

Hans Berger

Automating with SIMATIC S7-1200

Configuring, Programming and Testing
with STEP 7 Basic
Visualization with WinCC Basic

SIEMENS



Second Edition

Berger Automating with SIMATIC S7-1200

Automating with SIMATIC S7-1200

Configuring, Programming and
Testing with STEP 7 Basic
Visualization with HMI Basic

by Hans Berger

2nd enlarged and revised edition, 2013

Publicis Publishing

Bibliographic information published by 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

Print ISBN: 978-3-89578-385-2

ePDF ISBN: 978-3-89578-901-4

2nd edition, 2013

Editor: Siemens Aktiengesellschaft, Berlin and Munich

Publisher: Publicis Publishing, Erlangen

© 2013 by Publicis Erlangen, Zweigniederlassung der PWW GmbH

This 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 illustrations or 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* 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 newly developed SIMATIC S7-1200 automation system. The S7-1200 programmable controllers are of compact design and allow modular expansion. Many small applications can be solved using the CPU module with on-board I/O. The technological functions integrated in the CPU module mean that extremely versatile use of the device is possible. Two established programming languages are available for solving automation tasks: ladder logic (LAD) and function block diagram (FBD).

New SIMATIC HMI Basic Panels have been designed for operator control and monitoring appropriate to the S7-1200 programmable controllers, and provide a performance and functionality optimized for small applications. A touch screen with various monitor sizes and coordinated communication over Industrial Ethernet are ideal prerequisites for interaction with S7-1200.

The STEP 7 Basic engineering software makes it possible to use all S7-1200 controller options. STEP 7 Basic is the common tool for hardware configuration, generation of the control program, and for debugging and diagnostics. The SIMATIC WinCC Basic configuration software included in STEP 7 Basic is used to configure the Basic Panels. Modern and intuitive user guidance allows efficient and task-oriented engineering of control and visualization devices.

This book describes the S7-1200 automation system with S7-1200 programmable controllers and HMI Basic Panels. The description focuses on the generation of the control program using STEP 7 Basic engineering software Version 11 SP2.

Nuremberg, February 2013

Hans Berger

The contents of the book at a glance

Start

Introduction

SIMATIC S7-1200: Overview of the SIMATIC S7-1200 automation system.

STEP 7 Basic: Introduction to the engineering software for SIMATIC S7-1200.

SIMATIC project: Basic functions for the automation solution.

Devices & networks

The hardware components of S7-1200

Modules: Overview of the SIMATIC S7-1200 modules.

Device configuration

Hardware configuration: Configuration of the hardware design.

Network configuration: Configuration of a communication network.

PLC programming

The control program

Operating modes: How the CPU module responds with STARTUP, RUN and STOP.

Processing modes: Restart characteristics, main program, interrupt processing, and error handling define the processing of the control program.

Blocks: Organization blocks, function blocks, functions, and data blocks structure the control program.

The program editor

Programming: How the control program is produced.

Program information: Tools for supporting programming.

Ladder logic and function block diagram as programming languages

Program elements: The characteristics of LAD and FBD programming; the use of contacts, coils, standard boxes, Q boxes and EN/ENO boxes.

Tags and data types

Tags: Operand areas, project-wide and block-local tags, addressing.

Data types: Description of elementary and compound data types.

Description of the control functions

Basic functions: Binary operations, memory functions, edge evaluation, timer and counter functions.

Digital functions: Move, comparison, arithmetic, math, conversion, shift, and logic functions.

Program flow control: Jump functions, block end function, block calls.

Online & diagnostics

Connection of programming device to PLC station

Online operation: Establish connection to PLC station.

Status LEDs: The modules signal an error.

Diagnostics information: Find the error using the diagnostics information.

Online tools: Control the CPU module using the online tools.

Online & offline project data

Download: Download control program into CPU memory.

Blocks: Edit and compare the blocks offline/online.

Test: Test the control function using program status and monitoring tables.

Data communication

Open user communication

Data transmission: Data exchange from PLC to PLC over Ethernet.

Point-to-point connection

PtP: Data transmission with CM modules via RS232 and RS485.

Visualization

Configuration of Basic Panels

Introduction: Overview of Basic Panels.

Start: Create an HMI project, the HMI device wizard.

Connection to the PLC: Create HMI tags and area pointers.

Create screens: Configuration of process screens – templates, layers and screen changeover.

Working with image elements: Arrange and edit operator control and display elements, configure a message system, create recipes, transfer data records, configure user management.

Complete the HMI program

Simulation: Simulate the HMI program with PLC station or with tag table.

Connection: Transfer the HMI program to the HMI station.

Appendix

Integral and technological functions

Functions: High-speed counter, pulse generator, motion control, PID controller.

Global libraries

Overview: USS drive control, MODBUS blocks.

Table of contents

1 Introduction	21
1.1 Overview of the S7-1200 automation system	21
1.1.1 SIMATIC S7-1200	22
1.1.2 Overview of STEP 7 Basic	24
1.1.3 Three programming languages	25
1.1.4 Execution of the user program	27
1.1.5 Data management in the SIMATIC automation system	29
1.1.6 Operator control and monitoring with process images	30
1.2 Introduction to STEP 7 Basic for S7-1200	31
1.2.1 Installing STEP 7	31
1.2.2 Automation License Manager	31
1.2.3 Starting STEP 7 Basic	32
1.2.4 Portal view	32
1.2.5 Help Information system	33
1.2.6 The windows of the project view	34
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	41
1.3.4 Creating and editing libraries	42
 2 SIMATIC S7-1200 automation system	 43
2.1 S7-1200 station components	43
2.2 S7-1200 CPU modules	44
2.2.1 Integrated I/O	44
2.2.2 PROFINET connection	46
2.2.3 Status LEDs	47
2.2.4 SIMATIC Memory Card	47
2.2.5 Expansions of the CPU	47
2.3 Signal modules (SM)	49
2.3.1 Digital I/O modules	49
2.3.2 Analog input/output modules	50
2.3.3 Properties of the I/O connections	50
2.4 Communication modules (CM)	52
2.4.1 Point-to-point communication	52
2.4.2 PROFIBUS DP	52
2.4.3 Actuator/sensor interface	53
2.4.4 GPRS transmission	53
2.5 Further modules	54
2.5.1 Compact switch module (CSM)	54

