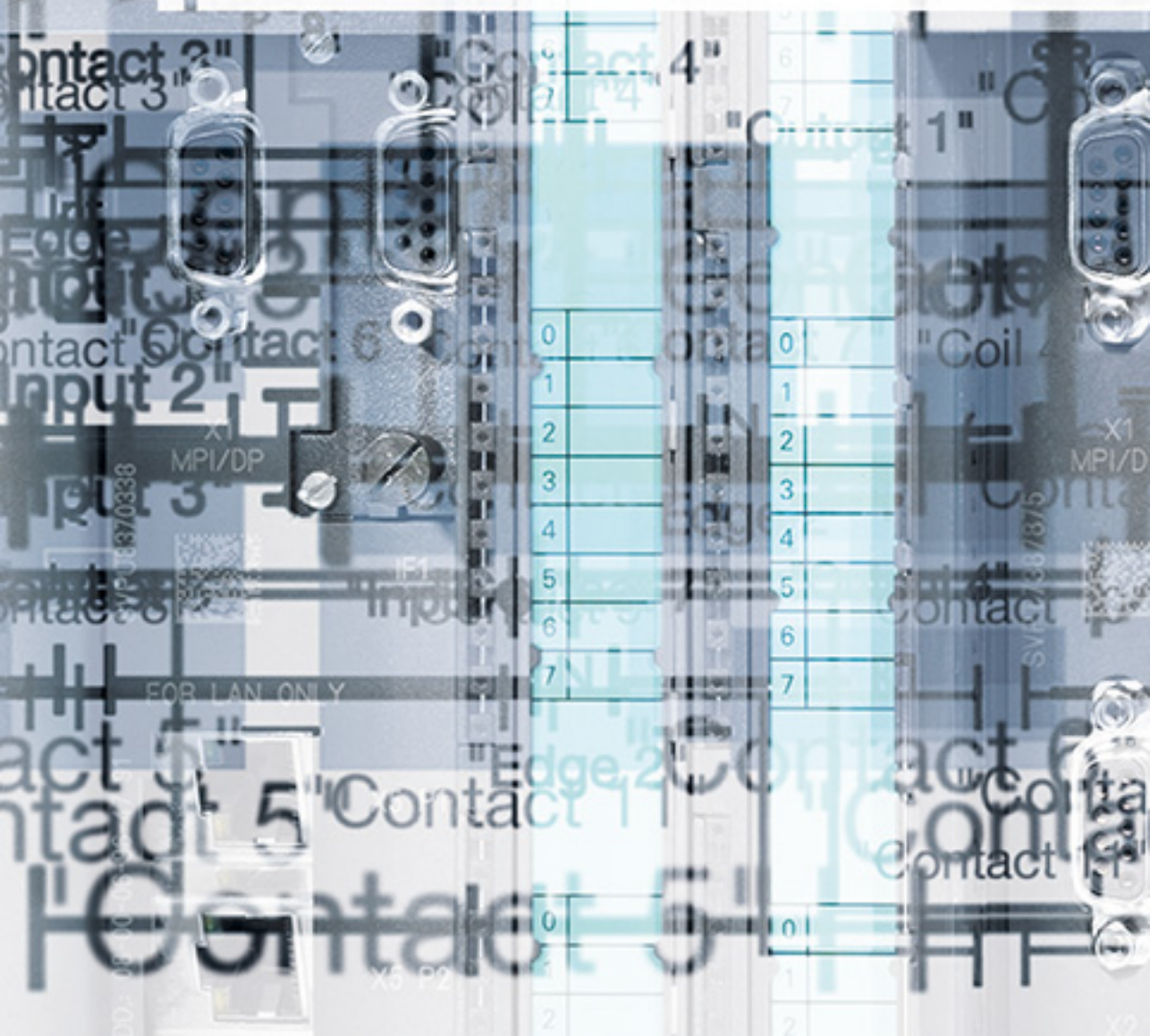


Hans Berger

Automating with SIMATIC S7-400 inside TIA Portal

Configuring, Programming and Testing
with STEP 7 Professional

SIEMENS



Berger
Automating with SIMATIC S7-400 inside TIA Portal

Automating with SIMATIC S7-400 inside TIA Portal

Configuring, Programming and Testing
with STEP 7 Professional

by Hans Berger

Publicis Publishing

Bibliographic information from the Deutsche Nationalbibliothek

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 publisher, author, and translators accept no liability, for whatever legal reasons, for any damage resulting from the use of the programming examples.

www.publicis-books.de

ISBN 978-3-89578-383-8

Editor: Siemens Aktiengesellschaft, Berlin and Munich

Publisher: Publicis Publishing, Erlangen

© 2013 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 extracts from the text.

Printed in Germany

Preface

The SIMATIC automation system unites all 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 S7-400 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 makes it possible to use the complete functionality of the S7-400 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 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 S7-400 automation system with the STEP 7 Professional engineering software Version 11 with Service Pack 4.

Erlangen, June 2013

Hans Berger

The contents of the book at a glance

Start

Overview of the SIMATIC S7-400 automation system.
Introduction to the SIMATIC STEP 7 Professional V11 engineering software.
The basis of the automation solution: Creating and editing a project.

SIMATIC S7-400 automation system

Overview of SIMATIC S7-400 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 module 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 testing

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.

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.

How PtP communication is implemented.

Annex

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	21
1.1 Overview of the S7-400 automation system	21
1.1.1 SIMATIC S7-400 programmable controller	22
1.1.2 Overview of STEP 7 Professional V11	23
1.1.3 Five programming languages	25
1.1.4 Execution of the user program	27
1.1.5 Data management in the SIMATIC automation system	30
1.2 Introduction to STEP 7 Professional V11	30
1.2.1 Installing STEP 7	30
1.2.2 Automation License Manager	31
1.2.3 Starting STEP 7 Professional	31
1.2.4 Portal view	31
1.2.5 Help information system	33
1.2.6 The windows of the Project view	33
1.2.7 Adapting the user interface	36
1.3 Editing a SIMATIC project	36
1.3.1 Structured representation of project data	37
1.3.2 Project data and editors for a PLC station	37
1.3.3 Creating and editing a project	41
1.3.4 Creating and editing libraries	43
2 SIMATIC S7-400 automation system	44
2.1 Components of an S7-400 station	44
2.2 S7-400 CPUs	48
2.2.1 CPU versions	48
2.2.2 Control and display elements	50
2.2.3 SIMATIC memory card	51
2.2.4 Memory areas in an S7-400 station	51
2.2.5 Bus interfaces	53
2.2.6 IF 964-DP interface module	54
2.3 Signal modules	54
2.3.1 Digital input modules	54
2.3.2 Digital output modules	55
2.3.3 Analog input modules	56
2.3.4 Analog output module	57
2.4 Function modules	57
2.5 Communication modules	58
2.6 Other modules	59
2.6.1 Interface modules	59
2.6.2 Power supply modules	60
2.7 SIPLUS S7-400	61

