown in the figure. Save your entries.

Assign distinctive short-names to your circuits. This makes it easier to scroll through large programs via scroll bar on the right side of the window. The names are displayed when you scroll the view.

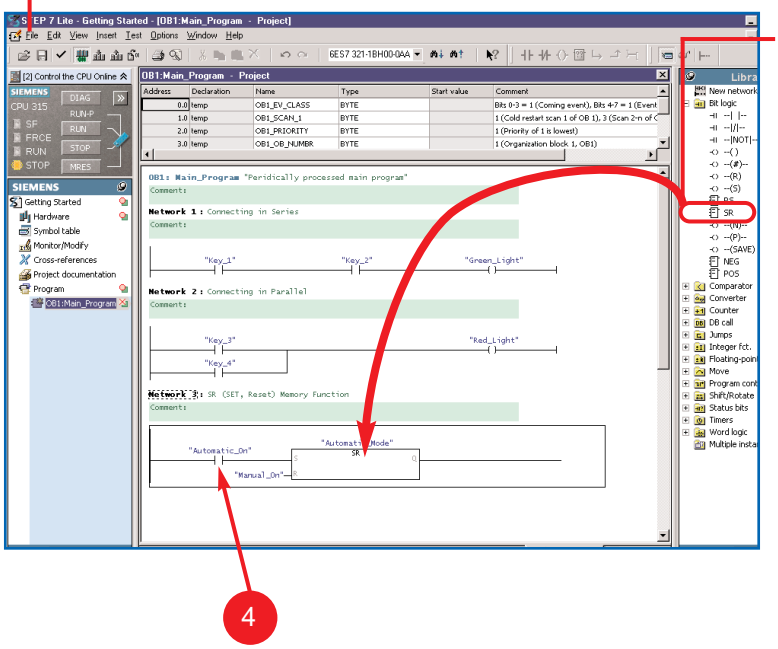


How to program memory functions in LAD

1 Highlight network 2, insert another network and enter the title ”SR Memory Function”.

2 Again, highlight the circuit.

6



3 In the Command tab, go to **Bit logic** and **SR element**. Insert this element.

4 Insert a NO contact upstream of input S.

5 Enter following symbolic names:

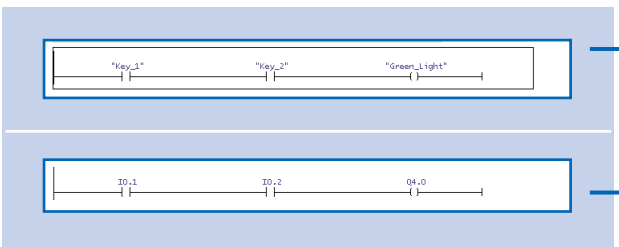
– Upper NO contact: ”Automatic_On”,
– Input R: ”Manual_On”,
– SR element: ”Automatic_Mode”.

6 Save your entries with **File > Save**.

How to adapt the programming interface

You can use STEP 7 Lite menu commands to customize your programming interface.

View menu - Examples:

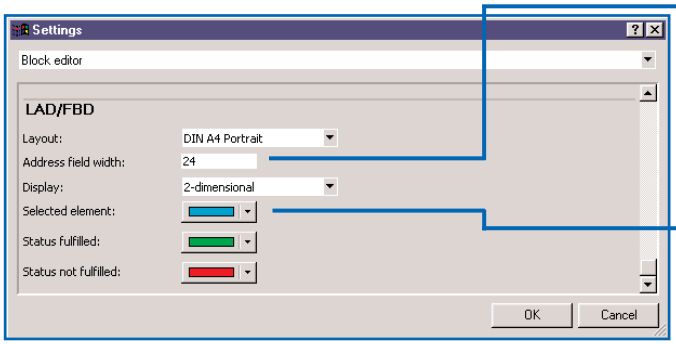


1 For symbolic addressing in LAD: Enable **View > Display with > Symbolic Representation.**

2 For absolute addressing in LAD: Disable **View > Display with > Symbolic Representation**

Changing the programming language: **View > LAD/FBD/STL**

Menu command **Options > Settings** - Example:



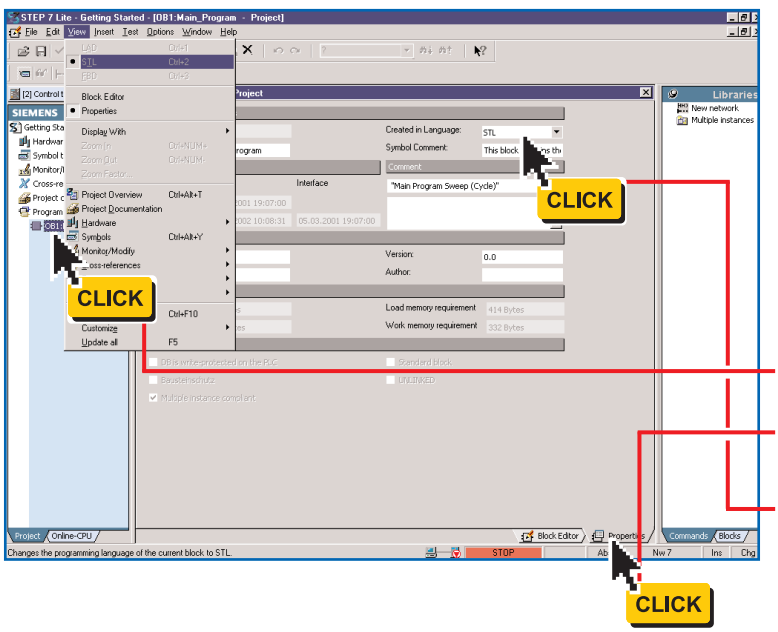
3 Specify a line break in symbolic addressing, between the 10th and 24th character: **Options > Settings > LAD/FBD > Address field width**

4 Modify the color setting of a circuit: **Options > Settings > LAD/FBD > Selected element**

7 Close the block via Windows icon **Close**.

Especially menu item **Options > Settings** offers a wide range of options for the customization of colors, fonts, address field width etc. of the STEP 7 Lite user interface.

Programming OB1 in STL

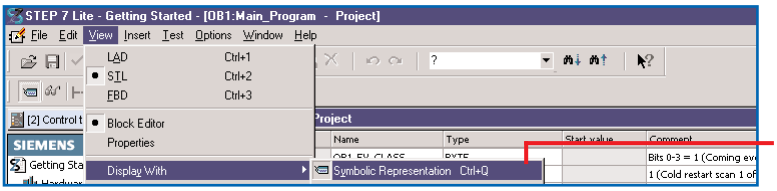


Using the programming language STL (Statement list), you are now going to program an OR and an AND instruction, as well as a set/reset memory functions.

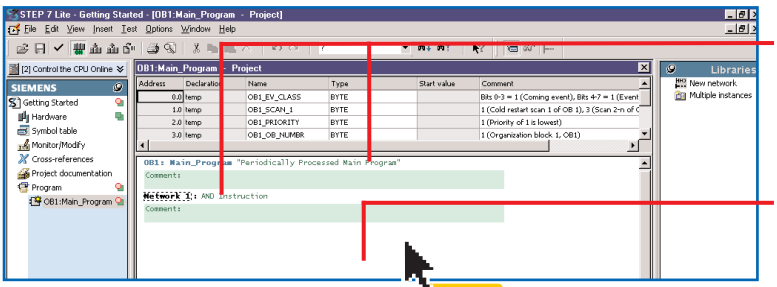
Specify the language you want to use to program and subsequently open OB1:

1. Double-click on **OB1**.
2. Click on **Properties**.
3. Select **STL**. As of now, OB1 will be opened in STL.
4. Exit the **Properties** dialog. **STL** is also marked in the **View**.

How to program AND instructions in STL

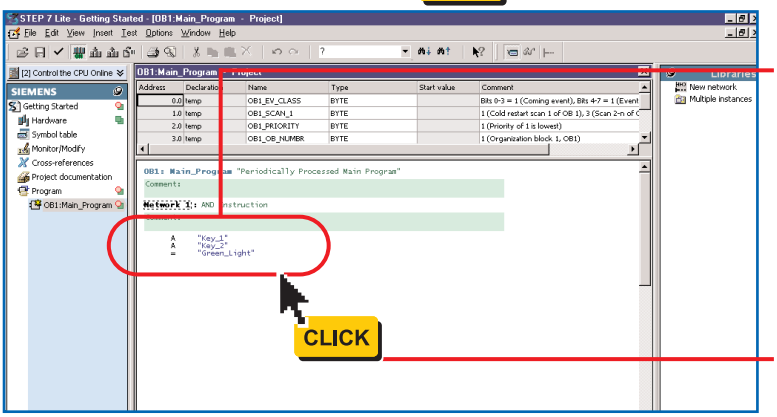


5 Select Symbolic representation from the View menu.



6 At **OB1**, specify ”Periodically Processed Master Program”. At **Network** 1, enter ”AND Instruction”.

7 Click on the input area.



8 Enter ”A” (for AND) in the first program line, followed by an empty string and the ”Key_1” symbol (with quotation marks).

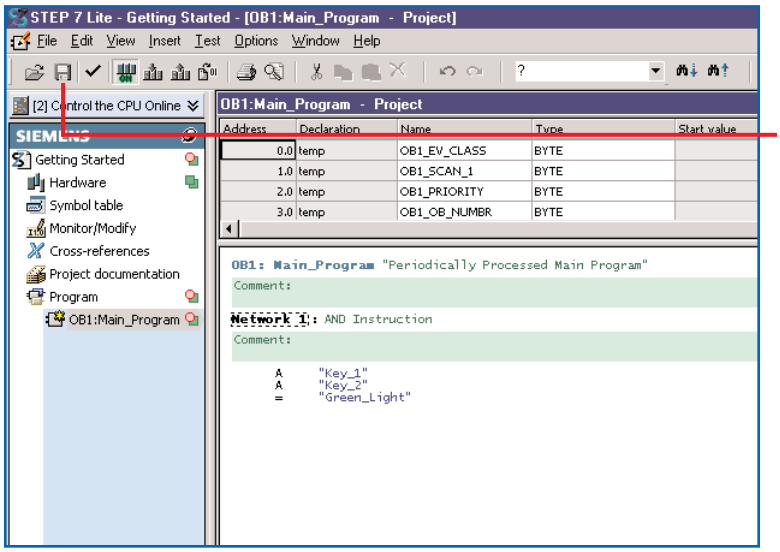
Close the row with Return.
The cursor moves to the next row.

9 In the next row, enter an ”A” once again. This time, right-click on the input area.

Right-click to open the pop-up menu. Select the menu command **Insert symbol**. Select ”Key_2” from the list and insert it.

In the next line, enter ”=”, and then ”Green_Light” either via the keyboard or the pop-up menu.

You do not have to start making your entries at the beginning of an input row. No matter where you start, STEP 7 Lite arranges the instructions clearly and in rows.



11

The program for your AND instruction is completed.

If no other text is highlighted in red color, click on the disk icon to save your entries.

Not only your entries in OB1 are saved, but rather all project elements.

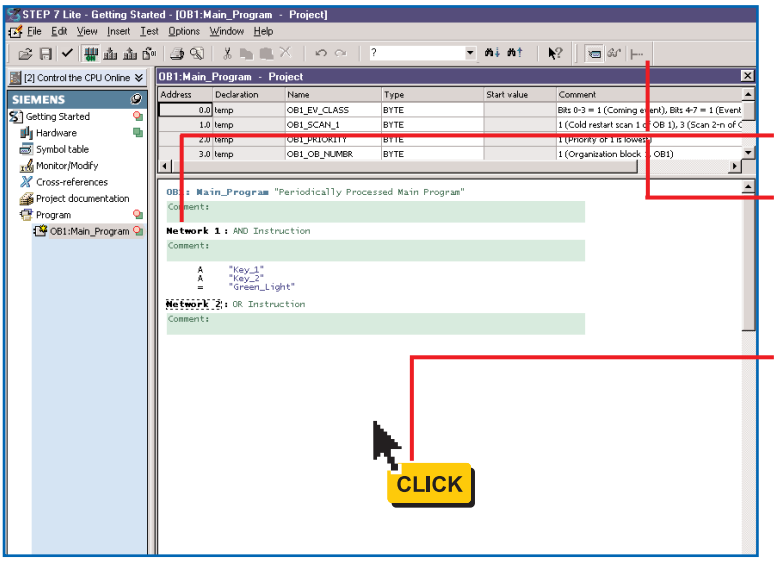
12

When you select **Edit > Apply**, your entries (this is always the content of the active window) are saved to a temporary file.
This file saving method is recommended in case you want to undo changes later.
After applying only data, you are prompted to save your changes when you close the project.

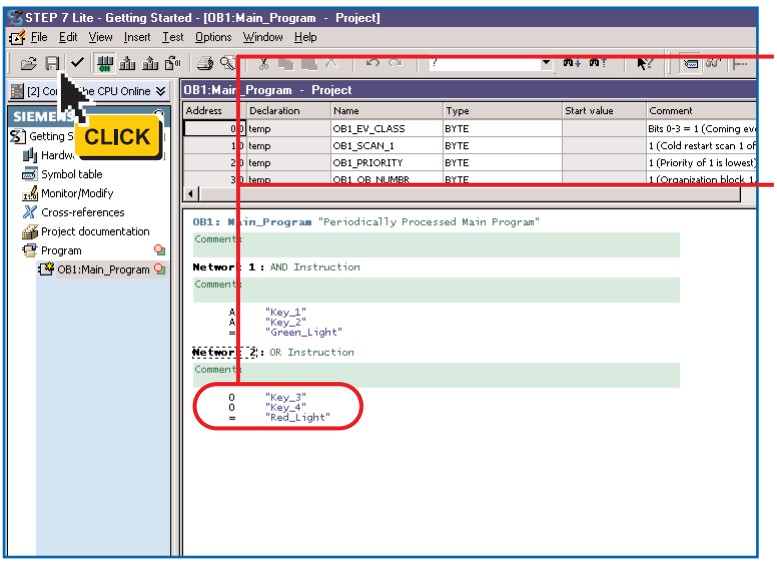
Symbols are displayed in red color, for example, if not included in the symbol table or if a syntax error has occurred.

In this case, you cannot save your entries and the lower section of the editor displays a plain text message informing you of appropriate procedures.

How to program an OR instruction in STL

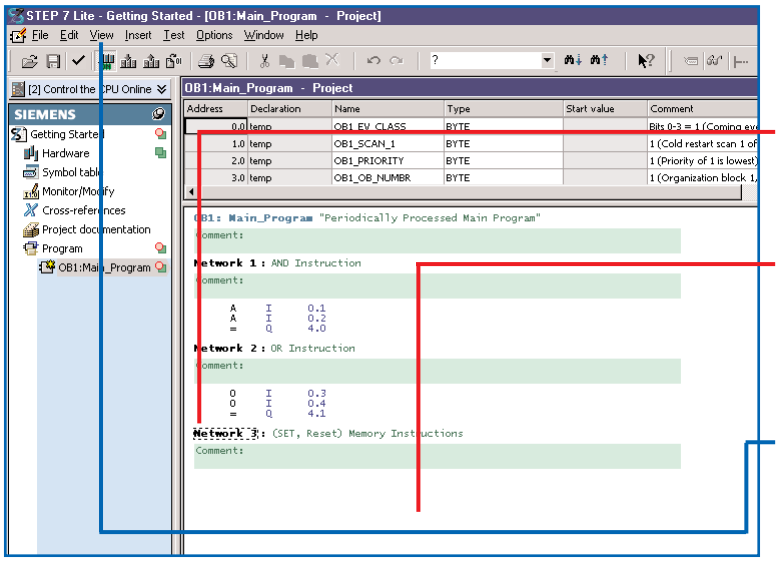


1. Highlight network 1.
2. Insert a new network.
   This action can also be performed via icon in the toolbar and via right-click or CTRL+R.
3. Again, click on the input area.



4. Enter an ”O” (for OR), followed by the ”Key_3” symbol (analog to AND).
5. Complete the OR instruction and save your entries.

Assign distinctive short-names to your circuits to make it easier to scroll through large programs via scroll bar on the right side of the window. The names are displayed when you scroll the view.

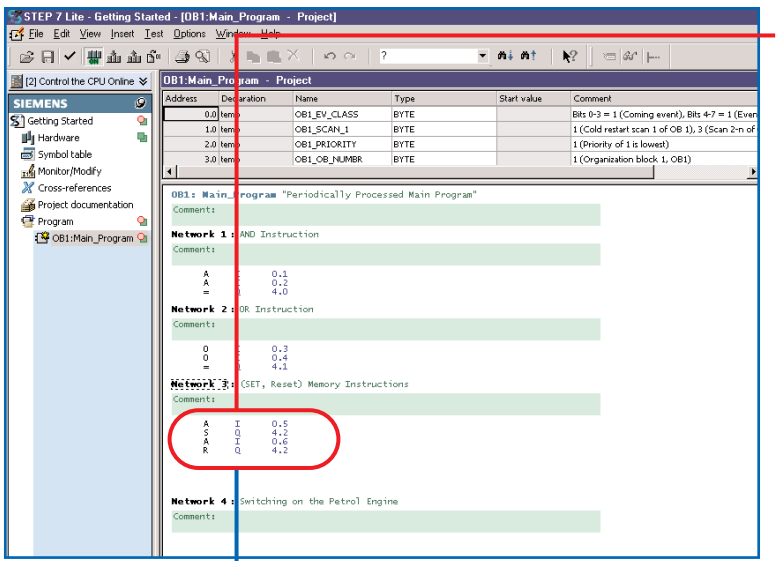


How to program memory instructions in STL

1 Highlight network 2. Add another network.

2 Again, click on the input area.

3 In **View > Display with > Symbolic Representation**, you can switch between symbolic and absolute representation.



4 In the first line, enter an "A" instruction with the symbolic name "Auto mode ON". Complete your memory instruction as follows:

Symbolic:

A "Automatic_On"
S "Automatic_Mode"
A "Manual_On"
R "Automatic_Mode"

5 Save your entries via **File > Save**.

6 If you switched the view in step 3, enter the following:

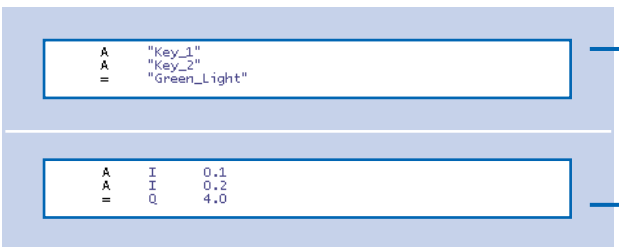
Absolute:

A I0.5
S Q4.2
A I0.6
R Q 4.2

How to adapt the programming interface

STEP 7 Lite allows you to customize your programming interface.

Menu **View** menu - Examples:

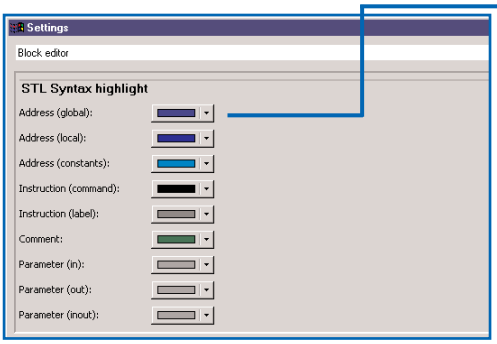


1 For symbolic addressing in STL: Enable **View > Display with > Symbolic Representation**

2 For absolute addressing in STL: Disable **View > Display with > Symbolic Representation**

Changing the programming language **View > LAD/FBD/STL**

Menu command **Options > Settings** - Example:

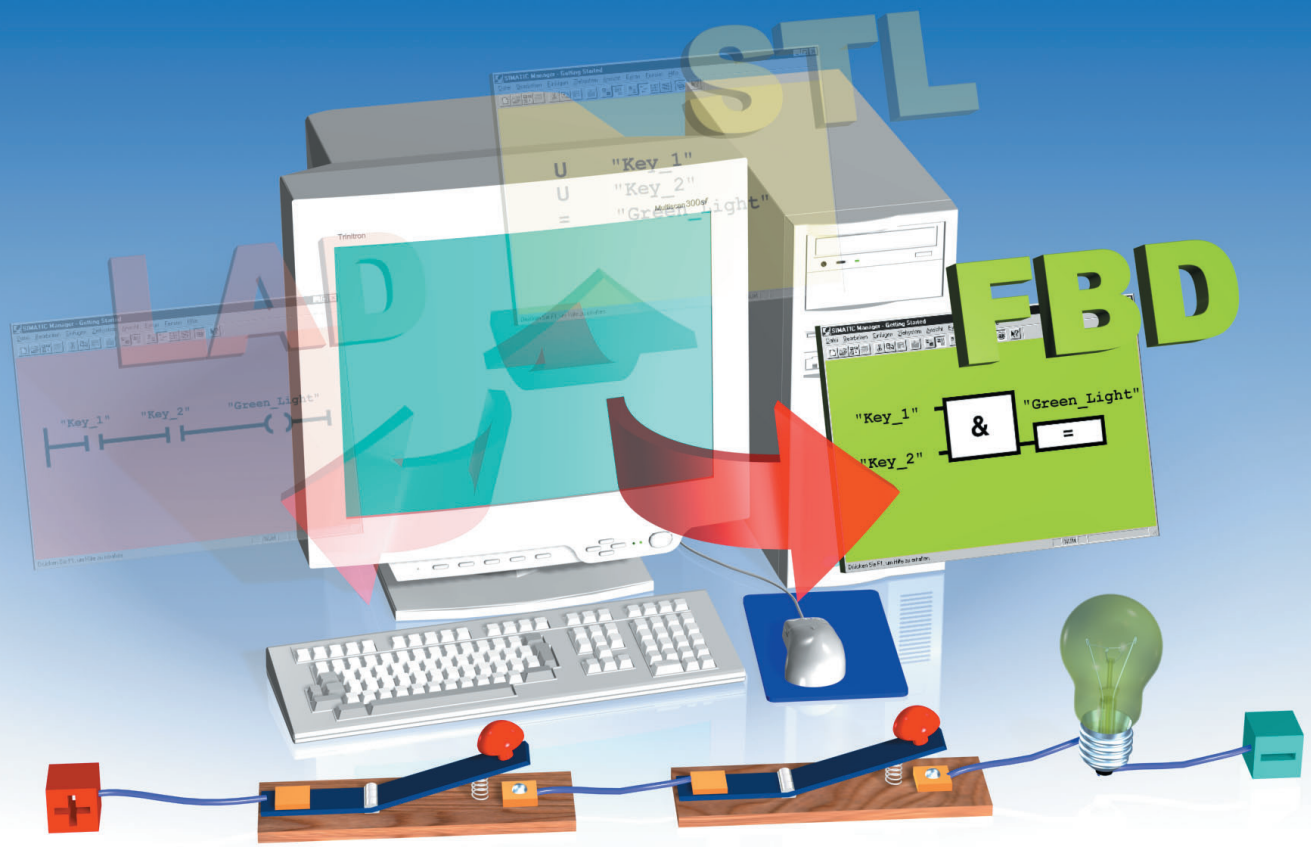


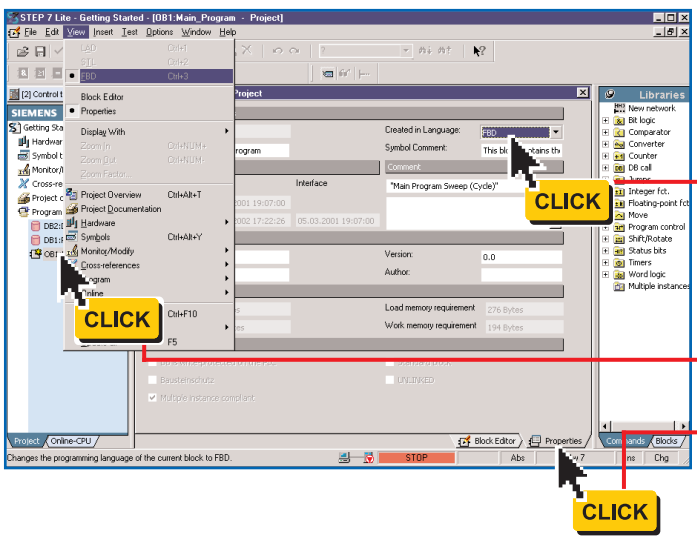
3 Changing the color of instructions: **Options > Settings > STL Syntax highlight**

7 Close the block via Windows icon **Close**

The menu item **Options > Settings** offers many options for the customization of STEP 7 Lite colors, fonts etc.

Programming OB1 in FBD



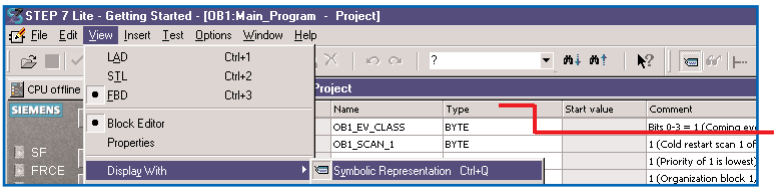


Using FBD (function block diagram), you are now going to program AND/OR/memory functions.

