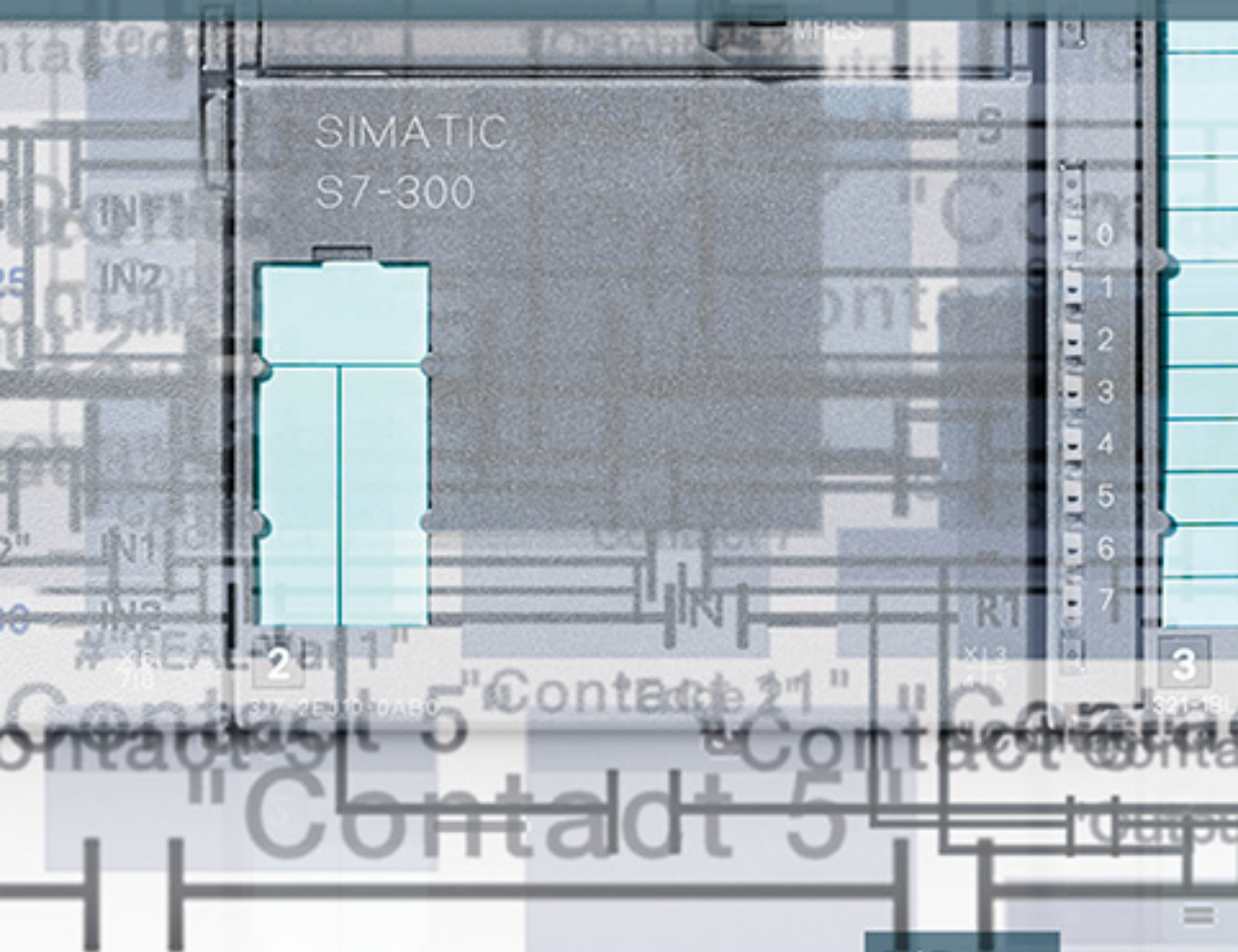
SIEMENS

Hans Berger

Automating with SIMATIC S7-300 inside TIA Portal

Configuring, Programming and Testing with STEP 7 Professional



2nd Edition

Berger Automating with SIMATIC S7-300 inside TIA Portal

Automating with SIMATIC S7-300 inside TIA Portal

Configuring, Programming and Testing
with STEP 7 Professional

by Hans Berger

2nd edition, 2014

Publicis Publishing

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.d-nb.de.