3 Device configuration	62
3.1 Introduction	62
3.2 Configuring a station	65
3.2.1 Adding a PLC station	65
3.2.2 Adding a module	65
3.2.3 Adding an expansion unit	66
3.3 Parameterization of modules	67
3.3.1 Parameterization of CPU properties	67
3.3.2 Addressing modules	70
3.3.3 Assigning parameters to signal modules	73
3.4 Configuring the network	74
3.4.1 Introduction, overview	74
3.4.2 Networking stations	75
3.4.3 Node addresses in a subnet	76
3.4.4 Connections	77
3.4.5 Configuring an MPI subnet	80
3.4.6 Configuring a PROFIBUS subnet	81
3.4.7 Configuring a PROFINET subnet	82
3.4.8 Configuring a PtP subnet	86
 4 Tags, addressing, and data types	 89
4.1 Operands and tags	89
4.1.1 Introduction, overview	89
4.1.2 Operand areas: inputs and outputs	90
4.1.3 Operand area: bit memory	92
4.1.4 Operand area: data	93
4.1.5 Operand area temporary local data	94
4.2 Addressing of operands and tags	95
4.2.1 Signal path	95
4.2.2 Absolute addressing of tags	96
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	106
4.3.3 Working with the address registers with STL	108
4.3.4 Direct access to complex local tags with STL	115
4.3.5 Indirect addressing with SCL	117
4.4 Elementary data types	119
4.4.1 Introduction	119
4.4.2 Bit-serial data types BOOL, BYTE, WORD, and DWORD	122
4.4.3 BCD numbers BCD16 and BCD32	122
4.4.4 Fixed-point data types with sign INT and DINT	122
4.4.5 Floating-point data type REAL	124
4.4.6 Data type CHAR	125
4.4.7 Data types for durations and points in time	126
4.5 Complex data types	127
4.5.1 STRING data type	128
4.5.2 Data type ARRAY	129
4.5.3 Data type STRUCT	131

4.6 Parameter types and pointers	133
4.6.1 Parameter types	133
4.6.2 Pointer	135
4.6.3 “Variable” ANY pointer with STL	138
4.6.4 “Variable” ANY pointer with SCL	138
4.7 PLC data types	141
4.8 Start information	141
5 Program execution	143
5.1 Operating states of the CPU	143
5.1.1 STOP operating state	144
5.1.2 STARTUP operating state	145
5.1.3 RUN operating state	149
5.1.4 HOLD operating state	149
5.1.5 Reset CPU memory	149
5.1.6 Restoring the factory settings	150
5.1.7 Retentive behavior of operands	151
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	162
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 Startup organization blocks OB 100, OB 101, and OB 102	171
5.4.2 Determining a module address	172
5.4.3 Parameterization of modules	173
5.5 Main program	177
5.5.1 Organization block OB 1	177
5.5.2 Process image	177
5.5.3 Cycle time and response time	182
5.5.4 Minimum cycle time and background processing	184
5.5.5 Compress, hold, stop, and protect program	186
5.5.6 Time	187
5.5.7 Determine system time and OB runtime	191
5.5.8 Runtime meter	195
5.6 Interrupt processing	196
5.6.1 Introduction to interrupt processing	196
5.6.2 Priority classes	197
5.6.3 Time-of-day interrupts, organization blocks OB 10 to OB 17	199
5.6.4 Time-delay interrupts, organization blocks OB 20 to OB 23	202
5.6.5 Cyclic interrupts, organization blocks OB 30 to OB 38	205
5.6.6 Hardware interrupts, organization blocks OB 40 to OB 47	207
5.6.7 Interrupts for DPV1 organization blocks OB 55 to OB 57	208
5.6.8 Synchronous cycle interrupts, organization blocks OB 61 to OB 64	209

5.6.9 Reading additional interrupt information	212
5.7 Error handling	213
5.7.1 Causes of errors and error responses	213
5.7.2 Synchronous error	213
5.7.3 Enabling and disabling synchronous error processing	215
5.7.4 Enter substitute value	218
5.7.5 Asynchronous errors	218
5.7.6 Disable, delay, and enable interrupts and asynchronous errors	224
5.8 Diagnostics	226
5.8.1 Diagnostics interrupt, organization block OB 82	226
5.8.2 Read system state list	227
5.8.3 Read start information	228
5.8.4 Determine connection status	229
5.8.5 System diagnostics with Report System Errors	231
5.9 Configure alarms	233
5.9.1 Introduction	233
5.9.2 Configuring alarms according to the message numbering	236
5.9.3 Message blocks for PLC alarms with instance data	241
5.9.4 Message blocks for PLC alarms without instance data	244
5.9.5 Blocks for working with alarms	246
5.9.6 Configuring a user diagnostic alarm	249
5.9.7 CPU alarm display	250
6 Program editor	253
6.1 Introduction	253
6.2 PLC tag table	254
6.2.1 Editing PLC tag tables	254
6.2.2 Defining PLC tags	254
6.2.3 Exporting and importing a PLC tag table	256
6.2.4 Constants tables	257
6.3 Programming a code block	257
6.3.1 Creating a new code block	257
6.3.2 Working area of program editor for code blocks	258
6.3.3 Specifying code block properties	260
6.3.4 Programming a block interface	260
6.3.5 Programming a control function	262
6.3.6 Editing tags	266
6.3.7 Working with program comments	268
6.4 Programming a data block	270
6.4.1 Creating a new data block	270
6.4.2 Working area of program editor for data blocks	270
6.4.3 Defining properties for data blocks	271
6.4.4 Declaring data tags	272
6.4.5 Entering data tags in global data blocks	273
6.5 Compiling blocks	273
6.5.1 Starting the compilation	273
6.5.2 Compiling SCL blocks	274
6.5.3 Eliminating errors following compilation	275
6.6 Program information	276
6.6.1 Cross-reference list	276
6.6.2 Assignment list	278