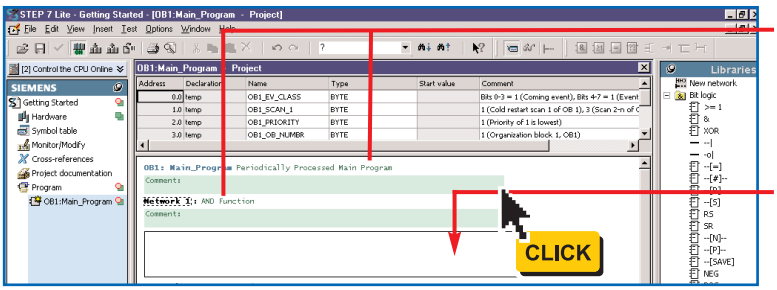
Specify the language you want to use to program and subsequently open OB1:

1. Double-click on **OB1**.
2. Click on **Properties**.
3. Select **FBD**. As of now, OB1 will be opened in FBD.
4. Exit the **Properties** dialog box. **FBD** is also marked in the **View** menu.

How to program AND functions in FBD



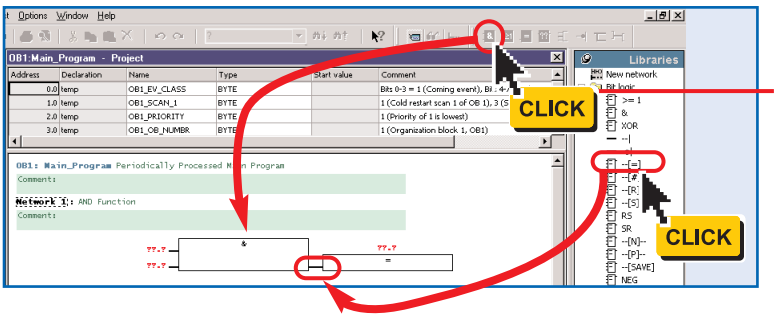
5 Under the **View** menu, select symbolic representation.



6 At **OB1**, enter ”Periodically Processed Main Program”. At **Network** 1, enter ”AND Function”.

7 Click on the input area.

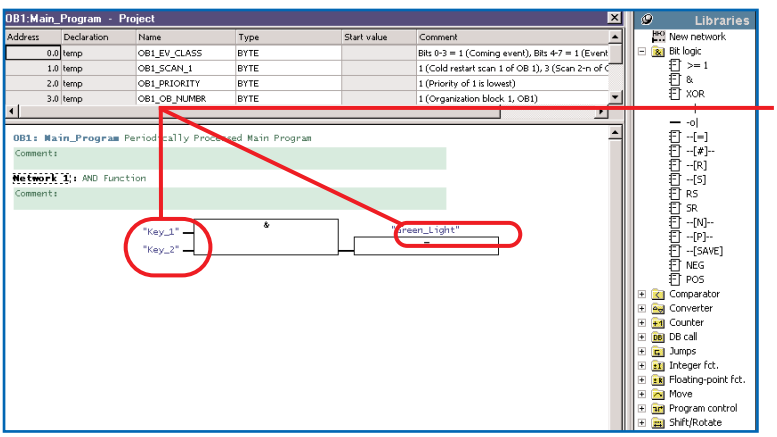
Use different methods to insert two program elements:



8 Click on the AND icon to insert it immediately.

9 Drag the instruction to the graphic frame. If you miss this target, the instruction is indicated below the AND box.

10 Alternative to pasting per drag-and-drop: Highlight the frame and double-click on the assign icon.

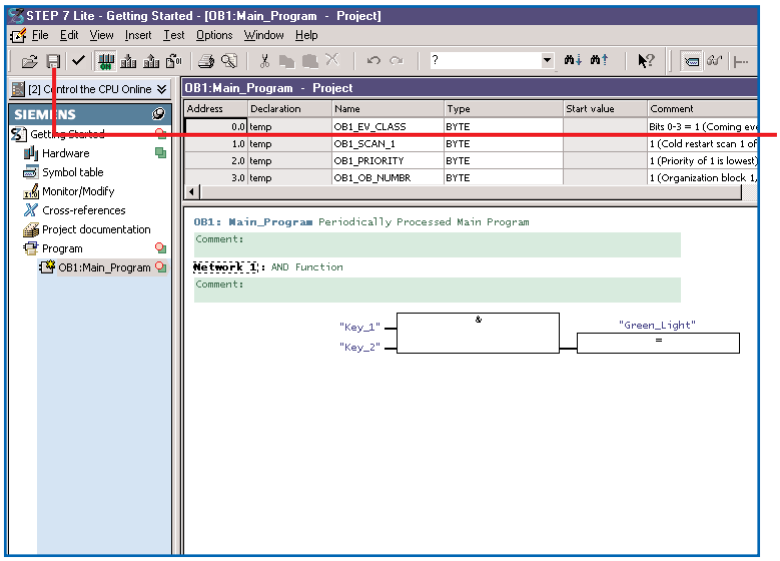


Now, the only thing still missing in your AND function is addressing:

11 Click on **??.?**. Enter the symbolic name ”Key_1” (with quotation marks). Or, click on the question mark, right click to select **Insert symbol** and then insert a name from the list.

Enter “Key_2” for the input of the AND box.

Assign the name ”Green_Light” to the function.



12

The AND function program is completed.

After no more symbols are displayed in red color, click on the disk icon to save your entries.

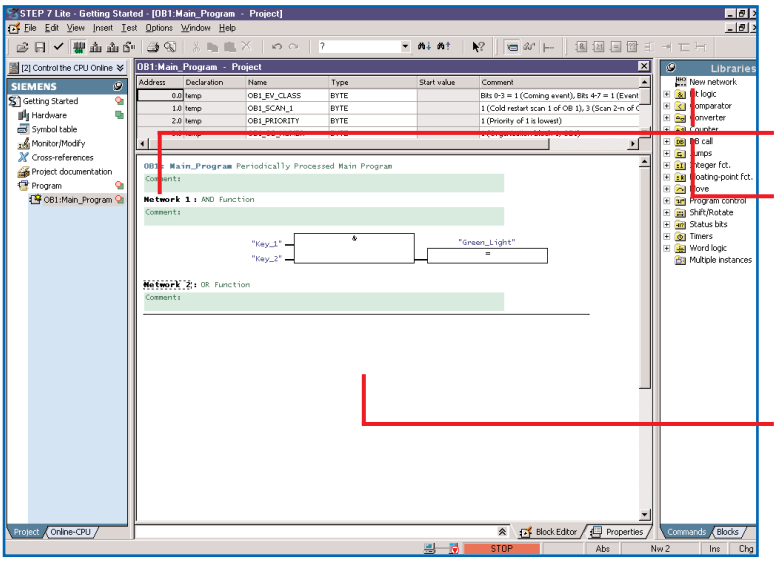
Not only your entries in OB1 will be saved, but rather all project elements.

When you select **Edit > Apply**, your entries (this is always the content of the active window) are saved to a temporary file.
This file saving method is recommended in case you want to undo changes later. After entering only data, you are prompted to save your changes when you close the project.

Symbols are displayed in red color, for example, if not included in the symbols table or if a syntax error has occurred.

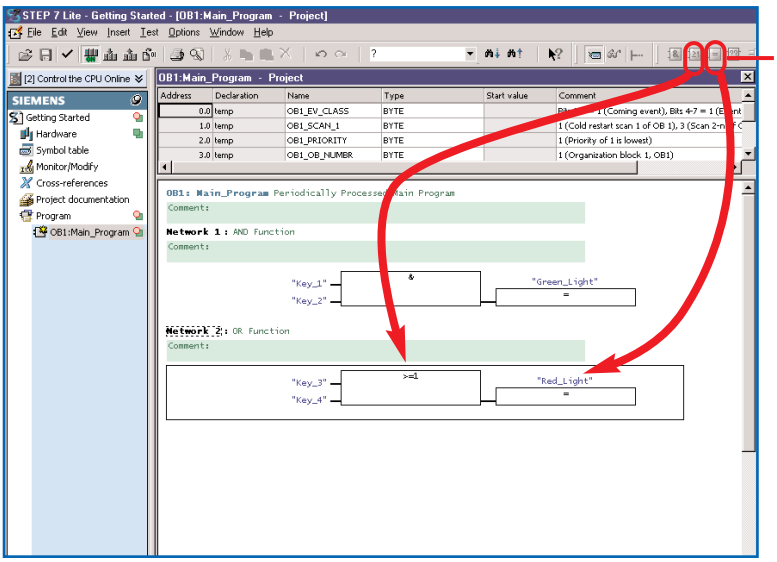
In this case, you cannot save your entries and the lower section of the editor displays a plain text message informing you of appropriate procedures.

How to program OR functions in FBD



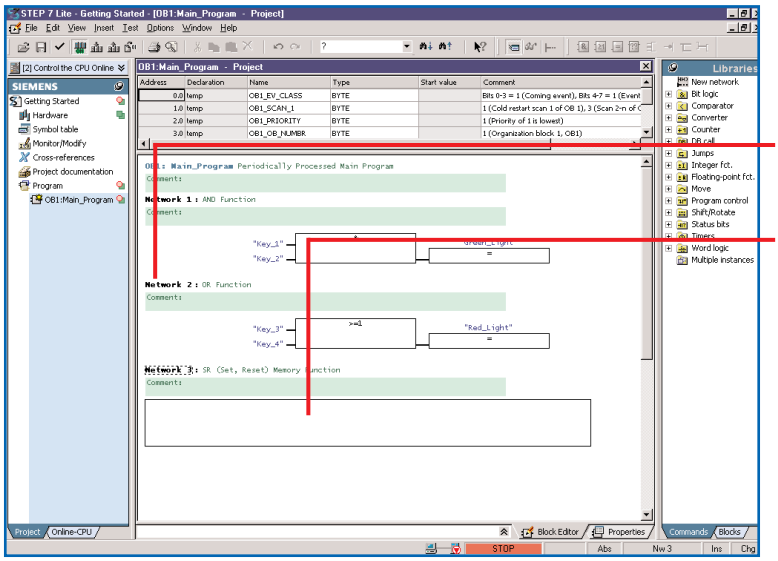
1. Highlight network 1.
2. Insert a new network.

   This action can also be performed via icon in the toolbar and with right-click or CTRL+R.
3. Again, highlight the input area.



4. Insert an OR function and an assign instruction.
5. Only addressing is still missing. Enter a name as shown in the left figure. Save your entries.

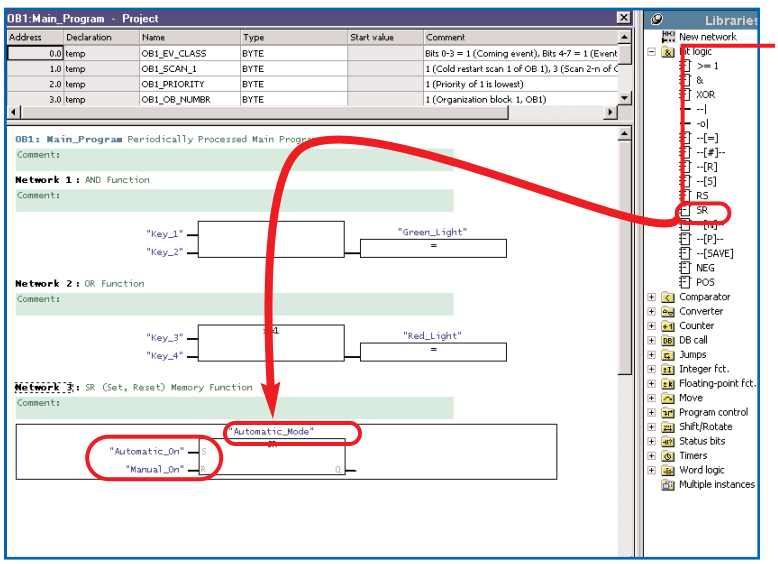
Assign distinctive short-names to your circuits to make it easier to scroll through large programs via scroll bar on the right side of the view. These names are displayed when you scroll the view.



How to program memory functions in FBD

1 Highlight network 2. Insert an additional network.

2 Again, highlight the input area.



3 In the instruction list, go to **Bit logic** and select **SR element**.

Insert the element.

4 Enter following symbolic names:

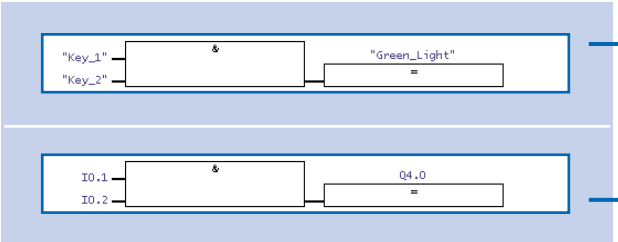
- Set ”Automatic_On”,
- Reset ”Manual_On”,
- Set/Reset memory function ”Automatic_Mode ”.

5 Save your entries via **File > Save**.

How to adapt the programming interface

The STEP 7 Lite menu commands allow you to customize your programming interface.

Menu **View** – Examples:

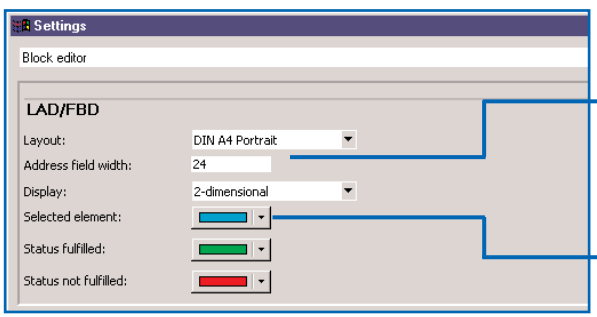


1 For symbolic addressing in FBD: Enable **View > Display with > Symbolic Representation**

2 For absolute addressing in LAD: Disable **View > Display with > Symbolic Representation**

Changing the programming language **View > LAD/FBD/STL**

Menu **Options > Settings** – Examples:



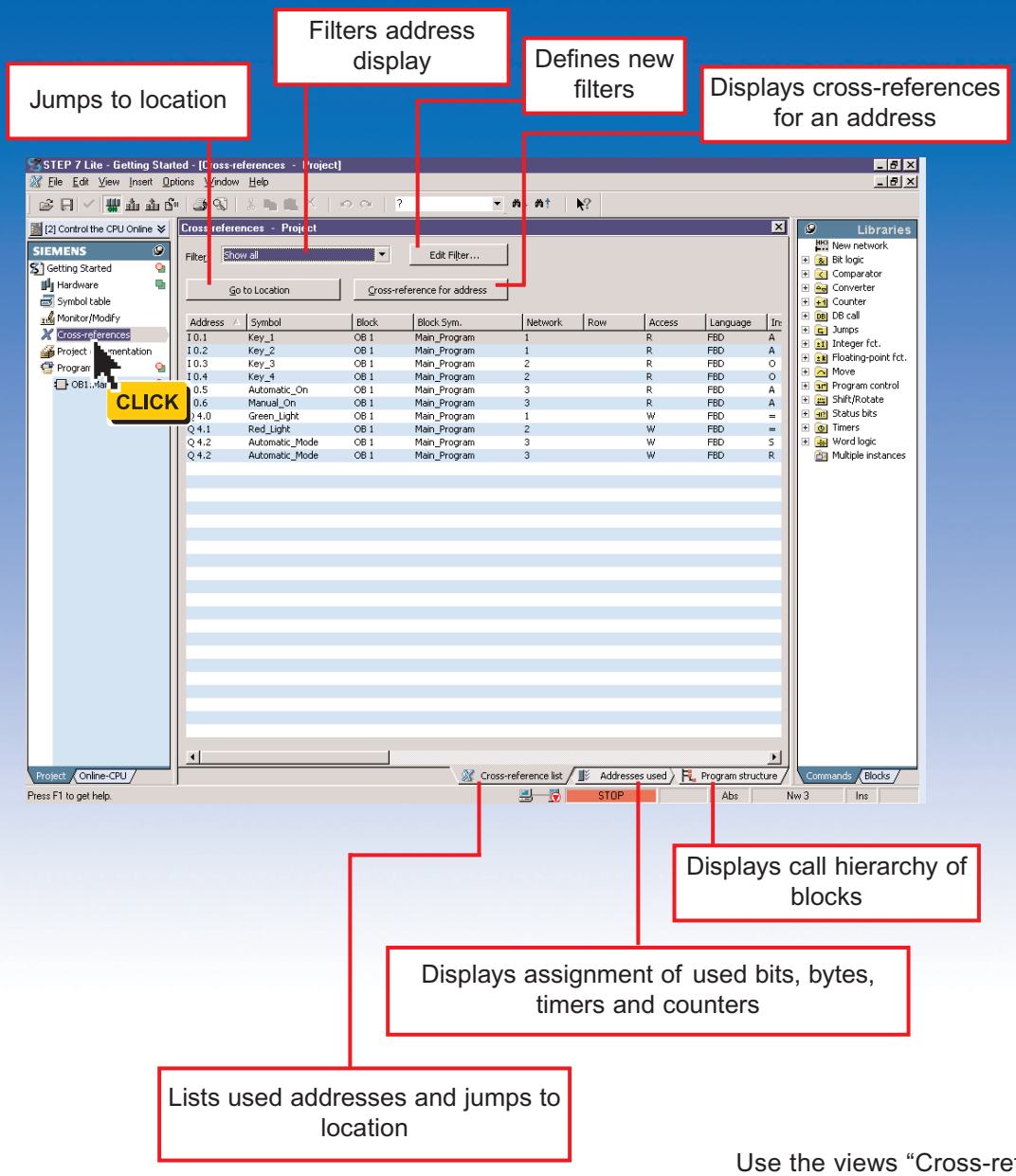
3 Line break in symbolic addressing, between the 10th and 24th character: **Options > Settings > LAD/FBD > Address field width**

4 Changing the view colors: **Options > Settings > LAD/FBD > Selected element**

6 Close the block via Windows icon **Close**.

Especially the **Options** menu offers many options for the customization of colors, fonts, address field width etc. in STEP 7 Lite.

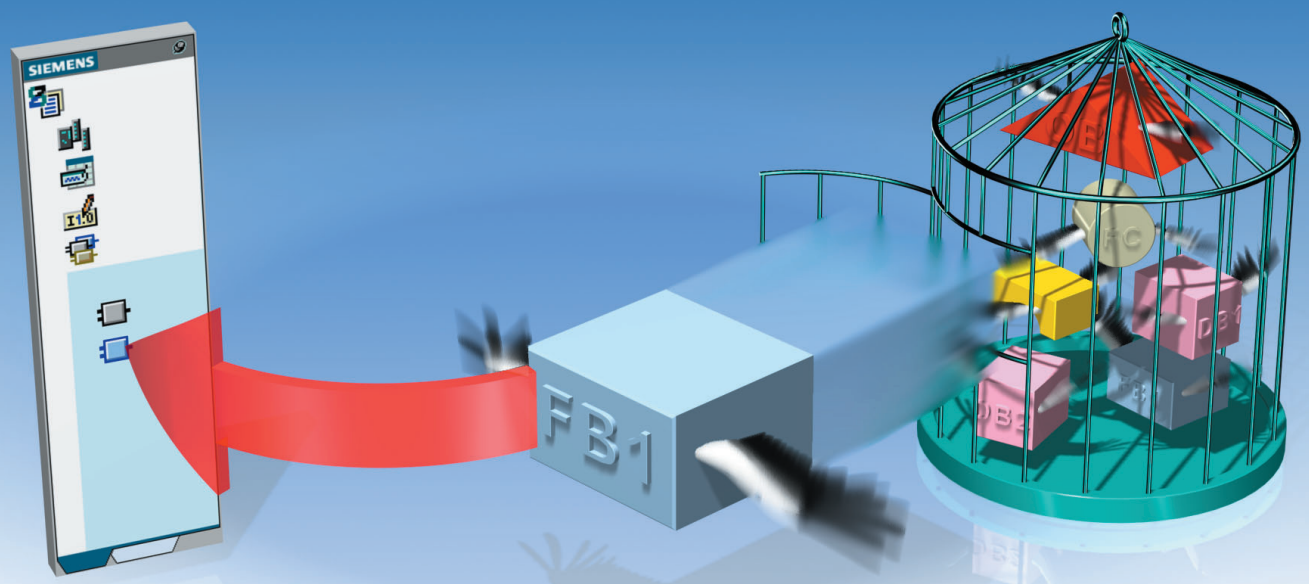
Displaying cross-references



Use the views “Cross-reference list”, “Addresses used” and “Program structure” to get an overview of how to use addresses, memory areas, blocks etc.
You can get to the cross-references by double-clicking the “Cross-references” symbol in the project window.

7 Using function blocks

Generating and opening function blocks (FB)



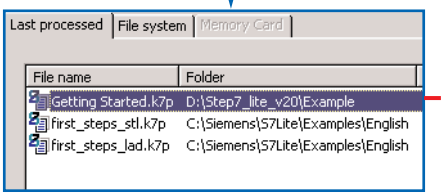
Function blocks are used in function programming if you have to save intermediate results or operating settings and operating modes until the next call. For this reason, they are also referred to as ”Blocks with memory”.

In your sample project, you program function block FB1 under the symbolic name ”Engine”.
Use the programming language you used in programming OB1.

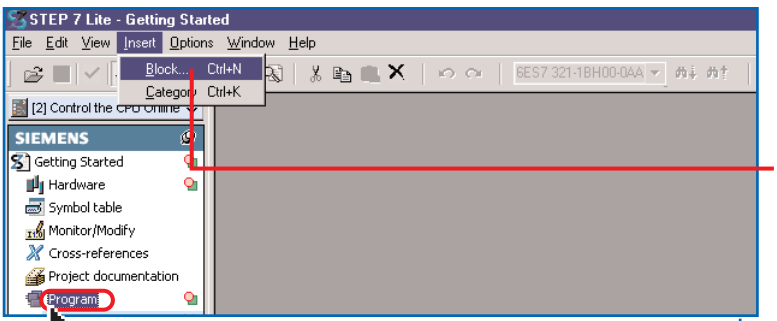
Your ”Getting Started” project should contain a copy of the symbol table before you continue with this chapter (see Page 5.5).

The dialog box for project selection is opened.

1 If required, open STEP 7 Lite.



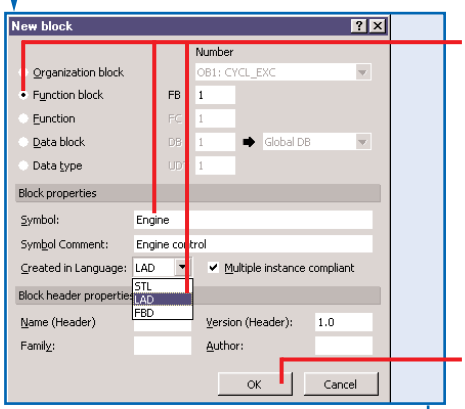
2 In the **Open project** dialog, double-click on "Getting started" to open the project.



3 In your project window, click on Program.

4 Select menu command **Insert > Block**, or right-click to open the pop-up menu and select the **New > Block** command.

The dialog box for creating new blocks is opened.

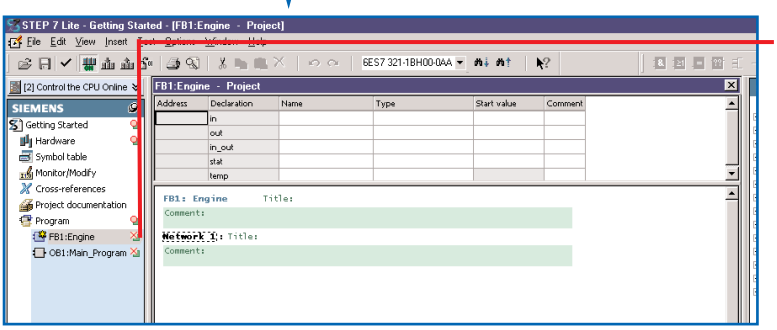


5 Highlight **Function block.**

Select the language you want to use for your programming in the **Created in language** drop-down list box.

6 Confirm with **OK**.

The block is inserted and opened immediately.