www.publicis-books.de

Print ISBN 978-3-89578-443-9
ePDF ISBN 978-3-89578-924-3

2nd edition, 2014

Editor: Siemens Aktiengesellschaft, Berlin and Munich
Publisher: Publicis Publishing, Erlangen

Printed in Germany

Preface

The SIMATIC automation system unites all of the subsystems of an automation solution under a uniform system architecture to form a homogenous whole from the field level right up to process control.

The *Totally Integrated Automation* (TIA) concept permits uniform handling of all automation components using a single system platform and tools with uniform operator interfaces. These requirements are fulfilled by the SIMATIC automation system, which provides uniformity for configuration, programming, data management, and communication.

This book describes the hardware components of the SIMATIC S7-300 automation system with standard controllers and the features provided for designing a distributed control concept with PROFIBUS and PROFINET. To permit communication with other automation systems, the controllers offer integrated bus interfaces for multipoint interface (MPI), PROFIBUS, and Industrial Ethernet.

The STEP 7 Professional engineering software inside TIA Portal makes it possible to use the complete functionality of the S7-300 controllers. STEP 7 Professional is the common tool for hardware configuration, generation of the user program, and for program testing and diagnostics.

STEP 7 Professional provides five programming languages for generation of the user program: Ladder logic (LAD) with a graphic representation similar to a circuit diagram, function block diagram (FBD) with a graphic representation based on electronic circuitry systems, statement list (STL) with formulation of the control task as a list of commands at machine level, a high-level Structured Control Language (SCL) similar to Pascal, and finally GRAPH as a sequencer with sequential processing of the user program.

STEP 7 Professional supports testing of the user program by means of watch tables for monitoring, control and forcing of tag values, by representation of the program with the current tag values during ongoing operation, and by offline simulation of the programmable controller.

This book describes the configuration, programming, and testing of the S7-300 automation system with the STEP 7 Professional engineering software Version 12 with Service Pack 1 Update 2.

Erlangen, June 2014 Hans Berger

The contents of the book at a glance

Start

Overview of the SIMATIC S7-300 automation system.

Introduction to the SIMATIC STEP 7 Professional V12 engineering software.

The basis of the automation solution: Creating and editing a project.

SIMATIC S7-300 automation system

Overview of the SIMATIC S7-300 modules: Design of an automation system, CPUs, signal, function and communication modules.

Device configuration

Configuration of a station, parameterization of modules, and networking of stations.

Tags, addressing, and data types

The properties of inputs, outputs, I/O, bit memories, data, and temporary local data as operand areas, and how they are addressed: absolute, symbolic, and indirect.

Description of elementary and compound data types, data types for block parameters, pointers, and user data types.

Program execution

How the CPU responds in the STARTUP, RUN, and STOP modes.

How the user program is structured with blocks, what the properties of these blocks are, and how they are called.

How the user program is executed: startup characteristics, main program, interrupt processing, troubleshooting, and diagnostics.

The program editor

Working with the PLC tag table, creating and editing code and data blocks, compiling blocks, and evaluating program information.

The ladder logic programming language LAD

The characteristics of LAD programming; series and parallel connection of contacts, the use of coils, standard boxes, Q boxes, and EN/ENO boxes.

The function block diagram programming language FBD

The characteristics of FBD programming; boxes for binary logic operations, the use of standard boxes, Q boxes, and EN/ENO boxes.

The statement list programming language STL

The characteristics of STL programming; programming of binary logic operations, application of digital functions, and control of program execution.