2.5.2 Power module (PM)	54
2.5.3 TS Adapter IE Basic	54
2.5.4 SIM 1274 simulator	55
2.6 SIPLUS S7-1200	55
3 Device configuration	57
3.1 Introduction	57
3.2 Configuring a station	60
3.2.1 Adding a PLC station	60
3.2.2 Arranging modules	61
3.2.3 Adding an HMI station	61
3.3 Assigning module parameters	61
3.3.1 Parameterization of CPU properties	61
3.3.2 Addressing input and output signals	64
3.3.3 Parameterization of digital inputs	65
3.3.4 Parameterization of digital outputs	65
3.3.5 Parameterization of analog inputs	66
3.3.6 Parameterization of analog outputs	66
3.4 Configuring the network	67
3.4.1 Introduction	67
3.4.2 Networking stations	68
3.4.3 Node addresses in a subnet	69
3.4.4 Connectors	70
3.4.5 Configuring a PROFINET subnet	73
3.4.6 Configuring a PROFIBUS subnet	75
3.4.7 Configuring an AS-i subnet	77
4 Variables and data types	79
4.1 Operands and tags	79
4.1.1 Introduction, overview	79
4.1.2 Operand areas: inputs and outputs	80
4.1.3 Operand area bit memory	82
4.1.4 Operand area data	84
4.1.5 Operand area temporary local data	85
4.2 Addressing	85
4.2.1 Signal path	85
4.2.2 Absolute addressing of an operand	86
4.2.3 Absolute addressing of an operand area	86
4.2.4 Symbolic addressing	88
4.2.5 Addressing a tag part	89
4.2.6 Addressing constants	89
4.2.7 Indirect addressing	89
4.3 General information on data types	92
4.3.1 Overview of data types	92
4.3.2 Implicit data type conversion	93
4.3.3 Overlying tags (data type views)	93
4.4 Elementary data types	95
4.4.1 Bit-serial data types BOOL, BYTE, WORD and DWORD	95

4.4.2 BCD-coded numbers BCD16 and BCD32	95
4.4.3 Unsigned fixed-point data types USINT, UINT and UDINT	97
4.4.4 Fixed-point data types with sign SINT, INT and DINT	98
4.4.5 Floating-point data types REAL and LREAL	98
4.4.6 Data type CHAR	100
4.4.7 Data type DATE	100
4.4.8 Data type TIME	100
4.4.9 TIME_OF_DAY (TOD) data type	101
4.5 Structured data types	101
4.5.1 Data type DTL	101
4.5.2 Data type STRING	102
4.5.3 Data type ARRAY	104
4.5.4 Data type STRUCT	104
4.6 Parameter types	107
4.6.1 Parameter types for IEC timer functions	107
4.6.2 Parameter types for IEC counter functions	108
4.6.3 Parameter type VARIANT	108
4.6.4 Parameter type VOID	109
4.7 PLC data types	109
4.8 System data types	110
4.8.1 IEC_TIMER system data type	110
4.8.2 IEC_COUNTER system data type	112
4.8.3 TCON_Param data type	112
4.8.4 TADDR_Param data type	112
4.8.5 Data type ErrorStruct	112
4.8.6 TimeTransformationRule data type	115
4.9 Hardware data types	115
5 Edit user program	117
5.1 Operating modes	117
5.1.1 STOP mode	118
5.1.2 STARTUP mode	118
5.1.3 RUN mode	119
5.1.4 Retentive behavior of operands	121
5.2 Creating a user program	122
5.2.1 Program draft	122
5.2.2 Program execution	123
5.2.3 Nesting depth	125
5.3 Programming blocks	125
5.3.1 Block types	125
5.3.2 Editing block properties	128
5.3.3 Configuring know-how protection	132
5.3.4 Copy protection	132
5.3.5 Block interface	133
5.3.6 Programming block parameters	136
5.4 Calling blocks	137
5.4.1 General information on calling logic blocks	137
5.4.2 Calling a function (FC)	139

5.4.3 Calling a function block (FB)	140
5.4.4 "Passing on" of block parameters	142
5.5 Start-up routine	142
5.6 Main program	143
5.6.1 Organization blocks for the main program	143
5.6.2 Process image update	143
5.6.3 Cycle time	144
5.6.4 Reaction time	146
5.6.5 Stop program execution	147
5.6.6 Time	148
5.6.7 Runtime meter	151
5.7 Interrupt processing	153
5.7.1 Introduction to interrupt processing	153
5.7.2 Time-delay interrupts	155
5.7.3 Cyclic interrupts	159
5.7.4 Process interrupts	163
5.7.5 Assigning interrupts during runtime	164
5.7.6 Delay and enable interrupts	166
5.8 Troubleshooting, diagnostics	167
5.8.1 Causes of errors and responses	167
5.8.2 Error display with the ENO output	168
5.8.3 Time error OB 80	168
5.8.4 Local error handling	169
5.8.5 Diagnostic functions in the user program	172
5.8.6 Diagnostics interrupt OB 82	176
6 Program editor	178
6.1 Introduction	178
6.2 PLC tag table	178
6.2.1 Creating and editing the PLC tag table	179
6.2.2 Defining PLC tags	179
6.2.3 Editing a PLC tag table	181
6.2.4 Exporting and importing a PLC tag table	181
6.2.5 Constants tables	182
6.3 Programming a code block	183
6.3.1 Creating a new code block	183
6.3.2 Working area of program editor for code blocks	184
6.3.3 Specifying code block properties	186
6.3.4 Programming a block interface	186
6.3.5 Programming control functions	188
6.3.6 Editing tags	192
6.3.7 Working with program comments	193
6.4 Programming a data block	194
6.4.1 Creating a new data block	194
6.4.2 Working area of program editor for data blocks	195
6.4.3 Defining properties for data blocks	196
6.4.4 Declaring data tags	196
6.4.5 Entering data tags in global data blocks	198