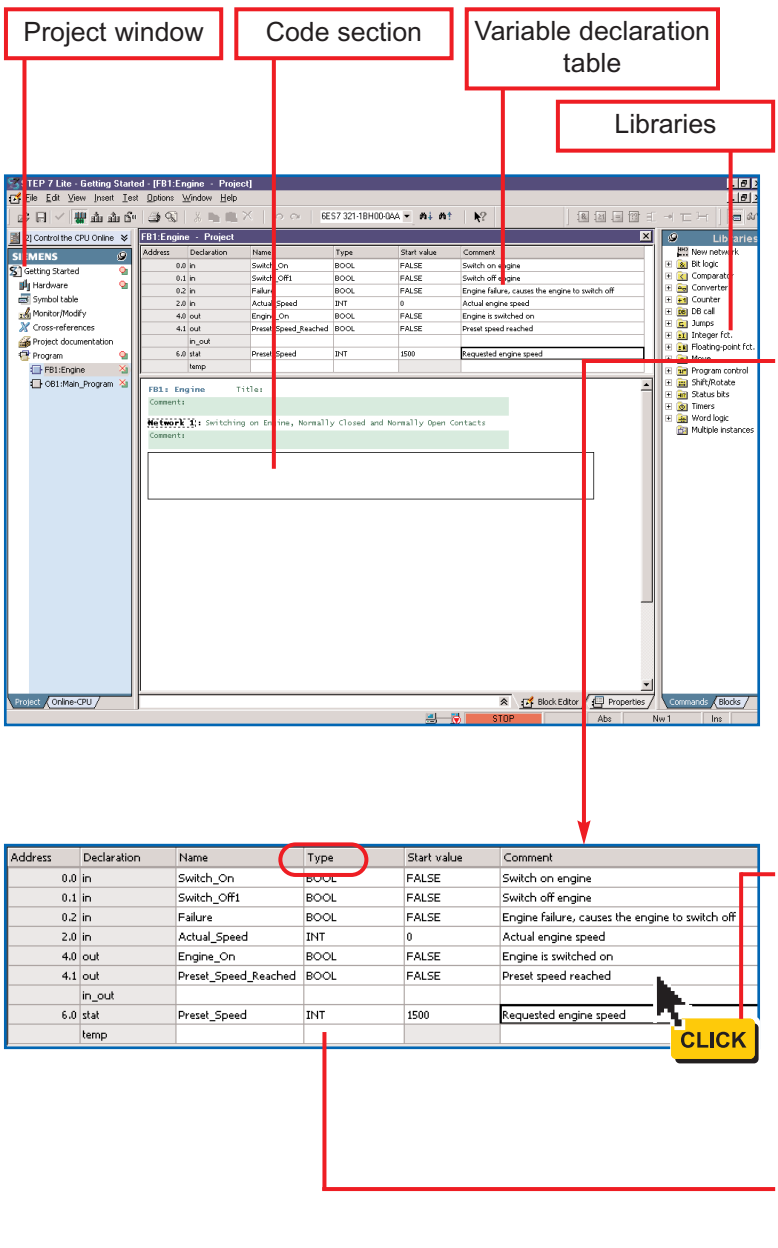


7 The new block is inserted and opened immediately in your project window.

How to edit the variable declaration table

We are going to show you how to program a function block that uses two data blocks to monitor and control a petrol and a diesel engine.

All ”engine-specific” signals, namely the block parameters, are transferred between the organization block and the function block.
Thus, they must be declared in the variable declaration table as I/O parameters (Declaration ”in” and ”out”). This defines the ”Interface” for calling your function block in the program.



Address	Declaration	Name	Type	Start value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off1	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

Next thing to do is to edit the variable declaration table of the FB, before you continue to program instruction elements.

Enter the variables as shown in the figure.

Click on a cell and enter a respective name and comment as shown in the figure.

Select **Type** from **Elementary types** via the pop-up menu by right-clicking the mouse button.

Press Return to move the cursor to the next column or to insert a new row.

1. Editing the variable declaration table
Only letters, numeral and underscore can be used to name block parameters in the variable declaration table.

2. Help on the variable declaration table?
More information is found via **F1 > Content > Programming blocks > Creating logic blocks** and **Editing the variable declaration table**.

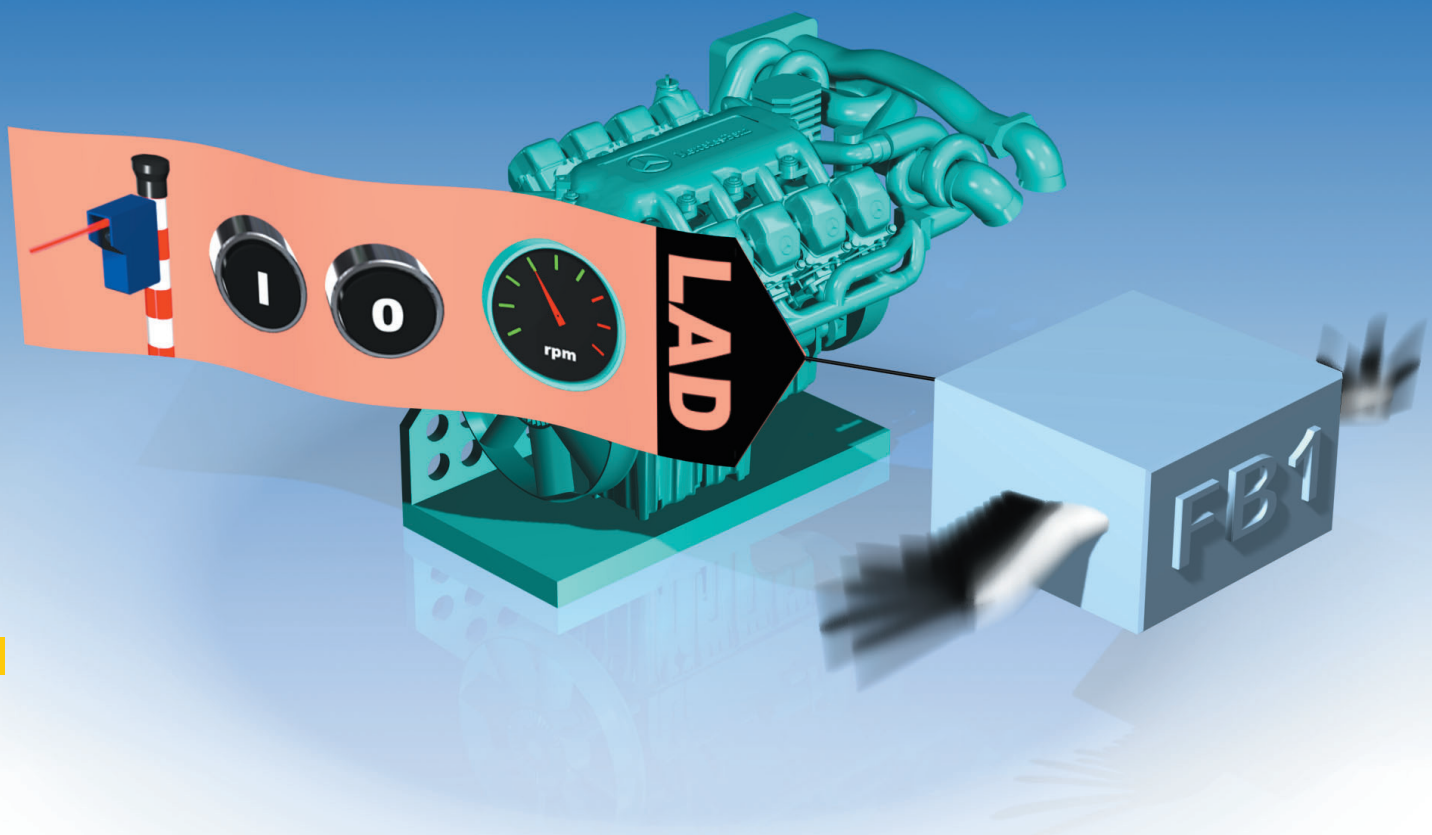
3. Tips on the chapters below
In the following chapters you are going to program an ON/OFF switching circuit and a speed monitoring circuit.

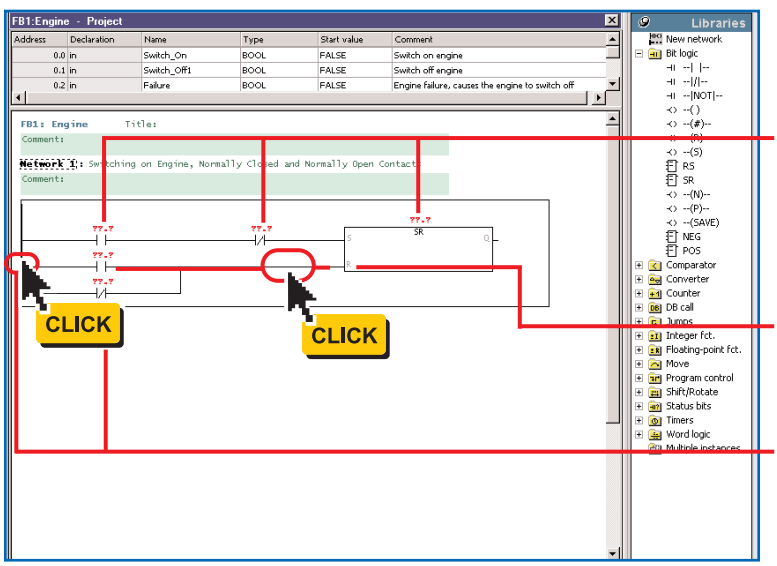
When is the engine switched on and off?
- The engine is switched on if the signal status of variable #Switch_On is ”1” AND if the signal status of variable ”Automatic_Mode” is ”0”.
- The engine is switched off if the signal status of variable #Switch_Off is ”1” OR if the signal status of variable #Failure is ”0”.

How does the comparator monitor the speed of the engine?
- The comparator compares the variables #Actual_Speed and #Preset_Speed. The result is written to variable #Preset_Speed_Reached (Signal status ”1”).

Depending on the language you have selected for programming your OB 1, go to:
Page 7.6 to 7.7, for LAD
Page 7.8 to 7.9, for STL
Page 7.10 to 7.11, for FBD.

Programming FBs in LAD



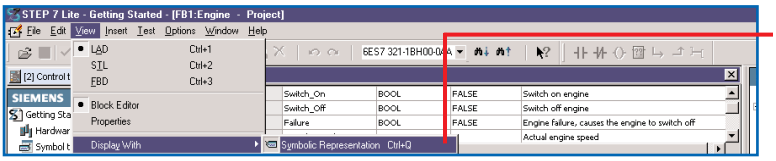


How to program an engine On/Off circuit

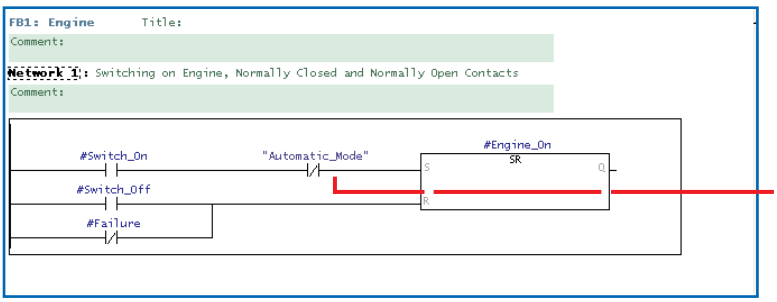
1. Highlight network 1. Open **Libraries > Commands**. Insert a series circuit consisting of a NO contact, NC contact and an SR flip-flop.

2. Next, highlight the circuit upstream in front of the R input and insert another NO contact.

3. Highlight the left circuit upstream of the NO contact. Insert a NC contact in parallel to the NO contact.



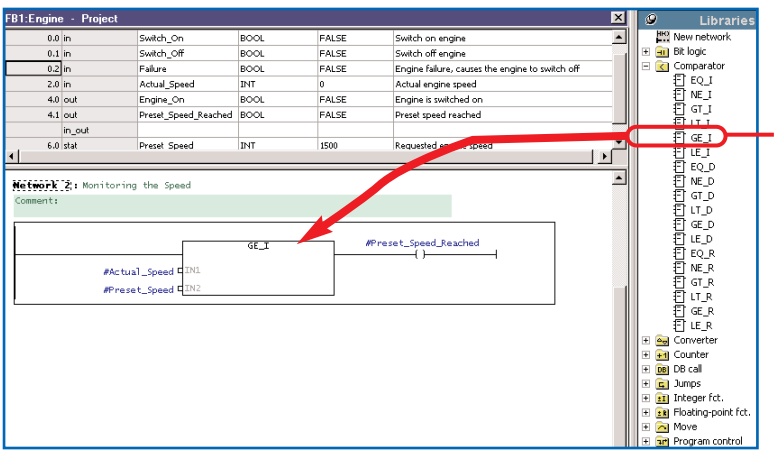
4 Check whether the symbolic representation is enabled.



5 Highlight all **??.?**. Enter the respective name in your variable declaration table (# is assigned automatically).

6 Assign the symbolic name ”Automatic_Mode” to the NC contact of the series circuit.

How to program a speed monitoring circuit



7 Insert a new network and highlight the circuit.

8 In the command library, select **Comparator** and insert **GE_I**. Also insert a coil in this circuit.

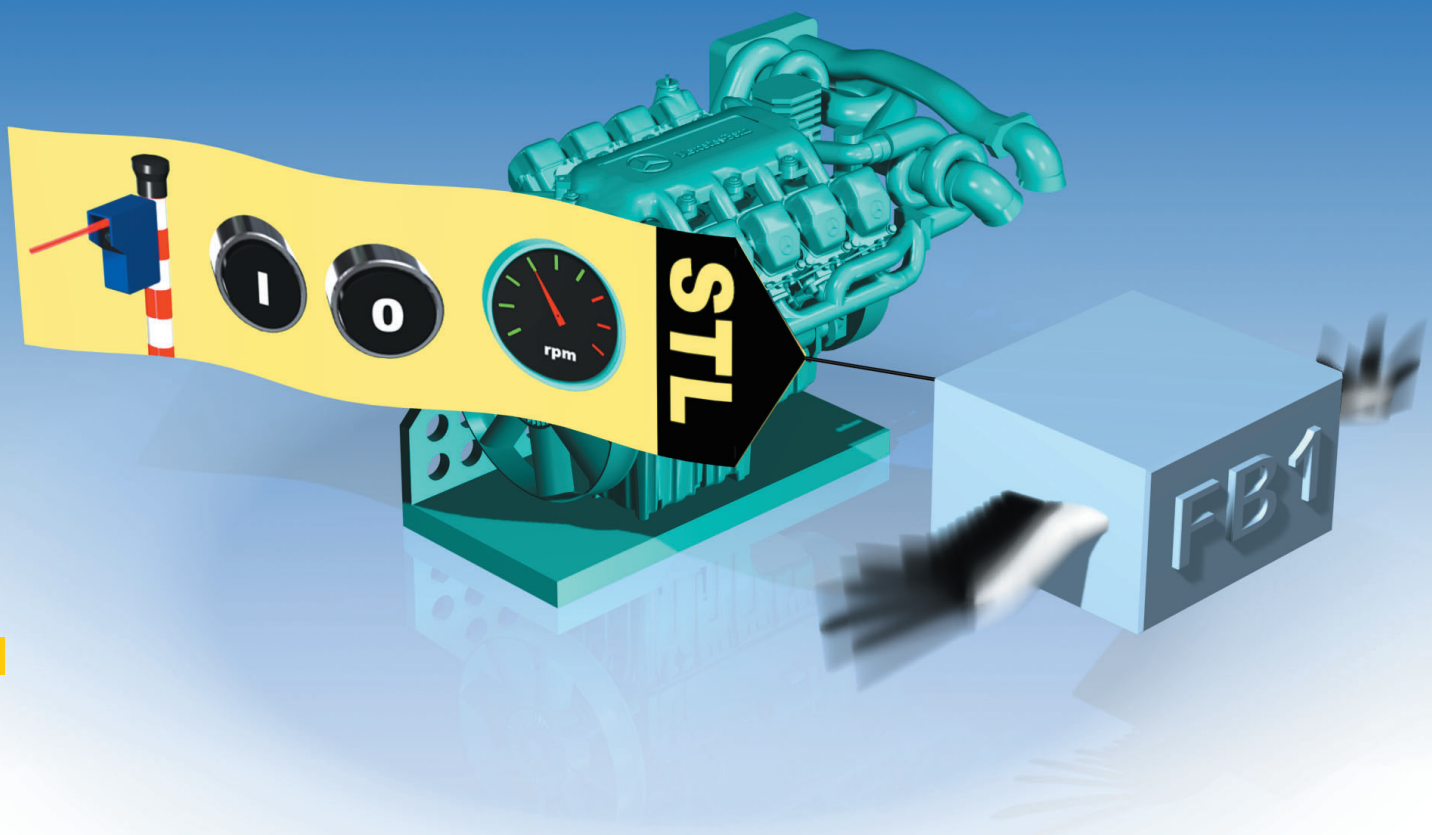
9 Again, highlight the question marks. Assign a name to the coil and to the comparator according to the variable declaration table.

Save your entries.

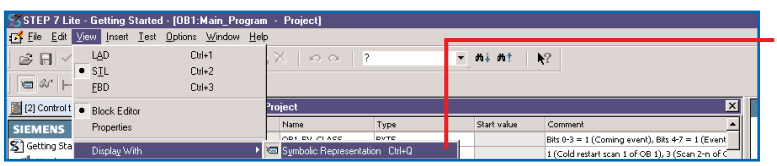
Local block variables are identified by # and they apply only in the block. Global variables are in quotation marks, they are defined in the symbol table and apply to the entire program. The signal status ”Automatic_Mode” is specified in OB1 (Network 3, compare Page 6-10) by another RS element and is now queried in FB1.

More information is found via **F1 > Content > Programming blocks > Creating logic blocks** and **Editing LAD elements**.

Programming FBs in STL



How to program an engine On/Off circuit



1 Check whether the symbolic representation is enabled.

```
FB1: Engine        Title:
First_Steps_STL

Network 1: Switching on Engine
Comment:

      A     #Switch_On
      AN    "Automatic_Mode"
      S     #Engine_On
      O     #Switch_Off
      ON    #Failure
      R     #Engine_On
```

2 Enter these STL instructions in network 1.

How to program speed monitoring

```
FB1: Engine       Title:
First_Steps_STL

Network 1 : Switching on Engine
Comment:

      A     #Switch_On
      AN    "Automatic_Mode"
      S     #Engine_On
      O     #Switch_Off
      ON    #Failure
      R     #Engine_On

Network 2 : Monitoring the Speed
Comment:

      L     #Actual_Speed
      L     #Preset_Speed
      >=I
      =     #Preset_Speed_Reached
```

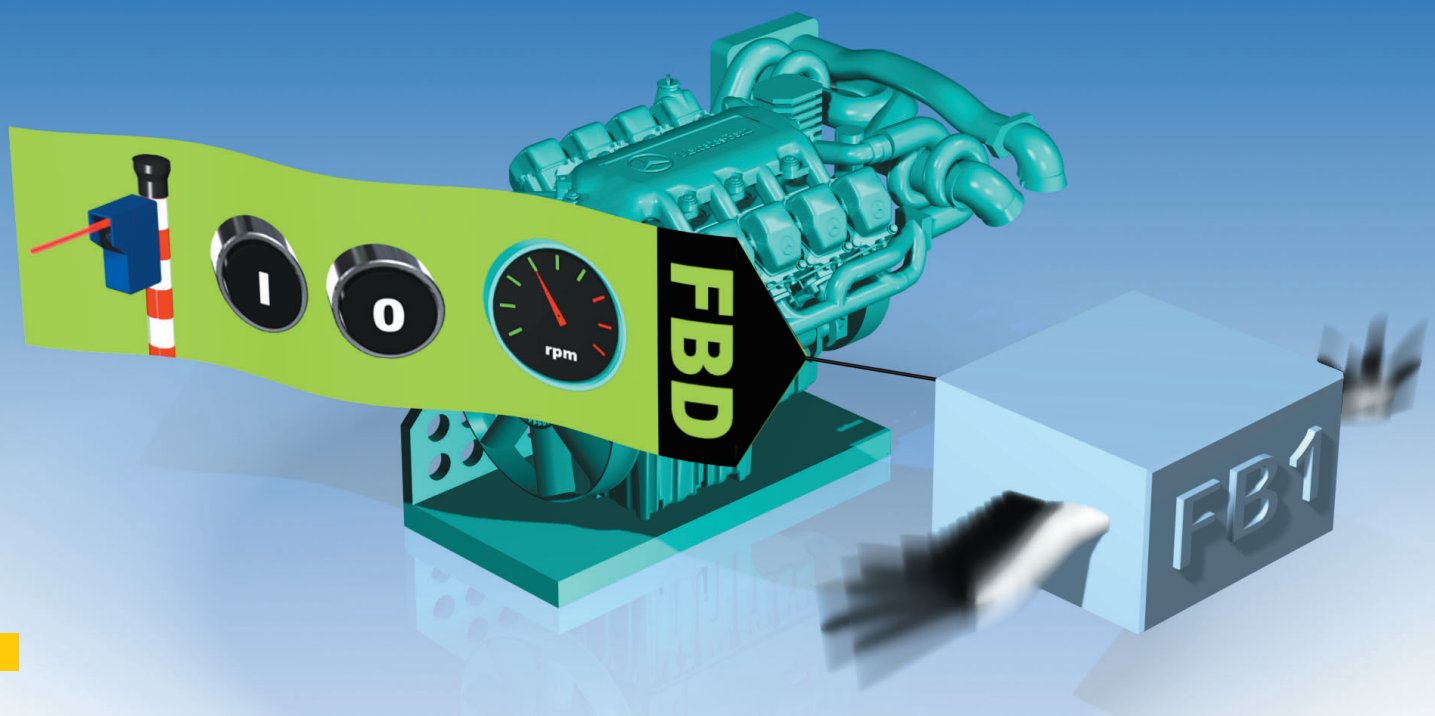
3 Insert a new network. Enter these STL instructions.

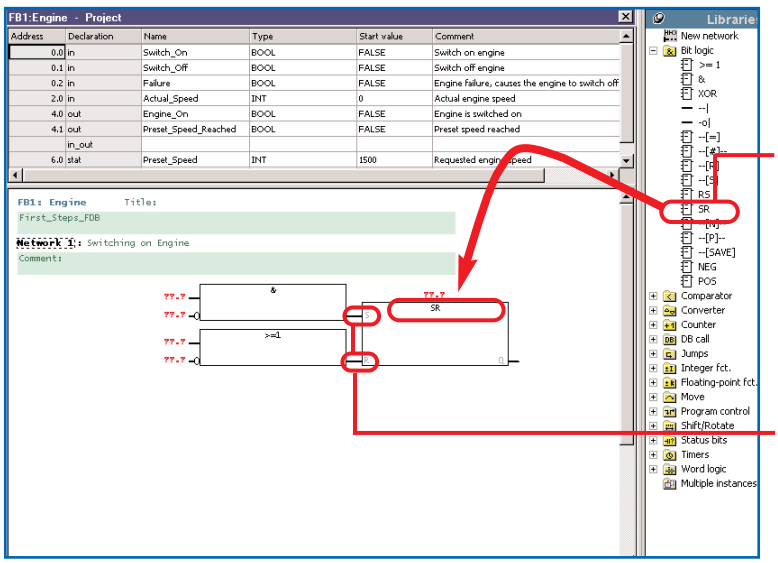
Save your entries.

Local block variables are identified by a # character and apply to this block only. Global variables are in quotation marks, and are defined in the symbol table and apply to the entire program. The signal status ”Automatic_Mode” is specified in OB1 (Network 3, compare Page 6-16) by another RS element and is now queried in FB1.

More information is found via **F1 > Content > Programming blocks > Creating logic blocks** and **Editing STL statements**.

Programming FBs in FBD

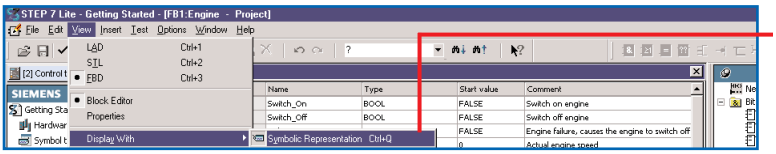




How to program an engine On/Off circuit

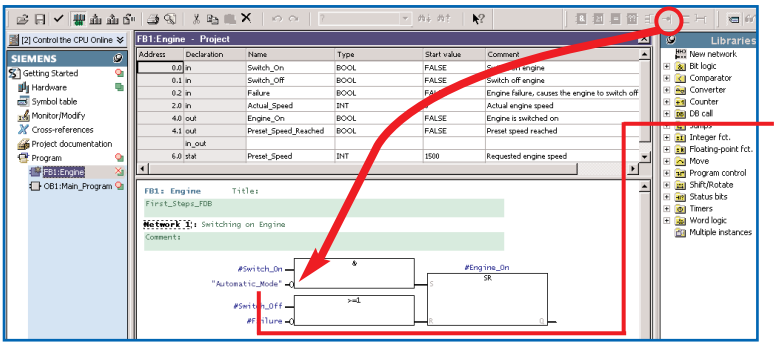
1 Insert an SR function in network 1 via command library.

2 Assign an AND box to input S (Set) and and an OR box to input R (Reset).



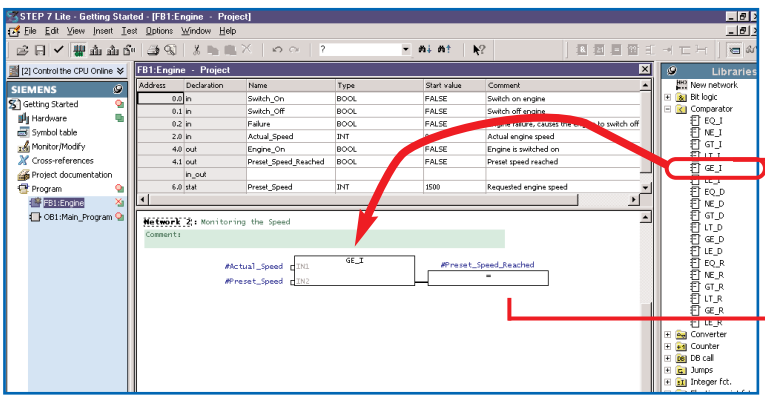
4 Check whether the symbolic representation is enabled.