The structured control language SCL

The characteristics of SCL programming; operators and expressions, working with binary and digital functions, control of program execution using control statements.

The S7-GRAPH sequential controller

What a sequential control is, and what its elements are: sequencers, steps, transitions, and branches. How a sequential control is configured using S7-GRAPH.

Description of the control functions

Basic functions: Functions for binary signals: binary logic operations, memory functions, edge evaluations, SIMATIC and IEC timer and counter functions.

Digital functions: Functions for digital tags: transfer, comparison, arithmetic, math, conversion, shift, and logic functions.

Program flow control: Working with status bits, programming jump functions, calling and closing blocks, using the master control relay.

Online operation and program test

Connecting a programming device to the PLC station, switching on online mode, transferring the project data, and protecting the user program.

Loading, modifying, deleting, and comparing the user blocks.

Working with the hardware diagnostics and testing the user program.

Distributed I/O

Overview: The ET 200 distributed I/O system.

How a PROFINET IO system is configured, and what properties it has.

How a PROFIBUS DP master system is configured, and what properties it has.

How an actuator/sensor interface system is configured, and what properties it has.

Communication

The properties of S7 basic communication and of S7 communication, and with what communication functions they are programmed.

The communication functions used to implement open user communication.

Appendix

How external source files are created and imported for STL and SCL blocks.

How a project created using STEP 7 V5.x is migrated to the TIA Portal.

How the user program is tested offline using the S7-PLCSIM simulation software.

How the Web server is configured in the CPU, and what features it offers.

How block parameters and local tags are saved in the memory.

Table of contents

1 Introduction

1.1 Overview of the S7-300 automation system

SIMATIC S7-300 is the modular mini PLC system for the lower and medium performance ranges (Fig. 1.1). Different versions of the controllers allow the performance to be matched to the respective application. Depending on the requirements, the programmable controller can be expanded by input/output modules for digital and analog signals in up to four racks with eight modules each.

Further expansion with input/output modules is made possible by the distributed I/O over PROFIBUS or PROFINET. Special designs of these modules for increased mechanical demands allow their installation directly on site on the machine or plant. STEP 7 is used to configure and program the SIMATIC S7-300 controllers. Data exchange between the controllers, the distributed I/O, and the programming device is carried out over SIMATIC NET.

