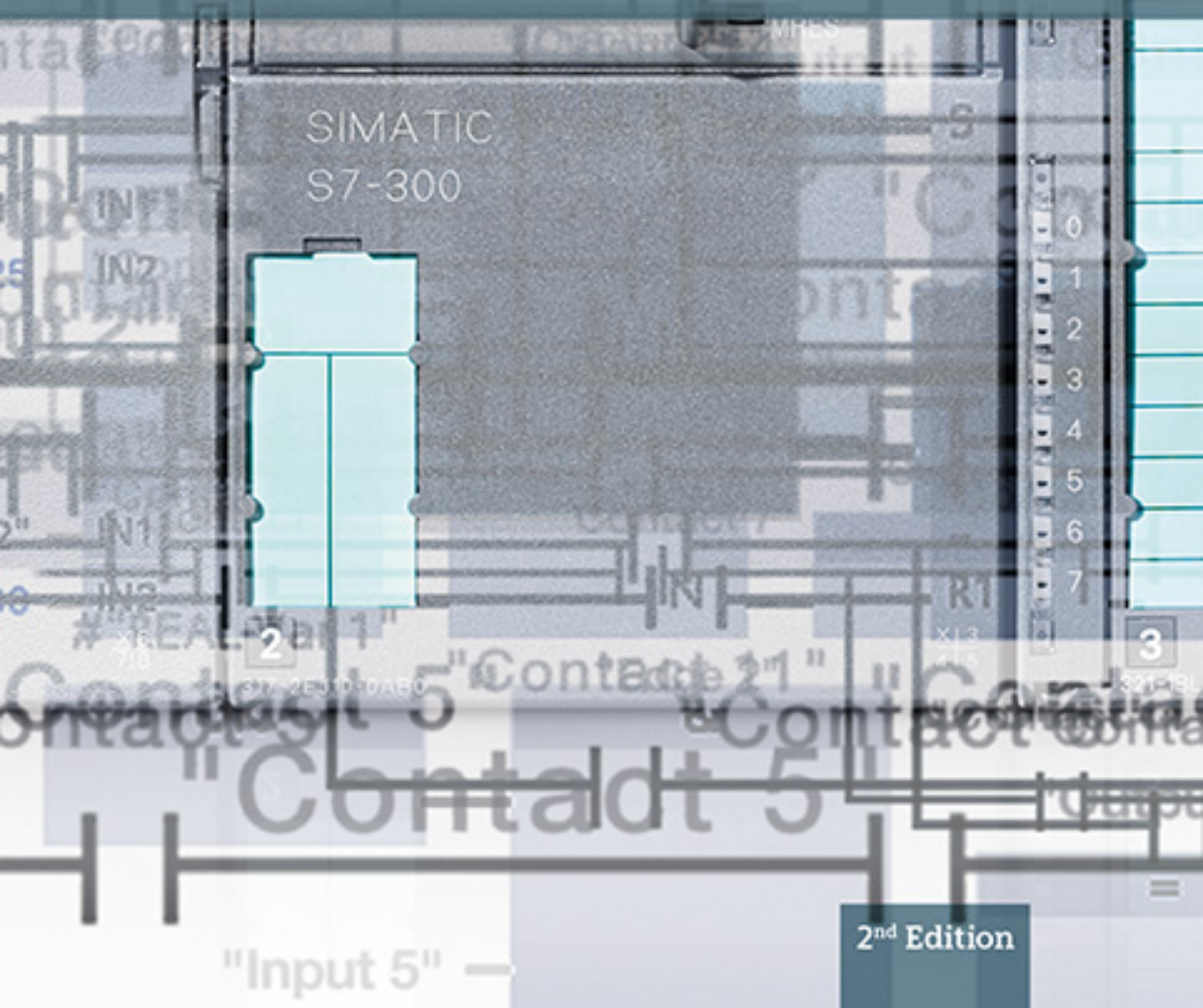


Hans Berger

Automating with SIMATIC S7-300 inside TIA Portal

Configuring, Programming and Testing
with STEP 7 Professional



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>.

The author, translators, and publisher have taken great care with all texts and illustrations in this book. Nevertheless, errors can never be completely avoided. The author, translators, and publisher accept no liability, for whatever legal reasons, for any damage resulting from the use of the programming examples.
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

© 2014 by Publicis Erlangen, Zweigniederlassung der PWW GmbH

The publication and all parts thereof are protected by copyright.

Any use of it outside the strict provisions of the copyright law without the consent of the publisher is forbidden and will incur penalties. This applies particularly to reproduction, translation, microfilming, or other processing, and to storage or processing in electronic systems. It also applies to the use of individual figures and extracts from the text.

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 multi-point 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 oper- and 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	22
1.1 Overview of the S7-300 automation system	22
1.1.1 SIMATIC S7-300 programmable controller	23
1.1.2 Overview of STEP 7 Professional V12	24
1.1.3 Five programming languages	26
1.1.4 Execution of the user program	28
1.1.5 Data management in the SIMATIC automation system	30
1.2 Introduction to STEP 7 Professional V12	31
1.2.1 Installing STEP 7	31
1.2.2 Automation License Manager	31
1.2.3 Starting STEP 7 Professional	32
1.2.4 Portal view	32
1.2.5 The windows of the Project view	33
1.2.6 Help information system	36
1.2.7 Adapting the user interface	36
1.3 Editing a SIMATIC project	37
1.3.1 Structured representation of project data	38
1.3.2 Project data and editors for a PLC station	39
1.3.3 Creating and editing a project	42
1.3.4 Working with reference projects	44
1.3.5 Creating and editing libraries	45
2 SIMATIC S7-300 automation system	46
2.1 S7-300 station components	46
2.2 S7-300 CPUs	48
2.2.1 CPU versions	48
2.2.2 Control and display elements	50
2.2.3 SIMATIC Micro Memory Card	51
2.2.4 Memory areas in an S7-300 station	51
2.2.5 Bus interfaces	53
2.3 Signal modules	55
2.3.1 Digital input modules	55
2.3.2 Digital output modules	56
2.3.3 Digital input/output modules	56
2.3.4 Analog input modules	57
2.3.5 Analog output modules	57
2.3.6 Analog input/output modules	58
2.4 Function modules	59
2.5 Communication modules	60