6.5 Compiling blocks	198
6.5.1 Starting the compilation	198
6.5.2 Compiling SCL blocks	199
6.5.3 Eliminating errors following compilation	200
6.6 Program information	201
6.6.1 Cross-reference list	201
6.6.2 Assignment list	203
6.6.3 Call structure	204
6.6.4 Dependency structure	205
6.6.5 Consistency check	206
6.6.6 CPU resources	206
6.7 Language setting	207
7 Ladder logic LAD	209
7.1 Introduction	209
7.1.1 Programming with LAD in general	209
7.1.2 Program elements of ladder logic	211
7.2 Programming with contacts	212
7.2.1 NO and NC contacts	212
7.2.2 Consideration of sensor type in ladder logic	213
7.2.3 Series connection of contacts	215
7.2.4 Parallel connection of contacts	215
7.2.5 Mixed series and parallel connections	216
7.2.6 T branch, open parallel branch in the ladder logic	217
7.2.7 Negating result of logic operation in the ladder logic	218
7.2.8 Edge evaluation of a binary tag in ladder logic	218
7.2.9 OK contact	219
7.2.10 Comparison contacts	219
7.3 Programming with coils	221
7.3.1 Simple and negated coils	222
7.3.2 Set and reset coil	223
7.3.3 Retentive response due to latching	223
7.3.4 Edge evaluation with pulse output in the ladder logic	224
7.3.5 Multiple setting and resetting (filling of bit field) in the ladder logic ..	225
7.3.6 Starting IEC timer functions in the ladder logic with coils	225
7.4 Programming with Q boxes in the ladder logic	226
7.4.1 Arrangement of Q boxes in the ladder logic	226
7.4.2 Memory boxes in the ladder logic	227
7.4.3 Edge evaluation of current flow	229
7.4.4 Example of binary scaler in the ladder logic	229
7.4.5 Controlling IEC timer functions in the ladder logic with Q boxes	230
7.4.6 Controlling IEC counter functions in the ladder logic with Q boxes ...	231
7.5 Programming with EN/ENO boxes in the ladder logic	233
7.5.1 Positioning of EN/ENO boxes in the ladder logic	234
7.5.2 Transfer functions in the ladder logic	235
7.5.3 Arithmetic functions for numerical values in the ladder logic	236
7.5.4 Arithmetic functions for time values in the ladder logic	236
7.5.5 Math functions in the ladder logic	237

7.5.6 Conversion functions in the ladder logic	238
7.5.7 Shift functions in the ladder logic	239
7.5.8 Logic functions in the ladder logic	240
7.5.9 Functions for strings in the ladder logic	240
7.6 Functions for program flow control (LAD)	241
7.6.1 Jump functions in the ladder logic	242
7.6.2 Jump list in the ladder logic	243
7.6.3 Jump distributor in the ladder logic	244
7.6.4 Block end function in the ladder logic	244
7.6.5 Block call functions in the ladder logic	245
8 Function block diagram FBD	246
8.1 Introduction	246
8.1.1 Programming with function block diagram in general	246
8.1.2 Program elements of the function block diagram	248
8.2 Programming of binary logic operations (FBD)	249
8.2.1 Scanning for signal states “1” and “0”	250
8.2.2 Taking account of the sensor type in the function block diagram	251
8.2.3 AND function	252
8.2.4 OR function	253
8.2.5 Exclusive OR function	254
8.2.6 Mixed binary logic operations	254
8.2.7 T branch in the function block diagram	255
8.2.8 Negate result of logic operation in the function block diagram	255
8.2.9 Edge evaluation of binary tags in the function block diagram	256
8.2.10 Validity checking of floating-point numbers in the function block diagram	257
8.2.11 Comparison functions in the function block diagram	258
8.3 Programming with standard boxes (FBD)	258
8.3.1 Assignment and negated assignment	259
8.3.2 Set and reset boxes	260
8.3.3 Edge evaluation with pulse output in the function block diagram	261
8.3.4 Multiple setting and resetting (filling of bit field) in the function block diagram	262
8.3.5 Starting IEC timer functions in the function block diagram with standard boxes	262
8.4 Programming with Q boxes (FBD)	264
8.4.1 Arrangement of Q boxes in the function block diagram	264
8.4.2 Memory boxes in the function block diagram	265
8.4.3 Edge evaluation of logic operation result in the function block diagram	266
8.4.4 Example of binary scaler in the function block diagram	267
8.4.5 Controlling IEC timer functions in the function block diagram with Q boxes	267
8.4.6 IEC counter functions in the function block diagram	268
8.5 Programming with EN/ENO boxes (FBD)	270
8.5.1 Positioning of EN/ENO boxes in the function block diagram	270
8.5.2 Transfer functions in the function block diagram	271

8.5.3 Arithmetic functions for numerical values in the function block diagram	273
8.5.4 Arithmetic functions with time values in the function block diagram ..	273
8.5.5 Math functions in the function block diagram	274
8.5.6 Conversion functions in the function block diagram	275
8.5.7 Shift functions in the function block diagram	276
8.5.8 Logic functions in the function block diagram	277
8.5.9 Functions for strings in the function block diagram	278
8.6 Functions for program flow control (FBD)	279
8.6.1 Jump functions in the function block diagram	280
8.6.2 Jump list in the function block diagram	281
8.6.3 Jump distributor in the function block diagram	281
8.6.4 Block end function in the function block diagram	282
8.6.5 Block call functions in the function block diagram	282
9 Structured Control Language SCL	284
9.1 Introduction to programming with SCL	284
9.1.1 Programming with SCL in general	284
9.1.2 SCL statements and operators	286
9.2 Programming binary logic operations with SCL	288
9.2.1 Scanning for signal states “1” and “0”	288
9.2.2 Taking account of the sensor type for SCL	289
9.2.3 AND function	291
9.2.4 OR function	291
9.2.5 Exclusive OR function	292
9.2.6 Combined binary logic operations	292
9.2.7 Negating the result of logic operation	293
9.3 Programming memory functions with SCL	294
9.3.1 Value assignment of a binary tag	294
9.3.2 Setting and resetting	294
9.3.3 Edge evaluation	295
9.4 Programming timer and counter functions with SCL	296
9.4.1 IEC timer functions	296
9.4.2 IEC counter functions	297
9.5 Programming digital functions with SCL	298
9.5.1 Transfer function, value assignment of a digital tag	298
9.5.2 Conversion functions	299
9.5.3 Comparison functions	301
9.5.4 Arithmetic functions	301
9.5.5 Mathematical functions	303
9.5.6 Word logic operations	303
9.5.7 Shift functions	304
9.6 Controlling the program flow with SCL	305
9.6.1 Working with the ENO tag	305
9.6.2 EN/ENO mechanism with SCL	306
9.6.3 Control statements	307
9.6.4 Block functions	316
9.7 Working with source files	319