6.6.3 Call structure	279
6.6.4 Dependency structure	280
6.6.5 Consistency check	281
6.6.6 Memory utilization of the CPU	281
7 Ladder logic LAD	283
7.1 Introduction	283
7.1.1 Programming with LAD in general	283
7.1.2 Program elements of ladder logic	285
7.2 Programming binary logic operations with LAD	286
7.2.1 NO and NC contacts	286
7.2.2 Series and parallel connection of contacts	287
7.2.3 T branch, open parallel branch	288
7.2.4 Negating result of logic operation	289
7.2.5 Edge evaluation of a binary tag	289
7.2.6 Comparison contacts	290
7.3 Programming memory functions with LAD	290
7.3.1 Simple coil, assignment	291
7.3.2 Set and reset coils	292
7.3.3 Retentive response due to latching	292
7.3.4 Coils with time response	293
7.3.5 Coils with counter response	294
7.4 Programming Q boxes with LAD	295
7.4.1 Memory boxes	296
7.4.2 Edge evaluation of current flow	296
7.4.3 SIMATIC timer functions	297
7.4.4 SIMATIC counter functions	298
7.4.5 IEC timer functions	299
7.4.6 IEC counter functions	300
7.5 Programming EN/ENO boxes with LAD	301
7.5.1 Transfer function, MOVE	302
7.5.2 Arithmetic functions	302
7.5.3 Math functions	303
7.5.4 Conversion functions	304
7.5.5 Shift functions	305
7.5.6 Word logic operations	306
7.6 Controlling the program flow with LAD	307
7.6.1 Working with status bits in the ladder logic	307
7.6.2 EN/ENO mechanism with LAD	309
7.6.3 Jump functions	310
7.6.4 Block functions	311
7.6.5 Master Control Relay (MCR)	313
8 Function block diagram FBD	315
8.1 Introduction	315
8.1.1 Programming with FBD in general	315
8.1.2 Program elements of the function block diagram	317
8.2 Programming binary logic operations with FBD	318
8.2.1 Scanning for signal states “1” and “0”	318
8.2.2 Programming a binary logic operation in the function block diagram	319

8.2.3 AND function	320
8.2.4 OR function	320
8.2.5 Exclusive OR function	321
8.2.6 Combined binary logic operations, negating result of logic operation	321
8.2.7 T branch	322
8.2.8 Edge evaluation of binary tags	322
8.2.9 Comparison functions	323
8.3 Programming standard boxes with FBD	324
8.3.1 Assign box	324
8.3.2 Set and reset boxes	325
8.3.3 Standard boxes with time response	326
8.3.4 Standard boxes with counter response	326
8.4 Programming Q boxes with FBD	327
8.4.1 Memory boxes	328
8.4.2 Edge evaluation of result of logic operation	329
8.4.3 SIMATIC timer functions	329
8.4.4 SIMATIC counter functions	331
8.4.5 IEC timer functions	331
8.4.6 IEC counter functions	332
8.5 Programming EN/ENO boxes with FBD	333
8.5.1 Transfer function MOVE	334
8.5.2 Arithmetic functions	335
8.5.3 Math functions	335
8.5.4 Conversion functions	336
8.5.5 Shift functions	338
8.5.6 Word logic operations	338
8.6 Controlling the program flow with FBD	340
8.6.1 Working with status bits in the function block diagram	340
8.6.2 EN/ENO mechanism with FBD	342
8.6.3 Jump functions	343
8.6.4 Block functions	344
8.6.5 Master Control Relay (MCR)	346
9 Statement list STL	348
9.1 Introduction	348
9.1.1 Programming with STL in general	348
9.1.2 Structure of an STL statement	349
9.2 Programming binary logic operations with STL	350
9.2.1 Processing of a binary logic operation, operation step	350
9.2.2 Scanning for signal states "1" and "0"	352
9.2.3 Programming a binary logic operation in the statement list	353
9.2.4 AND function	354
9.2.5 OR function	354
9.2.6 Exclusive OR function	354
9.2.7 Combined binary logic operations	355
9.2.8 Control of result of logic operation	357
9.3 Programming memory functions with STL	358
9.3.1 Assignment	359
9.3.2 Setting and resetting	359
9.3.3 Edge evaluation	360

9.4 Programming timer and counter functions with STL	361
9.4.1 SIMATIC timer functions	361
9.4.2 SIMATIC counter functions	363
9.4.3 IEC timer functions	364
9.4.4 IEC counter functions	365
9.5 Programming digital functions with STL	366
9.5.1 Transfer functions	367
9.5.2 Comparison functions	367
9.5.3 Arithmetic functions	370
9.5.4 Math functions	373
9.5.5 Conversion functions	374
9.5.6 Shift functions	375
9.5.7 Word logic operations	377
9.6 Controlling the program flow with STL	380
9.6.1 Working with status bits in the statement list	380
9.6.2 EN/ENO mechanism with STL	382
9.6.3 Jump functions	384
9.6.4 Jump list	385
9.6.5 Loop jump	385
9.6.6 Block functions	386
9.6.7 Master Control Relay (MCR)	389
9.7 Further STL functions	390
9.7.1 Accumulator functions	390
9.7.2 Adding of constants to accumulator 1	393
9.7.3 Decrementing, incrementing	394
9.7.4 Null instructions	395
10 Structured Control Language SCL	397
10.1 Introduction to programming with SCL	397
10.1.1 Programming with SCL in general	397
10.1.2 SCL statements and operators	398
10.2 Programming binary logic operations with SCL	401
10.2.1 Scanning for signal states “1” and “0”	401
10.2.2 AND function	402
10.2.3 OR function	403
10.2.4 Exclusive OR function	403
10.2.5 Combined binary logic operations	403
10.2.6 Negating result of logic operation	404
10.3 Programming memory functions with SCL	404
10.3.1 Value assignment of a binary tag	405
10.3.2 Setting and resetting	405
10.3.3 Edge evaluation	405
10.4 Programming timer and counter functions with SCL	406
10.4.1 SIMATIC timer functions	406
10.4.2 SIMATIC counter functions	407
10.4.3 IEC timer functions	408
10.4.4 IEC counter functions	408
10.5 Programming digital functions with SCL	409
10.5.1 Transfer function, value assignment of a digital tag	410
10.5.2 Comparison functions	410
10.5.3 Arithmetic functions	411
10.5.4 Math functions	412