2.6 Other modules	61
2.6.1 Interface modules (IM)	61
2.6.2 Power supply modules (PS)	62
2.6.3 Simulator module	62
2.6.4 Dummy module	62
2.7 SIPLUS S7-300	63
3 Device configuration	65
3.1 Introduction	65
3.2 Configuring a station	68
3.2.1 Adding a PLC station	68
3.2.2 Adding a module	68
3.2.3 Adding an expansion rack	69
3.3 Parameterization of modules	70
3.3.1 Parameterization of CPU properties	70
3.3.2 Addressing modules	73
3.3.3 Assigning parameters to signal modules	75
3.4 Configuring the network	76
3.4.1 Introduction, overview	76
3.4.2 Networking stations	77
3.4.3 Node addresses in a subnet	79
3.4.4 Connections	80
3.4.5 Configuring an MPI subnet	82
3.4.6 Configuring a PROFIBUS subnet	83
3.4.7 Configuring a PROFINET subnet	85
3.4.8 Configuring an AS-i subnet	89
4 Tags, addressing, and data types	90
4.1 Operands and tags	90
4.1.1 Introduction, overview	90
4.1.2 Operand areas: inputs and outputs	91
4.1.3 Operand area: bit memory	93
4.1.4 Operand area: data	94
4.1.5 Operand area: temporary local data	95
4.2 Addressing of operands and tags	96
4.2.1 Signal path	96
4.2.2 Absolute addressing of tags	97
4.2.3 Symbolic addressing of tags	101
4.2.4 Addressing constants	102
4.3 Indirect addressing	103
4.3.1 Memory-indirect addressing with STL	104
4.3.2 Register-indirect addressing with STL	107
4.3.3 Working with the address registers with STL	109
4.3.4 Direct access to complex local tags with STL	116
4.3.5 Indirect addressing with SCL	118

4.4 Elementary data types	120
4.4.1 Introduction	120
4.4.2 Bit-serial data types BOOL, BYTE, WORD, and DWORD	123
4.4.3 BCD numbers BCD16 and BCD32	123
4.4.4 Fixed-point data types with sign INT and DINT	123
4.4.5 Floating-point data type REAL	125
4.4.6 Data type CHAR	126
4.4.7 Data types for durations and points in time	127
4.5 Complex data types	128
4.5.1 Data type DATE_AND_TIME	128
4.5.2 Data type STRING	129
4.5.3 Data type ARRAY	131
4.5.4 Data type STRUCT	133
4.6 Parameter types and pointers	135
4.6.1 Parameter types	135
4.6.2 Pointer	136
4.6.3 “Variable” ANY pointer with STL	139
4.6.4 “Variable” ANY pointer with SCL	140
4.7 PLC data types	140
4.8 Start information	143
5 Program execution	145
5.1 Operating states of the CPU	145
5.1.1 STOP operating state	146
5.1.2 STARTUP operating state	147
5.1.3 RUN operating state	149
5.1.4 HOLD operating state	149
5.1.5 Reset CPU memory	150
5.1.6 Restoring the factory settings	150
5.1.7 Retentive behavior of operands	150
5.2 Creating a user program	151
5.2.1 Program draft	151
5.2.2 Program execution	155
5.2.3 Block types	156
5.2.4 Editing block properties	158
5.2.5 Block interface	161
5.2.6 Example of use of block parameters	163
5.3 Calling blocks	165
5.3.1 General information on calling of code blocks	165
5.3.2 Calling functions (FC)	165
5.3.3 Calling function blocks (FB)	167
5.3.4 “Passing on” of block parameters	170
5.4 Startup program	171
5.4.1 Organization block OB 100	171
5.4.2 Determining a module address	171
5.4.3 Parameterization of modules	173

5.5 Main program	176
5.5.1 Organization block OB 1	176
5.5.2 Process image updating	177
5.5.3 Cycle time and response time	178
5.5.4 Hold, stop, and protect program	181
5.5.5 Time	182
5.5.6 Read system time	184
5.5.7 Runtime meter	184
5.6 Interrupt processing	186
5.6.1 Introduction to interrupt processing	186
5.6.2 Priority classes	187
5.6.3 Time-of-day interrupt, organization block OB 10	188
5.6.4 Time-delay interrupts, organization blocks OB 20 and OB 21	191
5.6.5 Cyclic interrupts, organization blocks OB 32 to OB 35	193
5.6.6 Hardware interrupt, organization block OB 40	195
5.6.7 Interrupts for DPV1 organization blocks OB 55 to OB 57	196
5.6.8 Isochronous mode interrupt, organization block OB 61	197
5.6.9 Reading additional interrupt information	199
5.7 Error handling	200
5.7.1 Causes of errors and error responses	200
5.7.2 Synchronous error	201
5.7.3 Enabling and disabling synchronous error processing	202
5.7.4 Enter substitute value	205
5.7.5 Asynchronous errors	206
5.7.6 Disable, delay, and enable interrupts and asynchronous errors	209
5.8 Diagnostics	211
5.8.1 Diagnostic error interrupt, organization block OB 82	211
5.8.2 Read system state list	212
5.8.3 Read start information	214
5.8.4 Write user diagnostic event to the diagnostic buffer	215
5.8.5 System diagnostics with Report System Errors	216
6 Program editor	218
6.1 Introduction	218
6.2 PLC tag table	218
6.2.1 Working with PLC tag tables	219
6.2.2 Defining and processing PLC tags	220
6.2.3 Comparing PLC tags	221
6.2.4 Exporting and importing a PLC tag table	222
6.2.5 Constants tables	223
6.3 Programming a code block	223
6.3.1 Creating a new code block	223
6.3.2 Working area of the program editor for code blocks	224
6.3.3 Specifying code block properties	226
6.3.4 Programming a block interface	226
6.3.5 Programming a control function	228