5 Highlight all **??.?**. Enter the respective name in your variable declaration table (# is assigned automatically)..



6 Use the symbolic name "Automatic_Mode" to address an AND function input.
Invert the "Automatic_Mode" and #Failure inputs, using a corresponding symbol from the toolbar.

How to program a speed monitoring circuit



Insert a new network. Highlight the circuit.

7 In the command library, go to **Comparator** and enter **GE_I**. Address the inputs with a name from the variable declaration table.

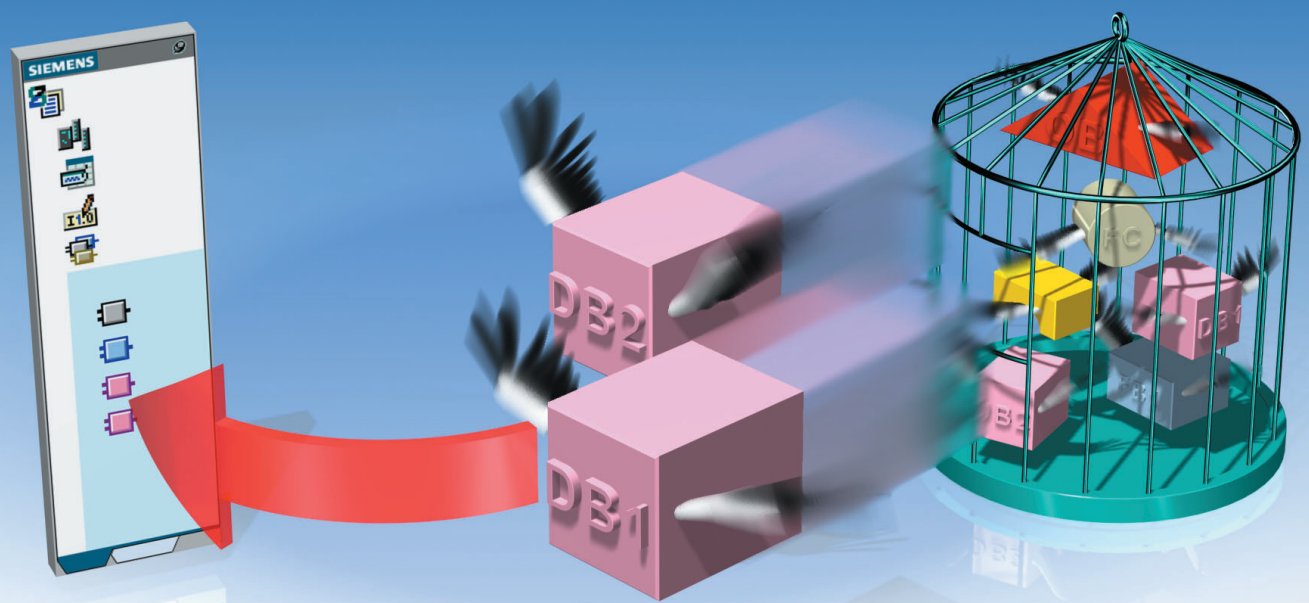
8 Assign an assign function to the comparator. Use a name from the variable declaration table to address this function.

Save your entries.

Local block variables identified by a # character apply to this block only. Global variables are in quotation marks, and are defined in the symbol table and apply to the entire program. The signal status "Automatic_Mode" is specified in OB1 (Network 3, compare Page 6-22) by another RS element and is now queried in FB1.

More information is found via **F1 > Content > Programming blocks > Creating logic blocks** and **Editing FBD elements**.

Generating instance DBs and modifying actual values

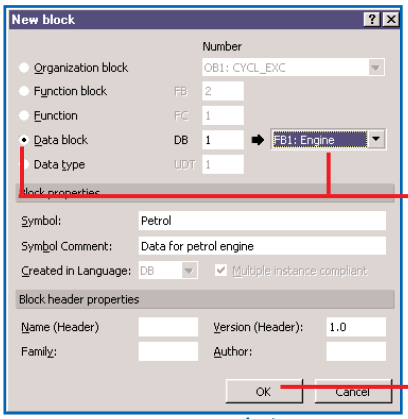


Inserting data blocks

To be able to program the call (CALL) of FB1 in OB1 later, you must generate a corresponding data block.

The FB is to control or monitor a petrol or diesel engine. The different speed setpoint values for the engines are stored in two separate DBs, that is, by modifying the respective actual value (#Preset_Speed).

You reduce your effort by programming a single, central FB.

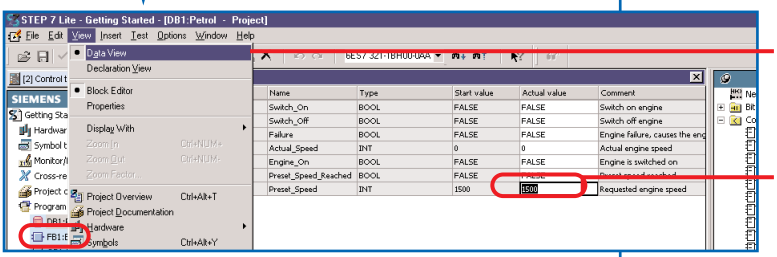


1 Right-click on the project window to open the pop-up menu. Select **New > Block**. The **New block** dialog pops up.

2 Highlight **Data block**. Enter FB1 as the assigned function block (as shown in the figure).

3 Confirm with **OK**. DB1 is inserted in your "Getting Started" project and opened immediately.

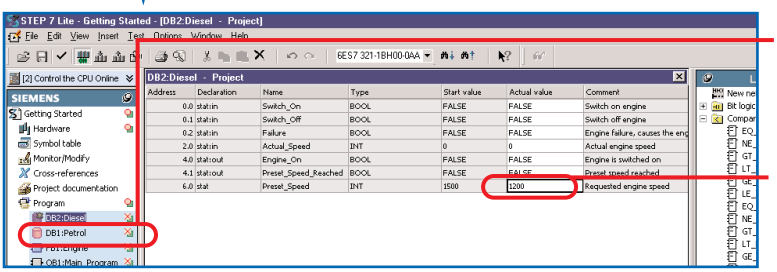
The block is inserted.



4 Enable **Data view**. You can edit the DB only in this view.

5 For the petrol engine, verify that you have entered the value "1500" in the **Start value** column.

The block is inserted.



6 Analogously, insert a DB2 "Diesel".

7 For the diesel engine, enter the **Start value** "1200" in the column. Save your entries.

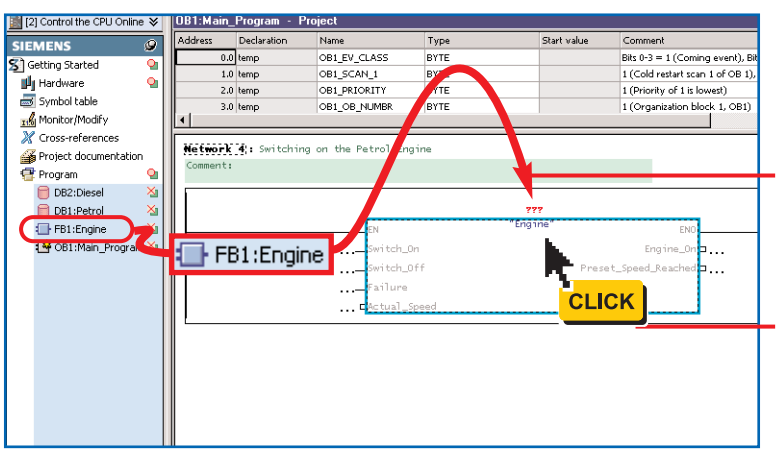
To program the FB call in OB1 in your selected programming language, continue at the respective chapter relating to LAD, FBD or STL.

By setting the actual values you have completed your preparations for controlling two engines with the help of a single function block. You only have to generate the respective data blocks to control further engines.

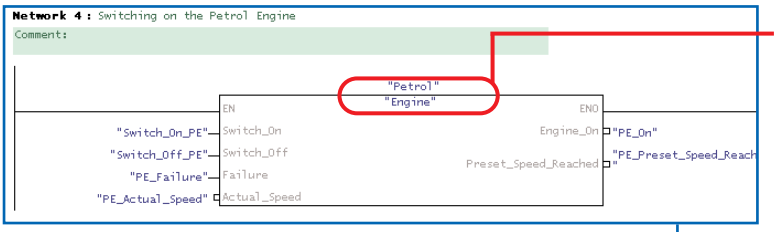
More information is found via **F1 > Content > Programming blocks > Creating data blocks**.

Programming block calls in LAD

The entire function block program will be ineffective unless it is called in OB1. To control both engines, one DB is used per FB call.



1. Open OB1 and insert network 4.
2. Drag **FB1** from the project window to network 4. All engine-specific variables will be displayed.
3. Click on **??.?** The symbol selection list pops up.

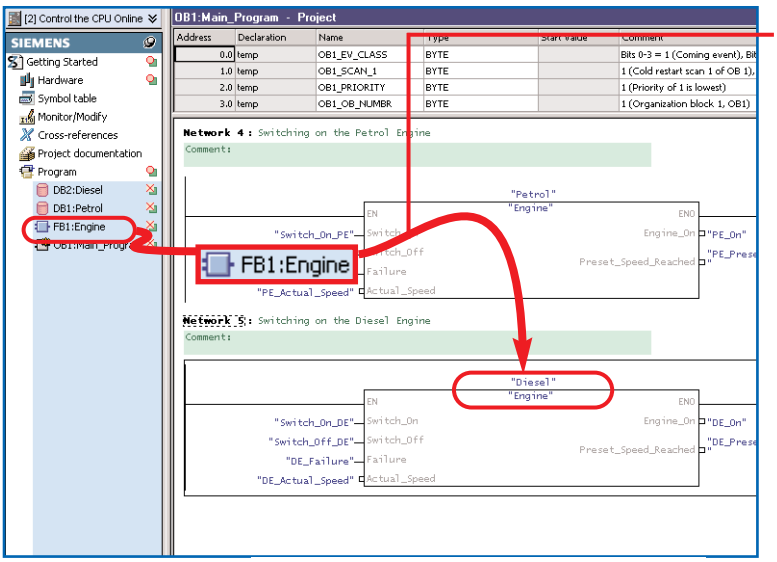


4 In this symbol selection list, select data block ”Petrol”.

5 Again, address all other FB parameters, using corresponding symbolic names.

Engine-specific I/O variables (Declaration ”in” and ”out”) are displayed in FB ”Engine”.
All variable are assigned a ”PE_xxx” signal for the petrol engine.

The call for the diesel engine is still missing



6 Insert network 5. Again, drag FB1 from the project window to the network. Analogously, program the call of FB ”Engine” (FB1) with DB ”Diesel” (DB2).

Assign all variables a ”DE_xxx” signal for the diesel engine.

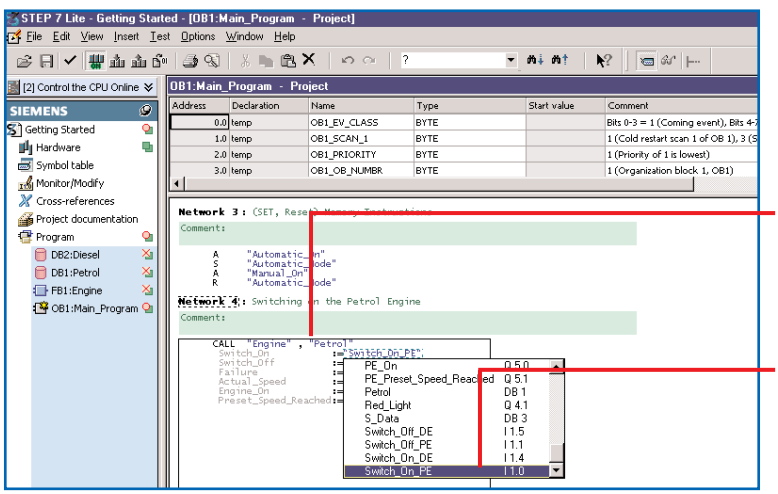
Save your entries and close the block.

When you create program structures which include OBs, FBs and DBs, you must declare the call of the lower level block (e.g. FB1) in the higher level block (e.g. OB1). This procedure is always the same. In the symbol table, you can also assign symbolic names to the different blocks (e.g. FB1 ”Engine” and DB1 ”Petrol”).

You can always print block programs via **File > Print**. Further information on printing is found via **F1 > Content > Printing project documentation**.

Programming block calls in STL

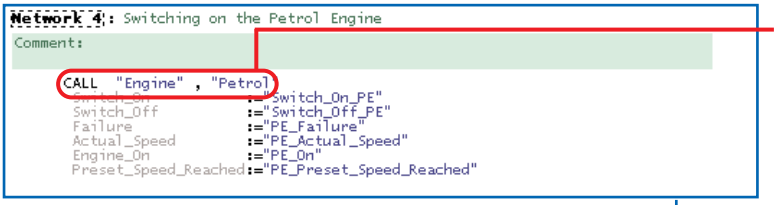
A function block program will be ineffective unless it is called in OB1. To control both engines, one DB is used per FB call.



1. Open OB1. Insert network 4.

2. Declare **"Engine", "Petrol"** in the **CALL** instruction. Confirm the entry with Return. All engine-specific variable are displayed.

3. Right click next to ":=" to open the pop-up menu. Select the insert symbol command. The symbol selection list pops up.

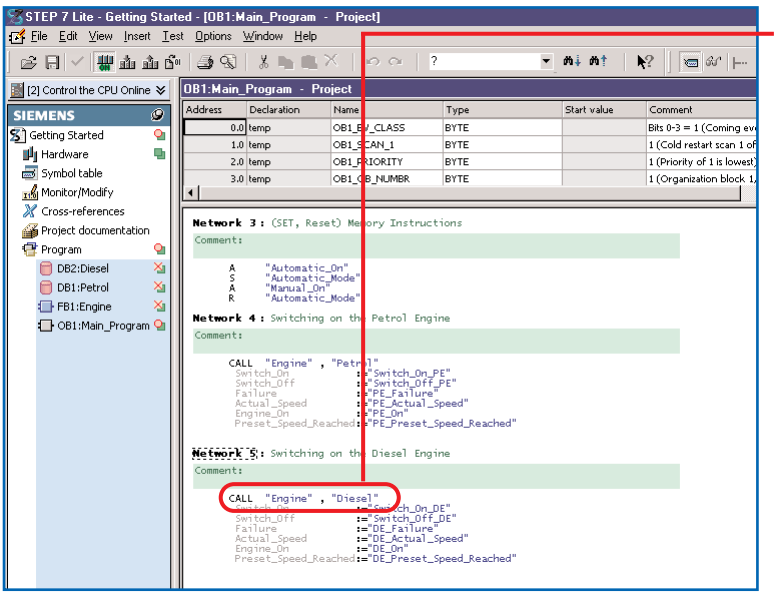


4

Again, address all other FB parameters, using corresponding symbolic names.

Engine-specific variables (Declaration "in" and "out") are displayed in FB "Engine".
All variables are assigned a "PE_xxx" signal for the petrol engine.

The call for the diesel engine is still missing.



6

Analogously, insert network 5 and program FB "Engine" (FB1) to call DB "Diesel" (DB2).

All variables are assigned a "DE_xxx" signal for the diesel engine.

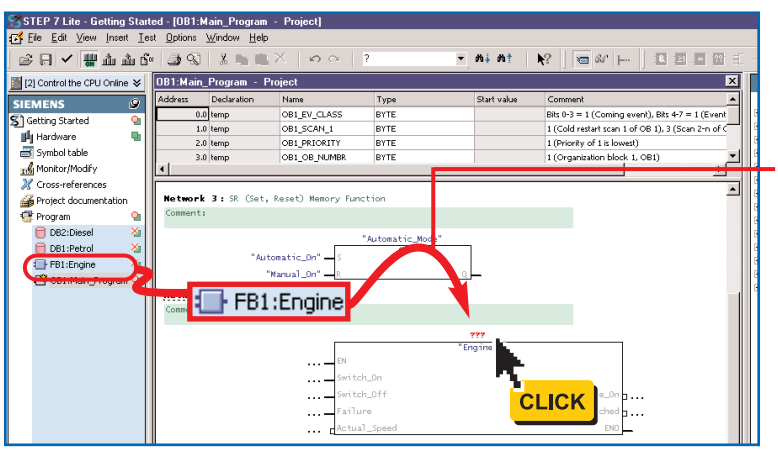
Save your entries and close the block.

When you create program structures which include OBs, FBs and DBs, you must declare the call of the lower level block (e.g. FB1) in the higher level block (e.g. OB1). This procedure is always the same. In the symbol table, you can also assign symbolic names to the different blocks (e.g. FB1 "Engine" and DB1 "Petrol").

You can always print block programs via **File > Print**. Further information on printing is found via **F1 > Content > Printing project documentation**.

Programming block calls in FBD

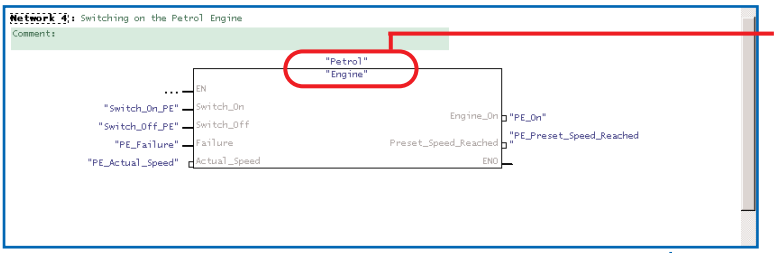
A function block program will be ineffective unless it is called in OB1. To control both engines, one DB is used per FB call.



1. Open OB1. Insert network 4.

2. Drag&drop **FB1** from the project window to network 4. All engine-specific variables will be displayed.

3. Click on **??.?** to open the symbol selection list.

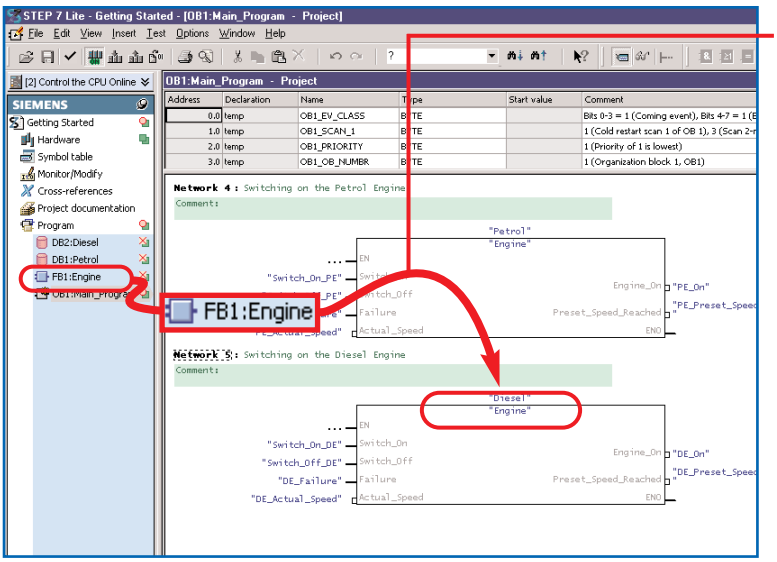


The call for the diesel engine is still missing.

4 Select DB ”Petrol” from the symbol selection list.

5 Again, address all other FB parameters, using corresponding symbolic names.

Engine-specific variables (Declaration ”in” and ”out”) are displayed in FB ”Engine”.
All variables are assigned a ”PE_xxx” signal for the petrol engine.



6 Insert network 5. Again, drag&drop FB ”Engine” (FB1) from the project window and program it to call DB ”Diesel” (DB2).

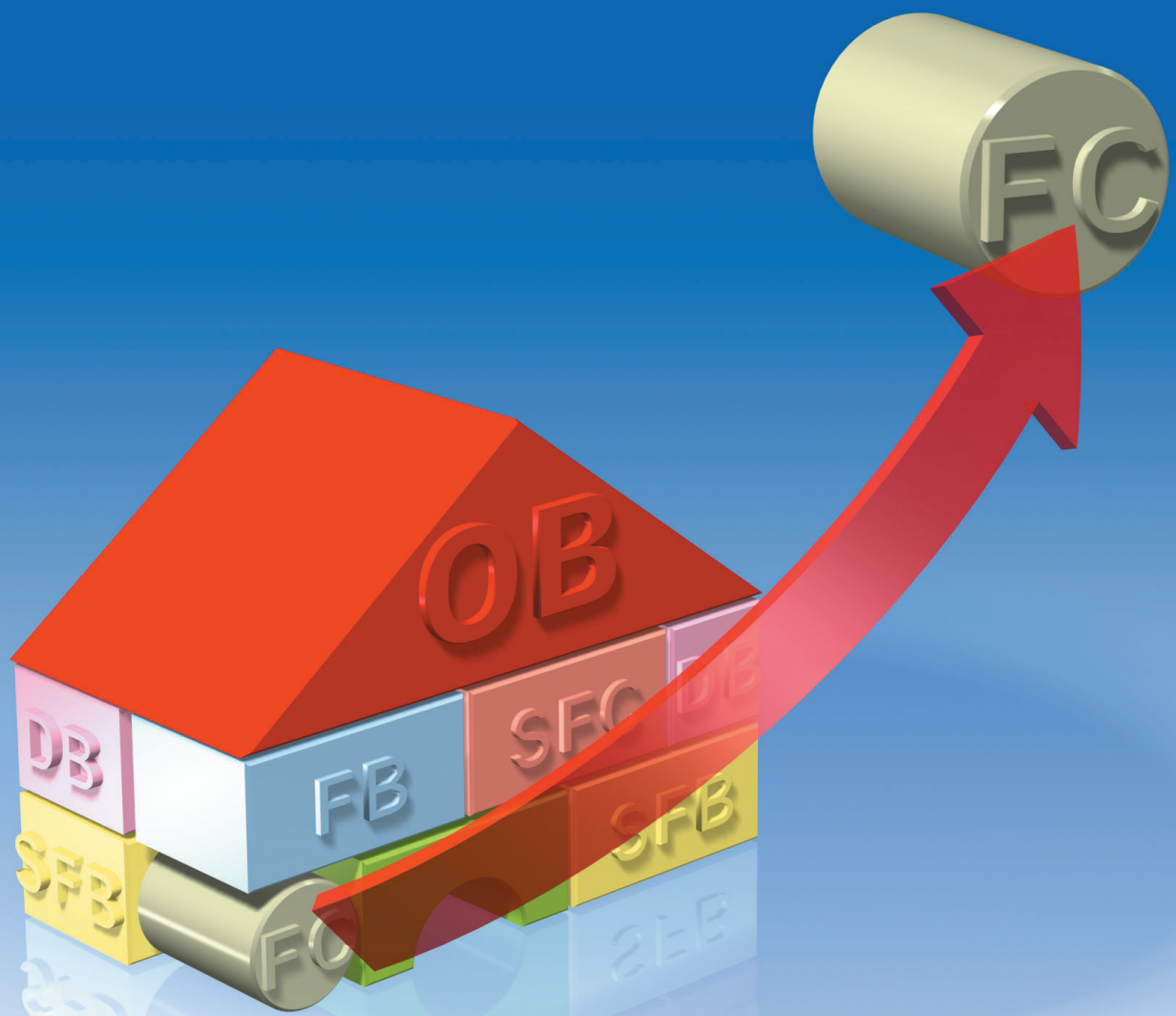
All variables are assigned a ”DE_xxx” signal for the Diesel engine.

Save your entries and close the block.