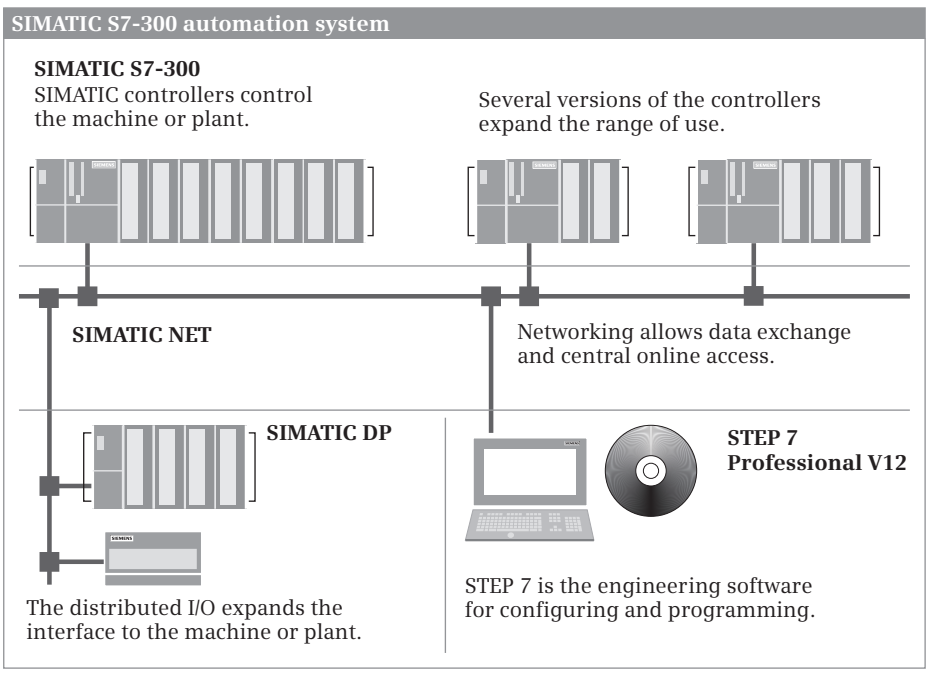


Fig. 1.1 Components of the SIMATIC S7-300 automation system

1.1.1 SIMATIC S7-300 programmable controller

The most important components of an S7-300 programmable controller are shown in Fig. 1.2.

The **CPU** contains the operating system and the user program. The user program is saved powerfail-proof on the *Micro Memory Card (MMC)*, which is inserted in the CPU. The user program is executed in the CPU's work memory. The bus interfaces present on the CPU establish the connection to other programmable controllers.

Signal modules (SM) are responsible for the connection to the controlled plant. These input and output modules are available for digital and analog signals.

The **function modules (FM)** are signal-preprocessing, "intelligent" I/O modules which prepare signals coming from the process independent of the CPU and either return them directly to the process or make them available at the CPU's internal interface. Function modules are responsible for handling functions which the CPU cannot usually execute quickly enough, such as counting pulses, positioning, or controlling drives.

The **CP modules** allow data transfer in excess of the possibilities provided by the standard interfaces with regard to protocols and communication functions.

In the case of an expansion, the **interface modules (IM)** connect the central rack to a maximum of three expansion racks.

Finally, a **power supply module** provides the voltage required by the programmable controller.

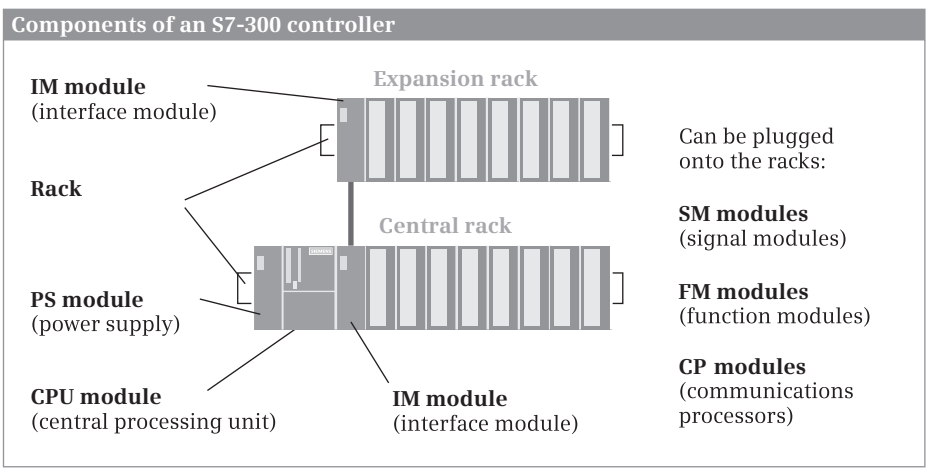


Fig. 1.2 Components of an S7-300 controller

1.1.2 Overview of STEP 7 Professional V12

STEP 7 is the central automation tool for SIMATIC. STEP 7 requires authorization (licensing) and is executed on the current Microsoft Windows operating systems. Configuration of an S7-300 controller is carried out in two views: the Portal view and the Project view.

The **Portal view** is task-oriented.