6.3.6 Editing tags	232
6.3.7 Working with program comments	234
6.4 Programming a data block	236
6.4.1 Creating a new data block	236
6.4.2 Working area of program editor for data blocks	236
6.4.3 Defining properties for data blocks	237
6.4.4 Declaring data tags	238
6.4.5 Entering data tags in global data blocks	239
6.5 Compiling blocks	239
6.5.1 Starting the compilation	239
6.5.2 Compiling SCL blocks	241
6.5.3 Eliminating errors following compilation	241
6.6 Program information	242
6.6.1 Cross-reference list	242
6.6.2 Assignment list	244
6.6.3 Call structure	245
6.6.4 Dependency structure	246
6.6.5 Consistency check	247
6.6.6 Memory utilization of the CPU	248
7 Ladder logic LAD	249
7.1 Introduction	249
7.1.1 Programming with LAD in general	249
7.1.2 Program elements of ladder logic	251
7.2 Programming binary logic operations with LAD	252
7.2.1 NO and NC contacts	252
7.2.2 Series and parallel connection of contacts	253
7.2.3 T branch, open parallel branch	254
7.2.4 Negating result of logic operation	255
7.2.5 Edge evaluation of a binary tag	255
7.2.6 Comparison contacts	256
7.3 Programming memory functions with LAD	257
7.3.1 Simple coil, assignment	257
7.3.2 Set and reset coils	258
7.3.3 Retentive response due to latching	259
7.3.4 Coils with time response	260
7.3.5 Coils with counter response	260
7.4 Programming Q boxes with LAD	261
7.4.1 Memory boxes	261
7.4.2 Edge evaluation of current flow	262
7.4.3 SIMATIC timer functions	263
7.4.4 SIMATIC counter functions	264
7.4.5 IEC timer functions	265
7.4.6 IEC counter functions	266
7.5 Programming EN/ENO boxes with LAD	267
7.5.1 Transfer function, MOVE	268

7.5.2 Arithmetic functions	269
7.5.3 Math functions	269
7.5.4 Conversion functions	270
7.5.5 Shift functions	272
7.5.6 Word logic operations	272
7.6 Controlling the program flow with LAD	274
7.6.1 Working with status bits in the ladder logic	274
7.6.2 EN/ENO mechanism with LAD	276
7.6.3 Jump functions	277
7.6.4 Block functions	278
7.6.5 Master Control Relay (MCR)	280
8 Function block diagram FBD	282
8.1 Introduction	282
8.1.1 Programming with FBD in general	282
8.1.2 Program elements of the function block diagram	284
8.2 Programming binary logic operations with FBD	285
8.2.1 Scanning for signal states “1” and “0”	285
8.2.2 Programming a binary logic operation in the function block diagram	286
8.2.3 AND function	287
8.2.4 OR function	287
8.2.5 Exclusive OR function	288
8.2.6 Combined binary logic operations, negating result of logic operation	288
8.2.7 T branch	289
8.2.8 Edge evaluation of binary tags	289
8.2.9 Comparison functions	290
8.3 Programming standard boxes with FBD	291
8.3.1 Assign box	291
8.3.2 Set and reset boxes	292
8.3.3 Standard boxes with time response	293
8.3.4 Standard boxes with counter response	294
8.4 Programming Q boxes with FBD	294
8.4.1 Memory boxes	295
8.4.2 Edge evaluation of result of logic operation	296
8.4.3 SIMATIC timer functions	297
8.4.4 SIMATIC counter functions	297
8.4.5 IEC timer functions	298
8.4.6 IEC counter functions	299
8.5 Programming EN/ENO boxes with FBD	300
8.5.1 Transfer function MOVE	301
8.5.2 Arithmetic functions	302
8.5.3 Math functions	303
8.5.4 Conversion functions	303
8.5.5 Shift functions	305
8.5.6 Word logic operations	306