9.7.1 General procedure	319
9.7.2 Programming a logic block in the source file	321
9.7.3 Programming a data block in the source file	325
9.7.4 Programming a PLC data type in the source file	327
10 Basic functions	328
10.1 Binary logic operations	328
10.1.1 Introduction	328
10.1.2 Scanning for signal states “1” and “0”, result of the scan	329
10.1.3 Negating the result of the logic operation, NOT contact	329
10.1.4 Testing floating-point tag, OK contact, OK box	330
10.1.5 AND function, series connection	331
10.1.6 OR function, parallel connection	332
10.1.7 Exclusive OR function, non-equivalence function	333
10.2 Memory functions	334
10.2.1 Introduction	334
10.2.2 Simple and negated coil, assignment	334
10.2.3 Single set and reset	335
10.2.4 Multiple setting and resetting	336
10.2.5 Dominant setting and resetting, memory boxes	337
10.3 Edge evaluation	338
10.3.1 Functional principle of an edge evaluation	338
10.3.2 Edge evaluation of the result of the logic operation	340
10.3.3 Edge evaluation of a binary tag	341
10.3.4 Edge evaluation with pulse output	342
10.4 Time functions	344
10.4.1 Introduction	344
10.4.2 Pulse generation TP	346
10.4.3 On-delay TON	347
10.4.4 OFF delay TOF	347
10.4.5 Accumulating ON delay TONR	348
10.5 Counter functions	349
10.5.1 Introduction	349
10.5.2 Up counter CTU	351
10.5.3 Down counter CTD	352
10.5.4 Up-down counter CTUD	353
11 Digital functions	355
11.1 Transfer functions	356
11.1.1 Introduction	356
11.1.2 Copy tag, MOVE box for LAD and FBD	356
11.1.3 Copy string, S_MOVE box for LAD and FBD	357
11.1.4 Value assignments with SCL	358
11.1.5 Copy data area (MOVE_BLK, UMOVE_BLK)	360
11.1.6 Filling the data area (FILL_BLK, UFILL_BLK)	361
11.1.7 Read and write the load memory (READ_DBL, WRIT_DBL)	362
11.1.8 Swap bytes (SWAP)	363
11.2 Comparison functions	364

11.2.1 Overview	364
11.2.2 Comparison of two tag values	364
11.2.3 Range comparison	365
11.3 Arithmetic functions for numerical values	366
11.3.1 Introduction	366
11.3.2 Addition ADD	367
11.3.3 Subtraction SUB	367
11.3.4 Multiplication MUL	367
11.3.5 Division DIV	367
11.3.6 Division with remainder as result MOD	368
11.3.7 Generation of absolute value ABS	368
11.3.8 Negation NEG	369
11.3.9 Decrement DEC, increment INC	369
11.4 Arithmetic functions for time values	369
11.4.1 Introduction	369
11.4.2 Addition T_ADD	371
11.4.3 Subtraction T_SUB	371
11.4.4 Difference T_DIFF	371
11.4.5 Combine T_COMBINE	371
11.5 Mathematical functions	372
11.5.1 Introduction	372
11.5.2 Trigonometric functions SIN, COS, TAN	373
11.5.3 Arc functions ASIN, ACOS, ATAN	373
11.5.4 Formation of square SQR	374
11.5.5 Extraction of square root SQRT	374
11.5.6 Exponentiate to base e EXP	374
11.5.7 Calculation of Napierian logarithm LN	374
11.5.8 Extracting decimal places FRAC	375
11.5.9 Exponentiation to any base EXPT	375
11.6 Conversion functions (Conversion of data type)	376
11.6.1 Introduction	376
11.6.2 Conversion function CONV	377
11.6.3 Conversion functions for floating-point numbers	378
11.6.4 Conversion functions SCALE_X and NORM_X	381
11.6.5 Conversion function T_CONV	383
11.6.6 Conversion function S_CONV	383
11.6.7 Conversion functions STRG_VAL and VAL_STRG	385
11.6.8 Conversion functions STRG_TO_CHARS and CHARS_TO_STRG	387
11.6.9 Conversion functions ATH and HTA	389
11.7 Shift functions	389
11.7.1 Introduction	389
11.7.2 Shift to right (SHR)	389
11.7.3 Shift to left (SHL)	391
11.7.4 Rotate to right (ROR)	391
11.7.5 Rotate to left (ROL)	392
11.8 Logic functions	392
11.8.1 Introduction	392
11.8.2 Word logic operations (AND, OR, XOR)	392
11.8.3 Invert (INV)	394

11.8.4 Coding functions DECO and ENCO	394
11.8.5 Selection functions SEL, MUX, and DEMUX	395
11.8.6 Minimum selection MIN, Maximum selection MAX	397
11.8.7 Limiter LIMIT	398
11.9 Processing of strings (Data type STRING)	398
11.9.1 Output length of a string LEN	399
11.9.2 Combine strings CONCAT	400
11.9.3 Output left part of string LEFT	400
11.9.4 Output right part of string RIGHT	401
11.9.5 Output middle part of string MID	401
11.9.6 Delete part of a string DELETE	401
11.9.7 Insert string INSERT	402
11.9.8 Replace part of string REPLACE	403
11.9.9 Find part of string FIND	403
11.10 Calculating with the CALCULATE box in LAD and FBD	404
12 Program flow control	406
12.1 Jump functions	406
12.1.1 Overview	406
12.1.2 Absolute jump	407
12.1.3 Conditional jump	408
12.1.4 Jump list JMP_LIST	409
12.1.5 Jump distributor SWITCH	410
12.2 Block end function	412
12.3 Calling of code blocks	413
12.3.1 Introduction	413
12.3.2 Calling a function FC	413
12.3.3 Calling a function block (FB)	415
12.4 EN/ENO mechanism	417
12.4.1 EN/ENO mechanism with LAD and FBD	418
12.4.2 EN/ENO mechanism with SCL	418
12.4.3 EN/ENO for user blocks	419
13 Online operation, diagnostics and debugging	420
13.1 Connecting a programming device to the PLC station	421
13.1.1 IP addresses of the programming device	421
13.1.2 Connecting the programming device to the PLC station	422
13.1.3 Assigning an IP address to the CPU module	424
13.1.4 Switching on the online mode	424
13.2 Transferring project data	425
13.2.1 Loading project data for the first time	425
13.2.2 Delta downloading of project data	427
13.2.3 Error message following downloading	428
13.2.4 Working with the memory card	428
13.2.5 Processing blocks offline/online	431
13.2.6 Comparing blocks offline/online	432
13.2.7 Editing online project without offline project	433
13.2.8 Uploading project data from the CPU	434