10.5.5 Conversion functions	413
10.5.6 Shift functions	414
10.5.7 Word logic operations, logic expression	415
10.6 Controlling the program flow with SCL	416
10.6.1 Working with the ENO tag	416
10.6.2 EN/ENO mechanism with SCL	417
10.6.3 Control statements	419
10.6.4 Block functions	428
11 S7-GRAPH sequential control	431
11.1 Introduction	431
11.1.1 What is a sequential control?	431
11.1.2 Properties of a sequential control	432
11.1.3 Program for a sequential control, quantity framework	433
11.1.4 Operating modes	433
11.1.5 Procedure for configuration	434
11.2 Elements of a sequential control	434
11.2.1 Steps and transitions	434
11.2.2 Jumps in a sequential control	436
11.2.3 Branching of a sequencer	436
11.2.4 GRAPH-specific tags	437
11.2.5 Permanent instructions	438
11.2.6 Step and transition functions	439
11.2.7 Processing of actions	442
11.3 Configuring a sequential control	448
11.3.1 Programming the GRAPH function block	448
11.3.2 Configuring the sequencer structure	449
11.3.3 Programming steps and transitions	451
11.3.4 Programming permanent instructions	452
11.3.5 Configuring block-independent alarms	453
11.3.6 Attributes of the GRAPH function block	453
11.3.7 Using the GRAPH function block	454
11.4 Testing the sequential control	455
11.4.1 Loading the GRAPH function block	456
11.4.2 Settings for program testing	456
11.4.3 Using operating modes	457
11.4.4 Synchronization a sequencer	458
11.4.5 Testing with program status	458
12 Basic functions	461
12.1 Binary logic operations	461
12.1.1 Introduction	461
12.1.2 Working with binary signals	462
12.1.3 AND function, series connection	464
12.1.4 OR function, parallel connection	465
12.1.5 Exclusive OR function, non-equivalence function	465
12.1.6 Negate result of logic operation, NOT contact	466
12.2 Memory functions	468
12.2.1 Introduction	468
12.2.2 Standard coil, assignment	469

12.2.3	Single setting and resetting	469
12.2.4	Dominant setting and resetting, memory function	471
12.2.5	Edge evaluation	472
12.3	SIMATIC timer functions	477
12.3.1	Overview	477
12.3.2	Programming a timer function	480
12.3.3	Timer response as pulse	482
12.3.4	Timer response as extended pulse	484
12.3.5	Timer response as ON delay	486
12.3.6	Timer response as retentive ON delay	487
12.3.7	Timer response as OFF delay	489
12.4	IEC timer functions	491
12.4.1	Introduction	491
12.4.2	Pulse generation TP	492
12.4.3	ON delay TON	493
12.4.4	OFF delay TOF	494
12.5	SIMATIC counter functions	495
12.5.1	Overview	495
12.5.2	Programming a counter function	498
12.5.3	Principle of operation of a counter function	499
12.5.4	Enabling a counter function with STL	501
12.6	IEC counter functions	502
12.6.1	Introduction	502
12.6.2	Up counter CTU	503
12.6.3	Down counter CTD	504
12.6.4	Up/down counter CTUD	505
13	Digital functions	507
13.1	General information	507
13.2	Transfer functions	508
13.2.1	General information on the “simple” transfer function	508
13.2.2	MOVE box with LAD and FBD	508
13.2.3	Loading and transferring with STL	510
13.2.4	Value assignments with SCL	511
13.2.5	Copying and filling a data area in the work memory	513
13.2.6	Control memory area with MCR dependency	515
13.3	Comparison functions	518
13.3.1	Execution of “simple” comparison function	518
13.3.2	Comparison function T_COMP	520
13.3.3	Comparison function S_COMP	521
13.4	Arithmetic functions	521
13.4.1	General function description	523
13.4.2	Data types and status bits for an arithmetic function	523
13.4.3	Execution of the arithmetic function	524
13.4.4	Arithmetic functions for date and time	525
13.5	Math functions	527
13.5.1	General function description	527
13.5.2	General execution of a math function	527
13.5.3	Trigonometric functions SIN, COS, TAN	528
13.5.4	Arc functions ASIN, ACOS, ATAN	529
13.5.5	Additional math functions	529

13.6 Conversion functions	531
13.6.1 Implicit data type conversion	531
13.6.2 Data type conversion of fixed-point numbers	532
13.6.3 Data type conversion of floating-point numbers	535
13.6.4 Data type conversion for date/time with T_CONV	537
13.6.5 Data type conversion for data type STRING with S_CONV	539
13.6.6 Data type conversion of hexadecimal numbers	540
13.6.7 Scaling and unscaling	541
13.6.8 Further conversion functions	543
13.7 Shift functions	544
13.7.1 General function description	544
13.7.2 General execution of a shift function	544
13.7.3 Shift to right	546
13.7.4 Shift to left	547
13.7.5 Rotate to right	547
13.7.6 Rotate to left	548
13.7.7 Rotating by the condition code bit CC1 (STL)	549
13.8 Logic functions	549
13.8.1 Word logic operations	549
13.8.2 Invert	552
13.8.3 Code bit and set bit number	552
13.8.4 Selection and limiting functions	553
13.9 Functions for strings	556
14 Program flow control	560
14.1 Status bits	561
14.1.1 Description of the status bits	561
14.1.2 Controlling the status bits	563
14.1.3 Setting and resetting the result of logic operation	563
14.1.4 Controlling the binary result	565
14.1.5 Evaluating the status bits	566
14.2 Jump functions	568
14.2.1 Introduction	568
14.2.2 Absolute jump	568
14.2.3 Conditional jump functions	570
14.2.4 Jump functions depending on status bits	572
14.3 Block end functions	575
14.3.1 Block end function RET (LAD and FBD)	575
14.3.2 Block end functions BEC, BEU, and BE (STL)	576
14.3.3 RETURN statement (SCL)	576
14.4 Calling of code blocks	576
14.4.1 General information on block calls	576
14.4.2 Calling a function (FC)	577
14.4.3 Calling a function block (FB)	579
14.4.4 Change to a block without block parameter	581
14.5 Data block functions	583
14.5.1 Opening a data block	583
14.5.2 Additional data block functions with STL	584
14.5.3 Creating, deleting, and testing data blocks	585
14.6 Master control relay	587
14.6.1 Introduction	587
14.6.2 MCR dependency	588