8.6 Controlling the program flow with FBD	307
8.6.1 Working with status bits in the function block diagram	308
8.6.2 EN/ENO mechanism with FBD	309
8.6.3 Jump functions	310
8.6.4 Block functions	312
8.6.5 Master Control Relay (MCR)	313
9 Statement list STL	315
9.1 Introduction	315
9.1.1 Programming with STL in general	315
9.1.2 Structure of an STL statement	316
9.2 Programming binary logic operations with STL	317
9.2.1 Processing of a binary logic operation, operation step	317
9.2.2 Scanning for signal states “1” and “0”	319
9.2.3 Programming a binary logic operation in the statement list	320
9.2.4 AND function	321
9.2.5 OR function	321
9.2.6 Exclusive OR function	321
9.2.7 Combined binary logic operations	322
9.2.8 Control of result of logic operation	324
9.3 Programming memory functions with STL	325
9.3.1 Assignment	326
9.3.2 Setting and resetting	326
9.3.3 Edge evaluation	327
9.4 Programming timer and counter functions with STL	328
9.4.1 SIMATIC timer functions	328
9.4.2 SIMATIC counter functions	330
9.4.3 IEC timer functions	331
9.4.4 IEC counter functions	332
9.5 Programming digital functions with STL	333
9.5.1 Transfer functions	333
9.5.2 Comparison functions	333
9.5.3 Arithmetic functions	337
9.5.4 Math functions	340
9.5.5 Conversion functions	341
9.5.6 Shift functions	342
9.5.7 Word logic operations	345
9.6 Controlling the program flow with STL	347
9.6.1 Working with status bits in the statement list	347
9.6.2 EN/ENO mechanism with STL	349
9.6.3 Jump functions	351
9.6.4 Jump list	352
9.6.5 Loop jump	353
9.6.6 Block functions	354
9.6.7 Master Control Relay (MCR)	356

9.7 Further STL functions	358
9.7.1 Accumulator functions	358
9.7.2 Adding of constants to accumulator 1	360
9.7.3 Decrementing, incrementing	361
9.7.4 Null instructions	361
10 Structured Control Language SCL	363
10.1 Introduction to programming with SCL	363
10.1.1 Programming with SCL in general	363
10.1.2 SCL statements and operators	365
10.2 Programming binary logic operations with SCL	367
10.2.1 Scanning for signal states “1” and “0”	367
10.2.2 AND function	368
10.2.3 OR function	369
10.2.4 Exclusive OR function	369
10.2.5 Combined binary logic operations	369
10.2.6 Negating result of logic operation	370
10.3 Programming memory functions with SCL	370
10.3.1 Value assignment of a binary tag	371
10.3.2 Setting and resetting	371
10.3.3 Edge evaluation	371
10.4 Programming timer and counter functions with SCL	372
10.4.1 SIMATIC timer functions	372
10.4.2 SIMATIC counter functions	373
10.4.3 IEC timer functions	374
10.4.4 IEC counter functions	375
10.5 Programming digital functions with SCL	375
10.5.1 Transfer function, value assignment of a digital tag	376
10.5.2 Comparison functions	376
10.5.3 Arithmetic functions	377
10.5.4 Math functions	378
10.5.5 Conversion functions	379
10.5.6 Shift functions	380
10.5.7 Word logic operations, logic expression	381
10.6 Controlling the program flow with SCL	382
10.6.1 Working with the ENO tag	382
10.6.2 EN/ENO mechanism with SCL	383
10.6.3 Control statements	385
10.6.4 Block functions	394
11 S7-GRAPH sequential control	397
11.1 Introduction	397
11.1.1 What is a sequential control?	397
11.1.2 Properties of a sequential control	398
11.1.3 Program for a sequential control, quantity framework	399
11.1.4 Operating modes	399

11.1.5	Procedure for configuration	400
11.2	Elements of a sequential control	400
11.2.1	Steps and transitions	400
11.2.2	Jumps in a sequential control	402
11.2.3	Branching of a sequencer	402
11.2.4	GRAPH-specific tags	403
11.2.5	Permanent instructions	404
11.2.6	Step and transition functions	405
11.2.7	Processing of actions	408
11.3	Configuring a sequential control	414
11.3.1	Programming the GRAPH function block	414
11.3.2	Configuring the sequencer structure	415
11.3.3	Programming steps and transitions	417
11.3.4	Programming permanent instructions	418
11.3.5	Configuring block-independent alarms	419
11.3.6	Attributes of the GRAPH function block	419
11.3.7	Using the GRAPH function block	420
11.4	Testing the sequential control	422
11.4.1	Loading the GRAPH function block	422
11.4.2	Settings for program testing	422
11.4.3	Using operating modes	423
11.4.4	Synchronization of a sequencer	424
11.4.5	Testing with program status	425
12	Basic functions	427
12.1	Binary logic operations	427
12.1.1	Introduction	427
12.1.2	Working with binary signals	428
12.1.3	AND function, series connection	431
12.1.4	OR function, parallel connection	432
12.1.5	Exclusive OR function, non-equivalence function	432
12.1.6	Negate result of logic operation, NOT contact	433
12.2	Memory functions	435
12.2.1	Introduction	435
12.2.2	Standard coil, assignment	435
12.2.3	Single setting and resetting	436
12.2.4	Dominant setting and resetting, memory function	437
12.2.5	Edge evaluation	438
12.3	SIMATIC timer functions	443
12.3.1	Overview	443
12.3.2	Programming a timer function	444
12.3.3	Timer response as pulse	449
12.3.4	Timer response as extended pulse	451
12.3.5	Timer response as ON delay	453
12.3.6	Timer response as retentive ON delay	455
12.3.7	Timer response as OFF delay	457