In the Start portal you can open an existing project, create a new project, or migrate a project. A "project" is a data structure containing all the programs and data required for your automation task. The most important STEP 7 tools and functions can be accessed from here via further portals (Fig. 1.3):

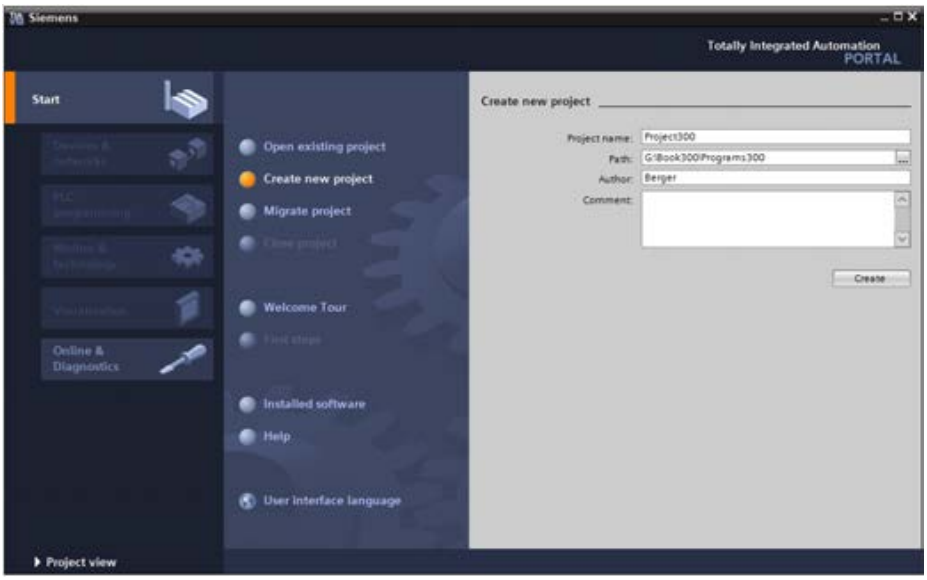


Fig. 1.3 Tools in the Start portal of STEP 7 Professional V12

- In the *Devices & networks* portal you configure the programmable controllers, i.e. you position the modules in a rack and set their parameters.
- In the *PLC programming* portal you create the user program in the form of individual sections referred to as "blocks".
- The *Visualization* portal provides the most important tools for configuration and simulation of HMI systems using SIMATIC WinCC.
- In the *Motion & Technology* portal, you insert a technology object for *PID Control* and edit it.

▷ The *Online & Diagnostics* portal allows you to connect the programming device online to a CPU. You can control the CPU's operating modes, and transfer and test the user program.

The **Project view** is an object-oriented view with several windows whose contents change depending on the current activity. In the *Device configuration*, the focal point is the working area with the device to be configured. The Device view includes the rack and the modules which have already been positioned (Fig. 1.4). A further window – the inspector window – displays the properties of the selected module, and the task card provides support by means of the hardware catalog with the available modules. The Network view allows networking between PLC and HMI stations.