14.6.3 MCR area and MCR zone	589
14.6.4 MCR area and MCR zone with a block change	589
14.6.5 Statements for the master control relay	591
15 Online operation and program test	592
15.1 Connection of a programming device to the PLC station	593
15.1.1 Settings on the programming device	593
15.1.2 Connecting the programming device to the PLC station	594
15.1.3 Switching on online mode	595
15.2 Transferring project data	596
15.2.1 Loading project data for the first time	596
15.2.2 Reloading the project data	598
15.2.3 Protection of the user program	599
15.2.4 Editing of online project without offline project	600
15.2.5 Working with the memory card	601
15.3 Block handling	602
15.3.1 Downloading a block to the CPU	602
15.3.2 Editing the online version of a block	603
15.3.3 Deleting a block	603
15.3.4 Packing the work memory	603
15.3.5 Offline/online data blocks	604
15.3.6 Comparing blocks	605
15.4 Hardware diagnostics	608
15.4.1 Status displays on the modules	608
15.4.2 Diagnostic information	609
15.4.3 Diagnostic buffer	609
15.4.4 Diagnostic functions	610
15.4.5 Online tools	611
15.4.6 Further diagnostic information via the programming device	612
15.5 Testing the user program	613
15.5.1 Process and test modes	613
15.5.2 Defining the call environment	614
15.5.3 Testing with program status	614
15.5.4 Testing in single step mode	618
15.5.5 Monitoring of PLC tags	621
15.5.6 Monitoring of data tags	621
15.5.7 Testing with watch tables	622
15.5.8 Enable peripheral outputs	627
15.5.9 Testing with the force table	628
16 Distributed I/O	631
16.1 Introduction, overview	631
16.2 ET 200 distributed I/O system	632
16.2.1 ET 200L	632
16.2.2 ET 200M	632
16.2.3 ET 200S	633
16.2.4 ET 200iSP	634
16.2.5 ET 200R	634
16.2.6 ET 200eco	634
16.2.7 ET 200pro	635

16.3 PROFINET IO	636
16.3.1 PROFINET IO components	636
16.3.2 Addresses with PROFINET IO	638
16.3.3 Special PROFINET configurations	641
16.3.4 Configuring PROFINET IO	642
16.3.5 Coupling modules for PROFINET IO	646
16.3.6 Real-time communication with PROFINET IO	647
16.4 PROFIBUS DP	649
16.4.1 PROFIBUS DP components	649
16.4.2 Addresses with PROFIBUS DP	652
16.4.3 Configuring PROFIBUS DP	656
16.4.4 Coupling modules for PROFIBUS DP	659
16.4.5 Special functions for PROFIBUS DP	661
16.5 System blocks for distributed I/O	665
16.5.1 System blocks for PROFIBUS DP	665
16.5.2 System blocks for PROFIBUS DP and PROFINET IO	668
16.5.3 System block for PROFINET IO	672
 17 Communication	 674
17.1 Overview	674
17.2 S7 basic communication	675
17.2.1 Basics of station-internal S7 basic communication	675
17.2.2 Configuring of station-internal S7 basic communication	676
17.2.3 System blocks for station-internal S7 basic communication	676
17.2.4 Basics of station-external S7 basic communication	678
17.2.5 Configuring of station-external S7 basic communication	679
17.2.6 System blocks for station-external S7 basic communication	679
17.3 S7 Communication	682
17.3.1 Basics	682
17.3.2 Configuring S7 communication	684
17.3.3 One-way data exchange	685
17.3.4 Two-way data exchange	686
17.3.5 Control functions	688
17.3.6 Monitoring functions	690
17.3.7 Send print data	691
17.4 Open user communication	692
17.4.1 Basics	692
17.4.2 Establishing and terminating connections	693
17.4.3 Data transfer with TCP native or ISO-on-TCP	696
17.4.4 Data transfer with UDP	698
17.5 Point-to-point communication	700
17.5.1 Basics	700
17.5.2 Data transmission with the 3964 (R) procedure	702
17.5.3 Data transmission with the RK 512 protocol	702
17.5.4 Data transmission with the ASCII driver	705
 18 Appendix	 707
18.1 Working with source files	707
18.1.1 General procedure	707
18.1.2 Programming a logic block in the source file	708

18.1.3 Programming a data block in the source file	714
18.1.4 Programming a PLC data type in the source file	715
18.2 Migrating projects	716
18.3 Simulation with the TIA Portal	720
18.3.1 Differences from a real CPU	720
18.3.2 Starting and saving the simulation	720
18.3.3 Using the simulation	722
18.3.4 Testing the program with the simulation	724
18.3.5 Additional functions of PLCSIM	725
18.4 Web server	727
18.4.1 Enable web server	727
18.4.2 Reading out web information	728
18.4.3 Standard web pages	728
18.5 Storage of local tags	731
18.5.1 Storage in global data blocks	731
18.5.2 Storage in instance data blocks	732
18.5.3 Storage in the temporary local data	733
18.5.4 Data storage of the block parameters of a function (FC)	734
18.5.5 Data storage of the block parameters of a function block (FB)	736
18.5.6 Data storage of a local instance in a multi-instance	739
Index	741

1 Introduction

1.1 Overview of the S7-400 automation system

SIMATIC S7-400 is the modular control system for the medium and upper performance range (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 modularly expanded by input/output modules for digital and analog signals in up to 21 racks with up to 18 modules.