13.3 Hardware diagnostics	436
13.3.1 Status displays on the modules	436
13.3.2 Diagnostics information	437
13.3.3 Diagnostics buffer	437
13.3.4 Diagnostics functions	439
13.3.5 Online tools	439
13.3.6 Further diagnostics information via the programming device	440
13.4 Testing the user program	441
13.4.1 Introduction to testing with program status	441
13.4.2 Program status with LAD and FBD	442
13.4.3 Program status in SCL	444
13.4.4 Monitoring with the PLC tag table	445
13.4.5 Monitoring of data tags	446
13.4.6 Testing with watch tables	447
13.4.7 Monitoring tags using watch tables	449
13.4.8 Modifying tags using watch tables	450
13.4.9 Enable peripheral outputs and “Modify now”	451
13.4.10 Forcing tags	452
14 Distributed I/O	455
14.1 Introduction, overview	455
14.2 PROFINET IO	456
14.2.1 PROFINET IO components	456
14.2.2 Addresses with PROFINET IO	457
14.2.3 Configuring PROFINET IO	459
14.2.4 Real-time communication with PROFINET IO	461
14.3 PROFIBUS DP	462
14.3.1 PROFIBUS DP components	462
14.3.2 Addresses with PROFIBUS DP	465
14.3.3 Configuring PROFIBUS DP	467
14.3.4 System functions for PROFINET IO and PROFIBUS DP	470
14.4 Actuator/sensor interface	473
14.4.1 Components of actuator/sensor interface	473
14.4.2 Configuring an AS-i master CM 1243-2	475
14.4.3 Configuring an AS-Interface	476
14.4.4 Interface to user program	477
14.5 Communication via Modbus	477
14.5.1 Modbus RTU	477
14.5.2 Modbus TCP	480
15 Communication	482
15.1 Overview	482
15.2 Open user communication	484
15.2.1 Basics	484
15.2.2 Open user communication with TCP and ISO-on-TCP	485
15.2.3 Open user communication with the UDP protocol	487
15.2.4 Communication functions for open user communication	489
15.2.5 Configuring open user communication	493

15.2.6 Configuring a PN interface with T_CONFIG	495
15.3 S7 communication	496
15.3.1 Basics	496
15.3.2 Data structure for one-way data exchange	496
15.3.3 Communication functions for one-way data exchange	497
15.3.4 Configuring S7 communication	498
15.4 Point-to-point communication	499
15.4.1 Introduction to point-to-point communication	499
15.4.2 Configuring the CM 1241 communication module	500
15.4.3 Point-to-point communication functions	501
15.4.4 USS protocol for drives	504
16 Visualization	507
16.1 Introduction to visualization	507
16.1.1 Overview of HMI Panels in STEP 7 Basic	508
16.1.2 Creating a project with an HMI station	510
16.1.3 Cross-references for HMI objects	512
16.2 Creating HMI tags and area pointers	513
16.2.1 Introduction to HMI tags	513
16.2.2 Creating an HMI tag	514
16.2.3 Creating an area pointer	515
16.3 Configuring process screens	517
16.3.1 Introduction to configuring process screens	517
16.3.2 Working window for process screens	518
16.3.3 Working with screen layers	519
16.3.4 Working with templates	519
16.3.5 Working with function keys	520
16.3.6 Creating a new screen	521
16.3.7 Configuring a screen change	522
16.3.8 Working with objects in process screens	522
16.3.9 Changing screen objects during runtime	524
16.3.10 Basic objects for screen configuration	524
16.4 HMI functions	525
16.4.1 Input and display of process values	525
16.4.2 Working with alarms	528
16.4.3 Working with recipes	535
16.4.4 Working with the user administration	539
16.5 Completing HMI configuration	542
16.5.1 Compiling the HMI configuration (Consistency test)	542
16.5.2 Simulation of HMI configuration	542
16.5.3 Downloading configuration to the HMI station	543
16.5.4 Maintenance of the HMI station	546
17 Appendix	548
17.1 Integral and technological functions	548
17.1.1 High-speed counter (HSC)	548
17.1.2 Pulse generator	554
17.1.3 Technology objects for motion control	557

17.1.4 Technology objects for PID control 561

17.2 Telephone network connections with TeleService 564

17.3 Telecontrol with CP 1242-7 565

17.4 Web server 567

17.4.1 Enable web server 567

17.4.2 Reading out web information 567

17.4.3 Standard web pages 567

17.5 Data logging 569

17.5.1 Introduction 569

17.5.2 Using data logging 569

17.5.3 Functions for data logging 570

Index 572

1 Introduction

1.1 Overview of the S7-1200 automation system

The SIMATIC S7-1200 automation system consists of the four controllers S7-1211C, S7-1212C, S7-1214C, and S7-1215C, which can exchange data with each other, with SIMATIC HMI Basic Panels, or with other programmable controllers over SIMATIC NET. STEP 7 (TIA Portal) is used to configure and program the devices (Fig. 1.1).

The **SIMATIC S7-1200 controllers** are programmable logic controllers (PLC) and constitute the basis of the automation system. Four different controllers with graded performances cover the low-end range of industrial controls.

SIMATIC HMI refers to the Human Machine Interface for operator control and monitoring. The Basic Panels are designed such that they interact optimally with SIMATIC S7-1200. The devices are available with display dimensions of 3.8, 5.7, 10.4 and 15 inches, and are operated using the touch screen. Except for the 15-inch device, they have additional function keys.