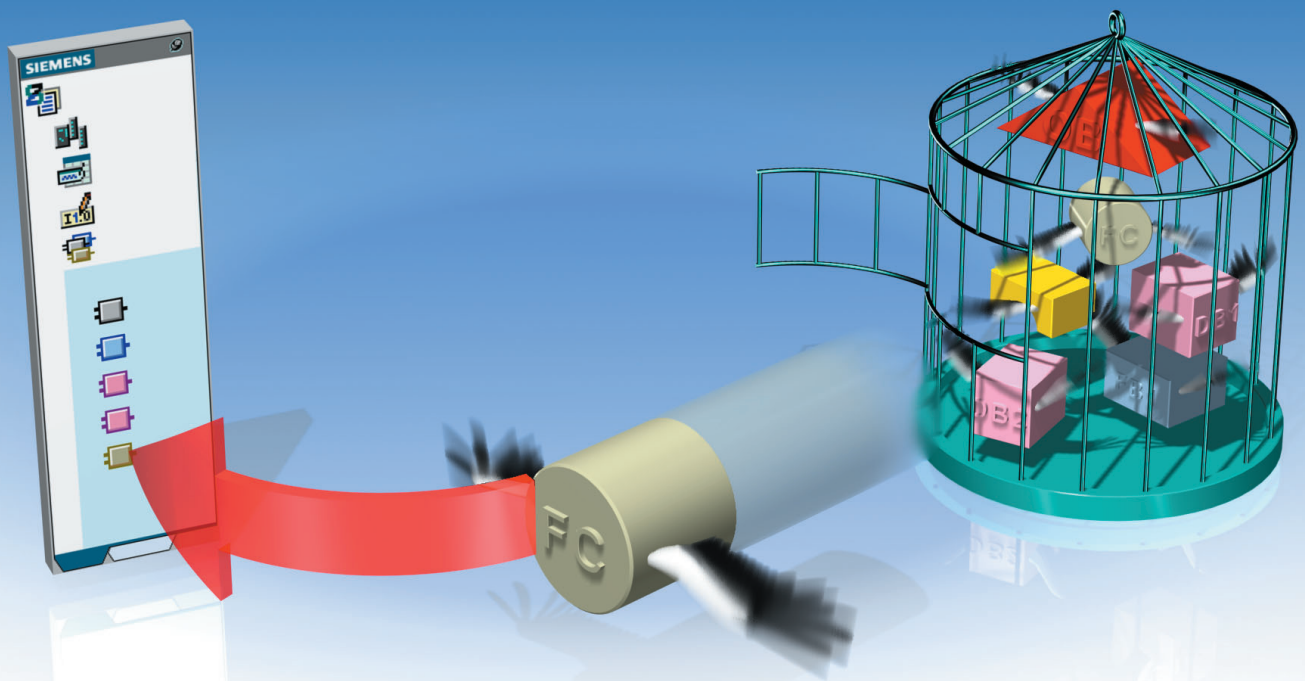
When you create program structures which include OBs, FBs and DBs, you must declare calls of a lower level block (e.g. FB1) in the higher level block (e.g. OB1). This procedure is always the same. In the symbol table, you can also assign symbolic names to the different blocks (e.g. FB1 ”Engine” and DB1 ”Petrol”).

You can always print block programs via **File > Print**. Further information on printing is found via **F1 > Content > Printing project documentation**.

8 Using functions



Creating and opening functions (FCs)

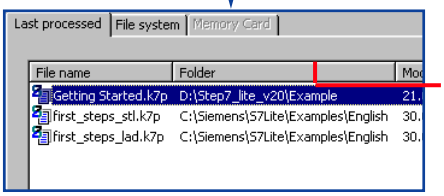


Functions are used if FC programming does not require of you to save intermediate results, mode settings or operating modes until the next block call. This is why they are also referred to as ”Blocks without memory”.

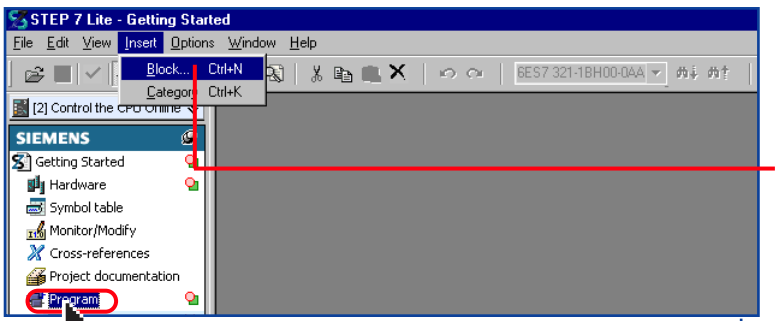
Your ”Getting Started” project should contain a copy of the symbol table before you continue with this chapter (see Page 5.5).

The dialog box for project selection is opened.

1 If required, open STEP 7 Lite.



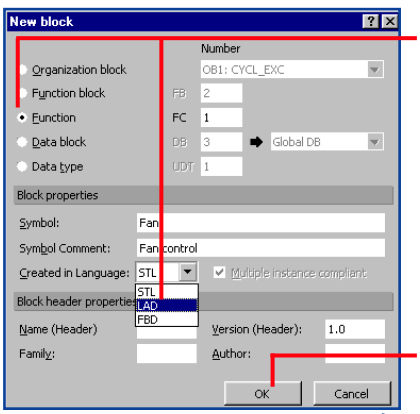
2 Click on "Getting started" in the **Open project** dialog to open your project.



3 Click on Program in the project window.

4 Select menu command **Insert > Block**, or right-click to open the pop-up menu and select **New > Block**.

The dialog box for creating new blocks is opened.

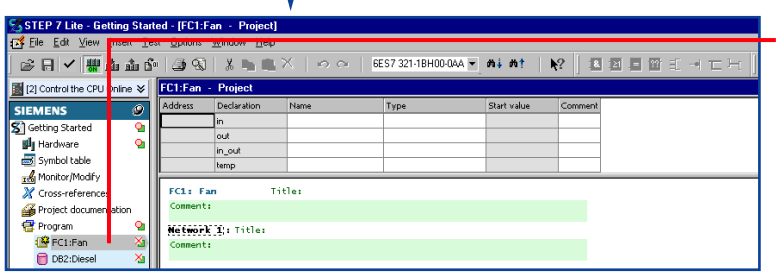


5 Highlight **Function.**

In the **Created in Language** box, select the language you have used to generate your "Getting Started" project.

6 Confirm with **OK**.

The block is inserted and opened immediately.



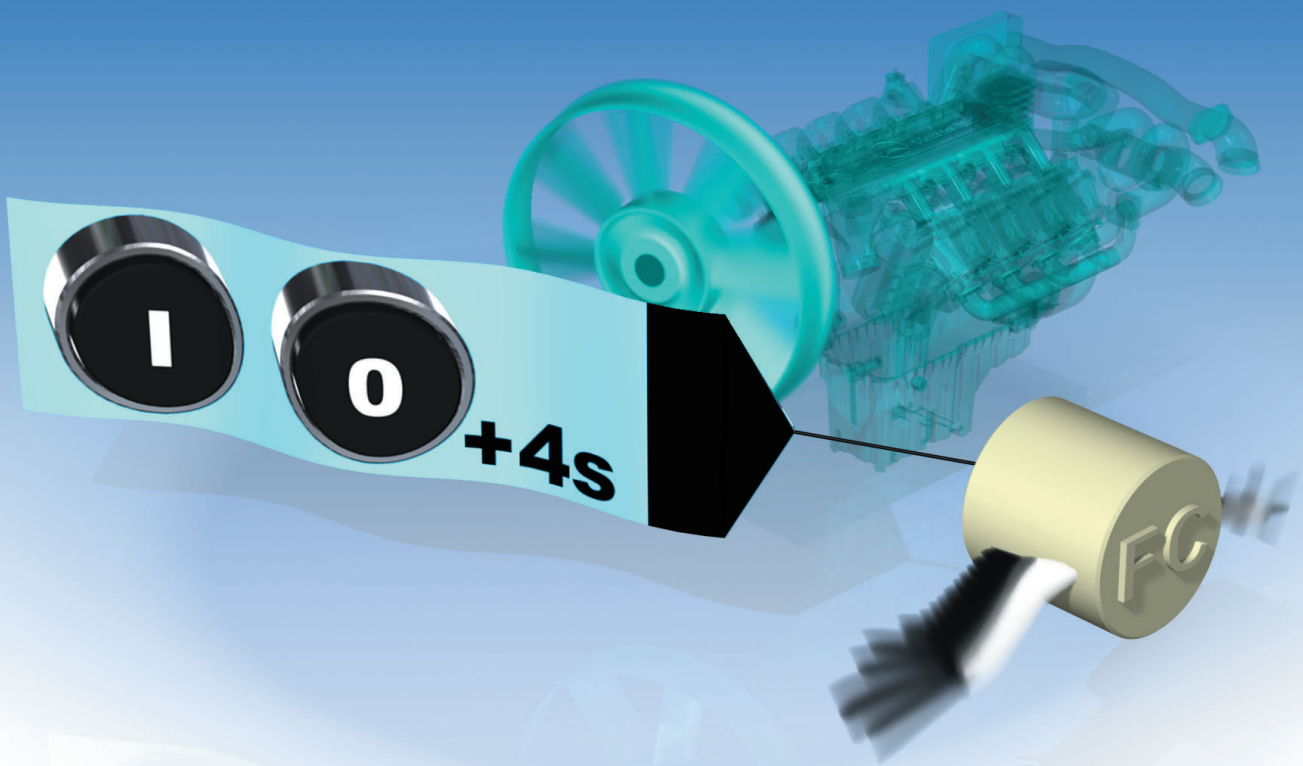
7 The new block is inserted into the project window and opened immediately.

Contrary to FBs, within a function you cannot declare static data in the variable declaration table.

You can refer to symbolic names from the symbol table to program the function in the way as you are used to.

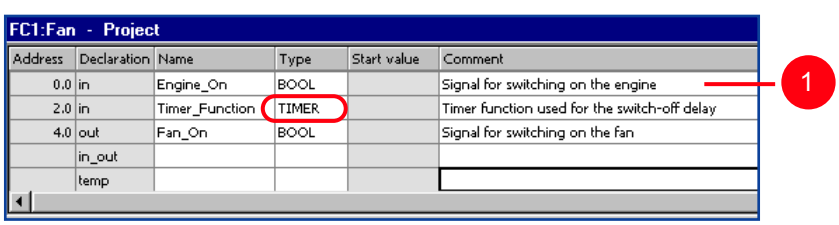
Further information is found via **F1 > Content > Basics of designing a program > Blocks in the user program**.

Programming functions



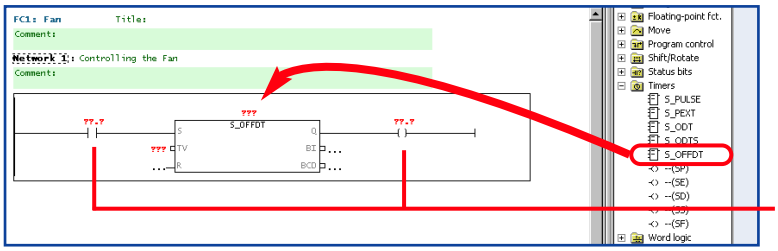
In our next example you are going to program a timer function.
When the timer is switched, the timer function simutaneously switches on a fan which runs for 4 seconds after the engine has been switched off (off delay)

Variable declaration table



Analogous to the FB, start by declaring the function's I/O parameters (Declaration "in" and "out") in the variable declaration table.
Right-click to open the pop-up menu **Parameter types**. Select TIMER.

How to program timer functions in LAD

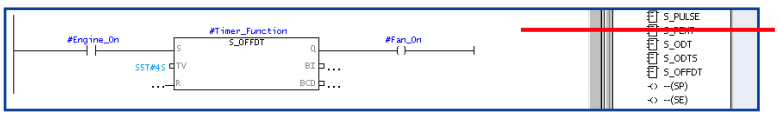


2

Go to **Libraries > Commands > Timers**. Select element **S_OFFDT** and insert it in network 1.

3

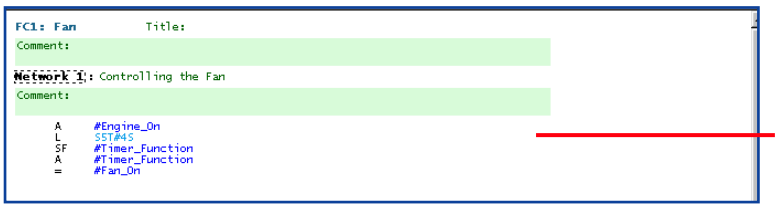
Insert an additional NO contact and a coil.



4

For **??.?**, enter the names from the variable declaration table. These are automatically marked with a # character. Enter the timer constant TV S5T#4s. Save your entries. Close the block.

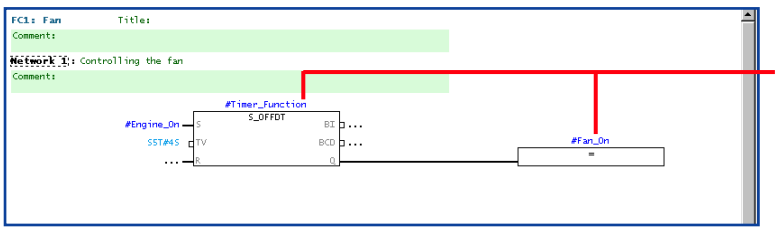
How to program timer functions in STL



2

Declare the instructions as shown at the side. Save your entries. Close the block.

How to program timer functions in FBD



2

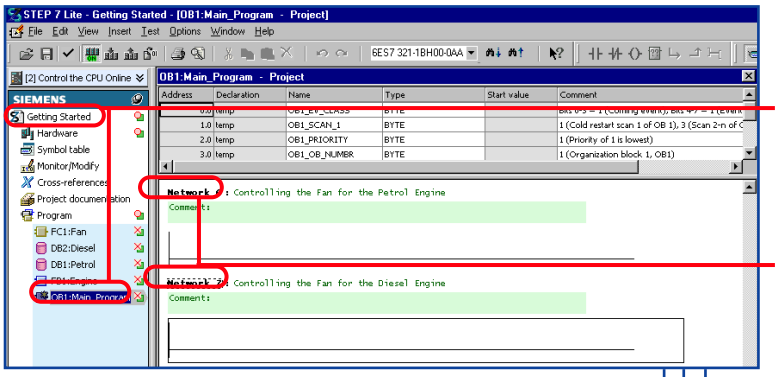
Same as in LAD: Copy the instructions from the library to your network, fill all **??.?** and declare the timer constant. Save your entries. Close the block.

Input parameter ”#Engine_on” starts ”#Timer_Function”. On subsequent calls in OB1, the petrol or Diesel engine parameters are assigned to the function accordingly (e.g. T1 for “PE_Follow_On“).

Calling functions in OB1

FC1 is called in OB1 in the same way as a function block. For function parameters OB1 provides corresponding address data for the petrol or Diesel engine.

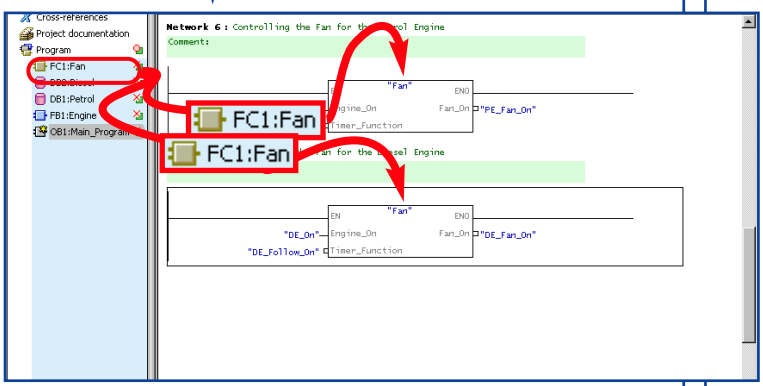
An address is part of a STEP 7 Lite instruction which specifies a function the CPU uses for processing. Addressing can be absolute or symbolic.



Opening OB1

1. Open the ”Getting Started” project which you have generated in LAD, FBD or STL. Open OB1.
2. Insert **Network 6** for the petrol engine and **Network 7** for the Diesel engine.

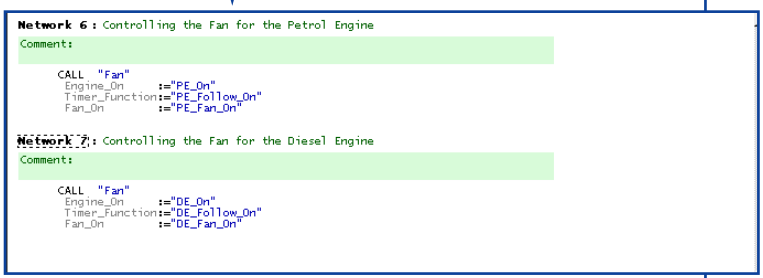
This is how it looks after programming in LAD ...



How to program block calls in LAD

3. Drag&drop FC1 to Network 6 and Network 7.
4. Edit all **??.?** as shown.
   Save and close the block.

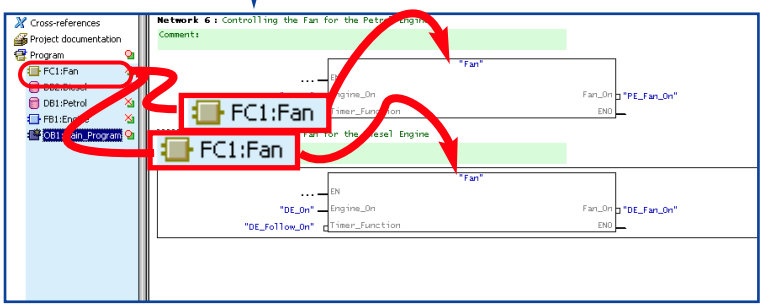
... how it looks in STL,



How to program block calls in STL

3. Enter the STL instructions as shown.
4. Save and close the block.

... how it looks in FBD.



How to program block calls in FBD

3. Drag FC1 to Network 6, petrol engine.
   Drag FC1 to Network 7, Diesel engine.
4. Edit all **??.?** as shown.
   Save and close the block.

1. Your screen differs from our screen shots?

Set symbolic programming via **View > Display with > Symbolic Representation**.

2. You want to see more on-screen information?

Enable **View > Display with > Symbol Information** to obtain information on specific network addresses.

To display several networks on-screen, disable **View > Display with > Comment** and, if required, **View > Display with > Symbol Information**.
Under **View > Zoom factor**, you can adjust the on-screen size of the networks.

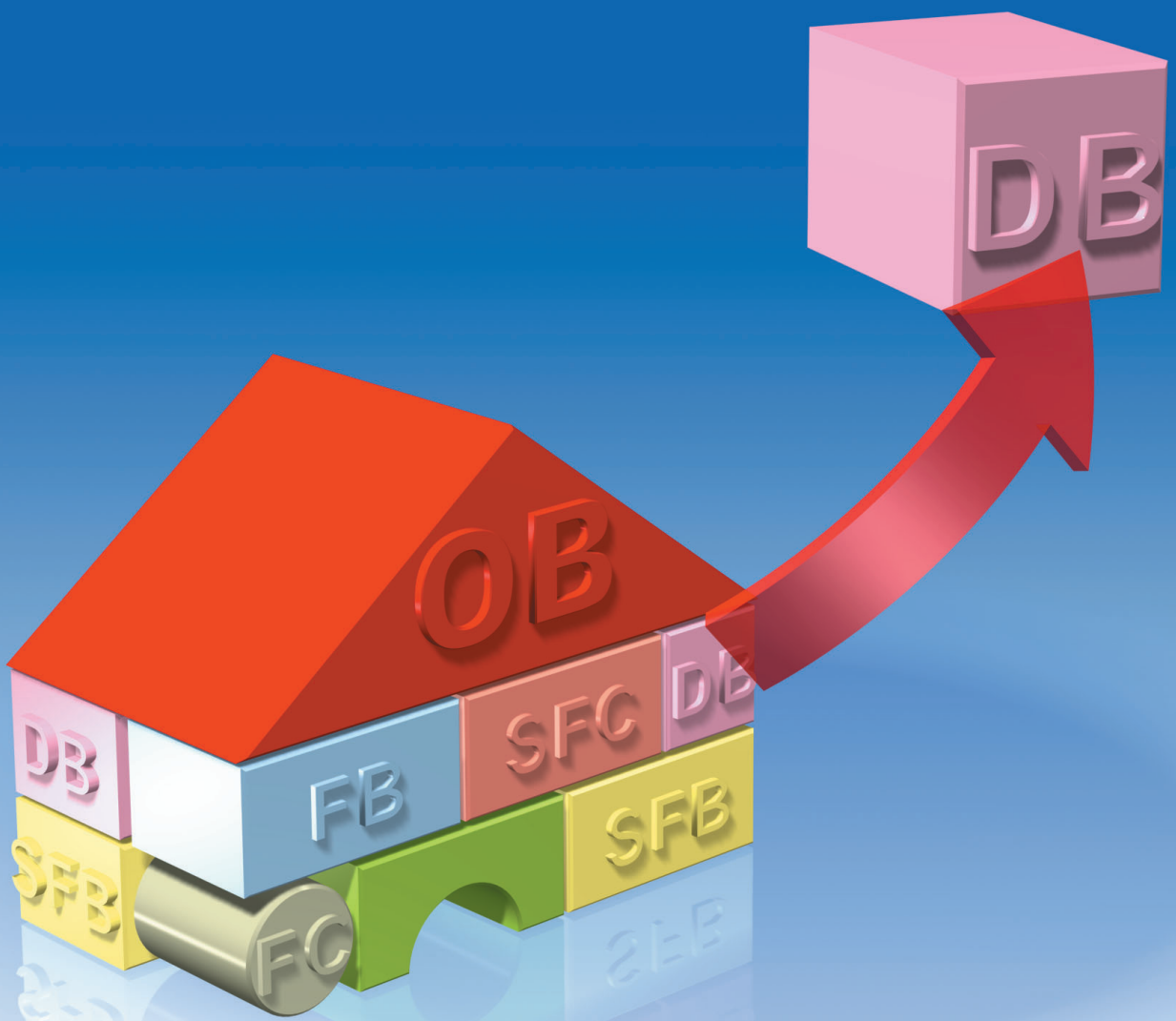
3. You need information on the programming languages LAD, STLL or FBD?

Further information is found via **F1 > Content > Calling Reference Helps > Language descriptions and block help.**

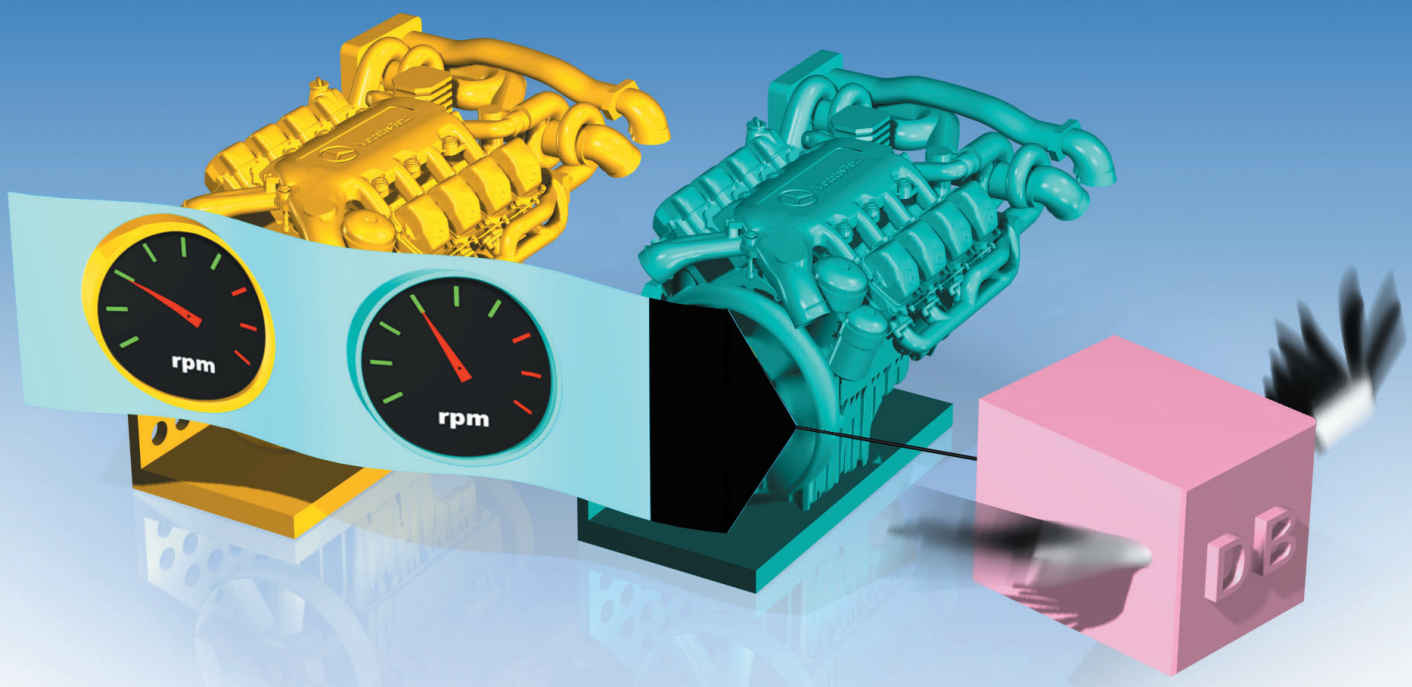
4. You do not always want to call the function?

In our example we have programmed an absolute function call, that is, the function is always going to be processed. Depending on requirements for your automation task, you can also program conditional FC or FB calls: e.g. via enable signal from an input or series circuit. The EN input or ENO output of the box are both available for programming conditional calls.

9 Using global data blocks



Generating and opening global data blocks (DBs)



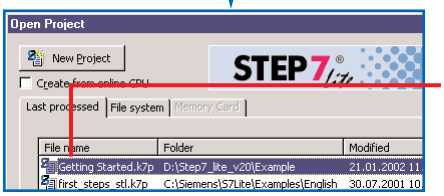
You can save selected data to a shared data block if the CPU cannot store anymore because it has run out of internal memory bits (memory cells).

Data of a shared DB are available to any other block. An instance DB, on the other hand, is assigned to a specific function block. Its data are only available locally in this FB (compare Chapter 7, How to generate instance DBs and modify actual values).