12.4 IEC timer functions	459
12.4.1 Introduction	459
12.4.2 Pulse generation TP	459
12.4.3 ON delay TON	460
12.4.4 OFF delay TOF	461
12.5 SIMATIC counter functions	462
12.5.1 Overview	462
12.5.2 Programming a counter function	463
12.5.3 Principle of operation of a counter function	467
12.5.4 Enabling a counter function with STL	468
12.6 IEC counter functions	470
12.6.1 Introduction	470
12.6.2 Up counter CTU	470
12.6.3 Down counter CTD	471
12.6.4 Up/down counter CTUD	472
13 Digital functions	475
13.1 General information	475
13.2 Transfer functions	476
13.2.1 General information on the “simple” transfer function	476
13.2.2 MOVE box with LAD and FBD	476
13.2.3 Loading and transferring with STL	478
13.2.4 Value assignments with SCL	479
13.2.5 Copying and filling a data area in the work memory	481
13.2.6 Transfer data area from and to load memory	483
13.2.7 Control memory area with MCR dependency	485
13.3 Comparison functions	487
13.3.1 Execution of “simple” comparison function	488
13.3.2 Comparison function T_COMP	488
13.3.3 Comparison function S_COMP	490
13.4 Arithmetic functions	491
13.4.1 General function description	491
13.4.2 Data types and status bits for an arithmetic function	493
13.4.3 Execution of the arithmetic function	494
13.4.4 Arithmetic functions for date and time	495
13.5 Math functions	496
13.5.1 General function description	496
13.5.2 General execution of a math function	497
13.5.3 Trigonometric functions SIN, COS, TAN	498
13.5.4 Arc functions ASIN, ACOS, ATAN	499
13.5.5 Additional math functions	499
13.6 Conversion functions	500
13.6.1 Implicit data type conversion	501
13.6.2 Data type conversion of fixed-point numbers	501
13.6.3 Data type conversion of floating-point numbers	505
13.6.4 Data type conversion for date/time with T_CONV	507

13.6.5	Data type conversion for data type STRING with S_CONV	509
13.6.6	Data type conversion of hexadecimal numbers	510
13.6.7	Scaling and unscaling	511
13.6.8	Further conversion functions	513
13.7	Shift functions	514
13.7.1	General function description	514
13.7.2	General execution of a shift function	514
13.7.3	Shift to right	516
13.7.4	Shift to left	517
13.7.5	Rotate to right	518
13.7.6	Rotate to left	518
13.7.7	Rotating by the condition code bit CC1 (STL)	519
13.8	Logic functions	519
13.8.1	Word logic operations	519
13.8.2	Invert	522
13.8.3	Code bit and set bit number	523
13.8.4	Selection and limiting functions	524
13.9	Functions for strings	526
14	Program flow control	530
14.1	Status bits	531
14.1.1	Description of the status bits	531
14.1.2	Controlling the status bits	533
14.1.3	Setting and resetting the result of logic operation	534
14.1.4	Controlling the binary result	535
14.1.5	Evaluating the status bits	538
14.2	Jump functions	539
14.2.1	Introduction	539
14.2.2	Absolute jump	539
14.2.3	Conditional jump functions	541
14.2.4	Jump functions depending on status bits	542
14.3	Block end functions	545
14.3.1	Block end function RET (LAD and FBD)	545
14.3.2	Block end functions BEC, BEU, and BE (STL)	546
14.3.3	RETURN statement (SCL)	546
14.4	Calling of code blocks	547
14.4.1	General information on block calls	547
14.4.2	Calling a function (FC)	547
14.4.3	Calling a function block (FB)	549
14.4.4	Change to a block without block parameter	551
14.5	Data block functions	553
14.5.1	Open data block	555
14.5.2	Additional data block functions with STL	555
14.5.3	Creating, deleting, and testing data blocks	556
14.6	Master control relay	560
14.6.1	Introduction	560

14.6.2 MCR dependency	560
14.6.3 MCR area and MCR zone	560
14.6.4 MCR area and MCR zone with a block change	563
14.6.5 Instructions for the master control relay	563
15 Online operation and program test	564
15.1 Connection of a programming device to the PLC station	565
15.1.1 Settings on the programming device	565
15.1.2 Connecting the programming device to the PLC station	566
15.1.3 Switching on online mode	566
15.2 Transferring project data	568
15.2.1 Loading project data for the first time	568
15.2.2 Reloading the project data	570
15.2.3 Protection of the user program	571
15.2.4 Editing of online project without offline project	572
15.2.5 Working with the Micro Memory Card	573
15.3 Working with blocks in online mode	574
15.3.1 Introduction	574
15.3.2 Editing the online version of a block	575
15.3.3 Downloading a block to the CPU	575
15.3.4 Packing the work memory	577
15.3.5 Uploading blocks from the CPU	577
15.3.6 Working with setpoints	579
15.3.7 Comparing blocks	581
15.4 Hardware diagnostics	583
15.4.1 Status displays on the modules	583
15.4.2 Diagnostic information	584
15.4.3 Diagnostic buffer	585
15.4.4 Diagnostic functions	586
15.4.5 Online tools	586
15.4.6 Further diagnostic information via the programming device	587
15.5 Testing the user program	588
15.5.1 Defining the call environment	589
15.5.2 Testing with program status	589
15.5.3 Testing in single step mode	593
15.5.4 Monitoring of PLC tags	596
15.5.5 Monitoring of data tags	596
15.5.6 Testing with watch tables	597
15.5.7 Monitoring and modifying in the STOP operating state	602
15.5.8 Testing with the force table	603
16 Distributed I/O	607
16.1 Introduction, overview	607
16.2 ET 200 distributed I/O system	608
16.2.1 ET 200M	608
16.2.2 ET 200MP	609