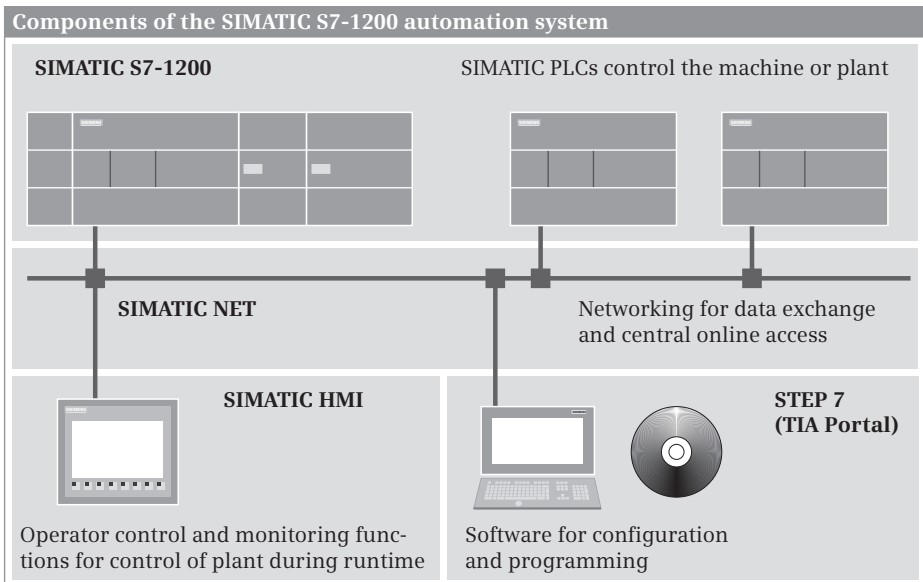


Fig. 1.1 Components of the SIMATIC S7-1200 automation system

SIMATIC NET links all SIMATIC stations, and allows trouble-free data exchange. SIMATIC S7-1200 with PROFINET interface uses the Industrial Ethernet network to exchange data with other PLC stations, HMI stations, and programming devices. Communication modules expand the communication capabilities to other networks such as PROFIBUS DP, AS-Interface, or point-to-point coupling based on RS232 or RS485.

The **STEP 7** programming software provides the nesting function for Totally Integrated Automation (TIA), the automation system with uniform configuration and programming, data management, and data transfer. STEP 7 is used to configure and parameterize the SIMATIC components, and STEP 7 is also used to generate and debug the user program. The *TIA Portal* is the central user interface for management of the tools and automation data. STEP 7 in the TIA Portal is available in the versions STEP 7 Professional and STEP 7 Basic. Both versions can be used to configure and program an S7-1200 station. This book describes the use of STEP 7 Basic.

1.1.1 SIMATIC S7-1200

SIMATIC S7-1200 is the modular microsystem for the lower and medium performance range. The central processing unit (**CPU**) contains the operating system and the user program. The user program is located in the load memory and is power failure-proof. The parts of the user program relevant to execution are processed in a work memory with fast access. Tags whose values are to be retained in the event of a power failure or when switching off/on are stored in the retentive memory (Fig. 1.2).

The user program can be transferred to the CPU using a plug-in memory card (**MC**) – as an alternative to transfer via an online connection to the programming device. The memory card can also be used as an external load memory or for updating the firmware.

The connections to the plant or process are made by **onboard inputs and outputs**, their number being determined by the CPU version. The onboard inputs and outputs are designed especially for operation of the integral high-speed counters (HSC). The operating system additionally includes pulse generators with a pulse-width modulated output and also the technology objects *Axis* for controlling stepper motors and servo motors with pulse interface and *PID Compact*, a PID controller with optimized self-tuning.

A signal board (SB) can be used to expand the onboard inputs and outputs. The communication board (CB) creates a point-to-point connection for the CPU and the battery board (BB) increases the power reserve of the integrated hardware clock to about one year.

If further inputs and outputs are required, signal modules (**SM**) can be plugged onto the CPU depending on its version. These are available for digital and analog signals.

The **PROFINET interface** connects the CPU to the Industrial Ethernet subnet. The programming device is connected to this interface if, for example, the user pro-

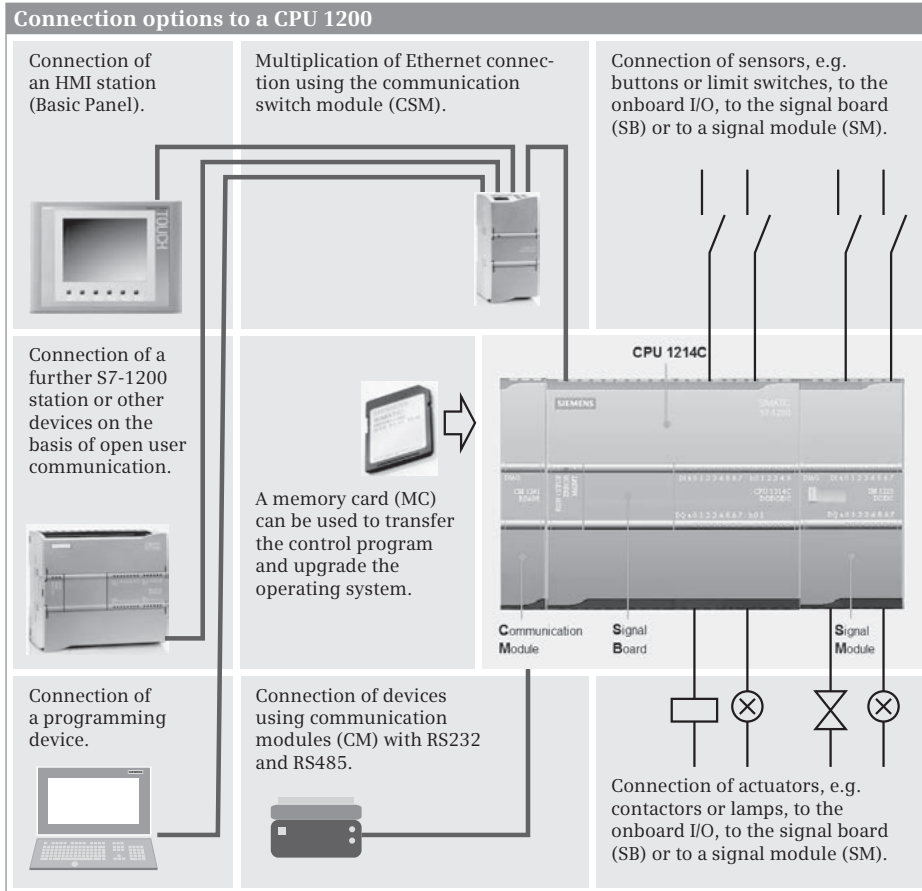


Fig. 1.2 Connection options to a PLC station with CPU 1200

gram is to be transferred online to the CPU and tested on the machine. Data is exchanged with HMI stations and other automation devices via this interface.