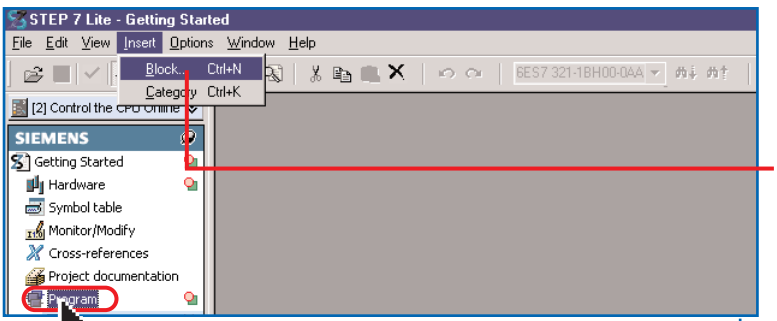
Your ”Getting Started” project should contain a copy of the symbol table before you continue with this chapter (see Page 5.5).

The dialog box for project selection is opened.

1 If required, open STEP 7 Lite.



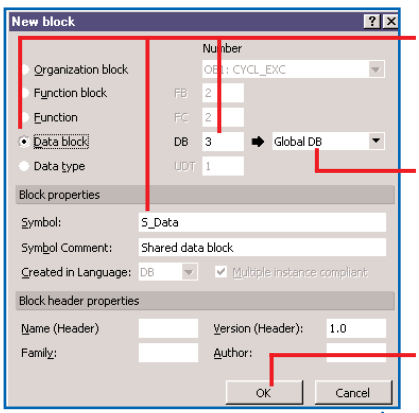
2 Double-click on ”Getting Started” in the **Open project** dialog to open your project.



3 In the project window, click on Program.

4 Select menu command **Insert > Block**, or right-click to open the pop-up menu and select the menu command **New > Block**.

The dialog box for creating new blocks is opened.

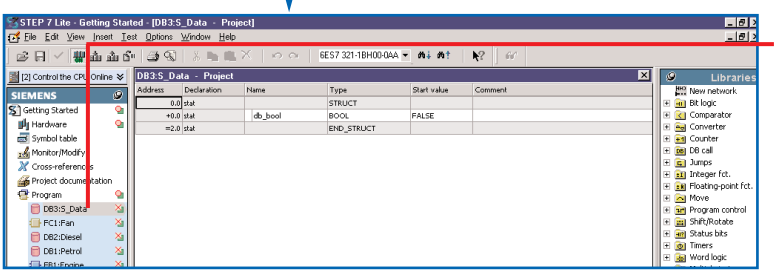


5 Highlight **Data block.**
A ”3” is written automatically to the **DB** field, and ”S_Data” to the **Symbol** field.

6 Here, select ”Global DB”.

7 Confirm with **OK**.

The block is inserted and opened immediately.



8 The new block is inserted into the project window and opened immediately.

Programming DB variables

DB3:S_Data - Project

Address	Declaration	Name	Type	Start value	Comment
0.0	stat		STRUCT		
+0.0	stat	PE_Actual_Speed	INT	0	Actual speed for petrol engine
+2.0	stat	DE_Actual_Speed	INT	0	Actual speed for diesel engine
+4.0	stat	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
=6.0	stat		END_STRUCT		

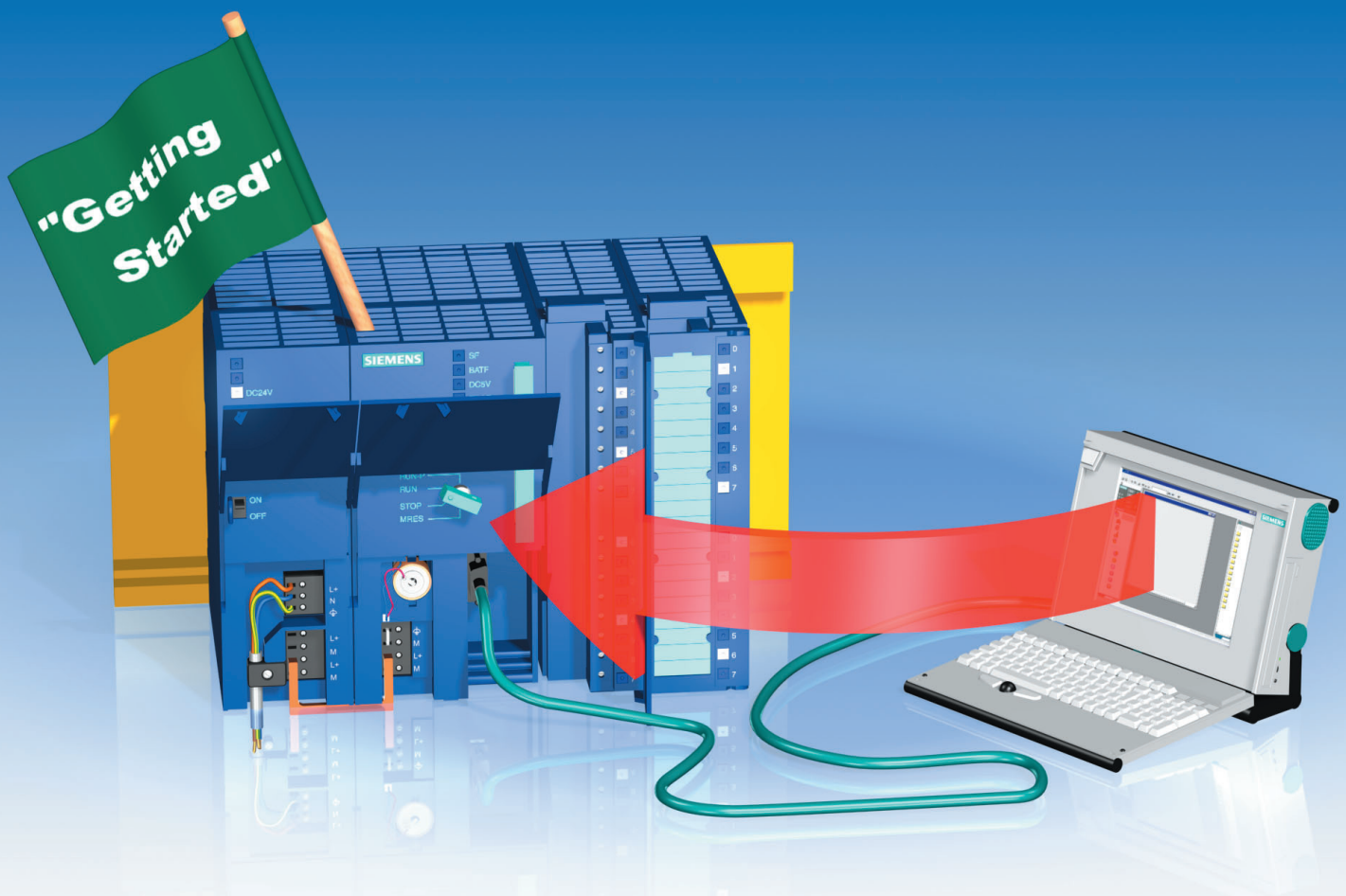
1 In the **Name** column, enter "PE_Actual_speed".

2 Right-click to open the pop-up menu. Select the **Type** under **Elementary types > INT**.
Complete the list as shown above.

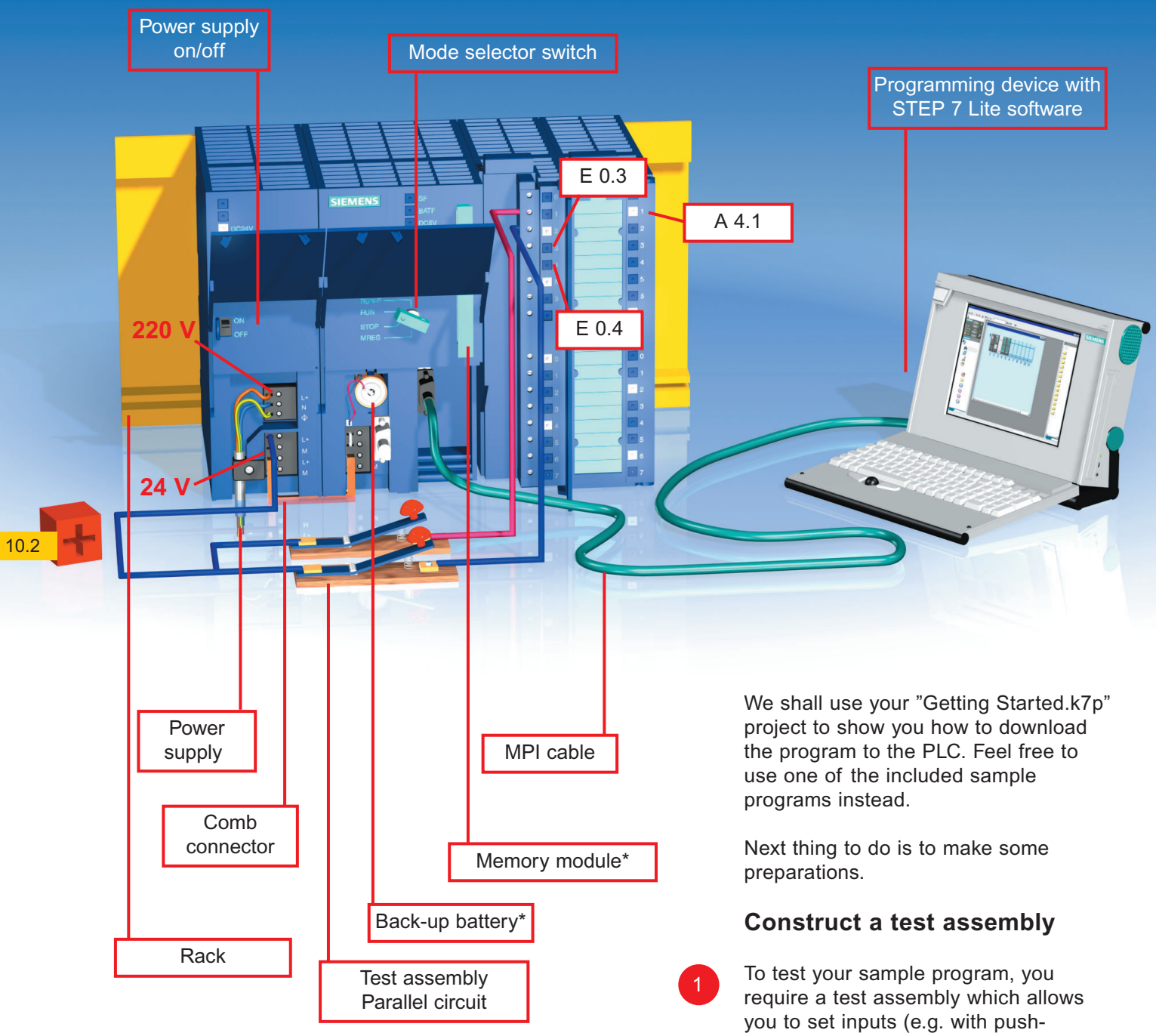
Save your entries. Close the block.

Further information is found via **F1 > Content > Programming blocks > Creating data blocks**.

10 Downloading programs to the CPU



Establishing an Online connection



We shall use your ”Getting Started.k7p” project to show you how to download the program to the PLC. Feel free to use one of the included sample programs instead.

Next thing to do is to make some preparations.

Construct a test assembly

1 To test your sample program, you require a test assembly which allows you to set inputs (e.g. with push-buttons).

*= Not absolutely necessary

2 Perform a program check

If you decide to use your ”Getting Started” project, you should at least have configured your hardware (Chapter 4) and programmed the parallel circuit (Chapter 6).

3 Perform a hardware check

Assemble your hardware and check once again:

- Are the bus connectors plugged into the modules?
- Are the modules attached to the profile rail, swung down and screw-tightened?
- Is the 220 V power supply connected?
- Is the comb connector inserted?
- If exist, have you inserted the backup battery and memory module?

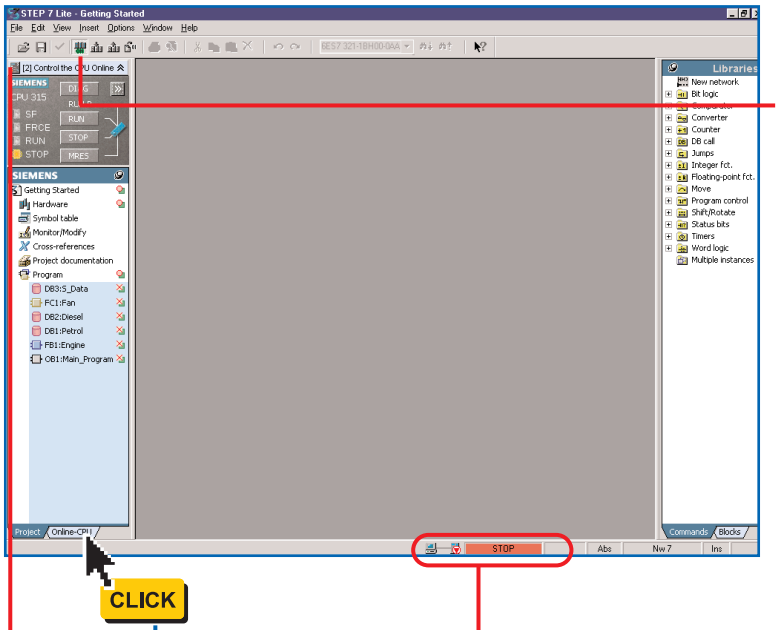
4 Establish the Online connection

To establish an Online connection means to interconnect the CPU and PG.

- Interconnect the CPU and the PG via MPI cable.

At the CPU:
- Switch **on** the power supply
- Set the mode selector switch to **STOP**.

At the PG:
– Switch **on** the master switch.
– Run STEP 7 Lite.
– Open ”Getting Started.k7p” or one of the sample projects.



Going Online

5 When started, STEP 7 Lite immediately attempts to go Online.

The **Online** button must be lit in green color. The status bar indicates the **Online** connection status, followed by CPU mode **STOP**.
If STEP7 Lite fails to connect to a CPU, it remains Offline and the status bar indicates the connection status **Offline**.
In this case, eliminate the cause of the Online connection failure (e.g. a cable is not properly plugged in or the CPU is switched off) and then click on the **Online** button.
You can always click on this button to go online or offline.

6 Verify that **Control the CPU Online** is indicated. If yes, the buttons on the CPU operator panel are enabled.

7 Check: The status bar now displays the actual CPU operating mode.

8 Try out: You can now toggle Offline view **Project** and Online view **Online CPU**.

The **Online CPU** tab shows all blocks of the CPU.

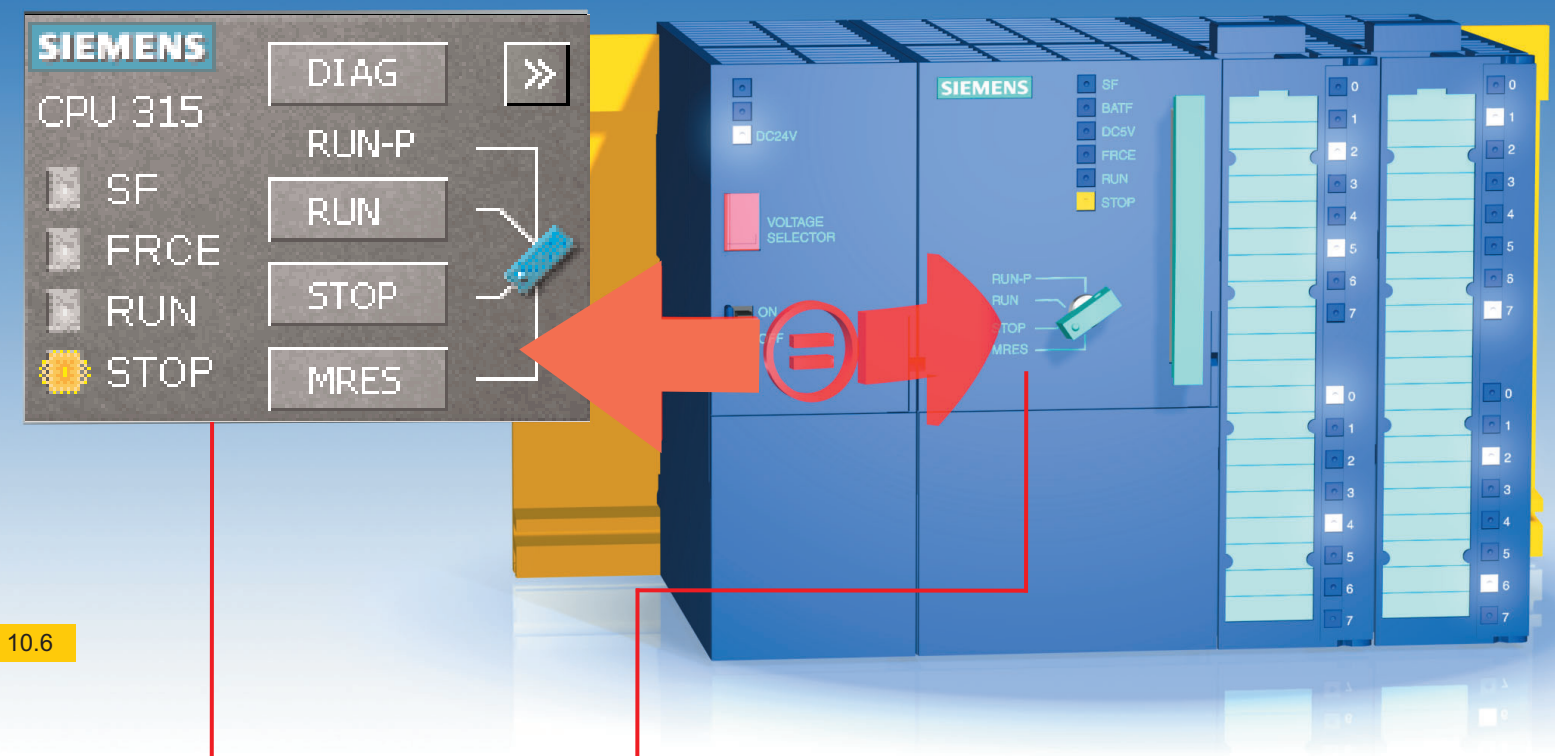
Icons on your project window will indicate data inconsistency between PG and CPU if you have not yet downloaded your program to the CPU.

You can also establish Online connections and perform tests (Chapter 11) if you use other than the hardware displayed on Page 10.2. You merely have to comply with I/O addressing conventions.

Further information on the assembly of PLC modules is found in the
”S7-300 – Hardware and Installation” manual.

Further information on establishing online connections is found in the Online Help via **F1 > Content > Establishing an Online Connection**.

Resetting CPU memory and downloading the program



CPU operator panel in STEP 7 Lite

CPU operator panel on the CPU

The CPU operator panel

Reset CPU memory prior to program download. You can use the CPU operator panel in STEP 7 Lite or directly at the CPU.

CPU operating modes are also set on the operator panel.

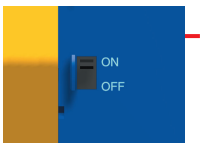
Due to safety reasons, however, the STEP 7 Lite operator panel only lets you select the operating mode set at the CPU.

Example:
– At the CPU: Switch is set to **RUN** - At the STEP 7 Lite operator panel: **STOP** instructions can be enabled.
– At the CPU: Switch is set to **STOP** - At the STEP7 Lite operator panel: **RUN** instructions cannot be enabled.

In danger situations you can also set the CPU to STOP mode via STEP 7 Lite.

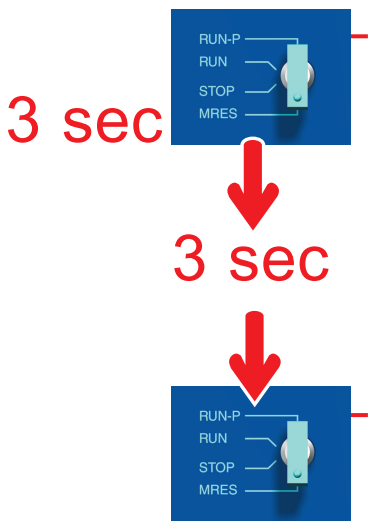
CPU memory reset at the CPU

Before you download your program to the CPU, delete all old data and programs on the CPU via reset memory instruction. To do this:



1 CPU **on**.

2 Mode selector switch to **STOP** position (if not already in STOP position). The LED is lit in red color.



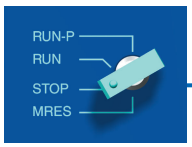
3 Turn the mode selector switch to **MRES** position. Hold it there for at least three seconds until the red STOP LED flashes at a slow rate.

4 Release the mode selector switch. Within three seconds, set the switch once again to MRES position and hold it there until the STOP LED flashes at a fast rate.

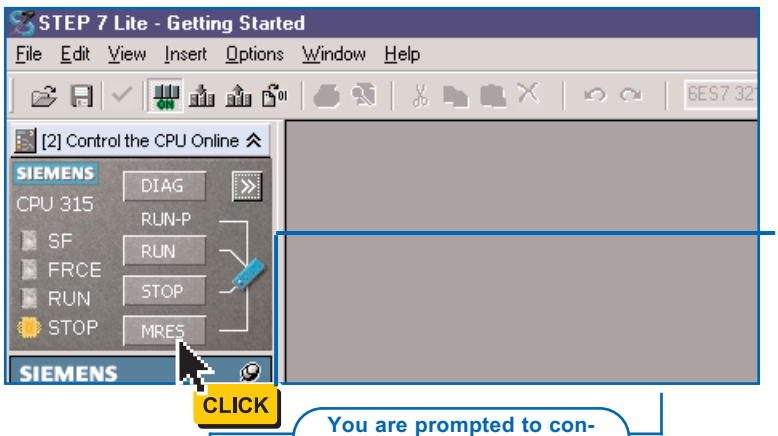
You have now completed CPU memory reset.

CPU memory reset in STEP7 Lite

You can also choose to reset CPU memory via STEP 7 Lite.

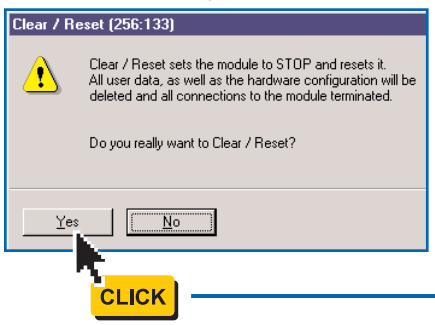


1 At the CPU: Set the mode selector **STOP** position.



2 In STEP7 Lite: Click on the **MRES** button.

You are prompted to confirm your action.

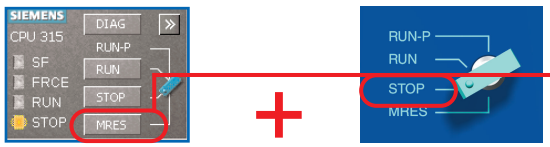


3 Confirm with **YES**. CPU memory is reset.

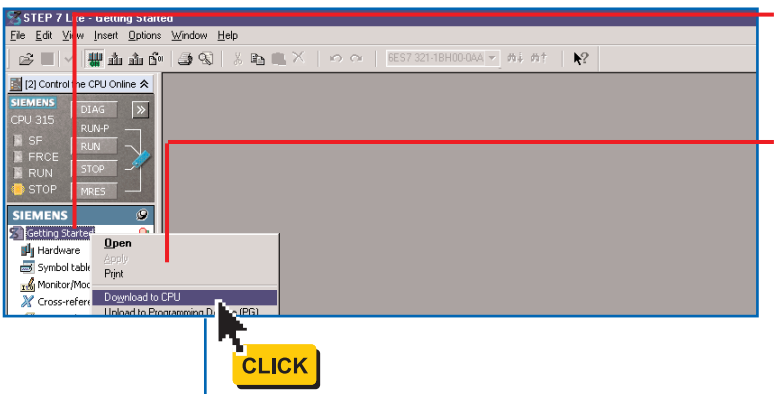
Information on operating states is found in the STEP 7 Lite Help via **F1 > Content > Appendix > Operating Modes**.

Note:
It is of advantage to be familiar with CPU operating modes when you program startup or test routines for the controller, as well as for error diagnostics.

How to download the program to the CPU



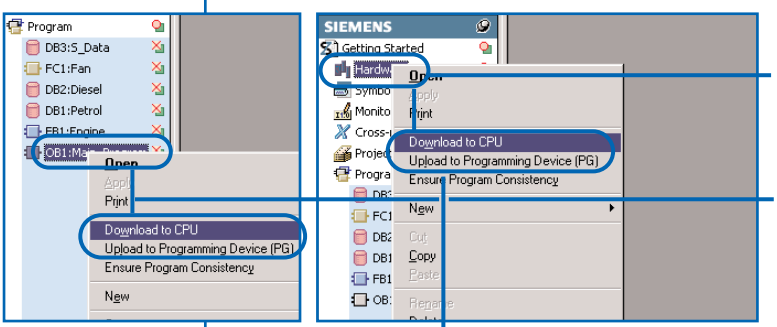
1 Verify that: During program download the mode selector switches at the CPU and in STEP 7 Lite must be set to **Stop** position.