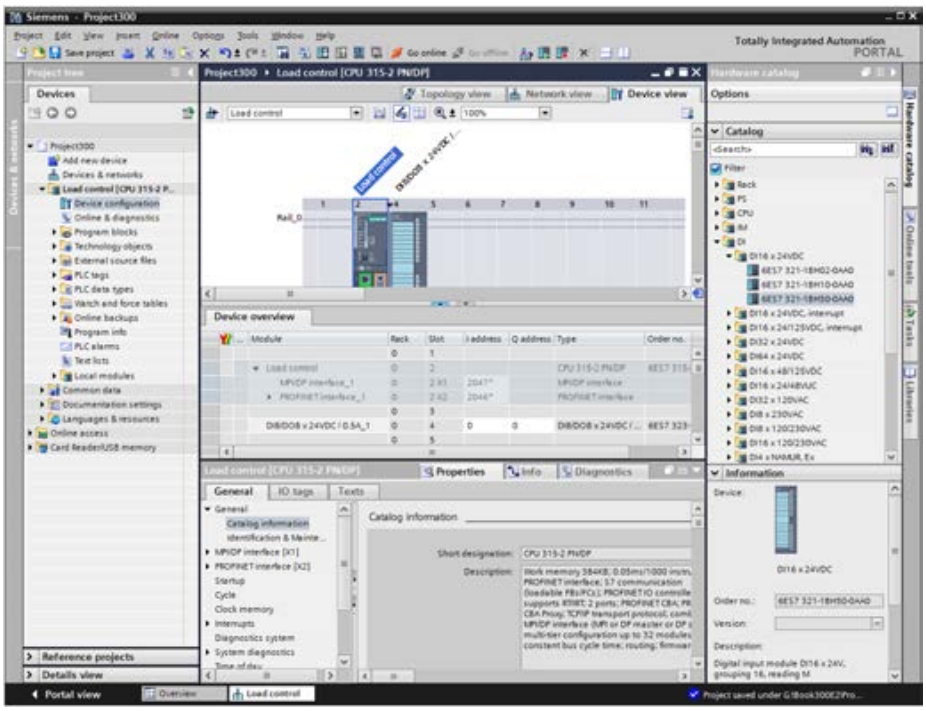


Fig. 1.4 Example of a Project view: working area of the device configuration

When carrying out *PLC programming*, you edit the selected block in the working area. You are again shown the properties of the selected object in the inspector window, where you can adjust them. In this case, the task card contains the program elements catalog with the available program elements and instructions. The same applies to the processing of PLC tags or to online program testing using watch tables.

And you always have a view of the *project tree*. This contains all objects of the STEP 7 project. You can therefore select an object at any time, for example a program block or watch table, and edit this object using the corresponding editors which start automatically when the object is opened.

1.1.3 Five programming languages

You can select between five programming languages for the user program: ladder logic (LAD), function block diagram (FBD), statement list (STL), structured control language (SCL), and sequential control (GRAPH).

Using the **ladder logic**, you program the control task based on the circuit diagram. Operations on binary signal states are represented by serial or parallel arrangement of contacts and coils (Fig. 1.5). Complex functions such as arithmetic functions are represented by boxes which you arrange like contacts or coils in the ladder logic.

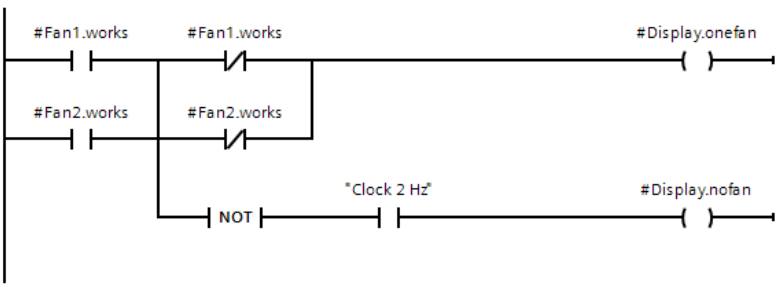


Fig. 1.5 Example of representation in ladder logic

Using the **function block diagram**, you program the control task based on electronic circuitry systems. Binary operations are implemented by linking AND and OR functions and are terminated by memory boxes (Fig. 1.6). Complex boxes are used to handle the operations on digital tags, for example with arithmetic functions.

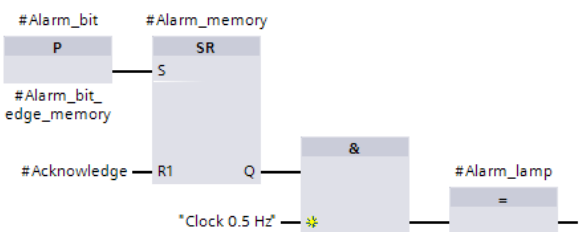


Fig. 1.6 Example of representation in function block diagram

Using the **statement list**, you program the control task using a sequence of statements. Every STL statement contains the specification of what has to be done, and possibly an operand with which the operation is executed. STL is equally suitable for binary and digital operations and for programming complex open-loop control tasks (Fig. 1.7).

```
1  //Motor memory
2        A     "Switch-on_manual"
3        A     "Manual_mode"
4        O
5        A     "Switch-on_automatic"
6        AN    "Manual_mode"
7        S     #Motor_memory        //Set memory
8
9        O     "Switch-off_manual"
10       O     "Switch-off_automatic"
11       ON    "Motor_fault"
12       R     #Motor_memory        //Reset memory
13
```