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 S7-400 controllers. Data

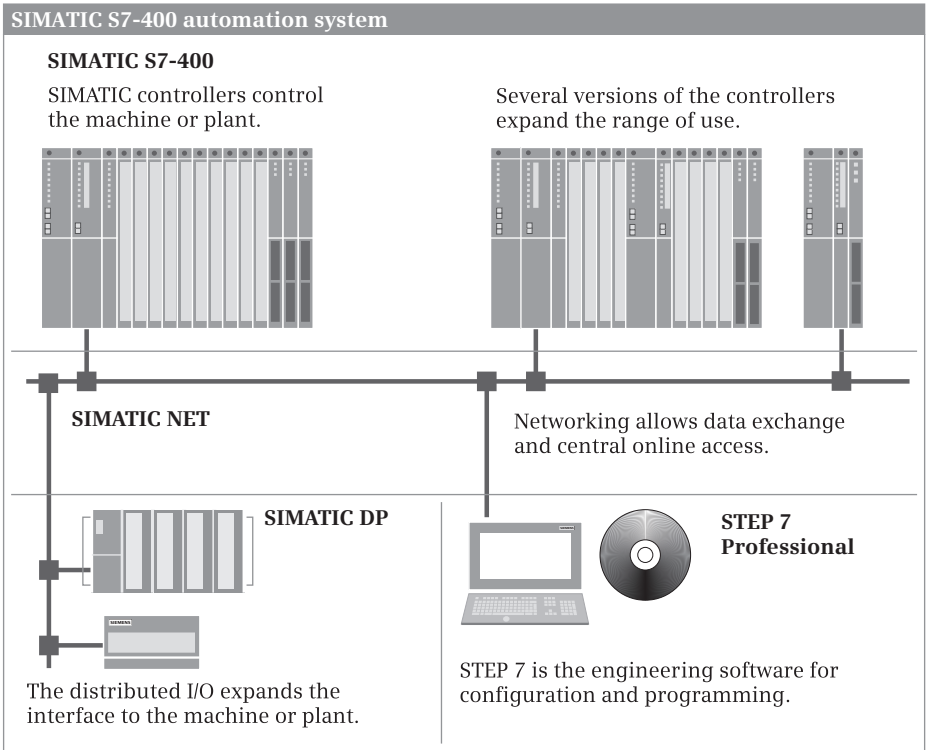


Fig. 1.1 Components of the SIMATIC S7-400 automation system

exchange between the controllers, the distributed I/O, and the programming device is carried out over SIMATIC NET.

1.1.1 SIMATIC S7-400 programmable controller

The most important components of an S7-400 programmable controller are shown in Fig. 1.2.

The **CPU** contains the operating system and the user program. The user program is in the load memory of the CPU, which can be expanded with a *SIMATIC Memory Card (MC)*. 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.

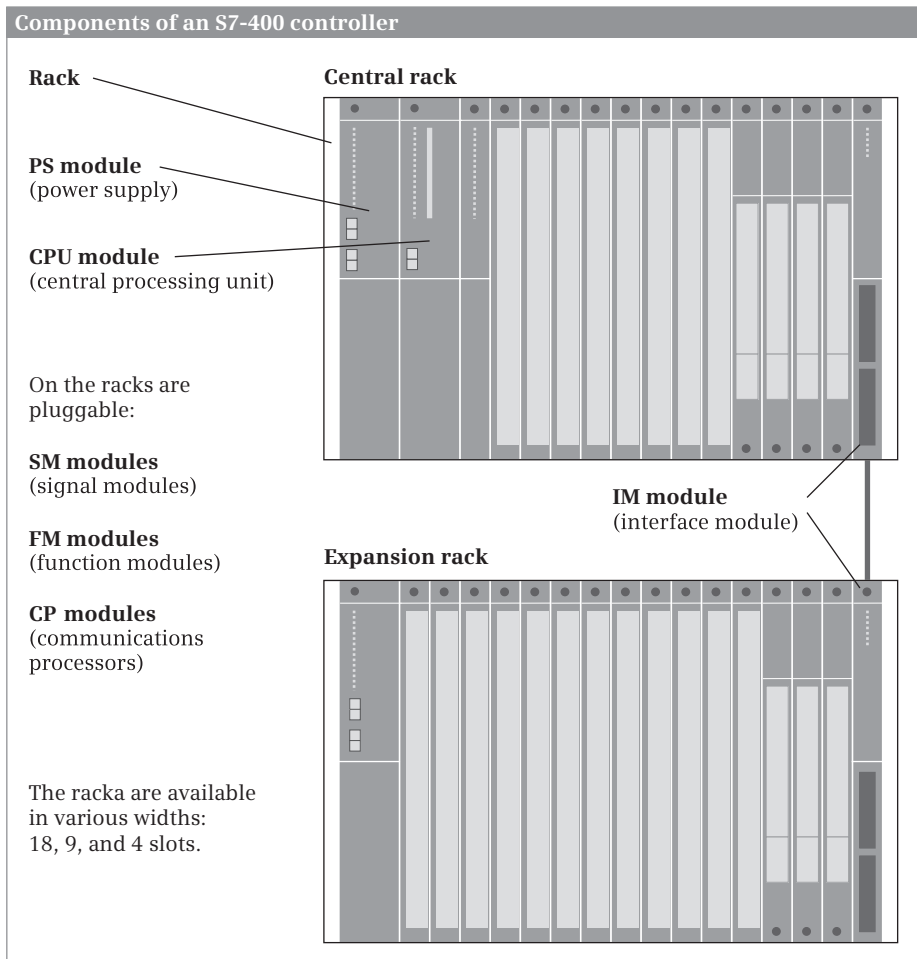


Fig. 1.2 Components of an S7-400 controller

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 on the CPU 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 21 expansion racks.

Finally, a **power supply module** provides the internal voltages required by the programmable controller. Load voltages or load currents must be provided via external load current supply units.