16.2.3 ET 200S	609
16.2.4 ET 200SP	610
16.2.5 ET 200iSP	611
16.2.6 ET 200pro	611
16.2.7 ET 200eco and ET 200eco PN	612
16.3 PROFINET IO	613
16.3.1 PROFINET IO components	613
16.3.2 Addresses with PROFINET IO	615
16.3.3 Special PROFINET configurations	618
16.3.4 Configuring PROFINET IO	619
16.3.5 Coupling modules for PROFINET IO	623
16.3.6 Real-time communication in PROFINET	624
16.4 PROFIBUS DP	628
16.4.1 PROFIBUS DP components	628
16.4.2 Addresses with PROFIBUS DP	632
16.4.3 Configuring PROFIBUS DP	635
16.4.4 Coupling modules for PROFIBUS DP	638
16.4.5 Special functions for PROFIBUS DP	640
16.5 Isochronous mode	641
16.5.1 Introduction	641
16.5.2 Isochronous mode with PROFINET IO	641
16.5.3 Isochronous mode with PROFIBUS	645
16.6 System blocks for distributed I/O	648
16.6.1 System blocks for PROFIBUS DP	648
16.6.2 System blocks for PROFIBUS DP and PROFINET IO	652
16.6.3 System blocks for PROFINET IO	655
16.7 Actuator/sensor interface	657
16.7.1 Components of actuator/sensor interface	657
16.7.2 Addresses on the actuator/sensor interface	660
16.7.3 Configuring the actuator/sensor interface with CP 343-2P	660
16.7.4 System functions for AS-i	661
17 Communication	663
17.1 Overview	663
17.2 S7 basic communication	664
17.2.1 Basics of station-internal S7 basic communication	664
17.2.2 Configuring of station-internal S7 basic communication	665
17.2.3 System blocks for station-internal S7 basic communication	665
17.2.4 Basics of station-external S7 basic communication	667
17.2.5 Configuring of station-external S7 basic communication	668
17.2.6 System blocks for station-external S7 basic communication	668
17.3 S7 communication	671
17.3.1 Basics	671
17.3.2 Configuring S7 communication	671
17.3.3 One-way data exchange	674

17.3.4 Two-way data exchange	675
17.3.5 Monitoring functions	678
17.4 Open user communication	678
17.4.1 Basics	678
17.4.2 Establishing and clearing connections	680
17.4.3 Data transfer with TCP native or ISO-on-TCP	682
17.4.4 Data transfer with UDP	685
18 Appendix	687
18.1 Working with source files	687
18.1.1 General procedure	687
18.1.2 Programming a code block in the source file	688
18.1.3 Programming a data block in the source file	692
18.1.4 Programming a PLC data type in the source file	695
18.2 Migrating projects	696
18.3 Simulation with the TIA Portal	700
18.3.1 Differences from a real CPU	700
18.3.2 Starting and saving the simulation	701
18.3.3 Using the simulation	702
18.3.4 Testing the program with the simulation	705
18.3.5 Additional functions of PLCSIM	707
18.4 Web server	707
18.4.1 Enable Web server	707
18.4.2 Reading out Web information	708
18.4.3 Standard Web pages	709
18.5 Storage of local tags	712
18.5.1 Storage in global data blocks	712
18.5.2 Storage in instance data blocks	713
18.5.3 Storage in the temporary local data	713
18.5.4 Data storage of the block parameters of a function (FC)	715
18.5.5 Data storage of the block parameters of a function block (FB)	717
18.5.6 Data storage of a local instance in a multi-instance	718
Index	721

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.

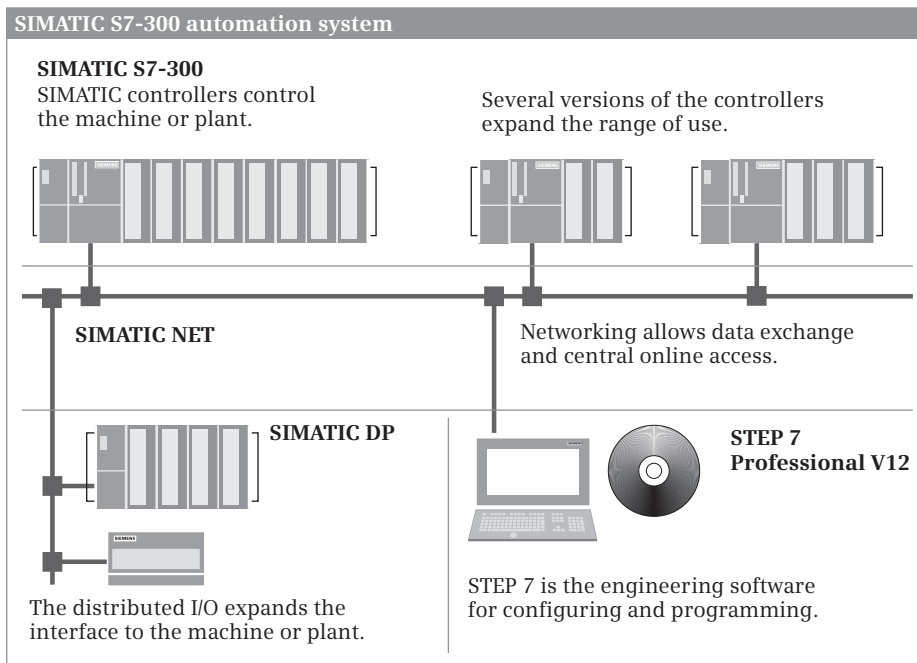


Fig. 1.1 Components of the SIMATIC S7-300 automation system

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.