Fig. 1.7 Example of STL statements

```
19  Write_register: //********************************************************
20 IF #Level = #Register_length - 1
21     THEN #Full := TRUE;
22     ELSE #Register[#Write_pointer] := #Input_value;
23          #Level := #Level + 1;
24          IF #Write_pointer = #Register_length
25              THEN #Write_pointer := 0;
26              ELSE #Write_pointer := #Write_pointer + 1;
27          END_IF;
28          #Empty := FALSE;
29  END_IF; RETURN;
```

Fig. 1.8 Example of SCL statements

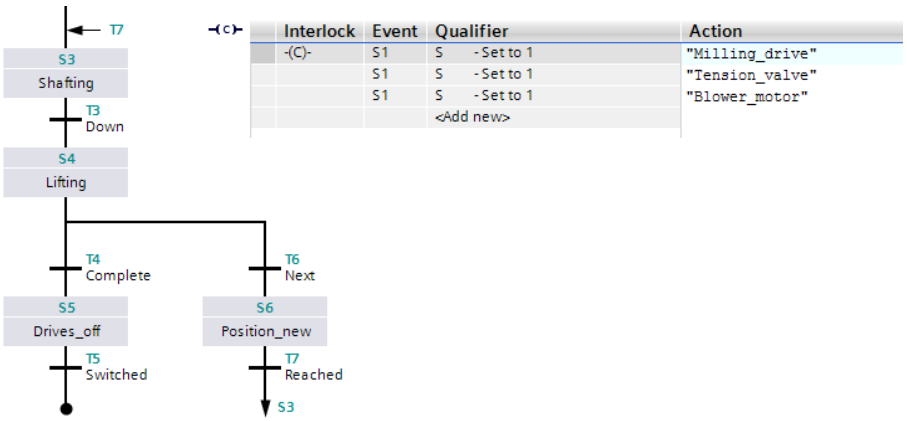


Fig. 1.9 Example of a GRAPH sequencer and step configuration

Structured control language is particularly suitable for programming complex algorithms or for tasks in the area of data management. The program is made up of SCL statements which, for example, can be value assignments, comparisons, or control statements (Fig. 1.8).

Using **GRAPH**, you program a control task as a sequential control in which a sequence of actions prevails. The individual steps and branches are enabled by step enabling conditions which can be programmed using LAD or FBD (Fig. 1.9).

1.1.4 Execution of the user program

After the power supply has been switched on, the control processor checks the consistency of the hardware and parameterizes the modules. A startup program is then executed once, if present. The startup program belongs to the user program which you produce. Modules can be initialized, for example, by the startup program.

The user program is usually divided into individual sections called “blocks”. Organization blocks (OB) represent the interface between operating system and user program. The operating system calls an organization block for specific events and the user program is then processed in it (Fig. 1.10).