If the CPU is only connected to one device over Ethernet, a standard or crossover cable can be used. If more than two devices that only have a PROFINET interface are networked, the connecting cables must be routed via a multiplier, e.g. the **communication switch module (CSM)**. A CPU 1215 has two ports connected with a switch so that they can be networked with the next programmable controller without an interposed connection multiplier.

Communication modules (**CM**) permit the operation on further bus systems such as PROFIBUS DP. Here, an S7-1200 station in a DP master system can be both DP master and DP slave. An S7-1200 station can be the AS-Interface master on AS-Interface and can control up to 62 AS-Interface field devices. The communication module for the point-to-point connection is available with RS232 or RS485 interface, to which, for example, a barcode or RFID reader can be connected.

1.1.2 Overview of STEP 7 Basic

STEP 7 is the central automation tool for SIMATIC. STEP 7 requires authorization (licensing), and is executed on the current Microsoft Windows operating systems. STEP 7 Basic can be used to configure the S7-1200 controllers and – with WinCC Basic – the Basic Panels. Configuration 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 an (HMI) 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 assign them 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 Basic Panels.
- ▷ 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

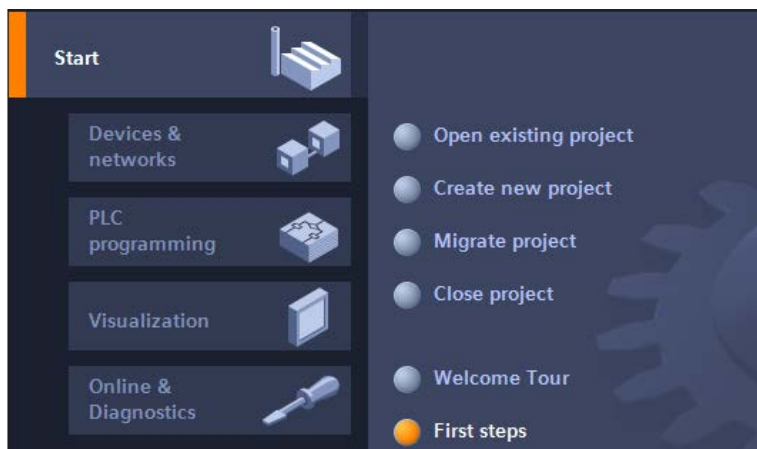


Fig. 1.3 Tools in the Start portal of STEP 7 Basic

window – the inspector window – displays the properties of the selected module, and the task window provides support by means of the hardware catalog with the available modules. The Network view shows the networking between the devices and permits the configuration of communication connections.

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 catalog of statements with the available program elements and functions. The same applies to the processing of PLC tags, to the online program test using watch tables, or to configuration of an HMI device.

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.

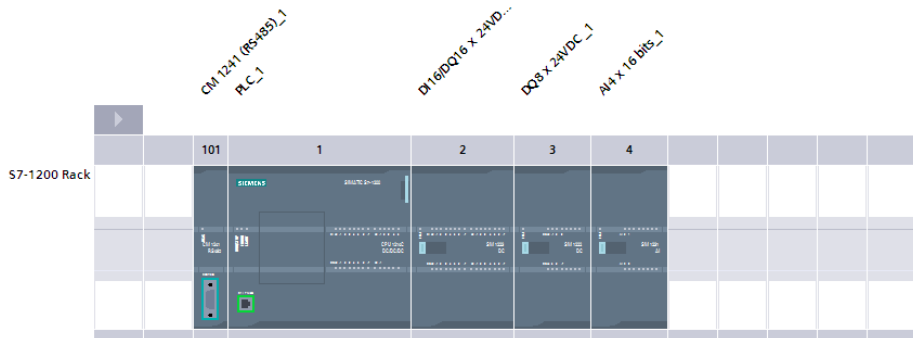


Fig. 1.4 Example of working area of device configuration

1.1.3 Three programming languages

You can select between three programming languages for the user program: ladder logic (LAD), function block diagram (FBD), and structured control language (SCL). The user program can be structured into individual parts known as “blocks”. The programming language is a property of a block, which means you can use the programming language that is best suited to resolve the block function for every block in the user program.

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 (Fig. 1.5). A current path is terminated by a coil. Complex functions are represented by boxes which you handle like contacts or coils. Examples of boxes are mathematical functions or functions for processing strings.

Using the **function block diagram**, you program the control task based on electronic circuitry systems. Binary operations are implemented by linking AND and OR

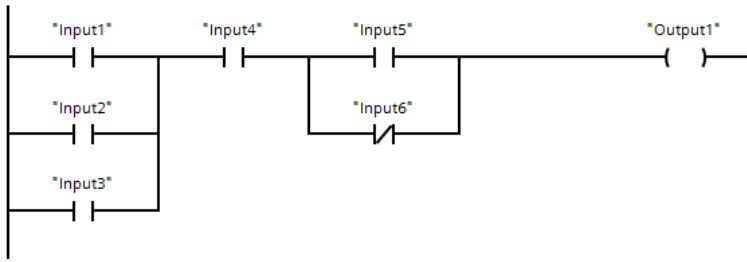


Fig. 1.5 Example of binary operations in ladder logic representation

functions and terminated by simple boxes (Fig. 1.6). Complex boxes are used to handle the operations on digital tags, for example with mathematical functions or functions for strings.