2 Download the complete project to the CPU: Select the ”Getting started” project from the project window.

3 Right-click on the project to open the context-sensitive menu. Select **Download to CPU**.
All project data are downloaded to the CPU, including the hardware configuration.

You can also choose to download only specific blocks or only the hardware configuration to the CPU.

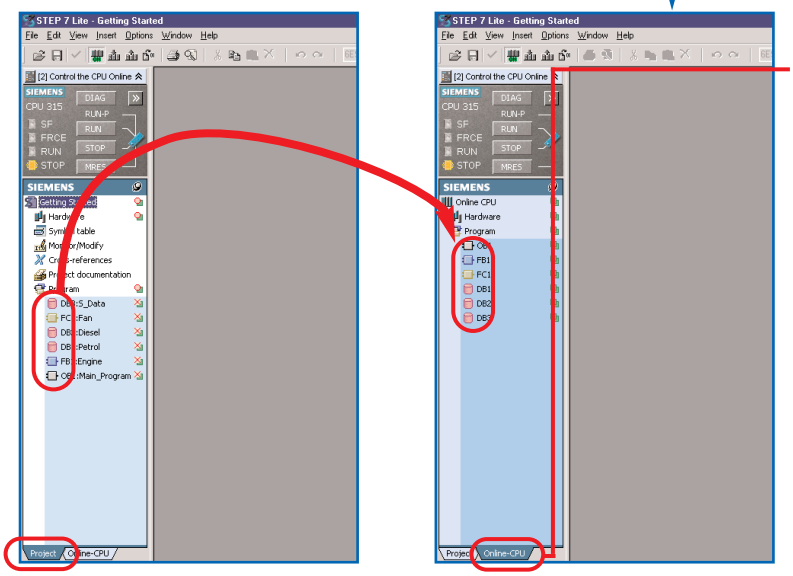


4 In this example, we have selected the hardware configuration for download.

5 In this example, we have selected a single block for download.

5 Depending on the selected elements, STEP 7 Lite lets you perform an **Upload to Programming Device (PG)**.

The view “Online CPU” has changed.

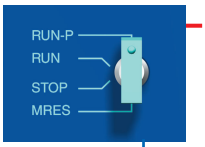


6 Click on **Online CPU**.
You are shown all CPU data.

– **Project** (Offline view)
= Data on the PG

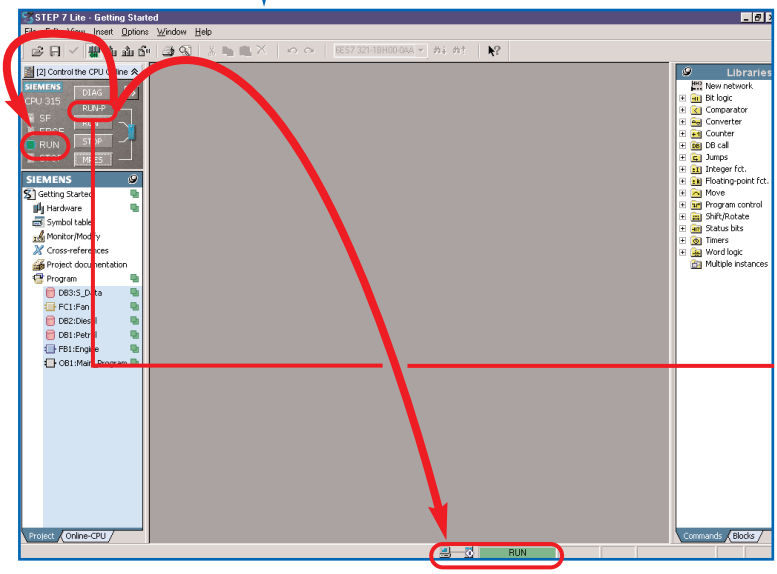
– **Online CPU** (Online view)
= Data on the CPU

How to switch on the CPU and check its operating status



1 Set the mode selector switch to **RUN-P** mode. The green **RUN** LED is on and the red **STOP** LED is off. The CPU is ready for operation.

The cyclic program processing is displayed.



2 Check the CPU:
When the green LED is lit, you can start your program test run.

The LED stays red if an error has occurred. Click on the ”DIAG” button to view the diagnostics buffer for error evaluation (also refer to the section ”Module status and error history”, Page 12.5).

3 Check in STEP 7 Lite:
STEP 7 Lite follows a CPU transition to **RUN-P** mode. Cyclic program processing is indicated by a green background.

Memory reset:

System function blocks (SFBs) and system functions (SFCs) are retentive in the CPU, irrespective of memory reset. The CPU provides these operating system functions. You neither have to download them, nor can you delete them.

How to download specific blocks:

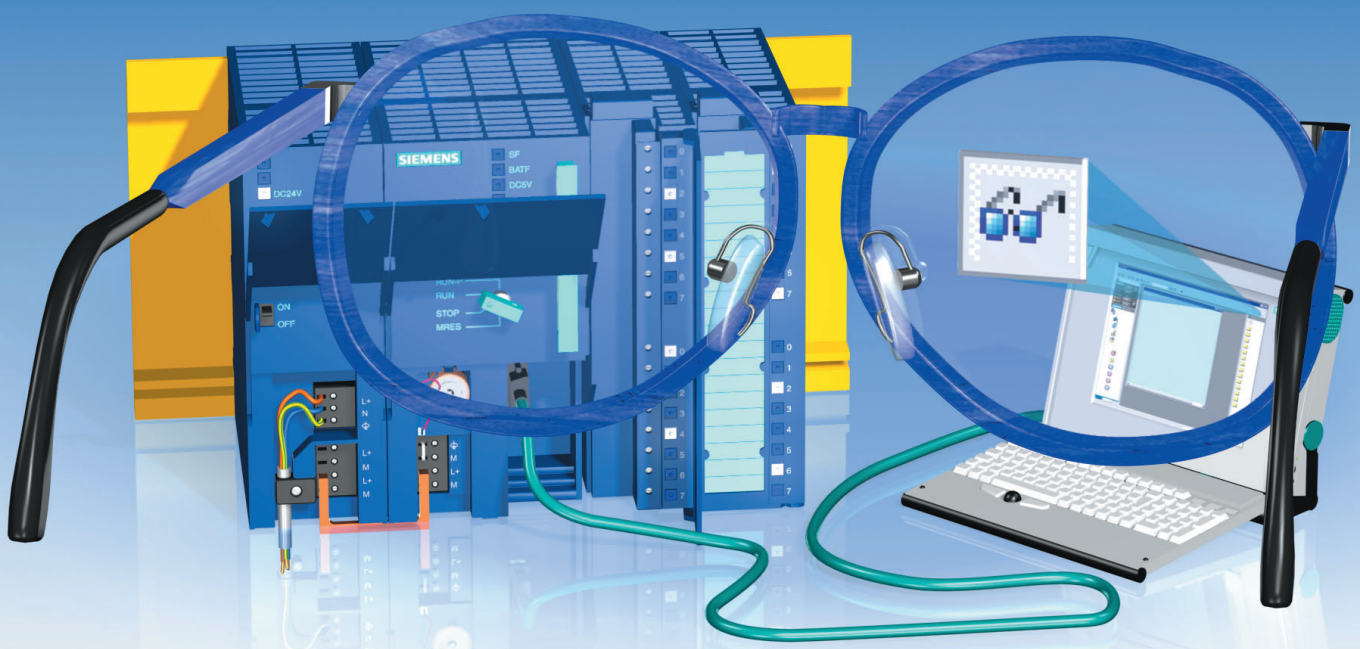
You can increase error response under live conditions by downloading single blocks to the CPU. To enable block downloads, the mode selector switch at the CPU must have been set to ”RUN-P” or ”STOP” position. Blocks downloaded in ”RUN-P” mode are enabled instantaneously. Here, note that:

If error-free blocks are overwritten by faulty blocks, the result is a malfunction of your system.

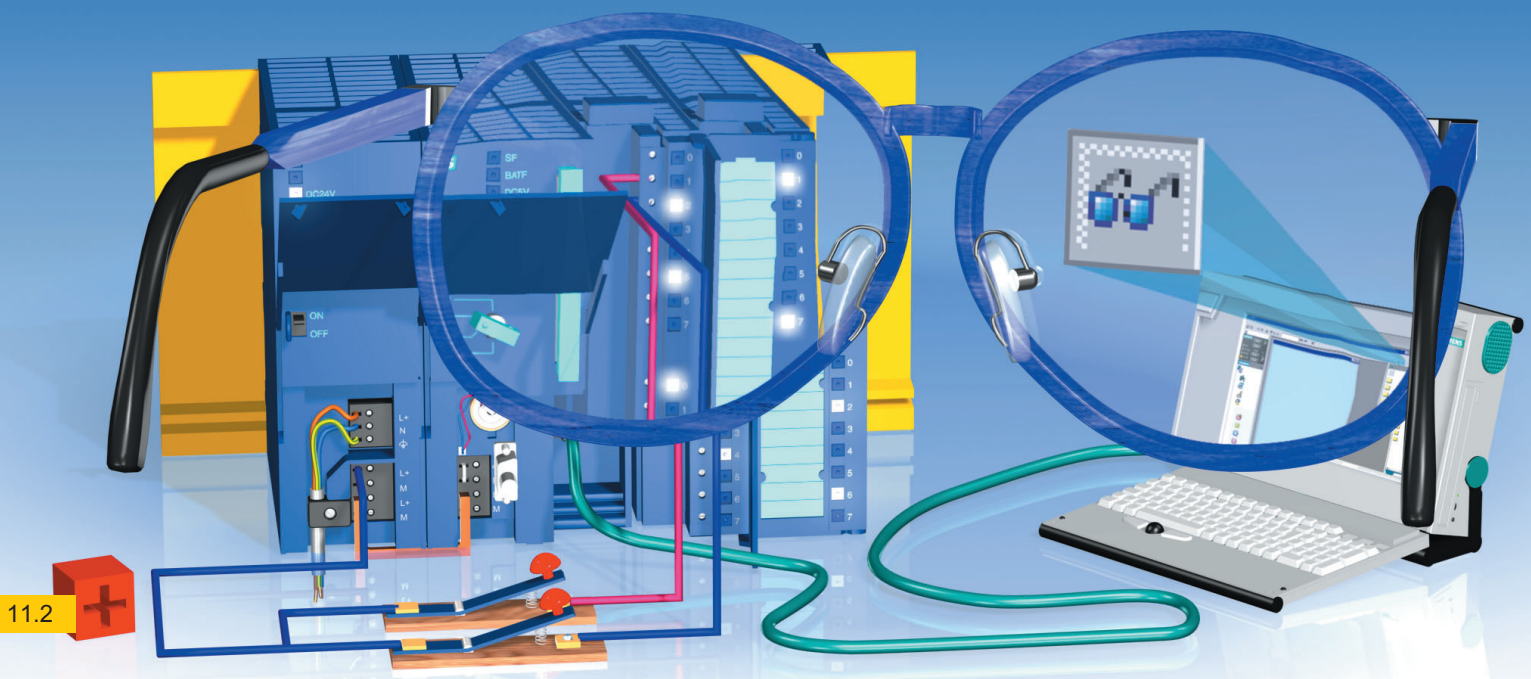
The CPU goes into STOP mode if you neglect the order in which blocks are loaded (e.g. a block that does not yet exist on the CPU is called in OB1.

CPU 31xC:
The mode selector of a CPU 31xC does not ship with a rotary switch, but a toggle switch and there is no RUN-P mode. However, the memory reset procedure is the same. You wil find information on Micro Memory Cards via:
F1 > Index > Micro Memory Card.

11 Program test run



Performing a program test run with program status



STEP 7 Lite lets you perform a program test run on the PLC. Following test run options are available:

1. Test run with program status – for monitoring the program cycle (see Page 11.3 to 11.5).

2. Test run with variable table – for monitoring and controlling addresses, e.g. inputs, outputs, memory bits (see Page 11.6 to 11.10).

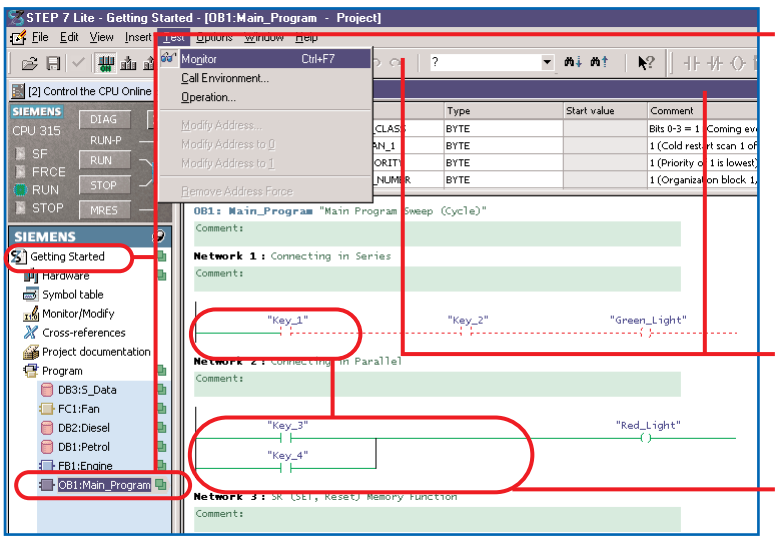
Requirements for test run with program status: The complete program must have been downloaded to the CPU.

Preparations

1 Establish an Online connection.

2 Set the key switch on the CPU to RUN or RUN-P mode.

3 For network 1: Wire the series circuit.
For network 2: Wire the parallel circuit (see the graphic)



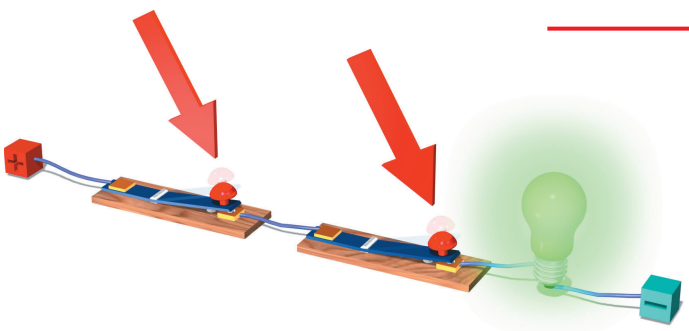
4 Open the ”Getting Started” project, or one of the sample projects, you have downloaded to the CPU.
Open OB1.

Monitoring

5 Call the monitoring function via **Test > Monitor**. This function is only available after you ”Connect Online”.

6 The black network circuits are now displayed in color.

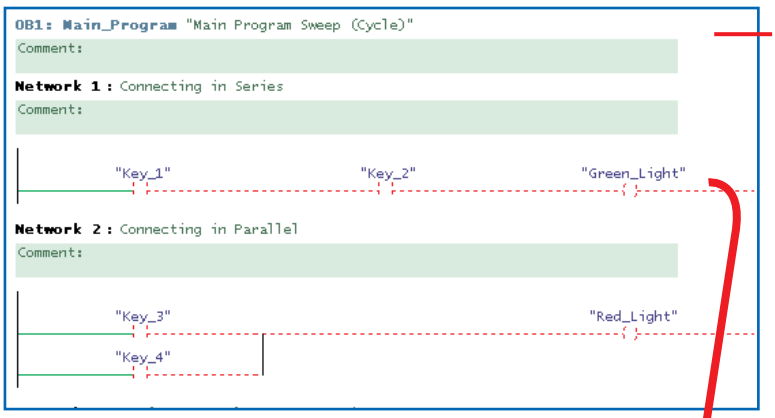
GREEN circuit: Current flow.
RED circuit: No current flow



7 Now close all pushbutton contacts in your test assembly, one after the other and monitor:

- In STEP 7 Lite: How the circuits change their color.
- At the modules: How the LEDs of the I/O modules are switched on and off.

Testing in LAD

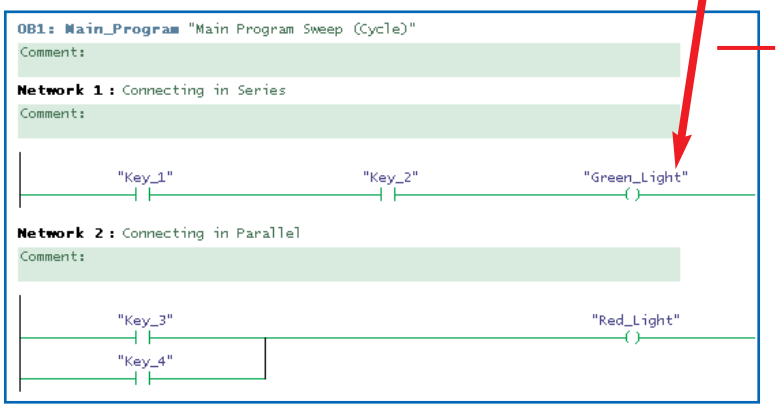


1

Leave all pushbutton contacts open.

In network 1 and 2, the current circuit is closed upstream of pushbuttons 1, 3 and 4. This circuit is indicated in green color. There is no current flow downstream of pushbuttons 1, 3 and 4; this circuit is indicated in red color.

The coloration indicates that the logical operation result is correct up to this position.

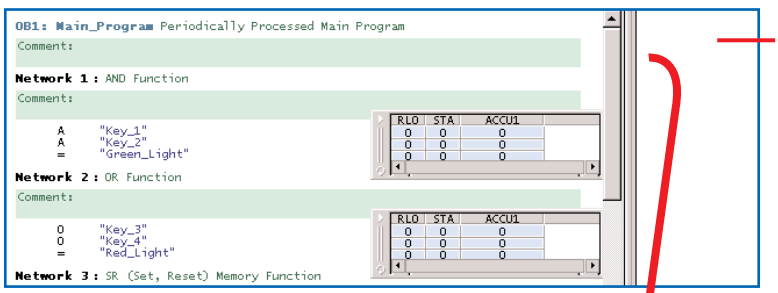


2

Now close pushbutton contacts 1, 2, 3 and 4.
All circuits are now under current.

If you have opened one of our sample projects, you can follow the comments to see which diodes should be lit on the I/O modules.

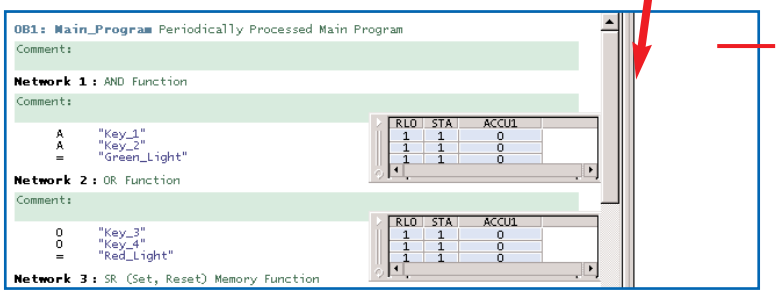
Testing in STL



1

Leave all pushbutton contacts open.

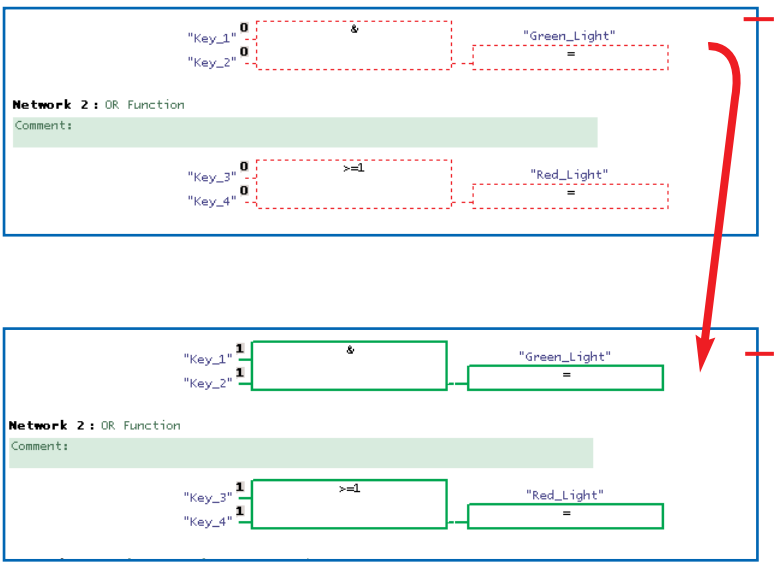
STL shows in a table listing the
– Logical operation result (RLO)
– Status bit (STA)
– Accumulator (ACCU1).



2

Now close pushbutton contacts 1, 2, 3 and 4.
The logical operation result is correct at all positions.

Testing in FBD



1 Leave all pushbutton contacts open.

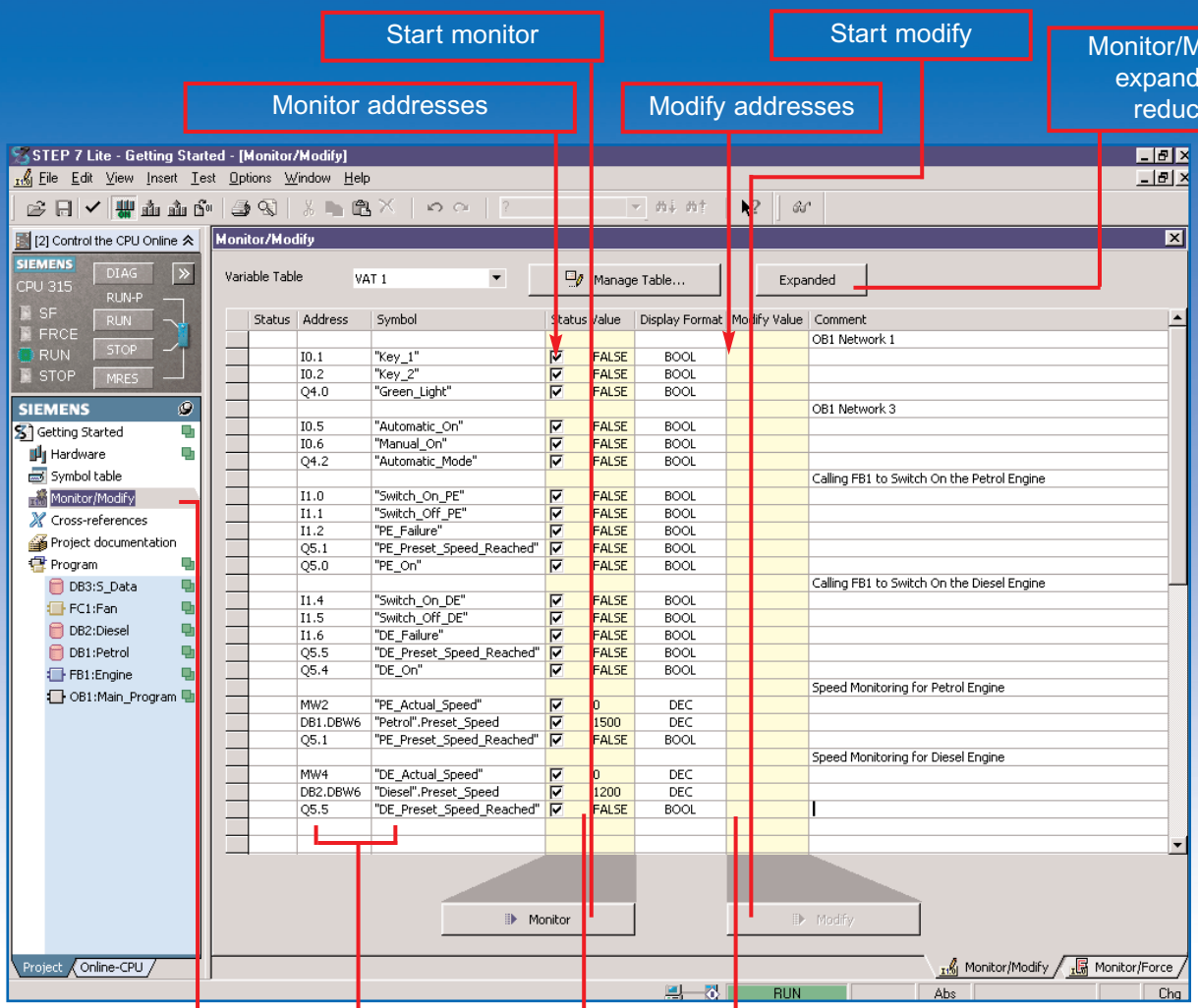
2 Now close pushbuttons contacts 1, 2, 3 and 4.
The coloration indicates that the logical operation result is correct up to this position.

3 Disable **Test > Monitor**. Close the window.

Under **Options > Settings > Block editor** you can modify the type of presentation of test results.

Further information is found via **F1 > Content > Debugging > Testing using program status**.