1.1.2 Overview of STEP 7 Professional V11

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 S7-400 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):

- ▷ 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.
- ▷ 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,

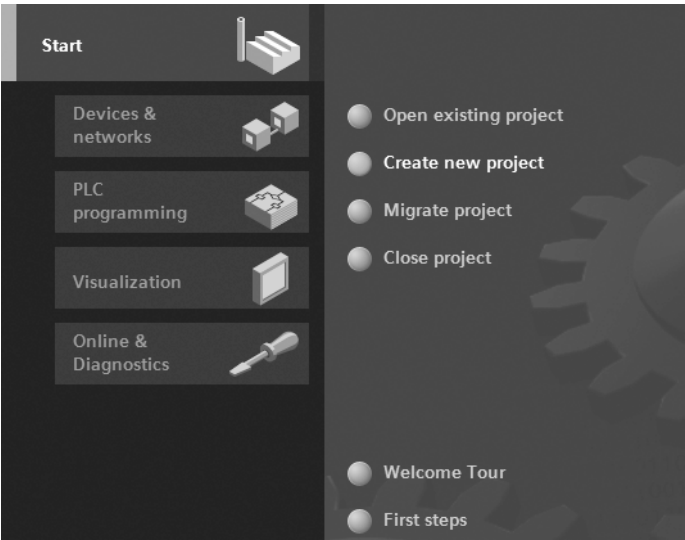


Fig. 1.3 Tools in the Start portal of STEP 7 Professional V11

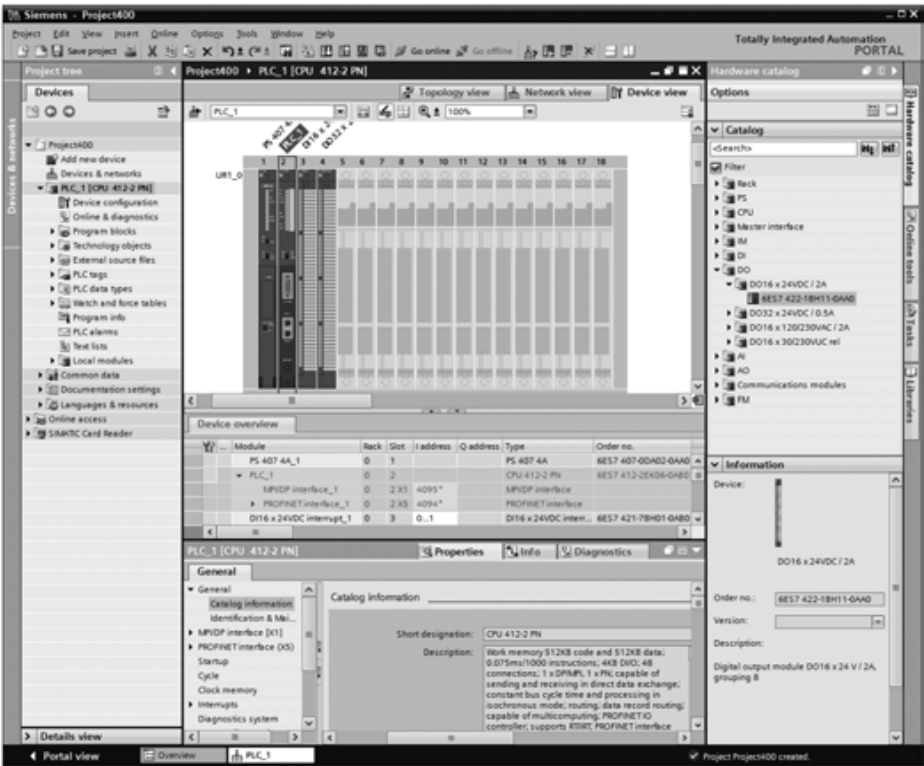


Fig. 1.4 Example of a Project view: working area of the device configuration

and the task window provides support by means of the hardware catalog with the available modules. The Network view allows networking between PLC and HMI stations.

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 window contains the program elements catalog with the available program elements and statements. 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 navigation*. 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.

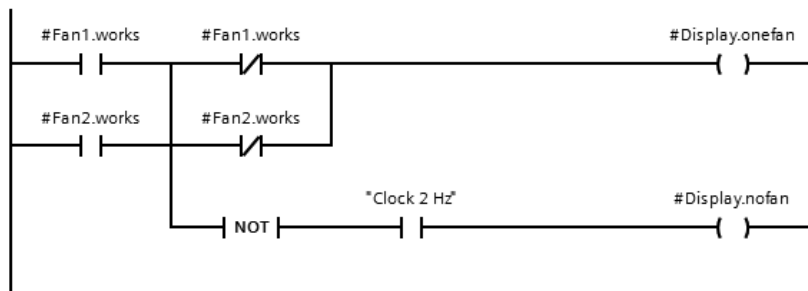
1.1.3 Five programming languages

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 diagram.

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 terminated by memory boxes (Fig. 1.6). Complex boxes are used to handle the operations on digital tags, for example with arithmetic functions.