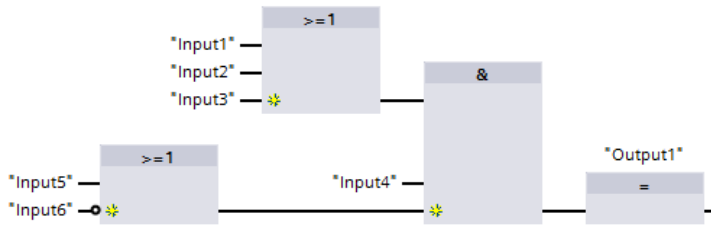


Fig. 1.6 Example of binary operations in function block diagram representation

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. 1.7).

```
18  (*****)  
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. 1.7 Example of SCL statements

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 that you program. Settings and initialization operations for the user program can be present here.

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. 1.8).

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. 1.9).

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; they are the tag memory of the function block. Global data blocks contain data which is not assigned to any block.

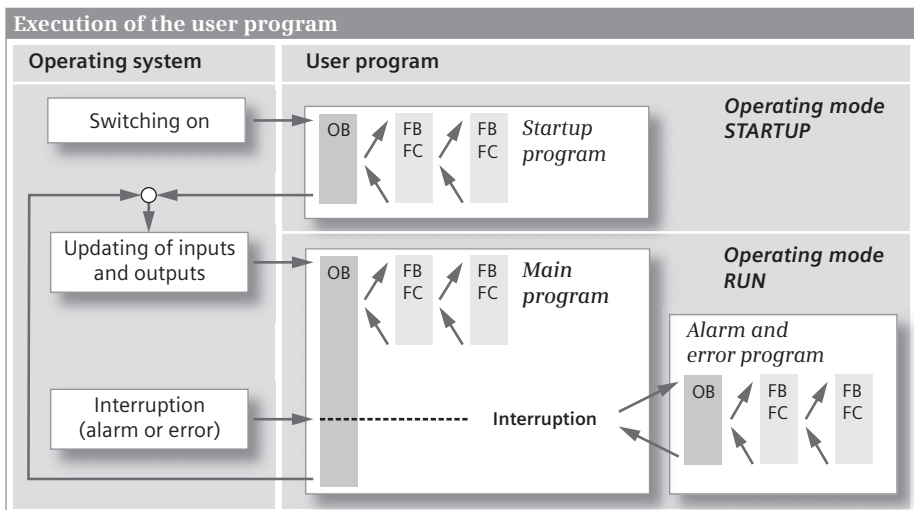


Fig. 1.8 Execution of the user program

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.

Cyclic program execution is a feature of programmable controllers. The user program is also executed if no actions are requested “from outside”, such as 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 logic controller has characteristics like those of 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. You can assign organization blocks to certain events, and these blocks are then processed in such a case. Once the interrupt program has been executed, the control processor continues execution of the main program from the point of interruption.

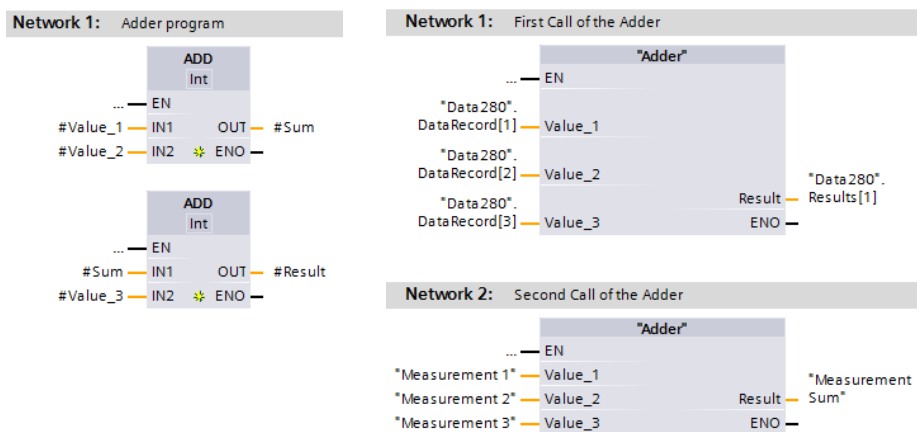


Fig. 1.9 Example of two block calls with different tags in each case

1.1.5 Data management in the SIMATIC automation system

The automation data is present in various memory locations in the automation system. Initially there is the programming device, referred to generally as the generation or engineering system. All automation data of a STEP 7 project is saved on its hard disk. Configuration and programming of the project data with STEP 7 is carried out in the main memory of the programming device (Fig. 1.10).

The automation data on the hard disk is also referred to as the *offline project data*. Once STEP 7 has appropriately compiled the automation data, this can be down-loaded to a programmable controller. The data downloaded into the user memory of the CPU module are known as the *online project data*.

The user memory on the CPU is divided into three components: The *load memory* contains the complete user program including the configuration data, the *work memory* contains the executable user program with the current control data, and the *retentive memory* contains the tags whose current values are saved power-failure-proof.

The memory card as a *transfer card* can transfer the user program to the CPU memory, or as a *program card* expand the CPU's internal load memory. When used as a program card, the memory card remains inserted in the CPU during runtime.

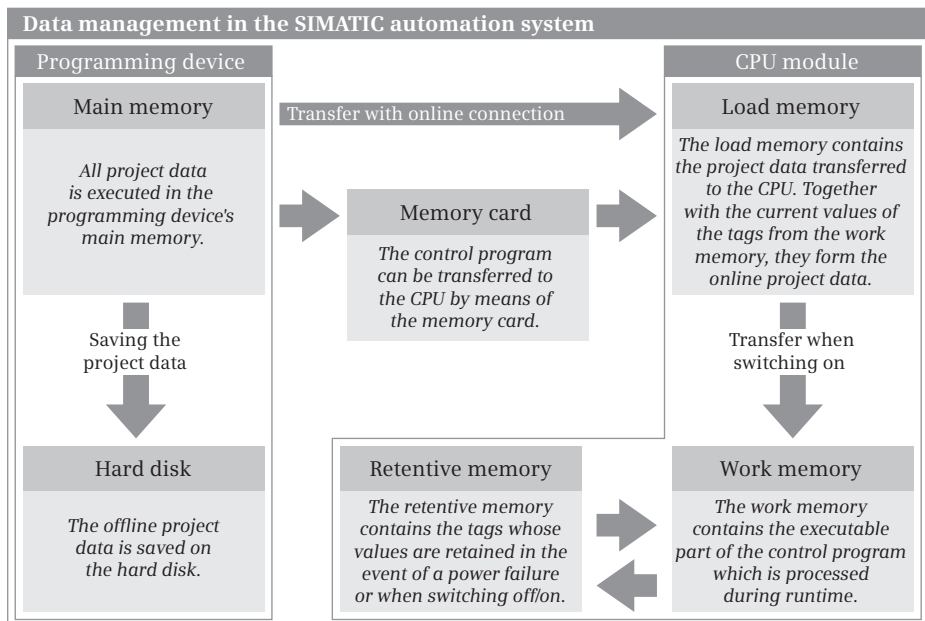


Fig. 1.10 Data management in the SIMATIC automation system