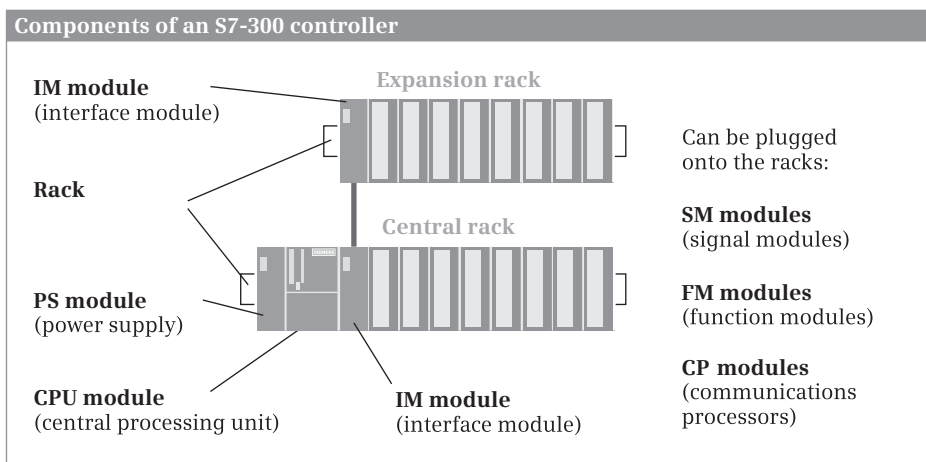


Fig. 1.2 Components of an S7-300 controller

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):

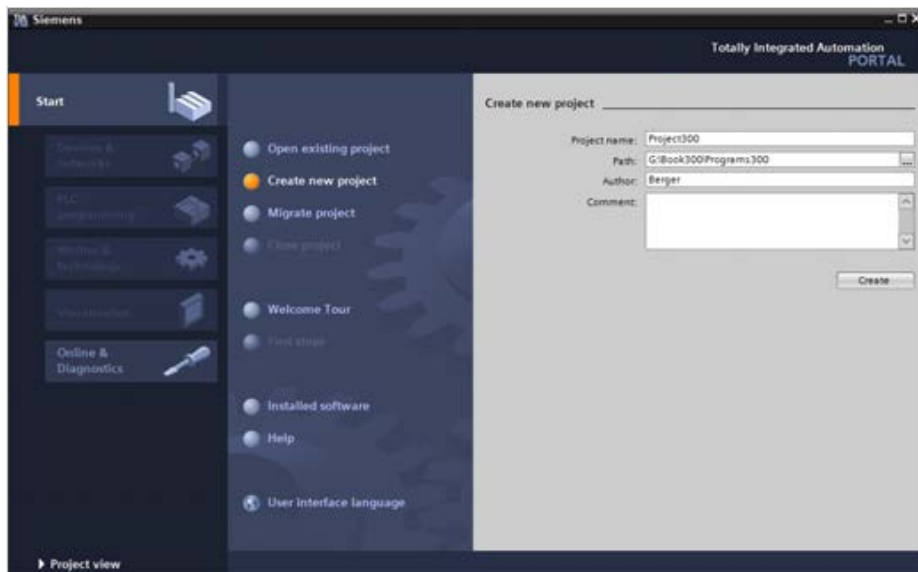


Fig. 1.3 Tools in the Start portal of STEP 7 Professional V12

- ▷ 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.

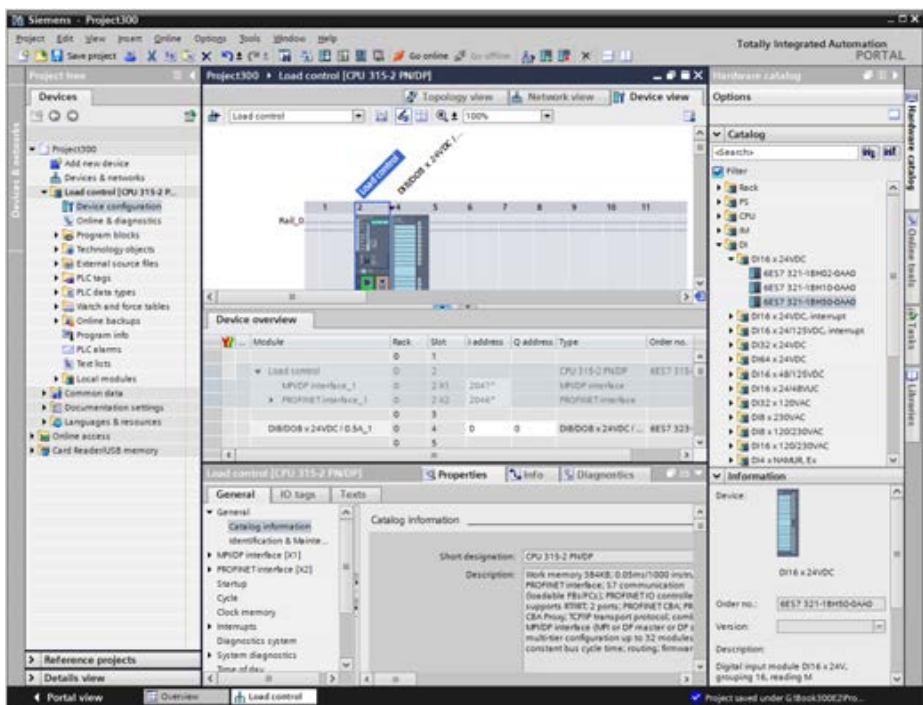


Fig. 1.4 Example of a Project view: working area of the device configuration

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.