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



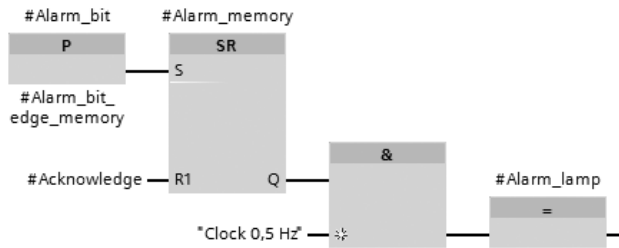


Fig. 1.6 Example of representation in function block diagram

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

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).

```

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

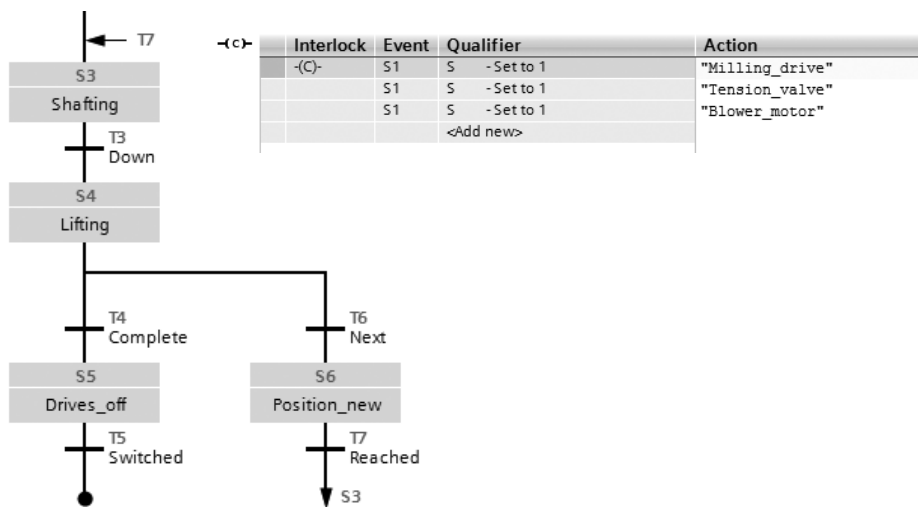


Fig. 1.9 Example of a GRAPH sequencer and step configuration

Using **GRAPH**, you program a control task as a sequence 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”. The 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).

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 statements 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).

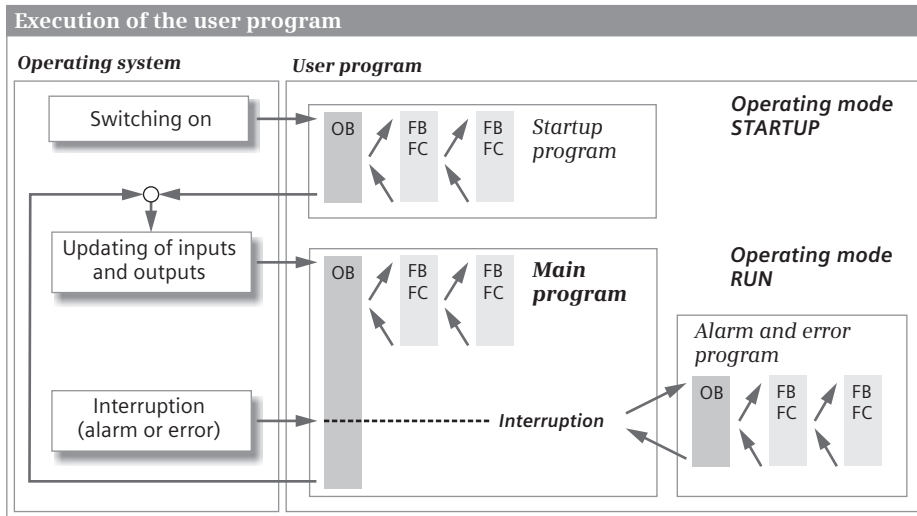


Fig. 1.10 Execution of the user program

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 restart, 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.

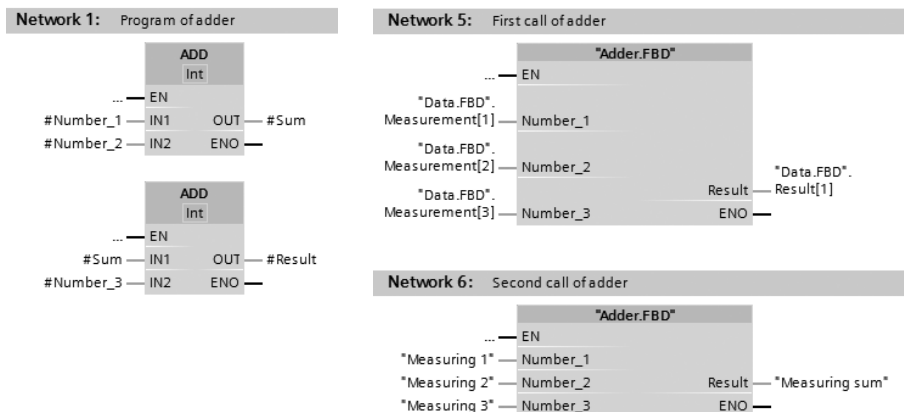


Fig. 1.11 Example of two block calls with different tags in each case

Cyclic program execution is a feature of programmable logic controllers. The user program is also 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 event for this. This can be a hardware interrupt, such as a request from the controlled machine for a fast response, or a cyclic interrupt, in other words an event which takes place at defined intervals.

The control processor interrupts execution of the main program when an event occurs, and calls the assigned interrupt program. Once the interrupt program has been executed, the control processor continues execution of the main program from the point of interruption.

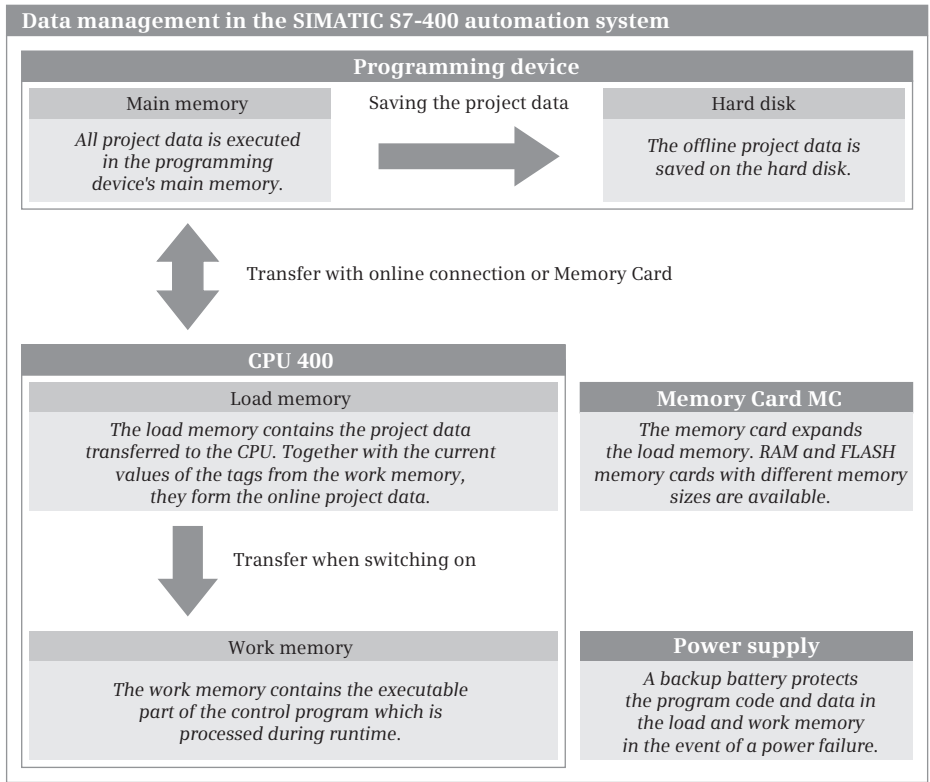


Fig. 1.12 Data management in the SIMATIC S7-400 automation system