Monitoring and modifying variables



Call monitor/ modify

Input area for creating the variable table

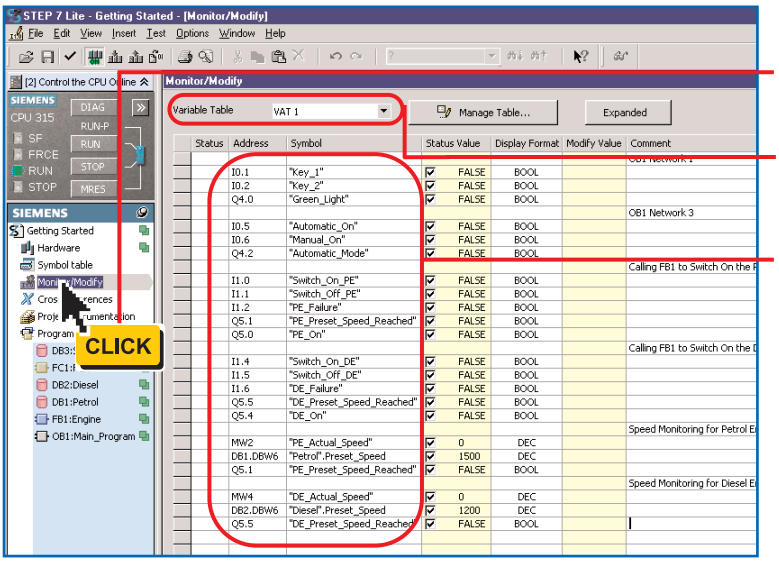
Display area for status value e.g. “true” or “false”

Input area for modify value

You can monitor addresses by writing them to the variable table. Condition is here that you are Online to the CPU.

You can modify addresses by specifying a modify value and then checkmark them. Also required: Online connection and CPU in RUN-P mode.

How to create the variable table



1 Open your ”Getting Started” project. Double-click on **Monitor/Modify**.

2 In the **Variable table** field, create a new table under the name ”VAT 1”.

3 Enter all variables for the ”Getting started” sample,or only the variables you want to modify.

To do this:
Enter ”I01.1” in the **Address** column. Press Return to have ”Key_1” automatically inserted from the symbol table. Complete the table as in our figure. You can also position the cursor in the address column and select the address from the list via shortcut **Ctrl + j**.

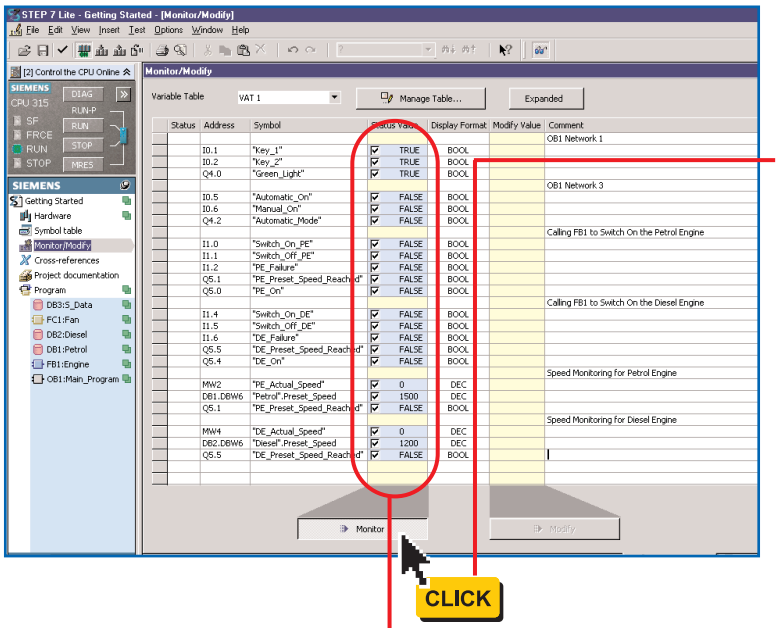
You can also copy the variable table from one of the sample projects.

4 Open one of the sample projects in a second instance of STEP 7 Lite. Again, click on **Monitor/Modify** to open **VAT 1**.
Highlight the complete table via **Ctrl + a** and copy it to the clipboard via **Ctrl + c**.

5 Go to your ”Getting Started” project. Paste the data from the clipboard to your file via **Ctrl + v**

How to monitor variables

When monitoring the variables, you can perform program test runs as well as check your hardware for error-free functioning.



1 Close contacts ”Key_1” and ”Key_2” in your test assembly.

2 Click on **Monitor**.
Now, the status value is highlighted on a blue background color and the variables are monitored.

3 You can monitor,

- how the displayed value in the **Status value** column is toggled from ”FALSE” to ”TRUE” and

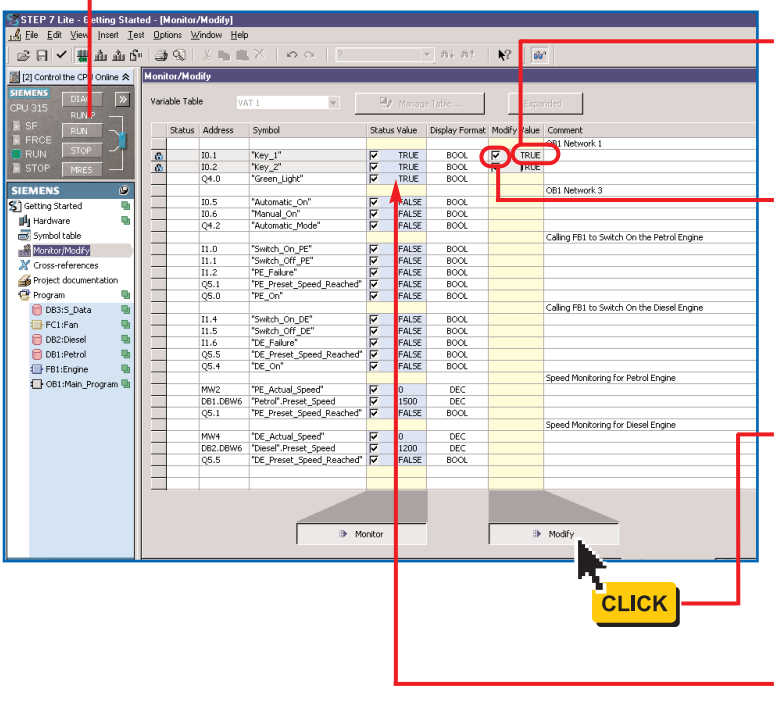
- at the same time, how I/O module LEDs are switched on or off when you press any of the pushbuttons on your test assembly.

4 For your program or hardware test run, perform a check the plausibility of your combination status
– Pushbutton contact open/close
– LED on/off
– Variable true/false.

How to modify variables

While you are modifying variables, you can assign them values in order to simulate specific program runtime situations.

Example:



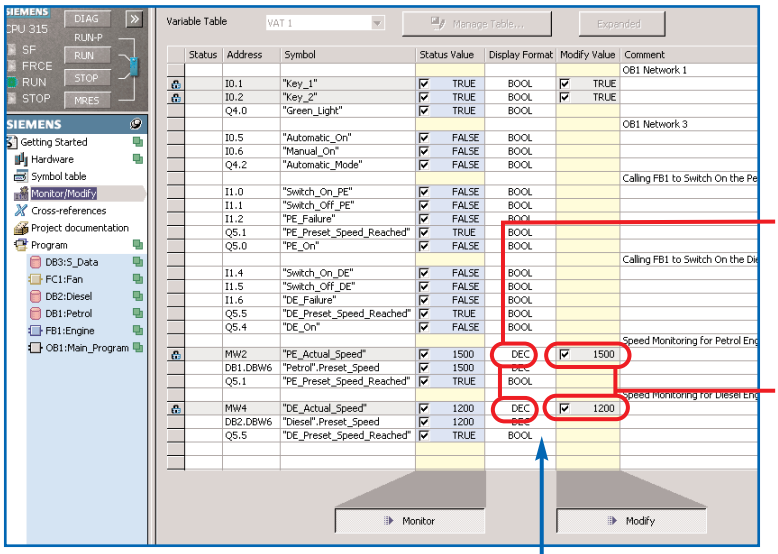
1. For modifying, your CPU must also be set to RUN-P mode. In this case, **Monitor** stays enabled.

2. In the **Modify value** column, enter the value ”TRUE”, for example. This modify value is not yet enabled.

3. Checkmark the options box to enable the modify value. This box is displayed immediately after you have entered the modify value.

4. Click on **Modify**.

5. Monitor the effects of the modified variables in the **Status value** column.



You can control binary as well as non-binary addresses. For the latter, you should first select the display format and then enter a respective modify value.

6 If you specify DEC for controlling speed setpoints in the **Display format** column,

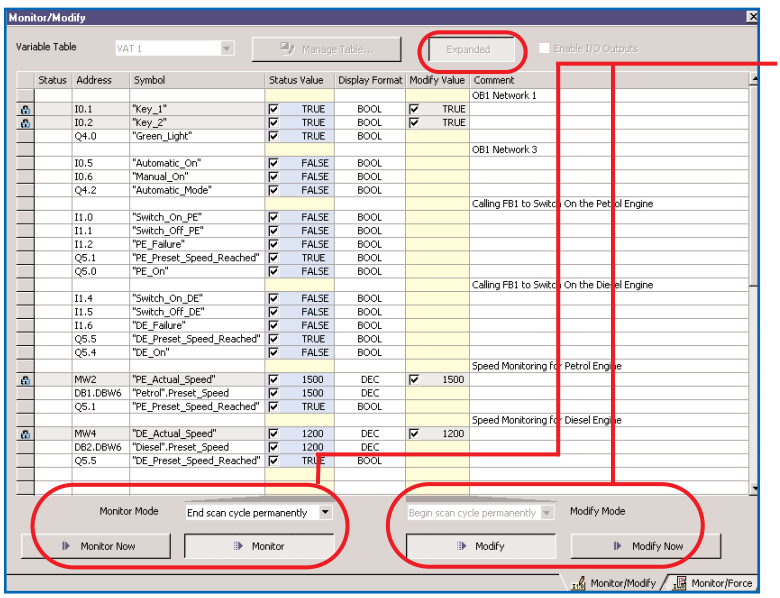
7 you can enter the speed modify values ”1200” and ”1500” in decimal format.

A cell highlighted on a red background color indicates that you have entered a value which is incompatible to display format. In this case, the **Modify value** column does not display the option boxes anymore.

8 You can change the display format of a variable, after you have clicked on the type in the **Display format** column.

11.10

Expanded view Monitor/Modify

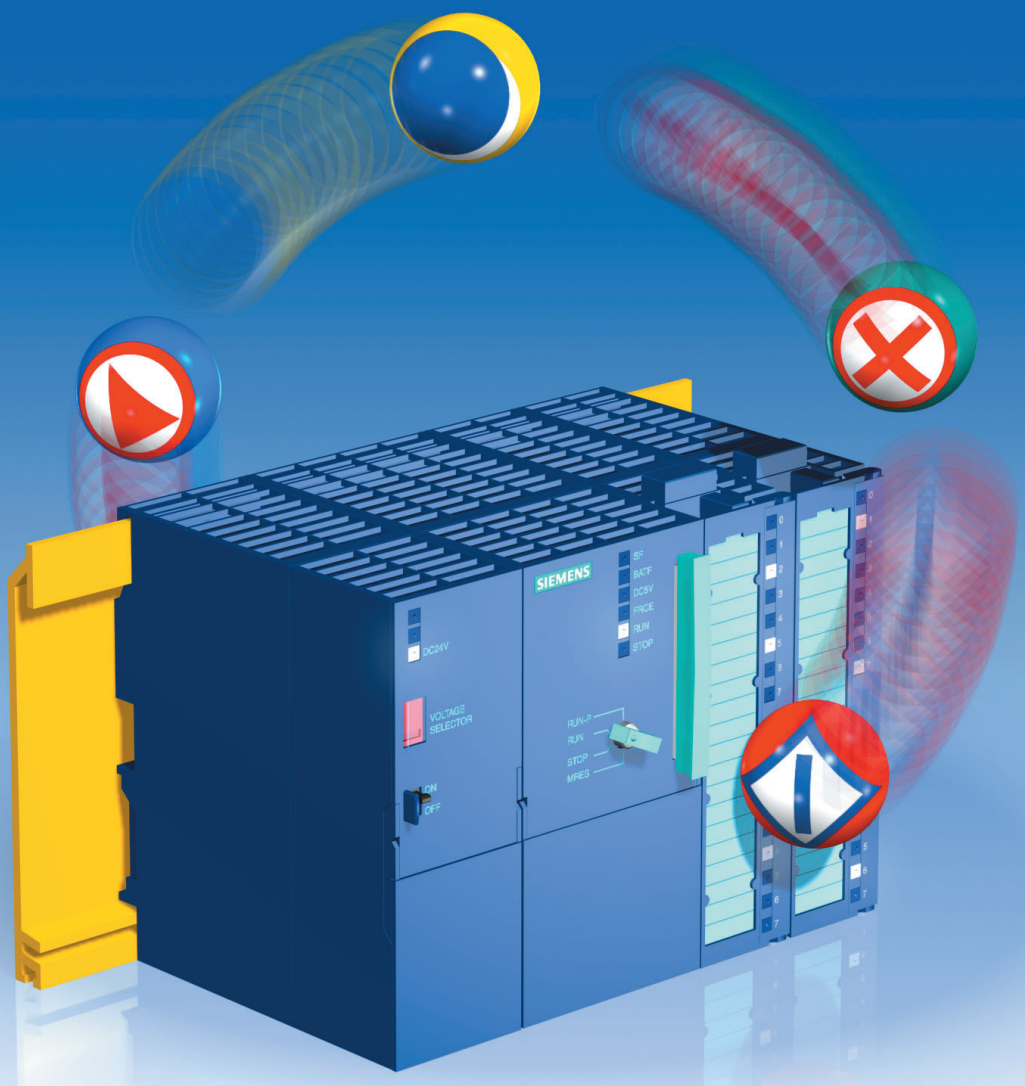


9 You can choose between different monitoring and modifying modes in the expanded view Monitor/Modify e.g. you can choose whether the control value:

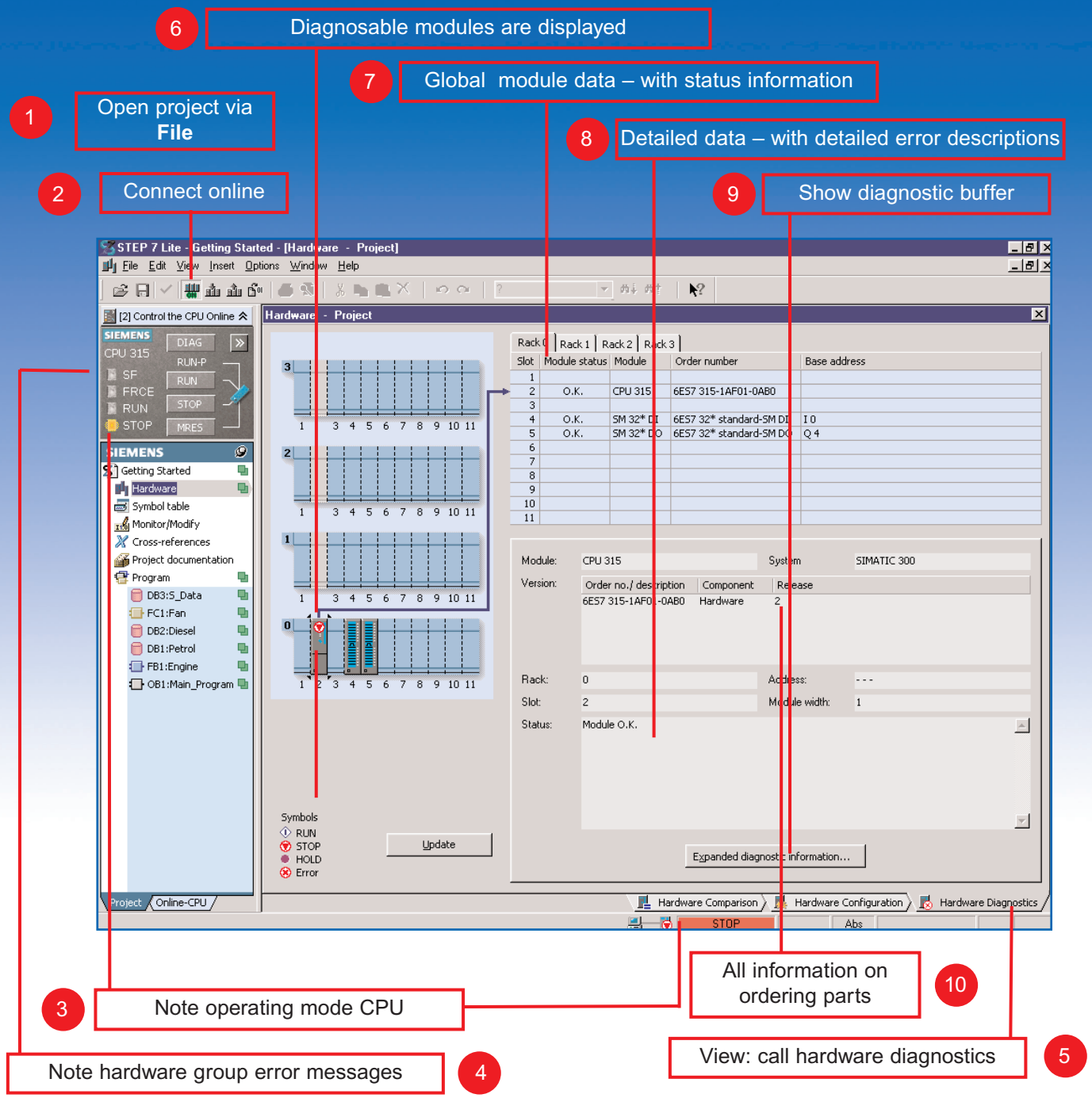
– should be set only when the cyclic status of OB1 is defined, e.g., when the cycle starts or

– or immediately in the middle of a running program.

12 Error diagnostics



A quick glance at HW diagnostics



12.2

STEP 7 Lite offers extensive error diagnostics at a glance after a hardware error has occurred in the PLC station. The code number will indicate the diagnostics path. This view is only available in online mode.

An error has occurred?

The status of a PLC in your plant has changed to STOP. The CPU key switch is set to **RUN** position.

A hardware error has occurred.

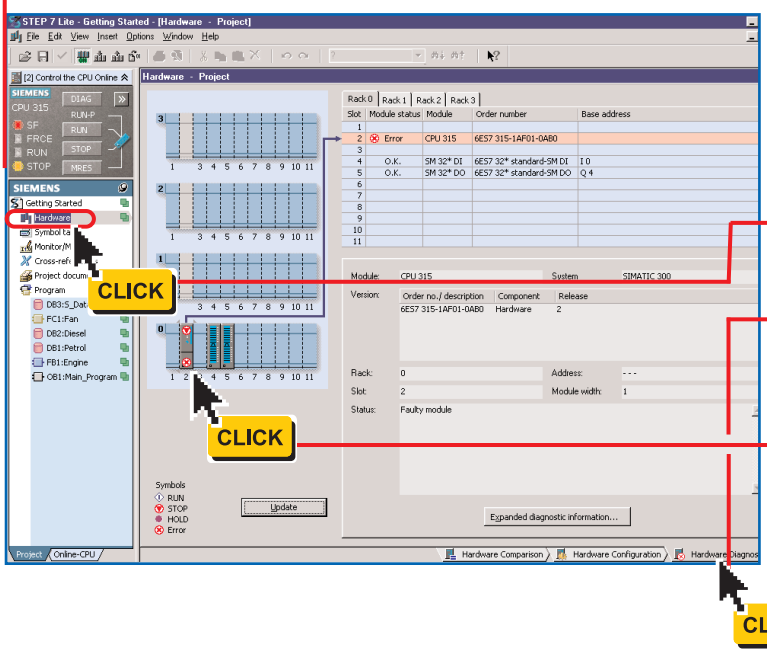
Troubleshooting

Set the key switch to **STOP** position.

1. Open the project which belongs to this PLC station and in which the hardware configuration error has occurred.

2. Establish an Online connection between the PG and the PLC (compare Chapter 10).

3. You already know that the CPU has changed to STOP mode. You need this information, for example, in situations not allowing visual contact to the CPU during commissioning.

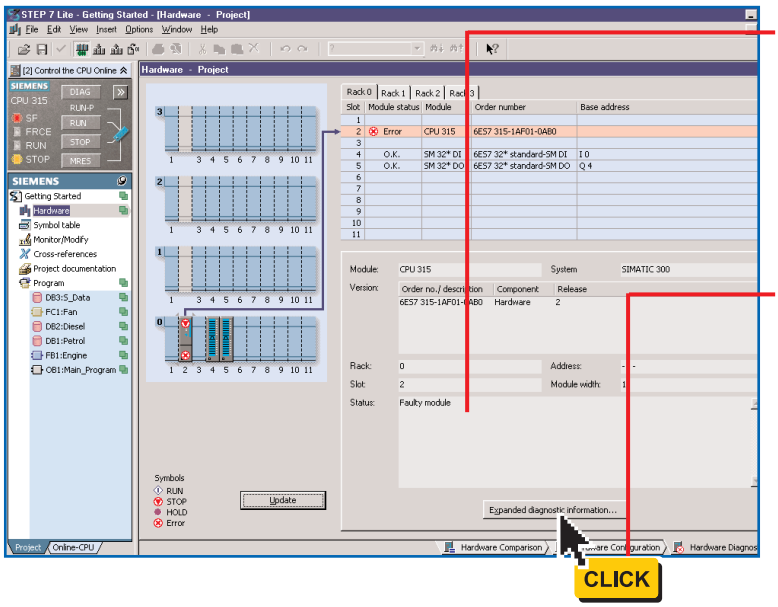


4. Double-click on **Hardware**, on the left side of your project window.

5. Open the **Hardware Diagnostics** view

6. A pictogram identifies the defective module in the rack. Click on this module.

Error diagnostics

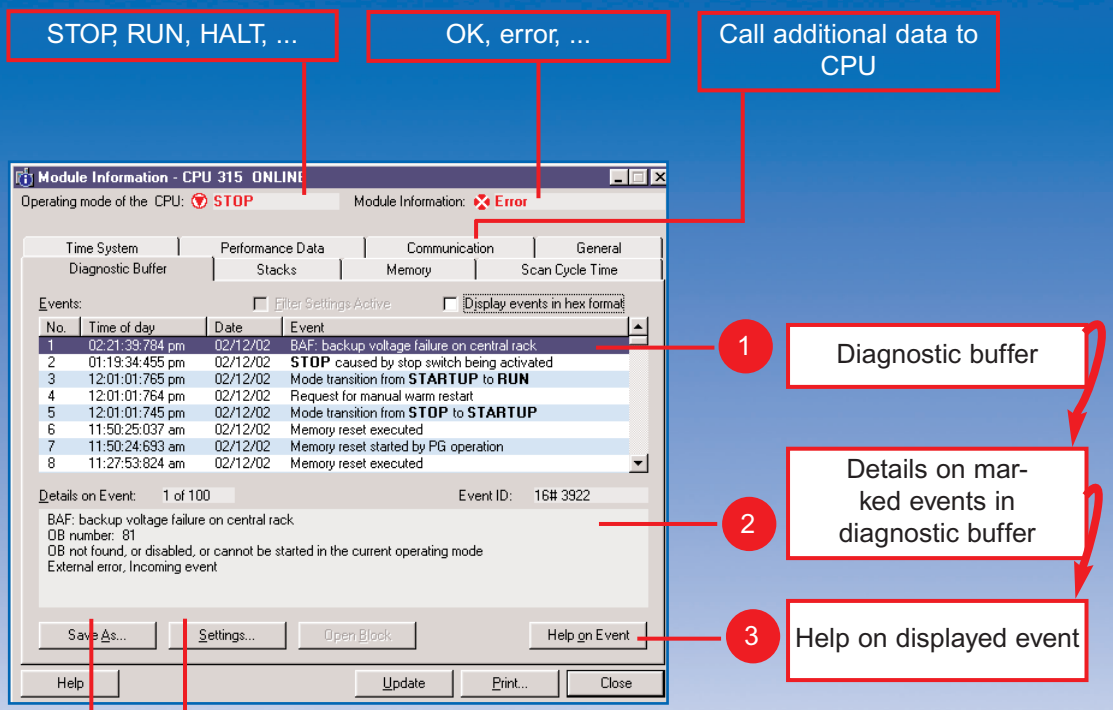


8 You will receive a plain text message:

- Module O. K.
- Faulty module.

9 If you need further information, please click on **Expanded diagnostics information**.

Module status and error history



Before you go ahead and remove a module you consider to be faulty, call the **Module information** function (Call, see Page 12.4) to check its status.

The diagnostics buffer, for example, records errors and all CPU events. You will need this information especially in situations requiring the distinction between a sequential error and the actual error event.

Further information is found via **F1 > Index > Calling the module information**.

If a wire break has occurred:
Check the wiring or whether any measuring range modules are inserted incorrectly.

When the CPU goes to STOP:
Evaluate message output from the diagnostics buffer. The fastest possible way to access this buffer is to click on the ”DIAG” button in the CPU operating panel.

With faulty module:
Switch off load voltage before you remove the module.

13 Index

6ES7810-3CC07-0YA05

Siemens Aktiengesellschaft

Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D-90327 Nuernberg

www.siemens.com/automation

Order No. 6ES7810-3CC07-0YA05

A5E00293886-01