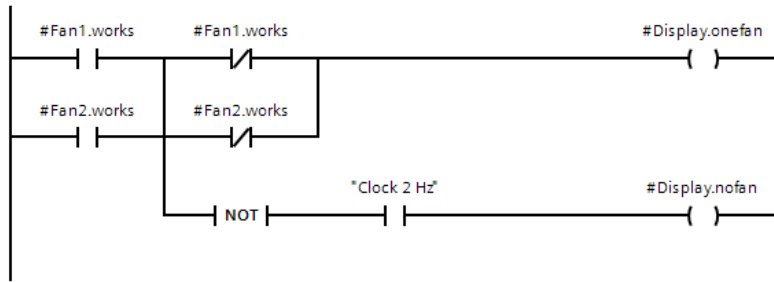


Fig. 1.5 Example of representation in ladder logic

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.

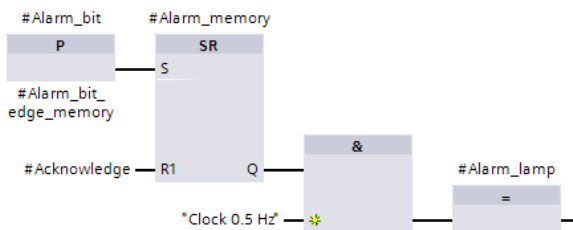


Fig. 1.6 Example of representation in function block diagram

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

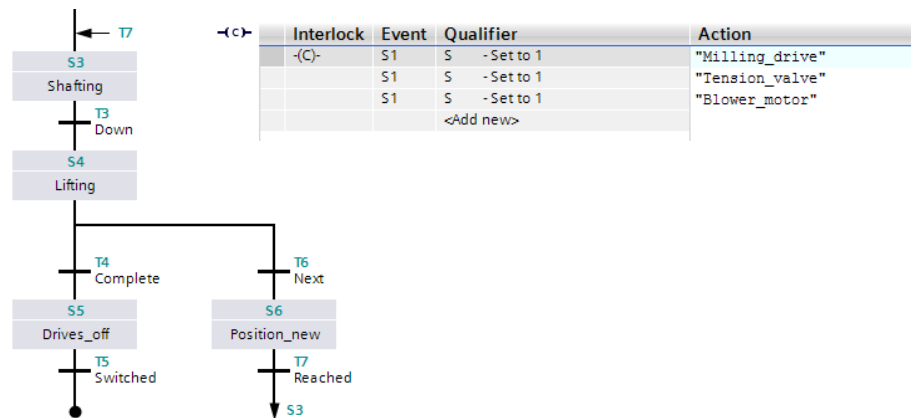


Fig. 1.9 Example of a GRAPH sequencer and step configuration

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).

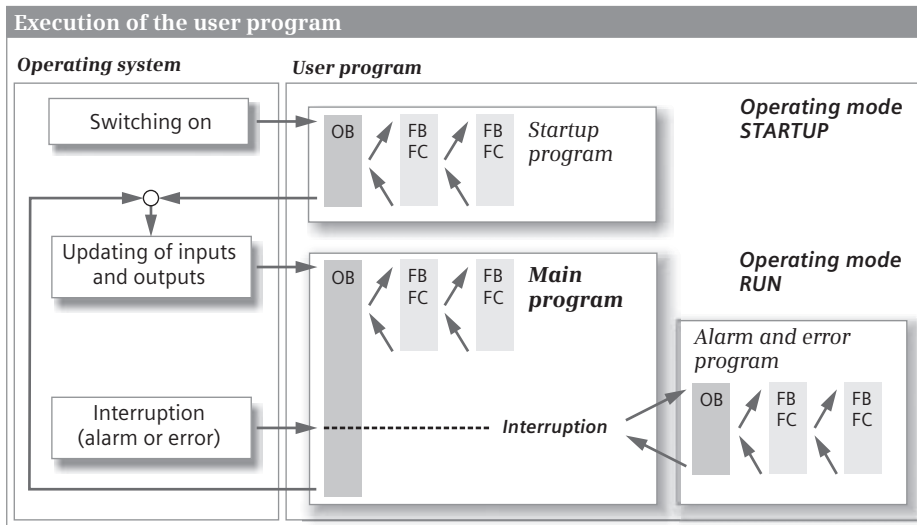


Fig. 1.10 Execution of the user program

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).

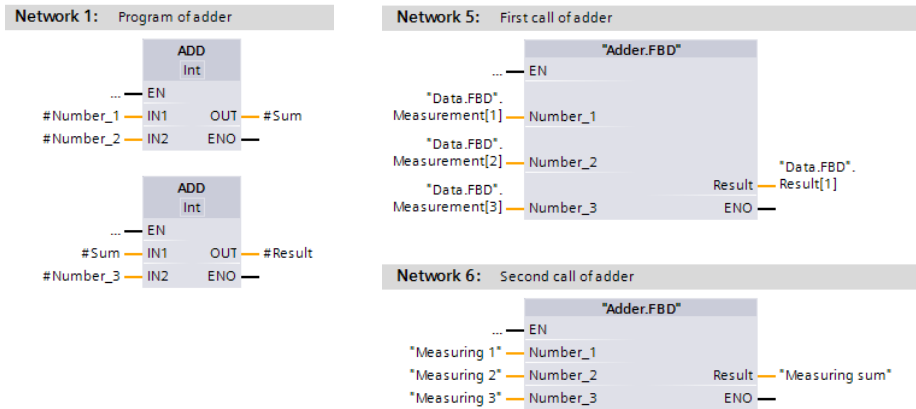


Fig. 1.11 Example of two block calls with different tags in each case

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