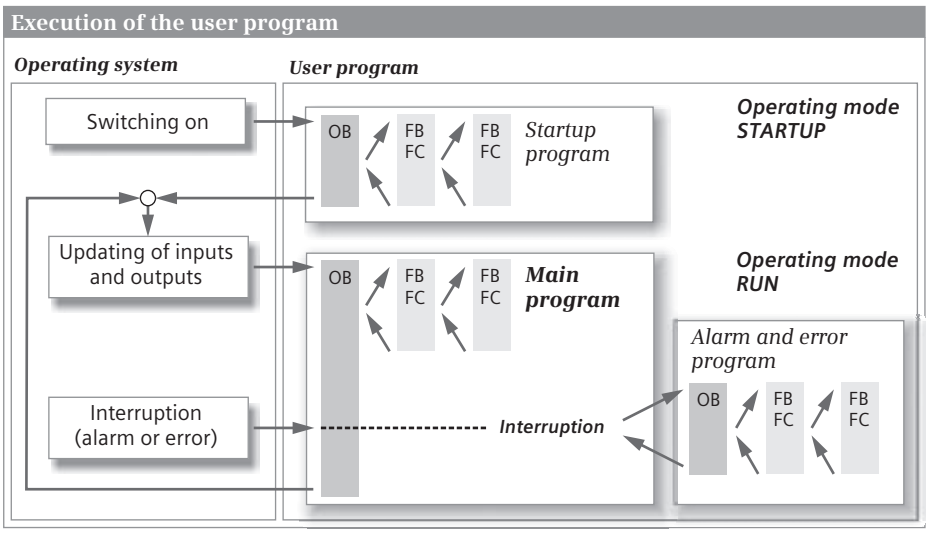


Fig. 1.10 Execution of the user program

Function blocks (FB) and functions (FC) are available for structuring the program. Function blocks have a memory in which local tags are saved permanently; functions do not have this memory.

Program instructions are available for calling function blocks and functions (start of execution). Each block call can be assigned inputs and outputs, referred to as “block parameters”. During calling, tags can be transferred with which the program in the block is to work. In this manner, a block can be repeatedly called with a certain function (e.g. addition of three tags), but with different parameters sets (e.g. for different calculations) (Fig. 1.11).

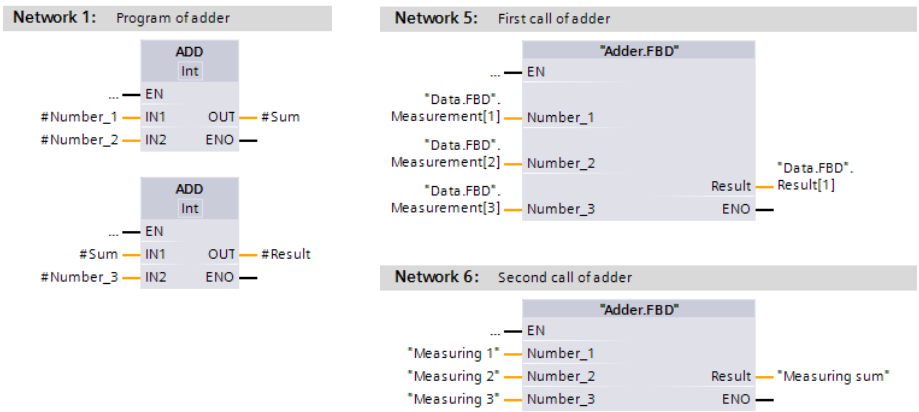


Fig. 1.11 Example of two block calls with different tags in each case

The data of the user program is saved in data blocks (DB). Instance data blocks have a fixed assignment to a call of a function block and are the tag memory of the function block. Global data blocks contain data which is not assigned to any block.

Following a startup, the control processor updates the input and output signals in the process images and calls the organization block OB 1. The main program is present here. Structuring is also possible (and recommended) in the main program. Once the main program has been processed, the control processor returns to the operating system, retains (for example) communication with the programming device, updates the input and output signals, and then recommences with execution of the main program.

Cyclic program execution is a feature of programmable logic controllers. The user program is even executed if no actions are requested “from outside”, e.g. if the controlled machine is not running. This provides advantages when programming: For example, you program the ladder logic as if you were drawing a circuit diagram, or program the function block diagram as if you were connecting electronic components. Roughly speaking, a programmable controller has a characteristic like, for example, a contactor or relay control: the many programmed operations are effective quasi simultaneously “in parallel”.

In addition to the cyclically executed main program, it is possible to carry out interrupt-controlled program execution. You must enable the corresponding interrupt